EPAS: Efficient Training with Progressive Activation Sharing

Anonymous ACL submission

Abstract

We present a novel method for **E**fficient training with **P**rogressive **A**ctivation **S**haring (EPAS). This method bridges progressive training paradigm with the phenomenon of redundant representation across deeper layers of transformers. EPAS gradually grows an activation sharing region during training by switching decoder layers to activation sharing mode. This results in throughput increase due to reduced compute. To utilize deeper layer redundancy, the sharing region starts from the deep end of the model and grows towards the shallow end. The EPAS trained models allow for variable activation-sharing region lengths for different compute budgets during inference. Empirical evaluations with attention sharing ($Q,\ K$) in LLaMA models ranging from 125M to 7B parameters show up to an 11.1% improvement in training throughput and up to a 29% improvement in inference throughput while maintaining similar loss curve to the baseline models. Furthermore, applying EPAS in continual pretraining to transform TinyLLaMA into an attention-sharing model yields up to a 10% improvement in average accuracy over state-of-the-art methods, emphasizing the significance of progressive training in activation-sharing models.

1 Introduction

Recent research in computational efficiency of Transformers has focused on efficient pretraining (He et al., 2024a), continual learning (Zhao et al., 2024), fine-tuning (Zhang et al., 2024b; Liu et al., 2024b; Ge et al., 2024) and inference (Alizadeh et al., 2024; Wu and Tu, 2024). However, holistic approaches to efficient training and inference remains underexplored as evident from large accuracy-efficiency trade-offs in this direction (Kim and Lee, 2024; Song et al., 2024). Surprisingly, large transformer models are found to compute redundant activations across deeper layers (Liao and Vargas, 2024; Brandon et al., 2024).

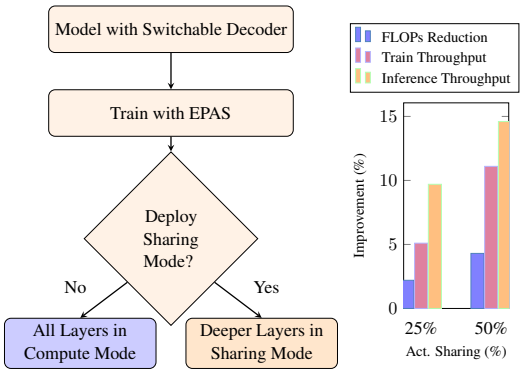


Figure 1: **Left:** Overall solution from efficient training to inference using EPAS. **Right:** TinyLLaMA (Zhang et al., 2024a) model FLOPs reduction and train/inference throughput improvement expanding activation sharing to 25% and 50% of the decoder layers.

Therefore, as a promising yet less explored direction, we focus on utilizing redundancy phenomenon towards an unified efficiency solution to training and inference with minimal tradeoffs to accuracy.

Deeper layers of transformer models have been found to exhibit redundancy in activations inside the attention block; for example, multiple deeper layers are found to compute mostly similar attention scores (Liao and Vargas, 2024; Ying et al., 2021; Mu et al., 2024). As an obvious direction, methods have been proposed to compute only the value,V, and reuse the computed attention score directly from a previous layer to make the model compute efficient (Liao and Vargas, 2024; Ying et al., 2021; Mu et al., 2024). Since attention score is not directly available in block factoring-based efficient attention algorithms, such as Flash-Attention (Dao et al., 2022; Dao, 2023), an alternative approach is sharing the query,Q, and the key,K across layers (Rajabzadeh et al., 2024). Meanwhile, some other approaches have proposed to compute the query,Q, and reuse the key,K, and value,V, from

a previous layer (Brandon et al., 2024; Sun et al., 2024). The sharing of $Q,\ K$ offers greater computational savings, while sharing $K,\ V$ has greater impact in reducing inference memory footprint. These approaches are generally known as activation sharing.

State-of-the-art activation sharing approaches enhance efficiency mostly by sharing activations across the layers of trained models during the inference (Liao and Vargas, 2024) or follow model distillation (Rajabzadeh et al., 2024). Few models incorporate efficient design from the training phase, and those that do primarily focus on optimizing inference efficiency while overlooking training efficiency (Sun et al., 2024). Since efficiency in both training and inference presents diverse design challenges, there is a growing need for a simple, holistic solution that addresses both aspects.

In another direction, efforts for efficient training has presented methods for progressive growth (Gong et al., 2019; Wu et al., 2024; K. et al., 2025), progressive layer drop (Zhang and He, 2020), and progressive dataset complexity (Li et al., 2022). Conversely, efficient inference approaches have focused on model pruning (Cheng et al., 2024), ,distillation (Yang et al., 2024), progressive low-rank decomposition (Hajimolahoseini et al., 2024b) and step-by-step distillation (Hu et al., 2022; Liu et al., 2024a; Hsieh et al., 2023). From a broader perspective, the progressive modification of model during training has proven superior to directly training the modified architecture, often with an additional advantage of many-in-one models (Cai et al., 2024).

The proposed progressive activation sharing combines progressive training and efficient inference in a unified training method for activation sharing models. This method allows for utilizing redundancy observed in deeper layers from early phases of training while preserving model accuracy. It improves pretraining throughput to reduce time to accuracy and derives a family of efficient models from a single end-to-end training process. Additionally, it enables flexible transformation of pretrained models into activation sharing models through a single, efficient continual pretraining process without requiring multiple rounds of knowledge distillation. Instead of sharing activations during inference of a pretrained model, an activation-sharing region is progressively expanded during pretraining or continual pretraining. This makes computation lighter as training progresses. This hot switching to activation sharing during training is achieved through a switchable decoder block that can conditionally reuse activation. The training algorithm employs an activation-sharing scheduler that activates sharing at configured intervals to gradually increase the size of the sharing block, enhancing efficiency throughout the training process. The key contributions are:

- EPAS enables faster training and flexible efficient model configurations during inference.
- EPAS transforms pretrained models into efficient ones through continual pretraining, eliminating the need for knowledge distillation.
- The proposed method achieved superior training and inference throughput while maintaining accuracy on par with baseline.

2 Progressive Activation Sharing

The proposed **E**fficient training with **P**rogressive **A**ctivation **S**haring (EPAS) method builds transformer models using the proposed switchable activation sharing decoder layer as building block. This decoder layer is a simple yet elegant extension that incorporates conditional activation sharing into the standard decoder layer. The term activation sharing generally refers to reusing some shared activations, such as attention scores, or $Q,\ K$, or $K,\ V$, from an early layer. This help to reduce the computation of somewhat redundant representations in the deeper parts of the model. At a high level, the proposed progressive activation sharing involves gradually growing the number of layers using activation sharing and hence increasing the throughput. In the following, we discuss the details of the newly designed decoder layer and the progressive training algorithm.

2.1 Switchable Activation Sharing Decoder

The hot switching of decoder layers to activation sharing mode during the progress of training is performed by switching a conditional branching of computation in the decoder layer. A pictorial illustration of this decoder extension, with a particular example of attention sharing, is presented in Figure 2. This example demonstrates sharing of $Q,\ K$ to make attention sharing compatible with Flash-Attention. The extension is quite simple without requiring any additional parameters. Hence, the modified architecture can reuse previously trained parameters for continual pretraining or post-training.

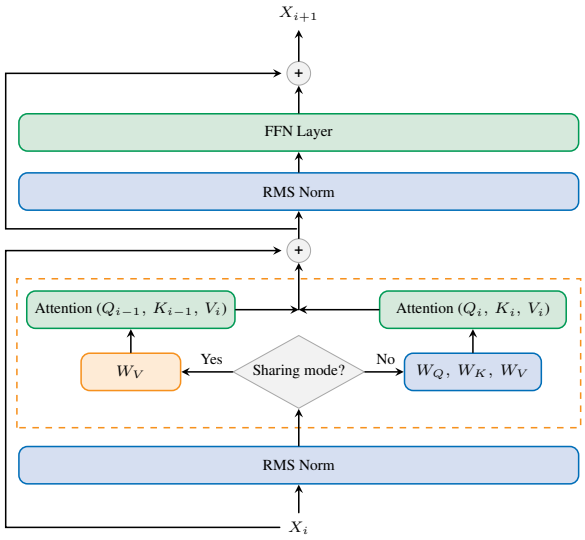


Figure 2: The Switchable Activation Sharing Decoder with an example of attention sharing (Q, K). This decoder layer extends conventional transformer decoder layer by adding a conditional switching branch to reuse Q_{i-1}, K_{i-1} from previous layer instead of computing in current layer(left branch inside dashed box). When not using activation sharing mode, the computation follows the right branch as like convention decoder layer.

This decoder simply branches out to either reuse some selected activations from a previous layer or to compute with current layer's own parameters. The most conventional activation sharing models use either attention scores, or $Q,\ K$, or $K,\ V$ as the set of activations for this purpose.

2.2 Progressive Activation Sharing

The progressive activation sharing approach aims to gradually expand the activation sharing region by switching decoder layers to activation sharing mode throughout the training steps. The growing of sharing layers follows a deterministic sharing strategy. The rationale behind deterministic sharing is to ensure that shared layers continuously grow, ultimately resulting in a model requiring fewer FLOPs than baseline during the inference.

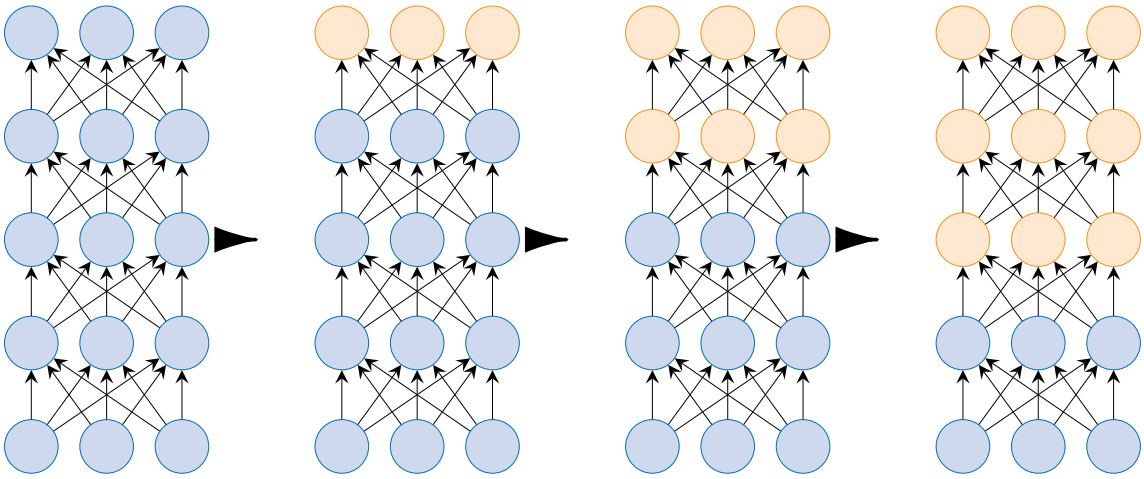
Initially, training begins with all of the L layers of a model, M, in compute mode. At this stage the model works like a conventional transformer model (Vaswani et al., 2017). A target sharing region, S, defines a list of layers that will progressively transition into activation sharing mode. We found that selecting a set of deeper layers and activated sharing sequentially from deep to shallow layers works well in this scenario. Over the course of T training steps, a group of B layers at the deep end of S is switched to activation sharing mode at every interval of I training steps. Rather than enabling activation sharing of the whole sharing region from the beginning of training, the method gradually expands the sharing region by gently allowing the model to adapt to activation sharing. Following this training strategy, it ends up with a target model of a predefined maximum activation sharing region. The steps are depicted with a schematic example in Figure 3.

In this proposed activation sharing scheme, the layer immediately before the sharing region shares its activations with the layers in sharing group. During the forward pass, if a layer detects that its subsequent layer is in sharing mode, it populates an activation cache with a selected set of activations, A. This progressive activation sharing method is compatible with sharing attention scores, $Q,\ K$, or $K,\ V$. The trained model benefits from reduced model FLOPs by leveraging the last state of the activation sharing group. Algorithm 1 presents the progressive activation sharing training algorithm instantiated with Q, K sharing; the adaptation to other forms of activation sharing, such as $K,\ V$ sharing is straightforward.

2.3 Applications

Progressively growing the sharing block during training gradually reduces model FLOPs and increases training throughput (tokens/sec). This results in reduced training time and cost, while the gradual change in the computation graph allows for training stability. EPAS enhances computational resource utilization and improves pretraining efficiency by minimizing redundant representations, thereby achieving a balanced trade-off between efficiency and model performance. EPAS can also transform existing pretrained models into activation-sharing architectures while used in continual pretraining setting. Moreover, continual pretraining with EPAS enables flexible sub-network selection during inference, allowing a single end to end training on a small dataset to derive a family of efficient models. This eliminates the need for complex multi-model training or repeated distillation.

During the inference phase, the activation sharing models trained with EPAS can achieve superior throughput compared to baseline models while maintaining similar performance. Attention score sharing, or $Q,\ K$ sharing, reduces computational overhead and minimizes $K,\ V$ cache memory requirement as the sharing layers only need to cache



(a) Baseline (b) Begin activation sharing (c) Grow sharing region (d) Activation sharing model

Figure 3: An example of EPAS training approach. Beginning with all L layers in **compute-mode** (e.g., 5 layers here), a region of B layers transition into **sharing-mode** (e.g., 1 layer here) at every I step intervals. The progressive growth continues till maximum S layers (e.g., 3 layers here) are in sharing mode. The trained model can be used either all layer in compute mode or up to S layers in activation sharing mode.

Algorithm 1 Progressive Activation Sharing

Input: Model, M, Interval, I, Target Sharing Layers, S_c, Sharing Region Growth Size B
Output: Trained model M
L**:** Layers in Base Model
T**:** Training Steps
S**:** Layers currently in sharing mode
assert $(B \geq 1 \ \ \& \ \ |S_c| \geq 1)$
assert $(|S_c| \mod B = 0)$
$C \leftarrow [], S \leftarrow []$
for $t = 1$ to T **do**
  for $i = 0$ to $|L| - 1$ **do**
    if $L[i] \in S$ **then**
      $Q_i, K_i \leftarrow Q_{i-1}, \ K_{i-1}$ from C
      Compute V_i
    else
      Compute $Q_i, \ K_i, V_i$
    end if
    Do the rest of the computations
    if $L[i+1]$ has sharing on **then**
      $C \leftarrow Q_i, K_i$
    end if
  end for
  if $t\%I == 0 \ \ \& \ \ |S_c| \geq B$ **then**
    $L_c \leftarrow$ pop last B element from S_c
    $S \leftarrow L_c + S$ ▷ Append beginning
  end if
end for
return M

V. Conversely, $K, \ V$ sharing saves more memory as it eliminates the need to cache both K and V in the sharing layers while offering less computational reduction. In either of the case, the inference process becomes faster and more resource-efficient compared to baseline model.

3 Experiments

To demonstrate the efficacy of EPAS, we considered $Q, \ K$ sharing as a particular instance of activation sharing. The experiments integrate EPAS with open-source transformer-based LLM. In particular, we perform extensive experiment with TinyLLaMA-1.1B (Zhang et al., 2024a) as our primary baseline and then further extend our empirical analysis with more LlaMA-based models (Touvron et al., 2023a,b) ranging from 125M to 7B parameters. We used a small subset of the open-source SlimPajama-627B dataset for the pretraining and continual pretraining experiments (Doe). Since our pretraining experiment is focused on efficiency analysis and comparing training dynamics for small number of training steps rather than training till convergence, this set up is sufficient for the intended purpose.

The empirical analysis compared training and inference efficiency as well as learning capacity during training. We measure theoretical FLOPs reduction, training throughput (tokens/sec) and inference throughput (tokens/sec). For evaluation of transformed pretrained model with continual pretrain-

Model	Model FLOPs/sample (TF)		
	Baseline	Q/K-Sharing	Reduction (%)
125M	1.81	1.72	4.9%
1.1B	14.98	14.34	4.3%
3B	41.41	38.39	7.3%
7B	85.62	79.21	8.1%

Table 1: Model FLOPs per sample in Terra-FLOPs (TF) for baseline versus Q, K sharing of 50% of the layers.

Model	Train Tokens/sec		
	Baseline	Q/K-Sharing	Improvement(%)
125M	15458.9	17143.1	10.8%
1.1B	3850.2	4259.8	11.1%
3B	1783.1	1961.3	10.9%
7B	1259.8	1367.9	8.6%

Table 2: Model size scaling and training efficiency on V100 GPU with distributed training setup.

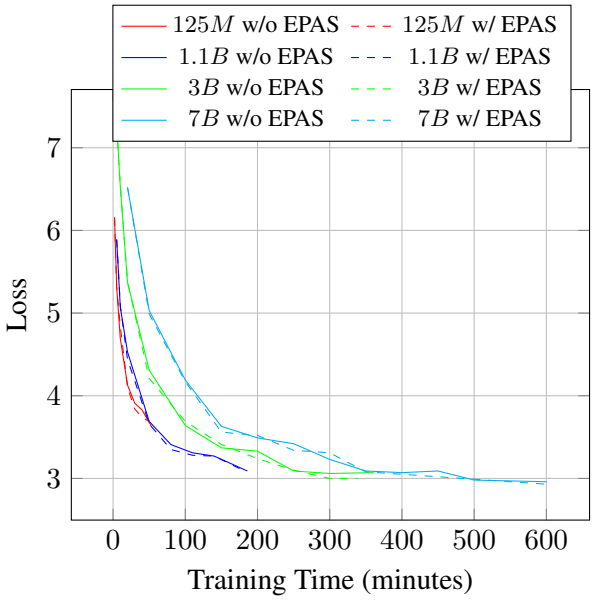


Figure 4: Smoothed loss versus time while scaling model sizes across LLaMA models with $125M$, $1.1B$, and $3B$ parameters. Models trained with EPAS also exhibit slightly faster convergence during training in addition to have higher throughput during inference.

ing setup of EPAS, we used lm-eval-harness (Gao et al., 2024). Furthermore, to compare the efficiency across diverse devices, experiments are conducted on Nvidia V100 GPU, Ascend 910A NPU, and Ascend 910B NPU.

Furthermore, we present extensive ablation study for critical understanding and justification of our findings and corresponding design choices. While the proposed method is compatible to share various activations, for the scope of this research we limit empirical analysis on attention sharing only. In particular, we used cross-layer query, Q, and key, K, sharing so that the method is compatible with Flash-Attention (Dao et al., 2022; Dao, 2023).

3.1 Training efficiency

To empirically assess training efficiency, we conduct model FLOPs and training throughput analysis by scaling model sizes from 125M to 7B following LLaMA architectures. Although EPAS presents a generalized training algorithm to train activation sharing architecture, we demonstrate for the example of half of the layers in final sharing mode following recent trends (Sun et al., 2024). Table 1 summarizes the reduction of theoretical model FLOPs for each of the model when sharing Q/K across second half of the layers. We observe up to 8% FLOPs reduction with this configuration. Table 2 presents the improvement in training tokens/second with a distributed training setup on 8 V100 GPU. The models show up to 11% train throughput improvement with the above configuration.

We further investigated the learning capacity of activation sharing model when trained with EPAS compared to training baseline model without activation sharing. This experiment was conducted for $0.25M$ tokens per step for 4000 steps resulting in a total of $1B$ tokens. We consider this setup sufficient for comparing the training dynamics of the baseline and activation sharing models by analyzing loss curves without conducting a full LM benchmark evaluation similar to recent literature (Sun et al., 2024).

We observe that EPAS has lower loss at equal time and needs less time to achieve equal loss as baseline. We present a loss curve comparison for scaling analysis of training w/ and w/o EPAS in Figure 4. The comparison of train loss vs. time shows faster training and convergence with EPAS while maintaining similar loss curve pattern as the baseline. This is particularly evident from the observation that at any given time during training, EPAS shows a lower loss, especially in the early phases of training.

The results in Table 4 compare the total training time and the final validation loss after training. The findings show that EPAS significantly speeds up training, while the difference in final validation loss remains negligibly small (less than 0.05), indicating faster convergence without sacrificing accuracy or increasing the risk of overfitting.

We also considered extending our experiments

Device Name	Device Spec	Single Device			Distributed (8 Device)		
		Baseline	Q/K Sharing	Improvement(%)	Baseline	Q/K Sharing	Improvement(%)
V100 GPU	32GB, 125 TF	4079.6	4423.6	10.8%	3850.2	4259.8	11.1%
910A NPU	32GB, 278 TF	4321.3	4874.2	12.8%	4788.2	5246.9	9.57%
910B NPU	64GB, 378 TF	11354.1	12443.7	9.6%	12902.1	13844.5	7.3%

Table 3: Empirical evidence of training efficiency across different hardware for activation sharing ($Q,\ K$) in the second half of the layers of the TinyLLaMA model.

Model	**Train Time (hh:mm:ss)**	**Validation Loss**
125M (w/o EPAS)	00:54:12	3.74
125M (w/ EPAS)	00:50:50	3.79
1.1B (w/o EPAS)	03:07:18	3.19
1.1B (w/ EPAS)	02:56:33	3.22
3B (w/o EPAS)	06:02:46	3.06
3B (w/ EPAS)	05:39:11	3.09
7B (w/o EPAS)	11:09:26	2.99
7B (w/ EPAS)	10:25:17	3.04

Table 4: Total time and final validation loss for several LLaMA models of varying parameter sizes, demonstrating that EPAS training is significantly faster with a negligible difference in final validation loss.

.

for various hardware types. For this cross hardware experiment, we fix a model (TinyLLaMA) and perform the same analysis across various hardware. Table 3 presents the observation of training efficiency across different hardware categories. The table shows a negligibly minor variation in train throughput improvement while changing hardware. In particular, the Q, K sharing model still maintains 8-10% train throughput improvement across the three types of hardware. This evidence further justifies that activation sharing makes a generic efficiency improvement on model's computation that persists across varieties of device specs.

3.2 Inference efficiency

We compare the generation throughput in terms of tokens/sec to measure the inference efficiency improvement of the proposed method. Since models trained with EPAS present a flexible architecture that can be used with various number of activation sharing layers during inference, we present inference throughput for sharing 25% and 50% of the layers. For example, LLaMA1.1B has 22 layers, hence 25% and 50% indicates 5 and 11 layers in sharing mode respectively. We observe a 3–22% improvement in generation throughput when sharing 25% of the layers, and a 6–30% improvement when sharing 50% of the layers across different models. The results are summarized Table 5.

Model	Inference Throughput(tok/sec)		
	Baseline	25% Sharing	50% Sharing
125M	63.8	78.3 (22.7%)	82.4 (29.2%)
1.1B	39.1	42.9 (9.7%)	44.8 (14.6%)
3B	38.1	40.9 (7.3%)	43.2 (13.4%)
7B	24.46	25.27 (3.3%)	26.03 (6.4%)

Table 5: Inference throughput of various models on V100 GPU with 25% and 50% of the layer in sharing mode.

3.3 Language Model Evaluations

We consider to evaluate model trained with EPAS in a continual pretraining setup to transform base models to attention-sharing models. We follow this direction due to the resource intensity of pretraining from scratch. For fair comparison, we also do continual pretraining on base models with same dateset. This experiment begins with a pretrained checkpoint and follows continual pretraining with progressively growing the activation sharing block from the deep end of the model towards the shallow end gradually growing one large activation sharing region. No additional sophisticated training approaches, e.g., knowledge distillation or additional auxiliary loss computation are used. In particular, we consider a pretrained TinyLLaMA model as baseline. Due to limited resources we constrain the activation sharing block to grow up to 25% of the model depth(e.g., 5 out of 22 layers). The training is done in a distributed setup of 8 device with a batch size 8 per device and gradient accumulation steps 16 to match the effective token per step being equal to the baseline pretraining configuration (e.g., $8 \times 8 \times 16 \times 2048 = 2M$). The training is run for $2K$ steps resulting in total 4 billion tokens. We use the lm-eval-harness (Gao et al., 2024) for evaluating the models on LM benchmarks.

The checkpoint from continual pretraining with EPAS is evaluated against applying inference time attention sharing, known as Beyond-KV-Cache (Liao and Vargas, 2024) under various activation sharing configurations. We first compare the baseline and EPAS trained model with both

of the models in full-compute mode. Then, we compare by using three and five layers in sharing mode. We observe that the model trained without EPAS shows a large drop in accuracy when evaluated with sharing mode. In contrast, EPAS trained model can retain most of the accuracy in attention sharing mode. When using five layers in sharing mode for both of the models, the EPAS trained model shows around 10% higher accuracy. The results are presented in Table 6, where each row pair shares the same activation sharing configuration.

Full model evaluation during inference without activation sharing improves accuracy in the EPAS-trained model. As expected, baseline accuracy declines more rapidly as additional layers are included in the sharing block. In contrast, the EPAS-trained model maintains robust accuracy as the activation sharing block expands. Notably, with a sharing block size of five layers (25% of model depth), the EPAS-trained model outperforms the baseline by approximately 12%. These findings highlight EPAS's effectiveness in preserving accuracy while progressively sharing activations, offering a robust approach for efficient transformer model training and inference.

3.4 Ablation Experiment

Impact of Last Layer. Recent research presents diverging perspectives on activation sharing regarding the role of the final layer. One approach argues for excluding the last layer or retaining it solely during inference due to its distinct attention pattern (Liao and Vargas, 2024), while an alternative view supports its inclusion within the activation-sharing block (Sun et al., 2024). To assess the optimal configuration, we conducted an analysis using continual pretraining of TinyLLaMA with EPAS, evaluating LM benchmark performance on a small dataset under two conditions: exclusion versus inclusion of the last layer within sharing blocks. As summarized in Table 7, the results indicate minimal performance differences, with a slight advantage observed when incorporating the last layer during training. Consequently, we adopt this strategy in our approach.

Single vs. Multiple Sharing Block. Recent approaches of attention, Q, K and K, V sharing demonstrate two distinct methodologies for applying activation sharing across multiple layers. One line of research advocates for small activation-sharing blocks (Ying et al., 2021; Rajabzadeh et al., 2024) while others argue that such fine-grained partitioning adds unnecessary complexity and computational overhead per block and favors instead a large activation-sharing block in the deeper end of the model (Sun et al., 2024). To examine the trade-off between simplicity and computational efficiency, we conduct a comparative experiment with an equal number of layers in activation reuse mode for TinyLLaMA. One setup employs three groups of two activation-reusing layers, while the computationally equivalent model with a single block arranges three activation-reusing layers sequentially, as schematically depicted in Fig. 5. Results in Table 8 consistently show superior performance with a single large activation-sharing block, attributed to deeper-layer sharing. This trend holds across training settings with and without EPAS. Adhering to Occam's razor, we favor the simplicity of a single large sharing block, as additional complexity yields no clear advantage.

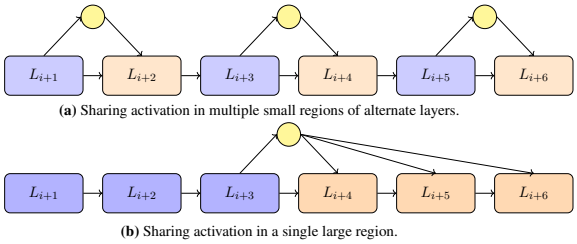


Figure 5: Schematic illustration of layer grouping: multiple small blocks versus a single large block. The colors indicate **compute-** and **sharing-mode**. The number of layers shown is for illustrative purposes; both approaches can accommodate a variable number of layers.

4 Related Works

Accelerating deep neural networks has been a tremendous research attention resulting in various approaches, such as focused on training data (Li et al., 2022; Hajimolahoseini et al., 2023b; Ataiefard et al., 2024), model parameters (Hajimolahoseini et al., 2023a; Ahmed et al., 2023; Hajimolahoseini et al., 2024a), computation graph (Fan et al., 2020; Zhang and He, 2020; He et al., 2024b) and activation sharing (Javadi et al., 2023). Notably, training efficiency has focused on progressively freezing some layers (Brock et al., 2017), progressive stacking (Gong et al., 2019), progressive layer drop (Zhang and He, 2020), and progressively increasing training overload in parallel to model growth (Li et al., 2022). Conversely, efficient inference has focused on model pruning (Cheng et al.,

Model	WG	PIQA	BoolQ	ARC-C	ARC-E	OBQA	HS	SciQ	LM(oa)	LM(std)	RTE	Average
w/o EPAS (No Sharing)	59.12	73.56	56.09	32.68	55.51	36.80	61.45	84.20	56.70	50.55	57.04	56.70
w/ EPAS (No Sharing)	58.25	73.01	63.33	31.57	56.44	36.20	58.17	85.90	56.10	52.45	56.32	57.07
w/o EPAS (Sharing 3/22)†	60.14	73.50	48.41	32.25	53.91	37.40	60.48	75.80	25.00	23.17	47.65	48.88
w/ EPAS (Sharing 3/22)	59.27	73.07	62.72	33.02	56.14	36.20	58.72	83.60	49.97	47.80	52.35	55.71
w/o EPAS (Sharing 5/22)†	55.64	67.03	48.41	27.13	42.55	31.00	51.51	75.80	25.00	23.17	47.65	44.99
w/ EPAS (Sharing 5/22)	58.02	73.18	60.70	31.23	55.85	36.00	57.95	83.10	47.58	44.69	50.90	54.47

Table 6: Evaluation of multiple inference configurations from a single trained model. The EPAS model uses continual pretraining with a target sharing region of five layers while varied sharing region length during inference. † represents our re-implementation of (Liao and Vargas, 2024) for smaller model for various sharing settings.

Model	**PIQA**	**WG**	**BoolQ**	**OBQA**	**HS**
Excluding last layer	73.29	58.01	52.94	36.00	58.66
Including last layer	**73.39**	**59.12**	**56.51**	**36.80**	**59.11**

Table 7: LM benchmark evaluation of continual pretraining reveals a minor difference between including or excluding the last layer in activation sharing.

Model	**PIQA**	**WG**	**BoolQ**	**OBQA**	**HS**
MB (w/o EPAS)	72.25	58.01	**49.02**	36.20	60.15
SB (w/o EPAS)	**73.50**	**60.14**	48.41	**37.40**	**60.84**
MB (w/ EPAS)	**73.67**	57.93	60.06	36.40	58.28
SB (w/ EPAS)	73.45	**58.41**	**60.31**	**37.00**	**58.55**

Table 8: Results comparing using a single large sharing block (SB) vs. multiple small sharing blocks (MB)

2024), low-rank adaptation (Hu et al., 2022; Liu et al., 2024a), and distillation approaches (Yang et al., 2024) where recent works claimed to find progressive low-rank decomposition (Hajimolahoseini et al., 2024b) and step by step distillation (Hsieh et al., 2023) as superior than single step low-rank decomposition or distillation.

Cross layer activation sharing to leverage the redundancy observed in representation of deeper layers of transformers has recently emerged as a promising direction for enhancing model efficiency. Prominent cross-layer activation sharing methods include sharing attention scores, queries and keys ($Q,\ K$), or keys and values ($K,\ V$) across layers. For example, LazyFormer (Ying et al., 2021) introduced "lazy blocks," where the first layer in a group computes the attention scores and shares them with subsequent layers. ShareAttn (Liao and Vargas, 2024) later extended this idea by using a single large block during inference following pretrained model analysis. LISA (Mu et al., 2024) further enhanced attention score sharing by adding transformations to better align shared representations. However, these methods are incompatible with efficient block-factorized attention mechanisms like Flash-Attention (Dao et al., 2022; Dao, 2023). An alternative strategy focused on sharing $Q,\ K$ across layers and employed distillation-based techniques to train the model (Rajabzadeh et al., 2024). Cross-layer $K,\ V$ sharing, on the other hand, reduces inference memory requirements by eliminating the need for separate K/V projections in each layer (Brandon et al., 2024; Sun et al., 2024).

While prior activation sharing approaches improved inference speed, there remains challenge in training these models and closing the gap in accuracy. Prior works in progressive training methods have been largely restricted to layer stacking or dropping and have not explored activation sharing. In contrast, EPAS introduces progressive activation sharing by bridging the ideas of the two areas to address the training challenges and severe accuracy drop of state-of-the-art activation sharing models.

5 Conclusion

EPAS presented an efficient transformer training algorithm incorporating a switchable decoder layer. It integrates cross-layer activation sharing as a generic property of the model and training process rather than a drop in modification of trained model. This design showcased a comprehensive solution for improving efficiency in both training and inference, balancing optimization to maintain performance while reducing computational demands. By applying EPAS during continual pretraining, pretrained models can be efficiently converted into activation-sharing models with adaptable computational budgets. Empirical evaluations demonstrated that EPAS enhances training and inference throughput with accuracy similar to the baseline. These findings underscore the potential of redundancy-aware training and inference as a scalable approach to optimizing transformer models. Furthermore, EPAS holds promise for extending to post-training as well as other domains including vision and speech, which merit further exploration in future research.

Limitations

The current state of EPAS is evaluated with a focus on small language models, while its extension to large language models is considered a future work. Empirical evaluation with large-scale models to explore structural modifications for better learning of activation sharing remains a limitation, which we plan to address in future studies. Despite the lack of empirical evidence, we expect activation sharing to perform well in deeper models. Our focus has been on efficient training of models with activation sharing, without further consideration of potential architectural modifications to cross-layer activation caching for improved learning. Moreover, empirical evaluations have been limited to transformer-based language models, while we consider validation for vision transformers as a future task. Additionally, adaptation of activation sharing to models other than transformers has not yet been explored in previous literature or within EPAS.

References

Walid Ahmed, Habib Hajimolahoseini, Austin Wen, and Yang Liu. 2023. Speeding up resnet architecture with layers targeted low rank decomposition. *arXiv preprint arXiv:2309.12412*.

Keivan Alizadeh, Seyed Iman Mirzadeh, Dmitry Belenko, S. Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2024. LLM in a flash: Efficient large language model inference with limited memory. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12562–12584, Bangkok, Thailand. Association for Computational Linguistics.

Foozhan Ataiefard, Walid Ahmed, Habib Hajimolahoseini, Saina Asani, Farnoosh Javadi, Mohammad Hassanpour, Omar Mohamed Awad, Austin Wen, Kangling Liu, and Yang Liu. 2024. Skipvit: Speeding up vision transformers with a token-level skip connection. *arXiv preprint arXiv:2401.15293*.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. 2024. Reducing transformer key-value cache size with cross-layer attention. *arXiv preprint arXiv:2405.12981*.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2017. Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024. Flextron: Many-in-one flexible large language model. In *International Conference on Machine Learning*, pages 5298–5311. PMLR.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

John Doe. Slimpajama-627b.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Xiou Ge, Ali Mousavi, Edouard Grave, Armand Joulin, Kun Qian, Benjamin Han, Mostafa Arefiyan, and Yunyao Li. 2024. Time sensitive knowledge editing through efficient finetuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 583–593, Bangkok, Thailand. Association for Computational Linguistics.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Efficient training of bert by progressively stacking. In *International conference on machine learning*, pages 2337–2346. PMLR.

Habib Hajimolahoseini, Walid Ahmed, and Yang Liu. 2023a. Training acceleration of low-rank decomposed networks using sequential freezing and rank quantization. *arXiv preprint arXiv:2309.03824*.

Habib Hajimolahoseini, Walid Ahmed, Austin Wen, and Yang Liu. 2024a. Accelerating the low-rank decomposed models. *arXiv preprint arXiv:2407.20266*.

Habib Hajimolahoseini, Omar Mohamed Awad, Walid Ahmed, Austin Wen, Saina Asani, Mohammad Hassanpour, Farnoosh Javadi, Mehdi Ahmadi, Foozhan Ataiefard, Kangling Liu, et al. 2023b. Swiftlearn: A data-efficient training method of deep learning models using importance sampling. *arXiv preprint arXiv:2311.15134*.

Habib Hajimolahoseini, Mohammad Hassanpour, Foozhan Ataiefard, Boxing Chen, and Yang Liu. 2024b. Single parent family: A spectrum of family members from a single pre-trained foundation model. *arXiv preprint arXiv:2406.19995*.

Nan He, Weichen Xiong, Hanwen Liu, Yi Liao, Lei Ding, Kai Zhang, Guohua Tang, Xiao Han, and Yang Wei. 2024a. SoftDedup: an efficient data reweighting method for speeding up language model pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4011–4022, Bangkok, Thailand. Association for Computational Linguistics.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024b. What matters in transformers? not all attention is needed. *arXiv preprint arXiv:2406.15786*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Farnoosh Javadi, Walid Ahmed, Habib Hajimolahoseini, Foozhan Ataiefard, Mohammad Hassanpour, Saina Asani, Austin Wen, Omar Mohamed Awad, Kangling Liu, and Yang Liu. 2023. Gqkva: Efficient pre-training of transformers by grouping queries, keys, and values. *arXiv preprint arXiv:2311.03426*.

Yano K., Takase S., Kobayashi S., Kiyono S., and J. Suzuki. 2025. Efficient construction of model family through progressive training using model expansion. In *Conference on Language Modeling*.

Yeachan Kim and SangKeun Lee. 2024. SparseFlow: Accelerating transformers by sparsifying information flows. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5937–5948, Bangkok, Thailand. Association for Computational Linguistics.

Changlin Li, Bohan Zhuang, Guangrun Wang, Xiaodan Liang, Xiaojun Chang, and Yi Yang. 2022. Automated progressive learning for efficient training of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12486–12496.

Bingli Liao and Danilo Vasconcellos Vargas. 2024. Beyond kv caching: Shared attention for efficient llms. *arXiv preprint arXiv:2407.12866*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Zeyu Liu, Souvik Kundu, Anni Li, Junrui Wan, Lianghao Jiang, and Peter Beerel. 2024b. AFLoRA: Adaptive freezing of low rank adaptation in parameter efficient fine-tuning of large models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–167, Bangkok, Thailand. Association for Computational Linguistics.

Yongyu Mu, Yuzhang Wu, Yuchun Fan, Chenglong Wang, Hengyu Li, Qiaozhi He, Murun Yang, Tong Xiao, and Jingbo Zhu. 2024. Cross-layer attention sharing for large language models. *arXiv preprint arXiv:2408.01890*.

Hossein Rajabzadeh, Aref Jafari, Aman Sharma, Benyamin Jami, Hyock Ju Kwon, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. 2024. Echoatt: Attend, copy, then adjust for more efficient large language models. *arXiv preprint arXiv:2409.14595*.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. 2024. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. 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.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. 2024. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*.

Haoyi Wu and Kewei Tu. 2024. Layer-condensed kv cache for efficient inference of large language models. In *Annual Meeting of the Association for Computational Linguistics*.

Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024. Survey on knowledge distillation for

large language models: methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology*.

Chengxuan Ying, Guolin Ke, Di He, and Tie-Yan Liu. 2021. Lazyformer: Self attention with lazy update. *arXiv preprint arXiv:2102.12702*.

Minjia Zhang and Yuxiong He. 2020. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in neural information processing systems*, 33:14011–14023.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024a. Tinyllama: An open-source small language mode. *arXiv preprint arXiv:2401.02385*.

Zhengxin Zhang, Dan Zhao, Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Qing Li, Yong Jiang, and Zhihao Jia. 2024b. Quantized side tuning: Fast and memory-efficient tuning of quantized large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–17, Bangkok, Thailand. Association for Computational Linguistics.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11641–11661.