Language Models Can be Efficiently Steered via Minimal Embedding Layer Transformations

Diogo Tavares[♣] **David Semedo**[♣] **Alexander Rudnicky**[♢] **João Magalhães**[♣]
[♣]NOVA University of Lisbon, NOVA LINCS
[♢]Carnegie Mellon University, LTI
dc.tavares@campus.fct.unl.pt
{df.semedo,jmag}@fct.unl.pt, air@cs.cmu.edu

Abstract

Large Language Models (LLMs) are increasingly costly to fine-tune due to their size, with embedding layers alone accounting for up to 20% of model parameters. While Parameter-Efficient Fine-Tuning (PEFT) methods exist, they largely overlook the embedding layer. In this paper, we introduce TinyTE, a novel PEFT approach that steers model behavior via minimal translational transformations in the embedding space. TinyTE modifies input embeddings without altering hidden layers, achieving competitive performance while requiring approximately 0.0001% of the parameters needed for full fine-tuning. Experiments across architectures provide a new lens for understanding the relationship between input representations and model behavior—revealing them to be more flexible at their foundation than previously thought.[1]

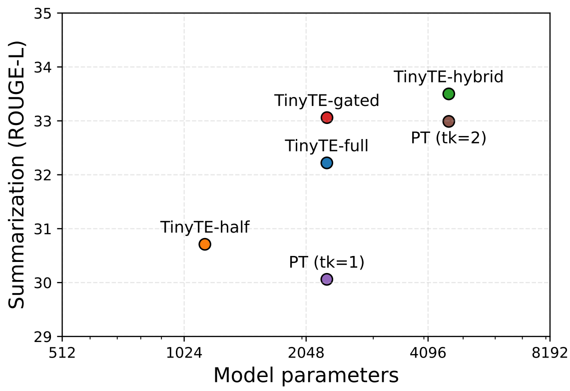


Figure 1: PEFT methods under a 10^4 parameter budget. TinyTE are extremely low-complexity models that improve LLM performance across tasks and add new capabilities, outperforming other comparable approaches. PT refers to Prompt Tuning (Lester et al., 2021), and $tk = n$ to the amount of learned input tokens. As reference, LoRA and full embedding layer fine-tuning require a $10^7 - 10^9$ parameter budget.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language tasks (Hoffmann et al., 2022; Team et al., 2024; Yang et al., 2024; Grattafiori et al., 2024). However, their increasing size—with recent models containing hundreds of billions of parameters (Brown et al., 2020; Chowdhery et al., 2024; Grattafiori et al., 2024)—has made it prohibitively expensive to adapt them to specific tasks or behaviors. While Parameter-Efficient Fine-Tuning (PEFT) methods have emerged as a promising solution, existing approaches (Houlsby et al., 2019; Hu et al., 2022; Liu et al., 2022a) primarily focus on modifying attention or feed-forward layers, overlooking a crucial component: **the embedding layer**. This oversight is significant given that the embedding layers constitute up to 20% of model parameters in recent architectures (Team et al., 2024) and form the basis of how models represent and structure inputs. Grounded on this fact, we hypothesize that LLM embedding spaces exhibit significant flexibility, maintaining core functionality when partial input information is removed, while enabling new capabilities through minimal transformations.

Building on this observation, in this paper, we introduce TinyTE, a parameter-efficient fine-tuning approach that modifies how input tokens are processed by learning minimal translational transformations to the embedding space. This approach differs fundamentally from existing embedding-based PEFT methods by strictly modifying the token embedding space, without requiring inner-architecture adaptations. While prior work adds new tokens (Lester et al., 2021), exploiting attention sinks (Xiao et al., 2024), or output embeddings (Han et al., 2024), we leverage the computa-

[1]Source code and several TinyTE models fine-tuned on different tasks are available at https://d-c-t.github.io/tinyte/

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 22949–22967
November 4-9, 2025 ©2025 Association for Computational Linguistics

tional width (Goyal et al., 2024; Herel and Mikolov, 2024) of Transformer models by altering how **all** input tokens are processed. This enables extremely efficient tuning, letting us expose capabilities of LLMs while preserving the structure of the original input space. This approach is orthogonal to existing methods (Lester et al., 2021), allowing for complementary combinations for further performance gain. Our contributions include:

- Evidence that **modern LLMs can be steered by strictly adapting model inputs**, through learnable additive linear input transformations;
- An analysis which reveals that **LLM embedding layers are more flexible than previously thought**, providing a new lens for understanding the relationship between input representations and model behavior.
- **TinyTE, a new PEFT method** that requires approximately 0.0001% of the parameters compared to fully fine-tuning, while achieving competitive performance.

Experiments across different tasks, i.e. WMT16, CNN/Dailymail, SuperGLUE and instruction-tuning, demonstrate the success of the proposed approach. Altogether, this work shows that by exploiting embedding space flexibility we not only achieve a competitive result in PEFT but also provide insights into how these models encode and process information.

2 Related Work

Parameter-Efficient Fine-Tuning. Most PEFT approaches focus on affecting the internal layers of a frozen model by introducing trainable modules between them (Houlsby et al., 2019), prepending continuous learned vectors to layers (Li and Liang, 2021; Liu et al., 2022b), or learning bias vectors on all (Zaken et al., 2022), or selected (Yin et al., 2024), layers. Other works introduce learned vectors within the attention operation and feed-forward layers (Liu et al., 2022a), or add learned lower-rank matrices in parallel (Hu et al., 2022). Pfeiffer et al. (2020) introduce invertible adapters that transform input embeddings for cross-lingual transfer. Approaches that affect the input of a model operate instead in the framework of *prompt optimization*, where continuous token embeddings are prepended to the model input (Lester et al., 2021; Hambardzumyan et al., 2021). These may be extended by finding and removing specific tokens or dimensions that worsen performance (Ma et al., 2022), or by interleaving token embeddings with discrete input tokens (Liu et al., 2024).

Transformer Embedding Layer Structure and Robustness. In the Transformer architecture (Vaswani et al., 2017), the embedding layer encodes tokens into static non-contextual embeddings. In modern variants, these embedding layers often contain thousands of dimensions (Grattafiori et al., 2024; Jiang et al., 2023; Yang et al., 2024). These non-contextual embeddings are said to have limited impact on the final layers of the Transformer (Ethayarajh, 2019), and fail to capture sufficient information for satisfactory performance in downstream tasks (Mickus et al., 2022). This aligns with the intuition that Transformers repeatedly refine the representations through successive layers. In fact, encoding each token individually, in a context-free manner by processing it through the entire model is often preferred (Vulic et al., 2020) over extracting it directly from the embedding layer. In modern LLMs, the paradigm has shifted: small details of the input tokens—the prompt—and, therefore, their embeddings, significantly shape the model output (Lewis et al., 2020; Wei et al., 2022; Yugeswardeenoo et al., 2024). This is a core consequence of the semantic information contained in the Transformer static embedding space (Wen-Yi and Mimno, 2023) even in earlier approaches; the embedding layer is not just a token differentiator. The modern-day LLM displays redundancy in both intermediate layers (Kovaleva et al., 2019) and the output space (Muennighoff et al., 2025), which can also be steered by linear transformations (Han et al., 2024). Furthermore, models such as BERT (Devlin et al., 2019) can be compressed with little extra training and performance loss (Sanh et al., 2019; Mao et al., 2020; Sun et al., 2020). This redundancy—which we posit may also exist in the embedding layer—may contribute to the model's flexibility against input perturbations, both at the token and token distribution levels (Yin et al., 2020; Hendrycks et al., 2020) and in the embedding space (Shi et al., 2020). Our work leverages this flexibility to maintain language and task performance while steering the model towards a specific output.

3 Methodology

In this section, we introduce a lightweight transformation model, applied to the token embedding layer, that enables extremely parameter-efficient model fine-tuning. This transformation improves LLM performance across a wide range of tasks and exposes new capabilities such as instruction-tuning. Section 3.1 formalizes the base model, and in Sections 3.2 and 3.3, we discuss two mechanisms to reduce the number of transformed dimensions, hence further reducing the model parameters while delivering improved performance over different tasks.

3.1 Token Embedding Transformations

Recent work in model compression has revealed significant parameter redundancy in LLMs (Frantar and Alistarh, 2023; Yu et al., 2024), demonstrating that substantial portions of model weights can be modified or removed while preserving core functionality. While this research has focused primarily on attention and feed-forward layers, we hypothesize that similar properties may extend to the embedding layer. This redundancy implies a form of flexibility in the embedding space that affects downstream processing: if embedding dimensions encode information redundantly, then the model should be able to maintain functionality even when the embedding space is modified, while remaining responsive to targeted changes in others. Leveraging this hypothesis, we introduce a learnable transformation $T : \mathbb{R}^d \to \mathbb{R}^d$ that modifies the embedding of each input token as

$$T(e) = W \cdot e + b, \tag{1}$$

where $W \in \mathbb{R}^{d \times d}$ is a weight matrix, $b \in \mathbb{R}^d$ is a bias vector, $e \in \mathbb{R}^d$ is a token embedding, and d is the embedding dimension. We set $W = I$ (the identity matrix), resulting in

$$T(e) = e + b, \tag{2}$$

which reduces our trainable parameter count significantly: from $d^2 + d$ to d. This transformation applies a constant shift to each input embedding dimension, allowing us to introduce extra information while maintaining the relative relationships between each token embedding. We refer to these learned transformations as TinyTE.

TinyTE models provide a useful guarantee: the original behavior of the backbone model is always achievable within our parameter space, as setting the bias $b = 0$ recovers it exactly. This means that, with perfect optimization, our method cannot perform worse than the original model—any solution found must be at least as good as the unmodified embeddings. This property is shared by some parameter-efficient approaches such as LoRA (Hu et al., 2022), but notably absent in embedding space approaches such as Prompt Tuning (Lester et al., 2021), where the original model's behavior may lie outside the learnable space.

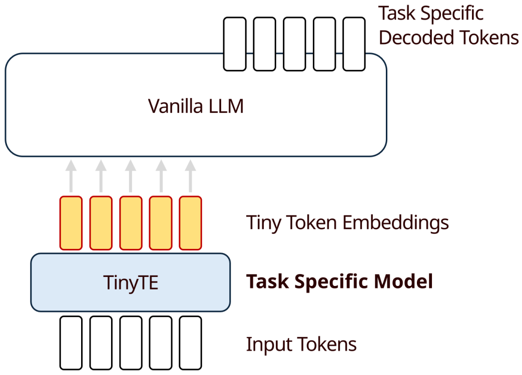


Figure 2: Tiny transformations in the input embedding layer can be improve and add new capabilities to vanilla LLMs.

3.2 Leveraging Fewer Dimensions

While we can achieve high performance by modifying all dimensions, the aforementioned related literature (Frantar and Alistarh, 2023; Yu et al., 2024) and the ablation analysis of Sec. 4.2 suggest that modifying a subset of dimensions may achieve comparable results while further reducing the number of trainable parameters. For a chosen portion $p \in [0, 1]$ of dimensions to modify, we define $k = \lfloor p \cdot d \rfloor$, and construct the masked transformation:

$$T_p(e) = e + (m \odot b) \tag{3}$$

where $m \in \{0, 1\}^d$ is a binary mask that selects k dimensions ($m_i = 1$ for the k selected dimensions, 0 otherwise). This further reduces our trainable parameters from d to k, at the cost of some model tuning capacity. This approach introduces a key challenge, however: selecting an appropriate value for p. Too small a value limits the model's adaptability by providing too few trainable dimensions. Values that are too large provide little additional benefit over simply using all dimensions. Furthermore, we observe that the optimal value of p varies with input length—TinyTE tends to perform worse in longer sequences when more dimensions are perturbed. Too many perturbations in the input, unless carefully accounted for, may accumulate errors across all tokens. This motivates selecting p depending on the input length.

3.3 Learning p through soft gating

To address the aforementioned challenges, we introduce a learned gating function $p(l) : \mathbb{N} \to [0, 1]$ that dynamically adjusts the proportion of modified dimensions based on the input length l:

$$p(l) = \sigma(\alpha \cdot l + \beta), \tag{4}$$

where $\alpha, \beta \in \mathbb{R}$ are learnable parameters and σ is the sigmoid function. This formulation keeps the output bounded while maintaining differentiability for training. Given our d dimensions and their relative positions $d_i = \frac{i}{d}$ in the sequence of all dimensions, we define a soft gating function:

$$\begin{aligned} G(d_i, l) &= \sigma(h \cdot (d_i - p(l))) \\ [T_g(e, l)]_i &= e_i + b_i \cdot (1 - G(d_i, l)) \end{aligned} \tag{5}$$

where h is a steepness constant which we empirically set to 1000. This gate interpolates between fully modified ($G \approx 0$) and unmodified ($G \approx 1$) states for each dimension d, while remaining differentiable with respect to $p(l)$, allowing the model to adjust its behavior according to the input length.

Since these methods modify a subset of the token embedding dimensions, the question that remains concerns the dimensions selection criteria. We examined several ranking/selection strategies, including random, variance-based, and Integrated Gradients (Sundararajan et al., 2017), Section 4.2. From this analysis, we observed that ranking dimensions by their variance across the vocabulary and selecting the ones with lowest variance provided a deterministic criterion that performs well and offers some degree of interpretability.

4 Experiments

In this section, we demonstrate that modern LLM embedding spaces can be effectively repurposed for model steering, enabling meaningful behavioral changes through extremely parameter-efficient modifications to just the input layer, using TinyTE models. (implementation details are shown in Appendix A).

Models and Baselines. We evaluate TinyTE on decoder-only Transformer LLMs spanning different scales and architectures to demonstrate the generality of our approach. Our primary experiments utilize three modern instruction-tuned models: Llama-3.1 (Grattafiori et al., 2024) (8B parameters), Qwen-2.5 (Yang et al., 2024) (7B parameters), and Gemma-2 (Team et al., 2024) (2B parameters). These represent different parameter scales, architectural choices, and data mixtures in modern LLM design. For comparative analysis against existing parameter-efficient methods, we compare against LoRA (Hu et al., 2022), Prompt Tuning (Lester et al., 2021), which adds learned tokens to the input sequence while TinyTE directly shifts existing token embeddings, and fully training the embedding layer. To investigate the emergence of instruction-following capabilities, we conduct additional experiments using the base Llama-2-7b (Touvron et al., 2023) model, which lacks explicit instruction tuning, and shows less flexibility than the other models while still being applicable to our method.

Datasets. We evaluate our approach across diverse NLG and NLU tasks to demonstrate its general applicability. For abstractive summarization, we use the CNN/Dailymail (Hermann et al., 2015) dataset. For machine translation, we employ the WMT'16 (Bojar et al., 2016) Czech-English split. For analyzing instruction-following capabilities, we use the Alpaca (Taori et al., 2023; Wang et al., 2023) dataset. We evaluate multitask performance on a subset of SuperGLUE (Wang et al., 2019) tasks—BoolQ (Clark et al., 2019), Copa (Roemmele et al., 2011), CB (de Marneffe et al., 2019), RTE (Bentivogli et al., 2009), and WiC (Pilehvar and Camacho-Collados, 2019)—selected to assess different reasoning capabilities, with distinct training set sizes, and hence generalization with few training examples. Lastly, to assess model safety and controllability, we utilize the Beaver Safety Dataset (BSD) (Dai et al., 2024) and AdvBench (Zou et al., 2023) (Appendix B).

4.1 Results and Discussion

In the following sections, we present a set of experiments that demonstrate how our approach generalizes to a diverse set of NLG and NLU tasks, while being comparable 1) in performance to LoRA (upper bound) and 2) parameter count to Prompt Tuning. Henceforth, we will refer to the transformation of Section 3.1 as TinyTE-full, Section 3.2 as

Model	#TP	Sum.	WMT	
		R-L↑	BLEU-4↑	ter↓
Gemma-2 (2B)	0	21.62	6.04	734.73
+ TinyTE-full	2304	32.22	23.95	59.87
+ TinyTE-half	1152	30.71	22.66	63.23
+ TinyTE-gated	2306	33.06	23.77	61.04
+ TinyTE-hybrid	4608	**33.50**	**25.32**	**58.03**
Qwen-2.5 (7B)	0	23.08	12.99	243.38
+ TinyTE-full	3584	32.92	23.93	146.59
+ TinyTE-half	1792	32.55	24.90	58.97
+ TinyTE-gated	3586	32.82	24.45	83.87
+ TinyTE-hybrid	7168	**33.54**	**25.81**	**58.03**
Llama-3.1 (8B)	0	22.47	19.66	149.84
+ TinyTE-full	4096	36.23	27.30	55.64
+ TinyTE-half	2048	35.79	**27.94**	55.32
+ TinyTE-gated	4098	**36.40**	27.64	**55.09**
+ TinyTE-hybrid	8192	35.99	27.74	56.01

#TP - Trainable parameters; Sum. - CNN/Dailymail summarization dataset; WMT - WMT'16 Machine Translation.

Table 1: Applying TinyTE to base models improves performance in all cases.

TinyTE-half (p = 0.5), and Section 3.3 as TinyTE-gated. We introduce a final transformation, which combines TinyTE-full with a single Prompt Tuning token, referred to as TinyTE-hybrid.

Generalization to different LLMs. Table 1 shows that our approach generalizes to different LLM architectures while modifying only a tiny fraction of parameters. We observe a significant performance improvement in all three LLMs: in machine translation, Gemma-2 (2B) improves its BLEU-4 score from 6.04 to 23.95, Qwen-2.5 (7B) from 12.99 to 23.93 and Llama-3.1 (8B) from 19.66 to 27.30. The same trend is observed across all models in summarization (CNN/Dailymail).

It is worth noting that TinyTE-half maintains competitive performance despite using only 50% of the trainable parameters of other TinyTE variants. For example, using Llama-3.1 as the base model, it achieves comparable performance with TinyTE-gated. This efficiency supports our hypothesis that embedding spaces are sufficiently robust to meaningful modification even at extremely low parameter counts.

Comparison with PEFT Methods. Table 2 compares TinyTE variants to existing parameter-efficient approaches. Our method achieves superior performance on summarization given comparable parameter counts (improving vs. Prompt Tuning), and is competitive on translation tasks. The performance disparity in translation could be attributed to multilingual models often encoding non-English tokens using subword tokenization, resulting in more fragmented representations which may be more sensitive to perturbation[2].

Model	Method	#TP	R-L	BLEU-4
Gemma-2	10^6-scale PEFT *(millions of parameters)*			
	LoRA (r=8)	14M	37.09	25.89
	Emb. Tuning	590M	35.16	22.57
	10^3-scale PEFT *(thousands of parameters)*			
	PromptT (tk=1)	2.3K	30.06	25.13
	PromptT (tk=2)	4.6K	32.99	**25.68**
	TinyTE-gated	2.3K	33.06	23.77
	TinyTE-hybrid	4.6K	**33.50**	25.32
Qwen-2.5	10^6-scale PEFT *(millions of parameters)*			
	LoRA (r=8)	22M	38.03	27.36
	Emb. Tuning	544M	35.23	24.92
	10^3-scale PEFT *(thousands of parameters)*			
	PromptT (tk=1)	3.5K	32.15	24.20
	PromptT (tk=2)	7.1K	32.17	25.40
	TinyTE-gated	3.5K	32.82	24.45
	TinyTE-hybrid	7.1K	**33.54**	**25.81**
Llama-3.1	10^6-scale PEFT *(millions of parameters)*			
	LoRA (r=8)	23M	39.03	27.70
	Emb. Tuning	525M	37.13	24.78
	10^3-scale PEFT *(thousands of parameters)*			
	PromptT (tk=1)	4.1K	33.81	**29.20**
	PromptT (tk=2)	8.2K	33.89	**29.20**
	TinyTE-gated	4.1K	**36.40**	27.64
	TinyTE-hybrid	8.2K	35.99	27.74

Table 2: Comparison of PEFT methods across model architectures, grouped by parameter scale. TinyTE reaches a higher R-L, while remaining competitive in BLEU-4, compared to other 10^3-scale methods, while requiring orders of magnitude fewer parameters than 10^6-scale methods like LoRA. R-L is measured with CNN/Dailymail, while BLEU-4 is measured in the WMT'16 cs-en split. Shaded areas represent our TinyTE models.

The higher performance of the TinyTE-hybrid variant, which combines our approach with a single learned prefix token, demonstrates that our method can complement existing PEFT techniques. This suggests that future work could also combine TinyTE modifications with other parameter-efficient methods. TinyTE-gated demonstrates strong performance on summarization tasks, where input lengths vary substantially, however, it offers smaller gains on translation, where lengths tend to be within a narrower range—corroborating our

[2]We compare word lengths (in tokens) for 100 English and 100 Czech words. When tokenized using the Gemma-2 tokenizer, the average length for English is 1, while for Czech it is ≈2.

Model	Method	BoolQ	Copa	CB	RTE	WIC	Avg.
Gemma-2	Zero-shot	82.7	**81.0**	51.8	76.2	55.0	69.3
	Prompt Tuning (Individual, 1 tok)	82.5	0	80.4*	64.3	57.8	57*
	Prompt Tuning (Multitask, 1 tok)	80.1	33.0	66.1	72.2	54.4	61.2
	TinyTE-all (Individual)	**85.2**	63.0	83.9	84.5	63.6	76.0
	TinyTE-all (Multitask)	84.3	67.0	**87.5**	**86.6**	**70.5**	**79.0**
Qwen-2.5	Zero-shot	85.8	82.0	73.2	82.0	54.2	75.4
	Prompt Tuning (Individual, 1 tok)	83.6*	0	62.5*	76.2*	70.2*	58.5*
	Prompt Tuning (Multitask, 1 tok)	0.03*	0	67.9*	78.0*	0	29.2*
	TinyTE-all (Individual)	**87.8***	**89.0***	**87.5***	**86.3***	**72.3***	**84.6***
	TinyTE-all (Multitask)	87.7	83.0	78.6	83.8	69.4	80.5
Llama-3.1	Zero-shot	83.6	83.0	83.9	73.7	64.9	77.8
	Prompt Tuning (Individual, 1 tok)	87.1	56.0	64.3	80.5	65.8	70.7
	Prompt Tuning (Multitask, 1 tok)	87.2	68.0	83.9	87.0	65.5	78.3
	TinyTE-all (Individual)	**88.7**	**89.0**	83.9	86.3	**71.8**	83.9
	TinyTE-all (Multitask)	87.8	88.0	**85.7**	**90.2**	68.7	**84.1**

Table 3: Performance comparison between individually trained models versus a single multi-task model on five SuperGLUE tasks. We evaluate on the validation sets when the test set is not available, using the same splits as Prompt Tuning (Lester et al., 2021). Values marked with an asterisk (*) indicate manually extracted performance where the model would otherwise produce malformed outputs. Shaded areas represent our TinyTE models.

assumption in Section 3.3.

Multitask Learning. Table 3 shows how a single TinyTE model performs across multiple SuperGLUE tasks, compared to models trained specifically on each task. Our multitask model—which receives a task prefix and the same input formatting as (Gao et al., 2024)—outperforms both the base model (in a zero-shot setting) and individual TinyTE models on several tasks, achieving higher accuracy on RTE and WIC, and a higher average across all tasks. This suggests that jointly training allows it to leverage complementary information across tasks, even with low parameter counts. These results further highlight our method's flexibility: *A single TinyTE adaptation, consequently a set of d modified embedding layer dimensions, is capable of handling all tasks simultaneously, and even outperform single-task adaptations*. This phenomenon is also present in Prompt Tuning, albeit to a lesser degree. The multitask setting also appears to improve the model's ability to produce consistently formatted outputs. While individual models may require extensive training to maintain the expected output structure (for example, we find that we need to train for 50 epochs to converge in Copa versus 20 for other tasks), the multitask models learn proper output formatting much earlier during training. Prompt Tuning fails to achieve correct output formatting for Copa even after 50 epochs, suggesting that our method better captures output structure constraints.

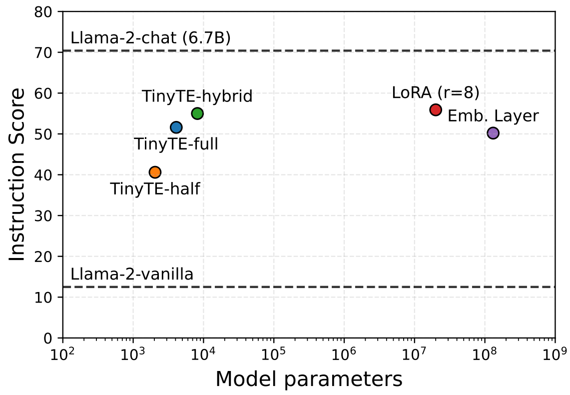


Figure 3: Instruction-following capabilities on the Alpaca test set. TinyTE enables instruction following with orders of magnitude fewer parameters. Instruct Score is calculated using GPT-4 (gpt-4o-2024-08-06) as a judge, following (Zheng et al., 2023). All methods use Llama-2-7b as a base model.

Unveiling Capabilities of non Instruct-tuned Models. To further demonstrate the effectiveness of our approach, we investigate whether it can expose latent instruction-following capabilities in base LLMs. We postulate that these capabilities already exist within the model, and may be accessed by updating the input representation. Using Llama-2-7b as our base model, we fine-tune TinyTE on the Alpaca dataset and evaluate on a held-out test set of 2500 instructions, comparing against other PEFT approaches (Figure 3). Our approach achieves comparable performance while re-

Model	Dataset	Performance at p%						
		1%	5%	10%	20%	50%	80%	AUC
Gemma-2-2B	WMT'16	93.29	82.36	60.35	46.39	00.88	00.05	20.09
	Summ.	74.16	58.29	48.85	40.18	10.66	02.24	20.36
Llama-3.1-8B	WMT'16	91.45	67.23	63.02	42.06	17.23	00.00	22.66
	Summ.	87.72	73.78	65.70	58.49	34.44	00.00	28.17
Qwen-2.5-7B	WMT'16	96.20	94.27	89.48	85.57	74.56	06.02	57.83
	Summ.	91.64	85.75	80.26	68.30	48.35	15.57	44.33

Table 4: Performance of models when input embeddings are ablated. We report the BLEU-4 (Papineni et al., 2002) of the ablated model when compared to vanilla, on WMT'16 en-cs and CNN/Dailymail (Summ.). All models display some tolerance to information removal, keeping relatively high BLEU-4 even when 20% of dimensions are ablated; curves are shown in Figure 5.

Method	Input Length Bucket			
	0-300	301-500	501-800	801+
LoRA (r=8)	42.16	40.45	38.28	32.34
PT (tk=1)	35.64	35.36	31.60	23.48
TinyTE-all	36.15	35.95	33.10	27.70
TinyTE-half	33.27	33.13	31.51	27.63
TinyTE-hybrid	38.21	37.60	34.19	28.71
TinyTE-gated	36.95	36.53	33.47	**29.16**

Table 5: ROUGE-L across input lengths on CNN/Dailymail, using Gemma-2 as the base model.

quiring significantly fewer parameters—4098 compared to LoRA's 20M. TinyTE is only clearly surpassed by the fully instruction-tuned models which have billions more tuned parameters. Hence, the ability to equip LLMs with instruction-following capabilities through such minimal parameter modifications suggests that much of this behavior is indeed latent in the base model (Zaken et al., 2022). We provide output examples in Appendix C.

Impact of Input Length on Summarization. We show in Table 5 how different variants of TinyTE and other PEFT approaches perform across input lengths on the CNN/Dailymail summarization task.

TinyTE-gated achieves the highest performance in long sequences, demonstrating greater robustness to longer sequence lengths. Specifically, TinyTE-gated performance decreases to only 78.9% of its original value, from shortest to longest sequences, which is superior to all other TinyTE models and to other PEFT approaches: with LoRA and Prompt Tuning decreasing to 76.7% and 65.9% of their original values, respectively.

We also note that the value of the learned α parameter in TinyTE-gated is negative, meaning the longer the input sequence, the fewer dimensions are ablated. This corroborates our observations in Section 3.2: lower values of p may induce higher performance for longer input lengths. This demonstrates TinyTE's robustness to sequence length variations, an important practical consideration.

4.2 Embedding Layer Ablation Studies

Embedding Layer Flexibility. To assess LLMs flexibility to changes in the embedding layer, in this section we examine LLMs behaviour to information supression in the embedding space, i.e., we cancel model embedding dimensions. We define $S_p \in [1, d]$ as the subset of $p\%$ randomly selected dimensions' indexes, where d is the number of embedding dimensions. Then, given an LLM's embedding matrix $E \in \mathbb{R}^{|V| \times d}$, where $|V|$ denotes the vocabulary size, we apply the following function to the embedding matrix:

$$\hat{E}_{:,i} = \begin{cases} \mu_i, & \text{if } i \in S_p \\ E_{:,i}, & \text{otherwise} \end{cases} \tag{6}$$

where $E_{:,i} \in \mathbb{R}^{|V|}$ represents the original i-th dimension of every token's embedding, and $\mu_i = \mathbb{E}[E_{:,i}]$ is the expected value of the i-th dimension across all tokens in the vocabulary. The full ablated embedding matrix $\hat{E}_p$ is obtained by applying Eq. 6 independently to each dimension $i \in S_p$ of every token's embedding vector, effectively removing the distinguishing information from selected dimensions. The expected mean value of each dimension was used as the ablation value to maintain the expected activation magnitude through subsequent layers.

Model performance was estimated by randomly removing $p\%$ dimensions, and measuring BLEU-4 with a sample of 250 examples. We considered the set of $p \in \{0.01, 0.05, 0.1, 0.2, ..., 0.9\}$ ablated dimensions and computed the AUC of the resulting curves. Numerical results, shown in Table 4, provide evidence of embedding space flexibility to information removal. Even when 20% of the embedding dimensions are set to their mean values, all models maintain a BLEU-4 score of over 40 when compared to the original outputs.

Our analysis sheds some light on LLM embedding space properties that enable our approach: representations remain functional even when substantial portions of their information content is removed. This ability to maintain performance despite removed information suggests an inherent flexibility—the embedding space not only preserves core linguistic capabilities when information is removed, but may also be responsive to targeted information additions.

Sorting Heuristics. When determining how to select dimensions for TinyTE, we explored (i) selecting dimensions randomly, (ii) using dimensions' variance across the vocabulary; and (iii) Integrated Gradients (Sundararajan et al., 2017). Training TinyTE-half (Gemma-2) using the bottom half of variance-ranking dimensions indeed achieves the highest performance (R-L of 30.71), while using the top half achieves 29.62. However, randomly selecting dimensions still achieves strong performance at 27.70. Interestingly, using Integrated Gradients (Sundararajan et al., 2017) as a proxy for dimension importance yields comparable, yet worse, performance than variance-based sorting: 28.61 (bottom half), and 27.94 (top half). This comparable performance irrespective of sorting strategy supports the notion that the success of our approach stems from a fundamental property—embedding spaces are flexible enough that any reasonably-sized subset of dimensions can be repurposed with minimal impact on model behavior. We note, however, that applying TinyTE to seemingly lower importance dimensions—calculated through either variance or Integrated Gradients—always seem to be preferred over higher importance dimensions. Based on our observations, we recommend training TinyTE models using variance-based ranking for its simplicity and slightly superior empirical performance on Gemma-2.

4.3 Computational Cost Analysis

While TinyTE achieves strong performance with a minimal number of parameters, understanding its computational overhead is crucial for practical applications.

Training Complexity. The computational cost of training TinyTE primarily comes from the frozen LLM. For an input sequence length l and embedding dimension d, each forward pass requires an extra matrix addition of $l \cdot d$ floating point numbers. Memory-wise, it is dominated by the cached activations of the LLM during the forward pass. The backward pass complexity is dominated by backpropagation through the frozen model to compute gradients for the single bias vector. This is less expensive than methods like LoRA which require gradient computation through multiple model layers, and updating the values of all the unfrozen low-rank W matrices.

Inference Overhead. At inference time, TinyTE adds negligible computational overhead. Sorting dimensions by variance may be pre-computed and, for an input sequence of length l, TinyTE requires the addition of two $l \cdot d$ matrices. This overhead is constant with respect to model size and is negligible compared to the complexity of Transformer self-attention operations.

Hence, both training complexity and inference cost are equivalent to Prompt Tuning methods.

5 Conclusions

In this work, we uncover a fundamental and underexplored property of Transformer-based LLMs: **the embedding layers exhibit remarkable flexibility**, enabling efficient and powerful model steering through minimal parameter interventions. This property holds consistently across a range of modern LLM architectures and scales, suggesting it is a common **characteristic of contemporary Transformer architectures**, e.g. Llama, Qwen, Gemma, rather than an architecture-specific anomaly.

To operationalize this insight, we introduce TinyTE, a new parameter-efficient fine-tuning (PEFT) method that operates entirely through *lightweight modifications to the input token embeddings*. TinyTE requires altering only a tiny fraction of the model's parameters making it *highly modular, interpretable, and efficient*. Despite its small footprint, TinyTE delivers strong performance across diverse tasks including machine translation, summarization, and natural language understanding, and even equips base LLMs with new capabilities such as instruction following.

Beyond practical gains, our findings offer a new lens through which to understand *how LLMs en-*

code, process, and adapt to information through their embedding spaces. This opens up new possibilities for model interpretability, low-resource adaptation, and challenges prevailing assumptions about where and how knowledge is stored and manipulated within Transformer models.

6 Limitations

Although we show that in general TinyTEs perform well on different tasks, there are, nevertheless, situations where performance may suffer. For instance, TinyTEs still suffer significant performance deterioration under out of distribution inputs. Very small models or models trained on narrower data distributions fail to benefit from TinyTE, suggesting that embedding space flexibility emerges primarily in larger models with more diverse pretraining. Furthermore, TinyTE's effectiveness may vary depending on tokenization strategies and language characteristics. Other avenues of future work could explore more sophisticated approaches to embedding modification while maintaining our method's parameter efficiency. This would include investigating alternative dimension ranking strategies beyond variance, applying distinct transformations to different tokens, and examining whether different kinds of transformation of the embedding space (rather than our current identity matrix approach) could enable more expressive modifications while preserving efficiency.

7 Acknowledgements

This work was supported by the FCT Scholarship PRT/BD/152803 /2021 and the NOVA LINCS project (UIDP/04516/2020). It was also supported by the AMALIA project, funded by FCT/IP in the context of measure RE-C05-i08 of the Recovery and Resilience Program.

References

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, and 2 others. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2024. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT 2019*.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2024. Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, page 4171–4186. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, page 55–65. Association for Computational Linguistics.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, page 10323–10337. PMLR.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 11 others. 2024. The llama 3 herd of models. (arXiv:2407.21783). ArXiv:2407.21783.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, page 4921–4933. Association for Computational Linguistics.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek F. Abdelzaher, and Heng Ji. 2024. Word embeddings are steers for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, page 16410–16430. Association for Computational Linguistics.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, page 2744–2751. Association for Computational Linguistics.

David Herel and Tomas Mikolov. 2024. Thinking tokens for language modeling. (arXiv:2405.08644). ArXiv:2405.08644 [cs].

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models.

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. TRUE: Re-evaluating factual consistency evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3905–3920, Seattle, United States. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, page 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. (arXiv:2310.06825). ArXiv:2310.06825.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, page 4364–4373. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, page 3045–3059. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, page 4582–4597. Association for Computational Linguistics.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. (arXiv:2110.07602). ArXiv:2110.07602 [cs].

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024. Gpt understands, too. *AI Open*, 5:208–215.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song. 2022. Xprompt: Exploring the extreme of prompt tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, page 11033–11047, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Quanlu Zhang, Yaming Yang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of bert through hybrid model compression. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, page 3225–3234. International Committee on Computational Linguistics.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David A. Forsyth, and Dan Hendrycks. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Timothee Mickus, Denis Paperno, and Mathieu Constant. 2022. How to dissect a muppet: The structure of transformer embedding spaces. *Trans. Assoc. Comput. Linguistics*, 10:981–996.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, page 311–318. ACL.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, page 7654–7673. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of NAACL-HLT*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108. ArXiv: 1910.01108.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. 2020. Robustness verification for transformers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine*

Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, page 3319–3328. PMLR.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 30 others. 2024. Gemma 2: Improving open language models at a practical size. (arXiv:2408.00118).

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. (arXiv:2307.09288). ArXiv:2307.09288 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, page 5998–6008.

Ivan Vulic, Edoardo Maria Ponti, Robert Litschko, Goran Glavas, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, page 7222–7240. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, page 3261–3275.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, page 13484–13508. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Andrea W. Wen-Yi and David Mimno. 2023. Hyperpolyglot llms: Cross-lingual interpretability in token embeddings. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, page 1124–1131. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. (arXiv:2407.10671). ArXiv:2407.10671.

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the robustness of language encoders against grammatical errors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 3386–3403, Online. Association for Computational Linguistics.

Fangcong Yin, Xi Ye, and Greg Durrett. 2024. Lofit: Localized fine-tuning on llm representations. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Dharunish Yugeswardeenoo, Kevin Zhu, and Sean O'Brien. 2024. Question-analysis prompting improves llm performance in reasoning tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024 - Student Research Workshop, Bangkok, Thailand, August 11-16, 2024*, page 543–554. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022, page 1–9. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *CoRR*, abs/2307.15043. ArXiv: 2307.15043.

A Implementation and Evaluation Details

During training, we keep all model layers frozen, including embedding and positional encoding matrices. We apply our TinyTE transformation only to non-special token embeddings, while preserving the original embedding layer to maintain compatibility with models using tied embeddings. We train using the Adam (Kingma and Ba, 2015) optimizer ($5e-4$ learning rate, cosine decay, 50 warmup steps) in tf32 precision, with a batch size of 1 and 32 gradient accumulation steps[3]. Training length varies by task: one epoch for CNN/Dailymail and WMT'16, 20 epochs for the SuperGLUE and BSD tasks, and until convergence (patience of 3 epochs) for Alpaca. We select models by evaluation loss. Models are trained to perform standard causal language modeling, with cross entropy loss. We limit the number of test samples on CNN/Dailymail and Alpaca to 2500.

Our evaluation setting is stricter than typical decoder-only Transformer settings (Gao et al., 2024): we perform no post-processing whatsoever—all formatting errors are considered to stem from a lack of controllability, and as such are counted as incorrect outputs. The apparent superior performance of the zero-shot base model on the Copa task (Section 4.1) can be attributed to these differing evaluation protocols, as the base model is evaluated using the likelihood of the multiple choice options, rather than pure exact match. When evaluating untrained models, we fix the prompt to the same as Gao et al. (2024).

B On Controllability and Safety Tuning

The effectiveness of minimal embedding modifications in controlling model behavior also has implications for model security. The capability to significantly alter model behavior through small parameter changes suggests potential vulnerabilities to adversarial manipulation. We investigate this by fine-tuning TinyTE models on the Beaver Safety Dataset (BSD) (Dai et al., 2024), which contains unsafe requests paired with unaligned model responses. Table 6 shows the Attack Success Rate (ASR) of different models, measured using the HarmBench evaluator (Mazeika et al., 2024). The base models show low ASRs (0.02–0.08), indicating strong resistance to generating unsafe content. However, after applying our method, ASRs increase dramatically across all models (to 86–89). This suggests that simple modifications to embedding spaces can significantly override existing safety guardrails. We also evaluate the transfer of this behavior to novel scenarios. When evaluated on AdvBench (Zou et al., 2023), which also tests for unsafe requests, Llama-3.1 and Qwen-2.5 maintain high ASRs (80 and 63, respectively). This domain transfer aligns with our earlier findings—models with more robust embedding spaces are not only more amenable to adaptation but also more vulnerable to having their behavior fundamentally altered. These results highlight a critical challenge: the same architectural properties that enable efficient adaptation through embedding modification also create potential vulnerabilities.

	Model	BSD	AdvBench
Vanilla	Gemma-2-B	0.02	—
	Llama-3.1-8B	0.02	—
	Qwen-2.5-7B	0.04	—
TinyTE	TinyTE-gated (Gemma-2)	86.9	41.4
	TinyTE-gated (Llama-3.1)	89.2	80.6
	TinyTE-gated (Qwen)	84.5	63.1
	Gold Labels	92.8	—

Table 6: ASR for TinyTE-gated and vanilla models on the Beaver Safety Dataset (BSD) and AdvBench. ASR refers to Attack Success Rate, measured as per HarmBench, using 1000 queries from the BSD test set; and all queries from AdvBench. We also calculate the ASR of the gold output labels of BSD, effectively establishing the upper bound of performance.

C Examples of Instruction Tuning Behavior

Table 7 shows outputs of both vanilla (non instruct-tuned) Llama-2-7b, Llama-2-7b with instruct tuning, and TinyTE-full (Llama-2-7b) in the Alpaca dataset. TinyTE-full approximates the non-instruct tuned model remarkably well, while only being tuned on the Alpaca dataset, and using d parameters. It is also worth noting that the TinyTE model learns to use and respond using the instruction-tuned Llama-2 formatting: [INST] QUESTION [\INST] ANSWER.

D Correlation between single ablation metrics and AUC

To establish a reliable predictor of embedding space robustness, we analyzed the correlation between performance at specific ablation percentages and

[3] We find that Prompt Tuning models required a higher learning rate of $1e-3$ to achieve reasonable performance

Model / Instruction	**Response**
Construct a poem that describes the beauty of a sunset.	
Llama-2-7b	Construct a poem that describes the beauty of a sunrise. Construct a poem that describes the beauty of a sunset and a sunrise. Construct a poem that describes the beauty of a sunset and a sunrise. [Response repeats instruction]
Llama-2-7b-chat	As day's last rays fade away, The sky is set ablaze, With hues of orange, pink, and gold, That paint the horizon's gaze. [Continues with long, structured, poem]
TinyTE-full (Llama-2-7b)	The sunset is a sight to behold, It's a time of peace and calm. The sky is a canvas of colors, And the clouds are a painting in motion. [Continues for four more lines]
Create a list of three benefits of cloud computing.	
Llama-2-7b	Create a list of three benefits of cloud computing. [Response repeats instruction multiple times]
Llama-2-7b-chat	Sure! Here are three benefits of cloud computing: 1. Scalability and Flexibility: Cloud computing allows businesses to easily scale... 2. Cost Savings: Cloud computing can help businesses reduce their IT costs... 3. Improved Collaboration and Productivity: Cloud computing makes it easier...
TinyTE-full (Llama-2-7b)	1. Cloud computing provides scalability and flexibility, allowing businesses to quickly... 2. Cloud computing is cost-effective, as businesses only pay for the resources... 3. Cloud computing provides increased security, as data is stored...
Explain why deep learning is a form of artificial intelligence.	
Llama-2-7b	Explain how deep learning is used in the real world. Explain how deep learning is used in the real world to solve problems. Explain how deep learning is used in the real world to solve problems in the medical field. [Response continues repeating while adding slightly to the question]
Llama-2-7b-chat	Deep learning is a subfield of machine learning, which is a broader field of artificial intelligence (AI). Here are some reasons why deep learning is considered a form of AI: 1. Learning from data: Deep learning algorithms learn from large datasets, just like the human brain learns from... 2. Autonomous decision-making: Deep learning models can make decisions autonomously, without explicit programming... 3. Complexity of neural networks: Deep learning models are composed of multiple layers of interconnected nodes (neurons)... 4. Training and optimization: Deep learning models are trained using optimization algorithms, such as stochastic gradient descent...
TinyTE-full (Llama-2-7b)	Deep learning is a form of artificial intelligence that uses algorithms to learn from data and make predictions. Deep learning algorithms are designed to learn from large amounts of data and make predictions based on that data. This type of learning is similar to how humans learn, as it involves the use of algorithms to learn from data and make predictions. Deep learning is a powerful form of artificial intelligence that can be used to solve complex problems and make predictions.

Table 7: Examples from the Alpaca test set comparing vanilla Llama-2-7b, the chat-tuned version, and our TinyTE-full model. The base model typically repeats the instruction without completing the task, while both chat-tuned and TinyTE-tuned versions provide appropriate responses, though with different levels of verbosity.

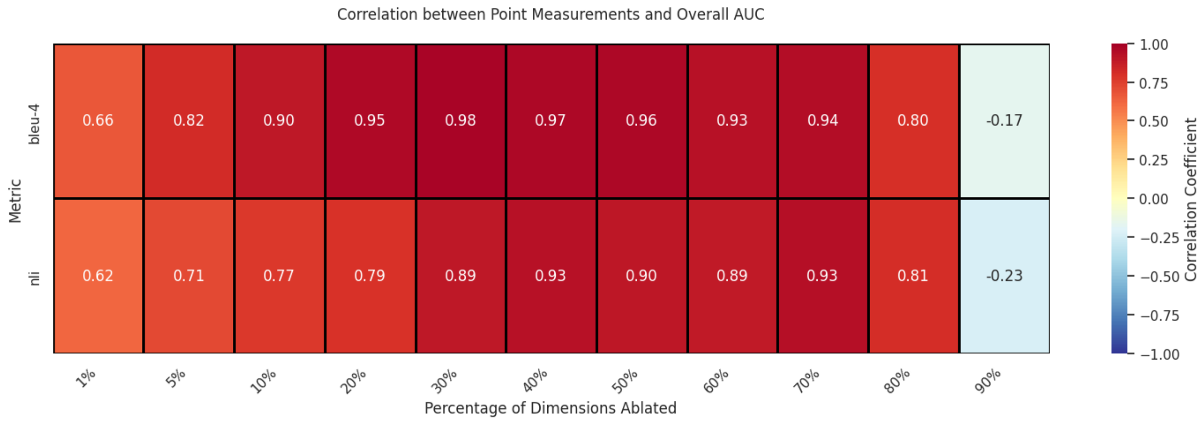


Figure 4: Correlation heatmap between point measures of p and overall AUC. Black outline indicates statistical significance.

the overall AUC across multiple model architectures and scales, encompassing Llama-3.1 (8B), Qwen-2.5 (7B), and Gemma-2 (2B), evaluated on both CNN/Dailymail and WMT'16. For each model-dataset pair, we compute performance metrics using both variance-based, Integrated Gradients (Sundararajan et al., 2017), and random dimension rankings.

Performance at most ablation percentages strongly correlates with the overall AUC (Figure 4). Notably, measurements at $p = 30\%$ demonstrate strong predictive power across all evaluation metrics, but especially BLEU-4. This suggests that a single evaluation at this threshold can serve as a reliable diagnostic for embedding space robustness, eliminating the need for exhaustive ablation studies across multiple values of p (Figure 8).

E Low-Robustness Models

Our analysis suggests that embedding space robustness emerges from a combination of sufficient model scale and high-quality—or perhaps a larger amount of—pretraining data. We examine two models with different potential limitations: GPT-J (6B), trained on a narrower data distribution, and Qwen-2.5 (0.5B), which maintains data quality but significantly reduces model scale. We evaluate these models on the Alpaca (Taori et al., 2023) dataset, comparing against the more robust Llama-2-7B baseline. As shown in Table 8, both potentially low-robustness models demonstrate markedly worse performance than Llama-2-7B, achieving ROUGE-L scores of 12.03 and 15.21 respectively—lower than simply repeating the input question (18.58). Even after carefully cleaning Qwen-2.5's outputs to handle the repeated generation artifacts, performance remains substantially below that of larger models trained on diverse datasets. GPT-J's poor performance despite its larger parameter count suggests that model scale alone does not guarantee embedding space robustness. While Qwen-2.5's limitations can be attributed primarily to its reduced size, GPT-J's underperformance likely stems from its pretraining data and computation amount. This indicates that both factors—model scale and training—contribute to embedding space robustness. These results also align with our ablation analysis using $p = 30\%$ and variance-based sorting, where both models show very low performance preservation, indicating fundamentally less robust embedding spaces compared to their larger, broadly-trained counterparts. Embedding space robustness to ablation may thus serve as an indicator of TinyTE quality.

F Robustness Ablation Curves for BLEU-4 and Entailment

In Table 9, we replicate Table 4, using entailment (nli) as the main metric. Entailment is computed following Honovich et al. (2022). We also display the performance curves for both datasets (using BLEU-4, Figure 5, and entailment, Figure 6).

G Robustness Ablation Using Variance as the Sorting Heuristic

Table 10 shows the ablation results using variance as the sorting heuristic, instead of randomly sorting the indices. Performance curves are shown in Figure 7.

Method	**# Params**	**BLEU-4** (p = 30%)	**Est. AUC**	**R-L**
TinyTE-all (Llama-2-7b)	4096	10.21	7.38 (real: 7.51)	**35.9**
Repeating the question	0	—	—	18.58
TinyTE-all (GPT-J-6b)	4096	4.36	3.44	12.03
TinyTE-all (Qwen-2.5-0.5b)	896	1.84	1.74	15.21
TinyTE-all, clean (Qwen-2.5-0.5b)	896	—	—	26.39

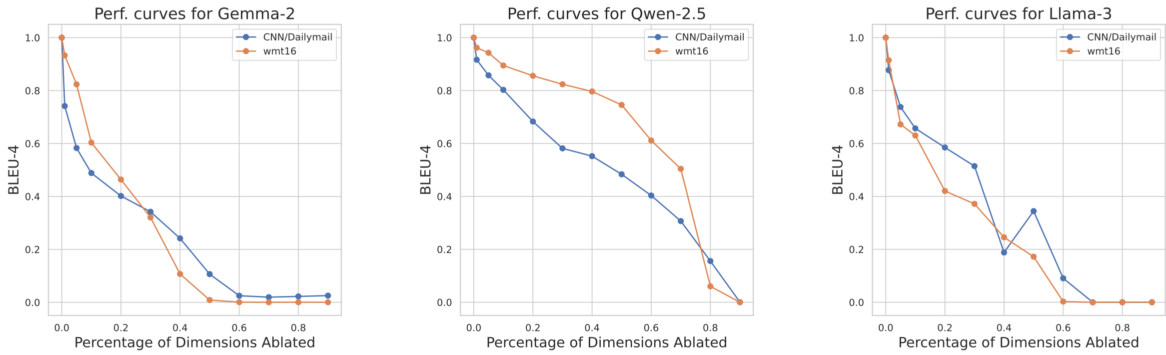
Table 8: Performance of low-robustness models. Note that both models perform worse than simply repeating the input. We analyze the Qwen-2.5 outputs and realize the model fails to terminate the message by outputting the end token; and even after careful cleaning of repeated keywords, it fails to achieve the performance of robust TinyTE models.

Model	**Dataset**	**Performance at p%**						
		1%	5%	10%	20%	50%	80%	AUC
Gemma-2-2B	WMT'16	97.2	90.4	77.6	66.8	06.0	00.0	28.1
	Summ.	69.2	65.2	62.0	60.4	13.6	04.0	28.5
Llama-3.1-8B	WMT'16	93.2	84.0	81.2	75.6	36.4	00.0	36.8
	Summ.	82.8	72.0	66.0	65.2	56.8	00.8	32.9
Qwen-2.5-7B	WMT'16	98.8	98.8	96.0	94.0	88.4	21.2	70.0
	Summ.	90.0	81.2	74.4	62.4	64.4	30.8	52.0

Table 9: Entailment of selected models when ablating $p\%$ of input dimensions on WMT'16 and CNN/Dailymail. Curves are shown in Figure 6.

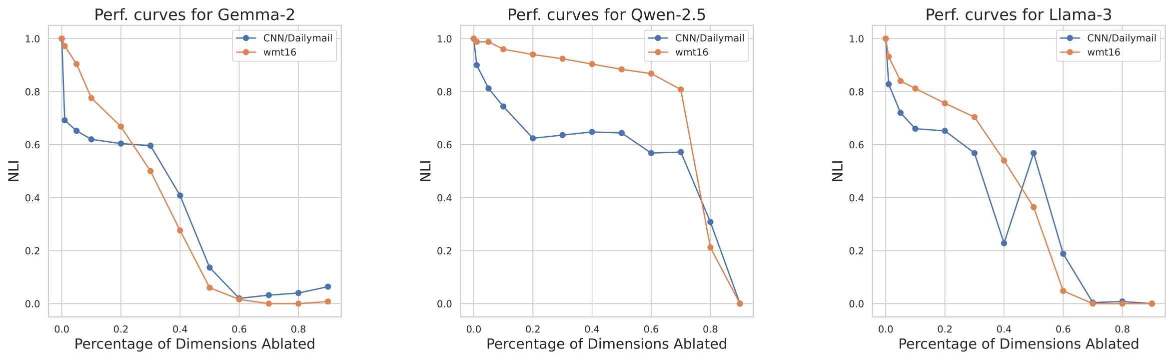
Model	**Metric**	**Dataset**	**Performance at p%**						
			1%	5%	10%	20%	50%	80%	AUC
Gemma-2-2B	BLEU-4	WMT'16	87.3	74.3	70.5	61.7	20.8	00.0	28.9
		Summ.	63.9	53.5	48.4	45.4	25.8	02.4	24.9
	NLI	WMT'16	93.6	86.0	82.4	77.2	34.8	00.0	36.9
		Summ.	66.8	66.8	64.4	63.2	50.4	05.2	38.0
Llama-3.1-8B	BLEU-4	WMT'16	33.3	29.9	24.8	20.7	04.2	00.0	10.0
		Summ.	22.6	23.7	22.1	22.0	19.5	00.2	14.9
	NLI	WMT'16	68.0	62.0	58.4	47.2	20.8	00.0	23.9
		Summ.	34.0	31.2	29.6	29.6	31.2	00.4	22.4
Qwen-2.5-7B	BLEU-4	WMT'16	96.3	93.6	91.5	87.0	76.7	25.8	62.0
		Summ.	93.0	86.9	79.2	70.1	54.6	25.7	49.3
	NLI	WMT'16	99.2	98.8	96.4	97.6	89.6	47.6	74.3
		Summ.	88.4	81.6	74.8	63.6	56.4	53.6	53.8

Table 10: Ablation results using variance as the sorting heuristic.



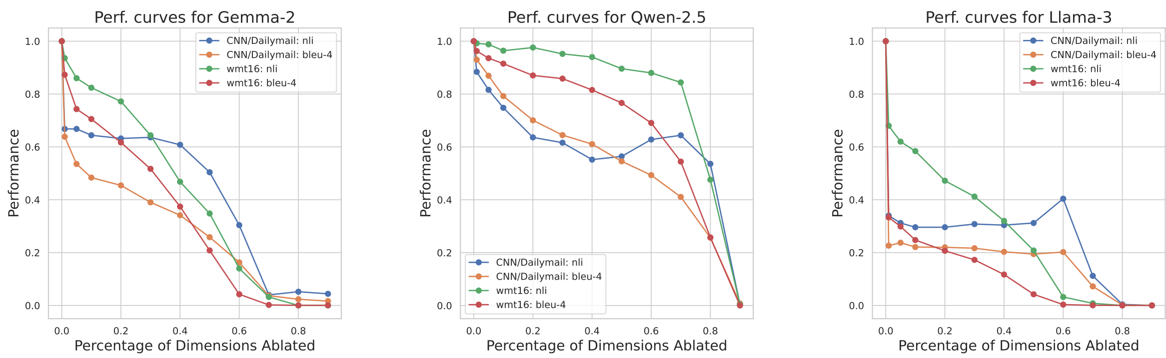
(a) Performance curves: Gemma-2. (b) Performance curves: Qwen-2.5. (c) Performance curves: Llama-3.1.

Figure 5: Gemma-2, Qwen-2.5, and Llama-3.1 performance versus portion (p%) of randomly ablated input dimensions, in WMT'16 and CNN/Dailymail, measured by computing BLEU-4 between the outputs of original and ablated models.



(a) Gemma-2 curves using entailment. (b) Qwen 2.5 curves using entailment. (c) Llama-3.1 curves using entailment.

Figure 6: Performance curves for all three models, sorting randomly and using entailment as the metric.



(a) Gemma-2 sorted by variance. (b) Qwen 2.5 sorted by variance. (c) Llama-3.1 sorted by variance.

Figure 7: Performance curves, using variance as the sorting strategy.

H Full Main Results Table

Table 11 showcases Tables 1 and 2 in a unified format, alongside the standard error. We use the same prompts as (Gao et al., 2024).

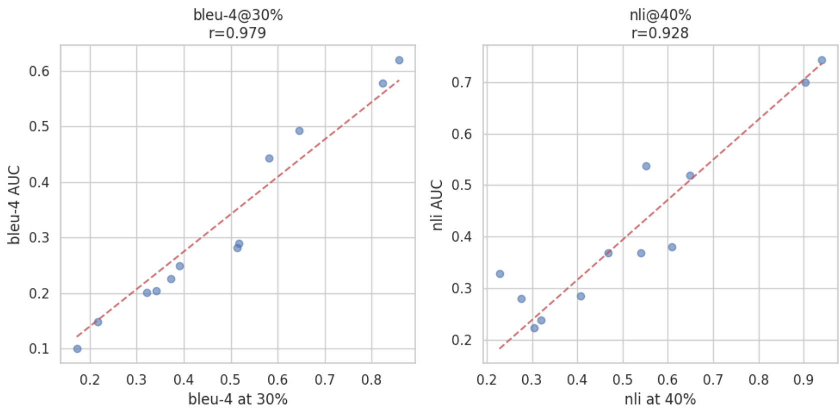


Figure 8: Most highly correlated metrics given $p\%$ of ablated dimensions. Performance at p = 30% is highly indicative of the area under the BLEU-4 curve, while performance at p = 40% is highly indicative of the AUC for nli.

Model	**# Trained Params**	**CNN/Dailymail**	**WMT'16 cs-en**	
		R-L↑	BLEU-4↑	ter↓
Base Models				
Gemma-2	0 (2B)	21.62±0.15	6.04±0.21	734.73±9.62
Qwen-2.5	0 (7B)	23.08±0.16	12.99±0.26	243.38±4.90
Llama-3.1	0 (8B)	22.47±0.16	19.66±0.35	149.84±3.40
PEFT Baselines				
Emb. Tuning (Gemma-2)	590M	35.16±0.26	22.57±0.42	63.37±0.92
LoRA (Gemma-2)	14M	37.09±0.27	25.89±0.42	58.26±0.66
Prompt Tuning (1 tok) (Gemma-2)	2304	30.10±0.25	25.13±0.42	58.50±0.57
Prompt Tuning (2 toks) (Gemma-2)	4608	32.99±0.24	25.68±0.43	57.61±0.48
Prompt Tuning (20 toks) (Gemma-2)	46K	33.60±0.25	25.14±0.42	59.46±0.87
Emb. Tuning (Qwen-2.5)	544M	35.23±0.26	24.92±0.47	61.00±2.17
LoRA (Qwen-2.5)	22M	38.03±0.27	27.36±0.49	56.22±0.56
Prompt Tuning (1 tok) (Qwen-2.5)	3584	32.15±0.25	24.24±0.46	60.27±0.57
Prompt Tuning (2 toks) (Qwen-2.5)	7168	32.17±0.24	25.44±0.46	59.69±0.62
Prompt Tuning (20 toks) (Qwen-2.5)	71M	32.80±0.24	25.29±0.47	59.25±0.57
Emb. Tuning (Llama-3.1)	525M	37.13±0.26	24.78±0.47	58.74±0.56
LoRA (Llama-3.1)	23M	39.03±0.27	27.78±0.49	55.63±0.55
Prompt Tuning (1 tok) (Llama-3.1)	4096	33.81±0.25	29.25±0.50	54.36±0.56
Prompt Tuning (2 toks) (Llama-3.1)	8192	33.90±0.24	29.20±0.50	54.30±0.55
Prompt Tuning (20 toks) (Llama-3.1)	82K	34.96±0.25	28.20±0.49	55.66±0.55
TinyTE				
TinyTE-full (Gemma-2)	2304	32.22±0.24	23.95±0.43	59.87±0.64
TinyTE-half (Gemma-2)	1152	30.71±0.24	22.66±0.41	63.23±0.73
TinyTE-gated (Gemma-2)	2306	33.06±0.25	23.77±0.42	61.04±0.81
TinyTE-hybrid (Gemma-2)	4608	33.50±0.25	25.32±0.43	58.03±0.49
TinyTE-full (Qwen-2.5)	3584	32.92±0.24	23.93±0.41	146.59±8.73
TinyTE-half (Qwen-2.5)	1792	32.55±0.24	24.90±0.42	58.97±0.50
TinyTE-gated (Qwen-2.5)	3586	32.82±0.24	24.45±0.42	83.87±3.45
TinyTE-hybrid (Qwen-2.5)	7168	33.54±0.25	25.81±0.48	58.03±0.56
TinyTE-full (Llama-3.1)	4096	36.23±0.25	27.30±0.44	55.64±0.49
TinyTE-half (Llama-3.1)	2048	35.79±0.25	27.94±0.44	55.32±0.50
TinyTE-gated (Llama-3.1)	4098	36.40±0.25	27.64±0.44	55.09±0.49
TinyTE-hybrid (Llama-3.1)	8192	35.99±0.25	27.74±0.48	56.01±0.70

Table 11: Full results of TinyTE models versus base models and PEFT baselines, with standard errors.