

# Dynamic Few-Shot Visual Learning without Forgetting

## Paper Information

**TITLE:** Dynamic Few-Shot Visual Learning without Forgetting

**AUTHORS:** Spyros Gidaris, Nikos Komodakis

**INSTITUTION:** Universite Paris Est, Ecole des Ponts ParisTech

**PUBLICATION:** CVPR 2018

**PDF:** <https://arxiv.org/abs/1804.09458>

**CODE:** <https://github.com/gidariss/FewShotWithoutForgetting>

## Notes

### Background and motivation

The human visual system exhibits the remarkable ability to be able to effortlessly learn novel concepts from only one or a few examples and reliably recognize them later on. Mimicking the same behavior on machine learning vision systems is an interesting and very challenging research problem. Research on this subject is usually termed *few-shot learning*.

The authors argue that a good realistic few-shot learning system has two important requirements:

- *dynamic few-shot learning*: fast learning of the novel categories.
- *without forgetting*: do not sacrifice any recognition accuracy on the initial categories that the model was trained on (base categories).


### Contributions

The authors 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.

In order to achieve the goal, they introduce two technical novelties:

- *few-shot classification weight generator* based on attention. Like a typical ConvNet based recognition model which computes per category classification scores by applying a set of classification weight vectors (one per category) to the extracted feature representation of an image, the authors propose to generate classification weight vectors for recognizing novel categories. It is assumed that the reason the human visual system is so efficient when learning novel concepts from only a few examples is that it exploits its past experiences about the (visual) world. So 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.
- *cosine-similarity based ConvNet recognition model*. The authors found that it is not feasible for the ConvNet model to be able to simultaneously handle the classification weight vectors of both base and novel categories with the typical dot-product based classifier (i.e., the last linear layer of a classification neural network). So they propose to implement the classifier of a ConvNet model as a cosine similarity function between the feature representations and the classification weight vectors, which not only unifies the recognition of both novel and base categories, but also leads to feature representations that generalize better on “unseen” categories.

## Methodology

 An overview of the proposed framework

### ConvNet-based recognition model.

Training dataset of  $K_{base}$  base categories with a large set of training data per category:

$$D_{train} = \bigcup_{b=1}^{K_{base}} \{x_{b,i}\}_{i=1}^{N_b},$$

where  $N_b$  is the number of training examples of the  $b$ -th category and  $x_{b,i}$  is its  $i$ -th training example.

(a) a feature extractor  $F(.|\theta)$  extracts a  $d$ -dimensional feature vector  $z = F(x|\theta) \in \mathbb{R}^d$  from an input image  $x$ .

(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, takes the feature representation  $z$  as input and returns probability classification scores  $p = C(z|W^*) \in \mathbb{R}^{K^*}$

When training ConvNet-based recognition model in the 1st training stage, we learn  $\theta$  and  $W_{base} = \{w_k\}_{k=1}^{K_{base}}$  such that by setting  $W^* = W_{base}$  the ConvNet model will be able to recognize the base

object categories.

### Few-shot classification weight generator.

Training datasets of  $K_{novel}$  novel categories with few training examples per category, which are sampled from  $D_{train}$ :

$$D_{novel} = \bigcup_{n=1}^{K_{novel}} \{x'_{n,i}\}_{i=1}^{N'_n},$$

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.

For each novel category  $n \in [1, K_{novel}]$ , few-shot classification weight generator  $G(\cdot, \cdot, |\phi)$  takes 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  $W_{base}$  as input and generates a classification weight vector  $w'_n = G(Z'_n, W_{base}|\phi)$  for that novel category.

When training few-shot classification weight generator in the 2nd training stage via a meta-learning mechanism, we learn  $\phi$ . By setting  $W^* = W_{base} \cup W_{novel}$  on the classifier  $C(\cdot|W^*)$ , where  $W_{novel} = \{w'_n\}_{n=1}^{K_{novel}}$  are generated classification weight vectors of the novel categories, we enable the ConvNet model to recognize both base and novel categories in a unified manner.

## 1. Cosine-similarity based recognition model

Standard setting for classification neural networks:

- extract feature vector  $z$ .
- estimate the classification probability vector  $p = C(z|W^*)$ :
  - compute the raw classification score  $s_k$  of each category  $k \in [1, K^*]$ :  $s_k = z^\top w_k^*$ , where  $w_k^*$  is the  $k$ -th classification weight vector in  $W^*$ .
  - apply softmax across all the  $K^*$  classification scores:  $p_k = \text{softmax}(s_k)$ , where  $p_k$  is the  $k$ -th classification probability of  $p$ .

Our few-shot learning setting:

- $w_k \in W_{base}$  and  $w_k \in W_{novel}$
- weight values, raw classification scores computed with the dot-product operation can have totally different magnitudes depending on whether they come from the base or the novel categories. This does not allow to have a unified recognition of both type of categories.

- modify the classifier  $C(\cdot | W^*)$  and compute the raw classification scores using the cosine similarity operator ( $l_2$  normalization):  $s_k = \tau \cdot \cos(z, w_k^*) = \tau \cdot \bar{z}^\top \bar{w}_k^*$ , where  $\bar{z} = \frac{z}{\|z\|}$  and  $\bar{w}_k^* = \frac{w_k^*}{\|w_k^*\|}$  are  $l_2$ -normalized vectors and  $\tau$  is a learnable scalar value.
- 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.
- remove the ReLU non-linearity 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, and thus significantly improve the recognition performance of novel categories ( $\sim$ ).

Advantages of cosine-similarity based classifier:

- it allows to unify the recognition of both base and novel categories without significantly altering the classification pipeline for the recognition of base categories.
- it leads the feature extractor to learn features that generalize significantly better on novel categories than features learned with the dot-product based classifier:
  - the cosine similarity based ConvNet model forces the feature extractor to learn feature vectors that form compact category-wise clusters.
  - it also forces the classification weight vectors to learn to be representative feature vectors of those clusters.

## 2. Few-shot classification weight generator

The few-shot classification weight generator  $G(\cdot, \cdot | \phi)$  takes 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  $W_{base}$ , and, based on them, infers a classification weight vector  $w' = G(Z', W_{base} | \phi)$  for that novel category.

### Feature averaging based weight inference.

Infer the classification weight vector  $w'$  by averaging the feature vectors of the training examples (after they have been  $l_2$ -normalized):

\$\$

$$w_{\{avg\}} = \frac{1}{N} \sum_{i=1}^N \bar{z}_i$$

$$w' = \phi_{\{avg\}} \odot w_{\{avg\}}$$

where  $\odot$  is Hadamard product and  $\phi_{\{avg\}} \in \mathbb{R}^d$  is a learnable weight vector.

It is inspired by the fact that the feature extractor trained solely on cosine-similarity based classification of base categories can generate features that form more compact and distinctive category-specific clusters (i.e., more discriminative features).

### Attention-based weight inference.

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

$$w' = \phi_{avg} \odot w_{avg} + \phi_{att} \odot w'_{att},$$

where  $\phi_q \in \mathbb{R}^{d \times d}$  is a learnable weight matrix that transforms the feature vector  $\bar{z}'_i$  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.  $Att(., .)$  is an attention kernel implemented as a cosine similarity function with a learnable scalar parameter  $\gamma$ , followed by a softmax operation over the  $K_{base}$  base categories.  $\odot$  is the Hadamard product and  $\phi_{avg}, \phi_{att} \in \mathbb{R}^d$  are learnable weight vectors.

It is inspired by the assumption 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. And one advantage of the cosine-similarity based classifier is that, after training the classifier on base categories, the base classification weight vectors learn to be representative feature vectors of their categories. Thus, the base classification weight vectors also encode visual similarity. 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.

### 3. Training procedure

Modules: feature extractor  $F(.\mid\theta)$ , classifier  $C(.\mid W^*)$ , few-shot classification weight generator  $G(., .\mid\phi)$ .

Training dataset:  $D_{train} = \bigcup_{b=1}^{K_{base}} \{x_{b,i}\}_{i=1}^{N_b}$  of  $K_{base}$  base categories.

The training procedure consists of 2 stages and at each stage a different cross-entropy loss of the following form is minimized:

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

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

#### 1st training stage.

We only learn the ConvNet recognition model without the few-shot classification weight generator

during this stage.

Learning the parameters  $\theta$  of the feature extractor  $F(\cdot|\theta)$  and the base classification weight vectors  $W_{base} = \{w_b\}_{b=1}^{K_{base}}$  is done in exactly the same way as for any other standard recognition model. In this case,  $W^* = W_{base}$ .

## 2nd training stage.

We train the learnable parameters  $\phi$  of the few-shot classification weight generator while we continue training the base classification weight vectors  $W_{base}$  but keep the feature extractor frozen.

Unlike the standard training procedure in the 1st training stage, the authors exploit a meta-learning mechanism. The episodes consisting of  $K_{novel}$  "fake" novel categories with  $N'$  training examples (typically  $N' \leq 5$ ) each are randomly sampled from the base categories. For each episode, the feature vectors  $Z' = \{z'_i\}_{i=1}^{N'}$  of each "fake" novel category are given to the few-shot classification weight generator  $G(\cdot, \cdot|\phi)$  to infer the novel classification weight vectors which are used for recognizing the "fake" novel categories.

In this case,  $W^*$  is the union of the "fake" novel classification generated by  $G(\cdot, \cdot|\phi)$  and the classification weight vectors of the remaining base categories.

## Experimental results

result1

result2

## Inspiration

aaa

bbb