A EXPERIMENTAL SETUP

A.1 DATASETS AND METRICS.

We perform classification tasks on four popular image classification benchmarks, including CIFAR-10, CIFAR-100 (Krizhevsky, 2009), Tiny ImageNet (Deng et al., 2009) and ImageNet (Krizhevsky et al., 2012) datasets. Additionally, we examine our method on the challenging long-tailed dataset, CIFAR10-LT and CIFAR100-LT datasets (Cao et al., 2019).

- **CIFAR10** is a collection of 60,000 32x32 color images spanning 10 different classes, such as automobiles, birds, and ships, with each class containing 6,000 images. It is commonly used in machine learning and computer vision tasks for object recognition, serving as a benchmark to evaluate the performance of various algorithms.
- **CIFAR100** is a diverse and challenging image dataset consisting of 60,000 32x32 color images spread across 100 different classes. Each class represents a distinct object or scene, making it a comprehensive resource for fine-grained image classification and multi-class tasks. CIFAR-100 is widely used in machine learning research to evaluate the performance of models in handling a wide range of object recognition challenges.
- **Tiny ImageNet** is a compact but diverse dataset containing thousands of small-sized images, each belonging to one of 200 categories. This dataset serves as a valuable resource for tasks like image classification, with each image encapsulating a rich variety of objects, animals, and scenes, making it ideal for training and evaluating machine learning models.

- **CIFAR10-LT & CIFAR100-LT** are the long-tailed version of CIFAR10 and CIFAR100 datasets with imbalance ratio $\rho = 100$.

For the out-of-distribution detection task, we use CIFAR-10, and CIFAR-100, Tiny ImageNet as in-distribution datasets, and use CIFAR-10, CIFAR-100, Tiny ImageNet, SVHN, LSUN, Gaussian Noise, Uniform Noise, as out-of-distribution datasets. Additionally, we adopt a more challenging OOD detection benchmark, named semantically coherent out-of-distribution detection (SC-OOD) (Yang et al., 2021).

- **SVHN.** The Street View House Numbers (SVHN) dataset is a comprehensive collection of house numbers captured from Google Street View images. It consists of over 600,000 images of house numbers from real-world scenes, making it a critical resource for tasks like digit recognition and localization. SVHN's diversity in backgrounds, fonts, and lighting conditions makes it a challenging but vital dataset for training and evaluating machine learning algorithms in the domain of computer vision.
- **LSUN.** The LSUN (Large-scale Scene Understanding) dataset is a vast collection of high-resolution images, primarily focused on scenes and environments. It encompasses a diverse range of scenes, including bedrooms, kitchens, living rooms, and more. LSUN serves as a valuable resource for tasks such as scene recognition and understanding due to its extensive coverage of real-world contexts and rich visual content.
- **Gaussian Noise and Uniform Noise.** After introducing Gaussian Noise or Uniform Noise to the dataset, we obtain a modified dataset, which is then utilized as an OOD dataset for our experiments. We implement this operation using the library from Kirchheim et al. (2022).

- **SC-OOD benchmark.** The SC-OOD (Semantically Coherent Out-of-Distribution) benchmark is designed for evaluating out-of-distribution detection models by focusing on semantic coherence. This benchmark addresses the limitations of traditional benchmarks that often require models to distinguish between objects with similar semantics from different datasets, such as CIFAR dogs and ImageNet dogs.

We use five key metrics to evaluate the performance of ID classification and OOD detection tasks.

- **Accuracy.** This is defined as the ratio of the number of correct predictions to the total number of predictions made. We report top-1 classification accuracy on the test(val) sets of ID datasets.

- **FPR (95% TPR).** This metric stands for 'False Positive Rate at 95% True Positive Rate'. It measures the proportion of negative instances that are incorrectly classified as positive when the true positive rate is 95%. A lower FPR at 95% TPR is desirable as it indicates fewer false alarms while maintaining a high rate of correctly identified true positives.
- **Detection Error (95% TPR).** Detection Error at 95% TPR is a metric that quantifies the overall error rate when the model achieves a true positive rate of 95%. It combines false negatives and false positives to provide a single measure of error. Lower detection error values indicate better performance, as the model successfully identifies more true positives with fewer errors.
- **AUROC** is short for Area Under the Receiver Operating Characteristic Curve(AUROC). This metric measures the ability of a model to distinguish between in-distribution and OOD samples. The ROC curve plots the true positive rate against the false positive rate at various threshold settings. The AUROC is the area under this curve, with higher values (closer to 1.0) indicating better discrimination between in-distribution and OOD samples.
- **AURP** is short for Area Under the Precision-Recall Curve (AUPR), this metric is particularly useful in scenarios where there is a class imbalance (a significant difference in the number of in-distribution and OOD samples). It plots precision (the proportion of true positives among positive predictions) against recall (the proportion of true positives identified). Higher AUPR values suggest better model performance, especially in terms of handling the balance between precision and recall.

Implementation Details. Our Split Ensemble model was trained over 200 epochs using a single NVIDIA A100 GPU with 80GB of memory, for experiments involving CIFAR-10, CIFAR-100, and Tiny ImageNet datasets. For the larger-scale ImageNet dataset, we employ 8 NVIDIA A100 GPUs, each with 80GB memory, to handle the increased computational demands. We use an SGD optimizer with a momentum of 0.9 and weight decay of 0.0005. We also adopt a 200-epoch cosine learning rate schedule with 10 warm-up epochs and a batchsize of 256. Our experiments typically run for approximately 2 hours on both CIFAR-10 and CIFAR-100 datasets, whereas on the Tiny ImageNet and ImageNet datasets, they take approximately 10 hours and 24 hours, respectively. We employ data augmentation techniques such as rotation and flip during the training phase, while the testing phase does not involve data augmentation. As for the backbone models in our experiments, we utilize the standard ResNet-18 and ResNet-34 architectures. We heuristically decide the number of submodels in the Split-Ensemble via ablation study, where we find 8 submodels for ImageNet-1K and 5 submodels for other datasets leads to the best performance in both ID and OOD detection. The classes are grouped based on semantic similarity into subtasks for the submodels to learn.

B Pseudo code for Split-Ensemble training

The Pseudo code of Split-Ensemble training is available in Algorithm 1.

C Additional results and visualizations

In this section, we provide additional results in comparison with baseline methods in different settings as well as ablation results on our design choices following the discussion in Section 5.3.

Subtask grouping strategy In Section 3.1, we propose to use the group of classes that are semantically-close to form each subtasks of the complementary task splitting. Here we verify this intuition against have random assignment of classes to each subtask. As illustrated in Table 5, having semantically-close subtask grouping significantly improves the OOD detection ability of the Split-Ensemble model over that of random grouping. This improvement is more significant with more subtask splittings. We believe that semantic grouping of subtasks help the submodels to better learn the difference between ID classes and OOD classes of the subtask, as the semantically-close ID classes may share more distinct features comparing to other classes.

Number of subtask splittings We conducted an analysis to explore the impact of the number of splits on the accuracy and OOD detection performance of the Split-Ensemble model. Unlike tradi-

Algorithm 1 Training the Split-Ensemble model

```
# Initialization and preparation
Load dataset {X, Y}
Subtask label conversion Y → Ŷ_i as Equation (3)
Initialize Split-Ensemble F with all submodels f_i sharing backbone model
# Split ensemble training
while Training do
    Update F to minimize L_ens in Equation (4) with SGD
    # Iterative splitting and pruning
    if Epoch % Prune_Interval == 0 then
        # Splitting
        if ∃ branch in F with multiple submodel f_i sharing all layers then
            for Layers in the branch shared only by f_i do
                Compute sensitivity map following Equation (5) for each f_i
                Compute MCT of the layer following Equation (6)
                if MCT < threshold then
                    Split branch at the layer
                    Break
        # Pruning
        if FLOPs > target then
            for All submodels f_i do
                Compute I^i_S for all S following Equation (7)
                Rank I^i_S to decide prunable structures with min_S I^L_S
                Remove structures prunable for (all) corresponding submodels
```

Table 5: **Ablation on subtask grouping strategy.** Models are trained on CIFAR-100. OOD detection is against the CIFAR-10 dataset.

# splits	Subtask grouping	Accuracy	AUROC
2	Random	77.3	78.9
	Semantic	**77.8**	**79.6**
4	Random	77.3	77.5
	Semantic	**77.5**	**79.1**
5	Random	77.4	77.3
	Semantic	**77.9**	**78.9**

tional ensemble that repeatedly learn the same task with more submodels, Split-Ensemble always learns a complementary subtask splitting corresponding to the original task. Increasing the amount of splits will therefore enable each submodel to learn a simpler subtasks with less ID classes, intuitively leading to a model architecture with more yet smaller branches. As shown in Table 6, the Split-Ensemble accuracy is not sensitive to the number of splits, showing the scalability of our learning algorithm. For OOD detection, a larger number of splits enables each submodel to learn its OOD-aware objective more easily, therefore leading to better AUROC. Yet the performance may suffer from aggressive pruning with too much branches in the Split-Ensemble, as observed with a large MCT threshold in Table 4. An interesting future direction would be automatically design the amount of subtask splitting and the grouping of each subtask during the training process to better fit the subtasks to the Split-Ensemble architecture.

Table 6: **Ablation on number of splits.** Models are trained on CIFAR-100. OOD detection is against the CIFAR-10 dataset. All models are constrained with single-model computation cost.

# splits	2	4	5	8	10
Accuracy	77.7	**78.0**	77.9	77.5	77.3
AUROC	78.1	78.2	79.9	**80.4**	77.3

Additional classification results on ImageNet We perform classification on the large-scale ImageNet1K dataset to examine our method. As shown in Table. 7, our method continues to outperform the single and $4\times$ more costly ensemble methods, demonstrating the effectiveness of our design.

Table 7: **Classification performance on ImageNet1K dataset.** The results are reported for models with ResNet-18 backbone. Best score in **bold**.

Method	Acc
Single	69.0
Naive Ensemble	69.4
Split-Ensemble (ours)	**70.9**

Additional classification and OOD detection results on CIFAR10-LT with SC-OOD benchmark We assess our method on CIFAR10-LT, a complex long-tailed dataset, to evaluate its robustness. As evidenced in Table 8, our approach consistently outperforms in all four metrics. Remarkably, this is achieved with only a quarter of the computational cost compared to baseline methods. This underscores our model's efficiency and effectiveness in managing intricate classification and OOD detection tasks.

Table 8: **Comparison between previous state-of-the-art ensemble-based methods and ours on the SC-OOD CIFAR10-LT benchmarks.** The results are reported for models with ResNet-18 backbone. Best score in **bold**, second best underlined.

Method	Accuracy ↑	FPR95 ↓	AUROC ↑	AUPR ↑
Naive Ensemble	12.7	98.4	45.3	50.9
MC-Dropout	63.4	90.6	66.6	66.1
MIMO	35.7	96.3	55.1	56.9
MaskEnsemble	67.7	89.0	66.82	67.4
BatchEnsemble	70.1	87.45	68.0	68.7
FilmEnsemble	<u>72.5</u>	<u>84.32</u>	<u>75.5</u>	<u>76.0</u>
Split-Ensemble (ours)	**73.7**	**80.5**	**81.7**	**77.6**

Additional OOD detection results on CIFAR10 with SC-OOD benchmark We further compare our methods with previous state-of-the-art methods. In Table. 9, our Split-Ensemble model outperforms single-model approaches in OOD detection without incurring additional computational costs or requiring extra training data. Its consistent high performance across key metrics highlights its robustness and efficiency, underscoring its practical utility in OOD tasks. In Table. 10, our Split-Ensemble model consistently outshines other ensemble-based methods in both image classification and OOD detection, achieving top rankings across all key metrics, which underscores the model's efficiency and effectiveness.

Table 9: **Comparison between previous state-of-the-art single-model-based methods and ours on the SC-OOD CIFAR10 benchmarks.** The results are reported for models with ResNet-18 backbone. Best score in **bold**, second best underlined.

Method	Additional Data	FPR95 ↓	AUROC ↑	AUPR ↑
ODIN	✗	52.0	82.0	85.1
EBO	✗	50.0	83.8	85.1
OE	✓	50.5	88.9	87.8
MCD	✓	73.0	83.9	80.5
UDG	✗	55.6	90.7	88.3
UDG	✓	**36.2**	**93.8**	**92.6**
Split-Ensemble (ours)	✗	<u>45.5</u>	<u>91.1</u>	<u>89.9</u>

Table 10: **Comparison between previous state-of-the-art ensemble-based methods and ours on the SC-OOD CIFAR10 benchmarks.** The results are reported for models with ResNet-18 backbone. Best score in **bold**, second best underlined.

Method	Additional Data	FPR95 ↓	AUROC ↑	AUPR ↑
Naive Ensemble	4x	42.3	90.4	90.6
MC-Dropout	4x	54.9	88.7	88.0
MIMO	4x	73.7	83.5	80.9
MaskEnsemble	4x	53.2	87.7	87.9
BatchEnsemble	4x	50.4	89.2	88.6
FilmEnsemble	4x	**42.6**	**91.5**	**91.3**
Split-Ensemble (ours)	1x	45.5	91.1	89.9

Model activation map visualization We visualize the learned feature map activations of a Split-Ensemble model across different layers using Score-CAM (Wang et al., 2020) in Figure 4. The shared feature maps, delineated by dashed lines, represent the common features extracted across different submodels, emphasizing the model's capacity to identify and leverage shared representations. The distinct feature maps outside the dashed boundaries correspond to specialized features pertinent to individual sub-tasks, demonstrating the Split-Ensemble model's ability to focus on unique aspects of the data when necessary. This visualization underscores the effectiveness of the Split-Ensemble architecture, highlighting its dual strength in capturing both shared and task-specific features within a single, cohesive framework, thereby bolstering its robustness and adaptability in handling diverse image classification and OOD detection tasks.

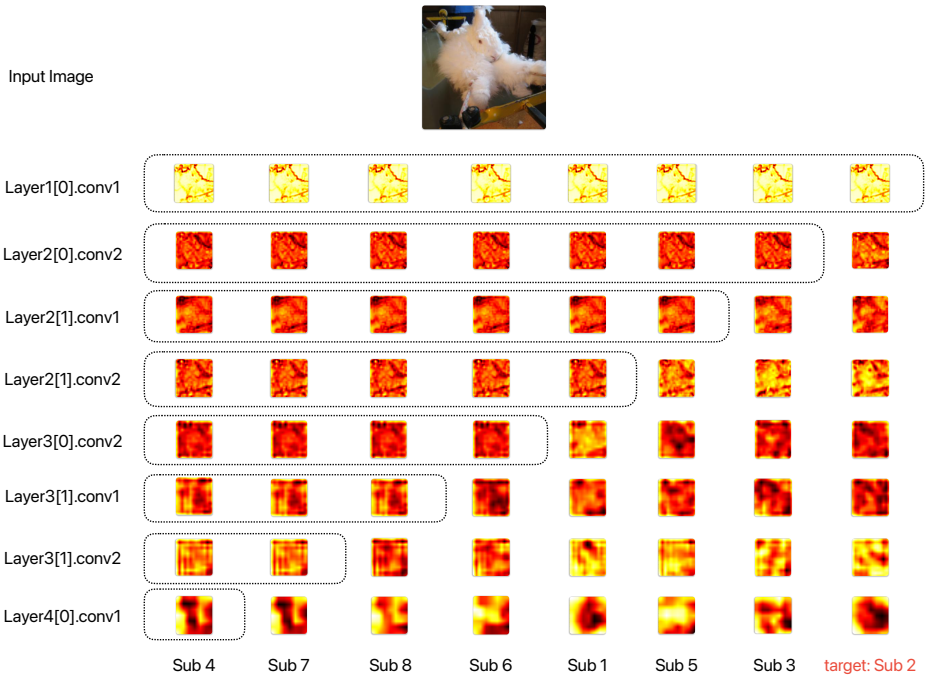

Figure 4: **Visualization of Split-Ensemble's learned features using Score-CAM.** The number of splits is set to 8 and the model is trained on ImageNet1K with ResNet-18 as backbone. The feature maps within the dashed lines across the layers indicate shared representations. The input image's class is 'Angora', targeted by submodel 2.

Model architecture visualization We visualize the Split-Ensemble models achieved under different MCT thresholds in Figure 5, as discussed in Table 4 of Section 5.3. Models here use 5 splits

and are trained on CIFAR-100 dataset with ResNet-18 as backbone. With a larger MCT threshold, the model will split into more branches at earlier layers. Meanwhile the model will also be pruned more aggressively to keep the overall computation cost unchanged. We can clearly see that with a proper MCT threshold, our method can learn a tree-like Split-Ensemble architecture with different submodels branching out at different layers, as designed by our iterative splitting and pruning algorithm in Section 4.

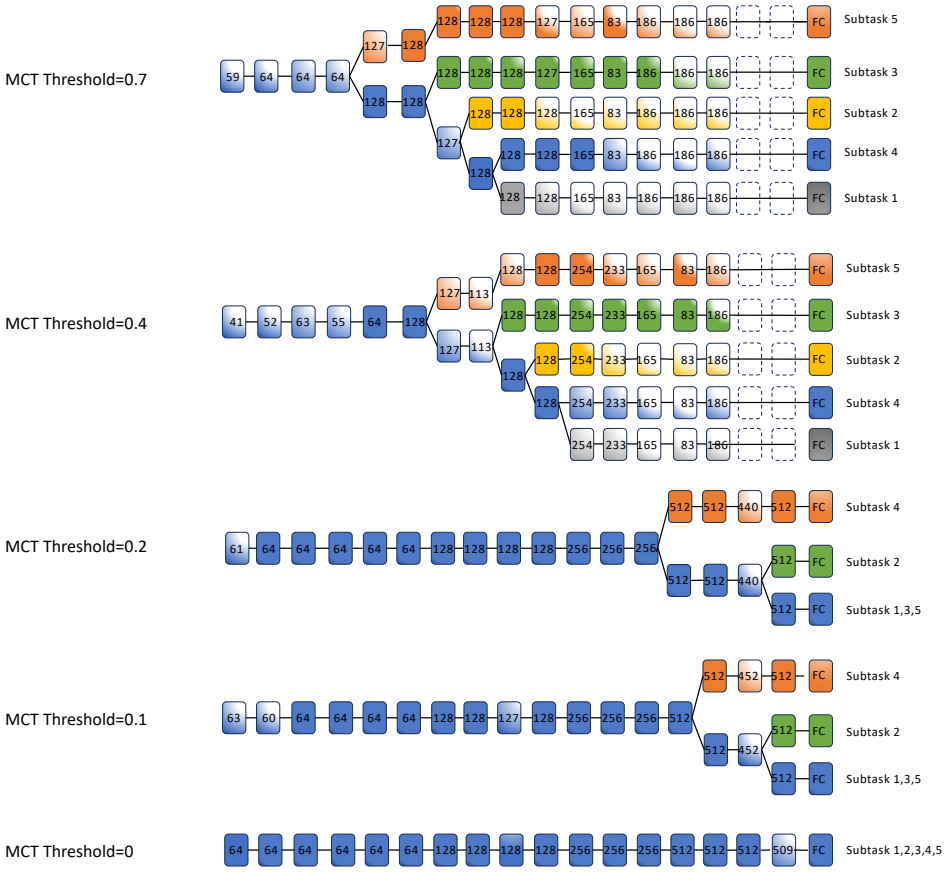

Figure 5: **Visualization of Split-Ensemble architectures under different MCT threshold.** The number of splits is set to 5. Number in each block denotes the number of filters in the layer.