LONG EXPOSURE: Accelerating Parameter-Efficient Fine-Tuning for LLMs under Shadowy Sparsity

Tuowei Wang*†‡§, Kun Li*§¶, Zixu Hao*†, Donglin Bai*,
Ju Ren†¶, Yaoxue Zhang†, Ting Cao*, Mao Yang*
* Microsoft Research
Beijing, China
† Tsinghua University
Beijing, China

***Abstract*—The adaptation of pre-trained large language models (LLMs) to diverse downstream tasks via fine-tuning is critical for numerous applications. However, the inefficiency of parameter-efficient fine-tuning (PEFT) techniques presents significant challenges in terms of time investments and operational costs. In this paper, we first introduce a nuanced form of sparsity, termed *Shadowy Sparsity*, which is distinctive in fine-tuning and has not been adequately addressed for acceleration. Under Shadowy Sparsity, we propose LONG EXPOSURE[1], an efficient system to accelerate PEFT for LLMs. LONG EXPOSURE comprises three key components: *Shadowy-sparsity Exposer* employs a prolonged sensing range to capture more sparsity details under shadowy sparsity; *Sequence-oriented Predictor* provides efficient yet accurate predictions to handle large sequence inputs and constantly-evolving parameters; and *Dynamic-aware Operator* facilitates more structured computational patterns and coalesced memory accesses, addressing dynamic sparse operations. Extensive evaluations show that LONG EXPOSURE outperforms state-of-the-arts with up to a $2.49\times$ speedup in end-to-end fine-tuning, offering promising advancements in accelerating PEFT for LLMs.**

***Index Terms*—Large Language Model, Fine-tuning, Sparsity**

I. INTRODUCTION

In natural language processing, the adaptation of pre-trained large language models (LLMs) [1]–[5] to diverse downstream tasks constitutes a fundamental aspect of many applications. This adaptation process, commonly known as *fine-tuning*, involves the comprehensive update of all parameters within the pre-trained model akin to training from scratch.

For the potential hundreds of thousands of downstream applications that rely on LLMs, the efficiency of fine-tuning directly affects their operational costs and time investments. Given that pre-trained LLMs need periodic updates, typically every few months, to integrate the latest knowledge, there is a pressing demand for accelerating the LLM fine-tuning process.

The major reason hindering the fine-tuning efficiency is the retention of the same number of parameters in the new model as in the original one. Efforts have been made to address this concern by introducing *parameter-efficient fine-tuning* (PEFT) techniques [10], which only selects or injects a minimal number of parameters for adaption to new tasks.

TABLE I: OPT-1.3B fine-tuning time breakdown. (ms/batch)

Phase	Forward	Backward	Optim. Step	Total
Full Param.	112.8(27.7%)	223.7(54.9%)	70.6(17.3%)	407.2
LoRA [6]	135.3(40.4%)	196.3(58.7%)	2.0(0.6%)	334.6
Adapter [7]	123.6(42.2%)	168.4(57.5%)	0.7(0.3%)	292.9
Bitfit [8]	117.6(40.5%)	172.4(59.4%)	0.2(0.07%)	290.3
P-Tuning [9]	137.5(40.1%)	193.9(56.6%)	11.1(3.2%)	342.6

One prominent approach in the domain of PEFT is low-rank adaption (LoRA) [6]. It freezes pre-trained model weights and injects smaller, trainable low-rank matrices into each transformer block. Compared to full fine-tuning, LoRA decreases the number of trainable parameters to less than 0.01%.

This substantial reduction in the number of trainable parameters mitigates the need for maintaining and updating the optimizer states for most parameters. However, PEFT techniques fall short of achieving an expected decrease in wall-clock time. As detailed in Table I, even with minimal parameters being trainable, techniques like LoRA only experience an 18% reduction in wall-clock time. While PEFT techniques notably cut down the optimization step's wall-clock time, they leave the duration of the forward and backward passes either unchanged or slightly increased. This is because, despite most pre-trained parameters being frozen, computing gradients for trainable parameters still requires complete forward and backward passes through the backbone model. Consequently, the forward and backward passes have emerged as the computational bottlenecks impeding further acceleration.

In this paper, we propose LONG EXPOSURE[2], an efficient system to accelerate parameter-efficient fine-tuning for LLMs. The design of LONG EXPOSURE is grounded in a crucial observation that PEFT and inference in LLMs exhibit high similarities in their computation patterns. In PEFT techniques, a majority of model parameters remain frozen, similar to the scenario in model inference where parameters also stay unaltered. Previous studies [11]–[23] have evidenced that LLMs typically exhibit considerable sparsity, with a great number of activations can be excluded from computation to expedite inference in wall-clock time while preserving quality. Guided

‡ Work done during an internship at Microsoft Research.
§ Tuowei Wang and Kun Li have equal contributions to this article.
¶ Corresponding authors (kunli@microsoft.com; renju@tsinghua.edu.cn).

[1]Long Exposure is available at https://github.com/HPHEX/LongExposure.

[2]Similar to employing a slow shutter speed in photography to capture more light for producing clearer images, this paper adopts a series of granular techniques to expose and leverage more sparsity for accelerating fine-tuning.

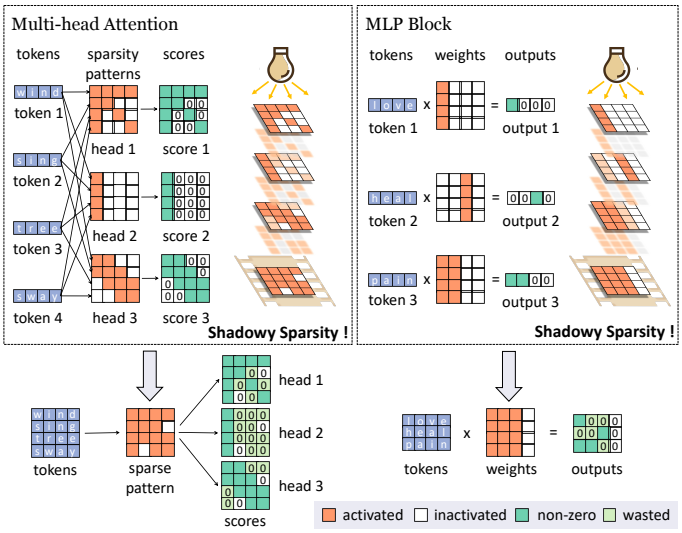


Fig. 1: Shadowy sparsity in LLM fine-tuning. Transformer-based models are generally composed of two primary components: multi-head attention and MLP block. In fine-tuning, the sparse patterns of various tokens within the input sequence exhibit a logical AND relationship, which restricts the sparsity degree and leads to computational waste in both components.

by this observation, the key insight of LONG EXPOSURE is inspired: given the striking similarities in computation patterns between PEFT and inference, *why not build a bridge to PEFT acceleration by capturing intrinsic sparsity like inference?*

However, this is not a low-hanging fruit, as the sparsity inherent in fine-tuning introduces distinct characteristics that diverge significantly from those encountered during inference. In inference, the model typically processes one token at a time, where the sparse pattern is easily discernible for each token. In contrast, fine-tuning involves feeding the model with a sequence of tokens, where the sparsity patterns heavily overlap across different tokens, as depicted in Figure 1. We coin this intricate sparsity observed in fine-tuning as *Shadowy Sparsity*. To accelerate PEFT for LLMs under this shadowy sparsity, several key technical challenges must be tackled carefully.

Firstly, *how to capture more sparse patterns under shadowy sparsity*. As illustrated in Figure 1, the sparse pattern of the input token sequence emerges from the logical AND combination of the sparse patterns of individual tokens in the sequence. This means that the dense units for a specific token might coincide with the sparse units for another token. The resulting shadowy sparsity exhibits a limited level of overall sparsity, despite the high sparsity degree of each token, leading to potential computational waste.

Secondly, upon capturing sparsity, the subsequent critical challenge lies in *how to predict efficient yet accurate sparse patterns* to minimize associated computational expenses before incurring actual costs. Due to the exact sparse patterns typically varying with different inputs, a commonly employed method in inference is to utilize neural networks for predicting sparse patterns at runtime [19]–[22]. However, directly flattening the token sequence in fine-tuning as network inputs could lead to an excessively large network, which is both memory-intensive and time-consuming. Moreover, the continuously-evolving trainable parameters during fine-tuning also add complexity to ensuring the correctness of the prediction.

Thirdly, *how to achieve effective performance improvements based on well-predicted sparsity*. The irregular computation patterns and scattered memory accesses associated with sparsity make it challenging to attain comparable performance improvements to the theoretical computation reductions. Moreover, these well-predicted sparsity patterns exhibit highly dynamic characteristics that vary with different inputs at runtime. This renders many existing tools ineffective in capturing and handling such dynamic variations.

LONG EXPOSURE employs a suite of techniques to address these challenges. The concept of 'LONG EXPOSURE' emphasizes that rather than simply harnessing the limited sparsity remaining in shadowy sparsity, we take a longer view which captures more intricate details of individual sparse pattern before they fade into shadow. The core of LONG EXPOSURE is the *Shadowy-sparsity Exposer*, a technique designed for exposing the latent sparsity hidden in shadowy sparsity. In multi-head attention, we introduce specific sparse patterns tailored to each attention head, avoiding the computational redundancy or oversight that can arise from employing a uniform mask. In MLP block, we take the importance of each activated neuron into consideration. By identifying and filtering out neurons whose activation can be safely disregarded, we transform shadowy sparsity into structured block-wise sparsity.

LONG EXPOSURE utilizes *Sequence-oriented Predictors* to address the conflicts between long sequence inputs and the associated neural network size. This technique is grounded in a two-stage design strategy: Initially, the predictor processes each token individually; then these predictions are subsequently consolidated. Moreover, to minimize the disruption caused by updating trainable parameters, we introduce specific training optimizations to bolster the predictor's robustness.

LONG EXPOSURE develops a collection of *Dynamic-aware Operators* to facilitate practical acceleration on hardware systems, covering all the sparse operations involved in multi-head attention and MLP block. Different from most existing tools, these operators avoid additional data conversion overhead, making them well-suited for dynamic scenarios. In addition, we design a two-stage algorithm for multi-head attention that adeptly balances precomputation with dynamic sparse patterns.

We evaluate LONG EXPOSURE across various PEFT methods and on two different GPU platforms. The results show that our system achieves up to $2.49\times$ speedup and $2.77\times$ memory savings in end-to-end fine-tuning compared with the state-of-the-art fine-tuning system, maintaining model accuracy.

In summary, our contributions are as follows:

- We are the first to identify and leverage the intrinsic sparsity within LLM fine-tuning, namely shadowy sparsity, to accelerate the PEFT process for LLMs.

- We introduce three key components that capture, predict, and exploit sparsity patterns, respectively. This approach provides a coherent strategy for optimizing both the multi-head attention and the MLP block within LLMs.
- We implement these techniques as an end-to-end fine-tuning system that is compatible with a variety of PEFT techniques. Our system achieves up to $2.49\times$ speedups and $2.77\times$ memory savings compared to the state-of-arts.

II. Background and Motivation

A. Parameter-efficient Fine-tuning (PEFT)

Adapting a large pre-trained language model for various downstream applications typically involves full fine-tuning, where all parameters of the pre-trained model are updated. However, as models grow in size, full fine-tuning has evolved from being inconvenient to almost impractical. To reduce the costs associated with full fine-tuning, various PEFT methods have emerged in recent years. The central concept behind these methods is to avoid updates of the full set of parameters without performance degradation.

A promising direction within PEFT involves freezing the pre-trained model's parameters while introducing a small number of new trainable parameters. One such method [7], [24], [25] involves the use of adapters, which are additional layers inserted between the model's existing layers. Specially, Low-Rank Adaptation (LoRA) [6] injects trainable low-rank matrices into each layer, reflecting the insight that the updates to model weights actually operate within a low intrinsic dimension. Other methods [9], [26] use prompts, adding trainable parameters to the model's input to leverage the pre-trained model's existing knowledge for new tasks. Instead of introducing new parameters, some methods [8], [27] selectively update a small portion of the pre-trained model's parameters, such as only the bias terms [8]. However, owing to the inherent computational flow of backpropagation, the significant reduction of trainable parameters provided by PEFT primarily benefits the optimizer step, leaving the forward and backward phases as the new bottlenecks.

B. Sparsity in LLMs

A significant number of activations within both two primary components of transformer-based LLMs: the multi-head attention and the MLP block, are found zero or nearly zero [23], [28]. These negligible activations can be disregarded with no or little impact, leading to sparsity in model's computational demands. In multi-head attention, sparsity typically emerges from the limited interactions among different tokens. Models can selectively mask out irrelevant tokens, leading to less computation without accuracy degradation. In MLP block, sparsity is primarily attributed to the properties of the ReLU activation function, which is increasingly utilized by many LLMs [4], [29], [30]. This function sets all negative activation values to zero, allowing them to be excluded from computations.

This inherent sparsity inspires us to leverage it for expediting the parameter-efficient fine-tuning of LLMs. Unlike

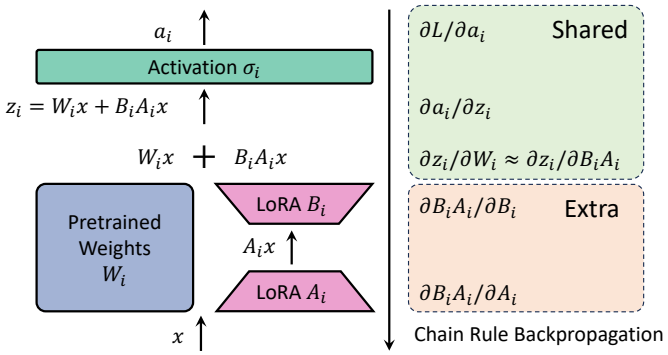


Fig. 2: An illustrated example of LoRA for showcasing the computational flow (forward and backward) in PEFT.

the original model training, the entire parameters of pre-trained model are frozen during PEFT. This allows for accurate predictions of sparse patterns in both multi-head attention and MLP block, surpassing the limitations of pre-defined sparse attention masks [14]–[18], which may not be suitable for all inputs. Although some research [20]–[22] has employed similar techniques to accelerate LLM inference, the unique characteristics of sparsity during fine-tuning present substantially different challenges. To our knowledge, no research has utilized this sparsity to accelerate the fine-tuning process.

C. Analysis: PEFT Computational Cost Breakdown

Fine-tuning a pre-trained model with trainable parameters consists of three phases: (1) the forward phase calculates the loss for current data batch; (2) the backward phase calculates the gradients of the trainable parameters; (3) the optimizer step updates the trainable parameters using these gradients.

Consider the use of LoRA to fine-tune an MLP block with alternating linear and activation layers, as shown in Figure 2. The ith linear layer consists of weight W_i, LoRA matrices A_i and B_i, and the ith activation layer is σ_i. For ith layer, we denote the output of ith linear layer as z_i, and the output following activation as a_i.

During the forward phase, the computational costs of PEFT are either unchanged or slightly increased. Taking our LoRA example, the output of ith linear layer is computed as follows:

$$z_i = W_i x + B_i A_i x$$

$$a_i = \sigma_i(z_i)$$

Compared to the original process, the injected LoRA matrices A_i and B_i slightly increase the computational costs.

During the backward phase, the situation is more complex. In backpropagation with loss L of full fine-tuning, the LoRA matrices A_i and B_i are absent, and the gradient for the trainable weight W_i is:

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial W_i}$$

In contrast, when using LoRA, the gradients relative to the trainable parameters A_i and B_i are:

$$\frac{\partial L}{\partial A_i} = \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial B_i A_i}\frac{\partial B_i A_i}{\partial A_i}$$
$$\frac{\partial L}{\partial B_i} = \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial B_i A_i}\frac{\partial B_i A_i}{\partial B_i}$$

Though using LoRA can skip the calculation of $\frac{\partial z_i}{\partial W_i}$, it introduces the calculations of $\frac{\partial z_i}{\partial B_i A_i}\frac{\partial B_i A_i}{\partial A_i}$ and $\frac{\partial z_i}{\partial B_i A_i}\frac{\partial B_i A_i}{\partial B_i}$ instead. Given that the computational costs of $\frac{\partial z_i}{\partial W_i}$ are comparable to $\frac{\partial z_i}{\partial B_i A_i}$, LoRA leads to additional computational steps involving $\frac{\partial B_i A_i}{\partial A_i}$ and $\frac{\partial B_i A_i}{\partial B_i}$. Apart from these differences, the remaining computational costs are essentially equivalent as a result of the chain rule, which both require traversing the layers of the extensive pre-trained model.

In the optimizer step, the computational costs of PEFT are reduced due to fewer trainable parameters. However, the extent of this saving can vary depending on the choice of optimizer and typically accounts for a small fraction of the overall. In summary, PEFT methods do not reduce the computational cost significantly compared to full fine-tuning.

D. Opportunity: Accelerate PEFT with LLM Sparsity

The application of PEFT techniques has greatly cut down the cost of the optimizer update phase, turning the forward and backward phases into new bottlenecks. Unlike standard fine-tuning or training, where model parameters are continuously updated through iterations, PEFT techniques maintain most of the parameters frozen. This closely mirrors the computational pattern observed during inference. Given that sparsity is commonly employed to diminish computational expenses during inference, there is a compelling opportunity to apply sparsity to streamline the fine-tuning as well. However, fine-tuning inherently involves both the forward and backward phases, while inference is limited to the forward phase. To successfully adopt sparsity in fine-tuning, it is necessary to analyze how sparsity impacts the backward phase.

Considering the same example shown in Figure 2, the output of the ith linear layer can be expressed as follows, with W_i denoting the ith row of W:

$$z = Wx + BAx = (W + \Delta W)x$$
$$= \big((W_1 + \Delta W_1)x, \ldots, (W_i + \Delta W_i)x, \ldots, (W_d + \Delta W_d)x\big)^T$$

To incorporate sparsity, we suppose $z_i = (W_i + \Delta W_i)x < 0$ and set the activation function as ReLU. Consequently, W_i is inactivated and the activation can be expressed as:

$$a = \text{ReLU}(z) = \big((W_1 + \Delta W_1)x, \ldots, 0, \ldots, (W_d + \Delta W_d)x\big)^T$$

In backpropagation, the gradient relative to z can be expressed as:

$$\frac{\partial L}{\partial z} = \text{ReLU}^{'} \odot \frac{\partial L}{\partial a}$$
$$= \big(1, \ldots, 0, \ldots, 1\big)^T \odot \big(\tfrac{\partial L}{\partial a_1}, \ldots, \tfrac{\partial L}{\partial a_i}, \ldots, \tfrac{\partial L}{\partial a_d}\big)^T$$
$$= \big(\tfrac{\partial L}{\partial a_1}, \ldots, 0, \ldots, \tfrac{\partial L}{\partial a_d}\big)^T$$

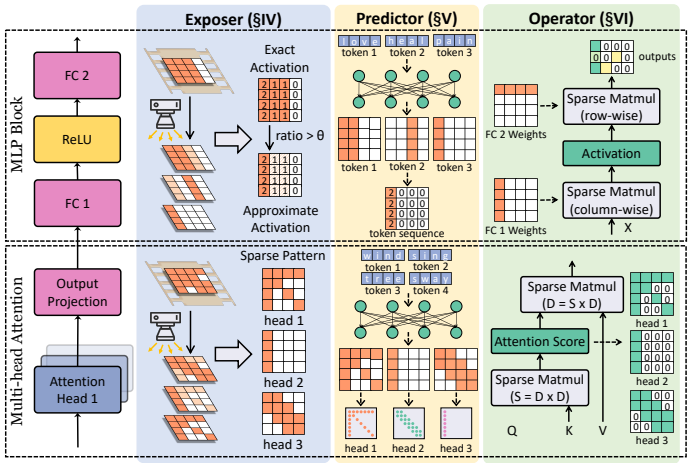


Fig. 3: LONG EXPOSURE overview

To calculate the gradients relative to the trainable parameters A_i and B_i for updating LoRA matrices, we first calculate the gradient relative to their product:

$$\frac{\partial L}{\partial B_i A_i} = \frac{\partial L}{\partial z}x^T$$
$$= \big(\tfrac{\partial L}{\partial z_1}, \ldots, 0, \ldots, \tfrac{\partial L}{\partial z_d}\big)^T \big(x_1, \ldots, x_i, \ldots, x_d\big)$$
$$= \begin{pmatrix} \frac{\partial L}{\partial z_1}x_1 & \ldots & \frac{\partial L}{\partial z_1}x_i & \ldots & \frac{\partial L}{\partial z_1}x_d \\ \vdots & & \vdots & & \vdots \\ 0 & \ldots & 0 & \ldots & 0 \\ \vdots & & \vdots & & \vdots \\ \frac{\partial L}{\partial z_d}x_1 & \ldots & \frac{\partial L}{\partial z_d}x_i & \ldots & \frac{\partial L}{\partial z_d}x_d \end{pmatrix}$$

This gradient does not involve $\frac{\partial L}{\partial z_i}$, implying that no corresponding W_i is involved either. It leads to an important conclusion: *if certain model parameters remain inactive during the forward phase, they are effectively excluded from the gradient computation in the backward phase*. This conclusion highlights the potential for leveraging sparsity to decrease the computational demands of PEFT, analogous to the efficiencies observed during inference.

III. OVERVIEW

We propose LONG EXPOSURE, an efficient fine-tuning system with LLM instinctive sparsity. Beyond prior works that concentrate exclusively on either parameter-efficiency during fine-tuning or computation-efficiency during inference, LONG EXPOSURE is efficient on aspects of both parameter and computation. Figure 3 presents an overview of our system.

Shadowy-sparsity Exposer (Section IV). Drawing from our experimental findings, we present several key observations of LLM sparsity in fine-tuning. Different from inference, the sparse patterns in fine-tuning are heavily overlapping across different tokens. We refer to this as *Shadowy Sparsity*. To expose the latent sparse patterns inherent in shadowy sparsity, we adopt a more granular approach. In multi-head attention, we consider the unique features of each head and employ a

head-specific sparse mask. In MLP block, we factor in the importance of activated neurons and introduce a neuron-filter to transform shadowy sparsity into block-wise sparsity.

Sequence-oriented Predictor (Section V). LONG EXPOSURE employs neural-network-based predictors to determine the desired sparse patterns at runtime. In multi-head attention, the goal is to predict the sparse mask for each head. In MLP block, the prediction targets the neurons that are activated in the model weights. To control the size of predictors, we initially process each token in the sequence individually and then combine the outputs. Data argument techniques and tailored loss metrics are introduced to increase predictor accuracy in the presence of trainable parameters during fine-tuning.

Dynamic-aware Operator (Section VI). LONG EXPOSURE integrates a suite of dynamic-aware operators that make efficient use of predicted sparse patterns. In multi-head attention, each head is associated with a distinct sparse pattern, which necessitates two sparse matrix multiplications. We design a two-stage algorithm that shifts the bulk of the data format conversion overhead in matrix multiplication to pre-runtime, while still meeting the dynamic nature of sparse patterns. In MLP block, model parameters are activated either by row or by column, i.e. a neuron. We optimize the standard block-wise matrix multiplication by taking into account the neuron-wise sparse pattern. Additionally, we optimize the data layout to enhance memory coalescing.

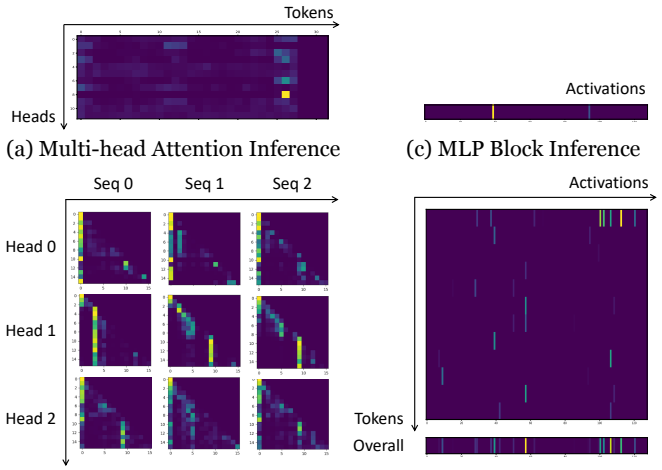
IV. SHADOWY-SPARSITY EXPOSER

A. Observation: Shadowy Sparsity

Many studies [11]–[13], [19], [28] have highlighted the inherent sparsity found in LLMs. Moreover, the sparsity within LLM is dynamic, indicating that the sparse patterns change with different inputs. Most of the existing works [20]–[22] concentrate on model inference, where the model input during decoding is a single token (sequence length is 1). However, we observe a distinct characteristic of sparsity during LLM fine-tuning, where the input is a sequence of tokens.

Multi-head Attention. Figure 4(a) shows the attention scores across different heads for an example token. The sparsity in multi-head attention during inference is characterized as certain heads giving heavy attention scores while others are rather uniform. This pattern offers an opportunity to selectively focus on those 'heavy hitter' heads while disregarding the rest. However, during fine-tuning, the attention scores become a matrix that describes the relationships among all tokens in the sequence, as shown in Figure 4(b). It becomes challenging to prune any particular head since each head might be activated by a certain token in the sequence. Several studies [14]–[18] suggest the use of sparse masks, designed to retain all critical attention scores for saving attention computation. However, existing sparse masks are typically pre-defined and uniformly applied to all heads, which fails to efficiently capture the various sparsity patterns in both the head and input dimensions.

MLP Block. A similar phenomenon is observed in MLP block. Figure 4(c) shows the activations after applying ReLU



(a) Multi-head Attention Inference (c) MLP Block Inference

(b) Multi-head Attention Fine-tuning (d) MLP Block Fine-tuning

Fig. 4: Visualization of attention scores in multi-head attention and activations in MLP block. Due to varying input lengths, these metrics manifest as vectors during inference and as matrices during fine-tuning. Brighter colors denote high values.

for a single token, which exhibits considerable sparsity. However, with a sequence of tokens as input, the sparsity of overall activations, which is measured by the reduction along the sequence length dimension, reduces greatly as shown in Figure 4(d). Besides, these scattered activations render the residual sparsity more unstructured, posing a challenge for leveraging it for practical acceleration.

We refer to this intricate sparsity observed in fine-tuning as *Shadowy Sparsity*, akin to each token casting a partial shadow, and when these shadows overlap, no light shines through at all. The presence of shadowy sparsity during fine-tuning makes it challenging to capture efficient sparsity patterns. On one hand, shadowy sparsity reduces the degree of sparsity, rendering it inadequate for achieving practical speedups. On the other hand, the remaining sparsity is typically highly unstructured, which is unaligned with the hardware characteristics.

B. Design: Long Exposure

To capture more sparse patterns under shadowy sparsity, our primary design lies in focusing on the intricate details of individual sparse pattern that constitutes the shadowy sparsity. We term this as *Long Exposure*, which entails employing a long-duration sensing range to capture more sparsity details.

Multi-head Attention. We employ a binary mask to capture the inter-token sparsity within a sequence. Each element of the mask corresponds to a block of attention scores, with 0 indicating non-computation and 1 indicating computation. Different from previous studies, our approach operates at the level of individual attention heads, instead of the entire multi-head attention. Concretely, for given inputs, we identify the optimal sparse masks independently for each attention head. These head-specific masks are then combined as the sparse mask of entire multi-head attention. This finer granularity

broadens the representational scope of sparse masks, thereby facilitating the capture of distinct sparse patterns hidden in shadowy sparsity. First, determining the optimal sparse mask for one head is relatively straightforward, because it is not required to account for other hands. Second, this approach can lead to further computational savings, since a score that is critical for one head might not be necessary for another.

MLP Block. Although overall activations are scattered, variations in activation frequency and values highlight the relative importance of each activation. Taking this detailed information into consideration, we selectively apply a filter to the activated neurons, effectively treating those of less importance as inactive. Given that the contributions of these neurons to the final outcome are minimal, their exclusion has a negligible impact. We implement this process in a block-wise manner, resulting in a structured sparse pattern that is well-suited to the characteristics of the hardware.

V. Sequence-oriented Predictor

Although the precise sparse patterns can emerge naturally from computation outcomes, this does not contribute to computational savings. To truly reduce computation from sparsity, it is necessary to identify sparse patterns before the actual computation. Utilizing low-rank neural networks to accurately predict these sparse patterns has been proven feasible in LLM inference [19]–[22]. Given the similarity in freezing the majority of model parameters, we adopt a similar neural-network-based approach. However, there are new challenges introduced in fine-tuning. First, the sequenced inputs could easily lead to an excessively large predictor, compromising efficiency. Second, the updates of trainable parameters introduce bias to the predictor inputs, adversely affecting accuracy.

A. Criterion I: Efficiency

We develop a two-stage design that guarantees the predictor's efficiency when handling sequence. In stage one, the predictor processes each token individually, which keeps the predictor's size constrained to the dimension of a single token. In stage two, we consolidate these individual predictions into an aggregated one that represents the final sparsity for the token sequence. Building on this sequence-oriented design, the detailed structure of predictor is specified as follows:

Multi-head Attention. Figure 5(a) shows the process of prediction in multi-head attention's one head. We construct a pair of trainable low-rank approximation matrices, $\hat{W_Q}$ and $\hat{W_K}$, to obtain the approximate queries $\hat{Q}$ and keys $\hat{K}$. Particularly, we down-sample the input X in the sequence dimension, reducing its size from s to $\sqrt{s}$ to decrease subsequent computation. This approach is rational because our primary concern lies with the overall distribution of attention scores for mask matching, rather than any individual value. The approximate attention scores are then calculated using the following formula:

$$\hat{S}_{attn} = \hat{Q}\hat{K}^T = X\hat{W}_Q X\hat{W}_K{}^T$$

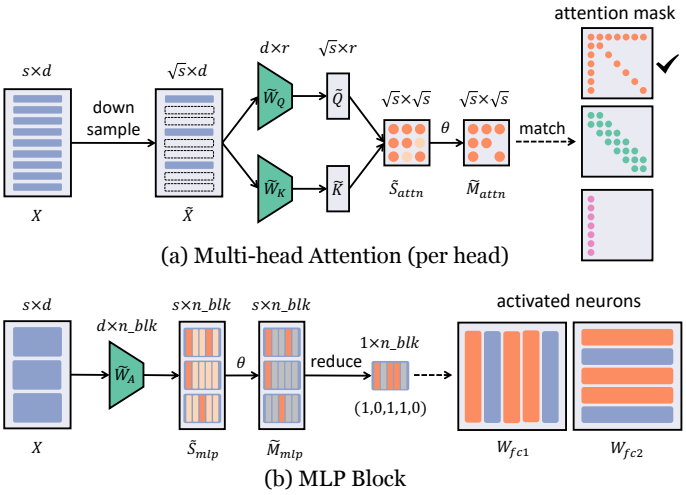


Fig. 5: The process of prediction (batch size = 1 for simplicity). Each predictor comprises a set of trainable parameters (green), which accepts tensors (blue) as input and produces approximations of attention scores or activations. These outcomes are then processed to filter out less significant values (light orange), and a reduction is performed if needed, to generate the final sparse pattern (orange) for the token sequence.

Here $\hat{S}_{attn} \in \mathbb{R}^{\sqrt{s}\times\sqrt{s}}, \hat{W_Q}, \hat{W_K} \in \mathbb{R}^{d\times r}$ and $r \ll d$. When two approximation matrices are well trained, $\hat{S}_{attn}$ can provide a close estimation of accurate attention scores S_{attn}.

The $\hat{S}_{attn}$ is converted into a binary mask $\hat{M}_{attn}$ with a threshold. Additionally, a reduction in the batch dimension is performed to generate the sparse pattern of the whole input. The resulting binary mask is then categorized into one of several pre-defined typical masks. This strategy not only aligns the sparse pattern with expert insights [14]–[18] but also provides convenience for the following efficient implementation.

MLP Block. Figure 5(b) shows our design of predictors for MLP block. Similarly, we construct a trainable low-rank matrix $\hat{W_A}$ to approximate the indices of the activated neuron blocks. Considering the correlated activation patterns of the two linear layers, the prediction result is applied to both layer weight matrices. The approximation can be expressed as:

$$\hat{S}_{mlp} = X\hat{W}_A$$

Here $\hat{S}_{mlp} \in \mathbb{R}^{s\times n\_blk}$ and $\hat{W}_A \in \mathbb{R}^{d\times n\_blk}$, where $n\_blk = \lceil d/blk\_size \rceil$ is the number of neuron blocks and blk_size is the number of neurons in each block. When $\hat{W}_A$ is well trained, $\hat{S}_{mlp}$ can indicate the importance of different neuron blocks in determining the final outputs for the given inputs.

The $\hat{S}_{mlp}$ is then binarized by applying a threshold, which serves to filter out blocks deemed less important. Finally, a reduction is performed on both the batch and sequence dimension of $\hat{M}_{mlp}$, to obtain the final sparsity pattern.

B. Criterion II: Accuracy

All predictors are pre-trained offline using data collected from model inference. Since the size of predictors is relatively

small, the training will come to convergence quickly and consume minimal resources compared to the following LLM fine-tuning. However, the updating of trainable parameters during fine-tuning could potentially skew the distribution of the original predictor inputs, thus impairing prediction accuracy.

Consequently, we employ two optimizations during predictor training. First, we add noise to the original data for data argumentation, which helps the predictor avoid overfitting and boosts its robustness. Second, we prioritize recall over precision in the computation of prediction loss. This is because the final outcome is primarily affected when weights that should be active are incorrectly predicted as inactive.

C. Analysis: Computational Savings and Overhead.

We finally analyze the computational savings and associated overhead introduced by predictors.

Savings. In the forward phase, the computational savings within multi-head attention are primarily derived from the computation of attention scores, which is reduced from $O(s^2)$ to $O(s)$ thanks to the use of sparse masks. In MLP block, the computational savings depend on the sparsity ratio, scaling by an order of magnitude corresponding to s. In the backward phase, computational savings mirror those seen in the forward phase, as detailed in the analysis presented in Section II-D.

Overhead. In multi-head attention, the extra overhead for one batch primarily consists of three matrix multiplications:

$$\text{Cost}_{attn} = \text{Cost}_Q + \text{Cost}_K + \text{Cost}_{QK} = \sqrt{s}dr + \sqrt{s}dr + sr$$

In MLP block, the overhead for one batch arises from one matrix multiplication and one reduction operation:

$$\text{Cost}_{mlp} = \text{Cost}_A + \text{Cost}_{AND} = sdr + s$$

Given that $r \ll d$ and d is a constant determined by model structure, the total computational complexity can be approximated as $O(s)$. Weighed against the substantial savings, the overhead introduced by predictor is considered acceptable.

VI. Dynamic-aware Operator

Sparse linear algebra often struggles to match the performance of its dense counterparts due to irregular computation patterns and scattered memory accesses. Many sparsity tools [31]–[34] manage to achieve comparable performance based on fixed sparsity pattern, typically relying on either static kernel compilation or data format conversion before runtime. However, during fine-tuning, sparse patterns are highly input-dependent, which can only be determined at runtime. This dynamic nature of sparsity falls outside the capabilities of these sparsity tools. Fewer sparse tools [23], [35] are tailored for dynamic scenarios. However, due to a lack of specific optimizations for our desired sparsity patterns, they only yield sub-optimal performance. To map the achieved sparsity onto hardware systems efficiently, we develop a suite of custom dynamic-aware operators. These operators make efficient use of predicted sparse patterns involved in the computation of both multi-head attention and MLP block.

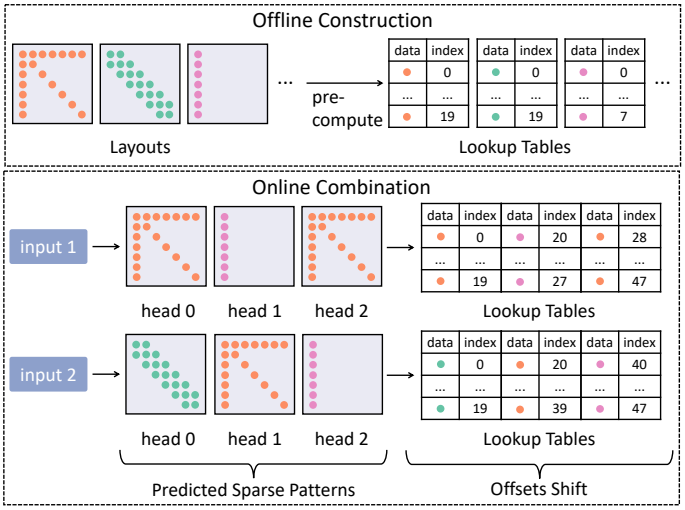


Fig. 6: Example of two-stage approach in multi-head attention.

A. Multi-Head Attention

The computation of sparse attention can be broken down into two distinct block-wise sparse matrix multiplications: SDD and DSD, where 'S' represents a sparse matrix and 'D' denotes a dense matrix. A typical optimization is to pre-calculate the sparse pattern layout, which reduces the calculations required during runtime. However, the dynamic property of sparsity at runtime inherently clashes with the premise of pre-computation. To preserve computational efficiency in the face of dynamic sparsity, we propose a two-stage approach, performed offline and online, as shown in Figure 6.

Offline Pool Construction. The irregular nature of data layouts in sparse operations means that data indexing constitutes a significant computational workload. Pre-computing the data layout indices and storing them into lookup tables is crucial for achieving optimal performance in sparse operations. Rather than pre-computing certain fixed sparse pattern layouts—which is impractical due to the dynamic nature of the sparse patterns—we construct a pool of common atomic sparse patterns and pre-calculate their layouts. This strategy stems from the observation that existing sparse attention patterns often consist of a combination of these atomic patterns [36].

Online Pattern Combination. Based on the predictor's output, each head is assigned a specific sparse pattern. These patterns will combined later to form the overall pattern of multi-head attention. During combination, only an offset needs to be added to the existing layout lookup tables. Finally, a list of data block indices is provided for sparse matrix multiplication. As the basic unit of operation is the block rather than the individual head, workload imbalance is avoided, even when the sparse patterns of different heads vary.

This two-stage approach enables a substantial portion of the computational workload to be shifted to the pre-runtime phase, while still preserving the flexibility needed for later integration to accommodate the demands of dynamic sparsity.

TABLE II: Models for evaluation.

Model	# Params	Batch Size	Seq Len
OPT	350M/1.3B/2.7B	2/4	512/1024
GPT-2	774M/1.5B	4/8	512/1024

TABLE III: Downstream tasks for evaluation.

Tasks	Description
PIQA [41]	Physical commonsense reasoning
Winogrande [42]	Physical interactions understanding
RTE [43]	Natural language understanding
COPA [44]	Commonsense causal reasoning
HellaSwag [45]	Natural language commonsense

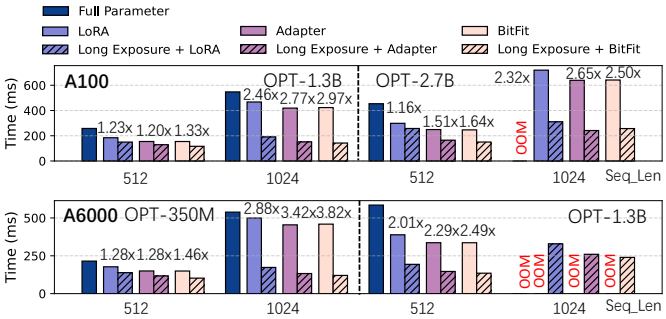


Fig. 7: Execution time per batch and speedup of OPT.

B. MLP Block

The sparsity introduced by ReLU within MLP block results in two sparse matrix multiplications occurring in both linear layers. Unlike common sparse operators, the sparse pattern here is uniquely column-wise or row-wise. It is because when an element within the MLP block's activation is zero, the corresponding column in the first linear layer as well as the row in the second linear layer are both rendered inactive. This particular characteristic allows us to craft two specific optimizations to achieve better performance.

Neuron Sparsity. Since the basic unit of activated weights is a column or a row, i.e. a neuron, we design a neuron-centric matrix multiplication based on the classic matrix multiplication tiling algorithm. Besides standard inputs, our specialized operator also accepts indices of activated neuron blocks. During computation, only the neuron blocks identified as active are loaded and computed. This approach is inherently compatible with the conventional tiling algorithm and eliminates the need for data format conversion.

Memory Coalescing. Another optimization arises from the data loading pattern of weights in the two linear layers: weights in the first layer are accessed column-wise, whereas those for the second linear layer are accessed row-wise. This inspires us to organize the weights in the two linear layers in a column-major format and a row-major format, respectively. This alignment with the memory access patterns of GPUs minimizes the overhead associated with data loading and enhances computational throughput.

VII. Evaluation

A. Experimental Setup

Machines. We conduct experiments on two platforms, covering both data-center workstation and desktop professional GPU. Platform A contains an AMD EPYC 7V13 processor and an Nvidia A100 80GB GPU. The A100 GPU provides 1,555 GB/s memory bandwidth and 19.5 TFLOPs FP32 operations. Platform B contains an AMD EPYC 7742 processor and 4 Nvidia A6000 48GB GPUs. Each A6000 GPU offers 768 GB/s memory bandwidth and 38.71 TFLOPS FP32 operations.

Models. The models used for evaluation are detailed in Table II. We choose models from two popular LLM families: OPT [4] and GPT-2 [1]. For GPT-2, we mainly concentrate on the sparsity within multi-head attention, given that its activation function is GeLU [37]. Across all experiments, we employ mixed-precision techniques [38] adhere to common practices, utilizing FP16 for parameters and FP32 for activations.

PEFT Methods. Our evaluation encompasses on three exemplary PEFT techniques: LoRA [6], adapter [7], and bitfit [8], which are collectively the most widely utilized ones. Specially, we conduct our ablation studies and accuracy validation using LoRA owing to its broad acceptance within the domain.

Datasets. For performance evaluation, we apply a real-world dataset E2E [39] to ensure that the sparsity patterns within LLM reflect real-world situations. For accuracy validation, we first fine-tune the model on the widely used instruction dataset Alpaca [40] and then evaluate its accuracy directly without further fine-tuning across a variety of representative downstream tasks, each providing unique challenges as detailed in Table III.

Baselines. We compare the overall performance of our system with the state-of-the-art fine-tuning library PEFT [46], which stands as the most relevant benchmark to our knowledge. Specially, to evaluate the performance of our design on multi-head attention, we draw comparisons with two classic sparse attention methods, Big Bird [16] and Longformer [15].

B. Overall Performance

Execution Time. We evaluate the execution time and corresponding speedup of Long Expusure on A100 and A6000 platforms, as shown in Figure 7. Integrating Long Exposure into three exemplary PEFT techniques, we examine two different parameter sizes and sequence lengths for each one. The results indicate that our system achieves up to $1.25\times$ speedup on average for OPT-1.3B with a sequence length of 512 on A100. As the sequence length doubles to 1024, the average speedup increases to $2.49\times$. This enhancement is attributed to the use of sparse attention masks, which alter the computation complexity from $O(s^2)$ to $O(s)$. With a larger 2.7B model, the speedup remains consistent, averaging $1.44\times$ and $2.49\times$, respectively. Parallel results are observed on A6000, underscoring the robustness and reliability of our system. Since the introduction of sparsity reduces the execution time by decreasing the overall computation workload, this ensures a consistent speedup across different model sizes or platforms.

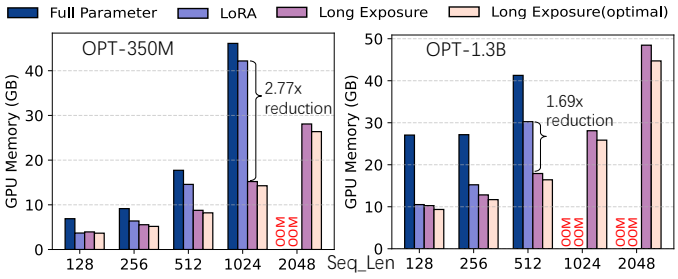


Fig. 8: Memory footprints of OPT fine-tuning on A100.

Memory Footprint. We also evaluate the memory footprints of LONG EXPOSURE as shown in Figure 8. Despite not being explicitly designed for memory efficiency, the application of head-specific sparse attention masks alters the memory complexity from $O(s^2)$ to $O(s)$, leading to lower memory footprints. Furthermore, selective activating model weights in MLP block permits the majority of the model to reside on the CPU, with only the active weights being transferred to the GPU for processing. This strategy can lead to additional memory savings, as presented by LONG EXPOSURE (optimal).

Model Accuracy. We investigate the impact of LONG EXPOSURE on model accuracy by comparing with original LoRA across a variety of downstream tasks, as shown in Tabel IV. We fine-tune OPT models of three distinct sizes on the Alpaca dataset. The results show that LONG EXPOSURE incurs only a minimal loss in downstream task accuracy across all model sizes and task types. This is because the essence of sparsity lies in disregarding the computation of elements that are zero or nearly zero, thereby only marginally affecting the final results.

C. Ablation Study

Performance Breakdown. Figure 10 provides a detailed performance breakdown. We measure the execution time for three major phases within fine-tuning: the forward pass, the backward pass and the optimizer step. Additionally, we measure the prediction time in LONG EXPOSURE to evaluate the overhead introduced by predictors. The results show that compared to full fine-tuning, PEFT techniques can significantly reduce the execution time for optimizer step, while leaving the forward and backward passes largely unaffected. Building on this, LONG EXPOSURE achieves further reductions in execution time for both forward and backward passes across all three PEFT techniques. Although predictors are introduced to capture the sparsity patterns at runtime, their overheads are proved minimal, ensuring that the efficiency gains are preserved.

TABLE IV: Comparative analysis of OPT model accuracy for downstream tasks after fine-tuning on the Alpaca dataset, with or without LONG EXPOSURE.

		350M-w/o	350M-w	1.3B-w/o	1.3B-w	2.7B-w/o	2.7B-w
PIQA	Acc.	65.13%	64.80%	72.25%	72.09%	74.70%	73.45%
	Stderr	1.11%	1.12%	1.05%	1.06%	1.02%	1.02%
Winog.	Acc.	53.04%	53.12%	58.88%	58.80%	62.27%	62.19%
	Stderr	1.40%	1.40%	1.38%	1.38%	1.37%	1.36%
RTE	Acc.	54.51%	55.60%	54.15%	54.51%	52.71%	53.79%
	Stderr	2.99%	3.01%	3.01%	3.01%	3.00%	2.04%
COPA	Acc.	69.00%	70.00%	81.00%	81.00%	78.00%	76.00%
	Stderr	4.61%	4.51%	4.23%	4.02%	4.29%	4.09%
Hella.	Acc.	32.26%	32.40%	42.08%	42.11%	46.76%	43.95%
	Stderr	0.47%	0.47%	0.499%	0.49%	0.50%	0.50%

Component I: Shadowy-sparsity Exposer. We begin by assessing the exposer's capability to capture model sparsity. Figure 9 (left) shows the sparsity ratios across different layers of OPT-1.3B when employing different methods. In multi-head attention, besides shadowy sparsity, we also use two traditional sparse attention methods, Longformer and Bigbird, as baselines. The results show that 'shadowy' presents the lowest sparsity ratio. While Longformer and Bigbird can identify more sparsity, their use of a uniform attention mask results in a trade-off with accuracy. LONG EXPOSURE outperforms all other methods by employing more granular head-wise masks that are adept at revealing the sparsity concealed within shadowy sparsity. In MLP block, shadowy sparsity exhibits a relatively low sparsity ratio, typically not exceeding 60% for most layers. LONG EXPOSURE utilizes a threshold-based filter to selectively ignore neurons that are activated but deemed less important. The results show that as the threshold (defined as a percentage of the peak values) is raised, the sparsity ratios increase correspondingly. By judiciously adjusting this threshold, LONG EXPOSURE strikes a balance between maintaining accuracy and enhancing efficiency.

In another view, we evaluate the corresponding performance gained from sparsity. Figure 9 (right) presents the execution time and speedups across different layers when fine-tuning OPT-1.3B. In multi-head attention, LONG EXPOSURE achieves $1.78\times$ speedup over the dense implementation and $1.33\times$ speedup over the 'shadowy' method. In MLP block, LONG EXPOSURE outperforms the dense implementation with a speedup of $4.22\times$. Particularly, the 'shadowy' baseline exhibits lower performance compared to the dense implementation. This is attributed to its unstructured sparsity pattern, which differs from the structured block-wise sparsity utilized by LONG EXPOSURE, resulting in a reduced arithmetic intensity.

Component II: Sequence-oriented Predictor. We first evaluate the necessity of predictors. We compare the fine-tuning loss curves of our system with those of two baselines that employ random sparse patterns in multi-head attention and MLP block, respectively. As Figure 11(a) shows, making accurate predictions of dynamic sparse patterns at runtime is crucial for model convergence with minimal loss. Beyond examining the loss curves, we also offer visual representations of the predictions from the multi-head attention's predictor. Figure 11(b) shows that the predicted attention scores can closely approximate the ground truth for identifying the proper sparse pattern. Within MLP block's predictors, we report recall metrics, achieving an impressive average of 96.35%.

Component III: Dynamic-aware Operator. We benchmark our operators against their dense counterparts under various sparsity ratios in Figure 12. In multi-head attention, the sparsity is applied block-wise, while in MLP block, the sparsity is neuron-wise. Both are aligned with the sparsity patterns

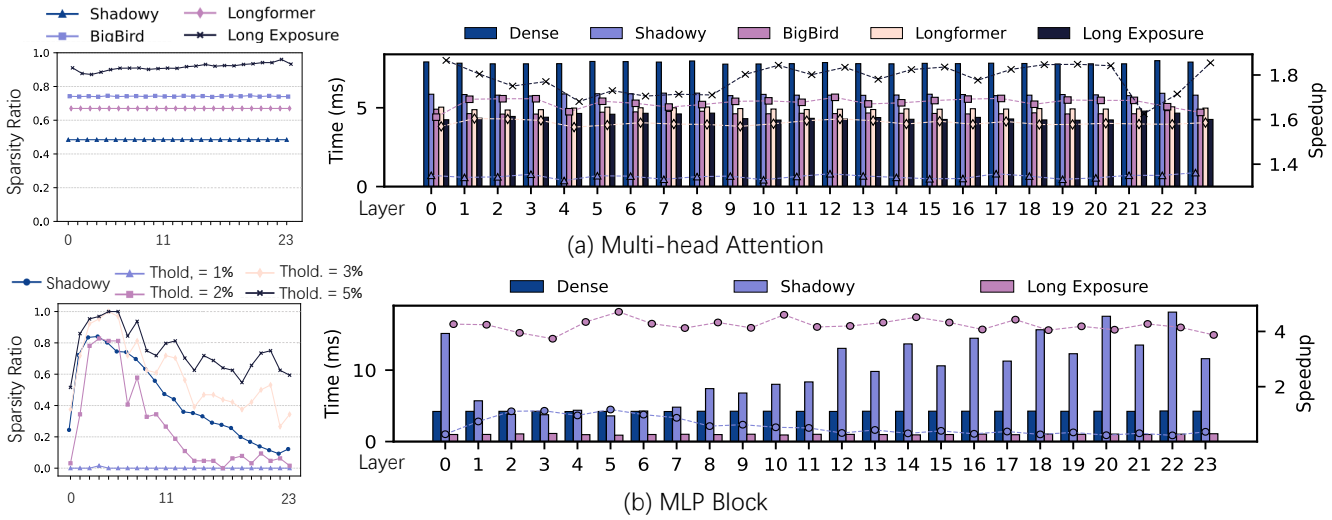


Fig. 9: Sparsity ratio (left) and corresponding performance (right) of LONG EXPOSURE across different layers of OPT-1.3B. The term 'Shadowy' denotes shadowy sparsity, which refers either to the sparsity in a uniform attention mask that covers all significant scores across all heads or to the sparsity present in the overall activations in MLP block.

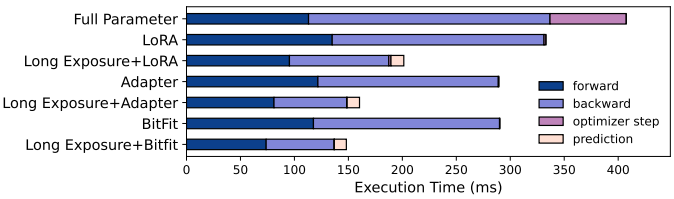


Fig. 10: OPT-1.3B fine-tuning performance breakdown.

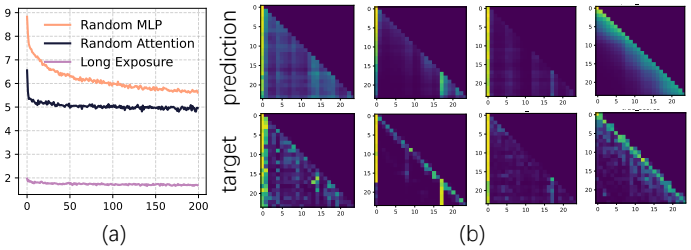


Fig. 11: Fine-tuning loss curve (a) and prediction visualizations of predictors in multi-head attention (b).

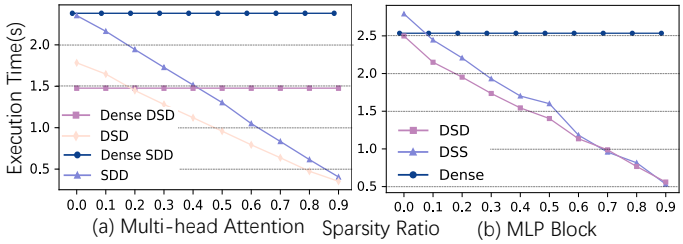


Fig. 12: Dynamic operator performance compared to dense.

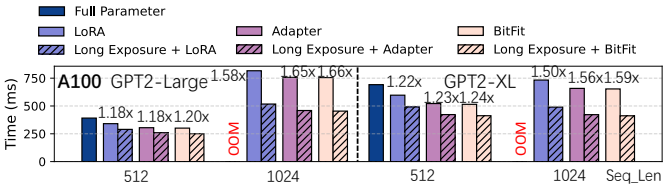


Fig. 13: Execution time per batch and speedup of GPT-2.

utilized in LONG EXPOSURE. The results show that all operators can attain speedups as sparsity ratio increases, achieving enhancements of up to $3-5\times$. Besides, the execution time of all dynamic operators exhibits an almost linear relationship with sparsity ratio, suggesting that our operators are adaptable and efficient in scenarios with dynamic sparsity levels.

D. Scalability

We explore the scalability of LONG EXPOSURE from two aspects. The first is the model type. Beyond OPT, we extend our experiments to GPT-2, a GeLU-based model. As shown in Figure 14, although only optimizations on multi-head attention are applied, LONG EXPOSURE consistently achieves average speedups of up to $1.63\times$ and $1.55\times$ on two different model sizes, respectively. The second is the number of GPU utilized. We maintain a constant dataset size and increase the GPU count to evaluate the strong scalability of our system. Figure 13 shows that the performance of our system scales linearly with the addition of more GPUs across three different model sizes. This is attributed to the fact that all optimizations within our system focus on the model computation workload, thereby introducing no extra communication overhead.

VIII. RELATED WORK

Model Pruning. Pruning aims to remove parameters without performance loss [47]. Static pruning stands out as one such

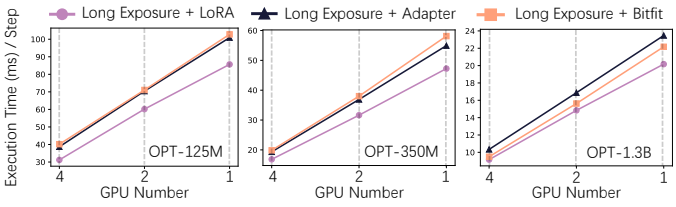


Fig. 14: Strong scalability of LONG EXPOSURE.

approach, which involves developing effective criteria for pruning models offline prior to subsequent inference [48]–[52]. In contrast, dynamic pruning is conducted during runtime. Certain studies incorporate this process into the model training [53]–[55], while others devise specialized optimizations for integration with fine-tuning [56], [57]. Different from these pruning techniques, LONG EXPOSURE does not prune any parameters but selectively activates a portion of them, preserving the model's original capacity for generalization. Furthermore, LONG EXPOSURE specifically aims to accelerate the fine-tuning process itself rather than the subsequent inference.

PEFT Optimization. The widespread adoption of PEFT techniques has spurred numerous studies focused on optimization. Some efforts aim to enhance PEFT performance. AdaLoRA [58] proposes to adaptively allocate the parameter budget by significance. LoHa [59] seeks to use more low-rank matrices for approximation and combines them with the Hadamard product. Some studies merge model compression with PEFT to improve further inference efficiency. Methods like QLoRA [60], SPA [61], PST [62], LRP [63] combine quantization or model pruning with PEFT. In addition, there are studies dedicated to optimizing PEFT's memory footprint. LST [64] introduces a ladder-side network for less gradient calculation. LoRA-FA [65] opts to freeze the down-projection matrix in LoRA for less activation memory. However, these studies primarily focus on algorithmic-level optimizations and overlook the wall-clock time impact of PEFT. LONG EXPOSURE accelerates PEFT holistically by addressing both algorithmic-level and system-level optimizations.

IX. CONCLUSION

We propose LONG EXPOSURE, a highly efficient system designed to accelerate parameter-efficient fine-tuning for LLMs. Our approach notably identifies the intrinsic sparsity within LLM fine-tuning and introduces three key components that systematically capture, predict, and exploit these sparse patterns. LONG EXPOSURE demonstrates up to $2.49\times$ speedup over state-of-the-art methods, underscoring the potential for more extensive exploitation of sparsity for PEFT acceleration.

X. ACKNOWLEDGMENT

The authors express their gratitude to the anonymous reviewers for their insightful and constructive feedback. The work of Ju Ren was supported in part by the National Key R&D Program of China under Grant No. 2022YFF0604502, the National Natural Science Foundation of China under Grant No. 62122095, 62341201 and 62072472, and by a grant from the Guoqiang Institute, Tsinghua University.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[4] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[6] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[7] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[8] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.

[9] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[10] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen *et al.*, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Machine Intelligence*, vol. 5, no. 3, pp. 220–235, 2023.

[11] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, 2021.

[12] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 97–110.

[13] Z. Zhou, J. Liu, Z. Gu, and G. Sun, "Energon: Toward efficient acceleration of transformers using dynamic sparse attention," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 1, pp. 136–149, 2022.

[14] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[15] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[16] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in neural information processing systems*, vol. 33, pp. 17 283–17 297, 2020.

[17] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[18] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena: A benchmark for efficient transformers," *arXiv preprint arXiv:2011.04006*, 2020.

[19] L. Liu, Z. Qu, Z. Chen, F. Tu, Y. Ding, and Y. Xie, "Dynamic sparse attention for scalable transformer acceleration," *IEEE Transactions on Computers*, vol. 71, no. 12, pp. 3165–3178, 2022.

[20] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, A. Shrivastava, C. Zhang, Y. Tian, C. Re *et al.*, "Deja vu: Contextual sparsity for efficient llms at inference time," in *International Conference on Machine Learning*. PMLR, 2023, pp. 22 137–22 176.

[21] Y. Song, Z. Mi, H. Xie, and H. Chen, "Powerinfer: Fast large language model serving with a consumer-grade gpu," *arXiv preprint arXiv:2312.12456*, 2023.

[22] K. Alizadeh, I. Mirzadeh, D. Belenko, K. Khatamifard, M. Cho, C. C. Del Mundo, M. Rastegari, and M. Farajtabar, "Llm in a flash: Efficient large language model inference with limited memory," *arXiv preprint arXiv:2312.11514*, 2023.

[23] N. Zheng, H. Jiang, Q. Zhang, Z. Han, L. Ma, Y. Yang, F. Yang, C. Zhang, L. Qiu, M. Yang *et al.*, "Pit: Optimization of dynamic sparse deep learning models via permutation invariant transformation," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 331–347.

[24] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[25] Z. Lin, A. Madotto, and P. Fung, "Exploring versatile generative language model via parameter-efficient transfer learning," *arXiv preprint arXiv:2004.03829*, 2020.

[26] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[27] D. Guo, A. M. Rush, and Y. Kim, "Parameter-efficient transfer learning with diff pruning," *arXiv preprint arXiv:2012.07463*, 2020.

[28] Z. Li, C. You, S. Bhojanapalli, D. Li, A. S. Rawat, S. J. Reddi, K. Ye, F. Chern, F. Yu, R. Guo *et al.*, "Large models are parsimonious learners: Activation sparsity in trained transformers," *arXiv preprint arXiv:2210.06313*, 2022.

[29] S. Team, "Sparse large language models with relu activation," 2023.

[30] I. Mirzadeh, K. Alizadeh, S. Mehta, C. C. Del Mundo, O. Tuzel, G. Samei, M. Rastegari, and M. Farajtabar, "Relu strikes back: Exploiting activation sparsity in large language models," *arXiv preprint arXiv:2310.04564*, 2023.

[31] N. Zheng, B. Lin, Q. Zhang, L. Ma, Y. Yang, F. Yang, Y. Wang, M. Yang, and L. Zhou, "SparTA: Deep-Learning Model sparsity via Tensor-with-Sparsity-Attribute," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 2022, pp. 213–232.

[32] H. Xia, Z. Zheng, Y. Li, D. Zhuang, Z. Zhou, X. Qiu, Y. Li, W. Lin, and S. L. Song, "Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity," *arXiv preprint arXiv:2309.10285*, 2023.

[33] M. Naumov, L. Chien, P. Vandermersch, and U. Kapasi, "Cusparse library," in *GPU Technology Conference*, 2010.

[34] T. Gale, M. Zaharia, C. Young, and E. Elsen, "Sparse GPU kernels for deep learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020*, 2020.

[35] T. Gale, D. Narayanan, C. Young, and M. Zaharia, "Megablocks: Efficient sparse training with mixture-of-experts," *Proceedings of Machine Learning and Systems*, vol. 5, 2023.

[36] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI open*, vol. 3, pp. 111–132, 2022.

[37] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[38] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[39] J. Novikova, O. Dušek, and V. Rieser, "The E2E dataset: New challenges for end-to-end generation," in *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany, 2017, arXiv:1706.09254. [Online]. Available: https://arxiv.org/abs/1706.09254

[40] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[41] Y. Bisk, R. Zellers, J. Gao, Y. Choi *et al.*, "Piqa: Reasoning about physical commonsense in natural language," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7432–7439.

[42] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.

[43] D. Giampiccolo, B. Magnini, I. Dagan, and W. B. Dolan, "The third pascal recognizing textual entailment challenge," in *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007, pp. 1–9.

[44] M. Roemmele, C. A. Bejan, and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in *2011 AAAI Spring Symposium Series*, 2011.

[45] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[46] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[47] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[48] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 68, no. 1, pp. 49–67, 2006.

[49] S. Hanson and L. Pratt, "Comparing biases for minimal network construction with back-propagation," *Advances in neural information processing systems*, vol. 1, 1988.

[50] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2554–2564.

[51] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 304–320.

[52] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[53] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *arXiv preprint arXiv:1808.06866*, 2018.

[54] A. Davis and I. Arel, "Low-rank approximations for conditional feedforward computation in deep neural networks," *arXiv preprint arXiv:1312.4461*, 2013.

[55] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," *arXiv preprint arXiv:1511.06297*, 2015.

[56] V. Sanh, T. Wolf, and A. Rush, "Movement pruning: Adaptive sparsity by fine-tuning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 378–20 389, 2020.

[57] D. Xu, I. E. Yen, J. Zhao, and Z. Xiao, "Rethinking network pruning–under the pre-train and fine-tune paradigm," *arXiv preprint arXiv:2104.08682*, 2021.

[58] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," in *The Eleventh International Conference on Learning Representations*, 2022.

[59] N. Hyeon-Woo, M. Ye-Bin, and T.-H. Oh, "Fedpara: Low-rank hadamard product for communication-efficient federated learning," *arXiv preprint arXiv:2108.06098*, 2021.

[60] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[61] L. Hedegaard, A. Alok, J. Jose, and A. Iosifidis, "Structured pruning adapters," *arXiv preprint arXiv:2211.10155*, 2022.

[62] Y. Li, F. Luo, C. Tan, M. Wang, S. Huang, S. Li, and J. Bai, "Parameter-efficient sparsity for large language models fine-tuning," *arXiv preprint arXiv:2205.11005*, 2022.

[63] J. Sun, S. Lapuschkin, W. Samek, and A. Binder, "Explain and improve: Lrp-inference fine-tuning for image captioning models," *Information Fusion*, vol. 77, pp. 233–246, 2022.

[64] Y.-L. Sung, J. Cho, and M. Bansal, "Lst: Ladder side-tuning for parameter and memory efficient transfer learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 991–13 005, 2022.

[65] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li, "Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning," *arXiv preprint arXiv:2308.03303*, 2023.