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 and floating-point operations. 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; Touvron et al., 2023b; Dubey et al., 2024; Yang et al., 2024a). 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).

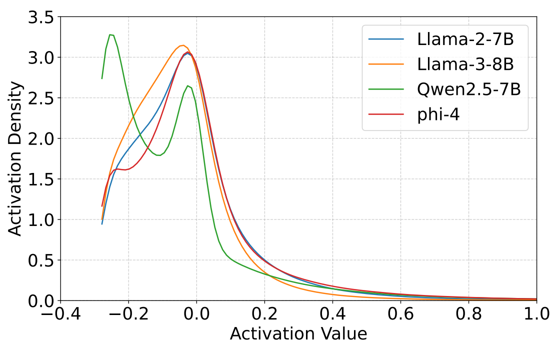

Figure 1: Distribution of activation function outputs across various models (Touvron et al., 2023b; Dubey et al., 2024; Yang et al., 2024a; Abdin et al., 2024), where all selected models use the SiLU activation function. The horizontal axis represents the activation values, while the vertical axis denotes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

Along with model scaling, recent studies disclose that dense models usually exhibit a sparse activation phenomenon during computation (Zhang et al., 2022), as illustrated in Figure 1. Specifically, sparse activation refers to the fact that most values output from the activation functions tend to be close to zero (Li et al.; 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 hence propose Finedeep, a new dense architecture with deep-layered fine-grained experts to mitigate sparse activation. Our framework partitions the feed-forward networks (FFNs) of a traditional dense model into fine-grained experts. Unlike previous Mixture-of-Experts (MoE) architectures that employ a single-layer expert arrangement, we adopt a multi-layer expert arrangement. While each layer has fewer total parameters, it achieves a higher magnitude of parameter activation. This design results in a richer representational space (Su et al., 2024b).

We further propose a novel routing strategy to efficiently combine depth-wise experts. Instead of computing routing scores based on expert inputs, as done in conventional approaches, we leverage expert outputs to determine routing scores. The final expert outputs are then combined using a soft-weighted summation. Previous soft-weighted summation often suffers from competition among different experts, as softmax normalization forces a probability distribution where a few experts dominate, leading to imbalanced contributions. To mitigate this, we build upon the insights from Liu et al. (2024) and use the sigmoid function to replace the traditional softmax function for routing score normalization.

It is important to highlight that our approach differs from the MoE framework. In our method, all parameters are actively involved in the computational process. Our goal is to increase the activation rate of all parameters and ensure that the parameters of all experts are fully utilized, rather than activating only a subset of experts to save computation, as is done in the MoE architecture. Furthermore, our approach does not introduce additional parameters; instead, it disassembles the original FFN into fine-grained experts, avoiding the need to expand parameters and consequently increasing the total model size as seen in MoE.

To validate the effectiveness of Finedeep, we conduct extensive LLM pre-training experiments. Through experiments across different model sizes and varying numbers of fine-grained experts, 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 overall model representation capacity.

The main contributions of our work are summarized as follows.

- To address the issue of sparse activation in dense models, we propose Finedeep, which partitions FFNs in dense models into fine-grained experts.

- Our approach introduces innovative expert arrangements and routing strategies, enhancing model performance while improving the stability of deep networks.

- 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 issue in dense models, Zhang et al. (2022) transforms the dense model into a MoE architecture. First, the pattern of sparse activation in the dense model is identified, and then experts are partitioned based on this pattern to maximize the activation density within each expert. 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 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

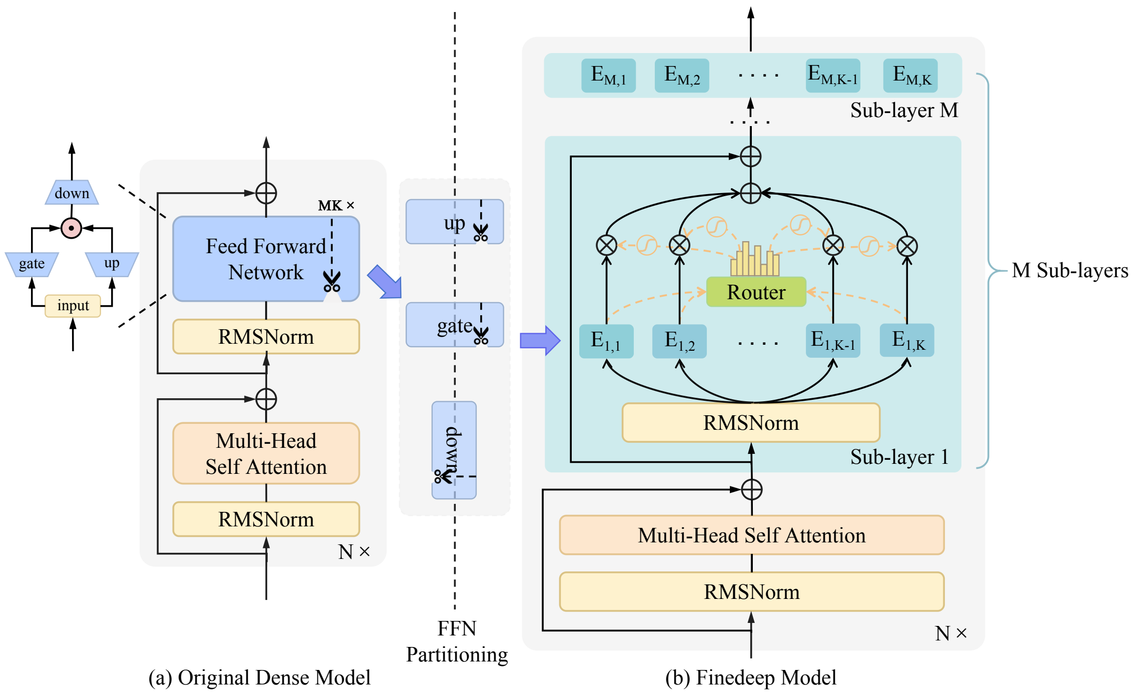

Figure 2: Illustration of the proposed Finedeep. Subfigure (a) illustrates the structure of the original dense model. Subfigure (b) demonstrates the structure of our proposed Finedeep model. Each FFN in the dense model is partitioned into $M \times K$ experts distributed along M sub-layers with K experts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

reduce sparse activation with all parameters contributing to the computation.

3 Methodology

The proposed method, Finedeep, is illustrated in Figure 2. First, to address the issue of sparse activation that commonly arises in dense models, we decompose FFNs of traditional dense models into fine-grained experts. Then, we present a novel expert arrangement and routing strategy: expanding the model depth by stacking multiple sub-layers of arranged experts and applying nonlinear routing to weight the expert outputs.

3.1 FFN Partitioning

In a traditional FFN layer, forward propagation involves projecting the input into an intermediate representation with a different dimension via an "up" projector before mapping it back to the representation with the original dimension via a "down" projector, as illustrated in Figure 2(a). Notably, most modern models employ the SiLU activation function, thus incorporating an additional gating matrix (Touvron et al., 2023b). The operations within the FFN layer are typically represented by the following equation:

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

where $\boldsymbol{W}_g$ is the gating matrix, $\boldsymbol{W}_{\mathrm{up}}$ is the projection matrix that expands the feature dimensions, while $\boldsymbol{W}_{\mathrm{down}}$ maps the features back to the original space. $\boldsymbol{W}_g$, $\boldsymbol{W}_{\mathrm{up}}$, and $\boldsymbol{W}_{\mathrm{down}}$ correspond to the gate, up, and down matrices of the FFN in the subfigure (a) of Figure 2, respectively. The function σ represents the activation function, introducing non-linearity to enhance the model's expressiveness. $\hat{\mathbf{h}}_t^l$ denotes the output of the Multi-Head Self Attention (MHA) module at the lth layer.

To mitigate the sparse activation phenomenon observed in dense models, we decompose the FFN into smaller expert units. 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:

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

$$\text{where} \quad 1 \le i \le MK \tag{2}$$

where $\boldsymbol{W}_g^{(i)}$, $\boldsymbol{W}_{\mathrm{up}}^{(i)}$ and $\boldsymbol{W}_{\mathrm{down}}^{(i)}$ represent the sliced weight matrices corresponding to expert i. This decomposition allows each expert to independently process a subset of the input space. Notably, when all experts are combined, the overall parameter scale remains comparable to that of the original FFN layer.

3.2 Expert Arrangement and Routing

In terms of expert arrangement, we adopt a **multi-layer expert arrangement** strategy. Specifically, after decomposing the FFN layer into fine-grained experts, we arrange these experts in multiple sub-layers, placing K experts per sub-layer across a total of M sub-layers. In fact, the reason we adopt a multi-layer arrangement of experts instead of a single-layer arrangement is that the single-layer expert arrangement is a special case of the multi-layer arrangement in terms of function space. The proof of this can be found in Appendix A.1. The choice to maintain a fixed number of K experts per sub-layer, given a fixed total number of experts, is intended to enhance representational diversity. Formally, the expert group in sub-layer j is defined as:

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

This structural design of multi-layer expert arrangement effectively increases the model's depth, allowing it to capture more complex features. 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 the MoE architecture, where the router processes the input to determine expert selection, our method operates within a dense framework, meaning all experts are always activated. Given this, we compute weight scores based on expert outputs rather than inputs, allowing for more precise routing. Since all expert outputs are available, this approach ensures a more accurate assessment of their contributions. 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 $[0, 1]$. It allows each expert to contribute independently rather than being normalized in a competitive manner. This helps mitigate excessive sparsity while maintaining flexibility in expert activation.

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, to prevent gradient vanishing during training and improve training stability, we add RMSNorm and residual connection operations between sub-layers. 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 $\mathrm{RMSNorm}_j$ represents 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 the lth layer. Similarly, 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. The function 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.

4 Experiments

We conducted extensive experiments across various model sizes and configurations, evaluating perplexity results and downstream benchmarks to validate the effectiveness of our proposed Finedeep approach.

4.1 Pre-training Dataset

To maximize the performance of our trained models, we curated high-quality open-source pre-training datasets from various domains. For general-domain data, we collected the FineWeb-Edu dataset, a subset of the FineWeb dataset, which was refined using an educational quality classifier to filter and extract a large volume of high-value educational web content (Penedo et al., 2024). In the domains of mathematics and code, we followed the OLMoE model to gather the OpenWebMath and StarCoder datasets (Muennighoff et al., 2024). OpenWebMath consists of high-quality mathematical text filtered and extracted from Common Crawl (Paster et al., 2024). StarCoder includes a diverse range of programming languages, GitHub issues, and Jupyter Notebook data, undergoing a rigorous data filtering process to ensure quality (Li et al., 2023). Additionally, synthetic data has been shown to enhance model performance (Abdin et al., 2024). To leverage this, we incorporated the Cosmopedia dataset, which consists of synthetic textbooks, blog posts, stories, posts and WikiHow articles, covering a wide range of topics (Ben Allal et al., 2024).

After collecting pre-training data from various domains, we mixed them according to the mix ratios in Appendix A.3. Our mixing strategy was informed by technical reports from other open-source models, as well as the dataset sizes we gathered across different domains. Given computational resource constraints, we set the total pre-training data size to 100B tokens, following best practices from related studies (Dai et al., 2024; Su et al., 2024b; Xie et al., 2023).

Before conducting pre-training experiments, we also preprocessed the 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).

4.2 Experimental Setup

Following the studies by Biderman et al. (2023) and Su et al. (2024a), we conducted pre-training experiments with three model configurations: *Small*, *Medium* and *Large*. The *Small* model setup consists of 665M parameters, the *Medium* model setup has 1.6B parameters, and the *Large* model setup includes 7.5B parameters. Specific training configurations are detailed in the Appendix A.5.

To evaluate the effectiveness of our method, we conducted experiments with varying numbers of sub-layers and experts per sub-layer. Specifically, in the *Small* model setup, we fixed the number of expert sub-layers to 2 and conducted experiments with 4, 8, and 16 experts per sub-layer. The same experimental configuration was applied to the *Medium* model. Additionally, we performed experiments where the total number of experts was fixed at 16, while varying the number of expert sub-layers to 4 and 8, respectively. For the *Large* model, we maintained the total number of experts at 16 and set the number of expert sub-layers to 2, demonstrating the effectiveness of our proposed method.

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 model's perplexity on the pile test set. In terms of benchmark evaluations, we used the lm-evaluation-harness (Gao et al., 2024) tool library for our evaluation. We performed both discriminative and generative tasks, reporting zero-shot results for the discriminative tasks and five-shot results for the generative tasks. The benchmarks we collected cover a broad range of domains to assess various aspects of the model's capabilities. A detailed description can be found in Appendix A.4.

4.3 Perplexity Results

Our proposed method significantly outperforms traditional dense models in terms of PPL on the PILE test set across different model scales, as shown in Table 1. 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 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 num-

Model	Sub-layer Counts	Experts per Sub-layer	Pile PPL (↓)
Small			
Standard Dense	N/A	N/A	14.36
Finedeep (Ours)	M=2	K=4	14.28
	M=2	K=8	**14.16**
	M=2	K=16	14.18
Medium			
Standard Dense	N/A	N/A	12.42
Finedeep (Ours)	M=2	K=4	12.24
	M=2	K=8	12.23
	M=2	K=16	12.24
	M=4	K=4	**12.13**
	M=8	K=2	12.17
Large			
Standard Dense	N/A	N/A	10.15
Finedeep (Ours)	M=2	K=8	**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.

ber 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 significantly worse than our method in terms of both PPL and benchmark evaluations. In some benchmarks, its performance is even inferior to the baseline, highlighting 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 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: Sigmoid vs. Softmax Comparison

Our proposed method computes the final routing score by applying a sigmoid function to the router's output, whereas traditional MoE structures typically use softmax normalization (Fedus et al., 2022), as shown in the following equation:

$$\boldsymbol{r}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j}) = \frac{\exp(\mathrm{E}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j})\boldsymbol{R}_{j,i})}{\sum_{i=1}^{N}\exp(\mathrm{E}_{j,i}(\tilde{\boldsymbol{h}}_t^{l,j})\boldsymbol{R}_{j,i})} \quad (9)$$

We compared these two approaches for computing routing scores using the *Medium* model, and experimental results presented in Table 4 demonstrate that the sigmoid-based routing in Finedeep

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 in the expert arrangement, K represents the number of experts per sub-layer. The AVG metric represents the average of the different benchmark results.

	w/o multi experts	w/o multi sub-layers	Finedeep
	M=2/K=1	M=1/K=16	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.

	softmax	sigmoid
	M=2/K=8	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 4: Experimental results comparing the Softmax and Sigmoid methods for computing routing scores.

achieves superior performance in terms of both PPL and benchmark results. 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.

5.3 Params and FLOPs of Finedeep

We compute the number of parameters and floating-point operations (FLOPs) of our proposed method and compare them with those of the traditional dense architecture, as shown in Table 5. Although our method introduces additional components, such as the router module and RMSNorm, the parameter overhead from these modules is minimal relative to the total model size. Our calculations indicate that across different model scales, our approach increases the parameter count by only 0.03%–0.06% and FLOPs by 0.03%–0.08% compared to the traditional dense model, which is an almost negligible difference. Despite maintaining nearly the same parameter count and FLOPs as the dense baseline, our method achieves significantly better performance, further demonstrating its effectiveness.

5.4 Mitigating Sparse Activation with Finedeep

We empirically observe that Finedeep effectively mitigates the issue of sparse activation, as illus-

	Params	GFLOPs
Small		
standard dense	665.37 M	138.33
Finedeep M=2/K=8	665.79 M	138.44
Medium		
standard dense	1.5992 B	344.29
Finedeep M=2/K=8	1.5997 B	344.37
Finedeep M=4/K=4	1.6002 B	344.51
Large		
standard dense	7.5269 B	1801.00
Finedeep M=2/K=8	7.5292 B	1801.46

Table 5: Comparison of parameter count and GFLOPs between our method and standard dense architectures. GFLOPs (Giga floating-point operations) are calculated by processing input samples with a batch size of 1 and a sequence length of 128.

trated in Figure 3. Specifically, we adopt a configuration with 2 expert sub-layers, each containing 8 experts, and visualize the distribution of the activation function outputs in the first and second sub-layers. This is compared to the activation function output distributions of the traditional dense model. Our findings indicate that the output distribution of Finedeep is more homogeneous, with fewer values concentrated around 0 and a broader distribution of larger values. Notably, this uniformity becomes more pronounced as the model depth increases.

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:

$$\mathrm{NSAR}_\tau = \frac{\sum_{i,j} \mathbb{I}\left(|\mathbf{A}_{i,j}| > \tau\right)}{B \times H}$$

$$\text{where} \quad \mathbb{I}\left(|\mathbf{A}_{i,j}| > \tau\right) = \begin{cases} 1, & \text{if } |\mathbf{A}_{i,j}| > \tau \\ 0, & \text{else} \end{cases} \tag{10}$$

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. In Figure 4, we visualize the $\mathrm{NSAR}_{0.1}$ metric across different model layers, clearly demonstrating that our method effectively mitigates the sparse activation phenomenon in the traditional dense model. By alleviating the sparse activation problem, our method increases the utilization of activation values, thereby expanding their representation capacity and enhancing the model's ability to represent

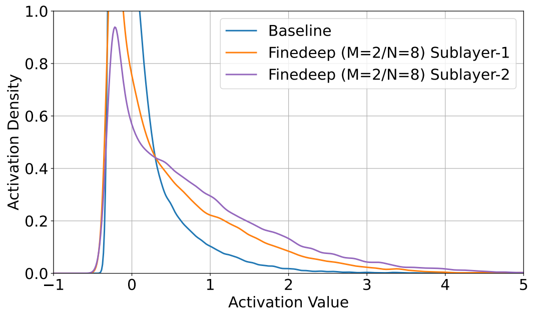

Figure 3: Output distributions of the activation functions for Finedeep and the baseline model.

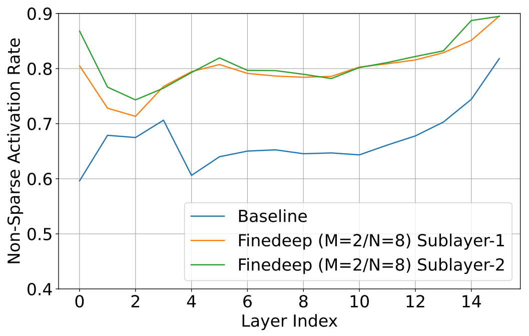

Figure 4: Variation of $\mathrm{NSAR}_{0.1}$ metrics across different model layers.

complex features, as demonstrated in Appendix A.2.

6 Conclusion

To address the sparse activation phenomenon observed in existing dense models, we have presented a novel architecture called Finedeep. It enhances the model's depth by splitting the FFN layer of traditional dense architectures into multiple experts, arranged across sub-layers. Routers within these sub-layers are employed to control the contribution of each expert. 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. Through ablation experiments, we further highlight the importance of both arranging multiple sub-layers and distributing multiple experts within each sub-layer. Our empirical results show that Finedeep effectively mitigates the issue of sparse activation.

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, et al. 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, et al. 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, et al. 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, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 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, et al. 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.

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

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, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *CoRR*, abs/2203.15556.

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, et al. 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, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason T. Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you! *Trans. Mach. Learn. Res.*, 2023.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 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, et al. 2024. Olmoe: Open mixture-of-experts language models. *CoRR*.

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, et al. 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. 2024a. 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.

Zhenpeng Su, Xing Wu, Zijia Lin, Yizhe Xiong, Minxuan Lv, Guangyuan Ma, Hui Chen, Songlin Hu, and

Guiguang Ding. 2024b. Cartesianmoe: Boosting knowledge sharing among experts via cartesian product routing in mixture-of-experts. *arXiv preprint arXiv:2410.16077*.

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, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 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 Relationship between multi-layer and single-layer experts arrangements

Our goal is to demonstrate that single-layer expert arrangement is a special case of multi-layer expert arrangement. In other words, the function space represented by multi-layer expert arrangement encompasses that of the single-layer approach. For simplicity, we consider the case where the number of sub-layers is 2, though the same reasoning can be extended to other configurations.

We can express the output of the second sub-layer $\hat{\boldsymbol{h}}_t^{l,2}$ as follows:

$$\hat{\boldsymbol{h}}_t^{l,2} = \sum_{i=1}^{K} \boldsymbol{r}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) + \hat{\boldsymbol{h}}_t^{l,1} \tag{11}$$

Here, for convenience, we directly use the output of the first sub-layer, $\hat{\boldsymbol{h}}_t^{l,1}$, as the input to the expert. The last term of the formula $\hat{\boldsymbol{h}}_t^{l,1}$ is added as the residual of the second sub-layer.

In fact, $\hat{\boldsymbol{h}}_t^{l,1}$ can also be further 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 $\hat{\boldsymbol{h}}_t^{l,1}$. So Equation 11 can be expressed in the following form:

$$\hat{\boldsymbol{h}}_t^{l,2} = \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0}) + \hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0} \tag{12}$$

We further expand the expert's computational procedure $\mathrm{E}_{2,i}$, expressed in the following form:

$$\begin{aligned}\hat{\boldsymbol{h}}_t^{l,2} = \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot (\mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) + \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,0}) + \Delta_1) \\ + \hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0}\end{aligned} \tag{13}$$

Since there is an activation function in the forward propagation process of the expert, here we use Δ_1 to represent the compensation for the effect of the nonlinear function. Equation 13 can also be interpreted in another way as a first-order Taylor expansion of equation 12.

We further expand Equation 13 as shown in the following equation:

$$\begin{aligned}\hat{\boldsymbol{h}}_t^{l,2} = \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,1}) + \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,0}) \\ + \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \Delta_1 + \hat{\boldsymbol{h}}_t^{l,1} + \hat{\boldsymbol{h}}_t^{l,0}\end{aligned} \tag{14}$$

$\hat{\boldsymbol{h}}_t^{l,1}$ can be expressed as the process of the first sub-layer expert computation, so the above equation can be transformed as:

$$\begin{aligned}\hat{\boldsymbol{h}}_t^{l,2} &= A + \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\hat{\boldsymbol{h}}_t^{l,0}) + \sum_{i=1}^{K} \boldsymbol{r}_{1,i} \cdot \mathrm{E}_{1,i}(\hat{\boldsymbol{h}}_t^{l,0}) + \hat{\boldsymbol{h}}_t^{l,0} \\ \text{where} \quad A &= \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \mathrm{E}_{2,i}(\sum_{i=1}^{K} \boldsymbol{r}_{1,i} \cdot \mathrm{E}_{1,i}(\hat{\boldsymbol{h}}_t^{l,0})) \\ &+ \sum_{i=1}^{K} \boldsymbol{r}_{2,i} \cdot \Delta_1 + \hat{\boldsymbol{h}}_t^{l,1}\end{aligned} \tag{15}$$

We can regard all the terms in the above equation except term A as the calculation process of single-layer expert arrangement. To summarize, we successfully show that single-layer expert arrangement is a special case of multi-layer expert arrangement, so we take the method of multi-layer expert arrangement.

A.2 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 5.

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

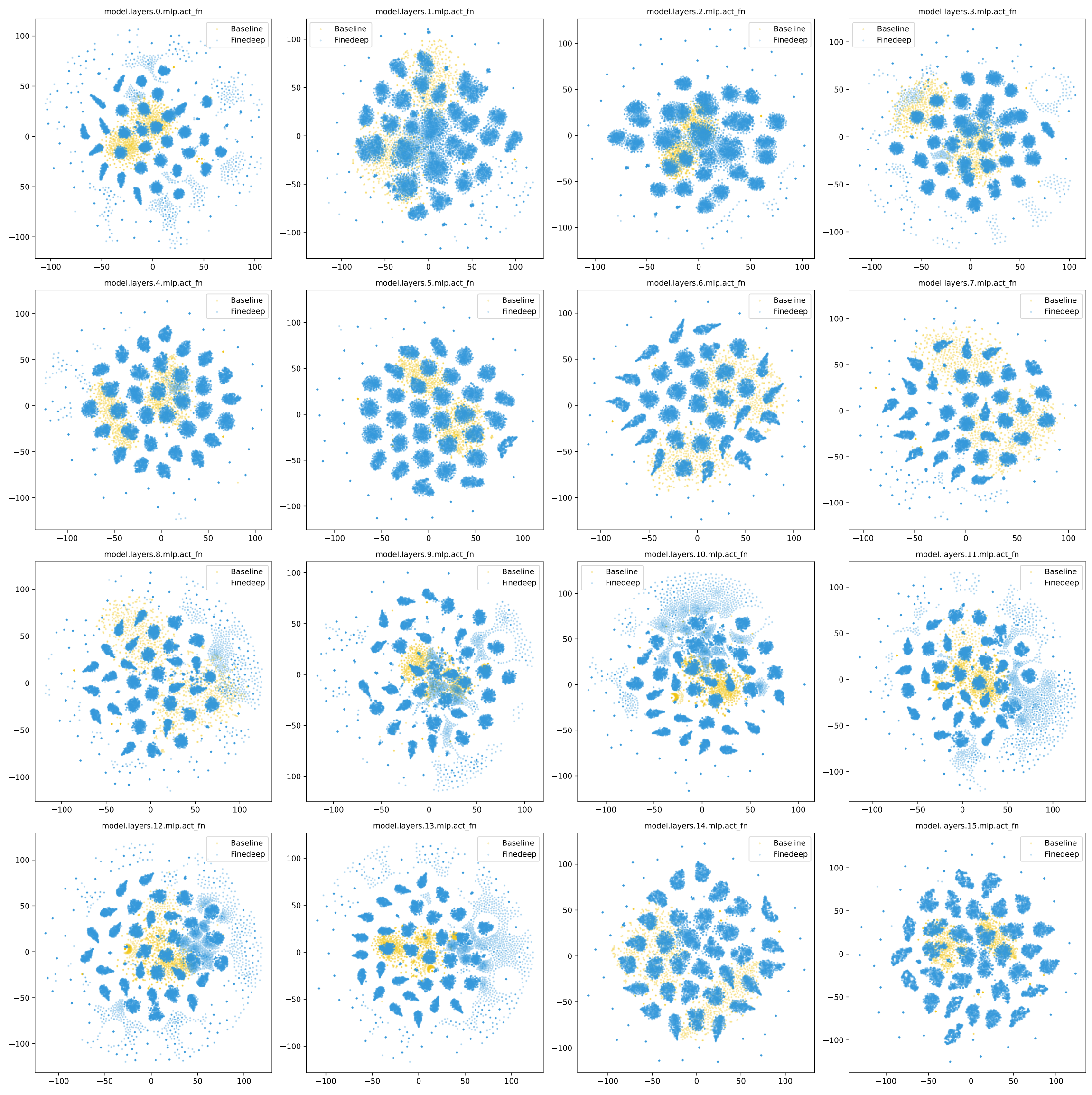

Figure 5: T-SNE clustering of activation values from different layers in the traditional dense model and the model trained with the Finedeep method.

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.3 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 6.

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

Table 6: Mixing ratios of pre-training data across different domains.

A.4 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.5 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 7: Experimental training configuration.