	*mini*ImageNet	
method	**5-shot val**	**5-shot test**
Soft Assignment	79.11 ± 0.27	77.16 ± 0.10
k-NN	75.82 ± 0.21	73.52 ± 0.12
1-NN centroid	$\mathbf{80.61 \pm 0.20}$	$\mathbf{78.30 \pm 0.14}$
	CIFAR-FS	
Soft Assignment	76.12 ± 0.32	83.31 ± 0.37
k-NN	73.46 ± 0.39	80.94 ± 0.38
1-NN centroid	$\mathbf{77.80 \pm 0.35}$	$\mathbf{85.13 \pm 0.32}$

Table 2: Comparison between the different evaluation methods discussed in Sec. 3.4

A Appendix

A.1 Performance improvements

Adapting to the support set. Prototypical Networks does not 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.

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. (2005). 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 $\arg\min_\theta \mathcal{L}_{\text{NCA}}(S)$ was beneficial (see Table 3). However, given the computational cost, we opted for non performing adaptation to the support sets in our experiments of Sec. 4.

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 that can be applied to several methods, we do not consider it for our experiments of Sec. 4.

Results on *mini*ImageNet an CIFAR-FS are shown in Table 3.

A.2 Details about the ablation studies of Section 4.3

Referring to the three key differences between the Prototypical Networks and the NCA losses listed in Sec. 3.3, in this section we detail how to obtain the ablations we used to perform the experiments of Sec. 4.3.

We can "disable" the creation of prototypes (point 1), which will change the prototypical loss of Eq. 2 to

$$\mathcal{L}(S, Q) = -\frac{1}{|Q| + |S|} \sum_{(\mathbf{q}_i, y) \in Q} \log \left(\frac{\sum\limits_{(\mathbf{s}_j, y') \in S_y} \exp -\|\mathbf{q}_i - \mathbf{s}_j\|^2}{\sum\limits_{(\mathbf{s}_k, y'') \in S} \exp -\|\mathbf{q}_i - \mathbf{s}_k\|^2} \right). \tag{4}$$

This is similar to $\mathcal{L}_{\text{NCA}}$ (Eq. 3), 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 2, which gives us

$$\mathcal{L}(S, Q) = -\frac{1}{|Q| + |S|} \sum_{(\mathbf{z}_i, y_i) \in Q \cup C} \log \left(\frac{\sum\limits_{\substack{(\mathbf{z}_j, y_j) \in Q \cup C \\ y_j = y_i \\ i \neq j}} \exp -\|\mathbf{z}_i - \mathbf{z}_j\|^2}{\sum\limits_{\substack{(\mathbf{z}_k, y_k) \in Q \cup C \\ k \neq i}} \exp -\|\mathbf{z}_i - \mathbf{z}_k\|^2} \right), \tag{5}$$

	*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 3: 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. Support set is optimised for 5 epochs using Adam with learning rate 0.0001 and weight decay 0.0005. For details, see Sec. A.1

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, which gives us

$$\mathcal{L}(S,Q) = -\frac{1}{|Q|+|S|} \sum_{(\mathbf{z}_i,y_i)\in Q\cup S} \log \left(\frac{\sum\limits_{\substack{(\mathbf{z}_j,y_j)\in Q\cup S \\ y_j=y_i \\ i\neq j}} \exp -\|\mathbf{z}_i - \mathbf{z}_j\|^2}{\sum\limits_{\substack{(\mathbf{z}_k,y_k)\in Q\cup S \\ k\neq i}} \exp -\|\mathbf{z}_i - \mathbf{z}_k\|^2} \right). \tag{6}$$

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.

A.3 Differences between the NCA and contrastive losses

Eq. 3 is similar to the contrastive loss functions (Khosla et al., 2020; Chen et al., 2020a) that are used in self-supervised learning and representation learning. The main differences are that 1.) In contrastive losses, the denominator only contains negative pairs and 2.) the inner sum in the numerator is moved outside of the logarithm in the supervised contrastive loss function from Khosla et al. (2020). 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.

A.4 Implementation details

Benchmarks. In our experiments, we use three popular FSL benchmarks. ***mini*ImageNet** (Vinyals et al., 2016) is a subset of ImageNet generated by randomly sampling 100 classes, each with 600 randomly sampled images. We adopt the commonly used splits of Ravi & Larochelle (2017) who use 64 classes for meta-training, 16 for meta-validation and 20 for meta-testing. **CIFAR-FS** was proposed by Bertinetto et al. (2019) as an anagolous version of *mini*Imagenet for CIFAR-100. It uses the same sized splits and same number of images per split as *mini*ImageNet. ***tiered*ImageNet** (Ren et al., 2018) is also constructed from ImageNet, but contains 608 classes, with 351 training classes, 97 validation classes and 160 test classes. The class split have been generated using WordNet (Miller, 1995) 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 introduced by Lee et al. (2019), as it is the one which is most frequently adopted by recent FSL methods. Unlike most methods, we do not use a DropBlock regulariser (Ghiasi et al., 2018), 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

	*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 4: Comparison of results on validation set of *mini*ImageNet and CIFAR-FS using the hyperparameters used in SimpleShot(Wang et al., 2019) and the hyperparameters used in this paper (**ours**). Results are on batch size 256 (as used in Wang et al. (2019)) with the PNs episodic batch being a=16, as it is the best performing episodic setup we found.

epochs have been trained, and train for a total of 120 epochs. As data augmentations, we use random horizontal flipping and centre cropping.

Only for the experiments of Sec. 4.4, we slightly change our training setup. On CIFAR-FS, we increase the number of training epochs from 120 to 240, which improved accuracy of about 0.5%. For *tiered*ImageNet, we train for 120 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, as we found it being beneficial. For the other datasets it did not improve performance. These changes affect all our methods and baselines: NCA, Prototypical Networks (with both old and new batch setup), and SimpleShot (Wang et al., 2019).

Projection network. Similarly to (Khosla et al., 2020; Chen et al., 2020a), we also experimented (for both PNs and NCA) with a *projection network* (but *only* for the comparison of Sec. 4.4). 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, which can be considered as a hyper-parameter. 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, and we set $P = 128$ (which worked best for both PNs and NCA). Note that this is not an unfair advantage over other methods. Compared to SimpleShot (Wang et al., 2019) and other simple baselines, we actually use fewer parameters without the projection network (effectively making our ResNet12 a ResNet11) since they use an extra fully connected layer to minimise cross entropy during pre-training.

A.5 Choice of hyper-parameters

During the experimental design, we wanted to ensure a fair comparison between the NCA and PNs. 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 (Wang et al., 2019; Chen et al., 2019)). In particular:

- We always use the normalisation strategy of Wang et al. (2019), as it is beneficial also for PNs.
- Unless expressively specified, we always used PNs 5-shot model, which in our implementation outperforms the 1-shot model (for both 1-shot and 5-shot evaluation). Instead, (Snell et al., 2017) train and tests with the same number of shots.
- Apart from the episodes hyper-parameters of PNs, which we did search and optimise over to create the plots of Fig. 2, the only other hyper-parameters of PNs 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 Wang et al. (2019) 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. As a sanity check, we trained both the NCA and PNs with the exact training schedule used by Wang et al. (2019). Results are reported in Table 4, 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 both

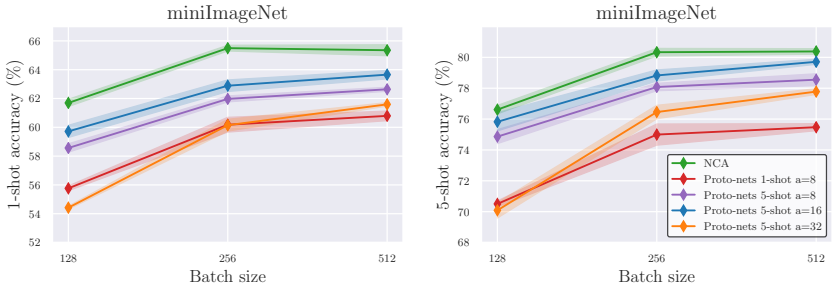

Figure 5: 1-shot (left) and 5-shot accuracies (right) on the validation set of *mini*ImageNet for different batch sizes. Models are trained using NCA or Proto-nets with different configurations: 1-shot with $a = 8$ and 5-shot with $a = 8$, 16 or 32. Reported values correspond to the mean accuracy of five models trained with different random seeds. Please see Sec. 4.2 for details.

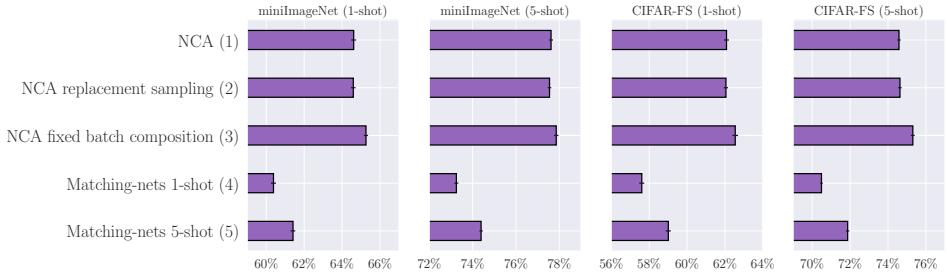

Figure 6: Ablation experiments on NCA and Matching Networks, both on batches or episodes of size 256 on the validation set of *mini*ImageNet and CIFAR-FS. All methods use *soft assignment* (Sec. 3.4) at test time.

NCA and PNs, and improvements in performance in NCA and PNs were highly correlated. This is to be expected given the high similarity between the two methods and losses.

A.6 Additional results for Sec. 4.2

Fig. 5 complements the results of Fig. 2 from Sec. 4.2

A.7 Additional results for Sec. 4.3

Matching Networks (Vinyals et al., 2016) are closely related to PNs (Snell et al., 2017), and even equivalent in the 1-shot case. The difference is in the use of the support set in the multi-shot case. Whereas PNs generate prototypes by averaging the embedding of the support set, Matching Networks adopt a weighted nearest-neighbour approach using an attention mechanism. If the attention mechanism is a softmax over the distances (which the authors suggest in Sec. 2.1.1 of their paper), we obtain the soft-assignment approach discussed in Sec. 3.4 of this paper. The only two differences between Matching Networks with a softmax attention mechanism and PNs are the lack of protoypes and the use of the cosine distance, instead of the Euclidean distance (Snell et al. (2017) has shown that the Euclidean distance is a better choice in FSL).

Given this similarity, and because of the relevance Matching Networks has in the few-shot learning community, we repeated the ablation experiment of Fig. 4. The results can be found in Table 6. In particular, we perform experiments on Matching Networks (without a Full Context Embedding) using the softmax attention mechanism, and using a Euclidean distance metric instead of a cosine distance metric. At training time, Matching Networks corresponds to the "no prototype" method in row 6 of Fig. 4. Therefore, the only difference between Matching Networks and NCA during training is the separation between the support and query set, leaving us with only one ablation to perform. At test time, evaluating Matching Networks is equivalent to using the soft-assignment approach described in Sec. 3.4. Therefore, for a fair comparison, for both NCA and "NCA fixed-batch composition" methods we also use the soft-assignment evaluation at test time.

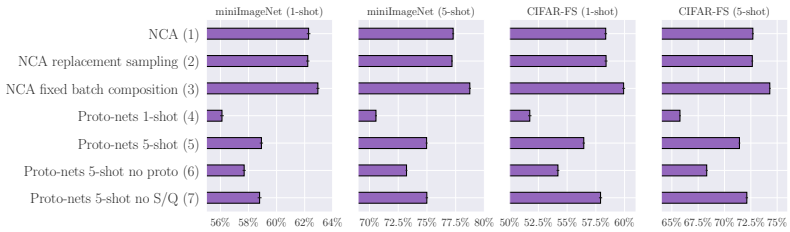

Figure 7: Ablation experiments on NCA and Prototypical Networks, both on batches or episodes of size 128 on the validation set of *mini*ImageNet and CIFAR-FS. Please refer to Sec. 4.3 for details.

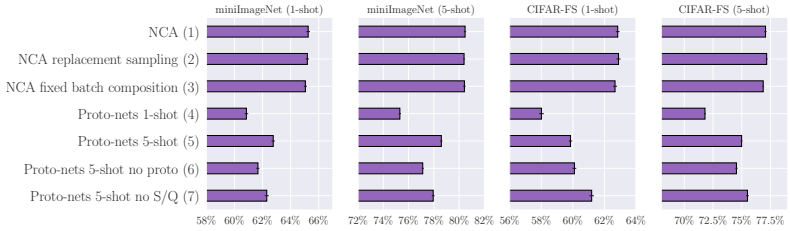

Figure 8: Ablation experiments on NCA and Prototypical Networks, both on batches or episodes of size 512 on the validation set of *mini*ImageNet and CIFAR-FS. Please refer to Sec. 4.3 for details.

As we can see, disregarding the separation between the support and query set also improves the performance of Matching Networks, and significantly so. This corroborates the findings of Sec. 4.3: the separation of roles between images in the support and query sets, typical of episodic learning, is detrimental to the performance of not only PN, but also Matching Networks. Instead, using the (closely related) NCA on standard random mini-batches allows for better exploitation of the training data, while simultaneously simplifying the training procedure.

Ablation experiments for different batch sizes. We repeated the ablation experiments done for batch size 256 (Fig. 4) also for size 128 and 512. Results can be found in Fig. 7 and Fig. 8. As we can see, the overall trend is maintained. A difference is the meaningful gap in performance between row 1 and 3 in Fig. 7 (size 128), which disappers in Fig. 8 (batch 512).

This is likely due to the number of positives available in an excessively small batch size. Since our vanilla NCA relies on using distance pairs and 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 a batch size of 128 and a training set of 64 classes, with a parameter-free sampler the NCA will have in expectation only one positive pair per class. Conversely, the NCA ablation with a fixed batch composition (i.e. with a set number of images per class) will have a higher number of positive pairs (at the cost of a reduced number of classes per batch). We believe this can explain the difference, as positive pairs constitute a less frequent (and potentially more informative) training signal. For the sake of simplicity, and since this only affects smaller batch sizes, we opted to use a vanilla, parameter-free sampler for the NCA in the rest of our experiments. Notice that, for batch size 512, there is even a slight (0.2%) decrease in performance using the fixed-batch composition w.r.t. the vanilla NCA.

A.8 Details about number of pairs description of Section 3.3

In this section we demonstrate that the total number of training pairs that NCA can exploit within a batch is always strictly superior than the one exploited by the episodic batch strategy used by Prototypical Networks.

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 NCA, i.e. $\binom{m+n}{2} w$, is always greater or equal than the one for PNs, which is mnw:

rank	method	# pos	# neg	# total pairs
1	*NCA*	1792	129024	130816
2	*5-shot a=16*	1760	54560	56320
3	*5-shot a=8*	960	60480	61440
4	*5-shot a=32*	2160	32400	34560
5	*1-shot a=8*	448	28224	28672

Table 5: Number of positives and negatives used in the batch size 512 experiments of Fig. 2.

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

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 Networks, and always has strictly more negative distances.

The total number of *extra* pairs that NCA can rely on is $\frac{w}{2}(w(m^2+n^2)-m-n)$.

A.9 Details about number of pairs description of Section 4.2

In Table 5 we plot the number of positives and negatives (gradients contributing to the loss) for the NCA and 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 the ranking can almost be fully explained by the number of total pairs in the right column. However, there are two exceptions to this: 5-shot with a=16 and 5-shot with a=8.

To understand this, we can see that the number of positive pairs is much higher for a=16 than for a=8. Since the positive pairs constitute a less frequent (and potentially more informative) training signal, this can explain the difference. The a=32 variant has an even higher number of positives than a=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. So, while indeed generally speaking the higher number of pairs the better (which is also corroborated by Fig. 3, 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.

A.10 Comparison with the state-of-the-art

We now benchmark our models on three FSL datasets, with the purpose of contextualising their performance against the modern literature.

	*mini*ImageNet		CIFAR-FS	
	1-shot	**5-shot**	**1-shot**	**5-shot**
Episodic methods				
adaResNet (Munkhdalai et al., 2018)	56.88 ± 0.62	71.94 ± 0.57	-	-
TADAM(Oreshkin et al., 2018)	58.50 ± 0.30	76.70 ± 0.30	-	-
Shot-Free (Ravichandran et al., 2019)	$60.71\pm$ n/a	$77.64\pm$ n/a	$69.2\pm$ n/a	$84.7\pm$ n/a
TEAM (Qiao et al., 2019)	$60.07\pm$ n/a	$75.90\pm$ n/a	-	-
MTL (Sun et al., 2019)	61.20 ± 1.80	75.50 ± 0.80	-	-
TapNet (Yoon et al., 2019)	61.65 ± 0.15	76.36 ± 0.10	-	-
MetaOptNet-SVM(Lee et al., 2019)	62.64 ± 0.61	78.63 ± 0.46	$\mathbf{72.0 \pm 0.7}$	84.2 ± 0.5
Variatonal FSL (Zhang et al., 2019)	61.23 ± 0.26	77.69 ± 0.17	-	-
Simple baselines				
Transductive finetuning (Dhillon et al., 2020)	62.35 ± 0.66	74.53 ± 0.54	70.76 ± 0.74	81.56 ± 0.53
RFIC-simple (Tian et al., 2020)	62.02 ± 0.63	$\mathbf{79.64 \pm 0.44}$	71.5 ± 0.8	86.0 ± 0.5
Meta-Baseline (Chen et al., 2020b)	$\mathbf{63.17 \pm 0.23}$	79.26 ± 0.17	-	-
Our implementations:				
Proto-nets (Snell et al. (2017) setup)	59.93 ± 0.23	75.89 ± 0.16	70.20 ± 0.22	83.96 ± 0.16
Proto-nets (our setup)	61.32 ± 0.23	77.77 ± 0.15	70.41 ± 0.31	84.46 ± 0.29
SimpleShot (Wang et al., 2019)	62.16 ± 0.23	78.33 ± 0.17	70.01 ± 0.21	84.50 ± 0.11
NCA (ours)	62.52 ± 0.24	78.3 ± 0.14	$\mathbf{72.48 \pm 0.40}$	$\mathbf{85.13 \pm 0.29}$

Table 6: Comparison of methods that use ResNet12 on *mini*ImageNet and CIFAR-FS (test set).

	*tiered*ImageNet	
Method	**1-shot**	**5-shot**
Shot-Free (Ravichandran et al., 2019)	$63.52\pm$ n/a	$82.59\pm$ n/a
RFIC-simple (Tian et al., 2020)	$\mathbf{69.74 \pm 0.72}$	$\mathbf{84.41 \pm 0.55}$
Meta-Baseline (Chen et al., 2020b)	68.62 ± 0.27	83.29 ± 0.18
MetaOptNet-SVM(Lee et al., 2019)	65.99 ± 0.72	81.56 ± 0.53
Our implementations:		
Proto-nets (Snell et al. (2017) setup)	65.45 ± 0.23	81.14 ± 0.17
SimpleShot (Wang et al., 2019)	66.17 ± 0.15	80.64 ± 0.20
NCA (ours)	68.36 ± 0.11	83.20 ± 0.18

Table 7: Comparison of methods that use ResNet12 on *tiered*ImageNet (test set).

When considering which methods to compare against, we chose those *a)* which have been recently published, *b)* that are using a ResNet12 (which we found the most commonly used) and *c)* with a setup that is not significantly more complicated than ours. For example, we only report the results of the main approach proposed by Tian et al. (2020) and not their *sequential self-distillation* (Furlanello et al., 2018) variant, which requires re-training multiple times and can be applied to most methods.

Results can be found in Table 6 for *mini*ImageNet and CIFAR-FS and Table 7 for *tiered*ImageNet. In Table 6 we report Prototypical Networks results for both the episodic setup from Snell et al. (2017) and the best one (batch size 512, 5-shot, a=16) found from the experiment of Fig. 2, which brings a considerable improvement over the original. We did not optimised for a new setup for Prototypical Networks on *tiered*ImageNet, as the larger dataset and the higher number of classes would have made the hyper-parameter search too demanding. Notice how our vanilla NCA is competitive or superior to recent methods, despite being extremely simple. It fairs 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.