On Episodes, Prototypical Networks, and Few-Shot Learning

Steinar Laenen [*]
School of Informatics
University of Edinburgh
V.S.E.Laenen@sms.ed.ac.uk

Luca Bertinetto
FiveAI
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 (or *episodes*), each divided into a small training and validation subset to mimic the circumstances encountered during evaluation. But is this always necessary? In this paper, we investigate the usefulness of episodic learning in methods which use nonparametric approaches, such as nearest neighbours, at the level of the episode. For these methods, we not only show how the constraints imposed by episodic learning are not necessary, but that they in fact lead to a data-inefficient way of exploiting training batches. We conduct a wide range of ablative experiments with Matching and Prototypical Networks, two of the most popular methods that use nonparametric approaches at the level of the episode. Their "non-episodic" counterparts are considerably simpler, have less hyperparameters, and improve their performance in multiple few-shot classification datasets.

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 symbols [27] to drug discovery [2]. 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. [25] informally describe it as *"the process of improving a learning algorithm over multiple learning episodes"*. In practical terms, following the compelling rationale that *"test and train conditions should match"* [53, 14], several seminal meta-learning papers (e.g. [53, 37, 15]) have emphasised the importance of organising training into *episodes*, i.e. learning problems with a limited amount of "training" (the *support* set) and "test" examples (the *query* set) to mimic the test-time scenario presented by FSL benchmarks.

However, several recent works (e.g. [10, 54, 12, 49]) showed that simple baselines can outperform established FSL meta-learning methods by using embeddings pre-trained with the standard cross-entropy loss, thus casting a doubt on the importance of episodes in FSL. Inspired by these results, we aim at understanding the practical usefulness of episodic learning in popular FSL methods relying on metric-based nonparametric classifiers such as Matching and Prototypical Networks [53, 45]. We chose this family of methods because they do not perform any adaptation at test time. This allows us

[*] Work done while research intern at FiveAI

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

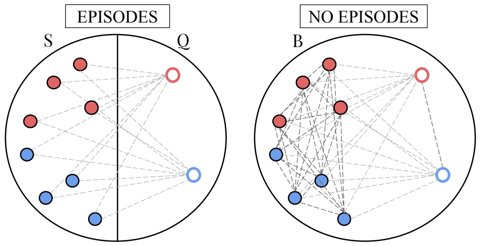

Figure 1: Difference in *batch exploitation* for metric-based methods between adopting or not adopting the concept of *episodes* during training, on an illustrative few-shot learning problem with 2 *ways* (classes), and 4 *shots* (examples) and 1 *query* per class.

	POSITIVES	NEGATIVES
NO EPISODES	$\binom{m+n}{2}w$	$\binom{w}{2}(m+n)^2$
EPISODES	wmn	$w(w-1)mn$
PAIRS LOST	$\frac{w}{2}(m^2+n^2-m-n)$	$\frac{w}{2}(w-1)(m^2+n^2)$

Table 1: The extra number of gradients that, on the *same batch*, a non-episodic method can exploit with respect to its episodic counterpart grows quadratically as $O(w^2(m^2+n^2))$, where w is the number of ways, and n and m are the number of shots and queries per class.

to test the efficacy of episodic training without having to significantly change the baseline algorithms, which could potentially introduce confounding factors.

In this work we perform a case study focussed on Matching Networks [53] and Prototypical Networks [45], and we show that within this family of methods episodic learning *a)* is detrimental for performance, *b)* is analogous to randomly discarding examples from a batch and *c)* introduces a set of superfluous hyperparameters that require careful tuning. Without episodic learning, these methods are closely related to the classic *Neighbourhood Component Analysis* (NCA) [21, 40] on deep embeddings and achieve, without bells and whistles, an accuracy that is competitive with recent methods on multiple FSL benchmarks: *mini*ImageNet, CIFAR-FS and *tiered*ImageNet.

PyTorch code is available at `https://github.com/fiveai/on-episodes-fsl`.

2 Background and method

This section is divided as follows: Sec. 2.1 introduces episodic learning and illustrates a data efficiency issue encountered with nonparametric few-shot learners based on episodes; Sec. 2.2 introduces the losses from Snell et al. [45], Vinyals et al. [53] and Goldberger et al. [21] which we use throughout our experiments; and Sec. 2.3 explains the three options we explored to perform FSL classification with previously-trained feature embeddings.

2.1 Episodic learning

A common strategy to train FSL methods 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] [53]. 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.

For most methods, this corresponds to training on a series of mini-batches in which each image belongs to *either* the support set *or* the query set. Support and query sets are constructed such that they both contain all the classes of L, and a fixed number of images per class. Therefore, episodes are defined by three variables: 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 $m = |Q_k|$. During evaluation, the set $\{w, n, m\}$ defines the problem setup. Instead, at training time $\{w, n, m\}$ can be seen as a set of hyperparameters controlling the batch creation, and that (as we will see in Sec. 3.2) requires careful tuning.

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

In a Maximum Likelihood Estimation framework, training on these episodes can be written as

$$\arg\max_{\theta} \mathop{\mathbb{E}}_{L\sim\hat{\mathcal{E}}} \mathop{\mathbb{E}}_{\substack{S\sim L \\ Q\sim L}} \left(\sum_{(q_i,y_i)\in Q} \log P_\theta \left(y_i|q_i, S, \rho\right) \right). \tag{1}$$

For the sake of brevity, with a slight abuse of notation we omit the function f_θ (e.g. a deep neural network) which is used to obtain a representation for the images in S and Q, and whose parameters θ are optimised during the training process. Note that the optimisation of Eq. 1 depends on an *optional* set of parameters ρ. This is obtained by an "inner" optimisation procedure, whose scope is limited to the current episode [25]. The idea is that the "outer" optimisation loop, by attending to a distribution of episodes, will appropriately shift the *inductive bias* of the algorithm located in the inner loop, thus learning how to learn [52]. In recent years, many interesting proposals have been made about what form ρ should have, and how it should be computed. For instance, in MAML [15] ρ takes the form of an update of the global parameters θ, while Ravi and Larochelle [37] learn to optimise by considering ρ as set of the hyper-parameters of the optimiser's update rule.

Other methods, such as Matching and Prototypical Networks [53, 45], avoid learning a separate set of parameters ρ altogether, and utilise a nonparametric learner (such as nearest neighbour classifiers) at the inner level. We chose to focus our case study on these methods not only because they have been seminal for the community, but also for ease of experimental design. Having $\rho = \varnothing$ considerably reduces the design complexity of the algorithm, thus allowing precise ablations to understand the efficacy of episodic learning without considerably changing the nature of the original algorithms.

Considerations on data efficiency. The constraints imposed by episodic learning on the role each image has in a training batch has subtle but important implications, illustrated in Fig. 1 by highlighting the number of distances contributing to the loss. By dividing batches between support and query set (S and Q) during training, *episodes* have the side effect of disregarding many of the distances between labelled examples that would constitute useful training signal for nonparametric FSL methods. More specifically, for metric-based nonparametric methods, the number of training distances that are omitted in a batch because of the episodic strategy grows quadratically as $O(w^2(m^2 + n^2))$ (derivation shown in Appendix A). Table 1 breaks down this difference in terms of gradients from positives and negatives distance pairs (which we simply refer to as *positives* and *negatives* throughout the rest of the paper). In a typical training batch with $w = 20$, $m = 15$ and $n = 5$ [45], ignoring the episodic constraints increases the number of both positives and negatives by more than 150%.

In the remainder of this paper, we conduct a case study to illustrate how this issue affects two of the most popular FSL algorithms relying on nonparametric approaches at the inner level: Prototypical Networks [45] and Matching Networks [53].

2.2 Loss functions

Prototypical Networks (PNs) [45] are one of the most popular and effective approaches in the few-shot learning literature. They are at the core of several recently proposed FSL methods (e.g. [32, 20, 1, 55, 7]), and they are used in a number of applied machine learning works (e.g. EEG scan analysis for autism [41] and glaucoma grading [18]).

During *training*, episodes consisting of a support set S and a query set Q are sampled as described in Sec. 2.1. 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 network 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:

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

where k' is an index that goes over all classes.

Matching Networks (MNs) [53] are closely related to PNs in the multi-shot case and equivalent in the 1-shot case. Rather than aggregating the embeddings of the same class into prototypes, this loss directly computes a softmax over individual embeddings of the support set, as:

$$\mathcal{L}_{\text{MNs}} = \frac{-1}{|Q|} \sum_{\substack{(\mathbf{q}_i, y) \\ \in Q}} \log \left(\frac{\sum_{\substack{\mathbf{s}_j \\ \in S_y}} \exp - \|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_j)\|^2}{\sum_{\substack{\mathbf{s}_k \\ \in S}} \exp - \|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_k)\|^2} \right).$$

In their work, Vinyals et al. [53] use the cosine rather than the Euclidean distance. However, (as [45]) we observed that the Euclidean distance is a better choice for FSL problems, and thus we use it in all the losses of our experiments. Note that Vinyals et al. [53] also suggest a variant to $\mathcal{L}_{\text{MNs}}$ (MNs with "Full Context Embeddings"), where an LSTM (with an extra set of parameters) is used to condition the way the inputs are embedded in the current support set. In our experiments, we did not consider this variant as it falls in the category of *adaptive* episodic learning approaches ($\rho \neq \varnothing$, see Sec. 2.1).

Neighbourhood Component Analysis (NCA). $\mathcal{L}_{\text{MNs}}$ and $\mathcal{L}_{\text{PNs}}$ sum over the likelihoods that a query image belongs to the same class of a certain sample (or prototype) from the support set by computing the softmax over the distances between the query and the support samples (or prototypes). This is closely related to the *Neighbourhood Component Analysis* approach by Goldberger et al. [21] (and expanded to the non-linear case by Salakhutdinov et al. [40] and Frosst et al. [16]), except for a few important differences which we discuss at the end of this section.

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

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

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. Importantly, note how the concepts of support set and query set here do not exist. More simply, the images (and respective labels) constituting the batch $B = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_b, y_b)\}$ are sampled uniformly.

Given the similarity between these three losses, and considering that PNs and MNs do not perform episode-specific parameter adaptation, $\{w, m, n\}$ can be simply interpreted as the set of hyper-parameters controlling the sampling of mini-batches during training. More specifically, PNs, MNs and NCA differ in three aspects:

I. First and foremost, due to the nature of episodic learning, PNs and MNs only consider pairwise distances between the query and the support set (Fig. 1 left); NCA instead uses *all* the distances within a batch and treats each example in the same way (Fig. 1 right).
II. Only PNs rely on the creation of prototypes.
III. Because of how L, S and Q are sampled in episodic learning (Eq. 1), for PNs and MNs some images might be sampled more frequently than others (sampling "with replacement"). NCA instead visits every image of the dataset once for each epoch (sampling "without replacement").

To investigate the effects of these three differences, in Sec. 3 we conduct a wide range of experiments.

2.3 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 intuitive for embeddings learned via metric-based losses like the ones described in Sec. 2.2. Note that, in the 1-shot case, all the evaluation methods considered coincide.

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.

Nearest centroid. Similar to k-NN, we can perform classification by inheriting the label of the closest class centroid, i.e. $y(\mathbf{q}_i) = \arg\min_{j \in \{1,\ldots,k\}} \|f_\theta(\mathbf{x}_i) - \mathbf{c}_j\|$. This is the approach used by Prototypical Networks [45], SimpleShot [54], and both baselines of Chen et al. [11].

Soft assignments. To classify an image $\mathbf{q}_i \in Q$, we compute the values

$$p_{ij} = \frac{\exp(-\|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_j))\|^2)}{\sum_{\mathbf{s}_k \in S} \exp(-\|f_\theta(\mathbf{q}_i) - f_\theta(\mathbf{s}_k)\|^2)}$$

for all $\mathbf{s}_j \in S$, which is the probability that i inherits its class from 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 is the approach for classification adopted by the original NCA paper [21] and Matching Networks [53].

We experiment with all three alternatives and observe that the *nearest centroid* approach is the most effective (details available in Appendix D). For this reason, unless differently specified, we use it as default in our experiments.

3 Experiments

In the following, Sec. 3.1 describes our experimental setup; Sec. 3.2 shows the important effect of the hyperparameters controlling the creation of episodes; in Sec. 3.3 we compare the episodic strategy to randomly discarding pairwise distances within a batch; in Sec. 3.4 we perform a set of ablations to better illustrate the relationship between PNs, MNs and NCA; finally, in Sec. 3.5 we compare our version of the NCA to several recent methods.

3.1 Experimental setup

We conduct our experiments on *mini*ImageNet [53], CIFAR-FS [5] and *tiered*ImageNet [39], using the popular ResNet-12 variant first adopted by Lee et al. [28] as embedding function f_θ [3] . A detailed description of benchmarks, architecture and choice of hyperparameters is deferred to Appendix F, while below we discuss the most important choices of the experimental setup.

Like Wang et al. [54], for all our experiments (including those with Prototypical and Matching 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 $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$.

As standard [25], performance is assessed on episodes of 5-way, 15-query and 1- or 5-shot. Each model is evaluated on 10,000 episodes sampled from the validation set during training, or from the test set during testing. To further reduce the variance, we trained each model *three times* with three different random seeds, for a total of 30,000 episodes per configuration, from which 95% confidence intervals are computed.

3.2 Batch size and episodes

Despite Prototypical and Matching Networks being among the simplest FSL methods, the creation of episodes requires the use of several hyperparameters ($\{w, m, n\}$, Sec. 2.1) which can significantly affect performance. Snell et al. [45] 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. In their experiments, they train 1-shot models with $w = 30$, $n = 1$, $m = 15$ and 5-shot models with $w = 20$, $n = 5$, $m = 15$, with batch sizes of 480 and 400, respectively. Since the corresponding batch sizes of these configurations differ, making direct comparisons between them is difficult.

[3] Note that, since we do not use a final linear layer for classification, our backbone is in fact a ResNet-11.

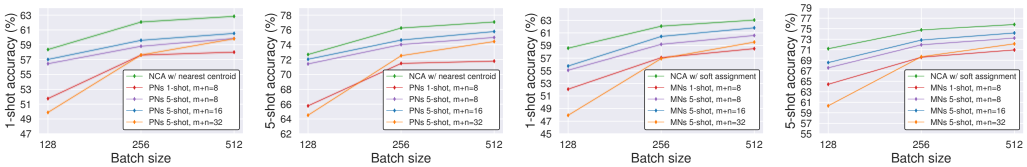

Figure 2: 1-shot and 5-shot accuracies on CIFAR-FS (val. set) for Prototypical and Matching Networks models trained with different episodic configurations: 1-shot with $m + n$=8 and 5-shot with $m + n$=8, 16 or 32. NCA models are trained on batches of size 128, 256 and 512 to match the size of the episodes. Reported values correspond to the mean accuracy of three models trained with different random seeds and shaded areas represent 95% confidence intervals. See Sec. 3.2 for details.

Instead, to directly compare configurations across batch sizes, we define an episode by its number of shots n, the batch size b and the total number of images per class $m + n$ (the sum of elements across support and query set). For example, if we train a 5-shot model with $m + n = 8$ and $b = 256$, its corresponding training episodes will have $n = 5$, $m = 8 - n = 3$, and $w = 256/(m + n) = 32$. Using this notation, we train configurations of PNs and MNs covering several combinations of these hyperparameters, so that the resulting batch size corresponding to an episode is 128, 256 or 512. Then, we train three configurations of the NCA, where the sole hyperparameter is the batch size b.

Results for CIFAR-FS can be found in Fig. 2, where we report results for NCA, PNs and MNs with $m + n = 8$, 16 or 32. Results for *mini*ImageNet observe the same trend and are deferred to Appendix H. For consistency in our comparisons, we evaluate performance using a *nearest centroid* classifier when comparing against PNs, and *soft assignments* when comparing against MNs (see Sec. 2.3). Note that PNs and MNs results for 1-shot with $m + n = 16$ and $m + n = 32$ are not reported, as they fare significantly worse. The 1-shot $m + n = 16$ is 4% worse in the best case compared to the lowest lines in Fig. 2, and the $m + n = 32$ is 10% worse in the best case. This is likely because these two setups exploit the fewest number of pairs among all the setups, which leads to the least training signal being available. In Appendix E we discuss whether the difference in performance between the different episodic batch setups of Fig. 2 can be solely 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.

Several things can be observed from Fig. 2. First, NCA-trained embeddings perform better than *all* configurations, no matter the batch size. Second, PNs and MNs are very sensitive to different hyperparameter configurations. For instance, with batches of size 128, PNs trained with episodes of 5-shot and m+n=32 perform worse than a PNs trained with 5-shot episodes and m+n=16. Note that, as we will show in Table 2, the best episodic configurations for PNs and MNs found with this hyperparameter search is superior to the setting used in the original papers.

3.3 Episodic batches vs. random sub-sampling

Despite the inferior performance with respect to the NCA, one might posit that, by training on episodes, PNs and MNs can somehow make better use of a smaller number of distances within a batch. This could be useful, for instance, in situations where it is important to train with very large batches. Given the increased conceptual complexity and the extra hyperparameters, the efficacy of episodic learning (in cases where a smaller number of distances should be considered) should be validated against the much simpler approach of random subsampling. We perform an experiment where we train NCA models by randomly discarding a fraction of the total number of distances used in the loss. Then, for comparison, we include different PNs and MNs models, after having computed to which percentage of discarded pairs (in a normal batch) their episodic batches correspond to.

Results can be found in Fig. 3. As expected, we can see how subsampling a fraction of the total available number of pairs within a batch negatively affects performance. More importantly, we can notice that the points representing PNs and MNs models lie very close to the under-sampling version of the NCA. This suggests that the episodic strategy is roughly equivalent, empirically, to only exploiting a fraction of the distances available in a batch. Note how, moving along the x-axis of Fig. 3, variants of PNs and MNs exploiting a higher number of distances perform better.

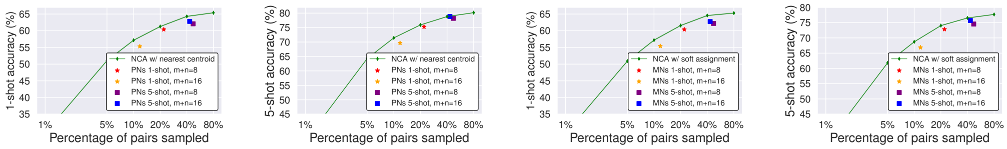

Figure 3: 1-shot and 5-shot accuracies on *mini*ImageNet (val. set) for NCA models trained by only sampling a fraction of the total number of available pairs in the batch (of size 256). Stars and squares represent models trained using the Prototypical or Matching Network loss, and are plotted on the x-axis based on the total number of distance pairs exploited in the loss, so that they can be directly compared with this "sub-sampling" version of the NCA. Reported values correspond to the mean accuracy of three models trained with different random seeds and shaded areas represent 95% confidence intervals. See Sec. 3.3 for details.

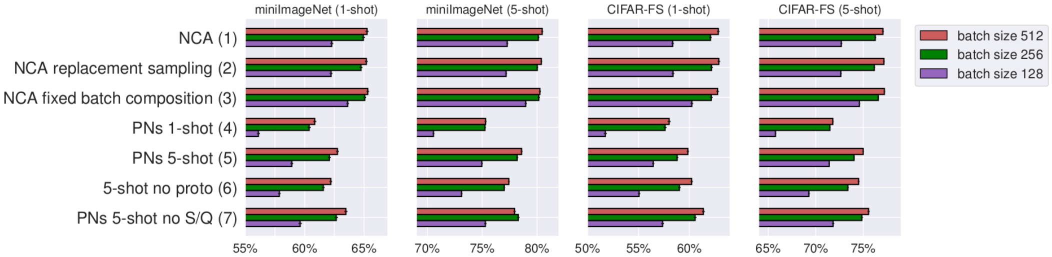

Figure 4: Ablation experiments on NCA and Prototypical Networks, both on batches (or episodes) of size 128, 256, and 512 on *mini*ImageNet and CIFAR-FS (val. set). Reported values correspond to the mean accuracy of three models trained with different random seeds and error bars represent 95% confidence intervals. See Sec. 3.4 for details.

3.4 Ablation experiments

To better analyse why NCA performs better, in this section we consider the three key differences discussed at the end of Sec. 2.2 by performing a series of ablations on models trained on batches of size 128, 256 and 512. Results are summarised in Fig. 4. We refer the reader to Appendix B to obtain detailed steps describing how these ablations affect the losses of Sec. 2.2.

First, we compare two variants of the NCA: one in which the sampling of the training batches happens sequentially and without replacement, as is standard in supervised learning, and one where batches are sampled with replacement. This modification (row 1 and 2 of Fig. 4) has a negligible effect, meaning that the replacement sampling introduced by episodic learning should 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. 2.1. To obtain a reasonably-performing model for both 1- and 5-shot models, we use configurations with $m + n = 8$. This means that, for PNs and MNs, models are trained with 8 images per class, and either 16, 32 or 64 classes (batches of size 128, 256 and 512 respectively). The batch size for NCA is also set to either 128, 256, or 512, allowing direct comparison.

The ablations of Fig. 4 compare PNs to NCA. First, we train standard PNs models (row 4 and 5 of Fig. 4). Next, we train a model where "prototypes" are not computed (row 6). This implies that, similar to what happens in MNs, distances are considered between individual points, but a separation between query and support set remains. This ablation allows us to investigate if the loss in performance by PNs compared to NCA can be attributed to prototype computation during training (which turned out not to be the case). Then, we perform an ablation where we ignore the separation between support and query set, 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 (row 7). 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 (row 3). 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 (row 3). 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

	*mini*ImageNet		CIFAR-FS		*tiered*ImageNet	
	1-shot	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Episodic methods						
adaResNet [31]	56.88 ± 0.62	71.94 ± 0.57	-	-	-	-
TADAM[32]	58.50 ± 0.30	76.70 ± 0.30	-	-	-	-
Shot-Free [38]	60.71± n/a	77.64± n/a	69.2± n/a	84.7± n/a	63.52± n/a	82.59± n/a
TEAM [35]	60.07± n/a	75.90± n/a	-	-	-	-
MTL [47]	61.20 ± 1.80	75.50 ± 0.80	-	-	-	-
TapNet [55]	61.65 ± 0.15	76.36 ± 0.10	-	-	63.08 ± 0.15	80.26 ± 0.12
MetaOptNet-SVM[28]	62.64 ± 0.61	78.63 ± 0.46	**72.0 ± 0.7**	84.2 ± 0.5	65.99 ± 0.72	81.56 ± 0.53
Variatonal FSL [56]	61.23 ± 0.26	77.69 ± 0.17	-	-	-	-
Simple cross-entropy baselines						
Transductive finetuning [12]	62.35 ± 0.66	74.53 ± 0.54	70.76 ± 0.74	81.56 ± 0.53	-	-
RFIC-simple [49]	62.02 ± 0.63	**79.64 ± 0.44**	71.5 ± 0.8	**86.0 ± 0.5**	**69.74 ± 0.72**	**84.41 ± 0.55**
Meta-Baseline [11]	**63.17 ± 0.23**	79.26 ± 0.17	-	-	68.62 ± 0.27	83.29 ± 0.18
Our implementations:						
MNs ([45] episodes)	58.91 ± 0.12	72.48 ± 0.10	69.28 ± 0.13	80.79 ± 0.10	65.75 ± 0.13	78.40 ± 0.10
PNs ([45] episodes)	59.78 ± 0.12	75.42 ± 0.09	69.94 ± 0.12	84.01 ± 0.09	65.80 ± 0.13	81.26 ± 0.10
MNs (our episodes)	60.77 ± 0.12	73.82 ± 0.09	71.86 ± 0.13	82.41 ± 0.10	66.53 ± 0.13	79.08 ± 0.10
PNs (our episodes)	61.32 ± 0.12	77.77 ± 0.09	70.41 ± 0.12	84.61 ± 0.10	66.89 ± 0.14	82.20 ± 0.09
SimpleShot [54]	62.16 ± 0.12	78.33 ± 0.09	69.98 ± 0.12	84.40 ± 0.09	66.67 ± 0.14	81.57 ± 0.10
NCA soft assignment (ours)	62.55 ± 0.12	76.93 ± 0.11	**72.49 ± 0.12**	83.38 ± 0.09	68.35 ± 0.13	81.04 ± 0.09
NCA nearest centroid (ours)	62.55 ± 0.12	78.27 ± 0.09	**72.49 ± 0.12**	85.15 ± 0.09	68.35 ± 0.13	83.20 ± 0.10

Table 2: Comparison of methods that use ResNet12 as f_θ, on the test set of *mini*ImageNet, CIFAR-FS, and *tiered*ImageNet. Values are reported with 95% confidence intervals. For our methods, reported values correspond to the mean accuracy of three models trained with different random seeds.

have a “no proto” ablation. 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. Since for PNs the amount of gradient signal is the same with (row 5, Fig.4) or without (row 6, Fig.4) the computation of prototypes, we believe that this could be motivated by the increased misalignment between the training and test setup present in the ablation of row 6. Nonetheless, enabling the computation between all pairs increases the performance (row 7) and, importantly, enabling *all* the ablations (row 3) completely recovers the performance lost by PNs. Note the meaningful gap in performance between row 1 and 3 in Fig. 4 for batch size 128, which disappears for batch size 512. This is likely due to the number of positives available in an excessively small batch size. Since our vanilla NCA creates batches by simply sampling images randomly from the dataset, there is a limit to how small a batch can be (which depends on the number of classes of the dataset). As an example, consider the extreme case of a batch of size 4. For the datasets considered, it is very likely that such a batch will contain no positive pairs for some classes. Conversely, the NCA ablation with a fixed batch composition (i.e. with a fixed number of images per class) will have a higher number of positive pairs (at the cost of a reduced number of classes per batch). This can explain the difference, as positive pairs constitute a less frequent (and potentially more informative) training signal. In Appendix E we extend this discussion, commenting on the role of positive and negative distances. In Appendix H we also report the results of a second set of ablations to compare NCA and Matching Networks, which are analogous to the ones with Prototypical Networks we just described and lead to the same conclusions.

These experiments highlight that the separation of roles between the images belonging to support and query set, which is typical of episodic learning [53], is detrimental for the performance of metric-based nonparametric few-shot learners. 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 hyperparameters for the creation of episodes $\{w, n, m\}$ no longer need to be considered.

3.5 Comparison with recent methods

We now evaluate our models on three popular FSL datasets to contextualise their performance with respect to the recent literature. When considering which methods to compare against, we chose those *a)* which have been recently published, *b)* that use a ResNet-12 architecture [28] (the most commonly used), and *c)* with a setup that is not significantly more complicated than ours. For example, we only report results for the main approach of Tian et al. [49]. We omit their self-distillation [17] variant, as it can be applied to most methods and involves multiple stages of training.

Results can be found in Table 2. Besides the results for the NCA loss, we also report PNs and MNs results with both the episodic setup from Snell et al. [45] and the best one (batch size 512, 5-shot, $m + n$=16 for both PNs and MNs) found from the experiment of Fig. 2, which brings a considerable improvement over the original and other PNs implementations (See Appendix I for a comparison of our PNs implementation to other works). Note that our aim is not to improve the state of the art, but rather to shed light on the practice of episodic learning. Nonetheless, our vanilla NCA is competitive and sometimes even superior to recent methods, despite being extremely simple. It fares surprisingly well against methods that use meta-learning (and episodic learning), and also against the high-performing simple baselines based on pre-training with the cross-entropy loss. Moreover, because of the explicit inductive bias that it encodes in terms of relative position in the embedding space of samples from the same class, the NCA loss is a useful tool to consider *alongside* cross-entropy trained baselines.

4 Related work

Pioneered by Utgoff [51], Schmidhuber [43, 44], Bengio et al. [4] and Thrun [48], the general concept of meta-learning is several decades old (for a survey see [52, 25]). 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 [42] uses a Neural Turing Machine [23] to save and access the information useful to meta-learn; Bertinetto et al. [6] and Munkhdalai et al. [30] propose a deep network in which a "teacher" branch is tasked with predicting the parameters of a "student" branch; Matching Networks [53] and Prototypical Networks [45] are two nonparametric 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 and Larochelle [37] propose instead to use an LSTM to learn the hyperparameters of SGD, while MAML [15] 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 [34, 12], fine-tuning [10], transductive fine-tuning [12] or logistic regression [49]. Wang et al. [54] suggest performing test-time classification by using the label of the closest centroid to the query image. Unlike these papers, which propose new methods, we are more focussed on shedding light on the possible causes behind the inefficiency of popular nonparametric few-shot learning algorithms such as Prototypical and Matching Networks.

Despite maintaining a support and a query set, the work of Raghu et al. [36] is similar in spirit to ours, and modifies episodic learning in MAML, showing that performance is almost entirely preserved when only updating the network head during meta-training and meta-testing. In this paper, we focussed on FSL algorithms just as established, and uncovered inefficiencies that not only allow for notable conceptual simplifications, but that also bring a significant boost in performance. Two related but different works are the ones of Goldblum et al. [22] and Fei et al. [13]. The former addresses PNs' poorly representative samples by training on episodic pairs with the same classes (but different instances) and using a regularizer enforcing consistency across them. The latter investigates meta-learning methods with *parametric* base learners, and shows interesting findings on the importance of having tightly clustered classes in feature space, which inspires a regularizer that improves non meta-learning models. Bai et al. [3] also show that the episodic strategy in meta-learning is inefficient by providing both theoretical and experimental arguments on methods solving a convex optimization problem at the level of the base learner. Similar to us, though via a different analysis, they show that the classic split is inefficient. Chen et al. [8] derive a generalisation bound for algorithms with a support/query separation. They do not provide any bounds for methods like NCA, which would be an interesting direction for future work. Triantafillou et al. [50] ignore the query/support separation in order to exploit all the available samples while working in a Structured SVM framework. Though the reasoning about batch exploitation is analogous to ours, the scope of the paper is very different. Finally, two recent meta-learning approaches based on Gaussian Processes [33, 46] also merge the support and query sets during learning to take full advantage of the available data within each episode.

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 investigate the role of episodes in popular nonparametric few-shot learning methods. We found that their performance is highly sensitive to the set of hyperparameters used to sample these episodes. By replacing the Prototypical Networks and Matching Networks losses with the closely related (and non-episodic) Neighbourhood Component Analysis, we were able to ignore these hyperparameters, while improving the few-shot classification accuracy. We found out that the performance discrepancy is in large part caused by the separation between support and query set within each episode, which negatively affects the number of pairwise distances contributing to the loss. Moreover, with nonparametric few-shot approaches, the episodic strategy is almost empirically equivalent to randomly discarding a portion of the distances available within a batch. Finally, we showed that our variant of the NCA achieves an accuracy on multiple popular FSL benchmarks that is competitive with recent methods, making it a simple and appealing baseline for future work.

Broader impact. We believe that progress in few-shot learning is important, as it can significantly impact important problems such as drug discovery and medical imaging. We also recognise that the capability of leveraging very small datasets might constitute a threat if deployed for surveillance by authoritarian entities (e.g. by applying it to problems such as re-identification and face recognition).

Acknowledgements and Disclosure of Funding

We would like to thank Jõao F. Henriques, Nicholas Lord, John Redford, Stuart Golodetz, Sina Samangooei, and the anonymous reviewers for insightful discussions and helpful suggestions to improve the manuscript and our analysis in general. This work was supported by Five AI Limited.

References

[1] K. R. Allen, E. Shelhamer, H. Shin, and J. B. Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, 2019.

[2] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande. Low data drug discovery with one-shot learning. *ACS central science*, 2017.

[3] Y. Bai, M. Chen, P. Zhou, T. Zhao, J. Lee, S. Kakade, H. Wang, and C. Xiong. How important is the train-validation split in meta-learning? In *International Conference on Machine Learning*, 2021.

[4] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas, 1992.

[5] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[6] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 2016.

[7] T. Cao, M. Law, and S. Fidler. A theoretical analysis of the number of shots in few-shot learning. In *International Conference on Learning Representations*, 2020.

[8] J. Chen, X.-M. Wu, Y. Li, Q. Li, L.-M. Zhan, and F.-l. Chung. A closer look at the training strategy for modern meta-learning. *Advances in Neural Information Processing Systems*, 2020.

[9] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[10] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. *International Conference on Learning Representations*, 2019.

[11] Y. Chen, X. Wang, Z. Liu, H. Xu, and T. Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

[12] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto. A baseline for few-shot image classification. In *International Conference on Learning Representations*, 2020.

[13] N. Fei, Z. Lu, T. Xiang, and S. Huang. Melr: Meta-learning via modeling episode-level relationships for few-shot learning. In *International Conference on Learning Representations*, 2021.

[14] C. Finn. Stanford cs330: Multi-task and meta-learning, 2019 | lecture 4 - non-parametric meta-learners. `https://www.youtube.com/watch?v=bc-6tzTyYcM&list=PLoROMvodv4rMC6zfYmnD7UG3LVvwaITY5&index=4`, 2019. Accessed on 26/01/2021.

[15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[16] N. Frosst, N. Papernot, and G. Hinton. Analyzing and improving representations with the soft nearest neighbor loss. *International Conference on Machine Learning*, 2019.

[17] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, 2018.

[18] G. García, R. Del Amor, A. Colomer, R. Verdú-Monedero, J. Morales-Sánchez, and V. Naranjo. Circum-papillary oct-focused hybrid learning for glaucoma grading using tailored prototypical neural networks. *Artificial Intelligence in Medicine*, 118:102132, 2021.

[19] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, 2018.

[20] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord. Boosting few-shot visual learning with self-supervision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[21] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2005.

[22] M. Goldblum, S. Reich, L. Fowl, R. Ni, V. Cherepanova, and T. Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*. PMLR, 2020.

[23] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[24] C. Guo, G. Pleiss, Y. Sun, and K. q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

[25] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[26] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[27] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

[28] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

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

[30] T. Munkhdalai and H. Yu. Meta networks. In *International Conference on Machine Learning*, 2017.

[31] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, 2018.

[32] B. Oreshkin, P. R. López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, 2018.

[33] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. J. Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 2020.

[34] H. Qi, M. Brown, and D. G. Lowe. Low-shot learning with imprinted weights. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[35] L. Qiao, Y. Shi, J. Li, Y. Wang, T. Huang, and Y. Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *IEEE International Conference on Computer Vision*, 2019.

[36] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[37] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[38] A. Ravichandran, R. Bhotika, and S. Soatto. Few-shot learning with embedded class models and shot-free meta training. In *IEEE International Conference on Computer Vision*, 2019.

[39] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[40] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, 2007.

[41] A. Salekin and N. Russo. Understanding autism: the power of eeg harnessed by prototypical learning. In *Proceedings of the Workshop on Medical Cyber Physical Systems and Internet of Medical Things*, pages 12–16, 2021.

[42] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

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

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

[45] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[46] J. Snell and R. Zemel. Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. *International Conference on Learning Representations*, 2021.

[47] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

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

[49] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola. Rethinking few-shot image classification: a good embedding is all you need? *European Conference on Computer Vision*, 2020.

[50] E. Triantafillou, R. S. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, 2017.

[51] P. E. Utgoff. Shift of bias for inductive concept learning. *Machine learning: An artificial intelligence approach*, 1986.

[52] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002.

[53] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[54] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[55] S. W. Yoon, J. Seo, and J. Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, 2019.

[56] J. Zhang, C. Zhao, B. Ni, M. Xu, and X. Yang. Variational few-shot learning. In *IEEE International Conference on Computer Vision*, 2019.

A Number of pairs lost with episodic batches

In this section we demonstrate that the total number of training pairs that the NCA loss can exploit within a batch is always strictly superior or equal to the one exploited by the episodic batch strategy used by Prototypical Networks (PNs) and Matching Networks (MNs).

To ensure we have a "valid" episodic batch with a nonzero number of both positive and negative distance pairs, we assume that $n, m \geq 1$, and $w \geq 2$. Below, we show that the number of positives for the NCA, i.e. $\binom{m+n}{2}w$, is always greater or equal than the one for PNs and MNs, which is mnw:

$$\begin{aligned}
\binom{m+n}{2}w &= \frac{(m+n)!}{2!(m+n-2)!}w \\
&= \frac{1}{2}(m+n)(m+n-1)w \\
&= \frac{1}{2}(m^2+2mn-m+n^2-n)w \\
&= \frac{1}{2}(m(m-1)+2mn+n(n-1))w \\
&\geq \frac{1}{2}(2mn)w = wmn.
\end{aligned}$$

Similarly, we can show for negative distance pairs that $\binom{w}{2}(m+n)^2 > w(w-1)mn$:

$$\begin{aligned}\binom{w}{2}(m+n)^2 &= \frac{w!}{2!(w-2)!}(m^2+2mn+n^2)\\ &= \frac{1}{2}w(w-1)(m^2+2mn+n^2)\\ &> \frac{1}{2}w(w-1)(2mn)\\ &= w(w-1)mn.\end{aligned}$$

This means that the NCA has at least the same number of positives as Prototypical and Matching Networks, and has always strictly more negative distances. In general, the total number of *extra* pairs that NCA can rely on is $\frac{w}{2}(w(m^2+n^2)-m-n)$.

B Details about the ablation studies of Section 3.4

Referring to the three key differences between the losses of Prototypical Networks, Matching Networks, and the NCA listed in Sec. 2.2, in this section we detail how to obtain the ablations we used to perform the experiments of Sec. 3.4.

We can "disable" the creation of prototypes (point II), which will change the prototypical loss $\mathcal{L}_{\text{PNs}}$ to the Matching Networks loss $\mathcal{L}_{\text{MNs}}$.

$$\mathcal{L}(S,Q) = \frac{-1}{|Q|}\sum_{\substack{(\mathbf{q}_i,y)\\ \in Q}} \log\left(\frac{\sum\limits_{\substack{(\mathbf{s}_j,y')\\ \in S_y}} \exp -\|\mathbf{q}_i-\mathbf{s}_j\|^2}{\sum\limits_{\substack{(\mathbf{s}_k,y'')\\ \in S}} \exp -\|\mathbf{q}_i-\mathbf{s}_k\|^2}\right).$$

This is similar to $\mathcal{L}_{\text{NCA}}$, where the positives are represented by the distances from Q to S_k, and the negatives by the distances from Q to $S \setminus S_k$. The only difference now is the separation of the batch into a query and support set.

Independently, we can "disable" point 1 for $\mathcal{L}_{\text{PNs}}$, which gives us

$$\mathcal{L}(S,Q) = \frac{-1}{|Q|+|C|}\sum_{\substack{(\mathbf{z}_i,y_i)\\ \in Q\cup C}} \log\left(\frac{\sum\limits_{\substack{(\mathbf{z}_j,y_j)\\ \in Q\cup C\\ y_j=y_i\\ i\neq j}} \exp -\|\mathbf{z}_i-\mathbf{z}_j\|^2}{\sum\limits_{\substack{(\mathbf{z}_k,y_k)\\ \in Q\cup C\\ k\neq i}} \exp -\|\mathbf{z}_i-\mathbf{z}_k\|^2}\right),$$

which essentially combines the prototypes with the query set, and computes the NCA loss on that total set of embeddings.

Finally, we can "disable" both point 1 and 2 for both $\mathcal{L}_{\text{MNs}}$ and $\mathcal{L}_{\text{PNs}}$, which gives us

$$\mathcal{L}(S,Q) = \frac{-1}{|Q|+|S|}\sum_{\substack{(\mathbf{z}_i,y_i)\\ \in Q\cup S}} \log\left(\frac{\sum\limits_{\substack{(\mathbf{z}_j,y_j)\\ \in Q\cup S\\ y_j=y_i\\ i\neq j}} \exp -\|\mathbf{z}_i-\mathbf{z}_j\|^2}{\sum\limits_{\substack{(\mathbf{z}_k,y_k)\\ \in Q\cup S\\ k\neq i}} \exp -\|\mathbf{z}_i-\mathbf{z}_k\|^2}\right).$$

This almost exactly corresponds to the NCA loss, where the only difference is the construction of batches with a fixed number of classes and a fixed number of images per class.

	*mini*ImageNet	
method	**5-shot val**	**5-shot test**
k-NN	75.82 ± 0.11	73.52 ± 0.10
Soft assignments	79.11 ± 0.12	77.16 ± 0.10
Nearest centroid	$\mathbf{80.61 \pm 0.12}$	$\mathbf{78.30 \pm 0.10}$
	CIFAR-FS	
k-NN	73.46 ± 0.12	80.94 ± 0.10
Soft assignments	76.12 ± 0.12	83.31 ± 0.10
Nearest centroid	$\mathbf{77.80 \pm 0.12}$	$\mathbf{85.13 \pm 0.10}$

Table 3: Comparison in terms of accuracy between the different classifications stategies of Section 2.3 when using models trained with the NCA loss.

C Differences between the NCA and contrastive losses

$\mathcal{L}_{\text{NCA}}$ is similar to the contrastive loss functions [26, 9] that are used in self-supervised learning and representation learning. The main differences are that *a)* in contrastive losses, the denominator only contains negative pairs and *b)* the inner sum in the numerator is moved outside of the logarithm in the supervised contrastive loss function from Khosla et al. [26]. We opted to work with the NCA loss because we found it performs better than the supervised constrastive loss in a few-shot learning setting. Using the supervised contrastive loss we only managed to obtain 51.05% 1-shot and 63.36% 5-shot performance on the *mini*Imagenet test set.

D Effectiveness of different classification strategies

In Table 3, we compare the inference methods discussed in Section 2.3 using embeddings trained with the NCA loss. It might sound surprising that the *nearest centroid* approach outperforms *soft assignments*, as the latter closely reflects NCA's training protocol. We speculate its inferior performance could be caused by poor model calibration [24]: since the classes between training and evaluation are disjoint, the model is unlikely to 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 ignored (their contribution is squashed toward zero). With the nearest centroid classification approach outliers might still represent an issue, but it is likely that their effect will be less dramatic.

E Extended discussion

Section 3.2 – Discussion on the number of pairs per episodic setup. In Table 4 we plot the number of positives and negatives (gradients contributing to the loss) for the NCA loss as well as for different episodic configurations of PNs, to see whether the difference in performance can be explained by the difference in the number of distance pairs that can be exploited in a certain batch configuration. This is often true, as within each sub-table (representing a different batch size) the ranking can almost be fully explained by the number of total pairs in the rightmost column. However, there are some exceptions to this: (i) the difference between 5-shot with m+n=16 and 5-shot with m+n=8 in (for all batch sizes), and (ii) the difference between 5-shot with m+n=32 and 1-shot with m+n=8 for batch size 128.

To understand (i), we can see that the number of positive pairs is much higher for m+n=16 than for m+n=8. Since the positive pairs constitute a less frequent (and potentially more informative) training signal, this can explain the difference. The m+n=32 variant has an even higher number of positives than m+n=16, but the loss in performance there could be explained by a drastically lower number of negatives, and by the fact that the number of ways used during training is lower.

To understand (ii), we see that the number of pairs for 5-shot with m+n=32 is higher than 1-shot with m+n=8. However, due to the small batch size of 128, the number of ways during training for 5-shot with m+n=32 is only 4, whereas for 1-shot with m+n=8 it is 16. This could explain the higher performance of 1-shot with m+n=8.

Batch size	1-shot	5-shot	Method	# pos	# neg	# tot
512	62.84 ± 0.13	77.07 ± 0.10	*NCA w/ nearest centroid*	1792	129024	130816
	60.53 ± 0.13	75.79 ± 0.11	PNs *5-shot m+n=16*	1760	54560	56320
	59.85 ± 0.13	74.99 ± 0.11	PNs *5-shot m+n=8*	960	60480	61440
	59.82 ± 0.13	74.44 ± 0.11	PNs *5-shot m+n=32*	2160	32400	34560
	58.01 ± 0.14	71.80 ± 0.11	PNs *1-shot m+n=8*	448	28224	28672
	51.75 ± 0.13	65.77 ± 0.12	PNs *1-shot m+n=16*	465	14415	14880
	41.49 ± 0.12	52.90 ± 0.12	PNs *1-shot m+n=32*	496	7440	7936
256	62.07 ± 0.14	76.26 ± 0.10	*NCA w/ nearest centroid*	384	32256	32640
	59.60 ± 0.13	74.64 ± 0.11	PNs *5-shot m+n=16*	880	13200	14080
	58.80 ± 0.13	74.03 ± 0.11	PNs *5-shot m+n=8*	480	14880	15360
	57.62 ± 0.13	72.53 ± 0.11	PNs *5-shot m+n=32*	1080	7560	8640
	57.61 ± 0.14	71.49 ± 0.11	PNs *1-shot m+n=8*	224	6944	7168
	48.46 ± 0.13	61.64 ± 0.12	PNs *1-shot m+n=16*	240	3600	3840
	37.96 ± 0.11	48.38 ± 0.11	PNs *1-shot m+n=32*	248	1736	1984
128	58.36 ± 0.14	72.69 ± 0.11	*NCA w/ nearest centroid*	64	8064	8128
	57.02 ± 0.13	72.05 ± 0.11	PNs *5-shot m+n=16*	440	3280	3520
	56.45 ± 0.13	71.42 ± 0.11	PNs *5-shot m+n=8*	240	3600	3840
	49.88 ± 0.13	64.50 ± 0.11	PNs *5-shot m+n=32*	540	1620	2160
	51.75 ± 0.13	65.77 ± 0.12	PNs *1-shot m+n=8*	112	1680	1792
	40.46 ± 0.12	52.27 ± 0.11	PNs *1-shot m+n=16*	120	840	960
	28.54 ± 0.09	33.50 ± 0.09	PNs *1-shot m+n=32*	124	372	496

Table 4: Number of positives and negatives used in the experiments of Figure 2 from the main paper. We also report the 1-shot and 5-shot accuracies on the validation set of CIFAR-FS using PNs and NCA with nearest centroid classification.

So, while indeed generally speaking the higher number of pairs the better (which is also corroborated by Figure 3 from the main paper, where moving right on the x-axis sees higher performance for both NCA and PNs), one should also consider how this interacts with the positive/negative balance and the number of classes present within a batch.

Section 3.4 – Discussion on the number of pairs used for the ablations. 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. To illustrate this, we use the formulas described in Table 1 from the main paper to compute the number of positives and negatives used for each ablation with a batch size of 256. Looking at the ablation results shown in Figure 4 from the main paper: for row 6 there are 480 positives, and 14,880 negatives; while for row 7 there are 576 positives and 20,170 negatives. In both cases, the number is significantly lower than the corresponding NCA with a fixed batch composition, which totals 896 positives and 31,744 negatives, which could explain the gap between row 6 (and 7) and row 3. Moreover, from row 5 (and 6) to row 7 we see a modest increase in performance, which can also be explained by the slightly larger number of distance pairs.

F Implementation details

Benchmarks. In our experiments, we use three popular FSL benchmarks. ***mini*ImageNet** [53] is a subset of ImageNet generated by randomly sampling 100 classes, each with 600 randomly sampled images. We adopt the commonly used splits from [37], with 64 classes for meta-training, 16 for meta-validation and 20 for meta-testing. **CIFAR-FS** [5] is an anagolous version of *mini*Imagenet, but for CIFAR-100. It uses the same sized splits and same number of images per split. ***tiered*ImageNet** [39] is also constructed from ImageNet, but contains 608 classes, with 351 training classes, 97 validation classes and 160 test classes. The class splits have been generated using WordNet [29] to ensure that the training classes are semantically "distant" to the validation and test classes. For all datasets, we use images of size 84×84.

Architecture. In all our experiments, f_θ is represented by a ResNet12 with widths $[64, 160, 320, 640]$. We chose this architecture, initially adopted by Lee et al. [28], as it is the one which is most frequently adopted by recent FSL methods. Unlike most methods, we do not use a DropBlock regulariser [19], as we did not notice it to meaningfully contribute to performance.

Optimisation. To train all the models used for our experiments, unless differently specified, we used a SGD optimiser with Nesterov momentum, weight decay of 0.0005 and initial learning rate of 0.1. For *mini*ImageNet and CIFAR-FS we decrease the learning rate by a factor of 10 after 70% of epochs have been trained, and train for a total of 120 epochs. As data augmentations, we use random horizontal flipping and centre cropping. For Section 3.5 only, we slightly change our training setup. On CIFAR-FS, we increase the number of training epochs from 120 to 240, which improved accuracy by about 0.5%. For *tiered*ImageNet, we train for 150 epochs and decrease the learning rate by a factor of 10 after 50% and 75% of the training progress. For *tiered*ImageNet only we increased the batch size to 1024, and train on 64 classes (like *mini*ImageNet and CIFAR-FS) and 16 images per class within a batch, as we found it being beneficial. These changes affect and are beneficial for all our implemented methods and baselines: NCA, Prototypical Networks, and Matching Networks (with both old and new batch setup), and SimpleShot [54].

Projection network. Similarly to [26, 9], we also experimented (for MNs, PNs, and NCA) with a *projection network* (but *only* for the comparison of Section 3.5). The projection network is a single linear layer $A \in \mathbb{R}^{M \times P}$ that is placed on top of f_θ at training time, where M is the output dimension of the neural network f_θ and P is the output dimension of A. The output of A is only used during training. At test time, we do not use the output of A and directly use the output of f_θ. For CIFAR-FS and *tiered*ImageNet, we found this did not help performance. For *mini*ImageNet, however, we found that this improved performance for the NCA – but not for PNs and MNs – and we set $P = 128$. Note that this is not an unfair advantage over other methods. Compared to SimpleShot [54] and other simple baselines, given that they use an extra fully connected layer to minimise cross entropy during pre-training, we actually use fewer parameters without the projection network (effectively making our ResNet12 a ResNet11).

Choice of hyperparameters. During the experimental design, we wanted to ensure a fair comparison between the NCA, PNs, and MNs. As a testimony of this effort, we obtained very competitive results for PNs (see for example the comparison to recent papers where architectures of similar capacity were used [54, 10]). In particular:

- We always use the normalisation strategy of SimpleShot [54], as it is beneficial also for both PNs and MNs.
- Unless expressively specified, we always used PNs' and MNs' 5-shot model, which in our implementation outperforms the 1-shot model (for both 1-shot and 5-shot evaluation). Instead, [45, 53] train and tests with the same number of shots.
- Apart from the episodes hyperparameters of PNs and MNs, which we did search and optimise over to create the plots of Figure 3.2 (main paper), the only other hyperparameters of PNs and MNs are those related to the training schedule, which are the same as the NCA. To set them, we started from the simple SGD schedule used by SimpleShot [54] and only marginally modified it by increasing the number of training epochs to 120, increasing the batch size to 512 and setting weight decay and learning rate to $5e-4$ and 0.1, respectively. To decide this setting, we run a small grid search and empirically verified that it is the best for all three methods. Moreover, as a sanity check we trained both the NCA and PNs with the exact training schedule used by SimpleShot [54]. Results are reported in Table 5, and show that the schedule we used for this paper is considerably better for both PNs and NCA. In general, we observed that the modifications were beneficial for NCA, PNs and MNs, and improvements in performance in NCA, PNs and MNs were highly correlated. This is to be expected given the high similarity between the three methods and losses.

Computing infrastructure details. For our experiments we had eight NVIDIA GeForce GTX 1080 Ti GPUs available. However, for each model we usually only needed 2 GPUs to train, except for the *tiered*ImageNet experiments using a batch size of 1024 where we needed 4 GPUs. All our *mini*ImageNet and CIFAR-FS experiments took about 2 hours to finish training, and the *tiered*ImageNet experiments took 8 hours per model.

G Performance improvements

Adapting to the support set. None of the algorithms we considered perform any kind of parameter adaptation at test time. On the one hand this is convenient, as it allows fast inference; on the other hand, useful information from the support set S might remain unexploited.

	*mini*ImageNet		CIFAR-FS	
method	**1-shot**	**5-shot**	**1-shot**	**5-shot**
PNs (SimpleShot)	57.99 ± 0.21	74.33 ± 0.16	53.76 ± 0.22	68.54 ± 0.19
PNs (ours)	62.79 ± 0.12	78.82 ± 0.09	59.60 ± 0.13	74.64 ± 0.11
NCA (SimpleShot)	61.21 ± 0.22	76.39 ± 0.16	59.41 ± 0.24	73.29 ± 0.19
NCA (ours)	64.94 ± 0.13	80.12 ± 0.09	62.07 ± 0.14	76.26 ± 0.10

Table 5: Comparison (on the validation set of *mini*ImageNet and CIFAR-FS) between using the hyper-parameters from SimpleShot [54] and the ones from this paper (**ours**). Models have been trained with batches of size 256, as in [54]. PNs episodic batch setup uses $m + n$=16, as it is the highest performing one. NCA is evaluated using nearest centroid classification.

	*mini*ImageNet		CIFAR-FS	
method	**1-shot**	**5-shot**	**1-shot**	**5-shot**
NCA	62.52 ± 0.24	78.3 ± 0.14	72.48 ± 0.40	85.13 ± 0.29
NCA multi-layer	63.21 ± 0.08	79.27 ± 0.08	72.44 ± 0.36	85.42 ± 0.29
NCA (ours) multi-layer + ss	-	$\mathbf{79.79 \pm 0.08}$	-	$\mathbf{85.66 \pm 0.32}$

Table 6: Comparison between vanilla NCA, NCA using multiple evaluation layers and NCA performing optimisation on the support set (ss). The NCA can only be optimised in the 5-shot case, since there are not enough positives distances in the 1-shot case. Optimisation is conducted for 5 epochs using Adam, with learning rate 0.0001 and weight decay 0.0005. NCA is always evaluated using nearest centroid classification.

In the 5-shot case it is possible to minimise the NCA loss since it can directly be computed on the support set: $\mathcal{L}_{\text{NCA}}(S)$. We tried training a positive semi-definite matrix A on the outputs of the trained neural network, which corresponds to learning a Mahalanobis distance metric as in Goldberger et al. [21]. However, we found that there was no meaningful increase in performance. Differently, we did find that fine-tuning the whole neural network f_θ by $\arg\min_\theta \mathcal{L}_{\text{NCA}}(S)$ is beneficial (see Table 6). Nonetheless, given the computational cost, we opted for non performing adaptation to the support sets in any of our experiments.

Features concatenation. For NCA, we also found that concatenating the output of intermediate layers modestly improves performance at (almost) no additional cost. We used the output of the average pool layers from all ResNet blocks except the first and we refer to this variant as NCA multi-layer. However, since this is an orthogonal variation, we do not consider it in any of our experiments. Results on *mini*ImageNet and CIFAR-FS are shown in Table 6.

H Additional results for Section 3.2

Figure 5 and Figure 6 complement the results of Figure 2 and Figure 4 from the main paper, respectively. As can be seen, these figures support the main conclusions made in the paper: in Figure 5 we can see that the NCA loss outperforms all the episodic setups also on *mini*ImageNet. Note that, since there are no prototypes in Matching Networks, Figure 6 only has one ablation. It is then clear from the figure that, similarly to what we showed for Prototypical Networks in Figure 4 from the main paper, disregarding the separation between support and query set is beneficial for Matching Networks as well.

I Comparing to other PN implementations

In Table 7 we compare our implementation and hyperparameter selection for PNs to previous works that have re-implemented PNs using variants of ResNet. We can see that our implementation achieves the best results, indicating that the better performance achieved by NCA is not the result of an uneven hyperparameter search.

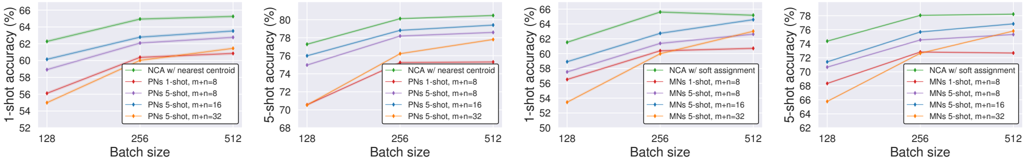

Figure 5: 1-shot and 5-shot accuracies on *mini*ImageNet (val. set) for Prototypical and Matching Networks models trained with different episodic configurations: 1-shot with $m + n$=8 and 5-shot with $m + n$=8, 16 or 32. NCA models are trained on batches of size 128, 256 and 512 to match the size of the episodes. Reported values correspond to the mean accuracy of three models trained with different random seeds and shaded areas represent 95% confidence intervals. See Sec. 3.2 for details.

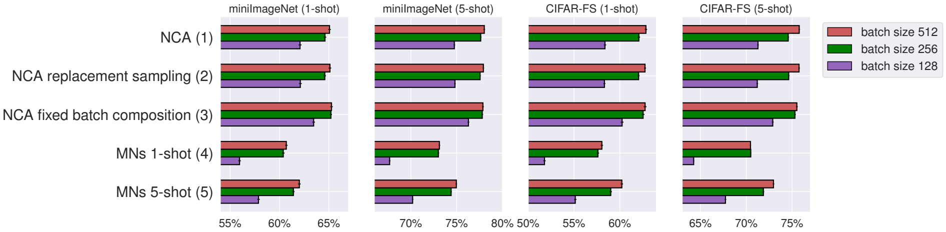

Figure 6: Ablation experiments on NCA and Matching Networks, both on batches (or episodes) of size 128, 256, and 512 on *mini*ImageNet and CIFAR-FS (val. set). Reported values correspond to the mean accuracy of three models trained with different random seeds and error bars represent 95% confidence intervals. See Sec. 3.4 for details.

PNs Implementation	Architecture	miniImageNet	
		1-shot	**5-shot**
[8]	ResNet10	51.98 ± 0.84	72.64 ± 0.64
[8]	ResNet18	54.16 ± 0.82	73.68 ± 0.65
[8]	ResNet34	53.90 ± 0.83	74.65 ± 0.64
[20]	WideResNet-28-10	55.85 ± 0.48	68.72 ± 0.36
[28]	ResNet12	59.25 ± 0.64	75.60 ± 0.84
Ours ([45] episodes)	ResNet12	59.78 ± 0.12	75.42 ± 0.09
Ours (best episodes)	ResNet12	$\mathbf{61.32 \pm 0.12}$	$\mathbf{77.77 \pm 0.09}$

Table 7: Comparison (on the test set *mini*ImageNet) across papers between implementations of PNs [45] on ResNet architectures. Our best episode configuration is selected from experiments in Section 3.2.