TUNING LANGUAGE MODELS BY MIXTURE-OF-DEPTHS ENSEMBLE

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer-based Large Language Models (LLMs) traditionally rely on final-layer loss for training and final-layer representations for predictions, potentially overlooking the predictive power embedded in intermediate layers. Surprisingly, we find that focusing training efforts on these intermediate layers can yield training losses comparable to those of final layers, with complementary test-time performance. We introduce a novel tuning framework, *Mixture-of-Depths* (MoD), which trains late layers as ensembles contributing to the final logits through learned routing weights. With the auxiliary distillation loss and additional normalization modules, we ensure that the outputs of the late layers adapt to language modeling. Our MoD framework, which can be integrated with any existing tuning method, shows consistent improvement on various language modelling tasks. Furthermore, by replacing traditional trainable modules with MoD, our approach achieves similar performance with significantly fewer trainable parameters, demonstrating the potential of leveraging predictive power from intermediate representations during training.

1 INTRODUCTION

Large Language Models (LLMs) are predominantly Transformer-based, processing sequences of input tokens by representing them as vectors and transforming them through multiple layers of transformers (Vaswani et al., 2017). Prior research has demonstrated the intermediate hidden states can carry meaningful information (Li et al., 2024), and leveraging these hidden states during decoding can improve trustworthiness (Chuang et al., 2024) and reasoning capabilities (O'Brien & Lewis, 2023). However, how to effectively utilize these intermediate layers during training remains unexplored. While each layer transformation creates new token representations added to the residual stream, only the final layer representations are used to obtain training loss. Consequently, loss minimization directly optimizes these final representations, leaving hidden representations optimized only implicitly, thereby obscuring their potential predictive power.

In this work, we investigate the predictive power of the late layers,[1] which have proven to be task-aware in early exiting language models (Schuster et al., 2022; Din et al., 2023). We begin by training models on late layers by applying the pretrained language model heads to each layer's output to calculate the loss. Our initial observations indicate that the training loss curves for the later layers started at higher values but eventually converged to similar levels, aided by simple distillation losses with respect to the output of the last layer, even without incorporating the weights of the subsequent layers (Figure 1). Figure 2 demonstrates that the trained "models" at these layers can even provide complementary evaluation results. These findings suggest that the late layers possess significant predictive potential. Given the overparameterization typical in large language models (Gao et al., 2023), the model can adapt effectively to downstream tasks even with fewer parameters.

[1] "Late" layers often refer to those closer to the output, e.g., layers 25-32 in a 32-layer LLaMA 7B models in different literatures (Din et al., 2023; Geva et al., 2023; Meng et al., 2023).

Following this observation, we introduce the *Mixture-of-Depths* (MoD) framework (§2). Unlike "mixture-of-experts" paradigm which utilized different trained models as experts for processing different input tokens Jiang et al. (2024), We propose the "mixture" across layers within a single model, where each layer output can be treated as a single model output. This approach allows us to add diversity and additional predictive power without significantly increasing parameters by training a simple gating network for the i-th late layer (§2.2).

We focus on tuning large language models. Our framework can be applied on top of any training methods as the hidden state dimensions remain consistent during training. Traditionally, language model heads in LLMs are trained to unembed hidden states from the last transformer layer. Applying the LM head directly to late layers during tuning can result in worse initial training performance, as shown in Figure 1. To ensure LM adaptation during tuning without interfering with the original model predictions, we apply an additional model distillation loss (§2.3) where the last layer output serves as the teacher. This method does not add any additional trainable parameters and ensures that the late layers adapt to the predictions. Experiments (§3) demonstrate that applying MoD tuning consistently improves performance on arithmetic and commonsense reasoning tasks with a minimal increase in trainable parameters (+0.04%). Furthermore, by replacing traditional trainable modules with MoD, we achieve similar performance with 97% fewer trainable parameters.

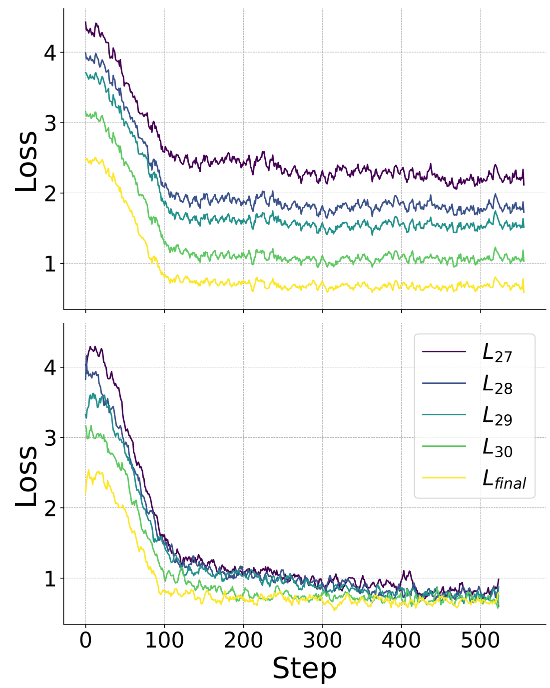

Figure 1: Tuning loss curves for LLaMA2-7B (Touvron et al., 2023b) on ARC dataset (Clark et al., 2018). Above shows the loss curve of late layers when optimizing the loss based on the last layer output when late layers are optimized implicitly; Below shows the loss curves when optimizing the loss on each late layer output with the distillation loss $\mathcal{L}_{distill}$ w.r.t. the last layer.

As analysis (§4), we study the learned patterns by MoD routing (§4), evaluate the performance when varying values of k, and explore the trade-off between performance and efficiency (§4.2, 4.3).

2 MIXTURE-OF-DEPTHS

Recent language models consist of an embedding layer, n stacked transformer layers L, and an affine layer $\phi(\cdot)$ for predicting the next-word distribution, often referred to as the language model head (Geva et al., 2022; Luo & Specia, 2024). We aim to identify a layer range k, where the last k layers carry higher-level task-aware information and can map hidden states to meaningful predictive logits (Belrose et al., 2023). For an LLM with n layers, we define the set of the last k layers as $\mathcal{K} = \{L_{n-k+1}, L_{n-k+2}, \ldots, L_n\}$. As shown in Figure 1, late layers exhibit learning loss curves similar to the final layer, indicating their task informativeness.

Additionally, metrics extracted from the inference process can dynamically determine this range. For example, Chuang et al. (2024) use the Jensen-Shannon Divergence between early and final layers as a distance measurement to decode in contrastively, ensuring that selected layers include more task-related knowledge.

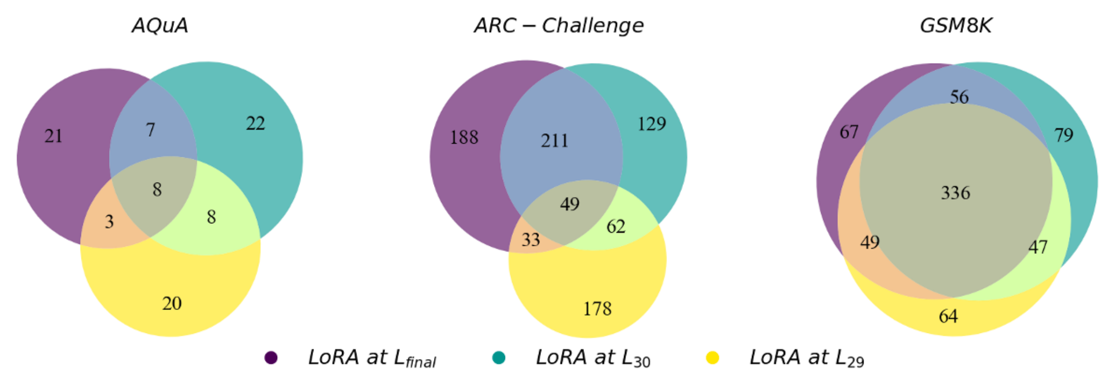

Figure 2: Intersection of solved problems by tuning loss layers on the AQuA (Ling et al., 2017), ARC-Challenge (Clark et al., 2018), and GSM8K (Cobbe et al., 2021b) datasets. The digits in the Venn diagram illustrate the number of overlapping solved problems and the complementary solved problems for each method.

However, we apply a simple empirical selection with a single k across all tasks to demonstrate the effectiveness of the MoD framework, leaving dynamic selection for future research.

2.1 Early-Exit for Late Layers

The idea of applying language heads directly to the hidden states of the middle layers, known as *early exit* (Teerapittayanon et al., 2016; Elbayad et al., 2020; Schuster et al., 2022), has proven effective even without a special training process (Kao et al., 2020). The residual connections (He et al., 2016) in transformer layers allow hidden representations to evolve gradually, enabling the formation of task-aware representations without abrupt changes.

Given a sequence of tokens $\{x_1, x_2, \dots, x_{t-1}\}$, the embedding layer first converts the tokens into a sequence of vectors $H_0 = \{h_1^{(0)}, \dots, h_{t-1}^{(0)}\}$, where $h_t^{(0)} \in \mathbb{R}^d$ and d is the hidden state dimension. This sequence H_0 is then processed successively by each transformer layer, with the output of the j-th layer denoted as H_j. The vocabulary head $\phi(\cdot)$ then outputs the logits ℓ_t of the next token x_t over the vocabulary set $\mathcal{V}$:

$$\ell(x_t \mid x_{<t}) = \phi\big(\mathcal{N}_p(h_t^{(N)})\big)\big)_{x_t}, \quad x_t \in \mathcal{V}.$$

Here, $\mathcal{N}_p$ is the pre-trained normalization module before the vocabulary head. This method is often considered a form of logit lens (Nostalgebraist, 2020), which uses the vocabulary head to probe into inner representations. However, the trainable predictive power of these representations remains unexplored. In §2.2, we show how to combine the train-time predictive power of late layers with final layer logits.

2.2 MoD Routing Network

Instead of applying $\phi(\cdot)$ only on the final layer, we incorporate the predictive power of late layers into the final prediction. We want to route the most informative representation for training to the final logit calculation. Motivated by the MoE framework (Fedus et al., 2022; Jiang et al., 2024), the output of the ensemble logits is given by:

$$\sum_{i=0}^{k-1} G(x)_i \cdot \ell_i(x), \quad G(x) := \text{Softmax}(x \cdot W_g).$$

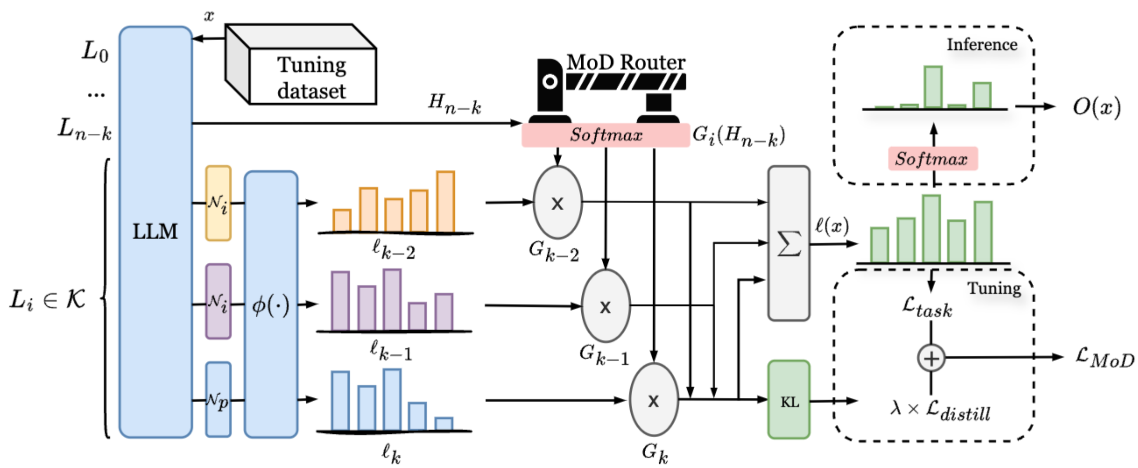

Figure 3: The overall framework of Mixture-of-Depths (MoD), which can be applied on top of any tuning method like LoRA (Hu et al., 2022). Given a pre-trained LLM and a tuning dataset, MoD applies trainable normalization $\mathcal{N}_k$ and pre-trained language model heads $\phi(\cdot)$ to the last k layers $\{L_{n-k+1}, \dots, L_n\}$. Each layer's output is combined using learned routing weights to produce the final logits. During training, a auxiliary teacher-enforced distillation loss $\mathcal{L}_{distill}$ is applied, where the final layer output serves as the teacher. MoD utilizes the ensemble logits during inference.

Here, $G(\cdot)_i$ denotes the output of the routing network for the i-th expert, and $\ell_i(\cdot)$ is the output logits of the i-th late layer. Here, $x = H_{n-k}$, which is the output of the layer before the last k layer. The routing network $G(x)_i$ is implemented by taking the softmax over a linear layer. The final logits are then obtained by summing the weighted logits from k layers:

$$\ell(x_t \mid x_{<t}) = \sum_{i=0}^{k-1} G(x)_i \cdot \ell_i(x)$$

Additionally, one advantage of the MoD is its potential to improve inference efficiency by avoiding excessive computation when the routing vector is sparse. Following Shazeer et al. (2017), we achieve this by applying the softmax over the Top-K logits of the linear layer:

$$G_{\text{TopK}}(x) := \text{Softmax}(\text{TopK}(x \cdot W_g)),$$

where $(\text{TopK}(\ell))_i := \ell_i$ if ℓ_i is among the top-K coordinates of logits $\ell \in \mathbb{R}^k$ and $(\text{TopK}(\ell))_i := -\infty$ otherwise.

In our main experiments (§3), we utilize $G(x)$ to demonstrate the effectiveness of the MoD framework. We investigate the performance and efficiency trade-offs of using $G_{\text{TopK}}(x)$ in §4.2. This exploration allows us to understand how sparse routing mechanisms can optimize computational resources while maintaining predictive accuracy.

2.3 Late Layers Adaptation by Normalization and Distillation

Directly combining the logits of late layers using the LM head can result in worse training loss at the start of tuning (Figure 1). Previous works (Belrose et al., 2023) have attempted to learn an affine matrix A_ℓ to map hidden states of layer ℓ to the input space of the LM head. We aim to investigate more efficient adaptation methods while minimizing interference with model predictions and avoiding excessive additional trainable parameters.

Inspired by normalization studies in neural networks and the effectiveness of tuning the normalization module for domain adaptation (Zhao et al., 2023), we propose tuning an additional normalization module for each late layer as a simple yet powerful adaptation method. We set the additional normalization module $\mathcal{N}_k$ to match the architecture of the pretrained $\mathcal{N}_p$. For instance, in the LLaMA2 model (Touvron et al., 2023b), we follow the LayerNorm setting (Ba et al., 2016). The learnable parameters in the normalization, γ_k and β_k, are trained individually for each k-th late layer to ensure specific adaptation for each layer.

Following our assumption in §2.2, we treat each of the $k-1$ late layers (excluding the final layer) as smaller models, with the final layer as the larger model with the most predictive power. We use the final layer as the teacher model to supervise the output of earlier layers for adaptation. We define a teacher-enforced distillation loss that measures the difference between the predictions of the intermediate models and the final layer's predictions. The distillation loss is computed as the sum of the KL divergence between each intermediate layer's output distribution P_i and the final layer's output distribution P_n:

$$\mathcal{L}_{distill} = \sum_{i=0}^{k-2} \mathrm{KL}(P_i \parallel P_n),$$

where P_i is the output distribution of layer i, and P_n is the output distribution of the final layer. The final loss is then the sum of the task loss and the distillation loss:

$$\mathcal{L}_{\mathrm{MoD}} = \mathcal{L}_{task} + \lambda \mathcal{L}_{distill},$$

where λ is a hyperparameter that controls the weight of the distillation loss. By tuning with the normalization modules and distillation loss, we adapt the $k-1$ layer representations to be more suitable for the language modeling task, ensuring their contributions are aligned with the original task loss.

3 Experiments

We evaluate the MoD framework on two types of language modeling tasks: arithmetic reasoning and commonsense reasoning. The MoD framework minimally increases trainable parameters and can be integrated with any existing training method, as the hidden state dimensions remain consistent during training. We use LoRA (Hu et al., 2022) as our base tuning method, which has been shown to reduce the number of tunable parameters while maintaining performance comparable to full finetuning. We define a single LoRA layer as L_{LoRA}. We use two baselines:

1. The model tuned with LoRA excluding the last k layers, denoted as $\mathrm{LoRA}_{\neg\mathcal{K}}$.
2. The model tuned with LoRA on all layers, denoted as $\mathrm{LoRA}_{\mathrm{all}}$.

The notation $\mathrm{LoRA}_{\mathrm{all}}$ represents the model tuned with LoRA applied to all layers, including the last k layers which is identical to $\mathrm{LoRA}_{\neg\mathcal{K}} + L_{\mathrm{LoRA}} \times |\mathcal{K}|$ specified in the tables.

As shown in Table 1, MoD consistently improves performance when applied on top of $\mathrm{LoRA}_{\mathrm{all}}$ with minimally added parameters. Though MoD is not designed as an additional training architecture, experiments also demonstrate that it can replace the LoRA module while retaining similar or even better performance with 97% [2] fewer trainable parameters. We conduct experiments with LLaMA-1 (Touvron et al., 2023a) and LLaMA-2 (Touvron et al., 2023b) models with 7B parameters. The weight of the distillation loss λ is set to

[2]The percentage is calculated by the additional parameters introduced by MoD divided by the additional parameters introduced by LoRA_{all}.

Table 1: Accuracy comparison of MoD built upon LoRA (Hu et al., 2022) for LLaMA-7B (Touvron et al., 2023a) and LLaMA2-7B (Touvron et al., 2023b) on seven arithmetic reasoning datasets. We train the models on a single combined dataset follow Hu et al. (2023) and report averaged performance of three runs with distinct random seeds. The number in parentheses (%) indicates the percentage of added trainable parameters relative to the LoRA$\neg|\mathcal{K}|$ baseline. We report the task-level averaged results in Avg.

METHOD	ADDSUB	AQUA	GSM8K	MAWPS	MULTIARITH	SINGLEEQ	SWAMP	AVG.
LLaMA-7B								
LoRA$_{\text{ALL}}$ (+10.3%)	41.3	15.4	38.5	58.0	81.0	**62.9**	44.2	48.8
LoRA$_{\neg\mathcal{K}}$	38.7	13.4	37.3	56.3	78.2	59.8	42.3	46.6
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ + **MoD$_{\mathcal{K}}$ (+10.4%)**	**42.0**	15.8	**39.1**	**58.5**	**81.3**	**62.9**	**44.9**	**49.2**
+ **MoD$_{\mathcal{K}}$ (+0.04%)**	41.5	**16.1**	38.2	58.4	80.7	62.3	43.8	48.7
LLaMA2-7B								
LoRA$_{\text{ALL}}$ (+10.3%)	51.1	24.4	43.6	62.6	84.2	66.9	47.7	54.5
LoRA$_{\neg\mathcal{K}}$	46.3	20.5	39.7	60.6	81.4	62.0	43.2	50.5
+ $L_{\text{LoRA}} \times |\mathcal{K}|$ + **MoD$_{\mathcal{K}}$ (+10.4%)**	**51.2**	**25.5**	**43.9**	63.1	**84.3**	**67.3**	**48.0**	**54.8**
+ **MoD$_{\mathcal{K}}$ (+0.04%)**	50.1	24.3	43.4	**63.7**	82.2	66.8	47.5	54.0

0.0001 for all datasets and models, and the routing network is Gaussian initialized with a standard deviation of 0.02 and a mean of 0. All experiments are run on NVIDIA A6000 GPUs. Detailed experimental settings are provided in Appendix B.

3.1 ARITHMETIC REASONING

Arithmetic reasoning includes seven datasets for math word problems: AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021a), MAWPS (Koncel-Kedziorski et al., 2016), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021). Models need to generate chain-of-thought (Wei et al., 2022) reasoning steps before the final answer. We replicate the experimental setup from Hu et al. (2023) on a combined dataset of these seven arithmetic reasoning tasks with LM-generated chain-of-thought steps (MATH7K) and report scores on all test sets. We only evaluate the correctness of the final numeric or multiple-choice answer. Details of the dataset are provided in Appendix A.1. For MATH7K, we set k to 3 for both LLaMA-1 and LLaMA-2 models across all datasets. Note that different models and datasets might benefit from a different value of k, or we could dynamically select k during training, which we leave for future research.

The results in Table 1 show that the MoD framework consistently improves performance on arithmetic reasoning tasks when applied on top of LoRA$_{\neg\mathcal{K}}$. Furthermore, MoD alone, even with only 0.19% added parameters, provides competitive performance with LoRA$_{\text{all}}$. These results validate our approach of utilizing late layer during training to enhance model performance in complex reasoning tasks.

3.2 COMMONSENSE REASONING AND GENERAL LANGUAGE MODELLING

Commonsense reasoning is evaluated using four datasets: the Challenge Set and Easy Set of ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), and OBQA (Mihaylov et al., 2018a). These tasks are formulated as multiple-choice problems. We follow the setup from Hu et al. (2023), but train each dataset separately to assess the effectiveness of our MoD framework on individual datasets. To evaluate general language modeling capability, we select 20% of the TruthfulQA dataset and report the True*Informative score. We also report the performance on the STEM subtasks of the MMLU benchmark, following the setup of Brown et al. (2020). Dataset details are provided in Appendix A.2. We maintain the same settings as described in

Table 2: Comparison of MoD on seven commonsense reasoning datasets and two general language modelling datasets. We train the models on each dataset and report the averaged performance of three runs with distinct random seeds. The number in parentheses (%) indicates the percentage of added trainable parameters relative to the LoRA$\neg|\mathcal{K}|$ baseline. We report the task-level averaged results in Avg.

METHOD	ARC-E	ARC-C	BOOLQ	OBQA	HELLASWAG	TRUTHFULQA	MMLU	AVG.
LLaMA-7B								
LoRA$_{\text{ALL}}$ (+10.3%)	**79.6**	42.0	68.2	79.8	79.2	**36.3**	28.3	59.1
LoRA$_{\neg\mathcal{K}}$	75.3	39.0	65.1	78.4	76.1	35.7	25.9	56.5
+ $L_{\text{LoRA}} \times \lvert\mathcal{K}\rvert$ + **MoD**$_{\mathcal{K}}$ **(+10.4%)**	**79.6**	**47.2**	**69.8**	**80.1**	**80.3**	36.2	29.1	**60.3**
+ **MoD**$_{\mathcal{K}}$ **(+0.04%)**	78.1	43.1	69.2	79.6	79.7	36.0	28.2	59.1
LLaMA2-7B								
LoRA$_{\text{ALL}}$ (+10.3%)	81.8	53.8	70.9	82.0	82.5	**49.6**	**37.3**	65.4
LoRA$_{\neg\mathcal{K}}$	75.9	48.0	70.5	80.4	79.9	46.5	35.8	62.4
+ $L_{\text{LoRA}} \times \lvert\mathcal{K}\rvert$ + **MoD**$_{\mathcal{K}}$ **(+10.4%)**	82.2	**56.4**	71.4	**83.4**	**84.0**	49.3	37.1	**66.2**
+ **MoD**$_{\mathcal{K}}$ **(+0.04%)**	**82.9**	53.4	**71.5**	82.1	83.5	48.9	36.9	65.6

§3.1. As shown in Table 2, the integration of MoD with LoRA leads to consistent performance improvements with only a minimal increase in trainable parameters, reinforcing the practicality of our approach.

3.3 INSTRUCTION FOLLOWING

Table 3: Average scores on MT-Bench assigned by GPT-4 to the answers generated by tuned LLaMA-7B/LLaMA2-7B models.

MODEL	METHOD	+PARAMS (%)	SCORE
LLaMA-7B	LoRA$_{\neg\mathcal{K}}$	-	3.61
	+ $L_{\text{LoRA}} \times \lvert\mathcal{K}\rvert$	10.3	**4.35**
	+ $L_{\text{LoRA}} \times \lvert\mathcal{K}\rvert$ + **MoD**$_{\mathcal{K}}$	10.4	**4.35**
	+ **MoD**$_{\mathcal{K}}$	0.04	4.16
LLaMA2-7B	LoRA$_{\neg\mathcal{K}}$	-	4.92
	+ $L_{\text{LoRA}} \times \lvert\mathcal{K}\rvert$	10.3	**5.47**
	+ $L_{\text{LoRA}} \times \lvert\mathcal{K}\rvert$ + MoD$_{\mathcal{K}}$	10.4	5.29
	+ **MoD**$_{\mathcal{K}}$	0.04	5.23

We evaluate the effectiveness of MoD across LLaMA-7B and LLaMA2-7B for instruction tuning using a 10K subset of the cleaned Alpaca dataset (Taori et al., 2023). The fine-tuned models are then assessed on the MT-Bench benchmark (Zheng et al., 2023) by generating responses to a predefined set of 80 multi-turn questions. These responses are subsequently evaluated by GPT-4 (OpenAI, 2023), which reviews each answer and assigns a numerical score out of 10.

Our findings indicate that the performance of MoD is comparable to the LoRA baseline, though no significant performance gains were observed. We hypothesize that this could be due to the nature of instruction-following tasks, which may require more processing in the later layers to appropriately format instructed responses. MoD, by contrast, bypasses these processes while maintaining similar performance. Future work may explore how the MoD framework can be adapted to enhance instruction-following capabilities in language modeling by learning more robust instruction-tuning mechanisms.

4 ANALYSIS

Using the training setup from §3, we conducted several analyses on our MoD framework. We examined the sparsity curve of the routing network at the route level across training tokens (§4.1), explored the advantages and trade-offs of sparse routing (§4.2), and performed ablation studies on the different components in MoD (§4.3).

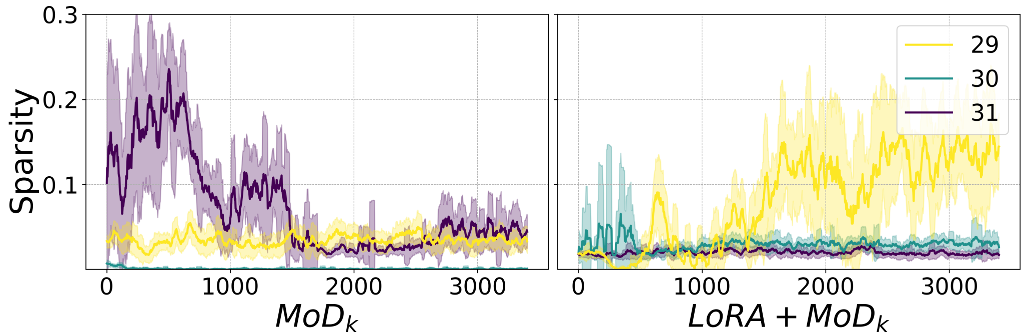

Figure 4: Sparsity scores for MoD (left) and MoD trained with k LoRA layers (right). The curve is smoothed using moving average smoothing.

4.1 Learned Routing Pattern Across Tokens

In this section, we analyze the routing patterns learned with MoD for the $\mathcal{K}$ ensemble layers during training. With a Gaussian-initialized routing network, we measure the sparsity of the weights across the training tokens, i.e., how many weights are close to zero. We calculate the proportion of weights below a threshold, ϵ, which we set to 1×10^{-5}. A lower level of sparsity often implies that the model is selectively using current routes while ignoring others, leading to the discussion in §4.2. We also record the mean and variance to measure the tendency and dispersion for each k route, as detailed in Appendix C.1. We evaluate MoD trained on top of LoRA and MoD trained without k LoRA layers using the LLaMA 7B model on the ARC easy subset. According to Figure 4, we notice an interesting learned pattern discrepancy between MoD trained with or without LoRA layers. When trained without k LoRA layers, the sparsity score for the last layer remains low, while the sparsity level of layer 30 is high initially and then decreases, and the sparsity level of layer 29 increases through training. This suggests that the model generally learns to rely more on the last two layers' outputs, especially the last layer for the ensemble. However, when trained with k LoRA layers in the ensemble, the sparsity level of the last layer is much higher, while the levels for the other two layers remain low. This indicates that the additional trainable modules inside MoD help the late layers contribute more to the ensembles and become more task-informative, aligning with our assumption in §2.3. Notably, both methods yield better performance than the baseline according to Table 3.2, suggesting that there is still significant predictive potential through different weight combinations for the ensembles.

4.2 MoD Sparse Routing

As shown in §4.1, the sparsity level of the MoD routing output can be high, suggesting the potential for sparse routing vectors during inference. In this section, we investigate whether we can train the MoD with the G_{TopK} variant introduced in §2.2. Ideally, if the routing can be sparse without compromising the ensemble effectiveness, we can improve inference efficiency by enabling early exit when the Top-K selected routes occur before the last layer. We introduce MoD Sparse Routing (MoD$_{sparse}$), which utilizes a routing network activated by G_{TopK}. To thoroughly examine the effectiveness of sparse routing, we select a larger ensemble layer range to potentially

Table 4: Acceleration ratios for different Top-K values when $k = 6$ compared to the LoRA baseline. The results represent the overall speedup across 1000 iterations for each dataset.

Dataset	Top-2	Top-3	Top-4	Top-5
ARC-e	1.4×	1.5×	1.4×	1.1×
ARC-c	1.6×	1.4×	1.3×	1.0×

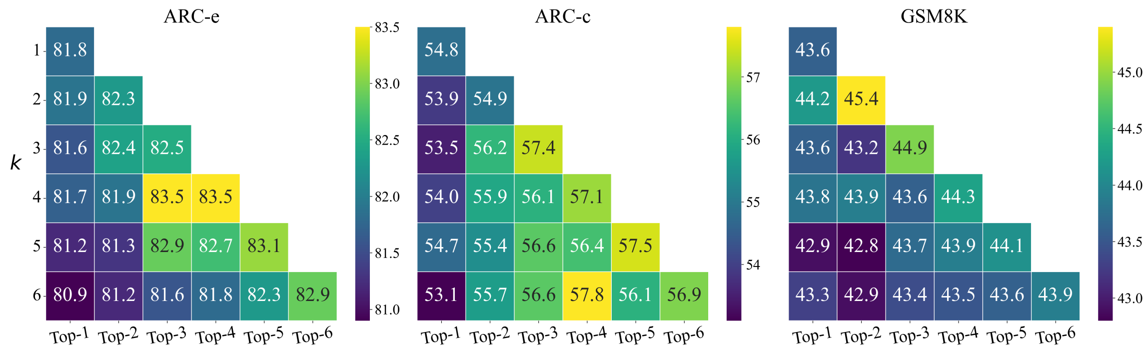

Figure 5: Accuracy scores for different k ensemble layer ranges and Top-K sparse routing values. Lighter colors indicate better performance.

increase opportunities for early exit. We use $k = 6$ for this section, with results for other datasets provided in Appendix C.2.

First, we investigate whether a larger ensemble range k provides more diverse tuning information to improve performance or introduces noise that negatively impacts it. In Figure 5, we observe that the optimal k value often occurs around 3 to 4. For relatively challenging datasets that require extensive reasoning, such as GSM8K, increasing k does not provide additional trainable information and can harm performance, as seen with $k = 6$ for GSM8K. Conversely, for relatively easier datasets like ARC-e, increasing k consistently improves performance. Second, we examine whether Top-K activation significantly interferes with MoD performance. Figure 5 shows the performance on the ARC-c dataset, varying different k values and Top-K values up to 6. When k = Top-K, it corresponds to the original MoD routing. We observe that the original MoD routing always provides the best performance. While Top-K activation slightly decreases MoD's performance, it still outperforms the baseline when $k = 1$. Additionally, Table 4 shows that larger Top-K values result in greater acceleration ratios for generation, suggesting a potential trade-off between utilizing MoD's additional predictive power and exploiting its sparsity to improve efficiency. This trade-off encourages further study in future research.

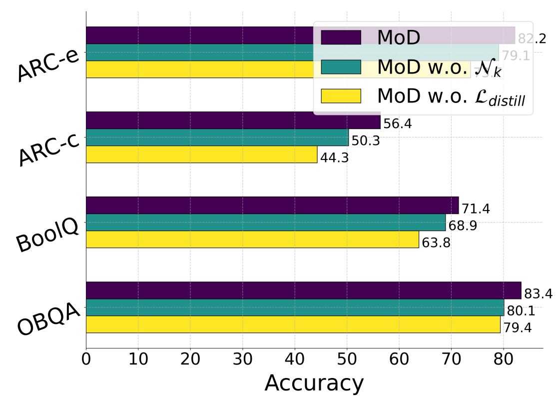

Figure 6: Ablation study results for MoD on four commonsense reasoning datasets using the LLaMA2-7B model.

4.3 Ablation Study on Adaptation Modules

Figure 1 shows the loss curve of late layers when optimizing the loss based on the last layer output when late layers are optimized implicitly and the loss curves when optimizing the loss on each late layer output with the distillation loss $\mathcal{L}_{distill}$ w.r.t. the last layer. We find that the introduction of $\mathcal{L}_{distill}$ makes bringing the activations to the final logits prediction at the very start of the training more stable within the first 200 steps even model's pretraining paradigm is not doing so. We also conduct an ablation study to analyze the impact of $\mathcal{L}_{distill}$ with the adaptation module $\mathcal{N}_k$ introduced in §2.3. We name different ablations of MoD as follows: 1) **MoD w.o.** $\mathcal{N}_k$: Instead of using a trained normalization for each ensemble layer, we use the pre-trained normalization before the LM head for all k ensemble layers. 2) **MoD w.o.** $\mathcal{L}_{distill}$: MoD tuned without the distillation loss $\mathcal{L}_{distill}$. The tuning loss is the original task loss, which is cross-entropy loss for language modeling. We apply the ablation study on four commonsense reasoning datasets using the LLaMA2-7B model. The results are presented in Figure 6. The findings are as follows: 1) The introduced normalization components for language modeling adaptation are effective. Removing any of these components harms performance. 2) The distillation loss is generally more important than the additional trainable normalization. This may be because the strong task signals provided by the supervision from the last layer are essential for the ensemble layers to adapt. For the approach of the supervison, there may be other effective methods such as JS divergence (Chuang et al., 2024) or supervision by Reinforcement Learning (Wu et al., 2024), which we leave for future study.

5 Related Work

Early Exit in Transformer Layers Early exit strategies in language models are often explored to improve efficiency. Several works focus on enhancing inference efficiency by terminating computation at dynamically-decided earlier layer outputs (Xin et al., 2020; Schuster et al., 2022). A common approach for adapting intermediate layer output to language modeling involves training an affine transformation (Belrose et al., 2023; Din et al., 2023). Early exit strategies have also been explored for interpretability, analyzing the linearity properties of transformer components (Geva et al., 2023; Hernandez et al., 2023). However, the utilization of intermediate layer output during training remains largely unexplored. A recent work (Elhoushi et al., 2024) applies layer dropout and an early exit loss to increase the accuracy of early exits, but its primary focus is still on inference efficiency. To the best of our knowledge, our work is the first to utilize early exit logits together with the final layer logits to incorporate task-aware representations from intermediate layers into the loss calculation.

Logit-Level Arithmetic Operations at the logit level have proven effective in steering the output of LLMs (Luo & Specia, 2024). From a multi-model perspective, there has been a growing body of work focusing on "mixturing" the abilities of different trained models in line with the Mixture-of-Experts framework (Shazeer et al., 2017; Jiang et al., 2024). Liu et al. (2021); Gera et al. (2023) have also shown the effectiveness of ensembling logits from multiple LMs. From a single model perspective, contrasting logits from different layers of a model (Chuang et al., 2024; Gera et al., 2023) has shown promising performance improvements in the trustworthiness of generation and addressing the resource-intensive issues of larger models (Liu et al., 2024). Our work builds upon logit-level arithmetic and follows the line of ensembling logits, focusing not on a multi-model perspective but rather on utilizing the late layers' outputs within a single model for tuning. This approach has been considered only during inference in previous work.

6 Conclusion

In this paper, we explored the predictive power of late layers in LLMs and introduced the *Mixture-of-Depths* (MoD) tuning framework. By tuning LLMs using ensembled logits from MoD routing and adaptation components, we demonstrated consistent improvements in reasoning tasks with minimal additional parameters. Additionally, our approach shows the potential to replace traditional training modules with significantly fewer parameters. Our findings highlight the effectiveness of leveraging intermediate layer representations during training, offering a lightweight and complementary direction for optimizing LLMs.

7 LIMITATIONS

Future work could explore dynamic layer selection methods and refine the layer range of the MoD framework to maximize its potential, rather than relying on empirical selection. Additionally, more effective tuning of other hyperparameters, such as λ, the weight of the distillation loss, should be investigated. Improving the effectiveness of MoD on a broader range of tasks, such as instruction following, remains an open question, as discussed in §3.3. Extending MoD to evaluate its performance on bidirectional LLMs, such as RoBERTa (Liu et al., 2019), would help determine if it generalizes well across different transformer-based language models. Due to hardware limitations, our experiments were restricted to LLMs at the 7B scale. Exploring the impact of MoD on larger models is an important direction for future research.

REFERENCES

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. CoRR, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In International Conference on Learning Representations (ICLR), 2024. URL https://arxiv.org/pdf/2309.03883.pdf.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv:1803.05457v1, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021a.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021b.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. Jump to conclusions: Short-cutting transformers with linear transformations, 2023.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In ICLR 2020-Eighth International Conference on Learning Representations, pp. 1–14, 2020.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding, 2024.

William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning, 2022.

Ze-Feng Gao, Kun Zhou, Peiyu Liu, Wayne Xin Zhao, and Ji-Rong Wen. Small pre-trained language models can be fine-tuned as large models via over-parameterization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3819–3834, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.212. URL https://aclanthology.org/2023.acl-long.212.

Ariel Gera, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. The benefits of bad advice: Autocontrastive decoding across model layers. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 10406–10420, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.580. URL https://aclanthology.org/2023.acl-long.580.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. arXiv preprint arXiv:2203.14680, 2022.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Empirical Methods in Natural Language Processing (EMNLP), 2023. URL https://arxiv.org/abs/2304.14767.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. 2023.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In EMNLP, pp. 523–533, 2014.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023.

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, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

Wei-Tsung Kao, Tsung-Han Wu, Po-Han Chi, Chun-Cheng Hsieh, and Hung-Yi Lee. Bert's output layer recognizes all hidden layers? some intriguing phenomena and a simple way to boost bert. arXiv preprint arXiv:2001.09309, 2020.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. Transactions of the Association for Computational Linguistics, 3:585–597, 2015.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1136. URL https://aclanthology.org/N16-1136.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. Advances in Neural Information Processing Systems, 36, 2024.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 158–167, 2017.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 6691–6706, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.522. URL https://aclanthology.org/2021.acl-long.522.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy, 2024. URL https://arxiv.org/abs/2401.08565.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

Haoyan Luo and Lucia Specia. From understanding to utilization: A survey on explainability for large language models, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In EMNLP, 2018a.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Conference on Empirical Methods in Natural Language Processing, 2018b. URL https://api.semanticscholar.org/CorpusID:52183757.

Nostalgebraist. Interpreting gpt: the logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. arXiv preprint arXiv:2309.09117, 2023.

OpenAI. Gpt-4 technical report. 2023. URL https://cdn.openai.com/papers/gpt-4.pdf.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Proceedings of NAACL, pp. 2080–2094, 2021. URL https://aclanthology.org/2021.naacl-main.168.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. arXiv preprint arXiv:1608.01413, 2016.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=uLYc4L3C81A.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html, 2023.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 2464–2469. IEEE, 2016.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. arXiv preprint arXiv:2201.11903, 2022.

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, and Gholamreza Haffari. Mixture-of-skills: Learning to optimize data usage for fine-tuning large language models, 2024.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 2246–2251, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.204. URL https://aclanthology.org/2020.acl-main.204.

Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2023.

A DATASETS

Table 5: Details of 11 datasets being evaluated according to Hu et al. (2023) and Hendrycks et al. (2021). Math: arithmetic reasoning. CS: commonsense reasoning.

DATASET	DOMAIN	# TRAIN	# TEST	ANSWER
MULTIARITH	MATH	-	600	NUMBER
ADDSUB	MATH	-	395	NUMBER
GSM8K	MATH	8.8K	1,319	NUMBER
AQUA	MATH	100K	254	OPTION
SINGLEEQ	MATH	-	508	NUMBER
SVAMP	MATH	-	1,000	NUMBER
MAWPS	MATH	-	238	NUMBER
BOOLQ	CS	9.4K	3,270	YES/NO
ARC-E	CS	2.3K	2,376	OPTION
ARC-C	CS	1.1K	1,172	OPTION
OBQA	CS	5.0K	500	OPTION
HELLASWAG	CS	39.9K	10042	OPTION
MMLU	-	99.8K	14042	OPTION

Dataset Statistics and Examples Dataset statistics and simplified training examples from each dataset are provided in Table 5. The original training dataset of Math10K accidentally includes testing examples from AddSub, MultiArith, and SingleEq tasks, as these tasks are part of the MAWPS training dataset, causing a data leak. To address this, we replicate the experimental setup suggested by Hu et al. (2023) on a combined training dataset (MATH7K). For the commonsense reasoning dataset, we trained individual datasets with a newly designed prompt format to address various issues reported with the LLaMA tokenizer in the original prompt format.

A.1 ARITHMETIC REASONING

We conduct extensive empirical studies on fourteen benchmark datasets, focusing on two categories of reasoning problems: **Arithmetic Reasoning:** 1. **GSM8K** (Cobbe et al., 2021b): A dataset comprising

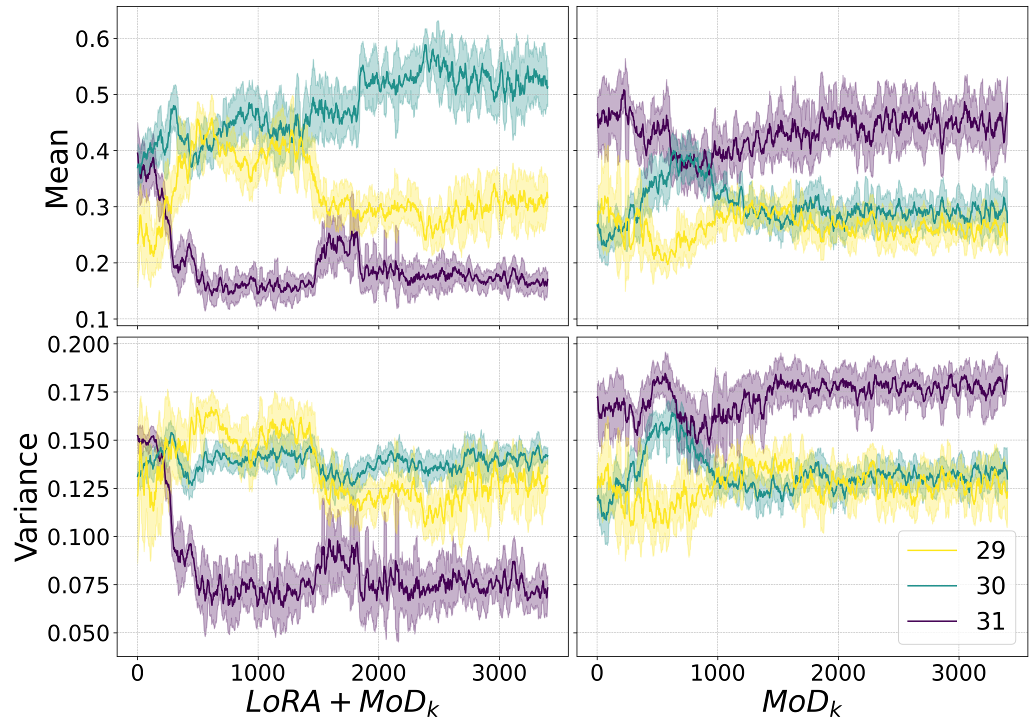

Figure 7: Mean and variance for MoD (right) and MoD trained with k LoRA layers (left). The curve is smoothed using moving average smoothing with a window size of 3 and $k = 3$.

high-quality, linguistically diverse grade school math word problems created by human problem writers. 2. **SVAMP** (Patel et al., 2021): A benchmark of one-unknown arithmetic word problems designed for up-to-4th grade students, created by making simple modifications to problems from an existing dataset. 3. **MultiArith** (Roy & Roth, 2016): A dataset featuring math word problems that require multiple reasoning steps and operations. 4. **AddSub** (Hosseini et al., 2014): A collection of arithmetic word problems focused on addition and subtraction. 5. **AQuA** (Ling et al., 2017): A dataset of algebraic word problems accompanied by natural language rationales. 6. **SingleEq** (Koncel-Kedziorski et al., 2015): A set of grade-school algebra word problems that map to single equations of varying lengths.

A.2 Commonsense Reasoning

We trained our method on four commonsense reasoning dataset separately. They are: 1. **BoolQ** (Clark et al., 2019): A question-answering dataset containing 15,942 naturally occurring yes/no questions generated in unprompted and unconstrained settings. 2. **ARC-c** and **ARC-e** (Clark et al., 2018): The Challenge Set and Easy Set of the ARC dataset, consisting of genuine grade-school level, multiple-choice science questions. 3. **OBQA** (Mihaylov et al., 2018b): A dataset containing questions that require multi-step reasoning, use of additional common and commonsense knowledge, and rich text comprehension.

B EXPERIMENT SETTINGS

We mainly follow the experimental settings of Hu et al. (2023). We maintain a batch size of 16 and set the learning rate for all methods to 3e-4. Each method is fine-tuned for two epochs on each dataset.

C ANALYSIS

C.1 MEAN AND VARIANCE FOR ROUTING PATTERN ACROSS TOKENS

In this section, we analyze the routing patterns learned with MoD for the $\mathcal{K}$ ensemble layers during training. We measure the mean and variance of the weights across the training tokens. A higher mean suggests that the model consistently chooses this route, while a higher variance indicates variability in the routes learned for different tokens. We evaluate MoD trained on top of LoRA and MoD trained without k LoRA layers using the LLaMA 7B model on the ARC easy subset.

According to Figure 7, for the mean metric, we observe a reverse trend with respect to the sparsity score in Table 4. This aligns with our intuition that when the sparsity score of the current route is low, the routing value will be relatively larger than other routes. For the variance, we notice that when MoD is trained without k LoRA layers, it maintains a high variance throughout tuning. This suggests that many tokens are trained to select this route, but they are dynamically changing. When MoD is trained with LoRA, both the variance and mean levels stay low, indicating that the other two layers primarily contribute to the final ensemble logits. This suggests that the additional k trainable module within the MoD framework provides more predictive power to the ensemble layers, aligning with our analysis in §4.1.

C.2 MOD SPARSE ROUTING WITH DIFFERENT TOP-K VALUES

We also select a larger ensemble layer range to increase opportunities for early exit. We use $k = 4$ for this section, with results for BoolQ, OBQA, and MAWPS presented in Figure 8

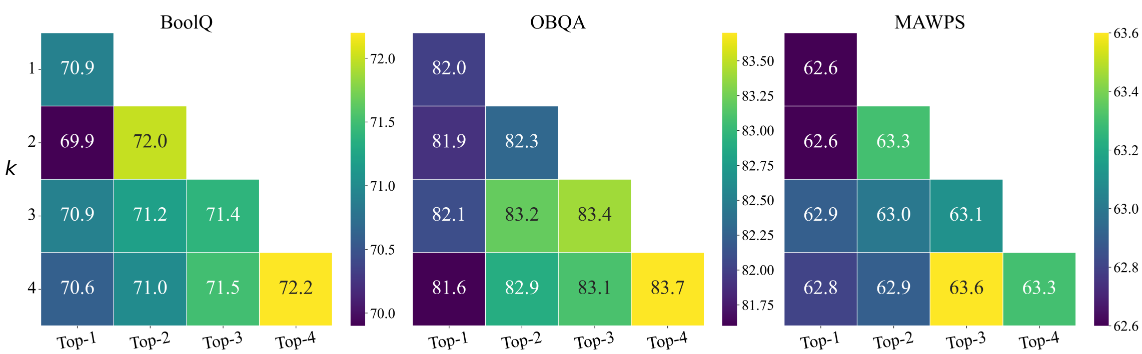

Figure 8: Accuracy scores for different k ensemble layer ranges and Top-K sparse routing values. Lighter colors indicate better performance. Results evaluated on BoolQ, OBQA, and MAWPS testset.