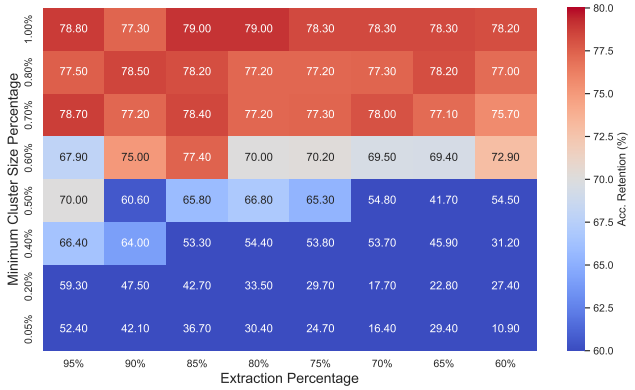
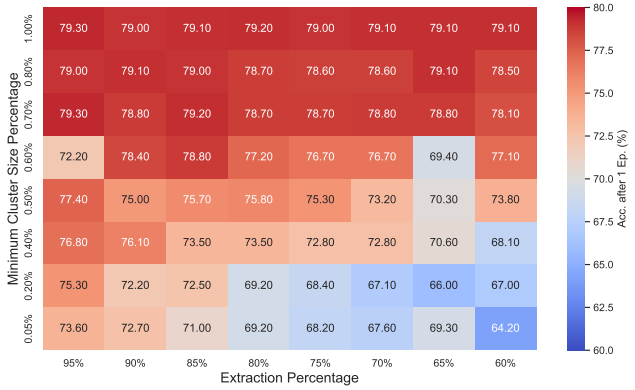
Efficient Data Driven Mixture-of-Expert Extraction from Trained Networks

Supplementary Material

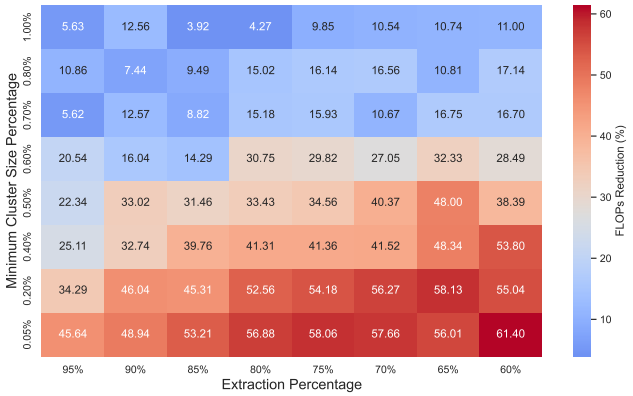


(a) Accuracy Retention After Expert Extraction on ImageNet-1k: The retention decreases significantly as the minimum cluster size percentage reduces below 0.6%.

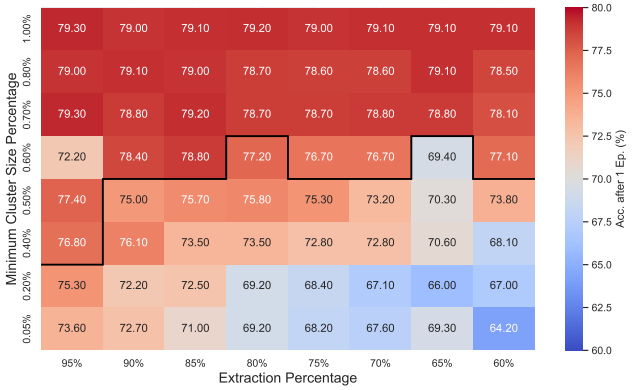


(b) Accuracy After One Epoch of Fine-Tuning on ImageNet-1k: The accuracy saturates at higher extraction percentages, with minimal gains for increasing the cluster size percentage beyond 0.6%.

Figure 6. Heatmaps as a function of extraction percentage and minimum cluster size: (a) immediately following expert extraction and (b) after a single fine-tuning epoch. A clear gradient is observed from bottom-right to top-left in both plots. Fine-tuning mitigates the impact of smaller cluster sizes, with saturation achieved at minimum cluster size percentages above 0.6%.



(a) MACs Reduction: The results demonstrate an increase in MACs Reduction with lower minimum cluster sizes and lower extraction percentages.



(b) Accuracy After One Epoch of Fine-Tuning on ImageNet-1k: The threshold highlights the region achieving significant computational efficiency gains.

Figure 7. Heatmaps as a function of extraction percentage and minimum cluster size: (a) MACs Reduction and (b) Top-1 Accuracy after a single fine-tuning epoch. The accuracy decreases as the MACs Reduction increases, highlighting the trade-off between computational efficiency and model performance. This suggests selecting the hyperparameters at the boundary where acceptable accuracy and computational efficiency intersect.

8. Sensitivity to Hyperparameters

Since we use a subset of the dataset as samples for expert extraction, to select an appropriate minimum cluster size for different sample sizes, we consider the minimum cluster size relative to the total number of samples as a percentage. Increasing this *minimum cluster size percentage* reduces the number of distinct experts because variations in token density are increasingly interpreted as noise within larger clusters by HDBSCAN. Since this reduction in the number of clusters corresponds directly to a reduction in the number of experts, this results in less specialized experts. Therefore, more neurons per expert are required, leading to improved accuracy but lower computational savings. Similarly, higher *extraction percentage* values yield larger experts by preserving more neurons, further enhancing accuracy but at the cost of reduced computational savings.

To select appropriate hyperparameters in this inverse relationship between computational efficiency and accuracy, we perform a hyperparameter search and show our results using heatmaps.

We use the DeiT-S model evaluated on ImageNet-1k and apply the same training settings as described in Section 4.

Figure 6 confirms the effect of our hyperparameters, on model accuracy. Figure 6a shows the Accuracy Retention immediately after expert extraction (without fine-tuning), while Figure 6b shows the accuracy after one epoch of fine-tuning. Both heatmaps show a clear gradient from the bottom-right to the top-left, indicating that larger cluster sizes and higher extraction percentages improve Accuracy Retention. These results also highlight the saturating effect of fine-tuning. After a single fine-tuning epoch, accuracy increases significantly for smaller cluster sizes, while cluster size percentages above 0.6% stagnate as they approach the baseline accuracy of 79.7%. Selecting our hyperparameters near this saturation threshold avoids significant accuracy drops while allowing fine-tuning to improve performance effectively. This suggests that hyperparameters can be selected either immediately after expert extraction or after a single epoch of fine-tuning, making the hyperparameter search simple and computationally efficient.

Furthermore, we aim not only to achieve baseline accuracy but also to achieve a significant reduction in MACs. For the hyperparameter selection, we therefore need to find the boundary region where computational efficiency and accuracy intersect. Figure 7 shows this trade-off between the computational efficiency and model performance. Figure 7a shows the MACs Reduction heatmap and Figure 7b again shows the accuracy after one epoch of fine-tuning, this time with a boundary higlighting a region that achieves a significant reduction in MACs, while maintaining a high accuracy. For our experiments, we select parameters along this threshold. As seen in Figure 7b, the most top-left point along this boundary corresponds to the best accuracy of 77.2% after a single fine-tuning epoch and a MACs Reduction of 30.75%. The selected hyperparameters for our experiments are thus a *minimum cluster size percentage* of 0.6% and an *extraction percentage* of 80%.

9. Sensitivity to Sample Size

To evaluate the effect of the sample size on our method, we use the DeiT-S model, applying the same training settings as described in Section 4 of the paper.

Table 3 presents the mean Top-1 Accuracy on ImageNet-1k over three random seeds, along with the standard deviation, for different numbers of input images used to identify clusters during expert extraction. The results indicate a clear trend: as the number of input samples increases, the standard deviation decreases. However, this improvement comes with a computational trade-off due to the non-linear runtime complexity of the HDBSCAN clustering algorithm with respect to the number of samples.

For our experiments, we select 640 input images, corresponding to 126,080 sample tokens clustered per layer.

Number of Input Images	**Top-1 Acc. (%)**
320	77.93 ± 2.281
640	78.23 ± 0.981
960	78.70 ± 0.529
1,280	78.67 ± 0.306
1,600	78.93 ± 0.404
1,920	79.00 ± 0.100
2,240	79.37 ± 0.153
2,560	79.00 ± 0.173

Table 3. Mean **Top-1 Accuracy (%)** of DeiT-S on ImageNet-1k as a function of the **Number of Input Images** used for the expert extraction. Each accuracy is presented alongside its standard deviation to demonstrate the variability. For a higher number of input images the standard deviation drops, indicating that the extraction procedure stabilizes.

Aspect	**MoEfication**	**Ours**
Expert Count	Manually Chosen	Data-Driven
Expert Structure	Disjoint Partitions	Overlapping Subnetworks
Routing Target	Mean Weight Columns	Mean Input Tokens
Domain Focus	Large Language Models	Vision Transformers

Table 4. Comparison of our method to MoEfication [29].

This configuration achieves a standard deviation below 1%, balancing runtime and accuracy consistency. Notably, this stabilization of the standard deviation and accuracy occurs at a number of input images (640) that cannot even represent each class in the ImageNet-1k dataset (1,000 classes). We attribute this to the fact that the clustering is based on tokens derived from image patches, which are redundant and shared across multiple classes, as seen in Section 14. This redundancy in token distributions allows for robust clustering even with fewer images than the total number of classes.

10. Comparison to MoEfication

Zhang et al. [29] rely on weight co-activation graphs and a manually set number of experts. In contrast, we cluster activations and let the data determine the number of experts automatically. Their disjoint partitioning contrasts with our variance-based extraction that allows overlapping experts, leading to fewer constraints on the experts. Moreover, while [29] routes tokens to the most similar mean weight column, we compute similarities in the input space directly (see Table 4). Our results on ImageNet-1k validate this design, showing performance gains in vision tasks.

Model	MACs (G)	Parameters (M)	Acc. Retention (%)	Top-1 Acc. (%)
Swin-T	4.50	28.29	–	81.19
Swin-T-MoEE (Ours)	3.59 (-20.2%)	17.17 (-39.3%)	58.14	79.02
Swin-S	8.76	49.61	–	83.20
Swin-S-MoEE (Ours)	6.19 (-29.4%)	30.65 (-38.2%)	51.16	81.21
Swin-B	15.46	87.77	–	83.47
Swin-B-MoEE (Ours)	10.42 (-32.6%)	53.05 (-39.6%)	50.17	83.12
ConvNeXt-T	4.47	28.59	–	82.12
ConvNeXt-T-MoEE (Ours)	3.53 (-21.0%)	18.57 (-35.0%)	35.28	81.82
ConvNeXt-S	8.71	50.22	–	83.11
ConvNeXt-S-MoEE (Ours)	6.83 (-21.6%)	34.00 (-32.3%)	34.54	82.68
ConvNeXt-B	15.38	88.59	–	83.80
ConvNeXt-B-MoEE (Ours)	10.85 (-29.5%)	49.16 (-44.5%)	34.81	83.23

Table 5. Performance and parameter comparison for additional architectures, evaluated using: **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) generalizes to other architectures by achieving competitive accuracy with significant reductions in MACs and parameters, especially in the Swin-S, Swin-B and ConvNeXt-B models.

Model	MACs (G)	Parameters (M)	Acc. Retention (%)	Top-1 Acc. (%)
DeiT-T	1.30	5.72	–	80.50
DeiT-T-MoEE (Ours)	1.03 (-20.8%)	4.56 (-20.3%)	10.24	79.32
DeiT-S	4.61	22.05	–	85.33
DeiT-S-MoEE (Ours)	3.55 (-23.0%)	16.70 (-24.3%)	55.85	84.94
DeiT-B	17.58	86.57	–	88.20
DeiT-B-MoEE (Ours)	11.73 (-33.3%)	56.83 (-34.4%)	15.76	86.71

Table 6. Performance and parameter comparison of the DeiT-MoEE models, evaluated on CIFAR-100 using: **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) generalizes to smaller datasets and thus data efficient settings.

11. Generalizability to other Datasets and Architectures

The results in Table 5 and Table 6 confirm that our method generalizes well across both hierarchical and convolution-inspired transformer architectures (Swin-Transformer [17] and ConvNeXt Models [18]), as well as to smaller datasets like CIFAR-100 [12]. Despite structural differences, all variants benefit from fewer MACs, reduced parameter counts and competitive final accuracies.

In particular, the larger base variants benefit most from expert extraction. For Swin-B-MoEE and ConvNeXt-B-MoEE, we reduce the MACs by 33% and 30%, and the parameter count by 40% and 45% respectively, while maintaining over 98% of the original accuracy after fine-tuning. These results confirm that our method becomes increasingly effective with model size, offering substantial savings in both compute and memory without compromising final performance.

Notably, we observe that the parameter reduction is more pronounced in models like Swin and ConvNeXt compared to DeiT. Which we primarily attribute to their hierarchical structure. Since experts predominantly form in the deeper layers, where token throughput is lower but embedding dimensions are larger, this results in higher parameter savings but lower relative MAC reductions.

Furthermore, ConvNeXt models show notably lower accuracy retention before fine-tuning. We attribute this to their use of convolutions, which are inherently more parameter-efficient than fully connected layers. Thus, removing even a small set of neurons can have a stronger relative effect on expressiveness. Nevertheless, the method still recovers strong final accuracy after fine-tuning, showing that even architectures with different inductive biases remain compatible with our approach.

Model	MACs (G)	Parameters (M)	Acc. Retention (%)	Top-1 Acc. (%)
DeiT-S	4.61	22.05	–	79.70
HDBSCAN (Ours)	**3.19 (-30.6%)**	16.54 (-25.0%)	**67.60**	**78.11**
DBSCAN	3.52 (-23.5%)	16.53 (-25.1%)	66.47	77.70
OPTICS	3.60 (-21.7%)	16.92 (-23.3%)	67.06	77.79
K-Means	3.54 (-23.1%)	16.60 (-24.7%)	47.34	76.68
BIRCH	3.51 (-23.7%)	**16.45 (-25.4%)**	48.33	77.01

Table 7. Performance and parameter comparison for different clustering algorithms, evaluated using: **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 (using HDBSCAN) provides the best performance among the density-based clustering algorithms and significantly outperforms the partition-based algorithms.

Method	Magnitude-Based			Variance-Based		
	MACs Reduction (%)	Acc. Retention (%)	Top-1 Acc. (%)	MACs Reduction (%)	Acc. Retention (%)	Top-1 Acc. (%)
Cosine	28.17	64.45	77.15	30.63	67.60	78.20
Euclidean	27.22	63.90	77.35	30.48	66.15	78.10
Hashing	22.34	40.92	76.75	24.98	46.53	77.11

Table 8. Comparison of routing methods (**Cosine Similarity**, **Euclidean Distance** and **Hashing**) using two extraction strategies (**Magnitude-Based** and **Variance-Based**) on DeiT-S and evaluated on ImageNet-1k. The table evaluates three metrics: **Accuracy Retention (%)**: retained accuracy after expert extraction before fine-tuning; **Top-1 Accuracy (%)**: final accuracy after fine-tuning; and **MACs Reduction (%)**: relative computational savings compared to the baseline model. Variance-Based extraction consistently outperforms Magnitude-Based extraction for both routing methods, achieving higher Accuracy Retention, Top-1 Accuracy, and MACs Reduction. Between the routing methods there is no significant difference, as both Cosine Similarity and Euclidean Distance show comparable results across all metrics. These results highlight the robustness of the extraction strategies regardless of the used routing method, with Variance-Based approaches generally providing better performance.

12. Effect of Clustering Algorithm

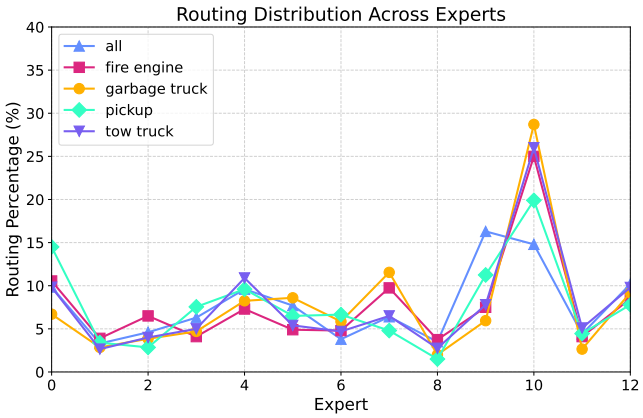
To evaluate impact of different clustering algorithms on expert extraction, we conduct experiments on the DeiT-S model using the same training settings as described in Section 4 and evaluate the resulting model on ImageNet-1k.

We compare our default method, HDBSCAN, against other density-based methods (DBSCAN, OPTICS) as well as partition-based alternatives (K-Means, BIRCH). For the partition-based algorithms, which require the number of clusters as a hyperparameter, we guide their selection using the number of experts extracted by HDBSCAN per layer (see Table 9). As shown in Table 7, density-based methods consistently outperform partition-based ones in the Top-1 Accuracies, particularly in accuracy retention. Among the density-based methods, HDBSCAN achieves the best trade-off across all metrics, leading to the highest accuracy retention and final accuracy, while also yielding the greatest reduction in MACs. These results justify our choice of HDBSCAN for all main experiments in this work.

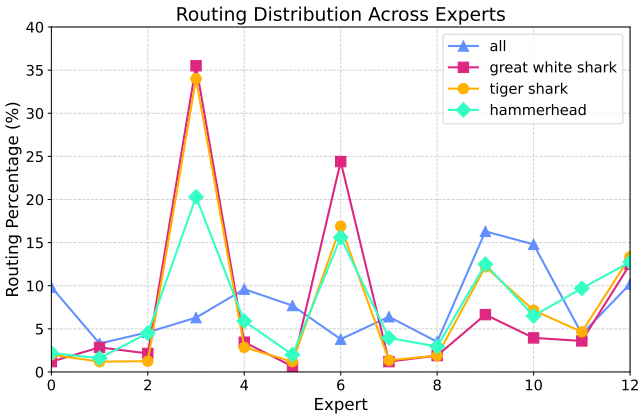
13. Effect of Extraction and Routing Method

In order to evaluate the effects of the extraction and routing methods, we perform another ablation study. For the extraction strategy, we consider two methods for selecting the hidden neurons of each expert. In the Magnitude-Based approach, neurons are selected based on their mean activation magnitude, prioritizing neurons with higher average activations. The Variance-Based approach instead prioritizes neurons with higher within-cluster variance, capturing diversity in activation patterns.

Additionally, we compare three routing approaches: Cosine Similarity, where new input tokens are routed to the expert with the highest cosine similarity; Euclidean Distance, which selects the expert with the smallest Euclidean distance to the cluster mean; and Hashing, an orthogonal method that uses a hash function for routing. Unlike Cosine and Euclidean routing, which assume a Gaussian cluster shape and rely on descriptive statistics of the inputs, hashing requires that the same function be used during extraction and inference.



(a) 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). These classes show similar routing patterns among themselves and align with the distribution of all classes.



(b) Token routing distributions at different layers for visually similar shark-like classes (great white shark, tiger shark, and hammerhead) compared to the distribution across all ImageNet-1k classes (all). These classes show similar routing patterns among themselves but differ significantly from the distribution of all classes.

Figure 8. Comparison of token routing distributions for visually similar classes in layer 11. (a) presents routing patterns for truck-like classes, which closely resemble the overall routing distribution of all classes. (b) shows routing patterns for shark-like classes, which form similar patterns that diverge from the overall distribution. These results suggest that truck-like classes result in more generic tokens shared across multiple classes.

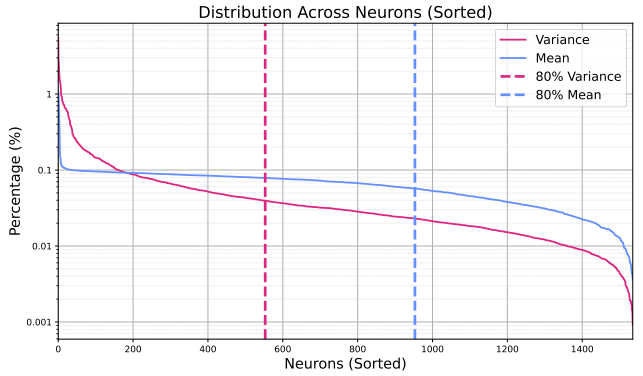


Figure 9. Distribution of variance and mean across neurons in layer 11. The y-axis uses a logarithmic scale to emphasize the differences between neurons. Vertical dashed lines indicate the points where the cumulative sum of the variance and mean reaches 80%. These results demonstrate that the mean is more uniformly distributed, requiring a larger number of neurons to achieve the same cumulative percentage compared to the variance.

Notably, Magnitude-Based prioritization results in a smaller reduction in MACs for a given extraction percentage compared to Variance-Based prioritization. As shown in Figure 9, this is because of differences in the distributions of variance and mean across neurons. The mean activations have a more uniform distribution, necessitating a larger number of neurons to account for a specific cumulative percentage.

In contrast, the variance is concentrated among fewer neurons, allowing Variance-Based extraction to cover a larger cumulative percentage with fewer neurons. This concentration results in greater MACs reductions for the same extraction percentage. To enable a fair comparison between the two methods at similar MACs reductions, the extraction percentage is adjusted from 80% for Variance-Based extraction to 60% for Magnitude-Based extraction.

Table 8, shows that Variance-Based extraction consistently outperforms Magnitude-Based extraction across all metrics: Accuracy Retention, Top-1 Accuracy, and MACs Reduction. On the other hand, the choice of routing method had no significant impact on the overall results, with Cosine Similarity and Euclidean Distance having comparable accuracy and computational savings in both extraction scenarios. As the Cosine Similarity can be computed without normalization, we chose the Cosine Similarity as the default routing method in all experiments, due to the better computational efficiency.

14. Insights into Routing Distributions

We further analyse the routing distribution in Figure 8, which shows the token routing distributions for selected visually similar classes in layer 11, compared to the distribution across all 1,000 ImageNet-1k classes. Specifically, Figure 8a presents the routing distributions for truck-like classes (fire engine, garbage truck, pickup, and tow truck), while Figure 8b shows the distributions for shark-like classes (great white shark, tiger shark, and hammerhead).

Model	Layer 0	1	2	3	4	5	6	7	8	9	10	11
DeiT-T-MoEE	–	–	–	–	–	–	–	4.67	7.67	8.33	8.00	10.33
DeiT-S-MoEE	–	–	–	–	–	–	2.67	6.00	7.67	8.67	8.67	10.67
DeiT-B-MoEE	–	–	–	–	–	–	7.67	8.00	9.00	8.33	9.00	11.00

Table 9. Mean number of experts extracted across layers for different DeiT-MoEE models. Specialization into distinct activation clusters emerges progressively in the deeper layers.

Model	Stage 0	Stage 1	Stage 2	Stage 3
Swin-T-MoEE	–	–	2.00	5.50
Swin-S-MoEE	–	–	2.10	9.00
Swin-B-MoEE	–	–	2.90	10.50
ConvNeXt-T-MoEE	–	–	2.00	4.50
ConvNeXt-S-MoEE	–	–	2.50	4.50
ConvNeXt-B-MoEE	–	–	2.15	6.00

Table 10. Mean number of experts activated across model stages for Swin and ConvNeXt MoEE variants. Expert specialization emerges predominantly in the later stages, especially in deeper variants.

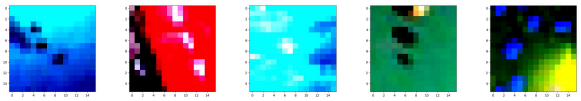
The routing patterns of visually similar classes show a high degree of overlap, suggesting similar classes are processed through similar expert selections. However, the routing distributions between truck-like and shark-like classes are noticeably different from each other, indicating that visual similarity influences token routing, even after the positional encodings have modified the token representations.

Furthermore, the distribution for the truck-like classes closely aligns with the distribution of all 1,000 ImageNet-1k classes, unlike the shark-like classes. These show cohesive routing patterns among themselves but differ significantly from the overall distribution. Since the routing is based on a Cosine Similarity to fixed mean tokens, this similarity in routing distributions means that tokens in truck-like classes are similar to tokens across all classes. This again confirms that the token redundancy allows the model to generalize routing patterns effectively (see Section 9).

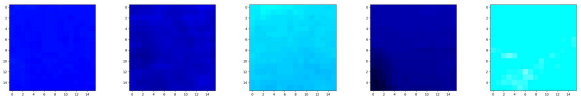
15. Insights into Expert Formations

Table 9 and 10 show the mean number of experts extracted in each layer for different models. Notably, the earlier layers do not exhibit any formed clusters, reflecting the more general feature representations in shallower layers. Deeper layers, on the other hand, display more experts reflecting the progressively stronger specialization, which aligns with our analysis that more discriminative or class-relevant features emerge later in the network.

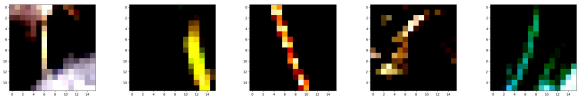
An important aspect is that each experts can vary in size and may partially overlap in their selected neurons. This partial overlap means that a strict one-to-one partition of the MLP into disjoint experts does not necessarily occur.



(a) Selected sample patches routed to expert 3 in layer 11.



(b) Selected sample patches routed to expert 6 in layer 11.



(c) Selected sample patches routed to expert 10 in layer 11.

Figure 10. Figure of selected sample patches from DeiT-B

Nonetheless, the union of all extracted experts in a given layer remains smaller than the original MLP of that layer, as evidenced by the reduced parameter counts in our results.

To provide a qualitative analysis, we additionally visualize the image patches corresponding to tokens routed to specific experts. We first save the token routings for the selected experts at layer 11 for random batches of validation data. We then identify the corresponding image patches from the original input for each token. Note that these patches have undergone multiple processing steps in MHSA and MLP layers, so the tokens may no longer resemble the original patches. For a better insight, the resulting patches are then clustered, and samples within one cluster are selected for visualization, providing a visual understanding of the type of patches routed to each expert (see Figure 10).