SPA: Towards A Computational Friendly Cloud-Base and On-Devices Collaboration Seq2seq Personalized Generation

Anonymous submission

Abstract

Large language models(LLMs) have shown its outperforming ability on various tasks and question answering.However, LLMs require high computation cost and large memory cost. At the same time, LLMs may cause privacy leakage when training or prediction procedure contains sensitive information.In this paper, we propose SPA(Side Plugin Adaption), a lightweight model for fast on-devices inference and privacy retaining on the constraints of strict on-devices computation and memory constraints. Compared with other on-devices seq2seq generation, SPA could make a fast and stable inference on privacy constraints, allowing it to obtain cost effiency. Our method establish an interaction between a pretrained LLMs on-cloud and additive parameters on-devices, which could provide the knowledge on both pretrained LLMs and private personal feature.Further more, SPA provides a framework to keep feature-base parameters on private guaranteed but low computational devices while leave the parameters containing general information on the high computational devices.

1 Introduction

Large Language Models (LLMs)(Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023) have demonstrated incredibly powerful capabilities in supporting downstream tasks. These models perform exceptionally well across various downstream tasks. Additionally, different LLMs such as GPT-4(OpenAI, 2023), Llama(Touvron et al., 2023), and ChatGLM(Zeng et al., 2022; Du et al., 2022) exhibit similar abilities in handling tasks across multiple dimensions(Bang et al., 2023). However, deploying and computing LLMs on resource-constrained devices introduces significant memory and computational pressure on these devices(Alizadeh et al., 2023). Therefore, researching reliable methods for deploying LLMs on-devices becomes critically urgent.

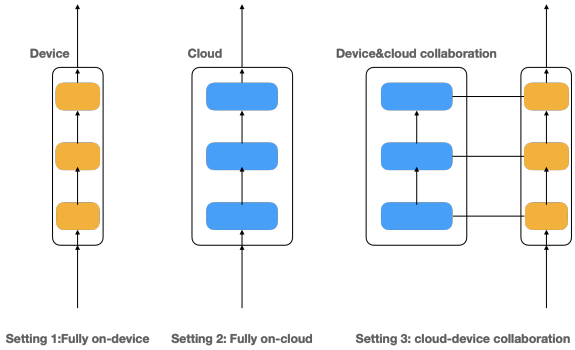

Figure 1: Three setting for model deployment.

To address the issue of model deployment on-devices, several methods have propose different approachs to adapt LLMs on edge devices. Subformer(Reid et al., 2021a) reduces the parameter sizes the model needs to store and the computational FLOPS required by sharing parameters. Similarly, EdgeFormer(Ge et al., 2022) further explores the impact of different architectures on model deployment on-devices and proposes a balanced sharing scheme. LLMCad(Xu et al., 2023) utilizes a collaborative model configuration mode to achieve lightweight model deployment, where a smaller LLM generates the simplest words, while a high-precision LLM verifies these words and corrects errors.

Furthermore, prior researches provide some useful improvements in parameter efficient fine-tuning LLMs(Houlsby et al.; Hu et al., 2021; Lee et al., 2022). These methods efficiently utilize the additional parameter space to adapt LLMs to downstream tasks flexibly while retaining the inherent capabilities of the models. Notably, efficient fine-tuning techniques for large language models are based on adapter strategies, where adapters integrate with the base model to produce effective capabilities for the required downstream tasks. This suggests that in fine-tuning large language models, adapters can be separated from the core model, en-

abling storage separation of these components(Hu et al., 2023). This separation brings significant advantages in device storage, as only a small portion of the original model parameters needs to be stored, facilitating the retention of adapters tailored for specific downstream tasks.

Considering the characteristic of fewer adapter parameters, deployment on-devices can use adapters as incremental deployments(Lin et al., 2020), with adapters serving as learned components for downstream task. It integrates incremental knowledge into the architecture of the existing model to obtain inherent characteristics of users. Leveraging the additional adapters, effective integration between this incremental knowledge and the base large language model can strongly support downstream tasks and encapsulate the process of obtaining user attributes. However, this design aspect needs to consider the requirements of inter-device data transmission(Ren et al., 2019). When deploying adapters on-device, complexities and inference delays related to inter-device communication arise due to the isolation of the language model and its adapters on different devices. Addressing how to reduce the latency of transferring this portion of data presents a major challenge for designs of separated models.

Our research proposes a method called **S**ide **P**lugin **A**daptation (**SPA**), which achieves the deployment of large models on-devices by separating adapters from pre-trained models. Specifically, our method introduces an side adapter classifier. This classifier allows us to choose between leveraging the inherent capabilities of the original model and integrating feature information generated by adapters. In this way, we aim to minimize the frequency of adapter usage while maximizing the utilization of common-sense knowledge inherent in the original large language model. This strategy offers a dual advantage: leveraging the unique feature attributes of adapters while maximizing the inherent capabilities of a widely based model. Compared to traditional on-devices model setups, our method significantly improves inference speed.

We comprehensively evaluate our approach on various datasets across different domains. These evaluations not only emphasize the advantages of our method compared to traditional approaches but also highlight its potential in improving overall model capability while achieving significant adapter effects. In addition to enhancing text generation and semantic understanding services, our approach demonstrates a stronger ability to understand textual features, better meeting user needs.

Our contributions can be summarized as follows:

- We innovatively propose the SPA framework, deploying large models on-device via adapters. Our method significantly reduces the difficulty of fully deploying large models on the edge and presents a feasible and efficient solution.

- We introduce a classifier to determine whether to use adapter content for inference. This innovative classifier design significantly improves the efficiency of cloud server to edge transmission while maintaining task performance.

- SPA has been thoroughly tested across multiple datasets. Experiments show that our framework performs well on-devices. Additionally, SPA-LST can achieve results close to the original model with lower transmission latency.

2 Related Work

2.1 Parameter-Efficient Transfer Learning

In recent years, efficient parameter-efficient transfer learning has become a popular method for fine-tuning large language models. It achieves performance close to full fine-tuning by refining a subset of parameters. Among these methods, adapters, as introduced by (Houlsby et al., 2019), incorporate two additional feed-forward layers, enhancing their adaptability to downstream tasks. Similarly, LoRA (Hu et al., 2021) employs a rank reduction followed by rank promotion method on the original language model. In the context of language model fine-tuning, prefix tuning (Li and Liang, 2021) involves introducing a trainable prefix vector before the keys and values in the multi-head attention mechanism, producing effects similar to the first two methods.

The MAM adapter (He et al., 2021) synthesizes these techniques and identifies superior results in parameter fine-tuning. Moreover, LST (Sung et al., 2022) provides a method for saving space during parameter fine-tuning. It achieves optimal parameter fine-tuning with the same memory consumption by leveraging adapters on the device. This approach significantly reduces memory usage when training large language models on edge devices. Meanwhile, the QLora (Dettmers et al., 2023) method combines quantization rules with

LoRA's rank promotion and demotion scheme, optimizing training resources and space savings.

Furthermore, the CuQA (Lee et al., 2022) method offers a classifier that selects between the original language model and edge models based on the relevance of training and testing data during training. Through this selection process, we observe that efficient parameter fine-tuning methods can effectively embed new knowledge, thereby expanding existing language models to incorporate new knowledge bases in support of specific tasks. Our method leverages the parameter-saving capabilities of efficient parameter-efficient transfer learning and the adaptive information contained in parameter fine-tuning inspired by CuQA. We provide robust support for various generative tasks, demonstrating enhanced capabilities.

2.2 On-device ML optimization

For on-device sequence-to-sequence models, various techniques have been adopted to reduce the parameters of pre-trained models, including methods such as quantization(Liu et al., 2018; Li et al., 2023) to reduce model size, knowledge distillation(Rao et al., 2023; Wang and Yoon, 2021) techniques that learn from large models and transfer knowledge to smaller models, and techniques such as compression(Frantar et al., 2022; Pope et al., 2023) and pruning(Bolya et al., 2022; Cai et al., 2022; Kim et al., 2022) to reduce the number of model parameters. Some methods like Edgeformer (Reid et al., 2021b) achieve model reduction by sharing model parameters. Low-rank decomposition is also used to reduce model size (Cahyawijaya, 2021). Methods like Delight (Mehta et al., 2020) achieve model size reduction by compressing attention mechanisms. These techniques are adjusted and adapted based on the parameters of the Transformer (Vaswani et al., 2017) architecture, and they yield effective parameter reduction results for models with relatively fewer parameters.

Given that a considerable portion of these methods can only be applied to models with small model, they may not produce substantial optimizations in the context of large models. Computational and storage resources on devices may not be sufficient to accommodate LLMs. Therefore, we propose a framework for hybrid cloud server and on-device sequence-to-sequence model. The framework aims to meet the needs of certain downstream tasks and user characteristics by combining the capabilities of the cloud and edge.

3 Side Plugin Adaption

Given the computational and storage capabilities on the edge, and the need to ensure strong data privacy during model inference, our concerns can be broken down into the following points:

- How much reduction in computation time and space cost can our model achieve?
- Does its capability improve compared to the original model under these costs?
- Is there a possibility of information leakage between the edge and cloud models during their interactions?
- Whether our network latency matches expectations, and whether the overall latency significantly decreases compared to the original model's cost.

These are some fundamental requirements and abilities for a small on-device model that guarantees data privacy. Considering the need to learn user features and context-related information for text generation, we propose adding a classifier after the output of the transformer. We explore two settings for this classifier, comparing them to the original edge adapter's results to assess downstream task support.

In the first setting, a binary choice approach is used, where the edge model is trained from fine-tuning and the output is generated either the origin LLM or side block results. The classifier selects the side whose output better aligns with the desired features, enabling the classifier to choose that path.

The second setting, called the plugin method, involves using the classifier to decide whether to combine the edge model with the base model stored in the cloud. This benefits by allowing the edge model to learn only the differences in features relevant to the current task. General information is then obtained by utilized the base model's results,which reduced latency cost on edge devices resources.

Specifically, The function f is the single layer on origin LLMs, and the function g is the corresponding layer on side model. the methods above could conclude to these two functions:

$$h = f(x) + \sigma g(x) \tag{1}$$

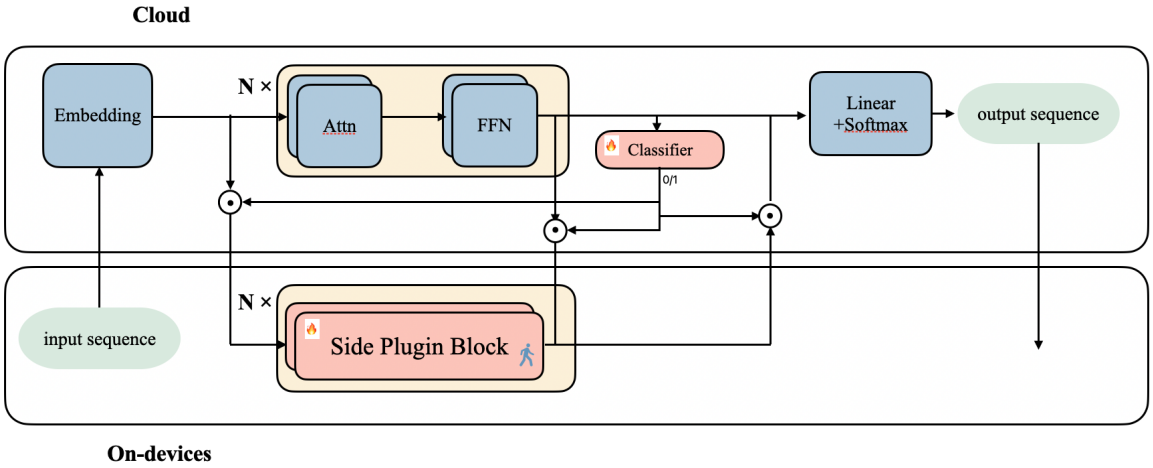

Figure 2: Cloud-base and on-devices collaboration model:SPA

$$\sigma = \arg\max \mathbf{M}_c f(x) \tag{2}$$

We employed a linear layer of $H \times 2$ as our classifier. The advantage of this design lies in our ability to use the Transformer's output as the hidden state, which is then softmax to a range between [0, 1], thereby obtaining a specific distribution for classification choices σ. Through such a linear mapping, the model can learn whether predicting a certain token is better accomplished by selecting the existing model's output (typically leveraging previously predicted information like "New + York"), or by employing a side-end incremental model (which relies more on user-specific feature information), thus determining the optimal choice.Considering σ is learnable, the loss function in the whole model could be depicted as:

$$\mathcal{L}_{LM} = -\sum_{i=1}^{T} \log_{(\Theta+\sigma\Delta\Theta)} \mathbf{P}(x_i|x_{<i}) \tag{3}$$

Here we keep $f(x)$ frozen and train the side model $g(x)$ and classifier σ. Unlike a fixed classifier σ' for text similarity on the original model, our linear-layer-based classifier σ can determine if a predicted word exists in the knowledge base of the base language model. This approach reduces inference times for highly repetitive word sequences, storing more general information in the cloud and generating user-specific features on the edge device.

From a privacy view, this classifier ensures that the cloud can only access the current classifier binary choice information, not the edge model's parameters. Thus, during inference, the cloud only receives a signal to involve the edge when necessary, significantly enhancing personal data privacy. Compared to EdgeFormer, our model's main bottleneck lies in the communication latency (Ren et al., 2019) between the edge and server. This can be managed by controlling communication frequency and times for an optimal value.

EdgeFormer (Reid et al., 2021b) scales down the original model by factors like four or eight, sharing parameters to achieve a smaller model with similar capabilities and fewer FLOPs. However, this design demands greater computational and proficiency requirements for edge devices. While our methods set the edge model small, its prediction speed relative to total latency is quite small. In this case, the edge devices computational ability would not be a bottleneck for predicting the results on-devices.

For communication between cloud servers and edge models, we adopt the architecture used in LST. LST improves memory capacity from back propagate. For prediction, if our transformer has L layers, Parallel adapter (He et al., 2021) or LoRA methods would cause a large cost on communication latency. Specifically, the latency including three parts. $T_{pretrained}$ indicates the original pretrained model computing latency. $T_{on-devices}$ indicates the edge devices model computing latency, it could be Adapter part, LoRA part or LST part for on-devices inference. What's more, T_{net} is the network latency when transmitting data between cloud and devices. Then, the total latency could be evaluated as:

$$T_{total} = T_{on-devices} + T_{pretrained} + T_{net} \tag{4}$$

Model	XSum	CNN-DM	CoQA	SciQ
LLaMA-7B + one-shot	17.32	22.36	15.32	17.21
LLaMA-7B + LST	28.18	32.15	31.24	23.24
LLaMA-7B + SPA	**35.52**	**39.22**	**37.30**	**25.38**

Table 1: The Rouge-L scores on LST model, one-shot instruct and SPA model on various datasets

While $T_{pretrained}$ keep the same for identical model.For model on edge devices, the inference latency is lay on the CPU or GPU computing capability, denoted as f^e. F_{data} is the FLOPs required to compute on edges, $C_{devices}$ indicates the numbers of CPU or GPU are available to compute on this model. Then:

$$T_{on-devices} = \frac{F_{data} \times C_{devices}}{f^e} \tag{5}$$

For the network latency when transmitting data between cloud and devices. Assume the TCP/IP connection latency is τ, The T_{net} could be formulated as:

$$T_{net} = M \times (\tau + T_{data}) \tag{6}$$

Where M is the transmission times for one token prediction, it would be changed for different architecture, because we have to transfer back the hidden state from side model to base model. LST effectively merges the process in scenarios where edge-cloud connectivity is involved.

4 Experiments Setup

4.1 Datasets

We evaluated our approach on the following training dataset.

1. **XSum** (Narayan et al., 2018): There are 227,000 summaries of online articles generated by the BBC available on XSum, and we assessed the relationships between the articles and their corresponding summaries.

2. **CNN Daily Mail** (Chopra et al., 2016): The CNN Daily Mail dataset consists of 312,000 summaries of CNN Daily Mail articles and its origin articles.

3. **CoQA** (Reddy et al., 2019): CoQA datasets contains 127,000 questions encompassing 8,000 conversation-based question-answer pairs, constituting a large-scale QA system spanning various domains.

4. **SciQ** (Welbl et al., 2017): The SciQ dataset comprises 13,679 publicly available scientific exam questions covering diverse disciplines such as biology, chemistry, physics, and others. Answers to specific questions are obtained through contextual information.

4.2 Implement Details

For each dataset, we uniformly trained for 15 epochs to enhance the adapter's learning capabilities for tasks or text features. Simultaneously, when evaluating our sequenced model, we conducted branch selection checks on the evaluated content to observe whether the model opted for adapters for adaptation. We also statistically analyzed the proportion of adapter usage. We compared our models under consistent trainable parameters, setting the learning rates to $\{ 2 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3} \}$ for each model to determine the optimal learning rate for that dataset. Throughout the training process, we maintained a consistent batch size of 8 and conducted training across an 8 Nvidia RTX V100 machines.Table 1 shows the comparison between our proposed model and the baseline model, as well as the LST model, demonstrating the differences in performance across various datasets. We utilize beam search for tokens generation.

5 Experiments

The performance of SPA surpasses that of LST. Table 1 shows that, under the premise of lightweight parameters, the blocks trained by SPA can effectively guide the original model. Consequently, even with a smaller computational load at the edge, the model can exhibit strong inference capabilities and enhance performance for the given task.

In the model we've designed, network transmission is a crucial aspect, particularly in the context of SPA. Therefore, as the discussion in Section 3, it's essential to compare the previous model with the SPA model and estimate the corresponding latency required for predicting the entire model. Given that our model is relatively large, we cannot deploy the entire model at the edge as in the case of Edgeformer. In the collaborative process between cloud-based and on-device components, transmitting critical information is necessary. Consequently, we will be conducting inference latency experiments to assess this aspect.

Model	% Param	Infer. latency	Net. latency	Total latency	M	XSum	CoQA
LLaMA-7B + LoRA	0.19	**3.26**	6.37	9.63	32.0	**38.24**	40.32
LLaMA-7B + Adapter	0.38	3.32	12.56	15.88	64.0	38.18	**40.51**
LLaMA-7B + LST	**0.2**	3.29	0.31	3.60	1.0	28.78	31.24
LLaMA-7B + SPA	0.21	3.30	**0.18**	**3.48**	**0.62**	35.52	37.30

Table 2: Latency evaluated from Different efficient tuning methods.Measuring new parameters percentage rate, Inference latency, Network latency and Total latency on prediction for 50 new tokens. M indicates the ratios of direct transmission times and pretrained model inferencing times. The Rouge-L scores on different PEFL methods and our model on various datasets.

Model	M	XSum	CNN-DM	CoQA	SciQ
LLaMA-7B + S1	1.0	**36.24**	**40.55**	**38.81**	**25.91**
LLaMA-7B + S2	0.68	30.29	34.19	32.11	23.71
LLaMA-7B + S3	1.0	28.18	32.15	31.24	23.24
LLaMA-7B + SPA	**0.62**	35.52	39.22	37.30	25.38

Table 3: The Rouge-L scores on different settings and SPA model on various datasets.

Model	% Param	M	Diff rates	Diff Pick rates
LLaMA-7B + SPA	0.21	0.62	78.2	56.8

Table 4: The Side Model Effect evaluation.

Data sizes	Xsum	CNNDM	CoQA	SciQ
SPA_{small}	10000	20000	4000	5000
SPA_{medium}	100000	150000	40000	10000
SPA_{full}	227000	312000	127000	13679

Table 5: Different training data sizes on datasets.

5.1 Latency on Prediction

To address concerns about network latency and overall inference delay, we conducted comparative experiments on data exchange for these models. Additionally, we compared the original Adapter, LoRA, and LST methods, evaluating them based on data exchange times and overall model latency. We performed these assessments using the same training configuration as mentioned above.

We also uniformly evaluated the time taken to generate 50 tokens. Additionally, our single-sided model and the baseline model were placed within a server cluster to simulate network transmission delays in an ideal network environment. This was done to assess the corresponding network transmission latency.The respective metric results obtained from the testing are recorded in Table 2.the difference between single and multiple communication latency is substantial, with inference costs relying mainly on cloud server computing power.

The network latency for the SPA model is superior compared with other methods. It's noticeable that SPA exhibits favorable characteristics. It can employ the classifier to choose between a pre-trained model or a model with an integrated adapter for predictions. Due to the LST structure employed on-devices, it offers a more independent architecture compared to methods like LoRA and Adapters.

5.2 Impact of Side Model

We have assessed the impact of the edge model on the overall model and analyzed the word repetition rates between the edge model and the original model. Based on this analysis, we evaluate whether the results of our model meet our expected outcomes when selecting the edge model for inference. This metric can also be used to assess the extent of the difference between the new model and the original model and the proportionality with the edge model we have chosen. We show the evaluation of side model effect on Table 3.

We could find that while the origin cloud model and on-devices model generate different results, the classifier σ would make a option that choose a more ideal information from the two alternations. When it comes to a same result, the classifier would choose to pick the cloud one for less network communication cost.

5.3 Analysis on Training Data Sizes

To assess the impact of training data sizes on model performance, we train the SPA model and utilize different sizes of training sets, as specified below:

We trained the models using the same network architecture and hyperparameters, with a consistent training epoch of 15 rounds. Text generation was performed using beam search.

From the results, it is evident that the model's performance improves across multiple datasets with an increase in training data sizes, especially from small to medium data. This indicates that augmenting training data enhances the model's

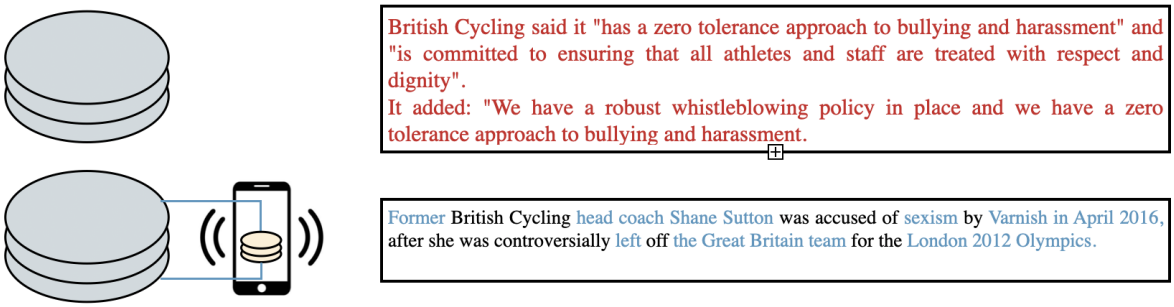

Figure 3: The summary case study on classifier option, The pretrained model choices were marked as blue, while SPA model choices were marked as black.

Data sizes	Xsum	CNNDM	CoQA	SciQ
SPA_{small}	31.5	36.2	34.7	22.1
SPA_{medium}	33.8	38.4	36.2	23.8
SPA_{full}	35.5	39.2	37.3	25.4

Table 6: The Rouge-L scores on different training data sizes on various datasets.

Data sizes	SPA_{small}	SPA_{medium}	SPA_{full}
Usage Percentage	82.3	71.5	61.8

Table 7: Evaluations on side model ratios.

generalization ability. However, as data sizes continues to increase to the full set, the performance improvement becomes less pronounced, indicating the presence of marginal effects. Considering the escalating computational resources, medium data size offers a favorable performance-to-cost ratio.

Additionally, we recorded the proportion of adapter usage by the model, as shown in Table 7. As data sizes increases, the percentage of adapter usage decreases, with a higher probability of selecting the original model's output. This further validates that increasing data sizes enhances the model's inherent language understanding and generation capabilities.

Increasing training data sizes significantly improves downstream task performance. However, there is a diminishing marginal effect with increased data sizes. Considering computational resources, a medium data size provides the best performance-to-cost ratio. More data reduces reliance on adapters, improving the model's language understanding capabilities.

5.4 Different Setting on Side Plugin

SPA outperforms both LST and the pre-trained model, showcasing better expressive capabilities. In order to identify the source of this enhanced performance, we conducted ablation experiments.

We present three configurations in our study. The three configurations we marks from S1 to S3. In the first setup, referred to as "Full Integration," the edge adapter directly interfaces with the original model. This arrangement allows the edge adapter to effectively learn and correct errors within the original model.In addition, the setup of S1 can be viewed as a parallelized LST setup, which has the advantage that he may perform better on generative tasks than the original serial option of LST.

In the second setup, a choice is made between the edge-adapted model and the original model. The classifier utilizes the original model's output to make decisions between the results of the original model and the edge-adapted model. In this case, the edge on-devices model operates with a higher degree of independence from the original model.

The final setup involves the existing LST model, which can be viewed as relying entirely on the edge adapter for result generation.

By comparing our model with these three setups, we aim to determine which configuration best aligns with the requirement for consistency between efficiency and capability that the model needs to achieve.The capabilities demonstrated by different experiments are summarized in Table 3.

The model's enhancement in downstream tasks primarily relies on the adjustments made by the adapter to capture the incremental differences. As illustrate on Table 3, While the introduction of the classifier in the SPA model leads to a slight performance drop compared to the scenario where adapters are integrated in all cases, the per-

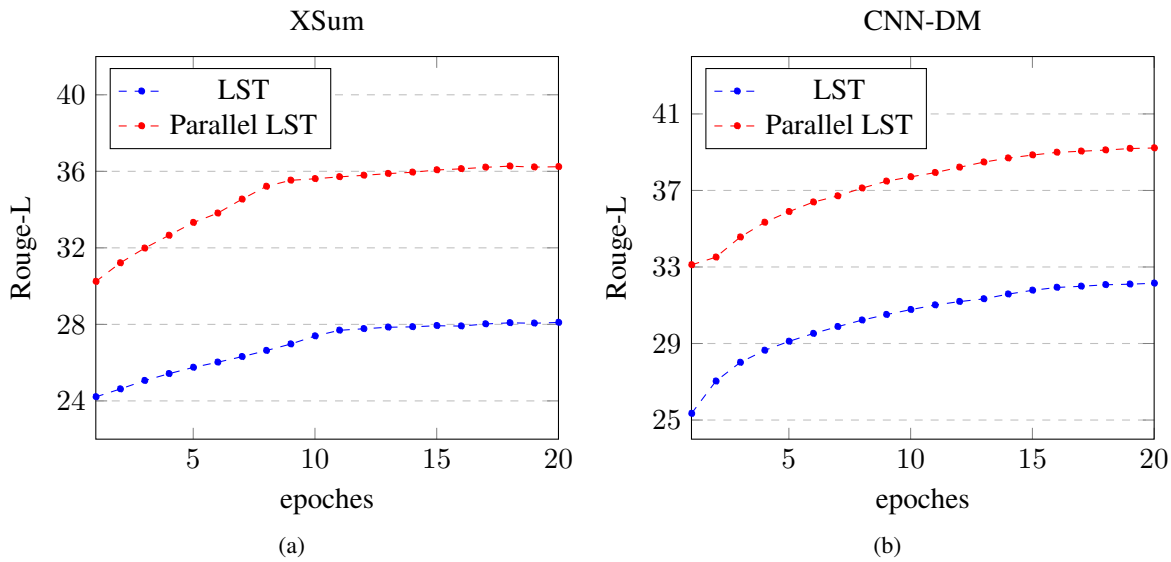

Figure 4: Model Rouge-L scores comparison on parallel/sequential side model.

formance reduction is relatively minimal. However, the improvement in inference speed is substantial. Considering our requirement for quick responses at the edge, such a classifier aligns better with our vision and needs. Simultaneously, for more fundamental information, predictions are made using the original model. The classifier significantly reduces the latency caused by continuous calls to the edge-side model during predictions, ensuring that the model learns both user and task features effectively.

We evaluate the performance of the sequence to sequence task between the original model and the model after parallel processing, and make corresponding comparisons under different datasets, and determine its performance under the corresponding task by the Rouge-L parameter.

In the parallelized design, the performance for our generative class of tasks is consistently higher than the original serialized design. As shown on Figure 4,such a parallelized design also highlights the plug-and-play nature of our model.After a very small number of rounds, the performance of parallelization is already more prominent than 20 rounds of generative training. Thus the use of a parallelized design in this paper is also in line with the corresponding sequence-to-sequence generation task.

5.5 Case Study

As shown on Figure 4.By detecting and processing the impact of classifiers on test cases, including the labels and branches used, we assess their influence and examine whether predictions are based on the original corpus or newly embedded knowledge. We analyze the top five results, observing key activation moments of the end-to-end model, as well as the predictions of the original pre-trained model at these moments.

The original model is primarily based on the usage of existing corpora and generated pronouns. These pronouns are generated within the original large-scale model, enabling it to learn the desired outcomes that users want to predict. For specific feature information, the output is primarily associated with embedded sentiment words and pronouns. These features are learned from the SPA model and are applicable to user profile features or downstream task requirements.

6 Conclusions

This paper introduces a model suitable for cloud-base and on-devices collaboration. This model leverages the capabilities of large models for sequence-to-sequence tasks and utilizes user-specific features from the edge model to guide the overall large model towards our desired objectives. Through a classifier, we can select the most suitable generated sentences. Our approach presents a new paradigm in addressing issues related to edge model deployment, providing valuable insights for future work. We believe that this approach is groundbreaking and can serve as a valuable reference for future research.

7 Limitation

Our model is an cloud-based and on-devices cooperative model, so it embodies a fairly strong inference ability on some downstream tasks, while relying on the high-parameter nature of the large model, the end-to-end prediction can also have a very good result. Our model still has some shortcomings. For example, in such a collaborative process, its delay is still unavoidable. In some scenarios that require high network resources, our model may still not be worked for reasoning and training in such scenarios.

In addition, there may be some privacy issues in our scenarios, because our corpus is ultimately delivered to the cloud, so the cloud services can still see the corresponding corpus content. However, we still protect the privacy of the model to a certain extent, because the parameters related to the model are kept on our local side, and we only need these parameters to realize the inference of the model for the user features. These problems also deserve to be further investigated and addressed in future work.

References

Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

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.

Samuel Cahyawijaya. 2021. Greenformers: Improving computation and memory efficiency in transformer models via low-rank approximation. *arXiv preprint arXiv:2108.10808*.

Han Cai, Ji Lin, Yujun Lin, Zhijian Liu, Haotian Tang, Hanrui Wang, Ligeng Zhu, and Song Han. 2022. Enable deep learning on mobile devices: Methods, systems, and applications. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27(3):1–50.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 93–98.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Tao Ge, Si-Qing Chen, and Furu Wei. 2022. EdgeFormer: A parameter-efficient transformer for on-device seq2seq generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10786–10798, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp.

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

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

Kyungjae Lee, Wookje Han, Seung-won Hwang, Hwaran Lee, Joonsuk Park, and Sang-Woo Lee. 2022. Plug-and-play adaptation for continuously-updated qa. *arXiv preprint arXiv:2204.12785*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*.

Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. 2018. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *Proceedings of the 16th annual international conference on mobile systems, applications, and services*, pages 389–400.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2020. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

R OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

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.

Jun Rao, Xv Meng, Liang Ding, Shuhan Qi, Xuebo Liu, Min Zhang, and Dacheng Tao. 2023. Parameter-efficient and student-friendly knowledge distillation. *IEEE Transactions on Multimedia*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021a. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021b. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. *arXiv preprint arXiv:2101.00234*.

Jinke Ren, Guanding Yu, Yinghui He, and Geoffrey Ye Li. 2019. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology*, 68(5):5031–5044.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Lin Wang and Kuk-Jin Yoon. 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. Llmcad: Fast and scalable on-device large language model inference.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.