공학석사학위논문

파라미터 효율적인 트랜스포머 지식 증류

Parameter-Efficient Knowledge Distillation on Transformer

2023년 2월

서울대학교 대학원
컴퓨터공학부
전효진

파라미터 효율적인 트랜스포머 지식 증류

Parameter-Efficient Knowledge Distillation on Transformer

지도교수 강 유

이 논문을 공학석사 학위논문으로 제출함

2022년 11월

서울대학교 대학원

컴퓨터공학부

전효진

전효진의 석사 학위논문을 인준함

2022년 12월

위 원 장 문봉기 (인)

부 위 원 장 강 유 (인)

위 원 이상구 (인)

Abstract

Parameter-Efficient Knowledge Distillation on Transformer

Hyojin Jeon
Department of Computer Science & Engineering
The Graduate School
Seoul National University

How can we obtain a small and computationally efficient transformer model, maintaining the performance of a large model? Transformers have shown significant performance in recent years. However, their large model size, expensive computation cost, and long inference time prohibit them to be deployed on resource-restricted devices. Existing transformer compression methods have mainly focused on only reducing an encoder although a decoder takes up most of their long inference time. In this paper, we propose PET (Parameter-Efficient Knowledge Distillation on Transformer), an efficient transformer compression method reducing the size of both the encoder and decoder. PET improves the knowledge distillation of the Transformer, designing an efficient compressed structure of both the encoder and decoder and enhancing the performance of the small model through an efficient pre-training task. Experiments show that PET succeeds in obtaining memory and time efficiencies by 81.20% and 45.20%, respectively, minimizing accuracy drop below 1%p. It outperforms

the competitors for various datasets in machine translation tasks.

Keywords : Model Compression, Transformer, Knowledge Distillation

Student Number : 2021-24350

Contents

List of Figures

List of Tables

Chapter 1

Introduction

How can we compress a large Transformer model into a smaller model, maintaining the original performance? Transformers [1] have achieved state-of-the-art performance in the field of Natural Language Processing (NLP). They have shown their potential to be utilized in a variety of practical applications such as language modeling, translation, and question-and-answering. These days, those applications run in various environments including mobile devices. Most mobile devices have restricted resources. They have limited memory and poor computation abilities. Also, low energy consumption and fast inference speed are important in the real world. On the contrary, enlarging a language model and improving its performance has been the main trend in NLP for the last few years. More and more large models have been introduced and achieved top performance. Latest models, such as GPT-3 [2] and Megatron-Turing NLG [3] have more than hundreds of billions of parameters. Despite their remarkable performance, their excessive memory usage, energy consumption, and long inference time prohibit them to be deployed to resource-limited devices or real-world applications. Therefore, an efficient transformer compression method is required.

Recently, several transformer compression methods have been proposed. However, most studies have mainly focused on compressing the encoder, including BERT [4] compression. (e.g., DistilBERT [5], TinyBERT [6], MobileBERT [7], Pea-KD [8] and SensiMix [9]) Translation, speech recognition, and speech translations are based on both transformer encoder and decoder. Moreover, the decoder mainly contributes to

a model's long inference time and takes more than half of the entire model size. it is not easy to directly apply existing BERT compression methods to compressing the decoder since the encoder and decoder have different architectures and properties. In most cases, significant accuracy loss is inevitable. Therefore, we need an efficient and flexible transformer compression method that shrinks the size of the encoder and decoder simultaneously.

In this thesis, we propose PET (Parameter-Efficient Knowledge Distillation on Transformer), an accurate transformer compression method that reduces the model size, computational cost, and inference time while conserving the accuracy of the original model. Especially, PET compresses the size of both the encoder and decoder simultaneously. PET improves a knowledge distillation on transformer optimizing the entire process: a student model design, initialization, and training. PET finds efficient pairs of modules for the encoder and decoder, respectively, and shares them to compress the model size minimizing an accuracy loss. PET succeeds in pre-training a student model more efficiently and enhances the model's accuracy. Moreover, PET shows a method to improve the model accuracy in training by optimizing the student model's shared layers with the most efficient parameters.

Our main contributions are as follows:

- **Algorithm.** We propose PET, an efficient transformer compression method that reduces the size of both encoder and decoder maintaining the original performance. It optimizes the entire process of knowledge distillation: a student model construction, initialization, and training.
- **Generality.** The proposed PET does not require additional model structure or expensive training to be applied to other tasks. It can be easily used in various transformer-based models.

Table 1: Table of the symbols.

Symbol	Description
x	Vector
x_i	i-th element of x
L_i	i-th Layer
Q_i^{self}, K_i^{self},V_i^{self}	Query, key, and value matrices of the self-attention sub-layer in the i-th layer
Q_i^{enc}, K_i^{enc}, V_i^{enc}	Query, key, and value matrices of the encoder-decoder-attention sub-layer in the i-th layer
f_t, f_s	teacher and student model

- **Performance.** The proposed PET achieves memory, computation, and inference time improvements by up to 81.20%, 80.16%, and 45.20%, while minimizing accuracy drop under 1%p. Extensive experiments on multiple real-world language datasets show that PET consistently achieves superior performance than other baselines.

The rest of the thesis is organized as follows. In Chapter 2, we explain the transformer and multi-head attention mechanism and review existing transformer compression methods based on knowledge distillation. In Chapter 3, we describe the proposed algorithm PET in detail. After showing experimental results in Chapter 4, we conclude in Chapter 5. The symbols frequently used in this thesis are summarized in Table 1.

Chapter 2

Background and Related Works

In this chapter, we first explain the architecture of the transformer and its output structure. We describe how encoder and decoder are different in terms of their architecture and the input information they process (Section 2.1). We then review the transformer compression algorithms based on knowledge distillation. We classify them into three categories according to their compression targets: encoder-only, decoder-only, and encoder-decoder (Section 2.2).

2.1 Transformers

Transformer [1] was introduced for sequence-to-sequence tasks such as summarization, and translation. It maps source and target sequences, like translating a source to a target language. A transformer consists of an encoder and a decoder. The encoder and decoder are both a stack of identical layers where several sub-layers exist, including multi-head attention and fully-connected feed-forward layer. You can see the architecture of the transformer in Figure 1.

2.1.1 The Architecture of Transformer

The encoder takes an input source sequence, encodes its hidden representation, and feeds it to a decoder. Multi-head attention in each encoder layer, called self-attention, uses query, key, and value generated from previous encoder states. The decoder receives the encoded source information from the encoder and target information from

previous decoder states to generate a target sequence. In other words, the decoder processes two types of data, source, and target, while the encoder processes just a single type of source data. There are two types of multi-head attention in each decoder layer. The first multi-head attention is self-attention. Like the same one in the encoder, it handles only a single type of information from the target sequence. It takes the query, key, and value generated from the previous decoder states. The second multi-head attention is encoder-decoder attention. As its name says, it attends to not only the states of the previous decoder but also the output of the encoder. Its key and value are from the encoder, whereas the query is from the decoder. As a result, the decoder processes two types of dissimilar information from a source and target sequence.

2.1.2 The Output Structure of Transformer

The output structure of the transformer is illustrated in Figure 2. The final output of the transformer is a three-dimensional tensor. Each dimension represents the number of sequences in a batch, the number of tokens in each sentence, and the number of the target vocabulary. The model predicts the probability for each token over the entire target vocabulary. In Figure 2, p_i is the prediction probability of the token i. The vocabulary of the largest probability is regarded as the predicted answer to the token i and compared with the target label y_i.

2.1.3 Multi-head Attention

Multi-Head Attention (MHA) implements h heads where each computes in parallel. MHA jointly attends to information from different subspaces with those heads. A detailed computation of MHA is as follows:

Given query Q, key K, value V,

$$\begin{aligned} \texttt{MultiHead}(Q,K,V) &= \texttt{concat}(\texttt{head}_1,\ldots,\texttt{head}_h) \\ \texttt{Attention}(QW_i^Q, KW_i^K, VW_i^V) &= \texttt{softmax}(\frac{(QW_i^Q)(KW_i^K)^T}{\sqrt{d_k}})(VW_i^V) \end{aligned} \tag{2.1}$$

where $\texttt{head}_i = \texttt{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, h is the number of the attention heads, d_k is a scaling factor, and W_i is the projection matrices.

2.2 Knowledge Distillation on Transformers

Knowledge distillation (KD) [10] transfers knowledge of a large and well-trained teacher model to a smaller student model to enhance the performance of the smaller model. Utilizing the knowledge of a trained model to improve a compressed model is a widely known and efficient approach for the compression of various models as well as Transformers, such as Convolution Neural Networks (e.g., FALCON [11]) and Graph Convolution Networks (e.g., MustaD [12]). The overall process of KD is as follows: given a large and well-trained teacher model, first, we construct a small student model and initialize it. Then, we train the student model using the predictions of the teacher model in addition to the true labels for the target. In this section, we review transformer compression methods based on knowledge distillation. We classify them according to their compression targets, whether they compress only an encoder or decoder, or both encoder and decoder. Also, we focus on how each method constructs and initializes the smaller student model.

2.2.1 Knowledge Distillation on Transformer Encoders

Knowledge Distillation on BERT falls into this category. Patient KD [13] extracts knowledge from the intermediate layers as well as the final prediction. It initializes the student model by taking some layers of the teacher. DistillBERT [5] introduces a triple loss combining task, distillation, and cosine-distance losses. It initializes the student model by taking one of the two layers of the teacher. TinyBERT [6] distills transformer (attention matrices and hidden states), embedding and prediction layer at two-stage learning: general and task-specific distillation. They initialize a student model for task-specific distillation with another student model trained at task-general distillation. MobileBERT [7] requires a specially designed teacher model equipped with an inverted-bottleneck structure to distill the knowledge to train the student model. Pea-KD [8] proposes a layer-sharing and shuffling query and key matrices of the encoder to enlarge an insufficient capacity of the student model. It generates four new labels for given binary classification tasks using the teacher's predictions. It initializes the student model by pre-training with those new labels to help them learn the teacher's high-level knowledge. During pre-training, the student model of Pea-KD uses a classification layer with an output size of four to classify those four generated labels. In the KD process, the main learning objective, the student model has to classify binary classes. Thus, it changes its pre-trained classification layer with a smaller output size of two. Pea-KD does not use the entire pre-trained parameters and replaces the last classification layers with non-pre-trained ones. Pea-KD has limitations in that it pre-trains extra parameters to be removed and does not initialize the exact classification layer used in the KD process. How we address this challenge is described in Chapter 3 (Section 3.2).

2.2.2 Knowledge Distillation on Transformer Decoders

Knowledge Distillation on GPT falls into this category. However, as mentioned in Chapter 1, much fewer works have been proposed compared to knowledge distillation on encoders. KnGPT2 [14] compresses the embedding and transformer layers of GPT-2, using Kronecker decomposition. It uses KD to compensate for the performance drop of the compressed model. However, a large computation cost remains during decomposition. CAN [15] compresses self-attention and cross-attention using simplified matrix multiplication. Although it addresses the transformer neural machine translation model consisting of an encoder and decoder, it only addresses decoder compression. Moreover, it makes encoder deeper (from 6 to 12 layers) to compensate for the accuracy loss caused by reducing the size of the decoder. Therefore, there remains room for compressing the entire model.

2.2.3 Knowledge Distillation on Transformer Encoders and Decoders

Weight Distillation [16] transfers knowledge in parameters of the teacher model through a parameter generator. It generates parameters of the student by weight grouping and projection. It consists of two phases: generating parameters of the student model and training the generated student model. It requires training both parameter generator and student model sequentially. Thus, about twice the computational cost for the training is inevitable. PET does not use the additional parameters for generating a smaller model and maximizes its training efficiency by training only the exact parameters of the target compression model. Once a target compression model is generated, PET does not transform its structure or replaces its components.

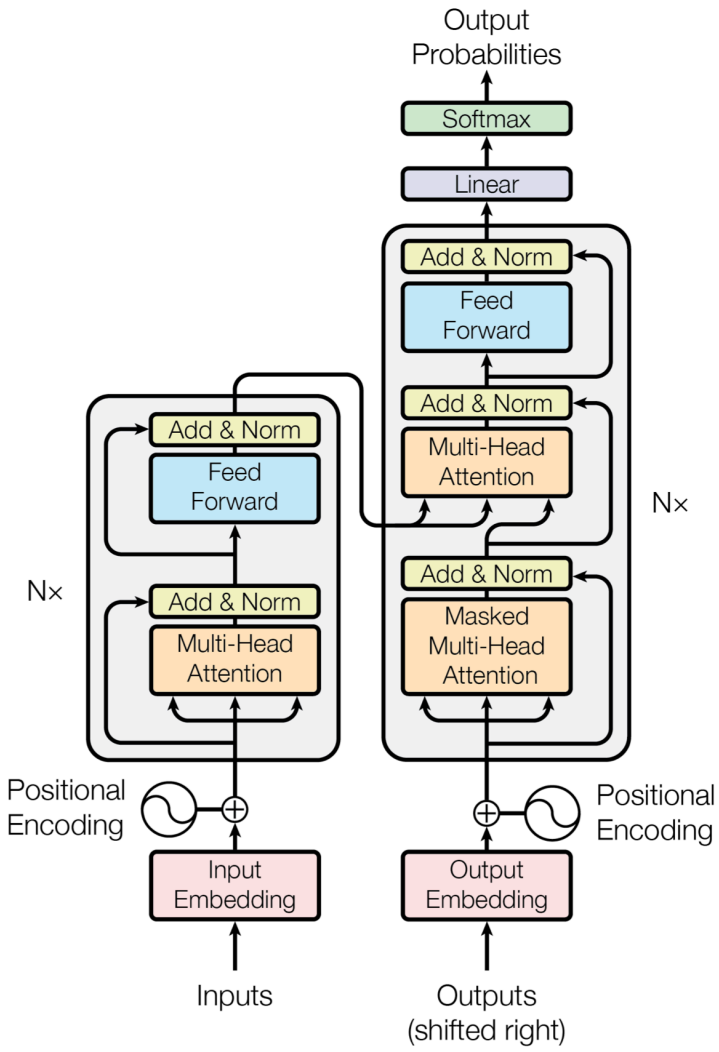

Figure 1: The model architecture of the transformer [1]. Transformer consists of N layers of encoder and decoder. The encoder (left) takes a input source sequence and feeds the output to a decoder. The decoder (right) takes an output from the encoder as well as states of the previous decoder layer and returns the final output of the entire model.

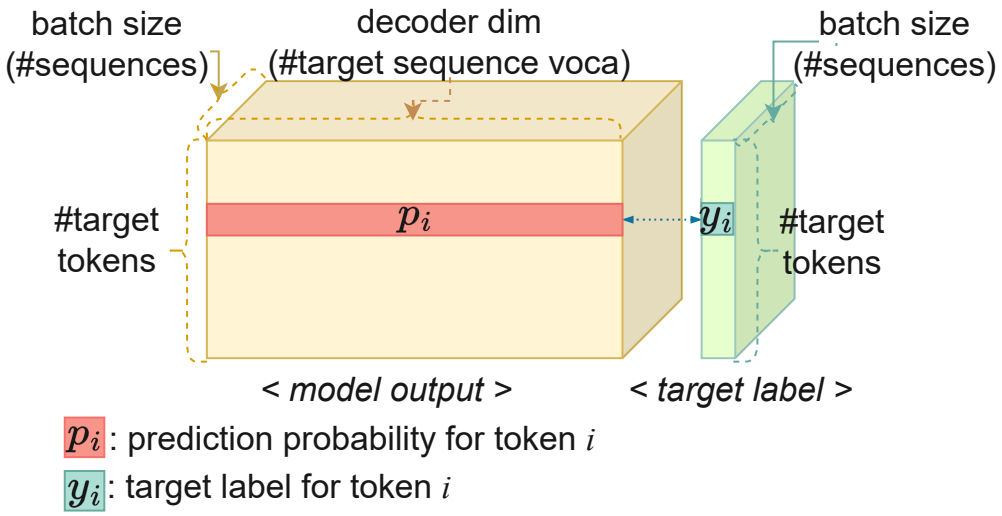

Figure 2: The model output structure of the transformer. The final output of the transformer is a tensor of three dimensions. Each dimension represents the number of sequences in a batch, the number of tokens for each sentence, and the number of target vocabulary. p_i is the prediction probability of the model for token i. y_i is the target label for the token i.

Chapter 3

Proposed Method

In this chapter, we propose PET, an efficient model compression method improving the knowledge distillation of the transformer. Given a large and well-trained transformer model and NLP task, PET returns a compact transformer model which is small, computationally efficient, and fast while preserving the performance of the large model. To use each parameter more efficiently, we propose a compressed model architecture, a model initialization method before training, and an attention head sampling method during training. In other words, we succeed in optimizing every step of the KD.

We attend to the following challenges to improve the KD of the transformer:

1. How can we compress both the encoder and decoder efficiently?
2. How can we initialize a student model that well adapts to a challenging task?
3. How can we improve an accuracy of the model maintaining its reduced size?

Our main ideas to solve those challenges are as follows:

1. *Find replaceable pairs of modules to be shared*

   To construct a student model with fewer parameters maintaining the accuracy of the given large model, we find *replaceable pairs* of the modules in each encoder and decoder, and share their parameters of them. *Replaceable pair* indicates that the paired modules are robust to parameter sharing. The exploring targets of PET are layers, query, key, and value matrices (Sections 3.1).

2. *Warm-up the student with a simplified task without a model transformation*

   To efficiently initialize a student model to make it well adapts to the given complicated task with fewer parameters, PET pre-trains the student with an easier task before the KD process. To pre-train the model using the simplified task with fewer classes than those of the original one, we propose a training method that does not require modifying the model structure and improves the pre-training effectiveness (Sections 3.2).

3. *Sample different attention heads by layers from the extended head pool*

   To further optimize the compressed model, PET enlarges the size of the attention layers with more attention heads and samples efficient heads by layers (Section 3.3).

The overall process of PET is as follows: Given a well-trained teacher model and NLP tasks, first, we construct a smaller student model using parameter-sharing according to replaceable pair. Once the parameter-shared model is constructed, we initialize the student model by pre-training with simplified task. Then, we train the well-initialized model with the original tasks. We optionally apply layer-wise attention head sampling for more accuracy gain. We first describe a small student model construction method of PET in Section 3.1. Then, we describe the pre-training process of PET in Sections 3.2. Finally, in Section 3.3, we describe the training process of PET to obtain a more efficient compressed model.

3.1 Finding Replaceable Pairs in Encoder and Decoder

How can we construct an efficient small student model compressing both the transformer encoder and decoder? The motivation of replaceable pair is that there will be a fixed parameter-pair that will not degrade accuracy when shared. We regard this pair as replaceable pair, a module pair robust to parameter sharing, and explore it in each encoder and decoder considering the different architectures and inputs of them. We find pairs for layers and query, key, and value matrices in attention sub-layers. In the case of the four-layered model, we find that a set of the layer pairs $\{(L_1, L_3), (L_2, L_4)\}$ is the best compared to others such as $\{(L_1, L_2), (L_3, L_4)\}$, $\{(L_2, L_3, L_4)\}$, etc. Thus, we share L_1-L_3 and L_2-L_4 in each encoder and decoder. For matrices, we find different pairs of query, key, and value matrices in the encoder and decoder, respectively. First, in the encoder, we make pairs with query and key matrices in each layer replaceable pairs. In Figure 3, $Q_3^{\text{self}}, K_3^{\text{self}}, Q_4^{\text{self}}$ and K_4^{self} are replaced by their paired matrices, $K_1^{\text{self}}, Q_1^{\text{self}}, K_2^{\text{self}}$ and Q_2^{self}, respectively.

Next, in the decoder, making the same pairs as the ones in the encoder degrades the accuracy significantly. This is because, unlike the encoder, query and key matrices are dissimilar in the decoder; as query and key are containing different information such as source and target language. Thus, if we share the query and key as we do in the encoder, two different contents of information conflict and disturb the model training. Thus, we explore more similar matrices that can be shared without those problems: queries in self-attention and encoder-decoder attention. Those matrices conserve the accuracy when shared and reduce the number of parameters of the model. In Figure 4, we illustrate the layer and matrix replaceable pairs on the left side. Those pairs are

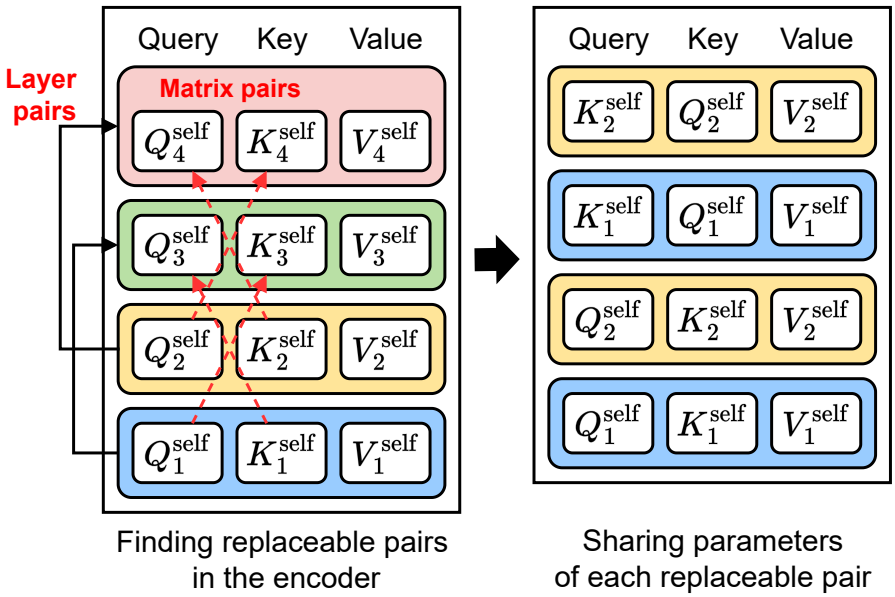

Figure 3: The example of the encoder of PET with four layers. Q_i^{self},K_i^{self},and V_i^{self} are query, key, value matrices of the self-attention in the i-th encoder layer. The left figure illustrates an uncompressed encoder, and the encoder of PET is on the right side. Note that the matrix replaceable pairs of $\{(Q_1^{\text{self}},K_3^{\text{self}}),(Q_2^{\text{self}},K_4^{\text{self}}),(K_1^{\text{self}},Q_3^{\text{self}}),(K_2^{\text{self}},Q_4^{\text{self}})\}$ are also shared.

shared on the right. The matrix replaceable pairs of Q^{self} and Q^{enc} in the left compress to a Q^{sh} in the right. In conclusion, considering that the encoder and decoder shown different replaceable pairs and those pairs are the most similar in the encoder and decoder, it important to design a compression rule differently for the encoder and decoder, and consider the similarity between the modules when finding the efficient matrix replaceable pairs.

3.2 Warm up with Simplified Task

To initialize the student that well adapts to challenging tasks efficiently, we propose a pre-training method to warm up the student model with a simplified task. Sequence-

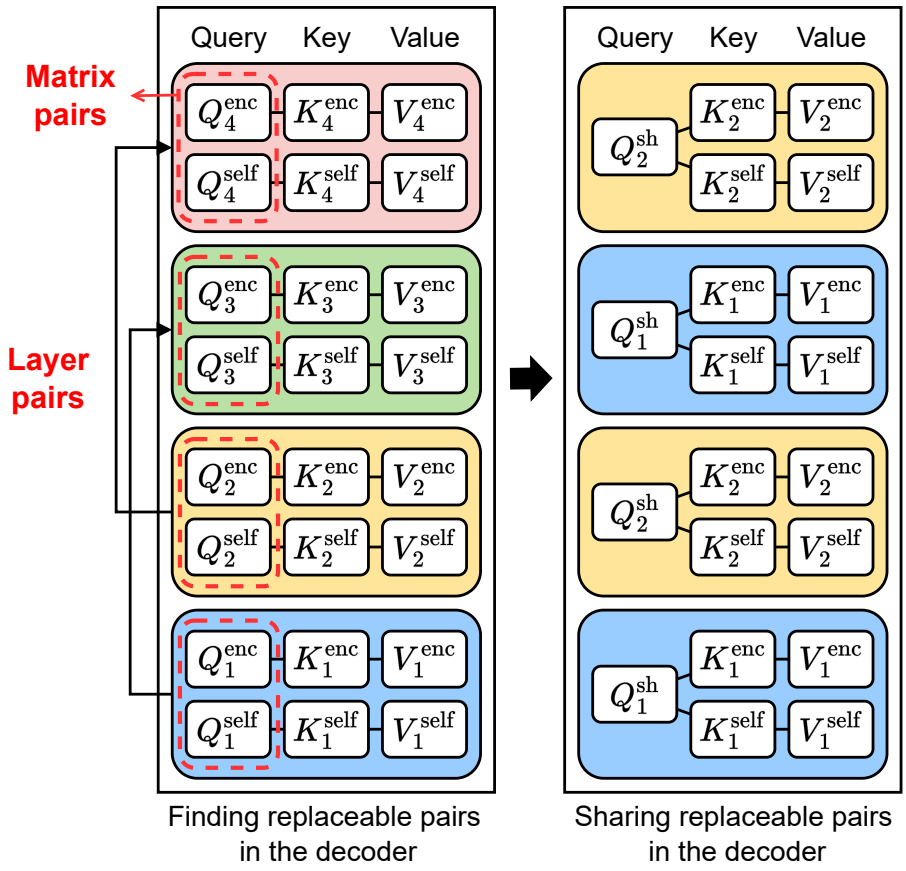

Figure 4: The example of the decoder of PET with four layers. Q_i^{self},K_i^{self}, and V_i^{self} are query, key, value matrices of the self-attention sub-layer in the i-th decoder layer. Similarly, Q_i^{enc},K_i^{enc}, and V_i^{enc} are those of the encoder-decoder attention sub-layer in the i-th decoder layer. We share (L_1, L_3) and (L_2, L_4). For the decoder, Q_i^{self} and Q_i^{enc} are shared and we denote the shared matrix as Q_i^{sh}.

to-sequence tasks can be reduced to a multi-classification task, classifying each token over the target vocabulary. These tasks usually have a lot of classes equal to the size of the target vocabulary. In our case, the student model has to classify each token into 3-40K classes. It is too challenging for a student model with a small number of parameters to classify that many classes. We make the student well prepared for the KD and its complicated task by pre-training it with an easier task that is less chal-

lenging enough for its small learning capacity.
In other words, we make a student model learn the knowledge step by step, starting with a much easier task and then solving difficult tasks.
However, there is one more challenge to be solved to enhance the pre-training performance. As we generate an easier task of classifying small number of classes, the output size of the model and the number of classes become different. Thus, it is impossible to obtain the prediction probabilities of the student model for the generated task. We solves it efficiently by modeling the prediction of the student.

The proposed method is different from the initialization methods in [8], in terms of their pursuits. simplified task addresses multi-classification tasks that are too complicated for the student to be trained with, thus, we simplify difficult tasks by reducing the number of classes from large to small. However, [8] addresses binary classification task where the student gets too simple information for the student to be trained with, therefore, it enlarges the number of classes giving more information at the pre-training step. We relieve the student with a fewer number of classes whereas [8] burdens the model with more information. Furthermore, we additionally covers how to pre-train the student model with the generated pre-training dataset more efficiently.

Warming up with a simplified task consists of two steps: generating pre-training labels of the simplified task and modeling the prediction probabilities of the student model for generated labels. We describe each step at Section 3.2.1 and 3.2.2.

3.2.1 Simplified task by Reducing the Number of Target Classes

To generate an easier task, we reduce the number of classes that the model has to classify. In [8], it uses the teacher's prediction of the original task to generate the pre-

Table 2: The label assigning rules for the teacher's predictions. We classify the prediction of the teacher into four classes.

Class	Condition
Confidently correct	$\texttt{argmax}(f_t(x_i)) = y_i$ and $\texttt{max}(f_t(x_i)) \geq \varepsilon_1$
Correct, but not confident	$\texttt{argmax}(f_t(x_i)) = y_i$ and $\texttt{max}(f_t(x_i)) \leq \varepsilon_1$
Confidently wrong	$\texttt{argmax}(f_t(x_i)) \neq y_i$ and $\texttt{max}(f_t(x_i)) \geq \varepsilon_2$
Wrong, but not confident	$\texttt{argmax}(f_t(x_i)) \neq y_i$ and $\texttt{max}(f_t(x_i)) \leq \varepsilon_2$

training dataset. We use the same approach, modifying it for sequence-to-sequence tasks. We classify the teacher's prediction $f_t(x_i)$ of each token x_i into four classes. The detailed rule for assigning a label for each data instance is described in Table 2. y_i is the target label for the token x_i and ε is the hyper-parameter for the model's confidence in its answer.

3.2.2 Modeling the Prediction Probabilities to Simplified Task Labels

We pre-train the student model with the generated pre-training dataset in Section 3.2.1. However, there is one more challenge. The output size of the student model does not match the number of classes because we reduce it to four. Figure 5 shows this mismatch. It is not possible to obtain the prediction probability of the student model for simplified task labels. There are two naïve solutions to solve this: reduce the output size of the model from the number of target classes to 4, or adopt an additional projection layer that maps the output vector of the model into 4-dimensional space. These two naïve solutions still have limitations in that they change the structure of the model by reducing the output layer size or applying an additional layer. They use different model structures for pre-training and KD and fail to efficiently leverage the pre-trained parameters in KD. That is, they use modified pre-trained knowledge as

they use different student model structures in KD from the one in the pre-training process. To use the student model as it pre-trained without any modification in its structure and maximize the effect of the pre-training, we model the prediction probabilities to the simplified labels: confidently correct, correct but not confident, confidently wrong, and wrong but not confident. How we compute $\mathbb{P}(\text{confidently correct})$ is as follows:

$$\mathbb{P}(\text{confidently correct}) = \mathbb{P}(\text{confident} \mid \text{correct})\mathbb{P}(\text{correct}) \tag{3.1}$$

We define $\mathbb{P}(\text{confident} \mid \text{correct})$ and $\mathbb{P}(\text{correct})$ as follows:

$$\begin{aligned} \mathbb{P}(\text{confident} \mid \text{correct}) &= \max(\sigma(f_s(x_i))) \\ \mathbb{P}(\text{correct}) &= f_s(x_i)_{y_i} \end{aligned} \tag{3.2}$$

where σ is the activation function.
Then, we get $\mathbb{P}(\text{confidently correct})$ as follows:

$$\mathbb{P}(\text{confidently correct}) = \max(\sigma(f_s(x_i)))f_s(x_i)_{y_i} \tag{3.3}$$

$\mathbb{P}(\text{correct, but not confident})$, $\mathbb{P}(\text{confidently wrong})$, $\mathbb{P}(\text{wrong, but not confident})$ follows the same rule. Because we model the prediction and do not modify the structure of the model in pre-training, we can directly use the same structure in KD without any modification. The right side of Figure 5 shows this. PET does not require an additional process for KD, such as replacing the classification layer or detaching the projection layer as naïve solutions do.

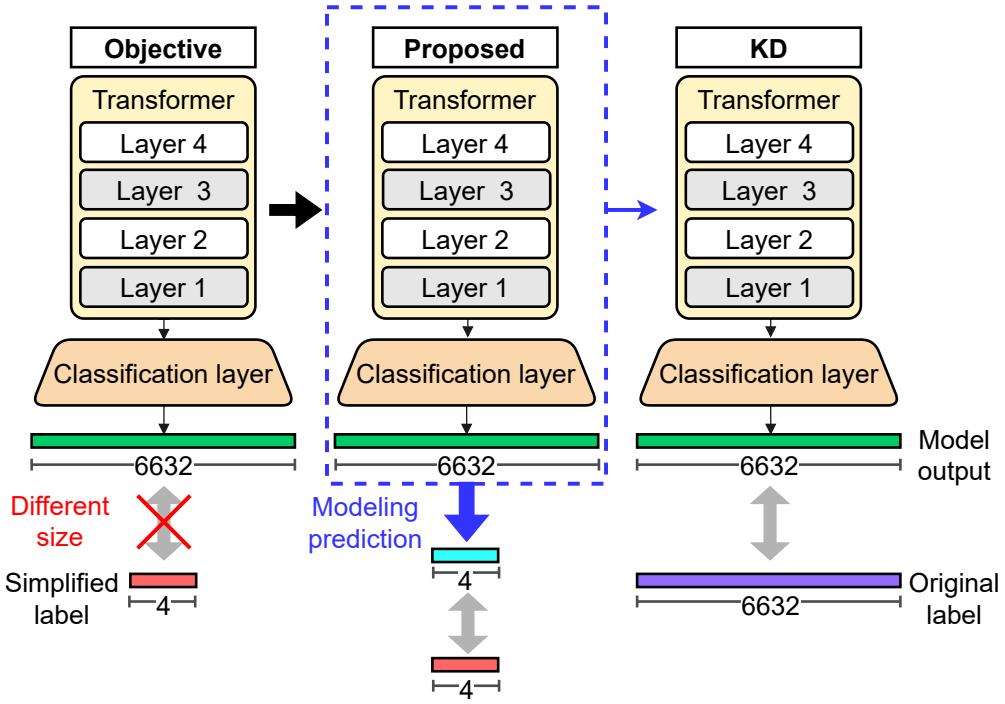

Figure 5: The example of the pre-training method PET with four layers. The number of simplified labels is much smaller than the output size of the model (In this case, 4 vs. 6632). Instead of modifying the model's structure to match the output size of the model with the size of the simplified labels, we model predictions of the model to the simplified labels. As we do not modify the structure, we can use the same parameters in the KD process.

3.3 Layer-wise Attention Head Sampling

To further improve the accuracy of the compressed model, we propose a method to optimize each shared layer with efficient attention heads. As we mentioned in Section 2.1, outputs of each attention head are concatenated to generate the final output of the multi-head-attention. We select more efficient heads among them by layer and compose each multi-head attention layer with the most accurate set of attention heads. Each layer becomes more optimized maintaining the number of parameters of the entire model. We find that if we use different heads between the shared layers,

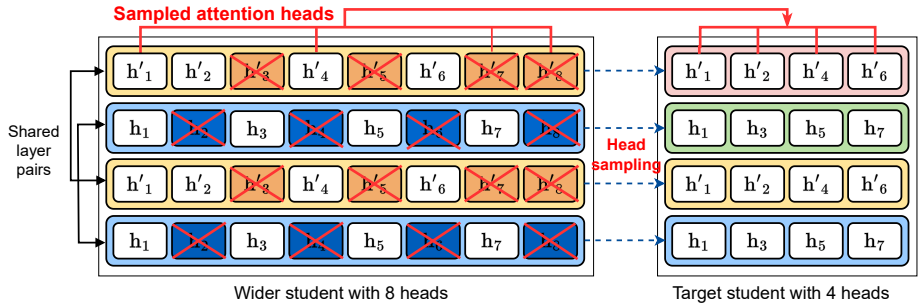

Figure 6: The example of the layer-wise attention head sampling. Wider student (Left) has eight attention heads where the size is equal to that of the target student. We sample four efficient heads per layer and construct a new target model (Right). Un-sampled attention heads are erased with an X mark, and only half of the heads move to the target student on the right.

the expressiveness of the model increases. However, there is a trade-off between the model size and accuracy gain when we select different heads in shared layers. If every shared layer select different heads from those in their paired layers, the number of the parameters remains the same for the worst case. Therefore, we use the idea of layer-wise attention head sampling as an optional technique for the proposed method, when the high accuracy is essential even with the possibility that the compression rate may be low.

In Figure 6, our compression target model is a four layer transformer model with four attention heads for all multi-head attention layers. We first train a wider student with more than four attention heads (in this case, eight heads). We apply replaceable pair and simplified task to train an accurate wider student. The wider student has the same size of attention heads as one of the target model's attention heads. We sample four heads by layers according to the magnitude of each head and fine-tune the sampled target model.

Chapter 4

Experiments

We perform experiments to answer the following questions:

Q1. **Translation accuracy.** How accurate is PET compared to the competitors? (Section 4.2)

Q2. **Translation speed.** How fast is PET compared to the competitors? (Section 4.3)

Q3. **Effectiveness of each module.** How do replaceable pair, simplified task, and layer-wise attention head sampling affect the accuracy of PET? (Sections 4.4, 4.5, and 4.6)

Q4. **Sensitivity analysis.** How much does the beam size affect the accuracy of PET? (Section 4.7)

4.1 Experimental Settings

4.1.1 Dataset

We evaluate PET on neural machine translation tasks with four datasets, IWSLT'14 DE↔ EN, and WMT'17 EN↔{FI, LV}. Detailed statistics of these datasets are summarized in Table 3. We download and preprocess IWSLT'14 following the implementation of Fairseq [17]. For WMT17 datasets, we use the official preprocessed version from WMT17 website[1], and apply the same preprocessing rule as we use for IWSLT'14 dataset.

Table 3: The summary of the datasets.

Dataset	Lang.	#classes	#sent.	#tokens
IWSLT'14 DE↔EN[1]	DE	8.85K	1.60M	40.36M
	EN	6.63K	8.89M	39.49M
WMT'17 EN↔FI[2]	EN	19.62K	10.80M	60.52M
	FI	3.54K	10.80M	52.65M
WMT'17 EN↔LV[2]	EN	23.57K	3.96M	63.05M
	LV	39.39K	3.96M	57.31M

4.1.2 Competitors

We compare PET with three competitors, Transformer-base [1], Patient KD [13], and CAN [15]. Transformer-base is a compression target for PET and other competitors (baseline). We implement PET and other competitors based on the open-source implementation [17] of the Transformer model.

4.1.3 Evaluation Metric

We use the BLEU score as an evaluation metric for accuracy, and sentence throughput for inference speed. Sentence throughput represents how many sentences the model process per second, and compression rate represents how many parameters are left compared to the original model. We include only the parameters in the transformer layer except those of the embedding layers to compute the compression rate. We run our experiments on a single machine with GeForce RTX 3090. For a fair comparison of inference speed, we have a break between each experiment to cool down the machine.

[1]http://dl.fbaipublicfiles.com/fairseq/data/iwslt14
[2]https://data.statmt.org/wmt17/translation-task/preprocessed

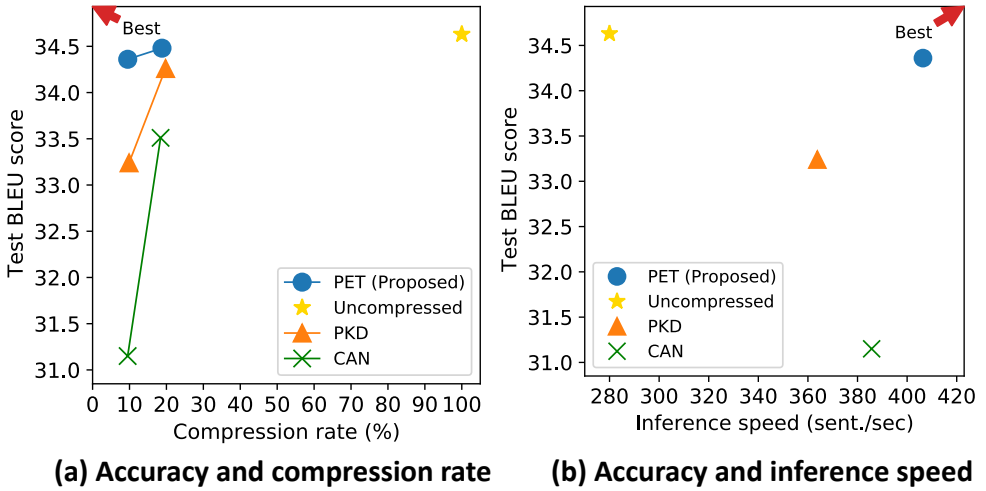

Figure 7: The translation accuracy and speed of the IWSLT'14 DE↔EN. PET shows the best BLEU performance in terms of both accuracy and inference speed. Especially, PET shows consistently high accuracy to the compression rate, while others show a significant accuracy drop as the number of the remaining parameters goes down.

4.2 Translation Accuracy of PET

We compare the translation accuracy of PET and competitors (Transformer, Patient-KD, and CAN). We set the model size the same to PET, Patient KD, and CAN. If it is difficult to compress the model size exactly the same, we set the size of PET smaller than others. Then we compare the BLEU score for each dataset.

Figure 7 (a) shows the comparison results for the accuracy of IWSLT'14 DE-EN, translating German to English. PET achieves the best BLEU score with the same or even less number of parameters. PET reduces the size of the transformer base model to 9.52% with an accuracy drop under 1%p (BLEU score loss: 0.27). Table 4 shows the Comparison of the BLEU score on the entire dataset. For all language pairs, PET achieves the best accuracy with the same or smaller size of the model. As we can see

Table 4: Comparison of the BLEU score [%]. △BLEU denotes the accuracy drop of the BLEU score compared to the baseline model. PET achieves the highest BLEU score with the same or smaller model size, for all datasets.

Task	Model	Comp. rate (%)	BLEU	△BLEU
IWSLT'14 DE→EN[2]	Baseline	100	34.63	-
	Patient-KD	9.88	33.24	1.39
	CAN	9.52	31.15	3.48
	PET (proposed)	**9.51**	**34.36**	**0.27**
IWSLT'14 EN→DE[2]	Baseline	100	29.41	-
	Patient-KD	9.60	27.50	2.47
	CAN	10.45	26.94	1.91
	PET (proposed)	**9.50**	**28.71**	**0.70**
WMT'17 EN→FI[3]	Baseline	100	14.76	-
	Patient-KD	23.55	11.76	3.00
	CAN	23.29	11.64	3.12
	PET (proposed)	**23.28**	**12.23**	**2.53**
WMT'17 EN→LV[3]	Baseline	100	17.04	-
	Patient-KD	7.13	13.63	3.41
	CAN	5.79	13.35	5.01
	PET (proposed)	**4.97**	**15.03**	**2.29**

in the case of translating English to Latvian, which has the largest number of target classes, the compressed models tend to have more difficulties in processing a classification task with a larger number of classes. PET shows it is robust to challenging tasks and datasets with many classes. This is because it is pre-trained with simplified task and made well-adapt to complicate tasks classifying over 30K number of classes. In summary, PET is an accurate and memory-efficient transformer compression method.

4.3 Translation Speed of PET

We compare the translation speed of PET and competitors on IWSLT'14 DE↔EN dataset. Figure 7(b) shows the results. We fix the size of the compression model the same as Figure 7(a) and compare the inference speed. PET shows the fastest performance processing 45.15% more sentences than baseline with only 0.27%p of accuracy drop and 10.19% faster than the slowest one with 1.12%p better accuracy. In summary, PET is an accurate and fast transformer compression method.

4.4 Effectiveness of Replaceable Pair

To verify the effect of the replaceable pair on the model accuracy, we evaluate the performance of PET on IWSLT'14 DE-EN with different sets of replaceable pair in the decoder. We fix the model size and change the pair sets of replaceable pair, and compare the BLUE score. Table 5 shows proposed replaceable pair achieves the best BLEU score with the same model size. How we pair the matrices have a significant influence on the accuracy. Unlike replaceable pair of $\{(Q^{\texttt{self}}, K^{\texttt{self}})\}$ enhance the performance of the encoder when shared, sharing those pairs ruin the performance of the decoder. This is because as we mentioned in 2.1, query and key contain different information about the source and target sequence. Therefore, sharing queries and keys disturbs the model learning by mixing different properties of data, like mixing two different languages and forcing the student to learn both of them simultaneously in translation tasks. In conclusion, it is important to compress the encoder and decoder, respectively considering their different properties such as similarities between modules.

Table 5: Comparison of the BLEU score [%] according to various replaceable pair. △BLEU is the accuracy drop of the BLEU score compared to the proposed PET. The accuracy of the model changes according to shared modules, and the proposed pair preserve the performance of the decoder the most. Accuracy loss from sharing Q^{self}, and K^{self} is serious unlike it was efficient for the encoder.

Set of pairs	BLEU	△BLEU
$\{(Q^{\texttt{self}}, Q^{\texttt{enc}})\}$ (proposed)	34.69	-
$\{(Q^{\texttt{self}}, K^{\texttt{self}})\}$	31.02	3.67
$\{(K^{\texttt{self}}, V^{\texttt{self}}), (K^{\texttt{enc}}, V^{\texttt{enc}})\}$	34.02	0.49

Table 6: Comparison of the BLEU score [%] according to w./w.o., simplified task pre-training. △BLEU is the accuracy drop of the BLEU score compared to the proposed PET. The accuracy of the model increase for PET with simplified training. Pre-training with a simplified task makes the student model warmed up for the teacher's challenging task.

Condition	BLEU	△BLEU
PET with simplified task pre-training (proposed)	34.69	-
PET without simplified task pre-training	34.39	-0.3

4.5 Effectiveness of Simplified Task

To evaluate how simplified task pre-training enhances the accuracy, we did two experiments. First, we evaluate how much simplified task affects the student model's accuracy. We compare the BLEU score of the model with and without simplified task pre-training. We set the other conditions identically. Table 6 shows that the student model well-adapt to the original challenging task at the KD process after being trained with an easier task. By learning simplified task, they can be efficiently initialized and on a better starting point.

Second, to verify how efficient our pre-training method is, we compare PET with

Table 7: Comparison of the BLEU score [%] according to pre-training methods. Pre-training method of PET succeeds in achieving the best BLUE score, by making the student model fully leverage the pre-trained parameters at the KD.

Model	Method	BLEU (%)
PET (proposed)	Compute the model predictions	34.69
Naïve solutions	Reduce the dimension of the output layer	33.19
	Use an additional projection layer	34.29

naïve solutions requiring the model structure modification. Given a simplified task of classifying only four classes for pre-training, German to English translation task classifying each token into 6632 classes, and the student model with the output size of 6632, PET and naïve solutions pre-train the student with the simplified task to improve the accuracy of the student for the challenging task. We compare the BLEU score of each student model pre-trained by PET and naïve solutions, respectively. Table 7 shows the results. As we proposed, learning simplified task without modifying the model architecture improves the pre-training performance. The student uses the pre-trained knowledge more efficiently in the KD process and obtains better accuracy.

4.6 Effectiveness of Layer-wise Attention Head Sampling

To evaluate how layer-wise attention head sampling enhances the accuracy, we compare the BLEU score of the model with and without layer-wise attention head sampling. Tables 8 shows the results. layer-wise attention head sampling optimizes the model by sampling efficient attention heads with the same number of the parameters which participates in the model prediction. Note that we mark the pruned heads as zero. Therefore, the actual number of the parameters of the PET with layer-wise

attention head sampling model are slightly more than that of the vanilla PET model.

Table 8: Comparison of the BLEU score [%] according to use of layer-wise attention head sampling. △BLEU is the accuracy drop in terms of BLEU score compared to the proposed PET. PET with layer-wise attention head sampling achieves better performance than vanilla PET. Layer-wise attention head sampling optimizes each layer of the student model by sampling the most efficient attention heads for each layer and succeeds in improving the accuracy.

Model	**BLEU**	**△BLEU**
PET	34.69	-
PET + layer-wise attention head sampling	34.90	0.21

4.7 Sensitivity Analysis

To analyze how the accuracy change according to the beam size, we compare the BLEU score with different beam sizes. We use the WMT'17 EN↔LV dataset. Translating English to Latvian is regarded as the most challenging task in our experiments in that it has the largest number of classes. Figure 8 shows the results. As the beam size decrease from 10 to 1, PET shows consistent accuracy while others show significant fluctuations in their accuracy. Especially with the beam size decreasing from 5 to 1, other methods show considerable degradations in their BLEU scores. However, PET slightly loses its accuracy. Even the uncompressed model shows more varying performance than PET. In conclusion, PET is robust to the beam sizes and shows a consistent performance.

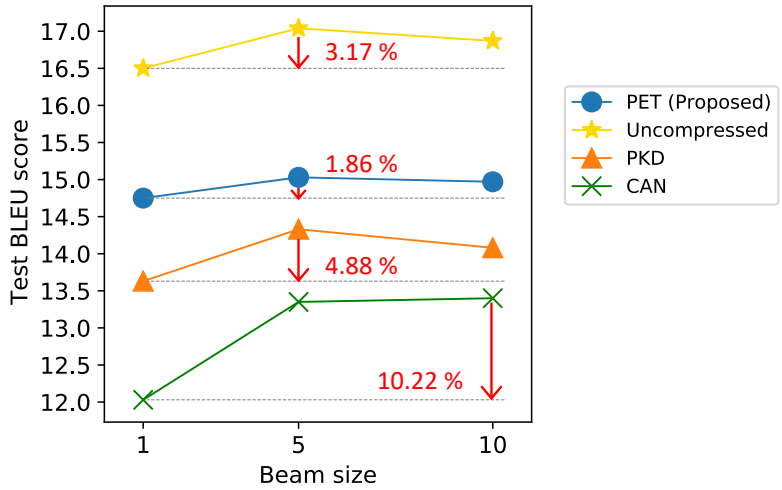

Figure 8: The translation accuracy according to the beam size. PET shows slight changes in the BLEU score, while the other competitors show significant fluctuations in the accuracy.

Chapter 5

Conclusion

We propose PET, an efficient transformer compression method reducing both the encoder and decoder. PET finds pairs of layers and matrices that are robust to parameter-sharing, and reduces the size of the model minimizing the accuracy loss. PET proposes a method to pre-train the student model more efficiently without modifying the structure of the model, and improves the pre-training performance. PET succeeds in achieving the memory efficiency and speed gain by up to 81.20% and 45.20%, respectively, with a small accuracy drop under 1%p for the English to German and German to English translation model. For future work, we will validate PET on various tasks including speech translation and multilingual translation. We will also study more delicate rules for sampling the attention heads.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

[2] Z. Wang, M. Li, R. Xu, L. Zhou, J. Lei, X. Lin, S. Wang, Z. Yang, C. Zhu, D. Hoiem, S. Chang, M. Bansal, and H. Ji, "Language models with image descriptors are strong few-shot video-language learners," *CoRR*, vol. abs/2205.10747, 2022.

[3] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zheng, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, "Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model," *CoRR*, vol. abs/2201.11990, 2022.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*, pp. 4171–4186, Association for Computational Linguistics, 2019.

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019.

[6] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, pp. 4163–4174, Association for Computational Linguistics, 2020.

[7] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic BERT for resource-limited devices," in *ACL*, pp. 2158–2170, Association for Computational Linguistics, 2020.

[8] I. Cho and U. Kang, "Pea-kd: Parameter-efficient and accurate knowledge distillation," *CoRR*, vol. abs/2009.14822, 2020.

[9] T. Piao, I. Cho, and U. Kang, "Sensimix: Sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for bert compression," *PLOS ONE*, vol. 17, pp. 1–22, 04 2022.

[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[11] J.-G. Jang, C. Quan, H. D. Lee, and U. Kang, "Falcon: Lightweight and accurate convolution," 2019.

[12] J. Kim, J. Jung, and U. Kang, "Compressing deep graph convolution network with multi-staged knowledge distillation," *PLOS ONE*, vol. 16, pp. 1–18, 08 2021.

[13] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 4322–4331, Association for Computational Linguistics, 2019.

[14] A. Edalati, M. S. Tahaei, A. Rashid, V. P. Nia, J. J. Clark, and M. Rezagholizadeh, "Kronecker decomposition for GPT compression," in *ACL (2)*, pp. 219–226, Association for Computational Linguistics, 2022.

[15] Y. Li, Y. Lin, T. Xiao, and J. Zhu, "An efficient transformer decoder with compressed sub-layers," in *AAAI*, pp. 13315–13323, AAAI Press, 2021.

[16] Y. Lin, Y. Li, Z. Wang, B. Li, Q. Du, T. Xiao, and J. Zhu, "Weight distillation: Transferring the knowledge in neural network parameters," in *ACL/IJCNLP (1)*, pp. 2076–2088, Association for Computational Linguistics, 2021.

[17] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

요 약

어떻게 하면 큰 모델의 성능을 유지하면서 작은 크기의 효율적인 연산량을 가진 트랜스포머 모델을 구할 수 있을까? 트랜스포머 모델은 지난 몇 년간 자연어 처리, 컴퓨터 비전 등 다양한 분야에 걸쳐 뛰어난 성과를 보여주고 있다. 최근 연구의 주요 추세는 모델의 크기를 늘려 모델의 성능을 높이는 것이나, 모델의 크기를 무한정 늘리는 것은 현실적인 측면에서 바람직하지 않다. 실제 서비스에 기술이 적용되기 위해선 높은 성능 뿐만 아니라 메모리 효율, 빠른 추론 속도, 에너지 소모량 등에 대한 고려가 필요하며, 대부분 모델의 크기가 큰 경우 이를 만족하기 어렵다. 따라서 큰 모델의 성능을 유지하면서 작고 빠른 모델을 얻기 위한 효과적인 트랜스포머 모델 압축 기술이 필요하다.

기존의 트랜스포머 압축 연구들은 트랜스포머 인코더 기반 모델에 대한 것이 대부분으로, BERT 압축이 대표적이다. 기존의 인코더 압축 기법을 기계 번역 모델 등 인코더와 디코더가 혼재하는 모델에 적용할 경우 정확도가 크게 손실되었다. 디코더는 동일 임베딩 사이즈와 레이어 수의 인코더보다 크기가 크며, 긴 추론 시간의 주요 원인이므로, 실용적인 트랜스포머 모델을 만들기 위해선 디코더 압축이 필수적이다. 이 논문에서는 트랜스포머의 인코더와 디코더를 모두 압축하기 위한 PET (Parameter-Efficient Knowledge Distillation on Transformer)를 제안한다. 제안 기법은 효과적인 모델 구조 설계와 초기화 기법의 개선을 통해 트랜스포머 지식 증류 기법의 성능을 높였다. 또한, 추가적인 최적화 기법을 제안하여 압축 모델의 정확도를 더욱 높이는 데에 성공하였다. 실험을 통해 제안 기법이 다양한 기계 번역 데이터 셋에서 경쟁 모델보다 우수한 성능을 보이는 것을 확인하였고, 독일어 영어 번역 데이터 셋에서는 원본 모델보다 18.30% (임베딩 레이어 제외 시 9.51%)의 파라미터 수로 45.2% 빠르면

서 정확도 감소를 1% p 이내로 줄이는 데 성공하였다.

주요어 : 모델 압축, 트랜스포머, 지식 증류

학번 : 2021-24350

감사의 글

많은 분들의 격려와 도움으로 이 논문을 완성할 수 있었기에 이 자리를 빌려 감사의 말씀을 전합니다.
먼저 , 2년 동안 많은 배움과 경험의 기회를 주신 강유 교수님께 진심으로 감사드립니다. 교수님의 연구에 대한 열정, 철저한 자기 관리를 보며 평생을 간직할 배움을 얻었고, 졸업 후에도 이를 되새기며 더욱 성장하도록 하겠습니다. 또한 바쁘신 중에도 귀한 시간을 내어 학위 논문 심사에 참여해주신 문봉기 교수님과 이상구 교수님께도 깊이 감사드립니다.
2년 간 동고동락했던 데이터 마이닝 연구실 선배님, 동료, 후배, 행정원 선생님께도 감사의 인사를 전합니다. 어려움이 생길 때다 자신의 일처럼 도와주고, 항상 서로를 응원해주는 연구실 분위기 덕에 행복한 연구실 생활을 할 수 있었습니다. 앞으로도 연구실에 좋은 일만 가득하길 진심으로 기원합니다.
대학원 진학 후 바쁘다는 핑계로 연락도 잘 못했는데 항상 저를 생각해주고 응원해주는 친구들에게 미안하고 고맙다는 인사를 전합니다. 친구들이 보내준 많은 응원과 사랑, 신뢰에 보답하겠습니다.
많은 의지와 힘이 되고 있는 덩어리에게 정말 고맙고 사랑한다는 말을 전합니다. 저 또한 언제나 의지할 수 있는 든든한 버팀목이 되겠습니다. 마지막으로 저를 항상 믿어주고 응원해주는 우리 가족, 너무나 사랑합니다. 엄마, 아빠의 딸이라서, 지현이의 언니라서 자랑스럽고, 감사하며 행복합니다. 모두 고맙습니다.