Supplementary Material: Learning Intra-Batch Connections for Deep Metric Learning

Jenny Seidenschwarz [1] **Ismail Elezi** [1] **Laura Leal-Taixé** [1]

Abstract

In this supplementary, we show the performance of our approach using different embedding sizes on Stanford Online Products and In-Shop Clothes *(Section 1)*, show the performance of our approach on using larger images *(Section 2)*, compare the test time settings presented in the main work to other possible settings *(Section 3)*, show more qualitative results *(Section 4)*, visualize our results using t-SNE (van der Maaten & Hinton, 2012) *(Section 5)*, visualize embeddings of the backbone CNN when trained with and without MPN *(Section 6)*, give an analysis on different numbers of message passing steps and attentions heads on Stanford Online Products *(Section 7)*, analyse the impact of the batch construction process *(Section 8)*, provide a reality check of our approach *(Section 9)* and, finally, show how our method can be used with a large number of classes *(Section 10)*.

1. Different Embedding Sizes

To investigate the robustness of our approach concerning different embedding sizes, we present the performance for several embedding dimensions on Stanford Online Products and In Shop Clothes, which complement the results presented on CUB-200-2011 and Cars196 in the main paper (see Figure 6 in the main paper). The results with embedding dimension up to 1024 can be found in Table 1. As one can see, the performance on the smaller datasets increases with increasing embedding dimension similar to Proxy Anchor (Kim et al., 2020) while the performance on Stanford Online Products remains the same. On the other hand, similar to the Multi-Similarity loss (Wang et al., 2019) the performance is shown to decrease on the In-Shop dataset if the size of the embedding layer becomes larger than 512.

[1]Department of Computer Science, Technical University of Munich, Munich, Germany. Correspondence to: Jenny Seidenschwarz <jenny.seidenschwarz@tum.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

2. Large Images and ProxyNCA++

Different from most approaches in the field of deep metric learning, (Teh et al., 2020; Jacob et al., 2019) crop the images during training to size 256×256, while at test time they first resize to 288×288 before again cropping to 256×256. Naturally, larger images are expected to lead to an increased performance. In the main work, for fair comparison, we report the performance of (Teh et al., 2020) on the typical image size, i.e., 227×227. To obtain these numbers we ran their provided code[1] and compared the results to the Recall@1 given in their supplementary to validate the correctness, see Table 2.

We also evaluate the performance of our approach on images of size 256×256 to show that our performance also increases when using larger images. As can be seen in Table 3, when we use larger images, our performance increases by $1.4pp$ Recall@1 and $0.3pp$ NMI on CUB-200-2011 and $1.9pp$ Recall@1 and $0.6pp$ NMI on Cars196. This leads to a even larger increase in performance compared to (Teh et al., 2020; Jacob et al., 2019). We also outperform (Kim et al., 2020) who also gave additional results for larger-size images (we already showed in the main paper that we outperform (Kim et al., 2020) on regular-sized images).

3. Different Settings during Test Time

In this section, we detail three methods for batch construction when using the MPN during inference, as well as a teacher-student approach to avoid batch construction at test time.

3.1. Batch Construction Based on Clustering

Ideally, we imitate the batch construction process that happens during training, i.e., sampling n classes and taking k samples for each. As we can not access ground truth labels during test time, we are not able to construct batches in this manner.

To this end, we first use randomly sampled batches to generate initial feature vectors using the backbone CNN. We

[1]`https://github.com/euwern/proxynca_pp`

	CUB-200-2011		Cars196		Stanford Online Products		In-Shop Clothes
	R@1	**NMI**	**R@1**	**NMI**	**R@1**	**NMI**	**R@1**
Dimension 64	61.8	68.8	76.3	68.1	73.8	91.2	87.5
Dimension 128	65.1	69.3	81.6	69.4	79.1	92.3	91.3
Dimension 256	68.4	73.4	86.3	72.1	80.8	92.6	92.5
Dimension 512	70.3	**74.0**	88.1	74.8	**81.4**	**92.6**	**92.8**
Dimension 1024	**71.8**	72.8	**89.5**	**75.0**	**81.4**	92.5	91.7

Table 1. Performance of our approach using different embedding sizes on CUB-200-2011, Cars196, Stanford Online Products and In-Shop Clothes datasets.

	CUB-200-2011	Cars196	Stanford Online Products	In-Shop Clothes
Our results on (Teh et al., 2020)	66.3	84.9	79.8	90.4
Results in (Teh et al., 2020)	64.7 ± 1.6	85.1 ± 0.3	79.6 ± 0.6	87.6 ± 1.0

Table 2. Comparison of Recall@1 on images of size 227×227 using ProxyNCA ++ (Teh et al., 2020) of the results reported in (Teh et al., 2020) and our results obtained by running their code.

	CUB-200-2011		Cars196		Stanford Online Products		In-Shop Clothes
	R@1	**NMI**	**R@1**	**NMI**	**R@1**	**NMI**	**R@1**
Horde^{512†} (Jacob et al., 2019)	66.3	-	83.9	-	80.1	-	90.4
Proxy NCA++^{512†} (Teh et al., 2020)	69.0	73.9	86.5	73.8	80.7	-	90.4
Proxy Anchor^{512†} (Kim et al., 2020)	71.1	-	88.3	-	80.3	-	92.6
Ours^{512}	70.3	74.0	88.1	74.8	81.4	92.6	92.8
Ours^{512†}	**71.7**	**74.3**	**90.2**	**75.4**	**81.7**	**92.3**	**92.9**

Table 3. Performance of our approach using larger images compared to the approaches that only report their results on larger images. † indicates results on larger images. Proxy Anchor (Kim et al., 2020) presents the results both in regular (shown in the tables in the main paper) and large size images.

then use an approximation of the ground truth class assignment by using a clustering algorithm. We compare 6 common clustering algorithms as can be seen in Table 4. Based on these clusters, we construct batches by sampling from n clusters, k samples each, analogous to the training procedure. This way, we ensure that every sample in the mini-batch communicates with samples from its own cluster (similar) and other clusters (dissimilar). We call the k samples belonging to one cluster a *chunk*. Finally, we compute refined feature vectors using MPNs and use those features for retrieval and clustering.

In general, clustering algorithms can be divided into two groups: the ones that require a fixed number of clusters to be generated, and the density-based cluster algorithms that need a minimum number of samples per cluster as well as a maximum distance ϵ between two samples to be considered as neighbors.

As in theory, we do not know the ground truth number of clusters during test time, we cannot use it for the cluster construction in the first group. Therefore, we conduct experiments on several different numbers of clusters and report the results on the number that performs best. To be specific, the algorithms achieve the best performance if we set the number of clusters to 900. This number is significantly larger than the number of ground truth classes which is 100 and 98 for CUB-200-2011 and Cars196, respectively. Intuitively, using at least as many clusters as the number of classes is necessary since otherwise samples of different classes will be assigned to the same clusters. As *overclustering* constructs more clusters than the ground truth number of classes and, therefore, smaller clusters than the ground truth class sizes, the cluster assignment is less prone to outliers. The performance of the algorithms that need a fixed number of classes can be seen in Table 4, indicated by †. However, they do not lead to a performance increase compared to the performance using solely the backbone architecture.

While we can bypass the oversampling issue by using the density-based algorithms (indicated by the * in Table 4), none of the used clustering algorithms assigns clusters in a sufficiently accurate way such that the performance of the backbone CNN gets improved.

3.2. Batch Construction Based on Nearest Neighbors

As another option, we sample chunks by randomly choosing one *anchor* and finding its $k-1$ nearest neighbors. Then we construct batches consisting of n of these chunks and feed them through the MPN. In every batch, we only update the feature vectors of the anchors, meaning that we build

	CUB-200-2011		CARS196	
	R@1	**NMI**	**R@1**	**NMI**
backbone only	70.3	74.0	88.1	74.8
K-means†	66.1	69.6	85.1	70.5
Ward clustering†	67.9	68.8	86.6	69.5
Spectral Clustering†	63.1	69.6	85.1	69.2
Birch†	65.5	69.5	86.1	70.0
DBSCAN*	65.7	72.0	85.0	70.3
Optics*	66.5	71.1	85.4	69.6
Nearest neighbors	62.1	69.2	84.8	70.2
Reciprocal kNN	**70.8**	**74.5**	**88.6**	**76.2**
Knowledge Distillation	65.7	70.3	85.6	71.6
Feature Imitation	65.3	70.1	85.6	70.9
Relational TS	64.9	68.7	84.7	70.5

Table 4. Performance of different settings of using the MPN during test time as well as the performance of teacher student approaches. † indicates clustering algorithms that need a fixed number of clusters (900 clusters), * indicates density-based clustering algorithms (eps=0.9, min sample=5)

one chunk for each image in the test set. As these chunks again can be highly noisy, the performance after the MPN drops compared to simply taking the embeddings from the backbone (see ”nearest neighbors” in Table 4).

3.3. Reciprocal-kNN Batch Construction

Since imitating the training batch construction during test time and simply using a sample’s $k-1$ nearest neighbors does not lead to a performance increase, we propose to construct batches more strictly during test time. To that end, we suggest constructing reciprocal k-nearest neighbor batches inspired by (Zhong et al., 2017). Different from (Zhong et al., 2017) who use reciprocal k-nearest neighbor for evaluation, we use a similar idea for batch construction (we still do the final evaluation regularly, by simply evaluating Recall@K and NMI). Knowing that samples that are highly similar to the query sample are more likely to be of the same class as the query image c_q than dissimilar samples, we first compute the k-nearest neighbor set N_q^k of a given query q (see the upper part in Figure 3.3). However, N_q^k still might contain noisy samples. Therefore, we reduce N_q^k to a reciprocal k-nn set $N_{r,q}^k$ by only taking samples $g \in N_q^k$ into account that also contain q in their own k-nn set N_g^k (indicated by the green frames in the middle part of Figure 3.3). Up to this step $N_{r,q}^k$ only contains samples that are already highly similar to the query image. Some gallery samples g of class c_q might not be directly contained in $N_{r,q}^k$, but in $N_{r,g}^k$ of some samples $g \in N_q^r, k$. Therefore, we expand $N_{r,q}^k$ to $\tilde{N}_{r,q}^k$ by the reciprocal $\frac{1}{2}k$-nn set $N_{r,g}^{0.5k}$ of the samples $g \in N_{r,q}^k$, if the following holds:

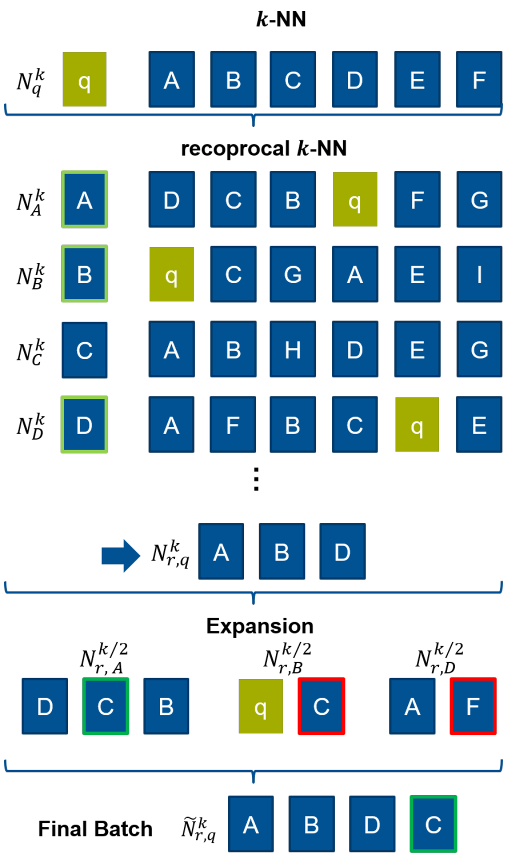

Figure 1. Reciprocal k-nearest neighbor batch sampling for MPN during inference.

$$|N_{r,q}^k \cap N_{r,g}^{0.5k}| \geq \alpha |N_{r,g}^{0.5k}| \quad (1)$$

where $\alpha \in [0,1]$, $|\tilde{N}_{r,q}^k| = k_r$ and k_r is a constant. In the expansion step of Figure 3.3, samples that fulfill the above mentioned constraint for $\alpha = \frac{2}{3}$ are visualized by a green frame, those who do not by a red frame. If $|\tilde{N}_{r,q}^k| < k_r$, we add the closest samples to q that are not yet contained in $\tilde{N}_{r,q}^k$. Finally, we feed $\tilde{N}_{r,q}^k$ into the Message Passing Network to refine the feature vector of q. As we have shown (see Tab. 3 in the main paper) this approach improved the performance during test time by $0.5pp$ Recall@1 and $0.5pp$ NMI on CUB-200-2011 dataset and $0.5pp$ (Recall@1) and $1.4pp$ (NMI) on the Cars196 dataset.

3.4. Teacher Student Approach

As the latter approach requires the usage of additional parameters during test time, we develop several teacher-student approaches to transfer the knowledge of the MPN, acting as a teacher, to the backbone CNN, acting as a student.

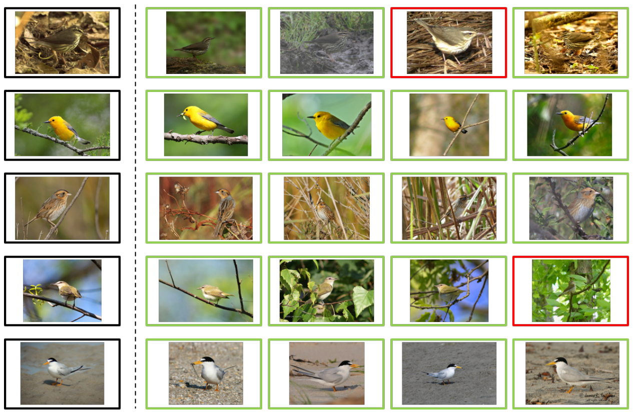
Figure 2. More qualitative results on CUB-200-2011

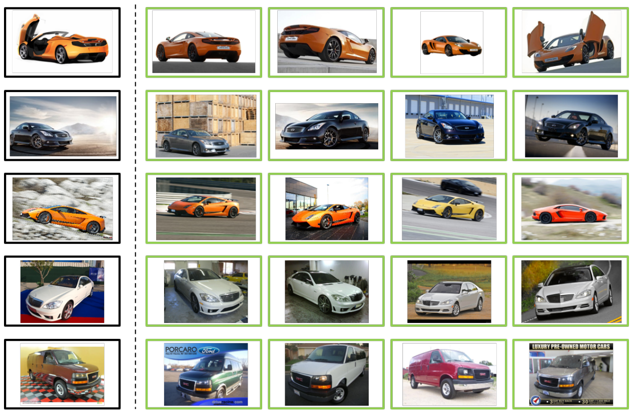
Figure 3. More qualitative results on Cars196

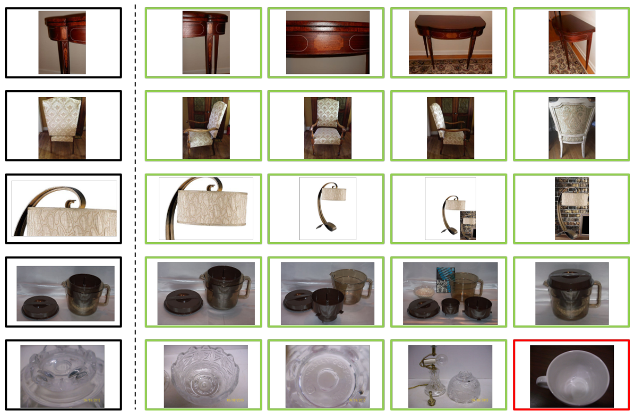
Figure 4. More qualitative results on Stanford Online Products

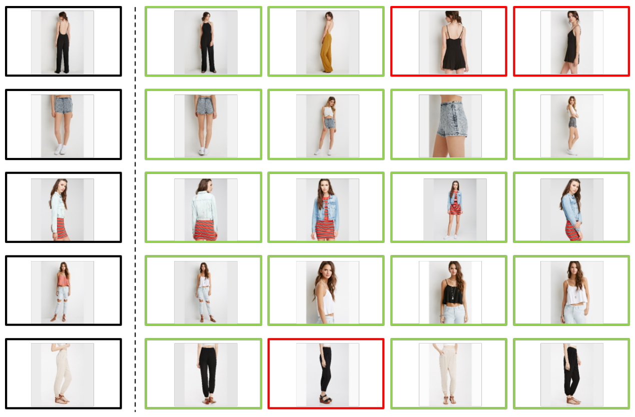
Figure 5. More qualitative results on In-Shop

Knowledge Distillation. As our approach is based on the cross-entropy loss function, we first use knowledge distillation (Hinton et al., 2015), where the class probability distributions of the teacher are imitated by the student. The advantage of this technique is that these probability distributions also contain information about the similarities between classes. However, the usage of this approach does not increase the performance compared to solely using the backbone CNN, but decreases it by $4.6pp$ Recall@1 on CUB-200-2011 and $2.5pp$ Recall@1 on Cars196 (see Table 4)

Feature Imitation. Since we are not directly interested in the class prediction quality of our network, but in the feature vectors themselves, our second approach forces the student to directly imitate the feature vectors. Further, those feature vectors of the training data after the MPN are highly discriminative. We apply the Huber Loss which is less sensitive to outliers. Again, the performance drops by $5pp$ Recall@1 on CUB-200-2011 and $2.5pp$ Recall@1 on Cars196 compared to solely using the backbone CNN.

Relational Teacher Student. Lastly, we utilize relational knowledge distillation (Park et al., 2019), where the student does not directly imitate the feature vectors, but the relations between different feature vectors. This also supports the paradigm of our MPN-based approach, where we refine feature vectors by taking the relations between samples into account. The performance, though, does not increase but drops even more by $5.4pp$ Recall@1 on CUB-200-2011 and $3.4pp$ Recall@1 on Cars196.

4. Qualitative Results

In the main work, we already showed several examples for our qualitative results. To show the robustness of our approach, we will now show several more samples of qualitative results on CUB-200-2011 (Figure 2), Cars196 (Figure 3), Stanford Online Products (Figure 4) and In-Shop Clothes (Figure 5). As can be seen, our approach is able to retrieve images of the same class even for harder examples, like in the first example of CUB-200-2011 (Figure 2).

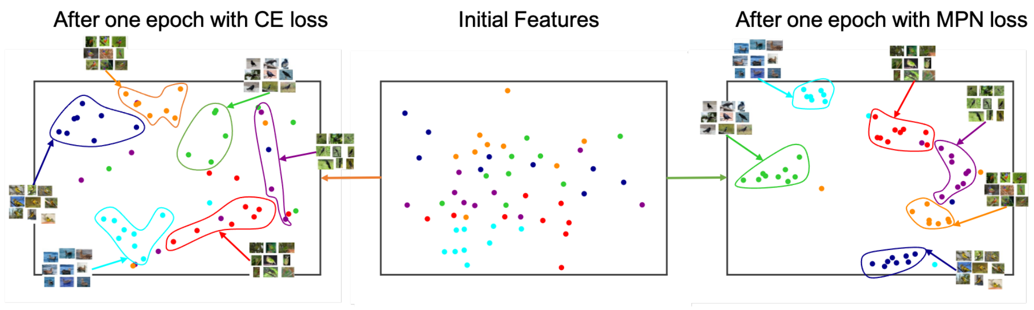

Figure 6. Comparison of the embeddings of a given batch after one epoch of training without and with MPN.

5. T-SNE

We now show a visualization of the embedding space obtained by our approach on *CUB-200-2011* using the t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten & Hinton, 2012) in Figure 12. Every scatter point represents a sample and different colors represent different classes. As can be seen, our approach is able to achieve representative clusters of many classes. We highlight several groups of samples, that can be best viewed when zoomed in.

6. MPN Matters 2.0 - Comparison of Performance of Training with and without MPN

In addition to the ablation studies in the main paper where we showed the performance increase of the backbone CNN trained with MPN over not using the MPN but solely cross-entropy loss during training (see MPN matters) as well as some visualizations of how the class prediction after the MPN is influenced by other samples in the batch, we provide some more visualizations that support these findings.

Firstly, we visualize the difference between the embeddings after the backbone CNN of a given batch after one epoch of training without and with MPN using t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten & Hinton, 2012). As can be seen in Figure 6, the features after the backbone CNN are much more clustered than when training without MPN.

Secondly, we again utilize t-SNE (van der Maaten & Hinton, 2012) to get low dimensional representations of the embeddings of all samples in a given test set after the whole training without as well as with MPN and sample 10 classes for the sake of clarity. In Figure 7 and Figure 8 we show three such subsets of classes for CUB-200-2011 and Cars196, respectively. The upper rows in both figures represent embeddings generated by the backbone CNN trained without MPN while the lower ones show embeddings generated by the backbone trained with MPN. The backbone CNN

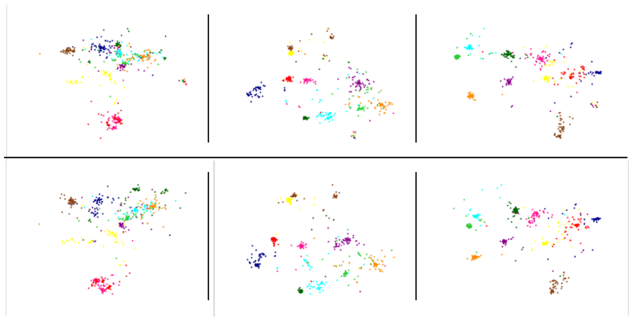

Figure 7. Visualization of 10 sampled classes from CUB-200-2011 test dataset when trained without MPN (upper row) and with MPN (lower row).

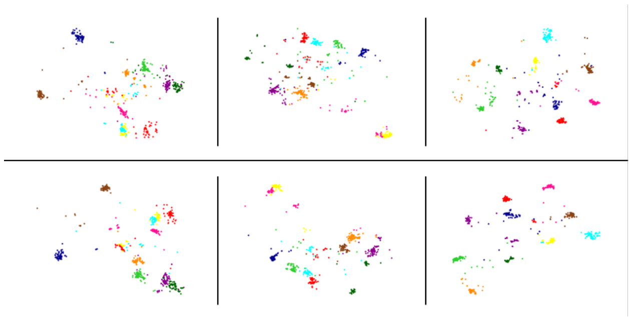

Figure 8. Visualization of 10 sampled classes from Cars196 test dataset when trained without MPN (upper row) and with MPN (lower row).

trained using solely cross-entropy loss performs well on many samples. However, our approach is able to better divide more difficult classes from the remaining classes in the embedding space as can be seen for example from the dark blue class in the first column of Figure 7. Further, it is less prone to outliers as can be seen in the second column in Figure 8, where there are fewer outliers in all classes in the lower row that shows the visualizations of the embeddings trained with MPN. Finally, the embeddings of samples of the same class most often lie closer together and are further

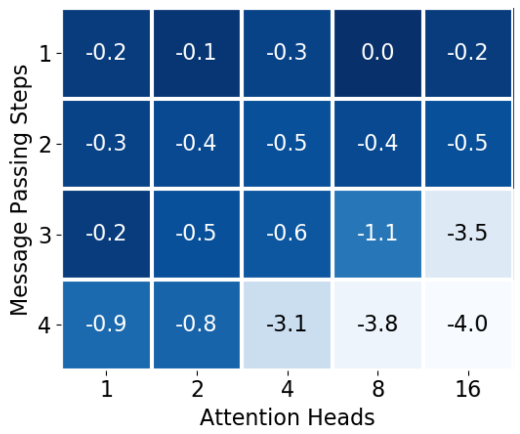

Figure 9. Relative Difference with respect to Best Recall@1 on Stanford Online Products.

apart from other classes as can be seen from the red and pink classes in the central column of Figure 7 or the dark blue, orange, and pink classes in the first column of Figure 8.

7. Analysis of Number of Message Passing Steps and Heads on Stanford Online Products

Additionally to the analysis of different numbers of message passing steps and attention heads on CUB-200-2011 and Cars196 in the main paper, we provide an equal analysis on Stanford Online Products. Investigating the results on CUB-200-2011 and Cars196 datasets one could assume that with increasing size of the dataset an increasing number of message passing steps is needed, as the best performing model on CUB-200-2011 utilizes only one message passing step while the best performing model on Cars196 utilizes two message passing steps and CUB-200-2011 is smaller than Cars196. However, as can be seen in Figure 9 this is not the case, and the performance of our approach on Stanford Online Products drops with an increasing number of message passing steps and attention heads. As already mentioned in the main paper, this is in line with (Velickovic et al., 2018), who also utilize few message passing steps when applying graph attention.

8. Different Number of Classes

We also conduct experiments to investigate the impact of the composition of the batches concerning the number of classes and the number of samples per class. Therefore we vary the number of classes and samples on CUB-200-2011 and Cars196 between five and ten. As can be seen in Figure 10, the performance on CUB-200-2011 tends

	CUB	CARS	SOP	In-Shop
Ours	**70.3**	**88.1**	**81.4**	**92.8**
Ours reality check	67.1±0.69	86.7±0.48	81.1±0.13	92.5±0.11

Table 5. Performance of our approach following (Musgrave et al., 2020) to find the hyperparameters (e.g., number of epochs) without feedback from the test set.

to go down with an increasing number of classes while on Cars196 (see Figure 11) the performance drops when only a few samples per class are used. However, it can be said that the performance is stable with the biggest drop in performance on CUB-200-2011 being $2.8pp$ and $3.4pp$ on Cars196.

9. Metric Learning Reality Check

(Musgrave et al., 2020) claim that the huge improvements of recent metric learning approaches over prior works is mainly caused by flaws in the experimental methodology like utilizing a more powerful backbone, unsuitable evaluation metrics or training with test set feedback. The authors show, that the performance of ResNet50 is worse on CUB-200-2011 and Cars196 than when using BN-Inception. To prove that our approach is robust to the hyperparameter choice we followed (Musgrave et al., 2020) to find the hyperparameters (e.g., number of epochs) without feedback from the test set and report the results here. We get state-of-the-art results in Cars196, Stanford Online Products, and In-Shop datasets, and competitive results on CUB-200-2011 dataset compared to results mentioned in (Musgrave et al., 2020) as can be seen in Table 5.

10. Large Number of Classes

Our method uses a fully connected layer to compute the final loss function. In cases where the number of classes increases, then the size of the classification layer increases too. While this typically is not a problem for metric learning datasets which contain from hundreds to tens of thousands of classes, it can become a problem for the closely related problem of face recognition, where datasets typically contain millions of classes. Unlike our method (and other classification-based methods (Zhai & Wu, 2019; Zheng et al., 2019; Qian et al., 2019; Elezi et al., 2020)), methods that use a pairwise (e.g. contrastive/triplet) loss function do not have this problem.

Nevertheless, there are ways of facing the problem. Normalized Softmax (Zhai & Wu, 2019) tackles the problem by sampling a mini-batch only from a certain number of classes, a strategy proposed also in Group Loss (Elezi et al., 2020). Our method uses this sampling strategy in default mode (in each mini-batch, we sample only a certain number of classes). Consequently, we know in advance which units

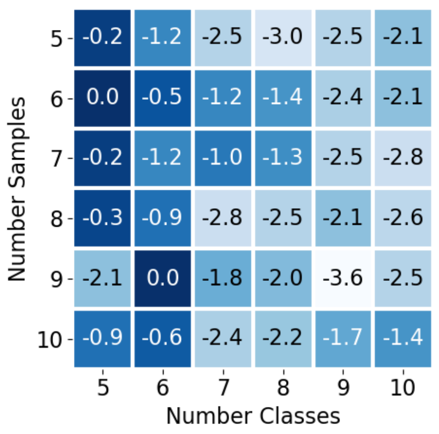

Figure 10. Relative Difference with respect to Best Recall@1 on CUB-200-2011.

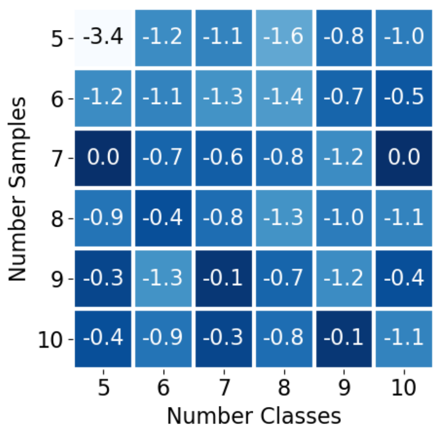

Figure 11. Relative Difference with respect to Best Recall@1 on Cars196.

of the last layer need to be modified, and all the other units can easily get masked out for a more efficient tensor-tensor multiplication.

Dealing with datasets that contain a large number of classes is a problem that has been widely studied in natural language processing (Mikolov et al., 2013), typically solved by replacing the softmax layer with hierarchical-softmax (Mnih & Hinton, 2008). Considering that the problem is similar, we could envision replacing softmax with hierarchical-softmax for our problem to have a more efficient method.

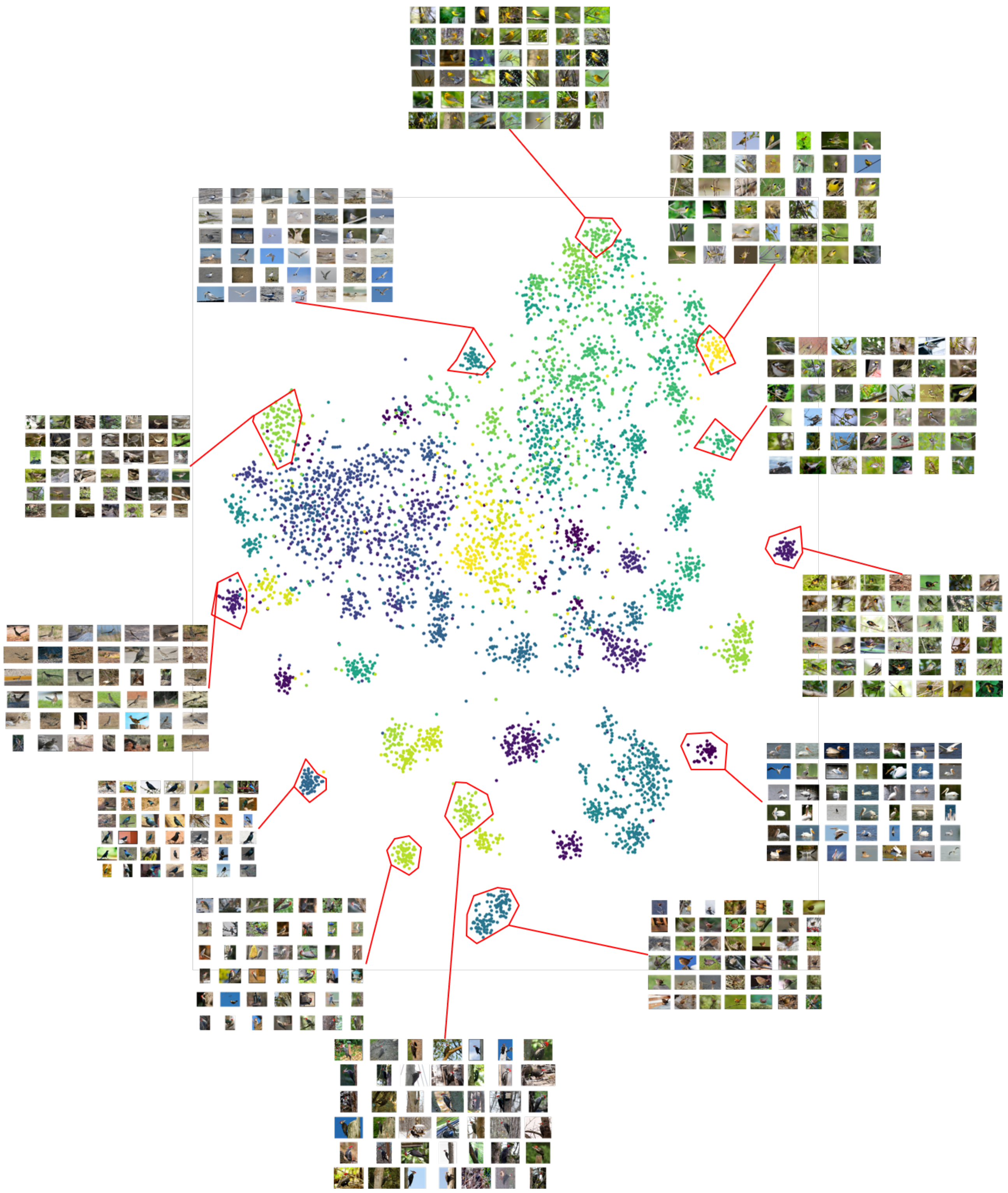

Figure 12. t-SNE (van der Maaten & Hinton, 2012) visualization of our embeddings on the CUB-200-2011 (Wah et al., 2011) dataset with some clusters highlighted. Best viewed on a monitor when zoomed in.

References

Elezi, I., Vascon, S., Torchinovich, A., Pelillo, M., and Leal-Taixé, L. The group loss for deep metric learning. In *European Conference in Computer Vision (ECCV)*, 2020.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, 2015.

Jacob, P., Picard, D., Histace, A., and Klein, E. Metric learning with HORDE: high-order regularizer for deep embeddings. In *International Conference on Computer Vision (ICCV)*, 2019.

Kim, S., Kim, D., Cho, M., and Kwak, S. Proxy anchor loss for deep metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations, ICLR*, 2013.

Mnih, A. and Hinton, G. E. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Musgrave, K., Belongie, S. J., and Lim, S. A metric learning reality check. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, volume 12370 of *Lecture Notes in Computer Science*, pp. 681–699, 2020.

Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3967–3976. Computer Vision Foundation / IEEE, 2019.

Qian, Q., Shang, L., Sun, B., Hu, J., Tacoma, T., Li, H., and Jin, R. Softtriple loss: Deep metric learning without triplet sampling. In *International Conference on Computer Vision (ICCV)*, 2019.

Teh, E. W., DeVries, T., and Taylor, G. W. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*, 2020.

van der Maaten, L. and Hinton, G. E. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1): 33–55, 2012.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. Multi-similarity loss with general pair weighting for deep metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zhai, A. and Wu, H. Classification is a strong baseline for deep metric learning. In *British Machine Vision Conference (BMVC)*, 2019.

Zheng, X., Ji, R., Sun, X., Zhang, B., Wu, Y., and Huang, F. Towards optimal fine grained retrieval via decorrelated centralized loss with normalize-scale layer. In *Conference on Artificial Intelligence (AAAI)*, 2019.

Zhong, Z., Zheng, L., Cao, D., and Li, S. Re-ranking person re-identification with k-reciprocal encoding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.