Recent Research Trend in Efficient NLP of Transformer

Eunhui Kim
Korea Institute of Science and Technology Information
ehkim@kisti.re.kr

Myunggwon Hwang
Korea Institute of Science and Technology Information
mgh@kisti.re.kr

Minho Lee
Korea Institute of Science and Technology Information
cokeman@kisti.re.kr

ABSTRACT

Deep learning models have outperformed the human cognitive performance in vision and natural language processing in 2015 and 2019, respectively. The main neural network model of this reform is currently being converted from convolutional neural network to an attention computation module-based transformer model. However, learning GPT-3 175B takes 7 months of learning time with 512 V-100 32 GB GPUs that is infeasible to companies in terms of the computational cost and time. In addition, the computational cost and time required for fine-tuning each task also pose limitations. In this regard, the latest studies that prove the efficiency and effectiveness of applying various computational transformations such as sparse attention, singular value decomposition (SVD), Fourier transform, and adapter for the attention module, which are the computational bottlenecks of the transformer model, are introduced. In order to service based on large scale language model, improvement of inference speed and improvement of switching speed between various tasks is required. This paper introduces the latest research on improving prompt tuning and speed conversion between tasks and model conversion speed to meet these industrial needs.

KEYWORDS

Language model, Efficient Design, Transformer, Prompt Tuning

1 Introduction

The development of deep learning models has had a great repercussion in the industry owing to the progress in artificial intelligence technology. In 2015, Residual Network (ResNet) [1], a type of Convolutional Neural Network (CNN) featuring the robustness of learning in the deep layer, exceeded human cognitive performance in the image processing field and attracted enormous attention in the industry beyond academia. The advent of Bidirectional Encoder Representations from Transformers (BERT) [2] in 2018 caused immense interest in the Transformer model family in the field of natural language processing (NLP). The method using BERT exhibited better performance than the individual learning method of optimization for each task traditionally processed based on NLP. It is a method of fine-tuning by task based on this pre-learning model after learning a large amount of language model BERT of 110M on a large amount of data.

In the image processing field, models after ResNet show suitable performance even in the deep layers, and the model size increases. Therefore, the need for lightweight modeling increases. In 2015, the Binary Neural Network (BNN) model [3] introduced the possibility of replacing the multiply operation of Neural Network (NN) with accumulation, using the binary weighting method of NN. Since then, lightweight modeling has been actively researched. Representative technologies include pruning, knowledge distillation, and quantization [30]. The training parameter of BERT, the initial language model of the Transformer series, is 110M. According to the power laws for neural language models [31], an increase in the amount of training data and model size entails performance improvement. Despite requiring enormous computational resources and training time, the size of the training parameters of the model is still increasing. Representative

flows include BERT (110M) [2] (2018), Roberta-large (354M) [4] (2019), T5 (10B) [5] (2020), GPT-3 (175B) [6] (2020), and Switch (1.6T) [7] (2022).

In order to increase the efficiency of large models, studies that transform the attention module, the main bottleneck of the Transformer, have been intensively conducted. Representative studies include research on Big Bird [8] that is essentially sparse attention by effectively reducing the attention module repeat patterns, Performer [9] based on Singular Value Decomposition (SVD), and FNet [10] using Fourier Transform.

Not only the amount of computation and time required for learning the exa-scale large model but also the amount of computation and time required for fine-tuning each task by the model are exceedingly large. In general, fine-tuning method copies all the training parameters of the Pretrained Language Model (PLM) and continual-learning (or transfer learning) based on the data according to a specific task. As a method of learning a task in a parameter-efficient manner, studies such as replacing the redundant operation of the Transformer module with an adapter have been proposed [20, 21, 22, 23], and by developing this, the conversion speed between multi-model and multi-task has been improved [25]. Furthermore, for large models over 24 layers, the optimized prompt execution method through the model with zero-shot and few-shot is of considerable interest [24, 25].

In the recent study conducted by Tay, both learning time and reasoning time have been shortened through the modification of the multi-head attention module[17]. This paper examines the development of fine-tuning learning methods that have been used to optimize PLM for each task with the advent of PLM. Different from this paper, we introduce studies to overcome the limitations associated with high resource requirements in fine-tuning an additionally large language model. Research on the latest parameter efficient tuning[21, 22, 23] and the latest studies that efficiently and flexibly approach various tasks through prompt tuning of large language models [24, 25] are reviewed.

At first, the architecture, the major characteristics and the computational bottleneck of a transformer are explained, and then the latest research trends are reviewed for efficient task-specific PLM beyond fine-tuning.

2 Understanding Transformer Architecture

2.1 Transformer structure and major computational bottlenecks

Transformers are neural networks usually used in sequential data processing, particularly NLP. Transformer architecture is largely divided into components called the encoder and decoder. The transformer block consists of a multi-head attention module and two-layer feed-forward neural network (FFNN) module. The input data is encoded owing to embedding (tokenized vector such as wordpiece [32] or sentencepiece [33]) and position. The transformer model is a structure in which transformer blocks are repeatedly stacked as many as the number of layers [12,13,14]. In the encoder, the contet query operation of a given input performs a bidirectional multi-head attention operation for each context key. The decoder is different from the encoder in that it applies the attention operation unidirectionally by applying a look-ahead mask, such that the word after the current word is not referenced. A decoder is an inference form that predicts the next word of a given input using the encoder's encoding.

The input text of the transformer consists of input vectors combining the tokenized embedding vector W_e and the positional embedding W_p considering the position of the token in the sequence. That is, vector representation e of the j-th token of Sequence s is

$$e = W_e[:, s_j] + W_p[:, j] \quad (1)$$

For the input represented as a vector, the dot product operation $q_i^T k_j$ between j key token k_j for the query vector q_i, of the current token i is expressed as Equation (2). This expresses the correlation between the query vector q_i and the token k_j considering the context.

$$\begin{aligned}\text{head}_{ij} &= Attention(\boldsymbol{q_i}, \boldsymbol{k_j}, \boldsymbol{v_j}) \\ &= Attention(\mathbf{W_q}\boldsymbol{e_i} + b_q, \mathbf{W_k}\boldsymbol{e_j} + b_k, \mathbf{W_v}\boldsymbol{e_j} + b_v) \\ &= softmax(\frac{\mathbf{q}_i^T \boldsymbol{k_j}}{\sqrt{d_k}})\boldsymbol{v_j}\end{aligned} \quad (2)$$

This derives the correlation data distribution between the query and the tokens in the context. The process of expressing the derived combination of context distribution and value vector as a probability value in the form of softmax is an attention module operation [12, 13, 14]. Equation (2) represents the attention operation for one head. Usually, each head operation updates the parameters of each head in parallel, and the multi-head attention operation concatenated is the same as that expressed by Equation (3).

$$\begin{aligned}&\text{MtheadAtt}(\text{q}, \text{k}, \text{v}) \\ &\quad = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_n) \times \mathbf{W}_o\end{aligned} \quad (3)$$

The amount of computation of head_i is O(B×N×N_h×d/N_h) in parallel operation. For an input sequence of length N, the attention computation $\text{Q}^T K$ requires N^2 (a quadratic computation of the input sequence length). Therefore, there are memory and time constraints.

One block of transformer has a multi-head attention layer composed of residual connections with layer normalization (as shown in Equation (4) and (5)) and a position-wise feed-forward neural network layer composed of residual connections with layer normalization.

$$\text{t}'_l = LNorm(t_{l-1} + MtheadAtt(t_{l-1})) \quad (4)$$

$$\text{t}_l = LNorm(\text{t}'_l + PwFFN(\text{t}'_l)) \quad (5)$$

Residual connection of ResNet [1] by He et al. that solves the problem of gradient vanishing as the layer deepens is borrowed from the transformer block. Different from the attention embedding processing that considers the position in the attention module, the feed forward neural network performs additional operations independent of the position.

2.2 Characteristics of each transformer layer

Geva et al. recently published an analysis study on the role of the layers of the transformer model by focusing on the two-layer FFNN of the transformer modules [15]. Of each transformer module, the lower layer of the FFNN plays the same role as a key (mapped to n-gram and semantic topics), and the upper layer of the FFNN reveals the data distribution of the corresponding key. The pattern was analyzed by mapping the activation function values for each dimension of each layer of the FFNN to the input data. That is, the degree of activation in the input sentence of n-grams (key) classified by topic was analyzed for each layer pattern. As a result, the lower layers of the transformer reveal shallow patterns, and the higher layers reveal semantic patterns. For example, a shallow pattern is a sentence group in which a sentence ends with the same word. Furthermore, in the semantic pattern, the end of a sentence group ends with an expression representing time information [15].

Clark et al. analyzed the attention layer and revealed the characteristics of each layer of the transformer. An analysis was conducted to map input tokens showing large attention values for each hidden dimension for each layer. Through this, the "CLS" token that is a special token that reveals the meaning of the entire context is frequently mapped for the low layer; the "SEP" token that is a special token that divides sentences in the middle layer is frequently mapped, and for the high layer, the "CLS" token is frequently mapped. Stop word expressions such as "the" were more frequently mapped to attention values [16].

Analysis based on the attention module and FFNN module shows similar results, wherein the upper layer reveals specific and subtle information of the input sentence.

3 Efficient NLP for various tasks based on PLM

3.1 Fine-tuning performance and improvements

The latest studies related to fine-tuning optimization for each task introduced along with the appearance of the pre-learning model are as follows. Fine-tuning was initially performed by using an optimizer such as BertAdam for 2–

3 epochs for the BERT 110M learning parameter [2]. The characteristic of the BertAdam optimizer is that it learns by applying the scaling of the learning-rate to the bias without warm-up [16]. However, the competition for each specific task of transformer-based models has sustained. When performing fine-tuning for different datasets of each task, the method of learning by re-initializing the upper layer in consideration of the role of the transformer upper layer revealing sentence details is effective [18]. To increase the effectiveness of fine-tuning, we started exploring various optimizers while learning 40–50 epochs. Recently, in addition to Adam [26], optimizers such as AdamP [27], AdamW [28], and RAdam [29] have been introduced and used in the fine-tuning competition. In order to improve the fine-tuning performance, studies have been introduced in which multi-tasks learning occurs simultaneously rather than learning individual tasks in the same dataset [18].

Linear probing is an efficient parameter learning method that updates only the last layer, unlike updating the entire PLM parameter during fine-tuning. In case of linear probing, the performance is not at par with fine-tuning. Usually, during fine-tuning and linear probing learning, better performance in in-domain than in the out-of-domain dataset is observed. It is analyzed that the cause of this performance degradation is that the pre-trained feature distribution is distorted during the process of re-learning the PLM model [19].

3.2 Parameter Efficient Tuning

In order to increase the training efficiency of large models, studies that transform the attention module, the main bottleneck of the Transformer, have been intensively conducted. Representative studies include research on Big Bird [8] that is essentially sparse attention by effectively reducing the attention module repeat patterns, Performer [9] based on Singular Value Decomposition (SVD), and FNet [10] using Fourier Transform. Recently, FNet becomes SOTA model in terms of LRA score and throughput [10, 17].

Fine-tuning undergoes an optimization process during the update of all PLM parameters based on the supervised dataset for PLM. GPT-3 [6] with 175B parameter requires 96 V100-32 GB or more for only fine-tuning, and 1 TB of storage is required every time a checkpoint is saved; the conversion time for each task consumes at least 1 min [11].

The unified model [20] that recently became a SOTA model with performance comparable to the fine-tuning method for the GPT-3 model with a parameter of 175B is one of the three representative parameter-efficient tuning methods that have been recently studied: (1) Prefix Tuning [21], (2) Adapter Tuning [22], and (3) LoRA [23] are analyzed and integrated framework.

In the performance aspects of the three existing representative parameter-efficient tuning models, the prefix tuning method exhibits performance limitations but no latency when implemented, whereas the adapter tuning method shows better performance than prefix tuning but involves latency. By applying LoRA, GPT-3 was able to perform fine-tuning using only 24 NVIDIA-32 GB GPUs, and it was reduced to 200 MB storage per checkpoint; the switching time between tasks was also reduced to seconds.

In perspective of the structure of parameter efficient tuning models, low rank adaptation (LoRA) reduces the amount of computation by using the low rank adaption weights for query and key operation of the attention module. The feed-forward operation of the attention module is the same as the general operation, but the backward operation uses weights to perform the calculation for each layer using the low rank adapter. That is, after mapping in a low dimension, it passes through non-linear activation and is mapped to the original dimension size. Prefix tuning reduces the amount of key and value product calculations by prepending the prefix values to the key and value of the multi-head attention modules. In this prefix tuning, the prepend hyperparameter

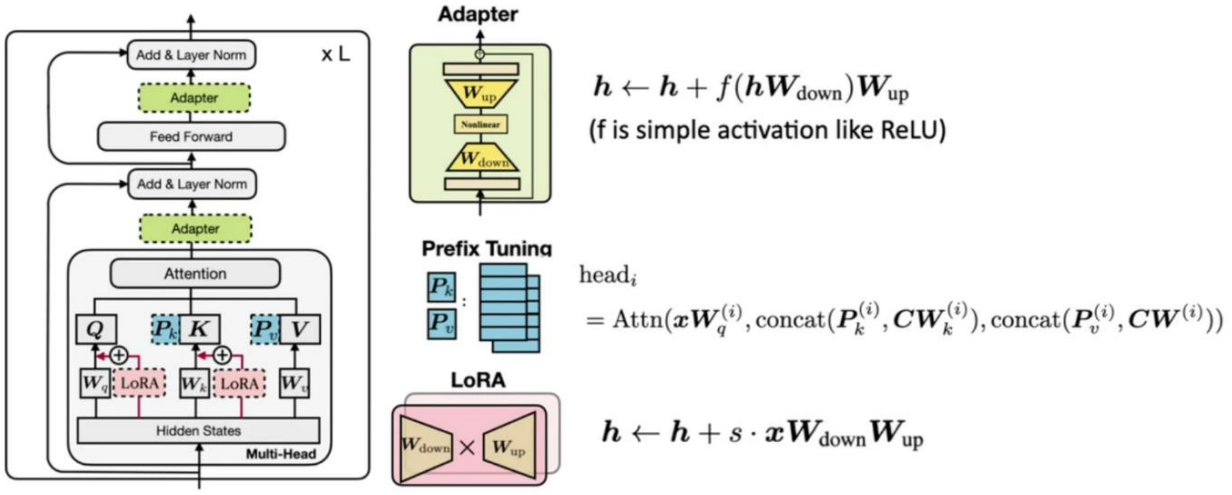

Figure 1 Three representative Parameter Efficient Tuning models: Adapter [11], Prefix Tuning [12], and LoRA [13].

plays a role of linear interpolation; thus, it is ultimately performing a similar operation to the adapter.

Figure 1 depicts these three representative parameter efficient tuning models. On the right side of the figure, each operation method of Adapter, Prefix Tuning, and LoRA is described from the top. On the left, each parameter efficient tuning method drawn on the right depicts the part applied to each block of the transformer.

Adapter is applied to the attention layer and FFNN, prefix tuning is applied to the key and value of attention module, and LoRA is applied to query and key of attention module.

Unified framework modifies the adapter by changing the sequential operation as the parallel operation and modifies LoRA by weight scaling before the "Add after Low Rank" operation [20]. The mix-and-match adapter of unified framework achieved similar results to the fine-tuning performance based on only 6.7% of the total parameter size of fine-tuning [20]. The mix-and-match adapter transforms the insertion module into parallel, and applies only a part of the multi-head attention module, FFNN module, and LoRA scaling into the proposed module.

3.3 Prompt Tuning

Prompt design conveyed the purpose of using this model to the prompt and extract the necessary knowledge without learning by copying all the model parameters.

As shown in Figure 2, the information describing the task and 2–3 examples can be inserted to form a prompt.

However, this prompt design features limitations in that it has to be made specific to the model, the number of supervised examples for input is limited, and the accuracy performance is low.

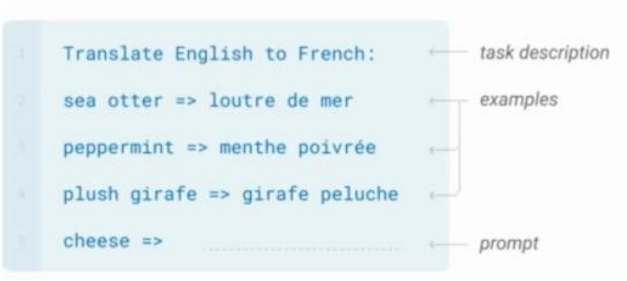

Figure 2 Prompt Design Example [24]

Prompt tuning incorporates a soft prompt (type of a virtual token) in front of the text in order to improve the limitations of the prompt design. Prompts and input representations flow similar to normal inputs to the model, and embeddings of these special tokens are learned by backpropagation. The parameters of the rest of the model are fixed. The advantages of such prompt tuning include the following: (1) it can learn based on the entire training dataset, (2) the model can be left frozen, and (3) the size of task prompts is extremely small. In comparison with such prompt tuning and general fine-tuning, fine-tuning requires memory for storing the size of the PLM model for

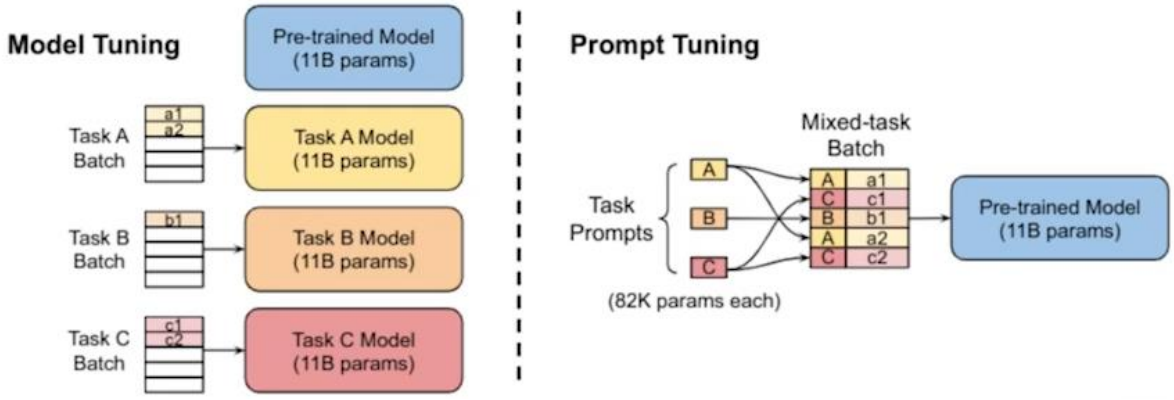

Figure 3 Difference between Model-Tuning (Fine-Tuning) and Prompt Tuning [24]

each task and requires memory for optimizing batch input data corresponding to the task. Furthermore, prompt tuning can derive results for batch input for each task by composing input batches in a fixed PLM memory size as a mixed task batch.

In other words, prompt tuning can create an efficient ensemble of large models. It is used to train a large model with small N prompts instead of N copies. Instead of performing N forward pass operations for the large model, it is suitable to perform one forward pass of N batch size inputs with different prompts for input data [24]. Figure 3 illustrates fine-tuning (model tuning) and prompt tuning.

Lester et al. [24] verified the efficiency and effectiveness of the prompt tuning method in the super glue score according to the number of model parameters of these model tuning, prompt design, and prompt tuning methods. However, this prompt tuning has a limitation in that it takes a long time to reach convergence while simultaneously learning multi-tasks, although the computation cost is low for knowledge transfer.

Accordingly, Su et al. [25] recently conducted a correlation analysis between tasks to improve the switching speed between tasks during such prompt tuning and cross-task through switching between similar tasks. Figure 4 describes correlation analysis for accelerating cross-task transfer based on the excessively large T5 model.

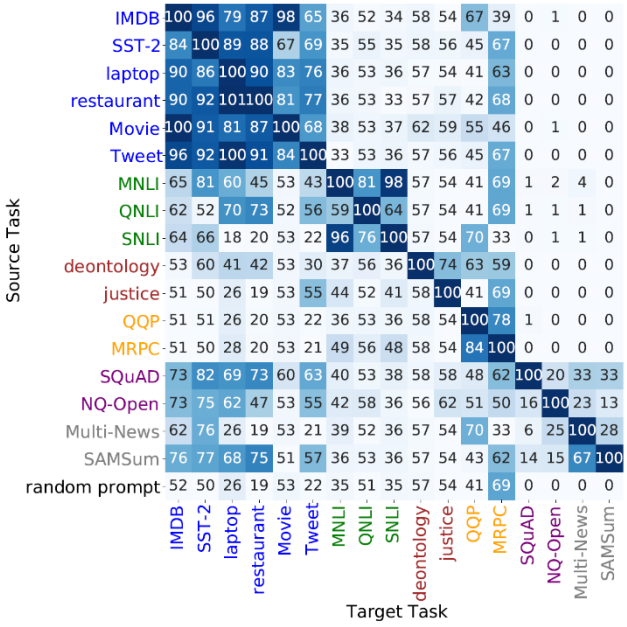

Figure 4 Correlation Analysis for accelerating cross-task transfer based on T-5 XXL model [25]

TPT_Task stands for task with initialization: when each task is initialized with the pre-learning parameters of the most similar task, then the learning speed is improved and better performance is demonstrated through experiments.

4 Conclusions

The model of the transformer series learning based on large data has improved the accuracy in the NLP and vision domains. However, when considering the value of coexistence, research on low-carbon production becomes

increasingly important. Furthermore, in the learning of an oversized model, the limitations of computational resources and time are considerable in the use of appropriate tools for the creation of productive values. For service using large scale language model, inference speed and exchange speed of the task is essential. In this study, the latest trends in transformer series for efficient NLP research considering requirement of reducing training time and inference speed are examined. The field of NLP promises considerable progress in the future, and various applications using PLM are expected.

ACKNOWLEDGMENTS

This work was supported by the Korea Institute of Science and Technology Information (grant no. K-22-L04-C07).

REFERENCES

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. *In European conference on computer vision* (pp. 630-645). Springer, Cham.

[2] Kenton, J. D. M. W. C., and Toutanova, L. K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT* (pp. 4171-4186).

[3] Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations." *Advances in neural information processing systems* 28 (2015).

[4] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[5] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140), 1-67.

[6] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.

[7] Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*, 23(120), 1-39.

[8] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 17283-17297.

[9] Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., ... & Weller, A. (2020, September). Rethinking Attention with Performers. *In International Conference on Learning Representations*.

[10] Lee-Thorp, J., Ainslie, J., Eckstein, I., & Ontanon, S. (2022, July). Fnet: Mixing tokens with fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 3949-3969).

[11] Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021, September). LoRA: Low-Rank Adaptation of Large Language Models. *In International Conference on Learning Representations*.

[12] VP Singh, H., and Mahmoud, Q. H. 2020. NLP-Based Approach for Predicting HMI State Sequences Towards Monitoring Operator Situational Awareness. Sensors, 20(11), 3228.

[13] Phuong, M., and Hutter, M. (2022). Formal Algorithms for Transformers. arXiv preprint arXiv:2207.09238.

[14] Kyong-Ha Lee, Eunhui Kim, 2021. A Study on the modern Transformer-based Language Model Techniques. ISBN Report 978-89-294-1187-9, (Oct., 2021)

[15] Geva, M., Schuster, R., Berant, J., & Levy, O. (2021, November). Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 5484-5495).

[16] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019, pp. 276–286.

[17] Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2022). Efficient transformers: A survey. ACM Computing Surveys (CSUR).

[18] Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., & Artzi, Y. (2020, September). Revisiting Few-sample BERT Fine-tuning. In *International Conference on Learning Representations*.

[19] Fabien, M., Villatoro-Tello, E., Motlicek, P., and Parida, S. (2020, December). BertAA: BERT fine-tuning for Authorship Attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)* (pp. 127-137).

[20] He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., & Neubig, G. (2021, September). Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*.

[21] Li, X. L., & Liang, P. (2021, August). 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) (pp. 4582-4597).

[22] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019, May). Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning* (pp. 2790-2799). PMLR.

[23] Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021, September). LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[24] Lester, B., Al-Rfou, R., & Constant, N. (2021, November). The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 3045-3059).

[25] Su, Y., Wang, X., Qin, Y., Chan, C. M., Lin, Y., Wang, H., ... & Zhou, J. (2022, July). On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 3949-3969).

[26] D. P. Kingma, J. Ba. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[27] H. Heo, S. Chun, S. Oh, D. Han, S. Yun, G. Kim, Y. Uh, J.W. Ha. (2021). AdamP: Slowing Down the Slowdown for Momentum Optimizers on Scale-invariant Weights. In *International Conference on Learning Representations*.

[28] I. Loshchilov, F. Hutter. (2019) Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[29] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiadong Liu, Jianfeng Gai, Jiawei Han, (2020). On the Variance of the Adaptive Learning Rate and Beyond. In *International Conference on Learning Representations*.

[30] E. Kim, K.-H. Lee, andW.-K. Sung, "Recent trends in lightweight technology for deep neural networks," *Communications of KIISE*, vol. 38, no. 8, pp. 18–29, 2020.

[31] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[32] Schuster, M., & Nakajima, K. (2012, March). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5149-5152).

[33] Kudo, T., & Richardson, J. (2018, November). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 66-71).