Few-Shot Learning with Localization in Realistic Settings

Davis Wertheimer
Cornell University
dww78@cornell.edu

Bharath Hariharan
Cornell University
bh497@cornell.edu

Abstract

Traditional recognition methods typically require large, artificially-balanced training classes, while few-shot learning methods are tested on artificially small ones. In contrast to both extremes, real world recognition problems exhibit heavy-tailed class distributions, with cluttered scenes and a mix of coarse and fine-grained class distinctions. We show that prior methods designed for few-shot learning do not work out of the box in these challenging conditions, based on a new "meta-iNat" benchmark. We introduce three parameter-free improvements: (a) better training procedures based on adapting cross-validation to meta-learning, (b) novel architectures that localize objects using limited bounding box annotations before classification, and (c) simple parameter-free expansions of the feature space based on bilinear pooling. Together, these improvements double the accuracy of state-of-the-art models on meta-iNat while generalizing to prior benchmarks, complex neural architectures, and settings with substantial domain shift.

1. Introduction

Image recognition models have purportedly reached human performance on benchmarks such as ImageNet, but depend critically on *large, balanced, labeled training sets* with hundreds of examples per class. This requirement is impractical in many realistic scenarios, where concepts may be rare or have very few labeled training examples. Furthermore, acquiring more labeled examples might require expert annotators and thus be too expensive. This problem is exacerbated in applications (e.g., robotics) that require learning new concepts on the fly in deployment, and cannot wait for a costly offline data collection process.

These considerations have prompted research on the problem of "few-shot" learning: recognizing concepts from small labeled sets [14, 18, 38, 41, 44]. This past work builds "learners" that can learn to distinguish between a small number of unseen classes (often fewer than 20) based on an extremely small number of training examples (e.g. 5 per class). However, multiple challenges plague these methods when they are applied to real-world recognition problems.

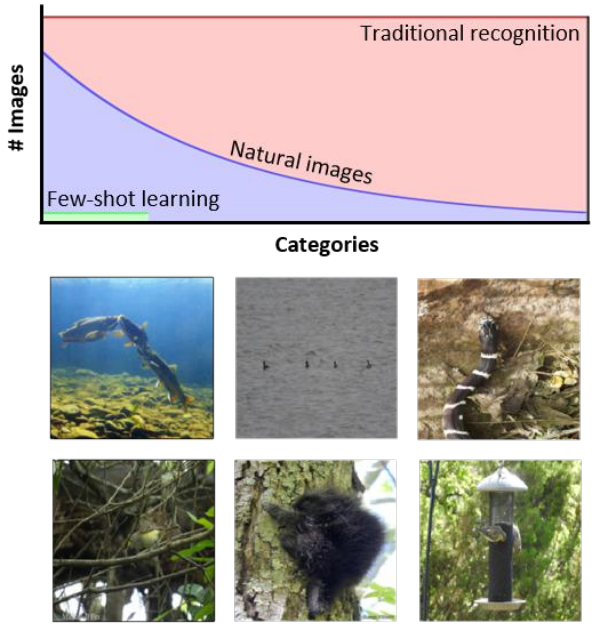

Figure 1. Discrepancies between existing benchmarks and real world problems. **Top:** Traditional recognition benchmarks use many, equally large classes, while few-shot benchmarks use few, equally small classes. Natural problems tend to be heavy-tailed. **Bottom:** Clockwise from top left: relevant objects may be overlapping, tiny, occluded, underemphasized (bird is on the feeder), blurry, or simply hard to delineate [40].

First, few-shot methods typically assume balanced datasets, and optimize the learner for an exact, often unrealistically small number of training examples per class. In contrast, real-world problems may have highly imbalanced, heavy-tailed class distributions, with orders of magnitude more data in some classes than in others. A practical learner must therefore *work equally well for all classes irrespective of the number of training examples*. It is unclear how or even if few-shot methods can handle such an imbalance.

Second, few-shot learning methods often assume the number of relevant concepts to be small, and as such highly distinct from each other. In contrast, real world applications

often involve thousands of classes with subtle distinctions. These distinctions can be particularly hard to detect when natural images are cluttered or difficult to parse (Figure 1, bottom). Thus, the learner must also be able to *make fine-grained class distinctions on cluttered natural images*.

We first evaluate prototypical networks [38], a simple yet state-of-the-art few-shot learning method, on a realistic benchmark based on the heavy-tailed class distribution and subtle class distinctions of the iNaturalist dataset [40]. We show that prototypical networks can struggle on this challenging benchmark, confirming the intuitions above.

We next present ways to address the challenge of heavy-tailed, fine-grained, cluttered recognition. We introduce modifications to prototypical networks that *significantly improve accuracy without increasing model complexity*.

First, to deal with heavy class imbalance, we propose a *new training method based on leave-one-out cross-validation*. This approach makes optimization easier and the learner more resilient to wider distributions of class sizes. This technique yields a **4 point gain** in accuracy.

Second, we posit that when objects are small or scenes are cluttered, the learner may find it difficult to identify relevant objects from image-level labels alone. To tackle this problem, we explore *new learner architectures that localize each object of interest before classifying it*. These learners use bounding box annotations for a tiny subset of the labelled images. Localization improves accuracy by **6 points**, more so when objects occupy under 40% of the image.

Even after localizing the object, the learner may need to look for subtle distinctions between concepts. Existing few-shot methods rely on the learning process alone to build informative feature representations. We show that straightforward, parameter-free adjustments can significantly improve performance. In particular, we find that *the representational power of the learner can be significantly increased by leveraging bilinear pooling* [7, 22, 27]. While in its original formulation, bilinear pooling significantly increases the model parameter count, we show that it can be applied to prototypical networks with *zero* increase. This modification significantly improves accuracy by up to **9 points**.

Together, these contributions **double** the accuracy of prototypical networks and other strong baselines on our challenging heavy-tailed benchmark, with negligible impact on model complexity. Our results suggest that our proposed approach provides significant benefits over prior techniques for realistic recognition problems in the wild.

2. Related Work

The ideas behind our proposed techniques have broad prior support, but appear in mostly disjoint or incompatible problem settings. We adapt these concepts into a unified framework for recognition in real-world scenarios.

Meta-learning: Prior work on few-shot learning has mainly focused on optimizing a *learner*: a function that takes a small labeled training set and an unlabeled test set as inputs, and outputs predictions on the test set. This learner can be expressed as a parametric function and *trained* on a dataset of "training" concepts so that it generalizes to new ones. Because these methods train a learner, this class of approaches is often called "meta-learning". Optimization may focus on the learner's parameterization [6, 15, 30, 37], its update schedule [29, 35, 36], the generalizability of a built-in feature extractor [12, 38, 41], or a learned distance metric in feature space [13, 21, 39, 49]. An orthogonal approach is to generate additional, synthetic data [18, 47].

In most cases, however, few-shot classifiers [6, 15, 21, 29, 35, 36, 38, 39] are evaluated on one or both of only two datasets: mini-ImageNet [41] and Omniglot [25]. The former presents only five classes at a time, with one or five training images per class. The latter is a handwritten character dataset, on which accuracy regularly surpasses 98% [12, 29, 30, 39, 41]. Some recent work has expanded the number of classes dramatically [18, 44], but still assumes that novel classes have the same number of examples. These benchmarks are therefore divorced from real-world conditions, which involve difficult problems, natural images, many concepts, and varying amounts of training data [28, 40, 45]. Many prior meta-learning approaches are incompatible with these settings. Notably, Wang et al [43] design an approach to heavy-tailed problems based on knowledge transfer from common to rare classes. Their approach is orthogonal to our improvements.

Heavy-tailed datasets: Heavy-tailed class distributions are common in the real world. MS-COCO [26], the SUN database [45], DeepFashion [28], MINC [5], and Places [51] are all examples where an order of magnitude separates the number of images in the most versus the least common classes. MINC and Places are especially noteworthy because they are explicitly designed to *narrow* this gap in data availability [5, 51], yet display heavy class imbalance anyway. Despite this trend, standard recognition benchmarks like ImageNet [11], CIFAR-10, and CIFAR-100 [23] heavily curate their data to ensure that classes remain nicely balanced and easily separable. The mini-ImageNet and Omniglot few-shot benchmarks encode class balance explicitly, as do other proposed few-shot benchmarks [18, 44, 47].

Improving feature space: It is well known that higher-order expansions of feature space can raise the expressive power of hand-designed feature extractors [19, 34]. Recent work has shown that similar techniques [7, 22, 27], learnable generalizations of these techniques [8, 48], and efficient approximations to these techniques [16, 20] also improve the performance of convolutional networks. The improvement is especially large in fine-grained classification settings, such as facial recognition [4, 9, 27]. However, using the resulting expanded feature space requires parameter-

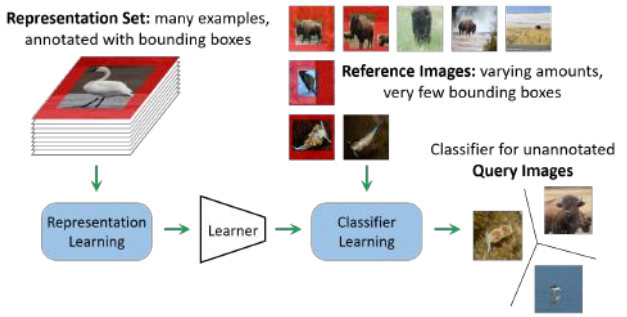

Figure 2. Our real-world learning benchmark. Initially, many images are available with bounding box annotations. The learner must then adapt to new classes using varying but limited amounts of data, with very few bounding boxes. At test time in the wild, there are no annotations.

heavy models, even in the few-shot setting [49]. We adapt bilinear pooling [27] as a truly parameter-free expansion, which no longer risks overfitting to small datasets.

Localization: A close relationship exists between localization and recognition. Networks trained solely on image-level, classification-based losses nevertheless learn to localize objects of interest [31, 50]. These learned localizations can act as useful data annotation, including for the original recognition task [42, 46, 50]. Very difficult problems, however, may require expensive ground truth annotations to begin bootstrapping. Fortunately, a very small set of annotations can be sufficient to predict the rest [37]. Semi-supervised localization further improves when image-level category labels are provided [17, 24]. Since each can bootstrap from the other, combining recognition and localization may prove a particularly effective remedy for data scarcity.

3. Problem Setup and Benchmark

Our goal is to build *learners*, systems that can automatically learn new concepts under challenging real-world conditions, with heavy-tailed distributions of classes and subtle class distinctions. Each learner may have tunable parameters or hyperparameters. As in prior work, these parameters are learned on a "representation set" of concepts ("base classes" in [18]) with many training examples (see Fig. 2).

Once trained, the learner must generalize to a disjoint "evaluation set" of novel categories. The evaluation set is split into a small collection of labeled "reference images" and a larger set of unlabeled "query images". The learner may use the reference images to define the new set of categories, estimate new parameters for those categories (e.g. a linear classifier) and/or fine-tune its feature representations.

Final accuracy is reported on the unannotated query images. We report top-1 and top-5 accuracy, both as a mean over images and over the categories of the evaluation set. The latter metric penalizes classifiers that focus on large categories while ignoring smaller ones.

Two approaches to the above problem act as illustrative examples. A traditional transfer learning approach is to train a softmax classifier on the representation set. On the evaluation set, the fully-connected layer is replaced by a new version with the appropriate number of categories, and fine-tuned on reference images. Query images form the test set. Meta-learning approaches, such as prototypical networks, train a parametric learner on tiny datasets sampled from the representation set, teaching the learner to adapt to novel tiny datasets. The learner processes the evaluation set in a single pass, with reference images forming the training set and query images forming the test set.

Object location annotations: As discussed in Section 1, a key challenge in real-world recognition problems is finding relevant objects in cluttered scenes. Small sets of image-level class labels may be insufficient. We therefore provide bounding boxes for *a small fraction ($\leq 10\%$) of the reference images in the evaluation set*. Note that with extremal point clicks, these annotations are cheap to acquire in practice [33]. We fully annotate the representation set, as such datasets tend to be heavily curated in the real world (Fig. 2).

3.1. Benchmark Implementation

We now convert this problem setup into a benchmark that accurately evaluates learners on real-world heavy-tailed problems. For this, the evaluation set must satisfy three key properties. First, as in many real-world problems, training sets should be heavily imbalanced, with orders of magnitude difference between rare and common classes. Yet the number of examples per class must be neither unnecessarily small (e.g. fewer than 10), nor unrealistically large (e.g. more than 200). Second, in contrast to past few-shot learning benchmarks that use five classes at a time [25, 41], there should be many (e.g. at least 20) categories in the evaluation set, with coarse- and fine-grained distinctions, as in the real world. Third, images must be realistically challenging, with clutter and small regions of interest.

We implement our benchmark using the iNat2017 dataset [40], an organically collected, crowdsourced compendium of living organisms, with fine- and coarse-grained species distinctions, a heavy-tailed class size distribution, and bounding box annotations for a significant subset. Of the appropriately-sized categories with bounding boxes, 80% are randomly assigned to the representation set, and the rest to the evaluation set. Within the evaluation set, 20% of images are reference images and the rest are query images, for an overall split of 80/4/16% representation, reference, and query. We propose this "meta-iNat" dataset as a realistic, heavy-tailed, fine-grained benchmark for meta-learning algorithms. Meta-iNat contains 1,135 animal species, the distribution for which can be found in Fig. 3.

While all images in meta-iNat have bounding box annotations, only 10% are allowed during evaluation (see Sec-

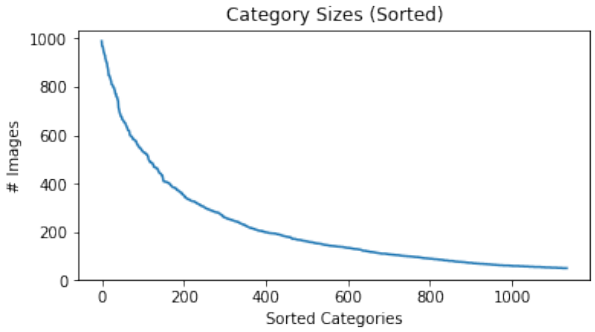

Figure 3. Category sizes in meta-iNat

tion 3). We run ten trials on the evaluation set with a different collection of annotated reference images in each trial.

4. Approach

We build upon prototypical networks [38] (Section 4.1) by introducing three light-weight and parameter-free improvements. Batch folding (Section 4.2) improves gradients during training and helps the learner generalize to large classes. Few-shot localization (Section 4.3) teaches the learner to localize an object before classifying it. Covariance pooling (Section 4.4) greatly increases the expressive power of prototype vectors without affecting the underlying network architecture. In addition to being parameter-free, these techniques are mutually compatible and mutually beneficial.

4.1. Prototypical Networks

We briefly review prototypical networks [38]. Prototypical networks are a learner architecture designed to learn novel classes using few training examples. The learner uses a feature extractor to embed labeled reference and unlabeled query images in feature space. Reference image embeddings are averaged within each class to generate a "prototype" vector for that class. Predictions are made on query embeddings based on L2 proximity to each class prototype.

Training a prototypical network amounts to setting the parameters of the feature extractor, since classification is non-parametric. The prototypical network is trained on the representation set by sampling *small* datasets of reference and query images. These are passed through the network to get class probabilities for the query images. Cross entropy loss on query images is then minimized. Through this training, the network learns a feature extractor that produces good prototypes from limited reference images.

4.2. Batch Folding

Batch folding is motivated by the fact that during training, every image in a batch is either a reference or a query image, but never both. While reference images learn to *form* good class centroids, conditioned on the other contributors, query images *gravitate toward* the correct centroid and *away* from others. Gradients for both are necessary for learning, but every image gets only one, so prototypical weight updates are noisy.

This reference/query distinction also limits the number of reference images a network can handle. For a prototype network to work on common classes as well as rare ones, it must be trained with a larger number of reference images [38]. Increasing the reference images per batch, however, requires either increasing the batch size, which runs into memory constraints, or decreasing the number of queries, producing noisier query gradients. Thus the original prototypical networks are *designed* for rare classes.

As an alternative, we propose to use leave-one-out cross-validation within each batch, abandoning the hard reference/query split. The entire batch is treated as reference images, and the contribution of each image is subtracted ("folded") out from its corresponding prototype whenever it acts as a query. Each image thus gets a combined, cleaner gradient, acting as both a reference and a query. Furthermore, the number of query / reference images can be as high as the batch size / one less. The result is stable training with large reference sets without violating memory constraints. We call this approach batch folding.

Procedure: Let n be the number of classes and p the number of images per class in a batch. Denote by $v_{i,j}$ the feature vector of the i-th image in the j-th category. Let $c_j = \frac{\sum_i v_{i,j}}{p}$ be the centroid of the j-th class. To make predictions for the i-th image in the j-th category, the network uses the following class prototypes:

$$c_1,\ c_2,\ \ldots\ c_{j-1},\ \frac{p}{p-1}(c_j - \frac{v_{i,j}}{p}),\ c_{j+1},\ \ldots\ c_n \qquad (1)$$

Overhead: Batch folding is efficiently parallelizable using tensor broadcasting. The necessary broadcast operations are built-in to most machine learning libraries, including NumPy [1], PyTorch [2], and TensorFlow [3].

Note also that standard prototypical network prediction already involves calculating the L2 distance between every centroid and every query image embedding. This has the same asymptotic cost as calculating (1) for every image, so long as query set size $n_{query} \approx n_{total}$. Generally this is true [38]. The overhead of batch folding also tends to be dominated by earlier convolutional layers.

4.3. Localization

Image-level labels are less informative when the object of interest is small and the scene cluttered, since it is unclear what part of the image the label refers to. Given many, sufficiently different training images, the machine eventually figures out the region of interest [50]. But with only a

few images and image-level labels, distinguishing relevant features from distractors becomes highly difficult.

For these reasons, isolating the region of interest (on *both* reference and query images) should make classification significantly easier. We consider two possible approaches. In **unsupervised** localization, the learner internally develops a category-agnostic "foregroundness" model on the representation set. **Few-shot** localization uses reference image bounding boxes on the *evaluation set* for such localization.

Procedure: In both approaches, the localizer is a submodule that classifies each location in the final 10×10 feature map as "foreground" or "background". This prediction is calculated as a softmax over each pixel embedding's negative L2 proximity to a foreground vector and to a background vector. In unsupervised localization, these vectors are learned parameters optimized on the representation set. In few-shot localization, the localizer gets a few reference images annotated with bounding boxes. We use these boxes as figure/ground masks, and average all the foreground pixel embeddings to produce the foreground vector. The background vector is computed similarly.

The output of the localizer is a soft foreground / background mask. Multiplying the feature map with its mask (and inverse mask) produces foreground and background maps, which are average-pooled then concatenated. This double-length feature vector is used to form prototypes and perform classification. Fig. 4 provides a visual explanation.

Training: Both localization approaches are trainable end-to-end, so we train them within the classification problem. We use no additional supervisory loss; localizers are trained only to be useful for classification. Despite this, the outputs are visually quite good. Examples are given in Fig. 5.

When a few-shot localizer is trained with batch folding, an additional round of folding during localization is required. Each image's contribution is removed from the foreground and background vectors. Otherwise, each image 'sees' its own ground truth bounding box during localization, preventing generalization to unannotated images.

4.4. Covariance Pooling

For hard classification problems, methods such as bilinear pooling [27], fisher vectors [34] and others [4, 16, 22] can be used to expand the feature space and increase expressive power. Unfortunately, a traditional learning framework uses these expanded representations as input to linear classifiers, fully-connected softmax layers, or multilayer networks [9, 20, 27, 49], dramatically increasing parameters and making the model prone to catastrophic overfitting.

However, these techniques can be adapted to prototypical networks without any parameter increase. We use bilinear pooling [27],[1] which improves fine-grained classification performance and generalizes many hand-designed feature descriptors such as VLAD [19], Fisher vectors [34], and Bag-of-Visual-Words [10]. This approach takes two feature maps (from e.g. a two-stream convolutional network) and computes the cross-covariance between them, by performing a pixel-wise outer product before average-pooling. In our localization models, the predicted foreground and background maps act as the two streams. Otherwise, we use the outer product of the feature map with itself. Both versions perform signed square-root normalization, as in bilinear pooling, but do not project to the unit sphere, as this heavily constrains the prototype prediction space.

It is worth emphasizing that this expansion adds no parameters. Unlike prior models, all improvement in performance comes from increased feature expressiveness, not from increased network capacity. To emphasize this distinction, we call this version covariance pooling.

[1] Similar techniques have been called second-order pooling [7], higher-order pooling [22], and covariance descriptors [32] in the literature.

	Top-1 Accuracy		Top-5 Accuracy	
Model	Mean	Per-Class	Mean	Per-Class
Softmax	13.35±.24	6.55±.19	34.46±.30	20.05±.30
Reweighted Softmax	6.92±.19	7.88±.16	21.94±.31	22.53±.29
Resampled Softmax	1.54±.06	.99±.02	3.77±.01	2.75±.03
Transfer Learning	17.39±.24	17.61±.10	41.03±.25	40.81±.27
PN	16.07±.19	17.55±.19	42.1±.21	41.98±.18
PN+BF	20.04±.04	20.81±.08	47.86±.31	46.57±.23
PN+BF+fsL*	26.25±.05	26.29±.04	55.43±.09	53.01±.08
PN+BF+usL	28.75±.13	28.39±.15	57.90±.24	55.27±.37
PN+BF+usL+CP	32.74±.13	30.52±.13	61.32±.14	56.62±.16
PN+BF+fsL+CP*	**35.52±.05**	**31.69±.06**	**63.76±.09**	**57.33±.10**

Table 1. Results on the meta-iNat benchmark, with 95% confidence intervals from 4 trials. PN is a prototypical network, BF is batch folding, fsL and usL are few-shot and unsupervised localization, and CP is covariance pooling. *Results are averaged over 10 runs of 4 trials, annotations randomly sampled per-run.

5. Experiments

We first present overall results on the meta-iNat benchmark (Table 1). We analyze localizer behavior, and then generalization to larger networks, tasks with domain shift, and the original mini-ImageNet. We use 4-layer convolutional learners closely mimicking prototypical networks [38], plus average-pooling (see supplementary).

5.1. Meta-iNat

Baseline results: Standard softmax classifiers trained from scratch on the evaluation set's reference images perform poorly, especially on rare classes. Upweighting rare classes during training improves the per-class accuracy only slightly. Oversampling the rare classes causes catastrophic overfitting. A second baseline is transfer learning: we train the same network on the representation set, but replace and re-train the final linear layer on the evaluation set, using class weights. This approach works significantly better than training from scratch, attaining 17.6% per-class accuracy.

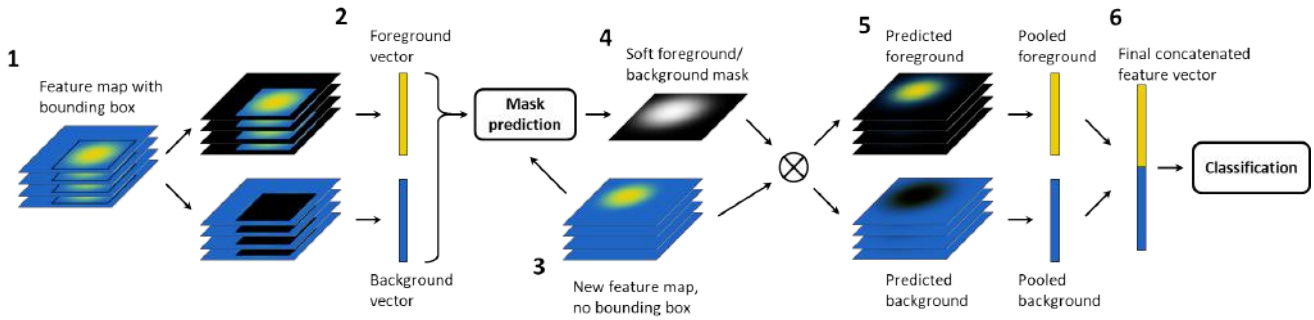

Figure 4. Few-shot localization. Provided bounding boxes mask off foreground and background regions (1), which are averaged to produce foreground and background feature vectors (2). Pixel features on new feature maps (3) are classified as foreground or background based on distance from those vectors (4). The predicted mask separates foreground and background regions (5), which are average pooled independently and concatenated (6). Unsupervised localization learns the foreground/background vectors as parameters, and begins at (3).

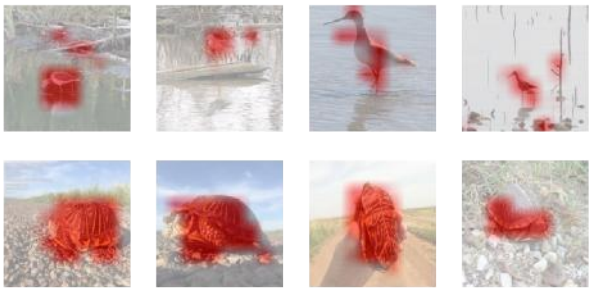

Figure 5. Example outputs of the few-shot localizer. The leftmost image provides the foreground and background centroids for each row. The network learns without supervision or dedicated parameters to isolate (mostly) appropriate regions of interest.

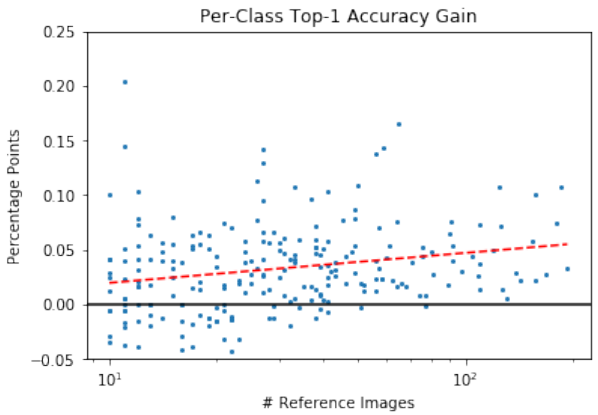

Figure 6. Batch folding improves accuracy for all class sizes in expectation, but particularly helps with large ones ($r^2 = .05$)

As our third baseline, prototypical networks trained on the meta-iNat representation set easily outperform the models trained from scratch, and are comparable to transfer learning without any need for label reweighting. This suggests that prototypical networks are inherently class-balanced, but provide no additional advantages over transfer learning in this heavy-tailed setting.

Batch folding: A prototypical network trained with batch folding outperforms all baselines by almost a **3-point margin**. The per-class accuracy gain as a function of class size is plotted in Fig. 6. We see gains across the board, suggesting that batch folding does provide higher-quality gradients. At the same time, by incorporating more reference images during training, batch folding helps models generalize to larger classes: the positive slope of the best-fit line suggests that large classes benefit more from batch folding, though not at the expense of small ones.

Localization: Incorporating few-shot localization leads to another significant boost in performance, about **6 percentage points**. Note that 10% of reference images are annotated, only 1 to 20 images per category. This relatively cheap annotation has an outsized impact on performance.

Interestingly, unsupervised localization provides a larger gain, about **8 percentage points**. We posit that few-shot localization underperforms its counterpart because it uses bounding boxes, a very coarse form of segmentation. Bounding boxes may include a significant amount of background, hurting the separation of foreground from background. Indeed, we find that when provided bounding boxes are large (e.g. occupying the full image), the few-shot localizer is unable to localize correctly.

As hypothesized, localization particularly helps when objects are small, and bounding boxes cover less than half the image (Fig. 7). The decrease in gain for tiny objects is not entirely surprising - classification is inherently harder when the relevant object contains only a few pixels.

Covariance pooling: Accuracy improves yet again with covariance pooling, yielding a **4 point** gain over unsupervised localization and **9 points** over few-shot localization. Notably, covariance pooling causes class balance to break: large categories benefit disproportionately (Fig. 8). We hypothesize that the high dimensionality of covariance space

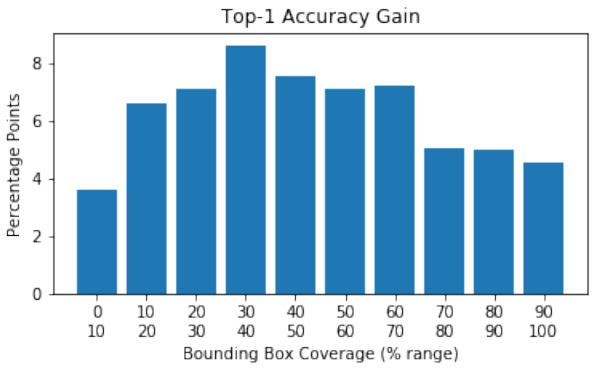

Figure 7. Few-shot localization is more helpful on images with small regions of interest

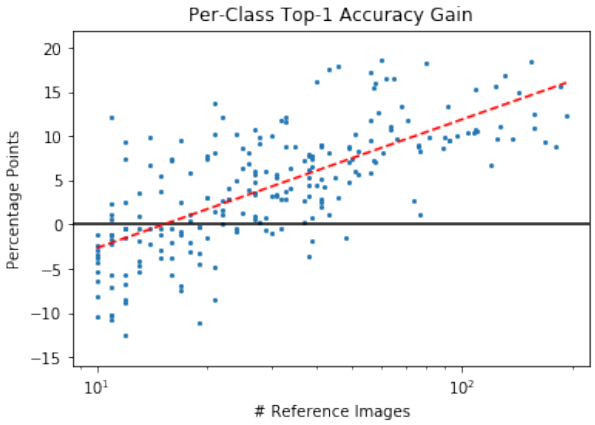

Figure 8. Covariance pooling improves performance on large classes, at the expense of some small ones ($r^2 = .50$)

is responsible. Small categories do not have enough reference images to span the space, so centroid quality suffers.

Unsupervised localization does not interact well with covariance pooling, perhaps because covariance space is too high-dimensional for reference images to span during training. Thus the learned foreground and background vectors may overfit to a particular manifold on the representation set. Few-shot localization, which calculates these vectors dynamically, does not have this problem. We conclude that both localization techniques are useful for different settings.

Using all three techniques, top-1 accuracy is doubled from the baseline prototypical network. The best performer uses **batch folding, few-shot localization** and **covariance pooling**. An ablation study is provided in supplementary.

5.2. Analyzing Few-Shot Localizer Behavior

We next evaluate the number of bounding box annotations needed for good classification accuracy. As shown in Table 2, performance saturates at 16% bounding box availability, but even at 1% (amounting to one box per class), performance decreases only slightly. This scarcity can go further: categories in meta-iNat are grouped into nine supercategories, so we also try using one box per supercategory, nine total. Accuracy does drop significantly, but is still better than models that do not localize. Thus localization can lead to real accuracy gains using hardly any annotations at all, to our knowledge a first-of-its-kind finding.

Localizer	% annotation	Mean acc.	Per-Class acc.
Untrained	10%	19.74±.03	20.42±.06
No Gradient	10%	22.77±.23	22.86±.18
Jointly trained	Supercategory	23.67±.79	24.08±.66
	1%	25.85±.11	25.96±.09
	4%	26.17±.08	26.22±.06
	16%	**26.28±.05**	**26.3±.04**
	64%	26.21±.04	26.25±.03

Table 2. Top-1 accuracy with 95% confidence intervals for few-shot localization models as annotations increase, with comparison to baseline localizers. All models use batch folding.

Joint training: Although the few-shot localizer never receives direct training supervision, it must still be learned jointly with the classifier. Table 2 also compares localizers that are not jointly trained. Applying few-shot localization to a network trained without it leads to a drop in performance ("Untrained"). Training the network to use localization, but preventing backpropagation through the localizer itself, also leads to a drop in performance ("No Gradient"). Localization thus provides a useful training signal, but must itself be trained with the classifier for maximum benefit.

5.3. Generalization

We evaluate our models on three new settings. To test generalization over domain shift, we create a second split of meta-iNat based on supercategories. To test generalization to other network architectures, we evaluate our techniques on meta-iNat using more powerful, pretrained ResNet architectures. Finally, these techniques are tested on mini-ImageNet, using evaluation methods from prior literature. With some expected caveats for mini-ImageNet, our results generalize extremely well to all settings.

Supercategory meta-iNat: We wish to evaluate our results in settings where transfer learning is more difficult, and switching from the representation set to the evaluation set involves substantial domain shift. To that end we construct a new version of meta-iNat, which we call Supercategory meta-iNat. Rather than assign categories to the representation and evaluation sets randomly, we instead split by supercategory. Insects and arachnids (354 total) form the evaluation set, and everything else (birds, fish, mammals, reptiles, etc.) is the representation set. Training and evaluation are performed as before, with results in Table 3.

Transfer learning on Supercategory meta-iNat is much harder than in the original setting. Scores are uniformly lower across the board. However, overall trends remain ex-

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
Reweighted Softmax	4.59±.21	5.38±.22	15.95±.58	16.57±.53
Transfer Learning	6.34±.23	6.19±.14	18.89±.48	17.86±.49
PN	5.33±.18	6.31±.18	17.41±.45	18.32±.27
PN+BF	7.29±.11	8.24±.13	22.09±.35	22.53±.37
PN+BF+fsL*	11.69±.06	12.38±.07	30.64±.11	29.86±.09
PN+BF+usL	12.46±.59	12.95±.51	32.28±1.1	31.18±.95
PN+BF+usL+CP	17.65±.21	16.72±.18	40.16±.26	36.19±.48
PN+BF+fsL+CP*	**20.02±.13**	**17.32±.09**	**43.45±.20**	**36.65±.15**

Table 3. Results on the Supercategory meta-iNat benchmark, with 95% confidence intervals. Models are as in Table 1.

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
Transfer Learn (top)	19.27±.17	18.72±.20	44.02±.30	41.2±.36
Transfer Learn (full)	22.52±.58	18.22±.40	48.16±.60	40.38±.48
PN	35.35±.24	35.59±.11	67.82±.13	66.33±.19
PN+BF	37.36±.15	36.73±.12	69.25±.16	67.03±.15
PN+BF+fsL*	46.2±.04	44.43±.08	75.87±.04	**73.26±.06**
PN+BF+fsL+CP*	**51.25±.13**	**46.04±.13**	**77.5±.06**	72.14±.05

Table 4. Results on meta-iNat using ResNet50 features, with 95% confidence intervals. Transfer Learning (top) adjusts unfrozen upper layers on reference images, while (full) fine-tunes the entire network. Other models are as in Table 1.

Model	5-shot Accuracy	1-shot Accuracy
PN	65.76±.29	49.97±.30
PN+BF	65.2±.29	47.67±.31
PN+fsL	67.85±.29	**51.1±.3**
PN+fsL+CP	**69.45±.28**	49.64±.31

Table 5. Five-class accuracy on mini-ImageNet with 95% confidence intervals over 10 dataset passes. "Shot" refers to number of reference images. Models are as in Table 1.

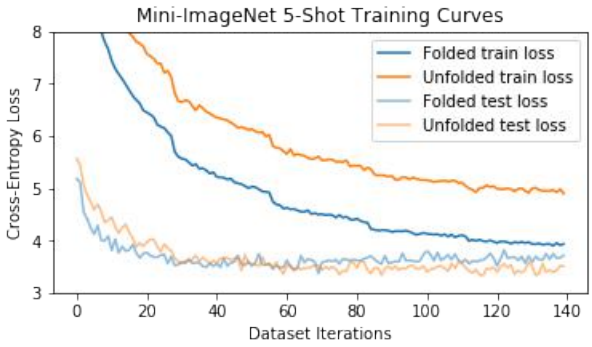

Figure 9. Batch folding causes overfitting on the smaller mini-ImageNet representation set. Models are trained on twenty categories but tested on five, so test loss is lower than training loss.

actly the same. Batch folding outperforms standard prototypical networks and transfer learning baselines by **2 points**. Few-shot and unsupervised localization lead to similar, substantial accuracy gains (**4 points**). Covariance pooling also improves (**5 points**), but again causes mean accuracy to outstrip per-class accuracy. Unsupervised localization underperforms few-shot localization when using covariance pooling, so we remove it from future tests.

ResNet-50: While batch folding, few-shot localization, and covariance pooling lead to substantial improvement on meta-iNat, accuracy is still low. For more powerful models, these improvements might disappear. To test this, we replace the bottom two prototypical network layers with a frozen ResNet-50 pretrained on ImageNet. Details can be found in supplementary. Results are presented in Table 4.

Using the pretrained ResNet-50 model, it is possible to perform transfer learning directly from ImageNet to the meta-iNat evaluation set. Freezing the ResNet, and training just the top two layers on reference images, works poorly given the power of the model. Fine-tuning the entire network on reference images works slightly better, but decreases per-class accuracy. Freezing the ResNet and training the top layers as a prototypical network improves top-1 accuracy by **13 percentage points**. Batch folding, few-shot localization, and covariance pooling provide another **16 points**. We conclude that these techniques are helpful for large neural architectures as well as small ones.

Mini-ImageNet: Batch folding, few-shot localization, and covariance pooling improve accuracy on large evaluation sets with long-tailed class distributions. To see if these techniques still help with the original, smaller few-shot learning problem, we construct a mock mini-ImageNet dataset with similar statistics but annotated with bounding boxes. Performance of prototypical networks on our dataset is similar to the published figures [38]. Table 5 shows the results.

An immediate departure from prior results is the fact that batch folding hurts performance. Batch folding does indeed result in better training and lower training loss, but overfits because the representation set is smaller (Fig. 9).

Few-shot localization and covariance pooling make modest but real improvements when five reference images are provided ("five shot"). There is little discernible effect on single-reference ("one-shot") performance, perhaps because with few reference images and a small set of classes, a more expressive feature space is unnecessary. Nevertheless, the small improvements do suggest that few-shot localization and covariance pooling generalize to few-shot learning.

6. Conclusion

In this paper, we have shown that past work on classical or few-shot balanced benchmarks fails to generalize to realistic heavy-tailed classification problems. We show that parameter-free localization from limited bounding box annotations, and improvements to training and representation, provide large gains beyond those previously observed in data abundant settings. Ours is but a first step in addressing broader questions of class balance and data scarcity.

Acknowledgements This work was partly funded by a grant from Aricent.

References

[1] Broadcasting - numpy v1.15 manual. https://docs.scipy.org/doc/numpy-1.15.0/user/basics.broadcasting.html. Accessed: 2018-11-16.

[2] Broadcasting semantics - pytorch master documentation. https://pytorch.org/docs/stable/notes/broadcasting.html. Accessed: 2018-11-16.

[3] Broadcasting semantics — xla — tensorflow. https://www.tensorflow.org/xla/broadcasting. Accessed: 2018-11-16.

[4] D. Acharya, Z. Huang, D. Pani Paudel, and L. Van Gool. Covariance pooling for facial expression recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[5] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[6] L. Bertinetto, J. a. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Neural Information Processing Systems*, 2016.

[7] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012.

[8] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *IEEE International Conference on Computer Vision*, 2015.

[9] A. R. Chowdhury, T. Lin, S. Maji, and E. Learned-Miller. One-to-many face recognition with bilinear cnns. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016.

[10] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *European Conference on Computer Vision*, 2004.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[12] H. Edwards and A. Storkey. Towards a neural statistician. In *International Conference on Learning Representations*, 2017.

[13] M. Fink. Object classification from a single example utilizing class relevance metrics. In *Neural Information Processing Systems*, 2004.

[14] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[16] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[17] M. Guillaumin, D. Küttel, and V. Ferrari. Imagenet auto-annotation with segmentation propagation. *International Journal of Computer Vision*, 110(3):328–348, Dec 2014.

[18] B. Hariharan and R. Girshick. Low-shot learning by shrinking and hallucinating features. In *IEEE International Conference on Computer Vision*, 2017.

[19] H. Jgou, M. Douze, C. Schmid, and P. Prez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2010.

[20] J. Kim, K. W. On, W. Lim, J. Kim, J. Ha, and B. Zhang. Hadamard product for low-rank bilinear pooling. *International Conference on Learning Representations*, 2017.

[21] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[22] P. Koniusz, F. Yan, P. Gosselin, and K. Mikolajczyk. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):313–326, Feb 2017.

[23] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[24] D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation propagation in imagenet. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *European Conference on Computer Vision*, 2012.

[25] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

[26] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *European Conference on Computer Vision*, 2014.

[27] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. 2017.

[28] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[30] T. Munkhdalai and H. Yu. Meta networks. In *International Conference on Machine Learning*, 2017.

[31] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[32] N. Otberdout, A. Kacem, M. Daoudi, L. Ballihi, and S. Berretti. Deep covariance descriptors for facial expression recognition. In *British Machine Vision Conference*, 2018.

[33] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. Extreme clicking for efficient object annotation. 2017.

[34] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision*, 2010.

[35] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[36] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

[37] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots. One-shot learning for semantic segmentation. In *British Machine Vision Conference*, 2017.

[38] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *Neural Information Processing Systems*, 2017.

[39] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[40] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[41] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016.

[42] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[43] Y. Wang, D. K. Ramanan, and M. Hebert. Learning to model the tail. In *Neural Information Processing Systems*, December 2017.

[44] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, 2016.

[45] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[46] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[47] Y. xiong Wang, R. Girshick, M. Herbert, and B. Hariharan. Low-shot learning from imaginary data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[48] Z. Yang, M. Moczulski, M. Denil, N. d. Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *ICCV*, 2015.

[49] H. Zhang and P. Koniusz. Power normalizing second-order similarity network for few-shot learning. In *IEEE Winter Conference on Applications of Computer Vision*, January 2019.

[50] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[51] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Neural Information Processing Systems*, pages 487–495. Curran Associates, Inc., 2014.

Supplementary Materials

Note: Code for the models and main results of the paper are available at https://github.com/daviswer/fewshotlocal. The full set of supplementary materials, including additional results, localizer visualizations and behavior, and the exact category assignments in meta-iNat by species, can be found here.

7. Ablation Study

Top-1 and top-5 mean and per-class accuracies are given below. Three-digit model names indicate the presence or absence of batch folding, localization, and covariance pooling, in that order. For example, '101' indicates a model with batch folding and covariance pooling, but no localization. Because two versions of localization exist, we use '0' to indicate no localization, '1' for few-shot localization, and '2' for unsupervised localization. A '*' indicates a model presented in the main paper.

	Top-1 Accuracy		Top-5 Accuracy	
Model	Mean	Per-Class	Mean	Per-Class
000*	$16.07 \pm .19$	$17.55 \pm .19$	$42.10 \pm .21$	$41.98 \pm .18$
100*	$20.04 \pm .04$	$20.81 \pm .08$	$47.86 \pm .31$	$46.57 \pm .23$
010	$21.55 \pm .09$	$22.37 \pm .08$	$50.20 \pm .08$	$48.70 \pm .08$
001	$24.31 \pm .48$	$24.64 \pm .39$	$53.29 \pm .76$	$51.09 \pm .63$
020	$24.32 \pm .15$	$24.68 \pm .81$	$53.16 \pm .89$	$51.44 \pm .94$
021	25.60 ± 1.02	$25.51 \pm .90$	54.66 ± 1.07	$52.04 \pm .91$
110*	$26.25 \pm .05$	$26.29 \pm .04$	$55.43 \pm .09$	$53.01 \pm .08$
011	$27.46 \pm .15$	$26.39 \pm .14$	$56.37 \pm .17$	$52.37 \pm .17$
120*	$28.75 \pm .13$	$28.39 \pm .15$	$57.90 \pm .24$	$55.27 \pm .37$
101	$31.06 \pm .61$	$29.07 \pm .53$	$60.41 \pm .67$	$55.50 \pm .62$
121*	$32.74 \pm .13$	$30.52 \pm .13$	$61.32 \pm .14$	$56.62 \pm .16$
111*	$35.52 \pm .05$	$31.69 \pm .06$	$63.76 \pm .09$	$57.33 \pm .10$

Between any two models with different numbers of features, the one with more always outperforms the one with less, regardless of combination. Thus batch folding, localization, and covariance pooling consistently and independently improve performance on meta-iNat.

8. Network Architectures

Our learner architectures mimic the original prototypical networks. Networks contain four 64-channel 3×3 convolutional layers, with Batchnorm, ReLU, and 2×2 max-pooling in between. This is followed by Batchnorm and 10×10 global average-pooling, for a 64-dimensional feature vector. Unlike prototypical networks, which flatten the feature map to a 6400-dimensional vector, we use average pooling to eliminate spatial priors on uncentered, uncropped images. Softmax classifier models use a fully-connected layer on top of the feature vector; prototype classifiers use the feature vector directly. Localization and covariance pooling models replace average-pooling layers with the appropriate algorithm(s).

Models based on ResNet50 use the first two stages of a ResNet50 model, pretrained on ImageNet, to produce 28×28 feature maps with 512 channels. Learned layers consist of 2×2 max-pooling, followed by 3×3 convolution with 128 channels, Batchnorm, ReLU, a second 3×3 convolution with 64 channels, and Batchnorm. The resulting 14×14 feature maps are localized, average pooled, or covariance pooled as appropriate. As in the original setting, softmax classifiers use an additional fully-connected layer, while prototype architectures use the pooled embeddings directly.

9. Training and Implementation Details

9.1. Training

We train meta-iNat models via SGD with Adam, using an initial learning rate of 10^{-3} and dividing by two every epoch. Each epoch consists of 10 passes over the representation set, with five epochs in total. For mini-ImageNet, we lengthen each epoch to 28 passes, to account for the smaller representation set.

In both settings we unit-normalize input color channels. We use random horizontal flipping but no other data augmentation. Because bounding boxes are downsized to very small resolutions (e.g. 10×10), we use 4×4 average-pooling before downsampling, as a fast approximation for anti-aliasing.

9.2. Batch Sampling

While mini-ImageNet uses random batch sampling with replacement during training and testing, meta-iNat uses a different sampling procedure for each. During each training iteration, classes are sampled without replacement: so long as classes have sufficient remaining images to create a batch, they are sampled proportionately to the number of available images they contain. This ensures that differently-sized classes occur at constant, representative rates throughout the entire sampling process. Images are then selected from the sampled classes, and those images are subsequently unavailable for further training until the next pass over the dataset.

During testing, we wish to evaluate classification accuracy for relatively large numbers of classes and images. For sufficiently high values, it becomes impossible to process the sampled datasets as single batches in computer memory. Category sizes also vary so widely that it no longer makes sense to use constant sample sizes. Rather than attempt to evaluate a given network on a large number of large and complicated datasets, we instead impose a single reference/query split over the evaluation set.

Evaluation consists of one pass over the reference images followed by one pass over the query images. During the reference pass, each category is split into manageable batches, and the class centroid is computed from a running total of the embedding vectors. The query pass is divided the same way, and the pre-computed centroids are used to make class predictions. When localization is used, we run 10 trials where different 10% subsets of the reference images are randomly selected to receive bounding box annotations.

The above applies to prototype-styled classifiers, which require representation and query sets for a limited selection of classes in every batch. For straightforward softmax classifiers, we instead use random sampling without replacement, and a batch size of 128. Annealing schedule is as above.