

Enhancing Large Language Model Fine-Tuning for Classification Using Conditional Mutual Information

by

Thanushon Sivakaran

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Thanushon Sivakaran 2025

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

Large language models (LLMs) have achieved impressive advancements in recent years, showcasing their versatility and effectiveness in various tasks such as natural language understanding, generation, and translation. Despite these advancements, the full potential of information theory (IT) to further enhance the development of LLMs has yet to be fully explored. This thesis aims to bridge this gap by introducing the information-theoretic concept of Conditional Mutual Information (CMI) and applying it to the fine-tuning process of LLMs for classification tasks. We explore the promise of CMI in two primary ways: minimizing CMI to optimize a model’s standalone performance and maximizing CMI to improve knowledge distillation (KD) and create more capable student models.

To implement CMI in LLM fine-tuning, we adapt the recently proposed CMI-constrained deep learning framework, initially developed for image classification tasks, with necessary modifications for LLMs. In our experiments, we focus on applying CMI to LLM fine-tuning and knowledge distillation using the GLUE benchmark, a widely used suite of classification tasks for evaluating the performance of language models. Through minimizing CMI during the fine-tuning process, we achieve superior performance on 6 out of 8 GLUE classification tasks compared to the baseline BERT model. Furthermore, we explore the use of CMI to maximize information transfer during the KD process, where a smaller “student” model is trained to mimic the behavior of a larger, more powerful “teacher” model. By maximizing the teacher’s CMI, we ensure that richer semantic information is passed to the student, improving performance. Our results show that maximizing CMI during KD leads to substantial improvements in 6 out of 8 GLUE classification tasks when compared to DistilBERT, a popular distilled version of BERT.

Acknowledgements

I would like to sincerely thank my graduate supervisor, Professor En-Hui Yang, whose insightful advice and commitment to fostering critical thinking have greatly contributed to my development as a researcher. I feel truly fortunate to have had his guidance and support at this stage of my life.

I would like to extend my heartfelt thanks to my incredible lab mates, Ahmed Hussein Salamah, Shayan Mohajer Hamidi, Kaixiang Zheng, Renhao Tan, Linfeng Ye, Likhil Babu Pallatti, Yiwen Liu and Siyu Chen who have become some of my closest friends. Their support has been instrumental, not only in assisting with research but also in providing advice that will be cherished for a lifetime. Each of them brought joy and laughter to this journey, offering friendship and encouragement throughout. I feel fortunate to have shared this experience with such wonderful friends.

Finally, I would like to acknowledge my friends Eric-Khang Dao, Faisal Sabri, and Vrut Patel, who have collaborated with me on countless projects throughout my undergraduate and graduate studies. Together, we worked hard, shared laughs, and created memories that will last a lifetime.

Dedication

This is dedicated to my parents, whose unwavering dedication and hard work have shaped me into the person I am today. My father, the embodiment of perseverance, instilled in me the value of hard work from a young age. My mother taught me that no matter how difficult the journey, there is always light at the end of the tunnel, and that patience through adversity leads to better days.

This is also dedicated to my younger brothers, who never fail to check in on me at random moments, making sure I stay on track with my work while also reminding me to take time to relax.

And finally, this is dedicated to Thaksha, whose steadfast love and support have made this journey possible.

Table of Contents

Author’s Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	ix
List of Tables	x
1 Introduction	1
1.1 Bridging Information Theory and Large Language Models	1
1.2 Contributions	3
2 Background and Review	5
2.1 High-Order Markov Models	5
2.2 LLMs as High-Order Markov Source Models	6
2.3 Fine-Tuning LLMs for Downstream Tasks	9
2.4 Knowledge Distillation (KD) in LLM Training	11

3	Methodology	13
3.1	Clusters and Centroids	13
3.1.1	Limitations of OPDCs in LLM Fine-Tuning	14
3.1.2	Feature Probability Distribution Clusters (FPDCs)	14
3.2	Conditional Mutual Information (CMI) for FPDCs	15
3.2.1	Defining CMI for FPDCs	15
3.2.2	CMI for a Single FPDC	16
3.2.3	Average CMI Across FPDCs	16
3.3	Minimizing CMI for Improved Model Performance	17
3.3.1	Optimization Objective	17
3.3.2	Reformulating the Optimization Problem	17
3.3.3	Parallel Computation and Algorithmic Approach	18
3.3.4	Practical Considerations	19
3.4	Maximizing CMI for Enhanced Knowledge Distillation	19
3.4.1	Optimization Objective	20
3.4.2	Formulation of the Optimization Problem	20
3.4.3	Training Algorithm	20
3.4.4	Benefits of Maximizing CMI	21
3.4.5	Practical Considerations	21
3.4.6	Impact on Student Model Performance	22
4	Experiments	23
4.1	Minimizing CMI for Model Optimization	23
4.2	Maximizing CMI for Knowledge Distillation in Student Models	25
4.3	Attention Pattern Comparison	26
4.3.1	Cross-Entropy Attention Patterns	27
4.3.2	CMI Minimization Attention Patterns	28
4.3.3	CMI Maximization Attention Patterns	32
4.3.4	Summary of Attention Pattern Comparisons	36

5 Conclusion	38
References	39
APPENDICES	45
A GLUE Benchmark	46
B Hyperparameters	48
B.1 Minimizing CMI	48
B.2 Maximizing CMI	50
C Interpreting Attention Pattern Visualizations	52

List of Figures

1.1	Relationship between LLMs and IT.	2
2.1	An illustration of a transformer-based LLM architecture (GPT-1 architecture [27]).	7
4.1	Attention patterns from the last three layers of the BERT-base model fine-tuned with cross-entropy (2.6) on an MRPC sentence pair.	27
4.2	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.05$) (3.5) on an MRPC sentence pair.	29
4.3	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.10$) (3.5) on an MRPC sentence pair.	30
4.4	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.15$) (3.5) on an MRPC sentence pair.	31
4.5	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.25$) (3.5) on an MRPC sentence pair.	32
4.6	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.05$) (3.7) on an MRPC sentence pair.	33
4.7	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.10$) (3.7) on an MRPC sentence pair.	34
4.8	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.15$) (3.7) on an MRPC sentence pair.	35
4.9	Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.25$) (3.7) on an MRPC sentence pair.	36
C.1	Example image of attention patterns from a BERT-base model. This image is identical to Figure 4.1.	52

List of Tables

4.1	Performance comparison of BERT variants on dev sets of GLUE tasks. Row with * is from the DistilBERT paper [32].	24
4.2	Performance comparison of DistilBERT variants on dev sets of GLUE tasks. Row with * is from the DistilBERT paper [32].	26
B.1	Hyperparameters of BERT-base experiments for minimizing CMI.	49
B.2	Optimal λ values for minimizing CMI across GLUE datasets.	49
B.3	Hyperparameters of BERT-base (teacher) and DistilBERT (student) experiments for maximizing CMI.	50
B.4	Optimal λ values for maximizing CMI for teacher models across GLUE datasets.	51
B.5	Optimal α and T values for KD across GLUE datasets for CMI maximization experiments.	51

Chapter 1

Introduction

Transformer-based large language models (LLMs), such as OpenAI's GPT series [27, 28, 3, 9] and Google's BERT [7], have brought about a paradigm shift in the field of natural language processing (NLP). These models have redefined the boundaries of artificial intelligence (AI), achieving unprecedented performance in various complex tasks, including but not limited to text generation [28, 3], summarization [22, 21], translation [38, 23], and question answering [7, 46]. The underlying technology driving these advancements is the transformer architecture, first introduced in [38]. This architecture's ability to efficiently process sequential data and capture long-range dependencies, along with nuanced contextual relationships, has made it the backbone of modern AI. Consequently, LLMs have become indispensable tools for research, industry, and applications where natural language understanding and generation are pivotal.

1.1 Bridging Information Theory and Large Language Models

Despite their transformative impact, the conceptual roots of LLMs in information theory (IT) are often understated or overlooked. While much attention is paid to their neural architectures and training methodologies, these models can be fundamentally viewed as sophisticated computational mechanisms for establishing high-order Markov source models. This perspective aligns with the principles of IT as introduced by Claude Shannon in his landmark 1948 paper [33], where he laid the groundwork for understanding how information sources can be systematically modeled. Shannon demonstrated that text, or

any sequence of symbols, could be effectively modeled using first-order and second-order Markov processes. He illustrated this with examples of randomly generated word sequences, showing that as the order of the Markov model increases, the generated sequences increasingly resemble coherent English text. This observation formed the basis for Shannon’s use of Markov processes as a framework for modeling information sources, leading to foundational concepts like entropy and mutual information to quantify statistical uncertainty and dependencies.

However, Shannon did not propose any concrete or practical Markov models for generating text. The Markov processes he explored were limited in scope and complexity. Subsequent research, particularly in the context of universal online probabilistic data compression, sought to address this limitation. Researchers developed adaptive Markov source models to better capture the statistical structure of data [44], [43]. Yet, these adaptive models were constrained by their low Markov order, small vocabulary sizes, and lack of semantic representation. They were effective for specific tasks but fell short of representing the complexities inherent in natural language.

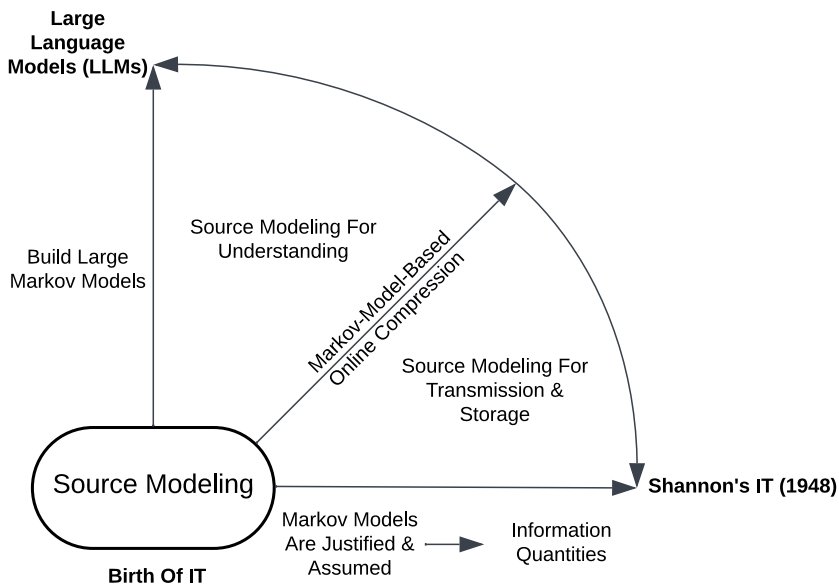


Figure 1.1: Relationship between LLMs and IT.

In contrast, LLMs have introduced a breakthrough by establishing Markov models that are orders of magnitude larger and more sophisticated. These models operate with vast vocabulary sizes and unprecedented Markov orders, enabling them to encode intricate

patterns and semantic meanings in text. Unlike traditional Markov models, which relied on limited contexts and statistical approximations, LLMs leverage massive datasets [3, 36, 37, 14] and extensive computational resources [34, 11, 31] to build highly adaptive, context-aware representations of language. In doing so, they provide a robust solution to the long-standing source modeling problem in IT, a challenge that has intrigued researchers since Shannon’s era.

This profound connection between LLMs and IT underscores the extent to which these models can be seen as an extension of Shannon’s pioneering ideas. As depicted in Fig. 1.1, LLMs transcend the limitations of earlier adaptive Markov models, bridging the gap between statistical modeling and semantic understanding. Their success demonstrates how IT principles can evolve and adapt to modern computational paradigms, offering new insights into source modeling, data representation, and the broader interplay between information and intelligence.

By situating LLMs within the framework of IT, we gain a deeper appreciation of their capabilities and their role in advancing our understanding of information processing. This perspective not only highlights the theoretical significance of LLMs but also opens up new avenues for exploring their applications in diverse domains where information modeling and prediction are critical.

1.2 Contributions

In what ways could LLMs and IT influence each other? As illustrated in Fig. 1.1, one potential avenue is leveraging LLMs to enhance data compression, a concept that has been partially investigated in [6] and [1]. Drawing inspiration from [45], this thesis takes the opposite approach by exploring how IT concepts can be applied to advance LLM methodologies. In this thesis, we establish a novel connection between large language models (LLMs) and information theory (IT), providing both theoretical insights and practical advancements. Specifically, the contributions are as follows:

1. **Reframing LLMs Through an Information-Theoretic Lens:** We describe LLMs in terms of IT by mathematically framing transformer decoder-based LLMs as computable high-order homogeneous Markov source models. This perspective provides a fresh theoretical foundation that aligns the underlying mechanisms of LLMs with key concepts in IT, bridging the gap between the two domains.

2. **Introducing Conditional Mutual Information (CMI) to LLM Fine-Tuning:**

Inspired by recent advancements in information-theoretic deep learning frameworks [45], we adapt the concept of Conditional Mutual Information (CMI) for fine-tuning LLMs in classification tasks. By integrating CMI constraints into the fine-tuning process, we explore its utility in two key areas:

- *Minimizing CMI to Enhance Standalone Model Performance:* Minimizing CMI during fine-tuning sharpens intra-class concentration, effectively aligning model predictions with their respective centroids. This leads to significant performance gains, as demonstrated on 6 out of 8 GLUE classification tasks, surpassing BERT’s results.
- *Maximizing CMI to Improve Knowledge Distillation (KD):* Maximizing CMI during the KD process enhances the transfer of richer contextual relationships from teacher to student models. This results in improved task-specific performance, with significant gains on 6 out of 8 GLUE classification tasks compared to DistilBERT.

3. **Adapting and Extending the CMI-Constrained Deep Learning Framework:**

We modify the existing CMI-constrained deep learning framework [45], originally designed for image classification, to suit the unique challenges of LLM-based classification tasks. This adaptation demonstrates the versatility and applicability of IT concepts in advancing LLM fine-tuning methodologies.

Chapter 2

Background and Review

Using a tokenization mechanism such as byte pair encoding (BPE) [10], a straightforward form of grammar-based coding [17, 44, 43], large language models (LLMs) transform a raw sequence of data symbols into a sequence of tokens. This preprocessing step is essential for handling textual data effectively [41]. The resulting collection of all tokens is referred to as a vocabulary, denoted by \mathcal{V} . The size of the vocabulary, $|\mathcal{V}|$, varies depending on the model architecture and design. For example, the vocabulary size is $|\mathcal{V}| = 50,257$ for GPT-3 [3] and $|\mathcal{V}| = 30,522$ for BERT [7]. Each token in the vocabulary is subsequently encoded as a d -dimensional vector of real numbers. These vector representations, often referred to as embeddings, serve as the input to the LLM, enabling it to process and learn from the underlying semantic and syntactic patterns in the data [25, 26].

2.1 High-Order Markov Models

Consider a sequence of tokens $t = [t_1, t_2, \dots, t_n]$, where t_i represents the i -th token in the sequence. The sequence t is said to follow a k -th order homogeneous Markov model if its joint probability distribution can be expressed as:

$$P(t_1, t_2, \dots, t_n) = P(t_1, t_2, \dots, t_k) \times \prod_{i=k+1}^n P(t_i | t_{i-k}, t_{i-k+1}, \dots, t_{i-1}), \quad (2.1)$$

for any sequence length $n > k$. Here, $P(t_1, t_2, \dots, t_k)$ is the initial distribution that determines the probability of the first k tokens. This initial distribution can be further

decomposed as:

$$P(t_1, t_2, \dots, t_k) = P(t_1)P(t_2 | t_1)P(t_3 | t_1, t_2) \dots \times P(t_k | t_1, t_2, \dots, t_{k-1}). \quad (2.2)$$

Subsequent tokens in the sequence are generated using conditional probabilities that depend on the previous k tokens. In this way, a k -th order homogeneous Markov model captures dependencies within the sequence by focusing on a fixed history length of k tokens.

The parameters of this model include:

- The initial token distribution $P(t_1, t_2, \dots, t_k)$, which specifies the probabilities of the first k tokens.
- The conditional probability distributions $P(t_{k+1} | t_1, \dots, t_k)$, $P(t_{k+2} | t_2, \dots, t_{k+1})$, and so on, which define how each subsequent token is generated based on its preceding k tokens.

This Markov framework provides a mathematical foundation for understanding the sequential nature of language. However, traditional Markov models are often limited in their capacity to model long-range dependencies due to their fixed-order constraints. In contrast, LLMs overcome these limitations by effectively implementing high-order Markov models with a large and dynamic vocabulary, significantly extending the range and complexity of dependencies they can model.

2.2 LLMs as High-Order Markov Source Models

Figure 2.1 illustrates the general architecture of a transformer decoder-based large language model (LLM), exemplified by the GPT-1 design [27]. Ignoring the final task-specific layer (whether for text generation, classification, or another downstream task), the core structure of an LLM can be decomposed into three fundamental components: (1) the input embedding layer, (2) a stack of recursive transformer blocks, and (3) the output layer, which computes conditional probability distributions over the next token. Below, we describe each of these components in the context of information-theoretic (IT) concepts.

Word Embedding. Each token $v \in \mathcal{V}$ in the vocabulary is encoded into a continuous d -dimensional vector representation $W_e(v) \in \mathbb{R}^d$. Collectively, the embeddings for all tokens in the vocabulary are represented as a matrix:

$$W_e = [W_e(v_1), W_e(v_2), \dots, W_e(v_{|\mathcal{V}|})]^T \in \mathbb{R}^{|\mathcal{V}| \times d}.$$

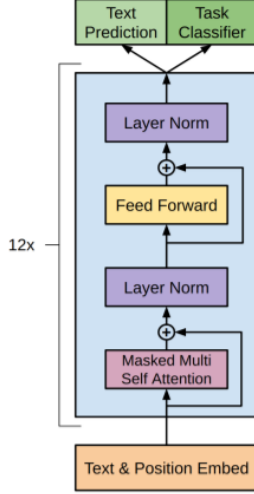


Figure 2.1: An illustration of a transformer-based LLM architecture (GPT-1 architecture [27]).

Position Embedding. To preserve the sequential structure of input data, each token’s position $1 \leq i \leq k$ (where k is the context length) is encoded as a position vector $W_p(i) \in \mathbb{R}^d$. These position embeddings are represented as a matrix:

$$W_p = [W_p(1), W_p(2), \dots, W_p(k)]^T \in \mathbb{R}^{k \times d}.$$

Input Embedding Layer. Given an input token sequence $t = [t_1, t_2, \dots, t_k]$, each token t_i is transformed into its input embedding as:

$$X_0(t_i) = W_e(t_i) + W_p(i),$$

where $W_e(t_i)$ represents the token embedding and $W_p(i)$ the position embedding. This process converts the input sequence into a matrix $X_0 = [X_0(t_1), X_0(t_2), \dots, X_0(t_k)]^T \in \mathbb{R}^{k \times d}$.

Recursive Transformer Blocks. The input embeddings are processed through a series of L transformer blocks. At layer l , the transformer block \mathcal{T}_l computes:

$$X_l(t_i) = \mathcal{T}_l(X_{l-1}(t_1), X_{l-1}(t_2), \dots, X_{l-1}(t_i)),$$

where X_{l-1} represents the output of the previous layer. The output of each block is causal, meaning it depends only on the current token t_i and all preceding tokens $[t_1, \dots, t_{i-1}]$, ensuring no information leakage from future tokens.

Output Layer. For each token t_i , the output layer computes the conditional probability distribution of the next token using:

$$P(\cdot | t_1, \dots, t_i) = \text{softmax}(X_L(t_i)W_e^T), \quad (2.3)$$

where $X_L(t_i)$ is the output of the final transformer block for token t_i , and W_e^T projects the embedding back to the token space. Note that while some models share parameters between the embedding matrix and the final pre-softmax un-embedding matrix, other models use separate parameters for these components, and the design choice may vary across implementations in the literature.

Causality and Markov Properties. In the above description, we have omitted the detailed operations performed within each transformer block \mathcal{T}_l . These blocks involve a combination of linear transformations, non-linear activation functions, attention mechanisms, and normalization steps, all parameterized by a set of trainable parameters Θ_l . Despite these complexities, a crucial characteristic of every transformer block \mathcal{T}_l is that it is inherently causal¹.

The causality constraint ensures that the computation of the output at position i depends only on the input at the current position t_i and all preceding tokens t_1, \dots, t_{i-1} , but not on any future tokens. As a result, the probability distribution in (2.3) can be interpreted as a conditional probability distribution that adheres strictly to the sequential nature of natural language data. This design is critical for autoregressive modeling, as it aligns the model’s internal structure with the temporal order of data generation.

At the start of an input sequence, a special token, typically denoted as $[Start]$, is used to initialize the model. This token acts as an anchor point, signaling the beginning of the sequence and providing context for generating the subsequent tokens.

From a mathematical perspective, the transformer decoder-based architecture described above can be equivalently represented as a family of k -th order homogeneous Markov source models:

$$\{P(\cdot | t_1, \dots, t_i; (W_e, W_p, \{\Theta_l\}_{l=1}^L)) : 1 \leq i \leq k\},$$

¹It is important to note that this property of causality does not apply to transformer encoder-based architectures like BERT [7], which are predominantly used for bidirectional tasks such as text classification or question answering.

where W_e and W_p represent the word and position embedding matrices, and $\{\Theta_l\}_{l=1}^L$ denote the parameters of the L transformer blocks. Collectively, these parameters are encapsulated as $\Theta = (W_e, W_p, \{\Theta_l\}_{l=1}^L)$.

The behavior of a specific LLM is determined during training, where the model parameters Θ are optimized to minimize a loss function. The objective function (referred to as cross-entropy loss [30]) for training is typically expressed as:

$$\mathcal{L}_1(\mathcal{U}) = \sum_{(u_1, \dots, u_k, u_{k+1}) \in \mathcal{U}} - \sum_{i=2}^{k+1} \log P(u_i | u_1, \dots, u_{i-1}; \Theta), \quad (2.4)$$

where \mathcal{U} represents the training corpus, consisting of sequences of tokens. The inner summation iterates over the tokens within each sequence, and the model is trained to maximize the likelihood of observing the next token u_i given the preceding tokens u_1, \dots, u_{i-1} .

After training, the word embedding matrix W_e encodes rich semantic information about each token, as reflected in the conditional probabilities computed in (2.3). These embeddings enable the model to capture intricate relationships between tokens and facilitate the generation of coherent and contextually relevant outputs.

The dimensionality of the embeddings (d) and the maximum context length (k) are key hyperparameters that influence the model’s capacity and performance. For example, GPT-3 [3] employs an embedding dimension of $d = 12,288$ and a context length of $k = 2,048$, enabling it to handle long-range dependencies and complex patterns in the data. In contrast, BERT [7], designed for bidirectional tasks, uses $d = 1,024$ and $k = 512$, reflecting its focus on shorter context windows and efficient representation learning.

2.3 Fine-Tuning LLMs for Downstream Tasks

Once an LLM has been pre-trained by minimizing the objective function defined in (2.4), it can be adapted for specific downstream tasks, such as text classification, sentiment analysis, or question answering, through a process called fine-tuning. Fine-tuning adjusts the model’s parameters to optimize performance on a smaller, labeled dataset tailored to the target task.

Let \mathcal{C} denote a labeled classification dataset with C distinct class labels. Each sample in \mathcal{C} consists of a sequence of tokens $x = [x_1, x_2, \dots, x_m]$ and a corresponding class label $y \in \{1, 2, \dots, C\}$. To provide context for the classification, a special token, such as the

[CLS] token, may be included at the beginning of x . If no such special token is available, the final token in the sequence is typically used as a representation of the entire sequence.

The fine-tuning process involves the following steps:

1. Pass the input sequence x through the pre-trained model to compute the output representation matrix X_L from the final transformer layer. Each row of X_L corresponds to the contextualized embedding of a token in x .
2. Extract the embedding vector \mathbf{h}_s corresponding to the special token [CLS] or, alternatively, the last token in the sequence. This vector serves as a condensed representation of the input sequence.
3. Add a fully connected classification layer, parameterized by a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times C}$, where d is the embedding dimension of \mathbf{h}_s . The model predicts the class probabilities using the following softmax function [2]:

$$P(\cdot | x) = \text{softmax}(\mathbf{h}_s \mathbf{W}). \quad (2.5)$$

The combined architecture, comprising the pre-trained LLM and the added classification layer, is then fine-tuned on the dataset \mathcal{C} . Fine-tuning is achieved by minimizing the following loss function over the model parameters (Θ, \mathbf{W}) :

$$\mathcal{L}_{Final} = \mathcal{L}_2(\mathcal{C}) + \gamma \mathcal{L}_1(\mathcal{C}), \quad (2.6)$$

where γ is a hyperparameter that balances the contribution of the pre-training and fine-tuning objectives. Here:

- $\mathcal{L}_2(\mathcal{C})$ is the supervised fine-tuning loss defined as:

$$\mathcal{L}_2(\mathcal{C}) = - \sum_{(x,y) \in \mathcal{C}} \log P(y | x; \Theta, \mathbf{W}), \quad (2.7)$$

where $P(y | x; \Theta, \mathbf{W})$ is the predicted probability of the correct class y given the input x .

- $\mathcal{L}_1(\mathcal{C})$ is the original pre-training objective (2.4) applied to the fine-tuning dataset \mathcal{C} . This term ensures that the model retains its pre-trained knowledge and avoids overfitting to the smaller labeled dataset [27].

The fine-tuning process is critical for adapting the general-purpose pre-trained LLM to the specific characteristics of the downstream task. By optimizing \mathcal{L}_{Final} , the model leverages both its pre-trained linguistic knowledge and task-specific labeled data to achieve high performance on the target task [27, 7, 24]. The embedding dimension d and the number of classes C directly influence the design of the added classification layer, making it versatile for a wide range of tasks and datasets.

2.4 Knowledge Distillation (KD) in LLM Training

The scalability challenges of LLMs, including their immense computational and memory requirements, necessitate effective techniques for model compression and optimization [16, 32, 15, 35]. Among these techniques, knowledge distillation (KD) has gained prominence as a powerful approach for reducing the size of models while maintaining competitive performance [4, 12].

KD leverages a **teacher-student framework**, where a large, pre-trained teacher model transfers its learned knowledge to a smaller, more computationally efficient student model. The student model is trained to mimic the behavior of the teacher by aligning its outputs with the predictions of the teacher. This process allows the student to approximate the teacher’s performance despite having significantly fewer parameters.

The training objective for KD combines two loss components:

$$\mathcal{L}_{KD} = (1 - \alpha)\mathcal{L}_{CE} + \alpha\mathcal{L}_{KL}, \tag{2.8}$$

where:

- \mathcal{L}_{CE} is the standard cross-entropy loss (2.4), which measures the difference between the student’s predictions and the ground truth labels from the training data. A student also tends to be pretrained with this loss prior to KD.
- $\mathcal{L}_{KL} = \text{KL}(P_{\text{teacher}} \parallel P_{\text{student}})$ represents the Kullback–Leibler (KL) divergence [19] between the softened probability distributions of the teacher (P_{teacher}) and the student (P_{student}). The softened predictions are obtained by applying a temperature scaling factor $T > 1$ to the logits, which enhances the information contained in the probability distribution of less-likely classes.
- $\alpha \in [0, 1]$ is a hyperparameter that determines the balance between the two components of the loss.

The softened probability distributions are computed as:

$$P_{\text{soft}}(y_i | x) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \quad (2.9)$$

where z_i represents the logit for class i , and T is the temperature. Higher values of T produce smoother probability distributions, enabling the student to capture the relative importance of less probable classes that are often ignored by hard labels in \mathcal{L}_{CE} .

Benefits of Knowledge Distillation

1. **Parameter Efficiency:** By learning from the teacher’s rich, nuanced knowledge, the student model achieves performance close to the teacher while requiring significantly fewer parameters, reducing memory and computational costs.
2. **Faster Inference:** Smaller student models enable faster inference, making them ideal for deployment in real-time or resource-constrained environments, such as mobile devices or edge computing systems.
3. **Transfer of Generalization:** The teacher model, having been trained on a vast corpus, encapsulates generalizable patterns. KD allows the student to inherit these patterns, even if the training data for the student is limited or task-specific.
4. **Flexibility in Model Design:** The teacher and student models do not need to have identical architectures. For instance, a teacher model based on transformers can distill knowledge into a student model based on convolutional [20, 18] or recurrent architectures [29, 13], depending on the deployment requirements.

Fine-Tuning with KD

During fine-tuning, KD can also be applied to adapt the student model to a specific downstream task. By combining the teacher’s knowledge with task-specific labeled data, the student can be optimized using a hybrid loss that integrates \mathcal{L}_{KD} with task-specific objectives. This approach ensures that the student retains the generalization capabilities of the teacher while excelling in the target task.

Chapter 3

Methodology

In this section, we present our methodology for leveraging conditional mutual information (CMI) to enhance large language model (LLM) fine-tuning for classification tasks. This uses the connection between LLMs and IT established earlier to demonstrate how IT principles can be applied to improve LLM performance. Specifically, we propose two complementary objectives: (1) minimizing CMI to improve the performance of standalone models, and (2) maximizing CMI to enhance the effectiveness of knowledge distillation (KD) in transferring knowledge from teacher to student models. Both approaches draw inspiration from CMI-constrained deep learning frameworks originally developed for image classification [45, 42, 47], but we adapt these techniques to address the unique challenges inherent in LLM-based classification tasks.

Notation

Let $N > 1$ be an integer, and define $[N] = \{1, 2, \dots, N\}$. For any discrete set S , let $\mathcal{P}(S)$ denote the set of all probability distributions over S . For a given labeled dataset \mathcal{C} with C distinct class labels, each sample instance is denoted as (x, y) , where x represents a sequence of input tokens, and $y \in [C]$ is the corresponding label.

3.1 Clusters and Centroids

For each sample instance $(x, y) \in \mathcal{C}$, the combined model processes the input x to generate a d -dimensional feature vector \mathbf{h}_s , denoted as \mathbf{h}_s^x and referred to as the *feature vector*.

Subsequently, this feature vector is transformed into a probability distribution $P(\cdot | x)$ over the C classes, as defined in (2.5).

To facilitate analysis, we group instances in \mathcal{C} by their labels. For each label $y \in [C]$, let:

$$\mathcal{C}_y = \{(x, y) : (x, y) \in \mathcal{C}\}$$

be the subset of all instances in \mathcal{C} with label y , and let $n_y = |\mathcal{C}_y|$ denote the size of this subset. The model maps \mathcal{C}_y into a cluster of output probability distributions:

$$\{P(\cdot | x) : (x, y) \in \mathcal{C}_y\} \subseteq \mathcal{P}([C]),$$

referred to as the *output probability distribution cluster* (OPDC) corresponding to label y . Note that this 'cluster' refers to a set of class-conditioned vectors residing within the probability simplex.

3.1.1 Limitations of OPDCs in LLM Fine-Tuning

For image classification tasks, information-theoretic quantities such as CMI, computed from OPDCs, have been shown to improve model performance by enhancing both intra-class concentration and inter-class separation within the output probability distribution space [45, 42, 47]. However, in LLM-based classification tasks, the number of classes C is typically very small (e.g., $C = 2$ or $C = 3$ for binary or ternary classification tasks). This small value of C limits the effectiveness of OPDCs in capturing meaningful variations within the data. To address this limitation, we propose an alternative representation: the *feature probability distribution cluster* (FPDC).

3.1.2 Feature Probability Distribution Clusters (FPDCs)

To construct FPDCs, we first compute a probability distribution over the feature space for each feature vector \mathbf{h}_s^x . Specifically, we define:

$$f_x = \text{softmax}(\mathbf{h}_s^x),$$

where $f_x \in \mathcal{P}([d])$ is a probability distribution in the d -dimensional space of features. Softmax is used to convert each feature vector into a probability distribution, enabling the subsequent use of KL divergence in the CMI derivation in Section 3.2.

For each label $y \in [C]$, the corresponding FPDC is defined as:

$$\{f_x : (x, y) \in \mathcal{C}_y\} \subseteq \mathcal{P}([d]).$$

Unlike OPDCs, FPDCs operate in a significantly higher-dimensional space ($d \gg C$), allowing for more granular representations of class-specific feature distributions. To summarize each FPDC, we compute its centroid as:

$$g_y = \frac{1}{n_y} \sum_{(x,y) \in \mathcal{C}_y} f_x, \quad (3.1)$$

where $g_y \in \mathcal{P}([d])$ is the centroid of the FPDC for label y .

3.2 Conditional Mutual Information (CMI) for FPDCs

We now introduce a new information-theoretic quantity to evaluate the concentration of feature probability distribution clusters (FPDCs). This measure leverages the relationship between the feature vectors generated by the model and their corresponding labels.

3.2.1 Defining CMI for FPDCs

Given an input instance x , the combined model generates the feature vector \mathbf{h}_s^x , which is subsequently transformed into a probability distribution $f_x = \text{softmax}(\mathbf{h}_s^x)$ over the feature dimensions. Using f_x , we can define a random variable Z that represents a feature dimension index. Specifically, for a given input x , the value of $Z = j$ is sampled with probability $f_x(j)$, where $j \in \{1, 2, \dots, d\}$.

Now, consider randomly selecting a sample instance (X, Y) from the dataset \mathcal{C} , where X and Y are random variables representing the input sequence and its corresponding label, respectively. The combined model processes X to produce the feature distribution f_X , and a corresponding random dimension index Z is sampled.

It can be shown that the random variables $Y \rightarrow X \rightarrow Z$ form a Markov chain:

$$P_{Z|XY}(Z = j | X = x, Y = y) = P_{Z|X}(Z = j | X = x) = f_x(j). \quad (3.2)$$

This relationship arises because Z is fully determined by X and is conditionally independent of Y given X .

3.2.2 CMI for a Single FPDC

For a specific label $y \in [C]$, consider the subset of samples $\mathcal{C}_y = \{(x, y) \in \mathcal{C}\}$. Within this subset, by the same arguments in [45, 42], the CMI between X (the input) and Z (the feature dimension index) conditioned on $Y = y$ (the label) is given by:

$$I(X; Z | Y = y) = \frac{1}{n_y} \sum_{(x,y) \in \mathcal{C}_y} \text{KL}(f_x \| g_y), \quad (3.3)$$

where:

- $n_y = |\mathcal{C}_y|$ is the number of samples with label y ,
- $\text{KL}(f_x \| g_y)$ is the Kullback–Leibler (KL) divergence [19] between the feature distribution f_x for sample x and the centroid g_y of the FPDC corresponding to label y , as defined in (3.1).

This CMI quantifies the concentration of the FPDC corresponding to label y , with lower values indicating tighter clustering of feature distributions around the centroid g_y .

3.2.3 Average CMI Across FPDCs

To evaluate the overall concentration across all FPDCs, we compute the average CMI between X and Z given Y :

$$\begin{aligned} I(X; Z | Y) &= \frac{1}{C} \sum_{y \in [C]} I(X; Z | Y = y) \\ &= \frac{1}{C} \sum_{y \in [C]} \frac{1}{n_y} \sum_{(x,y) \in \mathcal{C}_y} \text{KL}(f_x \| g_y) \\ &= \frac{1}{C} \sum_{(x,y) \in \mathcal{C}} \text{KL}(f_x \| g_y). \end{aligned} \quad (3.4)$$

Here:

- C is the total number of labels (classes),
- The summation over $(x, y) \in \mathcal{C}$ iterates over all samples in the dataset.

This average CMI provides a global measure of the concentration of FPDCs, capturing the overall alignment of feature distributions with their respective centroids across all labels.

3.3 Minimizing CMI for Improved Model Performance

To improve the performance of the standalone combined model for classification tasks, we incorporate the CMI, $I(X; Z | Y)$, into the model’s optimization objective. This inclusion aims to encourage tighter clustering of feature probability distributions, thereby enhancing the model’s representation of class-specific features.

3.3.1 Optimization Objective

The new optimization objective is defined as:

$$\mathcal{L}_{\text{MinCMI}} = \mathcal{L}_{\text{Final}} + \lambda I(X; Z | Y), \quad (3.5)$$

where:

- $\mathcal{L}_{\text{Final}}$ is the original loss function, defined in (2.6).
- $\lambda > 0$ is a hyperparameter that controls the trade-off between the original objective $\mathcal{L}_{\text{Final}}$ and the CMI term.

The term $I(X; Z | Y)$ quantifies the concentration of FPDCs, as introduced in (3.4). By minimizing $I(X; Z | Y)$, we encourage feature distributions f_x within the same class to align more closely with their respective centroids g_y .

3.3.2 Reformulating the Optimization Problem

The combined model is fine-tuned through a training process that solves the following minimization problem:

$$\begin{aligned} & \min_{(\Theta, \mathbf{W})} [\mathcal{L}_{\text{Final}} + \lambda I(X; Z | Y)] \\ &= \min_{(\Theta, \mathbf{W})} \left[\mathcal{L}_{\text{Final}} + \frac{\lambda}{C} \sum_{(x,y) \in \mathcal{C}} \text{KL}(f_x \| g_y) \right] \\ &= \min_{(\Theta, \mathbf{W})} \min_{\{G_y\}_{y \in [C]}} \left[\mathcal{L}_{\text{Final}} + \frac{\lambda}{C} \sum_{(x,y) \in \mathcal{C}} \text{KL}(f_x \| G_y) \right], \end{aligned} \quad (3.6)$$

where:

- (Θ, \mathbf{W}) are the trainable parameters of the combined model.
- For each $y \in [C]$, G_y is a dummy probability distribution in $\mathcal{P}([d])$.
- The inner minimization achieves its optimum when $G_y = g_y$, the centroid of the FPDC corresponding to label y .

This reformulation introduces dummy distributions G_y as placeholders for the centroids g_y , enabling a more flexible optimization process. The explanation for how the dummy distributions are generated is provided in the next section.

3.3.3 Parallel Computation and Algorithmic Approach

The objective function in (3.6) is sample-instance additive, allowing efficient parallelization on GPUs. Specifically, given (Θ, \mathbf{W}) and $\{G_y\}_{y \in [C]}$, the computation of $\text{KL}(f_x \| G_y)$ for each sample (x, y) can be distributed across multiple processors.

To solve the double minimization problem in (3.6), we adopt an alternating optimization approach inspired by [45]. The process involves the following steps:

1. **Centroid Update:** For fixed model parameters (Θ, \mathbf{W}) , update each dummy distribution G_y to minimize the KL divergence term. This is achieved by setting G_y to the centroid g_y of the FPDC for label y , as defined in (3.1).
2. **Model Parameter Update:** For fixed centroids $\{G_y\}_{y \in [C]}$, update the model parameters (Θ, \mathbf{W}) to minimize the combined loss $\mathcal{L}_{\text{MinCMI}}$ using gradient-based optimization techniques.
3. **Iterative Refinement:** Alternate between the centroid update and model parameter update steps until convergence.

This alternating optimization effectively handles the non-convexity of the objective function and ensures that the model learns compact and well-separated feature clusters for classification.

3.3.4 Practical Considerations

In practice, the hyperparameter λ plays a critical role in balancing the original loss $\mathcal{L}_{\text{Final}}$ and the CMI term. A small value of λ may underemphasize the importance of cluster concentration, while a large value may overly constrain the model’s flexibility. Grid search or Bayesian optimization techniques can be used to tune λ for optimal performance.

Furthermore, this methodology is computationally efficient for large-scale datasets due to its parallelizability, making it well-suited for training large language models on modern GPU architectures.

Benefits of Minimizing CMI

The inclusion of $I(X; Z | Y)$ in the optimization objective offers the following advantages:

- **Improved Class Separation:** By encouraging tighter clustering of feature distributions within each class, the model achieves better class separation, leading to improved classification accuracy.
- **Robustness to Noise:** Minimizing CMI reduces intra-class variability, making the model more robust to noisy inputs and outliers.
- **Enhanced Generalization:** Compact feature representations result in improved generalization performance on unseen data.

3.4 Maximizing CMI for Enhanced Knowledge Distillation

To improve the performance of distilled student models, we propose maximizing the CMI, $I(X; Z | Y)$, during the fine-tuning process of the teacher model. By maximizing CMI, the teacher model encodes richer and more diverse contextual information into its representations [47]. This facilitates the transfer of a broader range of information to the student model, ultimately enhancing the student’s ability to generalize to specialized classification tasks.

3.4.1 Optimization Objective

The new optimization objective for the teacher model is defined as:

$$\mathcal{L}_{\text{MaxCMI}} = \mathcal{L}_{\text{Final}} - \lambda I(X; Z | Y), \quad (3.7)$$

where:

- $\mathcal{L}_{\text{Final}}$ is the primary loss function for the task, as defined in (2.6).
- $\lambda > 0$ is a hyperparameter that balances the importance of the task-specific objective $\mathcal{L}_{\text{Final}}$ and the CMI term.

By including $-\lambda I(X; Z | Y)$ in the objective, the model is encouraged to spread feature probability distribution clusters (FPDCs) more widely around their centroids. This dispersion captures subtle interdependencies and fine-grained patterns in the data, which are crucial for effective knowledge distillation.

3.4.2 Formulation of the Optimization Problem

The teacher model is fine-tuned by solving the following optimization problem:

$$\begin{aligned} & \min_{(\Theta, \mathbf{W})} [\mathcal{L}_{\text{Final}} - \lambda I(X; Z | Y)] \\ & = \min_{(\Theta, \mathbf{W})} \left[\mathcal{L}_{\text{Final}} - \frac{\lambda}{C} \sum_{(x,y) \in \mathcal{C}} \text{KL}(f_x \| g_y) \right]. \end{aligned} \quad (3.8)$$

Here, (Θ, \mathbf{W}) are the parameters of the combined model acting as the teacher, and $I(X; Z | Y)$ measures the diversity within each FPDC. Maximizing this term ensures that FPDCs for each class become more dispersed, promoting internal diversity within each class while retaining subtle contextual patterns in the data.

3.4.3 Training Algorithm

To optimize the objective in (3.8), we use an alternating optimization strategy inspired by [47]. The process can be summarized as follows:

1. **Feature Cluster Diversification:** For each class $y \in [C]$, encourage the dispersion of feature probability distributions f_x by penalizing proximity to the centroid g_y . This increases the diversity of features within each class while retaining meaningful patterns.
2. **Model Parameter Update:** Optimize the teacher model parameters (Θ, \mathbf{W}) to minimize the task-specific loss $\mathcal{L}_{\text{Final}}$ and maximize CMI using gradient-based techniques.
3. **Iterative Refinement:** Alternate between the two steps until convergence. This iterative process ensures that the teacher model encodes both task-specific accuracy and diverse contextual knowledge.

3.4.4 Benefits of Maximizing CMI

Maximizing CMI during teacher model fine-tuning offers several key advantages:

- **Richer Knowledge Representation:** By increasing the dispersion of FPDCs, the teacher model captures finer-grained contextual dependencies, which are essential for transferring nuanced information to the student model.
- **Improved Knowledge Distillation:** A teacher trained to maximize CMI provides predictions that are more informative and diverse, enabling the student model to learn better representations and perform well on specialized classification tasks.
- **Enhanced Generalization:** The diversity encoded in the teacher’s representations leads to improved generalization for the student model on unseen data.
- **Preservation of Subtle Patterns:** While maximizing CMI promotes dispersion, it simultaneously retains subtle data patterns within each class, ensuring that the knowledge transferred to the student is both broad and precise.

3.4.5 Practical Considerations

The hyperparameter λ plays a critical role in balancing the task-specific loss and the CMI term. Tuning λ carefully is essential to avoid excessive dispersion of FPDCs, which could dilute meaningful class-specific information. Standard hyperparameter optimization

techniques, such as grid search or Bayesian optimization techniques, can be employed to find an optimal λ for a given task.

Furthermore, the algorithm is well-suited for parallel computation, as the optimization of $\text{KL}(f_x||g_y)$ for individual samples can be distributed across multiple GPUs, making it scalable for large datasets.

3.4.6 Impact on Student Model Performance

A teacher model trained using the proposed approach yields predictions that encode a wealth of diverse and contextually rich information. These predictions serve as a superior knowledge source for the student model, significantly improving the student’s performance on both specialized classification tasks.

Chapter 4

Experiments

In this section, we detail the experimental setup and present the results of leveraging CMI to improve LLM fine-tuning for classification tasks. Although the proposed methodologies (based on novel definitions of CMI) could be extended to the pretraining phase of LLM development, the computational requirements for such applications are significantly higher. Specifically, during pretraining, the number of centroids would correspond to the vocabulary size of the model, typically exceeding 30,000 tokens. Optimizing CMI under such conditions would entail prohibitive computational costs, both in terms of memory and processing time.

Given these constraints, we focus our experiments on the fine-tuning phase, where the application of CMI can be both computationally efficient and practically impactful. Fine-tuning provides a focused environment to evaluate the efficacy of CMI-enhanced optimization in improving classification performance. This approach allows us to analyze CMI’s ability to capture nuanced dependencies within class-specific feature probability distribution clusters (FPDCs) and its impact on classification accuracy.

4.1 Minimizing CMI for Model Optimization

To investigate the impact of CMI minimization on optimizing LLMs for standalone classification tasks, we conducted a series of experiments using the BERT-base model [7] on all classification tasks included in the GLUE benchmark [40]. Each model was initialized with pretrained weights and subsequently fine-tuned by minimizing its CMI, as detailed in Section 3.3. This approach aims to encourage the model to concentrate on task-relevant

features while mitigating the influence of extraneous contextual dependencies (such as irrelevant context or overfitting to irrelevant parts of the input), thereby enhancing classification accuracy.

Experimental Setup. The fine-tuning process involved training the BERT-base model with CMI minimization by incorporating the loss function defined in (3.5). To analyze the effect of the CMI trade-off parameter, the experiments explored a range of λ values, specifically $\lambda \in \{0, 0.05, 0.1, 0.15, 0.25, 0.4, 0.5, 0.65, 0.75, 0.9, 1.0\}$. Each model was fine-tuned for 5 epochs, with the optimal model checkpoint selected based on the highest task-specific evaluation metric (detailed in Appendix A). To ensure robustness and reduce the impact of random initialization, each configuration was repeated across three independent runs, and the median result was reported. Additional hyperparameters are reported in Appendix B.1.

Table 4.1: Performance comparison of BERT variants on dev sets of GLUE tasks. Row with * is from the DistilBERT paper [32].

Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	WNLI
BERT-base*	56.3	86.7	88.6	91.8	89.6	69.3	92.7	53.5
BERT-base (Min CMI)	62.1	84.7	91.3	91.7	91.4	72.2	93.6	57.7
	+5.8%	-2.0%	+2.7%	-0.1%	+1.8%	+2.9%	+0.9%	+4.2%

Results and Analysis. The results of our experiments are summarized in Table 4.1. Fine-tuning the BERT-base model with minimized CMI outperformed the standard BERT-base fine-tuning procedure across the majority of GLUE tasks. Specifically, CMI minimization resulted in significant performance improvements on 6 out of 8 datasets. The most substantial gains were observed for the CoLA and WNLI tasks, where the minimized CMI-trained model achieved relative improvements of 5.8% and 4.2%, respectively. These results highlight the potential of CMI minimization to refine the model’s internal representations by focusing on features that are most relevant to the classification task.

The observed improvements can be attributed to the tighter intra-class concentration achieved through CMI minimization. By aligning the FPDCs with their corresponding centroids, the model effectively reduces intra-class variability while suppressing the influence of noisy or irrelevant contextual dependencies. This enhanced representational clarity

enables the model to prioritize task-critical features, leading to more accurate and robust predictions.

4.2 Maximizing CMI for Knowledge Distillation in Student Models

This section investigates the effectiveness of maximizing CMI in facilitating knowledge transfer from teacher models to student models. Specifically, we fine-tuned teacher models by maximizing their CMI, as described in Section 3.4, before applying knowledge distillation (KD). The experiments followed a similar methodology to those described in Section 4.1, with notable modifications tailored to the knowledge transfer paradigm.

Experimental Setup. For the teacher models, we employed the BERT-base architecture [7], initialized with pretrained weights. The fine-tuning process involved maximizing CMI by incorporating the optimization objective outlined in (3.7). A range of λ values, specifically $\lambda \in \{0, 0.05, 0.1, 0.15, 0.25, 0.4, 0.5, 0.65, 0.75, 0.9, 1.0\}$, was explored to balance the two loss terms. Each model was trained for 5 epochs, and the best-performing epoch was selected based on the highest evaluation metric specific to the dataset (detailed in Appendix A). To ensure statistical robustness, each configuration was repeated across three independent runs. Additional hyperparameters are reported in Appendix B.2.

To distill knowledge into student models, we selected teacher models that demonstrated the highest evaluation metric-to-CMI ratio among all λ configurations. Interestingly, most of the selected teacher models corresponded to λ values between 0.05 and 0.40, suggesting an optimal trade-off between contextual richness and task-specific performance.

The student model used in this study was DistilBERT [32], initialized by retaining every other layer from the teacher model, as outlined in the original DistilBERT methodology. For the distillation process, we employed the traditional KD framework [12], utilizing $\alpha \in \{0.05, 0.1, 0.15, 0.25, 0.4, 0.5, 0.65, 0.75, 0.9\}$, and $T \in \{1, 2, 3, 4, 5\}$. The student models were trained for 10 epochs, with the best-performing epoch selected based on the highest evaluation metric for each dataset. To ensure reliability, each configuration was repeated across three independent runs, with the median performance reported.

Results and Analysis. Table 4.2 presents the results of our experiments. Student models distilled from teachers fine-tuned with maximized CMI achieved notable performance

Table 4.2: Performance comparison of DistilBERT variants on dev sets of GLUE tasks. Row with * is from the DistilBERT paper [32].

Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	WNLI
DistilBERT*	51.3	82.2	87.5	89.2	88.5	59.9	91.3	56.3
DistilBERT (Max CMI)	51.4	81.6	88.6	88.3	90.7	61.7	91.9	63.4
	+0.1%	-0.6%	+1.1%	-0.9%	+2.2%	+1.8%	+0.6%	+7.1%

improvements over those trained using the standard DistilBERT process. Specifically, these gains were observed on 6 out of 8 GLUE classification tasks, with WNLI and QQP showing the most pronounced improvements of 7.1% and 2.2%, respectively. These findings highlight the efficacy of maximizing CMI in enabling teacher models to encode and transfer richer contextual dependencies to their students.

By capturing broader patterns and preserving subtle dependencies in the data, maximized CMI-trained teacher models effectively enhance the performance of student models, particularly for specialized classification tasks. This approach demonstrates the potential for CMI maximization to bridge the gap between teacher and student models, enabling more effective knowledge transfer and improved task performance.

4.3 Attention Pattern Comparison

To investigate the impact of different training objectives on the attention mechanisms of LLMs, we employed BERTViz [39] to visualize the attention maps from the last three layers of the BERT-base model. We compared attention patterns across models fine-tuned using three different objectives: the standard cross-entropy loss (2.6), the CMI minimization objective (3.5), and the CMI maximization objective (3.7). These visualizations provide insight into how models allocate attention across input tokens and how these distributions vary depending on the applied objective. Appendix C shows how these attention pattern visualizations can be interpreted.

For this analysis, we selected a representative example from the Microsoft Research Paraphrase Corpus (MRPC) [8], a dataset focused on identifying semantic equivalence between sentence pairs. The specific sample used consists of the following sentences:

1. *Security lights have also been installed and police have swept the grounds for booby traps.*

2. *Security lights have also been installed on a barn near the front gate.*

These sentences are labeled as semantically equivalent, making them an ideal case for examining how different training objectives influence attention behavior in capturing contextual relationships. By analyzing these attention maps, we aim to understand whether CMI-based objectives lead to more structured and interpretable attention distributions, particularly in tasks requiring fine-grained semantic understanding.

4.3.1 Cross-Entropy Attention Patterns

The standard cross-entropy training objective (2.6) yields highly diverse and dispersed attention distributions. As illustrated in Figure 4.1, the attention heads in the cross-entropy trained model exhibit a wide range of activation patterns, often attending to multiple, seemingly disparate parts of the input.

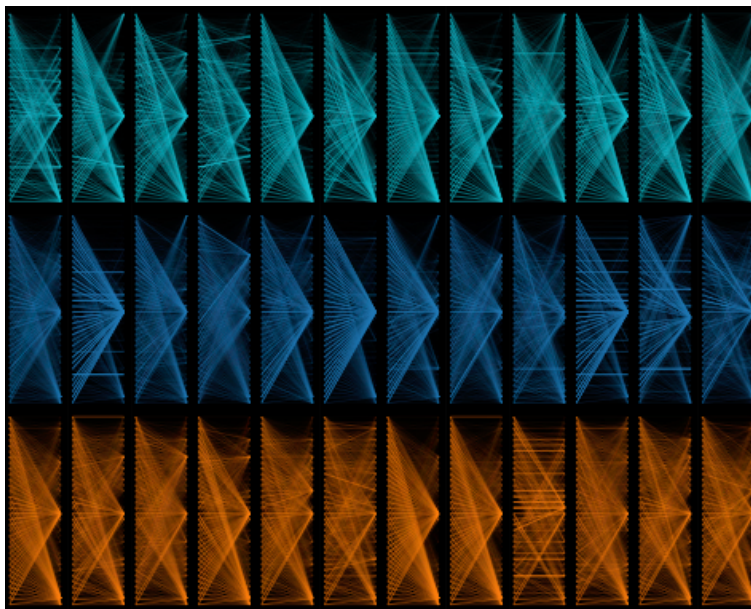


Figure 4.1: Attention patterns from the last three layers of the BERT-base model fine-tuned with cross-entropy (2.6) on an MRPC sentence pair.

The attention maps reveal a tendency for the model to distribute its attention broadly, rather than concentrating on a select few informative tokens. While this diffuse attention

might allow the model to capture a wider range of linguistic features and potentially model long-range dependencies, it comes at the cost of reduced focus and increased noise. The attention patterns appear irregular and unsynchronized, suggesting that different heads are attending to different, and potentially less relevant, aspects of the input. This scattering of attention can be interpreted as the model exploring a larger hypothesis space, but without a strong prior towards structured or consistent interpretations.

This lack of clear focus has several potential drawbacks. First, it can hinder the model’s ability to learn strong, discriminative features. By attending to too many parts of the input, including irrelevant or noisy tokens, the model might reduce the importance of truly informative features. Second, the dispersed attention can negatively impact the efficiency of information processing. The model may struggle to establish clear relationships between the most salient tokens for the given task, potentially leading to suboptimal performance on classification tasks. Finally, this irregularity poses challenges for KD. The lack of a well-defined, consistent attention strategy in the teacher model makes it difficult for a student model to effectively mimic its behavior and learn from its predictions. The student receives a noisy signal, making it harder to extract the essential, transferable knowledge.

4.3.2 CMI Minimization Attention Patterns

Minimizing CMI encourages a model to distribute its attention more evenly across key regions of the input. This moderation prevents excessive rigidity in attention allocation, allowing the model to generalize effectively across different sentence structures. While some variability is retained, attention is more consistently directed toward relevant tokens, reducing unnecessary noise while preserving the model’s ability to capture diverse linguistic relationships. This balance ensures that the model forms clearer token dependencies without over-regularizing attention patterns.

The following figures illustrate the evolution of attention patterns across increasing values of λ , demonstrating how higher CMI minimization influences the behavior of the model’s attention heads.

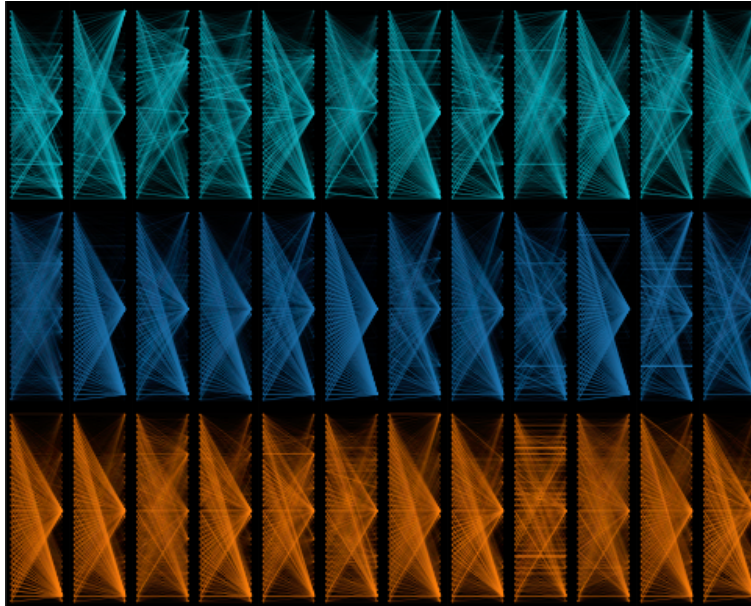


Figure 4.2: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.05$) (3.5) on an MRPC sentence pair.

At $\lambda = 0.05$, as shown in Figure 4.2, the attention patterns exhibit a high degree of variability and lack of clear structure. The attention heads appear to operate independently, focusing on different and seemingly arbitrary aspects of the input. This suggests that minimizing CMI at this level allows the model to explore a broad range of potential relationships, without enforcing any consistency or shared focus. While this diversity might potentially capture subtle nuances in the data, it also makes the model’s behavior less interpretable and potentially more susceptible to noise.

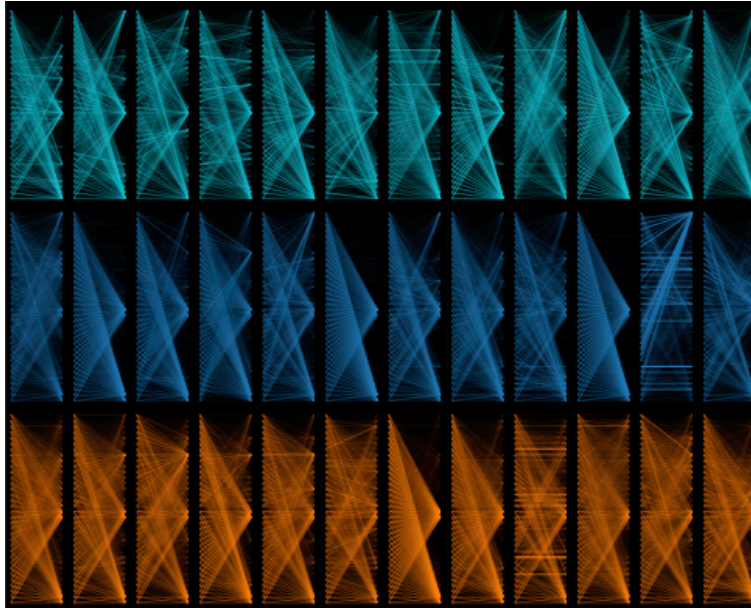


Figure 4.3: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.10$) (3.5) on an MRPC sentence pair.

Increasing λ to 0.10, as shown in Figure 4.3, reveals a subtle shift in the attention patterns. While diversity remains a key characteristic, certain heads begin to exhibit more focused attention, often converging on specific tokens or short phrases within the input. This suggests that as the influence of CMI minimization grows, the model starts to refine its exploration of potential dependencies. Instead of completely random attention, we observe the emergence of more localized and selective focus.

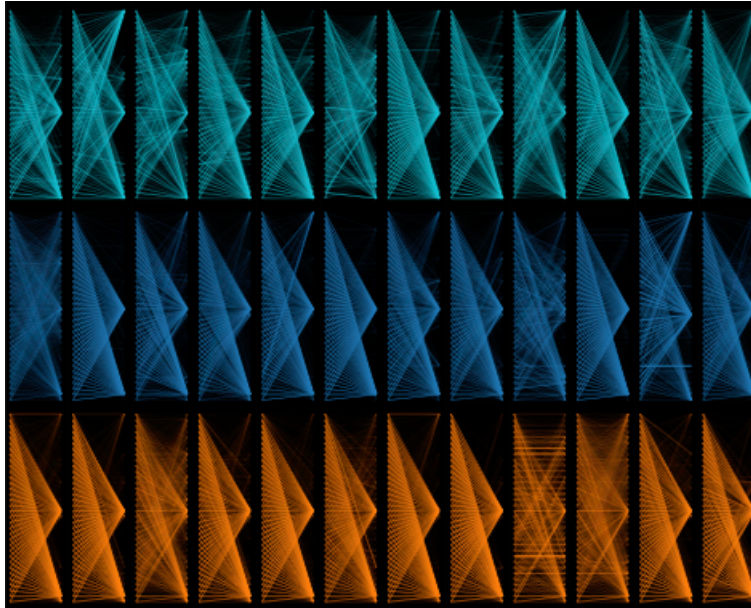


Figure 4.4: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.15$) (3.5) on an MRPC sentence pair.

With $\lambda = 0.15$, as illustrated in Figure 4.4, the attention patterns become even more refined. The trend towards focused attention within individual heads continues, with a growing number of heads exhibiting highly selective attention to specific tokens or short phrases. Different heads appear to specialize in capturing distinct aspects or features of the input, potentially contributing to a more comprehensive representation of the input sample.

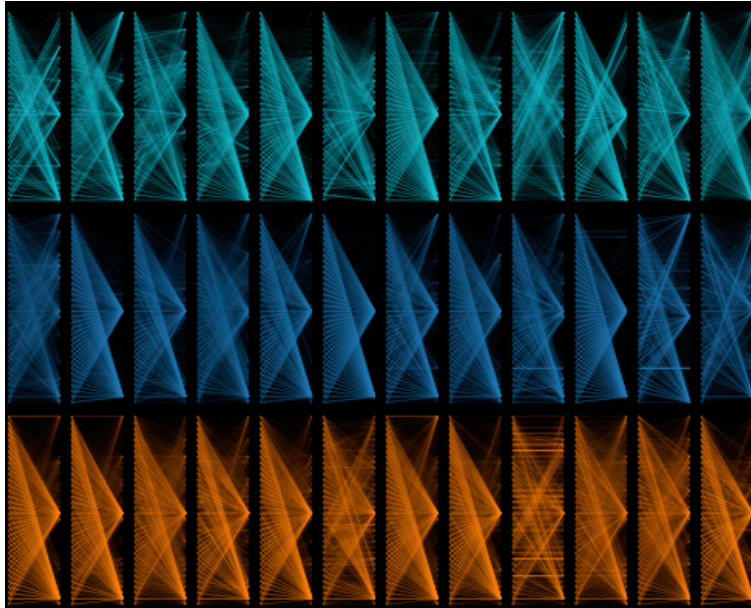


Figure 4.5: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI minimization ($\lambda = 0.25$) (3.5) on an MRPC sentence pair.

Finally, at $\lambda = 0.25$, as shown in Figure 4.5, the attention patterns reach a high degree of specialization. The individual attention heads exhibit extremely focused attention. The diversity is now highly structured, with different heads clearly specializing in distinct aspects of the input. This suggests that with strong CMI minimization, the model learns to decompose the input into a set of highly specific features, each captured by a different attention head.

4.3.3 CMI Maximization Attention Patterns

Maximizing CMI enforces more structured attention distributions by encouraging the model to focus on highly informative input tokens. This objective aims to stabilize attention mechanisms, reducing redundancy and suppressing irrelevant dependencies. As λ increases, we observe a clear trend toward more uniform and consistent attention patterns across heads. However, while this consistency can enhance interpretability and structured learning, it may also introduce over-regularization, potentially leading to a loss in performance due to an excessively rigid focus.

The following figures illustrate the evolution of attention patterns across increasing values of λ , demonstrating how higher CMI maximization influences the behavior of the model’s attention heads.

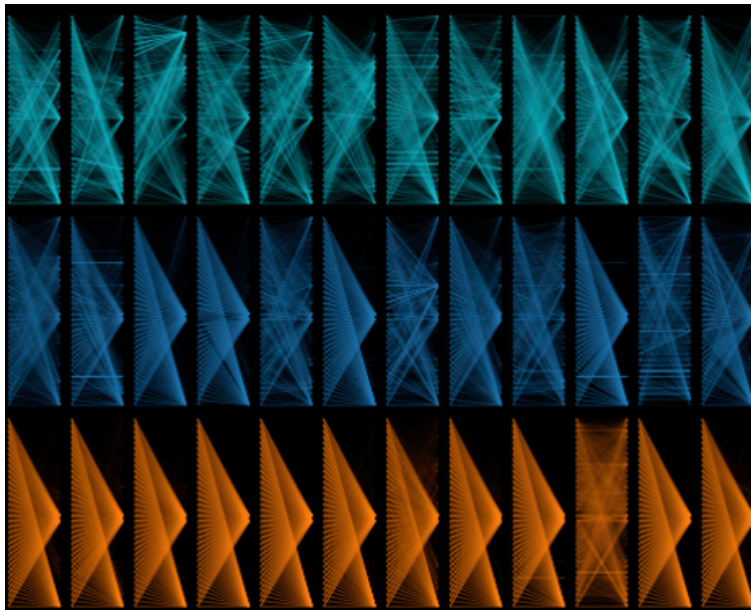


Figure 4.6: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.05$) (3.7) on an MRPC sentence pair.

At $\lambda = 0.05$, as shown in Figure 4.6, the model begins to display more structured attention compared to the cross-entropy baseline. While some variability remains, attention heads show early signs of consistency. This suggests that a small degree of CMI maximization is already effective in steering the model toward more stable attention distributions. The CMI score at this setting is 1.040443, and the model achieves an F1-score of 0.897959.

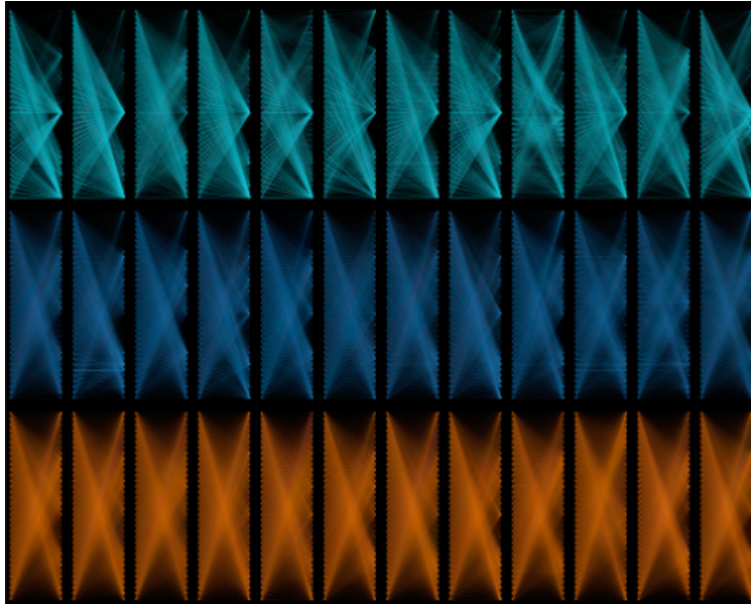


Figure 4.7: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.10$) (3.7) on an MRPC sentence pair.

With $\lambda = 0.10$, as shown in Figure 4.7, a more pronounced shift towards uniformity is observed. The attention heads exhibit reduced variability, indicating a more focused and stable distribution across input tokens. This increased stability suggests that maximizing CMI effectively guides the model toward more interpretable attention mechanisms. The CMI score rises to 1.402443, though the F1-score slightly decreases to 0.894737, implying a trade-off between structured attention and overall task performance.

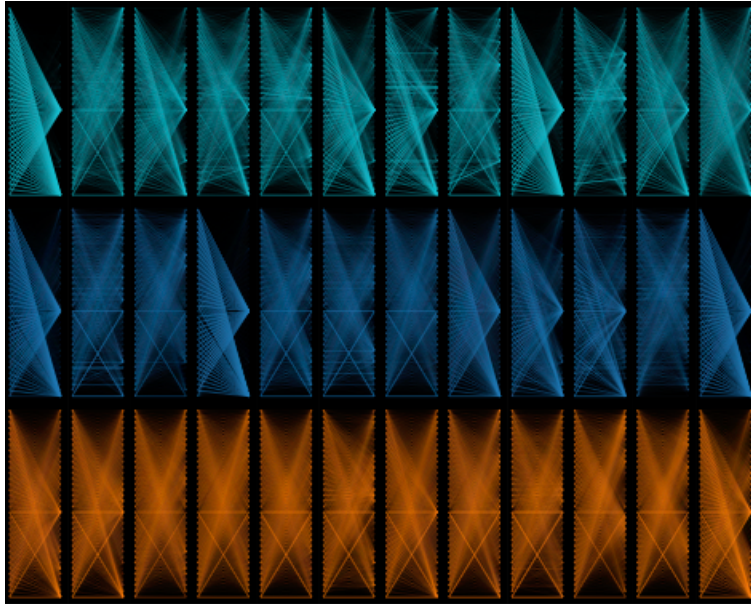


Figure 4.8: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.15$) (3.7) on an MRPC sentence pair.

Increasing λ to 0.15, as shown in Figure 4.8, attention heads in the last layer become highly aligned, displaying near-perfect consistency. This suggests that the CMI maximization objective is pushing the model towards rigid and deterministic attention patterns. While this alignment may enhance interpretability, it also reduces the diversity of attention heads, which could limit the model’s ability to capture nuanced relationships. The CMI score at this setting is 2.784619, while the F1-score declines to 0.840532, highlighting the potential drawbacks of excessive regularization.

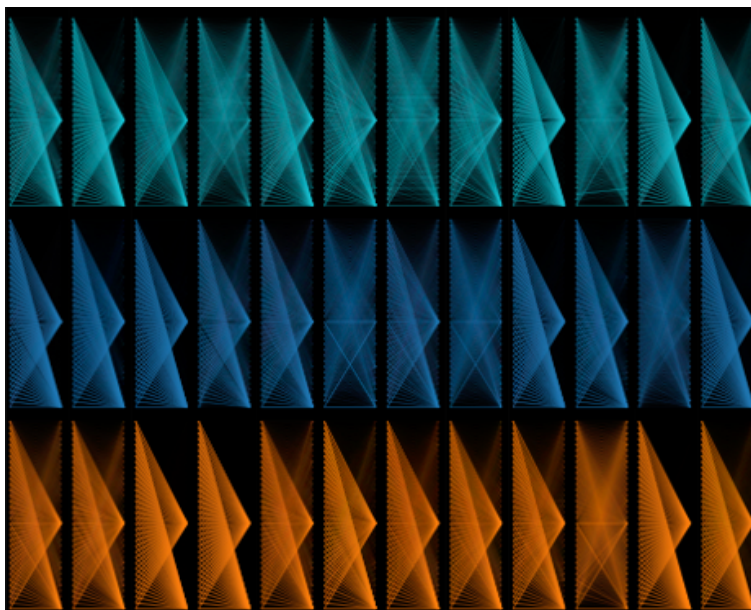


Figure 4.9: Attention patterns from the last three layers of a BERT-base model fine-tuned with CMI maximization ($\lambda = 0.25$) (3.7) on an MRPC sentence pair.

Finally, at $\lambda = 0.25$, as shown in Figure 4.9, the attention heads exhibit extreme uniformity, converging toward highly deterministic patterns even in earlier layers. While this enforces strong structural consistency, it also leads to over-regularization, where the model’s focus becomes overly narrow, potentially missing critical contextual information. This is reflected in the sharp increase in the CMI score to 5.483474, alongside a significant drop in F1-score to 0.815789, indicating that excessive structure may hinder task performance.

These results demonstrate that while maximizing CMI enhances attention consistency, excessive regularization can negatively impact generalization. Striking a balance is crucial to leveraging the benefits of structured attention without sacrificing model performance.

4.3.4 Summary of Attention Pattern Comparisons

The attention patterns observed across the different training objectives reveals a spectrum, ranging from highly scattered to rigidly structured. At one end of this spectrum lies the standard cross-entropy training. As shown in Figure 4.1, this objective yields scattered attention distributions, which characterizes a broad exploration of the input space. While

this approach allows the model to capture a wide range of linguistic features, it can also introduce noise and hinder the development of strong, focused representations.

Moving along the spectrum, we find that minimizing CMI, as depicted in Figures 4.2-4.5, offers a hybrid approach. This objective refines attention by reducing unnecessary variability while still preserving a degree of diversity. The resulting attention patterns exhibit localized concentration on tokens but still contain some variability across the input. This balance makes CMI minimization particularly well-suited for standalone model optimization as it allows the model to learn to focus on important tokens along with the tokens around them, leading to improved performance.

At the opposite end of the spectrum, maximizing CMI, as illustrated in Figures 4.6-4.9, enforces highly structured attention. This objective promotes consistency and focus across attention heads, leading to clearly defined and interpretable attention patterns. While this rigidity enhances KD by providing a clear signal for student models to mimic, it comes at the potential cost of over-regularization. As the CMI maximization strength increases, the model’s attention becomes overly deterministic, potentially hindering its ability to adapt to variations in the input and capture complex dependencies.

In summary, the choice of training objective dictates the nature of the learned attention patterns. Cross-entropy training prioritizes broad coverage at the expense of focus. CMI minimization seeks a balance between diversity and concentration, making it ideal for standalone model performance. CMI maximization emphasizes structure and interpretability, benefiting KD but risking over-regularization.

Chapter 5

Conclusion

In this thesis, we established a novel connection between LLMs and IT by demonstrating that transformer decoder-based LLMs can be mathematically described as computable high-order homogeneous Markov source models. Using this connection, we then applied another information-theoretic concept, the information-theoretic principle of Conditional Mutual Information (CMI), to enhance the performance of LLMs, particularly for classification tasks.

By minimizing CMI during fine-tuning, we improved the standalone performance of LLMs by tightening intra-class concentration and aligning predictions more closely with class centroids. This process reduced the influence of irrelevant contextual dependencies, enabling the model to focus on task-relevant features. In contrast, maximizing CMI during knowledge distillation allowed teacher models to encode and transfer richer contextual relationships to student models. This approach resulted in significant improvements in the student’s performance, as evidenced by gains on the GLUE benchmark, highlighting the potential of CMI-based optimization in improving knowledge transfer.

Future work could explore extending the CMI framework beyond classification tasks to areas such as text generation, multi-modal learning, and domain-specific applications. Additionally, applying CMI principles during the pretraining phase of LLM development or integrating these methods into multi-objective optimization frameworks may unlock further advancements in model efficiency, interpretability, and performance. These extensions offer promising directions for broadening the impact of information-theoretic approaches in NLP and other fields.

References

- [1] Sourya Basu, Moulik Choraria, and Lav R. Varshney. Transformers are universal predictors. In *ICML 2023 Workshop Neural Compression: From Information Theory to Applications*, 2023.
- [2] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995.
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- [4] Cristian Bucil, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 535–541. ACM, 2006.
- [5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 2nd edition, 2006.
- [6] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression. In *Proceedings of the 2024 International Conference on Learning Representations (ICLR)*, 2024.
- [7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill

- Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [8] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.
- [9] OpenAI et al. Gpt-4 technical report, 2024.
- [10] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, February 1994.
- [11] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically, 2017.
- [12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.
- [13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
- [14] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA, 2022. Curran Associates Inc.
- [15] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- [16] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online, November 2020. Association for Computational Linguistics.

- [17] J.C. Kieffer and En-Hui Yang. Grammar-based codes: a new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.
- [18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [19] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.
- [20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
- [21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.
- [22] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [23] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.
- [24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

- [26] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [27] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.
- [28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.
- [29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.
- [30] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning representations by back-propagating errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988.
- [31] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference, 2023.
- [32] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proceedings of the NeurIPS 2019 Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.
- [33] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.
- [34] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.
- [35] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

- [36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.
- [37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.
- [39] Jesse Vig. A multiscale visualization of attention in the transformer model. In Marta R. Costa-jussà and Enrique Alfonseca, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy, July 2019. Association for Computational Linguistics.
- [40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [41] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 4, COLING '92*, page 1106–1110, USA, 1992. Association for Computational Linguistics.

- [42] 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.
- [43] En-Hui Yang and Dake He. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform .2. with context models. *IEEE Transactions on Information Theory*, 49(11):2874–2894, 2003.
- [44] En-Hui Yang and J.C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. i. without context models. *IEEE Transactions on Information Theory*, 46(3):755–777, 2000.
- [45] En-Hui Yang, Shayan Mohajer Hamidi, Linfeng Ye, Renhao Tan, and Beverly Yang. Conditional mutual information constrained deep learning for classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–13, 02 2025.
- [46] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.
- [47] 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 (ICLR)*, 2024.

APPENDICES

Appendix A

GLUE Benchmark

For our experiments, we utilized all datasets from the GLUE benchmark [40], excluding the regression dataset STS-B, as it is not relevant to this work. The GLUE benchmark is a widely-used collection of datasets designed to evaluate various aspects of natural language understanding. Below, we provide a detailed description of each classification dataset, its task objective, and the primary evaluation metric:

- **CoLA (The Corpus of Linguistic Acceptability):** This dataset focuses on the binary classification task of assessing the linguistic acceptability of a given sentence. The evaluation metric is the Matthews Correlation Coefficient (MCC), which accounts for imbalanced classes and provides a robust measure of performance for this dataset.
- **MNLI (Multi-Genre Natural Language Inference):** A three-label classification task where the objective is to identify the relationship between a premise and a hypothesis as either entailment, neutral, or contradiction. The dataset includes examples from multiple genres, making it a challenging and diverse task. The evaluation metric is accuracy.
- **MRPC (Microsoft Research Paraphrase Corpus):** A binary classification task designed to determine whether two sentences in a pair are semantically equivalent. The dataset is particularly challenging due to the subtle semantic differences between sentence pairs. The evaluation metric is the F1-score, which balances precision and recall.

- **QNLI (Question Natural Language Inference):** This binary classification task involves determining whether a given sentence provides an answer to a question. Derived from the Stanford Question Answering Dataset (SQuAD), this task requires understanding of context and reasoning. The evaluation metric is accuracy.
- **QQP (Quora Question Pairs):** This binary classification task focuses on identifying whether two questions have the same meaning. The dataset is derived from Quora, featuring diverse linguistic variations in question phrasing. The evaluation metric is the F1-score.
- **RTE (Recognizing Textual Entailment):** A binary classification task aimed at determining whether a hypothesis is entailed by a premise. The dataset is relatively small compared to other GLUE tasks, posing additional challenges for model training. The evaluation metric is accuracy.
- **SST-2 (Stanford Sentiment Treebank):** A sentiment analysis dataset for binary classification, where sentences are labeled as expressing either positive or negative sentiment. This task emphasizes the model's ability to infer sentiment from nuanced textual cues. The evaluation metric is accuracy.
- **WNLI (Winograd Natural Language Inference):** A binary classification task based on the Winograd Schema Challenge, designed to test coreference resolution and common-sense reasoning. The goal is to determine whether a sentence is an entailment of another. The evaluation metric is accuracy.

The GLUE benchmark spans a variety of natural language understanding tasks, including linguistic acceptability, paraphrase detection, sentiment analysis, and natural language inference. By incorporating datasets with diverse objectives and challenges, it serves as a comprehensive evaluation suite for assessing the performance of natural language processing models. These datasets allow us to rigorously test the effectiveness of our proposed framework in enhancing model performance across multiple facets of natural language understanding.

Appendix B

Hyperparameters

In this section, we provide a detailed overview of the hyperparameters used for training the LLMs for both the minimizing and maximizing CMI methods. The training procedures for these methods are outlined in Sections [4.1](#) and [4.2](#), respectively.

B.1 Minimizing CMI

Table [B.1](#) summarizes the hyperparameters used for training the BERT-base model in the CMI minimization experiments. The optimal λ values used for minimizing CMI were determined empirically for each GLUE dataset by evaluating the model’s performance across different configurations and selecting the value that achieved the highest evaluation metric. Table [B.2](#) summarizes the chosen λ values for each dataset.

Table B.1: Hyperparameters of BERT-base experiments for minimizing CMI.

Hyperparameter	BERT-base
Number of Layers	12
Hidden Size	768
FFN Inner Hidden Size	3072
Attention Heads	12
Dropout Rate	0.1
Attention Dropout Rate	0.1
Warmup Steps	0
Batch Size	32
Optimizer	Adam
Starting Learning Rate	5×10^{-5}
Number of Epochs	5
Runs per Experiment	3

Table B.2: Optimal λ values for minimizing CMI across GLUE datasets.

Dataset	Optimal λ
CoLA	0.25, 0.4
MNLI	0.25
MRPC	0.4, 0.5
QNLI	0.15, 0.25, 0.4
QQP	0.15, 0.4
RTE	0.25
SST-2	0.65, 0.75
WNLI	0.05, 0.1

B.2 Maximizing CMI

To summarize the hyperparameter configurations for maximizing CMI, we provide the following tables. Table B.3 lists the general hyperparameters for training teacher (BERT-base) and student (DistilBERT) models. Table B.4 specifies the optimal λ values for teacher fine-tuning across GLUE datasets, while Table B.5 presents the optimal α and temperature (T) values used during knowledge distillation.

Table B.3: Hyperparameters of BERT-base (teacher) and DistilBERT (student) experiments for maximizing CMI.

Hyperparameter	BERT-base	DistilBERT
Number of Layers	12	6
Hidden Size	768	768
FFN Inner Hidden Size	3072	3072
Attention Heads	12	12
Dropout Rate	0.1	0.1
Attention Dropout Rate	0.1	0.1
Warmup Steps	0	0
Batch Size	32	32
Optimizer	Adam	Adam
Starting Learning Rate	5×10^{-5}	5×10^{-5}
Number of Epochs	5	10
Runs per Experiment	3	3

Table B.4: Optimal λ values for maximizing CMI for teacher models across GLUE datasets.

Dataset	Optimal λ
CoLA	0.05
MNLI	0.05, 0.25
MRPC	0.15, 0.40
QNLI	0.10, 0.15, 0.25
QQP	0.05, 0.1
RTE	0.10, 0.15
SST-2	0.05, 0.10, 0.15
WNLI	0.05, 0.15, 0.25

Table B.5: Optimal α and T values for KD across GLUE datasets for CMI maximization experiments.

Dataset	α	T
CoLA	0.15, 0.25	1, 2
MNLI	0.65, 0.75, 0.90	1, 2, 3
MRPC	0.75, 0.9	3, 4, 5
QNLI	0.75	4, 5
QQP	0.90	2, 3
RTE	0.65, 0.75, 0.90	4, 5
SST-2	0.4, 0.5	3, 4
WNLI	0.25, 0.4	4, 5

Appendix C

Interpreting Attention Pattern Visualizations

This appendix explains the visual representation of attention patterns from a BERT-base model [7] created from BERTViz [39], specifically focusing on the final layers of the model. Figure C.1 is an example image that depicts the attention patterns across multiple layers and attention heads in the BERT-base model.

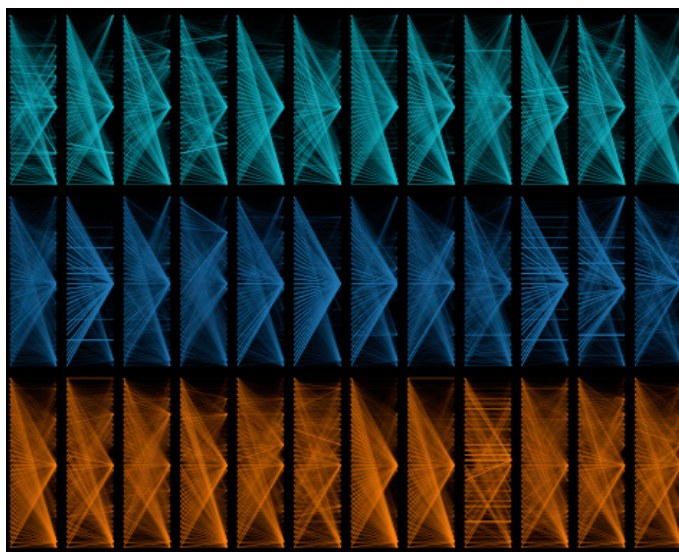


Figure C.1: Example image of attention patterns from a BERT-base model. This image is identical to Figure 4.1.

Each column in the image represents an attention head from one of the final layers of the BERT model. In the provided example, the bottom-most row corresponds to the last layer of the model, and each row above it represents earlier layers in the model.

The structure of each box represents the relationship between tokens in the input sequence. Each token is represented on both edges of the box, and the lines within the box indicate the attention mechanism's behavior. The thickness of these lines corresponds to the strength of attention that one token gives to another where thicker lines indicate stronger attention.

The attention heads capture different aspects of the input sequence's relationships, with each attention head focusing on different patterns of dependencies between tokens. These patterns highlight the flow of information across the layers of the model, showing how different tokens influence each other's embeddings based on the learned attention weights. This visual representation allows us to observe how the model focuses on various aspects of the input, such as syntactic and semantic relationships, during the final layers of its processing pipeline.