Continual Pre-training Compatible with Parameter-efficient Tuning

Anonymous ACL submission

Abstract

Parameter-efficient tuning (PET), which freezes a pre-trained language model (PLM) as a backbone and tunes additional task-specific modules, is an important paradigm of adapting one PLM to serve multiple tasks. However, as the backbone PLM is continually pre-trained on emerging data to improve its capabilities, those PET modules matching its old version may face difficulties when cooperating with its new versions. In this paper, we introduce a continual pre-training framework compatible with PET modules, by introducing a mechanism to replay already learned PET modules during the process of continual pre-training. In our experiments, we continually pre-train a backbone PLM on the data from three specific domains (computer science, biomedicine, and news). The experimental results show that our framework enables the PET modules learned on older backbone versions to work well on newer backbone versions, while obtaining better PET modules for domain-related tasks.

1 Introduction

In recent years, parameter efficient tuning (PET) methods (Houlsby et al., 2019a; Hu et al., 2021) have been widely explored to adapt pre-trained language models (PLM) to specific tasks. Specifically, PET methods freeze a unified PLM as a backbone among different tasks and then inject tiny tunable modules into the backbone to stimulate the knowledge distributed in the backbone to handle specific tasks. Compared to tuning all parameters of PLMs for different specific tasks, PET methods tune far fewer parameters and have lower memory overhead in multi-task serving while achieving comparable performance (Ding et al., 2023; Zhou et al., 2022).

PET methods have shown promise in reducing the cost of tuning and deploying LLMs for services. However, these methods encounter difficulties when it comes to updating their backbone models. Recent research (Kaplan et al., 2020) has indicated that the capabilities of PLMs are closely linked to the amount of training data utilized. Training PLMs on larger datasets tends to result in better performance. Moreover, new information emerges every day, much of which is not covered by the historical pre-training data of PLMs. Consequently, the continual pre-training of PLMs is necessary to ensure both optimal performance and timeliness. As a PLM undergoes further pre-training, the PET modules corresponding to the previous version of the PLM become incompatible with subsequent PLM versions. In other words, when a backbone PLM is updated, it becomes essential to retrain all task-specific PET modules to align with the new backbone, which incurs significant costs.

This paper proposes a continual pre-training framework compatible with PET modules. More specifically, when a PLM is continually pre-trained on new emerging data, besides language modeling objectives, we adopt an additional PET-compatible objective to ensure that the changing PLM is always compatible with those task-specific modules trained on older model versions, and the entire process does not need any task-specific data. In experiments, we continually pre-train a backbone PLM on the data from three specific domains (computer science, biomedicine, and news) and evaluate whether our framework can make the continual training of PLMs compatible with PET modules. The experimental results show that our framework enables the PET modules learned on older backbone versions to work well on newer ones while obtaining better PET modules for domain-related tasks.

2 Related Work

2.1 Parameter-efficient Learning

Large pre-trained models (PTMs) have achieved breakthrough in various tasks of natural language processing (Devlin et al., 2019; Raffel et al., 2020;

Brown et al., 2020; Han et al., 2021; Bommasani et al., 2021). But training and fine-tuning PTMs require great computational and storage overhead for each task. To address this issue, parameter-efficient learning, also known as delta tuning, is proposed for efficient PTMs adaptation via only fine-tuning a fraction of model paramters while keeping the rest parameters freezed (Ding et al., 2023; Liu et al., 2023), in which the trainable parameters is called as a delta object. According to the operations on delta objects, these methods can be divided into three types: (1) *Addition-based* methods insert additional paramters into exising PTMs, such as adapters (Houlsby et al., 2019b; Mahabadi et al., 2021) and prompts (Lester et al., 2021). (2) *Specification-based* methods specify some paramters in original PTMs for fine-tuning, such as BitFit (Zaken et al., 2022). (3) *Reparameterization-based* methods reparameter existing operations in PTMs to parameter-efficient form, and then only fine-tune certain modules, such as LoRA (Hu et al., 2021). Delta tuning methods can store task knowledge in delta objects and achieve efficient PTM adaptation. However, during the process of continual pre-training, the task knowledge stored in the delta objects will gradually be forgotten by PTMs. This requires repeatedly training the delta objects, which is highly computationally inefficient.

2.2 Continual Learning

Continual learning aims to incrementally enhance models with new knowledge while mitigating the catastrophic forgetting challenge (Wang et al., 2023). Exisiting methods can be divided into three categories: (1) *Memory-based methods* maintain existing knowledge via experience replay with old training samples (Rebuffi et al., 2017; Riemer et al., 2019; Qin et al., 2022), generated pseudo examples (Shin et al., 2017). (2) *Regularization-based methods* impose additional constraints on the model parameters (Kirkpatrick et al., 2017; Chaudhry et al., 2018; Aljundi et al., 2018) and objective functions (Li and Hoiem, 2017; Dhar et al., 2019; Gou et al., 2021) to balance the old and new knowledge. (3) *Architecture-based methods* are designed to construct dedicated parameters for new knowledge while keeping original model architectures fixed to maintain old knowledge (Serrà et al., 2018; Mallya et al., 2018; Rusu et al., 2022; Qin et al., 2022). In this paper, we explore a new setting for continual pre-training, in which we require the pre-trained models to maintain the ability to utilize the task knowledge contained in delta objects.

3 Method

3.1 Paradigm Description

Given a pre-trained model, $\mathcal{M}$, and a series of tasks, $\{\mathcal{T}_i\}_{i=1}^{N}$, where N refers to the number of tasks, we first adopt the delta tuning methods to train the corresponding delta object $\mathcal{D}_i$ for task $\mathcal{T}_i$. Then as the real-world data from various sources continually growing, the pre-trained model, $\mathcal{M}$, is required to be continually pre-trained to incorporate growing information. Here, we denote the additional pre-training corpus as $\mathcal{C}$, and the pre-trained model obtained after continual pre-training as $\mathcal{M}'$. During the process, the delta objects are freezed, and we intend to preserve the ability to utilize delta objects for $\mathcal{M}'$, i.e., when we equip $\mathcal{M}'$ with the delta object $\mathcal{D}_i$, it can achieve satisfactory results on task $\mathcal{T}_i$ without further fine-tuning.

4 Framework

Our proposed method consists of four steps: (1) Obtaining Delta Objects, (2) Constructing Delta Memory, (3) Assembling the teacher and student models, and (4) Performing pre-training with knowledge distillation. We will now introduce each step in turn.

4.1 Obtaining Delta Object

The first step is to perform Delta Tuning on a series of tasks$\{T_1, T_2, T_3, ...\}$ using the pre-trained model M_o, and obtain a series of Delta Objects $\{DO_1, DO_2, DO_3, ...\}$. Our method supports most popular Delta Tuning methods, such as Prompt, Adapter, and LoRA, and only the parameters corresponding to the added Delta Module part on the original model structure need to be recorded.

4.2 Constructing Delta Memory

The second step involves constructing the Delta Memory required for knowledge distillation based on the Delta Objects obtained. There are many ways to construct Delta Memory using a series of Delta Objects. For example, we can directly use the Delta Objects as our Delta Memory Set and randomly select one from them when Delta Memory is needed in the subsequent process. This is suitable for a small number of task sets. For a large number of task sets, we can cluster the above

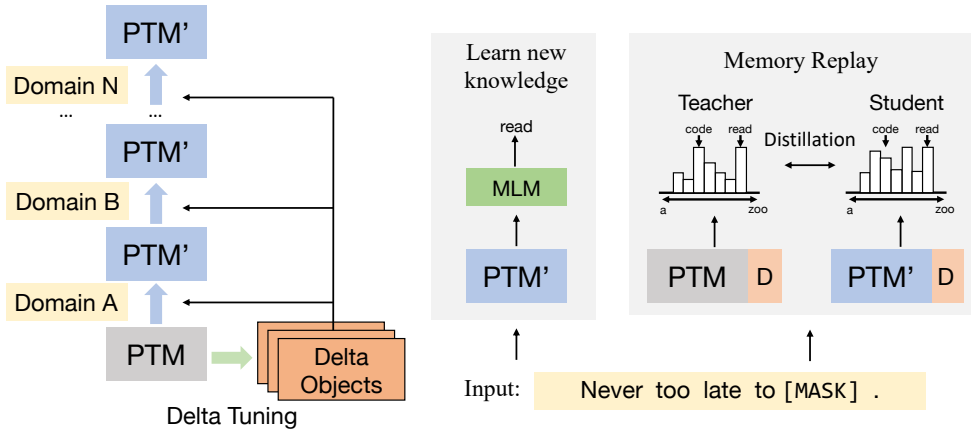

Figure 1: The architecture of our proposed model.

series of Delta Objects and obtain several cluster centers as our Delta Memory Set.

4.3 Assembling Teacher Model & Student Model

The third step is to initialize the Delta Module with the Delta Memory. First, combine it with the pre-trained model M_o to assemble the model M_o'. According to the settings of the previous steps, we can find that the model with a well-initialized Delta Module can achieve better performance on the task set. Therefore, we use it as a teacher model. At the same time, we combine the Delta Module initialized by the Delta Memory with the model M to assemble M' as the student model. We hope that in the pre-training process, M' can have similar outputs to the teacher model while ensuring adaptation to new domains. That is, under the same influence of the Delta Module, the models before and after continuous learning can maintain similar abilities, which also ensures the downward compatibility of the Delta Memory.

4.4 Pre-Training

The fourth step is to incorporate knowledge distillation into the pre-training process using the teacher model M_o' and student modelM' obtained in the previous step. Specifically, in addition to traditional MLM, we add a new pre-training task, which is knowledge distillation, using the same input. We pass masked sentences as input to the student model $M'$to obtain the probability distribution of the output at the masked position in the sentence. We also obtain the probability distribution of the output at the masked position in the sentence from the teacher model M_o' in the same way. We hope that these two probability distributions are as similar as possible, that is, the understanding and generation abilities of the two models under the current Delta Memory are as similar as possible. We use the $\mathcal{L}_{Distil}$ loss function to implement knowledge distillation, which is formulated as follows:

$$\mathcal{L}_{Distil} = \frac{1}{\sum_{i=1}^{L} m(i)} \sum_{i=1}^{L} m(i) KL(Q_i' \parallel P_i)$$

Here, L represents the number of tokens in the masked sentence, $m(i)$ represents whether the i-th token is masked (1 if masked, 0 otherwise). P_i is the predicted probability distribution of the i-th token by the student model, Q_i' is the softened probability distribution predicted by the teacher model for the same token, and $KL(Q_i' \parallel P_i)$ is the KL divergence between Q_i' and P_i, used to measure the difference between them.

$$Q_i' = Softmax(\frac{Q_i'}{T}) = \frac{\exp(\frac{Q_i}{T})}{\sum_j \exp(\frac{Q_j}{T})}$$

$$\mathcal{L} = \mathcal{L}_{MLM} + \lambda \mathcal{L}_{Distil}$$

Here λ is a weight parameter for the two loss, used to adjust the relative contribution of each loss to the overall loss function.

5 Experiments

5.1 Data

We selected the BERT as the language model and used computer science papers, biomedical papers and newstext from REALNEWS as the unlabelled training data for continual learning.We selected eight datasets from the GLUE or SuperGLUE

Delta	Domain	PT	BoolQ	SST-2	QNLI	MRPC	STSB	MNLI	CoLA	RTE	Avg.
Adapter	Original	–	84.13	95.87	93.66	87.8	86.4	88.06	84.56	84.47	88.12
	+ CS	MLM	79.57	92.43	92.54	82.55	85.09	85.36	81.21	80.87	84.95
		Distil	**82.75**	**93.81**	**93.34**	**87.07**	**86.14**	**87.08**	**84.56**	**83.39**	**87.27**
	+ BIO	MLM	80.70	91.28	92.22	85.42	84.27	83.82	82.36	80.87	85.12
		Distil	**83.09**	**93.69**	**93.19**	**86.88**	**86.92**	**87.14**	**84.18**	**83.03**	**87.27**
	+ NEWS	MLM	81.74	94.61	92.64	84.81	85.19	86.27	84.37	82.67	86.54
		Distil	**83.46**	**95.07**	**93.42**	**87.37**	**86.93**	**88.01**	**84.66**	**83.03**	**87.74**
Prompt	Original	–	81.19	95.07	93.44	82.67	85.78	85.08	84.56	68.23	84.32
	+ CS	MLM	65.20	81.77	85.89	62.84	3.20	55.2	74.59	57.40	68.98
		Distil	**80.28**	**93.12**	**92.83**	**81.76**	**79.40**	**83.1**	**82.17**	**61.37**	**82.09**
	+ BIO	MLM	64.53	81.42	81.94	65.47	2.10	54.21	72.10	55.23	67.84
		Distil	**80.61**	**93.81**	**92.87**	**83.16**	**82.51**	**83.25**	**82.65**	**63.18**	**82.79**
	+ NEWS	MLM	71.96	90.48	90.91	74.56	10.79	78.34	76.22	54.15	76.66
		Distil	**80.73**	**94.15**	**92.98**	**82.31**	**84.46**	**84.65**	**82.84**	**65.70**	**83.34**
LoRA	Original	–	84.04	95.53	94.08	89.26	87.94	89.27	86.1	85.92	89.02
	+ CS	MLM	81.65	92.66	92.45	85.23	86.31	84.99	82.07	81.23	85.82
		Distil	**83.33**	**95.18**	**93.85**	**89.02**	**87.17**	**88.69**	**85.52**	**85.20**	**88.50**
	+ BIO	MLM	80.73	91.74	92.60	84.38	84.52	84.99	82.26	83.39	85.58
		Distil	**83.73**	**95.07**	**93.63**	**89.02**	**87.27**	**88.68**	**85.81**	**84.48**	**88.46**
	+ NEWS	MLM	82.78	94.04	92.58	86.76	87.06	87.46	84.85	83.75	87.41
		Distil	**83.94**	**95.41**	**93.91**	**88.83**	**87.95**	**89.27**	**86.19**	**84.84**	**88.79**

Table 1: The performance of Delta Objects on different domain models (Domain) with different pre-training strategies (PT) tested on eight open-domain datasets, which can represent the downward compatibility of Delta Objects.

datasets as general datasets, including BoolQ, QNLI, MNLI, MRPC, SST2, STSB, CoLA, and RTE, to evaluate the degree of catastrophic forgetting of Delta Object in continual learning. Additionally, we selected five domain-specific datasets, including SciERC, ACL-ARC, ChemProt, RCT, and AGNews, to evaluate the domain adaptation ability of language models. We measured the performance of the models using traditional metrics for each dataset.

Pretraining Corpus The selection of CS, BIOMED, and REALNEWS corpora remained consistent with that of (Gururangan et al., 2020) . The reason for choosing them is that they are popular and have corresponding text classification datasets for each domain, facilitating the testing of the model's knowledge transfer ability.In addition, (Gururangan et al., 2020) suggests that the REALNEWS corpus is more similar to the original pretraining corpora for BERT, such as WIKIPEDIA and BOOKCORPUS, while CS and BIOMED are more dissimilar from other domains.
Unlike (Gururangan et al., 2020) we believe that the CS and BIOMED corpora are similar to each other. In (Gururangan et al., 2020), similarity between two corpora is measured by extracting 10K of the most frequent words from a random sample and assessing the overlap of domain word lists to measure domain similarity. While words can only capture a portion of domain similarity, the presence of numerous domain-specific nouns in the CS and BIOMED corpora can cause their similarity with other domains to be underestimated using this measurement method. Since both the CS and BIOMED corpora are chosen from scientific papers, their similarities in grammar and syntax should be much higher than their similarities in words. Therefore, we contend that CS and BIOMED are relatively similar domains, particularly for datasets that involve less specialized domain knowledge, such as GLUE and SuperGLUE.

Dataset For the general task sets, we selected eight datasets from GLUE and SuperGLUE and used traditional metrics for each dataset, i.e., accuracy, except for STSB, which was evaluated us-

ing both Pearson and Spearman correlation coefficients.

For domain-specific data, we referred to the task sets presented in (Gururangan et al., 2020). Specifically, we selected the ACL-ARC and SCIERC datasets for CS, the RCT and ChemProt datasets for BIOMED, and the AGNews dataset for REALNEWS. For all of these datasets, we evaluated the models' performance using accuracy as the metric. Our objective was to compare the models' comprehension abilities across various domains by performing Delta tuning on these datasets for different domain models.

5.2 Experimental Design

Experimental Procedure We conducted experiments using three Delta-tuning methods: Prompt-tuning, Adapter, and LoRA, following the experimental procedures outlined below:

(1) We trained the pre-training model on our general datasets using Delta tuning to obtain the Delta object for each general task.

(2) We used the proposed method and the Delta Memory set by the previous Delta object to conduct continual learning on the three domains.

(3) We evaluated the performance of the Delta Object on the three domain models and compared it with the results obtained using traditional pre-training method.

(4) We performed Delta Tuning on the aforementioned models for domain-specific tasks to evaluate the models' knowledge transfer ability.

Hyperparameter Settings The proposed method mainly involves two hyperparameters: distillation temperature and the setting of λ in the loss function. We chose to adjust the hyperparameters based on the Adapter method in the pretraining of the first domain. From the experimental results, we found that a higher distillation temperature enhances the ability to suppress forgetting, but it does not have ideal domain adaptation ability. On the other hand, a larger value of λ leads to better domain adaptation ability, but it also exacerbates catastrophic forgetting. Therefore, we need to balance between these two factors. In subsequent experiments, We chose a hyperparameter setting with a distillation temperature of 1 and a λ value of 1 in the loss function.

5.3 Experimental Results

(1) Catastrophic forgetting is observed to some degree in Adapter, Prompt, and LoRA, with Prompt showing the most severe effects and Adapter and LoRA exhibiting milder forgetting, likely due to their larger parameter sizes. Delta Object, with its larger number of parameters, has a more extensive fault-tolerant space and greater flexibility, resulting in better downward compatibility in continual learning.

(2) The degree of Delta Object forgetting is strongly correlated with characteristics of the domains , as observed in the domain settings of the experiment.

There was significant forgetting observed after pre-training on the first domain(CS), due to the considerable difference between the CS corpus and the domain trained by the original pre-training model (such as WIKI). However, there was no further increase in forgetting after two rounds of pre-training on the second domain(BIOMED), as it shared a high similarity with the CS domain, indicating that the training of first domain(CS) was sufficient. The performance improvement after pre-training on the third domain(REALNEWS) can be attributed to the high similarity between REALNEWS corpus and original pre-training corpus(such as WIKI), which allowed the Delta Object model trained on the WIKI corpus to perform well on the model pre-trained on the realnews domain, resulting in a certain degree of performance improvement.

(3) Based on performance comparisons, our proposed method is able to effectively mitigate catastrophic forgetting of pre-trained language models while still maintaining a certain level of domain adaptation ability. Compared to traditional pre-training methods, our method is able to largely maintain the performance of Delta Object during continual learning, ensuring downward compatibility. Additionally, our proposed method has varying effects on Prompt, Adapter, and LoRA due to their different parameter sizes, with larger parameter sizes in Delta Object leading to better performance maintenance.

(4) Our method involves a trade-off between retaining the old domain's knowledge and acquiring new domain knowledge, which can be

Delta	Domain	PT	CS		BIO		News
			SCIIE	ACL	RCT	ChenProt	AGNews
Adapter	Original	–	88.57	81.58	88.13	74.50	94.54
	+ CS	MLM	**90.77**	**84.21**	88.19	75.28	94.32
		Distil	90.11	**84.21**	88.21	74.78	94.38
	+ BIO	MLM	91.43	83.33	88.52	**77.09**	94.26
		Distil	90.11	82.45	**88.56**	75.36	94.38
	+ NEWS	MLM	89.01	81.58	88.21	76.35	94.48
		Distil	89.23	80.7	88.32	74.41	**94.58**
Prompt	Original	–	87.69	72.80	86.79	57.68	93.24
	+ CS	MLM	**89.67**	76.32	87.02	59.74	92.92
		Distil	88.57	**78.07**	86.90	59.53	92.98
	+ BIO	MLM	90.11	76.32	**87.71**	**62.88**	92.58
		Distil	89.01	76.32	87.34	60.12	92.90
	+ NEWS	MLM	86.37	71.93	87.31	61.88	**93.54**
		Distil	87.03	74.56	87.21	57.73	93.32
LoRA	Original	–	90.11	83.33	87.88	74.62	**94.60**
	+ CS	MLM	**91.86**	84.21	87.86	75.20	94.26
		Distil	90.99	**85.96**	88.03	74.04	94.46
	+ BIO	MLM	91.21	84.21	88.32	**77.30**	94.16
		Distil	91.43	83.33	**88.35**	75.69	94.32
	+ NEWS	MLM	90.99	81.58	88.23	75.73	94.52
		Distil	90.33	82.45	88.17	74.99	94.48

Table 2: Results of Delta-Tuning on different domain models with different pre-training strategies on domain-specific datasets are presented, and corresponding Delta Tuning methods are used for different Delta Object types. These results are averaged over five random Delta Module initializations, and accuracy is used to describe the performance on all datasets. The training order of the domains is from top to bottom in the table. The data highlighted in black represents the best Delta Tuning results for the corresponding domain under the current dataset and pre-training strategy.The results of standard deviation will be shown in the appendix.

fine-tuned by adjusting the hyperparameters in the experiment. By adjusting λ or distillation temperature as mentioned in 5.2, we can achieve the desired effect, prioritizing either the retention of old domain knowledge or the acquisition of new domain knowledge.

5.4 Limitations and Future Work

(1) The setting of domain corpora can be optimized. In this experiment, we conducted continual learning using three corpora in a fixed order. In future reasearch, we can use a more diverse range of corpora and rearrange their order to evaluate the impact of different domain corpora on Delta Object forgetting.

(2) The storage method for Delta memory can be optimized. In this experiment, we directly recorded the Delta Object of the benchmark task as a Delta memory set and randomly selected one to combine with the original model as the Teacher model. In future research, as the number of benchmark tasks increases, we can cluster multiple Delta Objects to simplify the storage space of the original Delta memory.

(3) The details of knowledge distillation can be optimized. In this experiment, performing knowledge distillation requires both the teacher and student models to perform forward computation simultaneously, which will double the time it takes to perform forward computation during pre-training. In future research, we can explore distilling knowledge on only some batches or sentences, instead of every sentence, to achieve comparable results while reducing the time cost.

(4) Additionally, building upon research in the field of continual learning, we can integrate a memory replay mechanism into the pre-training process, where the language corpus from the

original domain is merged with the current domain corpus for simultaneous training. This can further improve the performance of the model and its ability to handle a range of domains.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Computer Vision – ECCV 2018*, pages 556–572. Springer International Publishing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. 2019. Learning without memorizing.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, pages 1–16.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open*, 2:225–250.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019a. Parameter-efficient transfer learning for nlp. In *Proceedings of ICML*, pages 2790–2799. PMLR.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019b. Parameter-efficient transfer learning for nlp.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data.

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. *JMLR*, 21:1–67.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. icarl: Incremental classifier and representation learning.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2022. Progressive neural networks.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A comprehensive survey of continual learning: Theory, method and application.

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

Zhe Zhou, Xuechao Wei, Jiejing Zhang, and Guangyu Sun. 2022. PetS: A unified framework for Parameter-Efficient transformers serving. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 489–504, Carlsbad, CA. USENIX Association.