Supplementary Materials

7. Pseudo-Code for Eq. 10

Equation: $\bar{Q}_c = \rho \bar{W} S_c = \rho Q (S_c^T S_c + \lambda I)^{-1} S_c^T S_c$

```
# input tensors and their shape:
# query: [way*query_shot*r, d]
# support: [way, support_shot*r, d]
# r: rho
# lam: lambda

# output: Q_bar


def reconstruct(query,support,r,lam):

    reg = support.size(1)/support.size(2)
    st = support.permute(0,2,1)
    xtx = st.matmul(support)
    m_inv = (xtx+torch.eye(xtx.size(-1)).unsqueeze(0).mul(reg*lam)).inverse()
    hat = m_inv.matmul(xtx)
    Q_bar = query.matmul(hat).mul(r)

    return Q_bar
```

Listing 1. Pytorch pseudo-code for feature map reconstruction in Eq. 10 for a single meta-training episode. The whole calculation can be performed in parallel via batched matrix multiplication and inversion.

8. Training Details

We use two neural architectures in our experiments: Conv-4 and ResNet-12. Conv-4 consists of four convolutional layers with kernel size 3×3 and output size 64, followed by BatchNorm, ReLU, and 2×2 max-pooling. ResNet-12 consists of four residual blocks, each with three convolutional layers, with leaky ReLU (0.1) and 2×2 max-pooling on the main stem. We use drop-block as in the original implementation [23, 30, 31, 11][1][2][3][4]. Output sizes for each residual block are: $64, 160, 320, 640$. Due to the large output dimensionality, we found it necessary for stable training to normalize the ResNet-12 outputs by downscaling by a factor of $\sqrt{640}$.

Unless otherwise stated, all models are trained with a weight decay of 5e-4. Training details for each individual benchmark follow.

8.1. Fine-Grained Benchmarks

CUB: Our implementation follows [7, 11]. We train all Conv-4 models for 800 epochs using SGD with Nesterov momentum of 0.9 and an initial step size of 0.1. The step size decreases by a factor of 10 at epoch 400. ResNet-12 models train for 1200 epochs and scale down the step size at epochs 400 and 800. We use the validation set to select the best-performing model over the course of training, and validate every 20 epochs.

Conv-4 models are trained with the standard episodic setup: 20-way 5-shot for 5-shot models, and 30-way 1-shot for 1-shot models. We use 15 query images per class in both settings. In order to save memory we cut these values in half for ResNet-12 models: 5-shot models train on 10-way episodes, while 1-shot models train on 15-way episodes.

We temperature scale the output probability logits of all our models. We also normalize the logits for our baseline models, dividing by 64 for Conv-4 and 640 for ResNet-12. FRN did not benefit from this normalization.

Aircraft: Our implementation for the aircraft dataset roughly follows the CUB setup. We found however that the 1-shot ResNet-12 FRN is highly unstable at the very beginning of training and frequently collapses to the suboptimal uniform solution. We found it necessary to train this model using Adam with an initial step size of 1e-3, and no weight decay. All other hyperparameters remain the same as for CUB.

[1] https://github.com/WangYueFt/rfs
[2] https://github.com/Sha-Lab/FEAT
[3] https://github.com/icoz69/DeepEMD
[4] https://github.com/kjunelee/MetaOptNet

Decoder Network:	
Layer	**Output Size**
Input	$640 \times 5 \times 5$
ResBlock	$512 \times 10 \times 10$
ResBlock	$256 \times 20 \times 20$
ResBlock	$128 \times 40 \times 40$
ResBlock	$64 \times 80 \times 80$
Upsample	$64 \times 84 \times 84$
Conv3x3	$3 \times 84 \times 84$
Tanh	$3 \times 84 \times 84$

Residual Block:	
Input	
TranspConv4x4, stride 2	Conv3x3
	BatchNorm
	ReLU
	TranspConv4x4, stride 2
BatchNorm	BatchNorm
	ReLU
	Conv3x3
	BatchNorm
Addition	
ELU	

Table 9. Neural architecture for the image decoder network (left) and residual block (right). Transposed convolution layers have half as many output channels as input channels.

meta-iNat and Tiered meta-iNat: Our implementation follows [29], only we train for twice as long, as we found that the loss curves frequently failed to stabilize before each decrease in step size. We therefore train our models for 100 epochs using Adam with initial step size 1e-3. We cut the step size by a factor of two every twenty episodes. Because there is no validation set available, we simply use the final model at the end of training. Episode setup is as in CUB and Aircraft.

8.2. Mini-ImageNet

For non-episodic FRN pretraining, we run 350 epochs using SGD with initial step size 0.1 and Nesterov momentum 0.9. We cut the step size by a factor of 10 at epochs 200 and 300. Batch size is 128. Subsequent episodic fine-tuning uses the same optimizer for 150 epochs, but with initial step size 1e-3, decreased by a factor of 10 at epochs 70 and 120. For the 5-shot FRN model, we use the standard 20-way episodes. The 1-shot model uses 25-way episodes.

FRN models trained from scratch (for the ablation study) use the same optimizer for 300 epochs, with initial step size 0.1, decreased by a factor of ten at epochs 160 and 250. Similar to CUB, we cut the way of the training episodes in half in order to reduce memory footprint: 10-way for 5-shot models, and 15-way for 1-shot models.

We continue to use the validation set to select the best-performing model during episodic training, validating every 10 epochs.

8.3. Image Decoder

The decoder network for the visualizations in Section 5.3 takes the form of an inverted ResNet-12, with four residual blocks and a final projection layer to three channels. Upsampling is performed using strided transposed convolution in both the residual stem and the main stem of each residual block. These transposed convolution layers also downsize the number of channels. The final residual block outputs of size 80×80 are rescaled to the original 84×84 resolution using bilinear sampling before the final projection layer. The full architecture is described in Table 9.

The decoder network is trained through L-1 reconstruction loss using Adam with an initial step size of 0.01 and batch size of 200. The CUB decoder is trained for 500 epochs, with step size decreasing by a factor of 4 every 100 epochs. Mini-ImageNet is a larger datset than CUB, so the mini-ImageNet decoder is trained for 200 epochs, with step size decreasing every 40 epochs.

Over the course of training, we found that the FRN classifier learns to regularize the reconstruction problem heavily. This is problematic in that the regularized reconstructions all fall off the input manifold for the decoder network, producing uniformly flat grey-brown images. To prevent this, we removed the normalization on the classifier latent embeddings (multiplying by $\sqrt{640}$). This was sufficient to eliminate the negative impact of regularization and produce meaningful image reconstructions.

9. Additional Results on CUB

9.1. Class Split

Unlike mini-ImageNet, for which researchers tend to use the same class split as [17], CUB doesn't have an official split. For our CUB experiments, we use the same random train/validation/test split as Tang *et al*. [22]. However, many open-sourced baselines [32, 12, 15] use the dataset-generating code[5] from Chen *et al*. [3], using raw, non-cropped images as input.

[5] https://github.com/wyharveychen/CloserLookFewShot/blob/master/filelists/CUB/write_CUB_filelist.py

Model	class split	Backbone	1-shot	5-shot
FRN (ours) 1-shot	Tang *et al*. [22]	ResNet-12	82.02±0.20	-
FRN (ours) 5-shot	Tang *et al*. [22]	ResNet-12	**83.55±0.19**	**92.92±0.10**
FRN (ours) 1-shot	Chen *et al*. [3]	ResNet-12	82.23±0.19	-
FRN (ours) 5-shot	Chen *et al*. [3]	ResNet-12	**83.35±0.18**	**93.28±0.09**
Baseline++$^\flat$ [3]	Chen *et al*. [3]	ResNet-34	68.00±0.83	84.50±0.51
ProtoNet [3, 20]	Chen *et al*. [3]	ResNet-34	72.94±0.91	87.86±0.47
LaplacianShot$^\flat$ [32]	Chen *et al*. [3]	ResNet-18	80.96	88.68
S2M2$^\flat$ [15]	Chen *et al*. [3]	WRN-28-10	80.68±0.81	90.85±0.44
Neg-Cosine$^\flat$ [12]	Chen *et al*. [3]	ResNet-18	72.66±0.85	89.40±0.43

Table 10. Performance comparison for FRN under two different class split settings. We include baselines using the same split as Chen *et al*. [3] for reference. Results of FRN are averaged over 10,000 trials with 95% confidence interval. $\flat$ denotes the use of non-episodic pre-training.

training setting	1-shot	5-shot
from scratch 1-shot	82.02±0.20	-
from scratch 5-shot	83.55±0.19	92.92±0.10
after pre-train	73.46±0.20	84.20±0.13
finetune 1-shot	**83.70±0.18**	-
finetune 5-shot	83.23±0.18	**93.59±0.09**

Table 11. Impact of pre-training for FRN on CUB using raw images as input. Pre-training can slightly boost the performance compared to training from scratch.

Therefore, we re-run our method on CUB under the uncropped setting using the same class split as [3]. As shown in Table 10, FRN performance remains basically the same.

9.2. Pre-Training

We introduce the pre-training technique for FRN in Sec. 3.6 and apply it to mini-ImageNet in Table 5. Here, we apply it to CUB under the un-cropped setting with class splits as in [22].

For non-episodic FRN pretraining, we run 1200 epochs using SGD with initial step size 0.1 and Nesterov momentum 0.9. We cut the step size by a factor of 10 at epochs 600 and 900. Batch size is 128. Subsequent episodic fine-tuning uses the same optimizer for 600 epochs, but with initial step size 1e-3, decreased by a factor of 10 at epochs 300 and 500. For the 5-shot FRN model, we use the standard 20-way episodes. The 1-shot model uses 25-way episodes.

As shown in Table 11, pre-training still improves the final accuracy, but the gain compared to training from scratch becomes much smaller than for mini-ImageNet (see Table 7). This is reasonable, as CUB is only about $\frac{1}{6}$ the size of mini-ImageNet. It is thus comparatively easier in this setting for the FRN trained episodically from scratch to find a good optimum, and more difficult for pre-training to improve.

10. Additional Visualizations

Additional image reconstruction trials as in Fig. 3 of the main paper begin on the following page.

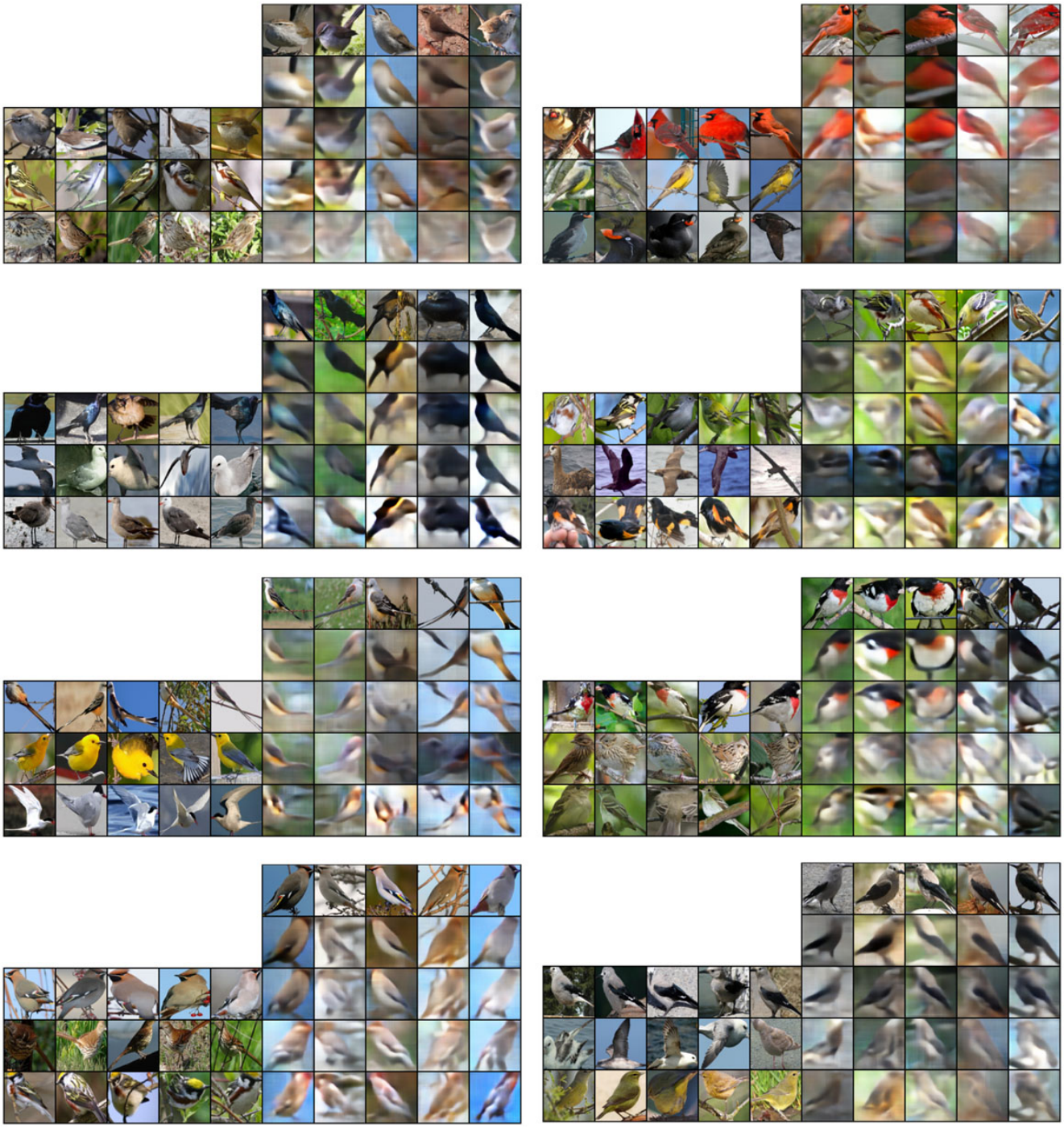

Figure 4. Additional image reconstruction visualizations for CUB. Formatting follows Fig. 3: support images are given on the left while target images and reconstructions are on the right. First row images are targets, second row images are autoencoded, third row images are reconstructed from the same class, and fourth and fifth row images are reconstructed from different classes. Same-class reconstructions are clearly superior to those from different classes.

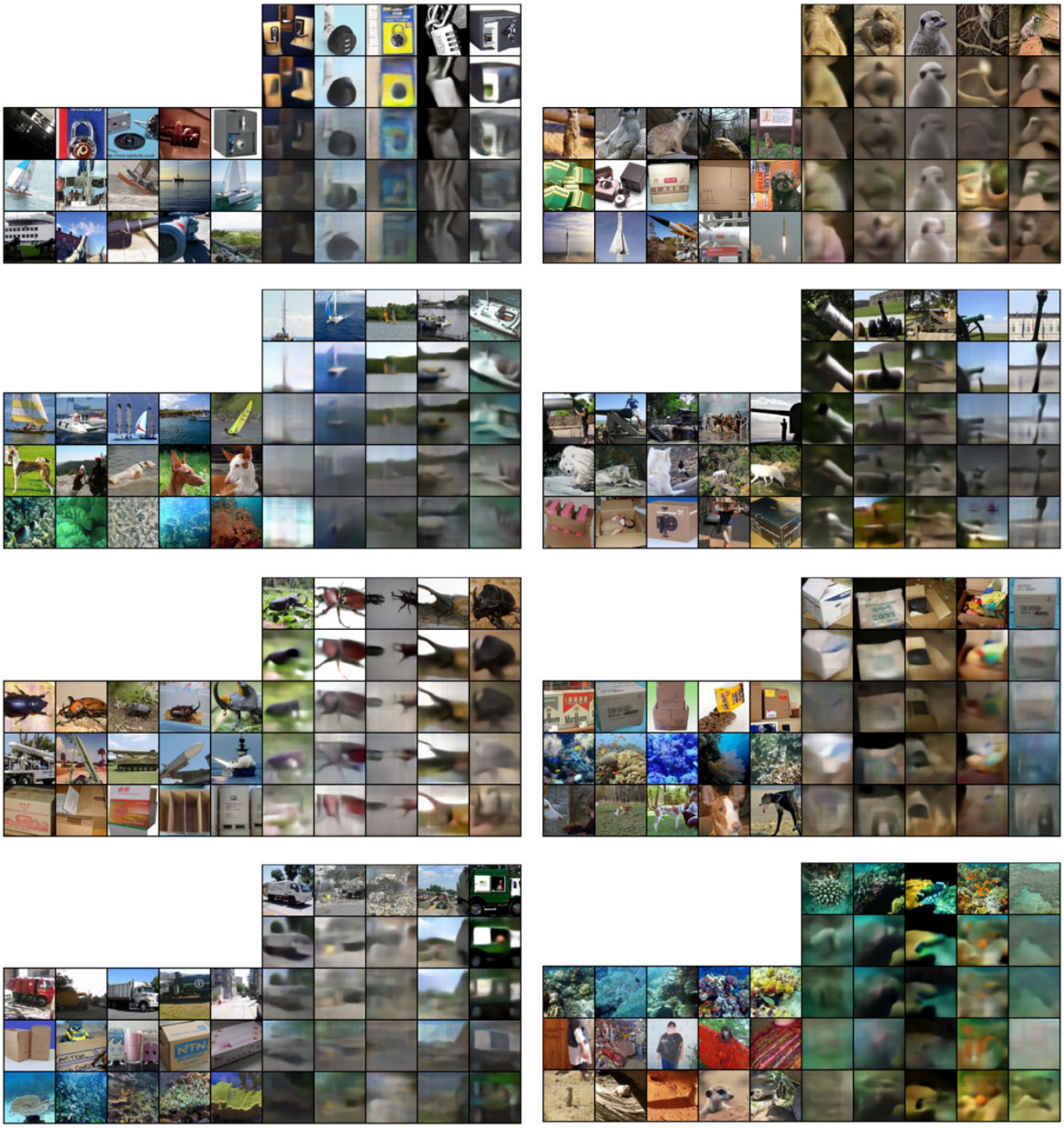

Figure 5. Additional image reconstruction visualizations for mini-ImageNet. Formatting follows Fig. 3: support images are given on the left while target images and reconstructions are on the right. First row images are targets, second row images are autoencoded, third row images are reconstructed from the same class, and fourth and fifth row images are reconstructed from different classes. Same-class reconstructions tend to gray out or darken the colors, but are much more faithful shape-wise than those from different classes.