Sparse Competition during Training For the Emergence of Specialized Modules

BMVC 2026 Submission # 1292

Abstract

Modularity in deep neural networks has been proposed as a means of improving both interpretability and training by promoting disentangled representations and reducing redundancy. In this work, we study the emergence of modular structure through competition dynamics between groups of neurons during training. We introduce a method that (i) maintains near-baseline accuracy, (ii) induces usage-based modularity by sparsely routing inputs to neuron groups, and (iii) encourages specialization of these modules, such that their activations are correlated with input classes. We evaluate the proposed approach on ImageNet-100 and CIFAR-100 and show that with it, specialized modules emerge without module-level supervision. These modules capture a meaningful high-level structure in the data, with individual modules responding to semantic categories (e.g., dogs or vehicles). We also study the emergence of a hierarchical partition of sub-tasks depending on the number of modules. Our results suggest that competitive dynamics can serve as a simple mechanism for inducing functional modularity in standard architectures.

1 Introduction

Modular computation has long been viewed as a promising direction towards more interpretable and reusable neural representations. Early work on Mixture-of-Experts and sparse routing [10, 16] proposed architectural solutions to enforce module boundaries. However, architectural separation does not guarantee that modules acquire meaningful functional roles. A neural network can be made of modules while still representing task-relevant information in a redundant or entangled way.

Recent mechanistic interpretability work [1, 5] reveals that networks decompose computation into identifiable sub-circuits. However, these studies reveal the existence of an internal structure without asking whether it is well-organized. Do networks learn reusable functions or redundant and cluttered heuristics that solve the task ? The field's focus on benchmark performance obscures this distinction. A neural network can achieve high accuracy through reusable localized computations or opaque distributed heuristics.

Subsequent work suggests that internal representations depend not only on the architecture and the task, but also on the training process. For example, open-ended evolutionary systems, such as Picbreeder can produce networks with reusable internal regularities and meaningful responses to perturbations [9, 14, 15]. This motivates our question: can standard network training be biased toward more factored internal organization?

A natural approach is to enforce modularity directly. The recent work on clusterability [8] proposes adding a differentiable penalty to the loss that encourages weights to cluster

within pre-defined module groups. This achieves structural modularity because the weight matrices become concentrated within the boundaries of the module. However, this comes at a cost. Enforcing clusterability damages the accuracy of the task and fails to induce semantic specialization, caused by the modules not developing meaningful task-related roles. This results in a tradeoff between accepting reduced accuracy for modular weights and abandoning the structural approach altogether. Therefore, can we achieve both modularity and strong task performance?

In this paper, we propose that modularity should emerge from learned routing, not from weight structure. We introduce a routing-based approach that uses module dropout with dual entropy objectives. The first objective concentrates each input on a few modules, while the second encourages different inputs to use different modules. Modules compete for input, which drives specialization without explicit architectural constraints. On ImageNet-100, sparse top-1 routing remains within roughly one percentage point of the vanilla baseline while inducing strong semantic specialization. Classes with the same coarse label are routed to the same modules approximately $3\times$ more often than expected by chance. This specialization also persists hierarchically as increasing module count refines partitions while preserving semantic coherence.

Moreover, we observe that dense training shows specialized modularity under the same competitive incentives. Yet, sparse training forces modules to be individually deployable, whereas dense training creates co-adapted specialists that rely on each other. This distinction separates the emergence of modular structure from the emergence of useful sparse modularity. The representation structure can be induced just by the training dynamics. Furthermore, we extend our approach in several directions. First, we propose learning module widths through a shared capacity budget to create competition over resources that drives specialization. Second, we apply module routing to multiple layers simultaneously and reveal how semantic hierarchy emerges across depths.

Our main contributions break down as follows:

- We introduce Sparse Competitive Routing, a label-free routing objective that induces usage-based modularity through per-sample sharpness and population-level balance.
- We show that the resulting modules acquire specialization, with module assignments carrying substantially more class information than without modules or with clustered baselines.
- We demonstrate that sparse competitive training is mechanistically necessary for sparse deployability: dense specialization is not enough.
- We analyze the semantic organization of learned modules across module counts and layers, showing that learned partitions refine hierarchically while preserving semantic coherence.

2 Related Works

Most of the recent efforts on modularity tackle either interpretability or efficiency. The interpretability goal consists of decomposing a model into weakly interacting components, making its internal functioning easier to interpret and analyze, while efficiency is achieved through conditional computation. Early work on modular representations in deep neural

networks [18] attempts to produce a simplified description of networks through the use of similar connection patterns. Subsequent work formalizes this approach through the notion of clusterability [7], where some parts of the network are thought of as clusters because its parts are densely connected, while they are sparsely connected to the rest. Following this, [8] measures clusterability in a differentiable manner, making it possible to train a neural network on this as an objective, through a loss function that encourages non-interacting clusters. They show that this produces modular components in fully connected layers, but find that these clusters do not reliably become more task-specialized than those in non-modular baselines. Our work addresses this gap. Instead of enforcing modularity as a static property of the weight graph, we study whether sparse competitive routing can induce usage-based modules that are preferentially selected by different visual classes. Moreover, recent work [2] in controlled neural systems shows that structural modularity alone is insufficient for functional specialization and that specialization might depend on resource constraints and architecture. Our work argues that an architecture-agnostic with usage-based modularity is sufficient to attain specialization in a neural network. The gradient routing approach [3] attempts to control the location of specific capabilities or information on the network through the manipulation of backpropagation. They use data-dependent masks to gradients to choose which part of the network will learn from which data. This allows for the localization of language-model capabilities for later ablation without damaging the accuracy of the model on the rest of its uses. Their work relies on the user-supplied information of *"what goes where"*, which is different from our work, in which we do not assign specific classes or semantic groups to modules. Instead, we study whether sparse competitive routing, complemented with an anti-collapse objective, without module-level supervision, is sufficient for semantic module usage to emerge.

As models scale to billions or more recently to trillions of parameters, Mixture-of-Experts (MoEs) [6, 11, 16] have emerged as a reliable approach to scaling in the architectural design space. MoEs consist of a routing function that activates only a subset of parameters for each input, allowing practitioners to increase model capacity without increasing the computation. In computer vision, V-MoE shows that sparse expert routing can be competitive with dense Vision Transformers while reducing inference cost [13]. Similarly, Soft-MoE shows that it is possible to design fully differentiable MoE architectures to avoid hard token routing [12]. MoEs also emerged in efficient vision architectures such as CNNs [19], showing that experts can be coupled with tightly vision-prior based architectures. Our goal differs from most MoE work. We investigate the relevance of adding experts as an architectural prior that constrains the network's exploration of the space and its expressivity. We show that softer steering through the loss function can lead to their emergence. This unsupervised emergence of modules is also interesting for interpretability reasons, as it allows us to study the learning dynamics and how they evolve compared to models with modularity enforced modularity through the architecture.

Finally, our work is related to mechanistic interpretability methods that seek to design or identify circuits from deep neural networks as human-understandable units [1, 4]. Although we do not introduce new circuit-discovery algorithms, we argue that our contribution on training for emergent modular specialization paves the way for constructing circuits in an unsupervised yet human-understandable manner.

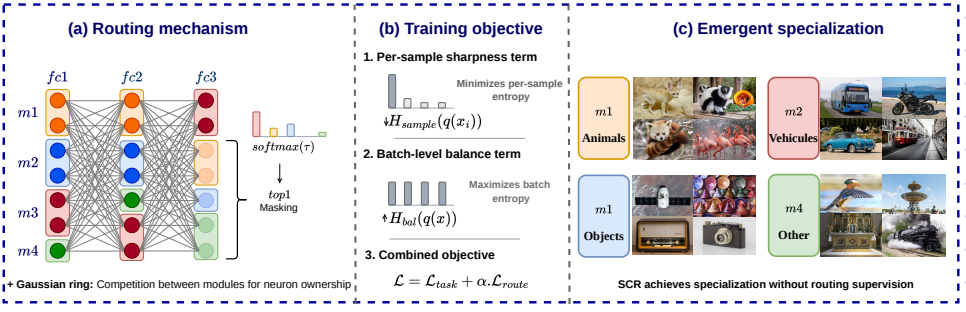

Figure 1: Overview of Sparse Competitive Modules. (a) A routed layer is partitioned into modules. For each input, module activation energies are converted into routing probabilities, and a hard top-1 mask keeps only the selected module active. (b) The routing objective combines a per-sample sharpness term, which encourages confident module selection, with a batch-level balance term, which prevents collapse by encouraging all modules to be used. (c) Without module-level or semantic supervision, the resulting routing assignments become class-informative, where visually related classes tend to share dominant modules.

3 Methodology

This section introduces *Sparse Competitive Routing*, for which an overview is presented on Figure 1, a training procedure that induces usage-based modularity through the loss function of neural networks. Unlike structural modularity objectives, our approach does not directly form weakly interacting clusters. Instead, it steers the single-module selection per input while preventing collapse. Only one module is activated per routed layer, and the objective encourages the network to balance usage across modules. Importantly, the routing decision is unsupervised by class labels, which are used only in the standard classification loss.

3.1 Modularizing a Layer

Let $h_\ell(x) \in \mathbb{R}^{d_\ell}$ be the activation of a routed layer $\ell \in \mathcal{R}$. We divide its coordinates into K_ℓ modules $\mathcal{G}_{\ell,1}, \ldots, \mathcal{G}_{\ell,K_\ell}$. In the simplest case, these are equal-width contiguous blocks of neurons. This partition defines where modules are, but does not assign any semantic role.

For each module k, let $m_{\ell,k} \in \{0,1\}^{d_\ell}$ be the corresponding binary mask. The activation supported by the module k is

$$h_{\ell,k}(x) = m_{\ell,k} \odot h_\ell(x) \tag{1}$$

and the full activation decomposes as

$$h_\ell(x) = \sum_{k=1}^{K_\ell} h_{\ell,k}(x). \tag{2}$$

We route inputs using the activation energy of each module,

$$E_{\ell,k}(x) = \left\| m_{\ell,k} \odot h_\ell(x) \right\|_2 + \varepsilon, \tag{3}$$

where ε is a small numerical constant stability. A module is selected on the basis of the magnitude of its response to the input. Unlike structural modularity objectives, which impose

constraints directly on the weight matrices, our method leaves the weights unconstrained and induces modularity through input-dependent usage patterns. We later relax the equal-width block assumption by using a Gaussian-ring, described at the end of the next subsection.

3.2 Sparse Competitive Routing Objective

The goal of our method is to induce structured module usage without providing any class-related information. For each training sample (x_i, y_i), the label y_i is used only through the standard classification loss. The routing objective does not observe y_i, it operates only on the activation patterns produced by the network itself. Hence if visual classes become associated with different modules, this structure emerges from competition between modules driven by input-dependent usage, rather than from any externally specified partition of the data.

Our objective combines two competing pressures. The first encourages each input to make a confident routing decision, localizing its computation to a small subset of modules. The second encourages balanced usage across inputs, ensuring that the network retains the capacity of all modules. This balance is necessary because sparse routing has a natural collapse mode that causes all inputs to be routed to the same module. Given k modules of equal size, this would effectively reduce the usable capacity of the routed layer to roughly $1/k$ of its original capacity. Collapse can be self-reinforcing: a slightly more responsive module receives more routed examples, hence more task gradients, which increases its dominance. Similarly, if one module initially learns a broadly useful classification strategy, task loss alone provides a short-term incentive to route many inputs through that module. As a result, the routing objective must simultaneously produce sharp per-sample decisions while maintaining diversity in module usage at the population level.

For a routed layer ℓ, let $E_{\ell,k}(x_i)$ denote the activation energy of the module k for input x_i, as defined above. We convert the energy into a differentiable routing distribution as follows:

$$q_{\ell,k}(x_i) = \frac{\exp(E_{\ell,k}(x_i)/\tau)}{\sum_{k'=1}^{K_\ell} \exp(E_{\ell,k'}(x_i)/\tau)}, \tag{4}$$

where $\tau > 0$ is a temperature parameter. Smaller values of τ make the distribution closer to a winner-take-all decision, while larger values provide smoother gradients.

We minimize the entropy of the per-sample routing distribution to induce the first pressure, encouraging each input sample to select a distinct module:

$$\mathcal{L}^{(\ell)}_{\text{sample}} = \frac{1}{B}\sum_{i=1}^{B} H(q_\ell(x_i)) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{k=1}^{K_\ell} q_{\ell,k}(x_i)\log q_{\ell,k}(x_i), \tag{5}$$

where B denotes the batch size.

We then add the second pressure that maximizes the entropy of the average usage distribution. This is a batch-level balance term under the assumption that, for a sufficiently large batch of inputs, module usage should be evenly distributed:

$$\mathcal{L}^{(\ell)}_{\text{bal}} = -H(\bar{q}_\ell) = \sum_{k=1}^{K_\ell} \bar{q}_{\ell,k}\log \bar{q}_{\ell,k}. \tag{6}$$

This objective can be interpreted as encouraging high mutual information between inputs and module assignments, without using labels. The sharpness term reduces the conditional entropy of module assignment for each input, while the balance term raises the marginal

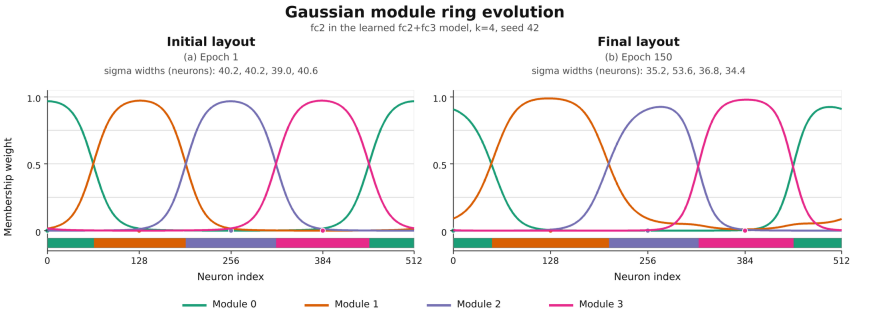

Figure 2: Illustration of the working of the Gaussian ring, with a run where the layer $fc2$ with 512 neurons is modularized into 4 modules. Left is the first epoch, and right is the last epoch. We can see the network learned to get a dominant module 1, without total collapse.

entropy of module usage across inputs. Together, these two terms encourage confident, yet diverse, routing decisions. The task loss then determines which assignments are useful for prediction, allowing semantic specialization to emerge indirectly during training.

The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha \mathcal{L}_{\text{route}}, \quad \mathcal{L}_{\text{route}} = \mathcal{L}_{\text{sample}} - \mathcal{L}_{\text{bal}} \tag{7}$$

where $\mathcal{L}_{\text{task}}$ denotes the standard supervised loss function and α controls the strength of the routing objective $\mathcal{L}_{\text{route}}$ during training.

Gaussian ring module masks for capacity competition. The default SCM formulation partitions a layer into K contiguous blocks of equal width. This imposes a fixed-capacity prior in which each module receives exactly the same number of neurons. To relax this assumption, we replace hard block masks with soft Gaussian masks on a circular neuron axis. We decide to make the axis circular to avoid border effects: with 4 modules, the modules 0 and 3 are only in contact with one module, which would disadvantage them, where we want a fair competition between modules to avoid collapse. The ring simply allows each module to be connected to two modules.

Neuron i and module m are assigned positions $p_i = i/d_\ell$ and $c_m = m/K$ on the unit ring, and module memberships are computed as

$$G_{i,m} = \text{softmax}_m \left(-\frac{d_{\text{ring}}(p_i, c_m)^2}{2\sigma_m^2 T_{\text{gauss}}} \right), \tag{8}$$

where d_{ring} denotes the periodic distance and σ_m is a learnable module width. The widths are constrained by a fixed total budget, so increasing the capacity of one module reduces the capacity available to others. The routing energies are then computed as in the previous formulation, replacing the hard module mask $m_{\ell,k}$ with the soft membership vector $G_{:,k}$.

This modification does not change the SCM objective or the sparse top-1 routing rule. Instead, it replaces fixed modules of equal width with soft capacity-adaptive modules. As shown in Table 2, the Gaussian ring does not substantially affect accuracy, while it consistently increases class-module mutual information. We therefore use SCM with Gaussian ring masks in the remaining experiments.

3.3 Training and Inference

The neural network is trained end-to-end using standard gradient-based training, with a hard top-1 mask applied during the forward, forcing the outputs of all the non-selected modules to be zero. Consequently, gradients obtained through the classification loss update only the selected module at that layer. The auxiliary routing loss is computed from the differentiable energy profile prior to the hard routing decision, allowing the model to reshape its activation energies during training to modify the preferred module assignment for a given input.

At inference time, we apply the same sparse top-1 routing rule as during training:

$$z_\ell(x) = \arg\max_k E_{\ell,k}(x), \tag{9}$$

$$\tilde{h}_\ell(x) = m_{\ell,z_\ell(x)} \odot h_\ell(x). \tag{10}$$

This consistency is important, as dense training relying on usage-based competition can yield class-structured energy profiles without producing modules that operate independently from the rest of the layer. We therefore distinguish sparse competitive training from a dense control setting, as shown in Figure 4-A, in which all modules remain active during training and sparse top-1 routing is applied only at inference time. Task gradients flow only through the selected module, while the routing loss is computed from the pre-argmax distribution.

3.4 Usage-Based Modularity Metrics

Following prior work on clusterability-based modularity [8], we aim to quantify how routing induces an informative and balanced distribution of module usage. Ideally, routing should reflect specialized computation, such that different modules are selectively activated for different semantic subsets of data. In a classification task, subtasks are classifications of superclasses, hence a specialized module should be detected by a module only being activated when inputs corresponding to classes of a given superclass are present.

Then, we quantify class–module specialization for each routed layer ℓ by computing the mutual information between the class variable C and the selected module index Z_ℓ:

$$MI(C;Z_\ell) = \sum_{c,k} p(c,k) \log \frac{p(c,k)}{p(c)p(k)}. \tag{11}$$

4 Experiments

4.1 Setup

Backbone. For comparability with the clusterability baseline, we use the same global architecture as [8]: a small CNN stem followed by a bias-free fully connected head. After validating our method on CIFAR-10, we scaled the network to train from scratch on CIFAR-100 and ImageNet-100. Our `scaled_cnn` maps $x \in \mathbb{R}^{3\times224\times224}$ through three Conv-BN-ReLU-MaxPool blocks with channels $32, 64, 128$, then adaptive-average-pools to 4×4, yielding a 2048-dimensional feature vector. The head is $2048 \to 512 \to 512 \to 512 \to 100$, with ReLU activations after the first three fully connected layers with no bias terms. We denote the three hidden fully connected layers by $fc1$, $fc2$, and $fc3$, where $fc1$ maps the convolutional feature vector to the first 512-dimensional hidden representation, while $fc2$

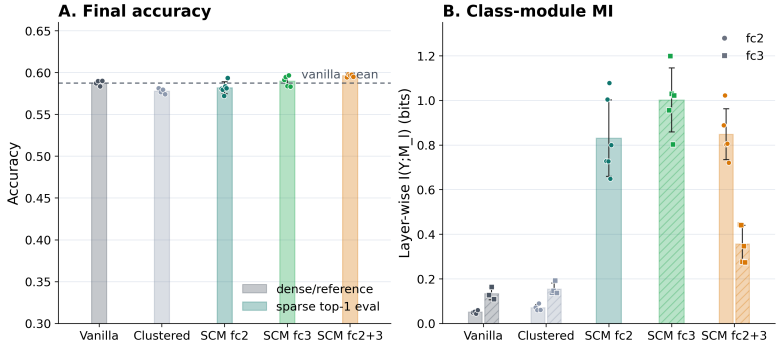

Figure 3: Measures of accuracies and MI of the same CNN model after a training on ImageNet-100 CMC. A: The accuracies of our method, clustered baseline from [8] and without modularization. Here, the accuracy of our method corresponds to the top-1 accuracy while other methods are evaluated on dense accuracies. B: the MI between the favored module and the input class.

and $fc3$ are 512-to-512 hidden transformations. Routing is applied only to the selected layers among $fc1$, $fc2$, and $fc3$, and the final 100-way classifier remains dense.

Dataset and training. Our main experiments are conducted on ImageNet-100, a subset of 100 classes from the 1000-class ImageNet dataset. Instead of using a random subset, which could significantly affect the results, we use the standard CMC subset [17], which is designed to provide a more balanced selection of classes across diverse categories. All methods are evaluated with the same backbone and training protocol within each comparison. We report sweeps over routed layers ($fc2$, $fc3$, and $fc2+fc3$) and over the number of modules $K \in \{4,8,16\}$. To demonstrate that our results are not specific to ImageNet-100, we additionally report results on CIFAR-100.

Training images are processed with ImageNet normalization. Validation images are resized to 256, center-cropped to 224, and normalized in the same manner. Models are trained for 150 epochs with Adam, a batch size of 128, a learning rate of 10^{-3}. Unless stated otherwise, we use a constant $\alpha = 0.1$.

We compare Sparse Competitive Routing against three baselines. The vanilla baseline is trained with the standard classification loss and no modular objective. The clustered baseline follows the clusterability approach of [8], which encourages weights to concentrate within predefined module groups. Unless otherwise stated, SCM refers to sparse competitive routing with Gaussian ring module masks. Finally, to show that with our method we in fact learn in a similar way as Mixture-of-Experts, but without a learned router, we add a learned router baseline. This baseline uses the same backbone, routed layers, module masks, and number of modules as SCM, but replaces the energy-based routing distribution with a learned linear router $q_\ell(x) = \mathrm{softmax}(W_\ell h_\ell(x)/\tau)$. During training and evaluation, the selected module is the top-1 router prediction, and all non-selected modules are masked out, as in SCM. The router is trained jointly with the classification objective and standard load-balancing auxiliary losses to avoid collapse. Routing probabilities are computed from module activation energies using temperature $\tau = 1.0$.

Table 1: Main comparison on ImageNet-100 CMC and CIFAR-100 for the routed `fc2` layer. Accuracies are reported in percentage points and are evaluated in each method's intended inference mode: dense for Vanilla and Clustered, and sparse top-1 for SCM. $\mathrm{MI}(Y;M_{\texttt{fc2}})$ is computed from the selected or highest-energy module at $fc2$.

Dataset	Method	Acc. (%)	$\mathrm{MI}(Y;M_{\texttt{fc2}})$
ImageNet-100 CMC	Vanilla	58.77 ± 0.38	0.036 ± 0.006
ImageNet-100 CMC	Clustered baseline	57.67 ± 0.26	0.049 ± 0.012
ImageNet-100 CMC	SCM	57.84 ± 0.59	0.649 ± 0.128
CIFAR-100	Vanilla	47.76 ± 2.10	0.114 ± 0.049
CIFAR-100	Clustered baseline	49.00 ± 1.10	0.139 ± 0.037
CIFAR-100	SCM	48.94 ± 2.40	0.493 ± 0.194

Evaluation. For our method, training and inference use the same sparse top-1 routing rule: only the selected module remains active at a routed layer. We also include a dense-control variant, where the routing objective is used during training but all modules remain active in the forward pass. This separates semantic specialization of the routing scores from sparse deployability of the selected modules.

We report classification accuracy and class-module mutual information $MI(Y;M_\ell)$, where M_ℓ is the selected module at layer ℓ. To detect collapse and measure the evenness of usage, we also report the effective number of used modules:

$$N_{\text{eff}} = \exp(H(M_\ell)). \tag{12}$$

For semantic analysis, we measure whether classes from the same coarse ImageNet group are assigned to the same module more often than expected by chance. In the module-count sweep, we additionally measure whether higher-K partitions refine lower-K partitions using pair retention and weighted child purity.

Finally, to test robustness to the choice of classes, we repeat the main ImageNet-100 experiment on two random class subsets. We report mean and standard deviation across matched seeds when available.

4.2 Results

Our method maintains near-baseline accuracy with module specialization Table 1 reports the main ImageNet-100 CMC comparison for routing at $fc2$ with $k = 4$ modules. These results are also shown on Figure 3. Out method obtains $57.84 \pm 0.59\%$, matching the clustered baseline and remaining within roughly one percentage point of the vanilla model. Thus, sparse competitive routing does not require accuracy degradation. We can observe the same trends on CIFAR-100.

The main difference is not accuracy but the information carried by module usage. Vanilla and clustered models have low class-module mutual information at $fc2$, with $MI(Y;M_{fc2}) = 0.036 \pm 0.006$ and 0.049 ± 0.012, respectively. In contrast, our method reaches 0.649 ± 0.128, an increase of approximately $18\times$ over vanilla and $13\times$ over the clustered baseline. This shows that the selected module is no longer an arbitrary subdivision of the layer: it becomes a class-informative routing variable.

To test whether the CMC split contains an unusually favorable semantic structure, we repeat the experiment on two random ImageNet-100 label subsets. The results in the table

of the section C of the Appendix show the same qualitative pattern. Thus, the specialization effect is not specific to the CMC label subset: across different class selections, sparse competitive routing preserves baseline-level accuracy while making module assignments substantially more predictive of class identity.

Increasing the number of modules gives finer specialization, Gaussian ring too. Table 2 isolates the effect of the Gaussian ring masks. Across $K \in \{4, 8, 16\}$, adding the ring changes top-1 accuracy only marginally, but increases $MI(Y; M_{\text{fc2}})$ consistently. The ring should therefore be interpreted as a specialization-enhancing capacity parameterization, rather than as an accuracy-improving component.

The Gaussian ring does not substantially change the accuracy: across $K \in \{4, 8, 16\}$, the difference relative to SCM without the ring is small. However, it consistently increases $MI(Y; M_{\text{fc2}})$, from 0.409 to 0.638 at $K = 4$, from 0.730 to 0.891 at $K = 8$, and from 1.150 to 1.297 at $K = 16$. Hence, we use SCM with Gaussian-ring competition in the remaining experiments, not for the accuracy, but as a mechanism that strengthens modular specialization.

The learned MoE baseline on Table 2 also shows that we can reach the accuracy and specialization of a learned router without explicitly using one. At $K = 4$, it reaches similar accuracy and mutual information to SCM, but as the number of modules increases, the learned MoE baseline loses in accuracy : 2.3 and 5.8 points in accuracy compared to our method while having a lower MI. The module usage shows also a worse balance overall. Our method is hence able to reproduce (or even surpass in more high pressure settings) a learned router, only with new training dynamics.

Sparse training is necessary for sparse deployability Figure 4-A and B separates two notions of modularity. A dense-control model can learn class-structured routing scores while all modules remain active in the forward pass. However, because every training example can rely on all modules, the selected module is not forced to be individually sufficient. Applying top-1 routing only at evaluation therefore produces modules that are semantically recognizable but not reliably deployable. In contrast, SCM trains and evaluates with the same hard top-1 mask. The selected module receives the task gradient for that example, while non-selected modules do not. This forces each routed module to support the computation assigned to it. The ablation therefore shows that semantic specialization of routing scores is not enough: sparse competitive training is required to obtain modules that remain functional under sparse inference.

Modules recover coarse semantic structure: the emergence of a hierarchy of classes For each class, we define its dominant module as the module selected most often on validation examples, and compare the resulting partition with coarse ImageNet groups, using these groups only for evaluation. Classes from the same coarse group are routed to the same module roughly $3\times$ more often than chance, suggesting that the modules capture semantic structure rather than arbitrary class identities. We further test whether this structure forms a hierarchy by training independent $fc2$ models with $K \in \{4, 8, 16\}$ modules. SCM shows clear coarse-to-fine consistency: pair retention is 28.32% for $4 \to 8$ and 27.15% for $8 \to 16$, compared with random baselines of $13.85 \pm 0.86\%$ and $6.40 \pm 0.89\%$. Weighted child purity is also high, at 69.0% and 72.0%. Thus, increasing the number of modules refines broad semantic specialists into finer ones, rather than producing unrelated partitions.

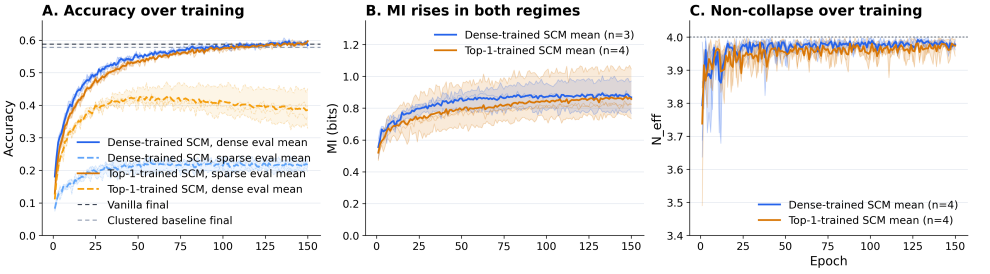

Figure 4: Evolution of metrics during trainings on ImageNet-100 CMC. A. The top-1 accuracies of our method (SCM) compared to no clustering (Vanilla) and the method from [7] (Clustered); in dotted lines are the opposite evaluations compared to training: the dense training is on sparse eval, and the sparse training is on dense eval. Both regimes fair better when evaluated like training, but sparse training is better on dense than dense on sparse eval. B. The evolution of MI between our method trained in sparse and dense training. C. The evolution of the effective number of modules at the end of training. The trainings on our method converge on $N_{eff} = 4$ meaning all modules are used evenly.

4.3 More analysis

Causal specialization tests Finally, we test whether class-module associations are causally relevant rather than merely correlational. For each class, we identify its dominant module, ablate that module at test time, and compare the accuracy drop on classes owned by the ablated module with the drop on all other classes. Table 3 shows that SCM has positive causal gaps at both routed layers: 10.99 percentage points at $fc2$ and 18.15 at fc3. Importantly, SCM maintains uniform ownership across the four modules.

The clustered baseline also exhibits causal gaps, but its ownership is highly imbalanced, especially at $fc3$, where the effective number of class-owning modules drops to 1.71. Thus, SCM does not merely increase class-module mutual information; it induces a balanced decomposition whose modules are functionally relevant for their assigned classes. We report these results in Appendix B.

Routing deeper layers improves the accuracy–specialization tradeoff. We next ask where sparse competitive routing should be applied. Table 4 reports a layer-placement sweep on ImageNet-100 CMC using the same `scaled_cnn`, and $K = 4$ modules. Routing at $fc1$ already induces important class-module specialization, but it comes with a 3.04% accuracy drop relative to the matched dense baseline. Hence, early fully connected features are still too general: forcing a sparse module choice at this stage removes useful shared computation.

Routing later layers gives a better tradeoff. At $fc2$, sparse top-1 routing slightly exceeds the matched baseline accuracy, 58.06% versus 57.92%, while increasing class-module mutual information from 0.0476 to 0.5049. Routing only $fc3$ gives the best single-layer specialization, with $MI(Y;M) = 0.6628$, and also improves accuracy by 0.32 percentage points. Routing both $fc2$ and $fc3$ gives the best overall result, with 59.42% top-1 accuracy and high MI at both layers. This shows that modularity works best on later layers, where features are already semantically organized enough for module selection to be class-informative.

Table 2: Module-count sweep for the routed $fc2$ layer on ImageNet-100 CMC. Acc. denotes top-1 validation accuracy. Mutual information is reported in nats. For K modules, the nominal maximum possible class–module mutual information is $\ln K$ nats, or 1.386, 2.079, and 2.773 nats for $K = 4, 8, 16$ respectively. We therefore also report the normalized quantity $MI(Y;M_{\text{fc2}})/\ln K$. Clustered models are evaluated densely, whereas SCM and Learned-MoE use sparse top-1 routing at inference.

K	Method	Acc. (%)	$MI(Y;M_{\text{fc2}})$	$MI/\ln K$	N_{eff}
4	Clustered baseline	57.92 ± 0.22	0.048 ± 0.015	0.035 ± 0.011	3.83 ± 0.02
4	Learned-MoE	57.60 ± 0.70	0.50 ± 0.10	0.361 ± 0.072	3.25 ± 0.01
4	SCM w/o Gaussian ring	57.53 ± 0.81	0.409 ± 0.091	0.295 ± 0.066	3.96 ± 0.02
4	SCM + Gaussian ring	57.92 ± 0.23	0.638 ± 0.111	0.460 ± 0.080	3.97 ± 0.01
8	Clustered baseline	58.74 ± 0.44	0.062 ± 0.012	0.030 ± 0.006	7.83 ± 0.03
8	Learned-MoE	55.50 ± 1.00	0.78 ± 0.12	0.375 ± 0.058	6.45 ± 0.03
8	SCM w/o Gaussian ring	57.61 ± 0.46	0.730 ± 0.039	0.351 ± 0.019	7.85 ± 0.05
8	SCM + Gaussian ring	57.79 ± 0.22	0.891 ± 0.050	0.428 ± 0.024	7.83 ± 0.08
16	Clustered baseline	57.56 ± 0.87	0.113 ± 0.019	0.041 ± 0.007	15.09 ± 0.08
16	Learned-MoE	51.00 ± 1.50	0.90 ± 0.15	0.325 ± 0.054	14.34 ± 0.09
16	SCM w/o Gaussian ring	55.77 ± 0.73	1.150 ± 0.030	0.415 ± 0.011	15.31 ± 0.12
16	SCM + Gaussian ring	56.81 ± 0.38	1.297 ± 0.026	0.468 ± 0.009	15.52 ± 0.10

Higher-resolution module partitions refine lower-resolution ones We ask next whether increasing k on completely unrelated runs produces unrelated partitions or a hierarchy of refinements that is compatible across seeds and module numbers: if modules meaningfully partition data on consistent granularity. Table 5 reports pair retention and weighted child purity for independently trained $K = 4, 8, 16$ runs. We compare two independently trained partitions using two metrics. *Pair retention* is the fraction of class pairs assigned to the same module at lower k that remain coassigned at higher k. We compare it to a random baseline obtained by permuting the higher-k assignments while preserving module sizes. *Weighted child purity* measures whether each higher-k module mostly comes from a single lower-k parent, weighted by the number of classes in the child module. Table 5 shows that SCM obtains a clear refinement structure. SCM has pair retention well above the random baseline for both 4→8 and 8→16 transitions. These fare better than the clustered baseline, being closer or even below their random baseline, and lower than our method. SCM also obtains higher weighted child purity than the clustered baseline. The alluvial visualization in Figure 5 illustrates the same pattern: SCM modules at higher resolution behave like semantic subdivisions of lower-resolution modules, whereas clustered partitions are less stable across independently trained resolutions.

Transfer to ResNet-18. To check that the effect is not specific to our `scaled_cnn` architecture, we repeat the main $fc2$ routing experiment with a ResNet-18 backbone on ImageNet-100. SCM sparse top-1 routing obtains 69.1% accuracy, compared with 70.1% for the vanilla model and 69.0% for the clustered baseline, while increasing class-module MI from 0.04–0.06 to 0.55. This suggests that sparse competitive routing transfers to a standard vision backbone. For computational reasons, these experiments remain limited in scope, and the remaining diagnostic ablations were performed with our smaller CNN only.

Table 3: Causal specialization diagnostic on ImageNet-100 with four modules at $fc2$ and $fc3$. Classes are assigned to their dominant module; we then ablate each module and measure the accuracy drop on its owned classes and on all other classes. Drops are percentage points. Full per-module results are given in Appendix B.

Model	Layer	Owned drop	Other drop	Gap	Ownership N_{eff}
SCM	$fc2$	22.38	11.39	10.99	3.99
SCM	$fc3$	27.32	9.17	18.15	3.99
Clustered	$fc2$	37.40	15.50	21.90	2.48
Clustered	$fc3$	24.48	10.59	13.89	1.71
Vanilla	$fc2$	32.22	21.64	10.58	3.39
Vanilla	$fc3$	19.90	13.13	6.77	3.32

Table 4: Layer-placement sweep on ImageNet-100 CMC. All rows use `scaled_cnn`, seed 42, 150 epochs, and $K = 4$ modules. Accuracy values are percentages.

Routed layers	Baseline acc.	Ours dense acc.	Ours top-1 acc.	Baseline MI	Ours MI
$fc1$	58.80	46.98	57.16	0.0179	0.4059
$fc2$	57.92	53.54	58.06	0.0476	0.5049
$fc3$	58.80	55.90	59.12	0.0763	0.6628
$fc2 + fc3$	58.12	36.24	59.42	0.049 / 0.095	0.615 / 0.307

5 Conclusion

In this work we proposed a method that yields modules in neural networks through competition. The competition is mainly carried through usage-based competition with Sparse Competitive Routing allowing for a modularization that produces meaningfully specialized modules. Our second, complementary way to further increase specialization of modules is through capacity-based competition with Gaussian ring ownership of neurons, where the width of modules is learned. These two ways, as we showed allow for modules that conserve the accuracy of a non-modularized network, while meaningfully gathering the data into groups at varying granularity, adapting to its number of modules. This work demonstrates a new way to modularize networks: through the loss only, via information theoretic metrics. This approach further manifests that internal organization does not need to be monitored, but given the right incentives, it can appear spontaneously.

6 Limitations and future works

6.1 Limited Experiment Sizes

Our experiments are intentionally conducted in a controlled setting, which makes module usage, specialization, and ablation easy to measure. However, this also limits the scope of our conclusions. Our experiments do not yet establish that the same behavior scales unchanged to larger architectures or datasets.

A second limitation is that only a small number of layers are modularized. Most experiments route one layer, and our multi-layer results are limited to 1 to 3 layers. This is enough

Table 5: Hierarchy metrics for the $fc2$ module-count sweep on ImageNet-100 CMC.

Method	Transition	Pair retention	Random baseline	Weighted child purity
Ours	$4 \to 8$	28.32	13.85 ± 0.86	69.00
Ours	$8 \to 16$	27.15	6.40 ± 0.89	72.00
Clustered	$4 \to 8$	21.70	23.72 ± 1.55	66.00
Clustered	$8 \to 16$	11.33	9.62 ± 0.83	48.00

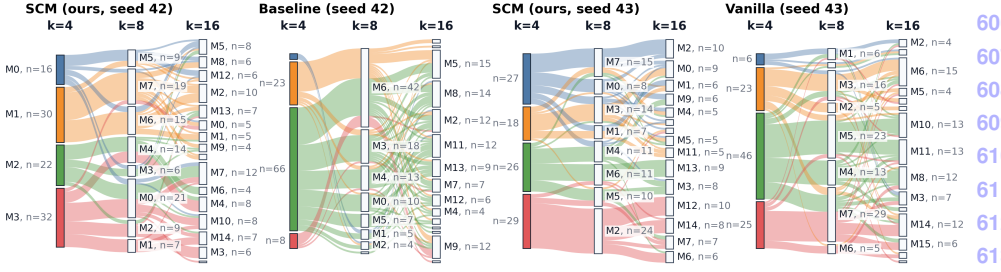

Figure 5: Alluvial visualization of class assignments across independently trained $fc2$ runs with $K = 4, 8, 16$ modules. Each block is a learned module and each flow tracks ImageNet-100 classes across module-count resolutions. We compare the cleaner refinement structure obtained by our method against the two alternative flows shown in the figure. To rule out this structure being a seed artifact, we also show two independent seeds of our method, both of which exhibit similar coarse-to-fine class flows.

to show that routing become class-informative and remain usable under sparse inference, but not enough to claim full-network modular computation or complete circuit recovery.

6.2 Future Work

The first direction is to test whether sparse competitive routing scales beyond the controlled vision setting we studied. Our experiments use relatively small networks on ImageNet-100, which makes module usage and class-level specialization easy to measure. A natural next step is to apply the same objective to larger backbones, larger vision datasets, and models with richer internal structure, language models, and vision-language models.

A second direction is to modularize more layers simultaneously. In this work, most of our analysis focuses on one or two layers. However, the main interpretability promise of modular training is to obtain whole *circuits*: connected chains of modules that support particular computations across depth. Our preliminary multi-layer experiment in the Appendix A, where $fc2$ is routed into four modules and $fc3$ into eight modules, suggests that this may occur naturally. This suggests that sparse competitive routing may already bias the network toward structured module-to-module pathways.

References

[1] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert

Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[2] Gabriel Béna and Dan F. M. Goodman. Dynamics of specialization in neural modules under resource constraints. *Nature Communications*, 16(1):187, 2025. doi: https://doi.org/10.1038/s41467-024-55188-9. URL https://www.nature.com/articles/s41467-024-55188-9.

[3] Alex Cloud, Jacob Goldman-Wetzler, Evžen Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks. *arXiv preprint arXiv:2410.04332*, 2024.

[4] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[5] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[6] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv*, abs/2101.03961, 2021. URL https://api.semanticscholar.org/CorpusID:231573431.

[7] Daniel Filan, Stephen Casper, Shlomi Hod, Cody Wild, Andrew Critch, and Stuart Russell. Clusterability in neural networks, 2021. URL https://arxiv.org/abs/2103.03386.

[8] Satvik Golechha, Maheep Chaudhary, Joan Velja, Alessandro Abate, and Nandi Schoots. Studying cross-cluster modularity in neural networks, 2025. URL https://arxiv.org/abs/2502.02470.

[9] Joost Huizinga, Kenneth O Stanley, and Jeff Clune. The emergence of canalization and evolvability in an open-ended, interactive evolutionary system. *Artificial life*, 24(3): 157–181, 2018.

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

[11] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

[12] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts, 2023.

[13] 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 *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235417196.

[14] Jimmy Secretan, Nicholas Beato, David B D Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O Stanley. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1759–1768, 2008.

[15] Jimmy Secretan, Nicholas Beato, David B D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T Folsom-Kovarik, and Kenneth O Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary computation*, 19(3):373–403, 2011.

[16] 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. *ArXiv*, abs/1701.06538, 2017. URL https://api.semanticscholar.org/CorpusID:12462234.

[17] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 776–794, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58621-8.

[18] Chihiro Watanabe, Kaoru Hiramatsu, and Kunio Kashino. Modular representation of layered neural networks. *Neural Networks*, 97, 03 2017. doi: 10.1016/j.neunet.2017.09.017.

[19] Yihua Zhang, Ruisi Cai, Tianlong Chen, Guanhua Zhang, Huan Zhang, Pin-Yu Chen, Shiyu Chang, Zhangyang Wang, and Sijia Liu. Robust mixture-of-expert training for convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 90–101, October 2023.