Meta-Meta Classification for One-Shot Learning

Arkabandhu Chowdhury[1], Dipak Chaudhari[2], Swarat Chaudhuri[2], and Chris Jermaine[1]

[1]Rice University, [2]University of Texas, Austin

Abstract

We present a new approach, called meta-meta classification*, to learning in small-data settings. In this approach, one uses a large set of learning problems to design an ensemble of learners, where each learner has high bias and low variance and is skilled at solving a specific type of learning problem. The meta-meta classifier learns how to examine a given learning problem and combine the various learners to solve the problem. The meta-meta learning approach is especially suited to solving few-shot learning tasks, as it is easier to learn to classify a new learning problem with little data than it is to apply a learning algorithm to a small data set. We evaluate the approach on a one-shot, one-class-versus-all classification task and show that it is able to outperform traditional meta-learning as well as ensembling approaches.*

1. Introduction

Meta-learning, often defined informally as "learning to learn" [43, 34], is a compelling approach for solving very small-data learning problems, such as one-shot or few-shot learning [11]. One can generate a data set that consists of a large number of learning problems, where each problem has just a few training examples, and then use that set to learn how to solve learning problems with just a few examples. This contrasts with competing approaches such as transfer learning [45], where one solves one or more learning problems, and then adapt those solutions to a new, small-data learning problem. Meta-learning learns the learning *process*, rather than how to re-purpose an existing learner.

In this paper, we introduce a new approach to meta-learning, called *meta-meta classification*. Here, we use a large set of learning problems to design a set of k different learners, each of which has high bias and low variance, so that it is skilled at solving a specific type of learning problem. Further, the meta-meta classifier also learns how to examine a new learning problem and select which of the k learners should be used to solve that particular learning problem.

We call the method *meta-meta classification* to distinguish it from *meta-classification*, a term commonly used in ensemble methods [7]. In ensembling, a meta-classifier is a classifier that aggregates the output from a family of learned scoring functions. For example, in bagging [4], a meta-classifier may average the scores output from a family of scoring functions. In more sophisticated methods, the meta-classifier may *itself* be trained so that it learns to produce an accurate output from a set of less accurate scoring functions.

In contrast, by training over a corpus of *learning problems* rather than a single problem, a meta-meta classifier designs a set of learners, while at the same time learning how to examine a new problem and choose which learners are best to solve that problem. Ultimately, given a new learning problem, the output of the meta-meta classifier is a problem-specific meta-classifier defined over the set of scoring functions produced by the learners. Note that while a meta-meta classifier learns how to produce a meta-classifier, it is not itself a meta-classifier.

Meta-meta classification is particularly natural for very small-data learning problems. The underlying assumption here is that it is easier to *classify* a new learning problem with little data than it is to *solve* the new learning problem with little data. Intuitively, this may be the case: learned scoring functions are successfully used all the time to look at a particular object and predict its label. It does not seem to be inherently more difficult to look at a single object and its label (or small set of labeled objects in the case of few-shot learning) and identify which learners may apply to solving the problem. If it *is* possible to look at a restricted number of training examples and choose an appropriately biased, low-variance learner that best applies to the learning task, then the variance reduction realized by choosing a learner that is highly biased for the problem may result in very low error, even on highly data-restricted problems.

Motivation. One-shot classification is an important but difficult problem, with many applications in science and technology. For example, consider the problem of searching a database of brain images, to find images with a particular type of brain lesion, where only a few (or one) images of the desired injury are available. Our proposed method could eas-

ily be used in this case. Note that this example application is an example of "one-vs-all, open-world, one-shot classification", which is under-studied (or never-before studied), and which we consider in this paper. Almost all prior work considers n-way classification (for $n = 5$). Our new method performs very well on this problem, compared to the obvious alternatives. For another particular application, consider the task of recognizing one animal species (for which only one positive image is available) from among a set of images taken by an automated wildlife camera. A set of negative training images may be available, but the set of negative classes cannot be controlled. Or, consider recognizing a particular vessel (boat) from among a large set of satellite images of vessels in the open ocean. In our opinion, these are as meaningful as the more common 5-way classification task seen in most few-shot learning papers. Our work (and future work) on this sort of one-vs-all problem could greatly increase the range of problems amenable to one-shot classification.

Our contributions. We define a new meta-learning strategy called *meta-meta classification*, in which a meta-meta classifier is trained to recognize the type of learning task at hand, and to use that recognition to choose a biased, low-variance learner appropriate for the task. We show how this strategy can be used to learn a highly accurate aggregate scoring function, even for one-shot learning problems. Note that our meta-meta classifier does not necessitate any particular type of learner to produce a classifier. We have chosen a gradient based optimizer, which is close to MAML [13] in essence, and hence we did a thorough comparison with ensembles of MAMLs. As an evaluation example, on a one-shot, one-class-versus-all classification task defined over the ImageNet corpus, meta-meta classification is able to achieve greater than 82% test accuracy, compared to less than 61% test accuracy for the baseline MAML approach, and less than 67% for a comparatively-sized ensemble of MAMLs.

2. Related Work

Meta-meta classification broadly falls under the meta-learning or "learning to learn" paradigm [19, 43, 2] which has been shown to produce promising results on few-shot classification problems.

Among various meta-learning methods, *metric-based methods* [20, 17, 12, 39, 40, 41, 15, 47, 42] aim to learn a similarity function or a distance metric between a pair of different samples. Siamese networks [20] use a pairwise verification loss to perform nearest-neighbours classification. Matching Networks [47] combine both embedding and classification to form an end-to-end differentiable nearest neighbours classifier. Prototypical Networks [41] apply an inductive bias in the form of class prototypes without full context embeddings.

Memory-augmented methods [26, 25, 8, 48, 38, 29] learn to adjust model states using memory-augmented recurrent networks. For example, [38] represents entries from a sample set in an external memory, AdaResNet [27] uses memory and the sample set to produce conditionally shifted neuron coefficients for the query set, and SNAIL [25] uses an explicit attention mechanism to leverage specific information from past experience.

Optimization based methods [13, 1, 14, 51, 23, 16, 28, 37, 35, 33, 52] learn a network initialization that can quickly adapt to new tasks within a distribution of tasks with a very few steps of regular gradient descent. MAML [13] backpropagates the meta-loss through an inner learning loop, Reptile [28] incorporates an L2 loss that updates the meta-model parameters towards the task-specific models, and [23] learns a layer-wise subspace where gradient-based adaptation is done.

There have recently been a few papers that have suggested that transfer-based methods can be more effective in classic few shot classification problems. Chen et al. [5] propose Baseline using a simple linear classifier on top of a pre-trained deep CNN, and Baseline++ using a distance-based classifier. Some authors [21, 44] suggest using a high-quality feature extractor by training a deep CNN on a large data set. Among others, [6] proposes a transductive fine-tuner, [10] proposes feature diversity through diversity in data sets, and [9] proposes the idea of ensembling for few-shot learning through training a number of diverse deep CNNs as feature extractors during meta-learning.

However, all of these papers mainly address the problem of few shot multi-class classification where only few images of a number of classes are available for training. Often in reality, we may have only a single scarce class with a few images (or even one image) available for training. The problem then becomes more of an image retrieval than classification. To the best of our knowledge, few shot literature hardly considers this application which we frame as a One-Vs-All classification problem in this paper.

3. Background and Problem Definition

3.1. Meta-Meta Classification: Overview

Meta-meta classification is an approach to supervised learning that is particularly relevant to the problem of one-shot or few-shot learning, as it relies on learning a set of learners designed specifically to have high inductive bias as a way to prevent over-fitting, as well as how to apply those learners when a new learning problem is encountered.

Specifically, for input (feature) domain X and output (label) domain Y, a meta-meta classifier takes as input a training set (a multi-set drawn from $X \times Y$), and then returns an aggregate scoring function $g^* : X \times Y \to \mathbb{R}$ that combines the output of the learners (in the context of ensemble-based

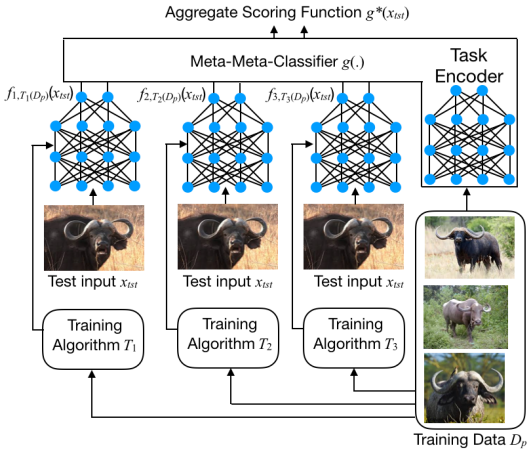

Figure 1. An aggregate scoring function realized via meta-meta classification. The meta-meta classifier $g(.)$ uses the training set D_P to select from among the k parameterized learners $f_{1,T_1(D_P)}$, $f_{2,T_2(D_P)}$, and so on, to realize an aggregate scoring function g^*.

learning, this aggregate scoring function is sometimes referred to as a *meta-classifier*). As in all forms of supervised learning, the goal is to produce an aggregate scoring function that gives relatively high values to pairs from $X \times Y$ that tend to occur together.

In contrast to classical ensemble approaches (such as stacking [50]), in meta-meta classification, the aggregate scoring function is constructed *without* examining how well the individual scoring functions output by the learners perform on the training set (or on a test set). Instead, the meta-meta classifier learns through experience how the learners should be combined for different types of problems. This makes meta-meta classification particularly attractive for few-shot learning problems, as there is no need to have enough data to test the accuracy of the output of the learners.

A meta-meta classifier has two parts: a set of *learners*, and a *meta-aggregation function*.

The learners. In classical supervised learning, we have a single scoring function and a learning algorithm. But in meta-meta classification, we instead assume an ensemble of k learners, from which we wish to build an aggregate scoring function. The ith learner consists of a scoring function $f_{i,\theta_i^f} : X \times Y \rightarrow \mathbb{R}$, as well as a training algorithm T_{i,θ_i^T}.

Let D be the set of all multi-sets drawn from $X \times Y$. The training algorithm $T_{i,\theta_i^T} : D \rightarrow \Theta_i^f$ maps a set of training examples drawn from $X \times Y$ to a particular value for θ_i^f. As is typical, the scoring function f_{i,θ_i^f} is parameterized on the parameter set θ_i^f chosen from parameter space Θ_i^f by the training algorithm. More atypically, the training algorithm is itself parameterized on a parameter set θ_i^T. This parameter set can contain any parameters that control the learning process: the learning rate, the number of learning iterations, the set of parameters to initialize the learning algorithm, etc.

The meta-aggregation function. The goal is to learn, by looking at a set of learning problems, how to examine a new problem, and combine those k learners to create a problem-specific meta-classifier g^*. The meta-aggregation function is given this task.

For a function $f : X_1 \times X_2 \times ... \rightarrow \mathbb{R}$, let $f(x_1, ..., x_m) : X_{m+1}, X_{m+2}, ... \rightarrow \mathbb{R}$ denote the function resulting from currying f with respect to the first m inputs, and then evaluating the resulting curried function at $(x_1, ..., x_m)$. Then $f_{i,T_{i,\theta_i^T}(D_{trn})}(x_{tst}) : Y \rightarrow \mathbb{R}$ is the result of applying the training algorithm in learner i—parameterized with θ_i^T—to training set D_{trn}, and then "pre-loading" the resulting scoring function with x_{test}.

A *meta-aggregation function* examines D_{trn}, and then conditioned on that D_{trn}, combines each of the k scoring functions $f_{i,T_{i,\theta_i^T}(D_{trn})}$ to create a new, more accurate aggregate scoring function.

Formally, a meta-aggregation function is a function:

$$g_{\theta_g} : D \times (Y \rightarrow \mathbb{R})^k \rightarrow (Y \rightarrow \mathbb{R})$$

By allowing the meta-aggregation function to examine the set D_{trn} and aggregate the scoring functions created by the k learners, we obtain the aggregate scoring function $g^*_{\langle \theta_g, \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle}(D_{trn}, x_{tst}, y_{tst}) \equiv$

$$g_{\theta_g}\Big(D_{trn}, f_{1,T_{1,\theta_1^T}(D_{trn})}(x_{tst}), f_{2,T_{2,\theta_2^T}(D_{trn})}(x_{tst}), ..., f_{k,T_{k,\theta_k^T}(D_{trn})}(x_{tst})\Big)(y_{tst}).$$

A depiction of how the learners and the meta-aggregation function together produce an aggregate scoring function g^* is given in Figure 1.

3.2. Intuition: Why Meta-Meta Classification?

If the training set D_{trn} is large, it is unclear that there is much benefit to meta-meta classification. For large $n = |D_{trn}|$, we may choose a general-purpose learner with small inductive bias that works well regardless of the problem at hand. However, if n is small—$n = 1$ in the case of one-shot learning—there may be a significant benefit to the introduction of a set of learners and a meta-meta classifier. If sufficient information about the problem-generating distribution P is available through past experience, that we may learn a high-quality meta-meta classifier. After learning the meta-meta classifier, tiny training set D_{trn} may give enough information as to the exact nature of the classification task that the meta-aggregation function can accurately select an appropriate learner. This learner will ideally have high inductive bias, and be tailored to the specific learning problem.

At the same time, it will hopefully have low variance, and will be accurate, even with the learner has been trained on very small D_{trn}.

In fact, this is the benefit of meta-meta classification: it allows for the use of a set of highly biased, low variance learners each of which covers a small subset of the set of classification problems that are expectedly encountered.

For this to work, a key assumption is that the task of recognizing which type of learning problem we are faced with is *less data-intensive* than the task of actually solving the learning problem. Hence, faced with limited training data, we use that data to first determine which type of learning problem we are faced with, and then use a high-bias learner that has been designed to perform well on that specific class of problem.

3.3. Relationship to Other Approaches

Meta-meta classification is related to several other ideas in machine learning. For example, consider neural architecture search [53, 31] and related ideas. Both approaches effectively appeal to a meta-meta classifier that attempts to choose the best learner for a given task. The key difference, however, is that neural architecture search typically assumes large n, so that the meta-meta classifier is trivial. When evaluating a learner, simply see how accurate the learner is on a holdout set. If the learned model is accurate on the holdout set, the learner is a good choice. In meta-meta classification, the assumption is that there is little data available to evaluate the accuracy of a constructed classifier, and so the meta-meta classifier g is introduced as an alternative to an accuracy test over a holdout set.

There is an obvious relationship between meta-meta classification and boosting, bagging [32], and other ensemble methods. The aggregate scoring function enabled by the meta-meta classifier is effectively controlling the use of an ensemble of learners. In ensemble methods, the function that aggregates the output from an ensemble of learners is often called a meta-classifier. However, the difference is that a meta-meta classifier is trained *how* to produce a task-specific meta-classifier, it is not itself a meta-classifier. By looking at a large number of learning problems, the meta-meta classifier learns how to select an appropriate, high-bias, low-variance learners from a set of learners, few of which are useful for any particular classification task.

Meta-meta classification is related to other meta-learning approaches, for example, [13], as they also assume a distribution of learning tasks, and apply meta-learning to try to solve the one-shot learning problem. The key difference is that Finn et al.'s approach can be seen as trying to design a *single* learner (scoring function plus training algorithm) that works well for small-sized D_P, for *any* data-generating P sampled according to P, rather than attempting to match the present learning task with an appropriate classifier.

Algorithm 1 End-to-End Gradient Descent

Meta-Learn (P, k, b, n_{trn}, n_{tst})
// P: Distribution of distributions to learn from
// k: # of learners
// b: Meta-learning batch size (# of problems)
// n_{trn}: # of training instances in a learning problem
// n_{tst}: # of test instances to evaluate a scoring function
Initialize $\theta = \langle \theta_g, \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle \leftarrow$ **rand**()
while loss decreases **do**
  for $j = 1$ to b **do**
    Sample $P \sim \mathsf{P}$
    Sample $D_{trn,j} = \{(x_i, y_i)\}_{i=1...n_{trn}} \sim P$
    Sample $D_{tst,j} = \{(x_i, y_i)\}_{i=1...n_{tst}} \sim P$
  end for
  $\theta \leftarrow \theta - \frac{\alpha}{b \times n_{tst}} \sum_{j=1}^{b} \sum_{(x_{tst}, y_{tst}) \in D_{tst,j}} \nabla \ell \left(g_\theta^*(D_{trn,j}, x_{tst}), y_{tst} \right)(\theta)$
end while
return θ

4. Learning a Meta-Meta Classifier

4.1. Background

Assume a universe of probability distributions $\mathcal{P}$, each defined over the domain $X \times Y$, and a distribution P defined over this universe. Hence P is a distribution *of* distributions. Now, consider the following hierarchical stochastic process for generating a triple $(D_{trn}, x_{tst}, y_{tst})$ from P:

1. Sample $P \sim \mathsf{P}$
2. Sample $D_{trn} = \{(x_i, y_i)\}_{i=1...n} \sim P$
3. Sample $(x_{tst}, y_{tst}) \sim P$

Here, D_{trn} is a training data set, and (x_{tst}, y_{tst}) is a test pair.

Assume some loss function $\ell : (Y \to \mathbb{R}) \times Y \to \mathbb{R}$. That is, ℓ takes as an argument a scoring function defined over domain Y, a "true" value for the output selected from Y, and scores how accurately the scores reflect the "true" output. Generally, any loss function can be used for ℓ: squared error if Y is the set of real numbers, cross-entropy if Y is a set of categories, etc. For example, for a scoring function $f : Y \to \mathbb{R}$, the squared error loss function is:

$$\ell_{l_2}(f, y) = \left(y - \text{argmax}_{\hat{y}} f(\hat{y})\right)^2$$

The goal when learning a meta-meta classifier is to choose $\langle \theta_g, \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle$ from the parameter space $\Theta_g \times \Theta_1^T \times \Theta_2^T \times ... \times \Theta_k^T$ so as to minimize the expected loss of the meta-meta classifier (or the "meta-loss"):

$$\mathbb{E}_{(D_{trn}, x_{tst}, y_{tst}) \sim \mathsf{P}} \left[\ell \left(g^*_{\langle \theta_g, \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle}(D_{trn}, x_{tst}), y_{tst} \right) \right]$$

There are many possible instantiations of this idea. We now briefly describe a couple of them.

4.2. Example: End-to-End Gradient Descent

Assume that each of the k learners utilizes gradient descent, and that g is differentiable with respect to θ_g. Further, assume that T_i performs one gradient update at learning rate λ using θ_i^T as the initialization of the gradient descent, so that $\Theta_i^f = \Theta_i^T$ and:[1]

$$T_{i,\theta_i^T}(D_{trn}) = \theta_i^T - \frac{\lambda}{n} \sum_{(x,y)\in D^T} \nabla \ell(f_{i,\theta_i^f}(x), y)(\theta_i^T).$$

Then, letting $\theta = \langle \theta_g, \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle$ we can run a gradient descent algorithm to learn the meta-aggregation function parameters θ_g as well as each of the θ_i^T parameters for the various learners. Assuming meta-learning rate α, we repeatedly sample $(D_{trn}, x_{tst}, y_{tst}) \sim \mathsf{P}$ and for each sample, apply the following update rule:

$$\theta = \theta - \alpha \nabla \ell \left(g_\theta^*(D_{trn}, x_{tst}), y_{tst} \right)(\theta)$$

Note that it is easily possible to extend this to training algorithms that perform more than a single gradient update; this merely requires expanding the expression computed by T_{i,θ_i^T} for an appropriate number of gradient steps. In practice, however, only a small number of gradient updates will be used in a small-data setting; a large number of steps will typically result in over-fitting.

Also, in practice, it may make sense to back-propagate the meta-loss from more than a single (x_{tst}, y_{txt}) test pair, as more test pairs may give a more stable estimate of the meta-loss and decrease time-until-convergence.

Finally, there is nothing preventing the use of a *batch* of learning problems $P_1, P_2, ...$ during each iteration of gradient descent. Again, this may result in a more stable algorithm that takes less time to converge.

The full algorithm for end-to-end gradient descent, which uses a batch of learning problems as well as an arbitrarily-sized test set for back-propagation is in Algorithm 1.

4.3. Example: Clustering Plus Gradient Descent

Unfortunately, the algorithm from the previous subsection may not work well in practice. Note that while the meta-meta classifier is being trained in a supervised manner—the goal is to learn a meta-meta classifier that can generate an accurate meta-classifier, in one important sense, the algorithm is unsupervised.

Ultimately, the meta-aggregation function g_{θ_g} must look at a specific training set D_{trn} and determine which of the learners is most appropriate for the underlying problem. If, at the time that g_{θ_g} is being learned, the learners themselves are being learned, this may be viewed as an unsupervised

[1] Here, $\nabla \ell(f_{i,\theta_i^f}(x), y)(\theta_i^T)$ denotes "the gradient of the ith loss function with respect to parameter set θ_i^f, evaluated at θ_i^T."

Algorithm 2 Three-Step-Meta-Learning

Meta-Learn (P, h, k, b, n_{trn}, n_{tst})
// P: Distribution of distributions to learn from
// h: Embedding function for problem instance
// k: # of learners
// b: Meta-learning batch size (# of problems)
// n_{trn}: # of training instances in a learning problem
// n_{tst}: # of test instances to evaluate a scoring function
Initialize $\theta = \langle \theta_g, \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle \leftarrow$ **rand**()

// Cluster a set of problem instances
$Q = \{\}$
for $m = 1$ to big **do**
  Sample $P \sim \mathsf{P}$;
  $Q = Q \cup \{h(\{(x_i, y_i)\}_{i=1...n_{trn}} \sim P)\}$
end for
Run k-means on Q to obtain $\mu_1, \mu_2, ..., \mu_k$

// Create and partition a set of training distributions
$\mathsf{P}_j = \{\}$ for $j = 1$ to k
for $m = 1$ to big **do**
  Sample $P \sim \mathsf{P}$;
  $D = \{(x_i, y_i)\}_{i=1...n_{trn}} \sim P$
  Add P to P_j where $j = \text{argmin}_j \, ||\mu_j - h(D)||_2$
end for

// Learn each of the training algorithms
for $j = 1$ to k **do**
  while loss decreases **do**
    for $l = 1$ to b **do**
      Sample $P \sim \mathsf{P}_j$
      Sample $D_{trn,l} = \{(x_i, y_i)\}_{i=1...n_{trn}} \sim P$
      Sample $D_{tst,l} = \{(x_i, y_i)\}_{i=1...n_{tst}} \sim P$
    end for
    $\theta_j^T \leftarrow \theta_j^T - \frac{\alpha}{b \times n_{tst}} \sum_{i=1}^{b} \sum_{(x_{tst}, y_{tst}) \in D_{tst,i}}$
      $\nabla \ell \left(f_{j, T_{j,\theta_j^T}(D_{trn})}(x_{tst}, y_{tst} \right) \left(\theta_j^T\right)$
  end while
end for

// Now, learn g
while loss decreases **do**
  for $j = 1$ to b **do**
    Sample $P \sim \mathsf{P}$
    Sample $D_{trn,j} = \{(x_i, y_i)\}_{i=1...n_{trn}} \sim P$
    Sample $D_{tst,j} = \{(x_i, y_i)\}_{i=1...n_{tst}} \sim P$
  end for
  $\theta_g \leftarrow \theta_g - \frac{\alpha}{b \times n_{tst}} \sum_{j=1}^{b} \sum_{(x_{tst}, y_{tst}) \in D_{tst,j}}$
    $\nabla \ell \Bigg(g_{\theta_g} \Bigg(D_{trn}, f_{1, T_{1,\theta_1^T}(D_{trn})}(x_{tst}),$
      $f_{2, T_{2,\theta_2^T}(D_{trn})}(x_{tst}), ...,$
      $f_{k, T_{k,\theta_k^T}(D_{trn})}(x_{tst}) \Bigg)(y_{tst}) \Bigg)(\theta_g)$
end while
return θ

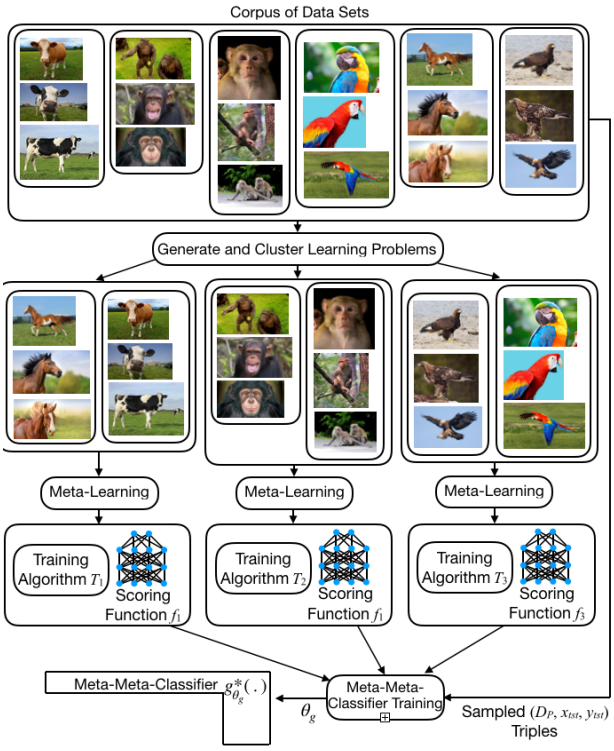

Figure 2. Learning a meta-meta classifier utilizing a pre-clustering of learning problems.

task; it is unclear how to segment the possible problems in $\mathcal{P}$ into categories so that a reasonable learner or learners can be designed for each category.

In practice, unsupervised learning tasks are notoriously sensitive to initialization. Few machine learning practitioners running a k-means algorithm would sample the initial means from a Normal$(\vec{0}, I)$ distribution, for example, as this would likely produce terrible results. Instead, the initial means may be sampled from the data set to be clustered.

Unfortunately, learning a meta-meta classifier consisting of a number of neural network learners via full gradient descent (Algorithm 1), starting with a typical, random neural-network initialization for individual learning parameters $\langle \theta_1^T, \theta_2^T, ..., \theta_k^T \rangle$, is akin to initializing a k-means algorithm poorly. In practice, all θ_i^T values will be terrible, but one will be slightly less terrible than the others, and the meta-aggregation function will learn to route most problems to the corresponding learner. As a result, the other learners are starved of training data and ignored, and the learned solution is equivalent to what would have been returned from the MAML method [13].

One way around this is to sample a large number of distributions from P and explicitly cluster those distributions as a separate step. This requires having some way to cluster distributions of problems; we assume some embedding problem-specific embedding function that is able to map problem distributions (possibly non-deterministically) into a high-dimensional space, where they can be clustered using a k-means algorithm (here k is the number of learners that are to be meta-learned).

A procedure that uses such an explicit clustering step is depicted in Algorithm 2. The procedure is depicted pictorially in Figure 2. After first producing the k clusters of problem distributions, one learner is meta-learned per distribution cluster. Then, in a final step, the procedure trains the meta-aggregation function so that it is able to combine the output of the learners.

Finally, we point out that Algorithm 1 and Algorithm 2 can be used together. Algorithm 2 could be used to produce a high-quality initialization that is refined using Algorithm 1; the combined procedure is likely to outperform either individual methodology.

5. Experimental Evaluation

5.1. One-vs-All One-Shot Image Classification

The first application we consider is open-world classification, where the goal is to recognize a single positive class from a large number of negative classes, some without training examples. This is one-vs-all (OvA) or one-vs-rest (OvR) classification [30]. Hence, we evaluate the utility of meta-meta classification for a series of one-shot image classification tasks, where the goal is to recognize—given a single example—members of a single class which are mixed in with a number of other, "background" classes. We wish to answer two key questions. First, does increasing k (the number of learners) actually increase classification accuracy? Second, does meta-meta classification outperform a simple ensemble of meta-learners? That is, does the biased ensembling of meta-meta classification outperform the simple tactic of just using a number of independent meta-learners?

Meta-learning relies on being able to generate a distribution of learning problems. To generate a learning problem, we sample 51 classes from the classes available for meta-learning, and one is randomly designated as a "positive" class. The training set D_{trn} is generated by sampling one image from the selected positive class, and 50 images from the 50 negative classes (some negative classes may have multiple samples, and some may not be represented in the sample set), and test set D_{tst} is similarly generated by sampling 50 images from the positive class, and 50 from the negative classes.

We consider several different image classification tasks, but the first is to learn to classify images from the ImageNet database. We use the ILSVRC-2012 dataset [36], the most popular flavor of ImageNet data. We hold back 10% of the 1000 ILSVRC-2012 classes for testing, and 90% of the classes are available for meta-learning.

Each f_i is the convolutional network architecture used by

Table 1. Experimental results. The 95% confidence interval of observed test accuracy, computed over 10,000 problems is given. k denotes the number of models trained.

k	WHOLE DATA HARD BAGGING	WHOLE DATA SOFT BAGGING	MM-CLASSIFIER ON WHOLE DATA	**NEAREST CLUSTER**	**META-META CLASSIFIER**
			IMAGENET ILSVRC-2012 RESULTS		
2	61.87 ± 0.22	62.27 ± 0.24	62.79 ± 0.22	61.71 ± 0.25	$\mathbf{66.26 \pm 0.20}$
4	62.48 ± 0.23	61.61 ± 0.24	63.74 ± 0.23	69.53 ± 0.22	$\mathbf{74.02 \pm 0.17}$
8	62.82 ± 0.24	62.40 ± 0.25	64.28 ± 0.23	74.45 ± 0.21	$\mathbf{77.92 \pm 0.17}$
16	63.12 ± 0.24	63.34 ± 0.25	66.11 ± 0.24	74.70 ± 0.22	$\mathbf{82.49 \pm 0.16}$
			CROSS-DOMAIN RESULTS (META-LEARNING ON ILSVRC-2012, TEST ON CUB-2011)		
2	63.60 ± 0.22	64.38 ± 0.25	64.63 ± 0.25	$\mathbf{71.53 \pm 0.20}$	70.87 ± 0.20
4	66.36 ± 0.22	66.27 ± 0.23	66.99 ± 0.23	69.76 ± 0.22	$\mathbf{72.44 \pm 0.17}$
8	66.94 ± 0.24	67.04 ± 0.25	67.52 ± 0.25	74.29 ± 0.15	$\mathbf{77.98 \pm 0.14}$
16	67.21 ± 0.25	67.72 ± 0.26	69.61 ± 0.26	84.04 ± 0.12	$\mathbf{85.67 \pm 0.11}$
			AIRCRAFT DATA SET RESULTS		
2	65.39 ± 0.31	65.66 ± 0.30	69.57 ± 0.24	68.88 ± 0.31	$\mathbf{70.65 \pm 0.26}$
4	70.62 ± 0.27	71.03 ± 0.26	73.00 ± 0.21	71.72 ± 0.28	$\mathbf{76.05 \pm 0.23}$
8	71.84 ± 0.26	72.23 ± 0.27	75.93 ± 0.19	73.35 ± 0.27	$\mathbf{78.61 \pm 0.23}$
			OMNIGLOT DATA SET RESULTS		
2	71.24 ± 0.35	70.69 ± 0.29	73.26 ± 0.35	73.57 ± 0.33	$\mathbf{78.70 \pm 0.31}$
4	73.83 ± 0.35	77.32 ± 0.29	79.16 ± 0.21	77.07 ± 0.28	$\mathbf{85.27 \pm 0.18}$
8	77.70 ± 0.31	77.61 ± 0.28	85.25 ± 0.20	80.15 ± 0.27	$\mathbf{90.87 \pm 0.15}$
16	79.38 ± 0.28	79.56 ± 0.31	88.04 ± 0.18	82.02 ± 0.26	$\mathbf{92.07 \pm 0.14}$

[13], which has 4 modules with a 3×3 convolutions and 32 filters, a ReLU nonlinearity, and 2×2 max-pooling. The scoring function is realized using a fully connected layer after the convolutions, and the last layer is fed into a softmax. Each θ_i^T is the initial set of weights used when training the ith network. During training, five iterations of gradient descent are performed.

The meta-aggregation function g is realized by a simple, fully-connected neural network with two 256-neuron hidden layers. As input, this network accepts:

1. $f_{i,\theta_i^f}(x_{tst}, -1)$ for i in $\{1...k\}$ (that is, the "no" score each learner gives to the test image)
2. $f_{i,\theta_i^f}(x_{tst}, +1)$ for i in $\{1...k\}$ (the "yes" score that each learner gives to the test image)
3. The 512-dimensional output of a ResNet network [18], where the final classification layers have been dropped, applied to the positive image in D_{trn}. This encoding allows the meta-aggregation function to classify the classification problem.

Here, θ_g consists of the weights used in the fully-connected neural network, as well as the ResNet network used to encode D_{trn}.

When using the three-step training process, we sample a training set from the distribution, and our embedding function h pushes the positive training instance in that set through a pre-trained ResNet network. We pre-trained a modified ResNet-152 classifier on the classes reserved for meta-learning and used the penultimate layer for feature extraction. We changed the number of output channels of the convolutions from [64, 128, 256, 512] to [64, 64, 128, 256] and block expansion from 4 to 2. This was done just to decrease the extracted feature size from the usual 2048 to 512.

Finally, each θ_i^T is the starting parameters of the gradient descent used by the ith learner. Hence, in this instantiation of meta-meta classification, we are learning a set of MAML learners [13].

Additional One-Shot Learning Problems. We test three additional one-shot learning problems.

(1) Meta-learn on ImageNet ILSVRC-2012, test on the CUB-2011 Birds data set [49]. In this task, meta-learning is performed exactly as above, on 900 classes selected from the ImageNet ILSVRC-2012 data set. However, the testing distribution is different. Each positive class for testing is selected from among the CUB-2011 Birds data set, and the negative classes are selected from among the 100 classes held back from the ILSVRC-2012 data set. The goal is to perform cross-domain testing.

(2) Meta-learn on 87 classes from the Aircraft data set [24], test on 15 classes. During testing, one of the 15 test

classes is chosen as the positive class, the other 14 classes are the negative classes. One training image is available from the positive class, and 50 from the 14 negative classes. The goal is to perform fine-grained testing.

(3) Meta-learn on 1200 characters from the Omniglot data set [22], test on 423 characters. During testing, a letter from the testing set is selected as the positive class, and 50 other test letters are selected as negative classes. Again, one image from the positive class is available, and 50 images of the other letters are available.

Competitive Methods Tested. To evaluate the efficacy of our ideas, we compare meta-meta classification against ensembles of meta-learners. In our experiments, the individual learners in the ensemble are MAML learners [13]. While a number of improvements to MAML have been suggested in the last couple of years (several of which are described in the Related Work section of this paper), we use MAML as a comparison point because our meta-meta classifier is effectively learning a set of MAML models. This facilitate an apples-to-apples comparison, though we note that MAML (both in meta-meta classification, and in the ensemble) could be replaced with any reasonable alternative.

Overall, we evaluate the following five classifiers: (1) *Whole-data hard bagging*: this is hard bagging over an ensemble of MAML models all trained on the entire data set. (2) *Whole-data soft bagging*: soft bagging over an ensemble of MAML models. (3) *Meta-meta classifier on whole data*: here we first learn a set of MAML models, each on the whole data, but then learn a meta-meta classifier (step three of three-step meta-learning) on the MAML models. This is useful for testing the utility of segmenting the data. (4) *Nearest cluster*: this is essentially the first two steps of three-step meta-learning, with the final classifier replaced with a simple nearest neighbor classifier on the ResNet features. (5) *Meta-meta classifier*: this is the full three-step meta-learning.

Results. For each data set and each of the five competitive methods, we increase the number of learners k logarithmically (with a step of 2) until we found no significant improvement in the classification performance. In each case, we randomly generate 10,000 learning problems to evaluate each method, and the method is scored using accuracy on 50 positive and 50 negative examples. For the evaluation, we use five iterations of gradient descent. All results (including average accuracy, 95% confidence interval width) are given in Table 1. For comparison, a single MAML model achieved 60.78% accruacy on ImageNet ILSVRC-2012, 62.37% accuracy on the cross-domain bird recognition problem, and 64.92% and 65.95% accuracy on the aircraft and Omniglot problems.

Discussion. Across all of the learning tasks, the meta-meta classifier consistently had the best accuracy—often considerably higher than the other options, and much higher than a single MAML model. For example, on the ILSVRC-2012 data set, a meta-meta classifier with 16 classes obtains more than 82% accuracy, compared to just under 61% accuracy with a single MAML model.

5.2. 5-way One-Shot Image Classification

While meta-meta classification is designed for OvA or OvR classification, it can easily be adapted to 5-way classification, which is more commonly studied in the meta-learning literature. This is done by treating a 5-way classification problem as five different OvA classification problems [3]. That is, given a 5-way, one-shot classification problem, we train the meta-meta classifier five times, where we cycle through the five classes, with each class in turn serving as the "one", and the other four classes serving as the "all". Then, when it is time to classify a test image, we choose class associated with the classifier giving the highest positive score.

To test the utility of meta-meta classification for 5-way classification, we follow the procedure in [13] with the ILSVRC-2012 data set. We randomly sample one image each from five randomly sampled classes as the support data for training, and randomly sample 15 images each from those classes as query data for testing. We sample 600 such problems from our test split, and measure the accuracy as well as 95% confidence interval. On the ILSVRC-2012 data set, 16-cluster meta-meta classification obtained 57.23$\pm$1.72% accuracy, whereas MAML gave us 49.64$\pm$1.07% accuracy, for a net gain of +7.59% compared to MAML. Comparing with the results presented in [46] which also tested 5-way classification accuracy on ILSVRC-2012, +7.59% net gain beats the two best meta-learning methods tested: Proto-MAML (proposed in [46]; +4.02%) and Proto-Net (proposed in [41], +4.99%). We also tested meta-meta classification for 5-way classification on the Aircraft data set, and obtained 57.12$\pm$1.86% accuracy compared to MAML's 51.35$\pm$1.1%, for a net gain of +5.77%.

6. Conclusion

We have explored a new type of meta-learning, called *meta-meta classification*. The idea is to learn a set of learners tailored to different problem types, as well as a function called a "meta-meta classifier" that is able to look at a particular problem and decide how to combine the learners to solve that problem. Meta-meta classification is predicated on the assumption that it is easier to classify a problem (and choose an appropriate set of learners) than it is to learn to solve the problem with little data. We have shown through a series of experiments that meta-meta classification can have much high accuracy.

Acknowledgments. The work in this paper was funded by NSF grant #1918651 and by NIH award # UL1TR003167.

References

[1] Antreas Antoniou, Harri Edwards, and Amos Storkey. How to train your maml. In *Seventh International Conference on Learning Representations*, 2019.

[2] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas, 1992.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

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

[6] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.

[7] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[8] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl $ˆ 2$: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[9] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3723–3731, 2019.

[10] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting Relevant Features from a Multi-domain Representation for Few-shot Classification. *arXiv e-prints*, page arXiv:2003.09338, Mar. 2020.

[11] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[12] Michael Fink. Object classification from a single example utilizing class relevance metrics. In *Advances in neural information processing systems*, pages 449–456, 2005.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[14] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[15] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[16] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[17] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186, 1987.

[20] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[21] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6, 2019.

[22] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[23] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. *arXiv preprint arXiv:1801.05558*, 2018.

[24] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[25] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[26] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2554–2563. JMLR. org, 2017.

[27] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. *arXiv preprint arXiv:1712.09926*, 2017.

[28] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2:2, 2018.

[29] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[30] Florent Perronnin, Zeynep Akata, Zaid Harchaoui, and Cordelia Schmid. Towards good practice in large-scale learning for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3482–3489. IEEE, 2012.

[31] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[32] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[33] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[34] Larry A Rendell, Raj Sheshu, and David K Tcheng. Layered concept-learning and dynamically variable bias management. In *IJCAI*, pages 308–314, 1987.

[35] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[37] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[38] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[39] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[40] Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. Attentive recurrent comparators. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3173–3181. JMLR. org, 2017.

[41] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[42] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Web-scale training for face identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2746–2754, 2015.

[43] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[44] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

[45] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[46] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[47] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[48] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860, 2018.

[49] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[50] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[51] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342, 2018.

[52] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018.

[53] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.