MoENet: Hidden Representations as Layer-Wise Experts for Adaptive Aggregation

Anonymous authors
Paper under double-blind review

Abstract

Network depth remains a pivotal yet sensitive design variable in deep learning, which intermediate layers can yield representations superior to the final layer, while excessive depth invites overfitting and optimization pathologies. We present **MoENet**, which systematically harnesses hidden representations across depth by treating each layer as a distinct expert and adaptively aggregating them via input-aware routing weights. We further introduce a sample-dependent softmax temperature, so that the sharpness of expert selection is modulated on a per-input basis. Extensive experiments demonstrate consistent gains across diverse backbones (e.g., ResNet, ConvNeXt) at only negligible additional computational and parameter overhead. Beyond accuracy, MoENet markedly exhibits stronger resilience to overfitting at extreme depths. Finally, incorporating MoENet into the representative baselines of seven additional image recognition benchmarks consistently yields significant improvements, underscoring the generality and broad applicability of our approach.

1 Introduction

Deep networks produce hierarchical representations, with different layers encoding information at different levels of abstraction. As an input propagates through successive nonlinear transformations, each hidden layer induces a distinct latent space, and the network can be viewed as progressively searching for representations adaptive to the downstream task. A fundamental question is thus raised: *is the final hidden representation necessarily the most suitable one for prediction?*

Different hidden layers may exhibit different degrees of adaptability with a given objective (Uselis & Oh, 2025). Increasing representation depth can be viewed as a sequential search through latent spaces, where each additional transformation implicitly bets that the newly constructed space is more favorable. This process is inherently lossy, i.e., a deeper transformation may enhance representation of task-relevant information while discarding cues preserved at preceding layers. This point indicates an intuitive explanation for why increasing representation depth helps in some settings but not others, and, more importantly, suggests that the final layer should not be the sole source of task-relevant information (Peters et al., 2018; Skean et al., 2025). Instead, representations across layers may contain complementary information that can be jointly exploited (Roh et al., 2024).

The above-specified point connects naturally to Mixture-of-Experts (MoE) (Jacobs et al., 1991), which involves routing to adaptively determine how experts contribute to a prediction. Existing MoE architectures introduce *additional* parameterized experts and combine them via dense or sparse routing (Vahidi et al., 2026; Dai et al., 2024; Fedus et al., 2022). We ask a different question: *can the hierarchical representations already produced by a standard network themselves serve as experts?* Here each hidden layer represents a level of abstraction, and the router no longer selects which module performs computation, but how much each layer-specific *representations* should contribute.

ELMo (Peters et al., 2018) pioneered the adaptive integration of all layers through task-aware weights, but this line of research has been explored predominantly in NLP, treating intermediate representations as adaptive experts remains relatively less explored in computer vision. This is particularly relevant for visual recognition, where continuous high-dimensional signals yield semantic information distributed across multiple abstraction levels (Chen et al., 2025). Rather than assuming the deepest representation subsumes all useful information, we investigate whether different depths can be explicitly treated as complementary feature extractors (experts).

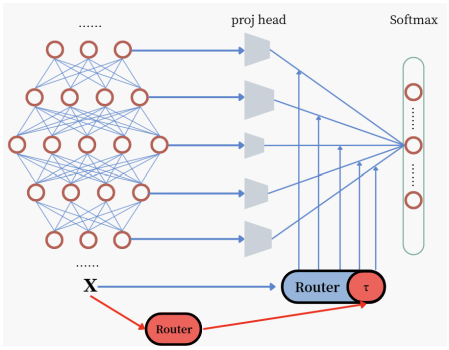
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We propose **MoENet** as a framework that treats hidden representations from different layers as representation experts and dynamically aggregates the experts via an MoE mechanism. The representation hierarchy itself becomes the expert space, while projection heads align heterogeneous representations into a common space. We also consider the temperature of the router as a parameter and have designed a strategy of additional router training. An L_1 penalty on the router parameters encourages sparse expert utilization.

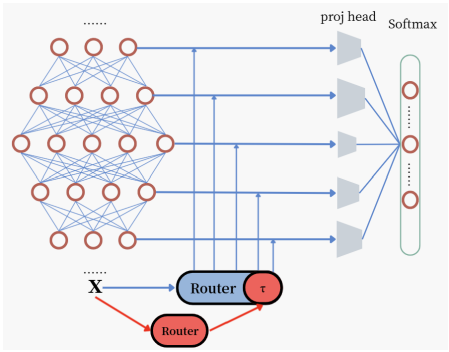
Our contributions are summarized as follows: Firstly, extensive experiments show that MoENet consistently improves a broad range of recognition baselines. Beyond accuracy, adaptive depth-wise aggregation improves robustness to overfitting in very deep networks. Secondly, further analysis of the learned weights reveals a sparse distribution. Those dominant experts exhibit substantially larger sample-wise variation in routing weights than the others, showing that fine-grained and input-dependent allocation matters. Finally, MoENet serves as a lightweight plug-in into the encoding or representation layers of state-of-the-art vision models, yielding consistent gains across seven challenging recognition tasks.

2 Related Work

Intermediate Representation Utilization. Whether the final layer always provides the most task-relevant representation is a long-standing question. ELMo (Peters et al., 2018) provided a seminal example in NLP, showing that different layers of a pretrained bidirectional language model capture complementary linguistic information and aggregating all layers with task-specific weights. However, these weights are static task-level parameters shared by all inputs. Complementary mechanisms have also been studied. Specifically, Deeply-Supervised Nets (DSN) (Lee et al., 2015) attach auxiliary heads to intermediate layers for direct supervision, BranchyNet (Teerapittayanon et al., 2016) allows confident samples to exit early, and LayerDrop (Fan et al., 2020) randomly drops layers during training to improve the robustness of different depth pathways. Recent analyses (Skean et al., 2025) further introduce representation-level measures (e.g., matrix-based entropy) and find that intermediate layers can be more transferable than subsequent ones in computer vision tasks. Collectively, these results suggest that useful information is not necessarily concentrated in the final layer, yet *adaptive* aggregation of features across layers remains underexplored in computer vision.



(a) Projection before weighting



(b) Weight before projection

Figure 1: Two architectures of MoENet layer-wise mixture.

Mixture-of-Experts and Dynamic Routing. MoE (Jacobs et al., 1991) combines multiple parameterized experts through input-dependent gating. Modern architectures use routing either to combine experts or to sparsely activate a subset of experts (Vahidi et al., 2026). Deep Mixture-of-Experts (DMoE) (Eigen et al., 2014) places experts and gates at successive layers. Nevertheless, its experts remain independently parameterized modules rather than representations naturally produced by different layers. Switch Transformer (Fedus et al., 2022) places parallel FFN experts within a layer and selects one per token, enabling parameter scaling at near-constant computation, with load-balancing losses preventing collapse. MICRO (AlKhamissi et al., 2026) decomposes Transformer layers into brain-inspired expert modules. In contrast, our experts are not separately instantiated modules but the layer-specific representations already produced by a conventional backbone.

A related line dynamically allocates computation across layers (representation depth), i.e., Mixture-of-Depths (MoD) (Raposo et al., 2024) uses routers to decide which tokens undergo deeper computation, reducing cost. Unlike MoD, our goal is not to skip computation but to retain and jointly exploit representations from multiple depths.

Overall, prior work explores intermediate representations via static aggregation, auxiliary supervision, early exiting, or dynamic depth allocation, while MoE methods route among parameterized modules or computational paths. MoENet bridges these views by treating hierarchical representations across layers as a unified pool of feature experts and performing sample-dependent layer-wise aggregation before prediction.

3 METHODOLOGY

As illustrated in Figure 1, MoENet converts the hierarchical representations produced by a deep neural network into a set of layer-wise representation experts and adaptively aggregates them for prediction in downstream tasks. Depending on the relative position of the projection head and the routing weight, we consider two closely related architectures.

3.1 DEEP NEURAL NETWORK BACKBONE

Consider a network that produces the representation of the n-th hidden layer as

$$\mathbf{f}_1 = \sigma(\mathbf{W}_1\mathbf{x}), \qquad \mathbf{f}_n = \sigma(\mathbf{W}_n\mathbf{f}_{n-1}), \quad n = 2, \dots, L, \tag{1}$$

with the input $\mathbf{x}$ regarded as layer 0. equation 1 only exposes the hierarchical dependency. In practice, MoENet attaches to architectures with convolutional blocks, residual connections, normalization and attention. The key observation is that $\{\mathbf{f}_1, \dots, \mathbf{f}_L\}$ forms a *hierarchy* of latent spaces rather than a sequence in which only the last one is useful. We therefore define a candidate expert set $\mathcal{F} = \{\mathbf{f}_{i_1}, \dots, \mathbf{f}_{i_M}\} \subseteq \{\mathbf{f}_1, \dots, \mathbf{f}_L\}$, where M is the number of depth experts. $\mathcal{F}$ does not have to contain every hidden layer. The granularity of the candidate depth set is itself a design variable.

3.2 DEPTH-WISE REPRESENTATION EXPERTS

Hidden representations from different layers have different dimensionalities and semantic scales, so direct summation is ill-defined. We introduce a layer-specific projection head $P_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_z}$ and consider two forms, corresponding to Figure 1a and 1b:

$$\mathbf{z} = \sum_{\mathbf{f}_i \in \mathcal{F}} \alpha_i P_i(\mathbf{f}_i), \qquad \mathbf{z} = \sum_{\mathbf{f}_i \in \mathcal{F}} P_i(\alpha_i \mathbf{f}_i). \tag{2}$$

The former representation is first projected into the common latent space and then subsequently modulated by the routing weight. For the latter one, a weight is first applied to the original hidden representation and the weighted feature is subsequently transformed by the projection head. When P_i is nonlinear, $P_i(\alpha_i \mathbf{f}_i) \neq \alpha_i P_i(\mathbf{f}_i)$, so the two representations are genuinely distinct. The projection head aligns heterogeneous representations into a common additive space and provides a task-adaptive transformation before integration. The aggregated $\mathbf{z} \in \mathbb{R}^{d_z}$ is mapped to predictions by a shared head:

$$\hat{\mathbf{y}} = C(\mathbf{z}), \qquad C : \mathbb{R}^{d_z} \to \mathbb{R}^K, \tag{3}$$

where C denotes a shared prediction head and K is the number of target classes.

Sample-dependent Weight Router. For visual signals, different samples may require different levels of abstraction, while different visual structures within the same task may be better represented at different depths. A fixed layer-weighting scheme cannot explicitly model such sample-level variation in routing weights. Motivated by this continuous and high-dimensional nature, MoENet introduces a router that maps the input $\mathbf{x}$ to a distribution over the candidate experts at various representation depths, so that the contribution of each expert is determined on a per-sample basis. Simply, we use a single-hidden-layer router:

$$\mathbf{r}(\mathbf{x}) = \mathbf{W}_{r2}\sigma(\mathbf{W}_{r1}\mathbf{x}), \tag{4}$$

where $\sigma(\cdot)$ denotes the activation and $\mathbf{W}_{r1}$ and $\mathbf{W}_{r2}$ are learnable router parameters. The output dimensionality of $\mathbf{r}$ equals the number of candidate experts M.

The routing weights are obtained by a temperature-scaled softmax function:

$$\boldsymbol{\alpha} = \mathrm{Softmax}\left(\frac{\mathbf{r}(\mathbf{x})}{\tau(\mathbf{x})}\right), \qquad \alpha_i \geq 0, \qquad \sum_{i=1}^{M} \alpha_i = 1. \tag{5}$$

Where $\tau(\mathbf{x})$ is the trainable temperature parameter that will be discussed later. The routing vector $\boldsymbol{\alpha}$ therefore represents the relative contribution of different abstraction levels to the prediction of a given input.

Input-adaptive Temperature Router. While routing weights determine which experts are preferred, whereas the temperature determines how decisively the router makes this choice. With a fixed temperature τ, all inputs share the same routing sharpness. It is implicitly assumed that different samples have comparable uncertainty about the appropriate representation depth, which may be inappropriate for the visual recognition task. A clear and well-resolved image may support a highly concentrated routing distribution, whereas a blurry, occluded, or semantically ambiguous image may benefit from a smoother combination of multiple experts at various representation depths. We therefore make the temperature itself input-dependent. Specifically, we introduce a separate temperature router:

$$g_\tau(\mathbf{x}) = \mathbf{W}_{\tau 2}\sigma(\mathbf{W}_{\tau 1}\mathbf{x}), \tag{6}$$

where $\mathbf{W}_{\tau 1}$ and $\mathbf{W}_{\tau 2}$ parameterize a lightweight single-hidden-layer network. To guarantee a positive temperature, we parameterize it as

$$\tau(\mathbf{x}) = \tau_{\min} + (\tau_{\max} - \tau_{\min})\ \mathrm{Sigmoid}(g_\tau(\mathbf{x})), \qquad 0 < \tau_{\min} < \tau_{\max}, \tag{7}$$

Because τ and the expert weights are coupled, joint training converges fast but plateaus later. Therefore, we freeze τ and its router once validation accuracy stagnates for several epochs, while the weights are updated continuously. The bounded range keeps τ stable and interpretable, interpolating between concentrated and smooth routing.

3.3 Collaborative and Responsibility-weighted Training

We consider two training mode: soft training and Expert-wise Training.

Soft Training. In soft training, all candidate experts jointly contribute to the final prediction. Let $\hat{\mathbf{y}}_i = C_i\left(P_i(\mathbf{f}_i)\right)$ denote the prediction of the i-th expert. The main training objective is

$$\mathcal{L}_{\mathrm{soft}} = \ell\left(\mathbf{y}, \sum_{i=1}^{M} \alpha_i \hat{\mathbf{y}}_i\right), \tag{8}$$

where $\ell(\cdot,\cdot)$ denotes the task loss. Under this objective, the experts cooperate to minimize a shared prediction error. Consequently, the gradient received by each expert depends on the residual of the aggregated prediction, introducing explicit coupling among expert outputs.

Expert-wise Training. We additionally consider a responsibility-weighted objective in which each expert is evaluated according to its own prediction: $\mathcal{L}_{\mathrm{ew}} = \sum_{i=1}^{M} \alpha_i \ell\left(\mathbf{y}, \hat{\mathbf{y}}_i\right)$. Compared with equation 8, each expert is directly optimized according to its own prediction error, with α_i determining its effective responsibility for the current sample. Therefore, this competitive learning approach encourages the winning expert to take all gradients. Note that there is no guarantee that this approach will be at least no worse than using the model with features from the last layer. Also, the experiments in this paper are mainly hold with soft training. In Section 4, we will discuss this in detail.

3.4 Depth-Coupled Gradient Analysis

A key property distinguishes MoENet from conventional mixture-of-experts architectures. In a standard MoE, experts are typically parameter-independent modules. If θ_i denotes the parameters of the i-th expert, then for $i \neq j$, $\frac{\partial \hat{\mathbf{y}}_j}{\partial \theta_i} = 0$.

In MoENet, however, the experts are not independent networks. They are representations produced by the same hierarchical backbone. Therefore, a expert that produces a deeper representation depends on all preceding representations. For $j > i$,

$$\frac{\partial \mathbf{f}_j}{\partial \mathbf{f}_i} = \prod_{t=i+1}^{j} \frac{\partial \mathbf{f}_t}{\partial \mathbf{f}_{t-1}} \neq 0 \tag{9}$$

in general. Consequently, the experts remain structurally coupled during optimization, regardless of whether the soft or expert-wise objective is used.

We first analyze the soft objective, which is the primary training scheme in MoENet. Let $\hat{\mathbf{y}} = \sum_{i=1}^{L} \alpha_i \hat{\mathbf{y}}_i$, $\mathcal{L}_{\text{soft}} = \ell(\mathbf{y}, \hat{\mathbf{y}})$. For clarity, we analyze the expert pathway while treating the routing weights α_i as fixed with respect to the hidden representations. Defining $\mathbf{q} = \frac{\partial \mathcal{L}_{\text{soft}}}{\partial \hat{\mathbf{y}}}$, the gradient with respect to the k-th hidden representation is

$$\frac{\partial \mathcal{L}_{\text{soft}}}{\partial \mathbf{f}_k} = \sum_{i=k}^{L} \alpha_i \left(\frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{f}_k} \right)^{\top} \mathbf{q}. \tag{10}$$

Using the hierarchical dependency in equation 9 and equation 10 can be written as

$$\frac{\partial \mathcal{L}_{\text{soft}}}{\partial \mathbf{f}_k} = \sum_{i=k}^{L} \alpha_i \left(\frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{f}_i} \prod_{t=k+1}^{i} \frac{\partial \mathbf{f}_t}{\partial \mathbf{f}_{t-1}} \right)^{\top} \mathbf{q}. \tag{11}$$

Thus, the k-th hidden representation receives gradient contributions from every selected expert at depth $i \geq k$. A shallow representation therefore receives not only the signal associated with its own prediction, but also indirect gradients propagated through all deeper experts that depend on it. Unlike expert-wise training, these contributions arise from a common mixture-level error signal $\mathbf{q}$, explicitly coupling the retained experts through their joint prediction.

Importantly, the same structural property also holds under the expert-wise objective. Let $\ell_i = \ell(\mathbf{y}, \hat{\mathbf{y}}_i)$ and define $\mathbf{q}_i = \frac{\partial \ell_i}{\partial \hat{\mathbf{y}}_i}$. Then

$$\frac{\partial \mathcal{L}_{\text{ew}}}{\partial \mathbf{f}_k} = \sum_{i=k}^{L} \alpha_i \left(\frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{f}_k} \right)^{\top} \mathbf{q}_i. \tag{12}$$

Hence, both objectives induce cross-depth gradient coupling. Their difference lies in the supervision signal, where soft training uses a shared error signal generated by the aggregated prediction, whereas expert-wise training uses expert-specific error signals.

Implicit Regularization from Shared Representations. The coupling above has an important implication for optimization. Unlike independent experts in an MoE setting, whose expert-specific parameters can be updated largely independently, MoENet forces different layer-wise experts to share and jointly shape their upstream representations. A change in a shallow feature therefore affects multiple downstream prediction branches, causing their training signals to accumulate at the shared backbone.

To make this interaction explicit, define the contribution from expert i to the gradient at depth k as

$$\mathbf{g}_k^{(i)} = \left(\frac{\partial \hat{\mathbf{y}}_i}{\partial \mathbf{f}_k} \right)^{\top} \mathbf{q}_i, \qquad i \geq k, \tag{13}$$

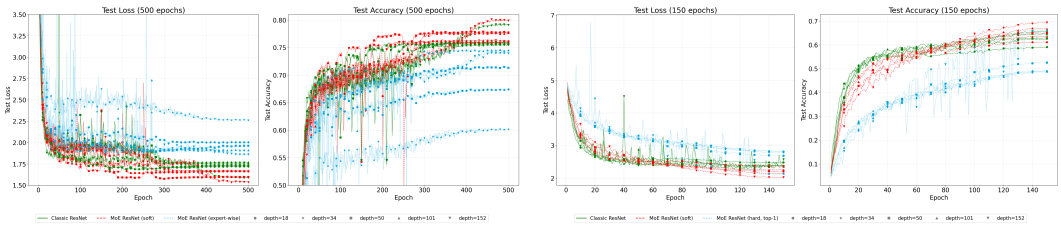
where $\mathbf{q}_i = \mathbf{q}$ for soft training and $\mathbf{q}_i = \frac{\partial \ell_i}{\partial \hat{\mathbf{y}}_i}$. for expert-wise Training. The resulting gradient is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{f}_k} = \sum_{i=k}^{L} \alpha_i \mathbf{g}_k^{(i)}, \tag{14}$$

and its squared norm is

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{f}_k} \right\|_2^2 = \sum_{i=k}^{L} \alpha_i^2 \left\| \mathbf{g}_k^{(i)} \right\|_2^2 + 2 \sum_{k \leq i < j \leq L} \alpha_i \alpha_j \left\langle \mathbf{g}_k^{(i)}, \mathbf{g}_k^{(j)} \right\rangle. \tag{15}$$

The second term explicitly characterizes the interaction among supervision signals originating from different representation depths. Positive inner products indicate gradient alignment and reinforce shared representations, whereas negative inner products indicate gradient conflict. This depth-coupled optimization provides a structural mechanism through which multiple experts jointly shape shared upstream representations rather than being optimized as isolated experts. We later empirically examine whether this mechanism mitigates routing collapse and whether its effect becomes more pronounced in very deep networks, where conventional models may suffer from overfitting.



(a) Training trajectory of CIFAR100 (b) Training trajectory of Tiny-Imagenet-200

Figure 2: Performance comparison between Resnet and MoENet on diverse depth.

Table 1: Transferability of MoENet across vision tasks. For each task, we select a representative and reproducible baseline and evaluate the corresponding baseline and MoENet-enhanced models.

Domain	Metric	Dataset	Baseline	Baseline_Result	MoENet	Δ
Edge Detection	ODS / OIS / AP	BSDS500	RCF	0.798 / 0.815 / 0.850	0.806 / 0.831 / 0.865	0.008 / 0.016 / 0.015
Object Detection	mAP	COCO	Faster R-CNN R50-FPN	37.4	38.3	0.9
Semantic Segmentation	mIoU	Cityscapes	DeepLabv3+	82.1	82.9	0.8
Remote Sensing Segmentation	mIoU	LoveDA	SegFormer-MiT-B5	67.08	68.11	1.03
3D Scene Understanding	mIoU	ScanNet	SpUNet34C	73.6	74.4	0.8
Self-Supervised Learning	Top-1 Acc.	Tiny-ImageNet-200	DINO ViT-B/16	62.6	64.9	2.3
Semi-Supervised Learning	Top-1 Acc.	Tiny-ImageNet-200 (10% labels)	SimMatchV2	63.1	66.0	2.9

3.5 Training Objective

The full objective combines the task loss with sparsity penalty on the router (consider soft training):

$$\mathcal{L} = \mathcal{L}_{\text{soft}} + \lambda_1 \Omega_{\text{route}} + \lambda_2 \Omega, \qquad \Omega_{\text{route}} = \sum_{w \in \theta_{\text{route}}} |w|, \qquad \Omega = \sum_{w \in \theta} ||w||_2^2 \tag{16}$$

where θ_{route} are the expert weight router parameters. Appropriate use of this optional L_1 penalty encourages sparse utilization of experts, and Ω denotes L_2 regularization of all parameters θ.

4 Experiments

We evaluate MoENet on three widely used image recognition benchmarks, CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and Tiny-ImageNet-200 (Deng et al., 2009; Le et al., 2015). Unless otherwise specified, we use a batch size of 512 on CIFAR-10 and CIFAR-100, with an L_1 regularization coefficient of 5×10^{-3}. On Tiny-ImageNet-200, we use a batch size of 256, with an L_1 coefficient of 5×10^{-3}. All models are optimized using SGD with momentum 0.9. The initial learning rate is set to 0.1 and adjusted using cosine annealing, with a weight decay of 10^{-4}. We choose all blocks of stage 4 as the candidate expert set for Resnet-101 (He et al., 2016) and ConvNeXt Liu et al. (2022b). All experiments are conducted on NVIDIA GeForce RTX 4090 GPUs. The conclusions drawn below are based on comparisons that pass our statistical significance test with p-value <0.001. Additional implementation details are provided in the Supplementary Material.

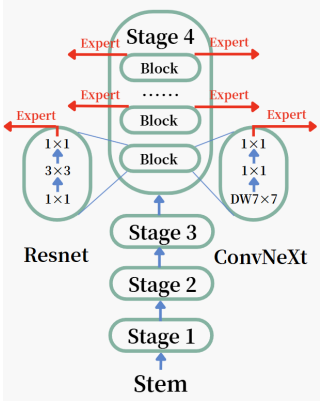


Figure 3: Candidate experts.

4.1 Main Results

We first examine whether MoENet consistently improves standard deep visual backbones. Table 2 summarizes the results and Figure 2 shows the training trajectory.

MoENet improves the performance of both ResNet and ConvNeXt across the evaluated datasets, demonstrating that depth-wise adaptive aggregation is not specific to a particular backbone family. For ResNet-101, the proposed configuration improves the accuracy from 94.36%to 95.12% on CIFAR-10 and from 74.97% to 78.53% on CIFAR-100, corresponding to absolute gains of 0.76 and 3.56 percentage points, respectively. On ImageNet, the same configuration improves accuracy from

Table 2: Main results, ablation study, and definitions of MoENet experimental variants.

(a) Main results and ablation study of MoENet on CIFAR-10, CIFAR-100 and Tiny-ImageNet-200.

	CIFAR-10			CIFAR-100			Tiny-ImageNet-200		
Method	Acc	Params	Time	Acc	Params	Time	Acc	Params	Time
ResNet backbone									
ResNet	0.9436	42.51M	1.00	0.7497	42.70M	1.00	0.6509	42.91M	1.00
Mixfirst	0.9503	46.45M	1.06	0.7853	46.54M	1.06	0.6675	46.65M	1.04
Mixafter	0.9512	46.48M	1.06	0.7785	46.85M	1.06	0.6773	47.27M	1.05
w/o proj head	0.9493	42.54M	1.01	0.7696	42.89M	1.01	0.6655	43.28M	1.00
Static weight	0.9503	46.47M	1.06	0.7740	46.84M	1.06	0.6717	47.26M	1.05
expert-wise	0.8899	46.48M	1.06	0.7513	46.85M	1.06	0.6201	47.27M	1.05
Fixed τ	0.9507	46.47M	1.06	0.7785	46.84M	1.06	0.6753	47.26M	1.05
w/o L_1	0.9483	46.48M	1.06	0.7768	46.85M	1.06	0.6557	47.27M	1.05
ConvNeXt backbone									
ConvNeXt	0.9404	7.68M	1.00	0.7612	7.73M	1.00	0.5776	7.79M	1.00
Mixfirst	0.9426	8.68M	1.09	0.7647	8.77M	1.09	0.6217	8.88M	1.05
Mixafter	0.9435	8.71M	1.09	0.7697	9.08M	1.10	0.6145	9.50M	1.07
w/o proj head	0.9443	7.69M	1.01	0.7692	7.78M	1.01	0.5825	7.80M	1.00
Static weight	0.9407	8.70M	1.09	0.7638	9.07M	1.10	0.5930	9.49M	1.07
expert-wise	0.9346	8.71M	1.09	0.7253	9.08M	1.10	0.5692	9.50M	1.07
Fixed τ	0.9421	8.70M	1.08	0.7521	9.07M	1.09	0.5959	9.49M	1.06
w/o L_1	0.9425	8.71M	1.09	0.7534	9.08M	1.10	0.6001	9.50M	1.07

(b) Explanations of some ablation variants.

Method	Explanation
Mixfirst / Mixafter	Correspond to the two architectures in Figure 1a and 1b, respectively
Expert-wise	Trained with the expert-wise loss function
Static weight	Static learnable expert weights without the dynamic router
Fixed τ	Temperature τ is fixed as a hyperparameter during training

65.09% to 67.73%, yielding a 2.64-point gain. Similar improvements are observed for ConvNeXt, although the magnitude varies with the dataset, suggesting that the benefit of MoENet is robust. Moreover, all the improvements are in a relatively insignificant increase in parameter count and time, indicating the strong Pareto balance of our model.

An additional experiment investigates the interaction between model depth and training dynamics. On CIFAR-100 and ImageNet, we compare a standard ResNet with our designed variant (ResNet+MoENet), while the model depth is set to 18, 34, 52, 101, and 152, respectively, over 500 epochs. Across all five model depths on both datasets, MoENet achieves a lower test loss in the later stages of training and ultimately surpasses the corresponding vanilla ResNet in accuracy. Interestingly, MoENet is sometimes slightly behind the baseline during the early training phase, but continues to improve steadily after the baseline begins to saturate. This delayed advantage is consistent with the fact that MoENet introduces additional projection parameters and requires the joint optimization of multiple depth-specific representations and their routing weights. Thus, sufficiently long training appears to be important for allowing the representation experts and the router to stabilize. Importantly, the observed behavior should be interpreted as an empirical optimization phenomenon rather than evidence that the vanilla backbone is intrinsically under-trained.

We further examine its transferability across a diverse set of computer vision tasks and learning paradigms with substantially different output structures and representation requirements. Our guiding principle is to introduce MoENet as a lightweight plug-in on top of an existing encoder or backbone while leaving the task-specific prediction head unchanged. We consider edge detection Arbelaez et al. (2010), object detection Lin et al. (2014), semantic segmentation Cordts et al. (2016), remote-sensing segmentation Wang et al. (2021), 3D scene understanding Dai et al. (2017), self-supervised representation learning, and semi-supervised image classification.

Table 1 summarizes the results. Across all seven task settings, enhancing the baseline by incorporating MoENet consistently improves the reported evaluation metrics Liu et al. (2017); Ren et al.

(2016); Chen et al. (2018); Xie et al. (2021); Liu et al. (2022a); Caron et al. (2021); Zheng et al. (2023). Though consistent, gains are modest on detection, segmentation benchmarks (+0.8–1.03 mIoU/mAP) but pronounced under Self-SL, Semi-SL (+2.3–2.9 Top-1). This phenomenon shows that the usefulness of representation experts with varied depths is largely task agnostic and that MoENet can serve as a general-purpose representation-level plug-in across heterogeneous computer vision systems. Detailed insertion strategies and training protocols are provided in the appendix.

4.2 Architectural Analysis

In this section, we investigate the design choices of MoENet, including the placement and necessity of the projection heads, and the benefit of input-dependent routing. We additionally compare the soft and expert-wise training objectives to examine how different forms of expert supervision affect optimization. Also, probe the influence of the trainable temperature and the L_1 regularization term.

We consider ResNet-101 and ConvNeXt as representative backbones because of their broad adoption and, in particular, the prevalence of residual connections in modern vision architectures. The output dimensionality of projection heads is 1024. The results are presented in Table 2 and relevant analyses are as follow. Other results for additional depths are presented in the appendix.

Projection heads. At most of the time, the projection heads consistently improve the performance of MoENet across datasets and backbones. Removing the projection heads often causes a clear degradation in accuracy, suggesting that probably directly aggregating heterogeneous representations from different depths is less effective than first mapping them into a shared latent space. In contrast, the order of projection and weighting has only a minor effect. The two variants exhibit comparable performance, with the slightly better choice depending on the dataset, indicating that the effectiveness of MoENet does not critically depend on either ordering. We retain both as choices.

Dynamic versus static weighting. Replacing the input-dependent routing weights with static layer weights still improves over the original backbone, demonstrating that aggregating representations from multiple layers is itself beneficial. However, the static variant consistently underperforms the proposed dynamic weighting strategy across the evaluated datasets. This result supports the core motivation of MoENet that the usefulness of a layer-specific representation is input dependent.

Soft versus expert-wise training. A substantial performance gap is observed between the soft and expert-wise objectives. The latter follows competitive training, where each expert is optimized by its own prediction error, with the routing weight determining its effective responsibility. In our setting this induces an overly aggressive winner-take-all dynamic, a layer that gains a larger weight early receives a disproportionately large optimization signal while the remaining experts receive none. So the router can lock onto a single intermediate depth before the others are adequately trained. Further tracking the trajectory of weights finds that the selected expert typically becomes fixed within a few early epochs and dominates throughout the rest of training. Because the winner depends on random initialization and the early trajectory, deeper representations that would become more useful later are prematurely suppressed. Figure 2 show the objective also causes markedly stronger oscillations in test performance, which we attribute to the positive feedback of responsibility-weighted competition: small early quality differences are amplified by unequal gradient allocation, yielding unstable competition among under-trained representations. Consequently, expert-wise training falls behind both soft training and, on most datasets, the plain backbone baseline. Soft training instead preserves joint contributions from multiple experts and lets their representations improve cooperatively.

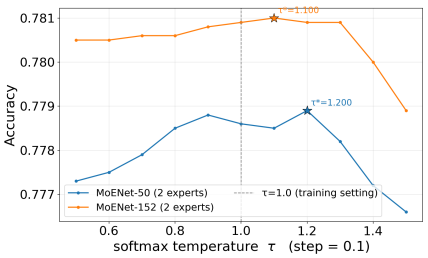


Figure 4: Sensitivity of the routing softmax temperature τ in CIFAR-100.

Necessity of Training τ. Learning τ dynamically indicates the essential point that an input-dependent $\tau(x)$ surpasses a fixed, hand-tuned τ. Viewing τ to a hyperparameter further reveals an extreme sensitivity, as Figure 4 perturbing it by as little as 0.1, the highest performance change can reach over 0.1%. Because such sensitivity leaves virtually no tolerance for manual tuning, τ is

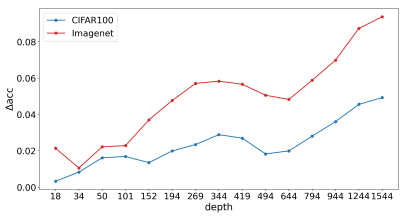


Figure 5: The performance gap between Resnet and MoENet with depth on CIFAR-100 and Imagenet-200.

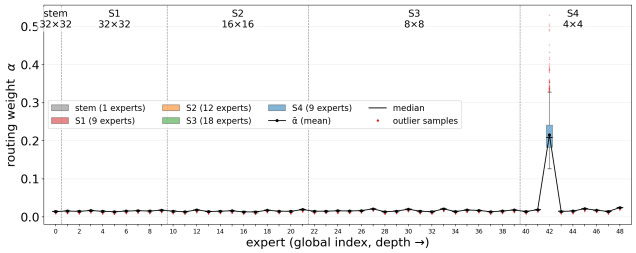


Figure 6: Resnet-50 per-expert routing weight $\bar{\alpha}$ (black line) and its distribution across samples (box) on CIFAR-100.

best treated as a learnable parameter determined implicitly by the model during training, especially when one possesses little prior knowledge of the task.

4.3 Robustness under Extreme Deep Network Settings

We examine the property, namely, the potential of MoENet to mitigate overfitting as representation depth increases. As shown in Figure 5, the performance gap between MoENet and the corresponding vanilla ResNet generally widens with increasing representation depth, although the trend is not strictly monotonic. This divergence becomes particularly pronounced in the deepest configurations, where on CIFAR-100, MoENet outperforms the vanilla ResNet by more than 0.05 points in accuracy, and even reach to almost 0.1 on Tiny-Imagenet-200. These results empirically support that MoENet appears to be more robust in mitigating the adverse effects associated with excessive network depth. Importantly, this robustness is not restricted to a particular depth configuration, but becomes increasingly evident as the backbone is pushed toward more extreme depths.

4.4 Expert Weight Analysis

As shown in Figure 6, while ResNet-50 is set as the backbone, we visualize the distribution of expert weights on CIFAR-100. Specifically, we consider all layers of the entire model as expert candidates. Results obtained using other model depths are provided in the appendix and lead to consistent conclusions.

The routing weights exhibit a clear concentration pattern that a few experts receive substantially larger weights than the remaining experts, and these dominant experts are consistently located in stage 4 but not always at the final layer. Shallow networks favor final-layer experts, and deep networks prefer middle-layer ones. The other experts receive smaller and comparatively uniform weights, revealing an intrinsic sparsification pattern in the learned routing distribution. Notably, this sparsity is not directly induced by the L_1 regularization alone, although the regularizer further promotes this sparsity and yields consistent performance improvements, as shown in Table 2.

Despite this concentration at the population level, the routing mechanism remains sample-aware. Experts assigned relatively small average weights exhibit only limited variation in α across samples, whereas the dominant experts show substantially larger sample-wise standard deviations. Combined with the performance gap between dynamic and static weighting in Table 2, this observation suggests that fine-grained and input-dependent routing is particularly important for the experts that contribute most strongly to the final prediction. Since these dominant experts can account for a substantial portion of the model's predictive behavior, allowing their weights to adapt at the sample level provides additional flexibility to exploit their strengths and further improve the final representation.

5 Conclusion

We presented MoENet, which rethinks network depth as a space of representation experts. Through structured routing adaptively aggregating depth-specific features, MoENet allows for input-aware collaboration of different levels. Experiments show consistent gains across backbones and stronger robustness at extreme depths. Its effectiveness across computer vision tasks further suggests it is a broadly useful lightweight plugin for improving representation quality across numerous baselines.

REFERENCES

Badr AlKhamissi, C. Nicolò De Sabbata, Greta Tuckute, Zeming Chen, Martin Schrimpf, and Antoine Bosselut. Mixture of Cognitive Reasoners: Modular Reasoning with Brain-like Specialization. In *International Conference on Learning Representations*, 2026. URL https://mlanthology.org/iclr/2026/alkhamissi2026iclr-mixture/.

Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Haoran Chen, Junyan Lin, Xinhao Chen, Yue Fan, Xin Jin, Hui Su, Jianfeng Dong, Jinlan Fu, and Xiaoyu Shen. Rethinking visual layer selection in multimodal llms. *arXiv e-prints*, pp. arXiv–2504, 2025.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision – ECCV 2018*, pp. 833–851, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01234-2.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2432–2443. IEEE, 2017.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. 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)*, pp. 1280–1297, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning Factored Representations in a Deep Mixture of Experts. In *International Conference on Learning Representations*, 2014. URL https://mlanthology.org/iclr/2014/eigen2014iclr-learning/.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SylO2yStDr.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022. ISSN 1532-4435.

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.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 03 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4), 2009.

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In Guy Lebanon and S. V. N. Vishwanathan (eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 562–570, San Diego, California, USA, 09–12 May 2015. PMLR. URL https://proceedings.mlr.press/v38/lee15a.html.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Xinhai Liu, Xinchen Liu, Yu-Shen Liu, and Zhizhong Han. Spu-net: Self-supervised point cloud upsampling by coarse-to-fine reconstruction with self-projection optimization. *IEEE Transactions on Image Processing*, 31:4213–4226, 2022a.

Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5872–5881. IEEE, 2017.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11966–11976, 2022b. doi: 10.1109/CVPR52688.2022.01167.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202/.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

Yuji Roh, Qingyun Liu, Huan Gui, Zhe Yuan, Yujin Tang, Steven Euijong Whang, Liang Liu, Shuchao Bi, Lichan Hong, Ed H Chi, et al. Levi: generalizable fine-tuning via layer-wise ensemble of different views. *arXiv preprint arXiv:2402.04644*, 2024.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann Lecun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/skean25a.html.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016. doi: 10.1109/ICPR.2016.7900006.

Arnas Uselis and Seong Joon Oh. Intermediate layer classifiers for ood generalization. In *International Conference on Learning Representations*, volume 2025, pp. 46059–46088, 2025.

Amirhossein Vahidi, Hesam Asadollahzadeh, Navid Akhavan Attar, Marie Moullet, Kevin Ly, Xingyi Yang, and Mohammad Lotfollahi. Dirmoe: Dirichlet-routed mixture of experts. In *International Conference on Learning Representations*, volume 2026, pp. 155806–155824, 2026.

Junjue Wang, Zhuo Zheng, Ailong Ma, Xiaoyan Lu, and Yanfei Zhong. Loveda: A remote sensing land-cover dataset for domain adaptive semantic segmentation. *arXiv preprint arXiv:2110.08733*, 2021.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

Mingkai Zheng, Shan You, Lang Huang, Chen Luo, Fei Wang, Chen Qian, and Chang Xu. Simmatchv2: Semi-supervised learning with graph consistency. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16386–16396. IEEE, 2023.

A SUPPLEMENTARY EXPERIMENTAL RESULTS

This chapter belongs to Appendix A and serves as a supplement to the experimental results in the main text's experiments section.

A.1 TRAINING TRAJECTORY OF CIFAR100

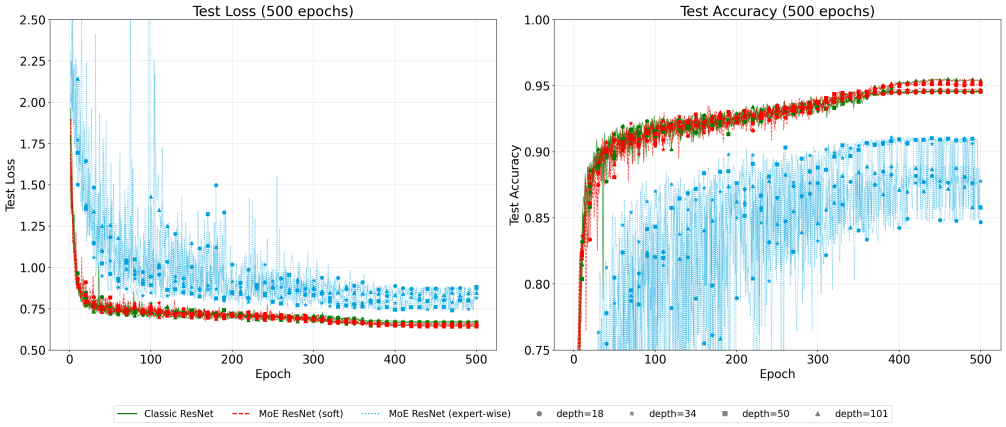


Figure 7: Training trajectory of CIFAR100.

A.2 EXPERT WEIGHT RESULTS VISUALIZATION

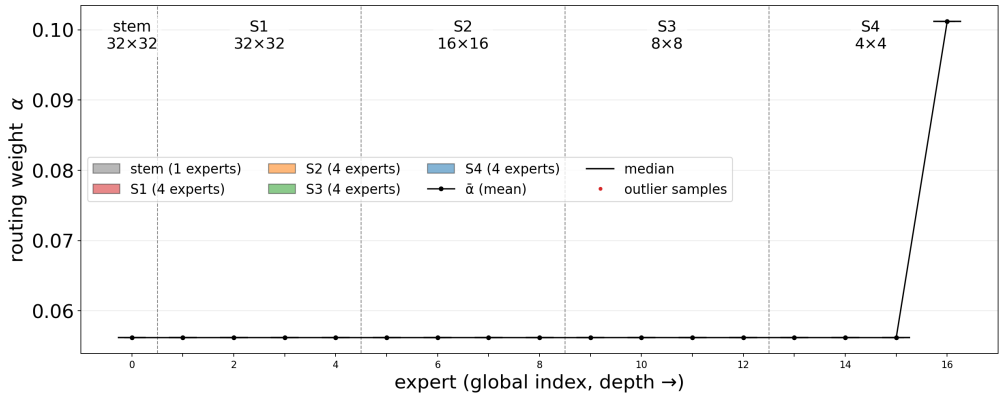


Figure 8: Resnet-18 per-expert routing weight $\bar{\alpha}$ (black line) and its distribution across samples (box) on CIFAR-100.

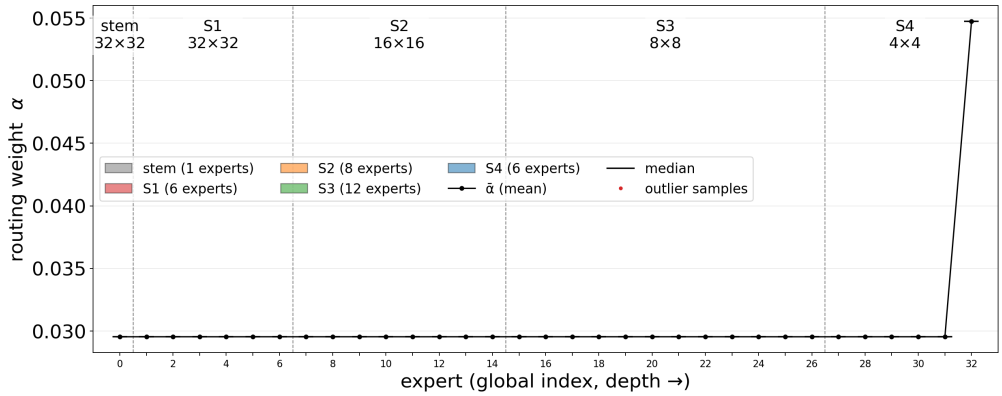


Figure 9: Resnet-34 per-expert routing weight $\bar{\alpha}$ (black line) and its distribution across samples (box) on CIFAR-100.

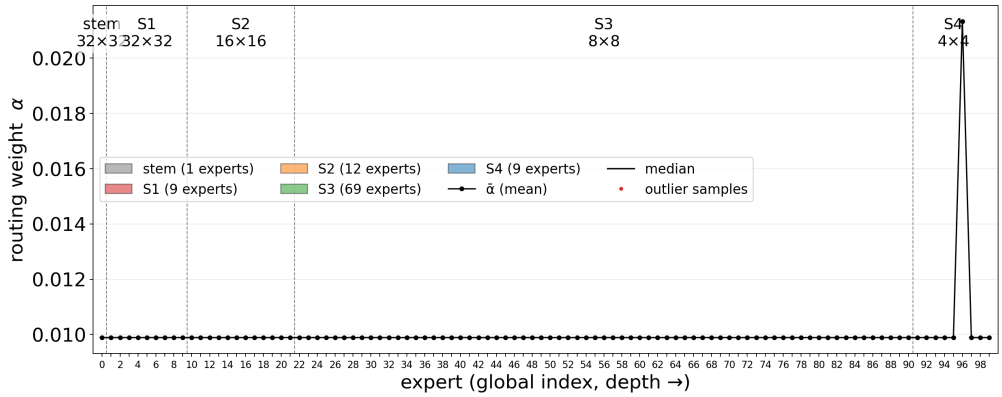


Figure 10: Resnet-101 per-expert routing weight $\bar{\alpha}$ (black line) and its distribution across samples (box) on CIFAR-100.

A.3 THE SENSITIVITY OF TEMPERATURE

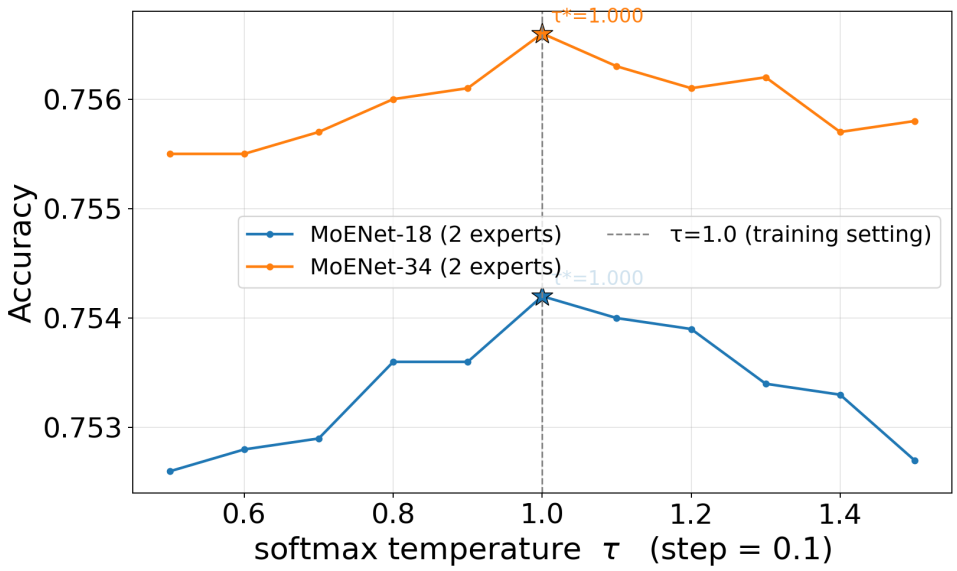


Figure 11: Sensitivity of the routing softmax temperature τ in CIFAR-100 (Resnet-18, -34).

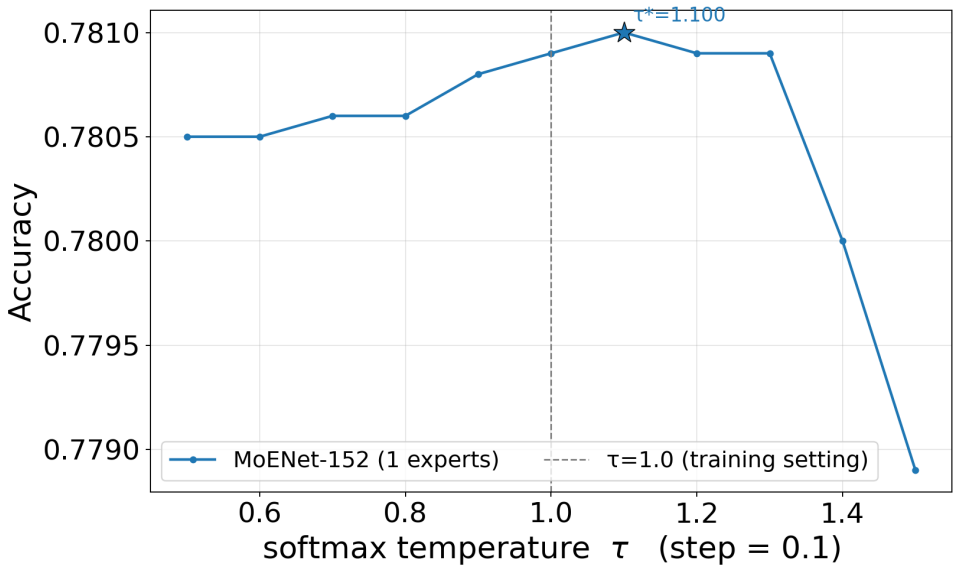


Figure 12: Sensitivity of the routing softmax temperature τ in CIFAR-100 (Resnet-152).

Table 3: Main results and ablation study of MoENet with ResNet-34 backbone.

	CIFAR-10			CIFAR-100			Tiny-ImageNet-200		
Method	Acc	Params	Time	Acc	Params	Time	Acc	Params	Time
ResNet-34	0.9405	21.29M	1.00	0.7455	21.34M	1.00	0.6450	21.39M	1.00
Mixfirst	0.9445	22.29M	1.05	0.7675	22.38M	1.05	0.6550	22.49M	1.03
Mixafter	0.9452	22.33M	1.05	0.7620	22.70M	1.05	0.6630	23.11M	1.04
w/o proj head	0.9437	21.30M	1.01	0.7555	21.39M	1.01	0.6535	21.49M	1.00
Static weight	0.9445	22.32M	1.05	0.7582	22.69M	1.05	0.6582	23.10M	1.04
expert-wise	0.8935	22.33M	1.05	0.7470	22.70M	1.05	0.6190	23.11M	1.04
Fixed τ	0.9448	22.32M	1.05	0.7620	22.69M	1.05	0.6614	23.10M	1.04
w/o L_1	0.9428	22.33M	1.05	0.7606	22.70M	1.05	0.6470	23.11M	1.04

Table 4: Main results and ablation study of MoENet with ResNet-50 backbone.

	CIFAR-10			CIFAR-100			Tiny-ImageNet-200		
Method	Acc	Params	Time	Acc	Params	Time	Acc	Params	Time
ResNet-50	0.9425	23.53M	1.00	0.7480	23.71M	1.00	0.6485	23.92M	1.00
Mixfirst	0.9477	27.46M	1.09	0.7765	27.56M	1.09	0.6617	27.66M	1.06
Mixafter	0.9485	27.50M	1.09	0.7705	27.87M	1.09	0.6707	28.28M	1.07
w/o proj head	0.9468	23.56M	1.01	0.7625	23.90M	1.01	0.6599	24.29M	1.00
Static weight	0.9477	27.49M	1.09	0.7663	27.86M	1.09	0.6655	28.27M	1.07
expert-wise	0.8925	27.50M	1.09	0.7496	27.87M	1.09	0.6205	28.28M	1.07
Fixed τ	0.9481	27.49M	1.09	0.7705	27.86M	1.09	0.6689	28.27M	1.07
w/o L_1	0.9458	27.50M	1.09	0.7689	27.87M	1.09	0.6517	28.28M	1.07

A.4 Robustness under Extreme Deep Network Setting (ConvNeXt)

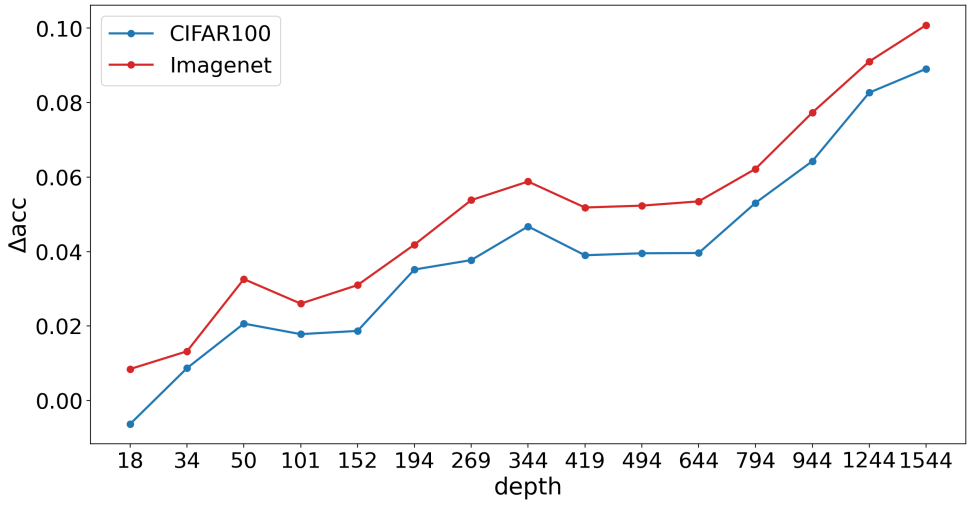


Figure 13: The performance gap between ConvNeXt and MoENet with depth on CIFAR-100 and Imagenet-200.

Table 5: Main results and ablation study of MoENet with ResNet-152 backbone.

	CIFAR-10			CIFAR-100			Tiny-ImageNet-200		
Method	Acc	Params	Time	Acc	Params	Time	Acc	Params	Time
ResNet-152	0.9448	58.16M	1.00	0.7518	58.35M	1.00	0.6538	58.55M	1.00
Mixfirst	0.9523	62.09M	1.05	0.7913	62.19M	1.05	0.6726	62.29M	1.03
Mixafter	0.9533	62.14M	1.05	0.7841	62.51M	1.05	0.6831	62.92M	1.04
w/o proj head	0.9512	58.19M	1.01	0.7743	58.54M	1.01	0.6704	58.92M	1.00
Static weight	0.9523	62.13M	1.05	0.7793	62.50M	1.05	0.6771	62.91M	1.04
expert-wise	0.8888	62.14M	1.05	0.7534	62.51M	1.05	0.6208	62.92M	1.04
Fixed τ	0.9528	62.13M	1.05	0.7841	62.50M	1.05	0.6809	62.91M	1.04
w/o L_1	0.9502	62.14M	1.05	0.7823	62.51M	1.05	0.6596	62.92M	1.04

A.5 Supplementary results of main experiment and ablation study of diverse depth

B MoENet Transferability: Insertion Strategies and Experimental Protocols

This appendix specifies, for each task in Table 1, (i) which representations serve as depth experts, (ii) where MoENet is inserted and how it interacts with the task-specific head, (iii) the training protocol, and (iv) the evaluation protocol. All configurations are chosen so that the comparison isolates the effect of MoENet alone.

B.1 General Insertion Contract

Let $f_i(\boldsymbol{x})$ denote the representation produced at depth level i of the shared backbone/encoder. MoENet obeys three structural rules, which we keep identical across all seven tasks:

1. **Experts are depths, not modules.** $\mathcal{E} = \{f_i(\boldsymbol{x})\}_{i\in\mathcal{D}}$, where $\mathcal{D}$ indexes stage / block / layer levels of one shared backbone. MoENet introduces no new expert networks; its experts are re-used backbone representations.
2. **The router reads only the stem.** $\alpha(\boldsymbol{x}) = \mathrm{softmax}\big(\mathrm{Router}(\mathrm{GAP}(f_{\mathrm{stem}}(\boldsymbol{x})))\big)$, where f_{stem} is strictly shallower than every expert. No f_i is ever fed to the router. This keeps the gating weights decoupled from the expert representations.
3. **The task head is untouched.** The aggregated representation is fed into the original task-specific head (FPN + RPN + RoI, ASPP + decoder, All-MLP decoder, DINO projection head, SimMatchV2 classifier), whose structure, number of classes, and loss form are preserved.

For classification we aggregate globally-pooled vectors; for dense prediction we **preserve spatial maps**: expert maps are first aligned to a single anchor resolution by bilinear upsampling and 1×1 convolution channel projection, then combined as $\mathbf{F}_{\text{enh}} = \sum_i \alpha_i \cdot P_i(\uparrow f_i)$, where α_i is broadcast over spatial positions (one gate vector per image, not per pixel). We use **MixFirst** (gate before projection) throughout, since every baseline here already shares a single prediction head; MixLast would replicate the head $|\mathcal{E}|$ times and confound depth routing with extra capacity.

B.2 Edge Detection: RCF on BSDS500

Experts and insertion. RCF Liu et al. (2017) attaches a 1×1 side branch to every convolution of VGG16 and removes `pool5`. We group the side branches into the five VGG stages (stride $1, 2, 4, 8, 16$; channels $64, 128, 256, 512, 512$) and treat the five stage-aggregated side features as depth experts (stage-intra accumulation is retained, as RCF validates it). The router takes GAP over `pool1` (64-D). Expert maps are projected to a common width and upsampled to full input resolution, then α-weighted and passed to RCF's original fusion layer and sigmoid side heads.

Training protocol. ImageNet-pretrained VGG16; SGD, momentum 0.9, weight decay 2×10^{-4}, base lr 10^{-6}, divided by 10 every 10k iterations, 40k iterations total; minibatch 10. Loss is the annotator-robust weighted cross-entropy of RCF (contested pixels with $0 < y \leq \eta$ are ignored), summed over the five side outputs plus the fused output. Optionally we add the router regularizer $\lambda_{\mathrm{L1}}\|\alpha\|_1$; λ_{L1} is ablated and fixed beforehand, never tuned per dataset.

Evaluation protocol. We report ODS / OIS / AP using the standard BSDS500 pipeline: upsample to the original resolution, (optional) thinning, and bipartite pixel matching with tolerance maxDist $= 0.0075$; ODS uses one dataset-wide threshold, OIS per-image thresholds, AP averages over the threshold sweep. We use the `test` split, and fix the NMS/thinning settings, input resize range, and single-scale testing across baseline and MoENet.

B.3 Object Detection: Faster R-CNN R50-FPN on COCO

Experts and insertion. ResNet-50 yields C_2–C_5 at strides $4, 8, 16, 32$ with $256, 512, 1024, 2048$ channels; FPN builds P_2–P_6 by 1×1 lateral projection followed by a top-down pathway. Because FPN must preserve the multi-scale pyramid, MoENet is inserted as a *single-point* enhancement rather than a pyramid collapse: we take C_3, C_4, C_5 as three experts (C_2 is ablated separately, since stride 4 mainly benefits small objects and semantic depth routing gains little there), project each to 256 channels, upsample them to stride 16, and form $\mathbf{F}_{\text{enh}} = \sum_i \alpha_i P_i$. $\mathbf{F}_{\text{enh}}$ then replaces M_5 and feeds the *unchanged* FPN top-down chain, so P_2–P_6, the RPN anchors and the RoI head are all preserved. The router reads GAP of the ResNet stem (`conv1` $\rightarrow$ `bn1` $\rightarrow$ `relu` $\rightarrow$ `maxpool`, 64-D).

Training protocol. Detectron2 `faster_rcnn_R_50_FPN_1x`: 90k iterations, base lr 0.02 with 1k-iteration linear warmup, stepped down by 0.1 at 60k and 80k; IMS_PER_BATCH $= 16$; SGD momentum 0.9, weight decay 10^{-4}; ImageNet-1K initialized R-50; horizontal flip and scale jitter (short side 800, long side ≤ 1333). Losses are the standard four terms (RPN cls/reg, RoI cls/reg). We use the $1\times$ schedule as the main comparison and report $3\times$ (270k iterations) only as a supplementary point, so schedule gains are not mistaken for MoENet gains.

Evaluation protocol. COCO box mAP averaged over IoU thresholds $0.50 : 0.95$, on `val2017`, with identical inference resize, score threshold, NMS and `max_detections` for baseline and MoENet. No multi-scale testing; no pretraining beyond ImageNet-1K.

B.4 Semantic Segmentation: DeepLabv3+ on Cityscapes

Experts and insertion. With output stride 16, ResNet stages give C_2–C_5 ($256, 512, 1024, 2048$; strides $4, 8, 16, 16$). We use all four as depth experts, project to a common width, align to stride 16, and insert the α-weighted map *before* ASPP. This preserves both structural elements of DeepLabv3+: ASPP's multi-scale context and the decoder's 48-channel low-level skip from `conv2`. Placing the aggregation after ASPP would overwrite the dilated-rate diversity, and replacing the decoder concatenation would interfere with boundary recovery; both are therefore only used as ablations. Router input is GAP of the ResNet stem (64-D).

Training protocol. Only `train_fine` (2975 images); random scale, horizontal flip and crop (crop size fixed across all runs); SGD momentum 0.9, weight decay 5×10^{-4}, poly LR decay; output stride 16 for both training and evaluation. Loss is per-pixel cross-entropy with `ignore_label`$=255$. We explicitly *do not* use the `train_extra` coarse annotations in the main comparison, since their presence changes the baseline by several mIoU points and would confound attribution.

Evaluation protocol. mIoU over the 19 evaluation classes, accumulated in a single global confusion matrix over the `val` split (500 images) and averaged per class. We fix the output stride, input resolution, and the test-augmentation setting: the main table reports *single-scale, no flip*; multi-scale + flip is reported only as a separate row, because it is itself worth more than 1 mIoU.

B.5 Remote Sensing Segmentation: SegFormer-MiT-B5 on LoveDA

Experts and insertion. The Mix Transformer encoder produces four stages at $^1/_4, ^1/_8, ^1/_{16}, ^1/_{32}$ with $64, 128, 320, 512$ channels (MiT-B5 depths $[3, 6, 40, 3]$). SegFormer's All-MLP decoder projects each stage to $C = 768$, upsamples all to $^H/_4 \times ^W/_4$, concatenates them, and applies $\mathrm{Linear}(4C, C)$ then $\mathrm{Linear}(C, N_{\mathrm{cls}})$. MoENet replaces only the static concatenation: each stage is projected to C, upsampled to $^1/_4$, and combined as $\mathbf{F} = \sum_i \alpha_i P_i$; because the result already has C channels, the original $\mathrm{Linear}(4C, C)$ fusion layer is dropped and $\mathbf{F}$ goes directly to $\mathrm{Linear}(C, N_{\mathrm{cls}})$. The router reads GAP of the first overlapping patch-embedding output (64-D), i.e. the stem before stage 1.

Training protocol. AdamW, base lr 6×10^{-5} with poly decay (factor 1.0), weight decay 0.01, warmup 1500 iterations; random scale 0.5–2.0, horizontal flip, random crop; ImageNet-1K pre-trained encoder, randomly initialized decoder; per-pixel cross-entropy. LoveDA: 2522 train / 1669 val / 1796 test images, 1024×1024, 7 classes. On 4×RTX 4090 we use gradient accumulation to reach the effective batch size when a 1024 crop would otherwise exceed per-GPU memory.

Evaluation protocol. mIoU over the 7 classes via a global confusion matrix on the `val` split (1669 images); the `test` split has no public labels and is used only for leaderboard submission. Main numbers are single-scale; we additionally report per-class IoU because LoveDA's class imbalance makes mIoU sensitive to rare classes (barren, forest).

B.6 3D Scene Understanding: SpUNet34C on ScanNet

Experts and insertion. SpUNet34C is a sparse-convolution U-Net (MinkowskiEngine / SpConv). Its encoder applies four $2\times$ downsampling stages ($32 \to 32 \to 64 \to 128 \to 256$ channels); the decoder mirrors them and a $1 \times 1 \times 1$ sparse convolution outputs per-voxel logits. We take the four encoder-stage outputs as depth experts. Because sparse tensors at different levels live on different voxel grids, E_2, E_3, E_4 are sparsely upsampled to the E_1 grid and projected to a common width D before the α-weighted sum; the result enters the original decoder and head. The router reads a global sparse pooling (mean over occupied voxels) of the *stem* features (32-D) and never sees any encoder stage output.

Training protocol. ScanNet v2, 20 classes, official split 1201 train / 312 val (100 test scans are held by the evaluation server). Voxel size 2 cm (changing it changes the baseline itself). Augmentation: random rotation about z, small x/y jitter, random flip, coordinate jitter, colour jitter. SGD momentum 0.9, weight decay 10^{-4}, initial lr 0.1, poly decay ($(1 - ^{\text{iter}}/_{\text{max_iter}})^{0.9}$); we report both the iteration count and the equivalent number of epochs. Batch is one scene per GPU (4 scenes total). The router is initialized near-uniform ($\alpha \approx ^1/_4$) so the training start matches the plain backbone.

Evaluation protocol. Voxel-level mIoU: per-class $\mathrm{IoU} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP} + \mathrm{FN})$ averaged over the 20 classes, computed on `val` (312 scenes) for model selection; final numbers are obtained from the official ScanNet test server. We also report per-class IoU, mAcc and allAcc. No sliding-window or rotation-averaging test-time augmentation unless stated.

B.7 Self-Supervised Learning: DINO on Tiny-ImageNet-200

Experts and insertion. ViT-B/16 has 12 blocks with hidden dimension 768; DINO Caron et al. (2021) normally feeds only the final block's [CLS] token into its projection head. We take the [CLS] tokens of blocks $\{3, 6, 9, 12\}$ as four depth experts (the 3-expert $\{4, 8, 12\}$ variant is an ablation), project each to D and form $\boldsymbol{z} = \sum_k \alpha_k P_k(\boldsymbol{x}_k)$, which then replaces the single-token input to the DINO head. The router reads the [CLS] token *after patch embedding but before block 1*, keeping it decoupled from all four experts. Both the student and the teacher carry the router and expert projections, and the teacher's router parameters are updated by the same EMA as the rest of the teacher, so the cross-view target stays consistent.

Training protocol. Because Tiny-ImageNet-200 images are 64×64, a 16×16 patchification would leave only $4 \times 4 = 16$ patch tokens; we therefore resize to 224×224 (bicubic) so the sequence has

$14 \times 14 + 1 = 197$ tokens, and use the same resolution at linear-evaluation time. We train from scratch (no ImageNet initialization) with AdamW, $\mathrm{lr} = 0.0005 \times {}^{\mathrm{batch}}/_{256}$, batch 256 (4×RTX 4090), 10-epoch warmup then cosine decay, weight decay cosine $0.04 \to 0.4$, 300 epochs, teacher EMA momentum $0.996 \to 1.0$, teacher temperature $0.04 \to 0.07$ (warmup over the first 30 epochs) and student temperature 0.1. Multi-crop is reduced to two global 224 crops; the full $2 \times 224 + 6 \times 96$ scheme is left as an ablation, and this reduction is reported as a resource constraint rather than a methodological choice.

Evaluation protocol. Linear probing: the full backbone (patch embedding, transformer, router and expert projections) is frozen, and a single linear layer is trained on the `train` split and evaluated as top-1 on `val`. We additionally report k-NN ($k = 20$) as a tuning-free auxiliary metric. All numbers come from the same backbone initialization, split, augmentation, and probing protocol.

B.8 Semi-Supervised Learning: SimMatchV2 on Tiny-ImageNet-200

Experts and insertion. We use the ResNet-50 encoder of SimMatchV2 Zheng et al. (2023). Four depth experts are the global-average-pooled stage outputs $(64, 256, 512, 2048)$. The router reads GAP of the stem (`conv1` → `BN` → `ReLU` → `maxpool`), never any stage output. Under MixFirst the gated sum $\boldsymbol{z} = \sum_k \alpha_k P_k(\boldsymbol{x}_k)$ is fed to the original projection head and classifier, so both the supervised cross-entropy and the four graph consistencies (node–node, node–edge, edge–edge, edge–node) act through the same fused representation. The router is zero-initialized, making the training start approximately the plain ResNet-50 baseline.

Training protocol. Tiny-ImageNet-200 has 100k training images over 200 classes (500 per class); the 10% setting uses 50 labelled images per class (10k labelled, 90k unlabelled), stratified with a fixed seed, and *the labelled index file is released with the code*; the official `val` split is never used as training data. SGD with Nesterov momentum 0.9, 5-epoch warmup to lr 0.03, cosine decay to 0, weight decay 5×10^{-4}, 300 epochs; each batch contains 64 labelled and 320 unlabelled images; loss weights follow the 10% setting ($\lambda_{nn} = \lambda_{ne} = 5$). Native 64×64 inputs are used; the strong augmentation is consequently reduced to fewer scales than the ImageNet 5-crop scheme, which we report explicitly as a protocol deviation.

Evaluation protocol. Top-1 accuracy on the 10k-image `val` split, with strong augmentation disabled and the same normalization and centre crop used for both baseline and MoENet; the EMA weights are used if an EMA is present, and this choice is fixed across all runs.

B.9 Common Fairness Controls

Across all seven settings we hold fixed: the random seed; the train/val/test split and its indexing; the input resolution or crop size; the data augmentation pipeline; the optimizer, learning-rate schedule, batch size and number of epochs; the loss form and the ignore-label handling; the initialization and any pretrained weights; and the evaluation post-processing. MoENet never introduces extra training data, extra pretraining, longer schedules, or additional losses relative to its paired baseline. For detection and segmentation we further avoid test-time augmentation in the main table. Router regularization coefficients, when used, are selected on the `val` split and fixed before the final run.

B.10 Notes on the Reported Baseline Numbers

We record several provenance facts that qualify how Table 1 should be read:

- **Edge detection.** RCF reports ODS/OIS $0.806/0.823$ (single scale) and $0.811/0.830$ (multi-scale) when trained on BSDS500 augmented with flipped PASCAL VOC Context; widely used PyTorch reproductions that train on BSDS500 alone report approximately $0.798/0.815$, and note that evaluation-pipeline differences alone can shift ODS by about 0.001. The two regimes are not interchangeable, so our gain is measured entirely within one regime and one evaluation pipeline. For reference, human consistency on BSDS500 is ODS ≈ 0.803, OIS ≈ 0.830.

- **Semantic segmentation.** The value 82.1 for DeepLabv3+ is obtained with an Xception-71 encoder, ASPP and decoder, *after* additional fine-tuning on the coarse annotations, and on the `test` split; the corresponding `val` number is 79.55. A ResNet-101 baseline therefore lies several points lower, and gains must be quoted within a fixed encoder, output stride, and train/val split.
- **Remote sensing.** The SegFormer-MiT-B5 reference value on LoveDA is implementation- and configuration-dependent (crop size, split, test augmentation, pretraining); we therefore re-run the baseline in our own codebase and report the gain relative to that re-run rather than to a differing published figure.
- **Self-supervised learning.** DINO ViT-B/16 attains 78.2 linear-probe top-1 (and 76.1 k-NN) on ImageNet-1K. The number reported here is *not* that figure: it is our own linear-probe result on Tiny-ImageNet-200 under the reduced-batch, reduced-crop, from-scratch protocol above, and is expected to be substantially lower because the dataset is smaller, the classes fewer, and the images upsampled from 64×64.
- **Semi-supervised learning.** SimMatchV2 reports 76.2 top-1 for ResNet-50 at the ImageNet 10%/300-epoch setting. Tiny-ImageNet-200 has no community-standard 10% split, so our labelled subset is defined by a fixed seed and released; the reported numbers are not comparable to the ImageNet figure.

In every case the quantity we attribute to MoENet is the difference between our own baseline re-run and the MoENet variant under *identical* code, data, schedule and evaluation, rather than the difference against a differing published number.