A Appendix

A.1 PCOC Preliminaries

Few-shot learning has seen increasing attention in recent years due to the difficulty of learning from a small amount of examples with deep neural network models [22, 25]. Meta-learning—or "learning-to-learn"—has been shown to be an effective approach to the few-shot learning problem [6, 28, 25]. By training on many different few-shot learning problems, an inner adaptation process is learned, capable of rapid adaptation given few examples. This primarily takes the form of learning an initialization or prior for a learning algorithm, an update rule to incorporate data, or both.

Although there are many taxonomies of meta-learning algorithms [12, 27], three common overarching categories have emerged, recurrence-based methods, optimization-based methods, and metric based-methods. Recurrence-based methods largely focus on a black-box update procedure; the inner learning algorithm typically takes the form of a recurrent neural network [11]. The black-box descriptor highlights the fact that the inner learning algorithm does not leverage any inductive biases from e.g. optimization. Optimization-based meta-learning aims to explicitly use an optimization procedure in the inner learning algorithm, and typically outperforms black-box methods as a result. These approaches rely on back-propagating through an optimization problem, yielding a bi-level optimization problem. Common approaches include back-propagation through the sequence of gradient updates for a set of parameters, as in MAML [6], or back-propagation through the fixed point of a convex optimization algorithm [9, 2, 21, 16]. Finally, metric-based meta-learners rely on the inductive bias of metric learning, in which nearby samples in an embedding space are likely to be of the same class. Generally, these methods aim to learn an embedding space and/or a metric in this space, such that the embeddings of inputs of the same class are close to each other, and different classes are separable based on the metric [25].

In this work, we build on metric-based meta-learners due to their flexibility. Typically, a Euclidean [25] or cosine [28] distance is used between embeddings. We focus our approach on the line of work originating from prototypical networks [25], in which a set of support data is embedded and then aggregated as class prototypes, via

$$\boldsymbol{c}_k = \frac{1}{|S_k|} \sum_{(\boldsymbol{x}_i, y_i) \in S_k} \phi(\boldsymbol{x}_k) \tag{1}$$

where $k = 1, \ldots, K$ indexes the (closed) classes, S_k is the set of input/label pairs corresponding to class k, and $\phi(\cdot)$ is the encoder neural network. Then, the predicted density of a new test point $\boldsymbol{x}_*$ is

$$p(y_* = k \mid \boldsymbol{x}_*) = \frac{\exp(-\|\phi(\boldsymbol{x}_*) - \boldsymbol{c}_k\|_2^2)}{\sum_{k'=1}^{K} \exp(-\|\phi(\boldsymbol{x}_*) - \boldsymbol{c}_{k'}\|_2^2)}. \tag{2}$$

This predictive likelihood may be used to train the neural network features. The inner learning procedure corresponds to linear discriminant analysis [18] with an isotropic covariance. This interpretation, in which in the inner learning algorithm is a Gaussian discriminant analysis learning algorithm and the outer loop learns neural network features, has spawned numerous extensions. In particular, [24] extend the model with both semi-supervised learning and learned covariances, and [1] extend the approach toward a limiting clustering framework.

Of particular interest to the work in this paper is PCOC [8], which performs Bayesian Gaussian discriminant analysis in the embedding space. PCOC fixes a Categorical-Gaussian generative model in the embedding space of the form

$$y \sim \mathrm{Cat}(p_1, \ldots, p_K), \qquad p_1, \ldots, p_K \sim \mathrm{Dir}(\boldsymbol{\alpha}_0) \tag{3}$$

$$\boldsymbol{z} \mid y = k \sim \mathcal{N}(\bar{\boldsymbol{z}}_k, \Sigma_k), \quad \bar{\boldsymbol{z}}_k \sim \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) \tag{4}$$

where $\boldsymbol{z} = \phi(\boldsymbol{x})$. In this model, each class k has a normal distribution in embedding space with mean $\bar{\boldsymbol{z}}_k$ and covariance Σ_k. Moreover, the embedding means are assumed normally distributed with mean $\boldsymbol{\mu}_0$ and covariance Σ_0. While [8] use the prior over embedding means to enable change-point detection, we will use this formulation to enable a general open-set learning framework. Whereas standard prototypical networks implicitly assume a uniform prior over classes, PCOC uses a Categorical-Dirichlet prior on the labels, where $\boldsymbol{\alpha}_0$ is a length k vector of parameters governing how tightly the Dirichlet prior is concentrated. In the limit of $\alpha \to \infty$ for each $\alpha \in \boldsymbol{\alpha}_0$, the uniform prior of prototypical networks is recovered.

This Categorical-Gaussian model is notable for having analytically tractable posteriors, thus enabling simple differentiation through the posterior. The update equations for posterior parameters given data $(\boldsymbol{x}_*, y_*)$ take the form

$$\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \mathbf{1}_k, \quad \boldsymbol{q}_k' = \boldsymbol{q}_k + \Sigma_k^{-1}\phi(\boldsymbol{x}_*), \quad \Lambda_k' = \Lambda_k + \Sigma_k^{-1} \tag{5}$$

for $k = y_*$, and where $\mathbf{1}_k$ denotes a one-hot vector at entry k. Here, the class mean $\bar{\boldsymbol{z}}_k = \Lambda_k^{-1}\boldsymbol{q}_k$. The parameters Λ_k^{-1}, $\boldsymbol{q}_k$ and $\boldsymbol{\alpha}$ are initialized as Λ_0^{-1}, $\boldsymbol{\mu}_0$, and $\boldsymbol{\alpha}_0$ respectively.

A.2 L-PCOC Algorithm

Lifelong PCOC. We extend the Bayesian prototype approach of PCOC to a non-parametric approach with a possibly infinite number of classes, which we refer to as lifelong PCOC (L-PCOC).

We replace the Dirichlet prior of the PCOC model, which assumes a fixed number of classes, with a Dirichlet process that allows for an infinite number of classes. In particular, we leverage the Chinese restaurant process which has been effectively used in other works [19, 1, 13]. For a more detailed description than the brief outline presented here, we refer the reader to [7, 20, 14].

The standard CRP has two parameters, α and β. Assume we have observed n data points belonging to classes $k = 1, \ldots, K$. Let n_k denote the number of data points in class k. Then, the prior probability of a new data point belonging to class $K + 1$ is

$$p(\boldsymbol{y}_{n+1} = K + 1) = \frac{\beta + \alpha K}{n + \beta} \tag{6}$$

and the probability of belonging to class j is

$$p(\boldsymbol{y}_{n+1} = k) = \frac{n_k - \alpha}{n + \beta}. \tag{7}$$

where $\alpha \in [0, 1]$ and $\beta > -\alpha$ to ensure the distribution is a valid probability measure. In this paper, we fix $\alpha = 0.5$ and treat this as a hyperparameter. The parameter β is implemented as the sum of $-\alpha$ and a strictly positive term.

Predictions in L-PCOC. Given a test point $\boldsymbol{x}_*$ and having observed in-episode data $\mathcal{D}_{\text{ep}}$, L-PCOC returns a distribution over the set of observed classes and the novel class (corresponding to predicting that $\boldsymbol{x}_*$ belongs to a previously unseen class). Let K denote the number of observed classes in $\mathcal{D}_{\text{ep}}$; thus, L-PCOC returns a probability distribution over $K+1$ classes. We fix the class noise covariance Σ_k to be the same for all classes, which we write Σ_ϵ. Following Bayes' rule, L-PCOC predictions are of the form

$$p(y \mid \boldsymbol{x}_*, \mathcal{D}_{\text{ep}}) = \text{softmax}_{k=1}^{K+1}(\log p(\boldsymbol{x}_*|y = k, \mathcal{D}_{\text{ep}}) + \log p(y = k|\mathcal{D}_{\text{ep}})).$$

In the above,

$$p(\boldsymbol{x}_*|y = k, \mathcal{D}_{\text{ep}}) = \mathcal{N}(\bar{\boldsymbol{z}}_k, \Lambda_k^{-1} + \Sigma_\epsilon) \tag{8}$$

for $k = 1, \ldots, K$, where $\bar{\boldsymbol{z}}_k, \Lambda_k^{-1}$ are computed via recursive update Eq. 5, computed via conditioning on $\mathcal{D}_{\text{ep}}$ for each k. The class prior $p(y = k|\mathcal{D}_{\text{ep}})$ is computed by the CRP update rules Eq. 7, again computed using $\mathcal{D}_{\text{ep}}$. For prediction of the novel class,

$$p(\boldsymbol{x}_*|y = K + 1, \mathcal{D}_{\text{ep}}) = \mathcal{N}(\bar{\boldsymbol{z}}_0, \Lambda_0^{-1} + \Sigma_\epsilon) \tag{9}$$

and $p(y = K + 1|\mathcal{D}_{\text{ep}})$ follows Eq. 6. The exact update rules for the maintained statistics are presented in Algorithm 1.

Shared Prior and New Class Instantiation. As described above, there exists a shared Gaussian prior over the mean of class embeddings, which has mean $\bar{\boldsymbol{z}}_0$ and covariance Λ_0^{-1}. This prior over means, together with the fixed noise covariance Σ_ϵ, results in a prior over all embeddings (across all classes) the is Gaussian with the same mean and covariance $\Lambda_0^{-1} + \Sigma_\epsilon$. In addition to providing a distribution for samples for unseen classes which is used for computing $p(\boldsymbol{x}_*|y = K+1, \mathcal{D}_{\text{ep}})$, these parameters are used for initialization of the update recursive equations for the statistics of the mean and variance of the class embedding means, $\boldsymbol{q}_k$ and Λ_k. The details of this recursion are provided in Algorithm 1. Critically, we fix the generic prior to be centered at the origin, which matches the regularization induced by our prior in the pre-training phase.

When a new class is instantiated, the number of base classes increases from K to $K + 1$, and a new $K + 2$ novel class bucket is created. The statistics for this new class are computed based on updates to the shared prior. This process is detailed in Algorithm 1.

Meta-Learning. In the meta-learning phase we assume access to a training set, $\mathcal{D}_{\text{train}}$, consisting of N_{train} classes. We sample meta-training tasks from $\mathcal{D}_{\text{train}}$. These are constructed by sampling a support set $\mathcal{D}_{\text{supp}}$ consisting of $K < N_{\text{train}}$ classes and a query set $\mathcal{D}_{\text{query}}$, such that that $\mathcal{D}_{\text{supp}} \cup$

$\mathcal{D}_{\text{query}} \subset \mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{supp}} \cap \mathcal{D}_{\text{query}} = \emptyset$. The model computes the posterior embedding given these support samples, without computing the loss. While this is not strictly lifelong training during the meta-training, it accelerates training with little drop in performance. Given the embeddings computed based on the support set, the model is evaluated over all data points in $\mathcal{D}_{\text{query}}$, making predictions about whether images belong to a support set or a novel incremental class.

To generate $\mathcal{D}_{\text{supp}}$, we sample $K < N_{\text{train}}$ classes and sample a small number of datapoints for each of these classes. We do not assume the number of images per support class is balanced. To compute $\mathcal{D}_{\text{query}}$, we sample a set of incremental classes from the $N_{\text{train}} - K$ remaining classes. Then, we sample images from the set of support and incremental classes. For classification purposes in the query phase of meta-training, there are $K+1$ classification buckets: K corresponding to the support classes and one corresponding to all incremental classes.

Adaptation Loss. While the meta-training scheme allows an efficiently-batched approximation of the lifelong learning setting, rapid adaptation to novel classes is a necessary property of any performant algorithm in the lifelong learning setting and is not necessarily emphasized in the previously discussed training pipeline. We propose an adaptation loss which provides the model with direct supervision to its ability to adapt to unobserved classes given a single labeled example. A single datapoint is sampled from each semantic class contained in the incremental class bucket. The posterior predictive distribution is computed for each class by independently conditioning the shared prior over embeddings on each respective sampled datapoint. The remaining data in the class bucket is classified via Bayesian Gaussian discriminant analysis and a negative log-likelihood loss (NLL) is computed.

Fine Tuning We fine-tune the final layer of the model in the deployment phase, based on the support data. In particular, given a small amount of support data from novel classes in the evaluation phase, we perform fine-tuning of the encoder using this data to improve performance. For all data in the support set, we iteratively generate a prediction, compute an NLL loss from this prediction, and condition on the data point. This loss is used to update the output layer—the last linear layer of the encoder generating the embeddings—via gradient descent.

The fine-tuning of this linear output layer may initially seem surprising. Updating the last layer of the encoder corresponds to a different linear transformation of the nonlinear features. However, because Gaussians are closed under linear transformations, this should not improve performance. However, the set of isotropic Gaussians is not closed under linear transformation, only the sense of dense-covariance Gaussian distributions are. Thus, this fine-tuning step uses the small amount of provided support data at test time to re-scale the features to enable better modeling of the test data via isotropic Gaussians.

In our experiments, we also investigated more aggressive fine-tuning schemes in which other layers of the encoder network were fine-tuned. We found that this was typically highly unstable. We also did not investigate combining in-episode fine-tuning with our approach. We view combining in-episode fine-tuning schemes, which potentially leverage ideas from continual learning [15, 23, 5], with our embedding-based lifelong learning scheme as a particularly promising direction of future work beyond scope of this paper.

Incremental Learning. In addition to the lifelong setting in which no train classes are present at test time, we also investigate the incremental setting in which some or all training classes also appear at test. Our focus on lifelong stems from the fact that effective open-world, few-shot, lifelong learning is a necessary prerequisite for effective incremental learning. Whereas our approach to lifelong learning, L-PCOC, discarded the pre-training embeddings, our incremental framework, I-PCOC, maintains these embeddings during the meta-training phase and during test. The pre-training phase for the lifelong and incremental setting are identical.

During an episode of meta-training, we sample $K_{\text{base}} < N_{\text{train}}$ base classes from the set of training classes. For these classes, we maintain the embeddings from the pre-training phase during that episode of meta-testing. These effectively correspond to the support set in the lifelong setting. Then, images are sampled i.i.d. from the full training set and provided to the learner. As in the lifelong setting, the learner returns probabilistic predictions of an image belonging to a previously observed class or a novel class. Given the label for the image, class embeddings are updated or instantiated, following exactly from the lifelong learning procedure. Thus, the only difference is generation of the support set, which is based on the pre-training embeddings as opposed to conditioning on a small support set. Thus, our embedding-based pre-training provides a convenient bridge linking our proposed lifelong setting to the incremental learning problem setting.

Algorithm 1 Lifelong-PCOC Prediction & Update

```
Require: $\boldsymbol{q}_0, \boldsymbol{\Lambda}_0^{-1}, \Sigma_\epsilon, \alpha, \beta, \mathcal{D}_{\text{supp}}$, query $= (\boldsymbol{x}_*, y_*)$
 1: Initialize CRP class counts: $\boldsymbol{n} = [0, ..., 0]_1^K$
 2: Initialize: $\boldsymbol{q} = [\boldsymbol{q}_0, ..., \boldsymbol{q}_0]_1^K$
 3: Initialize: $\boldsymbol{\Lambda}^{-1} = [\boldsymbol{\Lambda}_0^{-1}, ..., \boldsymbol{\Lambda}_0^{-1}]_1^K$
    Phase 1 – Condition
 4: for $(x_i, y_i)$ in $\mathcal{D}_{\text{supp}}$ do
 5:     Update: $\boldsymbol{n} = \boldsymbol{n} + \mathbf{1}_k$
 6:     Update: $\boldsymbol{q}_k' = \boldsymbol{q}_k + \Sigma_\epsilon^{-1}\phi(\boldsymbol{x}_i)$
 7:     Update: $\boldsymbol{\Lambda}_k' = \boldsymbol{\Lambda}_k + \Sigma_\epsilon^{-1}$
 8: end for
 9: $\boldsymbol{q} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_k, \boldsymbol{q}_0]_1^{K+1}$        ▷ Append shared prior
10: $\boldsymbol{\Lambda}^{-1} = [\boldsymbol{\Lambda}_1^{-1}, ..., \boldsymbol{\Lambda}_k^{-1}, \boldsymbol{\Lambda}_0^{-1}]_1^{K+1}$
    Phase 2 – Predict
11: $\hat{\boldsymbol{y}}$ = softmax$_{k=1}^{K+1}(\log p(\boldsymbol{x}_*|y = k, \mathcal{D}_{\text{supp}}) + \log p(y = k|\mathcal{D}_{\text{supp}}))$
    Phase 3 – Update
12: if $y_* == K + 1$ then        ▷ Novel Class
13:     Append: $\boldsymbol{n} = [n_1, ..., n_k, 1]_1^{K+2}$
14:     Append: $\boldsymbol{q} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_k, \boldsymbol{q}_0]_1^{K+2}$
15:     Append: $\boldsymbol{\Lambda}^{-1} = [\boldsymbol{\Lambda}_1^{-1}, ..., \boldsymbol{\Lambda}_k^{-1}, \boldsymbol{\Lambda}_0^{-1}]_1^{K+2}$
16:     Update $y_* = K + 2$
17: end if
18: Update: $\boldsymbol{n} = \boldsymbol{n} + \mathbf{1}_k$
19: Update $\boldsymbol{q}_k' = \boldsymbol{q}_k + \Sigma_\epsilon^{-1}\phi(\boldsymbol{x}_*)$
20: Update $\boldsymbol{\Lambda}_k' = \boldsymbol{\Lambda}_k + \Sigma_\epsilon^{-1}$
```

A.3 Experimental Details

Datasets Both MiniImageNet and TieredImageNet are subsets of ImageNet ILSVRC-12 [4]. MiniImageNet contains 100 classes, each with 600 images. We implement the standard class splits proposed by [22], which segment the dataset into 64 training, 12 validation and 24 test classes. TieredImageNet, proposed by [24], is significantly larger. The dataset contains 608 classes and over 700,000 images. The ImageNet hierarchical structure is leveraged to construct meta-train and meta-test class splits which minimize the semantic similarity of classes across the split. The result is a more challenging and realistic generalization task. All images, across both datasets, are resized to a uniform 84x84 dimension.

Lifelong Learning Meta-training tasks are sampled from the meta-train class splits. Each task samples 40 classes for inclusion in the support set and an additional 10 incremental classes. Each support set class samples between $[1, 10]$ datapoints. The query set samples a balanced 10 datapoints for each class. Meta-test tasks uniformly sample 10 support classes and 5 incremental classes from the meta-test class splits. Each class samples 10 datapoints for inclusion in the query set. Each model is trained for 60 epochs (each consisting of 1000 tasks) and is evaluated across 300 tasks.

Incremental Learning We create a training and test set for the incremental setting by further partitioning MiniImageNet. The meta-train data is segmented into base-class train and test data. The meta-training set is comprised of only the base-class train data and training tasks are sampled in the same manner as the lifelong setting. The test-set is constructed by augmenting the meta-test class data with the segmented base-class test data. Test tasks sample 5 incremental classes and 10 datapoints for each class. Note that we now perform a much more challenging 64+5-way classification. As in the lifelong setting, each model is trained for 60 epochs and evaluated over 300 tasks.