공학석사 학위논문

Reducing the Cost of Training a Transformer Model by Using a Trained Model

이미 학습된 모델의 활용을 통한 새로운 트랜스포머 모델의 학습 비용 감소

2022년 8월

서울대학교 대학원

컴퓨터공학부

한민희

Reducing the Cost of Training a Transformer Model by Using a Trained Model

이미 학습된 모델의 활용을 통한 새로운 트랜스포머 모델의 학습 비용 감소

지도교수 이재진

이 논문을 공학석사 학위논문으로 제출함
2022년 5월

서울대학교 대학원
컴퓨터공학부
한민희

한민희의 석사 학위논문을 인준함
2022년 5월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

1. Abstract

The cost of training a new language model is higher than ever, and it continues to increase. To mitigate the issue, this paper proposes reusing a trained model to reduce the cost of training a larger model. By using the methods used in Knowledge Distillation(KD), the knowledge of the present trained model can be transferred to the new model, even when the new model is larger than the trained model. This is done by 1) copying the weights and 2) logits matching. The former can be used for models of the same dimensions while the second can be used regardless of the dimensions, though it requires more computations than the former. In the experiments with the GPT-like models, it is shown that reusing a relatively small trained model reduced the training time of a relatively larger model.

Keyword : Deep Learning, NLP, Knowledge Distillation, Transformer Model, Model Training, Model Reuse
Student Number : 2020-25068

Table of Contents

Chapter 1. Introduction

1.1. Introduction

Recently published Natural Language Processing (NLP) models are showing lots of improvements, and their improvements are based on the scaling of the model. [8] For example, the original GPT model had 124 million parameters and the GPT-2 model had 1.5 billion parameters, while the most recent GPT-3 model had 175 billion parameters. [2, 3, 4] The trend of increase in the model size is expected to continue further. However, the increase in the model size is entailed by the increase in the training cost and time. It is known that the cost for training BERT was $ 6,912. [11] Though the research team has not officially disclosed the training cost, it is estimated to be up to $ 4.6 million, with a Tesla V100 cloud instance [18]

There have been many researches to reduce the training costs of these models, or to accelerate the training speed. [11, 12, 13] Though these approaches proved effective, they don't utilize the present models, which is an act of wasting these models. It would be beneficial if we can utilize a present model to reduce the training cost of a new model we're trying to train. Though the new model can be larger than the previous model, the previous model is often also a result of a large amount of training. Thus, even the present model that is smaller than the new model can contain much more information than the new model at the early stage of the training. Thus, if we can transfer the knowledge of the present model, which is already trained, to the large and new model, we can expect the overall training time to be reduced.

In transferring the knowledge of the previous model, we'll use the techniques from Knowledge Distillation (KD). Previous work regarding KD usually tries to compress a larger teacher model's information into a smaller student model. However, the same techniques can be used in the opposite case, where the teacher is smaller than the students. We used two techniques from KD, 1) weight copying and 2) matching the logits to transfer the knowledge from the teacher model to the student model. [6]

For the paper, we used smaller versions of GPT-2 models that have the same architecture with the original GPT-2 model but different

dimensions and layer numbers.

1.2. Related Work

Since the publication of the attention-based transformer model, [1] it became a dominant architecture in the area of Natural Language Processing (NLP). [2, 3, 4, 9, 14] Across different models, their model capacities are known to be determined by the model sizes, that are determined by the number of parameters in the model, the number of layers, etc.. As a consequence, recent NLP models are becoming larger then ever, and they actually constitute the largest models in the whole DNN domain. [8] This trends in the enlargements of NLP models are entailed by the huge increase in the training costs and time. [4, 8, 11]
Other than the traditional approaches like using mixed-precision and a large batch, there have been recent approaches targeted specifically at optimizing the training of language models. Megatron-LM alleviates the large memory constraints by using model parallelism. [13] DeepSpeed aims at reducing the GPU memory usage and effectively utilizing the distributed resources. [12] By using the both techniques, a language model with 530 billion parameters could be made. [12, 13, 14]
However, there was no attempt to reuse the already trained models to reduce the cost of training a new model. For our approach can be used together with other optimizations, it can still be beneficial even if the improvement is relatively less dynamic.

We use Knowledge Distillation(KD) techniques to use the knowledge in the present trained model. KD was originally used to compress a large model into a smaller one, or to compress an ensemble of models into a single model. [6] There have been various applications of the KD in different domains, and the application for language models are relatively recent. [17] DistilBERT is one example of such cases, and using the same method, DistilGPT-2 was also released. [16]
Unlike the previous work in KD, we're using the method not to compress the model, but to transfer the information, which actually is an intermediate goal for most KD approaches.

Chapter 2. Design and Implementation

For transformer models, the main factors that determine the model size is the embedding size and the number of the transformer layers. A transformer layer's size(i.e., the number of parameters in the layer) is proportional to the square of the embedding size. Often, for a same architecture, there are a few models with different model sizes, which was the case for BERT and GPT models. [3, 9] We assume a situation where there is a model which already finished its training, and there are two larger models to be trained from scratch, one of which has the same dimensions as the trained model but twice the number of layers. Another model has larger dimensions and more layers compared to the original model, but not as much as twice the layers. Those two models are set to have the same number of parameters in the transformer layers: twice that of the original model. We transfer the knowledge of the original, trained model to these two new models before training them. By initializing the new model's parameters using the original model, we can achieve the same accuracy with less training compared to the situation where we don't use the original model in the training of the new model.

We first randomly initialize the new model that we are trying to train. Next, we transfer the knowledge of the trained model to a part of the new model. Then, the new model with the transferred knowledge is trained with the usual training method with the dataset.

For transferring the knowledge of the teacher model to the student model, we use two methods. The first one can be used for the models with the same dimensions but different layer numbers. The second one can be used regardless of the dimensions, but requires a little bit more computations during the training period.

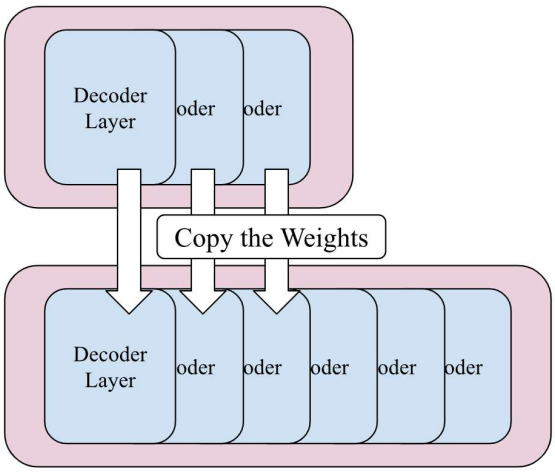

Fig 1. Copying the Weights

Method 1: Copying the Weights

It is known that the output "evolves" going through each transformer layers, becoming closer to the "correct" output. [7] Let's say that there are two trained models with different number of layers. One has 4 layers while the other has 8 layers. Then, the output of the 4-layer model will be similar to that of the output of the 4th layer in the 8-layer model and their parameter values will also be similar. This is why simply copying the weights from the teacher model to the student model can transfer the knowledge.

When the student and the teacher models have the same dimensions but different number of transformer layers, a transformer layer in each model has exactly the same size and the same number of parameters. We can directly copy the weights of the teacher model layers to the student model layers. The cost of copying the weights is trivial and can be ignored.

However, there is a possibility that the this method can result in the trap of 'local minima'. Thus, we train the models until they converge to see if they can reach the same loss even after large amounts of training.

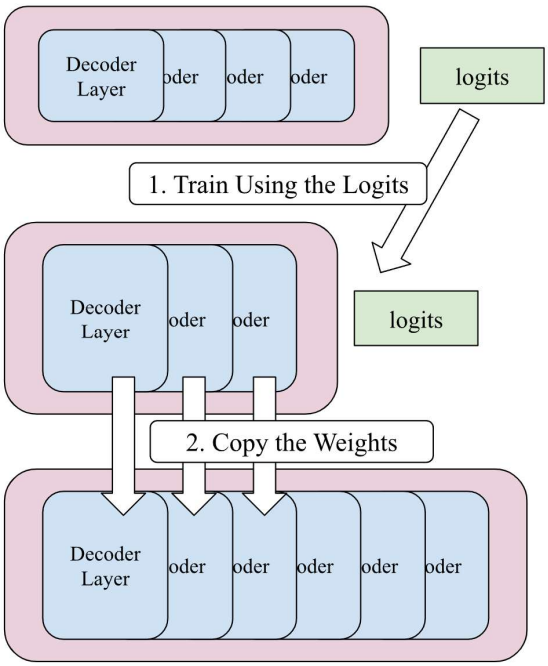

Fig 2. Matching the Logits

Method 2: Matching the Logits

Instead of directly copying the weights, we can use the output logits from the teacher model. Though their layer dimensions might vary, their outputs have the same dimensions and structures. For a GPT-like model trained to predict the next vocabularies, the output logits represents the possibility for each vocabulary, and this possibility distribution contains much more information than a hot label that denotes a single vocabulary. The student model can train on the logits from the teacher model. That is, we can match logits from two models. Though this method requires inferences from the teacher model, the volume of additional computations isn't high as it is possible for the information to be transferred with much fewer training steps compared to the usual training on hot labels. It is shown in the experiments that only 0.4 epochs of training worked.

For the new model is larger than the trained model, instead of directly matching the large new model's logits with the original model, we make a intermediate model that has the same dimensions as the new model, but fewer layers. The intermediate model's total parameters should be less than or equal to the teacher model's parameters.

After setting the intermediate model that way, we transfer the

knowledge of the teacher model to the intermediate model by training the intermediate model on the logits from the teacher model. After the intermediate training, the weights of the intermediate model is transferred to the large model by copying the weights.

For the intermediate training, training settings similar to DistilBERT was used. the cross entropy loss is used:

$$L_{ce} = \sum t_i \times log(s_i)$$

Eqn 1. cross entropy loss

t_i denotes the teacher's logits for the *ith* input, and s_i denotes the student's logits for the same input.

$$p_i = \frac{\exp(z_i / T)}{\sum_j \exp(z_j / T)}$$

Eqn 2. softmax-temperature

Also, a softmax-temperature is used so that the logits are softened. z_i denotes the model score for the class *i*, and the numerator denotes the sum of the scores for the all classes. Various temperatures were tried, and the temperature value (*T*) of 4 yielded the best results among many choices. [5]

We used the PyTorch based transformer library from Huggingface, and modified it to implement our design. [15]

Chapter 3. Experiments

3.1. Setup

For the experiments, we used smaller versions of GPT-2 models that have the same architecture with the original GPT-2 model but different model sizes.

Model	# of Parameters	# of Layers	Dimension
4-128	6.7M	4	128
8-128	8.4M	8	128
6-144	8.7M	6	144

Table 1. The models used in the experiments

We set three models for the experiment. 4-128 is the base "teacher" model. 8-128 is set such that it has twice the decoder layers of the 4-128. 6-144 has almost the same number of parameters in the decoder layers as 8-128. For models with larger dimensions and number of layers, the total number of parameters will more proportionate to the number of layers, but for our experiments, as the models used are relatively smaller, a large portion of parameters are for the layers other than decoders. For the training dataset, we randomly sampled 8GB of OpenWebText2 as the train set and 2GB as the test set. The training epochs were set to 5, 6 and 6 respectively for 4-128, 8-128 and 6-144. As recent work rarely gives information on the training epochs, it is difficult to choose the right epoch for the training. [10] We set it this way so that

$$\frac{\text{dataset size} \times \text{epochs}}{\text{number of parameters}}$$

to be similar to that of BERT. [9] 4 NVIDIA Tesla V100 PCIe 32 GB cards were used in the experiments. The batch size of 16 was used for all experiments, except for the case of training the intermediate model of 3-144. The training was done for the next sentence generation task with cross entropy loss as the loss computation.

We first trained each model to 5, 6 and 6 epochs respectively. Then, using the trained 4-128 model, we transferred the knowledge of the 4-128 model to newly initialized 8-128 and 6-144 model. We denotes the transferred version as 8-128' and 6-144' to distinguish them with ones without the transfer. After the transfer, we trained the two models with

the transferred knowledge in them. The training was stopped when the training loss reached the training loss of each model when training them for 6 epochs without the transfer.

3.2. Preliminary Experiment

First, to see how we should copy the weights, three ways of copying the weights were tested. The first was copying the teacher's weights to the front, the second was copying the weights in between the student layers (interpolating), and the last was copying the teacher's weights to the back. Below 3 figures show the three ways.

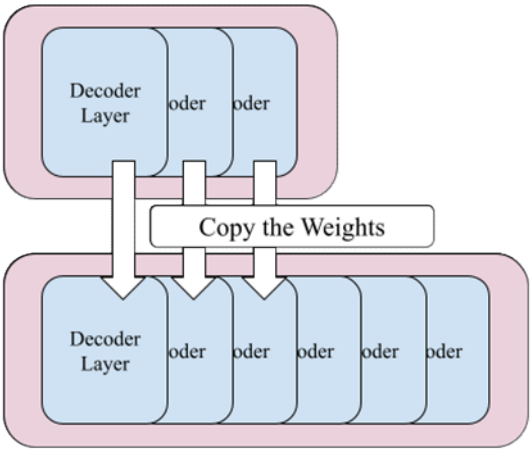

Fig 3. Copying to the front

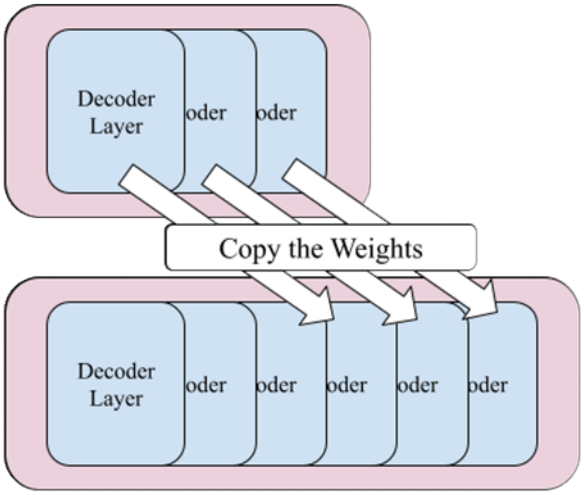

Fig 5. Copying to the back

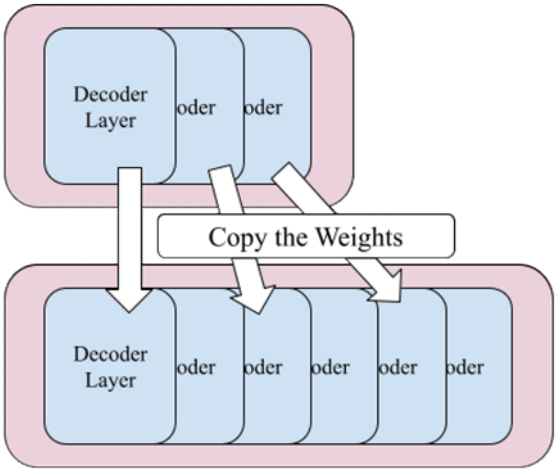

Fig 4. Interpolating

After initializing each model, we trained each model. The result shows that the copying to the front was the best among three, and copying the back was the worst. As it showed the fastest convergence among three, we copied the weights to the front of the new model in the succeding

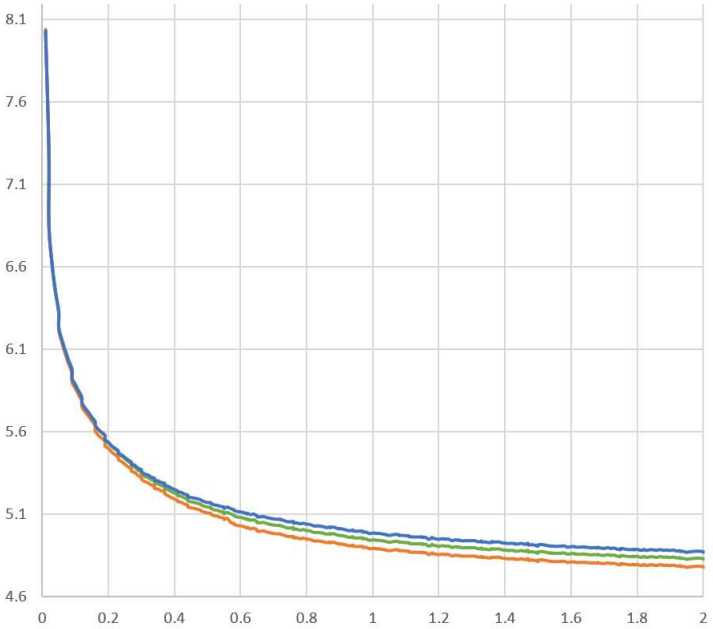

Fig 5. The training loss of 8-128' with three methods. orange: copying to the front, blue: copying to the back, green: interpolating

3.3. Results

Model	Epochs	Time (Hours)	Final Loss
4-128	5	24	4.919
8-128	6	55	4.68
6-144	6	58	4.71

Table 2. The training of each model without the transfer

1) 8-128 and 8-128'

Model	Epochs	Time (Hours)	Final Loss
8-128	6	55	4.68
8-128'	5.79	53	4.68

Table 3. The training of 8-128 and 8-128'

The weights from 4-128 is copied to a newly initialized 8-128', and training using the dataset was done for the model. The 8-128' model reached the loss of 4.68 in 5.79 epochs, thus saving about 3.5% of the training epochs and 2 hours of training compared to training the original 8-128 model without using the trained 4-128 model. 8-128' showed slightly lower loss overall.

Fig 6. The training loss of 8-128 and 8-128'

3) 6-144 and 6-144'

Model	Epochs	Time (Hours)	Final Loss
3-144		10	
6-144	6	55	3.87
8-128'	4.71	34	3.87

Table 4. The training of 6-144 and 6-144'

Fig 7. The training loss of 8-128 and 8-128'

For the transfer from the 4-128 to the intermediate 3-144, it took about 10 hours for the 0.4 epochs of training. As the loss started at 5.08 and reached below 0.4 in 0.4 epochs, no more training was needed for the transfer. It was done running the inference on 4-128 and training on 3-144 at the same time. The time for this can be further reduced when the inference on 4-128 and training on 3-144 with the inference results are separated, as concurrently running the two models restricted the use of large batch sizes.

After 3-144 finished its training, the weights were copied to a newly initialized 6-144', and the usual training was done. 6-144' reached the loss of 4.71 in just 3.87 epochs, reducing . If we add the time for training the intermediate model, the total saved time of training will be 21 hours of training, which means 18.9% reduction in the total training time.

Chapter 4. Conclusion

4.1. Conclusion

The experiments results showed that it is beneficial to transfer the knowledge of present, trained models to the new models, thus resulting thm. Though the amount of the benefits might vary, as transferring the knowledge of the teacher model is trivial, it can be regarded as an improvement. Further optimizations like tuning the hyper parameters or using different intermediate models could further improve the results.

4.2. Discussions

Most importantly, as we used a mini-sized model for the experiments, it can be different when using a full-sized model. However, as the SOTA models require huge amounts of computations, [14] it is difficult to actually train the full-sized models. Still, as the models used in our experiments share the same architectural characteristics with other SOTA models, we expect the method to work also for the larger models, though their effectiveness can vary. Moreover, it is possible that the same methods perform better for the real-sized models than our experiments models, as it is possible that the smallness of the teacher model might restricted its model capacity.

Though the main point of the paper was proved by the experiments, there is still room regarding the explainability aspect. Especially because the results were much better with 6-144' model compared to 8-128', further explanations are needed. This can be done by comparing the weight values and hidden states from each model and at each stage of the training.

It is possible to transfer the knowledge to a model with a different architecture. We did a preliminary research transferring a BERT model's knowledge to a GPT-like model we used in the experiments. Fine-tuning to make the same outputs and index conversion of different tokenizers were required. It yielded similar results, so it is expected that the techniques from this paper can be generally used among different models.

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. Attention Is All You Need, arXiv:1706.03762, 2017.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training, 2018.

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

[4] 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. Language Models are Few-Shot Learners, arXiv:2005.14165, 2020.

[5] Victor Sanh, Lysandre Debut, Julien Chaumond and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv:1910.01108, 2019.

[6] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. Distilling the Knowledge in a Neural Network, arXiv:1503.02531, 2015.

[7] Jay Alammar. Ecco: An Open Source Library for the Explainability of Transformer Language Models. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 249–257, Online, 2021.

[8] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu and Dario Amodei. Scaling Laws for Neural Language Models, arXiv:2001.0836, 2020.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805, 2018.

[10] Aran Komatsuzaki. One Epoch Is All You Need, arXiv:1906.06669, 2019.

[11] Or Sharir, Barak Peleg and Yoav Shoham. The Cost of Training NLP Models: A Concise Overview, arXiv:2004.08900, 2020.
[12] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase and Yuxiong He. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters, KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3505-3506, 2020.
[13] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, arXiv:1909.08053, 2019.
[14] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, Bryan Catanzaro. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model, arXiv:2201.11990, 2022.
[15] 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. HuggingFace's Transformers: State-of-the-art Natural Language Processing, arXiv:1910.03771, 2020.
[16] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao and Kaisheng Ma. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation, arXiv:1905.08094, 2019.
[17] Jianping Gou, Baosheng Yu, Stephen John Maybank and Dacheng Tao. Knowledge Distillation: A Survey, arXiv:2006.0552, 2021.
[18] Lambda Labs, OpenAI's GPT-3 Language Model: A Technical Overview, https://lambdalabs.com/blog/demystifying-gpt-3/, June 03, 2020, accessed on May 06. 2022.

국문 초록

새로운 자연어 처리 모델을 학습하는 비용은 어느 때보다도 높으며, 계속해서 증가하고 있다. 이런 문제를 해결하기 위해 이 논문은 이미 학습된 모델을 재활용하여 더 큰 모델을 학습하는 비용을 줄이는 방안을 제시한다. 지식 증류(Knowledge Distillation)의 기법들을 이용해 이미 학습된 모델의 지식을 새로운 모델로 이전하는 것이 가능한데, 이는 새로운 모델이 학습된 모델보다 더 큰 경우에도 그러하다. 이것은 1) 그 가중치(weights)를 복사하는 것과 2) 두 모델의 로짓(logit)을 같게 만드는, 두 가지 방법으로 가능하다. 전자는 두 모델의 차원(dimension)이 동일한 경우에만 사용 가능하지만, 후자는 그렇지 않은 경우에도 사용할 수 있다. GPT2와 비슷한 모델을 이용한 실험에서, 두 가지 방법은 학습 시간을 각각 3.5%, 18.9% 단축하였다. 이를 통해 비교적 작은, 학습된 모델을 재사용해 큰 모델의 학습 시간을 단축할 수 있음을 보였다.

주요어 : 딥러닝, 자연어처리, 지식 증류, 트랜스포머 모델, 모델 학습, 모델 재활용
학　번 : 2020-25068