JPEG-Inspired Encoding for Deep Learning

by

Ahmed Hussein Salamah

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Ahmed Hussein Salamah 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Zixiang Xiong Professor, Dept. of ECE, Texas A&M University
Supervisor(s):	En-Hui Yang Professor, Dept. of ECE, University of Waterloo
Internal Member:	Zhou Wang Professor, Dept. of ECE, University of Waterloo
	Otman Basir Professor, Dept. of ECE, University of Waterloo
Internal-External Member:	Charles Clarke Professor, School of Comp. Science, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

JPEG is the dominant standard for storing and transmitting digital images, while Deep Neural Networks (DNNs) have become the preeminent method for automated image understanding. This dissertation investigates how these two ubiquitous technologies can be synergistically integrated to enhance the performance of DNNs. JPEG was originally engineered for the Human Visual System (HVS), and its default parameters are not optimized for DNNs, which process visual information differently. This suboptimality, stemming from JPEG's default implementation, is not a fundamental limitation but rather an opportunity to adapt its core components—especially the non-linear quantization stage—for DNNs. This research addresses this suboptimality by first optimizing the trade-off between compression rate and classification accuracy, and second, by introducing a learnable, end-to-end differentiable JPEG layer whose quantization parameters are jointly trained with the underlying DNN. This dissertation demonstrates that this principle of a learnable, JPEG-inspired transformation extends beyond compression, offering a novel way to address challenges in related domains such as knowledge distillation (KD), where large 'teacher' models often overfit the training set. This overfitting causes them to generate overconfident, near one-hot probability vectors that serve as poor supervisory signals for the student model, suggesting the need for novel approaches to information transfer. This dissertation addresses these issues by systematically revisiting the relationship between JPEG encoding and deep learning. It charts a logical progression from adapting JPEG externally for DNNs, to integrating it internally as a learnable network component, and finally to repurposing its core principles to amplify knowledge transfer.

This progressive framework is methodically developed and empirically substantiated through three interconnected contributions:

- **Optimizing Compression for DNNs**. To improve the interaction between standard JPEG and pre-trained DNNs, this work first reframes compression from a human-centric "rate-distortion" problem to a DNN-centric "rate-accuracy" one. This is achieved by introducing the Sensitivity Weighted Error (SWE), a novel distortion measure derived from a DNN's loss sensitivity to frequency-domain perturbations, where higher sensitivity in a frequency band indicates its greater importance for the DNN's decision-making. The SWE guides the OptS algorithm to generate model-specific JPEG quantization tables. This approach produces fully compliant JPEGs optimized for DNN consumption, demonstrably improving the rate-accuracy trade-off by increasing accuracy up to 2.12% at the same rate, or enabling rate reductions up to 67.84% with no loss of model accuracy.
- **Integrating a Differentiable JPEG Layer into the DNN Architecture.** Building on this, the next contribution integrates the codec into the network architecture itself via the JPEG-Inspired Deep Learning (JPEG-DL) framework, which introduces a novel, end-to-end differentiable JPEG layer. By replacing JPEG's standard hard quantization with a

differentiable alternative, this layer's parameters are jointly optimized with the network's weights. This transforms the JPEG pipeline from a static pre-processor into a dynamic, learnable component, significantly improving model accuracy—by an average of 7% on fine-grained classification tasks with only 128 additional trainable parameters—and enhancing robustness against adversarial attacks.

- **Amplifying Knowledge Transfer via JPEG-Inspired Perturbation.**
  Finally, the differentiable layer is repurposed to address the "overconfident teacher" problem in KD by perturbing teacher inputs to force softer, more informative predictions. Crucially, this method requires no retraining or modification of the fixed teacher model, ensuring its practical utility with proprietary or deployed networks. Our investigation begins with Coded Knowledge Distillation (CKD), a practical heuristic that uses adaptive JPEG compression to perturb teacher inputs and soften their overconfident predictions. While effective, this approach prompted a search for a more principled theoretical foundation. This led to Generalized Coded Knowledge Distillation (GCKD), a framework that establishes the maximization of the teacher's Conditional Mutual Information (CMI) as the core objective. However, directly optimizing for CMI on a per-input basis is computationally prohibitive. This efficiency challenge is resolved in the culminating synthesis, Differentiable JPEG-based Input Perturbation (DJIP). DJIP operationalizes the GCKD theory by deploying the trainable differentiable JPEG layer as a fast, learnable, and amortized operator. Instead of performing a slow, per-input optimization search, the layer is trained once to automatically generate CMI-maximizing perturbations, making the process highly efficient. This approach demonstrably generates richer supervisory signals, boosting student model accuracy by up to 4.11%.

In conclusion, this dissertation demonstrates that the relationship between JPEG and DNNs can be systematically revisited to create a powerful synergy. By progressing from adaptation to integration and synthesis, this work transforms the suboptimal default interaction of JPEG and DNNs into a versatile architectural tool. The research delivers a suite of methods that not only improve the performance of DNNs on compressed images but also offer a theoretically-grounded solution to a key challenge in knowledge distillation. By demonstrating that legacy codecs can be repurposed to enhance model accuracy, efficiency, and knowledge transfer, this work thus reframes the role of classical codecs, proposing JPEG-inspired encoding as a principled foundation for the integration of classical compression and deep learning.

Acknowledgements

First and foremost, I wish to thank professor *En-Hui Yang*, who guided and supported me throughout my proposal preparation. Throughout my Ph.D. program, professor Yang assisted me with research, writing, and presentation skills. Besides developing my logical and critical thinking abilities, Professor Yang provided constant feedback regarding the quality of my research. The support I received from Professor Yang was always of great importance to my professional success.

I am honored that this dissertation has been examined by Professor *Zhou Wang*, Professor *Otman Basir*, and Professor *Charles Clarke*, and Professor *Zixiang Xiong*. I owe respect and thanks to them for their time and insight

I do not also forget the support of my lab-mates in Multimedia Lab, particularly, *Kaixiang Zheng*, *Shayan Mohajer Hamidi*, and *Linfeng Ye*, who are the invaluable resource of my improvement and happiness.

This dissertation is incredibly dedicated to the beloved people who have meant so much to me and who continue to do so. Though they are not around anymore, I still live my life by their memories. First and foremost, I would like to thank my paternal grandfather and grandmother, Mr. and Miss Noweir. Both always taught me the value of hard work and whose love for me knew no bounds.

Finally, an honourable mention goes to my family, starting with my beloved wife and best support system in the world, Rohaima Salama, for her continuous support and for being my driving force in hunting all my goals and achievements. Then to my mother and father, Dr. Nagwa and Dr. Hussein: My entire life is all about both of you. Seeing you or even remembering you is enough to get through any challenges. My mission, my life, is just to make you both happy. You are in my heart, and I will never let you down. A sign of gratitude goes to my sisters, Dalia and Sarah, for always being there for me. I am also deeply grateful to my best friend, Mahmoud Al Saad, and to my incredible in-laws for their continuous support and encouragement. Without the help of the people above, I would have faced many difficulties in this journey.

Any acknowledgment ends up with an incomplete list inevitably, but I still wish to thank those professors, the instructors of classes I have attended, and all my other friends.

Dedication

To my beloved wife, Rohaima, and my cherished daughter, Lina—your unwavering love and support have been my greatest source of strength and inspiration throughout this journey.

Table of Contents

Examining Committee Membership ii

Author's Declaration iii

Abstract iv

Acknowledgements vi

Dedication vii

List of Figures xiv

List of Tables xxi

1 Introduction 1

1.1 Motivation and Thesis Statement 1

1.2 Thesis Roadmap: From Adaptation to Synthesis 3

1.2.1 Part I: Revisiting JPEG's Foundations for Machine Vision 3

1.2.2 Part II: Integrating the Encoding Pipeline into the DNN 4

1.2.3 Part III: Re-Application to Knowledge Transfer: From Heuristic to Theory and Synthesis 4

1.3 Thesis Contributions 5

1.4 Thesis Organization 6

2 Background **8**

2.1 The JPEG Image Compression Standard . . . 8

2.2 Rate-Distortion Optimization for Human Vision . . . 12

2.2.1 Soft-Decision Quantization (SDQ) . . . 12

2.2.2 Optimal Distortion Quantization Table Design (OptD) . . . 13

2.3 JPEG Compression for Machine Vision . . . 14

2.4 Enhancing DNN Performance with Adaptive Compression . . . 15

2.5 Model Compression and Teacher Enhancement . . . 16

3 JPEG Compliant Compression for DNN Vision **18**

3.1 Introduction . . . 19

3.2 Background and Related Work . . . 20

3.2.1 Soft Decision Quantization . . . 20

3.2.2 DNN-oriented Compression . . . 21

3.3 Problem Formulation . . . 22

3.4 Distortion Measure for DNN Vision . . . 23

3.4.1 DNN's Sensitivity to DCT Perturbations . . . 23

3.4.2 Offline Estimation of the Sensitivity . . . 25

3.4.3 Similarity of Sensitivity Among Different DNN Models . . . 28

3.4.4 Sensitivity Weighted Error (SWE) . . . 31

3.5 Quantizer Design . . . 31

3.5.1 Soft Decision Quantization Revisited for DOC . . . 31

3.5.2 OptS: Designing Optimal Quantization Tables for DOC . . . 32

3.5.3 Analysis for Quantization Tables Designed by OptS . . . 37

3.6 Experimental Results . . . 39

3.6.1 Experimental Setup . . . 41

3.6.2 R-A Performance . . . 41

3.7 Chapter Summary and Future Work . . . 50

4 JPEG Inspired Deep Learning **51**

4.1 Introduction . . . 51

4.2 Background and Related Work . . . 54

4.3 JPEG-DL: JPEG Inspired DL . . . 55

4.3.1 Problem Formulation . . . 55

4.3.2 Differentiable Soft Quantizer . . . 56

4.3.3 Overall Framework of JPEG-DL . . . 57

4.3.4 Partial Derivatives of Q_d . . . 59

4.4 Experiments . . . 61

4.4.1 Experimental Setup . . . 61

4.4.2 Experimental Results . . . 64

4.4.3 Comparison with JPEG-based Data Augmentation and Other Preprocessing Methods . . . 67

4.5 Analysis and Discussion . . . 69

4.6 Chapter Summary . . . 76

5 A Coded Knowledge Distillation Framework Based on Adaptive JPEG Encoding **78**

5.1 Introduction . . . 78

5.2 Related Work . . . 80

5.3 Preliminaries and Motivation . . . 82

5.3.1 Preliminaries . . . 82

5.3.2 Motivation . . . 83

5.4 *Coded* Knowledge Distillation . . . 84

5.4.1 Optimal Adaptive JPEG Selection . . . 85

5.4.2 Analyzing the Proposed Adaptive JPEG Selection . . . 86

5.5 Experment Results . . . 89

5.5.1 ImageNet Dataset . . . 90

5.5.2 CIFAR100 dataset . . . 91

5.5.3 Fine-grained Classification Task . . . 94
5.5.4 More Evaluation on ImageNet Subset . . . 96
5.5.5 Analyzing student's generalization error in CKD . . . 98
5.6 Chapter Summary and Outlook on Upcoming Challenges . . . 99

6 Generalized Coded Knowledge Distillation 100
6.1 Introduction . . . 100
6.2 Related Works . . . 103
6.3 Notation and preliminaries . . . 105
6.3.1 Notation . . . 105
6.3.2 Conditional mutual information of a model . . . 107
6.3.3 Empirical Approximation of CMI . . . 107
6.3.4 Coded Knowledge Distillation . . . 108
6.4 Generalized Coded Knowledge Distillation . . . 109
6.4.1 Overall Framework . . . 109
6.4.2 Problem Formation . . . 110
6.4.3 Extension of CMI to the Adversarial Context . . . 111
6.4.4 Alternating Optimization for CMI Maximization . . . 113
6.4.5 Alternating Optimization Procedure . . . 115
6.5 Experiments . . . 120
6.5.1 Experimental Setup . . . 120
6.5.2 Ablation on Perturbation Generation Strategy . . . 120
6.5.3 Results on CIFAR-100 . . . 121
6.5.4 Results on ImageNet . . . 121
6.5.5 Comparison with MCMI . . . 121
6.5.6 Comparison with CKD and TALD . . . 123
6.6 Analysis . . . 124
6.6.1 Convergence of the Alternating Optimization . . . 124
6.6.2 Geometric Analysis of the GCKD Output Space . . . 124
6.7 Chapter Summary . . . 127

7 Differentiable JPEG-based Input Perturbation for Knowledge Distillation Amplification 129

7.1 Introduction . . . 129
7.2 Related Work . . . 131
7.2.1 Student-Oriented Teacher Design . . . 131
7.2.2 Input Perturbation in Knowledge Distillation . . . 131
7.3 Methodology . . . 132
7.3.1 Overall Framework . . . 132
7.3.2 Objective Function . . . 132
7.3.3 An Alternating CMI Maximization Algorithm . . . 135
7.4 Experiments . . . 136
7.4.1 Implementation Details . . . 136
7.4.2 Experimetnal Results . . . 139
7.5 Analysis . . . 142
7.5.1 Ablation Study . . . 142
7.5.2 GCKD's Advantage Through PGD in Solving the Inner Minimization Problem . . . 143
7.5.3 Orthogonality Over MCMI . . . 145
7.5.4 Comparison with CKD and MCMI . . . 145
7.5.5 Comparison with CKD and TALD . . . 146
7.5.6 Visualize the Output Probability Space in Different Epochs . . . 147
7.5.7 Trained Quantization Table Visualization . . . 147
7.5.8 Visualize the Output Probability Space of DNNs . . . 148
7.5.9 DJIP with other input perturbation method . . . 149
7.5.10 Few Shot Classification . . . 151
7.5.11 Computational Efficiency and Convergence Behavior of DJIP . . . 151
7.5.12 Ablation Studies on DJIP Hyper-parameters . . . 154
7.6 Chapter Summary . . . 156

8 Conclusion and Future Work **157**

8.1 Summary . . . 157

8.2 Thesis Contributions . . . 158

8.3 Directions for Future Work . . . 159

8.4 Discussion . . . 161

8.4.1 Dataset-Dependent Frequency Sensitivity . . . 161

References **164**

APPENDICES **176**

A Derivation of the KKT Conditions for Optimal Distortion Allocation **177**

A.1 The Optimization Problem . . . 177

A.1.1 Objective Function . . . 177

A.1.2 Constraints . . . 178

A.2 The Lagrangian Formulation . . . 178

A.3 The KKT Conditions . . . 179

A.4 Applying the Conditions: The Derivation . . . 179

A.4.1 Stationarity . . . 179

A.4.2 Analysis via Complementary Slackness . . . 180

A.5 Synthesis of the Result . . . 181

B Offline Estimation of the Sensitivity **182**

B.1 Model-Dependent Frequency Sensitivity . . . 182

C Rate–Distortion Evidence for OptD **185**

List of Figures

2.1 A typical pipeline of image acquisition, encoding, storage/transmission, and decoding for both human and machine consumption. 9

2.2 The baseline JPEG compression pipeline, adapted from [135]. The quantization and Huffman tables are free parameters that can be optimized by the encoder. . . 9

2.3 Default JPEG quantization tables for luminance (left) and chrominance (right) channels, designed for the Human Visual System [91]. 11

3.1 The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, with sensitivity indices arranged in the default zigzag order. . . 26

3.2 The estimated sensitivity is shown for different Vision Transform (ViT) Models pre-trained models on the ImageNet dataset, with sensitivity indices arranged in the default zigzag order. 27

3.3 The estimated sensitivity is shown for different pre-trained models on Pascal VOC 2012 dataset, with sensitivity indices arranged in the default zigzag order. . 28

3.4 Sensitivities for MobileNetV2 with two distinct pre-trained weights are compared in (a), (c) and (e). Sensitivities for RegNetY-400MF with two distinct pre-trained weights are compared in (b), (d) and (f). Each channel's sensitivity is normalized by the sensitivity value of the DC frequency for better visualization. 29

3.5 The sensitivity of different models within the RegNets and ConvNeXts families, normalized by the sensitivity value of the DC frequency for better visualization. It is evident that DNN variants within the same architecture family exhibit a similar sensitivity pattern. 30

3.6 The figure shows the sensitivity of target DNNs with the quantization tables designed by OptS, together with default JPEG quantization tables for comparison. Note that it is $1/q_i$ shown in the figure instead of q_i. Also, all curves are normalized by their respective values of sensitivity and inverse q corresponding to DC for better visualization. The first column is for the luminance channel, while the second column is for the chrominance channel. We select $d = 1$ for OptS over all target DNNs. 40

3.7 Performance evaluation of R-A for four prominent models from different families with HDQ coding using quantization tables from OptS, OptD, and JPEG. It is worth mentioning that for MobileNetV2, MnasNet, RegNetX-400MF, and ConvNeXt-T, DeepN-JPEG yields 68.43%, 70.1%, 71.2%, and 78.4%, respectively, of accuracy with the rate of 7.9 bpp (its accuracy is low). Therefore, in order not to distort the curve display, these points are removed from the figures. . 42

3.8 R-A performance and CE Loss of HDQ coding applied to the Swin-T transformer model using quantization tables provided by OptS, OptD, and JPEG. 43

3.9 R-A performance comparison among Tr-OptS, OptS, and JPEG, where Tr-OptS is designed for RegNet-400MF (ConvNeXt-T, resp.) and fixed when evaluated for each of other DNNs within the family of RegNets (ConvNeXts, resp.), and OptS is designed specifically for each of other DNNs within the family of RegNets (ConvNeXts, resp.). 44

3.10 Evaluation of the R-A performance following the distortion-matching setup. Fig. 3.10a and 3.10b correspond to Setting 1; Fig. 3.10c and 3.10d correspond to Setting 2. . 45

3.11 R-A performance of SDQ coding with quantization tables provided by OptS, OptD, and JPEG (HDQ results are presented by transparent lines as a reference). 47

3.12 This figure displays the CE Loss for Fig. 3.11 to illustrate that both HDQ and SDQ Coding, when OptS is used, converge faster towards the CE loss of the raw image, indicated by a black dotted horizontal line, compared to other methods. . . 48

4.1 (a) The JPEG-DL framework consists of a JPEG layer followed by a standard DNN, where the standard forward/inverse processes of JPEG depicted in white boxes are fixed and not trained. The JPEG pipeline receives a conventionally pre-processed input image $x \in \mathbb{R}^{3\times W\times H}$ sampled from the underlying task's dataset. The JPEG layer, equipped with a differentiable soft quantizer, and the underlying DNN form a unified new DNN architecture, which are shown in blue indicating that they are trainable components. (b) As an example, we show $\boldsymbol{z}_{1,:,:}$, i.e., the DCT representation of the Y channel of an image, by a tensor consisting of B blocks of DCT coefficients. Each 8×8 block contains $M = 64$ DCT frequencies, ordered from low to high in a zigzag manner. Then, we show how $\boldsymbol{z}_{1,1,:}$ and $\boldsymbol{z}_{1,M,:}$ are quantized by $Q_d(\cdot\,; q = 1, \alpha = 10)$ and $Q_d(\cdot\,; q = 0.5, \alpha = 16)$, respectively. 53

4.2 Illustration of Q_u vs. Q_d with $\alpha = 1, 3, 5, 10$, where L and q are set to 3 and 1, respectively. 58

4.3 This Figure illustrates the partial derivatives of $Q_d(z)$ w.r.t. the scaling factor α, where $b = 3$ and $q = 1$. 60

4.4 Figures 4.4a and 4.4b illustrate the partial derivatives of $Q_d(z)$ w.r.t. the parameter q for cases where $\alpha = 8$ and q varies, and α varies and $q = 2$, respectively. For both figures, we set $b = 6$. 61

4.5 Demonstrating impact of $\hbar$ on gradient magnitude for various combinations of q and α. We fixed $\hbar$=2 and b=6. 62

4.6 Evaluate the adversarial robustness of JPEG-DL models in comparison to standard DNN on VGG13 and Res56 for CIFAR-100 against FGSM and PGD attacks. 70

4.7 The estimated sensitivity is shown for two pre-trained models on the CUB200 (first row) and three pre-trained models on the CIFAR-100 (second row), with sensitivity indices arranged in the default zigzag order. 71

4.8 Initial and final quantization tables for Res56 trained on CIFAR-100 and DenseNet-121 trained on CUB200, with frequency indices arranged in the default zigzag order. 72

4.9 Initial and final quantization tables for VGG13 trained on CIFAR-100 and ResNet18 trained on CUB200, with frequency indices arranged in the default zigzag order. . 73

4.10 Feature maps of size 56×56 are shown after the first dense block in DenseNet-121 for both JPEG-DL and baseline models Figs. in 4.10a and 4.10c, respectively, using an original input shown in Fig.4.10b. The JPEG-DL model highlights the foreground (bird) more distinctly, while the baseline model shows less contrast, contributing to its misclassification. 74

4.11 Feature maps of size 56×56 are shown after the first dense block in DenseNet-121 for both JPEG-DL and baseline models in Figs. 4.11a and 4.11c, respectively, using an original input shown in Fig. 4.11b. The JPEG-DL model highlights the foreground (flower) more distinctly, while the baseline model shows less contrast, contributing to its misclassification. 74

4.12 GradCAM++ visualization for baseline and JPEG-DL models on CUB200 using two examples, where the baseline model incorrectly classified them and the JPEG-DL model correctly classified them. 75

4.13 Partial visualization of CPMFs $P_\alpha(\cdot|z = 0.5)$ computed using (4.6), with $L = 1023$, $q = 1$, and α values of (a) 1, (b) 3, (c) 5, and (d) 10. The reconstruction levels between two red dashed lines represent $\mathcal{M}$. 75

5.1 The simplex of output probability vector for a classification task with three classes. Our goal is to alter the original samples with peaky output probabilities (the blue dots) in order to obtain samples whose output probabilities are more uniform (the red dots). 84

5.2 Illustration of CKD: (a) the CKD framework with an adaptive JPEG encoding based *coded* teacher; and (b) the inner-working of adaptive JPEG encoding mechanism. 87

5.3 The decision boundary plots for Resnet56 pre-trained on CIFAR100 dataset, when the inputs are the elements from the sets: (left) $\mathcal{J}_{\text{Img1}}$, and (right) $\mathcal{J}_{\text{Img2}}$. Green and Red dots show the correctly and wrongly classified samples, respectively. Note that the raw images Img1 and Img2 are correctly and wrongly classified, respectively. 88

5.4 Comparison between the *coded* teacher and TALD when they are applied on top of different underlying knowledge transfer methods. The teacher-student pairs are reported in the caption of each figure. 93

5.5 Validation accuracy curves along the number of epochs over CUB200 dataset with (WRN-64) as the teacher and (W10) and (W16) as students: (a) **C**KD vs KD; and (b) **C**AT vs AT. 96

6.1 A diagram illustrating the GCKD encoding Framework. The encoder takes inputs X and Y, and generates a perturbed instance $X' = F(X, Y)$, constrained by $d(X, X') \leq \varepsilon$. The encoded input X' is then passed to a teacher model f_{θ_t} to produce the final output $\hat{Y}'$. 109

6.2 A conceptual illustration of different strategies for generating perturbed inputs. In this diagram, q_i is used as a shorthand for the auxiliary 'dummy' distribution, $Q'(x|i, y)$, which is central to the double minimization problem formulated in Theorem 2. (a) **FGSM** is a single-step method that provides a coarse approximation of the optimal perturbation. (b) **PGD** iteratively refines the perturbation x' to better solve the inner minimization problem while the set of dummy distributions $\{q_i\}$ is held fixed. (c) **Amortized PGD** represents a more efficient variant where the dummy distribution q_i is updated after each perturbation of x'_i, a process that is faster but potentially less precise than completing the full PGD optimization for a fixed $\{q_i\}$. 119

6.3 (a) Comparison with MCMI. (b) Comparison with CKD and TALD. 123

6.4 Evolution of key metrics during the alternating optimization stage (Stage 1 of GCKD) for a ResNet-34 teacher on ImageNet. The plots track the teacher's Top-1 accuracy and NLL on the evolving perturbed dataset, alongside the CMI-related losses, over several iterations within a single epoch. As the algorithm iterates, the CMI-related losses decrease, indicating successful CMI maximization. This is achieved at the cost of a slight, controlled increase in NLL and a decrease in the teacher's own accuracy, demonstrating the trade-off at the heart of GCKD. The two subplots compare the effect of different PGD step sizes, showing the stability of the optimization process. 125

6.5 The output probability vectors of a trained GCKD teacher for a three-class classification task, visualized on a 2-simplex over different epochs. Three classes are randomly selected from the CIFAR-100 training set. As training progresses, the empirical CMI increases, and the output distributions (colored dots) and class centroids (cross markers) move from the vertices toward the center, indicating the teacher is producing softer, more informative predictions. Temperature scaling ($T = 1.0$) is applied for visualization. 126

6.6 t-SNE visualization of the output probability distributions from the CIFAR-100 training set using a VGG-13 teacher. (a) CE teacher: The clusters are tightly concentrated, corresponding to low-entropy, overconfident predictions. (b) GCKD teacher: The clusters are visibly more dispersed, providing a visual confirmation that GCKD successfully increases the teacher's CMI by generating softer output distributions. 127

7.1 The overall training framework of DJIP is shown, centered around a differentiable JPEG coded teacher module (blue), which integrates a differentiable JPEG layer (orange). The process involves two stages: differentiable JPEG layer training and student distillation, with trainable parameters highlighted in red. 130

7.2 (a), (b) Comparison with MCMI under different student architectures on CIFAR-100. (c), (d) Comparison with CKD and TALD under different student architectures on CIFAR-100. 146

7.3 The output probability vectors of the trained DJIP teacher for a three-class classification task are visualized on a 2-simplex over different epochs within a single training run. Three classes are randomly selected from the CIFAR-100 training set, with 100 samples per class. Temperature scaling with $T = 4.0$ is applied to the softmax outputs to enhance distributional smoothness. In the plots, cross markers denote the class centroids, and the CMI values are computed based solely on these three selected classes. 147

7.4 Trained quantization tables on CIFAR-100 models used in Table 7.1 and 7.2. . . . 148

7.5 Trained quantization tables on ImageNet models in Table 7.3. 149

7.6 t-SNE [118] visualization of features extracted from the CIFAR-100 training set with 500 samples per cluster in all 100 categories. 150

7.7 Convergence curves of the proposed alternating optimization on ImageNet. The training DJIP loss, training CMI value, and total training loss exhibit stable and monotonic trends, confirming the empirical convergence of our method. 153

8.1 The estimated sensitivity is shown for two pre-trained models on the CUB200, with sensitivity indices arranged in the default zigzag order. 162

8.2 The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, with sensitivity indices arranged in the default zigzag order. . . 163

B.1 The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, normalized by the sensitivity value of the DC frequency for better visualization. 183

B.2 The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, normalized by the sensitivity value of the DC frequency for better visualization.. 184

C.1 Rate–Distortion (PSNR vs. bpp) comparison between the default JPEG encoder and OptD with *joint distortion allocation*. Across the tested bitrate range, the OptD curve consistently lies above the JPEG curve, indicating higher PSNR at the same rate—or equivalently, lower rate for the same PSNR—thus demonstrating the superiority of the proposed allocation strategy. 186

List of Tables

3.1 Top-1 validation accuracies are shown for MobileNetV2 and RegNetY-400MF with different pre-trained weights. 28

3.2 Correlation between sensitivity and the inverse of the designed quantization table by OptS averaged over the ImageNet validation set, is compared to the correlation between sensitivity and the inverse of the default JPEG quantization table. . 39

3.3 R-A performance summary, derived from Fig. 3.11, enables a comparison between SDQ and HDQ coding methods. This assessment is conducted with the underlying quantization table design by OptS, OptD, and JPEG, considering both the same rate and accuracy levels. 49

3.4 Accuracy improvement and CE loss reduction are shown by using the compressed images obtained through HDQ coding with OptS algorithm, compared to the raw images. 49

3.5 Accuracy improvement demonstrated through HDQ and SDQ coding with OptS method compared to raw image. 50

4.1 Hyper-parameter setting on EfficientFormer-L1. 63

4.2 Top-1 validation accuracy (%) for Baseline and JPEG-DL on CIFAR-100. The Baseline results are from [114]. For JPEG-DL, we report the mean and standard deviation of experimental results over three runs. 64

4.3 Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs. 65

4.4 Top-1 accuracy (%) on CIFAR-100 and two fine-grained image classification tasks using EfficientFormer-L1. We report the mean and standard deviation of the experimental results over three runs with different random seeds. The best results are indicated in bold, and values in parentheses indicate relative accuracy gains over the baseline. 65

4.5 Top-1 validation accuracy (%) on ImageNet with different model architectures. . 65

4.6 Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs. 67

4.7 JPEG-DL is compared to JPEG-based augmentation (orange), non-learnable (yellow), and learnable preprocessing (blue) methods. JPEG-DL employs three different chrominance subsampling schemes shown in green color. Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs. The best results are indicated in bold, and values in parentheses indicate relative accuracy gains over the baseline. 68

4.8 Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs. 69

4.9 $\mathcal{Q}_d(z = 0.5; q = 1, \alpha)$ values resulting from the full-space CPMF and the masked CPMF at different α, alongside the absolute difference Δ between the full-space and masked cases. 73

4.10 Comparison of inference time between different inference settings measured for Resnet18 on ImageNet, following the experimental setup mentioned in Section 4.4. Evaluation is conducted on a single NVIDIA RTX A5000 GPU. 76

5.1 The average (and standard deviation) of (i) the entropy of p_x and $p_{\hat{x}}$, and (ii) the GT probability in p_x and $p_{\hat{x}}$ from both the *Coded* teacher and original teacher, respectively, over CIFAR100 dataset, where p_x and $p_{\hat{x}}$ are generated using $\tau = 1$. The pre-trained teacher models are also used in [14, 19, 47, 77, 84, 114, 145]. . . . 89

5.2 The average (and standard deviation) of (i) the entropy of p_x, and (ii) the GT probability in p_x from students trained by CKD with the *Coded* teacher and other knowledge transfer methods with the original teacher on the CIFAR-100 dataset, where p_x is generated using $\tau = 1$. 90

5.3 The student accuracy on ImageNet dataset given by different KD methods, where the best and second-best results are **bold** and underlined, respectively. The experiment setup is the same as in [114]. *As we followed [75] for our experimental setups, the accuracy for the Resnet18 model differs from that reported in [41]. . . 90

5.4 Test accuracy (%) of students trained by different KD methods on CIFAR100, where the best and second-best results are **bold** and underlined, respectively. The student accuracy for TALD and BSS is obtained using the settings specified in [84]. 91

5.5 Test accuracy (%) of students on CIFAR100 dataset before and after the *coded* teacher is applied on top of the underlying distillation methods when the teacher and student pair has the same architectural style, where each underlying distillation method, when coupled with the coded teacher, is indicated by the bold prefix **C** and referred to as the corresponding coded distillation method. The accuracy for the underlying distillation methods is reported from [19, 114], and the pre-trained teacher models are also used in [14, 19, 47, 77, 84, 114, 145]. 94

5.6 Test accuracy (%) of students on CIFAR100 dataset before and after the coded teacher is applied to the underlying distillation methods when the teacher and student pair has different architectural styles, where each underlying distillation method, when coupled with the coded teacher, is indicated by the bold prefix **C** and referred to as the corresponding coded distillation method.. The accuracy for the underlying distillation methods is obtained from [19, 114], while the pre-trained teacher models are also used in [14, 19, 47, 77, 114, 145]. 95

5.7 The performance of CKD and CAT against conventional KD and AT. 96

5.8 CKD performance on ImageNet-1K subset with and without data augmentation. . 97

5.9 Ablation study on the adaptive layer in CKD on ImageNet dataset subset. 98

5.10 Covariance ($\times 10^{-5}$) for both *coded* and original teacher on the CIFAR-100. . . . 99

6.1 Comparison of student test accuracy (%) using different adversarial attack strategies to generate the coded dataset for the GCKD teacher. Results are shown for VGG-13 as the teacher and two different students (VGG-8, MobileNetV2). 'Conventional' refers to a standard NLL teacher. 121

6.2 The test accuracy (%) of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the same/different architectures. The subscript denotes the improvement achieved by replacing the standard teacher with the GCKD teacher. We use bold numbers and asterisk (*) to denote the best results. GCKD delivers consistent accuracy gains across a wide variety of architectures and KD methods. 122

6.3 Top-1 and Top-5 test accuracy (%) of student models on the ImageNet validation set for six KD methods, demonstrating the performance gain of GCKD. For both teacher-student pairs (ResNet-34 $\rightarrow$ ResNet-18 and ResNet-50 $\rightarrow$ MobileNetV1), using the GCKD teacher consistently improves student accuracy over the standard NLL teacher baseline across all KD methods. RN and MN denote ResNet and MobileNet, respectively. 123

7.1 The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the same architecture. The subscript denotes the improvement achieved by replacing the CE teacher with the DJIP teacher. We use asterisk $(^*)$ to identify the results reproduced on our local machines. Note that the test accuracy $(\%)$ of the teacher here is over the perturbed image. 140

7.2 The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the different architectures. The subscript denotes the improvement achieved by replacing the CE teacher with the DJIP teacher. We use asterisk $(^*)$ to identify the results reproduced on our local machines. Note that the test accuracy $(\%)$ of the teacher here is over the perturbed image. . 141

7.3 The test accuracy $(\%)$ of student networks on ImageNet, with teacher-student pairs of the same and different architectures. 142

7.4 The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of different architectural models. The subscript denotes the improvement achieved by replacing the CE teacher with a DJIP teacher. 142

7.5 The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the same- and different-architecture. Noted that here, method JMCMI refers to applying the fix-centroid strategy to train the differentiable JPEG layer only. 143

7.6 Impact of adversarially coded datasets on student performance during knowledge distillation using DIST as an orthogonal method. The table compares conventional NLL-based teachers with GCKD and DJIP teachers across two student architectures (VGG-8 and MobileNetV2). Results show that incorporating stronger adversarial attacks, such as PGD, consistently yields the highest accuracy improvements over the baseline. 144

7.7 Test accuracy (%) of student networks on CIFAR-100 under different teacher-student configurations. The comparison highlights the performance gains achieved by DJIP and GCKD over the conventional CE baseline, demonstrating their effectiveness in enhancing knowledge distillation across diverse architectures. . . . 144

7.8 The test accuracy (%) of student networks on CIFAR-100 (averaged over three runs), with teacher- student pairs of the same- and different- architecture. The results obtained by the MCMI method are used as the baseline. This table demonstrates that our method provides additional orthogonality to MCMI, which is itself an orthogonal approach. Note that in this context, DJIP-MCMI refers to applying our proposed method using the fine-tuned MCMI teacher as the underlying model. 145

7.9 The test accuracy $(\%)$ of students on CIFAR-100 with autoencoder as input perturbation method. 151

7.10 Comparison of student accuracy (%) under few-shot distillation settings on CIFAR-100. The teacher is trained on the full dataset, while only a β percent subset of samples per class is used to train the student. 152

7.11 Comparison of DJIP distillation complexity measured with different teacher-student pairs on CIFAR-100 with vanilla KD method and TALD, following the configuration specified in Section 7.4.1. 153

7.12 Effect of λ on student test accuracy (%) and the CMI/CE losses of the DJIP teacher. The results are compared with those of a standard CE teacher in the ResNet-34 $\rightarrow$ ResNet-18 setting on ImageNet with vanilla KD. **Bold** numbers indicate the best performance. 154

7.13 Effect of the number of quantization rounds on student accuracy (%) and on the CMI and CE losses of the DJIP teacher. Results are reported for the ResNet-34 $\rightarrow$ ResNet-18 setting on ImageNet under vanilla KD. **Bold** numbers denote the best performance. 155

7.14 Effect of learning rate on student accuracy (%), along with the CMI and CE losses of the DJIP teacher. Results are reported for the ResNet-34 $\rightarrow$ ResNet-18 setting on ImageNet under vanilla KD. **Bold** indicates the best performance. . . . 155

7.15 Effect of centroid update interval on student accuracy (%) on CIFAR-100 (averaged over 3 runs). Results are reported for the ResNet-34 → ResNet-18 setting, with update intervals of 3, 5, 50, and 500 iterations. All configurations consistently outperform the standard CE teacher, with the best performance achieved at an interval of 3 and 5. 156

Chapter 1

Introduction

1.1 Motivation and Thesis Statement

The ubiquitous JPEG format serves as the primary medium for digital imagery, while Deep Neural Networks (DNNs) represent the cornerstone of modern computer vision, powering state-of-the-art models for tasks such as image classification [41,104], object detection [33], and semantic segmentation [70]. This dissertation is motivated by the question of how these two foundational technologies, each conceived in isolation, can be productively combined to advance machine perception systems. The JPEG standard was developed with the explicit goal of optimizing for the Human Visual System (HVS) [91], often by discarding high-frequency details. This design choice creates a performance gap when these images are consumed by DNNs, which operate on different perceptual principles. This work posits that this inherent mismatch is not an insurmountable obstacle but rather a unique opportunity. Specifically, we propose that JPEG's core mechanism—the frequency-domain quantization stage—can be repurposed as a powerful tool for machine perception. Accordingly, the research is presented as a systematic inquiry progressing through three stages: first, adapting the standard externally for DNNs; second, integrating its mechanisms internally as a trainable network component; and finally, re-applying its principles to enhance the process of knowledge transfer.

Our investigation proceeds from the "outside-in," first treating the DNN as a fixed entity and asking: how can we adapt the external JPEG encoding process to serve the machine's task? This reframes the traditional 'rate-distortion' problem, which balances file size and human-perceived quality, into a new 'rate-accuracy' problem, seeking to maximize a network's inferential accuracy for a given bit rate. In our work on DNN-Oriented Compression, we introduce a novel, machine-centric distortion measure based on the network's sensitivity to frequency-domain perturbations.

This allows for the formulation of JPEG-compliant quantization tables that yield the counter-intuitive result of enabling significant compression while preserving, and in some cases even improving, the accuracy of a pre-trained network. This revealed a powerful principle: for a given task, machine-centric compression acts as a beneficial form of structured input regularization, a finding that foreshadows applications beyond efficient inference.

This initial success led to a more profound question: if external adaptation is beneficial, what could be achieved if the optimization were integrated within the network itself? To answer this, we addressed the primary obstacle to integration: the non-differentiable nature of JPEG's quantization step. By substituting the standard hard quantizer with a differentiable counterpart, we introduce the first trainable, end-to-end differentiable JPEG layer. When prepended to a standard DNN, this differentiable JPEG layer forms the core of our JPEG-Inspired Deep Learning (JPEG-DL) framework, enabling joint optimization of the network and the compression algorithm. This approach transforms JPEG from a static pre-processing utility into a dynamic, trainable component of the network itself, introducing a learnable, frequency-domain non-linearity.

This thesis further extends the principle of structured input regularization to the domain of Knowledge Distillation (KD) [44]. A core challenge in KD is the "overconfident teacher" that results from using the original training set, where models produce near-one-hot predictions that offer a poor supervisory signal for student learning [17, 78]. While perturbing the teacher's input is an established method for eliciting "softer" outputs, this process is often computationally expensive. Furthermore, the underlying theoretical objective—maximizing the teacher's Conditional Mutual Information (CMI)—can be hampered by inaccurate proxies in prior work [137]. We propose to address these dual challenges by reconceptualizing JPEG encoding as a structured and efficient perturbation operator.

To this end, our investigation proceeds systematically. The approach begins with Coded Knowledge Distillation (CKD), a practical heuristic demonstrating the viability of using adaptive JPEG for this task. Building on this, we develop a more principled theoretical foundation with Generalized Coded Knowledge Distillation (GCKD) to more accurately formulate the CMI objective. Finally, we resolve the computational demands of this objective with our culminating synthesis, Differentiable JPEG-based Input Perturbation (DJIP). This framework utilizes a trainable differentiable JPEG layer as a learnable, amortized operator, enabling the efficient generation of rich supervisory signals in a single forward pass and unifying the tools of machine-centric compression with the core challenges of knowledge transfer.

In conclusion, this dissertation demonstrates that the relationship between JPEG and DNNs can be systematically revisited to create a powerful synergy. By progressing from adaptation to integration and synthesis, this work transforms the suboptimal default interaction of JPEG and DNNs into a versatile methodological tool. It argues for a fundamental shift in perspective: from treating JPEG as a legacy constraint to reimagining it as a powerful component of modern

deep learning methodology. By demonstrating that legacy codecs can be repurposed to enhance model accuracy, efficiency, and knowledge transfer, this work reframes the role of classical compression, proposing JPEG-inspired encoding as a principled foundation for integrating classical signal processing and deep learning.

1.2 Thesis Roadmap: From Adaptation to Synthesis

The narrative of this thesis unfolds across three principal stages, each building upon the last to establish a comprehensive framework for JPEG-inspired deep learning. This progression represents a journey of inquiry, moving from adapting existing standards, to integrating them into novel network structures, and finally to synthesizing these innovations to solve challenges in a new domain.

1.2.1 Part I: Revisiting JPEG's Foundations for Machine Vision

The first stage of this inquiry confronts the inherent misalignment between the human-centric design of the JPEG standard and the perceptual requirements of deep learning. The conventional paradigm of Human-Oriented Compression (HOC), which underpins JPEG, is formulated to optimize the rate-distortion trade-off for the Human Visual System (HVS). This design philosophy becomes a liability when images are consumed by machines, whose perceptual mechanisms differ fundamentally from our own. This disconnect motivates a pivotal research question: Given a fixed, pre-trained DNN, how can we compress an image in a fully JPEG-compliant manner to maximize its inferential accuracy?

Our foundational work, **JPEG Compliant Compression for DNN Vision** [101, 144], addresses this by pioneering the paradigm of **DNN-Oriented Compression (DOC)**. This represents a fundamental shift in objective, replacing the traditional goal of visual fidelity with the machine-centric goal of accuracy, thereby optimizing the rate-accuracy trade-off. To this end, we introduce the **Sensitivity Weighted Error (SWE)**, a novel, machine-centric distortion measure that quantifies error not in terms of human perception, but in terms of a DNN's training loss sensitivity to perturbations in the Discrete Cosine Transform (DCT) domain. By incorporating SWE into an advanced JPEG pipeline, we propose **OptS**, an algorithm for deriving optimal, model-specific quantization tables. This approach creates JPEG-compliant bitstreams that are maximally informative for a target DNN, establishing a principled method for aligning the JPEG standard with machine perception. Critically, this work provides the first empirical evidence for a counter-intuitive phenomenon: that intelligent, lossy compression can preserve and, in some

cases, even improve a DNN's inferential accuracy, achieving significant rate reduction of up to 67.84% with no loss in performance.

1.2.2 Part II: Integrating the Encoding Pipeline into the DNN

Having established that JPEG's external parameters can be optimized for a fixed DNN, the next logical question is whether the encoding process itself can become a learned, integral component of the network. This stage of our research moves beyond treating compression as a static pre-processing step and integrates the encoding pipeline directly into the network as a trainable layer. The primary obstacle to this vision—the non-differentiable nature of JPEG's hard quantization step—is overcome within this layer by employing a differentiable soft quantization mechanism. This key substitution transforms the entire pipeline into a fully differentiable and trainable module. Prepending this layer to a standard network serves a dual purpose: it enables end-to-end joint optimization of the compression parameters and the network weights, and it injects a powerful, learnable frequency-domain non-linearity into the model.

Our work on **JPEG Inspired Deep Learning (JPEG-DL)** [99] realizes this vision. By prepending a lightweight, trainable JPEG layer to a standard DNN, JPEG-DL transforms the static codec into a dynamic feature extractor. The network learns not only what features to extract in the spatial domain but also how to optimally represent and quantize information in the frequency domain for its specific task. As a result, this fusion of classical signal processing and deep learning leads to significant improvements in both model accuracy and adversarial robustness, demonstrating that the JPEG processing pipeline itself can function as a powerful, differentiable component within modern deep learning.

1.2.3 Part III: Re-Application to Knowledge Transfer: From Heuristic to Theory and Synthesis

The final stage of this thesis applies the principles of JPEG-inspired encoding to an entirely different domain: knowledge transfer via Knowledge Distillation (KD) [44]. A common failure mode in KD is the "overconfident teacher," a large model that produces near-one-hot probability distributions, which serve as a poor, uninformative supervisory signal for the student [17, 78]. This motivates a new objective: using input encoding not for compression, but as a perturbation operator to modulate the teacher's outputs, making them "softer" and more information-rich for the student. Our investigation into this problem progresses through a "heuristic $\rightarrow$ theory $\rightarrow$ synthesis" arc.

The journey begins with a pragmatic heuristic. Our initial work, **Coded Knowledge Distillation (CKD)** [98], repurposes the adaptive JPEG encoding from Part I to perturb the teacher's input. While effective, this method is limited by its discrete search space. To build a more principled foundation, we first analyze the theoretical objective. While maximizing the teacher's **Conditional Mutual Information (CMI)** has been identified as the goal in prior literature, we identify that existing methods are hampered by an inaccurate CMI proxy that assumes fixed class centroids during optimization. Our work on **Generalized Coded Knowledge Distillation (GCKD)** [100] directly resolves this by introducing a more accurate formulation and a novel alternating optimization algorithm that dynamically updates these centroids. While theoretically sound, this framework introduces significant computational overhead due to its iterative, online optimization process.

This limitation motivates the final, culminating step of our research: an efficient synthesis. Our work on **Differentiable JPEG-based Input Perturbation (DJIP)** [15] achieves this by deliberately fusing the theoretical objective from GCKD with the practical mechanism from Part II. DJIP leverages the **trainable, differentiable JPEG layer** as a specialized, learnable operator to solve the CMI maximization problem. Instead of a costly online search, DJIP trains the layer offline to become an amortized CMI maximizer, replacing iterative optimization with a single, efficient forward pass. The result is a lightweight, plug-and-play "coded teacher" that is both theoretically grounded and highly efficient, representing the synthesis of the thesis's primary threads.

1.3 Thesis Contributions

This thesis establishes JPEG-inspired encoding as a versatile and powerful framework for enhancing deep learning models. The key contributions, systematically developed through a logical progression of research, are summarized as follows:

- **A JPEG-Compliant Compression Framework for Machine Vision.** This thesis introduces the first principled method for adapting the standard JPEG codec for machine consumption without altering decoder compatibility, demonstrating that optimized, task-aware compression can improve model performance.
    1. We propose a novel, machine-centric distortion measure, the **Sensitivity Weighted Error (SWE)**, which is based on a DNN's training loss sensitivity to perturbations in the DCT domain [101].
    2. We propose **OptS**, an algorithm that leverages SWE to derive optimal JPEG quantization tables, which demonstrably improve the rate-accuracy performance for a given pre-trained DNN.

- **An Integrated, Trainable Compression Layer for Deep Learning.** This thesis introduces **JPEG-Inspired Deep Learning (JPEG-DL)**, a novel paradigm that fuses the JPEG pipeline into the network as a dynamic, trainable component.
    1. We propose a novel differentiable JPEG layer that overcomes the primary technical barrier of standard JPEG by employing a soft quantization mechanism. This innovation renders the entire JPEG pipeline amenable to gradient-based optimization and transforms the layer into a trainable, frequency-domain non-linearity, enhancing the model's expressive power.
    2. We demonstrate that prepending this lightweight, trainable JPEG layer to a standard DNN and jointly optimizing its parameters with the network's weights leads to significant improvements in accuracy and adversarial robustness [99].
- **A Unified Framework for Knowledge Distillation Enhancement.** This thesis pioneers the use of JPEG-inspired encoding to solve the "overconfident teacher" problem in KD, progressing from a practical heuristic to a theoretically grounded and efficient synthesis.
    1. We introduce **Coded Knowledge Distillation (CKD)**, a practical method that uses adaptive JPEG encoding as a perturbation operator to elicit softer, more informative outputs from a teacher model [98].
    2. We formalize this approach with **Generalized Coded Knowledge Distillation (GCKD)**, an information-theoretic framework that defines the objective as maximizing the teacher's Conditional Mutual Information (CMI) and provides a general, albeit computationally intensive, optimization algorithm [100].
    3. We synthesize these ideas in **Differentiable JPEG-based Input Perturbation (DJIP)**, which utilizes the trainable JPEG layer as an amortized CMI maximizer. This results in a highly efficient, plug-and-play "coded teacher" that is both theoretically sound and computationally lightweight [15].

1.4 Thesis Organization

The remainder of this thesis is structured to follow the logical progression of the research journey outlined above.

- **Chapter 2** establishes the necessary background across three key domains. It first reviews the classical JPEG compression paradigm and its optimization for human vision. It then surveys the challenges and opportunities at the intersection of compression and deep

learning. Finally, it introduces foundational concepts from Knowledge Distillation and information theory, specifically Conditional Mutual Information (CMI), which are central to the later contributions of the thesis.

- **Chapter 3** formally introduces and evaluates our framework for JPEG-compliant, DNN-Oriented Compression. We detail the formulation of the Sensitivity Weighted Error (SWE), the formulation of the OptS algorithm, and present empirical results demonstrating that our optimized compression improves the rate-accuracy trade-off.
- **Chapter 4** presents JPEG-Inspired Deep Learning (JPEG-DL). This chapter details the structure of the trainable JPEG layer and the differentiable soft quantization mechanism that enables it, demonstrating how its joint optimization with a DNN improves model performance.
- **Chapter 5** begins our exploration of knowledge distillation by presenting the heuristic-based Coded Knowledge Distillation (CKD) framework, which uses adaptive JPEG compression as a practical tool for teacher output modulation.
- **Chapter 6** develops the theoretical foundation for this approach with the Generalized Coded Knowledge Distillation (GCKD) framework. It formalizes the objective as Conditional Mutual Information maximization and presents a general algorithm for its solution.
- **Chapter 7** introduces Differentiable JPEG-based Input Perturbation (DJIP), which provides an efficient and elegant synthesis of the thesis's core ideas. This chapter demonstrates how the trainable JPEG layer can be utilized to solve the CMI maximization problem with minimal computational overhead.
- **Chapter 8** concludes the dissertation by summarizing its key contributions and their broader implications. It also discusses the limitations of the current work and outlines promising directions for future research that build upon the methodologies developed herein.

Chapter 2

Background

This chapter establishes the foundational knowledge across two converging fields: image compression and machine learning. This provides the essential context for the contributions of this thesis, which lie at the intersection of these domains. We begin by deconstructing the baseline JPEG standard, the most ubiquitous compression scheme for still images, to understand its core mechanisms. We then trace its evolution through the lens of rate-distortion (R-D) theory, reviewing advanced, human-centric optimization frameworks like Soft-Decision Quantization (SDQ) and Optimal Distortion (OptD) that represent the pinnacle of compression for human vision.

Having established this classical paradigm, we pivot to the modern era where Deep Neural Networks (DNNs) have become the primary consumers of image data. We survey the literature demonstrating the detrimental impact of standard, HVS-tuned compression on DNN performance and review the common mitigation strategies developed by the computer vision community. We then explore a crucial counter-narrative: the recent discovery that intelligently applied, adaptive compression can paradoxically enhance DNN accuracy, though initial attempts to harness this phenomenon have been architecturally complex and suboptimal. To find a more principled path forward, we turn to the parallel field of model compression, specifically Knowledge Distillation (KD), and examine a state-of-the-art information-theoretic approach for improving teacher models by maximizing their Conditional Mutual Information (CMI).

2.1 The JPEG Image Compression Standard

The proliferation of digital imaging across consumer and cloud applications necessitates efficient methods for data storage and transmission. Image compression standards address this by reducing statistical and perceptual redundancies inherent in image data. As depicted in Figure 2.1,

a typical image processing pipeline involves an end device (e.g., a camera) capturing raw pixel data, an encoder compressing this data to meet channel bandwidth or storage constraints, and a decoder reconstructing the image for an observer. This observer may be a human, or increasingly, a machine learning model such as a Deep Neural Network (DNN).

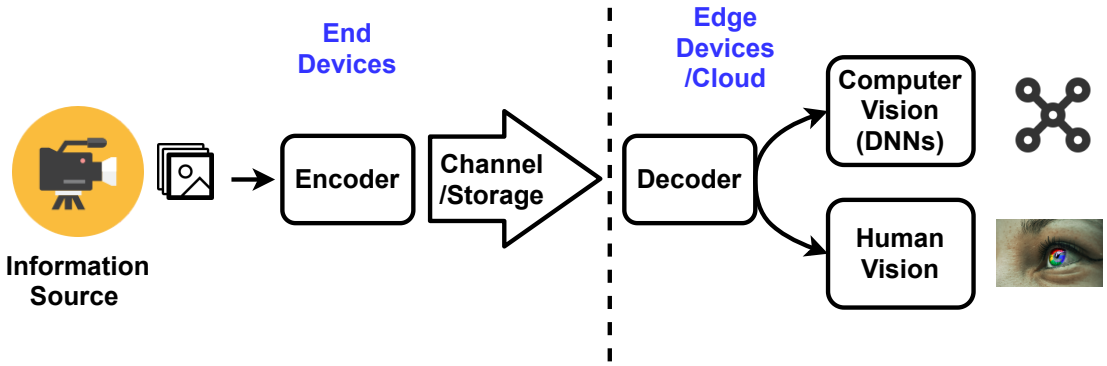


Figure 2.1: A typical pipeline of image acquisition, encoding, storage/transmission, and decoding for both human and machine consumption.

Among the various standards, the Joint Photographic Experts Group (JPEG) standard is the most dominant for continuous-tone still images [91]. The baseline JPEG encoding pipeline, illustrated in Figure 2.2, is a block-based transform coding scheme designed to exploit redundancies in a multi-stage process.

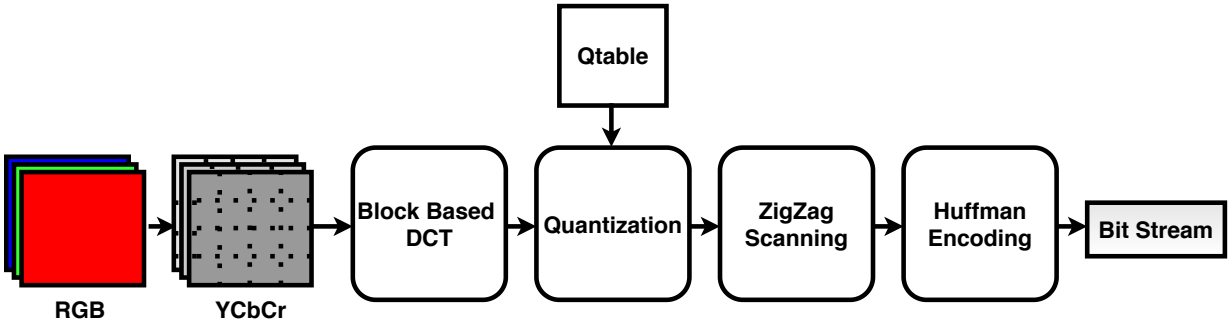


Figure 2.2: The baseline JPEG compression pipeline, adapted from [135]. The quantization and Huffman tables are free parameters that can be optimized by the encoder.

The encoding process begins with a color space transformation, typically from RGB to YCbCr. This conversion separates the image into a luminance (Y) channel, which represents intensity, and two chrominance (Cb, Cr) channels, which represent color-difference information.

The Human Visual System (HVS) is significantly more sensitive to fine details in luminance than in chrominance. JPEG exploits this by allowing the chrominance channels to be subsampled (e.g., 4:2:0 chroma subsampling), effectively reducing the spatial resolution of the color information and achieving a preliminary, lossless reduction in data volume.

Each channel is then partitioned into non-overlapping 8×8 pixel blocks. To decorrelate the spatially redundant pixel data within each block, a 2D Discrete Cosine Transform (DCT) is applied. The DCT converts the 8×8 block from the spatial domain to a frequency domain representation, defined by the forward transform:

$$F_{u,v} = \frac{1}{4} C(u)C(v) \sum_{i=0}^{7} \sum_{j=0}^{7} f_{i,j} \cos\left[\frac{(2i+1)u\pi}{16}\right] \cos\left[\frac{(2j+1)v\pi}{16}\right] \tag{2.1}$$

where $f_{i,j}$ is the pixel value at spatial coordinate (i, j), $F_{u,v}$ is the resulting DCT coefficient at frequency coordinate (u, v), and $C(k)$ is a normalizing factor, with $C(k) = 1/\sqrt{2}$ for $k = 0$ and $C(k) = 1$ otherwise. A key property of the DCT is its energy compaction; for typical images, most of the signal energy is concentrated into a few low-frequency coefficients, particularly the DC coefficient $F_{0,0}$, which represents the average intensity of the block.

The core of JPEG's lossy compression is the quantization stage. This non-linear and non-invertible operation reduces the precision of the DCT coefficients, discarding information that is perceptually less significant. In baseline JPEG, this is achieved through uniform scalar quantization, where each of the 64 DCT coefficients $F_{u,v}$ is divided by a corresponding quantization step size $q_{u,v}$ and rounded to the nearest integer:

$$F'_{u,v} = \lfloor \frac{F_{u,v}}{q_{u,v}} \rceil \tag{2.2}$$

and the dequantized value used for reconstruction is

$$\tilde{F}_{u,v} = q_{u,v}\, F'_{u,v}. \tag{2.3}$$

This deterministic rounding rule is known as **Hard-Decision Quantization (HDQ)**. The set of 64 step sizes, $\{q_{u,v}\}$, constitutes an 8×8 quantization table (Q-table). The JPEG standard provides default Q-tables (Figure 2.3), which were designed based on empirical studies of HVS sensitivity. These tables feature larger step sizes for high-frequency components, to which the human eye is less sensitive, and smaller step sizes for low-frequency components. These default tables can be scaled by a single Quality Factor (QF) to control the trade-off between compression ratio and visual quality.

$$\mathbf{Q}_{\text{luma}} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 14 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad \mathbf{Q}_{\text{chroma}} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Figure 2.3: Default JPEG quantization tables for luminance (left) and chrominance (right) channels, designed for the Human Visual System [91].

Finally, the quantized coefficients are prepared for entropy coding. The 8×8 block of quantized coefficients is scanned in a zig-zag order, converting it into a 1D sequence. This ordering groups low-frequency coefficients at the beginning of the sequence and high-frequency coefficients, many of which are now zero, at the end. This structure is highly amenable to Run-Length Coding (RLC), which efficiently represents long runs of zeros. The resulting RLC symbols are then entropy-coded, typically with Huffman coding, to produce the final compressed bitstream. The JPEG syntax leaves the quantization and Huffman tables as free parameters, offering a significant opportunity for optimization at the encoder.

For the Huffman coding stage, the JPEG standard provides a flexible framework that allows for one of two approaches. The first approach involves using a set of pre-defined, "standard" Huffman tables. These generic tables are based on the statistical properties of an average set of images and offer a fast and straightforward method of entropy coding, as they do not require any image-specific analysis. Most simple JPEG encoders use these fixed tables.

The second, more advanced approach is to generate custom Huffman tables tailored to the unique statistical properties of each individual image. This is typically achieved through a two-pass encoding process. In the first pass, the encoder analyzes the frequency of the quantized DCT coefficients throughout the image to build an optimal set of Huffman codes. In the second pass, these custom-generated tables are used to perform the actual entropy coding. For a decoder to correctly reconstruct the image, these custom tables must be embedded directly into the JPEG bitstream, usually within a Define Huffman Table (DHT) segment. While including these tables adds a small amount of overhead to the file size, the use of image-specific codes almost always results in a more efficient compression of the image data itself. This method yields a better compression ratio for the same level of visual quality, as the entropy coding is perfectly adapted to the image content. For the work presented in this thesis, we will exclusively utilize custom

Huffman tables generated for each image to ensure optimal entropy coding.

2.2 Rate-Distortion Optimization for Human Vision

The primary limitation of baseline JPEG is its reliance on fixed, image-independent quantization tables and the suboptimal HDQ process. This motivated extensive research into rate-distortion (R-D) optimization, aiming to design image-adaptive compression parameters that minimize distortion for a given rate budget (or vice versa), thereby improving performance for human observers. This pursuit led to two significant advancements: Soft-Decision Quantization, which optimizes the encoding process itself, and Optimal Distortion table design, which optimizes the quantization table for that process.

2.2.1 Soft-Decision Quantization (SDQ)

A significant advance beyond HDQ was the development and formalization of **Soft-Decision Quantization (SDQ)** [135]. In HDQ, the quantized DCT index is determined by a fixed rounding rule. In contrast, SDQ treats the selection of the quantized DCT indices as a free parameter, chosen to explicitly minimize an overall R-D cost. This allows the encoder to make more nuanced decisions. For instance, it might quantize a coefficient to zero even if its magnitude is slightly greater than half the quantization step size, if the resulting rate saving (from creating a longer zero-run) outweighs the small increase in distortion.

The SDQ framework formulates JPEG encoding as a joint optimization problem over the sequence of run-size pairs and amplitudes (R, S, A), the Huffman table H, and the quantization table Q. The goal is to minimize a Lagrangian cost function:

$$\min_{(R,S,A),H,Q} J(\lambda) = d[I_o, (R, S, A)_Q] + \lambda \cdot r[(R, S), H] \tag{2.4}$$

where I_o is the original image, $d[\cdot]$ is a distortion measure (typically Mean Squared Error, MSE), $r[\cdot]$ is the bit rate, and λ is the Lagrange multiplier that controls the trade-off between rate and distortion.

As direct minimization of Equation 2.4 is intractable, Yang and Wang [135] proposed an elegant iterative algorithm that alternates between two main steps:

1. **Graph-Based RLC Optimization:** For a fixed quantization table Q_t and Huffman table H_t, the optimal sequence of quantized coefficients (R, S, A) is found. This is solved independently for each 8×8 block by finding the shortest path on a specially constructed

directed graph. The graph contains 65 nodes (states), representing the 64 DCT coefficients in zig-zag order and a terminal "end-of-block" state. A transition from node j to node i ($i > j$) represents quantizing coefficients $j+1, \ldots, i-1$ to zero. The cost of each path edge is the incremental Lagrangian cost, which combines the distortion incurred and the rate required for that specific encoding decision. For a transition from state $i-r-1$ to i with size group s, this cost is:

$$\underbrace{\sum_{j=i-r}^{i-1} C_j^2 + (C_i - \hat{C}_i)^2}_{\text{Distortion (MSE)}} + \lambda \cdot \underbrace{(-\log_2 P(r,s) + s)}_{\text{Rate}} \tag{2.5}$$

where C_k is the original DCT coefficient, $\hat{C}_i$ is the reconstructed coefficient, and $P(r,s)$ is the probability of the run-size symbol.

2. **Quantization and Huffman Table Update:** For the fixed sequence (R_t, S_t, A_t) found in the previous step, the tables are updated. The Huffman table H_{t+1} is re-calculated based on the empirical statistics of the new run-size pairs. The quantization table Q_{t+1} is updated to the one that minimizes the distortion for the given quantized indices. For MSE, this has a closed-form solution for each step size q_i:

$$q_{i,t+1} = \frac{\sum_j C_{i,j} \cdot K_{i,j}}{\sum_j K_{i,j}^2} \tag{2.6}$$

where the sum is over all blocks j, and $C_{i,j}$ and $K_{i,j}$ are the original DCT coefficient and the chosen quantized index for frequency i in block j, respectively.

This iterative process continues until convergence. While highly effective, the performance of this SDQ framework can be sensitive to the choice of the initial quantization table Q_0, as the optimization is not guaranteed to find a global minimum.

2.2.2 Optimal Distortion Quantization Table Design (OptD)

To address the initialization dependency of iterative methods like SDQ, Yang et al. [134] revisited the problem of Q-table design from a fundamental, information-theoretic perspective. The objective was to design a single, image-adaptive quantization table that is optimal for the generic purpose of reconstruction, under the assumption that a powerful R-D optimal encoder (such as SDQ) will subsequently be used. This work, known as Optimal Distortion (OptD) design, established an elegant principle for deriving a near-optimal table in a non-iterative fashion.

The derivation begins by modeling the DCT coefficients at each of the 64 frequency positions as independent random sources, $\{X_i\}_{i=1}^{64}$. By approximating the R-D function of each source with its Shannon Lower Bound (SLB), the authors showed that for a total distortion budget D_T, the total rate is minimized when the distortion D_i allocated to each frequency component i follows a "reverse water-filling" profile:

$$D_i^* = \min(d^*, \sigma_i^2) \tag{2.7}$$

where σ_i^2 is the variance of the coefficients at frequency i, and d^* is a constant "water level" chosen to satisfy the total distortion budget, $\sum_i D_i^* = D_T$. This profile allocates more distortion to high-variance (high-energy) frequency components, up to a maximum of their total variance (at which point they can be discarded with no rate cost), while allocating a constant, smaller distortion d^* to lower-variance components.

With this optimal distortion profile D_i^* established, the OptD design principle is to find the quantization table $Q^* = \{q_i^*\}$ for which the expected distortion resulting from standard HDQ exactly matches this optimal profile:

$$E\left[(X_i - \lfloor \frac{X_i}{q_i^*} \rceil q_i^*)^2\right] = D_i^* \tag{2.8}$$

This formulation provides a direct, non-iterative method to compute a theoretically grounded, near-optimal quantization table based only on the image statistics (specifically, the DCT coefficient variances σ_i^2). This OptD table serves as an excellent initialization Q_0 for the iterative SDQ framework, effectively mitigating its dependency on a naive starting point and leading to state-of-the-art compression performance for human vision.

2.3 JPEG Compression for Machine Vision

The optimization techniques described in Section 2.2 were meticulously designed with the HVS as the target observer, using perceptually motivated distortion metrics like MSE. However, the paradigm is shifting as DNNs become the primary consumers of visual data for tasks such as classification, detection, and segmentation. This raises a critical question: how does compression designed for humans affect these machine observers?

A significant body of research has confirmed that standard, HVS-oriented compression can be detrimental to DNN performance. An early study by Dodge and Karam [25] examined the effect of JPEG and JPEG2000 compression on the accuracy of several popular DNNs on the ImageNet dataset [24]. They demonstrated that while DNNs exhibit a degree of resilience, their accuracy

consistently degrades as the compression level increases (i.e., as image quality decreases). This fundamental observation has been repeatedly confirmed in subsequent works [31, 36].

The predominant response from the computer vision community has been to treat compression as a source of data corruption and to focus on mitigation strategies. A common approach is robust training, where models are fine-tuned or trained from scratch on datasets augmented with compressed images. For instance, Ehrlich et al. [28] proposed randomly compressing training images at various quality factors as a form of data augmentation. This method improves the model's robustness to compression artifacts, but it often comes at the cost of reduced accuracy on original, uncompressed images.

In summary, the existing literature establishes that standard compression negatively impacts machine vision tasks and has primarily explored mitigation through data augmentation and robust training. The problem has largely been viewed from a computer vision perspective, rather than a fundamental compression theory perspective. The advanced R-D optimization principles, which proved so powerful for HVS-centric compression, have not been systematically applied to this new domain.

It is important to clarify that, throughout this dissertation, we treat the images provided by standard vision datasets as the "raw" images for our compression framework. In practice, truly raw sensor data (e.g., uncompressed Bayer-format images) is almost never available in the deep learning community. Instead, datasets such as ImageNet, CIFAR, CUB-200, and others distribute images that have already undergone at least one prior JPEG (or JPEG-like) compression stage before being released. Consequently, for the purpose of designing and evaluating DNN-oriented compression algorithms, the community universally accepts these already-compressed images as the effective raw inputs. Our formulations, experiments, and comparisons therefore consider the dataset-provided images as the baseline uncompressed representations, since no access to genuine camera-RAW imagery exists for large-scale DNN training or evaluation.

2.4 Enhancing DNN Performance with Adaptive Compression

Contrary to the conventional wisdom outlined above, recent work has demonstrated that intelligently applied compression can, paradoxically, improve DNN performance [131]. This phenomenon, termed "compression helps," is justified by the idea that compression can act as a form of task-aware filtering. By removing irrelevant high-frequency noise or distracting background textures, compression can effectively increase the signal-to-noise ratio for the features that are most salient to the DNN's task, leading to a better prediction.

The potential for this effect was first shown theoretically by Yang et al. [131] using an oracle-guided experiment. For a given image and a pre-trained DNN, an oracle would test every possible JPEG quality factor and select the one that resulted in the highest classification accuracy. Across a large dataset, this oracle-guided adaptive compression consistently outperformed the baseline of using uncompressed images, proving that an optimal compression level exists for each image that can enhance DNN performance.

To create a practical implementation of this idea, the same authors proposed a new DNN topology incorporating multiple parallel branches. In this architecture, one branch receives the raw image while the other branches receive versions of the image compressed at different, fixed quality levels. The penultimate layer representations from all branches are concatenated and passed to a final classification head, and the entire multi-branch structure is trained jointly. While this approach did show performance gains, it suffers from significant limitations:

- **Architectural Complexity:** The use of multiple parallel backbones results in a prohibitively large and complex model that is difficult to train and deploy.
- **Suboptimal Training:** Due to the immense computational cost, the authors resorted to partial training strategies, which likely resulted in a suboptimal final model.
- **Fixed Quantization:** The framework relied on the default, HVS-oriented JPEG quantization tables, which are not designed for machine perception and limit the potential gains.

This body of work successfully established the ”compression helps” phenomenon but revealed that harnessing it effectively requires a more principled and less complex approach than creating elaborate, non-standard DNN architectures.

2.5 Model Compression and Teacher Enhancement

Parallel to the advancements in data compression, the field of model compression has emerged to address the challenges of deploying large DNNs on resource-constrained devices. Among the various techniques, **Knowledge Distillation (KD)** is a particularly prominent paradigm [44]. The core idea is to transfer the ”knowledge” from a large, cumbersome ”teacher” model to a smaller, more efficient ”student” model. The student is trained not only to predict the correct class labels (hard targets) but also to mimic the full output probability distribution (soft targets) of the teacher. This richer supervisory signal helps the student learn a better-generalized representation than it could from hard labels alone.

A key challenge in KD, however, is that powerful teacher models often become overconfident when trained on finite datasets. Their output distributions become ”peaky,” closely resembling one-hot vectors, which diminishes the informational value of the soft-target signal [98]. This has

motivated a line of research focused on improving the quality of the teacher's supervision. Rather than simply mitigating overconfidence, advanced methods aim to produce a teacher whose output distributions provide a more principled and informative signal for the student.

A state-of-the-art approach to this problem is rooted in information theory, specifically aiming to maximize the **Conditional Mutual Information (CMI)** of the teacher model [137]. The CMI, denoted ($I(X; \hat{Y}|Y)$), measures the mutual information between the input (X) and the model's prediction ($\hat{Y}$), conditioned on the true label (Y). A teacher model with a higher CMI provides a more structured and informative signal, as its predictions are more consistent for samples within the same class while being more distinct across different classes. In essence, maximizing CMI encourages the teacher's output distributions to better approximate the true Bayes conditional probability distribution, ($P(Y|X)$), which is the ideal, most informative supervisor for the student [137].

While promising, the method proposed in the seminal MCMI paper [137] to optimize this objective has significant practical and theoretical limitations. First, it requires fine-tuning the parameters of the pre-trained teacher model. This is often intractable or undesirable in real-world applications where the teacher may be a proprietary black-box model, has already been deployed, or where modifying a high-performing model is simply not feasible. Second, the proposed optimization relies on a surrogate objective that uses fixed class-wise centroids. This approach is suboptimal because the centroids inevitably shift as the model is updated during fine-tuning. Failing to account for this dynamic nature leads to an inaccurate CMI estimation, which can accumulate errors and hinder the overall optimization process. These limitations underscore the need for a method that can enhance the teacher's CMI without altering its fixed parameters.

Chapter 3

JPEG Compliant Compression for DNN Vision

This chapter introduces a JPEG-compliant image compression framework tailored for deep neural network (DNN) vision, addressing the limitations of traditional human-oriented compression methods. A novel distortion metric, termed Sensitivity Weighted Error (SWE), is proposed to quantify the impact of discrete cosine transform (DCT) domain perturbations on a DNN's loss function, effectively capturing the model's sensitivity to different frequency components. By integrating SWE into the Soft Decision Quantization (SDQ) process, the framework transitions from human-oriented to DNN-oriented compression (DOC). To optimize this process, a new algorithm called OptS is developed to design quantization tables for both luminance and chrominance channels, adapting compression to the sensitivity profile of the target DNN. Extensive experiments on the ImageNet dataset using various DNN architectures, including MobileNetV2, MnasNet, RegNet, and ConvNeXt, demonstrate that the proposed method significantly improves rate-accuracy performance compared to standard JPEG. Notably, it achieves up to 67.84% reduction in compression rate without sacrificing classification accuracy, and in some cases, even enhances accuracy beyond that of raw images. The framework also exhibits robustness and generalizability, with sensitivity estimates proving transferable across models within the same architectural family. These results highlight the effectiveness of SWE and OptS in optimizing image compression for machine perception, offering a scalable and efficient solution for DNN-based vision systems.

The conference version of this chapter is available in [144], while the journal version can be found in [101].

3.1 Introduction

Image compression is one of the fundamental domains in image processing and computer vision (CV), which has been well developed in the past decades. Among all image compression approaches, JPEG [91] has become the *defacto* one, considering its extensive application. JPEG is a lossy compression technique developed for the human visual system, dubbed human-oriented compression (HOC) in this chapter. HOC algorithms can aggressively compress an image as they allow information loss during compression which is imperceptible for humans.

With the success of deep learning, a dramatic change is now happening in terms of how images are consumed. In the last decade, numerous deep neural networks (DNNs) have been proposed and used for CV tasks, such as image classification [41, 104], semantic segmentation [70, 97], object detection [32, 33], image generation [34, 53], etc. More and more images have been and will continue to be viewed by DNN-based intelligent machines, where humans would step in only when these intelligent machines fail their vision tasks. However, DNN perception differs in general from human perception [125]. The information loss incurred in the process of HOC may not be imperceptible for DNNs anymore, resulting in degraded DNN performance [8, 68, 125].

In order to aggressively compress images without sacrificing DNN's performance, DNN-oriented compression (DOC) is of great interest. So far, there are only a few of research works addressing the DOC problem [59, 60, 68]. These existing approaches, however, are subject to a variety of limitations, such as poor rate-accuracy (R-A) performance, high computational complexity, lack of scalability, and incompatibility with the existing JPEG-like compression framework.

In this chapter, we first propose a novel distortion measure dubbed the sensitivity weighted error (SWE). Given a DNN, the SWE distortion measure is customized for the vision of that DNN, based on the sensitivity of the training loss function of that DNN with respect to perturbations in the discrete cosine transform (DCT) domain. With the SWE distortion measure, any DCT-based HOC algorithm including JPEG can be converted to a DOC algorithm without additional complexity. Then, we embed SWE into soft decision quantization (SDQ) to switch it from an HOC approach to its DOC counterpart. Additionally, based on SWE and OptD [134], we propose OptS, a DOC quantization table design algorithm with full JPEG compatibility, to provide optimal quantization tables, i.e. reconstruction spaces, for the SDQ process.

The experimental results demonstrate that our proposed method, when applied to baseline JPEG encoding either with SDQ or without SDQ, provides outstanding R-A performance. The evaluation is carried out on the ImageNet validation set [24], adopting four prevalent models from different architectural families, namely MobileNetV2 [102], MnasNet [113] and RegNets [95],

and ConvNeXts [67], as the target DNNs. Comparing to the default JPEG at the same compression rate in bits per pixel (bpp), our DOC algorithm stands out by improving the classification accuracy of target DNNs by up to 2.12%. Alternatively, it can substantially reduce the compression rate of the default JPEG by as much as 67.84% while maintaining the classification accuracy. Furthermore, if we tolerate some accuracy loss up to 0.4% compared to the default JPEG, our DOC algorithm can achieve an impressive compression ratio (CR) of 9.1×[1], reducing the compression rate of the default JPEG by 73.72%. Remarkably, compression with OptS yields accuracy even slightly better than the inference accuracy of the raw dataset, while achieving a significant reduction in the rate. Indeed, this gain in accuracy can reach up to 0.07% with a CR of 3.77×.

The rest of the chapter is organized as follows. First, some background and related work are discussed in Section 3.2. Then, we formulate the problem in Section 3.3. After that, we propose a novel distortion measure for DNN vision in Section 3.4, based on which our DOC framework and algorithm are established in Section 3.5. To evaluate our proposed method, extensive experimental results are shown in Section 3.6. Finally, we conclude the paper in Section 3.7.

3.2 Background and Related Work

3.2.1 Soft Decision Quantization

JPEG encoding begins with the transformation of the non-overlapped partition input using discrete cosine transform (DCT) coefficients. Subsequently, the DCT coefficients undergo uniform quantization using a fixed quantization table, commonly known as hard decision quantization (HDQ). The DCT indices resulting from this quantization are subsequently subject to entropy coding, including run-length coding and Huffman coding. The JPEG framework offers a significant opportunity to apply rate distortion (R-D) optimization at the encoder, where three free parameters, namely quantization tables, Huffman tables, and run-length coding, can be optimized.

Soft Decision Quantization (SDQ), as outlined in [134–136], takes advantage of all these three parameters and optimizes them jointly, where the quantization process is tightly coupled with the lossless coding method. The SDQ process involves the joint optimization of Huffman tables, quantization step sizes, and DCT indices within a JPEG encoder using an iterative optimization algorithm. This algorithm alternates between two main steps: (1) graph-based run-length

[1]The compression ratio equals 24 divided by the compression rate.

coding optimization and (2) optimal quantization table updating. In the first step, given fixed Huffman tables and quantization step sizes, an efficient graph-based algorithm was proposed to find optimal DCT indices in the form of run-size-amplitude triples, denoted as $(\mathcal{R}, \mathcal{S}, \mathcal{A})$. Operating as an optimal path searching algorithm, this graph-based approach fully optimizes the DCT indices in the form of the $(\mathcal{R}, \mathcal{S}, \mathcal{A})$ sequence. It can not only drop less significant DCT indices but also can change them from one size group to another. Remarkably, it can even change a zero DCT index to a nonzero value if necessary in the R-D sense. In the second step, given the updated DCT indices from the optimized $(\mathcal{R}, \mathcal{S}, \mathcal{A})$ sequence, each quantization step size is updated. There's a cost function associated with the iterative algorithm, and the algorithm stops when the cost function converges or a specified number of iterations is reached. Finally, the algorithm outputs optimal DCT indices in the form of $(\mathcal{R}, \mathcal{S}, \mathcal{A})$ sequence, a quantization table for reconstruction purpose, and a customized Huffman table based on the statistics of the output run-size pairs $(\mathcal{R}, \mathcal{S})$.

Notably, although each updating step is optimal by itself, the iterative algorithm does not converge, in general, to a global optimum [134]. As a result, the SDQ process exhibits high sensitivity to the initialization of the quantization table. To address this problem, OptD was proposed in [134] as a quantization table optimization algorithm within the SDQ context, aiming to provide an optimal quantization table design for the iterative algorithm. Note that since OptD is a quantization table design method, it is feasible to utilize it within HDQ as well.

3.2.2 DNN-oriented Compression

In the literature, there are a few existing works on DOC codec, also known as *Coding for Machines*, wherein the image compression process is optimized for the benefits of DNN inference rather than the visual quality by the human eye. For example, DeepN-JPEG [68] claims that the standard deviation of a DCT coefficient determines its importance to DNN inference, thus using a piece-wise linear function to map the standard deviations of DCT coefficients to a quantization table. Moreover, GRACE [125] uses the gradient magnitude of DNN's training loss function with respect to its input image to measure DNN's perceptual sensitivity to different DCT frequencies and different color channels, based on which an algorithm is proposed to optimize the quantization tables and color space conversion[2]. Prakash *et al.* [94] employed the generation of a saliency map for content-aware JPEG encoding, utilizing different quality factors (QFs) for a single image. Strumpler *et al.* [107] proposed a joint learning of the JPEG quantization table along with the implementation of a frequency-domain pre-editing network, which is designed

[2]Note that, while both GRACE and our work propose the concept of sensitivity, it is important to highlight that the definition and application of sensitivity differ significantly between the two approaches.

to optimize the distribution of DCT coefficients. Choi *et al.* [18] proposed a task-specific approach to learning quantization tables by adjusting the objective function of the network. They mainly replaced the non-differentiable components in the encoder, such as run-length encoding and Huffman coding, by learning a differentiable loss function that approximates bitrates using simple network blocks. Lee *et al.* [59] proposed an autoencoder-based image compression technique dubbed N-epitomizer, where the scalable encoder is trained to extract the essential information and scale its output size to meet the accuracy requirements of inferences over the target model. DeepComp [115] is also based on autoencoder, where the decoder and the target DNN are connected to the encoder in parallel and they are trained with a modified rate-distortion loss function with additional DNN accuracy term to reconstruct an image well and improve DNN accuracy simultaneously [45, 123].

The above DOC approaches are subject to various limitations, reflected by poor R-A performance, high computational cost, lack of scalability, and incompliance with the existing JPEG-like compression framework. In contrast, our proposed method has addressed these problems by introducing a novel distortion measure customized for DNN perception, which is embedded into the SDQ pipeline to provide a comprehensive framework for solving the DOC problem. As a result, our approach has outstanding R-A performance, moderate complexity, high flexibility and scalability, and full JPEG compatibility.

3.3 Problem Formulation

In JPEG, an encoder first partitions an image x into B non-overlapping 8×8 blocks, and then applies DCT to each of these non-overlapping blocks to obtain DCT coefficients C. After flattening the DCT coefficients in each block in zigzag order, we get $M = 64$ sequences $\{C_i\}_{i=1}^{M}$ of DCT coefficients, each with length B, where each sequence $C_i = \{C_{i,j}\}_{j=1}^{B}$ corresponds to a distinct DCT frequency position $1 \leq i \leq M$. We further model these sequences as independent sources. Without loss of generality, each DCT source C_i is assumed to have a zero mean; otherwise, the mean should be subtracted.

After DCT, the next step of JPEG is to perform quantization. Define a quantization table $Q = \{q_1, q_2, \ldots, q_M\}$, where q_i is the quantization step size for C_i. In the conventional hard decision quantization (HDQ) framework, we quantize DCT coefficients $C_{i,j}$ to DCT indices and reconstruct them back with Q following $\tilde{C}_{i,j} = q_i \lfloor C_{i,j}/q_i \rceil$, where $\tilde{C}$ denotes the reconstructed DCT coefficients. We further denote the quantization error by $\Delta C = C - \tilde{C}$, which is the perturbation introduced to DCT coefficients by the quantization process.

In the case of SDQ, however, we can divide the quantization process into two components: a quantization table or reconstruction space, known as Q, and a mapping function or quantizer,

denoted as Q'_Q. This quantizer maps each sequence of DCT coefficients C_i at the ith frequency position into an index sequence of length B from $\{0, \pm 1, \pm 2, \ldots, \pm\lfloor G/q_i \rceil\}^B$, where G is the maximum possible magnitude of the DCT coefficients. During reconstruction, each index sequence is multiplied with q_i to obtain $\tilde{C}_i$, following the same manner as HDQ.

Given a lossless coding method ϕ for encoding index sequences (which is specified as Huffman coding in this chapter), quantizer design within the SDQ framework is equivalent to solving the following optimization problem:

$$\inf_{Q} \inf_{Q'_Q} R_\phi(Q, Q'_Q) \qquad \textbf{s.t. } D(Q, Q'_Q) \leq D_T \tag{3.1}$$

where $R_\phi(Q, Q'_Q)$ is the compression rate (in bits) of the image x resulting from using ϕ to encode the index sequences given by Q'_Q, $D(Q, Q'_Q)$ is the distortion resulting from the SDQ process Q'_Q in conjunction with Q, and D_T is a prescribed distortion budget.

Since ϕ and D_T are predetermined, given an image x, the resulting optimal Q and Q'_Q solely depend on the distortion measure D. For HOC, the inner and outer minimization problems in equation 3.1 have already been solved by SDQ [135] and OptD [134] respectively, assuming the mean squared error (MSE) distortion. In this chapter, we revisit equation 3.1 for DOC, by proposing a novel distortion measure D customized for DNN vision.

3.4 Distortion Measure for DNN Vision

From the above section, we can see that distortion is of pivot significance in our problem. Therefore, a natural question is how to measure the distortion in the case of DOC. Conventionally, MSE is widely used in HOC to measure the distortion between the original DCT coefficients C and reconstructed DCT coefficients $\tilde{C}$: $D(Q, Q') = \sum_{i=1}^{M} \sum_{j=1}^{B} \Delta C_{i,j}^2$. Note that in MSE, all DCT frequencies are treated equally. For DNN vision, however, observations in [125] and [124] reveal that the gradient magnitude of DNN's training loss function with respect to its input image varies across DCT frequencies, which motivates us to differentiate the distortions resulting from the quantization of different sources C_i. In this section, we first define DNN's sensitivity to DCT perturbations formally, and then propose a novel distortion measure customized for DNN vision based on it.

3.4.1 DNN's Sensitivity to DCT Perturbations

Given a DNN, we denote its loss function used in its training stage as L. We want to understand how sensitive L is with respect to the perturbation ΔC added to the DCT coefficients C of an

input image x. To simplify our discussion, the derivation below is limited to a single channel of an image but can be easily extended to multiple channels of the image. By Taylor expansion, we have

$$L(C + \Delta C) = L(C) + [\nabla L(C)]^T \Delta C + o(||\Delta C||) \tag{3.2}$$

when ΔC is small enough, where $\nabla L(C)$ is the gradient vector of L with respect to C, and $||\Delta C||$ denotes the l_2 norm of ΔC. Therefore, the loss increase caused by the perturbation is upper bounded by

$$\begin{aligned}
|\Delta L| &= |L(C + \Delta C) - L(C)| \\
&\leq |[\nabla L(C)]^T \Delta C| + |o(||\Delta C||)| \\
&= \left| \sum_{i=1}^{M} \frac{\mathrm{d}L}{\mathrm{d}C_i^T} \Delta C_i \right| + |o(||\Delta C||)| \\
&\leq \sum_{i=1}^{M} \left| \frac{\mathrm{d}L}{\mathrm{d}C_i^T} \Delta C_i \right| + |o(||\Delta C||)| \\
&\leq \sum_{i=1}^{M} \sqrt{\sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 \cdot \sum_{j=1}^{B} \Delta C_{i,j}^2} + |o(||\Delta C||)| \qquad (3.3)
\end{aligned}$$

where the last inequality follows from the Cauchy–Schwarz inequality. Squaring both sides of (3.3), we have

$$\begin{aligned}
\Delta L^2 \leq & \left(\sum_{i=1}^{M} \sqrt{\sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 \cdot \sum_{j=1}^{B} \Delta C_{i,j}^2} \right)^2 + o(||\Delta C||^2) \\
& + 2 \left(\sum_{i=1}^{M} \sqrt{\sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 \cdot \sum_{j=1}^{B} \Delta C_{i,j}^2} \right) \cdot o(||\Delta C||) \qquad (3.4) \\
= & \left(\sum_{i=1}^{M} \sqrt{\sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 \cdot \sum_{j=1}^{B} \Delta C_{i,j}^2} \right)^2 + o(||\Delta C||^2) \qquad (3.5) \\
\leq & M \sum_{i=1}^{M} \sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 \cdot \sum_{j=1}^{B} \Delta C_{i,j}^2 + o(||\Delta C||^2) \qquad (3.6)
\end{aligned}$$

where equation 3.5 holds given that the cross term in equation 3.4 is also in the order of $o(||\Delta C||^2)$ and hence absorbed into $o(||\Delta C||^2)$. Note that equation 3.6 is valid for any small perturbation

ΔC. When the perturbation ΔC is limited only to the DCT frequency i, equation 3.6 then becomes

$$\Delta L^2 \leq \sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 \cdot ||\Delta C_i||^2 + o(||\Delta C_i||^2) \tag{3.7}$$

where $||\Delta C_i||^2 = \sum_{j=1}^{B} \Delta C_{i,j}^2$. Therefore, the squared rate of change of L with respect to C_i is upper bounded by

$$\frac{\Delta L^2}{||\Delta C_i||^2} \leq \sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 + o(1) \tag{3.8}$$

In view of equation 3.8, we now define the given DNN's sensitivity to the perturbation at the i^{th} frequency position as

$$s_i = \sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}} \right)^2 = \left|\left| \frac{\mathrm{d}L}{\mathrm{d}C_i} \right|\right|^2, \quad 1 \leq i \leq M \tag{3.9}$$

Across all frequencies, DNN's sensitivity can be characterized by the set $S = \{s_1, s_2, \ldots, s_M\}$. Note that the computation of such defined DNN's sensitivity $S = \{s_1, s_2, \ldots, s_M\}$ depends on the input image C (in the DCT domain), the DNN, and the ground truth (GT) label of the input image C, which may not be available at the encoder.

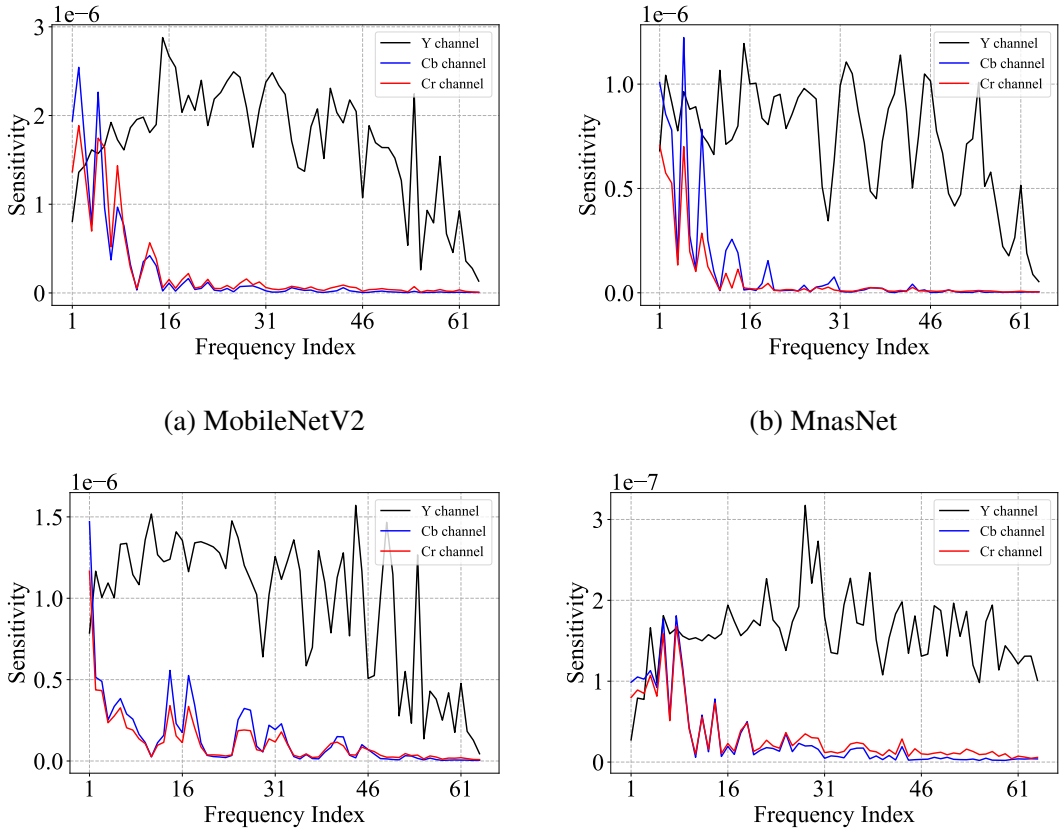
3.4.2 Offline Estimation of the Sensitivity

To remove the dependency on the availability of the DNN and GT label of the input image C at the time of encoding, we estimate the sensitivity S for each target DNN offline.

Specifically, for each target DNN, we first randomly select N image samples, and then estimate S by taking the sample mean of the sensitivity over the N image samples:

$$s_i = \frac{1}{N} \sum_{k=1}^{N} \sum_{j=1}^{B} \left(\frac{\partial L}{\partial C_{i,j}^k} \right)^2, \quad 1 \leq i \leq M \tag{3.10}$$

where C^k is the DCT coefficient of the k^{th} input image x^k. The partial derivatives in (3.10) are obtained through backpropagation.



(a) MobileNetV2

(b) MnasNet

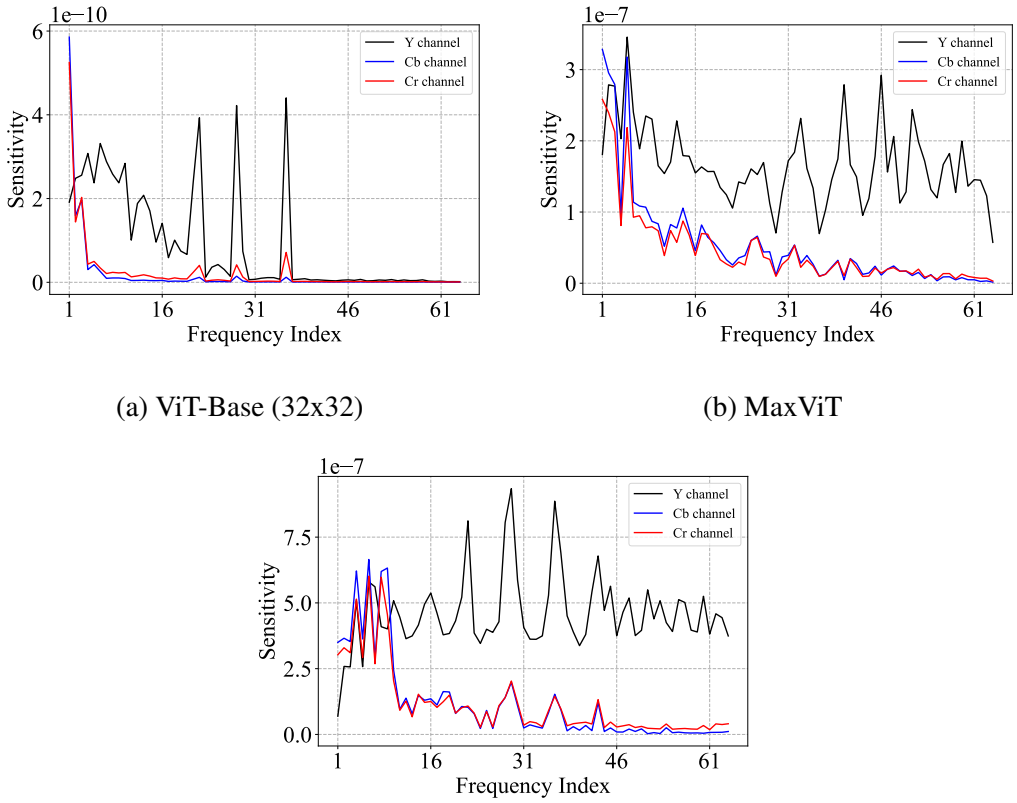
(c) RegNetY-400MF

(d) ConvNeXt-T

Figure 3.1: The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, with sensitivity indices arranged in the default zigzag order.

Following (3.10), for the image classification task, we estimate the sensitivity[3] for four popular convolutional-based architectures pre-trained on ImageNet [56], namely MobileNetV2 [102], MnasNet [113], RegNetX-400MF [95], and ConvNeXt-T [67], and visualize them in Fig.3.1. Additionally, the sensitivity for some transformer-based models is demonstrated in Fig.3.2. In this demonstration, the underlining loss function L is the cross-entropy loss as we select image classification as our CV task; the input images x are sampled from the ImageNet training set enclosing all classes, and $N = 10,000$. Note that, before inputting the image samples into the

[3]Following the convention, RGB images are converted into the YCbCr color space before compression, where Y channel is known as the luminance channel while Cb and Cr channels are known as chrominance channels. Accordingly, the sensitivity is also estimated for Y, Cb and Cr channels respectively.



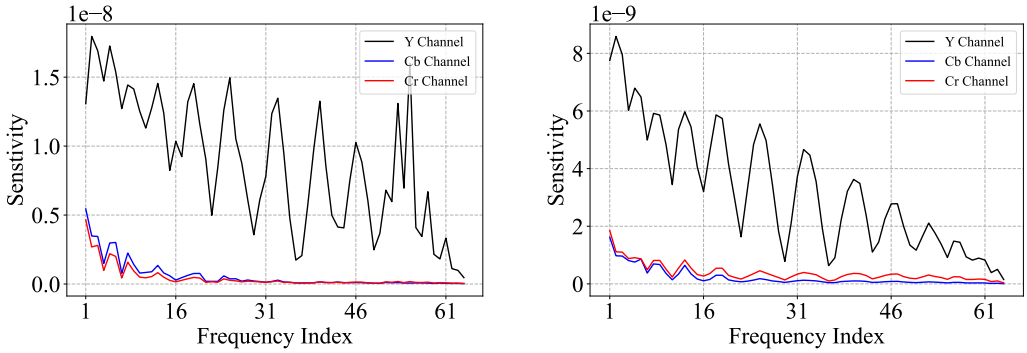
(a) ViT-Base (32x32)

(b) MaxViT

(c) Swin-T

Figure 3.2: The estimated sensitivity is shown for different Vision Transform (ViT) Models pre-trained models on the ImageNet dataset, with sensitivity indices arranged in the default zigzag order.

DNN models for estimation, all images are scaled to a common size of $H \times W$ as part of the standard preprocessing procedure, where the height H and width W depend on the input layer of the target DNN model. Additional information about the tested models is provided in Section 3.6.1. In Fig. 3.1, the sensitivity of the tested models is arranged following the default zigzag order, where low and high frequencies are arranged on the left and right, respectively. It is evident that the sensitivity of DNN differs significantly from that of the HVS, which is high at low frequencies and decreases dramatically as the frequency increases. A similar procedure is applied to the semantic segmentation task using DeepLabv3-MobileNet and DeepLabv3-ResNet50 [13] on the Pascal VOC 2012 dataset, with pre-trained models obtained from [30].



(a) DeepLabv3-MobileNet (b) DeepLabv3-ResNet50

Figure 3.3: The estimated sensitivity is shown for different pre-trained models on Pascal VOC 2012 dataset, with sensitivity indices arranged in the default zigzag order.

Table 3.1: Top-1 validation accuracies are shown for MobileNetV2 and RegNetY-400MF with different pre-trained weights.

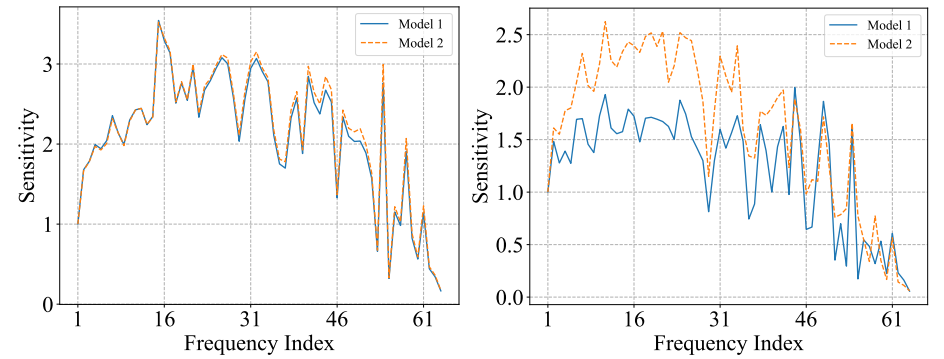
Model	w_1	w_2
MobileNetV2	71.870%	72.012%
RegNetY-400MF	74.026%	75.804%

To further support the idea that sensitivity is an inherent property of the model architecture rather than the result of specific training strategies, we compare the sensitivity of the same model using two distinct sets of pre-trained weights. These weights, obtained through different training methods, are provided by PyTorch[4]. For this evaluation, we select MobileNetV2 and RegNetY-400MF, each with two different pre-trained versions, and report their Top-1 validation accuracies in Table 3.1. As shown in Fig. 3.4, the sensitivity behavior remains consistent across the different pre-trained weights of the same model. Based on this observation, we firmly believe that the sensitivities are transferable among different pre-trained weights of the same DNN.

3.4.3 Similarity of Sensitivity Among Different DNN Models

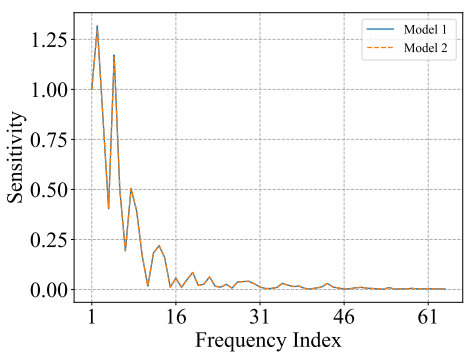
Given a specific loss function L, the sensitivity is an inherent trait of a DNN model, which solely depends on the pre-trained model itself. Therefore, if a DNN's architecture or weights vary, its sensitivity will change accordingly, which is reflected by Fig. 3.1. As such, we need to do

[4]PyTorch code can be found in https://github.com/pytorch/vision/tree/main/references/classification.

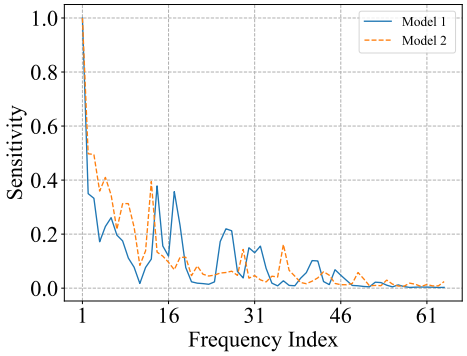


(a) Y Channel

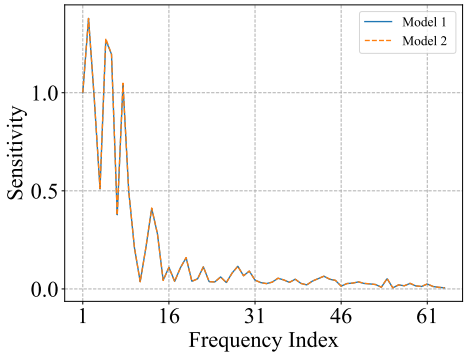
(b) Y Channel



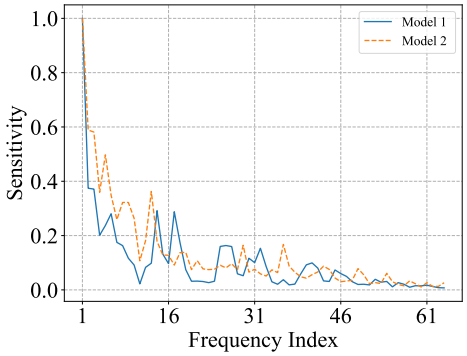
(c) Cb Channel



(d) Cb Channel

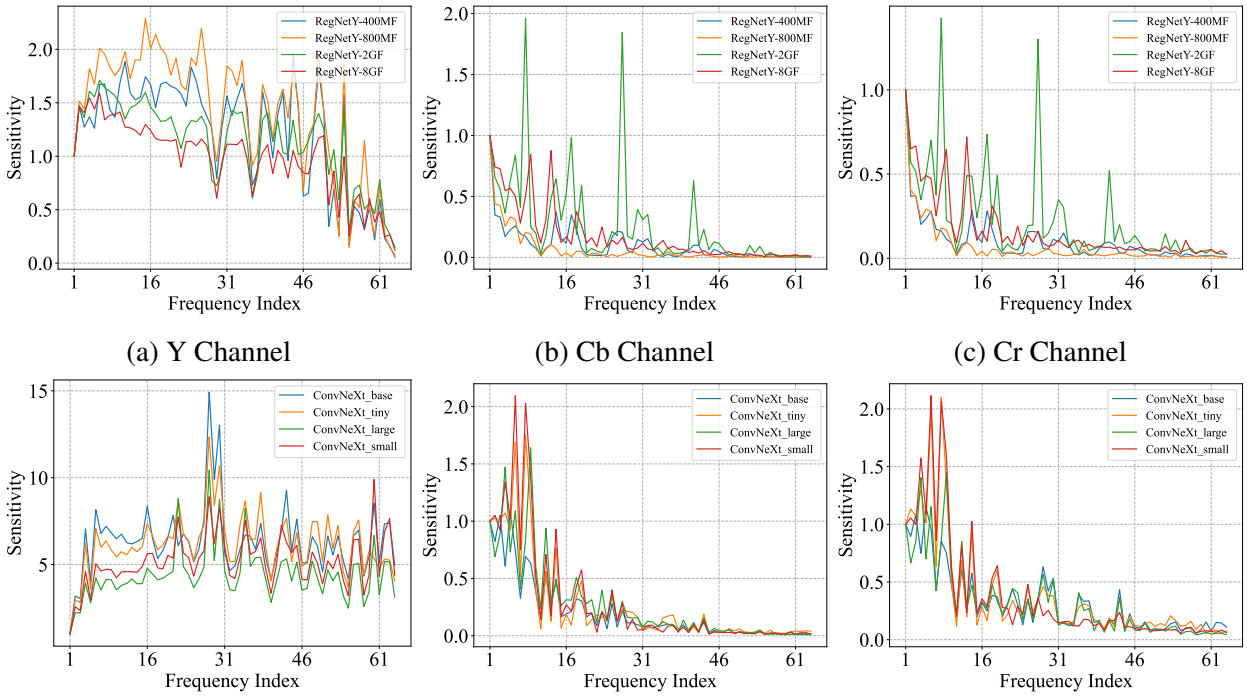


(e) Cr Channel



(f) Cr Channel

Figure 3.4: Sensitivities for MobileNetV2 with two distinct pre-trained weights are compared in (a), (c) and (e). Sensitivities for RegNetY-400MF with two distinct pre-trained weights are compared in (b), (d) and (f). Each channel's sensitivity is normalized by the sensitivity value of the DC frequency for better visualization.



(a) Y Channel (b) Cb Channel (c) Cr Channel

(d) Y Channel (e) Cb Channel (f) Cr Channel

Figure 3.5: The sensitivity of different models within the RegNets and ConvNeXts families, normalized by the sensitivity value of the DC frequency for better visualization. It is evident that DNN variants within the same architecture family exhibit a similar sensitivity pattern.

sensitivity estimation whenever we face a new DNN model, which is troublesome in reality if the target DNN changes frequently. However, we observe that the sensitivity of different DNN models within the same architecture family is quite similar, which alleviates the above problem to some extent.

As depicted in Fig. 3.5, we estimate the sensitivity for ConvNeXts and RegNets with different size and pre-trained weights. It is clear that the sensitivity estimates among DNN variants within the same architecture family highly resemble each other. This makes intuitive sense as these DNN variants share the same network architecture. Given this observation, we believe the sensitivity is transferable within the same architecture family without much error, which will be verified experimentally in Section 3.6.2.

3.4.4 Sensitivity Weighted Error (SWE)

Go back to equation 3.6. Note that $M = 64$ is fixed. With the sensitivity S estimated via equation 3.10, we can now measure the distortion between the original DCT coefficients C and reconstructed DCT coefficients $\tilde{C}$ as follow:

$$D(Q, Q') = \sum_{i=1}^{M} s_i \sum_{j=1}^{B} \Delta C_{i,j}^2. \tag{3.11}$$

The above distortion measure is called the sensitivity weighted error (SWE). In view of equation 3.6 and equation 3.9, the SWE for an image is a good estimation of the upper bound of the squared DNN loss increase caused by quantization. Minimizing the SWE subject to a rate constraint will in turn reduce the squared DNN loss increase.

3.5 Quantizer Design

Based on the proposed SWE, we now turn back to solve Problem equation 3.1 formulated in Section 3.3. Actually, to solve this problem for DOC, one can simply embed SWE into existing HOC algorithms as a substitute for MSE, since SWE is a generic distortion measure independent of the algorithm.

In this section, we first extend the SDQ process from HOC to DOC by incorporating SWE, thus solving the inner minimization problem in equation 3.1. Then, the HOC quantization table design algorithm, namely OptD, which was proposed to provide an optimal quantization table for the SDQ process, will be revisited. A new algorithm, dubbed OptS, will be presented as a DOC counterpart of OptD, by which the outer minimization problem in equation 3.1 can be solved. These two components as a whole provides a complete DOC quantizer design under SDQ, which fully solve Problem equation 3.1 for DOC.

3.5.1 Soft Decision Quantization Revisited for DOC

In this subsection, we extend SDQ from HOC to DOC by replacing MSE with SWE. As explained in Section 3.2.1, the SDQ process consists of two alternating steps. When MSE is replaced by SWE, the optimal solution in Step (2) remains the same. The major difference lies in Step (1).

In Step (1) of the iterative algorithm, a graph-based run-length coding optimization algorithm is employed to optimize the $(\mathcal{R}, \mathcal{S}, \mathcal{A})$ sequence for an image. This is formulated as minimizing the following Lagrangian cost according to [135]:

$$\min_{(\mathcal{R},\mathcal{S},\mathcal{A})} \{\mathcal{J}(\beta) = D[x, (\mathcal{R}, \mathcal{S}, \mathcal{A})_Q] + \beta \cdot R[(\mathcal{R}, \mathcal{S}), P]\} \tag{3.12}$$

where $D[x, (\mathcal{R}, \mathcal{S}, \mathcal{A})_Q]$ denotes the distortion between the original image x and the reconstructed image determined by $(\mathcal{R}, \mathcal{S}, \mathcal{A})$ and a fix quantization table (or reconstruction space) Q over all AC coefficients [5], and $R[(\mathcal{R}, \mathcal{S}), P]$ denotes the compression rate for all AC coefficients resulting from the chosen sequence $(\mathcal{R}, \mathcal{S}, \mathcal{A})$ and the Huffman table H. Note that we use the distribution P of run-size pairs $(\mathcal{R}, \mathcal{S})$ to replace the Huffman table H, since P completely determines a Huffman table. Lastly, β is a balancing weight between rate and distortion. In the graph-based algorithm proposed to solve this problem, the abstract Lagrangian cost $\mathcal{J}(\beta)$ is implemented by an incremental cost associated to each transition from state $l - r - 1$ to state l in the graph, shown as follow:

$$\sum_{i=l-r}^{l-1} C_i^2 + (C_l - q_l A_l)^2 + \beta \cdot (-\log_2 P(r, sz) + sz) \tag{3.13}$$

where r is the run of this transition, A_l represents the selected amplitude for the l^{th} DCT index in size group sz that gives rise to the minimum distortion to C_l among all allowed amplitudes within size group sz, and $q_l \in Q$ is the l^{th} quantization step size. Clearly, the distortion measure D is MSE here.

To incorporate our proposed SWE distortion into equation 3.12, a straightforward substitution is performed. Particularly, the distortion measure D used in equation 3.12 is switched from MSE to SWE. This entails scaling the MSE distortion to each DCT coefficient by its corresponding sensitivity value. As a result, the incremental cost equation 3.13 is updated to:

$$\sum_{i=l-r}^{l-1} s_i C_i^2 + s_l (C_l - q_l A_l)^2 + \beta \cdot (-\log_2 P(r, sz) + sz). \tag{3.14}$$

3.5.2 OptS: Designing Optimal Quantization Tables for DOC

In this subsection, we will propose a quantization table design algorithm dubbed OptS, which solves the outer minimization problem in equation 3.1 in order to provide optimal quantization tables for the SDQ process when DOC is considered. OptS consists of two components which

[5] The optimization of DC coefficients can be addressed separately using a trellis diagram.

are (1) OptS for the luminance channel and (2) OptS for chrominance channels. The first component is a straightforward extension of OptD [134], while the second component has a nontrivial difference from OptD. In the following, we will begin with briefly reviewing OptD, and then we will propose the two components of OptS one by one.

A Brief Review of OptD

In [134], authors assume that ϕ in equation 3.1 is universal and optimal so that the inner minimization in equation 3.1 can be approximated by the Shannon rate distortion functions of DCT coefficients, which can be further approximated by their Shannon lower bounds. In this way, Q and Q'_Q are decoupled, so that the optimal solution of Q no longer depends on the SDQ design. Therefore, following the derivation in [134], solving Problem equation 3.1 is equivalent to solving

$$\inf_{Q} \inf_{\substack{\{D_i\}_{i=1}^{M}:\sum_{i=1}^{M} D_i = D_T \\ D_i \geq D(C_i,q_i)}} \sum_{i=1}^{M} \left[R_{C_i}^{(SL)}(D_i) + \frac{c}{q_i}\frac{\ln B}{B} + o\left(\frac{\ln B}{B}\right)\right]. \tag{3.15}$$

where $c = O(1)$ is a positive bounded term, D_i is the distortion budget allocated to the i^{th} source, $R_{C_i}^{(SL)}(D_i)$ is the Shannon lower bound to the rate distortion function of C_i, and $D(C_i, q_i)$ is the minimal achievable distortion for the source C_i, given the reconstruction space

$$\left\{0, \pm 1 q_i, \pm 2 q_i, \ldots, \pm \left\lfloor \frac{G}{q_i} \right\rfloor q_i \right\}^B.$$

Note that $D(C_i, q_i)$ is actually equal to the distortion resulting from the HDQ of C_i with the quantization step size q_i. As shown in [134], the optimal quantization table $Q^* = \{q_1^*, q_2^*, \ldots, q_M^*\}$ resulting from equation 3.15 is given by

$$q_i^* = \sup\{q_i : D(C_i, q_i) \leq D_i^*\} \tag{3.16}$$

for any $1 \leq i \leq M$, where D_i^* is the optimal distortion budget allocated to the i^{th} source.

To design optimal quantization table for DCT-based image coding using equation 3.16, authors of [134] first model DC and AC coefficients with uniform and Laplacian distributions, so that $D(C_1, q_1) = q_1^2/12$ and $D(C_i, q_i)$, $2 \leq i \leq M$, can be approximated by $D_{Lap}(\lambda_i, q_i)$ calculated in (3.19). Then, a parameter d named water level is determined given the distortion budget D_T. With d and variances of sources $\{\sigma_i^2\}_{i=1}^M$, one can obtain the optimal distortion allocation

$\{D_i^*\}_{i=1}^M$. Therefore, given $\{D_i^*\}_{i=1}^M$ and $D(C_i, q_i)$, the optimal quantization table Q^* can be obtained based on equation 3.16.

$$\lambda_i = \frac{1}{B}\sum_{j=1}^{B} |C_{i,j}|\,,\ 2 \le i \le M \tag{3.17}$$

$$z_i = q_i - \lambda_i + \frac{q_i}{e^{q_i/\lambda_i} - 1} \tag{3.18}$$

$$D_{Lap}(\lambda_i, q_i) = 2\lambda_i^2 - \frac{2q_i(\lambda_i + z_i - 0.5q_i)}{e^{z_i/\lambda_i}(1 - e^{-q_i/\lambda_i})} \tag{3.19}$$

OptS for the Luminance Channel

OptS for the luminance channel is obtained by replacing MSE in OptD with SWE. This modification involves revisiting the Shannon lower bound to the rate distortion function of DCT coefficients. Specifically, we will derive the Shannon lower bound to the rate distortion function of C_i using SWE as the distortion measure.

Given the random variable C_i with probability density function (pdf) $p(c_i)$, the rate distortion function $R_{C_i}(D_i)$ of C_i is defined as

$$R_{C_i}(D_i) = \inf_{p(\tilde{c}_i|c_i):\mathbb{E}[D(C_i,\tilde{C}_i)]\le D_i} I(C_i; \tilde{C}_i) \tag{3.20}$$

where $p(\tilde{c}_i|c_i)$ is the conditional pdf of $\tilde{C}_i$ given $C_i = c_i$, and $D(.,.)$ is the SWE. Therefore, the constraint over the expected distortion can be rewritten as

$$\begin{aligned} \mathbb{E}[s_i(C_i - \tilde{C}_i)^2] &\le D_i \\ \mathbb{E}[(C_i - \tilde{C}_i)^2] &\le \frac{D_i}{s_i}. \end{aligned} \tag{3.21}$$

Also, we have

$$\begin{aligned} I(C_i; \tilde{C}_i) &= H(C_i) - H(C_i|\tilde{C}_i) \\ &= H(C_i) - H(C_i - \tilde{C}_i|\tilde{C}_i) \\ &\ge H(C_i) - H(C_i - \tilde{C}_i) \\ &\ge H(C_i) - \frac{1}{2}\log 2\pi e\tilde{\sigma}_i^2 \end{aligned} \tag{3.22}$$

where $H(C_i)$ is the differential entropy of C_i, $\tilde{\sigma}_i^2$ denotes the variance of $C_i - \tilde{C}_i$, and the last inequality is due to the maximum differential entropy lemma [20]. Since $\tilde{\sigma}_i^2 \le \mathbb{E}[(\tilde{C}_i - C_i)^2] \le \frac{D_i}{s_i}$, it follows from

Algorithm 1 OptS for the Luminance Channel

Input: x, $S = \{s_1, s_2, \ldots, s_M\}$, d, q_{max}
Output: $Q = \{q_1, q_2, \ldots, q_M\}$
  if $s_i \sigma_i^2 < d$ **then**
    set $q_i = q_{max}$
  else
    if $i = 1$ **then**
      set $q_i = min\left\{ \left\lfloor \sqrt{\frac{12d}{s_i}} \right\rfloor, q_{max} \right\}$
    else
      set $q_i = max\{q_i \in \mathcal{Q} : D_{Lap}(\lambda_i, q_i) \leq \frac{d}{s_i}\}$
    end if
  end if

equation 3.22 that

$$I(C_i, \tilde{C}_i) \geq H(C_i) - \frac{1}{2} \log 2\pi e \frac{D_i}{s_i}. \tag{3.23}$$

Therefore, we have

$$R_{C_i}(D_i) \geq H(C_i) - \frac{1}{2} \log 2\pi e \frac{D_i}{s_i}. \tag{3.24}$$

Also, the rate should be non-negative, so the Shannon lower bound to the rate distortion function of C_i is:

$$R_{C_i}^{(SL)}(D_i) = \begin{cases} H(C_i) - \frac{1}{2} \log 2\pi e \frac{D_i}{s_i} & \text{if } D_i < s_i \hat{\sigma}_i^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

where $\hat{\sigma}_i^2$ is chosen such that $H(C_i) = \frac{1}{2} \log 2\pi e \hat{\sigma}_i^2$.

As a result, we obtain can easily obtain the distortion expressions $D(C_1, q_1) = s_1 q_1^2/12$ and $D(C_i, q_i) = s_i D_{Lap}(\lambda_i, q_i)$, $2 \leq i \leq M$ from the pervious derivations. The resulting OptS algorithm for the luminance channel is demonstrated in Alg. 1, where $\mathcal{Q} = \{1, 2, \ldots, q_{max}\}$, and q_{max} is a predetermined maximum quantization step size.

OptS for Chrominance Channels

Since two chrominance channels share the same quantization table in the JPEG framework, we cannot obtain OptS algorithm for chrominance channels by tweaking OptD in a naive way. Although the theory foundation is more or less the same, yet OptS algorithm for chrominance channels has nontrivial differences from OptD. This is primarily because, in constructing a single quantization table for both

chrominance channels, we treat the Cb and Cr components jointly, forming a combined representation with $2M$ sources. In this formulation, the source i and source $i+M$ share the same quantization step size q_i, for $1 \leq i \leq M$. To solve this joint optimization problem, we follow the approach proposed in [134] and apply it to Problem equation 3.1 with $2M$ sources considered simultaneously, as shown in the squel.

We now consider Problem equation 3.1 with the adoption of SWE distortion. Based on the derivation in [134], we can convert this problem to

$$\inf_{Q} \inf_{\substack{\{D_i\}_{i=1}^{2M}: \sum_{i=1}^{2M} D_i = D_T \\ \forall i \in [1,M] \\ D_i \geq D(C_i, q_i) \\ D_{i+M} \geq D(C_{i+M}, q_i)}} \sum_{i=1}^{2M} \left[R_{C_i}^{(SL)}(D_i) + \frac{c}{q_i} \frac{\ln B}{B} + o\left(\frac{\ln B}{B}\right) \right]. \tag{3.26}$$

where $c = O(1)$ is a positive bounded term.

According to the KKT conditions, the inner minimization in equation 3.26 is achieved when

$$D_i = D_i(Q) \equiv \begin{cases} d & \text{if } D(C_i, q_i) \leq d \leq s_i \hat{\sigma}_i^2 \\ D(C_i, q_i) & \text{if } d < D(C_i, q_i) \\ s_i \hat{\sigma}_i^2 & \text{if } d > s_i \hat{\sigma}_i^2 \end{cases} \tag{3.27}$$

where $1 \leq i \leq M$, and

$$D_i = D_i(Q) \equiv \begin{cases} d & \text{if } D(C_i, q_{i-M}) \leq d \leq s_i \hat{\sigma}_i^2 \\ D(C_i, q_{i-M}) & \text{if } d < D(C_i, q_{i-M}) \\ s_i \hat{\sigma}_i^2 & \text{if } d > s_i \hat{\sigma}_i^2 \end{cases} \tag{3.28}$$

where $M+1 \leq i \leq 2M$. Here, d is chosen so that

$$\sum_{i=1}^{2M} D_i(Q) = D_T. \tag{3.29}$$

Following a derivation similar to that in [134], the optimal distortion allocation is

$$D_i^* = \begin{cases} d^* & \text{if } d \leq s_i \hat{\sigma}_i^2 \\ s_i \hat{\sigma}_i^2 & \text{otherwise} \end{cases} \tag{3.30}$$

where $1 \leq i \leq 2M$, and d^* is chosen such that

$$\sum_{i=1}^{2M} D_i^* = D_T. \tag{3.31}$$

Although the optimal distortion allocation here appears to be similar as that in [134], the optimal quantization step size q_i^* is a bit different when two channels are considered, where q_i is subject to two constraints instead of one. Accordingly,

$$q_i^* = \sup\{q_i : D(C_i, q_i) \leq D_i^*, D(C_{i+M}, q_i) \leq D_{i+M}^*\} \tag{3.32}$$

where $1 \leq i \leq M$.

Based on equation 3.30 and equation 3.32, there are four different cases for the selection of q_i^*:

(1) If $d^* > s_i\hat{\sigma}_i^2$ and $d^* > s_{i+M}\hat{\sigma}_{i+M}^2$, then $D_i^* = s_i\hat{\sigma}_i^2$ and $D_{i+M}^* = s_{i+M}\hat{\sigma}_{i+M}^2$. Based on equation 3.25, we see that $R_{C_i}^{(SL)}(D_i^*) = R_{C_{i+M}}^{(SL)}(D_{i+M}^*) = 0$. Therefore, we directly quantize all coefficients in both source i and source $i+M$ to zeros. As for q_i^*, both constraints in equation 3.32 are satisfied since $D(C_i, q_i)$ is upper bounded by $s_i\hat{\sigma}_i^2$ for all $2M$ sources. Therefore, we set q_i^* to a prescribed maximum quantization step size q_{max}.

(2) If $s_{i+M}\hat{\sigma}_{i+M}^2 \geq d^* > s_i\hat{\sigma}_i^2$, then $D_i^* = s_i\hat{\sigma}_i^2$ and $D_{i+M}^* = d^*$. Based on equation 3.25, we see that $R_{C_i}^{(SL)}(D_i^*) = 0$. Therefore, we directly quantize all coefficients in source i to zeros. As for q_i^*, the first constraint in equation 3.32 is satisfied. Therefore, ignoring the first constraint, we get

$$q_i^* = \sup\{q_i : D(C_{i+M}, q_i) \leq d^*\}. \tag{3.33}$$

(3) If $s_i\hat{\sigma}_i^2 \geq d^* > s_{i+M}\hat{\sigma}_{i+M}^2$, then $D_i^* = d^*$ and $D_{i+M}^* = s_{i+M}\hat{\sigma}_{i+M}^2$. Based on equation 3.25, we see that $R_{C_{i+M}}^{(SL)}(D_{i+M}^*) = 0$. Therefore, we directly quantize all coefficients in source $i+M$ to zeros. As for q_i^*, the second constraint in equation 3.32 is satisfied. Therefore, ignoring the second constraint, we get

$$q_i^* = \sup\{q_i : D(C_i, q_i) \leq d^*\}. \tag{3.34}$$

(4) If $d^* \leq s_i\hat{\sigma}_i^2$ and $d^* \leq s_{i+M}\hat{\sigma}_{i+M}^2$, then $D_i^* = D_{i+M}^* = d^*$. Therefore, we get

$$q_i^* = \sup\{q_i : D(C_i, q_i) \leq d^*, D(C_{i+M}, q_i) \leq d^*\}. \tag{3.35}$$

Based on the above four cases, and given that $D(C_1, q_1) = s_1 q_1^2/12$, $D(C_{1+M}, q_1) = s_{1+M} q_1^2/12$, $D(C_i, q_i) = s_i D_{Lap}(\lambda_i, q_i)$ for $i \in [2, M] \cup [2+M, 2M]$, Alg. 2 is obtained. Notably, $\hat{\sigma}_i^2$ is replaced by σ_i^2 in the algorithm to further simplify the computation of Q^* without involving the computation of differential entropy. Note that the sensitivity used in Alg. 2 has $2M$ entries as it includes the sensitivity of both Cb and Cr channels. Also, some notations are overloaded in Alg. 1 and Alg. 2 for simplicity.

3.5.3 Analysis for Quantization Tables Designed by OptS

To show the effectiveness of OptS in designing quantization tables for DNN vision, we compare the quantization tables designed by OptS with the default quantization tables in JPEG to demonstrate their

Algorithm 2 OptS for Chrominance Channels

Input: x, $S = \{s_1, \ldots, s_M, s_{M+1}, \ldots, s_{2M}\}$, d, q_{max}
Output: $Q = \{q_1, q_2, \ldots, q_M\}$

if $max\{s_i\sigma_i^2, s_{i+M}\sigma_{i+M}^2\} < d$ **then**
  set $q_i = q_{max}$
else if $s_i\sigma_i^2 < d \leq s_{i+M}\sigma_{i+M}^2$ **then**
  if $i = 1$ **then**
    set $q_i = min\left\{\left\lfloor\sqrt{\frac{12d}{s_{i+M}}}\right\rfloor, q_{max}\right\}$
  else
    set $q_i = max\{q_i \in \mathcal{Q} : D_{Lap}(\lambda_{i+M}, q_i) \leq \frac{d}{s_{i+M}}\}$
  end if
else if $s_{i+M}\sigma_{i+M}^2 < d \leq s_i\sigma_i^2$ **then**
  if $i = 1$ **then**
    set $q_i = min\left\{\left\lfloor\sqrt{\frac{12d}{s_i}}\right\rfloor, q_{max}\right\}$
  else
    set $q_i = max\{q_i \in \mathcal{Q} : D_{Lap}(\lambda_i, q_i) \leq \frac{d}{s_i}\}$
  end if
else if $d \leq min\{s_i\sigma_i^2, s_{i+M}\sigma_{i+M}^2\}$ **then**
  if $i = 1$ **then**
    set $q_i = min\left\{\left\lfloor\sqrt{\frac{12d}{max\{s_i, s_{i+M}\}}}\right\rfloor, q_{max}\right\}$
  else
    set $q_i = max\Big\{q_i \in \mathcal{Q} : D_{Lap}(\lambda_i, q_i) \leq \frac{d}{s_i},$
    $D_{Lap}(\lambda_{i+M}, q_i) \leq \frac{d}{s_{i+M}}\Big\}$
  end if
end if

difference. Intuitively, since we apply SWE as the distortion measure when designing quantization tables, more MSE distortion will be allocated at sources where the sensitivity values are relatively low. This is because, after being weighted by the low sensitivity values, the consumption of the SWE budget will not be very significant, and vice versa. As a result, larger quantization step sizes will be assigned to sources with lower sensitivity values. However, such a relationship between q_i and s_i does not exist in the case of default JPEG quantization tables, since they are designed for HVS rather than DNN vision.

To confirm our claim above, for each of the tested models, we randomly select an image sample from the ImageNet validation set and visually compare the following: (1) quantization tables designed for this image by OptS; (2) the default quantization tables in JPEG; and (3) the sensitivity of the tested model. This comparison is shown in Fig. 3.6, where it is important to note that the figure shows $1/q_i$ instead of

Table 3.2: Correlation between sensitivity and the inverse of the designed quantization table by OptS averaged over the ImageNet validation set, is compared to the correlation between sensitivity and the inverse of the default JPEG quantization table.

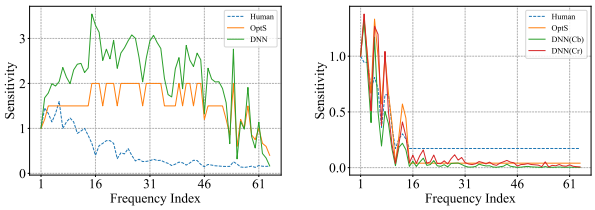
Model	OptS		JPEG	
	Y	CbCr	Y	CbCr
MobileNetV2	0.917	0.934	0.169	0.945
MnasNet	0.733	0.914	0.366	0.877
RegNetY-400MF	0.737	0.917	0.454	0.703
ConvNext-T	0.799	0.904	-0.303	0.833

q_i. Also, all curves are normalized by their respective values of sensitivity and inverse q corresponding to DC for better visualization. In Fig. 3.6, it is evident that the trend of $1/q_i$ for the designed quantization table by OptS closely resembles the trend of s_i for each underlying model, while there's no similar pattern between $1/q_i$ and s_i in the case of default JPEG. To further verify the correlation between $1/q_i$ and s_i in the case of OptS, we extend our investigation to include the average Pearson correlation coefficient between the inverse of Q and S as sequences over the complete ImageNet validation set [56], presented in Table 3.2. The resulting correlation coefficients are quite close to one for both luminance and chrominance channels and for all tested DNN model. However, for JPEG, when we calculate the correlation between the inverse of the default JPEG quantization table and S, although the correlation coefficients for chrominance channels are quite close to one, those for luminance channel are much closer to zero, demonstrating weak correlation. Based on the above analysis, our claim at the beginning of this subsection is well-supported, highlighting the effectiveness of OptS and SWE.

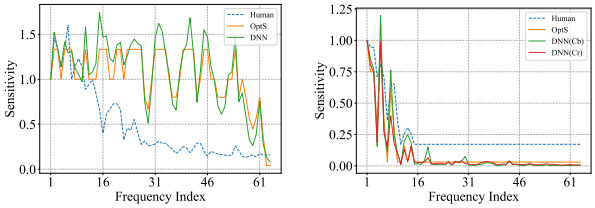
3.6 Experimental Results

Instead of the rate-distortion (R-D) performance for each image as in the HOC case, we evaluate the rate-accuracy (R-A) performance on the whole ImageNet ILSVRC 2012 validation set [56], which contains 50,000 images considered as raw images for compression. To evaluate the R-A performance of all coding methods, the average rate is computed as the average of the image-wise bit rate in bpp over the entire validation set, in conjunction with their corresponding top-1 accuracy. In the reported results for all coding methods, compression is applied to the pre-processed image, serving as the model's input layer. The rate per image is then calculated based on customized Huffman tables generated for the specific compression method.

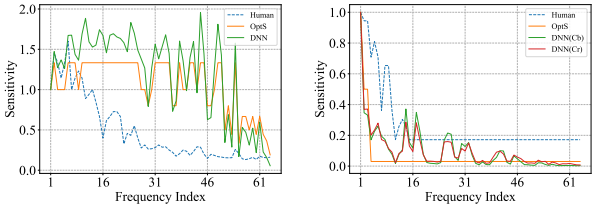
For rigorous testing, we evaluate both HDQ coding, denoted as SDQ off, and SDQ coding, denoted as SDQ on. When SDQ is activated, we will test the SDQ coding when incorporated by SWE, serving as a DOC, with its optimal quantization design by OptS. Besides, we will test the SDQ when incorporated by MSE, serving as a HOC, with its optimal quantization design by OptD. For consistency, we will also test



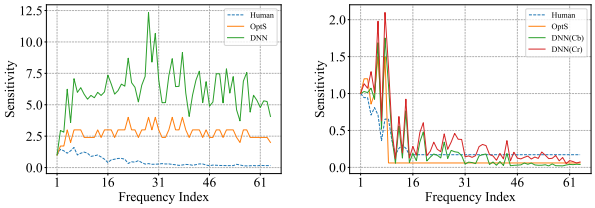
(a) MobileNetV2



(b) MnasNet



(c) RegNetY-400MF



(d) ConvNeXt-T

Figure 3.6: The figure shows the sensitivity of target DNNs with the quantization tables designed by OptS, together with default JPEG quantization tables for comparison. Note that it is $1/q_i$ shown in the figure instead of q_i. Also, all curves are normalized by their respective values of sensitivity and inverse q corresponding to DC for better visualization. The first column is for the luminance channel, while the second column is for the chrominance channel. We select $d = 1$ for OptS over all target DNNs.

the default JPEG quantization table when the SDQ is incorporated by MSE. All the previous quantization table design methods, including JPEG, OptD, and OptS, will also be evaluated as part of HDQ coding. Given that OptD, as proposed in [134], is not designed to accommodate colored channels, we extend its capabilities to accommodate color images, represented as a special case of OptS. This is done by utilizing Algorithm 2 with all s_i set to 1. OptD is presented as an ablation study to show the contribution of SWE alone.

For comparison within HDQ and SDQ coding frameworks, we select the baseline JPEG as our benchmark, as used in [50, 59, 60, 68, 125]. To provide a comparison with other DOC algorithms, we select DeepN-JPEG [68] as an example DOC benchmark since it is JPEG compliant. Note that although GRACE [125] is a DOC algorithm, it is not JPEG compliant.

3.6.1 Experimental Setup

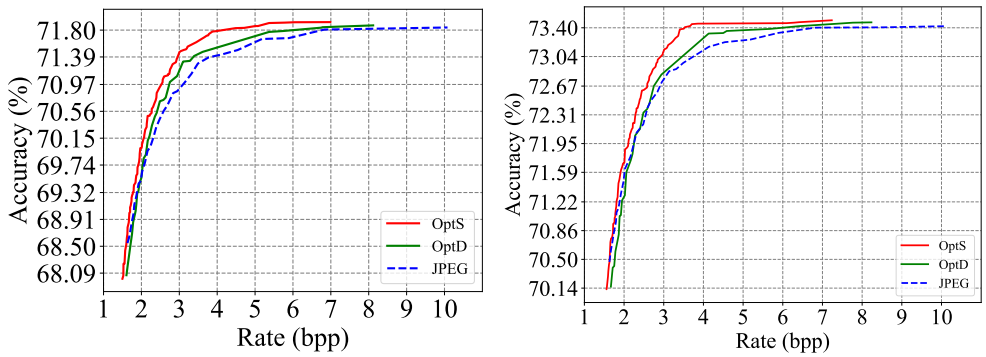
Tested models. Four prevailing pre-trained models from different architectural families, with some of their variants, namely MobileNetV2 [102], MnasNet [113], RegNetY-{400MF,800MF,2GF,6GF} [95], and ConvNeXt-{Tiny,Small,Base,Large} [67], are adopted as our target DNNs. Most of the models share the same preprocessing setup, with a resize size of 256 followed by a crop size of 224. However, ConvNeXts has customized resize sizes for its variants Tiny, Small, Base, and Large, which are 236, 230, 232, and 232, respectively, while maintaining the same crop size of 224. All pre-trained models are obtained from the Torchvision library [75].

Codecs setup. The quality of a compressed image is controlled by the quality factor (QF) in default JPEG, and by the water level d in OptS or OptD. In all experiments, q_{max} is selected to be 100 and color space sub-sampling is 4:4:4. For JPEG, QF is $\in [70, 100]$, for OptS and OptD $d \in [0.005, 15]$, and $\beta \in [0.005, 11]$ for the SDQ coding[6].

3.6.2 R-A Performance

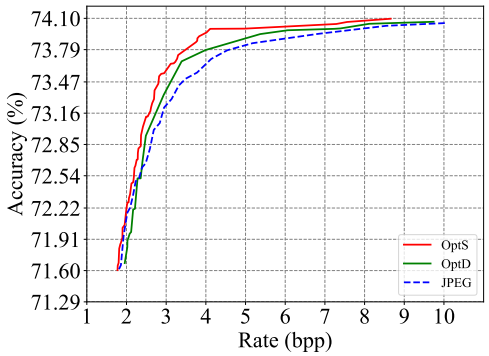
A straightforward experimental design, referred to as the one-to-all setup, involves using a fixed QF and water level d for Algorithms 1 and 2 across the entire ImageNet validation dataset. This approach is applied to both underlying coding frameworks, HDQ and SDQ. However, since this method does not control image-wise distortion, it often requires extensive manual tuning. To further address this limitation and highlight the advantage of OptS, we adopt a distortion-matching setup that enables a fairer comparison across compression algorithms. This setup is demonstrated exclusively for HDQ in a specific experiment, though it can be readily extended to SDQ.

[6] In the context of SDQ coding, the iterative algorithm runs for three iterations or until a convergence criterion is fulfilled.

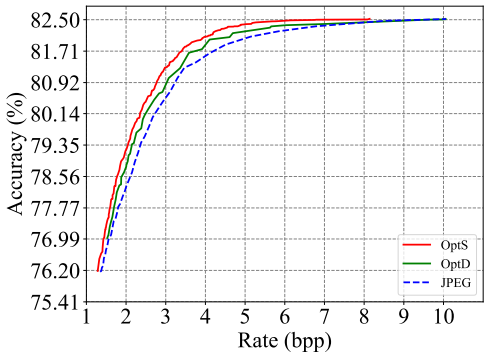


(a) MobileNetV2

(b) MnasNet

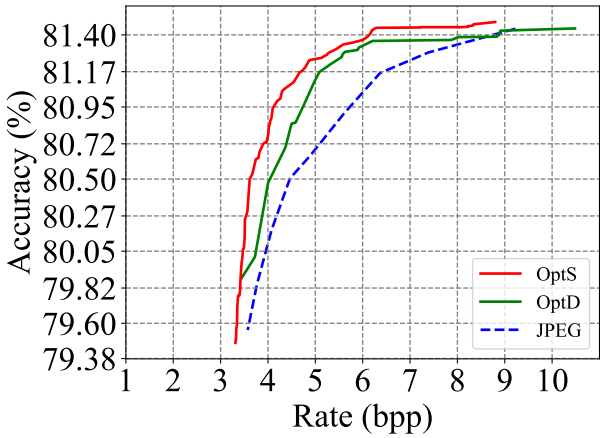


(c) RegNetY-400MF

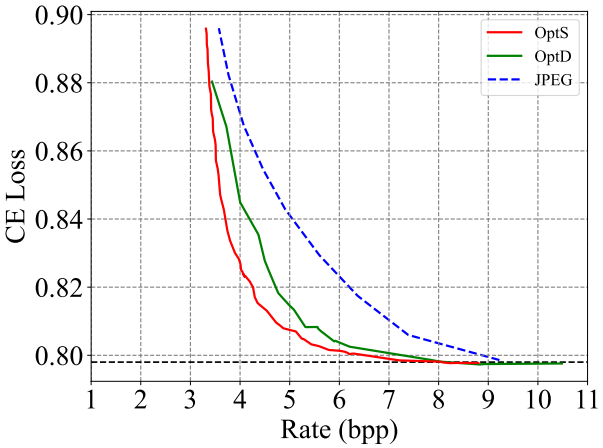


(d) ConvNeXt-T

Figure 3.7: Performance evaluation of R-A for four prominent models from different families with HDQ coding using quantization tables from OptS, OptD, and JPEG. It is worth mentioning that for MobileNetV2, MnasNet, RegNetX-400MF, and ConvNeXt-T, DeepN-JPEG yields 68.43%, 70.1%, 71.2%, and 78.4%, respectively, of accuracy with the rate of 7.9 bpp (its accuracy is low). Therefore, in order not to distort the curve display, these points are removed from the figures.

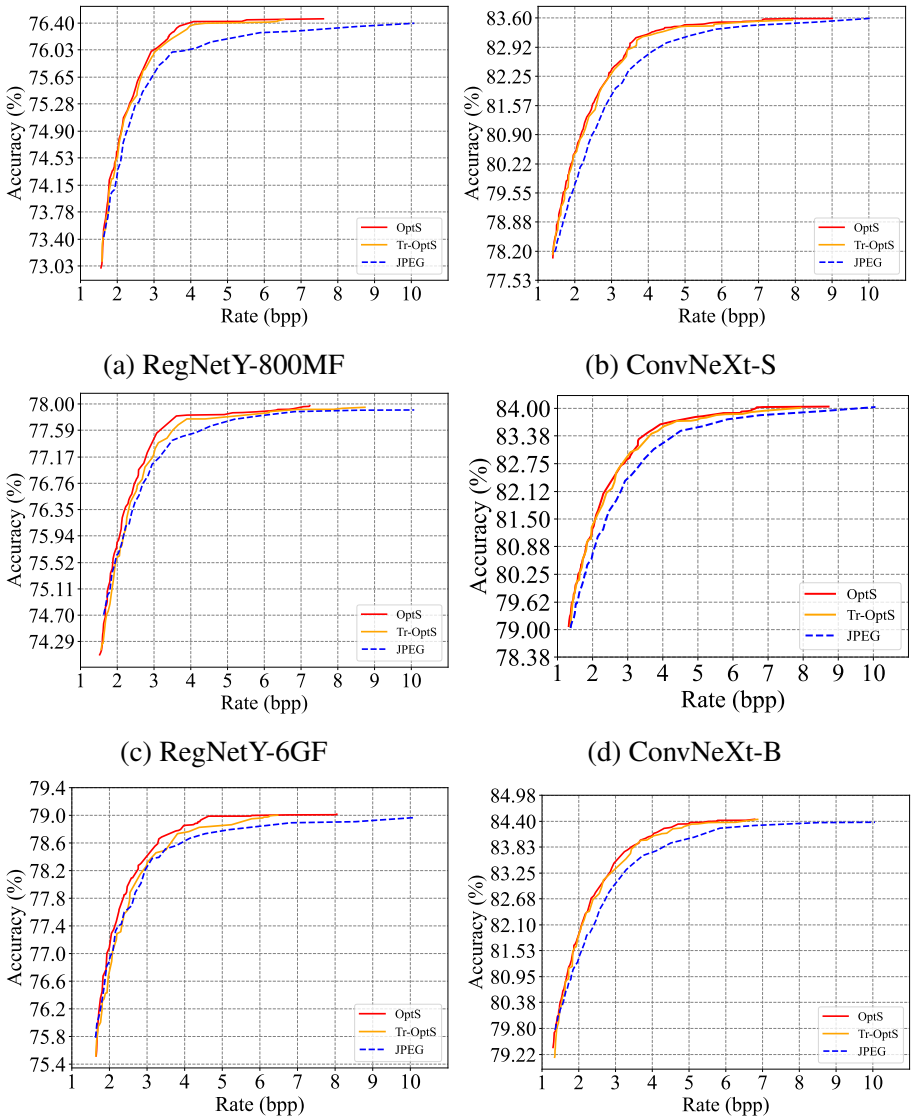


(a) R-A performance



(b) CE Loss

Figure 3.8: R-A performance and CE Loss of HDQ coding applied to the Swin-T transformer model using quantization tables provided by OptS, OptD, and JPEG.



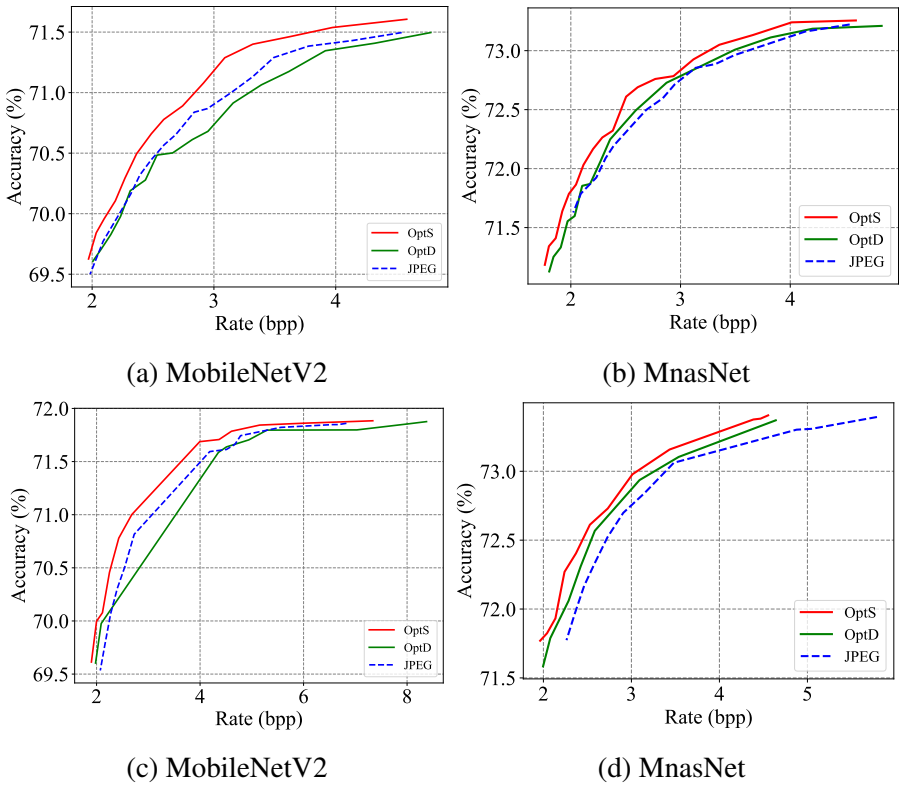
(a) RegNetY-800MF (b) ConvNeXt-S

(c) RegNetY-6GF (d) ConvNeXt-B

(e) RegNetY-2GF (f) ConvNeXt-L

Figure 3.9: R-A performance comparison among Tr-OptS, OptS, and JPEG, where Tr-OptS is designed for RegNet-400MF (ConvNeXt-T, resp.) and fixed when evaluated for each of other DNNs within the family of RegNets (ConvNeXts, resp.), and OptS is designed specifically for each of other DNNs within the family of RegNets (ConvNeXts, resp.).

HDQ Coding. Following the one-to-all setup, Fig. 3.7 presents the R-A performance of OptS, OptD, and JPEG within the HDQ framework, evaluated across four pre-trained CNN models and Swin-T as a



(a) MobileNetV2 (b) MnasNet

(c) MobileNetV2 (d) MnasNet

Figure 3.10: Evaluation of the R-A performance following the distortion-matching setup. Fig. 3.10a and 3.10b correspond to Setting 1; Fig. 3.10c and 3.10d correspond to Setting 2.

representative transformer-based model, shown in Fig. 3.8. Additionally, when we compare OptS with DeepN-JPEG, it is worth mentioning that for MobileNetV2, MnasNet, RegNetX-400MF, and ConvNeXt-T, DeepN-JPEG yields 68.43%, 70.1%, 71.2%, and 78.4%, respectively, of accuracy with a rate of 7.9 bpp, resulting in a relatively low accuracy. In order not to distort the curve display, these points are excluded from Fig. 3.7. As expected, OptS consistently outperforms both OptD and JPEG. This is because OptS tends to select a larger quantization step size q_i for a source C_i with a lower sensitivity s_i. Consequently, despite a notable increase in MSE, only a minor amount of SWE is introduced. This enables OptS to decrease the rate without inducing a substantial increase in DNN loss, leading to negligible accuracy degradation.

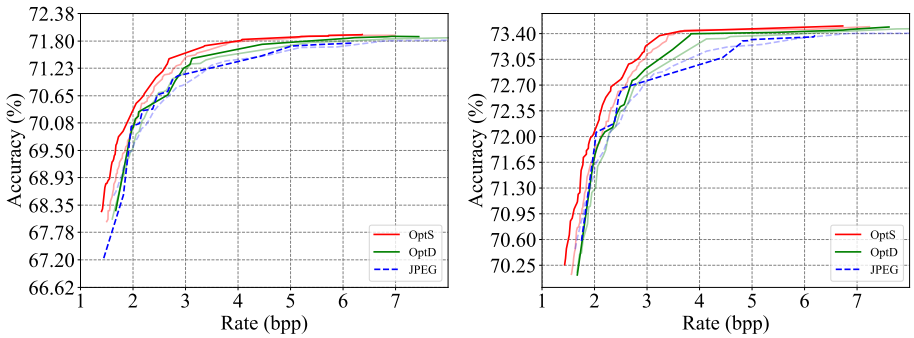
Evaluating sensitivity transferability in HDQ coding. From Fig. 3.5, we see that the sensitivity curves of DNNs within the same family resemble each other. We now want to evaluate the transferability of the sensitivity of one DNN to other DNNs within the same family in the context of HDQ coding with OptS. Specifically, take the families of RegNets and ConvNeXts as an example, where ConvNeXt-T and RegNetY-400MF are the shallowest models within their respective families. Consider the HDQ codec

with OptS designed specifically for ConvNeXt-T (RegNetY-400MF, resp.). Fix this codec and refer to it as Tr-OptS. We want to evaluate the R-A performance of Tr-OptS over other DNNs within the family of ConvNeXt-T (RegNetY-400MF, resp.), which is shown in Fig. 3.9. Also shown in Fig. 3.9 is the R-A performance of the HDQ codec with OptS designed specifically for each of other DNNs within the family of ConvNeXt-T (RegNetY-400MF, resp.) (simply referred to as OptS) along with the R-A performance of the default JPEG. From Fig. 3.9, it is clear that the R-A performance of Tr-OptS is almost the same as that of OptS for each of other DNNs within the family of ConvNeXt-T (RegNetY-400MF, resp.), and far better than that of the default JPEG, thus demonstrating the transferability of the sensitivity of ConvNeXt-T (RegNetY-400MF, resp.) to each of other DNNs within the family of ConvNeXt-T (RegNetY-400MF, resp.).

Distortion Matching in HDQ coding. In the distortion-matching setup, we design two experimental settings. *Setting 1:* Use a fixed QF for JPEG to compress all the images in the dataset, and record all the resulting SWEs for them. Then, for each image, adjust the image-adaptive d's in OptS and OptD to roughly match the recorded SWE using binary search. *Setting 2:* Use a fixed d for OptS to compress all the images in the dataset, and record all the resulting SWEs for them. Then, for each image, adjust the image-adaptive QF in JPEG and d in OptD to do the distortion matching again. For Setting 1, results are shown in Fig. 3.10a and 3.10b, with QF $\in [70, 98]$; for Setting 2, results are shown in Fig. 3.10c and 3.10d, with $d \in [0.005, 1]$. As expected, OptS always performs better than OptD and JPEG in both settings.

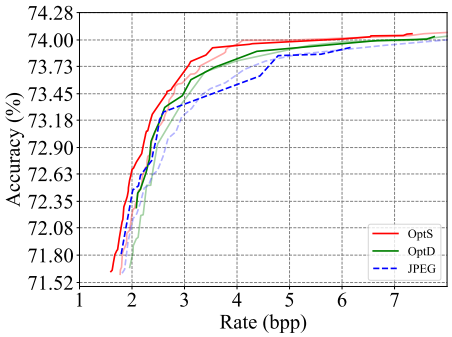
SDQ Coding. In Fig. 3.11, we evaluate our complete DOC framework which solves Problem equation 3.1 with OptS addressing the outer minimization and the revisited SDQ addressing the inner minimization. For comparison, we also show the results of original SDQ using quantization tables provided by OptD and JPEG following the one-to-all setup. Also, we include the HDQ results, visualized by transparent lines, as a reference. Clearly, our complete DOC framework provides an outstanding R-A performance compared to everything else. Additionally, some results shown in Fig. 3.11 are summarized in Table 3.3 to make comparison at a fixed rate or a fixed accuracy level.

To further demonstrate the effectiveness of the SWE as a good estimate of the change in the DNN training loss function, Fig. 3.12 shows the cross-entropy (CE) loss value corresponding to its R-A performance in Fig. 3.11 for both HDQ coding and SDQ coding. It is obvious that minimizing the SWE will reflect in the reduction of the CE loss for both HDQ coding and SDQ coding with OptS algorithm.

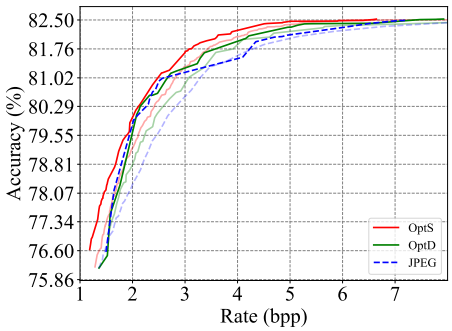


(a) MobileNetV2

(b) MnasNet

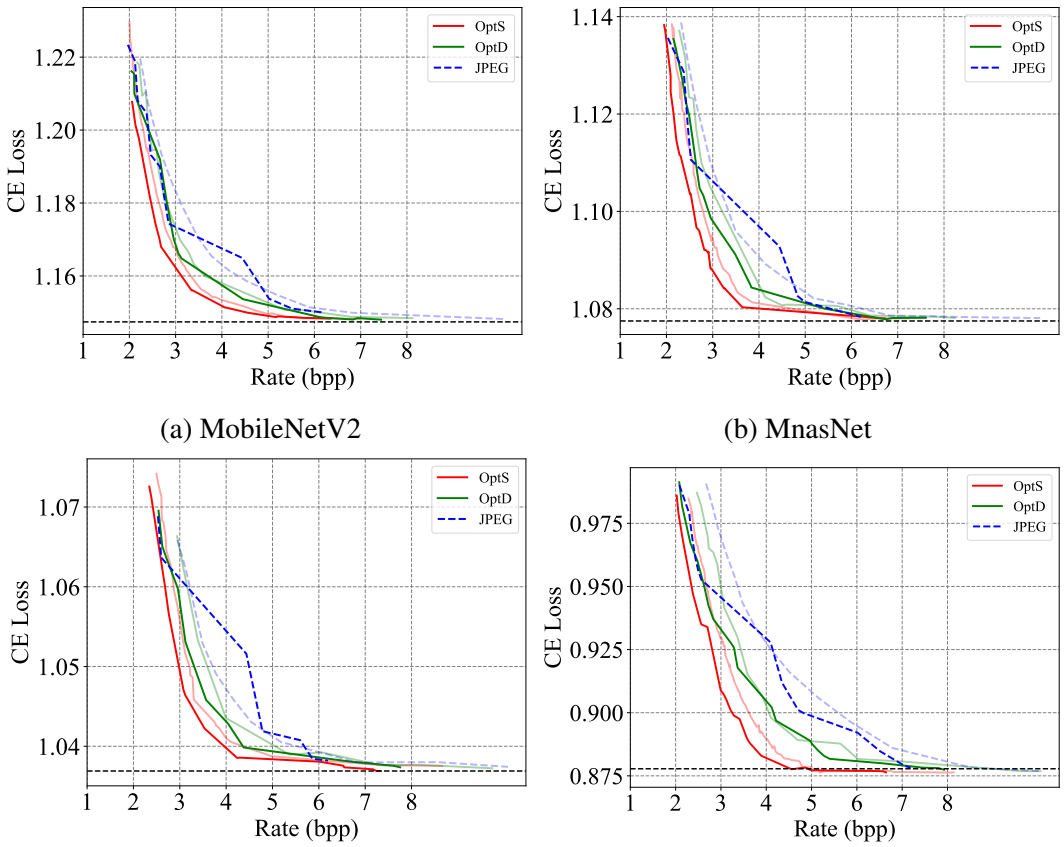


(c) RegNetY-400MF



(d) ConvNeXt-T

Figure 3.11: R-A performance of SDQ coding with quantization tables provided by OptS, OptD, and JPEG (HDQ results are presented by transparent lines as a reference).



(a) MobileNetV2

(b) MnasNet

(c) RegNetY-400MF

(d) ConvNeXt-T

Figure 3.12: This figure displays the CE Loss for Fig. 3.11 to illustrate that both HDQ and SDQ Coding, when OptS is used, converge faster towards the CE loss of the raw image, indicated by a black dotted horizontal line, compared to other methods.

Table 3.3: R-A performance summary, derived from Fig. 3.11, enables a comparison between SDQ and HDQ coding methods. This assessment is conducted with the underlying quantization table design by OptS, OptD, and JPEG, considering both the same rate and accuracy levels.

Model	HDQ						SDQ					
	JPEG		OptD		OptS		JPEG		OptD		OptS	
	Rate (bpp)	Acc (%)	Rate (bpp)	Acc (%)	Rate (bpp)	Acc (%)	Rate (bpp)	Acc (%)	Rate (bpp)	Acc (%)	Rate (bpp)	Acc (%)
MobileNetV2	1.67	68.63	1.71	68.53	1.73	69.24	1.82	68.56	1.67	68.24	1.72	69.79
	10.08	71.84	6.76	71.84	4.71	71.84	6.20	71.76	4.46	71.73	3.36	71.70
MnasNet	1.70	70.69	1.71	70.40	1.72	70.95	1.75	70.58	1.81	70.81	1.73	71.41
	10.08	73.42	6.40	73.42	3.66	73.42	6.20	73.38	5.43	73.41	3.24	73.38
RegNetY-400MF	1.90	71.65	1.96	71.67	2.02	72.28	2.01	72.46	2.08	72.29	2.02	72.69
	10.08	74.05	8.12	74.05	4.98	74	6.20	73.93	7.75	74.04	4.23	73.96
ConvNeXt-T	2.01	78.33	2.00	78.81	2.06	79.42	2.02	79.95	2.08	80.08	2.06	80.20
	10.08	82.51	9.63	82.50	6.14	82.48	7.19	82.49	7.47	82.50	4.91	82.45

Compression Helps. Interestingly, our experiments reveal that for both HDQ and SDQ, compression with OptS yields accuracy even slightly better than the inference accuracy of the raw dataset. This confirms the prediction made in [131] in practice. For HDQ coding, Table 3.4 presents the improvement in inference accuracy followed by a reduction in the CE value compared to the case of raw image inference. Moreover, the accuracy improvement is also presented in the case of SDQ, as shown in Table 3.5. Notably, the accuracy improvement for MobileNetV2 can reach up to 0.07%, where the corresponding CR is 3.77×.

The potential reason of this phenomenon is provided in [131], where authors declare that compression could possibly enhance the useful features and remove the disturbing features in an image; therefore, DNN models might make better decision on an image after it is compressed.

Table 3.4: Accuracy improvement and CE loss reduction are shown by using the compressed images obtained through HDQ coding with OptS algorithm, compared to the raw images.

Model	Raw		OptS	
	Acc (%)	Loss	Acc (%)	Loss
MobileNetV2	71.870	1.1474	**71.920**	1.1465
MnasNet	73.468	1.0775	**73.494**	1.0767
RegNetY-2GF	78.936	0.8205	**79.016**	0.8208
RegNetY-6GF	77.952	0.8601	**77.972**	0.8603
RegNetY-800MF	76.408	0.9184	**76.460**	0.9193
RegNetY-400MF	74.026	1.0369	**74.096**	1.0357
ConvNeXt-L	84.414	0.7757	**84.446**	0.7743
ConvNeXt-B	84.062	0.8603	84.040	0.8596
ConvNeXt-S	83.634	0.8568	83.586	0.8557
ConvNeXt-T	82.516	0.8778	**82.522**	0.8767

Table 3.5: Accuracy improvement demonstrated through HDQ and SDQ coding with OptS method compared to raw image.

Model	Raw	HDQ		SDQ	
	Acc (%)	Rate (bpp)	Acc (%)	Rate (bpp)	Acc (%)
MobileNetV2	71.870	6.73	<u>71.92</u>	6.36	**71.938**
MnasNet	73.468	7.23	<u>73.494</u>	6.73	**73.504**
RegNetY-400MF	74.026	8.66	**74.096**	7.33	<u>74.064</u>
ConvNeXt-T	82.516	8.14	**82.522**	6.64	**82.522**

3.7 Chapter Summary and Future Work

In this chapter, we revisit the JPEG rate distortion theory for DNN vision by proposing a novel distortion measure dubbed SWE. Simply replacing MSE distortion with SWE distortion for any DCT-based HOC algorithm will result in a DOC algorithm without incurring additional complexity. Following that, we adopt the SDQ framework in this chapter and convert it to its DOC counterpart by incorporating our SWE distortion measure. To further address the quantization table design problem for SDQ, we have proposed OptS algorithm to generate optimal quantization tables for the luminance channel and chrominance channels, respectively. Then, we rigorously evaluate our proposed DOC framework and algorithm with four prevalent DNN models on the entire ImageNet validation set, and obtain outstanding R-A performance.

Finally, it is worth noting that the proposal of SWE is based on Taylor expansion. Therefore, SWE may not be effective in the case where the perturbation ΔC is too large. As a result, addressing the DOC problem considering large DCT perturbation is left for future work.

Chapter 4

JPEG Inspired Deep Learning

This chapter introduces JPEG-DL, a novel deep learning framework that integrates a trainable JPEG compression layer into standard DNN architectures to enhance performance and robustness. Contrary to the conventional belief that lossy compression degrades model accuracy, JPEG-DL leverages a differentiable soft quantizer to replace the non-differentiable quantization step in JPEG, enabling end-to-end training of both the compression layer and the DNN. This unified architecture not only improves classification accuracy—achieving average gain of 7% on fine-grained datasets—but also enhances adversarial robustness. JPEG-DL is lightweight, adding only 128 trainable parameters, and is compatible with a wide range of models and tasks, demonstrating its scalability and effectiveness in optimizing deep learning pipelines through learned compression.

The conference version of this chapter is available in [99].

4.1 Introduction

JPEG compression [91] is the *defacto* lossy image compression technique with ubiquitous presence in real-world applications. With the development of deep learning, more and more images, potentially compressed by JPEG, are consumed by deep neural networks (DNNs). Naturally, it's of interest to study how JPEG compression will impact DNN performance for computer vision tasks, and extensive research has been conducted along this line such as [25, 68] and [125]. These initial explorations establish a widely accepted view that the information loss caused by JPEG obscures important features in the input image, thereby negatively impacting DNN performance.

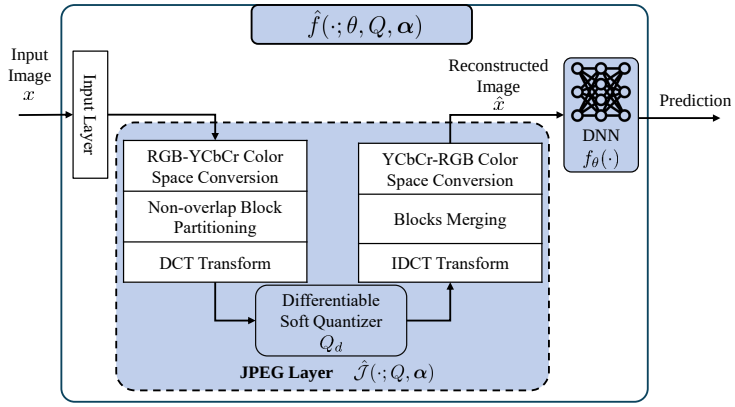
However, it was shown by [131] that the above conventional wisdom does not hold anymore if JPEG compression is applied intelligently and adaptively on a per-image basis. Indeed, [131] showed that if applied appropriately, JPEG compression can actually improve DNN performance, at least in theory. They

further proposed to train a DNN based on images with various JPEG quality levels. While they managed to demonstrate improved performance with a specially designed DNN topology, the model is too cumbersome to be fully trained, leading to suboptimal performance. Moreover, the adherence to the default JPEG quantization also limits the effectiveness of this method. On the other hand, given a fixed DNN model, [144] and [101] found that its performance could be slightly improved if the input images got compressed by JPEG with optimized quantization parameters. However, the performance gain is not significant due to the frozen DNN model. Although these existing works provide promising insights to improve DNN performance with JPEG compression, a solid solution that can fully unleash the potential of this idea has yet to be found.

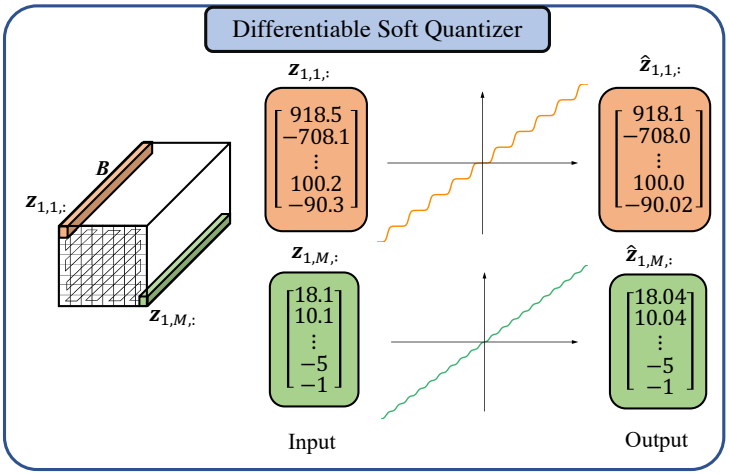
To address the above issue, in this chapter, we propose jointly optimizing both JPEG quantization operation and a DNN to achieve greater effectiveness. To this end, we first introduce a trainable JPEG compression layer, the structure of which is illustrated in Fig. 4.1a. This layer follows the standard JPEG pipeline to convert a preprocessed input image, after passing through the input layer, into blockwise Discrete Cosine Transform (DCT) coefficients in the YCbCr color space. A novel differentiable soft quantizer (Q_d) is then applied to quantize the DCT coefficients at each frequency position, followed by the standard JPEG inverse process to reconstruct the RGB image. We then present JPEG-DL, a novel deep learning (DL) framework that introduces a new DNN architecture by inserting a JPEG layer directly after the original input layer of any underlying DNN architecture. This layer can be considered an integral part of any underlying DNN architecture, forming a new unified DNN architecture, whose parameters get optimized jointly with DNN model weights during training. The core of JPEG-DL is Q_d as it substitutes the non-differentiable, hard quantization in JPEG with a differentiable, soft quantization operation defined by a neat analytical formula, which not only facilitates gradient-based optimization for quantization parameters but also introduces additional trainable non-linearity to the overall image understanding pipeline. To validate the effectiveness of JPEG-DL, we conducted extensive experiments for image classification on six datasets including four fine-grained classification datasets [51, 85, 87, 120], CIFAR-100 [55] and ImageNet [24]. Results show that JPEG-DL significantly and consistently outperforms the standard DL across various DNN architectures, with a negligible increase in model complexity. Specifically, JPEG-DL improves classification accuracy by up to 20.9% on some fine-grained classification dataset, while adding only 128 trainable parameters to the DL pipeline. Moreover, the superiority of JPEG-DL over the standard DL is further demonstrated by the enhanced adversarial robustness of the learned unified architecture.

The main contributions of this chapter can be summarized as follows:

- We introduce a novel trainable JPEG layer leveraging differentiable soft quantizers with nice analytical formulas.
- Based on the new JPEG layer, we propose a new DL framework dubbed JPEG-DL, which jointly optimizes the JPEG layer and the DNN model during training.
- The outstanding performance of JPEG-DL over the standard DL is verified by comprehensive experimental results on various image classification datasets across multiple DNN architectures and for a variety of tasks including adversarial defense.



(a)



(b)

Figure 4.1: (a) The JPEG-DL framework consists of a JPEG layer followed by a standard DNN, where the standard forward/inverse processes of JPEG depicted in white boxes are fixed and not trained. The JPEG pipeline receives a conventionally preprocessed input image $x \in \mathbb{R}^{3\times W\times H}$ sampled from the underlying task's dataset. The JPEG layer, equipped with a differentiable soft quantizer, and the underlying DNN form a unified new DNN architecture, which are shown in blue indicating that they are trainable components. (b) As an example, we show $\boldsymbol{z}_{1,:,:}$, i.e., the DCT representation of the Y channel of an image, by a tensor consisting of B blocks of DCT coefficients. Each 8×8 block contains $M = 64$ DCT frequencies, ordered from low to high in a zigzag manner. Then, we show how $\boldsymbol{z}_{1,1,:}$ and $\boldsymbol{z}_{1,M,:}$ are quantized by $Q_d(\cdot\,; q = 1, \alpha = 10)$ and $Q_d(\cdot\,; q = 0.5, \alpha = 16)$, respectively.

4.2 Background and Related Work

JPEG applications. JPEG was originally developed as a lossy image compression technique based on transform coding, which reduces image file sizes by generating their compact representations. Beyond its traditional use, JPEG has found numerous applications in deep learning: (1) it has been utilized as a data augmentation technique to improve robustness of DNNs against image compression [5]; (2) it has been employed as an empirical defense method against adversarial attacks, effectively reducing adversarial perturbations and enhancing the adversarial robustness of DNNs [23, 27, 38]; and (3) it has been integrated into the knowledge distillation [44] framework by [98], where it helps the teacher model transfer knowledge to the student model in a more effective way. In contrast, this chapter focuses on leveraging JPEG to enhance the natural performance, instead of the robust performance, of DNNs without relying on any teacher model.

Optimizing JPEG Compression for DNN vision. As a lossy image compression technique, JPEG is developed specifically for the human visual system. As a result, while the information loss introduced by JPEG is often imperceptible to humans, it can significantly degrade the performance of DNNs. This issue gives rise to a line of research which optimizes JPEG compression based on DNN perception. For instance, given a pretrained DNN, [125], [144] and [101] first derive its sensitivity to different DCT frequencies, based on which they customize JPEG quantization tables for this DNN to reach the optimal rate-accuracy tradeoff. Another popular direction is to make JPEG trainable, which integrates the JPEG encoder and DNN into an end-to-end differentiable training framework [71, 126]. These methods create differentiable proxies for image quantization and bitrate calculation, so that JPEG compression can be optimized via backpropagation in order to minimize a total loss considering both DNN performance and bitrate. However, all the above works focus on mitigating DNN performance degradation in the presence of JPEG compressed images, but offer little to no improvement on DNN performance when raw images are given as input. On the contrary, this chapter leverages JPEG purely as a tool to improve DNN performance on raw images, regardless of its compression capability.

Improving DNN performance with JPEG. Due to the aforementioned reason, JPEG compression generally hurts DNN performance. However, it's recently demonstrated by [131] that, with an oracle guiding the compression process, one can select an optimal quality level to compress each image, enabling the DNN to make its best possible prediction for it. By applying this adaptive JPEG compression across a set of images, one can actually improve the DNN prediction accuracy considerably. This phenomenon, termed "compression helps" in the original paper, is justified by the fact that compression can remove noise and disturbing background features, thereby highlighting the main object in an image, which helps DNNs make better prediction. Due to the need of ground truth labels in the compression stage, the above adaptive compression scheme is not realizable in most real world applications; however, the discovery of "compression helps" at least show the potential of using JPEG to improve DNN performance. Thus motivated, the authors in turn proposed an implementable way to improve DNN performance with JPEG. They built a new DNN topology that incorporates 11 parallel branches of an underlying pretrained model, with each branch obtaining as input either the raw image or its compressed version at a varying quality level. The penultimate layer representations from all branches are concatenated and fed into a classifica-

tion head, which is then trained together with those 11 model backbones as a unified DNN structure. This new DNN topology is clearly too complex to train, so the authors opted for partial training to mitigate the training complexity, therefore resulting in a suboptimal performance. In contrast, our proposed method introduces only 128 additional trainable parameters to the existing DL pipeline, causing negligible complexity increase. Moreover, it's noteworthy that our method is orthogonal to theirs, as our trainable JPEG quantization tables can be embedded into their framework to replace the default JPEG quantization tables, thereby further improving the performance.

Trainable Activation Functions. Relentless efforts have been made to search for activation functions that can improve DNN performance. Among all the directions, trainable activation functions have gained particular interest for their flexibility, expressive power, and adaptability during training. [11] propose the adjustable generalized hyperbolic tangent function, which extends the classic hyperbolic tangent function by introducing parameters to control the saturation level and slope of the function. [40] introduce the parametric ReLU (PReLU), a variant of ReLU with a trainable parameter that adjusts the negative part of ReLU. More recently, Kolmogorov-Arnold Networks (KANs) [69] employ trainable activation functions on edges, with nodes simply summing all the incoming activations. Interestingly, the differentiable soft quantizers in our JPEG layer can be interpreted as trainable activation functions whose input is the DCT representations of images. These trainable soft quantizers effectively introduce additional nonlinearity to DNN models, thus improving their expressive power.

4.3 JPEG-DL: JPEG Inspired DL

4.3.1 Problem Formulation

In the JPEG pipeline, an RGB image $x \in \mathbb{R}^{3\times W\times H}$ is first converted to the YCbCr color space and then partitioned into B non-overlapping 8×8 blocks, where DCT is applied to each of these blocks to obtain the corresponding DCT coefficients. For each color channel, the DCT coefficients in each block are then flattened following the zigzag order, resulting in $M = 64$ frequency positions ordered from low frequency to high frequency. Therefore, we denote the DCT coefficients of the image x as $\boldsymbol{z} = [z_{l,m,n}]$, where $l = 1, 2, 3$ corresponds to the color channel Y, Cb and Cr respectively, $1 \leq m \leq M$ is the index for frequency position, and $1 \leq n \leq B$ is the index for block. Up to this point, all operations involved are differentiable. Next, quantization tables $Q_Y = [q_1, q_2, \ldots, q_M]$ and $Q_C = [q_{M+1}, q_{M+2}, \ldots, q_{2M}]$ are used for the luminance (Y) and chrominance (CbCr) channels respectively to quantize their DCT coefficients. Following uniform quantization, we obtain quantized DCT coefficients $\hat{z}_{l,m,n} = \lfloor z_{l,m,n}/q_m \rceil \cdot q_m$ for $l = 1$, and $\hat{z}_{l,m,n} = \lfloor z_{l,m,n}/q_{M+m} \rceil \cdot q_{M+m}$ for $l = 2, 3$. Note that quantization is non-differentiable due to the use of the rounding operation. Finally, inverse operations including blockwise inverse DCT (IDCT) transform, blocks merging and YCbCr-to-RGB color space conversion are conducted over $\hat{\boldsymbol{z}}$ sequentially to obtain the reconstructed RGB image $\hat{x}$. Similar to their forward counterparts, all these inverse operations are differentiable. Denoting the composition of all the above operations as $\mathcal{J}$, we then have

$\hat{x} = \mathcal{J}(x; Q)$, where $Q = (Q_Y, Q_C)$. Hereafter, the mapping $\mathcal{J}$ stands for the JPEG encoding-decoding operation.

In supervised learning, each $x \in \mathcal{X}$ corresponds to a ground truth label $y \in \mathcal{Y}$. Let f_θ represent a DNN model with trainable weights θ, and let $\mathcal{L}$ denote the loss function used to train this DNN. In standard DL, the primary objective is to solve the following minimization problem:

$$\min_{\theta} \ \mathbb{E}[\mathcal{L}(f_\theta(x), y)]. \tag{4.1}$$

In contrast, JPEG-DL tries to improve the performance of DNN by jointly training it with the JPEG operation. As a result, the formulation should be instead:

$$\min_{\theta, Q} \ \mathbb{E}[\mathcal{L}(f_\theta(\mathcal{J}(x; Q)), y)]. \tag{4.2}$$

However, in order to solve (4.2) with gradient descent, the key challenge is caused by the non-differentiable quantization operation, which makes the gradients w.r.t. Q almost zero everywhere. To address this issue, we will introduce a *differentiable soft quantizer* (Q_d) in the next subsection, replacing the uniform quantizer (Q_u) used in $\mathcal{J}$.

4.3.2 Differentiable Soft Quantizer

Denote the index set of uniform quantization as

$$\mathcal{A} = \{-L, -L+1, \ldots, 0, \ldots, L-1, L\}. \tag{4.3}$$

For convenience, $\mathcal{A}$ is also regarded as a vector of length $2L+1$. Multiplying $\mathcal{A}$ with a quantization step size q, we get the corresponding reconstruction space

$$\hat{\mathcal{A}} = q \times [-L, -L+1, \ldots, 0, \ldots, L-1, L]. \tag{4.4}$$

Again, we will regard $\hat{\mathcal{A}}$ as both a vector and a set.

To randomly quantize a DCT coefficient z to an element in $\hat{\mathcal{A}}$, we invoke from [132] a trainable conditional probability mass function (CPMF) $P_\alpha(\cdot|z)$ over the reconstruction space $\hat{\mathcal{A}}$ or equivalently the index set $\mathcal{A}$ given z, where $\alpha > 0$ is a trainable parameter:

$$P_\alpha(iq|z) = \frac{e^{-\alpha(z-iq)^2}}{\sum_{j\in\mathcal{A}} e^{-\alpha(z-jq)^2}}, \ \forall i \in \mathcal{A}. \tag{4.5}$$

Extend z to a vector of length $2L+1$, i.e., $[z]_{2L+1} = [\overbrace{z, \ldots, z}^{2L+1 \text{ times}}]$. Then, the CPMF $P_\alpha(\cdot|z)$, regarded as a vector of length $2L+1$, can be easily computed via the softmax operation $\sigma(\cdot)$:

$$\big[P_\alpha(\cdot|z)\big]_{2L+1} = \sigma\Big(-\alpha \times \Big(\big[z\big]_{2L+1} - \hat{\mathcal{A}}\Big)^2\Big). \tag{4.6}$$

With the CPMF $P_\alpha(\cdot|z)$, z is now quantized to each $iq \in \hat{\mathcal{A}}$ with probability $P_\alpha(iq|z)$. Note that as $\alpha \to \infty$, $P_\alpha(\cdot|z)$ approaches an one-hot vector with probability 1 at the nearest point to z in $\hat{\mathcal{A}}$ and 0 elsewhere. Therefore, the resulting random quantizer effectively functions as the deterministic uniform quantizer $Q_u(z) = \lfloor z/q \rceil \cdot q$.

Based on the CPMF $P_\alpha(\cdot|z)$, we can now define a differentiable soft quantizer Q_d as the conditional expectation of iq given z, i.e.,

$$Q_d(z) = \mathbb{E}[iq|z] = \sum_{i \in \mathcal{A}} P_\alpha(iq|z) \cdot iq. \tag{4.7}$$

Similarly, as $\alpha \to \infty$, Q_d also goes to Q_u. Fig. 4.2 shows how the shape of Q_d varies w.r.t α, given a fixed q.

This soft quantizer Q_d serves as an analytical proxy for Q_u. It's differentiable everywhere, allowing gradients to flow through it smoothly. More importantly, compared to Q_u, Q_d involves a trainable parameter α which can adjust the softness of the quantizer, thereby introducing more flexibility. In view of these nice properties, Q_d is the ideal candidate in place of Q_u used in $\mathcal{J}$.

As a side note, for the reader who is not familiar with quantization, but familiar with the attention operation used in transformer models [119], Q_d can be regarded as an attention operation in broad sense, with the query being z, the key and value being $\hat{\mathcal{A}}$, and the similarity metric between the query and key being negative squared distance instead of dot product.

4.3.3 Overall Framework of JPEG-DL

Substituting Q_u in $\mathcal{J}$ with Q_d, we get a differentiable JPEG layer $\hat{\mathcal{J}}$ parameterized by Q and $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_Y, \boldsymbol{\alpha}_C)$. $\boldsymbol{\alpha}_Y = [\alpha_1, \alpha_2, \ldots, \alpha_M]$ and $\boldsymbol{\alpha}_C = [\alpha_{M+1}, \alpha_{M+2}, \ldots, \alpha_{2M}]$ are α tables for the luminance and chrominance channels respectively, used in conjunction with Q_Y and Q_C to quantize DCT coefficients. Following the proposed soft quantization, we obtain quantized DCT coefficients $\hat{z}_{l,m,n} = Q_d(z_{l,m,n}; q_m, \alpha_m)$ for $l = 1$, and $\hat{z}_{l,m,n} = Q_d(z_{l,m,n}; q_{M+m}, \alpha_{M+m})$ for $l = 2, 3$, where $Q_d(z; q, \alpha)$ denotes a differentiable soft quantizer defined in (4.7) parameterized by a quantization step q and a scaling factor α. Overall, for an input image x, we have $\hat{x} = \hat{\mathcal{J}}(x; Q, \boldsymbol{\alpha})$. Therefore, we can rewrite (4.2), the JPEG-DL formulation, as

$$\min_{\theta, Q, \boldsymbol{\alpha}} \ \mathbb{E}[\mathcal{L}(f_\theta(\hat{\mathcal{J}}(x; Q, \boldsymbol{\alpha})), y)], \tag{4.8}$$

where the expectation can be approximated by the empirical mean over a mini-batch in actual training. Thanks to the use of Q_d, (4.8) can now be solved by gradient descent with ease. The derivation will be shown for the derivative analytical formulas in Section 4.3.4.

After training with JPEG-DL, we get the optimized parameters θ^*, Q^* and $\boldsymbol{\alpha}^*$. Then, we consider the composition of the JPEG layer and the underlying DNN as a unified DNN model $\hat{f}(x; \theta^*, Q^*, \boldsymbol{\alpha}^*) = f_{\theta^*}(\hat{\mathcal{J}}(x; Q^*, \boldsymbol{\alpha}^*))$ to do validation. Concretely, any raw input x should be fed into $\hat{\mathcal{J}}$, instead of directly

to f_{θ^*}, thus allowing the reconstructed image $\hat{x}$ to be fed into the underlying DNN f_{θ^*}. In other words, the JPEG layer is prepended as the first layer of any underlying DNN.

To conclude this section, we refer readers to Fig. 4.1a and Fig. 4.1b for an illustration of the inner workings of the JPEG-DL framework.

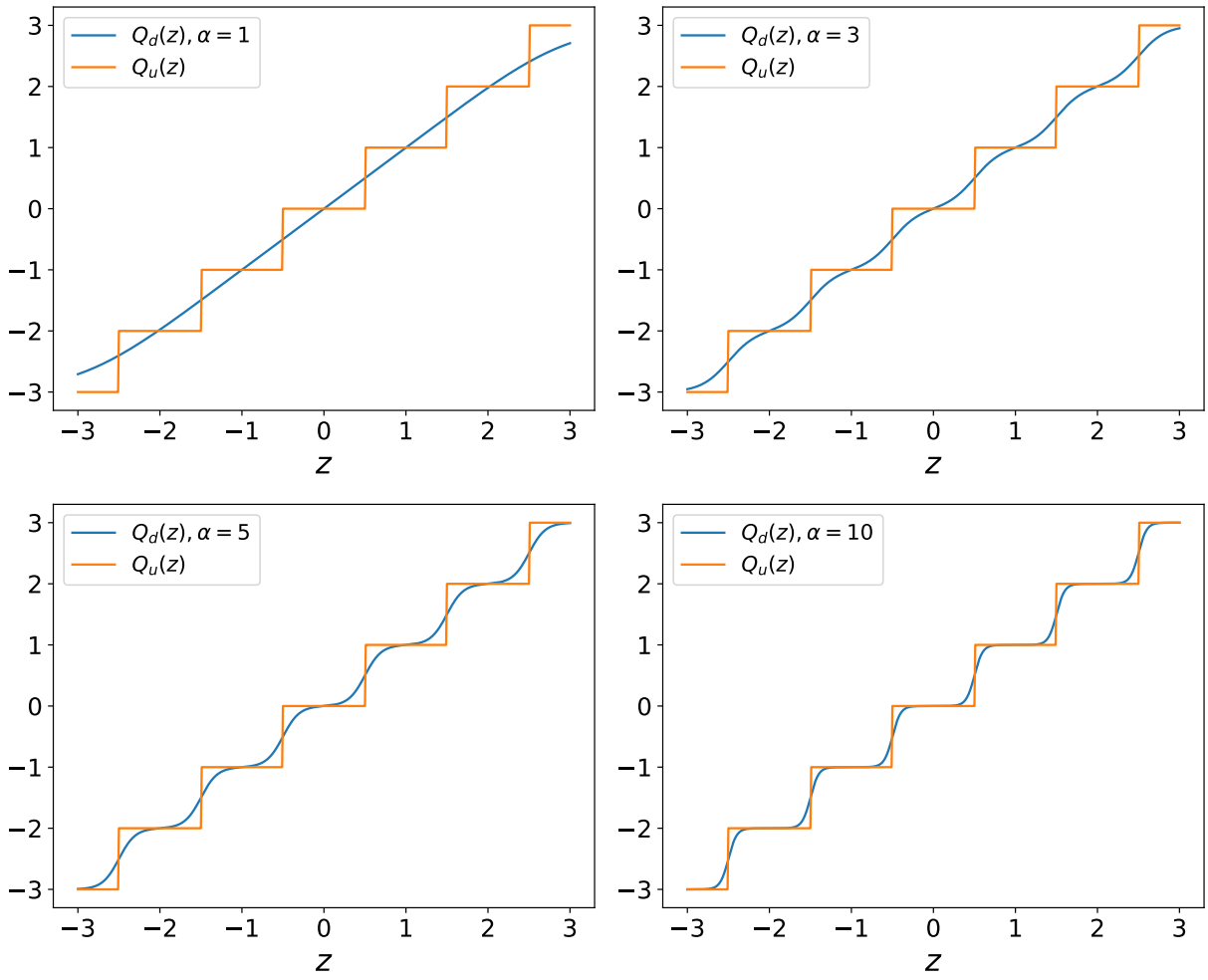


Figure 4.2: Illustration of Q_u vs. Q_d with $\alpha = 1, 3, 5, 10$, where L and q are set to 3 and 1, respectively.

4.3.4 Partial Derivatives of Q_d

With the CPMF $P_\alpha(\cdot|z)$ shown in (4.6), z is now quantized to each $\hat{z} \in \hat{\mathcal{A}}$ with probability $P_\alpha(\hat{z}|z)$. Denote this random mapping by

$$\hat{z} = Q_{\mathrm{p}}(z). \tag{4.9}$$

Note that given z, $\hat{z}$ is a random variable taking values in $\hat{\mathcal{A}}$ with distribution $P_\alpha(\cdot|z)$.

Quantization Step and Input

The partial derivatives of $Q_d(z)$ w.r.t. q and z are obtained as

$$\frac{\partial Q_d(z)}{\partial z} = 2\alpha \mathrm{Var}\big\{Q_p(z)\big\}, \tag{4.10a}$$

$$\frac{\partial Q_d(z)}{\partial q} = \frac{1}{q}\Big(\mathbb{E}\big\{Q_p(z)\big\} + (2\alpha z)\mathrm{Var}\big\{Q_p(z)\big\} - (2\alpha)\mathrm{Skew}_{\mathrm{u}}\big\{Q_p(z)\big\}\Big), \tag{4.10b}$$

where for any random variable V,

$$\mathrm{Skew}_{\mathrm{u}}(V) \triangleq \sum_v v^3 \mathbb{P}_V(v) - \Big(\sum_v v \mathbb{P}_V(v)\Big)\Big(\sum_v v^2 \mathbb{P}_V(v)\Big).$$

Scaling Factor α

The partial derivatives of $Q_d(Z)$ w.r.t. α obtained as

$$\begin{aligned}\frac{\partial Q_d(z)}{\partial \alpha} &= \frac{\partial \sum_{i\in\mathcal{A}} iq P_\alpha(iq|z)}{\partial \alpha} \\ &= \sum_{i\in\mathcal{A}} iq \frac{\partial P_\alpha(iq|z)}{\partial \alpha}\end{aligned} \tag{4.11}$$

$$\begin{aligned}\frac{\partial P_\alpha(iq|z)}{\partial \alpha} &= \frac{-(z-iq)^2 e^{-\alpha(z-iq)^2}}{\sum_{j\in\mathcal{A}} e^{-\alpha(z-jq)^2}} + \frac{e^{-\alpha(z-iq)^2}\sum_{j\in\mathcal{A}}(z-jq)^2 e^{-\alpha(z-jq)^2}}{(\sum_{j\in\mathcal{A}} e^{-\alpha(z-jq)^2})^2} \\ &= P_\alpha(iq|z)\sum_{j\in\mathcal{A}}(z-jq)^2 P_\alpha(jq|z) - (z-iq)^2 P_\alpha(iq|z)\end{aligned} \tag{4.12}$$

Plugging (2) in (1) yields

$$\begin{aligned}\frac{\partial Q_d(z)}{\partial \alpha} &= \sum_{i\in\mathcal{A}} iqP_\alpha(iq|z) \sum_{j\in\mathcal{A}} (z-jq)^2 P_\alpha(jq|z) - \sum_{i\in\mathcal{A}} iq(z-iq)^2 P_\alpha(iq|z) \\ &= \mathbb{E}\{Q_p(z)\}\mathbb{E}\{(z-Q_p(z))^2\} - \mathbb{E}\{Q_p(z)(z-Q_p(z))^2\} \\ &= -Cov\{Q_p(z),(z-Q_p(z))^2\} \end{aligned} \tag{4.13}$$

To gain a better understanding, we fix q and b, and analyzed how $\frac{\partial Q_d(z)}{\partial \alpha}$ behaves for different values of α. From Fig. 4.3, it is evident that the gradient magnitude decreases as α increases. This indicates that Q_d approaches the shape of the uniform quantizer Q_u, as shown previously in Fig. 4.2, leading to a reduction in the gradient magnitude for α. As a result, when α becomes sufficiently large, the gradient approaches zero, effectively preventing further updates to α, making it act as a non-trainable parameter at a certain point of the training process. Based on this observation and verified experimental results, we opt not to train α for simple image classification tasks like CIFAR-100 and fine-grained datasets, as it does not impact the overall performance.

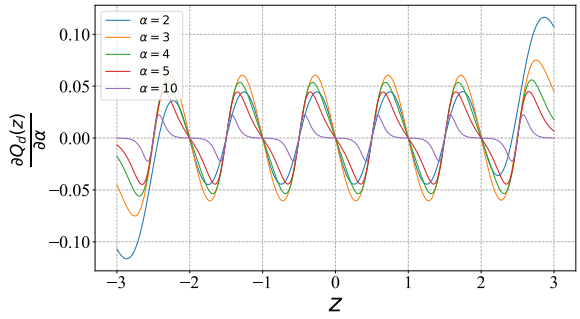
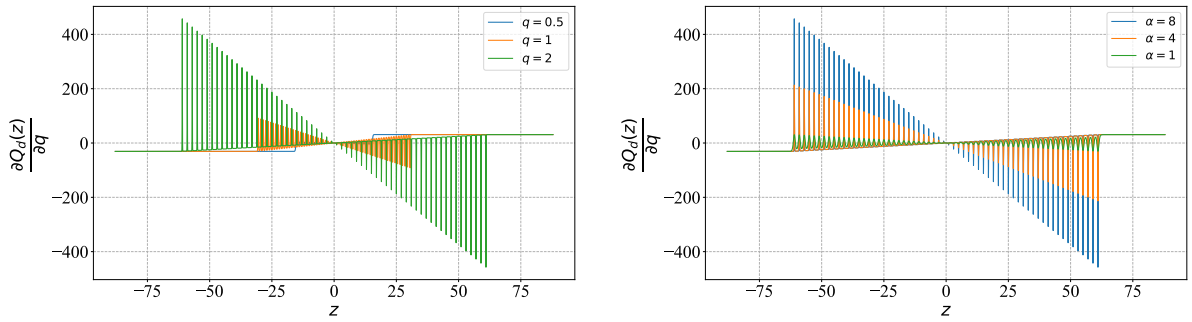


Figure 4.3: This Figure illustrates the partial derivatives of $Q_d(z)$ w.r.t. the scaling factor α, where $b = 3$ and $q = 1$.

Gradient Scaling

Figures 4.4a and 4.4b demonstrate the impact of varying α and q on the gradient magnitude of $Q_d(z)$ with respect to q. As observed, decreasing the value of q while keeping α constant leads to a decrease in the gradient magnitude. Similarly, decreasing the value of α while maintaining a fixed q also results in a reduction of the gradient magnitude. To address the instability in updating the trainable parameters Q, we propose utilizing α to regulate the magnitude of $\frac{\partial Q_d(z)}{\partial q}$, as illustrated in Figure 4.4b. By leveraging this control mechanism of adjusting α, we can stabilize the magnitude of the gradients that update q. This behavior suggests a relationship between q and α in controlling the gradient magnitude of $Q_d(z)$.

To explore the relationship between α and q, we refer to the exponent in (4.5), $\alpha(z - iq)^2$, which can be rewritten by expressing $z = cq$, resulting in the form $\alpha q^2(c - i)^2$. From this, we define a new term, $\hbar = \alpha q^2$, referred to as the *Gradient Scaling Constant*. To further illustrate this relationship, in Fig. 4.5, we set $\hbar = 2$, and by selecting different pairs of α and q values, we demonstrate that the maximum magnitude of $\frac{\partial Q_d(z)}{\partial q}$ remains invariant. This confirms that the gradient magnitude can be effectively controlled by adjusting α based on the last updated value of q, according to the specified $\hbar$ value. This adjustment allows for controlled quantization updates, reducing the potential instability during training. Moreover, by controlling the gradient magnitude, we simplify the optimization process, enabling the use of a single learning rate for all q values by using the SGD optimizer, instead of the ADAM optimizer. This gradient scaling mechanism is analogous to the ADAM optimizer, which adapts different learning rates for individual trainable parameters based on momentum and recent gradient magnitudes.



(a) (b)

Figure 4.4: Figures 4.4a and 4.4b illustrate the partial derivatives of $Q_d(z)$ w.r.t. the parameter q for cases where $\alpha = 8$ and q varies, and α varies and $q = 2$, respectively. For both figures, we set $b = 6$.

4.4 Experiments

4.4.1 Experimental Setup

Datasets

ImageNet-1K. For all experiments on this dataset, we utilize the standard training recipes shown by [89] without any modifications. We use SqueezeNet [49], ResNet-18 and ResNet-34 as our testing underlying models.

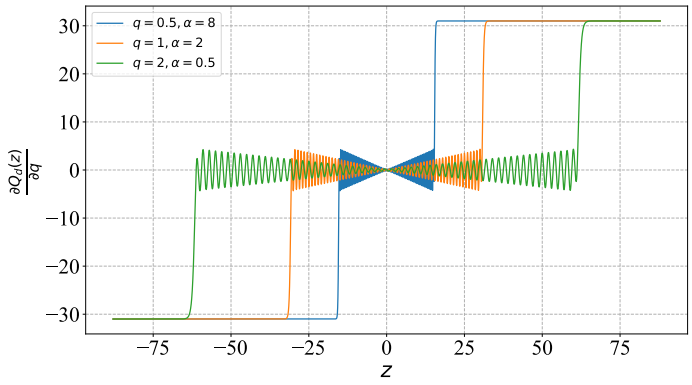


Figure 4.5: Demonstrating impact of $\hbar$ on gradient magnitude for various combinations of q and α. We fixed $\hbar$=2 and b=6.

CIFAR-100. We evaluate our proposed method using a transformer-based architecture and four state-of-the-art convolutional neural networks (CNNs): EfficientFormer-L1 [63], ResNet [41], VGG [104], MobileNet [102], and ShuffleNet [73]. For ResNet, we employ CIFAR-specific versions: ResNet32, ResNet56, and ResNet110. For VGG, we utilize VGG8 and VGG13. All CNN architectures follow the training recipe from CRD [114], while for the transformer-based architecture, EfficientFormer-L1, we adhere to the setup proposed by [127].

Fine-grained Tasks. These tasks involve visually similar classes and typically feature fewer training samples per class compared to conventional classification tasks. We evaluate our method on four datasets: CUB-200-2011 [120], Stanford Dogs [51], Flowers [85], and Pets [87]. For CNN architectures, we employ PreAct ResNet-18 [41] and DenseNet-BC [46], following the experimental setup and architecture modifications of [142]. For the transformer-based architecture, we use EfficientFormer-L1, adopting the setup outlined by [127].

Trained Model Setups

CNN-based Architectures Setting. For CIFAR100, we deploy a stochastic gradient descent (SGD) optimizer with a momentum of 0.9, a weight decay of 0.0005, and a batch size of 64. We initialize the learning rate as 0.05, and decay it by 0.1 every 30 epochs after the first 150 epochs until the last 240 epochs. For MobileNetV2, ShuffleNetV1 and ShuffleNetV2, we use a learning rate of 0.01 as this learning rate is optimal for these models in a grid search, while 0.05 is optimal for other models. For fine-grained classification tasks, all networks are trained from scratch and optimized by SGD with a momentum of 0.9, weight decay of 0.0001, and an initial learning rate of 0.1. The learning rate is divided by 10 after epochs 100 and 150 for all datasets, and the total epochs are 200. We set batch size 32 for these fine-grained classification tasks. We use the standard data augmentation technique for ImageNet [24], *i.e.*, flipping and

random cropping. This experimental setup is also outlined by [139]. For Fine-grained datasets, we use standard ResNet-18 with 64 filters and DenseNet-121 with a growth rate of 32 for image size 224×224. For fine-grained classification tasks, we use PreAct ResNet-18 [41], which modifies the first convolutional layer with kernel size 3×3, strides 1 and padding 1, instead of the kernel size 7×7, strides 2 and padding 3, for image size 32×32 by following [142]. We use the DenseNet-BC structure [46], and the first convolution layer of the network is also modified in the same way as in PreAct ResNet-18 for image size 32×32.

Transformer-based Settings. In this section, we compare the performance of JPEG-DL compared to its baseline using the EfficientFormer-L1 model [63] on CIFAR-100, as well as two fine-grained datasets, Flowers and Pets. We followed the experimental setup described by [127], adhering to the same configurations mentioned in Table 4.1. The learning rate was set to 0.003 for CIFAR-100 and 0.005 for the fine-grained tasks.

Table 4.1: Hyper-parameter setting on EfficientFormer-L1.

Settings	CIFAR-100	Pets	Flowers
batch size	512	512	512
warmup epochs	50	100	100
training epochs	300	600	600

JPEG-layer Settings

CIFAR-100 and fine-grained tasks. To train our JPEG layer, we first study the gradient nature of Q_d with respect to q and α in (4.5) by changing one parameter and fixing the other. Both q and α are sharable parameters among different numbers of blocks per image. During the calculation of the gradient of Q_d with respect to α, our analysis shows that when α is sufficiently large, such that Q_d is not too far from Q_u, the gradient magnitude for α is almost zero, as shown in Section 4.3.4. This indicates that $\boldsymbol{\alpha}$ will not be updated effectively if initialized within a reasonable range. Experiments also confirmed that making $\boldsymbol{\alpha}$ trainable does not significantly impact model performance. As a result, we choose not to train $\boldsymbol{\alpha}$ in our framework. However, when we explore the calculation of the gradient of Q_d with respect to q, the accumulated gradient received from different blocks per image during backpropagation results in unstable gradient magnitudes at a fixed value of α, as shown in our analysis in Section 4.3.4. To address this, we use the ADAM optimizer, which adapts the learning rate for each trainable parameter q in our JPEG layer, enabling more efficient training and better convergence. For CIFAR-100, we set the learning rate to 0.003 across all tested models. For the fine-grained datasets, we set the JPEG learning rate to 0.005. Across these datasets, we fix $\alpha_m = 5$ for all $1 \leq m \leq 2M$, and set $L = 2^{b-1}$ in (4.3), where b is a tunable hyperparameter set to 8.

ImageNet-1K. We will utilize specific settings to control the gradient magnitude to ensure more stable updates for Q. We define the Gradient Scaling Constants $\hbar_m = \alpha_m q_m^2$, $1 \leq m \leq 2M$, which allows us

to control the magnitude of gradients w.r.t Q. Specifically, we fix $\hbar_m = 0.7$ for all $1 \leq m \leq 2M$. During training, we update the value of α_m to be $\hbar_m / q_m^2$ before calculating the gradients w.r.t Q. As a result of controlling the maximum gradient magnitude, we can optimize Q using an SGD optimizer with 0.5 learning rate, instead of an ADAM optimizer. This approach ensures stable and efficient training for quantization table updates (see Subsection 4.3.4). For this dataset, b is equal to 11 across all tested models.

Quantization Table Initialization. For CIFAR-100 and fine-grained tasks, we initialize the quantization table Q based on the reciprocal of sensitivity for each DCT frequency, given a pre-trained model of the underlying DNN architecture, following the approach described by [101, 144]. Broadly speaking, the sensitivity of a frequency indicates the rate of change of the loss function w.r.t. the perturbation on this frequency, so we favor a smaller quantization step for a more sensitive frequency to limit the distortion amount on it. This sensitivity-based initialization is applied only to CNN architectures. For transformer-based models, we found that initializing all quantization steps to one—corresponding to the highest JPEG quality—consistently yielded the best performance. For ImageNet-1K, we adopt the strategy from [29], where the initialization of q_m is based on the average of absolute values of DCT coefficients across all blocks in all training images that will be quantized by q_m. Specifically,

$$q_m = \frac{2 \sum_{k=1}^{N} \sum_{n=1}^{B} \left| z_{1,m,n}^{(k)} \right|}{NB\sqrt{2^{b-1}}} \quad \text{for } 1 \leq m \leq M \quad \text{(i.e., } Q_Y\text{)},$$

and

$$q_m = \frac{\sum_{k=1}^{N} \sum_{l=2}^{3} \sum_{n=1}^{B} \left| z_{l,m,n}^{(k)} \right|}{NB\sqrt{2^{b-1}}} \quad \text{for } M+1 \leq m \leq 2M \quad \text{(i.e., } Q_C\text{)},$$

where k is the index for the image, and N is the number of training images.

Table 4.2: Top-1 validation accuracy (%) for Baseline and JPEG-DL on CIFAR-100. The Baseline results are from [114]. For JPEG-DL, we report the mean and standard deviation of experimental results over three runs.

Method	Res32	Res56	Res110	VGG8	VGG13	MobileNetV2	ShuffleNetV2
Baseline	71.14	72.34	73.79	70.36	73.77	64.6	71.82
JPEG-DL	**71.92**$_{\pm 0.31}$ (+0.78)	**73.39**$_{\pm 0.19}$ (+1.05)	**74.46**$_{\pm +0.11}$ (+0.67)	**71.10**$_{\pm +0.41}$ (+0.74)	**75.32**$_{\pm 0.10}$ (+1.55)	**65.91**$_{\pm 0.11}$ (+1.31)	**73.04**$_{\pm 0.16}$ (+1.22)

4.4.2 Experimental Results

CIFAR-100 and Fine-grained tasks Results. The performance of JPEG-DL is reported in Tables 4.2 and 4.3 for CNN models. Across all seven tested architectures on CIFAR-100, JPEG-DL consistently improves top-1 accuracy over its baseline, with gains of up to 1.53%. For fine-grained classification tasks,

Table 4.3: Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs.

Model	Method	CUB-200	Dogs	Flowers	Pets
ResNet-18	Baseline	54.00$_{\pm 1.43}$	63.71$_{\pm 0.32}$	57.13$_{\pm 1.28}$	70.37$_{\pm 0.84}$
	Balle *et al.* [4]	50.78$_{\pm 2.21}$ (-3.22)	53.47$_{\pm 7.37}$ (-10.24)	55.46$_{\pm 0.59}$ (-1.67)	56.14$_{\pm 17.16}$ (-14.23)
	Shin & Song [103]	55.34$_{\pm 0.14}$ (+1.34)	63.03$_{\pm 0.56}$ (-0.68)	55.78$_{\pm 1.44}$ (-1.35)	71.45$_{\pm 1.01}$ (+1.08)
	Esser *et al.* [29]	51.58$_{\pm 0.18}$ (-2.42)	60.45$_{\pm 0.23}$ (-3.26)	58.04$_{\pm 0.58}$ (+0.91)	68.81$_{\pm 0.55}$ (-1.56)
	JPEG-DL	**58.81**$_{\pm 0.12}$ (+4.81)	**65.57**$_{\pm 0.37}$ (+1.86)	**68.76**$_{\pm 0.57}$ (+11.63)	**74.84**$_{\pm 0.66}$ (+4.47)
DenseNet-121	Baseline	57.70$_{\pm 0.44}$	66.61$_{\pm 0.17}$	51.32$_{\pm 0.57}$	70.26$_{\pm 0.79}$
	Balle *et al.* [4]	52.00$_{\pm 1.41}$ (-5.70)	60.07$_{\pm 6.41}$ (-6.54)	46.60$_{\pm 2.87}$ (-4.72)	61.91$_{\pm 1.88}$ (-8.35)
	Shin & Song [103]	57.19$_{\pm 0.78}$ (-0.51)	66.90$_{\pm 0.13}$ (+0.29)	51.04$_{\pm 0.87}$ (-0.28)	69.95$_{\pm 1.21}$ (-0.31)
	Esser *et al.* [29]	56.46$_{\pm 0.30}$ (-1.24)	64.89$_{\pm 0.12}$ (-1.72)	55.98$_{\pm 0.24}$ (+4.60)	69.58$_{\pm 0.59}$ (-0.68)
	JPEG-DL	**61.32**$_{\pm 0.43}$ (+3.62)	**69.67**$_{\pm 0.58}$ (+3.06)	**72.22**$_{\pm 1.05}$ (+20.90)	**75.90**$_{\pm 0.68}$ (+5.64)

Table 4.4: Top-1 accuracy (%) on CIFAR-100 and two fine-grained image classification tasks using EfficientFormer-L1. We report the mean and standard deviation of the experimental results over three runs with different random seeds. The best results are indicated in bold, and values in parentheses indicate relative accuracy gains over the baseline.

Model	Method	Accuracy (%)
CIFAR-100	Baseline	80.27$_{\pm 0.33}$
	JPEG-DL	**80.49**$_{\pm 0.23}$ (+0.22)
Flowers	Baseline	69.78$_{\pm 0.33}$
	JPEG-DL	**73.05**$_{\pm 0.12}$ (+3.27)
Pets	Baseline	65.52$_{\pm 1.23}$
	JPEG-DL	**69.28**$_{\pm 0.42}$ (+3.76)

Table 4.5: Top-1 validation accuracy (%) on ImageNet with different model architectures.

Method	SqueezeNetV1.1	Resnet18	Resnet34
Baseline	57.95	69.75	73.31
JPEG-DL	**58.26** (+0.31)	**70.13** (+0.38)	**73.54** (+0.23)

JPEG-DL achieves substantial improvements over its baseline, with accuracy gains reaching up to 20.90% across all datasets using two different CNN models. For transformer-based models, Table 4.4 presents the top-1 validation accuracy, further demonstrating the significant performance enhancements achieved by JPEG-DL over the baseline.

Comparison with more Baselines. In Table 4.3, we have included comparisons against other baselines that employed JPEG quantization in the pipeline, but with heuristic approaches to handle the non-differentiability and zero derivative problems of JPEG quantization. For example, Balle *et al.* [4] allow optimization via stochastic gradient descent by replacing the quantizer with an additive i.i.d. uniform noise, which has the same width as the quantization bins, where $\hat{z} = z + q * U(-0.5, 0.5)$. However, during validation, they employed Q_u using the trained quantization tables. Shin & Song [103] employ a third-order polynomial approximation of the rounding function to make JPEG differentiable. The gradient approximation through the round function is a key aspect of certain neural network approaches. [29][1] employ a straight-through estimator, originally proposed in [6], to achieve this approximation. This method treats the round function as a pass-through operation during backpropagation, allowing for effective gradient estimation. Our experimental results demonstrate that we consistently outperform all tested baselines. Even though these methods find a way to make the quantization differentiable, they still hurt the performance. This is consistent with the well-known wisdom in the literature, which also shows that they did not take advantage of the higher level of non-linearity introduced into DNN architectures, as proposed by our JPEG-DL framework.

ImageNet-1K Results. The performance of JPEG-DL is shown in Table 4.5. With a trivial increase in complexity (adding 128 parameters), JPEG-DL achieves a gain of 0.31% in top-1 accuracy for SqueezeNetV1.1 compared to the baseline using a single round of Q_d quantization operation. By increasing the number of quantization rounds to five, we observe an additional improvement of 0.20%, leading to a total gain of 0.51% over the baseline. The best results are indicated in bold, and values in parentheses indicate relative accuracy gains over the baseline.

Fixing Hyperparamters across different datasets. In Table 4.6, we evaluate the difference between the hyperparameters used for CIFAR-100 and the fine-grained task, where the only difference lies in the learning rate. In the following setup, we present the results when a learning rate of 0.003 is applied to the fine-grained dataset. We observe that the performance difference is marginal, and in some cases, we can even achieve higher gains in model performance. This suggests that the choice of learning rate can have a significant impact on the effectiveness of the model.

Table 4.6: Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs.

Model	Learning Rate	CUB-200	Dogs	Flowers	Pets
ResNet-18	0.005	**58.81**$_{\pm 0.12}$ (+4.81)	**65.57**$_{\pm 0.37}$ (+1.86)	68.76$_{\pm 0.57}$ (+11.63)	74.84$_{\pm 0.66}$ (+4.47)
	0.003	58.62$_{\pm 0.50}$ (+4.61)	65.45$_{\pm 0.17}$ (+1.74)	**69.61**$_{\pm 1.16}$ (+12.48)	**74.90**$_{\pm 0.82}$ (+4.53)
DenseNet-121	0.005	**61.32**$_{\pm 0.43}$ (+3.62)	**69.67**$_{\pm 0.58}$ (+3.06)	72.22$_{\pm 1.05}$ (+20.90)	**75.90**$_{\pm 0.68}$ (+5.64)
	0.003	60.92$_{\pm 0.50}$ (+3.22)	69.53$_{\pm 0.49}$ (+2.92)	**72.45**$_{\pm 0.92}$ (+21.13)	75.83$_{\pm 0.64}$ (+5.56)

4.4.3 Comparison with JPEG-based Data Augmentation and Other Preprocessing Methods

JPEG-based Data Augmentation. We have compared and implemented JPEG-based data augmentation across three different sets, each with varying ranges of quantity factor (QF). For each tested range, we randomly select a QF for each image within the mini-batch. These sets have been tested on fine-grained datasets for all the models evaluated. Table 4.7 shows corresponding results for these tested sets. Consistent with the common knowledge in the literature, JPEG-based data augmentation in general degrades the accuracy performance.

Non-Learnable and Learnable Preprocessing Methods. We have tested the performance of non-learnable and learnable preprocessing methods during training and validation across all tested models on all tested fine-grained datasets, as demonstrated in Table 4.7. For non-learnable preprocessing, we have considered applying denoising using a Gaussian kernel and histogram equalization for each image within a mini-batch. As for the learnable one, we have compared it with [116] as a learnable resize module, which has a bandpass nature in that it learns to boost details in certain frequency subbands that benefit the downstream recognition models. We have applied the same setup mentioned in their paper, in which they fine-tuned the model to achieve some improvement [2].

Impact of chrominance subsampling on JPEG-DL. In Table 4.7, we evaluate various subsampling schemes for all tested models across all tested fine-grained datasets. Notably, although there is no clear winner among different chroma subsampling methods, DNNs indeed respond to color information differently from human; color information is more important to DNNs than human since chroma subsampling

[1]We followed the experimental setup defined by Esser *et al.* [29] using Stochastic Gradient Descent (SGD) with an initial learning rate of 0.01, adhering to the same learning rate decay schedule as the underlying model. Additionally, we initialized Q by setting it to $2|Q|/\sqrt{2^b - 1}$ and implemented gradient scaling during training mentioned in their paper. We also maintained consistency by using the same number of bits used by JPEG-DL.

[2]We have verified the correctness of our PyTorch implementation with the authors' TensorFlow implementation provided in https://colab.research.google.com/github/google-research/google-research/blob/master/muller/muller_demo.ipynb.

Table 4.7: JPEG-DL is compared to JPEG-based augmentation (orange), non-learnable (yellow), and learnable preprocessing (blue) methods. JPEG-DL employs three different chrominance subsampling schemes shown in green color. Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs. The best results are indicated in bold, and values in parentheses indicate relative accuracy gains over the baseline.

Dataset	Method	ResNet-18		DenseNet-121	
CUB-200	Baseline	54.00$_{\pm 1.43}$		57.70$_{\pm 0.44}$	
	Rand. QF [1:50]	52.86$_{\pm 0.69}$	(-1.14)	55.72$_{\pm 0.72}$	(-1.98)
	Rand. QF [50:100]	54.53$_{\pm 0.41}$	(+0.53)	56.98$_{\pm 0.90}$	(-0.72)
	Rand. QF [1:100]	53.91$_{\pm 0.43}$	(-0.09)	57.20$_{\pm 0.33}$	(-0.50)
	Denoising	54.82$_{\pm 0.40}$	(+0.82)	56.02$_{\pm 1.00}$	(-1.68)
	Equalization	49.00$_{\pm 0.31}$	(-5.00)	54.32$_{\pm 0.57}$	(-3.38)
	Learnable Resize	54.98$_{\pm 0.19}$	(+0.98)	57.35$_{\pm 0.22}$	(-0.35)
	JPEG-DL (4:2:0)	58.00$_{\pm 0.12}$	(+4.00)	60.55$_{\pm 0.71}$	(+2.85)
	JPEG-DL (4:2:2)	58.11$_{\pm 0.10}$	(+4.11)	**61.51**$_{\pm 0.41}$	(+3.81)
	JPEG-DL (4:4:4)	**58.81**$_{\pm 0.12}$	(+4.81)	61.32$_{\pm 0.43}$	(+3.62)
Dogs	Baseline	63.71$_{\pm 0.32}$		66.61$_{\pm 0.17}$	
	Rand. QF [1:50]	60.61$_{\pm 0.17}$	(-3.10)	64.93$_{\pm 0.17}$	(-1.68)
	Rand. QF [50:100]	63.14$_{\pm 0.30}$	(-0.57)	66.97$_{\pm 0.18}$	(+0.36)
	Rand. QF [1:100]	62.12$_{\pm 0.16}$	(-1.59)	65.59$_{\pm 0.54}$	(-1.02)
	Denoising	62.52$_{\pm 0.41}$	(-1.19)	66.36$_{\pm 0.34}$	(-0.25)
	Equalization	62.38$_{\pm 0.29}$	(-1.33)	67.12$_{\pm 0.19}$	(+0.51)
	Learnable Resize	63.10$_{\pm 0.43}$	(-0.61)	67.90$_{\pm 0.32}$	(+1.29)
	JPEG-DL (4:2:0)	65.57$_{\pm 0.31}$	(+1.86)	68.90$_{\pm 0.08}$	(+2.29)
	JPEG-DL (4:2:2)	**65.64**$_{\pm 0.16}$	(+1.93)	**69.85**$_{\pm 0.78}$	(+3.24)
	JPEG-DL (4:4:4)	65.57$_{\pm 0.37}$	(+1.86)	69.67$_{\pm 0.58}$	(+3.06)
Flowers	Baseline	57.13$_{\pm 1.28}$		51.32$_{\pm 0.57}$	
	Rand. QF [1:50]	58.33$_{\pm 0.48}$	(+1.20)	51.44$_{\pm 0.79}$	(+0.12)
	Rand. QF [50:100]	55.98$_{\pm 0.89}$	(-1.15)	51.67$_{\pm 1.10}$	(+0.35)
	Rand. QF [1:100]	57.55$_{\pm 0.66}$	(+0.42)	52.32$_{\pm 0.73}$	(+1.00)
	Denoising	57.28$_{\pm 1.06}$	(+0.15)	50.37$_{\pm 0.79}$	(-0.95)
	Equalization	60.56$_{\pm 0.98}$	(+3.43)	61.72$_{\pm 0.21}$	(+10.40)
	Learnable Resize	57.41$_{\pm 0.48}$	(+0.28)	51.21$_{\pm 0.82}$	(-0.11)
	JPEG-DL (4:2:0)	67.58$_{\pm 1.50}$	(+10.45)	68.01$_{\pm 1.17}$	(+16.69)
	JPEG-DL (4:2:2)	67.75$_{\pm 1.19}$	(+10.62)	68.79$_{\pm 1.08}$	(+17.47)
	JPEG-DL (4:4:4)	**68.76**$_{\pm 0.57}$	(+11.63)	**72.22**$_{\pm 1.05}$	(+20.90)
Pets	Baseline	70.37$_{\pm 0.84}$		70.26$_{\pm 0.79}$	
	Rand. QF [1:50]	69.71$_{\pm 0.54}$	(-0.66)	68.93$_{\pm 0.50}$	(-1.33)
	Rand. QF [50:100]	70.00$_{\pm 0.57}$	(-0.37)	68.63$_{\pm 0.82}$	(-1.63)
	Rand. QF [1:100]	69.52$_{\pm 0.62}$	(-0.85)	70.60$_{\pm 1.05}$	(+0.34)
	Denoising	69.15$_{\pm 1.07}$	(-1.22)	69.03$_{\pm 1.13}$	(-1.23)
	Equalization	73.00$_{\pm 0.79}$	(+2.63)	75.04$_{\pm 1.30}$	(+4.78)
	Learnable Resize	70.04$_{\pm 0.72}$	(-0.33)	68.83$_{\pm 1.85}$	(-1.43)
	JPEG-DL (4:2:0)	74.64$_{\pm 1.34}$	(+4.27)	**76.21**$_{\pm 0.35}$	(+5.95)
	JPEG-DL (4:2:2)	74.81$_{\pm 0.51}$	(+4.44)	76.17$_{\pm 1.83}$	(+5.90)
	JPEG-DL (4:4:4)	**74.84**$_{\pm 0.66}$	(+4.47)	75.90$_{\pm 0.68}$	(+5.64)

formats 4:4:4 and 4:2:2, in general, give rise to better accuracy performance than the 4:2:0 subsampling format which is adopted predominantly in the image and video coding.

4.5 Analysis and Discussion

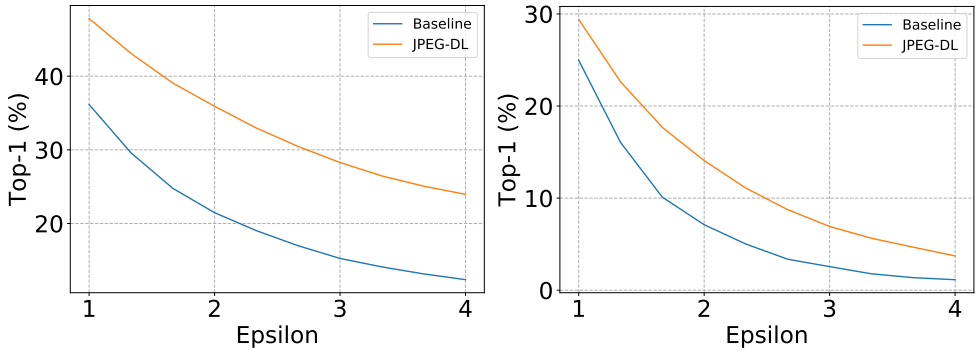
Robustness. JPEG has generally been used as an empirical defense mechanism against adversarial attacks by mitigating adversarial perturbations and enhancing the robustness of DNNs, as discussed in Section 4.2. To evaluate the adversarial robustness of JPEG-DL models in comparison to standard DNN, we conduct experiments using two attack methods, FGSM and PGD, on CIFAR-100 with two different models from Table 4.2. The perturbation budget, Epsilon, ranged from 1 to 4 for both attack methods. For PGD, we applied 5 steps with a perturbation step size of $(2.5 \times \text{Epsilon})/\text{steps}$, following the setup used by [74]. As shown in Fig. 4.6, the JPEG-DL models significantly improve the adversarial robustness compared to the standard DNN models, with improvements of up to 15% for FGSM and 6% for PGD. We will show some examples of the quantization tables used in this study in the next subsection.

Layer Replacement. We extend our evaluation of the JPEG-DL framework to include layer replacement, where we substitute the ReLU activation function with our JPEG layer in ResNet-18. This replacement is implemented directly after the first convolution layer, which has 64 kernels and outputs a feature map of 112x112. For this layer replacement, we did not perform any color space conversion and added a quantization table with a size of 8x8 for each kernel output. We follow the experimental setup mentioned in Section 4.4, except we use a learning rate of 0.001, α equal to 2, and b equal to 6 bits. The performance of this approach is shown in the last row of the following table and is compared to the performance of JPEG-DL when the JPEG layer is placed directly after the input layer shown in Fig. 4.1. The results shown in Table 4.8 confirm that this new architecture, resulting from layer replacement with the JPEG layer or inserting the JPEG layer directly after the Input Layer, indeed improves deep learning performance by a significant margin, thereby showing the potential of new forms of nonlinear operation in DNN architectures.

Table 4.8: Top-1 validation accuracy (%) on various fine-grained image classification tasks and model architectures. We report the mean and standard deviation of experimental results over three runs.

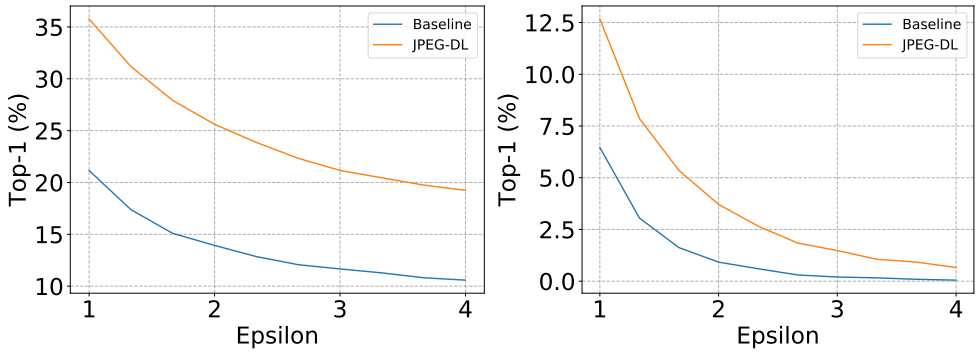
Method	CUB-200	Dogs	Flowers	Pets
JPEG-DL (Input Layer)	$58.81_{\pm 0.12}$ (+4.81)	$65.57_{\pm 0.37}$ (+1.86)	$68.76_{\pm 0.57}$ (+11.63)	$74.84_{\pm 0.66}$ (+4.47)
JPEG-DL (1^{st} Conv Layer)	$59.27_{\pm 0.04}$ (+5.27)	$65.33_{\pm 0.07}$ (+1.62)	$72.10_{\pm 1.46}$ (+14.97)	$76.11_{\pm 0.37}$ (+5.74)

Sensitivity and Designed Quantization Tables. Figure 4.7 presents the estimated sensitivity values for all models considered in the CUB200 dataset and a subset of models used for CIFAR-100, as proposed



(a) FGSM for VGG13

(b) PGD for VGG13



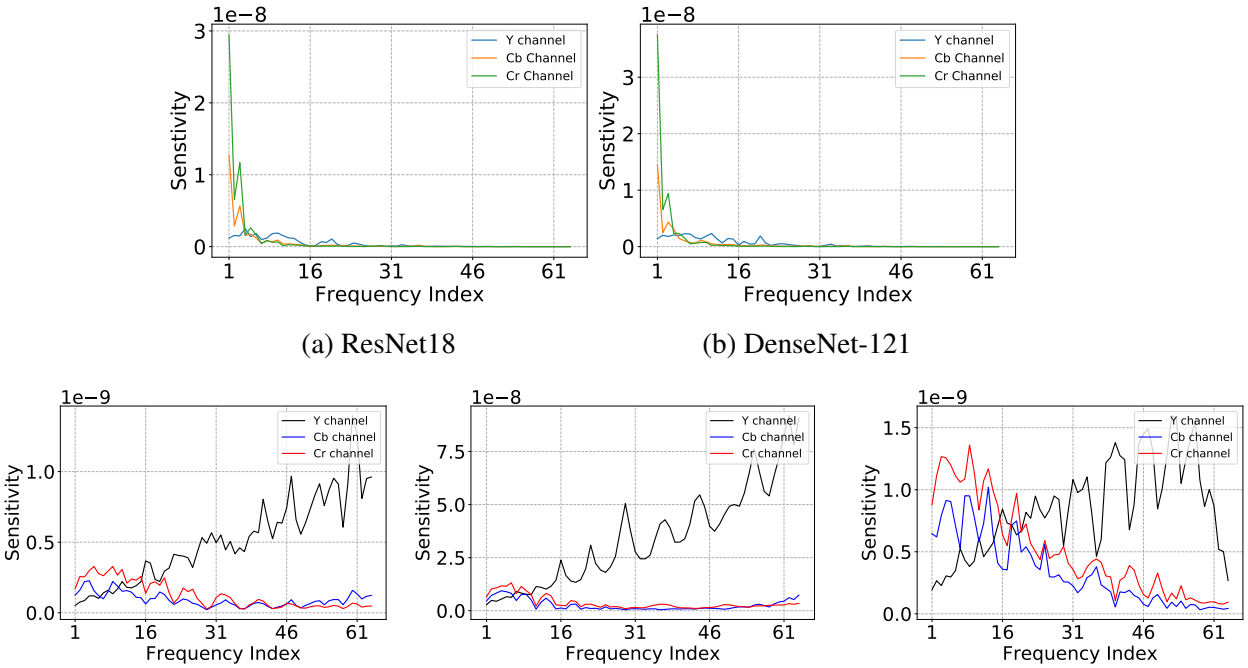
(c) FGSM for Res56

(d) PGD for Res56

Figure 4.6: Evaluate the adversarial robustness of JPEG-DL models in comparison to standard DNN on VGG13 and Res56 for CIFAR-100 against FGSM and PGD attacks.

by authors in [101, 144]. These estimated sensitivity values are used to initialize the quantization table for JPEG-DL. Figures 4.9 and 4.8 presents the Y and CbCr quantization tables, both at initialization and after convergence, for VGG13 trained on CIFAR-100 and ResNet-18 trained on the CUB200 dataset.

Feature maps visualization. Figure 4.10 presents the feature maps extracted after the first dense block in DenseNet-121 for both the JPEG-DL model and the baseline model, trained on the CUB200 dataset using the same model from Table 4.3. The output of the feature maps at this stage is of size 56×56, and both sets are shown in the same sequence using the same original image. The example shown was misclassified by the baseline model but correctly classified by the JPEG-DL model. As illustrated, the feature maps from the JPEG-DL model exhibit significantly better contrast between the foreground object (the bird) and the background. In particular, the foreground is enclosed within a well-defined contour, making it visually distinct from the background. In contrast, the baseline model's feature maps display a more blended structure, where the foreground contains higher energy in low-frequency components, causing it to merge more smoothly with the background. Figure 4.11 shows a similar phenomenon using



(a) ResNet18 (b) DenseNet-121

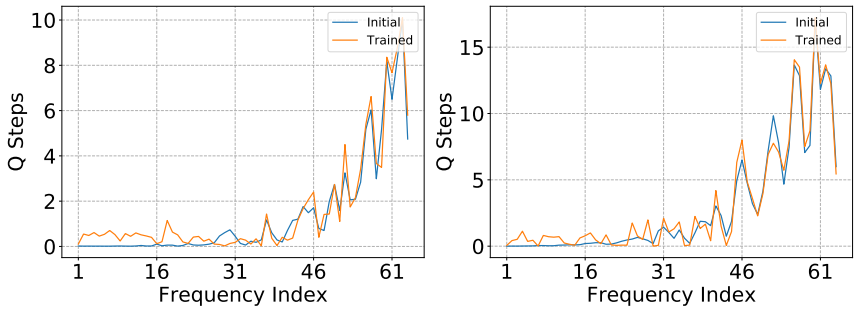
(c) VGG13 (d) Res56 (e) ShuffleNetV2

Figure 4.7: The estimated sensitivity is shown for two pre-trained models on the CUB200 (first row) and three pre-trained models on the CIFAR-100 (second row), with sensitivity indices arranged in the default zigzag order.

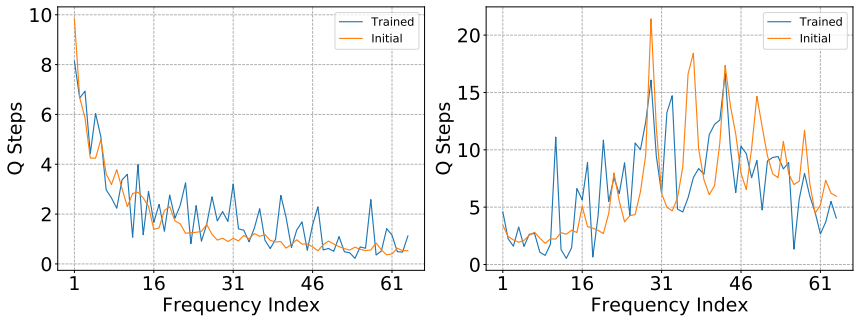
a second example from the Flowers dataset, trained with the model configuration in Table 4.3. In this case, the JPEG-DL model not only enhances foreground-background separation but also more effectively suppresses irrelevant background information. These discrepancies in feature map clarity and contrast are propagated through subsequent blocks of DenseNet-121, eventually contributing to the misclassification problem observed in the baseline model.

Interpretability using CAM. Fig. 4.12 illustrates GradCAM++ visualizations [10] for two examples from the CUB-200 dataset, comparing the baseline model with JPEG-DL using ResNet-18 as our underlying model. In both instances, the baseline model incorrectly classifies the input, while the JPEG-DL model correctly classifies it. The visualizations highlight how JPEG-DL focuses more precisely on the main object in the image, demonstrating the model's improved attention to key regions that contribute to correct classification. This highlights the effectiveness of JPEG-DL in enhancing model interpretability and performance, and it also supports the case for background removal discussed in the previous subsection.

Mitigating the Inference Overhead of Differentiable Soft Quantizers. The reconstruction space $\hat{\mathcal{A}}$



(a) DenseNet-121 (Y Channel) (b) DenseNet-121 (CbCr Channel)

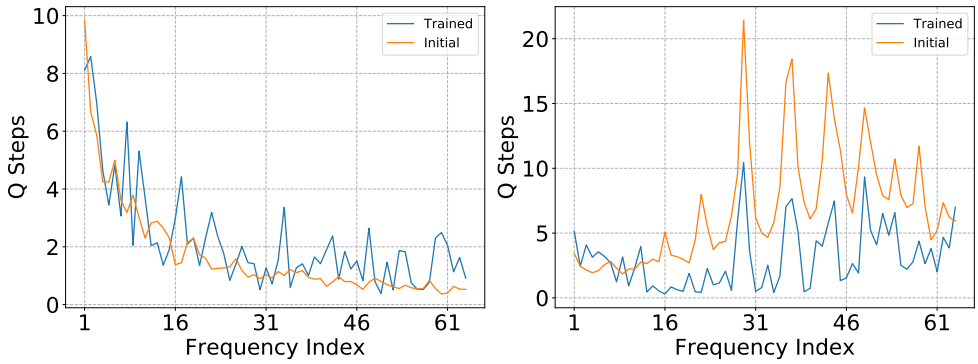


(c) Res56 (Y Channel) (d) Res56 (CbCr Channel)

Figure 4.8: Initial and final quantization tables for Res56 trained on CIFAR-100 and DenseNet-121 trained on CUB200, with frequency indices arranged in the default zigzag order.

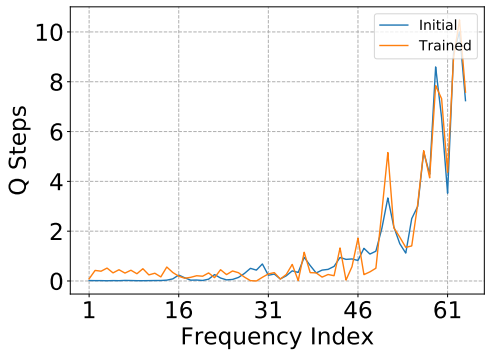
is configured with a specific size, determined by the parameter L, which is set to 2^{b-1}, as mentioned in Section 4.4. For CIFAR-100 and Fine-grained datasets, b is set to 8 bits, resulting in a reconstruction space length of 513. For ImageNet-1K, b is set to 11 bits, resulting in a length of 2047. This high dimensionality of the reconstruction space poses a computational challenge during inference when calculating the conditional probability mass function (CPMF) $P_\alpha(\cdot|z)$. To address this, the support of each CPMF is restricted to the five closest points in $\hat{\mathcal{A}}$ to the coefficient z being quantized. This simplification, referred to as the masked CPMF, significantly reduces computational complexity without compromising the effectiveness of the quantization scheme $\mathcal{Q}_d$.

Figure 4.13 demonstrates the rapid decay of probability mass in CPMFs as the reconstruction level $\hat{z}$ deviates from the quantized value the uniform quantizer $\mathcal{Q}_u(z)$. The set $\mathcal{M}$, consisting of five reconstruction levels centered around $\mathcal{Q}_u(z)$, captures a significant portion of the total probability mass, even in extreme cases where the quantization step q and the parameter α are very small. This observation suggests that restricting the CPMF support to $\mathcal{M}$ (masked CPMF) is a valid simplification. Instead of computing the full-space CPMF with length $2L + 1$, we can efficiently calculate the masked CPMF with length 5 using softmax on $\mathcal{M}$. The impact of this simplification on the quantization scheme $\mathcal{Q}_d$ is negli-

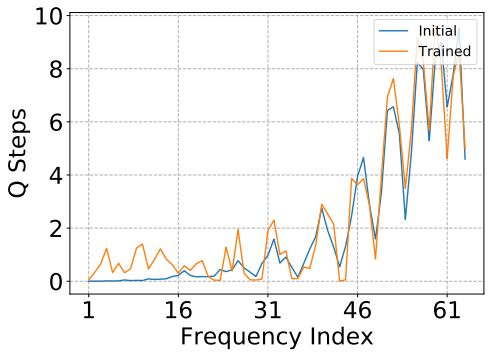


(a) VGG13 (Y Channel)

(b) VGG13 (CbCr Channel)



(c) ResNet-18(Y Channel)

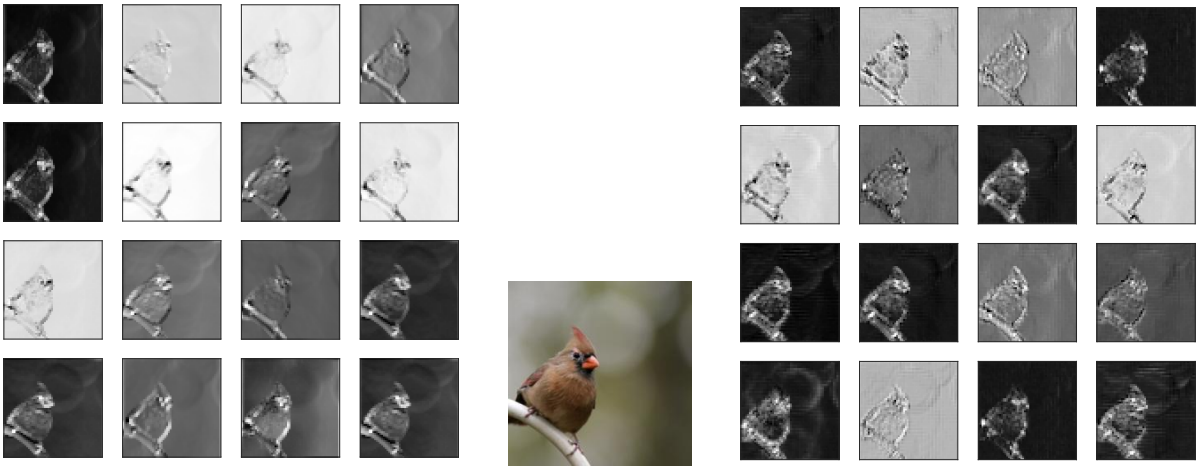


(d) ResNet-18 (CbCr Channel)

Figure 4.9: Initial and final quantization tables for VGG13 trained on CIFAR-100 and ResNet18 trained on CUB200, with frequency indices arranged in the default zigzag order.

α	$\mathcal{Q}_d(z = 0.5)$		
	Full Space	Masked	Δ
1	0.5	0.5027	0.0027
3	0.5	0.5	5.96e-08
5	0.5	0.5	5.96e-08
10	0.5	0.5	0

Table 4.9: $\mathcal{Q}_d(z = 0.5; q = 1, \alpha)$ values resulting from the full-space CPMF and the masked CPMF at different α, alongside the absolute difference Δ between the full-space and masked cases.



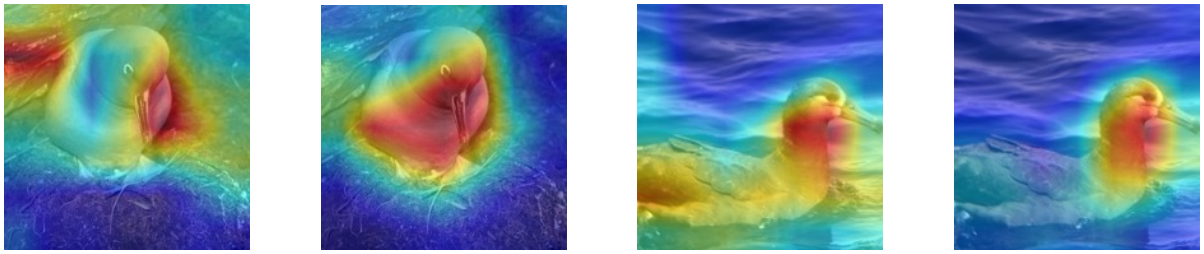
(a) Baseline (b) Input (c) JPEG-DL

Figure 4.10: Feature maps of size 56×56 are shown after the first dense block in DenseNet-121 for both JPEG-DL and baseline models Figs. in 4.10a and 4.10c, respectively, using an original input shown in Fig.4.10b. The JPEG-DL model highlights the foreground (bird) more distinctly, while the baseline model shows less contrast, contributing to its misclassification.



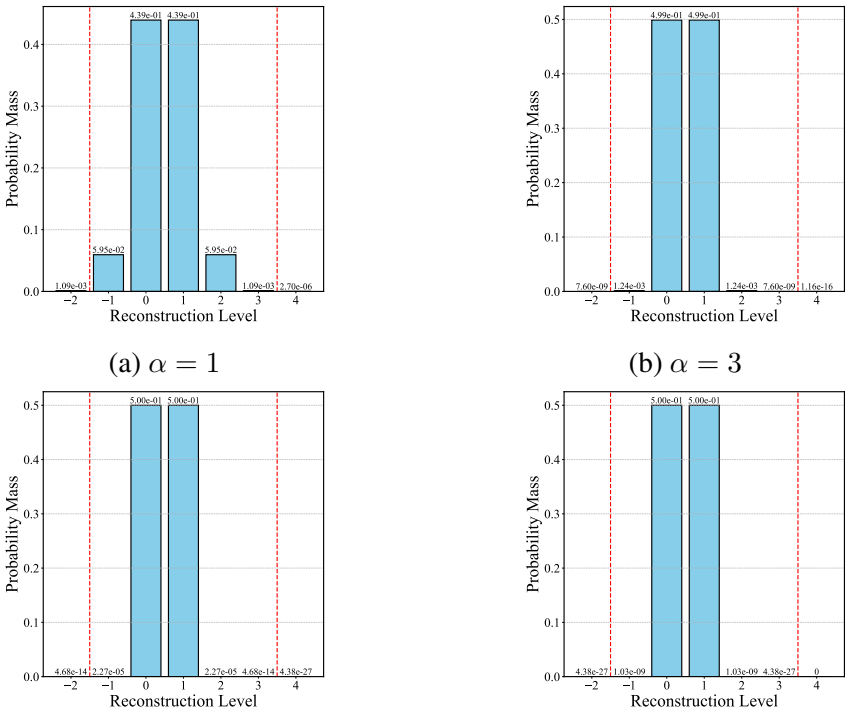
(a) Baseline (b) Input (c) JPEG-DL

Figure 4.11: Feature maps of size 56×56 are shown after the first dense block in DenseNet-121 for both JPEG-DL and baseline models in Figs. 4.11a and 4.11c, respectively, using an original input shown in Fig. 4.11b. The JPEG-DL model highlights the foreground (flower) more distinctly, while the baseline model shows less contrast, contributing to its misclassification.



(a) Baseline (b) JPEG-DL (c) Baseline (d) JPEG-DL

Figure 4.12: GradCAM++ visualization for baseline and JPEG-DL models on CUB200 using two examples, where the baseline model incorrectly classified them and the JPEG-DL model correctly classified them.



(a) $\alpha = 1$ (b) $\alpha = 3$

(c) $\alpha = 5$ (d) $\alpha = 10$

Figure 4.13: Partial visualization of CPMFs $P_\alpha(\cdot|z = 0.5)$ computed using (4.6), with $L = 1023$, $q = 1$, and α values of (a) 1, (b) 3, (c) 5, and (d) 10. The reconstruction levels between two red dashed lines represent $\mathcal{M}$.

Inference Settings	Inference time (milliseconds)	Inference Throughput (img/sec)
Baseline	5.46	180.65
JPEG-DL (Masked CPMFs)	7.43	133.79
JPEG-DL (Full Space)	18.86	52.37

Table 4.10: Comparison of inference time between different inference settings measured for Resnet18 on ImageNet, following the experimental setup mentioned in Section 4.4. Evaluation is conducted on a single NVIDIA RTX A5000 GPU.

gible, particularly for high values of α, as shown in Table 4.9. Therefore, using the masked CPMF in our implementation is a justified simplification that significantly reduces computational complexity without compromising accuracy.

Specifically, Fig. 4.13 illustrates how fast the probability mass of a reconstruction level $\hat{z}$ decays in CPMFs as $\hat{z}$ deviates from the uniform quantizer $\mathcal{Q}_u(z)$. It's clear that the set $\mathcal{M} = \{\mathcal{Q}_u(z)-2q, \mathcal{Q}_u(z)-q, \mathcal{Q}_u(z), \mathcal{Q}_u(z)+q, \mathcal{Q}_u(z)+2q\}$ of reconstruction levels retain a total probability mass close to 1, even in an adversarially chosen scenario where both q and α are extremely small[3] (i.e., 1) and the coefficient C being quantized lies exactly at a quantization threshold. Therefore, instead of computing the full-space CPMF with length $2L+1$ following (4.6), we only conduct softmax on $\mathcal{M}$ to get the masked CPMF with length 5. The error in $\mathcal{Q}_d$ caused by this masking is negligible as shown in Table 4.9, especially in the high α case. Therefore, it's totally justified to simplify the full-space CPMF to the masked CPMF in our implementation.

Table 4.10 presents a comparison of inference time and throughput between the standard model and our unified model, which incorporates the JPEG layer with the underlying model. The inference times for both the full-space CPMF and the masked CPMF are measured. The reported inference time is averaged across the entire validation set of ImageNet using Resnet18 from Table 4.5. These results demonstrate the effectiveness of the proposed approach in addressing the computational complexity of the JPEG-DL framework.

4.6 Chapter Summary

In contrast to the conventional understanding that JPEG compression negatively impacts the DL performance, this chapter introduces a novel trainable JPEG compression layer into the DL pipeline to improve DL performance, termed JPEG-DL. This layer enables gradient-based optimization for quantization parameters and can be integrated seamlessly with any underlying DNN to be trained jointly. To validate

[3]As q and α increase, the CPMF becomes even sharper, and as a result, $\mathcal{M}$ will capture increasingly higher probability mass.

the effectiveness of JPEG-DL, we conduct extensive experiments on six image classification datasets and show significant gains over the standard DL. Additionally, we demonstrate that JPEG-DL improves the adversarial robustness of the learned model.

Chapter 5

A Coded Knowledge Distillation Framework Based on Adaptive JPEG Encoding

This chapter introduces a novel framework for knowledge distillation (KD), termed coded knowledge distillation (CKD), which enhances student model generalization by mitigating the overconfidence of large teacher models. Unlike conventional KD, where the student mimics the teacher's response to raw inputs, CKD introduces a coded teacher that processes adaptively compressed inputs using JPEG encoding. This adaptive compression is guided by a new selection mechanism that ensures the compressed input maintains classification correctness while reducing output confidence, thereby producing more informative supervision signals. Experimental results across multiple datasets and architectures demonstrate that CKD consistently improves student accuracy, outperforming standard KD and other perturbation-based methods. Furthermore, CKD integrates effectively with various knowledge transfer techniques, offering a scalable and efficient solution for model compression and deployment in resource-constrained environments.

The journal version of this chapter is available in [98].

5.1 Introduction

Recent advances in many computer vision tasks have been mainly obtained by deep neural networks (DNNs) that are large in size, and therefore require high computation and memory during inference [3,44]. Consequently, there has been a surge of research endeavors focused on developing techniques to reduce the size of DNNs, enabling their practical implementation on devices with limited resources.

One promising approach to creating smaller and more efficient DNNs is knowledge distillation (KD). In KD, a small DNN (student) is trained to mimic the behavior of a pre-trained large DNN (teacher) so that the trained student can generalize in a way similar to the teacher [44]. Followed by [44], many researchers tried to understand why distillation works [76, 93, 137], and proposed to distill different forms of knowledge from the teacher model. These forms include, for example, (i) the output probability vector response [42, 44, 80, 145], (ii) feature maps [43, 52, 96, 114, 117, 138, 140], and (iii) relationships between different layers [86, 90].

Despite achieving significant success, we notice that due to its typically large size, the teacher is often susceptible to overfitting the training set [37, 83, 105]. In other words, for most training samples, the teacher's output probability vector closely resembles a one-hot vector, where the sole non-zero entry corresponds to the correct class label. This characteristic impedes the teacher model's ability to generalize effectively for unseen samples, thereby diminishing its capacity to support the student in achieving good generalization.

This chapter aims to tackle the aforementioned issue without resorting to retraining the teacher model. Instead, our approach focuses on inputting nonlinearly processed samples into the pre-trained teacher model to obtain improved and more generalizable output probability vectors. These vectors are then passed to the student model during the distillation process. To fulfill this goal, each training sample is first adaptively compressed and then fed into the pre-trained teacher model so that the following two conditions are satisfied:

- *Condition (I)*: If the teacher correctly classifies the original sample, the adaptively compressed sample is also correctly classified.
- *Condition (II)*: The output probability vector in response to the adaptively compressed sample is not as close to the one-hot probability vector as the output probability vector in response to the original sample.

During the distillation process, instead of passing the output probability vector of the teacher in response to the original sample, we pass the output probability vector of the teacher in response to the adaptively compressed sample to the student. Since generating a compressed sample involves an encoding process, we refer to such a modified KD framework as *coded* knowledge distillation (CKD), and the corresponding modified teacher as a *coded* teacher. In other words, compared to the teacher in KD, the *coded* teacher in CKD has an additional adaptive encoding layer in the front which encodes adaptively an input image into a compressed version and then feeds the compressed image into the teacher. In CKD, the student is trained to mimic instead the behavior of a *coded* teacher.

How would each original input image x be adaptively encoded by the *coded* teacher? In this chapter, we shall apply JPEG. As a popular lossy compression technique, JPEG in its primitive form uses the quality factor (QF) parameter $q \in \mathcal{Q}$ to establish a trade-off between human-perceived quality and compression ratio (CR) [91]. Different QF values q result in different compressed images $\hat{x}$. The *coded* teacher then adaptively selects a suitable q for each input image x, which is optimal in some sense, and then uses JPEG to encode x. The contributions of the paper are summarized as follows:

- We present a new KD framework, dubbed as CKD, in which an additional adaptive encoding layer is introduced before the teacher, resulting in a *coded* teacher, for the purpose of generating better, more generalizable output probability vectors to be passed to the student model during the distillation process.
- Within our proposed CKD framework, an adaptive encoding method in conjunction with JPEG is presented and justified.
- We show by experiments that in comparison with KD, CKD along with the proposed *coded* teacher can achieve better performance. This improvement is particularly evident when tested on the ImageNet and fine-grain image classification datasets. Notably, on the CUB200-2011 and Stanford Dog datasets, we observed a significant percentage gain of 3.7% and 12.08%, respectively.
- In addition, further experiments on CIFAR-100 illustrate that the proposed *coded* teacher can also be used in conjunction with many other existing knowledge transfer methods—including FitNet [96], FSP [138], FT [52], CC [90], SP [117], AB [43], and RKD [86]—to improve validation accuracy. We show that the gain of such a combination can be up to 5% over the underlying knowledge transfer method compared to TALD [84].

5.2 Related Work

• **Knowledge distillation.** KD aims to distill the knowledge learned by a complex teacher model into a smaller student model, allowing the student model to achieve performance comparable to that of the teacher model while being computationally more efficient. To achieve this, the student loss function typically comprises a conventional cross-entropy term combined with an additional component representing the knowledge distilled from the teacher model throughout the training process. In its original form, KD involves minimizing the KL-divergence between the output probability vector of the teacher and that of the student model, where the output probability vector of the teacher is the knowledge passed to the student. Since the seminal paper by Hinton [44], many different forms of knowledge from the teacher have been investigated and passed to the student. Below, we briefly discuss some of them, with a specific focus on those that we later compare our method to in our experiments.

• **Intermediate-level distillation.** The intuition behind intermediate-level KD is to exploit the rich representations learned by the teacher model at different layers of its architecture. A pioneering work in this realm is FitNets [96], where the authors suggested using the responses from multiple intermediate hidden layers of the teacher, referred to as hints, to distill knowledge into the student model. Inspired by this work, in attention transfer (AT) [140], the attention maps generated by the teacher model guide the student to learn similar attention patterns. On the other hand, factor transfer (FT) [52] presents a different approach where transportable features are extracted using convolutional operations to paraphrase the teacher's knowledge. In another approach, Heo et al. [43] focus on transferring knowledge by matching the activation boundaries (AB) formed by hidden neurons between a teacher and a student model. The activation boundaries refer to the regions in the input space where the teacher model's activations change for the neuron's response (active or not), yielding different predictions.

Moreover, intermediate feature distributions can also play a crucial role in KD. Flow-based KD [138], known as FSP, aims to match the distributions of intermediate features of the student model with those of the teacher. The matching is typically achieved by minimizing the KL divergence or the maximum mean discrepancy, between the feature distributions of the teacher and student models. In a similar vein, similarity-preserving (SP) knowledge distillation [117] is a KD method based on the fact that inputs with similar semantics tend to generate similar activation patterns. Hence, SP focuses on preserving pairwise similarities within the teacher's feature map by transforming it into a matrix. This matrix encoding retains the similarity information of activations at the teacher layer. However, this transformation process may result in some information loss.

Another line of methodology to leverage intermediate layers in KD, involves harnessing the relationship between samples to improve knowledge transfer. Contrastive representative distillation (CRD) [114] encourages representations of similar samples to be closer in the embedding space, while simultaneously pushing apart those of different samples, from different classes. To this aim, this method exploits the correlations and higher-order output dependencies. Relational knowledge distillation (RKD) captures the pairwise relationships between the data points [86]. This can be achieved through various techniques, such as penalizing the structure difference with distance-wise and angle-wise distillation loss between the data points and encouraging the student model to reproduce similar relationships as the teacher model. In correlation congruence for knowledge distillation (CC) [90], the goal is to ensure that the student model's predictions exhibit similar correlation patterns to those of the teacher model. In addition to using correlation loss, the benefits of deploying other loss functions such as covariance loss or higher-order statistical moments are also discussed in [90].

Nevertheless, intermediate-level KD adds complexity to the distillation process compared to the logit-based KD. This, in turn, increases the computational cost and training time of intermediate-level KD methods.

• **KD using input perturbation.** In the conventional KD, both the teacher and student models are fed with the same input samples (raw images) during the distillation process. Yet, these raw images might not be able to explore the properties of the teacher model properly. Thus, Heo et al. [42] propose a KD method in which a teacher pre-trained by clean (raw) images is fed with adversarial examples during the KD process. Particularly, an attack scheme is modified to explore the properties of the teacher model by searching an adversarial sample supporting a decision boundary, referred to as boundary supporting sample (BSS). However, the current BSS attack method has limitations in fully exploring the complete range of potential perturbations that the teacher model can exhibit, as it is restricted to generating a single adversarial example. To tackle this issue, TALD [84] addresses the insufficiency of a single adversarial example by examining a broader spectrum of possible teacher model perturbations through the generation of multiple adversarial examples for the same original sample. TALD generates diverse adversarial examples using a multiple particle-based search technique known as Stein Variational Gradient Descent (SVGD). The authors of [84] execute SVGD for a specified number of iterations, denoted as L, to generate each sample sequentially, employing K particles sampled from the teacher conditional adversarial local distribution, with each iteration involving forward and backward propagation. Additionally, in [141], the authors propose a framework that incorporates an adversarial phase utilizing GANs to generate diverse

examples. Afterward, a co-distillation phase involving multiple classifiers is employed to leverage the divergent examples for enhancing the distillation performance which is computationally expensive.

Although compressed images generated by a *coded* teacher in CKD can also be regarded as perturbed input images for the teacher, they serve a completely different purpose. In fact, these compressed images are generated and employed in CKD to prevent the over-confident responses of the teacher from being passed to the student during the distillation process. Later, we will show through experiments that indeed CKD holds a greater promise compared to these methods with perturbed raw images as adversarial examples.

5.3 Preliminaries and Motivation

5.3.1 Preliminaries

Knowledge Distillation Formulation

As previously discussed, Hinton et al. [44] introduced response-based knowledge, represented by the output probability vector of the teacher, which is incorporated into the training process through the KD loss function. The KD loss function comprises two essential components: (i) the traditional cross-entropy loss function $\mathcal{H}(\cdot,\cdot)$, and (ii) a regularization term accounting for the teacher supervision. Specifically, the loss function in KD could be written as

$$\mathcal{L}_{KD} = (1-\alpha)\mathbf{E}_{x\sim\mathcal{D}}\mathcal{H}(\hat{y}, p_s) + \alpha\tau^2\mathbf{E}_{x\sim P}\mathcal{L}_{soft}\big(p_t^\tau, p_s^\tau\big), \tag{5.1}$$

where $\mathcal{D}$ is the empirical distribution of the original training sample image x, $\hat{y}$ is the one-hot probability vector corresponding to the ground truth label, p_s is the output probability vector of the student in response to the input x, and $\alpha \in [0,1]$ is a factor to balance the two loss terms. In Eq.5.1, $p_t^\tau = p^\tau(x;\theta_t)$ and $p_s^\tau = p^\tau(x;\omega_s)$ stand for the teacher's and student's output probability vectors after softening the logits with a temperature τ, respectively, and $\mathcal{L}_{soft}$ is the response-based knowledge loss defined as follows

$$\mathcal{L}_{soft} = \mathcal{KL}\big(p_t^\tau \,\big\|\, p_s^\tau\big), \tag{5.2}$$

where $\mathcal{KL}(\cdot\|\cdot)$ represents KL divergence, and θ_t and ω_s are the teacher and student models' parameters, respectively. Note that in the traditional KD, both the teacher and student accept the same original sample image x as their respective inputs during the KD process.

JPEG Compression

JPEG [91] is widely used for lossy compression in digital image processing while maintaining visual quality. The JPEG encoding process includes three fundamental phases: discrete cosine transformation (DCT),

quantization, and entropy encoding. It begins with the partitioning of the input image into 8×8 blocks, systematically processed in a raster scan order. Each block undergoes a transformation from the pixel domain to the DCT domain via an 8×8 DCT. The resulting DCT coefficients are subsequently subjected to uniform quantization using an 8×8 quantization table, with its entries specifying the quantization step sizes for each frequency bin, thereby representing the only lossy step, i.e. non-linear. This quantization table for JPEG encoding is designed in alignment with the assigned quality factor (QF). The DCT indices resulting from quantization are further encoded through a combination of run-length coding and Huffman coding, yielding the final bitstream. The trade-off between compression ratio (CR) and compression quality is mainly impacted by the designed quantization step controlled by the chosen QF value; that is, a higher QF yields a lower CR and better compression quality. The level of compression can be adjusted by selecting a QF value from 1 to 100.

Overfitting

Overfitting occurs when a DNN becomes excessively tailored to a finite number of training samples, resulting in overconfidence, which leads to a diminished ability to generalize well to unseen data [92, 110]. This phenomenon arises when the network learns intricate details and noise present in the training set, rather than capturing the underlying patterns and relationships that are more relevant for making accurate predictions on unseen data. As a result, the network's performance tends to degrade when it is fed with new examples that differ from the training data, thereby compromising its overall effectiveness. In particular, this phenomenon is commonly observed in large models that possess a significant number of parameters.

5.3.2 Motivation

In the context of KD, when the teacher model suffers from overfitting, it is unable to effectively impart generalized knowledge to the student. Thus, our motivation in this work is to prevent the overconfident responses of the teacher from being passed to the student during the distillation process. To this end, during the distillation we feed the pre-trained teacher model with a compressed version of the input image such that the teacher's output probability vector for the compressed sample is more toward the uniform distribution.

To clarify our objective, consider a classification task with three classes, namely $\{c_1, c_2, c_3\}$. In this case, the teacher's output probability vector p_t has three entries denoted by $\{p_t[1], p_t[2], p_t[3]\}$, where $p_t[i]$ corresponds to the output probability for c_i, for $i \in \{1, 2, 3\}$. Figure 5.1 shows the simplex of output probability vectors of a pre-trained teacher is depicted when its inputs are samples from class c_1. The teacher would correctly classify those samples $x \in c_1$ for which the teacher's output probability vector lies on the surface of the polyhedron defined by $p_t[1] > p_t[2]$ and $p_t[1] > p_t[3]$.

However, the teacher makes over-confident decisions for the samples whose output probabilities lie in the corner of the simplex, meaning that the probability vectors generated by the teacher for these samples

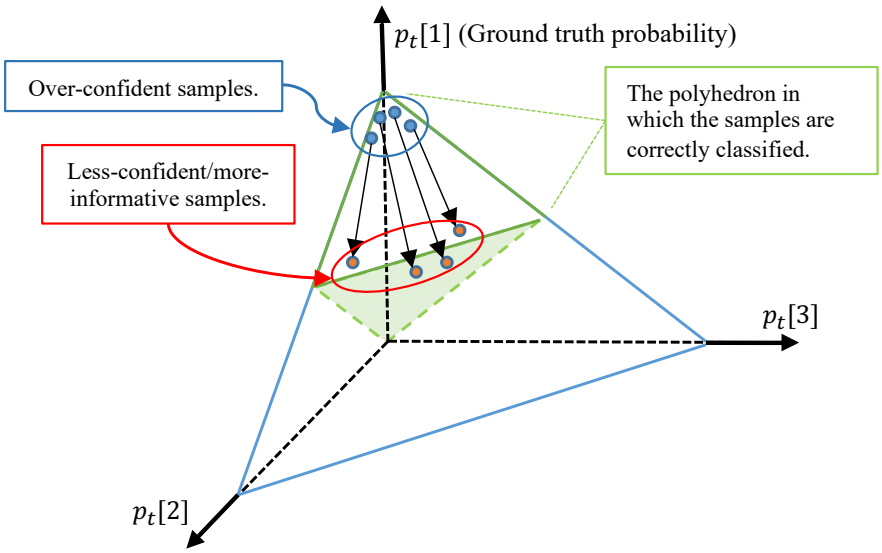


Figure 5.1: The simplex of output probability vector for a classification task with three classes. Our goal is to alter the original samples with peaky output probabilities (the blue dots) in order to obtain samples whose output probabilities are more uniform (the red dots).

are close to one-hot vectors. We refer to such probability vectors as *peaky* ones hereafter. In this case, the model shows over-confident responses to these specific samples and, in turn, cannot generalize well when fed with new unseen data of a similar type [92, 110].

To prevent peaky probability vectors from being passed to the student during the distillation process, it is desirable to alter the respective input images so that (i) the altered input images are still similar to the original ones; and (ii) the output probability vectors of the teacher in response to the altered input images are closer to a uniform distribution while remaining on the surface of the correct polyhedron, similar to the red samples in Fig. 5.1. One way to achieve this is through compression, which will be discussed in the next section.

5.4 *Coded* Knowledge Distillation

As discussed in Section 5.3.2, we have to manipulate the training samples so that the teacher's output probabilities for the new samples are more uniform. We aim to realize this by compressing the input samples to the pre-trained teacher model during the distillation process. Specifically, we deploy JPEG, the widely-used lossy compression technique. The key rationales behind utilizing JPEG are (i) its popularity as a lossy compression technique, and (ii) its potential for improving accuracy as pointed out by Yang *et al.* [131].

It is widely recognized that naively compressing all the input images to a DNN using the same QF by JPEG will generally lead to a degradation in the classification accuracy [101, 131, 144]. However, Yang *et al.* [131] demonstrated that if QF can be adaptively selected on a per image basis, then JPEG compression can significantly enhance classification. Nevertheless, in their framework, the adaptive selection of QF depends on the availability of the ground truth label of the original image, which makes it unrealistic during the testing stage. Inspired by this, we here employ an adaptive selection of QF during the distillation process instead for the teacher's input only, where a ground truth label is available and can indeed be utilized. That is, we adaptively select an image-dependent QF for JPEG to compress each image before it is fed to the teacher during the distillation process. To achieve this, an adaptive encoding layer is introduced in front of the teacher model. The combination of this new adaptive encoding layer with the originally pre-trained teacher forms what we refer to as a *coded* teacher. The output $\hat{x}_{j^*}$ from the adaptive encoder layer generally depends on the input x, the ground truth label c, and the teacher model itself. In the subsequent subsection, we elaborate on the inner-working of the utilized adaptive encoder layer.

5.4.1 Optimal Adaptive JPEG Selection

To realize the proposed adaptive JPEG selection, for each sample image x, we first generate a set of its compressed versions as explained in subsection 5.4.1; then, in subsection 5.4.1, we propose a systematic method to select one element from this set as the input to the teacher.

The set of compressed versions

Consider a predetermined set of QFs: $\mathcal{Q} = \{q_j \in \mathbb{N} \mid 0 \leqslant q_j \leqslant 100\}$, where $q_j = \delta \times j$, and δ is used as a step-size between two successive QF values in $\mathcal{Q}$. Denote by $\hat{x}_j$ the compressed version of x using QF equal to q_j. Then, for each image x in the training set, we construct the following set $\mathcal{J}_x = \{\hat{x}_j \mid q_j \in \mathcal{Q}\} \cup \{x\}$. Specifically, $\mathcal{J}_x$ contains the original training sample image x and its different compressed versions obtained using different $q_j \in \mathcal{Q}$. Therefore, the size of the $\mathcal{J}_x$ will depend on the selected value of δ, and obtained as $|\mathcal{J}_x| = \lfloor \frac{100}{\delta} \rfloor + 2$.

Selection criterion

With the set $\mathcal{J}_x$ created, this section focuses on presenting a systematic method to select an element from $\mathcal{J}_x$. This selection mechanism should ensure that both *Conditions (I)* & *(II)*—introduced in Section 5.1—are satisfied. As such, we first satisfy *Conditions (I)* by finding a subset of $\mathcal{J}_x$ whose elements are correctly classified by the teacher; that is, we find the set $\mathcal{J}_x^{Correct}$ [1] as follows

$$\mathcal{J}_x^{Correct} = \{\hat{x}_j \in \mathcal{J}_x \mid \arg\max_{z \in \mathcal{C}} p(z|\hat{x}_j; \theta_t) = c\}, \tag{5.3}$$

[1] If $\mathcal{J}_x^{Correct}$ is empty, we use the original sample.

where $\mathcal{C}$ represents the set of classes. Afterward, we aim to satisfy *Condition (II)* on top of *Conditions (I)*. To achieve this objective, given that the teacher's output probability for the original sample $p(x;\theta_t)$ is typically peaky, we select $\hat{x}_j \in \mathcal{J}_x^{Correct}$ whose output probability vector $p(\hat{x}_j;\theta_t)$ has the maximum distance to $p(x;\theta_t)$. For this purpose, among the possible distance metrics for the probability vectors, we utilize KL divergence. We refer to such selection mechanism as high KL (HKL) divergence selector. Henceforth, we find $\hat{x}_{j^*}$ as

$$\hat{x}_{j^*} = \underset{\hat{x}_j \in \mathcal{J}_x^{Correct}}{\arg\max} \; \mathcal{KL}\Big(p(x;\theta_t) \,\Big\|\, p(\hat{x}_j;\theta_t)\Big). \tag{5.4}$$

Using this methodology, in the *coded* teacher, for every original training sample image x, we select $\hat{x}_{j^*}$ given in Eq.5.4 as the teacher's input during the distillation process. The CKD framework along with its adaptive JPEG encoding layer are depicted in Fig. 5.2.

5.4.2 Analyzing the Proposed Adaptive JPEG Selection

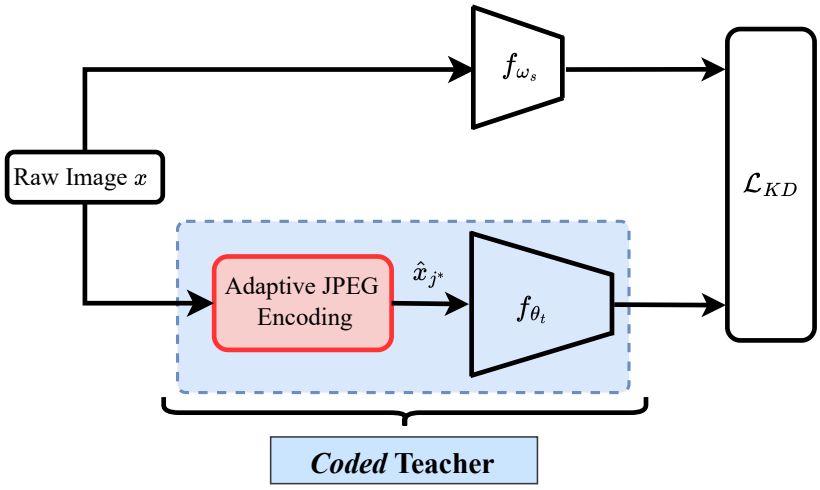
In this subsection, we provide justifications for the proposed adaptive selection mechanism by presenting compelling experimental results. For this purpose, we use the following two values to evaluate whether a probability vector is peaky:

- The magnitude of ground truth probability, in the sense that a smaller magnitude tends to result in a smoother and less peaked probability distribution.
- The uniformity of the vector; that is, a more uniform vector is less peaked. Among the various existing metrics for measuring the uniformity of a vector, we specifically use entropy in our experiments. As such, a probability vector with higher entropy is more uniform.

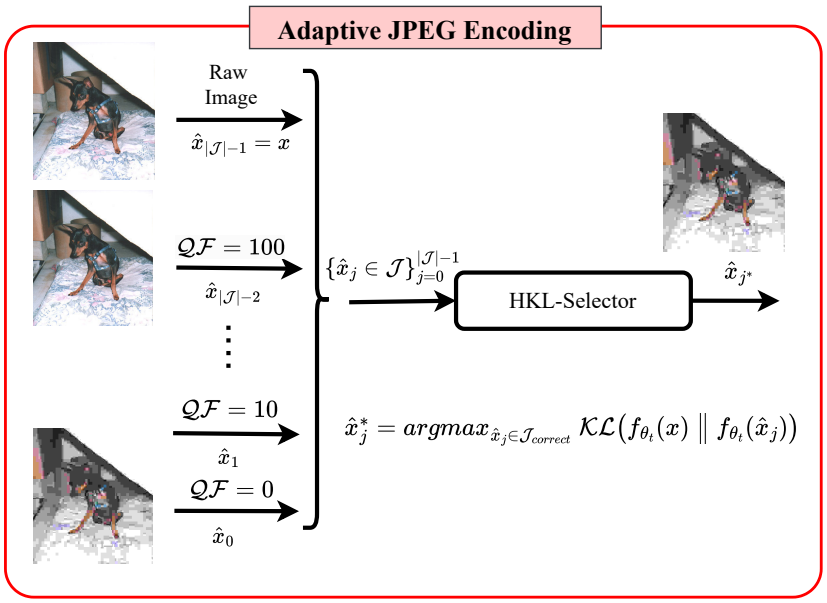
By using these two metrics, we can assess the effectiveness of the adaptive selection method in promoting a more uniform distribution of output probabilities. Now, consider the following three experiments.

Experiment one: Using randomly picked samples from CIFAR100's training set, we want to observe the elements in the set we constructed in subsection 5.4.1 to assess their level of peakiness or uniformity. To illustrate this, we utilize a pre-trained Resnet56 model, serving as the teacher, trained on the CIFAR100 dataset. We then randomly select two images, Img1 and Img2, belonging to different classes that are correctly and wrongly classified by the teacher, respectively. Then, we construct the sets $\mathcal{J}_{\text{Img1}}$ and $\mathcal{J}_{\text{Img2}}$ for these two samples (as explained subsection 5.4.1). A caricature of decision boundary for the samples in $\mathcal{J}_{\text{Img1}}$ and $\mathcal{J}_{\text{Img2}}$ are depicted in Fig. 5.3 in which we use the following notation: (i) 'KL' represents $\mathcal{KL}\big(p(x;\theta_t) \,\big\|\, p(\hat{x}_j;\theta_t)\big)$, (ii) 'H' is the entropy of $p(\hat{x}_j;\theta_t)$, and (iii) 'P' is the magnitude of the ground truth (GT) probability. The 'KL' value is written on top of the arrows connecting the original sample x to its compressed versions $\hat{x}_j$. In addition, for each compressed sample, the values for QF, 'H', and 'P' are written in a box beside that sample.

For Img1, (the left figure in Fig. 5.3), we observe that only seven elements in $\mathcal{J}_{\text{Img1}}$ with respective QF=$\{60, 75, 80, 90, 95, 100\}$ are correctly classified. Among these samples, the compression version with



(a) CKD Framework



(b) Adaptive JPEG Encoding Layer

Figure 5.2: Illustration of CKD: (a) the CKD framework with an adaptive JPEG encoding based *coded* teacher; and (b) the inner-working of adaptive JPEG encoding mechanism.

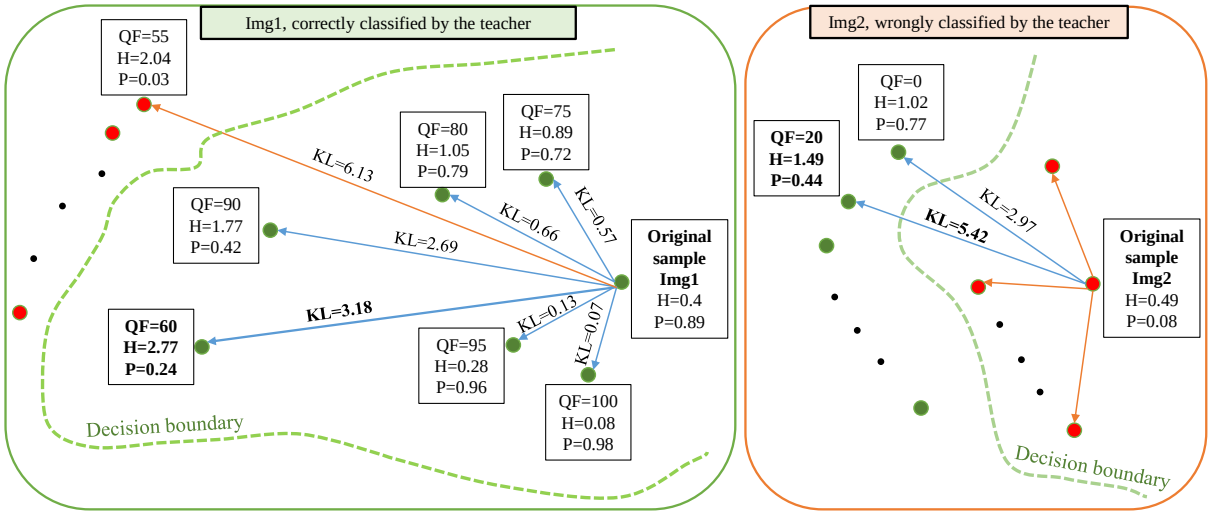


Figure 5.3: The decision boundary plots for Resnet56 pre-trained on CIFAR100 dataset, when the inputs are the elements from the sets: (left) $\mathcal{J}_{\text{Img1}}$, and (right) $\mathcal{J}_{\text{Img2}}$. Green and Red dots show the correctly and wrongly classified samples, respectively. Note that the raw images Img1 and Img2 are correctly and wrongly classified, respectively.

QF=60 will be chosen using the adaptive JPEG selector, as it has the highest 'KL' value. The selected compressed version by the HKL selector has the lowest 'P' and largest 'H' among all correctly classified samples. This demonstrates that the proposed selection method can successfully choose the least peaked sample from the set $\mathcal{J}_{\text{Img1}}^{correct}$.

Now, consider Img2, which is originally misclassified by the model. As seen in Fig. 5.3 (the right figure), there exist some elements in $\mathcal{J}_{\text{Img2}}$ which are correctly classified by the model. Again, the selected compressed version by HKL selector still gives rise to good values of 'P' and 'H'. Note that among all training samples, samples like Img2 are rare, as shown in the upper part of Table 5.1.

Experiment two: In this experiment, we aim to analyse the behaviour of the *coded* teacher over all the training samples (unlike the Experiment one where we only considered two specific samples), and compare it with that of the original uncoded teacher. In fact, we want to demonstrate that the adoption of such an adaptive selection method assists the teacher model in avoiding the generation of *peaky* probabilities. To this end, we report the following two values for the teacher model: the average (and standard deviation) of (i) the entropy of the output probability vectors, and (ii) the GT probability, where the average and standard deviation are computed over all training samples.

Again, we use CIFAR100 dataset. The average entropy and GT probability are computed for both the *coded* teacher and the original uncoded teacher. They are shown in Table 5.1 for various model architectures to illustrate the generality of the proposed method. As seen in Table 5.1, compared to the original teacher, the *coded* teacher gives rise to higher average entropy and smaller ground truth probability

Teacher Models	*Coded* Teacher		Original Teacher	
	Entropy	GT Probability	Entropy	GT Probability
Resnet56	1.4941 ($\pm$ 0.6634)	0.5184 ($\pm$ 0.1992)	0.24531 ($\pm$ 0.3816)	0.9275 ($\pm$ 0.1476)
Resnet110	1.3312 ($\pm$ 0.6513)	0.5607 ($\pm$ 0.2032)	0.10879 ($\pm$ 0.2195)	0.9730 ($\pm$ 0.0781)
VGG13	1.6524 ($\pm$ 0.8211)	0.5344 ($\pm$ 0.2205)	0.03797 ($\pm$ 0.0883)	0.9937 ($\pm$ 0.0265)
WRN-40-2	1.5653 ($\pm$ 0.7578)	0.5244 ($\pm$ 0.2121)	0.07115 ($\pm$ 0.1584)	0.9859 ($\pm$ 0.0458)
ResNet50	1.4399 ($\pm$ 0.7689)	0.5662 ($\pm$ 0.2207)	0.02423 ($\pm$ 0.0648)	0.9960 ($\pm$ 0.0200)
Resnet32$\times$4	1.8490 ($\pm$ 0.9192)	0.5004 ($\pm$ 0.2289)	0.02731 ($\pm$ 0.0656)	0.9962 ($\pm$ 0.0173)

Table 5.1: The average (and standard deviation) of (i) the entropy of p_x and $p_{\hat{x}}$, and (ii) the GT probability in p_x and $p_{\hat{x}}$ from both the *Coded* teacher and original teacher, respectively, over CIFAR100 dataset, where p_x and $p_{\hat{x}}$ are generated using $\tau = 1$. The pre-trained teacher models are also used in [14, 19, 47, 77, 84, 114, 145].

for all the tested models, confirming that the *coded* teacher produces probability vectors more towards the uniform distribution.

Experiment three: Here, we aim to show that CKD enhances the generalization of the student more significantly compared to other KD variants. For this purpose, we have conducted the following experiment: for three teacher-student pairs, we have reported the average (and standard deviation) of (i) the entropy of the student's output probability vectors, and (ii) the student's GT probability for a student trained via some KD variants (see Table 5.2). As depicted in Table 5.2, most of the KD variants result in improvement for the generalization of the student. However, the CKD method exhibits notably a significantly greater improvement and demonstrates orthogonality to temperature scaling in conventional KD. As we will discuss in the following section, it is the heightened level of uniformity passed from the *coded* teacher to the trained student that enables the latter to achieve higher accuracy compared to its baseline distiller. Nevertheless, while training a teacher model with label smoothing increases the degree of uniformity, it can negatively impact the student's accuracy by erasing information contained in intra-class relationships among individual training samples [82]. For more additional analysis, refer to the 5.5.5.

5.5 Experment Results

To evaluate the performance of CKD, we conduct various experiments. In this section, we report our experimental results and compare them with the state-of-the-art alternatives in the literature. In particular, we have carried out experiments on four datasets, namely ImageNet [56] in Section 5.5.1, and CIFAR-100 [55] in Section 5.5.2, and two different datasets for the fine-grained classification in Section 5.5.3. For each of these datasets, both teacher and student models are mentioned, and the training setup is also elaborated. Additionally, we highlight the adaptability of using *coded* teachers by integrating them into other well-established knowledge transfer methods found in the literature. This integration results in an improvement in the test accuracy for each respective approach. All experiments were conducted using

Methods	Resnet56 → Resnet20		Resnet32×4 → Resnet8×4		VGG13 → MobileNetV2	
	Entropy	GT Probability	Entropy	GT Probability	Entropy	GT Probability
KD	0.6423 (± 0.7002)	0.7623 (± 0.2972)	0.5386 (± 0.6714)	0.8457 (± 0.2283)	0.3751 (± 0.5116)	0.8763 (± 0.2117)
FitNet	0.7307 (± 0.7434)	0.7488 (± 0.2937)	0.6253 (± 0.6817)	0.8125 (± 0.2410)	0.4605 (± 0.5389)	0.8600 (± 0.2074)
CC	0.7308 (± 0.7393)	0.7497 (± 0.2923)	0.5574 (± 0.6540)	0.8337 (± 0.2304)	0.6194 (± 0.6498)	0.7958 (± 0.2616)
FT	0.8552 (± 0.8050)	0.7010 (± 0.3155)	0.7492 (± 0.7834)	0.7672 (± 0.2770)	1.1749 (± 0.8641)	0.5945 (± 0.3372)
AB	0.7427 (± 0.7466)	0.7464 (± 0.2931)	0.5111 (± 0.6207)	0.8526 (± 0.2116)	0.4044 (± 0.5054)	0.8768 (± 0.1966)
SP	0.8247 (± 0.7827)	0.7099 (± 0.3129)	0.7595 (± 0.8167)	0.7642 (± 0.2858)	0.5675 (± 0.7730)	0.7215 (± 0.3225)
RKD	0.7547 (± 0.7266)	0.7333 (± 0.3014)	0.6333 (± 0.6655)	0.8143 (± 0.2366)	0.5104 (± 0.5692)	0.8415 (± 0.2224)
FSP	0.8118 (± 0.7905)	0.7188 (± 0.3096)	0.5587 (± 0.6525)	0.8314 (± 0.2327)	n/a	n/a
TALD	0.5396 (± 0.6543)	0.7430 (± 0.3412)	0.4013 (± 0.6306)	0.8448 (± 0.2761)	0.3333 (± 0.5350)	0.8070 (± 0.3293)
CKD	**0.9511 (± 0.8326)**	**0.7001 (± 0.3018)**	**0.9651 (± 0.9541)**	**0.7764 (± 0.2544)**	**0.6468 (± 0.7211)**	**0.8367 (± 0.2257)**

Table 5.2: The average (and standard deviation) of (i) the entropy of p_x, and (ii) the GT probability in p_x from students trained by CKD with the *Coded* teacher and other knowledge transfer methods with the original teacher on the CIFAR-100 dataset, where p_x is generated using $\tau = 1$.

Accuracy	Teacher	Student	KD [44]	AT [140]	SP [117]	CC [90]	O-KD [147]	CKD
Top-1	73.31%	69.76%*	70.66%	<u>70.70</u>%	70.62%	69.96%	70.55%	**70.98**%
Top-5	91.42%	89.08%	89.88%	<u>90.00</u>%	89.80%	89.17%	89.59%	**90.04**%

Table 5.3: The student accuracy on ImageNet dataset given by different KD methods, where the best and second-best results are **bold** and <u>underlined</u>, respectively. The experiment setup is the same as in [114]. *As we followed [75] for our experimental setups, the accuracy for the Resnet18 model differs from that reported in [41].

PyTorch [89].

5.5.1 ImageNet Dataset

• **Dataset:** We use ImageNet ILSVRC 2012 dataset that contains 1.2M training images and 50K for testing images [56] with an average size of 469 × 387 to allow us to verify distillation performance for large input resolutions.

• **Teacher and student models:** We use pre-trained Resnet34 as the teacher and Resnet18 as the student model [41]. Following the experimental setup in [14, 19, 47, 77, 114, 145] , the pre-trained teacher model is obtained from torchvision library [75].

• **Training setup:** The training is performed for 100 epochs with a batch size of 256 using SGD with a momentum of 0.9 and weight decay equal to 1×10^{-4} with an initial learning rate γ of 0.1 that is divided by 10 at epochs 30, 60 and 90. Standard Inception-style pre-processing [110] for augmentation settings is used in the training of teacher models as well as students. For CKD, we set $\delta = 10$ resulting in $|\mathcal{J}_x| = 12$ versions of the raw input.

Teacher Student	Resnet56 Resnet20	Resnet32x4 Resnet8x4	VGG13 MobileNetV2	ResNet50 MobileNetV2
Student	69.06%	72.50%	64.6%	64.6%
KD [44]	70.66%	73.33%	67.37%	67.35%
AT [140]	70.55%	73.44%	59.40%	58.58%
VID [1]	70.38%	73.09%	65.56%	67.57%
PKT [88]	70.34%	73.64%	67.13%	66.52%
NST [48]	69.60%	73.30%	58.16%	64.96%
BSS [42]	70.70%	73.53%	67.43%	68.10%
TALD [84]	**70.90%**	<u>73.73%</u>	**68.50%**	**68.70%**
CKD	<u>70.81%</u>	**74.42%**	<u>67.80%</u>	<u>68.41%</u>

Table 5.4: Test accuracy (%) of students trained by different KD methods on CIFAR100, where the best and second-best results are **bold** and <u>underlined</u>, respectively. The student accuracy for TALD and BSS is obtained using the settings specified in [84].

• **Results and analysis:** The results are presented in Table 5.3, where the performance of CKD is compared with some knowledge transfer benchmarks. We note that we used O-KD to denote "Online KD" method [147]. Specifically, the results demonstrate that the *coded* teacher effectively improves the performance of the conventional KD, surpassing the performance of the AT distillation method. In addition, further experiments on the subset of ImageNet, namely ImageNet-1K subset, are presented in 5.5.4.

5.5.2 CIFAR100 dataset

• **Dataset:** This dataset contains 50K training images with 0.5K images per class and 10K test images [55].

• **Teacher and student models:** We use different pairs of teacher and student models, considering both similar and different architectural styles. Specifically, inspired from [14, 19, 47, 77, 114, 145], the teacher models are chosen among ResNet50, Resnet56, Resnet110, VGG13, WRN-40-2, and Resnet32x4.

• **Training setup:** When training CIFAR-100, we start with an initial learning rate of 0.01 or 0.05. After the first 150 epochs, we decrease the learning rate by a factor of 0.1 every 30 epochs until reaching the last 240 epochs. The initial learning rate is 0.01 for MobileNetV2, ShuffleNetV1, and ShuffleNetV2, and 0.05 for the other models. For CKD, we use $\delta = 5$, yielding $|\mathcal{J}_x| = 22$. The training setup follows the commonly used approach introduced in [14, 19, 47, 77, 114, 145].

• **Results and analysis:** We conduct three sets of experiments explained below.

Set 1: We first compare the performance of CKD against conventional KD methods. In addition, we compare CKD to other perturbation-based KD methods, namely BSS [42] and TALD [84]. As discussed in Section 5.2, these two methods are similar to CKD in the sense that both effectively perturb the inputs

to the pre-trained teacher model. We use the same student-teacher pairs as those used in [84], where both the same and different architectural styles were considered. The comparison is reported in Table 5.4.

Remarkably, CKD outperforms KD and its other four variants, namely AT, VID, PKT and NST. In particular, CKD exhibits an average gain of 0.67% compared to conventional KD. This superior performance is also evident in comparisons with BSS. However, TALD generally ranks as the best among the benchmark methods (except for one teacher-student pair). It is worth noting that both BSS and TALD have a higher level of complexity compared to CKD, as the former methods involve input perturbation using gradients (see Section 5.2). To ensure the accuracy and reproducibility of the results, we re-run the implementation provided by the authors in [84] to produce the results in Table 5.4. By utilizing their implementation [2], we confidently validated and directly compared these methods with the performance reported in their results, ensuring a reliable evaluation.

Set 2: In this set of experiments, we compare the *coded* teacher with TALD when both are applied on top of various knowledge transfer methods including FT, FitNet, CC, SP, and RKD (see Fig. 5.4). The cost function of these methods can be represented as $\mathcal{L} = \mathcal{H} + \beta\mathcal{L}_{\text{distill}}$[3], where $\mathcal{L}_{\text{distill}}$ denotes the loss function associated with a specific form of knowledge distilled into the student model. When applying the *coded* teacher on top of these methods, we tune β accordingly. From Figs. 5.4a and 5.4b, it is clear that the coded teacher delivers significant gains. Specifically, with VGG13 as the teacher model, the *coded* teacher shows notable gains of 5% and 2.12% when compared to TALD for FT and FitNet, respectively. Similarly, when ResNet50 is used as the teacher model, the *coded* teacher achieves gains of 4% and 1.3% compared to TALD for FT and FitNet, respectively. These results demonstrate the consistent improvement achieved by the *coded* teacher.

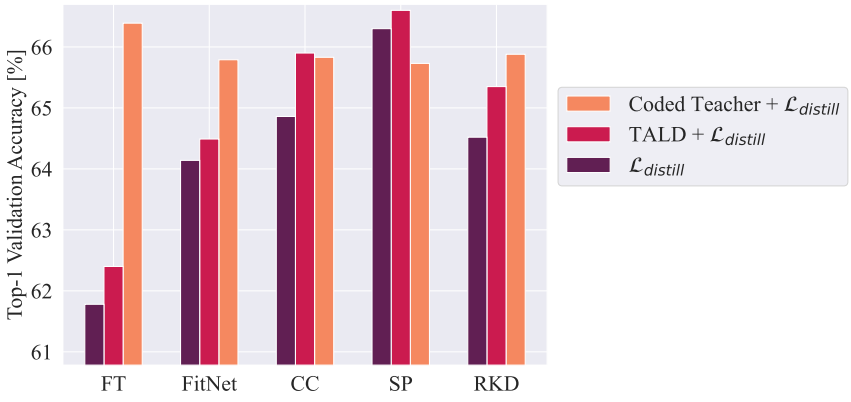
Set 3: In the third set of experiments, we want to show whether the *coded* teacher, when applied on top of existing knowledge transfer methods, can improve their accuracy performance of the latter. Particularly, we apply the *coded* teacher on top of the following knowledge transfer methods in the literature: FitNet [96], FSP [138], FT [52], CC [90], SP [117], AB [43], RKD [86]. In our experiments, FitNet and FT were re-implemented based on the original papers; other methods were implemented, using either author-supplied or author-verified source codes. For a fair comparison, similarly to [14, 19, 47, 77, 84, 114, 145], we use the same pre-trained teacher models for all the benchmarks as those used in [114][4].

Tables 5.5 and 5.6 illustrate the resulting accuracy results for student-teacher pairs with the same architectural styles and for student-teacher pairs with different architectural styles, respectively. From Tables 5.5 and 5.6, it is clear that the coded teacher can improve the accuracy performance of the underlying

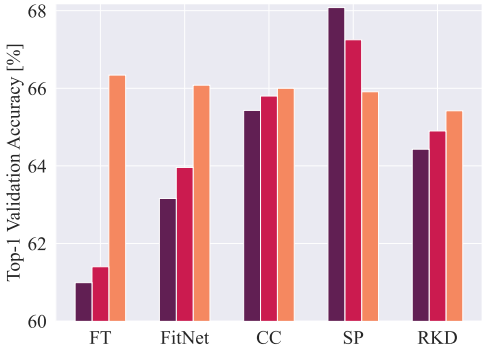
[2]The author-supplied TALD code for CIFAR100 in https://github.com/PotatoThanh/Adversarial-local-distribution-regularization-for-knowledge-distillation. For ImageNet, we attempted to reproduce their results, but encountered significant computational complexity, as explicitly noted by the authors. In addition, their experimental setup also deviates from the standards in the literature.

[3]In the TALD framework, KD loss is incorporated for all other distillation methods, plus the TALD loss, to enhance their performance. However, CKD operates directly with other distillation methods without the addition of KD loss.

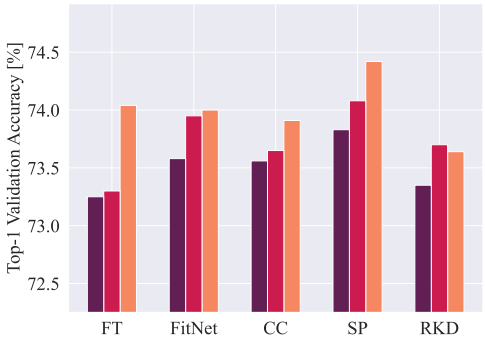
[4]We used the implementation of other methods from https://github.com/HobbitLong/RepDistiller.



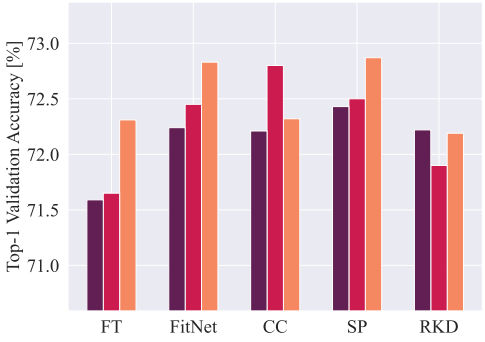
(a) VGG13→ MobileNetV2



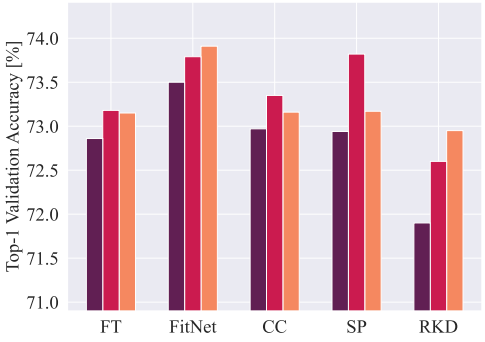
(b) ResNet50→ MobileNetV2



(c) WRN40-2 → WRN16-2



(d) WRN40-2 → WRN40-1



(e) Resnet32×4 → Resnet8×4

Figure 5.4: Comparison between the *coded* teacher and TALD when they are applied on top of different underlying knowledge transfer methods. The teacher-student pairs are reported in the caption of each figure.

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	Resnet56 Resnet20	Resnet110 Resnet20	Resnet110 Resnet32	Resnet32x4 Resnet8x4	VGG13 VGG8
Teacher	75.61%	75.61%	72.34%	74.31%	74.31%	79.42%	74.64%
Student	73.26%	71.98%	69.06%	69.06%	71.14%	72.50%	70.36%
CC	73.56%	72.21%	69.63%	69.48%	71.48%	72.97%	70.71%
CCC	**73.91**%	**72.32**%	**69.78**%	**70.21**%	**72.13**%	**73.16**%	**71.31**%
FitNet	73.58%	72.24%	69.21%	68.99%	71.06%	73.50%	71.02%
CFitNet	**74**%	**72.83**%	**69.49**%	**69.50**%	**71.66**%	**73.91**%	**71.14**%
FT	73.25%	71.59%	69.84%	70.22%	**72.37**%	72.86%	70.58%
CFT	**74.04**%	**72.31**%	**69.90**%	**70.25**%	71.87%	**73.15**%	**71.07**%
AB	**72.50**%	72.38%	69.47%	69.53%	70.98%	73.17%	70.94%
CAB	71.8%	**73.03**%	**70.11**%	**69.97**%	**72.11**%	**73.33**%	**71.35**%
SP	73.83%	72.43%	69.67%	70.04%	**72.69**%	72.94%	**72.68**%
CSP	**74.42**%	**72.87**%	**69.88**%	**70.10**%	71.98%	**73.17**%	71.85%
RKD	73.35%	**72.22**%	69.61%	69.25%	**71.82**%	71.90%	**71.48**%
CRKD	**73.64**%	72.19%	**69.85**%	**69.69**%	71.76%	**72.95**%	71.07%
FSP	72.91%	n/a	69.95%	70.11%	71.89%	72.62%	**70.23**%
CFSP	**73.13**%	n/a	**70.33**%	**70.37**%	**72.39**%	**73.08**%	69.10%

Table 5.5: Test accuracy (%) of students on CIFAR100 dataset before and after the *coded* teacher is applied on top of the underlying distillation methods when the teacher and student pair has the same architectural style, where each underlying distillation method, when coupled with the coded teacher, is indicated by the bold prefix **C** and referred to as the corresponding coded distillation method. The accuracy for the underlying distillation methods is reported from [19, 114], and the pre-trained teacher models are also used in [14, 19, 47, 77, 84, 114, 145].

distillation method in most cases. In particular, when the student-teacher pair has different architectural styles, the accuracy gain can be above 5%.

5.5.3 Fine-grained Classification Task

• **Datasets:** This task involves evaluating datasets to distinguish visually similar objects in the same category. The first dataset, CUB200 (Caltech-UCSD Birds-200-2011), has 11,788 images of 200 bird subcategories divided into 5,994 images for training, and each subcategory in the training set contains roughly 30 images. The second dataset, Stanford Dogs, has 20,580 images of 120 species, with 12K images for training, each class having 180 images, and the rest for validation.

• **Teacher and student models:** We use wide residual networks (WRN) with different depths and fixed widths. We utilize the same student-teacher pairs for both datasets. For our teachers, we utilize pre-trained WRN-64 and WRN-28 models, which achieved validation accuracies of 38.57% and 42.32% on

Teacher Student	VGG13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 VGG8	Resnet32x4 ShuffleNetV1	Resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
Teacher	74.64%	79.34%	79.34%	79.42%	79.42%	75.61%
Student	64.6%	64.6%	70.36%	70.5%	71.82%	70.5%
CC	64.86%	65.43%	70.25%	71.14%	71.29%	71.38%
CCC	**65.83%**	**66%**	**71.04%**	**72.02%**	**73.40%**	**72.02%**
FitNet	64.14%	63.16%	70.69%	73.59%	73.54%	**73.73%**
CFitNet	**65.79%**	**66.08%**	**71.07%**	**73.62%**	**74.48%**	73.46%
FT	61.78%	60.99%	70.29%	71.75%	72.50%	72.03%
CFT	**66.39%**	**66.34%**	**71.09%**	**72.48%**	**73.73%**	**72.21%**
AB	66.06%	**67.20%**	70.65%	73.55%	74.31%	73.34%
CAB	**66.58%**	66.93%	**71.02%**	**74.58%**	**75.19%**	**74.50%**
RKD	64.52%	64.43%	**71.50%**	72.28%	**73.21%**	72.21%
CRKD	**65.88%**	**65.42%**	71.09%	72.28%	73.06%	**72.89%**

Table 5.6: Test accuracy (%) of students on CIFAR100 dataset before and after the coded teacher is applied to the underlying distillation methods when the teacher and student pair has different architectural styles, where each underlying distillation method, when coupled with the coded teacher, is indicated by the bold prefix **C** and referred to as the corresponding coded distillation method.. The accuracy for the underlying distillation methods is obtained from [19, 114], while the pre-trained teacher models are also used in [14, 19, 47, 77, 114, 145].

the CUB200 dataset, and 46.60% and 54.37% on the Stanford Dog dataset, respectively.

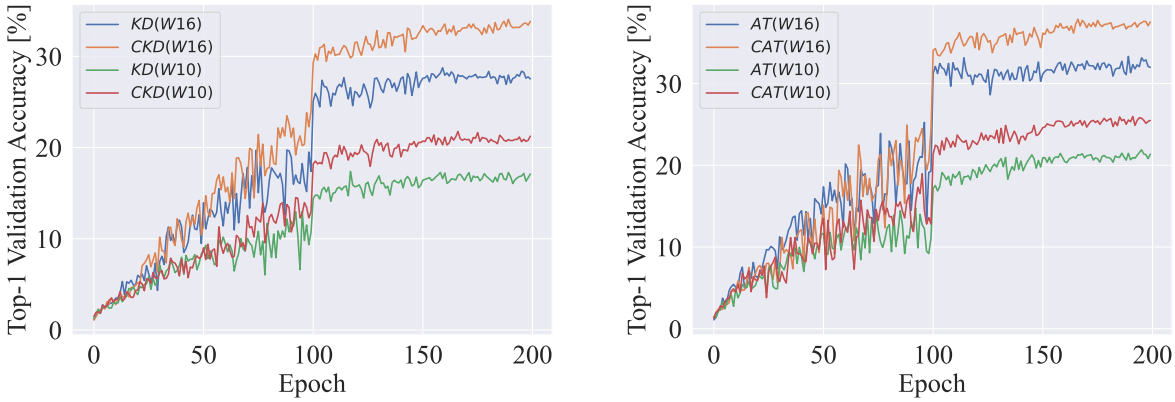
• **Training setup:** We use $\delta = 5$ yielding $|\mathcal{J}_x| = 22$. In our experiments, we use WRN-<d>-<w> as our baseline focusing on increasing the depth <d> and fixing the width <w> to 1, with the dropout probability of 0.3. The selected student models are WRN-10 (W10) and WRN-16 (W16), with 85K and 182K parameters, respectively. Following the settings in [58, 139], we trained the networks from scratch for 200 epochs with a batch size of 32 using SGD with a momentum of 0.9 and weight decay equal to 1×10^{-4} with an initial learning rate γ of 0.1 that is divided by 10 at epochs 100 and 150. The training of both teacher and student models used the same training setup and the standard Inception-style preprocessing for augmentation settings [110].The same values $\tau = 1$ and $\alpha = 0.5$ are used for both the KD and CKD.

• ***Coded* AT:** In addition to comparing CKD with KD, we also apply the coded teacher on the top of AT distillation method [140], yielding *coded* AT (CAT), and further compare CAT with AT. In this case, the β parameter in AT is set to 1000 as recommended in [140] .

Table 5.7 shows the resulting accuracy results. From Table 5.7, it is clear that CKD and CAT improve KD and AT by a significant margin, respectively. To shed some light on the respective training processes, Fig. 5.5 further illustrates the respective Top-1 validation accuracy curves along the number of training epochs in the case of CUB200 dataset. As shown in Fig. 5.5, the CKD significantly outperforms the

Dataset		CUB200		Stanford Dog	
Teacher \ Student		WRN-10 15.24%	WRN-16 27.63%	WRN-10 23.73%	WRN-16 34.79%
WRN-28	KD	17.68% (±1.21)	29.10% (±0.76)	25.79% (±0.88)	38.13% (±1.35)
	CKD	**21.27%(±0.52)**	**31.84%(±0.40)**	**28.32%(±0.70)**	**41.17%(±0.74)**
	AT	22.20% (±1.15)	32.94% (±0.65)	26.97% (±0.47)	39.04% (±0.57)
	CAT	**24.82%(±0.47)**	**37.10%(±0.10)**	**29.54%(±0.46)**	**41.84%(±0.33)**
WRN-64	KD	18.04% (±0.60)	29.9% (±1.04)	26.76% (±1.16)	38.30% (±1.63)
	CKD	**21.41% (±0.44)**	**33.1% (±0.89)**	**29.69% (±0.99)**	**41.66% (±0.38)**
	AT	22.55% (±0.82)	33.95% (**±0.71**)	27.14% (±1.36)	39.39% (±1.03)
	CAT	**24.94% (±0.92)**	**37.31%** (±0.77)	**29.73% (±0.41)**	**41.93% (±0.65)**

Table 5.7: The performance of CKD and CAT against conventional KD and AT.



(a) KD for WRN-64

(b) AT for WRN-64

Figure 5.5: Validation accuracy curves along the number of epochs over CUB200 dataset with (WRN-64) as the teacher and (W10) and (W16) as students: (a) **C**KD vs KD; and (b) **C**AT vs AT.

underlying distillation method around the 100-th epoch and stays that way afterwards in all tested cases.

5.5.4 More Evaluation on ImageNet Subset

• **Dataset:** We use a subset of the original ImageNet ILSVRC 2012 dataset [56], namely ImageNet-1K subset, which has also been used in the literature [16, 54]. This subset shares the same validation set as

Training Setup			Without Augmentation		With Augmentation	
Teacher \ Student			Resnet18	MobileNet-V2	Resnet18	MobileNet-V2
Resnet34	KD	T1:	40.28%	49.27%	47.68% (±0.25)	52.65% (±0.44)
		T5:	67.07%	75.52%	73.81% (±0.28)	78.01% (±0.35)
	CKD	T1:	**41.27%**	**49.87%**	**49.74%(±0.09)**	**54.39%(±0.18)**
		T5:	**67.86%**	**76.21%**	**75.56%(±0.17)**	**79.57%(±0.07)**
Resnet50	KD	T1:	44.51%	51.43%	49.37% (±0.37)	53.93% (±**0.16**)
		T5:	71.45%	77.46%	75.42% (±0.24)	79.21% (±**0.14**)
	CKD	T1:	**46.78%**	**52.78%**	**51.82%(±0.21)**	**55.11%**(±0.28)
		T5:	**73.17%**	**78.46%**	**77.08%(±0.06)**	**80.31%**(±0.21)

Table 5.8: CKD performance on ImageNet-1K subset with and without data augmentation.

the original ImageNet dataset. The training subset[5] is subsampled in a label-balanced fashion to result in a 1% configuration used in our experiments.

• **Training setup:** In the KD loss function in Eq.equation 5.1, α was set to 0.5. To determine the optimal value of τ for conventional KD, we conducted experiments for $\tau \in \{1, 2, 3\}$, and selected the one that yields the best performance. Specifically, we use $\tau = 2$ and $\tau = 3$ for Resnet34 and Resnet50, respectively. Then, the same values for α and τ are used in the CKD framework. This step was essential in ensuring the consistency of the source of gain between the conventional KD and CKD frameworks.

• **Results and analysis:** The results are reported in Table 5.8, encompassing two distinct columns outlining CKD with and without input augmentation, which can also serve as an ablation study. It is worth noting that the authors in [62] observed that various knowledge distillation approaches, such as vanilla KD and CRD [44, 114], can also achieve a modest improvement in the student model's performance solely through applying data augmentation. As such, the objective of conducting CKD with and without data augmentation was to demonstrate its efficacy in both scenarios.

As the first main column suggests (corresponding to without augmentation), CKD outperforms KD in both Top-1/Top-5 (T1/T5) validation accuracy. In the second column (with augmentation), we followed the best practice in the literature by performing three random runs with the default augmentation settings. The mean and standard deviation for T1/T5 validation accuracy is shown in Table 5.8. The Top-1 validation accuracy indicates 2.18% and 2.32% improvement over the standard KD when Resnet34 and Resnet50 are the teacher models, respectively.

• **Ablation study on the adaptive layer in CKD:** To elucidate the necessity of the adaptive layer, we remove this layer and perform the following two sets of experiments: (1) using a constant QF=50 for all images, and (2) using a random QF for each image ranging from 50 to 100. The results for these experiments are listed in 5.9. Compared to conventional KD and CKD in Table 5.8, these results suggest the necessity of using an adaptive layer method in all setups.

[5]For detailed information about the sampling methodology and specific subsets used in the experiments, please refer to https://www.tensorflow.org/datasets/catalog/imagenet2012_subset.

Teacher \ Student			Resnet18	MobileNet-V2
Resnet34	KD+QF50	T1:	48.11%	53.57%
		T5:	74.24%	78.91%
	KD+R(QF)	T1:	48.64%	53.48%
		T5:	74.68%	78.84%
Resnet50	KD+QF50	T1:	49.86%	54.53%
		T5:	75.61%	79.89%
	KD+R(QF)	T1:	49.89%	54.13%
		T5:	75.76%	79.42%

Table 5.9: Ablation study on the adaptive layer in CKD on ImageNet dataset subset.

5.5.5 Analyzing student's generalization error in CKD

In this section, we aim to demonstrate the reason why CKD reduces the generalization error of the student model. To this end, we will first establish a clear definition of generalization error.

In a classification task with C classes, a DNN can be regarded as a mapping $f_\theta : x \mapsto p = p(x;\theta)$, where θ represents all the model parameters, $x \in \mathbb{R}^d$ is an input image, and $p \in \Delta^{C-1}$ is a probability vector over the C classes. One may learn such a classifier by minimizing the **true** risk $R(f_\theta)$, defined as:

$$R(f_\theta) \triangleq \mathbb{E}_{(x,\hat{y})}\left[\ell\big(\hat{y}, p(x;\theta)\big)\right], \tag{5.5}$$

where $\ell(\cdot,\cdot)$ is the loss function and $\ell(\cdot,\cdot) \triangleq [\ell(1,\cdot),\ldots,\ell(C,\cdot)]$ is the vector of loss function.

On the other hand, to understand how KD works, let us write the loss function for the student in KD. To this end, denote by θ_t and ω_s the parameters of the pre-trained teacher and student networks, respectively. Then, the student uses the training dataset $\mathcal{D} \triangleq \{(x_n,\hat{y}_n)\}_{n=1}^N$ sampled from the joint distribution $P_{(x,\hat{y})}$ in order to minimize the following distillation risk function:

$$R_{\mathrm{S}}(f_\theta;\mathcal{D}) \triangleq \frac{1}{N}\sum_{n\in[N]}\big(p_t(x_n;\theta_t)\big)^T \cdot \log\big(p_s(x_n;\omega_s)\big), \tag{5.6}$$

Then, the generalization error for the student model is defined as follows [76]:

$$\mathbb{E}\Big\{\Big(R_{\mathrm{S}}(f_\theta;\mathcal{D}) - R(f_\theta)\Big)^2\Big\}. \tag{5.7}$$

The smaller this value, the lower the generalization error is.

In order to provide justification about why the student model trained by CKD generalizes better than that trained by KD, we use the theoretical results derived in [122]. Particularly, the authors in [122]

Teacher	*Coded* teacher	Original teacher
Resnet56	13.0308	40.7161
Resnet32×4	23.5506	29.1309
VGG13	3.7323	58.7183
ResNet50	16.2534	61.1902

Table 5.10: Covariance ($\times 10^{-5}$) for both *coded* and original teacher on the CIFAR-100.

showed that the generalization error for the student is lower when the teacher's output vectors exhibit a lower covariance.

Hence, by conducting some experiments, we show that the outputs of the *coded* teacher exhibit a lower covariance than those of a conventional teacher. Specifically, we use exactly the same method as that used in [122] to estimate the covariance of the teacher's output probability. The results for four teacher models trained on CIFAR-100 dataset are listed in Table 5.10 which suggest the output probability vectors of a *coded* teacher have lower covariance than those of an original teacher.

5.6 Chapter Summary and Outlook on Upcoming Challenges

In this chapter, we have proposed a novel KD framework referred to as *coded* KD (CKD). In CKD, we insert an additional adaptive encoding layer in front of the teacher, yielding a *coded* teacher. The purpose of the *coded* teacher is to generate less confident output responses to be passed to the student so as to help the student generalize better. A specific JPEG adaptive encoding layer has been presented. The effectiveness and accuracy performance advantage of CKD have been demonstrated via comprehensive experimental results when it is compared with the conventional KD method and when the coded teacher is applied on top of diverse underlying knowledge transfer methods.

Two areas for improvement in CKD are: (1) its time complexity; and (2) the sub-optimality of the proposed JPEG adaptive encoding. Compared to the conventional KD, CKD has a slight increase in time complexity due to using adaptive image encoding. Nonetheless, this time complexity increase is significantly lower than that incurred by other knowledge transfer methods using input perturbation, such as BSS and TALD. In our future work, we will address the sub-optimality of adaptive JPEG encoding by exploring more advanced and generalized encoding methods to further improve the accuracy performance of CKD.

Chapter 6

Generalized Coded Knowledge Distillation

In this chapter, we explore the use of maximizing conditional mutual information (CMI) to enhance the effectiveness of teacher networks in knowledge distillation (KD). Prior work achieves this by fine-tuning a pretrained teacher to maximize a proxy of its CMI. However, this strategy faces two key limitations: (1) modifying a high-performing teacher model may be impractical or undesirable in real-world applications, and (2) optimizing a proxy for CMI introduces estimation errors that can hinder performance. To address these challenges, we propose Generalized Coded Knowledge Distillation (GCKD)—a novel framework that maximizes the true CMI of the teacher without altering its parameters. GCKD perturbs the teacher's input space by leveraging adversarial attack methods, which generate perturbed examples—referred to as the *coded dataset*—by adding carefully crafted perturbations within a bounded constraint to clean inputs. These informative examples are optimized using a novel alternating algorithm designed to directly and accurately maximize CMI. Extensive experiments on CIFAR-100 and ImageNet, across diverse distillation methods and network architectures, demonstrate that GCKD consistently improves student model performance while maintaining full compatibility with the standard KD pipeline.

6.1 Introduction

Knowledge distillation (KD) [9, 44] is a cornerstone technique for compressing large models by transferring knowledge from a high-capacity teacher model to a smaller student model. This approach enables student models to achieve strong performance while avoiding the high computational costs associated with training or deploying large models. Compared to other compression strategies like pruning [108] and quantization [65], KD is often easier to implement and more effective at preserving accuracy, particularly in settings where simplicity and efficiency are important. As a result, KD stands out as a practical and reliable solution for deploying models under resource constraints.

Recent research on KD has focused on improving the student's ability to replicate the teacher and on understanding the underlying mechanisms of this process. Many studies have proposed methods to

enhance imitation [47, 52, 77, 86, 90, 96, 109, 114, 117, 140, 143, 145], while others have explored the theoretical foundations of KD [2, 22, 76, 79, 93, 137]. However, most of these approaches assume that the teacher is trained solely to minimize cross-entropy (CE) loss, without considering how the teacher's output distribution influences the student's generalization. This often results in overly confident, near one-hot predictions that reduce the informativeness of the distillation signal [17, 78]. Consequently, the effectiveness of KD can be limited by the quality of the teacher's output, which is frequently overlooked.

To improve the quality of supervision in the KD framework, student-oriented teacher models have been proposed to produce softer and more informative outputs. These approaches aim to enhance the distillation signal by adjusting the teacher's behavior to better support student learning [17, 26, 81, 106, 112, 128, 137]. Notably, MCMI by Ye *et al.* [137] demonstrates that maximizing conditional mutual information (CMI) during teacher training can significantly improve distillation performance. However, these methods typically require modifying the teacher's parameters, which is often impractical in real-world scenarios where the teacher model is fixed or proprietary. This limitation underscores the need for strategies that enhance distillation without altering the teacher.

To overcome this constraint, another line of research has explored introducing perturbations at the input level. In standard KD pipelines, both the teacher and student receive the same input images, which may limit the teacher's ability to fully transfer its representational knowledge. To address this, several studies [42, 84, 141] have proposed using perturbed examples of the same input—generated by the teacher and fed to the student—as an additional regularization term in the distillation objective. This encourages the student to learn the teacher's decision boundaries more effectively. Building on this idea, without introducing additional regularization terms, Coded Knowledge Distillation (CKD) purposed by Salamh *et al.* [98] demonstrates improved performance by proposing that adaptively compressing input images can reduce the teacher's overfitting and provide a more informative signal, while the student continues to receive the raw input. However, these methods often treat each input sample independently, even when samples belong to the same class-defined cluster, without considering the broader impact on the entire cluster. This limitation overlooks the benefits of maximizing conditional mutual information across clusters shown by Ye *et al.* [137], which has been shown to improve the consistency of the supervision signal and enhance the overall effectiveness of the distillation process.

To address these limitations, we introduce *Generalized Coded Knowledge Distillation* (GCKD). GCKD improves the teacher's input by applying carefully crafted perturbations using backpropagation-based gradient methods, generating perturbed examples with bounded magnitude designed to maximize the teacher's conditional mutual information. For instances, consider a classification DNN trained on a dataset $(x, y) \in \mathcal{D}$, modeled as a mapping from raw data $x \in \mathbb{R}^d$ to a probability distribution $f(x)$ over class labels, predicting an output label $\hat{y}$ with probability $f(x)[\hat{y}]$. Drawing inspiration from Yang *et al.* [133], we analyze the information geometry of the mapping $x \rightarrow f(x)$ by considering perturbed counterpart x' of each input x, and its corresponding output propability distribution $f(x')$. The set of perturbed distributions $\{f(x')\}$ for all instances x' sharing the same ground truth label y forms a cluster, referred to as the perturbed cluster corresponding to y, denoted $\mathcal{D}'_y$. Each cluster can be represented by a unique centroid, denoted S'_y, computed as the average of the output probability distributions of all instances $x' \in \mathcal{D}'_y$ within the cluster. Collectively, these clusters form the set $(x', y) \in \mathcal{D}'$, which we refer to as the *coded dataset*

used for teacher distillation.

Following Yang *et al.* [133], the adversarial objective in GCKD is formulated by measuring the concentration of each perturbed cluster using the CMI $I(X'; \hat{Y}' \mid Y = y)$, where X' is an perturbed sample corresponding to a benign sample X, $\hat{Y}'$ is the random label predicted by the DNN in response to X' with probability $f(X')[\hat{Y}']$, and Y is the ground truth label of X. Here, X, Y, X', and $\hat{Y}'$ are all treated as random variables. Averaging over all labels y with respect to the distribution $P_Y(y)$, the CMI $I(X'; \hat{Y}' \mid Y)$ quantifies the average concentration across all perturbed clusters generated by a given adversarial backpropagation-based gradient method. If, for each label y, the centroid of the perturbed cluster is not close to the one-hot probability vector corresponding to y, then a larger value of $I(X'; \hat{Y}' \mid Y)$ indicates a less concentrated cluster, which in turn leads to improved distillation performance. However, this improvement comes with a tradeoff: as the perturbed clusters become less concentrated, the cross-entropy between the teacher's predictions and the true labels may increase, potentially reducing classification fidelity. Therefore, the optimization of $I(X'; \hat{Y}' \mid Y)$ must balance cluster dispersion with the preservation of label information.

Both $I(X'; \hat{Y}' \mid Y = y)$ and $I(X'; \hat{Y}' \mid Y)$ depend on the underlying backpropagation-based method which generates an perturbed instance x' for each (x, y). Due to the diversity of potential backpropagation-based methods and the uncertainty regarding which may be employed against a fixed teacher model, we formulate the CMI maximization problem as identifying the perturbed sample X' that maximizes $I(X'; \hat{Y}' \mid Y)$ under the constraint $d(X', X) \leq \epsilon$, and define:

$$I(\epsilon) = \max_{X'} I(X'; \hat{Y}' \mid Y),$$

where the maximization is taken over all backpropagation-based methods satisfying the perturbation constraint.

It is worth noting that one of the core differences between our methodology for maximizing CMI and the approach proposed by Ye *et al.* [137] lies in how class-wise centroids are handled. While MCMI fixes these centroids during CMI maximization—potentially compromising the accuracy of the CMI proxy—our method introduces a novel alternating optimization algorithm that reformulates the objective as a double minimization problem. This allows centroids to be updated dynamically at each iteration of the proposed alternating algorithm, resulting in more stable and effective training. Overall, GCKD offers a practical and adaptive approach to improving distillation by enhancing the teacher's supervision signal without modifying its core parameters. Moreover, another key distinction from CKD by Salamah *et al.* [98] is that CKD perturbs the raw input by pushing it away from its original representation in a discrete space to mitigate the teacher's overfitting. In contrast, GCKD operates in a continuous space, optimizing perturbations through gradient-based methods to maximize CMI.

To demonstrate the effectiveness of GCKD, we conduct extensive experiments on two benchmark datasets, covering both same- and cross-architecture distillation scenarios. The results consistently highlight the orthogonality of GCKD to various KD pipelines and its superiority over Coded Knowledge Distillation (CKD) [98].

Our main contributions are summarized as follows:

- **Generalized Coded Knowledge Distillation (GCKD):** We introduce GCKD, a novel framework that enhances distillation from a fixed teacher. The method leverages gradient-based perturbations in the continuous input space to generate a teacher-specific coded dataset, which is optimized to maximize a conditional mutual information (CMI) objective without altering the teacher model.
- **Alternating CMI Maximization Algorithm:** We introduce an efficient an alternating optimization strategy that iteratively updates class centroids and input perturbations. This enable a more accurate CMI proxy and achieves stable and effective maximization without modifying the teacher and achieving more stable and effective maximization.
- **Comprehensive Empirical Evaluation:** We conduct extensive evaluations across diverse datasets and model architectures, demonstrating GCKD's generalizability, consistency, superiority, and compatibility with a wide range of existing knowledge distillation pipelines.

6.2 Related Works

Knowledge Distillation Methods (KD): Knowledge Distillation (KD), also referred to as knowledge transfer, is a widely adopted model compression technique wherein a compact student model is trained to replicate the behavior of a larger, more complex teacher model. The foundational idea was introduced by Bucilu *et al.* [9], who proposed training a smaller model to match the output logits of a larger ensemble. This concept was later refined and popularized by Hinton *et al.* [44], who introduced the use of temperature scaling to soften the output distributions of both teacher and student, thereby facilitating smoother knowledge transfer.

Over time, KD methodologies have diversified significantly. According to the taxonomy proposed by Yang *et al.* [129], existing KD approaches can be broadly categorized into three main groups. The first category, known as response- or logit-based methods, focuses on aligning the output logits or softened probability distributions of the teacher and student models. This includes a wide range of techniques such as those proposed in [7,12,43,44,47,57,64,77,81,106,109,143,145]. The second category, feature-based methods, aims to transfer intermediate representations or feature maps from the teacher to the student. Notable contributions in this area include [14,19,43,52,96,114,117,138,140]. The third category, relation-based methods, emphasizes preserving the structural relationships among data samples in the feature space, as explored in [66, 86, 90, 130].

These categories reflect the evolution of KD from simple output matching to more sophisticated strategies that consider internal representations and inter-sample relationships. Each approach offers unique advantages and trade-offs depending on the architecture, task complexity, and deployment constraints. This growing body of work has laid a strong foundation for further innovations in knowledge distillation, including the development of student-oriented teacher models and input perturbation techniques, which are discussed in the following sections.

KD with Student-Oriented Teacher Design: A growing body of work has shifted focus toward tailoring the teacher model to better suit the learning dynamics of the student. Rather than relying on a

teacher optimized solely for its own performance, these methods aim to enhance the informativeness and compatibility of the teacher's outputs.

For instance, Cho & Hariharan *et al.* [17] and Wang *et al.* [121] demonstrate that early stopping or using intermediate checkpoints during teacher training can preserve higher mutual information between inputs and outputs, resulting in more informative soft labels. Similarly, Tan & Liu [112] and Yang *et al.* [128] propose auxiliary loss functions that encourage the teacher to produce more dispersed output distributions, thereby enriching the knowledge signal.

More recently, Dong *et al.* [26] introduced constraints such as Lipschitz continuity and consistency regularization during teacher training to better approximate the true label distribution. In a related vein, Ye *et al.* [137] proposed training the teacher using a Maximum Conditional Mutual Information (MCMI) objective, which aims to approximate the Bayes conditional probability distribution (BCPD) more accurately than conventional cross-entropy objectives. This approach captures richer contextual dependencies and significantly boosts student performance.

Despite their effectiveness, these student-oriented methods often require modifying the teacher's parameters—a limitation in scenarios where the teacher is a black-box model or has already been deployed in production environments, making retraining infeasible.

Adversarial attack. State-of-the-art deep neural networks are reportedly vulnerable to attacks [35, 111]. Fast Gradient Sign Method (FGSM) [35], Projected Gradient Descent (PGD) [74], and Auto-Attack [21] are a few examples of adversarial attacks that add specially crafted perturbations to clean inputs to produce adversarial examples. The most popular technique for finding perturbations is using gradients to maximize a model's loss on given a clean input while limiting the perturbation size smaller than a specified amount referred to a radius constraint epsilon. In other words, the adversarial attacks find a path to transport clean inputs to cross model decision boundaries, which means to fool the model prediction. In knowledge distillation, exploring the properties of the teacher (e.g., decision boundaries) is the key to improving student performance but also for crafting inputs that steer the teacher's output towards a desired objective. Rather than using these perturbed examples to fool a model, they can be repurposed to generate a richer, more informative supervision signal from the teacher, which is then used for distillation to the student.

KD with Input Perturbation: Another promising direction involves perturbing the input space to enhance the diversity of knowledge transferred from teacher to student. Traditional KD pipelines typically use identical inputs for both models, which may limit the expressiveness of the teacher's knowledge. To address this, adversarial and divergent input perturbations have been explored. Heo *et al.* [42] showed that adversarial examples aligned with the teacher's decision boundary can help the student learn more robust and generalizable representations. However, their method generates only a single adversarial example per input, limiting its coverage of the input space. Nguyen-Duc *et al.* [84] extended this idea by modeling a local adversarial distribution around each input, thereby enabling a more comprehensive exploration of the teacher's decision boundaries. In the context of online co-distillation, , Zhang *et al.* [141] employed Generative Adversarial Networks (GANs) to synthesize divergent examples enriched with "dark knowledge," facilitating mutual learning among multiple student models.

To reduce the computational burden of such methods, Salamah *et al.* [98] proposed a lightweight alternative known as Coded Knowledge Distillation (CKD). CKD introduces an adaptive JPEG compression layer before the teacher model, generating multiple compressed variants of each input using different quality factors (QFs). The most informative variant is selected based on a predefined criterion. While CKD is efficient and effective, it is constrained by its reliance on a discrete set of predefined quantization tables, limiting its ability to fully explore the space of compression-induced perturbations.

Our Proposed Method – GCKD: The proposed method, **GCKD**, builds upon the CKD framework and addresses its key limitations through three major innovations:

- **No Teacher Weight Modification:** GCKD retains the coded teacher paradigm from CKD, eliminating the need to update the teacher's weights. This makes it highly suitable for real-world applications where the teacher model is immutable or provided as a black-box.
- **Continious Space Optmization:** We design a novel distillation dataset for the teacher model within the knowledge distillation framework, which fully exploits the continuous input space using backpropagation-based gradient methods, while keeping the teacher model fixed.
- **Student-Oriented Optimization via MCMI:** GCKD integrates the MCMI estimator from Ye *et al.* [137] but overcomes its computational limitations by introducing a novel alternating optimization algorithm. This algorithm dynamically updates both class-wise clusters and their centroids in the output probability space, providing an analytical solution to the MCMI objective. Unlike the fixed-centroid strategy in the original MCMI, which can accumulate estimation errors, our approach ensures accurate CMI estimation and stable training dynamics. Our proposed alternating algorithm provides an analytical solution to the MCMI objective, enabling efficient centroid updates and accurate CMI estimation.

In summary, GCKD represents a synthesis of input perturbation and student-oriented KD paradigms, offering a practical, efficient, and high-performing solution for real-world knowledge distillation scenarios.

6.3 Notation and preliminaries

This section establishes the notation and foundational concepts used throughout the paper. We begin by defining the core mathematical notation, followed by a review of key information-theoretic measures, including cross-entropy, Kullback-Leibler (KL) divergence, and conditional mutual information (CMI). Finally, we summarize the Coded Knowledge Distillation (CKD) framework, which serves as the basis for our proposed method.

6.3.1 Notation

Let K be a positive integer, and define the index set $[K] \triangleq \{1, \ldots, K\}$. Let the set of class labels be denoted by $[C] \triangleq \{1, \ldots, C\}$. The set of all probability distributions over $[C]$ is the $(C-1)$-dimensional

probability simplex, denoted by $\triangle^{C-1}$. For a vector $P \in \mathbb{R}^C$, we denote its i-th element by $P[i]$.

For any two distributions $P_1, P_2 \in \triangle^{C-1}$, the cross-entropy between them is defined as:

$$H(P_1, P_2) \triangleq \sum_{i=1}^{C} -P_1[i] \ln P_2[i], \tag{6.1}$$

where $\ln$ denotes the natural logarithm. The Kullback–Leibler (KL) divergence, or relative entropy, from P_1 to P_2 is given by:

$$D_{\mathrm{KL}}(P_1 \,\|\, P_2) \triangleq \sum_{i=1}^{C} P_1[i] \ln \frac{P_1[i]}{P_2[i]}. \tag{6.2}$$

It follows that $D_{\mathrm{KL}}(P_1 \,\|\, P_2) = H(P_1, P_2) - H(P_1)$, where $H(P_1) = H(P_1, P_1)$ is the Shannon entropy of P_1.

For any class label $y \in [C]$, let $\delta_y \in \triangle^{C-1}$ be the one-hot distribution where $\delta_y[y] = 1$ and $\delta_y[i] = 0$ for $i \neq y$. The cross-entropy between δ_y and a distribution $P \in \triangle^{C-1}$ simplifies to:

$$H(\delta_y, P) = -\ln P[y]. \tag{6.3}$$

Let (X, Y) be a pair of random variables representing an input and its corresponding label, with a joint probability distribution P_{XY}. We denote specific realizations as (x, y). The marginal distributions are P_X and P_Y, and conditional distributions are $P_{Y|X}$ and $P_{X|Y}$. The expectation with respect to a random variable $Z \sim P_Z$ is written as $\mathbb{E}_Z[\cdot]$ or simply $\mathbb{E}[\cdot]$ when the context is clear.

The mutual information between two random variables X and Y is defined as:

$$I(X; Y) = H(X) - H(X \mid Y),$$

where $H(X)$ is the entropy of X, and $H(X \mid Y)$ is the conditional entropy of X given Y. Intuitively, mutual information quantifies the amount of information that knowing Y provides about X, or vice versa. It measures the reduction in uncertainty about one variable due to knowledge of the other. If X and Y are independent, then $I(X; Y) = 0$, indicating that knowing one provides no information about the other. Building on this, the conditional mutual information (CMI) of X and Y given a third variable Z is defined as:

$$I(X; Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z). \tag{6.4}$$

CMI measures the amount of information shared between X and Y, conditioned on the knowledge of a third variable Z. It captures how much knowing Y helps to predict X, beyond what is already known from Z. This is particularly useful in scenarios where we want to isolate the direct dependency between X and Y while accounting for the influence of Z.

6.3.2 Conditional mutual information of a model

A DNN for classification can be viewed as a function $f_\theta : \mathcal{X} \to \triangle^{C-1}$ parameterized by weights θ, where $\mathcal{X}$ is the input space (e.g., $\mathbb{R}^d$). For an input $x \in \mathcal{X}$, the network outputs a probability distribution $f_\theta(x)$ over the C classes. We will write $f(x)$ for $f_\theta(x)$ when the parameters θ are clear from context.

Let X be a random variable for the input and Y be the corresponding ground-truth label. Let $\hat{Y}$ be the random variable for the label predicted by the model f. The model's output distribution for an input x is $f(x) = P_{\hat{Y}|X}(\cdot|x)$. This assumes the standard Markov chain $Y \to X \to \hat{Y}$, meaning the prediction $\hat{Y}$ is conditionally independent of Y given X based on the formal definition by Yang *et al.* [133]. The CMI between the input X and the prediction $\hat{Y}$, conditioned on a specific ground-truth label $Y = y$, can be expressed as an expected KL divergence. The derivation follows from the definition of CMI:

$$\begin{aligned} I(X;\hat{Y} \mid Y=y) &= \sum_x P_{X|Y}(x \mid y) \left[\sum_{i=1}^{C} P_{\hat{Y}|XY}(\hat{Y}=i \mid x,y) \ln \frac{P_{\hat{Y}|XY}(\hat{Y}=i \mid x,y)}{P_{\hat{Y}|y}(\hat{Y}=i \mid Y=y)} \right] \\ &= \mathbb{E}_{X|Y} \left[\left(\sum_{i=1}^{C} f(X)[i] \ln \frac{f(X)[i]}{P_{\hat{Y}|y}(\hat{Y}=i \mid Y=y)} \right) \mid Y=y \right] \\ &= \mathbb{E}_{X|Y} \left[D_{\mathrm{KL}}(f(X) \| S_y) \mid Y=y \right], \end{aligned} \tag{6.5}$$

where $f(X) = P_{\hat{Y}|X,Y}(\cdot \mid X, Y)$ denotes the model's predictive distribution over the C classes and S_y is the average predicted distribution for class y, formally defined as the centroid of the class-conditional predictions:

$$S_y \triangleq P_{\hat{Y}|Y}(\cdot \mid y) = \mathbb{E}_{X|Y=y}[f(X)].$$

This quantity, S_y, can be interpreted as the centroid of the cluster of output distributions $\{f(x)\}$ for all inputs x belonging to class y. Consequently, $I(X;\hat{Y} \mid Y=y)$ measures the average "distance" from individual predictions within the class-y cluster to their centroid, thus quantifying the concentration of the cluster. Averaging $I(X;\hat{Y} \mid Y=y)$ over all classes y with respect to the class prior distribution $P_Y(y)$ yields the total CMI:

$$I(X;\hat{Y} \mid Y) = \sum_{y=1}^{C} P_Y(y)\, I(X;\hat{Y} \mid Y=y) = \mathbb{E}_{X,Y} \left[D_{\mathrm{KL}}(f(X) \,\|\, S_Y) \right]. \tag{6.6}$$

6.3.3 Empirical Approximation of CMI

In practice, the true data distribution P_{XY} is unknown. Given a training set $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{N}$, we can approximate the CMI of a model f using its empirical estimate. Let $\mathcal{D}_y = \{x_j \mid (x_j, y_j) \in \mathcal{D}, y_j = y\}$ be the subset of inputs with label y, and let $N_y = |\mathcal{D}_y|$. The empirical centroid $\hat{S}_y$ for class y is the sample mean of the predictions for that class:

$$\hat{S}_y = \frac{1}{N_y} \sum_{x_j \in \mathcal{D}_y} f(x_j). \tag{6.7}$$

The empirical CMI, denoted $\hat{I}(X;\hat{Y} \mid Y)$, is then calculated as the weighted average of the KL divergences to these empirical centroids:

$$\begin{aligned}\hat{I}(X;\hat{Y} \mid Y) &= \sum_{y=1}^{C} \frac{N_y}{N} \left(\frac{1}{N_y} \sum_{x_j \in \mathcal{D}_y} D_{\mathrm{KL}}(f(x_j) \,\|\, \hat{S}_y) \right) \\ &= \frac{1}{N} \sum_{y=1}^{C} \sum_{x_j \in \mathcal{D}_y} D_{\mathrm{KL}}(f(x_j) \,\|\, \hat{S}_y). \end{aligned} \tag{6.8}$$

Hereafter, for notational simplicity within our algorithmic framework, we will use S_y and $I(\cdot)$ to refer to these empirical estimates, as all subsequent computations are performed on the finite dataset $\mathcal{D}$.

6.3.4 Coded Knowledge Distillation

The Coded Knowledge Distillation (CKD) framework, proposed by Salamah *et al.* [98], improves student model generalization by addressing the common issue of teacher overconfidence. The core principle of CKD is to perturb input images through an adaptive encoding layer before they are processed by the teacher model, a technique that can be applied orthogonally to any existing distillation method. This perturbation is specifically designed to mitigate the "peakiness" of the teacher's predictions—that is, to transform overconfident, near one-hot probability vectors into softer, more informative distributions from which the student can learn. Ideally, this process can be formulated as an optimization problem that seeks to find a perturbed input X' that increases the entropy of the teacher's output distribution while maintaining correct classification, as shown in equation 6.9:

$$\begin{aligned} \min_{X'=F(X,Y)} \quad & \mathbb{E}_{(X',Y)} \Big[-\log f(X')[Y] - \lambda H(f(X')) \Big] \\ \text{subject to} \quad & d(X, X') \leq \varepsilon, \end{aligned} \tag{6.9}$$

where $X' = F(X,Y)$ is the perturbed input generated by the encoding function F, λ is a hyperparameter balancing the classification and entropy terms, and the perturbation is constrained by $d(X, X') \leq \varepsilon$ with perturbation budget ε. However, as this formulation assumes a continuous encoding function, the original authors implemented a practical, discrete alternative using adaptive JPEG compression. Their method selects the compression quality factor that maximizes the KL divergence between the teacher's outputs for the original and compressed images, provided that the teacher still classifies the compressed image correctly. Consequently, CKD leverages controlled input perturbations to generate richer supervisory signals from the teacher, thereby improving the student's performance and generalization capabilities.

6.4 Generalized Coded Knowledge Distillation

This section details the theoretical foundation and algorithmic implementation of our proposed framework, Generalized Coded Knowledge Distillation (GCKD). We begin by formulating the core optimization problem that GCKD is designed to solve. Subsequently, we introduce a novel alternating optimization algorithm engineered to overcome the computational challenges inherent in this formulation to design perturbed samples that maximize conditional mutual information. The section concludes by presenting the complete GCKD framework, illustrating how these components integrate to improve distillation performance.

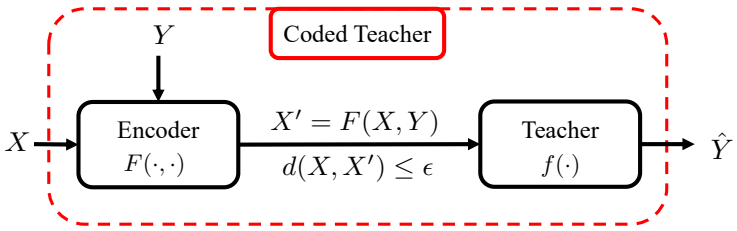


Figure 6.1: A diagram illustrating the GCKD encoding Framework. The encoder takes inputs X and Y, and generates a perturbed instance $X' = F(X,Y)$, constrained by $d(X,X') \leq \varepsilon$. The encoded input X' is then passed to a teacher model f_{θ_t} to produce the final output $\hat{Y}'$.

6.4.1 Overall Framework

The Generalized Coded Knowledge Distillation (GCKD) framework consists of two stages: (i) the teacher encoding stage, and (ii) the distillation stage.

In the teacher encoding stage, given a fixed teacher model f, we prepend a generalized encoder F to it that intakes the input X and its corresponding ground truth label Y and produce a perturbed input $X' = F(X,Y)$ which is subsequently fed into f. To prevent significant semantic shift of the input, we impose a constraint on the encoder. Given a distortion measure $d(\cdot,\cdot)$, the distortion between the raw input X and the perturbed input X' should be bounded, i.e., $d(X,X') = d(X,F(X,Y)) \leq \epsilon$, where ϵ is a predefined distortion budget. We then consider the encoder F and the teacher f as a whole and term it the *coded teacher* following CKD [98]. A diagram is presented in Fig. 6.1 to illustrate the structure of the coded teacher. To boost the distillation performance of the coded teacher without altering the teacher model f, we optimize the encoder F, equivalently $\hat{X}$, to maximize the CMI of the coded teacher. Details about the objective function and the optimization algorithm will be presented in the following sections. Given a training set $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^N$ to be used in distillation, the optimal solution of F is

characterized by the *coded dataset* $\mathcal{D}' = \{(x'_j, y_j)\}_{j=1}^{N}$, which contains perturbed samples generated from our CMI maximization algorithm.

In the distillation stage, we use the coded teacher instead of the original teacher for distillation. That is, given a training sample x during distillation, we use its perturbed version x' from the coded dataset as the input to the teacher f, and let the student learn from teacher's knowledge represented by its intermediate features or the final output using any distillation method. Note that the input to the student is always the raw input x. Given its design, GCKD is completely plug-and-play in the distillation stage, being compatible with all kinds of distillation method.

6.4.2 Problem Formation

The GCKD framework, illustrated in Figure 6.1, aims to generate more informative teacher signals by crafting input perturbations. We formulate this as a constrained optimization problem that seeks a deterministic encoding function $F(X, Y)$ to produce perturbed inputs X'. The objective balances two goals: maintaining classification accuracy on perturbed inputs, as measured by cross-entropy (CE), while simultaneously maximizing the CMI between the perturbed inputs and their corresponding predictions. The CMI term encourages the teacher's outputs to be structured in a way that is beneficial for distillation.

The optimization problem is:

$$\begin{aligned} \min_{F} \quad & \mathbb{E}_{(X,Y)}\left[H(\delta_Y, f(X'))\right] - \lambda I(X'; \hat{Y}' \mid Y) \\ \text{subject to} \quad & d(X, F(X, Y)) \leq \varepsilon \quad \forall (X, Y), \end{aligned} \tag{6.10}$$

where $X' = F(X, Y)$, $\hat{Y}'$ is the model's prediction for X', $\lambda > 0$ is a trade-off hyperparameter, and $d(\cdot, \cdot) \leq \varepsilon$ is a perturbation constraint (e.g., an ℓ_p-norm ball). The term $I(X'; \hat{Y}' \mid Y)$ is the perturbed CMI in our context, which we formally define next.

Directly optimizing Equation 6.10 is computationally intractable. The perturbed CMI term, $I(X'; \hat{Y}' \mid Y) = \mathbb{E}[D_{\mathrm{KL}}(f(X') \,\|\, S'_Y)]$, depends on the perturbed centroids $S'_Y = \mathbb{E}_{X|Y}[f(F(X, Y))]$. Each centroid S'_y is an expectation over all perturbed inputs X' of class y. This creates a global dependency: the optimal perturbation for a single input x depends on the perturbations of all other inputs in its class. This complex interdependency makes standard gradient-based optimization with respect to F infeasible. Prior work by Ye *et al.* [137] circumvents a similar issue by fixing the centroids S_y using values pre-calculated from a fixed teacher model. However, this approach is suboptimal because the very process of finding optimal perturbations X' should, in turn, cause the centroids themselves to shift. Using fixed centroids provides an inconsistent target and does not fully resolve the underlying optimization challenge.

To overcome this challenge, we propose an alternating optimization algorithm that decouples the problem, as detailed in Section 6.4.4.

6.4.3 Extension of CMI to the Adversarial Context

Our algorithm operates by replacing benign inputs X with their perturbed counterparts $X' = F(X, Y)$, where the encoding function F is deterministic. This modifies the model's output geometry. We adapt the definitions of CMI and class centroids to this setting, using the perturbed random variables X' and the corresponding model prediction $\hat{Y}'$.

The CMI for a specific class y and the total CMI are defined as:

$$I(X'; \hat{Y}'|Y = y) = \mathbb{E}_{X'|Y=y}\left[D_{\mathrm{KL}}(f(X') \,\|\, S'_y)\right] = \mathbb{E}_{X|Y=y}\left[D_{\mathrm{KL}}(f(F(X,Y)) \,\|\, S'_y)\middle|Y = y\right], \quad (6.11)$$

$$I(X'; \hat{Y}'|Y) = \mathbb{E}_{X',Y}\left[D_{\mathrm{KL}}(f(X') \,\|\, S'_Y)\right] = \mathbb{E}_{X,Y}\left[D_{\mathrm{KL}}(f(F(X,Y)) \,\|\, S'_Y)\right], \quad (6.12)$$

where the perturbed centroid for class y is the expected model prediction over all perturbed inputs of that class:

$$S'_y \triangleq \mathbb{E}_{X|Y=y}[f(F(X,Y))]. \quad (6.13)$$

A key theoretical question is whether the identity $I(X; \hat{Y}'|Y) = I(X'; \hat{Y}'|Y)$ holds in our framework, which formed through Equations 6.5 to 6.8. The standard formulation by Yang *et al.* [133] assumes the Markov chain $Y \to X \to \hat{Y}$. In contrast, our framework implies the chain $(X, Y) \to X' \to \hat{Y}'$, where the perturbation X' may depend on both X and Y as previously illustrated in Figure 6.1. The following lemma confirms that the identity still holds due to the deterministic nature of the encoding function F.

Lemma 1 *Given the Markov chain $(X, Y) \to X' \to \hat{Y}'$, where $X' = F(X, Y)$ is a deterministic function and the model f defines $P_{\hat{Y}'|X'}$, the following informational equivalence holds:*

$$I(X; \hat{Y}' \mid Y) = I(X'; \hat{Y}' \mid Y). \quad (6.14)$$

Proof: By definition, CMI is $I(A; B \mid C) = H(B \mid C) - H(B \mid A, C)$. Thus, we need to show:

$$H(\hat{Y}' \mid Y) - H(\hat{Y}' \mid X, Y) = H(\hat{Y}' \mid Y) - H(\hat{Y}' \mid X', Y).$$

This is equivalent to proving $H(\hat{Y}' \mid X, Y) = H(\hat{Y}' \mid X', Y)$. From the Markov chain $(X, Y) \to X' \to \hat{Y}'$, the prediction $\hat{Y}'$ is conditionally independent of (X, Y) given X'. Formally, $P(\hat{y}' \mid x, y, x') = P(\hat{y}' \mid x')$. The conditional entropy is $H(\hat{Y}' \mid X, Y, X') = \mathbb{E}_{X,Y,X'}[-\log P(\hat{Y}' \mid X, Y, X')] = \mathbb{E}_{X'}[-\log P(\hat{Y}' \mid X')] = H(\hat{Y}' \mid X')$. Since $X' = F(X, Y)$ is a deterministic function, conditioning on (X, Y) is equivalent to conditioning on (X, Y, X'). Therefore,

$$H(\hat{Y}' \mid X, Y) = H(\hat{Y}' \mid X, Y, X') = H(\hat{Y}' \mid X').$$

Similarly, conditioning on (X', Y) means we know X'. From the Markov property, knowledge of Y provides no additional information about $\hat{Y}'$. Thus,

$$H(\hat{Y}' \mid X', Y) = \mathbb{E}_{X',Y}[-\log P(\hat{Y}' \mid X', Y)] = \mathbb{E}_{X'}[-\log P(\hat{Y}' \mid X')] = H(\hat{Y}' \mid X').$$

Since both conditional entropies equal $H(\hat{Y}' \mid X')$, they are equal to each other, which completes the proof. □

Lemma 1 provides the theoretical justification for using $I(X'; \hat{Y}' \mid Y)$ in our objective function. It confirms that the information about the prediction is the same whether we consider the original input X or the perturbed input X', because X' is just a deterministic transformation of (X, Y). This allows us to focus our optimization on the more convenient term involving X'.

Alternative Proof of Lemma 1

This subsection provides an alternative proof for the informational equivalence stated in Lemma 1, namely $I(X; \hat{Y}'|Y) = I(X'; \hat{Y}'|Y)$. The proof proceeds by applying the law of total probability to demonstrate that the relevant conditional probability distributions are identical.

Under the GCKD Markov chain $(X, Y) \to X' \to \hat{Y}'$, the perturbed sample X' is generated by a deterministic function $X' = F(X, Y)$. This implies that the conditional probability density function $P_{X'|XY}(x' \mid x, y)$ is a Dirac delta function centered at $F(x, y)$:

$$P(x' \mid x, y) = \delta(x' - F(x, y)) = \begin{cases} 1 & \text{if } x' = F(x, y), \\ 0 & \text{otherwise.} \end{cases}$$

Using the law of total probability, we can express the conditional distribution of $\hat{Y}'$ given (X, Y) by marginalizing over all possible values of X':

$$\begin{aligned} P_{\hat{Y}'|XY}(\hat{y}' \mid x, y) &= \int P_{\hat{Y}'|X'XY}(\hat{y}' \mid x', x, y)\, P_{X'|XY}(x' \mid x, y)\, dx' \\ &= \int P_{\hat{Y}'|X'}(\hat{y}' \mid x')\, P_{X'|XY}(x' \mid x, y)\, dx' && \text{(by Markov property)} \\ &= P_{\hat{Y}'|X'}(\hat{y}' \mid F(x, y)) && \text{(by deterministic property of } X') \\ &= f(F(x, y))[\hat{y}'] = f(x')[\hat{y}']. \end{aligned}$$

Since $x' = F(x, y)$ and the prediction $\hat{y}'$ is conditionally independent of Y given X', we arrive at the following equality of conditional distributions:

$$P_{\hat{Y}'|XY}(\cdot \mid x, y) = P_{\hat{Y}'|X'}(\cdot \mid x') = P_{\hat{Y}'|X'Y}(\cdot \mid x', y). \tag{6.15}$$

The equivalence of these distributions directly implies an equivalence in their corresponding conditional entropies:

$$H(\hat{Y}' \mid X, Y) = H(\hat{Y}' \mid X', Y).$$

As established in the proof in the main text, this equality is sufficient to prove the lemma.

6.4.4 Alternating Optimization for CMI Maximization

Having established the theoretical validity of the perturbed CMI term, we now introduce the algorithm designed to solve the GCKD objective. The core challenge is the interdependence between the perturbed samples (X') and the perturbed centroids (S'_Y).

To address this, we reformulate the objective by introducing an auxiliary conditional distribution, $Q(\cdot \mid i, y)$. This technique allows us to convert the problem into a tractable double minimization, as formalized in the following theorem. The encoding function $F(X, Y)$ is instantiated using gradient-based adversarial attacks with a fixed number of steps (e.g., PGD), ensuring it remains a deterministic function as required by Lemma 1.

Theorem 2 *For any $i, y \in \{1, \ldots, C\}$, let $Q'(\cdot \mid i, y)$ denote a dummy conditional distribution over the input space X, which is assumed to be discrete for simplicity. Then the minimization of the objective function equation 6.10 can be computed by solving the following double minimization problems with $\lambda > 0$:*

$$
\begin{aligned}
I(\varepsilon) &= \min_{F:\ \|X - X'\|_p \le \varepsilon} \Big(\mathbb{E}_{(X,Y)} \big[H(\delta_Y, f_\theta(X')) \big] - \lambda\, I(X'; \hat{Y}' \mid Y) \Big) \\
&= \min_{F:\ \|X - X'\|_p \le \varepsilon} \ \min_{Q'(x|i,y)} \Big(\mathbb{E}_{(X,Y)} \big[H(\delta_Y, f_\theta(X')) \big] - \lambda \sum_{x,y} P(x,y) \sum_{i=1}^{C} f_\theta(x')[i] \ln Q'(x \mid i, y) \Big),
\end{aligned}
\tag{6.16}
$$

where the inner minimization above is achieved when

$$
Q'(x \mid i, y) = \frac{P_{X|Y}(x \mid y)\, f_\theta(x')[i]}{P_{\hat{Y}'|Y}(i \mid y)}. \tag{6.17}
$$

Proof of Theorem 2

For simplicity, we drop the subscript θ from $f(.)$, and assume that the space of X is discrete. To prove Theorem 2, we first introduce an auxiliary distribution $Q'(\cdot \mid i, y)$ and derive a new expression for

$I(X;\hat{Y}' \mid Y)$ as follows:

$$
\begin{aligned}
I(X;\hat{Y}' \mid Y) &= \sum_{y\in[C]} P_Y(y) I(X';\hat{Y}' \mid Y=y) \\
&= \sum_y \sum_x P(x,y) \sum_{i=1}^{C} P_{\hat{Y}|X'Y}(\hat{Y}=i \mid x',y) \ln \frac{P_{\hat{Y}|X'Y}(\hat{Y}=i \mid x',y)}{P_{\hat{Y}'|Y}(\hat{Y}'=i \mid Y=y)} \\
&= \sum_y \sum_x P(x,y) \sum_{i=1}^{C} f(F(x,y))[i] \ln \frac{f(F(x,y))[i]}{P_{\hat{Y}'|Y}(\hat{Y}'=i \mid Y=y)} \qquad (6.18) \\
&= \sum_y P_Y(y) \sum_x P_{X|Y}(x \mid y) \sum_{i=1}^{C} f(F(x,y))[i] \ln \frac{P_{X|Y}(x \mid y) f(F(x,y))[i]}{P_{X|Y}(x \mid y) P_{\hat{Y}'|Y}(i \mid y)} \\
&= \sum_y P_Y(y) \sum_{i=1}^{C} P_{\hat{Y}'|Y}(i \mid y) \sum_x \frac{P_{X|Y}(x \mid y) f(F(x,y))[i]}{P_{\hat{Y}'|Y}(i \mid y)} \ln \frac{P_{X|Y}(x \mid y) f(F(x,y))[i]}{P_{X|Y}(x \mid y) P_{\hat{Y}'|Y}(i \mid y)} \qquad (6.19) \\
&= \max_{\{Q'(x|i,y)\}} \sum_y P_Y(y) \sum_{i=1}^{C} P_{\hat{Y}'|Y}(i \mid y) \sum_x \frac{P_{X|Y}(x \mid y) f(F(x,y))[i]}{P_{\hat{Y}'|Y}(i \mid y)} \ln \frac{Q'(x \mid i,y)}{P_{X|Y}(x \mid y)} \\
&= \max_{\{Q'(x|i,y)\}} \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(F(x,y))[i] \ln \frac{Q'(x \mid i,y)}{P_{X|Y}(x \mid y)}, \qquad (6.20)
\end{aligned}
$$

where equation 6.18 is due to equation 6.15 , Equation 6.19 follows from the cross entropy inequality, and the maximization in Equation 6.20 is achieved when

$$
Q'^{*}(x \mid i,y) = \frac{P_{X|Y}(x \mid y) f(F(x,y))[i]}{P_{\hat{Y}'|Y}(i \mid y)}. \qquad (6.21)
$$

Thus, the original objective function 6.10 can be converted into a double minimization problem over

F and $\{Q'(x \mid i, y)\}$ as follows:

$$
\begin{aligned}
I(\epsilon) &= \min_{F:\|X-X'\|_p\leq\epsilon} \Big\{\mathbb{E}_{(X,Y)}[H(\delta_Y, f(X'))] - \lambda\, I(X;\hat{Y} \mid Y)\Big\} \\
&= \min_{F:\|X-X'\|_p\leq\epsilon} \left\{\mathbb{E}_{(X,Y)}[H(\delta_Y, f(X'))] - \lambda \max_{\{Q'(x|i,y)\}} \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(F(x,y))[i] \ln \frac{Q'(x \mid i,y)}{P_{X|Y}(x \mid y)}\right\} \\
&\equiv \min_{F:\|X-X'\|_p\leq\epsilon} \left\{\mathbb{E}_{(X,Y)}[H(\delta_Y, f(X'))] - \lambda \max_{\{Q'(x|i,y)\}} \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(F(x,y))[i] \ln Q'(x \mid i,y)\right\} \qquad (6.22) \\
&= \min_{F:\|X-X'\|_p\leq\epsilon} \left\{\mathbb{E}_{(X,Y)}[H(\delta_Y, f(X'))] + \lambda \min_{\{Q'(x|i,y)\}} -\sum_{x,y} P(x,y) \sum_{i=1}^{C} f(F(x,y))[i] \ln Q'(x \mid i,y)\right\} \\
&= \min_{F:\|X-X'\|_p\leq\epsilon} \min_{\{Q'(x|i,y)\}} \left\{\mathbb{E}_{(X,Y)}[H(\delta_Y, f(F(X,Y)))] - \lambda \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(F(x,y))[i] \ln Q'(x \mid i,y)\right\},
\end{aligned}
$$

□

where the symbol "≡" in Equation 6.22 indicates equivalence up to an additive constant independent of $Q'(x \mid i, y)$, which is omitted here for clarity.

This reformulation of the problem as $\min_F \min_Q$ is powerful because it lends itself directly to an alternating optimization scheme that solves the equivalent minimax problem $\min_F \max_{Q'}$. In practice, we work with an empirical estimate of the objective over a training set $\mathcal{D}$. The empirical loss can thus be written as:

$$
\begin{aligned}
\mathcal{L}_{\text{emp}} &= \mathcal{L}\big(\lambda,\, x',\, \{Q'(x \mid i, y)\}_{(x,y)\in\mathcal{D},\, i\in\{1,\ldots,C\}}\big) \\
&= \underbrace{\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \big(-\ln f_\theta(x')[y]\big)}_{\mathcal{L}_{\text{CE}}} - \lambda \underbrace{\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \left[\sum_{i=1}^{C} f_\theta(x')[i] \ln Q'(x \mid i, y)\right]}_{\mathcal{L}_{\text{GCKD}}},
\end{aligned} \qquad (6.23)
$$

where $x' = F(x, y)$. The structure of this objective leads to a practical algorithm where we minimize $\mathcal{L}_{\text{emp}}$ by iteratively alternating between updating the perturbed samples x' (minimization over F) and updating the auxiliary distributions Q' (maximization over Q), as detailed in the following section.

6.4.5 Alternating Optimization Procedure

The double minimization problem in Theorem 2 lends itself to an alternating optimization algorithm that implements a minimax strategy. This approach iteratively solves the GCKD objective by breaking it into

two sub-problems: generating optimal perturbations x' for a fixed auxiliary distribution Q', and then updating Q' based on the new perturbations.

Initialization: Before the main optimization loop, initial values for the perturbed centroids $\{S'_y\}$ and the auxiliary distributions $\{Q'\}$ are computed. The centroids are initialized using the teacher's predictions on the original, unperturbed dataset $\mathcal{D}$ as per Eq. equation 6.7. Subsequently, the initial Q' distributions are calculated based on these centroids and the original predictions.

The algorithm then proceeds by alternating between two steps for a fixed number of iterations:

Step 1: Perturbed Sample Generation (Fixed Q'). In this step, we hold the auxiliary distributions $\{Q'\}$ from the previous iteration constant. For each sample $(x_j, y_j) \in \mathcal{D}$, we find the optimal perturbed sample x'_j by minimizing the per-sample loss derived from Eq. equation 6.23:

$$\begin{aligned} \min_{x'_j} \quad & H(\delta_{y_j}, f(x'_j)) - \lambda \sum_{i=1}^{C} f(x'_j)[i] \ln Q'(x_j \mid i, y_j) \\ \text{subject to} \quad & d(x_j, x'_j) \leq \varepsilon. \end{aligned} \tag{6.24}$$

This objective is minimized using a gradient-based procedure (e.g., PGD), producing a new set of perturbed samples $\{x'_j\}_{j=1}^{N}$.

Step 2: Auxiliary Distribution Update (Fixed Perturbations). With the set of perturbed samples $\{x'_j\}$ now fixed, we update the auxiliary distributions $\{Q'\}$ for the next iteration. This involves two substeps:

1. **Centroid Update:** First, the perturbed centroids $\{S'_y\}$ are recalculated as the empirical mean of the teacher's predictions over the newly generated perturbed samples:

$$S'_y \leftarrow \frac{1}{N_y} \sum_{(x_j, y_j) \in \mathcal{D},\, y_j = y} f(x'_j). \tag{6.25}$$

2. Q' **Calculation:** Using the updated centroids and the new predictions $\{f(x'_j)\}$, the auxiliary distributions $\{Q'\}$ are recalculated according to Eq. equation 6.17 for each sample.

$$\begin{aligned} Q^{*'}(x \mid i, y) &\leftarrow \frac{P_{X|Y}(x \mid y)\, f(x')[i]}{P_{\hat{Y}'|Y}(i \mid y)} \\ &= \frac{N_{x,y} \cdot f(x')[i]}{N_y \cdot P_{\hat{Y}'|Y}(i \mid y)}, i \in [C], \end{aligned} \tag{6.26}$$

where $\mathcal{D}'_y = \{(x'_j, y_j) \in \mathcal{D} \mid x'_j = x', y_j = y\}$ and $N_{x,y} = |D_{x,y}|$. For typical datasets, $N_{x,y} = 1$.

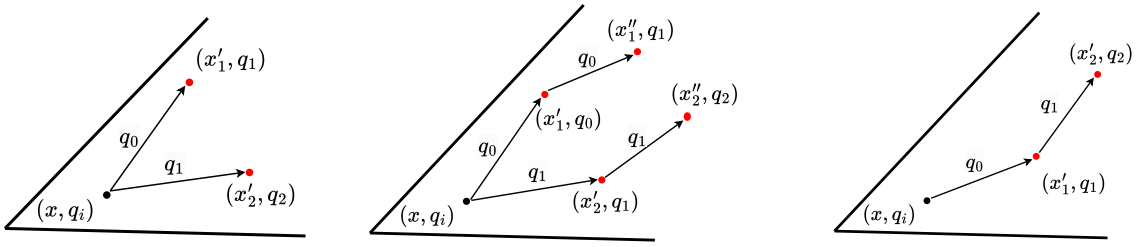
Step 3. Repeat Step 1 and Step 2 until convergence. Return the coded dataset D' at the last iteration.

The alternating optimization procedure constitutes the first stage of the overall GCKD training framework. As outlined in Algorithm 3, this stage is executed once per epoch to generate a fixed set of "coded" inputs X' that will serve as the "Coded Dataset" for the subsequent training iterations. In the second phastagese, this pre-computed dataset is used to train the student model. This two-stages approach ensures that the computationally intensive process of generating optimal perturbations is performed only once per epoch, after which standard distillation training can proceed.

Algorithm 3 Alternating Optimization for GCKD

Input: Fixed teacher f_{θ_t}, student f_{θ_s}, dataset $\mathcal{D}$, epochs T, alternating iterations K_{alt}, PGD steps K_{pgd}.
Input: Perturbation budget ε, PGD step size α, CMI weight λ, distillation weight β.

for epoch $e = 1, \ldots, T$ **do**
  // — Stage 1: Find Optimal Perturbations X' for the Epoch —
  Let X be the set of all original images in $\mathcal{D}$.
  Let X' be the set of perturbed images, initialized as $X' \leftarrow X$.
  Initialize centroids and Q':
  for each class y **do**
    $S'_y \leftarrow \frac{1}{|\mathcal{D}_y|} \sum_{x_j \in \mathcal{D}_y} f_{\theta_t}(x_j)$.
  end for
  for each sample (x_j, y_j) and class index i **do**
    $\ln Q'(x_j \mid i, y_j) \leftarrow \ln(1/|\mathcal{D}_{y_j}|) + \ln f_{\theta_t}(x_j)[i] - \ln S'_{y_j}[i]$.
  end for
  for iter $k = 1, \ldots, K_{\text{alt}}$ **do** ▷ Alternating optimization loop
    // Step (a): Generate perturbed X' using fixed Q' from previous iter
    for each sample $(x_j, y_j) \in \mathcal{D}$ **do**
      Define attack loss: $\mathcal{L}_{\text{attack}}(x'_j) = H(\delta_{y_j}, f_{\theta_t}(x'_j)) - \lambda \sum_{i=1}^{C} f_{\theta_t}(x'_j)[i] \ln Q'(x_j \mid i, y_j)$.
      Apply PGD for K_{pgd} steps to update x'_j, projecting onto the ℓ_∞-ball around x_j.
    end for
    // Step (b): Update centroids S' and Q' using new perturbed X'
    for each class $y \in [C]$ **do**
      $S'_y \leftarrow \frac{1}{|\mathcal{D}_y|} \sum_{x'_j \text{ from class } y} f_{\theta_t}(x'_j)$.
    end for
    for each sample (x_j, y_j) and class index i **do**
      $\ln Q'(x_j \mid i, y_j) \leftarrow \ln(1/|\mathcal{D}_{y_j}|) + \ln f_{\theta_t}(x'_j)[i] - \ln S'_{y_j}[i]$.
    end for
  end for
  At this point, X' contains the final set of perturbed images for the epoch.
  // — Stage 2: Train Student Model using fixed X and X' —
  for each mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
    Let $\mathcal{B}_x = \{x_j \mid (x_j, y_j) \in \mathcal{B}\}$.
    Let $\mathcal{B}_{x'} = \{x'_j \mid (x_j, y_j) \in \mathcal{B}\}$.
    Compute student loss: $\mathcal{L}_{\text{student}} \leftarrow \mathcal{L}_{\text{CE}}(z_s, y) + \beta \mathcal{L}_{\text{distill}}(f_{\theta_s}(\mathcal{B}_x), f_{\theta_t}(\mathcal{B}_{x'}))$.
    Update θ_s using $\nabla_{\theta_s} \mathcal{L}_{\text{student}}$.
  end for
end for
return Trained student model f_{θ_s}.



(a) FGSM (b) PGD (c) Amortized PGD

Figure 6.2: A conceptual illustration of different strategies for generating perturbed inputs. In this diagram, q_i is used as a shorthand for the auxiliary 'dummy' distribution, $Q'(x|i, y)$, which is central to the double minimization problem formulated in Theorem 2. (a) **FGSM** is a single-step method that provides a coarse approximation of the optimal perturbation. (b) **PGD** iteratively refines the perturbation x' to better solve the inner minimization problem while the set of dummy distributions $\{q_i\}$ is held fixed. (c) **Amortized PGD** represents a more efficient variant where the dummy distribution q_i is updated after each perturbation of x_i', a process that is faster but potentially less precise than completing the full PGD optimization for a fixed $\{q_i\}$.

The core of Stage 1 is the generation of a "coded dataset" through input perturbations. The choice of the perturbation generation strategy is critical, as different methods offer a trade-off between computational cost and the thoroughness of the optimization. We considered three prominent adversarial attack strategies for this task: the FGSM, PGD, and an Amortized PGD variant. These alternatives are conceptually illustrated in Figure 6.2.

The procedural differences between these methods are as follows:

- **FGSM** is a computationally inexpensive, single-step method that generates a perturbation in the direction of the loss function's gradient.
- **PGD** is an iterative, multi-step method that can be seen as a stronger version of FGSM. It repeatedly applies the gradient-based step and projects the result back into the allowed perturbation space, allowing for a more thorough search for an effective perturbation.
- **Amortized PGD** is a variant we considered that updates the auxiliary distribution after each perturbation step, aiming to reduce computational overhead compared to the full PGD procedure.

To empirically determine the optimal strategy for our framework, we conducted an ablation study comparing the final student performance when trained on coded datasets generated by each of these three methods. The results of this comparison are detailed in the Experimental Results section (Section 6.5).

6.5 Experiments

This section presents the empirical evaluation of the Generalized Coded Knowledge Distillation (GCKD) framework. We first detail the experimental setup for the CIFAR-100 and ImageNet datasets. We then present an ablation study on the choice of perturbation generation strategy, followed by the main results comparing the GCKD teacher against a conventional teacher across various state-of-the-art distillation methods.

6.5.1 Experimental Setup

All experiments follow the two-stages procedure outlined in Algorithm 3. In Stage 1, a "coded dataset" of perturbed inputs is generated. In Stage 2, this dataset is used to train a student model via knowledge distillation. The baseline for comparison is a conventional teacher trained with the standard Negative Log-Likelihood (NLL) loss.

CIFAR-100. Pretrained teacher models were obtained from the official CRD repository [114]. The GCKD coded dataset was generated once per epoch. The alternating optimization (Stage 1) used a 2-step PGD attack ($K_{\text{pgd}} = 2$) with a perturbation budget of $\varepsilon = 0.12/255$ and a step size of $\alpha = 0.06/255$. The CMI weight was set to $\lambda = 0.5$ for the WRN-40-2 teacher and $\lambda = 0.3$ for the VGG-13 teacher. Student training (Stage 2) followed the protocol in [114], training for 240 epochs with a batch size of 64. The learning rate was initialized to 0.05 and decayed by a factor of 0.1 at epochs 150, 180, and 210. For MobileNetV2 and ShuffleNetV1 students, the initial learning rate was 0.01. Results are reported as the average accuracy over three independent runs.

ImageNet. Pretrained teacher models were sourced from the official PyTorch repository. To manage computational load, the GCKD coded dataset was generated offline once, without data augmentation, prior to student training. The alternating optimization procedure was run for $K_{\text{alt}} = 12$ iterations. Perturbations were generated using a 7-step PGD attack ($K_{\text{pgd}} = 7$) with $\varepsilon = 0.1/255$ and $\alpha = 0.15/255$. The CMI weight was set to $\lambda = 0.5$ for the ResNet-34→ResNet-18 pair and $\lambda = 0.44$ for the ResNet-50→MobileNetV2 pair. Student training followed the setup in [143], training for 100 epochs with a batch size of 256 and an initial learning rate of 0.1.

6.5.2 Ablation on Perturbation Generation Strategy

To determine the optimal strategy for generating perturbed inputs in Stage 1, we compared the performance of students trained with coded datasets generated by three different attack methods: single-step FGSM, Amortized PGD, and multi-step PGD. Table 6.1 shows the resulting student test accuracies on CIFAR-100 for VGG-13 teacher models paired with VGG-8 and MobileNetV2 students, using both KD and DIST distillation methods. Multi-step PGD yielded the highest student accuracy on CIFAR-100. However, we selected the more computationally efficient Amortized PGD for subsequent experiments, as it offers a strong trade-off between performance and computational cost.

Table 6.1: Comparison of student test accuracy (%) using different adversarial attack strategies to generate the coded dataset for the GCKD teacher. Results are shown for VGG-13 as the teacher and two different students (VGG-8, MobileNetV2). 'Conventional' refers to a standard NLL teacher.

Teacher → Student	**Method**	**Conventional**	**FGSM**	**Amortized PGD**	**PGD**
VGG-13 → VGG-8	KD	72.98	73.61	73.86	**74.12**
	DIST	73.89	74.14	74.56	**74.71**
VGG-13 → MobileNetV2	KD	67.37	68.34	68.78	**68.91**
	DIST	68.50	68.93	69.48	**69.61**

6.5.3 Results on CIFAR-100

Table 6.2 presents the main results on the CIFAR-100 dataset. The table compares the test accuracies of student models trained with a conventional NLL teacher versus those trained with our GCKD teacher. The evaluation spans multiple teacher-student architecture pairs and a comprehensive set of distillation methods, including logit-based (KD, DKD, DIST, WTTM), relation-based (CC), and feature-based (AT, FitNet, FT, SP, ITRD) approaches. In all reported experiments, the student model trained using the GCKD teacher achieved a higher test accuracy than the student trained with the conventional teacher.

6.5.4 Results on ImageNet

Table 6.3 reports the performance of GCKD on the large-scale ImageNet dataset for two teacher-student pairs: ResNet-34→ResNet-18 and ResNet-50→MobileNetV2. We compare the GCKD teacher against the conventional teacher using five representative distillation methods. The table shows the Top-1 accuracy for the student models. For both teacher-student pairs and all tested distillation methods, the student trained with the GCKD teacher obtained a higher Top-1 accuracy than the student trained with the conventional teacher.

6.5.5 Comparison with MCMI

Our GCKD framework builds upon the principle of maximizing CMI, similar to MCMI. However, MCMI requires modifying the teacher model and relies on fixed, pre-calculated class centroids during optimization. As argued in Section 6.4, this leads to an inconsistent optimization target, since the very act of perturbing inputs should inherently shift these centroids. GCKD resolves this "fixed-centroid problem" directly. By employing a novel alternating optimization algorithm, GCKD dynamically updates the class

Table 6.2: The test accuracy (%) of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the same/different architectures. The subscript denotes the improvement achieved by replacing the standard teacher with the GCKD teacher. We use bold numbers and asterisk (*) to denote the best results. GCKD delivers consistent accuracy gains across a wide variety of architectures and KD methods.

GCKD distillation performance on CIFAR100 for **different** and **same** architectures.										
Teacher	WRN-40-2		VGG-13		WRN-40-2		WRN-40-2		VGG-13	
	CE	GCKD	CE	GCKD	CE	GCKD	CE	GCKD	CE	GCKD
Training Acc	99.87	99.84	99.96	99.85	99.87	99.84	99.87	99.84	99.96	99.85
NLL	0.0158	0.0407	0.0067	0.0122	0.0158	0.0407	0.0158	0.0407	0.0067	0.0122
CMI	0.0398	0.1328	0.0175	0.0414	0.0398	0.1328	0.0398	0.1328	0.0175	0.0414
		3.15×		2.37×		3.15×		3.15×		2.37×
Student	ShuffleNetV1		MobileNetV2		WRN-16-2		WRN-40-1		VGG-8	
Accuracy	70.50		64.60		73.26		71.98		70.36	
KD	74.83	76.30 +1.47	67.37	68.78 +1.41	74.92	75.61 +0.69	73.54	73.86 +0.32	72.98	73.85 +0.87
FitNet	73.73	74.41 +0.68	64.14	66.02 +1.88	73.58	74.41 +0.83	72.24	72.95 +0.71	71.02	71.92 +0.90
AT	73.32	75.28 +1.96	59.40	61.17 +1.77	74.08	74.87 +0.79	72.77	73.40 +0.63	71.43	72.13 +0.70
SP	74.52	76.51 +1.99	66.30	67.82 +1.52	73.83	74.36 +0.53	72.43	73.10 +0.67	72.68	73.75 +1.07
CC	71.38	72.34 +0.96	64.86	65.85 +0.99	73.56	73.93 +0.37	72.21	72.97 +0.76	70.71	71.25 +0.54
FT	72.03	73.91 +1.88	61.78	62.62 +0.84	73.25	74.37 +1.12	71.59	72.54 +0.95	70.58	71.46 +0.88
DIST	76.22	76.71 +0.49	68.50	69.48 +0.98	75.51	76.00 +0.49	74.26*	74.65 +0.39	73.89	74.56 +0.67
DKD	76.70	77.01 +0.31	69.71	69.91 +0.20	75.63	76.66 +1.03	74.85	74.97 +0.12	74.68	75.08 +0.40
ITRD	77.09	77.59 +0.50	70.39	70.92 +0.53	76.12	76.50 +0.38	75.18	75.31 +0.13	74.86*	75.17 +0.31
WTTM	75.42	75.60 +0.18	69.16	69.42 +0.26	76.37	76.56 +0.19	74.58	74.70 +0.12	74.33	74.64 +0.31

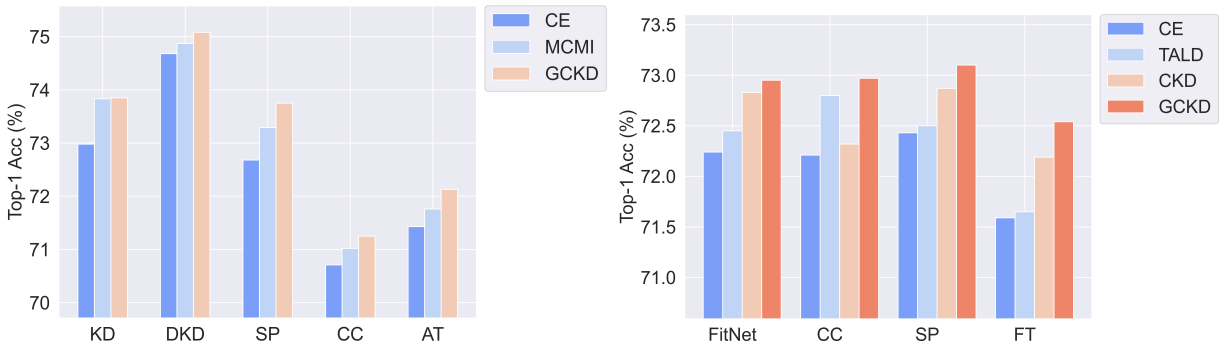
centroids in each iteration. This ensures a more stable and theoretically sound optimization process, leading to comparable or improved performance over MCMI without the need to modify the teacher model, as shown in the CIFAR-100 results in Fig. 6.3.

GCKD distillation performance on ImageNet														
Teacher-Student	**Method**	**Teacher Performance**			**KD**		**ReviewKD**		**DKD**		**LSKD**		**WSLD**	
		Training Acc.	**CMI**	**NLL**	**Top1**	**Top5**	**Top1**	**Top5**	**Top1**	**Top5**	**Top1**	**Top5**	**Top1**	**Top5**
RN34-RN18	NLL	84.690	0.718	0.560	70.66	90.05	71.61	90.51	71.70	90.41	71.42	90.29	71.73	90.45
	GCKD	83.685	0.930	0.433	71.41	90.13	71.77	90.55	71.88	90.46	71.70	90.58	71.98	90.43
	Δ	/	/	/	+0.75		+0.16		+0.18		+0.28		+0.25	
RN50-MNV1	NLL	87.457	0.598	0.448	70.50	89.80	72.56	91.00	72.05	91.05	72.18	91.06	72.02	91.03
	GCKD	87.191	0.829	0.312	70.84	89.79	72.85	91.23	72.50	91.06	72.57	91.06	72.60	90.91
	Δ	/	/	/	+0.34		+0.29		+0.45		+0.39		+0.58	

Table 6.3: Top-1 and Top-5 test accuracy (%) of student models on the ImageNet validation set for six KD methods, demonstrating the performance gain of GCKD. For both teacher-student pairs (ResNet-34 → ResNet-18 and ResNet-50 → MobileNetV1), using the GCKD teacher consistently improves student accuracy over the standard NLL teacher baseline across all KD methods. RN and MN denote ResNet and MobileNet, respectively.

6.5.6 Comparison with CKD and TALD

GCKD also advances perturbation-based distillation methods like Coded Knowledge Distillation (CKD). While CKD introduces perturbations to mitigate teacher overfitting, it operates within a discrete space and focuses on a sample-wise entropy objective. GCKD extends this concept by operating in a significantly larger continuous space, which allows it to directly maximize the true CMI of the teacher's output. This approach enables the generation of a richer, more informative "coded dataset." As a result, when applied to the same knowledge distillation methods on CIFAR-100, GCKD consistently achieves comparable or superior performance to CKD and another related method, TALD (Fig. 6.3). This demonstrates the benefit of performing gradient-based CMI maximization in a continuous domain.



(a) VGG-13 / VGG-8 (b) WRN-40-2 / WRN-40-1

Figure 6.3: (a) Comparison with MCMI. (b) Comparison with CKD and TALD.

6.6 Analysis

In this section, we provide a detailed analysis to validate the core mechanisms of our proposed GCKD framework. First, we demonstrate the convergence and stability of the alternating optimization algorithm. Second, we present a geometric analysis of the teacher's output probability space to visualize how GCKD achieves its objective of CMI maximization.

6.6.1 Convergence of the Alternating Optimization

The central innovation of GCKD is the alternating optimization procedure (Stage 1 of Algorithm 3), designed to solve the intractable CMI maximization problem defined in Equation equation 6.10. To verify its effectiveness, we track key metrics during the alternating optimization loop within a single epoch. Figure 6.4 plots the evolution of these metrics for a ResNet-34 teacher on ImageNet.

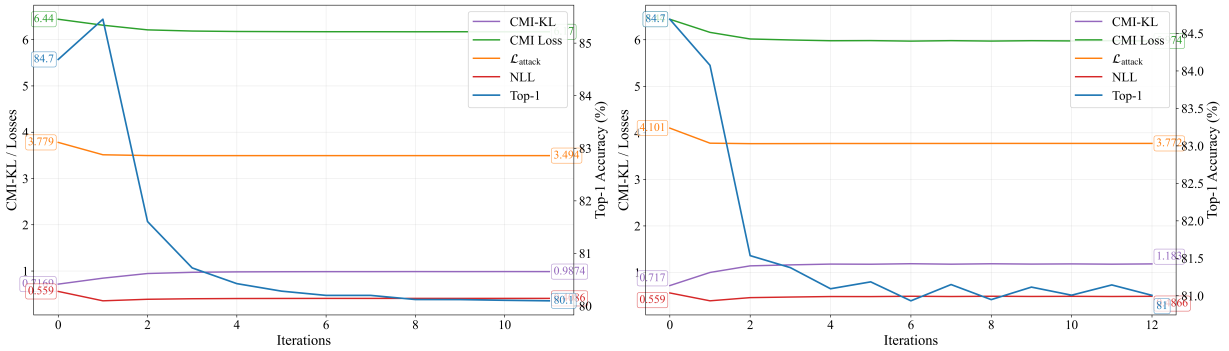
The optimization process aims to minimize the per-sample attack loss, $\mathcal{L}$attack, defined in Equation equation 6.24. This loss consists of a cross-entropy term, $H(\delta y_j, f_{\theta_t}(x'j))$, and a CMI-maximization term, $-\lambda \sum i f_{\theta_t}(x'_j)[i] \ln Q'(x_j \mid i, y_j)$. As shown in Figure 6.4, the total attack loss (L_{attack}) and the CMI-related component (CMI Loss) consistently decrease over the iterations. This demonstrates that the algorithm successfully optimizes the objective and finds perturbations that maximize the empirical CMI of the teacher's outputs.

This CMI maximization is achieved by design at the cost of a slight, controlled degradation in the teacher's performance on the perturbed dataset. We observe a modest increase in the Negative Log-Likelihood (NLL)—the cross-entropy component of the loss—and a corresponding drop in the teacher's Top-1 accuracy. This illustrates the fundamental trade-off at the heart of GCKD: sacrificing a small amount of teacher confidence to generate softer, more structured, and thus more informative (higher CMI) supervisory signals for the student. The stability of this process across different PGD step sizes ($\alpha = 0.03$ and $\alpha = 0.1$) further validates the robustness of our alternating optimization approach.

6.6.2 Geometric Analysis of the GCKD Output Space

To understand the effect of CMI maximization geometrically, we visualize the teacher's output probability distributions. According to our formulation, maximizing the class-conditional CMI, $I(X'; \hat{Y}'|Y = y)$, is equivalent to maximizing the average KL divergence between individual predictions $f(x')$ and their corresponding perturbed centroid S'_y (Equation 6.11). Geometrically, this means the cluster of output distributions for a given class should become less concentrated and more dispersed around its centroid.

Figure 6.5 provides a 2-simplex visualization for a 3-class subset of CIFAR-100, showing how the output space of a VGG-13 GCKD teacher evolves. Initially (left), the teacher produces overconfident, near one-hot predictions, which appear as tight clusters near the vertices of the simplex. As the GCKD



(a) $\alpha = 0.03$ (b) $\alpha = 0.1$

Figure 6.4: Evolution of key metrics during the alternating optimization stage (Stage 1 of GCKD) for a ResNet-34 teacher on ImageNet. The plots track the teacher's Top-1 accuracy and NLL on the evolving perturbed dataset, alongside the CMI-related losses, over several iterations within a single epoch. As the algorithm iterates, the CMI-related losses decrease, indicating successful CMI maximization. This is achieved at the cost of a slight, controlled increase in NLL and a decrease in the teacher's own accuracy, demonstrating the trade-off at the heart of GCKD. The two subplots compare the effect of different PGD step sizes, showing the stability of the optimization process.

optimization progresses across epochs, the output distributions (colored dots) and their corresponding perturbed centroids S'_y (cross markers) systematically shift away from the vertices and toward the center of the simplex. This visual "spreading" of the clusters directly corresponds to an increase in the empirical CMI, confirming that the GCKD teacher is learning to produce softer, more informative predictions while maintaining correct classification.

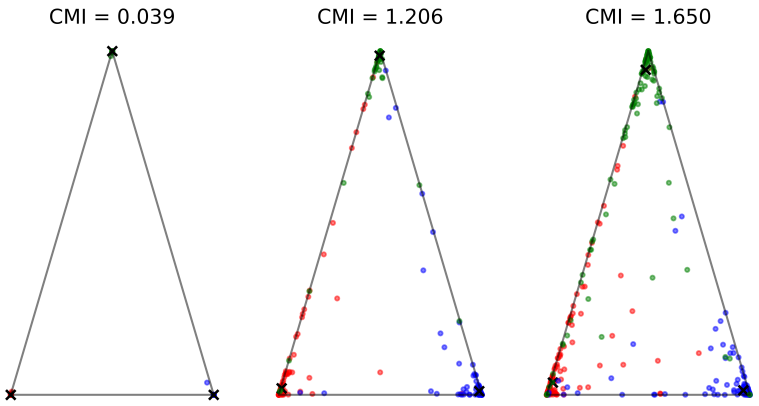
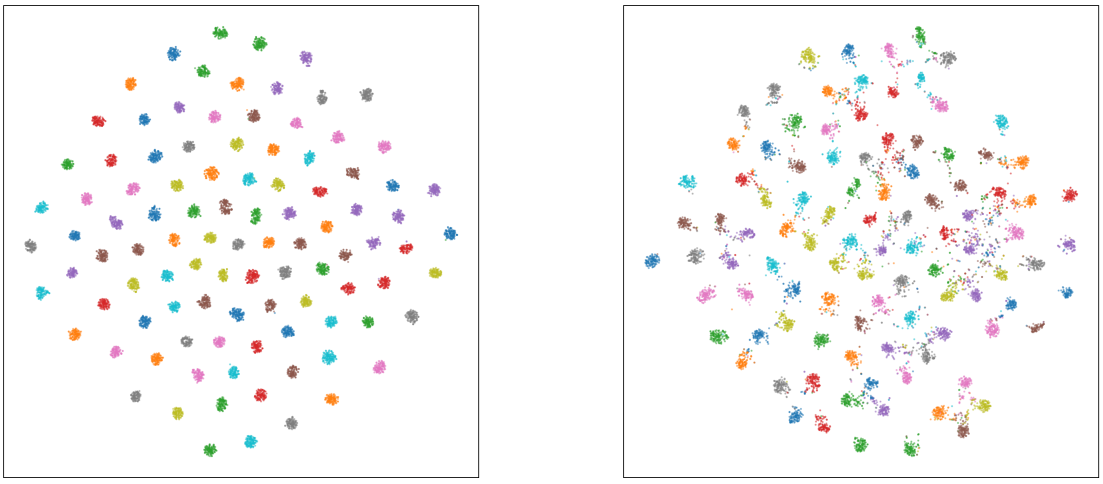


Figure 6.5: The output probability vectors of a trained GCKD teacher for a three-class classification task, visualized on a 2-simplex over different epochs. Three classes are randomly selected from the CIFAR-100 training set. As training progresses, the empirical CMI increases, and the output distributions (colored dots) and class centroids (cross markers) move from the vertices toward the center, indicating the teacher is producing softer, more informative predictions. Temperature scaling ($T = 1.0$) is applied for visualization.

Figure 6.6 offers a complementary, global visualization using t-SNE to project the output probability distributions for all 100 classes of CIFAR-100. A clear distinction emerges between a conventionally trained CE teacher and the GCKD teacher. The CE teacher's outputs form extremely tight, well-separated clusters (a), indicative of low-entropy, overconfident predictions. In contrast, the GCKD teacher's outputs form visibly more dispersed and spread-out clusters (b). This dispersion is not a flaw but the desired outcome of our CMI maximization objective. It provides compelling visual evidence that GCKD successfully reshapes the teacher's output geometry to generate a richer, higher-CMI supervisory signal, which is more beneficial for knowledge distillation.



(a) CE teacher

(b) GCKD teacher

Figure 6.6: t-SNE visualization of the output probability distributions from the CIFAR-100 training set using a VGG-13 teacher. (a) CE teacher: The clusters are tightly concentrated, corresponding to low-entropy, overconfident predictions. (b) GCKD teacher: The clusters are visibly more dispersed, providing a visual confirmation that GCKD successfully increases the teacher's CMI by generating softer output distributions.

6.7 Chapter Summary

In this chapter, we have introduced Generalized Coded Knowledge Distillation (GCKD), a novel framework that enhances knowledge transfer from a fixed teacher by optimizing its input space. GCKD leverages gradient-based adversarial perturbations to generate a "coded dataset" specifically crafted to maximize the teacher's Conditional Mutual Information (CMI). This is achieved via a novel alternating optimization algorithm that directly and accurately maximizes the true CMI, enabling the teacher to produce softer, more informative supervision signals without altering its parameters. Extensive experiments on CIFAR-100 and ImageNet, across diverse network architectures and distillation methods, demonstrate that GCKD consistently improves student model performance, achieving accuracy gains of up to 1.71%. Our convergence and geometric analyses further validate that the alternating optimization procedure is stable and effectively reshapes the teacher's output probability space to increase CMI, confirming the theoretical underpinnings of our approach. Beyond its empirical success, GCKD presents a practical and generaliz-

able paradigm for student-oriented teacher enhancement. By operating in a continuous input space and directly targeting CMI maximization, it overcomes the limitations of discrete perturbation methods and proxy-based CMI estimators. Crucially, because the teacher model remains unmodified, GCKD provides a powerful, drop-in solution for improving knowledge distillation in real-world scenarios where teacher models are proprietary, immutable, or already deployed.

Chapter 7

Differentiable JPEG-based Input Perturbation for Knowledge Distillation Amplification

While the Generalized Coded Knowledge Distillation (GCKD) framework introduced in Chapter 6 validates the principle of maximizing CMI via input perturbation, its methodology presents practical limitations. The reliance on iterative, backpropagation-based optimization for each sample is computationally demanding and scales imperfectly, creating a prohibitive bottleneck for large-scale tasks.

This chapter proposes a fundamentally more efficient solution: Differentiable JPEG-based Input Perturbation (DJIP). Our approach moves from a per-example optimization to a *generalized, amortized optimization* by training a single differentiable JPEG layer to learn a universal perturbation policy. This architectural shift replaces the iterative optimization with a fast forward pass, eliminates the need for a pre-generated coded dataset, and yields a simple "plug-and-play" module suitable for real-world applications. Consequently, DJIP transforms the theoretically sound but practically cumbersome approach of GCKD into a lightweight, scalable, and readily deployable framework.

7.1 Introduction

In the preceding chapter, we introduced Generalized Coded Knowledge Distillation (GCKD), a framework designed to enhance knowledge distillation by maximizing the teacher's Conditional Mutual Information (CMI) without altering the teacher's parameters. GCKD achieves this by generating a *coded dataset* through an iterative, backpropagation-based optimization process applied to each input sample. While this per-example optimization successfully validates the principle of teacher-agnostic CMI maximization, it introduces a significant computational bottleneck, hindering its practical scalability and deployment

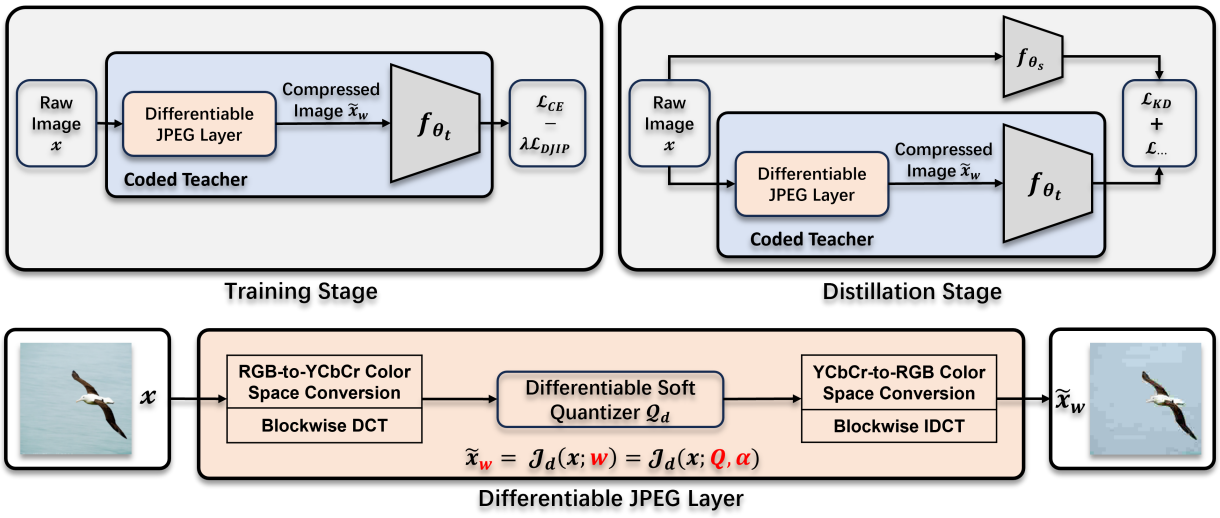


Figure 7.1: The overall training framework of DJIP is shown, centered around a differentiable JPEG coded teacher module (blue), which integrates a differentiable JPEG layer (orange). The process involves two stages: differentiable JPEG layer training and student distillation, with trainable parameters highlighted in red.

in resource-constrained environments. This challenge of high overhead is a common limitation among input-centric methods that require generating or individually optimizing supplementary samples.

To address this critical issue of computational inefficiency while preserving the benefits of CMI maximization [137], we propose **Differentiable JPEG-based Input Perturbation (DJIP)**. DJIP evolves the GCKD framework by replacing the costly per-example optimization with an efficient, amortized optimization strategy. This is accomplished by inserting a single, trainable differentiable JPEG layer before the fixed teacher model. This layer learns a universal perturbation policy—trained once and applied to any input via a fast forward pass—which effectively amortizes the optimization cost across the entire dataset.

The composite model, termed a differentiable JPEG *coded teacher*, enables end-to-end learning of the JPEG coding parameters tailored to maximize the teacher's perturbed CMI. Building upon the alternating optimization algorithm developed in Chapter 6, DJIP formulates the objective as a double minimization problem, allowing for the dynamic update of class-wise centroids at each iteration. This ensures a more stable and effective training process for the DJIP layer. In doing so, DJIP transforms the computationally intensive principles of GCKD into a lightweight, scalable, and practical solution for amplifying knowledge distillation performance.

Our primary contributions in this chapter are:

- **Differentiable JPEG-based Input Perturbation (DJIP):** We introduce a novel, plug-and-play framework that improves knowledge transfer by learning a universal input perturbation policy via

a trainable differentiable JPEG layer, significantly reducing computational overhead compared to per-example optimization methods.

- **Amortized CMI Maximization:** We extend the alternating optimization algorithm from the previous chapter to efficiently train the differentiable JPEG layer. This amortized approach effectively maximizes the perturbed CMI by iteratively updating both the coding parameters and the class centroids.
- **Comprehensive Empirical Evaluation:** We conduct extensive experiments across diverse datasets and architectures, demonstrating DJIP's generalizability, efficiency, and orthogonality to a wide range of existing knowledge distillation pipelines.

7.2 Related Work

This chapter builds upon the foundational and related works discussed in Chapter 6, with a specific focus on literature pertinent to amortized input perturbation and differentiable image processing within the context of knowledge distillation.

7.2.1 Student-Oriented Teacher Design

As established in Chapter 6, student-oriented teacher models aim to produce more informative supervision signals. Methods range from using earlier training checkpoints [17, 121] to explicitly optimizing the teacher with objectives like Maximum Conditional Mutual Information (MCMI) [137]. A common trait of these approaches is the requirement to modify or retrain the teacher model, a constraint our work aims to avoid. While DJIP adopts the CMI maximization principle central to MCMI, it does so within the teacher-agnostic paradigm of GCKD. Our key innovation is an efficient, amortized optimization framework that maximizes CMI without any teacher modification, addressing a fundamental limitation of prior student-oriented methods.

7.2.2 Input Perturbation in Knowledge Distillation

The preceding chapter provided an overview of input perturbation techniques, including adversarial methods that explore the teacher's decision boundaries [42, 84] and compression-based approaches like Coded Knowledge Distillation (CKD) shown in Chapter 5 [98]. These methods, alongside our GCKD framework from Chapter 6, typically rely on generating multiple variants or individually optimized versions of each input, which incurs substantial computational overhead.

DJIP distinguishes itself by shifting from this per-example paradigm to an amortized one. Unlike the adversarial methods that generate new samples or the discrete, selection-based compression of CKD, DJIP learns a single, continuous transformation via a differentiable JPEG layer. This allows for a more

flexible and computationally efficient exploration of the input perturbation space. By learning a global policy rather than optimizing locally for each sample, DJIP provides a scalable solution that maintains the core benefits of input perturbation while being practical for large-scale deployment. Furthermore, the use of a differentiable layer enables direct, gradient-based optimization of the perturbation parameters to maximize a global objective (CMI), a more principled approach than the heuristic selection criteria used in methods like CKD.

7.3 Methodology

In this section, we first present the overall DJIP framework, then detail the objective function we use, and finally introduce our novel alternating optimization algorithm.

7.3.1 Overall Framework

As illustrated in Figure 7.1, the overall DJIP framework consists of two stages:

Differentiable JPEG Layer Training: The JPEG-DL framework shown in Chapter 4 [99] introduces a differentiable JPEG layer into any underlying DNN. Specifically, JPEG-DL incorporates a novel differentiable JPEG layer as the input layer of the underlying DNN architecture. This layer simulates the standard JPEG codec but replaces the non-differentiable $\mathcal{Q}_u$ with a differentiable soft quantizer. In Figure 7.1, this layer first perturbs the input image x into $\tilde{x}_w = \mathcal{J}_d(x, w)$, which is then fed to the teacher model. Under the objective function proposed in the next section, which balances the CE loss and the DJIP loss introduced later, the JPEG coding parameters are optimized to maximize the perturbed CMI.

Student Distillation: In this stage, the trained JPEG layer is integrated into the standard KD framework to perturb the teacher's input images. With its input perturbed by the trained JPEG layer to increase the perturbed CMI, the teacher can provide more informative supervision signals to student.

7.3.2 Objective Function

With reference to Figure 7.1, given $\mathcal{J}_d$, $\tilde{X}_w$ is a deterministic function of X. Hence, the variables form a Markov chain $Y \to X \to \tilde{X}_w \to \hat{Y}$, which implies

$$I(X; \hat{Y} \mid Y) = I(X\tilde{X}_w; \hat{Y} \mid Y) = I(\tilde{X}_w; \hat{Y} \mid Y). \tag{7.1}$$

We refer to $I(\tilde{X}_w; \hat{Y} \mid Y)$ as the perturbed CMI. Our goal is to perturb the input X such that the perturbed CMI is maximized to a certain extent, while simultaneously minimizing the CE loss with respect

to the ground-truth labels. Following the spirit of Ye *et al.* [137], we formulate our CE–CMI optimization problem over w as

$$\min_{w} \Big\{ \mathbb{E}_X \big[H(P_{Y|X}, f(\tilde{X}_w)) \big] - \lambda \, I(\tilde{X}_w; \hat{Y} \mid Y) \Big\}, \tag{7.2}$$

where $\lambda > 0$ is a hyper-parameter that balances the CE-CMI trade-off. In contrast to Ye *et al.* [137], where optimization is carried out over the pretrained model parameters θ, the DJIP framework freezes the teacher and updates only the JPEG parameters w. Following the same preliminaries in Chapter 6, we can define a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ drawn from an unknown distribution $P_{X,Y}$, we can approximate the CMI of model f by its empirical value. To be specific, let $\mathcal{D}_y = \{x_j \in \mathcal{D} : y_j = y\}$. Denote the size of $\mathcal{D}_y$ by $|\mathcal{D}_y|$. The empirical values of CMI can be calculated as follows:

$$I(X; \hat{Y} \mid Y) = \frac{1}{N} \sum_{y \in [C]} \sum_{x_j \in \mathcal{D}_y} D_{\mathrm{KL}}(f(x_j) \| S_y), \quad \text{where } S_y = \frac{1}{|\mathcal{D}_y|} \sum_{x_j \in \mathcal{D}_y} f(x_j), \text{ for } y \in [C]. \tag{7.3}$$

However, as stated by Ye *et al.* [137], maximizing $I(X; \hat{Y} \mid Y)$ poses certain challenges because the term S_y depends on $f(x_j), \forall x_j \in \mathcal{D}_y$ (see Equation 7.3) which is not well-suited for numerical solutions and cannot be efficiently parallelized on GPUs. The same challenges apply to the optimization problem in Equation 7.2 as well. Ye *et al.* [137] circumvents this issue by fixing the centroids S_y obtained from a pretrained model. Nevertheless, these centroids may shift during fine-tuning, so using fixed centroids may partially mitigate the issue, but does not fully resolve the theoretical limitations of their method.

To address the above challenges theoretically, we introduce a dummy "backward channel" and re-formulate the optimization problem in Equation 7.2 as a double minimization problem, as shown in the following theorem and proved in as follows:

Theorem 3 *For any $i, y \in [C]$, let $Q(\cdot \mid i, y)$ denote a dummy conditional distribution over the input space X ("backward channel"). Then for any $\lambda > 0$,*

$$\begin{aligned} &\min_{w} \Big\{ \mathbb{E}_X \big[H(P_{Y|X}, f(\tilde{X}_w)) \big] - \lambda \, I(\tilde{X}_w; \hat{Y} \mid Y) \Big\} \\ &\equiv \min_{w} \min_{\{Q(\cdot|i,y)\}} \left\{ \sum_{x} P(x) H(P_{Y|X}(\cdot \mid x), f(\tilde{x}_w)) - \lambda \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln Q(x \mid i, y) \right\}, \end{aligned} \tag{7.4}$$

where the inner minimization above is achieved when

$$Q(x \mid i, y) = \frac{P_{X|Y}(x \mid y) f(\tilde{x}_w)[i]}{P_{\hat{Y}|Y}(i \mid y)}. \tag{7.5}$$

To prove Theorem 3, we first introduce an auxiliary distribution $Q(\cdot \mid i, y)$ and derive a new expression for $I(\tilde{X}_w; \hat{Y} \mid Y)$ as follows:

$$
\begin{aligned}
I(\tilde{X}_w; \hat{Y} \mid Y) = I(X; \hat{Y} \mid Y) &= \sum_{y \in [C]} P_Y(y) I(X; \hat{Y} \mid Y = y) \\
&= \sum_{y \in [C]} P_Y(y) \sum_x P_{X|Y}(x \mid y) \left[\sum_{i=1}^{C} P_{\hat{Y}|XY}(\hat{Y} = i \mid x, y) \times \ln \frac{P_{\hat{Y}|XY}(\hat{Y} = i \mid x, y)}{P_{\hat{Y}|Y}(\hat{Y} = i \mid Y = y)} \right] \\
&= \sum_y \sum_x P(x, y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln \frac{f(\tilde{x}_w)[i]}{P_{\hat{Y}|Y}(\hat{Y} = i \mid Y = y)} \\
&= \sum_y P_Y(y) \sum_x P_{X|Y}(x \mid y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln \frac{P_{X|Y}(x \mid y) f(\tilde{x}_w)[i]}{P_{X|Y}(x \mid y) P_{\hat{Y}|Y}(i \mid y)} \\
&= \sum_y P_Y(y) \sum_{i=1}^{C} P_{\hat{Y}|Y}(i \mid y) \sum_x \frac{P_{X|Y}(x \mid y) f(\tilde{x}_w)[i]}{P_{\hat{Y}|Y}(i \mid y)} \ln \frac{P_{X|Y}(x \mid y) f(\tilde{x}_w)[i]}{P_{X|Y}(x \mid y) P_{\hat{Y}|Y}(i \mid y)} \qquad (7.6) \\
&= \max_{\{Q(\cdot|i,y)\}} \sum_y P_Y(y) \sum_{i=1}^{C} P_{\hat{Y}|Y}(i \mid y) \sum_x \frac{P_{X|Y}(x \mid y) f(\tilde{x}_w)[i]}{P_{\hat{Y}|Y}(i \mid y)} \ln \frac{Q(x \mid i, y)}{P_{X|Y}(x \mid y)} \\
&= \max_{\{Q(\cdot|i,y)\}} \sum_{x,y} P(x, y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln \frac{Q(x \mid i, y)}{P_{X|Y}(x \mid y)}, \qquad (7.7)
\end{aligned}
$$

where Equation 7.6 follows from the cross entropy inequality, and the maximization in Equation 7.7 is achieved when

$$
Q^*(x \mid i, y) = \frac{P_{X|Y}(x \mid y) f(\tilde{x}_w)[i]}{P_{\hat{Y}|Y}(i \mid y)}. \qquad (7.8)
$$

Thus, the single minimization problem in equation 7.2 can be converted into a double minimization

problem over w and $\{Q(\cdot \mid i, y)\}$ as follows:

$$
\begin{aligned}
&\min_{w} \Big\{ \mathbb{E}_X H(P_{Y|X}, f(\tilde{X}_w)) - \lambda \, I(\tilde{X}_w; \hat{Y} \mid Y) \Big\} \\
&= \min_{w} \Big\{ \mathbb{E}_X H(P_{Y|X}, f(\tilde{X}_w)) - \lambda \, I(X; \hat{Y} \mid Y) \Big\} \\
&= \min_{w} \left\{ \mathbb{E}_X H(P_{Y|X}, f(\tilde{X}_w)) - \lambda \max_{\{Q(\cdot|i,y)\}} \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln \frac{Q(x \mid i, y)}{P_{X|Y}(x \mid y)} \right\} \\
&\equiv \min_{w} \left\{ \mathbb{E}_X H(P_{Y|X}, f(\tilde{X}_w)) - \lambda \max_{\{Q(\cdot|i,y)\}} \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln Q(x \mid i, y) \right\} \qquad (7.9) \\
&= \min_{w} \left\{ \mathbb{E}_X H(P_{Y|X}, f(\tilde{X}_w)) + \lambda \min_{\{Q(\cdot|i,y)\}} - \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln Q(x \mid i, y) \right\} \\
&= \min_{w} \min_{\{Q(\cdot|i,y)\}} \left\{ \sum_{x} P(x) H(P_{Y|X}(\cdot \mid x), f(\tilde{x}_w)) - \lambda \sum_{x,y} P(x,y) \sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln Q(x \mid i, y) \right\}, \quad \square
\end{aligned}
$$

where the symbol "$\equiv$" in Equation 7.9 indicates equivalence up to an additive constant independent of $Q(x \mid i, y)$ and w, which is omitted here for clarity.

In practice, the training set is randomly partitioned into B mini-batches $\mathcal{B}^b$ for $b \in [B]$, each of size $|\mathcal{B}|$. When the joint distribution of (X, Y) is unknown, the objective function can be approximated by its empirical estimate over a mini-batch $\mathcal{B}$. Accordingly, based on Theorem 3, the empirical objective function $\mathcal{L}_{\text{emp}}$ for the proposed DJIP method can be formulated as:

$$
\mathcal{L}_{\mathcal{B}}(\lambda, w, \{Q(\cdot|i,y)\}) = \underbrace{\frac{1}{|\mathcal{B}|} \sum_{(x,y) \in \mathcal{B}} (-\ln f(\tilde{x}_w)[y])}_{\mathcal{L}_{\text{CE}}} - \lambda \underbrace{\frac{1}{|\mathcal{B}|} \sum_{(x,y) \in \mathcal{B}} \left[\sum_{i=1}^{C} f(\tilde{x}_w)[i] \ln Q(x|i,y) \right]}_{\mathcal{L}_{\text{DJIP}}}. \qquad (7.10)
$$

7.3.3 An Alternating CMI Maximization Algorithm

Based on equation 7.4 to equation 7.10, we are now ready to present our algorithm for solving the optimization problem in equation 7.2, which optimizes w and $\{Q(\cdot \mid i, y)\}$ alternatively to minimize the objective function in equation 7.10:

Step 1: Fix w, let $\mathcal{D}_{x,y} = \{(x_j, y_j) \in \mathcal{D} : x_j = x, y_j = y\}$. According to 7.5, $\{Q(\cdot \mid i, y)\}$ can be updated in two steps. (1) update the centroids S_y empirically according to Equation 7.3; and (2) calculate

the empirical version of $Q(x \mid i, y)$ according to Equation 7.5:

$$S_y[i] = P_{\hat{Y}|Y}(i \mid y) = \frac{1}{|\mathcal{D}_y|} \sum_{x_j \in \mathcal{D}_y} f(\mathcal{J}_d(x_j, w))[i], \quad \forall i, y \in [C], \tag{7.11}$$

$$Q(x \mid i, y) = \frac{P_{X|Y}(x \mid y) f(\mathcal{J}_d(x, w))[i]}{P_{\hat{Y}|Y}(i \mid y)} = \frac{\frac{|\mathcal{D}_{x,y}|}{|\mathcal{D}_y|} f(\mathcal{J}_d(x, w))[i]}{P_{\hat{Y}|Y}(i \mid y)}, \quad \forall (x, y) \in \mathcal{D}, i \in [C]. \tag{7.12}$$

Step 2: Fix $\{Q(\cdot \mid i, y)\}$, w can be updated using a standard deep learning process through stochastic gradient descent (SGD).

A detailed pseudo-code of the alternating optimization algorithm is provided in Algorithm 4.

7.4 Experiments

7.4.1 Implementation Details

Setups for Teacher Training Stage

We follow the implementation and design of the differentiable JPEG layer in [99], but set the minimum quantization step to 1, which is more commonly used in reality, instead of 0. The stochastic gradient descent (SGD) optimizer is used with a momentum of 0.9 and a weight decay of 5×10^{-4} for all teacher training experiments.

A key feature of the differentiable soft quantizer $\mathcal{Q}_d$ in the JPEG layer is its variable softness, controlled by the trainable sharpness parameter α. However, as shown in [99], when α is sufficiently large, its gradient vanishes, preventing effective updates. Therefore, we initialize all entries of $\boldsymbol{\alpha}$ to a large constant value of 20 and exclude $\boldsymbol{\alpha}$ from training. Thus, the trainable parameters w of the DJIP teacher are the quantization tables $\boldsymbol{Q}$, which we initialize with all ones.

For **CIFAR-100**, all pretrained teachers are adopted from the official repository used in [114]. We use a learning rate of 0.1 and train for 20 epochs with a batch size of 32 over 2 GPUs (total batch size 64). For all CIFAR-100 experiments comparing with the CE teacher, including those involving ViT models, we set the hyperparameter λ in the objective function 7.10 to 0.5 by default. An exception is made for the VGG-13 and MobileNetV2 teacher-student pair in Table 7.2, where we set $\lambda = 0.6$. To demonstrate that DJIP is further orthogonal to MCMI in Table 7.8, we train the DJIP-MCMI teacher with $\lambda = 0.3$. Throughout the alternating training process, we always use the full training set $\mathcal{D}$ to update the centroids as proposed in Algorithm 4.

For **ImageNet**, we adopt pretrained teachers from the PyTorch official repository [89]. DJIP teachers are trained for one epoch (5005 iterations) with a learning rate of 0.01, using 4 GPUs and a batch size of 64

Algorithm 4 The proposed alternating algorithm for solving the optimization problem in Theorem 3.

Input: DNN f_θ as f; JPEG layer $\mathcal{J}_d$ with trainable parameters w; training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with its B mini-batches $\{\mathcal{B}^b\}_{b\in[B]}$; class labels C; number of epochs T and hyper-parameter λ.

Initialize w^0.
repeat
  for $b = 1$ to B **do**
    [*Update* S]:
    Fix w^{b-1}. Update centroids $\{S_y^b\}_{y\in[C]}$ according to Equation 7.11:

$$S_y^b[i] \leftarrow \frac{1}{|\mathcal{D}_y|} \sum_{x_j \in \mathcal{D}_y} f(\mathcal{J}_d(x_j, w^{b-1}))[i], \quad \forall i, y \in [C]. \tag{7.13}$$

    [*Update* Q]:
    Fix w^{b-1}. Calculate $Q^b(x \mid i, y)$, according to Equation 7.12:

$$Q^b(x \mid i, y) \leftarrow \frac{\frac{|\mathcal{D}_{x,y}|}{|\mathcal{D}_y|} f(\mathcal{J}_d(x, w^{b-1}))[i]}{S_y^b[i]}, \quad \forall (x, y) \in \mathcal{B}^b, i \in [C]. \tag{7.14}$$

    [*Update* w]
    Fix $\{Q^b(\cdot \mid i, y)\}_{(x,y)\in\mathcal{B}^b}^{i\in[C]}$. Update weights w^{b-1} to w^b by using SGD over the objective function

$$\mathcal{L}_{\mathcal{B}^b}(\lambda, w^{b-1}, \{Q^b(\cdot \mid i, y)\}_{(x,y)\in\mathcal{B}^b}^{i\in[C]}).$$

  end for
  Set $w^0 \leftarrow w^B$
until T epochs are completed

return Trained parameters w^B.

per GPU (total batch size 256). We experiment with various values of λ, namely $\{0.4, 1.0, 1.4, 2.0, 2.4, 3.0\}$, and select $\lambda = 1.4$ for ResNet-34 and $\lambda = 2.0$ for ResNet-50.

When applied to large-scale datasets such as ImageNet, updating centroids at each iteration via Algorithm 4 becomes computationally expensive. To reduce this complexity, we construct a balanced subdataset $\hat{\mathcal{D}}$ by randomly sampling $|\hat{\mathcal{D}}_y| = 16$ instances per class from the training set. By replacing the full training set in Equation 7.11 with $\hat{\mathcal{D}}$, a lightweight variant of the alternating algorithm can be derived. In the first iteration, the dummy distribution $\{Q(\cdot \mid i, y)\}$ is updated using the entire training set. Thereafter,

it is updated using the sub-dataset and an exponential moving average (EMA) with a smoothing factor $\alpha = 0.9$.

Thus, the resulting pseudo-code on ImageNet experiments is similar to Algorithm 4, except that Equation 7.13 is replaced by:

$$S_y^b[i] \leftarrow \alpha \times S_y^{b-1}[i] + (1-\alpha) \times \frac{1}{|\hat{\mathcal{D}}_y|} \sum_{x_j \in \hat{\mathcal{D}}_y} f_{\theta^b}(x_j)[i], \quad \forall i, y \in [C]. \tag{7.15}$$

To further handle the increased complexity of ImageNet tasks compared to CIFAR-100 tasks, following the setups of [99], we incorporate five rounds of $\mathcal{Q}_d$ quantization operation with independent trainable parameters inside the differentiable JPEG layer to expand the compression search space. Moreover, we follow the gradient magnitude control method proposed by [99], with the Gradient Scaling Constants $\hbar_m$ set to 20 to control the gradient magnitude to ensure more stable updates for $\boldsymbol{Q}$.

Setups for Student Distillation Stage

For experimental setups of all knowledge distillation variants in this chapter, we use SGD with momentum 0.9 and weight decay of 1×10^{-4} as the optimizer.

For **CIFAR-100**, we follow the distillation setups in [114]. Specifically, we train the student models for 240 epochs with a batch size of 64 on 1 GPU. The initial learning rate is set to 0.05 and decayed by a factor of 0.1 at epochs 150, 180, and 210. For MobileNetV2, ShuffleNetV1, and ShuffleNetV2, we use a smaller initial learning rate of 0.01.

For **ImageNet**, we adopt the training configuration in [143]: 100 epochs, a learning rate of 0.1, and a batch size of 256 on a single GPU. The only exception is ReviewKD [14], which uses a batch size of 128 and is trained on two GPUs.

For both datasets, we evaluated several state-of-the-art distillation methods using their official implementations and reported hyper-parameters, and successfully reproduced the results reported in their original papers.

Terminologies. To evaluate the performance of DJIP, we conduct a series of experiments. This section presents the main experimental results and demonstrates the extent to which DJIP improves accuracy over state-of-the-art KD methods. For clarity, we denote the teachers trained solely with CE loss, with MCMI estimator [137], and with the proposed DJIP method as the *CE* teacher, *MCMI* teacher, and *DJIP* teacher, respectively.

Plug-and-Play Nature. In all experiments reported, the JPEG layer functions as a lens to facilitate improved teacher distillation. Once removed, the model reverts to its conventional form without any residual effect. Furthermore, no hyperparameters of the underlying KD methods are modified; all configurations remain identical to those used in the original benchmark settings.

We conduct extensive experiments on ImageNet and CIFAR-100 with diverse model architectures. Moreover, we show that DJIP can be effectively applied to cross-paradigm distillation between CNNs and Vision Transformers (ViTs). The results further demonstrate that DJIP is complementary to existing techniques and remains orthogonal to the latest state-of-the-art benchmarks.

7.4.2 Experimetnal Results

CIFAR-100 Results. The CIFAR-100 dataset is a widely used benchmark for image classification, comprising 60,000 color images with a resolution of 32×32 pixels, categorized into 100 classes. Following the experimental setup of [114], we conduct experiments with 7 teacher-student pairs sharing identical architectures (see Table 7.1) and 6 pairs using different architectures (see Table 7.2). Each experiment is repeated across three independent runs, and the average accuracy is reported.

For a comprehensive comparison, we evaluate our DJIP teacher against the conventional CE teacher using state-of-the-art distillation methods. These include logit-based approaches: KD [44], DKD [143], DIST [47], and WTTM [145]; relation-based approaches: CC [90] and RKD [77]; and feature-based approaches: AT [140], FitNet [96], FT [52], SP [117], CRD [114], LSKD [109], and ITRD [77]. Methods of the same category are grouped in Tables 7.1 and 7.2.

For both CE and DJIP teachers, we report their test accuracy, as well as CE loss and CMI values measured on the training set without data augmentation. As shown in Tables 7.1 and 7.2, replacing the CE teacher with the DJIP teacher consistently improves student performance, regardless of whether the teacher and student architectures are the same. These improvements are observed across all evaluated distillation methods, with accuracy gains of up to 2.44%. Notably, the improvements are more pronounced when the teacher and student architectures differ, i.e., when there exists a larger capacity gap between them.

ImageNet Results. ImageNet is a large-scale dataset used in visual classification tasks, containing approximately 1.2 million training images and 50,000 validation images. Following the implementation in [143], we conduct experiments on two widely used teacher-student pairs (see Table 7.3) and four representative distillation methods: KD [44], DKD [143], WSLD [146], ReviewKD [14], and LSKD [109].

Across all knowledge transfer methods reported in Table 7.3, we observe that replacing the CE teacher with the DJIP teacher consistently improves the Top-1 accuracy. Details on the training setups, including training of DJIP teachers, student distillation, and the choice of λ, as well as an analysis on the effect of λ for DJIP teachers, are provided in the Subsection 7.4.1.

Vision Transformer Results. In previous experiments on CIFAR-100, we demonstrated the effectiveness of DJIP for distillation between CNN architectures, whether identical or different. In this section, following [39, 61], we extend DJIP to address the challenge of cross-paradigm distillation between CNNs and ViTs on CIFAR-100. The experimental results are summarized in Table 7.4. We include one teacher-student pair for each paradigm setting: ViT-to-CNN, CNN-to-ViT, and ViT-to-ViT. The results clearly show that our proposed method is also effective for knowledge transfer across heterogeneous architectural paradigms and highlight the potential of DJIP in distilling ViTs.

Table 7.1: The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the same architecture. The subscript denotes the improvement achieved by replacing the CE teacher with the DJIP teacher. We use asterisk $(^*)$ to identify the results reproduced on our local machines. Note that the test accuracy $(\%)$ of the teacher here is over the perturbed image.

DJIP distillation on CIFAR-100 with the **same architectures**.														
Teacher	WRN-40-2		WRN-40-2		ResNet-56		ResNet-110		ResNet-110		VGG-13		ResNet-32×4	
	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP
Acc	75.61	70.30	75.61	70.30	72.34	66.32	74.31	69.03	74.31	69.03	74.64	72.67	79.41	75.51
CELoss	0.052	0.370	0.052	0.370	0.101	0.594	0.033	0.421	0.033	0.421	0.007	0.130	0.004	0.185
CMI	0.026	0.501	0.026	0.501	0.158	0.724	0.061	0.565	0.061	0.565	0.015	0.252	0.006	0.276
Student	WRN-16-2		WRN-40-1		ResNet-20		ResNet-20		ResNet-32		VGG-8		ResNet-8×4	
Acc	73.26		71.98		69.06		69.06		71.14		70.36		72.50	
KD	74.92	75.64 +0.72	73.54	74.41 +0.87	70.66	71.20 +0.54	70.67	71.65 +0.98	73.08	73.71 +0.63	72.98	74.01 +1.03	73.33	74.38 +1.05
DKD	75.63*	76.08 +0.45	74.85	75.14 +0.29	71.58*	71.86 +0.28	71.51	71.72 +0.21	74.11	74.22 +0.11	74.68	74.93 +0.25	76.32	76.55 +0.23
DIST	75.51	75.98 +0.47	74.26	74.99 +0.73	71.75	71.97 +0.22	71.65	71.90 +0.25	73.69	73.90 +0.21	73.89	74.31 +0.42	76.31	76.60 +0.29
WTTM	76.37	76.70 +0.33	74.58	74.98 +0.40	71.92	72.15 +0.23	71.67	71.90 +0.23	74.13	74.32 +0.19	74.44	74.81 +0.37	76.06	76.61 +0.55
CC	73.56	73.80 +0.24	72.21	72.48 +0.27	69.63	69.93 +0.30	69.48	69.87 +0.39	71.48	71.86 +0.38	70.71	71.12 +0.41	72.97	73.23 +0.26
RKD	73.35	74.09 +0.74	72.22	72.36 +0.14	69.61	70.19 +0.58	69.25	69.85 +0.60	71.82	72.45 +0.63	71.48	71.84 +0.36	71.90	72.69 +0.79
AT	74.08	74.51 +0.43	72.77	73.22 +0.45	70.55	70.81 +0.26	70.22	70.63 0.41	72.31	72.84 +0.53	71.43	72.03 +0.60	73.44	73.94 +0.50
FitNet	73.58	74.20 +0.62	72.24	72.84 +0.60	69.21	69.83 +0.62	68.99	69.40 +0.41	71.06	71.42 +0.36	71.02	71.92 +0.90	73.50	73.83 +0.33
FT	73.25	73.55 +0.30	71.59	71.93 +0.34	69.84	70.20 +0.36	70.22	70.70 +0.48	72.37	72.54 +0.17	70.58	71.28 +0.70	72.86	73.76 +0.90
SP	73.83	74.35 +0.52	72.43	72.99 +0.56	69.67	70.83 +1.16	70.04	71.05 +1.01	72.69	73.37 +0.68	72.68	73.42 +0.74	72.94	73.62 +0.68
ITRD	76.12	76.33 +0.21	75.18	75.23 +0.05	71.26*	71.44 +0.18	71.52*	71.86 +0.34	74.26	74.30 +0.04	74.86	75.00 +0.14	76.19	76.28 +0.09
CRD	75.48	76.01 +0.53	74.14	74.57 +0.43	71.16	71.61 +0.45	71.46	71.79 +0.33	73.48	73.92 +0.44	73.94	74.35 +0.41	75.51	75.85 +0.34
LSKD	76.11	76.35 +0.24	74.37	74.89 +0.52	71.26*	71.39 +0.13	71.48	71.60 +0.12	73.67*	73.92 +0.25	74.36	74.90 +0.54	76.62	76.96 +0.34

Table 7.2: The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the different architectures. The subscript denotes the improvement achieved by replacing the CE teacher with the DJIP teacher. We use asterisk $(^*)$ to identify the results reproduced on our local machines. Note that the test accuracy $(\%)$ of the teacher here is over the perturbed image.

DJIP distillation on CIFAR-100 with **different architectures**.												
Teacher	ResNet-50		ResNet-50		ResNet-32×4		ResNet-32×4		WRN-40-2		VGG-13	
	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP	CE	DJIP
Acc	79.34	75.32	79.34	75.32	79.41	75.51	79.41	75.51	75.61	70.30	74.64	72.62
CELoss	0.004	0.229	0.004	0.229	0.004	0.185	0.004	0.185	0.052	0.370	0.007	0.130
CMI	0.009	0.341	0.009	0.341	0.006	0.276	0.006	0.276	0.026	0.501	0.015	0.252
Student	MobileNetV2		VGG-8		ShuffleNetV1		ShuffleNetV2		ShuffleNetV1		MobileNetV2	
Acc	64.60		70.36		70.50		71.82		70.50		64.60	
KD	67.35	69.50 +2.15	73.81	74.48 +0.67	74.07	75.64 +1.57	74.45	76.24 +1.79	74.83	76.36 +1.53	67.37	68.87 +1.50
DKD	70.35	71.18 +0.83	73.94	75.87 +1.93	76.45	77.24 +0.79	77.07	77.52 +0.45	76.70	77.16 +0.46	69.71	70.20 +0.49
DIST	68.66	69.59 +0.93	74.11	74.80 +0.69	76.34	76.64 +0.30	77.35	77.95 +0.60	76.22	76.69 +0.47	68.50	69.01 +0.51
WTTM	69.59	69.98 +0.39	74.82	75.19 +0.37	74.37	74.95 +0.58	76.55	77.30 +0.75	75.42	76.24 +0.82	69.16	69.41 +0.25
CC	65.43	65.79 +0.36	70.25	70.92 +0.67	71.14	72.17 +1.03	71.29	73.25 +1.96	71.38	72.22 +0.84	64.86	65.83 +0.97
RKD	64.43	65.56 +1.13	71.50	71.93 +0.43	72.28	73.40 +1.12	73.21	74.27 +1.06	72.21	73.85 +1.64	64.52	66.04 +1.52
AT	58.58	60.02 +1.44	71.84	72.42 +0.58	71.73	73.93 +2.20	72.73	74.16 +1.43	73.32	75.27 +1.95	59.40	61.24 +1.84
FitNet	63.16	64.05 +0.89	69.39*	69.57 +0.18	73.59	74.47 +0.88	73.54	74.65 +1.11	73.73	74.02 +0.29	64.14	65.46 +1.32
FT	60.99	62.82 +1.83	70.29	71.24 +0.95	71.75	73.41 +1.66	72.50	73.80 +1.30	72.03	73.80 +1.77	61.78	62.57 +0.79
SP	68.08	68.58 +0.50	73.34	73.87 +0.53	73.48	75.92 +2.44	74.56	76.28 +1.72	74.52	76.32 +1.80	66.30	67.68 +1.38
ITRD	71.34	72.06 +0.72	75.49	75.91 +0.42	76.91	77.30 +0.39	77.40	77.92 +0.52	77.09	77.31 +0.22	70.39	70.91 +0.52
CRD	69.11	70.01 +0.90	74.30	74.55 +0.25	75.11	75.79 +0.68	75.65	76.32 +0.67	76.05	76.21 +0.16	69.73	69.89 +0.16
LSKD	69.02	70.39 +1.37	74.88*	75.21 +0.33	75.67*	76.38 +0.71	75.56	77.12 +1.56	76.56*	76.78 +0.22	68.61	69.93 +1.32

Table 7.3: The test accuracy (%) of student networks on ImageNet, with teacher-student pairs of the same and different architectures.

DJIP distillation on ImageNet											
Teacher	Student	Method	Acc	CEloss	CMI	KD	AT	DKD	LSKD	WSLD	ReviewKD
ResNet-34	ResNet-18	CE	73.310	0.560	0.720	70.66	70.70	71.70	71.42	71.73	71.61
		DJIP	72.832	0.581	0.738	71.65	70.78	72.08	71.65	71.87	71.72
73.31	69.76	Δ	/	/	/	+0.99	+0.08	+0.38	+0.23	+0.14	+0.11
ResNet-50	MobileNetV1	CE	76.130	0.449	0.600	70.50	69.56	72.05	72.18	72.02	72.56
		DJIP	75.088	0.497	0.649	70.92	70.57	72.52	72.37	72.78	73.04
76.16	68.87	Δ	/	/	/	+0.42	+1.01	+0.47	+0.19	+0.76	+0.48

Table 7.4: The test accuracy (%) of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of different architectural models. The subscript denotes the improvement achieved by replacing the CE teacher with a DJIP teacher.

DJIP distillation on CIFAR-100 with **different model paradigms**.											
Teacher	Student	Method	Acc	CELoss	CMI	KD	DIST	DKD	CC	RKD	CRD
ViT-S	ResNet-18	CE	92.04	0.330	0.100	77.26	76.49	78.10	74.26	73.72	76.60
		DJIP	88.72	0.404	0.171	79.06	78.04	80.15	74.87	75.85	77.75
92.04	74.01	Δ	/	/	/	+1.80	+1.55	+2.05	+0.61	+2.13	+1.15
ConvNeXt-T	DeiT-T	CE	88.41	0.421	0.229	72.99	73.55	74.60	68.01	69.79	65.94
		DJIP	86.89	0.460	0.256	75.00	74.70	75.57	69.60	70.38	66.63
88.41	68.00	Δ	/	/	/	+2.01	+1.15	+0.97	+1.59	+0.59	+0.69
ViT-S	DeiT-T	CE	92.04	0.330	0.100	69.86	70.57	71.41	68.62	69.39	65.46
		DJIP	88.72	0.404	0.171	73.97	73.96	74.90	69.86	70.11	65.75
92.04	68.00	Δ	/	/	/	+4.11	+3.39	+3.49	+1.24	+0.72	+0.29

7.5 Analysis

7.5.1 Ablation Study

DJIP introduces two key components: (1) a differentiable JPEG layer, and (2) an alternating CMI maximization algorithm. In this section, we conduct an ablation study to isolate and analyze the effectiveness of each component.

As shown in Table 7.5, when the differentiable JPEG layer is added but the fixed-centroid method from [137] is retained (referred to as 'JMCMI'), the resulting method generally outperforms various KD baselines. Furthermore, when the alternating maximization algorithm is additionally applied, resulting in our proposed DJIP, the performance is further improved over JMCMI in general.

In summary, both the differentiable JPEG layer and the alternating CMI maximization algorithm contribute positively to the overall performance. Each component individually enhances the KD process, and their combined use in DJIP yields the best results, demonstrating a synergistic effect.

Table 7.5: The test accuracy $(\%)$ of student networks on CIFAR-100 (averaged over 3 runs), with teacher-student pairs of the same- and different-architecture. Noted that here, method JMCMI refers to applying the fix-centroid strategy to train the differentiable JPEG layer only.

DJIP Vs. JMCMI Vs. CE teacher on CIFAR-100															
Teacher-Student	Method	Acc	CELoss	CMI	KD	DKD	DIST	CC	RKD	AT	FitNet	FT	SP	ITRD	CRD
WRN-40-2 WRN-16-2	CE	75.61	0.052	0.026	74.92	75.63	75.51	73.56	73.35	74.08	73.58	73.25	73.83	76.12	75.48
	JMCMI	70.02	0.373	0.505	75.53	75.81	75.83	73.62	73.74	74.28	73.75	73.39	74.12	75.93	76.39
	DJIP	70.30	0.370	0.501	75.64	76.08	75.98	73.80	74.09	74.51	74.20	73.55	74.35	76.33	76.01
WRN-40-2 ShuffleNetV1	CE	75.61	0.052	0.026	74.83	76.70	76.22	71.38	72.21	73.32	73.73	72.03	74.52	77.09	76.05
	JMCMI	70.02	0.373	0.505	75.95	76.56	76.85	71.92	73.39	75.10	73.81	73.25	75.96	77.14	76.14
	DJIP	70.30	0.370	0.501	76.36	77.16	76.69	72.22	73.85	75.27	74.02	73.80	76.32	77.31	76.21

7.5.2 GCKD's Advantage Through PGD in Solving the Inner Minimization Problem

While DJIP introduces an efficient amortized approach for generating perturbations via a differentiable JPEG layer, it is important to acknowledge that GCKD still stands out when solving the inner minimization problem defined in Equation (6.17). This superiority stems from its use of Projected Gradient Descent (PGD) within an alternating optimization framework.

The GCKD objective is formulated as a double minimization problem over the perturbed inputs X' and auxiliary distributions Q', as shown in Equation (6.17). The inner loop seeks to minimize:

$$\min_{x'_j} \; H(\delta_{y_j}, f(x'_j)) - \lambda \sum_{i=1}^{C} f(x'_j)[i] \ln Q'(x_j|i, y_j) \quad \text{s.t. } d(x_j, x'_j) \leq \epsilon$$

This optimization is computationally challenging because it requires finding perturbations that maximize CMI while preserving classification accuracy. PGD provides a principled, iterative solution to this inner minimization, refining perturbations over multiple steps and projecting them back into the allowed perturbation space. Compared to single-step methods like FGSM or amortized variants, PGD achieves a more thorough search for optimal perturbations, resulting in coded datasets that significantly improve student performance.

Empirical results (see Table 7.6) confirm that multi-step PGD consistently yields the highest student accuracy across all tested configurations, reinforcing its role as the most effective strategy for generating

adversarially coded datasets in GCKD. This robustness comes at a higher computational cost, but the gains in distillation quality justify its selection as an alternative perturbation method in our framework. Additionally, in Table 7.7, we compared GCKD when we use the amortized PGD with DJIP. This shows that GCKD still outperforms DJIP in general.

Table 7.6: Impact of adversarially coded datasets on student performance during knowledge distillation using DIST as an orthogonal method. The table compares conventional NLL-based teachers with GCKD and DJIP teachers across two student architectures (VGG-8 and MobileNetV2). Results show that incorporating stronger adversarial attacks, such as PGD, consistently yields the highest accuracy improvements over the baseline.

Teacher → Student	**Method**	**Conventional**	**GCKD**			**DJIP**
			FGSM	**Amortized PGD**	**PGD**	
VGG-13 → VGG-8	KD	72.98	73.61	73.86	**74.12**	74.01
	DIST	73.89	74.14	74.56	**74.71**	74.31
VGG-13 → MobileNetV2	KD	67.37	68.34	68.78	**68.91**	68.87
	DIST	68.50	68.93	69.48	**69.61**	69.01

Table 7.7: Test accuracy (%) of student networks on CIFAR-100 under different teacher-student configurations. The comparison highlights the performance gains achieved by DJIP and GCKD over the conventional CE baseline, demonstrating their effectiveness in enhancing knowledge distillation across diverse architectures.

DJIP vs GCKD distillation performance on CIFAR100 for **different** and **same** architectures.

Teacher	WRN-40-2			VGG-13			WRN-40-2			WRN-40-2			VGG-13		
	CE	DJIP	GCKD	CE	DJIP	GCKD	CE	DJIP	GCKD	CE	DJIP	GCKD	CE	DJIP	GCKD
Student	ShuffleNetV1			MobileNetV2			WRN-16-2			WRN-40-1			VGG-8		
Accuracy	70.50			64.60			73.26			71.98			70.36		
KD	74.83	**76.36** +1.53	76.30 +1.47	67.37	**68.87** +1.50	68.78 +1.41	74.92	**75.64** +0.72	75.61 +0.35	73.54	**74.41** +0.87	73.86 +0.22	72.98	**74.01** +1.03	73.85 +0.87
FitNet	73.73	74.02 +0.29	**74.41** +0.68	64.14	65.46 +1.32	**66.02** +1.88	73.58	74.20 +0.62	**74.41** +0.83	72.24	72.84 +0.60	**72.95** +0.71	71.02	71.92 +0.90	71.92 +0.90
AT	73.32	75.27 +1.95	**75.28** +1.96	59.40	**61.24** +1.84	61.17 +1.77	74.08	74.51 +0.43	**74.87** +0.79	72.77	73.22 +0.45	**73.40** +0.63	71.43	72.03 +0.60	**72.13** +0.70
SP	74.52	76.32 +1.80	**76.51** +1.71	66.30	67.68 +1.38	**67.82** +1.52	73.83	74.35 +0.52	**74.36** +0.53	72.43	72.99 +0.56	**73.10** +0.37	72.68	73.42 +0.74	**73.75** +1.07
CC	71.38	72.22 +0.84	**72.34** +0.96	64.86	65.83 +0.97	**65.85** +0.99	73.56	73.80 +0.24	**73.93** +0.67	72.21	72.48 +0.27	**72.97** +0.76	70.71	71.12 +0.41	**71.25** +0.54
FT	72.03	73.80 +1.77	**73.91** +1.88	61.78	62.57 +0.79	**62.62** +0.84	73.25	73.55 +0.30	**74.37** +1.12	71.59	71.93 +0.34	**72.54** +0.57	70.58	71.28 +0.70	**71.46** +0.88
DIST	76.22	76.69 +0.47	**76.71** +0.49	68.50	69.01 +0.51	**69.48** +0.98	75.51	75.98 +0.47	**76.00** +0.49	74.26*	**74.99** +0.73	74.65 +0.39	73.89	74.31 +0.42	**74.56** +0.67
DKD	76.70	**77.16** +0.46	77.01 +0.31	69.71	**70.20** +0.49	69.91 +0.20	75.63	76.08 +0.45	**76.66** +1.03	74.85	**75.14** +0.29	74.97 +0.12	74.68	74.93 +0.25	**75.08** +0.40
ITRD	77.09	77.31 +0.22	**77.59** +0.50	70.39	70.91 +0.52	**70.92** +0.53	76.12	76.33 +0.21	**76.50** +0.38	75.18	75.23 +0.05	**75.31** +0.13	74.86*	75.00 +0.14	**75.17** +0.31
WTTM	75.42	**76.24** +0.82	75.60 +0.18	69.16	69.41 +0.25	**69.42** +0.26	76.37	**76.70** +0.33	76.56 +0.19	74.58	**74.98** +0.40	74.70 +0.12	74.44	**74.81** +0.37	74.64 +0.20

7.5.3 Orthogonality Over MCMI

Since DJIP shares MCMI's goal of maximizing the teacher's CMI, we examine whether our method is orthogonal to MCMI and can further improve performance. Following the experimental setup in [137], we prepend our trainable JPEG layer to the MCMI teacher to obtain the DJIP–MCMI teacher reported in Table 7.8.

The results in Table 7.8 indicate that applying our DJIP method yields additional improvements in Top-1 accuracy, with gains of up to 0.92% absolute. These findings suggest that DJIP is not only complementary but also orthogonal to MCMI, as it explicitly optimizes the input space, which is not considered in MCMI.

Table 7.8: The test accuracy (%) of student networks on CIFAR-100 (averaged over three runs), with teacher- student pairs of the same- and different- architecture. The results obtained by the MCMI method are used as the baseline. This table demonstrates that our method provides additional orthogonality to MCMI, which is itself an orthogonal approach. Note that in this context, DJIP-MCMI refers to applying our proposed method using the fine-tuned MCMI teacher as the underlying model.

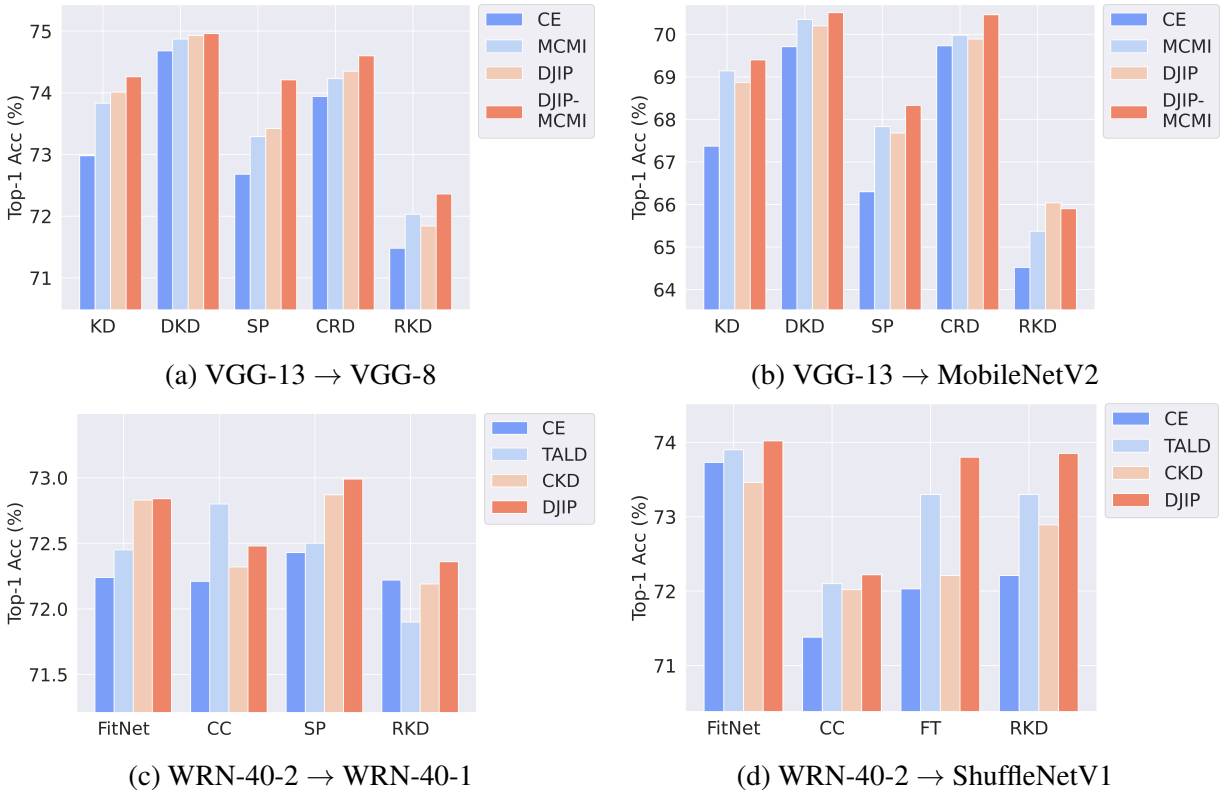
DJIP-MCMI teacher Vs. MCMI teacher on CIFAR-100										
Teacher	Student	Method	Acc	CELoss	CMI	KD	DKD	SP	CRD	RKD
VGG-13	VGG-8	MCMI	73.96	0.076	0.1298	73.83	74.87	73.29	74.23	72.03
		DJIP-MCMI	73.76	0.067	0.1402	74.26	74.96	74.21	74.60	72.36
74.64	70.36	Δ	/	/	/	+0.43	+0.09	+0.92	+0.37	+0.33
VGG-13	MobileNetV2	MCMI	73.96	0.076	0.1298	69.14	70.35	67.83	69.98	65.37
		DJIP-MCMI	73.76	0.067	0.1402	69.40	70.51	68.33	70.46	65.90
74.64	64.60	Δ	/	/	/	+0.26	+0.16	+0.50	+0.48	+0.53

7.5.4 Comparison with CKD and MCMI

Our DJIP method addresses the fixed-centroid problem of MCMI by introducing an alternating optimization algorithm. Consequently, when compared with the CIFAR-100 and ImageNet results[1] of MCMI applied to the same KD methods, our method achieves comparable or better performance across most cases, as presented in Figure 7.2.

It is important to note that MCMI has a much larger degree of freedom, as it optimizes the parameters of the pretrained model, whereas our method only adjusts the 128 quantization parameters of the JPEG layer. Despite this limitation, our method still matches or outperforms MCMI in multiple scenarios, demonstrating the effectiveness of learning solely through compression parameters.

[1] Please refer to Table 2 of [137]



(a) VGG-13 → VGG-8

(b) VGG-13 → MobileNetV2

(c) WRN-40-2 → WRN-40-1

(d) WRN-40-2 → ShuffleNetV1

Figure 7.2: (a), (b) Comparison with MCMI under different student architectures on CIFAR-100. (c), (d) Comparison with CKD and TALD under different student architectures on CIFAR-100.

7.5.5 Comparison with CKD and TALD

As discussed in Section 7.2, our DJIP method is capable of exploring a significantly larger continuous quantization space than CKD. Consequently, when compared with the CIFAR-100 and ImageNet results[2] of CKD applied to the same KD methods, our approach consistently achieves comparable or superior performance. A brief comparison of CIFAR-100 experimental results is shown in Figure 7.2.

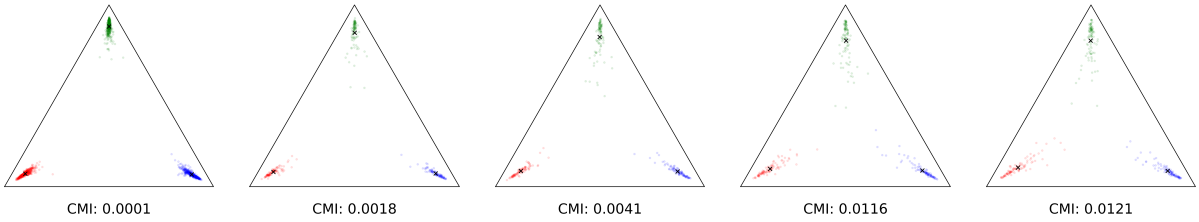
It is worth noting that CKD adaptively selects the optimal quantization table for each input image; that is, two different images may be compressed using different tables chosen to maximize teacher effectiveness, which is computationally expensive. In contrast, our method employs a fixed quantization table shared across all input images throughout the entire distillation process. Moreover, since CKD achieves comparable performance to another input-perturbed-based method, TALD [84], we also include TALD for comparison in Figure 7.2.

[2]Please refer to Table 3 of [98]

7.5.6 Visualize the Output Probability Space in Different Epochs

We visualize how the output probability distributions of the ResNet-32×4 DJIP teacher evolve across epochs during a single training run, as illustrated in Figure 7.3. The simplex is visualized by projecting the 3-class softmax probability vectors onto a 2D plane, where the one-hot vectors of each class are mapped to the vertices of an equilateral triangle. Temperature scaling with $T = 4.0$ is applied to the softmax outputs.

Initially, the teacher produces highly confident predictions resembling one-hot vectors, with output probabilities concentrated near the corners of the simplex. As training progresses, facilitated by the differentiable JPEG layers, both the output distributions and the centroids of each class cluster gradually shift toward the center of the simplex, leading to an increase in CMI values. While maintaining correct classifications, the predictions become less confident yet more informative. Figure 7.3 further illustrates the convergence of centroid trajectories, thereby validating the convergence of the alternating optimization algorithm.

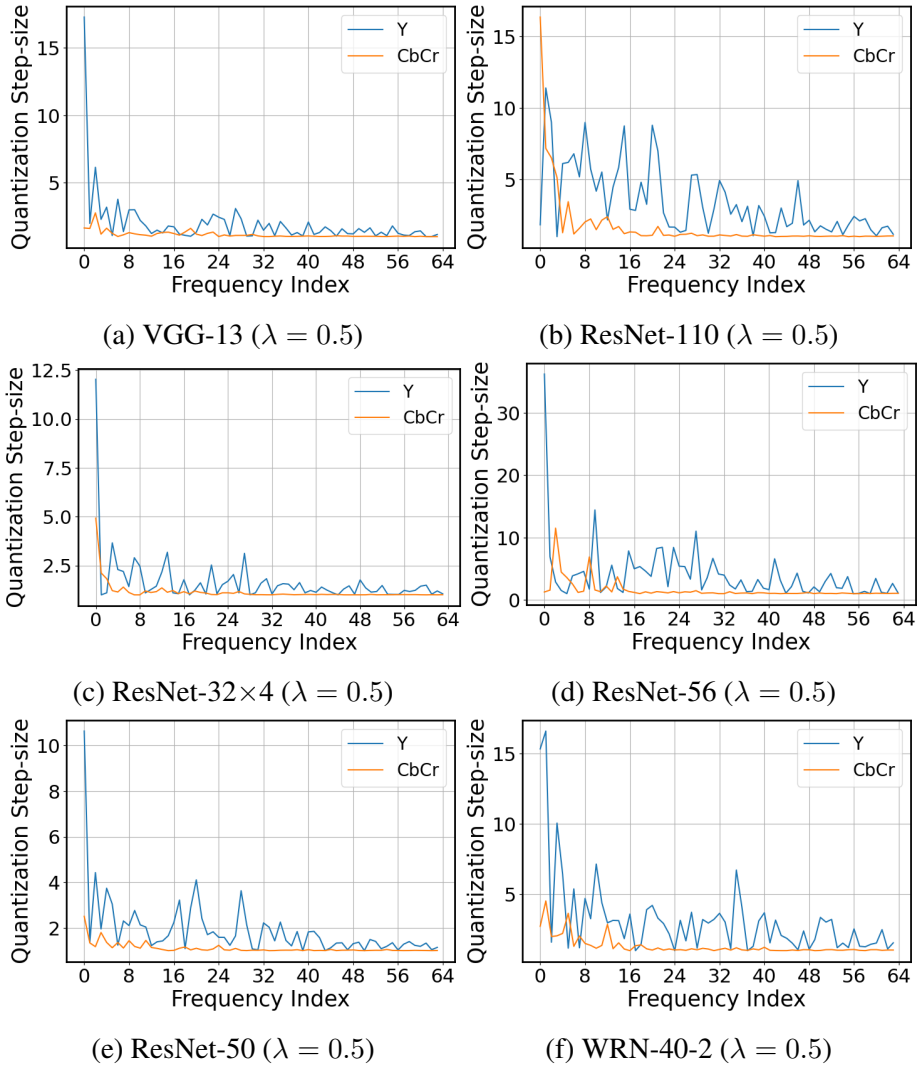


(a) Untrained (b) Epoch 5 (c) Epoch 10 (d) Epoch 15 (e) Epoch 20

Figure 7.3: The output probability vectors of the trained DJIP teacher for a three-class classification task are visualized on a 2-simplex over different epochs within a single training run. Three classes are randomly selected from the CIFAR-100 training set, with 100 samples per class. Temperature scaling with $T = 4.0$ is applied to the softmax outputs to enhance distributional smoothness. In the plots, cross markers denote the class centroids, and the CMI values are computed based solely on these three selected classes.

7.5.7 Trained Quantization Table Visualization

The quantization tables trained for the DJIP teacher, corresponding to the results in Tables 7.1, 7.2, and 7.3, are presented in Figures 7.4 and 7.5. These results indicate that, across different datasets, the models consistently apply stronger compression to the Y channel than to the Cb and Cr channels, which in turn leads to higher CMI values after training.



(a) VGG-13 ($\lambda = 0.5$)

(b) ResNet-110 ($\lambda = 0.5$)

(c) ResNet-32×4 ($\lambda = 0.5$)

(d) ResNet-56 ($\lambda = 0.5$)

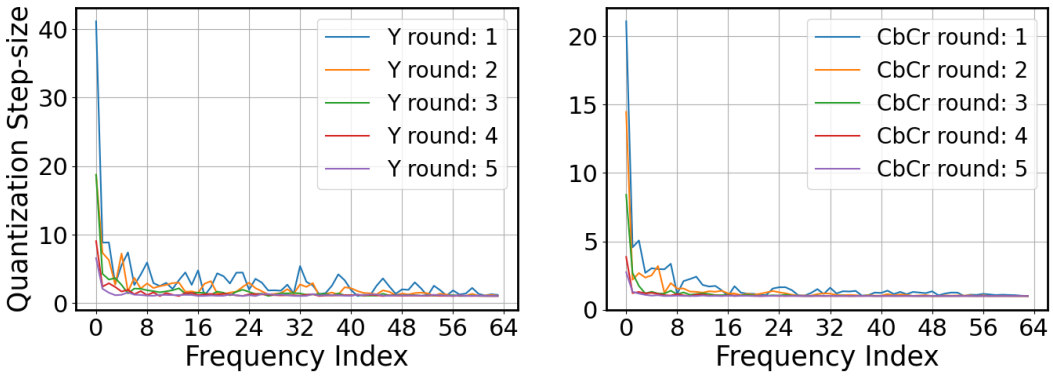
(e) ResNet-50 ($\lambda = 0.5$)

(f) WRN-40-2 ($\lambda = 0.5$)

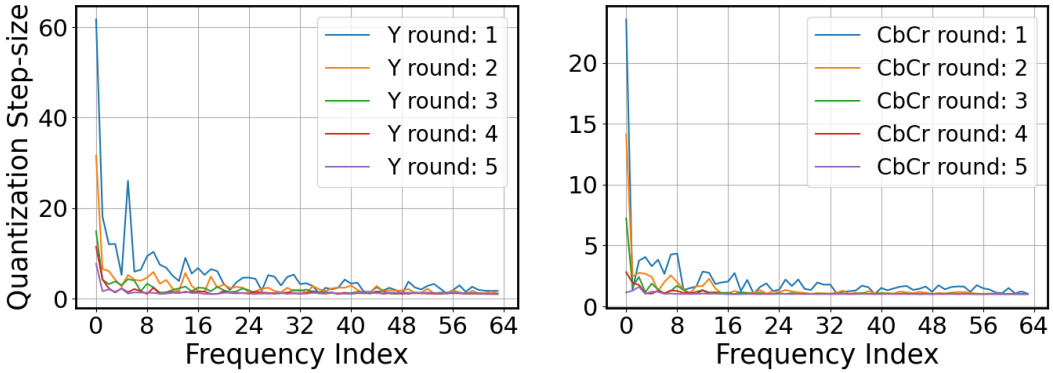
Figure 7.4: Trained quantization tables on CIFAR-100 models used in Table 7.1 and 7.2.

7.5.8 Visualize the Output Probability Space of DNNs

A more comprehensive comparison of the output probability space between the CE teacher and the DJIP teacher is shown in Figure 7.6, with temperature scaling $T = 4.0$ as well. We observe that the clusters corresponding to the DJIP teacher become less concentrated.



(a) ResNet-34 ($\lambda = 1.4$)



(b) ResNet-50 ($\lambda = 2.0$)

Figure 7.5: Trained quantization tables on ImageNet models in Table 7.3.

7.5.9 DJIP with other input perturbation method

As stated in Theorem 3, the parameter ω represents arbitrary perturbation parameters and is fully agnostic to the specific choice of input perturbation mechanism. Consequently, the proposed alternating optimization framework is not restricted to JPEG quantization tables. The differentiable JPEG module adopted in our experiments serves merely as a convenient instantiation, selected due to its simplicity, computational efficiency, and the availability of robust differentiable implementations. In principle, this perturbation layer may be replaced by any differentiable codec. As another illustrative example, we additionally incorporate a convolutional autoencoder as the input perturbation module.

We employ a symmetric convolutional autoencoder designed for CIFAR-resolution images. The encoder is composed of convolution, batch normalization, and ReLU layers, followed by a max-pooling operation in the middle. The decoder mirrors this structure using convolution and transposed-convolution layers to upsample feature maps and recover the original spatial resolution. As observed in Section 7.5.7,

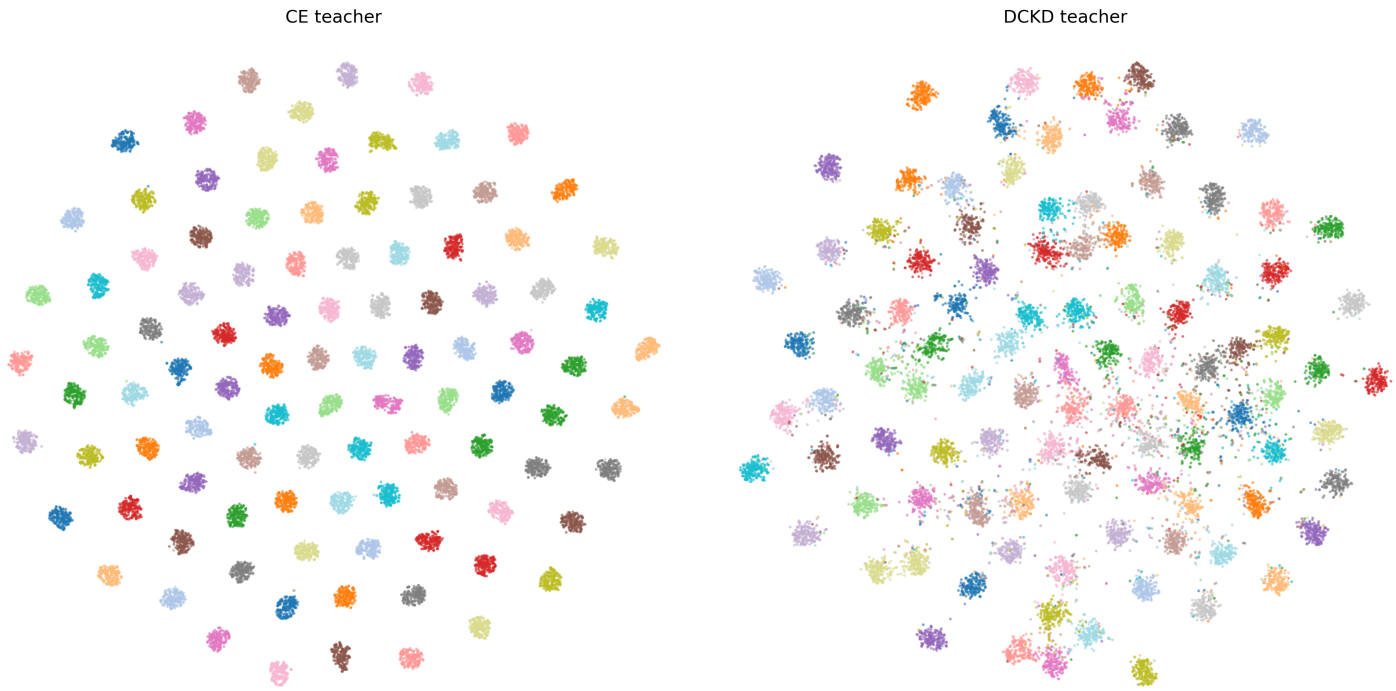


Figure 7.6: t-SNE [118] visualization of features extracted from the CIFAR-100 training set with 500 samples per cluster in all 100 categories.

the optimized quantization tables consistently exhibit small step sizes. Motivated by this, we choose a latent dimensionality comparable to the input, enabling the autoencoder to learn an identity-preserving or denoising transformation. A final sigmoid activation constrains the reconstructed output to the normalized pixel range.

For pretraining, the autoencoder is trained on CIFAR-100 with standard data augmentation. Optimization is performed using Adam with a learning rate of 1×10^{-3}, and the reconstruction objective is mean squared error (MSE). Training proceeds for 200 epochs with a batch size of 64. On the CIFAR-100 test set, the pretrained autoencoder achieves a PSNR of $41.52\,\mathrm{dB}$, an SSIM of 0.9961, and an MSE of 8.3×10^{-5}, indicating high-fidelity reconstruction.

Following the training framework in Figure 7.1, we insert the autoencoder before the classifiers and optimize its parameters using the alternating optimization algorithm. We train for 20 epochs with a learning rate of 1×10^{-3} and set $\lambda = 0.3$. After optimizing the perturbation module, standard distillation is performed. The final results are reported in Table 7.9.

Table 7.9: The test accuracy $(\%)$ of students on CIFAR-100 with autoencoder as input perturbation method.

Teacher	Student	Method	CMI	KD	FitNet	FT	SP	CRD	CC	RKD
ResNet-34	ResNet-18	CE	0.015	72.98	71.02	70.58	72.68	73.94	70.71	71.48
		DJIP	0.067	73.31	71.31	71.03	72.92	74.05	71.13	71.77
74.64	70.36	Δ	/	+0.33	+0.29	+0.45	+0.24	+0.11	+0.42	+0.29

7.5.10 Few Shot Classification

In conventional few-shot classification, only a β percent of instances from each class are made available for model training [72]. Translating this idea into the KD setting, the JPEG layer is trained on the full dataset, while only a β percent subset of samples per class is used to train the student during distillation.

To examine the effectiveness of the proposed DJIP teacher under limited-data conditions, we conduct experiments on CIFAR-100 using WRN-40-2 WRN-16-2 teacher-student pair. We evaluate several values of β, namely $5, 10, 20, 50, 75$, over the vanilla KD method. The results are presented in Table 7.10. As shown, the student consistently benefits from distillation with the DJIP teacher, and the improvement is especially notable in more challenging low-data regimes (smaller β). These results further support the robustness of our method under constrained data scenarios.

7.5.11 Computational Efficiency and Convergence Behavior of DJIP

In this section, we analyze the practical training characteristics of the proposed DJIP framework from two complementary perspectives: computational efficiency and optimization behavior. We first examine the

Table 7.10: Comparison of student accuracy (%) under few-shot distillation settings on CIFAR-100. The teacher is trained on the full dataset, while only a β percent subset of samples per class is used to train the student.

Teacher	Student	Method	β				
			5	10	20	50	75
WRN-40-2	WRN-16-2	CE	34.75	50.73	60.93	69.60	72.17
		DJIP	40.56	53.95	62.98	70.87	73.15
		Δ	+5.81	+3.22	+2.06	+1.27	+0.98

online distillation complexity of DJIP by quantifying its training throughput, GPU memory footprint, and overall runtime across multiple teacher–student configurations. This analysis highlights the amortized cost of the offline JPEG-layer training and demonstrates that DJIP maintains competitive efficiency relative to standard knowledge distillation baselines.

We then study the convergence properties of the alternating optimization procedure underlying DJIP. While theoretical guarantees arise from the analytical solution of the inner minimization step, we further provide empirical evidence showing stable and monotonic training dynamics across large-scale models. Together, these results confirm that DJIP is both computationally practical and optimizes reliably in real-world training scenarios.

DJIP Online Distillation Complexity

As mentioned in Figure 7.1, DJIP contains 2 stages. The first stage trains the JPEG layer in an offline manner. Since the learned JPEG layer can be reused for training multiple student models, this stage is executed only once and its training time is amortized and therefore negligible in the overall framework.

For the second online stage, we present the student training throughput, GPU peak memory, and total runtime in Table 7.11. The student distillation stage is conducted on a single RTX 2080 Ti GPU.

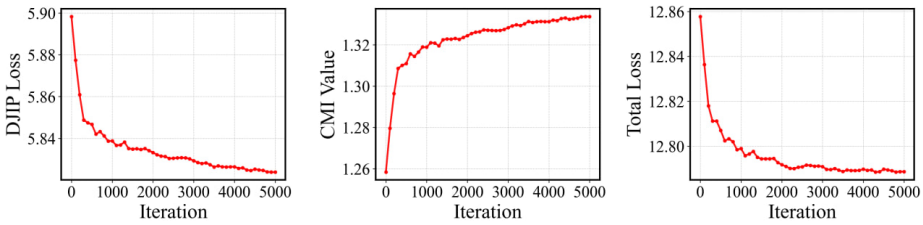
Convergence Analysis and Empirical Evidence

The inner minimization step in Equation □ satisfies the cross-entropy inequality. Suppose the impact of the random mini-batch sampling and SGD is ignored. In that case, the alternating algorithm is guaranteed to converge in theory, since given w, the optimal $Q^*(x \mid i, y)$ can be found analytically via 7.8. Although in practice the alternating algorithm may not converge to a global minimum, this is not the limitation of our algorithm, but the nature of all SGD-based deep learning algorithms.

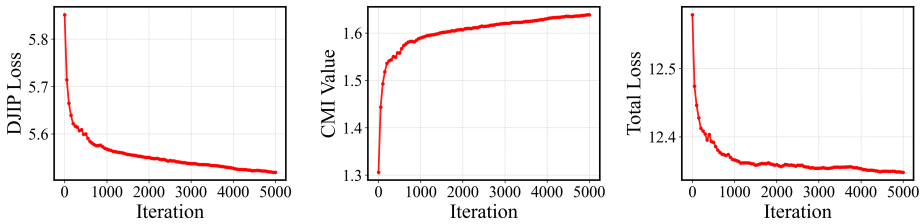
Empirically, our alternating procedure exhibits stable behavior throughout training. As shown in Figure 7.7, the training CMI value consistently improves, and the training DJIP loss and the overall training

Table 7.11: Comparison of DJIP distillation complexity measured with different teacher-student pairs on CIFAR-100 with vanilla KD method and TALD, following the configuration specified in Section 7.4.1.

Teacher Student	Method	Throughput (img/ms)	Peak GPU Memory (MB/GPU)	Total Runtime (s)
VGG-13	Vanilla	7.11	614	2026.6
	DJIP	5.60	624	2572.8
VGG-8	TALD	2.48	1060	5829.6
ResNet-56	Vanilla	4.61	408	3124.8
	DJIP	3.70	410	3895.2
ResNet-20	TALD	1.38	946	10447.2
WRN-40-2	Vanilla	3.53	668	4084.8
	DJIP	2.98	670	4826.4
WRN-40-1	TALD	1.15	1262	12516.0



(a) ResNet-34 ($\lambda = 1.4$)



(b) ResNet-152 ($\lambda = 2.0$)

Figure 7.7: Convergence curves of the proposed alternating optimization on ImageNet. The training DJIP loss, training CMI value, and total training loss exhibit stable and monotonic trends, confirming the empirical convergence of our method.

loss converge smoothly over iterations on both ResNet-34 and ResNet-152. These observations indicate

that the proposed alternating algorithm behaves stably in practice at scale.

7.5.12 Ablation Studies on DJIP Hyper-parameters

In this section, we conduct a series of ablation studies to investigate the sensitivity of hyper-parameters and how those key hyper-parameters affect the behavior of the DJIP teacher and the resulting student performance. Specifically, we examine the influence of the weighting coefficient λ, the number of quantization rounds in JPEG, the learning rate for JPEG layer's updates, and the cluster centroid update interval.

Effect of λ in DJIP

In Table 7.12, we examine the influence of the hyper-parameter λ on the optimization of the objective function defined in Equation 7.10. To this end, we train the DJIP teacher with various values of $\lambda \in \{0.4, 1.0, 1.4, 2.0, 2.4, 3.0\}$. As anticipated, increasing λ leads to a higher CMI of the teacher while simultaneously increasing its CE loss as well. Moreover, the student's Top-1 accuracy exhibits a quasiconcave trend with respect to λ, achieving its peak at $\lambda = 1.4$. This suggests that $\lambda = 1.4$ yields a favorable balance between the teacher's informativeness (CMI) and its predictive correctness (CE), thereby facilitating a higher student performance during distillation.

Table 7.12: Effect of λ on student test accuracy (%) and the CMI/CE losses of the DJIP teacher. The results are compared with those of a standard CE teacher in the ResNet-34 $\rightarrow$ ResNet-18 setting on ImageNet with vanilla KD. **Bold** numbers indicate the best performance.

DJIP teacher Vs. CE teacher on ImageNet (ResNet-34 $\rightarrow$ ResNet-18)								
KD	CE	DJIP with λ						Δ
		0.4	1.0	1.4	2.0	2.4	3.0	
Teacher CELoss	0.560	0.568	0.568	0.581	0.630	0.595	0.627	/
Teacher CMI	0.7180	0.7257	0.7268	0.7382	0.7869	0.7523	0.7825	/
Student Acc	70.660	71.362	71.452	**71.654**	71.274	71.370	71.290	+0.702

Effect of Number of Rounds in DJIP

We investigate the effect of increasing the number of quantization rounds in JPEG. Here, one round refers to applying quantization to the DCT coefficients once using a separate quantization table. Introducing multiple rounds effectively expands the perturbation space, which is particularly beneficial for complex datasets such as ImageNet. As the number of rounds increases, the teacher's CMI consistently improves;

however, excessive rounds can lead to elevated CE loss, indicating a loss of predictive reliability. As shown in Table 7.13, the student's accuracy grows steadily from 1 to 5 rounds and reaches its peak at 5 rounds, suggesting that a moderate degree of iterative quantization provides the most favorable balance between informativeness and stability.

Table 7.13: Effect of the number of quantization rounds on student accuracy (%) and on the CMI and CE losses of the DJIP teacher. Results are reported for the ResNet-34 $\rightarrow$ ResNet-18 setting on ImageNet under vanilla KD. **Bold** numbers denote the best performance.

KD	CE	DJIP with number of rounds equals to			
		1	3	5	6
Teacher CELoss	0.560	0.560	0.561	0.582	0.768
Teacher CMI	0.7180	0.7178	0.7185	0.7382	0.9097
Student Acc	70.660	71.278	71.610	**71.654**	70.960

Effect of Learning Rate in DJIP

We evaluate teacher learning rates of 0.1, 0.02, 0.01, and 0.001. As shown in Table 7.14, an overly high learning rate leads to unstable JPEG layer updates, manifested by excessively large CMI values and noticeably increased CE loss, which ultimately undermines the student's performance. The learning rate of 0.01 yields the best overall results, indicating that a stable yet sufficiently responsive update regime is essential for effective distillation.

Table 7.14: Effect of learning rate on student accuracy (%), along with the CMI and CE losses of the DJIP teacher. Results are reported for the ResNet-34 $\rightarrow$ ResNet-18 setting on ImageNet under vanilla KD. **Bold** indicates the best performance.

KD	CE	DJIP with learning rate equals to			
		0.1	0.02	0.01	0.001
Teacher CELoss	0.560	0.875	0.601	0.582	0.561
Teacher CMI	0.7180	0.9961	0.7577	0.7382	0.7191
Student Acc	70.660	70.628	71.444	**71.654**	71.432

Effect of Centroid Update Interval in DJIP

We further study the impact of the centroid update interval of DJIP in Table 7.15. When using intervals of 3, 5, 50, and 500 iterations, we observe that the student accuracy remains largely stable, with the best results appearing at intervals of 3 and 5. Although both intervals yield similar accuracy, we select the

interval of 5 due to efficiency concerns. As the update interval becomes longer, the student's accuracy shows a mild decrease; however, it consistently surpasses that obtained with the standard CE teacher. This suggests that moderately frequent centroid updates provide more reliable centroid estimates while avoiding unnecessary computation, thereby enabling a more accurate inner minimization of Equation □ and ultimately improving distillation performance.

Table 7.15: Effect of centroid update interval on student accuracy (%) on CIFAR-100 (averaged over 3 runs). Results are reported for the ResNet-34 → ResNet-18 setting, with update intervals of 3, 5, 50, and 500 iterations. All configurations consistently outperform the standard CE teacher, with the best performance achieved at an interval of 3 and 5.

Teacher	Student	Method	CMI	KD	DKD	DIST	CC	RKD	AT	FitNet	FT	SP	ITRD	CRD
ResNet-50	VGG-8	CE	0.009	73.81	73.94	74.11	70.25	71.50	71.84	69.39	70.29	73.34	75.49	74.30
		Interval=3	0.341	74.49	75.87	74.58	70.92	71.87	72.50	69.50	71.29	73.58	76.02	74.56
		Interval=5	0.341	74.48	75.87	74.80	70.92	71.93	72.42	69.57	71.24	73.87	75.91	74.55
		Interval=50	0.341	74.33	75.58	74.63	70.78	71.81	72.38	69.35	71.23	73.68	75.91	74.24
		Interval=500	0.342	74.32	75.49	74.52	70.68	71.73	72.37	69.02	71.06	73.61	75.81	74.21

7.6 Chapter Summary

In this chapter, we have introduced Differentiable JPEG-based Input Perturbation (DJIP), a novel framework that enhances the transferability of a fixed teacher by incorporating a differentiable JPEG compression layer. By jointly optimizing CE loss and CMI values through a tailored alternating algorithm, our method enables the teacher to produce more informative supervision signals without modifying its original weights. Extensive experiments on ImageNet and CIFAR-100, across both CNNs and ViTs architectures, demonstrate that DJIP consistently improves student accuracy (up to 4.11% over standard baselines). Comprehensive analyses further confirm the effectiveness and stability of our alternating CMI optimization. Beyond empirical gains, DJIP offers a lightweight and generalizable mechanism for enhancing knowledge distillation by exploiting input-space perturbations using trainable quantization. Since the teacher remains unchanged, DJIP is particularly suitable for scenarios with strict deployment or integrity constraints.

Chapter 8

Conclusion and Future Work

This chapter concludes the dissertation by summarizing the research journey, restating the principal contributions, and outlining promising directions for future inquiry.

8.1 Summary

This dissertation was motivated by the observation that the world's dominant image format, JPEG, and the leading paradigm for image understanding, Deep Neural Networks (DNNs), were designed with distinct objectives. JPEG was engineered for the Human Visual System (HVS), while DNNs have their own complex criteria for optimal performance. This dissertation investigates how these two foundational technologies can be more effectively integrated, systematically demonstrating that their initial sub-optimal interaction is not a limitation but an opportunity. We have shown that the architectural principles of JPEG—specifically, frequency-domain quantization—can be repurposed to synergistically integrate with and enhance DNN performance. This research charts a logical progression from adapting the JPEG standard externally for DNN consumption, to integrating it internally as a learnable network component, and finally, to repurposing its core principles to amplify knowledge transfer.

Our first contribution revisited the relationship between JPEG's default parameters and a given DNN by reframing the compression objective from a human-centric "rate-distortion" problem to a machine-centric "rate-accuracy" one. To this end, we introduced the **Sensitivity Weighted Error (SWE)**, a novel distortion measure derived from a DNN's loss sensitivity to frequency-domain perturbations. This measure guides our **OptS** algorithm to generate model-specific, JPEG-compliant quantization tables that demonstrably improve the rate-accuracy trade-off for pre-trained models [101, 144].

Building on this insight, the second part of the thesis investigated whether the encoding process itself could become an intrinsic, learnable component of a neural network. This led to the creation of **JPEG-Inspired Deep Learning (JPEG-DL)**. The key technical barrier—the non-differentiable hard quantiza-

tion step of JPEG—was overcome by an end-to-end differentiable JPEG layer. This innovation served a dual purpose: it not only rendered the compression pipeline trainable but also acted as a new form of trainable non-linearity, enhancing the model's expressive power. By prepending a lightweight, trainable JPEG layer to any standard DNN, we transformed the JPEG pipeline from a static pre-processor into a dynamic, task-aware feature transformation layer, leading to significant gains in both classification accuracy and adversarial robustness.

The final part of this dissertation synthesized these concepts to improve knowledge transfer in Knowledge Distillation (KD). To address the "overconfident teacher" problem, we developed a series of frameworks that use input perturbations to elicit softer, more informative supervisory signals. This began with the heuristic **Coded Knowledge Distillation (CKD)**, which adaptively selects JPEG quality factors to modulate teacher outputs. We then established a theoretical foundation with **Generalized Coded Knowledge Distillation (GCKD)**, an information-theoretic framework that frames the problem as maximizing the teacher's **Conditional Mutual Information (CMI)**. Our work culminated in **Differentiable JPEG-based Input Perturbation (DJIP)**, an elegant architectural synthesis that ingeniously resolves the computational complexity of GCKD. By deploying the trainable JPEG layer from Chapter 4 as a practical mechanism to achieve the theoretical CMI maximization goal, DJIP creates a highly efficient, plug-and-play "coded teacher" that unifies all preceding concepts into a single, powerful solution.

Throughout this dissertation, our theoretical proposals were substantiated by extensive experiments. The results consistently demonstrate that by reformulating JPEG's encoding philosophy for the machine, we provide a unified approach to improving model performance, architectural efficiency, and the efficacy of knowledge transfer.

8.2 Thesis Contributions

The research presented in this dissertation establishes JPEG-inspired encoding as a novel and powerful framework for enhancing deep learning models. The primary contributions are:

1. **A JPEG-Compliant Compression Framework for DNN Vision.** We developed a principled method for adapting the standard JPEG codec for machine consumption without altering its decoder compatibility.
    - We proposed a novel, machine-centric distortion measure, **Sensitivity Weighted Error (SWE)**, based on a DNN's training loss sensitivity to DCT perturbations [101, 144].
    - We developed **OptS**, an algorithm that leverages SWE to design optimal JPEG quantization tables that demonstrably improve the rate-accuracy performance for any given DNN.
2. **An Integrated and Trainable Compression Layer for DNN Architectures.** We designed a novel, end-to-end trainable JPEG layer as the core of our JPEG-Inspired Deep Learning (JPEG-DL) framework, seamlessly integrating the compression pipeline directly into the network.

- We introduced a novel **differentiable soft quantizer**, a core component that enables end-to-end training of the layer via backpropagation while also serving as a trainable non-linearity to enhance the model's expressive power.
- We demonstrated that prepending this lightweight layer to a standard DNN and jointly optimizing its parameters with the network's weights achieves significant and consistent improvements in model accuracy and adversarial robustness [99].

3. **A Suite of Frameworks for Enhancing Knowledge Distillation.** We pioneered and formalized the use of JPEG-inspired input perturbation to improve the supervisory signal from a fixed teacher model in KD.
    - We introduced **Coded Knowledge Distillation (CKD)**, a practical and efficient framework that uses adaptive JPEG compression to elicit softer, more informative outputs from an overconfident teacher [98].
    - We generalized this concept with **Generalized Coded Knowledge Distillation (GCKD)**, an information-theoretic framework that defines the problem as maximizing the teacher's Conditional Mutual Information (CMI) via input perturbations [100].
    - We synthesized our prior work in **Differentiable JPEG-based Input Perturbation (DJIP)**, an efficient architectural solution that operationalizes the CMI maximization goal of GCKD by using the trainable JPEG layer, creating a lightweight and practical "coded teacher" [15].

8.3 Directions for Future Work

The research presented in this dissertation opens up several promising avenues for future investigation. The frameworks and concepts developed herein can be extended and refined in the following directions:

1. **Per-Coefficient Sensitivity and Non-JPEG-Compliant Compression.** The Sensitivity Weighted Error (SWE) framework was intentionally designed for JPEG compliance by using a single sensitivity weight, s_i, for each frequency across all spatial blocks. A promising future direction is to relax this constraint and explore a more granular, non-JPEG-compliant approach using per-coefficient sensitivities, $s_{i,j}$. This would enable a more precise distortion allocation but introduces significant research challenges. It would necessitate not only the design of a new compression codec but also a fundamental extension of rate-distortion theory to accommodate time-varying fidelity criteria, as the foundational Shannon lower bound would no longer apply. Overcoming these theoretical hurdles could lead to compression schemes with even greater rate-accuracy performance.

2. **Addressing Large Perturbations in DNN-Oriented Compression.** The formulation of Sensitivity Weighted Error (SWE) relies on a first-order Taylor expansion, which is most effective for small perturbations. This approximation can become less accurate for large quantization steps that induce significant changes in the DCT coefficients. A valuable direction for future research is to

develop DOC methods that can effectively handle large perturbations, perhaps by incorporating higher-order terms of the Taylor expansion or exploring alternative non-local analysis techniques. One primary solution is instead of training the model on the original on training set, we can compress the training set and train a model to accommodate the large distortion. However, it still not favourable to use different weight for different regions.

3. **Explicit Rate-Accuracy Optimization for JPEG-DL.** The current JPEG-DL framework successfully enhances model accuracy by jointly learning quantization parameters and network weights, but it does so without an explicit objective for the resulting compression rate. The rate is an uncontrolled byproduct of the optimization, not a target. A significant future direction is to explicitly optimize the full rate-accuracy trade-off. The primary challenge lies in the non-differentiable nature of the entropy coding stage (e.g., Huffman coding), which makes the true bit rate intractable for gradient-based optimization. A promising path forward would be to develop a differentiable, information-theoretic rate estimator. This could potentially be achieved by leveraging the probabilistic nature of the soft quantizer itself. Since the soft quantizer defines a probability distribution over the quantized coefficients, the Shannon entropy of this distribution serves as a well-established proxy for the optimal coding rate. Developing a method to analytically compute this entropy and integrate it into the loss function would enable end-to-end training that explicitly minimizes a Lagrangian of inference loss and bit rate, thereby allowing for precise control over the rate-accuracy Pareto frontier.

4. **From Input Compression to Compressible Activations for Distributed Systems.** The current JPEG-DL framework successfully enhances model accuracy by jointly learning quantization parameters and network weights, but it does so without an explicit objective for the resulting compression rate. A significant future direction is to explicitly optimize this rate-accuracy trade-off. For the input image, this can be achieved by leveraging the probabilistic nature of the soft quantizer; its inherent conditional probability mass function (CPMF) allows the Shannon entropy to serve as a differentiable proxy for the bit rate, enabling end-to-end training that minimizes a Lagrangian of inference loss and bit rate. However, a far more transformative extension would be to apply this principle not to the input image, but to the network's internal activations. A fundamental bottleneck in training and deploying massive models, such as LLMs, is the communication cost of transferring high-precision activation tensors between distributed compute nodes. One could re-envision the trainable soft quantizer, decoupled from the JPEG pipeline, as a novel form of activation function. By replacing standard functions like ReLU or GELU with this probabilistic quantization mechanism, and including the entropy of the activations as a regularization term in the main loss function, it may be possible to train a model that learns to compress its own internal representations.

5. **Theoretical Analysis of Adversarial Robustness in JPEG-DL.** The JPEG-DL framework was empirically shown to confer significant improvements in adversarial robustness. A compelling avenue for future work is to conduct a deeper theoretical analysis to understand the precise mechanisms behind this enhancement. Investigating how the trainable JPEG layer alters the network's loss landscape, input gradients, and decision boundaries could provide fundamental insights into

building models that are inherently more robust to adversarial attacks.

6. **Generalizing the JPEG Layer as a Novel Architectural Component.** The JPEG-DL framework establishes that a trainable JPEG layer, centered on our differentiable soft quantizer, functions as a novel, structured non-linear operator. This opens a broader research direction into designing and integrating other forms of compression-inspired non-linearities as architectural components. One could explore replacing standard activation functions like ReLU in deeper layers of a network with similar trainable quantizers operating on feature maps, potentially leading to more efficient, expressive, and interpretable models.

7. **Generative Models for Coded Dataset Creation.** The GCKD and DJIP frameworks currently use gradient-based or JPEG-based methods to generate input perturbations. Future research could explore the use of more sophisticated generative models, such as Generative Adversarial Networks (GANs) or diffusion models, to create the "coded dataset." Training a generator to produce CMI-maximizing perturbations could yield a more diverse and globally optimal set of inputs, potentially leading to further improvements in the quality of the teacher's supervisory signal and the final student performance.

8. **Extension of the Coded Teacher Paradigm to Large Language Models (LLMs).** The fundamental principle of the coded teacher"—using input perturbations to maximize the CMI of a fixed teacher—is not limited to the vision domain and presents a compelling opportunity in Natural Language Processing. Massive foundation models like GPT-4 are often deployed as static, highly optimized services using inference engines like vLLM to handle many concurrent prompts efficiently. In such an environment, fine-tuning the teacher or deploying a separate, modified version solely for distillation is operationally infeasible and prohibitively expensive, as it would require maintaining multiple instances of a multi-billion parameter model. Our "coded teacher" paradigm offers a solution that respects these real-world constraints. Future work could develop a **learnable text transformation model**, analogous to how DJIP uses a JPEG layer for images. This model would be trained to apply subtle, meaning-preserving modifications to input prompts (e.g., through paraphrasing or token manipulation) with the explicit goal of maximizing the CMI of the large, fixed teacher LLM. Such a framework could significantly enhance knowledge distillation for LLMs, making it possible to create smaller, more efficient student models without ever needing to alter the deployed foundation model. This addresses a critical challenge in making powerful language models more accessible and widely deployable.

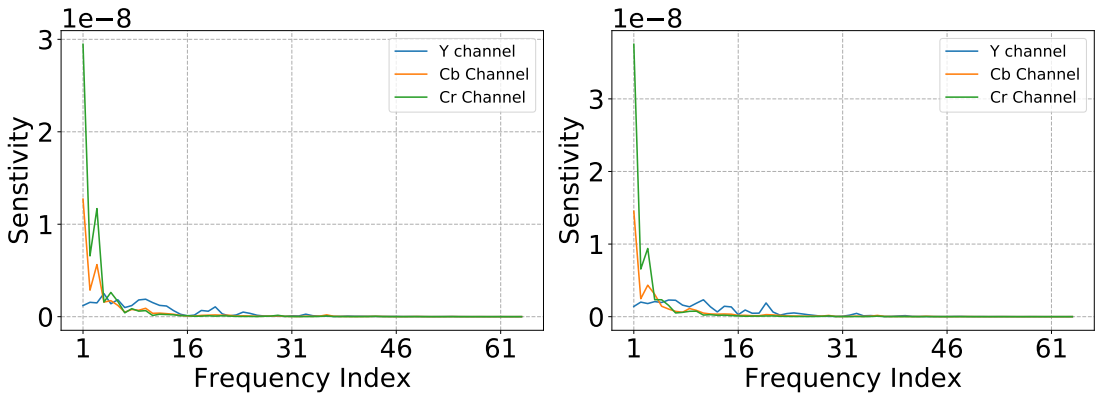
8.4 Discussion

8.4.1 Dataset-Dependent Frequency Sensitivity

Figures 8.1–8.2 compare the DCT-domain sensitivities $\{s_i\}$ estimated from cross-entropy loss gradients for the *same task family* (image classification) under *two markedly different data regimes*: (i) a large, het-

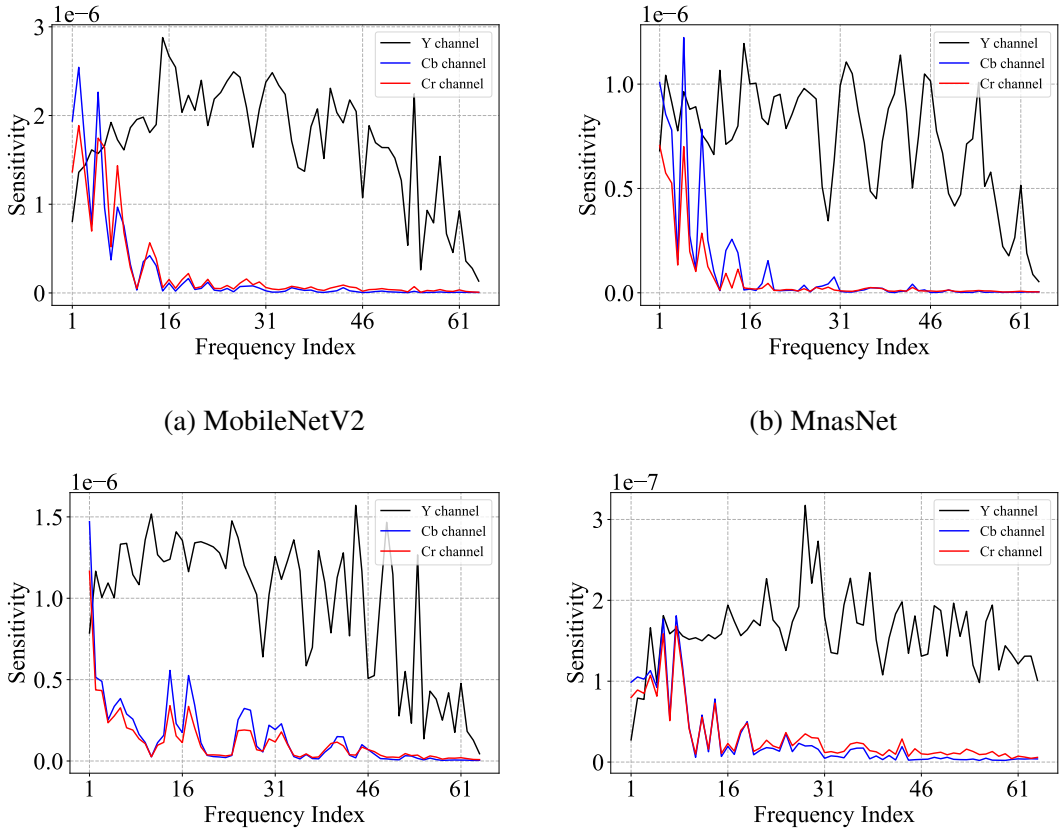
erogeneous dataset with broad intra-class variability and diverse backgrounds (ImageNet), and (ii) a fine-grained dataset focused on subtle inter-class distinctions among closely related categories (CUB200). A salient observation is a pronounced discrepancy in the *mid-frequency band* of the luminance (Y) channel: ImageNet models maintain non-negligible sensitivity across mid frequencies (with model-dependent oscillations), whereas CUB200 models concentrate sharply at very low frequencies with a rapid decay—mid-band sensitivities are comparatively suppressed.

Interpretability objective. The primary motivation for a reader-led follow-up is *interpretability*: to determine which visual features are more diagnostic for the *machine* than for the *human* when solving the *same* recognition task. The dataset-dependent sensitivity gap we observe (CUB200 vs. ImageNet) provides a natural testbed to ask whether models lean more on edges/part contours, broader mid-band textures, or other frequency-localized cues than humans do. Rather than establishing causality, the goal is to compare human performance and model behavior on matched stimuli and report where they diverge—thereby turning our machine-centric compression analysis into an opportunity to *explain* model reliance on specific frequency bands and structures, and to surface feature–reliance discrepancies that can guide future, more interpretable encoder designs.



(a) ResNet18 (b) DenseNet-121

Figure 8.1: The estimated sensitivity is shown for two pre-trained models on the CUB200, with sensitivity indices arranged in the default zigzag order.



(a) MobileNetV2

(b) MnasNet

(c) RegNetY-400MF

(d) ConvNeXt-T

Figure 8.2: The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, with sensitivity indices arranged in the default zigzag order.

References

[1] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9163–9171, 2019.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning, 2023.

[3] Hossam Amer, Ahmed H Salamah, Ahmad Sajedi, and En-Hui Yang. High performance convolution using sparsity and patterns for inference in deep convolutional neural networks. *arXiv preprint arXiv:2104.08314*, 2021.

[4] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[5] Tajeddine Benbarrad, Salaheddine Kably, Mounir Arioua, and Nabih Alaoui. Compression-based data augmentation for cnn generalization. In *International Conference on Cybersecurity, Cybercrimes, and Smart Emerging Technologies*, pages 235–244. Springer, 2022.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[7] Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10925–10934, 2022.

[8] Neelanjan Bhowmik, Jack W Barker, Yona Falinie A Gaus, and Toby P Breckon. Lost in compression: the impact of lossy image compression on variable size object detection within infrared imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 369–378, 2022.

[9] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data*

Mining, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery.

[10] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision*, pages 839–847. IEEE, 2018.

[11] Chyi-Tsong Chen and Wei-Der Chang. A feedforward neural network with function shape autotuning. *Neural networks*, 9(4):627–641, 1996.

[12] Defang Chen, Jian-Ping Mei, Can Wang, Yan Feng, and Chun Chen. Online knowledge distillation with diverse peers, 2019.

[13] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[14] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5008–5017, June 2021.

[15] Siyu Chen, Kaixiang Zheng, Ahmed H. Salamah, and En-Hui Yang. Differentiable JPEG-based input perturbation for knowledge distillation amplification via conditional mutual information maximization. In *Proceedings of the International Conference on Learning Representations*, 2026.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[17] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4794–4802, 2019.

[18] Jinyoung Choi and Bohyung Han. Task-aware quantization network for jpeg image compression. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 309–324. Springer, 2020.

[19] Linhang Cai Chuanguang Yang, Zhulin An and Yongjun Xu. Hierarchical self-supervised augmented knowledge distillation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1217–1223, 2021.

[20] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[21] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[22] Tri Dao, Govinda M Kamath, Vasilis Syrgkanis, and Lester Mackey. Knowledge distillation as semiparametric inference, 2021.

[23] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *ArXiv*, abs/1705.02900, 2017.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[25] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2016.

[26] Chengyu Dong, Liyuan Liu, and Jingbo Shang. Toward student-oriented teacher network training for knowledge distillation, 2024.

[27] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. A study of the effect of JPG compression on adversarial images. *CoRR*, abs/1608.00853, 2016.

[28] Max Ehrlich, Larry Davis, Ser-Nam Lim, and Abhinav Shrivastava. Analyzing and mitigating jpeg compression defects in deep learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2357–2367, 2021.

[29] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[30] Gongfan Fang. Deeplabv3plus-pytorch. https://github.com/VainF/DeepLabV3Plus-Pytorch, 2019.

[31] Sanjukta Ghosh, Rohan Shet, Peter Amon, Andreas Hutter, and André Kaup. Robustness of deep convolutional neural networks for image degradations. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2916–2920, 2018.

[32] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[35] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[36] Klemen Grm, Vitomir Štruc, Anais Artiges, Matthieu Caron, and Hazım K Ekenel. Strengths and weaknesses of deep learning models for face recognition against image degradations. *IET Biometrics*, 7(1):81–89, 2017.

[37] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[38] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[39] Zhiwei Hao, Jianyuan Guo, Kai Han, Yehui Tang, Han Hu, Yunhe Wang, and Chang Xu. One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation, 2023.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[42] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3771–3778, 2019.

[43] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3779–3787, 2019.

[44] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[45] Chen-Hsiu Huang and Ja-Ling Wu. Unveiling the future of human and machine coding: A survey of end-to-end learned image compression. *Entropy*, 26(5), 2024.

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[47] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. *arXiv preprint arXiv:2205.10536*, 2022.

[48] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[49] Forrest N Iandola. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[50] Seonghye Jeong, Seongmoon Jeong, Simon S. Woo, and Jong Hwan Ko. An overhead-free region-based jpeg framework for task-driven image compression. *Pattern Recognition Letters*, 165:1–8, 2023.

[51] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2. Citeseer, 2011.

[52] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. *Advances in neural information processing systems*, 31, 2018.

[53] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[54] Klemen Kotar, Gabriel Ilharco, Ludwig Schmidt, Kiana Ehsani, and Roozbeh Mottaghi. Contrasting contrastive self-supervised representation learning pipelines. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9949–9959, October 2021.

[55] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[57] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble, 2018.

[58] Hyoje Lee, Yeachan Park, Hyun Seo, and Myungjoo Kang. Self-knowledge distillation via dropout. *Computer Vision and Image Understanding*, 2023.

[59] Sungyong Lee, Wooseung Nam, Jinsung Lee, Sangtae Ha, and Kyunghan Lee. N-epitomizer: Enabling semantic offloading for neural network inferences. In *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 71–79. IEEE, 2023.

[60] Hongshan Li, Yu Guo, Zhi Wang, Shutao Xia, and Wenwu Zhu. Adacompress: Adaptive compression for online computer vision services. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2440–2448, 2019.

[61] Kehan Li, Runyi Yu, Zhennan Wang, Li Yuan, Guoli Song, and Jie Chen. Locality guidance for improving vision transformers on tiny datasets, 2022.

[62] Wei Li, Shitong Shao, Weiyan Liu, Ziming Qiu, Zhihao Zhu, and Wei Huan. What role does data augmentation play in knowledge distillation? In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 2204–2220, December 2022.

[63] Yanyu Li, Geng Yuan, Yang Wen, Eric Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *arXiv preprint arXiv:2206.01191*, 2022.

[64] Zheng Li, Ying Huang, Defang Chen, Tianren Luo, Ning Cai, and Zhigeng Pan. Online knowledge distillation via multi-branch diversity enhancement, 2020.

[65] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024.

[66] Yufan Liu, Jiajiong Cao, Bing Li, Chunfeng Yuan, Weiming Hu, Yangxi Li, and Yunqiang Duan. Knowledge distillation via instance relationship graph. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7089–7097, 2019.

[67] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[68] Zihao Liu, Tao Liu, Wujie Wen, Lei Jiang, Jie Xu, Yanzhi Wang, and Gang Quan. DeepN-JPEG: A deep neural network favorable JPEG-based image compression framework. In *Proceedings of the 55th Annual Design Automation Conference*, page 18. ACM, 2018.

[69] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[70] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[71] Xiyang Luo, Hossein Talebi, Feng Yang, Michael Elad, and Peyman Milanfar. The rate-distortion-accuracy tradeoff: Jpeg case study. *arXiv preprint arXiv:2008.00605*, 2020.

[72] Xu Luo, Hao Wu, Ji Zhang, Lianli Gao, Jing Xu, and Jingkuan Song. A closer look at few-shot classification again, 2023.

[73] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[74] Aleksander Madry. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[75] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery.

[76] Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, Seungyeon Kim, and Sanjiv Kumar. A statistical perspective on distillation. In *International Conference on Machine Learning*, pages 7632–7642. PMLR, 2021.

[77] Roy Miles, Adrian Lopez-Rodriguez, and Krystian Mikolajczyk. Information theoretic representation distillation. In *British Machine Vision Conference (BMVC)*, 2022.

[78] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant, 2019.

[79] Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-distillation amplifies regularization in hilbert space, 2020.

[80] Shayan Mohajer Hamidi. Training neural networks on remote edge devices for unseen class classification. *IEEE Signal Processing Letters*, 31:1004–1008, 2024.

[81] Shayan Mohajer Hamidi, Deng Xizhen, Renhao Tan, Linfeng Ye, and Ahmed H. Salamah. How to train the teacher model for effective knowledge distillation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[82] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[83] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[84] Thanh Nguyen-Duc, Trung Le, He Zhao, Jianfei Cai, and Dinh Phung. Adversarial local distribution regularization for knowledge distillation. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4670–4679, 2023.

[85] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[86] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[87] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[88] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Probabilistic knowledge transfer for lightweight deep representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2030–2039, 2020.

[89] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[90] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5007–5016, 2019.

[91] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[92] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[93] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International conference on machine learning*, pages 5142–5151. PMLR, 2019.

[94] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Semantic perceptual image compression using deep convolution networks. In *2017 Data Compression Conference (DCC)*, pages 250–259. IEEE, 2017.

[95] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020.

[96] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[97] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[98] Ahmed H Salamah, Shayan Mohajer Hamidi, and En-Hui Yang. A coded knowledge distillation framework for image classification based on adaptive jpeg encoding. *Pattern Recognition*, page 110966, 2024.

[99] Ahmed H. Salamah, Kaixiang Zheng, Yiwen Liu, and En-Hui Yang. Jpeg inspired deep learning. 2025.

[100] Ahmed H. Salamah, Kaixiang Zheng, and En-Hui Yang. Generalized coded knowledge distillation via conditional mutual information maximization. *Transactions on Machine Learning Research*, 2025.

[101] Ahmed H Salamah, Kaixiang Zheng, Linfeng Ye, and En-Hui Yang. Jpeg compliant compression for dnn vision. *IEEE Journal on Selected Areas in Information Theory*, 2024.

[102] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[103] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NIPS 2017 Workshop on Machine Learning and Computer Security*, volume 1, page 8, 2017.

[104] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[105] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19, 2018.

[106] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34, 2021.

[107] Yannick Strümpler, Ren Yang, and Radu Timofte. Learning to improve image compression without changing the standard decoder. In *European Conference on Computer Vision*, pages 200–216. Springer, 2020.

[108] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models, 2024.

[109] Shangquan Sun, Wenqi Ren, Jingzhi Li, Rui Wang, and Xiaochun Cao. Logit standardization in knowledge distillation, 2024.

[110] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[111] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[112] Chao Tan and Jie Liu. Improving knowledge distillation with a customized teacher. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2290–2299, 2024.

[113] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[114] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

[115] Robert Torfason, Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Towards image understanding from deep compression without decoding. *arXiv preprint arXiv:1803.06131*, 2018.

[116] Zhengzhong Tu, Peyman Milanfar, and Hossein Talebi. Muller: Multilayer laplacian resizer for vision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6877–6887, 2023.

[117] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1365–1374, 2019.

[118] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[119] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[120] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[121] Chaofei Wang, Qisen Yang, Rui Huang, Shiji Song, and Gao Huang. Efficient knowledge distillation from model checkpoints. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 607–619. Curran Associates, Inc., 2022.

[122] Huan Wang, Suhas Lohit, Michael N Jones, and Yun Fu. What makes a" good" data augmentation in knowledge distillation-a statistical perspective. *Advances in Neural Information Processing Systems*, 35:13456–13469, 2022.

[123] Zhuojie Wu, Heming Du, Shuyun Wang, Ming Lu, Haiyang Sun, Yandong Guo, and Xin Yu. Cmamba: Learned image compression with state space models, 2025.

[124] Xuedou Xiao, Juecheng Zhang, Wei Wang, Jianhua He, and Qian Zhang. Dnn-driven compressive offloading for edge-assisted semantic video segmentation. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1888–1897. IEEE, 2022.

[125] Xiufeng Xie and Kyu-Han Kim. Source compression with bounded dnn perception loss for iot edge computer vision. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[126] Xiufeng Xie, Ning Zhou, Wentao Zhu, and Ji Liu. Bandwidth-aware adaptive codec for dnn inference offloading in iot. In *European Conference on Computer Vision*, pages 88–104. Springer, 2022.

[127] Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. Initializing models with larger ones. In *The Twelfth International Conference on Learning Representations*, 2023.

[128] Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5628–5635, 2019.

[129] Chuanguang Yang, Xinqiang Yu, Zhulin An, and Yongjun Xu. Categories of response-based, feature-based, and relation-based knowledge distillation, 2023.

[130] Chuanguang Yang, Helong Zhou, Zhulin An, Xue Jiang, Yongjun Xu, and Qian Zhang. Cross-image relational knowledge distillation for semantic segmentation, 2022.

[131] En-Hui Yang, Hossam Amer, and Yanbing Jiang. Compression helps deep learning in image classification. *Entropy*, 23(7), 2021.

[132] En-Hui Yang and Shayan Mohajer Hamidi. Systems and methods for training deep learning models. 2024.

[133] En-Hui Yang, Shayan Mohajer Hamidi, Linfeng Ye, Renhao Tan, and Beverly Yang. Conditional mutual information constrained deep learning for classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[134] En-Hui Yang, Chang Sun, and Jin Meng. Quantization table design revisited for image/video coding. *IEEE Transactions on Image Processing*, 23(11):4799–4811, 2014.

[135] En-Hui Yang and Longji Wang. Joint optimization of run-length coding, huffman coding, and quantization table with complete baseline jpeg decoder compatibility. *IEEE Transactions on Image Processing*, 2008.

[136] En-Hui Yang and Zhen Zhang. Variable-rate trellis source encoding. *IEEE Transactions on Information Theory*, 45(2):586–608, 1999.

[137] Linfeng Ye, Shayan Mohajer Hamidi, Renhao Tan, and En-Hui Yang. Bayes conditional distribution estimation for knowledge distillation based on conditional mutual information. In *The Twelfth International Conference on Learning Representations*, 2024.

[138] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4133–4141, 2017.

[139] Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13876–13885, 2020.

[140] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Fifth International Conference on Learning Representations*, 2017.

[141] Haoran Zhang, Zhenzhen Hu, Wei Qin, Mingliang Xu, and Meng Wang. Adversarial co-distillation learning for image recognition. *Pattern Recognition*, 111:107659, 2021.

[142] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[143] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11953–11962, 2022.

[144] Kaixiang Zheng, Ahmed H. Salamah, Linfeng Ye, and En-Hui Yang. Jpeg compliant compression for dnn vision. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 1875–1879, 2023.

[145] Kaixiang Zheng and En-Hui Yang. Knowledge distillation based on transformed teacher matching. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024.

[146] Helong Zhou, Liangchen Song, Jiajie Chen, Ye Zhou, Guoli Wang, Junsong Yuan, and Qian Zhang. Rethinking soft labels for knowledge distillation: A bias–variance tradeoff perspective. In *International Conference on Learning Representations*, 2021.

[147] Xiatian Zhu, Shaogang Gong, et al. Knowledge distillation by on-the-fly native ensemble. *Advances in neural information processing systems*, 31, 2018.

APPENDICES

Appendix A

Derivation of the KKT Conditions for Optimal Distortion Allocation

This appendix outlines the detailed derivation of the Karush-Kuhn-Tucker (KKT) conditions for the rate-distortion optimization problem central to the DNN-Oriented Compression (DOC) framework, as introduced in Chapter 3. The objective is to determine the optimal allocation of mean-squared error (MSE) distortion to each frequency component to minimize the total rate, subject to a budget on the total *Sensitivity Weighted Error* (SWE).

A.1 The Optimization Problem

The objective is to select the distortion allocation, $D_1, D_2, \ldots, D_{2M}$, for each frequency component to minimize the overall file size, subject to a total weighted distortion budget and physical constraints on the distortion for each component specified in Section 3.5.2.

A.1.1 Objective Function

Minimize the total rate, which is the sum of the rates for each component:

$$\min_{D_1,\ldots,D_{2M}} \sum_{i=1}^{2M} R_i(D_i) \tag{A.1}$$

where D_i is the MSE distortion for component i. The rate-distortion function $R_i(D_i)$ is approximated by the Shannon Lower Bound, which relates the rate to the component's variance σ_i^2 and its allocated MSE

distortion D_i:

$$R_i(D_i) \approx \frac{1}{2} \log_2 \left(\frac{\sigma_i^2}{D_i} \right) \tag{A.2}$$

A.1.2 Constraints

1. **Budget Constraint:** The sum of the *sensitivity-weighted* distortions must equal a fixed total budget, D_T^{SWE}. This aligns with the SWE distortion measure defined in the main text.

$$\sum_{i=1}^{2M} s_i D_i = D_T^{\text{SWE}} \tag{A.3}$$

   Here, s_i is the pre-calculated sensitivity of the DNN to perturbations in frequency component i.

2. **Physical Constraints:** For each frequency component i, the allocated MSE distortion D_i must be physically meaningful. It cannot be less than the minimum possible distortion achievable by the quantizer, $D_{\min,i}$, and it cannot be greater than the component's signal power (variance), σ_i^2.

$$D_i \leq \sigma_i^2 \quad \forall i \in \{1, \ldots, 2M\} \tag{A.4}$$

$$D_i \geq D_{\min,i} \quad \forall i \in \{1, \ldots, 2M\} \tag{A.5}$$

   where $D_{\min,i}$ is shorthand for $D(C_i, q_i)$.

A.2 The Lagrangian Formulation

We combine the objective function and all constraints into the Lagrangian function $\mathcal{L}$, using Lagrange multipliers λ (for the equality budget constraint) and μ_i, ν_i (for the inequality physical constraints).

$$\begin{aligned} \mathcal{L}(D, \lambda, \mu, \nu) = \sum_{i=1}^{2M} R_i(D_i) &+ \lambda \left(\left(\sum_{i=1}^{2M} s_i D_i \right) - D_T^{\text{SWE}} \right) \\ &+ \sum_{i=1}^{2M} \mu_i (D_i - \sigma_i^2) \\ &+ \sum_{i=1}^{2M} \nu_i (D_{\min,i} - D_i) \end{aligned} \tag{A.6}$$

A.3 The KKT Conditions

For a candidate solution to be optimal, it must satisfy the following four Karush-Kuhn-Tucker (KKT) conditions.

1. **Stationarity:** The gradient of the Lagrangian with respect to the decision variables D_i must be zero.
$$\frac{\partial \mathcal{L}}{\partial D_i} = 0 \quad \forall i$$

2. **Primal Feasibility:** The solution must satisfy all original constraints.
$$\sum_{i=1}^{2M} s_i D_i = D_T^{\text{SWE}}$$
$$D_i - \sigma_i^2 \leq 0 \quad \forall i$$
$$D_{\min,i} - D_i \leq 0 \quad \forall i$$

3. **Dual Feasibility:** The multipliers for the inequality constraints must be non-negative.
$$\mu_i \geq 0 \quad \forall i$$
$$\nu_i \geq 0 \quad \forall i$$

4. **Complementary Slackness:** For each inequality constraint, the product of the multiplier and the constraint slack must be zero. This means either the constraint is active (holds with equality) or its corresponding multiplier is zero.
$$\mu_i(D_i - \sigma_i^2) = 0 \quad \forall i$$
$$\nu_i(D_{\min,i} - D_i) = 0 \quad \forall i$$

A.4 Applying the Conditions: The Derivation

A.4.1 Stationarity

First, we find the partial derivative of the rate-distortion function $R_i(D_i)$ with respect to D_i. For mathematical convenience, we use the natural logarithm, as the constant factor $\ln(2)$ can be absorbed into the Lagrange multiplier λ.
$$R_i(D_i) = \frac{1}{2\ln(2)} \left(\ln(\sigma_i^2) - \ln(D_i)\right)$$

$$\frac{\partial R_i}{\partial D_i} = -\frac{1}{2\ln(2) D_i}$$

Now, setting the partial derivative of the Lagrangian with respect to D_i to zero (and absorbing the constant $1/(2\ln(2))$ into a redefined multiplier λ' for notational clarity):

$$\frac{\partial \mathcal{L}}{\partial D_i} = -\frac{1}{D_i} + \lambda' s_i + \mu_i - \nu_i = 0$$

Rearranging this gives our fundamental relationship for the optimal distortion D_i:

$$\frac{1}{D_i} = \lambda' s_i + \mu_i - \nu_i \quad \Rightarrow \quad D_i = \frac{1}{\lambda' s_i + \mu_i - \nu_i} \tag{A.7}$$

A.4.2 Analysis via Complementary Slackness

We now analyze three mutually exclusive cases for any given frequency component i based on which constraints are active.

Case 1: Internal Solution (Unconstrained). The optimal distortion D_i lies strictly between its physical bounds: $D_{\min,i} < D_i < \sigma_i^2$.

- Since $D_i < \sigma_i^2$, the constraint $(D_i - \sigma_i^2)$ is inactive (non-zero). By complementary slackness, this forces its multiplier $\boldsymbol{\mu_i = 0}$.
- Since $D_i > D_{\min,i}$, the constraint $(D_{\min,i} - D_i)$ is also inactive. This implies its multiplier $\boldsymbol{\nu_i = 0}$.

Substituting $\mu_i = 0$ and $\nu_i = 0$ into Equation A.7:

$$D_i = \frac{1}{\lambda' s_i}$$

If we define a constant "water-level" parameter $d = 1/\lambda'$, the optimal distortion is $D_i = d/s_i$. This result is highly intuitive: the allocated MSE distortion is inversely proportional to the component's sensitivity.

Case 2: Upper Bound is Active. The optimal distortion is capped at its maximum possible value: $D_i = \sigma_i^2$.

- The lower bound constraint is inactive, so from complementary slackness, $\boldsymbol{\nu_i = 0}$.
- The upper bound constraint is active, so its multiplier μ_i can be non-zero ($\mu_i \geq 0$).

This scenario occurs when the unconstrained solution would have violated the upper bound, i.e., when $d/s_i > \sigma_i^2$. The constraint becomes binding, and the allocation is limited to $D_i = \sigma_i^2$. At this point, the component is effectively discarded with no rate cost.

Case 3: Lower Bound is Active. The optimal distortion is set at its minimum possible value: $D_i = D_{\min,i}$.

- The upper bound constraint is inactive, so from complementary slackness, $\boldsymbol{\mu_i = 0}$.
- The lower bound is active, so its multiplier ν_i can be non-zero ($\nu_i \geq 0$).

This scenario occurs when the unconstrained solution would have violated the lower bound, i.e., when $d/s_i < D_{\min,i}$. The constraint becomes binding, forcing the distortion to its minimum feasible level.

A.5 Synthesis of the Result

Combining these three cases yields a complete, piecewise function for the optimal MSE distortion allocation D_i^*. The allocation for each component is determined by the single global parameter d (which is determined by λ'), scaled by the component's specific sensitivity s_i.

$$D_i^*(d) = \begin{cases} \sigma_i^2 & \text{if } \frac{d}{s_i} > \sigma_i^2 \quad \text{(Case 2: Capped at max power)} \\ D_{\min,i} & \text{if } \frac{d}{s_i} < D_{\min,i} \quad \text{(Case 3: Floored at min distortion)} \\ \frac{d}{s_i} & \text{otherwise} \quad \text{(Case 1: Inverse sensitivity allocation)} \end{cases} \tag{A.8}$$

This can be expressed more compactly as:

$$D_i^*(d) = \text{clip}\left(\frac{d}{s_i}, D_{\min,i}, \sigma_i^2\right) \tag{A.9}$$

This is precisely the reverse water-filling logic that underpins the OptS algorithm. The global parameter d is found numerically such that the total weighted budget constraint, $\sum_{i=1}^{2M} s_i D_i^*(d) = D_T^{\text{SWE}}$, is satisfied. This derivation provides the theoretical foundation for the optimal distortion profiles used to generate DOC-compliant quantization tables in Chapter 3.

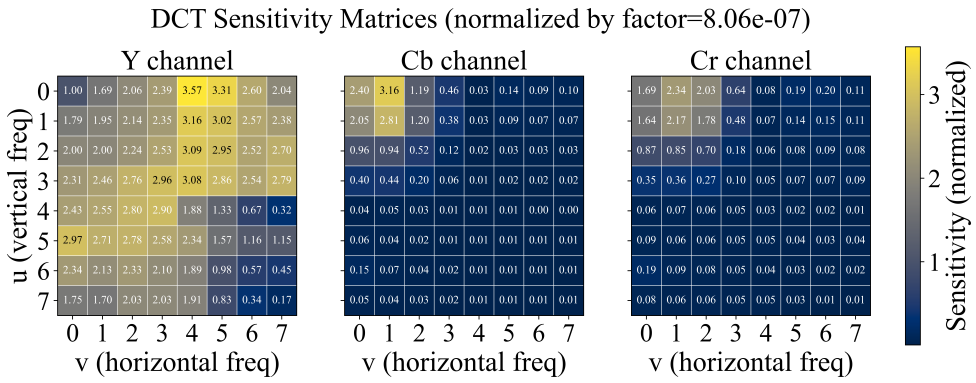
Appendix B

Offline Estimation of the Sensitivity

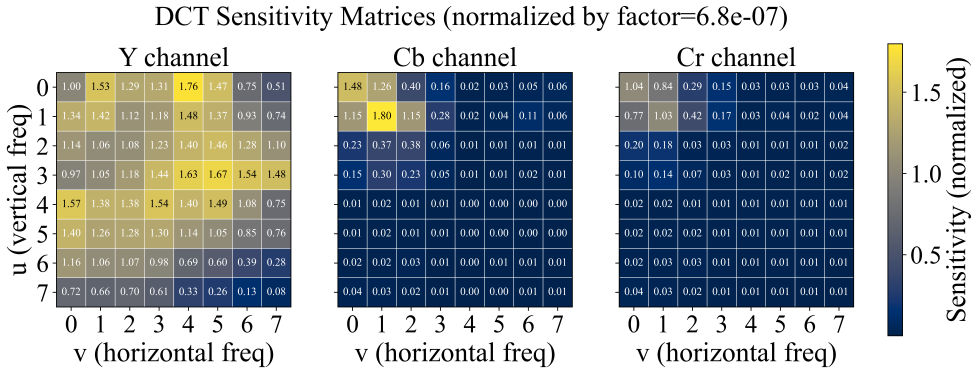
In Figs. B.1 and B.2, we plot the estimated DCT sensitivities introduced in Chapter 3; all values are normalized by the Y–channel DC sensitivity to enable relative comparison across channels (the normalization factor $\alpha = S_Y(0,0)$ is shown in the title), and each heatmap is displayed in the original 8×8 DCT layout (u, v) (via inverse zigzag), with the DC term at the top–left and higher vertical/horizontal frequencies along u/v.

B.1 Model-Dependent Frequency Sensitivity

Across the four models shown in Figure 8.2, the estimated sensitivity of the luma (Y) channel exhibits a broadly consistent qualitative pattern—high sensitivity concentrated in the low-frequency DCT components with a monotonic decay toward higher frequencies—yet the magnitude and sharpness of this decay vary noticeably across architectures. These differences stem from each model family's inductive biases and feature-extraction strategies. Lightweight CNNs such as MobileNetV2 and MnasNet display a pronounced emphasis on very low-frequency coefficients, reflecting their stronger reliance on coarse spatial structures. In contrast, larger architectures such as RegNetY-400MF and ConvNeXt-T present a flatter sensitivity distribution extending into the low- and mid-frequency bands, indicating that they exploit a broader range of frequency information for decision-making. Despite these architectural differences, all models consistently attribute substantially higher importance to Y-channel low frequencies compared to chrominance channels, underscoring the central role of luminance-driven structure in modern DNN feature hierarchies.

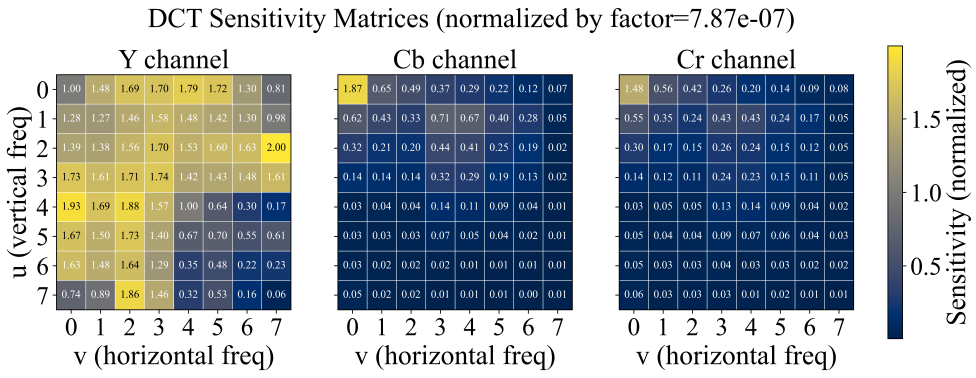


(a) MobileNetV2

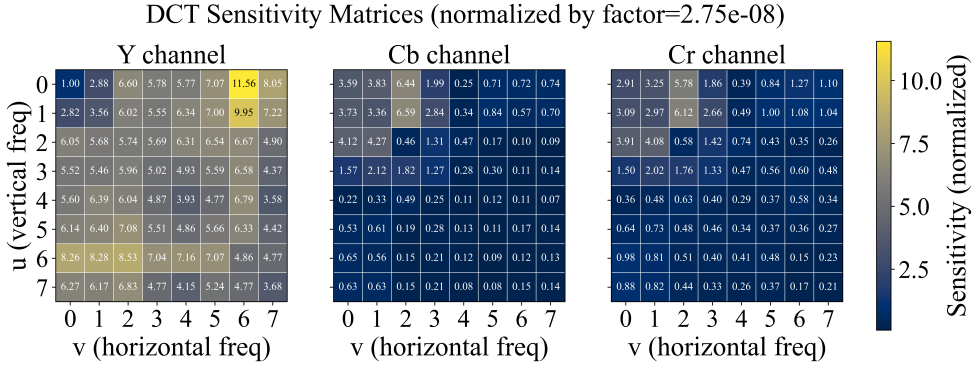


(b) MnasNet

Figure B.1: The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, normalized by the sensitivity value of the DC frequency for better visualization.



(a) RegNetY-400MF



(b) ConvNeXt-T

Figure B.2: The estimated sensitivity is shown for four popular pre-trained models on the ImageNet dataset, normalized by the sensitivity value of the DC frequency for better visualization..

Appendix C

Rate–Distortion Evidence for OptD

Figure C.1 reports the rate–distortion (PSNR vs. bpp) comparison for *OptD* implemented in our pipeline in Chapter 3 as a special case of *OptS* with *uniform sensitivities*, i.e., $s_i \equiv 1$ for all DCT frequencies. Thus, the proposed encoder uses SDQ with quantization tables designed by OptD, while the baseline curve corresponds to default JPEG. The joint distortion allocation and index optimization inherent to SDQ+OptD yield consistently higher PSNR at the same bitrate (or, equivalently, lower bitrate for the same PSNR) than standard JPEG, confirming the advantage of the proposed allocation framework in rate-distortion setup.

Purpose. The plot aggregates the R–D points corresponding to all SDQ+OptD R–A operating points reported in the main results. The resulting curve maintains a favorable R–A trade-off, supporting the premise that SDQ+OptS can be tuned to balance human-centric (R–D) and machine-centric (R–A) objectives.

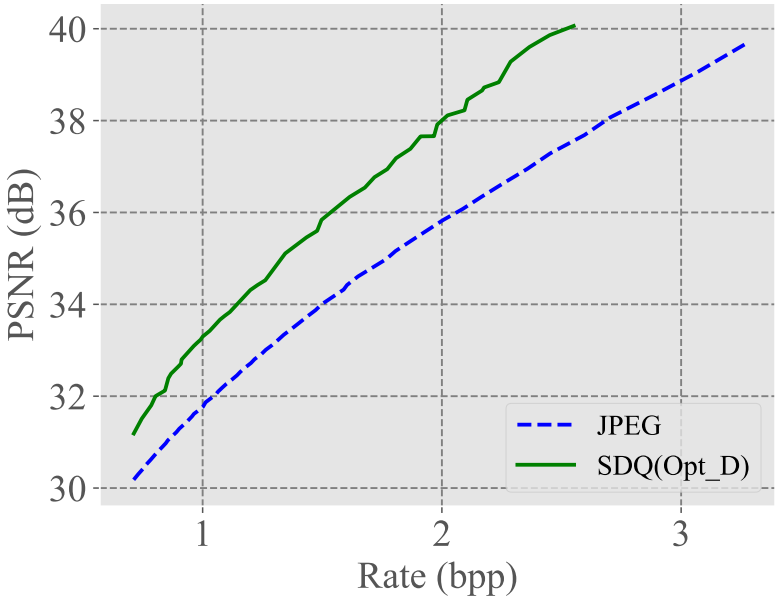


Figure C.1: Rate–Distortion (PSNR vs. bpp) comparison between the default JPEG encoder and OptD with *joint distortion allocation*. Across the tested bitrate range, the OptD curve consistently lies above the JPEG curve, indicating higher PSNR at the same rate—or equivalently, lower rate for the same PSNR—thus demonstrating the superiority of the proposed allocation strategy.