Adapters in Machine Translation

Aditya Kurniawan
Charles University, MFF UFAL
akurniawan.cs@gmail.com

Abstract

Pre-trained language models received extensive attention in recent years. However, it is still challenging to incorporate a pre-trained model such as BERT into Natural Language Generation tasks. In this work, we conduct a study of adapters in Machine Translation as an alternative for naive fine-tuning. We show that with proper initialization, adapters can help achieve better performance than training models from scratch while training substantially fewer weights than the original model. We further show that even randomly set weights used as the base models for fine-tuning we can achieve a similar performance to one of the baseline models, bypassing the need to train hundreds of millions of weights in the pre-training phase.

1 Introduction

Pre-trained language models (Devlin et al., 2019; Howard and Ruder, 2018) have received extensive attention in recent years. These models are trained on a large scale corpus and then fine-tuned for a particular downstream task. This method allows pre-trained models to perform well on various tasks related to natural language processing. One of the most successful models is BERT (Devlin et al., 2019). BERT has been most extensively used for common Natural Language Understanding (NLU) tasks. It has been shown that BERT can achieve great performance with relatively straightforward fine-tuning.

While fine-tuning BERT is relatively easy for classification-like tasks, it is still very challenging for natural language generation. According to Zhu et al. (2020), simply incorporating BERT into the encoder side of the seq2seq architecture can hurt the performance. On the decoder side, it is quite challenging to incorporate BERT as the bidirectional nature of the model was significantly different from the conditional language model (predicting the next word) objective we are aiming for machine translation task.

Fine-tuning all BERT's parameters is inefficient considering the number of parameters contained in a single model of BERT. Naive fine-tuning also often results in catastrophic forgetting, where the models forget the previous knowledge they have acquired while improving on the new domain (McCloskey and Cohen, 1989; Yogatama et al., 2019). This may explain why it is considered harmful to simply fine-tune an initialized encoder component with BERT. It is also known that large pre-trained language models are unstable and fragile on small datasets.

Adapters are an alternative approach that allows for fine-tuning a model without altering the original network (Houlsby et al., 2019; Bapna and Firat, 2019). By leveraging adapters, one can reduce the number of parameters in fine-tuning and make the process computationally less expensive while still achieving similar results. Another property of the approach with adapters that is quite useful is that they are more robust against catastrophic forgetting than fine-tuning (Han et al., 2021).

In this work, we propose to use Transformer architecture with BERT as the pre-train weights in both encoder and decoder components in Machine Translation tasks and use adapters during fine-tuning. We separate the experiments into four different areas:

- Use BERT weights[1] as the pre-train weights (`Pre-trained BERT`)
- Use Transformer architecture with BERT configuration and pre-train the models with MLM objective on IWSLT and WMT data (`Pre-trained Transformer`)

[1] We use publicly available BERT weights from Huggingface hub https://huggingface.co

- Use Transformer architecture with BERT configuration and fully random weights as the pre-train weights (`Pre-trained random`)
- Use Transformer architecture with BERT weights as the pre-train weights where the weights are shuffled (`Pre-trained shuffled`)

The following are the contributions and findings from our work:

- We show that compared to the baseline models where we trained Transformer architectures from scratch, we managed to get better results as much as 20% in `Pre-trained BERT`.
- We show that extending the volume of training data on baseline models does not always lead to performance improvements. We argue this is due to the fact that it is not straightforward to train big models such as BERT with relatively small data compared to BERT-like size data.
- We compare the result of `Pre-trained Transformers` and baseline models of the same architecture and same overall training data but no fine-tuning phase. The baseline models are trained with the combination of WMT and IWSLT data in machine translation objectives. `Pre-trained Transformers` uses the same combination of data, but they are used on the pre-training setup and later fine-tuned with IWSLT data. In contrast to the baseline models, `Pre-trained Transformers` leads to 7% improvement in BLEU score when the pre-training dataset size is increased.
- We show that we can achieve quite competitive performance on `Pre-trained random` and `Pre-trained shuffled` compared to the baseline model that is trained with 2 million pairs of sentences. We found this interesting as we only fine-tuned the adapters weight and the cross-attention layers while keeping the rest of the parameters intact.

2 Previous Works

Pre-trained language models such as the work by Devlin et al. (2019) has become the backbone of many state-of-the-art natural language processing (NLP) models. In transfer-learning, the language model is first trained on a large corpus that covers broad domain and text varieties, and then fine-tuned to a specific application (Howard and Ruder, 2018).

Adapters were proposed by Houlsby et al. (2019) as an alternative to fine-tuning. It is often assumed that an adapter has a perfect memory as the original model's parameters are unchanged. Furthermore, adapters also improve the fine-tuning speed by training significantly fewer parameters than fine-tuning. The most common setup is appending new adapter modules between the layers of a pre-trained model. There are two different adapter placements as proposed by Bapna and Firat (2019) and Houlsby et al. (2019). The former leverages the adapters by incorporating them in two different parts of the sub-layers. The latter only appends the adapters on top of each layer with layer normalization added within the adapter architecture. As of this writing, there are no direct comparisons between these two techniques. However, the work of Bapna and Firat (2019) is simpler to implement and has been adopted in other works (Pfeiffer et al., 2020; Rücklé et al., 2020; Pfeiffer et al., 2021).

There have been several applications of adapters in NLP. Bapna and Firat (2019) show the benefit of using adapters when adapting machine translation models to new domains and languages; Houlsby et al. (2019) proposed to use adapters in various natural language understanding (NLU) domains such as commonsense reasoning, sentiment analysis, and natural language inference (NLI) Pfeiffer et al. (2021); Pfeiffer et al. (2020) show that adapters can be extended to work efficiently in a multi-language setup where they are incorporated with new languages in multi-lingual models such as Devlin et al. (2019).

Adapters in Machine Translation The work on adapters in machine translation was led by Bapna and Firat (2019) where they introduced experiments incorporating adapters to fine-tune a pre-trained Transformer. This work has shown that adapters give comparable or better results when compared with the standard full fine-tuning or bilingual baselines in different pairs setup without any hyperparameter tuning in various dataset sizes and model capacities. From this success, many works have started to incorporate BERT with Adapter to machine translation. BERT has proven to be chal-

lenging to incorporate in machine translation setup due to the different nature of pre-training objective Guo et al. (2021). It is further exacerbated as the number of parameters in BERT is enormous, causing the fine-tuning process to become brittle and unstable. Guo et al. (2021) attempted to incorporate adapters into BERT architecture and managed to achieve strong performance in contrast to the auto-regressive baseline.

3 Methodology

3.1 Base model architecture

We use the Transformer architecture (Vaswani et al., 2017) as our base model. In addition to that, we also use BERT (Devlin et al., 2019) configuration throughout our experiments. This includes our baseline models as well as our pre-trained models. We employ BERT in both encoder and decoder with an extra cross-attention layer randomly initialized in the decoder component. All the implementations are built on top of Huggingface framework (Wolf et al., 2020).

3.2 Dataset

We conduct the experiments with two different datasets IWSLT2014 German→English and WMT19 German→English. The IWSLT2014 is used as the main dataset to fine-tune and evaluate, while we treat WMT19 as an additional dataset to increase the training data for Masked-Language Model (MLM). Furthermore, there is a domain difference between WMT (mainly news) and IWSLT (mainly TED talks), which the pre-training techniques also help to bridge.

3.3 Pre-trained models

The pre-trained models are used for further fine-tuning with adapters in IWSLT data. There are two types of pre-trained models in this work. We use publicly available BERT models and as well as train our own pre-trained models.

BERT We build our models by combining two publicly available configurations as well as the pre-trained models package from Huggingface Hub. For English, we use `bert-base-uncased` and for German we use `bert-base-german-dbmdz-uncased`. Both configurations use the same model architecture and settings ($n_{layers} = 12, n_{heads} = 12, d_{hidden} = 768, d_{FFN} = 3072$).

MLM Pre-training Other than just leveraging the pre-trained models, we also experiment with our own pre-training models. Specifically, we train the BERT-like models from scratch for both languages with different dataset configurations. The pre-training are done with Masked Language Model (MLM) objective from BERT pre-training framework for both languages. Here is a summary of our pre-training setups:

- Pre-train both English and German models with only IWSLT14 dataset.
- Pre-train both English and German models with IWSLT14 dataset and a subset of WMT19 dataset with a total of 500k sentence pairs for each languages.
- Pre-train both English and German models with IWSLT14 dataset and a subset of WMT19 dataset with a total of 2 million sentence pairs for each languages.

3.4 Random and Shuffled BERT Pre-trained weights

To show to what extent the Adapter approach benefits from the exact pre-trained weights vs. some of their general distribution properties vs. just the network structure, we conduct experiments where we start by shuffling BERT weights. To perform the experiment, we separate the weights initialization into two approaches: 1) We shuffled the weights from a column perspective. This means in all the weights matrices in the BERT network, we shuffled the weights by preserving columns but shuffling their order. 2) We shuffled the weights from both column and row perspectives.

Furthermore, we also conduct experiments where randomly set weights on all base network layers as the pre-training models. During the fine-tuning, we only update the weights of the adapter and keep the rest of the weights intact.

3.5 Adapters

We follow the adapter implementation from Pfeiffer et al. (2020). The adapter module is defined as a bottleneck architecture that is put on top of each transformer layer. The adapter A_l at layer l consists of a down-projector D_l followed by a non-linear activation $ReLU$ (Agarap, 2018), and an up-projection layer U_l. Let h_l be the output of current transformer layer l. It is first passed through a layer normalization (Ba et al., 2016) LN before

being transformed by a down-projection layer D_l of the adapter module. The output of up-projection is again forwarded through layer normalization.

4 Results and Discussions

4.1 Adapters Comparison

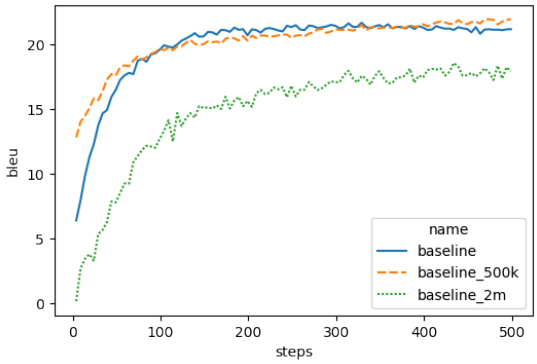

Figure 1: Comparison between baseline models trained with different size of datasets. `baseline` represents the model trained only using IWSLT; `baseline_500k` represents the model trained using IWSLT and WMT with total of 500k sentence pairs; `baseline_2m` represents the model trained using IWSLT and WMT with total of 2 million sentence pairs.

In this section, we compare the result of the baseline models with the models that are fine-tuned with adapters. From Figure 1, adding more data to the baseline models does not necessarily improve the performance. We suspect the models require more time to train to get better performance. There is a clear gap between `baseline_2m` and the rest of the baseline models. `baseline` and `baseline_500k` performs really well from the start while `baseline_2m` lag behind. We suspect this is the effect of including more sentences from different domains. `baseline_500k` is the best mix given the training time constraint. It provides a balance in between not overfitting in the correct domain and not too much data out of the domain. `baseline_2m` shows the impact on domain difference. It does not perform well on IWSLT but it is growing and has a chance to improve the performance further. At a later stage, the `baseline_2m` output would deserve manual evaluation, because the lower bleu may not necessarily reflect a lower quality, we can see some of the result in Table 2.

To see the impact of including adapters, we compare the result on different sizes of pre-training used for the base model. The base models are then fine-tuned with the adapters module on the IWSLT data. As we can see from Figure 2, BERT achieves the best performance from the earlier steps compared to the rest of the pre-training size. In contrast to the baseline models, we see the benefit of adding more sentences to the pre-training. We can see the performance progression between the model that was trained using 500k data has lower performance than the one using 2 million data. On the other hand, models that only use IWSLT as the pre-training data suffers from performance degradation in the middle of the fine-tuning. We observed that this is due to the gradient explosion on the cross-attention layer. The IWSLT model eventually managed to achieve a similar performance to the 500k model in the later steps.

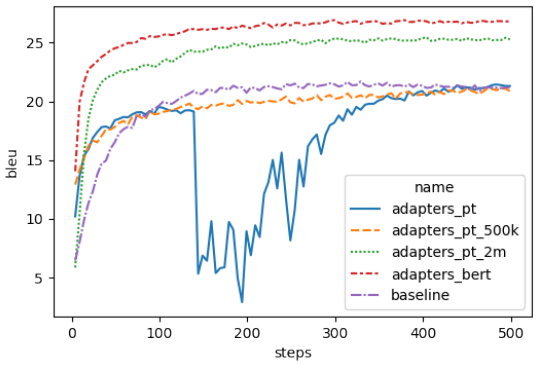

Figure 2: Comparison between adapters pre-trained with different size of datasets. `baseline` represents the model trained only using IWSLT; `adapters_pt` represents the model pre-trained only using IWSLT; `adapters_pt_500k` represents the model pre-trained using IWSLT and WMT with total of 500k sentence pairs; `adapters_pt_2m` represents the model pre-trained using IWSLT and WMT with total of 2 million sentence pairs; `adapters_bert` represents the model that uses BERT weights.

4.2 Random and Shuffled Pre-training Weights

We can see from Figure 3 that the performance of the model that uses random weights as the pre-training model is more stable than the one using shuffled BERT weights. Both of the shuffled BERT models suffer from gradient explosion similar to the IWSLT model we show in the previous section. Although the performance of the random model is still below the baseline model, it is interesting to see that only fine-tuning adapters and the cross-attention layer manage to achieve a reasonable BLEU score, considering that the pre-training

Random Weights + Adapters
input: wir tanzen im tempel und werden zu gott. & quot ;
prediction:we & apos ; re going to be able to become god. & quot ;
input: aber gleichzeitig hatten sie eine klare kenntnis des waldes, die erstaunlich war.
prediction:but at the same time, they had a clear of the audience who was amazing..
input: es ist so wunderbar. ihr musst es beschutzen. & quot ;
prediction:it & apos ; s wonderful. you have to protect it. & quot ;

Table 1: Prediction results from randomly set pre-trained model fine-tuned with adapters

models do not contain any meaningful information.

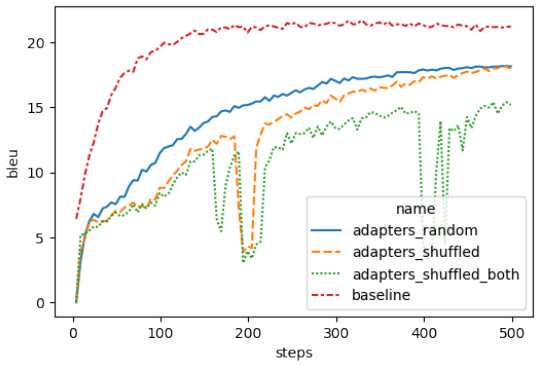

Figure 3: Comparison between adapters using shuffled BERT and random weights as the pre-trained models. `baseline` represents the model trained only using IWSLT; `adapters_random` represents the model pre-trained only using random weights; `adapters_shuffled` represents the model pre-trained using column-wise shuffled BERT model; `adapters_shuffled_both` represents the model pre-trained using shuffled BERT model.

4.3 Random Pretraining vs. Out-of-Domain Data

We further analyze the random weights by comparing the result with the best performing, baseline, and transformer models we pre-trained ourselves. From Figure 4, we can see that the performance of the random model achieves a similar result to the baseline model that uses 2 million training sentence pairs. While the performance is still far from the baseline that is trained using only IWSLT data, this result shows the base model's structure helps the adapter achieve a meaningful performance with very small weights required for the fine-tuning.

We perform a quick check to the output of the model in Table 1. We can see from the first two lines that the model hash difficulty to capture complex phrases. On the first row, the model missed **tanzen im tempel** which means **dancing in the temple**. For the second row, the model confuses **knowledge of the forest** and output **audience instead**. Furthermore, the model also does not translate the word **kenntnis** and makes the translation unclear since the object of the sentence is missing. Despite those mistakes, the model still can capture simple sentences as shown on the third row. Another observation that we noticed in the generated output is the tokenization of `quot;` and `amp;`. Instead of being treated as a single token, the tokenizer treats the token as three different subword token. We notice that this is due to the inavailability of the aforementioned token in the pre-trained BERT vocabulary.

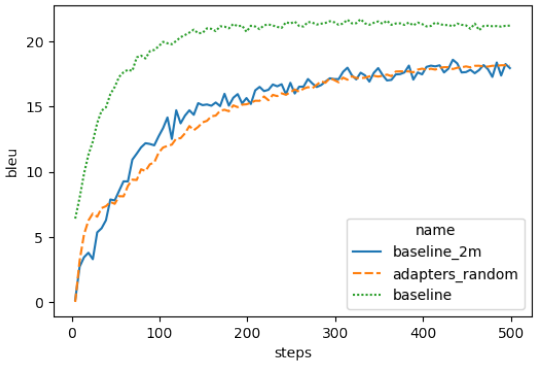

Figure 4: Comparison between pre-trained random weights and baseline model. `baseline` represents the model trained only using IWSLT; `baseline_2m` represents the baseline model trained with a combination of IWSLT and WMT sentence pairs; `adapters_random` represents the model pre-trained only using random weights; `adapters_pt_2m` represents the model pre-trained using IWSLT and WMT with total of 2 million sentence pairs; `adapters_bert` represents the model that uses BERT weights

4.4 Qualitative Comparison

We perform a sanity check to compare the generated results on some of our models. This sanity check is to check the errors produced by the mod-

Baseline IWSLT	IWSLT + WMT (total 2m)	BERT + Adapters
input: erinnerst du dich an den patienten mit dem gereizten rachen? **prediction**: do you remember reading to the patients? on the	**input**: erinnerst du dich an den patienten mit dem gereizten rachen? **prediction**: do you remember the patient with the tingling revenge?	**input**: erinnerst du dich an den patienten mit dem gereizten rachen? **prediction**: remember the patient with the bruised remorse?
input: großartig, sagte ich. legte auf. **prediction**: great, i said. got up..	**input**: großartig, sagte ich. legte auf. **prediction**: great, i said. put on..	**input**: großartig, sagte ich. legte auf. **prediction**: great, i said. put it down.
input: - - aber in unserer entdeckung der welt, haben wir alle arten unterschiedlicher methoden. **prediction**: but in our discovery of the world, we & apos ; ve got all sorts of different	**input**: - - aber in unserer entdeckung der welt, haben wir alle arten unterschiedlicher methoden. **prediction**: - - but in our discovery of the world, we have all kinds of different methods.	**input**: - - aber in unserer entdeckung der welt, haben wir alle arten unterschiedlicher methoden. **prediction**: but in our discovery of the world, we have all sorts of different ways of doing things.

Table 2: Prediction results from 1) Baseline model trained with only IWSLT data; 2) Pre-trained model with adapters where we pre-train the model with IWSLT and WMT with a total of 2 million pre-training data; 3) BERT with adapters.

els on different techniques. From Table 2, we can see for the first example none of the models managed to generate the correct result. However, BERT + adapters and 2 million pre-trained base models manage to generate the proper context where the result is still discussing **the patient**. The wrong part is when the model generates an incorrect translation regarding the patient's disease. The second example shows that the BERT + adapters create the correct and better output than the other models. The final example shows that BERT + adapters generate an interesting output where it manages to remove unimportant characters such as −− and produce readable output. There may be a slightly different opinion on this example as the 2 million pre-trained base model generate a more concise output.

5 Conclusion

In this work, we show that with the right initialization, adapters can help achieve better performance than training models from scratch while only training far fewer weights than the original model. We further show that even with random fixed weights in the main part of the model, the adapters and cross-attention can recover and achieve a similar performance to one of the baseline models where we do not need to train hundreds of millions of weights. Although the random pre-trained weights result shows an interesting result, further experiments may be required to understand the limitation of the adapters in this setup.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Junliang Guo, Zhirui Zhang, Linli Xu, Boxing Chen, and Enhong Chen. 2021. Adaptive adapters: An efficient way to incorporate bert into neural ma-

chine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1740–1751.

Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 854–861, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

M. McCloskey and N. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, N. Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. *ArXiv*, abs/2010.11918.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome T. Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, A. Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and P. Blunsom. 2019. Learning and evaluating general linguistic intelligence. *ArXiv*, abs/1901.11373.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wen gang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating bert into neural machine translation. *ArXiv*, abs/2002.06823.