PEMA: An Offsite-Tunable Plug-in External Memory Adaptation for Language Models

Anonymous ACL submission

Abstract

Pre-trained language models (PLMs) show impressive performance in various downstream NLP tasks. However, pre-training large language models demands substantial memory and training compute. Furthermore, due to the substantial resources required, many PLM weights are confidential. Consequently, users are compelled to share their data with model owners for fine-tuning specific tasks. To overcome the limitations, we introduce Plug-in External Memory Adaptation (PEMA), a Parameter-Efficient Fine-Tuning (PEFT) method, enabling PLM fine-tuning without requiring access to all the weights. PEMA integrates with context representations from test data during inference to perform downstream tasks. It uses external memory to store PLM-generated context representations mapped with target tokens. Our method utilizes LoRA-based weight matrices in the PLM's final layer to enhance efficiency. Our approach also includes Gradual Unrolling, a novel interpolation strategy to improve generation quality. We validate PEMA's effectiveness through experiments on syntactic and real datasets for machine translation and style transfer. Our findings show that PEMA outperforms other PEFT approaches in memory and latency efficiency for training, and also excels in maintaining sentence meaning and generating appropriate language and styles.

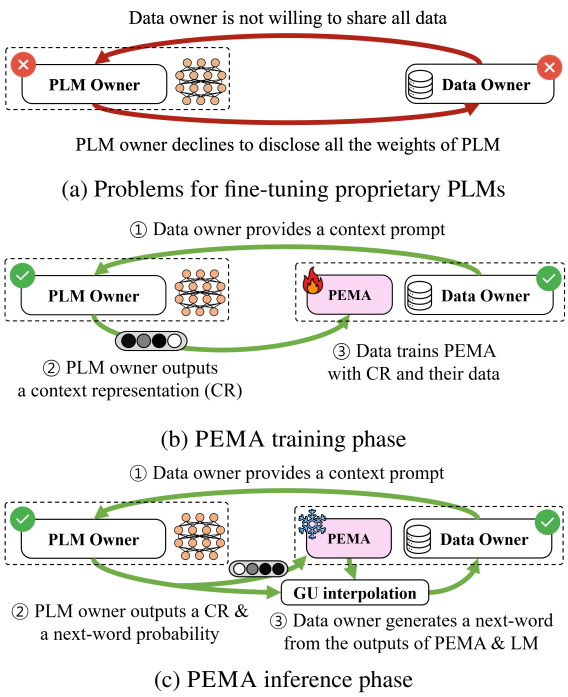

Figure 1: A motivation for PEMA. (a) The data owners who want to fine-tune PLMs encounter a problem when the PLM owner refuses to share all the weights of the PLM. (b) In the PEMA training phase, the data owner takes a CR from the PLM owner by providing a context prompt. They subsequently train their PEMA model with their dataset. (c) At inference, the data owner takes a CR for test data from the PLM owner. Using Gradual Unrolling (GU), they generate the next-token by interpolating between PEMA and PLM next-token probabilities.

1 Introduction

Pre-trained language models (PLMs) are widely used in downstream NLP tasks (Devlin et al., 2019a). Recent developments in large language models have shown remarkable performance in zero-shot and few-shot learning scenarios (Brown et al., 2020; Hendy et al., 2023; OpenAI, 2023b; Anil et al., 2023; Chowdhery et al., 2022). However, fine-tuning is still required to optimize the performance of the NLP tasks such as machine translation (Üstün and Cooper Stickland, 2022; Huang et al., 2020; Ding et al., 2022). The most straightforward approach to fine-tuning is full fine-tuning (Raffel et al., 2020; Qiu et al., 2020), which involves fine-tuning all parameters in a PLM. Yet, this approach requires substantial resources regarding memory and training compute (Iyer et al., 2022; Zhang et al., 2022; Touvron et al., 2023). To overcome this limitation, researchers have proposed Parameter-Efficient Fine-Tuning (PEFT) methods to fine-tune a full model efficiently. Adapter tuning (Pfeiffer et al., 2021; He et al., 2021; Houlsby et al., 2019) utilizes small, additional parameters known as adapters inserted between layers within a PLM. On the other hand, LoRA (Hu et al., 2022) uses trainable low-rank matrices that incrementally update the pre-trained weights. These fine-tuning methods require access to all the weights of PLMs.

However, proprietary PLMs such as ChatGPT (OpenAI, 2022), Bard (Pichai, 2023), and Claude (AnthropicAI, 2023) are confidential. Hence, the owners of these PLMs do not reveal all the model weights. Consequently, data owners possessing their datasets and wishing to fine-tune proprietary PLMs for specific downstream tasks must provide their datasets to the PLM owners for fine-tuning (OpenAI, 2023a). However, this process can be challenging due to the confidential nature of the datasets, which may involve privacy concerns (Guinney and Saez-Rodriguez, 2018). Figure 1a shows problems for fine-tuning proprietary PLMs. To overcome this situation, (Xiao et al., 2023) proposes the offsite-tuning approach that uses one-third of the middle layers of a PLM, referred to as the emulator. Nevertheless, this approach still needs a large parameter size, and compressing the full model into an emulator requires a computationally intensive distillation process.

To address the challenges mentioned above, we introduce a novel PEFT method named Plug-in External Memory Adaptation (PEMA) designed for efficient fine-tuning of proprietary PLMs in machine translation tasks. PEMA utilizes LoRA-based weight matrices designed for learning downstream tasks with accessible features provided by OpenAI API (OpenAI, 2022) and minimal part of PLM's weight (language model head).

In the training phase, the data owner begins the process by providing a prompt with initial input to the PLM owner, which includes an instruction and a source sentence from a parallel corpus. The PLM owner receives this initial input to generate a context representation and predict the next-token. Then, it iteratively processes subsequent inputs containing the predicted next-tokens. This approach avoids the need for the full dataset from the data owner. Throughout this process, the data owner builds an external memory comprised of context representations and corresponding desired target tokens. They train PEMA by reconstructing the stored context representations and predicting target tokens based on these representations. Figure 1b shows the training phase process of PEMA.

During the inference phase, the data owner uses a prompt to request a context representation for test data from the PLM owner. The PLM owner then outputs a context representation and a next-token probability given the prompt. PEMA also outputs a next-token probability based on a context representation. These probabilities are interpolated to compute a final next-token probability. We propose Gradual Unrolling (GU), an interpolation strategy that initially emphasizes PEMA's distribution, gradually shifts to the PLM's context-based predictions as the sentence progresses. Figure 1c illustrates the inference phase process of PEMA.

We evaluate PEMA by comparing it with other PEFT methods. PEMA shows better resource efficiency, consuming less GPU memory and running faster. Additionally, PEMA outperforms other baselines in translating English sentences into German and paraphrasing informal sentences into formal ones while preserving the original meaning. Lastly, we conduct ablation studies to assess the effectiveness of each component of PEMA. PEMA is publicly available for further exploration into offsite-tunable efficient fine-tuning.[1]

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning aims to fine-tune PLMs to address resource constraints in memory and training compute. (Iyer et al., 2022; Zhang et al., 2022; Touvron et al., 2023). Several approaches have been proposed to overcome this limitation. Adapter tuning (Pfeiffer et al., 2021; He et al., 2021; Houlsby et al., 2019) inserts small parameters, known as adapters, between layers within a PLM. Prefix and Prompt tuning (Li and Liang, 2021; Liu et al., 2021; Lester et al., 2021) incorporate additional trainable prefix tokens to a PLM's input or hidden layers. Low-Rank Adaptation (LoRA) (Hu et al., 2022) uses trainable low-rank matrices, denoted as B and A, that incrementally update PLM weights. B and A are reduced to a low-rank r. This adaptation can be mathematically represented as transitioning from $h = W_0x$ to $h = W_0x + \Delta Wx = W_0x + BAx$, where $W_0 \in \mathbb{R}^{k\times d}$, $B \in \mathbb{R}^{k\times r}$, and $A \in \mathbb{R}^{r\times d}$. UniPELT (Mao et al., 2022) combines multiple PEFT methods, using a gating mechanism to activate the most suitable components for given data or

[1]The Github repository link will be provided after review.

tasks. We propose a novel adaptation method that leverages LoRA parameters and is offsite-tunable.

2.2 k-Nearest Neighbors Language Model

The k-Nearest Neighbors Language Model (kNN-LM) estimates the next-token distribution by interpolating the output distributions from a pre-trained language model (P_{LM}), and an external memory (P_{kNN}) (Khandelwal et al., 2020). The memory is used to perform a kNN search and to integrate out-of-domain data, thereby enabling a single language model to be adaptive across various domains. Given a context represented as a sequence of tokens $c_i = (w_1, ..., w_{i-1})$, the kNN-LM utilizes a pre-trained language model $f(\cdot)$ to generate a context representation $f(c_i)$. This representation is then paired with the desired target token y_i to create the external memory (referred to as a datastore in (Khandelwal et al., 2020)) $\{(f(c_i), y_i)|(c_i, y_i) \in \mathcal{E}\}$ from the training dataset $\mathcal{E}$. The next-token distribution from the external memory, P_{kNN}, is computed using a k-nearest neighborhood approach with the squared L^2 distance. The final next-token distribution is then obtained by interpolating between P_{kNN} and P_{LM} as: $P(y_i|c_i) = \lambda P_{kNN}(y_i|c_i) + (1-\lambda) P_{LM}(y_i|c_i)$.

We adapt the concept of external memory and interpolation of different next-token distributions to PEMA. Instead of employing a kNN-based approach, we employ a neural network-based model that directly learns to estimate the next-token, which is more effective in mitigating overfitting to the training data. Additionally, we use the Gradual Unrolling interpolation strategy to enhance the quality of interpolation. The kNN-LM method relies on kNN for external memory search to adapt the language model to diverse domains. However, it is well known that the non-parametric model kNN can potentially overfit. Therefore, it often requires a large amount of training data. To address this, we introduce a parametric approach within PEMA to improve its performance on downstream tasks. This approach is better suited for limited training data scenarios. It involves replacing the existing kNN with a parametric model in PEMA, thus enabling effective adaptation to various domains in terms of performance.

3 Plug-in External Memory Adaptation

This section describes Plug-in External Memory Adaptation (PEMA), which aims to fine-tune a pre-trained language model without requiring a full model during training. PEMA is integrated into the language model during inference to facilitate downstream NLP tasks. It uses external memory to build a context representation $f(c_i)$, mapped with the desired target token y_i. Using the external memory, we train PEMA in two phases. The first phase involves reconstruction training to reconstruct $f(c_i)$ with $B_{rct}A$, resulting in the output of a reconstruction loss. Subsequently, the joint retraining phase focuses on generating the next-token probability P_{PEMA} that predicts target token y_i given $Af(c_i)$ with B_{pd}. Simultaneously, it uses pre-trained B_{rct} to retain the original feature $f(c_i)$. During the inference stage, the next-token probabilities from both the pre-trained generative language model P_{LM} and PEMA P_{PEMA} are interpolated to generate the next-token. Figure 2 shows the structure of PEMA.

3.1 Building an External Memory

The first step of PEMA is to build an external memory. The output $f(c_i)$ represents a context representation obtained from the final layer's feed-forward network output of a pre-trained language model.

For the i-th token training example in external memory $(c_i, y_i) \in \mathcal{E}$, a paired representation is created by defining an input prompt c_1 and a corresponding target token sequence. Predicted token sequences are generated by sequentially extending the input prompt. ❶ Initially, the input prompt c_1 is fed into the pre-trained language model, resulting in the predicted next-token $\hat{w}_1$ and ❷ the corresponding context representation $f(c_1)$. ❸ Including $\hat{w}_1$ in the input prompt extends it to the next context $c_2 = \{c_1, \hat{w}_1\}$, subsequently producing the next predicted token $\hat{w}_2$ and its context representation $f(c_2)$. This iterative process yields a sequence of context representations $(f(c_1), f(c_2), ..., f(c_t = \{c_1, \hat{w}_1, ..., \hat{w}_{t-1}\})$ for training, with each context c_i corresponding to the i-th position in the token sequence and t denoting the total number of tokens in a token sequence of one sentence training example.

We map the context representation $f(c_i) \in \mathbb{R}^{1\times d}$, where d is the size of the context representation with the target token y_i, resulting in the pair $(f(c_i), y_i)$. The external memory $(f(C), Y)$ is formed by collecting all such context and token pairs constructed from the training set $\mathcal{E}$ as below:

$$(f(C), Y) = \{(f(c_i), y_i)|(c_i, y_i) \in \mathcal{E}\} \quad (1)$$

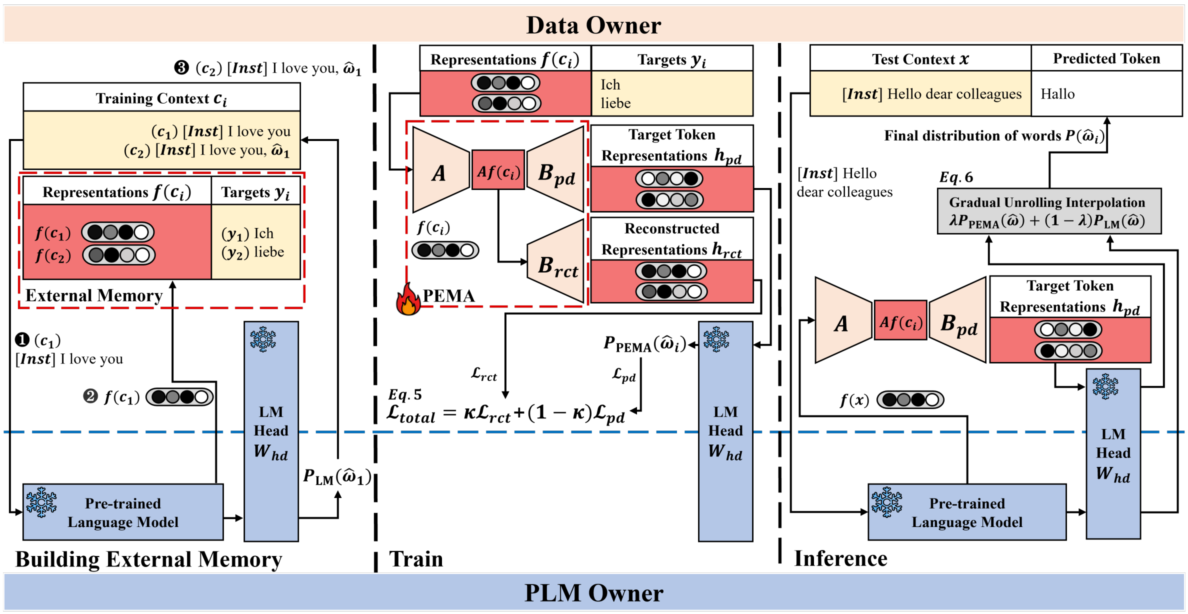

Figure 2: An illustration of PEMA. The areas of the PLM owner and the data owner are separated by the blue horizontal line. The data owner can train and infer using only the PLM's LM head. PEMA builds an external memory from the training context with an instruction $[Inst]$ given to a PLM. The PLM outputs the representation $f(c_i)$ and predicts the next-token distribution $P_{LM}(\hat{w}_i)$. The representation $f(c_i)$ is then aligned with its target y_i. In the training phase, PEMA uses external memory for two tasks: preserving the original representation via reconstruction training with B_{rct} and generating a target token probability distribution using B_{pd}. For inference, the model inputs a test data representation to generate two probability distributions: $P_{LM}(\hat{w}_i)$ and $P_{PEMA}(\hat{w}_i)$. These are then interpolated using Gradual Unrolling to obtain the final token distribution.

3.2 PEMA Adaptation Model

We incorporate LoRA (Hu et al., 2022), a low-rank parameterization adaptation known for its effectiveness in various adaptation tasks, into PEMA for adapting to multiple text generation tasks.

The PEMA consists of three weight matrices: $A \in \mathbb{R}^{r\times d}$, $B_{rct} \in \mathbb{R}^{d\times r}$, and $B_{pd} \in \mathbb{R}^{d\times r}$ where d is the size of the context representation and r is a rank-size that $r < d$. Given $Af(c_i)$ where $f(c_i) \in \mathbb{R}^{1\times d}$, B_{rct} is used to reconstruct the context representation input $f(c_i)$, with the goal of approximating $h_{rct i} \approx f(c_i)$, Additionally, B_{pd} is used to produce a representation h_{pd_i} that maximizes target token prediction when fed into the frozen weight of a language model head (LM head) $W_{hd} \in \mathbb{R}^{v\times d}$ where v is the vocabulary size that outputs the predicted next-token $\hat{w}_i$.

$$\begin{aligned} h_{rct i} &= \Delta W_{rct} f(c_i) = B_{rct} f(c_i) \\ h_{pd_i} &= \Delta W_{pd} f(c_i) = B_{pd} f(c_i) \\ P_{PEMA}(\hat{w}_i|c_i) &= \text{softmax}(W_{hd} h_{pd_i}) \end{aligned} \quad (2)$$

3.3 Model Training

The training process consists of two distinct phases: initial reconstruction training to preserve the general knowledge within the context representation of PLM and subsequent joint retraining, encompassing both the reconstruction of context representations and the prediction of next-tokens.

Initial Reconstruction Training. First, we train the decoder B_{rct} by reconstructing the i-th original context representation of the n-th sentence training example $f(c_i)^n$. We use a mean-square error loss between original input $f(c_i)^n$ and the output $h^n_{rct i}$ as below:

$$\mathcal{L}_{rct} = \frac{1}{|\mathcal{E}|}\sum_{n=1}^{|\mathcal{E}|}\sum_{i=1}^{t_n}(f(c_i)^n - h^n_{rct i})^2 \quad (3)$$

where t_n is the number of tokens in a token sequence of n-th sentence training example and $|\mathcal{E}|$ is the size of the training dataset.

Joint Retraining After completing the initial reconstruction training, we proceed to the joint retraining phase, using the pre-trained B_{rct} and randomly initialized A. Our first objective is to acquire a representation $h^n_{pd_i}$ that is optimized for predicting the target token y^n_i. We utilize a cross-entropy loss based on the softmax function of the output of

$W_{hd}h^n_{pd_i}$ given the target token y^n_i as below:

$$\mathcal{L}_{pd} = -\frac{1}{|\mathcal{E}|}\sum_{n=1}^{|\mathcal{E}|}\sum_{i=1}^{t_n} y^n_i \log P_{PEMA}(y^n_i|W_{hd}h^n_{pd_i}) \quad (4)$$

The second objective is to reconstruct the input context representation x_i using the randomly initialized A and pre-trained B_{rct} with the reconstruction loss function as depicted in Equation 3. The reconstruction loss intends to retain the general knowledge obtained from the pre-trained language model while maximizing the target token prediction. We introduce a parameter κ that can be fine-tuned to adjust the emphasis on the objectives as below:

$$\mathcal{L}_{total} = \kappa\mathcal{L}_{rct} + (1-\kappa)\mathcal{L}_{pd} \quad (5)$$

3.4 Model Inference

To generate the next-token $\hat{w}$, we exclude B_{rct} and use B_{pd} and A. The PLM receives the input context x from the test dataset, and generates $f(x)$, which serves as input for two pathways. One pathway uses PEMA's A and B_{pd} to create h_{pd} for x. Subsequently, it is passed through W_{hd} to produce a distribution of the next-token $P_{PEMA}(\hat{w}|x)$. The other pathway directly feeds r into W_{hd} to produce the next-token distribution $P_{LM}(\hat{w}|x)$. Finally, these two distributions are blended using a tuned parameter λ to produce the final distribution of tokens for the desired task as below:

$$P(\hat{w}|x) = \lambda P_{PEMA}(\hat{w}|x) + (1-\lambda)P_{LM}(\hat{w}|x) \quad (6)$$

4 Gradual Unrolling Interpolation

Given that an adaptation model trained with only a limited number of parameters may lack the context-awareness and language-generation capabilities of pre-trained language models, it is more effective to use the adaptation model to guide the generation of tokens of the desired task at the beginning of the sentence, and rely on a pre-trained language model to provide context for the rest of the sentence. To achieve this, we suggest the Gradual Unrolling strategy, which aims for strong $P_{PEMA}(\hat{w}|x)$ interpolation at the beginning of generation and gradually decreases the interpolation. As the sentence progresses, the pre-trained language model increasingly contributes to providing the necessary context, as shown in Figure 3.

In the context of sentence generation, we define SL as the input sentence length, excluding instruction and user-defined variables λ_{max}. λ represents the proportion of the adaptation model's interpolation ($0 \leq \lambda \leq 1$). We also have the dependent variables of the current step (CS) and the step size (SS). The step size is computed as $SS = \lambda_{max}/SL$, and CS is initialized to λ_{max} at the start of sentence generation. At each token generation step, CS decreases by SS until the end of the sentence (i.e., $CS_{cur} = CS_{past} - SS$ where CS_{past} is the latest token's CS variable). Then, we calculate the current interpolation proportion λ_{cur} (i.e., λ at Equation 6) as $\lambda_{cur} = CS^2_{cur}$.

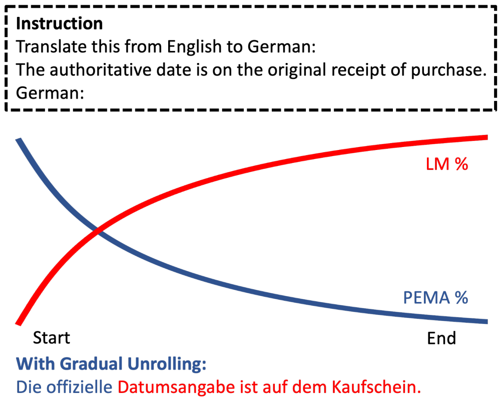

Figure 3: The intuition of Gradual Unrolling. Given the input sentence (Black), the interpolation percentage of the adaptation model (Blue) decreases gradually while that of the language model (Red) increases as the sentence is being generated. This strategy ensures that the adaptation model generates tokens trained for the desired task at the beginning of the sentence, and the language model provides the necessary context in the remaining part of the sentence.

5 Experiments

This section describes the experiments and results to show both the computational efficiency and performance in downstream tasks of PEMA. First, we perform an experiment on the computational efficiency of PEMA. Subsequently, we evaluate PEMA across two downstream tasks: the WMT22 EN→DE machine translation task (Kocmi et al., 2022) and the GYAFC formal style transfer task (Rao and Tetreault, 2018). Lastly, we conduct an ablation study to show the gradual improvement by incorporating each idea of PEMA.

5.1 Computational Efficiency

To evaluate the computational efficiency of PEMA, we conduct a comparison of different fine-tuning methods based on their resource utilization during both training and inference. We follow the approach of previous work (Pope et al., 2023) that

Method	Tr-MC	Tr-Lat	Inf-MC	Inf-Lat	FLOPs
FT	20,082	$250.4_{\pm 140.6}$	5,021	$17.1_{\pm 1.0}$	2.41e9
FT-top2	7,355	$70.3_{\pm 108.6}$	5,021	$17.3_{\pm 1.3}$	2.41e9
kNN-LM	None	$20.3_{\pm 567.2}$	5,021	$37.5_{\pm 1.4}$	FT+6.29e6
LoRA	5,056	$21.6_{\pm 0.4}$	5,031	$20.5_{\pm 1.5}$	FT+4.19e6
UniPELT	5,138	$30.3_{\pm 0.1}$	5,047	$21.3_{\pm 0.6}$	FT+1.49e7
OT Emulator	11,713	$88.4_{\pm 309.4}$	None	None	FT+8.03e8
OT Plug-in	5,267	$59.6_{\pm 107.8}$	5,269	$21.3_{\pm 0.1}$	FT+4.82e8
PEMA	478	$18.5_{\pm 1.0}$	5,043	$18.2_{\pm 0.5}$	FT+4.19e6

Table 1: Comparison of various training and inference resource utilization methods with OPT-IML-MAX-1.3B. We evaluate memory consumption (MC) and latency (Lat) for training (Tr) and inference (Inf), as well as FLOPs per token, using 10-token length sequences. Memory size is measured in megabytes, and latency is measured in milliseconds. PEMA stands out by using only one-tenth of the training memory utilized by LoRA. Furthermore, PEMA demonstrates the fastest training latency among the methods.

employs a fixed size of input tensors. We use input tensors with the size [1, 10], equivalent to sequences of 10 tokens with OPT-IML-MAX-1.3B. The resource utilization metrics encompass training memory consumption, training latency, inference memory consumption, inference latency, and floating point operations per token.

The evaluation involves several steps. First, we clear the CUDA cache to compute the memory and ensure no background GPU processes. GPU memory utilization is determined using the memory_summary function provided by Pytorch (Paszke et al., 2019). We calculate the time difference before inputting the data into the model and after obtaining the output. For training latency, we consider the time encompassing the entire backpropagation process. To ensure the accuracy of latency, we compute the mean and variance based on ten trials of inputs for each fine-tuning method. We conducted a comparative analysis with the offsite-tuning baseline approach, Offsite-Tuning (Xiao et al., 2023). Offsite-Tuning involves knowledge distillation (OT Emulator) and downstream task training using the OT Emulator (OT Plug-in). Subsequently, it utilizes the OT Plug-in to interact with the PLM during the inference phase.

As shown in Table 1, PEMA demonstrates the efficiency by utilizing one-tenth of the training memory consumption compared to LoRA. In addition, PEMA shows the fastest training latency among all the methods. This is because PEMA uses external memory to store context representations and does not require access to a pre-trained language model during the training phase, as illustrated in Figure 2. These results highlight the significance of PEMA's reduced training memory consumption and improved training latency, making it an appealing choice for efficient natural language generation tasks.

5.2 Performance of Downstream Tasks

We present a comprehensive analysis of the performance of PEMA and baseline models on two downstream tasks: the WMT22 (EN→DE) translation task and the GYAFC task involving Family & Relationships and Entertainment & Music.

For the machine translation task, we use the EN→DE news-commentary dataset to address the limitation noted in (Brown et al., 2020), where translations into English tend to be stronger than those from English due to training set biases. We evaluate our model using the latest test set provided by (Hendy et al., 2023).

For the formality style transfer task, we use the GYAFC dataset (Rao and Tetreault, 2018), which consists of a parallel training set of informal and formal sentences. The test set comprises four reference sentences paired with one informal sentence. In this task, our objective is to transfer the style of informal sentences into formal ones.

We use three pre-trained language models: OPT-IML-MAX-1.3B, LLaMA-7B, and OPT-IML-MAX-30B (Iyer et al., 2022; Touvron et al., 2023). We compare PEMA with the following methods:

Full fine-tuning (FT) updates all pre-trained model parameters, including weights and biases.

Fine-tuning top-2 (FT-Top2) updates the last two layers while the remaining layers are frozen.

k-Nearest Neighbors Language Model (kNN-LM) (Khandelwal et al., 2020) uses kNN search within an external memory to derive a next-token distribution P_{kNN}, which is then interpolated with P_{LM} to produce an adapted next-token distribution.

LoRA (Hu et al., 2022) uses two additional trainable matrices. We apply LoRA at the last layer output projection matrices in the self-attention module.

UniPELT (Mao et al., 2022) is a state-of-the-art PEFT method that combines Adapter tuning (Houlsby et al., 2019), Prefix tuning (Li and Liang, 2021), and LoRA (Hu et al., 2022) with a gating mechanism to select the optimal approaches. We apply UniPELT at the last layer.

Offsite-Tuning (Xiao et al., 2023) is an offsite-tunable method that uses a distilled PLM emulator with an adapter, which includes multiple copies at

Model	Tr-MC (MB)	WMT22 (EN→DE) sBLEU	PPL	COMET	GYAFC (F&R) sBLEU	PPL	FormImp	GYAFC (E&M) sBLEU	PPL	FormImp
OPT-1.3B	None	9.55	51.30	57.24	55.00	**18.98**	11.05	53.98	20.89	10.67
OPT-1.3B (FT)	20,082	10.15	40.83	61.44	29.17	24.82	52.28	31.50	27.99	46.82
OPT-1.3B (FT-Top2)	7,355	3.57	51.36	38.35	21.60	24.33	**59.00**	23.94	27.07	51.52
OPT-1.3B (kNN-LM)	None	8.07	91.37	41.75	56.69	20.87	16.26	54.74	23.15	14.46
OPT-1.3B (LoRA)	5,025	4.28	61.25	39.32	20.98	19.07	45.71	15.57	**19.71**	46.32
OPT-1.3B (UniPELT)	5,138	9.15	47.09	56.30	51.38	44.43	52.22	46.67	22.08	**53.31**
OPT-1.3B (Offsite-Tuning)	5,267	7.65	**36.91**	52.85	59.01	20.70	24.82	57.01	23.25	23.76
OPT-1.3B (PEMA)	478	**12.87**	42.62	**64.16**	**64.82**	23.15	41.90	**61.24**	24.28	36.28
LLaMA-7B	None	2.78	78.49	39.49	20.18	34.53	42.81	24.14	37.33	44.81
LLaMA-7B (kNN-LM)	None	0.07	85.09	38.53	1.72	41.50	55.13	1.94	46.31	68.61
LLaMA-7B (LoRA)	13,237	11.46	51.36	67.48	52.67	**22.42**	**72.23**	52.15	**24.74**	**71.28**
LLaMA-7B (UniPELT)	13,810	9.13	**46.62**	56.31	59.81	22.95	71.69	58.07	25.35	68.33
LLaMA-7B (PEMA)	996	**14.50**	54.26	**70.31**	**63.99**	23.19	61.40	**60.88**	26.00	60.94
OPT-30B	None	18.22	**45.81**	77.41	60.41	**20.04**	29.33	57.60	**21.97**	23.88
OPT-30B (kNN-LM)	None	16.65	74.06	62.98	61.02	20.86	29.80	58.58	22.75	23.39
OPT-30B (LoRA)	58,083	8.26	46.97	69.41	61.39	22.00	**73.10**	59.76	23.97	**68.29**
OPT-30B (UniPELT)	59,028	15.57	47.34	73.42	64.54	21.72	47.14	56.86	23.77	34.08
OPT-30B (PEMA)	1,909	**19.22**	46.62	**79.21**	**70.84**	22.04	52.35	**65.43**	25.53	44.63

Table 2: Comparison of various models across different tasks. The evaluated tasks include WMT22 (EN→DE) translation and GYAFC Family & Relationships (F&R) and GYAFC Entertainment & Music (E&M) style transfer. The models considered for evaluation are OPT-IML-MAX-1.3B, LLaMA-7B, and OPT-IML-MAX-30B, each with specific adaptations and configurations.

the PLM's beginning and end. We use four adapter layers for training and inference.

We use widely used evaluation metrics to assess the performance of PEMA as follows:

Sacre-Bleu (sBLEU) (Post, 2018) is a commonly used metric to calculate the n-gram accuracy between the source and target sentences. It evaluates how well the generated sentence preserves the meaning of the reference and captures target domain distribution. Higher scores are better.

Perplexity (PPL) (Jelinek et al., 1977) is to assess the fluency of generated sentences. We use pre-trained GPT-2 large (Radford et al., 2019) to calculate the exponential of the negative log-likelihood of a current token given the previous context. Lower scores are better.

COMET (Rei et al., 2020) is a neural network-based metric for assessing machine translation quality. It shows a positive correlation with human judgments. We utilize the default, pre-trained COMET model[2] for the WMT22. Higher scores are better.

Formality Improvement (FormImp) measure formality improvement based on XFORMAL (Briakou et al., 2021a). To measure the formality score of a sentence, we train a BERT-Large (Devlin et al., 2019b) on an external formality dataset consisting of 4K human-annotated examples (Pavlick and Tetreault, 2016). We compute the formality score for each formal reference sentence (FR), informal input sentence (II), and generated sentence (G). Then, we measure the relative distance using the formula: $\frac{G}{FR-II} \times 100$. We employ this metric for the GYAFC task. Higher scores are better.

[2]https://github.com/Unbabel/COMET

5.2.1 Results

For the WMT22 (EN→DE) translation task, we evaluated sBLEU, PPL, and COMET metrics. As Table 2 shows, PEMA outperforms baselines in sBLEU and COMET. Offsite-Tuninig, LoRA, and UniPELT perform slightly better than a naive pre-trained language model and PEMA in terms of PPL. However, they require more memory consumption for training than PEMA. Finally, PEMA generates more appropriate translated sentences than other baselines for sBLEU with relatively small memory consumption.

For the GYAFC style transfer task, we evaluated sBLEU, PPL, and Formality Improvement (FormImp) metrics. As Table 2 shows, PEMA consistently achieves favorable performance. PEMA shows the highest sBLEU scores, effectively maintaining meaning preservation across different domains and models. PEMA performs slightly better than a naive pre-trained language model and is comparable to other baselines in terms of FormImp. Furthermore, we observe a trade-off between sBLEU and formality. These findings support previous observations in the same formality style transfer task with multilingual formality (Briakou et al., 2021b).

WMT22 (EN→DE)	sBLEU	PPL	COMET
OPT-30B	18.22	**45.81**	77.41
OPT-30B+B_{pd}	18.74	48.05	77.76
OPT-30B+B_{pd}+GU	19.17	48.60	78.57
OPT-30B+B_{pd}+GU+B_{rct} (PEMA)	**19.22**	46.62	**79.21**
GYAFC (F&R)	sBLEU	PPL	FormImp
OPT-30B	60.41	20.04	29.33
OPT-30B+B_{pd}	70.00	20.38	47.38
OPT-30B+B_{pd}+GU	70.29	**16.95**	51.24
OPT-30B+B_{pd}+GU+B_{rct} (PEMA)	**70.84**	22.04	**52.35**
GYAFC (E&M)	sBLEU	PPL	FormImp
OPT-30B	57.60	**21.97**	23.88
OPT-30B+B_{pd}	64.37	26.76	38.80
OPT-30B+B_{pd}+GU	64.82	25.62	42.61
OPT-30B+B_{pd}+GU+B_{rct} (PEMA)	**65.43**	25.53	**44.63**

Table 3: Ablation results of PEMA over our proposed approaches. The techniques include a token prediction decoder (B_{pd}), Gradual Unrolling (GU), and a reconstruction decoder (B_{rct}). We use OPT-IML-MAX-30B as a baseline. Implementing all techniques together enhances overall performance.

λ/λ_{max}	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
With GU	47.45	46.61	46.62	46.18	46.12	46.03	45.85	45.89	45.84
Without GU	54.29	51.87	50.22	49.70	49.45	48.09	47.76	47.67	47.52

Table 4: Impact of Gradual Unrolling (GU) on perplexity across different λ/λ_{max} values. Using GU consistently outperforms the approach without GU for all λ/λ_{max} values, ranging from 0.1 to 0.9.

5.3 Ablation Study

To assess the effectiveness of PEMA, we conduct ablation studies to demonstrate the incremental improvement achieved by incorporating each component of PEMA. We utilize a token prediction decoder (B_{pd}) to predict the target token based on the context representation obtained from the pre-trained language model. As shown in Table 3, the token prediction decoder enhances task performance. Building on this, we incorporated Gradual Unrolling (GU) and the Reconstruction Decoder (B_{rct}) to further improve performance. The inclusion of these three methods yields the highest performance gains, as shown in the results.

Interpolation Parameter (λ_{max}) We propose the Gradual Unrolling (GU) interpolation strategy, where PEMA initially guides the generation of a new task and subsequently leverages the language model for contextual completion of sentences. Table 3 shows the effectiveness of GU in enhancing performance by enabling the language model to provide context completion. We further compare with and without GU by adjusting the λ_{max} hyperparameter in the WMT22 task. As shown in Figure 4, with GU maintains better performance stability at higher λ_{max} values while achieving noticeable performance improvement over without GU. We also report details on the impact of incorporating λ_{max} in Figure 5 in the appendix. Additionally, we conduct an experiment to measure perplexity. Table 4 shows that GU consistently outperforms across λ/λ_{max} values from 0.1 to 0.9.

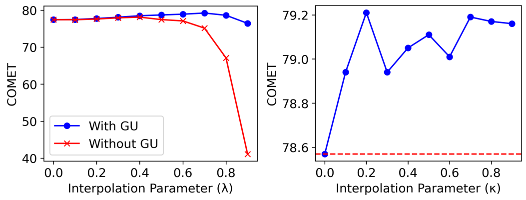

Figure 4: Performance variations on the WMT22 task with interpolation values λ_{max} (left) and κ (right). For λ_{max}, using Gradual Unrolling (GU) prevents performance degradation and enhances results, unlike without GU, where performance drops sharply. With κ when λ_{max} is set at 0.7, combining reconstruction loss with next-token prediction loss improves performance over excluding reconstruction loss (red dotted line), as indicated by better results when κ is above zero.

Interpolation Parameter (κ) We investigate the effectiveness of the reconstruction decoder, which reconstructs the original vector $f(c_i)$. Table 3 and Figure 4 demonstrate that incorporating the reconstruction decoder improves performance across desired tasks, demonstrating its efficacy in enhancing generation quality. We also report details on the impact of incorporating κ in Figure 6 in the appendix.

6 Conclusion

In this paper, we present PEMA, a novel parameter-efficient fine-tuning approach for language models. Unlike existing PEFT methods, PEMA utilizes minimal pre-trained model parameters during training, making it an efficient and adaptable method for offsite-tuning. PEMA includes a token prediction decoder, Gradual Unrolling, and a reconstruction decoder to improve model performance. Our comprehensive evaluations on translation and style transfer tasks demonstrate PEMA's effectiveness in generating text that more closely follows target domain distributions. Additionally, PEMA proves its computational efficiency by utilizing minimal training memory and achieving faster training latency with a syntactic dataset. Overall, PEMA offers efficient fine-tuning and presents a promising direction for an offsite-tunable PEFT approach in downstream NLP tasks.

Limitations

PEMA introduces a novel Parameter-Efficient Fine-Tuning (PEFT) method for privacy-preserving offsite-tuning. However, this process requires data owners to share predicted next-tokens with PLM owners during inference, which raises potential privacy concerns. These concerns necessitate further investigation of effective mitigation strategies.

Additionally, sharing the W_{hd} weight between PLM owners and data owners poses challenges related to model privacy. In our experiments, we used open-source PLMs due to the confidentiality issues associated with proprietary PLMs. Our future work will explore enabling data owners to generate a new Language Model (LM) head using a shared tokenizer from the PLM owner, enhancing privacy between the PLM and the data owner.

Finally, through PEMA, data and PLM owners can fine-tune efficiently and effectively with minimal communication. However, the way data owners use PEMA could unintentionally lead to data leakage issues. Subsequent research will explore solutions to address this challenge.

While our research has been focused on machine translation tasks, it can be applied to various NLP tasks depending on the initial input. Consequently, future studies will investigate the application of our method across a range of NLP tasks.

Ethics Statement

The results of our research are based on existing studies, and all generation models and datasets used are publicly available and used for their intended use with no ethical concerns.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

AnthropicAI. 2023. Introducing claude. https://www.anthropic.com/index/introducing-claude. Accessed: 2023-08-15.

Eleftheria Briakou, Di Lu, Ke Zhang, and Joel Tetreault. 2021a. Olá, bonjour, salve! XFORMAL: A benchmark for multilingual formality style transfer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3199–3216, Online. Association for Computational Linguistics.

Eleftheria Briakou, Di Lu, Ke Zhang, and Joel Tetreault. 2021b. Olá, bonjour, salve! XFORMAL: A benchmark for multilingual formality style transfer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3199–3216, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Justin Guinney and Julio Saez-Rodriguez. 2018. Alternative models for sharing confidential biomedical data. *Nature biotechnology*, 36(5):391–392.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210*.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Changwu Huang, Yuanxiang Li, and Xin Yao. 2020. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation*, 24(2):201–216.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*.

Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel, and Maja Popović. 2022. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. `https://online-chatgpt.com/`. Accessed: 2023-08-15.

OpenAI. 2023a. Fine-tuning - openai api. `https://platform.openai.com/docs/guides/fine-tuning`. Accessed: 2023-08-15.

OpenAI. 2023b. Gpt-4 technical report.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ellie Pavlick and Joel Tetreault. 2016. An Empirical Analysis of Formality in Online Communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

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.

Sundar Pichai. 2023. An important next step on our ai journey. `https://blog.google/intl/en-africa/products/explore-get-answers/an-important-next-step-on-our-ai-journey/`. Accessed: 2023-08-15.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5.

Matt Post. 2018. A call for clarity in reporting bleu scores. *WMT 2018*, page 186.

Sharpened Productions. 2023. Slang.net: The slang dictionary. https://slang.net/. Accessed: 2023-08-14.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ahmet Üstün and Asa Cooper Stickland. 2022. When does parameter-efficient transfer learning work for machine translation? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7919–7933, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *arXiv*.

Yahoo. 2007. L6 - yahoo! answers comprehensive questions and answers version 1.0. https://webscope.sandbox.yahoo.com/. Accessed: 2023-07-02.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Performance on Different Rank Sizes

Model	WMT22 (EN→DE)	GYAFC (F&R)	GYAFC (E&M)
OPT-1.3B (LoRA$_{r=8}$)	3.25	23.13	18.41
OPT-1.3B (LoRA$_{r=512}$)	4.28	20.98	15.57
OPT-1.3B (PEMA$_{r=8}$)	<u>11.75</u>	<u>56.29</u>	<u>54.22</u>
OPT-1.3B (PEMA$_{r=512}$)	**12.87**	**64.82**	**61.24**
LLaMA-7B (LoRA$_{r=8}$)	10.92	14.80	12.69
LLaMA-7B (LoRA$_{r=512}$)	<u>11.46</u>	<u>52.67</u>	<u>52.15</u>
LLaMA-7B (PEMA$_{r=8}$)	3.88	48.88	45.73
LLaMA-7B (PEMA$_{r=512}$)	**14.50**	**63.99**	**60.88**
OPT-30B (LoRA$_{r=8}$)	16.05	61.28	59.48
OPT-30B (LoRA$_{r=512}$)	16.03	61.39	59.76
OPT-30B (PEMA$_{r=8}$)	<u>18.33</u>	<u>62.87</u>	<u>60.12</u>
OPT-30B (PEMA$_{r=512}$)	**19.22**	**70.84**	**65.43**

Table 5: Experiment on LoRA and PEMA on meaning preservation (sBLEU) across rank variations ($r = \{8, 512\}$). The result shows PEMA consistently outperforms LoRA on sBLEU and COMET.

LoRA (Hu et al., 2022) states performance remains comparable with a small rank size. However, AdaLoRA (Zhang et al., 2023) finds a large rank size in the last layer of PLMs is needed for better performance. Performance evaluation on PEMA and baseline PEFT methods is conducted at the last layer of PLMs. For this reason, we set $r = 512$ for LoRA and PEMA to minimize the effect on performance with rank size. However, LoRA uses a rank size between 1 to 64 for their experiment. As PEMA is a LoRA-based PEFT method, we compared the performance on meaning preservation using the rank size employed in LoRA (8) and the rank size used in our experiment (512). As Table 5 shows, a larger rank size generally achieves favorable performance. In the case of LoRA, using a rank size of 512 outperforms 8 in 6 out of 9 cases. PEMA with a rank size of 512 performs better than PEMA with a rank size of 8 at all tasks.

B Measuring Informal Language Patterns

The GYAFC dataset for style transfer includes common informal input patterns that are frequently occur. To analyze the amount of mitigation, we categorize these patterns into four types. The four informal patterns are as follows. **Slang abbreviations**

	Informal Input	Formal Reference	Naive OPT-30B	kNN-LM	LoRA	UniPELT	Offsite-Tuning	PEMA
Family & Relationships								
Slang Abbreviation	525	307.75	346	339	356	322	361	**289**
All Capital	68	0	61	60	8	5	65	**3**
Redundant Word	39	2	1	1	2	**0**	17	3
Non-Capital Start	636	1.5	16	2	1	1	2	**0**
Entertainment & Music								
Slang Abbreviation	651	485.75	541	538	530	534	529	**463**
All Capital	36	0	31	34	9	9	37	**0**
Redundant Word	49	17.75	5	5	7	**3**	16	32
Non-Capital Start	655	7	24	2	**0**	1	3	**0**

Table 6: Count of informal patterns for each generated formal sentence. The result shows that PEMA performs better in mitigating informal patterns than baseline approaches. Lower is better.

are informal short forms of words or phrases (e.g., "LOL"-"laughing out loud"). To identify the presence of slang words, we check how many words from the predicted target sentence are present in the slang dictionary from (Productions, 2023). **All capital** is a pattern in which all characters in a generated word are capitalized (e.g., "FUNNY"). We calculate how many generated words are all capitalized. **Redundant word** occurs when two consecutive words are the same. For example, "I lie lie lie and then I lie some more." has two redundant words. **Non-capital start** is counted when a sentence does not start with a capital letter (e.g., "i only want points").

Table 6 shows the count of each informal pattern in generated sentences for both the baseline and PEMA. We also show an informal pattern count on informal input and formal reference. There are four reference sentences for each example in the test set. We show the average count for each pattern using the formal reference. It shows PEMA is good at mitigating slang abbreviation, all capital, and non-capital start compared to other baseline approaches. Interestingly, PEMA outperforms formal references in mitigating slang abbreviations and non-capital start.

C Dataset

C.1 Data Statistic

Table 7 shows data statistics of GYAFC and WMT22. For WMT22, we use a news-commentary v16 (EN→DE) for training. The test set for GYAFC has four references, while WMT22 has one reference for each test input.

Dataset	Train	Valid	Test	Length of $\mathcal{E}$
GYAFC (F&R)	51,967	2,788	1,332	691,531
GYAFC (E&M)	52,595	2,877	1,416	695,465
WMT22	388,482	2,203	1,984	20,983,482

Table 7: Data statistic of GYAFC and WMT22 with length of external memory $\mathcal{E}$.

Task	Example	
WMT22	English:	In better shape, but not alone.
	German:	In besserer Verfassung, aber nicht allein.
GYAFC	Informal:	I'd say it is punk though.
	Formal:	However, I do believe it to be punk.

Table 8: Example of parallel dataset GYAFC and WMT22.

C.2 Dataset Examples

Table 8 demonstrates examples of parallel datasets of GYAFC and WMT22.

C.3 Prompts

Table 9 presents prompt input used for evaluation. WMT22 and GYAFC have two placeholders. This includes [English Input] and [Informal Input]. [Generated Output] is a predicted output sentence generated by PLMs.

[English Input] represents the English input sentence in WMT22. [Informal Input] is the informal input sentence in GYAFC. An example of the parallel data input can be found in Table 8.

C.4 Post-processing

We use three decoder-based pre-trained language models for evaluation: OPT-IML-MAX-1.3B, LLaMA-7B, and OPT-IML-MAX-30B. These

Task	Prompt
WMT22	Translate this from English to German: [English Input] German: [Generated Output]
GYAFC	Convert the following informal sentence into a formal sentence: Informal: [Informal Input] Formal: [Generated Output]

Table 9: Prompt used for evaluation. [ ] represents the placeholder.

Model	Common hallucination patterns
OPT	I'm not sure . . . I 50% ... Convert the following informal sentence . . . Translate this from English to German: . . . I. . . . .
LLaMA	Informal: . . . ### . . . Comment: . . . \\[. . . \begin ... Answer: . . .

Table 10: Common hallucination patterns after generating a predicted sentence.

models are capable of generating tokens continuously. This characteristic makes decoder-based language models generate beyond the predicted sentences, typically called hallucinations. We find common hallucination patterns in each pre-trained language model. We post-process hallucinations generated after the predicted sentence for evaluation. Table 10 shows common hallucination patterns that are removed.

D Implementation Details

We use three RTX 8000 GPUs with 48GB GDDR6 memory for our experiment. For OPT-IML-MAX-1.3B, we use full precision (FP32) for training and inference. For LLaMA-7B and OPT-IML-MAX-30B, we use half-precision (FP16) and distribute the model across three GPUs using the HuggingFace Accelerate library. The hyperparameters for PEMA and the baselines are in Table 11. The best hyperparameter is selected using a grid search.

E Examples of Generated Outputs

The generated formal outputs of GYAFC are shown in Table 13 and Table 12. In WMT22, the German output generated is presented in Table 14. It shows PEMA understands the meaning of abbreviated

PEMA	
Random seed	123
Batch size	40,960
Adam lr	1e-03
Adam (β_1, β_2)	(0.9, 0.999)
Adam eps	1e-08
Number of rank	512
Optimal λ_{max}	0.7 to 0.9
Offsite-Tuning	
Random seed	42
Batch size	18
Emulator size	$\frac{1}{3}$ of PLM
Adam lr	1e-04
Adam (β_1, β_2)	(0.9, 0.999)
Adam eps	1e-08
LoRA	
Random seed	123
Batch size	10 to 30
Adam lr	1e-03
Adam (β_1, β_2)	(0.9, 0.999)
Adam eps	1e-08
Number of rank	512
LoRA α	1
Merge weight	FALSE
kNN-LM	
Random seed	1
Number of centroids learn	4,096
Quantized vector size	64
Number of clusters to query	32
Distance function	L2 Distance
UniPELT	
Random seed	123
Batch size	10 to 30
Adam lr	1e-03
Adam (β_1, β_2)	(0.9, 0.999)
Adam eps	1e-08
Prefix gate	True
Prefix length	10
Prefix mid dimension	512
LoRA gate	True
Number of rank	10
LoRA α	16
Adapter gate	True
Adapter down sample	$D_{hid}/2$
Used PEFT methods	Adapter Prefix tuning LoRA

Table 11: Hyper-parameter setup of each baseline method. We select the batch size between 10 to 30. D_{hid} represent hidden size of a model.

format (e.g., translating "5'4" to "5 feet 4 inches"), or removing the informal word (e.g., "flirt" which typically refers to playful or teasing behavior). Mit-

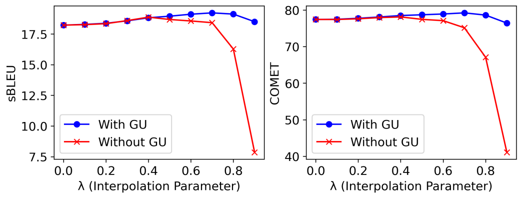

Figure 5: Performance variation for each interpolation value λ_{max} in the WMT22 task. With both Gradual Unrolling (GU) (blue) and without GU (red), there is a decline in performance at a specific point of λ_{max}. However, when utilizing GU, the model is not only robust to performance degradation but also gains performance improvement.

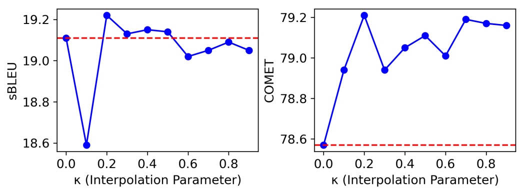

Figure 6: Impact of mixing ratio values between reconstruction loss and predicting the next-token loss in the WMT22 task. When κ is 0, it means excluding reconstruction loss (red dashed line). We fix the λ_{max} value as 0.7. The graphs show that combining reconstruction loss and predicting the next-token loss is superior to excluding reconstruction loss.

igating common informal patterns such as all capital words (e.g., "PINK FLOYD" to "Pink Floyd") while preserving the meaning of input (e.g., "Wir" means "We" in German).

F Difference Between PEMA and LoRA at W_{hd}

Applying LoRA to $W_{hd} \in \mathbb{R}^{v \times d}$, a larger set of parameters is required due to the difference in input and output sizes (d and v). Conversely, PEMA operates more efficiently, utilizing computation resources by receiving an input of size d and yielding an output of the same size. For instance, OPT-1.3B has $d = 2,048$ and $v = 50,272$.

G Impact on Interpolation λ and κ

In the WMT22 task, we observe performance variation with different interpolation values, λ_{max} in Figure 5. Additionally, we investigate the impact of the mixing ratio values between reconstruction loss and predicting the next-token loss in Figure 6.

H Licensing Information

Models OPT is licensed under the MIT License. The LLaMA is licensed under the GNU General Public License (GPL) version 3.
Fine-tuning Methods kNN-LM, LoRA, and Offsite-Tuning are licensed under the MIT License. UniPELT is licensed under the Creative Commons Attribution-NonCommercial (CC-BY-NC) license.
Dataset GYAFC is based on the Yahoo Answers corpus (L6 - Yahoo! Answers Comprehensive Questions and Answers version 1.0) (Yahoo, 2007), and is designated for research purposes. Access to the GYAFC dataset requires prior access to Yahoo Answers corpus. WMT22 is freely available for research purposes for academic and educational activities.

Input		he is probably wondering if your interested in him at all....flirt back!!	sBLEU
Reference	1	He is likely wondering if you are interested in him at all; Flirt back with him.	
	2	He probably wants to know if you're interested in him.	
	3	He is probably wondering if you are interested in him at all, so flirt back.	
	4	He is probably wondering if you are interested in him at all. Flirt back.	
Output	PEMA	He is probably wondering if you are interested in him at all.	100.0
	LoRA	He is probably wondering if you are interested in him at all. If you are interested, flirt back.	66.78
	kNN-LM	It is most likely that he is wondering if you are interested in him at all....flirt back!!	42.60
	UniPELT	He is probably wondering if your interested in him at all....flirt back!	50.82
	Offsite-Tuning	He probably is wondering if you are interested in him at all. Flirt back!!	72.98
	Naïve OPT-30B	In informal situations he is probably wondering if your interested in him at all.	46.03
Input		I don't know!...I just want the points...lol	
Reference	1	I only want points.	
	2	I do not know. I merely want the points.	
	3	I do not know; I just want the points.	
	4	I do not know, I only want the points.	
Output	PEMA	I do not know, but I just want the points.	73.49
	LoRA	I don't know!... I just want the points. I am not sure what I am doing.	25.31
	kNN-LM	I don't know!...I just want the points...lol	34.90
	UniPELT	I don't know!...I just want the points...lol	34.90
	Offsite-Tuning	-	0.00
	Naïve OPT-30B	I don't know!...I just want the points...lol	34.90
Input		No way im 5'4 and he's 6'2	
Reference	1	No, I am 5ft 4inches and he is 6ft and 2inches.	
	2	No way, I am only 5'4" and he is 6'2".	
	3	Not at all. I am five feet four inches tall and he is 6 feet 2 inches tall. '	
	4	No chance, I am five feet four inches tall and he is six feet two inches tall.	
Output	PEMA	No way, I am 5 feet 4 inches tall and he is 6 feet 2 inches tall.	74.44
	LoRA	No way, I am 5'4 and he is 6'2.	51.52
	kNN-LM	No way, I am 5'4 and he is 6'2	50.05
	UniPELT	No way, I am 5'4 and he is 6'2	50.05
	Offsite-Tuning	No way im 5'4 and he's 6'2.	7.78
	Naïve OPT-30B	No way, I am 5'4 and he is 6'2	45.72

Table 12: Examples of generated formal output of GYAFC (Family & Relationships) for given informal input. One interesting example is PEMA can understand the meaning of abbreviated height descriptions like "5'4" and "6'2". And rewrite them into more formal forms "5 feet 4 inches" and "6 feet 2 inches".

Input		I respect that tyra banks is trying show hosting..but the big O [aka OPRAH] won...	sBLEU
Reference	1	While I respect that Tyra Banks is branching out into television hosting, Oprah Winfrey is the better example.	
	2	I respect the hostess Tyra Banks, but I prefer Oprah.	
	3	I admire Tyra Banks for trying to become a tv show host. However, Oprah is clearly the best hostess.	
	4	I respect that Tyra Banks is trying out show hosting, but I think that Oprah won.	
Output	PEMA	I respect that Tyra Banks is trying to show hosting..but the big O [aka Oprah Winfrey] won.	35.56
	LoRA	I respect that tyra banks is trying show hosting, but the big O won.	24.32
	kNN-LM	I respect that tyra banks is trying show hosting..but the big O [aka OPRAH] won.	12.02
	UniPELT	I respect that tyra banks is trying show hosting..but the big O won...	13.40
	Offsite-Tuning	I respect that tyra banks is trying show hosting..but the big O [aka OPRAH] won...	10.90
	Naïve OPT-30B	In informal: I respect that tyra banks is trying show hosting..but the big O [aka OPRAH] won...	9.56
Input		Not sure what's gonna happen, sure it will be good and OH YEAH!!!!!!!	
Reference	1	I am not sure what is going to happen but I am sure it will be good.	
	2	I'm not sure what is going to happen, I'm sure it will be good.	
	3	I am not sure what will happen, but I am sure it will be good. Oh, yes.	
	4	I am not sure what will happen, however, I am sure that it will be good.	
Output	PEMA	I am not sure what is going to happen, but I am sure it will be good and OH YEAH!!!!!!!	61.11
	LoRA	Not sure what's going to happen, but I am sure it will be good and OH YEAH!!!!!!!.	43.05
	kNN-LM	Not sure what's gonna happen, sure it will be good and OH YEAH!!!!!!!	20.22
	UniPELT	Not sure what's going to happen, but	16.74
	Offsite-Tuning	Not sure what's gonna happen, sure it will be good and OH YEAH!!!!!!!	20.22
	Naïve OPT-30B	Not sure what's gonna happen, sure it will be good and OH YEAH!!!!!!!	20.22
Input		Journey, PINK FLOYD, The POLICE, The EAGLES & RUSH... omg!	
Reference	1	Journey, Pink Floyd, The Police, The Eagles, and Rush - oh my!	
	2	I like Journey, Pink Floyd, The Police, The Eagles, and Rush.	
	3	Oh goodness, Journey, Pink Floyd, The Police, the Eagles, and Rush!	
	4	Journey, Pink Floyd, The Police, The Eagles, and Rush are all great classic bands.	
Output	PEMA	I love Journey, Pink Floyd, The Police, The Eagles and Rush.	69.01
	LoRA	Journey, PINK FLOYD, The Police, The Eagles & Rush. I love it!	36.45
	kNN-LM	Journey, PINK FLOYD, The Police, The Eagles & Rush... omg!	35.66
	UniPELT	Journey, PINK FLOYD, The Police, The Eagles & Rush... omg!	35.66
	Offsite-Tuning	Journey, Pink Floyd, The Eagles, Rush, and The Police. Oh my god!	47.29
	Naïve OPT-30B	Journey, PINK FLOYD, The Police, The Eagles & Rush... omg!	35.66

Table 13: Examples of generated formal output of GYAFC (Entertainment & Music) for given informal input. It shows that PEMA is capable of restoring All Capital patterns to their formal format. For example, PEMA successfully restore "Oprah Winfrey" given "OPRAH" as an input.

Input		In Berlin he trained as an opera singer and in 1986 he founded the Palast Orchester together with his classmates.	sBLEU
Reference		In Berlin studierte er Operngesang, 1986 gründete er zusammen mit Kommilitonen das Palast Orchester.	
Output	PEMA	In Berlin studierte er als Opernsänger und gründete 1986 mit seinen Schülern das Palast Orchester.	33.16
	LoRA	In Berlin wurde er als Opernsänger trainiert und 1986 gründete er mit seinen Schülern das Palast Orchester.	22.79
	kNN-LM	In Berlin schloss er 1986 mit seinen Schülern die Palast-Orchester zusammen.	6.64
	UniPELT	In Berlin trainierte er als Opernsänger und gründete 1986 mit seinen Schülern das Palast Orchester. When	19.35
	Offsite-Tuning	ich war in Berlin als Opernstätte und 1986 gründete er die Palast Orchester gemeinsam mit seinen Kollegen.	11.31
	Naive OPT-30B	In Berlin wurde er als Opernsänger trainiert und 1986 gründete er mit seinen Schülern das Palast Orchester.	22.79
Input		Since then the baritone has established himself with his repertoire of hits and chansons in the style of the Comedian Harmonists.	
Reference		Seitdem hat sich der Bariton mit seinem Repertoire aus Schlagern und Chansons im Stil der Comedian Harmonists etabliert.	
Output	PEMA	Seitdem hat er sich mit seinem Repertoire aus Hits und Chansons im Stil der Comedian Harmonists etabliert.	60.26
	LoRA	Seitdem ist er mit seinem Repertoire aus Hitparaden und Chansons im Stil der Comedian Harmonists aufgestiegen.	46.00
	kNN-LM	Seitdem ist er mit seinem Repertoire aus Hitparaden und Chansons im Stil der Comedian Harmonistslipperlipper dieserenedenauit festgelegt.	39.81
	UniPELT	Seitdem ist der Bariton mit seinem Repertoire aus Hitparaden und Chansons im Stil der Comedian Harmonists auf dem Gebiet der Kulturlandschaft in der Schweiz etabliert.	42.85
	Offsite-Tuning	ich habe seitdem die Baritone seine Rekorde aus der Stil der Comedian Harmonisten errichtet.	8.91
	Naive OPT-30B	Seitdem ist er mit seinem Repertoire aus Hitparaden und Chansons im Stil der Comedian Harmonists auf dem Gebiet der Baritonen geschrieben.	39.81
Input		We have the book Terra, but I am generally not very satisfied with the assignments in the book.	
Reference		Wir haben das Buch Terra und mit den meisten Aufgabenstellungen im Buch bin ich eher unzufrieden.	
Output	PEMA	Wir haben das Buch Terra, aber ich bin im Allgemeinen nicht sehr zufrieden mit den Aufgaben in dem Buch.	22.37
	LoRA	ich habe das Buch Terra, aber ich bin im Allgemeinen nicht sehr zufrieden mit den Aufgaben in dem Buch.	10.11
	kNN-LM	ich habe das Buch Terra, aber ich bin im Allgemeinen nicht sehr zufrieden mit denenteilen in dem Buch.	9.38
	UniPELT	ich habe das Buch Terra, aber in der Regel bin ich nicht sehr zufrieden mit den Aufgaben in dem Buch.	10.06
	Offsite-Tuning	ich habe die Buch Terra, aber ich bin allgemein nicht sehr begeistert mit den Schreibungen in der Buch.	6.44
	Naive OPT-30B	ich habe das Buch Terra, aber ich bin im Allgemeinen nicht sehr zufrieden mit den Aufgaben in dem Buch.	10.11

Table 14: Examples of generated German output in WMT22 test set. The result shows that PEMA is capable of generating German output that preserves its meaning.