Adaptive Adapters: An Efficient Way to Incorporate BERT Into Neural Machine Translation

Junliang Guo, Zhirui Zhang, Linli Xu, Boxing Chen, and Enhong Chen, *Senior Member, IEEE*

***Abstract*—Large-scale pre-trained language models (e.g., BERT) have attracted great attention in recent years. It is straightforward to fine-tune them on natural language understanding tasks such as text classification, however, effectively and efficiently incorporating them into natural language generation tasks such as neural machine translation remains a challenging problem. In this paper, we integrate two pre-trained BERT models from the source and target language domains into a sequence-to-sequence model by introducing light-weight adapter modules. The adapters are inserted between BERT layers and tuned on downstream tasks, while the parameters of BERT models are fixed during fine-tuning. As pre-trained language models are usually very deep, inserting adapters into all layers will result in a considerable scale of new parameters. To deal with this problem, we introduce latent variables to decide whether using adapters or not in each layer, which are learned during fine-tuning. In this way, the model is able to automatically determine which adapters to use, therefore hugely promoting the parameter efficiency and decoding speed. We evaluate the proposed framework on various neural machine translation tasks. Equipped with parallel sequence decoding, our model consistently outperforms autoregressive baselines while reducing the inference latency by half. With automatic adapter selection, the proposed model further achieves 20% speedup while still outperforming autoregressive baselines. When applied to autoregressive decoding, the proposed model can also achieve comparable performance with the state-of-the-art baseline models.**

***Index Terms*—Pre-trained language model, adapter, neural machine translation.**

I. Introduction

In recent years, Pre-trained Language Models (PLMs) [1]–[5] have achieved outstanding performance on many natural language understanding tasks such as text classification and reading comprehension [6], [7]. The deployment of these PLMs typically consists of two steps. Firstly, the PLM is pre-trained on large scale corpora in a self-supervised manner, then the model is fine-tuned on downstream tasks with task-specific loss functions and datasets. Fine-tuning a PLM on natural language understanding tasks are straightforward, usually by treating the PLM as a feature extractor and introducing a new classification module above it. However, on natural language generation tasks which are mostly based on the sequence-to-sequence framework (e.g., neural machine translation (NMT) [8], [9] and text summarization [10]), how to incorporate PLMs remains a challenging problem.

In the literature, several recent works [11]–[14] have tried to incorporate BERT, which is one of the most successful representatives of PLMs, into natural language generation tasks, mainly by leveraging the feature representations encoded by BERT. And most of them only utilize the pre-trained BERT model either on the source side or the target side. Given the sequence-to-sequence framework as the backbone model, we explore utilizing the pre-trained BERT models from both the encoder and decoder sides to maximize the use of pre-trained information, and therefore promote the final performance. However, it is non-trivial to achieve this goal, due to the challenges summarized below.

On the encoder side, simply initializing the encoder with a pre-trained BERT model and then fine-tuning it on downstream tasks does not bring improvements, and sometimes even hurts the performance [14]. We blame it to the catastrophic forgetting problem [15] when fine-tuning pre-trained models on complex downstream tasks with rich resources (e.g., machine translation). On the decoder side which is usually conditioned on the encoder representations and trained in an autoregressive manner, it is naturally non-trivial to marry it with the unconditionally and non-autoregressively pre-trained BERT model. In addition, given the enormous parameter scale of recent pre-trained language models [5], fine-tuning the whole model is parameter inefficient while being unstable and fragile on small datasets [16].

To tackle these challenges, in our preliminary work [17], we introduce light-weight neural network components named *adapters*, based on which we propose a new paradigm of incorporating BERT into the sequence-to-sequence framework. Specifically, we first choose two pre-trained BERT models from the source/target side respectively, and consider them as the encoder/decoder. For example, on the WMT14 English-German machine translation task, we take `bert-base-cased` as the encoder and `bert-base-german-cased` as the decoder.

Manuscript received January 4, 2021; revised April 3, 2021; accepted April 22, 2021. Date of publication April 30, 2021; date of current version May 28, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant U20A20229, by Anhui Provincial Natural Science Foundation under Grant 2008085J31, and by Alibaba Group through Alibaba Research Intern Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sakriani Sakti. *(Corresponding author: Linli Xu)*

Junliang Guo and Enhong Chen are with the Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: guojunll@mail.ustc.edu.cn; cheneh@ustc.edu.cn).

Zhirui Zhang and Boxing Chen are with Alibaba Damo Academy, Hangzhou 310052, China (e-mail: zhirui.zzr@alibaba-inc.com; boxing.cbx@alibaba-inc.com).

Linli Xu is with the Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science, and Technology of China, Hefei 230026, China and also with IFLYTEK Co., Ltd, Hefei Anhui , China (e-mail: linlixu@ustc.edu.cn).

Digital Object Identifier 10.1109/TASLP.2021.3076863

Then, we introduce *adapter* layers and insert them into each BERT layer to achieve the adaptation to new tasks. While fine-tuning on task-specific datasets, we freeze the parameters of BERT layers and only tune the adapter layers. We design different architectures for adapters. Specifically, we stack two feed-forward networks as the encoder adapter, mainly inspired from [18]; and an encoder-decoder attention module is considered as the decoder adapter. Considering that BERT utilizes bi-directional context information and ignores conditional dependency between tokens, we build our framework on a parallel sequence decoding algorithm named Mask-Predict [19] to make the most of BERT and keep the consistency between training and inference.

In this way, the proposed framework (termed as Adapter-Bert Networks, AB-Net) achieves the following benefits. 1) By introducing the adapter modules, we decouple the parameters of the pre-trained language model and task-specific adapters, therefore bypassing the catastrophic forgetting problem. And the conditional information can be learned through the cross-attention based adapter on the decoder side; 2) Our model is parameter efficient and robust while tuning as a benefit from the lightweight nature of adapter modules. In addition, thanks to parallel decoding, the proposed framework achieves better performance than autoregressive baselines while doubling the decoding speed; 3) Each component in the framework can be considered as a plug-in unit, making the framework very flexible and task agnostic. For example, our framework can be adapted to autoregressive decoding straightforwardly by only incorporating the source-side BERT encoder and adapters while keeping the original Transformer decoder.

As the pre-trained BERT models are usually very deep (12 layers for `bert-base` models while 24 layers for `bert-large` models), inserting adapters into *every* BERT layer will bring a non-negligible number of new parameters. In addition, recently, several works [20], [21] have illustrated that some layers in deep Transformers can be pruned with little loss of performance. From this point, in this paper, we investigate whether it is necessary to insert adapters into every BERT layer. Specifically, we introduce latent variables to decide whether to use an adapter or not at each layer, which is learned with a probabilistic method. While inference, the adapters are selected by the discrete decisions sampled from the latent variables, which are learned in an end-to-end way by utilizing the Gumbel-Softmax approximation [22] while training. The latent variables are optimized with variational inference [23], and we can also control the number of adapter layers by introducing an extra loss function. In this way, we achieve automatic pruning of adapter layers, which hugely reduces the parameter scale of adapters as well as the decoding latency of the model while inference. In addition, the selection decisions learned by the model also reveals instructive information, for example, different languages prefer to insert adapters into different BERT layers.

In experiments, we evaluate our framework on various neural machine translation tasks. The proposed AB-Net achieves 36.49/33.57 BLEU scores on the IWSLT14 German-English/WMT14 German-English translation tasks, achieving 3.5/0.88 improvements over traditional autoregressive baselines with half of the inference latency. Equipped with automatically selecting and utilizing half of adapter layers, the inference speed of the model is further accelerated by 20%, while still outperforming the corresponding autoregressive baselines. When adapting to autoregressive decoding, the AB-Net achieve 30.60/43.56 BLEU scores on the WMT14 English-German/English-French translation tasks, on par with the state-of-the-art baseline models.

The rest of this paper is organized as follows. In Section II, we introduce related works and backgrounds. The proposed framework is introduced in Section III, followed by extensive experimental results and analyses in Section IV. Finally, Section V concludes the paper.

II. Related Work

A. Pre-Trained Language Models

Pre-trained Language Models (PLMs) aim at learning powerful and contextual language representations from a large text corpus by self-supervised learning [1]–[5], [24]–[26], and they have remarkably boosted the performance of standard natural language understanding tasks such as the GLUE benchmark [6]. BERT [3] is one of the most popular pre-training approaches, whose pre-training objective consists of masked language modeling (MLM) and next sentence prediction. The idea of MLM has been applied widely to other tasks such as neural machine translation [19]. Given an input sentence $x = (x_1, x_2, \ldots, x_n)$, MLM first randomly chooses a fraction (usually 15%) of tokens in x and substitutes them by a special symbol `[MASK]`, then predicts the masked tokens by the residual ones. Denote x^m as the masked tokens and x^r as the residual tokens, the objective function of MLM can be written as:

$$L_{\text{MLM}}(x^m|x^r;\theta_{\text{enc}}) = -\sum_{t=1}^{|x^m|}\log p(x_t^m|x^r;\theta_{\text{enc}}), \qquad (1)$$

where $|x^m|$ indicates the number of masked tokens.

Among the alternative pre-training methods, UniLM [25] extends BERT with unidirectional and sequence-to-sequence predicting objectives, making it possible to fine-tune the pre-trained model for natural language generation tasks. XLM [27] achieves cross-lingual pre-training on supervised parallel datasets with a similar objective function as MLM. MASS [28] proposes a sequence-to-sequence monolingual pre-training framework where the encoder takes the residual tokens x^r as input and the decoder predicts the masked tokens x^m autoregressively. BART [29] adopts a similar framework and trains the model as a denoising autoencoder. Although achieving impressive results on various natural language tasks, these models are equipped with large-scale training corpora, therefore are time and resource consuming to train from scratch. In this paper, we focus on leveraging public pre-trained BERT models in the sequence-to-sequence framework.

B. PLMs in Sequence-to-Sequence Framework

There exist several recent works trying to incorporate pre-trained language models, or specifically, BERT, into the sequence-to-sequence framework, which are mainly focused on leveraging the feature representations of BERT. Knowledge

distillation [30], [31] is applied in [11]–[13] to transfer the knowledge from BERT to either the encoder [12] or decoder side [11], [13]. Extra attention based modules are introduced in [14] to fuse the BERT representation with the encoder representation. Most of these methods only incorporate BERT on either the source side or the target side. Our framework, on the other hand, is able to utilize the information of BERT from both sides.

Directly fine-tuning the pre-trained language models requires delicately tuning hyper-parameters such as the learning rate [32], which is also unstable and sensitive as studied in Section IV-E. Therefore, we introduce adapters to deal with this problem. Adapters are usually light-weight neural networks added into internal layers of pre-trained models to achieve the adaptation to downstream tasks, and have been successfully applied to fine-tune vision models [33], language models [34], [35] and multilingual machine translation models [18], [36]. Different from these works, we explore combining two pre-trained models from different domains into a sequence-to-sequence framework with the help of adapters.

C. Parallel Decoding

Parallel sequence decoding hugely reduces the inference latency by neglecting the conditional dependency between output tokens, based on novel decoding algorithms including non-autoregressive decoding [37]–[40], insertion-based decoding [41], [42] and Mask-Predict [19], [43]. In Mask-Predict, the framework is trained as a conditional masked language model as:

$$L_{\text{CMLM}}(y^m|y^r,x;\theta_{\text{enc}},\theta_{\text{dec}}) = -\sum_{t=1}^{|y^m|}\log p(y_t^m|y^r,x;\theta_{\text{enc}},\theta_{\text{dec}}), \tag{2}$$

where (x, y) is a sample of parallel training pairs from the dataset, y^m and y^r are the masked/residual target tokens, θ_{enc} and θ_{dec} are the parameters of the encoder and decoder respectively. During inference, the model iteratively generates the target sequence in a mask-and-predict manner, which fits well with the bi-directional and conditional independent nature of BERT. Inspired by that, we conduct training and inference of our model in a similar way, which is introduced in Section III-D.

D. Network Pruning on Transformers

Network pruning [44] on deep neural networks, which aims at removing redundant parameters of deep and large neural networks without a significant drop of performance, has been widely studied in recent years, especially on computer vision tasks as well as convolutional neural networks [45]–[49]. On the Transformer based networks, related works have studied to prune the multi-head attention components [50] or the entire layer [20], as well as dynamically selecting layers in very deep models [51], [52]. In addition, various works have been proposed to compress the pre-trained language model BERT [53]–[56] with knowledge distillation [30]. Inspired by the layer selection method in deep Transformers [52], in this paper, we propose to automatically select and prune adapter layers in inference, to achieve further improvement of the decoding speed and enhance the interpretability of our model.

III. Framework

In this section we will first introduce the proposed framework of fine-tuning BERT with adapters (termed as Adapter-Bert Networks, AB-Net), followed with the adapter selection component, which is illustrated in Figure 1. We start with the problem definition.

Problem Definition Given two pre-trained BERT models XBERT and YBERT on the source side and the target side respectively, we aim at fine-tuning them in a sequence-to-sequence framework by introducing adapter modules, on a parallel training dataset $(\mathcal{X}, \mathcal{Y})$ which consists of pairs of source and target sequences $(x, y) \in (\mathcal{X}, \mathcal{Y})$. The loss function of our framework is defined in a similar way as the conditional MLM loss introduced in Equation (2):

$$L(y^m|y^r,x;\theta_{\text{AENC}},\theta_{\text{ADEC}}) = -\sum_{t=1}^{|y^m|}\log p(y_t^m|y^r,x;\theta_{\text{AENC}},\theta_{\text{ADEC}}), \tag{3}$$

where θ_{AENC} and θ_{ADEC} indicate the parameters of encoder adapters and decoder adapters respectively.

A. Adapter-Bert Networks

The architecture of BERT [3] is akin to a Transformer encoder [9], which is constructed by self-attention, feed-forward layers, layer normalization [57] and residual connections [58]. We denote a BERT layer block as $\text{XBERT}(\cdot)$ or $\text{YBERT}(\cdot)$.

To fine-tune BERT for natural language generation, we introduce adapter layers and insert them into each BERT layer. On the encoder side, we follow [18] and simply construct the adapter layer with layer normalization as well as two feed-forward networks with non-linearity between them:

$$Z = W_1 \cdot (\text{LN}(H)), \quad H_{\text{AENC}} = H + W_2 \cdot (\text{ReLU}(Z)), \tag{4}$$

where H and H_{AENC} are the input and output hidden states of the adapter respectively, LN indicates layer normalization, W_1 and W_2 are the parameters of the feed-forward networks. The only hyper-parameter brought by this module is the dimension d_{Aenc} of the internal hidden state Z, through which we can flexibly control the capacity and efficiency of adapter layers. Denoting the encoder adapter layer as $\text{AENC}(\cdot)$, for each encoder layer in the framework, the hidden state is computed as:

$$H_{l+1}^E = \text{AENC}(\text{XBERT}(H_l^E)), \tag{5}$$

where H_l^E is the output hidden state of the l-th encoder layer. And we take the hidden state H^E of the last encoder layer as the representation of the source sequence.

As for the decoder, the introduced adapter modules should be able to model the conditional information from the source side. Therefore, we adopt the multi-head cross-attention computed over the encoder output and decoder input as the adapter. Denote the attention-based adapter as $\text{ADEC}(Q, K, V)$ where Q, K, V indicate the query, key and value matrices respectively, it consists of the attention module, feed-forward layers, layer

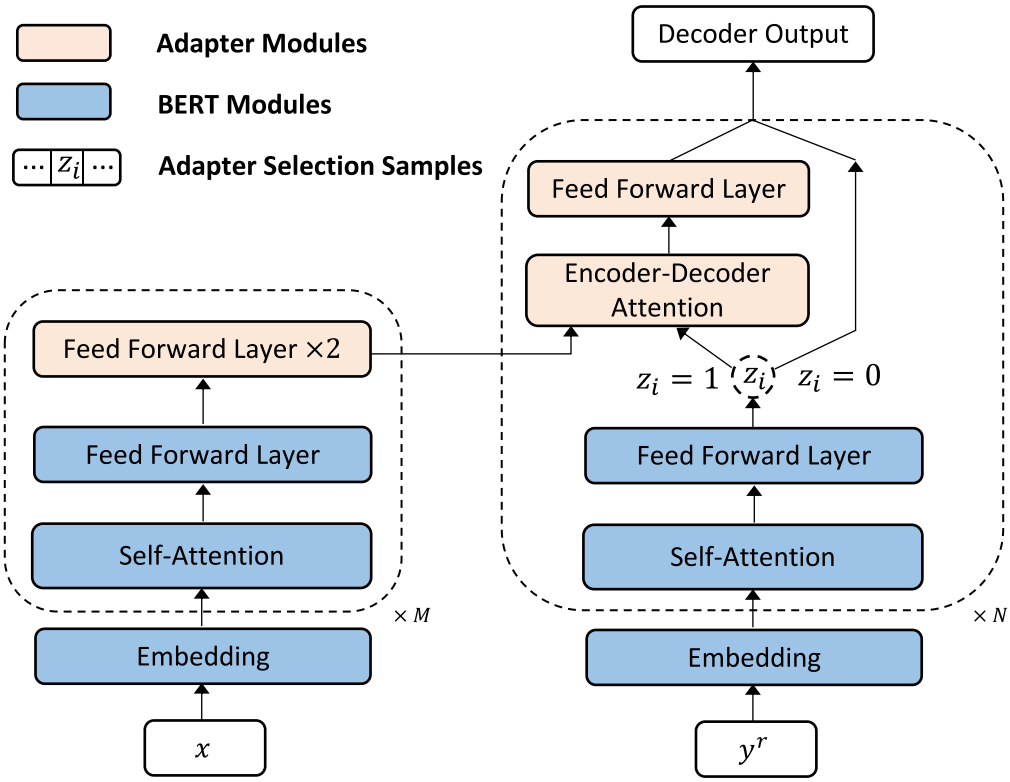


Fig. 1. An illustration of the proposed framework. Blue blocks constitute the pre-trained BERT models which are frozen during fine-tuning, and orange blocks represent the adapter components which are inserted into each BERT layer and trained during fine-tuning. x and y^r represent the source sequence and the residual target sequence in Equation (3) respectively. M and N indicate the number of layers of the encoder and decoder.$z = (z_0, \ldots, z_{N-1})$ indicates the latent variables regarding layer selection sampled from the distribution $p(z)$, and we illustrate the adapter selection on the decoder.For simplicity, we omit some architecture details such as layer normalization and residual connections.

normalization and residual connections. The attention module is computed as follows,

$$\text{ATTN}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{6}$$

where d_k is the hidden dimension of the key matrix K. We follow [9] and implement the multi-head version of the attention module. In our framework, the query vector is from the decoder side (denoted as H_l^D) while the key and value vectors are both from the encoder side (denoted as H^E). In our experiments, the hidden dimension of encoder and decoder representations are the same, therefore we have $d_q = d_k = d_v = d_{\text{Adec}}$.

Following the attention layer are the feed-forward layers:

$$\text{FFN}(H) = \text{ReLU}(HW_1 + b_1) \cdot W_2 + b_2, \tag{7}$$

where $W_1 \in \mathbb{R}^{d_{\text{Adec}} \times d_{\text{FFN}}}, W_2 \in \mathbb{R}^{d_{\text{FFN}} \times d_{\text{Adec}}}, b_1 \in \mathbb{R}^{d_{\text{FFN}}}, b_2 \in \mathbb{R}^{d_{\text{Adec}}}$ are the parameters to learn in the FFN layers, and the internal dimension d_{FFN} is set to be consistent with the Transformer baseline. Along with layer normalization (LN) and residual connections, the computation flow of the proposed decoder adapter can be written as:

$$\begin{aligned} \hat{H} &= \text{LN}\left(\text{ATTN}(\text{YBERT}(H_l^D), H^E, H^E) + \text{YBERT}(H_l^D)\right), \\ H_{l+1}^D &= \text{LN}\left(\text{FFN}(\hat{H}) + \hat{H}\right). \end{aligned} \tag{8}$$

For simplicity, given the decoder adapter ADEC, the hidden output of the encoder H^E, and the hidden output H_l^D of the l-th decoder layer, the hidden state of the $(l+1)$-th layer is calculated as:

$$H_{l+1}^D = \text{ADEC}(\text{YBERT}(H_l^D), H^E, H^E). \tag{9}$$

By introducing and carefully designing the adapter modules on the encoder and decoder, our framework is able to utilize the pre-trained information from both sides as well as build the conditional dependency, making it possible to apply the model to natural language generation tasks.

B. Adapter Layer Selection

As discussed in Section I, some layers in deep Transformer models can be pruned without severely hurting the performance. Similarly, we conjecture that some adapter layers can also be pruned because not all adapters play important roles while fine-tuning. Therefore, we propose a probabilistic method to let the model automatically decide to select and use adapter layers. We follow the notations in [52].

Specifically, we introduce a discrete latent variable $z_l \sim p(z)$ for the l-th adapter layer, sampled from a Bernoulli distribution $\mathcal{B}(\beta_l)$, where β_l is the probability for selecting the l-th adapter layer. $z_l \in \{0, 1\}$ and $z_l = 1$ indicates selecting and utilizing the l-th adapter layer while $z_l = 0$ indicates skipping it. More formally, take the adapter on the decoder side as an example, with adapter layer selection, the computation flow described in

Equation (9) is re-written as:

$$H_{l+1}^{D} = \begin{cases} \textsc{Adec}(\textsc{YBert}(H_l^D), H^E, H^E), & \text{if } z_l = 1 \\ \textsc{YBert}(H_l^D). & \text{if } z_l = 0 \end{cases} \tag{10}$$

Denote the trainable parameters of the AB-Net as $\theta = (\theta_{\textsc{Aenc}}, \theta_{\textsc{Adec}})$, the prediction of the target sequence y^m in Equation (3) reduces to:

$$p(y^m|y^r, x; \theta, z) = \int_z p(y^m|y^r, x; \theta, z)p(z)\mathrm{d}z. \tag{11}$$

The above integral of the marginal likelihood is intractable for most deep neural networks, therefore we utilize variational inference and maximize the evidence lower bound (ELBO) [59] of the above equation,

$$\log p(y^m|y^r, x) \geq \mathbb{E}_{q_\phi(z)}[\log p_\theta(y^m|y^r, x, z)] - D_{\mathrm{KL}}(q_\phi(z) \parallel p(z)). \tag{12}$$

The first RHS term is akin to the original prediction loss of our framework defined in Equation (3), except for the expectation over the latent variable z. In the second RHS term, D_{KL} indicates the KL-divergence, $q_\phi(z)$ is an approximation of the true posterior, and we choose the uniform prior as $p(z)$ in our experiments to avoid introducing prior preference of selecting or skipping adapter layers, but let the model learn to decide it.

To make the model differentiable in end-to-end training, we utilize the Gumbel-Softmax reparameterization [22] to deal with $q_\phi(z)$. The i-th sample $z_l(i)$ in the l-th layer latent variable z_l is generated as:

$$z_l(i) = \frac{\exp((\beta_l(i) + g(i))/\tau)}{\sum_{j\in\{0,1\}} \exp((\beta_l(j) + g(j))/\tau)}, \quad g(i) \overset{\text{i.i.d}}{\sim} \text{Gumbel}(0,1) \quad \text{for } i \in \{0,1\}, \tag{13}$$

where $\tau \in (0, \infty)$ is the hyper-parameter that controls the temperature, and $\tau \to 0$ indicates the samples become one-hot. The Gumbel Distribution Gumbel$\tilde{}(0,1)$ can be sampled using the inverse transform sampling of an auxiliary random uniform variable $u(i) \sim \text{Uniform}(0,1)$ and $g(i) = -\log(-\log u(i))$. In this way, the layer selection probability β can be learned by the model while training in an end-to-end way.

To achieve model pruning and inference speedup, in addition to Equation (12), we also introduce a loss function to restrict the number of selected layers. Given the latent variables $z = (z_0, \ldots, z_{N-1})$ which are sampled independently across all the N layers and the target layer number K, the loss function is defined based on the L_2 distance

$$L_K(z) = \parallel \sum_{l=0}^{N-1} z_l - K \parallel_2 . \tag{14}$$

Rewrite Equation (12) and integrate the proposed loss functions, our framework is trained by minimizing the following objective,

$$L(y^m|y^r, x; \theta, \phi) = \mathbb{E}_{q_\phi(z)}[-\log p_\theta(y^m|y^r, x, z)] + \mu D_{\mathrm{KL}}(q_\phi(z) \parallel p(z)) + \eta L_K(z), \tag{15}$$

where μ and η are hyper-parameters that control the weights of different loss functions.

C. Discussion

Different from most previous works that plainly utilize BERT as a feature extractor [11], [12], [14], we directly exploit BERT as the encoder and decoder to make the most of pre-trained models. Comparing with the related works that also utilize adapter modules while fine-tuning [18], [34], [35], we do not constrain the architectures of adapters to be fixed, but adopt different architectures on the encoder and decoder sides. In addition, we can easily extend the architectures of adapters to adjust to different downstream tasks. For example, while our framework is designed for parallel decoding, it is straightforward to transform it to traditional autoregressive decoding by extending the cross-attention based adapter to a traditional Transformer decoder. We show in Table IV that our autoregressive variant is able to achieve strong performance.

Meanwhile, integrating two large scale pre-trained models into a sequence-to-sequence framework also introduces non-negligible issues. The main limitation is the extra computation cost brought by the enormous pre-trained parameters. Fortunately, thanks to the lightweight and flexible adapter modules, the scale of parameters that require training in our framework is smaller than that of an autoregressive Transformer model. Additionally, with the proposed layer selection method, we can only utilize a part of adapters while inference, which is able to reduce the computation cost as well as accelerate the decoding speed when deployed on real-world applications. We have also explored heuristic ways to adjust the scale of adapters. For example, while training, instead of inserting adapter layers to all BERT layers, we can only insert them into the top layers to speed up training. While still outperforming autoregressive baselines, this heuristic adapter selection method is inferior to the proposed latent adapter selection, illustrating the benefits of letting the model learn to select. We can also reduce the hidden dimensions of adapters to control the parameter scale with negligible degradation of performance. A thorough study is conducted regarding the flexibility of adapters in Section IV-E. It can be shown that, at the inference stage, even with two large pre-trained models introduced, our framework based on parallel decoding can still achieve faster decoding speed than traditional autoregressive baselines.

D. Training and Inference

We mainly follow the training and inference paradigm used in [19]. To decode the target sequence in parallel, the model needs to predict the target length conditioned on the source sequence, i.e., modeling $P(|y||x)$. We add a special `[LENGTH]` token to the encoder input, and take its encoder output as the representation, based on which the target length is predicted. The length prediction loss is added to the word prediction loss in Equation (15) as the final loss of our framework. In Equation (15), given a training pair (x, y), we randomly mask a set of tokens in y with `[MASK]`, and the number of the masked tokens $|y^m|$ is uniformly sampled from 1 to $|y|$ instead of being computed by a fixed ratio as BERT [3]. The masked tokens are denoted as y^m while the residual tokens are denoted as y^r,

While inference, the target sequence is generated iteratively in a mask-and-predict manner. Specifically, after the length of the

TABLE I
AN ILLUSTRATION OF THE PROPOSED MODEL WITH DIFFERENT NUMBER OF DECODING ITERATIONS k ON THE TEST SET OF THE IWSLT14 DE-EN TASK. "##" INDICATES THE SEGMENT SYMBOL OF WORDPIECE TOKENS

Source:	der film wird " the g ##rea ##test mo ##vi ##e ev ##er sol ##d " heißen .
Target:	the movie will be called " the greatest movie ever sold . "
AB-Net with different iteration k	
$k = 1$:	the film is called called " the the movie ever o . "
$k = 2$:	the film is called " " the the movie ever computer . "
$k = 3$:	the film is being called " the the movie ever known . "
$k = 4$:	the movie will be called " the greatest movie ever known . "

TABLE II
THE BLEU SCORES OF THE PROPOSED AB-NET AND THE BASELINE METHODS ON THE IWSLT14 DE-EN, WMT16 RO-EN AND WMT14 EN-DE/DE-EN TASKS. THE PER-SENTENCE DECODING LATENCY AND THE NUMBER OF TRAINED PARAMETERS ON THE WMT14 EN-DE TASK ARE ALSO REPORTED

Models	**IWSLT14** De−En	**WMT16** Ro−En	**WMT14** En−De	**WMT14** De−En	Latency	#Introduced Parameters
Transformer-Base [9]	33.59*	34.46*	28.04*	32.69*	778* ms	74M
Mask-Predict [19]	31.71*	33.31	27.03	30.53	161* ms	75M
BERT-Fused NAT [14]	33.14*	34.12*	27.73*	32.10*	260* ms	90M
AB-Net	**36.49**	**35.63**	**28.69**	**33.57**	327 ms	67M
AB-Net-Enc	34.45	/	28.08	/	165 ms	78M
AB-Net w/ ADEC Selection	36.25	35.20	27.80	33.17	259 ms	46M

TABLE III
THE PERFORMANCE OF THE PROPOSED AB-NET ON IWSLT14 LOW-RESOURCE LANGUAGE PAIRS. MASK-PREDICT AS WELL AS THE AUTOREGRESSIVE TRANSFORMER-BASE MODEL ARE CONSIDERED AS BASELINES

Models	En-It	It-En	En-Es	Es-En	En-Nl	Nl-En
Transformer-Base [9]	29.26	33.57	36.04	39.31	31.30	36.19
Mask-Predict [19]	26.05	29.50	32.15	35.37	25.78	32.91
AB-Net	**31.81**	**34.20**	**37.45**	**42.66**	**32.52**	**38.94**

TABLE IV
THE RESULTS OF MACHINE TRANSLATION WITH AUTOREGRESSIVE DECODING OF OUR FRAMEWORK AND BASELINE METHODS

Models	**WMT14** En−De	**WMT14** En−Fr	**WMT16** Ro−En
Transformer-Big [9]	28.91*	42.23*	36.46*
BERT-Distilled [13]	27.53	/	/
CT-NMT [12]	30.10	42.30	/
BERT-Fused [14]	30.75	43.78	39.10
AB-Net-Enc	30.60	43.56	39.21

target sequence is predicted by the encoder, the decoder input is initialized with the `[MASK]` symbol at all positions. After the prediction process of the decoder, a number of tokens with the lowest probabilities in the decoder output are replaced by `[MASK]`. The obtained sequence is taken as the decoder input of the next iteration until the stop condition is hit. The number of masked tokens at each iteration follows a linear decay function utilized in [19], i.e.,

$$|y^m| = \left\lfloor |y| \cdot \frac{T-t}{T} \right\rfloor, \tag{16}$$

where $\lfloor \cdot \rfloor$ indicates the floor function, and T is the upper bound of the iteration times while t indicates the number of the current iteration. We set $T = 10$ over all tasks, therefore after the initial iteration when all positions are predicted, we will then mask $90\%, 80\%, \ldots, 10\%$ tokens in following iterations. And for each iteration, we only update the probabilities of masked tokens while keeping the probabilities of unmasked tokens unchanged. As for the stop condition, the final result is obtained either when the upper bound of iterations is reached, or the obtained target sequence do not change between two consecutive iterations. In Table I we provide an illustration of the decoding process of our model, where the underlined words indicate the masked words in the next iteration.

IV. EXPERIMENTS

We mainly conduct experiments on neural machine translation to validate our framework. We evaluate the proposed AB-Net in Section IV-B, and we provide detailed analyses on the proposed adapter selection module in Section IV-D. We also explore the autoregressive variant of AB-Net in Section IV-C, followed with ablation studies in Section IV-E.

A. Experimental Setup

Datasets. We evaluate our framework on benchmark datasets including IWSLT14 German → English (IWSLT14 De-En),[1] WMT14 English ↔ German translation (WMT14 En-De/De-En)[2], and WMT16 Romanian → English (WMT16 Ro-En).[3]

[1]https://wit3.fbk.eu/
[2]https://www.statmt.org/wmt14/translation-task
[3]https://www.statmt.org/wmt16/translation-task

We further show the generality of our method on several low-resource datasets including IWSLT14 English ↔ Italian/Spanish/Dutch (IWSLT14 En ↔ It/Es/Nl). We additionally consider WMT14 English → French translation (WMT14 En-Fr) for autoregressive decoding. We follow the dataset configurations of previous works strictly. For IWSLT14 tasks, we adopt the official split of train/valid/test sets. For WMT14 tasks, we utilize `newstest2013` and `newstest2014` as the validation and test set respectively. For WMT16 tasks, we use `newsdev2016` and `newstest2016` as the validation and test set. For autoregressive decoding, we consider WMT16 Ro-En augmented with back translation data[4] to keep consistency with baselines [14].

Model Configurations. We mainly build our framework on `bert-base` models ($n_{\text{layers}} = 12$, $n_{\text{heads}} = 12$, $d_{\text{hidden}} = 768$, $d_{\text{FFN}} = 3072$). Specifically, for English we use `bert-base-uncased` on IWSLT14 and `bert-base-cased` on WMT tasks. We use `bert-base-german-cased` for German and `bert-base-multilingual-cased` for all other languages. We tokenize and segment each word into wordpiece tokens with the internal preprocessing code in BERT[5] using the same vocabulary as pre-trained BERT models, resulting in vocabularies with 30 k tokens for each language. It is worth noting that the dictionary of `bert-base-multilingual-cased` is much larger than other pre-trained models as it consists of the common tokens among 104 languages. For each low-resource language considered in our experiments, directly loading the whole embedding matrix of the multilingual BERT model will waste a lot of GPU memory. Therefore we only consider tokens that appear in the training and validation set, and manually modify the checkpoint of the multilingual BERT to omit the embeddings of unused tokens. In this way, we obtain dictionaries that contain 24 k/16 k/17 k/16 k tokens for Ro/It/Es/Nl respectively, which ultimately save around 77 M parameters in average. When extending to autoregressive decoding, we utilize `bert-large-cased` ($n_{\text{layers}} = 24$, $n_{\text{heads}} = 16$, $d_{\text{hidden}} = 1024$, $d_{\text{FFN}} = 4096$) for English to keep consistency with [14]. For adapters, on the encoder side, we set the hidden dimension between two FFN layers as $d_{\text{Aenc}} = 2048$ for WMT tasks and 512 for IWSLT14 tasks. On the decoder side, the hidden dimension of the cross-attention module is set equal to the hidden dimension of BERT models, i.e., $d_{\text{Adec}} = 768$ for `bert-base` models and $d_{\text{Adec}} = 1024$ for `bert-large` models. For the hyper-parameters w.r.t adapter selection, we apply an anneal strategy to μ and η in Equation (15) which control the weights of the KL-divergence loss and the layer number loss, i.e., the training depends more on the original prediction loss function at the early stage, and then progressively considers the layer selection process. The values of μ and η both start from 0 and are limited to 0.1 after 10 k training steps. We train our framework on 1/8 Nvidia 1080Ti GPUs for IWSLT14/WMT tasks, and it takes 1/7 days to finish the training.

Baselines. We denote the proposed framework as AB-Net, and to make a fair comparison with baseline models, we also consider a variant of our model that only incorporates BERT on the source-side with encoder adapter layers and denote it as AB-Net-Enc. We apply the adapter selection on the decoder side for main results, as the decoder is more crucial for the inference speed. We set $K = 6$ in Equation (14) to encourage the model to select and prune half of the inserted adapters, and we denote this variant as AB-Net w/ ADEC Selection. In Section IV-E, we will conduct a more thorough study regarding the performance of selecting adapters on the encoder or the decoder side. With parallel decoding, we consider Mask-Predict [19] as the backbone training and inference algorithm, based on which we re-implement BERT-Fused [14] and take it as the main baseline, denoted as BERT-Fused NAT. With autoregressive decoding where BERT is utilized only on the source-side, we compare our framework with BERT-Fused [14], BERT-Distilled [13] and CT-NMT [12] with their reported scores.

Inference and Evaluation. For parallel decoding, we utilize sequence-level knowledge distillation [31] on the training set of WMT14 En-De/De-En tasks, to keep consistency with [19]. This technique has been proved by previous non-autoregressive models that it can produce less noisy and more deterministic training data [37]. We use the raw training data for all other tasks. While inference, we generate multiple translation candidates by taking the top B length predictions into consideration, and select the translation with the highest probability as the final result. We set $B = 4$ for all tasks. And the upper bound of iterative decoding is set to 10. For autoregressive decoding, we use beam search with width 5 for all tasks. We utilize BLEU scores [60] as the evaluation metric. Specifically, we use `multi-bleu.perl` and report the tokenized case-insensitive scores for IWSLT14 tasks and tokenized case-sensitive scores for WMT tasks.

B. Results

The results of the proposed AB-Net with parallel decoding are listed in Table II, where "$*$" indicates the results obtained by our implementation, "/" indicates the corresponding result is not provided. The autoregressive Transformer model with `base` configuration [9] is also compared as a baseline. In addition to BLEU scores, we also report the per-sentence decoding latency on the `newstest2014` test set as well as the number of trained parameters on the WMT14 En-De task. As can be observed from Table II, with parallel decoding, Mask-Predict achieves considerable inference speedup but also suffers from performance degradation at the same time. Equipped with pre-trained BERT models from both sides, our framework obtains a huge performance promotion compared with Mask-Predict. In addition, we also outperform the autoregressive baseline Transformer-Base by a firm margin with similar scales of trainable parameters, while achieving 2.38 times speedup regarding the inference speed. Compared with BERT-Fused NAT [14] which utilizes BERT only on the encoder side, our framework as well as the BERT-encoder-only variant AB-Net-Enc both achieve better performance with less parameters to train, illustrating that the introduced adapter modules are able to leverage more information in a more efficient way.

As for the proposed adapter selection method, with utilizing half of the decoder adapters ($K = 6$ versus 12 layers in `bert-base` models), the decoding speed of the model is

[4]http://data.statmt.org/rsennrich/wmt16_backtranslations/ro-en

[5]https://github.com/huggingface/transformers/blob/master/src/transformers/tokenization_bert.py

further accelerated by 20% with slight drop of performance between 0.66% and 3.10%. Comparing with the autoregressive baseline, AB-Net with decoder adapter selection still achieves better performance on all tasks with around 3 times speedup of the inference speed except WMT14 En-De. As the pre-trained BERT model on English is more powerful than that on German, the decoder adapters are more crucial for the WMT14 En-De task, and thus selecting and pruning them will lead to more performance drop than other translation tasks.

Regarding the scale of trained parameters, we can notice that AB-Net-Enc actually introduces more parameters than AB-Net which utilizes BERT from both sides. The reason lies in the embedding layer. By incorporating BERT through adapters, the proposed framework gets rid of training the giant embedding layer which usually introduces ˜15 M parameters to train on each side if embeddings are not shared. Comparing with BERT-Fused NAT [14], our framework is able to save ˜26% parameters while incorporating information from both sides, providing a more cost-effective solution for leveraging pre-trained models based on adapters. Equipped with the proposed adapter selection method, the introduced parameters can be further pruned by 32% while inference, which also promotes the efficiency of the proposed framework.

Results on Low-Resource Language Pairs We also study the performance of AB-Net on three low-resource language pairs in the IWSLT14 dataset. Results on both directions are shown in Table III. The proposed AB-Net consistently outperforms the compared baselines among various language pairs, demonstrating the generality of our method.

C. Exploration on Autoregressive Decoding

Here we explore the application of AB-Net on autoregressive decoding. As the bidirectional and conditional independent nature of BERT prevents it from being applied to autoregressive decoding, to show the flexibility of the proposed framework, we directly use AB-Net-Enc as the autoregressive variant, whose encoder is initialized with the source-side BERT model and equipped with encoder adapter layers, while the decoder is an autoregressive Transformer Decoder. We compare our model with three fine-tuning baselines including BERT-Fused [14], BERT-Distilled [13] and CT-NMT [12]. Results are shown in Table IV, where we directly copy the best results reported in baseline papers. And Transformer-Big indicates Transformer with `transformer-big` configuration. Our framework outperforms the Transformer-Big baseline over all three translation tasks, with improvements from 1.33 to 2.75 BLEU scores. And we achieve comparable performance with the state-of-the-art baseline BERT-Fused [14].

D. Analyses on Adapter Selection

We provide detailed analyses on the proposed adapter selection method in this section. We conduct experiments on the IWSLT14 De-En task.

Selection on AENC *and* ADEC. As we only conduct adapter selection on the decoder side for main results, here we explore the alternatives of adapter selection on the encoder side as well as the decoder side. Results are shown in Table V, where we set $K = 6$ both for the encoder and the decoder adapter selection, and Δ indicates the change of BLEU scores of model variants compared with AB-Net. We can observe that conducting encoder adapter selection reduces the translation performance more harshly than decoder adapter selection, possibly due to the error propagation from the encoder to the decoder. Recently some works also point out that the final performance of NAT models relies more on the encoder than the decoder [43]. Therefore, in our framework, we only conduct decoder adapter selection for main results.

TABLE V
THE RESULTS OF SELECTING AENC AND ADEC ON THE TEST SET OF IWSLT14 DE-EN

Model Variants	BLEU	Δ
AB-Net	36.49	-
+ ADEC Selection	36.25	-0.24
+ AENC Selection	35.18	-1.31
+ AENC and ADEC Selection	35.01	-1.48
+ ADEC on top 6 layers	34.60	-1.89
+ AENC and ADEC on top 6 layers	33.78	-2.71

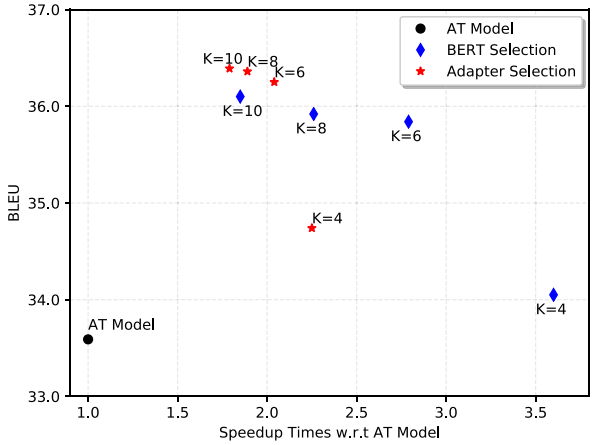


Fig. 2. The results of adapter layer selection and BERT layer selection when varying the number of target layers K. The x-axis indicates the inference speedup w.r.t the autoregressive model, and the y-axis indicates the BLEU score of translation results, which are all evaluated on the test set of the IWSLT14 De-En translation task.

We also compare the proposed adapter selection method with heuristic selection policy, i.e., only inserting adapters to the top 6 layers of the pre-trained BERT model. Results are shown in the bottom half of Table V, and large performance gaps can be observed between the heuristic methods and the proposed layer selection methods, which demonstrates the effectiveness of our method.

Effect of K. We vary K in Equation (14) to study the effect of the number of selected layers. Results are shown in Figure 2. Clearly, increasing K promotes the translation performance, but also slows the decoding speed. Setting $K = 4$ drastically decreases the performance, indicating that a small K will cause the loss of necessary information. Choosing a $K \geq 6$ results in similar performance, and we choose $K = 6$ for main results for a trade-off between the translation accuracy and the decoding speed.

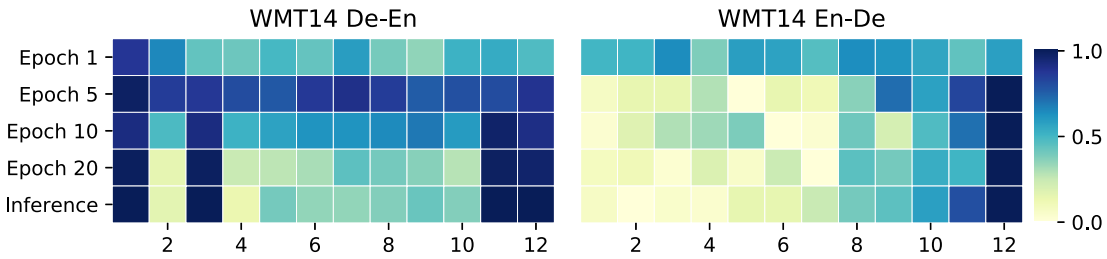


Fig. 3. An illustration of the decoder adapter layer selection samples at different epochs while training as well as while inference. We provide samples for the WMT14 En-De/De-En translation tasks. The x-axis indicates different layers and the y-axis indicates different stages.

Selection of BERT Layers. Instead of only selecting adapter layers, it is straightforward to extend our method to also selecting BERT layers, by re-writing Equation (10) as:

$$H^D_{l+1} = \begin{cases} \text{ADEC}(\text{YBERT}(H^D_l), H^E, H^E), & \text{if } z_l = 1 \\ H^D_l. & \text{if } z_l = 0 \end{cases} \quad (17)$$

Results on selecting BERT layers are shown in Figure 2. When BERT layers are also selected and pruned during inference, the decoding speed of our model can be further improved with slightly sacrificing the performance. Specifically, by only selecting and utilizing 4 BERT layers as well as adapter layers, the proposed framework achieves over 3.5 times inference speedup and comparable translation performance with the autoregressive baseline. In this way, we verify the effectiveness and the generality of the proposed adapter selection method, which provides solutions for both effectiveness preferred and efficiency preferred scenarios.

Analysis of the Latent Variable z. In Figure 3, we provide a visual illustration of the learned latent variable z which controls adapter layer selections. We can observe that as the training proceeds, the latent variable successfully learns certain adapter layer selection policy, and the model follows the learned policy in inference. Interestingly, in different tasks, the model has various preferences on which layers to select. For example, on WMT14 En-De the model tends to skip the adapters on bottom layers but utilizes adapters on top layers, which indicates that the early hidden representations may be less important and do not need the transformation by adapters. While on WMT14 De-En, the model prefers adapters on both bottom and top layers but skip them on middle layers. The different preferences illustrate that the proposed method can be generally applied to different tasks, and the learned selection policies may provide guidance for analyses on the interpretability of deep Transformer models, and we leave that for future work.

E. Ablation Study on AB-Net

In this subsection, we further conduct ablation studies regarding the scale of adapters, the proposed different components, different fine-tuning strategies and baselines with back-translation. Experiments are conducted on the IWSLT14 De-En dataset with parallel decoding.

Ablations on the Scale of Adapters. We investigate the influence of the scale of adapters in Figure 4. Specifically, we fix the scale of the decoder adapter and tune the hidden dimension of the encoder adapter d_{Aenc} in a wide range (2^6 to 2^{10}). We also plot the number of trained parameters on the encoder side in our framework and in the Transformer-Base model to make a comparison. From Figure 4, we find that our framework is robust to the scale of adapters, e.g., halving the dimension from 2^9 to 2^8 only results in a drop of 0.4 BLEU score. Compared with the autoregressive Transformer baseline, our framework is able to achieve better performance with only 5% parameters to train (getting 34.81 score when $d_{\text{Aenc}} = 64$), illustrating the efficiency of adapters.

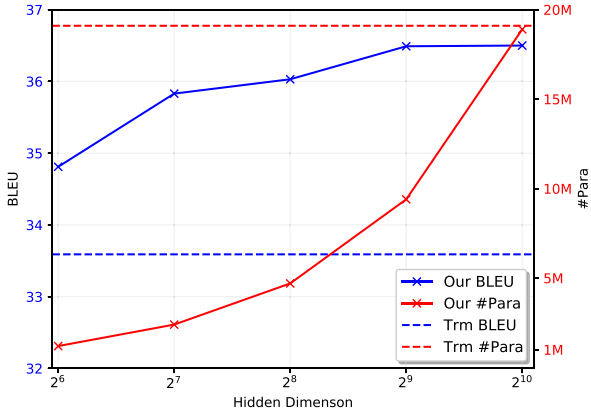
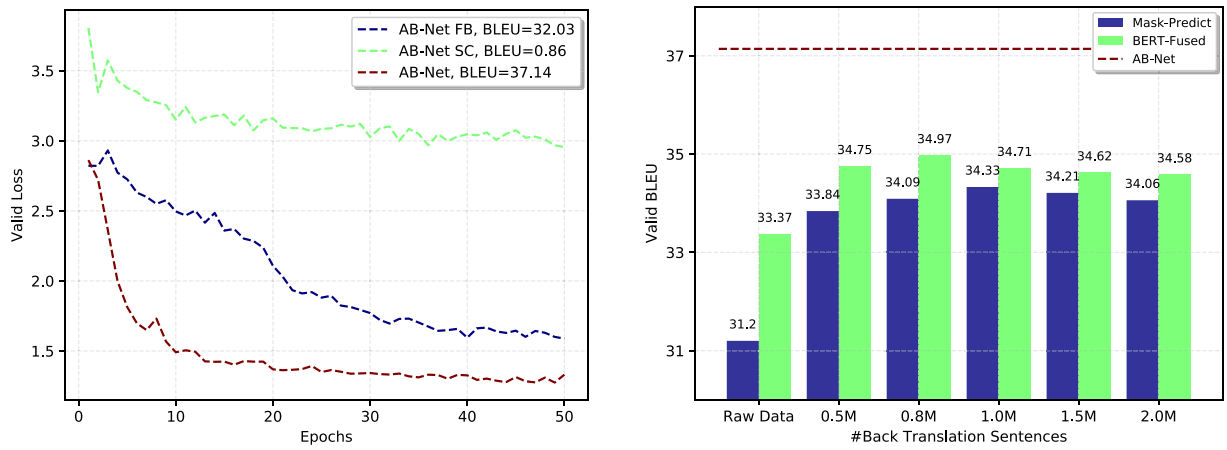


Fig. 4. The study on the scale of encoder adapters. Blue lines with the left y-axis indicate BLEU scores while red lines with the right y-axis indicate the number of parameters to train on the encoder side. Trm indicates the Transformer-Base model. Best view in color.

TABLE VI
THE ABLATION STUDY ON DIFFERENT COMPONENTS OF THE PROPOSED MODEL CONDUCTED ON THE TEST SET OF IWSLT14 DE-EN

Model Variants	BLEU
Transformer-Base	33.59
(1): XBERT + Decoder	×
(2): (1) + AENC	34.45
(3): (2) + YBERT	×
(4): (3) + ADEC	36.49

Ablations on the Different Proposed Components. In Table VI, we study the influence of different components in our framework including the pre-trained BERT models (XBERT and YBERT) and adapter layers (AENC and ADEC), where "Decoder" indicates a traditional Transformer decoder and × indicates the setting with no convergence reached during training. We can find that without utilizing adapters on either side, the model cannot converge during training, indicating the necessity of the adapter modules.



(a) (b)

Fig. 5. (a): Results of different fine-tuning strategies. (b): Results of baselines trained with extra monolingual data via back-translation. All settings are evaluated on the validation set of the IWSLT14 De-En task.

Comparison With Different Fine-Tuning Strategies. In addition to the proposed model which freezes the BERT components and only tunes the adapters while training, we also consider the variant that fine-tunes the full model in AB-Net (AB-Net FB), or trains AB-Net from scratch (AB-Net SC). We train all variants for 50 epochs and evaluate on the validation set of IWSLT14 De-En. Results are shown in Figure 5 a, where AB-Net converges significantly faster than AB-Net FB, and AB-Net SC does not converge. When fine-tuning the full model, more GPU memory is required because more gradient states need to be stored, therefore we have to halve the batchsize to fit the model into GPUs, which slows down the training process. With the same batchsize, AB-Net saves 29% GPU memory and 26% wall-clock training time compared with AB-Net FB. Moreover, we find that directly fine-tuning BERT is very unstable and sensitive to the learning rate, while only tuning the adapters alleviates this problem and is relatively more robust.

Comparison With Back-Translation. Back-translation is a simple yet effective data augmentation method in NMT [61], [62]. While we leverage BERT models which are pre-trained with extra monolingual data, we also consider baselines trained with extra monolingual data via back-translation to construct fair comparisons. Specifically, we first train an AT model on the IWSLT14 En-De task (with a 28.96 BLEU score), and then use it to generate additional training pairs on the English Wikipedia data, which is a subset of the training corpus of BERT. Results are shown in Figure 5b. We can find that the gains brought by back-translation are limited, and adding over 1 M monolingual data samples actually brings a performance drop. In addition, comparing with our method, back-translation requires training another model and decoding a large amount of monolingual data, which is time consuming.

V. Conclusion

In this paper, we propose a new paradigm of incorporating BERT into text generation tasks with a sequence-to-sequence framework. We initialize the encoder/decoder with a pre-trained BERT model on the source/target side, and insert adapter layers into each BERT layer. While fine-tuning on downstream tasks, we freeze the BERT models and only train the adapters. Instead of utilizing all inserted adapters in each BERT layer, we propose an adapter layer selection method to let the model automatically determine which adapter to use,thus accelerating the inference speed with little loss of performance. We build our framework on a parallel decoding method named Mask-Predict to match the bidirectional and conditional independent nature of BERT, and extend it to traditional autoregressive decoding in a straightforward way. The proposed framework is flexible and efficient, and achieves strong performance on neural machine translation while doubling the decoding speed of the Transformer baseline. In addition, the framework avoids the catastrophic forgetting problem and is robust when fine-tuning pre-trained language models.

In the future, our framework can be extended from two perspectives. Firstly, we will try to combine two different pre-trained models together in our framework, such as a BERT encoder and a GPT/XLNet decoder, to explore more possibilities on autoregressive decoding. Secondly, it is interesting to extend the proposed adapter layer selection method to multilingual or multi-domain scenarios, such as multilingual neural machine translation. Instead of introducing an adapter for each language/domain, we can encourage the model to utilize a small number of adapters and share the adapters among different languages/domains, and therefore achieve the reduction of the parameter scale with little loss of performance.

References

[1] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proc. 2018 Conf. North Amer. Ch. Asso. Comput. Linguist.: Hum. Langu. Technol.*, vol. 1, Jun. 2018, pp 2227–2237.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018, [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper.pdf

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, " Bert:Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Ch. Asso. Comput. Linguist.: Hum. Langu. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.

[4] Z. Yang *et al.*, "Generalized autoregressive pretraining for language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, Jun. 2019, pp. 5754–5764.

[5] T. B. Brown. *et al.*, "Language models are few-shot learners," *Adv. Neural Informat. Proce. Syst.*, vol. 33, pp. 1877–1901, 2020.

[6] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *Proc. EMNLP Work. Blackbox NLP: Anal. Inter. Neural Netw. NLP*, Nov. 2018, pp. 353–355.

[7] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100000 questions for machine comprehension of text," in *Proc. Conf. Empirical Methods Nat. Lang. Proc.*, Nov. 2016, pp. 2383–2392.

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *3rd Int. Conf. Learn. Represent.*, 2015.

[9] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Jun. 2017, pp. 5998–6008.

[10] R. Nallapati *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," in *Proc. 20th SIGNLL Conf. Comput. Nat. Lang. Learn.*, Aug. 2016, pp. 280–290.

[11] R. Weng, H. Yu, S. Huang, S. Cheng, and W. Luo, "Acquiring knowledge from pre-trained model to neural machine translation," *Thirty-Fourth AAAI Conf. Art. Intel.*, pp. 9266–9273, 2020.

[12] J. Yang *et al.*, "Towards making the most of bert in neural machine translation," *Thirty-Fourth AAAI Conf. Art. Intel.*, pp. 9378–9385, 2020.

[13] Y.-C. Chen, Z. Gan, Y. Cheng, J. Liu, and J. Liu, "Distilling the knowledge of bert for text generation," *Proc. 58th Ann. Meet. Asso. Comput. Linguist.*, pp. 7893–7905, Jul. 2020.

[14] J. Zhu *et al.*, "Incorporating bert into neural machine translation," *8th Inter. Conf. Learn. Represent.*, Feb. 2020, *arXiv:2002.06823*.

[15] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychol. Learn. Motivation*. Elsevier, Jan. 1989, vol. 24, pp. 109–165.

[16] C. Lee, K. Cho, and W. Kang, "Mixout: Effective regularization to finetune large-scale pretrained language models," *8th Inter. Conf. Learn. Represent. ICLR*, 2020 Addis Ababa, Ethiopia, Apr. 2020, *arXiv:1909.11299*.

[17] J. Guo, Z. Zhang, L. Xu, H.-R. Wei, B. Chen, and E. Chen, "Incorporating bert into parallel sequence decoding with adapters," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 10843–10854, Oct. 2020.

[18] A. Bapna, N. Arivazhagan, and O. Firat, "Simple, scalable adaptation for neural machine translation," in *Proc. 2019 Conf. Empirical Methods in Natural Lang. Proc. 9th Int. Joint Conf. Nat. Lang. Proc. (EMNLP-IJCNLP)*, Nov. 2019, pp. 1538–1548.

[19] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.* , Apr. 2019, pp. 6114–6123.

[20] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *8th Int. Conf. Learn. Represent.*, 2020, *arXiv:1909.11556*.

[21] M. A. Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," *Proc. 5th Work. Repres. Learn. NLP*, pp. 143–155, Feb. 2020, *arXiv:2002.08307*.

[22] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *5th Int. Conf. Learn. Repres.*, 2017, *arXiv:1611.01144*.

[23] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, Apr. 2017.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, Feb. 2019.

[25] L. Dong *et al.*, "Unified language model pre-training for natural language understanding and generation," in *Proc. Adv. Neural Inf. Process. Syst.*, May 2019, pp. 13042–13054 .

[26] Y. Liu *et al.*, "Roberta: A. robustly optimized bert pretraining approach," Jul. 2019, *arXiv:1907.11692*.

[27] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *Adv. Neural Info. Proc. Syst.*, pp. 7057–7067, 2019, *arXiv:1901.07291*.

[28] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," in *Proc. 36th Int. Conf. Machine Learn.*, vol. 97, 2019, pp. 5926–5936.

[29] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. 58th Ann. Meet. Asso. Comput. Linguist.*, 2020, pp. 7871–7880.

[30] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," Mar. 2015, *arXiv:1503.02531*.

[31] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *Proc. Conf. Empirical Methods Nat. Lang. Proc.*, Jun. 2016, pp. 1317–1327.

[32] S. Rothe, S. Narayan, and A. Severyn, "Leveraging pre-trained checkpoints for sequence generation tasks," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 264–280, Jun. 2020.

[33] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *Adv. Neural Inf. Process. Syst.*, May 2017, pp. 506–516.

[34] N. Houlsby *et al.*, "Parameter-efficient transfer learning for NLP," in *Proc. 36th Inter. Conf. Machine Learn.*, vol. 97, 2019, pp. 2790–2799.

[35] R. Wang *et al.*, "K-adapter: Infusing knowledge into pre-trained models with adapters," Feb. 2020, *arXiv:2002.01808*.

[36] B. Ji, Z. Zhang, X. Duan, M. Zhang, B. Chen, and W. Luo, "Cross-lingual pre-training based transfer for zero-shot neural machine translation," in *Proc. 34th AAAI Conf. Art. Intell.*, Apr. 2020, pp. 115–122.

[37] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," *6th Inter. Conf. Learn. Represent.*, 2018, *arXiv:1711.02281*.

[38] J. Guo, X. Tan, D. He, T. Qin, L. Xu, and T.-Y. Liu, "Non-autoregressive neural machine translation with enhanced decoder input," in *Proc. 33rd AAAI Conf. Artif. Intell.*, Jul. 2019, pp. 3723–3730.

[39] Z. Sun, Z. Li, H. Wang, D. He, Z. Lin, and Z. Deng, "Fast structured decoding for sequence models," in *Adv. Neural Inf. Process. Syst.*, Oct. 2019, pp. 3011–3020.

[40] J. Guo, X. Tan, L. Xu, T. Qin, E. Chen, and T.-Y. Liu, "Fine-tuning by curriculum learning for non-autoregressive neural machine translation," in *Proc. 34th AAAI Conf. Art. Intell.*, Apr. 2020, pp. 7839–7846.

[41] M. Stern, W. Chan, J. Kiros, and J. Uszkoreit, "Insertion transformer: Flexible sequence generation via insertion operations," in *Proc. 36th Inter. Conf. Machine Learn.*, vol. 97, 2019, pp. 5976–5985.

[42] J. Gu, C. Wang, and J. Zhao, "Levenshtein transformer," in *Adv. Neural Inf. Process. Syst.*, 2019, pp. 11179–11189 .

[43] J. Guo, L. Xu, and E. Chen, "Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 376–385.

[44] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Adv. Neural Inf. Process. Syst.*, vol. 2, pp. 598–605, 1989.

[45] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *5th Inter. Conf. Learn. Represent.*, 2017, *arXiv:1608.08710*.

[46] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Adv. Neural Inf. Process. Syst.*, Aug. 2016, vol. 29, pp. 2074–2082.

[47] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1389–1397.

[48] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "CondenseNet: An efficient densenet using learned group convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognition*, Oct. 2018, pp. 2752–2761.

[49] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *7th Int. Conf. Learn. Represent.*, 2019, *arXiv:1810.05270*.

[50] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," in *Proc. 57th Conf. Asso. Comput. Linguist., ACL*, vol. 1, Florence, Italy, Jul. 2019, pp. 5797–5808.

[51] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *INTERSPEECH 20th Ann. Conf. Int. Speech Commun. Assoc.*, pp. 66–70, 2019.

[52] X. Li, A. C. Stickland, Y. Tang, and X. Kong, "Deep transformers with latent depth," in *Proc. Conf. Empirical Methods Nat. Lang. Proc.: Findings*, 2020, pp. 4163–4174.

[53] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin, "Distilling task-specific knowledge from bert into simple neural networks," Mar. 2019, *arXiv:1903.12136*.

[54] X. Jiao *et al.*, "Tinybert: Distilling bert for natural language understanding," in *Proc. Conf. Empirical Methods Nat. Lang. Process.: Findings*, 2020, pp. 4163–4174.

[55] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter," Oct. 2019, *arXiv:1910.01108*.

[56] L. Hou, Z. Huang, L. Shang, X. Jiang, X. Chen, and Q. Liu, "Dynabert: Dynamic bert with adaptive width and depth," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 9782–9793, Apr. 2020.
[57] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," Jul. 2016, *arXiv:1607.06450*.
[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[59] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *2nd Inter. Conf. Learn. Represent.*, 2014, *arXiv:1312.6114*.
[60] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2002, pp. 311–318.
[61] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," *Proc. Conf. Empirical Methods Nat. Lang. Proc.*, pp. 489–500, 2018.
[62] Z. Zhang, S. Liu, M. Li, M. Zhou, and E. Chen, "Joint training for neural machine translation models with monolingual data," *Proc. Thirty-Second AAAI Conf. Arti. Intel.*, 2018.

Junliang Guo received the B.E. degree in 2016 in computer science from the University of Science and Technology of China, Hefei, China, where he is currently working toward the Ph.D. degree with the School of Computer Science and Technology, under the advisory of Professor Linli Xu. He has authored or coauthored several papers in refereed conference proceedings, such as NeurIPS, ACL, AAAI, IJCAI and ICDM. His research interests include machine learning, specifically in sequence modeling and representation learning, and their applications in natural language processing and heterogeneous types of data, such as networks.

Zhirui Zhang received the Ph.D. degree from the University of Science and Technology (USTC), in 2019, supervised by Prof. Enhong Chen from USTC and Prof. Harry Shum from Microsoft Research Asia. He is currently an Algorithm Expert with Language Technology Lab, Alibaba DAMO Academy. He has authored or coauthored more than Ten papers in refereed confereces and journals, including ACL, EMNLP, NAACL, NeurIPS, AAAI and TASLP. His general research interests include natural language processing, machine learning and reinforcement learning, specifically in machine translation, dialogue systems, and natural language generation.

Linli Xu received the B.Sc. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2002 and the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 2007. She is currently a Professor with the School of Computer Science, University of Science and Technology of China. She has authored or coauthored more than 30 papers at top-tier journals and conferences. Her general research interests include machine learning, with research interests on unsupervised learning and semi-supervised learning, multi-task learning and transfer learning, neural methods and applications, and optimization. She is a Member of the Anhui Province Key Laboratory of Big Data Analysis and Application. She was the recipient of the Best Overall Paper Honorable Mention of the 26th International Conference on Machine Learning in 2009.

Boxing Chen received the Ph.D. degree from the Chinese Academy of Science, Beijing, China. He is currently a Senior Staff Algorithm Engineer with the Machine Intelligence Lab, DAMO Academy of Alibaba Group. Prior to Alibaba, he was a Research Officer with National Research Council Canada, Ottawa, ON, Canada, a Senior Research Fellow with the Institute for Infocomm Research, Singapore, a Postdoc with FBK in Italy, and a Postdoc with the University of Grenoble, Grenoble, France. He has coauthored more than 60 papers in the NLP conferences and journals. He was the recipient of The Best Paper Award from MT Summit 2013 and The Best Paper Award Nomination from ACL 2013. He was an Area Chair for ACL, EMNLP and NLPCC. His team ranked the first place more than 20 times in various MT competitions, such as WMT2018, WMT 2017, NIST2012 OpenMT, IWSLT2007, and IWSLT2005.

Enhong Chen (Senior Member, IEEE) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China. He is currently a Professor and the Vice Dean of the School of Computer Science, University of Science and Technology of China. He has authored or coauthored more than 100 papers in refereed conferences and journals, including IEEE Trans. KDE, IEEE Trans. MC, KDD, ICDM, NIPS, and CIKM. His general research interests include data mining and machine learning, social network analysis, and recommender systems. He was on program committees of numerous conferences, including KDD, ICDM, SDM. He was the recipient of the Best Application Paper Award on KDD-2008, the Best Research Paper Award on ICDM-2011, SDM-2015 and KDD-2018. His research is supported by the National Science Foundation for Distinguished Young Scholars of China.