Efficient Data Driven Mixture-of-Expert Extraction from Trained Networks

Uranik Berisha[1,2], Jens Mehnert[1] and Alexandru Paul Condurache[1,2]
[1]Automated Driving Research, Robert Bosch GmbH, 70469 Stuttgart, Germany
[2]Institute for Signal Processing, University of Lübeck, 23562 Lübeck, Germany
{Uranik.Berisha,JensEricMarkus.Mehnert,AlexandruPaul.Condurache}@de.bosch.com

Abstract

Vision Transformers (ViTs) have emerged as the state-of-the-art models in various Computer Vision (CV) tasks, but their high computational and resource demands pose significant challenges. While Mixture-of-Experts (MoE) can make these models more efficient, they often require costly retraining or even training from scratch. Recent developments aim to reduce these computational costs by leveraging pretrained networks. These have been shown to produce sparse activation patterns in the Multi-Layer Perceptrons (MLPs) of the encoder blocks, allowing for conditional activation of only relevant subnetworks for each sample.

Building on this idea, we propose a new method to construct MoE variants from pretrained models. Our approach extracts expert subnetworks from the model's MLP layers post-training in two phases. First, we cluster output activations to identify distinct activation patterns. In the second phase, we use these clusters to extract the corresponding subnetworks responsible for producing them. On ImageNet-1k recognition tasks, we demonstrate that these extracted experts can perform surprisingly well out of the box and require only minimal fine-tuning to regain 98% *of the original performance, all while reducing MACs and model size, by up to* 36% *and* 32% *respectively.*

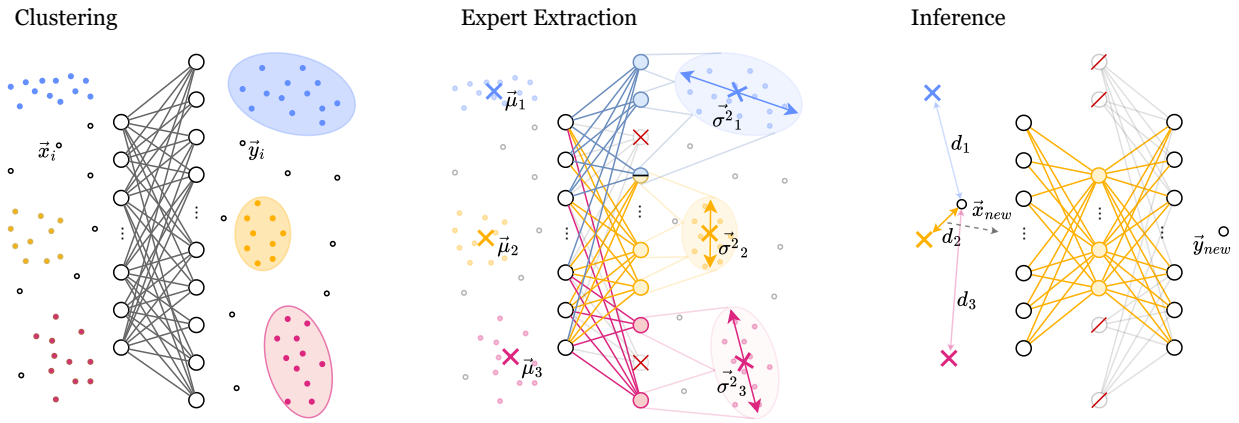
1. INTRODUCTION

Convolutional Neural Networks (CNNs) have long dominated CV tasks such as image recognition [6]. However, recent advancements have increasingly shifted towards transformers as the base architecture [6, 25]. These models have achieved remarkable results but at a significant computational cost [6]. Approaches, such as MoE methods [2, 3, 23], address these computational demands during inference but require expensive and complex training procedures, including additional load-balancing loss terms to ensure balanced expert selection [14]. Moreover, these models must be trained from scratch on massive datasets like ImageNet-21k, containing over 14 million samples, or Google's JFT-300M dataset, comprising 300 million samples [6]. Other solutions, such as DeiT [25], aim to mitigate the dependency on large datasets by employing strong data augmentations along with regularizations.

In this work, we approach the problem from a different angle by making use of the vast collection of pretrained networks available. Recent studies have shown that MLPs in pretrained language transformers tend to group neurons into subnetworks responsible for distinct functions [30]. For most input, Zhang et al. [29] show that the MLPs exhibit a high sparsity in their activation patterns, which they use to form MoE variants of prevalent language transformers. Since vision transformers must process spatially structured image data, the transferability of methods from language transformers, with their sequential structure, is not trivial. Text tokens can exhibit different activation and modularity patterns due to linguistic rather than spatial structures [6].

Building upon these works, our goal is to investigate whether similar phenomenons emerge in ViTs and leverage them to reduce the size of pretrained ViTs post-training. To this end, we introduce a novel expert extraction method that determines the number and size of the experts from the data. This grounds expert configurations in data-driven insights rather than manual selection, enhancing the robustness of the method. Our expert extraction method consists of two phases. First, we cluster the hidden activations in the MLPs using the Hierarchical Density Based Spatial Clustering for Applications with Noise (HDBSCAN) algorithm [1]. We choose this algorithm, as it allows us to find the number and shape of the experts from the data. In the second phase, we use these clusters to extract subnetworks from the linear layers, prioritizing components based on their variance within the activation patterns. During inference, a simple and efficient routing mechanism, derived from the similarity to the input cluster mean of the corresponding subnetwork, allows us to conditionally select the appropriate expert for processing.

This extraction process, performed once post-training, generates a MoE variant of the model, which is why we refer to our method as "expert extraction".



(a) Activations in the hidden layer of the MLP $\vec{y}_i$ are clustered using HDBSCAN, which automatically selects the number of clusters.

(b) 1. Hidden neurons with the highest variance within clusters $\vec{\sigma^2}_c$ are extracted. 2. Clusters are mapped back to the input space and averaged to produce a mean input token $\vec{\mu}_c$ for each expert.

(c) Incoming tokens $\vec{x}_{new}$ are routed based on similarity d_c to the mean input tokens $\vec{\mu}_c$. The selected expert is constructed by removing the hidden neurons not extracted in the expert extraction.

Figure 1. Illustration of the expert extraction process. The process begins with the clustering of activations (a), followed by the extraction of experts from these clusters (b), and finishes with the inference stage, using the extracted experts (c).

Our contributions are summarized as follows:

- We introduce a simple yet effective and efficient method for extracting experts from pretrained models, allowing their conversion into MoE variants.
- We apply our method to ViTs and demonstrate that we can reduce the computational cost and memory footprint while preserving model performance.
- We explore the behavior of our method and the resulting MoE model on different inputs, to find general structural tendencies of emerging substructures and the specialization in their MLP layers.

2. RELATED WORK

2.1. Transformers

The transformer architecture, first introduced by Vaswani et al. [26], revolutionized Natural Language Processing (NLP) by employing self-attention mechanisms to capture long-range dependencies. This architecture was later adapted for CV by Dosovitskiy et al. [6] with the introduction of ViTs, where images are split into patches that serve as tokens. Although ViTs achieve state-of-the-art performance, their success depends on the availability of very large datasets and substantial computational resources, making them challenging to train [25]. To address this issue of high data dependency, Touvron et al. [25] proposed a series of data augmentation, regularization, and distillation techniques to reduce reliance on large datasets. Despite these innovations, the computational demands of ViTs remain a bottleneck for practical deployment, particularly on resource-constrained hardware [28, 31].

An emerging research direction involves understanding how transformers store knowledge [9], whether this knowledge can be modified [4], and consequently, can be extracted to conditionally activate only subnetworks, to reduce computational costs during inference. [29]

2.2. Model Compression

To make ViTs more practical for deployment on limited hardware, model compression techniques are essential. Pruning methods, which reduce model size by eliminating less important components, remain among the most prevalent approaches [22, 27, 28, 31]. Another promising direction for model compression is clustering-based methods, where clustering algorithms are used to group tokens that can be processed together, improving computational efficiency without sacrificing performance [10, 15, 16].

While we also make use of clustering algorithms, our work is most closely related to dynamic inference through MoEs. MoEs divide the model into specialized experts and dynamically route inputs to these experts, thus reducing computational costs while maintaining a large number of parameters [13]. Inspired by Jacobs et al. [11], MoEs were successfully extended to Deep Learning (DL) by Eigen et al. [7], scaling MoEs for deep models. This led to significant advancements in both NLP [8, 24] and CV [2, 3, 23], where MoEs have proven particularly effective in transformer-based architectures, enabling efficient scaling [23]. However, conventional MoEs typically define experts before training, which introduces complexities in the training process. Ensuring balanced expert selection requires additional load-balancing loss terms [14].

Avoiding this, Zhang et al. [29] introduced a novel technique for extracting experts post-training in transformer-based large language models, which aligns with our work.

2.3. Our Approach

Our work builds upon the research on emerging subnetworks in pretrained language transformers [29], which addresses the limitations of traditional MoEs by enabling the extraction of experts post-training. This offers a more flexible approach to training MoEs through standard training procedures or even to omit the training step alltogether by using pretrained models. We investigate these findings and explore these emerging structures within the context of vision. In contrast to Zhang et al. [29], we employ a data-driven approach to the expert configuration, extracting only expert structures that naturally arise in different layers. To do so, we group activations with similar patterns across layers and thereby identify possible subnetworks based on the variance of these activation clusters.

3. METHODOLOGY

In this section, we introduce our method for extracting experts and explain how these experts are structured, represented, and used during inference. As is common in MoE literature, the MLPs are partitioned into experts that operate on the token level. We therefore employ a token-wise clustering of the activations in the hidden linear layers to identify subnetworks, which can then dynamically process individual tokens, through efficient routings.
Our method consists of three major steps:

1. **Activation Clustering** The layer activations are clustered over multiple batches (Fig. 1a).
2. **Expert Extraction** For each cluster, a subnetwork is identified based on the cluster variance (Fig. 1b).
3. **Inference** The tokens are routed to the subnetworks based on the clusters to generate an expert-specific output (Fig. 1c).

3.1. Step 1: Activation Clustering

First, we need to identify clusters of similar activations. To achieve this, we assume that a randomly sampled subset of the data can sufficiently represent the underlying activation patterns (see Section 5.4 for further details). Based on this assumption, we record the activations $\mathbf{y}_i$ of individual tokens $\mathbf{x}_i$ in each hidden layer, which are then clustered.

While our method does not require a specific clustering algorithm, it must enable a data-driven estimation that avoids restrictive assumptions about the number of clusters k or potential activation patterns, as we want to extract the number and shape of the experts from the data. Given the high dimensionality e of the embedding space, the clustering algorithm must also be computationally efficient and be able to handle noise.

An algorithm fulfilling these constraints is HDBSCAN, a robust clustering algorithm well-suited for high-dimensional data. It has one primary hyperparameter, *minimum cluster size*, which acts like a blurring filter by determining the minimum number of points needed to form an individual cluster. This merges smaller noise clusters into larger, more significant ones and thus controls the granularity of the experts. It works well across a range of values, which makes it easy to determine (see Section 4.1 for further details). HDBSCAN's flexibility allows us to apply it across all layers. This avoids manual tuning of further hyperparameters, as it has only data-dependent decisions on where specialization patterns emerge. This means that the number of experts in a layer l is given by the number of clusters detected k_l. If no clusters are detected in a layer, we assume that the linear layer has not yet formed specialized subnetworks and is instead processing general features, so we leave the layer unchanged.

3.2. Step 2: Expert Extraction

Given these activation clusters, our goal is to form the desired experts by extracting specialized subnetworks. This requires representing these clusters in a way that facilitates both subnetwork extraction and an efficient comparison of new inputs with existing clusters, to allow effective input routing.

3.2.1. Expert Representation

We use simple descriptive statistics, specifically the variance $\vec{\sigma^2}_c$ of the activation vectors for each cluster c, to identify which components are most critical. Activations with higher variances within the clusters capture more diverse patterns, while those with lower variance are assumed to contribute less to the specificity of the cluster (see Section 5.2 for further details).

After prioritizing activation components based on their variance, we extract the experts by selecting the subset of hidden neurons within the MLP that produced these components, thus identifying parts of the network that are most important for representing the clusters. This corresponds to choosing columns in the weight matrix $\mathbf{W}^{(1)}$. Importantly, experts are extracted independently, allowing them to overlap and not necessarily cover all weights. This independent extraction enables the removal of neurons not selected by any expert, reducing the parameter count and minimizing the memory footprint. To further optimize memory, weights for each expert are shared across the layer, eliminating the need to store redundant parameters.

Inspired by other variance-based algorithms, such as Principal Component Analysis (PCA), we define a hyperparameter *extraction percentage* $p\%$, which determines how much of the total variance the cumulative variance of selected output neurons must cover.

This parameter reflects how much information we want to keep, allowing adjustment of the model's level of detail (see Section 4.1 for further details).

3.2.2. Input Cluster Representation

Because linear transformations preserve clusters, we can map clusters from the activation space back to their respective inputs, thereby forming input clusters. To assess which cluster a new input $\vec{x}_{new}$ belongs to, several methods can be used, such as explicit density estimations or k-nearest neighbors. However, these approaches are resource-intensive, which may be impractical in real-time inference or on resource-constrained hardware. Instead, we make a simplifying assumption that the clusters are spherical. This allows us to represent each cluster by its mean input vector $\vec{\mu}_c$, similar to the K-Means algorithm. We then compute the mean input vector for each cluster, which we use for efficient routing of new inputs. We make this simplifying assumption in favor of a faster compute, by relying on the flexibility of the neural network to compensate for minor errors during the fine-tuning phase, where the model can adjust its parameters to correct routing inaccuracies.

3.3. Step 3: Inference

With the precomputed mean input vectors $\vec{\mu}_c$ and the extracted sub-networks for each cluster, we now describe how the model operates during inference. The process involves routing tokens to the appropriate expert and generating expert-specific outputs.

3.3.1. Expert Routing

For each token, we calculate the pairwise cosine similarity d_c (see Section 5.3 for further details) between the input token vector $\vec{x}_{new}$ and the mean input vectors of all experts $\vec{\mu}_c$. The cluster with the highest cosine similarity is then selected as the most appropriate match, and the token is routed to the corresponding expert. This token-wise routing ensures that each token is processed by the expert most specialized for handling its features.

3.3.2. Expert Output

Once routed to an expert, the token is processed by a subset of weights from the original linear layer, selected using binary masks $\mathbf{M}$ that correspond to columns of the weight matrix $\mathbf{W}^{(1)}$. Since each expert uses fewer neurons than the full linear layer, the output may not match the original embedding dimension e. To address this mismatch, the input layer of the second linear layer is reduced accordingly by removing the corresponding rows of the weight matrix $\mathbf{W}^{(2)}$, using the transposed binary mask $\mathbf{M}^T$. This ensures that outputs from different experts are compatible and further reduces the computational load.

3.3.3. Computational Efficiency

Our approach saves resources by removing neurons that are unused by any expert and by reducing the number of hidden neurons computed for each token. Although the routing mechanism introduces a small overhead due to the matrix multiplication required for calculating the pairwise cosine similarity, this overhead is negligible compared to the savings gained from reducing the size of computed activations. Specifically, for k experts, we perform dot products with cluster means of dimension e, resulting in a $k \times e$ matrix multiplication with each incoming token. In contrast, the upward projection from e to a hidden representation of size $3e$, amounts to a $3e \times e$ matrix multiplication per token. Because $k \approx 10 << 3e$ (see Appendix Table 9), the savings significantly outweigh the additional routing cost.

4. EXPERIMENTS

We apply our expert extraction method to standard vision transformer architectures, namely DeiT-Tiny, DeiT-Small, and DeiT-Base [25], pretrained on the ImageNet-1k dataset [5]. This results in their respective Mixture-of-Extracted-Experts (MoEE) forms: DeiT-T-MoEE, DeiT-S-MoEE, and DeiT-B-MoEE. We compare the results with our reimplementation of Zhang et al. [29] for ViT, denoted by the MoEfication tag in the respective experiments. Before fine-tuning, we evaluate the performance of these models directly on ImageNet-1k to measure how much accuracy is retained immediately after the expert extraction.

The models are then fine-tuned for 30 epochs with a batch size of 32 using knowledge distillation from the corresponding unmodified model. We use the AdamW optimizer [20] with a weight decay of 0.01 and an initial learning rate of 1.5e-5, which is decayed using a cosine annealing scheduler [19]. To demonstrate our commitment to efficient machine learning and to highlight the efficacy of our method, all models are evaluated and fine-tuned on a single NVIDIA Tesla T4 GPU each.

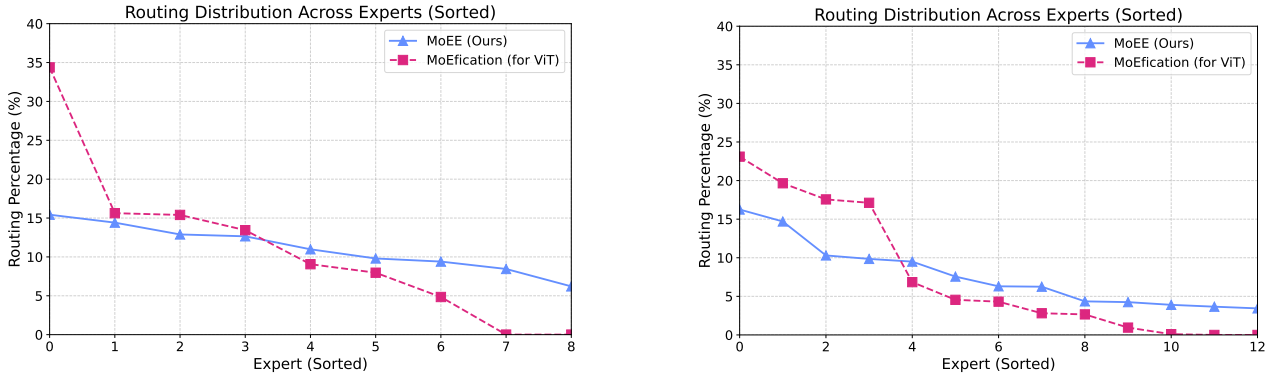
4.1. Effect of Hyperparameters

Our method introduces two key hyperparameters, namely the *minimum cluster size* and the *extraction percentage*, which can be tuned to extract more or less and bigger or smaller experts respectively. Both of which influence the trade-off between MACs reduction and model accuracy.

When using HDBSCAN, as the *minimum cluster size* increases, the number of distinct experts decreases because variations in token density are increasingly interpreted as noise within larger clusters. This results in less specialized experts, requiring more neurons to represent the cluster. Consequently, increasing the minimum cluster size while keeping the extraction percentage constant leads to a reduction in MACs savings but improves accuracy retention.

Model	MACs (G)	Parameters (M)	Acc. Retention (%)	Top-1 Acc. (%)
DeiT-T	1.26	5.72	–	72.02
DeiT-T-MoEfication	**0.94 (-27.4%)**	5.72	**57.41**	68.91
DeiT-T-MoEE (ours)	0.95 (-27.0%)	**4.57 (-20.1%)**	55.33	**69.73**
DeiT-S	4.61	22.05	–	79.70
DeiT-S-MoEfication	3.31 (-29.0%)	22.05	67.08	77.10
DeiT-S-MoEE (ours)	**3.19 (-30.6%)**	**16.54 (-25.0%)**	**67.60**	**78.11**
DeiT-B	17.58	86.57	–	81.73
DeiT-B-MoEfication	11.41 (-34.8%)	86.57	57.20	77.63
DeiT-B-MoEE (ours)	**11.14 (-36.3%)**	**58.55 (-32.4%)**	**68.54**	**80.12**

Table 1. Performance and parameter comparison, evaluating four metrics: **Accuracy Retention**: retained accuracy after expert extraction, before fine-tuning; **Top-1 Accuracy**: final accuracy after fine-tuning; **MACs**: computational operations, measured in billions of operations; and **Parameters**: the total model size in millions of parameters. Our method (MoEE) achieves competitive accuracy with significant reductions in MACs and parameters, especially in the DeiT-S and DeiT-B models.



(a) Layer 6 (b) Layer 11

Figure 2. Sorted routing distributions across experts for all ImageNet-1k classes at different layers, demonstrating a more balanced routing load compared to ViT-MoEfication across different layers, effectively reducing the need for additional load-balancing terms.

We performed a hyperparameter search, evaluating minimum cluster sizes ranging from 0.05% to 1.5% of the total token sample size for the extraction. We identified 0.6% as a favorable trade-off between MACs reduction and accuracy.

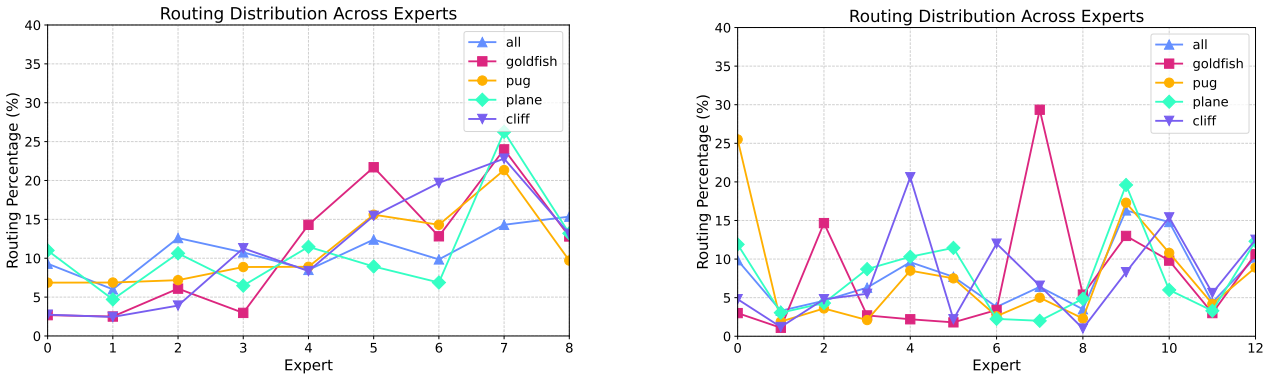
The *extraction percentage* determines how much of the representational capacity of each expert is retained during extraction. Higher percentages result in larger experts in terms of neuron count, leading to better model accuracy but reduced MACs savings. Through a hyperparameter search, we find that an extraction percentage between 70% and 90% offers a good balance between computational savings and accuracy. We use an extraction percentage of 80% for all following experiments (see Appendix Section 8 for further details).

4.2. Mixture-of-Expert Extraction on ImageNet

Table 1 shows the main results of our method compared to the baseline DeiT architectures and the MoEfication approach.

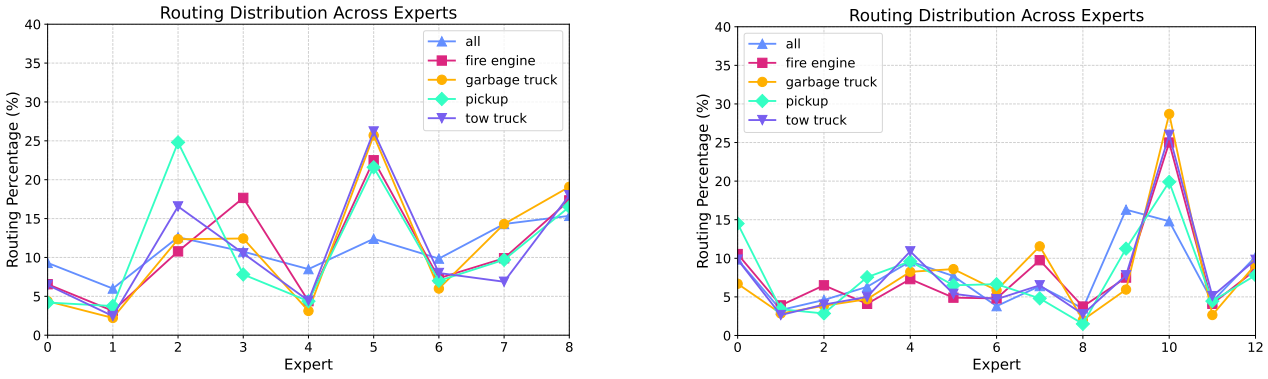
In our evaluation we report the Top-1 Accuracy before finetuning (Acc. Retention), the Top-1 Accuracy after finetuning (Top-1 Acc.), the MACs and the parameters. As the two key metrics, we mainly consider the final accuracy of our models after finetuning and the computational savings.

The expert extraction reduces MACs by 27%, 31% and 36%, while also reducing the memory footprint by 20%, 25% and 32% for the tiny, small and base variants respectively. Notably, the larger the model, the more effective our method appears to be in both accuracy retention and MACs reduction. For instance, the DeiT-B-MoEE model achieves a significant MACs reduction of 36.3% and a memory reduction of 32.4% while maintaining 84% of the Top-1 Accuracy of the unmodified DeiT-B baseline out of the box, and regaining 98% of the original performance. We demonstrate the generalizability of our method to other architectures and dataset by applying the expert extraction on Swin- [17] and ConvNeXt Models [18] as well as the DeiT models trained on CIFAR-100 [12] (see Appendix Section 11).



(a) Layer 6 (b) Layer 11

Figure 3. Token routing distributions at different layers for randomly selected classes (goldfish, pug, plane, and cliff) compared to the distribution across all ImageNet-1k classes (all). Layer 6 shows a relatively even distribution across experts, indicating less class-specific specialization, Layer 11 shows distinct spikes in the routings, as tokens are more selectively routed based on class-specific features.



(a) Layer 6 (b) Layer 11

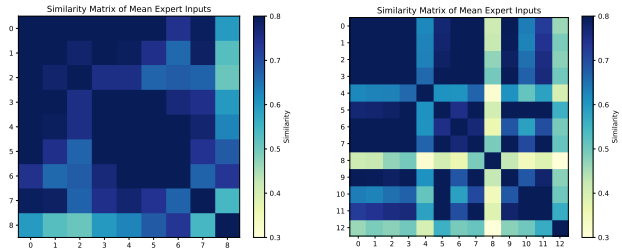
Figure 4. Token routing distributions at different layers for visually similar truck-like classes (fire engine, garbage truck, pickup, and tow truck) compared to the distribution across all ImageNet-1k classes (all). Both layers shows similar distributions across experts for all truck-like classes. This indicates a processing through similar expert selections, reflecting the effectiveness of the routing mechanism.

When compared to MoEfication (see Appendix Section 10 for an overview of the differences), our approach particularly improves the Top-1 Accuracy. In the DeiT-B variant, DeiT-B-MoEE achieves a 2.5%-points higher Top-1 Accuracy with 1.5%-points higher MACs reduction, while simulatneously removing 32.4% of the parameters. This suggests that our data-driven expert extraction process is more effective in capturing relevant activation patterns, allowing for an improved trade-off between computational savings and accuracy, particularly for larger vision models.

Figure 2 presents the sorted routing distributions across experts for both our method and our implementation of the MoEfication approach adaptet for ViTs, highlighting differences in load balancing between the two methods. As noted in Zhang et al. [29], the MoEfication approach tends to exhibit an unbalanced routing distribution, with certain experts receiving a disproportionately high percentage of tokens while others remain underutilized. This imbalance necessitates additional load-balancing terms in conventional MoEs to maintain a more even distribution. In contrast, our method achieves a significantly more balanced distribution, with routing percentages closer to the optimal load of around 11%, at Layer 6, and around 8%, in Layer 11. This balanced routing distribution highlights another advantage of our approach, as it avoids the need for supplementary load-balancing mechanisms.

4.3. Expert Routing Analysis

To better understand the behavior of our approach, we analyze the distribution of tokens routed to each expert at different layer depths. To do this, we count the number of tokens routed to each expert during the evaluation of validation data across individual classes. We then compare randomly selected classes, visually similar classes (selected manually), and the overall routing behavior across all classes to highlight differences in routing patterns.



(a) Layer 6 (b) Layer 11

Figure 5. Similarity matrices of mean expert inputs at different layers, illustrating the relationships between inputs of different experts. Layer 6 shows high similarities, with several experts' inputs being closely related, indicating less specialization. Layer 11 exhibit distinct differences, with some experts showing near-orthogonal inputs, indicating higher specialization.

In this analysis, we use the DeiT-B-MoEE model, where we apply our expert extraction to all 12 layers (indexed 0 to 11). However, since earlier layers of ViTs capture more general features, as noted by recent studies [21], HDBSCAN does not detect distinct expert clusters in the initial layers. Instead, we observe distinct expert formations only in layers 6 through 11 (see Appendix Table 9), as these deeper layers show sufficient variation in activation patterns. By examining the first and last extracted expert layers, we identify that experts in deeper layers show more specialized routing behavior, confirming that the network develops increasing specialization with depth, as shown in the following visualizations.

Figure 3 illustrates the token routing distributions for randomly selected classes: goldfish, pug, plane, and cliff. At Layer 6 (Fig. 3a), we observe that the routing behaviors do show class specific differences, but still are distributed across the available experts without much variance, suggesting that the earlier layers process these classes more generally. In contrast, at Layer 11 (Fig. 3b), the routing patterns become more distinct, with each class exhibiting clear spikes in expert selection. These spikes indicate that the model has learned to differentiate between these classes more effectively at this layer, assigning class-specific processing paths.

Figure 4 shows the token routing distributions for selected truck-like classes: fire engine, garbage truck, pickup, and tow truck. At both layers, the routing patterns across these classes show a high degree of overlap, suggesting that the model processes visually similar classes through similar expert paths in all layers. This consistent routing pattern across related classes reflects the effectiveness of the routing mechanism, even at layers with less specialization. Notably, the distribution for the truck-like classes closely aligns with the distribution of all 1000 ImageNet-1k classes.

HDBSCAN Clustering	Variance Extraction	Input Routing	Top-1 Acc. (%)
✗	✗	✗	58.64
✓	✗	✗	75.36
✓	✓	✗	77.68
✓	✓	✓	80.12

Table 2. Ablation study results showing the impact of each introduced component (HDBSCAN clustering, variance-based extraction, and input-based routing) on Top-1 Accuracy. Each row adds a component to demonstrate its effect, with all components together yielding the highest accuracy (80.1%).

This contrasts with other groups of visually similar classes, such as shark-like classes (shown in Figure 8), which also display a cohesive routing pattern among themselves but differ significantly from the distribution of all classes. This suggests that the truck images contain more generic features that are common across many classes.

Figure 5 compares the similarity matrices of mean expert inputs. In Layer 6 (Fig. 5a), the similarity between the mean inputs is relatively high, with several experts (specifically experts 0, 1, 3, 4, and 5) showing closely related mean tokens with a cosine similarity of over 0.8. This again suggests a lower level of specialization among experts at this earlier layer. By contrast, the similarity matrix at Layer 11 (Fig. 5b) reveals a more diverse distribution of experts, with distinct differences between them. Notably, experts 12 and 8 exhibit near-orthogonal relationships with most other experts, indicating a higher degree of specialization.

5. ABLATION

To further evaluate our design choices, we conduct an ablation study and test the assumptions made in the methodology. This provides deeper insights into the effect of these assumptions on model performance. We systematically remove or replace each component of our methodology with a random counterpart and evaluate the resulting ablated models after the previously described fine-tuning. This includes selecting random clusters, selecting random hidden neurons for the expert extraction, and routing incoming tokens to random experts.

The results of our ablation study, presented in Table 2, showcase the usefulness of each component in our methodology. When HDBSCAN clustering alone is used without variance-based extraction or input-based routing, the model achieves a Top-1 Accuracy of 75.4%, an improvement of 16.8 percentage points over the baseline with random expert extraction (58.6%). Adding the variance-based neuron selection increases this accuracy by 2.3 percentage points, demonstrating that our variance-based prioritization of neurons contributes significantly to the expert extraction.

Including our input-based routing raises the Top-1 Accuracy by an additional 2.4 percentage points, demonstrating the effectiveness of routing tokens to the cluster with the most similar input mean. Together, these results confirm the efficacy of each component.

5.1. Alternative Clustering Methods

While our method uses HDBSCAN for its ability to extract a variable number of clusters per layer without manual tuning, we also evaluate alternative clustering algorithms. These include DBSCAN and OPTICS as related density-based methods, and K-Means and BIRCH as partition-based baselines. We find that density-based methods perform significantly better in accuracy retention as well as final accuracy, with HDBSCAN consistently yielding the best results across all metrics (see Appendix Section 12).

5.2. Variance vs. Magnitude-Based Extraction

In our variance-based approach, components with higher variance are prioritized, assuming they capture more relevant information. An alternative method would be to prioritize based on the magnitude of mean activations, where components with higher mean activations are assumed to be more significant. While both methods select important components, the variance-based approach better captures the diversity in activations, which proved to be more beneficial for expert extraction (see Appendix Section 13).

5.3. Alternative Routing Methods

We also compare the cosine similarity to a pairwise Euclidean distance for expert routing. The expert with the smallest Euclidean distance is selected as the best match. Both routing approaches yield similar end accuracies. However, cosine similarity offers computational efficiency as it can be calculated via a single unnormalized matrix multiplication, since we only need to select the relative distances. This advantage makes cosine similarity our default choice for routing (see Appendix Section 13).

5.4. Effect of Sample Size on Extraction Stability

To validate our assumption that a randomly sampled subset of the data can sufficiently represent the underlying activation patterns, we vary the number of tokens used for the extraction and look at the number and size of experts extracted by our method. To this end, we find that stability of our method is indeed affected by the number of samples used for extraction. At very low sample sizes, the extraction is highly dependent on the specific samples chosen, leading to variability in expert configurations. However, starting from approximately 100,000 tokens (which corresponds to around 500 sample images), our method begins to show significant reductions in variance between different sample sets. Notably, these 500 images do not even cover all 1,000 ImageNet-1k classes. We attribute this early stabilization to the presence of common tokens across classes, similar to the routings of truck-like classes, where the tokens follow the routing distribution of all classes (see Appendix Section 9).

6. DISCUSSION

Our approach of extracting experts post-training and fine-tuning the model offers two key advantages over traditional methods of constructing MoEs. Firstly, it enables the creation of MoE variants from existing ViTs without the need for retraining from scratch, allowing significant computational savings when working with pretrained models. Secondly, it eliminates the need for specialized training procedures typically associated with MoEs. By training the base model using standard optimization algorithms and extracting experts afterwards, this method can lead to faster overall training times, as the fine-tuning phase is relatively lightweight compared to full retraining.

Although the accuracy retention after extraction is not sufficient for direct deployment, it is noteworthy that the model retains substantial accuracy given the major structural changes introduced during expert extraction. By conditionally activating specific subnetworks for each input, our approach retains a high capacity for diverse representations, as experts can specialize in processing different input types, while still reducing the computational load per token. This enables an immediate evaluation of the model and provides a reliable estimate of the accuracy-to-savings trade-off without requiring fine-tuning, allowing for straightforward hyperparameter selection.

7. CONCLUSION

We introduce a method for constructing MoE-variants from pretrained ViTs, with minimal computational costs compared to training a new MoE model from scratch. Demonstrated by the experiments, our approach offers significant efficiency gains post-training, while maintaining comparable performance levels. Extensive ablation studies further highlight the effectiveness of our method.

In addition to demonstrating practical gains, our work provides analytical insights into activation clustering, expert formation, and the modularity emerging in ViTs at varying depths. For future work, we plan to explore extensions of this method to other architectures, broadening its applicability and impact, and to investigate potential combinations with pruning and other compression techniques to achieve further gains in efficiency.

We believe the novelty and simplicity of baseing our approach on cluster statistics offers a new perspective on model compression and optimization, and we hope it inspires further developments in the CV research community.

References

[1] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 1

[2] Tianlong Chen, Xuxi Chen, Xianzhi Du, Abdullah Rashwan, Fan Yang, Huizhong Chen, Zhangyang Wang, and Yeqing Li. Adamv-moe: Adaptive multi-task vision mixture-of-experts. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 17300–17311. IEEE, 2023. 1, 2

[3] Erik A. Daxberger, Floris Weers, Bowen Zhang, Tom Gunter, Ruoming Pang, Marcin Eichner, Michael Emmersberger, Yinfei Yang, Alexander Toshev, and Xianzhi Du. Mobile v-moes: Scaling down vision transformers via sparse mixture-of-experts. *CoRR*, abs/2309.04354, 2023. 1, 2

[4] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. 2

[5] 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*, pages 248–255, 2009. 4

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 2

[7] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. 2

[8] 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), 2022. 2

[9] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. 2

[10] Ryan Grainger, Thomas Paniagua, Xi Song, Naresh Cuntoor, Mun Wai Lee, and Tianfu Wu. Paca-vit: Learning patch-to-cluster attention in vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 18568–18578. IEEE, 2023. 2

[11] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, 1991. 2

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 5, 3

[13] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 2

[14] Hanxue Liang, Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, and Zhangyang Wang. M^3vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *CoRR*, abs/2210.14793, 2022. 1, 2

[15] James C. Liang, Yiming Cui, Qifan Wang, Tong Geng, Wenguan Wang, and Dongfang Liu. Clusterformer: clustering as a universal visual learner. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 2

[16] Weicong Liang, Yuhui Yuan, Henghui Ding, Xiao Luo, Weihong Lin, Ding Jia, Zheng Zhang, Chao Zhang, and Han Hu. Expediting large-scale vision transformer for dense prediction without fine-tuning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. 2

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021. 5, 3

[18] 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)*, pages 11966–11976, 2022. 5, 3

[19] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 4

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 4

[21] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 7

[22] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision

transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 13937–13949, 2021. 2

[23] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2021. Curran Associates Inc. 1, 2

[24] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 2

[25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 10347–10357. PMLR, 2021. 1, 2, 4

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 60006010, Red Hook, NY, USA, 2017. Curran Associates Inc. 2

[27] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 2964–2972. AAAI Press, 2022. 2

[28] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 3143–3151. AAAI Press, 2022. 2

[29] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland, 2022. Association for Computational Linguistics. 1, 2, 3, 4, 6

[30] Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4066–4083, Toronto, Canada, 2023. Association for Computational Linguistics. 1

[31] Mingjian Zhu, Yehui Tang, and Kai Han. Vision transformer pruning, 2021. 2