ChamaleonLLM: Batch-Aware Dynamic Low-Rank Adaptation via Inference-Time Clusters

Kamer Ali Yuksel & Hassan Sawaf
aiXplain Inc., San Jose, CA, USA
{kamer, hassan}@aixplain.com

Abstract

Recent advances in large language models (LLMs) have shown remarkable performance across diverse tasks. However, these models are typically deployed with fixed weights, which limits their ability to adapt dynamically to the variability inherent in real-world data during inference. This paper introduces ChamaleonLLM, a novel framework that enables inference-time adaptation of LLMs by leveraging batch-aware clustering and on-the-fly generation of low-rank updates. Unlike traditional fine-tuning approaches such as Low-Rank Adaptation (LoRA) or methods that rely on a fixed set of pre-learned uniforms (changeable masks), our method dynamically generates adaptive modifications to the decoder weights based on the aggregated statistics of clustered batches. By intelligently grouping similar inputs and computing context-aware low-rank updates via a hyper-network, ChamaleonLLM achieves significant performance gains, outperforming conventional LoRA methods while eliminating the overhead of maintaining multiple expert models. Our experiments highlight the potential of our approach to serve as a versatile and highly adaptive solution for language model inference. ChamaleonLLM is open-sourced to ensure the reproducibility of our experiments.

1 Introduction

Large language models have revolutionized natural language processing by demonstrating unprecedented text generation, summarization, translation, and beyond capabilities. Despite their impressive performance, a persistent limitation is their static nature during inference: once deployed, the weights of these models remain fixed regardless of the variability in input data. This static configuration can lead to suboptimal performance when encountering novel or contextually distinct inputs. Traditionally, fine-tuning methods such as Low-Rank Adaptation (LoRA) have been employed during training to inject task-specific updates into the model parameters in a computationally efficient manner (Hu et al., 2021). LoRA achieves this by freezing most of the pre-trained model's parameters and introducing trainable low-rank matrices that serve as corrections. However, even with these efficient updates, the low-rank modifications remain static during inference, meaning that the model cannot adjust to nuances in the input data it receives on-the-fly.

ChamaleonLLM is designed to address this limitation by enabling the model to adapt its decoder weights during inference based on the structure and statistics of the input batch. Inputs are grouped into clusters based on their semantic and syntactic similarities. By leveraging precomputed token embeddings, the inference engine identifies coherent groups within each batch, ensuring that similar examples are processed together. Rather than relying on pre-learned and fixed uniforms (changeable masks) or static LoRA updates, ChamaleonLLM employs a hyper-network to generate low-rank adaptation parameters in real-time. This dynamic generation is based on the aggregated statistics of the clustered batch, allowing the model to tailor its adaptations to the prevailing context. This approach not only removes the necessity for multiple expert models or a vast wardrobe of pre-stored masks but also leverages the collective context of the batch, leading to enhanced performance across a range of tasks. The primary contributions of this work are three-fold: (1) We introduce a method to dynamically generate low-rank updates based on batch statistics, resulting in a self-adaptive model during inference. (2) We employ a clustering-based on normalized token embeddings that groups similar inputs, ensuring context-aware adaptation. (3) We provide experimental results demonstrating the superiority of ChamaleonLLM over traditional LoRA fine-tuning.

2 Related Work

LoRA has emerged as a popular method for fine-tuning large pre-trained models with minimal computational resources. The fundamental idea behind LoRA is to freeze the pre-trained weights and introduce a pair of trainable, low-rank matrices that approximate the necessary updates for a given task. By operating in a low-dimensional subspace, LoRA significantly reduces the number of trainable parameters while enabling effective adaptation. Despite its success, standard LoRA is inherently static during inference. Once the low-rank updates are learned, they are fixed and applied uniformly, regardless of the heterogeneity of the incoming data. This static nature can limit the model's responsiveness to dynamic input distribution or context changes. An alternative to direct low-rank adaptation is using changeable masks as pre-learned uniforms (Sun et al., 2025). This method involve training a set of masks—each tailored to specific tasks or data distributions—that can be blended or selected during inference. The blending is implemented through a softmax-based selector mechanism that chooses the most appropriate mask based on a task embedding or a similar indicator. While this method introduces a degree of flexibility, it comes with drawbacks:

- **Storage Overhead:** A large set of masks must be maintained, and these masks need to be loaded and blended during inference, increasing both memory and computational demands.
- **Static Adaptation:** Even though blending introduces some variability, the masks are fine-tuned individually and not co-optimized for joint adaptation. This can lead to suboptimal performance when the masks are applied in a blended fashion.
- **Task Limitation:** Many of these approaches rely on task embeddings, which restrict adaptation to known task categories rather than capturing the broader context of the input batch.

Recent research has also explored using task description embeddings derived from powerful models, such as GPT-4, to generate low-rank updates on a per-sample basis (Charakorn et al., 2024). This method generates the low-rank parameters for each input based on a task representation. While effective in certain scenarios, this per-sample adaptation may not fully exploit the shared context available when processing batches of similar inputs. ChamaleonLLM differentiates itself by using batch-level context for adaptation. By clustering inputs and computing aggregated statistics, our method generates low-rank updates that reflect the collective characteristics of the batch. Aggregating across a batch helps mitigate the noise or outlier effects in single-sample adaptation. A uniform adaptation across a cluster of similar inputs can lead to more coherent and consistent outputs. By not being limited to task-specific embeddings, our method can adapt to a broader range of scenarios, including open-domain and instruction-based tasks. Methods that use task embeddings typically generate per-sample low-rank updates based on a fixed task identifier. In contrast:

- **Batch vs. Sample-Level Adaptation:** Our method leverages batch-level statistics, leading to a more coherent adaptation that benefits from the collective context of multiple samples.
- **Flexibility in Unstructured Environments:** Task embeddings require clear task boundaries, whereas our context-based approach naturally adapts to real-world data.
- **Out-of-Sample Robustness:** Aggregating over a batch reduces the over-fitting sensitivity of training, an important advantage when dealing with heterogeneous or noisy datasets.

Similarly, another recent approach Tan et al. (2024) introduces a hypernetwork-based method for dynamic layer operations, which generates context-dependent low-rank updates by adjusting the model's depth on a per-sample basis; where the hypernetwork adapts the network structure for each individual input—effectively generating adjustments for model depth dynamically at inference time. In contrast to per-sample or depth-focused adaptation methods, our current work—ChamaleonLLM—exploits batch-level context to generate adaptive low-rank updates. Rather than adapting each sample independently, we cluster similar inputs and compute aggregated token embedding statistics across the batch; these statistics then drive our hyper-network to generate low-rank updates that reflect the collective context. ChamaleonLLM achieves a coherent and context-aware adaptation that naturally overcomes the limitations of fixed, per-sample updates and the storage overhead associated with large sets of pre-learned masks, such as the impact of noisy or outlier samples, and the storage overhead associated with maintaining pre-learned masks.

3 Methodology

ChamaleonLLM is built upon a pre-trained causal language model (e.g., GPT-2), where the conventional transformer layers and language modeling head are augmented with low-rank adaptation modules. The innovation lies in the dynamic generation of low-rank parameters based on batch-aware statistics. This process involves two major components: (1) batch-aware clustering and (2) adaptive low-rank update generation. The central hypothesis is that inputs arriving in a batch often exhibit significant similarity—whether semantic, syntactic, or stylistic. By grouping similar inputs, we can extract shared context and exploit common features in the batch. This grouping not only aids in noise reduction by averaging but also ensures that the low-rank updates are tailored to the collective characteristics of the batch rather than isolated samples. Using the normalized token embeddings, we perform k-means clustering. The number of clusters is chosen based on the batch size to ensure each cluster is large enough to provide robust statistics while capturing meaningful variations between groups. The clustering algorithm minimizes the Euclidean distance between points and cluster centroids, and through iterative refinement, it assigns each input to a cluster that best represents its features. Once the clusters are defined, the batch is reconstructed such that each mini-batch contains inputs from the same cluster. This reorganization is critical because it ensures that subsequent low-rank adaptations are computed on a homogenous set of inputs.

The core idea of dynamic adaptation is to generate low-rank modifications specific to the current batch's context. This work proposes a more flexible alternative to LoRA that uses a hyper-network. Instead of applying a fixed set of low-rank updates (as in standard LoRA), ChamaleonLLM employs a hyper-network to produce these parameters on-the-fly. The hyper-network takes as input the mean of the token embeddings from the entire batch, processes this aggregated statistic through several fully connected layers with non-linear activations, and outputs the parameters for the low-rank update. This on-the-fly generation allows the model to dynamically adjust to the specifics of the batch, capturing nuances that a static update might miss. The overall architecture maintains the integrity of the pre-trained model while introducing minimal additional parameters. The transformer layers are wrapped with LoRA modules that operate similarly to conventional implementations, ensuring that most of the model's capacity remains unchanged. For the LM head, the hyper-network variant ensures that adaptation is context-dependent. During inference, the model first processes the batch to compute the necessary embeddings to cluster the inputs and then generates a custom low-rank update for the LM head based on the cluster's aggregated statistics. The final output is produced by applying this adapted LM head to the last hidden states of the transformer.

4 Experiments

For our experiments, we have used WikiText-2 (Merity et al., 2016) and Alpaca (Taori et al., 2023) datasets, providing meaningful benchmarks due to their diverse and natural language texts. The datasets are split into training and validation sets. Each text sample is tokenized using a pre-trained GPT-2 tokenizer by truncating to a maximum length and padding to ensure uniform sequence lengths. LM input token embeddings are computed for every example, and normalized to facilitate robust clustering. For the Alpaca dataset, we calculate the token embeddings from the instruction part of the input prompts. A crucial part of ChameleonLLM is creating data loaders that reflect the clustered nature of the inputs. Using the precomputed token embeddings, we apply k-means clustering with the number of clusters based on the desired batch size and overall dataset size. After clustering, indices are grouped so each mini-batch contains examples from a single cluster. This ensures that each batch is contextually coherent. We implement a custom data loader that leverages these clusters as batch samplers. The data loader returns batches fed into the model during training and evaluation. The evaluation is performed at the end of each epoch on the validation set, where batches are processed through the clustering pipeline, and the average loss is computed. This systematic evaluation ensures that our comparisons are fair and reflect real-world performance. All codes for ChameleonLLM are open-sourced to ensure the reproducibility of experimental results.

Experimental results demonstrate that ChameleonLLM significantly improves over the traditional LoRA approach, consistently achieving a lower average validation loss on both datasets, and suggest that dynamic adaptation based on batch statistics leads to better convergence and generalization. As shown in Table 1, ChameleonLLM achieves a validation loss reduction of approximately **25%** compared to traditional LoRA, along with a perplexity improvement of roughly 12%. Table 2 further indicates that it reduces the validation loss by almost **30%** on instruction fine-tuning. These improvements underscore the effectiveness of the batch-level context adaptation strategy and can largely be attributed to the exploitation of batch-level context. By averaging token embeddings over each batch, the hyper-network input becomes less sensitive to individual outliers, leading to robust low-rank parameter generation. Clustering groups together semantically or stylistically similar inputs allows the hyper-network to extract stronger, more relevant signals for adaptation.

Table 1: Comparison of Low-Rank Adaptation Regimes on WikiText-2 Dataset

Adaptation Regime	Parameters	Training Loss	Validation Loss	Val. Perplexity
Unadapted GPT-2	124,439,808	13.8144	13.7876	972,500
Traditional LoRA	204,100	0.5504	0.5023	1.6525
ChameleonLLM	6,786,596	**0.2359**	**0.3753**	**1.4554**

Table 2: Comparison of Low-Rank Adaptation Regimes on Alpaca Dataset

Adaptation Regime	Parameters	Training Loss	Validation Loss	Val. Perplexity
Unadapted GPT-2	124,439,808	13.3881	13.3881	652,166
Traditional LoRA	204,100	0.1254	0.1527	1.1650
ChameleonLLM[*]	6,786,596	**0.0810**	**0.1082**	**1.1143**

[*] Token embeddings for adaptation are calculated from the instruction part of the prompt only.

- **Traditional LoRA Fine-Tuning:** Here, all transformer layers and the LM head use the static LoRA adaptation. Standard cross-entropy loss (with appropriate masking for padding tokens) is computed, and the low-rank parameters are optimized using AdamW optimizer.
- **ChameleonLLM Fine-Tuning:** In this regime, while the transformer layers continue to use static LoRA modules, the LM head is adapted by a hyper-network LoRA module that generates low-rank update parameters based on the mean token embeddings of the batch. The loss is computed on the LM head outputs, and optimized similarly.

Although ChameleonLLM has more trainable parameters than LoRA, this increase is justified by many practical benefits, as it dynamically generates low-rank updates on-the-fly rather than training and storing separate adaptation matrices for each sample or task. Avoiding the storage of numerous pre-learned matrices significantly reduces memory and processing overheads during inference. The additional parameters enable context-aware, batch-level adaptation, leading to better performance. Despite it also introduces some extra computational steps during inference, the overhead is modest compared to the overall inference time. The trade-off between increased computation and improved performance is highly favorable. Our approach is not tied to pre-defined task categories and adapts to the nuances in the data at inference time, offering a versatile solution for open-domain tasks.

5 CONCLUSION

In this paper, we presented ChameleonLLM—a novel framework that enables inference-time adaptation of large language models through batch-aware clustering and dynamic low-rank parameter generation. By clustering similar inputs and using a hyper-network to generate low-rank updates based on the aggregated batch statistics, our method provides a flexible and efficient alternative to static fine-tuning approaches such as traditional LoRA or uniform mask blending. Our extensive experiments on WikiText-2 and Alpaca demonstrate that ChameleonLLM achieves lower validation loss and perplexity than baseline methods with dynamic batch-aware contextual adaptation. The proposed framework reduces the need for storing multiple expert masks and adapts to diverse input distributions in real-time, making it highly suitable for modern, dynamic inference environments.

REFERENCES

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Instant transformer adaption via hyperlora. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Qi Sun, Edoardo Cetin, and Yujin Tang. Transformer^2: Self-adaptive llms. *arXiv preprint arXiv:2501.06252*, 2025.

Zhen Tan, Daize Dong, Xinyu Zhao, Jie Peng, Yu Cheng, and Tianlong Chen. Dlo: Dynamic layer operation for efficient vertical scaling of llms, 2024. ArXiv preprint arXiv:2407.11030.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.