How Bidirectionality Helps Language Models Learn Better via Dynamic Bottleneck Estimation

Md Kowsher[1]*, **Nusrat Jahan Prottasha**[1]*, **Shiyun Xu**[2], **Shetu Mohanto**[3],
Chen Chen[1], **Ozlem Garibay**[1], **Niloofar Yousefi**[1]
[1]University of Central Florida [2]University of Pennsylvania [3]Delineate Inc.
github.com/Kowsher/BidiVsUniLM

Abstract

Bidirectional language models have better context understanding and perform better than unidirectional models on natural language understanding tasks, yet the theoretical reasons behind this advantage remain unclear. In this work, we investigate this disparity through the lens of the Information Bottleneck (IB) principle, which formalizes a trade-off between compressing input information and preserving task-relevant content. We propose FlowNIB, a dynamic and scalable method for estimating mutual information during training that addresses key limitations of classical IB approaches, including computational intractability and fixed trade-off schedules. Theoretically, we show that bidirectional models retain more mutual information and exhibit higher effective dimensionality than unidirectional models. To support this, we present a generalized framework for measuring representational complexity and prove that bidirectional representations are strictly more informative under mild conditions. We further validate our findings through extensive experiments across multiple models and tasks using FlowNIB, revealing how information is encoded and compressed throughout training. Together, our work provides a principled explanation for the effectiveness of bidirectional architectures and introduces a practical tool for analyzing information flow in deep language models.

1 Introduction

Large language models have brought significant advancements in natural language understanding (NLU) tasks. Among them, bidirectional models such as BERT have demonstrated superior performance in natural language understanding, while unidirectional models like GPT dominate generation tasks. As shown in Table 1 of [11], the BERT-base model outperforms GPT [33] across all GLUE benchmarks [46] despite having a comparable model size – for example, achieving 66.4% accuracy on the RTE task versus GPT's 56.0%. Moreover, both theoretical and empirical evidence [23, 24, 35, 8] consistently demonstrate that bidirectional models outperform unidirectional models on a wide range of NLU tasks.

While the empirical advantage of bidirectional models is well-documented, the theoretical explanation for this phenomenon remains limited. To address this, we adopt an information-theoretic perspective grounded in the Information Bottleneck (IB) principle [43]. Let $I(X;Z)$ denote the mutual information between input X and an intermediate representation Z and $I(Z;Y)$ the mutual information between Z and output label Y. At the core, IB seeks representations that compress the input information $I(X;Z)$ while preserve task-specific content $I(Z;Y)$. This trade-off is visualized through the information plane, which plots $I(X;Z)$ and $I(Z;Y)$ over training epochs and across network layers.

*Equal contribution

Preprint. Under review.

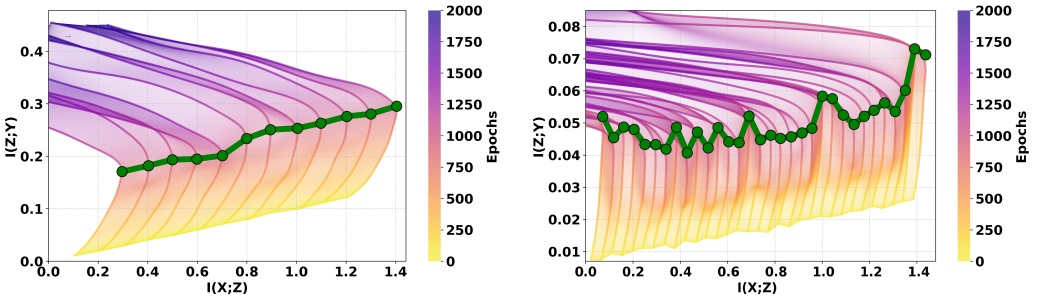

Figure 1: Information plane trajectories under FlowNIB training for (left) DeBERTaV3-Base and (right) MobileLLM-350M on MRPC. Each curve shows mutual information $I(Z;Y)$ vs. $I(X;Z)$ over training epochs, colored by epoch progression. We added a constant offset of $+0.05$ to $I(X;Z)$ at each successive layer to visually separate layer-wise trajectories. The green line represents Mutual Information Coordinate (MIC).

To make the IB analysis applicable to LLMs, we formalize the following:

Definition 1.1 (A valid Information Plane)**.** Let a deep neural network consist of L hidden layers, with each layer output denoted by Z_ℓ for $\ell = 1, \ldots, L$. Let X be the input and Y be the target label. Define the information plane as the set of mutual information tuples:

$$\mathcal{I}_{\text{plane}} := \{(I(X;Z_\ell),\ I(Z_\ell;Y)) \mid \ell = 1, \ldots, L\}\,.$$

We say that the information plane $\mathcal{I}_{\text{plane}}$ is *well-defined* if it satisfies the following properties: (1) **Layerwise Resolution:** Each hidden layer Z_ℓ yields a unique and distinguishable point on the plane. (2) **Temporal Trajectories:** For each training epoch or iteration t, the tuple $\left(I^{(t)}(X;Z_\ell), I^{(t)}(Z_\ell;Y)\right)$ is well-defined and continuously trackable. (3) **Mutual Information Coordinate Optimality:** Each point $(I(X;Z_\ell), I(Z_\ell;Y))$ is called a *Mutual Information Coordinate (MIC)*. A coordinate is considered high-quality when both mutual information values are simultaneously large, indicating that the representation Z_ℓ retains rich input information and aligns well with the prediction target. (4) **Estimator Stability:** The mutual information quantities are estimated using consistent and differentiable approximations (e.g., variational bounds or neural MI estimators) that are robust to high-dimensional noise. (5) **Phase Interpretability** The plane exhibits a distinguishable two-phase dynamic:- (i) **Fitting Phase:** Both $I(X;Z_\ell)$ and $I(Z_\ell;Y)$ increase as the network memorizes input-label associations. (ii) **Compression Phase:** $I(X;Z_\ell)$ decreases while $I(Z_\ell;Y)$ continues to increase, indicating abstraction and denoising.

While recent works [2, 30, 1] have applied IB principle to improve neural network training, [41, 6] employed IB as a visualization tool to uncover the underlying training dynamics of deep networks. However, challenges remain in applying IB to large language models: 1) **computational bottleneck** in estimating mutual information $I(X;Z)$ and $I(Z;Y)$ prevents from tracking the information flow during training; 2) **static trade-off control**: the IB Lagrangian uses a fixed parameter β fails to adapt as training phase transits from input memorization to predictive sufficiency.

To address the estimation and adaptivity limitations, we introduce the Flow Neural Information Bottleneck (FlowNIB), a dynamic framework that facilitates both accurate estimation and principled interpretation of information dynamics in deep language models. FlowNIB employs a time-varying weighting function $\alpha(t)$ that gradually transits the training objective from memorization to prediction.

Our contributions are threefold: **(1)** We introduce a novel and computationally efficient method for **estimating IB dynamics during training**. i) we employs Mutual Information Neural Estimator (MINE) [4] to efficiently approximate $I(X;Z)$ and $I(Z;Y)$ in high-dimentional settings. ii) we proposes a novel dynamic normalized IB objective that progressively transitions from input memorization ($I(X;Z)$) to predictive sufficiency ($I(Z;Y)$), enabling scalable and stable mutual information tracking in deep networks. **(2)** We present **a theoretical proof for our estimator consistency** and **an empirical explanation for the advantages of bidirectional language models** by information plane. As shown in Figure 1, the information flow of bidirectional language models (left) is higher compared to unidirectional models (right). **(3)** We modified PredGen [19] to perform **single-token generation**

to evaluate the model with higher prediction quality and efficiency and conducted comprehensive ablation studies on the effect of different scheduler α, dimension of intermediate data Z and output Y, the complexity of output. It shows our method is stable in various settings.

Applying FlowNIB to large language models yields a well-formed information plane that reveals distinct layerwise behaviors, enables stable mutual information estimation over time, and captures interpretable training dynamics—characterized by an initial fitting phase followed by progressive compression. Within this framework, we empirically demonstrate that bidirectional models follow a more efficient information-theoretic training trajectory: they achieve greater predictive gain and a more favorable compression–relevance trade-off, resulting in superior positioning and evolution on the information plane. This, in turn, helps explain their empirically observed advantages in natural language understanding tasks.

2 Methodology

Unidirectional language models, such as GPT, construct each hidden representation using only left-to-right context [3]. In contrast, bidirectional models like BERT encode each token using both past and future context [15, 24]. This architectural asymmetry raises a fundamental question: do bidirectional representations preserve more information about the input?

Let X denote the input sequence and $Z^{\rightarrow}$ and $Z^{\leftarrow}$ the forward and backward representations, respectively. The unidirectional model produces representations $Z^{\rightarrow} = (z_1^{\rightarrow}, \ldots, z_n^{\rightarrow})$, where $z_t^{\rightarrow}$ depends only on $x_{\leq t}$. The bidirectional model augments this with backward representations $Z^{\leftarrow} = (z_1^{\leftarrow}, \ldots, z_n^{\leftarrow})$, where $z_t^{\leftarrow}$ depends only on $x_{\geq t}$. We define the full bidirectional representation as $Z^{\leftrightarrow} = (Z^{\rightarrow}, Z^{\leftarrow})$. We measure representational quality via mutual information:

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

where $H(X \mid Z)$ is the conditional entropy of the input given the latent representation. Because $Z^{\leftrightarrow}$ includes strictly more context than $Z^{\rightarrow}$, it can better reduce uncertainty about X.

Theorem 2.1 (Full version in Appendix A.3)**.** *Bidirectional representations preserve more mutual information about the input and output:*

$$I(X; Z^{\leftrightarrow}) \geq I(X; Z^{\rightarrow}), I(Z^{\leftrightarrow}; Y) \geq I(Z^{\rightarrow}; Y),$$

This result follows directly from the monotonicity of conditional entropy: conditioning on more information reduces entropy (Theorem A.2). Therefore, bidirectional models produce latent representations that retain more information about the input sequence. In Section 3, we empirically validate this claim using FlowNIB framework 2.

While mutual information quantifies how much information a representation preserves about the input or output, it does not reveal the internal structure or complexity of the representation itself. To complement this perspective, we analyze the spectral properties of latent representations through the lens of *effective dimensionality*, which captures how many orthogonal directions in representation space carry significant variance. This helps us understand how richly a model encodes information at each layer.

Definition 2.2 (Generalized Effective Dimensionality)**.** Let $\lambda_1, \ldots, \lambda_n$ be the non-zero eigenvalues of the covariance matrix of a representation Z, and define the normalized spectrum $p_i := \lambda_i / \sum_j \lambda_j$. The generalized effective dimensionality of Z under a measure $\mathcal{M}(p)$ is defined as $d_{\text{eff}}(Z; \mathcal{M}) := \exp(\mathcal{M}(p))$, where $\mathcal{M}(p)$ satisfies three properties: (i) non-negativity: $\mathcal{M}(p) \geq 0$; (ii) maximality: $\mathcal{M}(p) \leq \log n$, with equality iff $p_i = 1/n$; and (iii) Schur-concavity: $\mathcal{M}(p') \leq \mathcal{M}(p)$ if $p' \succ p$.

Examples: (1) Shannon entropy: $\mathcal{M}(p) = -\sum_i p_i \log p_i$ yields the standard effective rank $d_{\text{eff}}(Z) = \exp(H(p))$; (2) ℓ_2-based participation ratio: $\mathcal{M}(p) = \log(1/\sum_i p_i^2)$ gives $d_{\text{eff}}(Z) = (\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$. This generalizes the classic effective rank definition from [36] and provides a flexible framework for quantifying spectral complexity. Unless otherwise stated, we adopt the ℓ_2-norm-based version as the default. The effect of alternative measures is explored in the ablation study C.

Lemma 2.3 (Bidirectional Representations Exhibit Higher Spectral Complexity)**.** *Let $Z^{\rightarrow} \in \mathbb{R}^D$ denote the unidirectional representation and $Z^{\leftrightarrow} := (Z^{\rightarrow}, Z^{\leftarrow}) \in \mathbb{R}^{2D}$ the concatenated bidirectional*

representation of an input X. Then, if $\mathrm{Cov}(Z^{\leftarrow}, Z^{\rightarrow})$ is non-singular,

$$d_{\text{eff}}(Z^{\leftrightarrow}; \mathcal{M}) \geq d_{\text{eff}}(Z^{\rightarrow}; \mathcal{M}),$$

with equality iff $Z^{\leftarrow}$ is conditionally redundant given $Z^{\rightarrow}$, i.e., $\mathrm{Cov}(Z^{\leftarrow} \mid Z^{\rightarrow}) = 0$.

See Appendix A.6 for the theoretical proof and Ablation C for empirical validation.

Key Finding

Bidirectional representations retain strictly more mutual information about the input than unidirectional representations. They also exhibit higher effective dimensionality throughout depth, reflecting richer and more expressive latent spaces.

Flow Neural Information Bottleneck: Classical Information Bottleneck (IB) methods [43] require estimating mutual information (MI) terms $I(X;Z)$ and $I(Z;Y)$, involving high-dimensional divergences such as $D_{\mathrm{KL}}(p(x,z)\|p(x)p(z))$ [4]. These computations are intractable in the latent spaces of deep neural networks, including large-scale language models. Furthermore, classical IB uses a static tradeoff parameter β, limiting flexibility during optimization.

To address these challenges, we propose **FlowNIB** (Flow Neural Information Bottleneck), a dynamic variant of IB that progressively transitions from input memorization to predictive sufficiency. FlowNIB minimizes:

$$\mathcal{L}(\theta; t) = -\Big(\alpha(t)\,\hat{I}(X;Z) + (1-\alpha(t))\,\hat{I}(Z;Y)\Big),$$

where $\alpha(t) : \mathbb{N} \to [0,1]$ is a discrete, monotonically decreasing scheduling function, and $\hat{I}$ denotes a neural variational estimate of mutual information, normalized by effective dimensionality for scalability [36].

Intuitively, FlowNIB traverses three phases during training: (1) an *expansion phase* (when $\alpha(t) \approx 1$) emphasizing $I(X;Z)$, leading Z to retain broad input information; (2) a *compression phase* as $\alpha(t) \to 0$, focusing on $I(Z;Y)$ and progressively filtering irrelevant input components; (3) a *flow alignment phase* where $I(X;Z)$ and $I(Z;Y)$ converge toward the information bottleneck frontier.

The decrement step $\delta := \alpha(t) - \alpha(t+1)$ plays a critical role in controlling the speed of this transition: (i) a small δ ($\delta \to 0$) yields a slow schedule, allowing Z to retain input features longer and delaying compression; (ii) a large δ accelerates the transition, potentially compressing Z prematurely before sufficient input structure has been learned.

This dynamic flow enables FlowNIB to balance memorization and predictive sufficiency without explicitly minimizing $I(X;Z)$, producing representations that are both compressed and predictive. Full derivations are provided in Appendix B.

Theorem 2.4 (Consistency under Optimal Critics)**.** *Let random variables $(X,Z) \sim p(x,z)$ and $(Z,Y) \sim p(z,y)$ define the Markov chain $X \to Z \to Y$. Suppose the critic functions T^*_{xz} and T^*_{zy} achieve the optimal solutions:*

$$T^*_{xz}(x,z) = \log \frac{p(x,z)}{p(x)p(z)}, \quad T^*_{zy}(z,y) = \log \frac{p(z,y)}{p(z)p(y)}.$$

Then, the dimension-normalized mutual information estimators used in the FlowNIB loss satisfy:

$$\hat{I}(X;Z) \xrightarrow{T_{xz} \to T^*_{xz}} \frac{I(X;Z)}{d_{\text{eff}}(Z)^2}, \quad \hat{I}(Z;Y) \xrightarrow{T_{zy} \to T^*_{zy}} \frac{I(Z;Y)}{d_{\text{eff}}(Y)^2},$$

and thus consistently estimate the normalized mutual information. Full proof in Appendix B.1.

Key Finding

We observe that the step size δ controlling the decay of $\alpha(t)$ critically governs the speed of transition from input memorization to predictive compression. **Small** δ delays compression,

preserving input features longer; **large** δ accelerates compression but risks underlearning input structure. Details in the Ablation Study C.

3 Experiments

In this section, we validate our theoretical findings by analyzing bidirectional and unidirectional language models using the FlowNIB framework. We evaluate them on both classification and regression tasks, as language models are used in both settings. This helps us better understand how well the learned representations support different types of predictions.

All models are fine-tuned using the RoCoFT parameter-efficient fine-tuning (PEFT) method [18]. RoCoFT updates only a few rows of the weight matrices without adding extra trainable parameters. This allows the model to preserve more information from the pretrained backbone while staying efficient. In our setup, we allow only 3 rows to be updated. Additional results using LoRA are reported in Appendix Table 4.

We focus on answering the following three research questions: (i) Do bidirectional models preserve more useful information than unidirectional models? (ii) Does higher mutual information lead to better context modeling? (iii) Does predicting a single token (e.g., masked token or next token) lead to better performance than traditional methods?

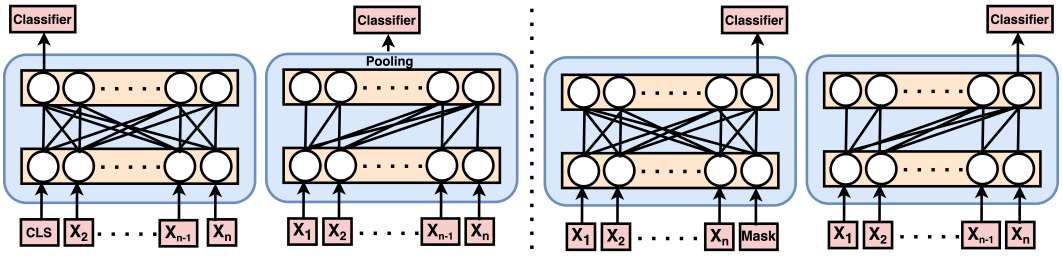

Figure 2: Illustration of representation extraction methods: (a) prediction from CLS-token (bidirectional), (b) prediction from pooled embedding (unidirectional), (c) prediction from masked token (bidirectional), and (d) prediction from next-token generation (unidirectional).

Model Framework: We compare bidirectional and unidirectional language models under a unified evaluation setup. While standard approaches apply a pooling operation over the final hidden states followed by a classifier, we adopt an alternative strategy inspired by the PredGen framework [19].

PredGen demonstrated that leveraging the model's generative capability—rather than relying solely on pooled representations—retains higher mutual information with the input, improving prediction quality. However, a key limitation of PredGen is the increased computational cost of multi-token generation, especially for regression-type tasks.

To address this, we introduce a modified framework that performs *single-token generation or masked prediction*, as illustrated in Figure 2 (Right). Specifically, the model predicts a single masked token at a designated position, from which we extract the corresponding final hidden state. This representation is then passed through a lightweight MLP classifier.

Key Finding

We illustrate a simplified variant of the PredGen framework that replaces multi-token generation with single-token generation or masked prediction. This approach achieves comparable performance to PredGen while substantially reducing inference cost and training complexity. See Appendix Table33 for the comparison between single token-based prediction and PredGen.

Datasets: We evaluate our models across 16 diverse NLP datasets spanning classification and regression tasks to ensure a comprehensive analysis of representational learning under the information bottleneck framework. For classification, we include **SST-2**, **MRPC**, **QNLI**, **RTE**, **MNLI**, and

Table 1: Overview of bidirectional (top) and unidirectional (bottom) model architectures evaluated in our experiments, including FLOPs and MACs.

Model	Layer	#Heads	Embedding Dim	Max Length	Vocab Size	Total Params	FLOPs	MACs
ModernBERT-base	22	12	768	8192	50368	149M	28.258	14.118
ModernBERT-large	28	16	1024	8192	50368	395M	87.883	43.923
RoBERTa-base	12	12	768	514	50265	125M	21.76	10.87
RoBERTa-large	24	16	1024	514	50265	355M	77.344	38.656
DeBERTa-v3-base	12	12	768	512	128100	184M	39.275	19.629
DeBERTa-v3-large	24	16	1024	512	128100	435M	136.943	68.451
GPT2-small	12	12	768	1024	50257	117M	21.756	10.872
GPT2-medium	24	16	1024	1024	50257	345M	77.342	38.655
GPT2-large	36	20	1280	1024	50257	762M	181.254	90.597
SmolLM-135M	30	9	576	2048	49152	135M	27.185	13.59
SmolLM-360M	32	15	960	2048	49152	360M	80.541	40.265
MobileLLM-125M	30	9	576	2048	32000	125M	31.9	15.95
MobileLLM-600M	40	18	1152	2048	32000	600M	154.408	77.196

CoLA from the GLUE benchmark [46], as well as **BoolQ** [7], **HellaSwag** [48], and **SocialIQA** [37], covering a range of linguistic challenges such as sentiment analysis, natural language inference, grammatical acceptability, question answering, and commonsense reasoning. The regression tasks comprise **STS-B** [5], **SICK** [26], **WASSA** [45], **LCP** [39], **CRP** [39], and **Humicroedit** [17], addressing semantic textual similarity, lexical complexity prediction, and humor detection. Dataset sizes range from approximately 2,500 to 400,000 examples, with either binary or multi-class classification labels, or continuous-valued targets for regression. We exclude generation-based datasets because bidirectional language models are not designed for autoregressive generation; instead, we focus on tasks requiring strong contextual representations to assess representational sufficiency under the information bottleneck. Additional dataset statistics are provided in Table 5 in the Appendix. In addition, the details of used models architecture, hyperparameters, evaluation metrics, and environment setup are provided in Appendix H, Appendix I, Appendix G, and Appendix F, respectively.

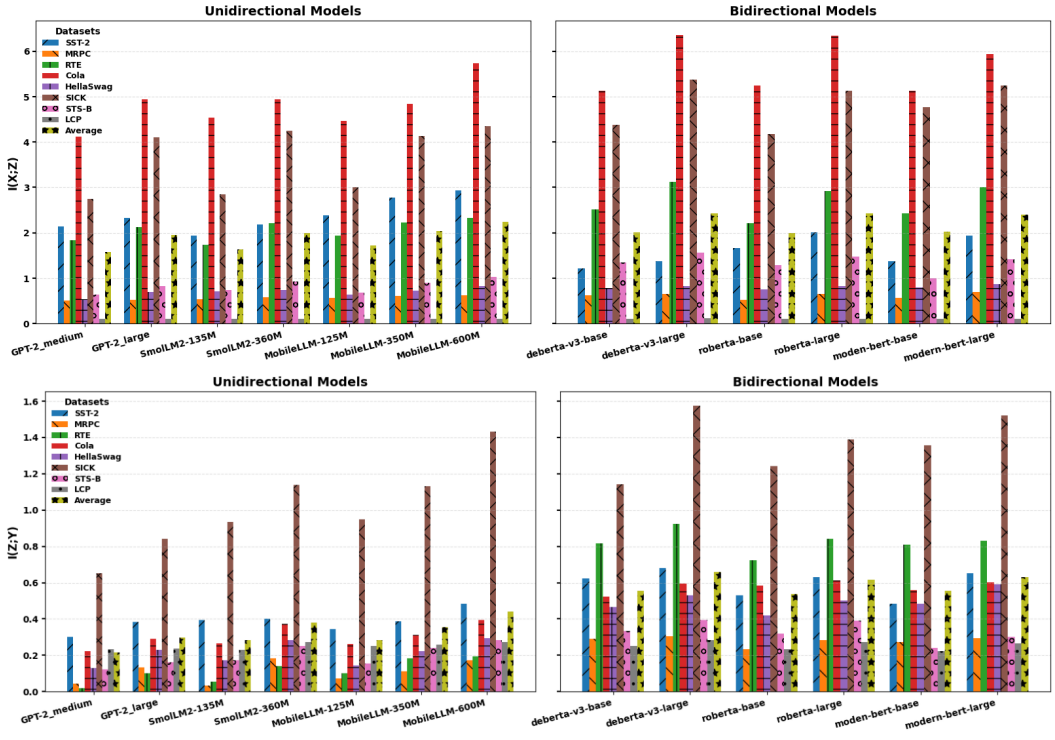

Figure 3: Mutual information $I(X;Z)$ (top) and $I(Z;Y)$ (bottom) across unidirectional (left) and bidirectional (right) models over multiple datasets. Bars show dataset-wise and average values, comparing information flow differences between architectures.

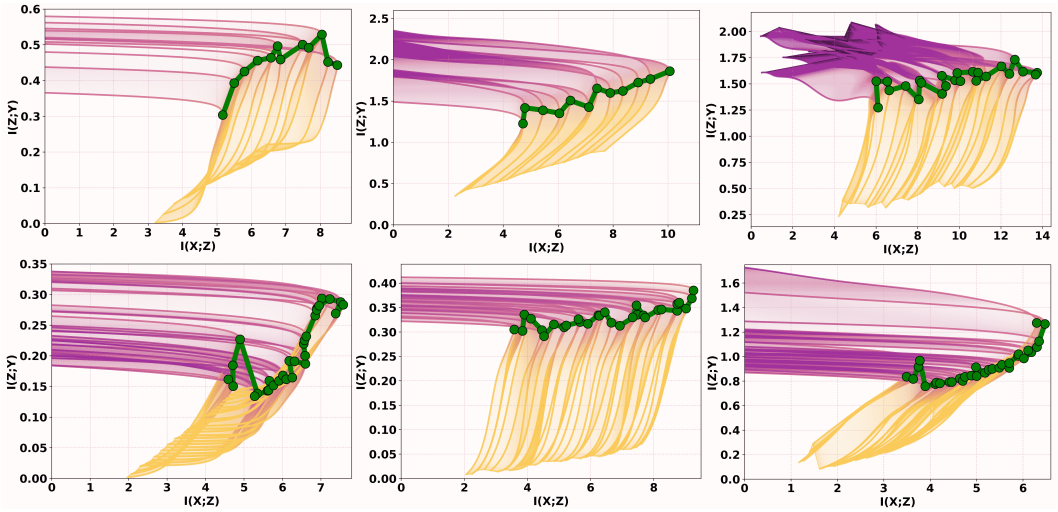

Figure 4: Mutual information flow comparison between bidirectional (top) and unidirectional (bottom) models across three datasets. The first column shows results on the CoLA dataset using DeBERTa-base and MobileLLM-350M. The second column shows SST-2 results using RoBERTa-base and MobileLLM-350M. The third column presents results on the SICK dataset using DeBERTa-v3-Large and MobileLLM-600M.

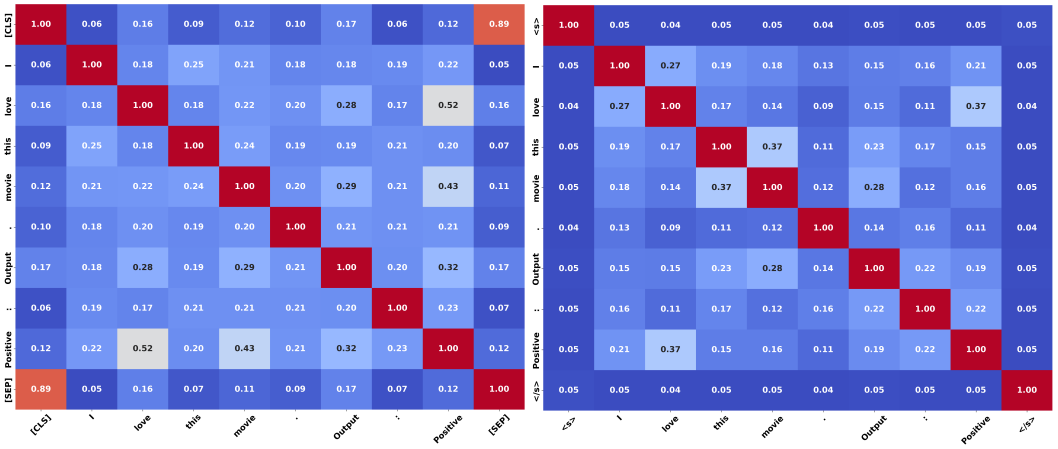

Figure 5: Token-level mutual information matrix on the SST-2 dataset for sentiment classification, computed from the final hidden layer representations. (Left) RoBERTa-base; (Right) SmallM2-360

Table 2: Accuracy(%) results across nine NLP classification tasks comparing bidirectional and unidirectional models under pooling, masking, and generation inference strategies.

Model	Method	SST-2	MRPC	QNLI	RTE	CoLA	MNLI	BoolQ	HellaSwag	SIQA	Avg.
DeBERTa-v3-Base	Pooling	95.52	89.21	92.43	83.48	86.23	86.43	64.23	56.00	47.54	77.90
	Masking	95.75	91.17	92.48	84.98	87.44	87.22	64.23	69.49	60.90	81.52
DeBERTa-v3-Large	Pooling	95.67	93.45	93.58	88.38	93.34	90.76	64.73	57.34	51.43	80.96
	Masking	96.11	94.04	94.14	89.93	92.95	91.43	64.98	73.43	65.53	84.73
RoBERTa-Base	Pooling	94.24	84.53	91.96	83.45	86.34	86.34	63.82	52.43	45.64	76.53
	Masking	95.14	85.13	92.27	84.58	87.44	86.38	63.96	64.53	60.16	79.95
RoBERTa-Large	Pooling	95.68	89.54	94.17	86.32	93.85	90.87	64.82	57.35	48.69	80.14
	Masking	96.23	91.25	94.38	87.84	95.83	91.13	63.82	71.43	63.67	83.95
ModernBERT-Base	Pooling	94.35	83.33	91.98	82.81	84.92	87.44	63.70	55.32	46.81	76.74
	Masking	95.38	85.43	92.43	84.12	84.43	88.21	62.17	63.54	61.86	79.73
ModernBERT-Large	Pooling	95.37	89.43	94.22	86.74	89.95	93.23	64.22	60.32	49.67	80.35
	Masking	95.89	89.93	94.57	87.78	90.79	92.98	64.72	73.18	64.68	83.84
GPT-2 Medium	Pooling	93.80	85.78	91.17	69.67	80.24	78.81	63.43	37.83	38.45	71.02
	Generation	94.14	85.93	91.93	69.83	81.43	80.18	63.54	37.93	43.45	72.04
GPT-2 Large	Pooling	93.97	86.27	84.01	66.78	83.89	80.06	64.13	40.32	41.91	71.26
	Generation	94.24	87.23	84.56	67.34	83.87	82.34	64.16	39.53	45.34	72.07
SmolLM2-135M	Pooling	92.58	84.59	90.56	68.12	81.48	82.83	62.43	38.34	41.41	71.37
	Generation	93.00	84.83	90.68	68.93	82.48	83.58	62.27	41.78	47.86	72.82
SmolLM2-360M	Pooling	94.26	84.80	91.61	70.70	82.07	85.12	63.13	42.45	42.43	72.95
	Generation	94.65	85.32	92.32	71.11	84.53	84.89	62.92	43.69	50.20	74.40
MobileLLM-125M	Pooling	93.05	82.43	90.58	69.32	80.29	82.98	60.73	33.45	41.45	70.48
	Generation	93.15	83.35	90.54	69.53	80.53	83.24	61.26	37.42	48.23	71.92
MobileLLM-350M	Pooling	93.85	83.68	90.85	70.33	82.38	83.45	63.42	36.28	42.74	71.89
	Generation	94.68	83.57	91.09	71.43	82.87	84.58	63.71	40.13	51.54	73.73
MobileLLM-600M	Pooling	94.86	87.34	91.34	72.45	84.56	84.93	64.18	45.32	45.54	74.50
	Generation	95.14	87.87	91.37	72.29	86.30	84.79	64.12	48.53	58.54	76.55

Table 3: Regression results (MSE/MAE) across six NLP regression tasks comparing bidirectional and unidirectional models under pooling, masking, and generation inference strategies.

Model	Method	WASSA	SICK	STSB	LCP	CRP	Humicroedit	Avg.
DeBERTa-v3-Base	Pooling	0.017/0.107	0.163/0.297	0.363/0.455	0.007/0.076	0.429/0.518	0.278/0.432	0.209/0.314
	Masking	0.013/0.091	0.135/0.277	0.373/0.462	0.006/0.060	0.385/0.478	0.274/0.423	0.197/0.298
DeBERTa-v3-Large	Pooling	0.016/0.102	0.140/0.281	0.353/0.442	0.007/0.073	0.345/0.457	0.263/0.419	0.187/0.295
	Masking	0.012/0.075	0.132/0.274	0.348/0.414	0.005/0.051	0.340/0.459	0.268/0.421	0.184/0.282
RoBERTa-Base	Pooling	0.016/0.097	0.168/0.300	0.364/0.452	0.007/0.066	0.465/0.535	0.293/0.438	0.218/0.314
	Masking	0.015/0.094	0.145/0.294	0.353/0.448	0.007/0.065	0.431/0.517	0.289/0.431	0.206/0.308
RoBERTa-Large	Pooling	0.015/0.097	0.153/0.291	0.351/0.439	0.006/0.060	0.376/0.469	0.283/0.432	0.197/0.298
	Masking	0.016/0.099	0.152/0.291	0.350/0.429	0.603/0.059	0.366/0.475	0.281/0.431	0.294/0.297
ModernBERT-Base	Pooling	0.016/0.092	0.207/0.350	0.469/0.517	0.006/0.069	0.376/0.469	0.302/0.447	0.229/0.324
	Masking	0.015/0.093	0.173/0.328	0.482/0.536	0.006/0.067	0.364/0.471	0.281/0.430	0.220/0.320
ModernBERT-Large	Pooling	0.016/0.093	0.160/0.307	0.378/0.468	0.006/0.060	0.341/0.453	0.302/0.449	0.200/0.305
	Masking	0.150/0.294	0.150/0.292	0.371/0.462	0.006/0.005	0.344/0.457	0.293/0.441	0.219/0.325
GPT-2 Medium	Pooling	0.019/0.112	0.662/0.619	0.427/0.499	0.008/0.084	0.369/0.476	0.394/0.535	0.313/0.387
	Generation	0.018/0.111	0.673/0.620	0.412/0.490	0.008/0.083	0.345/0.457	0.347/0.493	0.300/0.375
GPT-2 Large	Pooling	0.018/0.105	0.623/0.583	0.442/0.522	0.007/0.080	0.324/0.443	0.318/0.463	0.288/0.366
	Generation	0.017/0.107	0.583/0.523	0.423/0.499	0.007/0.078	0.326/0.446	0.323/0.473	0.279/0.354
SmolLM2-135M	Pooling	0.017/0.105	0.192/0.336	0.424/0.489	0.007/0.076	0.369/0.476	0.304/0.450	0.218/0.322
	Generation	0.017/0.106	0.175/0.319	0.403/0.484	0.007/0.076	0.366/0.475	0.295/0.442	0.210/0.317
SmolLM2-350M	Pooling	0.017/0.104	0.173/0.310	0.407/0.488	0.006/0.061	0.340/0.459	0.338/0.463	0.213/0.314
	Generation	0.017/0.105	0.170/0.298	0.394/0.481	0.006/0.060	0.332/0.454	0.323/0.462	0.207/0.310
MobileLLM-125M	Pooling	0.020/0.111	0.197/0.354	0.419/0.492	0.006/0.070	0.323/0.446	0.302/0.451	0.211/0.320
	Generation	0.019/0.113	0.192/0.324	0.410/0.491	0.006/0.068	0.312/0.448	0.293/0.442	0.205/0.314
MobileLLM-350M	Pooling	0.018/0.104	0.191/0.336	0.394/0.482	0.006/0.063	0.310/0.436	0.282/0.431	0.200/0.308
	Generation	0.017/0.105	0.187/0.320	0.391/0.478	0.006/0.063	0.309/0.437	0.278/0.421	0.198/0.304
MobileLLM-600M	Pooling	0.017/0.105	0.181/0.320	0.384/0.474	0.006/0.063	0.301/0.432	0.274/0.421	0.193/0.302
	Generation	0.017/0.105	0.172/0.318	0.381/0.472	0.006/0.063	0.308/0.419	0.278/0.438	0.193/0.302

MI Results: To compute mutual information, we first extract the hidden representations from all layers after fine-tuning each model on the target dataset. We then train a simple two-layer neural network for each hidden layer representation Z, using the FlowNIB objective. This allows us to estimate both $I(X;Z)$ and $I(Z;Y)$ independently for every layer. The pseudocode is described in Algorithm 1.

Figure 3 compares the mutual information coordinate between the input and hidden representations $I(X;Z)$, and between the hidden representations and output $I(Z;Y)$, computed using FlowNIB. Since each model contains multiple hidden layers, we report the average mutual information across all layers.

From the figure, we observe that bidirectional models consistently retain higher mutual information for both $I(X;Z)$ and $I(Z;Y)$. Interestingly, even small bidirectional models such as RoBERTa-base (125M) outperform larger unidirectional models like MobileLLM-600M and SmollM2-360 in terms of MIC. While bidirectional models are often assumed to be more computationally expensive—approximately twice the cost of unidirectional models—we find that in practice, smaller bidirectional models achieve better MIC while maintaining comparable or even lower compute cost. For example, as shown in Table 1, RoBERTa-base-125M requires only 21.76 GFLOPs and 10.87 GMACs, whereas the comparable unidirectional MobileLLM-125M requires 31.9 GFLOPs and 15.95 GMACs.

Main Results: Our results show that bidirectional models consistently outperform unidirectional models across both classification and regression tasks (Table 2, Table 3). For example, in classification, DeBERTa-v3-Large achieves the highest average accuracy of 84.73% using masked token prediction, improving by +3.77 percentage points over its pooling-based variant. Furthermore, we observe that even RoBERTa-base outperforms MobileLLM-600M in several tasks, highlighting a consistent trend with mutual information (MI): better MI is correlated with improved context modeling and task performance.

To further understand this behavior, we use FlowNIB to visualize the information flow in Figure 4, comparing bidirectional and unidirectional MI across datasets. The results show that bidirectional models retain higher mutual information in both $I(X;Z)$ and $I(Z;Y)$. Additionally, Figure 5 presents token-level mutual information from the final layer after fine-tuning on the SST-2 dataset, further illustrating the representational advantage of bidirectional models.

Overall, these findings highlight that masking inference yields stronger gains in bidirectional models, while generation provides modest improvements for unidirectional models but fails to close the accuracy and error gap, reinforcing the advantage of bidirectional context and masking for both classification and regression.

Key Finding

MIC is strongly correlated with model performance: representations with higher MIC values—i.e., high mutual information with both the input and the output—consistently yield better downstream task accuracy.

4 Related work

Information bottleneck in deep learning The IB principle has been studied from both practical and theoretical perspectives in deep learning. On the practical side, [2, 16, 1] formulated the IB problem as a deep learning objective and introduced variational approximations to enable optimization via gradient descent. On the theoretical side, [44, 41] provided an information-theoretic framework for understanding deep learning, establishing the IB as a foundational tool for analyzing representation learning and generalization in deep learning. These fundamental ideas have inspired a wide range of follow-up works [14, 38, 40] that further investigate deep learning dynamics through the lens of information theory.

Mutual information estimation Mutual information quantifies the statistical dependence between two random variables and plays an important role in the IB principle. However, the mutual information is notoriously difficult to estimate between continuous high-dimensional random variables. Traditional

nonparametric approaches [13, 29, 10, 42, 22, 20] typically are not scalable with dimension and sample size. To achieve an efficient estimator, recent work [31, 32] characterized the mutual information of two random variables with the Kullback-Leibler (KL-) divergence [21] between their joint distribution and the product of the marginals and used a dual representations to cast the KL divergence. The Mutual Information Neural Estimator (MINE) [4] utilized the dual representation of the KL divergence and estimated mutual information via gradient descent over neural networks and thus scaled well.

5 Limitations

While our study offers new insights into information flow and effective dimensionality in language models, it has some limitations. First, our analysis focuses primarily on bidirectional models, which are well-suited for understanding and classification tasks but cannot be directly used for autoregressive generation. As a result, we do not evaluate FlowNIB in generative settings, which may limit the generality of our findings. Second, while FlowNIB shows consistent behavior on small- to medium-scale models, we have not yet validated its performance or scalability on large-scale models (e.g., with billions of parameters). Future work will explore how FlowNIB behaves in generation-oriented architectures and whether the observed trends in mutual information and effective dimensionality hold at larger model scales. Two promising directions arise from these limitations. First, a compelling question is whether bidirectionality can be partially introduced into autoregressive models to enhance contextual understanding without disrupting generation capabilities. Second, we envision extending FlowNIB to interpret and analyze large-scale deep models, potentially offering theoretical explanations for emergent behaviors, training instabilities, or generalization trends in frontier LLMs.

6 Conclusion

This work investigates why bidirectional models outperform unidirectional ones in natural language understanding tasks and context modeling, from both a theoretical perspective and empirical evidence. To explain this, we introduce **FlowNIB**, a dynamic framework based on the Information Bottleneck principle. FlowNIB tracks mutual information across layers and training time. Our results show that bidirectional models retain more input and predictive information, leading to more effective representations and stronger performance. FlowNIB provides a principled explanation for this advantage and opens new directions for analyzing and improving deep language models.

Contents

References

[1] Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905, 2018.

[2] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[3] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. Smollm - blazingly fast and remarkably powerful, 2024.

[4] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[5] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[6] Hao Cheng, Dongze Lian, Shenghua Gao, and Yanlin Geng. Utilizing information bottleneck to evaluate the capability of deep neural networks for image classification. *Entropy*, 21(5):456, 2019.

[7] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[8] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[9] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

[10] Georges A Darbellay and Igor Vajda. Estimation of the information by an adaptive partitioning of the observation space. *IEEE Transactions on Information Theory*, 45(4):1315–1321, 1999.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *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, 2019.

[12] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on pure and applied mathematics*, 28(1):1–47, 1975.

[13] Andrew M Fraser and Harry L Swinney. Independent coordinates for strange attractors from mutual information. *Physical review A*, 33(2):1134, 1986.

[14] Ziv Goldfeld and Yury Polyanskiy. The information bottleneck problem and its applications in machine learning. *IEEE Journal on Selected Areas in Information Theory*, 1(1):19–38, 2020.

[15] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[16] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

[17] Nabil Hossain, John Krumm, and Michael Gamon. " president vows to cut< taxes> hair": Dataset and analysis of creative text editing for humorous headlines. *arXiv preprint arXiv:1906.00274*, 2019.

[18] Md Kowsher, Tara Esmaeilbeig, Chun-Nam Yu, Mojtaba Soltanalian, and Niloofar Yousefi. Rocoft: Efficient finetuning of large language models with row-column updates. *arXiv preprint arXiv:2410.10075*, 2024.

[19] Md Kowsher, Nusrat Jahan Prottasha, Prakash Bhat, Chun-Nam Yu, Mojtaba Soltanalian, Ivan Garibay, Ozlem Garibay, Chen Chen, and Niloofar Yousefi. Predicting through generation: Why generation is better for prediction. *arXiv preprint arXiv:2502.17817*, 2025.

[20] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 69(6):066138, 2004.

[21] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[22] Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE transactions on pattern analysis and machine intelligence*, 24(12):1667–1671, 2002.

[23] Qintong Li, Zhiyong Wu, Lingpeng Kong, and Wei Bi. Explanation regeneration via information bottleneck. *arXiv preprint arXiv:2212.09603*, 2022.

[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. *arXiv preprint arXiv:1907.11692*, 2019.

[25] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*, 2024.

[26] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[27] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. The sick (sentences involving compositional knowledge) dataset for relatedness and entailment. *URL: https://doi. org/10.5281/zenodo*, 2787612, 2014.

[28] Rainer Martin and Carl Masreliez. Robust estimation via stochastic approximation. *IEEE Transactions on information Theory*, 21(3):263–271, 1975.

[29] Young-Il Moon, Balaji Rajagopalan, and Upmanu Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318, 1995.

[30] Thanh T Nguyen and Jaesik Choi. Layer-wise learning of stochastic neural networks with information bottleneck. *arXiv preprint arXiv:1712.01272*, 2017.

[31] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[32] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

[33] Alec Radford. Improving language understanding with unsupervised learning. *OpenAI Res*, 2018.

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[36] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

[37] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

[38] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[39] Matthew Shardlow, Michael Cooper, and Marcos Zampieri. Complex: A new corpus for lexical complexity prediction from likert scale data. *arXiv preprint arXiv:2003.07008*, 2020.

[40] Ravid Shwartz-Ziv. Information flow in deep neural networks. *arXiv preprint arXiv:2202.06749*, 2022.

[41] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[42] Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating mutual information by maximum likelihood density ratio estimation. In *New challenges for feature selection in data mining and knowledge discovery*, pages 5–20. PMLR, 2008.

[43] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[44] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. Ieee, 2015.

[45] R Vinayakumar, B Premjith, Sachin Kumar, Soman Kp, and Prabaharan Poornachandran. deepcybernet at emoint-2017: Deep emotion intensities in tweets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 259–263, 2017.

[46] 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. *arXiv preprint arXiv:1804.07461*, 2018.

[47] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*, 2024.

[48] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[49] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

A Bidirectional vs Unidirectional Representation

Theorem A.1 (Conditioning Reduces Entropy)**.** *Let X and Y be continuous random variables with joint density $f_{X,Y}(x, y)$, marginal densities $f_X(x)$, $f_Y(y)$, and conditional density $f_{X|Y}(x|y)$. The differential entropy satisfies:*

$$H(X) \geq H(X|Y),$$

where $H(X)$ and $H(X|Y)$ denote the marginal and conditional differential entropy, respectively. [9]

Proof. For continuous random variables, differential entropy is defined as:

$$H(X) = -\int f_X(x) \log f_X(x) dx,$$

$$H(X|Y) = -\iint f_{X,Y}(x, y) \log f_{X|Y}(x|y) dxdy.$$

Substituting $f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}$ into $H(X|Y)$, we derive:

$$H(X|Y) = -\iint f_{X,Y}(x, y) \log \frac{f_{X,Y}(x, y)}{f_Y(y)} dxdy$$

Expanding the logarithm:

$$H(X|Y) = -\underbrace{\iint f_{X,Y}(x, y) \log f_{X,Y}(x, y)\, dxdy}_{H(X,Y)}$$

$$+ \iint f_{X,Y}(x, y) \log f_Y(y)\, dxdy.$$

The second term simplifies using the marginal $\int f_{X,Y}(x, y)dx = f_Y(y)$:

$$\iint f_{X,Y}(x, y) \log f_Y(y) dxdy =$$

$$\int f_Y(y) \log f_Y(y) dy = -H(Y).$$

Thus,

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

To show $H(X) \geq H(X|Y)$, we invoke the non-negativity of the Kullback-Leibler (KL) divergence:

$$D_{\text{KL}}(f_{X,Y} \| f_X f_Y) =$$

$$\iint f_{X,Y}(x, y) \log \frac{f_{X,Y}(x, y)}{f_X(x) f_Y(y)} dxdy \geq 0.$$

Expanding the integrand:

$$D_{\text{KL}} = \iint f_{X,Y}(x,y) \log f_{X,Y}(x,y) dxdy$$
$$- \iint f_{X,Y}(x,y) \log f_X(x) dxdy -$$
$$\iint f_{X,Y}(x,y) \log f_Y(y) dxdy.$$

Recognizing the entropy terms:

$$D_{\text{KL}} = -H(X,Y) + H(X) + H(Y) \geq 0$$
$$\implies H(X) + H(Y) \geq H(X,Y).$$

Substituting $H(X,Y) = H(X|Y) + H(Y)$ into the inequality:

$$H(X) \geq H(X|Y).$$

□

Theorem A.2 (Monotonicity of Conditional Entropy)**.** *Let X, Y, Z be continuous random variables. Then the differential entropy satisfies:*

$$H(X \mid Y) \geq H(X \mid Y, Z),$$

with equality if and only if $X \perp Z \mid Y$. More generally, for any sequence $Y_1, \ldots, Y_n$,

$$H(X \mid Y_1) \geq H(X \mid Y_1, Y_2) \geq \cdots \geq H(X \mid Y_1, \ldots, Y_n).$$

Proof. We begin with the definition of conditional differential entropy:

$$H(X \mid Y) = -\iint f_{X,Y}(x,y) \log f_{X|Y}(x \mid y)\, dx\, dy,$$
$$H(X \mid Y, Z) = -\iiint f_{X,Y,Z}(x,y,z) \log f_{X|Y,Z}(x \mid y,z)\, dx\, dy\, dz.$$

Recall that:

$$f_{X|Y}(x \mid y) = \int f_{X|Y,Z}(x \mid y,z) f_{Z|Y}(z \mid y)\, dz.$$

Now apply Jensen's inequality using the convexity of $-\log(\cdot)$:

$$-\log\left(\int f_{X|Y,Z}(x \mid y,z) f_{Z|Y}(z \mid y)\, dz\right) \leq -\int f_{Z|Y}(z \mid y) \log f_{X|Y,Z}(x \mid y,z)\, dz.$$

Multiplying both sides by $f_{X|Y}(x \mid y)$ and integrating over x, y, we obtain:

$$\begin{aligned} H(X \mid Y) &= -\iint f_{X,Y}(x,y) \log f_{X|Y}(x \mid y)\, dx\, dy \\ &\geq -\iiint f_{X,Y,Z}(x,y,z) \log f_{X|Y,Z}(x \mid y,z)\, dx\, dy\, dz \\ &= H(X \mid Y, Z). \end{aligned}$$

Equality holds iff Jensen's inequality becomes an equality, which occurs if and only if

$$f_{X|Y,Z}(x \mid y,z) = f_{X|Y}(x \mid y) \quad \text{a.e. in } z,$$

i.e., $X \perp Z \mid Y$.

For the generalization, apply this result inductively:

$$H(X \mid Y_1) \geq H(X \mid Y_1, Y_2) \geq \cdots \geq H(X \mid Y_1, \ldots, Y_n).$$

□

Theorem A.3 (Bidirectional Representations Preserve More Mutual Information)**.** *Let X denote a sequence input $x_1, x_2, \ldots, x_n$. Let $Z^{\rightarrow}$ denote the unidirectional hidden representation constructed from the forward context:*

$$Z^{\rightarrow} = (z_1^{\rightarrow}, z_2^{\rightarrow}, \ldots, z_n^{\rightarrow}) \quad \textit{with } z_t^{\rightarrow} = f(x_1, \ldots, x_t),$$

and $Z^{\leftarrow}$ the backward representation:

$$Z^{\leftarrow} = (z_1^{\leftarrow}, z_2^{\leftarrow}, \ldots, z_n^{\leftarrow}) \quad \textit{with } z_t^{\leftarrow} = g(x_t, \ldots, x_n).$$

Let the bidirectional representation be:

$$Z^{\leftrightarrow} = (Z^{\rightarrow}, Z^{\leftarrow}).$$

Then the mutual information between X and the bidirectional representation satisfies:

$$I(X; Z^{\leftrightarrow}) \geq I(X; Z^{\rightarrow}),$$

with equality if and only if $Z^{\leftarrow} \perp X \mid Z^{\rightarrow}$.

—

Proof. We begin with the identity:

$$I(X; Z) = H(X) - H(X \mid Z).$$

Apply this to both representations:

$$I(X; Z^{\rightarrow}) = H(X) - H(X \mid Z^{\rightarrow}),$$
$$I(X; Z^{\leftrightarrow}) = H(X) - H(X \mid Z^{\rightarrow}, Z^{\leftarrow}).$$

Since $Z^{\leftrightarrow}$ contains strictly more information than $Z^{\rightarrow}$, we can invoke the *monotonicity of conditional entropy* A.2:

$$H(X \mid Z^{\rightarrow}) \geq H(X \mid Z^{\rightarrow}, Z^{\leftarrow}),$$

with equality iff $X \perp Z^{\leftarrow} \mid Z^{\rightarrow}$.

Subtracting both sides from $H(X)$ gives:

$$I(X; Z^{\leftrightarrow}) = H(X) - H(X \mid Z^{\rightarrow}, Z^{\leftarrow}) \geq H(X) - H(X \mid Z^{\rightarrow}) = I(X; Z^{\rightarrow}).$$

Thus:

$$I(X; Z^{\leftrightarrow}) \geq I(X; Z^{\rightarrow}).$$

Equality holds iff:

$$H(X \mid Z^{\rightarrow}) = H(X \mid Z^{\rightarrow}, Z^{\leftarrow}),$$

which by the equality condition of monotonicity of conditional entropy holds iff:

$$X \perp Z^{\leftarrow} \mid Z^{\rightarrow}.$$

Similarly with respect to output we can show:

$$I(Z^{\leftrightarrow}; Y) \geq I(Z^{\rightarrow}; Y).$$

This completes the proof.

□

Theorem A.4 (General Bound on Representation Difference)**.** *Let $Z^{\leftrightarrow}, Z^{\rightarrow} \in \mathbb{R}^d$ denote the bidirectional and unidirectional representations of the same input token at a given layer, and define:*

$$\Delta_Z := Z^{\leftrightarrow} - Z^{\rightarrow}.$$

Then the expected squared difference satisfies:

$$\mathbb{E}\|\Delta_Z\|^2 = \operatorname{tr}\operatorname{Cov}(Z^{\leftrightarrow}) + \operatorname{tr}\operatorname{Cov}(Z^{\rightarrow}) - 2\operatorname{tr}\operatorname{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow}) + \|\mathbb{E}[\Delta_Z]\|^2.$$

In particular, we have the following bound:

$$\begin{aligned}
&\operatorname{tr}\operatorname{Cov}(Z^{\leftrightarrow}) + \operatorname{tr}\operatorname{Cov}(Z^{\rightarrow}) - 2|\operatorname{tr}\operatorname{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow})| \\
&\quad \leq \mathbb{E}\|\Delta_Z\|^2 - \|\mathbb{E}[\Delta_Z]\|^2 \\
&\quad \leq \operatorname{tr}\operatorname{Cov}(Z^{\leftrightarrow}) + \operatorname{tr}\operatorname{Cov}(Z^{\rightarrow}) + 2|\operatorname{tr}\operatorname{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow})|.
\end{aligned}$$

Proof. By the covariance identity, we have:

$$\mathrm{Cov}(\Delta_Z) = \mathrm{Cov}(Z^{\leftrightarrow}) + \mathrm{Cov}(Z^{\rightarrow}) - \mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow}) - \mathrm{Cov}(Z^{\rightarrow}, Z^{\leftrightarrow}).$$

Taking the trace and noting that $\mathrm{tr}(A^\top) = \mathrm{tr}(A)$, we obtain:

$$\mathrm{tr}\,\mathrm{Cov}(\Delta_Z) = \mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}) + \mathrm{tr}\,\mathrm{Cov}(Z^{\rightarrow}) - 2\,\mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow}).$$

The expected squared norm decomposes as:

$$\mathbb{E}\|\Delta_Z\|^2 = \mathrm{tr}\,\mathrm{Cov}(\Delta_Z) + \|\mathbb{E}[\Delta_Z]\|^2.$$

Substituting the expression for $\mathrm{Cov}(\Delta_Z)$ yields the stated identity.

Finally, since for any real scalar a, we have $-|a| \le a \le |a|$, it follows:

$$-|\,\mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow})| \le \mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow}) \le |\,\mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow})|,$$

which implies:

$$\mathrm{tr}\,\mathrm{Cov}(\Delta_Z) \in \left[\mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}) + \mathrm{tr}\,\mathrm{Cov}(Z^{\rightarrow}) - 2|\,\mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow})|,\ \mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}) + \mathrm{tr}\,\mathrm{Cov}(Z^{\rightarrow}) + 2|\,\mathrm{tr}\,\mathrm{Cov}(Z^{\leftrightarrow}, Z^{\rightarrow})|\right].$$

Substitute into the expectation equation to complete the proof. □

Theorem A.5 (Bidirectional Representation Generates a Finer Sigma-Algebra)**.** *Let $Z^{\rightarrow}$ denote the unidirectional (left-to-right) representation of input X, and $Z^{\leftarrow}$ denote the backward (right-to-left) representation. Define the bidirectional representation:*

$$Z^{\leftrightarrow} := (Z^{\rightarrow}, Z^{\leftarrow}).$$

Let $\sigma(Z)$ denote the sigma-algebra generated by random vector Z. Then:

$$\sigma(Z^{\rightarrow}) \subseteq \sigma(Z^{\leftrightarrow}),$$

with equality if and only if $Z^{\leftarrow}$ is measurable with respect to $\sigma(Z^{\rightarrow})$; that is, there exists a measurable function f such that:

$$Z^{\leftarrow} = f(Z^{\rightarrow}) \quad \textit{almost surely}.$$

Proof. By definition, the sigma-algebra generated by $Z^{\leftrightarrow}$ is:

$$\sigma(Z^{\leftrightarrow}) := \sigma(Z^{\rightarrow}, Z^{\leftarrow}),$$

the smallest sigma-algebra making both $Z^{\rightarrow}$ and $Z^{\leftarrow}$ measurable.

Since $Z^{\rightarrow}$ is a component of $Z^{\leftrightarrow}$, every set measurable with respect to $Z^{\rightarrow}$ is measurable with respect to $Z^{\leftrightarrow}$. Therefore:

$$\sigma(Z^{\rightarrow}) \subseteq \sigma(Z^{\leftrightarrow}).$$

Equality holds if and only if $\sigma(Z^{\leftarrow}) \subseteq \sigma(Z^{\rightarrow})$; that is, if every event measurable under $Z^{\leftarrow}$ is already measurable under $Z^{\rightarrow}$. By the definition of generated sigma-algebra, this is equivalent to $Z^{\leftarrow}$ being measurable with respect to $\sigma(Z^{\rightarrow})$; i.e., there exists a measurable function f such that:

$$Z^{\leftarrow} = f(Z^{\rightarrow}) \quad \text{almost surely.}$$

Otherwise, if no such f exists, then $Z^{\leftarrow}$ introduces additional independent information, and the inclusion is strict:

$$\sigma(Z^{\rightarrow}) \subset \sigma(Z^{\leftrightarrow}).$$

□

Lemma A.6 (Effective Dimensionality of Bidirectional Representations)**.** *Let $Z^{\rightarrow} \in \mathbb{R}^D$ denote the unidirectional representation and $Z^{\leftrightarrow} := (Z^{\rightarrow}, Z^{\leftarrow}) \in \mathbb{R}^{2D}$ the concatenated bidirectional representation of input X. Define ℓ_2-norm-based effective dimension as*

$$d_{\text{eff}}(Z) := \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2},$$

where λ_i are eigenvalues of the covariance matrix of Z. If $\mathrm{Cov}(Z^{\leftarrow}, Z^{\rightarrow})$ is non-singular, then:

$$d_{\text{eff}}(Z^{\leftrightarrow}) \ge d_{\text{eff}}(Z^{\rightarrow}),$$

with equality iff $Z^{\leftarrow}$ is conditionally redundant given $Z^{\rightarrow}$ (i.e., $\mathrm{Cov}(Z^{\leftarrow} \mid Z^{\rightarrow}) = 0$).

Proof. Let $\Sigma^{\rightarrow} := \mathrm{Cov}(Z^{\rightarrow}) \in \mathbb{R}^{D\times D}$ and $\Sigma^{\leftrightarrow} := \mathrm{Cov}(Z^{\leftrightarrow}) \in \mathbb{R}^{2D\times 2D}$ denote the covariance matrices of unidirectional and bidirectional representations, respectively.

By block structure:

$$\Sigma^{\leftrightarrow} = \begin{bmatrix} \Sigma^{\rightarrow} & C \\ C^{\top} & \Sigma^{\leftarrow} \end{bmatrix},$$

where $C := \mathrm{Cov}(Z^{\rightarrow}, Z^{\leftarrow})$.

Let $\{\lambda_i^{\rightarrow}\}_{i=1}^{D}$ be eigenvalues of $\Sigma^{\rightarrow}$, and $\{\lambda_j^{\leftrightarrow}\}_{j=1}^{2D}$ eigenvalues of $\Sigma^{\leftrightarrow}$.

Since $\Sigma^{\leftrightarrow}$ augments $\Sigma^{\rightarrow}$ with additional variables $Z^{\leftarrow}$ and cross-covariance C, by eigenvalue interlacing theorem (Cauchy's interlacing), we have:

$$\sum_{j=1}^{2D} \lambda_j^{\leftrightarrow} \geq \sum_{i=1}^{D} \lambda_i^{\rightarrow},$$

and

$$\sum_{j=1}^{2D} (\lambda_j^{\leftrightarrow})^2 \geq \sum_{i=1}^{D} (\lambda_i^{\rightarrow})^2,$$

with strict inequality if C or $\Sigma^{\leftarrow}$ is nonzero.

Applying definition:

$$d_{\text{eff}}(Z^{\leftrightarrow}) = \frac{(\sum_j \lambda_j^{\leftrightarrow})^2}{\sum_j (\lambda_j^{\leftrightarrow})^2}.$$

Since numerator and denominator both increase under positive-definite augmentation, and quadratic-over-linear ratio increases under positive additive terms (Jensen's inequality), we conclude:

$$d_{\text{eff}}(Z^{\leftrightarrow}) \geq d_{\text{eff}}(Z^{\rightarrow}).$$

Equality holds iff $\Sigma^{\leftarrow} = 0$ and $C = 0$, implying $Z^{\leftarrow}$ carries no additional variance or covariance beyond $Z^{\rightarrow}$. □

B FlowNIB: Flow Neural Information Bottleneck

We consider the Markov chain:

$$X \longrightarrow Z \longrightarrow Y,$$

where X denotes the input, Z the intermediate representation, and Y the target variable.

Our goal is to learn a representation Z that:

- Compresses the input information by minimizing $I(X; Z)$,
- Preserves predictive information by maximizing $I(Z; Y)$.

The classical **Information Bottleneck** (IB) principle [43, 44] formalizes this tradeoff as the optimization problem:

$$\min_{p(z|x)} \; I(X; Z) - \beta I(Z; Y),$$

where $\beta > 0$ controls the balance between compression and prediction.

Limitations of Classical IB: Mutual information (MI) involves estimating high-dimensional integrals over P_{XZ} and $P_X \otimes P_Z$, which is computationally intractable for high-dimensional X, Z. The KL divergence

$$D_{\mathrm{KL}}\big(\mathbb{P}_{XZ} \,\|\, \mathbb{P}_X \otimes \mathbb{P}_Z\big)$$

is especially problematic because it requires knowledge of the joint and marginal distributions, which are rarely known in practice [4]. Moreover, classical IB assumes that the underlying joint and marginal distributions are known, which is unrealistic in high-dimensional settings. As a result, classical IB methods are difficult to apply directly to deep learning models, such as large-scale language models.

FlowNIB Approach: To address these challenges, we introduce **FlowNIB** (Flow Neural Information Bottleneck), which dynamically transitions from input preservation to output prediction during training. Specifically, we propose a time-dependent tradeoff parameter $\alpha \in [0, 1]$ that linearly decays from 1 to 0 as training progresses.

Thus, the FlowNIB loss at each training step becomes:

$$\mathcal{L}(\theta, t) = -\big(\alpha(t) \times I(X; Z) + (1 - \alpha(t)) \times I(Z; Y)\big),$$

where:

- Early training ($\alpha \approx 1$) prioritizes maximizing $I(X; Z)$,
- Late training ($\alpha \approx 0$) prioritizes maximizing $I(Z; Y)$.

Each mutual information term is defined as:

$$I(X; Z) = D_{\mathrm{KL}}\big(p(x, z) \,\|\, p(x)p(z)\big), \quad I(Z; Y) = D_{\mathrm{KL}}\big(p(z, y) \,\|\, p(z)p(y)\big),$$

where D_{KL} denotes the Kullback–Leibler divergence.

However, computing D_{KL} exactly is infeasible for high-dimensional X, Z. To overcome this, we employ a variational lower bound inspired by MINE [4]:

$$I(X; Z) \geq \mathbb{E}_{p(x,z)}\big[T_{xz}(x, z)\big] - \log \mathbb{E}_{p(x)p(z)}\big[e^{T_{xz}(x,z)}\big],$$
$$I(Z; Y) \geq \mathbb{E}_{p(z,y)}\big[T_{zy}(z, y)\big] - \log \mathbb{E}_{p(z)p(y)}\big[e^{T_{zy}(z,y)}\big],$$

where T_{xz} and T_{zy} are learned scalar-valued critic functions.

Since X, Z, Y may have different or high dimensions, we normalize mutual information estimates by the square of the effective dimension [36]:

$$d_{\mathrm{eff}}(Z) = \frac{\big(\sum_i \lambda_i\big)^2}{\sum_i \lambda_i^2},$$

where $\{\lambda_i\}$ are the eigenvalues obtained by applying PCA on Z .

The normalized mutual information estimates are:

$$\hat{I}(X; Z) = \frac{\mathbb{E}_{p(x,z)}[T_{xz}(x, z)] - \log \mathbb{E}_{p(x)p(z)}\big[e^{T_{xz}(x,z)}\big]}{d_{\mathrm{eff}}(Z)^2},$$
$$\hat{I}(Z; Y) = \frac{\mathbb{E}_{p(z,y)}[T_{zy}(z, y)] - \log \mathbb{E}_{p(z)p(y)}\big[e^{T_{zy}(z,y)}\big]}{d_{\mathrm{eff}}(Y)^2}.$$

Thus, the final loss optimized during training is:

$$\mathcal{L}(\theta, t) = -\Big(\alpha(t)\, \hat{I}(X; Z) + (1 - \alpha)(t)\, \hat{I}(Z; Y)\Big),$$

which, when expanded fully, becomes:

$$\mathcal{L}(\theta, t) = -\left(\alpha \frac{\mathbb{E}_{p(x,z)}[T_{xz}(x, z)] - \log \mathbb{E}_{p(x)p(z)}\big[e^{T_{xz}(x,z)}\big]}{d_{\mathrm{eff}}(Z)^2} + (1-\alpha) \frac{\mathbb{E}_{p(z,y)}[T_{zy}(z, y)] - \log \mathbb{E}_{p(z)p(y)}\big[e^{T_{zy}(z,y)}\big]}{d_{\mathrm{eff}}(Y)^2}\right).$$

Here, θ denotes the parameters of both the encoder $p(z \mid x; \theta)$ and the critics T_{xz} and T_{zy}.

Theorem B.1 (**Proof of Theorem 2.4** Consistency under Optimal Critics)**.** *Let random variables $(X, Z) \sim p(x, z)$ and $(Z, Y) \sim p(z, y)$ define the Markov chain $X \to Z \to Y$. Suppose the critic functions T^*_{xz} and T^*_{zy} achieve the optimal solutions:*

$$T^*_{xz}(x, z) = \log \frac{p(x, z)}{p(x)p(z)}, \quad T^*_{zy}(z, y) = \log \frac{p(z, y)}{p(z)p(y)}.$$

Then, the dimension-normalized mutual information estimators used in the FlowNIB loss satisfy:

$$\hat{I}(X; Z) \xrightarrow{T_{xz} \to T^*_{xz}} \frac{I(X; Z)}{d_{\mathrm{eff}}(Z)^2}, \quad \hat{I}(Z; Y) \xrightarrow{T_{zy} \to T^*_{zy}} \frac{I(Z; Y)}{d_{\mathrm{eff}}(Y)^2},$$

and thus consistently estimate the normalized mutual information.

Proof. We prove the result explicitly for the pair (X, Z); the proof for (Z, Y) follows analogously.

The mutual information $I(X; Z)$ can be expressed as a KL divergence [21]:

$$I(X; Z) = D_{\mathrm{KL}}\big(p(x, z) \,\|\, p(x)p(z)\big).$$

Using the Donsker–Varadhan variational representation [12], we have:

$$I(X; Z) = \sup_{T_{xz} \in \mathcal{F}} \mathbb{E}_{p(x,z)}[T_{xz}(x, z)] - \log \mathbb{E}_{p(x)p(z)}\left[e^{T_{xz}(x,z)}\right],$$

where $\mathcal{F}$ denotes the class of measurable functions $T_{xz} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$.

It is known that the supremum above is uniquely achieved at the optimal critic:

$$T^*_{xz}(x, z) = \log \frac{p(x, z)}{p(x)p(z)}.$$

Substituting T^*_{xz} into the above variational bound yields:

$$\mathbb{E}_{p(x,z)}\left[\log \frac{p(x, z)}{p(x)p(z)}\right] - \log \mathbb{E}_{p(x)p(z)}\left[\frac{p(x, z)}{p(x)p(z)}\right].$$

Evaluating the second expectation explicitly, we obtain:

$$\mathbb{E}_{p(x)p(z)}\left[\frac{p(x, z)}{p(x)p(z)}\right] = \iint p(x)p(z)\frac{p(x, z)}{p(x)p(z)}\, dx\, dz = \iint p(x, z)\, dx\, dz = 1.$$

Thus, the variational bound becomes exactly:

$$I(X; Z) = \mathbb{E}_{p(x,z)}\left[\log \frac{p(x, z)}{p(x)p(z)}\right] = I(X; Z).$$

This proves that the MINE [4] estimator exactly recovers $I(X; Z)$ under the optimal critic.

Recall the normalized mutual information estimator defined as:

$$\hat{I}(X; Z) = \frac{\mathbb{E}_{p(x,z)}[T_{xz}(x, z)] - \log \mathbb{E}_{p(x)p(z)}[e^{T_{xz}(x,z)}]}{d_{\mathrm{eff}}(Z)^2},$$

where effective dimension $d_{\mathrm{eff}}(Z)$ is:

$$d_{\mathrm{eff}}(Z) = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2}, \quad \{\lambda_i\} \text{ eigenvalues from PCA on } Z.$$

Using the optimal critic T^*_{xz}, the numerator exactly equals $I(X; Z)$. Hence, we have the limiting behavior:

$$\hat{I}(X; Z) \xrightarrow{T_{xz} \to T^*_{xz}} \frac{I(X; Z)}{d_{\mathrm{eff}}(Z)^2}.$$

The same reasoning holds for $\hat{I}(Z; Y)$ using the optimal critic T^*_{zy}.

Therefore, the dimension-normalized estimators used in FlowNIB consistently estimate the normalized mutual information under optimal critic conditions, completing the proof. □

Theorem B.2 (Convergence Behavior of FlowNIB Training)**.** *Let $\mathcal{L}(\theta, t)$ denote the FlowNIB dynamic loss at training time t, with a scheduling function $\alpha(t) : \mathbb{R}_{\geq 0} \to [0, 1]$ satisfying:*

- *$\alpha(t)$ is continuous and monotonically decreasing,*
- *$\alpha(0) = 1$, $\lim_{t\to\infty} \alpha(t) = 0$.*

Suppose that at each training step t, the parameter update $\theta(t)$ approximately minimizes $\mathcal{L}(\theta, t)$.

Then, the learned representation Z satisfies the following:

1. *During early training* $(\alpha(t) \approx 1)$*,* Z *captures maximal information about* X*.*
2. *During late training* $(\alpha(t) \approx 0)$*,* Z *preserves maximal relevant information about* Y *while minimizing irrelevant information about* X*.*

Moreover, in the limit $t \to \infty$*, the optimization of* $\mathcal{L}(\theta, t)$ *asymptotically focuses on maximizing* $\hat{I}(Z; Y)$*.*

Proof. We split the proof into three parts: early stage behavior, late stage behavior, and asymptotic convergence.

Step 1: Early training $(\alpha(t) \approx 1)$**.**

Recall that the FlowNIB loss is:

$$\mathcal{L}(\theta, t) = -\Big(\alpha(t)\hat{I}(X; Z) + (1 - \alpha(t))\hat{I}(Z; Y)\Big).$$

When $\alpha(t) \approx 1$, the loss is dominated by:

$$\mathcal{L}(\theta, t) \approx -\hat{I}(X; Z).$$

Thus, minimizing $\mathcal{L}(\theta, t)$ approximately corresponds to **maximizing** $\hat{I}(X; Z)$. This forces the encoder $p(z|x; \theta)$ to preserve as much information about X as possible, encouraging a near-invertible or high-fidelity mapping initially.

Step 2: Late training $(\alpha(t) \approx 0)$**.**

As training progresses, $\alpha(t)$ decays toward 0. In the late phase $(t \to \infty)$, $\alpha(t) \approx 0$, and the loss is dominated by:

$$\mathcal{L}(\theta, t) \approx -\hat{I}(Z; Y).$$

Thus, the optimization now focuses primarily on maximizing $\hat{I}(Z; Y)$. This encourages the encoder to preserve only the aspects of Z that are predictive of Y, discarding irrelevant or non-predictive information from X.

Step 3: Asymptotic Convergence Behavior.

Under standard stochastic approximation assumptions [28] (e.g., learning rates η_t satisfying $\sum_t \eta_t = \infty, \sum_t \eta_t^2 < \infty$), the sequence of parameters $\{\theta(t)\}$ converges almost surely to a stationary point of the limiting objective.

Since $\alpha(t) \to 0$, in the limit $t \to \infty$ the dynamic loss becomes:

$$\lim_{t \to \infty} \mathcal{L}(\theta, t) = -\hat{I}(Z; Y).$$

Thus, asymptotically, FlowNIB optimizes to maximize $\hat{I}(Z; Y)$ exclusively, leading Z to be a compressed yet sufficient representation for predicting Y.

This completes the proof. □

Theorem B.3 (Dynamic Tradeoff Leads to Optimal Compressed Predictor)**.** *Suppose the dynamic weighting parameter* $\alpha(t)$ *satisfies:*

$$\alpha(t) \searrow 0 \quad \textit{as} \quad t \to \infty,$$

and at each time t*, the parameters* θ_t *approximately minimize the FlowNIB loss:*

$$\mathcal{L}(\theta, t) = -\Big(\alpha(t)\, \hat{I}(X; Z) + (1 - \alpha(t))\, \hat{I}(Z; Y)\Big).$$

Let Z' *denote any alternative representation obtained from* X*, i.e.,* $Z' = f'(X)$ *for some measurable function* f'*, satisfying the sufficiency condition:*

$$I(Z'; Y) = I(X; Y).$$

Then, under optimal critic estimation and sufficient model expressiveness, in the limit $t \to \infty$*, the learned representation* Z *satisfies:*

$$I(Z; Y) = I(X; Y), \quad \textit{and} \quad I(X; Z) \leq I(X; Z').$$

Proof. When $\alpha(t)$ is close to 1 (early training), the FlowNIB loss is dominated by maximizing $\hat{I}(X;Z)$:

$$\mathcal{L}(\theta, t) \approx -\hat{I}(X;Z).$$

Thus, the optimization encourages the encoder to maximize mutual information between X and Z, i.e., capturing as much information from X as possible into Z.

Hence, initially, Z encodes nearly all information about X.

As $t \to \infty$, $\alpha(t) \to 0$, and the FlowNIB loss becomes dominated by maximizing $\hat{I}(Z;Y)$:

$$\mathcal{L}(\theta, t) \approx -\hat{I}(Z;Y).$$

Thus, the optimization now focuses on maximizing predictive sufficiency: making Z preserve all information about Y.

Formally, in the limit, the optimization enforces:

$$I(Z;Y) = I(X;Y),$$

because if Z lost any predictive information about Y, the term $\hat{I}(Z;Y)$ would not be maximized.

Thus, Z becomes a **sufficient statistic** for Y.

While maintaining sufficiency ($I(Z;Y) = I(X;Y)$), the training still implicitly minimizes $\hat{I}(X;Z)$ whenever $\alpha(t) > 0$.

Thus, throughout training, even as $\alpha(t)$ decays, there remains a weak regularization pressure to reduce $I(X;Z)$ while maintaining predictive sufficiency.

Since $\alpha(t)$ is strictly decreasing and vanishes only asymptotically, compression pressure persists but decays over time.

Thus, among all representations satisfying $I(Z;Y) = I(X;Y)$, the final Z minimizes $I(X;Z)$:

$$Z = \arg \min_{Z' :\, I(Z';Y) = I(X;Y)} I(X;Z').$$

This satisfies the definition of a **minimal sufficient representation**.

Hence, the learned representation Z is simultaneously sufficient and maximally compressed, completing the proof. □

Theorem B.4 (Compression Bound)**.** *At any training time t, suppose the FlowNIB loss $\mathcal{L}(\theta, t)$ is minimized, and critics T_{xz}, T_{zy} achieve optimality. Then, the effective mutual information $I(X;Z)$ satisfies the upper bound:*

$$I(X;Z) \leq \frac{d_{\text{eff}}(Z)^2}{\alpha(t)} \left(-\mathcal{L}(\theta, t) + (1 - \alpha(t)) \hat{I}(Z;Y) \right),$$

where $\hat{I}(Z;Y)$ is the dimension-normalized estimate of $I(Z;Y)$.

Proof. By definition of the FlowNIB dynamic objective at time t, we have:

$$\mathcal{L}(\theta, t) = -\left(\alpha(t)\, \hat{I}(X;Z) + (1 - \alpha(t))\, \hat{I}(Z;Y) \right).$$

Rearranging terms:

$$-\mathcal{L}(\theta, t) = \alpha(t)\, \hat{I}(X;Z) + (1 - \alpha(t))\, \hat{I}(Z;Y).$$

Thus:

$$\alpha(t)\, \hat{I}(X;Z) = -\mathcal{L}(\theta, t) - (1 - \alpha(t))\, \hat{I}(Z;Y),$$

which implies:

$$\hat{I}(X;Z) = \frac{-\mathcal{L}(\theta, t) - (1 - \alpha(t))\, \hat{I}(Z;Y)}{\alpha(t)}.$$

Recall that the dimension-normalized estimator relates to the true mutual information by:

$$\hat{I}(X;Z) = \frac{I(X;Z)}{d_{\text{eff}}(Z)^2},$$

under the assumption that critics T_{xz}, T_{zy} are optimal (cf. Theorem 2.4).

Substituting this relation yields:

$$\frac{I(X;Z)}{d_{\text{eff}}(Z)^2} = \frac{-\mathcal{L}(\theta,t) - (1-\alpha(t))\,\hat{I}(Z;Y)}{\alpha(t)}.$$

Multiplying both sides by $d_{\text{eff}}(Z)^2$ gives:

$$I(X;Z) = \frac{d_{\text{eff}}(Z)^2}{\alpha(t)} \left(-\mathcal{L}(\theta,t) - (1-\alpha(t))\,\hat{I}(Z;Y)\right).$$

Taking into account that mutual information $I(X;Z)$ is non-negative, and $\hat{I}(Z;Y) \geq 0$, we obtain the bound:

$$I(X;Z) \leq \frac{d_{\text{eff}}(Z)^2}{\alpha(t)} \left(-\mathcal{L}(\theta,t) + (1-\alpha(t))\,\hat{I}(Z;Y)\right),$$

as required.

□

Theorem B.5 (Strong Effective Dimension Compression under FlowNIB)**.** *Let $(X,Y) \sim p(x,y)$ with Y a low-entropy random variable. Suppose Z_t denotes the representation at training time t, and let Σ_{Z_t} be its covariance matrix. Suppose FlowNIB minimizes the dynamic loss:*

$$\mathcal{L}(\theta,t) = -\left(\alpha(t)\,\hat{I}(X;Z) + (1-\alpha(t))\,\hat{I}(Z;Y)\right),$$

with optimal critics. Then, under mild regularity assumptions (bounded second moments, smooth optimization), the following holds:

$$t_1 \leq t_2 \quad \Rightarrow \quad d_{\text{eff}}(Z_{t_1}) \geq d_{\text{eff}}(Z_{t_2}),$$

where

$$d_{\text{eff}}(Z) = \frac{(\text{tr}(\Sigma_Z))^2}{\text{tr}(\Sigma_Z^2)}$$

is the effective dimension.

Proof. We proceed in formal steps.

Step 1: Relation between $I(X;Z)$ and the entropy of Z. Recall that for continuous random variables, the differential entropy $h(Z)$ is linked to mutual information via:

$$I(X;Z) = h(Z) - h(Z|X),$$

where $h(Z)$ is the entropy of Z, and $h(Z|X)$ is the conditional entropy.

If the decoder is deterministic (or near-deterministic) given X, then $h(Z|X) \approx 0$, so:

$$I(X;Z) \approx h(Z).$$

Thus, minimizing $I(X;Z)$ under FlowNIB approximately minimizes $h(Z)$.

Step 2: Entropy and Covariance Spectrum. For a multivariate Gaussian variable $Z \sim \mathcal{N}(\mu, \Sigma_Z)$, the differential entropy is:

$$h(Z) = \frac{1}{2}\log\left((2\pi e)^d \det(\Sigma_Z)\right).$$

Thus, minimizing $h(Z)$ corresponds to minimizing $\det(\Sigma_Z)$, the determinant of the covariance matrix.

By the inequality between arithmetic mean (trace) and geometric mean (determinant) for positive semidefinite matrices (e.g., by AM-GM inequality over eigenvalues), we have:

$$\det(\Sigma_Z) \leq \left(\frac{\operatorname{tr}(\Sigma_Z)}{d}\right)^d,$$

with equality if and only if Σ_Z has all eigenvalues equal.

Thus, as $\det(\Sigma_Z)$ shrinks (due to decreasing entropy), either: - $\operatorname{tr}(\Sigma_Z)$ shrinks, or - the eigenvalue distribution becomes more concentrated.

In particular, if $\operatorname{tr}(\Sigma_Z)$ remains roughly stable (common in practice with BatchNorm, etc.), then the shrinkage happens by concentration: a few eigenvalues dominate.

Step 3: Behavior of Effective Dimension. Recall:

$$d_{\text{eff}}(Z) = \frac{(\operatorname{tr}(\Sigma_Z))^2}{\operatorname{tr}(\Sigma_Z^2)}.$$

If eigenvalues concentrate (i.e., variance is captured by fewer directions), then $\operatorname{tr}(\Sigma_Z^2)$ increases relative to $(\operatorname{tr}(\Sigma_Z))^2$, and hence $d_{\text{eff}}(Z)$ decreases.

Formally:

- Suppose Σ_Z has eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$, - As λ_1 grows and others shrink (energy concentration), - $\operatorname{tr}(\Sigma_Z^2) = \sum_i \lambda_i^2$ increases relative to $(\sum_i \lambda_i)^2$, - Hence, $d_{\text{eff}}(Z)$ decreases.

Step 4: Conclusion. Thus, under FlowNIB minimization:

- $I(X; Z)$ decreases, - Therefore $h(Z)$ decreases, - Thus $\det(\Sigma_Z)$ shrinks, - Leading to concentration of the spectrum, - Which in turn decreases $d_{\text{eff}}(Z)$.

Hence:

$$t_1 \leq t_2 \quad \Rightarrow \quad d_{\text{eff}}(Z_{t_1}) \geq d_{\text{eff}}(Z_{t_2}),$$

as required.

□

Lemma B.6 (Mutual Information under Compression)**.** *Let X, Z, and Z' be random variables forming the Markov chain:*

$$X \to Z \to Z',$$

where Z' is a (possibly stochastic) measurable function of Z. Then, mutual information satisfies:

$$I(X; Z) \geq I(X; Z').$$

Proof. We use the **Data Processing Inequality** (DPI) for mutual information.

Recall that for any Markov chain $A \to B \to C$, it holds that:

$$I(A; B) \geq I(A; C).$$

In our setting: - $A = X$, - $B = Z$, - $C = Z'$.

Thus:

$$I(X; Z) \geq I(X; Z').$$

automatically follows by application of the Data Processing Inequality.

For completeness, we briefly derive the DPI.

Recall that:

$$I(X; Z) = \mathbb{E}_{p(x,z)}\left[\log \frac{p(x, z)}{p(x)p(z)}\right],$$

$$I(X; Z') = \mathbb{E}_{p(x,z')}\left[\log \frac{p(x, z')}{p(x)p(z')}\right].$$

By the Markov assumption $X \to Z \to Z'$, we have:

$$p(x, z, z') = p(x)p(z \mid x)p(z' \mid z).$$

Thus:

$$p(x, z') = \int p(x, z, z')\, dz = p(x) \int p(z \mid x)p(z' \mid z)\, dz,$$

$$p(z') = \int p(x, z')\, dx = \int p(x)p(z' \mid x)\, dx.$$

An alternative way to express mutual information is:

$$I(X; Z) = \inf_{q(x|z)} \mathbb{E}_{p(x,z)} \left[\log \frac{p(x \mid z)}{q(x \mid z)} \right],$$

where $q(x \mid z)$ are variational approximations to $p(x \mid z)$.

Similarly:

$$I(X; Z') = \inf_{q(x|z')} \mathbb{E}_{p(x,z')} \left[\log \frac{p(x \mid z')}{q(x \mid z')} \right].$$

Since Z' is a compression (coarsening) of Z, the mapping $z \mapsto z'$ loses information about x, unless it is invertible (rare).

Hence, $p(x \mid z')$ is an average (mixture) of $p(x \mid z)$ over all z mapping to z'.

By convexity of KL-divergence, and the fact that averaging (marginalizing) reduces distinguishability, we have:

$$\mathbb{E}_{p(x,z)} \left[\log \frac{p(x \mid z)}{q(x \mid z)} \right] \geq \mathbb{E}_{p(x,z')} \left[\log \frac{p(x \mid z')}{q(x \mid z')} \right].$$

Thus:

$$I(X; Z) \geq I(X; Z').$$

Hence, mutual information cannot increase under compression, completing the proof.

□

Lemma B.7 (Mutual Information Lower Bound Tightness)**.** *Let random variables $(X, Z) \sim p(x, z)$. Consider the variational lower bound on mutual information:*

$$I(X; Z) \geq \mathbb{E}_{p(x,z)}[T_{xz}(x, z)] - \log \mathbb{E}_{p(x)p(z)} \left[e^{T_{xz}(x,z)} \right],$$

where $T_{xz} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ is any measurable critic function. Then, the bound is tight if and only if the critic satisfies:

$$T^*_{xz}(x, z) = \log \frac{p(x, z)}{p(x)p(z)}.$$

Proof. We recall that mutual information can be written as the Kullback–Leibler divergence:

$$I(X; Z) = D_{\mathrm{KL}}(p(x, z) \,\|\, p(x)p(z)).$$

Using the Donsker–Varadhan variational representation of KL divergence, we have:

$$D_{\mathrm{KL}}(p \,\|\, q) = \sup_T \left\{ \mathbb{E}_p[T] - \log \mathbb{E}_q[e^T] \right\},$$

where the supremum is over all measurable functions $T : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$.

Substituting $p = p(x, z)$ and $q = p(x)p(z)$, we obtain:

$$I(X; Z) = \sup_T \left\{ \mathbb{E}_{p(x,z)}[T(x, z)] - \log \mathbb{E}_{p(x)p(z)} \left[e^{T(x,z)} \right] \right\}.$$

Thus, the inequality:

$$I(X;Z) \geq \mathbb{E}_{p(x,z)}[T_{xz}(x,z)] - \log \mathbb{E}_{p(x)p(z)}\left[e^{T_{xz}(x,z)}\right]$$

always holds for any T_{xz}, and equality is achieved if and only if T_{xz} achieves the supremum.

It is well-known (and standard in variational inference theory) that the supremum is attained at:

$$T^*_{xz}(x,z) = \log \frac{p(x,z)}{p(x)p(z)}.$$

Substituting this into the variational bound, we get:

$$\mathbb{E}_{p(x,z)}\left[\log \frac{p(x,z)}{p(x)p(z)}\right] - \log \mathbb{E}_{p(x)p(z)}\left[\frac{p(x,z)}{p(x)p(z)}\right].$$

Evaluating the second term:

$$\mathbb{E}_{p(x)p(z)}\left[\frac{p(x,z)}{p(x)p(z)}\right] = \iint p(x)p(z)\frac{p(x,z)}{p(x)p(z)}\,dx\,dz = \iint p(x,z)\,dx\,dz = 1,$$

thus:

$$\log(1) = 0.$$

Therefore, the variational bound simplifies exactly to:

$$\mathbb{E}_{p(x,z)}\left[\log \frac{p(x,z)}{p(x)p(z)}\right] = I(X;Z).$$

Thus, the lower bound is tight (achieves equality) if and only if $T_{xz} = T^*_{xz}$.

□

Lemma B.8 (Non-Monotonic Dependence of Mutual Information on Output Dimension)**.** *Let $X \in \mathbb{R}^{d_X}$, $Z \in \mathbb{R}^{d_Z}$, and $Y \in \mathbb{R}^{d_Y}$ denote input, latent, and output variables, respectively, with d_X, d_Z fixed and d_Y variable.*

Then under FlowNIB optimization, the mutual information $I(X;Z)$ and $I(Z;Y)$ are non-monotonic functions of d_Y, satisfying:

$$\frac{\partial I(X;Z)}{\partial d_Y} > 0 \quad \textit{for } d_Y < k, \quad \frac{\partial I(X;Z)}{\partial d_Y} < 0 \quad \textit{for } d_Y > k$$

and similarly for $I(Z;Y)$, for some critical threshold $k \approx d_X$.

Proof Sketch. FlowNIB optimizes a tradeoff between $I(X;Z)$ and $I(Z;Y)$, constrained by the model's representational capacity d_Z and data complexity.

When d_Y is small ($d_Y \ll d_X$), the predictive target contains limited information; thus $I(Z;Y)$ is small and the latent representation does not need high complexity.

As d_Y increases toward d_X, the predictive task demands richer information; both $I(X;Z)$ and $I(Z;Y)$ increase to capture relevant features.

However, once $d_Y > d_X$, the output space exceeds the input manifold's capacity; the latent representation Z cannot fully carry the increased predictive information due to fixed d_Z, leading to saturation and eventual decline in both $I(X;Z)$ and $I(Z;Y)$ as redundant or noisy output components exceed representational limits.

This yields a non-monotonic dependency of mutual information on d_Y, peaking around $d_Y \approx d_X$, then declining as d_Y further increases.

□

Algorithm 1 FlowNIB: Flow Neural Information Bottleneck

Require: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, pretrained model f_θ, MI critics T_{xz} and T_{zy}, scheduler $\alpha(t)$, number of training steps T

Initialize FlowNIB parameters and critics
for $t = 1$ to T **do**
  Sample mini-batch $\{(x, y)\}$ from $\mathcal{D}$
  Compute hidden representation $Z = f_\theta(x)$
  Estimate $I(X; Z)$ using MINE:
  $\hat{I}(X; Z) \leftarrow \mathbb{E}_{p(x,z)}[T_{xz}(x, z)] - \log \mathbb{E}_{p(x)p(z)}[e^{T_{xz}(x,z)}]$
  Estimate $I(Z; Y)$ using MINE:
  $\hat{I}(Z; Y) \leftarrow \mathbb{E}_{p(z,y)}[T_{zy}(z, y)] - \log \mathbb{E}_{p(z)p(y)}[e^{T_{zy}(z,y)}]$
  Normalize MI by effective dimensions:
  $\hat{I}_n(X; Z) \leftarrow \frac{\hat{I}(X;Z)}{d_{\text{eff}}(Z)^2}, \quad \hat{I}_n(Z; Y) \leftarrow \frac{\hat{I}(Z;Y)}{d_{\text{eff}}(Y)^2}$
  Compute dynamic loss:
  $\mathcal{L}_{\text{FlowNIB}} \leftarrow -\Big(\alpha(t) \cdot \hat{I}_n(X; Z) + (1 - \alpha(t)) \cdot \hat{I}_n(Z; Y)\Big)$
  Update schedule: $\alpha(t + 1) \leftarrow \max(0, \alpha(t) - \delta)$
  Backpropagate and update θ, T_{xz}, T_{zy}
end for

Proposition B.9 (Effective Dimensionality Adaptation under FlowNIB)**.** *Let $X \in \mathbb{R}^{d_X}$ and $Y \in \mathbb{R}^{d_Y}$ be input and output random variables with dimensions d_X, d_Y. Let Z_ℓ denote the latent representation at layer ℓ produced by a model trained under FlowNIB.*

Then, under optimal critic approximation and continuous optimization, the effective dimension $d_{\text{eff}}(Z_\ell)$ exhibits the following dependence on d_Y (with d_X fixed):

$$\frac{\partial d_{\text{eff}}(Z_\ell)}{\partial d_Y} \begin{cases} < 0 & \text{if } d_Y \ll d_X \\ \approx 0 & \text{if } d_Y \approx d_X \\ > 0 & \text{if } d_Y \gg d_X \end{cases}$$

i.e., the effective dimension $d_{\text{eff}}(Z_\ell)$ decreases with d_Y when d_Y is small, plateaus when $d_Y \approx d_X$, and increases when d_Y exceeds d_X.

Proof Sketch. Under FlowNIB, the latent representation Z_ℓ is optimized to balance information preservation $I(X; Z_\ell)$ and predictive sufficiency $I(Z_\ell; Y)$, modulated dynamically by $\alpha(t)$.

When $d_Y \ll d_X$, the predictive information $I(Z_\ell; Y)$ is small; the model prioritizes compressing irrelevant input variance, resulting in reduced $d_{\text{eff}}(Z_\ell)$.

When $d_Y \approx d_X$, the predictive complexity of Y matches the input complexity; the model maintains $d_{\text{eff}}(Z_\ell)$ to balance preserving input and predictive information.

When $d_Y \gg d_X$, the model must expand Z_ℓ to capture sufficient predictive capacity, increasing $d_{\text{eff}}(Z_\ell)$ to span a higher-dimensional output manifold.

Empirical observations support this trend, where $d_{\text{eff}}(Z_\ell)$ traces a non-monotonic dependency on d_Y, reflecting an intrinsic adaptation of latent geometry to output complexity.

□

C Ablation Study

Effect of step size δ on FlowNIB dynamics: We conducted an ablation study on the MRPC dataset to analyze the influence of the step size δ controlling the decay of $\alpha(t)$ in FlowNIB . Specifically, we varied δ logarithmically from 10^{-1} to 10^{-11} and measured the evolution of mutual information $I(X; Z)$ and $I(Z; Y)$ throughout training. Figure 6(left) shows the corresponding trajectories in the Information Plane. We observe that large step sizes (e.g., $\delta = 10^{-1}$) induce rapid compression,

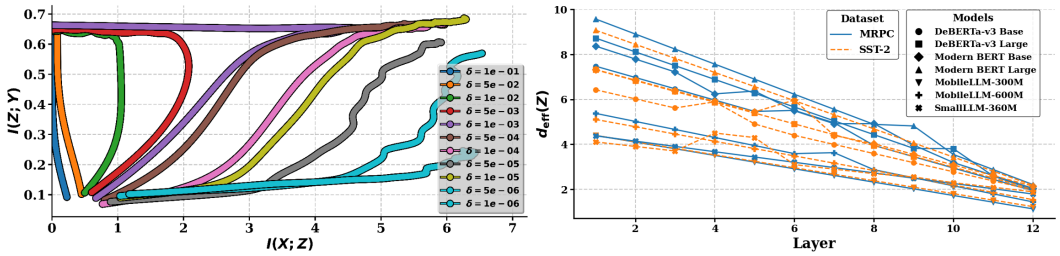

Figure 6: (Left)Information plane trajectories under varying step sizes δ for $\alpha(t)$ in FlowNIB. Each curve shows the progression of mutual information $I(X;Z)$ and $I(Z;Y)$ across 2000 training epochs . (Right) Effective dimensionality $d_{\text{eff}}(Z)$ across layers for different models on MRPC and SST-2. Bidirectional models show higher $d_{\text{eff}}(Z)$ than unidirectional models at every layer.

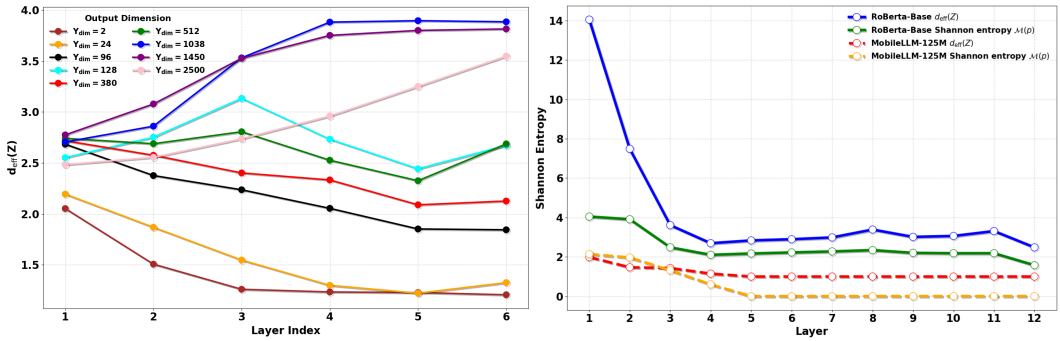

Figure 7: Effective dimension and Shannon entropy across network layers. **Left:** Effective dimension $d_{\text{eff}}(Z)$ across layers for different output dimensions Y_{dim}. **Right:** Shannon entropy $\mathcal{M}(p)$ across layers for RoBERTa-Base and MobileLLM-125M. Both plots use bold markers and shadows to emphasize trends in representation capacity and information compression.

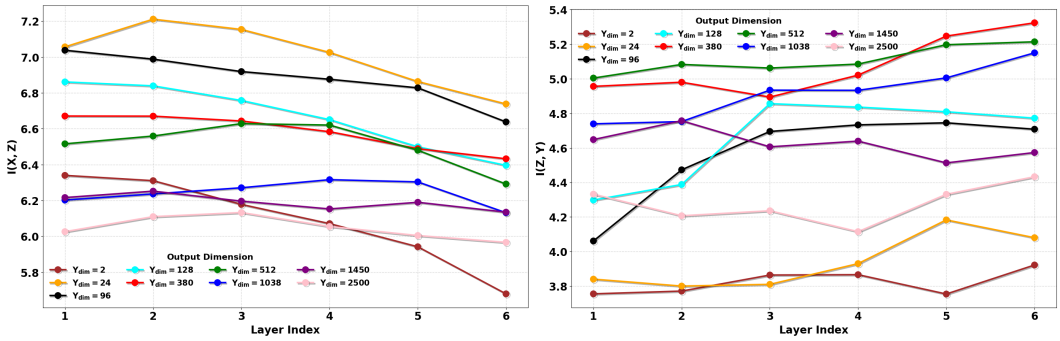

Figure 8: Visualization of mutual information across layers for different output dimensions. The left plot shows $I(X;Z)$ and the right plot shows $I(Z;Y)$ for various output dimensions Y_{dim}. Each curve represents a specific output dimension, with bold markers and shadows to highlight the trends. This analysis provides insights into the evolution of representation capacity and target alignment across network layers as the output dimension increases.

sharply reducing $I(X;Z)$ early in training but failing to preserve sufficient predictive information $I(Z;Y)$, likely due to premature information loss. Conversely, very small step sizes (e.g., $\delta = 10^{-6}$) cause negligible decay of $\alpha(t)$, leading to nearly static representations that retain high $I(X;Z)$ but fail to increase $I(Z;Y)$. Intermediate step sizes (e.g., $\delta = 10^{-3}$ to $\delta = 10^{-4}$) achieve the most desirable balance, gradually reducing $I(X;Z)$ while increasing $I(Z;Y)$, effectively steering the model toward

the information bottleneck frontier. These findings empirically validate our theoretical insight that δ serves as a critical control knob governing the speed and quality of information compression in FlowNIB.

Effective Dimensionality Across Models: We measure effective dimensionality $d_{\text{eff}}(Z)$ across layers for DeBERTaV3 (base, large), ModernBERT (base, large), MobileLLM (300M, 600M), and SmallLLM (360M) on MRPC and SST-2. To ensure fair comparison across models with different depths, we normalize layer indices to a common scale of 1 to 12. Figure 6(right) shows that $d_{\text{eff}}(Z)$ decreases monotonically with depth for all models, reflecting progressive compression (reasons of decreasing in Ablation Study C).

Importantly, bidirectional models consistently exhibit higher $d_{\text{eff}}(Z)$ than unidirectional models at every layer. For example, on MRPC, DeBERTaV3-Large starts at 8.73 and compresses to 1.98, while MobileLLM-600M starts at 5.38 and compresses to 1.44. Similar trends appear on SST-2. These findings empirically support Lemma 2.3, confirming that bidirectional representations retain richer and more expressive features throughout depth.

Effective Dimensionality vs. Output Complexity: We study how the effective dimensionality $d_{\text{eff}}(Z)$ of the latent representations changes with different output dimensions using the time-series forecasting dataset `ETTh1` [49] by following Proposition B.9. We use a fixed 6-layer network with each layer having 128 units and keep the input dimension fixed at $d_X = 380$. We vary the output dimension d_Y from very small ($d_Y = 2$) to much larger than the input ($d_Y = 2500$). As shown in Figure 7, when the output dimension is much smaller than the input ($d_Y \ll d_X$), the effective dimension $d_{\text{eff}}(Z)$ decreases across layers, showing that the representation becomes more compressed. As d_Y grows closer to or larger than d_X, we observe a non-monotonic trend: the dimension first compresses, then expands. When $d_Y \gg d_X$, the effective dimension increases across layers, suggesting that the model adjusts the complexity of its representations to match the complexity of the prediction task. This behavior occurs even without directly optimizing for it in FlowNIB, showing that the shape of the output affects how the model organizes its internal representations.

Mutual Information Dynamics Across Output Dimensions and Layers: We explore how changing the output dimension Y_{dim} affects mutual information and model performance by following Lemma B.8. We trained the same model with different output sizes: $Y_{\text{dim}} \in \{2, 24, 96, 128, 380, 512, 1038, 1450, 2500\}$, and measured the mutual information between inputs and hidden layers $I(X; Z)$, and between hidden layers and outputs $I(Z; Y)$, after training. As shown in Figure 8, $I(X; Z)$ generally decreases across layers, especially for larger Y_{dim}, meaning more information is lost as the network gets deeper. At the same time, $I(Z; Y)$ increases with depth, but for large Y_{dim}, it saturates early—suggesting it's harder for the model to align with very high-dimensional outputs. Interestingly, models with intermediate output dimensions (like $Y_{\text{dim}} = 96$ or 128) show a better balance: they retain useful input information and achieve strong alignment with the output. This balance leads to better performance. Overall, we find that output dimensionality plays a key role in controlling how well the model balances input compression and predictive accuracy, making it an important hyperparameter to tune.

Validating Generalized Effective Dimensionality: To validate our definition of generalized effective dimensionality, we compare the layerwise trends of $d_{\text{eff}}(Z)$ (based on the ℓ_2-norm participation ratio) and the Shannon entropy $\mathcal{M}(p)$ across two models: RoBERTa-Base and MobileLLM-125M. As shown in Figure 7 (Right), both metrics follow similar trends across layers—confirming that higher entropy leads to higher effective dimension, consistent with our definition $d_{\text{eff}}(Z; \mathcal{M}) := \exp(\mathcal{M}(p))$. Notably, RoBERTa-Base maintains higher entropy and effective dimension than MobileLLM-125M at every layer, reflecting its richer representational capacity. The first few layers show a sharp drop in entropy, followed by a stable regime, aligning with the known compression phase in transformer representations. This empirical behavior confirms that both the entropy and d_{eff} satisfy the expected monotonicity and boundedness properties outlined in Definition 2.2, including non-negativity and the Schur-concavity property.

D LoRA Based Performance Comparison

Table 4 shows the performance comparison between bidirectional and unidirectional models using LoRA.

Table 4: Accuracy results across nine NLP classification tasks comparing bidirectional and unidirectional models under pooling, masking, and generation inference strategies using LoRA fine-tuning.

Model	Method	SST-2	MRPC	QNLI	RTE	CoLA	MNLI	BoolQ	HellaSwag	SIQA	Avg.
DeBERTa-v3-Base	Pooling	95.12	88.75	91.75	82.85	85.43	85.96	63.55	55.22	46.74	77.15
	Masking	96.22	90.03	93.10	85.92	88.55	88.10	65.05	68.33	61.92	81.81
DeBERTa-v3-Large	Pooling	96.25	92.88	94.67	88.90	94.12	91.92	65.48	58.15	52.04	81.82
	Masking	96.94	94.95	95.35	90.85	93.05	91.96	65.12	74.10	66.41	85.30
RoBERTa-Base	Pooling	93.80	83.40	91.13	82.20	85.45	85.95	62.10	51.78	44.63	75.72
	Masking	94.80	86.10	93.42	86.02	88.25	87.20	63.80	65.33	61.12	80.45
RoBERTa-Large	Pooling	95.12	88.40	93.76	86.10	93.02	90.14	64.00	56.23	47.15	79.66
	Masking	96.67	91.98	95.10	88.45	95.33	90.92	64.25	70.35	62.45	83.83
ModernBERT-Base	Pooling	93.70	82.40	90.25	81.52	84.22	86.02	62.00	54.18	45.70	75.78
	Masking	94.92	84.05	92.88	85.00	85.80	88.55	61.35	62.00	60.00	78.95
ModernBERT-Large	Pooling	95.00	88.55	93.50	87.32	90.25	92.80	63.50	59.00	48.50	79.82
	Masking	96.32	91.10	95.12	88.50	91.02	92.10	63.90	72.42	64.33	83.42
GPT-2 Medium	Pooling	92.70	84.32	90.42	68.50	79.15	78.02	62.33	36.80	37.42	69.96
	Generation	93.40	85.72	91.65	69.02	80.10	79.43	63.00	36.55	42.12	71.00
GPT-2 Large	Pooling	93.75	85.50	83.35	65.90	82.85	79.55	63.50	39.20	40.50	70.68
	Generation	94.05	87.05	85.12	67.88	84.23	81.72	64.05	39.70	45.02	71.98
SmolLM2-360M	Pooling	93.80	84.20	90.92	69.90	81.22	84.10	62.75	41.20	41.55	72.18
	Generation	94.52	85.85	91.93	70.50	83.80	85.10	62.60	42.40	49.45	73.68
SmolLM2-135M	Pooling	91.90	83.05	89.43	67.55	80.15	81.52	61.35	37.00	40.25	70.13
	Generation	92.80	83.85	90.05	68.12	81.82	82.78	61.70	40.00	46.20	71.59
MobileLLM-125M	Pooling	92.25	81.42	89.82	68.42	79.12	81.35	59.50	32.30	40.40	69.07
	Generation	92.98	82.35	90.22	68.92	80.42	82.20	60.25	36.12	47.33	70.53
MobileLLM-350M	Pooling	93.00	82.65	90.32	69.55	81.58	82.55	62.05	35.42	41.50	70.73
	Generation	94.10	82.98	90.85	70.25	82.62	83.40	62.85	39.20	50.05	72.15
MobileLLM-600M	Pooling	94.25	86.80	90.92	71.32	83.92	84.12	63.50	44.50	44.20	73.06
	Generation	94.95	87.55	91.50	72.02	85.92	84.30	63.75	47.80	57.32	75.68

E Dataset

The details of datasets are described in Table 5

Table 5: Overview of the 16 benchmark datasets used in our experiments across classification and regression tasks.

Dataset	Task Type	Domain	Description
SST-2 [46]	Classification	Sentiment Analysis	The Stanford Sentiment Treebank, a binary sentiment classification dataset labeling sentences as positive or negative.
MRPC [46]	Classification	Paraphrase Detection	The Microsoft Research Paraphrase Corpus for detecting whether two sentences are semantically equivalent.
QNLI [46]	Classification	Question Answering / NLI	A question natural language inference dataset built from SQuAD, determining if a context sentence contains the answer.
RTE [46]	Classification	Natural Language Inference	The Recognizing Textual Entailment dataset for determining if a hypothesis is entailed by a premise.
MNLI [46]	Classification	Natural Language Inference	Multi-Genre Natural Language Inference dataset covering entailment, neutral, and contradiction relations across multiple genres.
CoLA [46]	Classification	Grammatical Acceptability	Corpus of Linguistic Acceptability, evaluating whether sentences conform to English grammatical rules.
BoolQ [7]	Classification	Reading Comprehension	Boolean Questions dataset with yes/no questions based on Wikipedia passages requiring reading comprehension.
HellaSwag [48]	Classification	Commonsense Reasoning	Tests commonsense reasoning by selecting the most plausible continuation of a given scenario.
SIQA [37]	Classification	Social Intelligence	Social IQa dataset evaluating models' understanding of social situations, emotions, and intentions.
WASSA [45]	Regression	Emotion Intensity	WASSA-2017 dataset for predicting emotion intensity scores for tweets across multiple emotions.
SICK [26]	Regression	Semantic Similarity	Sentences Involving Compositional Knowledge dataset for measuring sentence similarity and entailment.
STSB-regression [5]	Regression	Semantic Similarity	Semantic Textual Similarity Benchmark scored on a continuous scale from 0 to 5.
LCP [39]	Regression	Lexical Complexity	Lexical Complexity Prediction dataset for predicting the complexity of words within their context.
CRP [39]	Regression	Complex Word Identification	Complex Word Identification dataset from SemEval, labeling words as simple or complex in context.
Humicroedit [17]	Regression	Humor Perception	SemEval humor dataset evaluating the impact of small text edits (micro-edits) on humor perception.

F Environment Setup

All experiments are conducted using PyTorch 2.0 and Hugging Face Transformers version 4.50. Training and evaluation are performed on a single NVIDIA A100 GPU with 80GB of memory. We use Python 3.10 within an Anaconda virtual environment configured with CUDA 12.1. Key dependencies include NumPy, SciPy, scikit-learn, and tqdm for data processing and evaluation. Random seeds are fixed across all runs to ensure reproducibility.

G Evaluation Metrics

We evaluate our models using task-specific metrics selected for their interpretability, relevance, and comparability to prior work. For **classification tasks**, we adopt *accuracy* as the primary metric, defined as the ratio of correct predictions to the total number of predictions:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Accuracy provides a straightforward measure of model correctness and aligns with standard practices in classification benchmarks [46].

For **regression tasks**, we report both *mean squared error (MSE)* and *mean absolute error (MAE)* to capture complementary aspects of prediction error. MSE emphasizes larger errors due to the squared term, while MAE reflects the average magnitude of errors:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \quad \text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|,$$

where N is the number of samples, y_i is the ground-truth label, and $\hat{y}_i$ is the predicted value. These metrics ensure a robust evaluation of both typical and extreme prediction errors [5, 27].

In addition to task performance metrics, we measure the *mutual information* between the input X and the learned representation Z, denoted $I(X;Z)$. Mutual information quantifies how much information about the input is preserved in Z, providing insight into the information bottleneck trade-off [44]. We estimate $I(X;Z)$ using a variational lower bound based on Mutual Information Neural Estimation [4], following prior work in information-theoretic analyses of neural networks.

All metrics are computed using scikit-learn and official benchmark evaluation scripts. Model selection is performed based on validation set performance, with final metrics reported on the held-out test sets.

H Model Description

We compare our method with a range of pretrained language models covering both bidirectional and unidirectional architectures. The bidirectional baselines include **DeBERTaV3-Base** [15], **DeBERTaV3-Large** [15], **RoBERTa-Base** [24], **RoBERTa-Large** [24], **ModernBERT-Base** [47], and **ModernBERT-Large** [47]. The unidirectional baselines include **GPT-2 Medium** [34], **GPT-2 Large** [34], **MobileLLM-125M** [25], **MobileLLM-350M** [25], **MobileLLM-630M** [25], **SmolLM-135M** [3], and **SmolLM-360M** [3]. These models are selected to cover a range of sizes and architectures, enabling a fair and broad evaluation of representational learning. We focus on smaller model sizes to allow fair comparisons since large bidirectional models are not readily available. All baseline models are fine-tuned using RoCoFT adapters with an adapter rank of $r = 3$, enabling efficient fine-tuning without modifying the main model parameters. We use a cosine learning rate schedule for training.

I Hyperparameters

We select hyperparameters systematically to ensure consistent and balanced evaluation across all tasks and models. For classification tasks, we set the learning rate to 1×10^{-4} with batch sizes between 8 and 16. For regression tasks, we increase the learning rate to 1×10^{-3} with batch sizes ranging from 8 to 32. All models are fine-tuned using the AdamW optimizer with a cosine learning rate schedule,

weight decay values in the range of 0.1 to 0.2, and a warmup ratio of 0.1. Gradient accumulation steps are varied between 1 and 8 depending on GPU memory capacity. To improve training stability, gradients are clipped at a maximum norm of 1.0, and label smoothing with a factor of 0.1 is applied where applicable. Each model is trained for 2 to 30 epochs, with warmup steps selected between 100 and 500. These hyperparameter settings are held consistent across experimental runs to ensure fair comparisons and reproducibility. This finding aligns with earlier work showing the benefits of bidirectional models for non-autoregressive NLP tasks. A detailed breakdown of the hyperparameters used for each dataset and model is provided in Appendix, including Table 6 (Humicroedit), Table 7 (WASSA), Table 8 (SICK), Table 9 (STS-B), Table 10 (LCP), Table 11 (SST-2), Table 12 (MRPC), Table 13 (QNLI), Table 14 (RTE), Table 15 (CoLA), Table 16 (MNLI), Table 17 (BoolQ), Table 18 (HellaSwag), and Table 19 (SIQA).

Table 6: Hyperparameter settings for the Humicroedit dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
MobileLLM-350M	6e-4	16	1	0.2	Cosine	3	512	10 / 100
SmolLM-360M	6e-4	16	1	0.2	Cosine	3	512	10 / 100
SmolLM-135M	6e-4	16	1	0.2	Cosine	3	512	10 / 100
ModernBERT-base	6e-4	16	1	0.2	Cosine	3	512	10 / 100
GPT2-medium	6e-4	16	1	0.2	Cosine	3	512	10 / 100
GPT2-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100
deberta-v3-base	6e-4	16	1	0.2	Cosine	3	512	10 / 100
roberta-base	6e-4	16	1	0.2	Cosine	3	512	10 / 100
roberta-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100
deberta-v3-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100
Mobile-llm-125	6e-4	16	1	0.2	Cosine	3	512	10 / 100
Mobile-llm-630	6e-4	16	1	0.2	Cosine	3	512	10 / 100
moden-bert-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100

Table 7: Hyperparameter settings for the WASSA dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-135M	5e-4	14	1	0.2	Cosine	3	512	10 / 100
MobileLLM-350M	5e-4	14	1	0.2	Cosine	3	512	10 / 100
SmolLM-360M	5e-4	14	1	0.2	Cosine	3	512	10 / 100
GPT2-medium	5e-4	14	1	0.2	Cosine	3	512	10 / 100
GPT2-large	5e-4	14	1	0.2	Cosine	3	512	10 / 100
ModernBERT-base	5e-4	14	1	0.2	Cosine	3	512	10 / 100
deberta-v3-base	6e-4	16	1	0.2	Cosine	3	512	10 / 100
roberta-base	6e-4	16	1	0.2	Cosine	3	512	10 / 100
roberta-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100
deberta-v3-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100
Mobile-llm-125	6e-4	16	1	0.2	Cosine	3	512	10 / 100
Mobile-llm-630	6e-4	16	1	0.2	Cosine	3	512	10 / 100
moden-bert-large	6e-4	16	1	0.2	Cosine	3	512	10 / 100

Table 8: Hyperparameter settings for the SICK dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	1e-3	14	1	0.2	Cosine	3	512	20 / 100
SmolLM-135M	1e-3	14	1	0.2	Cosine	3	512	20 / 100
ModernBERT-base	1e-3	8	2	0.2	Cosine	3	512	20 / 100
deberta-v3-base	1e-3	8	2	0.2	Cosine	3	512	20 / 100
GPT2-medium	1e-3	14	1	0.2	Cosine	3	512	20 / 100
GPT2-large	1e-3	14	1	0.2	Cosine	3	512	20 / 100
roberta-base	1e-3	8	2	0.2	Cosine	3	512	20 / 100
roberta-large	1e-3	8	2	0.2	Cosine	3	512	20 / 100
deberta-v3-large	1e-3	8	2	0.2	Cosine	3	512	20 / 100
Mobile-llm-125	1e-3	8	2	0.2	Cosine	3	512	20 / 100
Mobile-llm-630	1e-3	8	2	0.2	Cosine	3	512	20 / 100
moden-bert-large	1e-3	8	2	0.2	Cosine	3	512	20 / 100

J Model Profile Information

We conduct a comprehensive CPU profiling analysis of twelve transformer models to understand the computational bottlenecks and runtime behavior that influence performance. The models we evaluate include DeBERTa-v3-Base Table 20, DeBERTa-v3-Large Table 21, RoBERTa-Base Table 22, RoBERTa-Large Table 23, ModernBERT-Base Table 24, ModernBERT-Large Table 25, GPT-2 Medium Table 26, GPT-2 Large Table 27, SmolLM-135M Table 28, SmolLM-360M Table 29, MobileLLM-125M Table 30, and MobileLLM-600M Table 32. Our CPU profiling shows that bidirectional models are often comparable to unidirectional models. For example, DeBERTa-v3-Base Table 20 and ModernBERT-Base Table 24 complete inference in 502ms and 347ms, respectively, while GPT-2 Medium Table 26 takes 1126ms—more than double the time. Larger bidirectional models like DeBERTa-v3-Large Table 21 and RoBERTa-Large Table 23 have runtimes comparable to

Table 9: Hyperparameter settings for the STSB dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps	Max Grad Norm
SmolLM-360M	2e-4	8	1	0.1	Cosine	3	512	10 / 100	1
MobileLLM-350M	2e-4	8	1	0.1	Cosine	3	512	10 / 100	1
SmolLM-135M	2e-4	8	1	0.1	Cosine	3	512	10 / 100	1
deberta-v3-base	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
roberta-base	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
roberta-large	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
deberta-v3-large	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
Mobile-llm-125	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
Mobile-llm-630	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
moden-bert-large	6e-4	16	1	0.2	Cosine	3	512	20 / 100	1
GPT2-medium	1e-4	16	4	0.0	Cosine	3	512	10 / 100	1
GPT2-large	1e-4	16	4	0.0	Cosine	3	512	10 / 100	1
ModernBERT-base	1e-4	16	4	0.0	Cosine	3	512	10 / 100	1

Table 10: Hyperparameter settings for the LCP dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	5e-4	4	4	0.2	Cosine	3	512	10 / 100
MobileLLM-350M	5e-4	4	4	0.2	Cosine	3	512	10 / 100
SmolLM-135M	5e-4	4	4	0.2	Cosine	3	512	10 / 100
ModernBERT-base	5e-4	4	4	0.2	Cosine	3	512	10 / 100
GPT2-medium	5e-4	4	4	0.2	Cosine	3	512	10 / 100
GPT2-large	5e-4	4	4	0.2	Cosine	3	512	10 / 100
roberta-base	1e-3	10	1	0.2	Cosine	3	512	10 / 100
roberta-large	1e-3	10	1	0.2	Cosine	3	512	10 / 100
deberta-v3-large	1e-3	10	1	0.2	Cosine	3	512	10 / 100
Mobile-llm-125	1e-3	10	1	0.2	Cosine	3	512	10 / 100
Mobile-llm-630	1e-3	10	1	0.2	Cosine	3	512	10 / 100
moden-bert-large	1e-3	10	1	0.2	Cosine	3	512	10 / 100
deberta-v3-base	2e-3	32	1	0.2	Cosine	3	512	10 / 100

Table 11: Hyperparameter settings for the SST-2 dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	1e-4	8	2	0.1	Cosine	3	512	3 / 500
MobileLLM-350M	1e-4	8	2	0.1	Cosine	3	512	3 / 500
SmolLM-135M	1e-4	8	2	0.1	Cosine	3	512	3 / 500
ModernBERT-base	1e-4	8	2	0.1	Cosine	3	512	3 / 500
deberta-v3-base	1e-4	16	4	0.00	Cosine	3	512	3 / 100
roberta-base	1e-4	16	4	0.00	Cosine	3	512	3 / 100
roberta-large	1e-4	16	4	0.00	Cosine	3	512	3 / 100
deberta-v3-large	1e-4	16	4	0.00	Cosine	3	512	3 / 100
Mobile-llm-125	1e-4	16	4	0.00	Cosine	3	512	3 / 100
Mobile-llm-630	1e-4	16	4	0.00	Cosine	3	512	3 / 100
moden-bert-large	1e-4	16	4	0.00	Cosine	3	512	3 / 100
GPT2-medium	1e-4	8	2	0.1	Cosine	3	512	3 / 500
GPT2-large	3e-3	32	1	0.00	Cosine	3	512	2 / 100

Table 12: Hyperparameter settings for the MRPC dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	5e-4	4	4	0.1	Cosine	3	512	10 / 100
MobileLLM-350M	5e-4	4	4	0.1	Cosine	3	512	10 / 100
SmolLM-135M	5e-4	4	4	0.1	Cosine	3	512	10 / 100
ModernBERT-base	5e-4	4	4	0.1	Cosine	3	512	10 / 100
deberta-v3-base	1e-3	64	1	0.00	Cosine	3	512	10 / 100
roberta-base	1e-3	64	1	0.00	Cosine	3	512	10 / 100
roberta-large	1e-3	64	1	0.00	Cosine	3	512	10 / 100
deberta-v3-large	1e-3	64	1	0.00	Cosine	3	512	10 / 100
GPT2-medium	5e-4	4	4	0.1	Cosine	3	512	10 / 100
GPT2-large	1e-4	16	2	0.00	Cosine	3	512	10 / 100
Mobile-llm-125	3e-3	16	1	0.00	Cosine	3	512	5 / 100
Mobile-llm-630	3e-3	16	1	0.00	Cosine	3	512	5 / 100
moden-bert-large	5e-4	4	4	0.1	Cosine	3	512	10 / 100

Table 13: Hyperparameter settings for the QNLI dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	2e-4	8	2	0.1	Cosine	3	512	2 / 500
MobileLLM-350M	2e-4	8	2	0.1	Cosine	3	512	2 / 500
SmolLM-135M	2e-4	8	2	0.1	Cosine	3	512	2 / 500
ModernBERT-base	2e-4	8	2	0.1	Cosine	3	512	2 / 500
GPT2-medium	2e-4	8	2	0.1	Cosine	3	512	2 / 500
GPT2-large	1e-4	12	4	0.00	Cosine	3	512	2 / 100
deberta-v3-base	1e-4	12	4	0.00	Cosine	3	512	2 / 100
roberta-base	1e-4	12	4	0.00	Cosine	3	512	2 / 100
roberta-large	1e-4	12	4	0.00	Cosine	3	512	2 / 100
deberta-v3-large	1e-4	12	4	0.00	Cosine	3	512	2 / 100
Mobile-llm-125	1e-4	12	4	0.00	Cosine	3	512	2 / 100
Mobile-llm-630	1e-4	12	4	0.00	Cosine	3	512	2 / 100
moden-bert-large	1e-4	12	4	0.00	Cosine	3	512	2 / 100

Table 14: Hyperparameter settings for the RTE dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	1e-4	4	8	0.00	Cosine	3	512	30 / 100
MobileLLM-350M	1e-4	4	8	0.00	Cosine	3	512	30 / 100
SmolLM-135M	1e-4	4	8	0.00	Cosine	3	512	30 / 100
ModernBERT-base	1e-4	4	8	0.00	Cosine	3	512	30 / 100
GPT2-medium	1e-4	4	8	0.00	Cosine	3	512	30 / 100
GPT2-large	1e-3	16	2	0.00	Cosine	3	512	30 / 100
deberta-v3-base	1e-4	16	8	0.00	Cosine	3	512	30 / 100
roberta-base	1e-4	16	8	0.00	Cosine	3	512	30 / 100
roberta-large	1e-4	16	8	0.00	Cosine	3	512	30 / 100
deberta-v3-large	1e-4	16	8	0.00	Cosine	3	512	30 / 100
Mobile-llm-125	1e-4	16	8	0.00	Cosine	3	512	30 / 100
Mobile-llm-630	1e-4	16	8	0.00	Cosine	3	512	30 / 100
moden-bert-large	1e-4	16	8	0.00	Cosine	3	512	30 / 100

Table 15: Hyperparameter settings for the COLA dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	2e-5	8	1	0.1	Cosine	3	512	10 / 500
MobileLLM-350M	2e-5	8	1	0.1	Cosine	3	512	10 / 500
SmolLM-135M	2e-5	8	1	0.1	Cosine	3	512	10 / 500
ModernBERT-base	2e-5	8	1	0.1	Cosine	3	512	10 / 500
GPT2-medium	2e-5	8	1	0.1	Cosine	3	512	10 / 500
GPT2-large	1e-3	64	1	0.00	Cosine	3	512	10 / 100
deberta-v3-base	2e-5	4	8	0.00	Cosine	3	512	10 / 100
roberta-base	2e-5	4	8	0.00	Cosine	3	512	10 / 100
roberta-large	2e-5	4	8	0.00	Cosine	3	512	10 / 100
deberta-v3-large	2e-5	4	8	0.00	Cosine	3	512	10 / 100
Mobile-llm-125	5e-4	4	4	0.1	Cosine	3	512	10 / 100
Mobile-llm-630	5e-4	4	4	0.1	Cosine	3	512	10 / 100
moden-bert-large	5e-4	4	4	0.1	Cosine	3	512	10 / 100

Table 16: Hyperparameter settings for the MNLI dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
SmolLM-360M	2e-4	8	4	0.00	Cosine	3	512	2 / 500
MobileLLM-350M	2e-4	8	4	0.00	Cosine	3	512	2 / 500
SmolLM-135M	2e-4	8	4	0.00	Cosine	3	512	2 / 500
ModernBERT-base	2e-4	8	4	0.00	Cosine	3	512	2 / 500
GPT2-medium	2e-4	8	4	0.00	Cosine	3	512	2 / 500
GPT2-large	1e-3	32	1	0.00	Cosine	3	512	2 / 100
deberta-v3-base	1e-3	14	1	0.00	Cosine	3	512	2 / 100
roberta-base	1e-3	14	1	0.00	Cosine	3	512	2 / 100
roberta-large	1e-3	14	1	0.00	Cosine	3	512	2 / 100
deberta-v3-large	1e-3	14	1	0.00	Cosine	3	512	2 / 100
Mobile-llm-125	1e-3	14	1	0.00	Cosine	3	512	2 / 100
Mobile-llm-630	1e-3	14	1	0.00	Cosine	3	512	2 / 100
moden-bert-large	2e-4	8	4	0.00	Cosine	3	512	2 / 500

Table 17: Hyperparameter settings for the BoolQ dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
ModernBERT-base	3e-4	128	1	0.00	Cosine	3	512	100 / 100
MobileLLM-350M	3e-4	128	1	0.00	Cosine	3	512	100 / 100
SmolLM-360M	3e-4	128	1	0.00	Cosine	3	512	100 / 100
SmolLM-135M	3e-4	128	1	0.00	Cosine	3	512	100 / 100
GPT2-medium	3e-4	128	1	0.00	Cosine	3	512	100 / 100
GPT2-large	3e-4	128	1	0.00	Cosine	3	512	100 / 100
deberta-v3-base	3e-4	128	1	0.00	Cosine	3	512	100 / 100
roberta-base	3e-4	128	1	0.00	Cosine	3	512	100 / 100
roberta-large	3e-4	128	1	0.00	Cosine	3	512	100 / 100
deberta-v3-large	3e-4	128	1	0.00	Cosine	3	512	100 / 100
Mobile-llm-125	3e-4	128	1	0.00	Cosine	3	512	100 / 100
Mobile-llm-630	3e-4	128	1	0.00	Cosine	3	512	100 / 100
moden-bert-large	3e-4	128	1	0.00	Cosine	3	512	100 / 100

Table 18: Hyperparameter settings for the HellaSwag dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
deberta-v3-base	1e-4	16	1	0.00	Cosine	3	512	12 / 100
mobilellm-350M	1e-4	16	1	0.00	Cosine	3	512	12 / 100
SmolLM-360M	1e-4	16	1	0.00	Cosine	3	512	12 / 100
SmolLM-135M	1e-4	16	1	0.00	Cosine	3	512	12 / 100
ModernBERT-base	1e-4	16	1	0.00	Cosine	3	512	12 / 100
GPT2-medium	1e-4	16	1	0.00	Cosine	3	512	12 / 100
GPT2-large	1e-4	16	1	0.00	Cosine	3	512	12 / 100
roberta-base	1e-4	16	1	0.00	Cosine	3	512	12 / 100
roberta-large	1e-4	16	1	0.00	Cosine	3	512	12 / 100
deberta-v3-large	1e-4	16	1	0.00	Cosine	3	512	12 / 100
Mobile-llm-125	1e-4	16	1	0.00	Cosine	3	512	12 / 100
Mobile-llm-630	1e-4	16	1	0.00	Cosine	3	512	12 / 100
moden-bert-large	1e-4	16	1	0.00	Cosine	3	512	12 / 100

Table 19: Hyperparameter settings for the SIQA dataset for each evaluated model.

Model	Learning Rate	Batch Size	Grad Accum	Weight Decay	LR Scheduler	Rank	Max Length	Epochs / Warmup Steps
deberta-v3-base	3e-4	16	1	0.00	Cosine	3	512	4 / 100
mobilellm-350M	3e-4	16	1	0.00	Cosine	3	512	4 / 100
SmolLM-360M	3e-4	16	1	0.00	Cosine	3	512	4 / 100
SmolLM-135M	3e-4	16	1	0.00	Cosine	3	512	4 / 100
ModernBERT-base	3e-4	16	1	0.00	Cosine	3	512	4 / 100
GPT2-medium	3e-4	16	1	0.00	Cosine	3	512	4 / 100
GPT2-large	3e-4	16	1	0.00	Cosine	3	512	4 / 100
roberta-base	3e-4	16	1	0.00	Cosine	3	512	4 / 100
roberta-large	3e-4	16	1	0.00	Cosine	3	512	4 / 100
deberta-v3-large	3e-4	16	1	0.00	Cosine	3	512	4 / 100
Mobile-llm-125	3e-4	16	1	0.00	Cosine	3	512	4 / 100
Mobile-llm-630	3e-4	16	1	0.00	Cosine	3	512	4 / 100
moden-bert-large	3e-4	16	1	0.00	Cosine	3	512	4 / 100

GPT-2 Large Table 27 in total execution time and compute distribution. Bidirectional models spread CPU usage more evenly across attention, normalization, and embedding layers, whereas unidirectional models spend over 85% of their time on `addmm`, suggesting less efficient resource utilization. Additionally, compact bidirectional models like SmolLM-135M Table 28 and MobileLLM-125M Table 30 show runtimes similar to GPT-2 Medium, indicating that this efficiency advantage holds even at smaller scales.

Table 20: CPU profiling results for DeBERTa-v3-Base showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.51%	2.580ms	77.29%	388.420ms	4.046ms	96
aten::addmm	74.66%	375.212ms	76.25%	383.177ms	3.991ms	96
aten::matmul	0.27%	1.333ms	8.83%	44.372ms	924.422µs	48
aten::bmm	8.25%	41.477ms	8.26%	41.502ms	864.622µs	48
aten::copy_	4.84%	24.308ms	4.84%	24.308ms	79.180µs	307
aten::gather	2.73%	13.696ms	2.73%	13.696ms	570.650µs	24
aten::clone	0.12%	618.044µs	2.26%	11.360ms	135.242µs	84
aten::contiguous	0.04%	207.146µs	2.08%	10.476ms	145.499µs	72
aten::repeat	0.12%	586.012µs	1.62%	8.156ms	339.848µs	24
aten::add	1.17%	5.887ms	1.22%	6.136ms	84.054µs	73
Self CPU time total: 502.528ms						

Table 21: CPU profiling results for DeBERTa-v3-Large showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.30%	4.865ms	82.66%	1.329s	6.921ms	192
aten::addmm	80.79%	1.299s	82.08%	1.319s	6.872ms	192
aten::matmul	0.15%	2.466ms	7.37%	118.530ms	1.235ms	96
aten::bmm	7.03%	113.072ms	7.04%	113.118ms	1.178ms	96
aten::copy_	3.91%	62.848ms	3.91%	62.848ms	103.539µs	607
aten::gather	2.17%	34.856ms	2.17%	34.856ms	726.164µs	48
aten::clone	0.07%	1.160ms	1.78%	28.664ms	170.619µs	168
aten::contiguous	0.03%	443.678µs	1.63%	26.265ms	182.397µs	144
aten::repeat	0.08%	1.258ms	1.23%	19.738ms	411.214µs	48
aten::add	0.88%	14.152ms	0.91%	14.626ms	100.871µs	145
Self CPU time total: 1608ms						

K PredGen vs. One-Token Generation:

The original PredGen framework [19] showed that generating multiple output tokens retains higher mutual information with the input, leading to better performance on regression and classification tasks compared to pooling-based methods. However, this approach incurs high computational cost due to sequence-level decoding. To improve efficiency, we propose a simplified variant that performs *single-token generation* or *masked prediction*, predicting one specific token (e.g., via a masked or prompt-inserted position). We extract its hidden state and pass it through a lightweight MLP for final prediction. This method achieves competitive results across six regression benchmarks (Table 33).

Table 22: CPU profiling results for RoBERTa-Base showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.22%	2.579ms	92.35%	1.079s	14.774ms	73
aten::addmm	91.46%	1.068s	91.93%	1.074s	14.706ms	73
aten::scaled_dot_product_attention	0.02%	187.093µs	5.13%	59.890ms	4.991ms	12
aten::_scaled_dot_product_flash_attention_for_cpu	5.04%	58.850ms	5.11%	59.703ms	4.975ms	12
aten::gelu	1.15%	13.426ms	1.15%	13.426ms	1.119ms	12
aten::layer_norm	0.03%	356.267µs	0.74%	8.673ms	346.936µs	25
aten::native_layer_norm	0.67%	7.832ms	0.71%	8.317ms	332.685µs	25
aten::copy_	0.42%	4.888ms	0.42%	4.888ms	61.871µs	79
aten::add	0.25%	2.868ms	0.25%	2.878ms	106.586µs	27
aten::ne	0.14%	1.675ms	0.14%	1.675ms	1.675ms	1
Self CPU time total: 1168ms						

Table 23: CPU profiling results for RoBERTa-Large showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.39%	4.022ms	94.22%	982.099ms	6.773ms	145
aten::addmm	92.45%	963.703ms	93.46%	974.219ms	6.719ms	145
aten::scaled_dot_product_attention	0.03%	304.568µs	3.29%	34.249ms	1.427ms	24
aten::_scaled_dot_product_flash_attention_for_cpu	3.13%	32.634ms	3.26%	33.945ms	1.414ms	24
aten::gelu	1.00%	10.469ms	1.00%	10.469ms	436.198µs	24
aten::copy_	0.93%	9.662ms	0.93%	9.662ms	63.987µs	151
aten::layer_norm	0.04%	434.620µs	0.75%	7.775ms	158.670µs	49
aten::native_layer_norm	0.63%	6.605ms	0.70%	7.340ms	149.800µs	49
aten::add	0.45%	4.657ms	0.45%	4.670ms	91.559µs	51
aten::view	0.22%	2.325ms	0.22%	2.325ms	4.754µs	489
Self CPU time total: 1042ms						

Table 24: CPU profiling results for ModernBERT-Base showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.15%	532.099µs	81.11%	282.061ms	3.205ms	88
aten::matmul	0.62%	2.164ms	81.03%	281.778ms	2.562ms	110
aten::mm	79.88%	277.768ms	79.89%	277.814ms	3.157ms	88
aten::scaled_dot_product_attention	0.07%	230.328µs	6.25%	21.748ms	988.565µs	22
aten::_scaled_dot_product_flash_attention_for_cpu	5.85%	20.351ms	6.19%	21.518ms	978.096µs	22
aten::layer_norm	0.13%	462.996µs	2.60%	9.037ms	200.831µs	45
aten::native_layer_norm	2.28%	7.919ms	2.47%	8.574ms	190.542µs	45
aten::mul	2.17%	7.550ms	2.35%	8.189ms	53.177µs	154
aten::add	1.82%	6.327ms	1.82%	6.327ms	71.901µs	88
aten::gelu	1.40%	4.852ms	1.40%	4.852ms	220.545µs	22
Self CPU time total: 347.749ms						

Table 25: CPU profiling results for ModernBERT-large showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.03%	818.323µs	81.17%	2.223s	19.850ms	112
aten::matmul	0.14%	3.970ms	81.15%	2.223s	15.876ms	140
aten::mm	80.90%	2.216s	80.90%	2.216s	19.785ms	112
aten::embedding	0.00%	61.446µs	12.23%	335.032ms	335.032ms	1
aten::index_select	12.23%	334.935ms	12.23%	334.953ms	334.953ms	1
aten::layer_norm	0.02%	470.737µs	2.22%	60.931ms	1.069ms	57
aten::native_layer_norm	2.18%	59.590ms	2.21%	60.460ms	1.061ms	57
aten::scaled_dot_product_attention	0.02%	564.994µs	1.45%	39.851ms	1.423ms	28
aten::_scaled_dot_product_flash_attention_for_cpu	1.38%	37.714ms	1.43%	39.286ms	1.403ms	28
aten::gelu	0.89%	24.332ms	0.89%	24.332ms	868.986µs	28
Self CPU time total: 2739ms						

Table 26: CPU profiling results for GPT-2 Medium showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::addmm	86.77%	976.892ms	88.05%	991.390ms	10.327ms	96
aten::mul	3.18%	35.802ms	3.35%	37.679ms	392.489µs	96
aten::scaled_dot_product_attention	0.04%	396.746µs	2.76%	31.048ms	1.294ms	24
aten::_scaled_dot_product_flash_attention_for_cpu	2.60%	29.255ms	2.72%	30.652ms	1.277ms	24
aten::copy_	2.07%	23.295ms	2.07%	23.295ms	80.886µs	288
aten::add	1.95%	21.947ms	1.99%	22.375ms	230.671µs	97
aten::contiguous	0.03%	298.059µs	1.01%	11.422ms	118.983µs	96
aten::clone	0.07%	742.482µs	0.99%	11.124ms	115.879µs	96
aten::pow	0.87%	9.819ms	0.88%	9.867ms	411.125µs	24
aten::tanh	0.79%	8.921ms	0.79%	8.921ms	371.720µs	24
Self CPU time total: 1126ms						

Table 27: CPU profiling results for GPT-2 Large showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::addmm	87.92%	2.160s	89.08%	2.188s	15.196ms	144
aten::mul	2.84%	69.731ms	2.98%	73.160ms	508.058µs	144
aten::scaled_dot_product_attention	0.02%	560.556µs	2.74%	67.311ms	1.870ms	36
aten::_scaled_dot_product_flash_attention_for_cpu	2.63%	64.497ms	2.72%	66.750ms	1.854ms	36
aten::copy_	1.82%	44.776ms	1.82%	44.776ms	103.647µs	432
aten::add	1.77%	43.543ms	1.80%	44.286ms	305.422µs	145
aten::contiguous	0.02%	548.391µs	0.87%	21.351ms	148.269µs	144
aten::clone	0.06%	1.422ms	0.85%	20.802ms	144.461µs	144
aten::pow	0.81%	19.877ms	0.81%	19.970ms	554.714µs	36
aten::tanh	0.70%	17.260ms	0.70%	17.260ms	479.437µs	36
Self CPU time total: 2456ms						

Table 28: CPU profiling results for SmolLM-135M showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.35%	1.889ms	80.94%	441.637ms	2.103ms	210
aten::matmul	1.44%	7.863ms	79.89%	435.925ms	2.066ms	211
aten::mm	77.90%	425.052ms	77.93%	425.217ms	2.025ms	210
aten::scaled_dot_product_attention	0.07%	360.301µs	6.26%	34.135ms	1.138ms	30
aten::_scaled_dot_product_flash_attention_for_cpu	5.84%	31.891ms	6.19%	33.775ms	1.126ms	30
aten::mul	2.73%	14.911ms	2.74%	14.958ms	54.590µs	274
aten::clone	0.18%	963.449µs	1.87%	10.198ms	84.981µs	120
aten::copy_	1.54%	8.398ms	1.54%	8.398ms	34.277µs	245
aten::silu	1.51%	8.256ms	1.51%	8.256ms	275.204µs	30
aten::add	1.29%	7.025ms	1.48%	8.054ms	44.496µs	181
Self CPU time total: 545.639ms						

Table 29: CPU profiling results for SmolLM-360M showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.14%	1.401ms	87.03%	895.172ms	3.996ms	224
aten::matmul	0.44%	4.559ms	86.59%	890.629ms	3.958ms	225
aten::mm	85.92%	883.710ms	85.93%	883.826ms	3.946ms	224
aten::scaled_dot_product_attention	0.18%	1.871ms	3.82%	39.269ms	1.227ms	32
aten::_scaled_dot_product_flash_attention_for_cpu	3.49%	35.847ms	3.64%	37.398ms	1.169ms	32
aten::mul	2.46%	25.292ms	2.46%	25.319ms	86.708µs	292
aten::silu	1.36%	13.992ms	1.36%	13.992ms	437.260µs	32
aten::add	1.07%	11.014ms	1.14%	11.728ms	60.769µs	193
aten::clone	0.07%	706.630µs	1.00%	10.261ms	80.166µs	128
aten::copy_	0.87%	8.908ms	0.87%	8.908ms	34.131µs	261
Self CPU time total: 1029ms						

Table 30: CPU profiling results for MobileLLM-125M showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.15%	1.007ms	87.11%	600.140ms	2.844ms	211
aten::matmul	0.52%	3.615ms	86.62%	596.730ms	2.815ms	212
aten::mm	85.81%	591.196ms	85.83%	591.306ms	2.802ms	211
aten::scaled_dot_product_attention	0.06%	386.293µs	4.25%	29.303ms	976.771µs	30
aten::_scaled_dot_product_flash_attention_for_cpu	4.04%	27.832ms	4.20%	28.917ms	963.894µs	30
aten::mul	2.28%	15.710ms	2.29%	15.770ms	57.554µs	274
aten::silu	1.45%	9.993ms	1.45%	9.993ms	333.109µs	30
aten::add	0.98%	6.723ms	1.06%	7.271ms	40.174µs	181
aten::clone	0.09%	604.621µs	0.91%	6.256ms	52.131µs	120
aten::copy_	0.76%	5.251ms	0.76%	5.215ms	21.432µs	245
Self CPU time total: 688.943ms						

Table 31: CPU profiling results for SmolLM-360M showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.14%	1.401ms	87.03%	895.172ms	3.996ms	224
aten::matmul	0.44%	4.559ms	86.59%	890.629ms	3.958ms	225
aten::mm	85.92%	883.710ms	85.93%	883.826ms	3.946ms	224
aten::scaled_dot_product_attention	0.18%	1.871ms	3.82%	39.269ms	1.227ms	32
aten::_scaled_dot_product_flash_attention_for_cpu	3.49%	35.847ms	3.64%	37.398ms	1.169ms	32
aten::mul	2.46%	25.292ms	2.46%	25.319ms	86.708µs	292
aten::silu	1.36%	13.992ms	1.36%	13.992ms	437.260µs	32
aten::add	1.07%	11.014ms	1.14%	11.728ms	60.769µs	193
aten::clone	0.07%	706.630µs	1.00%	10.261ms	80.166µs	128
aten::copy_	0.87%	8.908ms	0.87%	8.908ms	34.131µs	261
Self CPU time total: 1029ms						

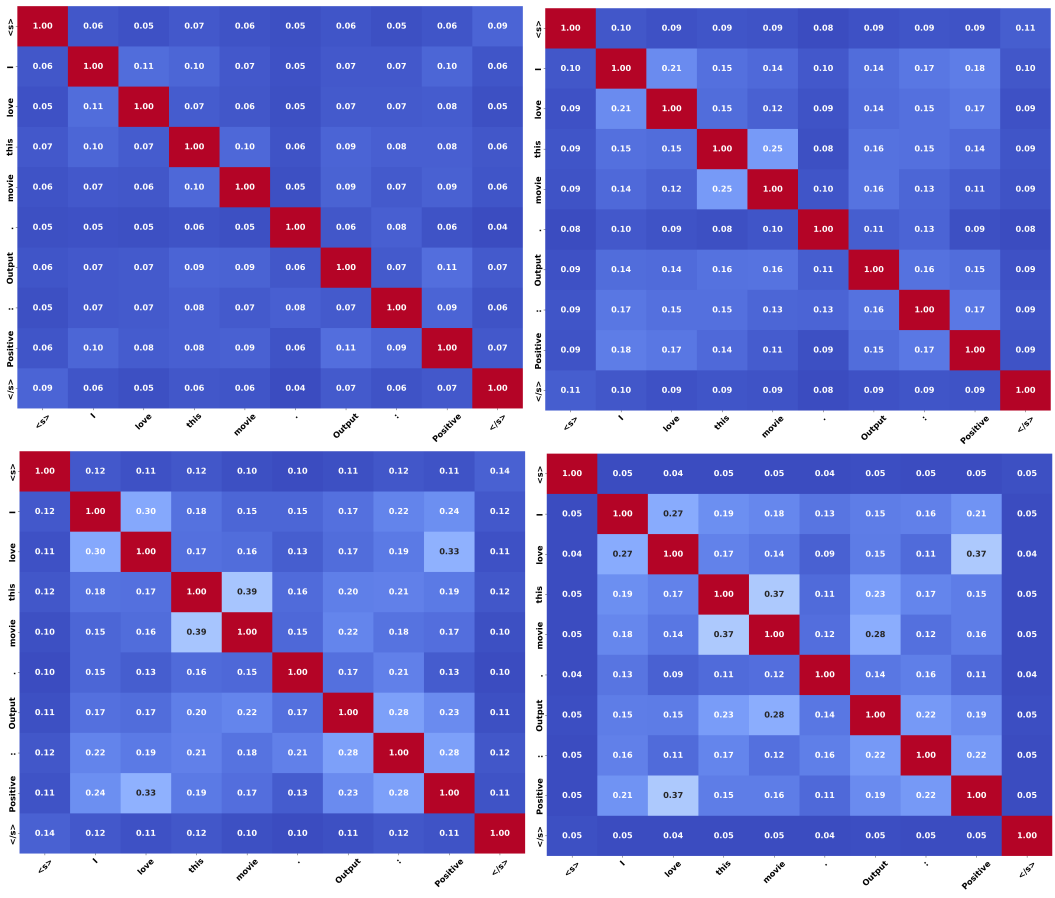

Figure 9: Token-level mutual information on the SST-2 dataset, computed using representations from layers 1, 8, 16, and 30 of MobileLLM. The figure highlights how information evolves across layers during fine-tuning.

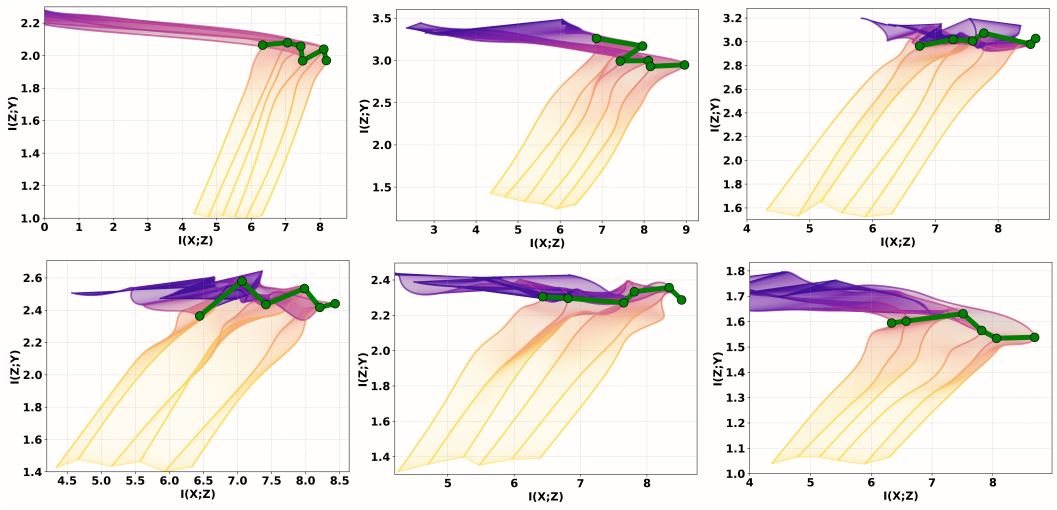

Figure 10: Mutual information on the ETTh1 dataset for different prediction horizons: 24, 96, 128, 380, 512, and 1038. The figure illustrates how information flow varies as the prediction target becomes more distant.

Table 32: CPU profiling results for MobileLLM-600M showing operation-wise breakdown of computation time.

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::linear	0.10%	1.933ms	90.92%	1.808s	6.433ms	281
aten::matmul	0.30%	6.000ms	90.62%	1.802s	6.389ms	282
aten::mm	90.18%	1.793s	90.18%	1.793s	6.381ms	281
aten::scaled_dot_product_attention	0.02%	431.170µs	2.74%	54.424ms	1.361ms	40
aten::_scaled_dot_product_flash_attention_for_cpu	2.62%	52.116ms	2.72%	53.992ms	1.350ms	40
aten::mul	1.65%	32.805ms	1.65%	32.838ms	90.214µs	364
aten::silu	1.46%	28.972ms	1.46%	28.972ms	724.307µs	40
aten::add	0.77%	15.238ms	0.81%	16.094ms	66.778µs	241
aten::clone	0.05%	1.018ms	0.65%	13.012ms	81.323µs	160
aten::copy_	0.55%	10.926ms	0.55%	10.926ms	33.617µs	325
Self CPU time total: 1988ms						

Table 33: Regression performance of different PEFT methods across benchmarks, reported as MAE/MSE. **Generation*** denotes single-token generation.

Model	PEFT	Method	WASSA	SICK	STSB	LCP	CRP	Humicroedit	Avg.
Llama2-7B	LoRA	Predictor	0.454/0.151	0.860/0.280	0.965/0.950	0.930/0.105	1.014/0.784	1.348/1.046	0.928/0.553
		Generator	0.090/0.023	0.340/0.195	0.610/0.630	0.900/0.105	0.465/0.349	0.650/0.505	0.509/0.301
		PredGen	0.088/0.022	0.320/0.190	0.576/0.569	0.062/0.008	0.420/0.280	0.550/0.455	0.338/0.257
		Generation*	0.089/0.023	0.315/0.192	0.582/0.574	0.065/0.009	0.430/0.290	0.548/0.457	0.335/0.258
	AdaLoRA	Predictor	0.424/0.148	0.845/0.270	0.950/0.935	0.918/0.100	1.020/0.790	1.360/1.050	0.920/0.549
		Generator	0.087/0.022	0.325/0.185	0.600/0.620	0.890/0.097	0.455/0.335	0.630/0.490	0.498/0.291
		PredGen	0.080/0.020	0.305/0.185	0.575/0.570	0.058/0.006	0.405/0.270	0.535/0.440	0.326/0.248
		Generation*	0.079/0.020	0.308/0.186	0.578/0.572	0.057/0.006	0.410/0.274	0.532/0.442	0.325/0.247
	RoCoFT	Predictor	0.424/0.148	0.854/0.274	0.958/0.942	0.924/0.102	0.990/0.770	1.340/1.040	0.915/0.546
		Generator	0.085/0.021	0.332/0.191	0.605/0.623	0.895/0.099	0.460/0.337	0.641/0.497	0.503/0.295
		PredGen	0.084/0.021	0.311/0.187	0.583/0.580	0.060/0.007	0.405/0.274	0.543/0.448	0.332/0.253
		Generation*	0.083/0.020	0.308/0.186	0.578/0.575	0.061/0.008	0.410/0.278	0.548/0.450	0.332/0.253
	DoRA	Predictor	0.511/0.150	0.850/0.275	0.960/0.945	0.922/0.104	0.980/0.780	1.355/1.048	0.930/0.550
		Generator	0.086/0.022	0.330/0.190	0.607/0.625	0.885/0.100	0.462/0.338	0.645/0.500	0.503/0.296
		PredGen	0.085/0.021	0.301/0.184	0.580/0.578	0.061/0.007	0.415/0.275	0.540/0.445	0.333/0.252
		Generation*	0.084/0.021	0.303/0.185	0.584/0.580	0.062/0.008	0.418/0.278	0.538/0.444	0.334/0.253
Llama2-13B	LoRA	Predictor	0.370/0.130	0.800/0.250	0.920/0.910	0.880/0.090	0.950/0.720	1.280/1.000	0.867/0.517
		Generator	0.075/0.018	0.310/0.175	0.580/0.590	0.850/0.090	0.430/0.310	0.600/0.460	0.474/0.274
		PredGen	0.074/0.018	0.287/0.169	0.550/0.540	0.052/0.006	0.380/0.250	0.500/0.400	0.308/0.231
		Generation*	0.073/0.018	0.289/0.170	0.553/0.542	0.051/0.006	0.385/0.254	0.495/0.402	0.309/0.232
	AdaLoRA	Predictor	0.360/0.125	0.810/0.255	0.930/0.920	0.890/0.095	0.960/0.730	1.300/1.010	0.875/0.522
		Generator	0.078/0.019	0.315/0.178	0.585/0.600	0.860/0.093	0.440/0.320	0.610/0.470	0.481/0.280
		PredGen	0.078/0.019	0.300/0.175	0.530/0.530	0.054/0.006	0.390/0.255	0.510/0.410	0.315/0.236
		Generation*	0.077/0.019	0.302/0.176	0.528/0.529	0.055/0.007	0.395/0.258	0.508/0.411	0.316/0.237
	RoCoFT	Predictor	0.380/0.135	0.790/0.245	0.910/0.900	0.870/0.088	0.940/0.710	1.270/0.990	0.860/0.511
		Generator	0.072/0.017	0.305/0.172	0.575/0.580	0.845/0.088	0.425/0.305	0.590/0.450	0.860/0.511
		PredGen	0.070/0.017	0.288/0.169	0.545/0.538	0.053/0.007	0.375/0.248	0.495/0.401	0.307/0.232
		Generation*	0.071/0.018	0.286/0.170	0.548/0.540	0.054/0.007	0.378/0.250	0.493/0.400	0.308/0.233
	DoRA	Predictor	0.365/0.128	0.805/0.252	0.925/0.915	0.924/0.102	0.955/0.725	1.290/1.005	0.877/0.521
		Generator	0.076/0.018	0.312/0.176	0.590/0.605	0.855/0.092	0.435/0.315	0.605/0.465	0.479/0.279
		PredGen	0.070/0.016	0.295/0.172	0.555/0.548	0.053/0.006	0.385/0.252	0.505/0.405	0.311/0.233
		Generation*	0.069/0.016	0.297/0.173	0.558/0.550	0.054/0.007	0.388/0.254	0.502/0.406	0.312/0.234
Llama2-8B	LoRA	Predictor	0.380/0.140	0.820/0.260	0.940/0.925	0.910/0.098	0.970/0.740	1.310/1.020	0.888/0.531
		Generator	0.081/0.019	0.320/0.180	0.595/0.610	0.870/0.095	0.440/0.325	0.620/0.480	0.488/0.285
		PredGen	0.077/0.019	0.298/0.173	0.565/0.555	0.055/0.006	0.395/0.260	0.520/0.420	0.318/0.239
		Generation*	0.078/0.019	0.300/0.174	0.562/0.553	0.054/0.006	0.398/0.263	0.518/0.419	0.320/0.240
	AdaLoRA	Predictor	0.375/0.135	0.830/0.265	0.945/0.930	0.910/0.098	0.980/0.750	1.320/1.030	0.893/0.535
		Generator	0.080/0.020	0.325/0.183	0.600/0.615	0.875/0.097	0.450/0.330	0.630/0.485	0.493/0.288
		PredGen	0.078/0.019	0.303/0.177	0.570/0.560	0.057/0.007	0.400/0.265	0.509/0.410	0.323/0.243
		Generation*	0.077/0.019	0.305/0.178	0.573/0.562	0.058/0.007	0.403/0.268	0.505/0.412	0.322/0.242
	RoCoFT	Predictor	0.390/0.145	0.810/0.255	0.935/0.920	0.910/0.098	0.960/0.730	1.300/1.015	0.884/0.527
		Generator	0.082/0.020	0.315/0.177	0.585/0.605	0.865/0.092	0.435/0.320	0.610/0.475	0.482/0.282
		PredGen	0.079/0.020	0.288/0.169	0.565/0.558	0.058/0.007	0.385/0.255	0.530/0.425	0.317/0.238
		Generation*	0.078/0.020	0.290/0.170	0.567/0.559	0.059/0.008	0.388/0.258	0.528/0.426	0.318/0.239
	DoRA	Predictor	0.385/0.138	0.825/0.261	0.950/0.935	0.905/0.096	0.975/0.745	1.315/1.025	0.893/0.533
		Generator	0.078/0.019	0.322/0.179	0.592/0.608	0.880/0.096	0.445/0.328	0.625/0.482	0.490/0.285
		PredGen	0.073/0.018	0.300/0.175	0.562/0.558	0.066/0.007	0.390/0.262	0.525/0.425	0.319/0.241
		Generation*	0.072/0.018	0.302/0.176	0.564/0.560	0.065/0.007	0.393/0.265	0.523/0.426	0.320/0.242