MoE Jetpack: From Dense Checkpoints to Adaptive Mixture of Experts for Vision Tasks

Xingkui Zhu[*] **Yiran Guan**[*] **Dingkang Liang** **Yuchao Chen**
Yuliang Liu **Xiang Bai**[†]
Huazhong University of Science and Technology
`{adlith, yiranguan, xbai}@hust.edu.cn`

Abstract

The sparsely activated mixture of experts (MoE) model presents a promising alternative to traditional densely activated (dense) models, enhancing both quality and computational efficiency. However, training MoE models from scratch demands extensive data and computational resources. Moreover, public repositories like timm mainly provide pre-trained dense checkpoints, lacking similar resources for MoE models, hindering their adoption. To bridge this gap, we introduce MoE Jetpack, an effective method for fine-tuning dense checkpoints into MoE models. MoE Jetpack incorporates two key techniques: (1) *checkpoint recycling*, which repurposes dense checkpoints as initial weights for MoE models, thereby accelerating convergence, enhancing accuracy, and alleviating the computational burden of pre-training; (2) *hyperspherical adaptive MoE (SpheroMoE) layer*, which optimizes the MoE architecture for better integration of dense checkpoints, enhancing fine-tuning performance. Our experiments on vision tasks demonstrate that MoE Jetpack significantly improves convergence speed and accuracy when fine-tuning dense checkpoints into MoE models. Our code will be publicly available at `https://github.com/Adlith/MoE-Jetpack`.

1 Introduction

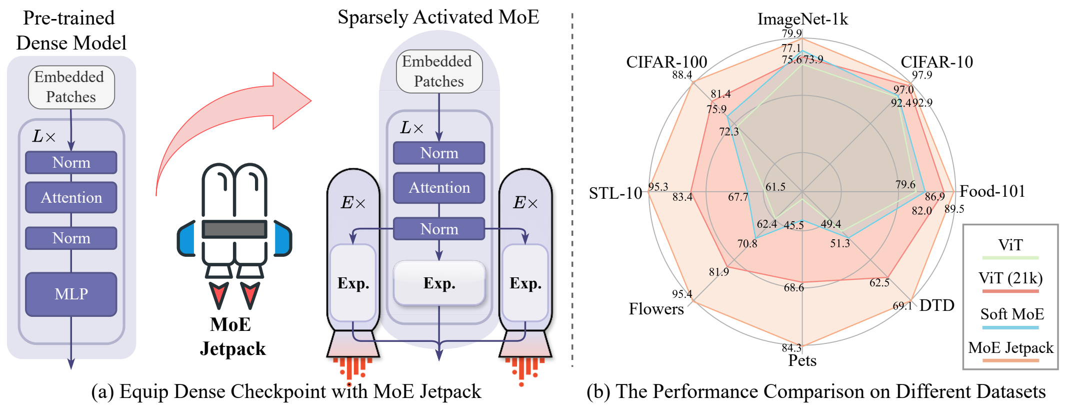
Increased scale is one of the key factors boosting performance in deep learning [1, 2, 3]. However, as models expand in size, their computational demands surge, resulting in considerable slowdowns during both training and inference phases. A promising approach that decouples model size from computational costs is the *sparsely activated mixture of experts* (MoE) [4, 5, 6]. Unlike *densely activated models* (referred to as *dense models* hereafter) [7, 8] apply the full network parameters to all inputs, MoE dynamically activates different pieces of the model for distinct input tokens. This allows for model scaling without substantially increasing the FLOPs[1], thereby maintaining training and inference speeds during model upscaling. Recent advancements have seen successful implementations of MoE across various domains [11, 12, 13].

Despite their potential, MoE models face significant adoption challenges primarily due to the lack of pre-trained models. Unlike dense models, which benefit from a rich repository of pre-trained models accessible through communities like Hugging Face [14] (∼81k models) and Timm [15] (∼800 models), most MoE models must be trained from scratch using randomly initialized weights. The absence of pre-trained weights necessitates substantial GPU hours and extensive data for training

[*]Equal contribution. † Corresponding author.

[1]FLOPs means the floating point operations per second. The vanilla design of MoE does not inherently provide runtime advantages and requires additional parallelization strategies [9, 10] for acceleration. In our implementation, we offer an effective matrix multiplication method for parallelization, detailed in Appendix C.

Preprint. Under review.



(a) Equip Dense Checkpoint with MoE Jetpack (b) The Performance Comparison on Different Datasets

Figure 1: (a) Our MoE Jetpack converts pre-trained dense models into MoE models, enhancing convergence and performance while maintaining equivalent FLOPs. Here, **Exp.** represents individual experts, E denotes the number of experts, and L indicates the total number of layers. (b) Performance comparison of ViT trained from scratch, pre-trained ViT, Soft MoE [6] trained from scratch, and MoE Jetpack across various datasets. MoE Jetpack shows significant performance improvements.

Mixture of Experts (MoE) models, thereby restricting MoE research to a limited number of research teams. Consequently, our research aims to reduce the training time and data requirements for MoE models by leveraging the pre-trained knowledge from dense checkpoints. *We will specifically investigate whether utilizing dense checkpoints can enhance the accuracy and convergence speed of MoE models during fine-tuning.*

In this paper, we propose MoE Jetpack, a new approach for fine-tuning pre-trained dense checkpoints into MoE models. As illustrated in Fig. 1(a), MoE Jetpack leverages the sunk cost of dense pre-training to enhance MoE model performance and accelerate convergence. It comprises two key techniques. The first is **checkpoint recycling**, which initializes MoE models using dense checkpoints. Unlike sparse upcycling [16], which merely copies the Multilayer Perceptron (MLP) to construct experts, checkpoint recycling leverages various dense checkpoints and multiple weight selection methods. This approach provides greater flexibility and results in superior MoE initialization weights. The second technique is the **hyperspherical adaptive MoE (SpheroMoE) layer**, which presents an optimized MoE architecture for seamless integration of dense checkpoints and enhanced fine-tuning performance. Existing MoE architectures, such as Switch Transformers [4] and Soft MoE [6], are not designed to leverage pre-existing dense checkpoints, which may lead to optimization and over-specialization challenges during fine-tuning. The SpheroMoE layer mitigates these challenges by normalized token mixing, expert regularization, and adaptive dual-path.

By equipping dense checkpoints with MoE Jetpack, the fine-tuned MoE models achieve significantly higher accuracy than those trained from scratch while maintaining the same FLOPs as the original dense models. Comprehensive evaluations of MoE Jetpack across various image classification datasets of differing scales demonstrate its effectiveness. As shown in Fig. 1(b), MoE Jetpack leverages existing dense checkpoints to enhance fine-tuning convergence speed and performance, maintaining efficiency comparable to the original dense models. In summary, our contributions are as follows:

- We introduce *checkpoint recycling*, which pioneers the selection of dense checkpoints to initialize MoE experts, enhancing initialization flexibility, diversifying experts, and eliminating the computational burden of MoE pre-training.
- We develop the *spheroMoE layer*, optimized for fine-tuning dense checkpoints into MoE architectures, alleviating optimization challenges, and preventing the over-specialization of experts.

2 Background

In this section, we recap the main components used in MoE Jetpack: the sparsely activated mixture of expert (MoE) architectures and different routing mechanisms in MoE. Existing MoE models are typically derived from dense Vision Transformer (ViT) architectures by replacing certain multilayer perceptron (MLP) layers with MoE layers. Each MoE layer comprises a Router$(x; \theta_{gate})$ and several "experts", each parameterized as MLP$(\cdot; \theta_i)$. MoE models typically use similar experts, with differences primarily in their routing mechanisms. Several routing algorithms have been developed, including Top-K [17], BASE and Sinkhorn-BASE layers [18, 19], Hash layers [20], Expert Choice routing [21], and soft routing [6]. The most commonly employed routing mechanism is the Top-K, which effectively reduces computational overhead by sparsely activating only the top-K experts relevant to the input token. The routing decision is formulated as:

$$\text{Router}(x; \theta_{gate}) = \text{Top-K}\left(\text{softmax}(\text{MLP}(x; \theta_{gate}))\right), \tag{1}$$

$$y = x + \sum_{i \in E} \text{Router}(x; \theta_{gate}) \cdot \text{Expert}(x; \theta_i), \tag{2}$$

where θ denotes the weights, E is the set of activated experts, and $|E| = K$. However, Top-K routing faces challenges such as imbalanced expert utilization, token dropping, and scalability issues.

Enhanced mechanisms like Soft MoE [6] address these issues and serve as our baseline. The Soft MoE routing algorithm processes input tokens $\mathbf{X} \in \mathbb{R}^{m \times d}$, where m represents the number of tokens, and d is their dimensionality. It uses learnable parameters $\Phi \in \mathbb{R}^{d \times (e \cdot s)}$ to reconfigure these tokens into $e \times s$ slots. The transformed input slots $\tilde{\mathbf{X}} \in \mathbb{R}^{(e \cdot s) \times d}$ are combinations of the input tokens: $\tilde{\mathbf{X}} = \text{softmax}(\mathbf{X}\Phi)^\top \mathbf{X}$. Each MoE layer includes e expert functions $\{f_i : \mathbb{R}^d \to \mathbb{R}^d\}_{i=1}^{e}$, with each expert handling s slots. The intermediate output $\tilde{\mathbf{Y}} \in \mathbb{R}^{(e \cdot s) \times d}$ is obtained by applying the expert functions to the transformed slots: $\tilde{\mathbf{Y}}_{i,j} = f_i(\tilde{\mathbf{X}}_{i,j})$ for $i \in \{1, \ldots, e\}$ and $j \in \{1, \ldots, s\}$. The output tokens $\mathbf{Y} \in \mathbb{R}^{m \times d}$ are generated by reassembling the outputs of the experts: $\mathbf{Y} = \text{softmax}(\mathbf{X}\Phi)\tilde{\mathbf{Y}}$.

3 MoE Jetpack

In this section, we present the overarching concept of the MoE Jetpack. It is divided into two phases: initializing MoE models with checkpoint recycling and fine-tuning MoE models using the hyperspherical adaptive MoE (SpheroMoE) layer.

3.1 Checkpoint Recycling

Checkpoint recycling is a foundational phase in the MoE Jetpack framework, transforming pre-trained dense model checkpoints (**predecessors**) into high-quality initialization weights for MoE models (**successors**). This approach ensures efficient utilization of resources invested in predecessors, boosting the performance and convergence speed of successors. The recycling procedure involves splitting the predecessors' multilayer perceptrons (MLPs) into multiple experts, ensuring expert diversity and adaptability in expert size to meet varied needs.

To define the process of checkpoint recycling (as illustrated in Fig. 2(a)), consider predecessors with N layers L_i, a channel dimension of d, and a hidden dimension (neuron) of $4d$. We aim to transform these into a successor MoE model S with N layers L'_i, a reduced channel dimension d', where $d' \le d$. Following the Soft MoE [6], the successor comprises two segments: a dense part with N_1 layers and an MoE part with N_2 layers, where $N_1 = N_2 = \frac{N}{2}$. Formally, the successor model is represented as:

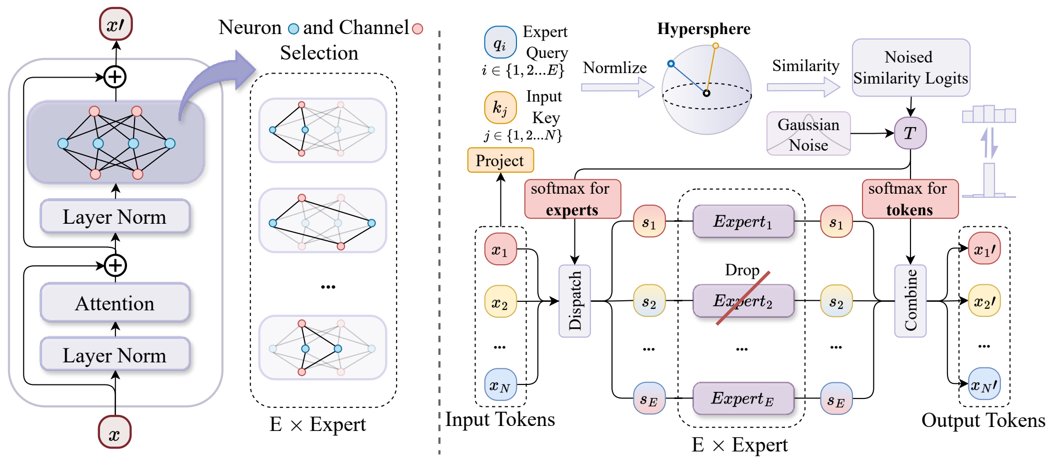
$$S = \left(\{L'_i\}_{i=0}^{\frac{N}{2}-1}, \{L'_i\}_{i=\frac{N}{2}}^{N-1} \right). \tag{3}$$

Inspired by Weight Selection [22], our recycling process maintains channel consistency. We explore four primary strategies to guide the recycling of checkpoints:

Importance-Based Weight Sampling (default): Weights are sampled based on their importance, determined by activation values. We pass batches of images through the predecessor model and obtain the activation values for each channel and neuron in the MLP layers. For channel selection, we average the activation values across the N layers and select the top-d' channels:

$$A_c = \frac{1}{N} \sum_{i=0}^{N-1} A_{L_i}^{c}, \quad \text{Top-}d' = \text{argmax}_c(A_c), \quad |c| = d', \tag{4}$$

where $A_{L_i}^{c}$ is the activation value of channel c in layer L_i, A_c is the averaged activation value of channel c across N layers, and Top-d' represents the indices of the top-d' channels with the highest average activation values.



(a) Checkpoints Recycling (b) SpheroMoE Layer

Figure 2: (a) Checkpoint Recycling selects neurons and channels from the MLP of pre-trained dense checkpoints using weight sampling methods. This process transforms pre-trained knowledge into multiple experts of any size for initializing MoE models. (b) The SpheroMoE layer uses cross-attention to adaptively dispatch input tokens to expert slots. It starts with a randomly initialized query and uses keys and values derived and normalized from the input. The similarity logits between the query and key are calculated in a hyperspherical space, stabilizing the random query. The outputs from the experts are then combined back into the input using the generated similarity logits.

For hidden dimensions, we convert the activation values into a probability distribution and sample different experts based on this distribution:

$$P(h|H) = \frac{A_h}{\sum_{h' \in H} A_{h'}}, \quad h_{\text{successor}} \sim P(h|H), \quad |h_{\text{successor}}| = 4d', \tag{5}$$

where A_h is the activation value of hidden neuron h, and H is the set of all hidden neurons. This method ensures that the most important weights are selected for the successor model.

Co-Activation Graph Partitioning: This strategy groups frequently co-activated neurons into one expert. We construct a co-activation graph by counting the co-activations of neurons in the predecessor for training samples. Each neuron is a vertex in the graph, and edges represent their co-activation frequency. Formally, let $G = (V, E)$ be the co-activation graph, where V represents the neurons and E represents edges with weights indicating co-activation counts. Using the Metis graph partitioning [23], we get several subgraphs:

$$G = \bigcup_{i=1}^{k} G_i, \quad G_i = (V_i, E_i), \quad V_i \cap V_j = \emptyset \text{ for } i \neq j. \tag{6}$$

Experts are formed by the combination of sub-graphs. This method leverages the natural grouping of neurons, ensuring each expert captures a specific functional subset of the predecessor model.

Uniform Weight Selection: Weights are selected uniformly across channels. For a predecessor with channel dimension d and a successor with dimension d', weights are chosen as:

$$W^{(i)}_{\text{successor}} = W^{(k)}_{\text{predecessor}}, \quad k = \left\lfloor \frac{i \cdot d}{d'} \right\rfloor, \quad i \in \{0, \ldots, d' - 1\}. \tag{7}$$

This method ensures an even distribution of the pre-trained weights across the successor MoE.

Random Weight Sampling: Weights are randomly selected from the predecessor model. Let S be a random subset of channel indices:

$$S \subseteq 0, \ldots, d - 1, \quad |S| = d'. \tag{8}$$

Then, the weights for the successor are chosen as:

$$W^{(i)}_{\text{successor}} = W^{(j)}_{\text{predecessor}}, \quad j \in S, \quad i \in \{0, \ldots, d' - 1\}. \tag{9}$$

Through the ablation in Sec. 4.3, Importance-Based Weight Sampling is identified as the default method for recycling dense checkpoints to initialize MoE models.

3.2 SpheroMoE Layer

Following the initialization of MoE weights through Checkpoint Recycling, the next step is fine-tuning on downstream datasets. To enhance performance and stability, we designed the hyperspherical adaptive MoE (SpheroMoE) layer (Fig. 2(b)), introducing three key improvements: SpheroMoE Routing to alleviate optimization challenges, Expert Regularization to prevent over-specialization, and Adaptive Dual-path MoE (Fig. 3 for better performance and efficiency. Additionally, the pseudo-code detailing these features' implementation can be found in Appendix C.

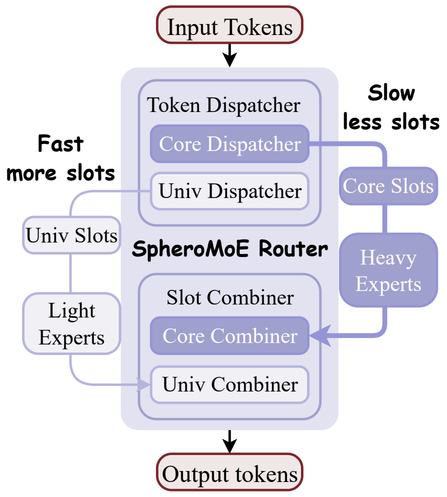


Figure 3: The Adaptive Dual-path MoE structure enhances the SpheroMoE Router by adapting it into a dual-branch system, designed to optimize computational efficiency and model performance. This configuration directs high-impact tokens to a core path with fewer but larger experts, while routing less critical tokens to a universal path equipped with a greater number of smaller experts.

SpheroMoE Routing: As shown in Fig. 2(b), the proposed hyperspherical MoE (SpheroMoE) routing mechanism utilizes cross-attention [24] to distribute inputs across experts. Each expert receives an input slot, a weighted average of all input tokens. To maintain consistency between dense checkpoints and MoE layers $M = \{L_i'\}_{i=N/2}^{N-1}$, input tokens $\mathbf{X} \in \mathbb{R}^{b\times n\times d}$ (where b represents the batch size, n represents the token length, and n represents the input dimension) are layer normalized inherited from dense checkpoints, resulting in $\mathbf{X}_{\text{norm}}$. Queries $\mathbf{Q} \in \mathbb{R}^{b\times(e\times s)\times d}$ are randomly initialized and similarly normalized to align with $\mathbf{X}_{\text{norm}}$, producing $\mathbf{Q}_{\text{norm}}$. The layer normalization process ensures the consistency of distributions between the MoE model, input queries, and the pre-trained dense model. The normalized $\mathbf{X}_{\text{norm}}$ are projected to form keys $\mathbf{K} \in \mathbb{R}^{b\times n\times d}$ for the cross-attention mechanism. To address numerical instability with randomly initialized queries, $\mathbf{Q}_{\text{norm}}$ are projected onto a hyperspherical space using L2 normalization. The similarity between $\mathbf{Q}_{\text{norm}}$ and $\mathbf{K}$ is computed in this space, yielding similarity logits $\mathbf{S} \in \mathbb{R}^{b\times(e\times s)\times n}$: $\mathbf{S} = \mathbf{Q}_{\text{norm}}\mathbf{K}^T$. We do not apply L2 normalization to $\mathbf{K}$ to preserve their scale information, enhancing the matching process. Input slots $\tilde{\mathbf{X}} \in \mathbb{R}^{b\times(e\times s)\times d}$ for experts are formed by a softmax operation along the n dimension of similarity logits:

$$\tilde{\mathbf{X}} = \frac{\exp(\mathbf{S}_{ijk})}{\sum_{k'=1}^{n}\exp(\mathbf{S}_{ijk'})}\mathbf{X}_{\text{norm}}. \tag{10}$$

Each expert processes its corresponding input slots $\tilde{\mathbf{X}}_i$ independently, generating outputs $\tilde{\mathbf{Y}}_i$. These outputs are then weighted by $\mathbf{S}$ (after applying a softmax operation along the $(e\times s)$ dimension) to aggregate the experts' contributions, producing the final output $\mathbf{Y} \in \mathbb{R}^{b\times n\times d}$ of the MoE layer:

$$\mathbf{Y} = \frac{\exp(\mathbf{S}_{ijk})}{\sum_{j'=1}^{e\times s}\exp(\mathbf{S}_{ij'k})}\tilde{\mathbf{Y}}. \tag{11}$$

In summary, SpheroMoE routing leverages layer normalization, hyperspherical projection, and cross-attention to effectively distribute inputs across experts, ensuring consistency with pre-trained dense models for improved optimization.

Expert Regularization: To prevent over-specialization and enhance generalization during fine-tuning, we regulate SpheroMoE Routing and expert behavior. We aim to prevent experts from overly focusing on specific inputs and to avoid outputs becoming overly dependent on particular experts. For the former, we introduced a learnable softmax temperature T. During the early stages of fine-tuning, T is initialized to a large value, causing experts to distribute their attention across all input tokens and preventing early convergence on specific tokens. As training progresses, T gradually decreases, enabling experts to focus more on specific relevant features and enhancing their specialization where beneficial. Additionally, we added a certain level of expert noise to the similarity logits $\mathbf{S}$, which improves generalization. For the latter, we utilized stochastic expert dropout, where each expert i is randomly deactivated with a probability p. It ensures that no single expert becomes a crutch for the entire output, promoting a more balanced utilization of all experts. These techniques form an expert regularization strategy that maintains expert versatility and mitigates overfitting, ensuring the MoE model performs robustly on downstream datasets.

Adaptive Dual-path MoE: To mitigate computational redundancy for less critical tokens and enhance MoE model performance, we introduce the Adaptive Dual-path MoE structure. Building on the checkpoint recycling, which enables MoE models to inherit pre-trained knowledge from dense checkpoints and discern crucial from non-crucial tokens, our structure employs two pathways. The first pathway features fewer core experts with more parameters for processing important tokens. Conversely, the second pathway consists of a larger number of universal experts, each with approximately one-fourth of the parameters of the core experts, designated for handling less critical tokens. The structure is illustrated in Fig. 3. The SpheroMoE Routing mechanism segments input tokens into core and universal slots, assigning them to the respective pathways. The core path processes high-impact tokens, while the universal path handles less important ones, ensuring optimal resource utilization and preserving model accuracy while accelerating the MoE model.

4 Experiments

4.1 Experimental Setups

Models. We conduct experiments using Vision Transformer (ViT) [7] and ConvNeXt [8] to validate our approach. Specifically, we transform the ImageNet 21K pre-trained dense checkpoints of ViT-S and ConvNeXt-T into the initialization weights of V-JetMoE-T and C-JetMoE-F through checkpoint recycling. As detailed in Sec. 3.1, V-JetMoE-T comprises dense layers in the first half and is equipped with SpheroMoE layers in the latter half. Each SpheroMoE layer consists of $N/2$ core experts and N universal experts, where N is the number of input tokens. Further details are in Appendix A.

Datasets. We evaluate MoE Jetpack on 8 image classification datasets, including ImageNet-1K [25], CIFAR-10, CIFAR-100 [26], Flowers [27], Pets [28], STL-10 [29], Food-101 [30], and DTD [31], encompassing a diverse range of tasks, including object classification, fine-grained species recognition, and texture classification.

Baseline Implementation. We follow the implementation details outlined by Xu et al. [22] for comparisons of the dense models. For the MoE models, we employ Soft MoE [6] as the baseline and have replicated it across all datasets. Our MoE Jetpack and Soft MoE utilize the same training strategies as the dense models to ensure comparison fairness. All implementations were executed using the MMpretrain framework [32] on RTX4090. More information can be found in Appendix B.

4.2 Main Results

Tab. 1 compares the performance of the MoE Jetpack with Dense ViT models (trained from scratch and with pre-trained weights on ImageNet-21k) and Soft MoE models (trained from scratch) on various image classification datasets using ViT-T (a) and ConvNeXt-F (b) architectures. All models maintain approximately the same number of FLOPs. The MoE Jetpack, which inherits the knowledge from dense checkpoints pre-trained on ImageNet-21k, consistently outperforms MoE models trained from scratch, especially on smaller datasets. These results highlight the effectiveness of MoE Jetpack.

Table 1: Performance comparison on visual recognition tasks with ViT-T and ConvNeXt-F.

Dataset (↓)	Dense	Dense (21k)	Soft MoE [6]	MoE Jetpack
ImgNet-1k	73.9	75.6	77.1	79.9 (+2.8)
Food-101	79.6	86.9	82.0	89.5 (+7.5)
CIFAR-10	92.4	97.0	92.9	97.9 (+5.0)
CIFAR-100	72.3	81.4	75.9	88.4 (+12.5)
STL-10	61.5	83.4	67.7	95.3 (+27.6)
Flowers	62.4	81.9	70.8	95.4 (+24.6)
Pets	25.0	68.6	45.5	84.3 (+38.8)
DTD	49.4	62.5	51.3	69.1 (+17.8)

(a) ViT-T

Dataset (↓)	Dense	Dense (21k)	Soft MoE [6]	MoE Jetpack
ImgNet-1k	76.1	76.4	79.1	80.5 (+1.4)
Food-101	86.9	89.0	88.7	90.7 (+2.0)
CIFAR-10	96.6	97.4	97.3	98.2 (+0.9)
CIFAR-100	81.4	84.4	82.8	88.5 (+5.7)
STL-10	81.4	92.3	79.4	98.7 (+19.3)
Flowers	80.3	94.5	83.3	98.6 (+15.3)
Pets	72.9	87.3	77.4	94.9 (+17.5)
DTD	63.7	68.8	64.7	79.5 (+14.8)

(b) ConvNeXt-F

4.3 Ablations

We perform ablation studies to assess the impact of various components and hyperparameters within the MoE Jetpack. By default, we use a ViT-T model with the SpheroMoE layer integrated from layers 7 to 12, comprising 98 core experts and 196 universal experts (detailed in Appendix A). The Checkpoint Recycling method transforms dense checkpoints of ViT-S and ViT-T, pre-trained on ImageNet-21k, into initial weights for our V-JetMoE-T model.

Effect of MoE Jetpack Components. We conducted the ablation of two key components of the MoE Jetpack on three datasets. As shown in Tab. 2, integrating Checkpoint Recycling with the Soft MoE baseline significantly improves performance across all datasets, with a mean accuracy increment of 9.8%. The SpheroMoE layer further enhances performance, achieving a mean accuracy of 87.9%. These results demonstrate the efficacy of both components, especially when used together, highlighting their synergistic effect in boosting performance.

Checkpoint Recycling vs. Sparse Upcycling. To compare the four checkpoint recycling strategies mentioned in Sec. 3.1 and the method of using duplicated MLPs to construct experts in Sparse Upcycling [16], we conducted experiments on ImageNet. For fairness, we also employed our SpheroMoE layer in the Sparse Upcycling. The results, summarized in Tab. 3, show that Importance-Based Sampling achieves the highest performance, demonstrating its effectiveness in leveraging critical weights to enhance model performance and convergence speed. Additionally, Checkpoint Recycling is highly flexible, allowing the construction of experts of varying sizes to meet different needs, a feature not provided by sparse upcycling.

Table 3: Checkpoint Recycling vs. Sparse Upcycling

Method	Construction	ImageNet
Sparse Upcycling [16]	Copy	79.1
Checkpoint Recycling	Random Sampling	79.5
	Uniform Selection	79.6
	Graph Partitioning	79.8
	Importance-based Sampling	**79.9**

Core Experts Ratio. To assess the impact of the Adaptive Dual-path MoE structure introduced in Sec. 3.2 on the accuracy of MoE models, we aimed to determine the ideal balance between performance and resource allocation. We conducted experiments on the Cifar-100 dataset with a constant number of total experts, varying the ratio of core experts. The results, illustrated in Fig. 4, indicate that optimal accuracy is achieved when the proportion of core experts is set at $1/3$.

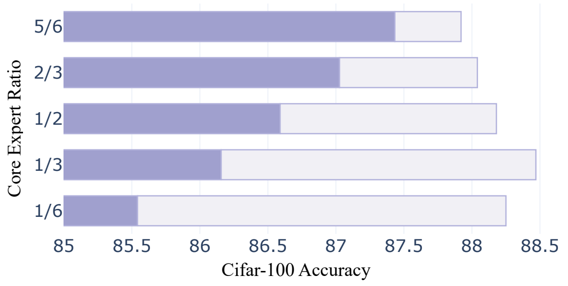


Figure 4: This chart shows CIFAR-100 accuracy across different ratios of core (dark) to universal (light) experts, highlighting optimal performance at a 1/3 core ratio.

Different MoE Jetpack Configurations. This part evaluates the impact of various MoE Jetpack configurations on model performance, as summarized in Tab. 4. The experiments focus on the placement of SpheroMoE layers, the number of experts per layer, and the base size of converted dense checkpoints. Results indicate that more SpheroMoE layers generally enhance performance, though placing it before layer 7 slightly hurt the performance. Consequently, SpheroMoE layers were incorporated into layers $7-12$. Additionally, models with more experts exhibit improved accuracy, highlighting the benefits of increased expert specialization and diversity. Models converted from larger dense checkpoints demonstrate superior performance. These findings suggest that MoE network performance can be improved by increasing the number of MoE layers, incorporating more experts, and utilizing larger base models.

Table 2: Ablation Study on MoE Jetpack Components.

Soft MoE [6]	Checkpoints Recycling	SpheroMoE	ImageNet	CIFAR-100	Flowers	Mean Acc.
Baseline ViT-T			73.9	72.3	62.4	69.5
✓			77.1	75.9	70.8	74.6 (+5.1)
✓	✓		78.4	84.7	91.2	84.8 (+15.3)
	✓	✓	79.9	88.4	95.4	**87.9** (+18.4)

Table 4: Comparison of Model Variants with Different Configurations

model	Weight Init.	MoE Layers	Expert Number	Param (M)	FLOPs (G)	CIFAR-100	ImageNet
ViT-T	-	-	-	6	1.1	72.3	73.9
Soft MoE-T [6]	-	7:12	197	354	1.2	75.9	77.1
Soft MoE-S [6]	-	7:12	197	1412	4.5	77.5	80.3
ViT-T	✓	-	-	6	1.1	81.4	75.5
V-JetMoE-T	✓	11:12	core: 98, univ: 196	92	1.1	87.4	-
V-JetMoE-T	✓	9:12	core: 98, univ: 196	179	1.1	87.8	-
V-JetMoE-T	✓	5:12	core: 98, univ: 196	352	1.2	86.7	-
V-JetMoE-T	✓	7:12	core: 32, univ: 64	89	0.8	87.8	-
V-JetMoE-T	✓	7:12	core: 64, univ: 128	175	1.0	88.0	-
V-JetMoE-T	✓	7:12	core: 98, univ: 196	265	1.1	88.4	79.9
V-JetMoE-S	✓	7:12	core: 98, univ: 196	1058	4.3	**89.9**	**82.4**

4.4 Analysis

In this section, we investigate the influence of the MoE Jetpack on enhancing the convergence speed of MoE models when trained on the ImageNet and CIFAR-100 datasets. Additionally, we provide some intuition regarding the attention patterns of the experts and the contribution of each expert to the final results.

Accelerating MoE Convergence with MoE Jetpack. The impact of MoE Jetpack on convergence speed is evident in Fig. 5 for ImageNet (left) and CIFAR-100 (right). In both cases, models with MoE Jetpack reach the target accuracy significantly faster. For ImageNet, the model with MoE Jetpack achieves approximately 77% top.1 accuracy before 150 epochs, **2 times faster** than training from scratch. Notably, for smaller datasets like CIFAR-100, the acceleration effect of MoE Jetpack is more pronounced: The model with MoE Jetpack reaches the 76% top.1 accuracy at around 40 epochs, **8 times faster** than the model without it. These results demonstrate that MoE Jetpack substantially accelerates convergence speed, enhancing fine-tuning efficiency and reducing computational resources.

Intuition of Expert Attention Patterns. We visualize the attention maps of experts in Fig. 6(a), which illustrates that different experts focus on different parts of the input image. This diversity in attention suggests that each expert specializes in capturing unique aspects of the input, enhancing the model's ability to represent features comprehensively. The specialization allows the MoE model to combine multiple perspectives, resulting in a more robust and detailed understanding of the input.

Contribution of Each Expert to Final Results. Fig. 6(b) demonstrates the varying contributions of core and universal experts across different layers of the MoE model. Core experts show an increasing influence in the later layers, emphasizing their role in refining specific and highly relevant features. Additionally, the contributions among core experts are markedly uneven, some experts can impact output tokens $17\times$ more than others, reflecting greater specialization and diversity in their focus areas. In contrast, universal experts maintain a relatively consistent contribution level, indicating a more uniform integration of broader contextual information throughout the network. This hierarchical structure, balancing the specialized refinement by core experts with the generalized understanding provided by universal experts, enhances the model's overall performance and robustness.

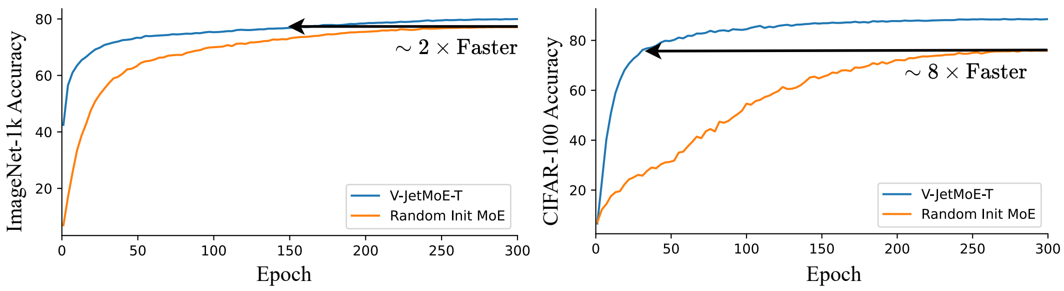
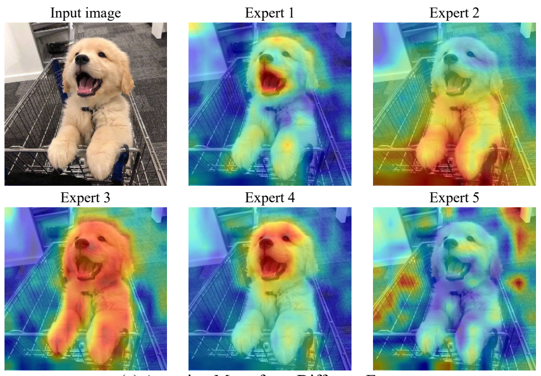
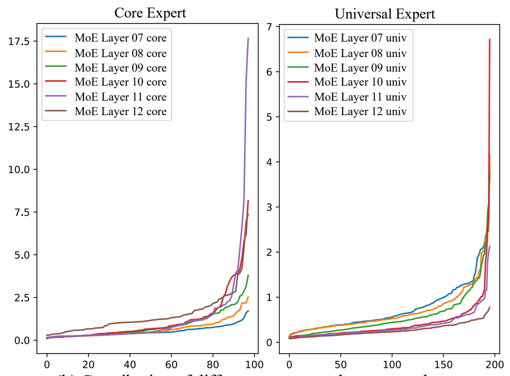


Figure 5: Comparison of convergence speeds using MoE Jetpack versus training from scratch on ImageNet (left) and CIFAR-100 (right). MoE Jetpack achieves target accuracies significantly faster, demonstrating a 2x speed increase on ImageNet and an 8x increase on CIFAR-100.



(a) Attention Maps from Different Experts



(b) Contribution of different experts to the output tokens

Figure 6: (a) This figure illustrates the attention maps generated by five experts in response to an input image, highlighting the experts' specialization. (b) These line charts show varying contributions of core and universal experts, with core experts' influence peaking in later layers, emphasizing their detailed feature refinement, contrasted with the consistent input of universal experts.

5 Related Work

Sparsely activated Mixture of Experts (MoE). Scaling Laws [33] indicate that increasing model parameters can enhance performance. However, traditional densely activated models (dense models) [7, 8] activate all parameters for every input, resulting in high computational costs as models scale. In contrast, MoE models [12, 34, 35, 36] activate only a subset of parameters for specific input tokens, enabling efficient scaling to trillions of parameters with sublinear increases in computational costs [37, 5, 4]. To optimize input token allocation among experts, various routing mechanisms have been developed. BASELayer [18] formulates token-to-expert allocation as a linear assignment problem, while EC-CF2 [21] propose expert choice routing, soft routing methods like SMEAR [38], and Soft MoE [6] implicit soft assignments involving all tokens. However, few studies explore leveraging dense model checkpoints to accelerate MoE training [16].

Knowledge transfer with pre-trained models. Knowledge transfer occurs between **identical** or **distinct** models. Pre-training followed by fine-tuning is well-established for **identical models**, utilizing large datasets through supervised learning (e.g., ImageNet21k [25], JFT-300M [39]) or self-supervised methods (e.g., BERT [40], CLIP [41], MAE [42], DINO [43], EVA [44, 45]). These approaches produce foundation models with broad applicability, and subsequent fine-tuning consistently improves performance. For **distinct models**, knowledge distillation [46] trains a smaller student model to mimic the larger teacher model, enhancing efficiency. Additional strategies include weight pruning [47, 48, 48, 49, 50], which removes redundant parameters, and weight selection [22] initializes a smaller model with a subset of weights from a pre-trained larger model.

Research on transferring knowledge from dense checkpoints to MoE models is limited. MoEfication [51] partitions a dense model into MoE components, while Sparse Upcycling [16] replicates a dense model multiple times to form a MoE model. Our MoE Jetpack recycles important weights from larger dense checkpoints to initialize experts of various sizes, combining the flexibility of knowledge transfer across distinct models with the efficiency of transfer between identical models.

6 Conclusion

In this paper, we introduced MoE Jetpack, a novel framework for fine-tuning pre-trained dense checkpoints into Mixture of Experts. Our approach leverages checkpoint recycling, which inherits the knowledge of open-source dense checkpoints and the hyperspherical adaptive MoE (SpheroMoE) layer to enhance fine-tuning performance. These innovations contribute to improved convergence speed and model accuracy. The MoE Jetpack significantly improved various visual tasks while maintaining computational efficiency.

The **limitation** of our approach is its dependency on the quality of pre-trained dense checkpoints; poorly trained or inadequately generalized dense models could limit the performance enhancements. Additionally, while our experiments focused on visual tasks, further research is needed to validate the generalizability of MoE Jetpack across other domains, such as natural language processing and reinforcement learning. We believe future work will address these limitations, enhance the scalability and robustness of the framework, and extend MoE applicability to a broader range of tasks.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. of Advances in Neural Information Processing Systems*, vol. 25, 2012. 1

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 1

[3] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 12 104–12 113. 1

[4] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022. 1, 2, 9

[5] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby, "Scaling vision with sparse mixture of experts," *Proc. of Advances in Neural Information Processing Systems*, vol. 34, pp. 8583–8595, 2021. 1, 9

[6] J. Puigcerver, C. R. Ruiz, B. Mustafa, and N. Houlsby, "From sparse to soft mixtures of experts," in *Proc. of Intl. Conf. on Learning Representations*, 2023. 1, 2, 3, 6, 7, 8, 9

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. of Intl. Conf. on Learning Representations*, 2020. 1, 6, 9

[8] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986. 1, 6, 9

[9] Z. Fan, R. Sarkar, Z. Jiang, T. Chen, K. Zou, Y. Cheng, C. Hao, Z. Wang *et al.*, "M^{3}vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design," *Proc. of Advances in Neural Information Processing Systems*, vol. 35, pp. 28 441–28 457, 2022. 1

[10] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, "Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale," in *Proc. of Intl. Conf. on Machine Learning*. PMLR, 2022, pp. 18 332–18 346. 1

[11] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024. 1

[12] D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu *et al.*, "Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models," *arXiv preprint arXiv:2401.06066*, 2024. 1, 9

[13] B. Lin, Z. Tang, Y. Ye, J. Cui, B. Zhu, P. Jin, J. Zhang, M. Ning, and L. Yuan, "Moe-llava: Mixture of experts for large vision-language models," *arXiv preprint arXiv:2401.15947*, 2024. 1

[14] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019. 1

[15] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019. 1

[16] A. Komatsuzaki, J. Puigcerver, J. Lee-Thorp, C. R. Ruiz, B. Mustafa, J. Ainslie, Y. Tay, M. Dehghani, and N. Houlsby, "Sparse upcycling: Training mixture-of-experts from dense checkpoints," in *Proc. of Intl. Conf. on Learning Representations*, 2022. 2, 7, 9

[17] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proc. of Intl. Conf. on Learning Representations*, 2016. 3

[18] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer, “Base layers: Simplifying training of large, sparse models,” in *Proc. of Intl. Conf. on Machine Learning*. PMLR, 2021, pp. 6265–6274. 3, 9

[19] A. Clark, D. de Las Casas, A. Guy, A. Mensch, M. Paganini, J. Hoffmann, B. Damoc, B. Hechtman, T. Cai, S. Borgeaud *et al.*, “Unified scaling laws for routed language models,” in *Proc. of Intl. Conf. on Machine Learning*. PMLR, 2022, pp. 4057–4086. 3

[20] S. Roller, S. Sukhbaatar, J. Weston *et al.*, “Hash layers for large sparse models,” *Proc. of Advances in Neural Information Processing Systems*, vol. 34, pp. 17 555–17 566, 2021. 3

[21] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, J. Laudon *et al.*, “Mixture-of-experts with expert choice routing,” *Proc. of Advances in Neural Information Processing Systems*, vol. 35, pp. 7103–7114, 2022. 3, 9

[22] Z. Xu, Y. Chen, K. Vishniakov, Y. Yin, Z. Shen, T. Darrell, L. Liu, and Z. Liu, “Initializing models with larger ones,” in *Proc. of Intl. Conf. on Learning Representations*, 2024. 3, 6, 9

[23] G. Karypis, “Metis: Unstructured graph partitioning and sparse matrix ordering system,” *Technical report*, 1997. 4

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Proc. of Advances in Neural Information Processing Systems*, vol. 30, 2017. 5

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255. 6, 9

[26] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009. 6

[27] M.-E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *Indian conference on computer vision, graphics & image processing*. IEEE, 2008, pp. 722–729. 6

[28] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, “Cats and dogs,” in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3498–3505. 6

[29] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223. 6

[30] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101–mining discriminative components with random forests,” in *Proc. of European Conference on Computer Vision*. Springer, 2014, pp. 446–461. 6

[31] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 3606–3613. 6

[32] M. Contributors, “Openmmlab’s pre-training toolbox and benchmark,” https://github.com/open-mmlab/mmpretrain, 2023. 6

[33] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020. 9

[34] B. Lin, Z. Tang, Y. Ye, J. Cui, B. Zhu, P. Jin, J. Zhang, M. Ning, and L. Yuan, “Moe-llava: Mixture of experts for large vision-language models,” *arXiv preprint arXiv:2401.15947*, 2024. 9

[35] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, “Mixtral of experts,” *arXiv preprint arXiv:2401.04088*, 2024. 9

[36] X. Wu, S. Huang, and F. Wei, "Mole: Mixture of lora experts," in *Proc. of Intl. Conf. on Learning Representations*, 2023. 9

[37] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," in *Proc. of Intl. Conf. on Learning Representations*, 2020. 9

[38] M. Muqeeth, H. Liu, and C. Raffel, "Soft merging of experts with adaptive routing," *arXiv preprint arXiv:2306.03745*, 2023. 9

[39] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Porc. of IEEE Intl. Conf. on Computer Vision*, 2017, pp. 843–852. 9

[40] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186. 9

[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. of Intl. Conf. on Machine Learning*. PMLR, 2021, pp. 8748–8763. 9

[42] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009. 9

[43] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. HAZIZA, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *Transactions on Machine Learning Research*, 2023. 9

[44] Y. Fang, Q. Sun, X. Wang, T. Huang, X. Wang, and Y. Cao, "Eva-02: A visual representation for neon genesis," *arXiv preprint arXiv:2303.11331*, 2023. 9

[45] Q. Sun, Y. Fang, L. Wu, X. Wang, and Y. Cao, "Eva-clip: Improved training techniques for clip at scale," *arXiv preprint arXiv:2303.15389*, 2023. 9

[46] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2014. 9

[47] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Proc. of Advances in Neural Information Processing Systems*, vol. 28, 2015. 9

[48] S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman, "Slicegpt: Compress large language models by deleting rows and columns," in *Proc. of Intl. Conf. on Learning Representations*, 2023. 9

[49] M. Xia, T. Gao, Z. Zeng, and D. Chen, "Sheared llama: Accelerating language model pre-training via structured pruning," in *Proc. of Intl. Conf. on Learning Representations*, 2023. 9

[50] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width & depth pruning for vision transformers," in *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 3143–3151. 9

[51] Z. Zhang, Y. Lin, Z. Liu, P. Li, M. Sun, and J. Zhou, "Moefication: Transformer feed-forward layers are mixtures of experts," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 877–890. 9

[52] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. of Intl. Conf. on Learning Representations*, 2018. 14

[53] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition workshops*, 2020, pp. 702–703. 14

[54] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. of Intl. Conf. on Learning Representations*, 2018. 14

[55] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in *Porc. of IEEE Intl. Conf. on Computer Vision*, 2019, pp. 6023–6032. 14

[56] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” in *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008. 14

[57] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826. 14

[58] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, “Going deeper with image transformers,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 32–42. 14

A Detailed Model Configurations

In this section, we present the detailed model configurations for the main experiments in Sec. 4 in Tab. 5. We refer to pre-trained dense checkpoints as **predecessors** and the derived MoE models as **successors**. We use ImageNet-21k pre-trained predecessor from timm with our Checkpoint Recycling algorithm to generate initialized weights for the successor.

Table 5: Configurations for Models.

Configuration	Successors		Predecessors	
Model	V-JetMoE-T	C-JetMoE-F	ViT-S/16	ConvNext-T
FLOPs (G)	1.1	1.1	1.1	1.1
Initialization	Checkpoint Recycling	Checkpoint Recycling	ImageNet-21k	ImageNet-21k
MoE Layers	7:12	10:18	-	-
Core Expert Number	98	[98, 24]	-	-
Universal Expert Number	196	[196, 48]	-	-

B Experiment Settings and Time Costs

In this section of the appendix, we provide a comprehensive description of the training settings used in our experiments. Tab. 6 outlines the standard training configuration utilized across our experiments. Tab. 7 details the dataset-specific training configurations, capturing variations in batch size, warmup epochs, total training epochs, and drop path rates for each dataset employed in our experiments.

Our experiments were conducted on RTX 4090 GPU. Training V-JetMoE-T on the CIFAR-100 dataset (60,000 images) required 2.5 GPU hours while training on the ImageNet-1K dataset (1,281,167 images) required 120 GPU hours. Training C-JetMoE-F on CIFAR-100 also required 2.5 GPU hours and 156 GPU hours on ImageNet-1K. For V-JetMoE-S, training on CIFAR-100 required 8 GPU hours and 200 GPU hours on ImageNet-1K. Compared to the original dense models (ViT-Tiny, ConvNeXt-Femto, ViT-Small), our method achieves nearly equivalent training times.

For all the experiments presented in our paper, we required $3,300$ GPU hours for training. In total, we spent approximately $8,000$ GPU hours for exploration and validation of our work.

Table 6: Our basic recipe for model training.

Training Setting	Configuration
image resolution	224×224
optimizer	AdamW[52]
base learning rate	4×10^{-3}
weight decay	0.05
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$
batch size	4096
training epochs	300
learning rate schedule	cosine decay
warmup epochs	50
warmup schedule	linear
randaugment [53]	$(9, 0.5)$
mixup [54]	0.8
cutmix [55]	1.0
random erasing [56]	0.25
label smoothing [57]	0.1
layer scale [58]	1×10^{-6}

Table 7: Hyper-parameter setting on ViT-T.

Setting	Batch Size	Warmup Epochs	Training Epochs	Drop Path Rate
C-10	512	50	300	0.1
C-100	512	50	300	0.1
Pets	512	100	600	0.1
Flowers	512	100	600	0.1
STL-10	512	50	300	0
Food101	512	50	300	0.1
DTD	512	100	600	0.2
IN1k	4096	50	300	0

C Implementation of SpheroMoE Layer

Algorithm 1: Simple implementation of SpheroMoE.

```
def parallel_expert_forward(x, experts)
    """
    Traditional MoE models use a for loop to process each token through the experts.

    By merging all expert weights into a large matrix, our implementation allows
    for a single matrix multiplication operation for each layer across all tokens
    and experts, replacing multiple individual operations.
    """
    x = einsum(x, experts.weight_1, "b e s d1, e d2 d1 -> b e s d2")
    x = x + rearrange(experts.bias_1, "e d2 -> () e () d2")
    x = experts.act(x)
    x = einsum(x, experts.weight_2, "b e s d2, e d1 d2 -> b e s d1")
    x = x + rearrange(experts.bias_2, "e d1 -> () e () d1")
    return x

def spheromoe_layer(X, Q, T, core_experts, univ_experts):
    """
    Performs the Spheromoe layer operation.

    Parameters:
    X (tensor): tensor with shape (batch, token_num, channel).
    Q (tensor): tensor with shape (expert_num, slots_per_expert, channel).
    T (float): temperature parameter for the softmax function.
    core_experts, univ_experts (expert): expert weight for MoE layer.

    Returns:
    tensor: Output tensor after applying the Spheromoe layer operations.
    """
    X_norm = inherit_layer_norm(X, dim=-1)
    Q_norm = l2_norm(inherit_layer_norm(Q, dim=-1))
    K = K_project(X_norm)

    # Compute similarity logits S.
    S = einsum(K, Q_norm, "b n d, e s d -> b n e s")

    # Add normal noise
    noise = normal_noise(S) * self.noise_mult
    S = S + noise

    # Apply softmax to similarity logits.
    Dispatch = softmax(S/T, dim=1)
    Combine = softmax(S/T, dim=[-1,-2])

    # Token dispatch.
    X_hat = einsum(Dispatch, X_norm, "b n d, b n e s -> b e s d")
    X_core = X_hat[:, :core_num, :, :]
    X_univ = X_hat[:, core_num:, :, :]

    # Using core experts and universal experts processes each slot.
    Y_hat = stack([
        parallel_expert_forward(X_core, core_experts),
        parallel_expert_forward(X_univ, univ_experts)
    ], dim=1)

    # Expert dropout.
    Y_hat = expert_drop(Y_hat)

    # Token combine.
    Y = einsum(Combine, Y_hat, "b n e s, b e s d -> b n d" )

    return Y
```

D Dynamic Allocation and Focus Regions of Experts in MoE Jetpack

In this section, we discuss the dynamic allocation and focus regions of core and universal experts across different layers of MoE Jetpack. We used the same test images as in the main text, visualizing the focus regions of the most important (i.e., those with the highest output contribution) core and universal experts for each MoE layer in Fig. 7. The corresponding contribution values for these experts are listed in Tab. 8.

Our findings are as follows: Initially, in the shallower network layers (MoE Layer 7 and 8), the core experts contribute less than the universal experts, and their focus regions are relatively dispersed. As the network deepens, in MoE Layer 9, the most important core and universal experts show similar contribution values and focus regions. With further depth (MoE Layers 10, 11, and 12), the dominance of the core experts becomes increasingly evident, with significantly higher contribution values than the universal experts. Core experts focus on prominent objects in the images and are inclined to capture global information.

These experts' dynamic allocation and different focus region tendencies are crucial to our method. Different experts have varying capabilities in extracting information at various granularities, and the network facilitates collaboration among these experts to produce the final output. This illustrates the effective utilization of expert diversity in the MoE model.

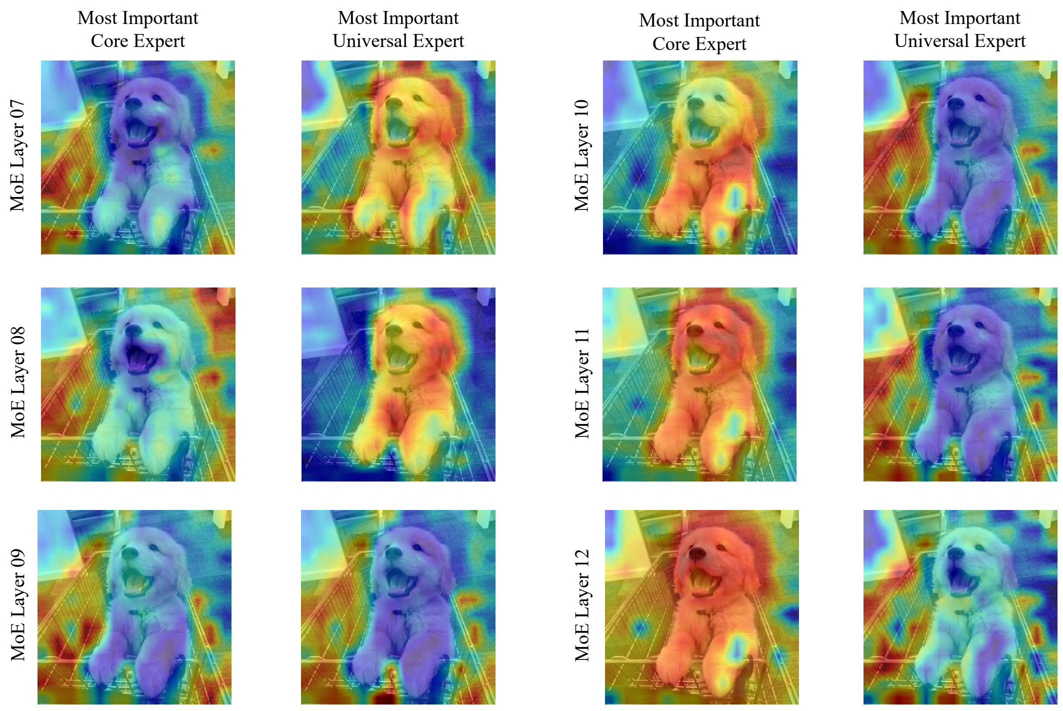

Most Important Core Expert
Most Important Universal Expert
Most Important Core Expert
Most Important Universal Expert
MoE Layer 07
MoE Layer 08
MoE Layer 09
MoE Layer 10
MoE Layer 11
MoE Layer 12

Figure 7: Visualization of the attention map identified by the most important core experts and universal experts across different layers (MoE Layer 07 to MoE Layer 12). The images show the regions deemed most relevant by each type of expert at each layer.

Table 8: Contribution values of core and universal experts across network layers.

MoE Layer	Core Expert Contribution	Universal Expert Contribution
7	1.71	3.91
8	2.52	4.16
9	3.78	3.77
10	8.17	6.71
11	17.66	2.12
12	7.36	0.77

E Broader Impacts

The proposed MoE Jetpack framework significantly enhances the accessibility and efficiency of MoE models by utilizing pre-existing dense checkpoints to substantially reduce the computational costs associated with training these models from scratch. This method not only minimizes the environmental footprint by decreasing the reliance on extensive GPU resources but also bridges the resource gap, facilitating wider adoption and fostering innovation across the AI community. Additionally, our commitment to open-sourcing all experimental code promotes greater transparency and collaboration in research. We have carefully considered the potential societal impacts of our method and believe it does not pose any significant ethical or fairness concerns, thereby ensuring its responsible application.