LESA: Learnable LLM Layer Scaling-Up

Anonymous ACL submission

Abstract

Training Large Language Models (LLMs) from scratch requires immense computational resources, making it prohibitively expensive. Model scaling-up offers a promising solution by leveraging the parameters of smaller models to create larger ones. However, existing depth scaling-up methods rely on empirical heuristic rules for layer duplication, which result in poorer initialization and slower convergence during continual pre-training. We propose **LESA**, a novel learnable method for depth scaling-up. By concatenating parameters from each layer and applying Singular Value Decomposition, we uncover latent patterns between layers, suggesting that inter-layer parameters can be learned. LESA uses a neural network to predict the parameters inserted between adjacent layers, enabling better initialization and faster training. Experiments show that LESA outperforms existing baselines, achieving superior performance with less than half the computational cost during continual pre-training. Extensive analyses demonstrate its effectiveness across different model sizes and tasks.

1 Introduction

Recent advancements in Natural Language Processing (NLP) have been largely driven by Transformer-based architectures (Vaswani et al., 2017), with Large Language Models (LLMs) demonstrating exceptional capabilities in addressing a wide range of complex tasks (Brown et al., 2020; Achiam et al., 2023; Bai et al., 2023; Touvron et al., 2023a; Yang et al., 2024a; AI@Meta, 2024; Jiang et al., 2023; Almazrouei et al., 2023; Bi et al., 2024). As the parameter size continues to grow, in accordance with scaling laws (Kaplan et al., 2020), the computational resources required to train LLMs from scratch have become increasingly prohibitive, demanding millions of GPU hours and significant energy consumption. This immense resource demand largely arises from the need to randomly reinitialize model parameters, preventing the transfer of ability from existing LLMs.

To address this limitation, a common approach is model scaling-up, which leverages the parameters of smaller models to construct larger ones, either for immediate deployment or as a better initial checkpoint for more effective further continual pre-training. Existing model scaling-up methods can be divided into width scaling-up and depth scaling-up. Width scaling-up (Chen et al., 2015, 2021a; Wang et al., 2023; Samragh et al., 2024) primarily involves expanding matrix dimensions, rather than increasing the number of layers [1]. In contrast, depth scaling-up involves repurposing trained Transformer blocks from a smaller model to build a larger one with additional layers (Wu et al., 2024; Kim et al., 2023; Gong et al., 2019; Pan et al., 2024; Agarwal et al., 2024; Parmar et al., 2024). This strategy is widely applicable to modern LLMs based on the Transformer architecture, preserving the internal structure, such as matrix sizes. It is also compatible with existing parallel training frameworks, better preserving the model's knowledge, contributing to its increasing popularity in recent model scaling-up approaches.

However, current depth scaling-up methods rely on heuristic rules, typically duplicating one or more blocks before integrating them into the model. These approaches overlook parameter change patterns between layers, limiting the model's ability to specialize each layer effectively. As a result, newly upscaled layers replicate the previous ones, neglecting layer-specific specialization (Voita et al., 2019b,a). This not only leads to suboptimal model initialization performance but also prevents the model from fully utilizing its expanded capacity. By treating all layers equally, these methods fail to capture the nuanced relationships between layers,

[1] A "layer" refers to a Transformer block for simplicity.

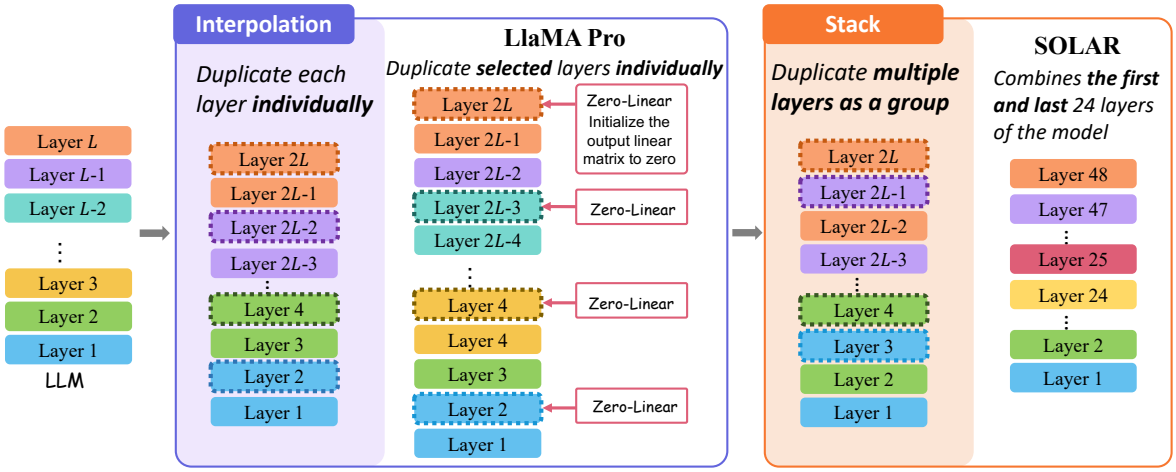

Figure 1: Existing depth scaling-up methods can be categorized into two types: "Interpolation" and "Stack". LLaMA Pro and SOLAR can be seen as specific examples of these two types. Layers with the same color represent identical parameters, and the dashed boxes indicate those obtained through duplication.

causing slower convergence during training and yielding less effective models.

In this paper, we propose a novel approach for depth scaling-up called **LESA** (**LE**arnable LLM Layer **S**c**A**ling-Up). We are the first to observe that by concatenating the parameters of each Transformer block and applying Singular Value Decomposition (SVD), patterns such as continuity can be identified between layers. Based on this observation, it is hypothesized that latent patterns exist between Transformer layers in a well-trained LLM, suggesting that model parameters can be learned across layers. To predict these parameters, we propose training a neural network. Once trained, the network can generate intermediate layers between adjacent layers, insert them into the model for depth scaling-up, and serve as a better initialization checkpoint, enabling faster convergence during continual pre-training. Our key contributions are summarized as:

- We first observe, through SVD, latent patterns such as continuity between Transformer layers, suggesting that inter-layer parameters can potentially be learned.
- We introduce **LESA**, which predicts intermediate layer parameters from adjacent layers for depth scaling-up. Experiments show that **LESA** outperforms existing baselines, with better model initialization and faster convergence during continual pre-training.
- Extensive experiments confirm that **LESA** works across various model sizes and families, including domain-specific tasks like code-related tasks. We also perform ablation studies to explore different method configurations.

2 Related Works

2.1 Model Scaling-up

Model scaling-up can be broadly categorized into width and depth scaling-up. Width scaling-up increases the matrix size while ensuring that the output of a layer or consecutive layers remains consistent with the output of the original network before expansion. Net2Net (Chen et al., 2015) is one of the first to transfer parameters from a smaller model to initialize a larger one using function-preserving transformations. bert2BERT (Chen et al., 2021a) extends this approach to Transformer-based models. LiGO (Wang et al., 2023) learns a linear mapping to initialize larger models. HyperCloning (Samragh et al., 2024) expands LLM to fit a larger model with more hidden dimensions. However, while these methods increase matrix size, they are less compatible with parallel training frameworks, which are better suited for depth scaling-up. Moreover, depth scaling-up better preserves the model's knowledge.

Current depth scaling-up methods expand the model by duplicating and adding layers based on heuristic rules, which can be broadly categorized into "Interpolation" and "Stack" (Pan et al., 2024), as shown in Figure 1. Interpolation involves adding a copy of each layer after the original, while Stack treats consecutive layers as a group and duplicates them together. Recent popular methods like LLaMA Pro (Wu et al., 2024) and SOLAR (Kim

et al., 2023) can be seen as special cases of these two types. LLaMA Pro copies only a selected few layers, while SOLAR duplicates the first 24 and the last 24 layers of a previous 32-layer model and combines them. However, these methods are based on heuristic rules, which hinder layer specialization, leading to suboptimal performance and limiting the model's potential.

2.2 Progressive Training

Progressive training involves gradually transitioning from simpler, smaller models to more complex, larger ones (Chang et al., 2017; Wen et al., 2020; Dong et al., 2020; Wei et al., 2016; Fayek et al., 2020). It is often combined with model scaling-up, where the model size is progressively increased during training. Prior to the era of LLMs, many methods (Chen et al., 2021a; Gu et al., 2020; Wang et al., 2023; Yang et al., 2020; Yao et al., 2023) are developed to train smaller models, such as BERT (Devlin et al., 2018). In recent years, LLaMA Pro (Wu et al., 2024) and Apollo (Pan et al., 2024) have applied progressive learning and model scaling-up strategies to train LLMs. YODA (Lu et al., 2024) introduces a novel teacher-student progressive learning framework that enhances model fine-tuning by emulating the teacher-student educational process. Du et al. offer a comprehensive evaluation and empirical guidelines for progressive learning and model scaling-up.

3 Method

This section discusses the patterns observed between model layers through SVD analysis of the model's parameters. Based on these patterns, we hypothesize that there are underlying patterns in the trained model that can be learned by a neural network. We then use this trained network to predict intermediate layers that can be inserted between adjacent layers for depth scaling-up.

3.1 SVD-Based Layer Pattern

Inspired by recent work using SVD for LLM compression or merging (Wang et al., 2024c; Stoica et al., 2024; Wang et al., 2024a), it is realized that SVD can map the model's parameters into one space for analysis. Specifically, assume we have weight matrices $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_L$ from L layers of an LLM, where $\mathcal{W}_i \in \mathbb{R}^{d1 \times d2}$ represents a matrix from each Transformer block, such as the up-projection matrix in MLP, Query matrix in self-attention. These L matrices can be concatenated horizontally into a single matrix, denoted as denoted as $\mathcal{W} \in \mathbb{R}^{d_1 \times Ld_2}$. SVD can be used to decompose this matrix into three components: U, Σ and V^T.

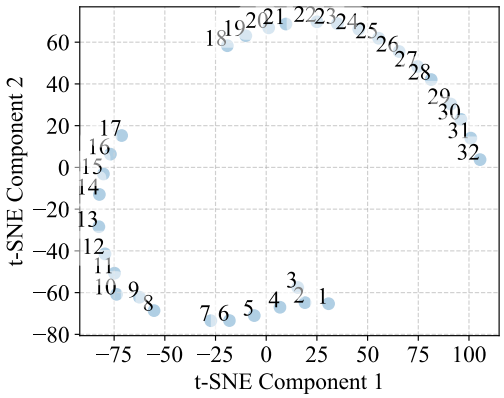

Figure 2: The inter-layer continuity pattern exhibited by the gate_proj matrix of Llama3-8B in the SVD space. The numbers represent the layer indices.

According to SVD, Σ is a diagonal matrix of size $d_1 \times Ld_2$, containing the singular values of $\mathcal{W}$. U is a unitary matrix that spans a set of standard orthogonal bases. If we treat Σ as a scaling transformation on each orthogonal basis in U, then $U\Sigma$ forms a new set of orthogonal bases. For the i-th layer's $\mathcal{W}_i$, it can be recovered as $\mathcal{W}_i = U\Sigma V_i$, where $V_i = V^T_{:,(i-1)*d2:i*d2} \in \mathbb{R}^{d_1 \times d_2}$. This means that the parameter $\mathcal{W}_i$ of each layer is a linear combination of the orthogonal bases from $U\Sigma$, with V_i representing the coefficients of this combination. By projecting the parameters of each layer into the space spanned by $U\Sigma$, we can analyze the patterns in the coefficients V_i for the i-th layer.

Since larger singular values correspond to eigenvectors that capture more information about the matrix, we select the eigenvector corresponding to the largest singular value (top-1) from each layer's V_i for visualization. We use t-SNE (Van der Maaten and Hinton, 2008) to reduce V_i to two dimensions. The visualization results of the gate-projection in the MLP of Llama3-8B (AI@Meta, 2024) are presented in Figure 2, where we observe a clear continuity in the distribution of these V_i. This continuity pattern, derived from the top-1 singular value of the gate-projection using t-SNE, is also present in Llama2 (Touvron et al., 2023b), Llama3 (AI@Meta, 2024), and Qwen2 (qwe, 2024). This suggests that the model's parameters may exhibit unique inter-layer patterns. More visualization results can be found in Appendix B.

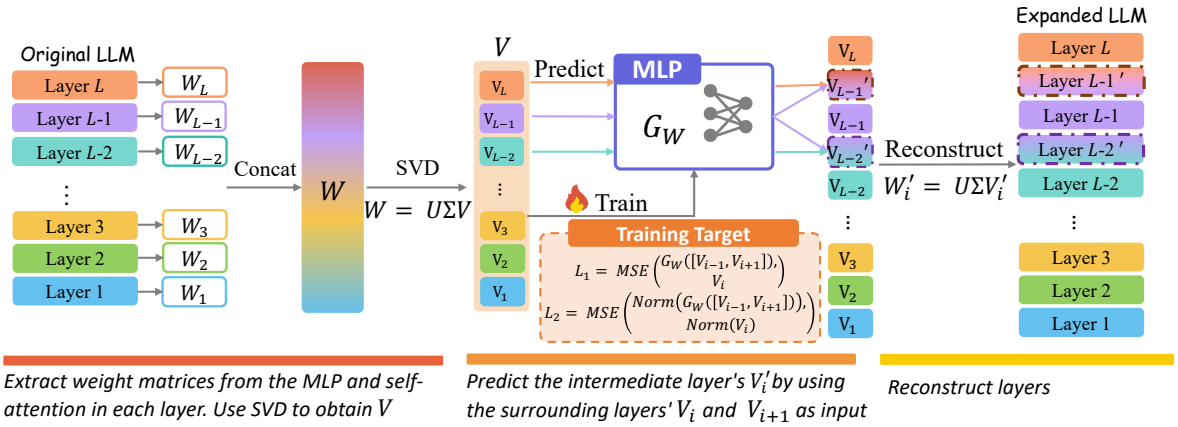

Figure 3: Overview of the proposed **LESA** . We first extract the weight matrices from the MLP and self-attention layers. Next, we apply SVD and train a neural network to predict the intermediate layers. Finally, we reconstruct the expanded LLM.

Although this continuity is currently only observed in the gate projection and not in other parameters such as the up-projection or down-projection in the MLP through our t-SNE visualization, this may be due to the limitations of our analysis method. An intuitive approach, therefore, is to use a neural network to learn these potential patterns.

3.2 Learnable LLM Layer Scaling-Up

Inspired by the aforementioned SVD-Based Layer Patterns, we hypothesize that there may be inter-layer patterns in the parameters. However, these patterns might not be easily observed using simple visualization techniques or fitted with specific distributions, such as Gaussian mixtures. Therefore, a direct approach is to learn these patterns through a neural network.

We present our method in Figure 3. After obtaining V_i as described in Section 3.1, we train an MLP $\mathcal{G}_\mathcal{W}$ to learn the patterns. Our training objective is to enable the MLP to predict an intermediate layer given any two layers that are one layer apart.

Formally, for a weight matrix $\mathcal{W} \in \left\{ \begin{array}{l} \text{q\_proj, k\_proj, v\_proj,} \\ \text{o\_proj, up\_proj, down\_proj, gate\_proj} \end{array} \right\}$, we use SVD to obtain V_i following Section 3.1. We then train an MLP $\mathcal{G}_\mathcal{W}$ specific to $\mathcal{W}$ with the objective of predicting V_i by using the concatenation of V_{i-1} and V_{i+1} as input. We optimize $\mathcal{G}_\mathcal{W}$ using MSELoss (Mean Squared Error Loss):

$$\mathcal{L}_1 = MSE(\mathcal{G}_\mathcal{W}([V_{i-1}, V_{i+1}]), V_i) \quad (1)$$

whose goal is to enable $\mathcal{G}_\mathcal{W}$ to predict accurately.

In subsequent experiments, we find that directly training with $\mathcal{L}_1$ will result in the norm of the predicted $\mathcal{G}_\mathcal{W}([V_{i-1}, V_{i+1}])$ approaching zero, meaning that the predicted $V_i^{'}$ parameters are close to zero, which leads to parameter degradation. To address this issue, we add a norm loss:

$$\mathcal{L}_2 = MSE(Norm(\mathcal{G}_\mathcal{W}([V_{i-1}, V_{i+1}])), Norm(V_i)) \quad (2)$$

where the $Norm$ represents the L2 norm, and $\mathcal{L}_2$ aims to ensure that the norm of the model's predicted $V_i^{'}$ is close to that of V_i. Thus, the final loss for training $\mathcal{G}_\mathcal{W}$ is:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_2 \quad (3)$$

where λ is a hyper-parameter.

Once trained, $\mathcal{G}_\mathcal{W}$ can predict the parameters of an intermediate layer based on its surrounding ones. Thus, for adjacent V_i and V_{i+1}, we use $\mathcal{G}_\mathcal{W}$ to predict the intermediate layer $V_i^{'}$ to insert between them. We then reconstruct $V_i^{'}$ using the $U\Sigma$ decomposition from the previous step, forming the predicted matrix $\mathcal{W}^{'}$, and insert it between the layers to expand the LLM.

4 Main Experiments

4.1 Settings

4.1.1 LESA Settings

We conduct experiments on the Llama3-8B model, which has 32 layers. To construct the training data for $\mathcal{G}_\mathcal{W}$, we use consecutive triplets of layers, namely (1, 2, 3), (2, 3, 4), (3, 4, 5), ..., (30, 31, 32), resulting in 30 samples. We define $\mathcal{G}_\mathcal{W}$ as a three-layer MLP with a ReLU (Agarap, 2018) activation function, where the hidden dimension is 256. $\mathcal{G}_\mathcal{W}$ is trained for 5 epochs on these samples

using the AdamW (Loshchilov, 2017) optimizer with a learning rate of 1e-3. The λ is set to 5e-5. To compare with baselines (Wu et al., 2024; Kim et al., 2023), we use **LESA** to scale up Llama3-8B to 48 layers by inserting an intermediate layer between each pair of adjacent layers in the original 15th to 31st layers. The expanded models all have 11.5 billion parameters.

4.1.2 Continual Training

For the models expanded using **LESA** and baseline methods, we continue pre-training with Wikipedia data from November 2024, which is released after the training of Llama3-8B and has not been used in its original training. We use the Llama-Factory (Zheng et al., 2024) training framework, with a cutoff length of 4096, a warmup ratio of 0.1, and a cosine learning rate scheduler. The optimizer is AdamW with a learning rate of 5e-5. The batch size per GPU is 2, with 4 gradient accumulation steps. For LESA and LLaMA Pro, we only train the newly expanded layers, freezing the other layers. Following the original setting, we perform full parameter fine-tuning for SOLAR.

For the Supervised Fine-Tuning (SFT) stage, we use Alpaca-GPT4 (Peng et al., 2023) for training, following SOLAR. The hyper-parameters are the same as those in the continual pre-training, except that we perform full parameter fine-tuning without freezing any layers for all models. All experiments are conducted on a server with 8 Nvidia A100 80GB GPUs.

4.2 Benchmarks

For the continual pre-training models, since they lack instruction-following capabilities, we use the OpenCompass framework (Contributors, 2023) with the PPL (perplexity) [2] mode for evaluation, focusing on five areas: Reasoning, Language, Knowledge, Examination, and Understanding, with selected benchmarks for each category. **Reasoning**: CMNLI (Xu et al., 2020), HellaSwag (HeSw) (Zellers et al., 2019), PIQA (Bisk et al., 2019). **Language**: CHID (Zheng et al., 2019), WinoGrande (Wino) (Sakaguchi et al., 2019). **Knowledge**: CommonSenseQA (CSQA) (Talmor et al., 2018), BoolQ (Clark et al., 2019). **Examination**: MMLU (Hendrycks et al., 2021), CMMLU (Li et al., 2023). **Understanding**: Race-High/Middle (H/M) (Lai et al., 2017). Evaluations use OpenCompass official scripts in zero-shot or few-shot settings. Scores are computed by OpenCompass, with higher values indicating better performance. We also evaluate the trained models' perplexity on 500 unseen Wikipedia plain sentences.

For the models after SFT, which have gained instruction-following capabilities, we use the generation mode of OpenCompass for evaluation. We conduct evaluations on ARC (Clark et al., 2018), TruthfulQA (Lin et al., 2021), GSM8K (Cobbe et al., 2021), HellaSwag (Zellers et al., 2019), and MMLU (Hendrycks et al., 2021).

[2] https://opencompass.readthedocs.io/en/latest/get_started/faq.html

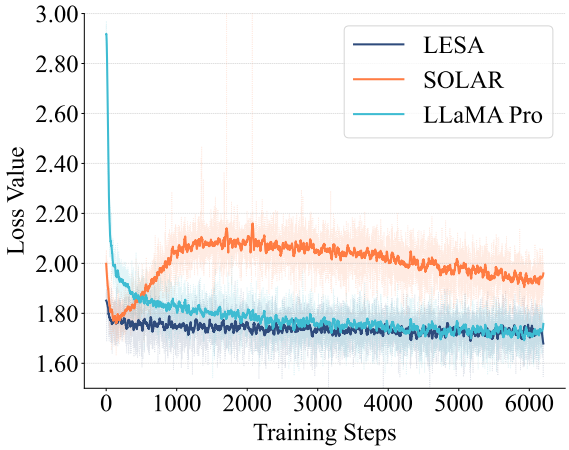

Figure 4: The continual pre-training loss curves of models expanded by different methods. **LESA** starts with a lower initial loss, indicating a better initialization. It stabilizes after 2k steps, reaching the same convergence level as LLaMA Pro after 5k steps, and converges much faster than SOLAR, achieving the same loss with less than half the training cost.

Model	Pro	SOLAR	LESA
Time	56.4h(124%)	75.6h(166%)	**45.6h**

Table 1: Training time of continual pre-training. When trained on the same dataset, the baselines require 124% and 166% of the training time compared to our method.

4.3 Results

We first present the training loss curves of the three models in Figure 4. From the figure, we observe that our method starts with a lower initial loss compared to the baselines, indicating a better initialization checkpoint. Throughout training, our model's loss consistently remains the lowest. SOLAR even fails to converge to a low loss level even after training on the dataset. Although LLaMA Pro's loss approaches ours after 5k steps, by the end, the

Model	PPL	Average	Reasoning			Language		Knowledge		Examination		Understanding	
			CMNLI	HeSw	PIQA	CHID	Wino	CSQA	BoolQ	MMLU	CMMLU	Race_H	Race_M
Pro-3k	6.06	60.89(-3.41)	**32.99**	69.14	79.21	67.33	55.79	69.19	68.10	45.53	49.66	65.32	67.55
Pro-6k	5.44	62.67(-1.63)	32.97	70.15	78.94	69.80	55.09	69.04	66.79	64.31	50.30	62.66	69.36
SOLAR-3k	9.82	45.34(-18.96)	32.97	61.11	73.23	51.49	53.68	54.55	52.94	34.98	28.97	27.36	27.44
SOLAR-6k	8.09	47.86(-16.44)	32.98	62.36	74.65	48.02	54.38	59.54	61.53	43.74	28.68	31.76	28.83
LESA-3k	5.27	64.11(-0.19)	**32.99**	71.18	79.65	72.77	**57.89**	**69.78**	**70.46**	66.66	50.91	65.32	67.55
LESA-6k	**5.13**	**64.30**	**32.99**	**71.51**	**79.92**	**73.30**	57.54	69.21	69.94	**66.67**	**51.00**	**65.72**	**69.50**

Table 2: We evaluate the performance of models after expanding Llama3-8B from 32 layers to 48 layers (11.5B parameters) using different baseline methods, followed by continual pre-training. Pro (LLaMA Pro (Wu et al., 2024)) and SOLAR (Kim et al., 2023) are two strong baselines for model depth scaling-up. We evaluate the model performance at two stages: after training with half the data (3k steps) and after training with the full data (6k steps).

Model	Average	ARC-e	ARC-c	TruthfulQA	GSM8K	HellaSwag	MMLU
Pro-SFT	24.38(77%)	28.92	23.73	21.91	21.95	25.44	**24.33**
SOLAR-SFT	26.47(84%)	37.10	24.25	19.34	33.45	25.16	19.52
LESA-SFT	**31.57(100%)**	**42.86**	**32.54**	**22.28**	**37.14**	**32.09**	22.49

Table 3: After continual pre-training and subsequent SFT, the model expanded with LESA still achieves better task performance, with baselines scoring less than 85% of our model's average score.

model expanded using our method still has the lowest loss. Additionally, our method's loss stabilizes after 2k steps, while LLaMA Pro reaches a similar convergence level only after 5k steps. This demonstrates that models expanded using LESA achieve the same loss convergence with less than half the training cost.

We list the time taken to train on the full dataset in Table 1 and find that the time taken by **LESA** is significantly shorter. It is worth noting that the training of $\mathcal{G}_\mathcal{W}$ in **LESA** is very fast, taking less than 5 minutes, making its cost nearly negligible compared to the overhead of continual pre-training.

For model performance after continual pre-training, we present the results on various benchmarks in Table 2. It can be inferred that the performance of models expanded with **LESA** consistently outperforms the baselines in all categories. Specifically, LESA-6k (6k steps) achieves the highest performance across all tasks and PPL. Even with only half of the data used for continual pre-training (3k steps), the models expanded using LESA outperform the baselines trained on the full dataset (6k steps). We also present the results in Table 3 for models trained on the full dataset and then fine-tuned with SFT, showing performance across different tasks. The results still indicate that the models expanded using LESA achieve the best performance.

Model	CSQA	BoolQ	TriviaQA	NQ
Pro-3k	69.19	68.10	62.50	24.82
Pro-6k	69.04	66.79	63.68	26.73
SOLAR-3k	54.55	52.94	43.06	13.57
SOLAR-6k	59.54	61.53	47.72	16.40
LESA-3k	**69.78**	**70.46**	**67.15**	23.30
LESA-6k	69.21	69.94	67.05	**26.76**

Table 4: The scores of different models on knowledge-related tasks after continual pre-training. LESA consistently performs better overall.

The above analysis proves that **LESA** effectively inherits the original model's parameters, enabling better initialization, faster continual training, and enhanced model performance.

4.4 Evaluation on Knowledge-Related Tasks

Previous studies, such as LLaMA Pro, highlight that a key advantage of model expansion is the ability to inherit knowledge from the original model. We focus on evaluating performance in knowledge-related tasks. In addition to the main results, we further evaluate performance on two additional knowledge tasks: TriviaQA (Joshi et al., 2017) and NQ (Kwiatkowski et al., 2019). The results in Table 4 show that **LESA** outperforms previous approaches on all knowledge tasks.

5 Ablation Study

5.1 Evaluation across Different Model Families

We also aim to explore whether LESA is effective across different model sizes and families. Specifically, we select several current mainstream model families Llama3, Qwen2.5, Mistral (Mistral@AI, 2025) and use LESA to expand the final layers of the models, increasing their layer count by 1.5x of the original. We use SOLAR initialization as the baseline. Since their method only applies to 32-layer models by concatenating the first 24 and last 24 layers, we adapt it for models with different layer counts. We concatenate the first and last n layers to create a model with 1.5 times the original layers and measure initialization performance using PPL. The results are shown in Table 5.

Model	Original	+LESA	+SOLAR
Llama3-8B	5.20	6.35	7.81
Llama3-70B	1.98	2.62	4.21
Qwen2.5-1.5B	9.30	10.52	11.75
Qwen2.5-7B	6.03	7.04	7.99
Qwen2.5-32B	3.78	5.67	INF
Mistral-Small-24B	4.43	5.17	6.51

Table 5: PPL of LESA and SOLAR during 1.5x layer expansion initialization for different models, along with the PPL of the original models.

The results show that LESA outperforms SOLAR in initialization performance. Unlike SOLAR, which experiences a PPL explosion on Qwen2.5-32B, LESA remains stable, highlighting the superiority of LESA's predicted parameters over SOLAR's heuristic-based expansion.

5.2 Analysis of $\mathcal{G}_{\mathcal{W}}$'s Ability

We investigate whether $\mathcal{G}_{\mathcal{W}}$ can predict intermediate layers between adjacent layers accurately, demonstrating this through loss changes.

Due to the limited number of samples available for training $\mathcal{G}_{\mathcal{W}}$ on individual LLM layers, which makes it difficult to separate a test set and increases the risk of overfitting, we select several models: Llama3-8B, and fine-tuned versions of it, including Llama3-8B-Lexi-Uncensored (Orenguteng, 2024), Meta-Llama3-8B-Instruct, Llama-3-Smaug-8B (Pal et al., 2024), and Llama3-8B-Chinese-Chat (Wang et al., 2024b). Following the procedure outlined in Section 4.1.1, we sequentially select three consecutive layers as samples, resulting in a total of 150 samples. We use 120 samples for training and 30 for testing. The hyperparameters for training are set consistent with those used in the main experiment.

Matrix	Random Loss	Training Loss	Test Loss
down_proj	5.7	0.0005	0.0004
up_proj	0.055	0.015	0.015
gate_proj	0.056	0.015	0.015
q_proj	0.153	0.016	0.016
v_proj	0.545	0.017	0.016
o_proj	0.147	0.016	0.015
k_proj	0.6	0.016	0.016

Table 6: Loss values for different matrices during training and testing. All values are multiplied by 10^4 for convenience.

Method	Pro	SOLAR	LESA
HumanEval	10.98	2.44	**25.00**
MBPP	21.69	13.93	**28.60**

Table 7: The results of Llama3-8B after expansion with different methods, pre-trained on the BigCode dataset, on two code benchmarks. The results show that LESA consistently performs better.

We present the loss values of $\mathcal{G}_{\mathcal{W}}$ on both the training and test sets after training in Table 6. For comparison, we also show the loss on the training set after random initialization. The results demonstrate that $\mathcal{G}_{\mathcal{W}}$ significantly reduces the loss on the training set after training, typically lowering it to below 10% of the random initialization loss. Moreover, the loss on the test set remains at the same level as the training set loss, indicating that $\mathcal{G}_{\mathcal{W}}$ effectively learns the underlying patterns of the model parameters.

5.3 Single-Domain Pre-training

In addition to general-domain pre-training experiments, we explore whether models expanded using our method show greater potential for continual pre-training in a single-domain setting. We conduct experiments in the code domain, using a subset of BigCode (Kocetkov et al., 2022), one of the largest code pre-training datasets, while keeping other settings unchanged. Each model is trained for 40-60 hours and then evaluated on the HumanEval (Chen et al., 2021b) and MBPP (Austin et al., 2021) benchmarks. Table 7 shows that after continual pre-training on the same code dataset, models expanded using our method outperform previous approaches, demonstrating its effectiveness in single-domain pre-training.

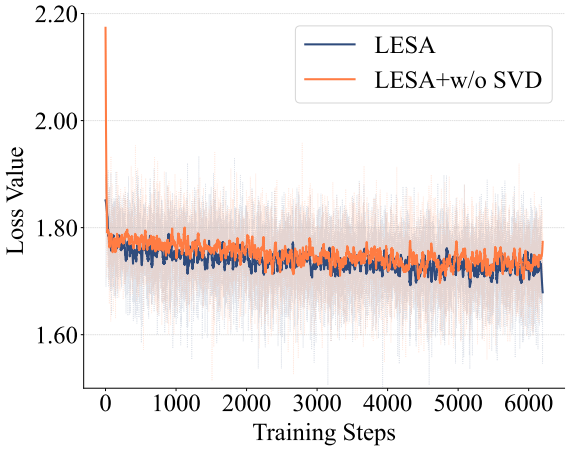

Figure 5: The continual pre-training loss curves without SVD, compared to the main experiment, show that with SVD, the model's initial loss and the final converged loss are both slightly lower.

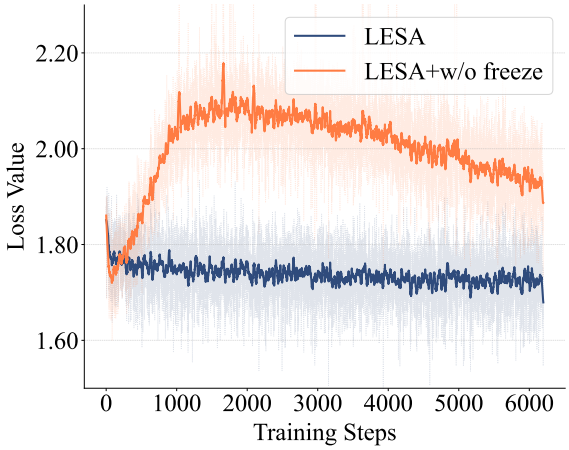

Figure 6: The training curves for LESA and LESA without freezing layers. When not freezing, the loss fluctuates and converges more slowly.

Model	PIQA	BoolQ	HeSw	Wino
LESA-6k	79.92	69.94	71.51	57.54
- SVD	79.54(-0.38)	68.81(-0.13)	70.44(-1.07)	57.29(-0.25)
Pro-6k	78.94	66.79	70.15	55.09

Table 8: Without SVD, performance on several tasks is lower than with SVD, but still surpasses LLaMA Pro.

5.4 Impact of SVD

We observe inter-layer patterns of matrices in the SVD space, as shown in Figure 2, which inspires us to train $\mathcal{G}_{\mathcal{W}}$ in the SVD space for prediction. We also explore whether $\mathcal{G}_{\mathcal{W}}$ can still predict effective matrices for layer expansion without SVD.

We conduct an ablation study where we remove the SVD decomposition step while keeping other aspects of the method unchanged. Instead, we directly input the matrices to train $\mathcal{G}_{\mathcal{W}}$, which predicts the parameters to be inserted between adjacent layers. We conduct experiments on Llama3-8B, expanding it to 48 layers and performing pre-training with the same data and hyper-parameters as in the main experiment. The loss curves with/without SVD are shown in Figure 5. Without SVD, the model performs worse, with higher loss in the early stages and an average loss of 0.03 higher than with SVD after 3k steps. Thus, the addition of SVD is beneficial. We evaluate the models on several tasks, as shown in Table 8. The results show that while the model expanded without SVD performs slightly worse, it still outperforms the LLaMA Pro baseline. This demonstrates the effectiveness of LESA, with SVD further enhancing performance.

5.5 Impact of Freezing Layers during Continual Pre-training

Following LLaMA Pro, we train only the newly expanded layers during continual pre-training. We also explore full parameter fine-tuning without freezing any layers. Compared to the main experiment, we directly fine-tune all parameters while keeping the training data and hyperparameters consistent. The loss curves are shown in Figure 6. The figure shows that without freezing layers, loss converges much slower, with fluctuations in the curve. This suggests that, similar to LLaMA Pro, freezing the original parameters is essential for faster and better loss convergence.

More experiments on hyper-parameter settings, loss design, and the effectiveness on MoE model can be found in Appendix A.

6 Conclusion

In this paper, we introduce **LESA** , a novel approach for depth scaling-up of LLMs that overcomes the limitations of current heuristic-based methods. Using SVD and a neural network, **LESA** predicts intermediate layer parameters, resulting in improved model initialization and faster convergence during continual pre-training. Extensive experiments show that **LESA** outperforms existing baselines, delivering superior performance with lower computational costs. Furthermore, **LESA** is effective across various model sizes, families, and domain-specific tasks, offering a promising solution for scaling LLMs efficiently. Our discovery of inter-layer patterns also provides new insights for future model design and training.

Limitations

This work does not yet consider scaling the model to sizes larger than three times the parameters. Based on current model design practices, when increasing the number of layers significantly, it is typically necessary to expand the matrix size of each layer as well, which requires width scaling-up. We plan to explore this in future work.

Although we have conducted a preliminary exploration of LESA on MoE model, the research is still limited by the challenges of constructing routers for the predicted layers and the current large size of MoE models. Further investigation into MoE models is needed, and we consider this as future work.

References

2024. Qwen2 technical report.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AF Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Naman Agarwal, Pranjal Awasthi, Satyen Kale, and Eric Zhao. 2024. Stacking as accelerated gradient descent. *arXiv preprint arXiv:2403.04978*.

AI@Meta. 2024. Llama 3 model card.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

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

Zouying Cao, Yifei Yang, and Hai Zhao. 2024. Head-wise shareable attention for large language models. *arXiv preprint arXiv:2402.11819*.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. 2017. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2021a. bert2bert: Towards reusable pretrained language models. *arXiv preprint arXiv:2110.07143*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021b. Evaluating large language models trained on code.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2015. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias

Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chengyu Dong, Liyuan Liu, Zichao Li, and Jingbo Shang. 2020. Towards adaptive residual network training: A neural-ode perspective. In *International conference on machine learning*, pages 2616–2626. PMLR.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. 2024. Stacking your transformers: A closer look at model growth for efficient llm pre-training. *arXiv preprint arXiv:2405.15319*.

Haytham M Fayek, Lawrence Cavedon, and Hong Ren Wu. 2020. Progressive learning: A deep learning framework for continual learning. *Neural Networks*, 128:345–357.

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.

Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2020. On the transformer growth for progressive bert training. *arXiv preprint arXiv:2010.12562*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. 2023. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas

Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Jianqiao Lu, Wanjun Zhong, Yufei Wang, Zhijiang Guo, Qi Zhu, Wenyong Huang, Yanlin Wang, Fei Mi, Baojun Wang, Yasheng Wang, et al. 2024. Yoda: Teacher-student progressive learning for language models. *arXiv preprint arXiv:2401.15670*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Mistral@AI. 2025. Mistral small 3.

Orenguteng. 2024. Llama-3-8b-lexi-uncensored.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

Yu Pan, Ye Yuan, Yichun Yin, Jiaxin Shi, Zenglin Xu, Ming Zhang, Lifeng Shang, Xin Jiang, and Qun Liu. 2024. Preparing lessons for progressive training on language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18860–18868.

Jupinder Parmar, Sanjev Satheesh, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Reuse, don't retrain: A recipe for continued pretraining of language models. *arXiv preprint arXiv:2407.07263*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Mohammad Samragh, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. 2024. Scaling smart: Accelerating large language model pre-training with small model initialization. *arXiv preprint arXiv:2409.12903*.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. 2024. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

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*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019a. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. *arXiv preprint arXiv:1909.01380*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019b. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. 2024a. Basis sharing: Cross-layer parameter sharing for large language model compression. *arXiv preprint arXiv:2410.03765*.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. 2023. Learning to grow pretrained models for efficient transformer training. *arXiv preprint arXiv:2303.00980*.

Shenzhi Wang, Yaowei Zheng, Guoyin Wang, Shiji Song, and Gao Huang. 2024b. Llama3-8b-chinese-chat (revision 6622a23).

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2024c. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. 2016. Network morphism. In *International conference on machine learning*, pages 564–572. PMLR.

Wei Wen, Feng Yan, Yiran Chen, and Hai Li. 2020. Autogrow: Automatic layer growing in deep convolutional networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 833–841.

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

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Cheng Yang, Shengnan Wang, Chao Yang, Yuechuan Li, Ru He, and Jingqiao Zhang. 2020. Progressively stacking 2.0: A multi-stage layerwise training method for bert training speedup. *arXiv preprint arXiv:2011.13635*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. 2023. Masked structural growth for 2x faster language model pre-training. *arXiv preprint arXiv:2305.02869*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. ChID: A large-scale Chinese IDiom dataset for cloze test. In *ACL*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*.

A More Experiments

In this section, we provide additional experiments and analyses on hyperparameter settings, loss design, and the effectiveness on the MoE model.

A.1 Impact of Layer Insertion Location

Previous studies (Yang et al., 2024b; Men et al., 2024; Cao et al., 2024) suggest that LLMs are generally less sensitive to layers near the output end, which can be modified. Therefore, our main experiment focuses on expanding layers closer to the output end. We also aim to explore the performance of our method when expanding layers near the input end. Building on the main experiment, we change the range of the expanded layers from the original 15th to 31st layers to the 1st to 17th layers. We then compare the PPL on Wikipedia for the models after initialization, without further training.

Layer Interval	**15-31**	**1-17**
PPL	6.35	57.32

Table 9: The model's initialization performance is better when layers are inserted at the output than at the input end.

The results in Table 9 show that expanding layers near the input end results in poorer initialization performance than expanding near the output. This suggests that our method is more effective when layers are inserted closer to the output, aligning with previous findings.

A.2 Ablation on Norm Loss

We investigate whether it is possible to train $\mathcal{G}_{\mathcal{W}}$ without adding the norm loss $\mathcal{L}_2$. Compared to the main experiment, we remove this loss and calculate the average norm of the matrices in the newly inserted layers predicted by $\mathcal{G}_{\mathcal{W}}$.

Model	Llama3-8B	+LESA	+LESA+w/o $\mathcal{L}_2$
down_proj	80.88	80.70	13.18
up_proj	81.88	81.57	10.26
gate_proj	104.96	105.39	13.34
q_proj	69.16	69.28	8.85
k_proj	52.44	51.01	7.32
v_proj	19.88	18.98	2.91
o_proj	40.25	40.49	5.29

Table 10: Without the norm loss $\mathcal{L}_2$, the norms of the matrices predicted by $\mathcal{G}_\mathcal{W}$ are very small, leading to parameter degradation.

As shown in Table 10, without $\mathcal{L}_2$, the predicted matrices have very small norms, causing their values to approach zero and leading to degeneration. However, with $\mathcal{L}_2$, the norms of the predicted matrices align with those of the original Llama3-8B matrices.

A.3 Hyper-parameter Impact on Model Initialization

In this section, we explore the impact of key hyper-parameters during the training of $\mathcal{G}_\mathcal{W}$. We find that the number of epochs and learning rate affect the initialization performance of the model obtained through layer expansion. We also conduct experiments on Llama3-8B, varying the learning rate and epochs while keeping other hyper-parameters consistent with the main experiment.

Learning Rate	Epoch	PPL
1e-3	5	6.35
1e-4	5	102.42
5e-4	5	6.82
1e-4	10	39182.51
5e-4	10	6.94

Table 11: Ablation study on the hyperparameters during the training of $\mathcal{W}_\mathcal{G}$.

The results in Table 11 show that adjusting the learning rate and epochs can sometimes cause the expanded model's PPL to explode during initialization. This may be due to the limited number of training samples generated from a single model, leading to training instability. However, after tuning the hyper-parameters a few times, we are able to achieve a good initialization performance, with PPL values typically ranging between 6 and 7.

Additionally, we find that the hidden-state size and the number of layers in $\mathcal{G}_\mathcal{W}$ have no significant impact on the performance of the expanded model. The loss's λ only affects the matrix norm, but has minimal effect on the model's performance. Adjusting λ to match the predicted matrix norm with that of the original model is sufficient.

A.4 Effectiveness on MoE Model

Recently, LLMs based on the Mixture-of-Experts (MoE) architecture have become increasingly popular. In this section, we explore the effectiveness of **LESA** on such models. Due to the large size of current MoE models, such as DeepSeek-R1 with 671B parameters (DeepSeek-AI et al., 2025), which cannot be loaded onto our server, we conduct experiments on the smaller LLaMA-MoE-3.0B (Zhu et al., 2024), which has 32 layers.

We use LESA to expand the model to 48 layers. However, a unique aspect of MoE models is that each layer has an MLP router, and we have not yet devised a method to generate routers for the newly added layers, since the router is highly dependent on the performance of each expert. Our current approach is to replicate the previous layer's router for the newly expanded layer. We use SOLAR as the baseline and then evaluate the PPL of the expanded model after initialization. The results are shown in Table 12.

Model	+LLaMA-MoE-3.0B	+LESA	+SOLAR
PPL	7.70	1923.14	76.50

Table 12: The MoE model's initialization performance on PPL with different scaling-up methods.

The results show that LESA experiences a significant increase in PPL, which we attribute to the mismatch between the router and the expanded parameters. We will continue investigating this issue in future work. Meanwhile, SOLAR also performs poorly, increasing PPL by 10 times. This suggests that scaling-up methods for MoE models require further research.

B SVD-Based Patterns

We present the t-SNE visualizations of the top 1 singular values corresponding to the vectors of V, obtained after applying SVD decomposition to the matrices in the MLP and self-attention of different models, in Figure 7 and Figure 8, respectively.

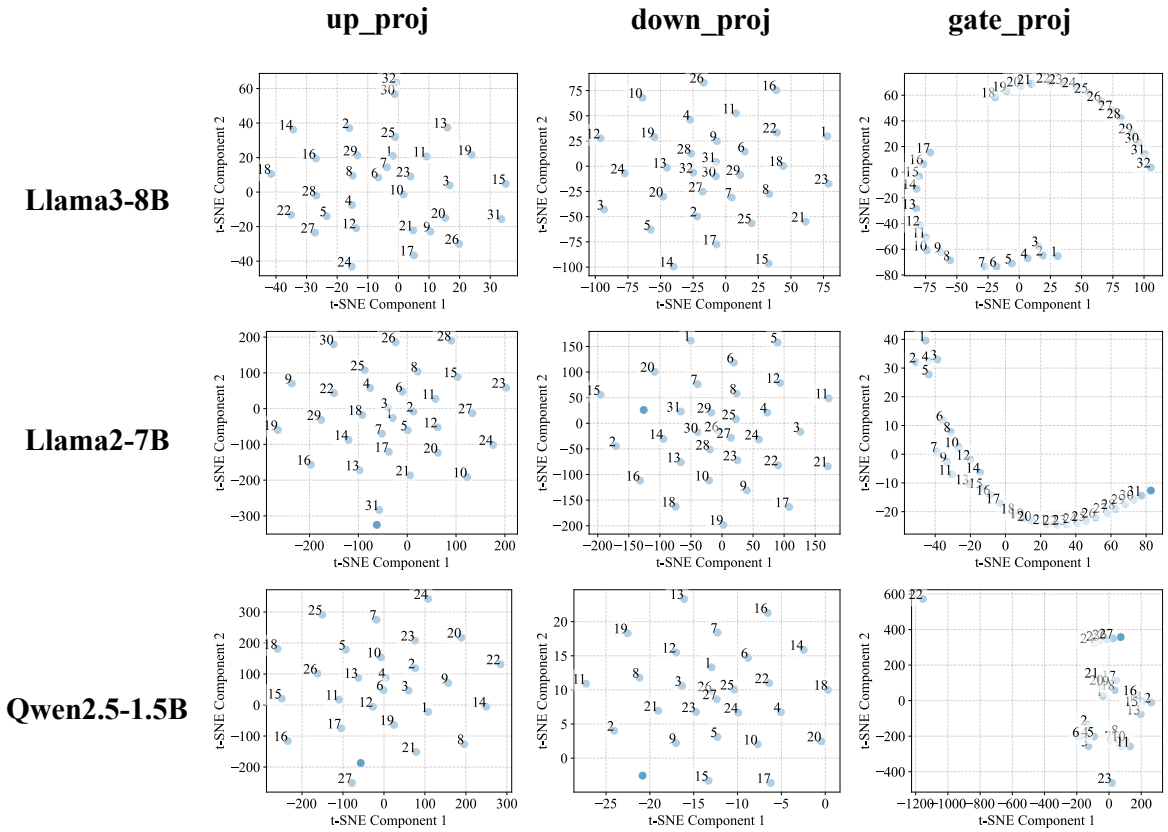

Figure 7: The gate_proj parameter matrices in the MLP of different models exhibit clear patterns of continuity or clustering. This suggests that after applying SVD, the model's parameters may be learnable. The parameter distributions of other matrices appear more uniform in our visualizations.

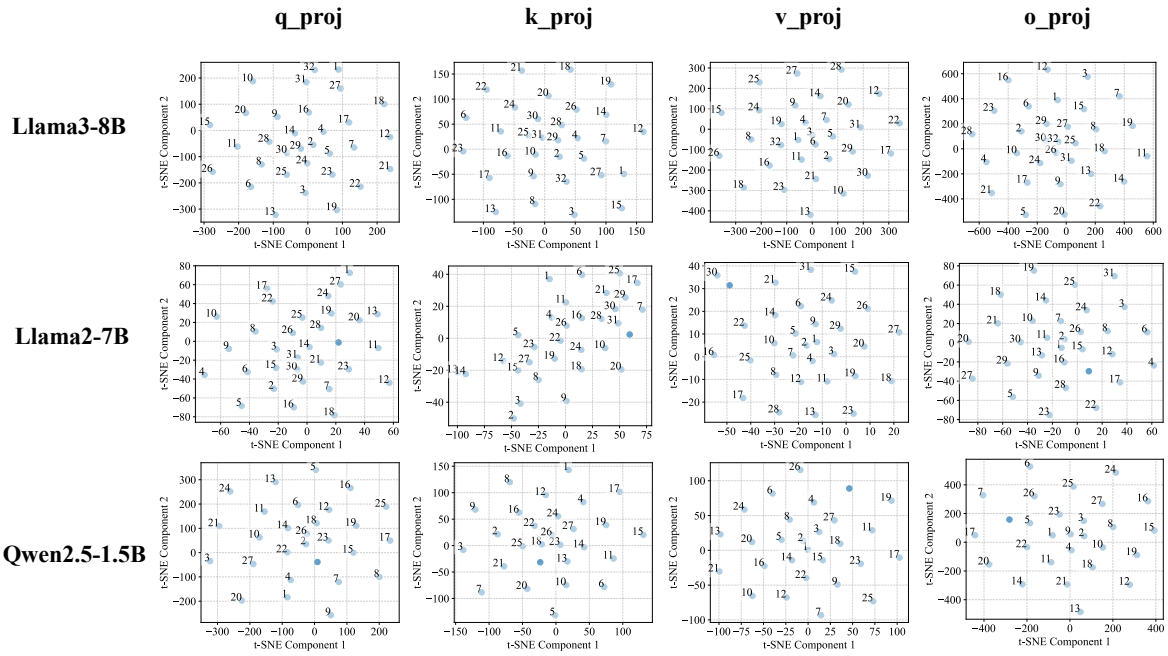

Figure 8: The parameter distributions of the matrices in the self-attention layers across different models appear relatively uniform in our visualizations.