Dynamic Few-Shot Visual Learning without Forgetting

Spyros Gidaris
University Paris-Est, LIGM
Ecole des Ponts ParisTech
spyros.gidaris@enpc.fr

Nikos Komodakis
University Paris-Est, LIGM
Ecole des Ponts ParisTech
nikos.komodakis@enpc.fr

Abstract

The human visual system has the remarkably ability to be able to effortlessly learn novel concepts from only a few examples. Mimicking the same behavior on machine learning vision systems is an interesting and very challenging research problem with many practical advantages on real world vision applications. In this context, the goal of our work is to devise a few-shot visual learning system that during test time it will be able to efficiently learn novel categories from only a few training data while at the same time it will not forget the initial categories on which it was trained (here called base categories). To achieve that goal we propose (a) to extend an object recognition system with an attention based few-shot classification weight generator, and (b) to redesign the classifier of a ConvNet model as the cosine similarity function between feature representations and classification weight vectors. The latter, apart from unifying the recognition of both novel and base categories, it also leads to feature representations that generalize better on "unseen" categories. We extensively evaluate our approach on Mini-ImageNet where we manage to improve the prior state-of-the-art on few-shot recognition (i.e., we achieve 56.20% and 73.00% on the 1-shot and 5-shot settings respectively) while at the same time we do not sacrifice any accuracy on the base categories, which is a characteristic that most prior approaches lack. Finally, we apply our approach on the recently introduced few-shot benchmark of Bharath and Girshick [4] where we also achieve state-of-the-art results.

1. Introduction

Over the last few years, deep convolutional neural networks [9, 21, 23, 5] (ConvNets) have achieved impressive results on image classification tasks, such as object recognition [18] or scene classification [27]. In order for a ConvNet to successfully learn to recognize a set of visual categories (e.g., object categories or scene types), it requires to manually collect and label thousands of training examples per target category and to apply on them an iterative gradient based optimization routine [10] that is extremely computationally expensive, e.g., it can consume hundreds or even thousands of GPU hours. Moreover, the set of categories that the ConvNet model can recognize remains fixed after training. In case we would like to expand the set of categories that the ConvNet can recognize, then we need to collect training data for the novel categories (i.e., those that they were not in the initial training set) and restart the aforementioned computationally costly training procedure this time on the enhanced training set such that we will avoid catastrophic interference. Even more, it is of crucial importance to have enough training data for the novel categories (e.g., thousands of examples per category) otherwise we risk overfitting on them.

In contrast, the human visual system exhibits the remarkably ability to be able to effortlessly learn novel concepts from only one or a few examples and reliably recognize them later on. It is assumed that the reason the human visual system is so efficient when learning novel concepts is that it exploits its past experiences about the (visual) world. For example, a child, having accumulated enough knowledge about mammal animals and in general the visual world, can easily learn and generalize the visual concept of "rhinoceros" from only a single image. Mimicking that behavior on artificial vision systems is an interesting and very challenging research problem with many practical advantages, such as developing real-time interactive vision applications for portable devices (e.g., cell-phones).

Research on this subject is usually termed *few-shot learning*. However, most prior methods neglect to fulfill two very important requirements for a good few-shot learning system: **(a)** the learning of the novel categories needs to be fast, and **(b)** to not sacrifice any recognition accuracy on the initial categories that the ConvNet was trained on, i.e., to not "forget" (from now on we will refer to those initial categories by calling them base categories). Motivated by this observation,

in this work we propose to tackle the problem of few-shot learning under a more realistic setting, where a large set of training data is assumed to exist for a set of base categories and, using these data as the sole input, we want to develop an object recognition learning system that, not only is able to recognize these base categories, but also learns to dynamically recognize novel categories from only a few training examples (provided only at test time) while also not forgetting the base ones or requiring to be re-trained on them (*dynamic few-shot learning without forgetting*). Compared to prior approaches, we believe that this setting more closely resembles the human visual system behavior (w.r.t. how it learns novel concepts). In order to achieve our goal, we propose two technical novelties.

Few-shot classification-weight generator based on attention. A typical ConvNet based recognition model, in order to classify an image, first extracts a high level feature representation from it and then computes per category classification scores by applying a set of classification weight vectors (one per category) to the feature. Therefore, in order to be able to recognize novel categories we must be able to generate classification weight vectors for them. In this context, the first technical novelty of our work is that we enhance a typical object recognition system with an extra component, called *few-shot classification weight generator* that accepts as input a few training examples of a novel category (e.g., no more than five examples) and, based on them, generates a classification weight vector for that novel category. Its key characteristic is that in order to compose novel classification weight vectors, it explicitly exploits the acquired past knowledge about the visual world by incorporating an attention mechanism over the classification weight vectors of the base categories. This attention mechanism offers a significant boost on the recognition performance of novel categories, especially when there is only a single training example available for learning them.

Cosine-similarity based ConvNet recognition model. In order for the *few-shot classification weight generator* to be successfully incorporated into the rest of the recognition system, it is essential the ConvNet model to be able to simultaneously handle the classification weight vectors of both base and novel categories. However, as we will explain in the methodology, this is not feasible with the typical dot-product based classifier (i.e., the last linear layer of a classification neural network). Therefore, in order to overcome this serious issue, our second technical novelty is to implement the classifier as a cosine similarity function between the feature representations and the classification weight vectors. Apart from unifying the recognition of both base and novel categories, features learned with the cosine-similarity based classifier turn out to generalize significantly better on novel categories than those learned with a dot-product based classifier. Moreover, we demonstrate in the experimental section

that, by simply training a cosine-similarity based ConvNet recognition model, we are able to learn feature extractors that when used for image matching they surpass prior state-of-the-art approaches on the few-shot recognition task.

To sum up, our contributions are: **(1)** We propose a few-shot object recognition system that is capable of dynamically learning novel categories from only a few training data while at the same time does not forget the base categories on which it was trained. **(2)** In order to achieve that we introduced two technical novelties, an attention based few-shot classification weight generator, and to implement the classifier of a ConvNet model as a cosine similarity function between feature representations and classification vectors. **(3)** We extensively evaluate our object recognition system on Mini-ImageNet, both w.r.t. its few-shot object recognition performance and its ability to not forget the base categories, and we report state-of-the-art results that surpass prior approaches by a very significant margin. **(4)** Finally, we apply our approach on the recently introduced fews-shot benchmark of Bharath and Girshick [4] where we achieve state-of-the-art results.

In the following sections, we provide related work in §2, we describe our few-shot object learning methodology in §3, we provide experimental results in §4, and finally we conclude in §5.

2. Related work

Recently, there is resurgence of interest on the few-shot learning problem. In the following we briefly discuss the most relevant approaches to our work.

Meta-learning based approaches. Meta-learning approaches typical involve a meta-learner model that given a few training examples of a new task it tries to quickly learn a learner model that "solves" this new task [20, 24, 1, 13, 19]. Specifically, Ravi and Larochelle [16] propose a LSTM [6] based meta-learner that is trained given as input a few training examples of a new classification task to sequentially generate parameter updates that will optimize the classification performance of a learner model on that task. Their LSTM also learns the parameter initialization of the learner model. Finn *et al*. [3] simplified the above meta-learner model and only learn the initial learner parameters such that only a few gradient descent steps w.r.t. those initial parameters will achieve the maximal possible performance on the new task. Mishra *et al*. [12] instead propose a generic temporal convolutional network that given as input a sequence of a few labeled training examples and then an unlabeled test example, it predicts the label of that test example. Our system also includes a meta-learner network component, the few-shot classification weight generator.

Metric-learning based approaches. In general, metric learning approaches attempt to learn feature representations that preserve the class neighborhood structure (i.e., features of the same object are closer than features of different ob-

jects). Specifically, Koch *et al.* [8] formulated the one-shot object recognition task as image matching and train Siamese neural networks to compute the similarity between a training example of a novel category and a test example. Vinyals *et al.* [25] proposed Matching Networks that in order to classify a test example it employs a differentiable nearest neighbor classifier implemented with an attention mechanism over the learned representations of the training examples. Prototypical Networks [22] learn to classify test examples by computing distances to prototype feature vectors of the novel categories. They propose to learn the prototype feature vector of a novel category as the average of the feature vectors extracted by the training examples of that category. A similar approach was proposed before by Mensink *et al.* [11] and Prototypical Networks can be viewed as an adaption of that work for ConvNets. Despite their simplicity, Prototypical Networks demonstrated state-of-the-art performance. Our few-shot classification weight generator also includes a feature averaging mechanism. However, more than that, it also explicitly exploits past knowledge about the visual world with an attention based mechanism and the overall framework allows to perform unified recognition of both base and novel categories without altering the way base categories are learnt and recognized.

In a different line of work, Bharath and Girshick [4] propose to use during training a l_2 regularization loss on the feature representations that forces them to better generalize on "unseen" categories. In our case, the cosine-similarity based classifier, apart from unifying the recognition of both base and novel categories, it also leads to feature representations that are able to better generalize on "unseen" categories. Also, their framework is able to recognize both base and novel categories as ours. However, to achieve that goal they re-train the classifier on both the base categories (with a large set of training data) and the novel categories (with few training data), which is in general slow and requires constantly maintaining in disc a large set of training data.

3. Methodology

As an input to our object recognition learning system we assume that there exists a dataset $D_{train} = \bigcup_{b=1}^{K_{base}} \{x_{b,i}\}_{i=1}^{N_b}$ of K_{base} base categories, where N_b is the number of training examples of the b-th category and $x_{b,i}$ is its i-th training example. Using this as the only input, the goal of our work is to be able to both learn to accurately recognize base categories and to learn to perform few-shot learning of novel categories in a dynamic manner and without forgetting the base ones. An overview of our framework is provided in Figure 1. It consists of two main components, a *ConvNet-based recognition model* that is able to recognize both base and novel categories and a *few-shot classification weight generator* that dynamically generates classification weight vectors for the novel categories at test time:
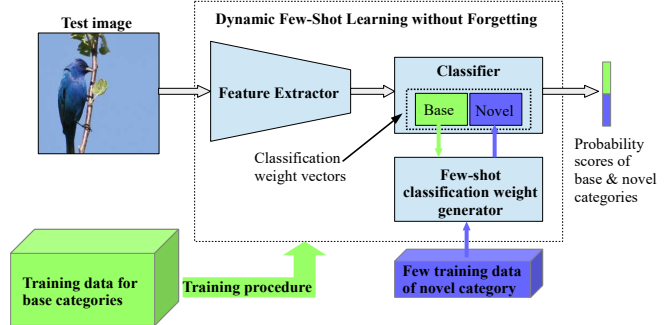

Figure 1: Overview of our system. It consists of: (a) a *ConvNet based recognition model* (that includes a feature extractor and a classifier) and (b) a *few-shot classification weight generator*. Both are trained on a set of base categories for which we have available a large set of training data. During test time, the weight generator gets as input a few training data of a novel category and the classification weight vectors of base categories (green rectangle inside the classifier box) and generates a classification weight vector for this novel category (blue rectangle inside the classifier box). This allows the ConvNet to recognize both base and novel categories.

ConvNet-based recognition model. It consists of **(a)** a feature extractor $F(.|\theta)$ (with learnable parameters θ) that extracts a d-dimensional feature vector $z = F(x|\theta) \in \mathbb{R}^d$ from an input image x, and **(b)** a classifier $C(.|W^*)$, where $W^* = \{w_k^* \in \mathbb{R}^d\}_{k=1}^{K^*}$ are a set of K^* classification weight vectors - one per object category, that takes as input the feature representation z and returns a K^*-dimensional vector with the probability classification scores $p = C(z|W^*)$ of the K^* categories. Note that in a typical convolutional neural network the feature extractor is the part of the network that starts from the first layer and ends at the last hidden layer while the classifier is the last classification layer. During the single training phase of our algorithm, we learn the θ parameters and the classification weight vectors of the base categories $W_{base} = \{w_k\}_{k=1}^{K_{base}}$ such that by setting $W^* = W_{base}$ the ConvNet model will be able to recognize the base object categories.

Few-shot classification weight generator. This comprises a meta-learning mechanism that, during test time, takes as input a set of K_{novel} novel categories with few training examples per category $D_{novel} = \bigcup_{n=1}^{K_{novel}} \{x'_{n,i}\}_{i=1}^{N'_n}$, where N'_n is the number of training examples of the n-th novel category and $x'_{n,i}$ is its i-th training example, and is able to dynamically assimilate the novel categories on the repertoire of the above ConvNet model. More specifically, for each novel category $n \in [1, N_{novel}]$, the few-shot classification weight generator $G(.,.|\phi)$ gets as input the feature vectors $Z'_n = \{z'_{n,i}\}_{i=1}^{N'_n}$ of its N'_n training examples, where $z'_{n,i} = F(x'_{n,i}|\theta)$, and the classification weight vectors of the base categories W_{base} and generates a classification weight vector $w'_n = G(Z'_n, W_{base}|\phi)$ for that novel category. Note that ϕ are the learnable parameters of the few-shot weight

generator, which are learned during the single training phase of our framework. Therefore, if $W_{novel} = \{w'_n\}_{n=1}^{K_{novel}}$ are the classification weight vectors of the novel categories inferred by the few-shot weight generator, then by setting $W^* = W_{base} \cup W_{novel}$ on the classifier $C(.|W^*)$ we enable the ConvNet model to recognize both base and novel categories.

A key characteristic of our framework is that it is able to effortlessly (i.e., quickly during test time) learn novel categories and at the same time recognize both base and novel categories in a unified manner. In the following subsections, we will describe in more detail the ConvNet-based recognition model in §3.1 and the few-shot weight generator in §3.2. Finally, we will explain the training procedure in §3.3.

3.1. Cosine-similarity based recognition model

A crucial difference of our ConvNet based recognition model compared to a standard one is that it should be able to dynamically incorporate at test time a variable number of novel categories (through the few-shot weight generator).

The standard setting for classification neural networks is, after having extracted the feature vector z, to estimate the classification probability vector $p = C(z|W^*)$ by first computing the raw classification score s_k of each category $k \in [1, K^*]$ using the dot-product operator $s_k = z^\mathsf{T} w_k^*$, where w_k is the k-th classification weight vector in W^*, and then applying the softmax operator across all the K^* classification scores, i.e., $p_k = softmax(s_j)$, where p_k is the k-th classification probability of p. In our case the classification weight vectors w_k^* could come both from the base categories, i.e., $w_k^* \in W_{base}$, and the novel categories, i.e., $w_k^* \in W_{novel}$. However, the mechanisms involved during learning those classification weights are very different. The base classification weights, starting from their initial state, are slowly modified (i.e., slowly learned) with small SGD steps and thus their magnitude changes slowly over the course of their training. In contrast, the novel classification weights are dynamically predicted (i.e., quickly learned) by the weight generator based on the input training feature vectors and thus their magnitude depends on those input features. Due to those differences, the weight values in those two cases (i.e., base and novel classification weights) can be completely different, and so the same applies to the raw classification scores computed with the dot-product operation, which can thus have totally different magnitudes depending on whether they come from the base or the novel categories. This can severely impede the training process and, in general, does not allow to have a unified recognition of both type of categories. In order to overcome this critical issue, we propose to modify the classifier $C(.|W^*)$ and compute the raw classification scores using the cosine similarity operator:

$$s_k = \tau \cdot cos(z, w_k^*) = \tau \cdot \overline{z}^\mathsf{T} \overline{w}_k^* , \qquad (1)$$

where $\overline{z} = \frac{z}{\|z\|}$ and $\overline{w}_k^* = \frac{w_k^*}{\|w_k^*\|}$ are the l_2-normalized vec-

tors (from now on we will use the overline symbol $\overline{z}$ to indicate that a vector z is l_2-normalized), and τ is a learnable scalar value[1]. Since the cosine similarity can be implemented by first l_2-normalizing the feature vector z and the classification weight vector w_k^* and then applying the dot-product operator, the absolute magnitudes of the classification weight vectors can no longer affect the value of the raw classification score (as a result of the l_2 normalization that took place).

In addition to the above modification, we also choose to remove the ReLU non-linearity [14] after the last hidden layer of the feature extractor, which allows the feature vector z to take both positive and negative values, similar to the classification weight vectors. Note that the removal of the ReLU non-linearity does not make the composition of the last hidden layer with the classification layer a linear operation, since we l_2-normalize the feature vectors, which is a non-linear operation. In our initial experiments with the cosine similarity based classifier we found that such a modification can significantly improve the recognition performance of novel categories.

We note that, although cosine similarity is already well established as an effective similarity function for classifying a test feature by comparing it with the available training features vectors [25, 11, 17], in this work we use it for a different purpose, i.e., to replace the dot-product operation of the last linear layer of classification ConvNets used for applying the learnable weights of that layer to the test feature vectors. The proposed modification in the architecture of a classification ConvNet allows to unify the recognition of base and novel categories without significantly altering the classification pipeline for the recognition of base categories (in contrast to [11, 17]). To the best of our knowledge, employing the cosine similarity operation in such a way is novel in the context of few shot learning. Interestingly, concurrently to us, Qi *et al.* [15] also propose to use the cosine similarity function in a similar way for the few-shot learning task. In a different line of work, very recently Chunjie *et al.* [2] also explored cosine similarity for the typical supervised classification task.

Advantages of cosine-similarity based classifier. Apart from making possible the unified recognition of both base and novel categories, *the cosine-similarity based classifier leads the feature extractor to learn features that generalize significantly better on novel categories than features learned with the dot-product based classifier*. A possible explanation for this is that, in order to minimize the classification loss of a cosine-similarity based ConvNet model, the l_2-normalized feature vector of an image must be very closely matched with

---

[1] The scalar parameter τ is introduced in order to control the peakiness of the probability distribution generated by the softmax operator since the range of the cosine similarity is fixed to $[-1, 1]$. In all of our experiments τ is initialized to 10.

the l_2-normalized classification weight vector of its ground truth category. As a consequence, the feature extractor is forced to (a) learn to encode on its feature activations exactly those discriminative visual cues that also the classification weight vectors of the ground truth categories learn to look for, and (b) learn to generate l_2-normalized feature vectors with low intraclass variance, since all the feature vectors that belong to the same category must be very closely matched with the single classification weight vector of that category (see also §1 of supplementary material for t-SNE scatter plots of the cosine similarity based and dot product based features). Moreover, our cosine-similarity based classification objective resembles the training objectives typically used by metric learning approaches [7]. In fact, it turns out that *our feature extractor trained solely on cosine-similarity based classification of base categories, when used for image matching, it manages to surpass all prior state-of-the-art approaches on the few-shot object recognition task.*

3.2. Few-shot classification weight generator

The few-shot classification weight generator $G(.,.|\phi)$ gets as input the feature vectors $Z' = \{z'_i\}_{i=1}^{N'}$ of the N' training examples of a novel category (typically $N' \leq 5$) and (optionally) the classification weight vectors of the base categories W_{base}. Based on them, it infers a classification weight vector $w' = G(Z', W_{base}|\phi)$ for that novel category. Here we explain how the above few-shot classification weight generator is constructed.

Feature averaging based weight inference. Since, as we explained in section § 3.1, the cosine similarity based classifier of the ConvNet model forces the feature extractor to learn feature vectors that form compact category-wise clusters and the classification weight vectors to learn to be representative feature vectors of those clusters, an obvious choice is to infer the classification weight vector w' by averaging the feature vectors of the training examples (after they have been l_2-normalized): $w'_{avg} = \frac{1}{N'} \sum_{i=1}^{N'} \overline{z}'_i$. The final classification weight vector in case we only use the feature averaging mechanism is: $w' = \phi_{avg} \odot w'_{avg}$, where $\odot$ is the Hadamard product, and $\phi_{avg} \in \mathbb{R}^d$ is a learnable weight vector. Similar strategy has been previously proposed by Snell *et al.* [22] and has demonstrated very good results. However, it does not fully exploit the knowledge about the visual world that the ConvNet model acquires during its training phase. Furthermore, in case there is only a single training example for the novel category, the averaging cannot infer an accurate classification weight vector.

Attention-based weight inference. We enhance the above feature averaging mechanism with an attention based mechanism that composes novel classification weight vectors by "looking" at a memory that contains the base classification weight vectors $W_{base} = \{w_b\}_{b=1}^{K_{base}}$. More specifically, an extra attention-based classification weight vector w'_{att} is

computed as:

$$w'_{att} = \frac{1}{N'} \sum_{i=1}^{N'} \sum_{b=1}^{K_{base}} Att(\phi_q \overline{z}'_i, k_b) \cdot \overline{w}_b, \quad (2)$$

where $\phi_q \in \mathbb{R}^{d \times d}$ is a learnable weight matrix that transforms the feature vector $\overline{z}'_i$ to query vector used for querying the memory, $\{k_b \in \mathbb{R}^d\}_b^{K_{base}}$ is a set of K_{base} learnable keys (one per base category) used for indexing the memory, and $Att(.,.)$ is an attention kernel implemented as a cosine similarity function[2] followed by a softmax operation over the K_{base} base categories. The final classification weight vector is computed as a weighted sum of the average based classification vector w'_{avg} and the attention based classification vector w'_{att}, $w' = \phi_{avg} \odot w'_{avg} + \phi_{att} \odot w'_{att}$, where $\odot$ is the Hadamard product, and $\phi_{avg}, \phi_{att} \in \mathbb{R}^d$ are learnable weight vectors.

Why using an attention-based weight composition? Thanks to the cosine-similarity based classifier, the base classification weight vectors learn to be representative feature vectors of their categories. Thus, the base classification weight vectors also encode visual similarity, e.g., the classification vector of a mammal animal should be closer to the classification vector of another mammal animal rather than the classification vector of a vehicle. Therefore, the classification weight vector of a novel category can be composed as a linear combination of those base classification weight vectors that are most similar to the few training examples of that category. This allows our few-shot weight generator to explicitly exploit the acquired knowledge about the visual word (here represented by the base classification weight vectors) in order to improve the few-shot recognition performance. This improvement is very significant especially in the one-shot recognition setting where averaging cannot provide an accurate classification weight vector.

3.3. Training procedure

In order to learn the ConvNet-based recognition model (i.e. the feature extractor $F(.|\theta)$ as well as the classifier $C(.|W^*)$) and the few-shot classification weight generator $G(.,.|\phi)$, we use as the sole input a training set $D_{train} = \bigcup_{b=1}^{K_{base}} \{x_{b,i}\}_{i=1}^{N_b}$ of K_{base} base categories. We split the training procedure into 2 stages and at each stage we minimize a different cross-entropy loss of the following form:

$$\frac{1}{K_{base}} \sum_{b=1}^{K_{base}} \frac{1}{N_b} \sum_{i=1}^{N_b} loss(x_{b,i}, b), \quad (3)$$

where $loss(x, y)$ is the negative log-probability $-log(p_y)$ of the y-th category in the probability vector $p = C(F(x|\theta)|W^*)$. The meaning of W^* is different on each of the training stages, as we explain below.

---

[2]The cosine similarity scores are also scaled by a learnable scalar parameter γ in order to increase the peakiness of the softmax distribution.

1st training stage: During this stage we only learn the ConvNet recognition model without the few-shot classification weight generator. Specifically, at this stage we learn the parameters θ of the feature extractor $F(.|\theta)$ and the base classification weight vectors $W_{base} = \{w_b\}_{b=1}^{K_{base}}$. This is done in exactly the same way as for any other standard recognition model. In this case W^* is equal to the base classification weight vectors W_{base}.

2nd training stage: During this stage we train the learnable parameters ϕ of the few-shot classification weight generator while we continue training the base classification weight vectors W_{base} (in our experiments during that training stage we freezed the feature extractor). In order to train the few-show classification weight generator, in each batch we randomly pick K_{novel} "fake" novel categories from the base categories and we treat them in the same way as we will treat the actual novel categories after training. Specifically, instead of using the classification weight vectors in W_{base} for those "fake" novel categories, we sample N' training examples (typically $N' \leq 5$) for each of them, compute their feature vectors $Z' = \{z_i'\}_{i=1}^{N'}$, and give those feature vectors to the few-shot classification weight generator $G(., .|\phi)$ in order to compute novel classification weight generators. The inferred classification weight vectors are used for recognizing the "fake" novel categories. Everything is trained end-to-end. Note that we take care to exclude from the base classification weight vectors that are given as a second argument to the few-shot weight generator $G(., .|\phi)$ those classification vectors that correspond to the "fake" novel categories. In this case W^* is the union of the "fake" novel classification weight vectors generated by $G(., .|\phi)$ and the classification weight vectors of the remaining base categories. More implementation details of this training stage are provided in §2 of supplementary material.

4. Experimental results

We extensively evaluate the proposed few-shot recognition system w.r.t. both its few-shot recognition performance of novel categories and its ability to not "forget" the base categories on which it was trained.

4.1. Mini-ImageNet experiments

Evaluation setting for recognition of novel categories. We evaluate our few-shot object recognition system on the Mini-ImageNet dataset [25] that includes 100 different categories with 600 images per category, each of size 84×84. For our experiments we used the splits by Ravi and Laroche [16] that include 64 categories for training, 16 categories for validation, and 20 categories for testing. The typical evaluation setting on this dataset is first to train a few-shot model on the training categories and then during test time to use the validation (or the test) categories in order to form few-shot tasks on which the trained model is evaluated. Those few-shot

tasks are formed by first sampling K_{novel} categories and one or five training example per category (1-shot and 5-shot settings respectively), which the trained model uses for meta-learning those categories, and then evaluating it on some test examples that come from the same novel categories but do not overlap with the training examples.

Evaluation setting for the recognition of the base categories. When we evaluate our model w.r.t. few-shot recognition task on the validation / test categories, we consider as base categories the 64 training categories on which we trained the model. Since the proposed few-shot object recognition system has the ability to not forget the base categories, we would like to also evaluate the recognition performance of our model on those base categories. Therefore, we sampled 300 extra images for each training category that we use as validation image set for the evaluation of the recognition performance of the base categories and also another 300 extra images that are used for the same reason as test image set. Therefore, when we evaluate our model w.r.t. the few-shot learning task on the validation / test categories we also evaluate w.r.t. recognition performance of the base categories on the validation / test image set of the training categories.

4.1.1 Ablation study

In Table 1 we provide an ablation study of the proposed object recognition framework on the validation set of mini-ImageNet. We also compare with two prior state-of-the-art approaches, Prototypical Networks [22] and Matching Nets [25], that we re-implemented ourselves in order to ensure a fair comparison. The feature extractor used in all cases is a ConvNet model that has 4 convolutional modules, with 3×3 convolutions, followed by batch normalization, ReLU nonlinearity[3], and 2×2 max-pooling. Given as input images of size 84×84 it yields feature maps with spatial size 5×5. The first two convolutional layers have 64 feature channels and the latter two have 128 feature channels.

Cosine-similarity based ConvNet model. First we examine the performance of the cosine-similarity based ConvNet recognition model (entry Cosine Classifier) without training the few-shot classification weight generator (i.e., we only perform the 1st training stage as was described in section 3.3). In order to test its performance on the novel categories, during test time we estimate classification weight vectors using feature averaging. *We want to stress out that in this case there are no learnable parameters involved in the generation of the novel classification weight vectors and also the ConvNet model it was never trained on the few-shot recognition task. Despite that, the features learned by*

---

[3]Unless otherwise stated, our cosine-similarity based models as well as the re-implementation of Matching-Nets do not have a ReLU nonlinearity after the last convolutional layer, since in both cases this modification improved the recognition performance on the few-shot recognition task

the cosine-similarity based ConvNet model matches or even surpasses the performance of the Matching-Nets and Prototypical Networks, which are explicitly trained on the few-shot object recognition task.* By comparing the cosine-similarity based ConvNet models (Cosine Classifier entries) with the dot-product based models (Dot Product entries) we observe that the former drastically improve the few-shot object recognition performance, which means that the feature extractor that is learned with the cosine-similarity classifier generalizes significantly better on "unseen" categories than the feature extractor learned with the dot-product classifier. Notably, the cosine-similarity classifier significantly improves also the recognition performance on the base categories.

Removing the last ReLU unit. In our work we propose to remove the last ReLU non-linearity from the feature extractor when using a cosine classifier. Instead, keeping the ReLU units (Cosine w/ ReLU entries) decreases the accuracy on novel categories while increasing it on base categories.

Few-shot classification weight generator. Here we examine the performance of our system when we also incorporate on it the proposed few-shot classification weight generator. In Table 1 we provide two solutions for the few-shot weight generator: the entry Cosine Classifier & Avg. Weight Gen that uses only the feature averaging mechanism described in section 3.2 and the entry Cosine Classifier & Att. Weight Gen that uses both the feature averaging and the attention based mechanism. Both types of few-shot weight generators are trained during the 2nd training stage that is described in section 3.3. We observe that both of them offer a very significant boost on the few-shot recognition performance of the cosine similarity based model (entry Cosine Classifier). Among the two, the attention based solution exhibits better few-shot recognition behavior, especially in the 1-shot setting where it has more than 3 percentage points higher performance. Also, it is easy to see that the few-shot classification weight generator does not affect the recognition performance of the base categories, which is around 70.50% in all the cosine-similarity based models. Moreover, by introducing the few-shot weight generator, the recognition performance in both type of categories (columns Both) increases significantly, which means that the ConvNet model achieves better behavior w.r.t. our goal of unified recognition of both base and novel categories. The few-shot recognition performance of our full system, which is the one that includes the attention based few-shot weight generator (entry Cosine classifier & Att. Weight Gen), offers a very significant improvement w.r.t. the prior state-of-the-art approaches on the few-shot object recognition task, i.e., from 72.67% to 74.92% in the 5-shot setting and from 55.53% to 58.55% in the 1-shot setting. Also, our system achieves significantly higher performance on the recognition of base categories compared to Prototypical Networks[4].

---

[4]In order to recognize base categories with Prototypical Networks, the

Models	5-Shot learning – K_{novel}=5			1-Shot learning – K_{novel}=5		
	Novel	Base	Both	Novel	Base	Both
Matching-Nets [25]	68.87 ± 0.38%	-	-	55.53 ± 0.48%	-	-
Prototypical-Nets [22]	72.67 ± 0.37%	62.10%	32.70%	54.44 ± 0.48%	52.35%	26.68%
Ours						
Cosine Classifier	72.83 ± 0.35%	70.68%	51.89%	54.55 ± 0.44%	70.68%	39.17%
Cosine Classifier & Avg. Weight Gen	74.66 ± 0.35%	70.92%	60.26%	55.33 ± 0.46%	70.45%	48.56%
Cosine Classifier & Att. Weight Gen	**74.92 ± 0.36%**	70.88%	60.50%	**58.55 ± 0.50%**	70.73%	50.50%
Ablations						
Dot Product	64.58 ± 0.38%	63.59%	31.80%	46.09 ± 0.40%	63.59%	24.76%
Dot Product & Avg. Weight Gen	60.30 ± 0.39%	62.15%	46.41%	44.31 ± 0.40%	61.99%	39.05%
Dot Product & Att. Weight Gen	67.81 ± 0.37%	62.11%	48.70%	53.88 ± 0.48%	62.28%	42.41%
Ablations						
Cosine w/ ReLU.	71.04 ± 0.36%	**72.51%**	58.16%	52.91 ± 0.45%	**72.51%**	43.17%
Cosine w/ ReLU. & Avg. Weight Gen	71.30 ± 0.38%	72.47%	59.33%	53.19 ± 0.45%	71.70%	49.53%
Cosine w/ ReLU. & Att. Weight Gen	73.03 ± 0.38%	72.26%	**61.05%**	56.09 ± 0.54%	72.34%	**51.25%**

Table 1: Average classification accuracies on the validation set of Mini-ImageNet. The Novel columns report the average 5-way and 1-shot or 5-shot classification accuracies of novel categories (with 95% confidence intervals), the Base and Both columns report the classification accuracies of base categories and of both type of categories respectively. In order to report those results we sampled 2000 tasks each with 15×5 test examples of novel categories and 15×5 test examples of base categories.

Models	Feature Extractor	5-Shot learning – K_{novel}=5			1-Shot learning – K_{novel}=5		
		Novel	Base	Both	Novel	Base	Both
Matching-Nets [25]	C64F	55.30%	-	-	43.60%	-	-
Ravi and Laroche [16]	C32F	60.20 ± 0.71%	-	-	43.40 ± 0.77%	-	-
Finn *et al.* [3]	C64F	63.10 ± 0.92%	-	-	48.70 ± 1.84%	-	-
Prototypical-Nets [22]	C64F	68.20 ± 0.66%	-	-	49.42 ± 0.78%	-	-
Mishra *et al.* [12]	ResNet	68.88 ± 0.92%	-	-	55.71 ± 0.99%	-	-
Ours	C32F	70.27 ± 0.64%	61.08%	52.45%	54.33 ± 0.81%	61.09%	43.05%
Ours	C64F	72.81 ± 0.62%	68.13%	57.72%	**56.20 ± 0.86%**	68.08%	48.09%
Ours	C128F	**73.00 ± 0.64%**	70.90%	**59.35%**	55.95 ± 0.84%	70.72%	49.08%
Ours	ResNet	70.13 ± 0.68%	80.16%	56.04%	55.45 ± 0.89%	80.24%	**51.23%**

Table 2: Average classification accuracies on the test set of Mini-ImageNet. In order to report those results we sampled 600 tasks in a similar fashion as for the validation set of Mini-ImageNet.

4.1.2 Comparison with state-of-the-art

Here we compare the proposed few-shot object recognition system with other state-of-the-art approaches on the Mini-ImageNet test set.

Explored feature extractor architectures. Because prior approaches use several different network architectures for implementing the feature extractor of the ConvNet model, we evaluate our model with each of those architectures. Specifically the architectures that we evaluated are: C32F is a 4 module ConvNet network (which was described in § 4.1.1) with 32 feature channels on each convolutional layer, C64F has 64 feature channels on each layer, and in C128F the first two layers have 64 channels and the latter two have 128 channels (exactly the same as the model that was used in § 4.1.1). With ResNet we refer to the ResNet [5] like network that was used from Mishra *et al.* [12] (for more details we refer to [12]).

In Table 2 we provide the experimental results. In all cases, our models (that include the cosine-similarity based ConvNet model and the attention-based few-shot weight

---

prototypes for the base categories are computed by averaging all the available training features vectors

Approach	Novel N'=1	2	5	10	20	All N'=1	2	5	10	20	All with prior N'=1	2	5	10	20
Prior work															
Prototypical-Nets [22] (from [26])	39.3	54.4	66.3	71.2	73.9	49.5	61.0	69.7	72.9	74.6	53.6	61.4	68.8	72.0	73.8
Matching Networks [25] (from [26])	43.6	54.0	66.0	72.5	76.9	54.4	61.0	69.0	73.7	76.5	54.5	60.7	68.2	72.6	75.6
Logistic regression (from [26])	38.4	51.1	64.8	71.6	76.6	40.8	49.9	64.2	71.9	76.9	52.9	60.4	68.6	72.9	76.3
Logistic regression w/ H [4] (from [26])	40.7	50.8	62.0	69.3	76.5	52.2	59.4	67.6	72.8	76.9	53.2	59.1	66.8	71.7	76.3
SGM w/ H [4]	-	-	-	-	-	54.3	62.1	71.3	75.8	78.1	-	-	-	-	-
Batch SGM [4]	-	-	-	-	-	49.3	60.5	71.4	75.8	78.5	-	-	-	-	-
Concurrent work															
Prototype Matching Nets w/ H [26]	45.8	**57.8**	69.0	74.3	77.4	57.6	64.7	71.9	75.2	77.5	56.4	63.3	70.6	74.0	76.2
Prototype Matching Nets [26]	43.3	55.7	68.4	74.0	77.0	55.8	63.1	71.1	75.0	77.1	54.7	62.0	70.2	73.9	75.9
Ours															
Cosine Classifier & Avg. Weight Gen	45.23	56.90	68.68	74.36	77.69	57.65	64.69	72.35	76.18	78.46	56.43	63.41	70.95	74.75	77.00
	±.25	±.16	±.09	±.06	±.06	±.15	±.10	±.06	±.04	±.04	±.15	±.10	±.06	±.04	±.03
Cosine Classifier & Att. Weight Gen	**46.02**	57.51	**69.16**	**74.83**	**78.11**	**58.16**	**65.21**	**72.72**	**76.50**	**78.74**	**56.76**	**63.80**	**72.72**	**75.02**	**77.25**
	±.25	±.15	±.09	±.06	±.05	±.15	±.09	±.06	±.04	±.03	±.15	±.10	±.06	±.04	±.04

Table 3: Top-5 accuracy on the novel categories and on all categories (with and without priors) fot the ImageNet based few-shot benchmark proposed in [4] (for more details about the evaluation metrics we refer to [26]). For each novel category we use $N' = 1, 2, 5, 10$ or 20 training examples. Methods with "w/ H" use mechanisms that hallucinate extra training examples for the novel categories. The second rows in our entries report the 95% confidence intervals.

generator) achieve better few-shot object recognition performance than prior approaches. *Moreover, it is very important to note that our approach is capable to achieve such excellent accuracy on the novel categories while at the same time it does not sacrifice the recognition performance of the base categories, which is an ability that prior methods lack.*

4.2. Few-shot benchmark of Bharath & Girshick [4]

Here we evaluate our approach on the ImageNet based few-shot benchmark proposed by Bharath and Girshick [4] using the improved evaluation metrics proposed by Wang *et al*. [26]. Briefly, this benchmark splits the ImageNet categories into 389 base categories and 611 novel categories; 193 of the base categories and 300 of the novel categories are used for cross validation and the remaining 196 base categories and 311 novel categories are used for the final evaluation (for more details we refer to [4]). We use the same categories split as they did. However, because it was not possible to use the same training images that they did for the novel categories[5], we sample ourselves N' training images per novel category and, similar to them, evaluate using the images in the validation set of ImageNet. We repeat the above experiment 100 times (sampling each time a different set of training images for the novel categories) and report in Table 3 the mean accuracies and the 95% confidence intervals for the recognition accuracy metrics proposed in [26].

Comparison to prior and concurrent work. We compare our full system (Cosine Classifier & Att. Weight Gen entry) against prior work, such as Prototypical-Nets [22], Matching Networks [25], and the work of Bharath and Girshick [4]. We also compare against the work of Wang *et*

al. [26], which is concurrent to ours. We observe that in all cases our approach achieves superior performance than prior approaches and even exceeds (in all but one cases) the Prototype Matching Net [26] based approaches that are concurrent to our work.

Feature extractor: The feature extractor of all approaches is implemented with a *ResNet-10* [5] network architecture[6] that gets as input images of 224×224 size. Also, when training the attention based few-shot classification weight generator component of our model (2nd training stage) we found helpful to apply dropout with 0.5 probability on the feature vectors generated by the feature extractor.

5. Conclusions

In our work we propose a dynamic few-shot object recognition system that is able to quickly learn novel categories without forgetting the base categories on which it was trained, a property that most prior approaches on the few-shot learning task neglect to fulfill. To achieve that goal we propose a novel attention based few-shot classification weight generator as well as a cosine-similarity based ConvNet classifier. This allows to recognize in a unified way both novel and base categories and also leads to learn feature representations with better generalization capabilities. We evaluate our framework on Mini-ImageNet and the recently introduced fews-shot benchmark of Bharath and Girshick [4] where we demonstrate that our approach is capable of both maintaining high recognition accuracy on base categories and to achieve excellent few-shot recognition accuracy on novel categories that surpasses prior state-of-the-art approaches by a significant margin.

---

[5]It was not possible to establish a correspondence between the index files that they provide and the ImageNet images

[6]Similar to what it is already explained, our model does not include the last ReLU non-linearity of the *ResNet-10* feature extractor

References

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016. 2

[2] L. Chunjie, Y. Qiang, et al. Cosine normalization: Using cosine similarity instead of dot product in neural networks. *arXiv preprint arXiv:1702.05870*, 2017. 4

[3] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017. 2, 7

[4] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. *arXiv preprint arXiv:1606.02819*, 2016. 1, 2, 3, 8

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 7, 8

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[7] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 5

[8] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015. 3

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[11] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Computer Vision–ECCV 2012*, pages 488–501. Springer, 2012. 3, 4

[12] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017. 2, 7

[13] T. Munkhdalai and H. Yu. Meta networks. *arXiv preprint arXiv:1703.00837*, 2017. 2

[14] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 4

[15] H. Qi, M. Brown, and D. G. Lowe. Learning with imprinted weights. *arXiv preprint arXiv:1712.07136*, 2017. 4

[16] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016. 2, 6, 7

[17] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 4

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Ima-genet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1

[19] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016. 2

[20] J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997. 2

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[22] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 3, 5, 6, 7, 8

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[24] S. Thrun. Lifelong learning algorithms. *Learning to learn*, 8:181–209, 1998. 2

[25] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016. 3, 4, 6, 7, 8

[26] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. *arXiv preprint arXiv:1801.05401*, 2018. 8

[27] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014. 1