Finedeep: Mitigating Sparse Activation in Dense LLMs via Multi-Layer Fine-Grained Experts

Anonymous ACL submission

Abstract

Large language models have demonstrated exceptional performance across a wide range of tasks. However, dense models usually suffer from sparse activation, where many activation values tend towards zero (i.e., being inactivated). We argue that this could restrict the efficient exploration of model representation space. To mitigate this issue, we propose **Finedeep**, a **deep**-layered **fine**-grained expert architecture for dense models. Our framework partitions the feed-forward neural network layers of traditional dense models into small experts, arranges them across multiple sub-layers. A novel routing mechanism is proposed to determine each expert's contribution. We conduct extensive experiments across various model sizes, demonstrating that our approach significantly outperforms traditional dense architectures in terms of perplexity and benchmark performance while maintaining a comparable number of parameters. Moreover, we find that Finedeep achieves optimal results when balancing depth and width, specifically by adjusting the number of expert sub-layers and the number of experts per sub-layer. Empirical results confirm that Finedeep effectively alleviates sparse activation and efficiently utilizes representation capacity in dense models.

1 Introduction

Large language models (LLMs) have recently gained much attention for their exceptional performance across various tasks (Achiam et al., 2023; Dubey et al., 2024). Scaling laws at the pre-training stage of LLMs suggest that increasing model size could consistently enhance performance on downstream tasks (Kaplan et al., 2020; Hoffmann et al., 2022). However, such improvement often comes at an exorbitant computational cost. As a result, maximizing model performance within a fixed parameter budget has emerged as an efficient paradigm, aiming to push the upper bound of the model performance without significantly increasing resource demands (Zhang et al., 2024).

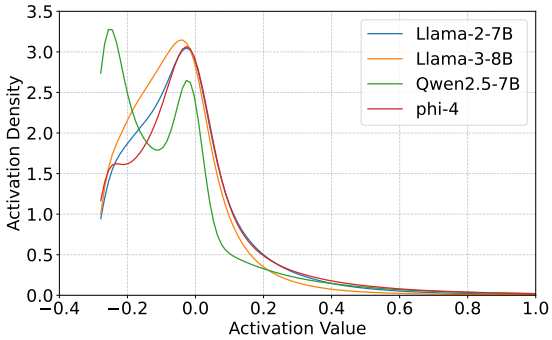

Figure 1: Distribution of activation function outputs across various models (Touvron et al., 2023b; Dubey et al., 2024; Yang et al., 2024a; Abdin et al., 2024), where all selected models use the SiLU activation function. The horizontal axis represents the activation values. The vertical axis denotes the distribution of activation values across different models.

Along with model scaling, recent studies disclose that dense models usually suffer from sparse activation issue during computation (Zhang et al., 2022) as illustrated in Figure 1. Specifically, sparse activation refers that most values output from the activation functions tend to be close to zero (Luo et al., 2024). Since these small values contribute marginally when multiplied by model parameters, their impact on the final output remains limited, leading to inefficient activation utilization. We argue that addressing sparse activation could serve as a new channel for further improving the upper limit of model performance. By improving the effective utilization of activation values, we could enhance the representational capacity of models, enabling them to capture additional complex features.

Following this direction, we identify two key limitations in standard dense Feed-Forward Networks (FFNs) contributing to sparsity: 1) Large, monolithic FFNs exhibit stronger global sparsity tendencies; 2) The single-layer nonlinear transformation

structure may prematurely suppress features with limited recovery mechanisms. To address these limitations, we propose Finedeep, a novel dense architecture employing deep-layered fine-grained experts within the FFN module. Finedeep tackles sparse activation through: 1) **Fine-grained expert decomposition**: We decompose the standard large FFN into multiple smaller, specialized experts. Each expert operates on a localized feature subspace, inherently reducing the propensity for global sparsity observed in large monolithic FFNs. 2) **Multi-layer expert arrangement**: Finedeep organizes experts into sequential sub-layers, forming a deeper processing structure within the FFN. Residual connections bridge these sub-layers. This design allows information suppressed in early sub-layers to be preserved and potentially reactivated by specialized experts in subsequent layers, mitigating the risk of permanent feature suppression inherent in single-layer FFNs. By alleviating the sparse activation issue, Finedeep achieves more effective utilization of activation values, leading to a richer representational space and improved performance.

While decomposing the FFN into multi-layer fine-grained experts addresses the root cause of sparse activation, it introduces a key challenge: how to effectively integrate the diverse outputs from these deeply arranged experts. Traditional MoE uses input-guided softmax routing to assign expert weights and select a sparse subset. However, this approach is ill-suited for Finedeep's dense, fully activated, and multi-layer structure, for two main reasons: 1) Competitive suppression: softmax normalization enforces competition among experts, suppressing the contributions of non-dominant ones, which contradicts our goal of leveraging all experts for every input; 2) Error propagation: suboptimal routing in early layers can degrade representations and affect downstream processing. To overcome these issues, we propose a novel output-guided sigmoid routing strategy. It computes routing scores based on expert outputs, directly reflecting each expert's actual contribution, and applies sigmoid normalization to evaluate contributions independently. This encourages collaboration rather than competition, and the final output is a weighted sum of all expert outputs.

To validate the effectiveness of Finedeep, we conduct extensive LLM pre-training experiments. Through experiments across different model sizes, we demonstrate that Finedeep consistently outperforms traditional dense models in both perplexity (PPL) and downstream benchmarks, while maintaining a comparable number of parameters. Furthermore, hyper-parameter studies reveal that optimal results are achieved when width and depth are balanced. Finally, our empirical analysis confirms that Finedeep effectively mitigates sparse activation, enhancing the overall representation capacity of the model.

The main contributions of our work are summarized as follows.

- To address the sparse activation in dense models, we propose Finedeep, which mitigates the issue by decomposing the FFN into fine-grained experts arranged across multiple sub-layers.
- We theoretically demonstrate that Finedeep is capable of mitigating the sparse activation problem.
- Through extensive experiments in LLM pre-training, we demonstrate the superiority of Finedeep over traditional dense models and empirically validate its ability to alleviate sparse activation in dense models.

2 Related Work

Current dense model architectures are predominantly based on the decoder-only transformer, which effectively leverages the parameter space to encode rich knowledge and deliver strong performance (Brown et al., 2020; Touvron et al., 2023a,b; Dubey et al., 2024). FFN layers in these models are often regarded as a key component for storing substantial amounts of knowledge (Geva et al., 2021; Dai et al., 2022). However, it has been observed that FFN layers in dense models exhibit sparse activation during the training process (Zhang et al., 2022), where the majority of the output values from the activation function are low, contributing marginally to subsequent matrix multiplications. This indicates that the activation values are not fully utilized, leading to a potential waste of resources. Moreover, the phenomenon of sparse activation becomes increasingly pronounced as the training process progresses (Luo et al., 2024).

To address the sparse activation in dense models, Zhang et al. (2022) converts dense architectures into MoE by identifying activation patterns and grouping neurons into experts to increase activation density. Yang et al. (2024b) also identifies the sparse activation problem within individual experts in the MoE architecture and mitigates this by dividing the experts into fine-grained experts. Unlike

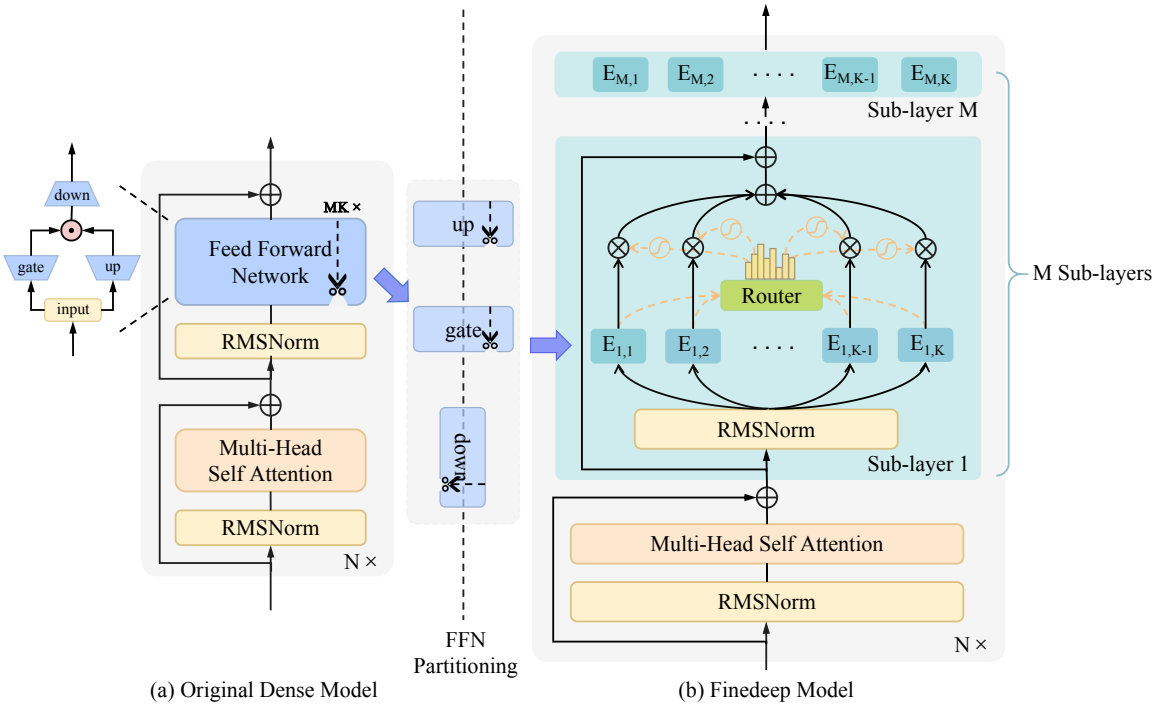

Figure 2: Illustration of the proposed Finedeep. Subfigure (a) shows the structure of the original dense model. Subfigure (b) demonstrates the structure of our proposed Finedeep model. Each FFN in the dense model is partitioned into $M \times K$ experts distributed along M sub-layers with K experts per sub-layer. The connection between subfigures (a) and (b) represents the transformation from the original dense model to the Finedeep model.

the aforementioned methods, our approach works entirely within the dense model framework. We do not adopt the common MoE strategy of activating the top-k experts to avoid sparse activation (Fedus et al., 2022; Lepikhin et al., 2021). Instead, we reduce sparse activation with all parameters contributing to the computation.

3 Methodology

The proposed method, Finedeep, is illustrated in Figure 2. To tackle the sparse activation problem commonly observed in dense models, we first decompose the FFNs into fine-grained experts. We then introduce a novel expert arrangement and routing strategy: stacking multiple sub-layers of organized experts and applying output-guided routing mechanism to weight their outputs.

3.1 FFN Partitioning

In a standard FFN layer, the input is first projected onto a higher-dimensional space through a "up" projector, activated by SiLU with an additional gating matrix, and then mapped back to the original dimension using a "down" projector, as shown in Figure 2(a) (Touvron et al., 2023b). This process can be described by the following equation:

$$\mathrm{FFN}(\hat{\mathbf{h}}_t^l) = (\sigma(\hat{\mathbf{h}}_t^l \boldsymbol{W}_g) \odot \hat{\mathbf{h}}_t^l \boldsymbol{W}_{\mathrm{up}}) \boldsymbol{W}_{\mathrm{down}} \quad (1)$$

Here, $\boldsymbol{W}_g$, $\boldsymbol{W}_{\mathrm{up}}$ and $\boldsymbol{W}_{\mathrm{down}}$ denote the gating, up and down projection matrices respectively. σ is the activation function. $\hat{\mathbf{h}}_t^l$ is the output of the Multi-Head Self-Attention (MHA) module at layer l.

We observe that sparse activation is more prominent in large-scale FFNs than in smaller ones. To address this, we decompose the FFN into smaller experts. Specifically, we first determine the number of experts to be sliced, which is given by the number of sub-layers containing experts, M, multiplied by the number of experts per sub-layer, K. We then partition the three matrices $\boldsymbol{W}_g$, $\boldsymbol{W}_{\mathrm{up}}$ and $\boldsymbol{W}_{\mathrm{down}}$ along the intermediate dimensions. This ensures that the computational logic of each expert remains consistent with that of the original FFN layer, differing only in the intermediate dimensions. For a given expert i, the computation is as follows:

$$\mathrm{FFN}_i(\hat{\mathbf{h}}_t^l) = (\sigma(\hat{\mathbf{h}}_t^l \boldsymbol{W}_g^{(i)}) \odot \hat{\mathbf{h}}_t^l \boldsymbol{W}_{\mathrm{up}}^{(i)}) \boldsymbol{W}_{\mathrm{down}}^{(i)}$$

$$\text{where} \quad 1 \leq i \leq MK \quad (2)$$

Here, $\boldsymbol{W}_g^{(i)}$, $\boldsymbol{W}_{\text{up}}^{(i)}$, and $\boldsymbol{W}_{\text{down}}^{(i)}$ are the sliced weight matrices of expert i. This decomposition allows each expert to process a distinct subspace of the input. Notably, the total parameter count remains comparable to the original FFN layer.

3.2 Expert Arrangement and Routing

To mitigate the sparse activation caused by single-layer nonlinear transformations, we adopt a **multi-layer expert arrangement** strategy. Specifically, we arrange K experts per sub-layer across a total of M sub-layers. The expert group in sub-layer j is defined as:

$$\mathrm{E}_j = \{\mathrm{FFN}_{(j-1)K+1}, ..., \mathrm{FFN}_{jK}\} \tag{3}$$

For clarity, we denote the ith expert in the jth sub-layer as:

$$\mathrm{E}_{j,i} = \mathrm{FFN}_{(j-1)K+i} \tag{4}$$

Regarding the routing approach, we propose an **output-guided sigmoid routing** mechanism. Unlike MoE architectures, where the router selects a sparse subset of experts based on the input, our approach activates all experts and computes routing weights based on their outputs. This allows for a more accurate assessment of each expert's contribution and helps reduce error propagation. Once the weight scores are obtained, we forgo the standard softmax normalization. Softmax enforces competition among experts, often amplifying sparse activation by suppressing weaker expert contributions. Instead, inspired by Liu et al. (2024), we apply a sigmoid function to nonlinearly transform the router's weights into the range $[0, 1]$. This enables each expert to contribute independently, mitigating the problem of competitive suppression.

It is important to note that since our method increases the model's depth, direct training may lead to gradient vanishing issues. To mitigate this, we employ a **sub-layer residual normalization** operation. Specifically, we apply the normalization operation before the expert inputs and perform residual connections after weighting the routing scores. Formally, the computation process in the jth sub-layer can be expressed as follows:

$$\tilde{\boldsymbol{h}}_t^{l,j} = \mathrm{RMSNorm}_j(\hat{\boldsymbol{h}}_t^{l,j-1}) \tag{5}$$

$$\boldsymbol{r}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j}) = \sigma(\mathrm{E}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j})\boldsymbol{R}_{j,i}) \tag{6}$$

$$\hat{\boldsymbol{h}}_t^{l,j} = \sum_{i=1}^{K} \boldsymbol{r}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j}) \cdot \mathrm{E}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j}) \tag{7}$$

$$\hat{\boldsymbol{h}}_t^{l,j} := \hat{\boldsymbol{h}}_t^{l,j} + \hat{\boldsymbol{h}}_t^{l,j-1} \tag{8}$$

Here $\mathrm{RMSNorm}_j$ denotes the RMSNorm module in the jth sub-layer. $\hat{\boldsymbol{h}}_t^{l,j-1}$ denotes the output of the $(j-1)$th sub-layer at time step t in layer l. The final output of the jth sub-layer expert group is given by $\hat{\boldsymbol{h}}_t^{l,j}$, while $\tilde{\boldsymbol{h}}_t^{l,j}$ represents the output of the RMSNorm module in the jth sub-layer. σ denotes the sigmoid activation function. $\boldsymbol{R}_{j,i}$ refers to the ith column of the routing matrix in the jth sub-layer. Finally, $\boldsymbol{r}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j})$ represents the routing score assigned by the router in the jth sub-layer to the ith expert.

Overall, the first expert group sub-layer takes the output of the current layer's MHA module as input and processes it according to the intra-layer computation described above. The hidden states produced by each expert group sub-layer are then sequentially passed to the next sub-layer until all expert group sub-layers have been processed.

3.3 Multi-layer Experts Enable Feature Reactivation

The multi-layer design mitigates sparse activation by enabling feature reactivation, allowing suppressed features to be recovered in deeper sub-layers. Consider the case of $M = 2$ sub-layers (which generalizes naturally to $M > 2$). Let $\hat{\boldsymbol{h}}_t^{l,j}$ denote the hidden state after the jth sub-layer at time step t in layer l. The output of the second sub-layer can be expressed as follows:

$$\hat{\boldsymbol{h}}_t^{l,2} = \sum_{i=1}^{K} \boldsymbol{r}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) + \hat{\boldsymbol{h}}_t^{l,1} \tag{9}$$

In fact, $\hat{\boldsymbol{h}}_t^{l,1}$ can be expanded into the residuals of the first sub-layer $\hat{\boldsymbol{h}}_t^{l,0}$ and the result of the weighted summation of the experts of the first sub-layer $\hat{\boldsymbol{h}}_t^{l,1}$ according to Equation 8. So Equation 9 can be expressed in the following form:

$$\hat{\boldsymbol{h}}_t^{l,2} = \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0}) + \hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0} \tag{10}$$

Using Taylor expansion with remainder Δ_1 reveals the reactivation mechanism:

$$\begin{aligned}\mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0}) =& \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) + \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,0}) \\ &+ \underbrace{\nabla \mathrm{E}_{2,i}(\xi) \cdot (\hat{\boldsymbol{h}}_t^{l,1} \odot \hat{\boldsymbol{h}}_t^{l,0})}_{\Delta_1}\end{aligned} \tag{11}$$

Here, ξ is an intermediate point between the two inputs. The cross-term Δ_1 captures interactions

between the current features $\hat{\boldsymbol{h}}_t^{l,1}$ and the earlier ones $\hat{\boldsymbol{h}}_t^{l,0}$, allowing suppressed information to be reactivated via expert-specific gradients $\nabla \mathrm{E}_{2,i}$.

Full expansion demonstrates strict generalization of single-layer:

$$
\begin{aligned}
\hat{\boldsymbol{h}}_t^{l,2} = & \underbrace{\sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,0}) + \sum_{i=1}^{K} \boldsymbol{r}_{1,i} \cdot \mathrm{E}_{1,i}(\hat{\boldsymbol{h}}_t^{l,0}) + \hat{\boldsymbol{h}}_t^{l,0}}_{\text{Conventional single-layer}} \\
& + \underbrace{\sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i} \left(\sum_{j=1}^{K} \boldsymbol{r}_{1,j} \cdot \mathrm{E}_{1,j}(\hat{\boldsymbol{h}}_t^{l,0}) \right)}_{\text{Deep reprocessing}} \\
& + \underbrace{\sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \left(\nabla \mathrm{E}_{2,i}(\xi) \cdot (\hat{\boldsymbol{h}}_t^{l,1} \odot \hat{\boldsymbol{h}}_t^{l,0}) \right)}_{\text{Feature interaction reactivation}} \\
& + \underbrace{\hat{\boldsymbol{h}}_t^{l,1}}_{\text{Residual pathway}}
\end{aligned}
\tag{12}
$$

The additional terms implement feature reactivation: 1. **Deep Reprocessing**: Re-evaluate shallow outputs that may contain suppressed features. 2. **Feature Interaction**: Reactivate via cross-feature modulation. 3. **Residual Pathway**: Preserve temporarily suppressed features for later recovery.

This structure provides exponentially more reactivation pathways than single-layer designs, directly addressing sparse activation at its root.

4 Experiments

We conducted extensive experiments across various model sizes and configurations, evaluating PPL and downstream benchmarks to validate the effectiveness of Finedeep.

4.1 Pre-training Dataset

To maximize model performance, we curated high-quality open-source pre-training datasets from various domains. For general-domain data, we utilized FineWeb-Edu dataset (Penedo et al., 2024). For mathematics and code, we followed OLMoE and incorporated OpenWebMath and StarCoder (Muennighoff et al., 2024). We also included synthetic data from Cosmopedia, which has been shown to enhance model performance (Abdin et al., 2024; Ben Allal et al., 2024). We set the total pre-training data size to 100B tokens, following best practices from related studies (Dai et al., 2024).

Before conducting pre-training experiments, we also preprocessed data for tokenization. Specifically, we utilized LLaMA 3's tokenizer, which has a vocabulary size of 128K, to tokenize the mixed dataset while enforcing a maximum sequence length of 1,024 (Dubey et al., 2024).

Model	Config	Params	Pile PPL (↓)
Small			
Standard Dense	N/A	665.37 M	14.36
Finedeep (Ours)	M=2 / K=4	665.59 M	14.28
	M=2 / K=8	665.79 M	**14.16**
	M=2 / K=16	666.18 M	14.18
Medium			
Standard Dense	N/A	1.5992 B	12.42
Finedeep (Ours)	M=2 / K=4	1.5994 B	12.24
	M=2 / K=8	1.5997 B	12.23
	M=2 / K=16	1.6002 B	12.24
	M=4 / K=4	1.6002 B	**12.13**
	M=8 / K=2	1.5999 B	12.17
Large			
Standard Dense	N/A	7.5269 B	10.15
Finedeep (Ours)	M=2 / K=8	7.5292 B	**10.08**

Table 1: Perplexity results for models with different configurations. The best results are highlighted in **bold**. M denotes the number of sub-layers in the expert arrangement, K represents the number of experts per sub-layer.

4.2 Experimental Setup

Following Biderman et al. (2023) and Su et al. (2024), we pre-trained three model sizes: Small (665M), Medium (1.6B), and Large (7.5B). To evaluate the effectiveness of our method, we conducted experiments with varying numbers of sub-layers and experts per sub-layer.

For the evaluation, we conducted both PPL and benchmark evaluations. For the PPL evaluation, we followed the approach outlined by Dai et al. (2024), testing the perplexity of the model on the pile test set. In terms of benchmark evaluations, we performed discriminative and generative tasks, reporting 0-shot results for discriminative tasks and 5-shot results for generative tasks. The benchmarks we collected cover a broad range of domains to assess various aspects of the model's capabilities.

4.3 Perplexity Results

As shown in Table 1, our proposed method significantly outperforms standard dense models in terms of PPL on the pile test set across various model scales, while maintaining a comparable number of parameters. Specifically, for the optimal choice of the number of experts per sub-layer, we find that in the Small model setup, when the number of expert sub-layers is kept constant, the configuration

Model	SQuAD	LAMBADA	ARC	HellaSwag	PIQA	SIQA	Wino	NaturalQs	TriviaQA	AVG
Small										
Standard Dense	6.22	41.14	33.87	49.79	70.78	41.15	54.14	7.04	20.79	36.10
Finedeep M=2/K=4	7.34	**42.23**	34.13	50.44	70.51	40.23	55.01	6.79	**21.79**	36.50
Finedeep M=2/K=8	**9.53**	42.01	**36.09**	50.58	71.22	40.79	**56.27**	6.51	21.54	**37.17**
Finedeep M=2/K=16	6.89	41.76	33.79	**50.60**	**71.87**	**41.20**	55.64	**7.04**	21.25	36.67
Medium										
Standard Dense	7.16	46.52	38.99	56.45	73.67	42.43	56.91	8.56	28.25	39.88
Finedeep M=2/K=4	**15.65**	46.59	39.68	57.52	72.96	41.50	56.35	9.28	29.43	41.00
Finedeep M=2/K=8	14.95	**48.30**	39.93	57.49	**74.10**	**43.60**	56.83	8.53	28.99	**41.41**
Finedeep M=2/K=16	14.76	47.99	39.68	57.24	73.45	42.17	56.99	8.81	29.39	41.16
Finedeep M=4/K=4	12.22	47.80	**40.19**	**58.11**	73.72	42.48	59.19	8.23	**30.20**	41.35
Finedeep M=8/K=2	12.64	48.19	38.91	57.29	73.99	41.97	**59.27**	**9.78**	29.98	41.34
Large										
Standard Dense	19.50	55.00	**46.93**	66.05	76.28	43.50	62.19	13.74	42.26	47.27
Finedeep M=2/K=8	**19.92**	**56.26**	45.90	**66.25**	**76.99**	**43.86**	**62.43**	**14.27**	**43.33**	**47.69**

Table 2: Benchmark results for models with different configurations. The best results are highlighted in **bold**, while the second-best results are underlined. Here, M denotes the number of sub-layers, K represents the number of experts per sub-layer. The AVG metric represents the average of the different benchmark results.

with 8 experts per sub-layer outperforms the configurations with 4 or 16 experts per sub-layer. A similar trend was observed in the Medium model setup. This suggests that appropriately increasing the number of experts per sub-layer can enhance model performance, as a sufficient number of experts allows the sub-layer to capture more complex features. However, excessive increases in the number of experts can reintroduce the sparse activation problem, leading to inefficient activation utilization and diminished performance.

Regarding the optimal choice of the number of expert sub-layers, we find that in the Medium setup, increasing the number of expert sub-layers from 2 to 4, while keeping the total number of experts constant, enhances model performance. However, further increasing the number of sub-layers from 4 to 8 results in a performance drop. This indicates that while increasing the number of expert sub-layers can benefit model performance, excessive depth in the model can be detrimental. The underlying reason is analogous to the earlier observation, as keeping a fixed total number of experts, too many sub-layers will result in fewer experts per sub-layer, and too few sub-layers will concentrate too many experts in each sub-layer.

In summary, we aim to strike a balance between width and depth in the expert arrangement process.

4.4 Benchmark Results

As shown in Table 2, our method outperforms the traditional dense model across a range of benchmarks covering multiple domains. We observe that the AVG metrics for these benchmarks follow a similar trend to the PPL metrics. Specifically, configurations with 2 expert sub-layers and 8 experts per layer, or 4 expert sub-layers and 4 experts per layer, yield the best performance. This reinforces the conclusion that our method achieves optimal results when there is a balanced trade-off between width and depth.

5 Analysis

5.1 Ablation Study

To further demonstrate the necessity of splitting multiple experts per sub-layer and arranging multiple sub-layers, we conducted ablation experiments using the Medium size model. Experimental results are presented in Table 3.

First, we validated the necessity of arranging multiple experts within each sub-layer. Specifically, we set the number of sub-layers to 2 and assigned only one expert per sub-layer. Notably, since there was only one expert per layer in this setup, we removed the router responsible for assigning weights to each expert. The results show that this configuration performs worse than our method in terms of both PPL and benchmark evaluations. This highlights the importance of arranging multiple experts within each sub-layer.

Furthermore, we verified the necessity of using multiple expert sub-layers. In this experiment, we set the number of sub-layers to 1 while assigning

	w/o multi experts M=2/K=1	w/o multi sub-layers M=1/K=16	Finedeep M=2/K=8
PPL (↓)	12.42	12.42	**12.23**
SQuAD	10.11	12.65	**14.95**
LAMBADA	46.48	45.06	**48.30**
ARC	39.08	38.65	**39.93**
HellaSwag	56.58	56.46	**57.49**
PIQA	73.50	72.96	**74.10**
SIQA	41.45	40.74	**43.60**
Wino	57.30	**57.38**	56.83
NaturalQs	**9.11**	8.39	8.53
TriviaQA	28.61	28.61	**28.99**
AVG	40.25	40.10	**41.41**

Table 3: Ablation experiment results on the impact of multiple experts per sub-layer and multiple sub-layers.

	input-guided M=2/K=8	output-guided M=2/K=8
PPL (↓)	12.24	**12.23**
SQuAD	14.27	**14.95**
LAMBADA	47.97	**48.30**
ARC	38.82	**39.93**
HellaSwag	57.21	**57.49**
PIQA	73.23	**74.10**
SIQA	41.50	**43.60**
Wino	**57.93**	56.83
NaturalQs	**9.17**	8.53
TriviaQA	**29.85**	28.99
AVG	41.11	**41.41**

Table 4: Comparison of experimental results for input-guided and output-guided routing.

16 experts within that single sub-layer. The results indicate that this setup also leads to suboptimal performance, further emphasizing the importance of arranging multiple sub-layers.

Additionally, these two ablation studies reinforce our conclusion that achieving a balance between the number of experts per sub-layer and the number of sub-layers leads to optimal results. Both experimental configurations represent extreme imbalances, which result in poor performance.

5.2 Routing Scores Computation: Input-guided vs. Output-guided

In Finedeep, we adopt an output-guided routing strategy, where the routing weights are computed based on the experts' outputs and used to aggregate their contributions. This approach is inspired by traditional MoE architectures, which typically compute routing weights based on the inputs of each expert (Fedus et al., 2022). However, in MoE models, it is not feasible to use expert outputs for routing, since each token is only routed to a subset of experts and does not access all expert outputs. In contrast, Finedeep operates within a dense architecture, allowing every token to pass through all experts. This enables the use of output-based routing. As shown in Table 4, we compare the performance of the output-guided and input-guided routing strategies. The results demonstrate that output-guided routing achieves better performance in our framework. This is because the expert outputs directly reflect how well each expert handles the input, allowing for more accurate and dynamic weighting during aggregation, while also reducing the risk of error propagation.

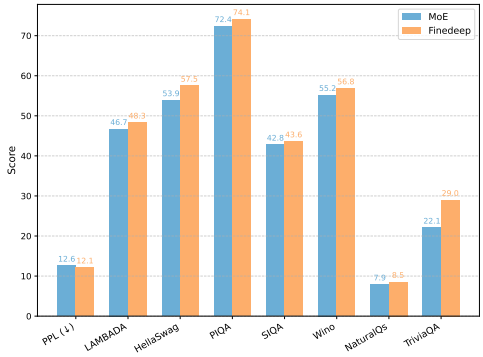

Figure 3: Comparison of experimental results for MoE and Finedeep architectures. Finedeep is configured with 16 experts in total, arranged as 2 layers of 8 experts each. The sparse MoE baseline also contains 16 experts overall, but only the top-8 experts are activated per layer, resulting in 1.2B activated parameters. Both models are trained on the same dataset with 100B tokens.

5.3 Comparison with MoE architectures

We further compare Finedeep with MoE architectures. As shown in Figure 3, Finedeep consistently outperforms the sparse MoE baseline across all evaluated benchmarks and also achieves a lower perplexity on the validation dataset. Furthermore, the setting of M=1/K=16 in Table 3 can be viewed as a dense-activated MoE variant (Muqeeth et al., 2024), where all experts are activated and the total number of active parameters matches the model size. Under an equivalent parameter budget, Finedeep also shows stronger performance than this variant.

In summary, these results show that under the same total parameter budget, Finedeep outperforms MoE, demonstrating the effectiveness of its structural design.

5.4 Mitigating Sparse Activation with Finedeep

To better illustrate that our approach mitigates the sparse activation problem, we introduce a metric called NSAR (i.e., **N**on-**S**parse **A**ctivation **R**ate), defined as follows:

$$\mathrm{NSAR}_{\tau} = \frac{\sum_{i,j} \mathbb{I}\left(|\mathbf{A}_{i,j}| > \tau\right)}{B \times H}$$
$$\text{where} \quad \mathbb{I}\left(|\mathbf{A}_{i,j}| > \tau\right) = \begin{cases} 1, & \text{if } |\mathbf{A}_{i,j}| > \tau \\ 0, & \text{else} \end{cases} \tag{13}$$

Here, $\mathbf{A}$ represents the activation matrix of a model layer, B is the batch size, H denotes the number of neurons, and τ is a predefined threshold. This metric directly reflects the effective utilization of neurons: a lower NSAR value indicates a more severe sparse activation problem and less efficient use of activation values by the model; conversely, a higher NSAR value suggests denser activations and more effective utilization of activation values.

In Figure 4, we visualize the NSAR metric across different model layers. Specifically, we identify two key findings:

- **Fine-grained experts alleviate sparse activation**: All sub-layers exhibit significantly higher NSAR values compared to the baseline model, demonstrating that fine-grained expert decomposition effectively disrupts the global sparsity pattern of conventional FFNs. Each expert focuses on a local feature subspace, reducing the likelihood of simultaneous neuron suppression.

- **Multi-layer expert architecture mitigates sparse activation**: The NSAR metric in the second sub-layer is higher than in the first, attributed to the synergy of deep reprocessing, feature interaction, and the residual pathway. These mechanisms help revive features that were suppressed in earlier layers.

Furthermore, the mitigation of sparse activation directly enhances the model's representational capacity, as evidenced by the t-SNE visualization in Figure 6. In the baseline model, activations exhibit a contracted clustering pattern, indicating limited feature expressiveness. In contrast, the activations of the Finedeep model display broader coverage and more distinct cluster separation. This distribution pattern suggests improved feature disentanglement and an expanded representation space, enabling the model to capture finer-grained feature variations and thereby laying the foundation for its performance improvements.

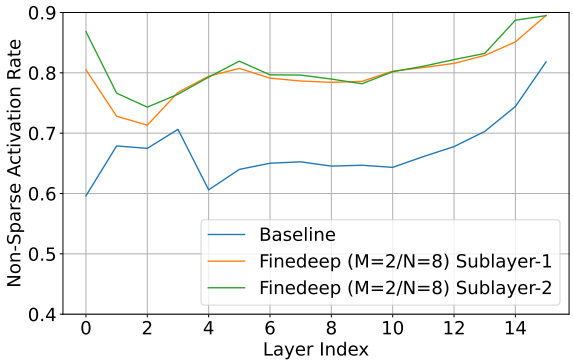

Figure 4: Variation of $\mathrm{NSAR}_{0.1}$ metrics across different model layers.

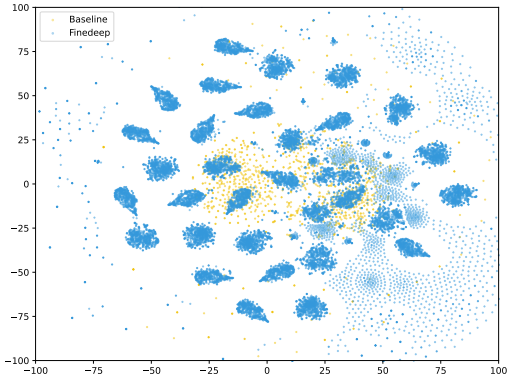

Figure 5: T-SNE clustering of activation values in the traditional dense model and the Finedeep model.

6 Conclusion

To address the sparse activation issue observed in existing dense LLMs, we propose Finedeep, a novel dense architecture. Finedeep mitigates sparse activation by decomposing the standard FFN layer into multiple experts and arranging them across multiple sub-layers. To better integrate the outputs of fine-grained experts, we introduce an output-based sigmoid routing strategy. We conduct extensive experiments across multiple model sizes, and the PPL and benchmark results demonstrate that our method significantly outperforms existing dense architectures with identical parameter counts. Additionally, we find that the model performs optimally when the number of expert sub-layers and the number of experts per sub-layer are balanced. Finally, we empirically demonstrate that Finedeep effectively mitigates sparse activation, thereby enhancing the model's representational capacity.

Limitations

Due to computational resource constraints, we trained all model configurations on only 100B tokens and did not explore the impact of training on a larger token budget. Additionally, our largest model size was limited to 7.5B parameters, leaving the potential benefits of scaling to larger models unexplored. Furthermore, while our approach mitigates sparse activation in dense models, we believe there is still room for further improvement. We leave this for our future research.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. arXiv preprint arXiv:2412.08905.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. Cosmopedia.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 7432–7439.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

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. CoRR, abs/1803.05457.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 1280–1297. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8493–8502.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120):1–39.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5484–5495.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. CoRR, abs/2203.15556.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. CoRR, abs/2401.04088.

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 (Volume 1: Long Papers), pages 1601–1611.

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. CoRR, abs/2001.08361.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, and 48 others. 2023. Starcoder: may the source be with you! Trans. Mach. Learn. Res., 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437.

Yuqi Luo, Chenyang Song, Xu Han, Yingfa Chen, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2024. Sparsing law: Towards large language models with greater activation sparsity. arXiv preprint arXiv:2411.02335.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. CoRR.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. 2024. Soft merging of experts with adaptive routing. Trans. Mach. Learn. Res., 2024.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1525–1534.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2024. Openwebmath: An open dataset of high-quality mathematical web text. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. arXiv preprint arXiv:2406.17557.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 784–789.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8732–8740.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4463–4473.

Zhenpeng Su, Zijia Lin, Baixue Baixue, Hui Chen, Songlin Hu, Wei Zhou, Guiguang Ding, and W Xing. 2024. Mile loss: a new loss for mitigating the bias of learning difficulties in generative language models. In Findings of the Association for Computational Linguistics: NAACL 2024, pages 250–262.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. CoRR, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. 2024b. Xmoe: Sparse models with fine-grained and adaptive expert selection. In Findings of the Association for

Computational Linguistics ACL 2024, pages 11664–11674.

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 Annual Meeting of the Association for Computational Linguistics, pages 4791–4800.

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

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Moefication: Transformer feed-forward layers are mixtures of experts. In Findings of the Association for Computational Linguistics: ACL 2022, pages 877–890.

A Appendix

A.1 Activation Clustering

Our method enhances the utilization of activation values by addressing the sparse activation phenomenon, thereby expanding the representation space of these values and boosting the model's overall representational capacity. This is the key reason why our approach outperforms traditional dense models. To demonstrate how our method expands the activation value representation space, we apply t-SNE dimensionality reduction to the activation values of both the traditional dense model and the model trained using our method, as shown in Figure 6.

We select the activation representations of 500 mid-frequency words for dimensionality reduction. Specifically, our method utilizes a setup with two expert sub-layers, each containing eight experts. To ensure a fair comparison, we concatenate the activation values from different experts in the first and second sub-layers before performing dimensionality reduction, keeping the number of sample points consistent with the baseline model. As shown in the figure, our method covers a broader representation space across different layers, making the token representations more discriminative and better separated. This enhanced separation in the representation space indicates that our model can better distinguish between different semantic concepts and capture more nuanced relationships between tokens, which directly explains its superior performance compared to traditional dense models.

A.2 Mix Ratios of Different Pre-training Datasets

Referring to technical reports from other open-source models and the dataset sizes we collected from various domains, we finalized the data mixing ratios for each domain, as shown in Table 5.

Domain	Ratio
Cosmopedia	3.18%
Fineweb-Edu	86.31%
OpenWebMath	1.38%
StarCoder	9.13%

Table 5: Mixing ratios of pre-training data across different domains.

A.3 Evaluation Benchmarks

To comprehensively assess the performance of our method, we evaluate across a diverse set of benchmarks covering various aspects. These benchmarks include tasks related to reading comprehension, language understanding, commonsense reasoning, and closed-book question answering.

- Reading comprehension: We evaluate our method on SQuAD V2 (Rajpurkar et al., 2018), which tests the ability to answer questions based on given passages.
- Language understanding: We use LAMBADA (Paperno et al., 2016), a benchmark that requires models to predict the final word of a sentence, assessing long-range context understanding.
- Commonsense reasoning: We include ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), and Winogrande (Sakaguchi et al., 2020), which test the model's ability to infer commonsense knowledge across various scenarios.
- Closed-book question answering: We assess factual knowledge recall using Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017), where models must generate correct answers without relying on external documents.

A.4 Training Configuration

	Small	Medium	Large
Hidden Size	1024	2048	4096
Intermediate Size	4096	8192	11008
Attention Heads	16	8	32
Layers	24	16	32
Learning Rate	3e-4	3e-4	3e-4
Weight Decay	0.1	0.1	0.1
RMSNorm Epsilon	1e-05	1e-05	1e-05

Table 6: Experimental training configuration.

A.5 Training Dynamics: Validation Perplexity Curve

Figures 7, 8, and 9 illustrate that across all model sizes, our method maintains consistently lower validation perplexity than the baseline throughout training, highlighting its robustness and scalability. Notably, the advantage becomes more pronounced in

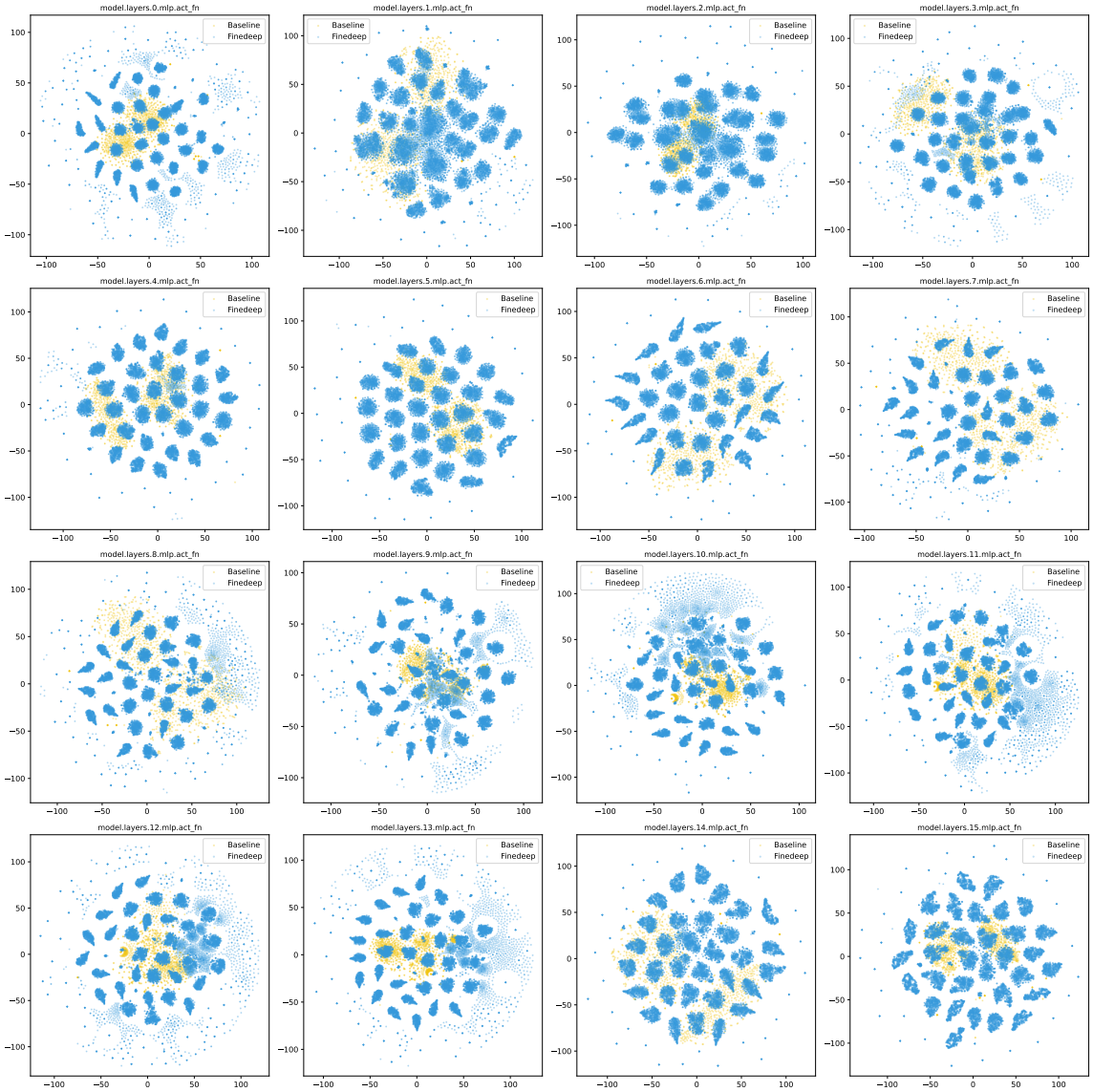

Figure 6: T-SNE clustering of activation values from different layers in the traditional dense model and the model trained with the Finedeep method.

later training stages, indicating that Finedeep not only performs well in the early phases but also demonstrates strong scalability and convergence as training progresses.

A.6 Pre-training Dataset Collection Details

We curated training data from a variety of domains to improve the generalization and reasoning capabilities of our model.

- FineWeb-Edu: This is a filtered subset of the FineWeb dataset, selected using an educational quality classifier to retain high-value educational web content (Penedo et al., 2024).
- OpenWebMath: Adopted from the OLMoE pipeline (Muennighoff et al., 2024), this

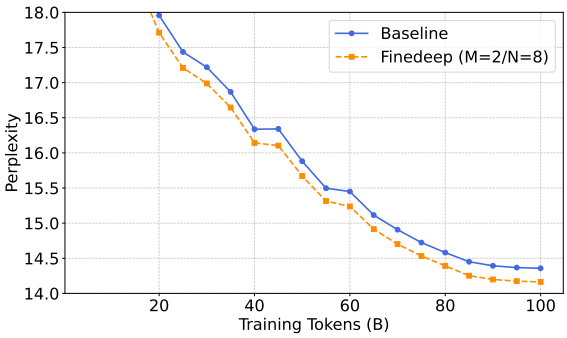

Figure 7: Validation perplexity trends of Finedeep and baseline during training in Small size model.

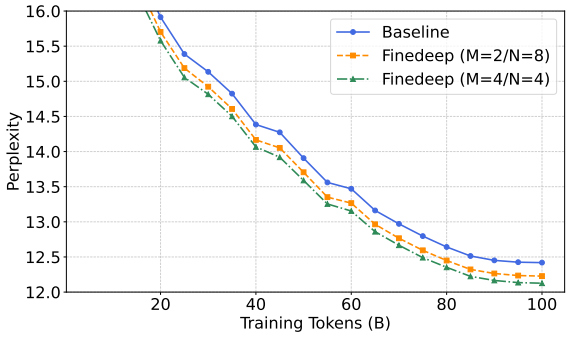

Figure 8: Validation perplexity trends of Finedeep and baseline during training in Medium size model.

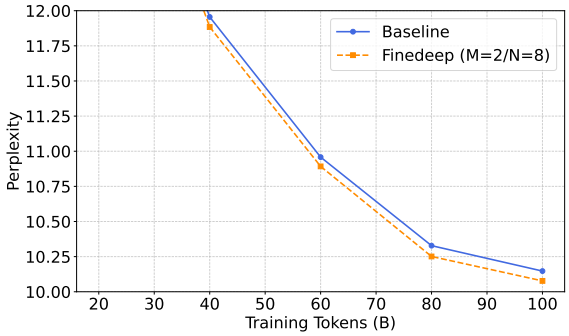

Figure 9: Validation perplexity trends of Finedeep and baseline during training in Large size model.

dataset contains mathematical texts filtered from Common Crawl with heuristics designed to preserve high-quality mathematical reasoning and content (Paster et al., 2024).

- StarCoder: A diverse dataset of programming languages, GitHub issues, and Jupyter Notebooks, cleaned through a rigorous data filtering process to ensure relevance and quality (Li et al., 2023).
- Cosmopedia: A synthetic dataset composed of textbooks, stories, blog posts, and instructional content (e.g., WikiHow-style articles), covering a wide range of topics (Ben Allal et al., 2024).

A.7 Routing Scores Normalization: Sigmoid vs. Softmax

Our proposed method computes the final routing score by applying a sigmoid function to the router's output, whereas some other approaches normalize expert outputs using the softmax function (Jiang et al., 2024). We compared these two approaches for normalizing routing scores using the Medium model, and experimental results presented in Table 7 demonstrate that the sigmoid-based routing in Finedeep achieves superior performance. This improvement can be attributed to the fact that softmax enforces competition among experts, whereas sigmoid allows each expert to contribute independently. As a result, the sigmoid method reduces unnecessary competition, leading to a more balanced utilization of model capacity. Given that our approach activates all expert parameters, maintaining this balance is particularly crucial for maximizing performance.

	softmax M=2/K=8	sigmoid M=2/K=8
PPL (↓)	12.27	**12.23**
SQuAD	**14.99**	14.95
LAMBADA	47.04	**48.30**
ARC	39.85	**39.93**
HellaSwag	56.80	**57.49**
PIQA	72.63	**74.10**
SIQA	42.37	**43.60**
Wino	**57.30**	56.83
NaturalQs	**8.67**	8.53
TriviaQA	27.94	**28.99**
AVG	40.84	**41.41**

Table 7: Experimental results comparing the Softmax and Sigmoid methods for normalizing routing scores.

	GFLOPs	Training Throughput
Standard Dense	1801.00	3903
Finedeep M=2/K=8	1801.46	3237

Table 8: Comparison of GFLOPs and training throughput between the standard dense architecture and Finedeep under the 7.5B Large model setting. Throughput is measured in tokens processed per GPU per second. GFLOPs (Giga floating-point operations) are calculated by processing input samples with a batch size of 1 and a sequence length of 128.

A.8 Training Throughput

Table 8 presents a comparison of training throughput between the standard dense architecture and Finedeep on the scale of the 7.5B model. To ensure a fair comparison, both models were trained using the same configuration: 8 H100 nodes with ZeRO parallelism. Specifically, we employed DeepSpeed ZeRO Stage 3 with offloading, setting the batch size per GPU to 8, the gradient accumulation steps to 4, and the global batch size to 2048.

As shown in the table, the training throughput

Layer	0	1	2	3	4	5	6	7
Baseline	0.7956	0.8402	0.8382	0.8520	0.7868	0.8124	0.8176	0.8177
Finedeep Sublayer-1	0.9036	0.8615	0.8544	0.8836	0.8949	0.9027	0.8900	0.8920
Finedeep Sublayer-2	0.9376	0.8848	0.8667	0.8817	0.8949	0.9102	0.8975	0.9007
Layer	8	9	10	11	12	13	14	15
Baseline	0.8143	0.8159	0.8147	0.8227	0.8335	0.8475	0.8673	0.9106
Finedeep Sublayer-1	0.8923	0.8942	0.9027	0.9042	0.9085	0.9133	0.9238	0.9504
Finedeep Sublayer-2	0.8990	0.8913	0.9003	0.9040	0.9103	0.9190	0.9447	0.9495

Table 9: Layer-wise NASR values for baseline and Finedeep `M=2/K=8` with $\tau = 0.05$.

Layer	0	1	2	3	4	5	6	7
Baseline	0.4102	0.5201	0.5125	0.5608	0.4618	0.4899	0.5022	0.5091
Finedeep Sublayer-1	0.7006	0.5981	0.5871	0.6552	0.6932	0.7063	0.6934	0.6784
Finedeep Sublayer-2	0.7854	0.6422	0.6276	0.6551	0.6921	0.7255	0.7002	0.6884
Layer	8	9	10	11	12	13	14	15
Baseline	0.4955	0.4976	0.5001	0.5136	0.5393	0.5700	0.6283	0.7216
Finedeep Sublayer-1	0.6740	0.6785	0.7023	0.7094	0.7188	0.7389	0.7740	0.8407
Finedeep Sublayer-2	0.6830	0.6709	0.6990	0.7132	0.7283	0.7468	0.8303	0.8310

Table 10: Layer-wise NASR values for baseline and Finedeep `M=2/K=8` with $\tau = 0.15$.

of Finedeep is slightly lower than that of the standard dense model, but remains within an acceptable range. This is mainly due to the introduction of additional expert sub-layers in Finedeep, which increases model depth and results in more sequential computation. However, it is important to note that our implementation is built upon DeepSpeed and the Transformers library, without operator-level optimizations or advanced parallel strategies. We believe that the gap in throughput can be further reduced with future improvements in operator efficiency and parallelism techniques. Meanwhile, this paper focuses on the algorithmic feasibility of our method, and leaves system-level optimization of training efficiency to future work.

A.9 FLOPs Comparison Between Finedeep and Baseline

We compute the floating point operations (FLOPs) of our proposed method and compare them with those of the traditional dense architecture, as shown in Table 8. Although our method includes additional components such as the router module and RMSNorm, their contribution to the overall computation is minimal because they account for only a small portion of the total model parameters. Our calculations indicate that across different model scales, our approach increases FLOPs by 0.03% compared to the traditional dense model, which is an almost negligible difference. Despite maintaining nearly the same FLOPs as the dense baseline, our method achieves significantly better performance, further demonstrating its effectiveness. From another perspective, this also suggests that there is potential for further optimization in training throughput.

A.10 Robustness Analysis of NASR Thresholds

In this section, we analyze the robustness of the Non-Sparse Activation Rate (NASR) metric introduced in the paper, focusing on how the choice of threshold τ affects its measurement. In the main text, we set $\tau = 0.1$, considering neurons with activations below this value as sparsely activated. This threshold was determined empirically based on activation distributions across different model layers, and it serves to quantify how effectively Finedeep mitigates sparse activation.

To evaluate the sensitivity of NASR to the threshold choice, we additionally report results for $\tau = 0.05$, $\tau = 0.15$, and $\tau = 0.20$. Tables 9, 10, and

Layer	0	1	2	3	4	5	6	7
Baseline	0.2509	0.3701	0.3520	0.4166	0.3368	0.3534	0.3682	0.3776
Finedeep Sublayer-1	0.5726	0.4805	0.4715	0.5469	0.5931	0.6054	0.5913	0.5628
Finedeep Sublayer-2	0.6673	0.5154	0.5171	0.5495	0.5900	0.6270	0.5976	0.5752
Layer	8	9	10	11	12	13	14	15
Baseline	0.3654	0.3667	0.3744	0.3890	0.4120	0.4462	0.5127	0.6189
Finedeep Sublayer-1	0.5594	0.5601	0.5900	0.5981	0.6169	0.6380	0.6906	0.7797
Finedeep Sublayer-2	0.5730	0.5581	0.5905	0.6057	0.6312	0.6512	0.7656	0.7576

Table 11: Layer-wise NASR values for baseline and Finedeep `M=2/K=8` with $\tau = 0.20$.

11 present the layer-wise NASR values for both the baseline and Finedeep models. For Finedeep, values are reported separately for the two sublayers in the multi-layer expert arrangement (M=2/K=8).

These results show that, across all tested threshold values, Finedeep consistently achieves higher NASR than the baseline. This demonstrates that the effectiveness of Finedeep in alleviating sparse activation is robust to different choices of τ and not sensitive to the specific threshold used. In other words, the proposed method reliably increases the fraction of active neurons, highlighting its ability to mitigate sparse activation under various criteria.