HyperMix: Out-of-Distribution Detection and Classification in Few-Shot Settings

Nikhil Mehta[1,2,*] Kevin J Liang[2,*] Jing Huang[2] Fu-Jen Chu[2] Li Yin[2] Tal Hassner[2]
Duke University[1] Meta[2] Equal Contribution[*]
nikhilmehta.dce@gmail.com

Abstract

Out-of-distribution (OOD) detection is an important topic for real-world machine learning systems, but settings with limited in-distribution samples have been underexplored. Such few-shot OOD settings are challenging, as models have scarce opportunities to learn the data distribution before being tasked with identifying OOD samples. Indeed, we demonstrate that recent state-of-the-art OOD methods fail to outperform simple baselines in the few-shot setting. We thus propose a hypernetwork framework called HyperMix, using Mixup on the generated classifier parameters, as well as a natural out-of-episode outlier exposure technique that does not require an additional outlier dataset. We conduct experiments on CIFAR-FS and MiniImageNet, significantly outperforming other OOD methods in the few-shot regime.

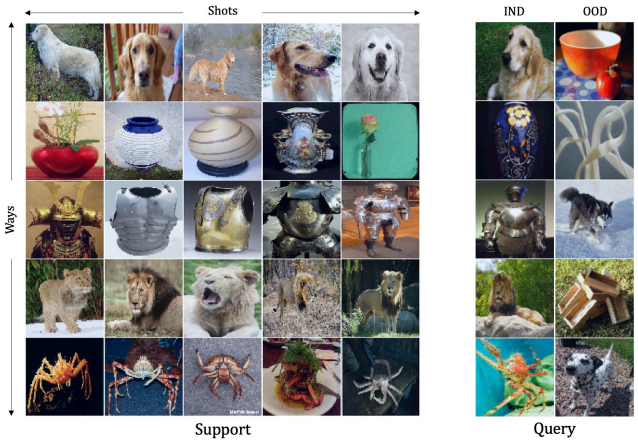

Figure 1. **Few-shot OOD detection and classification.** We show a 5-way 5-shot support set and 10 query examples, from MiniImageNet [54]. The goal is to correctly classify the IND samples (left column) while also detecting the OOD samples (right column).

1. Introduction

Out-of-Distribution (OOD) detection [20, 25, 61] has attracted much attention from the research community due to its practical implications, but almost all recent work has primarily considered OOD in standard supervised settings with a large number of samples. These many examples provide much information, *e.g.*, clear boundaries, for distinguishing In-Distribution (IND) samples from future OOD inputs. However, access to abundant IND data is not always available in many settings, necessitating algorithms that can learn from few examples. Such few-shot settings [5, 59] commonly arise where data is expensive or difficult to come by [1, 36], quick adaptation to new classes is necessary [51], or models are personalized to users [15].

Although few-shot OOD (FS-OOD) detection (Figure 1) is a natural fit for many real-world scenarios [15, 27, 66], the intersection of these two problems is relatively underexplored, and solutions are not entirely straightforward as FSL and OOD each exacerbate some of the major challenges of the other: In few-shot learning, methods are generally starved for generalizability due to the few examples to learn from [11, 35], but in an open-world setting, one must also be wary of over-generalizing, which risks misclassifying OOD samples as one of the IND classes. Conversely, for OOD detection, having a limited number of samples can handicap learning the IND data distribution, making it significantly more challenging to detect those that are OOD, especially since neither the IND classes nor OOD classes are known ahead of time during offline meta-training. As we empirically show, popular few-shot and OOD methods tend to struggle in the FS-OOD setting. In particular, we find that many recent state-of-the-art OOD methods fail to outperform the simple baseline of maximum softmax predictive probability (MSP) [25] in the few-shot setting.

To address this gap, we develop a novel framework for few-shot OOD detection that brings together strategies from few-shot learning and OOD detection approaches. In particular, we adopt a hypernetwork-based [7, 21] approach that we call HyperMix. HyperMix utilizes Mixup [64] to augment the samples in a few-shot episode in two distinct ways. While Mixup has been used to perform augmentations in the input [64] or feature [52] space, we propose Mixup in the *parameter space of hypernetwork-generated classifier*

weights. We show that this greatly improves generalization and OOD detection in few-shot settings.

In addition, previous OOD methods found that exposure to OOD samples during training can be helpful [17, 26, 65]. Such methods, however, risk overfitting to the chosen OOD set and make the assumption of a relevant OOD set being available during training. This assumption is at odds with the unpredictable nature of OOD samples in the real world [33, 61]. Instead, we show that we can naturally incorporate outlier exposure through the meta-learning process common to few-shot methods [47, 48, 54], *without an additional outlier dataset*: classes not sampled during a particular meta-training task can serve as OOD samples.

We rigorously test our proposed methods on standard FSL benchmarks CIFAR-FS [6, 29] and MiniImageNet [12, 54], but in an OOD setting. Our results clearly show that the method we propose outperforms popular OOD methods, as well as the MSP baseline, in FS-OOD settings. We will release the code to promote the reproduction of our results.

In summary, we make the following contributions:

1. We empirically show that the current state-of-the-art OOD detection methods have limited success in the few-shot learning scenario: the baseline MSP detector [25] and a related variant [38] are more reliable than other OOD methods in few-shot settings.

2. We propose a support-set augmentation for hypernetworks that mixes generated weight parameters and out-of-episode (OOE) Mixup [64] samples to supplement the query set. We name the combination of these two techniques HyperMix.

3. Through extensive experiments on CIFAR-FS and MiniImageNet, we show that our proposed HyperMix leads to improved FS-OOD detection.

2. Related Work

Out-of-Distribution (OOD) Detection. As artificial intelligence is increasingly being deployed for real-world use cases, understanding failure modes has become critical for the safety of such systems [2, 3, 37, 55]. One important axis is how such systems react when presented with unknown inputs [61]. Adversarial examples [20, 31, 50] have infamously demonstrated the ability to induce arbitrary responses in neural networks, but even untampered, normal examples can induce unexpected responses when not from the training distribution [24, 41]. While machine learning models do tend to be overconfident on OOD samples, the confidence on such samples is often lower than that of in-distribution samples and thus can be used as a simple way of identifying OOD samples [25]. Subsequent methods have sought to improve OOD detection with more sophisticated strategies, such as learning class-specific mean and covariance of embeddings and then using the Mahalanobis distance [34]. ODIN [38] uses small perturbations and temperature scaling to further separate IND and OOD score distributions. pNML [8] proposed to use a specific generalization error, the predictive normalized maximum likelihood regret, as the confidence score for OOD detection.

Outlier exposure has also been a popular technique: by showing the model OOD examples and explicitly optimizing for lower confidence, these models are often more robust to outliers. Some approaches source outliers from other datasets [13, 26, 65], while others generate them [33, 53]. However, the outlier distribution is often unknown in advance, and outliers exposed during training, whether from other datasets or synthesized, may not be representative.

Few-Shot Learning (FSL). While many deep learning advancements were enabled by large supervised datasets [12, 30], for many problems, labeled data can be scarce or expensive; being able to learn from few samples has thus been an active field of research recently. We direct interested readers to surveys [5, 59] for a more thorough treatment, highlighting several relevant works here.

Many few-shot methods are metric-based, which seek to learn an embedding such that classification can be done by comparing query and support samples in the learned feature space. A prominent example of this are Prototypical Networks [47], which meta-learns a feature extractor such that the mean of the support sample embeddings for a class can be used for Euclidean distance nearest neighbors. Mahalanobis distance [4], Earth Mover's Distance [63], and cosine similarity [10] have also been utilized. Alternatively, methods such as RelationNet [48] learn a metric for comparing samples. POODLE [32] leverages OOD samples to help the model ignore irrelevant features.

Approaches with hypernetworks [21]–auxiliary models that generate weights for the primary task model–have also been popular for FSL. Learnnet [7] proposed generating full weights of a deep discriminitive model from a single sample, scaling parameters by factorizing linear and convolutional layers. Hypernetworks have subsequently appeared in several FSL works, such as using task embeddings to influence model weights [42, 58], generating task-specific classifiers on top of a pre-trained encoder [45, 46], or synthesizing convolutional filters for object detection [43, 62].

Few-Shot OOD Detection. Few attempts have been made to solve the challenging problem of OOD detection under the few-shot constraint until recently. FS-OOD [57] introduced Out-of-Episode Classifier (OEC) that utilizes examples from the same dataset but not the current episode and showed its effectiveness in few-shot OOD detection. OOD-MAML [28] incorporated model-agnostic meta-learning (MAML) [16], a popular meta-learning approach for FSL, and proposed to synthesize OOD examples from unknown

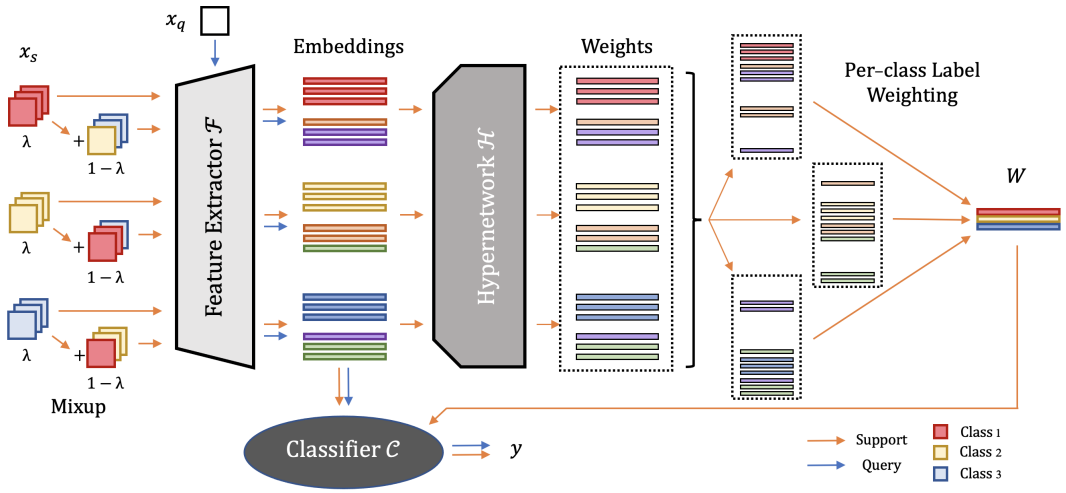

Figure 2. A diagram of the proposed ParamMix, for a 3-way 3-shot example episode. *Support samples (orange path)*: To generate the classifier, each of the KN inputs x_s (with label y_s denoted by color) in the support set are fed to the feature extractor $\mathcal{F}$; Mixup-augmented samples are also processed, resulting in $2KN$ embeddings. Each embedding is then passed to hypernetwork $\mathcal{H}$ to generate a corresponding classifier weight. The final N classifier weights W are calculated by weighted aggregation based on the samples' Mixup labels (Equation 7). *Query samples (blue path)*: To perform inference, we compose the feature extractor and classifier to produce the prediction $y = \mathcal{C}(\mathcal{F}(x_q))$, using W previously generated from the support set.

classes via gradient updating of special meta-parameters to help the model learn a sharper decision boundary. FROB [14] performed self-supervised few-shot negative data augmentation on the distribution confidence boundary and combined it with outlier exposure.

3. Background

OOD Detection methods seek to identify unfamiliar OOD inputs during inference. More precisely, OOD methods [25] commonly frame the problem as training a model on "in-distribution" data $\{x_{IND}, y_{IND}\} \sim \mathcal{D}_{IND}$, but during test, inputs are drawn from an expanded distribution $\mathcal{D}_{test} = \mathcal{D}_{IND} \cup \mathcal{D}_{OOD}$, with $\mathcal{D}_{OOD}$ being the out-of-distribution data. Generally, the OOD distribution is unknown during training, though some methods utilizing outlier exposure [26] may assume access to an auxiliary dataset of OOD samples $\hat{\mathcal{D}}_{OOD}$ during training, which may be distinct from the inference time outliers $\mathcal{D}_{OOD}$. Samples from this dataset $\hat{x} \sim \hat{\mathcal{D}}_{OOD}$ are used to augment the IND samples during training. A common label to assign such OOD samples is one of maximum entropy: $p(\hat{y}|\hat{x}) = 1/N$ for an N-way classification problem.

Few-Shot Classification is typically framed as a K-shot N-way classification problem, where the goal is to leverage $K \times N$ labeled examples from N unseen classes to learn a model $\mathcal{M}$ that can classify Q unlabeled examples from the same classes. We denote the labeled $K \times N$ examples as $\mathcal{D}_s = \{x_s, y_s\}_{s=1}^{KN}$ and the Q unlabeled examples as $\mathcal{D}_q = \{x_q\}_{q=1}^{Q}$. The labeled $\mathcal{D}_s$ and the unlabeled $\mathcal{D}_q$ examples are often referred to as support and query examples of the Few-Shot Classification (FSC) task.

To produce an effective model $\mathcal{M}^*$ for novel classes C_n, many FSC methods [47, 54] meta-train $\mathcal{M}$ on many K-shot N-way classification tasks sampled from a large labeled dataset of base classes C_b, which are assumed to be distinct from the classes in C_n, *i.e.* $C_n \cap C_b = \emptyset$. Each task sampled from the training dataset of base classes is referred to as an FSC episode, which contains a support and query set. Specifically, for episode i, $\{\mathcal{D}_s^{(i)}, \mathcal{D}_q^{(i)}\} \sim \mathcal{D}_{train}$, where $\mathcal{D}_{train}$ contains labeled examples from classes C_b, and $\{\mathcal{D}_s^{(i)}, \mathcal{D}_q^{(i)}\}$ contain labeled examples from classes $C_b^{(i)} \subset C_b$ and $|C_b^{(i)}| = N$. The optimal classifier is produced by solving the following optimization problem:

$$\mathcal{M}^* = \underset{\mathcal{M}}{\operatorname{argmin}}\, \mathbb{E}_{\{\mathcal{D}_s^{(i)}, \mathcal{D}_q^{(i)}\}} \left[\mathcal{L}_{EP} \left(\mathcal{M}(\mathcal{D}_s^{(i)}); \mathcal{D}_q^{(i)} \right) \right], \quad (1)$$

$$\text{where } \mathcal{L}_{EP} \left(\mathcal{M}(\mathcal{D}_s^{(i)}); \mathcal{D}_q^{(i)} \right) = \mathbb{E}_{D_q^{(i)}} \left[\mathcal{L} \left(\hat{y}_q^{(i)}, y_q^{(i)} \right) \right]. \quad (2)$$

Here the prediction $\hat{y}_q^{(i)} = \mathcal{M}(\mathcal{D}_s^{(i)})(x_q^{(i)})$ and $\mathcal{L}_{EP}$ corresponds to the loss for the i^{th} episode.

Mixup [64] is a data augmentation technique aiming to linearize model behaviour between samples. Given a pair of inputs $(x_1, y_1), (x_2, y_2) \sim \mathcal{D}$ drawn from the data distribution, Mixup constructs an augmented sample $(\tilde{x}_1, \tilde{y}_1)$ as a convex combinations of the two inputs and labels:

$$\tilde{x} = \lambda\, x_1 + (1 - \lambda)\, x_2 \quad (3)$$

$$\tilde{y} = \lambda\, y_1 + (1 - \lambda)\, y_2 \quad (4)$$

with $\lambda \in [0, 1]$ being a weighting factor drawn randomly; Uniform and Beta distributions are common. These augmented samples are then used as additional training data.

4. Methods

4.1. FS-OOD Classification and Detection

We formulate FS-OOD detection and classification as the intersection of the OOD detection and few-shot classification problems. Given an N-way K-shot support set $\mathcal{D}_s = \{x_s, y_s\}_{s=1}^{KN}$ (in-distribution data $\mathcal{D}_{IND}$) and query samples $\mathcal{D}_q = \{x_q\}_{q=1}^{Q}$, where x_q is drawn from either the N classes or the OOD distribution $\mathcal{D}_{OOD}$, the goal is to classify each query x_q as one of the N classes or as OOD.

As FSC aims to generalize to new tasks during meta-test, the FS-OOD detection problem poses a notable challenge over standard OOD detection in that the IND data distribution $\mathcal{D}_{IND}$ is unknown during meta-training, in addition to the usually unknown OOD distribution $\mathcal{D}_{OOD}$. Instead, a model that can quickly adapt to a new IND distribution given a few samples must be learned.

4.2. Hypernetwork Framework

Because of their strong adaptation capabilities [18, 45], we adopt a hypernetwork [21] framework for FS-OOD: We instantiate model $\mathcal{M}$ as a feature extractor $\mathcal{F}$ and a linear classifier $\mathcal{C}$ parameterized by W. Following previous works [10], we first pre-train $\mathcal{F}$ on the dataset of base classes $\mathcal{D}_b$. Rather than directly learning W by gradient descent, we instead learn a hypernetwork $\mathcal{H}$ that generates the parameters for $\mathcal{C}$ from the support set. A hypernetwork-based model allows us to leverage the *learning to learn* paradigm, training a universal architecture model which can be quickly adapted to a new task by generating task-specific weights. Unlike fine-tuning and second-order procedures (inner and outer loop), hypernetworks are "training-free" when producing a classifier for a new task, resulting in faster adaptation and lower computational requirements.

Meta-training of $\mathcal{H}$. The hypernetwork $\mathcal{H}$ is trained on many classification tasks sampled from the dataset of base classes C_b in the meta-training stage. This meta-training is done in an episodic fashion by drawing M few-shot tasks from $\mathcal{D}_{train}$. For each episode i, the hypernetwork takes the support embeddings extracted using $\mathcal{F}$ and generates sample-specific weight codes $\mathcal{H}(\mathcal{F}(x_s^{(i)}))$, which are used to generate the classifier weights as follows:

$$w_n = \frac{1}{K} \sum_{s=1}^{KN} \mathbf{1}_{[y_s=n]} \mathcal{H}\left(\mathcal{F}(x_s^{(i)})\right). \tag{5}$$

Let the classifier weights derived from the support set of episode i be $W(\mathcal{D}_s^{(i)}) = [w_1, \ldots, w_n]$. The class-logits for queries are predicted as $\hat{y}_q = \mathcal{C}\left(\mathcal{F}(x_q) | W(\mathcal{D}_s^{(i)})\right)$. The loss function at the meta-training stage is as follows:

$$\mathcal{H}^* = \underset{\mathcal{H}}{\operatorname{argmin}}\ \mathbb{E}_{\{\mathcal{D}_s^{(i)}, \mathcal{D}_q^{(i)}\}} \left[\mathcal{L}_{EP}\left(W(\mathcal{D}_s^{(i)}); \mathcal{D}_q^{(i)}\right)\right], \tag{6}$$

$$\text{where } \mathcal{L}_{EP}\left(W(\mathcal{D}_s^{(i)}); \mathcal{D}_q^{(i)}\right) = \mathbb{E}_{\mathcal{D}_q^{(i)}}\left[\mathcal{L}_{CCE}\left(\hat{y}_q^{(i)}, y_q^{(i)}\right)\right]$$

$$\text{and } \mathcal{L}_{CCE}\left(\hat{y}_q^{(i)}, y_q^{(i)}\right) = -\, y_q^{(i)} \cdot \log\left(\hat{y}_q^{(i)}\right)$$

is the Categorical Cross Entropy (CCE) loss. Note, the loss gradient in (6) can also be backpropagated into $\mathcal{F}$ to fine-tune the feature extractor, providing some improvement.

OOD Detection during Inference. After meta-training the model, we use the maximum softmax probability [25] as the score for the OOD detection task during inference. In particular, we define the score $s(x_q)$ for a given query x_q at meta-testing as $s(x_q) = \max_c [\hat{y}_{q,c}]$, where $\hat{y}_{q,c}$ is the c^{th} component of the softmax vector $\hat{y}_q = \mathcal{C}(\mathcal{F}(x_q) | W)$. If $s(x_q) < \tau$, with τ being the OOD threshold, we call the query an OOD sample. However, note that the OOD detection task is evaluated using metrics that alleviate the need to set a fixed threshold τ (see Section 5.2).

4.3. HyperMix

We find our hypernetwork framework for FS-OOD outperforms other common few-shot approaches, but such a set-up alone does not incorporate any OOD detection capabilities. We thus propose HyperMix: a support and query set augmentation technique based on Mixup [64] for training the hypernetwork during the meta-training stage. In contrast to the existing Mixup data augmentation technique, which uses linear Mixup of the model inputs and target labels, HyperMix uses Mixup in two novel ways uniquely adapted to our hypernetwork framework and FS-OOD settings, which we call ParamMix and OOE-Mix.

ParamMix. ParamMix augments the support set using Mixup and uses weighted aggregation to compute the classifier weights. Let the mixed support samples and the corresponding label be denoted as $\tilde{x}_s$ and $\tilde{y}_s$ respectively. The mixing parameter λ_{PM} is sampled from a fixed Beta distribution, and the extracted parameters of the classifiers are calculated using weighted aggregation of sample-specific weight codes, where the aggregation weights are determined based on the probabilities of mixed samples. ParamMix modifies the weight generation in (5) for soft labels:

$$w_n = \frac{1}{\sum_{s=1}^{S} \tilde{y}_n^{(s)}} \sum_{s=1}^{S} \tilde{y}_n^{(s)}\ \mathcal{H}(\mathcal{F}(\tilde{x}_s)). \tag{7}$$

where $\tilde{x}_s = \lambda_{PM}\ x_s^{(1)} + (1-\lambda_{PM})\ x_s^{(2)}$ is a mixed sample formed from two support set images, $\tilde{y}_n^{(s)}$ corresponds to the probability of $\tilde{x}_s$ belonging to the label n, and S is the total number of support samples after augmentation with Mixup. We summarize ParamMix in Figure 2.

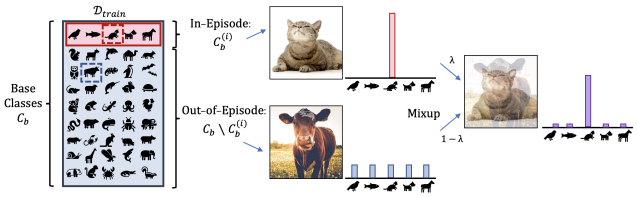

Figure 3. Illustration of OOE-Mix augmentation process. For every few-shot episode during meta-training, N classes $C_b^{(i)}$ are sampled from the base classes C_b. The remaining out-of-episode (OOE) classes can be used as a source for realistic near OOD examples. We mix these OOE samples with the sampled in-episode (INE) examples to improve generalization.

OOE-Mix. Outlier exposure [17, 26, 65] has been shown to be highly effective for OOD detection, but it often assumes access to a dataset of outliers, which is often practical. On the other hand, we note that the meta-learning process of sampling a subset of the base classes $C_b^{(i)}$ to construct a few-shot episode means there are classes $C_b \setminus C_b^{(i)}$ available that are not part of the N-way classification for episode i. We refer to the samples from $C_b^{(i)}$ as in-episode (INE) and samples from $C_b \setminus C_b^{(i)}$ as out-of-episode (OOE). As the OOE samples are from the same dataset, not only are these samples representative and of the same domain as the INE data, but the mutually exclusive nature of the labels means that they are also not in-distribution. We can thus consider the ample samples from $C_b \setminus C_b^{(i)}$ as realistic "near" outliers [60] for this particular few-shot episode.

While we can expose OOE samples during meta-training as is, we find mixing INE and OOE samples to be highly effective. Therefore, we propose OOE-Mix to augment the query set with mixed INE and OOE samples (Figure 3). Specifically, we sample INE and OOE samples as:

$$x_1^{(i)} \sim \mathcal{D}_{INE}^{(i)}; \quad x_2^{(i)} \sim \mathcal{D}_{OOE}^{(i)}; \quad p(y|x_2^{(i)}) = 1/N \tag{8}$$

As we consider the OOE samples to be OOD, we assign OOE samples with a uniform distribution label, to enforce maximal uncertainty. Similar to Equations 3 and 4, we then apply Mixup, but specifically between pairs of INE and OOE samples:

$$\tilde{x}^{(i)} = \lambda_{OM}\, x_1^{(i)} + (1 - \lambda_{OM})\, x_2^{(i)} \tag{9}$$

$$p(y|\tilde{x}^{(i)}) = \lambda_{OM}\, p(y|x_1^{(i)}) + (1 - \lambda_{OM})\, p(y|x_2^{(i)}) \tag{10}$$

with $\lambda_{OM} \sim \text{Beta}(a_{OM}, b_{OM})$. We augment the query dataset with the generated mixed samples while optimizing the loss function in Equation (6). We find this helpful as the addition of Mixup between INE and OOE samples encourages smooth behaviour in the classifier's decision space. This helps with generalization, a critical component for few-shot settings.

5. Experiments

5.1. Datasets

We run experiments on popular few-shot learning datasets CIFAR-FS [6] and MiniImageNet [54], but we adapt them for few-shot OOD detection. In particular, for each meta-testing task, we include query images from the unsampled meta-test classes (those not in the N-way task) as OOD inputs (see Evaluation Protocol in Section 5.2).

CIFAR-FS consists of 32×32 color images from the CIFAR-100 dataset [29], split into 64, 16, and 20 classes for meta-training, meta-validation, and meta-testing, with 600 images per class.

MiniImageNet [54] is a common few-shot benchmark consisting of 84×84 color images, with 100 classes of 600 examples each. Similar to CIFAR-FS, we use 64 classes for meta-training, 16 for validation, and 20 for meta-testing.

5.2. Set-up

Our experimental settings closely resemble the traditional few-shot classification benchmarks, with the primary difference being that half the meta-test query samples are drawn from outside of the N-way classification problem, *i.e.* OOD samples. We use 5-shot and 10-shot settings during the meta-training phase. All experiments were done on a GeForce GTX 1080 GPU with 12GB memory.

Implementation of $\mathcal{F}$. We implement the feature extractor $\mathcal{F}$ as a ResNet-12 [23], which has 4 residual blocks with 3 convolutional layers of 64, 160, 320, and 640 3×3 kernels. The first three blocks are followed by 2×2 average pooling, resulting in features of size 640. Inspired by recent self-supervised learning methods for few-shot classification [19, 39, 44], we train $\mathcal{F}$ using a combination of supervised and self-supervised criteria. In particular, we follow the strategy of [44], where the image-label pair $\{x, y\}$ is augmented to produce $\{x(r), y, r\}$, where $x(r)$ is the image x rotated by r degrees and $r \in \{0, 90, 180, 270\}$. In the pre-training stage, a stack of two linear classifiers are learned on top of $\mathcal{F}$, where the first classifier predicts the class logits $\hat{y}$ for the $|C_b|$-way classification task, and the second classifier predicts the rotation logits $\hat{r}$. The model is trained using $\mathcal{L}_{CCE}\left(\text{softmax}(\hat{y}), y\right) + \mathcal{L}_{BCE}\left(\text{sigmoid}(\hat{r}), r\right)$, where CCE and BCE correspond to the Categorical and Binary Cross-Entropy loss. After convergence, the two linear classifiers are discarded [9], and $\mathcal{F}$ is used to extract image embeddings in the meta-training stage. During pre-training of $\mathcal{F}$, the batch size was set to 64. Each image in the batch was rotated three times at 90, 180, and 270 degrees, resulting in an effective batch size of 256. We train $\mathcal{F}$ for 200 epochs using Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum [49] of 0.9 and weight decay of 5e-4. We start with an initial learning rate of 0.1 and decay by 0.2 at 60, 120, and 160 epochs.

Table 1. **FS-OOD Classification and Detection.** *OE is the outlier exposure method and thus requires auxiliary data, which we do not assume in our experiments; we instead use OOE sampling as the auxiliary outliers. Best value shown in **bold**; second best shown in *italics*.

		5-shot 5-way			10-shot 5-way		
	Method	IND Acc ↑	AUROC ↑	FPR@90 ↓	IND Acc ↑	AUROC ↑	FPR@90 ↓
		FS-CIFAR-100 [6]					
	Fine-tune w/ MSP [25]	80.38 ± 0.41	72.00 ± 0.40	75.20 ± 0.68	**85.04 ± 0.72**	74.30 ± 0.81	71.25 ± 1.43
ProtoNet	MSP [25]	80.09 ± 0.42	75.42 ± 0.37	66.42 ± 0.72	83.56 ± 0.76	75.55 ± 0.75	67.72 ± 1.46
ProtoNet	ODIN [38]	80.09 ± 0.42	75.31 ± 0.37	66.69 ± 0.72	83.56 ± 0.76	75.39 ± 0.75	67.90 ± 1.44
ProtoNet	DM [34]	80.09 ± 0.42	66.24 ± 1.14	77.43 ± 1.47	83.56 ± 0.76	68.03 ± 0.71	79.69 ± 1.20
ProtoNet	OEC [56]	76.45 ± 0.93	74.32 ± 0.94	67.84 ± 0.81	80.86 ± 0.78	74.40 ± 0.77	66.00 ± 1.48
ProtoNet	OE* [26]	78.31 ± 0.41	73.13 ± 0.45	68.61 ± 0.94	81.01 ± 0.82	74.14 ± 0.91	68.68 ± 1.01
Hypernetwork	MSP [25]	78.89 ± 0.45	78.38 ± 0.39	57.65 ± 0.78	81.58 ± 0.41	81.42 ± 0.35	54.19 ± 0.79
Hypernetwork	Entropy [40]	78.89 ± 0.45	69.67 ± 0.43	70.33 ± 0.70	81.58 ± 0.41	71.92 ± 0.41	69.02 ± 0.71
Hypernetwork	ODIN [38]	78.89 ± 0.45	78.73 ± 0.38	56.42 ± 0.78	81.58 ± 0.41	81.83 ± 0.35	52.95 ± 0.79
Hypernetwork	DM [34]	78.89 ± 0.45	63.67 ± 0.22	80.06 ± 0.29	81.58 ± 0.41	67.48 ± 0.18	76.23 ± 0.27
Hypernetwork	pNML [8]	78.89 ± 0.45	60.41 ± 0.49	78.53 ± 0.65	81.58 ± 0.41	66.43 ± 0.42	72.56 ± 0.46
Hypernetwork	OEC [56]	77.51 ± 0.44	79.11 ± 0.34	56.86 ± 0.79	80.61 ± 0.61	82.06 ± 0.41	53.19 ± 0.74
Hypernetwork	OE* [26]	79.46 ± 0.42	*81.68 ± 0.35*	*55.21 ± 0.80*	81.69 ± 0.54	83.14 ± 0.33	52.16 ± 0.79
Hypernetwork	ParamMix (ours)	78.45 ± 0.49	79.24 ± 0.37	57.12 ± 0.77	81.22 ± 0.42	82.23 ± 0.35	52.33 ± 0.78
Hypernetwork	OOE-Mix (ours)	*80.88 ± 0.42*	81.11 ± 0.35	56.38 ± 0.80	83.46 ± 0.40	**84.08 ± 0.32**	*50.97 ± 0.81*
Hypernetwork	HyperMix (ours)	**81.33 ± 0.41**	**82.43 ± 0.33**	**53.69 ± 0.79**	*83.79 ± 0.38*	*83.20 ± 0.34*	**50.66 ± 0.80**
		MiniImageNet [54]					
Hypernetwork	MSP [25]	72.13 ± 0.47	72.70 ± 0.41	69.97 ± 0.73	75.15 ± 0.44	75.24 ± 0.39	66.40 ± 0.75
Hypernetwork	Entropy [40]	72.13 ± 0.47	62.73 ± 0.43	79.64 ± 0.59	75.15 ± 0.44	71.92 ± 0.41	69.02 ± 0.71
Hypernetwork	ODIN [38]	72.13 ± 0.47	72.86 ± 0.41	68.84 ± 0.73	75.15 ± 0.44	75.45 ± 0.38	65.94 ± 0.75
Hypernetwork	DM [34]	72.13 ± 0.47	61.00 ± 0.18	83.71 ± 0.22	75.15 ± 0.44	63.58 ± 0.18	81.71 ± 0.24
Hypernetwork	pNML [8]	72.13 ± 0.47	56.32 ± 0.34	85.94 ± 0.46	75.15 ± 0.44	62.11 ± 0.42	83.15 ± 0.59
Hypernetwork	OEC [56]	71.11 ± 0.51	72.81 ± 0.36	68.61 ± 0.71	73.94 ± 0.40	75.01 ± 0.44	64.94 ± 0.71
Hypernetwork	OE* [26]	72.64 ± 0.44	73.81 ± 0.39	68.78 ± 0.73	75.81 ± 0.43	74.84 ± 0.40	65.46 ± 0.74
Hypernetwork	ParamMix (ours)	71.52 ± 0.45	73.87 ± 0.40	69.14 ± 0.73	74.99 ± 0.44	75.46 ± 0.39	65.69 ± 0.75
Hypernetwork	OOE-Mix (ours)	*73.95 ± 0.45*	*74.75 ± 0.39*	*66.62 ± 0.74*	**78.25 ± 0.41**	*76.66 ± 0.38*	*64.55 ± 0.77*
Hypernetwork	HyperMix (ours)	**74.38 ± 0.41**	**75.03 ± 0.43**	**65.57 ± 0.75**	*77.81 ± 0.42*	**77.50 ± 0.38**	**63.15 ± 0.76**

Implementation of $\mathcal{H}$. We parameterize the hypernetwork as a multi-layered perceptron, containing two fully-connected layers of size 256. We generate the classifier weights using the procedure described in Equations 5 and 7. Note that while the number of parameters generated in Equation 5 corresponds to the number of classes N, in practice, we generate $N+1$ parameters and split them into classifier weights and biases. We meta-train $\mathcal{H}$ for 50 epochs with SGD optimizer having a fixed learning rate of 0.001. Each epoch had 200 batches, where each batch contained upto 4 episodes/tasks. The input images are augmented using random crop, color jitter, and horizontal flip.

Evaluation Protocol. We sample 400 episodes randomly from the meta-test set containing novel classes C_n that the model has not previously encountered. In each episode, the model is given a labeled support set of KN samples that is referred to as the IND samples, and a set of 20 unlabeled queries. Of the 20 queries, 10 belong to the N IND classes and the remaining 10 are OOD. In our experiments, we consider the *near*-OOD detection task, which is considered a harder problem compared to *far*-OOD detection [60]. Unlike *far*-OOD detection, which uses OOD samples from a different dataset for evaluation, in *near*-OOD detection, we reserve classes from the same dataset as OOD during evaluation. In our experiments, we consider the classes outside a given episode as OOD during evaluation. Note that both IND and OOD samples belong to classes that are not known a priori to the model. We evaluate the model on the few-shot N-way classification task for IND queries, and the binary OOD detection task. We use classification accuracy to measure the few-shot classification performance, and AUROC and FPR@90 to measure the OOD detection performance.

5.3. Baselines

Few-shot OOD detection and classification is an under-explored problem, meaning there are relatively few previous works. Standard few-shot methods lack a mechanism to reject outlier samples [36], which leads to misclassifications for all OOD samples. Thus, we instead primarily compare against several recently proposed OOD methods, with restricted numbers (shots) of IND samples:

- Maximum Softmax predictive Probability (MSP) [25]:

MSP uses maximum probability from the model as the IND confidence score.

- Entropy [40]: It uses the predictive entropy of the output probabilities as the OOD score.
- ODIN [38]: ODIN also uses MSP, however, it employs temperature scaling and input perturbation based on the gradient of the loss function to increase the margin between maximum probability obtained for IND and OOD samples.
- Deep Mahalanobis (DM) [34]: DM calculates the class-conditional mean and covariance for each class to calculate the confidence score of query samples at each layer of the encoder. Scores are computed by a logistic regression model learned with validation data.
- Predictive Normalized Maximum Likelihood (pNML) [8]: pNML uses a generalization error based on predictive normalized maximum likelihood regret as the OOD score.
- Outlier Exposure (OE) [26]: OE approaches expose OOD samples to the model, trained with a uniform distribution label in the meta-training stage. Note that this assumes the availability of an auxiliary outlier dataset, which we do not use in our experiments. Instead, we show OE with our proposed strategy of leveraging OOE samples from $C_b \setminus C_b^{(i)}$.
- Out-of-Episode Classifier (OEC) [56]: OEC is a recently proposed method for FS-OOD detection, which uses binary cross-entropy objective to learn an IND/OOD binary classifier. Similar to OE, we provide OOE samples to OEC during meta-training.

We primarily perform our comparisons using the hypernetwork implementation described in Section 5.2, but we also perform experiments with some of the baselines using fine-tuning [10] and ProtoNet [47] as the few-shot learning algorithm, to demonstrate the efficacy of our hypernetwork. We further describe how we adapted the above methods to the FS-OOD setting in Appendix A. In addition to our proposed HyperMix, we also perform ablations, analyzing the two components of our approach individually:

- ParamMix: We apply Mixup to the parameters W generated by hypernetwork $\mathcal{H}$ as described in Section 4.3, but without OOE samples during meta-training.
- OOE-Mix: Similar to OE [26], we provide OOE samples to the model from the unsampled base classes, and additionally apply Mixup between pairs of INE and OOE samples as described in Section 4.3.

Note that our proposed HyperMix is the combination of ParamMix and OOE-Mix. The hyperparameters for all the methods are chosen based on grid search done on validation dataset. The chosen hyperparameters and candidates are discussed in Appendix B.

Table 2. ParamMix, OOE-Mix, and Hyper-Mix on mixed samples on CIFAR-FS. Best values shown in **bold**; second best in *italics*.

Method	IND Acc ↑	AUROC ↑	FPR@90 ↓
	Test OOD Dataset: INE-OOE Mixture		
ParamMix	80.44 ± 0.86	80.61 ± 0.73	55.94 ± 1.58
OOE-Mix	*84.24* ± 0.47	**83.47** ± 0.65	*52.60* ± 1.61
HyperMix	**84.51** ± 0.78	*83.43* ± 0.66	**50.59** ± 1.60
	Test OOD Dataset: INE-INE Mixture		
ParamMix	81.61 ± 0.86	**65.93** ± 0.65	**83.03** ± 1.04
OOE-Mix	*84.13* ± 0.76	61.03 ± 0.61	87.86 ± 0.84
HyperMix	**84.46** ± 0.75	*62.00* ± 0.62	*86.41* ± 0.89

5.4. Few-shot In-Distribution Classification and OOD Detection

We report FS-OOD performance on CIFAR-FS [6] and MiniImageNet [54] in Table 1. We quantify OOD detection performance with area under the receiver operating characteristic curve (AUROC) and false positive rate at 90% recall rate (FPR@90); we simultaneously also measure classification accuracy of in-distribution samples (IND Acc). Note that many of the baselines are post-hoc methods on the same meta-trained model and thus have the same IND accuracy.

Choice of Few-shot Framework. Our proposed hypernetwork framework for FS-OOD outperforms more basic approaches like fine-tuning or ProtoNet. For example, for the MSP baseline, a hypernetwork approach improves AUROC and FPR@90 by almost 3% and 9%, respectively, compared to ProtoNet on 5-shot 5-way classification on FS-CIFAR-100; this grows to a 6.38% and 17.55% improvement in FS-CIFAR-100 AUROC and FPR@90 compared to MSP with fine-tuning. Similar improvements to OOD detection can be seen in the 10-shot 5-way case.

Performance of Popular OOD Detection Methods. We observe that several recent OOD detection methods underperform the simple baseline MSP in the FS-OOD setting. In particular, we found that popular OOD methods such as DM and pNML had limited success in detecting OOD samples; DM for example has almost 15% worse AUROC and almost 23% worse FPR@90 compared to MSP, which stands in sharp contrast to its large improvements in the many-shot setting. We posit that this is due to the limited number of support samples per episode, leading to the empirical covariance/data matrix used in these approaches to be degenerate. We indeed find this to be the case when comparing the singular values of the covariance matrix used in DM for few-shot and many-shot settings (see Appendix C). This limits the generalization ability of these methods in differentiating IND from OOD samples.

HyperMix. In contrast, we observe that our proposed HyperMix performs the best. Using HyperMix at meta-training, the model not only does better on the OOD detection task, but also leads to improved IND accuracy compared to the baselines with hypernetworks. We see that the

Table 3. **Noisy FS-OOD Detection (10-shot 5-way).** HyperMix outperforms the baseline methods even in noisy support set settings.

	10% Noise		20% Noise		30% Noise		40% Noise	
Method	AUROC ↑	FPR@90 ↓	AUROC ↑	FPR@90 ↓	AUROC ↑	FPR@90 ↓	AUROC ↑	FPR@90 ↓
	FS-CIFAR-100 [6]							
MSP [25]	79.93 ± 0.36	57.15 ± 0.79	77.35 ± 0.38	62.28 ± 0.78	72.17 ± 0.42	70.11 ± 0.73	63.88 ± 0.47	78.42 ± 0.62
Entropy [40]	71.61 ± 0.83	69.16 ± 1.40	70.68 ± 0.41	71.92 ± 0.70	67.93 ± 0.43	76.56 ± 0.66	63.28 ± 0.45	81.49 ± 0.59
ODIN [38]	80.33 ± 0.36	55.69 ± 0.78	77.79 ± 0.38	60.88 ± 0.77	72.56 ± 0.42	68.98 ± 0.73	64.08 ± 0.46	77.73 ± 0.63
DM [34]	63.52 ± 0.18	80.92 ± 0.24	60.33 ± 0.18	83.63 ± 0.22	57.40 ± 0.18	85.60 ± 0.20	54.79 ± 0.18	87.11 ± 0.19
pNML [8]	62.77 ± 0.91	74.40 ± 1.09	60.64 ± 0.93	75.56 ± 1.12	59.81 ± 0.92	78.61 ± 1.04	57.12 ± 0.92	80.04 ± 0.99
OE [26]	81.60 ± 0.35	56.71 ± 0.80	78.46 ± 0.37	63.59 ± 0.77	72.70 ± 0.41	71.95 ± 0.70	64.55 ± 0.45	79.39 ± 0.60
OEC [56]	81.05 ± 0.74	57.68 ± 0.83	76.19 ± 0.51	64.17 ± 0.94	71.11 ± 0.78	74.46 ± 1.05	61.97 ± 0.86	80.09 ± 1.44
ParamMix	81.02 ± 0.36	55.28 ± 0.78	78.74 ± 0.37	60.16 ± 0.78	73.80 ± 0.40	67.94 ± 0.73	65.10 ± 0.44	77.32 ± 0.61
HyperMix	**81.94 ± 0.34**	**53.67 ± 0.79**	**79.69 ± 0.36**	**58.80 ± 0.80**	**74.87 ± 0.40**	**67.00 ± 0.76**	**66.36 ± 0.45**	**76.46 ± 0.64**
	MiniImageNet [54]							
MSP [25]	73.75 ± 0.39	68.16 ± 0.74	71.04 ± 0.41	71.97 ± 0.71	66.77 ± 0.43	77.01 ± 0.65	60.13 ± 0.45	82.80 ± 0.56
Entropy [40]	63.03 ± 0.42	79.87 ± 0.57	62.12 ± 0.41	81.38 ± 0.56	60.32 ± 0.42	83.28 ± 0.53	57.38 ± 0.44	85.58 ± 0.50
ODIN [38]	73.99 ± 0.39	67.80 ± 0.74	71.23 ± 0.41	71.62 ± 0.71	66.87 ± 0.43	76.81 ± 0.65	60.19 ± 0.45	82.78 ± 0.56
DM [34]	60.40 ± 0.18	84.69 ± 0.21	57.71 ± 0.17	86.61 ± 0.20	55.41 ± 0.17	87.80 ± 0.18	53.32 ± 0.17	88.77 ± 0.17
pNML [8]	59.78 ± 0.43	84.03 ± 0.47	58.60 ± 0.49	85.08 +- 0.59	56.79 ± 0.89	86.31 ± 0.71	53.43 ± 0.98	87.21 ± 0.85
OE [26]	73.09 ± 0.40	67.90 ± 0.73	70.30 ± 0.42	72.31 ± 0.70	65.87 ± 0.44	77.39 ± 0.64	59.64 ± 0.47	82.69 ± 0.56
OEC [56]	73.61 ± 0.52	68.08 ± 0.79	69.94 ± 0.53	71.59 ± 0.84	65.46 ± 0.71	78.46 ± 0.85	60.22 ± 0.76	82.17 ± 0.81
ParamMix	73.46 ± 0.41	67.47 ± 0.73	70.88 ± 0.43	71.17 ± 0.69	66.03 ± 0.48	76.85 ± 0.63	59.83 ± 0.43	81.80 ± 0.59
HyperMix	**76.02 ± 0.37**	**65.22 ± 0.73**	**74.09 ± 0.38**	**70.61 ± 0.70**	**69.67 ± 0.40**	**75.03 ± 0.63**	**62.55 ± 0.43**	**81.80 ± 0.55**

combination of both ParamMix and OOE-Mix is important. While each individually is comparable or does better than the baselines, the combination of the two significantly outperforms the baseline methods.

Why is the combination of ParamMix and OOE-Mix especially effective? We hypothesize that each makes HyperMix robust to certain types of outliers: ParamMix smooths the inter-support class decision boundary, while OOE-Mix regularizes behavior on outside classes. We verify this by testing ParamMix, OOE-Mix, and HyperMix on outliers created by mixing inlier class samples (INE-INE) and inliers with OOE samples (INE-OOE) in Table 2; we indeed observe ParamMix and OOE-Mix have different tendencies, and that HyperMix tends to combine their strengths.

5.5. Noisy Labels

Even in few-shot settings, outliers can have noisy labels and could be present in the support sets, and such mislabeled samples can have a significant impact on performance [36]. We thus also evaluate all the models in the FS-OOD settings when noisy labels are present during meta-testing, reporting OOD detection results in Table 3 and few-shot accuracy in Figure 4. We restrict our experiments to hypernetwork-based few-shot learning, as it saw the strongest OOD detection performance in Table 1. We experiment with varying levels of noise present in the support set ranging from 10% to to 40% of the total support samples. For each evaluation scenario, we randomly select IND samples ($\{x, y\}$) from the support set and replace them with OOD samples ($\{\hat{x}, y\}$). To make this scenario especially challenging, we pair each IND class with an OOE class, to model real-world classes being similar [22]. As expected, we observe that all methods, including ours, see worse performance with higher noise levels. On the other hand, we observe that HyperMix maintains a performance improvement gap even up to 40% support set noise.

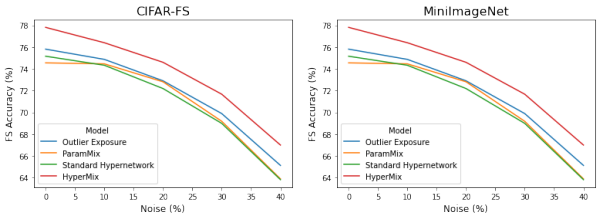

Figure 4. FS accuracy on FS-CIFAR-100 (left) and MiniImageNet (right) with varying levels of label noise in the support set.

6. Conclusion

Standard few-shot methods are unequipped to deal with OOD samples during test time and will tend to misclassify them as one of the IND classes by default. At the same time, while recently proposed methods have led to improvements in OOD detection in settings with abundant in-distribution data, we find that many of them struggle in few-shot settings, failing to outperform the simple MSP [25] baseline. To address this gap, we show that a hypernetwork-based framework for FS-OOD outperforms ProtoNet or fine-tuning based approaches. We further show that exposing the model to out-of-episode (OOE) outliers mixed with the query set during meta-training helps prepare the model for test OOD samples, and mixup of hypernetwork weights generated from the support set further helps the model generalize. This combination of augmentation techniques, which we call HyperMix, strongly outperforms baseline methods in FS-OOD experiments on FS-CIFAR-100 and MiniImageNet in a variety of settings.

References

[1] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 2017. 1

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. 2

[3] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 2021. 2

[4] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *CVPR*, 2020. 2

[5] Nihar Bendre, Hugo Terashima Marín, and Peyman Najafirad. Learning from few samples: A survey. *arXiv preprint arXiv:2007.15484*, 2020. 1, 2

[6] Luca Bertinetto, João F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *ICLR*, 2019. 2, 5, 6, 7, 8, 13

[7] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, 2016. 1, 2

[8] Koby Bibas, Meir Feder, and Tal Hassner. Single layer predictive normalized maximum likelihood for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 6, 7, 8, 12

[9] Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, and Pascal Vincent. Guillotine regularization: Improving deep networks generalization by removing their head. *arXiv preprint arXiv:2206.13378*, 2022. 5

[10] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019. 2, 4, 7

[11] Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-level semantic feature augmentation for one-shot learning. *IEEE Transactions on Image Processing*, 2019. 1

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, 2009. 2

[13] Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. Reducing network agnostophobia. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[14] Nikolaos Dionelis, Mehrdad Yaghoobi, and Sotirios A. Tsaftaris. FROB: Few-shot ROBust model for classification with out-of-distribution detection, 2022. 3

[15] Kristen Grauman et al. Ego4D: Around the World in 3,000 Hours of Egocentric Video. *CVPR*, 2022. 1

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 2017. 2

[17] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. Exploring the limits of out-of-distribution detection. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 5

[18] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 2020. 4

[19] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8058–8067, 2019. 5

[20] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[21] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *ICLR*, 2017. 1, 2, 4

[22] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 2018. 8

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 5

[24] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019. 2

[25] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017. 1, 2, 3, 4, 6, 8, 12, 13

[26] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018. 2, 3, 5, 6, 7, 8, 12, 13

[27] Nathan A Inkawhich, Eric K Davis, Matthew J Inkawhich, Uttam K Majumder, and Yiran Chen. Training sar-atr models for reliable operation in open-world environments. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2021. 1

[28] Taewon Jeong and Heeyoung Kim. Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification. In *Advances in Neural Information Processing Systems*, 2020. 2

[29] Alex Krizhevsky et al. Learning multiple layers of features from tiny images, 2009. 2, 5

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012. 2

[31] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, 2018. 2

[32] Duong Le, Khoi Duc Nguyen, Khoi Nguyen, Quoc-Huy Tran, Rang Nguyen, and Binh-Son Hua. Poodle: Improving few-shot learning via penalizing out-of-distribution samples.

Advances in Neural Information Processing Systems, 2021. 2

[33] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *International Conference on Learning Representations*, 2018. 2

[34] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in neural information processing systems*, 2018. 2, 6, 7, 8, 12, 13

[35] Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1

[36] Kevin J Liang, Samrudhdhi B Rangrej, Vladan Petrovic, and Tal Hassner. Few-shot learning with noisy labels. In *CVPR*, 2022. 1, 6, 8

[37] Kevin J Liang, John B Sigman, Gregory P Spell, Dan Strellis, William Chang, Felix Liu, Tejas Mehta, and Lawrence Carin. Toward automatic threat recognition for airport x-ray baggage screening with deep convolutional object detection. *arXiv preprint arXiv:1912.06329*, 2019. 2

[38] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. 2, 6, 7, 8, 12, 13

[39] Puneet Mangla, Mayank Kumar Singh, Abhishek Sinha, Nupur Kumari, Vineeth N. Balasubramanian, and Balaji Krishnamurthy. Charting the right manifold: Manifold mixup for few-shot learning. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2207–2216, 2020. 5

[40] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021. 6, 7, 8, 12

[41] Khanh Xuan Nguyen and Brendan O'Connor. Posterior calibration and exploratory analysis for natural language processing models. In *EMNLP*, 2015. 2

[42] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in neural information processing systems*, 2018. 2

[43] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. Incremental few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 2

[44] Jathushan Rajasegaran, Salman Hameed Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. Self-supervised knowledge distillation for few-shot learning. *ArXiv*, abs/2006.09785, 2020. 5

[45] Samrudhdhi Bharatkumar Rangrej, Kevin J Liang, Xi Yin, Guan Pang, Theofanis Karaletsos, Lior Wolf, and Tal Hassner. Revisiting linear decision boundaries for few-shot learning with transformer hypernetworks, 2021. 2, 4

[46] Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks. *arXiv preprint arXiv:2203.11378*, 2022. 2

[47] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017. 2, 3, 7

[48] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 2

[49] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, 2013. 5

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[51] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. A meta-learning perspective on cold-start recommendations for items. *Advances in neural information processing systems*, 2017. 1

[52] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, 2019. 1

[53] Sachin Vernekar, Ashish Gaurav, Vahdat Abdelzad, Taylor Denouden, Rick Salay, and Krzysztof Czarnecki. Out-of-distribution detection in classifiers via generation. *arXiv preprint arXiv:1910.04241*, 2019. 2

[54] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016. 1, 2, 3, 5, 6, 7, 8, 13

[55] Fei Wang and Anita Preininger. Ai in health: state of the art, challenges, and future directions. *Yearbook of medical informatics*, 2019. 2

[56] K Wang, Paul Vicol, Eleni Triantafillou, and Richard Zemel. Few-shot out-of-distribution detection. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020. 6, 7, 8, 13

[57] Kuan-Chieh Wang, Paul Vicol, Eleni Triantafillou, and Richard Zemel. Few-shot out-of-distribution detection. In *International Conference on Machine Learning (ICML) Workshop on Uncertainty and Robustness in Deep Learning*, 2020. 2

[58] Xin Wang, Fisher Yu, Ruth Wang, Trevor Darrell, and Joseph E Gonzalez. Tafe-net: Task-aware feature embeddings for low shot learning. In *CVPR*, 2019. 2

[59] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 2020. 1, 2

[60] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, et al. Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566*, 2020. 5, 6

[61] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021. 1, 2

[62] Li Yin, Juan M Perez-Rua, and Kevin J Liang. Sylph: A hypernetwork framework for incremental few-shot object detection. In *CVPR*, 2022. 2

[63] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, 2020. 2

[64] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 1, 2, 3, 4, 13

[65] Jingyang Zhang, Nathan Inkawhich, Yiran Chen, and Hai Li. Fine-grained out-of-distribution detection with mixup outlier exposure. *arXiv preprint arXiv:2106.03917*, 2021. 2, 5

[66] Liang Zheng, Yi Yang, and Alexander G Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*, 2016. 1

A. Baseline adaptations for OOD detection in the few-shot classification setting

As reported in Section 5.3, we adapt several existing OOD detection methods for the few-shot classification setting. In particular, the adaptations are carried out for the hypernetwork model $\mathcal{M}$ learned for few-shot classification (FSC). The OOD detection metrics (AUROC and FPR@90) reported in Section 5.4 are calculated using the IND score, which we denote as $s_{\mathcal{M}}$. For all baselines, we use the same implementations for the feature extractor $\mathcal{F}$ and the hypernetwork $\mathcal{H}$ as described in Section 5.2.

Below we provide details on how $s_{\mathcal{M}}$ is calculated for each method.

1. MSP [25]: The MSP uses the largest value of the softmax as the IND score, *i.e.*,

$$s_{\mathcal{M}}(x_q; \mathcal{D}_s) = \max_{c \in C_n} p(Y = c|x_q) \tag{11}$$

where $p(Y = c|x_q) = \text{softmax}\left(\mathcal{C}\left(\mathcal{F}\left(x_q\right)|W\right)\right)[c]$, and C_n corresponds to the set of labels in a few-shot episode as described in Section 3.

2. Entropy [40]: We also consider using the Shannon entropy that depicts the uncertainty with respect to the probabilities predicted by the model. Specifically, we compute

$$s_{\mathcal{M}}(x_q; \mathcal{D}_s) := -H(Y) \tag{12}$$

$$= \sum_{c \in C_n} p(Y = c|x_q) \cdot \log\left(p(Y = c|x_q)\right), \tag{13}$$

where $p(Y = c|x_q) = \text{softmax}\left(\mathcal{C}\left(\mathcal{F}\left(x_q\right)|W\right)\right)[c]$.

3. ODIN [38]: The ODIN detector uses temperature scaling and small input perturbation to improve the MSP baseline for OOD detection. In particular, ODIN introduces hyperparameter S for temperature scaling and ϵ for the magnitude of input perturbation that are used to adjust the IND confidence score:

$$s_{\mathcal{M}}(x_q; \mathcal{D}_s, T, \epsilon) = \max_{c \in C_n} p(\tilde{Y} = c|\tilde{x}_q; T) \tag{14}$$

where $\tilde{x}_q$ is the perturbed input calculated as

$$\tilde{x}_q = x_q - \epsilon \cdot \text{sign}\left(-\nabla_{x_q} \log\left(\max_{c \in C_n} p(Y = c|x_q; T)\right)\right),$$

$p(Y|x_q; T)$ is the softmax probability with temperature scaling, and $p(\tilde{Y}|\tilde{x}_q; T)$ denotes the temperature scaled probability of the perturbed input. In our hypernetwork framework, we calculate $p(Y = c|x_q; T) = \text{softmax}\left(\mathcal{C}\left(\mathcal{F}\left(x_q\right)|W\right)\right)[c]$.

4. DM [34]: The DM OOD detector uses the features extracted from each block of the encoder $\mathcal{F}$. We denote the feature extracted at each block as $f(x, l)$. Given the labels in the support set along with their corresponding features, the parameters of the Gaussian distribution $\mu_{l,c}$ and Σ_l is fitted to each block for each class c in the episode. The covariance is shared across all the classes. The layer-specific score s_l for query x_q is computed as:

$$s_l(x_q; \mathcal{D}_s) := \max_{c \in C_n} \log\left(\mathcal{N}\left(f\left(x_q, l\right); \mu_{l,c}, \Sigma_l\right)\right) \tag{15}$$

The final score can be computed using a linear combination of layer-specific scores, *i.e.* $s_{\mathcal{M}}(x_q; \mathcal{D}_s) := \alpha_l \cdot s_l(x_q; \mathcal{D}_s)$, where the logistic regression model with parameters α_l are found using the validation dataset. However, we found that learning the logistic regression model overfits the support set containing few samples and the OOD samples used in the validation set. Instead, we use $s_{\mathcal{M}}(x_q; \mathcal{D}_s) := \max_l s_l(x_q; \mathcal{D}_s)$ to work better and we use that in all our experiments. Note that DM doesn't use the adapted task-specific classifier to detect OOD samples.

5. pNML [8]: Predictive Normalized Maximum Likelihood (pNML) learner is a recently proposed OOD detection approach that uses generalization error to detect OOD samples. [8] derived a pNML regret for a single layer NN that can be used for features extracted from a pre-trained deep encoder, and can be used as the confidence measure for detecting OOD samples. Namely, the pNML regret of a single layer NN is

$$\Gamma(x; \mathcal{D}_s) = \log \sum_{i=1}^{C_n} \frac{p_i}{p_i + {p_i}^{x^T g}(1 - p_i)} \tag{16}$$

where p_i is the output probability for the i^{th} class and g is a function of the inverse of the data matrix of the features of the IND data as defined in Equation 12 of [8]. We defined the IND score as $s_{\mathcal{M}}(x_q; \mathcal{D}_s) := 1 - \Gamma(x; \mathcal{D}_s)$. In our hypernetwork-based framework for few-shot classification, we use the output probabilities from the adapted task-specific classifier to define p_i. The pNML parameter g is calculated using the features in the support set of a given episode during meta-testing. Due to the limited number of IND samples in the support set, we found the pNML regret to have limited success in detecting OOD samples in the few-shot setting.

6. OE [26]: The Outlier Exposure (OE) uses additional OOE samples during meta-training stage to learn the model. To use OE, we apply the following training

Table 4. The candidate values and the final determined values for the hyperparameters of all the methods.

Method	Candidates	Determined	
		5-shot 5-way	10-shot 5-way
	FS-CIFAR-100 [6]		
ODIN [38]	$\epsilon \in \{0.2, 0.02, 0.002\}$ $T \in \{0.1, 1.0, 10.0\}$	$\epsilon = 0.002$ $T = 1.0$	$\epsilon = 0.002$ $T = 10.0$
OE [26]	$\beta \in \{0.1, 1.0, 10.0\}$	$\beta = 1.0$	$\beta = 1.0$
ParamMix	$a_{PM} \in \{0.1, 1.0, 2.0\}$ $b_{PM} \in \{5.0, 10.0, 20.0\}$	$a_{PM} = 2.0$ $b_{PM} = 5.0$	$a_{PM} = 2.0$ $b_{PM} = 5.0$
OOE-Mix	$a_{OM} \in \{1.0, 10.0, 20.0\}$ $b_{OM} \in \{1.0, 10.0, 20.0\}$	$a_{OM} = 20.0$ $b_{OM} = 20.0$	$a_{OM} = 20.0$ $b_{OM} = 20.0$
	MiniImageNet [54]		
ODIN [38]	$\epsilon \in \{0.2, 0.02, 0.002\}$ $T \in \{0.1, 1.0, 10.0\}$	$\epsilon = 0.002$ $T = 1.0$	$\epsilon = 0.002$ $T = 10.0$
OE [26]	$\beta \in \{0.1, 1.0, 10.0\}$	$\beta = 1.0$	$\beta = 1.0$
ParamMix	$a_{PM} \in \{0.1, 1.0, 2.0\}$ $b_{PM} \in \{5.0, 10.0, 20.0\}$	$a_{PM} = 0.1$ $b_{PM} = 10.0$	$a_{PM} = 0.1$ $b_{PM} = 10.0$
OOE-Mix	$a_{OM} \in \{1.0, 10.0, 20.0\}$ $b_{OM} \in \{1.0, 10.0, 20.0\}$	$a_{OM} = 10.0$ $b_{OM} = 10.0$	$a_{OM} = 10.0$ $b_{OM} = 10.0$

objective when meta-training the hypernetwork framework:

$$\mathcal{L}_{OE} = \mathbb{E}_{\{x,y\}\sim\mathcal{D}_q^{IND}}\left[\mathcal{L}_{CCE}\left(\mathcal{M}\left(x_q;\mathcal{D}_s\right),y\right)\right] + \beta\cdot\mathbb{E}_{(\tilde{x},\tilde{y})\sim\mathcal{D}_q^{OOE}}\left[\mathcal{L}_{CCE}\left(\mathcal{M}\left(\tilde{x}_q;\mathcal{D}_s\right),\tilde{y}\right)\right] \tag{17}$$

where $\mathcal{M}(x_q;\mathcal{D}_s)$ are the output logits for query x_q in the episode containing the support set $\mathcal{D}_s$, $\mathcal{L}_{CCE}$ computes the cross entropy loss, β is a hyperparameter that weighs the loss for OOE samples in the episode, and $\mathcal{D}_q^{IND}$ and $\mathcal{D}_q^{OOE}$ are the IND and OOE samples in the queries of a given episode.

7. OEC [56]: Similar to OE, Out-of-Episode Classifier (OEC) leverages OOE inputs at the meta-training stage. However, unlike OE which uses uniform label distribution for a multi-class objective, OEC learns a binary classifier that distinguishes IND classes from OOE classes. The binary classifier uses the following objective during meta-training stage

$$\mathcal{L}_{OEC} = -\sum_{x_q\in\mathcal{D}_q^{IND}}\log\left(\sigma\left(s\left(x_q;\mathcal{D}_s\right)\right)\right) - \sum_{\tilde{x}_q\in\mathcal{D}_q^{OOE}}\log\left(1-\sigma\left(s\left(\tilde{x}_q;\mathcal{D}_s\right)\right)\right) \tag{18}$$

where $s_{\mathcal{M}}(x_q;\mathcal{D}_s) = \max_{c\in C_n}\log p(Y=c|x_q)$, and $\mathcal{D}_q^{IND}$ and $\mathcal{D}_q^{OOE}$ are the IND and OOE samples in the queries of a given episode. In the few-shot setting, we use the prediction as $\hat{y} = \text{argmax}_{c\in C_n}\log p(Y=c|x_q)$ during meta-testing for calculating the accuracy.

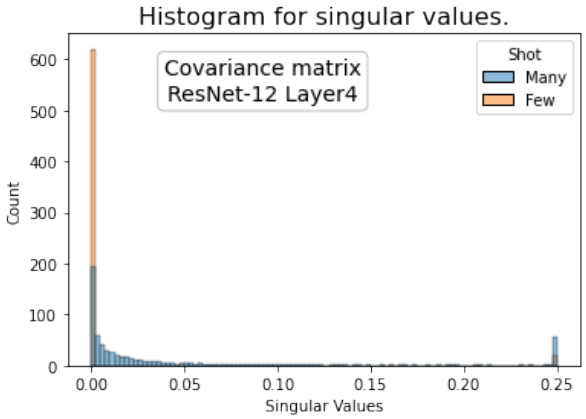

Figure 5. Covariance matrix singular values distribution for many- and few-shot. In the few-shot case, the empirical covariance matrix is degenerate, which leads OOD methods like DM [34] to fail.

B. Hyperparameters

The hyperparameters used for the various baseline methods in our experiments are included in Table 4, showing the settings tried and the value providing the best results. For OOE-Mix and ParamMix, we use a random draw from the Beta distribution to determine the mixup coefficient λ per sample, as is common practice for Mixup [64]. We use MSP [25] for detecting OOD samples after meta-training, using the proposed HyperMix approach. We find the best hyperparameter settings to be fairly consistent across number of shots and the dataset.

C. Effects of few samples in OOD detection

Many out-of-distribution detection methods operate by learning the statistics of the in-distribution samples in some manner or another. In the typical OOD setting, it is common for there to be thousands of examples per class for a model to learn a strong understanding of the in-distribution classes. In contrast, in our FS-OOD, the model has very limited examples in the support set, which may conflict with assumptions made by some OOD methods. An example of this is Deep Mahalanobis [34], which estimates the class-conditional mean and covariance per class. With a few examples, it is difficult to get a good estimate on such statistics. As shown in Figure 5, the data matrix for few examples is degenerate in few-shot cases, leading to especially bad performance (Table 1).