On Episodes, Prototypical Networks, and Few-shot Learning

Steinar Laenen[*] **& Luca Bertinetto**
Oxford Research Group
FiveAI
{steinar.laenen,luca.bertinetto}@five.ai

Abstract

Episodic learning is a popular practice among researchers and practitioners interested in few-shot learning. It consists of organising training in a series of learning problems, each relying on small "support" and "query" sets to mimic the few-shot circumstances encountered during evaluation. In this paper, we investigate the usefulness of episodic learning in Prototypical Networks, one of the most popular algorithms making use of this practice. Surprisingly, in our experiments we found that, for Prototypical Networks, it is detrimental to use the episodic learning strategy of separating training samples between support and query set, as it is a data-inefficient way to exploit training batches. This "non-episodic" version of Prototypical Networks, which corresponds to the classic Neighbourhood Component Analysis, reliably improves over its episodic counterpart in multiple datasets, achieving an accuracy that is competitive with the state-of-the-art, despite being extremely simple.

1 Introduction

The problem of few-shot learning (FSL) – classifying examples from previously unseen classes given only a handful of training data – has considerably grown in popularity within the machine learning community in the last few years. The reason is likely twofold. First, being able to perform well on FSL problems is important for several applications, from learning new characters (Lake et al., 2015) to drug discovery (Altae-Tran et al., 2017). Second, since the aim of researchers interested in meta-learning is to design systems that can quickly learn novel concepts by generalising from previously encountered learning tasks, FSL benchmarks are often adopted as a practical way to empirically validate meta-learning algorithms.

To the best of our knowledge, there is not a widely recognised definition of meta-learning. In a recent survey, Hospedales et al. (2020) informally describe it as *"the process of improving a learning algorithm over multiple learning episodes"*. Several popular papers in the FSL community (e.g. Vinyals et al. (2016); Ravi & Larochelle (2017); Finn et al. (2017); Snell et al. (2017)) have emphasised the importance of organising training into *episodes*, i.e. learning problems with a limited amount of training and (pseudo-)test examples that resemble the test-time scenario. This popularity has reached such a point that an "episodic" data-loader is often at the core of new FSL algorithms, a practice facilitated by frameworks such as Deleu et al. (2019) and Grefenstette et al. (2019).

Despite the considerable strides made in FSL over the past few years, several recent works (e.g. Chen et al. (2019); Wang et al. (2019); Dhillon et al. (2020); Tian et al. (2020)) showed that simple baselines can outperform established meta-learning methods by using embeddings pre-trained with standard classification losses. These results have cast a doubt in the FSL community on the usefulness

[*] steinar9@gmail.com

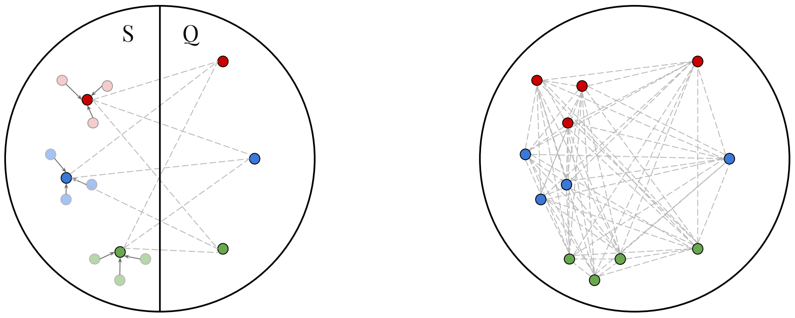

Figure 1: **Batch exploitation** during training for Prototypical Networks (Snell et al., 2017) (left) vs. Neighborhood Component Analysis (Goldberger et al., 2005) (right) on a toy few-shot problem of 3 "ways", 3 "shots" and 1 "query" per class. By dividing batches between support set S and query set Q, Prototypical Networks disregards many of the distances between labelled examples that would constitute useful training signal. Details in Sec. 3.3.

of meta-learning and its pervasive episodes. Inspired by these results, we aim at understanding the practical usefulness of episodic learning in arguably the simplest method which makes use of it: Prototypical Networks (Snell et al., 2017). We chose to analyse Prototypical Networks not only for their simplicity, but also because they often appear as important building blocks of newly-proposed methods (e.g. Oreshkin et al. (2018); Cao et al. (2020); Gidaris et al. (2019); Yoon et al. (2019)).

With a set of ablative experiments, we show that for Prototypical Networks episodic learning *a)* is detrimental for performance, *b)* is analogous to randomly discarding examples from a batch and *c)* it introduces a set of unnecessary hyper-parameters that require careful tuning. We also show how, without episodic learning, Prototypical Networks are connected to the classic *Neighbourhood Component Analysis* (NCA) (Goldberger et al., 2005; Salakhutdinov & Hinton, 2007) on deep embeddings. Without bells and whistles, our implementation of the NCA loss achieves an accuracy that is competitive with the state-of-the-art on multiple FSL benchmarks: *mini*ImageNet, CIFAR-FS and *tiered*ImageNet.

2 Related work

Pioneered by the seminal work of Utgoff (1986), Schmidhuber (1987, 1992), Bengio et al. (1992) and Thrun (1996), the general concept of meta-learning is several decades old (for a survey see Vilalta & Drissi (2002); Hospedales et al. (2020)). However, in the last few years it has experienced a surge in popularity, becoming the most used paradigm for learning from very few examples. Several methods addressing the FSL problem by learning on episodes were proposed. MANN (Santoro et al., 2016) uses a Neural Turing Machine (Graves et al., 2014) to save and access the information useful to meta-learn; Bertinetto et al. (2016) propose a deep network in which a "teacher" branch is tasked with predicting the parameters of a "student" branch; Matching Networks (Vinyals et al., 2016) and Prototypical Networks (Snell et al., 2017) are two non-parametric methods in which the contributions of different examples in the support set are weighted by either an LSTM or a softmax over the cosine distances for Matching Networks, and a simple average for Prototypical Networks; Ravi & Larochelle (2017) propose instead to use an LSTM to learn the hyper-parameters of SGD, while MAML (Finn et al., 2017) learns to fine-tune an entire deep network by backpropagating through SGD. Despite these works widely differing in nature, they all stress on the importance of organising training in a series of small learning problems (*episodes*) that are similar to those encountered during inference at test time.

In contrast with this trend, a handful of papers have recently shown that simple approaches that forego episodes and meta-learning can perform well on FSL benchmarks. These methods all have in common that they pre-train a feature extractor with the cross-entropy loss on the "meta-training classes" of the dataset. Then, at test time a classifier is adapted to the support set by weight imprinting (Qi et al., 2018; Dhillon et al., 2020), fine-tuning (Chen et al., 2019), transductive fine-tuning (Dhillon et al., 2020) or logistic regression (Tian et al., 2020). Wang et al. (2019) suggest performing test-time classification by using the label of the closest centroid to the query image.

Different from these papers, we try to shed some light on one of the possible causes behind the poor performance of episodic-based algorithms like Prototypical Networks. An analysis similar to ours in spirit is the one of Raghu et al. (2020). After showing that the efficacy of MAML in FSL is due to the adaptation of the final layer and the "reuse" of the features of previous layers, they propose a variant with the same accuracy and computational advantages. In this paper, we focus on an FSL algorithm just as popular and uncover inefficiencies that allow for a notable conceptual simplification of Prototypical Networks, which surprisingly also brings a significant boost in performance.

3 Background and method

This section is divided as follows: Sec. 3.1 introduces episodic learning and the formalism used in FSL, Sec. 3.2 reviews Prototypical Networks (often referred to as PNs from now on), Sec 3.3 describes the classic NCA loss and how exactly it relates to PNs, and Sec. 3.4 explains the three options we explored to perform FSL classification with an NCA-trained feature embedding.

3.1 Episodic learning

A common strategy to train few-shot learning algorithms is to consider a distribution $\hat{\mathcal{E}}$ over possible subsets of labels that is as close as possible to the one encountered during evaluation $\mathcal{E}$ [2] . Each episodic batch $B_E = \{S, Q\}$ is obtained by first sampling a subset of labels L from $\hat{\mathcal{E}}$, and then sampling images constituting both *support set* S and *query set* Q from the set of images with labels in L, where $S = \{(\mathbf{s}_1, y_1), \ldots, (\mathbf{s}_n, y_n)\}$, $Q = \{(\mathbf{q}_1, y_1), \ldots, (\mathbf{q}_m, y_m)\}$, and S_k and Q_k denote the sets of images with label $y = k$ in the support set and query set respectively. In a Maximum Likelihood Estimation framework, training on these episodes can be written (Vinyals et al., 2016) as the optimisation

$$\arg\max_{\theta} E_{L\sim\hat{\mathcal{E}}}\left[E_{S\sim L, Q\sim L}\left[\sum_{(q_i,y_i)\in Q} \log P_\theta\left(y_i|q_i, S\right)\right]\right]. \tag{1}$$

In most implementations this corresponds to training on a series of mini-batches in which each image belongs either to the support or the query set. Support and query sets are constructed such that they both contain all the classes of L, and a constant number of images per class. Therefore, episodes are characterised by three values: the number of classes $w = |L|$ (the "ways"), the number of examples per class in the support set $n = |S_k|$ (the "shots"), and the number of examples per class in the query set $m = |Q_k|$. Importantly, during evaluation the triplet $\{w, n, m\}$ defines the problem setup. Usually the triplet remains unchanged across methods, although the Meta-Dataset benchmark of Triantafillou et al. (2019) evaluates on episodes with a variable number of ways and shots. At training time the triplet $\{w, n, m\}$ can be seen as a set of hyper-parameters controlling the batch creation, and that (as we will see in Sec. 4.2) requires careful tuning.

3.2 Prototypical Networks (PNs)

Prototypical Networks (Snell et al., 2017) are one of the most popular and effective approaches in the few-shot learning literature, and they are at the core of many newly proposed methods (e.g. Oreshkin et al. (2018); Gidaris et al. (2019); Allen et al. (2019); Yoon et al. (2019); Cao et al. (2020)).

During *training*, episodes constituted by support set S and query set Q are sampled as described in Sec. 3.1. Then, a *prototype* for each class k is computed as the mean embedding of the samples from the support set belonging to that class: $\mathbf{c}_k = (1/|S_k|) \cdot \sum_{(\mathbf{s}_i, y_k)\in S_k} f_\theta(\mathbf{s}_i)$, where f_θ is a deep neural networks with parameters θ learned via Eq. 1.

Let $C = \{(\mathbf{c}_1, y_1), \ldots, (\mathbf{c}_k, y_k)\}$ be the set of prototypes and corresponding labels. The loss can be written as follows:

$$\mathcal{L}_{\text{proto}}(S, Q) = -\frac{1}{|Q|} \sum_{(\mathbf{q}_i, y_i)\in Q} \log\left(\frac{\exp -\|f_\theta(\mathbf{q}_i) - \mathbf{c}_{y_i}\|^2}{\sum_{k'} \exp -\|f_\theta(\mathbf{q}_i) - \mathbf{c}_{k'}\|^2}\right). \tag{2}$$

[2] Note that, in FSL, the sets of classes of training and evaluation are disjoint.

	positives	**negatives**
PNs	wmn	$w(w-1)mn$
NCA	$\binom{m+n}{2}w$	$\binom{w}{2}(m+n)^2$

Table 1: Number of gradients from positive and negative distance pairs contributing to the loss within a batch for NCA and PNs. Respectively, w, n and m represent the number of ways, shots and queries. In Appendix A.8 we show that the extra number of pairs NCA can exploit grows as $O(w^2(m^2+n^2))$.

Here, k' is an index that goes over all classes. This loss is minimised over a number of training episodes. After training, given a query image $\mathbf{x}_i$ from a new test episode, classification is conducted by simply consulting the nearest-neighboring prototype computed from the support set of the episode, i.e. $y(\mathbf{x}_i) = \arg\min_{j\in\{1,\dots,k\}} \|f_\theta(\mathbf{x}_i) - \mathbf{c}_j\|$.

3.3 Neighbourhood Component Analysis (NCA)

Eq. 2 computes the likelihood that a query image belongs to the class a certain prototype is representative of by computing the softmax over the distances to all prototypes. This formulation is similar to the *Neighbourhood Component Analysis* approach by Goldberger et al. (2005) (and expanded to the non-linear case by Salakhutdinov & Hinton (2007)), except for a few important differences which we will now discuss.

Let $i \in [1, b]$ be the indices of the images within a batch B. The NCA loss can be written as:

$$\mathcal{L}_{\text{NCA}}(X) = -\frac{1}{b}\sum_{i\in 1,\dots,b} \log\left(\frac{\sum\limits_{\substack{j\in 1,\dots,b\\ j\neq i\\ y_i=y_j}} \exp -\|\mathbf{z}_i-\mathbf{z}_j\|^2}{\sum\limits_{\substack{k\in 1,\dots,b\\ k\neq i}} \exp -\|\mathbf{z}_i-\mathbf{z}_k\|^2}\right), \tag{3}$$

where $\mathbf{z}_i = f_\theta(\mathbf{x}_i)$ is an image embedding and y_i its corresponding label. By minimising this loss, distances between embeddings from the same class will be minimised, while distances between embeddings from different classes will be maximised. This bears similarities to the (supervised) contrastive loss (Khosla et al., 2020; Chen et al., 2020a), which we discuss in Appendix A.3. For ease of discussion, we refer to the distances between pairs of embeddings from the same class as *positives*, and to the distances between pairs of embeddings from different classes as *negatives*.

Importantly, the concepts of support set and query set of Sec. 3.1 and 3.2 here do not apply. More simply, the images (and respective labels) constituting the batch $B = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_b, y_b)\}$ are sampled i.i.d. from the training dataset, as normally happens in standard supervised learning. Considering that in PNs there is no parameter adaptation happening at the level of each episodic batch B_E, S and Q *do not* have a functional role in the algorithm, and their defining values $\{w, m, n\}$ can be interpreted as hyper-parameters controlling the data-loader during training. More specifically, PNs differ from NCA in three key aspects:

1. PNs rely on the creation of prototypes, while NCA does not.
2. Due to the nature of episodic learning, PNs only consider pairwise distances between the query and the support set; the NCA instead uses *all* the distances within a batch and treats each example in the same way.
3. Because of how L, S and Q are sampled in episodic learning, some images will be inevitably sampled more frequently than others, and some will likely never be seen during training (this corresponds to sampling "with replacement"). NCA instead visits every image of the dataset once and only once within each epoch (sampling "without replacement").

Notes on data-efficiency in batch exploitation. To expand on point 2 above, Fig. 1 illustrates the difference in batch exploitation. For PNs (*left*) and a training episode of w=3 ways, n=3 shots and m=1 queries, the total number distances contributing to the loss consists of $wmn = 9$ positives and $w(w-1)mn = 18$ negatives [3]. If we consider the *same* training batch in a non-

[3]The number of distances computed in the batch is actually even smaller for PNs: wm=3 positives and $w(w-1)m$=6 negatives. However, since gradients are propagated through prototypes, the distances to the individual support points can be considered as contributing to the loss. This is what we do throughout the paper.

episodic way, when computing the NCA loss (Fig. 1 *right*) we have $\binom{m+n}{2}w = 18$ positives and $\binom{w}{2}(m+n)^2 = 48$ negatives (we summarise these equations in Table 1). This difference in batch exploitation is significant; in Appendix A.8 we show that it grows exponentially as $O(w^2(m^2+n^2))$.

To investigate the effect of the three key differences between PNS and NCA illustrated in this section, in Sec. 4 we conduct a wide range of experiments.

3.4 Few-shot classification during evaluation

Once f_θ has been trained, there are many possible ways to perform few-shot classification during evaluation. In this paper we consider three simple approaches that are particularly aligned for embeddings learned via metric-based losses such as Eq. 2 or Eq. 3.

k-NN. To classify an image $\mathbf{q}_i \in Q$, we first compute the Euclidean distance to each support point $\mathbf{s}_j \in S$: $d_{ij} = \|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_j))\|^2$. Then, we simply assign $y(\mathbf{q}_i)$ to be majority label of the k nearest neighbours. A downside here is that k is a hyper-parameter that has to be chosen, although a reasonable choice in the FSL setup is to set it equal to the number of "shots" n.

1-NN with Class Centroids. Similar to k-NN, we can perform classification by inheriting the label of the closest class centroid, i.e. $y(\mathbf{q}_i) = \arg\min_{j\in\{1,\dots,k\}} \|f_\theta(\mathbf{x}_i) - \mathbf{c}_j\|$. This is the approach used at test-time by Snell et al. (2017) and Wang et al. (2019).

Soft Assignments. This is what the original NCA paper (Goldberger et al., 2005) used for evaluation. To classify an image $\mathbf{q}_i \in Q$, we compute the values $p_{ij} = \exp(-\|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_j))\|^2)/\sum_{\mathbf{s}_k\in S}\exp(-\|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_k)\|^2)$ for all $\mathbf{s}_j \in S$, which is the probability that image i is sampled from image j. We then compute the likelihood for each class k: $\sum_{s_j\in S_k} p_{ij}$, and choose the class with the highest likelihood $y(\mathbf{q}_i) = \arg\max_k \sum_{s_j\in S_k} p_{ij}$. This approach is the only one closely aligned with the training procedure, and has a direct probabilistic interpretation.

We also experimented with using the NCA loss on the support set to perform adaptation at test time, which we discuss in Appendix A.1.

4 Experiments

In the following, Sec. 4.1 describes our experimental setup. Sec. 4.2 shows the effect of the hyper-parameters controlling the creation of episodes in PNS. In Sec. 4.3 we perform a set of ablation studies to better illustrate the relationship between PNS and the NCA. Finally, in Sec. 4.4 we compare our version of the NCA to several recent methods on three FSL benchmarks.

4.1 Experimental setup

We conduct our experiments on *mini*ImageNet (Vinyals et al., 2016), CIFAR-FS (Bertinetto et al., 2019) and *tiered*ImageNet (Ren et al., 2018), using the ResNet12 variant first adopted by Lee et al. (2019) as embedding function f_θ. A detailed description of benchmarks, architecture and implementation details is deferred to Appendix A.4, while below we discuss the most important choices of the experimental setup.

Like Wang et al. (2019), for all our experiments (including those with Prototypical Networks) we centre and normalise the feature embeddings before performing classification, as it is considerably beneficial for performance. After training, we compute the mean feature vectors of all the images in the training set: $\bar{\mathbf{x}} = \frac{1}{|\mathcal{D}^{\text{train}}|}\sum_{\mathbf{x}\in\mathcal{D}^{train}} \mathbf{x}$. Then, all feature vectors in the test set are updated as $\mathbf{x}_i \leftarrow \mathbf{x}_i - \bar{\mathbf{x}}$, and normalised by $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$.

We compared the inference methods discussed in Sec. 3.4 on *mini*ImageNet and CIFAR-FS. Results can be found in Appendix A.1 in Table 2. We chose to use 1-NN with class centroids in all our experiments, as it performs significantly better than k-NN or Soft Assignment. This might sound surprising, as the Soft Assignment approach closely reflects the training protocol and outputs class probabilities. We speculate its inferior performance could be caused by poor model calibration (Guo et al., 2017): since the classes between training and evaluation are disjoint, the model is unlikely to

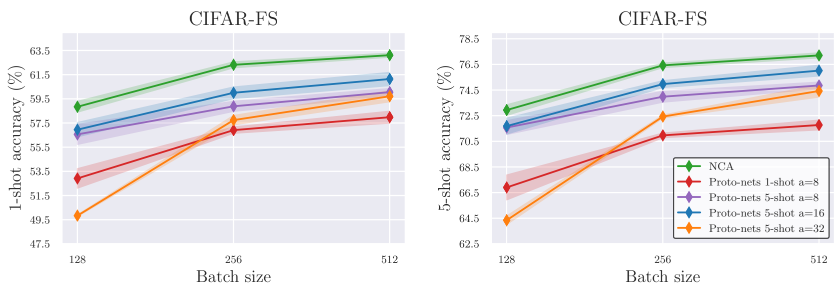

Figure 2: 1-shot (left) and 5-shot accuracies (right) on the val. set of CIFAR-FS. Models trained using NCA, or Proto-nets with different configurations: 1-shot with a=8 and 5-shot with a=8, 16 or 32. Values correspond to the mean accuracy of five models trained with different random seeds. Please see Sec. 4.2 for details.

produce calibrated probabilities. As such, within the softmax, outliers behaving as false positives can happen to highly influence the final decision, and those behaving as false negatives can end up being almost completely ignored (their contribution is squashed toward zero). With the nearest centroid classification approach outliers are still clearly an issue, but their effect can be less dramatic.

As standard, performance is assessed on episodes of 5-way, 15-query and either 1- or 5-shot. Each model is evaluated on 10,000 episodes sampled from the test set (or the validation set, in some experiments). To further reduce the variance, we trained each model five times with five different random seeds, for a total of 50,000 episodes per configuration, from which error bars are computed.

4.2 Considerations on episodes and data efficiency

Despite Prototypical Networks being one of the simplest FSL methods, the creation of episodes requires the use of several hyper-parameters ($\{w, m, n\}$, Sec. 3.1) which can significantly affect performance. Snell et al. (2017) state that the number of shots n between training and testing should match and that one should use a higher number of ways w during training time. In their experiments, they train 1-shot models with $w = 30$, $n = 1$, $m = 15$ and 5-shot models with $w = 20$, $n = 5$, $m = 15$. This makes the corresponding batch sizes of these episodes 480 and 400, respectively. Since, as seen in Sec. 3.3, the number of positives and negatives grows rapidly for both PNs and NCA (although at a different rate), this makes a fair comparison between models trained on different types of episodes difficult.

We investigate the effect of changing these hyper-parameters in a systematic manner. To compare configurations fairly across episode/batch sizes, we define each configuration by its number of shots n, the batch size b and the total number of images per class a (which accounts for the sum between support and query set, $a = n + m$). For example, if we train a 5-shot model with $a = 8$ and $b = 256$, it means that its corresponding training episodes will have $n = 5$, $q = 8 - 5 = 3$, and $w = 256/8 = 32$. Using this notation, we train configurations of PNs covering several combinations of these hyper-parameters so that the resulting batch size corresponds to an episode is 128, 256 or 512. Then, we train three configurations of NCA, where the only hyper-parameter is the batch size.

Results for CIFAR-FS can be found in Fig. 2, where we report results for NCA and PNs with $a = 8$, 16 or 32. Results for *mini*ImageNet observe the same trend and are deferred to Appendix A.6. Note that the results of 1-shot with $a = 16$ and $a = 32$ are not reported, as they fare significantly worse. Several things can be noticed. First, NCA performs better than *all* PNs configurations, no matter the batch size. Second, PNs is very sensitive to different hyper-parameter configurations. For instance, with batches of size 128, PNs trained with episodes of 5-shot and $a = 32$ performs significantly worse than a PNs trained with episodes of 5-shot and $a = 16$. Finally, we can also notice that, contrary to what has been previously reported (Snell et al., 2017; Cao et al., 2020), the 5-shot model with the best configuration is always strictly better than any 1-shot configuration. We speculate that this is probably due to the fact that we compare configurations keeping the batch size constant.

Episodic batches vs random sub-sampling of standard batches. Despite the NCA outperforming all PNs configurations (Fig. 2), one might posit that by using more distances within a batch, NCA is more computationally costly and thus the episodic strategy of PNs can sometimes be advantageous (e.g. real-time applications with large batches). To investigate this, we perform an experiment where we train NCA models by randomly sampling a fraction of the total number of distances

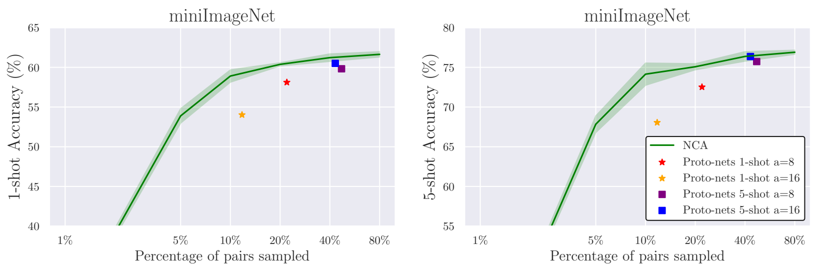

Figure 3: 1-shot and 5-shot accuracies on the *mini*ImageNet test set for models trained with a batch size of 256 while only sampling a % of the total number of available pairs. Reported values correspond to the mean accuracy of five models trained with different random seeds. Individual points contain models trained using PNs which are plotted on the x-axis based on the relative percentage of distance pairs that are used in their computation compared to NCA on the same batch size. Please see Sec. 4.2 for details.

used in the loss. Then, for comparison, we include different PNs models after having computed to which percentage of discarded pairs (in a normal batch) their episodic batch corresponds to.

Results can be found in Fig. 3. As expected, we can see how sub-sampling a fraction of the total available number of pairs within a batch negatively affects performance. To answer to possible concerns on computational efficiency, we can notice that the PNs points lie very close to the under-sampling version of the NCA, which signals that the episodic strategy of PNs is roughly equivalent to train with the NCA and only exploiting a fraction of the distances available in a batch.

We see as we move right on the curve in Fig. 3, that the PN performance increases as well. In Appendix A.9 we look more carefully whether the difference in performance between the different episodic batch setups of Fig. 2 can be explained by the differences in the number of distance pairs used in the batch configurations. We indeed find that generally speaking the higher the number of pairs the better, however, one should also consider the positive/negative balance and the number of classes present within a batch.

4.3 Ablation experiments

To better analyse why NCA performs better than PNs, in this section we consider the three key differences discussed earlier by performing a series of ablations on models trained on batches of size 256. Results are summarised in Fig. 4. We refer the reader to Appendix A.1 to obtain detailed steps describing how these ablations affect Eq. 2 and Eq. 3, while Appendix A.7 contains the same analysis also for batches of size 128 and 512.

First, we compare two variants of NCA: one in which the sampling of the training batches happens sequentially and without replacement, as it is standard in supervised learning, and one where the batches are sampled with replacement. Interestingly, this modification (row 1 and 2 of Fig. 4) has a negligible effect across the two datasets and four splits considered, meaning that the replacement sampling introduced by episodic learning will not interfere with the other ablations.

We then perform a series of ablations on episodic batches, i.e. sampled with the method described in Sec. 3.1. For each ablation, we perform experiments on PNs trained with 1-shot and 5-shot, both with $a = 8$. This means that both the 1-shot and 5-shot models have 32 classes and 8 images per class, allowing a fair comparison. The batch size is 256 for the NCA too. We first train standard PNs models. Next, we train a PNs model where "prototypes" are not computed (point 1 of Sec. 3.3), meaning that distances are considered between individual points, but a separation between query and support set remains. Then, we perform an ablation where we ignore the separation between support and query set (point 2 of Sec. 3.3), and compute the NCA on the union of the support and query set, while still computing prototypes for the points that would belong to the support set. Last, we perform an ablation where we consider all the previous points together: we sample with replacement, we ignore the separation between support and query set and we do not compute prototypes. This amounts to the NCA loss, except that it is computed on batches with a fixed number of classes and a fixed number of images per class. Notice that in Fig. 4 there is only one row dedicated to 1-shot models. This is because we cannot generate prototypes from 1-shot models, so we cannot have a "no proto" ablation.

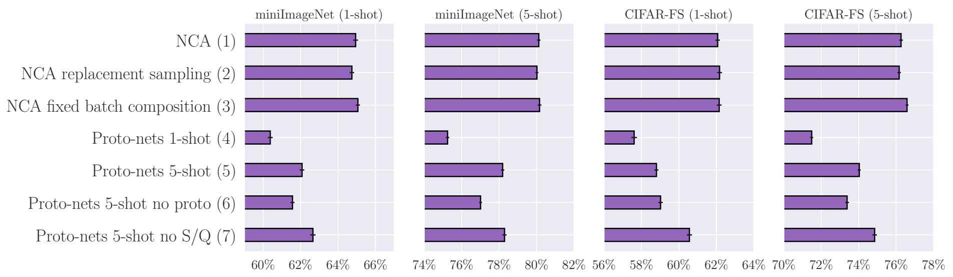

Figure 4: Ablation experiments on NCA and Prototypical Networks, both on batches or episodes of size 256 on the validation set of *mini*ImageNet and CIFAR-FS. Please refer to Sec. 4.3 for details.

Furthermore, for 1-shot models the "no S/Q" ablation is equivalent to the NCA with a fixed batch composition. From Fig. 4, we can see that disabling prototypes (row 6) negatively affects the performance of 5-shot (row 5), albeit slightly. On the other hand, enabling the computation between all pairs increases the performance (last row), Importantly, enabling all the ablations (row 3) completely recovers the performance lost by PNs.

The fact that each single ablation does not have much influence on the performance, but their combination does, could be explained by the number of distance pairs exploited by the individual ablations. Using the formulas described at the end of Section 3.3, we compute the number of positives and negatives used for each ablation. For row 6 there are 480 positives, and 14,880 negatives. For row 7 there are 576 positives and 20,170 negatives. In both cases, the number is significantly lower than the corresponding NCA, which gets 896 positives and 31,744 negatives, and this could explain the jump in performance from row 6/7 to 3. Moreover, from row 5/6 to 7 we see a slight increase in performance, which can also be explained by the (slightly) larger number of distance pairs.

These experiments nonetheless highlight that the separation of roles between the images belonging to support and query set, which is typical of episodic learning (Vinyals et al., 2016), is detrimental for the performance of Prototypical Networks. Instead, using the NCA loss on standard mini-batches allows full exploitation of the training data and significantly improves performance. Moreover, the NCA has the advantage of simplifying the overall training procedure, as the hyper-parameters for the creation of episodes $\{w, n, m\}$ no longer need to be considered.

Additionally, in Appendix A.7 we repeat the same analysis done here to the simplest variant of Matching Networks (Vinyals et al., 2016) and obtain the same conclusion; also in this case, the separation of roles between support and query samples severely affects performance.

4.4 Comparison with the state-of-the-art

We benchmark our models on *mini*ImageNet, CIFAR-FS, and *tiered*ImageNet, and show that NCA fairs surprisingly well against methods that use meta-learning and also against recent high-performing baselines which pre-train with the cross-entropy loss. Detailed results in Appendix A.10.

5 Conclusion

Towards the aim of understanding the reasons behind the poor competitiveness of meta-learning methods with respect to simple baselines, in this paper we start by investigating the role of episodes in the popular Prototypical Networks. We found that their performance is highly sensitive to the set of hyper-parameters used to sample the episodes. By replacing the Prototypical Networks' loss with the classic Neighbourhood Component Analysis, we are able to ignore these hyper-parameters while significantly improving the few-shot classification accuracy. With a series of experiments, we found out that the performance discrepancy mostly arises from the separation between support and query set within each episode, and that Prototypical Networks' episodic strategy is almost empirically equivalent to randomly discarding a large fraction of distances within a standard mini-batch. Finally, we show that our variant of the NCA achieves an accuracy on multiple popular FSL benchmarks that is comparable or superior with state-of-the-art methods of similar complexity, making it a simple and appealing baseline for future work.

References

Kelsey R Allen, Evan Shelhamer, Hanul Shin, and Joshua B Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, 2019.

Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 2017.

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

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 2016.

Luca Bertinetto, João F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

Tianshi Cao, Marc Law, and Sanja Fidler. A theoretical analysis of the number of shots in few-shot learning. In *International Conference on Learning Representations*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020a.

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

Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020b.

Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A meta-learning library for pytorch. *arXiv preprint arXiv:1909.06576*, 2019.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *International Conference on Learning Representations*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, 2018.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, 2018.

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2005.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, 2018.

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

Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Yonghong Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *IEEE International Conference on Computer Vision*, 2019.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *IEEE International Conference on Computer Vision*, 2019.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, 2007.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 1992.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, 1996.

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

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2019.

Paul E Utgoff. Shift of bias for inductive concept learning. *Machine learning: An artificial intelligence approach*, 1986.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, 2019.

Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. In *IEEE International Conference on Computer Vision*, 2019.