Proceedings of Machine Learning Research 222, 2023 ACML 2023

Supplementary Material: Advancing Deep Metric Learning With Adversarial Robustness

Inderjeet Singh INDERJEET78@NEC.COM
Kazuya Kakizaki KAZUYA1210@NEC.COM
Toshinori Araki TOSHINORI_ARAKI@NEC.COM
NEC Corporation, Kawasaki, Kanagawa, Japan

Editors: Berrin Yanıkoğlu and Wray Buntine

1. Evaluation Metrics

We use the standard evaluation metrics in deep metric learning (DML): Recall@K (R@K) (Jegou et al. (2010)) with $k = \{1, 4\}$, Normalized Mutual Information (NMI) (Manning et al. (2010)), and π_{ratio}. Increased $R@k$ and NMI values indicate improved image retrieval performance and clustering quality, respectively, and the decreased π_{ratio} values approximately indicate increased inter-class and decreased intra-class distances in the embedding space of the trained model.

1.1. Recall@k (Jegou et al. (2010))

For a given DML function f, let $\mathcal{F}_q^k$ be the set of first k nearest neighbors of a sample $x_q \in \mathcal{X}_{test}$ defined as

$$\mathcal{F}_q^k = \underset{\mathcal{F} \subset \mathcal{X}_{\text{test}}\ , |\mathcal{F}|=k}{\arg\min} \sum_{x_n \in \mathcal{F}} d\left(f\left(x_q\right), f\left(x_n\right)\right) \tag{1}$$

Finally, Recall@k is calculated as

$$R@k = \frac{1}{|\mathcal{X}_{\text{test}}\ |} \sum_{x_q \in \mathcal{X}_{\text{test}}} \begin{cases} 1 & \exists x_i \in \mathcal{F}_q^k \text{ s.t. } y_i = y_q \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This means Recall@k measures the average number of cases in which, for a given query x_q, there is at least one sample among its top k nearest neighbors x_i with the same class, i.e., $y_i = y_q$.

1.2. Normalized Mutual Information (NMI) (Manning et al. (2010))

NMI quantifies the clustering quality in the embedding space of a DML model f. To calculate NMI for the embedding space $\Phi_{X_{test}}$ of all test samples $x_i \in \mathcal{X}_{test}$, we assign a cluster label w_i corresponding to each sample x_i indicating the closest cluster center and define $\Omega = \{\omega_k\}_{k=1}^K$ with $\omega_k = \{i|w_i = k\}$ and $K = |\mathcal{C}|$ being the number of classes and clusters. Similarly for the true labels y_i we define $\Upsilon = \{\upsilon_c\}_{c=1}^K$ with $\upsilon_c = \{i|y_i = c\}$.

© 2023 I. Singh, K. Kakizaki & T. Araki.

The NMI is then computed with mutual Information $I(\cdot,\cdot)$ between cluster and labels, and entropy $H(\cdot,\cdot)$ on the clusters and labels, respectively, as

$$NMI(\Omega,\Upsilon)=\frac{I(\Omega,\Upsilon)}{2(H(\Omega)+H(\Upsilon))} \tag{3}$$

1.3. Embedding Space Density (π_{ratio})

We define embedding space density π_{ratio} as

$$\pi_{ratio}(\Phi)=\frac{\pi_{intra}(\Phi)}{\pi_{inter}(\Phi)} \tag{4}$$

where $\pi_{intra}(\Phi)$ is the intra class distance and $\pi_{inter}(\Phi)$ inter class distance in the feature space $\Phi_{\mathcal{X}}:=\{f_\theta(x)\mid x\in\mathcal{X}_{test}\}$ of a DML model f_θ and they are calculated as follows:

$$\pi_{intra}(\Phi)=\frac{1}{Z_{\text{intra}}}\sum_{y_l\in\mathcal{Y}}\sum_{\phi_i,\phi_j\in\Phi_{y_l},i\neq j} d\left(\phi_i,\phi_j\right) \tag{5}$$

$$\pi_{inter}(\Phi)=\frac{1}{Z_{\text{inter}}}\sum_{y_l,y_k,l\neq k} d\left(\mu\left(\Phi_{y_l}\right),\mu\left(\Phi_{y_k}\right)\right) \tag{6}$$

Here, $\Phi_{y_l}=\{\phi_i:=f_\theta\left(x_i\right)\mid x_i\in\mathbb{X},y_i=y_l\}$ denotes the set of embedded samples of a class y_l. $\mu(\Phi_{y_l})$ their mean embedding and Z_{intra}, Z_{inter} are the normalization constants.

2. Benchmarks

We evaluate the performance on the CUB200 (Wah et al. (2011)), CARS 196 (Krause et al. (2013)), and Stanford Online Products (Oh Song et al. (2016)) benchmarks following the experimental setting by Roth et al. (2021) for data pre-processing.
CUB200 (Wah et al. (2011)) contains 200 bird classes over 11,788 images, whereas the first and last 100 classes with 5864/5924 images are used for training and testing, respectively.
CARS196 (Krause et al. (2013)) contains 196 car classes and 16,185 images, where again, the first and last 98 classes with 8054/8131 images are used to create the training/testing split.
Stanford Online Products (SOP) (Oh Song et al. (2016)) is built around 22,634 product classes over 120,053 images and contains a provided split: 11318 selected classes with 59551 images are used for training, and 11316 classes with 60502 images for testing.

3. Complete Experimental Setup

We provide comprehensive experimental details for the evaluation of MDProp in order to ensure reproducibility. To maintain consistency with Roth et al. (2021), we follow their setup with the exception of frozen batch normalization (Ioffe and Szegedy (2015)). Frozen batch normalization is exclusively used for baselines in order to replicate prior results and conduct a fair comparison with state-of-the-art methods. Our experiments employ ResNet18, ResNet50, and ResNet152 architectures (He et al. (2016)) with output embeddings normalized with 128 dimensions and optimization with Adam (Kingma and Ba (2014)) using a learning rate of 10^{-5} and weight decay of $4 \cdot 10^{-4}$. Training images were randomly resized and cropped to 224×224 pixels, with further augmentation through random horizontal flipping with $p = 0.5$. During testing, center crops of size 224×224 were used, with a batch size of 112. All experiments were conducted on CUB200 and CARS 196 for 150 epochs, and on SOP for 100 epochs, without any learning rate schedule.

Additionally, we implemented S2SD (Roth et al. (2021)) with ResNet50 architecture and Multisimilarity loss (Wang et al. (2019)), keeping all other hyperparameters in line with the defaults outlined in Tab. 1 of Roth et al. (2020). The implementation was conducted in PyTorch (Paszke et al. (2019)), on GPU servers consisting of Nvidia Tesla V100, Titan V, and RTX 1080Ti, with data parallelization and distributed training. Results presented in Tab. 1 of the paper are averaged over three seeds, while those in Tab. 2 are the average of two seeds. We report means and standard deviations for better reproducibility and validity.

We generated single and multi-targeted adversarial examples (MTAXs) during training through projected gradient descent (PGD) update (Madry et al. (2017)), setting the number of iterations to 1, L_∞ constraint ϵ on the adversarial noise to 0.01, and the PGD learning rate as ϵ/Attack Iterations. Four values of T (i.e., 2, 3, 5, and 10) were used as attack targets during the MTAX generation. We kept the loss function as squared L_2 norm for generating feature space AXs. To assess the robustness of the DML models trained with MDProp against multiple input distributions during inference, we generated single and multi-targeted AXs.

Unadversarial Example Generation Setting. In generating unadversarial examples during training with MDProp, we used the same update settings as PGD, with $\epsilon = 0.01$, but with a modified loss function for unadversarial examples.

Few-Shot Learning Experiments. We evaluate the performance of our method in the few-shot evaluation setting, where we aim to classify images with few labeled examples. To generate training subsets for this evaluation, we uniformly removed samples from all classes of the dataset, creating three subsets with different fractions of the original training data. Specifically, we generated subsets with 0.25, 0.50, and 0.75 fractions of the original training data. This few-shot evaluation setting allows us to assess the generalization ability of our method in scenarios where data is limited, and classify images with fewer labeled examples.

3.1. DML Loss Functions

3.1.1. Multisimilarity (Wang et al. (2019))

Multisimilarity loss (Wang et al. (2019)) uses the concept of different types of similarities in all positive and negative samples for an anchor x_i in training data while using hard sample mining:

$$d_c^*(i,j) = \begin{cases} d_c\left(\psi_i, \psi_j\right) & d_c\left(\psi_i, \psi_j\right) > \min_{j \in \mathcal{P}_i} d_c\left(\psi_i, \psi_j\right) - \epsilon \\ d_c\left(\psi_i, \psi_j\right) & d_c\left(\psi_i, \psi_j\right) < \max_{k \in \mathcal{N}_i} d_c\left(\psi_i, \psi_k\right) + \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

We used additive angular margin penalty $\gamma = 0.5$. The radius of the effectively utilized hypersphere $\mathbb{S}$ denoted as the scaling $s = 16$ was used. The class centers were optimized with a learning rate of 0.0005.

3.2. Adversarial Training with Targeted Attacks

It is well known that adversarial training results in highly robust models, but causes a reduction in the clean data performance of the model. In this study, our primary focus is to improve the accuracy of clean data using AXs in the form of multi-distribution inputs. Hence, to make the comparison fair and effectively evaluate the effect of separate BN layers, we used both clean and adversarial data during training without using separate BN layers. For generating adversarial data, we use the same single targeted AXs x^t_{adv} used in the AdvProp-D case of MDProp, which are generated as

$$x^f_{adv} = x^j_i + \delta^t_f \quad s.t. \quad \delta^t_f = \underset{||\delta||_\infty \leq \epsilon}{\arg\min} \left[\mathcal{L}(f(x^j_i + \delta), f(x^k_i)) \right] \tag{10}$$

where $\mathcal{L}$ measures the distance, f is the DML model, and x^k_i is the target identity's image.

Finally, the objective of the adversarial training in our setting is as follows:

$$\mathcal{Z}_1 = \underset{\theta}{\arg\min} \left[\mathbb{E}_{\left\{ \substack{(x,y)\sim\mathbb{D} \\ \delta^t_f \sim \mathbb{D}'} \right\}} \mathcal{L}\left(\theta, (x,y)\,, \left(x + \delta^t_f, y\right)\right) \right] \tag{11}$$

where $(x, y) \sim \mathbb{D}$ denotes a clean data instance. $\mathcal{L}$ denotes the DML training loss. $\theta = \{\theta_n, \theta_b\}$ are the parameters of the model that does not have auxiliary BN layers.

3.3. Evaluating Multi-Distribution Inputs

For robustness assessment, the STAX and MTAX datasets were generated corresponding to the clean samples in the test sets of the CUB200 (Wah et al. (2011)), CARS 196 (Krause et al. (2013)), and SOP (Oh Song et al. (2016)) datasets. We used the *PGD* (Madry et al. (2017)) update with 20 iterations, calling it *PGD-20* attacks. We used 0.01 and 0.1 for the

ϵ constraint. for MTAXs, we used $T = 5$. The remaining attack hyper-parameters were kept the same as during the training time of attack generation.

4. Detailed Results

This section presents the detailed results of the comparison of our methods against baselines on clean data performance in Tab. 1, robustness against STAX inputs in Tab. 2, robustness against powerful STAX inputs generated using $\epsilon = 0.1$ in Tab. 3, clean data performance and adversarial robustness across architectures and SOTA S2SD methods in Tab. 5, and clean data performance for larger models with ***larger embedding dimensions*** in Tab. 7. Each table also presents the results for the case where MTAXs ***without separate batch normalization*** were used, which is included in the adversarial training method case. In addition, these tables show the results for ***additional values of the number of targets*** T for MTAX generation.

4.1. Performance on MTAX Inputs

We also evaluate the performance against MTAX inputs to test check decreased overlapped feature space in the MDProp models. The results for MTAX inputs are presented in Tab. 4. Clearly, MDProp models result in improved metrics for MTAX inputs.

4.2. Effect of Number of Adversarial Targets T

Fig. 1 illustrates the effect of T parameters on the performance of the trained model using MDProp. We conducted experiments using five values of T: 1, 2, 3, 5, and 10. It was found that increasing T improves performance on clean data only up to a certain number for which the predefined *generation* recipe's hyperparameters provide sufficient semantic capability to the attack generation procedure, causing the positions in the embedding space of generated MTAXs shift to the overlapped regions of the DML model under training. In particular, MDProp using clean and MTAXs performed best for $T = 3$, and MDProp using clean, STAXs, and MTAXs performed best for $T = 5$. For smaller values of T, lesser performance improvements result because of the decreased probability of finding highly overlapped embedding-space regions.

4.3. Results for PGD-20 attacks with $\epsilon = 0.1$

To evaluate the robustness gains for powerful attacks, we generate attacks with larger values of the ϵ constraint. We use $\epsilon = 0.1$ for generating single targeted AXs to compare the reduction in performance of AdvProp-D and MDProp. Similar to the case for PGD-20 attacks with $\epsilon = 0.01$, robustness gain was found to be marginally higher for the AdvProp-D followed by MDProp, which can be seen in Tab. 3. AdvProp-D and MDProp result in significantly high adversarial robustness compared to the baseline standard training and the adversarial training methods. Hence, we can conclude that our proposed AdvProp-D and the MDProp methods provide significant robustness gains for attacks of varying strength with different sizes of adversarial noise.

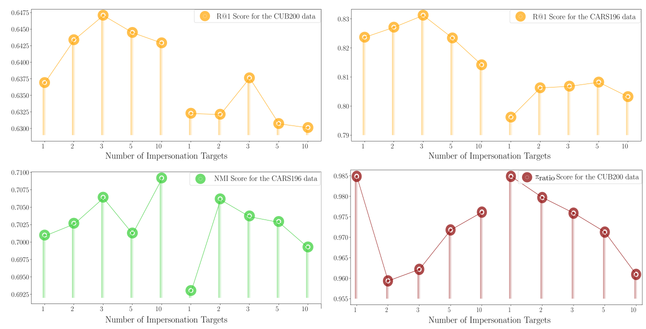


Figure 1: Impact of the number of attack targets T on the clean data performance. The sample trends generally demonstrate *improved* R@1, NMI, and π_{ratio} scores with the increase in T initially and then the decrease due to increased MTAX generation complexity and restricted attack generation procedure.

4.4. Results When MDProp Use 4 Separate BN Layers

Tab. 6 presents the results when MDProp uses three additional BN layers for the STAXs and MTAXs data generated for two different numbers of targets. Clearly, there were significant performance gains. However, the performance gains remained marginally lower than those of MDProp using the three separate BN layers presented in the paper.

4.5. Results for MDProp With Unadversarial Examples

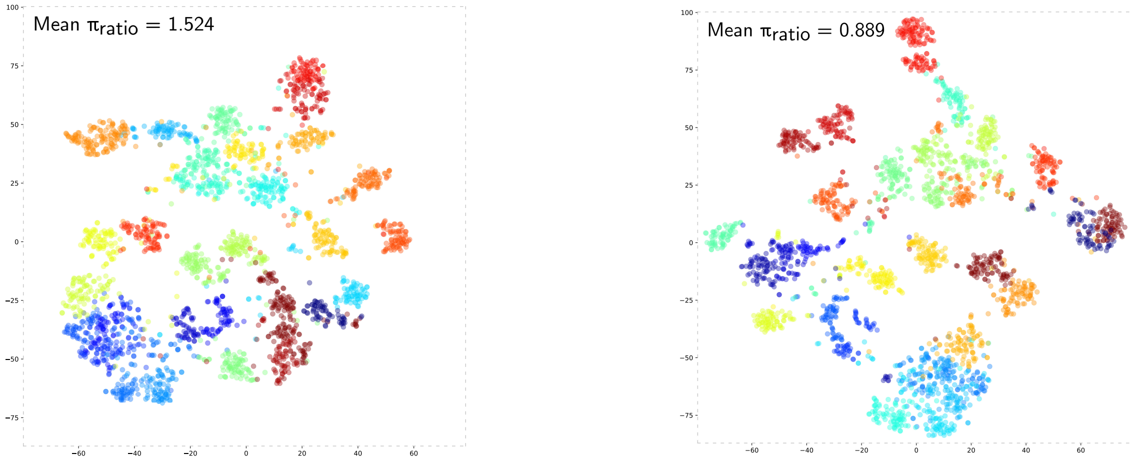
Tab. 8 displays the performance improvements achieved by our MDProp method using unadversarial examples. The combination of unadversarial examples and clean data in MDProp with two separate BN layers resulted in a clean data R@1 score of 63.56% for the Multisimilarity loss setting. However, no further performance gains were observed with the combination of unadversarial examples and adversarial data in the MDProp setting. This could be due to incompatibility with adversarial data or overlapping effects with the combination of clean and adversarial data during training. This issue remains an open research question.

We also evaluated the effect of using clean data augmentation techniques, such as standard crop and affine transformations, when using separate BN layers. However, no improvement was achieved, and in some cases, separate BN layers even resulted in a reduction in performance on clean data.

References

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.



(a) Standard Training (b) MDProp

Figure 2: t-SNE (Van der Maaten and Hinton (2008)) visualization of embedding space of DML models trained using (a) *standard training* and (b) *MDProp* on the *CARS196* dataset (Krause et al. (2013)). The decreased mean π_{ratio} score for MDProp means sparser embedding space.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1): 117–128, 2010.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016.

Method	T	CUB200 Data								CARS196 Data							
		Multisimilarity Loss				ArcFace Loss				Multisimilarity Loss				ArcFace Loss			
		R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}
ST	-	62.80 [±0.70]	83.70 [±0.54]	68.55 [±0.38]	1.007	62.22 [±0.01]	83.18 [±0.23]	67.79 [±0.42]	0.726	81.68 [±0.19]	93.47 [±0.27]	69.43 [±0.38]	1.129	79.17 [±0.73]	92.23 [±0.21]	66.99 [±0.04]	0.661
AT	1	61.73 [±0.71]	83.20 [±0.07]	68.04 [±0.51]	1.001	60.18 [±0.22]	82.61 [±0.31]	67.75 [±0.05]	0.721	80.02 [±0.42]	92.59 [±0.09]	68.56 [±0.06]	1.082	76.43 [±0.11]	91.14 [±0.06]	67.14 [±0.04]	0.686
	2	61.52 [±0.81]	82.84 [±0.18]	67.87 [±0.21]	0.994	60.30 [±0.10]	82.78 [±0.01]	67.78 [±0.01]	0.718	79.56 [±0.23]	92.38 [±0.13]	68.85 [±0.54]	1.091	76.70 [±0.04]	91.27 [±0.09]	66.90 [±0.06]	0.692
	3	61.72 [±0.33]	83.02 [±0.48]	67.93 [±0.54]	0.987	60.03 [±0.21]	82.64 [±0.02]	67.61 [±0.14]	0.737	79.59 [±0.04]	92.54 [±0.21]	69.09 [±0.08]	1.107	76.79 [±0.04]	91.31 [±0.36]	67.41 [±0.25]	0.687
	5	61.62 [±0.52]	83.13 [±0.41]	68.20 [±0.02]	0.993	60.26 [±0.62]	82.82 [±0.06]	67.81 [±0.09]	0.718	79.49 [±0.16]	92.32 [±0.04]	68.47 [±0.01]	1.094	76.90 [±0.07]	91.36 [±0.12]	66.96 [±0.34]	0.692
	10	61.24 [±0.66]	83.03 [±0.28]	67.82 [±0.52]	0.996	60.11 [±0.42]	82.55 [±0.69]	67.81 [±0.11]	0.704	79.75 [±0.18]	92.53 [±0.05]	68.75 [±0.57]	1.078	76.67 [±0.15]	91.37 [±0.16]	67.13 [±0.16]	0.693
AP$'$	1	63.69 [±0.13]	84.47 [±0.36]	69.15 [±0.27]	0.985	63.23 [±0.09]	84.11 [±0.05]	69.83 [±0.50]	0.723	82.37 [±0.96]	93.54 [±0.51]	70.10 [±1.13]	1.074	79.62 [±0.23]	92.63 [±0.18]	69.31 [±0.59]	0.681
MP$'$	2	64.34 [±0.69]	84.22 [±0.45]	69.50 [±0.28]	0.959	63.22 [±0.09]	84.24 [±0.13]	69.73 [±0.16]	0.720	82.72 [±0.18]	93.77 [±0.05]	70.27 [±0.78]	1.064	80.63 [±0.23]	93.24 [±0.18]	70.62 [±0.59]	0.689
	3	64.71 [±0.41]	84.45 [±0.25]	69.73 [±0.14]	0.962	63.77 [±0.04]	84.60 [±0.45]	69.90 [±0.89]	0.718	83.13 [±0.22]	93.81 [±0.16]	70.64 [±0.24]	1.056	80.69 [±0.16]	93.12 [±0.06]	70.38 [±0.30]	0.689
	5	64.45 [±0.38]	84.33 [±0.24]	69.63 [±0.28]	0.972	63.08 [±0.49]	84.22 [±0.31]	69.56 [±0.19]	0.716	82.35 [±0.06]	93.70 [±0.14]	70.13 [±0.24]	1.080	80.83 [±0.56]	93.24 [±0.54]	70.30 [±0.64]	0.688
	10	64.30 [±0.29]	84.46 [±0.31]	69.37 [±0.51]	0.976	63.02 [±0.19]	84.22 [±0.26]	69.56 [±0.19]	0.712	81.42 [±0.02]	93.65 [±0.23]	70.92 [±0.30]	1.084	80.33 [±0.08]	93.05 [±0.13]	69.93 [±0.17]	0.687
MP$''$	1,2	65.40 [±0.16]	84.72 [±0.01]	70.21 [±0.28]	0.956	64.13 [±1.01]	84.59 [±0.57]	70.26 [±0.13]	0.710	83.61 [±0.19]	94.21 [±0.11]	71.93 [±0.44]	1.049	81.75 [±0.45]	93.47 [±0.26]	71.52 [±0.03]	0.695
	1,3	65.41 [±0.17]	84.78 [±0.16]	69.90 [±0.06]	0.966	64.21 [±0.42]	85.03 [±0.12]	70.33 [±0.62]	0.709	83.41 [±0.41]	94.27 [±0.06]	71.87 [±0.20]	1.069	81.86 [±0.13]	93.86 [±0.01]	71.64 [±0.31]	0.696
	1,5	65.76 [±0.28]	85.23 [±0.21]	70.43 [±0.04]	0.974	64.07 [±0.11]	84.78 [±0.15]	70.32 [±0.06]	0.703	83.81 [±0.49]	94.31 [±0.26]	71.59 [±0.56]	1.055	82.02 [±0.36]	93.65 [±0.30]	72.43 [±0.18]	0.697
	1,10	65.03 [±0.05]	84.89 [±0.13]	69.77 [±0.01]	0.962	63.57 [±0.06]	84.36 [±0.07]	69.94 [±0.35]	0.707	83.61 [±0.25]	94.30 [±0.06]	71.75 [±0.17]	1.073	81.75 [±0.12]	93.71 [±0.18]	71.35 [±0.52]	0.699

Table 1: This table provides detailed results for the clean data performance of our AdvProp-D (AP$'$) and MDProp (MP) methods, as well as standard training (ST) (Roth et al. (2021)) and adversarial training (AT) baselines. The MP' and MP'' variants represent the addition of one and two extra BN layers, respectively. We evaluated model performance using multisimilarity (Wang et al. (2019)) and ArcFace (Deng et al. (2019)) losses. In comparison to Tab. 1 in the paper, this table demonstrates additional results for models trained using multiple MTAX targets T, as well as the effect of using separate batch normalization (BN) layers. Specifically, we provide results for the use of MTAXs in the AT setting without separate BN layers ($T = 2, 3, 5, 10$ for method AT in the table).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjorn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *International Conference on Machine Learning*, pages 8242–8252. PMLR, 2020.

Karsten Roth, Timo Milbich, Bjorn Ommer, Joseph Paul Cohen, and Marzyeh Ghassemi. Simultaneous similarity-based self-distillation for deep metric learning. In *International Conference on Machine Learning*, pages 9095–9106. PMLR, 2021.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

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/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019.

Method	T	CUB200 Data								CARS196 Data							
		Multisimilarity Loss				ArcFace Loss				Multisimilarity Loss				ArcFace Loss			
		R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}
ST	-	32.96 [±0.32]	64.40 [±0.35]	54.38 [±0.45]	1.429	38.45 [±1.19]	67.62 [±1.66]	55.92 [±0.46]	0.761	51.98 [±0.91]	79.99 [±0.80]	54.29 [±0.74]	1.455	34.82 [±1.14]	64.43 [±0.19]	42.85 [±0.20]	0.943
AT	1	38.88 [±0.60]	70.67 [±0.18]	58.33 [±0.21]	1.087	39.44 [±1.80]	70.04 [±0.51]	58.68 [±0.44]	0.743	52.42 [±0.11]	81.13 [±0.15]	56.52 [±0.60]	1.239	36.84 [±0.80]	67.87 [±0.62]	45.99 [±0.56]	0.829
	2	39.63 [±1.31]	71.55 [±0.68]	57.85 [±0.38]	1.078	39.78 [±1.07]	69.61 [±0.93]	58.65 [±0.57]	0.732	52.84 [±0.04]	81.31 [±0.42]	56.15 [±0.20]	1.214	37.33 [±0.42]	67.73 [±0.34]	46.28 [±0.01]	0.838
	3	39.53 [±1.62]	71.79 [±0.82]	58.28 [±0.66]	1.101	38.90 [±0.49]	69.51 [±0.47]	57.34 [±0.62]	0.764	51.90 [±0.38]	80.74 [±0.13]	55.20 [±0.32]	1.245	37.71 [±0.22]	68.78 [±0.56]	46.44 [±0.76]	0.833
	5	40.76 [±0.61]	72.24 [±0.59]	58.86 [±0.48]	1.088	39.27 [±0.86]	70.01 [±1.37]	57.59 [±0.08]	0.758	52.16 [±0.44]	80.74 [±0.23]	56.37 [±0.47]	1.216	37.27 [±1.03]	68.41 [±0.33]	46.26 [±0.45]	0.844
	10	39.35 [±1.08]	72.07 [±0.40]	67.82 [±0.52]	1.080	39.52 [±0.32]	69.18 [±0.18]	58.07 [±0.67]	0.745	52.32 [±1.63]	81.21 [±1.11]	55.55 [±0.28]	1.264	37.75 [±0.21]	68.60 [±0.66]	46.45 [±0.16]	0.827
AP′	1	58.80 [±2.15]	83.38 [±0.08]	62.21 [±0.29]	0.921	51.24 [±0.96]	77.81 [±0.09]	63.33 [±1.04]	0.712	79.11 [±1.35]	93.05 [±0.43]	70.87 [±0.99]	0.978	65.67 [±1.24]	87.11 [±0.83]	62.01 [±0.57]	0.723
MP′	2	56.99 [±0.45]	82.13 [±0.28]	66.69 [±0.30]	0.831	50.58 [±0.82]	78.23 [±0.49]	63.30 [±1.29]	0.691	79.78 [±0.47]	92.89 [±0.54]	71.09 [±0.76]	0.911	65.29 [±1.24]	87.38 [±0.83]	62.17 [±0.57]	0.728
	3	57.06 [±1.15]	81.42 [±0.80]	67.03 [±0.68]	0.838	50.96 [±0.28]	77.84 [±0.42]	62.73 [±0.36]	0.705	78.18 [±0.52]	92.55 [±0.38]	71.21 [±0.52]	0.896	64.76 [±0.71]	86.88 [±0.31]	62.38 [±0.25]	0.726
	5	55.11 [±1.65]	81.09 [±1.25]	66.52 [±0.97]	0.858	49.74 [±0.94]	77.24 [±0.77]	62.48 [±0.12]	0.698	78.27 [±0.01]	92.62 [±0.23]	69.39 [±0.59]	0.914	63.53 [±0.05]	86.47 [±0.06]	61.66 [±0.16]	0.735
	10	55.28 [±4.07]	81.00 [±1.19]	65.46 [±0.93]	0.841	49.01 [±0.93]	76.97 [±0.44]	61.99 [±0.63]	0.707	77.63 [±0.37]	92.37 [±0.04]	69.75 [±0.75]	0.925	61.54 [±0.54]	85.05 [±0.48]	59.03 [±1.14]	0.739
MP″	1,2	57.75 [±0.19]	82.93 [±0.18]	68.07 [±0.62]	0.838	53.30 [±1.22]	81.22 [±0.46]	67.79 [±0.67]	0.662	80.84 [±0.13]	93.35 [±0.23]	72.58 [±0.76]	0.912	74.88 [±0.76]	91.86 [±0.27]	70.67 [±0.28]	0.681
	1,3	58.23 [±0.15]	82.13 [±0.15]	67.73 [±0.19]	0.807	54.01 [±0.35]	80.96 [±0.10]	67.70 [±0.22]	0.654	80.17 [±0.01]	93.45 [±0.08]	72.20 [±1.12]	0.894	73.94 [±0.39]	91.67 [±0.35]	70.57 [±0.21]	0.691
	1,5	57.27 [±1.91]	82.21 [±0.84]	68.06 [±0.61]	0.836	51.93 [±0.33]	80.15 [±0.27]	66.28 [±0.02]	0.645	80.25 [±0.19]	93.50 [±0.01]	72.37 [±0.06]	0.899	73.91 [±0.07]	91.28 [±0.08]	69.01 [±1.13]	0.688
	1,10	55.51 [±1.08]	81.78 [±0.27]	67.20 [±0.01]	0.846	51.64 [±0.50]	79.90 [±0.42]	65.86 [±0.74]	0.655	79.31 [±1.10]	93.10 [±0.02]	71.17 [±0.94]	0.919	72.86 [±0.13]	90.95 [±0.14]	69.00 [±1.02]	0.686

Table 2: This table provides detailed results for the adversarial data performance of our AdvProp-D (AP′) and our MDProp (MP) methods, as well as the baseline standard training (ST) (Roth et al. (2021)) and adversarial training (AT) methods, when subjected to single-targeted PGD-20 attacks generated with $\epsilon = 0.01$. Performance is evaluated for models trained using multisimilarity (Wang et al. (2019)) and ArcFace (Deng et al. (2019)) losses. In addition to the results presented in Tab. 1 of the paper, this table includes results for models trained with multiple targeted adversarial examples (MTAXs) and the impact of using separate batch normalization (BN) layers, demonstrating the additional results for the AT setting without separate BN layers ($T = 2, 3, 5, 10$ for method AT). These results offer a comprehensive analysis of the performance and robustness of our methods in adversarial data scenarios, enabling better comparison and evaluation of their effectiveness.

Method	T	CUB200 Data								CARS196 Data							
		Multisimilarity Loss				ArcFace Loss				Multisimilarity Loss				ArcFace Loss			
		R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}
ST	-	16.60 [±0.17]	35.64 [±0.37]	30.32 [±0.37]	2.957	10.35 [±0.36]	30.69 [±0.86]	38.68 [±0.36]	1.252	17.80 [±0.21]	36.84 [±0.75]	23.15 [±0.09]	5.245	10.44 [±0.35]	31.81 [±0.47]	34.53 [±0.04]	1.398
AT	1	16.58 [±0.29]	44.67 [±0.45]	46.14 [±0.18]	1.914	16.67 [±0.02]	42.70 [±0.26]	47.65 [±0.05]	1.009	22.53 [±0.24]	53.85 [±0.40]	45.52 [±0.01]	2.351	11.94 [±0.44]	35.25 [±0.32]	36.70 [±0.42]	1.148
	2	16.06 [±0.84]	43.48 [±0.89]	45.99 [±0.22]	1.901	17.48 [±1.02]	43.38 [±1.26]	47.79 [±0.49]	0.992	20.84 [±1.03]	52.65 [±0.97]	44.32 [±0.43]	2.273	12.33 [±0.49]	35.90 [±0.22]	37.21 [±0.11]	1.153
	3	17.08 [±0.91]	41.19 [±0.50]	46.33 [±0.10]	1.951	16.30 [±0.14]	42.88 [±0.19]	47.80 [±0.32]	1.031	21.82 [±0.04]	53.52 [±0.09]	43.55 [±0.67]	2.337	12.54 [±0.45]	35.77 [±0.47]	36.63 [±0.84]	1.142
	5	16.47 [±0.56]	44.29 [±0.95]	46.16 [±0.40]	1.921	16.38 [±0.16]	42.80 [±0.58]	47.81 [±0.23]	1.021	21.20 [±0.20]	53.43 [±0.13]	44.32 [±0.65]	2.297	11.52 [±0.36]	35.81 [±1.03]	36.81 [±1.19]	1.170
	10	16.60 [±0.42]	44.21 [±0.69]	46.13 [±0.44]	1.899	17.12 [±0.28]	43.96 [±1.46]	47.81 [±0.03]	1.010	21.26 [±0.25]	53.60 [±0.18]	44.30 [±0.60]	2.394	11.86 [±1.02]	35.58 [±0.68]	37.50 [±0.35]	1.144
AP$'$	1	20.91 [±0.44]	50.80 [±0.68]	47.97 [±0.47]	1.857	18.11 [±0.33]	44.58 [±0.30]	48.82 [±0.41]	1.030	26.40 [±0.32]	59.86 [±0.67]	46.92 [±0.52]	2.133	14.42 [±0.28]	39.58 [±0.72]	39.42 [±0.39]	1.125
MP$'$	2	19.33 [±0.20]	48.62 [±0.52]	47.61 [±0.56]	1.740	17.98 [±1.39]	44.72 [±1.11]	49.00 [±0.08]	1.010	26.86 [±1.07]	59.34 [±0.53]	47.53 [±0.23]	1.992	14.38 [±0.28]	39.94 [±0.72]	39.72 [±0.40]	1.155
	3	18.90 [±0.23]	48.00 [±0.23]	47.56 [±0.14]	1.760	17.80 [±0.16]	44.97 [±0.86]	49.39 [±0.73]	1.020	25.83 [±0.36]	59.43 [±0.50]	47.28 [±0.79]	2.021	14.35 [±0.09]	39.61 [±0.30]	39.93 [±0.02]	1.157
	5	18.44 [±0.62]	48.09 [±0.74]	47.48 [±0.50]	1.794	18.96 [±1.49]	45.32 [±0.51]	48.73 [±0.30]	1.011	25.05 [±0.45]	58.85 [±0.47]	46.69 [±0.92]	2.047	13.57 [±0.84]	39.24 [±0.33]	40.02 [±0.94]	1.149
	10	18.80 [±1.22]	48.30 [±1.08]	47.21 [±0.41]	1.735	17.73 [±0.40]	45.06 [±0.84]	48.70 [±0.57]	1.022	25.10 [±0.91]	58.48 [±0.14]	46.23 [±0.25]	2.039	13.10 [±0.11]	38.75 [±0.76]	39.31 [±0.79]	1.156
MP$''$	1,2	19.84 [±0.79]	50.38 [±0.56]	49.11 [±1.21]	1.729	18.14 [±0.01]	49.04 [±0.29]	49.47 [±0.41]	1.020	28.40 [±0.40]	64.10 [±0.27]	50.03 [±0.19]	1.985	26.95 [±0.53]	61.25 [±0.14]	49.45 [±0.37]	0.991
	1,3	20.25 [±0.10]	50.51 [±0.37]	49.00 [±0.29]	1.636	18.29 [±0.44]	47.45 [±0.72]	49.75 [±0.16]	1.011	28.84 [±0.73]	63.87 [±0.61]	49.98 [±0.06]	1.933	26.46 [±0.15]	60.62 [±0.33]	49.53 [±0.44]	1.001
	1,5	20.61 [±0.16]	50.61 [±0.23]	49.42 [±0.62]	1.731	18.93 [±0.38]	48.45 [±0.79]	49.53 [±0.48]	0.994	28.19 [±0.65]	63.81 [±0.25]	50.12 [±0.37]	1.961	26.27 [±0.19]	60.81 [±0.26]	49.68 [±0.05]	1.002
	1,10	20.01 [±0.86]	50.06 [±0.62]	49.06 [±0.65]	1.705	17.69 [±0.77]	47.53 [±1.51]	48.66 [±0.71]	0.999	27.97 [±0.24]	63.24 [±0.44]	50.04 [±0.45]	1.979	26.80 [±0.58]	60.57 [±0.33]	48.92 [±0.44]	1.004

Table 3: This table presents a detailed comparison of the adversarial data performance of our AdvProp-D (AP$'$) and MDProp (MP) methods against the baseline standard training (ST) and adversarial training (AT) (Roth et al. (2021)), with a ***stronger*** single-targeted PGD-20 attack generated using $\epsilon = 0.1$. We evaluated the performance of the models trained using both Multisimilarity (Wang et al. (2019)) and ArcFace (Deng et al. (2019)) losses. Compared to Tab. 1 in the paper, this table includes additional results for the models' robustness trained using multiple MTAX targets T and the impact of separate batch normalization layers on the results. We present the additional results for the AT setting, where separate BN layers were not used, with $T = 2, 3, 5, 10$ for the method AT in the table. This comprehensive comparison provides insights into the effectiveness of our methods against different types of attacks and highlights the impact of various hyperparameters on the models' performance.

Method	T	R@1	R@4	NMI	π_{ratio}
ST	-	36.35 [±0.41]	62.23 [±0.87]	47.69 [±0.42]	1.447
AT	1	55.31 [±0.70]	82.40 [±0.25]	67.83 [±0.22]	0.755
	2	53.76 [±0.92]	81.62 [±0.29]	66.84 [±0.26]	0.757
	3	54.16 [±1.01]	82.43 [±0.19]	67.21 [±0.66]	0.775
	5	54.72 [±0.17]	81.91 [±0.64]	67.73 [±0.54]	0.758
	10	54.32 [±0.22]	81.66 [±0.25]	67.74 [±0.15]	0.765
AP′	1	59.97 [±0.17]	83.83 [±1.02]	71.27 [±1.74]	0.746
MP′	2	61.00 [±0.53]	86.43 [±0.50]	72.14 [±0.76]	0.612
	3	61.13 [±0.85]	86.05 [±0.41]	71.88 [±0.61]	0.617
	5	60.65 [±0.54]	86.19 [±0.55]	71.92 [±0.58]	0.632
	10	60.55 [±2.49]	85.76 [±1.39]	71.45 [±0.29]	0.618
MP″	1,2	62.69 [±0.00]	86.96 [±0.45]	72.68 [±0.81]	0.621
	1,3	62.04 [±0.07]	86.34 [±0.18]	72.81 [±0.10]	0.606
	1,5	61.41 [±0.38]	86.49 [±0.17]	72.33 [±0.75]	0.624
	1,10	60.11 [±1.56]	86.38 [±0.28]	71.37 [±0.33]	0.619

Table 4: Detailed results for the adversarial data performance of our AdvProp-D (AP′) and our MDProp (MP) methods, compared against the baseline standard training (ST) (Roth et al. (2021)) and adversarial training (AT) methods, when subjected to **white-box multi-targeted PGD-20 attacks with** $T = 5$ using $\epsilon = 0.01$. The evaluation was conducted on models trained with multisimilarity loss (Wang et al. (2019)) on CUB200 (Wah et al. (2011)) data.

Method	T	ResNet50+S2SD Method								ResNet18				ResNet152			
		Clean CUB200 Data				*Adversarial* CUB200 Data				Clean CUB200 Data				Clean CUB200 Data			
		R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}
ST	-	67.69 [±0.13]	86.32 [±0.08]	71.46 [±0.13]	1.123	47.35 [±1.24]	76.08 [±0.64]	60.26 [±0.40]	1.393	58.81 [±0.52]	81.34 [±0.33]	66.12 [±0.45]	1.131	65.11 [±0.28]	84.64 [±0.10]	69.70 [±0.02]	0.967
AT	1	66.46 [±0.59]	85.63 [±0.12]	70.78 [±0.40]	1.092	45.13 [±1.09]	75.40 [±0.40]	60.89 [±0.25]	1.416	58.33 [±0.13]	81.15 [±0.11]	65.54 [±0.31]	1.093	64.98 [±0.47]	84.83 [±0.46]	70.56 [±0.14]	0.896
	2	66.10 [±0.45]	85.52 [±0.15]	70.69 [±0.11]	1.109	46.49 [±0.04]	75.59 [±0.05]	60.51 [±0.46]	1.383	58.90 [±0.52]	81.27 [±0.15]	65.58 [±0.40]	1.088	64.44 [±0.19]	84.59 [±0.21]	70.49 [±0.03]	0.923
	3	66.14 [±0.95]	85.63 [±0.37]	70.76 [±0.07]	1.099	45.90 [±0.27]	75.73 [±0.08]	61.23 [±0.29]	1.419	58.62 [±0.15]	81.35 [±0.01]	65.90 [±0.23]	1.092	64.61 [±0.88]	84.32 [±0.17]	69.91 [±0.54]	0.893
	5	66.18 [±0.99]	85.55 [±0.11]	70.67 [±0.35]	1.095	46.70 [±1.85]	75.71 [±0.81]	60.50 [±0.12]	1.391	58.45 [±0.23]	81.15 [±0.02]	65.85 [±0.28]	1.093	64.46 [±0.40]	84.27 [±0.31]	70.60 [±0.05]	0.910
	10	66.08 [±0.67]	85.50 [±0.27]	70.91 [±0.50]	1.113	44.48 [±0.41]	75.50 [±0.51]	60.59 [±1.23]	1.414	58.47 [±0.16]	81.37 [±0.04]	65.77 [±0.03]	1.091	64.06 [±0.26]	84.18 [±0.14]	70.22 [±0.72]	0.893
AP$'$	1	68.14 [±0.16]	86.45 [±0.05]	71.18 [±0.10]	1.091	62.47 [±1.37]	84.18 [±0.59]	69.64 [±0.11]	1.102	60.91 [±0.47]	82.52 [±0.44]	66.52 [±0.57]	1.028	66.95 [±0.04]	85.88 [±0.23]	71.72 [±0.21]	0.916
MP$'$	2	68.32 [±0.23]	86.52 [±0.01]	71.80 [±0.01]	1.111	63.10 [±0.84]	85.10 [±0.19]	69.81 [±0.69]	1.094	60.91 [±0.54]	82.33 [±0.00]	66.59 [±0.30]	1.042	66.99 [±0.33]	85.65 [±0.21]	71.80 [±0.13]	0.907
	3	68.76 [±0.24]	86.47 [±0.27]	71.78 [±0.29]	1.106	62.47 [±0.13]	84.66 [±0.84]	69.62 [±0.86]	1.109	60.92 [±0.18]	82.82 [±0.11]	66.56 [±0.30]	1.024	66.66 [±0.24]	85.77 [±0.03]	71.73 [±0.35]	0.910
	5	68.54 [±0.43]	86.45 [±0.16]	71.86 [±0.01]	1.108	62.88 [±1.05]	84.41 [±0.02]	69.79 [±0.02]	1.105	61.41 [±0.62]	82.59 [±0.23]	66.46 [±0.30]	1.045	66.77 [±0.08]	85.51 [±0.28]	71.53 [±0.03]	0.914
	10	68.48 [±0.45]	86.46 [±0.14]	71.86 [±0.01]	1.149	62.01 [±0.40]	84.07 [±0.02]	69.84 [±0.10]	1.138	60.82 [±0.38]	82.33 [±0.27]	66.49 [±0.14]	1.054	66.38 [±0.64]	85.28 [±0.35]	71.25 [±0.01]	0.910
MP$''$	1,2	68.62 [±0.26]	86.76 [±0.23]	72.28 [±0.17]	1.197	65.23 [±0.21]	86.75 [±0.15]	70.42 [±0.94]	1.044	61.68 [±0.66]	83.00 [±0.17]	67.58 [±0.32]	1.042	67.48 [±0.61]	86.08 [±0.13]	72.25 [±0.13]	0.907
	1,3	69.04 [±0.21]	86.88 [±0.16]	71.99 [±0.13]	1.165	64.83 [±0.54]	86.05 [±1.06]	70.27 [±0.29]	1.026	61.64 [±0.92]	83.18 [±0.54]	67.57 [±0.69]	1.048	67.22 [±0.05]	86.00 [±0.25]	72.48 [±0.43]	0.910
	1,5	69.08 [±0.23]	87.19 [±0.19]	71.98 [±0.17]	1.252	65.01 [±0.02]	86.60 [±0.08]	71.14 [±0.21]	1.034	61.67 [±0.47]	82.75 [±0.17]	67.38 [±0.47]	1.091	67.63 [±0.16]	86.20 [±0.06]	72.61 [±0.01]	0.902
	1,10	68.74 [±0.00]	86.87 [±0.24]	72.54 [±0.39]	1.208	64.35 [±0.17]	86.52 [±0.24]	70.65 [±0.89]	1.039	61.49 [±0.11]	82.84 [±0.19]	67.17 [±0.36]	1.052	67.12 [±0.31]	86.27 [±0.13]	72.05 [±0.08]	0.920

Table 5: The table presents detailed results showcasing the clean data performance and robustness gains achieved by the AdvProp-D and MDProp methods across ResNet18, ResNet50, and ResNet152 architectures of varying sizes on the CUB200 (Wah et al. (2011)) dataset. The implementation of ResNet50 utilized the state-of-the-art S2SD method (Roth et al. (2021)). The table highlights the effectiveness of our proposed methods in improving the clean data performance and robustness of DML models on the CUB200 dataset. The results demonstrate the generalization capability of our methods across various architecture sizes, showcasing their ability to provide consistent improvements in performance.

Method	T	CUB200 Data				CARS196 Data			
		R@1	R@4	NMI	π_{ratio}	R@1	R@4	NMI	π_{ratio}
MP'''	1,3,5	65.13 [±0.84]	85.07 [±0.58]	70.25 [±0.01]	0.985	84.08 [±0.03]	94.57 [±0.25]	72.23 [±0.47]	1.058
	1,3,10	65.11 [±0.55]	84.93 [±0.47]	69.93 [±0.01]	0.988	84.07 [±0.23]	94.46 [±0.11]	71.98 [±0.25]	1.078

Table 6: Results of MDProp (MP) method with three additional BN layers on CUB200 (Wah et al. (2011)) and CARS 196 (Krause et al. (2013)) datasets using multisimilarity (Wang et al. (2019)) loss. The table presents the performance of the method for different numbers of attack targets T used for adversarial data generation.

Method	T	R@1	R@4	NMI	π_{ratio}
ST	-	64.97	85.15	66.52	1.163
AT	1	65.20	84.65	66.59	1.348
	2	64.87	84.49	66.89	1.352
	3	65.03	84.78	66.76	1.352
	5	65.23	84.88	67.09	1.354
	10	65.30	84.92	66.81	1.346
AP'	1	68.85	68.91	68.33	1.214
MP	2	68.39	86.56	69.30	1.211
	3	68.91	86.69	68.85	1.181
	5	68.19	86.36	68.63	1.190
	10	68.62	86.95	69.24	1.170

Table 7: Results when an embedding size of 512 was used while training the ResNet152 architecture with the S2SD (Roth et al. (2021)) method on the CUB200 (Wah et al. (2011)) dataset. MDProp, followed by AdvProp-D, demonstrated significant clean data performance gains over the baselines.

Method	R@1	R@4	NMI
ST	62.80 [±0.70]	83.70 [±0.54]	68.55 [±0.38]
MP	61.68 [±0.52]	82.55 [±0.32]	68.05 [±0.21]
MP′	63.56 [±0.84]	83.75 [±0.58]	68.97 [±0.41]

Table 8: Comparison of MDProp's performance on CUB200 Wah et al. (2011) dataset, using multisimilarity Wang et al. (2019) loss for training with *unadversarial examples*. The table compares the use of separate BN layers (MP') versus training without separate BN layers (MP) in addition to the standard training baseline (ST). All experiments were conducted using ResNet50 architecture. T denotes the number of attack targets used for generating different types of adversarial data.