Supplementary Information - ProxyNCA++: Revisiting and Revitalizing Proxy Neighborhood Component Analysis

Eu Wern Teh[1,2], Terrance DeVries[1,2], and Graham W. Taylor[1,2]

[1] University of Guelph, ON, Canada
[2] Vector Institute, ON, Canada
{eteh,terrance,gwtaylor}@uoguelph.ca

1 A comparison with NormSoftMax [7]

In this section, we compare the differences between ProxyNCA++ and NormSoftMax [7]. Both ProxyNCA++ and NormSoftMax are proxy-based DML solutions. By borrowing notations from Equation 6 in the main paper, ProxyNCA++ has the following loss function:

$$L_{\text{ProxyNCA++}} = -\log\left(\frac{\exp\left(-d(\frac{x_i}{||x_i||_2}, \frac{f(x_i)}{||f(x_i)||_2}) * \frac{1}{T}\right)}{\sum_{f(a)\in A}\exp\left(-d(\frac{x_i}{||x_i||_2}, \frac{f(a)}{||f(a)||_2}) * \frac{1}{T}\right)}\right) \tag{1}$$

And NormSoftMax has the following loss function:

$$L_{\text{NormSoftMax}} = -\log\left(\frac{\exp\left(\frac{x_i}{||x_i||_2}^\top \frac{f(x_i)}{||f(x_i)||_2} * \frac{1}{T}\right)}{\sum_{f(a)\in A}\exp\left(\frac{x_i}{||x_i||_2}^\top \frac{f(a)}{||f(a)||_2} * \frac{1}{T}\right)}\right) \tag{2}$$

The main difference between Equation 1 and 2 is the distance function. In ProxyNCA++, we use a euclidean squared distance function instead of cosine distance function.

Based on our sensitivity studies on temperature scaling and proxy learning rate, we show that NormSoftMax perform best when the temperature scale, T is set to $1/2$ and the proxy learning rate is set to $4e^{-1}$ (see Figure 1 and 2).

We perform an ablation study of NormSoftMax in Table 1 and 2. On the CUB200 dataset, we show that the Global Max Pooling (GMP) component (max) improves NormSoftMax by 2.2pp on R@1. However, the fast proxies component (fast) reacts negatively with NormSoftMax by decreasing its performance by 0.6pp. By combining both the GMP and the fast proxies components into NormSoftMax, we see a small increase of R@1 performance (0.6pp).

On the CARS196 dataset, there is a slight increase in R@1 performance (0.3pp) by adding fast proxies component to NormSoftMax. When we add the GMP component to NormSoftMax, we observe an increase of R@1 by 1.1pp.

Fig. 1. A sensitivity study of temperature scaling for NormSoftMax [7] without layer norm (norm), class balanced sampling (cbs), and fast proxies (fast). We show a plot of R@1 with different temperature scales on CUB200 [5]. The shaded areas represent one standard deviation of uncertainty.

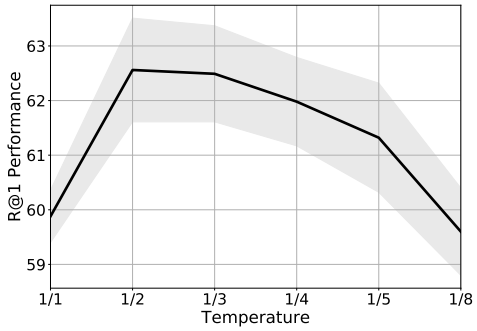


Fig. 2. A sensitivity study of proxy learning rate for NormSoftMax [7] without layer norm (norm), class balanced sampling (cbs) and with temperature scaling (scale) $T = 1/2$. We show a plots of R@1 with different proxy learning rates on CUB200 [5]. The shaded areas represent one standard deviation of uncertainty.

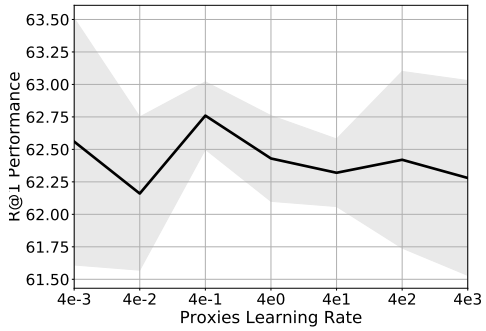


Table 1. A comparison of ProxyNCA++ and NormSoftMax [7] on CUB200 [5]. All models are experimented with embedding size of 2048. For NormSoftMax [8], we use a temperature scaling of $T = 1/2$, a proxy learning rate of $4e^{-1}$ (fast) and learning rates of $4e-3$ for the backbone and embedding layers. It is important to note that, NormSoftMax [7] does not have max pooling and fast proxy component.

R@k	1	2	4	8	NMI
ProxyNCA++	72.2±0.8	82.0±0.6	89.2±0.6	93.5±0.4	75.8±0.8
-max	69.0±0.6	80.3±0.5	88.1±0.4	93.1±0.1	74.3±0.4
-fast	70.3±0.9	80.6±0.4	87.7±0.5	92.5±0.3	73.5±0.9
-max -fast	69.1±0.5	79.6±0.4	87.3±0.3	92.7±0.2	73.3±0.7
NormSoftMax (+max, +fast)	65.0±1.7	76.6±1.1	85.5±0.6	91.6±0.4	69.6±0.8
NormSoftMax (+fast)	63.8±1.3	75.9±1.0	84.9±0.8	91.4±0.6	70.8±1.1
NormSoftMax (+max)	67.6±0.4	78.4±0.2	86.7±0.4	92.2±0.3	71.2±0.9
NormSoftMax	64.4±1.1	76.1±0.7	85.0±0.7	91.4±0.3	70.0±1.1

Combining both the GMP and the fast proxies components, there is a 1.0pp increase in R@1 performance.

Table 2. A comparison of ProxyNCA++ and NormSoftMax [7] on CARS196 [2]. All models are experimented with embedding size of 2048. For NormSoftMax [8], we use a temperature scaling of $T = 1/2$, a proxy learning rate of $4e^{-1}$ (fast) and learning rates of $4e-3$ for the backbone and embedding layers. It is important to note that, NormSoftMax [7] does not have max pooling and fast proxy component.

R@k	1	2	4	8	NMI
ProxyNCA++	90.1±0.2	94.5±0.2	97.0±0.2	98.4±0.1	76.6±0.7
-max	87.8±0.6	93.2±0.4	96.3±0.2	98.0±0.1	76.4±1.3
-fast	89.2±0.4	93.9±0.2	96.5±0.1	98.0±0.1	74.8±0.7
-max -fast	87.9±0.2	93.2±0.2	96.1±0.2	97.9±0.1	76.0±0.5
NormSoftMax (+max, +fast)	86.0±0.1	92.0±0.1	95.5±0.1	97.6±0.1	68.6±0.6
NormSoftMax (+fast)	85.3±0.4	91.6±0.3	95.5±0.2	97.6±0.1	72.2±0.7
NormSoftMax (+max)	86.1±0.4	92.1±0.3	95.5±0.2	97.6±0.2	68.0±0.5
NormSoftMax	85.0±0.6	91.4±0.5	95.3±0.4	97.5±0.3	70.7±1.1

2 Two moon classifier

In Section 3.4 (About Temperature Scaling) in the main paper, we show a visualization of the effect of temperature scaling on the decision boundary of a softmax classifier on a two-moon synthetic dataset. In detail, we trained a two-layers linear model. The first layer has an input size of 2 and an output size of 100. This is followed by a ReLU unit. The second layer has an input size of 100 and an output size of 2. For the synthetic dataset, we use the scikit-learn's [1] moons data generator to generate 600 samples with noise of 0.3 and a random state of 0.

3 Regarding crop size of images

Image crop size can have a large influence on performance. Current SOTA method [1] for embedding size 512 uses a crop size of 256×256, which we also use for our experiments (See Table 3). We repeat these experiments with a crop-size of 227×227 to make it comparable with older SOTA method [6] (See Table 4). In this setting, we outperform SOTA for CARS and SOP. We tie on CUB, and we underperform on InShop. However, since no spread information is reported in SOTA [6], it is hard to make a direct comparison.

Table 3. A comparison of ProxyNCA++ and the current SOTA [1] in the embedding size of 512 and a crop size of 256×256.

R@k	SOTA [1]	Ours
CUB	66.8	69.0±0.8
CARS	86.2	86.5±0.4
SOP	80.1	80.7±0.5
InShop	90.4	90.4±0.2

Table 4. A comparison of ProxyNCA++ and the current SOTA [6] in the embedding size of 512 and a crop size of 227×227.

R@k	SOTA [6]	Ours
CUB	65.7	64.7±1.6
CARS	84.2	85.1±0.3
SOP	78.2	79.6±0.6
InShop	89.7	87.6±1.0

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

4 Regarding the implementation of baseline

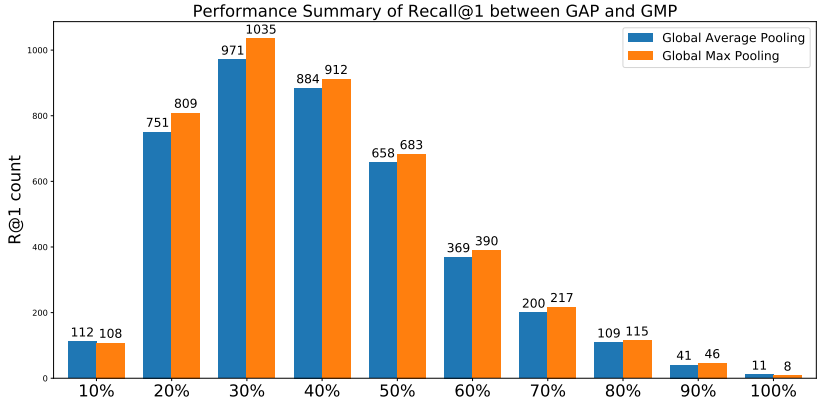
We follow Algorithm 1 in the original paper [3] when implementing our baseline. In the original paper, there is an α variable that resembles temperature scaling. However, α choice is ambiguous and is used to prove the error bound theoretically. We replicate [3] on CUB, with embedding size of 64, crop size of 227, and GoogLeNet [4] backbone. With the temperature scale, T=1/3, we obtain a R@1 of 49.70, which is close to the reported R@1 49.2. This indicates our implementation of ProxyNCA is correct.

The baseline ProxyNCA is implemented using the same training set up as the proposed ProxyNCA++. As mentioned in our paper (Sec 4.1), we split the original training set into the training (1st half) and validation set (2nd half). We did not perform an extensive sweep of hyperparameters. In our experiment, we first select the best hyperparameter for baseline ProxyNCA (i.e., learning rate [1e-3 to 5e-3]) before adding any enhancements corresponding to ProxyNCA++. We believe that it is possible to obtain better results for both ProxyNCA and ProxyNCA++ with a more extensive sweep of hyperparameters.

5 Regarding the Global Max Pooling (GMP) vs. Global Average Pooling (GAP)

In our paper, we show that GMP is better than GAP empirically. However, we could not find any consistent visual evidence as to why GMP works better. We initially hypothesized that GAP was failing for small objects. But after we controlled for object size, we did not observe any consistent visual evidence to support this hypothesis. In Figure 3, GMP consistently outperform GAP regardless of object size; this evidence disproved our initial hypothesis.

Fig. 3. Performance summary (R@1) between GMP and GAP on various object sizes (in percent w.r.t. image size) in the CUB dataset.



6 Regarding the computation complexity of ProxyNCA++

The inference time to compute embeddings for ProxyNCA++ and baseline ProxyNCA will depend on the base architecture. In our experiments, we used a ResNet-50 model as a backbone, so inference time would be comparable to that of a ResNet-50 classifier. There are two differences which have a negligible effect on inference time: (a) The removal of the softmax classification layer, and (b) the addition of layer norm.

As for training time complexity, ProxyNCA++ is comparable to ProxyNCA both theoretically and in terms of runtime. Given a training batch size of B, we only need to compute the distance between each sample w.r.t the proxies, K. After that, we compute a cross-entropy of these distances, where we minimize the probability of a sample being assigned to its own proxy. Therefore the runtime complexity in a given batch is O(BK).

References

1. Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. *arXiv preprint arXiv:1908.02735*, 2019. 4
2. Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 3
3. Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017. 5
4. C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. 5
5. Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 3
6. Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 4
7. Andrew Zhai, Hao-Yu Wu, and US San Francisco. Classification is a strong baseline for deep metric learning. 2019. 1, 2, 3
8. Feng Zheng, Cheng Deng, Xing Sun, Xinyang Jiang, Xiaowei Guo, Zongqiao Yu, Feiyue Huang, and Rongrong Ji. Pyramidal person re-identification via multi-loss dynamic training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3