The Interplay of Information Theory and Deep Learning: Frameworks to Improve Deep Learning Efficiency and Accuracy

by

Shayan Mohajer Hamidi

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, 2024

© Shayan Mohajer Hamidi 2024

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:	Xiaolin Wu Professor, Dept. of ECE, McMaster University
Supervisor(s):	En-Hui Yang Professor, Dept. of ECE, University of Waterloo
Internal Member:	Zhou Wang Professor, Dept. of ECE, University of Waterloo
	Mark Crowley Associate 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

The intersection of information theory (IT) and machine learning (ML) represents a promising, yet relatively under-explored, frontier with significant potential for innovation. Despite the clear benefits of combining these fields, progress has been limited by two main challenges: (i) the highly specialized nature of IT and ML, which creates a barrier to cross-disciplinary expertise, and (ii) the computational complexity involved in applying information-theoretic concepts to large-scale ML problems.

This dissertation seeks to overcome these challenges and explore the rich possibilities at the intersection of IT and ML. By leveraging powerful tools and concepts from IT, we aim to uncover novel insights and develop innovative ML algorithms. Given that deep neural networks (DNNs) form the backbone of modern ML models, the integration of IT principles into ML requires a focus on optimizing the training and performance of DNNs using information-theoretic frameworks.

While DNNs have a broad range of applications, this thesis narrows its focus to two key areas: *classification* and *generative* DNNs. The objective is to harness IT principles to enhance the performance of these models.

• **Classification DNNs**. For classification DNNs, this dissertation targets improvements in three critical areas:

(i) Improving classification accuracy. The performance of classification DNNs is traditionally measured by classification accuracy, but we argue that conventional error metrics are insufficient for capturing a model's true performance. By introducing the concepts of conditional mutual information (CMI) and normalized conditional mutual information (NCMI), we propose a new metric for evaluating DNNs. The CMI measures intra-class concentration, while the ratio of CMI to NCMI reflects inter-class separation. We then modify the standard loss function in deep learning (DL) framework to minimize the standard cross entropy function subject to an NCMI constraint, yielding CMI constrained deep learning (CMIC-DL). Then, via extensive experiment results, we show that DNNs trained within CMIC-DL achieves a higher classification accuracy compared to the state-of-the-art models trained within the standard DL and other loss functions in the literature.

(ii) Enhancing distributed learning accuracy. In the context of distributed learning, particularly federated learning (FL), we tackle the challenge of class imbalance using information-theoretic concepts to improve the accuracy of the shared global model. To this end, we introduce new information-theoretic quantities into FL and propose a modified loss function based on these principles. This leads to the development of a federated learning framework, Fed-IT, which enhances the classification accuracy of models trained in distributed environments.

(iii) Reduce the size and training/inference complexity. We introduce coded deep learning (CDL), a novel framework aimed at reducing the computational and storage complexity of classification DNNs. CDL achieves this by compressing model weights and activations through probabilistic quantization. Both forward and backward passes during training are performed using quantized weights and activations, significantly reducing floating-point operations and computational overhead. Furthermore, CDL imposes entropy constraints on weights and activations, ensuring compressibility at every stage of training, which also reduces communication costs in parallel computing environments. This leads to models that are more efficient in both training and inference, with lower storage and computational requirements.

• **Generative DNNs**. For generative DNNs, this dissertation focuses on diffusion models and their application to solving inverse problems. Inverse problems are common in fields like medical imaging, signal processing, and physics, where the goal is to recover an underlying cause from corrupted or incomplete observations. These problems are often ill-posed, with multiple possible solutions or high sensitivity to small changes in the data. In this dissertation, we enhance the performance of diffusion models by incorporating probabilistic principles, making them more effective at capturing the posterior distribution of the underlying causes in inverse problems. This approach improves the model's ability to accurately reconstruct signals and provides more reliable solutions in challenging inverse problem scenarios.

Overall, this dissertation demonstrates the powerful synergy between IT and ML, showcasing novel methods that improve the accuracy and efficiency of both classification and generative DNNs. By addressing key challenges in training and optimization, this work lays the foundation for future research at the intersection of these two fields.

Acknowledgements

First and foremost, I wish to thank professor *En-Hui Yang*, who guided and supported me throughout my thesis preparation. During my PhD 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 *Mark Crowley*, professor *Charles Clarke*, and professor *Xiaolin Wu*. 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, *Ahmed H. Salamah, Linfeng Ye, Cameron Tan* and *Kaixiang Zheng* who are the invaluable resource of my improvement and happiness.

I owe an immense debt of gratitude to my parents, whose unwavering devotion and dedication have shaped every success I have achieved. Words can never fully express the depth of their support and the profound impact they have had on my journey. I am also deeply grateful to my late sister, *ava* for her constant love and encouragement.

Last but certainly not least, I wish to express my deepest gratitude to my beloved wife. Her unwavering encouragement, patience, and love have been the quiet force behind everything I have accomplished over the past five years. It feels unjust that her name does not grace the first page of this dissertation, as her contribution, through the love and joy she brings, is no less significant than mine.

Dedication

To my family.

Table of Contents

Examining Committee ii

Author's Declaration iii

Abstract iv

Acknowledgements vi

Dedication vii

List of Figures xiii

List of Tables xvii

1 Introduction 1

1.1 Thesis Motivation 1

1.1.1 Improving the Accuracy of Classification DNNs 2

1.1.2 Reducing the Size of Classification DNNs 2

1.1.3 Reducing the Training/Testing Computation of Classification DNNs 4

1.1.4 Improving the quality of images synthesized by generative DNNs . . 4

1.2 Thesis Contributions 5

1.3 Thesis organization 7

2 Background and Preliminaries **8**
2.1 Machine Learning . . . 8
2.2 Information Theory . . . 10

3 Conditional Mutual Information Constrained Deep Learning for Classification **14**
3.1 Introduction . . . 15
3.2 Performance of DNNs: Concentration and Separation . . . 17
3.2.1 Notation Used in This Chapter . . . 17
3.2.2 Error Rate . . . 18
3.2.3 Concentration . . . 20
3.2.4 Separation and NCMI . . . 23
3.2.5 Related Works . . . 24
3.3 NCMI Vs. Accuracy . . . 26
3.4 CMIC Deep Learning . . . 26
3.4.1 Optimization Problem Formulation . . . 27
3.4.2 Algorithm for Solving the Optimization in (5.46) . . . 29
3.5 Experiment Results . . . 30
3.5.1 Experiments on CIFAR-100 . . . 31
3.5.2 Experiments on ImageNet . . . 32
3.5.3 CMIC-DL Counterpart of Table 3.1 . . . 34
3.5.4 Concentration and Separation Visualization . . . 35
3.5.5 Evolution of CMI, Γ, NCMI, and error rate . . . 36
3.5.6 Semantic Meaning of CMI Through Centroid Visualization . . . 38
3.5.7 Robustness against adversarial attacks . . . 39
3.6 Other Information Quantities for Separation . . . 42
3.6.1 Information Quantity Γ' . . . 43
3.6.2 Information Quantity Γ'' . . . 45
3.7 NCMI Vs. Accuracy on ImageNet . . . 46
3.8 Chapter Summary and Directions for Future Work . . . 47

4 Fed-IT: Addressing Class Imbalance in Federated Learning through an Information-Theoretic Lens **49**

4.1 Introduction . . . 50
4.2 Related Work . . . 52
4.2.1 Class imbalance in centralized training . . . 52
4.2.2 Class imbalance in FL . . . 53
4.3 Notation and Preliminaries . . . 55
4.3.1 Notation . . . 55
4.3.2 FL setup . . . 55
4.4 Motivation and Methodology . . . 56
4.4.1 Motivation . . . 56
4.4.2 Methodology . . . 57
4.4.3 New information theoretic Loss for Fed-IT . . . 60
4.4.4 Algorithm for Fed-IT . . . 62
4.5 Experiment Results . . . 62
4.5.1 Analysis of the Results . . . 63
4.5.2 Ablation study . . . 63
4.5.3 Visualization of DNNs's Output with Different CMI Values . . . 64
4.5.4 Fig. 4.3 with Different Classes . . . 65
4.6 Chapter Summary . . . 66

5 Coded Deep Learning **67**

5.1 Introduction . . . 68
5.2 Related Works . . . 71
5.2.1 Quantization-Aware Training (QAT) . . . 71
5.2.2 Accounting for Compression during Training . . . 72
5.2.3 Modes of Parallelism . . . 72
5.3 Notation and Preliminaries . . . 73

5.3.1 Notation Used in This Chapter . . . 73
5.3.2 Preliminaries . . . 74
5.4 Probabilistic Quantization Method . . . 74
5.4.1 Probabilistic Quantizer $\mathsf{Q}_{\mathrm{p}}(\cdot)$ . . . 74
5.4.2 *Soft Deterministic* Quantizer $\mathsf{Q}_{\mathrm{d}}(\cdot)$ . . . 77
5.5 CDL and R-CDL . . . 80
5.5.1 Incorporating Trainable Probabilistic Quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ Into CDL . . . 81
5.5.2 Computation of Gradients of the Loss Function (5.24) . . . 82
5.5.3 Entropy Computation of Quantized Weights and Activations . . . 85
5.5.4 New Objective Function with Entropy Constraints . . . 86
5.5.5 Algorithm for Solving the Optimization in (5.46) . . . 89
5.5.6 R-CDL . . . 90
5.6 Experiments . . . 91
5.6.1 Performance Metrics . . . 91
5.6.2 Implementation Details of CDL and R-CDL . . . 91
5.6.3 Experiments on ImageNet . . . 92
5.6.4 Experiments on CIFAR-100 . . . 94
5.6.5 Evolution of Weight Distribution . . . 95
5.6.6 Average Number of Bits vs Epoch for Facilitating Data/Model Parallelism . . . 95
5.6.7 Convergence Curves for CDL and R-CDL . . . 95
5.7 CDL with Non-uniform quantization . . . 96
5.7.1 Training on top of pre-trained models . . . 96
5.8 Chapter Summary . . . 97

6 Enhancing Diffusion Models for Inverse Problems with Covariance-Aware Posterior Sampling 108
6.1 Introduction . . . 109

6.2 Related Work . . . 111
6.2.1 Approaches for Solving Inverse Problems . . . 111
6.2.2 Diffusion Models for Inverse Problems . . . 113
6.2.3 Higher Order Approximation of reverse process . . . 113
6.3 Background and Preliminaries . . . 113
6.3.1 Diffusion Models . . . 113
6.3.2 Diffusion Models for Solving Inverse Problems . . . 115
6.4 Covariance-Aware Diffusion Posterior Sampling . . . 117
6.5 Experiments . . . 123
6.5.1 Quantitative Results. . . . 124
6.5.2 Qualitative Results . . . 125
6.5.3 Toy Dataset . . . 126
6.5.4 Toy dataset . . . 127
6.5.5 SW Distance Calculation . . . 129
6.5.6 More Qualitative Results . . . 129
6.6 Chapter Summary . . . 130

7 Conclusions and future work **133**
7.1 Summary . . . 133
7.2 Contributions . . . 134
7.3 Directions for Future Work . . . 135

References **137**

List of Figures

3.1 The mappings from the label space to the input space, and from the input space to the output space of a DNN. Here caricatures are used to depict label and input spaces, where each of the three instances in the label space is mapped to two instances in input space according to $P_{X|Y}(\cdot|y_i)$, for $i \in \{1, 2, 3\}$. On the other hand, the figure for the output space is obtained from a real example, where for the ResNet56 model trained on CIFAR-100, the output probability vectors corresponding to all validation sample instances from three randomly-picked classes are projected over the two-dimensional probability simplex. 21

3.2 The error rate ϵ^* vs NCMI value over the validation set of popular pre-trained models on CIFAR-100 dataset. 27

3.3 Visualization and comparison of concentration and separation: ResNet56 trained via CE vs ResNet56 trained via CMIC, where different shapes indicate different classes. 36

3.4 The evolution curves of (a) CMI, (b) Γ, (c) NCMI, and (d) ϵ^* over the course of training ResNet-56 over CIFAR-100 using CE and CMIC frameworks. . 37

3.5 Illustration of CMI's semantic meaning—(i) top: goldfish; (ii) bottom: bee-eater—where the first two columns show the synthetic centroid images, and the last three columns show natural images from the respective classes. . . 39

3.6 Illustration of CMI's semantic meaning: synthetic centroid images (top row) vs natural images in class goldfish. The CMI $I(X; \hat{Y}|Y = y)$ for class goldfish is 0.927. For each natural image in the class, its value of $KL(q_x||s_y)$ is printed at its top left corner. 40

3.7 Illustration of CMI's semantic meaning: synthetic centroid images (top row) vs natural images in class bee-eater. The CMI $I(X;\hat{Y}|Y=y)$ for class bee-eater is 0.204. For each natural image in the class, its value of $KL(q_x||s_y)$ is printed at its top left corner. 41

3.8 The robustness of a simple DNN over MNIST dataset trained within the conventional CE-based DL and CMIC-DL frameworks against (a) FGSM attack and (b) PGD attack with 5 iteraions, respectively. 43

3.9 The error rate vs NCMI value, over the validation set, of popular pre-trained models on ImageNet dataset. 47

4.1 Illustration of the output probability distribution clusters of ResNet 56 corresponding to three randomly selected classes, projected over the two-dimensional probability simplex: (a) trained over the CIFAR-100 class-balanced dataset using the conventional CE loss; (b) trained over a class-imbalanced dataset using the conventional CE loss; (c) trained over the class-imbalanced dataset using the proposed information theoretic loss $\mathcal{L}_{\text{Fed-IT}}$. 57

4.2 Visualization of the output probability space of the DNN with (a) low CMI value, and (b) high CMI value. 64

4.3 The output probability space of a DNN to three classes of CIFAR-100 dataset over the two-dimensional probability simplex. (a) The dataset is class-balanced and CE loss is used. (b) The dataset is class-imbalanced, and the ratio of samples in class 0, 1 and 2 is 1:5:10, and CE loss is used. (c) The dataset is class-imbalanced similar to (b), and loss in $\mathcal{L}_{\text{Fed-IT}}$. . . . 65

5.1 Illustration of the partial derivatives of $\mathsf{Q}_{\text{d}}(\theta)$ w.r.t. θ (left), and q (right) for $\alpha = \{100, 300, 500, 700\}$, where b and q equal 3 and 0.1, respectively. . . 75

5.2 Illustration of $\mathsf{Q}_{\text{u}}(\cdot)$ vs $\mathsf{Q}_{\text{d}}(\cdot)$ with $\alpha = \{100, 300, 500, 700\}$, where b and q are set to 3 and 0.1, respectively. 81

5.3 Illustration of the CDL's mechanism. 83

5.4 Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) on ImageNet for ResNet-18. All models are trained from **scratch**. 98

5.5 Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) on ImageNet for ResNet-34. All models are trained from **scratch**. 99

5.6 Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the test accuracy vs the average number of bits per weight (top)/activation (bottom) on CIFAR-100 dataset: (a) ResNet-20 and (b) ResNet-44. 100

5.7 Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the test accuracy vs the average number of bits per weight (top)/activation (bottom) on CIFAR-100 dataset: (a) ResNet-56 and (b) ResNet-110. 101

5.8 Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the test accuracy vs the average number of bits per weight (top)/activation (bottom) on CIFAR-100 dataset: (a) VGG-13 and (b) Wide-ResNet-28-10. 102

5.9 The evolution of weight distribution of a convolutional layer in ResNet-110 during training via CDL on CIFAR-100. 103

5.10 The average number of bits required to encode (a) CDL weights and (b) CDL activations vs epochs for ResNet-110 trained over CIFAR-100. 103

5.11 The average number of bits required to encode (a) R-CDL weights and (b) R-CDL activations vs epochs for ResNet-110 trained over CIFAR-100. . . . 104

5.12 The convergence of objective function in (a) CDL, and (b) R-CDL over the course of training ResNet-110 on CIFAR-100. 105

5.13 Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) for LCQ, and R-CDL+LCQ on ImageNet: (a) ResNet-18, and (b) ResNet-34. All models are trained from **scratch**. 106

5.14 Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) on ImageNet: (a) ResNet-18, and (b) ResNet-34. All models are trained from **pre-trained** FP models. 107

6.1 Qualitative results on FFHQ dataset. 124

6.2 Visualizing the first two dimensions of the estimated posterior distributions for the configuration ($d = 80$, $m = 1$, $\sigma = 10^{-1}$) for a randomly generated $\mathbf{A}$. 128
6.3 Qualitative results on FFHQ dataset. 131
6.4 Qualitative results on FFHQ dataset. 132

List of Tables

3.1 CMI, Γ, NCMI, and error rate values, over the validation set, of the models trained with **CE** loss on CIFAR-100. 25

3.2 The validation accuracies (%) of different models trained by CMIC-DL and different benchmark methods over **CIFAR-100 dataset**, which are averaged over three different random seeds, and where **bold** and underlined values denote the best and second best results, respectively. 33

3.3 The respective NCMI values, measured over the validation set, of the models trained in Table 3.2 via different benchmark methods. The values are averaged over thee different runs. 33

3.4 **Hyper-parameters**, λ^* and β^* used for CMIC-DL in Table 3.2. 33

3.5 The validation accuracies (%) of different models trained by CMIC-DL and different benchmark methods on **ImageNet**. 34

3.6 CMI, Γ, NCMI, and error rate values, over the validation set, of the models trained by **CMIC-DL** on CIFAR-100. 35

3.7 CMI, Γ, NCMI, and error rate values, over the validation set, of some pre-trained models on ImageNet dataset. 46

4.1 The validation accuracy (%) for different FL methods over CIFAR-$\{10, 100\}$ and TinyImageNet datasets. The reported numbers (avg±std) are obtained from 5 runs. **Bold** and underlined values denote the best and second best results, respectively. 62

4.2 The effect of hyper-parameter α and γ in Fed-IT. 64

5.1 (Top-1 accuracy, Average # of bits per weight, Average # of bits per activation) given by models trained by CDL and R-CDL with $(\lambda, \gamma) = (0, 0)$ and $b = \{2, 3, 4\}$. 93

6.1 **Quantitative results on the $1k$ validation set of FFHQ 256×256 and ImageNet 256×256 dataset.** We use **bold** and underline for the best and second best results, respectively. 123

6.2 SW distance between the true and estimated posterior on toy dataset. . . . 127

6.3 Sliced Wasserstein for VE-DDPM. 130

6.4 Sliced Wasserstein for the GMM case for the reverse VE SDEs discretized with Euler-Maruyama. 131

Chapter 1

Introduction

1.1 Thesis Motivation

Despite several efforts [61, 178, 198], the intersection of IT and ML remains a relatively under-explored frontier, rich with opportunities for further investigation. While both fields have made significant strides independently, their synergistic potential remains largely untapped. This oversight is particularly surprising given the fundamental compatibility of their core concepts. IT, rooted in the quantification of uncertainty and information transmission, provides a natural framework for analyzing and optimizing ML algorithms [40].

Despite the clear benefits of such an interdisciplinary approach, the intersection of IT and ML has been hindered by several factors. One major obstacle is the specialized nature of these fields, leading to a lack of cross-disciplinary expertise. Researchers in ML may not be familiar with the intricacies of IT, while information theorists may be less acquainted with the challenges and opportunities presented by ML. This divide has limited the exchange of ideas and collaborations between the two communities.

Another factor contributing to the under-exploration of this intersection is the computational complexity associated with applying information-theoretic concepts to large-scale ML problems. Many information-theoretic measures, such as mutual information and entropy, are computationally expensive to compute, particularly for high-dimensional data [9, 61, 88, 115]. This has hindered the practical application of these concepts in real-world ML scenarios.

This dissertation aims to address these challenges and delve deeper into the unexplored

territory of the intersection between IT and ML. By leveraging the powerful tools and concepts from IT, we seek to uncover novel insights and develop innovative ML algorithms.

We believe that by bridging the gap between these two fields, we can unlock new possibilities for ML. Applying information-theoretic principles can offer rigorous frameworks for optimizing tasks like classification, generative modeling, and inference. Moreover, IT provides a deeper theoretical lens for analyzing the inherent limitations, efficiency, and trade-offs in learning algorithms, helping to guide the development of more robust and efficient models.

Since DNNs form the core of modern ML models (see Chapter 2 for more details on DNNs), any attempt to integrate IT into ML must focus on leveraging information-theoretic principles during the training and optimization of DNNs. DNNs are employed for a variety of tasks, including classification [84], regression [146], reinforcement learning [106,213], and generative modeling [65]. Given the broad scope of these applications, this dissertation narrows its focus to *classification* and *generative* DNNs, where the aim is to harness IT principles to enhance the DNNs performance.

In particular, for *classification* DNNs, we aim to enhance their performance in three key areas: (i) improving classification accuracy, (ii) reducing model size, and (iii) reducing training and testing computation. Additionally, for *generative* DNNs, our goal is to enhance the quality of generated images both quantitatively and qualitatively. Each of these aspects will be briefly discussed in the sequel.

1.1.1 Improving the Accuracy of Classification DNNs

The performance of classification DNNs is typically assessed by their classification accuracy (or, conversely, their error rate) [127]. Thus, our primary objective in applying concepts from IT to DNN training is to utilize it as a tool for enhancing classification accuracy. By integrating information-theoretic concepts into the learning process, we can bolster the network's ability to differentiate between classes and reduce error rates. In addition to conventional centralized training, where all data is aggregated on a central server, we also explore the distributed learning paradigm, with a particular focus on federated learning (FL) [147]. Our aim is to improve classification accuracy within this distributed framework.

1.1.2 Reducing the Size of Classification DNNs

The number of parameters in a high performance DNN can be as large as hundreds of millions or even billions. This makes it difficult to deploy such a DNN in resource limited

environments due to its (i) large storage overhead, and (ii) high demand for computation and memory. In general, there could be four reasons motivating the compression of DNNs:

1. A smaller model means **less overhead** when exporting models to clients/users. For instance; Tesla periodically copies new models from their servers to customers' cars. Obviously, smaller models require less communication in such over-the-air updates, making frequent updates more feasible. Another example is the Apple Store: mobile applications above 100 MB will not download until a user connects to WiFi. As a result, a new feature that increases the binary size by 100MB will receive much more scrutiny than one that increases it by 10MB. Thus, putting a large DNN model in a mobile application is infeasible.

2. The second reason is **inference speed**. Numerous mobile scenarios necessitate low-latency and real-time inference, encompassing self-driving cars and AR glasses, where ensuring safety and optimizing user experience heavily relies on minimizing latency. Employing a smaller model plays a pivotal role in enhancing the inference speed on these devices. From a computational standpoint, smaller DNN models demand fewer arithmetic operations and computation cycles, while from a memory standpoint, they require fewer memory reference cycles. In fact, if the model's size is sufficiently compact, it can even be accommodated within the on-chip SRAM, enabling faster access compared to off-chip DRAM memory.

3. The third reason is **energy consumption**. Running large DNNs requires significant memory bandwidth to fetch the weights. This consumes considerable energy and is problematic for battery-constrained mobile devices. As a result, IOS 10 requires iPhones to be plugged into chargers while performing photo analysis. Memory access dominates energy consumption. Smaller neural networks require less memory access to fetch the model, saving energy and extending battery life.

4. The fourth reason is **cost**. When deploying DNNs on Application-Specific Integrated Circuits (ASICs), a sufficiently small model can be stored on-chip directly. As smaller models require less on-chip SRAM, this permits a smaller ASIC die thus making the chip less expensive.

Smaller deep learning models are also appealing when deployed in large-scale data centers as cloud AI. Future data center workloads would be populated with AI applications, such as Google Cloud Machine Learning and Amazon Rekognition. The cost of maintaining such large-scale data centers is tremendous. Smaller DNN models reduce the computation of the workload and take less energy to run. This helps to reduce the electricity bill and the total cost of ownership of running a data center with deep learning workloads.

1.1.3 Reducing the Training/Testing Computation of Classification DNNs

Reducing the computational demands of training and testing DNNs is crucial for several reasons, particularly in the context of classification tasks. First, training large-scale DNNs requires substantial computational resources, often involving high-performance GPUs or distributed computing clusters [95,118,193,219]. This makes the training process expensive, both in terms of time and energy consumption. For organizations or researchers without access to extensive computational infrastructure, the ability to reduce training time can significantly lower the barrier to deploying state-of-the-art models. Additionally, more efficient training processes allow for faster iterations, enabling quicker experimentation, model tuning, and deployment of optimized networks.

On the testing side, the computational cost of inference can also be a bottleneck, especially when deploying DNNs in real-time applications. For instance, tasks such as image recognition in autonomous vehicles [103], facial recognition in security systems [161], or object detection in mobile devices [170] demand low-latency performance. In these settings, reducing the computational footprint during inference is critical to achieving real-time processing. Efficient inference also extends the use of DNNs to resource-constrained devices like smartphones, wearables, or edge devices, where memory and power consumption are limited. By minimizing the computational overhead, we can broaden the applicability of DNNs to a wider range of practical scenarios.

Moreover, as DNNs grow larger and more complex, the environmental impact of training and testing models becomes a growing concern. The energy consumption of large-scale models has been increasingly scrutinized due to the environmental costs associated with powering data centers. Reducing computation not only makes DNNs more sustainable but also aligns with the global push towards greener, more energy-efficient AI solutions. Therefore, focusing on reducing the computational requirements of classification DNNs is not only essential for improving accessibility and usability but also for addressing broader concerns related to energy efficiency and sustainability in machine learning.

1.1.4 Improving the quality of images synthesized by generative DNNs

For generative DNNs, the quality of generated images is a critical measure of success. Quantitatively, this can be assessed through metrics such as Fréchet Inception Distance (FID) or Inception Score (IS), which evaluate the similarity between generated images and

real-world samples. Qualitatively, the goal is to produce visually convincing and high-resolution images that are indistinguishable from real images to human observers. By applying information-theoretic principles, we aim to improve the fidelity and diversity of generated images.

1.2 Thesis Contributions

The primary goal of this thesis is to establish a framework for improving DNNs in the areas outlined in the previous section. Specifically, the key contributions of this thesis are as follows:

- *For classification DNNs*

  1. **Improve their classification accuracy.** We discuss that the conventional error rate used for evaluating DNNs is not a good metric to gauge DNNs. Then, by using information theoretic concepts, we define a new metric to evaluate DNNs. Next, we use this metric to modify the loss function in the conventional training to obtain DNNs with higher classification accuracy. To shed more light, we introduce the concepts of conditional mutual information (CMI) and normalized conditional mutual information (NCMI) to measure the concentration and separation performance of a classification DNN in the output probability distribution space of the DNN, where CMI and the ratio between CMI and NCMI represent the intra-class concentration and inter-class separation of the DNN, respectively. Then, we use the NCMI to modify the loss function in conventional training to increase the classification accuracy of the DNNs.

  2. **Improve classification accuracy in distributed learning.** We also explore the distributed learning setting, particularly FL, and address the class-imbalance problem by leveraging information-theoretic concepts to enhance the classification accuracy of the shared model. Building on this, we introduce novel information-theoretic quantities into FL, propose a new loss function based on these principles, and develop a federated learning framework called Fed-IT.

  3. **Reduce the size and training/inference complexity of classification DNNs.** We introduce coded deep learning (CDL), a novel approach designed to significantly compress model weights and activations, reduce computational complexity during both the training and inference stages, and enable efficient model and data parallelism. CDL achieves these improvements through a series of innovative

techniques aimed at optimizing both the memory and computational requirements of deep neural networks. Specifically, CDL operates through the following key contributions:

- *Probabilistic Quantization of Weights and Activations.* We propose a novel probabilistic method for quantizing both model weights and activations. This method includes a soft, differentiable variant that allows for seamless gradient calculation during training, offering an analytic solution to ensure efficient backpropagation.
- *Quantized Training for Reduced Complexity*: Both the forward and backward passes during training are executed over quantized weights and activations, drastically reducing the number of floating-point operations. This approach leads to a substantial decrease in training computational complexity while maintaining the model's ability to learn effectively.
- *Entropy-Constrained Quantization for Compression:* During training, both weights and activations are constrained by entropy, ensuring that they are compressible in an information-theoretic sense at any stage of the training process. This constraint not only facilitates efficient compression but also minimizes communication overhead in cases where model or data parallelism is used, enabling faster distributed training.
- *Quantized Model for Efficient Inference:* The trained model produced by CDL is inherently in a quantized format, with compressible weights that reduce both post-training inference complexity and the overall model storage requirements. This makes CDL particularly well-suited for deployment in resource-constrained environments such as edge devices or large-scale systems requiring lightweight inference.

By integrating these components, CDL offers a comprehensive framework that enhances the efficiency of deep learning models, both during training and in post-training deployment, while maintaining high performance and scalability.

- *For generative DNNs*

1. **Enhance the performance of diffusion models for solving inverse problems.** Given the broad scope of generative DNNs, we narrow our research focus to diffusion models [33], a specific and powerful class of generative models. Our research particularly centers on utilizing diffusion models to solve inverse problems, which are fundamental to many disciplines, including medical imaging, signal processing, and physics. Inverse problems are central to many fields, including medical

imaging, signal processing, and physics, where the goal is to recover an underlying cause from observed data. These problems are often ill-posed, meaning that multiple solutions may exist or small changes in the data can lead to large changes in the solution. Improving the performance of models that solve inverse problems is crucial for applications like reconstructing images from noisy data (e.g., in MRI or CT scans), deblurring images, or estimating physical parameters from incomplete observations. In this thesis, we leverage probabilistic principles to enhance diffusion models, making them more effective at capturing the posterior distribution of the underlying cause.

1.3 Thesis organization

The rest of the thesis is organized as follows.

- Chapter 2 discusses essential background on the fundamental principles of DNNs. Additionally, it introduces key concepts from information theory that are crucial for understanding the subsequent discussions in this thesis.
- Chapter 3 introduces the concepts of normalized conditional mutual information (NCMI) to measure the concentration and separation performance of a classification DNN in the output probability distribution space of the DNN, where conditional mutual information (CMI) and the ratio between CMI and NCMI represent the intra-class concentration and inter-class separation of the DNN, respectively.
- Chapter 4 introduces a novel federated learning mechanism, inspired by information-theoretic principles, which we refer to as Fed-IT.
- Chapter 5 proposes coded deep learning (CDL). CDL integrates information-theoretic coding concepts into the inner workings of DL, aiming to substantially compress model weights and activations, reduce computational complexity at both training and post-training inference stages, and enable efficient model/data parallelism.
- Chapter 6 leverages probabilistic techniques to enhance the performance of diffusion models in solving inverse problems.
- Chapter 7 concludes the dissertation by summarizing its key contributions and insights. It also outlines potential directions for future research, offering ideas for further exploration and refinement of the methodologies introduced in this work.

Chapter 2

Background and Preliminaries

In this chapter, we provide an overview of essential concepts in machine learning and information theory, which are covered in Sections 2.1 and 2.2, respectively.

2.1 Machine Learning

In this subsection, we explain some of the terms frequently used in this thesis, and further discuss the inner-working of deep learning.

• Artificial Neural Networks (ANNs): ANNs are the foundation of deep learning. They are composed of interconnected nodes or artificial neurons, organized in layers. Information flows through the network from the input layer, through hidden layers, and finally to the output layer. Each neuron applies a mathematical function to the input and passes the result to the next layer.

• Deep Neural Networks (DNNs): DNNs are ANNs with multiple hidden layers. They are capable of learning complex patterns and representations by extracting hierarchical features from the input data. The depth of the network allows for more expressive power, enabling it to solve intricate tasks.

• Activation Functions: Activation functions introduce non-linearity into neural networks, enabling them to model complex relationships. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU). They determine the output of a neuron based on its weighted sum of inputs.

• Loss Functions: Loss functions quantify the discrepancy between the predicted output of a neural network and the true target values. They serve as a measure of how well the

model is performing during training. Popular loss functions include mean squared error (MSE) for regression tasks and categorical cross-entropy for classification tasks.

• Backpropagation: Backpropagation is an algorithm used to train neural networks. It involves iteratively adjusting the network's weights based on the gradient of the loss function with respect to the weights. By propagating the error backward through the network, the algorithm updates the weights in a way that minimizes the loss and improves the model's predictions.

• Gradient Descent: Gradient descent is the most common technique for training deep neural networks. It is a first-order optimization method by calculating the gradient of the loss function over the variable and moving the variable in the negative direction of the gradient. The step size is proportional to the absolute value of the gradient. The ratio between the step size and the absolute value of the gradient is called learning rate. Calculating the gradient is the key step when performing gradient descent, which is based on the back-propagation algorithm. To calculate the gradient with back-propagation, we need to first calculate each layer's activation by performing a feed-forward pass from the input neuron to the output neuron. This forward pass is also called inference, the output of inference could either be a continuous value in regression problems, or a discrete value in classification problems. The inference result could be correct or wrong, which is quantitatively measured by the loss function. Next, we calculate the gradient of the loss function for each neuron and each weight. The gradients are calculated iteratively from the output layer to the input layer according to the chain rule. Then, we update the weights with gradient descent $\boldsymbol{w}^{t+1}[i,j] = \boldsymbol{w}^{t}[i,j] - \alpha \frac{\nabla L}{\boldsymbol{w}^{t}[i,j]}$. Such feed-forward, back-propagation, and weight update constitute one training iteration. It usually takes hundreds of thousands of iterations to train a deep neural network. Training ResNet-50 on ImageNet, for example, takes 450450 iterations.

• Overfitting and Regularization: Overfitting occurs when a model performs well on the training data but fails to generalize to unseen data. Regularization techniques such as L1 and L2 regularization, dropout, and early stopping are used to combat overfitting. They introduce constraints on the model's complexity, preventing it from memorizing the training data and encouraging generalization.

• Convolutional Neural Networks (CNNs): CNNs are specialized deep learning architectures designed for analyzing visual data such as images. They leverage convolutional layers to extract spatial patterns and hierarchies of features. CNNs have achieved remarkable success in image classification, object detection, and other computer vision tasks.

Directly programming a multi-core CPU or a GPU can be difficult, but luckily the neural network computation can be abstracted into a few basic operations such as convolution,

matrix multiplication, etc. Using these operations that are already highly optimized for high-performance hardware, users only need to focus on high-level neural network architectures rather than low-level implementations. Deep learning frameworks provide these abstractions of neural networks computation, so programmers only need to write a description of the computation, and the deep learning frameworks then efficiently map the computation to high-performance hardware.

In ordinary multi-class classification, let $\mathcal{X} \subseteq \mathbb{R}^d$ be the instance space, and $\mathcal{Y} = [C]$ be the label space, where C is the number of classes. For a specific instance $x \in \mathcal{X}$, denote by $y \in [C]$ the corresponding true label. In addition, for both fully-connected and convolutional layers, denote by $\boldsymbol{w}_l$ the flattened vector representation of weight parameters at layer l, and by L the number of layers. Furthermore, denote by m_l the number of elements in $\boldsymbol{w}_l$. In the legacy DL, the goal is to learn a classifier $F : \mathbb{R}^d \mapsto \mathbb{R}^C$ which minimizes

$$\mathbb{E}_{x\sim\mathcal{P}}\{\mathcal{L}\left(F\left(x|\{\boldsymbol{w}_l\}_{l\in[L]}\right), y\right)\}, \tag{2.1}$$

where $\mathcal{P}$ is a probability distribution over $\mathcal{X}$, and $\mathcal{L}$ is a real-valued loss function.

Neural networks are typically over-parameterized, and there is significant redundancy for deep learning models. This results in a waste of both computation and memory. Therefore, compression of DNNs has become a popular line of research in the recent years. A well-known method in this area is to quantize the parameters (weights) of the neural network. As our work could be related to this approach, in the following, we review some of the recent works in the realm of compression on DNNs.

2.2 Information Theory

Information theory, originally developed by Claude Shannon in 1948, is a mathematical framework for quantifying information, uncertainty, and the efficiency of data transmission [40]. It plays a foundational role in fields such as telecommunications, data compression, and cryptography, but its principles are increasingly being applied to machine learning. At its core, information theory provides tools to measure how much information is present in a signal, how uncertain we are about outcomes, and how we can optimize the encoding and transmission of data to minimize loss and maximize efficiency.

Here is a list of fundamental terminologies in information theory, along with *Explanation*s for each:

• Entropy: Entropy, denoted as $H(X)$, measures the average amount of uncertainty or information content associated with a random variable X. It quantifies how unpredictable or surprising the outcome of X is.

Explanation: If all outcomes of X are equally likely, the entropy is maximized, meaning we have the most uncertainty. If one outcome is highly probable, entropy is low, indicating less uncertainty. In machine learning, entropy can help gauge the unpredictability of data or the uncertainty of predictions.

Formula:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x),$$

where $p(x)$ is the probability of x.

• Joint Entropy: Denoted as $H(X, Y)$, is the measure of uncertainty about two random variables X and Y considered together.

Explanation: Joint entropy quantifies how uncertain we are about the combined outcomes of two variables. If the variables are independent, the joint entropy is simply the sum of their individual entropies. If they are dependent, the joint entropy will be lower due to the shared information between X and Y.

Formula:

$$H(X, Y) = -\sum_{x \in X, y \in Y} p(x, y) \log_2 p(x, y),$$

where $p(x, y)$ is the joint probability of $X = x$ and $Y = y$.

• Conditional Entropy: Denoted as $H(X|Y)$, measures the uncertainty remaining in a random variable X given that we know the value of another random variable Y.

Explanation: Conditional entropy tells us how much uncertainty remains about X once we know Y. If X and Y are independent, knowing Y does not reduce the uncertainty about X, and conditional entropy equals $H(X)$. If Y provides complete information about X, conditional entropy will be zero.

Formula:

$$H(X|Y) = H(X, Y) - H(Y),$$

• Mutual Information (MI): Denoted as $I(X; Y)$, quantifies the amount of information that one random variable X contains about another random variable Y.

Explanation: Mutual information measures the reduction in uncertainty about one variable due to the knowledge of another. In machine learning, it is often used for feature

selection, as it helps identify which features share the most information with the target variable.

Formula:

$$I(X;Y) = H(X) - H(X|Y),$$

or equivalently,

$$I(X;Y) = \sum_{x,y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}.$$

• Kullback-Leibler Divergence (KL Divergence): Denoted as $D_{KL}(P||Q)$, measures how one probability distribution P differs from a second probability distribution Q.

Explanation: KL divergence is used to compare how much information is lost when using an approximate distribution Q to represent a true distribution P. In machine learning, KL divergence is widely used in variational inference and other optimization techniques.

Formula:

$$D_{KL}(P||Q) = \sum_{x} P(x) \log_2 \frac{P(x)}{Q(x)}.$$

• Cross-Entropy: It measures the difference between two probability distributions P and Q by quantifying the total entropy in P plus the extra entropy introduced by approximating P with Q.

Explanation: Cross-entropy is commonly used as a loss function in classification tasks, where the goal is to minimize the difference between the true probability distribution P (ground truth) and the predicted distribution Q (model predictions).

Formula:

$$H(P,Q) = -\sum_{x} P(x) \log_2 Q(x).$$

• Information Bottleneck: The information bottleneck method is a technique that seeks to compress a signal by extracting only the most relevant information for a particular task while discarding irrelevant details.

Explanation: In machine learning, this method is used to reduce the complexity of models by retaining only the essential features of the input data. It helps in constructing simpler, more interpretable models and is often applied in deep learning to control overfitting.

These core terminologies provide the foundation for understanding and applying information theory to machine learning. Through concepts like entropy, mutual information,

and KL divergence, information theory helps quantify and manage uncertainty, relevance, and efficiency in machine learning systems.

Chapter 3

Conditional Mutual Information Constrained Deep Learning for Classification

This chapter aims to borrows idea from information theory to introduce a novel metric to evaluate deep neural networks (DNNs). This metric is then incorporated into the conventional deep learning loss function to improve classification accuracy.

Particularly, the concepts of conditional mutual information (CMI) and normalized conditional mutual information (NCMI) are introduced to measure the concentration and separation performance of a classification DNN in the output probability distribution space of the DNN, where CMI and the ratio between CMI and NCMI represent the intra-class concentration and inter-class separation of the DNN, respectively. By using NCMI to evaluate popular DNNs pretrained over CIFAR-100 and ImageNet in the literature, it is shown that their validation accuracies are more or less inversely proportional to their NCMI values. Based on this observation, the standard deep learning (DL) framework is further modified to minimize the standard cross entropy function subject to an NCMI constraint, yielding CMI constrained deep learning (CMIC-DL). A novel alternating learning algorithm is proposed to solve such a constrained optimization problem. Extensive experiment results show that DNNs trained within CMIC-DL outperform the state-of-the-art models trained within the standard DL and other loss functions in the literature in terms of both accuracy and robustness against adversarial attacks. In addition, visualizing the evolution of learning process through the lens of CMI and NCMI is also advocated.

The conference version of this chapter is available in [222], while the journal version

can be found in [221].

3.1 Introduction

In recent years, deep neural networks (DNNs) have been applied in a wide range of applications, revolutionizing fields like computer vision, natural language processing, and speech recognition [64, 127]. Typically, a DNN consists of cascaded non-linear layers that progressively produce multi-layers of representations with increasing levels of abstraction, starting from raw input data and ending with a predicted output label. The success of DNNs is largely attributable to their ability to learn these multi-layers of representations as features from the raw data through a deep learning (DL) process.

Putting its neural architecture aside, a classification DNN is, mathematically, a mapping from raw data $x \in \mathbb{R}^d$ to a probability distribution q_x over the set of class labels, predicting an output label $\hat{y}$ with probability $q_x(\hat{y})$. Given a pair of random variables (X, Y), the distribution of which governs either a training set or testing set, where $X \in \mathbb{R}^d$ represents the raw data and Y is the ground truth label of X, the prediction performance of the DNN is often measured by its error rate

$$\epsilon = \Pr\{\hat{Y} \neq Y\},$$

where $\hat{Y}$ is the label predicted by the DNN with probability $q_X(\hat{Y})$ in response to the input X. The accuracy of the DNN is equal to $1 - \epsilon$. The error rate is further upper bounded by the average of the cross entropy between the conditional distribution of Y given X and q_X (see Section 3.2). To have better prediction performance, a DL process is then applied to minimize the error rate ϵ or its cross entropy upper bound [64, 127].

Although the error rate of a DNN is its most important performance as far as its prediction is concerned, focusing entirely on the error rate is not enough, and can actually lead to several problems. First, the error rate of a DNN depends not only on the DNN itself, but also on the governing joint distribution of (X, Y). When a DNN has a small error rate for one governing joint distribution of (X, Y), it does not necessarily imply that it would have a small error rate for another governing joint distribution of (X, Y), especially when two distributions are quite different. This is essentially related to the well-known overfitting and robustness problems [22, 64, 66, 143]. Second, even when a DNN works well across different governing distributions of (X, Y), it remains a block box to us, especially when its architecture is huge. We don't know why it works and how it works. Its error rate does not reveal any useful information about the intrinsic mapping structure such as the

intra-class concentration and inter-class separation of the DNN in its output probability distribution space.

To gain deep insights into the intrinsic mapping structure of a DNN as a mapping from $x \in \mathbb{R}^d$ to q_x, in this chapter we introduce information quantities from information theory [40] to measure intra-class concentration and inter-class separation of the DNN. Specifically, we propose to use the conditional mutual information (CMI) $I(X;\hat{Y}|Y)$ between X and $\hat{Y}$ given Y as the measure for the intra-class concentration of the DNN as a mapping $x \in \mathbb{R}^d \to q_x$. For each class label y, consider all input samples x with its ground truth label being y. The DNN then maps this subset of input samples x into a cluster of distributions q_x in the output probability distribution space. The conditional mutual information $I(X;\hat{Y}|Y=y)$ between X and $\hat{Y}$ given $Y=y$ tells how all these probability distributions q_x in the cluster are concentrated around its "centroid", the conditional probability distribution $P_{\hat{Y}|Y}(\cdot|y)$. The smaller $I(X;\hat{Y}|Y=y)$ is, the more concentrated all distributions q_x in the cluster corresponding to $Y=y$ are around its centroid. We further introduce another information quantity (see Section 3.2) to measure the inter-class separation of the DNN as a mapping $x \in \mathbb{R}^d \to q_x$. Define the ratio between $I(X;\hat{Y}|Y)$ and the inter-class separation as the normalized conditional mutual information (NCMI) between X and $\hat{Y}$ given Y. One may interpret CMI and NCMI as certain mapping structure traits of the DNN. Then in addition to its error rate, the DNN can also be evaluated in terms of its CMI and NCMI.

Equipped with our new concepts of CMI and NCMI, we further evaluate popular DNNs pretrained in the literature over CIFAR-100 and ImageNet in terms of their respective CMI and NCMI. It turns out that their validation accuracies over the ImageNet (CIFAR-100, resp.) validation data set are more or less inversely proportional to their NCMI values. In other words, even though these DNNs have different architectures and different sizes, their error rates and NCMI values have more or less a positive linear relationship. Indeed, the correlation between the error rate and NCMI is above 0.98 for CIFAR-100 and above 0.99 for the ImageNet dataset. This implies that given a DNN architecture, one may be able to further improve the effectiveness of DL by simultaneously minimizing the error rate (or cross entropy upper bound) and NCMI of the DNN during the learning process, where the error rate and NCMI represent the prediction performance and the concentration/separation mapping structure performance of the DNN, respectively. This in turn motivates us to modify the standard DL framework to minimize the standard cross entropy function subject to an NCMI constraint, yielding CMI constrained deep learning (CMIC-DL). To this end, a novel alternating learning algorithm is further proposed to solve such a constrained optimization problem. Extensive experiment results show that DNNs trained within CMIC-DL outperform the state-of-the-art models trained within the standard DL and other loss functions in the literature in terms of both accuracy and

robustness against adversarial attacks.

The remainder of this chapter is organized as follows. In Section 3.2, we formally introduce the concepts of CMI and NCMI to measure intra-class concentration and inter-class separation structure performance of a DNN when it is viewed as a mapping from $x \in \mathbb{R}^d$ to q_x. In Section 3.3, we use NCMI to evaluate and compare popular DNNs pretrained in the literature over ImageNet and CIFAR-100. These DNNs have different architectures and different sizes. Section 3.4 is devoted to the full development of CMIC-DL. In Section 6.5, extensive experiment results are presented and compared with the prior art in the literature; visualizing the evolution of learning process through the lens of CMI and NCMI is also advocated. Finally, conclusions are drawn along with some open problems in Section 3.8.

3.2 Performance of DNNs: Concentration and Separation

A DNN can be described either by its neural architecture along with its connection weights, the number of which can be in billions, or by its mathematical mapping from $x \in \mathbb{R}^d$ to q_x. Both perspectives are useful. In this and next sections, we will take the second perspective and regard a DNN simply as a mapping $x \in \mathbb{R}^d \to q_x$. Before formally introducing CMI and NCMI, we set up notation to be used throughout the this chapter.

3.2.1 Notation Used in This Chapter

For a positive integer K, let $[K] \triangleq \{1, \ldots, K\}$. Assume that there are C class labels with $[C]$ as the set of class labels. Let $\mathcal{P}([C])$ denote the set of all probability distributions over $[C]$. For any two probability distributions $q_1, q_2 \in \mathcal{P}([C])$, the cross entropy (CE) of q_1 and q_2 is defined as

$$H(q_1, q_2) = \sum_{i=1}^{C} -q_1(i) \ln q_2(i), \tag{3.1}$$

where ln denotes the logarithm with base e; the Kullback–Leibler (KL) divergence (or relative entropy) between q_1 and q_2 is defined as

$$D(q_1||q_2) = \sum_{i=1}^{C} q_1(i) \ln \frac{q_1(i)}{q_2(i)}. \tag{3.2}$$

For any $y \in [C]$ and $q \in \mathcal{P}([C])$, write the CE of the one-hot probability distribution corresponding to y and q as

$$H(y, q) = -\ln q(y). \tag{3.3}$$

For any random vector (X, Y), denote its joint probability distribution by $P_{XY}(x, y)$ or simply $P(x, y)$ whenever there is no ambiguity, its marginal distributions by $P_X(x)$ and $P_Y(y)$ or simply $P(x)$ and $P(y)$ whenever there is no ambiguity, respectively, the conditional distribution of Y given $X = x$ by $P_{Y|X}(\cdot|x)$, and the conditional distribution of X given $Y = y$ by $P_{X|Y}(\cdot|y)$. Given a DNN: $x \in \mathbb{R}^d \to q_x$, let θ denote its weight vector consisting of all its connection weights; whenever there is no ambiguity, we also write q_x as $q_{x,\theta}$, and $q_x(y)$ as $q(y|x, \theta)$ for any $y \in [C]$. If Y takes values in $[C]$, then for any x, $H(P_{Y|X}(\cdot|x), q_x)$ is well defined and equal to the CE of the distributions $P_{Y|X}(\cdot|x)$ and q_x.

3.2.2 Error Rate

Fix a DNN: $x \in \mathbb{R}^d \to q_x$. As in Section 6.1, let (X, Y) be a pair of random variables representing the raw input data and the corresponding ground truth label; let $\hat{Y}$ be the label predicted by the DNN with probability $q_X(\hat{Y})$ in response to the input X, that is, for any input $x \in \mathbb{R}^d$ and any $\hat{y} \in [C]$

$$P_{\hat{Y}|X}(\hat{y}|x) = q_x(\hat{y}) = q(\hat{y}|x, \theta). \tag{3.4}$$

Note that $Y \to X \to \hat{Y}$ forms a Markov chain in the indicated order. Therefore, given $X = x$, Y and $\hat{Y}$ are conditionally independent.

The error rate of the DNN for (X, Y) is equal to

$$\epsilon = \Pr\{\hat{Y} \neq Y\},$$

which can be upper bounded by the average of the CE of the conditional probability distribution of Y given X, $P_{Y|X}(\cdot|X)$, and q_X, as shown in the following theorem.

Theorem 1. *For any DNN: $x \in \mathbb{R}^d \to q_x$ and any (X, Y),*

$$\begin{aligned} \epsilon &\leq \mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_X) \right] \\ &= \sum_x P_X(x) H(P_{Y|X}(\cdot|x), q_x), \end{aligned} \tag{3.5}$$

where $\mathbf{E}_X$ denotes the expectation with respect to X[1].

[1]Throughout this chapter, the summation over x will be replaced by the corresponding integral if X is continuous.

Proof. Let $I_{\{\hat{Y}\neq Y\}}$ denote the indicator function of the event $\{\hat{Y}\neq Y\}$. Then

$$\begin{aligned}
\epsilon &= \Pr\{\hat{Y}\neq Y\} \\
&= \mathbf{E}[I_{\{\hat{Y}\neq Y\}}] \\
&= \mathbf{E}_X\left[\mathbf{E}[I_{\{\hat{Y}\neq Y\}}|X]\right] \\
&= \mathbf{E}_X\left[1-\sum_{i=1}^{C} P_{Y|X}(i|X)P_{\hat{Y}|X}(i|X)\right] \qquad (3.6)\\
&= \mathbf{E}_X\left[1-\sum_{i=1}^{C} P_{Y|X}(i|X)q_X(i)\right] \qquad (3.7)\\
&= \mathbf{E}_X\left[\sum_{i=1}^{C} P_{Y|X}(i|X)(1-q_X(i))\right] \\
&\leq \mathbf{E}_X\left[\sum_{i=1}^{C} -P_{Y|X}(i|X)\ln q_X(i)\right] \qquad (3.8)\\
&= \mathbf{E}_X\left[H(P_{Y|X}(\cdot|X),q_X)\right], \qquad (3.9)
\end{aligned}$$

where (3.6) follows from the fact that Y and $\hat{Y}$ are conditionally independent given X, (3.7) is attributable to (3.4), and (3.8) is due to the inequality $\ln z \leq z-1$ for any $z>0$. This completes the proof of Theorem 3. □

Given $X=x$, what happens if the DNN outputs instead the top one label $\hat{Y}^*$

$$\hat{Y}^* = \arg\max_{i\in[C]} q_x(i)?$$

In this case, the error rate of the DNN for (X,Y) is equal to

$$\epsilon^* = \Pr\{\hat{Y}^*\neq Y\},$$

which can also be upper bounded in terms of $\mathbf{E}_X\left[H(P_{Y|X}(\cdot|X),q_X)\right]$.

Corollary 1. *For any DNN: $x\in\mathbb{R}^d\to q_x$ and any (X,Y),*

$$\epsilon^* \leq C\epsilon \leq C\mathbf{E}_X\left[H(P_{Y|X}(\cdot|X),q_X)\right]. \qquad (3.10)$$

Proof.

$$\begin{aligned}
\epsilon^* &= \Pr\{\hat{Y}^* \neq Y\} \\
&= \mathbf{E}_X\left[1 - P_{Y|X}(\hat{Y}^*|X)\right] \\
&\leq C\mathbf{E}_X\left[q_X(\hat{Y}^*)(1 - P_{Y|X}(\hat{Y}^*|X))\right] \qquad (3.11) \\
&\leq C\mathbf{E}_X\left[\sum_{i=1}^{C} q_X(i)(1 - P_{Y|X}(i|X))\right] \\
&= C\mathbf{E}_X\left[1 - \sum_{i=1}^{C} P_{Y|X}(i|X)q_X(i)\right] \\
&= C\epsilon \leq C\mathbf{E}_X\left[H(P_{Y|X}(\cdot|X), q_X)\right], \qquad (3.12)
\end{aligned}$$

where (3.11) follows from the fact that $q_X(\hat{Y}^*) \geq 1/C$, and (3.12) is due to (3.7) and (3.9). □

In view of Theorem 3 and Corollary 1, no matter which form of error rate ϵ or ϵ^* is used, minimizing the average of the cross entropy $\mathbf{E}_X[H(P_{Y|X}(\cdot|X), q_X)]$ would have an effect to reduce ϵ and ϵ^*. This provides mathematical justifications for the use of the average of the cross entropy $\mathbf{E}_X[H(P_{Y|X}(\cdot|X), q_X)]$ as an objective function or a major component thereof in DL and knowledge distillation, where $P_{Y|X}$ is approximated by the one-hot probability vector corresponding to Y in DL [64, 127], and by the output probability distribution of the teacher in knowledge distillation [87, 149, 245].

3.2.3 Concentration

The error rates ϵ and ϵ^* of the DNN: $x \in \mathbb{R}^d \to q_x$ for (X, Y) do not provide any useful information on the intrinsic mapping structure of the DNN in the probability distribution space $\mathcal{P}([C])$. Two important mapping structure properties the DNN: $x \in \mathbb{R}^d \to q_x$ possesses, are its intra-class concentration and inter-class separation in the space $\mathcal{P}([C])$. In this and next subsections, we formally introduce information quantities to quantify these two mapping structure properties, respectively.

Visualize the DNN: $x \in \mathbb{R}^d \to q_x$ according to Fig. 3.1. Given $Y = y$, $y \in [C]$, the input data X is conditionally distributed according to the conditional distribution $P_{X|Y}(\cdot|y)$ and then mapped into q_X, a random point in the space $\mathcal{P}([C])$. The instances (or realizations)

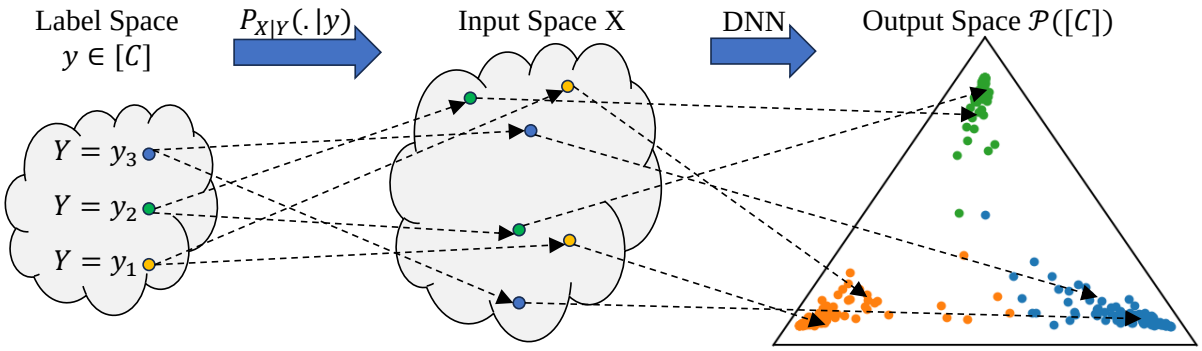


Figure 3.1: The mappings from the label space to the input space, and from the input space to the output space of a DNN. Here caricatures are used to depict label and input spaces, where each of the three instances in the label space is mapped to two instances in input space according to $P_{X|Y}(\cdot|y_i)$, for $i \in \{1, 2, 3\}$. On the other hand, the figure for the output space is obtained from a real example, where for the ResNet56 model trained on CIFAR-100, the output probability vectors corresponding to all validation sample instances from three randomly-picked classes are projected over the two-dimensional probability simplex.

of this random point q_X form a cluster corresponding to y in the space $\mathcal{P}([C])$. The centroid of this cluster is the average of q_X with respect to the conditional distribution $P_{X|Y}(\cdot|y)$

$$s_y \triangleq \sum_x P_{X|Y}(x|y) q_x \tag{3.13}$$

$$= \sum_x P_{X|Y}(x|y) P_{\hat{Y}|X}(\cdot|x) = P_{\hat{Y}|Y}(\cdot|y), \tag{3.14}$$

where (4.6) is due to (3.4) and the Markov chain $Y \to X \to \hat{Y}$. Now measure the "distance" between each q_x in the cluster and the centroid s_y of the cluster by their KL divergence $D(q_x||s_y)$. Averaging the KL divergence $D(q_x||s_y)$ within the cluster with respect to the

conditional distribution $P_{X|Y}(\cdot|y)$, we get

$$\begin{aligned}
&\mathbf{E}\left[D(q_X||s_y)|Y=y\right] \\
&= \sum_x P_{X|Y}(x|y)\left[\sum_{i=1}^{C} q_x(i)\ln\frac{q_x(i)}{s_y(i)}\right] \\
&= \sum_x P_{X|Y}(x|y)\left[\sum_{i=1}^{C} P_{\hat{Y}|X}(i|x)\ln\frac{P_{\hat{Y}|X}(i|x)}{P_{\hat{Y}|Y}(i|y)}\right] &(3.15)\\
&= \sum_x P_{X|Y}(x|y)\left[\sum_{i=1}^{C} P_{\hat{Y}|XY}(i|x,y)\ln\frac{P_{\hat{Y}|XY}(i|x,y)}{P_{\hat{Y}|Y}(i|y)}\right] &(3.16)\\
&= I(X;\hat{Y}|y), &(3.17)
\end{aligned}$$

where $I(X;\hat{Y}|y)$ is the conditional mutual information between X and $\hat{Y}$ given $Y=y$. (Please refer to [40] for the notions of mutual information and conditional mutual information.) In the above, (3.15) is due to (3.4) and (4.6); (3.16) follows from the Markov chain $Y \to X \to \hat{Y}$.

The quantity $I(X;\hat{Y}|y)$ quantifies the concentration of the cluster corresponding to $Y=y$. Averaging $I(X;\hat{Y}|y)$ across all clusters with respect to the distribution $P_Y(y)$, we get the conditional mutual information $I(X;\hat{Y}|Y)$ between X and $\hat{Y}$ given Y:

$$\begin{aligned}
I(X;\hat{Y}|Y) &= \sum_{y\in[C]} P_Y(y) I(X;\hat{Y}|y) \\
&= \mathbf{E}\left[\mathbf{E}\left[D(q_X||s_Y)|Y\right]\right] = \mathbf{E}\left[D(q_X||s_Y)\right] &(3.18)\\
&= \sum_y \sum_x P(x,y)\left[\sum_{i=1}^{C} q_x(i)\ln\frac{q_x(i)}{s_y(i)}\right]. &(3.19)
\end{aligned}$$

The CMI $I(X;\hat{Y}|Y)$ can then be regarded as a measure for the intra-class concentration of the DNN: $x \in \mathbb{R}^d \to q_x$ for (X,Y).

When the joint distribution $P(x,y)$ is unknown in practice, it can be approximated by its empirical distribution from a data sample $\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$. In the case, the averages in (3.13) and (3.18) become the respective sample means; in particular, for each $y \in [C]$,

$$s_y = \frac{1}{n_y}\sum_{(x_j,y_j):y_j=y} q_{x_j}, \tag{3.20}$$

where

$$n_y = |\{j : y_j = y, 1 \le j \le n\}|, \tag{3.21}$$

and $|S|$ denotes the cardinality of a set S, and

$$I(X;\hat{Y}|Y) = \frac{1}{n}\sum_{(x_j,y_j)} D(q_{x_j}||s_{y_j}) = \frac{1}{n}\sum_{j=1}^{n} D(q_{x_j}||s_{y_j}). \tag{3.22}$$

3.2.4 Separation and NCMI

Let (U,V) be a pair of random variables independent of (X,Y), and having the same joint distribution as that of (X,Y). With reference to Fig. 3.1, we interpret (X,Y) and (U,V) as two independent samples from a training (or testing) dataset, and define the following information quantity[2]

$$\Gamma = \mathbf{E}\left[I_{\{Y\neq V\}} H(q_X, q_U)\right], \tag{3.23}$$

and use Γ as a measure for the inter-class separation of the DNN: $x \in \mathbb{R}^d \to q_x$. It is clear that the larger Γ is, the further apart different clusters are from each other on average.

Ideally, we want $I(X;\hat{Y}|Y)$ to be small while keeping Γ large. This leads us to consider the ratio between $I(X;\hat{Y}|Y)$ and Γ:

$$\hat{I}(X;\hat{Y}|Y) \triangleq \frac{I(X;\hat{Y}|Y)}{\Gamma}. \tag{3.24}$$

We call $\hat{I}(X;\hat{Y}|Y)$ the normalized conditional mutual information between X and $\hat{Y}$ given Y.

Once again, when the joint distribution $P(x,y)$ is unknown, it can be approximated by its empirical distribution from a data sample $\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$. In parallel with (3.22), Γ can be computed in this case as follows:

$$\Gamma = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} I_{\{y_j\neq y_k\}} H(q_{x_j}, q_{x_k}), \tag{3.25}$$

from which and (3.22), $\hat{I}(X;\hat{Y}|Y)$ can be computed accordingly.

[2]Other information quantities can also be defined and used as a measure for the inter-class separation of the DNN: $x \in \mathbb{R}^d \to q_x$, which will be explored in Section 3.6. Although they are more or less equivalent, the information quantity Γ defined here is more convenient for the selection of hyper parameters in our proposed CMIC-DL. Unless indicated otherwise, the expectation $\mathbf{E}$ is with respect to the distribution or joint distribution of random variables appearing inside the brackets of $\mathbf{E}$.

3.2.5 Related Works

The extracted deep features from a trained Convolutional Neural Network (CNN) are generally expected to exhibit high intra-class similarity and low inter-class similarity, which are essential for effective classification and representation learning [137, 207, 211]. However, raw data often suffer from the opposite characteristics: low intra-class similarity and relatively high inter-class similarity [41, 158]. Addressing these limitations has led to the development of various methods to enhance the discriminative power of deep features, with metric learning emerging as a prominent alternative [135, 145, 179, 210].

One widely studied approach is the center loss (CL) algorithm and its variants, which aim to minimize the intra-class distance by clustering feature vectors from the same class around a learnable class center [137, 203, 207, 211]. From the perspective of prototype learning, these class centers can be viewed as prototypes representing each class [224, 238, 239]. However, the update rules in the original CL framework [211] limit the ability to directly learn these prototypes from data, as highlighted in [238]. Additionally, the center loss framework can sometimes lead to instability during training [207].

To address these challenges, convolutional prototype learning (CPL) was introduced, which leverages a distance-based cross-entropy loss (DCE) to automatically learn prototypes from data [238]. CPL enhances its robustness by incorporating a prototype loss (PL), resulting in generalized convolutional prototype learning (GCPL) [238]. The integration of CNNs with GCPL forms convolutional prototype networks (CPNs) [224], which adopt a unified approach to distance-based representation learning. In GCPL, the classification loss (e.g., softmax or DCE) focuses on separating features across classes, while the distance-based penalization (e.g., CL or PL) reduces intra-class variance. This combination synergistically improves both intra-class compactness and inter-class separability. Notably, when features and prototypes are normalized, GCPL behaves as a linear classifier, maximizing the cosine similarity between feature vectors and their corresponding prototypes [239].

An alternative approach involves directly enhancing the cosine similarity between feature vectors and class weight vectors [137, 203, 244]. One widely used technique is the L_2-normalization of feature vectors, which eliminates the influence of magnitude variations and integrates the capability to learn compact representations into the softmax loss. This normalization explicitly reduces correlations among softmax weights, thereby improving inter-class separation. Numerous frameworks, such as the orthogonality constraint loss (OCL) [207], uniform loss [55], and regularization-based approaches [240], build upon this principle. OCL focuses on orthogonalizing the softmax weights, while the uniform loss aims to distribute the weights uniformly.

Table 3.1: CMI, Γ, NCMI, and error rate values, over the validation set, of the models trained with **CE** loss on CIFAR-100.

Model	CMI	Γ	NCMI	Error rate ϵ^*	Model	CMI	Γ	NCMI	Error rate ϵ^*
ResNet18	0.837	11.389	0.073	0.239	WRN16_1	1.125	11.001	0.102	0.414
ResNet34	0.794	11.551	0.068	0.228	WRN40_1	0.993	12.189	0.0814	0.324
ResNet50	0.825	12.345	0.066	0.213	WRN16_2	1.017	12.907	0.079	0.312
ResNet101	0.774	12.151	0.064	0.209	WRN40_2	0.934	12.716	0.073	0.264
ResNet152	0.764	12.098	0.063	0.208	Xception	0.759	11.270	0.067	0.226
ResNext50	0.792	11.967	0.066	0.213	MobileNet-0.25	1.166	11.376	0.103	0.405
ResNext101	0.774	12.151	0.065	0.206	MobileNet-0.5	1.140	11.816	0.096	0.356
ResNext152	0.764	12.098	0.063	0.204	MobileNet-1	1.036	11.729	0.088	0.327
SqueezeNet	1.084	12.377	0.087	0.307	DenseNet121	0.752	11.703	0.064	0.209

However, L_2-normalization as implemented in methods like [137] and [244] only utilizes the unit direction of feature vectors, leaving their magnitudes unnormalized. This results in substantial variations in feature magnitudes, which can undermine intra-class compactness [136]. Consequently, further enhancements are needed to address this gap. While L_2-normalization introduces normalization factors that improve feature alignment, it can also complicate the computation of gradient vectors, adding complexity to optimization objectives. This trade-off highlights the importance of balancing compactness, separability, and computational efficiency in designing loss functions for discriminative feature learning.

In contrast, in this this chapter we investigate the intra-class concentration and inter-class separation of a DNN in its output probability distribution space $\mathcal{P}([C])$, where the DNN is viewed as a mapping from $x \in \mathbb{R}^d$ to q_x. This perspective allows us to introduce information quantities, CMI, Γ, and NCMI, to quantify the intra-class concentration and inter-class separation of each DNN. In addition, our introduced CMI and NCMI can also be regarded as additional performance metrics for any DNN, which are in parallel with the error rate performance metric, are independent of any learning process, and represent mapping structure properties of a DNN. As additional performance metrics, they can be used to evaluate and compare different DNNs regardless of the architectures and sizes of DNNs.

In the next section, CMI and NCMI will be used to evaluate and compare popular DNNs in the literature trained over CIFAR-100[3].

[3]Similar analyses over ImageNet dataset are presented in Section 3.7.

3.3 NCMI Vs. Accuracy

The popular DNNs we selected for evaluation according to their respective CMI and NCMI are ResNet-$\{18, 34, 50, 101, 152\}$ [84], ResNext-$\{50, 101, 152\}$ [216], SqueezeNet [97], Wide-ResNet-$\{40_2, 16_2, 40_1, 16_1\}$ [231], Xception [30], MobileNet-$\{1, 0.5, 0.25\}$ [91], and DenseNet121 [94].

Table 3.1 lists the values of CMI, Γ, and NCMI of the selected DNNs, which are calculated, according to (3.22), (3.25), and (3.24), over the CIFAR-100 validation set, along with their respective error rate ϵ^*. In Table 3.1, models within the same family are listed in the order of increasing model size. As the model size within the same family increases, two observations can be made from Table 3.1: (1) there is no clear increasing or decreasing trend in either the CMI or Γ value; (2) however, the NCMI value always decreases.

Even more interesting is the relationship between the NCMI and error rate ϵ^*. Within the same family, as the NCMI value decreases, so does the error rate ϵ^*. This relationship is also more or less valid in general even across all models evaluated. To make the relationship between the NCMI and error rate ϵ^* more transparent, Figure 3.2 illustrates the relationship graphically. From Figure 3.2, it seems that the NCMI and error rate ϵ^* have a positive linear relationship; indeed, the Pearson correlation coefficient ρ [36] between them is $\rho = 0.9823$, strongly supporting the former statement[4]. As such, the NCMI value of a DNN can be used to gauge the prediction performance of the DNN. Additionally, we note that such a trend cannot be observed using CMI and Γ values even in the same family.

To conclude this section, let us draw some analogies. If a DNN is analogized with a student, then the error rate and NCMI of the DNN can be analogized with the testing score of the student in an exam and certain trait of the student, respectively. In a way similar to using the trait of the student to predict the student's testing performance, one can also use the NCMI value of the DNN to predict the DNN's testing performance.

3.4 CMIC Deep Learning

The discussions in the above section suggest a new way of learning. In the learning process, instead of minimizing the average of cross entropy $\mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_X) \right]$ alone, one also needs to look after the NCMI $\hat{I}(X; \hat{Y}|Y)$. This leads to a new form of learning framework dubbed CMI constrained deep learning (CMIC-DL), which is described next.

[4]For ImageNet dataset, $\rho = 0.9929$ (please refer to Section 3.7).

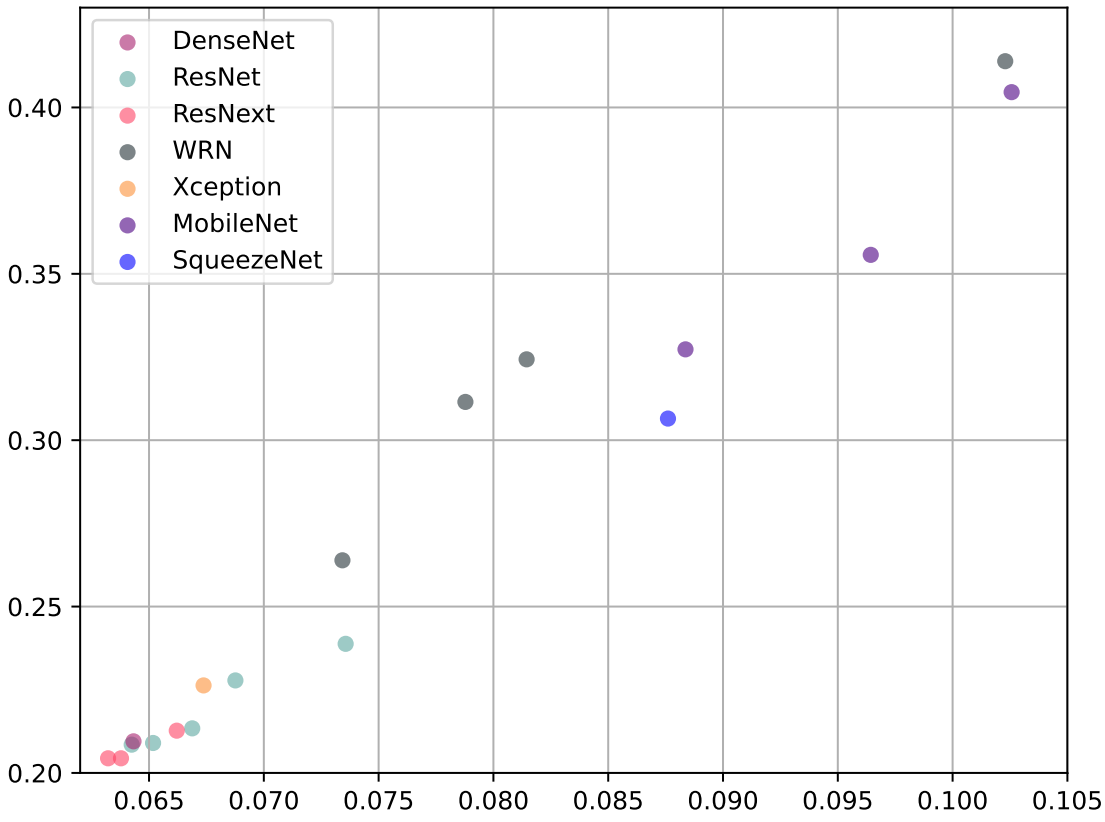


Figure 3.2: The error rate ϵ^* vs NCMI value over the validation set of popular pre-trained models on CIFAR-100 dataset.

3.4.1 Optimization Problem Formulation

In CMIC-DL, the optimization problem to be solved is as follows:

$$\begin{aligned} &\min_{\theta} \mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_{X,\theta}) \right] \\ &\text{s.t.} \quad \hat{I}(X;\hat{Y}|Y) = r, \end{aligned} \tag{3.26}$$

where r is a positive constant. By interpreting $\hat{I}(X;\hat{Y}|Y)$ as a rate, and $\mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_{X,\theta}) \right]$ as a distortion, the above optimization problem resembles the rate distortion problem in information theory [12,40,223]. By rewriting the constraint in (3.26), and using the Lagrange multiplier method, the constrained optimization problem in (3.26) could be formulated as

the following unconstrained one

$$\min_{\theta} \mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_{X,\theta})\right] + \lambda I(X;\hat{Y}|Y) - \beta \mathbf{E}\left[I_{\{Y\neq V\}} H(q_{X,\theta}, q_{U,\theta})\right], \tag{3.27}$$

where $\lambda > 0$ is a scalar, and $\beta = \lambda r$.

Note that the objective function in (3.27) is not amenable to parallel computation via GPU due to the dependency of $I(X;\hat{Y}|Y)$ on the centroid s_y of each cluster corresponding to $Y = y$ (see (3.19)). To overcome this, we first convert it into a double unconstrained minimization problem by introducing a dummy distribution $Q_y \in \mathcal{P}([C])$ for each $y \in [C]$, as shown in the following theorem.

Theorem 2. *For any $\lambda > 0$ and $\beta > 0$,*

$$\begin{aligned}
&\min_{\theta} \Big\{\mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_{X,\theta})\right] \\
&\quad + \lambda I(X;\hat{Y}|Y) - \beta \mathbf{E}\left[I_{\{Y\neq V\}} H(q_{X,\theta}, q_{U,\theta})\right]\Big\} \\
&= \min_{\theta} \min_{\{Q_c\}_{c\in[C]}} \Big\{\mathbf{E}_X[H(P_{Y|X}(\cdot|X), q_{X,\theta})] \\
&\quad + \lambda \mathbf{E}_{XY}[D(q_{X,\theta}||Q_Y)] - \beta \mathbf{E}[I_{\{Y\neq V\}} H(q_{X,\theta}, q_{U,\theta})]\Big\}.
\end{aligned} \tag{3.28}$$

Proof. Since $\lambda > 0$ and $\beta > 0$, it suffices to show that

$$I(X;\hat{Y}|Y) = \min_{\{Q_c\}_{c\in[C]}} \mathbf{E}[D(q_{X,\theta}||Q_Y)]. \tag{3.29}$$

To this end, we apply (3.19) to get the following:

$$\begin{aligned}
I(X;\hat{Y}|Y) &= \sum_y \sum_x P(x,y) \left[\sum_{i=1}^{C} q_{x,\theta}(i) \left[\ln \frac{q_{x,\theta}(i)}{Q_y(i)} + \ln \frac{Q_y(i)}{s_y(i)}\right]\right] \\
&= \sum_y \sum_x P(x,y) D(q_{x,\theta}||Q_y) + \sum_y \sum_x P(x,y) \times \left[\sum_{i=1}^{C} q_{x,\theta}(i) \ln \frac{Q_y(i)}{s_y(i)}\right] \\
&= \mathbf{E}[D(q_{X,\theta}||Q_Y)] + \sum_y P_Y(y) \left[\sum_{i=1}^{C} \left[\sum_x P_{X|Y}(x|y) q_{x,\theta}(i)\right] \ln \frac{Q_y(i)}{s_y(i)}\right] \\
&= \mathbf{E}[D(q_{X,\theta}||Q_Y)] + \sum_y P_Y(y) \left[\sum_{i=1}^{C} s_y(i) \ln \frac{Q_y(i)}{s_y(i)}\right] \qquad (3.30) \\
&= \mathbf{E}[D(q_{X,\theta}||Q_Y)] - \mathbf{E}[D(s_Y||Q_Y)] \\
&\leq \mathbf{E}[D(q_{X,\theta}||Q_Y)], \qquad (3.31)
\end{aligned}$$

for any $Q_y \in \mathcal{P}([C]), y \in [C]$, where (3.30) is due to (3.13), and (3.31) follows from the nonnegativity of KL divergence. Thus

$$I(X;\hat{Y}|Y) \leq \min_{\{Q_c\}_{c\in[C]}} \mathbf{E}[D(q_{X,\theta}||Q_Y)]. \tag{3.32}$$

On the other hand, (3.31) becomes an equality whenever

$$Q_c = s_c, \forall c \in [C]. \tag{3.33}$$

This, together with (3.31), implies (3.29), and hence completes the proof of Theorem 2. □

Given θ and $\{Q_c\}_{c\in[C]}$, the objective function in (5.46) is now sample additive and can be computed in parallel via GPU. In particular, when the joint distribution $P(x, y)$ is unknown, it can be approximated by its empirical distribution from a data sample (such as a mini-batch in the DL process) $\mathcal{B} = \{(x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \cdots, (x_{i_m}, y_{i_m})\}$. In this case, all expectations in (5.46) are approximated by their respective sample means; with $P_{Y|X}(\cdot|x_{i_j})$ replaced by the one-hot probability distribution corresponding to y_{i_j}, the objective function in the double minimization (5.46) then becomes $J_{\mathcal{B}}\left(\lambda, \beta, \theta, \{Q_c\}_{c\in[C]}\right)$ shown in (5.51).

$$\begin{aligned} J_{\mathcal{B}}\left(\lambda, \beta, \theta, \{Q_c\}_{c\in[C]}\right) = &\frac{1}{|\mathcal{B}|}\sum_{(x,y)\in\mathcal{B}} H(y, q_{x,\theta}) + \lambda\frac{1}{|\mathcal{B}|}\sum_{(x,y)\in\mathcal{B}} D(q_{x,\theta}||Q_y) \\ &- \beta\frac{1}{|\mathcal{B}|^2}\sum_{(x,y),(u,v)\in\mathcal{B}} I_{\{y\neq v\}} H(q_{x,\theta}, q_{u,\theta}). \end{aligned} \tag{3.34}$$

3.4.2 Algorithm for Solving the Optimization in (5.46)

Having addressed how to approximate the objection function in the double minimization (5.46), we are now ready to present an algorithm for solving (5.46). In fact, by reformulating the single minimization problem as a double minimization problem, Theorem 2 lends us an alternating algorithm[5] that optimizes θ and $\{Q_c\}_{c\in[C]}$ alternatively to minimize the objective function in (5.46), given that the other is fixed.

Given $\{Q_c\}_{c\in[C]}$, θ can be updated using the same strategy as in the conventional DL through stochastic gradient descent iterations over mini-batches, where the training set is

[5]If the impact of the random mini-batch sampling and stochastic gradient descent is ignored, the alternating algorithm is guaranteed to converge in theory since given θ, the optimal $\{Q_c\}_{c\in[C]}$ can be found analytically via (3.35), although it may not converge to a global minimum.

divided into B mini-batches $\{\mathcal{B}_b\}_{b\in[B]}$ with each batch of size $|\mathcal{B}|$. Given θ, how is $\{Q_c\}_{c\in[C]}$ updated? This is where differences arise. In view of (3.13) and (3.33), the optimal $\{Q_c\}_{c\in[C]}$ given θ is equal to

$$Q_c = s_c = \sum_x P_{X|Y}(x|c) q_{x,\theta}, \tag{3.35}$$

for any $c \in [C]$. Therefore, to update $\{Q_c\}_{c\in[C]}$ given θ, we construct, at each iteration, C mini-batches $\{\mathfrak{B}_c\}_{c\in[C]}$ in the following manner: to make $\mathfrak{B}_c$, $\forall c \in [C]$, we randomly sample $|\mathfrak{B}_c|$ instances from the training samples whose ground truth labels are c. It then follows from (3.35) that for any $c \in [C]$, Q_c is updated as[6]

$$Q_c = \frac{\sum_{(x,c)\in\mathfrak{B}_c} q_{x,\theta}}{|\mathfrak{B}_c|}. \tag{3.36}$$

The procedure for solving the optimization problem (5.46) is now summarized in Algorithm 2, which is shown to converge in Section 6.5 (see Fig. 3.4), where we use $(\cdot)^t_{c,b}$ to indicate class c at the b-th batch updation during the t-th epoch. We also use $(\cdot)^t_{c,B}$ as $(\cdot)^t_c$ whenever necessary, and set $(\cdot)^t_{c,0} = (\cdot)^{t-1}_c$.

3.5 Experiment Results

To demonstrate the effectiveness of CMIC-DL and compare it with some state-of-the-art alternatives, we have conducted a series of experiments. Specifically, we have performed experiments on two popular image classification datasets, namely CIFAR-100 [121] and ImageNet [122]. In Subsections 5.6.4 and 5.6.3, we present their respective accuracy results. In Subsection 3.5.3, we present the CMIC-DL counterpart of Table 3.1 where the models therein have been trained via CMIC-DL. In Subsection 3.5.4, we explore how to visualize the concentration and separation of a DNN, which is made possible by viewing the DNN as a mapping from $x \in \mathbb{R}^d$ to q_x; using such a visualization method, the concentration and separation of ResNet-56 trained within our CMIC-DL framework are then compared with those of ResNet-56 trained within the standard DL framework.

In the literature, a deep learning process is typically analyzed experimentally through the evolution curve of its error rate. With our newly introduced performance metrics, CMI, Γ (separation), and NCMI, the learning process can also be analyzed through the evolution curves of CMI, Γ, and NCMI, which show interestingly how the mapping structure

[6] To update $\{Q_c\}_{c\in[C]}$, we may use momentum to make the updation more stable and less noisy.

Algorithm 1 The proposed alternating algorithm for solving the optimization problem in (5.46).

Input: The training set $\mathcal{T}$, all mini-batches $\{\mathcal{B}_b\}_{b\in[B]}$, number of epochs T, λ, and β.

Initialization:
Initialize θ^0 and $\{Q_c^0\}_{c\in[C]}$.
for $t = 1$ to T **do**
  for $b = 1$ to B **do**
    [*Updating* θ]:
    Fix $\{Q_{c,b-1}^t\}_{c\in[C]}$.
    Update θ_{b-1}^t to θ_b^t by using (stochastic) batch gradient descent over the loss function $J_{\mathcal{B}_b}\left(\lambda, \beta, \theta_{b-1}^t, \{Q_{c,b-1}^t\}_{c\in[C]}\right)$.
    [*Updating* $\{Q_c\}_{c\in[C]}$]:
    Fix θ_b^t.
    Construct mini-batches $\{\mathfrak{B}_c\}_{c\in[C]}$ from $\mathcal{T}$.
    Update $Q_{c,b-1}^t$ to $Q_{c,b}^t$, $\forall c \in [C]$, according to (3.36):

$$Q_{c,b}^t = \frac{\sum_{(x,c)\in\mathfrak{B}_c} q_{x,\theta_b^t}}{|\mathfrak{B}_c|}. \tag{3.37}$$

  end for
end for
return Model parameters θ^T.

in terms of CMI, Γ, and NCMI evolves over the course of learning process. In Subsection 4.5.2, we use ResNet-56 as an example, and illustrate and compare the evolution curves of CMI, Γ, NCMI, and error rate within our CMIC-DL framework vs within the standard DL framework. Lastly, in Subsection 3.5.7, we evaluate the robustness of models trained within our CMIC-DL framework against two different adversarial attacks, and show that in comparison with the standard DL, CMIC-DL improves the robustness of DNNs as well.

3.5.1 Experiments on CIFAR-100

CIFAR-100 dataset contains 50K training and 10K test colour images of size 32×32 labeled for 100 classes.

• **Models**: To show the effectiveness of CMIC-DL, we have conducted experiments on three different model architectural families. Specifically, we have selected (i) three models

from ResNet family [84], namely ResNet-$\{32, 56, 110\}$; (ii) VGG-13 from VGG family [185]; and (iii) Wide-ResNet-28-10 from Wide-ResNet family [231]. Note that these models are slightly different from those in Table 3.1 and used by benchmark methods.

• **Benchmarks**: We evaluate the performance of the DNNs trained via CMIC-DL against those trained by conventional cross entropy loss (CE), center loss (CL) [211] which promotes clustering the features, focal loss (FL) [134] which uses regularization, large-margin Gaussian Mixture (L-GM) loss [202] which imposes margin constraints, orthogonal projection loss (OPL) [171] which imposes orthogonality in the feature space, constrained center loss (CCL) [183] which constrain all class centers to the surface of a hypersphere, and hypersphere loss (HL) [204] which enhances the performance of CCL by ensuring that feature vectors from different categories are adequately separated.

• **Training settings**: We have deployed an SGD optimizer with a momentum of 0.9, a weight decay of 0.0005, and a batch size of 64. We have trained the models for 200 epochs, and adopted an initial learning rate of 0.1, which is further divided by 10 at the epochs 60, 120 and 160. We have reproduced the results of all the benchmark methods using their respective best hyper-parameters reported in their original papers. In addition, in Algorithm 2, we set $Q_c^0(i) = \frac{1}{C}$, for all $i, c \in [C]$, use $|\mathfrak{B}_c| = 8$, $\forall c \in [C]$, and update $Q_{c,b}^t$ using the momentum of 0.9999.

The results are summarized in Table 3.2. As seen, the models trained within our CMIC-DL framework outperform those trained by the benchmark methods. Importantly, the improvement is consistent across the models from different architectural families, showing that CMIC-DL can effectively train DNNs from different families. As a rule of thumb, compared to the CE method, CMIC-DL yields DNNs with almost 1.3% higher validation accuracy for the ResNet models.

Furthermore, in Table 3.3 we report the NCMI values $\hat{I}(X;\hat{Y}|Y)$, over the validation set, for the models we trained in Table 3.2, where we use the notation $\hat{I}_{\text{Loss}}$ to denote the NCMI value when the underlying DNN is trained using "Loss" method. As observed, $\hat{I}_{\text{CMIC}}$ has the smallest value compared to the other counterparts.

In addition, in Table 3.4, we report the λ^* and β^* values for which we obtained the best validation accuracies. As observed, the λ^* and β^* values are almost the same for all the models.

3.5.2 Experiments on ImageNet

ImageNet is a large-scale dataset used in visual recognition tasks, containing around 1.2 million training samples and 50,000 validation images.

Table 3.2: The validation accuracies (%) of different models trained by CMIC-DL and different benchmark methods over **CIFAR-100 dataset**, which are averaged over three different random seeds, and where **bold** and underlined values denote the best and second best results, respectively.

Loss	Res32	Res56	Res110	VGG13	WRN-28-10
CL	70.23	72.70	74.20	74.50	80.97
FL	<u>71.62</u>	73.20	74.35	74.53	81.24
LGM	71.50	73.06	<u>74.39</u>	74.57	81.29
OPL	71.03	72.60	73.98	74.11	81.12
CCL	71.07	73.15	74.19	<u>74.59</u>	81.47
HL	71.44	<u>73.33</u>	74.21	74.50	<u>81.55</u>
CE	70.90	72.40	73.79	73.77	80.93
CMIC	**72.24**	**73.66**	**75.08**	**74.62**	**81.63**

Table 3.3: The respective NCMI values, measured over the validation set, of the models trained in Table 3.2 via different benchmark methods. The values are averaged over thee different runs.

Loss	Res32	Res56	Res110	VGG13	WRN-28-10
$\hat{I}_{\text{CL}}$	0.057	0.045	0.0395	0.0395	0.0309
$\hat{I}_{\text{FL}}$	0.053	0.046	0.0393	0.0399	0.0312
$\hat{I}_{\text{LGM}}$	0.054	0.047	0.0390	0.0398	0.0310
$\hat{I}_{\text{OPL}}$	0.056	0.050	0.0397	0.0402	0.0314
$\hat{I}_{\text{CCL}}$	0.055	0.046	0.0395	0.0399	0.0308
$\hat{I}_{\text{HL}}$	0.054	0.043	0.0394	0.0401	0.0305
$\hat{I}_{\text{CE}}$	0.057	0.053	0.0402	0.0408	0.0317
$\hat{I}_{\text{CMIC}}$	0.051	0.042	0.0382	0.0392	0.0303

Table 3.4: **Hyper-parameters**, λ^* and β^* used for CMIC-DL in Table 3.2.

Params.	Res32	Res56	Res110	VGG13	WRN-28-10
(λ^*,β^*)	(0.7,0.4)	(0.7,0.4)	(0.7,0.2)	(0.8,0.3)	(0.7,0.4)

- **Models**: We have conducted experiments on two models from ResNet family, namely

Table 3.5: The validation accuracies (%) of different models trained by CMIC-DL and different benchmark methods on **ImageNet**.

Method	ResNet-18		ResNet-50	
	top-1	top-5	top-1	top-5
CE (Baseline)	69.91	89.08	76.15	92.87
OPL	70.27	89.60	76.32	93.09
CMIC	**70.47**	**89.96**	**76.52**	**93.44**

ResNet-18 and ResNet-50.

• **Benchmarks**: We evaluate the performance of CMIC-DL against CE and OPL.

• **Training settings**: We have deployed an SGD optimizer with a momentum of 0.9, a weight decay of 0.0001, and a batch size of 256. We have trained the models for 90 epochs, and adopted an initial learning rate of 0.1, which is further divided by 10 at the 30-th and 60-th epochs. In Algorithm 2, we set $\{Q_c^0(i)\}_{c\in[C]} = \frac{1}{C}$, for $i \in [C]$, use $|\mathfrak{B}_c| = 8$, $\forall c \in [C]$, and also update $Q_{c,b}^t$ using the momentum of 0.9999.

The top-$\{1, 5\}$ accuracies are reported in Table 3.5. As seen, in comparison with the CE method, CMIC-DL increases the top-1 validation accuracy for ResNet-18 and ResNet-50 by 0.56% and 0.37%, respectively. The improvement is also consistent for the top-5 validation accuracy.

The hyper parameters (λ^*, β^*) used in CMIC-DL for ResNet-18 and ResNet-50 are $(0.6, 0.1)$ and $(0.6, 0.2)$, respectively. The corresponding NCMI values are $\hat{I}_{\text{CE}} = 0.110$ and $\hat{I}_{\text{CMIC}} = 0.102$ for ResNet-18, and $\hat{I}_{\text{CE}} = 0.091$ and $\hat{I}_{\text{CMIC}} = 0.088$ for ResNet-50.

3.5.3 CMIC-DL Counterpart of Table 3.1

To provide a comparison with Table 3.1, we have trained the models therein via CMIC-DL on CIFAR-100, where the hyperparameters (λ, β) are selected from Table 3.4 with $(0.7, 0.4)$ for the families WRN and Xception, $(0.8, 0.3)$ for the family Resnext, and $(0.7, 0.2)$ for the rest of families. Their respective CMI, Γ, NCMI, and error rate values are reported in Table 3.6. In comparison with Table 3.1, it is clear that both the NCMI and error rate in Table 3.6 are smaller than their respective counterparts in Table 3.1. Again, since models within the same family share the same pair of hyperparameters (λ, β), the same-family-phenomenon for Table 3.1 remains valid for Table 3.4 as well—as the NCMI value decreases, so does the error rate ϵ^*. This relationship remains more or less valid for models across families when

Table 3.6: CMI, Γ, NCMI, and error rate values, over the validation set, of the models trained by **CMIC-DL** on CIFAR-100.

Model	CMI	Γ	NCMI	Error rate ϵ^*	Model	CMI	Γ	NCMI	Error rate ϵ^*
ResNet18	0.571	10.187	0.056	0.223	WRN16_1	0.852	10.449	0.0776	0.412
ResNet34	0.382	10.011	0.038	0.221	WRN40_1	0.602	10.184	0.059	0.321
ResNet50	0.324	9.811	0.0330	0.210	WRN16_2	0.852	11.131	0.077	0.308
ResNet101	0.325	9.870	0.0329	0.206	WRN40_2	0.490	10.023	0.049	0.250
ResNet152	0.302	9.797	0.0308	0.204	Xception	0.194	9.570	0.020	0.215
ResNext50	0.303	9.850	0.0307	0.203	MobileNet-0.25	1.018	10.830	0.094	0.395
ResNext101	0.289	9.763	0.030	0.201	MobileNet-0.5	0.549	9.826	0.055	0.343
ResNext152	0.281	9.765	0.029	0.200	MobileNet-1	0.484	9.729	0.050	0.321
SqueezeNet	0.987	11.599	0.085	0.299	DenseNet121	0.533	10.364	0.051	0.204

they share the same pair of hyperparameters (λ, β), save and except that the correlation coefficient ρ is now around 0.85.

3.5.4 Concentration and Separation Visualization

In this subsection, we explore how to visualize concentration and separation of a DNN. Consider the data set CIFAR-100. To visualize concentration and separation of a DNN in a dimension reduced probability space, we randomly select three class labels. Restrict ourselves to a subset consisting of all validation sample instances with labels from the three selected labels. Given a DNN, feed each validation sample instance from the subset into the DNN, keep only three logits corresponding to the three selected labels, and then convert these three logits into a 3 dimension probability vector through the softmax operation. Following these steps in the indicated order, the DNN then maps each validation sample instance from the subset into a 3 dimension probability vector. Further project the 3 dimension probability vector into the 2 dimension simplex. Then the concentration and separation properties of the DNN for the three selected classes can be more or less visualized through the projected 2 dimension simplex.

Using the above visualization method, Fig. 3.3 compares the concentration and separation properties of ResNet-56 trained within our CMIC-DL framework with those of ResNet-56 trained within the standard CE framework. From Fig. 3.3, it is clear that the three clusters in the case CMIC-DL are more concentrated than their counterparts in the case of CE, and also further apart from each other than their counterparts in the case of CE. Again, this is consistent with the NCMI values reported in Table 3.3.

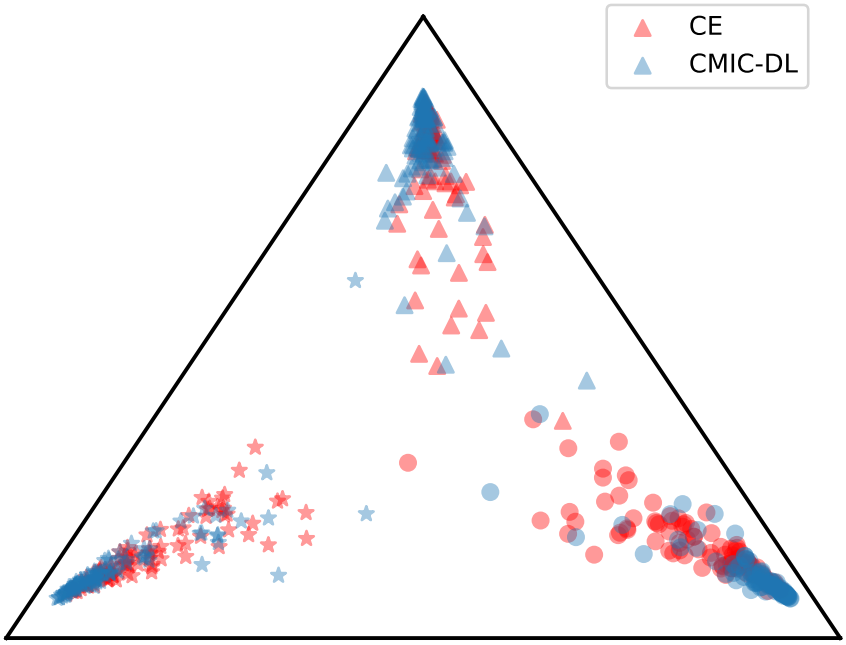
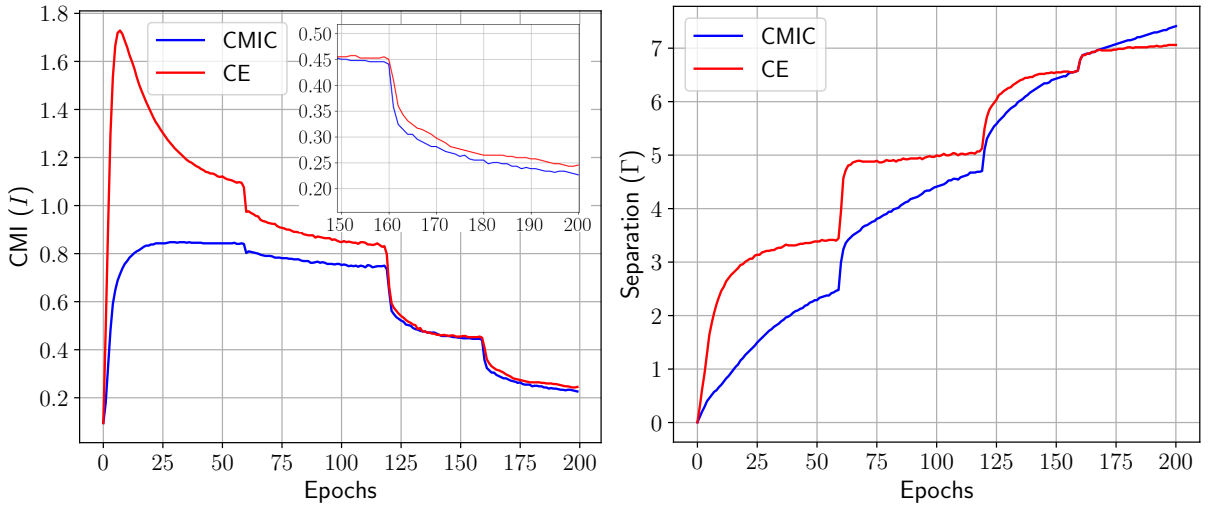


Figure 3.3: Visualization and comparison of concentration and separation: ResNet56 trained via CE vs ResNet56 trained via CMIC, where different shapes indicate different classes.

3.5.5 Evolution of CMI, Γ, NCMI, and error rate

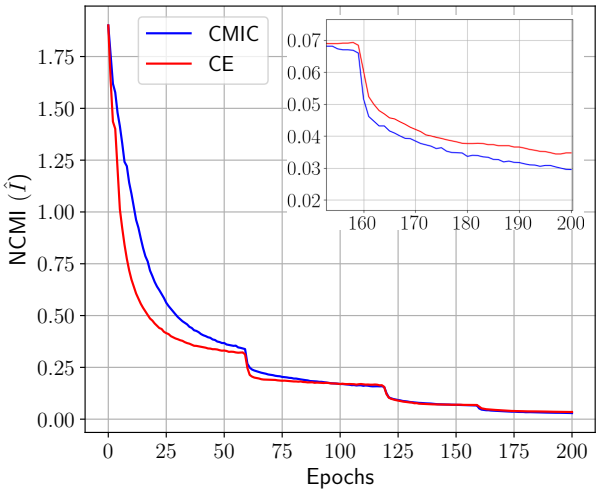
In this subsection, we analyze and visualize a learning process within either our CMIC-DL framework or the conventional CE-based DL framework through the lens of CMI, Γ, NCMI, and error rate. Fig. 3.4 shows the evolution curves of CMI, Γ, NCMI, and error rate over the validation set during the course of training ResNet-56 on CIFAR-100 dataset in each case, where the training setup is the same as that used in Subsection 5.6.4, and we use $\lambda = 0.7$ and $\beta = 0.4$ in the case of CMIC-DL.

As seen in Fig. 3.4a, the CMI value in both CE and CMIC-DL cases is small at the beginning of the training (epoch zero). This is because at the beginning, all clusters in the output probability distribution space $\mathcal{P}([\mathcal{C}])$ stick around together, as shown from the separation distance curve (see Fig. 3.4b), and probability distributions within each cluster are not separated at all. After the training starts and for the first a few epochs, the clusters move away from each other; during the course of movement, probability distributions within each cluster move in different speed, and become separated. As such, both the values of CMI and Γ increase. Indeed, this is shown in Fig. 3.4a and Fig. 3.4b. Hereafter, the clusters continue to move away from each other, while at the same time, probability

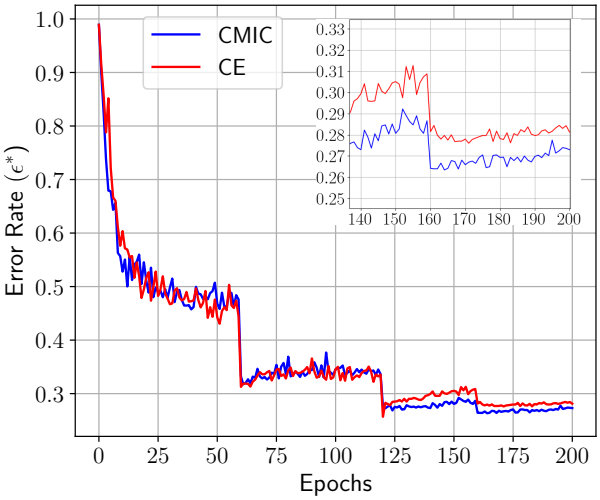


(a) CMI (I).

(b) Separation (Γ).



(c) NCMI ($\hat{I}$).



(d) Error rate (ϵ^*).

Figure 3.4: The evolution curves of (a) CMI, (b) Γ, (c) NCMI, and (d) ϵ^* over the course of training ResNet-56 over CIFAR-100 using CE and CMIC frameworks.

distributions within each cluster tend to move together. Thus the Γ value continues to increase, while the CMI value decreases, as shown again in Fig. 3.4a and Fig. 3.4b.

The above summarizes the general behaviour of the CMI and Γ evolution curves in both CE and CMIC-DL cases. Let us now examine the differences between them. From Fig. 3.4a, it is clear that the CMI evolution curve in the case of CMIC-DL always remains below its counterpart in the CE case. On the other hand, as shown in Fig. 3.4b, although initially the Γ value increases faster in the CE case than in the CMIC-DL case, after the first a few epochs, the rate of increase in Γ value is consistently higher in the CMIC-DL case than in the CE case to the extent that the Γ value in the CMIC-DL case surpasses its counterpart in the CE case in the late stage of the learning process.

Fig. 3.4c and Fig. 3.4d illustrate convergence in the learning process. In addition, from Fig. 3.4c and Fig. 3.4d, we can see that once the learning process is more or less stabilized, both the NCMI value and error rate in the CMIC-DL case are consistently smaller than their counterparts in the CE case. Once again, this is consistent with our observation in Fig. 3.2: the smaller the NCMI value, the lower the error rate. In conjunction with the visualization method discussed in Subsection 3.5.4, we have created a video available at https://youtu.be/G0fDwv6o9Ek to illustrate the learning process during the course of training ResNet-56 on CIFAR-100 dataset in each of the CE and NMIC-DL cases through the lens of CMI and Γ, where concentration and separation are shown for three randomly selected classes, and the evolution curves of CMI and Γ are shown for all classes.

3.5.6 Semantic Meaning of CMI Through Centroid Visualization

Given a DNN: $x \in \mathbb{R}^d \to q_x$, in Shannon sense the CMI $I(X;\hat{Y}|Y=y)$ is the amount of information about X obtained from $\hat{Y}$ given the class label $Y=y$, which is also equal to the average KL divergence between q_X and the centroid s_y in the probability space $\mathcal{P}([C])$. Now, we want to attach semantic meaning to $I(X;\hat{Y}|Y=y)$. To this end, we construct an inverse mapping of the DNN, which maps each probability distribution in $\mathcal{P}([C])$ into a synthetic image (there are many such inverse mappings). Take the DNN as ResNet34 pretrained over ImageNet Dataset as an example. By slightly modifying the reverse mapping algorithm in [209] with BigGANs [17] as the generator, two synthetic images are generated for the centroid of each of the "goldfish" and "bee-eater" classes, which are shown in the columns (a) and (b) in Fig. 3.5. Also shown in Fig. 3.5 are three randomly selected natural images from the respective class (columns (c) to (e)). As seen, the synthetic centroid images have plain background without any context, while the natural images contain rich contextual information. What is the meaning of the CMI

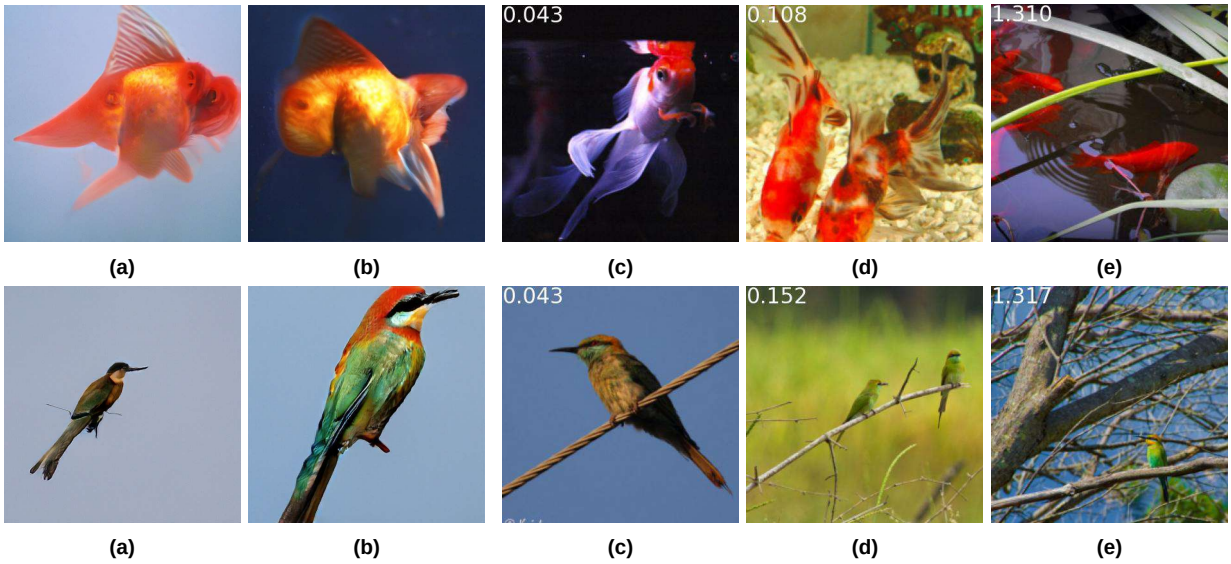


Figure 3.5: Illustration of CMI's semantic meaning—(i) top: goldfish; (ii) bottom: bee-eater—where the first two columns show the synthetic centroid images, and the last three columns show natural images from the respective classes.

$I(X;\hat{Y}|Y=y)$ in the image space? It quantifies the degree of contextual variations of natural images in the class $Y=y$ with respect to the centroid image through the lens of the DNN.

A more comprehensive comparison between the synthetic centroid images and natural images of the whole class is further provided in Figs 3.5.6 and 3.5.6.

3.5.7 Robustness against adversarial attacks

As a by-product, we would expect that DNNs trained within the CMIC-DL framework are more robust against adversarial attacks, in comparison with their counterparts trained within the standard CE-based DL framework. This is because when a DNN is trained within our CMIC-DL framework, its clusters in its output probability distribution space are more compact, and also further separated from each other, in comparison with its counterpart trained within the standard CE-based DL framework. As such, it is harder for an adversary to craft a perturbation which, when added to a clean sample, would result in an attacked sample falling into a cluster with a different label. Our purpose in this subsection is to confirm this by-product. To this end, we have performed the following experiments.



Figure 3.6: Illustration of CMI's semantic meaning: synthetic centroid images (top row) vs natural images in class goldfish. The CMI $I(X;\hat{Y}|Y=y)$ for class goldfish is 0.927. For each natural image in the class, its value of $KL(q_x||s_y)$ is printed at its top left corner.

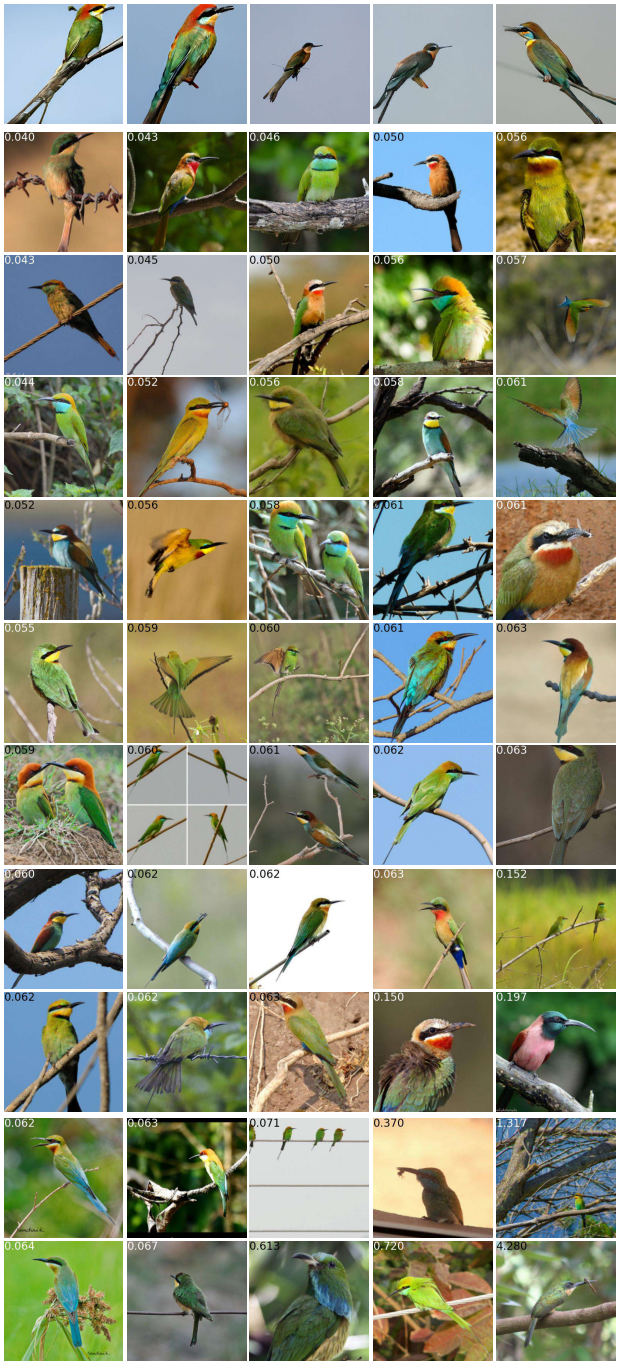


Figure 3.7: Illustration of CMI's semantic meaning: synthetic centroid images (top row) vs natural images in class bee-eater. The CMI $I(X;\hat{Y}|Y=y)$ for class bee-eater is 0.204. For each natural image in the class, its value of $KL(q_x||s_y)$ is printed at its top left corner.

• **Dataset**: We have used MNIST dataset [128] comprising of 10-class handwritten digits.

• **Model**: We have selected a simple DNN with three convolutional and one fully connected layers.

• **Attacks**: Two white-box attacks have been selected, where the adversary has an access to the gradients of the underlying model. Specifically, FGSM [66] and PGD attack [143] with 5 iterations were employed with attack perturbation budgets $\|\epsilon\|_\infty = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35\}$.

• **Training settings**: We have deployed an SGD optimizer with a batch size of 64. We have trained the models for 15 epochs and adopted an step learning rate annealing with decay factor of 0.7. The hyper parameters were selected to be $\lambda^* = 2$ and $\beta^* = 9$ in our CMIC-DL framework due to the fact that the classification task over MNIST dataset is far simpler than that over CIFAR-100 and ImageNet dataset.

Fig. 3.8 illustrates the resulting trade-offs between robust accuracy and perturbation budget. From Fig. 3.8, it is clear that the DNN trained within the CMIC-DL framework is more robust against both FGSM and PGD attacks, in comparison with its counterpart trained within the standard CE-based DL framework, thus confirming the by-product. In addition, the clean accuracy for the models trained within the CE-based DL and CMIC-DL frameworks are 99.14% and 99.21%, respectively, showcasing that the accuracy over the benign samples is not sacrificed for a higher robust accuracy.

We conclude this subsection by pointing out that although CMIC-DL can improve the robustness of DNNs trained therein against adversarial attacks, CMIC-DL itself is not a framework for adversarial training. In our future work, we will fully address CMIC adversarial training by extending the performance metrics of CMI, Γ (separation), and NCMI to the new concepts of robust CMI, robust separation, and robust NCMI.

3.6 Other Information Quantities for Separation

In this section, we explore other information quantities which can also be defined and used as a measure for the inter-class separation of the DNN: $x \in \mathbb{R}^d \to q_x$. Specifically, two more information quantities Γ' and Γ'' are introduced and compared with Γ defined in (3.23). Although they are more or less equivalent, Γ is more convenient for selecting hyper parameters in our CMIC-DL framework.

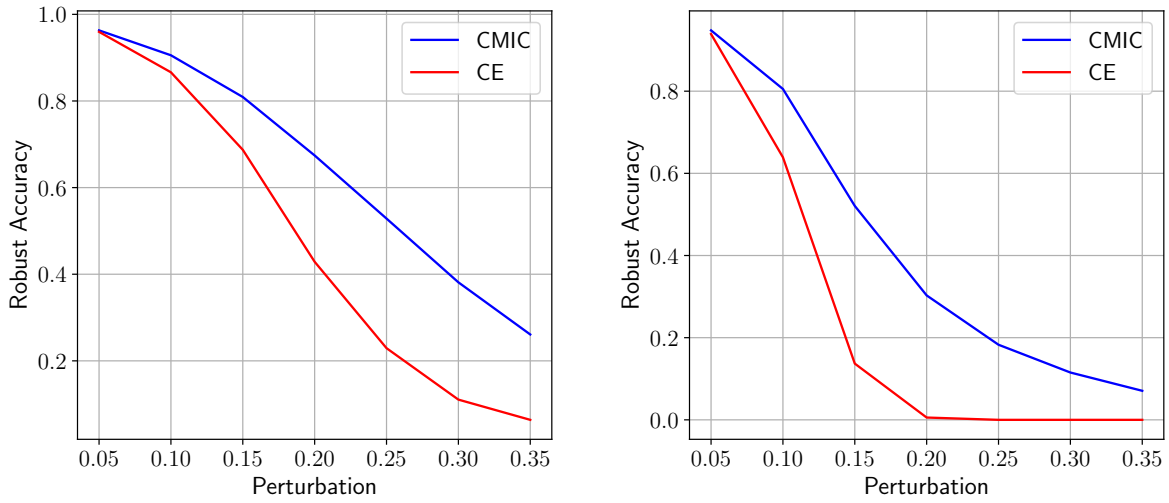


Figure 3.8: The robustness of a simple DNN over MNIST dataset trained within the conventional CE-based DL and CMIC-DL frameworks against (a) FGSM attack and (b) PGD attack with 5 iteraions, respectively.

3.6.1 Information Quantity Γ'

A possible information quantity for measuring inter-class separation can be defined as follows

$$\Gamma' = \mathbf{E}\left[I_{\{Y \neq V\}} D(q_X || q_U)\right], \tag{3.38}$$

where the cross entropy function $H(q_X, q_U)$ in (3.23) is replaced by the KL divergence $D(q_X||q_U)$. To connect Γ' with CMI and Γ, we simplify Γ' as follows:

$$\begin{aligned}
\Gamma' &= \mathbf{E}\left[I_{\{Y\neq V\}}\sum_{i=1}^{C} q_X(i)\ln\frac{q_X(i)}{q_U(i)}\right] \\
&= \mathbf{E}\left[I_{\{Y\neq V\}}\sum_{i=1}^{C} q_X(i)\left(\ln\frac{q_X(i)}{s_Y(i)}+\ln\frac{s_Y(i)}{q_U(i)}\right)\right] \\
&= \mathbf{E}\left[I_{\{Y\neq V\}}D(q_X||s_Y)\right] \\
&+ \mathbf{E}\left[I_{\{Y\neq V\}}\sum_{i=1}^{C} q_X(i)\ln\frac{s_Y(i)}{q_U(i)}\right] \qquad (3.39)\\
&= \mathbf{E}\left[(1-P(Y))D(q_X||s_Y\right] \qquad (3.40)\\
&+ \mathbf{E}\left[I_{\{Y\neq V\}}\sum_{i=1}^{C} s_Y(i)\ln\frac{s_Y(i)}{q_U(i)}\right] \qquad (3.41)\\
&= \mathbf{E}\left[(1-P(Y))D(q_X||s_Y)\right] + \mathbf{E}\left[I_{\{Y\neq V\}}D(s_Y||q_U)\right], \qquad (3.42)
\end{aligned}$$

where (3.40) is due to the fact that V is independent of (X, Y) and given Y, $\mathbf{E}_V[I_{\{Y\neq V\}}] = 1 - P(Y)$, and (3.41) follows from the independence of (X, Y) and (U, V) and (3.13).

Note that the first expectation in (3.42) is related to the CMI $I(X;\hat{Y}|Y)$. Indeed, when $P(Y)$ is equal to a constant, i.e., $1/C$, which is true in most empirical cases, it follows from (3.18) that

$$\mathbf{E}\left[(1-P(Y))D(q_X||s_Y)\right] = (1-\frac{1}{C})I(X,\hat{Y}|Y),$$

which, together with (3.42), implies that

$$\Gamma' = (1-\frac{1}{C})I(X,\hat{Y}|Y) + \mathbf{E}\left[I_{\{Y\neq V\}}D(s_Y||q_U)\right]. \qquad (3.43)$$

Plugging (3.43) into the optimization problem in (3.27), we get the following optimization problem

$$\begin{aligned}
\min_{\theta}\mathbf{E}_X\left[H(P_{Y|X}(\cdot|X), q_{X,\theta})\right] &+ (\lambda-(\beta-\frac{\beta}{C}))I(X;\hat{Y}|Y) \\
&- \beta\mathbf{E}\left[I_{\{Y\neq V\}}D(s_Y||q_{U,\theta})\right]. \qquad (3.44)
\end{aligned}$$

Thus, if Γ' was used as a measure for inter-class separation, then it would cancel out part of the CMI, making the selection of hyper parameters λ and β become harder.

3.6.2 Information Quantity Γ''

Equations (3.43) and (3.44) suggest that one might use the following information quantity as a measure for inter-class separation instead

$$\Gamma'' = \mathbf{E}\left[I_{\{Y\neq V\}} D(s_Y||q_U)\right]. \tag{3.45}$$

In fact, Γ'' has a descent physical meaning in that it measures the average of distances between the output distributions of the DNN in response to input sample instances and the centroids of the clusters with different ground truth labels.

To connect Γ'' with CMI and Γ, we further simplify Γ'' as follows

$$\begin{aligned}
\Gamma'' &= \mathbf{E}\left[I_{\{Y\neq V\}} \sum_{i=1}^{C} q_X(i) \ln \frac{s_Y(i)}{q_U(i)}\right] && (3.46)\\
&= \mathbf{E}\left[I_{\{Y\neq V\}} H(q_X, q_U)\right] + \mathbf{E}\left[I_{\{Y\neq V\}} \sum_{i=1}^{C} q_X(i) \ln s_Y(i)\right] \\
&= \Gamma + \mathbf{E}\left[I_{\{Y\neq V\}} \sum_{i=1}^{C} s_Y(i) \ln s_Y(i)\right] && (3.47)\\
&= \Gamma - \mathbf{E}\left[(1-P(Y))H(s_Y, s_Y)\right]. && (3.48)
\end{aligned}$$

In the above, (3.46) follows from (3.39) and (3.42), (3.47) is due to (3.13) and the fact that X is independent of V, and (3.48) is attributable to the independence of V and Y.

Note again that the second term in (3.48) is related to the CMI $I(X;\hat{Y}|Y)$. Indeed, when $P(Y)$ is equal to a constant, i.e., $1/C$, which is true in most empirical cases, it follows that

$$\begin{aligned}
&\mathbf{E}\left[(1-P(Y))H(s_Y, s_Y)\right] \\
&= (1-\frac{1}{C})H(\hat{Y}|Y) \\
&= (1-\frac{1}{C})\left[I(X;\hat{Y}|Y) + H(\hat{Y}|X,Y)\right] \\
&= (1-\frac{1}{C})\left[I(X;\hat{Y}|Y) + H(\hat{Y}|X)\right], && (3.49)
\end{aligned}$$

where $H(W|Z)$ denotes the Shannon conditional entropy of the random variable W given the random variable Z, and (3.49) is due to the Markov chain $Y \to X \to \hat{Y}$. Combining (3.49) with (3.48) yields

$$\Gamma'' = \Gamma - (1-\frac{1}{C})\left[I(X;\hat{Y}|Y) + H(\hat{Y}|X)\right]. \tag{3.50}$$

Table 3.7: CMI, Γ, NCMI, and error rate values, over the validation set, of some pre-trained models on ImageNet dataset.

Models	CMI	Γ	NCMI	Error rate ϵ^*	Models	CMI	Γ	NCMI	Error rate ϵ^*
ResNet18	0.999	9.891	0.101	0.302	AlexNet	1.331	9.830	0.135	0.434
ResNet34	0.902	9.919	0.090	0.266	EfficientNet-B0	0.692	9.433	0.073	0.220
ResNet50	0.815	9.929	0.082	0.238	EfficientNet-B1	0.661	9.114	0.072	0.213
ResNet101	0.779	9.948	0.078	0.226	EfficientNet-B2	0.639	9.224	0.069	0.193
ResNet152	0.749	9.953	0.075	0.216	EfficientNet-B3	0.627	9.365	0.067	0.180
VGG11	0.959	9.899	0.096	0.296	Wide-ResNet50	0.749	9.935	0.075	0.215
VGG13	0.930	9.909	0.094	0.284	Wide-ResNet101	0.734	9.937	0.073	0.211
VGG16	0.878	9.925	0.088	0.266	MobileNet-V3-Small	1.088	9.898	0.110	0.323
VGG19	0.860	9.930	0.086	0.257	MobileNet-V3-Large	0.922	9.956	0.092	0.259

Plugging (3.50) into the optimization problem in (3.27), we get the following optimization problem

$$\min_{\theta} \mathbf{E}_X \left[H(P_{Y|X}(\cdot|X), q_{X,\theta}) \right] + (\lambda + (\beta - \frac{\beta}{C}))I(X;\hat{Y}|Y) + \beta(1 - \frac{1}{C})H(\hat{Y}|X) - \beta\Gamma. \tag{3.51}$$

Thus, if Γ'' was used as a measure for inter-class separation, then it would further enhance the effect of the CMI, making the selection of hyper parameters λ and β harder as well.

3.7 NCMI Vs. Accuracy on ImageNet

In this section, we replicate the analyses conducted in Section 3.3, but now for the ImageNet dataset. To this end, we pick ResNet-$\{18, 34, 50, 101, 152\}$ [84], VGG-$\{11, 13, 16, 19\}$ [185], EfficientNet-$\{$B0, B1, B2, B3$\}$ [195], Wide-ResNet-$\{50, 101\}$ [231], MobileNet-V3-$\{$small, large$\}$ [241], and AlexNet [122]. They are all pre-trained on ImageNet dataset and obtained from the Pytorch official website[7].

Table 3.7 presents the calculated values of CMI, Γ, and NCMI, along with their respective error rate ϵ^*, for the selected DNNs. In addition, the relationship between the NCMI and error rate ϵ^* is depicted in Fig. 3.9 which shows a positive linear relationship with $\rho = 0.9929$.

[7]https://pytorch.org/vision/stable/models.html.

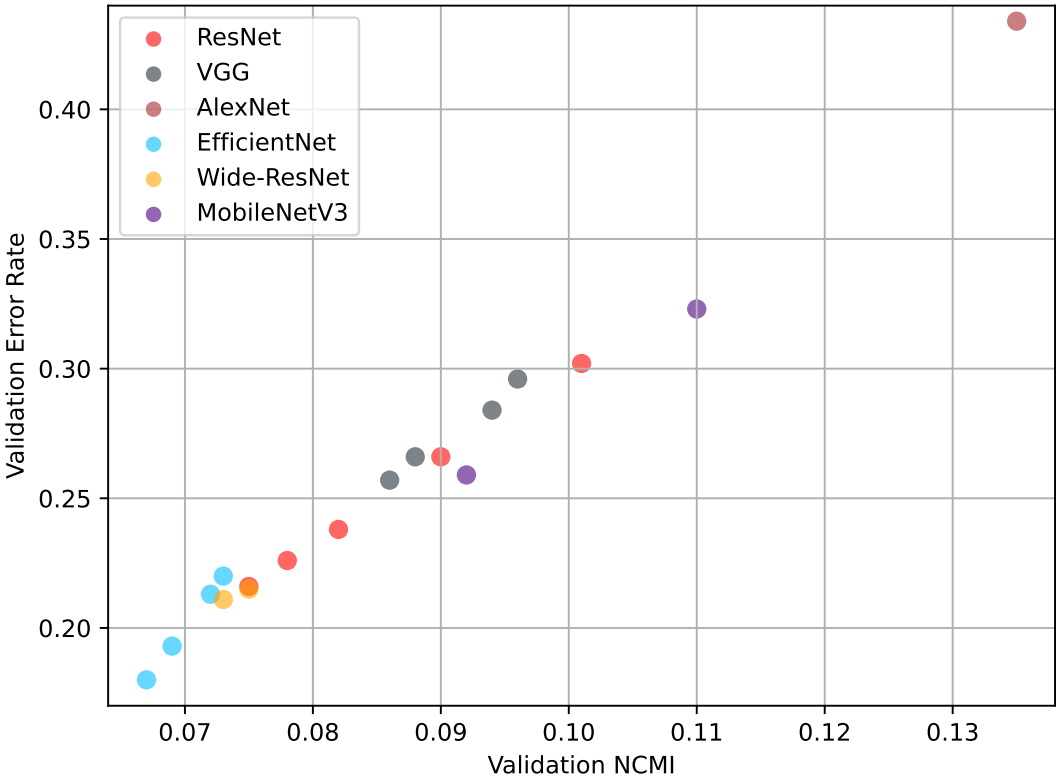


Figure 3.9: The error rate vs NCMI value, over the validation set, of popular pre-trained models on ImageNet dataset.

3.8 Chapter Summary and Directions for Future Work

Viewing a DNN as a mapping from $x \in \mathbb{R}^d$ to q_x, in this chapter we have introduced conditional mutual information (CMI) and normalized conditional mutual information (NCMI) as new performance metrics of the DNN to measure the intra-class concentration and inter-class separation of the DNN. As new performance metrics, CMI and NCMI are in parallel with error rate. We then have used CMI and NCMI to evaluate and compare DNNs of different architectures and sizes. It turns out that NCMI and error rate have essentially a positive linear relationship with their correlation ≥ 0.98 for tested models trained with cross entropy (CE) loss. As such, the NCMI value of a DNN can be used to gauge the prediction performance of the DNN.

Based on NCMI, we have then developed a learning framework called CMI constrained deep learning (CMIC-DL) within which the conventional CE loss is minimized subject to a NCMI constraint. An novel alternating learning algorithm has been further proposed to solve such a constrained optimization problem. Extensive experiment results consistently show that DNNs trained within the CMIC-DL framework outperform those trained using the other DL benchmark methods discussed in this chapter. In addition, with CMI and NCMI as performance metrics for measuring the concentration and separation of a DNN, the learning process of the DNN can also be analyzed and visualized through the evolution of CMI and NCMI.

Open problems include (1) how to extend CMI and NCMI to define concepts of robust CMI, robust separation, and robust NCMI; (2) how to extend CMIC-DL to robust CMIC-DL to fully address adversarial training; (3) how to use CMI to help estimate the conditional probability distribution of Y given X; and (4) the investigation of minimizing NCMI alone without using the standard CE objective function by modifying a predictor. These problems will be addressed in the future.

Chapter 4

Fed-IT: Addressing Class Imbalance in Federated Learning through an Information-Theoretic Lens

In the previous chapter, we explored methods to improve the classification accuracy of DNNs, assuming all data is centrally available for training. In contrast, the focus of this chapter is on applying information-theoretic concepts to distributed learning, with an emphasis on federated learning (FL) [147], where data is distributed across multiple clients.

FL is a promising technology wherein edge devices/clients collaboratively train a machine learning model under the orchestration of a central server. However, due to the inherent data heterogeneity among clients, local datasets on individual clients often exhibit class imbalance, i.e., samples from *majority* classes vastly outnumber those from *minority* classes. This imbalance significantly diminishes the performance of the trained model. To understand why, we first closely examine the output probability distribution clusters of the local deep neural networks (DNNs) in the probability space over the label set, and observe that for class imbalanced datasets, FL has two interesting phenomena: (1) dispersion problem—clusters corresponding to minority classes tend to disperse; and (2) gravity problem—clusters corresponding to minority classes are drawn toward those of majority classes. To overcome these two problems, we then introduce information quantities into FL, propose a new information theoretic loss function for FL, and develop a new FL framework called Fed-IT. It is shown that Fed-IT significantly outperforms previous counterparts, while maintaining client privacy.

4.1 Introduction

Traditionally, machine learning (ML) models are trained in a centralized manner, where the training data is available in a data center or a cloud server. However, in many new applications, devices/clients are reluctant to share their private data with a remote server. To this end, federated learning (FL) emerged as a feasible solution to allow clients to contribute to training a global model using only their local datasets, aided by a central server, but without sharing their local datasets with the server [147]. In FL, the training process begins with the server distributing the global model to all clients. Each client then subsequently computes its local updates to the model parameters using its own private dataset, and then sends its local updates (such as gradients), back to the server; next, the server aggregates the local updates to update the global model, which is then redistributed to clients. These interactive steps continue until convergence.

A primary challenge in FL arises from the diverse sources of data owned by different clients, a phenomenon known as *class imbalance*, where data at clients from a subset of classes (called majority classes) dominate the data population, while data from minority classes, stemming from rare events, are scarce [83]. This class imbalance phenomenon adversely affects the model performance for the minority classes, leading to lower training and testing accuracy [93, 182]. In fact, in numerous real-world scenarios, minority classes play a much more important role beyond their proportion in data. For example, in autonomous vehicle systems, it is imperative for collision detection algorithms to prioritize rare and critical events like sudden obstacles rather than routine road conditions. Similarly, in fraud detection systems, accurately identifying and flagging unusual and potentially fraudulent transactions takes precedence over routine financial transactions.

Utilizing techniques tailored to mitigate class imbalance in centralized training is not applicable to FL. This is partly due to the restricted communication between clients and the server, limited to only the local updates; indeed, for privacy reasons, it is preferable that the server does not request additional information about clients' local data. For further details, please refer to our discussions in Section 6.2. As such, some methods were proposed in the literature to tackle FL class imbalance directly. For example, [53] directly addresses class imbalance in FL, but requires clients to upload their local data distributions, which may expose latent backdoors to attacks and lead to privacy leakage. Moreover, it also requires placing a number of proxy servers, which increases the FL complexity and incurs more computation overhead. Other methods include those proposed in [180, 206], yet their performance is not promising when the dataset heterogeneity is high in the local datasets (as also supported by our experiments in Section 6.5).

As a remedy, in this chapter, we systematically address the class imbalance issue in

Federated learning by employing concepts from **I**nformation **T**heory (IT) and proposing a new information theoretic loss function for FL, a method which we refer to as **Fed-IT** hereafter. To this end, we first follow the approach in [221] and treat a deep neural network (DNN) as a mapping from the raw image space to the output probability distribution space. The instances of each class in the DNN's output space form distinct clusters. Through a series of experiments, we observed that in cases of class imbalance within the training dataset, the clusters corresponding to minority classes (i) are disperse, and (ii) tend to be drawn towards the clusters associated with majority classes[1], two interesting phenomena referred to as the dispersion problem and the gravity problem, respectively. To mitigate these problems, in Fed-IT, the clients' normal loss functions are replaced by their new information theoretic loss functions so that they can learn the model with output clusters in the probability distribution space possessing the following two properties:

P1 They are compact.

P2 They are well-separated and located on pre-defined positions.

For [P1], as shown in [221], the cluster's compactness in the DNN's output space can be measured via conditional mutual information (CMI) $I(X;\hat{Y}|\ Y)$, where X, Y and $\hat{Y}$ are three random variables representing the input image, the ground truth label, and the label predicted by the DNN, respectively. The lower this CMI value is, the more compact the cluster would be. Hence, to satisfy [P1], each client trains its local model to minimize $I(X;\hat{Y}|\ Y)$.

To satisfy [P2], we push the centroid of each cluster to its respective fixed probability distribution. Specifically, we select these fixed probability distributions to be *smoothed* one-hot vectors, ensuring that they are (i) fixed for all the clients, (ii) well-separated, and (iii) not over-confident as one-hot vectors. Thus, the clients minimize the KL divergence between the cluster centroids and the *smoothed* one-hot vectors, which we refer to as the position loss.

Henceforth, in Fed-IT, the new information theoretic loss function at each client comprises three terms: the conventional cross-entropy (CE), the CMI term, and the position loss. In summary, the contributions of this chapter are as follows:

• We empirically discover that for class imbalanced datasets, deep learning in general and FL in particular have two interesting phenomena: (1) dispersion problem—clusters

[1]In [42], the authors similarly observed that the DNN's representation spaces for minority classes are pulled towards those of majority classes.

corresponding to minority classes tend to disperse; and (2) gravity problem—clusters corresponding to minority classes are drawn toward those of majority classes.

• To overcome the dispersion and gravity problems, we develop a new FL framework dubbed Fed-IT by proposing a novel information theoretic loss function for FL.

• Unlike many existing FL class imbalance mitigation methods, Fed-IT preserves client privacy by not requiring clients to share additional information with the server.

• By conducting thorough experiments over different datasets, we show that in the presence of label imbalance, Fed-IT yields a higher classification accuracy compared to its counterparts.

4.2 Related Work

In supervised learning, data imbalance is prevalent in various scenarios, such as in image recognition for disease diagnosis [214], and object detection [134]. We review some techniques in the literature to tackle this issue in centralized and FL settings in subsections 4.2.1 and 4.2.2, respectively.

4.2.1 Class imbalance in centralized training

In centralized training, methods for addressing the class imbalance issue can be categorized into two approaches: (i) data-level, and (ii) algorithm-level.

• **Data-level**: Over-sampling minority classes or under-sampling majority classes are straightforward yet effective methods to achieve more balanced class distributions [83,123]. Nonetheless, these approaches may be susceptible to overfitting on minority classes or potential information loss on majority classes. Data augmentation is another technique [129, 168]; however, it cannot be applied in the FL setting due to privacy constraints.

• **Algorithm-level**: One approach in this category involves cost-sensitive learning [150, 208]. The cost-sensitive approaches need to analyze the distribution of training data, e.g., re-weighting the loss via inverse class frequency, and are not effective for FL due to the mismatch between local data distribution and the global one. Another approach is to change the loss function [113, 134]. Nevertheless, due to the data heterogeneity in the FL setting, different clients might have different dominant classes, and hence the direct adoption may not be applicable.

4.2.2 Class imbalance in FL

Class imbalance poses significant challenges in FL due to its decentralized nature. Unlike traditional machine learning, where data is centralized, FL involves training models across multiple client devices with non-uniform and often unbalanced datasets. This imbalance manifests at different levels:

- Local Level: Individual client devices may have highly skewed class distributions, leading to poor local model updates.
- Global Level: Systematic imbalance across the collective data of all clients can adversely affect the global model's performance.

The decentralized nature of FL exacerbates these challenges due to the potential mismatch between class distributions at the client and server levels. For instance, the discrepancies in class distributions across clients introduce variability in the local updates, which, when aggregated, can degrade the overall model performance [31, 215]. Moreover, global-level class imbalance, where some classes are underrepresented across all clients, has been shown to delay model convergence and reduce classification accuracy [49, 186]. This is mathematically supported in [54], where the impact of imbalanced training data on FL applications is rigorously analyzed.

Another critical aspect of FL is the preservation of user privacy, which complicates addressing class imbalance. Since data privacy is a core principle of FL, accessing even basic statistical information about the class distributions on client devices is restricted. This lack of visibility increases the difficulty of estimating the severity of the imbalance, let alone mitigating it effectively. Consequently, many strategies for managing class imbalance in FL systems adopt a two-stage approach:

1. Class Distribution Estimation: Inferring the class distribution while maintaining data privacy.
2. Addressing Class Imbalance: Applying techniques to mitigate the imbalance without direct access to client data.

• Existing Surveys on Federated Learning and Class Imbalance

Given the rapid advancements in FL, several survey papers have explored its various aspects. Comprehensive overviews of FL and its applications can be found in [227, 233],

while a detailed exposition of the field's advances and challenges is presented in [107]. Privacy-preserving techniques, a cornerstone of FL, are the focus of [152], where threat analyses and methods to enhance data security are examined.

Class imbalance, a well-recognized problem in traditional machine learning (ML), has also attracted significant attention. Surveys like [68, 109] provide systematic overviews of class imbalance in ML, offering insights into state-of-the-art solutions for addressing this pervasive issue. Although traditional ML solutions lay the groundwork, the decentralized and privacy-preserving nature of FL introduces unique complexities that require tailored approaches.

• Class Imbalance and Non-IID Data in FL

The class imbalance problem in FL is closely related to the broader issue of non-IID data, which reflects the statistical heterogeneity commonly found in FL training datasets. Non-IID data often arises from diverse user behaviors, device-specific biases, or regional variations in data collection, all of which contribute to uneven class distributions. These challenges have been explored in detail in [243], which highlights the degradation of FL models due to non-IID data.

The interplay between non-IID data and class imbalance has prompted the development of specialized strategies. For instance, [249] provides a detailed review of recent research addressing the impact of non-IID data on FL performance. Strategies to mitigate these issues include modifications to aggregation algorithms, personalized models for individual clients, and data-sharing mechanisms that respect privacy constraints.

• Addressing Class Imbalance in FL

To address class imbalance in FL, several methods have been proposed, ranging from re-weighting strategies and data augmentation techniques to algorithmic adjustments for model training. These methods often involve:

- Re-weighting Loss Functions: Adjusting the contribution of underrepresented classes in the loss calculation.
- Synthetic Data Generation: Using techniques such as GANs (Generative Adversarial Networks) to augment underrepresented classes.
- Personalized FL Models: Creating client-specific models to better adapt to local class distributions while leveraging knowledge from global training.

The survey in [249] categorizes these approaches and evaluates their effectiveness in both IID and non-IID settings. However, there remains a significant gap in the literature regarding methods that simultaneously address class imbalance, non-IID data, and privacy preservation.

The authors in [53] select clients with complementary class distributions to perform updates, yet this process requires clients to reveal the class distribution to the server. [226] attempts to recombine clients to pair complementary ones for collaborative training. However, in cases where certain clients fail to find matching counterparts, they might miss the opportunity to participate in federated aggregation. In [141], balanced virtual features are employed to retrain the classifier. However, this approach requires collecting the global class distribution, which escalates the risk of privacy disclosure.

Ratio-Loss [206] operates under the assumption that an auxiliary dataset with balanced classes is accessible on the server. It utilizes this dataset to infer the composition of the training data and devises a ratio loss mechanism to alleviate the effects of class imbalance. Also, CReFF [180] re-trains a classifier based on federated features in a privacy preserving manner. Nevertheless, for both [180, 206], the performance declines with an increase in the degree of data heterogeneity.

4.3 Notation and Preliminaries

4.3.1 Notation

For a positive integer C, let $[C] \triangleq \{1, \ldots, C\}$. Denote by $P[i]$ the i-th element of a vector P. We use $|\mathcal{C}|$ to denote the cardinality of a set $\mathcal{C}$. The cross entropy of two probability distributions P_1, P_2 is defined as $H(P_1, P_2) = \sum_{c=1}^{C} -P_1[c] \ln P_2[c]$, and their Kullback–Leibler (KL) divergence is defined as $\mathrm{KL}(P_1||P_2) = \sum_{c=1}^{C} P_1[c] \ln \frac{P_1[c]}{P_2[c]}$.

Let $\mathbb{E}_X[\cdot]$ denote the expected value w.r.t. X. The mutual information between two random variables X and Y is given by $I(X, Y) = H(X) - H(X|Y)$, and the conditional mutual information of X and Y given a third random variable Z is $I(X, Y|Z) = H(X|Z) - H(X|Y, Z)$.

4.3.2 FL setup

In this chapter, we consider a classification FL task with K clients. The local dataset for client k, is denoted by $\mathcal{D}_k$, $k \in [K]$, and the global dataset is represented by $\mathcal{D} = \bigcup_{k\in[K]} \mathcal{D}_k$.

The FL task is defined as follows

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \mathcal{D}) \triangleq \frac{1}{K} \sum_{k \in [K]} f_k\left(\boldsymbol{\theta}, \mathcal{D}_k\right), \tag{4.1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector representing the parameters of the shared DNN model, and $f_k(\cdot)$ is the local loss for client k, conventionally cross entropy loss.

The widely adopted approach to problem (4.1) is to employ distributed (stochastic) gradient descent method (DGD/DSGD). In the t-th round of DGD, the server shares $\boldsymbol{\theta}^t$ with all clients, each client computes its local gradient with respect to $\boldsymbol{\theta}^t$ and transmits it to the server which aggregates them to form the global gradient and perform one GD step as:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta^g \nabla f(\boldsymbol{\theta}^t) = \boldsymbol{\theta}^t - \frac{\eta^g}{K} \sum_{k \in [K]} \nabla f_k(\boldsymbol{\theta}^t), \tag{4.2}$$

where η^g is the global learning rate.

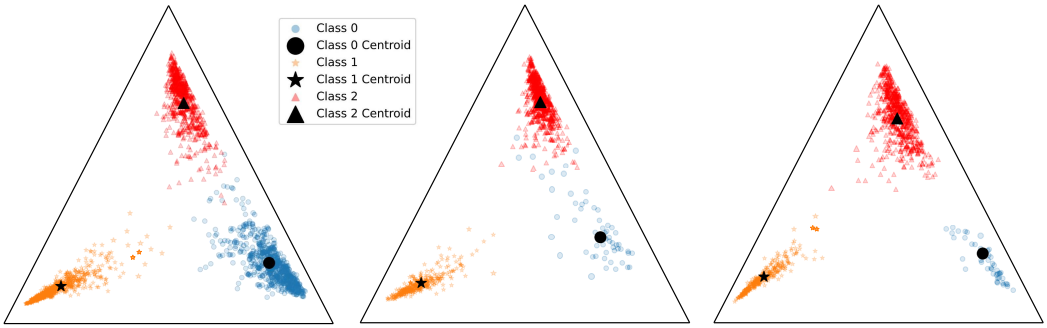
4.4 Motivation and Methodology

Given a shared DNN model, in this section, we first motivate our method by observing the DNN's output probability distribution space when the training dataset is class-imbalanced in Subsection 4.4.1; then, based on this observation, we develop a new FL algorithm in Subsection 4.4.2.

4.4.1 Motivation

Assume the shared DNN model is ResNet 56. The outputs of the DNN in response to all samples from a class form a cluster in the probability distribution space. Use the conventional CE loss to train ResNet 56 over CIFAR-100 dataset, which is class-balanced. Randomly select three classes labeled as Class 0, Class 1, and Class 2. Fig. 4.3a shows the three clusters corresponding to these three classes, which are projected into the two-dimensional probability simplex. Please refer to [221] and the Appendix for how the projection is conducted.

Now let us create a class-imbalanced dataset from CIFAR-100 dataset by reducing the sizes of Class 0 and Class 1 to 10% and 50% of their original sizes, respectively while



(a) Balanced, CE loss (b) Imbalanced, CE loss (c) Imbalanced, $\mathcal{L}_{Fed-IT}$ loss

Figure 4.1: Illustration of the output probability distribution clusters of ResNet 56 corresponding to three randomly selected classes, projected over the two-dimensional probability simplex: (a) trained over the CIFAR-100 class-balanced dataset using the conventional CE loss; (b) trained over a class-imbalanced dataset using the conventional CE loss; (c) trained over the class-imbalanced dataset using the proposed information theoretic loss $\mathcal{L}_{\text{Fed-IT}}$.

keeping other classes intact. The size reduction was conducted by randomly selecting 90% (50%, resp.) samples from Class 0 (1, resp.) for deletion. Use the conventional CE loss to train ResNet 56 again from scratch over the class-imbalanced dataset. The resulting three clusters, projected into the two-dimensional probability simplex, are shown in Fig. 4.3b. As seen, the cluster corresponding to class 0 becomes dispersed and is also drawn towards the cluster corresponding to class 2; for convenience, we refer to the former as the *dispersion problem*, and the latter as the *gravity problem*. This in turn lowers the classification accuracy.

In the next section, we introduce methods to overcome the dispersion and gravity problems, resulting in the three clusters shown in Fig. 4.3c for the same class-imbalanced dataset.

4.4.2 Methodology

Following [221], we now regard the shared DNN model as a mapping $f : x \to P_x$, where $x \in \mathbb{R}^d$ is an input image, and P_x is the resulting output probability distribution over the class label set $[C]$. Let y denote the ground-truth label of x.

Consider a pair of random variables (X, Y), where $X \in \mathbb{R}^d$ represents the raw input to the DNN and Y denotes the ground truth label of X. The distribution of (X, Y) governs

either a training set or a testing set, as the case may be. Let $\hat{Y}$ denote the label predicted by the DNN with probability $P_X(\hat{Y})$ in response to the input X.

Adopting this probabilistic perspective, we leverage information-theoretic quantities to train each client in FL such that it can learn compact clusters with their centroids close to fixed and prescribed positions.

Compactness of clusters

As discussed in [221], the concentration (compactness) of the DNN's clusters can be measured via CMI value $I(X;\hat{Y}|\,Y)$, where X, Y and $\hat{Y}$ are three random variables representing the input image, the ground truth label, and the label predicted by the DNN, respectively. Specifically, $I(X;\hat{Y}|Y=y)$, i.e., the CMI between X and $\hat{Y}$ given $Y=y$ can be calculated as follows [221]:

$$I(X;\hat{Y}|\,Y=y) = \sum_x P_{X|Y}(x|y)\Big[\sum_{i=1}^{C} P(\hat{Y}=i|x,y) \times \ln \frac{P(\hat{Y}=i|x,y)}{P_{\hat{Y}|y}(\hat{Y}=i|Y=y)}\Big] \tag{4.3}$$

$$= \mathbb{E}_{X|Y}\left[\left(\sum_{i=1}^{C} P_X[i] \ln \frac{P_X[i]}{P_{\hat{Y}|y}(\hat{Y}=i|Y=y)}\right)|Y=y\right] \tag{4.4}$$

$$= \mathbb{E}_{X|Y}\Big[\mathrm{KL}(P_X||P_{\hat{Y}|y})|Y=y\Big] \tag{4.5}$$

where (4.4) is due to the fact that $Y \to X \to \hat{Y}$ forms a Markov chain. What is the rationale behind using this quantity to assess the compactness of the cluster? In fact, the centroid of the cluster corresponding to samples whose ground-truth labels are y is the average of P_X with respect to the conditional distribution $P_{X|Y}(\cdot|y)$, which is exactly the conditional distribution of $\hat{Y}$ given $Y=y$

$$P_{\hat{Y}|y} = \mathbb{E}[P_X|Y=y]. \tag{4.6}$$

Hence, the information quantity $I(X;\hat{Y}|y)$ quantifies the concentration of the cluster formed by the instances of the random point P_X given $Y=y$ around its centroid $P_{\hat{Y}|y}$. We additionally provide visualizations in section 4.5.3 illustrating the impact of CMI values on the compactness of clusters. Averaging $I(X;\hat{Y}|y)$ with respect to the distribution $P_Y(y)$

of Y, we get the conditional mutual information $I(X;\hat{Y}|Y)$ between X and $\hat{Y}$ given Y:

$$I(X;\hat{Y}|Y) = \sum_{y\in[C]} P_Y(y) I(X;\hat{Y}|y) \tag{4.7}$$

$$= \mathbb{E}\Big[D(P_X||P_{\hat{Y}|Y})\Big] \tag{4.8}$$

$$= \sum_y \sum_x P(x,y)\Big[\sum_{i=1}^{C} P(\hat{Y}=i|x) \times \ln \frac{P(\hat{Y}=i|x)}{P_{\hat{Y}|y}(\hat{Y}=i|Y=y)}\Big] \tag{4.9}$$

In practice, when a client does not know the true $P(x,y)$, it can approximate $I(X;\hat{Y}|Y)$ by its empirical value using its local dataset. To this end, for client k we define $\mathcal{D}_{k,y} = \{(x_j,y_j) \in \mathcal{D}_k \ : y_j = y\}$ the multiset of all training samples in $\mathcal{D}_k$ whose ground truth labels are y. Then, the empirical $I(X;\hat{Y}|Y)$ for client k is calculated as

$$I_{k,\text{emp}}(X;\hat{Y}|Y) = \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} \text{KL}(P_x \ || \ P_{k,y}), \tag{4.10}$$

$$\text{where} \quad P_{k,y} = \frac{1}{|\mathcal{D}_{k,y}|} \sum_{(x_j,y)\in\mathcal{D}_{k,y}} P_{x_j}, \quad \text{for} \ \ y \in [C]. \tag{4.11}$$

Fixed and prescribed ideal centroid positions

To overcome the gravity problem illustrated in Fig. 4.3b, we next want the centroids of all clusters at each client to be nicely located.

Where are the ideally targeted positions for the centroids of all clusters at each client? Intuitively, these positions should be well-separated to prevent different clusters from overlapping with each other. One possible choice is to select the one-hot vector $\mathbf{e}_y$, which is the C-dimensional probability vector where the y-th entry is 1 and all other entries are 0, as the targeted position for the centroid of the cluster corresponding to label y. However, this choice is too aggressive in the sense that it would force all P_x in the corresponding cluster to be or very close to the one-hot vector $\mathbf{e}_y$, resulting in an overfitting problem, which is common and inherent in the conventional CE loss. To alleviate this issue, we instead target the centroid of the cluster corresponding to label y at a *smoothed* one-hot vector whose j-th entry, $j \in [C]$, is defined as follows

$$\tilde{\mathbf{e}}_{y,\epsilon}[j] = \begin{cases} 1-\epsilon & \text{if } j = y \\ \frac{\epsilon}{C-1} & \text{if } j \neq y \end{cases} \tag{4.12}$$

During FL, we encourage the clusters at each client to be located around $\tilde{\mathbf{e}}_{y,\epsilon}$ for each label y by minimizing

$$\mathbb{E}\Big[\mathrm{KL}(P_{\hat{Y}|Y}||\tilde{\mathbf{e}}_{Y,\epsilon})\Big] = \sum_{y} P_Y(y)\mathrm{KL}(P_{\hat{Y}|y}||\tilde{\mathbf{e}}_{y,\epsilon}). \tag{4.13}$$

We refer to the quantity in (4.13) as the position loss. When the joint distribution $p(x, y)$ is unknown, $P_{\hat{Y}|y}$ in (4.13) could be approximated by (4.11) for Client k.

4.4.3 New information theoretic Loss for Fed-IT

To overcome both the dispersion and gravity problems in FL, the above discussion now suggests a new way to FL dubbed Fed-IT. Instead of minimizing the conventional CE loss alone at each client, we now minimize in Fed-IT the following new information theoretic loss at each client:

$$\mathcal{L}_{\text{Fed-IT}} = \mathbb{E}\Big[-\ln P_X(Y)\Big] + \alpha I(X;\hat{Y}|Y) + \gamma\mathbb{E}\Big[\mathrm{KL}(P_{\hat{Y}|Y}||\tilde{\mathbf{e}}_{Y,\epsilon})\Big], \tag{4.14}$$

where $\alpha > 0$ and $0 < \gamma < \alpha$ are two scalars representing the trade-offs among the CE, CMI, and position losses. In other words, in Fed-IT Client k solves the following minimization problem locally through its local gradient computation:

$$\min_{\theta}\left[\mathcal{L}_{\text{Fed-IT}}(\theta, \mathcal{D}_k) = \mathbb{E}\Big[-\ln P_{X,\theta}(Y)\Big] + \alpha\ I(X;\hat{Y}|Y) + \gamma\ \mathbb{E}\Big[\mathrm{KL}(P_{\hat{Y}|Y}||\tilde{\mathbf{e}}_{Y,\epsilon})\Big]\right], \tag{4.15}$$

where $P_{X,\theta}$ is the output probability distribution of the shared DNN model in response to input X when the model parameters are θ. Note that $\hat{Y}$ in (4.15) also depends on θ.

The objective function in (4.15) is not amenable to parallel computation via GPU. To overcome this issue, we follow [221], and introduce a dummy distribution Q_y over the label set $[C]$ for each $y \in [C]$, and convert the loss function into a format suitable for GPU-based deep learning, as shown in the following theorem.

Theorem 3. *For any $\alpha > 0$ and $0 < \gamma < \alpha$,*

$$\begin{aligned}
&\min_{\theta}\left[\mathbb{E}\Big[-\ln P_{X,\theta}(Y)\Big] + \alpha\ I(X;\hat{Y}|Y) + \gamma\ \mathbb{E}\Big[KL(P_{\hat{Y}|Y}||\tilde{\mathbf{e}}_{Y,\epsilon})\Big]\right] \\
&= \min_{\theta}\min_{Q_c : c\in[C]}\left[\mathbb{E}\Big[-\ln P_{X,\theta}(Y)\Big] + (\alpha-\gamma)\mathbb{E}\Big[KL(P_{X,\theta}||Q_Y)\Big] + \gamma\ \mathbb{E}\Big[KL(P_{X,\theta}||\tilde{\mathbf{e}}_{Y,\epsilon})\Big]\right].
\end{aligned} \tag{4.16}$$

Proof. To simplify our notation, we ignore the notation θ and work with the loss function $\mathcal{L}_{\text{Fed-IT}}$ in (4.14) directly. Since $Y \to X \to \hat{Y}$ forms a Markov chain, it can be verified that

$$\begin{aligned}
\mathcal{L}_{\text{Fed-IT}} &= \mathbb{E}\Big[-\ln P_X(Y)\Big] + \alpha\mathbb{E}\Big[\ln \frac{P_X(\hat{Y})}{P_{\hat{Y}|Y}(\hat{Y})}\Big] + \gamma\mathbb{E}\Big[\ln \frac{P_{\hat{Y}|Y}(\hat{Y})}{\tilde{\mathbf{e}}_{Y,\epsilon}(\hat{Y})}\Big] \\
&= \mathbb{E}\Big[-\ln P_X(Y)\Big] + (\alpha-\gamma)\mathbb{E}\Big[\ln \frac{P_X(\hat{Y})}{P_{\hat{Y}|Y}(\hat{Y})}\Big] \\
&\quad + \gamma\mathbb{E}\Big[\ln P_X(\hat{Y})\Big] - \gamma\mathbb{E}\Big[\ln \tilde{\mathbf{e}}_{Y,\epsilon}(\hat{Y})\Big] \\
&= \mathbb{E}\Big[-\ln P_X(Y)\Big] + (\alpha-\gamma)\mathbb{E}\Big[\ln \frac{P_X(\hat{Y})}{P_{\hat{Y}|Y}(\hat{Y})}\Big] + \gamma\mathbb{E}\Big[\ln \frac{P_X(\hat{Y})}{\tilde{\mathbf{e}}_{Y,\epsilon}(\hat{Y})}\Big] \\
&= \mathbb{E}\Big[-\ln P_X(Y)\Big] + (\alpha-\gamma)\mathbb{E}\Big[\ln \frac{P_X(\hat{Y})}{P_{\hat{Y}|Y}(\hat{Y})}\Big] \\
&\quad + \gamma\mathbb{E}\Big[\mathrm{KL}(P_X||\tilde{\mathbf{e}}_{Y,\epsilon})\Big] \\
&= \mathbb{E}\Big[-\ln P_X(Y)\Big] + (\alpha-\gamma)\min_{Q_c:c\in[C]}\mathbb{E}\Big[\ln \frac{P_X(\hat{Y})}{Q_Y(\hat{Y})}\Big] \qquad (4.17) \\
&\quad + \gamma\mathbb{E}\Big[\mathrm{KL}(P_X||\tilde{\mathbf{e}}_{Y,\epsilon})\Big] \\
&= \mathbb{E}\Big[-\ln P_X(Y)\Big] + (\alpha-\gamma)\min_{Q_c:c\in[C]}\mathbb{E}\Big[\mathrm{KL}(P_X||Q_Y)\Big] \qquad (4.18) \\
&\quad + \gamma\mathbb{E}\Big[\mathrm{KL}(P_X||\tilde{\mathbf{e}}_{Y,\epsilon})\Big] \\
&= \min_{Q_c:c\in[C]}\mathbb{E}\Big[-\ln P_X(Y) + (\alpha-\gamma)\mathrm{KL}(P_X||Q_Y) + \gamma\mathrm{KL}(P_X||\tilde{\mathbf{e}}_{Y,\epsilon})\Big]. \qquad (4.19)
\end{aligned}$$

In the above, (4.17) is due to the fact that KL divergence is nonnegative and the minimization therein is achieved when

$$Q_y(\hat{y}) = P_{\hat{Y}|y}(\hat{y}),\ \forall y,\ \hat{y} \in [C]. \qquad (4.20)$$

From (4.19), Theorem 3 is concluded. □

Note that given θ and $\{Q_c\}_{c=1}^C$, the objective function in (4.16) is additive w.r.t. the distribution of (X, Y), or equivalently input data, and thus friendly to GPU-based learning/inference.

Table 4.1: The validation accuracy (%) for different FL methods over CIFAR-$\{10, 100\}$ and TinyImageNet datasets. The reported numbers (avg±std) are obtained from 5 runs. **Bold** and underlined values denote the best and second best results, respectively.

Method	Cifar10		Cifar100		TinyImageNet	
	Dir(0.3)	Dir(1.0)	Dir(0.3)	Dir(1.0)	Dir(0.3)	Dir(1.0)
FedAvg	66.40 ± 3.13	73.07 ± 1.60	35.14 ± 0.48	36.07 ± 0.41	34.63 ± 0.26	37.65 ± 0.37
FedPer	61.58 ± 0.43	63.33 ± 0.53	15.04 ± 0.06	14.69 ± 0.03	15.28 ± 0.14	13.71 ± 0.07
Ditto	66.71 ± 1.10	73.41 ± 0.77	35.55 ± 0.33	36.12 ± 0.35	34.70 ± 0.35	37.61 ± 0.24
FedRep	40.13 ± 0.17	47.92 ± 0.38	5.42 ± 0.03	6.37 ± 0.04	3.27 ± 0.02	3.91 ± 0.03
FedBABU	62.78 ± 3.09	70.34 ± 1.72	32.41 ± 0.40	32.34 ± 0.49	26.36 ± 0.32	30.25 ± 0.32
FedROD	**72.31 ± 0.16**	**75.50 ± 0.15**	33.83 ± 0.25	35.20 ± 0.19	35.46 ± 0.28	37.71 ± 0.31
Ratio-Loss	65.24 ± 0.13	70.12 ± 0.15	20.33 ± 0.42	19.41 ± 0.11	24.85 ± 0.22	26.51 ±0.29
CReFF	66.46 ± 1.40	71.63 ± 0.61	22.90 ± 0.30	22.21 ± 0.15	25.82 ± 0.41	27.87 ± 0.38
Fed-IT	68.77 ± 0.62	72.55 ± 1.12	**38.44 ± 0.21**	**39.29 ± 0.15**	**37.02 ± 0.30**	**38.60 ± 0.36**

4.4.4 Algorithm for Fed-IT

Similar to many recent FL algorithms [77, 92, 131, 147, 151], we allow each client to perform a couple of local epochs e before sending its gradient update to the server. The overall interactions between each client and the server are the same as in the conventional FedAvg method. However, local iterations with each epoch are different. Based on Theorem 3 and following [221], an alternating iteration procedure is applied instead locally at each client. For the detailed description of the local Fed-IT algorithm, please refer to our full paper [1].

4.5 Experiment Results

In this section, we conduct experiments to show the superior effectiveness of Fed-IT compared to some counterparts.

• **Datasets and setups**: We use three commonly used datasets in FL realm, namely CIFAR-$\{10, 100\}$ [121] and TinyImageNet [126]. We follow the setup in [25], and use a simple convolutional neural network for CIFAR-$\{10, 100\}$ datasets, and ResNet18 [84] for TinyImageNet dataset.

To induce class imbalance in the local datasets of clients, we distribute the dataset among the clients deploying Dirichlet allocation Dir(β) [205], where we test $\beta \in \{0.3, 1.0\}$

in our experiments (following [42]). This allocation scheme results in each client likely having one or two dominant classes, with few or even zero samples from the remaining classes.

Also, in all the experiments, we consider 100 clients where 10% clients participate in each round.

• **Benchmarks**: We use eight FL methods: FedAvg [147], five personalized FL methods Ditto [130] and FedPer [4], FedRep [38], FedBABU [159], FedROD [25] and two FL methods CReFF [180], Ratio-Loss [206] tailored to tackle class imbalance[2].

For Fed-IT, we set $\epsilon = 0.3$, $\alpha = 0.8$ and $\gamma = 0.05$. In addition, the implementation details for all the benchmark methods (including Fed-IT) are deferred to the Appendix.

4.5.1 Analysis of the Results

We report the testing accuracy for all the methods in Table 4.1. It is evident that Fed-IT achieves higher test accuracy for both CIFAR-100 and TinyImageNet datasets. However, for CIFAR-10, some benchmarks outperform Fed-IT. It is worth noting that CIFAR-100 and TinyImageNet datasets more closely resemble real-world datasets, as they contain more classes with a larger number of samples.

Furthermore, for the same three classes as those used to depict Fig. 4.3a-b, we visualize their clusters learned by Fed-IT in Fig. 4.3c. It is evident that the clusters learned by Fed-IT are more compact, and notably, the cluster for class 0 (minority) is pushed apart from class 2 (majority).

4.5.2 Ablation study

To analyze the impact of parameters α and γ in Fed-IT, we conduct an ablation study on CIFAR-100. Specifically, we set each parameter alternatively to zero while keeping the other parameter unchanged, allowing us to observe the effect of each parameter individually. The results are reported in Table 4.2 which show the necessity of both parameters α and γ.

[2]As noted in [42], when dataset heterogeneity is significant, FedAvg demonstrates superior performance compared to imbalance-resistant FL algorithms.

Table 4.2: The effect of hyper-parameter α and γ in Fed-IT.

Method	Dir(0.3)	Dir(1.0)
Fed-IT ($\alpha = 0.8$, $\gamma = 0.05$)	38.44 ± 0.21	39.29 ± 0.15
Fed-IT ($\alpha = 0$, $\gamma = 0.05$)	36.12 ± 0.40	36.17 ± 0.25
Fed-IT ($\alpha = 0.8$, $\gamma = 0$)	35.18 ± 0.25	36.29 ± 0.16

4.5.3 Visualization of DNNs's Output with Different CMI Values

In this section, our objective is to conduct an experiment to verify that the CMI value of a DNN measures the concentration of the DNN's output centroid. To this end, we trained two DNNs on the CIFAR-10 dataset: one with high CMI values and another with low CMI values. Specifically, to train a DNN with low CMI value, we have incorporated a CMI loss into the DNN's loss function. Then, we utilized t-SNE [**?**] to project the 10-dimensional output probability space of each DNN onto a 2-dimensional plane. The visualization of this process is illustrated in Fig. 4.2. It is evident from the visualization that the clusters for the DNN with lower CMI values exhibit greater concentration and less dispersion.

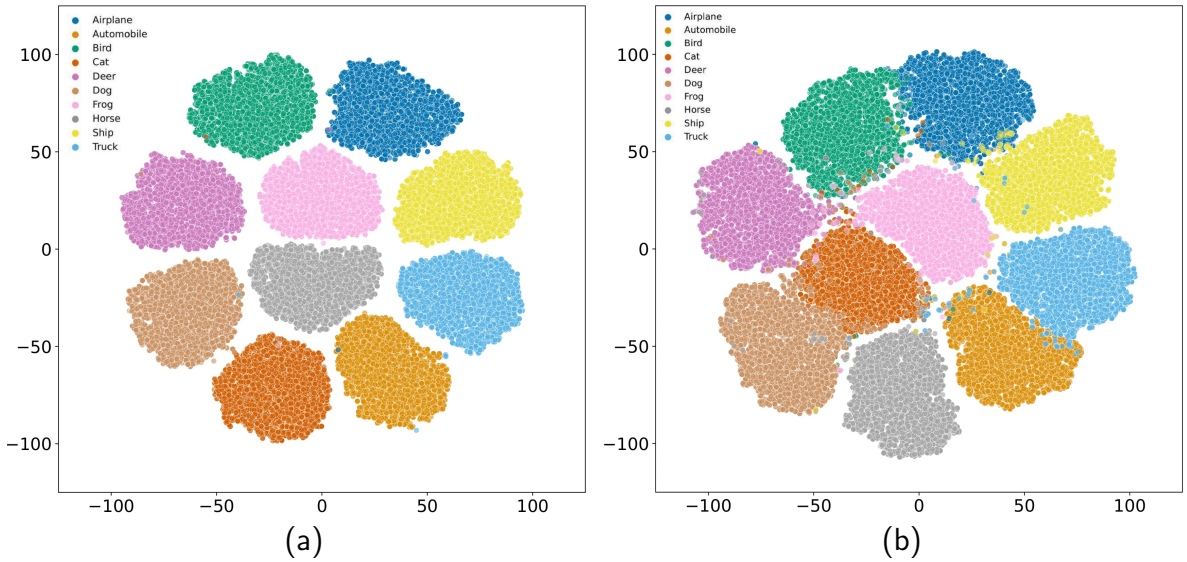


Figure 4.2: Visualization of the output probability space of the DNN with (a) low CMI value, and (b) high CMI value.

4.5.4 Fig. 4.3 with Different Classes

In this subsection, our goal is to demonstrate that the conclusion drawn from Fig. 4.3 is class-agnostic. In other words, it remains consistent regardless of the specific three classes chosen from the CIFAR-100 dataset. As long as the classes exhibit an imbalance, the same observation as depicted in Fig. 1 can be replicated.

To achieve this objective, we replicate the same figures as those presented in Fig. 4.3. However, we select three different sets of classes for visualization purposes. The resulting figures are depicted in Fig. 4.3.

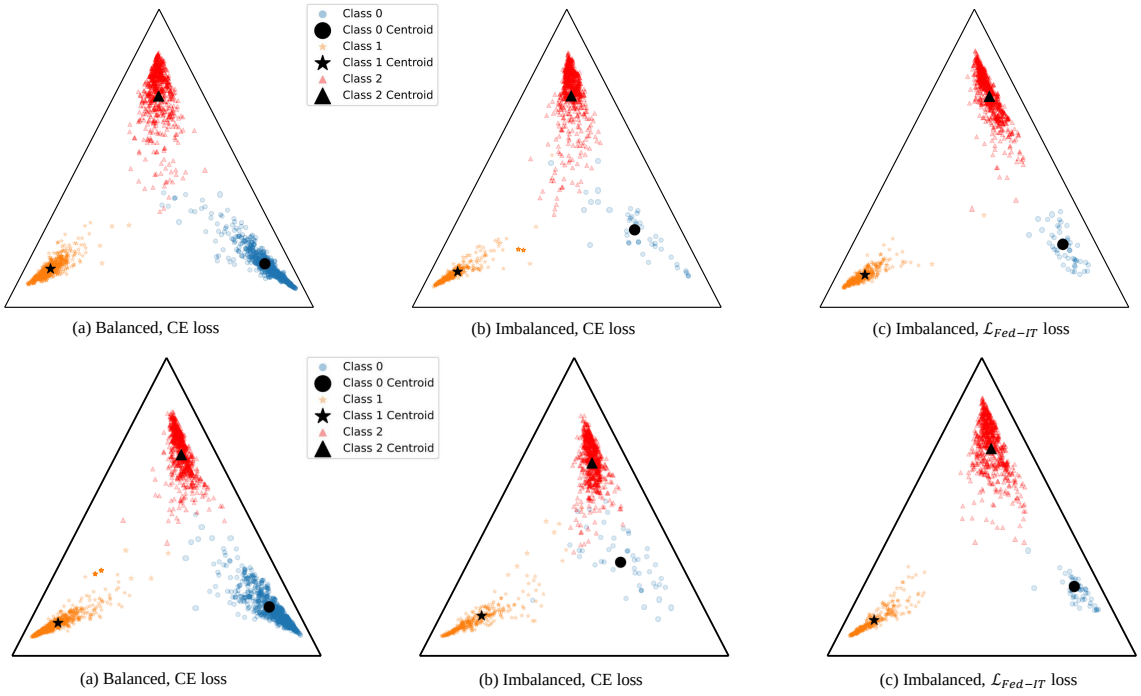


Figure 4.3: The output probability space of a DNN to three classes of CIFAR-100 dataset over the two-dimensional probability simplex. (a) The dataset is class-balanced and CE loss is used. (b) The dataset is class-imbalanced, and the ratio of samples in class 0, 1 and 2 is 1:5:10, and CE loss is used. (c) The dataset is class-imbalanced similar to (b), and loss in $\mathcal{L}_{\text{Fed-IT}}$.

4.6 Chapter Summary

In this chapter, by examining the output probability distribution clusters of local DNNs in the probability space over the label set, we identified two issues in FL for class-imbalanced datasets: (1) dispersion problem—clusters for minority classes tend to disperse, and (2) gravity problem—clusters for minority classes are drawn toward majority classes. To address these issues, we introduced an information-theoretic loss function for clients, and presented Fed-IT, a novel FL framework. Fed-IT outperforms previous methods while ensuring client privacy.

Chapter 5

Coded Deep Learning

Chapters 3 and 4 explored how information-theoretic concepts can be leveraged to enhance classification accuracy of deep neural networks (DNNs) in both centralized and distributed settings. In contrast, this chapter shifts the focus to applying information-theoretic principles to improve other aspects of DNN performance beyond just accuracy.

Specifically, this chapter proposes a new framework dubbed "coded deep learning" (CDL), which integrates information-theoretic coding concepts into the inner workings of DL, aiming to substantially compress model weights and activations, reduce computational complexity at both training and post-training inference stages, and enable efficient model/data parallelism. Specifically, within CDL,

- We first propose a novel probabilistic method for quantizing both model weights and activations, and its soft differentiable variant which offers an analytic formula for gradient calculation during training;
- Both the forward and backward passes during training are executed over quantized weights and activations, which eliminates a majority of floating-point operations and reduces the training computational complexity;
- During training, both weights and activations are entropy constrained so that they are compressible in an information-theoretic sense at any stage of training, which in turn reduces communication costs in cases where model/data parallelism are adopted;
- The trained model in CDL is in a quantized format by default with compressible quantized weights, reducing post-training inference complexity and model storage complexity.

Additionally, a variant of CDL, namely relaxed CDL (R-CDL), is presented to further improve the trade-off between validation accuracy and compression at the disadvantage of full precision operation involved in forward and backward passes during training with other advantageous features of CDL intact. Extensive empirical results show that CDL and R-CDL outperform the state-of-the-art algorithms in DNN compression in the literature.

The journal version of this chapter can be found in [96], and the corresponding patent is available at [220].

5.1 Introduction

Deep neural networks (DNNs) have demonstrated remarkable performance across diverse applications, from computer vision [84, 122] to natural language processing [39, 112]. The exceptional performance of these DNNs is largely attributed to their large model sizes, with high-performance models often containing hundreds of millions or even billions of parameters. However, this poses challenges for deployment in resource-limited environments due to (i) significant storage overhead, and (ii) intensive computational and memory requirements during both training and post-training inferences.

To partially address the above challenges and facilitate the efficient deployment of large DNNs on resource-limited devices, a multitude of model compression techniques have been proposed in the literature. These methods include quantization [13, 58, 154, 166], pruning [138, 139, 142, 246], and knowledge distillation [48, 70, 74, 87, 177, 194, 228]. Among these techniques, quantization methods have attracted significant attention due to their promising hardware compatibility across various architectures [104, 181].

Quantization can be conducted either during training or after training, with the former called quantization-aware training (QAT) [13, 29, 58], and the latter called post-training quantization (PTQ) [32]. Generally, PTQ methods often experience significant performance degradation when deployed for low-bit quantization. On the other hand, QAT methods proposed in the literature also face several drawbacks: (i) the non-differentiability of quantization functions used in QAT methods necessitates the use of gradient approximation techniques during training [11, 86], leading to inferior results; (ii) since they are often applied to pre-trained full-precision (FP) models[1] to control performance degradation [13, 58, 132, 155, 196], they depend on the availability of pre-trained FP models, which

[1] Although some QAT methods train models from scratch [29, 235], their performance in achieving low-bit quantization is typically inferior compared to methods that start from a pre-trained model (see Section 6.5).

may not always be available in the first place, and also introduce more training computation complexity even if pre-trained FP models are available; and (iii) more importantly, they often neglect the cost of weight and activation communication required during training in model and data parallelism that is crucial for handling massive DNNs [156].

To tackle the issues mentioned above, in this chapter we shall take a different approach to DNN compression. Specifically, by introducing information-theoretic coding concepts into the core principles of deep learning (DL), we aim to establish a new training framework, dubbed "coded deep learning" (CDL), to meet the following four *criteria* simultaneously:

• ***C1*** **[High validation accuracy]:** The new framework should preserve or even improve the classification accuracy achieved by models trained using conventional FP methods. Additionally, it should offer better accuracy-compression trade-offs compared to existing quantization methods in the literature.

• ***C2*** **[Reduced training and inference complexity]:** In the new framework, forward and backward passes during training should be executed on quantized weights and activations so that a majority of floating-point operations (FLOPs) can be replaced with low-precision computations, resulting in, possibly, accelerated training times, reduced memory usage, lower power consumption, and enhanced area efficiency [62, 247]. In addition, the trained model via the new framework should be, by default, in a quantized format so that the same benefits in terms of memory, power, area efficiency,etc. can be carried over to post-training inferences.

• ***C3*** **[Improved model and data parallelism]:** To train a massive model such as GPT-3 with 175 billion parameters [18, 156], two different modes of parallelism are often employed: (i) pipeline model parallelism [95, 118, 219], where model layers are distributed across multiple devices, necessitating Communication of activation functions between devices; and (ii) data parallelism [217], where the input dataset is partitioned and distributed among different devices, requiring communication of model weights. To have improved model/data parallelism, in the new framework, weights and activations should be compressible in an information-theoretic sense at any stage of training so that the cost of weight and activation communication in model/data parallelism can be reduced. This means that their statistical properties allow for encoding them into fewer bits on average, as determined by their entropy. This is distinct from heuristic or hardware-specific compression techniques, which may not explicitly rely on the underlying information content or entropy of the weights.

• ***C4*** **[Reduced storage complexity]:** In the new framework, the weights of the trained model should be compressible in an information-theoretic sense so that after encoding, they require less storage complexity.

How does CDL meet the aforementioned criteria simultaneously? The first key step is to introduce novel probabilistic quantizers each using a trainable conditional probability mass function (CPMF), in a way similar to rate distortion theory [12, 223]. These probabilistic quantizers are then used to quantize both model weights and activations during training, and model activations during post-training inferences to satisfy *C2*. Based on its distortion analysis, each of these probabilistic quantizers has a neat *soft deterministic* differentiable approximation. The latter, in turn, has a nice analytic formula for computing its partial derivatives with respect to (w.r.t.) its input θ and the quantization step size q. Since each of the probabilistic quantizers and its *soft deterministic* approximation are very close to each other, the analytic formula can be utilized as a proxy for gradient calculation during the training stage. Due to this analytic computation, models trained using CDL achieve validation accuracy surpassing that of QAT methods that primarily rely on coarse approximations of the gradients, thereby satisfying *C1* effectively.

The second key step is to incorporate information-theoretic coding ideas into the inner workings of DL to make model weights and activations compressible at any stage of training. To this end, given that in CDL, weights and activations are quantized using the proposed probabilistic quantizers, the marginal probability mass functions (MPMFs) of randomly quantized weights and activations for each layer are calculated, and the entropies of these MPMFs are then constrained during training. In other words, in CDL, both the cross-entropy (CE) loss used in DL and entropies of these MPMFs are minimized jointly. Since the compression rates of quantized weights and activations are roughly equal to the entropies of these MPMFs if entropy coding (such as Huffman coding) is used, by minimizing these entropies during training, we make sure that both weights and activations are compressible in an information-theoretic sense at any stage of training, thereby satisfying both *C3* and *C4*.

To further improve the trade-off between validation accuracy and compression, CDL is modified by replacing each of the probabilistic quantizers with its *soft deterministic* approximation, yielding a variant referred to as relaxed-CDL (R-CDL). Except that forward and backward passes are now operated with full precision, all other features of CDL remain there in R-CDL. In particular, weights and activations are still compressible at any stage of training in R-CDL. They are quantized using the newly obtained CPMFs whenever the inter-device weight and activation communication need arises in model/data parallelism; weights are still quantized at the end of training. In comparison with CDL, R-CDL has the advantage that all gradients can now be computed analytically without approximation during training at the expense of FP forward and backward passes. This advantage helps R-CDL deliver better accuracy-compression performance than CDL.

We have conducted a comprehensive set of experiments on CIFAR-100 [121] and Ima-

geNet datasets [122]. It is shown that CDL and R-CDL offer superior performance compared to state-of-the-art (SOTA) QAT methods in the literature. The remainder of this chapter is organized as follows. Section 5.2 briefly reviews related works. Section 5.3 first introduces the notation to be used throughout the chapter, and then discusses the preliminary knowledge based on which CDL is established. In Section 5.4, we present the novel probabilistic quantization method along with its differentiable *soft deterministic* variant. Section 5.5 describes CDL and R-CDL in details. In addition, extensive experimental results on some datasets along with comparison with SOTA methods are reported in Section 6.5, and finally Section 5.8 provides a summary the chapter.

5.2 Related Works

5.2.1 Quantization-Aware Training (QAT)

In QAT, weights and activations are quantized, and the forward and backward passes are performed over quantized weights and activations, while the original floating-point weights are updated [58,174]. A fundamental challenge in the QAT formulation stems from the lack of an interpretable gradient for the quantization function, making gradient-based training impractical without ad hoc approximation. One widely adopted approach to mitigate this challenge is, by employing the straight-through estimator (STE) [11, 86], to approximate the true gradient during training. Essentially, this involves approximating the gradient of the rounding operator as 1 within the quantization limits [58, 248].

The STE gradient approximation has gained widespread adoption in recent literature, effectively narrowing the accuracy gap between quantized and full-precision models across various tasks and network architectures [13, 58, 67, 100, 120, 154, 248].

Some studies showed that the derivative of the clipped ReLU provides a superior approximation compared to the vanilla STE (derivative of identity) [19, 229]. Notably, the former effectively sets the gradient outside the quantization grid to zero, and has become the predominant formulation in QAT literature [29, 58, 199]. In addition, much prior research concentrated on highlighting the sub-optimality of the STE as a gradient approximator, and proposed some alternative gradient estimators which could be mainly categorized into two types: (i) multiplicative methods which apply a scaling to the gradient vectors [63, 99, 114, 225], and (ii) additive methods that add an input-independent term to the gradient [23, 26, 81, 165].

Nonetheless, while these approaches offer alternatives to STE, they all rely on certain

approximations that can result in sub-optimal results. In addition, neither weight entropy nor activation entropy is considered.

5.2.2 Accounting for Compression during Training

Some papers explicitly account for compression while training DNNs. For instance, [2] introduced a regularization term that encourages the parameter matrix of each layer to have low rank during training so that some parameters of the DNN can be pruned in the post-processing stage. Similarly, the works in [5,7] introduced other types of regularization terms to train compressible DNNs.

In addition, the authors in [27] showed that uniform distribution of weights are more robust to quantization than normally distributed weights. To this end, they introduced KURE (KUrtosis REgularization) to minimize the kurtosis of weights and ensure a uniform distribution. [82] proposed a method to constrain weights into predefined bins based on the quantization bit-width. However, selecting these bins can be challenging, and the model's performance is highly sensitive to the bin selection. [124] introduced range regularization, which imposes constraints on the weights to ensure they fall within predefined value ranges.

However, in general, the methodologies proposed by these works lack analytic justifications and primarily rely on empirical observations. More importantly, they do not satisfy all four criteria *C1-C4* simultaneously as achieved by CDL.

5.2.3 Modes of Parallelism

Training large models poses significant challenges for two reasons: (i) it is no longer feasible to accommodate the parameters of these models in the main memory, even with the largest GPU configurations available (such as NVIDIA's 80GB-A100 cards); (ii) even if the model can be fit into a single GPU (e.g., by utilizing techniques like swapping parameters between host and device memory as in ZeRO-Offload [172]), the high volume of compute operations required can lead to unrealistically long training times (for instance, training GPT-3 with 175 billion parameters [18] would necessitate approximately 288 years using a single V100 GPU [156]). This underscores the need for parallelism.

Here, we provide a brief overview of parallelism techniques; specifically, we discuss two key approaches: model parallelism and data parallelism.

Model parallelism

Pipeline model parallelism divides the model into multiple chunks of layers and distributes them across different devices [95,118,219]. During the forward pass, the activation outputs from one stage are sent to the next stage on a different device. During the backward pass, gradients are propagated back through the pipeline in a similar fashion.

Nonetheless, this approach incurs communication overhead between devices when sending activations between pipeline stages. This can potentially negate some of the speedup benefits, and can impact performance, particularly on systems with slower network connections.

Data parallelism

In data parallelism, the DNN is replicated across multiple devices [10, 44]. The training dataset is divided into mini-batches. Each device receives different mini-batches. Devices simultaneously process their assigned mini-batches, computing weight updates (or gradients) for the same model [71–73,75–77,80,151]. Then, the devices update their local models by averaging the weights/gradients received from all the devices.

To facilitate model/data parallelism, it is desirable for weights and activations to be compressible at any stage of training.

5.3 Notation and Preliminaries

5.3.1 Notation Used in This Chapter

For a positive integer N, let $[N] \triangleq \{1, \ldots, N\}$. Scalars are denoted by lowercase letters (e.g., w), and vectors by bold-face letters (e.g., $\boldsymbol{w}$). The i-th element of vector $\boldsymbol{w}$ is denoted by $\boldsymbol{w}[i]$. Also, the length of vector $\boldsymbol{w}$ is denoted by $|\boldsymbol{w}|$.

The probability of event E is denoted by $P(E)$. In addition, for a discrete random variable X, its probability mass function, expected value, and variance are denoted by $\mathbb{P}_X$, $\mathbb{E}\{X\}$, and $\mathrm{Var}\{X\}$, respectively. For a C-dimensional probability distribution P, denote by $\mathsf{H}(P) = \sum_{c\in[C]} -P[c] \log P[c]$, the Shannon entropy of P. The softmax operation over the vector $\boldsymbol{w}$ is denoted by $\boldsymbol{\sigma}(\boldsymbol{w})$. That is, for any $i \in [|\boldsymbol{w}|]$

$$\boldsymbol{\sigma}(\boldsymbol{w})[i] = \frac{e^{\boldsymbol{w}[i]}}{\sum_{j\in[|\boldsymbol{w}|]} e^{\boldsymbol{w}[j]}}.$$

5.3.2 Preliminaries

In ordinary multi-class classification, let $\mathcal{X} \subseteq \mathbb{R}^d$ be the instance space, and $\mathcal{Y} = [C]$ be the label space, where C is the number of classes.

A DNN is a cascade of L layers. For both fully-connected and convolutional layers, denote by $\boldsymbol{w}_l$ the flattened vector representation of weight parameters at layer l. Hence the number of weight parameters at layer l equals $|\boldsymbol{w}_l|$. In addition, $\boldsymbol{w} = \{\boldsymbol{w}_l\}_{l\in[L]}$ represents all the parameters in the DNN.

The input to the first layer of the DNN is a raw sample $\boldsymbol{x}_0 \in \mathcal{X}$. The output of the l-th layer, which is also called an activation map, is denoted by vector $\boldsymbol{x}_l$. Hence, the number of activations at the output of the l-th layer is equal to $|\boldsymbol{x}_l|$, and the output logit vector is represented by $\boldsymbol{x}_L$. Let $\boldsymbol{x} = \{\boldsymbol{x}_l\}_{l=1}^{L-1}$.

Let (X, Y) be a pair of random variables, the distribution of which governs a training set, where $X \in \mathcal{X}$ represents the raw input sample and Y is the ground truth label of X. In traditional DL, one trains a classifier by solving the following minimization problem

$$\min_{\boldsymbol{w}} \mathbb{E}_{(X,Y)}\{\mathcal{L}\left(X, Y, \boldsymbol{w}\right)\}, \tag{5.1}$$

where $\mathcal{L}(\boldsymbol{x}_0, y, \boldsymbol{w})$ is a real-valued loss function (typically the CE loss).

5.4 Probabilistic Quantization Method

In this section, we first propose a probabilistic quantization method in Subsection 5.4.1 that forms the foundation for the CDL framework. Next, in Subsection 5.4.2, we introduce its *soft deterministic* approximation.

5.4.1 Probabilistic Quantizer $\mathsf{Q}_{\mathrm{p}}(\cdot)$

Consider b-bit uniform quantization.

Remark. *The proposed probabilistic quantization method can be easily extended to non-uniform quantization (see section 5.7).*

In general, in this chapter we intentionally focused on basic uniform quantization as the primary quantization method to demonstrate the core principles and effectiveness of the CDL framework. The choice of uniform quantization allows for a straightforward analysis

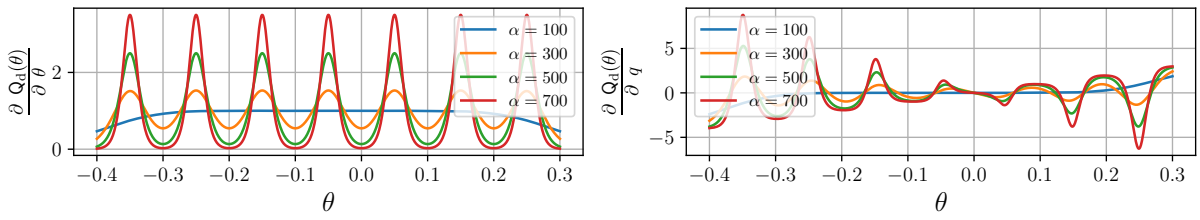


Figure 5.1: Illustration of the partial derivatives of $\mathsf{Q}_\mathrm{d}(\theta)$ w.r.t. θ (left), and q (right) for $\alpha = \{100, 300, 500, 700\}$, where b and q equal 3 and 0.1, respectively.

and implementation while maintaining compatibility with a wide range of DNN architectures.

That said, the CDL method itself is general and not restricted to uniform quantization. It can be seamlessly integrated with other advanced quantization techniques, including lattice vector quantization (LVQ). Incorporating LVQ or similar sophisticated methods into the CDL framework could further enhance its performance, particularly in scenarios requiring higher precision or tailored quantization strategies.

Future work could explore how CDL performs when combined with LVQ or other advanced quantization schemes to provide a more comprehensive comparison and unlock additional benefits of the framework. I appreciate your suggestion and will consider it for extending this research.

Let $\mathcal{A}$ denote its corresponding quantization index set

$$\mathcal{A} = \{-2^{b-1}, -2^{b-1}+1, \ldots, 2^{b-1}-2, 2^{b-1}-1\}. \tag{5.2}$$

For convenience, $\mathcal{A}$ is also regarded as a vector of dimension 2^b. Multiplying $\mathcal{A}$ by a quantization step-size q, we get the corresponding reproduction alphabet

$$\hat{\mathcal{A}} = q \times [-2^{b-1}, -2^{b-1}+1, \ldots, 2^{b-1}-2, 2^{b-1}-1]. \tag{5.3}$$

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

Let θ represent either a weight parameter or an activation of a DNN. We want to quantize θ randomly to $\hat{\theta} \in \hat{\mathcal{A}}$. To this end, we introduce a trainable CPMF $P_\alpha(\cdot|\theta)$ over the reproduction alphabet $\hat{\mathcal{A}}$ or equivalently the index set $\mathcal{A}$ given θ, where $\alpha > 0$ is a trainable parameter:

$$P_\alpha(\hat{\theta}|\theta) = \frac{e^{-\alpha(\theta-\hat{\theta})^2}}{\sum_{j\in\mathcal{A}} e^{-\alpha(\theta-jq)^2}},\ \forall \hat{\theta} = iq,\ i \in \mathcal{A}. \tag{5.4}$$

The specific form $e^{-\alpha(\theta-\hat{\theta})^2}$ intuitively ensures that values of $\hat{\theta}$ closer to θ are assigned higher probabilities, while values farther away are exponentially less likely. This behavior closely mirrors a Gaussian-like distribution, which makes it a natural and effective choice for modeling uncertainties around θ.

For instance, one could replace $e^{-\alpha(\theta-\hat{\theta})^2}$ with $e^{-\alpha|\theta-\hat{\theta}|}$, which follows a Laplacian distribution. While this alternative emphasizes linear deviations rather than quadratic ones, it introduces sharper transitions between probabilities. The quadratic form used in the current framework provides smoother gradients, which are particularly advantageous for stable and efficient optimization during training.

Remark. *The probabilistic quantizer handles extreme values outside the predefined dynamic range similarly to other values, by assigning probabilities to all points in the reproduction alphabet based on their distance to the extreme value. However, since extreme values lie far outside the range, the probabilities for most points in the alphabet become negligible, except for those closest to the dynamic range boundaries. As a result, the quantizer most likely maps these extreme values to the minimum or maximum values in the alphabet, effectively "clipping" them to the nearest valid point.*

Define the following notation

$$\big[\theta\big]_{2^b} = [\underbrace{\theta, \ldots, \theta}_{2^b \text{ times}}]. \tag{5.5}$$

Regard the CPMF $P_\alpha(\cdot|\theta)$ as a vector of dimension 2^b. Then it can also be easily computed via the softmax operation:

$$\big[P_\alpha(\cdot|\theta)\big] = \boldsymbol{\sigma}\Big(-\alpha \times \Big(\big[\theta\big]_{2^b} - \hat{\mathcal{A}}\Big)^2 \Big), \tag{5.6}$$

where $\boldsymbol{\sigma}(\cdot)$ denotes the softmax operation.

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

$$\hat{\theta} = \mathsf{Q}_{\mathrm{p}}(\theta). \tag{5.7}$$

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

Remark. *Note that although $\mathsf{Q}_{\mathrm{p}}(\theta)$ is a probabilistic quantizer, it quantizes θ into the nearest point in $\hat{\mathcal{A}}$ with probability approaching 1 as $\alpha \to \infty$. Therefore, when α is large, $\mathsf{Q}_{\mathrm{p}}(\theta)$ is equal to the uniformly quantized value with high probability.*

When the probabilistic quantizer $\mathsf{Q}_{\mathrm{p}}(\theta)$ is employed in DL, how would one compute the gradients w.r.t. θ and q during backward passes? This is addressed in the next subsection.

5.4.2 *Soft Deterministic* Quantizer $\mathsf{Q}_{\mathrm{d}}(\cdot)$

Given θ, let us first analyze the expected distortion (defined as the squared error) between θ and $\mathsf{Q}_{\mathrm{p}}(\theta)$. To this end, consider the conditional expectation of $\hat{\theta}$ given θ, i.e.,

$$\mathsf{Q}_{\mathrm{d}}(\theta) = \mathbb{E}\{\hat{\theta}|\theta\} = \sum_{i\in\mathcal{A}} P_\alpha(iq|\theta)\ iq. \tag{5.8}$$

We have the following proposition.

Proposition 1. *For any θ and $\alpha > 0$,*

$$\mathbb{E}\big\{(\theta - \mathsf{Q}_{\mathrm{p}}(\theta))^2 \mid \theta\big\} = (\theta - \mathsf{Q}_{\mathrm{d}}(\theta))^2 + Var\big\{\mathsf{Q}_{\mathrm{p}}(\theta) \mid \theta\big\},$$

where $Var\big\{\mathsf{Q}_{\mathrm{p}}(\theta) \mid \theta\big\}$ is the conditional variance of $\mathsf{Q}_{\mathrm{p}}(\theta)$ given θ.

Proof. Note that in view of (5.7) and (5.8),

$$\mathsf{Q}_{\mathrm{d}}(\theta) = \mathbb{E}\big[\mathsf{Q}_{\mathrm{p}}(\theta)|\theta\big]. \tag{5.9}$$

Then it follows that

$$\begin{aligned}
&\mathbb{E}\big[(\theta - \mathsf{Q}_{\mathrm{p}}(\theta))^2 \mid \theta\big] && (5.10)\\
&= \mathbb{E}\big[\big(\theta - \mathsf{Q}_{\mathrm{d}}(\theta) + \mathsf{Q}_{\mathrm{d}}(\theta) - \mathsf{Q}_{\mathrm{p}}(\theta)\big)^2|\theta\big] && (5.11)\\
&= \mathbb{E}\big[\big(\theta - \mathsf{Q}_{\mathrm{d}}(\theta)\big)^2|\theta\big] + \mathbb{E}\big[\big(\mathsf{Q}_{\mathrm{p}}(\theta) - \mathsf{Q}_{\mathrm{d}}(\theta)\big)^2|\theta\big] + 2\mathbb{E}\big[(\theta - \mathsf{Q}_{\mathrm{d}}(\theta))(\mathsf{Q}_{\mathrm{d}}(\theta) - \mathsf{Q}_{\mathrm{p}}(\theta))|\theta\big]\\
&= \big(\theta - \mathsf{Q}_{\mathrm{d}}(\theta)\big)^2 + \mathrm{Var}\big\{\mathsf{Q}_{\mathrm{p}}(\theta) \mid \theta\big\} + 2(\theta - \mathsf{Q}_{\mathrm{d}}(\theta))\mathbb{E}\big[(\mathsf{Q}_{\mathrm{d}}(\theta) - \mathsf{Q}_{\mathrm{p}}(\theta))|\theta\big]\\
&= \big(\theta - \mathsf{Q}_{\mathrm{d}}(\theta)\big)^2 + \mathrm{Var}\big\{\mathsf{Q}_{\mathrm{p}}(\theta) \mid \theta\big\}, && (5.12)
\end{aligned}$$

where the last equality (5.12) is due to (5.9). This completes the proof of Proposition 1. □

When α is sufficiently large, the conditional variance $\mathrm{Var}\big\{\mathsf{Q}_{\mathrm{p}}(\theta) \mid \theta\big\}$ becomes negligible. Thus, as per Proposition 1, the expected distortion of the probabilistic quantizer is approximately equal to the distortion between θ and $\mathsf{Q}_{\mathrm{d}}(\theta)$. Note that $\mathsf{Q}_{\mathrm{d}}(\theta)$ is deterministic. This suggests that one may use $\mathsf{Q}_{\mathrm{d}}(\theta)$ to approximate the probabilistic quantizer $\mathsf{Q}_{\mathrm{p}}(\theta)$. We refer to $\mathsf{Q}_{\mathrm{d}}(\cdot)$ as a *soft deterministic* quantizer or approximation. The quantization is *soft* in the sense that the output is not strictly in the reproduction alphabet $\hat{\mathcal{A}}$, but rather maintains a degree of continuous representation.

In view of (5.8) and (5.4), the *soft deterministic* quantizer $\mathsf{Q}_{\mathrm{d}}(\cdot)$ as a function of θ and q is analytic and has nice analytic formulas for computing its partial derivatives w.r.t. θ and q, as shown in Lemma 1 below.

Lemma 1. *The partial derivatives of* $\mathsf{Q}_{\mathrm{d}}(\theta)$ *w.r.t.* q *and* θ *are obtained as*

$$\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial \theta} = 2\alpha\, Var\big\{\mathsf{Q}_{\mathrm{p}}(\theta)\big\}, \tag{5.13a}$$

$$\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial q} = \frac{1}{q}\Big(\mathbb{E}\big\{\mathsf{Q}_{\mathrm{p}}(\theta)\big\} + (2\alpha\theta)\, Var\big\{\mathsf{Q}_{\mathrm{p}}(\theta)\big\} Skew_{\mathrm{u}}\big\{\mathsf{Q}_{\mathrm{p}}(\theta)\big\}\Big), \tag{5.13b}$$

where for any random variable X*,*

$$Skew_{\mathrm{u}}(X) \triangleq \sum_x x^3 \mathbb{P}_X(x) - \Big(\sum_x x\mathbb{P}_X(x)\Big)\Big(\sum_x x^2\mathbb{P}_X(x)\Big).$$

Proof. To find $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial \theta}$, we have

$$\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial \theta} = \sum_{i\in\mathcal{A}} iq \frac{\partial P_\alpha(iq|\theta)}{\partial \theta} \tag{5.14}$$

To continue, it follows that

$$\begin{aligned}\frac{\partial P_\alpha(iq|\theta)}{\partial \theta} &= \frac{-2\alpha e^{-\alpha(\theta-iq)^2}(\theta-iq)}{\sum_{j\in\mathcal{A}} e^{-\alpha(\theta-jq)^2}} \\ &\qquad + \frac{e^{-\alpha(\theta-iq)^2}\sum_{j\in\mathcal{A}} 2\alpha e^{-\alpha(\theta-jq)^2}(\theta-jq)}{\big(\sum_{j\in\mathcal{A}} e^{-\alpha(\theta-jq)^2}\big)^2} \\ &= -2\alpha P_\alpha(iq|\theta)(\theta-iq) + 2\alpha P_\alpha(iq|\theta)\sum_{j\in\mathcal{A}}(\theta-jq)P_\alpha(jq|\theta).\end{aligned} \tag{5.15}$$

Then, by plugging (5.15) into (5.14), we obtain

$$\begin{aligned}
\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial \theta} &= -2\alpha q \sum_{i\in\mathcal{A}} i(\theta - iq) P_\alpha(iq|\theta) \\
&\quad + 2\alpha q \Big(\sum_{i\in\mathcal{A}} i P_\alpha(iq|\theta) \Big) \Big(\sum_{j\in\mathcal{A}} (\theta - jq) P_\alpha(jq|\theta) \Big) \\
&= -2\alpha q\theta \sum_{i\in\mathcal{A}} i P_\alpha(iq|\theta) + 2\alpha \sum_{i\in\mathcal{A}} (iq)^2 P_\alpha(iq|\theta) \\
&\quad + 2\alpha q \Big(\sum_{i\in\mathcal{A}} i P_\alpha(iq|\theta) \Big) \Big(\theta - \sum_{j\in\mathcal{A}} jq P_\alpha(jq|\theta) \Big) \\
&= 2\alpha \sum_{i\in\mathcal{A}} (iq)^2 P_\alpha(iq|\theta) \\
&\quad - 2\alpha \Big(\sum_{i\in\mathcal{A}} iq P_\alpha(iq|\theta) \Big) \Big(\sum_{j\in\mathcal{A}} jq P_\alpha(jq|\theta) \Big) \\
&= 2\alpha \mathbb{E}\Big\{ \Big(\mathsf{Q}_{\mathrm{p}}(\theta) \Big)^2 \Big\} - 2\alpha \Big(\mathbb{E}\big\{ \mathsf{Q}_{\mathrm{p}}(\theta) \big\} \Big) \Big(\mathbb{E}\big\{ \mathsf{Q}_{\mathrm{p}}(\theta) \big\} \Big) \\
&= 2\alpha \mathrm{Var}\big\{ \mathsf{Q}_{\mathrm{p}}(\theta) \big\}. \qquad (5.16)
\end{aligned}$$

On the other hand, to find $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial q}$, we have

$$\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial q} = \sum_{i\in\mathcal{A}} i P_\alpha(iq|\theta) + \sum_{i\in\mathcal{A}} iq \frac{\partial P_\alpha(iq|\theta)}{\partial q}. \qquad (5.17)$$

In addition,

$$\begin{aligned}
\frac{\partial P_\alpha(iq|\theta)}{\partial q} &= \frac{2\alpha e^{-\alpha(\theta - iq)^2} i(\theta - iq)}{\sum_{j\in\mathcal{A}} e^{-\alpha(\theta - jq)^2}} \\
&\qquad - \frac{e^{-\alpha(\theta - iq)^2} \sum_{j\in\mathcal{A}} 2\alpha e^{-\alpha(\theta - jq)^2} j(\theta - jq)}{\big(\sum_{j\in\mathcal{A}} e^{-\alpha(\theta - jq)^2} \big)^2} \\
&= 2\alpha i P_\alpha(iq|\theta)(\theta - iq) - 2\alpha P_\alpha(iq|\theta) \sum_{j\in\mathcal{A}} j(\theta - jq) P_\alpha(jq|\theta). \qquad (5.18)
\end{aligned}$$

Plugging (5.18) in (5.17) yields

$$
\begin{aligned}
\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial q} &= \sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta) + 2\alpha q \sum_{i\in\mathcal{A}} i^2(\theta - iq)P_\alpha(iq|\theta) \\
&\quad - 2\alpha q\Big(\sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta)\Big)\Big(\sum_{j\in\mathcal{A}} j(\theta - jq)P_\alpha(jq|\theta)\Big) \\
&= \sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta) + 2\alpha q\theta \sum_{i\in\mathcal{A}} i^2P_\alpha(iq|\theta) - 2\alpha q^2 \sum_{i\in\mathcal{A}} i^3P_\alpha(iq|\theta) \\
&- 2\alpha q\theta\Big(\sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta)\Big)\Big(\sum_{j\in\mathcal{A}} jP_\alpha(jq|\theta)\Big) \\
&\quad + 2\alpha q^2\Big(\sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta)\Big)\Big(\sum_{j\in\mathcal{A}} j^2P_\alpha(jq|\theta)\Big) \\
&= \sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta) + 2\alpha q\theta\Big(\sum_{i\in\mathcal{A}} i^2P_\alpha(iq|\theta) - \big(\sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta)\big)^2\Big) \\
&\quad - 2\alpha q^2\Big(\sum_{i\in\mathcal{A}} i^3P_\alpha(iq|\theta) - \big(\sum_{i\in\mathcal{A}} iP_\alpha(iq|\theta)\big)\big(\sum_{i\in\mathcal{A}} i^2P_\alpha(iq|\theta)\big)\Big) \\
&= \frac{1}{q}\left(\mathbb{E}\{\mathsf{Q}_{\mathrm{p}}(\theta)\} + 2\alpha\theta\mathrm{Var}\{\mathsf{Q}_{\mathrm{p}}(\theta)\} - 2\alpha\mathrm{Skew}_{\mathrm{u}}\{\mathsf{Q}_{\mathrm{p}}(\theta)\}\right), \qquad (5.19)
\end{aligned}
$$

which concludes the proof. □

To grasp a better understanding, $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial \theta}$ and $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial q}$ are depicted in Fig. 5.1 for different values of α. For each θ, denote its nearest value in $\hat{\mathcal{A}}$ by $\lfloor\theta\rceil_q$. It is informative to note from Fig. 5.1 that as θ moves away from $\lfloor\theta\rceil_q$ in either direction, $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\theta)}{\partial \theta}$ increases. This important property bodes well with quantization and helps the training process push θ towards points inside $\hat{\mathcal{A}}$, which is more or less confirmed by Fig. 5.9 in Section 6.5.

We conclude this section by comparing the *soft deterministic* quantizer $\mathsf{Q}_{\mathrm{d}}(\theta)$ with the real uniform quantizer $\mathsf{Q}_{\mathrm{u}}(\theta) = \lfloor\theta\rceil_q$. Fig. 5.2 depicts the curves of $\mathsf{Q}_{\mathrm{u}}(\theta)$ and $\mathsf{Q}_{\mathrm{d}}(\theta)$ for different α values. As observed, as α increases, $\mathsf{Q}_{\mathrm{d}}(\cdot)$ converges to the uniform quantizer.

5.5 CDL and R-CDL

This section is devoted to the detailed description of the inner workings of CDL and R-CDL. As shown in Fig. 5.3, key elements of CDL include: (i) incorporating trainable probabilistic quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ to quantize both model weights and activations; (ii) enforcing entropy constraints on quantized weights and activations during training, and (iii)

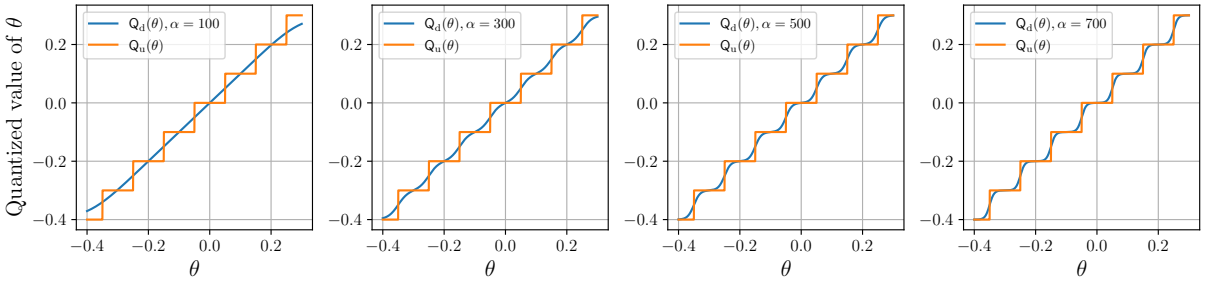


Figure 5.2: Illustration of $\mathsf{Q}_{\mathrm{u}}(\cdot)$ vs $\mathsf{Q}_{\mathrm{d}}(\cdot)$ with $\alpha = \{100, 300, 500, 700\}$, where b and q are set to 3 and 0.1, respectively.

using the *soft deterministic* quantizers $\mathsf{Q}_{\mathrm{d}}(\cdot)$ corresponding to $\mathsf{Q}_{\mathrm{p}}(\cdot)$ as a proxy to facilitate gradient computation in backpropagation. To facilitate our discussion and make it easier to understand, we decompose it into several parts: (i) incorporation of trainable probabilistic quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ into CDL; (ii) computation of gradients of the loss function without entropy constraint via backpropagation deploying $\mathsf{Q}_{\mathrm{d}}(\cdot)$; (iii) entropy computation of quantized weights and activations; (iv) new objective function with entropy constraints; (v) new overall learning algorithm; and (vi) R-CDL with trainable quantizers $\mathsf{Q}_{\mathrm{d}}(\cdot)$.

5.5.1 Incorporating Trainable Probabilistic Quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ Into CDL

Consider b-bit precision quantization. In CDL, both weights and activations are quantized using trainable probabilistic quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ with the following specifications.

First, due to the adoption of ReLU function in DNN architecture, activations take only non-negative values. As such, we use the following sets $\mathcal{A}$ and $\mathcal{B}$ as the quantization index sets for weights and activations, respectively

$$\mathcal{A} = \{-2^{b-1}, -2^{b-1}+1, \ldots, 2^{b-1}-2, 2^{b-1}-1\}, \tag{5.20a}$$

$$\mathcal{B} = \{0, 1, \ldots, 2^{b}-2, 2^{b}-1\}. \tag{5.20b}$$

Second, to accommodate different characteristics of different layers, we use different trainable probabilistic quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ for different layers. Let q_l and s_l denote the quantization step-sizes for weights and activations at layer l, $l \in [L]$, respectively. Then the corresponding reproduction alphabets for weights $\boldsymbol{w}_l$ and activations $\boldsymbol{x}_l$ at layer l, become

$$\hat{\mathcal{A}}_l = q_l \times [-2^{b-1}, -2^{b-1}+1, \ldots, 2^{b-1}-2, 2^{b-1}-1], \tag{5.21a}$$

$$\hat{\mathcal{B}}_l = s_l \times [0, 1, \ldots, 2^b-2, 2^b-1]. \tag{5.21b}$$

Finally, as per (5.6), we deploy (i) a probabilistic quantizer $\mathsf{Q}_{\mathrm{p}}(\cdot)$ with trainable parameters $q_l > 0$ and $\alpha_l > 0$ to quantize each entry of $\boldsymbol{w}_l$ according to the following CPMF

$$\big[P_{\alpha_l}(\cdot|\boldsymbol{w}_l[i])\big] = \boldsymbol{\sigma}\Big(-\alpha_l \times \big(\big[\boldsymbol{w}_l[i]\big]_{2^b} - \hat{\mathcal{A}}_l\big)^2\Big), \tag{5.22}$$

and (ii) another probabilistic quantizer $\mathsf{Q}_{\mathrm{p}}(\cdot)$ with trainable parameters $s_l > 0$ and $\beta_l > 0$ to quantize each entry of $\boldsymbol{x}_l$ according to the following CPMF

$$\big[P_{\beta_l}(\cdot|\boldsymbol{x}_l[j])\big] = \boldsymbol{\sigma}\Big(-\beta_l \times \big(\big[\boldsymbol{x}_l[j]\big]_{2^b} - \hat{\mathcal{B}}_l\big)^2\Big). \tag{5.23}$$

Remark. *Note that the activations at the last layer are not required to be quantized. As such, trainable quantization parameters are* $\boldsymbol{q} = \{q_l\}_{l\in[L]}$, $\boldsymbol{s} = \{s_l\}_{l\in[L-1]}$, $\boldsymbol{\alpha} = \{\alpha_l\}_{l\in[L]}$, *and* $\boldsymbol{\beta} = \{\beta_l\}_{l\in[L-1]}$, *which will be learned jointly with weights* $\boldsymbol{w}$ *during the training process.*

Henceforth, the training calculation (forward and backward) passes are performed over these quantized weights and activations. Accordingly, with the incorporation of probabilistic quantizers, the conventional loss in DL becomes

$$\mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big), \tag{5.24}$$

where the notation $\{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}$ implies that both weights and activations are quantized according to $\mathsf{Q}_{\mathrm{p}}(\cdot)$ operation.

5.5.2 Computation of Gradients of the Loss Function (5.24)

The loss function (5.24) involves random jumps from $\boldsymbol{w}_l[i]$ to $\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}_l[i])$ and from $\boldsymbol{x}_l[j]$ to $\mathsf{Q}_{\mathrm{p}}(\boldsymbol{x}_l[j])$ and hence is not differentiable. How to compute its gradients with respect to $\boldsymbol{w}$, $\boldsymbol{x}$, $\boldsymbol{q}$, $\boldsymbol{s}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$? To overcome this difficulty, we invoke Proposition 1 and Lemma 1, and use $\mathsf{Q}_{\mathrm{d}}(\cdot)$ as a proxy for gradient calculation as explained below.

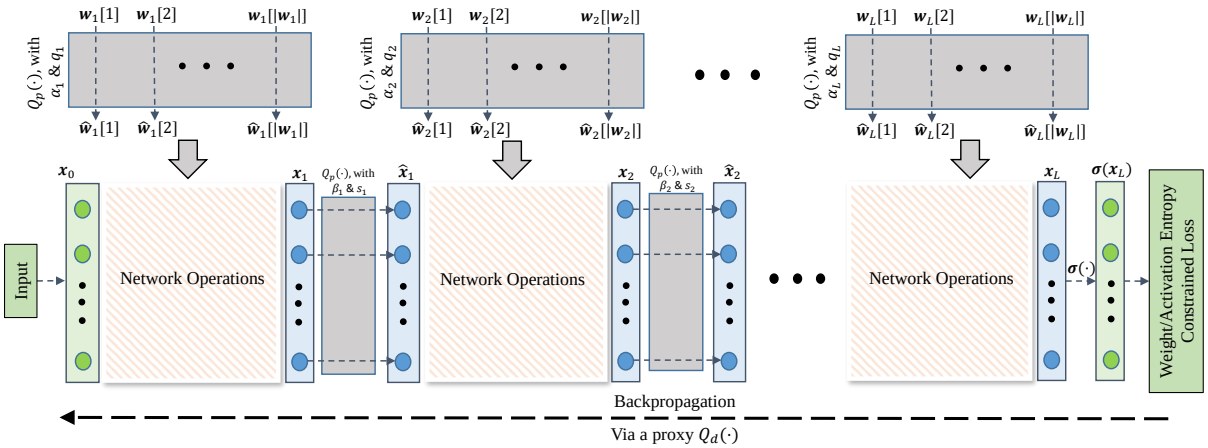


Figure 5.3: Illustration of the CDL's mechanism.

Firstly, following (5.8), the quantized values of $\boldsymbol{w}_l[i]$ and $\boldsymbol{x}_l[j]$ by the *soft deterministic* quantizers $\mathsf{Q}_\mathrm{d}(\cdot)$ are

$$\mathsf{Q}_\mathrm{d}(\boldsymbol{w}_l[i]) = \mathbb{E}\{\hat{w}|\boldsymbol{w}_l[i]\} = \sum_{j\in\mathcal{A}} P_{\alpha_l}(jq_l|\boldsymbol{w}_l[i])\ jq_l, \tag{5.25a}$$

$$\mathsf{Q}_\mathrm{d}(\boldsymbol{x}_l[j]) = \mathbb{E}\{\hat{x}|\boldsymbol{x}_l[j]\} = \sum_{i\in\mathcal{B}} P_{\beta_l}(is_l|\boldsymbol{x}_l[j])\ is_l. \tag{5.25b}$$

Next, we compute the partial derivatives of $\mathcal{L}(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\})$ w.r.t. $\boldsymbol{w}_l[i]$. To this end, we first need to find the partial derivatives w.r.t. $\boldsymbol{x}_l[j]$ as follows

$$\frac{\partial\mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial\boldsymbol{x}_l[j]} \tag{5.26}$$

$$= \frac{\partial\mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial\mathsf{Q}_\mathrm{p}(\boldsymbol{x}_l[j])} \frac{\partial\mathsf{Q}_\mathrm{p}(\boldsymbol{x}_l[j])}{\partial\boldsymbol{x}_l[j]} \tag{5.27}$$

$$\approx \frac{\partial\mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial\mathsf{Q}_\mathrm{p}(\boldsymbol{x}_l[j])} \frac{\partial\mathsf{Q}_\mathrm{d}(\boldsymbol{x}_l[j])}{\partial\boldsymbol{x}_l[j]}, \tag{5.28}$$

where the approximation in (5.28) is obtained by replacing $\mathsf{Q}_\mathrm{p}(\boldsymbol{x}_l[j])$ in the second partial derivative of (5.27) with its differentiable approximation $\mathsf{Q}_\mathrm{d}(\boldsymbol{x}_l[j])$. The first partial derivative in (5.28) can be calculated by backpropagation over the layers of DNN, and the second partial derivative could be computed according to (5.13a) with corresponding

parameters. Now, the partial derivatives w.r.t. $\boldsymbol{w}_l[i]$ can be calculated as

$$\begin{aligned}
&\frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial \boldsymbol{w}_l[i]} \qquad (5.29)\\
&= \sum_j \frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial \boldsymbol{x}_l[j]} \frac{\partial \boldsymbol{x}_l[j]}{\partial \mathsf{Q}_\mathrm{p}(\boldsymbol{w}_l[i])} \frac{\partial \mathsf{Q}_\mathrm{p}(\boldsymbol{w}_l[i])}{\partial \boldsymbol{w}_l[i]}\\
&\approx \frac{\partial \mathsf{Q}_\mathrm{d}(\boldsymbol{w}_l[i])}{\partial \boldsymbol{w}_l[i]} \sum_j \frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial \boldsymbol{x}_l[j]} \frac{\partial \boldsymbol{x}_l[j]}{\partial \mathsf{Q}_\mathrm{p}(\boldsymbol{w}_l[i])}, \qquad (5.30)
\end{aligned}$$

where the summation above is taken over all j corresponding to which $\boldsymbol{x}_l[j]$ depends on $\mathsf{Q}_\mathrm{p}(\boldsymbol{w}_l[i])$.

In (5.30), the first partial derivative is obtained using (5.13a) with corresponding parameters, and the rest partial derivatives can be obtained via backpropagation and (5.28).

In addition, the partial derivative of $\mathcal{L}(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\})$ w.r.t. q_l is calculated as follows

$$\begin{aligned}
&\frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial q_l}\\
&\approx \sum_{i \in [|\boldsymbol{w}_l|]} \frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial \mathsf{Q}_\mathrm{p}(\boldsymbol{w}_l[i])} \frac{\partial \mathsf{Q}_\mathrm{d}(\boldsymbol{w}_l[i])}{\partial q_l}, \qquad (5.31)
\end{aligned}$$

where the second partial derivative in (5.31) could be computed according to (5.13b). Similarly,

$$\begin{aligned}
&\frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial s_l}\\
&\approx \sum_{j \in [|\boldsymbol{x}_l|]} \frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial \mathsf{Q}_\mathrm{p}(\boldsymbol{x}_l[j])} \frac{\partial \mathsf{Q}_\mathrm{d}(\boldsymbol{x}_l[j])}{\partial s_l}. \qquad (5.32)
\end{aligned}$$

Also, for the mapping parameters α_l and β_l, we use the following approximations

$$\begin{aligned}&\frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)}{\partial \alpha_l}\\ &\approx \sum_{i\in[|\boldsymbol{w}_l|]} \frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)}{\partial \mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}_l[i])} \frac{\partial \mathsf{Q}_{\mathrm{d}}(\boldsymbol{w}_l[i])}{\partial \alpha_l},\end{aligned} \tag{5.33}$$

and

$$\begin{aligned}&\frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)}{\partial \beta_l}\\ &\approx \sum_{j\in[|\boldsymbol{x}_l|]} \frac{\partial \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)}{\partial \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x}_l[j])} \frac{\partial \mathsf{Q}_{\mathrm{d}}(\boldsymbol{x}_l[j])}{\partial \beta_l},\end{aligned} \tag{5.34}$$

where $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\boldsymbol{w}_l[i])}{\partial \alpha_l}$ and $\frac{\partial \mathsf{Q}_{\mathrm{d}}(\boldsymbol{x}_l[j])}{\partial \beta_l}$ can be computed analytically.

5.5.3 Entropy Computation of Quantized Weights and Activations

Consider layer l. All weights at layer l are randomly quantized according to the CPMF defined in (5.22). Pick a weight W randomly at layer l. Let $\hat{W}$ denote its corresponding randomly quantized weight according to (5.22). The marginal distribution of W is the empirical distribution of weights at layer l. The MPMF of $\hat{W}$ is then given by

$$P_{\alpha_l}(\hat{w}) = \frac{1}{|\boldsymbol{w}_l|} \sum_{i\in[|\boldsymbol{w}_l|]} P_{\alpha_l}(\hat{w}|\boldsymbol{w}_l[i]),\ \forall \hat{w} \in \hat{\mathcal{A}}_l. \tag{5.35}$$

If entropy coding (such as Huffman coding) is applied to encode all quantized weights at layer l, the average number of bits per weight would be roughly equal to the Shannon entropy of $\hat{W}$

$$H(\hat{W}) = \mathsf{H}\big(\frac{1}{|\boldsymbol{w}_l|} \sum_{i\in[|\boldsymbol{w}_l|]} [P_{\alpha_l}(\cdot|\boldsymbol{w}_l[i])]\big).$$

Accordingly, the total number of bits required to represent all quantized weights at layer l can be represented by

$$\mathsf{H}\big(\boldsymbol{w}_l\big) = |\boldsymbol{w}_l|\ \mathsf{H}\big(\frac{1}{|\boldsymbol{w}_l|}\sum_{i\in[|\boldsymbol{w}_l|]}[P_{\alpha_l}(\cdot|\boldsymbol{w}_l[i])]\big), \tag{5.36}$$

and the total number of bits required to represent all (quantized) weights of the whole DNN is

$$\mathsf{H}\big(\boldsymbol{w}\big) = \sum_{l\in[L]}\mathsf{H}\big(\boldsymbol{w}_l\big), \tag{5.37}$$

which can be referred to as the description complexity of the DNN model.

Similarly, the total number of bits required to represent $\boldsymbol{x}_l$ is

$$\mathsf{H}\big(\boldsymbol{x}_l\big) = |\boldsymbol{x}_l|\ \mathsf{H}\big(\frac{1}{|\boldsymbol{x}_l|}\sum_{i\in[|\boldsymbol{x}_l|]}[P_{\beta_l}(\cdot|\boldsymbol{x}_l[i])]\big), \tag{5.38}$$

where $P_{\beta_l}(\cdot|\boldsymbol{x}_l[i]))$ is calculated according to (5.23). And the total number of bits required to represent all quantized actvations of the whole DNN is

$$\mathsf{H}\big(\boldsymbol{x}\big) = \sum_{l\in[L-1]}\mathsf{H}\big(\boldsymbol{x}_l\big). \tag{5.39}$$

5.5.4 New Objective Function with Entropy Constraints

To make both weights and activations compressible at any stage of training, we now impose constraints on $\mathsf{H}\big(\boldsymbol{w}\big)$ and $\mathsf{H}\big(\boldsymbol{x}\big)$ during training. This can be effectively achieved by incorporating $\mathsf{H}\big(\boldsymbol{w}\big)$ and $\mathsf{H}\big(\boldsymbol{x}\big)$ into the objective function so that $\mathsf{H}\big(\boldsymbol{w}\big)$ and $\mathsf{H}\big(\boldsymbol{x}\big)$ can be minimized jointly with the original loss $\mathcal{L}(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\})$. Thus the new objective function in CDL is

$$\begin{aligned}
&\hat{\mathcal{L}}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}, \boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}\Big) \\
&= \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big) + \gamma\mathsf{H}(\boldsymbol{x}) + \lambda\mathsf{H}(\boldsymbol{w}) &(5.40)\\
&= \mathcal{L}_a\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big) + \lambda\mathsf{H}(\boldsymbol{w}), &(5.41)
\end{aligned}$$

where $\lambda \geq 0$ and $\gamma \geq 0$ are hyperparameters representing the trade-offs among the three terms in (5.40), and

$$\begin{aligned}&\mathcal{L}_a\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)\\&\quad= \mathcal{L}\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big) + \gamma \mathsf{H}(\boldsymbol{x}).\end{aligned} \tag{5.42}$$

In CDL, the learning process is then to solve the following minimization problem

$$\begin{aligned}&\min_{(\boldsymbol{q},\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w})} \mathbb{E}\Big\{\hat{\mathcal{L}}\Big(X, Y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}, \boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}\Big)\Big\}\\&= \min_{(\boldsymbol{q},\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w})} \Big\{\mathbb{E}\Big\{\mathcal{L}_a\big(X, Y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\big)\Big\} + \lambda \mathsf{H}(\boldsymbol{w})\Big\} \qquad (5.43)\\&= \min_{(\boldsymbol{q},\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w})} \Big\{\lambda \mathsf{H}(\boldsymbol{w}) + \mathbb{E}_{X,Y,\mathsf{Q}_{\mathrm{p}}}\big\{\mathcal{L}_a\big(X, Y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\big)\big\}\Big\}. \qquad (5.44)\end{aligned}$$

In the above, (5.43) is due to the fact that $\mathsf{H}(\boldsymbol{w})$ is deterministic given $\boldsymbol{w}$; the expectation in (5.44) is with respect to the random sample (X, Y) and random quantizers Q_{p}.

In practice, when the joint distribution of the random sample (X, Y) is unknown, the expectation in (5.44) can be approximated by the corresponding sample mean over a mini-batch $\mathcal{B}$ during learning, where for each sample instance $(\boldsymbol{x}_0, y) \in \mathcal{B}$, only one instance of each probabilistic quantizer Q_{p} is taken. In this case, the objective function in (5.44) becomes

$$\begin{aligned}&\mathcal{J}_{\mathcal{B}}\Big(\{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}, \boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}\Big) =\\&\lambda \mathsf{H}(\boldsymbol{w}) + \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x}_0, y) \in \mathcal{B}} \mathcal{L}_a\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big),\end{aligned} \tag{5.45}$$

where $|\mathcal{B}|$ denotes the size of $\mathcal{B}$. The learning process is to iteratively solve the following minimization problem through a sequence of mini-batches

$$\min_{(\boldsymbol{q},\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w})} \Big\{\lambda \mathsf{H}(\boldsymbol{w}) + \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x}_0, y) \in \mathcal{B}} \mathcal{L}_a\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)\Big\}. \tag{5.46}$$

Let $\boldsymbol{\Omega} = (\boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w})$. Before we conclude this subsection, let us compute the gradient of $\mathcal{J}_{\mathcal{B}}(\{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}, \boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ with respect to $\boldsymbol{\Omega}$. First note that the partial derivatives of $\mathsf{H}(\boldsymbol{w})$ with respect to $(\boldsymbol{w}, \boldsymbol{q}, \boldsymbol{\alpha})$ can be computed analytically.

Algorithm 2 CDL

Input: All mini-batches $\{\mathcal{B}_b\}_{b\in[B]}$, maximum number of epochs $E_{\max}$, λ, γ, the quantization index sets $\mathcal{A}$ and $\mathcal{B}$, and learning rates $\{\eta_w, \eta_q, \eta_s, \eta_\alpha, \eta_\beta\}$.

Initialization: Initialize $\boldsymbol{\Omega}^0 = (\boldsymbol{q}^0, \boldsymbol{s}^0, \boldsymbol{\alpha}^0, \boldsymbol{\beta}^0, \boldsymbol{w}^0)$.
for $e = 1$ to $E_{\max}$ **do**
  for $b = 1$ to B **do**
    Randomly quantize the weights $(\boldsymbol{w})_{b-1}^e$ according to $\mathsf{Q}_{\mathrm{p}}(\cdot)$ with parameters $(\boldsymbol{q})_{b-1}^e$ and $(\boldsymbol{\alpha})_{b-1}^e$.
    For each sample $(\boldsymbol{x}_0, y) \in \mathcal{B}_b$, perform forward pass based on quantized weights $\mathsf{Q}_{\mathrm{p}}((\boldsymbol{w})_{b-1}^e)$ while successively quantizing the activations according to $\mathsf{Q}_{\mathrm{p}}(\cdot)$ with parameters $(\boldsymbol{s})_{b-1}^e$ and $(\boldsymbol{\beta})_{b-1}^e$.
    For each sample $(\boldsymbol{x}_0, y) \in \mathcal{B}_b$, perform its corresponding backward pass over quantized weights and activations to find the following partial derivatives

$$\frac{\partial \mathcal{L}_a\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}\Big)}{\partial \boldsymbol{\Omega}} \tag{5.47}$$

    at $\boldsymbol{\Omega} = (\boldsymbol{\Omega})_{b-1}^e$.
    Compute $\frac{\partial \mathsf{H}(\boldsymbol{w})}{\partial \boldsymbol{\Omega}}$ at $\boldsymbol{\Omega} = (\boldsymbol{\Omega})_{b-1}^e$.
    Use (5.49) to compute

$$\frac{\partial \mathcal{J}_{\mathcal{B}_b}\Big(\{\mathsf{Q}_{\mathrm{p}}(\boldsymbol{w}), \mathsf{Q}_{\mathrm{p}}(\boldsymbol{x})\}, \boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}\Big)}{\partial \boldsymbol{\Omega}}$$

    at $\boldsymbol{\Omega} = (\boldsymbol{\Omega})_{b-1}^e$.
    Update $(\boldsymbol{\Omega})_{b-1}^e$ to $(\boldsymbol{\Omega})_b^e$ by GD with the learning rate η_w and the layer-wise learning rates given in (5.50).
  end for
end for
return $\boldsymbol{\Omega}^{E_{\max}}$.

Second, the gradient of $\mathcal{L}_a(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\})$ with respect to $\boldsymbol{\Omega}$ can be computed via Eqs. (5.26) to (5.34) and backpropagation with $\mathcal{L}$ replaced by $\mathcal{L}_a$ in all equations therein. Finally, we have

$$\frac{\partial \mathcal{J}_\mathcal{B}\Big(\{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}, \boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}\Big)}{\partial \boldsymbol{\Omega}} \tag{5.48}$$

$$= \lambda \frac{\partial \mathsf{H}(\boldsymbol{w})}{\partial \boldsymbol{\Omega}} + \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x}_0, y) \in \mathcal{B}} \frac{\partial \mathcal{L}_a\Big(\boldsymbol{x}_0, y, \{\mathsf{Q}_\mathrm{p}(\boldsymbol{w}), \mathsf{Q}_\mathrm{p}(\boldsymbol{x})\}\Big)}{\partial \boldsymbol{\Omega}}. \tag{5.49}$$

5.5.5 Algorithm for Solving the Optimization in (5.46)

In order to solve the optimization problem in (5.46) over mini-batches $\{\mathcal{B}_b\}_{b \in [B]}$, we deploy the conventional gradient descent (GD) strategy to update the five sets of parameters $\boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{w}$ in (5.46). The partial derivatives of $\mathcal{J}_\mathcal{B}(\cdot)$ w.r.t. these five sets of parameters can be computed via (5.49).

For learning rates, we make an adjustment for each set of the trainable parameters appropriately. For updating $\boldsymbol{w}$, we use a similar η_w to that of conventional DL. On the other hand, for the other four sets of parameters we should take a careful consideration. Specifically, it has been demonstrated that effective convergence occurs during training when the ratio of the average update magnitude to the average parameter magnitude remains consistent across all weight layers within a network [230]. In addition, we anticipate the step-size parameter to decrease with increasing precision, as finer quantization has smaller quantization step sizes. Likewise, since quantization parameters at layer l impact all weights/activations at layer l, their impact on the loss function would be the aggregated impact from all weights/activations they are used to quantize. As such, their learning rates should be inversely affected by the number of weights/activations they are used to quantize. This, together with [57], motivates us to scale the learning rates for layer l,

$l \in [L]$, as follows

$$\eta_{q_l} = \frac{1}{\sqrt{|\boldsymbol{w}_l| 2^{b-1}}} \eta_q, \tag{5.50a}$$

$$\eta_{s_l} = \frac{1}{\sqrt{|\boldsymbol{x}_l| 2^{b}}} \eta_s, \tag{5.50b}$$

$$\eta_{\alpha_l} = \frac{1}{\sqrt{|\boldsymbol{w}_l|}} \eta_\alpha, \tag{5.50c}$$

$$\eta_{\beta_l} = \frac{1}{\sqrt{|\boldsymbol{x}_l|}} \eta_\beta, \tag{5.50d}$$

for some $\eta_q, \eta_s, \eta_\alpha, \eta_\beta > 0$.

The proposed algorithm for optimization problem (5.46) is summarized in Algorithm 2. To simplify our notation, we use $(\cdot)_b^e$ to indicate parameters of the b-th batch updation during the e-th epoch of the algorithm. We further write $(\cdot)_B^e$ as $(\cdot)^e$ whenever needed, and set $(\cdot)_0^e = (\cdot)^{e-1}$.

5.5.6 R-CDL

R-CDL is almost identical to CDL except that (i) the probabilistic quantizers $\mathsf{Q}_\mathrm{p}(\cdot)$ are replaced with their respective *soft deterministic* approximations $\mathsf{Q}_\mathrm{d}(\cdot)$; and (ii) forward and backward passes are operated on softly quantized weights and activations with full precision. In R-CDL, the learning process is to solve the following optimization problem

$$\min_{(\boldsymbol{q},\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{w})} \mathbb{E}_{(X,Y)} \Big\{ \mathcal{L}(X, Y, \{\mathsf{Q}_\mathrm{d}(\boldsymbol{w}), \mathsf{Q}_\mathrm{d}(\boldsymbol{x})\}) + \lambda \mathsf{H}(\boldsymbol{w}) + \gamma \mathsf{H}(\boldsymbol{x}) \Big\}. \tag{5.51}$$

Compared to CDL, R-CDL offers the advantage of computing all gradients analytically without approximation during training, albeit at the cost of increased computation for forward and backward passes using floating-point operations. This advantage enables R-CDL to achieve superior accuracy-compression trade-off compared to CDL.

Remark. *In R-CDL, weights and activations are still compressible at any stage of training. They can be quantized using the newly obtained CPMFs whenever the inter-device weight and activation communication need arises in model/data parallelism; weights are still quantized at the end of training.*

5.6 Experiments

To demonstrate the effectiveness of CDL and R-CDL and compare them with existing SOTA methods in the literature, we have conducted a comprehensive series of experiments. This section presents these experiment results. We begin with defining performance metrics to be used to compare CDL and R-CDL with the benchmark methods.

5.6.1 Performance Metrics

In all cases, we report the test accuracy of each trained DNN vs the average number of bits required to represent the weights/activations of the DNN, where the latter is defined as

$$\text{average \# of bits per weight} = \frac{\sum_{l=1}^{L} |\boldsymbol{w}_l| b_{\boldsymbol{w}_l}}{\sum_{l=1}^{L} |\boldsymbol{w}_l|}, \tag{5.52a}$$

$$\text{average \# of bits per activation} = \frac{\sum_{l=1}^{L-1} |\boldsymbol{x}_l| b_{\boldsymbol{x}_l}}{\sum_{l=1}^{L-1} |\boldsymbol{x}_l|}, \tag{5.52b}$$

where $b_{\boldsymbol{w}_l}$ and $b_{\boldsymbol{x}_l}$ denote the average number of bits per weight and activation, respectively, required to represent the l-th layer of the DNN using Huffman coding. Particularly, to calculate $b_{\boldsymbol{x}_l}$, we first perform an inference over the samples in a mini-batch of the training dataset to obtain the respective $\boldsymbol{x}_l$ for these samples, and then encode these activations using Huffman coding.

Furthermore, for a fair comparison with the benchmark methods, we always apply Huffman coding to encode the weights and activations of the fully-trained models obtained using the benchmark methods.

5.6.2 Implementation Details of CDL and R-CDL

In this subsection, we delve into the implementation details of CDL and R-CDL which is used throughout this section.

Quantization operations in a mini-batch

As the softmax operation is time-consuming, deploying quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ and $\mathsf{Q}_{\mathrm{d}}(\cdot)$ incurs some computation overheads. This will not be an issue when deploying these quantizers

for the weights as the weights are quantized only once per mini-batch (refer to Algorithm 2). On the other hand, when quantizing the activations, these quantizers should be applied over all the samples in a mini-batch, which incurs time complexity. To address this issue, for quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ and $\mathsf{Q}_{\mathrm{d}}(\cdot)$ used for activations, each of their corresponding CPMFs $\left[P_{\beta_l}(\cdot|\boldsymbol{x}_l[j])\right]$ is modified to be concentrated proportionally on the top five values with the rest of conditional probabilities to be 0.

First and last layer quantization

We quantize all weights to a low bit-width, with the exception of the weights in the first and last layers, which are kept at 8-bits, similar to the approach used in benchmark methods [13, 58, 235].

Common hyper-parameters of Algorithm 1

For all performance curves reported in this chapter, we always set $b = 6$ for both weights and activations in Algorithm 2, and test different λ and γ values. For all the five sets of parameters $\{\boldsymbol{q}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}\}$, we use a similar optimizer, and the same values for the learning rates $\{\eta_w, \eta_q, \eta_s, \eta_\alpha, \eta_\beta\}$. Note that the learning rates for quantization parameters are further adaptively scaled layer-wise during training according to (5.50).

Inspired from [58], we set $\boldsymbol{q}^0 = \frac{2|\overline{\boldsymbol{w}}|}{\sqrt{2^{b-1}}}$ and $\boldsymbol{s}^0 = \frac{2|\overline{\boldsymbol{x}}|}{\sqrt{2^{b-1}}}$, where $\overline{|\boldsymbol{w}|}$ is the average of absolute values of initial weights, and $\overline{|\boldsymbol{x}|}$ is the average of absolute values for the first batch of activations. Also, we set $\boldsymbol{\alpha}^0 = [500]_L$ and $\boldsymbol{\beta}^0 = [500]_{L-1}$.

5.6.3 Experiments on ImageNet

ImageNet is a large-scale dataset containing around 1.2 million training samples and 50,000 validation images.

• **Models**: We have conducted experiments on two models from ResNet family, namely ResNet-18 and ResNet-34 [84].

• **Benchmarks**: We evaluate the performance of CDL (and R-CDL) against several QAT methods, including LSQ [58], PACT [29], LQ-Nets [235]. In this subsection, all the benchmark methods, including CDL, are trained from **scratch**. Additionally, in section 5.7.1, we explore the effect of initializing training from pre-trained models on validation

Table 5.1: (Top-1 accuracy, Average # of bits per weight, Average # of bits per activation) given by models trained by CDL and R-CDL with $(\lambda, \gamma) = (0, 0)$ and $b = \{2, 3, 4\}$.

Model	Method	$b = 4$	$b = 3$	$b = 2$
ResNet-18	CDL	(70.45%, 3.08, 3.18)	(69.72%, 2.25, 2.23)	(67.13%, 1.50, 1.66)
	R-CDL	(70.50%, 2.88, 2.76)	(69.95%, 2.10, 2.12)	(67.88%, 1.40, 1.51)
ResNet-34	CDL	(73.80%, 3.33, 3.31)	(73.01%, 2.25, 2.34)	(70.88%, 1.55, 1.52)
	R-CDL	(73.92%, 3.04, 3.12)	(73.25%, 2.20, 2.29)	(71.44%, 1.45, 1.46)

accuracy, where we also add two additional benchmarks, namely APoT [132] and DMBQ [242]. These two benchmarks are not included in the current subsection due to their poorer performance when trained from scratch.

• **Training settings**: We have deployed an SGD optimizer with a momentum of 0.9, a weight decay of 0.0001, and a batch size of 256. We have trained the models for 90 epochs, and adopted an initial learning rate of 0.1, which is further divided by 10 at the 30-th and 60-th epochs.

We have run Algorithm 2 for different pairs of λ and γ values
$(\lambda, \gamma) = \{(0, 0), (0.01, 0.01), \ldots, (0.09, 0.09)\}$.

• **Results**: Figs. 5.4 and 5.5 show the performance of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) for models trained by CDL, R-CDL, and benchmark methods, for ResNet-18 and ResNet-34, respectively.

To demonstrate the performance gain delivered only by the proposed quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ and $\mathsf{Q}_{\mathrm{d}}(\cdot)$, we have also run Algorithm 2 with $(\lambda, \gamma) = (0, 0)$, and $b = \{2, 3, 4\}$. The corresponding results for both ResNet-18 and ResNet-34 on ImageNet are listed in Table 5.1.

By examining the curves in Figs. 5.4 and 5.5, and also the numbers in Table 5.1, it is fair to make the following observations:

- *Observation 1*: The accuracy of the models trained by R-CDL is always above those trained by CDL. This is because unlike CDL, the gradient calculations in R-CDL are performed with no approximation.

- *Observation 2*: The models trained by CDL and R-CDL can even have slightly higher test accuracy compared to the respective pre-trained FP models while achieving significant compression. For example, ResNet-18 trained by R-CDL has 70.55% accuracy (slightly better than 70.5% FP accuracy) with 2.34 bits per weight, achieving

more than 13 fold compression; likewise, ResNet-18 trained by CDL has 70.52% accuracy (slightly better than 70.5% FP accuracy) with 3.34 bits per weight, achieving more than 9 fold compression. Similar results are valid for ResNet-34 as well.

- *Observation 3*: In the case of low bit rates, the accuracy of the models trained via CDL and R-CDL is significantly better than that of those trained by benchmark methods.

- *Observation 4*: The proposed quantizers $\mathsf{Q}_{\mathrm{p}}(\cdot)$ and $\mathsf{Q}_{\mathrm{d}}(\cdot)$ on their own also outperforms benchmark quantization methods in terms of model test accuracy.

5.6.4 Experiments on CIFAR-100

CIFAR-100 dataset contains 50K training and 10K test images of size 32×32, which are labeled for 100 classes.

• **Models**: To demonstrate the effectiveness of CDL and R-CDL, we have conducted experiments using three different model architectural families. Specifically, we have selected (i) four models from the ResNet family [84], namely ResNet-$\{20, 44, 56, 110\}$; (ii) VGG-13 from the VGG family [185]; and (iii) Wide-ResNet-28-10 from the Wide-ResNet family [231].

• **Benchmarks**: We have trained the selected DNN models from scratch via CDL and R-CDL. They are evaluated against those trained via LSQ either on top of pre-trained models (denoted as P-LSQ) or from scratch (denoted as S-LSQ).

• **Training settings**: We have deployed an SGD optimizer with a momentum of 0.9, a weight decay of 0.0005, and a batch size of 64. We have trained the models for 200 epochs, and adopted an initial learning rate of 0.1, which is further divided by 10 at the 60-th, 120-th and 160-th epochs. To have a fair comparison, we have reproduced the results of LSQ on our local machine.

We have run Algorithm 2 for different pairs of λ and γ values
$(\lambda, \gamma) = \{(0, 0), (0.01, 0.01), \ldots, (0.09, 0.09)\}$.

• **Results**: The results for ResNet-20, ResNet-44, ResNet-56, ResNet-110, VGG-13, and Wide-ResNet-28-10 are illustrated in Figs. 5.6a, 5.6b, 5.7a, 5.7b, 5.8a, and 5.8b, respectively. Again, the observations made in Subsection 5.6.3 for ImageNet remain valid as well for CIFAR-100. Additionally, we note that, generally, the accuracy gap between CDL and R-CDL diminishes as the model size increases.

5.6.5 Evolution of Weight Distribution

Since the entropy term $\mathsf{H}(\boldsymbol{w})$ (and $\mathsf{H}(\boldsymbol{x})$) in the optimization problem (5.46) imposes certain structures on the weights (and activations), it is interesting to see how CDL changes the distribution of the weights in favour of higher compression ratio. Fig. 5.9 illustrates the evolution of the distribution of flattened weights in a convolutional layer of ResNet-110 during the training process within CDL on CIFAR-100. Specifically, $b = 6$, and the weight distribution is depicted for epochs 0, 50, 100, 150 and 200 using 2^6 bins. As observed, over time, the distribution of weights becomes more concentrated around the center, leading to reduced entropy. This reduction in entropy in turn leads to a higher compression ratio when employing Huffman coding.

5.6.6 Average Number of Bits vs Epoch for Facilitating Data/Model Parallelism

To show that weights and activations are indeed compressible at any stage of training within CDL and R-CDL, we depict the average number of bits required to encode the weights and activations of ResNet-110 trained via CDL or R-CDL over the course of training on CIFAR-100 in Fig. 5.11. We calculate the average number of bits as follows. In the case of CDL, since the weights and activations are quantized at each iteration step during training, we proceed to encode them after the completion of each epoch using Huffman coding. For R-CDL, after each epoch is finished, we use $\mathsf{Q}_{\mathrm{p}}(\cdot)$ with the newly obtained quantization parameters to quantize the weights and activations, and then encode the quantized values using Huffman coding.

The results for CDL weights and activations are shown in Figs. 5.10a and 5.10b, respectively, while those for R-CDL weights and activations are displayed in Figs. 5.11a and 5.11b, respectively. As seen, for $\lambda, \gamma > 0$, the average number of bits drops quickly after a few epochs. This in turn would reduce the inter-device weight and activation communication costs quickly in model/data parallelism.

5.6.7 Convergence Curves for CDL and R-CDL

To illustrate the convergence of Algorithm 2 in both cases of CDL and R-CDL, we plot the evolution of the objective function of CDL given in (5.46) and of R-CDL given in (5.51) over the course of training ResNet-110 on CIFAR-100 in Fig. 5.12. Particularly, the curves for CDL and R-CDL are depicted in Figs. 5.12a and 5.12b, respectively. As observed, both

curves flatten out near the end of the training, indicating that the algorithm has converged in each case.

5.7 CDL with Non-uniform quantization

In this section, we demonstrate a simple way to use non-uniform quantization to further enhance the performance of R-CDL.

To this end, we use the learnable *companding* quantization (LCQ) method as introduced in [218] to learn a non-uniform quantizer during training. Specifically, this non-uniform quantization is composed of three functions: (i) a compressing function $f_{\text{NL}}(\cdot)$, (ii) a uniform quantizer $\mathsf{Q}_{\text{u}}(\cdot)$, (iii) and an expanding function $f_{\text{NL}}^{-1}(\cdot)$. The quantizer obtained in this manner is referred to as a *companding* quantizer, and it is derived as follows:

$$\text{comp}(\theta) = (f_{\text{NL}}^{-1} \circ \mathsf{Q}_{\text{u}} \circ f_{\text{NL}})(\theta), \tag{5.53}$$

where $f_{\text{NL}}(\cdot)$ is a learnable piece-wise linear function (see equation (5) in [218]) with its parameters being optimized during training.

To integrate LCQ method into R-CDL, we replace $\mathsf{Q}_{\text{d}}(\cdot)$ quantizers in R-CDL with $(f_{\text{NL}}^{-1} \circ \mathsf{Q}_{\text{d}} \circ f_{\text{NL}})(\theta)$ quantizers, resulting in a companding version of R-CDL referred to as R-CDL+LCQ. To show the effectiveness of R-CDL+LCQ, we have conducted experiments on ImageNet using ResNet-$\{18, 34\}$ with the similar setup as those used in Section 5.13. Additionally, we use the same hyper-parameters for $f_{\text{NL}}(\cdot)$ as those employed in [218] for the ImageNet experiments. The average numbers of bits for weights and activations are still computed in the same manner as before in R-CDL, that is, after being processed by $f_{\text{NL}}(\cdot)$, they are randomly quantized using the trained CPMFs and then Huffman encoded.

The results for R-CDL+LCQ and LCQ are presented in Figure 5.13, where all models were trained from **scratch**. Based on this figure, the following observations can be made:

• By comparing Figs. 5.13 and 5.4 and 5.5, it is evident that the performance of R-CDL is enhanced with the implementation of non-uniform quantization.

• The performance of R-CDL+LCQ is superior to LCQ.

5.7.1 Training on top of pre-trained models

In this section, we compare CDL and R-CDL with APoT [132], LSQ [58] and DMBQ [242] when they all are applied on top of pre-trained models for ImageNet. All models in this

section are trained starting from pre-trained FP models. The training settings otherwise for CDL, R-CDL, and benchmark methods are the same as those used in Subsection 5.6.3.

Fig. 5.14 presents the performance comparison of Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) for models trained using CDL, R-CDL, and benchmark methods. Specifically, these curves are depicted for ResNet-18 and ResNet-34 in Figs. 5.14a and 5.14b, respectively.

Comparing Fig. 5.14 with Fig. 5.4 and 5.5 in Subsection 5.6.3, it is fair to make the following observations:

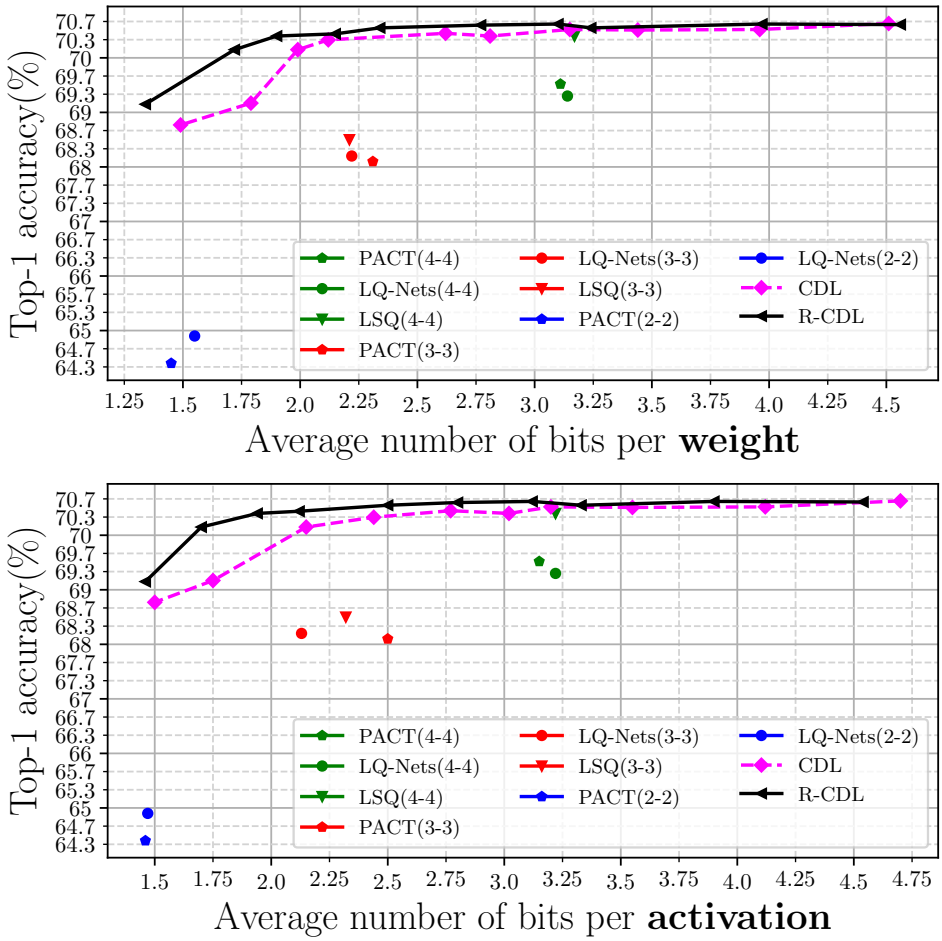
• CDL and R-CDL perform better when applied on top of pre-trained FP models compared to starting from scratch.

• CDL and R-CDL continue to outperform the benchmark methods when they all are applied on top of pre-trained FP models. Again in low bits, the gains offered by CDL and R-CDL, particularly by R-CDL are quite significant.

• When applied on top of pre-trained FP models, CDL and R-CDL can improve the accuracy performance of the pre-trained FP models by a non-negligible margin.

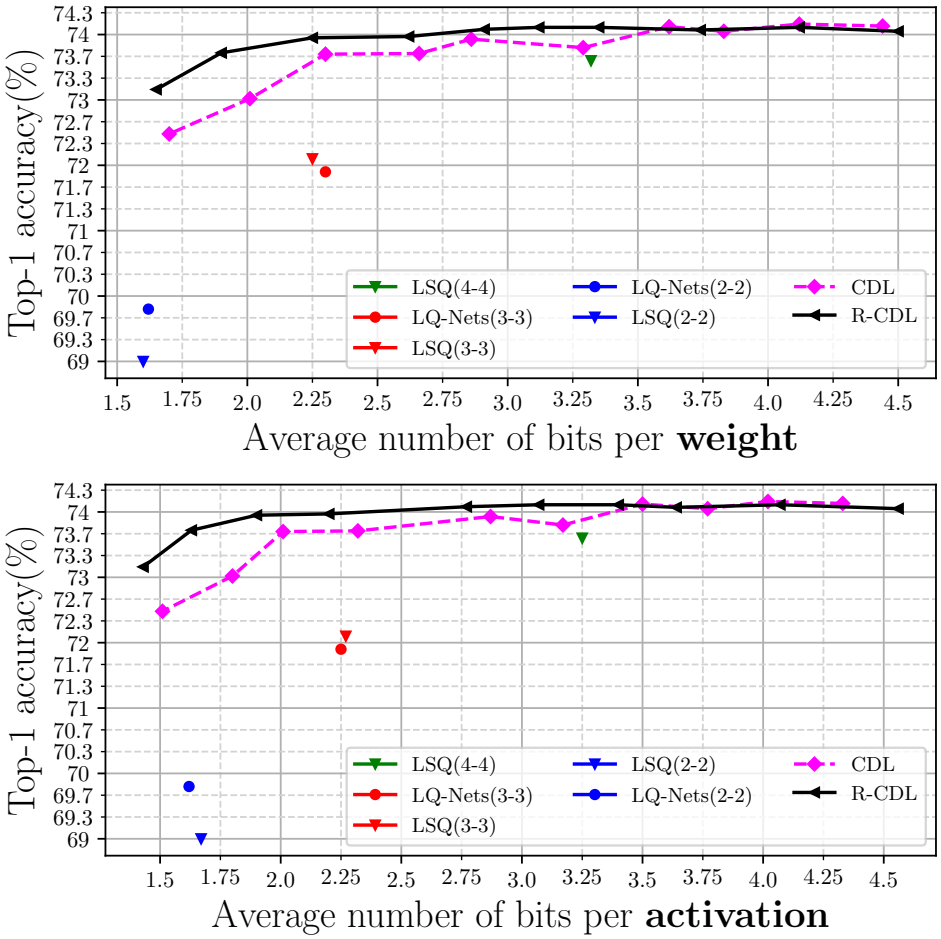
5.8 Chapter Summary

In this chapter, we have introduced a novel probabilistic quantization method and its soft deterministic variant. Empowered by these quantization methods, we have then developed "coded deep learning"' (CDL), a new framework that harnesses information-theoretic coding techniques in the training and inference of deep neural network (DNN) model to achieve substantial compression of model weights and activations, while maintaining or even improving model accuracy, reducing computational complexity at both training and inference stages, and enabling efficient implementation of model/data parallelism. Key elements of CDL include: (1) incorporating trainable probabilistic quantizers to quantize both model weights and activations; (2) enforcing entropy constraints on quantized weights and activations during training, and (3) using the soft deterministic counterparts of the trainable probabilistic quantizers as a proxy to facilitate gradient computation in backpropagation. A variant of CDL called "Relaxed CDL" (R-CDL) has also been proposed to further improve the trade-off between validation accuracy and compression at the disadvantage of full precision operation in both forward and backward passes during training. Extensive experimental results have demonstrated that both CDL and R-CDL outperform state-of-the-art methods in the field of DNN compression, with significant gains in the low bit rate case.



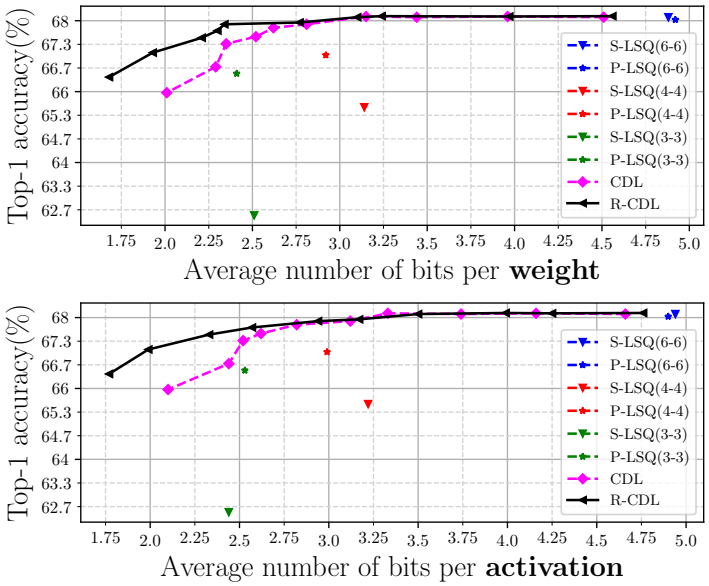
(a) ResNet-18 (FP accuracy:70.5).

Figure 5.4: Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) on ImageNet for ResNet-18. All models are trained from **scratch**.

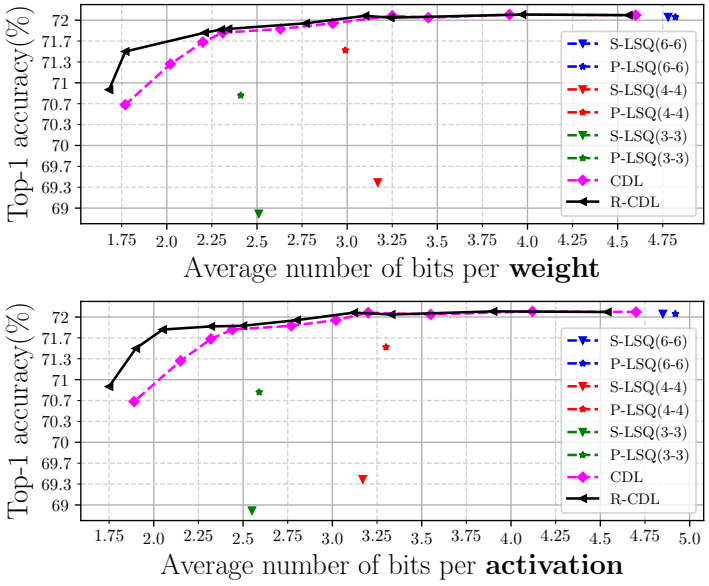


(a) ResNet-34 (FP accuracy:74.1).

Figure 5.5: Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) on ImageNet for ResNet-34. All models are trained from **scratch**.

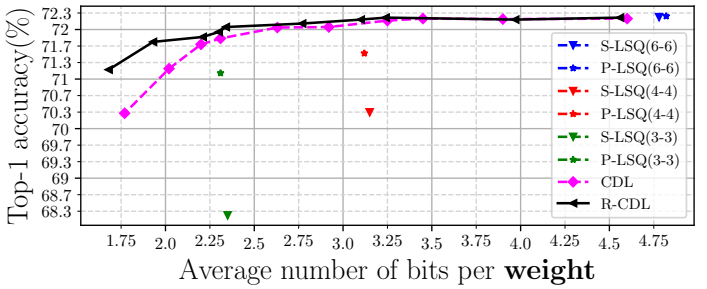


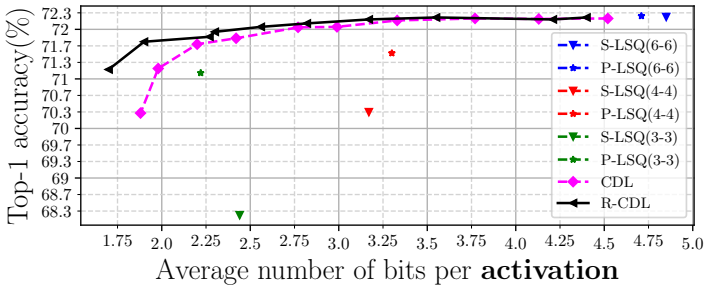
(a) ResNet-20 (FP accuracy: 67.92%).



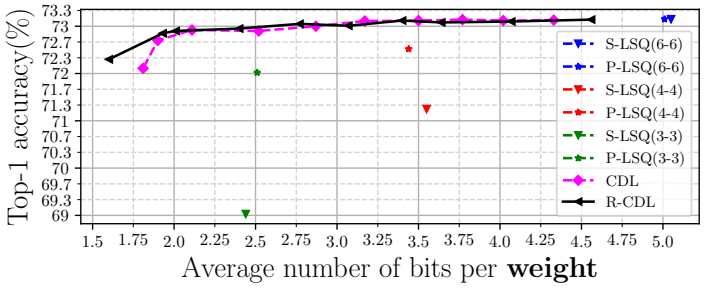
(b) ResNet-44 (FP accuracy: 71.85%).

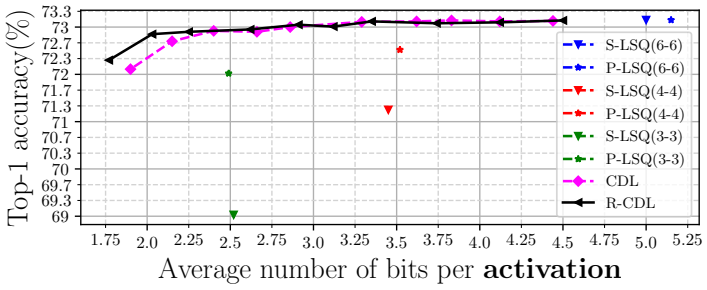
Figure 5.6: Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the test accuracy vs the average number of bits per weight (top)/activation (bottom) on CIFAR-100 dataset: (a) ResNet-20 and (b) ResNet-44.





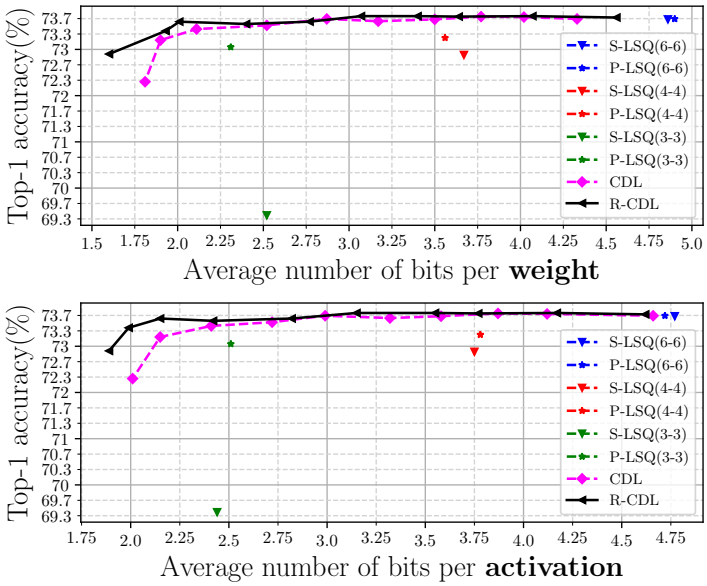
(a) ResNet-56 (FP accuracy: 72.05%).



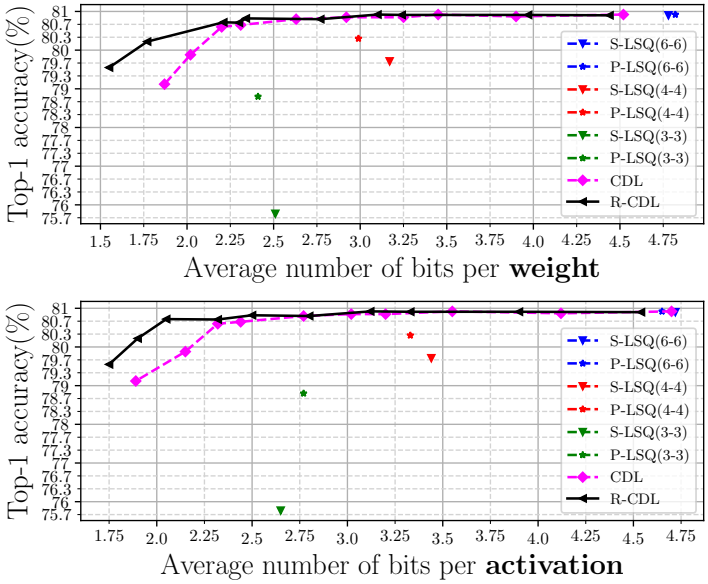


(b) ResNet-110 (FP accuracy: 73.04%).

Figure 5.7: Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the test accuracy vs the average number of bits per weight (top)/activation (bottom) on CIFAR-100 dataset: (a) ResNet-56 and (b) ResNet-110.



(a) [VGG13 (FP: 73.70%).



(b) WRN-28-10 (FP: 80.88%).

Figure 5.8: Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the test accuracy vs the average number of bits per weight (top)/activation (bottom) on CIFAR-100 dataset: (a) VGG-13 and (b) Wide-ResNet-28-10.

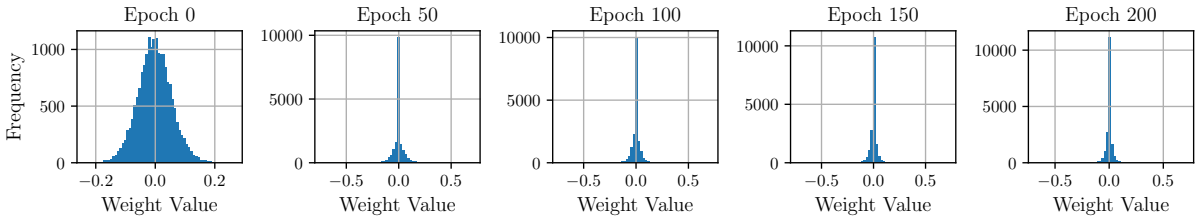
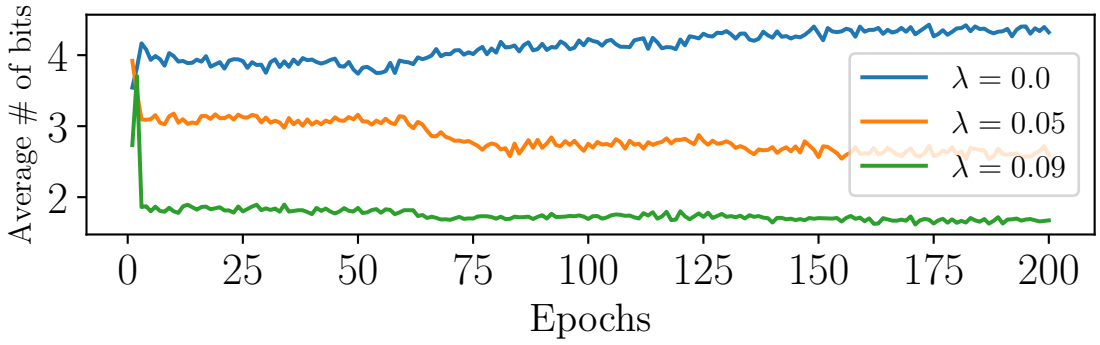
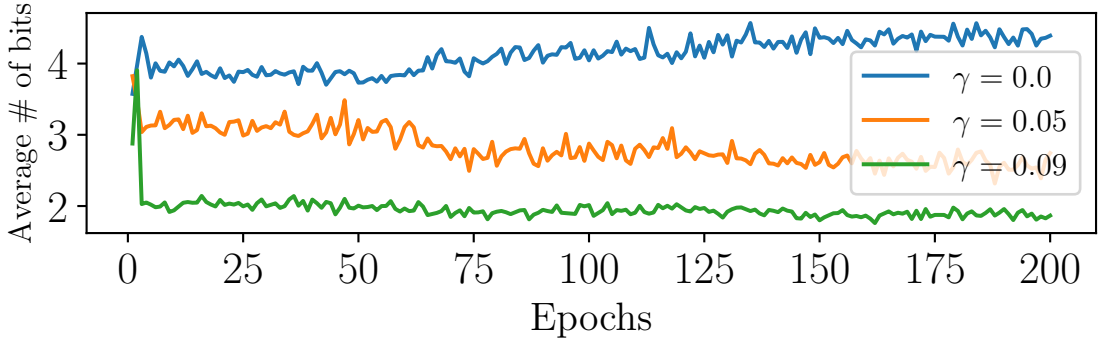


Figure 5.9: The evolution of weight distribution of a convolutional layer in ResNet-110 during training via CDL on CIFAR-100.

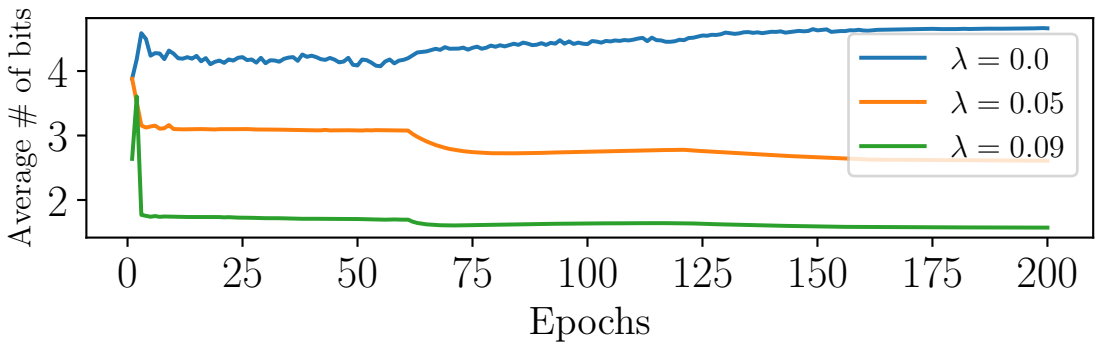


(a) CDL Weights.

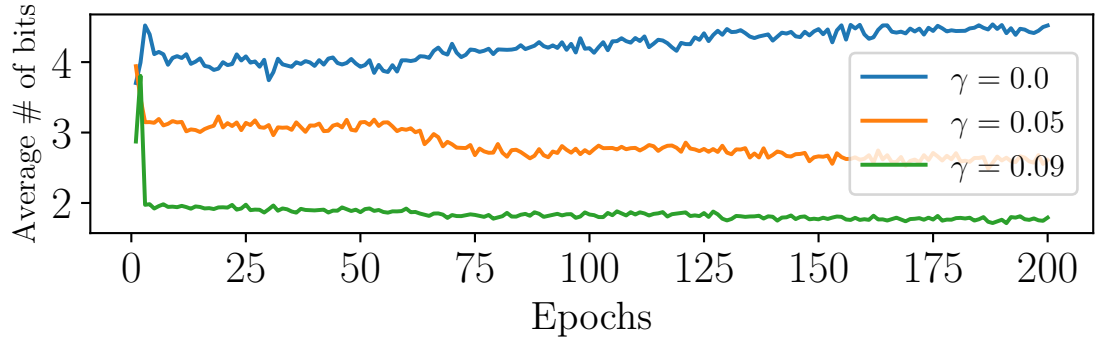


(b) CDL Activations.

Figure 5.10: The average number of bits required to encode (a) CDL weights and (b) CDL activations vs epochs for ResNet-110 trained over CIFAR-100.

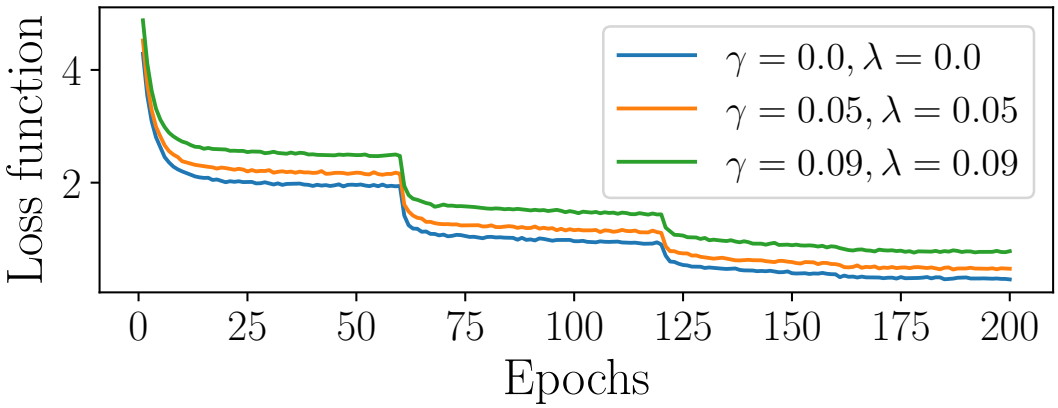


(a) R-CDL Weights.

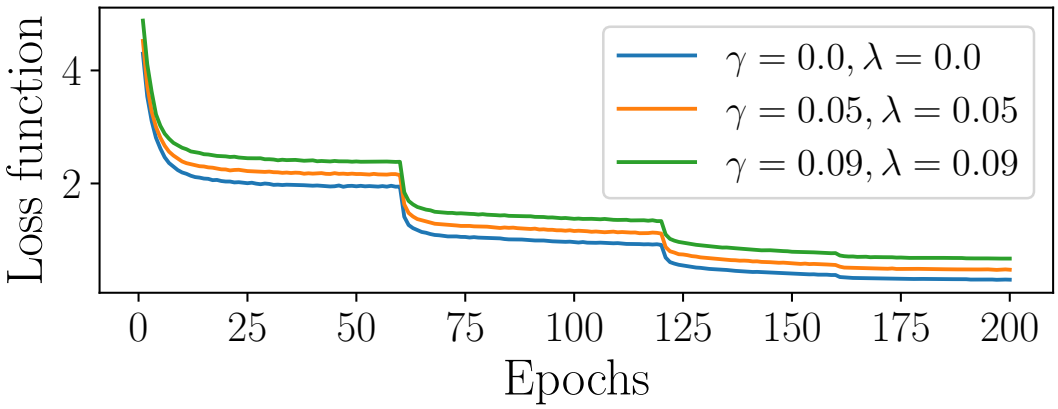


(b) R-CDL Activations.

Figure 5.11: The average number of bits required to encode (a) R-CDL weights and (b) R-CDL activations vs epochs for ResNet-110 trained over CIFAR-100.

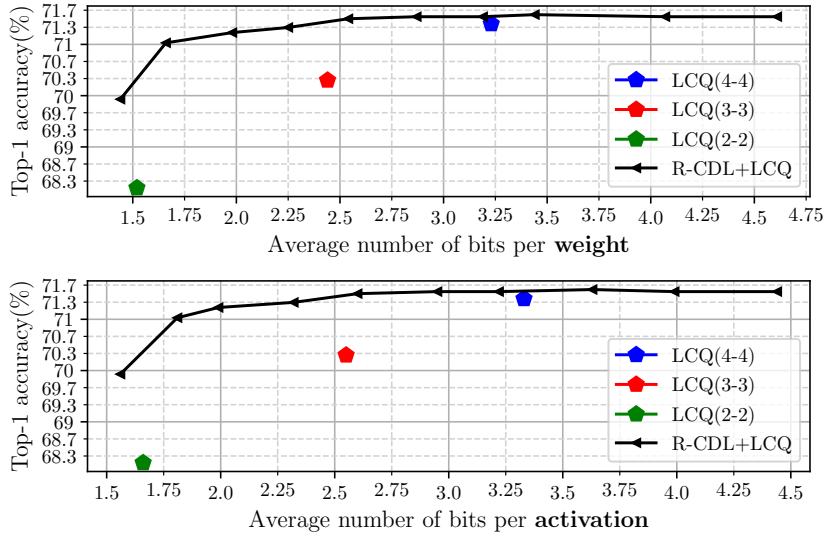


(a) CDL.

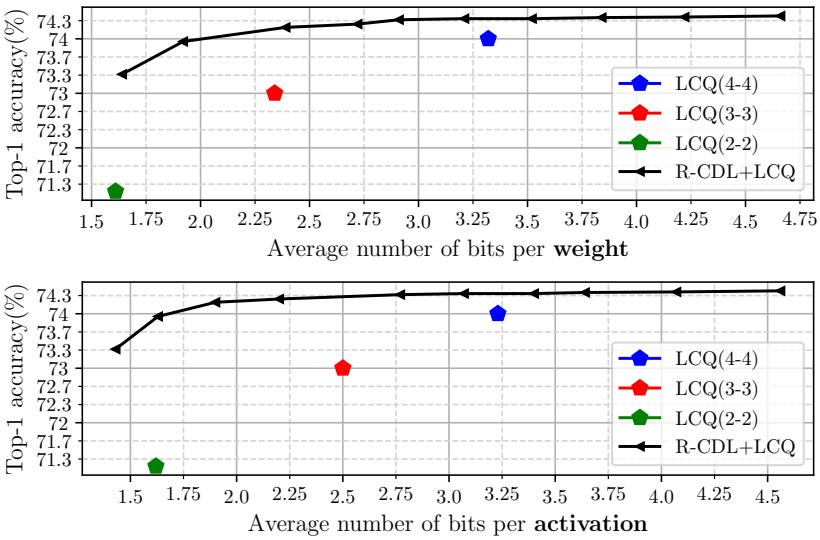


(b) R-CDL.

Figure 5.12: The convergence of objective function in (a) CDL, and (b) R-CDL over the course of training ResNet-110 on CIFAR-100.

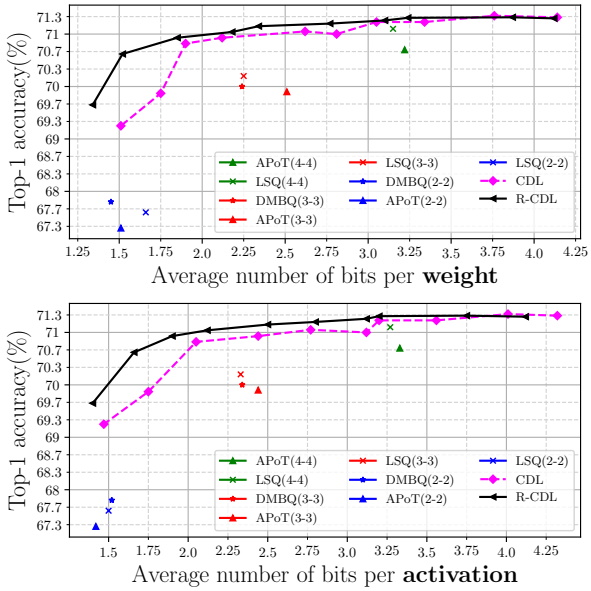


(a) ResNet-18 (FP accuracy:70.5).

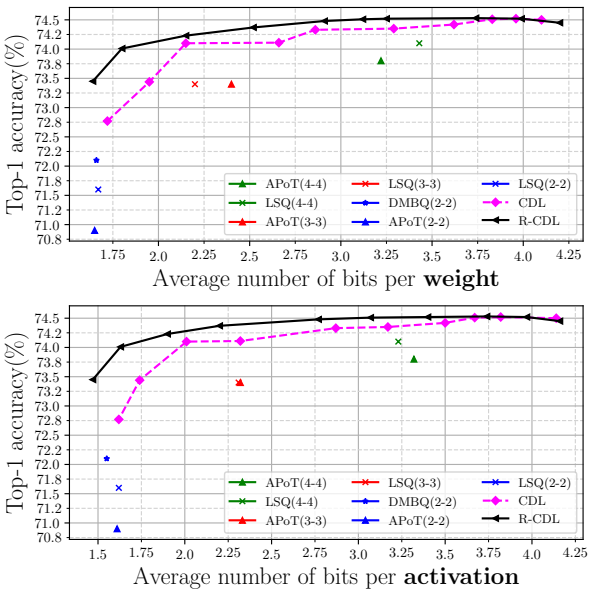


(b) ResNet-34 (FP accuracy:74.1).

Figure 5.13: Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) for LCQ, and R-CDL+LCQ on ImageNet: (a) ResNet-18, and (b) ResNet-34. All models are trained from **scratch**.



(a) ResNet-18 (FP accuracy:70.5).



(b) ResNet-34 (FP accuracy:74.1).

Figure 5.14: Comparison of models trained by CDL, R-CDL, and benchmark methods in terms of the Top-1 accuracy vs the average number of bits per weight (top)/activation (bottom) on ImageNet: (a) ResNet-18, and (b) ResNet-34. All models are trained from **pre-trained** FP models.

Chapter 6

Enhancing Diffusion Models for Inverse Problems with Covariance-Aware Posterior Sampling

The previous chapters concentrated on improving the performance of deep neural networks (DNNs) for classification tasks. Specifically, in Chapters Chapter 3 and Chapter 4, we applied information-theoretic principles to enhance the classification accuracy of DNNs in both centralized and distributed settings, showcasing the versatility of these concepts in optimizing predictive performance.

In contrast, this chapter shifts the focus from classification to generative modeling, with a particular emphasis on improving the performance of diffusion models. Diffusion models, as introduced in [89], have shown significant promise in generating high-quality samples by modeling complex data distributions. Here, we explore the potential of leveraging probabilistic techniques to enhance the efficiency and effectiveness of diffusion models, especially in the context of solving inverse problems. By utilizing the probabilistic framework of diffusion models, this chapter aims to demonstrate how these generative models can offer robust solutions to inverse problems, advancing their practical utility beyond traditional generative tasks.

Inverse problems exist in many disciplines of science and engineering. In computer vision, for example, tasks such as inpainting, deblurring, and super-resolution can be effectively modeled as inverse problems. Recently, denoising diffusion probabilistic models

(DDPMs) are shown to provide a promising solution to noisy linear inverse problems without the need for additional task-specific training. Specifically, with the prior provided by DDPMs, one can sample from the posterior by approximating the likelihood. In the literature, approximations of the likelihood are often based on the mean of conditional densities of the reverse process, which can be obtained using Tweedie's formula. To obtain a better approximation to the likelihood, in this chapter we first derive a closed-form formula for the covariance of the reverse process. Then, we propose a method based on finite difference method to approximate this covariance such that it can be readily obtained from the existing pre-trained DDPMs, thereby not increasing the complexity compared to existing approaches. Finally, based on the mean and approximated covariance of the reverse process, we present a new approximation to the likelihood. We refer to this method as **c**ovariance-**a**ware **d**iffusion **p**osterior **s**ampling (CA-DPS). Experimental results show that CA-DPS significantly improves reconstruction performance without requiring hyperparameter tuning. The code for the chapter is put in the supplementary materials.

6.1 Introduction

Denoising diffusion probabilistic models (DDPMs) [89] have made remarkable advancements in data synthesis over the past few years, revolutionizing fields such as image synthesis [**?**, 176, 234], video generation [90] and audio synthesis [117].

Given the powerful ability of DDPMs to estimate target distributions, one promising application is to use them to solve linear inverse problems such as denoising, inpainting, deblurring, and super-resolution. These tasks aim to recover a signal $\boldsymbol{x}_0$ (e.g., a face image) from a measurement $\boldsymbol{y}$, where $\boldsymbol{y}$ is related to $\boldsymbol{x}_0$ through the forward measurement operator $\mathbf{A}$ and detector noise $\boldsymbol{n}$ [33, 52, 167, 187, 188]. A naive approach to using DDPMs for solving inverse problems is to train a conditional DDPM to estimate the posterior $p(\boldsymbol{x}_0|\boldsymbol{y})$ through supervised learning. However, this approach can be computationally demanding, as it requires training separate models for different measurement operators.

To tackle the issue mentioned above, a newer method to approximate the posterior seeks to leverage pre-trained unconditional DDPMs that estimate the prior $p(\boldsymbol{x}_0)$, thereby avoiding the need for additional training. In this approach, the prior $p(\boldsymbol{x}_0)$ obtained from DDPMs is combined with the likelihood $p(\boldsymbol{y}|\boldsymbol{x}_0)$ to sample from the posterior distribution for inverse problems. However, because the likelihood term $p(\boldsymbol{y}|\boldsymbol{x}_0)$ is analytically intractable in the context of DDPMs due to their time-dependent nature, it must be approximated in some way [33, 78, 79].

To approximate the likelihood $p(\boldsymbol{y}|\boldsymbol{x}_0)$, there are mainly two approaches in the literature as we discuss in the sequel. The first approach relies on projections onto the measurement subspace [28, 35, 190]. However, these projection-based methods perform poorly in the presence of noise in the measurements, as the noise tends to be amplified during the generative process due to the ill-posed nature of inverse problems [33]. The second approach leverages the relationship $p(\boldsymbol{y}|\boldsymbol{x}_t) = \int p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0|\boldsymbol{x}_t)d\boldsymbol{x}_0$ in DDPMs; as such, assuming that $p(\boldsymbol{y}|\boldsymbol{x}_0)$ is known, one can approximate $p(\boldsymbol{y}|\boldsymbol{x}_t)$ by estimating $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. Although the distribution of $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is still intractable, the conditional mean $\tilde{\boldsymbol{x}}_0 = \mathbb{E}(\boldsymbol{x}_0|\boldsymbol{x}_t)$ can be analytically obtained using Tweedie's formula [56]. The conditional mean $\tilde{\boldsymbol{x}}_0$ is then used by [33] to approximate $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ by delta distribution $\delta(\boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_0)$, and used by [188] to approximate $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ with a Gaussian distribution $\mathcal{N}(\tilde{\boldsymbol{x}}_0, r_t^2\mathbf{I})$ with a heuristically selected variance r_t^2.

Nonetheless, approximating $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ using only its first moment (mean) is prone to sub-optimal performance due to biases in reconstruction [102, 148]. As a remedy, this chapter aims to improve the approximation of $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ by incorporating its second moment. Particularly, we derive a closed-form expression for the conditional covariance $\mathrm{Cov}(\boldsymbol{x}_0|\boldsymbol{x}_t)$ in DDPMs, and show that it depends on the Hessian $\mathbf{H}_t = \nabla^2_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$. Yet, the Hessian $\mathbf{H}_t$ is not directly available for DDPMs, as these models only provide the score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$. To address this, we approximate the Hessian $\mathbf{H}_t$ with a diagonal matrix $\tilde{\mathbf{H}}_t$, where the diagonal elements are derived from the gradient vector $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$ using the finite difference method. In this regard, this approximation can be easily obtained from existing pre-trained DDPMs, thus avoiding any additional complexities. Then, using $\tilde{\mathbf{H}}_t$ we obtain an approximation to the true $\mathrm{Cov}(\boldsymbol{x}_0|\boldsymbol{x}_t)$, which we denote by $\tilde{\boldsymbol{\Sigma}}_t$. Henceforth, we approximate $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ with $\mathcal{N}(\tilde{\boldsymbol{x}}_0, \tilde{\boldsymbol{\Sigma}}_t)$. We refer to this method of deploying DDPMs for solving inverse probelms as **c**ovariance-**a**ware **d**iffusion **p**osterior **s**ampling (CA-DPS) hereafter.

It is worth noting that while some prior work has investigated using second-order approximations for the posterior, these approaches either (i) rely on the availability of second-order scores or the Jacobian of the first-order score from the diffusion model [16], or (ii) require retraining existing unconditional diffusion models to output both posterior mean and variance, which increases time and memory complexity [167] (see Section 6.2.3 for further discussion).

The contributions of this chapter are summarized as follows:

- For a general exponential conditional distribution family, we derive a closed-form expression for its posterior covariance (see Theorem 4).
- Using Theorem 4, we determine a closed-form formula for the conditional covariance

$\mathrm{Cov}(\boldsymbol{x}_0|\boldsymbol{x}_t)$ in DDPMs (see Corollary 1). Based on this closed-form formula, we then introduce a method based on the finite difference approach to approximate the conditional covariance $\mathrm{Cov}(\boldsymbol{x}_0|\boldsymbol{x}_t)$, allowing it to be readily obtained from existing pre-trained DDPMs.

• By conducting experiments on two popular datasets FFHQ [108] and ImageNet [46], we show that CA-DPS outperforms existing approaches across various tasks, including inpainting, deblurring, and super-resolution, while also eliminating the need for hyperparameter tuning.

6.2 Related Work

6.2.1 Approaches for Solving Inverse Problems

Historical Context of Inverse Problems

Inverse problems have a longstanding and rich history, with methodologies evolving significantly over the years [6, 173]. Although providing a comprehensive review of this extensive field is beyond the scope of this work, we outline key developments to provide context. Early efforts, particularly during the 2000s, often formulated inverse problems as optimization tasks [20, 43, 51, 60, 184]. These approaches focused on balancing two primary objectives: fidelity to observed data and regularization terms that encouraged desirable properties in the solution. Examples include enforcing smoothness [8] or sparsity in specific representations, such as wavelets and dictionaries [20, 43, 51, 60, 69]. Regularization-based methods provided a structured way to incorporate prior knowledge about the underlying signal, laying the groundwork for more advanced approaches.

The Impact of Deep Learning

The emergence of deep learning marked a paradigm shift in solving inverse problems [160]. Researchers began leveraging large paired datasets to train neural networks to directly map measurements to clean signals [50, 133, 197]. These data-driven methods optimize for specific reconstruction objectives, often minimizing a loss function to reduce reconstruction errors while tailoring solutions for particular goals, such as perceptual quality [98, 125].

Traditional methods aimed at recovering a single reconstruction through approaches like minimizing the mean squared error (MMSE) or identifying the Maximum a Posteriori (MAP) estimate, which seeks to find the most probable reconstruction $\boldsymbol{x}$ by maximizing

the posterior probability $p(\boldsymbol{x}|\boldsymbol{y})$. However, these techniques can suffer from "regression to the mean," where the predicted reconstruction represents an average of plausible solutions. This averaging can lead to outputs lacking critical details or deviating from the desired solution space [14, 45]. For example, minimizing a simple distortion metric, such as the L2 loss, may produce blurry reconstructions that fail to capture the diversity of plausible solutions [14]. While effective for certain applications, this limitation motivates alternative approaches capable of generating more diverse and detailed reconstructions.

Integrating Denoising into Inverse Problems

Recent advances have revealed a compelling link between denoising algorithms and inverse problems. State-of-the-art denoisers, often built using deep learning, implicitly encode rich prior knowledge about natural signals. These denoisers can be integrated into optimization frameworks to enhance the performance of inverse problem solvers [24, 37, 175, 191, 200]. By using pre-trained denoisers as priors, this approach bridges the gap between traditional regularization techniques and modern data-driven methods, achieving exceptional results across a wide range of inverse problems. The "plug-and-play" (PnP) framework is one notable example, where the denoiser is used as a modular component within optimization routines, enabling the integration of powerful priors without retraining [24,191]. This fusion of denoising and optimization offers a flexible and robust paradigm for tackling complex inverse problems.

Bayesian Inference for Inverse Problems

Another perspective on inverse problems is through the lens of Bayesian inference. Instead of recovering a single solution, this approach focuses on sampling from the posterior distribution $p(\boldsymbol{x}|\boldsymbol{y})$, which represents the range of plausible signals $\boldsymbol{x}$ given the observed measurements $\boldsymbol{y}$. This probabilistic framework provides a more comprehensive understanding of the solution space, capturing uncertainty and variability in the reconstructions. By modeling the posterior distribution, Bayesian methods allow for the generation of diverse reconstructions that better align with the true distribution of the underlying signals. These methods are particularly appealing for applications where uncertainty quantification is critical or when the solution space contains multiple plausible solutions.

6.2.2 Diffusion Models for Inverse Problems

The use of diffusion models for solving inverse problems by sampling from the posterior has recently gained significant traction in various fields, including image denoising [110], compressed sensing [15, 105], magnetic resonance imaging (MRI) [101], projecting score-based stochastic differential equations (SDEs) [189], and variational approaches [59, 144]. In particular, the most relevant line of work, which we will review in detail in Section 6.3.2, involves using Tweedie's formula [56] to approximate the smoothed likelihood, as deployed in methods like diffusion posterior sampling (DPS) [33] and pseudo-guided diffusion models (ΠGDM) [188]. Similar strategies are also employed using singular-value decomposition (SVD) based approaches [111].

6.2.3 Higher Order Approximation of reverse process

The approach presented in this chapter can be seen as a variant of high-order denoising score matching [140, 148], which aims to train a diffusion model capable of learning the higher-order moments of the reverse process. However, these methods are typically limited to small-scale datasets due to their computational complexity.

Similarly to our work, [16] aims to estimate the covariance of the reverse process. However, their method require that the second-order scores or the Jacobian of the first-order score be available by the diffusion model. In addition, [167] proposed a method to optimize the posterior likelihood. They proposed two methods for (i) when reverse covariance prediction is available from the given unconditional diffusion model, and (ii) when reverse covariance prediction is not available. Their first approach is different from our proposed method as our method do not require reverse covariance to be available. Additionally, their second approach is based on Monte Carlo estimation which incurs extra complexity to the sampling process. We also acknowledge the work of [192], who explored a maximum-a-posteriori approach to estimate the moments of the posterior.

6.3 Background and Preliminaries

6.3.1 Diffusion Models

Diffusion models characterize a generative process as the reverse of a noise addition process. In particular, [190] introduced the Itô stochastic differential equation (SDE) to describe

the noise addition process (i.e., the forward SDE) for the data $\boldsymbol{x}(t)$ over the time interval $t \in [0, T]$, where $\boldsymbol{x}(t) \in \mathbb{R}^d$ for all t.

In this chapter, we adopt the variance-preserving form of the SDE (VP-SDE) [190], which is equivalent to the DDPM framework [89] whose equation is given as follows

$$d\boldsymbol{x} = -\frac{\beta(t)}{2}\boldsymbol{x}\, dt + \sqrt{\beta(t)}\, d\boldsymbol{w}, \tag{6.1}$$

where $\beta(t) : \mathbb{R} \to \mathbb{R}^+$ represents the noise schedule of the process, which is typically chosen as a monotonically increasing linear function of t [89]. The term $\boldsymbol{w}$ denotes the standard d-dimensional Wiener process. The data distribution is defined at $t = 0$, i.e., $\boldsymbol{x}(0) \sim p_{\text{data}}$, while a simple and tractable distribution, such as an isotropic Gaussian, is achieved at $t = T$, i.e., $\boldsymbol{x}(T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The goal is to recover the data-generating distribution from the tractable distribution. This can be accomplished by formulating the corresponding reverse SDE for Equation (6.1), as derived in [3]:

$$d\boldsymbol{x} = \left[-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right] dt + \sqrt{\beta(t)} d\bar{\boldsymbol{w}}, \tag{6.2}$$

where dt represents time running backward, and $d\bar{\boldsymbol{w}}$ corresponds to the standard Wiener process running in reverse. The drift function now depends on the time-dependent score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$, which is approximated by a neural network $\boldsymbol{s}_\theta$ trained using denoising score matching [201]:

$$\theta^* = \arg\min_\theta \mathbb{E}_{t\sim U(\varepsilon,1),\boldsymbol{x}(t)\sim p(\boldsymbol{x}(t)|\boldsymbol{x}(0)),\boldsymbol{x}(0)\sim p_{\text{data}}} \tag{6.3}$$

$$\left[\|\boldsymbol{s}_\theta(\boldsymbol{x}(t), t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}(t)|\boldsymbol{x}(0))\|_2^2\right], \tag{6.4}$$

where $\varepsilon \simeq 0$ represents a small positive constant. Once the optimal parameters θ^* are obtained through Equation (6.3), the approximation $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) \simeq \boldsymbol{s}_{\theta^*}(\boldsymbol{x}_t, t)$ can be used as a plug-in estimate to replace the score function in Equation (6.2).

Discretizing Equation (6.2) and solving it yields samples from the data distribution $p(\boldsymbol{x}_0)$, which is the ultimate goal of generative modeling. In discrete settings with N time steps, we define $\boldsymbol{x}_i \triangleq \boldsymbol{x}(iT/N)$ and $\beta_i \triangleq \beta(iT/N)$. Following [89], we then introduce $\alpha_i \triangleq 1 - \beta_i$ and $\bar{\alpha}_i \triangleq \prod_{j=1}^{i} \alpha_j$.

6.3.2 Diffusion Models for Solving Inverse Problems

We consider the linear inverse problems for reconstructing an unknown signal $\boldsymbol{x}_0 \in \mathbb{R}^d$ from noisy measurements $\boldsymbol{y} \in \mathbb{R}^m$:

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x}_0 + \boldsymbol{n}, \tag{6.5}$$

where $\mathbf{A} \in \mathbb{R}^{m \times d}$ is a known measurement operator and $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is an i.i.d. additive Gaussian noise with a known standard deviation of σ. This gives a likelihood function $p(\boldsymbol{y}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{y}|\mathbf{A}\boldsymbol{x}_0, \sigma^2\mathbf{I})$.

Usually, we are interested in the case when $m < d$, which follows many real-world scenarios. When $m < d$, the problem is ill-posed and some kind of *prior* is necessary to obtain a meaningful solution. In the Bayesian framework, one utilizes $p(\boldsymbol{x}_0)$ as the *prior*, and samples from the *posterior* $p(\boldsymbol{x}_0|\boldsymbol{y})$, where the relationship is formally established with the Bayes' rule: $p(\boldsymbol{x}_0|\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x}_0)p(\boldsymbol{x}_0)/p(\boldsymbol{y})$. Leveraging the diffusion model as the prior, it is straightforward to modify Equation (6.2) to arrive at the reverse diffusion sampler for sampling from the posterior distribution:

$$d\boldsymbol{x} = \Big[-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)(\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)) \Big] dt + \sqrt{\beta(t)} d\bar{\boldsymbol{w}}, \tag{6.6}$$

where we have used the fact that

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{y}) = \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t). \tag{6.7}$$

In Equation (6.6), there are two terms that need to be computed: the score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ and the likelihood $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$. To compute the former, involving $p_t(\boldsymbol{x}_t)$, we can directly use the pre-trained score function $\boldsymbol{s}_{\theta^*}$. However, the latter term is challenging to obtain in closed-form due to its dependence on time t (note that there is only an explicit relationship between $\boldsymbol{y}$ and $\boldsymbol{x}_0$). As such, the likelihood $p_t(\boldsymbol{y}|\boldsymbol{x}_t)$ shall be estimated. One approach to achieve this estimation is to factorize $p(\boldsymbol{y}|\boldsymbol{x}_t)$ as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}_t) = \int p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{x}_t) p(\boldsymbol{x}_0|\boldsymbol{x}_t) d\boldsymbol{x}_0 = \int p(\boldsymbol{y}|\boldsymbol{x}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_t) d\boldsymbol{x}_0, \tag{6.8}$$

where the second equality comes from that $\boldsymbol{y}$ and $\boldsymbol{x}_t$ are conditionally independent on $\boldsymbol{x}_0$. Assuming that the measurement model $p(\boldsymbol{y}|\boldsymbol{x}_0)$ is known, based on Equation (6.8), one can approximate $p(\boldsymbol{y}|\boldsymbol{x}_t)$ by approximating $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. Although the exact form of $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is intractable, the conditional mean of $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$ under $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$, denoted by $\tilde{\boldsymbol{x}}_0 = \mathbb{E}(\boldsymbol{x}_0|\boldsymbol{x}_t)$, can be analytically obtained using Tweedie's formula [56]:

Proposition 2 (Tweedie's formula)**.** *Let $p(\boldsymbol{y}|\boldsymbol{\eta})$ belong to the exponential family distribution*

$$p(\boldsymbol{y}|\boldsymbol{\eta}) = p_0(\boldsymbol{y}) \exp(\boldsymbol{\eta}^\top T(\boldsymbol{y}) - \varphi(\boldsymbol{\eta})), \tag{6.9}$$

where $\boldsymbol{\eta}$ is the canonical vector of the family, $T(\boldsymbol{y})$ is some function of $\boldsymbol{y}$, and $\varphi(\boldsymbol{\eta})$ is the cumulant generation function which normalizes the density, and $p_0(\boldsymbol{y})$ is the density up to the scale factor when $\boldsymbol{\eta} = \boldsymbol{0}$. Then, the posterior mean $\mathbb{E}[\boldsymbol{\eta}|\boldsymbol{y}]$ should satisfy

$$(\nabla_{\boldsymbol{y}} T(\boldsymbol{y}))^\top \mathbb{E}[\boldsymbol{\eta}|\boldsymbol{y}] = \nabla_{\boldsymbol{y}} \log p(\boldsymbol{y}) - \nabla_{\boldsymbol{y}} \log p_0(\boldsymbol{y}). \tag{6.10}$$

Now, note that in DDPM sampling process we have

$$p(\boldsymbol{x}_t|\boldsymbol{x_0}) = \frac{1}{(2\pi(1-\bar{\alpha}(t)))^{d/2}} \exp\left(-\frac{\|\boldsymbol{x}_t - \sqrt{\bar{\alpha}(t)}\boldsymbol{x_0}\|^2}{2(1-\bar{\alpha}(t))}\right), \tag{6.11}$$

which is a Gaussian distribution. The corresponding canonical decomposition is then given by

$$p(\boldsymbol{x}_t|\boldsymbol{x_0}) = p_0(\boldsymbol{x}_t) \exp\left(\boldsymbol{x_0}^\top T(\boldsymbol{x}_t) - \varphi(\boldsymbol{x_0})\right), \tag{6.12}$$

where

$$p_0(\boldsymbol{x}_t) := \frac{1}{(2\pi(1-\bar{\alpha}(t)))^{d/2}} \exp\left(-\frac{\|\boldsymbol{x}_t\|^2}{2(1-\bar{\alpha}(t))}\right), \tag{6.13a}$$

$$T(\boldsymbol{x}_t) := \frac{\sqrt{\bar{\alpha}(t)}}{1-\bar{\alpha}(t)}\boldsymbol{x}_t, \tag{6.13b}$$

$$\varphi(\boldsymbol{x_0}) := \frac{\bar{\alpha}(t)\|\boldsymbol{x_0}\|^2}{2(1-\bar{\alpha}(t))}. \tag{6.13c}$$

Therefore, substituting Equation (6.13) in Equation (6.10), we obtain

$$\frac{\sqrt{\bar{\alpha}(t)}}{1-\bar{\alpha}(t)}\tilde{\boldsymbol{x}}_0 = \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \frac{1}{1-\bar{\alpha}(t)}\boldsymbol{x}_t,$$

which leads to

$$\tilde{\boldsymbol{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}(t)}}\left(\boldsymbol{x}_t + (1-\bar{\alpha}(t))\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right). \tag{6.14}$$

Then, two recent studies deploy the expected value $\tilde{\boldsymbol{x}}_0$ to approximate $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$, which we discuss their methodologies in detail in the following.

(I) DPS [33] Denoising posterior sampling (DPS) approximates $p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$ using a delta distribution $\delta(\boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_0)$ centered at the posterior mean estimate $\tilde{\boldsymbol{x}}_0$. As such, the likelihood $p_t(\boldsymbol{y}|\boldsymbol{x}_t)$ is approximated by

$$p_t(\boldsymbol{y}|\boldsymbol{x}_t) \approx \int p(\boldsymbol{y}|\boldsymbol{x}_0)\delta(\boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_0)\mathrm{d}\boldsymbol{x}_0 = p(\boldsymbol{y}|\boldsymbol{x}_0 = \tilde{\boldsymbol{x}}_0). \tag{6.15}$$

However, directly using Eq. Equation (6.15) does not perform well in practice, and [33] empirically adjusts the strength of guidance by approximating the likelihood score $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$ with $-\zeta_t \nabla_{\boldsymbol{x}_t} \|\boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0\|_2^2$, where $\zeta_t = \zeta / \|\boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0\|_2$ with a hyper-parameter ζ.

(II) ΠGDM [188] The delta distribution used in DPS is a very rough approximation to $p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as it completely disregards the uncertainty of $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$. As t increases, the uncertainty in $p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$ becomes larger and is closed to the original data distribution $p(\boldsymbol{x}_0)$. Thus, it is more reasonable to choose a positive r_t. In ΠGDM, r_t is heuristically selected as $\sqrt{\sigma_t^2/(1+\sigma_t^2)}$ under the assumption that $p(\boldsymbol{x}_0)$ is the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. In such case, the likelihood $p_t(\boldsymbol{y}|\boldsymbol{x}_t)$ is approximated by

$$\begin{aligned} p_t(\boldsymbol{y}|\boldsymbol{x}_t) &\approx \int \mathcal{N}(\boldsymbol{y}|\mathbf{A}\boldsymbol{x}_0, \sigma^2\mathbf{I})\mathcal{N}(\boldsymbol{x}_0|\tilde{\boldsymbol{x}}_0, r_t^2\mathbf{I})\mathrm{d}\boldsymbol{x}_0 \\ &= \mathcal{N}(\boldsymbol{y}|\mathbf{A}\tilde{\boldsymbol{x}}_0, \sigma^2\mathbf{I} + r_t^2\mathbf{A}\mathbf{A}^\top). \end{aligned} \tag{6.16}$$

6.4 Covariance-Aware Diffusion Posterior Sampling

In this section, we aim to improve the approximation of the reverse process $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ compared to DPS and ΠGDM. Specifically, instead of heuristically approximating the conditional covariance of $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$ as done by ΠGDM, we derive a closed-form formula for it. To this end, we first introduce the following theorem.

Theorem 4. *Under the same conditions as in Proposition 2, the posterior covariance* $Cov(\boldsymbol{\eta}|\boldsymbol{y})$ *satisfies*

$$\begin{aligned} &(\nabla_{\boldsymbol{y}} T(\boldsymbol{y}))^\top Cov(\boldsymbol{\eta}|\boldsymbol{y}) \nabla_{\boldsymbol{y}} T(\boldsymbol{y}) \\ &\quad = \nabla_{\boldsymbol{y}}^2 \log p(\boldsymbol{y}) - \nabla_{\boldsymbol{y}}^2 \log p_0(\boldsymbol{y}) - \nabla_{\boldsymbol{y}}^2 T(\boldsymbol{y}) \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}), \end{aligned} \tag{6.17}$$

where $\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})$ *is obtained using Tweedie's formula in Proposition 2. Additionally, the operator* $\odot$ *denotes a contraction operation between the three dimensional tensor* $\nabla_{\boldsymbol{y}}^2 T(\boldsymbol{y})$

and the vector $\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})$*. Specifically, assuming that* $\boldsymbol{y} \in \mathbb{R}^r$ *and* $\boldsymbol{\eta} \in \mathbb{R}^k$ *(which yields* $\nabla^2_{\boldsymbol{y}} T(\boldsymbol{y}) \in \mathbb{R}^{r\times r\times k}$ *and* $\mathbb{E}[\boldsymbol{\eta}|\boldsymbol{y}] \in \mathbb{R}^k$*), then* $\nabla^2_{\boldsymbol{y}} T(\boldsymbol{y}) \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}) \in \mathbb{R}^{r\times r}$ *is defined as*

$$[\nabla^2_{\boldsymbol{y}} T(\boldsymbol{y}) \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})]_{ij} = \sum_k [\nabla^2_{\boldsymbol{y}} T(\boldsymbol{y})]_{ijk} \mathbb{E}(\eta_k|\boldsymbol{y}). \tag{6.18}$$

Proof. The marginal distribution $p(\boldsymbol{y})$ could be expressed as

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{\eta}) p(\boldsymbol{\eta}) d\boldsymbol{\eta} \tag{6.19}$$

$$= \int p_0(\boldsymbol{y}) \exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y}) - \varphi(\boldsymbol{\eta})\big) p(\boldsymbol{\eta}) d\boldsymbol{\eta}. \tag{6.20}$$

Then, the derivative of the marginal distribution $p(\boldsymbol{y})$ with respect to y becomes

$$\begin{aligned}
\nabla_{y_i} p(\boldsymbol{y}) &= \nabla_{y_i} p_0(\boldsymbol{y}) \int \exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y}) - \varphi(\boldsymbol{\eta})\big) p(\boldsymbol{\eta}) d\boldsymbol{\eta} \\
&\quad + \int (\nabla_{y_i} T(\boldsymbol{y}))^\top \boldsymbol{\eta} p_0(\boldsymbol{y}) \exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y}) - \varphi(\boldsymbol{\eta})\big) p(\boldsymbol{\eta}) d\boldsymbol{\eta} && (6.21) \\
&= \frac{\nabla_{y_i} p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})} \int p(\boldsymbol{y}|\boldsymbol{\eta}) p(\boldsymbol{\eta}) d\boldsymbol{\eta} + (\nabla_{y_i} T(\boldsymbol{y}))^\top \int \boldsymbol{\eta} p(\boldsymbol{y}|\boldsymbol{\eta}) p(\boldsymbol{\eta}) d\boldsymbol{\eta} && (6.22) \\
&= \frac{\nabla_{y_i} p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})} p(\boldsymbol{y}) + (\nabla_{y_i} T(\boldsymbol{y}))^\top \int \boldsymbol{\eta} p(\boldsymbol{y}, \boldsymbol{\eta}) d\boldsymbol{\eta} && (6.23) \\
& && (6.24)
\end{aligned}$$

Therefore,

$$\frac{\nabla_{\boldsymbol{y}} p(\boldsymbol{y})}{p(\boldsymbol{y})} = \frac{\nabla_{\boldsymbol{y}} p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})} + (\nabla_{\boldsymbol{y}} T(\boldsymbol{y}))^\top \int \boldsymbol{\eta} p(\boldsymbol{\eta}|\boldsymbol{y}) d\boldsymbol{\eta} \tag{6.25}$$

which is equivalent to

$$(\nabla_{\boldsymbol{y}} T(\boldsymbol{y}))^\top \mathbb{E}[\boldsymbol{\eta}|\boldsymbol{y}] = \nabla_{\boldsymbol{y}} \log p(\boldsymbol{y}) - \nabla_{\boldsymbol{y}} \log p_0(\boldsymbol{y}) \tag{6.26}$$

Now, we take another derivative w.r.t. $\boldsymbol{y}[j]$ from both sides of Equation (6.21):

$$\begin{aligned}
\nabla_{y_j}\nabla_{y_i}p(\boldsymbol{y}) &= \nabla_{y_j}\nabla_{y_i}p_0(\boldsymbol{y})\int \exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y})-\varphi(\boldsymbol{\eta})\big)p(\boldsymbol{\eta})d\boldsymbol{\eta}\\
&\quad+\nabla_{y_i}p_0(\boldsymbol{y})(\nabla_{y_j}T(\boldsymbol{y}))^\top\int \boldsymbol{\eta}\exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y})-\varphi(\boldsymbol{\eta})\big)p(\boldsymbol{\eta})d\boldsymbol{\eta}\\
&\quad+\Big(\nabla_{y_j}\nabla_{y_i}T(\boldsymbol{y})\Big)^\top\odot p_0(\boldsymbol{y})\int \boldsymbol{\eta}\exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y})-\varphi(\boldsymbol{\eta})\big)p(\boldsymbol{\eta})d\boldsymbol{\eta}\\
&\quad+\nabla_{y_j}p_0(\boldsymbol{y})(\nabla_{y_i}T(\boldsymbol{y}))^\top\int \boldsymbol{\eta}\exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y})-\varphi(\boldsymbol{\eta})\big)p(\boldsymbol{\eta})d\boldsymbol{\eta}\\
&\quad+(\nabla_{y_i}T(\boldsymbol{y}))^\top p_0(\boldsymbol{y})\int \boldsymbol{\eta}\boldsymbol{\eta}^\top\nabla_{y_j}T(\boldsymbol{y})\exp\big(\boldsymbol{\eta}^\top T(\boldsymbol{y})-\varphi(\boldsymbol{\eta})\big)p(\boldsymbol{\eta})d\boldsymbol{\eta} \qquad (6.27)\\
&=\frac{\nabla_{y_j}\nabla_{y_i}p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})}p(\boldsymbol{y})+\frac{\nabla_{y_i}p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})}(\nabla_{y_j}T(\boldsymbol{y}))^\top\int \boldsymbol{\eta}p(\boldsymbol{y},\boldsymbol{\eta})d\boldsymbol{\eta}\\
&\quad+\Big(\big(\nabla_{y_j}\nabla_{y_i}T(\boldsymbol{y})\big)^\top\odot p_0(\boldsymbol{y})+\nabla_{y_j}p_0(\boldsymbol{y})(\nabla_{y_i}T(\boldsymbol{y}))^\top\Big)\frac{\int \boldsymbol{\eta}p(\boldsymbol{y},\boldsymbol{\eta})d\boldsymbol{\eta}}{p_0(\boldsymbol{y})}\\
&\quad+(\nabla_{y_i}T(\boldsymbol{y}))^\top p_0(\boldsymbol{y})\int \boldsymbol{\eta}\boldsymbol{\eta}^\top\nabla_{y_j}T(\boldsymbol{y})p(\boldsymbol{y},\boldsymbol{\eta})d\boldsymbol{\eta} \qquad (6.28)
\end{aligned}$$

Now, we divide both sides of Equation (6.28) by $p(\boldsymbol{y})$ to get

$$\begin{aligned}
\frac{\nabla_{y_j}\nabla_{y_i}p(\boldsymbol{y})}{p(\boldsymbol{y})} &= \frac{\nabla_{y_j}\nabla_{y_i}p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})}+\frac{\nabla_{y_i}p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})}(\nabla_{y_j}T(\boldsymbol{y}))^\top\int \boldsymbol{\eta}p(\boldsymbol{\eta}|\boldsymbol{y})d\boldsymbol{\eta}\\
&\quad+\big(\nabla_{y_j}\nabla_{y_i}T(\boldsymbol{y})\big)^\top\odot\int \boldsymbol{\eta}p(\boldsymbol{\eta}|\boldsymbol{y})d\boldsymbol{\eta}+\frac{\nabla_{y_j}p_0(\boldsymbol{y})(\nabla_{y_i}T(\boldsymbol{y}))^\top}{p_0(\boldsymbol{y})}\int \boldsymbol{\eta}p(\boldsymbol{\eta}|\boldsymbol{y})d\boldsymbol{\eta}\\
&\quad+(\nabla_{y_i}T(\boldsymbol{y}))^\top p_0(\boldsymbol{y})\int \boldsymbol{\eta}\boldsymbol{\eta}^\top\nabla_{y_j}T(\boldsymbol{y})p(\boldsymbol{\eta}|\boldsymbol{y})d\boldsymbol{\eta} \qquad (6.29)\\
&=\frac{\nabla_{y_j}\nabla_{y_i}p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})}+\frac{\nabla_{y_i}p_0(\boldsymbol{y})(\nabla_{y_j}T(\boldsymbol{y}))^\top}{p_0(\boldsymbol{y})}\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})\\
&\quad+\big(\nabla_{y_j}\nabla_{y_i}T(\boldsymbol{y})\big)^\top\odot\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})+\frac{\nabla_{y_j}p_0(\boldsymbol{y})(\nabla_{y_i}T(\boldsymbol{y}))^\top}{p_0(\boldsymbol{y})}\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})\\
&\quad+(\nabla_{y_i}T(\boldsymbol{y}))^\top\mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^\top|\boldsymbol{y})\nabla_{y_j}T(\boldsymbol{y}) \qquad (6.30)
\end{aligned}$$

By isolating the term $(\nabla_{y_i}T(\boldsymbol{y}))^\top\mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^\top|\boldsymbol{y})\nabla_{y_j}T(\boldsymbol{y})$ in Equation (6.30), and subtracting

$(\nabla_{y_i} T(\boldsymbol{y}))^\top \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})\mathbb{E}(\boldsymbol{\eta}^\top|\boldsymbol{y})\nabla_{y_j} T(\boldsymbol{y})$ from it, we obtain

$$(\nabla_{y_i} T(\boldsymbol{y}))^\top \mathrm{Cov}(\boldsymbol{\eta}|\boldsymbol{y})\nabla_{y_j} T(\boldsymbol{y}) \tag{6.31}$$

$$= (\nabla_{y_i} T(\boldsymbol{y}))^\top \mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^\top|\boldsymbol{y})\nabla_{y_j} T(\boldsymbol{y}) - (\nabla_{y_i} T(\boldsymbol{y}))^\top \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})\mathbb{E}(\boldsymbol{\eta}^\top|\boldsymbol{y})\nabla_{y_j} T(\boldsymbol{y}) \tag{6.32}$$

$$\begin{aligned} &= \frac{\nabla_{y_j}\nabla_{y_i} p(\boldsymbol{y})}{p(\boldsymbol{y})} - \frac{\nabla_{y_j}\nabla_{y_i} p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})} \\ &\quad - \frac{\nabla_{y_i} p_0(\boldsymbol{y})(\nabla_{y_j} T(\boldsymbol{y}))^\top}{p_0(\boldsymbol{y})}\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}) - \big(\nabla_{y_j}\nabla_{y_i} T(\boldsymbol{y})\big)^\top \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}) \\ &\quad - \frac{\nabla_{y_j} p_0(\boldsymbol{y})(\nabla_{y_i} T(\boldsymbol{y}))^\top}{p_0(\boldsymbol{y})}\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}) - (\nabla_{y_i} T(\boldsymbol{y}))^\top \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y})\mathbb{E}(\boldsymbol{\eta}^\top|\boldsymbol{y})\nabla_{y_j} T(\boldsymbol{y}) \end{aligned} \tag{6.33}$$

$$\begin{aligned} &= \frac{\nabla_{y_j}\nabla_{y_i} p(\boldsymbol{y})}{p(\boldsymbol{y})} - \frac{\nabla_{y_j}\nabla_{y_i} p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})} - \nabla_{y_i}\log p_0(\boldsymbol{y})\Big(\nabla_{y_j}\log p(\boldsymbol{y}) - \nabla_{y_j}\log p_0(\boldsymbol{y})\Big) \\ &\quad - \nabla_{y_j}\log p_0(\boldsymbol{y})\Big(\nabla_{y_i}\log p(\boldsymbol{y}) - \nabla_{y_i}\log p_0(\boldsymbol{y})\Big) \\ &\quad - \big(\nabla_{y_j}\nabla_{y_i} T(\boldsymbol{y})\big)^\top \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}) \\ &\quad - \Big(\nabla_{y_i}\log p(\boldsymbol{y}) - \nabla_{y_i}\log p_0(\boldsymbol{y})\Big)\Big(\nabla_{y_j}\log p(\boldsymbol{y}) - \nabla_{y_j}\log p_0(\boldsymbol{y})\Big) \end{aligned} \tag{6.34}$$

$$\begin{aligned} &= \frac{\nabla_{y_j}\nabla_{y_i} p(\boldsymbol{y})}{p(\boldsymbol{y})} - \frac{\nabla_{y_j}\nabla_{y_i} p_0(\boldsymbol{y})}{p_0(\boldsymbol{y})} + \nabla_{y_i}\log p_0(\boldsymbol{y})\nabla_{y_j}\log p_0(\boldsymbol{y}) \\ &\quad - \nabla_{y_i}\log p(\boldsymbol{y})\nabla_{y_j}\log p(\boldsymbol{y}) - \big(\nabla_{y_j}\nabla_{y_i} T(\boldsymbol{y})\big)^\top \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}) \end{aligned} \tag{6.35}$$

$$= \nabla_{y_j}\nabla_{y_i}\log p(\boldsymbol{y}) - \nabla_{y_j}\nabla_{y_i}\log p_0(\boldsymbol{y}) - \big(\nabla_{y_j}\nabla_{y_i} T(\boldsymbol{y})\big)^\top \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}). \tag{6.36}$$

Thus

$$(\nabla_{\boldsymbol{y}} T(\boldsymbol{y}))^\top \mathrm{Cov}(\boldsymbol{\eta}|\boldsymbol{y})\nabla_{\boldsymbol{y}} T(\boldsymbol{y}) = \nabla^2_{\boldsymbol{y}}\log p(\boldsymbol{y}) - \nabla^2_{\boldsymbol{y}}\log p_0(\boldsymbol{y}) - \nabla^2_{\boldsymbol{y}} T(\boldsymbol{y}) \odot \mathbb{E}(\boldsymbol{\eta}|\boldsymbol{y}), \tag{6.37}$$

which concludes the proof. □

Next, we use Theorem 4 for DDPMs to find a closed-form expression for the conditional covariance $\mathrm{Cov}(\boldsymbol{x_0}|\boldsymbol{x}_t)$.

Corollary 2. *Using Equation (6.13) in Theorem 4 we obtain*

$$\Big(\frac{\sqrt{\bar\alpha(t)}}{1-\bar\alpha(t)}\Big)^2 Cov(\boldsymbol{x_0}|\boldsymbol{x}_t) = \nabla^2_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t) + \frac{1}{1-\bar\alpha(t)}\mathbf{I}, \tag{6.38}$$

which leads to

$$Cov(\boldsymbol{x_0}|\boldsymbol{x}_t) = \frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)}\Big(\mathbf{I} + (1-\bar{\alpha}(t))\nabla^2_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)\Big). \tag{6.39}$$

As seen in Equation (6.39), $\mathrm{Cov}(\boldsymbol{x_0}|\boldsymbol{x}_t)$ depends on the Hessian $\mathbf{H}_t = \nabla^2_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$. Nevertheless, the Hessian $\mathbf{H}_t$ is not available for DDPMs (note that DDPMs only return the score function $\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$). To this aim, we shall resort to approximating $\mathbf{H}_t$. Given that the sampling process in DDPMs is inherently time-consuming, our approximation method for $\mathbf{H}_t$ needs to be straightforward to avoid adding complexity to the sampling process. Thus, we approximate the Hessian $\mathbf{H}_t$ with a diagonal matrix $\tilde{\mathbf{H}}_t$, where the diagonal elements are obtained from the gradient vector $\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$ using the finite difference method:

$$\tilde{\mathbf{H}}_t = \frac{\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_{t-1}}\log p_{t-1}(\boldsymbol{x}_{t-1})}{t-(t-1)}. \tag{6.40}$$

In this respect, $\tilde{\mathbf{H}}_t$ can be readily obtained from the existing pre-trained DDPMs, and thereby does not incur any extra complexities. Then, using $\tilde{\mathbf{H}}_t$ in Equation (6.39), we obtain an approximation to the true $\mathrm{Cov}(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as

$$\tilde{\boldsymbol{\Sigma}}_t = \frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)}\Big(\mathbf{I} + (1-\bar{\alpha}(t))\tilde{\mathbf{H}}_t\Big). \tag{6.41}$$

Henceforth, we approximate $p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as follows

$$p_t(\boldsymbol{x}_0|\boldsymbol{x}_t) \approx \mathcal{N}(\tilde{\boldsymbol{x}}_0, \tilde{\boldsymbol{\Sigma}}_t), \tag{6.42}$$

which leads to the following approximation for the likelihood $p_t(\boldsymbol{y}|\boldsymbol{x}_t)$

$$\begin{aligned} p_t(\boldsymbol{y}|\boldsymbol{x}_t) &\approx \int \mathcal{N}(\mathbf{A}\boldsymbol{x}_0, \sigma^2\mathbf{I})\mathcal{N}(\tilde{\boldsymbol{x}}_0, \tilde{\boldsymbol{\Sigma}}_t)\mathrm{d}\boldsymbol{x}_0 \\ &= \mathcal{N}(\mathbf{A}\tilde{\boldsymbol{x}}_0, \sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top). \end{aligned} \tag{6.43}$$

Lastly, the gradient of the log-likelihood, $\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$, can be approximated using the Jacobian-vector product, similar to the approach in [188]:

$$\begin{aligned} &\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t) \\ &\approx \big(\nabla_{\boldsymbol{x}_t}\tilde{\boldsymbol{x}}_0\big)\mathbf{A}^\top\big(\sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top\big)^{-1}\big(\boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0\big). \end{aligned} \tag{6.44}$$

To find the term $\left(\nabla_{\boldsymbol{x}_t}\tilde{\boldsymbol{x}}_0\right)$ in Equation (6.44), we take a derivative w.r.t. $\boldsymbol{x}_t$ from both sides of Equation (6.14) to obtain

$$\nabla_{\boldsymbol{x}_t}\tilde{\boldsymbol{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}(t)}}\left(\mathbf{I} + (1-\bar{\alpha}(t))\nabla^2_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)\right) \tag{6.45}$$

$$\approx \frac{1}{\sqrt{\bar{\alpha}(t)}}\left(\mathbf{I} + (1-\bar{\alpha}(t))\tilde{\mathbf{H}}_t\right) \tag{6.46}$$

$$= \frac{\sqrt{\bar{\alpha}(t)}}{1-\bar{\alpha}(t)}\tilde{\boldsymbol{\Sigma}}_t. \tag{6.47}$$

Thus, using Equation (6.47) in Equation (6.44) we obtain

$$\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$$
$$\approx \frac{\sqrt{\bar{\alpha}(t)}}{1-\bar{\alpha}(t)}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top\left(\sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top\right)^{-1}\left(\boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0\right). \tag{6.48}$$

Note that in general, computing the inverse $\left(\sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top\right)^{-1}$ for high-dimensional images is computationally challenging. To this end, we use a low complexity method to find this inverse. In particular, we first define $\boldsymbol{\lambda} = \left(\sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top\right)^{-1}\left(\boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0\right)$. As the matrix $\sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top$ is is symmetric and positive-definite, then $\boldsymbol{\lambda}$ can be obtained by solving the following linear equation

$$\left(\sigma^2\mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t\mathbf{A}^\top\right)\boldsymbol{\lambda} = \boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0. \tag{6.49}$$

Therefore, $\boldsymbol{\lambda}$ can be computed with acceptable precision using a sufficient number of conjugate gradient (CG) iterates[1].

Once $\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$ is computed, the gradient of the posterior in Equation (6.7) can be calculated and used during the reverse denoising pass. This will slightly modify the unconditional DDPMs reverse pass as shown in Algorithm 3. Note that the only difference of CA-DPS algorithm with an unconditional sampling is in line 8 (written in blue), where the conditioning is applied.

[1]In the experiments, we utilize the black-box CG method implemented in 'scipy.sparse.linalg.cg', with a tolerance of tol = 1e−4.

Dataset	Method	Inpaint (Random)			Inpaint (Box)			Deblur (Gaussian)			Deblur (Motion)			SR (4×)		
		FID ↓	LPIPS ↓	SSIM ↑	FID ↓	LPIPS ↓	SSIM ↑	FID ↓	LPIPS ↓	SSIM ↑	FID ↓	LPIPS ↓	SSIM ↑	FID ↓	LPIPS ↓	SSIM ↑
FFHQ	DPS	21.19	0.212	0.851	33.12	0.168	**0.873**	44.05	0.257	0.811	39.92	0.242	0.859	39.35	0.214	0.852
	ΠGDM	21.27	0.221	0.840	34.79	0.179	0.860	40.21	0.242	0.825	33.24	0.221	0.887	34.98	0.202	0.854
	DDRM	69.71	0.587	0.319	42.93	0.204	0.869	74.92	0.332	0.767	–	–	–	62.15	0.294	0.835
	MCG	29.26	0.286	0.751	40.11	0.309	0.703	101.2	0.340	0.051	–	–	–	87.64	0.520	0.559
	PnP-ADMM	123.6	0.692	0.325	151.9	0.406	0.642	90.42	0.441	0.812	–	–	–	66.52	0.353	**0.865**
	Score-SDE	76.54	0.612	0.437	60.06	0.331	0.678	109.0	0.403	0.109	–	–	–	96.72	0.563	0.617
	ADMM-TV	181.5	0.463	0.784	68.94	0.322	0.814	186.7	0.507	0.801	–	–	–	110.6	0.428	0.803
	CA-DPS	**20.14**	**0.207**	**0.881**	**26.33**	**0.132**	0862	**32.74**	**0.238**	**0.832**	**27.59**	**0.217**	**0.921**	**28.41**	**0.196**	0.855
ImageNet	DPS	35.87	0.303	0.739	38.82	0.262	0.794	62.72	0.444	0.706	56.08	0.389	0.634	50.66	0.337	0.781
	ΠGDM	41.82	0.356	0.705	42.26	0.284	0.752	59.79	0.425	**0.717**	54.18	0.373	**0.675**	54.26	0.352	0.765
	DDRM	114.9	0.665	0.403	45.95	0.245	**0.814**	63.02	0.427	0.705	–	–	–	59.57	0.339	**0.790**
	MCG	39.19	0.414	0.546	39.74	0.330	0.633	95.04	0.550	0.441	–	–	–	144.5	0.637	0.227
	PnP-ADMM	114.7	0.677	0.300	78.24	0.367	0.657	100.6	0.519	0.669	–	–	–	97.27	0.433	0.761
	Score-SDE	127.1	0.659	0.517	54.07	0.354	0.612	120.3	0.667	0.436	–	–	–	170.7	0.701	0.256
	ADMM-TV	189.3	0.510	0.676	87.69	0.319	0.785	155.7	0.588	0.634	–	–	–	130.9	0.523	0.679
	CA-DPS	**32.37**	**0.214**	**0.755**	**33.24**	**0.247**	0.807	**56.36**	**0.391**	0.712	**52.06**	**0.352**	0.644	**47.30**	**0.316**	0.777

Table 6.1: **Quantitative results on the $1k$ validation set of FFHQ 256×256 and ImageNet 256×256 dataset.** We use **bold** and underline for the best and second best results, respectively.

Algorithm 3 CA-DPS

Input: The number of iterations N, $\boldsymbol{y}$, noise levels $\{\tilde{\sigma}\}$.
$\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = N-1, N-2, \ldots, 0$ **do**
  $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$
  $\tilde{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x}_t + (1-\bar{\alpha}_t)\hat{\boldsymbol{s}}\right)$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
  $\boldsymbol{x}'_{t-1} \leftarrow \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\tilde{\boldsymbol{x}}_0 + \tilde{\sigma}_t \boldsymbol{z}$.
  $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}'_{t-1} - \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\tilde{\boldsymbol{\Sigma}}_t \mathbf{A}^\top \left(\sigma^2 \mathbf{I} + \mathbf{A}\tilde{\boldsymbol{\Sigma}}_t \mathbf{A}^\top\right)^{-1}\left(\boldsymbol{y} - \mathbf{A}\tilde{\boldsymbol{x}}_0\right)$.
end for
Output: $\boldsymbol{x}_0$

6.5 Experiments

In this section, to demonstrate the superior performance of CA-DPS compared to state-of-the-art alternatives, we evaluate its effectiveness across a range of inverse problems—including inpainting, deblurring, and super-resolution—using two popular datasets. Specifically, we present quantitative and qualitative results in Section 6.5.1 and Section 6.5.2,

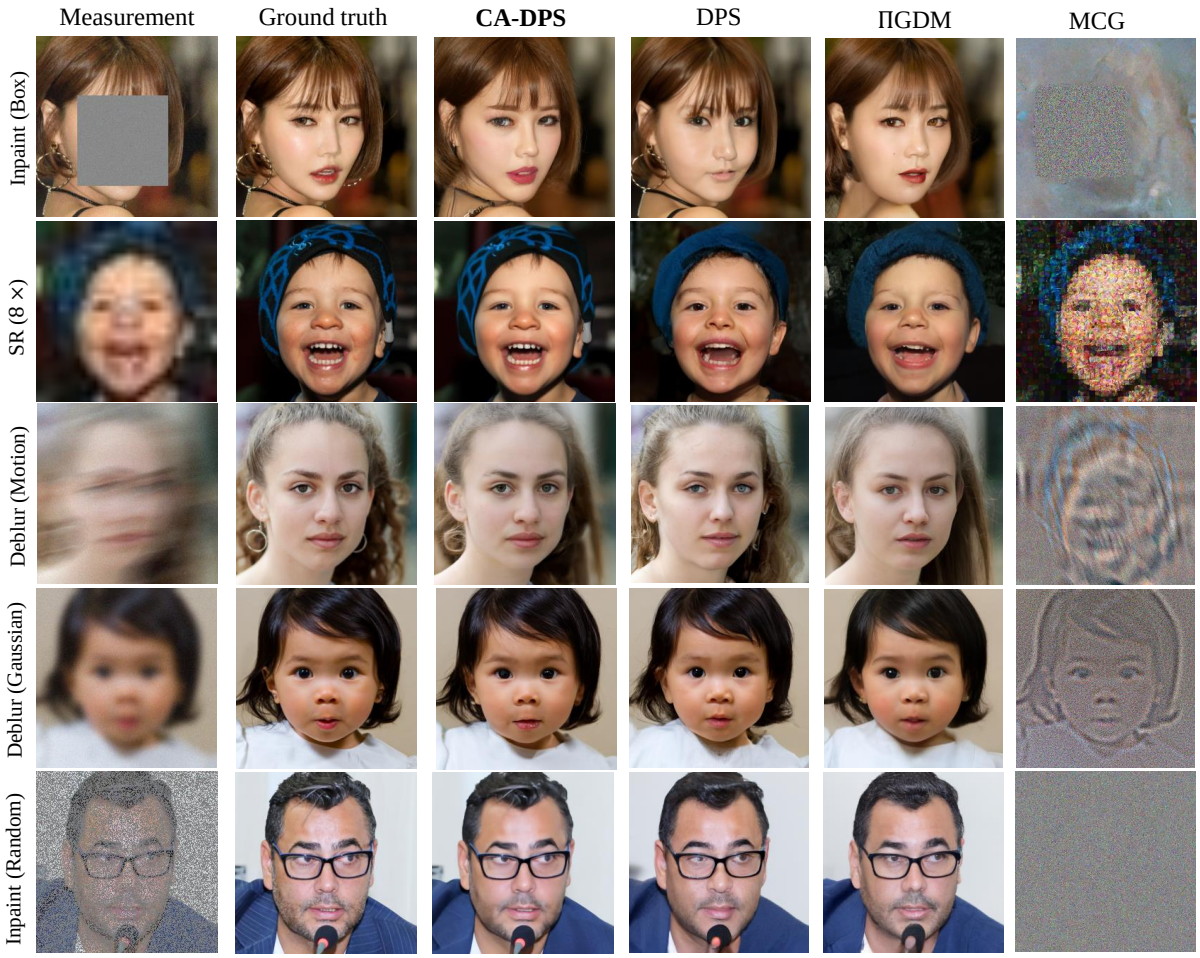


Figure 6.1: Qualitative results on FFHQ dataset.

respectively. To further illustrate that the superiority of CA-DPS stems from its improved approximation of the true posterior, we compare its ability to estimate the true posterior against benchmark methods in Section 6.5.3, using a toy dataset with a known posterior.

6.5.1 Quantitative Results.

• **Experimental setup.** Following [33, 52], we perform experiments on FFHQ 256×256 [108] and ImageNet 256×256 datasets [46], on 1k validation images each. All images are normalized to the range $[0, 1]$. For a fair comparison, we use the experimental settings

in [33] for all the methods. All measurements are corrupted by Gaussian noise with mean zero and $\sigma = 0.05$. For the backward process during the inference, we set the number of time steps as $N = 1000$ and use the pre-trained score model from [33] for the FFHQ dataset, and the score model from [47] for the ImageNet dataset.

The measurement models used are mostly based on [33]: (i) for box-type inpainting, we mask out a 128×128 box region, and for random-type inpainting, we mask out 92% of the total pixels (across all RGB channels); (ii) for super-resolution (SR), we perform bicubic downsampling; (iii) for Gaussian blur, we use a kernel size of 61×61 with a standard deviation of 3.0, and for motion blur, we use randomly generated kernels from the code[2], with a size of 61×61 and an intensity value of 0.5 (these kernels are then convolved with the ground truth image to produce the measurements).

• **Benchmark methods.** We compare the performance of CA-DPS with the following benchmark methods: DPS [33], ΠGDM [188], denoising diffusion restoration models (DDRM) [110], manifold constrained gradients (MCG) [34], Plug-and-play alternating direction method of multipliers (PnP-ADMM) [24] , Score-SDE [190] and total-variation (TV) sparsity regularized optimization method (ADMM-TV). For a fair comparison, we used the same score function for all the different methods that are based on diffusion (i.e. CA-DPS, DPS, DDRM, MCG, score-SDE).

• **Evaluation metrics.** To evaluate different methods, we follow [33] to use three metrics: (i) learned perceptual image patch similarity (LPIPS) [237], (ii) Frechet inception distance (FID) [**?**], and (iii) structure similarity index measure (SSIM). These metrics enable a comprehensive assessment of image quality. All our experiments are carried out on a single A100 GPU.

• **Experimental results.** The results for both datasets are listed in Table 6.1. The results demonstrate that CA-DPS outperforms baselines significantly in almost all the tasks. It is remarkable that in the challenging inpainting tasks (box and random), CA-DPS achieves the best performance. When assessing performance across the three metrics, CA-DPS emerges as the front-runner in three of them, with superior results compared to the other benchmark methods.

6.5.2 Qualitative Results

In this section, we aim to visualize the reconstructed images from CA-DPS and compare them with those reconstructed from the benchmark methods. To this end, we randomly

[2] https://github.com/LeviBorodenko/motionblur

select five images from the FFHQ test dataset and corrupt them using the measurement methods discussed in Section 6.5.1, with the exception that we apply 8× super-resolution for better visualization. The images are depicted in Figure 6.1, where each row corresponds to a different measurement method, and each column corresponds to a different benchmark method.

It is observed that the images reconstructed by CA-DPS show greater similarity to the ground truth images compared to those reconstructed by the benchmark methods. Specifically, for the super-resolution task, CA-DPS is the only method that successfully reconstructs the pattern of the hat. Additionally, in the motion-deblurring task, the eyes and other facial textures reconstructed by CA-DPS are much closer to the ground truth image.

6.5.3 Toy Dataset

In this subsection, we aim to illustrate that the superior results obtained by CA-DPS arise from its enhanced approximation of the true posterior. To support this, we shall compare its ability to estimate the true posterior against that of benchmark methods.

To this end, we generate a toy dataset whose distribution $p_0(\boldsymbol{x}_0)$ is a mixture of 25 Gaussian distributions[3]. The means and variances for each mixture component are detailed in the *Supplementary* materials, where we also explain how, for a given set of observations $\boldsymbol{y}$, measurement matrix $\mathbf{A}$, and noise standard deviation σ, the target posterior can be computed exactly.

To assess the effectiveness of posterior sampling methods, we generate multiple measurement models $(\boldsymbol{y}, \mathbf{A}) \in \mathbb{R}^m \times \mathbb{R}^{m \times d}$ for combinations of dimensions and observation noise levels $(d, m, \sigma) \in \{8, 80, 800\} \times \{1, 2, 4\} \times \{10^{-2}, 10^{-1}, 10^{0}\}$, while each Gaussian mixture component is equally weighted. By choosing different dimension sizes, we aim to understand how posterior sampling methods perform across varying dimensions, while controlling the noise level allows us to evaluate how these methods perform at different signal-to-noise ratios.

Next, we generate 1000 samples for each of the above scenarios ($3 \times 3 \times 3 = 27$ scenarios). Then, we use CA-DPS, ΠGDM and DPS to estimate the posterior probability through 1000 denoising steps. Afterward, to evaluate how well each algorithm estimates the posterior distribution compared to the target posterior, we utilize the sliced Wasserstein (SW) distance [116]. We calculate the SW distance using 10^4 slices for 1000 samples.

[3]We followed [16, 21] to generate this dataset.

	d	8	8	8	80	80	80	800	800	800
	m	1	2	4	1	2	4	1	2	4
$\sigma = 10^{-2}$	CA-DPS	**2.2**	**1.5**	**0.5**	**2.9**	**1.7**	**0.4**	**3.3**	**2.5**	**0.3**
	DPS	4.7	1.8	0.7	5.6	3.2	1.2	5.8	3.5	1.4
	ΠGDM	2.6	2.1	3.8	3.2	2.8	0.6	3.5	3.1	0.4
$\sigma = 10^{-1}$	CA-DPS	**1.8**	**0.9**	**0.6**	**2.5**	**1.7**	**0.4**	**2.8**	**2.3**	**0.4**
	DPS	4.7	1.5	0.8	5.1	3.1	1.0	5.7	3.1	1.3
	ΠGDM	2.2	1.6	3.8	2.9	2.7	0.6	3.3	2.7	**0.4**
$\sigma = 10^{0}$	CA-DPS	**1.2**	**1.9**	**0.9**	1.7	**1.2**	**0.8**	**1.6**	**1.5**	0.7
	DPS	5.2	3.5	2.5	6.9	3.9	1.7	6.8	4.7	0.9
	ΠGDM	1.5	2.3	1.8	**1.6**	1.4	0.9	2.0	2.0	**0.6**

Table 6.2: SW distance between the true and estimated posterior on toy dataset.

Table 6.2 shows the 95% confidence intervals, derived from 20 randomly selected measurement models ($\mathbf{A}$) for each parameter setting (d, m, σ). In addition, Figure 6.4 illustrates the first two dimensions of the estimated posterior distributions for the configuration (80, 1) from Table 6.2, using one randomly generated measurement model ($\mathbf{A}, \sigma = 0.1$). This visualization gives insights into how well the algorithms estimate the posterior distribution, showing that CA-DPS provides a more accurate estimate of the target posterior compared to ΠGDM and DPS, as it captures all modes, whereas ΠGDM and DPS do not.

6.5.4 Toy dataset

The generation of this dataset is inspired from [16].

As explained earlier in the chapter, we model $p_0(\boldsymbol{x}_0)$ as a mixture of 25 Gaussian distributions. Each of these Gaussian components has a mean vector $\mathbf{U}_{i,j}$ in $\mathbb{R}^d$, defined as $\mathbf{U}_{i,j} = (8i, 8j, \ldots, 8i, 8j)$ for each pair (i, j) where i and j take values from the set $\{-2, -1, 0, 1, 2\}$. All components have the same variance of 1. The unnormalized weight associated with each component is $\omega_{i,j} = 1.0$. Additionally, we have set the variance of the noise, σ_δ^2, to 10^{-4}.

Recall that the distribution $p_t(\boldsymbol{x}_t)$ can be expressed as an integral: $p_t(\boldsymbol{x}_t) = \int p_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0$. Since $p_0(\boldsymbol{x}_0)$ is a mixture of Gaussian distributions, $p_t(\boldsymbol{x}_t)$ is also a mixture of Gaussians. The means of these Gaussians are given by $\sqrt{\alpha_t}\mathbf{U}_{i,j}$, and

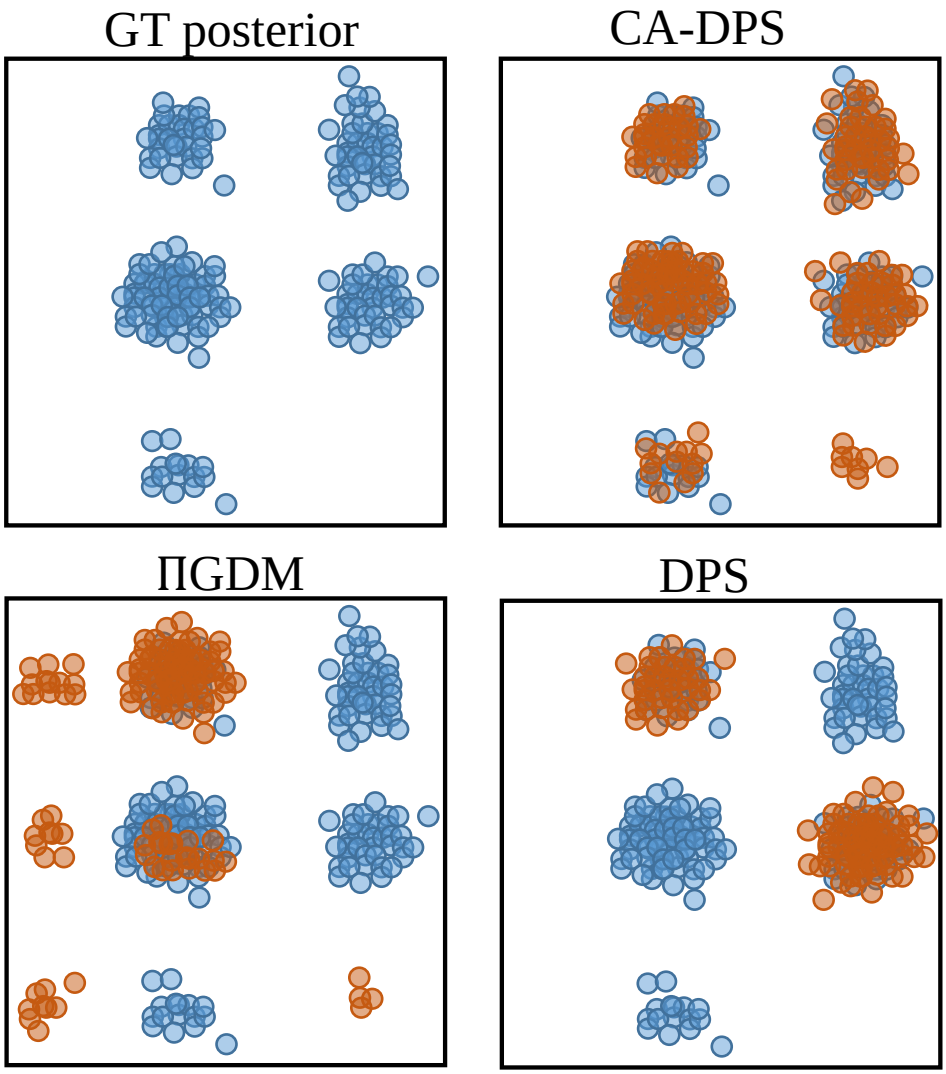


Figure 6.2: Visualizing the first two dimensions of the estimated posterior distributions for the configuration ($d = 80$, $m = 1$, $\sigma = 10^{-1}$) for a randomly generated $\mathbf{A}$.

each Gaussian has unit variance. By using automatic differentiation libraries, we can efficiently compute the gradient $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$.

We have set the parameters $\beta_{\max} = 500.0$ and $\beta_{\min} = 0.1$, and we use 1000 timesteps to discretize the time domain. For a given pair of dimensions and a chosen observation noise standard deviation (d, m, σ), the measurement model $(\boldsymbol{y}, \mathbf{A})$ is generated as follows:

• Matrix $\mathbf{A}$: First, we sample a random matrix $\tilde{\mathbf{A}}$ from a Gaussian distribution $\mathcal{N}(\mathbf{0}_{m\times d}, \mathbf{I}_{m\times d})$. We then compute its singular value decomposition (SVD), $\tilde{\mathbf{A}} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$. For each pair (i, j) in $\{-2, -1, 0, 1, 2\}^2$, we draw a singular value $s_{i,j}$ from a uniform distribution on the interval $[0, 1]$. Finally, we construct the matrix $\mathbf{A} = \mathbf{U}\mathrm{diag}(\{s_{i,j}\}_{(i,j)\in\{-2,-1,0,1,2\}^2})\mathbf{V}^\top$.

• Observation vector $\boldsymbol{y}$: Next, we sample a vector $\boldsymbol{x}_*$ from the distribution p_0. The observation vector $\boldsymbol{y}$ is then obtained by applying the matrix $\mathbf{A}$ to $\boldsymbol{x}_*$ and adding Gaussian noise $\boldsymbol{z}$, where $\boldsymbol{z}$ is sampled from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_m)$.

Once we have drawn both $\boldsymbol{x}_* \sim p_0$ and $(\boldsymbol{y}, \mathbf{A}, \sigma)$, the posterior can be exactly calculated using Bayes formula and gives a mixture of Gaussians with mixture components $c_{i,j}$ and associated weights $\tilde{\omega}_{i,j}$,

$$c_{i,j} := \mathcal{N}(\boldsymbol{\Sigma}(\mathbf{A}^\top\boldsymbol{y}/\sigma^2 + \mathbf{U}_{i,j}), \boldsymbol{\Sigma}), \tag{6.50}$$

$$\tilde{\omega}_i := \omega_i\mathcal{N}(\boldsymbol{y}; \mathbf{A}\mathbf{U}_{i,j}, \sigma_\delta^2\mathbf{I}_d + \mathbf{A}\mathbf{A}^\top), \tag{6.51}$$

where $\boldsymbol{\Sigma} = (\mathbf{I}_d + \frac{1}{\sigma_\delta^2}\mathbf{A}^\top\mathbf{A})^{-1}$.

6.5.5 SW Distance Calculation

To compare the posterior distribution estimated by each algorithm with the target posterior distribution, we use 10^4 slices for the SW distance and compare 1000 samples of the true posterior distribution.

Table 6.3 and Table 6.4 indicate the 95% confidence intervals obtained by considering 20 randomly selected measurement models ($\mathbf{A}$) for each setting (d, m, σ).

6.5.6 More Qualitative Results

In this section, we depict more reconstructed images using CA-DPS and compare it with those reconstructed by DPS. To this end, we pick 9 images from FFHQ dataset, and conduct super-resolution task (16×) with a Gaussian noise whose standard deviation is $\sigma = 0.05$. The results are depicted in Figure 6.4.

Furthermore, to visualize the reconstruction process over 1000 timesteps, we select a single image and display the reconstructed images throughout the denoising process, as illustrated in Figure 6.3.

Table 6.3: Sliced Wasserstein for VE-DDPM.

		$\sigma = 0.01$			$\sigma = 0.1$			$\sigma = 1.0$		
d	m	CA-DPS	ΠGDM	DPS	CA-DPS	ΠGDM	DPS	CA-DPS	ΠGDM	DPS
8	1	**1.9** ± **0.5**	2.6 ± 0.9	4.7 ± 1.5	**1.4** ± **0.6**	2.2 ± 0.9	4.7 ± 1.6	**1.2** ± **0.6**	1.5 ± 0.4	5.2 ± 1.3
8	2	**0.8** ± **0.4**	2.1 ± 1.0	1.8 ± 1.5	**1.0** ± **0.4**	1.6 ± 0.6	1.5 ± 0.9	**1.0** ± **0.3**	2.3 ± 0.4	3.5 ± 1.2
8	4	**0.4** ± **0.2**	3.8 ± 2.3	0.7 ± 0.6	**0.2** ± **0.2**	3.8 ± 2.2	0.8 ± 0.6	**0.7** ± **0.3**	1.8 ± 0.3	2.5 ± 0.9
80	1	**2.7** ± **0.7**	3.2 ± 1.0	5.6 ± 1.8	**2.4** ± **0.8**	2.9 ± 0.8	5.1 ± 1.8	**1.5** ± **0.7**	1.6 ± 0.5	6.9 ± 1.8
80	2	**1.1** ± **0.6**	2.8 ± 1.3	3.2 ± 1.9	**1.3** ± **0.4**	2.7 ± 1.2	3.1 ± 1.9	**1.0** ± **0.3**	1.4 ± 0.2	3.9 ± 1.2
80	4	**0.4** ± **0.2**	0.6 ± 0.4	1.2 ± 1.1	**0.5** ± **0.3**	0.6 ± 0.4	1.0 ± 1.1	**0.9** ± **0.3**	0.9 ± 0.2	1.7 ± 0.6
800	1	**3.1** ± **0.7**	3.5 ± 1.1	5.8 ± 1.6	**3.0** ± **0.5**	3.3 ± 0.9	5.7 ± 1.6	**1.4** ± **0.5**	2.0 ± 0.4	6.8 ± 1.0
800	2	**1.5** ± **0.5**	3.1 ± 1.1	3.5 ± 1.7	**1.2** ± **0.4**	2.7 ± 0.9	3.1 ± 1.4	**1.3** ± **0.4**	2.0 ± 0.5	4.7 ± 1.3
800	4	0.5 ± 0.3	**0.4** ± **0.2**	1.4 ± 1.0	**0.3** ± **0.2**	0.4 ± 0.2	1.3 ± 0.9	0.9 ± 0.2	**0.7** ± **0.3**	0.9 ± 0.4

6.6 Chapter Summary

In this chapter, we proposed CA-DPS, a method designed to enhance the performance of DDPMs in solving inverse problems. To achieve this, we derived a closed-form expression for the covariance of reverse process in DDPMs. We then proposed a method based on finite differences to approximate this covariance, making it easily obtainable from existing pre-trained DDPMs. Utilizing the mean and the approximated covariance of the reverse process, we present a new approximation for the likelihood. Finally, we conducted three sets of experiments to demonstrate the superiority of CA-DPS: (i) quantitative evaluations using various metrics to assess the quality of reconstructed images, (ii) qualitative assessments by visualizing some of the reconstructed images, and (iii) testing the proximity of estimated posterior to the true posterior using a toy dataset with a known posterior.

Table 6.4: Sliced Wasserstein for the GMM case for the reverse VE SDEs discretized with Euler-Maruyama.

		$\sigma = 0.01$			$\sigma = 0.1$			$\sigma = 1.0$		
d	m	CA-DPS	ΠGDM	DPS	CA-DPS	ΠGDM	DPS	CA-DPS	ΠGDM	DPS
8	1	**1.6** ± 0.4	1.5 ± 0.4	5.7 ± 2.2	**1.3** ± 0.4	1.2 ± 0.4	5.6 ± 2.1	**0.8** ± 0.3	0.9 ± 0.3	0.9 ± 0.3
8	2	**0.6** ± 0.3	0.4 ± 0.3	6.2 ± 0.8	**1.0** ± 0.4	0.5 ± 0.3	6.2 ± 2.4	**0.8** ± 0.2	1.0 ± 0.3	1.2 ± 0.4
8	4	**0.4** ± 0.2	0.1 ± 0.1	-	**0.4** ± 0.2	0.1 ± 0.0	8.4 ± 3.1	**0.7** ± 0.2	0.2 ± 0.1	0.3 ± 0.2
80	1	**2.5** ± 0.7	2.9 ± 1.4	9.1 ± 1.3	**2.1** ± 0.8	2.1 ± 1.1	4.7 ± 1.8	**1.4** ± 0.7	1.8 ± 0.8	1.9 ± 0.9
80	2	**1.2** ± 0.4	0.8 ± 0.7	2.2 ± 0.9	**1.1** ± 0.5	0.8 ± 0.7	6.0 ± 2.1	**1.3** ± 0.3	1.3 ± 0.5	1.5 ± 0.5
80	4	**0.4** ± 0.1	0.1 ± 0.0	-	**0.3** ± 0.2	0.1 ± 0.1	4.4 ± 1.6	**0.8** ± 0.3	0.4 ± 0.2	0.5 ± 0.3
800	1	**3.2** ± 0.6	3.2 ± 1.0	6.8 ± 1.2	**2.8** ± 0.5	2.8 ± 0.7	6.4 ± 1.5	**1.4** ± 0.4	1.3 ± 0.3	1.3 ± 0.3
800	2	**1.4** ± 0.3	0.8 ± 0.5	7.4 ± 0.9	**1.2** ± 0.3	0.8 ± 0.4	6.4 ± 1.9	**1.3** ± 0.4	1.1 ± 0.3	1.1 ± 0.3
800	4	**0.4** ± 0.2	0.6 ± 0.5	-	**0.3** ± 0.2	0.1 ± 0.0	5.8 ± 1.4	**0.8** ± 0.3	0.4 ± 0.2	0.4 ± 0.2

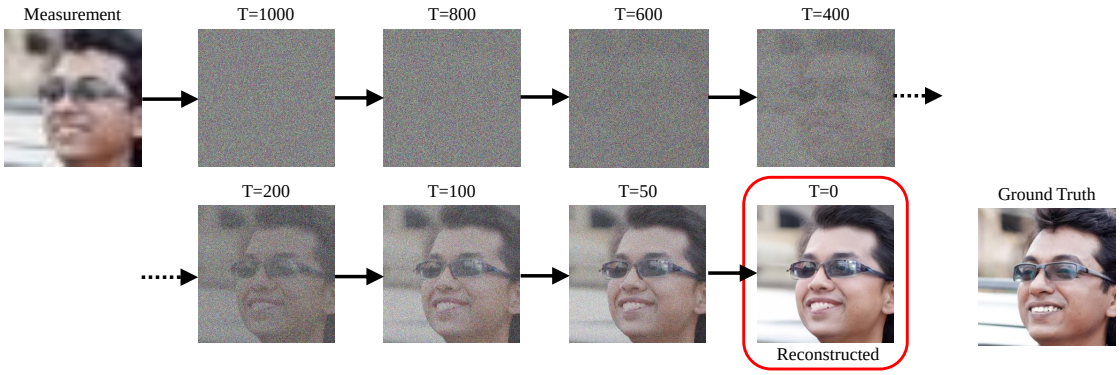


Figure 6.3: Qualitative results on FFHQ dataset.

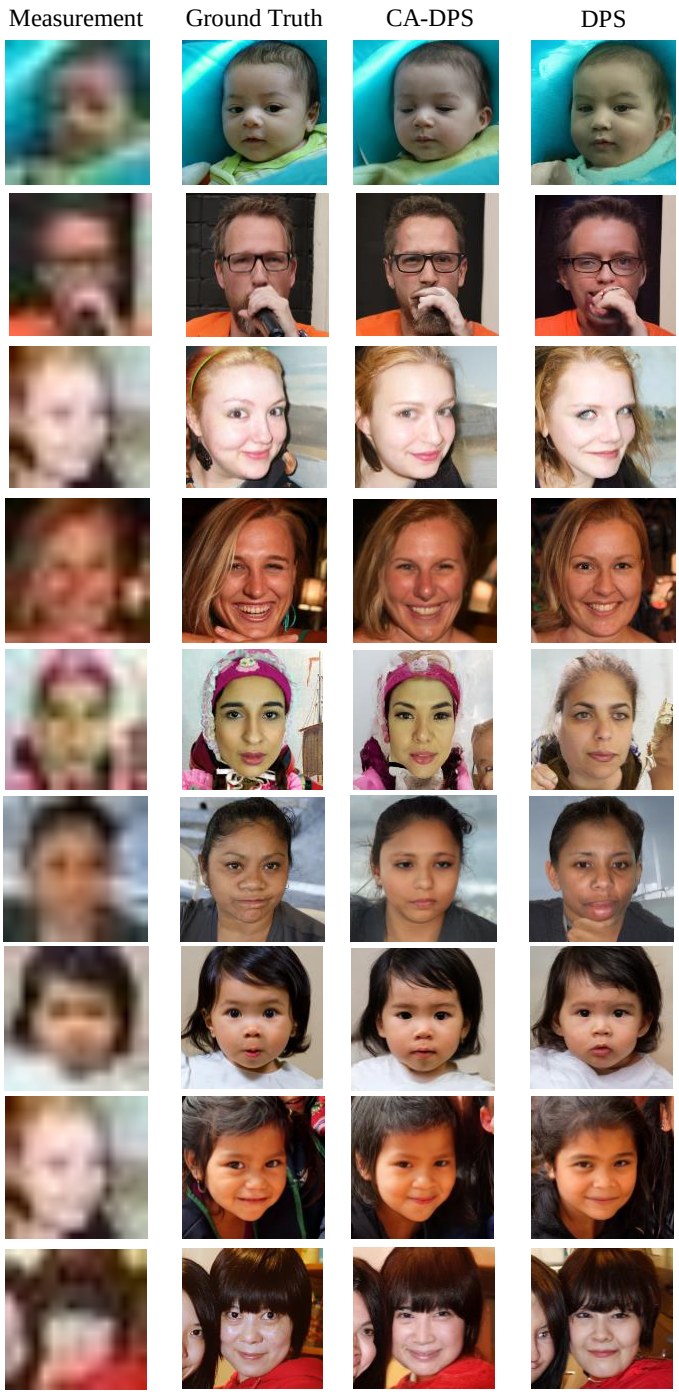


Figure 6.4: Qualitative results on FFHQ dataset.

Chapter 7

Conclusions and future work

7.1 Summary

The primary objective of this dissertation was to integrate ideas from information theory into the core mechanisms of machine learning, with the goal of improving the performance of deep neural networks (DNNs). The work is structured into two main parts: the first focuses on classification DNNs, while the second part shifts attention to diffusion DNNs, a type of generative model. By addressing both discriminative and generative tasks, this thesis provides a comprehensive exploration of how information-theoretic and probabilistic principles can enhance the overall capabilities of modern DNNs.

In the first part of the dissertation, we concentrated on advancing the performance of classification DNNs from multiple angles: (i) improving classification accuracy in both centralized and distributed learning environments, (ii) reducing the size and complexity of DNN models to make them more efficient, and (iii) minimizing the computational overhead associated with both training and inference. By leveraging information-theoretic principles, we were able to introduce novel techniques that not only boost accuracy but also address the pressing challenges of model efficiency and scalability. These improvements are crucial for deploying DNNs in resource-constrained settings, such as edge devices or large-scale distributed systems.

In the second part of the dissertation, we extended our focus to generative DNNs, specifically diffusion models, which have emerged as a powerful tool for generating data and solving complex inverse problems. Here, we explored the potential of applying probabilistic frameworks to enhance the performance of diffusion models in inverse problem

settings. Inverse problems, which involve inferring underlying data from incomplete or noisy observations, are critical in many domains, such as medical imaging, physics-based simulations, and signal processing. By incorporating probabilistic reasoning into the diffusion model framework, we demonstrate how these models can achieve superior performance in accurately reconstructing latent variables, thereby broadening their applicability and effectiveness in real-world tasks.

Throughout both parts of the dissertation, we validate our theoretical findings with extensive experimental results. These experiments consistently demonstrate that the proposed methods outperform state-of-the-art alternatives across various benchmarks. The improvements observed in accuracy, efficiency, and robustness highlight the effectiveness of integrating information-theoretic and probabilistic principles into machine learning, paving the way for more advanced DNN architectures and applications.

7.2 Contributions

The research presented in this dissertation made the following main contributions:

- We introduced a novel metric, called normalized conditional mutual information (NCMI), to assess the performance of classification deep neural networks (DNNs). Leveraging NCMI, we then modified the traditional training loss function used in standard deep learning approaches. Through extensive experimentation, we demonstrate that DNNs trained with our method consistently achieve higher accuracy compared to conventional counterparts, proving the efficacy of our NCMI-based approach in improving classification performance.

- We introduced a novel federated learning mechanism, inspired by information-theoretic principles, which we refer to as Fed-IT. This approach specifically addresses the class-imbalance problem, a common challenge in federated learning, by leveraging insights from information theory to balance the distribution of information across decentralized clients. By approaching the class imbalance from an information-theoretic perspective, we ensure that the shared global model receives a more balanced and representative input from each client. This leads to improved learning dynamics and significantly enhances the classification accuracy of the shared model, demonstrating the effectiveness of Fed-IT in overcoming the limitations imposed by uneven data distribution in federated learning environments.

- We used information-theoretic concepts into the inner-workings of DL, and introduced a new framework for training DNNs which we refer to as coded deep learning (CDL). CDL aims to substantially compress model weights and activations, reduce computational complexity at both training and post-training inference stages, and enable efficient model/data parallelism. Specifically, within CDL, (i) we first propose a novel probabilistic method for quantizing both model weights and activations, and its soft differentiable variant which offers an analytic formula for gradient calculation during training; (ii) both the forward and backward passes during training are executed over quantized weights and activations, which eliminates a majority of floating-point operations and reduces the training computation complexity; (iii) during training, both weights and activations are entropy constrained so that they are compressible in an information-theoretic sense at any stage of training, which in turn reduces communication costs in cases where model/data parallelism is adopted; and (iv) the trained model in CDL is by default in a quantized format with compressible quantized weights, reducing post-training inference complexity and model storage complexity.

- We leverage probabilistic methods to enhance the performance of diffusion models in solving inverse problems. By incorporating probabilistic knowledge into the diffusion process, we refine the model's ability to recover underlying signals from noisy or incomplete data. This approach improves the model's capability to accurately infer latent variables and reconstruct original inputs in inverse problem scenarios, such as medical imaging, physics-based simulations, and signal reconstruction.

7.3 Directions for Future Work

Further research on NCMI and CDL, may include the topics listed below:

- In the current version of the conditional mutual information-constrained framework, we define NCMI within the probability space of DNNs. A potential improvement to this approach could involve redefining NCMI in the feature space of DNNs.

- The concept of NCMI could be expanded to *robust NCMI* in order to develop robust DNNs that are resilient to adversarial attacks.

- By treating the probabilistic quantization method used in CDL as a non-linear function, we observed an improvement in the classification accuracy of DNNs. This

insight suggests that the benefits of probabilistic quantization can extend beyond compression, hinting at a new direction for enhancing model performance. Specifically, one could modify the activation functions of DNNs in a similar probabilistic manner to further boost classification accuracy. By designing activation functions that incorporate probabilistic elements, akin to how quantization operates in CDL, we can potentially introduce more flexible and adaptive non-linear activation functions, leading to better feature learning and overall model performance.

References

[1] The full paper is under preparation, this is the link to the project repository: https://github.com/camerontan0527/fed-it.

[2] Jose M Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. *Advances in neural information processing systems*, 30, 2017.

[3] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[5] Caglar Aytekin, Francesco Cricri, and Emre Aksu. Compressibility loss for neural network weights. *arXiv preprint arXiv:1905.01044*, 2019.

[6] Harrison H Barrett and Kyle J Myers. *Foundations of image science*. John Wiley & Sons, 2013.

[7] Chaim Baskin, Brian Chmiel, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Cat: Compression-aware training for bandwidth reduction. *Journal of Machine Learning Research*, 22(269):1–20, 2021.

[8] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11):2419–2434, 2009.

[9] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[10] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

[11] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[12] Toby Berger. *Rate distortion theory: A mathematical basis for data compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[13] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[14] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018.

[15] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR, 2017.

[16] Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and O Deniz Akyildiz. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

[17] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[19] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

[21] Gabriel Cardoso, Yazid Janati El Idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided diffusion for bayesian linear inverse problems. *arXiv preprint arXiv:2308.07983*, 2023.

[22] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[23] Sek M. Chai. Quantization-guided training for compact tiny{ml} models. In *Research Symposium on Tiny Machine Learning*, 2021.

[24] Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2016.

[25] Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021.

[26] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. In *Neural Information Processing Systems (NeuRIPS)*, 2019.

[27] Brian Chmiel, Ron Banner, Gil Shomron, Yury Nahshan, Alex Bronstein, Uri Weiser, et al. Robust quantization: One model to rule them all. *Advances in neural information processing systems*, 33:5308–5317, 2020.

[28] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning method for denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[29] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, I Pierce, Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. 2018.

[30] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[31] Yen-Hsiu Chou, Shenda Hong, Chenxi Sun, Derun Cai, Moxian Song, and Hongyan Li. Grp-fed: Addressing client imbalance in federated learning via global-regularized personalization. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 451–458. SIAM, 2022.

[32] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019.

[33] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[34] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *arXiv preprint arXiv:2206.00941*, 2022.

[35] Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[36] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

[37] Regev Cohen, Michael Elad, and Peyman Milanfar. Regularization by denoising via fixed-point projection (red-pro). *SIAM Journal on Imaging Sciences*, 14(3):1374–1406, 2021.

[38] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[39] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32, 2019.

[40] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[41] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1153–1162, 2016.

[42] Yutong Dai, Zeyuan Chen, Junnan Li, Shelby Heinecke, Lichao Sun, and Ran Xu. Tackling data heterogeneity in federated learning with class prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7314–7322, 2023.

[43] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[44] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[45] Mauricio Delbracio and Peyman Milanfar. Inversion by direct iteration: An alternative to denoising diffusion for image restoration. *Transactions on Machine Learning Research*, 2023. Featured Certification.

[46] 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.

[47] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[48] Fei Ding, Yin Yang, Hongxin Hu, Venkat Krovi, and Feng Luo. Dual-level knowledge distillation via knowledge alignment and correlation. *IEEE Trans. Neural Netw. Learn. Syst.*, 35(2):2425–2435, 2024.

[49] Senapati Sang Diwangkara and Achmad Imam Kistijantoro. Study of data imbalance and asynchronous aggregation algorithm on federated learning system. In *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 276–281. IEEE, 2020.

[50] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[51] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[52] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[53] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):59–71, 2020.

[54] Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In *2019 IEEE 37th international conference on computer design (ICCD)*, pages 246–254. IEEE, 2019.

[55] Yueqi Duan, Jiwen Lu, and Jie Zhou. Uniformface: Learning deep equidistributed representation for face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3415–3424, 2019.

[56] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[57] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2019.

[58] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020.

[59] Berthy T Feng and Katherine L Bouman. Efficient bayesian computational imaging with a surrogate score-based prior. *arXiv preprint arXiv:2309.01949*, 2023.

[60] Mário AT Figueiredo and Robert D Nowak. An em algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

[61] Marylou Gabrié, Andre Manoel, Clément Luneau, Nicolas Macris, Florent Krzakala, Lenka Zdeborová, et al. Entropy and mutual information in models of deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[62] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[63] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit dnns. *International Conference on Computer Vision (ICCV)*, 2019.

[64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[65] 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.

[66] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[67] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning, (ICML)*, 2015.

[68] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, 73:220–239, 2017.

[69] Elaine T Hale, Wotao Yin, and Yin Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. *CAAM TR07-07, Rice University*, 43(44):2, 2007.

[70] Shayan Mohajer Hamidi. Training neural networks on remote edge devices for unseen class classification. *IEEE Signal Processing Letters*, 31:1004–1008, 2024.

[71] Shayan Mohajer Hamidi and Ali Bereyhi. Rate-constrained quantization for communication-efficient federated learning, 2024.

[72] Shayan Mohajer Hamidi, Ali Bereyhi, Saba Asaad, and H. Vincent Poor. Gp-fl: Model-based hessian estimation for second-order over-the-air federated learning, 2024.

[73] Shayan Mohajer Hamidi, Ali Bereyhi, Saba Asaad, and H. Vincent Poor. Over-the-air fair federated learning via multi-objective optimization, 2025.

[74] Shayan Mohajer Hamidi, Xizhen Deng, Renhao Tan, Linfeng Ye, and Ahmed Hussein Salamah. How to train the teacher model for effective knowledge distillation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2025.

[75] Shayan Mohajer Hamidi, Mohammad Mehrabi, Amir K. Khandani, and Deniz Gündüz. Over-the-air federated learning exploiting channel perturbation. In *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, pages 1–5, 2022.

[76] Shayan Mohajer Hamidi, Renhao Tan, Linfeng Ye, and En-Hui Yang. Fed-it: Addressing class imbalance in federated learning through an information- theoretic lens. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pages 1848–1853, 2024.

[77] Shayan Mohajer Hamidi and EN-HUI YANG. Adafed: Fair federated learning via adaptive common descent direction. *Transactions on Machine Learning Research*, 2024.

[78] Shayan Mohajer Hamidi and En-Hui Yang. Enhancing diffusion models for inverse problems with covariance-aware posterior sampling. *arXiv preprint arXiv:2412.20045*, 2024.

[79] Shayan Mohajer Hamidi and En-Hui Yang. Conditional mutual information based diffusion posterior sampling for solving inverse problems. *arXiv preprint arXiv:2501.02880*, 2025.

[80] Shayan Mohajer Hamidi and Linfeng Ye. Distributed quasi-newton method for fair and fast federated learning. 2024.

[81] Tiantian Han, Dong Li, Ji Liu, Lu Tian, and Yi Shan. Improving low-precision network quantization via bin regularization. In *International Conference on Computer Vision (ICCV)*, 2021.

[82] Tiantian Han, Dong Li, Ji Liu, Lu Tian, and Yi Shan. Improving low-precision network quantization via bin regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5261–5270, 2021.

[83] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[84] 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.

[85] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[86] Geoffrey Hinton. Neural networks for machine learning, lectures 15b. 2012.

[87] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[88] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[89] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[90] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

[91] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[92] Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu. Federated learning meets multi-objective optimization. *IEEE Transactions on Network Science and Engineering*, 2022.

[93] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.

[94] 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.

[95] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[96] En hui Yang and Shayan Mohajer Hamidi. Coded deep learning: Framework and algorithm, 2025.

[97] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[98] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[99] B. Ham J. Lee, D. Kim. Network quantization with element-wise gradient scaling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[100] Sambhav R. Jain, Albert Gural, Michael Wu, and Chris Dick. Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware. *arxiv preprint arxiv:1903.08066*, 2019.

[101] Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust compressed sensing mri with deep generative priors. *Advances in Neural Information Processing Systems*, 34:14938–14954, 2021.

[102] Ajil Jalal, Sushrut Karmalkar, Jessica Hoffmann, Alex Dimakis, and Eric Price. Fairness for image generation with uncertain sensitive attributes. In *International Conference on Machine Learning*, pages 4721–4732. PMLR, 2021.

[103] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[104] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[105] Zahra Kadkhodaie and Eero Simoncelli. Stochastic solutions for linear inverse problems using the prior implicit in a denoiser. *Advances in Neural Information Processing Systems*, 34:13242–13254, 2021.

[106] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[107] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[108] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[109] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM computing surveys (CSUR)*, 52(4):1–36, 2019.

[110] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

[111] Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in Neural Information Processing Systems*, 34:21757–21769, 2021.

[112] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[113] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.

[114] Jangho Kim, KiYoon Yoo, and Nojun Kwak. Position-based scaled gradient for model quantization and pruning. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2020.

[115] Maciej Koch-Janusz and Zohar Ringel. Mutual information, neural networks and the renormalization group. *Nature Physics*, 14(6):578–582, 2018.

[116] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. *Advances in neural information processing systems*, 32, 2019.

[117] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*.

[118] Atli Kosson, Vitaliy Chiley, Abhinav Venigalla, Joel Hestness, and Urs Koster. Pipelined backpropagation at scale: training large models without batches. *Proceedings of Machine Learning and Systems*, 3:479–501, 2021.

[119] Eli Kravchik, Fan Yang, Pavel Kisilev, and Yoni Choukroun. Low-bit quantization of neural networks for efficient inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[120] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[121] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[122] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[123] Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, page 179. Citeseer, 1997.

[124] Arnav Kundu, Chungkuk Yoo, Srijan Mishra, Minsik Cho, and Saurabh Adya. r^2: Range regularization for model compression and quantization, 2024.

[125] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8183–8192, 2018.

[126] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[127] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[128] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[129] Hansang Lee, Minseok Park, and Junmo Kim. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In *2016 IEEE international conference on image processing (ICIP)*, pages 3713–3717. IEEE, 2016.

[130] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[131] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2019.

[132] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019.

[133] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[134] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[135] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704, 2018.

[136] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. *arXiv preprint arXiv:1612.02295*, 2016.

[137] Yu Liu, Hongyang Li, and Xiaogang Wang. Rethinking feature discrimination and polymerization for large-scale recognition. *arXiv preprint arXiv:1710.00870*, 2017.

[138] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[139] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2018.

[140] Cheng Lu, Kaiwen Zheng, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Maximum likelihood training for score-based diffusion odes by high order denoising score matching. In *International Conference on Machine Learning*, pages 14429–14460. PMLR, 2022.

[141] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

[142] Xiaolong Ma, Sheng Lin, Shaokai Ye, Zhezhi He, Linfeng Zhang, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, Kaisheng Ma, and Yanzhi Wang. Non-structured dnn weight pruning—is it beneficial in any platform? *IEEE Trans. Neural Netw. Learn. Syst.*, 33(9):4930–4944, 2022.

[143] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[144] Morteza Mardani, Jiaming Song, Jan Kautz, and Arash Vahdat. A variational perspective on solving inverse problems with diffusion models. *arXiv preprint arXiv:2305.04391*, 2023.

[145] Kanti V Mardia and Peter E Jupp. *Directional statistics*. John Wiley & Sons, 2009.

[146] Dastan Maulud and Adnan M Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1(2):140–147, 2020.

[147] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[148] Chenlin Meng, Yang Song, Wenzhe Li, and Stefano Ermon. Estimating high order gradients of the data distribution by denoising. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[149] 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.

[150] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[151] Shayan Mohajer Hamidi and Oussama Damen. Fair wireless federated learning through the identification of a common descent direction. *IEEE Communications Letters*, 28(3):567–571, 2024.

[152] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[153] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019.

[154] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[155] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pages 16318–16330. PMLR, 2022.

[156] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro,

et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

[157] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[158] 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.

[159] Jaehoon Oh, SangMook Kim, and Se-Young Yun. Fedbabu: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations*, 2021.

[160] Gregory Ongie, Ajil Jalal, Christopher A Metzler, Richard G Baraniuk, Alexandros G Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[161] Michel Owayjan, Amer Dergham, Gerges Haber, Nidal Fakih, Ahmad Hamoush, and Elie Abdo. Face recognition security system. In *New trends in networking, computing, E-learning, systems sciences, and engineering*, pages 343–348. Springer, 2015.

[162] Edouard Oyallon. Building a regular decision boundary with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5106–5114, 2017.

[163] Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *The Journal of Machine Learning Research*, 21(1):10197–10260, 2020.

[164] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[165] Eunhyeok Park and Sungjoo Yoo. PROFIT: A novel training method for sub-4-bit mobilenet models. In *European Conference on Computer Vision (ECCV)*, 2020.

[166] Hanyu Peng, Jiaxiang Wu, Zhiwei Zhang, Shifeng Chen, and Hai-Tao Zhang. Deep network quantization via error compensation. *IEEE Trans. Neural Netw. Learn. Syst.*, 33(9):4960–4970, 2022.

[167] Xinyu Peng, Ziyang Zheng, Wenrui Dai, Nuoqian Xiao, Chenglin Li, Junni Zou, and Hongkai Xiong. Improving diffusion models for inverse problems using optimal posterior covariance. In *Forty-first International Conference on Machine Learning*, 2024.

[168] Samira Pouyanfar, Yudong Tao, Anup Mohan, Haiman Tian, Ahmed S Kaseb, Kent Gauen, Ryan Dailey, Sarah Aghajanzadeh, Yung-Hsiang Lu, Shu-Ching Chen, et al. Dynamic sampling in convolutional neural networks for imbalanced data classification. In *2018 IEEE conference on multimedia information processing and retrieval (MIPR)*, pages 112–117. IEEE, 2018.

[169] Ce Qi and Fei Su. Contrastive-center loss for deep neural networks. In *2017 IEEE international conference on image processing (ICIP)*, pages 2851–2855. IEEE, 2017.

[170] Zheng Qin, Zeming Li, Zhaoning Zhang, Yiping Bao, Gang Yu, Yuxing Peng, and Jian Sun. Thundernet: Towards real-time generic object detection on mobile devices. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6718–6727, 2019.

[171] Kanchana Ranasinghe, Muzammal Naseer, Munawar Hayat, Salman Khan, and Fahad Shahbaz Khan. Orthogonal projection loss. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12333–12343, 2021.

[172] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.

[173] Alejandro Ribes and Francis Schmitt. Linear inverse problems in imaging. *IEEE Signal Processing Magazine*, 25(4):84–99, 2008.

[174] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, 14(6):1–50, 2023.

[175] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[176] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim

Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[177] 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*, 158:110966, 2025.

[178] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[179] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[180] Xinyi Shang, Yang Lu, Gang Huang, and Hanzi Wang. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. *arXiv preprint arXiv:2204.13399*, 2022.

[181] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 764–775. IEEE, 2018.

[182] Zebang Shen, Juan Cervino, Hamed Hassani, and Alejandro Ribeiro. An agnostic approach to federated learning with class imbalance. In *International Conference on Learning Representations*, 2021.

[183] Zhanglei Shi, Hao Wang, and Chi-Sing Leung. Constrained center loss for convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 34(2):1080–1088, 2023.

[184] Nir Shlezinger, Jay Whang, Yonina C Eldar, and Alexandros G Dimakis. Model-based deep learning. *Proceedings of the IEEE*, 111(5):465–499, 2023.

[185] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[186] Pramote Sittijuk and Kriengsuk Tamee. Performance measurement of federated learning on imbalanced data. In *2021 18th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6. IEEE, 2021.

[187] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR*, 2021.

[188] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

[189] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2022.

[190] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR*, 2021.

[191] Suhas Sreehari, S Venkat Venkatakrishnan, Brendt Wohlberg, Gregery T Buzzard, Lawrence F Drummy, Jeffrey P Simmons, and Charles A Bouman. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Transactions on Computational Imaging*, 2(4):408–423, 2016.

[192] Tristan SW Stevens, Hans van Gorp, Faik C Meral, Junseob Shin, Jason Yu, Jean-Luc Robert, and Ruud JG van Sloun. Removing structured noise with diffusion models. *arXiv preprint arXiv:2302.05290*, 2023.

[193] Peng Sun, Yonggang Wen, Ruobing Han, Wansen Feng, and Shengen Yan. Gradientflow: Optimizing network performance for large-scale distributed dnn training. *IEEE Transactions on Big Data*, 8(2):495–507, 2019.

[194] Chao Tan and Jie Liu. Improving knowledge distillation with a customized teacher. *IEEE Trans. Neural Netw. Learn. Syst.*, 35(2):2290–2299, 2024.

[195] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[196] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance. In *European Conference on Computer Vision*, pages 259–275. Springer, 2022.

[197] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[198] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[199] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. In *International Conference on Learning Representations (ICLR)*, 2020.

[200] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948. IEEE, 2013.

[201] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[202] Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9117–9126, 2018.

[203] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049, 2017.

[204] Hao Wang, Jinpeng Cao, Zhang-Lei Shi, Chi-Sing Leung, Ruibin Feng, Wenming Cao, and Yuxin He. Image classification on hypersphere loss. *IEEE Transactions on Industrial Informatics*, 2024.

[205] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[206] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10165–10173, 2021.

[207] Weinong Wang, Wenjie Pei, Qiong Cao, Shu Liu, Guangming Lu, and Yu-Wing Tai. Push for center learning via orthogonalization and subspace masking for person re-identification. *IEEE Transactions on Image Processing*, 30:907–920, 2020.

[208] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. *Advances in neural information processing systems*, 30, 2017.

[209] Yulin Wang, Gao Huang, Shiji Song, Xuran Pan, Yitong Xia, and Cheng Wu. Regularizing deep networks with semantic data augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3733–3748, 2022.

[210] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[211] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *Computer vision–ECCV 2016: 14th European conference, amsterdam, the netherlands, October 11–14, 2016, proceedings, part VII 14*, pages 499–515. Springer, 2016.

[212] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A comprehensive study on center loss for deep face recognition. *International Journal of Computer Vision*, 127:668–683, 2019.

[213] Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012.

[214] Mingyang Xia, Guoshan Zhang, Chaoxu Mu, Bin Guan, and Mengxuan Wang. Cervical cancer cell detection based on deep convolutional neural network. In *2020 39th Chinese Control Conference (CCC)*, pages 6527–6532. IEEE, 2020.

[215] Chenguang Xiao and Shuo Wang. An experimental study of class imbalance in federated learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2021.

[216] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[217] Eric P Xing, Qirong Ho, Wei Dai, Jin-Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. In *Proceedings of the 21th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1335–1344, 2015.

[218] Kohei Yamamoto. Learnable companding quantization for accurate low-bit neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5029–5038, 2021.

[219] Bowen Yang, Jian Zhang, Jonathan Li, Christopher Ré, Christopher Aberger, and Christopher De Sa. Pipemare: Asynchronous pipeline parallel dnn training. *Proceedings of Machine Learning and Systems*, 3:269–296, 2021.

[220] En-hui Yang and Shayan Mohajer Hamidi. Systems and methods for training deep learning models, November 21 2024. US Patent App. 18/666,958.

[221] En-Hui Yang, Shayan Mohajer Hamidi, Linfeng Ye, Renhao Tan, and Beverly Yang. Conditional mutual information constrained deep learning for classification. *arXiv preprint arXiv:2309.09123*, 2023.

[222] En-Hui Yang, Shayan Mohajer Hamidi, Linfeng Ye, Renhao Tan, and Beverly Yang. Conditional mutual information constrained deep learning: Framework and preliminary results. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pages 569–574, 2024.

[223] En-Hui Yang, Zhen Zhang, and Toby Berger. Fixed-slope universal lossy data compression. *IEEE Trans. Inf. Theory*, 43(5):1465–1476, 1997.

[224] Hong-Ming Yang, Xu-Yao Zhang, Fei Yin, Qing Yang, and Cheng-Lin Liu. Convolutional prototype network for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2358–2370, 2020.

[225] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[226] Miao Yang, Ximin Wang, Hongbin Zhu, Haifeng Wang, and Hua Qian. Federated learning with class imbalance reduction. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 2174–2178. IEEE, 2021.

[227] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[228] 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.

[229] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations(ICLR)*, 2019.

[230] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[231] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *British Machine Vision Conference 2016*, York, France, January 2016. British Machine Vision Association.

[232] John Zarka, Florentin Guth, and Stéphane Mallat. Separation and concentration in deep networks. In *International Conference on Learning Representations*, 2020.

[233] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[234] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion models in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.

[235] Dongqing Zhang, J. Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of European conference on computer vision (ECCV)*, pages 365–382, 2018.

[236] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.

[237] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[238] S Zhang, R Benenson, and B Schiele. Proc. ieee/cvf conf. comput. vis. pattern recognit. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit*, pages 185–194, 2017.

[239] Xu-Yao Zhang, Cheng-Lin Liu, and Ching Y Suen. Towards robust pattern recognition: A review. *Proceedings of the IEEE*, 108(6):894–922, 2020.

[240] Kai Zhao, Jingyi Xu, and Ming-Ming Cheng. Regularface: Deep face recognition via exclusive regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1136–1144, 2019.

[241] Liquan Zhao and Leilei Wang. A new lightweight network based on mobilenetv3. *KSII Transactions on Internet & Information Systems*, 16(1), 2022.

[242] Sijie Zhao, Tao Yue, and Xuemei Hu. Distribution-aware adaptive multi-bit quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9281–9290, 2021.

[243] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[244] Xuefei Zhe, Shifeng Chen, and Hong Yan. Directional statistics-based deep metric learning for image classification and retrieval. *Pattern Recognition*, 93:113–123, 2019.

[245] Kaixiang Zheng and En-Hui Yang. Knowledge distillation based on transformed teacher matching. In *To be submitted to International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2024.

[246] Yu-Jie Zheng, Si-Bao Chen, Chris H. Q. Ding, and Bin Luo. Model compression based on differentiable network channel pruning. *IEEE Trans. Neural Netw. Learn. Syst.*, 34(12):10203–10212, 2023.

[247] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations*, 2016.

[248] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[249] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.