Incremental pre-training from smaller language models

Han Zhang[1,2], **Hui Wang**[2,*], **and Ruifeng Xu**[1,2,3*]
[1] Harbin Institute of Technology, Shenzhen, China
[2] Peng Cheng Laboratory, Shenzhen, China
[3] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies
hanlardresearch@gmail.com, wangh06@pcl.ac.cn, xuruifeng@hit.edu.cn

Abstract

Large language models have recently become a new learning paradigm and led to state-of-the-art performance across a range of tasks. As explosive open-source pre-trained models are available, it is worth investigating how to better utilize existing models. We propose a simple yet effective method, Incr-Pretrain, for incrementally pre-training language models from smaller well-trained source models. Different layer-wise transfer strategies were introduced for model augmentation including parameter copying, initial value padding, and model distillation. Experiments on multiple zero-shot learning tasks demonstrate satisfying inference performance upon transferring and promising training efficiency during continuing pre-training. Compared to training from scratch, Incr-Pretrain can save up to half the training time to get a similar testing loss.

1 Introduction

Large language models have led to state-of-the-art accuracies across a range of tasks and have demonstrated strong performances with few-shot in-context learning (Zhang et al., 2020b; Zeng et al., 2021; Brown et al., 2020). From GPT (Radford et al., 2018) to Switch-Transformer (Fedus et al., 2021), the number of parameters grows from 125 million to 1.6 trillion at an exponential rate. The study of GPT3 (Brown et al., 2020) shows that a large language model (up to 175 billion) can have strong context learning ability, and obtains comparable performances with state-of-the-art fine-tune style methods even without any parameter updating. An empirical scaling law (Kaplan et al., 2020) shows that the larger models with wider and deeper architecture are significantly more sample-efficient on a relatively modest amount of data. Furthermore, as the model size increases, there is still room for performance improvement.

However, training large language models from scratch always costs huge computing resources and time. For instance, NVIDIA leveraged their Selene supercomputer to perform scaling studies and used up to 3,072 A100 GPUs for training the largest Megatron (Shoeybi et al., 2019) model (1 trillion parameters). OpenAI spent 355 GPU-years for training GPT-3 (Brown et al., 2020), and the total costs are more than ten million dollars. Most existing model transfer methods aim at improving the performance of downstream tasks, e.g. transfer learning (Zhuang et al., 2019) or speeding up the inference process, e.g. knowledge distillation (Gou et al., 2020), but studies for accelerating model pre-training from scratch remain limited. To our knowledge, no research on how to transfer a small pre-trained model to a large model has been done.

We introduce Incr-Pretrain to augment a smaller source Transformer model to a larger target model and make them have comparable performances, both upon transferring and after continuing pre-training. Different layer-wise transfer strategies are introduced for model augmentation including parameter copying, padding and model distillation. Specifically, we propose a KL-divergence-based approximation method to distill the LayerNorm layer to address a mathematically intractable issue during transferring. We tested our method's performance on zero-shot tasks of BERT-base and GPT-2, and the results show that the augmented models obtain satisfying performances. When incrementally training a dialogue-GPT model on different scales, the training and testing losses can continue declining from the values before transferring. The total training time can be saved up to half compared with that training from scratch.

To the best of our knowledge, this is the first parameter-based method for incrementally pre-training language models. Our method can help reduce the heavy resource cost of training large language models from scratch and can be applied

* R. Xu and H. Wang are corresponding authors.

Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10), pages 36–44

to almost any open-source pre-trained model in the Transformers library (Wolf et al., 2020). The proposed method is also compatible with mainstream parallel training techniques. We summarize our contributions as follows: 1) We prove that it is feasible to train a larger language model from smaller Transformer models without training from scratch; 2) We propose a distillation-based method to transfer the LayerNorm parameters.

2 Method

We present the implementation of Incr-Pretrain in the scenarios of both widening and deepening a Transformer model. For widening the model, we use parameter copying and padding to transfer the embedding, attention and MLP layers and a distillation-based method to adjust the LayerNorm's parameters to the new input distribution due to the changed input dimension problem. For deepening the model, we initialize the deeper layers with small parameter values, the noise of which could be overwritten by the residual connection setting and have less adverse impact on the entire model. The overall framework is shown in Figure 1.

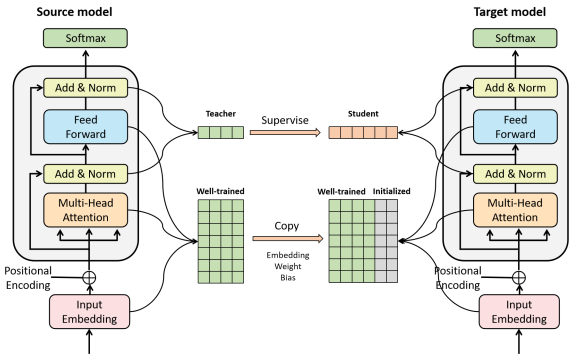

Figure 1: The Incr-Pretrain framework

2.1 Widen the model

Linear transformation is the basic operator that exists in both the multi-head attention and feed-forward layer. Incr-Pretrain transfers a smaller-size matrix of the linear transformation from the source model to a bigger matrix in the target model. As shown in Figure 2, by padding small random values or zeros at the tail of the source matrix, both vertically and horizontally, the result of the matrix multiplication is approximate to that by directly doing matrix multiplication on the source matrix. This is ensured by the block matrix multiplication rule. We also prove that if we pad random values $\theta \sim N(0, \sigma^2)$ to the dense layer, the changes on the output can be controlled in $O(\sigma^2)$ (Appendix). Especially, if σ reduces to zero, the nonzero values in the matrix multiplication result will be unchanged.

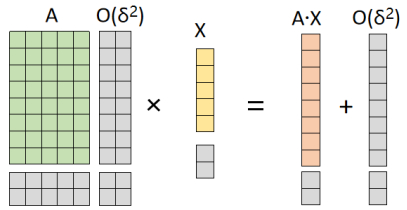

Figure 2: Block matrix multiplication

In the multi-head attention layer, parameters are the weights and biases in linear transformation for queries, keys, and values. So we can also apply the above method to the attention layer. To ensure the attention score of each head is valid, we can keep each attention head dimension fixed and only increase the head number, or keep the head number and pad values to each attention head, either combination is feasible. In the embedding layer, we directly pad small random values or zeros to the source embeddings. According to the block matrix multiplication rule, the inner product similarity of any two-word vectors will not change much, which is critical to the attention layer.

2.2 Transfer LayerNorm

LayerNorm is a technique to normalize the distributions of intermediate layers. It enables smoother gradients and faster training by re-centering and re-scaling both inputs and weight matrix. However, both re-centering and re-scaling operations are related to the hidden size, which would change after transferring. Mathematical inequivalence will affect the performance of the target model, but the scaling weight and bias can be updated fast after training several steps to adjust new model parameters.

We introduce a distillation-based method for transferring the LayerNorm layer. Let input $x \in \mathbb{R}^H$, LayerNorm re-centers and re-scales x as $h_i = g_i \cdot N(x_i) + b_i$, where $N(x_i) = (x_i - \mu)/\sigma$, $\mu = (\sum_{i=1}^{H} x_i)/H$, $\sigma = (\sum_{i=1}^{H}(x_i - \mu)^2/H)^{1/2}$. h is the output of the LayerNorm layer, $(\cdot)_i$ is the scalar value of the i-th dimension, and μ and σ are the mean and standard deviation of the input. The bias b and gain g are parameters with the same

dimension H.

Let $\hat{x} = (x_1, x_2, \ldots, x_H, \theta_1, \ldots, \theta_{D-H})$ be the input e.g. padded word embeddings, $\hat{x} \in \mathbb{R}^D$. Since we padded *D-H* values to the input, the mean and variance were changed. We define $\hat{h_i} = \hat{g_i} \cdot \hat{N(x_i)} + \hat{b_i}$, $\hat{N(x_i)} = \frac{\hat{x_i} - \hat{\mu}}{\hat{\sigma}}$, $\hat{\mu} = (\sum_{i=1}^{H} x_i + \sum_{i=1}^{D-H} \theta_i)/D$, $\hat{\sigma} = ((\sum_{i=1}^{H}(x_i - \hat{\mu})^2 + \sum_{i=1}^{D-H}(\theta_i - \hat{\mu})^2)/D)^{1/2}$. To make the outputs of source and target LayerNorm equal, for every integer i in section $[1, H]$, we need to let

$$g_i \cdot N(x_i) + b_i = \hat{g_i} \cdot \hat{N(x_i)} + \hat{b_i}, \quad \forall i \in [1, H] \cap \mathbb{N} \tag{1}$$

So we established a equation where the variables are $\hat{g_i}$ and $\hat{b_i}, i \in [1, H] \cap \mathbb{N}$. We need to find a set of solutions to Eq. 1 which are viable for $\forall(x_1, x_2, \ldots, x_H)$. In particular, for any word index $k \in [1, |Vocab|] \cap \mathbb{N}$, the equation k is

$$\begin{pmatrix} \hat{N^k}, & \mathbf{E} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{g}} \\ \hat{\mathbf{b}} \end{pmatrix} = \mathbf{h}^k \tag{2}$$

where $\mathbf{E}$ is the unit matrix, $\hat{N^k} = Diag(\hat{N(x_1^k)}, \hat{N(x_2^k)}, \ldots, \hat{N(x_H^k)})$. Unfortunately, we found that Eq. 2 is intractable, the proof is presented in Appendix.

The gain and bias are parameters that can be updated based on gradient, so we construct a loss function to train the LayerNorm parameters in the target model by calculating the KL-divergence between the outputs from the target and source. The loss function L is defined as

$$L = D_{KL}(P(x|\theta_{source}), P(x|\theta_{target})) \tag{3}$$

By minimizing the loss, the target and source LayerNorm outputs are converging.

2.3 Deepen the model

Deepening the neural network is the most common way to increase the model size. When we transfer a source model with few layers to a deeper target model, the parameters in deeper layers need to be initialized with small values. In both self-attention and MLP layers, the small parameters θ will result in small layer output $layer(x|\theta)$, so the output through residual connection $layer(x|\theta) + x$ approximates to x. It enables the deeper layers will not change the output distribution of shallow layers much, so a deeper target model can have similar output distributions to the source model.

3 Experiments

We conducted extensive experiments on inference upon transferring and continuing pre-training. We tested BERT on the cloze tasks and GPT on the next word prediction tasks, which are corresponding objectives at their stages of pre-training. To validate the time efficiency of using Incr-Pretrain, we continued to pre-train the target model and compared the loss curve with that of training from scratch.

3.1 Inference upon transferring

We tested BERT on the LAMA (Petroni et al., 2020) dataset and GPT-2 on the Lambada (Paperno et al., 2016), ClozeStory (Bugert et al., 2017), and HellaSwag (Zellers et al., 2019) datasets with a zero-shot method without any continuing pre-training.

Datasets BERT is a masked language model whose primary pre-training task is mask filling (cloze), so the performance on the cloze task is the most effective indicator. The language model tested on LAMA needs to understand the whole sentence and predict the masked keyword. Considering that some samples are too difficult to BERT in zero-shot tasks, to reduce the impact of randomness, we let BERT predict 5 times for each sample, and if any time the correct answer is predicted, we consider it correct.

GPT is a causal language model (CLM) that is pre-trained by predicting the next word with only one side of the text visible. LAMBADA, storyCloze, and HellaSwag are all datasets that aims to predict the ending text piece(s), so they are consistent with the pre-training process of the causal language model. In this task, we let GPT predict only one time on the test part of LAMBADA, the test part of StoryCloze, and the dev part of HellaSwag. On StoryCloze and HellaSwag datasets, the inference method is the same as the perplexity-based method (Zeng et al., 2021).

Configuration We compared three types of models on both datasets. The *Source* models exist as open-resource models, i.e. BERT and GPT-2. The *Target* models are the basic enlarged versions in which parameters of each layer are directly copied from the Source models with padded values. For attention, we let the number of heads increase but the dimension of each head is unchanged. We set σ as zero to make the calculation as equivalent as possible and also reduced the impact of randomness on the experiment. Compared to the target

models, the *Target-LN* models further transfer the LayerNorm parameters using the distillation-based method training on only 2,425 short dialogues (Eric and Manning, 2017).

The results of the inference tasks are shown in Table 1. We observe that after transferring, the performances of the Target models drop dramatically compared with the Source models. This is likely due to the LayerNorm part, which is not mathematically equivalent when transferring. In comparison, the Target-LN (Both GPT and BERT) models are comparable with the Source models, which shows that the distillation-based approximation method is effective.

Table 1: Results on zero-shot tasks. All datasets are evaluated by accuracy, and perplexity(PPL) is evaluated on LAMBADA. ***Target-LN** uses the distillation-based method to adjust the LayerNorm parameters of target model.

Model	Dataset	Source	Target	Target-LN
BERT	ConceptNet	26.00	15.24	22.66
BERT	Squad	15.89	10.26	15.89
BERT	Google_RE	5.06	4.29	5.23
BERT	TREx	19.45	11.22	17.49
GPT	PPL	80.37	214.1	95.56
GPT	LAMBADA	20.28	16.83	20.88
GPT	StoryCloze	59.40	53.10	59.20
GPT	HellaSwag	21.65	22.87	23.08

3.2 Continuing pre-training

We pre-trained three sizes of Dialog-GPT models on a benchmark dialog corpus (Zhang et al., 2020a) to validate our incremental pre-training method, including small, medium, and large versions. We conducted transfers between models of different sizes and the model configuration details are shown in Appendix. We pre-trained the three GPT models for 14.5k steps with the batch size 32, and performed the model transfer from the checkpoints of the 5,000th steps. The testing losses of different models are shown in Figure 3. In general, the loss values can continue declining from a smaller value after transferring. We did not use the distillation-based method to transfer LayerNorm parameters since it is no longer needed to make the model converge to the source model during continuing pre-training. As shown in Table 2, training GPT-m using Incr-Pretrain only costs 7.05k steps to reach a testing loss value comparable with that of 14.5k steps training from scratch. This shows that our method saves about 51% of the training time. However, as more parameters are padded, e.g. from small to large, the amount of training time that can be saved declines, which is likely due to pre-training larger models needing more computation. (Kaplan et al., 2020).

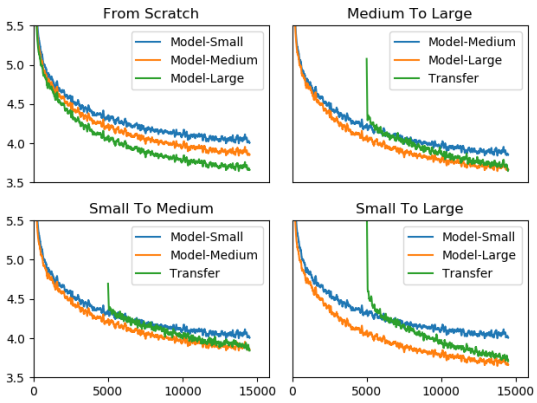

Figure 3: Testing Loss

Table 2: Pre-training time saved. Pre-training steps number when reaching the same loss. ECPN: enlargement coefficient of parameters number, PPTS: percentage of pre-training time saved.

Train Mode	Steps	ECPN	PPTS
GPT-S/M/L from scratch	14.5k	-	-
GPT-S to GPT-M	7.05k	2	51%
GPT-M to GPT-L	8.50k	3	41%
GPT-S to GPT-L	9.50k	6	34%

Further experiments showed that using our transfer method, the amount of pre-training time saved depends on not only the enlargement of the number of parameters but also the padding values. When padding zeros, the testing loss can decline starting from near to the loss value of the source model but converging slowly. Considering previous work (Glorot and Bengio, 2010; He et al., 2015) on parameter initialization, we padded smaller random values instead of zeros, and the convergence can accelerate much. More experimental details are presented in the Appendix.

4 Conclusion

We propose a transfer strategy that can incrementally pre-train language models with acceptable performance decreases. The inference performances show that the target models are comparable with the source models. The continuing pre-training experiment demonstrates that the transfer method is computationally efficient compared to pre-training a large model from scratch. Our future directions will be transferring with different parallel styles and exploring the influence of padding values.

Limitations

During the process of incremental pre-training, although this method effectively reduces training time and achieves test losses close to those obtained by training from scratch, the percentage of pre-training time saved gradually decreases as the number of model parameters increases. For instance, transitioning from a small to a medium-sized model can save approximately 51% of training time, but extending from a small directly to a large model reduces the saving to 34%. This indicates a diminishing marginal return in pre-training efficiency as the model scale expands.

Acknowledgements

This research was supported in part by the Major Key Proiect of PCL (NO.PCL2023A09 and NO.PCL2023AS7-1), the National Key Research and Development Program of China (2021ZD0112905), the National Natural Science Foundation of China (62176076), the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies(2022B1212010005), Natural Science Foundation of Guangdong (2023A1515012922), and Shenzhen Foundational Research Funding (JCYJ20220818102415032).

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.

Michael Bugert, Yevgeniy Puzikov, Andreas Rücklé, Judith Eckle-Kohler, Teresa Martin, Eugenio Martinez Camara, Daniil Sorokin, Maxime Peyrard, and Iryna Gurevych. 2017. LSDSem 2017: Exploring Data Generation Methods for the Story Cloze Test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*, pages 56–61, Valencia, Spain. Association for Computational Linguistics.

Mihail Eric and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

X. Glorot and Y. Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9:249–256.

J. Gou, B. Yu, S. J. Maybank, and D. Tao. 2020. Knowledge distillation: A survey.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.

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.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence?

Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. 2021. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020a. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL, system demonstration*.

Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, Fanchao Qi, Xiaozhi Wang, Yanan Zheng, Guoyang Zeng, Huanqi Cao, Shengqi Chen, Daixuan Li, Zhenbo Sun, Zhiyuan Liu, Minlie Huang, Wentao Han, Jie Tang, Juanzi Li, Xiaoyan Zhu, and Maosong Sun. 2020b. Cpm: A large-scale generative chinese pre-trained language model.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2019. A comprehensive survey on transfer learning.

A Experimental Details

In the distillation-based method, we used the Adam optimizer, the learning rate was set as 1E-4, batch size as 8, and the sequence length as 512. Other model configurations are shown in Table 3. We used the KVRET corpus to train the LayerNorm parameters in GPT-2 for 20k steps (about 3 hours on NVIDIA-V100) and discovered that using pseudo inputs constructed by randomly choosing word indices produced better results than those from the real corpus when transferring BERT's LayerNorm parameters.

In the continuing pre-training experiments, we used the Adam optimizer and linear warm-up at the first 100 steps, the learning rate was 1.5E-4, and the batch size was 32. As shown in Figure 4, we compared two padding strategies, padding zeros and padding random values $\theta \sim N(\mu, \sigma^2), \mu = 0, \sigma = 0.02$. When padding zeros, the testing loss starts from that of before-transferring, and it proved that our transferring method is feasible. However, the test loss converged slowly since padding zeros disturbed the initialization distribution. To speed up the incremental training process, we padded random values that follow a normal distribution with small variance instead of zeros. Although padding random values breaks the mathematical equivalence of transferring a bit and the loss value is higher at the beginning, the acceleration for the convergence is remarkable.

Table 3: Model configurations for inference

Model	Heads	Layers	Dim	Total
Source BERT	12	12	768	119.5M
Source GPT	12	12	768	124.4M
Target(LN) BERT	16	12	1024	184.0M
Target(LN) GPT	16	12	1024	203.7M

Table 4: Model configurations for pre-training

Model	Heads	Layers	Dim	Total
Model-Small	6	6	384	15.8M
Model-Medium	8	8	512	32.2M
Model-Large	12	12	768	95.5M

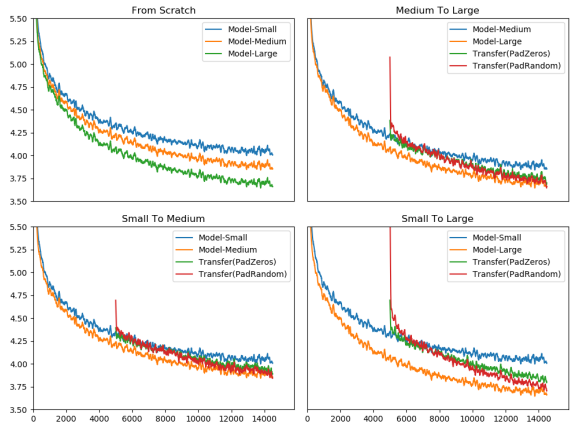

Figure 4: Comparison on the testing loss

B Proof-1

We prove that if we pad random values $\theta \sim N(0, \sigma^2)$ to the parameters of the dense layer, the changes of output can be controlled in $O(\sigma^2)$.

For simplicity of proof, we consider a fully connected layer with a mapping function $F : \mathbb{R}^H \to \mathbb{R}^H$, $F(x) = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{H \times H}$, $\mathbf{b} \in \mathbb{R}^H$, and $\mathbf{x}$ is a input column vector such as word embeddings. The padded layer can be expressed alike, i.e. $\hat{F} : \mathbb{R}^D \to \mathbb{R}^D$, $\hat{F}(\hat{\mathbf{x}}) = \hat{\mathbf{A}}\hat{\mathbf{x}} + \hat{\mathbf{b}}$, $\hat{\mathbf{A}} \in \mathbb{R}^{D \times D}$, $\hat{\mathbf{b}} \in \mathbb{R}^D$, where

$$\hat{A} = \begin{pmatrix} \mathbf{A}, & \Lambda_1 \\ \Lambda_2, & \Lambda_3 \end{pmatrix}, \hat{b} = \begin{pmatrix} \mathbf{b} \\ \beta \end{pmatrix}, \hat{\mathbf{x}} = \begin{pmatrix} x \\ \chi \end{pmatrix},$$

All of the $\Lambda_i (i = 1, 2, 3)$, β, and χ are independent and identically distributed to the Gaussian distribution $N(0, \sigma^2)$, while A and x are constants.

$$\hat{F}(\hat{x}) = \begin{pmatrix} \mathbf{A}x + \Lambda_1\chi + \mathbf{b} \\ \Lambda_2 x + \Lambda_3\chi + \beta \end{pmatrix}$$

So the mathematical expectation is

$$\begin{aligned} \mathbb{E}(\hat{F}(\hat{\mathbf{x}})) &= \begin{pmatrix} \mathbb{E}(\mathbf{Ax}) + \mathbb{E}(\Lambda_1\chi) + \mathbb{E}(\mathbf{b}) \\ \mathbb{E}(\Lambda_2\mathbf{x}) + \mathbb{E}(\Lambda_3\chi) + \mathbb{E}(\beta) \end{pmatrix} \\ &= \begin{pmatrix} \mathbb{E}(\mathbf{Ax}) + \mathbb{E}(\Lambda_1)\mathbb{E}(\chi) + \mathbb{E}(\mathbf{b}) \\ \mathbb{E}(\Lambda_2)\mathbf{x} + \mathbb{E}(\Lambda_3)\mathbb{E}(\chi) + \mathbb{E}(\beta) \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{Ax} + \mathbf{b} \\ 0 \end{pmatrix} \end{aligned}$$

The variance is

$$\begin{aligned} \mathbb{D}(\hat{F}(\hat{\mathbf{x}})) &= \begin{pmatrix} \mathbb{D}(\mathbf{Ax}) + \mathbb{D}(\Lambda_1\chi) + \mathbb{D}(\mathbf{b}) \\ \mathbb{D}(\Lambda_2\mathbf{x}) + \mathbb{D}(\Lambda_3\chi) + \mathbb{D}(\beta) \end{pmatrix} \\ &= \sigma^2 \begin{pmatrix} (D-H)\sigma^2 E_1 \\ (1 + |\mathbf{x}|^2 + (D-H)\sigma^2)E_2 \end{pmatrix} \end{aligned}$$

where $\mathbf{1}_{(*)}$ is all-one tensor with the same shape of tensor (*) ,$E_1 \in \mathbb{R}^H and E_2 \in \mathbb{R}^{D-H}$ are all-one column vectors, $\mathbf{x}_e^2$ is elements squared of $\mathbf{x}$, and $|\mathbf{x}|^2$ is the square of the norm of $\mathbf{x}$.

Above all, we proved that the padded dense layer maintains the same mathematical expectation with that of before-padding, and the variance is $O(\sigma^2)$. Therefore, if we pad random values with small variances, the output of the model will not change much. Especially, if the variance reduces to zero, the output will be unchanged.

When applying the method to the multi-head attention layer, we can pad random values $\theta \sim N(0, \sigma^2)$ to the parameters of queries, keys, and values. The output of the attention layer is $SoftMax((Q^T + O(\sigma^2)) \times (K + O(\sigma^2))) \times (V + O(\sigma^2))$, so if the variance σ^2 is zero, the output of attention layer remain unchanged after padding.

C Proof-2

We prove that Eq. 2 has no solution. According to the laws of linear equations, we only need to prove that the rank of the coefficient matrix is not equal to that of the augmented matrix. The augmented matrix $\hat{A}_0$ of Eq. 2 is

$$\hat{A}_0 = \begin{pmatrix} \hat{N}^1, & \mathbf{E}, & \mathbf{h}^1 \\ \hat{N}^2, & \mathbf{E}, & \mathbf{h}^2 \\ \vdots & \vdots & \vdots \\ \hat{N}^{|Vocab|}, & \mathbf{E}, & \mathbf{h}^{|Vocab|} \end{pmatrix}$$

Let the last $|Vocab| - 1$ row blocks subtract the first row block, the augmented matrix $\hat{A}_0$ changes to

$$\hat{A}_1 = \begin{pmatrix} \hat{N}^1, & \mathbf{E}, & \mathbf{h}^1 \\ \hat{N}^2 - \hat{N}^1, & \mathbf{0}, & \mathbf{h}^2 - \mathbf{h}^1 \\ \vdots & \vdots & \vdots \\ \hat{N}^{|Vocab|} - \hat{N}^1, & \mathbf{0}, & \mathbf{h}^{|Vocab|} - \mathbf{h}^1 \end{pmatrix}$$

Let the first column block subtract the product of $\hat{N}^1$ and the second column block, and the third column block subtract the product of $\mathbf{h}^1$ and the second column block, the augmented matrix $\hat{A}_1$ changes to

$$\hat{A}_2 = \begin{pmatrix} \mathbf{0}, & \mathbf{E}, & \mathbf{0} \\ \hat{N}^2 - \hat{N}^1, & \mathbf{0}, & \mathbf{h}^2 - \mathbf{h}^1 \\ \vdots & \vdots & \vdots \\ \hat{N}^{|Vocab|} - \hat{N}^1, & \mathbf{0}, & \mathbf{h}^{|Vocab|} - \mathbf{h}^1 \end{pmatrix}$$

After going through a similar matrix transformation, the coefficient matrix A_0 changes to

$$A_2 = \begin{pmatrix} \mathbf{0}, & \mathbf{E} \\ \hat{N}^2 - \hat{N}^1, & \mathbf{0} \\ \vdots & \vdots \\ \hat{N}^{|Vocab|} - \hat{N}^1, & \mathbf{0} \end{pmatrix}$$

We define $\hat{N}^*$ and h^* as follows:

$$\hat{N}^* = \begin{pmatrix} \hat{N}^2 - \hat{N}^1 \\ \hat{N}^3 - \hat{N}^1 \\ \vdots \\ \hat{N}^{|V|} - \hat{N}^1 \end{pmatrix}, h^* = \begin{pmatrix} \mathbf{h}^2 - \mathbf{h}^1 \\ \mathbf{h}^3 - \mathbf{h}^1 \\ \vdots \\ \mathbf{h}^{|V|} - \mathbf{h}^1 \end{pmatrix}$$

According to the calculation of $\hat{N}^*$ and h^*, they are not linearly dependent. Obviously, it is equivalent to that the matrices $\hat{A}_2$ and A_2 have different ranks. So Eq. 2 has no solution.

D Figures

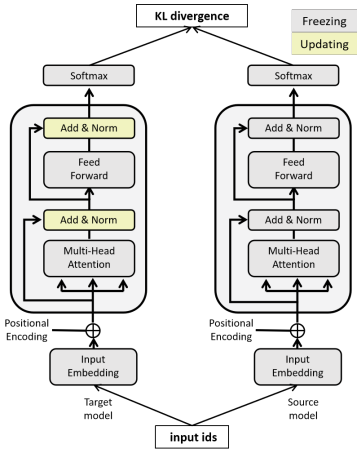

Figure 5: Overview of the distillation-based method