Ahead-of-Time P-Tuning

Anonymous ACL submission

Abstract

This paper proposes a new parameter-efficient method for fine-tuning, AoT P-Tuning. This method adds input-dependent biases before evaluating the Transformer layer, reducing the required evaluation time when compared to P-Tuning. Same as P-Tuning, AoT P-Tuning allows multi-task inference with a single backbone model for evaluating different tasks in a single batch. We experimented with the proposed method on the GLUE and SuperGLUE benchmarking datasets using RoBERTa-Base, RoBERTa-Large, and DeBERTa-XL backbone models. Our observations show that AoT P-tuning performed on par with or better than P-Tuning v2 while being up to $1.3\times$ times faster during inference.

1 Introduction

P-Tuning (Liu et al., 2021b,a; Lester et al., 2021) is a promising way to fine-tune large Language Models (LMs) (Devlin et al., 2019; Lan et al., 2020; Liu et al., 2019; Radford et al., 2019). While it currently underperforms compared to other methods for parameter-efficient fine-tuning (Hu et al., 2022; Houlsby et al., 2019) on a wide range of tasks (Ding et al., 2022), it has a practical, valuable property that allows it to evaluate different trained prompts parallel in a multi-task manner (i.e., a single backbone LM could be used for different tasks during inference, which can simplify model serving in real-world applications) (Lester et al., 2021). This property is why researchers aim to further develop P-Tuning methods.

Although it is possible to perform multi-task evaluation with P-Tuning, it introduces significant computational overhead due to the concatenation of prefixes to sequences and the evaluation of the attention mechanism (Vaswani et al., 2017) on longer sequences.

We propose a simple mechanism for parameter-efficient fine-tuning of Language Models, namely

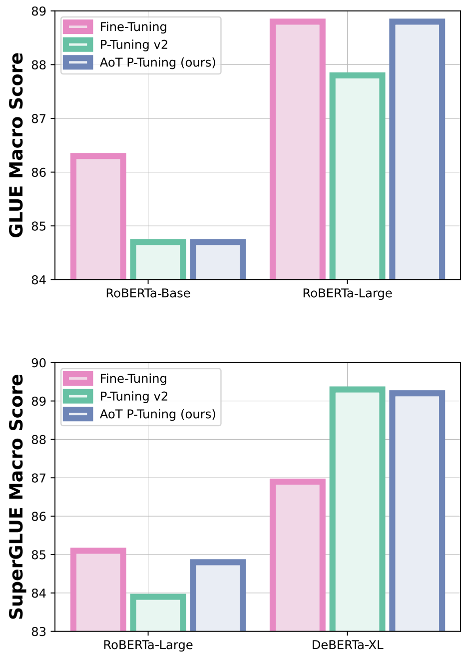


Figure 1: GLUE and SuperGLUE Macro scores (higher is better) for different backbone model scales with plain Fine-Tuning, P-Tuning v2, and proposed AoT P-Tuning (with FC reparametrization). Based on these experiments, AoT P-Tuning performed on par with or better than P-Tuning v2. See Section 5.2 for more details.

Ahead-of-Time (AoT) P-Tuning, for which we add input-dependent bias before each Transformer layer. Same as P-Tuning, it is possible to use AoT P-Tuning in multi-task inference setups when a single backbone LM is used for several downstream tasks.

The contributions of this paper can be summarized as follows:

1. We described the intuition behind AoT P-Tuning, which illustrates the connection of the proposed method with P-Tuning.

2. We proposed two reparameterizations of AoT P-Tuning weights: first based on a factorized

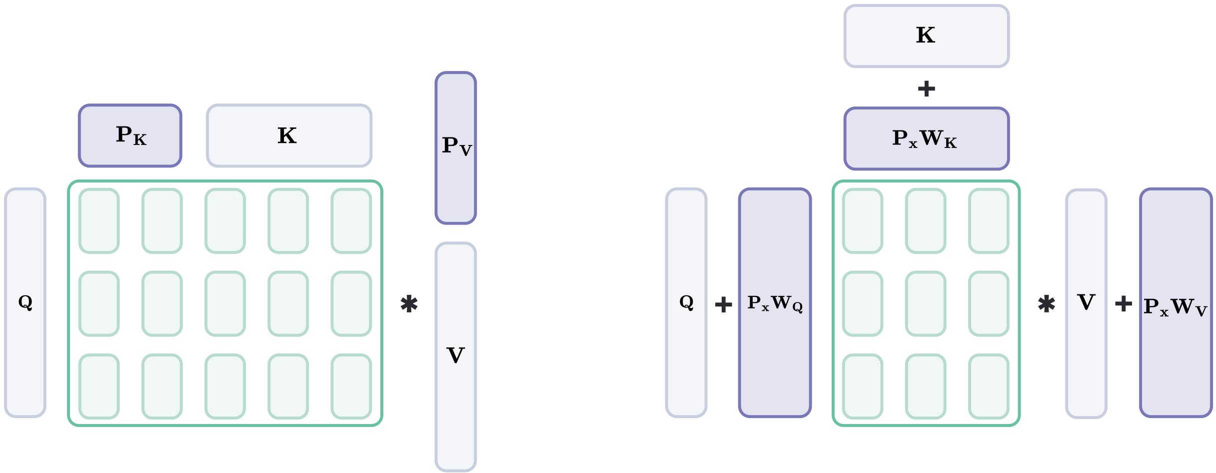


Figure 2: Schematic comparison of P-Tuning v2 (left), and AoT P-Tuning (right). While plain P-Tuning concatenates soft prompts to sequences and thus causes computational overhead, AoT P-Tuning directly adds input-dependent biases to $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ matrices. See Section 4 for more details on AoT P-Tuning architecture. Since the sequence length is not increased, AoT P-Tuning takes significantly less time to evaluate, only requiring the overhead of adding biases to the input sequence (See Section 5.3 for experiments with inference speed).

matrix trained from scratch, and second based on a LM's embeddings matrix passed through a trainable Fully Connected network.

3. We experimented with the proposed method on GLUE and SuperGLUE Benchmarking Datasets (Wang et al., 2018, 2019) with the RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2020) models and observed that AoT P-Tuning performed on par with or better than P-Tuning v2 (Liu et al., 2021a) while being up to $1.3\times$ times faster during evaluation.

2 Recent Works

Currently, a wide range of different methods could be referenced with P-Tuning. Liu et al. (2021b) proposed to add soft prompts to the embeddings of GPT-2's input sequence (Radford et al., 2019) to train it on classification tasks. Lester et al. (2021) proposed a scheme similar to the one used in Liu et al. (2021b), but trained a T5 model (Raffel et al., 2020) with P-Tuning to show how the performance of the method changes with the increased scale of the backbone model.

Recently, Qin and Eisner (2021); Li and Liang (2021); Liu et al. (2021a) proposed to add prefixes not only to input embeddings but also at each layer of the Transformer model. In addition, Liu et al. (2021a) suggested training a linear classification head on top of the backbone model instead of utilizing a LM head to obtain classification results.

Due to this range of similar methods, we will follow the naming used by Liu et al. (2021a) and refer to Prompt-Tuning (adding soft prompts to the input embeddings) as P-Tuning v1 and to Prefix-Tuning (adding soft prefixes at each layer of Transformer backbone) as P-Tuning v2.

3 Background

3.1 P-Tuning v1

For readers conviniece, we provided background of Transformer evaluation in Section A. Having a pre-trained Transformer LM with parameters Θ, instead of fine-tuning all parameters of this model on a downstream task, it is possible to define soft prompts $\boldsymbol{P} \in \mathbb{R}^{p \times d}$(Liu et al., 2021b), where p is the length of prompt. $\boldsymbol{P}$ is then concatenated to input sequence embeddings as:

$$\boldsymbol{H}'^{0} = \text{concat}(\boldsymbol{P}, \boldsymbol{H}^{0}) \in \mathbb{R}^{(p+n) \times d}. \quad (1)$$

Then, only $\boldsymbol{P}$ and Classification Head are fine-tuned on a downstream task, while Θ remains frozen[1]. Such parametrization of fine-tuning makes it possible to perform multi-task inference.

3.2 P-Tuning v2

Instead of concatenation of a single prompt $\boldsymbol{P}$ to the $\boldsymbol{H}^{0}$, Liu et al. (2021a) proposed to con-

[1]Original implementation of P-Tuning v1 (Liu et al., 2021b) implied utilizing the LM Head of a pre-trained model instead of training a Classification Head. However, Liu et al. (2021a) later showed that using a separate Classification Head performs marginally better.

catenate soft prefixes at each layer of the Transformer model. To apply P-Tuning v2, soft prefixes $\boldsymbol{P}_K, \boldsymbol{P}_V \in \mathbb{R}^{p \times d}$ are defined for each layer and concatenated to the $\boldsymbol{K}$ and $\boldsymbol{V}$ matrices before evaluating the attention $\boldsymbol{K}' = \text{concat}(\boldsymbol{P}_K, \boldsymbol{K})$, $\boldsymbol{V}' = \text{concat}(\boldsymbol{P}_V, \boldsymbol{V})$. Then, Attention is evaluated as follows:

$$\boldsymbol{A}' = \text{attention}(\boldsymbol{Q}, \boldsymbol{K}', \boldsymbol{V}'), \tag{2}$$

where i-th component of $\boldsymbol{A}'$ could be then written as:

$$\begin{aligned} \boldsymbol{A}'_i = \sum_{j=1}^{p} \boldsymbol{a}_j(\boldsymbol{Q}_i, \boldsymbol{K}')\boldsymbol{P}_{\boldsymbol{V}_j} + \\ + \sum_{k=1}^{n} \boldsymbol{a}_{k+p}(\boldsymbol{Q}_i, \boldsymbol{K}')\boldsymbol{V}_k. \end{aligned} \tag{3}$$

Note that $\boldsymbol{a} \in \mathbb{R}^{p+n}$ are attention weights for the i-th token (we omit the i-th index for simplicity) and thus $\sum_{j=1}^{p+n} \boldsymbol{a}_j = 1$.

As for P-Tuning v1, only parameters of soft prefixes $\boldsymbol{P}_K, \boldsymbol{P}_V$ and Classification Head are optimized on a downstream task while freezing the parameters of a backbone model.

3.3 On the Overhead of P-Tuning

While the Transformer model has $\mathbb{O}(n^2)$ time complexity and GPU memory consumption for sequence length n. For P-Tuning v1, this complexity transforms into $\mathbb{O}((n+p)^2)$ since the length of input sequence is increased by the length of the prompt p, while for P-Tuning v2 the complexity is equal to $\mathbb{O}(n(n+p))$.

Liu et al. (2021a) showed that for some tasks, the prompt length p could reach values of 100, increasing time and memory footprint during evaluation.

4 Ahead-of-Time P-Tuning

4.1 Proposed Mechanism

With AoT P-Tuning, we propose to augment each Transformer layer with a simple procedure. We define trainable matrices $\boldsymbol{P} \in \mathbb{R}^{|V| \times d}$ for each layer. Then, before the evaluation of the i-th layer, we modify the hidden states as follows:

$$\boldsymbol{H}'^i = \boldsymbol{H}^i + \{\boldsymbol{P}_{x_1}, \dots, \boldsymbol{P}_{x_n}\} \in \mathbb{R}^{n \times d}, \tag{4}$$

where $\boldsymbol{P}_{x_j} \in \mathbb{R}^d$ is a lockup of x_j-th prompt embedding from $\boldsymbol{P}$. Such a scheme allows us to save a significant amount of time during evaluation since AoT P-Tuning does not imply an increase in sequence length. While $\boldsymbol{P}$ in naive implementation will require lot of memory to store parameters, in the following Section 4.3, we describe reparametrizations which make training more tractable.

Note that AoT P-Tuning, same as plain P-Tuning, could be evaluated in parallel with several tasks in a batch due to the fact that performing look-up from $\boldsymbol{P}$ can be easily parallelized.

As for P-Tuning v1 and P-Tuning v2, we only optimize parameters of $\boldsymbol{P}$ and Classification Head during fine-tuning.

4.2 Intuition Behind AoT P-Tuning and Connection to the P-Tuning

One may note that the proposed method is more similar to Adapters Tuning (Houlsby et al., 2019) than P-Tuning. Although, Adapters do not imply performing multi-task inference, thus we refer to the proposed method as a variant of P-Tuning, rather than a special case of Adapters. Furthermore, considering Ding et al. (2022); He et al. (2022), most methods for parameter-efficient fine-tuning could be seen with a unified view, and thus Adapters could be seen as a variant of P-Tuning and vice versa.

Having $\boldsymbol{H}'$, after passing through $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ we obtain $\boldsymbol{Q}'$, $\boldsymbol{K}'$, and $\boldsymbol{V}'$. Note that $\boldsymbol{V}' = \boldsymbol{H}\boldsymbol{W}_V + \{\boldsymbol{P}_{x_1}, \dots, \boldsymbol{P}_{x_n}\}\boldsymbol{W}_V \stackrel{\text{def}}{=} \boldsymbol{V} + \boldsymbol{P}_x\boldsymbol{W}_V$.

The result of evaluating Attention with AoT P-Tuning could be seen as:

$$\begin{aligned} \boldsymbol{A}'_i = \sum_{j=1}^{n} \boldsymbol{a}_j(\boldsymbol{Q}'_i, \boldsymbol{K}')\boldsymbol{P}_{x_j}\boldsymbol{W}_V + \\ + \sum_{j=1}^{n} \boldsymbol{a}_j(\boldsymbol{Q}'_i, \boldsymbol{K}')\boldsymbol{V}_j. \end{aligned} \tag{5}$$

From such a perspective, there is a clear connection between AoT P-Tuning (Equation 5) and P-Tuning v2 (Equation 3) with the following changes:

1. For AoT P-Tuning, attention weights $\boldsymbol{a}_j$, $j \in \overline{1, l}$ are used for both terms in Equation 5.

2. For AoT P-Tuning, attention is evaluated on modified $\boldsymbol{Q}'$. In addition, there is a difference in the form of dependency of $\boldsymbol{K}'$ and $\boldsymbol{V}'$ on prefix weight. For AoT P-Tuning, we add

prefixes to $\boldsymbol{K}$ and $\boldsymbol{V}$, while for P-Tuning v2, prefixes are concatenated to these matrices.

3. For AoT P-Tuning, the first term of Equation 5 implies evaluation of Attention with a prompt which is dependent on the input text, while for P-Tuning v2, the prompt $\boldsymbol{P}_V$ is constant.

Considering Equation 5, AoT can be seen as a form of the P-Tuning method, for which we embed prefixes before evaluating the attention layer[2].

4.3 On the Parameter Efficiency of AoT P-Tuning

It is notable that, in most cases, it is not feasible to optimize the weight $\boldsymbol{P} \in \mathbb{R}^{|V| \times d}$ for each layer. If we consider training RoBERTa-Large with such a scheme (which has $|V| = 50265$, $d = 1024$ and $l = 24$), then storing all biases $\boldsymbol{P}$ will exceed 1.2B parameters, while the model itself has roughly 350M parameters.

To overcome this limitation, we propose two reparametrizations of $\boldsymbol{P}$ so that it can use fewer parameters during training.

The first is based on the Kronecker product (namely, **Kronecker AoT P-Tuning**). More specifically, we reparametrize $\boldsymbol{P}$ as

$$\boldsymbol{P} = (\boldsymbol{W}_L \otimes \boldsymbol{W}_M)\boldsymbol{W}_R, \tag{6}$$

where $\boldsymbol{W}_L \in \mathbb{R}^{a \times r}$, $\boldsymbol{W}_M \in \mathbb{R}^{b \times r}$, $\boldsymbol{W}_R \in \mathbb{R}^{r^2 \times d}$, a and b are selected in such a way so $a * b = |V|$, r is the factorization rank which is a hyperparameter to tune, and $\otimes$ denotes the Kronecker product.

With this reparametrization, training AoT P-Tuning becomes tractable. E.g., for RoBERTa-Large, with $a = 256$, $b = 200$, and $r = 20$, $\boldsymbol{P}$ will contain roughly 10M parameters, which is less than 3% of the total number of parameters in the model[3].

The second approach to work with $\boldsymbol{P}$, which we used in our experiments, is based on passing the embeddings matrix $\boldsymbol{E}$ through a learnable Fully Connected network (namely, **FC AoT P-Tuning**). Thus, we reparametrize $\boldsymbol{P}$ as

$$\boldsymbol{P} = f(\boldsymbol{E}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{7}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d \times r}$, $\boldsymbol{b}_1 \in \mathbb{R}^r$, $\boldsymbol{W}_2 \in \mathbb{R}^{r \times d}$, $\boldsymbol{b}_2 \in \mathbb{R}^d$, f is a non-linearity, and r is the mapping rank, which is also a hyperparameter to tune, same as for Kronecker AoT P-Tuning.

With FC AoT P-Tuning, we utilize knowledge stored in the pre-trained embeddings matrix $\boldsymbol{E}$, which should hypothetically perform better than training $\boldsymbol{P}$ from scratch as Kronecker AoT P-Tuning.

Note that for both Kronecker and FC AoT P-Tuning, we can evaluate only specific rows $\{\boldsymbol{P}_{x_i}, \dots, \boldsymbol{P}_{x_n}\}$ for input sequence $\{x_1, \dots, x_n\}$, making training more efficient.

For both reparametrizations, $\boldsymbol{P}$ could be fused once training is complete, and thus the rank of factorization r does not affect inference speed. During the evaluation, there is no need to store the full $\boldsymbol{P}$ in GPU memory. Instead, it could be stored in RAM, and only rows of these matrices should be placed in GPU memory to be added to the hidden states before each layer.

From a certain perspective, choosing between AoT P-Tuning and P-Tuning is a trade-off between evaluation speed and RAM consumption during inference. If RAM is limited, then usual P-Tuning could be used at the cost of slower inference. In other cases, AoT P-Tuning is viable if there is enough RAM and inference speed is crucial. Although, in most cases, $\boldsymbol{P}$ matrices for different tasks could be easily stored in the RAM. For RoBERTa-Large, a single task parameter will require roughly 2.4Gb if stored in half-precision.

However, as we observed later in our experiments, performing fusing is not crucial for FC AoT P-Tuning, and the re-evaluation of $\{\boldsymbol{P}_{x_i}, \dots, \boldsymbol{P}_{x_n}\}$ for each sequence ran at 98.5% the speed of fused $\boldsymbol{P}$ (See Section 5.3 for more details).

5 Experiments

5.1 Experimental Details

We compared AoT P-Tuning (Kronecker and FC reparametrizations of $\boldsymbol{P}$) with other fine-tuning methods capable of performing multi-task inference: P-Tuning v1, P-Tuning v2 on GLUE and SuperGLUE (Wang et al., 2018, 2019) Benchmarking

[2]It is possible to think of AoT P-Tuning as a method which adds bias **after** the evaluation of the Transformer layer. In this case, it could be seen as a method that directly models the result of the evaluation of P-Tuning v2 with a slightly different computation order. However, we believe that this way is more difficult to consider.

[3]One may note that $256 * 200 = 51200 \neq 50265$. However, 50265 is difficult to factorize efficiently since $50265 = 1117 * 3^2 * 5$. Because of this, we chose to mostly factorize $\boldsymbol{P}$ in such a way as to make it slightly larger than the original vocabulary size. Doing so allows us to select more appropriate a and b from the perspective of parameter and computational efficiency.

Model	STS-B	SST-2	RTE	QQP	
RoBERTa-Base					
Fine-Tuning	90.6 ± 0.3	95.0 ± 0.2	81.2 ± 0.7	89.6 ± 0.2	
P-Tuning v1	86.9 ± 0.9	94.0 ± 0.3	60.3 ± 2.4	82.2 ± 1.5	
P-Tuning v2	89.2 ± 0.3	**94.6 ± 0.2**	**80.5 ± 3.4**	86.4 ± 3.3	
Kron. AoT P-Tuning (ours)	89.7 ± 0.2	94.0 ± 0.2	77.6 ± 1.4	88.2 ± 0.1	
FC AoT P-Tuning (ours)	**90.0 ± 0.2**	94.4 ± 0.3	78.0 ± 1.3	**87.9 ± 0.2**	
	QNLI	MRPC	MNLI	CoLA	Macro
Fine-Tuning	92.4 ± 0.1	90.8 ± 0.5	87.0 ± 0.3	63.8 ± 1.4	86.3
P-Tuning v1	88.3 ± 0.5	82.0 ± 1.7	80.8 ± 0.6	45.8 ± 27.1	77.5
P-Tuning v2	**91.9 ± 1.6**	89.1 ± 1.1	85.3 ± 0.2	**60.7 ± 2.6**	**84.7**
Kron. AoT P-Tuning (ours)	90.7 ± 0.4	89.5 ± 1.1	84.6 ± 0.1	59.3 ± 1.2	84.2
FC AoT P-Tuning (ours)	91.3 ± 0.4	**90.3 ± 0.3**	**85.4 ± 0.1**	60.3 ± 2.2	**84.7**
RoBERTa-Large					
Model	STS-B	SST-2	RTE	QQP	
Fine-Tuning	91.9 ± 0.2	96.1 ± 0.4	88.1 ± 1.5	90.3 ± 0.2	
P-Tuning v1	75.5 ± 6.3	94.4 ± 0.4	62.8 ± 2.3	76.9 ± 2.5	
P-Tuning v2	91.0 ± 0.4	96.1 ± 0.3	87.4 ± 1.5	86.6 ± 0.6	
Kron. AoT P-Tuning (ours)	91.1 ± 0.8	96.2 ± 0.2	84.8 ± 1.3	**89.4 ± 0.1**	
FC AoT P-Tuning (ours)	**91.7 ± 0.4**	**96.7 ± 0.1**	**88.4 ± 0.9**	88.7 ± 0.2	
	QNLI	MRPC	MNLI	CoLA	Macro
Fine-Tuning	94.3 ± 0.2	91.6 ± 0.6	89.9 ± 0.2	68.1 ± 1.9	88.8
P-Tuning v1	79.1 ± 2.4	79.0 ± 1.1	75.9 ± 18.3	24.7 ± 17.6	71.0
P-Tuning v2	94.0 ± 1.1	91.2 ± 0.9	89.4 ± 0.7	66.9 ± 1.5	87.8
Kron. AoT P-Tuning (ours)	**94.2 ± 0.1**	89.7 ± 0.9	89.3 ± 0.1	65.5 ± 1.9	87.5
FC AoT P-Tuning (ours)	94.1 ± 0.2	**91.6 ± 0.8**	**89.6 ± 0.1**	**69.2 ± 0.9**	**88.8**

Table 1: Results on the GLUE Dev set. Each result is median and std across several seeds, and the Macro column is a mean score across all tasks. Fine-tuning is omitted from comparison with other methods and was not bolded for visibility. See Section 5.2 for details.

Datasets[4]. We also evaluated plain fine-tuning for reference even though it is impossible to perform multi-task inference with it. For each fine-tuning approach, we experimented with the RoBERTa-Base, RoBERTa-Large, and DeBERTa-XL backbone models.

For each task, we performed a grid hyperparameter search (see Appendix Table 4 for hyperparameter ranges). For RoBERTa models, we evaluated each hyperparameter set with 5 different seed values and reported median and std score values for each task. For DeBERTa-XL, we used to assess each hyperparameter assignment with a single seed due to longer training time. See Appendix Table 3 for a list of metrics used for each task.

We used the Adam (Kingma and Ba, 2015) optimizer with a constant learning rate for each task. We stopped training once the validation metric stopped increasing (see the "patience" parameter in Appendix Table 5).

For Kronecker AoT P-Tuning with RoBERTa models, we parametrized the matrix $\boldsymbol{P} = (\boldsymbol{W}_L \otimes \boldsymbol{W}_M)\boldsymbol{W}_R$ with $a = 256$, and $b = 200$, while for DeBERTa, we used $a = b = 360$. $\boldsymbol{W}_L$ and $\boldsymbol{W}_M$ were initialized randomly, while $\boldsymbol{W}_R$ was initialized as a zero matrix. For FC AoT P-Tuning, we

[4]Based on this experimental design choice, we exclude experiments with Adapters (Houlsby et al., 2019; He et al., 2022), as well as with LoRA (Hu et al., 2022). While a wide range of efficient fine-tuning methods could be similar to the proposed method (Ding et al., 2022; He et al., 2022), they do not allow to perform multi-task inference, which is the motivation for using AoT P-Tuning.

RoBERTa-Large				
Model	RTE	COPA	WSC	WiC
Fine-Tuning	88.1 ± 1.5	87.0 ± 10.2	80.8 ± 6.3	73.8 ± 1.6
P-Tuning v1	62.8 ± 2.3	75.0 ± 4.3	66.3 ± 1.3	64.1 ± 0.9
P-Tuning v2	87.4 ± 1.5	**87.0 ± 6.3**	75.0 ± 7.7	70.8 ± 1.5
Kron. AoT P-Tuning (ours)	84.8 ± 1.3	72.0 ± 9.1	67.3 ± 3.0	71.0 ± 1.0
FC AoT P-Tuning (ours)	**88.4 ± 0.9**	85.0 ± 10.1	**79.8 ± 4.1**	**72.1 ± 1.5**
	MultiRC	CB	BoolQ	Macro
Fine-Tuning	83.3 ± 1.1	97.3 ± 2.8	85.6 ± 0.3	85.1
P-Tuning v1	54.3 ± 2.9	81.4 ± 3.0	64.3 ± 1.2	66.9
P-Tuning v2	82.4 ± 0.6	**100.0 ± 0.8**	85.0 ± 0.6	83.9
Kron. AoT P-Tuning (ours)	**82.8 ± 0.8**	97.3 ± 2.3	84.8 ± 0.5	80.0
FC AoT P-Tuning (ours)	82.7 ± 19.3	**100.0 ± 0.0**	**85.5 ± 10.3**	**84.8**
DeBERTa-XL				
Model	RTE	COPA	WSC	WiC
Fine-Tuning	89.9	96.0	76.9	75.9
P-Tuning v1	78.3	90.0	67.3	66.8
P-Tuning v2	90.6	97.0	89.4	**76.5**
Kron. AoT P-Tuning (ours)	88.8	96.0	87.5	71.8
FC AoT P-Tuning (ours)	**91.0**	**98.0**	**94.2**	74.1
	MultiRC	CB	BoolQ	Macro
Fine-Tuning	84.3	98.4	86.7	86.9
P-Tuning v1	82.1	93.8	79.4	79.7
P-Tuning v2	**87.1**	**97.3**	87.0	**89.3**
Kron. AoT P-Tuning (ours)	86.3	83.1	87.3	85.8
FC AoT P-Tuning (ours)	86.5	92.3	**88.1**	89.2

Table 2: Results on the SuperGLUE Dev set. For RoBERTa-Large, each result is median and std across several seeds, and the Macro column is a mean score across all tasks. For DeBERTa-XL, we evaluated each hyperparameter assignment with a single seed and reported its metric score. Fine-tuning is omitted from comparison with other methods and was not bolded for visibility. See Section 5.2 for details.

initialized $\boldsymbol{W}_1$ randomly, while $\boldsymbol{W}_2$, $\boldsymbol{b}_1$, and $\boldsymbol{b}_2$ were initialized with zeros. For Kronecker AoT P-Tuning, we applied dropout (Srivastava et al., 2014) to the $\boldsymbol{P}_x$ with a fixed probability equal to 0.1. In contrast, for FC AoT P-Tuning, we applied dropout to $\boldsymbol{E}$ before multiplying it with $\boldsymbol{W}_1$.

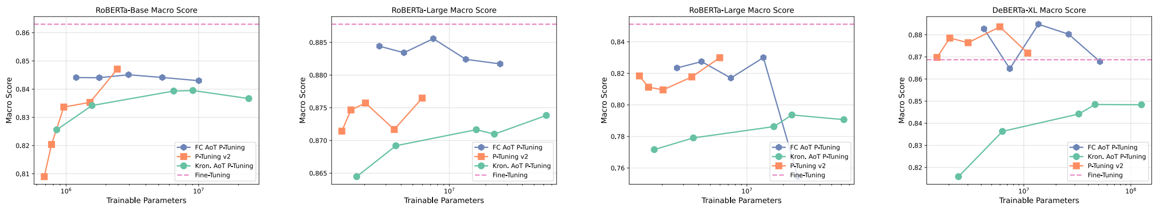
Each experiment was run on a single NVIDIA A100 GPU with a total computation time of roughly 750 days.

5.2 Results

See Tables 1, 2 for the results of trained models. We observed that FC AoT P-Tuning performed better than Kronecker AoT P-Tuning, and hypothesize that this result is mostly caused by the fact that FC reparametrization utilized a pre-trained embedding matrix rather than learning biases from scratch.

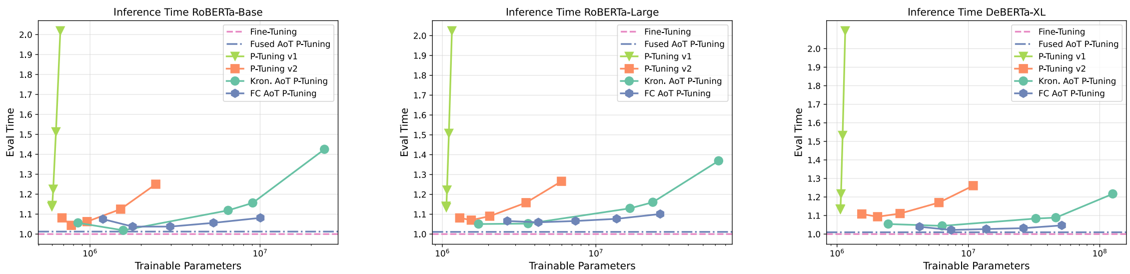
For RoBERTa-Base, FC AoT P-Tuning performed on par with P-Tuning v2 and produced the same Macro score. For RoBERTa-Large, FC AoT P-Tuning outperformed P-Tuning v2 on GLUE tasks and showed a Macro score equal to plain Fine-Tuning. AoT P-Tuning with DeBERTa-XL performed on par with P-Tuning v2 (89.2 vs 89.3 macro scores respectively).

We also observed that both AoT P-Tuning reparametrizations mainly showed a lower variance of metrics across different seeds. Note that P-Tuning v1 showed unstable performance and improved results with RoBERTa-Base (although still



(a) (b) (c) (d)

Figure 3: (a-b) GLUE macro scores for AoT P-Tuning, P-Tuning v1, and P-Tuning v2 with RoBERTa-Base and RoBERTa-Large models. (c-d) SuperGLUE macro score for RoBERTa-Base and DeBERTa-XL models. P-Tuning v2 performing on par with or worse than AoT P-Tuning across different prefix sizes. See Section 5.2 for details.



(a) (b) (c)

Figure 4: Comparison of AoT P-Tuning evaluation time with P-Tuning v1 and P-Tuning v2 for (a) RoBERTa-Base, (b) RoBERTa-Large, and (c) DeBERTa-XL models. We evaluated AoT P-Tuning in two scenarios: with fused weight $\boldsymbol{P}$ and with the re-evaluation of $\boldsymbol{P}$ during inference to reduce memory footprint (See Section 4.3 for more details). Fused AoT P-Tuning adds negligible computational overhead compared to plain Fine-Tuning and is up to $1.3\times$ times faster than P-Tuning v2. See Section 5.3 for more details.

underperforming by a large margin when compared to other methods).

See Figure 3 for macro scores of P-Tuning v2 and AoT P-Tuning with different prefix lengths p and prefix ranks r[5]. We observed that P-Tuning v2 performed worse for RoBERTa-Base with shorter prompt lengths and was comparable to or better than AoT P-Tuning when $p > 50$. For GLUE tasks with RoBERTa-Large, FC AoT P-Tuning performed better for all prefixes p, while dropping performance for large rank r. For DeBERTa-XL, both P-Tuning v2 and FC AoT P-Tuning performed on par. We also provide per-task results with different prefix scales (see Appendix Figures 5, 7). It is notable that in most cases, P-Tuning v2 suffers from a small prefix size p for Base and Large models, and achieves results comparable with AoT P-Tuning with a larger p (which corresponds with the results in Figure 3). At the same time, FC AoT P-Tuning mostly showed stable performance across different ranks r, only performing unstably on a MultiRC task with a large rank r. We also analyzed trained $\boldsymbol{P}$ matrices for FC AoT P-Tuning with the DeBERTa-XL model. See Appendix Section B for more details.

With per-task Expected Validation Performance (EVP) (Dodge et al., 2019), we observed that AoT P-Tuning highly depends on the number of hyperparameter assignments (see Appendix Figures 6, 8). Although, in most cases, using less than 100 hyperparameter assignments for AoT P-Tuning is enough for it to outperform P-Tuning v2, which is not crucial in most cases.

5.3 Inference Speed Overhead

In Figure 4, we also investigated the computational overhead of AoT P-Tuning compared to other baselines.

To estimate inference speed overhead, we evaluated each model 100 times on a sequence with length $n = 128$ and batch size 256.

We evaluated AoT P-Tuning in two setups. The first setup fuses $\boldsymbol{P}$ for inference, thus saving computational time at the cost of a higher memory foot-

[5]Note that the best macro result across different scales of prefixes in these Figures differs from the macro result from Tables 1 and 2, since the macro score from Tables 1 and 2 aggregates scores with different prefix scales.

print. Since $\boldsymbol{P}$ is fused, it no longer depends on factorization rank r for both FC and Kronecker AoT P-Tuning.

For the second setup, we did not fuse $\boldsymbol{P}$, but rather evaluated $\{\boldsymbol{P}_{x_i}, \ldots, \boldsymbol{P}_{x_n}\}$ for each sequence. This approach emulates a setup with limited memory during inference, where fusing $\boldsymbol{P}$ is not feasible.

The growth of p P-Tuning v1 quickly reaches $2\times$ speed overhead since its complexity quadratically depends on p. While P-Tuning v2 involves linear dependency on p (see Section 3.3 for details), it also reaches up to $1.3\times$ inference speed overhead for large prefix lengths p.

Fused AoT P-Tuning adds negligible computational overhead (less than 1%) compared to plain Fine-Tuning. Compared to P-Tuning v2, Fused AoT P-Tuning performed up to $1.3\times$ times faster depending on the prefix sizes used for P-Tuning v2.

When $\boldsymbol{P}$ is not fused, FC AoT P-Tuning performs $1.13 - 1.25\times$ times faster than P-Tuning v2 with large prefixes p. This indicates that **performing weight fusing is not crucial in most cases** for this reparametrization, and that a significant increase in inference speed can be achieved without it. Although not performing fusing of $\boldsymbol{P}$ could reduce memory footprint during inference, it is not possible to perform multi-task inference in such a setup, which is available for both P-Tuning v1/v2 and Fused AoT P-Tuning.

Kronecker's reparametrization performed worse. For small factorization rates (e.g., $r \in [5, 10]$), it showed results comparable to FC AoT P-Tuning. However, it performed up to $1.12\times$ times slower than P-Tuning v2 for larger r values. This makes it is important to fuse $\boldsymbol{P}$ with such a reparametrization when using a large rank r.

It is important to note that the contribution of re-evaluation of $\boldsymbol{P}$ for both Kronecker and FC reparametrizations of AoT P-Tuning becomes lower with model growth. E.g., in the worst-case scenario (with $r = 512$), RoBERTa-Base re-evaluation of $\boldsymbol{P}$ with FC AoT P-Tuning adds $1.09\times$ inference time overhead compared to models trained with plain Fine-Tuning, while DeBERTa-XL showed an overhead of $1.05\times$. The same holds true for small ranks ($r = 64$), where we observed $1.02\times$ inference time overhead for DeBERTa-XL compared to the plain model.

6 Conclusion and Future Work

In this paper, we proposed AoT P-Tuning, which is a new method for parameter-efficient fine-tuning of pre-trained models, and two reparametrizations of learnable weights for this method.

We observed that AoT P-Tuning performed on par or better than P-Tuning v2 based on the macro scores of GLUE and SuperGLUE Benchmarking Datasets.

Moreover, AoT P-Tuning performed up to $1.3\times$ times faster than P-Tuning v2, adding a negligible inference time footprint compared to plain Fine-Tuning. When FC AoT P-Tuning is used, we observed that one could not fuse weights $\boldsymbol{P}$ in order to not introduce memory footprint since it performs up to $1.25\times$ times faster than P-Tuning v2.

We experimented with two reparametrizations based on the Kronecker product and FC network. It is possible to explore other possible reparametrizations for weight $\boldsymbol{P}$, which could further increase the performance of the proposed method. In addition, while we proposed a simple method, there are many possible architectural changes which could also boost the performance of AoT P-Tuning and reduce the number of necessary hyperparameter assignments.

References

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *CoRR*, abs/2006.03654.

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*, volume 97 of *Proceedings of Machine Learning Research*, pages 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 *ICLR 2022*.

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*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. Cite arxiv:1907.11692.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. Cite arxiv:1804.07461Comment: https://gluebenchmark.com/.

A Evaluation of Transformer

Having an input sequence $\boldsymbol{x} = \{x_1, \dots, x_n\}$, where x_i is token index, the embeddings of input texts are evaluated as $\boldsymbol{H}^0 = \{\boldsymbol{E}_{x_1}, \dots, \boldsymbol{E}_{x_n}\}$, where $\boldsymbol{E} \in \mathbb{R}^{|V| \times d}$ is the embeddings matrix, $|V|$ is the vocabulary size, d is the size of the hidden state of the model, and $\boldsymbol{E}_{x_i}$ is an embedding of the token x_i. Hidden states $\boldsymbol{H}^i$ are then passed to the $(i+1)$-th layer of the Transformer to evaluate $\boldsymbol{H}^{i+1}$ with a total l number of layers. To do so, $\boldsymbol{H}^i$ are first mapped through three matrices $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V \in \mathbb{R}^{d \times d}$ to get $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$, which are then used to evaluate the attention layer's results as:

$$\boldsymbol{A} = \text{attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \in \mathbb{R}^{n \times d}. \tag{8}$$

After $\boldsymbol{A}$ is evaluated, it is passed through the remaining layers[6], including residual connections and FC layers to get $\boldsymbol{H}^{i+1}$. Here and later, we omit the layer index i for attention result $\boldsymbol{A}$ for visibility.

B Analysis of Trained Weights

We investigated trained $\boldsymbol{P}$ matrices for WSC, COPA, CB, and RTE tasks with the DeBERTa-XL model. Since FC AoT P-Tuning performed better than Kronecker factorization, we selected this reparametrization method to report the results.

More specifically, we sorted rows of $\boldsymbol{P}$ matrices for each layer measured by the L_2 norm and reported the appropriate tokens for these rows. See Tables 6, 7, 9, 8 for results.

For the WSC task, there is a clear interpretation of trained rows for $\boldsymbol{P}$, since rows with a large L_2 norm represent tokens responsible for pronouns and names, which is crucial for solving WSC. For the COPA task, we observed that the model tends to assign large norms for verb tokens. For the RTE and CB tasks, $\boldsymbol{P}$ also assigns large norms for name tokens, which often occur in the training data, while CB primarily modifies adverbs for later layers.

Task	Metric	Task	Metric
CoLA	Mattews Correlation	BoolQ	Accuracy
MRPC	$\frac{\text{Accuracy}+\text{F1}}{2}$	CB	$\frac{\text{Accuracy}+\text{F1}}{2}$
RTE	Accuracy	RTE	Accuracy
SST-2	Accuracy	COPA	Accuracy
MNLI	Accuracy	MultiRC	$\frac{\text{Accuracy}+\text{F1}}{2}$
QNLI	Accuracy	WSC	Accuracy
QQP	$\frac{\text{Accuracy}+\text{F1}}{2}$	WiC	Accuracy
STSB	$\frac{\text{Pearson}+\text{Spearman}}{2}$		

Table 3: Metrics used in our experiments for each task. See Section 5.1 for more details.

[6]In fact, Transformer architecture implies evaluation of multi-head Attention. We omit this in this paper for simplicity since all derivations could be easily extended on the multi-head case.

Parameter	Range
All Tasks, except RTE	
P-Tuning v1/v2/AoT	
batch size	16, 64
learning rate	1e−4, 5e−4, 5e−3, 1e−3
p	5, 10, 20, 50, 100
Kron. r	5, 10, 25, 30, 50
FC r	32, 64, 128, 256, 512
Fine-Tuning	
learning rate	1e−5, 5e−5, 1e−4, 5e−4, 5e−3
RTE	
batch size	16, 32, 64, 128
learning rate	1e−5, 5e−5, 1e−4, 5e−4, 5e−3, 1e−3, 2e−3, 1e−2

Parameter	Range
P-Tuning v1/v2/AoT	
batch size	16, 32, 64
learning rate	5e−5, 1e−4, 3e−4, 5e−4, 1e−3, 2e−3, 5e−3
p	5, 10, 20, 50, 100
Kron. r	5, 10, 25, 30, 50
FC r	32, 64, 128, 256, 512
Fine-Tuning	
learning rate	1e−5, 5e−5, 1e−4, 5e−4, 5e−3

Table 4: Hyperparameter ranges used in experiments with GLUE and SuperGLUE benchmarking datasets for RoBERTa (left) and DeBERTa (right) models. p is the prompt length used for P-Tuning v1/v2, and r is the rank of weight factorization used for AoT P-Tuning (See Section 4.3). For GLUE experiments, each hyperparameter set was evaluated with different seed values. See Section 5.1 for more details.

	RTE	**MNLI, QQP**	**QNLI**	**Other Tasks**	**WiC**	**CB, COPA, WSC**	**MultiRC**	**Other Tasks**
Epochs	200	5	10	100	500	500	10	100
Patience	20	2	2	10	20	100	4	10

Table 5: The number of maximum epochs used for each GLUE and SuperGLUE Task. Once the Dev score stopped increasing for "patience" steps, training was halted. See Section 5.1 for more details.

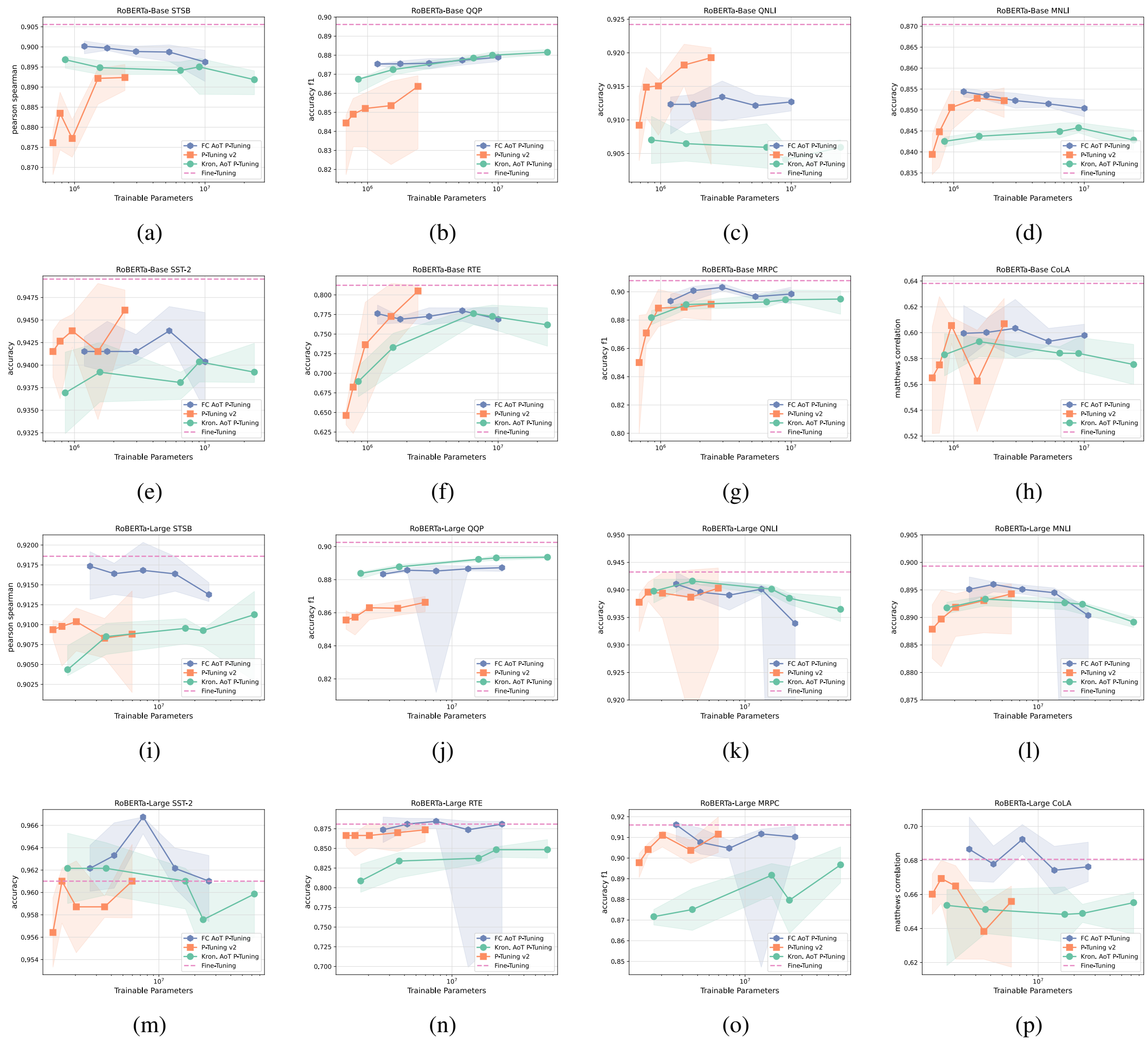
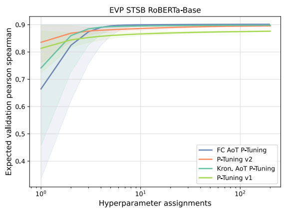
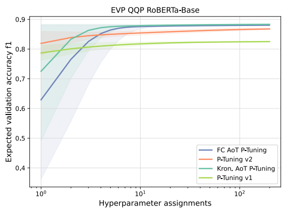


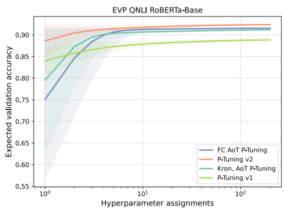
Figure 5: Per-task GLUE Benchmarking Dataset results for a different number of trained parameters of P-Tuning v2 and AoT P-Tuning with RoBERTa-Base (a-h) and RoBERTa-Large (i-p). We also provide results of plain fine-tuning for reference. See Section 5.2 for more details.



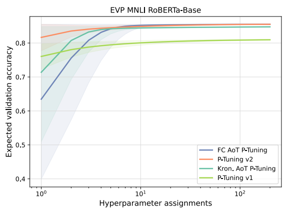
(a)



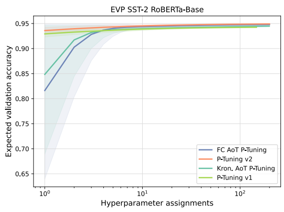
(b)



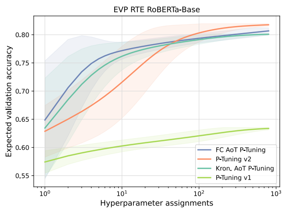
(c)



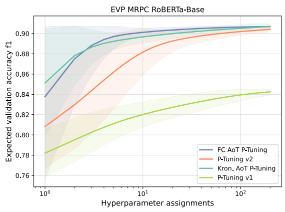
(d)



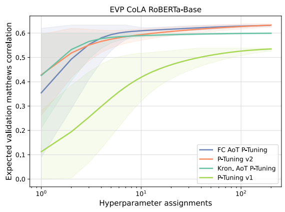
(e)



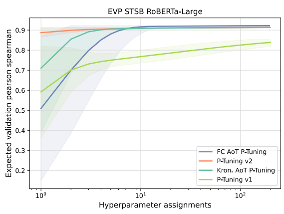
(f)



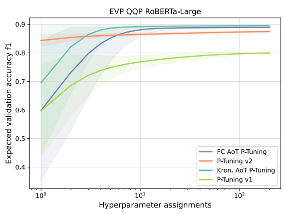
(g)



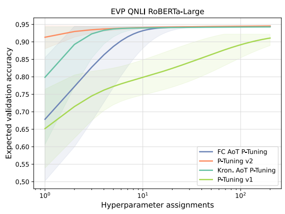
(h)



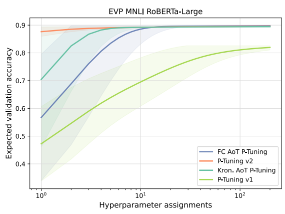
(i)



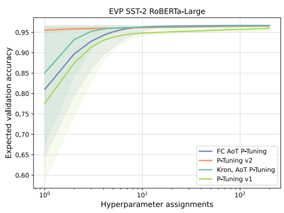
(j)



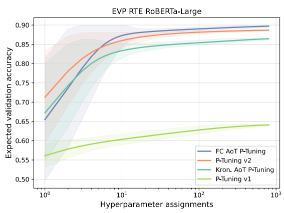
(k)



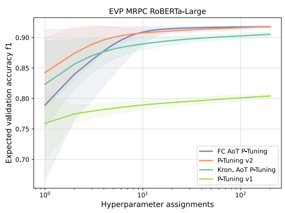
(l)



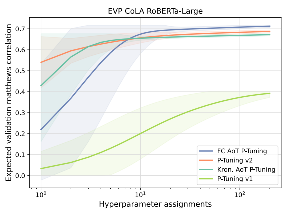
(m)



(n)



(o)



(p)

Figure 6: Expected Validation Performance (Dodge et al., 2019) of trained models with GLUE Benchmarking Datasets for RoBERTa-Base (a-h) and RoBERTa-Large (i-p). See Section 5.2 for more details.

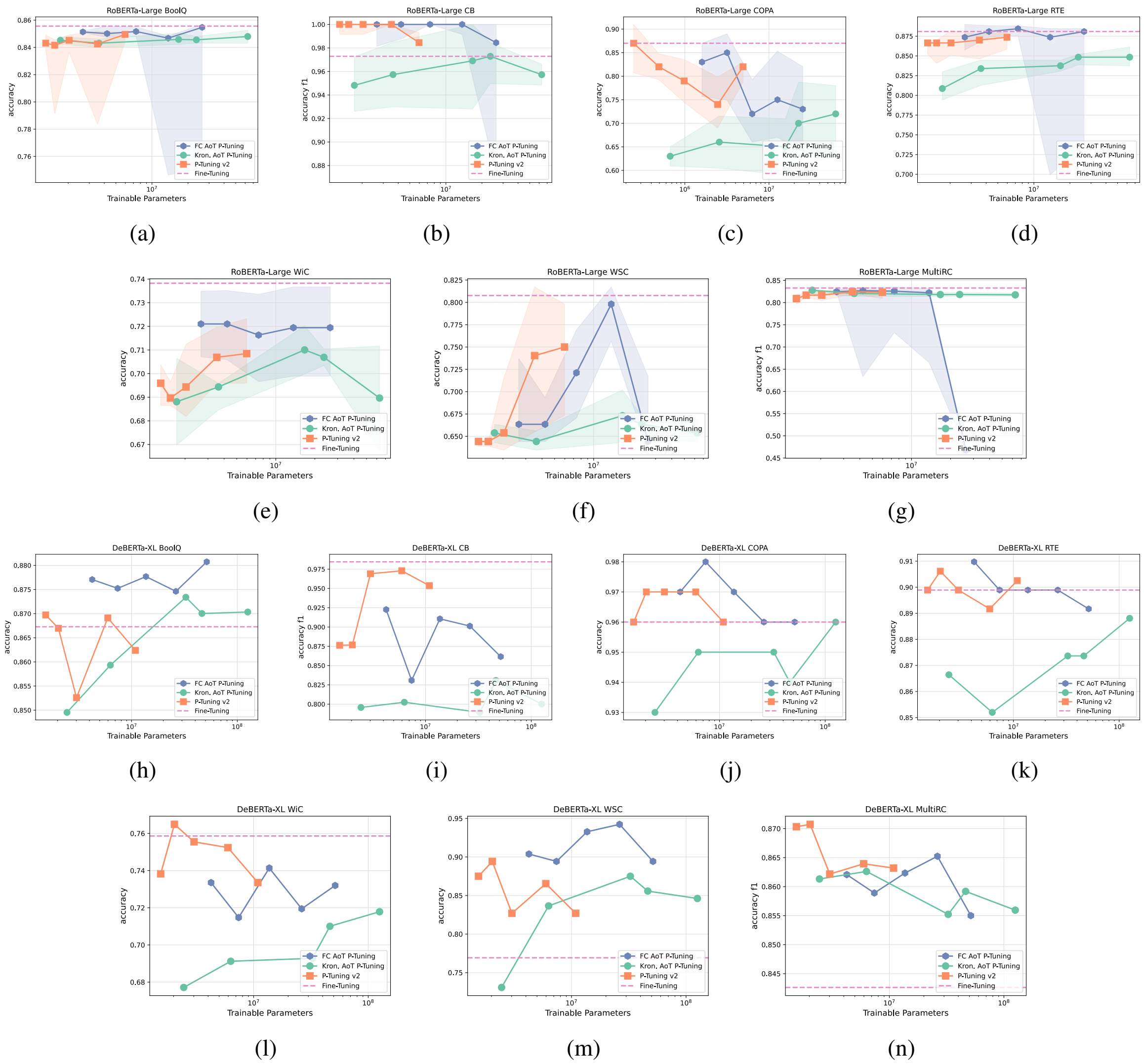


Figure 7: Per-task SuperGLUE Benchmarking Dataset results for a different number of trained parameters of P-Tuning v2 and AoT P-Tuning with RoBERTa-Large (a-g) and RoBERTa-Large (h-n). We also provide results of plain fine-tuning for reference. See Section 5.2 for more details.

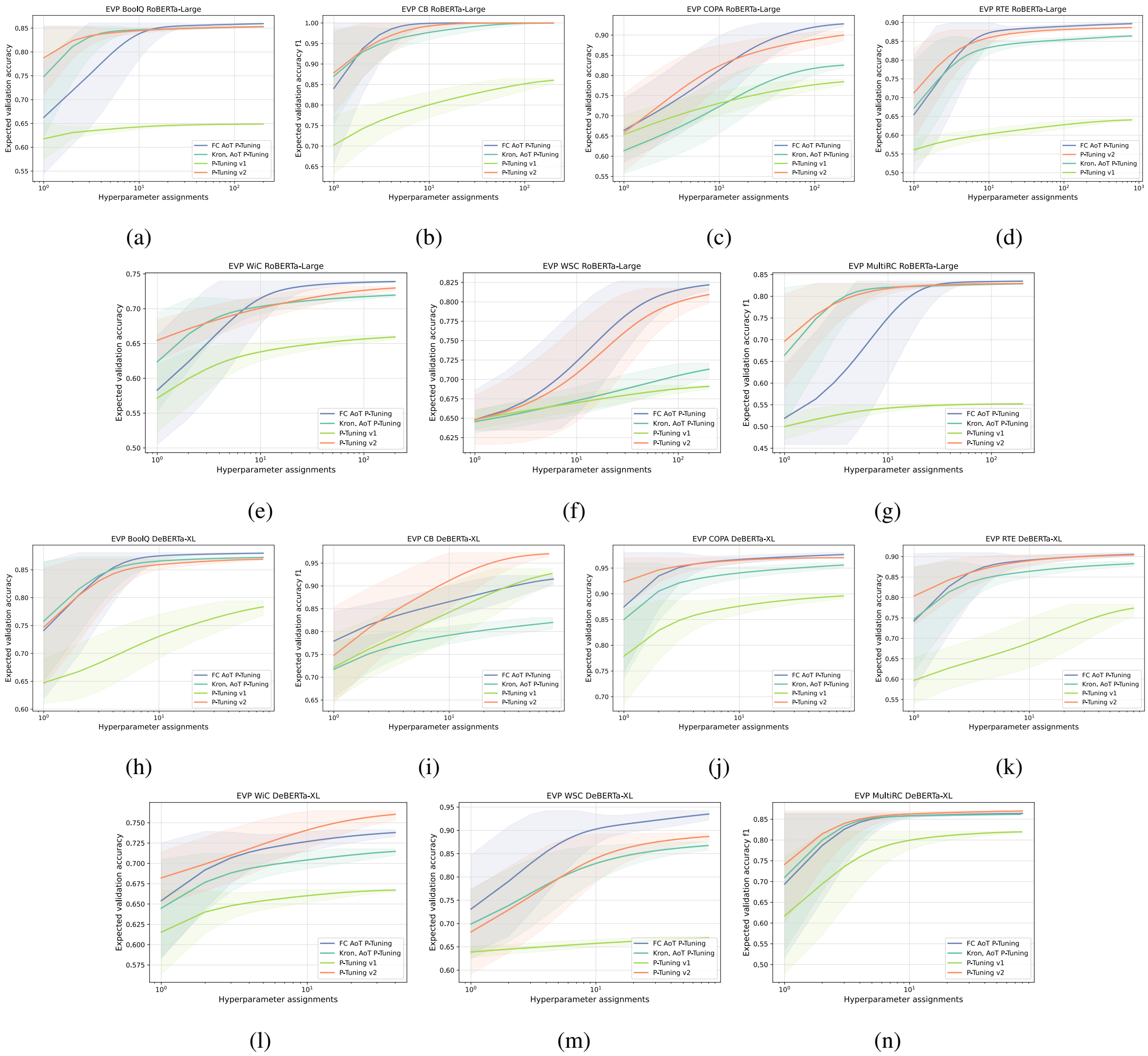


Figure 8: Expected Validation Performance (Dodge et al., 2019) of trained models with SuperGLUE Benchmarking Datasets for RoBERTa-Base (a-g) and RoBERTa-Large (h-n). See Section 5.2 for more details.

$l\#$	Tokens x with largest norm $\|\|\boldsymbol{P}_x\|\|_2$
0	likes, a, is, loves, was, to, as, wants, s, ., pony, eded, himself, Man, were, and, I, has, I, are, Frank, ., hates, As, A, A, like, It, crop, Frank, After, ,, joins, As, Eric, Likes, It, just, would, onna, him, To, behaving, after, in, because, behaves, Is, We, Like
5	,, ., narrower, doorway, backdoor, window, lousy, shortest, nicer, checkpoint, knob, thinner, narrowing, oub, quieter, BAD, ;, VID, rectangle, tighter, crappy, intruder, tongues, fing, rimination, blocker, and, raiding, detector, unmarked, sharper, knife, coolest, thicker, hoops, DOWN, lightsaber, asshole, millisec, KEY, sharp, token, slashing, Defenders, jug, Donna, slider, wedge, dding, kb
10	her, Her, herself, him, above, she, out, hers, him, HER, She, she, care, HER, above, Her, bold, CARE, cared, over, harder, louder, Above, smarter, sooner, her, cares, better, Out, vind, stronger, She, taller, tougher, Him, ahead, so, HIM, Susan, happier, up, Harry, aloud, higher, Above, SHE, could, apart, barking, inem
22	., there, dry, for, There, Her, her, sword, the, arse, wy, dry, duc, The, it, took, cr, Rig, og, There, landing, the, wide, centrally, red, grass, sw, oa, above, engine, FT, spir, cd, Coun, Ross, there, ws, guy, starter, mans, aniel, green, freely, d, wide, stall, far, artz, THERE, didn
32	it, me, olit, Polit, Pat, him, Private, Susan, pat, he, her, Self, Ins, Doc, Coun, Ang, Aut, Sil, ochond, me, Nob, IT, Senator, Professional, Dri, itized, Je, Capt, Hillary, Whe, He, Kid, Registered, itious, Michelle, Political, It, Shut, Phot, BIT, Politics, Bit, Jacob, ruct, Young, HE, Tu, them, Mot, itu
37	him, they, it, they, her, them, their, his, he, it, its, hers, theirs, was, he, Susan, old, older, THEY, They, ITS, forth, Georg, Thom, Tom, erved, Carl, Anna, nob, anos, itans, to, Eric, itcher, Harry, Tim, Jen, them, Kid, Jeremy, JOHN, Jennifer, hands, Todd, put, Thomas, she, Dan, Michelle, s
46	chased, erved, house, houses, life, chasing, market, self, chase, raised, chester, hunt, castle, HOU, atics, Singer, western, ogenous, rounded, stretched, esian, essed, omorphic, horse, SER, central, ledge, hole, asio, Self, Self, iverse, oker, Judd, DF, aday, paced, ourced, erness, Barkley, scape, sey, ationally, owned, landers, ded, study, directed, OWN, produced

Table 6: Tokens with the largest L_2 norm of $\boldsymbol{P}$ entries for the WSC task. See Section B for more details.

l#	Tokens x with largest norm $\|\|\boldsymbol{P}_x\|\|_2$
0	fit, Loud, as, Air, Upon, Sets, Bound, Apart, scratched, sets, fit, Upon, hosted, Shot, Unt, Host, fitt, Sight, atri, Ocean, ceed, ashore, set, enture, underwent, planes, boats, Waves, Ali, shi, Active, Set, Atmosp, Airways, Host, chat, Endless, pelled, rew, ached, unct, fitted, Proud, flu, itable, anson, Bound, Assets, host, sets
5	set, Set, sets, Set, SET, Setting, setting, SET, set, padd, Setting, Sets, sets, bed, Cause, setting, the, cause, tread, itch, paddle, cause, thirsty, Khe, he, anned, this, ?, of, Cause, What, bidding, This, This, what, What, a, his, lic, The, wish, fugitive, they, Bed, Air, wake, conscience, ., crowd, Let
10	?, ?, ?!, ??, ., ?", "?,)?, ???, ’?, set, .?, !?, ?), !, ????, ...?, set, ?’, „ Set, Set, ed, to, ??, ????????, as, ?????, ?)., setting, ?„ ?’", ?], sets, ..., -, lt, —, :, lic, ???, led, ur, …, punching, of, t, ?"., sets, um
22	What, set, What, out, Set, sets, on, to, ’?, what, Set, in, WHAT, Sets, Setting, WHAT, from, Setting, dropped, of, Dig, Got, ?’, set, Exper, Gets, Ground, ...?, happened, Whatever, Your, decom, Getting, Got, overlooked, Crack,)?, He, police, !?, happens, Suc, sets, what, Detective, GOT, Whatever, SET, Getting, Flying
32	glued, hid, melted, ., sent, breaths, etz, breath, Breath, baptized, watch, putting, tongues, braces, put, hid, bleach, icating, burying, aver, lifting, Illuminati, orneys, melting, withdrawing, numb, radios, inserts, amins, avert, breathing, puts, informants, lifting, hide, conscience, recommending, withdrawn, ransom, catch, Gael, Vern, roth, ears, Put, gins, breathed, attorneys, loss, biblical
37	hid, dropped, raided, fought, ungle, Hide, destruct, smuggled, abandoned, looted, attacked, barric, slid, dodged, drop, shut, drowned, hide, destruct, buggy, battled, shutdown, Hide, Attack, hid, rawl, inaccessible, avalanche, slipped, deleted, rawling, encrypted, withdrawn, Killed, dug, dropping, hoard, weapon, swallowed, defensive, destroy, exited, destroy, fight, Fighting, lost, deny, suppress, encrypt, aggressively
45	hopped, chats, pumped, paints, backed, spun, tread, coached, reefs, privately, noodles, buddies, malls, whisper, endorsements, squeezed, pals, blush, comed, edits, rallies, gigs, recol, mocked, curs, Bare, bubbles, warmed, chat, profiles, emails, Dreams, pads, chalk, interviewed, sneakers, rocked, Gloves, hubs, docs, shaved, Rise, primaries, listened, shy, essays, whispers, leeve, girlfriends, socks

Table 7: Tokens with the largest L_2 norm of $\boldsymbol{P}$ entries for the COPA task. See Section B for more details.

l#	Tokens x with largest norm $\|\|\boldsymbol{P}_x\|\|_2$
0	gression, rium, History, orer, aic, history, oration, ré, orative, amic, history, version, ural, osa, avage, ory, lia, range, History, rica, nation, root, USE, á, ination, ulation, mentation, issance, state, rum, adal, idden, jection, oly, ó, esis, orean, discovery, ria, ada, uration, entry, ord, verse, inations, ugal, itus, olics, ESSION, ativity
5	., to, in, for, and, of, ,, s, the, be, ’s, by, on, or, from, at, or, with, :, ly, a, an, ;, on, -, ., in, under, an, as, I, and, !, about, er, In, but, ?, A, is, ed, a, that, o, ers, S, ing, now,), -
10	., of, and, for, morph, votes, elector, with, uild, igraph, tatt, Assignment, as, contribut, advant, are, hod, Voters, matically, Init, rede, olon, on, rehabilit, neum, mog, looted, req, by, Claim, the, ynchron, dule, promot, socio, portfolios, goto, vulner, vote, setup, nominate, anism, s, subscrib, iop, lihood, slot, elist, ramid, ysc
22	in, In, in, be, In, .", being, Straw, -, its, a, Majority, of, a, Latest, the, Jack, ine, latest, it, Lawyers, Watts, "., "-, Massachusetts, their, .’, been, ure, Till, ’.", Signs, .’", Seventh, ?", Taxes, Atlanta, !", electric, at, IN, ide, Current, Ladies, KP, Jersey, Students, Knights, it, Anders
32	Se, Hum, Brazil, Mur, Hur, aver, Hum, Yugoslavia, Mour, jud, a, Hawai, Pag, Kant, ibal, Malaysia, EFF, Hur, .", adj, mur, Islam, and, Guinea, Britain, Sadd, Def, Niger, ,, Holland, amus, Hay, Ma, Appro, Mur, Countries, Wid, Asians, Nor, else, Calendar, Hed, Ved, ldom, english, Hind, mur, bury, Ded, hol
37	[SEP], +., Sk, Ble, Gre, cloud, Else, ., +,, "., uran, cs, Ever, 2048, Ble, Keefe, Hyp, athan, Lib, Fra, Exp, bro, Edit, Ros, Bean, Bo, Beck, Shell, sit, !., Saud, Phys, -, shell, Ol, BLIC, -, Over, ea, orthy, Shot, pn, pas, ester, Reviewed, Spe, sell, 2024
45	Chance, Sw, chance, Nine, Shares, Chance, Scientists, Tw, Besides, Prof, chances, Sn, sw, TW, EFF, J, IJ, Besides, chance, Between, icist, GU, SW, pan, Ja, Psy, tw, Between, xon, Bj, Conj, Shares, Moh, UTH, Prediction, science, intend, Science, iov, Nine, jp, dds, NJ, Jr, y, Nin, etsy, Ibid, ymm, Reporting

Table 8: Tokens with the largest L_2 norm of $\boldsymbol{P}$ entries for the RTE task. See Section B for more details.

l#	Tokens x with largest norm $\|\|\boldsymbol{P}_x\|\|_2$
0	.”, didn, doesn, don, ”., ,”, Didn, Doesn, didn, doesn, Don, wouldn, Wouldn, Does, ”, couldn, DON, Did, hadn, But, Don, Isn,).", DON, shouldn, “, Obviously, Obviously, Isn, don, hasn,))., Does, "?,].", wasn, Did,],", ,., Naturally, ...",),", Would, “, But, ”;, Naturally,]., ,), DOES
5	,, ’t, !,, .,, ?,, didn, , not, to, +,, *,,),, the, .,],, considered, doesn, in, , ,), a, /,, ,[, you, don, ,,, ,., shouldn, (),, hasn, ;, for, thought, weren, hadn, thought, wasn, NOT, hair, ’, .;, aren, ‘,, Said, ,, couldn, isn, .—, idered
10	Shant, Georg, Expect, Led, Assistant, Amph, Registered, Ear, McA, THEIR, Prev, Emb, -,, Called, Gw, Alc, Until, Rhod, Introduced, that, Lat, Unt, Ul, Sv, Gh, to, of, Fernand, ,, elta, jac, unch, Ov, Sebast, apologised, JOHN, !"., Ll, hid, Somewhere, Been, Recently, and, Somebody, Fram, Coh, ’)., Sty, Elsewhere, Unt
22	’s, ’re, A, ’ve, the, a, her, ’, be, A, a, have, DOES, LIKE, ?", "?, ’d,)?, ?, s, ABOUT, ", Like, Pant, didnt, ’m, ’?, E, The, doesnt, Was, re, :, ie, Surely, ’ll, Corinth, At, Across, your, their, ?,, THEY, ...?, or, Fra, HOW,)/
32	I, ’?, he, "?, ?’,)?, ’t, ”., .?, ...?, .”, ?"., ’:, He, !?, ?,, He, I, and, ?’", ?!", ?", ?)., !’, he, .:, ?!, she, +.,)!, ’., .’, !?", ,”, ’)., ???, !.,).", we, CLOSE, ‘., "!, .], .–, ????, ’/, ’re, .’"
37	., ’s, ’t, ?, ’, :, I, -, ’, ,, of, he, B, B, in, I, and, ’m, -, s, ", ’d, by, for, ;, b, on, you, !, ", He, to, /, ’ve, y, ’re, ed, with, ., ’ll, a, back, the, b, she, He, E, C
45	’t, not, NOT, the, not, Not, never, Not, ’s, ’re, NOT, ’ve, you, of, [SEP], t, nt, Never, The, in, NEVER, he, to, the, [CLS], hardly, never, neither, I, ,, ’m, cannot, no, The, annot, it, Their, me, didnt, He, and, doesnt, Ear, a, ., Never, none, if, on, nobody

Table 9: Tokens with the largest L_2 norm of $\boldsymbol{P}$ entries for the CB task. See Section B for more details.