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.

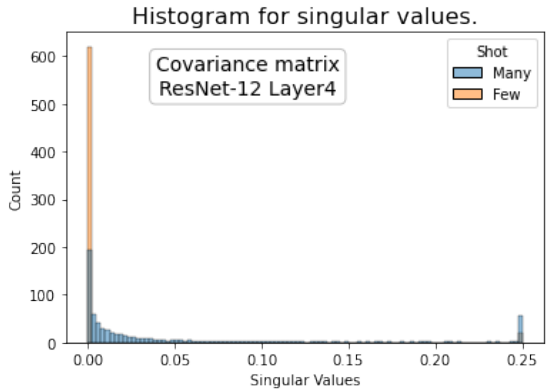

Figure 5. Covariance matrix singular values distribution for many- and few-shot. In the few-shot case, the empirical covariance matrix is degenerate, which leads OOD methods like DM [34] to fail.

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).