DimA: A Parameter-efficient Fine-tuning Method with Knowledge transfer based on Transformer

Wenxuan Zhang, Min Huang, Zhuoyang Song, Qinghai Miao*
School of Artificial Intelligence
University of Chinese Academy of Sciences
Beijing, China
{zhangwenxuan21@mails.ucas.ac.cn, huangm@ucas.ac.cn, songzhuoyang20@mails.ucas.ac.cn,
*Corresponding author: miaoqh@ucas.ac.cn}

Abstract

Fine-tuning is a widely used technique for leveraging pre-trained language models (PLMs) in downstream tasks, but it can be computationally expensive and storage-intensive. To address this challenge, researchers have developed parameter-efficient methods that balance performance and resource cost. However, these methods often come with trade-offs like increased inference latency, token length usage, or limited adaptability for multitasking scenarios. This paper introduces a novel parameter-efficient method called **DimA** (**Dim**ensionality **A**ugmentation), which enhances the Transformer architecture by increasing the dimensionality. **DimA** achieves state-of-the-art results in GLUE and XSUM tasks while utilizing less than 1% of the original model's parameters. Moreover, **DimA** introduces a novel approach to knowledge transfer that enables the simultaneous utilization of knowledge learned from multiple tasks to handle new tasks. This method significantly enhances the performance of the model on new tasks. Its versatility in model structure also enables its application to various Transformer-based models.

Keywords: Transformer, Parameter-efficient, Knowledge-transfer

1. Introduction

Pre-trained language models (PLMs) based on the transformer architecture (Vaswani et al., 2017; Devlin et al., 2019; Liu et al., 2019b; Radford et al.) have demonstrated impressive performance across various downstream tasks. Fine-tuning, which involves adjusting all learned parameters, is a widely used approach to adapt PLMs (Howard and Ruder, 2018). However, the large number of parameters in PLMs makes it expensive to save and share copies of the models for different tasks (Cai et al., 2020; Ding et al., 2022). To address the cost issue, two major research directions have been proposed.

One direction aims to leverage the capabilities of large PLMs through the use of Hard prompts (Petroni et al., 2019; Brown et al., 2020; Schick and Schütze, 2021; Song et al., 2023), which modifies the form of different tasks with natural language templates to fit the frozen language models. Although hard prompts are user-friendly and yield strong performance, the efficacy relies on the model's size and the careful selection of appropriate prompts. (Shin et al., 2020; Jiang et al., 2020; Wei et al., 2022). Recent studies have also combined hard prompts with fine-tuning methods to enhance the effect of hard prompts, known as instruction tuning (Ouyang et al.; Peng et al., 2023).

The other direction is parameter-efficient fine-tuning (PEFT) methods, which adapt the model for downstream tasks using a small number of parameters. As representative approaches, **Adapter** (Houlsby et al., 2019; Pfeiffer et al., 2021) inserts

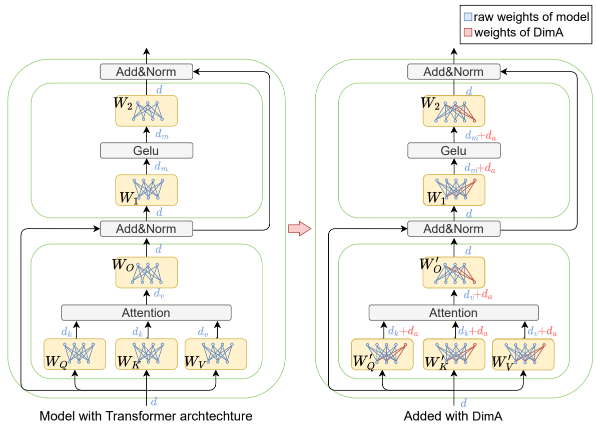

Figure 1: The encoder structure using DimA. The blue part represents the original weights of the model, while the red part represents the added weights that will extract more feature dimensions.

a two-layer nonlinear network in the model, but increases the inference delay. Whereas **Soft prompt** (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2022) replace the manually designed prompts in Hard prompts with learnable embedding, but also increases the size of the attention matrix in the computation. And methods like **LoRA** (Hu et al., 2022; Guo et al., 2021) try to impose low-rank and sparse variations on the model weights based on the assumptions of the intrinsic dimensionality (Li et al., 2018), while they still lack application in multitasking scenarios compared to AdapterFusion (Pfeiffer

LREC-COLING 2024, pages 4922–4934

et al., 2021).

This paper proposes **DimA** (**Dim**ensionality **A**ugmentation), a method by expanding the intermediate dimension in attention computation and forward network of the transformer, as shown in Fig 1. DimA achieved state-of-art results in the GLUE (Wang et al., 2018) and XSUM (Narayan et al., 2018) datasets, utilizing less than 1% of the original model's parameters. DimA enables Knowledge transfer similar to AdapterFusion but with an exceedingly small number of parameters. DimA is highly versatile and suitable for a wide range of application scenarios and models compared to other PEFT methods.

2. Relate Work

This section describes the methods to fine-tune the model and Knowledge transfer applied in multi-tasking scenarios.

2.1. Parameter-efficient fine-tuning

PEFT can perform comparably to fine-tuning by learning only a few parameters, significantly reducing the resources required to store copies of tasks.

Adapter, proposed by (Houlsby et al., 2019), embeds a two-layer nonlinear forward network after each layer of the model. However, compared to other parameter-efficient methods, Adapters still requires a large number of parameters. Additionally, the inference latency is a concern with this approach. Several variations of the Adapter method have been proposed to address these issues. AdapterDrop (Rücklé et al., 2021) enhances efficiency by removing certain modules. Another approach, Compacter, utilizes the Kronecker product approach, further reducing the number of Adapter parameters (Karimi Mahabadi et al., 2021). LST (Sung et al., 2022) separates the tuning module of the embedded model into independent pathways, significantly reducing training time. However, despite these advancements, the problem of inference latency persists.

Soft prompt (Li and Liang, 2021; Liu et al., 2022) influences the model's output by introducing learnable embedding in the attention computation process. Subsequent research (Tang et al., 2022; Jin et al., 2022) follow its focus on the attention part, which shifted towards utilizing context-related prompts instead of fixed continuous prompts, resulting in an increased number of parameters. However, all such methods increase the size of the attention matrix.

LoRA (Hu et al., 2022) assumes that the variance matrix of a model applied to a downstream task has a low intrinsic rank and uses the product of the low-rank matrix to approximate the change in weights in fine-tuning. Similar methods include BitFit (Ben Zaken et al., 2022), which focuses on adjusting only the biased part of the model. PASTA (Yang et al., 2023) targets adjustment of specifically marked embeddings within the model. Diff-pruning (Guo et al., 2021) learns a sparse parameter updating vector and a method to adjust the model weights by imposing a mask on them (Zhao et al., 2020). PST (Li et al., 2022) applies a filter on the LoRA-like approximation weights and selects a fixed number of significant weights for updating. These methods aim to learn sparse parameters to approximate fine-tuned changes. They avoid inference delays by overriding the original weights but are also challenging to use in multi-tasking scenarios compared to Adapter and Soft prompt.

Composite method (Mao et al., 2022; He et al., 2022) propose structures that integrate different methods into one. These composite methods have better results than individual methods, at the cost of increased parameter count and complexity.

2.2. Knowledge transfer in multi-tasking

Knowledge transfer is the process of leveraging knowledge from one task to improve performance in other tasks. As discussed in (Pfeiffer et al., 2021), Continuous learning (Phang et al., 2019) requires a certain sequence to learn different tasks, but there are difficulties with catastrophic forgetting (French, 1999) and sequence selection. While Multi-task learning (Caruana, 1997; Liu et al., 2019a; Zhang and Yang, 2022) requires simultaneous exposure to all tasks and balancing of samples each time, which reduces the effectiveness of rich sample tasks. (Lee et al., 2017). AdapterFusion utilizes Adapter modules acquired through independent learning and conducts inter-module attention computation at each layer to identify patterns relevant to the current task. This approach facilitates knowledge transfer across different tasks.

The pluggable embedding of independent modules in AdapterFusion offers greater convenience than fine-tuning methods that necessitate mixed learning of different tasks.

3. Method

This section introduces the structure of **DimA** and its principles in single-task fine-tuning and knowledge transfer.

3.1. DimA

According to Transformer's definitions (Vaswani et al., 2017), DimA just augments the intermediate dimensions $\{d_k, d_v, d_m\}$, as illustrated in Fig.1, which remains consistent with the Transformer architecture.

For the PLM $\mathcal{M}_\theta$, its corresponding pre-trained weights W_θ and bias b_θ with additional weights W_μ and bias b_μ provided by DimA are concatenated to achieve dimensional changes:

$$W_\theta \oplus W_\mu \to W'_\theta \tag{1}$$

$$b_\theta \oplus b_\mu \to b'_\theta \tag{2}$$

Where $W_\theta \in \mathbb{R}^{d\times d_{k/v/m}}, W_\mu \in \mathbb{R}^{d\times d_a}, W'_\theta \in \mathbb{R}^{d\times (d_{k/v/m}+d_a)}, b_\theta \in \mathbb{R}^{d_{k/v/m}}, b_\mu \in \mathbb{R}^{d_a}, b' \in \mathbb{R}^{d_{k/v/m}+d_a}$ and W_θ can be specific weights like W_Q, W_K, among others. Here the d symbol is the hidden dimension of $\mathcal{M}_\theta$, while $d_{k/v/m}$ is the intermediate dimension of the attention part and the forward network part of each layer, and d_a denotes the augmented dimension provided by DimA. It is worth noting that here the $\oplus$ symbol denotes the concatenation operation of the tensors.

For the **attention part**, The ith attention head's output, $Head_i(X)$, is concatenated and then transformed by W_O.

$$Head_i(X) = softmax(\frac{XW_Q^i {W_K^i}^T X^T}{\sqrt{d_k}})XW_V^i \tag{3}$$

$$\begin{aligned} MulHead(X) &= (\bigoplus_{i=1}^{n} Head_i(X))W_O \\ &= \sum_{i=1}^{n} Head_i(X)W_O^i \\ &= \sum_{i=1}^{n} softmax(\frac{f_\theta^i(X)}{\sqrt{d_k}})g_\theta^i(X) \end{aligned} \tag{4}$$

Where $X \in \mathbb{R}^{n\times d}, W_Q, W_K^i \in \mathbb{R}^{d\times d_k}, W_V^i, \in \mathbb{R}^{d\times d_v}, W_O^i \in \mathbb{R}^{d_v\times d}, W_O \in \mathbb{R}^{hd_v\times d}$. n is the length of input. h is the number of attention heads, and W_O^i is obtained by partitioning W_O according to h. The functions $f_\theta^i(X) = XW_Q^i(W_K^i)^T X^T$ and $g_\theta^i(X) = XW_V^i W_O^i$ replace the weights and variables of the Eq.(4).

When DimA is used for weights, the product of pairs of weights in $f_\theta^i(X)$ and $g_\theta^i(X)$ can be divided:

$$\begin{aligned} {f_\theta^i}'(X) = X{W_Q^i}'({W_K^i}')^T X^T = XW_Q^i(W_K^i)^T X^T \\ + X{W_Q^i}^\mu({W_K^i}^\mu)^T X^T = f_\theta^i(X) + f_\mu^i(X) \end{aligned} \tag{5}$$

$$\begin{aligned} {g_\theta^i}'(X) = X{W_V^i}'{W_O^i}' = XW_V^i(W_O^i) \\ + X{W_V^i}^\mu {W_O^i}^\mu = g_\theta^i(X) + g_\mu^i(X) \end{aligned} \tag{6}$$

Where ${W_Q^i}^\mu, {W_K^i}^\mu, {W_V^i}^\mu, {W_O^i}^\mu$ are the weights of DimA, according to Eq.(1) and $f_\mu^i(X)$ and $g_\mu^i(X)$ are functions of DimA.

For the **forward network**, the two-layer nonlinear network FFN(Y) accepts the normalized output Y from the attention layer:

$$FNN(Y) = \phi(YW_1 + b_1)W_2 + b_2 \tag{7}$$

Where $Y \in \mathbb{R}^{n\times d}$ and ϕ is the activation function. When using DimA, the same divide on the equation can be made:

$$\begin{aligned} FNN'(Y) &= \phi(XW'_1 + b'_1)W'_2 + b_2 \\ &= \phi(XW_1 + b_1)W_2 + b_2 \\ &\quad + \phi(XW_1^\mu + b_1^\mu)W_2^\mu \\ &= FNN(X) + FNN_\mu(X) \end{aligned} \tag{8}$$

Where W_1^μ, b_1^μ are the weights of DimA and $FNN_\mu(X)$ is function of DimA.

As shown in Eqs.(5,6,8), similar to LoRA's low-rank approximation for weight changes, DimA approximates changes at the level of weight pairs. The paired weights can also be understood as knowledge neurons (Dai et al., 2022),i.e., additional neurons are introduced to change the overall superposition effect. The weights learned by these additional neurons represent the knowledge specific to the respective task.

3.2. Single-task fine-tuning

Suppose there is a $Task_i$ $(Input, Label)$. For model $\mathcal{M}'_\theta$ adopted the DimA method, it only needs to fine-tune the W_μ part while keeping the pre-trained parameters W_θ unchanged.:

$$Prediction = \mathcal{M}'_\theta(Input|W'_\theta) \tag{9}$$

$$\arg\min_{W_\mu} \mathcal{L}(Label, Prediction) \tag{10}$$

Where $\mathcal{L}$ represents the loss function. Therefore, the learned weight W_μ contains the knowledge to apply $\mathcal{M}'_\theta$ to $Task_i$, and can be named as W_μ^i. Compared to the overall fine-tuning, W_μ^i utilizes less than 1% of the total parameter count.

During training, the augmented dimensional weight W_μ^i is trained independently and then integrated into the model's original weights during application, as shown in Fig.2.

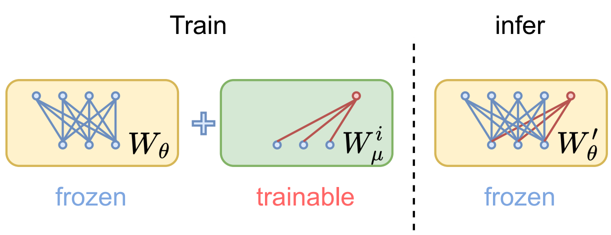

Figure 2: The difference between training and inference. The training process concatenates the W_θ with the W_μ^i, while the inference process uses the integrated weights.

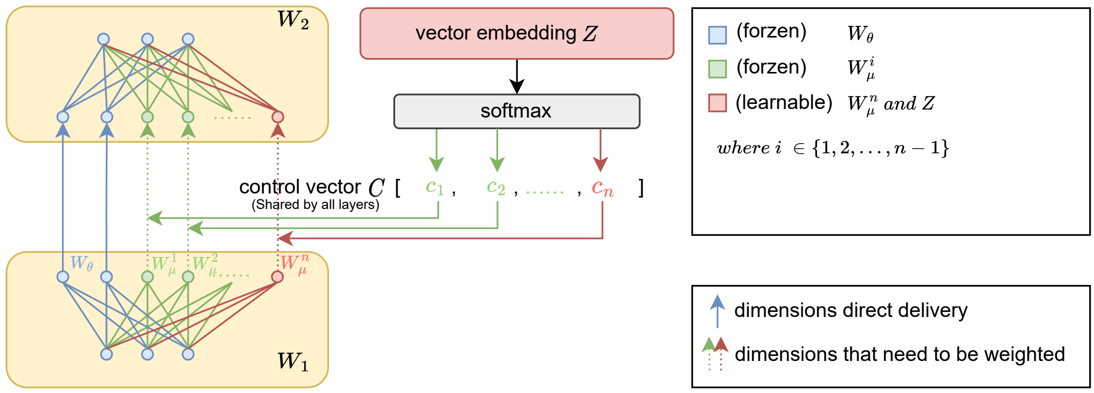

Figure 3: Demonstration of Knowledge Transfer. The green weights in the W_1 and W_2 modules come from the tasks providing the knowledge, and the red weights belonging to the current new task. The participation of these weights in the current task is controlled by a shared Dimensional weight across all layers.

3.3. Knowledge transfer

The above steps describe how W_μ^i is acquired and why it is considered as knowledge learnt from the task $Task_i$. This section will explore how W_μ^i, as task knowledge, can facilitate transfer between tasks.

Assuming the multiple tasks, denoted as $\mathbb{T} = \{Task_1, Task_2, ..., Task_{n-1}\}$, then the knowledge obtained on it by DimA method can be represented as $\mathbb{K} = \{W_\mu^1, W_\mu^2, ..., W_\mu^{n-1}\}$.

For a new $Task_n$, DimA offers a straightforward and easily portable approach to knowledge transfer. The W_μ^n that $Task_n$ itself needs to learn is also merged into $\mathbb{K}$, i.e., $\mathbb{K}' = \mathbb{K} \bigcup W_\mu^n$, using $\mathbb{K}'$ to stand in for all DimA weights.

As shown in the Fig. 3, the knowledge transfer is implemented by concatenating the weights in the knowledge $\mathbb{K}'$ onto the pre-trained weights W_θ of the model. By setting vector $C \in \mathbb{R}^n$, it is possible to control the level of involvement of different task knowledge W_μ^i to maximise the contribution to the task effect. The constraints for C are as following:

1. $\sum_{i=1}^{n} c_i = 1$: The weights c_i should add up to 1, ensuring that the contributions from different W_μ^i are balanced with the W_θ.

2. $c_i \geq 0$: The weights c_i should be non-negative, allowing for positive contributions from each W_μ^i.

After completing the aforementioned design, the Eq.(11-14) representing the knowledge transfer learning process can be derived:

$$W_\theta'' = W_\theta \bigoplus_{i=1}^{n} c_i \cdot W_\mu^i \tag{11}$$

$$b_\theta'' = b_\theta \bigoplus_{i=1}^{n} c_i \cdot b_\mu^i \tag{12}$$

$$Prediction = \mathcal{M}_\theta''(Input|W_\theta'') \tag{13}$$

$$\arg\min_{W_\mu^n, C} \mathcal{L}(Label, Prediction) \tag{14}$$

The W_θ'' represents the weights after concatenating. In this process, only C and W_μ^n need to be learned. $\mathbb{K}$ and W_θ remain frozen throughout the process. In terms of the number of parameters that need to be learned, compared to Eq.(10), only the addition of c_i introduces a relatively small number of parameters, which could be negligible.

3.4. Details of Control Vector

For the first constraint $\sum_{i=1}^{n} c_i = 1$ on the control vectors, it can be viewed in conjunction with the Eqs. (5, 11):

$$\begin{aligned} f_\theta''(X) &= f_\theta(X) + \sum_{i=1}^{n} c_i \cdot f_\mu^i(X) \\ &= \sum_{i=1}^{n} c_i \cdot (f_\theta(X) + f_\mu^i(X)) \\ &= \sum_{i=1}^{n} c_i \cdot f_\theta'(X) \end{aligned} \tag{15}$$

Where the serial number of the attention header is omitted, and Eqs.(6,8) for $g(X), h(X)$ also have the same properties as $f(X)$ as well. Each knowledge weight W_μ^i works with W_θ as W_θ', and the Eq.(15) illustrates the fact that C is essentially the assignment of weights to the knowledge learned on different tasks.

For the second constraint, the weights were set non-negative out of common sense belief that knowledge from a would either has a positive impact or no impact at all.

In practice, the direct learning parameter is not C but the vector embedding weights denoted by $Z \in \mathbb{R}^n$. The Softmax function will be applied to Z to realize these two constraints:

Task Type	Task Name
inference	MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE
similarity and paraphrase	MRPC (Aghajanyan et al., 2021), QQP
single-sentence	CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013)

Table 1: The task categorization within GLUE is applied in both the single-task fine-tuning and the knowledge transfer of the multi-task scenarios. Specifically, the MNLI, QNLI, and QQP tasks are utilized to provide the knowledge in multi-task learning. The RTE and MRPC tasks are compared to assess the presence of any biases towards similar tasks. Furthermore, the CoLA and SST-2 tasks, which lack similar tasks in MNLI, QNLI, and QQP, are used as datasets to explore the possibility of incorporating dissimilar tasks in the knowledge transfer process.

$$C = \text{Softmax}(Z) \tag{16}$$

4. Experiment

This section compares various approaches to fine-tuning based on different tasks and models. The objective is to assess the effectiveness of the DimA method in both individual task fine-tuning and knowledge transfer scenarios.

4.1. Experiment settings

Baseline For Single-task fine-tuning, the baselines chosen for comparison in this section include Adapter, Soft prompt (Liu et al., 2022), BitFit (Ben Zaken et al., 2022), LoRA (Hu et al., 2022) and FT (Fine-tuning)(Howard and Ruder, 2018). These methods are all representative and have shown remarkable performance. In the context of knowledge transfer, three methods were employed. Seq-FT (Sequence Fine-tuning) involves sequentially fine-tuning the model on the knowledge-providing datasets. The resulting model is then applied to the downstream tasks; Mul-FT (Multi-task Fine-tuning) mixes the knowledge-providing datasets, and the model learns multiple tasks simultaneously (Liu et al., 2019a). This multi-task trained model is later applied to the downstream tasks; AdapterFusion (Pfeiffer et al., 2021) shares adapter-specific knowledge obtained by individually fine-tuning the model on the knowledge-providing datasets using an attention-like computation.

Models and Optimizer This paper evaluates different scales of RoBERTa models (Liu et al., 2019b) and the GPT-2 model (Radford et al.) pre-trained by HuggingFace (Wolf et al., 2020). The experiments employ the Adam optimizer (Kingma and Ba, 2017) for all tasks.

Datasets This paper assesses the performance of models on two categories of tasks: Natural Language Understanding (NLU) tasks and Generation (NLG) tasks. The datasets were chosen from those utilized by other PEFT methods. For the NLU task, this paper selects datasets from GLUE (Wang et al., 2018) and classifies these tasks into three categories, as shown in Table.1. On the other hand, for the NLG task, the paper selects the XSUM dataset (Narayan et al., 2018), which involves generative summary summarization. This task presents challenges because of its intricate content and the constraints imposed by the language model's input length. The training text is truncated to the first 128 words at input to reduce the input length. By evaluating models on both GLUE and XSUM datasets, respectively, this paper provides a comprehensive analysis of their performance in different types of tasks.

Metrics The performance of models on the GLUE dataset is evaluated using the GLUE benchmark[1]. For the XSUM dataset, the quality and effectiveness of the generated summaries are measured using several evaluation metrics, including BLEU (Papineni et al., 2001), ROUGE-L (Lin, 2004), TER (Koehn, 2004), and Meteor (Lavie and Agarwal, 2007). These metrics quantitatively assess the generated summaries' similarity to reference summaries and their overall quality.

Hyperparameter The experimental setup gave All methods the same training epochs and optimizer. Additionally, efforts were made to maintain similar parameters across different methods, ensuring fairness in the comparison. The hyperparameter setting is shown in code[2].

4.2. Single-task fine-tuning

By evaluating various types of tasks, DimA demonstrates a slight performance advantage over existing PEFT methods in fine-tuning individual tasks. It achieves comparable results to FT on the GLUE dataset but falls slightly behind on the XSUM dataset.

4.2.1. GLUE

As shown in Table.2, DimA achieves results comparable to fine-tuning while surpassing other PEFT

[1] https://gluebenchmark.com/

[2] The source code of DimA method is available at https://github.com/mazehart/DimA.

	param(%)	RTE Acc.	MRPC F1 / Acc.	CoLA Matt.	SST-2 Acc.	QNLI Acc.	MNLI-m Acc.	MNLI-mm Acc.	QQP F1 / Acc.	average Acc.
RoBERTa-base										
FT	100	71.60(0.42)	90.30/86.57(0.42/0.14)	58.30(0.42)	95.10(0.14)	**93.07(0.04)**	**87.30(0.00)**	**87.10(0.14)**	**72.40/89.00(0.47/0.47)**	**87.11**
BitFit	0.08	70(0.49)	88.77/85.27(0.50/0.69)	58.87(1.31)	94.67(0.09)	89.17(0.05)	83.13(0.19)	83.73(0.05)	66.67/85.9(0.21/0.32)	81.34
P-tuning	0.59/7.93	67.07(3.43)	88.70/84.63(0.29/0.19)	55.50(1.35)	95.00(0.21)	89.93(0.18)	83.73(0.12)	84.07(0.25)	66.73/85.67(0.04/0.17)	84.30
Adapter	0.96	**72.20(1.04)**	90.23/86.97(0.31/0.26)	**60.40(0.71)**	94.80(0.22)	92.60(0.00)	86.16(0.44)	86.23(0.05)	71.06/88.07(0.19/0.19)	86.72
LoRA	0.71	71.83(0.66)	89.37/85.73(0.46/0.54)	57.60(0.71)	**95.23(0.24)**	92.63(0.29)	85.90(0.08)	85.70(0.28)	70.20/87.83(0.37/0.33)	86.41
DimA	0.53	71.63(0.73)	**90.60/87.36(0.28/0.17)**	58.43(1.07)	95.06(0.53)	92.97(0.04)	86.30(0.22)	86.40(0.22)	71.07/88.16(0.05/0.12)	86.84
RoBERTa-large										
FT	100	81.20(1.93)	**91.00/87.97(0.50/0.56)**	64.43(0.78)	96.37(0.46)	93.97(0.74)	90.23(0.12)	89.87(0.25)	**73.6/89.63(0.22/0.12)**	89.89
BitFit	0.07	77.43(1.02)	88.4/84.7(0.28/0.34)	62.33(0.41)	96.20(0.16)	91.97(0.05)	87.80(0.08)	87.80(0.08)	67.97/86.27(0.25/0.31)	87.45
P-tuning	0.55/7.25	82.20(0.96)	90.93/87.90(0.45/0.50)	63.23(0.96)	**96.57(0.33)**	93.17(0.41)	88.90(0.08)	88.63(0.21)	67.90/86.67(1.80/0.66)	89.15
Adapter	0.89	81.06(0.68)	90.43/87.40(0.25/0.37)	63.30(0.37)	96.30(0.29)	94.47(0.09)	89.93(0.09)	89.80(0.29)	71.70/88.43(0.08/0.04)	89.63
LoRA	0.66	79.20(1.16)	90.47/87.33(0.90/0.87)	64.30(1.80)	96.30(0.45)	94.57(0.12)	89.83(0.12)	89.50(0.00)	71.40/88.33(0.33/0.25)	89.29
DimA	0.66	**82.63(1.01)**	90.50/87.30(0.22/0.51)	**65.20(1.35)**	96.40(0.24)	**94.80(0.08)**	**90.27(0.16)**	**90.07(0.21)**	72.30/88.67(0.08/0.09)	**90.02**

Table 2: Results for the GLUE test set, with tasks listed in order of increasing data volume. The experiments were conducted in three iterations, and the average performance of each method is reported, along with the standard deviations indicated in parentheses, to account for fluctuations in task performance. The evaluation metrics used are F1 score(F1), Accuracy (Acc.), and Matthew's Correlation Coefficient (Matt.) following the GLUE benchmark.

	GPT-2					GPT-2-medium					GPT-2-large				
methods	parameter(%)	rouge-L	bleu	meteor	ter ↓	parameter(%)	rouge-L	bleu	meteor	ter↓	parameter(%)	rouge-L	bleu	meteor	ter↓
FT	100	**25.52**	**7.09**	22.97	**88.31**	100	**28.13**	**8.71**	**28.89**	**85.63**	100	**35.39**	**9.12**	**29.53**	**84.57**
P-tuning*	0.59	14.17	2.81	15.23	116.52	0.55	18.84	3.40	15.90	100.75	0.47	19.69	4.74	20.13	93.43
BitFit	0.08	19.82	3.82	19.59	92.03	0.07	21.98	5.26	22.15	90.05	0.06	22.01	5.39	22.23	89.91
Adapter	0.96	22.4	5.72	22.97	108.98	0.89	26.83	7.76	27.44	86.37	0.76	34.67	8.70	28.71	85.06
LoRA	0.94	23.16	5.39	23.02	89.7	0.87	26.15	7.12	26.37	86.49	0.76	34.40	8.39	28.53	85.18
DimA	0.53	23.45	5.55	**23.45**	89.43	0.66	26.37	7.38	26.7	86.97	0.71	35.04	8.86	29.09	84.49

Table 3: Result on XSUM, different fine-tuning methods were compared at three model scales. For P-tuning, a prefix length of 40 was deliberately chosen to maintain similar parameter amount across the different methods, even though this prefix length may not be considered ideal.

methods, on average, with the same number of parameters. Overall, the differences between the various fine-tuning methods are not significant.

4.2.2. XSUM

As shown in Table.3, for this task, FT exhibits a more pronounced advantage over PEFT methods, although this gap diminishes as the model size increases. It is worth highlighting that DimA demonstrates performance closer to FT than other methods.

4.2.3. Number of Augmented Dimensions

The number of dimensions added serves as the only hyperparameter setting for DimA, with a default configuration of one dimension per attention head. Nevertheless, this subsection explores the consequences of increasing the number of dimensions per attention head.

Experimental Design: By added 1, 3, and 9 dimensions to each attention header while keeping other settings consistent with single-task experiments, changes in task effects is observed.

Results Analysis: The results shown in Table.4 and 5 demonstrate that increasing the dimensionality has a limited impact on performance improvement. From a **task perspective**, it is observed considerable variation in the optimal number of dimensions required for different tasks. Tasks with higher accuracy tend to benefit more from fewer augmented dimensions, suggesting that they may already capture most task-specific patterns within the original model weights. From a **model perspective**, smaller models exhibit more significant improvements as the number of augmented dimensions increases, compared to larger models. This suggests that smaller models can leverage the additional capacity provided by augmented dimensions more effectively.

These findings underscore the significance of considering task characteristics and model size when determining the optimal number of augmented dimensions for improved performance.

4.3. Knowledge transfer

In this section, the GLUE dataset is divided into two portions as explained in the Table.1. The first portion consists of MNLI, QNLI, and QPP tasks, which serve as knowledge providers for the Few-shot scenarios. The second portion comprises RTE, CoLA, MRPC, and SST-2 tasks with Few-shot settings to validate the enhancement effect of Knowledge transfer.

4.3.1. Few-shot on GLUE

As shown in the Table.6, DimA+kt, which uses additional knowledge, has a significant advantage over the other methods, while its number of trained parameters is much smaller than other Knowledge transfer methods.

	RTE	MRPC	CoLA	SST-2	QNLI	MNLI-m	MNLI-mm	QQP	average
add dim for each attention head	Acc.	F1 / Acc.	Matt.	Acc.	Acc.	Acc.	Acc.	F1 / Acc.	Acc.
RoBERTa-base									
DimA (1dim)	71.63(0.73)	90.60/87.36(0.28/0.17)	58.43(1.07)	**95.06(0.53)**	**92.97(0.04)**	86.30(0.22)	86.40(0.22)	71.07/88.16(0.05/0.12)	86.84
DimA (3dim)	72.27(0.79)	90.40/87.10(0.37/0.45)	59.63(1.04)	94.90(0.00)	92.97(0.05)	86.83(0.12)	86.47(0.17)	**71.87/88.73(0.05/0.04)**	87.04
DimA (9dim)	**72.47(1.39)**	**90.67/87.37(0.17/0.25)**	**60.00(0.94)**	94.80(0.08)	92.97(0.12)	**87.00(0.08)**	**86.83(0.05)**	71.80/88.63(0.08/0.04)	**87.15**
RoBERTa-large									
DimA (1dim)	**82.63(1.01)**	90.50/87.30(0.22/0.51)	65.20(1.35)	**96.40(0.24)**	94.80(0.08)	90.27(0.16)	**90.07(0.21)**	72.30/88.67(0.08/0.09)	90.02
DimA (3dim)	82.33(2.23)	**91.40/88.43(0.45/0.46)**	63.10(1.53)	95.80(0.36)	**94.93(0.17)**	90.23(0.09)	89.80(0.17)	**72.87/89.10(0.12/0.08)**	**90.09**
DimA (9dim)	79.73(1.22)	90.97/87.97(0.34/0.47)	**65.27(1.96)**	96.30(0.00)	94.83(0.25)	**90.47(0.05)**	89.70(0.08)	72.40/88.73(0.08/0.12)	89.68

Table 4: The effect of the different DimAed dimensions on the GLUE test set, adding 1, 3, and 9 dimensions to each attention head, respectively.

	GPT-2				GPT-2-medium				GPT-2-large			
methods	rouge-L	bleu	meteor	ter ↓	rouge-L	bleu	meteor	ter↓	rouge-L	bleu	meteor	ter↓
DimA (1dim)	23.45	5.55	23.45	89.43	26.37	7.38	26.70	86.97	35.04	8.86	29.09	84.49
DimA (3dim)	23.94	6.03	24.18	**89.68**	**27.06**	**7.78**	**27.50**	**85.92**	**35.43**	**9.02**	**29.45**	**84.14**
DimA (9dim)	**24.12**	**6.05**	**24.39**	89.26	27.05	**7.78**	27.48	85.99	35.30	8.94	29.34	84.31

Table 5: The effect of different DimAed dimensions on the XSUM test set by adding 1, 3, and 9 dimensions to each attention head, respectively. To investigat the impact of increasing the number of dimensions on task effectiveness by imposing various numbers of dimensions.

4.3.2. Improvement of Knowledge transfer

For further investigation into the variations in knowledge transfer effects on enhancing different tasks, a comparison was made between DimA utilizing Knowledge transfer and DimA that does not utilize it, as presented in Table.7.

From the perspective of **task type**, it is observed that tasks like RTE and MRPC, which have similar counterparts in MNLI, QNLI, and QQP, experience more notable improvements through Knowledge transfer with DimA. Conversely, CoLA and SST-2 tasks lack similar counterparts and do not demonstrate substantial improvement. This suggests that similar tasks are crucial in Knowledge transfer for DimA.

Regarding **data size**, it is noticed that the impact of Knowledge transfer with DimA gradually reduces as the volume of task data increases. This implies that Knowledge transfer is particularly effective for tasks with relatively limited data volume, and its benefits reduce as more data becomes available.

4.3.3. The Role of control vector

Building upon the analysis of the enhancement effects of knowledge transfer, this section delves deeper into evaluating whether the control vector, which plays a central role in the knowledge transfer process, effectively reflects the importance of knowledge from different tasks.

Experimental Design Section 3.1 discusses that dimensions learned for a particular task can be viewed as a particular kind of knowledge. Drawing from the experimental design employed in the search for knowledge neurons by (Dai et al., 2022), the importance of each task-specific dimension involved in knowledge transfer can be judged by observing the effect of the gradual disappearance of that dimension on the model's confidence in predicting the correct option.

Experimental interpretation As shown in Fig.4, The 16 subgraphs represent sub-experiments conducted under different settings. In each subfigure, non-gray colors represent knowledge from existing tasks (including MNLI-m, MNLI-mm, QNLI, QQP), while gray color represents knowledge corresponding to the current task.

The upper half of each subgraph displays the variation curve of the model's confidence in predicting the correct answer. This curve shows the impact of decay weights assigned to the knowledge, ranging from 2.0 to 0.0. The fluctuation of the curve reflects the importance of the knowledge for the current task.

The lower half of each subgraph illustrates the weight assigned to the corresponding knowledge by the control vectors. Each subplot contains two parts: The upper section displays the variation curve of the model's confidence in predicting the correct answer with the decay weight of the knowledge range from 2.0 to 0.0. The lower section indicates the weight the control vector assigns to the corresponding knowledge.

If the confidence of the model in predicting the correct option sharply decreases as a knowledge gradually disappears due to decay weights, it indicates that this knowledge is important for the current task. Conversely, if the confidence does not significantly decrease, it suggests that the knowledge is not crucial for the task. By comparing the extent of the curve's decline with the weights assigned by the control vectors, we can assess whether the control vectors accurately reflect the importance of the knowledge.

Results Analysis Fig.4 demonstrates a correlation between the degree of fluctuation in the con-

Few-shot setting	FT	BitFit	P-tuning	Adapter	LoRA	DimA	Seq-FT	Multi-FT	AdapterFusion	DimA+kt
	No Knowledge transfer methods						Knowledge transfer methods			
RoBERTa-base										
learnable param(%)	100	0.08	0.59	0.96	0.71	0.53	100	100	12.61	0.53
K=50	68.57	67.82	60.63	68.39	68.80	66.78	71.05	69.37	73.41	**75.30**
	(2.67)	(1.70)	(1.69)	(1.25)	(2.60)	(2.78)	(0.83)	(3.02)	(1.29)	**(1.00)**
K=100	68.01	66.24	63.18	69.58	67.86	68.93	71.98	72.01	**76.46**	76.06
	(2.24)	(1.56)	(3.01)	(1.56)	(2.17)	(1.52)	(0.6)	(1.73)	**(0.82)**	(1.39)
K=200	75.54	72.33	69.97	74.73	74.87	74.91	71.83	73.86	80.30	**80.45**
	(1.97)	(1.22)	(2.78)	(0.60)	(0.60)	(1.20)	(3.34)	(1.92)	(0.84)	**(1.31)**
K=400	78.90	76.40	75.04	78.57	77.65	78.52	79.92	77.97	83.05	**83.92**
	(1.94)	(0.68)	(2.38)	(1.06)	(0.90)	(1.23)	(0.12)	(1.81)	(0.20)	**(0.82)**
RoBERTa-large										
learnable param(%)	100	0.07	0.55	0.89	0.66	0.66	100	100	10.50	0.66
K=50	72.23	65.35	59.72	66.12	62.34	64.55	77.89	73.29	77.91	**80.39**
	(2.52)	(2.10)	(2.67)	(2.87)	(1.90)	(3.66)	(1.78)	(2.89)	(1.65)	**(0.62)**
K=100	72.02	67.25	67.42	68.06	68.75	68.99	79.47	73.51	79.74	**81.65**
	(3.87)	(2.50)	(3.65)	(1.68)	(2.47)	(1.70)	(0.97)	(1.38)	(2.15)	**(0.56)**
K=200	77.85	69.67	74.12	72.28	74.50	75.25	82.89	79.90	84.55	**84.65**
	(0.74)	(1.01)	(1.96)	(3.94)	(1.77)	(0.88)	(0.44)	(2.34)	(0.68)	**(0.73)**
K=400	82.39	74.67	81.73	79.52	80.88	80.79	86.46	83.44	86.33	**86.64**
	(1.53)	(2.61)	(3.39)	(1.63)	(1.89)	(2.22)	(0.06)	(1.85)	(0.62)	**(0.72)**

Table 6: Few-shot results on the four datasets RTE,MRPC, CoLA, and SST-2, recording the average of the accuracy and standard deviation under the three random seeds, where standard deviations are marked in parentheses. K is the number of samples in each category in the training set, validation set. The bolded portion is the optimal value, while underlining represents suboptimal. Where learnable param style is the number of parameters learned in this process as a percentage of the overall parameters, and DimA+kt is DimA with additional task knowledge used. All Knowledge transfer methods obtained knowledge from the MNLI, QNLI and QQP datasets as knowledge providers. In contrast, methods without knowledge transfer learn and test directly on the dataset.

	RTE				MRPC				CoLA				SST-2				average
Same type of task	MNLI,QNLI				QQP				-				-				
	k=50	k=100	k=200	k=400	k=50	k=100	k=200	k=400	k=50	k=100	k=200	k=400	k=50	k=100	k=200	k=400	
RoBERTa-base																	
DimA	50.06	50.66	58.72	63.42	72.71	72.14	79.98	82.76	57.91	64.33	70.76	77.12	86.43	88.57	90.18	90.79	72.28
	(3.98)	(1.27)	(1.99)	(1.16)	(3.79)	(2.19)	(0.51)	(0.99)	(2.16)	(2.09)	(1.16)	(1.03)	(1.19)	(0.52)	(1.13)	(1.75)	
DimA+kt	76.90	76.77	79.78	81.83	80.88	83.01	84.64	85.87	56.89	56.25	66.89	77.28	86.54	88.19	90.48	90.71	78.93
	(0.63)	(0.75)	(1.08)	(0.21)	(0.65)	(0.93)	(1.11)	(1.63)	(1.03)	(3.76)	(2.63)	(1.13)	(1.67)	(0.10)	(0.41)	(0.61)	
	+26.84	**+26.11**	**+21.06**	**+18.41**	**+8.17**	**+10.87**	**+4.66**	**+3.11**	**-1.02**	**-8.08**	**-2.87**	**+0.16**	**+0.11**	**-0.38**	**+0.30**	**-0.08**	**+6.65**
RoBERTa-large																	
DimA	48.74	50.42	59.69	69.31	66.58	70.91	74.67	80.31	54.30	62.86	74.15	80.22	88.57	91.78	92.47	93.31	72.39
	(1.81)	(2.05)	(1.50)	(5.60)	(2.62)	(2.84)	(0.57)	(1.44)	(8.83)	(1.59)	(1.22)	(1.14)	(1.37)	(0.33)	(0.24)	(0.70)	
DimA+kt	87.00	86.64	87.24	87.00	80.23	81.21	84.07	86.27	62.83	67.05	74.18	79.26	91.48	91.70	93.12	94.04	83.33
	(0.63)	(0.36)	(0.55)	(0.36)	(0.86)	(0.51)	(1.49)	(0.00)	(0.60)	(1.30)	(0.99)	(1.25)	(0.40)	(0.06)	(0.20)	(0.30)	
	+38.26	**+36.22**	**+27.55**	**+17.69**	**+13.65**	**+10.30**	**+9.40**	**+5.96**	**+8.53**	**+4.19**	**+0.03**	**-0.96**	**+2.91**	**-0.08**	**+0.75**	**+0.73**	**+9.94**

Table 7: Comparison of the effects of DimA with and without Knowledge transfer. The experiments are conducted with accuracy as the measurement metric. DimA+kt refers to the utilization of Knowledge transfer. The portions in bold highlight the boosting effects observed.

fidence curve and the control vector. This implies that the control vector reflects the importance of different dimensions in Knowledge transfer. It should be noted that the gray portion represents the dimensions specific to the current task. Consequently, as the amount of task data increases, the weights associated with these dimensions gradually increase. This observation aligns with the conclusion discussed in the section 4.3.2 regarding the decay of the Knowledge transfer effect.

Task filtering The consistent trend of the flat curves with lower weights observed in the figure indicates that knowledge with lower weights has a minimal impact on the confidence level. This suggests that less weighted knowledge can be effectively filtered out using the weights assigned by the control vectors. Consequently, the need for manual screening is reduced, as the weighted knowledge provides a mechanism to automatically identify and exclude less impactful tasks.

5. Conclusion

DimA is a parameter-efficient method that achieves comparable results to Fine-tuning in single-task scenarios and allows for Knowledge transfer in multi-task scenarios. It demonstrates that knowledge of tasks exists in dimensions and can be interacted with through knowledge transfer to enhance the effectiveness of different tasks. Compared to the existing methods, it is superior in the way and effect of knowledge transfer.

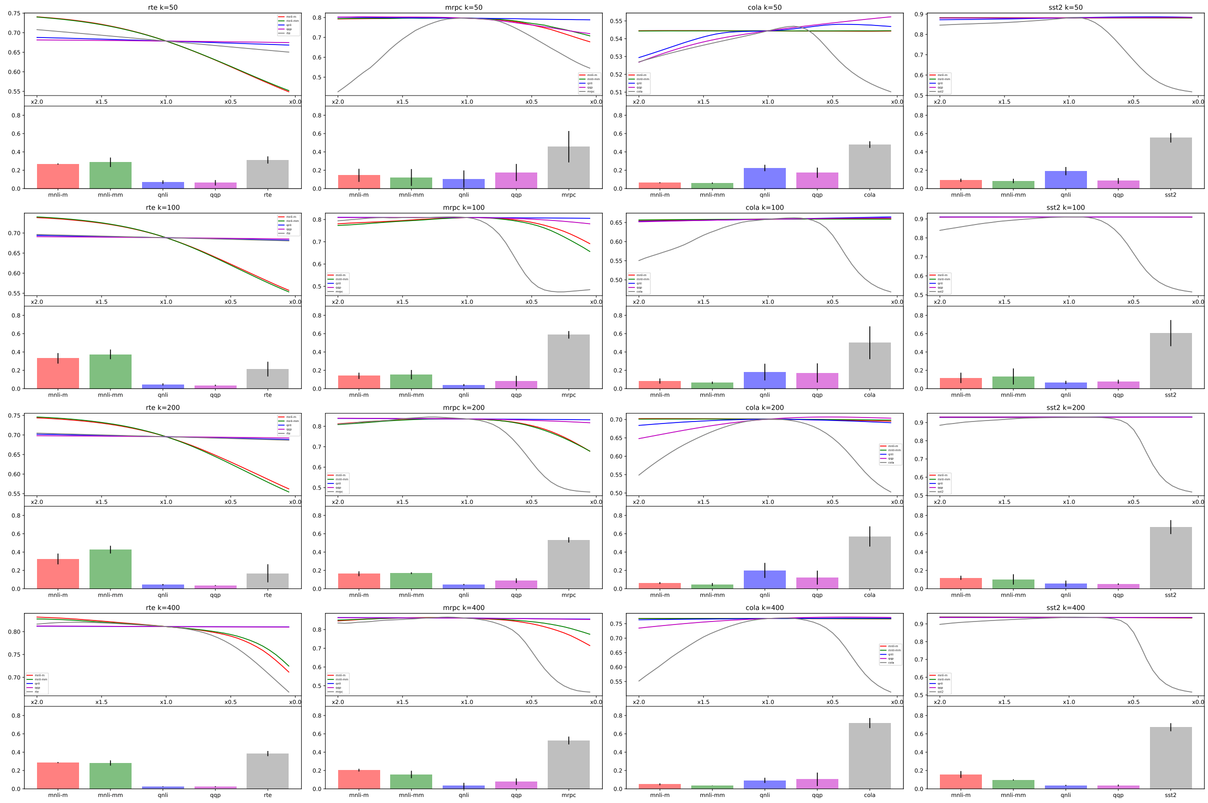

Figure 4: The weights assigned by the control vectors versus the importance of the task providing knowledge for prediction. For the sixteen subplots, the four horizontal columns are RTE, MRPC, CoLA, and SST-2 as the current prediction task, respectively; while the four vertical rows are the sample sizes for each category of K corresponding to the training and validation sets under different Few-shot settings. Each subfigure represents the average results of three independent experiments while maintaining a correlation between the task category and sample size in the horizontal and vertical directions, respectively.

Limitations

The DimA approach accomplishes task fine-tuning and Knowledge transfer by augmenting the model's dimensions. As a result, it introduces additional inference time compared to Fine-tuning, as demonstrated in the Table. 8.

	inference time(s)		
	RoBERTa-base	RoBERTa-large	%
fine-tuning	1161.31	3523.52	baseline
BitFIt	1161.40	3524.31	+0.02%
P-tuning	1199.85	3612.18	+2.71%
Adapter	1232.25	3693.90	+5.15%
LoRA	1160.65	3521.32	-0.07%
DimA	1220.92	3626.39	+3.46%

Table 8: Sum of the three inference times for the GLUE test set, labeled as a percentage increase in time relative to the baseline. However, the exact times depend on the device tested and the parameters set, and are for reference only.

Ethics Consideration

This paper uses publicly available pre-trained models and datasets with no copyright conflicts or ethical risks.

Bibliographical References

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)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. 2020. TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 11285–11297. Curran Associates, Inc.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41–75.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge Neurons in Pretrained Transformers. ArXiv:2104.08696 [cs].

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. preprint, In Review.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4).

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. 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 4884–4896, Online. 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.

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*, pages 2790–2799. PMLR. ISSN: 2640-3498.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. ArXiv:1801.06146 [cs, stat].

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LORA: LOW-RANK ADAPTATION OF LARGE LAN- GUAGE MODELS.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2022. Instance-aware Prompt Learning for Language Understanding and Generation. *arXiv:2201.07126 [cs]*. ArXiv: 2201.07126.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 1022–1035. Curran Associates, Inc.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. ArXiv:1412.6980 [cs].

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation - StatMT '07*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine

Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

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.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. MEASURING THE INTRINSIC DIMENSION OF OBJECTIVE LANDSCAPES.

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.

Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. 2022. Parameter-Efficient Sparsity for Large Language Models Fine-Tuning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 4223–4229, Vienna, Austria. International Joint Conferences on Artificial Intelligence Organization.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. ArXiv:2110.07602 [cs].

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv:1907.11692 [cs].

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, page 311, Philadelphia, Pennsylvania. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. ArXiv:2304.03277 [cs].

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2019. Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks. ArXiv:1811.01088 [cs].

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and

Iryna Gurevych. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Zhuoyang Song, Min Huang, Qinghai Miao, and Fei-Yue Wang. 2023. Parallel learning for legal intelligence: A hanoi approach based on unified prompting. *IEEE Transactions on Computational Social Systems*.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning. ArXiv:2206.06522 [cs].

Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Context-Tuning: Learning Contextualized Prompts for Natural Language Generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6340–6354, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ashish Vaswani, Noam Shazeer, and et al. Parmar. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. ArXiv:2206.07682 [cs].

Thomas Wolf, Lysandre Debut, Victor Sanh, and et al. Chaumond. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiaocong Yang, James Y. Huang, Wenxuan Zhou, and Muhao Chen. 2023. Parameter-Efficient Tuning with Special Token Adaptation. ArXiv:2210.04382 [cs].

Yu Zhang and Qiang Yang. 2022. A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an Efficient Alternative to Finetuning for Pretrained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

Language Resource References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. 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 7319–7328, Online. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. ArXiv:1808.08745 [cs].

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.