Incremental Few-Shot Meta-Learning via Indirect Discriminant Alignment

Qing Liu[1], Orchid Majumder[2], Alessandro Achille[2], Avinash Ravichandran[2], Rahul Bhotika[2], and Stefano Soatto[2]

[1] Johns Hopkins University
[2] Amazon Web Services

Abstract. We propose a method to train a model so it can learn new classification tasks while improving with each task solved. This amounts to combining meta-learning with incremental learning. Different tasks can have disjoint classes, so one cannot directly align different classifiers as done in model distillation. On the other hand, simply aligning features shared by all classes does not allow the base model sufficient flexibility to evolve to solve new tasks. We therefore indirectly align features relative to a minimal set of "anchor classes." Such *indirect discriminant alignment* (IDA) adapts a new model to old classes without the need to re-process old data, while leaving maximum flexibility for the model to adapt to new tasks. This process enables incrementally improving the model by processing multiple learning *episodes*, each representing a different learning task, even with few training examples. Experiments on few-shot learning benchmarks show that this incremental approach performs favorably compared to training the model with the entire dataset at once.

1 Introduction

Meta-learning aims to train a model to learn new tasks leveraging knowledge accrued while solving related tasks. Most meta-learning methods do not incorporate experience from learning new tasks to improve the "base" (meta-learned) model. Our goal is to enable such improvement, thus creating a virtuous cycle whereby every new task learned enhances the base model in an incremental fashion, without the need to re-process previously seen data. We call this *incremental meta-learning* (IML).

While visual classification with a large number of training samples per class has reached performance close to human-level, learning from few samples ("shots") remains a challenge, as we discuss in Sect. 4. We explore the hypothesis that incrementally learning a model from many different tasks, each with few training examples, yields a model comparable to one trained with a large number of images at once. Accordingly, we focus on IML for *few-shot learning*.

IML is not merely meta-training [31, 3, 35, 6, 25, 19] done by processing the training set in chunks: In IML, the meta-training set keeps changing, and we require both the performance of the model for the new tasks, as well as the base model, to improve. IML is also not just incremental (or continual) learning

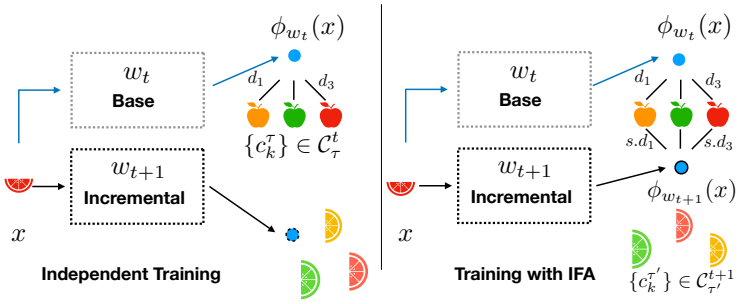

Fig. 1: *Indirect Discriminant Alignment (IDA): Before alignment (left), an orange (new input data) processed through a base model backbone yields an embedding that has a different distance-vector to apples (old class anchors) compared with the one processed through the incremental model backbone. After performing alignment, they produce embeddings that have a similar distance-vector signature, while the incremental model can use the remaining degrees of freedom to adapt the embedding to solve new tasks.*

[26, 20, 21, 4, 17, 33], which focuses on a *single* model to tackle new tasks while avoiding catastrophic forgetting. In IML, we want to continuously improve the meta-trained model so that, presented with an unseen task, it achieves better performance now that it would have before solving the previous task. Moreover, we want to improve the base learner without the need to re-process old data, since that may no longer be accessible, or it may become too expensive to re-process. However, we also want IML to allow exploiting old data, if that is available. Thus far we have used terms like "training task" or "model" informally. In the next section, we elaborate on these terms and then make them formal in Sect. 2.1.

Nomenclature. We identify a *learning task* with a training set. This dataset, together with a function to be minimized (*loss*) and the set of functions to minimize it (*models*), defines an optimization problem whose solution is *a* trained model that "solves the task." So, a learning task can be identified with both a training set, and a suitably trained model. In Sect. 2.1, we will introduce empirical cross-entropy as a loss, deep neural networks (DNNs) as a set of functions, and stochastic gradient descent (SGD) as an optimization scheme. A model consists of a *feature representation*, or *embedding*, obtained by processing each datum through a *backbone*, which is then used to test each hypothesis, or *class*, using a *discriminant function*. A *discriminant* is a function that maps a feature to the hypothesis space, where its value is used to render a decision. A discriminant vector is the collection of discriminant values associated to each hypothesis or class. A *classifier* is a function that outputs the minimizer (or maximizer) of a discriminant function, which corresponds to a *predicted* class or hypothesis. For instance, the Bayesian discriminant is the posterior density of the classes given the data. The corresponding discriminant vector is the collection of posterior probabilities for each hypothesis. The optimal Bayesian classifier is one one that returns the hypothesis with the maximum a-posteriori probability (MAP).

1.1 Key Contribution and Organization

The main contribution of this paper can be more easily understood for the case of metric classifiers, where each class is represented by a prototype, or "center" (Fig. 1), and the discriminant compares features to prototypes, for instance using the Euclidean distance in the embedding space, although our method is not restricted to this case. Each new learning task has a set of classes that is possibly disjoint from those of old tasks. The goal of IML is to update the base model incrementally while solving each new task, without necessarily requiring access to data from old tasks, and despite each new task having only few samples per class. Simply imposing that all tasks use the same features would be too restrictive, as different tasks may require the embedding to change while preserving the old centers. On the other hand, we cannot compare discriminants or classifiers directly since they map to different hypothesis spaces.

Since we cannot compare classifiers directly, and we do not want to restrict the model's freedom to evolve, the **key idea** is to align models for the old and new tasks *indirectly*, by imposing that their discriminants (in the metric case, the vector of distances to the class centers) be aligned *relative to a minimal set of "anchor classes,"* and otherwise leaving the embedding free to adapt to new tasks. The minimal anchor set is represented by the old centers. Thus, *indirect discriminant alignment* (IDA) is performed by mapping the data to old centers, through both the new and the old backbones, and minimizing the misalignment between the two resulting discriminant vectors. Misalignment can be measured in a number of ways, and the process can be conducted by only using new data *incrementally*, resulting in a continuous improvement of the old model. The more general case, which applies to non-metric classifiers, is explained in Sect. 2.2. It results in the **main contribution** of our work, which is to propose what is, to the best of our knowledge, *the first method for incremental few-shot meta-learning.*

We tackle the case of few-shot learning since, in the presence of large amounts of data for the classes of interest, pre-training a large model and fine-tuning it for the task at hand already yields a strong baseline. This is not the case for few-shot learning, where the current state-of-the-art still lags far behind [5]. Our method directly generalizes several meta-learning algorithms [19, 35, 25], and is applicable to more yet.

In Sect. 2.3 we describe two implementations for performing incremental meta-learning and a number of baselines (Sect. 3.1), which form the basis for empirical evaluation in Sect. 3.3 on few-shot benchmark datasets outlined in Sect. 3.2. We also introduce DomainImageNet in Sect. 3.2 to measure the effect of meta-learning on incremental learning when new classes are both in- and out-of-domain. We highlight some limitations and failure cases in Sect. 3.4. We further discuss related work and future opportunities in Sect. 4.

2 Method

The next section establishes the notation and describes incremental learning and meta-learning in a formalism that will make it easy to describe our key

contribution in Sect. 2.2. At that point, it is a small step to incremental meta-learning, as described in Sect. 2.2.

2.1 Preliminaries

A model for a classification task is a parametric approximation $p_w(y|x)$ of the posterior distribution of the class $y \in \{c_1, \dots, c_K\}$ given the test datum x. For a given class of functions (architecture), the model may be identified with its parameters (*weights*) w. The model is trained by minimizing a loss function L that depends on a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, which defines the task, so that $w_0 \doteq \arg\min_w L(w; \mathcal{D})$.

Incremental learning assumes that an *incremental dataset* $\mathcal{E}$ is provided in addition to $\mathcal{D}$. If the two are disjoint, we can write

$$L(w; \mathcal{D} \cup \mathcal{E}) = L(w; \mathcal{D}) + L(w; \mathcal{E}) \tag{1}$$

If L is differentiable with respect to w and we train until convergence ($\nabla_w L(w_0, \mathcal{D}) = 0$), we can expand L to second-order around the previous parameters w_0 to obtain

$$\begin{aligned} L(w; \mathcal{D} \cup \mathcal{E}) &= L(w; \mathcal{D}) + L(w; \mathcal{E}) \\ &\simeq L(w_0 + \delta w; \mathcal{E}) + L(w_0; \mathcal{D}) + \delta w^T H(w_0; \mathcal{D}) \delta w \end{aligned}$$

where $w = w_0 + \delta w$ and $H(w_0; \mathcal{D})$ is the Hessian of the loss $L(w; \mathcal{D})$ computed at w_0. Ignoring the constant term $L(w_0; \mathcal{D})$ yields the derived loss

$$\mathcal{L}(w) = L(w; \mathcal{E}) + \delta w^T H(w_0; \mathcal{D}) \delta w \tag{2}$$

minimizing which corresponds to fine-tuning the base model for the new task while ensuring that the *parameters* change little, using the Hessian as the metric,[3] such as in Elastic Weight Consolidation [17]. Note that, even if the incremental set $\mathcal{E}$ is small, there is no guarantee that the weight update δw will be small. Moreover, making δw small in eq. (2) is unnecessary, since the weights can often change considerably without changing the network behavior.

Distillation is based on approximating the loss *not* by perturbing the weights, $w_0 \to w_0 + \delta w$, but by perturbing the discriminant function, $p_{w_0} \to p_{w_0 + \delta w}$, which can be done by minimizing

$$\mathcal{L}(w) = L(w; \mathcal{E}) + \lambda \mathbb{E}_{x \sim \mathcal{D}} \, \mathrm{KL}(p_{w_0}(y|x) || p_w(y|x)) \tag{3}$$

where the Kullback-Leibler (KL) divergence measures the perturbation of the new discriminant p_w with respect to the old one p_{w_0} in units λ. The losses in eq. (2) and eq. (3) are equivalent up to first-order, meaning that a local first-order

[3] A metric is a positive semi-definite symmetric bilinear form. Since the Hessian H for deep networks typically may not be, it is often approximated by the Fisher Information Matrix, which is positive semi-definite and easier to compute.

optimization would yield the same initial step when minimizing them. Eq. (3) may be interpreted as model distillation [2, 13] or trust-region optimization [32]. The drawback of this method is that it needs access to old samples to compute the loss, since the KL is averaged over $\mathcal{D}$. Our goal is to extend these ideas to meta-learning where $\mathcal{D}$ may no longer be accessible.

Meta-Learning presents an additional difficulty: A meta-training dataset consists of several tasks, indexed by τ, each learned in a separate training *episode,* represented by a different dataset $\mathcal{D}_\tau$ with possibly different classes $\{c_1^\tau, \dots, c_K^\tau\} = \mathcal{C}_\tau$. Rather than training a single model to minimize the loss on a single dataset, a meta-learning algorithm aims to produce a task-agnostic model that minimizes the loss across all meta-training tasks. For the case of empirical cross-entropy:

$$L(w; \mathcal{D}) = \frac{1}{N_\tau} \sum_\tau \frac{1}{|\mathcal{D}_\tau|} \sum_{(x_i, y_i) \in \mathcal{D}_\tau} -\log p_w^\tau(y_i|x_i) \tag{4}$$

The first sum ranges over however many *meta-training tasks* $\mathcal{D}_\tau$ are available. To formalize the core idea in the next section, without loss of generality we write the posterior density $p(y|x)$ in terms of a "backbone" function ϕ that maps each sample x to a feature vector: $z = \phi(x)$, and a discriminant "head" f that maps a feature vector to the posterior $p(y|x) = f(y|\phi(x))$.

2.2 Indirect Discriminant Alignment (IDA)

The challenge in extending incremental learning (3) to meta-learning (4) is that each task $\mathcal{D}_\tau$ in the latter has a different discriminant f_τ for a different set of classes $\mathcal{C}_\tau$. Thus, aligning the discriminants directly would be imposing alignment between different classes, which is not meaningful. A naive solution would be to just align the features $\phi_w(x)$ on all inputs, for instance by minimizing their average distance

$$\mathbb{E}_x \|\phi_{w_t}(x) - \phi_{w_{t+1}}(x)\|^2 \tag{5}$$

However, this would be needlessly restrictive: Completely different features can yield the same posterior density, and we want to exploit this flexibility for incremental learning. Moreover, we want our method to only process new data, rather than keep re-processing old data, which would defy the goal of incremental processing. To simplify the notation, we refer to p_{w_τ} as p_{old} and $p_{w_{\tau+1}}$ as p_{new}, and so for the heads $f_{\text{old}}, f_{\text{new}}$ and the backbones $\phi_{\text{old}}, \phi_{\text{new}}$. Each can be trained on different tasks, or episodes, τ.

The key idea of this paper is to enable *aligning the old and new discriminants using "class anchors" from the old task τ, while processing only data from the new task τ'.* This is done through *indirect discriminative alignment* (IDA), illustrated in Fig. 1, which addresses the challenge that the tasks τ and τ' may not share any classes. IDA uses the classes defined by the old discriminants as "anchors," and imposes that the features processed through the old and new embeddings share the same discriminant relative to these anchor classes. For

metric-based classifiers, the classes can be represented by points in latent (feature, or embedding) space, and the anchors are just an under-complete basis of this space, with the discriminant vector represented by the Euclidean distance to each anchor class representative. The under-complete alignment leaves the residual degrees of freedom free for continual learning. However, the method is more general, allowing any discriminant function.

To make the dependency on the anchor classes and episodes explicit, we write the model $p_w^\tau(y|x) = f_w^\tau(y|\phi_w(x))$. Indirect discriminative alignment of the new model to the old one is then performed by minimizing:

$$\text{IDA}_{\mathcal{E}}(\phi_{\text{new}}|\phi_{\text{old}};\mathcal{C}_{\text{old}}) = \mathbb{E}_{x\sim\mathcal{E},\tau'}\big[\,\text{KL}(f_{\text{old}}^{\tau'}(y|\phi_{\text{old}}(x))||f_{\text{old}}^{\tau'}(y|\phi_{\text{new}}(x)))\big] \tag{6}$$

where $\mathcal{C}_{\text{old}}$ is a set of classes obtained after training on the old training set and τ' are tasks sampled from the new dataset $\mathcal{E}$.

Intuitively, we reuse the old class representatives $\mathcal{C}_{\text{old}}$ and ask that the new features ϕ_{new} remain compatible with the discriminant f_{old}. Moreover, instead of sampling from the old dataset $\mathcal{D}$ – which we may no longer have access to – we sample from $x \sim \mathcal{E}$. In the case of metric classifiers, this can be interpreted as aligning the new features to a set of anchor points, which in particular are the old class representatives.

Note that f can be any discriminant that can process data generated via a representation function ϕ, where both f and ϕ have their own parameters. Also, the choice of KL-divergence to measure the discrepancy between discriminant vectors is due to the fact that it yields a simple expression for most commonly used models, but IDA is not limited to it and any other divergence measure could be employed instead.

Incremental Meta-Learning Given (6), incremental meta-learning consists of solving

$$w_{t+1} = \arg\min_{w_{t+1}} L(w_{t+1};\mathcal{E}) + \lambda\ \text{IDA}_{\mathcal{E}}(\phi_{w_{t+1}}|\phi_{w_t};\mathcal{C}_t) \tag{7}$$

where the first term corresponds to fine-tuning the base model on the new data, while the second term enforces indirect discriminant alignment relative to the anchors from old classes. In the next section, we describe our implementation and empirical evaluation.

2.3 Implementation

The simplest implementation of our method eq. (7) is obtained by using a metric classifier as the base meta-learner. This choice limits us to each task having the same number of classes, a choice we will discuss and extend in appendix. We represent a metric-based classifier using a function ψ_w that computes the class representatives, or prototypes, or "centers," $c_k^\tau = \psi_w(\mathcal{D}_\tau)_k$,[4] and a function

[4] We overload the notation c to indicate the classes in $\mathcal{C}$ and the class representation, which are the argument of χ, since both represent the classes.

(metric) $\chi_w(z_i, c_k^\tau)$ that scores the fit of a datum, represented by the feature vector z_i, with an hypothesis corresponding to a class c_k^τ. Each function can be fixed [35] or learned [25]. Note that the backbone ϕ_w is common to all tasks, whereas the metric changes with each few-shot task $\mathcal{D}_\tau$, since $c_k^\tau \in \mathcal{C}_\tau$ and $\mathcal{C}_\tau = \{c_k^\tau = \psi_w(\mathcal{D}_\tau)_k\}_{k=1}^K$. According to this model, the optimal (Bayesian) discriminant for the task $\mathcal{D}_\tau$ is of the form:

$$p(y = k|z) = \frac{e^{\chi_w(z, c_k^\tau)}}{\sum_j e^{\chi_w(z, c_j^\tau)}} \tag{8}$$

where $z = \phi_w(x)$. Note that χ_w and $p(y|x)$ are equivalent discriminants: Maximizing the posterior is equivalent to minimizing the negative log, which yields a loss of the form

$$L(w; \mathcal{D}) = \frac{1}{N_\tau} \sum_\tau \frac{1}{|\mathcal{D}_\tau|} \sum_{(x_i, y_i) \in \mathcal{D}_\tau} -\chi_w(z_i, c_{y_i}^\tau) + \log\left(\textstyle\sum_{k=1}^K e^{\chi_w(z_i, c_k^\tau)}\right) \tag{9}$$

Our first implementation has a trained backbone ϕ_w but fixes the metric χ to be the L_2 distance and the class representatives to be the means:

$$\begin{aligned} \chi(z, c) &:= -\|z - c\|^2, \\ \psi(\mathcal{D}_\tau, k) &:= \tfrac{1}{|\mathcal{C}_k|} \textstyle\sum_i \delta_{y_i, k} z_i \\ \mathcal{C}_\tau &= \psi(\mathcal{D}_\tau, k)_{k=1}^K. \end{aligned}$$

The detailed computation of the loss eq. (9) is described in the appendix. After every training episode, we discard the data used for meta-training and only retain the class anchors $\mathcal{C}_\tau$. Our paragon (oracle), that will be described in eq. (10), does not retain any class anchors but trains a new meta-learner at every episode, utilizing all data seen thus far. Ideally, the final performance of the two should be similar, which would justify incremental processing of new datasets without the need to re-process old data, which was our working hypothesis and the basis of eq. (2). Indeed, this is what we observe in Sect. 3.3.

For meta-training, we sample few-shot tasks using episodic sampling i.e., each batch consists of K classes sampled at random. We then sample N_s samples as the support samples and N_q samples as the query samples. The class representations are calculated only using the support samples, while the query samples are used to compute the loss. For training the base model, we sample few-shot tasks τ from the old dataset $\mathcal{D}$ and train the model using the loss function in eq. (4). To train the incremental model we sample a few-shot task τ' from the new dataset $\mathcal{E}$. We then sample K random class anchors (of the old dataset $\mathcal{D}$) from C_τ^t, which are calculated and preserved after the previous training phase. During incremental phase(s), the network is trained by minimizing eq. (9).

3 Empirical Validation

We compare our simplest method, which is based on a Prototypical Network architecture (PN) [35] as the base meta-learner, with several baselines as well as

the paragon model that uses the same architecture but is free to re-process all past data along with new data. In Sect. 3.3 we assess performance on standard few-shot image classification benchmarks (MiniImageNet and TieredImageNet) as well as on a newly curated dataset described in Sect. 3.2. To show that our method is not tied to the specifics of PN, we also perform the same experiments using ECM [25]. That is the basis for extending our simplest method to the case where each task has a different number of classes, described in the appendix.

Implementation Details: We use a ResNet-12 [12] following [23] as our feature extractor ϕ_w. It consists of four residual layers each with 3×3 convolutional layers followed by a max-pooling layer. We use DropBlock regularization [9], a form of structured dropout with a keep-rate of 0.9 after the max-pooling layers. At each round, we train for 200 epochs, each consisting of 800 few-shot training tasks containing 5 (1) support examples per class for 5-shot 5-way (1-shot 5-way). We use 15 query points per class for computing the loss to update the network parameters. Test performance is also measured with 15 query points per class. We use Adam [16] with an initial learning-rate of 0.001 which is reduced by a factor of 0.5 when performance on the validation set does not improve for more than 3 epochs. We use cross-entropy loss with softmax temperature 2.0, following [20]. For IDA, we choose λ to be 1.0 and we show the effect of varying λ in the range of [0.0, 10.0] in the appendix.

3.1 Baselines and Ablation Studies

To evaluate the method quantitatively, we need an upper-bound (oracle) represented by a model that performs meta-training using all the data as well as few other baselines to enable a fair comparison and ablation studies.

No Update (NU) is the simplest baseline, that is a model meta-trained only using the old dataset.

Fine-Tuning (FT) starts with the model meta-trained on old data and performs additional steps of SGD on the new data wit no additional constraint, using the first term of eq. (7).

Direct Feature Alignment (DFA) adds to the first term of eq. (7) a penalty for the direct misalignment of features (5) averaged over the new tasks

$$\mathrm{DFA}_{\mathcal{E}}(\phi_{w_{t+1}}|\phi_{w_t}) = \mathbb{E}_{x\sim\mathcal{E}^{\tau'}}\|\phi_{w_{t+1}(x)} - \phi_{w_t}(x)\|_2^2$$

akin to feature distillation.

Exemplar-based incremental meta-learning (EIML) has access to (possibly a subset of) the old data, so we can add an additional term to eq. (7) to foster tighter alignment via

$$\begin{aligned}\mathcal{L}(w_{t+1}) = & L(w_{t+1};\mathcal{E}) \\ & + \lambda\, \mathbb{E}_{x\in\mathcal{D}_\tau}\left[\,\mathrm{KL}(f^\tau_{w_t}(y|\phi_{w_t}(x))||f^\tau_{w_{t+1}}(y|\phi_{w_{t+1}}(x)))\right] \\ & + \lambda\, \mathbb{E}_{x\in\mathcal{E}_{\tau'}}\left[\,\mathrm{KL}(f^\tau_{w_t}(y|\phi_{w_t}(x))||f^\tau_{w_t}(y|\phi_{w_{t+1}}(x)))\right] \end{aligned} \tag{10}$$

where $\mathcal{E}_{\tau'}$ is task sampled from the new dataset and $\mathcal{C}_t$ and $\mathcal{C}_{t+1}$ are obtained by re-processing $\mathcal{D}_\tau$ (a task sampled from the old dataset) through the old and the new embeddings respectively. We expect this method to perform best, as it has access to old data. However, it is computationally more expensive than IDA as we need to re-process old data.

Full training paragon (PAR) consists of meta-learning using the union of data from old and new datasets, minimizing the left-hand side of eq. (1). There is no incremental training, so this method serves as an upper bound for performance.

3.2 Datasets

We test our algorithm on MiniImageNet [38], TieredImageNet [28] and another variant of ImageNet [29] which we call DomainImageNet. MiniImageNet consists of images of size 84×84 sampled from 100 classes of the ILSVRC dataset [29], with 600 images per class. We used the data split outlined in [24], where 64 classes are used for training, 16 classes for validation and 20 for testing. We further split the 64 training classes randomly into 32 for meta-training the base model and the remaining for training the incremental model; 16 validation classes are only used for assessing generalization during meta-training for both the base and incremental models. For a fair measurement of performance on the old data, we also use a separate test set comprising 300 new images per class [10].

TieredImageNet is a larger subset of ILSVRC, with $779,165$ images of size 84×84 representing 608 classes that are hierarchically grouped into 34. This dataset is split to ensure that sub-classes within the 34 groups are not shared among training, validation and test sets. The result is $448,695$ images in 351 classes for training, $124,261$ images in 97 classes for validation, and $206,209$ images in 160 classes for testing. For a fair comparison, we use the same training, validation and testing splits of [28] and use the classes at the lowest level of the hierarchy. Similar to MiniImageNet, we randomly pick 176 classes from the training set for meta-training the base model and use the remaining 175 classes for incremental meta-training. Here we also use a separate test set of about 1000 images per class for measuring old task performance.

To investigate the role of domain gap in IML, we assemble DomainImageNet, along the format of MiniImageNet, with 32 old meta-training classes, 32 new meta-training classes, 16 meta-validation classes and 40 meta-test (unseen) classes. All classes are sampled from the ILSVRC dataset, but old, new and meta-test set have two subdivisions, one sampled from *natural* categories, the other sampled from *man-made* categories. 40 unseen classes consist of 20 classes each of natural and man-made categories. The domain split we use follows [41].

3.3 Quantitative Results

We test IML on each dataset using two common few-shot scenarios: 5-shot 5-way and 1-shot 5-way. We refer to the data used to train the base model as old classes, and that of the incremental model as new classes. We refer to unseen classes as classes that the model has not seen in any training. Final performance of

Table 1: Classification accuracy on 3 different sets: tasks sampled from old, new and unseen classes of MiniImageNet using PN [35] and different IML methods.

Model	1-shot 5-way			5-shot 5-way		
	Old classes (32)	New classes (32)	Unseen classes (20)	Old classes (32)	New classes (32)	Unseen classes (20)
NU	73.84 ± 0.50	49.05 ± 0.48	50.55 ± 0.42	91.17 ± 0.18	68.35 ± 0.39	68.60 ± 0.33
FT	60.64 ± 0.49	72.61 ± 0.51	53.60 ± 0.42	82.25 ± 0.26	$\mathbf{89.63 \pm 0.22}$	72.13 ± 0.33
DFA	60.77 ± 0.49	72.23 ± 0.51	53.81 ± 0.42	82.53 ± 0.26	89.32 ± 0.22	72.07 ± 0.33
EIML	$\mathbf{68.95 \pm 0.50}$	71.43 ± 0.52	54.86 ± 0.42	$\mathbf{90.20 \pm 0.20}$	86.91 ± 0.25	74.39 ± 0.50
IDA	66.54 ± 0.49	$\mathbf{71.92 \pm 0.51}$	$\mathbf{55.52 \pm 0.43}$	89.14 ± 0.21	87.32 ± 0.25	$\mathbf{75.11 \pm 0.31}$
PAR	74.65 ± 0.49	75.85 ± 0.50	56.88 ± 0.43	91.77 ± 0.17	92.49 ± 0.17	75.27 ± 0.13

Table 2: Classification accuracy on 3 different sets: tasks sampled from old, new and unseen classes of MiniImageNet using ECM [25] and different IML methods.

Model	1-shot 5-way			5-shot 5-way		
	Old classes (32)	New classes (32)	Unseen classes (20)	Old classes (32)	New classes (32)	Unseen classes (20)
NU	73.82 ± 0.43	53.00 ± 0.43	52.77 ± 0.37	89.38 ± 0.38	71.90 ± 0.36	71.57 ± 0.36
FT	63.71 ± 0.43	75.05 ± 0.43	56.00 ± 0.38	82.90 ± 0.21	89.37 ± 0.21	74.29 ± 0.32
DFA	64.66 ± 0.42	$\mathbf{75.71 \pm 0.43}$	56.68 ± 0.39	83.37 ± 0.21	$\mathbf{89.70 \pm 0.21}$	74.69 ± 0.31
IDA	$\mathbf{72.52 \pm 0.42}$	68.43 ± 0.44	$\mathbf{57.13 \pm 0.39}$	$\mathbf{88.46 \pm 0.27}$	85.45 ± 0.27	$\mathbf{75.55 \pm 0.30}$
PAR	74.40 ± 0.40	75.74 ± 0.42	59.02 ± 0.39	89.68 ± 0.21	89.93 ± 0.21	77.60 ± 0.30

the meta-learner is reported as the mean and 95% confidence interval of the classification accuracy across 2000 episodes or few-shot tasks.

Results of the different methods using PN as a meta-learner are shown in Table 1 for MiniImageNet, Table 3 for TieredImageNet and Table 4 for DomainImageNet. Further, we show results using ECM as a meta-learner in Table 2 for MiniImageNet. We also show the results using ECM on DomainImageNet for 5-shot 5-way in Table 5. All results for DomainImageNet are using natural objects as the old domain and man-made objects as the new domain. In the appendix, we show the results for 1-shot 5-way and also for all combinations of shots and meta-learners while using man-made objects as the old domain and natural objects as the new domain.

Catastrophic Forgetting: Tables 1,2, 3, 4, 5 and 6 show that the classification accuracy on old classes using the incremental model drops significantly when compared with the base model for methods that perform IML without using the old data (i.e., FT and DFA). This holds for both 1-shot 5-way and 5-shot 5-way, both PN and ECM, and across all datasets.

Incremental Meta Learning (IML) with any of the methods described above yields increased performance on both the new classes and the unseen classes. If performance on the old classes is not a priority, any IML method will perform better on the new classes with an added bonus of better performance on unseen classes compared with the base model. Again, these conclusions hold across shots, meta-learners and datasets.

Table 3: Classification accuracy on 3 different sets: tasks sampled from old, new and unseen classes of TieredImageNet using PN [35] and different IML methods.

Model	1-shot 5-way			5-shot 5-way		
	Old classes (176)	New classes (175)	Unseen classes (160)	Old classes (176)	New classes (175)	Unseen classes (160)
NU	73.10 ± 0.52	66.18 ± 0.43	56.82 ± 0.50	89.03 ± 0.27	81.97 ± 0.37	75.78 ± 0.43
FT	71.87 ± 0.52	$\mathbf{71.03 \pm 0.52}$	58.63 ± 0.50	87.77 ± 0.29	$\mathbf{87.60 \pm 0.30}$	78.20 ± 0.42
DFA	72.03 ± 0.51	70.83 ± 0.53	$\mathbf{58.81 \pm 0.50}$	87.82 ± 0.29	87.38 ± 0.30	78.11 ± 0.42
IDA	$\mathbf{72.65 \pm 0.51}$	70.17 ± 0.53	58.71 ± 0.50	$\mathbf{89.13 \pm 0.15}$	86.91 ± 0.31	$\mathbf{78.40 \pm 0.42}$
PAR	78.57 ± 0.51	77.43 ± 0.50	61.87 ± 0.51	91.05 ± 0.24	90.44 ± 0.26	80.58 ± 0.40

EIML vs IDA: Table 1 shows that the difference in performance of between EIML and IDA is not significant. While we expected EIML to dominate IDA, in some cases EIML performed worse (Table 1: 1-shot 5-way case for MiniImageNet). This illustrates the limited benefit of re-processing old data, justifying IML. We also varied the number of samples we retained from the old dataset in the range of 15 to 120 and noticed that the performance was almost constant (shown in the appendix). Hence, we do not run tests on EIML using ECM. Furthermore, for a class of methods that learn the class anchors such as [19], running EIML is far more expensive as we need to run an additional inner optimization at every step of IML. For completeness, the performance on different datasets using EIML (with PN) is shown in the appendix.

IDA outperforms all baselines for unseen classes across all scenarios shown in this section, except for 1-shot 5-way in Table 3. We further notice better performance compared with FT and DFA for old classes. For new classes, IDA trails FT and DFA but overall it performs best on average, approaching the paragon when new tasks are sampled across old, new and unseen classes.

TieredImageNet: Table 3 shows results using PN [35] as the meta-learner. For this dataset, the improvement from using more classes is relatively small compared with MiniImageNet (Table 1). When the base model is trained with a large number of classes, the generalization ability of the network is already satisfactory, and we observe negligible catastrophic forgetting or increase in meta-learning performance. We also see that IDA is similar to the baselines. This raises the question of what new classes would best improve performance in IML. Our experiments on DomainImageNet address this question.

DomainImageNet: Results for 5-shot 5-way are shown for PN [35] in Table 4 and for ECM [25] in Table 5. The model is first trained using natural classes and then incrementally trained using man-made classes. This helps evaluate the effect of domain shift between old and new training classes. We test on five different sets: seen and unseen classes from natural objects, seen and unseen classes from man-made objects and unseen classes from a mixture of the two.

The tables show that the accuracy on the joint test set improves significantly compared with the baselines. Most of the gain is for the new domain, i.e.,, man-

Table 4: Results of 5-shot 5-way classification accuracy on different sets of DomainImageNet using PN [35] and different IML methods.

Model	Old classes from old domain (32)	New classes from new domain (32)	Unseen classes from old domain (20)	Unseen classes from new domain (20)	Unseen classes from both domains (40)
NU	86.94 ± 0.22	49.14 ± 0.36	57.66 ± 0.38	51.72 ± 0.32	59.59 ± 0.35
FT	64.42 ± 0.35	$\mathbf{84.80 \pm 0.28}$	50.72 ± 0.38	71.16 ± 0.32	65.44 ± 0.40
DFA	65.12 ± 0.35	83.95 ± 0.29	51.33 ± 0.38	70.46 ± 0.33	65.52 ± 0.40
IDA	$\mathbf{81.26 \pm 0.27}$	82.06 ± 0.30	$\mathbf{59.32 \pm 0.39}$	$\mathbf{70.61 \pm 0.32}$	$\mathbf{70.36 \pm 0.36}$
PAR	87.44 ± 0.22	88.77 ± 0.25	58.59 ± 0.37	74.46 ± 0.32	74.02 ± 0.37

Table 5: Results of 5-shot 5-way classification accuracy on different sets of DomainImageNet using ECM [25] with different IML methods.

Model	Old classes from old domain (32)	New classes from new domain (32)	Unseen classes from old domain (20)	Unseen classes from new domain (20)	Unseen classes from both domains (40)
NU	87.86 ± 0.20	56.71 ± 0.39	63.30 ± 0.38	58.10 ± 0.35	66.09 ± 0.35
FT	67.35 ± 0.34	$\mathbf{89.68 \pm 0.20}$	55.37 ± 0.38	$\mathbf{74.00 \pm 0.31}$	69.98 ± 0.39
DFA	69.33 ± 0.33	88.72 ± 0.22	57.06 ± 0.38	73.97 ± 0.31	70.77 ± 0.38
IDA	$\mathbf{86.09 \pm 0.22}$	81.82 ± 0.28	$\mathbf{64.22 \pm 0.38}$	69.92 ± 0.33	$\mathbf{72.64 \pm 0.33}$
PAR	86.83 ± 0.22	88.84 ± 0.21	65.77 ± 0.38	75.98 ± 0.31	77.31 ± 0.33

made objects. Also, catastrophic forgetting is significant since there is domain shift between the classes from the old and new domains. This effect is also seen with unseen classes on the same domain. IDA is shows improvement across the board relative to the baselines. The results for 1-shot 5-way and using the reverse domain training (i.e.,, old domain is man-made objects and incremental domain is natural objects) on the three sets for all IML algorithms show similar trends. This suggests that it matters what classes are selected for incremental training. Adding classes with diverse statistics yields maximum advantage. While we can expect this to be the trend for samples belonging to the same class, we find it to be true for samples belonging to unseen classes as well from the same domain. Our method successfully mitigates catastrophic forgetting to a large extent and performs well across different domains.

Multiple Rounds of IML: In the above experiments, our configuration consists of one old and one new dataset. In Table 6, we show the performance of different IML algorithms for a scenario where there are multiple new datasets. We split the new classes of MiniImageNet into two sets each having 16 classes (classes are split randomly) and run IML for a 5-shot 5-way setup using PN. From the table, we can observe that IDA does not incur any performance loss and achieves similar accuracy on the unseen classes compared to a single training with 32 classes. For other methods like DFA and FT, we can observe some performance drop when comparing with Table 1, which shows that IDA scales better beyond single incremental training.

Table 6: Results of 5-shot 5-way classification accuracy on MiniImageNet using PN [35] with 2 rounds of incremental meta-training, where each round consists of an 16 new classes.

Model	Incremental - Round I			Incremental - Round II		
	Old classes (32)	New classes (16)	Unseen classes (20)	Old classes (32+16)	New classes (16)	Unseen classes (20)
NU	91.17 ± 0.18	65.60 ± 0.39	68.60 ± 0.33	82.25 ± 0.37	71.45 ± 0.38	68.60 ± 0.33
FT	80.70 ± 0.31	87.67 ± 0.37	67.45 ± 0.37	76.03 ± 0.36	90.72 ± 0.23	70.57 ± 0.32
DFA	$\mathbf{87.69 \pm 0.26}$	88.43 ± 0.36	68.20 ± 0.36	80.69 ± 0.38	91.27 ± 0.21	71.19 ± 0.37
IDA	87.30 ± 0.25	$\mathbf{89.56 \pm 0.20}$	$\mathbf{72.08 \pm 0.36}$	$\mathbf{84.21 \pm 0.30}$	$\mathbf{93.25 \pm 0.17}$	$\mathbf{75.15 \pm 0.35}$
PAR	93.94 ± 0.05	93.09 ± 0.06	72.10 ± 0.13	93.03 ± 0.06	95.58 ± 0.05	75.27 ± 0.13

3.4 Limitations and Failure Cases

The implementation we chose, based on [35] and [25], and the tests we performed limit our assessment to tasks that share the same number of classes, $K = 5$, as customary in the literature. While technically not a limitation as one could always build a set of models, each for a different number of classes, and indeed it is not uncommon to train and fine-tune different models for different "ways" as seen in the literature, we use the same model for all tests. It is nonetheless desirable to have a meta-learner that can handle an arbitrary number of classes, different for each training episode. While our general framework eq. (7) enables it, our simplest implementation described in eq. (2.3) does not. However, in appendix, we describe a modified implementation that is not subject to this restriction. Since benchmarks in the literature most commonly refer to the cases $K = 1, 5$, we use the simpler model in our experiments.

Further, sampling K classes among many has low probability of yielding hard tasks that can be informative of meta-learning. Even simple classifiers can easily tell 5 random classes from ImageNet apart. *Hard task mining* could be done by selecting tasks using a distance such as [1], by sampling a random class and picking the 4 closest ones in Task2Vec space for a 5-way setup.

Finally, in our experiments we have noticed that there is still a performance gap between IDA and the paragon. The performance is matched for the case of unseen classes, but there is room for improvement in tasks sampled from new/current task distribution across shots, datasets and methods.

4 Discussion and Related Work

The natural occurrence of classes in the world is a long tailed distribution [42, 37, 39], whereby instances for most classes are rare and instances for few classes are abundant. Deep neural networks trained for classification [18, 12, 14] do not fare well when trained with small datasets typical of the tail [37], leading to increased interest in few-shot learning. Meta-learning [36, 22, 31] for few-shot learning [24, 6, 30, 38, 35, 19, 25, 10] uses a relatively "meta training" dataset from which several few-shot tasks are sampled to mimic phenomena at the tail. Once meta-trained

on the old dataset, these methods cannot take advantage of the new few-shot tasks to update the meta-learner. The obvious fix, to re-train the meta-learner every time a few-shot task arises, is impractical if at all possible, as one may not have access to all past data.

Incremental learning, or continual learning, is typically performed by adapting a neural network model, trained using some dataset, using a new dataset, to arrive at a single model. The main challenge here is to prevent catastrophic forgetting [8]. A few relevant works in this area include [26, 20, 21, 4, 17, 33]. To learn classifiers in a class-incremental way where new classes are added progressively, [26] proposed to keep exemplars from old classes based on their representation power and the computational budget. [20] used knowledge distillation to preserve the model's capabilities on the old tasks when only new data is accessible. [21] and its extension [4] leveraged a small episodic memory to alleviate forgetting. [17] slowed down learning on the weights that were important for old tasks, while [33] extended it by training a knowledge-base in alternating phases.

Methods that used few-shot incremental sets such as [10] unified the recognition of both new and old classes using attention based few-shot classification. Similarly, [27] used recurrent back-propagation to train a set of new weights to achieve good overall classification on both old and new classes. [40, 34] extended this class-incremental framework to other visual recognition tasks like semantic segmentation and attribute recognition. Accordingly, despite being called incremental few-shot learning, these methods are accurately described as incremental learning using few-shot datasets.

On-line meta-learning [7] can be done by exposing an agent to new tasks in a sequential manner. One may see this experimental setup to be similar to ours; however, unlike IML, [7] retains data from all previous tasks and leverages it for meta-training, thus forgoing incremental learning. In our setup, we retain minimal amounts of data from the old training set. [15, 11] on the other hand, tackled a continual/online meta-learning setup where an explicit delineation between different tasks is not available, whereas in our experimental setup we are primarily trying to solve new classification tasks with clear task boundaries.

To summarize, there are several approaches to solve IML: One that biases new weights to remain similar to those of the base model (elastic weight consolidation) as in eq. (2), and one that looks at function space and imposes that the activations remain similar to that of the base model (knowledge distillation), as in eq. (4). We adopt the latter and empirically test how our general framework performs in the case of two metric-based meta-learners [35, 25]. This is a particular instance of the right-hand side of eq. (1), that was our starting point in Sect. 2.1. It yields empirical performance comparable to meta-learning on the union of the old and new datasets, which is the gold standard. This gives empirical validation to our method, and to the many possible variants that can be explored considering combinations of meta and few-shot set, choices of metrics, classifiers, divergence measures, and a myriad of other ingredients in the IML recipe to minimize eq. (7). We have tested several options in our experiments and in the appendix, and many more are open for investigation in future work.

References

1. Achille, A., Lam, M., Tewari, R., Ravichandran, A., Maji, S., Fowlkes, C.C., Soatto, S., Perona, P.: Task2vec: Task embedding for meta-learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6430–6439 (2019)
2. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2654–2662. Curran Associates, Inc. (2014), http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf
3. Bengio, S., Bengio, Y., Cloutier, J., Gecsei, J.: On the optimization of a synaptic learning rule. In: Preprints Conf. Optimality in Artificial and Biological Neural Networks. vol. 2. Univ. of Texas (1992)
4. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. In: International Conference on Learning Representations (2019)
5. Dhillon, G.S., Chaudhari, P., Ravichandran, A., Soatto, S.: A baseline for few-shot image classification. arXiv preprint arXiv:1909.02729 (2019)
6. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1126–1135. JMLR. org (2017)
7. Finn, C., Rajeswaran, A., Kakade, S., Levine, S.: Online meta-learning. arXiv preprint arXiv:1902.08438 (2019)
8. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
9. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. In: Advances in Neural Information Processing Systems. pp. 10727–10737 (2018)
10. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4367–4375 (2018)
11. Harrison, J., Sharma, A., Finn, C., Pavone, M.: Continuous meta-learning without tasks. arXiv preprint arXiv:1912.08866 (2019)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. NIPS 2014 Deep Learning Workshop (2015)
14. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
15. Jerfel, G., Grant, E., Griffiths, T., Heller, K.A.: Reconciling meta-learning and continual learning with online mixtures of tasks. In: Advances in Neural Information Processing Systems. pp. 9119–9130 (2019)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference

on Neural Information Processing Systems - Volume 1. pp. 1097–1105. NIPS'12 (2012)
19. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 10657–10665 (2019)
20. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
21. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: Advances in Neural Information Processing Systems. pp. 6467–6476 (2017)
22. Naik, D.K., Mammone, R.J.: Meta-neural networks that learn by learning. In: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. vol. 1, pp. 437–442. IEEE (1992)
23. Oreshkin, B., López, P.R., Lacoste, A.: Tadam: Task dependent adaptive metric for improved few-shot learning. In: Advances in Neural Information Processing Systems. pp. 721–731 (2018)
24. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: ICLR 2017 (2017)
25. Ravichandran, A., Bhotika, R., Soatto, S.: Few-shot learning with embedded class models and shot-free meta training. In: International Conference on Computer Vision (2019)
26. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
27. Ren, M., Liao, R., Fetaya, E., Zemel, R.S.: Incremental few-shot learning with attention attractor networks. arXiv preprint arXiv:1810.07218 (2018)
28. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676 (2018)
29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
30. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. arXiv preprint arXiv:1807.05960 (2018)
31. Schmidhuber, J.: Evolutionary principles in self referential learning. On learning how to learn: The meta-meta-... hook. Diploma thesis, Institut f. Informatik, Tech. Univ. Munich (1987)
32. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning. pp. 1889–1897 (2015)
33. Schwarz, J., Luketina, J., Czarnecki, W.M., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International Conference on Machine Learning (2018)
34. Siam, M., Oreshkin, B.: Adaptive masked weight imprinting for few-shot segmentation. arXiv preprint arXiv:1902.11123 (2019)
35. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems. pp. 4077–4087 (2017)
36. Thrun, S., Pratt, L.: Learning to learn. Springer Science & Business Media (2012)
37. Van Horn, G., Perona, P.: The devil is in the tails: Fine-grained classification in the wild. arXiv preprint arXiv:1709.01450 (2017)

38. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in neural information processing systems. pp. 3630–3638 (2016)
39. Wang, Y.X., Ramanan, D., Hebert, M.: Learning to model the tail. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 7029–7039. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7278-learning-to-model-the-tail.pdf
40. Xiang, L., Jin, X., Ding, G., Han, J., Li, L.: Incremental few-shot learning for pedestrian attribute recognition. arXiv preprint arXiv:1906.00330 (2019)
41. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 3320–3328. Curran Associates, Inc. (2014), http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf
42. Zhu, X., Anguelov, D., Ramanan, D.: Capturing long-tail distributions of object subcategories. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 915–922. CVPR '14, IEEE Computer Society, Washington, DC, USA (2014). https://doi.org/10.1109/CVPR.2014.122, https://doi.org/10.1109/CVPR.2014.122