Supplementary Material: Graph convolutional networks for learning with few clean and many noisy labels

Ahmet Iscen[1], Giorgos Tolias[2], Yannis Avrithis[3], Ondřej Chum[2], and Cordelia Schmid[1]

[1] Google Research
[2] VRG, Faculty of Electrical Engineering, Czech Technical University in Prague
[3] Inria, Univ Rennes, CNRS, IRISA

A The role of base classes

The proposed method is applicable with any given feature extractor. Herein, we describe the learning of the feature extractor on a set of base classes according to a standard few-shot learning setup and benchmark [12]. Then, we describe the extended classifiers to the union of all classes, *i.e.* base classes and novel classes, which are the ones used in Section 5.

A.1 Representation learning on base classes

We are given a set $X_{\mathcal{B}} \subset \mathcal{X}$ of examples, each having a clean label in a set of *base classes* $C_{\mathcal{B}}$ with $|C_{\mathcal{B}}| = K_{\mathcal{B}}$. Base classes $C_{\mathcal{B}}$ are disjoint from C, which are also known as novel classes. These data are used to learn a feature representation, *i.e.* a feature extractor g_θ, by learning a $K_{\mathcal{B}}$-way base-class classifier for unseen data in $\mathcal{X}$. The parameters θ of the feature extractor and $W_{\mathcal{B}}$ of the classifier are jointly learned by minimizing the cross entropy loss

$$L_{\mathcal{B}}(C_{\mathcal{B}}, X_{\mathcal{B}}; \theta, W_{\mathcal{B}}) = -\sum_{c \in C_{\mathcal{B}}} \frac{1}{|X_{\mathcal{B}}^c|} \sum_{x \in X_{\mathcal{B}}^c} \log(\boldsymbol{\sigma}(s \hat{W}_{\mathcal{B}}^\top \hat{g}_\theta(x))_c). \tag{1}$$

The learned feature extractor parameters θ and the learned scale parameter s are used by our method as described Sections 4 and 5.

A.2 Classification on all classes

The classifier parameters $W_{\mathcal{B}}$ are used, combined with classifier parameters W learned as described in Section 5, for classification on *all classes* $C_{\mathcal{A}} = C \cup C_{\mathcal{B}}$.

Class prototypes. The concatenated parameter matrix $W_{\mathcal{A}} = [W_{\mathcal{B}}, W]$ is used for $K_{\mathcal{A}}$-way prediction on all (base and novel) classes by $\pi_{\theta, W_{\mathcal{A}}}$, where $K_{\mathcal{A}} = K + K_{\mathcal{B}}$. $W_{\mathcal{B}}$ is learned according to $L_{\mathcal{B}}(C_{\mathcal{B}}, X_{\mathcal{B}}; \theta, W_{\mathcal{B}})$ (1), while W is learned according to (5).

Cosine classifier learning. Prediction on all classes is made as in the previous case, but W is learned according to (6).

Deep network fine-tuning. We now assume that base class examples are accessible too and, given all examples $X_{\mathcal{A}} = X_{\mathcal{B}} \cup X_{\mathcal{E}}$, we jointly learn the parameters θ of the

Method	Top-5 accuracy on all classes				
	k=1	2	5	10	20
	ResNet-10 – Few Clean Examples				
Proto.-Nets [33]†	49.5	61.0	69.7	72.9	74.6
Logistic reg. w/ H [41]†	54.4	61.0	69.0	73.7	76.5
PMN w/ H [41]†	40.8	49.9	64.2	71.9	76.9
Class proto. [9]	57.0±0.36	64.7±0.16	72.5±0.18	75.8±0.16	77.4±0.19
Class proto. w/ Att. [9]	58.1±0.48	65.2±0.15	72.9±0.25	76.6±0.18	78.8±0.16
	ResNet-10 – Few Clean & Many Noisy Examples				
Ours - class proto. (5)	**70.3±0.05**	**72.1±0.18**	**74.1±0.12**	**75.6±0.13**	**76.9±0.09**
Ours - cosine (6)	**72.4±0.07**	**73.4±0.21**	**77.2±0.20**	**78.8±0.21**	**79.2±0.17**
Ours - fine-tune	**76.0±0.10**	**77.3±0.13**	**78.7±0.19**	**80.7±0.25**	**82.2±0.14**
	ResNet-50 – Few Clean Examples				
Proto.-Nets [33]†	61.4	71.4	78.0	80.0	81.1
PMN w/ H [41] †	65.7	73.5	80.2	82.8	84.5
	ResNet-50 – Few Clean & Many Noisy Examples				
Ours - class proto. (5)	**73.8±0.33**	**76.6±0.36**	**78.9±0.19**	**80.8±0.21**	**82.2±0.14**
Ours - cosine (6)	**78.2±0.25**	**79.6±0.23**	**80.4±0.18**	**82.4±0.19**	**84.1±0.09**
Ours - fine-tune	**81.6±0.20**	**83.2±0.16**	**84.3±0.23**	**86.2±0.17**	**87.8±0.03**

Table 1. Comparison to the state of the art on the Low-shot ImageNet benchmark. We report top-5 accuracy on all classes. We use class prototypes (5), cosine classifier learning (6) and deep network fine-tuning for classification with our GCN-based data addition method. † denotes numbers taken from the corresponding papers. All other experiments are re-implemented by us.

feature extractor and $W_{\mathcal{A}} = [W_{\mathcal{B}}, W]$ of the $K_{\mathcal{A}}$-way cosine classifier for all classes by minimizing loss function

$$L_{\mathcal{A}}(C_{\mathcal{A}}, X_{\mathcal{A}}; \theta, W_{\mathcal{A}}) = L_{\mathcal{B}}(C_{\mathcal{B}}, X_{\mathcal{B}}; \theta, W_{\mathcal{B}}) + L(C, X_{\mathcal{E}}; \theta, W). \tag{2}$$

Note that in contrast to (6), the last term of (2) optimizes parameters θ too. As mentioned earlier, such learning is typically avoided in a few-shot learning setup. In few cases, it takes the form of fine-tuning including all base class data [26], or only lasts for a few iterations when the base class data is not accessible [6].

A.3 Results on all classes

We report the accuracy over all classes in Table 1. When fine-tuning the network by (2), the learned W is used to initialize the corresponding part of $W_{\mathcal{A}}$ and we train all layers for 10 epochs with learning rate 0.01. The results indicate that our method still brings significant improvements when all classes are used.

B Results on Mini-Imagenet

We evaluate the proposed method on another popular benchmark, *i.e.* few-shot learning on Mini-ImageNet [38]. The dataset is a subset of ImageNet [32], and contains 100

Method	k=1	k=5
	Few Clean Examples	
Class proto. [9]	54.2$_{\pm 0.77}$	71.2$_{\pm 0.61}$
Class proto. w/ Att. [9]	56.2$_{\pm 0.81}$	72.9$_{\pm 0.62}$
Few Clean & Many Noisy Examples - Class proto. (5)		
β-weighting, $\beta = 1$	63.5$_{\pm 0.77}$	65.2$_{\pm 0.81}$
Label Propagation	67.0$_{\pm 0.74}$	74.8$_{\pm 0.61}$
MLP	65.9$_{\pm 0.78}$	73.9$_{\pm 0.63}$
Ours	68.2$_{\pm 0.76}$	74.7$_{\pm 0.59}$

Table 2. Comparison with baselines using noisy examples on the Mini-ImageNet dataset. We report the accuracy for 5-way k-shot experiments where $k = 1$ and $k = 5$.

different classes, split into 64 base, 16 validation and 20 test classes [27]. Each class contains 600 images that are re-sized to a resolution of 84×84. We use the ConvNet-128 model with cosine classifier, following [9]. Novel categories are classified using class prototypes (5).

Table 2 shows the accuracy on Mini-Imagenet for the 5-way k-shot classification scenario with $k = 1$ and $k = 5$. We report the average accuracy over 600 trials along with the confidence interval. Our method brings significant improvements for $k = 1$, showing its generalization across different few-shot datasets and benchmarks.