LEMON: Reviving Stronger and Smaller LMs from Larger LMs with Linear Parameter Fusion

Yilong Chen[1,2], **Junyuan Shang**[3‡], **Zhenyu Zhang**[3], **Shiyao Cui**[1], **Tingwen Liu**[1,2†], **Shuohuan Wang**[3], **Yu Sun**[3], **Hua Wu**[3]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] Baidu Inc.

{chenyilong, cuishiyao, liutingwen}@iie.ac.cn
{shangjunyuan, zhangzhenyu07, wangshuohuan, sunyu02}@baidu.com

Abstract

In the new era of language models, small models (with billions of parameter sizes) are receiving increasing attention due to their flexibility and cost-effectiveness in deployment. However, limited by the model size, the performance of small models trained from scratch may often be unsatisfactory. Learning a stronger and smaller model with the help of larger models is an intuitive idea. Inspired by the observing modular structures in preliminary analysis, we propose LEMON to learn competent initial points for smaller models by fusing parameters from larger models, thereby laying a solid foundation for subsequent training. Specifically, the parameter fusion process involves two operators for layer and dimension, respectively, and we also introduce controllable receptive fields to model the prior parameter characteristics. In this way, the larger model could be transformed into any specific smaller scale and architecture. Starting from LLaMA 2-7B, we revive two stronger and smaller models with 1.3B and 2.7B. Experimental results demonstrate that the fusion-based method exhibits flexibility and outperforms a series of competitive baselines in terms of both effectiveness and efficiency.

1 Introduction

Transformer-based language models (LMs) shine in various natural language tasks due to their powerful understanding and generation capabilities (Anthropic, 2023; OpenAI, 2023; Touvron et al., 2023). Considering that the training and deployment of large-scale LMs require a large amount of computing resources, small LMs are more cost-effective in actual production environments (Gunasekar et al., 2023; Li et al., 2023). However, in the pre-training stage, it has been proven that there is a certain scaling law between model size and performance (Kaplan et al., 2020; Hoffmann et al., 2022). How to break through this limitation to train a stronger and smaller model with the help of large models has recently attracted the attention of researchers (Ma et al., 2023; Xia et al., 2023; Xu et al., 2023).

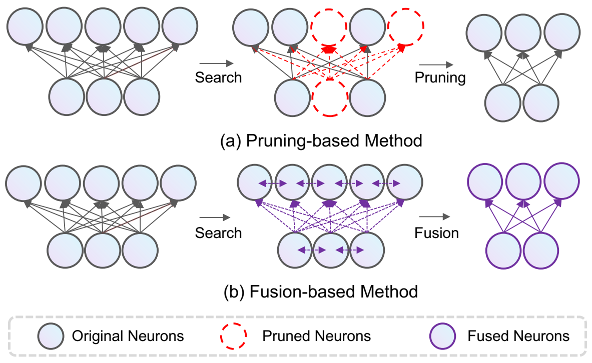

Figure 1: (a) The pruning-based method searches for important neurons and removes unimportant neurons, while (b) our fusion-based method utilizes all neurons to preserve parameter functionality.

Inspired by the idea of function-reserving transformation in efficient training (Chen et al., 2015), there are some model compression efforts that utilize layer dropping, structured pruning, or weight selection to preserve the parameter functionality of large models as much as possible, so as to accelerate pre-training and improve the final performance of small models (Zhang and He, 2020a; Xia et al., 2022; Xu et al., 2023). However, since there are tens or even hundreds of billions of parameters in modern LMs, searching for the essential parameter combinations requires significant computing resources, while pruning inappropriate parameters inevitably leads to performance degradation in downstream tasks (Frantar and Alistarh, 2023). In this case, how to construct an appropriate initial point with capabilities close to large LMs at the lowest possible cost remains an open problem.

Given the limited understanding of parameter characteristics and roles in modern large LMs, we perform an empirical analysis from the perspectives

†Corresponding author. ‡ Project lead.

of representation similarity, and observe that there are some modular structures with high internal similarity. In particular, there is a high similarity in the parameters of adjacent layers, while the similarity of distant layers to a lesser extent, which coincides with the sparsity found in previous studies (Qin et al., 2022; Zhang et al., 2023). Nonetheless, for a parameter module, each part has its own unique role, and thus no model parameter should be arbitrarily discarded. Based on the observation, we believe that the functionality of a module should be reconstructed by selectively fusing corresponding parameters during the initial point construction phase, as illustrated in Figure 1.

In this paper, we propose LEMON to construct compact initial points for small models through linear fusion of large model parameters. Specifically, we first identify the importance of parameters in larger models with fusion operators, and then compress them into new parameters for the smaller model, maintaining the same functionality of related parameters before and after transformation. To support mapping a larger model into any specific architecture, we decompose the fusion process into layer and dimension operations. The layer operator establishes layer-to-layer correspondence between larger and smaller models, while the dimension operator aims to learn the fusion of different dimensions within each parameter matrix. Recalling the modular structures among layers, we further factorize the fusion operators into Kronecker products (Schacke, 2004) with controllable receptive fields, instead of directly learning it on a huge number of parameters. Eventually, the fusion operators are initiated by the prior features of modular parameters and optimized on unlabeled data. Based on well-learned operators, it is convenient to efficiently achieve compact initial points for smaller models and facilitate subsequent training.

To verify the effectiveness, we conduct experiments on the LLaMA 2-7B model (Touvron et al., 2023) and compress it into two smaller models, 1.3B and 2.7B, respectively. Within only 0.26 billion tokens, it is sufficient to learn competent initial points for smaller models, and with an additional 50 billion tokens of post-training, LEMON outperforms a series of powerful baselines on 11 representative downstream tasks. Meanwhile, compared to traditional models trained from scratch (Biderman et al., 2023; TogetherAI, 2023; Xinyang Geng, 2023), LEMON only requires 5.26% of pre-training

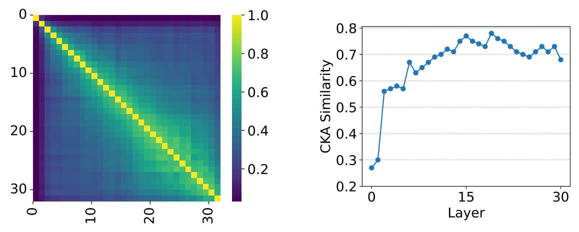

Figure 2: Interlayer parameter similarity in LLaMA 2-7B. (a) The similarity heatmap of multi-head attention in each layer, (b) The similarity curve of adjacency layers.

token budget. When achieving comparable performance to recent pruning-based method (Xia et al., 2022), the learning process of our fusion-based method saves 62.5% of training budget. Overall, LEMON exhibits considerable flexibility with controllable layer and dimension operation, making it a unified view of existing model compression techniques, which explore a new way for model compression using in-model parameters fusion.

2 Parameters Characteristics in Large Language Models

To figure out the inherent characteristics of the parameters in the large language model, we calculate the layer-by-layer parameter similarity of individual components with Centered Kernel Alignment (stated in Appendix A.6). From Figure 2 (a) (here we provide the parameter similarity of multi-head attention in LLaMA 2-7b, see Appendix A.5 for the similarity of feed forward layer), one can observe that the parameters of large LMs tend to form cluster-like modular during pre-training, in which the parameter within the cluster, especially in neighboring layers, shows a high degree of similarity, while there are significant differences in the layers outside the cluster. It is intuitive to speculate that these modules naturally distinguish the functional areas of large models, allowing for the processing of language features at different levels. Specifically, the cluster-like similarity structure primarily emerges in the middle to deeper layers, which indicates a shift from explicit textual information processing in shallower layers to more abstract and progressive feature processing in deeper layers. On the contrary, the start and final layers of these models show distinct characteristics, highlighting their unique functions.

This enlightens that, on the one hand, the observation of similar parameters shows that there is a large amount of redundancy in the pre-trained large model, which opens up the exploration of merging modules with high parameter similarity to reduce the model size. On the other hand, the highest parameter similarity is still less than 80% (see Figure 2 (b)), indicating that each part in the module has its unique function behavior and should not be directly discarded. Further motivated by findings that deep neural networks' parameter optima can be connected via linear low-loss paths (Garipov et al., 2018; Qin et al., 2022), is imperative to explore the linear fusion of parameter modules in large models to achieve model compression.

3 Method

Building a stronger small model typically involves two steps. The first step is to construct a compact initial point from a larger model while preserving its functional behavior, and the second step is to perform (restorative) pre-training to further improve performance. The second step is technically no different from training from scratch, we focus on the first step in this paper.

Formally, we define a model as a set of layer weights $\boldsymbol{\Theta}_{L,D} = [\mathrm{W}_1, \mathrm{W}_2, \cdots, \mathrm{W}_L]$, where $\mathrm{W}_l \in \mathbb{R}^{D\times D'}$ denotes the weight of layer l with row dimension D and column dimension D'. In the initialization stage, our goal is to transfer knowledge from a larger model $\boldsymbol{\Theta}^{(\text{large})}_{L_1,D_1}$ to a smaller one $\boldsymbol{\Theta}^{(\text{small})}_{L_2,D_2}$, where total layers $L_1 > L_2$, hidden dimension $D_1 > D_2$, by learning a fusion operation that minimizes the functional difference between original and fused parameters,

$$\begin{gathered}\arg\min_{\mathcal{M}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\, \mathcal{L}\left(\boldsymbol{x}; \boldsymbol{\Theta}^{(\text{small})} - \boldsymbol{\Theta}^{(\text{large})}\right) \\ \text{subject to } \boldsymbol{\Theta}^{(\text{small})} = \mathcal{M}(\boldsymbol{\Theta}^{(\text{large})})\end{gathered} \tag{1}$$

where $\mathcal{M}$ is the fusion operator, $\mathcal{D}$ is the training data set, and $\mathcal{L}$ is the loss function. However, there are a total of $\mathcal{O}\left(L_1 D_1 D_1 L_2 D_2 D_1\right)$ mapping relations between the parameters of two models, simply optimizing the above loss function is infeasible due to the high computational complexity. In the subsequent sections, we decompose the fusion operator into layer operator and dimension operator, to achieve an optimal balance between computational efficiency and model performance.

3.1 Layer Operators: Vertical Fusion with Layer-by-Layer Mapping

Based on the observing inherent modularity and redundancy observed in large models, it is very intuitive to learn layer-by-layer mapping from large models to small models, in order to achieve similar functional behaviors of corresponding clusters with fewer layers.

Following this line of thinking, we introduce Monarch matrix (Dao et al., 2022) to reduce the computational cost of matrix multiplication. By decompose the fusion operator $\mathcal{M}$ into layer operator $\mathrm{L}_{\text{layer}}$ and dimension operator R_{dim}. i.e. $\mathcal{M} = \mathrm{L}_{\text{layer}}\mathrm{R}_{\text{dim}}$. The layer operator $\mathrm{L}_{\text{layer}} = \boldsymbol{\gamma}\otimes\mathrm{I}$, where $\boldsymbol{\gamma}$ is the layer-by-layer mapping matrix, I is a highly sparse block diagonal matrix. The parameter module in the smaller model is a linear sum of the parameter modules of multiple layers in the larger model: $\mathrm{W}^{\text{small}}_i = \sum_{j=1}^{L_1} \gamma_{i,j}(\mathrm{W}^{\text{large}}_j)$, where $\gamma_{i,j}$ represents the mapping relationship from the j-th layer of the original model to the i-th layer of the target model.

$$\mathcal{M} = \underbrace{\left(\begin{bmatrix} \gamma_{1,1} & \cdots & \gamma_{1,L_1} \\ \vdots & \ddots & \vdots \\ \gamma_{L_2,1} & \cdots & \gamma_{L_2,L_1} \end{bmatrix} \otimes \mathrm{I}\right)}_{\text{Layer Operators}} \underbrace{\begin{bmatrix} \mathrm{R}^{1}_{\text{dim}} & & \\ & \ddots & \\ & & \mathrm{R}^{L_1}_{\text{dim}} \end{bmatrix}}_{\text{Dimension Operators}} \tag{2}$$

where $\mathrm{R}^{l}_{\text{dim}} \in \mathbb{R}^{D_2\times D_2}$ is the dimension operator of l-th layer. $\mathrm{L}_{\text{layer}}$ performs linear mapping of parameter modules layer by layer, enabling precise and efficient structural transformations while preserving layer functionality. This approach simplifies parameter learning and mapping, especially in large-scale models with extensive parameters[1].

3.2 Dimension Operator: Horizontal Fusion with Controllable Receptive Field

In this section, we create R_{dim}, a set of dimension operators for compressing parameters into desired shapes, by splitting R into B and A, which independently manage row and column transformations. Given a source matrix $\mathrm{W} \in \mathbb{R}^{D_1\times D_1'}$ and a target matrix $\mathrm{W}' \in \mathbb{R}^{D_2\times D_2'}$, we utilize the Kronecker product: since $\mathrm{vec}(\mathrm{CAB}) = \left(\mathrm{B}^\top \otimes \mathrm{C}\right)\mathrm{vec}(\mathrm{A})$ (Schacke, 2004), operator $\mathrm{R}_l = \mathrm{A}_l \otimes \mathrm{B}_l$, where $\mathrm{A} \in \mathbb{R}^{D_1'\times D_2'}$ and $\mathrm{B} \in \mathbb{R}^{D_1\times D_2}$, achieving $\mathrm{W}' = \mathrm{B}_l^\top \mathrm{W}\mathrm{A}_l$.

Due to the considerable size of A and B in large models, we further refine them into sparse and low-rank matrices A^* and B^* to minimize memory and computational demands. We illustrate this with B^* acting on W, defining the parameter modularization's receptive fields $\boldsymbol{\theta}$ within W as r, thus simpli-

[1]Reduce to $O(D_1 D_1 D_2 D_2 + L_1 L_2)$

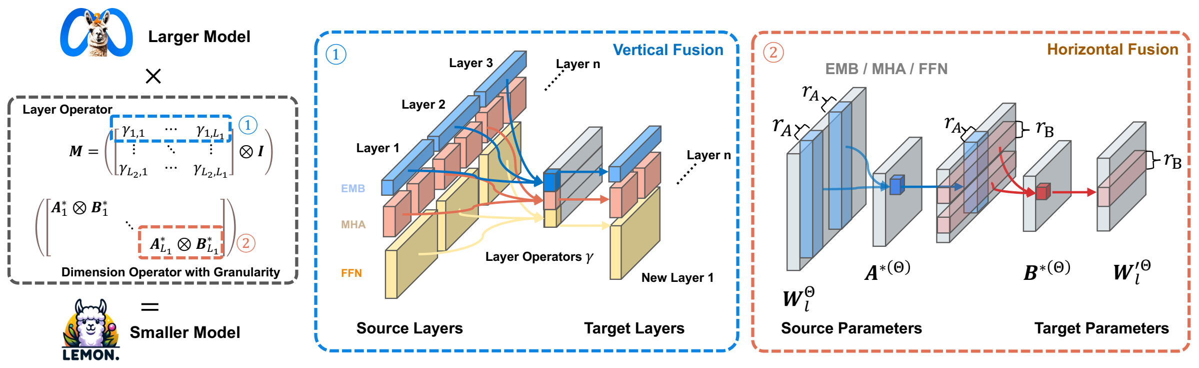
Larger Model
×
Layer Operator
$M = \left(\begin{bmatrix}\gamma_{1,1} & \cdots & \gamma_{1,L_1}\\ \vdots & \ddots & \vdots \\ \gamma_{L_2,1} & \cdots & \gamma_{L_2,L_1}\end{bmatrix} \otimes I\right)$ ①
$\left(\begin{bmatrix}A_1^* \otimes B_1^* & & \\ & \ddots & \\ & & A_{L_1}^* \otimes B_{L_1}^*\end{bmatrix}\right)$ ②
Dimension Operator with Granularity
=
Smaller Model
LEMON.
① Vertical Fusion
Layer 3
Layer n
Layer 2
Layer 1
EMB
MHA
FFN
Layer n
Layer Operators γ
New Layer 1
Source Layers
Target Layers
② Horizontal Fusion
EMB / MHA / FFN
r_A
r_A
r_A
r_B
r_B
$A^{*(\Theta)}$
$B^{*(\Theta)}$
W_l^{Θ}
$W_l'^{\Theta}$
Source Parameters
Target Parameters

Figure 3: The schematic diagram of LEMON, where the entire fusion operator M is decomposed into layer operator (middle) and dimension operator (right). The layer operator performs vertical fusion at layer fields, while dimension operator performs column fusion and row fusion at fields on the parameter matrix.

fying the formulation to $\mathrm{B}^* = \sum_{i=1}^{D_2/r} \sum_{j=1}^{D_1/r} b_{i,j}$ and $\mathrm{W} = \sum_{i=1}^{D_1/r} \boldsymbol{\theta}_{i,j}$, the formula as:

$$\mathrm{B}^*\mathrm{W} = \underbrace{\left(\begin{bmatrix} b_{1,1} & \cdots & b_{1,D_1/r} \\ \vdots & \ddots & \vdots \\ b_{D_2/r,1} & \cdots & b_{D_2/r,D_1/r}\end{bmatrix} \otimes \mathrm{I}\right)}_{\text{Row mapping matrix}} \underbrace{\begin{bmatrix} \boldsymbol{\theta}_1 & & \\ & \ddots & \\ & & \boldsymbol{\theta}_{D_1/r}\end{bmatrix}}_{\text{Parameter matrix}} \quad (3)$$

The row scaling operator B is provided with a receptive range. Each element b in B maps the parameters within its receptive range in the original matrix as a whole. By providing different receptive ranges to different parameter modules, we not only ensure the integrity of the modular parameter structure but also ensure the diversity in the mapping of parameter matrices with different depths and functionalities. Moreover, this approach optimises the operator complexity in a controlled way[2]. Finally, we have $\mathrm{R}\Theta = \begin{bmatrix} \mathrm{B}_1^*\mathrm{W}_1\mathrm{A}_1^{*\top} & \cdots & \mathrm{B}_{L_1}^*\mathrm{W}_{L_1}\mathrm{A}_{L_1}^{*\top} \end{bmatrix}$.

In Figure 3, we intuitively describe the workflow of build a compact initial point by fusing parameters from larger models, which adeptly preserves essential information embedded in the model parameters while effectively reducing the model size. Subsequently, we perform routine pre-training to optimize the goal of language modeling, the initial point will help us break through the scaling law, and quickly surpass the model performance training from scratch.

3.3 Implementation in Compressing LLaMA

Our approach is theoretically applicable to transforming parameters across various neural network designs, focusing on preserving the knowledge within large language models parameters.

[2]Reduce to $O((D_1D_2 + D_1'D_2')/r^2 + L_1L_2)$.

	Fusion Operators	Receptive Fields
Layer	$\boldsymbol{\gamma} \in \mathbb{R}^{L_1L_2}$	1
Hidden	$\boldsymbol{B}^{*\text{emb}} \in \mathbb{R}^{D_1D_2/r_h}$	r_h
Attention	$\boldsymbol{A}^{*\text{attn}} \in \mathbb{R}^{L_1D_1D_2/r_a}$	r_a
Intermediate	$\boldsymbol{A}^{*\text{ffn}} \in \mathbb{R}^{L_1H_1H_2/r_i}$	r_i

Table 1: Fusion operators designed for various modules using various receptive fields.

Using the LLaMA model as a case study, we implement our compression technique on all parameters. By compressing the embedding layer and output heads using $\mathrm{B}^{*\text{emb}}$ and sharing parameters across the compression operator's hidden size dimension, this method significantly reduces the number of parameters required for learning, enhancing efficiency. In the multi-head attention and feed forward components, due to distinct parameter roles, we tailor operators $\mathrm{A}^{*\text{attn}}$ and $\mathrm{A}^{*\text{ffn}}$ for each unique module, ensuring effective compression without loss of function.

We quantify the discrepancy between source (p_s) and target (p_t) model distributions using KL divergence $\mathcal{L}_{\text{kl}} = D_{\text{KL}}(p_s \| p_t)$ and guide mapping learning with language model loss. The total loss merges these aspects:

$$\mathcal{L}_{\text{final}} = \lambda\mathcal{L}_{\text{lm}} + (1-\lambda)\mathcal{L}_{\text{kl}} \quad (4)$$

Due to the flexibility of the LEMON, we can represent the results of other model compression methods (see Sec 3.4) and base our training on these. In implementation, to conserve resources, we experimented with various training-free initialization methods: such as directly truncating the model, extracting similar layers based on modular features, and deletion results based on pruning algorithms, among others. The pseudocode for a single forward shown in algorithmic 1 in Appendix A.

3.4 The Connection Between LEMON and Existing Methods

Layer Dropping Methods (Zhang and He, 2020b; Sajjad et al., 2023) remove layers to reduce the model's depth. The selection of subset layers $\mathbb{L}_2$ to keep ($\mathbb{L}_2 \subseteq \mathbb{L}_1$, $\mathbb{L}_1$ denotes the set of total layers) indicates the depth reduction process as:

$$W_i^{\text{init}} = \sum_{j=1}^{L_1} \gamma_{i,j} W_j^{\text{large}}, \gamma_{i,j} = \begin{cases} 1 & \text{if } j \in \mathbb{L}_2, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Structured Pruning Methods (Jiang et al., 2023; Ma et al., 2023; Xia et al., 2023) prune both rows and columns of the matrix and removes layers to achieve a target model size. It shares the layer reduction formula with layer dropping. For row and column pruning, represented by $\mathbb{D}_2 \subseteq \mathbb{D}_1$ and $\mathbb{D}_2' \subseteq \mathbb{D}_1'$ respectively, the pruning process is:

$$W_{i,m}^{\text{init}} = \sum_{j=1}^{D_1} \sum_{n=1}^{D_1'} a_{i,j} b_{m,n} W_j^{\text{large}},$$
$$a_{i,j} = \begin{cases} 1 & \text{if } j \in \mathbb{D}_2, \\ 0 & \text{otherwise.} \end{cases}, b_{m,n} = \begin{cases} 1 & \text{if } n \in \mathbb{D}_2', \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Parameter Selection Methods (Xu et al., 2023; Kim et al.) for initializing models with a subset of original parameters, including regular, random, and search-based selection, mirror the structured pruning's approach to matrix dimensionality reduction. These methods apply similar principles to selectively maintain or discard matrix dimensions and layers.

4 Experiments

4.1 Setup

Data To learn LEMON operators and continue pre-training, we use the RedPajama (TogetherAI, 2023) dataset, mirroring the training data in LLaMA across seven domains (CommonCrawl, C4, Github, Wikipedia, Books, ArXiv, Stack-Exchange). The dataset includes a validation set of 2 million tokens, a training set of 4 billion tokens, and a continued pre-training set of 50 billion tokens.

Training Our setup uses Sheared-LLaMA (Xia et al., 2023) code base on Composer package (Team, 2021), tested on 8 NVIDIA A100 GPUs (80GB). The models were trained at the sequence length of 4096 using a batch size of 32 during the pruning phase and continued pre-training phase. Following the setup in Sheared-LLaMA (Xia et al., 2023), baseline models were trained for 6400 steps (22 hours) with dynamic batch gradient loading (DoReMi (Xie et al., 2023)) during the pruning phase. In contrast, LEMON were trained for 1200 steps (8 hours) without DoReMi. For continued pre-training, we employed 50 billion tokens with DoReMi with 256 batch size (387 hours). Learning rates were set to 1e-4 for language modeling loss, and for layer and dimension operators, 5e-4 and 5e-5, respectively. For more information, please refer to Appendix A.3.

Evaluation Following the multiple assessment metrics reported in the baseline studies (Xia et al., 2023), we utilized lm-evaluation-harness (Gao et al., 2023) to evaluate our models. In common sense and reading comprehension evaluation, we report 0-shot accuracy of SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), WinoGrande (Abbreviated as Wino.) (Sakaguchi et al., 2020), ARC Easy (Clark et al., 2018a), HellaSwag (Zellers et al., 2019), 25-shot accuracy of ARC Challenge (Clark et al., 2018b). In world knowledge evaluation, we report accuracy of 32-shot NQ (Kwiatkowski et al., 2019) and 5-shot MMLU (Hendrycks et al., 2021). In Continued QA and text understanding evaluation, we report accuracy of 0-shot LogiQA (Liu et al., 2020), 32-shot BoolQ (Clark et al., 2019) and 0-shot LAMBADA (Paperno et al., 2016).

4.2 Baseline

Structured Pruning Methods construct initial points by searching and eliminating non-essential neurons, followed by continued pre-training. We designate Sheared-LLaMA (Xia et al., 2023) as our primary baseline. Sheared-LLaMA provided a robust training framework, dynamic batch gradient pre-training methods, and an evaluation framework, achieving competitive outcomes.

Pre-training from Scratch Baseline Models train from scratch without utilizing any pre-existing model parameters. We choose similar-scale open-source LLMs includes Pythia-1.4B & -2.8B (Biderman et al., 2023), which use different training data than RedPajama. We choose INCITE-Base-3B (TogetherAI, 2023) as baseline which use the same training data.

4.3 Result

Foundational Capabilities. Table 2 shows the foundational capacity performance of the model

Model (#tokens for training)	**SciQ**	**PIQA**	**WinoGrande**	**ARC-E**	**ARC-C (25)**	**HellaSwag (10)**
	Commonsense & Reading Comprehension					
LLaMA 2-7B (2T)†	93.7	78.1	69.3	76.4	53.0	78.6
Pythia-1.4B (300B)†	86.4	70.9	57.4	60.7	31.2	53.0
Sheared-LLaMA-1.3B (50B)	87.3	73.4	57.9	61.5	33.5	60.7
LEMON-1.3B (50B)	**87.9**	**73.7**	**58.5**	**62.2**	**36.3**	**61.5**
Pythia-2.8B (300B)†	88.3	74.0	59.7	64.4	36.4	60.8
INCITE-Base-3B (800B)	90.7	74.6	63.5	67.7	40.2	64.8
Sheared-LLaMA-2.7B (50B)	90.8	75.8	64.2	67.0	41.2	70.8
LEMON-2.7B (50B)	**91.5**	**76.5**	**65.4**	**68.2**	**42.2**	**71.9**

	Continued		**LM**	**World Knowledge**		
Model (#tokens for training)	**LogiQA**	**BoolQ (32)**	**LAMBADA**	**NQ (32)**	**MMLU (5)**	**Average**
LLaMA 2-7B (2T)†	30.7	82.1	28.8	73.9	46.6	64.6
Pythia-1.4B (300B)†	27.3	57.4	61.6	6.2	25.7	48.9
Sheared-1.3B (50B)	**26.9**	64.0	61.0	9.6	25.7	51.2
LEMON-1.3B (50B)	26.4	**65.5**	**62.1**	**10.2**	**26.1**	**51.9**
Pythia-2.8B (300B)†	28.0	66.0	64.7	9.0	26.9	52.5
INCITE-Base-3B (800B)	27.7	65.9	65.3	14.9	**27.0**	54.7
Sheared-LLaMA-2.7B (50B)	**28.9**	73.7	68.4	16.5	26.4	56.7
LEMON-2.7B (50B)	28.5	**74.3**	**69.6**	**18.1**	26.9	**57.6**

Table 2: Comprehensive assessment of model's fundamental capabilities, in which LEMON models demonstrate competitive performance while requiring significantly fewer training resources.

after initialization with LEMON operators then continued pre-training with 50B tokens. We also evaluate capabilities of similarly sized models and models built from pruning methods. Experiments show that our approach achieves even better performance with very little training resource overhead (i.e. 50B tokens of training data) compared to models built from scratch. LEMON-1.3B outperforms the Pythia-1.4B models, which were initially pretrained with 300B tokens. LEMON-2.7B also outperforms 300B tokens pretrained Pythia-1.4B models and 300B tokens pretrained INCITE-Base-3B. Compared to prune-based initialization methods, we obtain better initial points retaining the knowledge of larger model, which leads to substantial savings in pre-training. Our LEMON-2.7B, LEMON-1.3B outperforming Sheared-LLaMA-2.7B and Sheared-LLaMA-1.3B respectively.

Better Initialization. We check whether LEMON produces a better initialization than existing language models of the same size. We continue to pre-train the LEMON-2.7B model on the original RedPajama data, comparing it to the INCITE-Base-3B adn Sheared-LLaMA-2.7B model. Figure 4 shows the INCITE-Base-3B model has much higher initial accuracy, but its performance stagnates throughout the continued pre-training. Sheared-LLaMA and LEMON-2.7B has lower initial accuracy but rapidly improves and eventually outperforms the INCITE-Base-3B model. LEMON-2.7B has higher initial accuracy and final performance than Sheared-LLaMA-2.7B. The parameter-fused model outperforms the pruned model and is more suitable for initialising a strong model for further pre-training.

Fast Convergence. We report the Eval PPL during training process for LEMON and Sheared-LLaMA in the same configuration. As shown in the Figure 4 (Middle), although the initial point is not sufficiently satisfactory, LEMON can converge quickly within 2,000 steps, using only 33% of Sheared-LLaMA training steps. LEMON reaches a PPL of 7.47 at 2,000 steps, which is below the 7.49 PPL of Sheared-LLaMA at 4,800 steps. The PPL of LEMON achieves parity with Sheared-LLaMA at 500 steps and surpass the Sheared-LLaMA by 3.5 point at 1000 steps, demonstrating the ability to effectively retain the information in parameters. We can continue to allocate the LEMON training

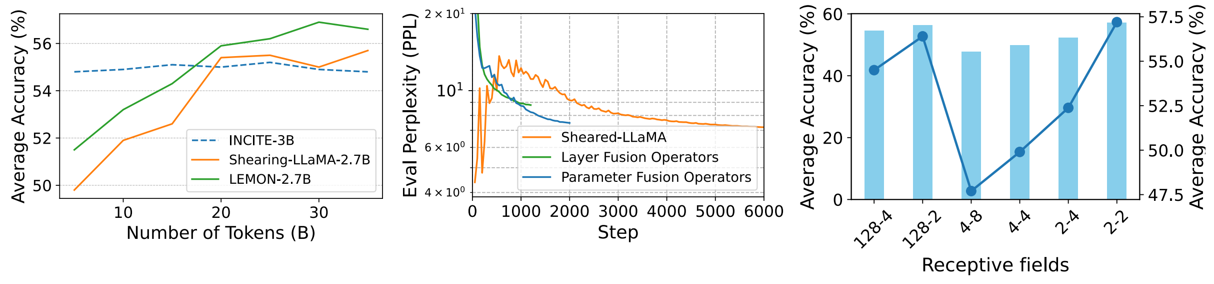

Figure 4: **Left:** Comparative downstream task accuracy of continued pre-training. **Middle:** Eval Perplexity (PPL) during training of LEMON and Sheared-LLaMA (in the 2.7B setting). **Right:** Average accuracy (%) of dimension operators on different receptive fields. (r_1-r_2) represents the receptive fields of MHA r_1 and FFN r_2.

Method	L.	SciQA	PiQA	Wino.	ARC-E	LogiQA	AVG
LLaMA2-7B	32	93.5	78.0	69.7	76.3	30.7	69.6
Cut	24	81.7	65.7	59.0	53.8	26.0	57.2
Random		45.4	66.6	54.1	34.7	25.0	45.2
Avg		31.4	55.7	50.0	28.9	26.3	38.4
Shear		82.8	71.3	55.0	**62.9**	26.3	59.7
LEMON-Prune		87.2	71.9	56.0	60.9	26.4	60.4
LEMON-Cut		**92.0**	**71.9**	**64.0**	55.2	**26.9**	**62.0**
Cut	16	28.4	56.6	57.4	32.3	25.0	40.0
Random		23.6	53.9	50.2	26.4	24.0	35.6
Avg		20.6	52.9	46.9	25.3	26.1	34.4
Shear		33.2	55.7	49.4	29.8	24.0	38.4
LEMON-Prune		**72.2**	**62.8**	50.5	35.6	22.1	48.6
LEMON-Cut		65.9	62.2	**59.4**	**44.3**	**24.9**	**51.3**

Table 3: Results of ablation experiments, here we only use the layer operator to compress the model.

Method	SciQA	PiQA	Wino.	ARC-E	LogiQA	AVG
LLaMA2-7B	93.5	78.0	69.7	76.3	30.7	69.6
Shear	83.9	68.9	**56.0**	**51.0**	22.4	56.4
LEMON 128-2	84.0	70.0	54.4	48.9	25.6	56.5
LEMON 2-4	79.0	63.5	51.7	44.1	23.2	52.3
LEMON 2-2	**84.4**	**69.8**	55.3	50.8	**25.7**	**57.2**
w/o LEMON	20.7	52.9	50.2	25.8	21.0	34.1

Table 4: Results of ablation experiments, here we only use the dimension operator to compress the model.

budget to reduce the initial PPL, which involves a trade-off in computational resource allocation between compression and pre-training.

4.4 Analysis

In this section, we evaluate our layer fusion operators by reducing LLaMA2-7B's depth from 30 to 24/16 layers **(L.)** and evaluate dimension operators through compress parameter matrix to 62.5% origin dimensions. We compare this with the following methods: last layers cutting **(Cut)**, random layer removal **(Random)**, layer parameter averaging **(Avg)**, and Sheared-LLaMA **(Shear)**. For weight averaging, we cluster similar layers, averaging within clusters for new parameters. We choose the optimal result from Sheared-LLaMA as our baseline. The LEMON-Cut and LEMON-Prune operator initialized from the **Cut** and **Shear**, then were trained for 1200 steps. The results are show in Table 3.

Layer Fusion As shown in 3, even in the dimension of layer fusion alone, our method shows great advantages. New model parameters after fusion improve 6.1% and 7.5% absolute accuracy compared to the initial point in the scenarios with 8-layer deletion and 16-layer deletion, and 3.4% and 7.9% compared to pruning approach. The effectiveness of LEMON improves as the magnitude of model compression increases, suggesting LEMON help to preserve the Large Model performance at large-scale parameter discarding.

Dimension Fusion We report the accuracy of LEMON dimension compression for different configurations of Multi-Head Attention **(MHA)** and Feedforward Neural Network **(FFN)** in the Table 4 and Figure 4 (Right). Under the simple **Cut** initialization method, our operator achieves Sheared-LLaMA performance. Even at high dimension compression settings (4096 to 2560), the fused model performance still rise dramatically compared to the initial point. This indicates the strong adaptability and generalisation of our method.

Receptive Fields Table 4 and Figure 4 (Right) shows the performance of dimension compression operators under different receptive fields settings (MHA-FFN). (r_1-r_2) represents the MHA operator receptive fields of r_1 and the FFN operator receptive fields of r_2. In the experiment, Attention modules show robustness to larger receptive fields (e.g.128), highlighting heads' modular parameter structure. Conversely, hidden and inter-

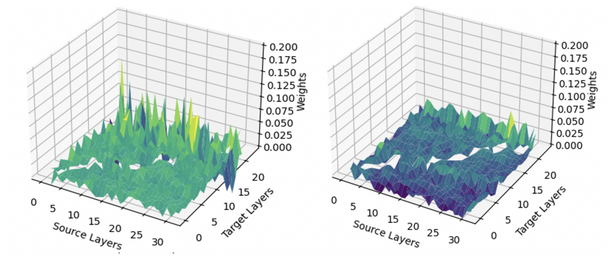

Figure 5: Visualization of the weight distribution of layer operators in LLaMA 2-7B. **Left:** the operator on MHA; **Right:** the operator on FFN.

mediate sizes are more sensitive, where finer settings yield greater performance boosts. However, the increased computational overhead from finer receptive fields suggests a balance is necessary. Optimal receptive fields settings should consider the model's parameter modularity, waiting further systematic investigation.

Operators Initialization **Cut** and **Prune** represent the operators' initialization based on direct cutting model and pruning algorithm results. As Shown in Table 3, the starting point of operators affects the performance after LEMON fusion. Although the performance is lower after cutting model, through learning, LEMON enhancing the average accuracy by 4.8% when retaining 24 layers and by 11.3% when retaining 16 layers. These improvements surpass the pruning search algorithm by 2.3% and 12.9% under the same experimental settings. The experiments also demonstrate that LEMON operator can be combined with existing compression methods, improving upon the pruning methods by 0.7% and 10.2%. Experiments with 7B and 3B models indicate that providing LEMON with better initial points helps enhance model's performance ceiling. This is a valuable area for future research. For more information about compressing 3B model, please refer to Appendix A.4.

Explainability Analysis We visualized the layer fusion operators of the LLaMA model in Figure 5 and Appendix A.7. Through learning, the dimension operators autonomously form and aggregate clusters of similar layers within the model, reflecting observed parameter module similarity. The operator weights decrease with distance, even turning negative, indicating the operators deem distant parameter modules irrelevant or even detrimental to the current module. After fusion, the parameter distribution within the LEMON model still shows consistency with the original model (discussed in Appendix A.5, Figure 6). This indicates that new fused parameters reconstructed the functions of multiple parameters from the original model.

5 Related Work

Efficient pre-training approaches In recent years, the ability of incremental training to accelerate large-scale model training by studying how to obtain the optimal initialization point for training has thus attracted much attention (Xie et al., 2017; Wu et al., 2019). Asymptotic training (Zhang and He, 2020a) is used to learn larger-scale models by training small transformers with a small number of layers, and then gradually expanding them by stacking layers. Net2Net (Chen et al., 2015) uses function-holding transformations to expand the width by duplicating neurons, and uses a unitary layer implementation to expand the depth. LiGO (Wang et al., 2023) proposes a learnable expansion method that can be used at the initial initialization point of a transformer. learned expansion methods that can expand models of arbitrary structure at initialization. LEMON is inspired by these methods, but we investigate how to learn to map the parameter matrix from large to small without losing the ability of the larger model itself.

Model Compression Our approach is dedicated to obtaining a high-performance lightweight language model, which is the same goal as the task of model compression. Quantization (Gray and Neuhoff, 1998) reduces the numerical accuracy of model weights and activations, and speeds up training and inference, but results in a loss of model accuracy and the inability to freely build target-specific models. CRash (Zhang et al., 2023) and LayerDrop (Zhang and He, 2020b; Sajjad et al., 2023) methods discard ineffective layers during training, which do not allow for target-specific structuring and come with a large performance loss. Pruning (Wang et al., 2020) minimizes the impact on performance by cutting out redundant neurons that over-parameterize the model. In the LLM era, this leads to a significant reduction in neuron redundancy as models move from task-specific to generalized (Frantar and Alistarh, 2023). Pruning LLM leads to performance degradation at larger pruning magnitudes. LLMsheairng (Xia et al., 2023) uses the results of pruning as initialization for continuous pre-training of the model to recover performance, but this approach requires more data and computational overhead. We avoid the information loss caused, by learning the parameter fusion matrix of the model to reach a specific structure, thus obtaining better initialization points and reducing the overhead of continuous pre-training.

6 Conclusion

In this paper, we propose a new paradigm for building smaller LMs based on larger LMs. By learning a parameter fusion operator with controllable receptive fields from a larger model to smaller models, we can obtain a good starting point to facilitate subsequent training. This fusion mapping operator consists of a layer operator and a dimension operator. Experimental results demonstrate that our method can compress larger LMs into smaller LMs of arbitrary architectures and better preserve the knowledge in large models. By comprising the requirements of pre-training data, we demonstrate the effectiveness of the method against several baseline approaches in terms of training speedup and computational expenditure savings. The paradigm has greater research value and has the potential to reach smaller performance losses and lower computational effort in the future.

Limitation

There are limitations to our approach: First, we have only explored the use of linear methods for parameter fusion in our model. In the future, nonlinear methods deserve more exploration as they have the potential to better link different parameters and reach optimality. Second, limited by computational resources, we only experiment on 7B-scale and 3B-scale models. However, our method is scalable (discussed in Appendix A.4) and can be extended to models of arbitrary size in future work. Third, although we minimize the computational complexity of the fusion operator, it still requires a lot of memory and computation for optimization. Finally, we believe that the fusion of parameters within models is a direction that has not been fully explored. How to solve the information loss caused by parameter fusion is a problem we will try to solve in the future.

Ethical Consideration

In our study, we leverage open and online accessible data and techniques, mitigating privacy concerns. Our method emphasizes enhancing model parameter efficiency and reducing size to create powerful, compact, and openly accessible models, thereby promoting the open dissemination and democratization of NLP technologies. By employing pre-training strategies, we aim to mitigate biases through extensive training and large corpus sizes, contributing to an ethical AI development that prioritizes openness, efficiency, and bias reduction. Our work is committed to advancing accessible and efficient NLP technologies, fostering a more inclusive and automated future for AI.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and constructive suggestions. We would like to thank Yinqi Yang, Jiawei Sheng, Xinhua Zhang, Shicheng Wang, Chuanyu Tang and members of the IIE KDsec group for their valuable feedback and discussions. We are very grateful to Mengzhou Xia for providing the concise and effective ShearingLLaMA experimental code and for her assistance during the reproduction process. Work done during Yilong Chen's internship in Baidu Inc. This research is supported by the National Key Research and Development Program of China (Grant No.2021YFB3100600) and the Youth Innovation Promotion Association of CAS (Grant No.2021153).

References

Anthropic. 2023. Introducing claude.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

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. 2018a. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018b. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Tri Dao, Beidi Chen, Nimit Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. 2022. Monarch: Expressive Structured Matrices for Efficient and Accurate Training. ArXiv:2204.00595 [cs].

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot.

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. 2023. A framework for few-shot language model evaluation.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Robert M. Gray and David L. Neuhoff. 1998. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Ting Jiang, Deqing Wang, Fuzhen Zhuang, Ruobing Xie, and Feng Xia. 2023. Pruning Pre-trained Language Models Without Fine-Tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, 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, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. SOLAR 10.7B: Scaling Large Language Models with Simple yet Effective Depth Up-Scaling.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, page abs/2303.08774.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Yujia Qin, Cheng Qian, Jing Yi, Weize Chen, Yankai Lin, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2022. Exploring Mode Connectivity for Pre-trained Language Models.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Kathrin Schacke. 2004. On the kronecker product. *Master's thesis, University of Waterloo*.

The Mosaic ML Team. 2021. composer. `https://github.com/mosaicml/composer/`.

TogetherAI. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

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.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured Pruning of Large Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 94–106. Association for Computational Linguistics.

Lemeng Wu, Dilin Wang, and Qiang Liu. 2019. Splitting steepest descent for growing neural architectures. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10655–10665.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*.

Di Xie, Jiang Xiong, and Shiliang Pu. 2017. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5075–5084. IEEE Computer Society.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. *CoRR*, abs/2305.10429.

Hao Liu Xinyang Geng. 2023. Openllama: An open reproduction of llama.

Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. 2023. Initializing Models with Larger Ones.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Kaiyan Zhang, Ning Ding, Biqing Qi, Xuekai Zhu, Xinwei Long, and Bowen Zhou. 2023. CRaSh: Clustering, Removing, and Sharing Enhance Fine-tuning without Full Large Language Model.

Minjia Zhang and Yuxiong He. 2020a. Accelerating Training of Transformer-Based Language Models with Progressive Layer Dropping. In *Advances in Neural Information Processing Systems*, volume 33, pages 14011–14023. Curran Associates, Inc.

Minjia Zhang and Yuxiong He. 2020b. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in Neural Information Processing Systems*, 33:14011–14023.

A Appendix

A.1 LEMON Operator

Our final operator is as follows:

$$\boldsymbol{M} = \underbrace{\left(\begin{bmatrix} \gamma_{1,1} & \cdots & \gamma_{1,L_1} \\ \vdots & \ddots & \vdots \\ \gamma_{L_2,1} & \cdots & \gamma_{L_2,L_1} \end{bmatrix} \otimes \boldsymbol{I} \right)}_{\boldsymbol{L}_{\text{Layer}}} \underbrace{\left(\begin{bmatrix} \boldsymbol{A}_1^* \otimes \boldsymbol{B}_1^* & & \\ & \ddots & \\ & & \boldsymbol{A}_{L_1}^* \otimes \boldsymbol{B}_{L_1}^* \end{bmatrix} \right)}_{\boldsymbol{R}_{\text{Param}}} \quad (7)$$

We freeze the parameters of LLaMA model and optimise only Lemon Fusion operators. In the LLaMA-7B model, our single forward operation is summarised in Algorithm 1.

Algorithm 1 Learning operators in LLaMA

```
1: Model parameters $\theta$, Hidden dimensions $D_1$, $D_2$, Number of layers $L_1$ and $L_2$, Intermediate sizes $H_1$, $H_2$, Embedding sizes $E$.
2: $\boldsymbol{W}^{(\text{emb})} \in \mathbb{R}^{D_1 \times E}, \boldsymbol{W}_l^Q, \boldsymbol{W}^K, \boldsymbol{W}_l^V, \boldsymbol{W}_l^O \in \mathbb{R}^{D_1 \times D_1}, \boldsymbol{W}_l^{(\text{up})}, \boldsymbol{W}_l^{(\text{gate})} \in \mathbb{R}^{H_1 \times D_1}, \boldsymbol{W}_l^{(\text{down})} \in \mathbb{R}^{D_1 \times H_1}, \boldsymbol{W}_l^{(\text{ln1})}, \boldsymbol{W}_l^{(\text{ln2})} \in \mathbb{R}^{D_1}, \forall l \in [L_1], \boldsymbol{W}^{(\text{out})} \in \mathbb{R}^{E \times D_1}$
3: initialization operator $\gamma, A^*, B^*$
4: Step 1: Learning Depth Mapping
5: for $l = 1, \cdots, L_2$ do
6:     for $\Theta = Q, K, V, O, up, gate, down, ln1, ln2$ do
7:         $\boldsymbol{W'}_l^{\Theta} \leftarrow \sum_{j=1}^{L_1} \gamma_{l,j}^{\Theta} \boldsymbol{W}_j^{\Theta}$
8:     end for
9: end for
10: Update $\theta, \gamma$ with $\mathcal{L}(\theta, \gamma)$
11: Step 2: Learning Width Mapping
12: $\boldsymbol{W'}^{(\text{emb})} \leftarrow \boldsymbol{B}^{*(\text{emb})} \boldsymbol{W}^{*(\text{emb})}$
13: for $l = 1, \cdots, L_1$ do
14:     for $\Theta = Q, K, V,$ up, gate do
15:         $\boldsymbol{W'}_l^{\Theta} \leftarrow \boldsymbol{A}^{*(\Theta)} \boldsymbol{W}_l^{\Theta} \boldsymbol{B}^{*(\text{emb})\top}$
16:     end for
17:     for $\Theta = O,$ down do
18:         $\boldsymbol{W'}_l^{\Theta} \leftarrow \boldsymbol{B}^{*(emb)} \boldsymbol{W}_l^{\Theta} \boldsymbol{A}^{*(\Theta)\top}$
19:     end for
20:     for $\Theta = ln1, ln2$ do
21:         $\boldsymbol{W'}_l^{(\text{ln2})} \leftarrow \boldsymbol{B}^{*(\text{emb})} \boldsymbol{W}_l^{(\text{ln2})}$
22:     end for
23: end for
24: $\boldsymbol{W'}^{(\text{out})} \leftarrow \boldsymbol{W}^{(\text{out})} \boldsymbol{B}^{(\text{emb})\top}$
25: Update $\theta, A^*, B^*$ with $\mathcal{L}(\theta, A^*, B^*)$
26: Step 3: Continuous-pretrain model with parameters $\theta$.
```

A.2 Model Configurations

For a fairer comparison, we directly used the model architecture of baseline works (Xia et al., 2023) to construct LEMON-2.7B & -1.3B. Beyond the initial, in order to perform ablation experiments to verify the performance of the layer fusion operator and the dimension fusion operator, we constructed LEMON-Vertical, LEMON-Horizontal -37.5% and -50% . Table 5 report the parameter configurations adopted for the different scale models in our experiments.

A.3 Training Details

The hyperparameters used in our experiments are presented in Table 6. We employ fully sharded data parallel to efficiently train our models in parallel. A cosine learning rate scheduler is used, with the learning rate decaying to a minimum of 10% of the peak value. Preliminary experiments were conducted to determine the optimal peak learning rate for learning the fusion operators and Lagrange multipliers.

A.4 Pre-experiments in 2.7B Model and LEMON Scalability

Considering the experimental cost and compression efficiency, we conduct pre-experiments on Sheared-LLaMA 2.7B to explore the optimal compression operators and hyperparameter comparisons, which are later applied to the 7B-scale LLaMA model.

As shown in Table 7, LEMON demonstrated the same excellent performance on both the compressed 3B model and the compressed 7B model. The LEMON fusion operator with the clipping and pruning algorithms as initial points improves performance by 3.5% and 4.2%, respectively, when compressing the 2.7B model.This is consistent with our observation that the LEMON fusion operator with clipping as an initial point improves performance by 4.8% on the 7b scale. On more layers of compression, our approach achieves equally impressive performance at 2.7B and 7B. This suggests that our method has good scale scalability and can be applied to larger models with good performance.

It is worth noting that our approach is validated at different model scales with appropriate hyperparameters and applied at larger scales. For example, in both 2.7B and 7B model compression we adopt a layer fusion operator learning rate of 5e-4, and a dimension fusion operator learning rate of 5e-5, as shown in Table 7. In the pre-experiments, we also found that layer fusion (deleting layers) inevitably brings more information loss, so we perform dimension fusion first and then layer fusion.

A.5 Parameter Modularity in LLaMA2 Model and LEMON Model

Here we show the modularity phenomenon observed in the llama model at all scales. For details,

	hidden_size	intermediate_size	attention_heads	hidden_layers
LLaMA-7B	4096	11008	32	32
LEMON-Vertical	4096	11008	32	24/16
LEMON-Horizontal -37.5%	2560	6912	20	32
LEMON-Horizontal -50%	2048	5504	16	32
LEMON-2.7B	2560	6912	20	32
LEMON-1.3B	2048	5504	16	24

Table 5: Parameter Configurations of LEMON models

	Fusion	Contined Pre-training
Training budget	0.26B	50B
Learning rate of $L_{\text{layer}} \in \mathcal{M}$	0.0005	-
Learning rate of $R_{\text{dim}} \in \mathcal{M}$	0.00005	-
Learning Rate of θ	0.0001	0.0001
LR warmup ratio	10%	3%
Batch size (tokens)	131K	1M
Evaluation interval m (steps)	40	40
Steps	$2,000$	$5,000$
# GPUs	8	8

Table 6: Training hyper-parameters

Method	**L.**	**AVG**	**Method**	**L.**	**AVG**
ShearedLLaMA-2.7B	32	63.4	LLaMA2-7B	32	69.6
Cut	24	50.3	Cut	24	57.2
Random		44.7	Random		45.2
Avg		31.2	Avg		38.4
Shear		49.9	Shear		59.7
LEMON-Prune		**54.1**	LEMON-Prune		60.4
LEMON-Cut		53.8	LEMON-Cut		**62.0**
Cut	16	30.9	Cut	16	40.0
Random		29.0	Random		35.6
Avg		28.1	Avg		34.4
Shear		32.1	Shear		38.4
LEMON-Prune		**41.5**	LEMON-Prune		48.6
LEMON-Cut		39.6	LEMON-Cut		**51.3**

Table 7: Results of pre-experiments in Sheared-LLaMA 2.7B, here we only use the layer operator to compress the model.Sheared-LLaMA 2.7B has the same number of layers as LLaMA-7B, but with smaller hidden dimension and numbers of heads.

see Figure 7, 8. We visualized the similarity of the Attention query matrices between layers in a 24-layer model compressed using LEMON layer fusion operators in Figure 6. We compared this with the original, uncompressed LLaMA2-7B model. Although the fused model has not yet undergone pre-training, the parameter distribution within the model still shows consistency with the original model. This indicates that LEMON successfully reduced the size of similar parameter clusters and that the new fused parameters successfully reconstructed the functions of multiple parameters from the original model.

A.6 Centered Kernel Alignment (CKA)

Centered Kernel Alignment (CKA) is a statistical measure used to quantify the similarity between two sets of data representations. Unlike traditional correlation measures, CKA is designed to be invariant to orthogonal transformations and scaling of

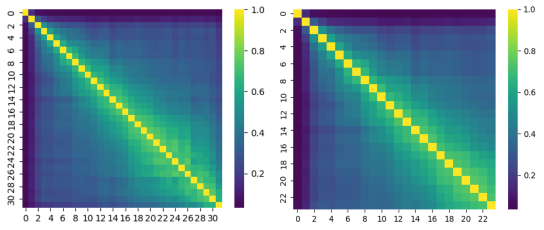
Figure 6: Visualization of Attention Query Matrices Similarity. **Left:** LLaMA2-7B; **Right:** LEMON-Vertical-24 w.o. Continued Pre-training.

the data.

To calculate the similarity between two sets of representations using CKA, we employ a kernel function to map the original data into a higher-dimensional space, where the alignment of their central tendencies can be more easily measured. The CKA value ranges from 0 to 1, where 0 indicates no similarity and 1 indicates identical representations.

The mathematical formulation of CKA, when using a linear kernel, is given by the following equation:

$$\text{CKA}(X, Y) = \frac{\|X^T Y\|_F^2}{\sqrt{\|X^T X\|_F^2 \cdot \|Y^T Y\|_F^2}}$$

Here, X and Y are matrices whose columns are the vectors of the representations to be compared, $\| \cdot \|_F$ denotes the Frobenius norm, and X^T and Y^T are the transposes of X and Y, respectively.

A.7 Operator Visualisation

In order to better observe the parameter distributions of the operators and investigate the explainable principles, we visualised the post-training distributions of the operators initialised based on Cut's method in Figure 10.

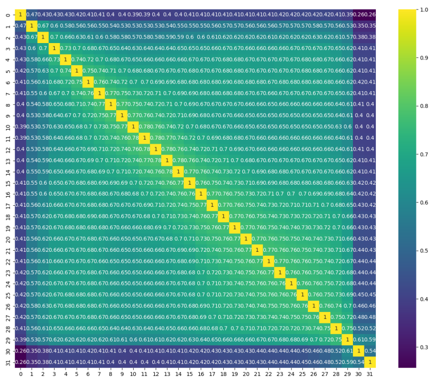
Figure 7: LLaMA2-7B-FFN

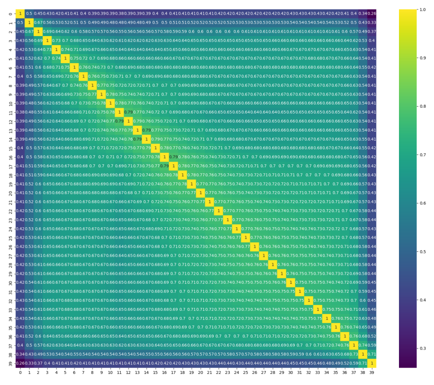
Figure 8: LLaMA2-13B-FFN

Figure 9: FFN Weights Similarity of different scale LLaMA2 Models

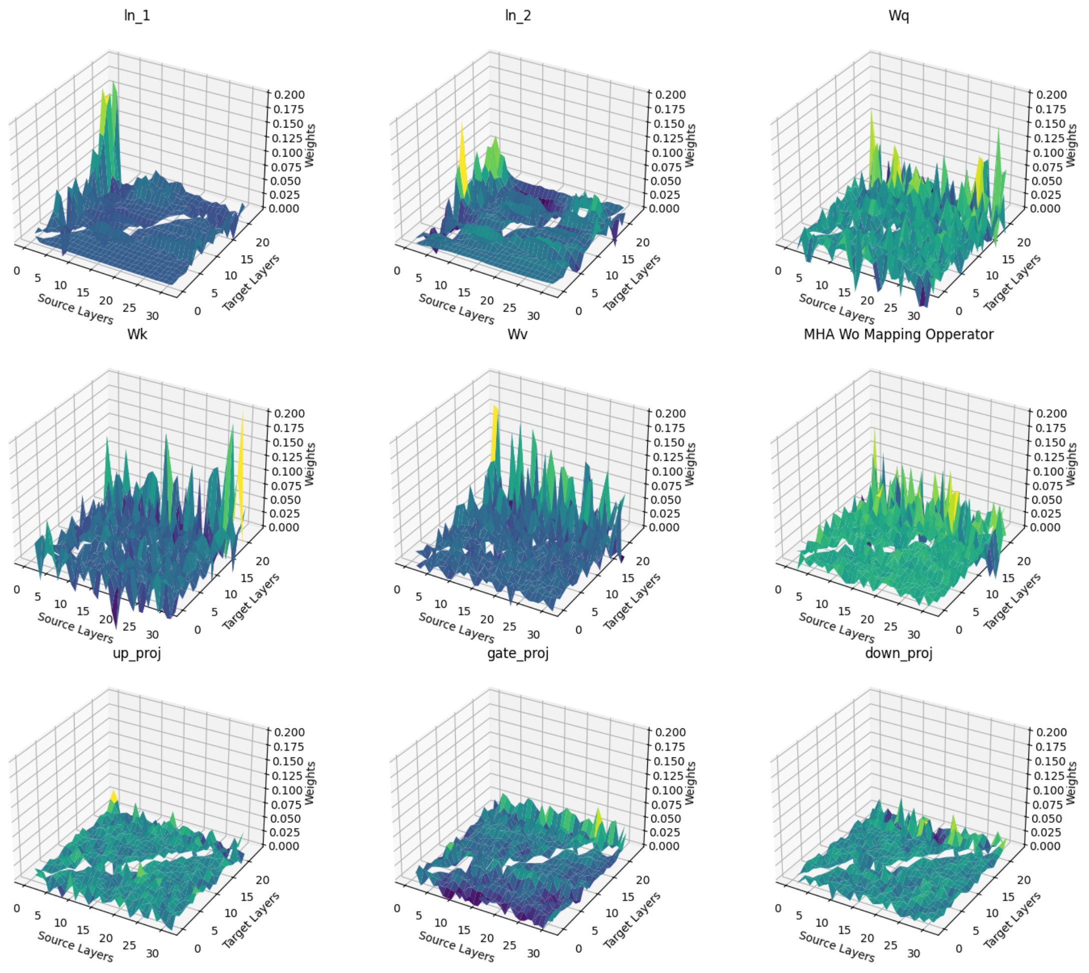

Figure 10: Visualisation of operator parameters under Cut initial points