

# Pluggable Neural Machine Translation Models via Memory-augmented Adapters

Yuzhuang Xu<sup>1,†</sup>, Shuo Wang<sup>1,†</sup>, Peng Li<sup>2,\*</sup>, Xuebo Liu<sup>3</sup>, Xiaolong Wang<sup>1</sup>, Weidong Liu<sup>1,4</sup>, Yang Liu<sup>1,2,\*</sup>

<sup>1</sup>Department of Computer Science & Technology, Tsinghua University, Beijing, China

<sup>2</sup>Institute for AI Industry Research, Tsinghua University, Beijing, China

<sup>3</sup>Harbin Institute of Technology, Shenzhen, China

<sup>4</sup>Zhongguancun Laboratory, Beijing, China

{xyz21thu, wangshuo.thu}@gmail.com, lipeng@air.tsinghua.edu.cn,  
liuyang2011@tsinghua.edu.cn

**Abstract**—Although neural machine translation (NMT) models perform well in the general domain, it remains rather challenging to control their generation behavior to satisfy the requirement of different users. Given the expensive training cost and the data scarcity challenge of learning a new model from scratch for each user requirement, we propose a memory-augmented adapter to steer pretrained NMT models in a pluggable manner. Specifically, we construct a multi-granular memory based on the user-provided text samples and propose a new adapter architecture to combine the model representations and the retrieved results. We also propose a training strategy using memory dropout to reduce spurious dependencies between the NMT model and the memory. We validate our approach on both style- and domain-specific experiments and the results indicate that our method can outperform several representative pluggable baselines.

**Index Terms**—Neural machine translation, style customization, domain customization, pluggable, memory, adapter.

## I. INTRODUCTION

IN recent years, modern neural machine translation (NMT) [1] systems are often developed with large-scale parallel data extracted from the Web [2], [3], whose style and content are driven by the average distribution of data from many domains [4]. Therefore, the performance of strong NMT models is close to or even better than human translators in the general domain [5], [6].

However, MT customers may have some special requirements, including both style- and domain-specific individual demands [7], [8]. For instance, some users may want translations in a special style, while some others may need to translate medical texts. These requirements can be quite diverse among different customers and retraining or fine-tuning the model for each user group entails significant development costs. Moreover, the customers can not always provide sufficient data to train strong NMT models.

Fortunately, *pluggable* methods [9]–[11] bring hope to handle the aforementioned user requirements, which employ additional modules to steer pretrained models. As shown in Figure 1, the users can provide some text samples for the NMT model to imitate. We will then learn a plugin to control the

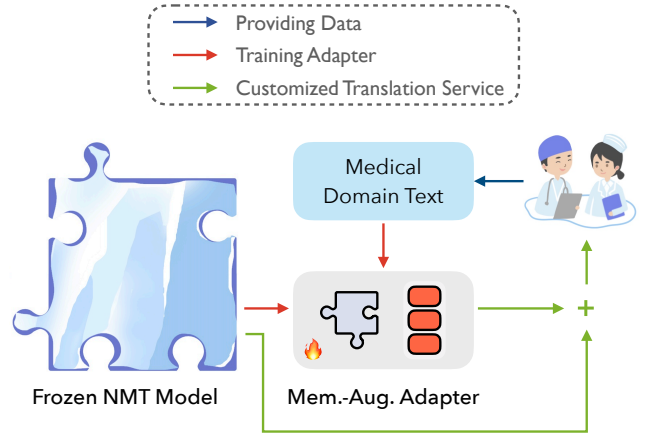


Fig. 1. A frozen and pluggable NMT model using memory-augmented plugins. For each user group with special requirements, we can develop a plugin for them without affecting other users.

NMT model to satisfy the user demands without optimizing the parameters in the original model. An advantage of plug-and-play approaches is that we can maintain the performance of the pretrained model, alleviating the risk of catastrophic forgetting [12].

Some researchers advocate using lightweight parametric modules as plugins to control pretrained language models [13]–[18]. For machine translation, we can also use a series of parametric modules to adjust the model behavior to satisfy various user demands. However, recent studies find that there exists a performance bottleneck of fully-parametric pluggable methods [14], [17], [19]: increasing the number of trainable parameters can not always lead to better performance. Inspired by the recent progress of retrieval-augmented models [11], [20]–[23], we propose to increase the expressive power [17] of parametric plugins through external memories, the resulting method is called *memory-augmented adapter*.

The main challenges of the memory-augmented adapter are two-fold: (1) how to *construct* memories that can provide useful customization information; and (2) how to *integrate* the memories into existing NMT models while not degrading the translation quality. Although long phrases can provide

<sup>†</sup>Equal contribution

<sup>\*</sup>Corresponding authors

more contextualized information, matching long sequences between queries and memory items is more difficult than matching shorter ones. We propose to build multi-granular memories to balance the amount of contextualized information and the retrieval difficulty. Different from many previous works [11], [21], [22] that encode the source sentence and the target prefix as the key and the next token as the value, our memory can provide multi-scale translation knowledge [24] that is suitable for queries coming from different layers of the NMT model [25]. For memory integration, we propose a *new adapter architecture* to better interpolate the original model representation and the retrieved vectors. Moreover, we propose a new training strategy with memory dropout to reduce spurious dependencies between the NMT model and the provided memory. We conduct experiments for both style- and domain-related customizations and the results show the superiority of our method over many representative baselines.

## II. RELATED WORK

### A. Style and Domain Adaptation for NMT

Adapting NMT models to translate texts of a specific style or domain has been investigated in several previous works [26]–[28]. For stylized NMT, many previous works focus on the formality control of NMT models [29], [30], of which the style has a clear definition. Most existing works need to train a specific model for each style. For instance, Niu and Carpuat [31] mix the training data of both style transfer and machine translation to learn a formality-sensitive NMT model. Given that the user-provided styles can be of great diversity in the real world, we aim to satisfy different style demands in a pluggable manner.

For domain adaptation, Luong and Manning [26] propose an effective method that fine-tunes an out-of-domain model with small-sized in-domain supervised corpora. Hu et al. [32] further design an unsupervised method, since parallel data is hardly available in many domains. Zheng et al. [33] extend  $k$ NN-MT [22] to perform unsupervised domain adaptation. Our work is different from Zheng et al. in both memory design and usage and the experiments show that our proposed framework performs not only better but also faster than their approach.

### B. Machine Translation Customization

Machine translation customization aims to satisfy the special requirements of different users. Vu and Moschitti [4] propose to select data that is similar to the user-provided text samples and then train or fine-tune an NMT model for the corresponding user. Following Michel and Neubig [7], we believe that MT customization has some specific traits that distinguish it from common style and domain adaptation settings: (1) The number of customization requirements is very large due to the personal variation among different MT system users; (2) The available data is often very limited (even monolingual, let alone parallel) for each customization requirement. Therefore, we propose to leverage pluggable methods to customize existing NMT models.

### C. Pluggable Pretrained Models

Pluggable methods aim to control the generation behavior of pretrained models without optimizing model parameters [10], [34], [35], which can effectively avoid catastrophic forgetting. Some works propose to use parametric plugins. Bapna and Firat [14]; Houlsby et al. [13] insert some adapters between pretrained layers and Li and Liang [17] prepend some trainable vectors before the hidden states in attention modules. Hu et al. [36] leverage learnable pairs of rank decomposition matrices to steer pretrained models. Retrieval-augmented models can also be treated as pluggable methods, which augment the model with non-parametric memory.  $k$ NN-MT [22] combines the model prediction and retrieval-based distribution at the output layer. Borgeaud et al. [37] build a chunk-level memory for language modeling. Chen et al. [38] encode questions and answers into key-value pairs for question answering. In this work, our goal is to combine the merits of both parametric and non-parametric plugins. We propose a new type of memory for machine translation, which explicitly considers the alignment between source and target phrases of different granularities.

## III. BACKGROUND

### A. Transformer Model

In order to better explain our proposed memory-augmented adapter, we first give a description of some components in the Transformer [1] model. Given the input sentence  $\mathbf{x}$ , Transformer maps it into vectors via an encoder:

$$\mathbf{E} = \text{encoder}(\mathbf{x}) \quad (1)$$

where  $\mathbf{E} \in \mathbb{R}^{|\mathbf{x}| \times d}$  and  $d$  is the hidden size of the model. The encoder output is then utilized by the decoder, which is a stack of several independent layers. We use  $\mathbf{D}^{(i)}$  to denote the output of the  $i$ -th decoder layer. Specifically, each decoder layer firstly employs a self-attention module to model the dependency between the target-side words:

$$\begin{aligned} \mathbf{S}^{(i)} &= \text{attn}(\mathbf{D}^{(i-1)}, \mathbf{D}^{(i-1)}, \mathbf{D}^{(i-1)}) \\ \mathbf{L}_1^{(i)} &= \text{layernorm}(\mathbf{D}^{(i-1)} + \mathbf{S}^{(i)}) \end{aligned} \quad (2)$$

where  $\text{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$  is the multi-head attention and  $\text{layernorm}$  is the layer normalization.

After that, a cross-attention module is adopted to integrate the source-side information:

$$\begin{aligned} \mathbf{C}^{(i)} &= \text{attn}(\mathbf{L}_1^{(i)}, \mathbf{E}, \mathbf{E}) \\ \mathbf{L}_2^{(i)} &= \text{layernorm}(\mathbf{L}_1^{(i)} + \mathbf{C}^{(i)}) \end{aligned} \quad (3)$$

The output of the cross-attention module is then projected with a feed-forward layer. The decoder output is finally used to estimate the probability  $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$ , where  $\mathbf{y}$  is the target sentence and  $\boldsymbol{\theta}$  denotes the set of model parameters.

### B. Style Customization in NMT

Similar to generating images with specific styles [39], [40], style customization in NMT means outputting translations with user-specified styles [7], [41]. For example, we want to output translations in Shakespeare style in a Zh-En translation task. A simple example is as follows.

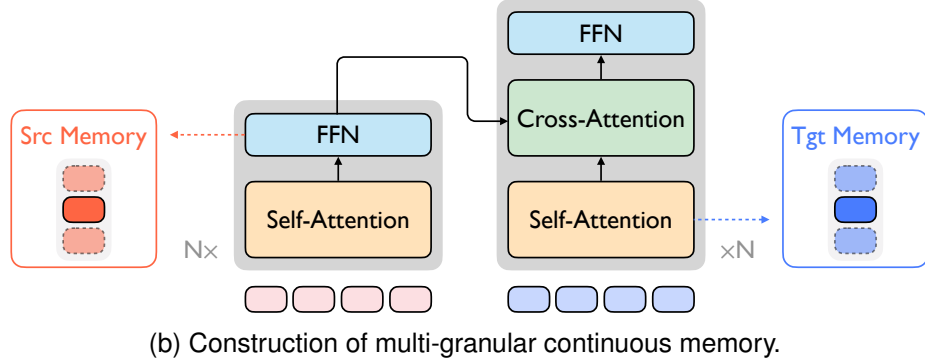
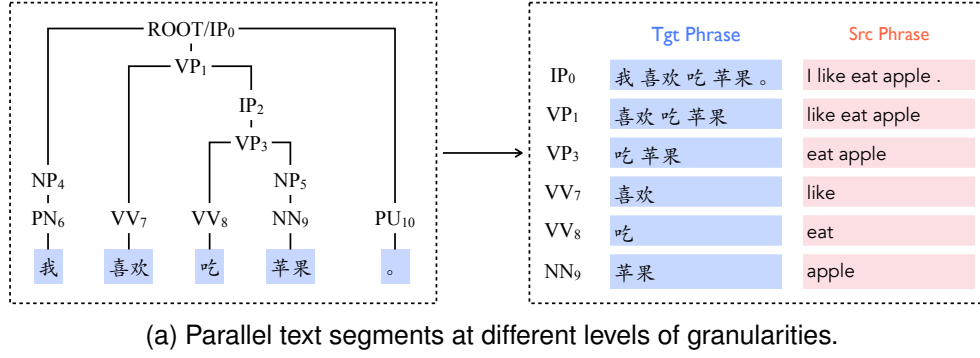


Fig. 2. We leverage parse trees to obtain multi-granular phrases. Each monolingual phrase is then translated by NMT models. For each phrase pair, we perform a forward computation in the teacher-forcing manner and record some intermediate representations into the memory. See Section IV-B for more details.

**Zh:** 哦上帝啊，请赐予我力量吧！  
**En (G):** Oh God, please grant me strength!  
**En (S):** Oh Lord, do thou endow me with thy might!

“**En (G)**” denotes the output of vanilla translation model, and “**En (S)**” denotes the output of style-customized translation model using Shakespeare corpus. The underlined expression are typical representatives of Shakespeare style.

The task most closely related to style customization in NMT is author-stylized rewriting [41], [42], which aims to rewrite a given text in the style of a specific author. Syed et al. sum up the user or author style into three levels, namely surface level, lexical level, and syntactic level styles [41]. These levels capture subtle differences in punctuation, word usage, and even sentence construction unique to individual authors, thereby making author-stylized rewriting a challenging task. Style customization in NMT not only shares the same challenges as author-stylized rewriting, but it must also simultaneously translate the provided text into the target language, presenting its own unique challenges.

#### IV. APPROACH

##### A. Overview

In this work, we aim to let MT system users be able to control existing NMT models by simply providing some examples. To this end, we propose a memory-augmented adapter to help NMT models imitate the user-provided text samples. Specifically, we propose the multi-granular continuous

memory that can better leverage multi-scale patterns, which have proven to be important for machine translation [24]. We also propose a new type of adapter (i.e., memory-augmented adapter) to integrate external memory into NMT models. We will explain how to construct and utilize the memory in the following two subsections, respectively.

##### B. Multi-granular Continuous Memory

We expect our memory to not only extract essential information from user-provided data but also be easy to retrieve for NMT models. For the first desideratum, we propose to build the memory with parallel phrase pairs, which can reflect the translation pattern required by the customer. However, it is non-trivial to determine the granularity of the used phrases. Storing only short phrases may waste a lot of contextualized information while storing too much long sequences would make it difficult to match the query and the memory items. To address this issue, we propose to construct a multi-granular memory to balance the amount of contextualized information and the retrieval difficulty. Multi-granular information is also shown to be important for NMT models [24]. As shown in Figure 2, we use parse trees to extract multi-granular phrases, which can identify more meaningful boundaries than random splitting. The extracted phrases are then translated by NMT models to form parallel phrase pairs.

For the second desideratum, we propose to use the same model to build and utilize the memory. For each phrase pair, we perform a forward computation to get the continuous representation at each layer of the involved NMT model. We

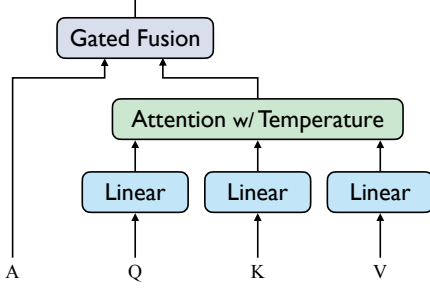


Fig. 3. Architecture of the memory-augmented adapter. Please refer to Section IV-C for more details.

store the encoder output  $\mathbf{E}$  as the source-side memory and the self-attention output  $\mathbf{S}^{(i)}$  at every decoder layer as the target-side memory. See Eq. (1) and (2) for more details of the stored representations. Each memory item is averaged among the representations of all tokens in a phrase, whose size is  $d$ . Figure 2 shows an example. The reason we extract  $\mathbf{E}$  and  $\mathbf{S}^{(i)}$  as our memory is that these representations are at the same layer where we perform memory retrieval. Our motivation is to narrow down the gap between the memory items and the queries, making it easier for the model to read the memory.

We focus on using monolingual user-provided data in this work since parallel data is often unavailable for most requirements. However, our method can be easily extended for bilingual data, from which we can automatically extract phrase pairs based on unsupervised word alignment algorithms [43], [44].

### C. Memory-augmented Adapter

a) *Adapter Architecture*: Different from existing parametric plugins [19] that only adapt model representations, we propose a new type of adapter to read memory. The proposed memory-augmented adapter has four inputs: anchor, query, key, and value, which can be represented as  $\mathbf{A}$ ,  $\mathbf{Q}$ ,  $\mathbf{K}$ , and  $\mathbf{V}$ , respectively. Anchors and queries are derived from the frozen NMT model while keys and values come from the memory. As depicted in Figure 3, we use an attention module to generate the retrieved result:

$$\mathbf{R} = \text{softmax}(\mathbf{Q}\mathbf{W}_q\mathbf{W}_k^\top\mathbf{K}^\top/T)\mathbf{V}\mathbf{W}_v \quad (4)$$

where  $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$  and the retrieved result  $\mathbf{R}$  has the same shape with  $\mathbf{Q}$ .  $T$  is a hyperparameter to control the sharpness of the retrieval distribution. To avoid the model being completely dependent on the retrieved result  $\mathbf{R}$  that can be erroneous in some cases, we also take in an anchor from the original model, which is combined with  $\mathbf{R}$  via a gated fusion module:

$$\begin{aligned} \lambda &= \text{sigmoid}(\text{relu}([\mathbf{A}; \mathbf{R}]\mathbf{W}_1)\mathbf{W}_2) \\ \mathbf{O} &= \lambda\mathbf{A} + (1 - \lambda)\mathbf{R} \end{aligned} \quad (5)$$

where  $\mathbf{O}$  is the adapter output, which has the same shape as the anchor  $\mathbf{A}$ .  $\mathbf{W}_1 \in \mathbb{R}^{2d \times d}$  and  $\mathbf{W}_2 \in \mathbb{R}^{d \times 1}$ .  $\lambda$  is the learned interpolation ratio.

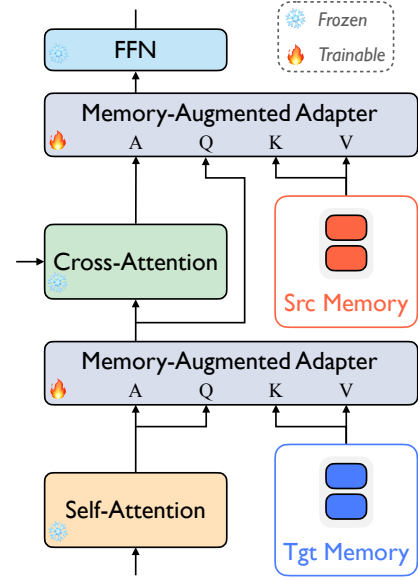


Fig. 4. Illustration of adapter integration. The original model parameters are frozen. See Section IV-C for details.

b) *Adapter Integration*: We apply the memory-augmented adapter to the self- and cross-attention modules in the decoder, since these two types of components are important for target-side language modeling and source-side information utilization, respectively. At the  $i$ -th decoder layer, we use the self-attention output  $\mathbf{S}^{(i)}$  as queries to read the target-side memory:

$$\mathbf{O}_1^{(i)} = \text{memadapt}(\mathbf{S}^{(i)}, \mathbf{S}^{(i)}, \mathbf{M}_t^{(i)}, \mathbf{M}_t^{(i)}) \quad (6)$$

where  $\text{memadapt}(\mathbf{A}, \mathbf{Q}, \mathbf{K}, \mathbf{V})$  denotes the memory-augmented adapter.  $\mathbf{M}_t^{(i)} \in \mathbb{R}^{N_t^{(i)} \times d}$  represents the target-side memory, where  $N_t^{(i)}$  denotes the number of items in  $\mathbf{M}_t^{(i)}$ . The adapter output  $\mathbf{O}_1$  is then provided to the layer normalization module:

$$\mathbf{L}_1^{(i)} = \text{layernorm}(\mathbf{D}^{(i-1)} + \mathbf{O}_1^{(i)}) \quad (7)$$

Similarly, we read the source-side memory in the cross-attention module:

$$\begin{aligned} \mathbf{O}_2^{(i)} &= \text{memadapt}(\mathbf{C}^{(i)}, \mathbf{L}_1^{(i)}, \mathbf{M}_s^{(i)}, \mathbf{M}_s^{(i)}) \\ \mathbf{L}_2^{(i)} &= \text{layernorm}(\mathbf{L}_1^{(i)} + \mathbf{O}_2^{(i)}) \end{aligned} \quad (8)$$

Figure 4 shows an example. To reduce the redundancy that a phrase pair would repeatedly appear in memories at every decoder layer, we split all the phrase pairs into  $L$  parts, where  $L$  is the number of decoder layers. Each layer only stores one part of phrase pairs. In other words, the memories used in different layers do not overlap with each other, in terms of the corresponding phrase pairs.

c) *Training Strategy*: Inspired by dropout [45] that can effectively reduce spurious co-adaptation between model parameters, we propose a memory dropout approach to prevent NMT models from being too dependent on some specific memory items. When training the memory-augmented adapter, we randomly drop part of the memory items. Let  $\mathbf{M}$  be the

full memory and  $\hat{\mathbf{M}}$  be the remained memory after memory dropout, the overall loss can be given by

$$\begin{aligned} \mathcal{L} = & \underbrace{\mathcal{L}_{\text{NLL}}(P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{M}))}_{\text{loss using full memory}} \\ & + \underbrace{\alpha \mathcal{L}_{\text{NLL}}(P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{M}}))}_{\text{loss using dropped memory}} \\ & + \underbrace{\beta \mathcal{L}_{\text{dist}}(P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{M}), P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{M}}))}_{\text{loss modeling the agreement}} \end{aligned} \quad (9)$$

where  $\alpha$  and  $\beta$  are hyperparameters.  $\mathcal{L}_{\text{NLL}}$  is the conventional negative log-likelihood. The agreement loss (i.e.,  $\mathcal{L}_{\text{dist}}$ ) [46] measures the distance between two distributions:

$$\mathcal{L}_{\text{dist}}(p, q) = \frac{1}{2}(D_{\text{KL}}(p||q) + D_{\text{KL}}(q||p)) \quad (10)$$

*d) Extension:* Since our method does not change the model decoding, it can also be combined with the retrieval-based decoding algorithm as presented in *k*NN-MT [22], which interpolates the model probability with a retrieved distribution. We call this decoding method *k*NN decoding. See Section A in appendix for details.

## V. STYLE CUSTOMIZATION

### A. Setup

*a) NMT Model Training:* In the pluggable scenario, we should first have an existing NMT model, which can serve as the foundation for further customization. We use the training corpus of the WMT20 En $\leftrightarrow$ Zh translation task<sup>1</sup> to train NMT models, which contains 23.9M sentence pairs. We use SentencePiece<sup>2</sup> to preprocess the data and the sentence piece model we used is released by mBART [2]. The architecture of our NMT models is Transformer [1], whose hidden size is 512 and depth is 6. Please refer to Section A in appendix for more details.

*b) Customization Data:* We evaluate the customization effect of our method in two translation directions: En-Zh and Zh-En. We use the works of two world-renowned writers as stylized text samples, including Shakespeare and Lu Xun. Their works created representative styles for English and Chinese, respectively. We extract their texts from the web and then split the data into training, validation, and test sets. For Shakespeare’s style, the training set contains 20K English sentences while the validation and test sets contain 500 sentences, respectively. The target-language (i.e., English) training and validation sentences are then translated by the NMT model, while the test set is translated by human translators. For Lu Xun’s style, the training set consists of 37K sentences while the validation and test sets contain 500 sentences. Similarly, the test set is also translated by humans while the training and validation sets are translated by NMT models. The resulting corpus is called Machine Translation with Style Customization (MTSC)<sup>3</sup>. We will add more styles in different languages for machine translation research in the future.

*c) Memory Construction:* We first build parse trees for target-side sentences using Stanford Parser<sup>4</sup> and then extract multi-granular phrases. As mentioned in Section IV-C, we evenly divide the extracted phrases according to their lengths into  $L$  parts to avoid information redundancy between different layers. We did not store the representations of phrases longer than a pre-specified threshold  $l_{\text{max}}$ , since the occurrence of long phrases is very low.  $l_{\text{max}}$  is set to 10 for Zh and 8 for En.

*d) Adapter Training:* The general NMT model is frozen when training the memory-augmented adapter. We determine the value of the hyperparameters based on the validation performance. Specifically, the temperature in Eq. (4) is set to 0.5. Both the  $\alpha$  and  $\beta$  in Eq. (9) are set to 5. The memory dropout rate is set to 0.1. We provide more details of adapter training in Section A in appendix.

*e) Baselines:* We compare our approach with the following representative baselines:

- *Extreme* [7]: adding style specific bias vector in the output layer.
- *Adapter* [13]: inserting adapters before residual connection.
- *MT+Rewrite* [41]: using a translation and a monolingual rewriting model.
- *kNN-MT* [22]: integrating the datastore based on stylized texts after the last decoder layer during inference.
- *DExperts* [35]: controlling the generation behavior using language models.

*f) Evaluation Metrics:* We use both automatic and human evaluation to make a thorough comparison between the involved methods. The automatic evaluation metrics are as follows:

- *BLEU*: measuring the translation quality of model outputs. We use sacreBLEU<sup>5</sup> [47] to estimate the BLEU score.
- *Perplexity*: measuring the fluency of model outputs. We fine-tune a pretrained Transformer LM [48] with stylized text to calculate perplexity.
- *Classifier Score*: measuring the similarity between model outputs and the stylized text samples. We follow Li et al. [49] to train style classifiers to quantify the style similarity. The classifier we used is TextCNN [50]. For Lu Xun’s style, the classifier can achieve an accuracy of 93.5%. For Shakespeare’s style, the classifier can achieve an accuracy of 94.5%. We use these classifiers to estimate whether the output is in the desired style.

### B. Main Results

*a) Automatic Evaluation:* Table I shows the performance of all the involved methods in the style customization task. When decoding with vanilla beam search, our method can outperform all the baselines in terms of BLEU and classifier score on average, indicating the effectiveness of the proposed

<sup>1</sup><https://www.statmt.org/wmt20/translation-task.html>

<sup>2</sup><https://github.com/google/sentencepiece>

<sup>3</sup>We will release MTSC shortly.

<sup>4</sup><https://nlp.stanford.edu/software/lex-parser.html>

<sup>5</sup>English-Chinese: nrefs:1 | case:mixed | eff:no | tok:zh | smooth:exp | version:2.3.1. Chinese-English: nrefs:1 | case:mixed | eff:no | tok:13a | smooth:exp | version:2.3.1.

TABLE I  
AUTOMATIC EVALUATION FOR STYLE CUSTOMIZATION. WE BOLD THE **best** SCORES AND HIGHLIGHT THE SECOND BEST SCORES.

Method	BLEU( $\uparrow$ )			Perplexity( $\downarrow$ )			Classifier Score( $\uparrow$ )		
	En-Zh	Zh-En	Avg.	En-Zh	Zh-En	Avg.	En-Zh	Zh-En	Avg.
Vanilla [1]	13.7	15.7	14.7	459.6	127.4	293.5	18.0	28.4	23.2
Extreme [7]	16.0	17.7	16.9	315.2	113.7	214.5	37.4	43.4	40.4
Adapter [13]	16.8	19.4	18.1	351.1	121.0	236.1	33.8	58.2	46.0
MT+Rewrite [41]	16.3	15.7	16.0	<u>222.4</u>	127.5	349.8	<u>47.0</u>	28.4	37.7
kNN-MT [22]	18.9	20.0	19.5	230.7	98.5	164.6	42.2	<u>70.4</u>	56.3
DExperts [35]	13.8	15.9	14.9	467.0	127.3	297.2	18.4	31.4	24.9
Memory-augmented Adapter	<u>20.8</u>	<u>21.1</u>	<u>21.0</u>	257.8	110.9	184.3	<u>47.0</u>	69.4	<u>58.2</u>
+ kNN decoding	<b>21.3</b>	<b>21.8</b>	<b>21.6</b>	<b>199.6</b>	<b>95.1</b>	<b>147.4</b>	<b>53.2</b>	<b>85.2</b>	<b>69.2</b>

TABLE II  
HUMAN EVALUATION FOR STYLE CUSTOMIZATION IN EN-ZH. THE COMPARISON IS PERFORMED BETWEEN kNN-MT AND OUR METHOD WITH VANILLA BEAM SEARCH. “WIN” MEANS OUR METHOD PERFORMS BETTER.  $\kappa$  DENOTES FLEISS’ KAPPA.

Human	Content Preservation			Sentence Fluency			Style Similarity		
	Win	Tie	Lose	Win	Tie	Lose	Win	Tie	Lose
Rator 1	64.0%	12.0%	24.0%	61.0%	9.0%	30.0%	69.0%	5.0%	26.0%
Rator 2	64.0%	13.0%	23.0%	57.0%	13.0%	30.0%	63.0%	11.0%	26.0%
Rator 3	67.0%	9.0%	24.0%	62.0%	5.0%	33.0%	71.0%	6.0%	23.0%
Avg.	65.0%	11.3%	23.7%	60.0%	9.0%	31.0%	67.7%	7.3%	25.0%
$\kappa$	0.476			0.446			0.656		

memory-augmented adapter in controlling the output style of NMT models. The perplexity of kNN-MT is better than ours, but its BLEU score is much worse. When combined with kNN decoding, which is illustrated in **Extension** in Section IV-C, our method can be further improved, achieving the best performance across all the three automatic metrics. These results re-demonstrate that our method is complementary to kNN-MT.

*b) Human Evaluation:* We also perform a human evaluation to assess the translation quality of different methods. We follow previous works [51], [52] to ask human evaluators to compare the outputs of different methods. Since human evaluation is time-consuming and labor-intensive, we only compare our method with the strongest baseline (i.e., kNN-MT) in En-Zh. Note that our outputs used for human evaluation are generated using vanilla beam search. Following Hu et al. [53], each sentence is evaluated in terms of content preservation, sentence fluency, and style similarity. Table II shows the results, from which we find our approach performs better than the baseline in all the three evaluation aspects. The agreement of the three human evaluators is estimated through Fleiss’ kappa [54] and the results demonstrate *moderate agreement* ( $0.4 \leq \kappa \leq 0.6$ ) in terms of both content preservation and sentence fluency and *good agreement* ( $0.6 \leq \kappa \leq 0.8$ ) regarding style similarity. Table VIII gives some translation examples for style customization.

### C. Performance at Different Data Scales

In some cases, the user-provided data can be of extremely small scale [7]. We thus investigate the performance of the involved methods using customization data of different scales. Figure 5 shows the results. Our memory-augmented adapter consistently outperforms the baselines at different data scales, even with only 250 exemplary sentences. These results show that our method can be applied to extremely low-resource adaptation scenarios.

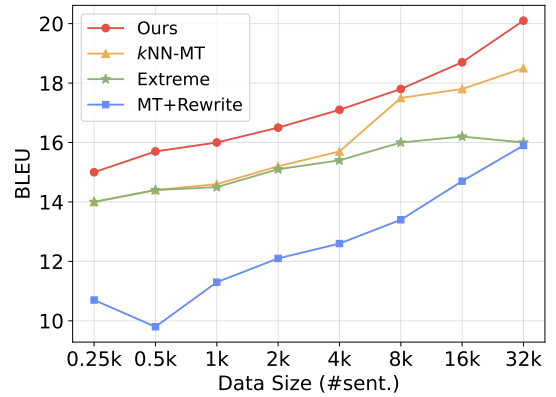


Fig. 5. Performance of style customization at different data scales. “Ours” does not use kNN decoding.

We conduct a thorough **ablation study** and report the results in Section VII-A.

## VI. DOMAIN CUSTOMIZATION

### A. Setup

*a) NMT Model Training:* We train the NMT model using the WMT14 De-En training corpus <sup>6</sup>, including 4.5M sentence pairs. The training data is preprocessed in the same way as style customization. See Section A in appendix for more details.

*b) Customization Data:* To evaluate the pluggable performance in the domain customization setting, we follow previous works [33], [55] to use a multi-domain dataset, which includes four domains: *IT*, *Medical*, *Law* and *Koran*. To simulate real-world user customization where the user-provided data is often of small scale, we randomly select 20K sentences for IT, Medical, and Law, and use all the 18K

<sup>6</sup><https://www.statmt.org/wmt14/translation-task.html>



TABLE III  
BLEU SCORES IN THE DOMAIN CUSTOMIZATION TASK. WE HIGHLIGHT THE **best** AND SECOND BEST SCORES.

Method	IT		Medical		Law		Koran		Avg.	
	valid	test	valid	test	valid	test	valid	test	valid	test
Vanilla	28.1	28.4	26.4	27.6	36.2	35.9	10.9	11.5	25.4	25.9
Adapter	<u>30.9</u>	30.5	26.8	28.0	36.0	35.6	12.9	13.5	26.7	26.9
$k$ NN-MT	28.8	29.2	<u>30.0</u>	<u>32.3</u>	<u>38.3</u>	<u>38.4</u>	14.6	15.1	27.9	28.8
Memory-augmented Adapter	<b>31.2</b>	<u>31.1</u>	<u>30.0</u>	32.0	<u>37.5</u>	37.3	<u>14.7</u>	<u>15.3</u>	<u>28.4</u>	<u>28.9</u>
+ $k$ NN decoding	30.5	<b>31.4</b>	<b>31.3</b>	<b>33.5</b>	<b>38.8</b>	<b>38.6</b>	<b>15.7</b>	<b>16.5</b>	<b>29.1</b>	<b>30.0</b>

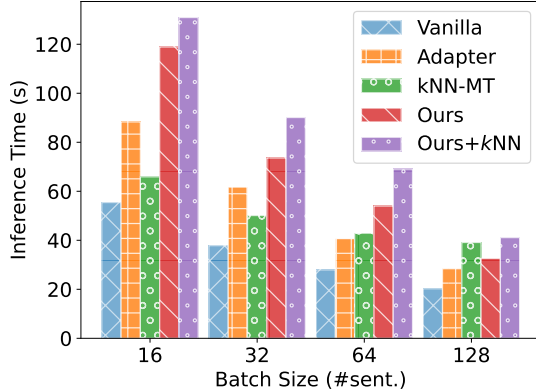


Fig. 6. Inference time reported on the IT domain. “Ours” is not combined with  $k$ NN decoding.

sentences for Koran. We also use only the target-side training data to simulate real-world cases and use NMT models to generate synthetic parallel data. All the validation and test sets are authentic parallel data.

*c) Memory Construction and Adapter Training:* We filter phrases longer than 10 during memory construction. For adapter training,  $T$  is set to 0.1 for Medical and Law, and 0.5 for the other two domains. Both  $\alpha$  and  $\beta$  are set to 5 on all the four domains. The memory dropout rate is set to 0.1.

*d) Baselines:* We compare our proposed method with two representative pluggable domain adaptation baselines: adapter [13] and  $k$ NN-MT [22]. See Section B for details on the baselines.

## B. Main Results

The adaptation performance on different domains is shown in Table III. On average, our method can outperform the two baselines even without  $k$ NN decoding, demonstrating the effectiveness of our motivation to boost parametric plugins with external memory. When combined with  $k$ NN decoding, our method can achieve better results on Medical, Law, and Koran. Using  $k$ NN decoding, our method can improve 3.1 and 1.2 BLEU scores over Adapter and  $k$ NN-MT on the test sets, respectively.

## C. Inference Time

A concern for retrieval-augmented methods is that they may significantly slow down the inference process. As shown in Figure 6, our method is slower than Adapter, but the difference between the two methods becomes very slight when using

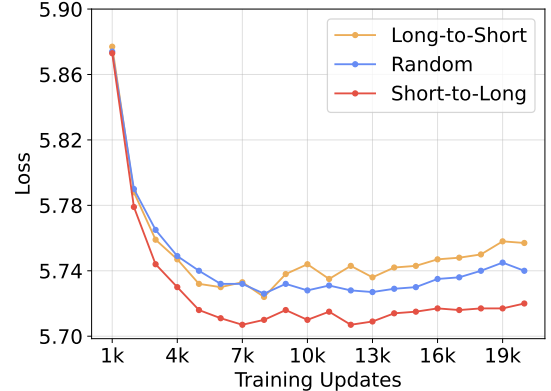


Fig. 7. Effect of granularity distribution across the decoder layers. Each curve denotes the validation loss in the En-Zh style customization task. We only use the vanilla NLL loss to rule out the effect of the training strategy. “Long-to-Short”: lower layers store the representations of longer phrases while higher layers store the representations of shorter layers. “Random”: each phrase pair is stored in a randomly selected layer. “Short-to-Long”: lower layers store shorter phrases while higher layers store longer phrases.

big batch sizes. For instance, our inference time is only 1.15 times that of Adapter with a batch size of 128. Our method is also comparable to  $k$ NN-MT. In particular, when the batch size is set to 128, our method is slightly faster than  $k$ NN-MT. When also using  $k$ NN decoding, our method is slightly slower than  $k$ NN-MT (i.e., 41.1s vs. 39.1s with batch size = 128). We implement the  $k$ NN algorithm with `Faiss-gpu` [56] to accelerate the retrieval process.

## VII. DISCUSSION

### A. Effect of Different Components

We conduct thorough ablation studies to better understand the effect of the proposed components in our method.

*a) Granularity Distribution:* As mentioned in Section IV-B, we divide all the phrase pairs into several different parts to reduce the redundancy of information among the decoder layers. Our basic idea is that a certain phrase pair only needs to appear in one layer of the decoder. The phrase pairs are divided according to their lengths. We investigate three ways to distribute the phrase pairs to the decoder layers: (1) *long-to-short* where the phrase length decreases from the bottom layer to the top layer; (2) *short-to-long* where the phrase length increases from the bottom layer to the top layer; and (3) *random* where the memory item of a certain phrase pair is stored by a randomly selected layer. Figure 7 shows the results of the three ways, where we find short-to-long achieves the best performance. We think the reason is that different

TABLE IV  
BLEU SCORES WHEN USING DIFFERENT TYPES OF MEMORY DROPOUT.  
THE RESULTS ARE REPORTED ON THE VALIDATION SET IN THE EN-ZH  
STYLE CUSTOMIZATION TASK.

Method	BLEU
No memory dropout	18.2
+ item-level memory dropout	18.2
+ layer-level memory dropout	18.6

TABLE V  
ANALYSIS OF MEMORY USAGE.

Method	BLEU
Ours	18.6
– gated fusion	18.3
– source-side memory	16.5
– target-side memory	16.1

TABLE VI  
MODEL PERFORMANCE WHEN INTEGRATING MEMORY INTO DIFFERENT  
LAYERS. "✓": EQUIPPED WITH THE MEMORY. "–": NOT EQUIPPED WITH  
THE MEMORY.

Selected Layers						BLEU
L1	L2	L3	L4	L5	L6	
✓	–	–	–	–	–	15.1
✓	✓	–	–	–	–	15.5
✓	✓	✓	–	–	–	15.6
✓	✓	✓	✓	–	–	16.2
✓	✓	✓	✓	✓	–	16.9
✓	✓	✓	✓	✓	✓	<b>18.6</b>
–	✓	✓	✓	✓	✓	17.5
–	–	✓	✓	✓	✓	17.2
–	–	–	✓	✓	✓	17.2
–	–	–	–	✓	✓	17.2
–	–	–	–	–	✓	16.4

layers may carry various types of linguistic properties in the Transformer model [57], which requires information of different granularities. When reading the memory, queries from lower layers may contain less contextualized information [25], thus short phrases are more suitable for them. At higher layers, long phrases that can provide more contextualized information performs better. We thus use short-to-long as the default setting.

*b) Effect of Memory Dropout:* We investigate the performance of two types of memory dropout: (1) *item-level memory dropout* that drop each memory item with a certain probability; and (2) *layer-level memory dropout* that drop all the memories at a decoder layer with a certain probability. Table IV shows the results, where all the models are trained using the overall loss function (i.e.,  $\mathcal{L}$  in Eq. (9)). We find the layer-level memory dropout performs better. In the following experiments, we use layer-level memory dropout by default.

*c) Effect of Memory Granularity:* To validate the necessity of building memory in a multi-granular form, we compare the performance of single- and multi-granular memories. Figure 8 shows the results, where we find using multi-granular memory can achieve lower validation loss, indicating the effectiveness of our method. We use multi-granular memory in the following experiments by default.

TABLE VII  
PERFORMANCE OF FINE-TUNING ALL MODEL PARAMETERS ON THE TEST  
SET OF EN-ZH STYLE CUSTOMIZATION.

Method	BLEU	PPL	Class.
Transformer	13.7	459.6	18.0
Fine-tuning	22.3	255.5	51.2
+ Mem.-Aug. Adapter	23.2	250.6	50.4

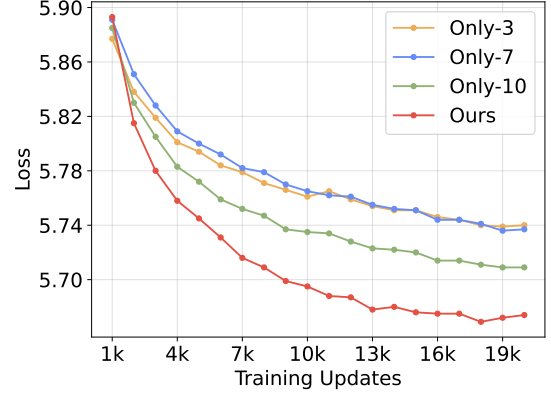


Fig. 8. Comparison between single-granular and multi-granular memory. Each curve is plotted on the validation set in the En-Zh style customization task. "Only-x": only using phrase pairs of length x to build the memory. "Ours": using the multi-granular memory.

*d) Analysis of Memory Usage:* Table V shows the effect of different components that are related to memory usage. Firstly, we notice that the gated fusion mechanism has a positive effect on translation quality, indicating the necessity of learning an input-dependent interpolation ratio between original model representations and retrieved results. In addition, we observe that there is a significant performance drop when using only either the source- or target-side memory. These results demonstrate that building parallel memory using phrase pairs is very useful.

*e) Effect of Integration Layers:* We also integrate the memory into different layers to better understand our method. Table VI shows the results, from which we find using the memories at all layers performs best and higher layers tend to be more important than lower layers. A potential reason is that memories at higher layers contain more contextualized information.

## B. Comparison with Fine-tuning

Although our goal in this work is to better build pluggable NMT models, our method is not only limited to this setting. For instance, the proposed memory-augmented adapter can also be used when the NMT model is not frozen (i.e., fine-tuning [26]). Table VII shows the results, where we observe that our method can also improve the performance of fine-tuning. This result implies that the external memory may provide essential information that is complementary to that stored in model parameters.

## C. Application to Larger Model

We also conduct experiments on a model of a larger scale, whose hidden size is 1024 and parameter size is



TABLE VIII  
CASE STUDY ON EN-ZH STYLE CUSTOMIZATION. FOR CLARITY, WE ONLY CHOOSE THE REPRESENTATIVE BASELINES (I.E., ADAPTER [13] AND  $k$ NN-MT [22]) FOR COMPARISON.

<b>Source</b>	So biological truth is by no means a talisman for polygamy.
<b>Adapter</b>	因此，生物学真理决不是一夫多妻制的护身符。
<b><math>k</math>NN-MT</b>	所以，生物学的真理，决不是一夫多妻制的护身的挡牌。
<b>Ours</b>	所以生物学上的道理，决不是一夫多妻的护身符。
<b>Reference</b>	所以生物学的真理，决非多妻主义的护符。
<b>Source</b>	Could it be that when he built the tower, he didn't think that the tower would fall down after all.
<b>Adapter</b>	难道他修建塔的时候，总不认为塔会倒塌吗。
<b><math>k</math>NN-MT</b>	倘若他建造雷峰塔的时候，他没有想到那塔终究会倒塌。
<b>Ours</b>	难道他造塔的时候，总不觉得那塔到底要倒下去么。
<b>Reference</b>	莫非他造塔的时候，竟没有想到塔是终究要倒的么。
<b>Source</b>	It was the morning of the fifth day that everyone dragged him up early in the morning and stood on the shore listening to the call.
<b>Adapter</b>	这是第五天早晨，大家一大早就把他拖起来，站在岸上听着呼唤。
<b><math>k</math>NN-MT</b>	第二天的早晨，大家一大早把他拖起来，站在岸上听听差打电话。
<b>Ours</b>	这是第五天的早晨，大家早把他拖起来，站在岸上听着叫。
<b>Reference</b>	就是这第五天的早晨，大家一早就把他拖起来，站在岸上听呼唤。
<b>Source</b>	Really, until now, I really haven't eaten the good beans like that night, and I haven't seen the good show like that night anymore.
<b>Adapter</b>	真的，到目前为止，我实在没有吃过那一晚这样的好豆子，我再也没有见过那晚这样的好秀了。
<b><math>k</math>NN-MT</b>	真的，直到现在，我实在没有吃过昨夜的豆子，我也从来没见过这样的好东西。
<b>Ours</b>	真的，直到现在，我实在没有吃过那夜那样好的豆子了，我也再没有见过那夜那样的好节目了。
<b>Reference</b>	真的，一直到现在，我实在再没有吃到那夜似的好豆，也不再看到那夜似的好戏了。
<b>Source</b>	This affair happened at the turn of winter and spring. The wind was not so cold anymore, and I wandered outside for a longer time; by the time I got home, it had been probably already dark.
<b>Adapter</b>	这事发生在冬春交替的时候，风不再那么冷了，我在外面漫游了更长的时间；到我到家的时候，大概是已经暗了。
<b><math>k</math>NN-MT</b>	这事发生在冬或今年春末，微风不再那么冷，我徘徊了大半天，到我回家的时候，天气大概已经很深了。
<b>Ours</b>	这事发生于冬春之交，风已不再那么冷，我在外面徘徊了更长的时间；到我到家的时候，大概已经天黑了。
<b>Reference</b>	这是冬春之交的事，风已没有这么冷，我也更久地在外面徘徊；待到回家，大概已经昏黑。
<b>Source</b>	For example, to build a railway, if we tell them how beneficial this thing is, they will never listen. If we, according to a myth, tell that previously a great immortal pushed a wheelbarrow over the rainbow, and now we imitate him to build a road, then everything can be done.
<b>Adapter</b>	举例来说，要修建铁路，如果我们告诉他们这事是何等的益处，他们就决不肯听从，如果我们，根据一个神话，告诉以前一个伟大的不朽的推车推着彩虹之上，现在我们模仿他修筑一条路，那么一切都可以完成了。
<b><math>k</math>NN-MT</b>	譬如说，如果我们告诉他们，铁路的建设是多么有益，他们决不听，如果我们根据神话说，先前一个大不朽的独轮车从彩虹上推过，现在我们模仿他造路，便可以做点事了。
<b>Ours</b>	譬如造铁路罢，倘若告诉他们这东西有多大益处，他们便决不肯听话，倘使我们据传说，说先前一个伟大的不朽的车手推着彩虹，现在就模仿他来造一条路，那么，一切便都可以做。
<b>Reference</b>	譬如要造一条铁路，倘若对他们说这事如何有益，他们决不肯听；我们如果根据神话，说从前某某大仙，曾推着独轮车在虹霓上走，现在要仿他造一条路，那便无所不可了。

596.0M. On the test set of En-Zh style customization, our memory-augmented adapter (without  $k$ NN decoding) outperforms Adapter [13] by 2.9 BLEU and  $k$ NN-MT [22] by 2.2 BLEU. This demonstrates that our method is also effective when the model size is larger. How to apply our method to larger models deserves further exploration.

#### D. Case study

We place some translation examples in Table VIII to provide a better understanding of the difference between the involved methods. There are 6 sentences from short to long. We can find that our method always outputs better translations. Also,

$k$ NN-MT is not always better than Adapter, see the 3-rd and 4-th cases.

Syed et al. believe that the author-style can be understood at three levels, from punctuation, word usage to syntax [41]. In our cases, we can find that our method can learn to generate author-style better in different granularities. From case 2, our method correctly translates the phrase “build the tower” to “造塔” while the other methods translate it to “修建塔” or “建造雷峰塔”. Although the meaning is the same, our translation is closer to the expression style of the original author/user. Similarly, our method properly translates the word “call” to “呼唤” while the other methods translate it to “打电话” or “叫” in case 3. Also, our method translates the phrase “that night” to “那夜” while the other methods translate it to “那一晚” or “昨夜” in case 4. Furthermore, it can also be easily found that our method can generate similar sentences that have similar syntactic styles. From case 1 and case 6, we can see that the sentence generated by our method is more similar to the reference in terms of sentence segmentation.

These cases from lexical level to syntactic structure also demonstrate the rationality and effectiveness of our multi-granularity memory design.

## VIII. CONCLUSION

In this work, we propose a memory-based adapter to build pluggable NMT models, which can let the users customize the generation behavior of NMT models by simply providing some text samples. We improve both the memory design and utilization to help existing models better adapt to the user-demanded styles or domains. Experiments demonstrate the superiority of our proposed method over several representative baselines.

In the future, we will validate the performance of our method on some stronger models in NMT, e.g., M2M100 12B [3] and NLLB 54.5B [58]. By changing the memory format, also we believe our method can be applied to some other sequence generation tasks.

## APPENDIX EXPERIMENTAL SETUP

### A. Training Details

*a) NMT Model Training:* We use WMT14 [59] De-En and WMT20 [60] Zh-En training data to train NMT models. For all the involved language pairs (i.e., En-Zh, Zh-En, and De-En), we train the Transformer model using the following hyper-parameters. All the models are optimized by Adam [61], with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.98$  and  $\epsilon = 10^{-9}$ . We train each model for 200K iterations on 4 NVIDIA A100 GPUs, where the training speed is 8.5 iterations per second. We use the learning schedule presented in Vaswani et al. [1], with a maximum learning rate of  $7e-4$  and the warm-up step is 4K. Each mini-batch contains 32K tokens in total. Both the dropout rate and the label smoothing penalty are set to 0.1. During inference, the beam size is 4. For En-Zh and Zh-En, the NMT models have 253.9M parameters. For De-En, the model has 198.3M parameters.

*b) Adapter Training:* We train the proposed memory-augmented adapter for 20K iterations. The maximum learning rate is set to  $2e-4$  and we restart the learning rate schedule when training adapters. Each mini-batch contains 8K tokens. Each experiment is conducted through a single run. We tune the values of the hyperparameters on the validation set through grid search.

*c)  $k$ NN Decoding:* To apply  $k$ NN decoding to our method, we should firstly build a datastore in the same way as that illustrated in Khandelwal et al. [22] using our model augmented with the proposed adapters. We use the open-sourced implementation of  $k$ NN decoding.<sup>7</sup>

### B. Details on Baselines

In this subsection, we provide the essential details of the baselines in this work:

- *Adapter* [13]: we use the same adapter architecture as presented in Houlisby et al. [13] The training hyperparameters are the same as our method, excluding some newly introduced hyperparameters (e.g.,  $\alpha$  and  $\beta$ ). The default dimension of the hidden layer is set to 64, following Houlisby et al. [13] Since our method use more parameters than Adapter, we also train a larger adapter to assimilate the parameter count, whose hidden dimension is set to 512. The bigger adapter still performs worse than our method (16.9 vs. 20.8 in terms of BLEU), indicating that our performance improvement is not totally caused by the larger adapter size.
- *MT+Rewrite* [41]: we train a rewriting model to refine the output of the NMT model. Specifically, we fine-tune a pretrained encoder-decoder model to transfer the model outputs into stylized texts.
- *$k$ NN-MT* [22]: there are mainly three hyperparameters that have a significant impact on performance:  $k$ ,  $T$ , and  $\lambda$ . We tune the hyperparameters on the validation set. For style customization,  $k = 128$ ,  $T = 30$  and  $\lambda = 0.7$  in En-Zh. In Zh-En,  $k = 16$ ,  $T = 40$  and  $\lambda = 0.6$ . For domain customization,  $k = 16$  across all the four domains.  $T = 4$  in IT, Medical, and Law.  $T = 40$  in Koran.  $\lambda$  is tuned to be 0.2, 0.3, 0.3, 0.6 in IT, Medical, Law, and Koran, respectively.
- *DExperts* [35]: we should learn two independent language models, of which one language model serves as an expert and another one is an anti-expert. The expert model is fine-tuned on the user-provided data while the anti-expert is trained on the general domain data.  $\alpha$  is tuned on the validation set and the final value is 0.2.

## REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of NeurIPS 2017*, 2017. [Online]. Available: <https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- [2] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, “Multilingual denoising pre-training for neural machine translation,” *Transactions of the ACL*, vol. 8, pp. 726–742, 2020. [Online]. Available: <https://aclanthology.org/2020.tacl-1.47>

<sup>7</sup><https://github.com/urvashik/knnmt>

- [3] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary, N. Goyal, T. Birch, V. Liptchinsky, S. Edunov, M. Auli, and A. Joulin, "Beyond english-centric multilingual machine translation," *Journal of Machine Learning Research*, vol. 22, no. 107, pp. 1–48, 2021. [Online]. Available: <http://jmlr.org/papers/v22/20-1307.html>
- [4] T. Vu and A. Moschitti, "Machine translation customization via automatic training data selection from the web," *CoRR*, vol. abs/2102.10243, 2021. [Online]. Available: <https://arxiv.org/abs/2102.10243>
- [5] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, and M. Zhou, "Achieving human parity on automatic chinese to english news translation," *CoRR*, vol. abs/1803.05567, 2018. [Online]. Available: <http://arxiv.org/abs/1803.05567>
- [6] T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, T. Gowda, Y. Graham, R. Grundkiewicz, B. Haddow, R. Knowles, P. Koehn, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, M. Novák, M. Popel, M. Popović, and M. Shmatova, "Findings of the 2022 conference on machine translation (WMT22)," in *Proceedings of the Seventh Conference on Machine Translation*, 2022. [Online]. Available: <https://aclanthology.org/2022.wmt-1.1>
- [7] P. Michel and G. Neubig, "Extreme adaptation for personalized neural machine translation," in *Proceedings of ACL 2018*, 2018. [Online]. Available: <https://aclanthology.org/P18-2050>
- [8] P. Zhang, Z. Guan, B. Liu, X. Ding, T. Lu, H. Gu, and N. Gu, "Building user-oriented personalized machine translator based on user-generated textual content," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 1–26, 2022.
- [9] N. S. Keskar, B. McCann, L. Varshney, C. Xiong, and R. Socher, "CTRL - A Conditional Transformer Language Model for Controllable Generation," *arXiv preprint arXiv:1909.05858*, 2019.
- [10] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, "Plug and play language models: A simple approach to controlled text generation," in *Proceedings of ICLR 2020*, 2020. [Online]. Available: <https://openreview.net/forum?id=H1edEyBKDS>
- [11] J. He, G. Neubig, and T. Berg-Kirkpatrick, "Efficient nearest neighbor language models," in *Proceedings of EMNLP 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.emnlp-main.461>
- [12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [13] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of ICML 2019*, 2019. [Online]. Available: <https://proceedings.mlr.press/v97/houlsby19a.html>
- [14] A. Bapna and O. Firat, "Simple, scalable adaptation for neural machine translation," in *Proceedings EMNLP-IJCNLP 2019*, 2019. [Online]. Available: <https://aclanthology.org/D19-1165>
- [15] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "AdapterFusion: Non-destructive task composition for transfer learning," in *Proceedings of EACL 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.eacl-main.39>
- [16] A. Rücklé, G. Geigle, M. Glockner, T. Beck, J. Pfeiffer, N. Reimers, and I. Gurevych, "AdapterDrop: On the efficiency of adapters in transformers," in *Proceedings EMNLP 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.emnlp-main.626>
- [17] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of ACL 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.acl-long.353>
- [18] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, S. Yih, and M. Khabsa, "UniPELT: A unified framework for parameter-efficient language model tuning," in *Proceedings of ACL 2022*, 2022. [Online]. Available: <https://aclanthology.org/2022.acl-long.433>
- [19] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. Li, and M. Sun, "Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models," *arXiv preprint arXiv:2203.06904*, 2022. [Online]. Available: <https://arxiv.org/abs/2203.06904>
- [20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proceedings of NeurIPS 2020*, 2020.
- [21] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Generalization through memorization: Nearest neighbor language models," in *Proceedings of ICLR 2020*, 2020. [Online]. Available: <https://openreview.net/forum?id=HkIBjCEKvH>
- [22] U. Khandelwal, A. Fan, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Nearest neighbor machine translation," in *Proceedings of ICLR 2021*, 2021. [Online]. Available: <https://openreview.net/forum?id=7wCBOFJ8hJM>
- [23] Q. He, G. Huang, Q. Cui, L. Li, and L. Liu, "Fast and accurate neural machine translation with translation memory," in *Proceedings of ACL-IJCNLP 2021*, 2021.
- [24] B. Li, T. Zheng, Y. Jing, C. Jiao, T. Xiao, and J. Zhu, "Learning multiscale transformer models for sequence generation," in *Proceedings of ICML 2022*, 2022. [Online]. Available: <https://proceedings.mlr.press/v162/li22ac.html>
- [25] J. Hewitt and P. Liang, "Designing and interpreting probes with control tasks," in *Proceedings of EMNLP-IJCNLP 2019*, 2019. [Online]. Available: <https://aclanthology.org/D19-1275>
- [26] M.-T. Luong and C. Manning, "Stanford neural machine translation systems for spoken language domains," in *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*, 2015. [Online]. Available: <https://aclanthology.org/2015.iwslt-evaluation.11>
- [27] X. Niu, M. Martindale, and M. Carpuat, "A study of style in machine translation: Controlling the formality of machine translation output," in *Proceedings of EMNLP 2017*, 2017. [Online]. Available: <https://aclanthology.org/D17-1299>
- [28] C. Chu and R. Wang, "A survey of domain adaptation for neural machine translation," in *Proceedings of COLING 2018*, 2018. [Online]. Available: <https://www.aclweb.org/anthology/C18-1111>
- [29] X. Niu, S. Rao, and M. Carpuat, "Multi-task neural models for translating between styles within and across languages," in *Proceedings of COLING 2018*, 2018. [Online]. Available: <https://aclanthology.org/C18-1086>
- [30] X. Wu, J. Liu, X. Li, J. Xu, Y. Chen, Y. Zhang, and H. Huang, "Improving stylized neural machine translation with iterative dual knowledge transfer," in *Proceedings of IJCAI 2021*, 2021. [Online]. Available: <https://www.ijcai.org/proceedings/2021/0547.pdf>
- [31] X. Niu and M. Carpuat, "Controlling neural machine translation formality with synthetic supervision," in *Proceedings of AAAI 2020*, 2020. [Online]. Available: <https://aaai.org/ojs/index.php/AAAI/article/view/6379>
- [32] J. Hu, M. Xia, G. Neubig, and J. Carbonell, "Domain adaptation of neural machine translation by lexicon induction," in *Proceedings of ACL 2019*, 2019. [Online]. Available: <https://aclanthology.org/P19-1286>
- [33] X. Zheng, Z. Zhang, S. Huang, B. Chen, J. Xie, W. Luo, and J. Chen, "Non-parametric unsupervised domain adaptation for neural machine translation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.findings-emnlp.358>
- [34] K. Yang and D. Klein, "FUDGE: Controlled text generation with future discriminators," in *Proceedings of NAACL 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.naacl-main.276>
- [35] A. Liu, M. Sap, X. Lu, S. Swayamdipta, C. Bhagavatula, N. A. Smith, and Y. Choi, "DExperts: Decoding-time controlled text generation with experts and anti-experts," in *Proceedings of ACL-IJCNLP 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.acl-long.522>
- [36] E. J. Hu, Y. Long, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proceedings of ICLR 2022*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZvKeeFYf9>
- [37] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. De Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," in *Proceedings of ICML 2022*, 2022. [Online]. Available: <https://proceedings.mlr.press/v162/borgeaud22a.html>
- [38] W. Chen, P. Verga, M. de Jong, J. Wieting, and W. Cohen, "Augmenting pre-trained language models with qa-memory for open-domain question answering," *arXiv preprint arXiv:2204.04581*, 2022.
- [39] Y. Jing, Y. Mao, Y. Yang, Y. Zhan, M. Song, X. Wang, and D. Tao, "Learning graph neural networks for image style transfer," in *ECCV 2022*. Springer, 2022, pp. 111–128.

- [40] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of CVPR 2023*, 2023, pp. 22 500–22 510.
- [41] B. Syed, G. Verma, B. V. Srinivasan, A. Natarajan, and V. Varma, "Adapting language models for non-parallel author-stylized rewriting," in *Proceedings of AAAI 2020*, 2020.
- [42] H. Singh, G. Verma, A. Garimella, and B. V. Srinivasan, "DRAG: Director-generator language modelling framework for non-parallel author stylized rewriting," in *Proceedings of EACL 2021*, 2021, pp. 863–873.
- [43] C. Dyer, V. Chahuneau, and N. A. Smith, "A simple, fast, and effective reparameterization of IBM model 2," in *Proceedings of NAACL 2013*, 2013. [Online]. Available: <https://aclanthology.org/N13-1073>
- [44] C. Chen, M. Sun, and Y. Liu, "Mask-align: Self-supervised neural word alignment," in *Proceedings of ACL-IJCNLP 2021*, 2021. [Online]. Available: <https://aclanthology.org/2021.acl-long.369>
- [45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>
- [46] N. Kambhatla, L. Born, and A. Sarkar, "CipherDAug: Ciphertext based data augmentation for neural machine translation," in *Proceedings of ACL 2022*, 2022. [Online]. Available: <https://aclanthology.org/2022.acl-long.17>
- [47] M. Post, "A call for clarity in reporting bleu scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018.
- [48] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of ACL 2019*, 2019.
- [49] J. Li, R. Jia, H. He, and P. Liang, "Delete, retrieve, generate: a simple approach to sentiment and style transfer," in *Proceedings of NAACL 2018*, 2018.
- [50] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of EMNLP 2014*, 2014. [Online]. Available: <https://aclanthology.org/D14-1181>
- [51] J. Zhang, H. Luan, M. Sun, F. Zhai, J. Xu, M. Zhang, and Y. Liu, "Improving the transformer translation model with document-level context," in *Proceedings of EMNLP 2018*, 2018. [Online]. Available: <https://aclanthology.org/D18-1049>
- [52] P. Ke, F. Huang, M. Huang, and X. Zhu, "ARAML: A stable adversarial training framework for text generation," in *Proceedings of EMNLP-IJCNLP 2019*, 2019. [Online]. Available: <https://aclanthology.org/D19-1436>
- [53] Z. Hu, R. K.-W. Lee, C. C. Aggarwal, and A. Zhang, "Text style transfer: A review and experimental evaluation," *ACM SIGKDD Explorations Newsletter*, vol. 24, no. 1, pp. 14–45, 2022.
- [54] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.
- [55] R. Aharoni and Y. Goldberg, "Unsupervised domain clusters in pretrained language models," in *Proceedings of ACL 2020*, 2020. [Online]. Available: <https://aclanthology.org/2020.acl-main.692>
- [56] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, pp. 535–547, 2017.
- [57] E. Voita, R. Sennrich, and I. Titov, "The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives," in *Proceedings of EMNLP-IJCNLP 2019*, 2019. [Online]. Available: <https://aclanthology.org/D19-1448>
- [58] NLLB Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang, "No language left behind: Scaling human-centered machine translation," *arXiv preprint arXiv:2207.04672*, 2022. [Online]. Available: <https://arxiv.org/abs/2207.04672>
- [59] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014. [Online]. Available: <https://aclanthology.org/W14-3302>
- [60] L. Barrault, M. Biesialska, O. Bojar, M. R. Costa-jussà, C. Federmann, Y. Graham, R. Grundkiewicz, B. Haddow, M. Huck, E. Joanis, T. Kocmi, P. Koehn, C.-k. Lo, N. Ljubešić, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, S. Pal, M. Post, and M. Zampieri, "Findings of the 2020 conference on machine translation (WMT20)," in *Proceedings of the Fifth Conference on Machine Translation*, 2020. [Online]. Available: <https://aclanthology.org/2020.wmt-1.1>
- [61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.