Composable Sparse Subnetworks via Maximum-Entropy Principle

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Neural networks implicitly learn class-specific functional modules. In this work, we ask: Can such modules be isolated and recombined? We introduce a method for training sparse networks that accurately classify only a designated subset of classes while remaining deliberately uncertain on all others, functioning as class-specific subnetworks. A novel KL-divergence-based loss, combined with an iterative magnitude pruning procedure, encourages confident predictions when the true class belongs to the assigned set, and uniform outputs otherwise. Across multiple datasets (MNIST, Fashion MNIST, tabular data) and architectures (shallow and deep MLPs, CNNs), we show that these subnetworks achieve high accuracy on their target classes with minimal leakage to others. When combined via weight summation, these specialized subnetworks act as functional modules of a composite model that often recovers generalist performance. We experimentally confirm that the resulting modules are mode-connected, which justifies summing their weights. Our approach offers a new pathway toward building modular, composable deep networks with interpretable functional structure.

1 Introduction

Modern neural networks (NNs) implicitly develop internal subgraphs of neurons and connections tuned to respond to specific classes. These structures, sometimes referred to as *circuits* [Olah et al., 2020, O'Neill and Bui, 2024], emerge during training but are difficult to isolate, reuse, or compose. Representations for different classes are often entangled, resulting in shared neurons or features—a phenomenon known as *superposition* [Mu and Andreas, 2020, Saphra and Wiegreffe, 2024]—which makes clean modularity elusive.

This lack of class-level modularity limits our ability to understand, edit, or compose networks. If we could reliably extract a *functional module*—a subnetwork that specializes in recognizing one class (or a small subset) while ignoring others—we could enable new forms of continual learning, unlearning, and compositional reasoning [De Lange et al., 2022]. Crucially, such modules must not only function in isolation, but also be designed to compose smoothly, without fine-tuning or alignment [Ilharco et al., 2023, Hazimeh et al., 2024]. In this work, we want to answer the following question:

> Can we train sparse, class-specialized subnetworks that remain ignorant outside their domain, and compose into accurate, generalist models?

We answer affirmatively, proposing a methodology that leverage two key concepts: the Maximum-Entropy (MaxEnt) principle and sparse training. The MaxEnt principle [Jaynes, 1957] advocates choosing the most uninformative distribution consistent with known constraints. Originally applied in statistical mechanics, it has become central to inference theory [Kuić, 2016, De Martino and De Martino, 2018].

We claim that MaxEnt can guide functional isolation: modules are trained to make confident predictions only for their class, and uniform predictions otherwise. This differs from standard entropy regularization used in calibration or selective prediction [Marczak et al., 2024].

On the other hand, sparsity plays a crucial role in this context. Sparse models tend to exhibit less overlap in their parameters or activations, reducing interference when modules are merged. We demonstrate its practical benefits by leveraging the Lottery Ticket Hypothesis (LTH). In general, LTH claims that sparse subnetworks can match dense models if trained with the right initialization [Frankle and Carbin, 2018, Zhou et al., 2019, Liu et al., 2024]. In our scenario, we show that sparse subnetworks can achieve better performance than dense ones while enhancing modular compatibility.

In light of this, we propose a framework based on two main components: a novel **Max-Entropy loss** (ME) coupled with an **Iterative Magnitude Pruning** (IMP) strategy [Frankle and Carbin, 2018], and a model merging step to achieve generalist models. The goal of the MaxEnt loss is to enforce confident predictions on a subnetwork's rewarded class set and uniform predictions otherwise, while the IMP removes spurious capacity and reinforces specialization through sparsity [Burkholz, 2022, Girish et al., 2021]. We call the resulting components *subnetwork modules*: sparse networks that specialize in a class or class subset, and can be combined to form more generalist models.

To the best of our knowledge, this is the first work that trains a NN by isolation and merging. Our results suggest a new approach to NN construction: rather than learning entangled solutions end-to-end, we can build systems from independently trained, entropy-regularized modules. To summarize, our contribution is threefold: *(i)* we introduce a novel MaxEnt loss able to train specialized modules; *(ii)* we propose a new training pipeline to achieve better isolation and model merging to get generalist models; *(iii)* we validate our findings through extensive experiments on several architectures and datasets.

2 Related Work

In this section, due to lack of space, we put only the most relevant related work. An extended version of related works involving model merging, pruning methods, modular training and MaxEntropy principle is available in Appendix B.

Modular Neural Networks and Interpretability. Several works have explored modular neural architectures that decompose computation across tasks [Kirsch et al., 2018, Han et al., 2021, Salem et al., 2023]. Notably, Kirsch et al. [2018] introduced end-to-end learning of both modules and their composition via a controller. They rely on subsequent modules for learning broader tasks, while using Expectation-Maximisation (EM) for learning a specific module. While sharing a similar motivation to our work, we build *class-specific functional modules*, trained to focus on a single class while producing high-entropy on different classes. This specific goal is hindered by their structure of subsequent modules, that are chosen by a controller and reused across tasks, making it difficult to isolate class-specific functionality. Additionally, our approach relies on model merging, rather than leveraging subsequent modules.

Interestingly, Malakarjun Patil et al. [2023] highlight the importance of pruning methods in unveiling the hierarchical structure of NNs. In particular, they propose an iterative approach for hierarchy detection, where the hierarchy is the underlying hierarchy of sub-functions in a specific task, delivering interesting network analysis. Our work partially shares the goal, but focuses more on a complete training pipeline and on submodules isolation without seeking a hierarchical structure, rather than network analysis about submodules.

Interpretability Our work inevitably aligns with the goal of producing interpretable networks. Among the methods that tackle this challenge, we can distinguish between post hoc and interpretable-by-design methods. While post hoc interpretability methods [Ribeiro et al., 2016, Lundberg and Lee, 2017] are constrained to work with trained models, our methodology is applied *during training*, yielding units with clear semantics and compositional behavior. In particular, our approach aligns with mechanistic interpretability methods [Saphra and Wiegreffe, 2024, Olah et al., 2020], where subnetworks (or circuits) are studied as functional entities. Interestingly, Liu et al. [2023] propose a brain-inspired modular training procedure, relying on network embedding into geometric spaces, and neurons swapping. Instead of leveraging pruning methodologies, they rely mainly on regularization.

However, they mainly use synthetic datasets and never scale to CNN, while we show that our technique is applicable also to a broader range of tasks.

Finally, our work also complements neuron-level analyses like Mu and Andreas [2020], which seek meaningful internal representations but suffer from polysemanticity. Instead, we encourage modularity at the subnetwork level. This is conceptually related to Marchetti et al. [2024], where symmetry constraints shape functional structure—our modules may reflect class-wise symmetry components in such a framework.

3 Learning Composable Class-Specific Subnetwork Modules

We propose a method for constructing composable subnetwork modules, each specialized in classifying a specific subset of classes while remaining unresponsive to others. These modules are trained to produce peaked predictions on their target classes and uniform predictions otherwise, in line with the MaxEnt principle. Our proposed procedure consists of two key elements that we are going to discuss: a custom loss function and an iterative magnitude pruning schedule. Our final goal is to merge trained modules efficiently, resulting in a more powerful model, built by simple merging. Figure 1, provide an high level overview of our method and its components.

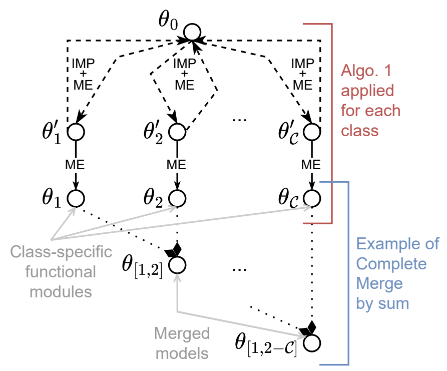


Figure 1: Overview of our method. Starting from common initialization θ_0 we find, via Iterative Magnitude Pruning (IMP) exploiting our Max-Entropy (ME) loss (see Algorithm 1), for each class in $\mathcal{C}$ a specialized subnetwork. Modules can be summed via weight summation.

3.1 Max-Entropy Loss

Let $\mathcal{C}$ be the full set of classes and $\mathcal{R} \subseteq \mathcal{C}$ the set of *rewarded classes* for a given subnetwork module. For a training sample (x, y), where $y \in \mathcal{C}$, we define a *target distribution* $\tilde{y} \in \mathbb{R}^{|\mathcal{C}|}$ as:

$$\tilde{y}_i = \begin{cases} \delta_{i=y} & \text{if } y \in \mathcal{R} \\ \frac{1}{|\mathcal{C}|} & \text{otherwise} \end{cases} \tag{1}$$

For example, if $\mathcal{C} = \{0, 1, 2\}$ and $\mathcal{R} = 0$ we will use $\tilde{y} = (1, 0, 0)$ for class 0, $\tilde{y} = (0.33, 0.33, 0.33)$ for classes 1 and 2. Let $\hat{y} = \mathrm{softmax}(f_\theta(x))$ be the predicted class probability distribution produced by the model. The *MaxEnt loss* is then defined as the Kullback-Leibler (KL) divergence between the target distribution $\tilde{y}$ and the predicted distribution $\hat{y}$:

$$\mathcal{L}_{\mathrm{ME}}(x, y) = \mathrm{KL}(\tilde{y} \parallel \hat{y}) = \sum_{i=1}^{|\mathcal{C}|} \tilde{y}_i \log \left(\frac{\tilde{y}_i}{\hat{y}_i} \right) \tag{2}$$

This formulation encourages the model to output *peaked predictions* (low entropy) for samples in rewarded classes and *uniform predictions* (high entropy) for non-rewarded ones. In doing so, it promotes *functional isolation*, ensuring that each subnetwork module specializes only in its intended class subset and remains maximally uncertain elsewhere.

Unlike one-vs-all formulations, our approach enforces *neuron-permutation invariance* on non-rewarded samples—a crucial property for composability. In a one-vs-all setup, training a module for class 0 would implicitly push other output neurons (e.g., neuron 1) to represent "not class 0." However, if another module uses neuron 1 to represent "class 1," summing the two leads to a semantic collision: the same neuron would encode both "class 1" and "not class 0", which includes many other classes.

3.2 Iterative Magnitude Pruning

To enhance specialization and remove spurious capacity, we apply **Iterative Magnitude Pruning** (IMP) [Frankle and Carbin, 2018] as in Algorithm 1. At each iteration, we train with the MaxEnt loss, prune a fixed percentage of low-magnitude weights, and reinitialize the remaining ones. After several rounds, a final training phase is applied to the sparse subnetwork. This process encourages class-specific specialization and improves uncertainty calibration on non-rewarded classes.

Algorithm 1 Iterative Magnitude Pruning with Max-Entropy Loss

```
Require: Initial weights $\theta_0$, training data $\mathcal{D}$, rewarded classes $\mathcal{R}$, number of pruning iterations $N$,
    pruning percentage $P$, epochs per iteration $E$
1: $\theta \leftarrow \theta_0$
2: for $i = 1$ to $N$ do
3:     Train model $f_\theta$ on $\mathcal{D}$ using max-entropy loss with rewarded classes $\mathcal{R}$ for $E$ epochs
4:     Prune $K\%$ of weights in $\theta$ with smallest absolute value, with $K = 1 - (1 - P)^{\frac{1}{N}}$
5:     Reset remaining weights in $\theta$ to their initial values from $\theta_0$
6: end for
7: Train the final pruned subnetwork on $\mathcal{D}$ using max-entropy loss with rewarded classes $\mathcal{R}$ for $E$
    epochs
```

3.3 Model Merging via Weight Summation

To compose a generalist model, we merge submodules by summing their weights: $\theta_{\text{merged}} = \sum_i \theta_i$. This operation is enabled by our design: submodules are trained to specialize on disjoint subsets of classes and to behave identically—via uniform predictions—on all others, minimizing interference.

We consider both pairwise merges ($\theta_1 + \theta_2$) and complete merge ($\sum_i \theta_i$), and evaluate compositionality using the same metrics used for individual modules. To understand when and why summation preserves performance, we analyze the loss landscape through the lens of mode connectivity. Crucially, unlike typical mode connectivity studies—where θ_1 and θ_2 solve the *same task*—our submodules are specialized for *different sets of classes*. The merged model is intended to solve the union task, and is evaluated using the *MaxEnt loss* over the combined rewarded classes. This distinction makes it essential to assess whether the merged weights lie in a low-loss region for the composite task.

Following Frankle et al. [2020] and Lubana et al. [2023], we say that θ_1 and θ_2 are *mode connected* along a path $\gamma(t)$ if:

$$\forall t \in [0,1], \quad \mathcal{L}(f_{\gamma(t)}(\mathcal{D})) \leq (1-t)\mathcal{L}(f_{\theta_1}(\mathcal{D})) + t\mathcal{L}(f_{\theta_2}(\mathcal{D})) + \epsilon \tag{3}$$

where ϵ is a small margin, set to 2% of the first term on the r.h.s. following Frankle et al. [2020], and $\mathcal{L}$ is our MaxEnt loss evaluated on a dataset $\mathcal{D}$ with labels restricted to $\mathcal{R}_1 \cup \mathcal{R}_2$, the union of the rewarded classes for the two modules.

To test this, we define the following *piecewise-linear* path, designed such that $\theta_1 + \theta_2$ appears as an intermediate point:

$$\gamma_{\theta_1 \to \theta_2}(t) = \begin{cases} \theta_1 + 2t \cdot \theta_2 & \text{if } t \leq 0.5 \\ 2(1-t) \cdot \theta_1 + \theta_2 & \text{if } t > 0.5 \end{cases} \tag{4}$$

This path satisfies: $\gamma(0) = \theta_1$, $\gamma(1) = \theta_2$, and $\gamma(0.5) = \theta_1 + \theta_2$. We evaluate the loss along $\gamma(t)$ and interpret low-barrier profiles as evidence that the modules are composable by construction.

4 Experiments

We empirically evaluate our method on multiple image and tabular datasets, aiming to answer the following questions: (i) Can MaxEnt training enforce meaningful class-specific specialization? (ii) How well do merged submodules recover generalist performance? (iii) Does iterative pruning

Model	IMP	MNIST		Fashion MNIST		HAR		Yeast	
		Entropy	Rewarded Acc	Entropy	Rewarded Acc	Entropy	Rewarded Acc	Entropy	Rewarded Acc
Shallow MLP	No	2.296 (0.003)	0.998 (0.002)	2.296 (0.003)	0.998 (0.002)	1.762 (0.017)	0.997 (0.007)	1.298 (0.062)	0.995 (0.009)
	Yes	2.293 (0.004)	0.999 (0.001)	2.293 (0.004)	0.999 (0.001)	1.757 (0.023)	0.996 (0.008)	1.297 (0.059)	0.998 (0.006)
Deep MLP	No	2.298 (0.002)	0.997 (0.003)	2.285 (0.013)	0.995 (0.004)	1.772 (0.014)	0.992 (0.013)	1.302 (0.064)	0.996 (0.009)
	Yes	2.300 (0.001)	0.998 (0.002)	2.291 (0.008)	0.991 (0.007)	1.762 (0.023)	0.999 (0.005)	1.302 (0.056)	1.000 (0.000)
CNN	No	2.302 (0.000)	0.998 (0.004)	2.302 (0.000)	0.996 (0.004)	-	-	-	-
	Yes	2.302 (0.000)	0.994 (0.005)	2.302 (0.000)	0.992 (0.005)	-	-	-	-

Table 1: Single submodule behaviour when using MaxEnt Loss with and without IMP

improve the quality of submodules and composability? (iv) Does our training procedure induce mode connectivity between independently trained modules and does it scale with the number of modules?

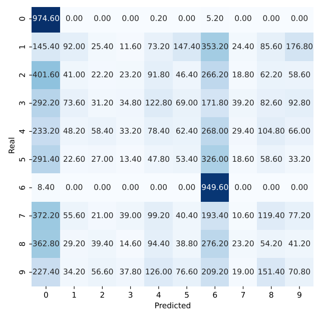
4.1 Training and Evaluation Protocol

We apply our MaxEnt loss and iterative pruning independently to each class or subset $\mathcal{R} \subseteq \mathcal{C}$. Each submodule is trained as in Algorithm 1, and evaluated on: (i) **rewarded accuracy**, i.e., classification accuracy on samples from $\mathcal{R}$; (ii) **non-rewarded entropy**, i.e., the average predictive entropy on inputs from $\mathcal{C} \setminus \mathcal{R}$; and (iii) the **confusion matrix**, used qualitatively to assess specialization and leakage.

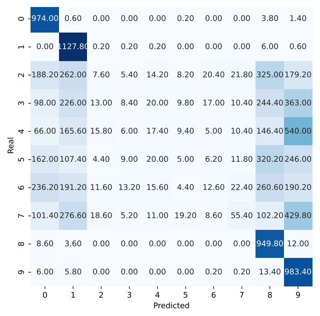
In particular, it is worth mentioning that in our training procedure we simulate a scenario in which only the classes in $\mathcal{R}$ are labeled, but we still have access to the other samples.

To evaluate composability, we used $|\mathcal{R}| \in \{1, 2, 5\}$ to test pairwise merging. For a specific cardinality, the model is trained to recognize solely a selected set of classes, of the given cardinality and sampled among the ones available for that specific dataset. We sample 10 pairs of class sets $\{(\mathcal{R}_{2k}, \mathcal{R}_{2k+1})\}_{k=0}^{9}$, with $\mathcal{R}_{2k} \bigcap \mathcal{R}_{2k+1} = \emptyset$ and $\mathcal{R}_{2k}, \mathcal{R}_{2k+1}$ spanning in the set of combinations for the dataset's class set. For each pair, we perform 5 independent random seeds and merge a pair into a single model. For example, with $|\mathcal{R}| = 1$, the procedure consists of: training a NN using our MaxEnt loss rewarding one specific class $\mathcal{R}_0$, then a different NN is trained on a different single class $\mathcal{R}_1$, and then the two models are merged and tested. The resulting model should perform well on the 2 classes. This procedure is repeated for 5 seeds to ensure statistical evidence, and for a total of 10 paired sets. When $|\mathcal{R}| = n > 1$, two NNs are trained to specialize on two different sets of n classes at the same time, and there is no overlap between the sets. Figure 2 depicts an example of resulting confusion matrices for these pairwise-merging experiments.

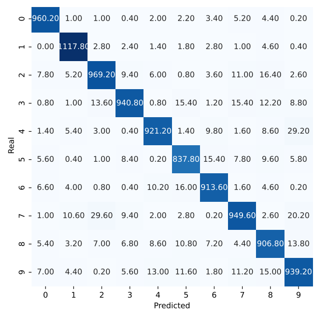
Formal definitions and implementation details are provided in the Appendix.



(a) Classes [0]+[6]



(b) Classes [0,1]+[8,9]



(c) Classes [0–4]+[5–9]

Figure 2: Confusion matrices for three representative pairwise merges of shallow MLP submodules on the MNIST dataset. Even after merging, each rewarded class remains highly separable, and predictions on non-rewarded classes remain nearly uniform.

4.2 Experimental Setup

We evaluate our approach on three model classes: shallow MLPs (1 hidden layer), deeper MLPs (2 hidden layers), and CNNs (LeNet-style). For image data, we use MNIST [LeCun et al., 1998] and Fashion MNIST [Xiao et al., 2017]. These datasets are in line with similar works [Liu et al., 2023, Malakarjun Patil et al., 2023], but to get a broader impact we include two tabular classification datasets from the UCI repository: *Human Activity Recognition* (HAR) [Reyes-Ortiz et al., 2012], and *Yeast* [Dua and Graff, 2017]. We compare our MaxEntropy loss with two baselines, showing that entropy maximization plays a crucial role in isolation. In particular, we compare MaxEnt with:

- Quasi-MaxEnt loss: a loss function that on non-rewarded classes ignores the rewarded classes and produces a uniform distribution solely on non-rewarded classes; Formally referring to (1):

$$\tilde{y}_i = \begin{cases} \delta_{i=y} & \text{if } y \in \mathcal{R} \\ \delta_{i \neq j, j \in \mathcal{R}} \frac{1}{|\mathcal{C} \setminus \mathcal{R}|} & \text{otherwise} \end{cases} \tag{5}$$

- CrossEntropy loss: the standard loss used in standard classification tasks. In this case, we do not modify the labels, which remain standard, but the model is exposed only to the classes in $\mathcal{R}$, rather than all available classes.

Full dataset details and preprocessing steps are reported in Appendix C. Each submodule is trained on a specific subset of classes using the MaxEnt loss, with or without IMP.

4.3 Submodule Behavior with MaxEnt Loss

We begin by evaluating submodules trained solely with the MaxEnt loss, *without pruning*. Each subnetwork is tasked with recognizing one specific class from the original dataset, and to output uniform predictions on all others.

For each of 5 random seeds, we train a separate submodule per class. The results in Table 1 are aggregated across both classes and seeds, providing average values and standard deviations for overall trends.

Table 1 reports the accuracy on rewarded classes and the average entropy on non-rewarded samples. Accuracy is consistently high (close to 100%) across all classes, confirming that submodules correctly specialize on their class. The entropy, meanwhile, approaches the theoretical value of $\log(N_{\text{classes}})$ for a uniform distribution, e.g. $\log(10) \approx 2.30$ for MNIST and Fashion MNIST, suggesting that predictions on excluded classes are nearly uniform.

These results suggest that our proposed submodules can effectively recognize individual classes and exhibit the theoretical behavior they were designed to emulate, across different architectures.

4.4 Pairwise merging

We further evaluate the effectiveness of our method through pairwise merging experiments across different configurations. Table 2 reports the aggregated metrics over seeds and merged pairs. Rewarded accuracy remains high (often $\geq$ 0.90), indicating that each merged module retains its classification ability. The mean entropy on non-rewarded classes stays close to the theoretical maximum of $\log(N_{\text{classes}})$, e.g. it is around 2.28 for MNIST and Fashion MNIST, confirming minimal interference.

Surprisingly, we can see that MaxEnt outperforms the Quasi-MaxEnt in all cases. Besides being such a small change in the learning phase (MaxEnt enforces uniformity over all neurons for non-rewarded inputs, while Quasi-MaxEnt enforces uniformity over all but the rewarded ones), this loss function loses in performance with respect to ours. This surprising result is probably due to the leakage of information on non-rewarded classes. In particular, the QME incentivizes changes in the flow of information even for non-rewarded classes, which violates the MaxEnt principle. In contrast, ME encourages only information from the rewarded classes, improving the specialization and better avoiding spurious activations. Our method also outperforms standard CrossEntropy. This was expected, since the CrossEntropy is intended to train with all the labels at once, while in this case its effectiveness is hindered by the reduced number of classes on which it is optimized.

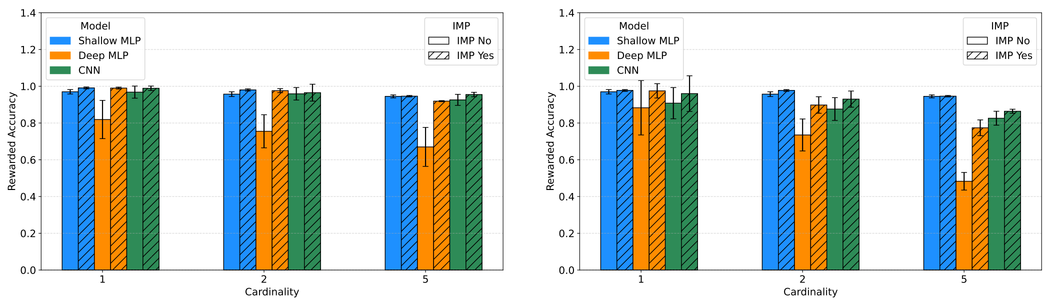


Figure 3: Impact of IMP on pairwise merging for different architectures and cardinalities, considering the MNIST and Fashion MNIST datasets, and using the ME loss.

Table 2: Average entropy and rewarded accuracy across datasets, models, and loss functions. Bold indicates the best result per group according to Welch's test at 95% confidence.

			FMNIST		MNIST		HAR		Yeast	
Model	$\lvert\mathcal{R}\rvert$	Loss	Entropy	R-Acc	Entropy	R-Acc	Entropy	R-Acc	Entropy	R-Acc
MLP	1	XE	0.183 (0.151)	0.877 (0.186)	0.507 (0.150)	0.708 (0.179)	1.255 (0.191)	0.855 (0.197)	1.378 (0.004)	0.631 (0.101)
		QME	2.031 (0.132)	0.908 (0.105)	2.082 (0.005)	0.973 (0.014)	1.196 (0.073)	0.946 (0.046)	1.056 (0.082)	0.799 (0.106)
		ME	2.287 (0.005)	**0.977** (0.005)	2.287 (0.005)	**0.991** (0.005)	1.716 (0.033)	**0.985** (0.016)	1.146 (0.077)	**0.853** (0.127)
	2	XE	0.277 (0.108)	0.786 (0.141)	0.332 (0.080)	0.838 (0.074)	1.167 (0.217)	0.871 (0.055)	–	0.381 (0.036)
		QME	1.824 (0.207)	0.917 (0.049)	1.683 (0.216)	0.959 (0.016)	1.013 (0.223)	0.891 (0.024)	–	0.559 (0.011)
		ME	2.267 (0.007)	**0.977** (0.005)	2.268 (0.007)	**0.980** (0.006)	1.670 (0.045)	**0.935** (0.027)	–	**0.616** (0.010)
	5	XE	–	0.480 (0.075)	–	0.842 (0.027)	–	–	–	–
		QME	–	0.741 (0.026)	–	0.905 (0.023)	–	–	–	–
		ME	–	**0.946** (0.004)	–	**0.946** (0.004)	–	–	–	–
Deep MLP	1	XE	0.215 (0.150)	0.802 (0.205)	0.422 (0.190)	0.689 (0.194)	1.059 (0.313)	0.840 (0.210)	1.386 (0.0003)	0.553 (0.144)
		QME	1.842 (0.329)	0.868 (0.176)	2.061 (0.046)	0.954 (0.045)	1.098 (0.128)	0.819 (0.155)	0.883 (0.118)	0.783 (0.114)
		ME	2.245 (0.072)	**0.975** (0.039)	2.291 (0.031)	**0.990** (0.005)	1.697 (0.050)	**0.988** (0.016)	1.063 (0.109)	**0.859** (0.120)
	2	XE	0.166 (0.083)	0.678 (0.081)	0.208 (0.054)	0.743 (0.111)	1.065 (0.344)	0.824 (0.068)	–	0.382 (0.046)
		QME	1.347 (0.484)	0.863 (0.088)	1.224 (0.352)	0.912 (0.038)	0.665 (0.251)	0.787 (0.093)	–	0.562 (0.023)
		ME	2.197 (0.095)	**0.898** (0.045)	2.275 (0.026)	**0.976** (0.011)	1.607 (0.134)	**0.930** (0.028)	–	**0.589** (0.018)
CNN	1	XE	0.016 (0.026)	0.557 (0.150)	0.041 (0.128)	0.529 (0.080)	–	–	–	–
		QME	0.981 (0.322)	0.880 (0.136)	1.001 (0.377)	0.961 (0.078)	–	–	–	–
		ME	2.297 (0.039)	**0.960** (0.097)	2.286 (0.057)	**0.989** (0.012)	–	–	–	–
	2	XE	0.052 (0.038)	0.604 (0.169)	0.070 (0.042)	0.624 (0.138)	–	–	–	–
		QME	0.705 (0.397)	0.888 (0.071)	0.312 (0.307)	0.928 (0.074)	–	–	–	–
		ME	1.984 (0.304)	**0.930** (0.044)	1.950 (0.456)	**0.965** (0.046)	–	–	–	–

Figure 2 shows qualitative examples of confusion matrices for representative merges with increasing class coverage. In Figure 3, we show the effectiveness of IMP on MNIST and Fashion MNIST on all the architectures and cardinalities, while merging. We can see in particular that IMP improves the performances consistently for a Deep MLP, while benefiting more modestly the Shallow MLP and the CNN.

For what it concerns IMP itself, for each model we have tried different level of pruning. Interestingly, MLP and Deep MLP did not show specific harm with pruning, enabling us to use a percentage of 99%. For the CNN the optimal pruning ratio was 60%. More details about this are provided in the Appendix. Notice that each model is pruned over two iterations, followed by a final training phase.

In Table 1 we also report the performances of single submodules when applying IMP, showing the results aggregated over both seeds and classes. We can see that pruning has little impact on rewarded accuracy and entropy on non-rewarded inputs remains high and close to the theoretical maximum.

These results suggest that pruning removes capacity responsible for encoding unintended structure, thus reinforcing the functional isolation induced by the loss. In addition, the results from Figure 3 suggest that IMP is beneficial for pairwise merging.

4.5 Merging Analysis

In this section, we further evaluate our method in a more challenging scenario and provide further analysis.

Complete Merge We perform a *complete merge* experiment. For each dataset, we randomly generate 10 permutations of the class labels. For each of 5 random seeds, we train one submodule per class using either MaxEnt loss alone or in combination with IMP. We then merge the submodules incrementally, following the order dictated by the permutation: at step k, the merged model contains submodules for the first k classes in the permuted list.

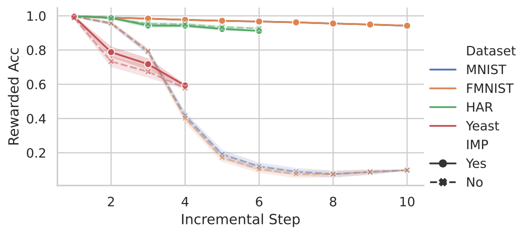
Figure 4 reports the average rewarded accuracy at each merge step, aggregated across permutations and seeds. We can see that submodules trained with IMP (solid lines) consistently outperform or match those trained without pruning (dashed lines).

On Yeast and HAR, performance degrades more rapidly as modules accumulate. Still, pruning helps mitigate this drop in the shallow MLP setting. In the deep MLP on HAR, pruning slightly underperforms—potentially due to the presence of narrower layers, which seem to be more affected by IMP.

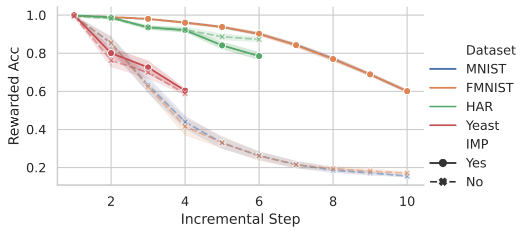
Shallow MLPs show good composability on MNIST and Fashion MNIST, where accuracy stays above 90% up to the full merge. On more complex tabular datasets like HAR and Yeast, performance drops more rapidly, although IMP consistently improves results. Deep MLPs do not outperform shallower ones overall. CNNs also face a performance degradation, but IMP importantly benefits their performance.

The specific comparison between Shallow and Deep MLP suggests that additional depth and narrower layers do not benefit composability, augmenting the possibility of intereference. This suggests that width has a crucial role in enabling composability.

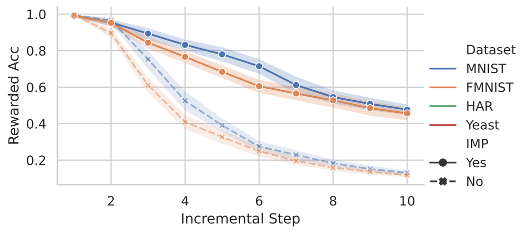
Overall, these results confirm that pruning improves composability across model types and datasets, especially in combination with the MaxEnt loss. For Deep MLP and CNN the performances might be improved by better pruning/merging techniques or width analysis. With these results, we have shown that it is possible to train in our proposed way, but we retain optimized techniques for future work.



(a) Shallow MLP

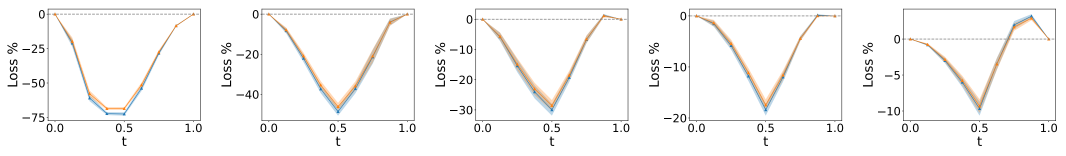


(b) Deep MLP



(c) CNN

Figure 4: Average rewarded accuracy across incremental submodule merging. All models are pruned at 0.99%.



θ_1 = [0] θ_2 = [1]

θ_1 = [0]+[1]+[2] θ_2 = [3]

θ_1 = [0]+. . .+[4] θ_2 = [5]

θ_1 = [0]+. . .+[6] θ_2 = [7]

θ_1 = [0]+. . .+[8] θ_2 = [9]

Figure 5: Values for $\gamma_{\theta_1 \to \theta_2}(t)$ considering multiple steps of the complete merge starting from 0 to 9 for the Fashion MNIST dataset (train and test errors, in blue and orange respectively, mean and std across 5 runs). Loss values at each t are relativized with respect to the corresponding interpolated errors of θ_1 and θ_2 and rescaled as percentages. Values close to or lower than 0 are the desired outcomes.

Mode connectivity To better understand why merging submodules via weight summation is often effective, we study the connectivity of models in weight space. Specifically, we follow the formulation of Lubana et al. [2023], and evaluate whether the merged model lies on a low-loss path between its components, using the MaxEnt loss as criterion.

We analyze complete merges on the Fashion MNIST dataset using shallow MLPs. For each step k in the merge sequence—e.g., combining a submodule obtained from the merge $[0] + \cdots + [k-1]$ and one trained on class k—we evaluate the MaxEnt loss along a piecewise linear path between the previous model and the new submodule, with the midpoint corresponding to their weight summation.

Figure 5 shows the relative loss barrier for each step, computed as the gap between the loss at the midpoint (merged model) and the linear interpolation of the endpoint losses. Values near or below zero indicate mode connectivity.

In the majority of cases, merged models correspond to local minima or lie along smooth, low-loss trajectories, with no significant barriers emerging along the merge path. This supports the hypothesis that the training procedure induces functional composability. However, small barriers (around 2%) begin to emerge in the later steps of the complete merge—when many submodules have already been added—suggesting mild interference.

5 Discussion and Limitations

Our proposed modular design might open new avenues, but also presents limitations and open challenges.

Trade-offs Between Specialization and Scale As the number of merged modules increases, we observe degradation in accuracy and entropy, suggesting a trade-off between the degree of modularity and the robustness of the final model. Understanding and mitigating this scaling limit is a key direction for future work.

Composability and Mode Connectivity For shallow MLPs, merged models lie in regions reachable whithout incurring in loss-barriers, indicating mode connectivity. This property appears to emerge naturally from our training procedure and supports reliable merging. While we rely on simple weight summation, more sophisticated merging techniques—e.g., Git-Rebasin or PLeas [Hazimeh et al., 2024, Zeng et al., 2025]—may further improve composability, particularly in challenging architectures.

Width Sensitivity analysis We do not currently provide a compatibility metric between submodules. However, we empirically observed that modules trained on wider fully connected layers (both for MLPs and LeNet-style networks) tend to compose more reliably than those from narrower networks. We hypothesize that a minimal expressive width may be required to support isolated functional representations. Identifying this threshold, or adapting the procedure to narrower architectures, remains an important open direction—e.g. for applications to transformer blocks.

6 Conclusions

We introduced a principled framework for training sparse neural submodules that specialize in recognizing a designated subset of classes while deliberately remaining ignorant of others. This behavior is encouraged via a novel KL-based MaxEnt loss and an iterative magnitude pruning procedure. Together, they yield compact, high-performing subnetworks that can be composed via simple weight summation—a procedure supported by empirical evidence showing that such modules are mode connected.

Our approach opens new avenues for modular deep learning: from continual learning to model editing, model understanding and multi-source knowledge integration. In the long term, we envision functional submodules becoming a fundamental abstraction for scalable and interpretable neural systems.

Code & configurations. Scripts and models' configurations are provided at the following link: https://anonymous.4open.science/r/MaxEntSubmodulesMechIntWorkshop-9592 .

References

Anisa T Bajkova. The generalization of maximum entropy method for reconstruction of complex functions. *Astronomical and Astrophysical Transactions*, 1(4):313–320, 1992.

Rebekka Burkholz. Convolutional and residual networks provably contain lottery tickets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2414–2433. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/burkholz22a.html`.

Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=SYB4WrJql1n`.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446.

Andrea De Martino and Daniele De Martino. An introduction to the maximum entropy approach and its application to inference problems in biology. *Heliyon*, 4(4), 2018.

Dheeru Dua and Casey Graff. Uci machine learning repository: Yeast data set. `https://archive.ics.uci.edu/ml/datasets/Yeast`, 2017. Accessed: 2024-05-01.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 2020. URL `http://proceedings.mlr.press/v119/frankle20a.html`.

Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 762–771, 2021.

Wei Han, Changgang Zheng, Rui Zhang, Jinxia Guo, Qinli Yang, and Junming Shao. Modular neural network via exploring category hierarchy. *Information Sciences*, 569:496–507, 2021.

Adam Hazimeh, Alessandro Favero, and Pascal Frossard. Task addition and weight disentanglement in closed-vocabulary models. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023.

Sohei Itahara, Takayuki Nishio, Masahiro Morikura, and Koji Yamamoto. Lottery hypothesis based unsupervised pre-training for model compression in federated learning. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–5. IEEE, 2020.

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

Louis Kirsch, Julius Kunze, and David Barber. Modular networks: Learning to decompose neural computation. *Advances in neural information processing systems*, 31, 2018.

Domagoj Kuić. Maximum information entropy principle and the interpretation of probabilities in statistical mechanics- a short review. *The European Physical Journal B*, 89:1–7, 2016.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Bohan Liu, Zijie Zhang, Peixiong He, Zhensen Wang, Yang Xiao, Ruimeng Ye, Yang Zhou, Wei-Shinn Ku, and Bo Hui. A survey of lottery ticket hypothesis. *arXiv preprint arXiv:2403.04861*, 2024.

Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1):41, 2023.

Ekdeep Singh Lubana, Eric J. Bigelow, Robert P. Dick, David Scott Krueger, and Hidenori Tanaka. Mechanistic mode connectivity. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22965–23004. PMLR, 2023. URL `https://proceedings.mlr.press/v202/lubana23a.html`.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Shreyas Malakarjun Patil, Loizos Michael, and Constantine Dovrolis. Neural sculpting: Uncovering hierarchically modular task structure in neural networks through pruning and network analysis. *Advances in Neural Information Processing Systems*, 36:18495–18531, 2023.

Giovanni Luca Marchetti, Christopher J Hillar, Danica Kragic, and Sophia Sanborn. Harmonics of learning: Universal fourier features emerge in invariant networks. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 3775–3797. PMLR, 2024.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pages 379–395. Springer, 2024.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Charles O'Neill and Thang Bui. Sparse autoencoders enable scalable and reliable circuit identification in language models. *arXiv preprint arXiv:2405.12522*, 2024.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

Jorge Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto, and Xavier Parra. Human activity recognition using smartphones. *UCI Machine Learning Repository*, 4:2, 2012.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Nosseiba Ben Salem, Younès Bennani, Joseph Karkazan, Abir Barbara, Charles Dacheux, and Thomas Gregory. Modular neural network approaches for surgical image recognition. *arXiv preprint arXiv:2307.08880*, 2023.

Naomi Saphra and Sarah Wiegreffe. Mechanistic? *arXiv preprint arXiv:2410.09087*, 2024.

Ryan Van Soelen and John W Sheppard. Using winning lottery tickets in transfer learning for convolutional neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jiménez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *Proceedings of the 41st International Conference on Machine Learning*, pages 50268–50287, 2024.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Siqi Zeng, Yifei He, Weiqiu You, Yifan Hao, Yao-Hung Hubert Tsai, Makoto Yamada, and Han Zhao. Efficient model editing with task vector bases: A theoretical framework and scalable approach. *arXiv preprint arXiv:2502.01015*, 2025.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf`.

Composable Sparse Subnetworks via Maximum-Entropy Principle - Appendix

A Appendix Structure

The appendix is structured as follows:

- Appendix B describes the extended **related work**;
- Appendix C focuses on detailing the **experimental setting** of our work. Specifically, we further describe the **training configurations and pipeline** (Appendix C.1), the **metrics** employed to evaluate the proposed methods (Appendix C.2), the procedure we developed to test for presence or absence of **mode connectivity** (Appendix C.3), the **datasets** considered for our experiments (Appendix C.4), the **pruning ratio per iteration** (Appendix C.5), the **hyperparameters** of our proposed procedure (Appendix C.6);
- Appendix D focuses on providing additional **experimental results** to support the findings in the main text. We report extended **confusion matrices** for individual submodules without (Appendix D.1) and with pruning (Appendix D.2), as well as for **pairwise** (Appendix D.3) and **incremental merges** (Appendix D.4). We also include **loss landscape plots** from the mode connectivity analysis (Appendix D.5), and a comprehensive table comparing merging results **with and without pruning across all loss functions** (Appendix D.6).

B Extended Related Work

Modular Neural Networks and Interpretability. Several works have explored modular neural architectures that learn to decompose computation across tasks [Kirsch et al., 2018, Han et al., 2021, Salem et al., 2023]. In particular, Kirsch et al. [2018] introduced a method to jointly learn both the modules and their composition end-to-end, showing interpretable specialization through controller-based selection. However, their approach does not isolate class-specific knowledge: modules are reused across multiple tasks and selected based on input context, making it difficult to extract independently meaningful functional units.

In contrast, our work explicitly aims to construct *class-specific functional modules*, where each module is trained to specialize in a single class and produce maximum-entropy outputs for all others. This sharp functional separation not only promotes compositionality via weight merging, but also improves *interpretability*: submodules can be understood in terms of their response to a single concept. Unlike these post hoc analyses, we impose structure *during training*, yielding functional units with clear class semantics and compositional behavior. This aligns with the goals of mechanistic interpretability [Saphra and Wiegreffe, 2024], where subnetworks (or “circuits” [Olah et al., 2020]) are analyzed as self-contained structures responsible for identifiable sub-tasks.

Our method can be seen as a complement to neuron-level analysis techniques such as those in Mu and Andreas [2020], which decompose single neuron activations into logical or perceptual components. However, such analyses often suffer from *polysemanticity*—the same neuron responding to unrelated stimuli—limiting their explanatory power. Instead, we enforce functional modularity at the network level, allowing the identification of interpretable, composable subgraphs. This modular decomposition is also conceptually connected to recent work on *harmonics of learning* [Marchetti et al., 2024], where symmetry structures constrain the function space; our class-specific modules may correspond to symmetry-aligned components in this broader framework.

Maximum Entropy Principle. The principle of maximum entropy (MaxEnt), introduced by Jaynes [1957], asserts that when making inferences under partial knowledge, one should prefer the probability distribution that satisfies known constraints while remaining otherwise maximally uninformative. Originally formulated to connect statistical mechanics and information theory, MaxEnt has since become a foundational tool in probabilistic modeling and statistical inference [Kuić, 2016, De Martino and De Martino, 2018]. Its generalization to complex-valued functions has also been explored, for example in image reconstruction tasks [Bajkova, 1992].

In our setting, MaxEnt serves as a guiding principle for functional specialization: we train submodules to be highly confident only on their assigned class and maximally uncertain (i.e., high-entropy) elsewhere. This interpretation differs from standard entropy regularization, which is typically used to smooth predictions or calibrate confidence scores [Marczak et al., 2024]. Rather than encouraging mild uncertainty, our max-entropy loss enforces *uniformity* outside the rewarded class set—actualizing the MaxEnt principle to promote class exclusion and module isolation. To our knowledge, this use of entropy as a strict compositional prior is novel within the context of modular deep learning.

Lottery Ticket Hypothesis and Sparse Training. Our approach builds on the Lottery Ticket Hypothesis (LTH), which states that sparse subnetworks in randomly initialized networks can be trained to match the performance of the full dense model [Frankle and Carbin, 2018, Zhou et al., 2019, Liu et al., 2024]. This idea has been extended theoretically to show that such subnetworks exist with high probability even in convolutional architectures with ReLU activations [Burkholz, 2022]. Empirical studies also demonstrate that global unstructured pruning outperforms structured pruning in high-sparsity regimes [Girish et al., 2021]. We adopt global pruning in our setting to retain the most functionally relevant connections across the entire network. LTH has been further explored in transfer learning [Van Soelen and Sheppard, 2019, Burkholz et al., 2022] and federated learning [Itahara et al., 2020]. In contrast to prior work that uses pruning primarily for compression or transfer, we leverage it to enforce *functional isolation*: pruning removes spurious capacity, encouraging submodules to specialize only on their rewarded classes.

Mode Connectivity and Model Merging. Recent work on model merging has explored how independently trained networks can be combined through linear or non-linear paths in weight space, a property known as *mode connectivity* [Ilharco et al., 2023]. This has inspired techniques for task arithmetic [Ortiz-Jimenez et al., 2023, Yang et al., 2024, Wang et al., 2024], which define task vectors as differences between fine-tuned and base models, enabling operations like unlearning and multi-task interpolation. These methods typically assume large pre-trained models and alignment in parameter space, and focus on editing rather than modular design.

Other approaches, such as Git-Rebasin or PLeas [Hazimeh et al., 2024, Zeng et al., 2025], address the challenge of merging models with different initializations via weight alignment. While these techniques focus on how to combine models *after* training, our method focuses on how to construct modules that are composable *by design*. Specifically, we show that submodules trained with max-entropy loss and pruning often exhibit linear mode connectivity—allowing for effective merging via weight summation—while also identifying cases where destructive interference occurs.

Our results suggest that mode connectivity may emerge naturally when modules are trained under entropy-based functional isolation, without requiring explicit alignment. This positions our work as complementary to task arithmetic and merging strategies: we do not assume pre-trained baselines, but instead provide a procedure to build sparse, specialized components that are compatible by construction.

C Extended Experimental setting

C.1 Training Configurations and Pipeline

We report here the key training configurations used in our experiments. All models were trained using the max-entropy loss and iterative magnitude pruning (IMP), with two pruning iterations and a final sparsity of 99% for MLPs and 60% for CNNs, unless stated otherwise. The optimizer used was Adam. We use the highest accuracy achieved on the validation set as our checkpointing strategy. All experiments are conducted on a workstation equipped with an Intel Core i9-10940X (14-core CPU running at 3.3GHz), 256GB of RAM, and a single Nvidia RTX A6000 GPU with 48GB of VRAM.

Temperature scaling for the softmax was fixed to t. Inputs were normalized when appropriate, and all datasets were used with standard training/validation splits. Each experiment was run with 5 random seeds.

All YAML configuration files are released alongside the code repository for full reproducibility.

Model Architectures and Hyperparameters

Common Settings. All models use ReLU activations, are trained with a batch size of 64, and are evaluated using the same training and pruning pipeline.

Table 3: Configurations.

Model	**Hidden Layers**	**Dropout**	t
Shallow MLP	[512]	0.5	5.0
Deep MLP	[512, 256]	0.5	5.0
CNN (LeNet-style)	Conv + [120, 64]	0.	1.0

The learning rate was set to 10^{-3} for MNIST and FMNIST across all architectures, and to 10^{-4} for Yeast and Human Activity Recognition, where only Shallow and Deep MLPs were evaluated.

Dataset-specific Input Dimensions and Classes

- **MNIST / FMNIST:** 784 input features (flattened 28×28 images), 10 classes.
- **Human Activity Recognition (HAR):** 561 input features, 5 classes.
- **Yeast:** 8 input features, 4 classes.

Training Pipeline Here we present an overview of a generic experiment. For each experiment:

- Given the set of classes $\mathcal{C}$ in the dataset, choose a subset of classes $\mathcal{C}' \subset \mathcal{C}$.
- Train one subnetwork module per class or per subset $\mathcal{R} \subseteq \mathcal{C}'$, using the max-entropy loss and IMP.
- Evaluate each subnetwork using:
  - **Accuracy** on rewarded classes.
  - **Mean entropy** on non-rewarded classes.
  - **Confusion matrix** on the validation set.

C.2 Formal Definition of Metrics

Accuracy on Rewarded Classes. Let $\mathcal{R} \subseteq \mathcal{C}$ be the set of rewarded classes, and let $\mathcal{D}_{\text{val}}^{\text{R}}$ denote the subset of validation data whose labels belong to $\mathcal{R}$. The accuracy over rewarded classes is defined as:

$$\text{Accuracy}_{\text{rewarded}} = \frac{1}{|\mathcal{D}_{\text{val}}^{\text{R}}|} \sum_{(x,y)\in\mathcal{D}_{\text{val}}^{\text{R}}} \mathbb{1}[\hat{y}(x) = y] \tag{6}$$

where $\mathbb{1}[\cdot]$ is the indicator function. This measures how well the subnetwork performs on the class subset it is trained to recognize.

Mean Entropy on Non-Rewarded Classes. Let $\mathcal{D}_{\text{val}}^{\text{non-R}}$ be the validation subset containing samples from non-rewarded classes, and let $N_{\text{non-R}}$ be its cardinality. We define the average entropy of the model's predictions over this set as:

$$\text{Entropy}_{\text{non-rewarded}} = \frac{1}{N_{\text{non-R}}} \sum_{x\in\mathcal{D}_{\text{val}}^{\text{non-R}}} H\left(\hat{y}(x)\right) \tag{7}$$

where $\hat{y}(x)$ is the predicted probability distribution for input x, and $H(p)$ denotes the entropy of a distribution p, defined as:

$$H(p) = -\sum_{i=1}^{|\mathcal{C}|} p_i \log p_i \tag{8}$$

This metric quantifies the uncertainty the model exhibits on inputs it is not supposed to classify, with higher values indicating closer adherence to the maximum entropy principle.

Confusion Matrix. Given a validation dataset $\mathcal{D}_{\text{val}}$ and a trained model f_θ, we define the confusion matrix $\mathbf{M} \in \mathbb{N}^{|\mathcal{C}|\times|\mathcal{C}|}$ such that each entry $M_{i,j}$ counts the number of samples with true label i that are predicted as class j:

$$M_{i,j} = \sum_{(x,y)\in\mathcal{D}_{\text{val}}} \mathbb{1}[y = i] \cdot \mathbb{1}[\hat{y}(x) = j] \tag{9}$$

where $\hat{y}(x) = \arg\max_k f_\theta(x)_k$ is the predicted class. This matrix allows us to qualitatively inspect specialization and interference across class predictions.

C.3 Additional explanations on Mode Connectivity

To further shed light on why merging via summation works (as demonstrated in the main paper), we study also the loss landscape through the lens of mode connectivity.

Mode connectivity refers to a phenomenon observed in the loss landscape of neural networks. When a neural network is trained, the optimization process (like SGD) typically finds a set of weights that minimizes the loss function. This set of weights is called a *mode* or a *local minimum* in the high-dimensional loss landscape. Several contributions [Frankle et al., 2020, Lubana et al., 2023] pointed out that these different modes (solutions found by, for example, training the same model architecture multiple times with different initializations or training procedures) are often not isolated; rather, they can frequently be connected by paths along which the loss value remains consistently low.

Many paths of different shapes can connect two models, θ_1 and θ_2, in weight space. We are specifically interested in testing for a path without significant loss barriers that includes the model $\theta_{[1,2]} = \theta_1 + \theta_2$, which results from merging the original models via summation. A key aspect of our setup, diverging from typical mode connectivity studies, is that θ_1 and θ_2 are initially specialized for different sets of rewarded classes. The merged model $\theta_{[1,2]}$ is intended to operate on the union of these class sets. Consequently, we evaluate for barriers using our *max-entropy loss* function. This loss function is consistently applied across the path and for the endpoint models (θ_1, θ_2), and it considers the full union of rewarded classes relevant to $\theta_{[1,2]}$.

Formally, in line with Lubana et al. [2023], let θ_1 and θ_2 be two sets of weights, $\mathcal{D}$ a dataset, f_θ a neural network parametrized by weights θ, and $\mathcal{L}$ our specific *max-entropy loss* function (evaluated on $\mathcal{D}$ considering the union of rewarded classes as described above). We say that θ_1 and θ_2 are mode connected along the path $\gamma_{\theta_1\to\theta_2}(t)$ if $\forall t \in [0,1]\ \mathcal{L}(f_{\gamma_{\theta_1\to\theta_2}(t)}(\mathcal{D})) \leq (1-t)\cdot\mathcal{L}(f_{\theta_1}(\mathcal{D})) + t\cdot\mathcal{L}(f_{\theta_2}(\mathcal{D})) + \epsilon$, where ϵ is a small margin, set to 2% of the first term on the r.h.s. following Frankle et al. [2020].

Each point along the path $\gamma_{\theta_1\to\theta_2}(t)$ represents a valid set of weights for the network $f(\cdot)$. Standard *linear* mode connectivity considers the path $\gamma^{\text{LMC}}_{\theta_1\to\theta_2}(t) = (1-t)\cdot\theta_1 + t\cdot\theta_2$. For our case, we require a path $\gamma_{\theta_1\to\theta_2}(t)$ that satisfies: (i) for $\gamma_{\theta_1\to\theta_2}(0) = \theta_1$, (ii) for $\gamma_{\theta_1\to\theta_2}(1) = \theta_2$, and (iii) $\exists t' \in [0,1]\ s.t.\ \gamma_{\theta_1\to\theta_2}(t') = \theta_1 + \theta_2$. As a consequence, we define the mode connectivity with respect to the following *piecewise linear* path:

$$\gamma_{\theta_1\to\theta_2}(t) = \begin{cases} \theta_1 + 2t\cdot\theta_2 & \text{if } t \leq 0.5 \\ 2(1-t)\cdot\theta_1 + \theta_2 & \text{if } t > 0.5 \end{cases} \tag{10}$$

It is trivial to see that our definition fulfill our properties, especially (iii) for $t = 0.5$.

C.4 Dataset Details

MNIST and Fashion-MNIST. Both are standard 10-class image classification benchmarks with grayscale 28×28 images. We normalize pixel values to $[0,1]$ and use the standard train/validation splits (60,000 train, 10,000 test).

Human Activity Recognition (HAR). This dataset contains sensor data from smartphones, labeled with six different physical activities. Each example has 561 standardized features. We adopt the original train/test split and reserve a validation portion from the training set.

Yeast. A protein localization dataset with 10 classes and 8 features. We normalize each feature to zero mean and unit variance, and split the dataset using a 65/15/20 ratio. Since the class distribution is highly imbalanced, we focused on the most represented 4 classes (i.e., `CYT`, `NUC`, `MIT`, `ME3`).

Preprocessing. For all tabular datasets, we apply z-score normalization and encode labels as integers. We use early stopping on the validation set and batch sizes of 64 during training.

C.5 Iterative Pruning Schedule Derivation

We follow the Iterative Magnitude Pruning (IMP) framework, in which a neural network is pruned over multiple iterations. At each pruning step, a fraction K of the remaining weights (those with the smallest absolute value) is removed. This process is repeated for N iterations until a target sparsity level $P \in (0, 1)$ is reached, corresponding to retaining only a $1 - P$ fraction of the original parameters.

Pruning Rate Schedule Let S_i denote the fraction of weights remaining after the i-th pruning iteration. The pruning process is multiplicative, meaning that:

$$S_i = (1 - K)^i,$$

assuming $S_0 = 1$ (i.e., all weights are initially present). After N pruning iterations, we desire a final sparsity P, which implies a remaining fraction $S_N = 1 - P$. Therefore, we solve:

$$(1 - K)^N = 1 - P.$$

Solving for K, we obtain the pruning rate per iteration:

$$K = 1 - (1 - P)^{1/N}.$$

This schedule ensures that pruning a fixed fraction K of the remaining weights at each of the N steps results in an overall sparsity of P at the end of the iterative process.

Pruning Algorithm Algorithm 2 summarizes the full procedure used in our experiments.

Algorithm 2 Iterative Magnitude Pruning with Max-Entropy Loss

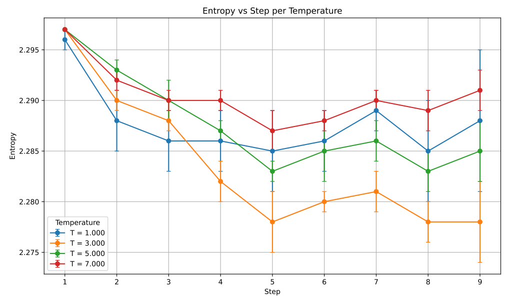
```
Initial weights $\theta_0$, dataset $D$, rewarded classes $R$, number of pruning iterations $N$, target sparsity $P$, epochs per iteration $E$
$\theta \leftarrow \theta_0$
$K \leftarrow 1 - (1 - P)^{1/N}$
iteration $i = 1$ to $N$
    Train model $f_\theta$ on $D$ using max-entropy loss with reward set $R$ for $E$ epochs
    Prune the fraction $K$ of weights in $\theta$ with the smallest absolute value
    Reset remaining weights in $\theta$ to their initial values from $\theta_0$
Train the final pruned subnetwork on $D$ using max-entropy loss with $R$ for $E$ epochs
```

C.6 Hyperparameter Tuning

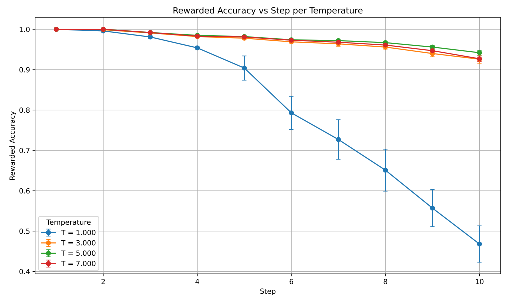
We perform a small-scale hyperparameter study with respect to two key components of the training pipeline: the softmax temperature used in the max-entropy loss, and the pruning ratio applied in iterative magnitude pruning. The temperature analysis is conducted on MNIST using a shallow MLP in the *complete merge* setting. For the pruning ratio, we provide two complementary studies: one on the shallow MLP in the same complete merge scenario, and one on CNNs (LeNet-style) in the *pairwise merge* setting, using both MNIST and FashionMNIST. All results are averaged over 5 random seeds.

Temperature. We study the effect of the temperature hyperparameter $t \in \{1, 3, 5, 7\}$, which scales the logits before the softmax activation and thus controls the sharpness of the output distribution. Lower temperatures produce more peaked distributions, while higher values encourage uniformity.

Figure 6a and 6b show the average entropy on non-rewarded classes and the rewarded accuracy across merge steps for different temperatures.



(a) Entropy on non-rewarded samples at each incremental merge step for different temperatures.

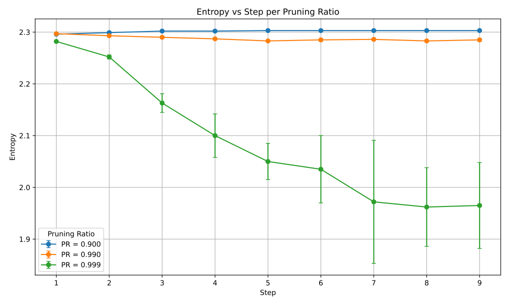


(b) Rewarded accuracy at each incremental merge step for different temperatures.

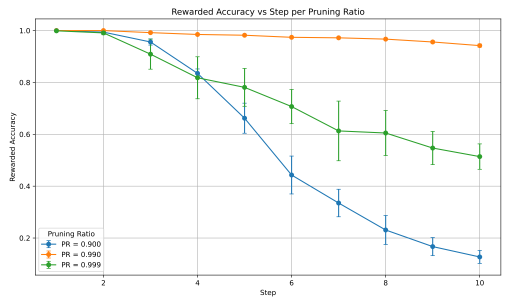
Figure 6: Effect of temperature scaling on specialization and composability. Temperature t controls the confidence of the softmax output; $t = 5$ achieves a good trade-off between high entropy on non-rewarded classes and accuracy on rewarded ones.

Pruning Ratio. We evaluate the impact of pruning ratio on submodule performance and merge behavior for two architectures: a shallow MLP and a CNN (LeNet-style). For the shallow MLP, we focus on the *complete merge* setting, testing three final sparsity levels—90%, 99%, and 99.9%—adjusted across two IMP iterations. As shown in Figure 7, pruning to 99% yields the best trade-off between specialization and compositionality: it maintains high entropy on non-rewarded inputs and preserves rewarded accuracy throughout the merge process. Lower sparsity (90%) retains excessive capacity, causing interference when modules are merged. Higher sparsity (99.9%) leads to underfitting and degraded performance.

For CNNs, we instead study pruning in the *pairwise merge* setting, focusing on class subset sizes $|\mathcal{R}| = 2$ and 5, across MNIST and FashionMNIST. The results, summarized in Figure 8, show that a pruning ratio of 0.6 consistently yields the highest average rewarded accuracy across both datasets and cardinalities. This indicates that, unlike MLPs, CNNs benefit from less aggressive sparsification.



(a) Entropy on non-rewarded samples across merge steps for different pruning ratios.



(b) Rewarded accuracy across merge steps for different pruning ratios.

Figure 7: Effect of pruning ratio on submodule specialization and merge stability. A pruning ratio of 0.99 yields a good balance between maintaining accuracy and increasing entropy on non-rewarded classes.

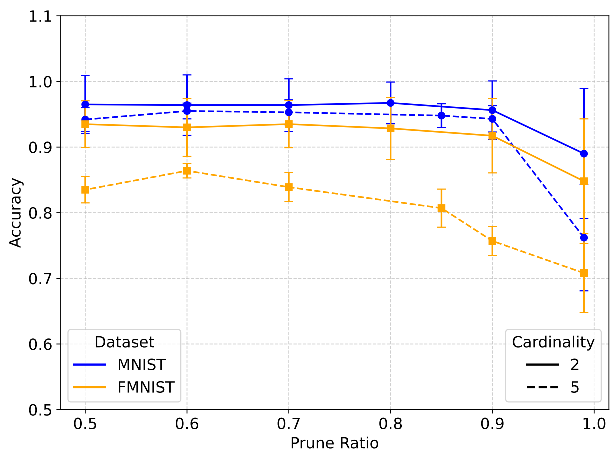


Figure 8: Effect of pruning ratio on CNN accuracy after pairwise merging on MNIST and FashionMNIST, with cardinality $|\mathcal{R}| = 2$ and 5. The optimal pruning ratio is 0.6.

D Extended Experiments

In this section, we present additional experimental results and qualitative insights.

First, we report representative examples for three settings:

- **Shallow MLP on MNIST**
- **Deep MLP on Human Activity Recognition**
- **CNN (LeNet-style) on FashionMNIST**

Then, we provide a comprehensive table reporting the full results for pairwise submodule merging across all configurations—varying model architectures, pruning settings (with and without IMP), cardinalities $|\mathcal{R}|$, and loss functions. This extended comparison complements the main paper, where only pruned results were shown, and helps quantify the impact of pruning across a wider set of conditions.

D.1 Confusion Matrices (No Pruning)

In this subsection, we show submodules trained using the max-entropy loss without applying iterative magnitude pruning (IMP). For each model-dataset pair, we show the confusion matrix of a submodule trained on a single rewarded class. These visualizations allow us to assess the degree of specialization and the presence of residual structure in the predictions for non-rewarded classes.

In all cases, the models exhibit strong activation on the rewarded class, while showing varying degrees of non-uniformity on the excluded classes—especially evident in the confusion matrix rows corresponding to non-rewarded labels.

D.2 Confusion Matrices After Pruning

We report confusion matrices for submodules trained with both the max-entropy loss and iterative magnitude pruning (IMP). These models are expected to exhibit stronger functional isolation, with reduced leakage across non-rewarded classes. Comparisons with Section D.1 highlight the impact of pruning on output uniformity and specialization.

D.3 Confusion Matrices for Pairwise Merges

We show representative confusion matrices for pairwise merges of submodules, each specialized on a distinct class. These examples illustrate that, even after weight summation, the merged model

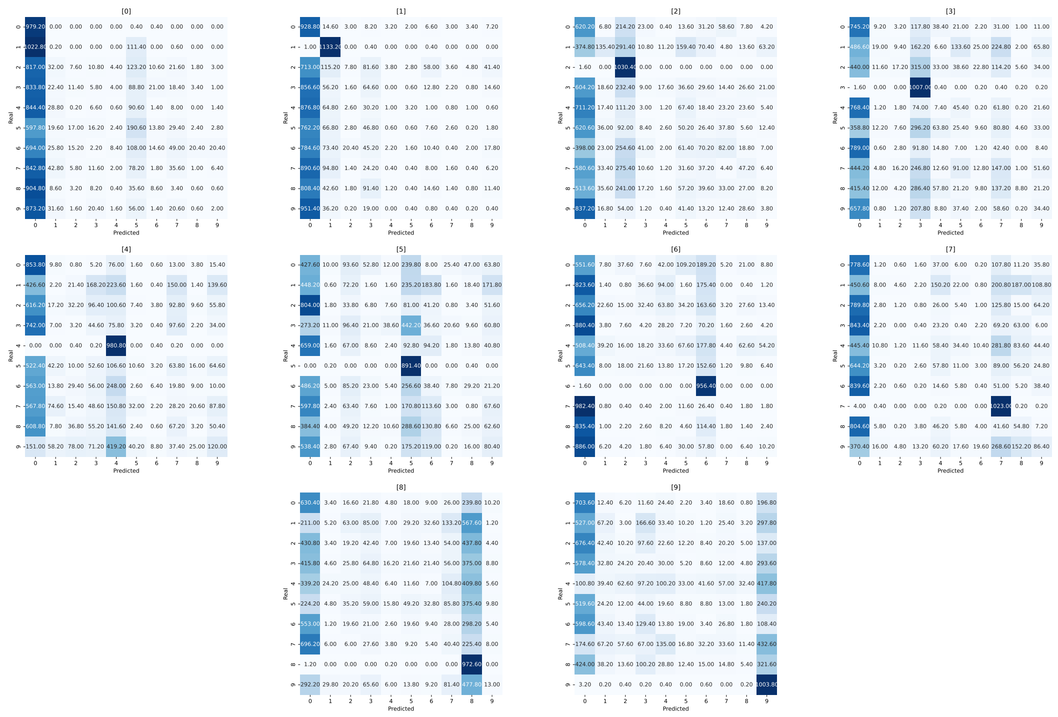


Figure 9: Confusion matrices for submodules trained on classes 0–9 without pruning, using the max-entropy loss. Each model is trained to specialize on one class and suppress predictions on others. Results refer to the **Shallow MLP** architecture applied to the **MNIST** dataset.

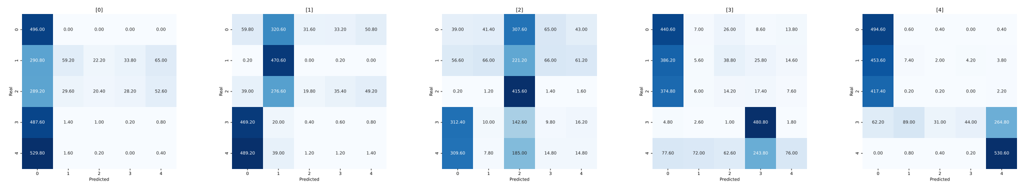


Figure 10: Confusion matrices for submodules trained on classes 0–4 without pruning, using the max-entropy loss. Each model is trained to specialize on one class and suppress predictions on others. Results refer to the **Deep MLP** architecture applied to the **HAR** dataset.

retains high confidence on rewarded classes and preserves uniform predictions on others—confirming compatibility and lack of interference.

D.4 Confusion Matrices for Incremental Merges

This section presents confusion matrices from various stages of incremental merging, where multiple submodules are combined one by one. The results demonstrate how compositional behavior is maintained across merge steps, and how the model continues to correctly isolate rewarded class behavior while suppressing predictions on excluded ones

D.5 Mode connectivity

In this section, we present additional plots analyzing the loss landscape between merged submodules. Following the framework described in Appendix C.3, we evaluate the relative difference in loss along the piecewise linear path connecting two modules and their sum.

We show incremental merging results for a **Deep MLP** on the **HAR** dataset and for a **CNN** on **FMNIST**, for both incremental and pairwise merging settings. In the majority of the cases, the pictures describe the desired behaviour (no or limited increase above 0). As already mentioned in the

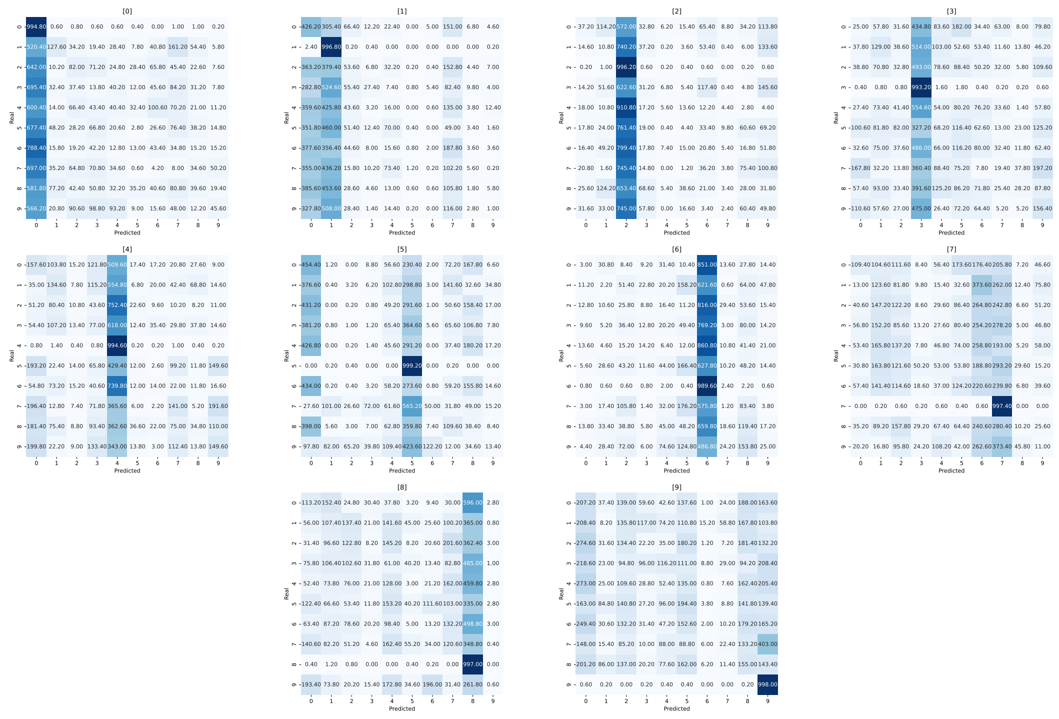


Figure 11: Confusion matrices for submodules trained on classes 0–9 without pruning, using the max-entropy loss. Each model is trained to specialize on one class and suppress predictions on others. Results refer to the **CNN** architecture applied to the **FMNIST** dataset. The CNN was pruned with a 99% pruning ratio.

main paper, it can also be observed that for the CNN, after many incremental merges (e.g., steps 9 and 10), the difficulty in achieving an optimal merge is reflected in the presence of barriers in the loss function landscape.

D.6 Complete Results for Pairwise Submodule Merging

In the main paper (Section 4), we report the results of pairwise submodule merging under the Iterative Magnitude Pruning (IMP) regime, as this generally led to the best overall performance. For completeness, here we provide the full set of results, including both pruned and non-pruned configurations. Table 4 reports the average rewarded accuracy and average entropy on non-rewarded classes for all model families (Shallow MLP, Deep MLP, CNN), across multiple datasets (MNIST, Fashion MNIST, HAR, Yeast), cardinalities $|\mathcal{R}| \in 1, 2, 5$, and loss functions (CrossEntropy, Quasi-MaxEnt, and MaxEnt).

We observe that:

- Models trained with IMP consistently outperform their non-pruned counterparts in most settings, particularly when using the MaxEnt loss.
- The MaxEnt loss yields the most reliable and composable modules across all model types and datasets, both with and without pruning.
- Non-pruned models trained with standard CrossEntropy struggle to maintain entropy on non-rewarded classes, often resulting in degraded compositional performance.
- The benefit of pruning is especially marked in deep MLPs, where without IMP the merging process suffers from significant degradation in rewarded accuracy.

These extended results support the conclusions drawn in the main paper: pruning strengthens functional specialization and enhances composability, especially when combined with entropy-based objectives.

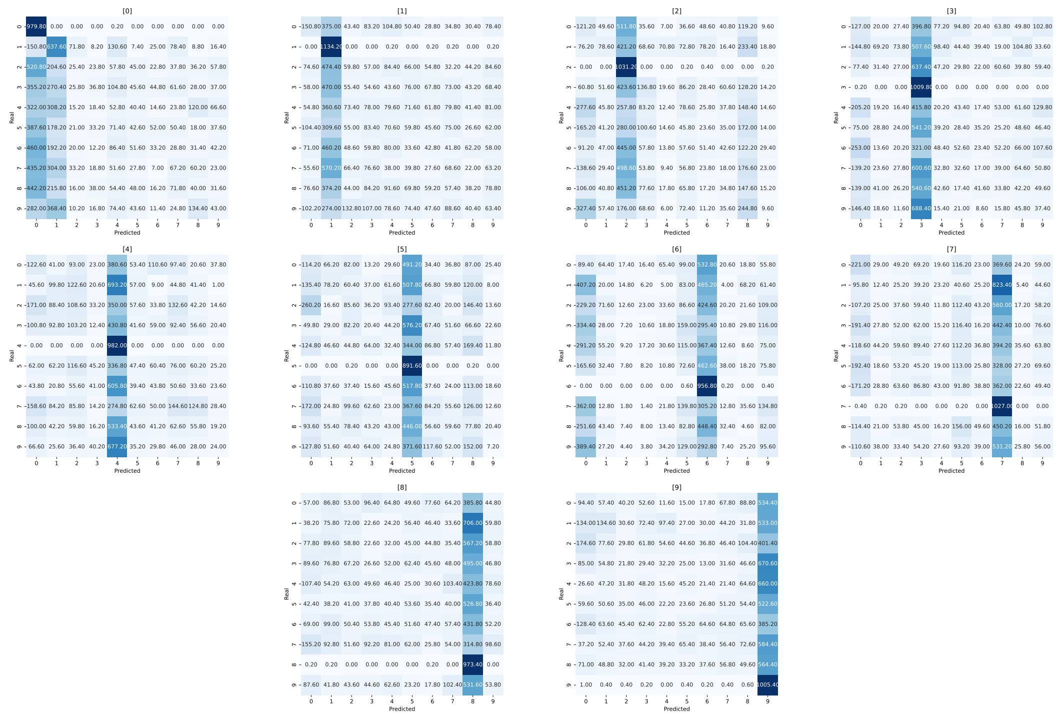


Figure 12: Confusion matrices for submodules trained on classes 0–9, using the max-entropy loss and pruning. Each model is trained to specialize on one class and suppress predictions on others. Results refer to the **Shallow MLP** architecture applied to the **MNIST** dataset.

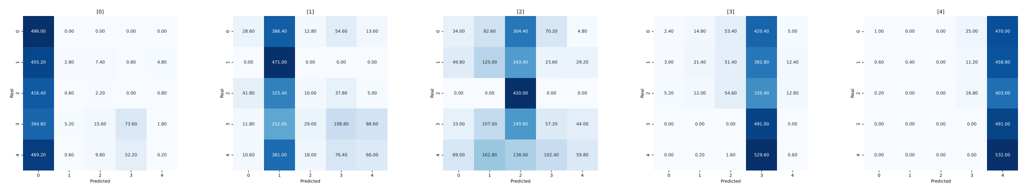


Figure 13: Confusion matrices for submodules trained on classes 0–4, using the max-entropy loss and pruning. Each model is trained to specialize on one class and suppress predictions on others. Results refer to the **Deep MLP** architecture applied to the **HAR** dataset.

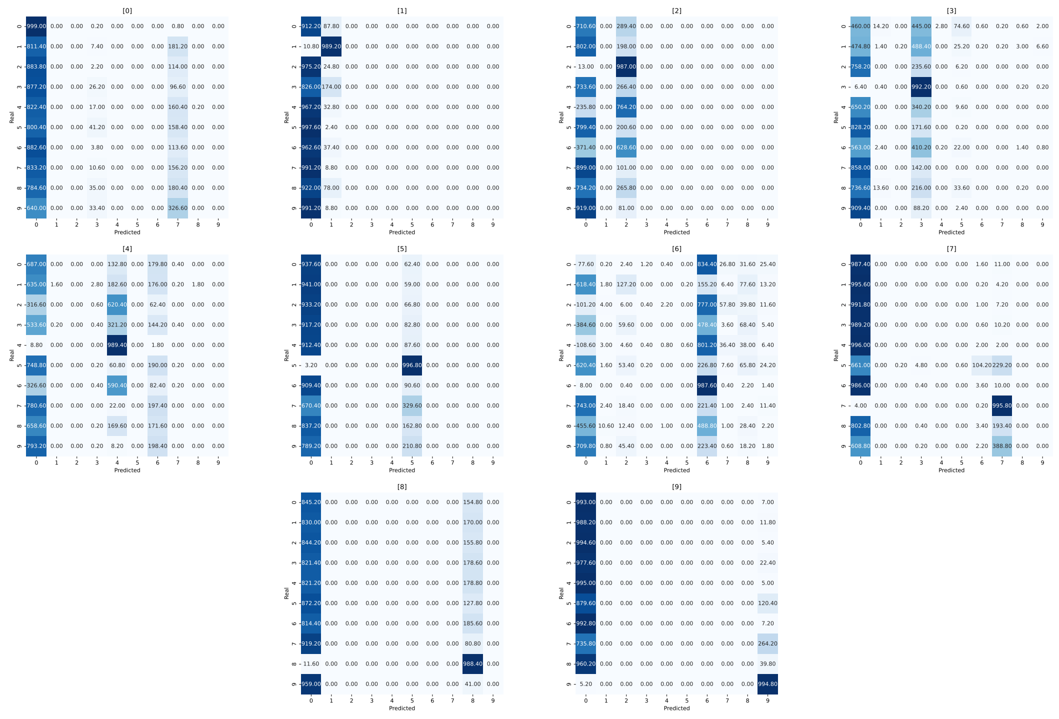


Figure 14: Confusion matrices for submodules trained on classes 0–9, using the max-entropy loss and pruning. Each model is trained to specialize on one class and suppress predictions on others. Results refer to the **CNN** architecture applied to the **FMNIST** dataset. The CNN was pruned with a 99% pruning ratio.

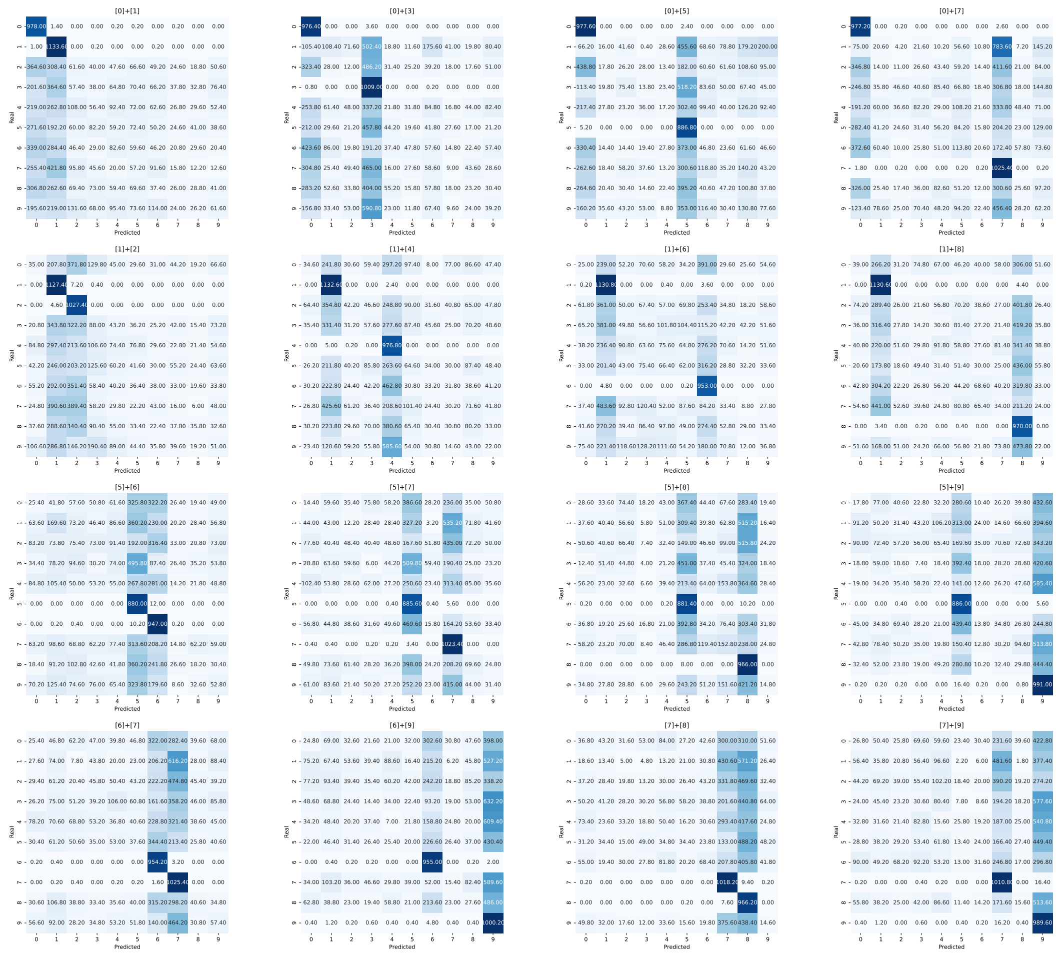
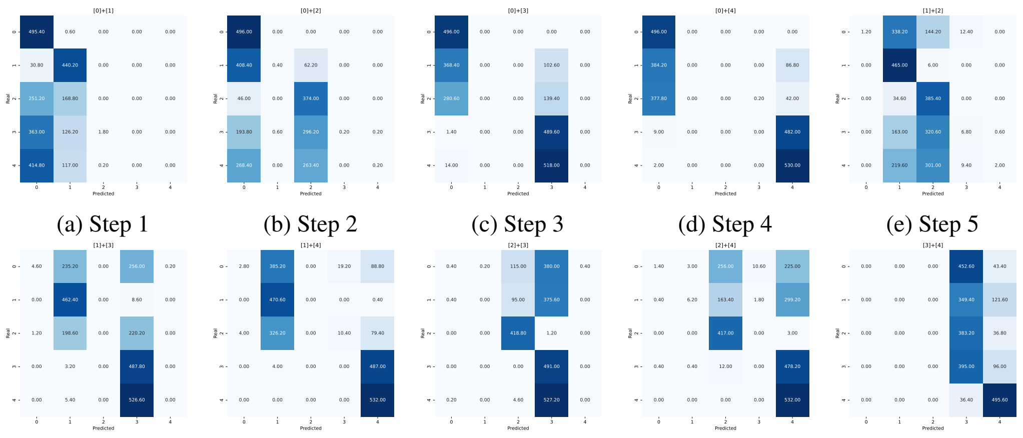


Figure 15: Confusion matrices for 16 pairwise merges of **Shallow MLP** submodules on **MNIST**. Each cell shows the average across 5 seeds.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4 (e) Step 5

(f) Step 6 (g) Step 7 (h) Step 8 (i) Step 9 (j) Step 10

Figure 16: Confusion matrices for 10 pairwise merges of **Deep MLP** submodules on **HAR**. Each cell shows the average across 5 seeds.

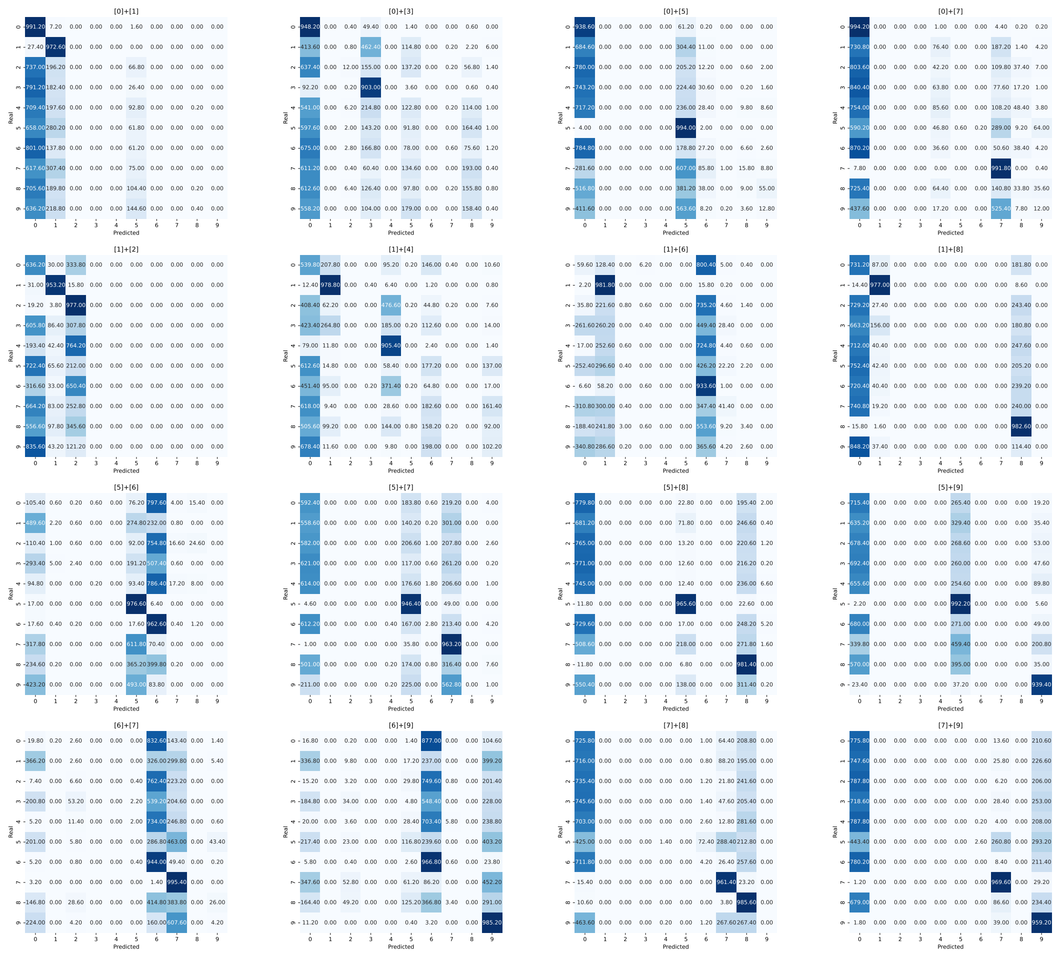
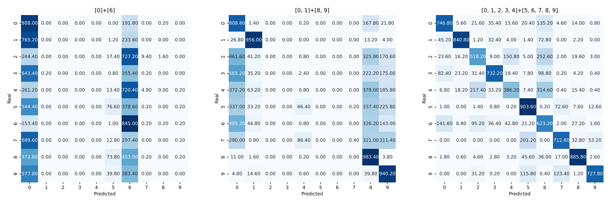
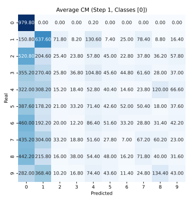


Figure 17: Confusion matrices for 16 pairwise merges of **CNN** submodules on **FMNIST**. Each cell shows the average across 5 seeds. The CNN was pruned with a 99% pruning ratio.

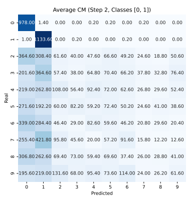


(a) Card. 1 (b) Card. 5 (c) Card. 5

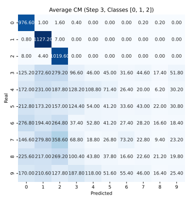
Figure 18: Confusion matrices for pairwise merges of **CNN** submodules on **FMNIST**, for exemplary modules of cardinality 1, 2, and 5 each. Each cell shows the average across 5 seeds. The CNN was pruned with a 99% pruning ratio.



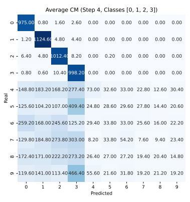
(a) Step 1



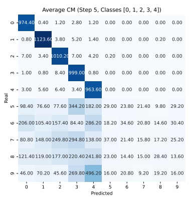
(b) Step 2



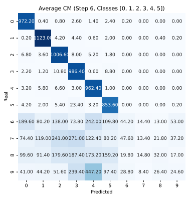
(c) Step 3



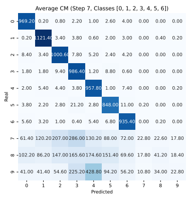
(d) Step 4



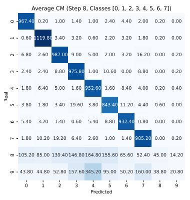
(e) Step 5



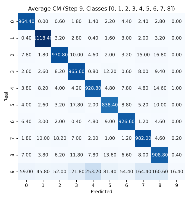
(f) Step 6



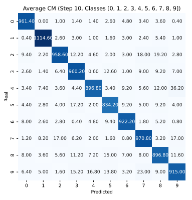
(g) Step 7



(h) Step 8

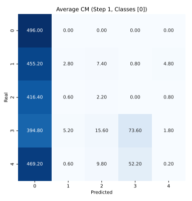


(i) Step 9

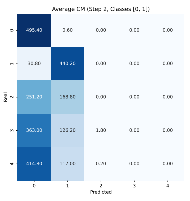


(j) Step 10

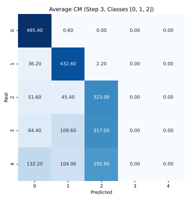
Figure 19: Confusion matrices for incremental merge steps of a **Shallow MLP** on **MNIST**. Each submodule is trained independently and merged following increasing order. Results are averaged over 5 seeds. The structure of predictions remains clean and interpretable throughout the merge.



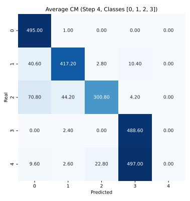
(a) Step 1



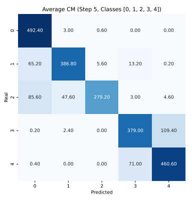
(b) Step 2



(c) Step 3

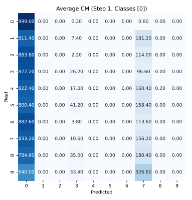


(d) Step 4

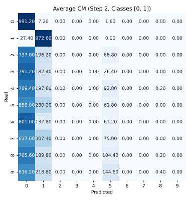


(e) Step 5

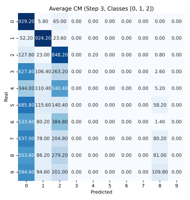
Figure 20: Confusion matrices for incremental merge steps of a **Deep MLP** on **HAR**. Each submodule is trained independently and merged following increasing order. Results are averaged over 5 seeds.



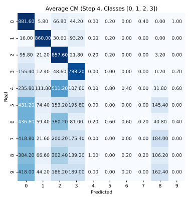
(a) Step 1



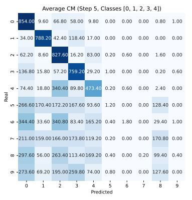
(b) Step 2



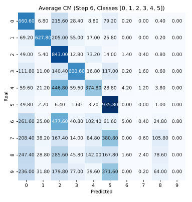
(c) Step 3



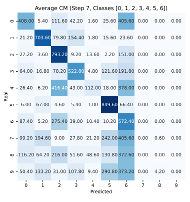
(d) Step 4



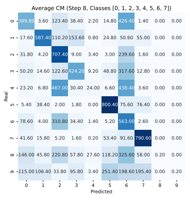
(e) Step 5



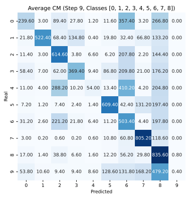
(f) Step 6



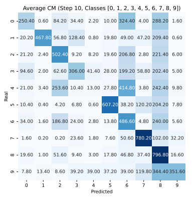
(g) Step 7



(h) Step 8

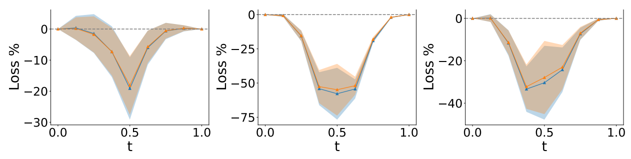


(i) Step 9



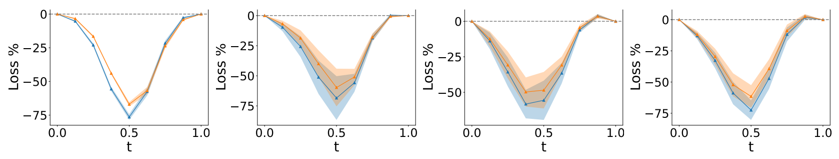
(j) Step 10

Figure 21: Confusion matrices for incremental merge steps of a **CNN** on **FMNIST**. Each submodule is trained independently and merged following increasing order. Results are averaged over 5 seeds. The CNN was pruned with a 99% pruning ratio.



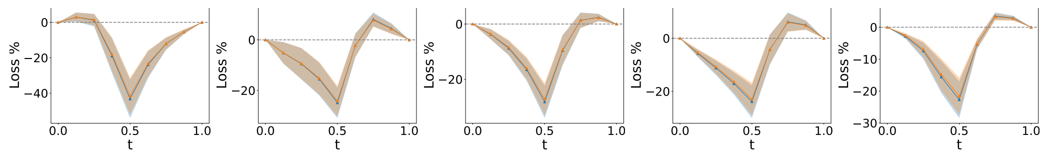
(a) Card. 1 (b) Card. 2 (c) Card. 5

Figure 22: Values for $\gamma_{\theta_1 \to \theta_2}(t)$ for pairwise merges of **CNN** submodules on **FMNIST**, for exemplary modules of cardinality 1, 2, and 5 each (same as Figure 18). Results are averaged over 5 seeds. The CNN was pruned with a 99% pruning ratio.

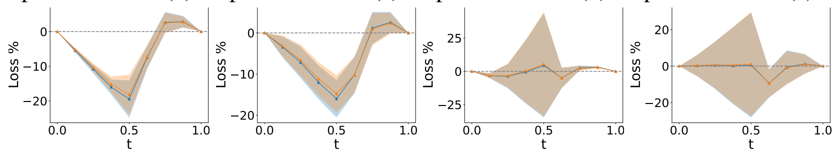


(a) Step 2 (b) Step 3 (c) Step 4 (d) Step 5

Figure 23: Values for $\gamma_{\theta_1 \to \theta_2}(t)$ for incremental merge steps of a **Deep MLP** on **HAR** (same as Figure 20). Each submodule is trained independently and merged following increasing order. Results are averaged over 5 seeds.



(a) Step 2 (b) Step 3 (c) Step 4 (d) Step 5 (e) Step 6



(f) Step 7 (g) Step 8 (h) Step 9 (i) Step 10

Figure 24: Values for $\gamma_{\theta_1 \to \theta_2}(t)$ for incremental merge steps of a **CNN** on **FMNIST** (same as Figure 21). Each submodule is trained independently and merged following increasing order. Results are averaged over 5 seeds. The CNN was pruned with a 99% pruning ratio.

			Dataset	FMNIST		MNIST		HAR		Yeast	
Model	$\lvert\mathcal{R}\rvert$	IMP	loss	Entropy	Rewarded Acc	Entropy	Rewarded Acc	Entropy	Rewarded Acc	Entropy	Rewarded Acc
Shallow MLP	1	No	XE	0.028 (0.016)	0.752 (0.248)	0.001 (0.003)	0.492 (0.024)	0.463 (0.521)	0.853 (0.191)	0.237 (0.259)	0.571 (0.162)
			QME	2.041 (0.064)	0.910 (0.083)	2.038 (0.056)	0.964 (0.037)	0.958 (0.146)	0.971 (0.041)	0.843 (0.107)	0.792 (0.097)
			ME	2.300 (0.001)	0.970 (0.012)	2.300 (0.001)	0.970 (0.012)	1.598 (0.067)	**0.987** (0.010)	0.827 (0.175)	0.806 (0.153)
		Yes	XE	0.183 (0.151)	0.877 (0.186)	0.507 (0.150)	0.708 (0.179)	1.255 (0.191)	0.855 (0.197)	1.378 (0.004)	0.631 (0.101)
			QME	2.031 (0.132)	0.908 (0.105)	2.082 (0.005)	0.973 (0.014)	1.196 (0.073)	0.946 (0.046)	1.056 (0.082)	0.799 (0.106)
			ME	2.287 (0.005)	**0.977** (0.005)	2.287 (0.005)	**0.991** (0.005)	1.716 (0.033)	0.985 (0.016)	1.146 (0.077)	**0.853** (0.127)
	2	No	XE	0.068 (0.069)	0.569 (0.111)	0.127 (0.050)	0.572 (0.114)	0.320 (0.274)	0.761 (0.196)	–	0.435 (0.005)
			QME	1.980 (0.095)	0.883 (0.061)	1.719 (0.204)	0.954 (0.023)	0.415 (0.300)	0.930 (0.029)	–	0.566 (0.011)
			ME	2.297 (0.002)	0.957 (0.013)	2.297 (0.002)	0.957 (0.013)	1.521 (0.171)	**0.940** (0.017)	–	0.527 (0.012)
		Yes	XE	0.277 (0.108)	0.786 (0.141)	0.332 (0.080)	0.838 (0.074)	1.167 (0.217)	0.871 (0.055)	–	0.381 (0.036)
			QME	1.824 (0.207)	0.917 (0.049)	1.683 (0.216)	0.959 (0.016)	1.013 (0.223)	0.891 (0.024)	–	0.559 (0.011)
			ME	2.267 (0.007)	**0.977** (0.005)	2.268 (0.007)	**0.980** (0.006)	1.670 (0.045)	0.935 (0.027)	–	**0.616** (0.010)
	5	No	XE	–	0.456 (0.036)	–	0.571 (0.076)	–	–	–	–
			QME	–	0.687 (0.077)	–	0.896 (0.012)	–	–	–	–
			ME	–	0.945 (0.008)	–	0.945 (0.008)	–	–	–	–
		Yes	XE	–	0.480 (0.075)	–	0.842 (0.027)	–	–	–	–
			QME	–	0.741 (0.026)	–	0.905 (0.023)	–	–	–	–
			ME	–	**0.946** (0.004)	–	**0.946** (0.004)	–	–	–	–
Deep MLP	1	No	XE	0.041 (0.055)	0.675 (0.220)	0.000 (0.000)	0.505 (0.025)	0.519 (0.611)	0.823 (0.227)	0.068 (0.169)	0.591 (0.144)
			QME	1.846 (0.184)	0.920 (0.110)	1.983 (0.175)	0.978 (0.020)	0.812 (0.190)	0.975 (0.026)	0.655 (0.107)	0.796 (0.102)
			ME	2.290 (0.019)	0.883 (0.148)	2.299 (0.007)	0.819 (0.104)	1.716 (0.058)	0.984 (0.012)	0.891 (0.177)	0.836 (0.136)
		Yes	XE	0.215 (0.150)	0.802 (0.205)	0.422 (0.190)	0.689 (0.194)	1.059 (0.313)	0.840 (0.210)	1.386 (0.000)	0.553 (0.144)
			QME	1.842 (0.329)	0.868 (0.176)	2.061 (0.046)	0.954 (0.045)	1.098 (0.128)	0.819 (0.155)	0.883 (0.118)	0.783 (0.114)
			ME	2.245 (0.072)	**0.975** (0.039)	2.291 (0.031)	**0.990** (0.005)	1.697 (0.050)	**0.988** (0.016)	1.063 (0.109)	**0.859** (0.120)
	2	No	XE	0.034 (0.030)	0.565 (0.097)	0.140 (0.062)	0.602 (0.118)	0.324 (0.268)	0.708 (0.150)	–	0.438 (0.007)
			QME	2.032 (0.145)	0.865 (0.079)	1.822 (0.227)	0.897 (0.065)	0.213 (0.194)	0.903 (0.060)	–	0.570 (0.021)
			ME	2.283 (0.016)	0.735 (0.087)	2.297 (0.008)	0.755 (0.090)	1.725 (0.039)	**0.946** (0.026)	–	0.587 (0.036)
		Yes	XE	0.166 (0.083)	0.678 (0.081)	0.208 (0.054)	0.743 (0.111)	1.065 (0.344)	0.824 (0.068)	–	0.382 (0.046)
			QME	1.347 (0.484)	0.863 (0.088)	1.224 (0.352)	0.912 (0.038)	0.665 (0.251)	0.787 (0.093)	–	0.562 (0.023)
			ME	2.197 (0.095)	**0.898** (0.045)	2.275 (0.026)	**0.976** (0.011)	1.607 (0.134)	0.930 (0.028)	–	**0.589** (0.018)
	5	No	XE	–	0.534 (0.053)	–	0.580 (0.119)	–	–	–	–
			QME	–	0.627 (0.101)	–	0.800 (0.025)	–	–	–	–
			ME	–	0.483 (0.048)	–	0.670 (0.106)	–	–	–	–
		Yes	XE	–	0.400 (0.082)	–	0.710 (0.044)	–	–	–	–
			QME	–	0.701 (0.084)	–	0.834 (0.036)	–	–	–	–
			ME	–	**0.774** (0.043)	–	**0.919** (0.003)	–	–	–	–
CNN	1	No	XE	0.038 (0.065)	0.639 (0.206)	0.072 (0.168)	0.519 (0.072)	–	–	–	–
			QME	1.247 (0.280)	0.916 (0.087)	1.261 (0.291)	0.974 (0.034)	–	–	–	–
			ME	2.297 (0.004)	0.908 (0.085)	2.285 (0.019)	0.968 (0.033)	–	–	–	–
		Yes	XE	0.016 (0.026)	0.557 (0.150)	0.041 (0.128)	0.529 (0.080)	–	–	–	–
			QME	0.981 (0.322)	0.880 (0.136)	1.001 (0.377)	0.961 (0.078)	–	–	–	–
			ME	2.297 (0.039)	**0.960** (0.097)	2.286 (0.057)	**0.989** (0.012)	–	–	–	–
	2	No	XE	0.090 (0.038)	0.632 (0.151)	0.092 (0.052)	0.736 (0.131)	–	–	–	–
			QME	1.203 (0.321)	0.910 (0.047)	0.742 (0.359)	0.934 (0.054)	–	–	–	–
			ME	2.137 (0.135)	0.876 (0.062)	2.012 (0.241)	0.959 (0.034)	–	–	–	–
		Yes	XE	0.052 (0.038)	0.604 (0.169)	0.070 (0.042)	0.624 (0.138)	–	–	–	–
			QME	0.705 (0.397)	0.888 (0.071)	0.312 (0.307)	0.928 (0.074)	–	–	–	–
			ME	1.984 (0.304)	**0.930** (0.044)	1.950 (0.456)	**0.965** (0.046)	–	–	–	–
	5	No	XE	–	0.519 (0.080)	–	0.698 (0.151)	–	–	–	–
			QME	–	0.798 (0.029)	–	0.892 (0.056)	–	–	–	–
			ME	–	0.826 (0.038)	–	0.926 (0.030)	–	–	–	–
		Yes	XE	–	0.512 (0.066)	–	0.777 (0.056)	–	–	–	–
			QME	–	0.779 (0.066)	–	0.850 (0.084)	–	–	–	–
			ME	–	**0.864** (0.011)	–	**0.955** (0.012)	–	–	–	–

Table 4: Average performance of pairwise-summed sub-modules. For each group model-cardinality-dataset, the best approach is highlighted in **bold**.