ADVERSARIAL DEEP METRIC LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning a distance metric between pairs of examples is widely important for various tasks. Deep Metric Learning (DML) utilizes deep neural network architectures to learn semantic feature embeddings where the distance between similar examples is close and dissimilar examples are far. While the underlying neural networks produce good accuracy on naturally occurring samples, they are vulnerable to adversarially-perturbed samples that can reduce their accuracy. To create robust versions of DML models, we introduce a robust training approach. A key challenge is that metric losses are not independent — they depend on all samples in a mini-batch. This sensitivity to samples, if not accounted for, can lead to incorrect robust training. To the best of our knowledge, we are the first to systematically analyze this dependence effect and propose a principled approach for robust training of deep metric learning networks that accounts for the nuances of metric losses. Using experiments on three popular datasets in metric learning, we demonstrate the DML models trained using our techniques display robustness against strong iterative attacks while their performance on unperturbed (natural) samples remains largely unaffected.

1 INTRODUCTION

Many machine learning (ML) tasks rely on ranking entities based on the similarities of data points in the same class. Deep Metric Learning (DML) is a useful technique for such tasks, particularly for applications involving test-time inference of classes that are not present during training (e.g., zero-shot learning). Example applications of DML include person re-identification (Hermans et al., 2017), face verification (Schroff et al., 2015; Deng et al., 2019), phishing detection (Abdelnabi et al., 2020), and image retrieval (Wu et al., 2017; Roth et al., 2019). At its core, DML relies on state-of-the-art deep learning techniques that can produce lower-dimensional semantic feature embeddings of high-dimensional inputs. Points in this embedding space cluster similar inputs together while dissimilar inputs are far apart.

Unfortunately, the underlying deep learning models are vulnerable to adversarial examples (Szegedy et al., 2014; Biggio et al., 2013) — inconspicuous input changes that can cause the model to output attacker-desired values. Thus, DML models themselves are vulnerable to adversarial examples. Given their wide usage in diverse ML tasks, including security-oriented ones, it is important to train robust DML models that withstand attacks. This paper tackles the open problem of training DML models using robust optimization techniques (Ben-Tal et al., 2009; Madry et al., 2018).

A key challenge in robust training of DML models concerns the so-called *metric losses* (Wu et al., 2017; Wang et al., 2019; Chechik et al., 2010; Schroff et al., 2015). Unlike loss functions used in typical deep learning settings, DML loss for a single data point depends on the other data points in the mini-batch. A sampling process selects points for a mini-batch, and thus, the DML losses are sensitive to this process as well. For example, the widely-used triplet loss requires three input points: an anchor, a positive sample similar to the anchor, and a negative sample dissimilar to the anchor. To compute this loss, for a batch of size B, this would require O(B^3), making the training process inefficient. Thus, a sampling process ensures that a mini-batch contains enough positive and negative examples for the training to be useful while keeping the batch small enough to be efficient.

This *dependence* of the DML loss on the contents of the mini-batch poses a challenge to adversarial training: (1) it is unclear what points should be adversarially perturbed; and (2) it is unknown whether the perturbations would cause training instability. Training a DML model is sensitive to

the sampling process, and selecting samples that are too "hard" or "negative" can lead to training collapse (Wu et al., 2017).

We systematically approach the above challenges and contribute a robust training objective formulation for DML models by considering the widely-used triplet loss. Our key insight is that during an inference-time attack, the positive point in the triplet will be modified by the attacker, and thus, during training, we must perturb positive points in the triplets, instead of anchors or negative points. Recent work has attempted to train robust DML models, but they do not consider the issue of loss dependence and sensitivity to sampling (Abdelnabi et al., 2020). This leads to non-robust DML models (Panum et al., 2020).

Contributions.

- We contribute a principled robust training framework for Deep Metric Learning models by considering the dependence of triplet loss on the other data points in the mini-batch and the sensitivity to sampling.
- We experiment with three commonly-used datasets for vision-based deep metric learning (CUB200-2011, CARS196, SOP) and show that naturally-trained models do not have any robustness — their accuracy drops to close to zero when subjected to PGD attacks that we formulate.
- Using our robust formulation, we achieve good robustness. For example, for a PGD attacker with 5 iterations and $\|\delta\|_\infty < 0.01$, we obtain an adversarial accuracy of 46.0 compared to the state-of-the-art natural accuracy baseline of 66.0 for the SOP dataset (in terms of R@1 score, a common metric in DML to assess the accuracy of models). Furthermore, the resulting robust model accuracies are largely unaffected for natural (unperturbed) samples.

2 Related Work

Deep Metric Learning. DML is a popular technique to obtain semantic feature embeddings with the property that similar inputs are geometrically close to each other in the embedding space while dissimilar inputs are far apart (Roth et al., 2020). DML employs a variety of *metric losses* such as contrastive (Hadsell et al., 2006), triplet (Schroff et al., 2015), lifted-structure (Hermans et al., 2017), and angular loss (Wang et al., 2017). Recent surveys (Roth et al., 2020; Musgrave et al., 2020) highlight that performance of newer metric losses are lesser than previously reported. Thus, we choose to focus on the two established metric losses, contrastive and triplet loss, as they are widely used and have good performance.

Adversarial Robustness. Since early work in the ML community discovered adversarial examples in deep learning models (Szegedy et al., 2014; Biggio et al., 2013), a big focus has been to train adversarially-robust models. We focus on robust optimization-based training that utilizes a saddle-point formulation (min-max) (Ben-Tal et al., 2009; Madry et al., 2018). To the best of our knowledge, no prior work has considered training DML models using robust-optimization techniques. Recent work, however, has used metric losses to improve adversarial training for standard deep network architectures (e.g., CNNs) (Mao et al., 2019; Li et al., 2019). These techniques use metric losses (e.g., triplet) instead of traditional ones like cross-entropy. By contrast, our goal is to create a robust training objective for DML models themselves. This requires considering the dependence of metric losses on mini-batch items and the sampling process that derives those items. We propose a principled framework for robustly training DML models that considers these factors.

Duan et al. (2018) propose the "Deep Adversarial Metric Learning" framework that uses generative models (e.g., GANs (Goodfellow et al., 2014)) during training to generate hard negative samples from easy negatives. We observe that this work is concerned with better *natural* training of DML models rather than adversarial training, which is the focus of this work.

3 TOWARDS ROBUST DEEP METRIC MODELS

First, we describe some basic machine learning (ML) notation and concepts required in this paper. We assume a data distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the sample space and $\mathcal{Y} = \{y_1, \cdots, y_L\}$ is the finite space of labels. Let $\mathcal{D}_{\mathcal{X}}$ be the marginal distribution over $\mathcal{X}$ induced by $\mathcal{D}$ [1]. Given $Y \subseteq \mathcal{Y}$ we define $\mathcal{D}_Y$ to be the measure of the subsets of $\mathcal{X} \times Y$ induced by $\mathcal{D}$. For $y \in \mathcal{D}$, $\mathcal{D}_y$ and $\mathcal{D}_{-y}$ denote the measures of the sets $D_{\{y\}}$ and $D_{\mathcal{Y}\setminus\{y\}}$, respectively.

In the *empirical risk minimization (ERM)* framework we wish to solve the following optimization problem:

$$\min_{w \in \mathcal{H}} \; E_{(\mathbf{x},y)\sim\mathcal{D}} \; l(w, \mathbf{x}, y)$$

In the equation given above $\mathcal{H}$ is the hypothesis space and l is the loss function. We will denote vectors in boldface (e.g. $\mathbf{x}$, $\mathbf{y}$). Since the distribution is usually unknown, a learner solves the following problem over a data set $S = \{(\mathbf{x}_1, y_1), \cdots .(\mathbf{x}_n, y_n)\}$ sampled from the distribution $\mathcal{D}$.

$$\min_{w \in \mathcal{H}} \; \frac{1}{n} \sum_{i=1}^{n} l(w, \mathbf{x}_i, y_i)$$

Once we have solved the optimization problem given above, we obtain a $w^* \in \mathcal{H}$ which yields a classifier $F \colon \mathcal{X} \to \mathcal{Y}$ (the classifier is usually parameterized by w^*, but we will omit it for brevity).

3.1 DEEP METRIC MODELS

A *deep metric model* f_θ is function from $\mathcal{X}$ to S^d, where $\theta \in \Theta$ is a parameter and S^d is an unit sphere in $\mathbb{R}^d$ (i.e. $\mathbf{x} \in S^d$ iff $\|\mathbf{x}\|_2 = 1$). Since deep metric models embed a space $\mathcal{X}$ (which can itself be a metric space) in another metric space, we also sometimes refer to them deep embedding. Deep metric models use very different loss functions than typical classification networks described previously. Next we discuss two kinds of loss functions – contrastive and triplet. Let $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ be a dataset drawn from $\mathcal{D}$. A *contrastive* loss function l_c is defined over a pair $(\mathbf{x}, y)$, $(\mathbf{x}_1, y_1)$ of labeled samples from $\mathcal{X} \times \mathcal{Y}$ and is defined as:

$$l_c(\theta, (\mathbf{x}, y), (\mathbf{x}_1, y_1)) \quad = \quad 1_{y=y_1} \; d^2_\theta(\mathbf{x}, \mathbf{x}_1) + 1_{y \neq y_1} \; [\alpha - d^2_\theta(\mathbf{x}, \mathbf{x}_1)]$$

In the equation given above, 1_E is an indicator function for event E (1 if event E is true and 0 otherwise), and $d^2_\theta(\mathbf{x}, \mathbf{x}_1)$ is $\sum_{j=1}^{d} (f_\theta(\mathbf{x})_j - f_\theta(\mathbf{x}_1)_j)^2$, the square of the ℓ_2 distance in the embedding space. A *triplet* loss function l_t is defined over three $(\mathbf{x}, y)$, $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$ labeled samples and is defined as follows:

$$l_t(\theta, (\mathbf{x}, y), (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)) \quad = \quad 1_{y=y_1} \; 1_{y \neq y_2} \; [d^2_\theta(\mathbf{x}, \mathbf{x}_1) - d^2_\theta(\mathbf{x}, \mathbf{x}_2) + \alpha]_+$$

In the equation given above $[x]_+$ is $\max(x, 0)$.

3.2 ATTACKS ON DEEP METRIC MODELS

Assume that we have learned a deep embedding network with parameter $\theta \in \Theta$ using one of the loss functions described above. Next we describe how the network is used. Let $A = \{(\mathbf{a}_1, c_1), \cdots, (\mathbf{a}_m, c_m)\}$ be a reference or test dataset (e.g. a set of faces along with their label). A is distinct from the dataset S used during training time. Suppose we have a sample $\mathbf{z}$ and let $k(A, \mathbf{z})$ be the index that corresponds to $\arg\min_{j \in \{1, \cdots, m\}} d_\theta(\mathbf{a}_j, \mathbf{z})$. We predict the label of $\mathbf{z}$ as $lb(A, \mathbf{z}) = c_{k(A,\mathbf{z})}$ (we will use the functions $k(.,.)$ and $lb(.,.)$ throughout this section).

Next we describe test-time attacks on a deep embedding with parameter θ. Let $A_y \subset A$ be the subset of the reference dataset with label $y \in \mathcal{Y}$ (i.e. A_y is equal to $\{(\mathbf{a}_j, y) \mid (\mathbf{a}_j, y) \in A\}$). Let $\mathbf{z} \in \mathcal{X}$. *Untargeted attack* on $\mathbf{z}$ can be described as follows (we want the perturbed point to have a different label than before):

$$\begin{array}{l} \min_{\delta \in \mathcal{X}} \; \mu(\delta) \\ \textit{such that } lb(A, \mathbf{z}) \neq lb(A, \mathbf{z} + \delta) \end{array}$$

[1] The measure of set $Z \subseteq \mathcal{X}$ under distribution $\mathcal{D}_{\mathcal{X}}$ is the measure of the set $Z \times \mathcal{Y}$ in distribution $\mathcal{D}$.

Targeted attack (with a target label $t \neq lb(A, \mathbf{z})$) can be described as follows (we desire to the predicted label of the perturbed point to be a specific label):

$$\begin{array}{l} \min_{\delta \in \mathcal{X}} \ \mu(\delta) \\ \textit{such that } lb(A, \mathbf{z} + \delta) = t \end{array}$$

In the formulations given above we assume that $\mathcal{X}$ is a metric space with μ a metric on $\mathcal{X}$ (e.g. $\mathcal{X}$ could $\mathbb{R}^n$ with usual norms, such as ℓ_∞, ℓ_1, or ℓ_p (for $p \geq 2$)). Any algorithm that solves the optimization problem described above leads to a specific attack on deep metric models.

3.3 Robust Deep Metric Models

Let $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ be a dataset drawn from distribution $\mathcal{D}$. For a sample $(\mathbf{x}_i, y_i)$ where $1 \leq i \leq n$ we define the following surrogate loss function $\hat{l}(\theta, (x_i, y_i), S)$ for the contrastive loss function l_c :

$$\hat{l}(\theta, (x_i, y_i), S) \quad = \quad \frac{1}{n} \sum_{j=1}^{n} l_c(\theta, (\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j))$$

Similarly, for the triplet loss function l_t we can define the following surrogate loss function:

$$\hat{l}(\theta, (x_i, y_i), S) \quad = \quad \frac{1}{n_{y_i} n_{y_i}^-} \sum_{j=1}^{n_{y_i}} \sum_{k=1}^{n_{y_i}^-} l_t(\theta, (\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j), (\mathbf{x}_k, y_k))$$

Let S_y and S_{-y} be defined as the following sets: $\{(\mathbf{x}, y) \ | \ (\mathbf{x}, y) \in S\}$ and $\{(\mathbf{x}, y') \ | \ (\mathbf{x}, y') \in S \text{ and } y' \neq y\}$. In the equation given above the sizes of the sets S_y and S_{-y} are denoted by n_y and n_y^-.

Having defined the surrogate loss function $\hat{l}$ the learner's problem can be defined as:

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \hat{l}(\theta, (\mathbf{x}_i, y_i), S)$$

Recall that the learner's problem for the usual classification case is:

$$\min_{w \in \mathcal{H}} \ \frac{1}{n} \sum_{i=1}^{n} l(w, \mathbf{x}_i, y_i)$$

Note that in the classification case the loss function $l(w, \mathbf{x}_i, y_i)$ of a sample $(\mathbf{x}_i, y_i)$ does not depend on the other samples in the dataset S. However, in the deep metric model case the surrogate loss function $\hat{l}(\theta, (\mathbf{x}_i, y_i), S)$ for a sample $(\mathbf{x}_i, y_i)$ depends on the rest of the data set S. This is the main difference between the embedding and classification scenarios.

Formulation 1. Let $B_p(\mathbf{x}, \epsilon)$ denote the ϵ-ball around the sample $\mathbf{x}$ using the ℓ_p-norm. The straightforward robust formulation is given in the equation below.

$$\min_{\theta \in \Theta} \max_{(\mathbf{z}_1, \cdots, \mathbf{z}_n) \in \prod_{i=1}^{n} B_p(\mathbf{x}_i, \epsilon)} \ \frac{1}{n} \sum_{i=1}^{n} \hat{l}(\theta, (\mathbf{z}_i, y_i), S)$$

In the formulation given above, all samples are adversarially perturbed at the same time (note that the $\max$ is outside the summation). Therefore, this formulation is not convenient for current training algorithms, such as SGD and ADAM. This is because the entire dataset S has to be perturbed at the same time. Moreover, this formulation is not conducive to various sampling strategies used in training of deep metric models.

Formulation 2. In this formulation we push the $\max$ inside the summation sign of the optimization formulation.

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{z} \in B_p(\mathbf{x}_i, \epsilon)} \hat{l}(\theta, (\mathbf{z}, y_i), S)$$

In the formulation given above, the sample $\mathbf{x}_i$ is adversarially perturbed while other samples in the dataset S are kept intact while computing the surrogate loss function $\hat{l}$. This formulation is more conducive to current training techniques, such as SGD and ADAM. Moreover, this formulation is also conducive to various strategies for sampling pairs and triplets used in deep metric models.

Formulation 3. Our third formulation adds a regularizer which enforces the following informal constraint: if $\mathbf{x}$ changes a bit, the distance in the embedding space does not change too much.

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} [\hat{l}(\theta, (\mathbf{z}, y_i), S) \ + \ \lambda \max_{\mathbf{z} \in B_p(\mathbf{x}_i, \epsilon)} d_\theta(\mathbf{z}, \mathbf{x}_i)]$$

3.4 Adversarial Training

An algorithm for solving one of the robust optimization formulations leads to a robust model. Our algorithm can be viewed as adversarial training for deep metric models (i.e. replace a sample by the "worst case" sample before normal training). Roughly speaking, in the embedding space an adversary wants to push a sample further away from a sample of the same label and closer to the sample of a different label. Our adversarial training is based on this intuition. For the sake of concreteness we consider the ℓ_∞ metric, but all our ideas apply to other norms.

Let $\rho(\mathbf{x}_i, \mathbf{x}_j)$ be a function that outputs a perturbed variant of the datapoint $\mathbf{x}_i$ with respect to $\mathbf{x}_j$, and is given by:

$$\rho(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i + \lambda \left[1_{y_i = y_j} \left[\underset{\delta \in B_\infty(0,\epsilon)}{\arg\max} \ d_\theta(\mathbf{x}_i + \delta, \mathbf{x}_j) \right] + 1_{y_i \neq y_j} \left[\underset{\delta \in B_\infty(0,\epsilon)}{\arg\min} \ d_\theta(\mathbf{x}_i + \delta, \mathbf{x}_j) \right] \right]$$

Note that when $1_{y_i = y_j}$ we maximize the distance of the perturbed point from $\mathbf{x}_j$; otherwise we try minimize the distance from $\mathbf{x}_j$. The parameter $\lambda \in \{0, 1\}$ is a discrete random variable for which $P(\lambda = 1) \in [0; 1]$ is a specified attack rate.

The robust optimization objective for contrastive loss functions be given by:

$$\underset{\theta \in \Theta}{\arg\min} \ l(\theta, (\mathbf{x}, y), (\rho(\mathbf{x}_1, \mathbf{x}), y_1)) \ .$$

Similarly, let the robust optimization objective for triplet-based metric losses be given by:

$$\underset{\theta \in \Theta}{\arg\min} \ l(\theta, (\mathbf{x}, y), (\rho(\mathbf{x}_1, \mathbf{x}), y_1), (\mathbf{x}_2, y_2)) \ .$$

4 Experiments

Our experiments support the following research questions.

Q1. How robust are *naturally* trained Deep Metric Learning (DML) models towards established adversarial example attacks?

Among commonly used datasets for visual similarity, we find that DML models, trained with established hyper-parameters, are vulnerable to adversarial examples, similar to non-DML models (Table 1). This forms our baseline.

Q2. What is the accuracy of DML models when they are trained using our robust formulation?

We find that DML models can be trained to become more robust towards a given threat model. For example, for a PGD attacker with 5 iterations and $\|\delta\|_\infty < 0.01$, we obtain an adversarial accuracy of 46.0 compared to the state-of-the-art natural accuracy baseline of 66.0 for the SOP dataset (Table 2).

4.1 EXPERIMENTAL SETUP

For all experiments, unless otherwise stated, we use hyper-parameter choices from Roth et al. (2020). These choices achieve state-of-the-art accuracy for naturally-trained DML networks. We used an NVIDIA Tesla V100 GPU. Anonymized code is available at https://github.com/starving-panda/adversarial-metric-learning.

Model & Optimizer. We use a ResNet50 model (He et al., 2016) initialized with pre-trained ImageNet weights and frozen batch-norm layers. We replace the last output layer with a fully-connected untrained embedding layer. ResNet models are preferable in DML due to their reduced amount of parameters (Musgrave et al., 2020; Roth et al., 2020), although other DNN architectures have similar performance. The embedding layer's size is 128. We use ADAM (Kingma & Ba, 2015) with a learning rate[2] of 10^{-6}, weight decay of $4 \cdot 10^{-4}$ and the margin for triplet loss as 0.2.

Datasets. We use common metric learning datasets: (1) CUB200-2011 (Welinder et al., 2010) containing 11788 images of birds across 200 species; (2) CARS196 (Krause et al., 2013) containing 16185 images of cars across 196 different models; (3) SOP (Song et al., 2016) containing 120053 images of 22634 different online products. Each dataset has a training and testing set of approximately the same size. Similar to Roth et al. (2020), we train CUB200-2011 and CARS196 for 150 epochs and SOP for 100 epochs due to its volume.

Batches & Sampling. We use mini-batches of 112 data points. Each contains 2 samples-per-class (Roth et al., 2020), for every class present in the batch, and is the same for all training. We derive tuplet- and triplet-sets from the mini-batches. The triplet-set's size is the same as the mini-batch, ensuring that each data point is used as an anchor. The tuplet-set's size is double that of the mini-batch, thus balancing out the number of data points being compared relative to the triplet-set. Furthermore, each data point within the tuplet-set is used in a positive and negative pair. DML models are prone to collapse during training, a phenomenon caused by too many distant negative samples in the tuplet- or triplet-sets. In this situation, the model maps any data point to the same point in the embedding space. We use distance-weighted sampling for negative samples (Wu et al., 2017) to combat this issue.

Data Augmentation. We augment the dataset using the following operations for each input image: (1) random cropping to an image patch of size 60-100% of the original image area; (2) scaling; (3) normalization of pixel intensities. One difference is that our patch sizes differ from Roth et al. (2020) that employs patches of size 8-100% of original area. We change this parameter because recent work suggests that computer vision models can be biased by backgrounds and textures during during (Xiao et al., 2020). To combat this, we use cropping and scaling values based on Szegedy et al. (2015).

Attacks. We use Projected Gradient Descent (PGD) for assessing adversarial robustness (Madry et al., 2018). We use the notation $\text{PGD}_{i=5}$ to signal a PGD attack with five iterations ($i = 5$). Each attack use a step size of $2(\frac{\epsilon}{i})$, where ϵ is a specified threat model under the ℓ_∞-norm (Wong et al., 2020). To discover an adversarial perturbation δ (under the condition described in Section 3.2) for a data point $\mathbf{x}_i$ where $(\mathbf{x}_i, y_i) \in S$. Let $(x_j, y_j) = \arg\min_{(x,y)\in S} d_\theta(\mathbf{x}_i, \mathbf{x})$ be the nearest data point in the embedding space along with its class label. We then use PGD to approximate a solution for $\arg\max_{\delta \in B_\infty(0,\epsilon)} d_\theta(\mathbf{x}_i + \delta, x_j)$ if $i \neq j$ and $y_i = y_j$, for which the output is the adversarial perturbation for $\mathbf{x}_i$.

Evaluation Metrics. We use two DML-specific evaluation metrics proposed by Musgrave et al. (2020), namely: Recall at one (R@1) and Mean Average Precision at R (mAP@R). R@1 is the accuracy of using the embeddings that the model produces for inferring the output class label using the class of the nearest neighbor anchor. Given a test set $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$, R@1 is

[2]This learning rate differ from the one stated by Roth et al. (2020) in their publication, 10^{-5}, but reflects the actual learning rate used throughout their experiments. See the field "lr" within experiment configuration: https://bit.ly/3a4FyHP.

		CUB200-2011		CARS196		SOP	
	$\ell_\infty(\epsilon)$	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
Contrastive	0 (*B*)	58.9	21.9	72.5	20.9	66.0	39.1
	0.01	1.8	2.9	0.3	1.6	0.2	1.3
	0.05	0.0	2.4	0.1	1.6	0.0	1.2
	0.10	0.0	2.4	0.1	1.6	0.0	1.2
Triplet	0 (*B*)	58.8	21.2	72.9	21.3	69.1	41.4
	0.01	2.2	2.9	0.3	1.6	0.5	1.4
	0.05	0.0	2.5	0.1	1.6	0.0	1.2
	0.10	0.0	2.5	0.1	1.6	0.0	1.2

Table 1: Performance of naturally-trained DML models against $\text{PGD}_{i=5}$ attacks, with the proposed attack formulation, for various threat models $\ell_\infty(\epsilon)$. Recall that, R@1 reflects a model's inference accuracy, while mAP@R reflects its ability to rank similar entities. Naturally-trained DML models are not robust to the adversarial examples that our proposed technique generates.

given by:

$$\text{R@1} = \frac{1}{m}\sum_{i=1}^{m} \text{Prec}(i,1) \ \text{, where Prec}(i,k) = \frac{1}{k}\sum_{j \in I(k)} \mathbb{1}_{y_i = y_j}$$

$$I(k) = \underset{\substack{|K|=k \\ i \notin K}}{\arg\min} \sum_{j \in K} d_\theta(\mathbf{x}_i, \mathbf{x}_j)$$

mAP@R (Musgrave et al., 2020) measures a given model's ability to rank among classes in the embedding space; we adopted this metric for the reasons covered by Musgrave et al. (2020). It is defined as:

$$\text{mAP@R} = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{R(i)} \sum_{k=1}^{R(i)} \text{Prec}(i,k) \text{ , where } R(i) = \sum_{\substack{(x_j, y_j) \in S \\ i \neq j}} \mathbb{1}_{y_i = y_j}$$

4.2 Experimental Results

To examine the robustness of naturally-trained DML models, we apply the attack loss defined in Section 3 using $\text{PGD}_{i=5}$ across various threat models, where the adversarial perturbation δ satisfy $\|\delta\|_\infty \leq \epsilon$. We use five iterations for PGD because: (1) It quick saturates the maximum attack effect because R@1 ≈ 0; and (2) In this situation, adding more iterations leads to stagnation (Wong et al., 2020). We choose three threat models, $\epsilon \in \{0.01, 0.05, 0.1\}$, that occupy common values for attacks on computer vision.[3] Table 1 summarizes the above findings, with results averaged over five random seeds.

We observe that naturally-trained DML models achieve very low robustness to our proposed attack, experiencing a drop in both R@1 (inference) and mAP@R (ability to rank) by several orders of magnitude. Thus, to answer Q1, we conclude that DML models are not inherently robust to established attacks, suggesting that prior studies which report higher robustness of naturally-trained DML models is a potential sign of ineffective attacks instead (Abdelnabi et al., 2020).

Next, we examine the effectiveness of our robust training objective (from Section 3.4) in creating robust DML models. We set a threat model $\ell_\infty(\epsilon = 0.01)$ and optimize a min-max objective while re-using hyper-parameter choices from non-robust training. For the inner-maximization we use $\text{PGD}_{i=5}$. Table 2 summarizes the results across five attack rates $P(\lambda = 1)$.

We observe that our robust formulation does significantly improve model robustness against a strong PGD iterative attacker. We also observe a general trend of higher robustness with higher attack rates.

[3] https://www.robust-ml.org/defenses/

		CUB200-2011		CARS196		SOP	
	$P(\lambda = 1)$	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
Contrastive	0 (*B*)	1.8	2.9	0.3	1.6	0.2	1.3
	0.1	8.7	4.7	7.4	2.5	14.3	8.1
	0.25	14.1	6.3	25.1	7.1	30.9	18.1
	0.5	19.8	8.2	35.4	10.5	41.6	24.5
	0.75	21.0	8.9	39.7	11.8	44.9	26.0
	1.0	**26.5**	**10.6**	**40.5**	**11.9**	**46.0**	**26.1**
Triplet	0 (*B*)	2.2	2.9	0.3	1.6	0.5	1.4
	0.1	22.3	**8.8**	28.9	7.7	25.1	14.3
	0.25	21.5	8.4	33.3	9.1	34.0	19.1
	0.5	21.8	8.3	37.1	10.8	35.2	19.9
	0.75	**21.5**	8.4	39.9	11.4	**45.2**	**26.3**
	1.0	10.7	4.3	**40.5**	**11.7**	42.6	24.0

Table 2: Performance of robust DML models, trained using the proposed robust training objective with $\text{PGD}_{i=5}$ attacks $\ell_\infty(\epsilon = 0.01)$, across various configurations of the hyper-parameter attack rate ($P(\lambda = 1)$). Each model's inference accuracy (R@1) increase from the non-robust baseline (*B*), in addition to the models ability to rank similar entities (mAP@R). **Bold** marks best performance over dataset, metric, loss combinations. Thereby, the proposed robust training objective improves the robustness towards adversarial perturbations significantly.

CUB200-2011 with a triplet loss and attack rate of 1.0 displays subpar robustness. We hypothesize that this is due to the training instabilities that are inherent to the DML setting because the *entire* set of positive examples is being perturbed. As future work, we plan on investigating this relationship further. See Appendix A for more detailed discussion on this matter, along with the choice of perturbation target in the attack formulation. Finally, we note that the performance of the models on unperturbed natural data remains generally unaffected (see Appendix B, Table 3). Appendix B contains more details on robust training, including loss trajectories.

5 Conclusion

Deep Metric Learning (DML) creates feature embedding spaces where similar input points are geometrically close to each other, while dissimilar points are far apart. However, the underlying DNNs are vulnerable to adversarial inputs, thus making the DML models themselves vulnerable. We demonstrate that naturally-trained DML models are vulnerable to strong attackers, similar to other types of deep learning models. To create robust DML models, we contribute a robust training framework that can account for the *dependence* of metric losses — the phenomenon that the loss at any point depends on the other items in the mini-batch and the sampling process that was used to derive the mini-batch. Our robust training formulation yields robust DML models that can withstand powerful PGD attackers without severely degrading their performance on natural inputs.

References

Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS 2020, November 9-13, 2020*. ACM, 2020.

A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. *Lecture Notes in Computer Science*, pp. 387–402, 2013. ISSN 1611-3349. doi: 10.1007/978-3-642-40994-3_25. URL http://dx.doi.org/10.1007/978-3-642-40994-3_25.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *J. Mach. Learn. Res.*, 11:1109–1135, March 2010. ISSN 1532-4435.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.00482. URL http://dx.doi.org/10.1109/CVPR.2019.00482.

Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep Adversarial Metric Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2780–2789, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06. IEEE Computer Society, 2006. doi: 10.1109/CVPR.2006.100. URL https://doi.org/10.1109/CVPR.2006.100.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *ArXiv*, abs/1703.07737, 2017.

Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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.

Pengcheng Li, Jinfeng Yi, Bowen Zhou, and Lijun Zhang. Improving the robustness of deep neural networks via adversarial training with triplet loss. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2909–2915. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/403. URL https://doi.org/10.24963/ijcai.2019/403.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness, 2019.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check, 2020.

T. K. Panum, K. D. Hageman, R. R. Hansen, and J. M. Pedersen. Towards Adversarial Phishing Detection, 2020.

Karsten Roth, Biagio Brattoli, , and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. doi: 10.1109/iccv.2019.00809. URL http://dx.doi.org/10.1109/ICCV.2019.00809.

Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. Revisiting Training Strategies and Generalization Performance in Deep Metric Learning, 2020.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2612–2620, 2017.

Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.00516. URL http://dx.doi.org/10.1109/CVPR.2019.00516.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. doi: 10.1109/iccv.2017.309. URL http://dx.doi.org/10.1109/ICCV.2017.309.

Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition, 2020.

A Discussion

Wu et al. (2017) covered an instability of metric losses that can arise, in the search for more efficient training, and is caused by sampling (hard) negative data points near the anchor point. Following this, the introduced distance weighted sampling technique seeks to provide a more efficient distribution of negative data points, as data points are transform into tuples and triplets, such that this instability is less like to occur. Seeing the instabilities that can occur in the presented adversarial training scenarios of metric losses, these stability measures are not directly applicable, as the adversarial perturbations used for the robust training objective are relative to the individual perturbed tuples or triplets. Thereby, it remains an open question if (and how) adversarial perturbations should be included prior to sampling, to gain stability from sampling, or if post-sampling measures to stabilize weight gradients is more ideal.

In the formulation proposed in Section 3, we chose to have the positive data point be the target for the adversarial perturbation, to ensure that relationships across anchors and negatives remain unchanged, due to stability issues previously mentioned. To further investigate this choice, we conducted a set of experiments included in Appendix C, comparing the training process and performance across the three types of data point undergoing perturbation. For most scenarios we find the proposed formulation performs the best, however, the results are inconclusive.

		CUB200-2011		**CARS196**		**SOP**	
	$P(\lambda = 1)$	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
Contrastive	0 (*B*)	58.9	21.9	72.5	20.9	66.0	39.1
	0.1	41.1	11.4	72.3	18.6	**64.2**	**37.1**
	0.25	56.9	**19.5**	**73.3**	**20.6**	**64.2**	36.9
	0.5	**55.4**	17.6	71.6	19.1	61.5	34.2
	0.75	53.8	17.3	70.7	18.1	60.0	32.9
	1.0	*52.5*	*16.2*	*68.4*	*17.0*	*58.3*	*31.2*
Triplet	0 (*B*)	58.8	21.2	72.9	21.3	69.1	41.4
	0.1	51.9	*16.8*	70.7	18.2	**66.0**	**38.2**
	0.25	**52.7**	**17.1**	70.0	17.2	65.2	37.2
	0.5	51.1	16.3	**71.4**	**18.6**	64.4	36.4
	0.75	*51.7*	16.7	70.9	17.7	*61.3*	*33.9*
	1.0	49.1	14.0	*69.2*	*17.0*	57.9	30.7

Table 3: Performance of robust DML models, trained using the proposed robust training objective with $\text{PGD}_{i=5}$ attacks $\ell_\infty(\epsilon = 0.01)$, across various configurations of the hyper-parameter attack rate ($P(\lambda = 1)$). Each model's inference accuracy (R@1), and ability to rank similar entities (mAP@R), are near the baseline performance (*B*) of a non-robustly trained DML model. **Bold** marks best performance over dataset, metric, loss combinations. *Italic* marks the configuration for robustness against adversarial perturbations over dataset, metric, loss combinations. Thereby, the proposed robust training objective does not significantly alter a model's ability to perform on benign input.

B Experiment Details: Attack Rates

This section expands upon the training process and performance of the DML models trained using the proposed robust formulation. In Table 3, the performance for these models against benign (unperturbed) input can be seen. Importantly, the baselines (*B*) reflect naturally-trained DML models (as effectively no data points are being perturbed during training when $P(\lambda = 1) = 0$). Generally, it can be seen that the robust formulation has minor impact on performance on benign, as each model's model evaluation metrics are close to performance of the respective baseline. Interestingly, for contrastive loss on CARS196 with $P(\lambda = 1) = 0.25$, the performance even exceeds the baseline from 72.5 (R@1) to 73.3 (R@1).

As mention in Section 4.2, certain instabilities of the training procedure caused certain models to drop in performance as attack rates increased (seen in Table 2). For clarity, we provide more details about the training procedure with Figure 1 and Figure 2. These figures cover the training process for contrastive loss and triplet loss, respectively.

C Experiment: Effect of Perturbation Target

For the proposed formulation of adversarial training for DML models using metric losses, covered in Section 3.4, we choose to perturb the positive data point within the losses. Each of the types of data points (anchor, positive, negative) can be mapped to corresponding relationships between data point. Perturbing anchor would effectively be be joint maximizing between positive and negative relationships, while perturbing the other types (positive and negative) would only affect a single type of data point relationship, namely either the positive or negative relationship (respective to a designated anchor point). Considering this, we choose to attack the positive data point within the losses, as large distances for negative relationships (effect of attack) for the training objective of metric losses have been demonstrated to cause instabilities (Wu et al., 2017). However, as a measure to establish clarity experimented with perturbing other data points and the training process for these experiments can be seen in Figure 3 and Figure 4, for contrastive and triplet loss respectively. Losses for the positive, as perturbation target, seems yield more consistent reductions in loss as training progresses. However, this does not necessarily cause the trained models to obtain higher accuracy

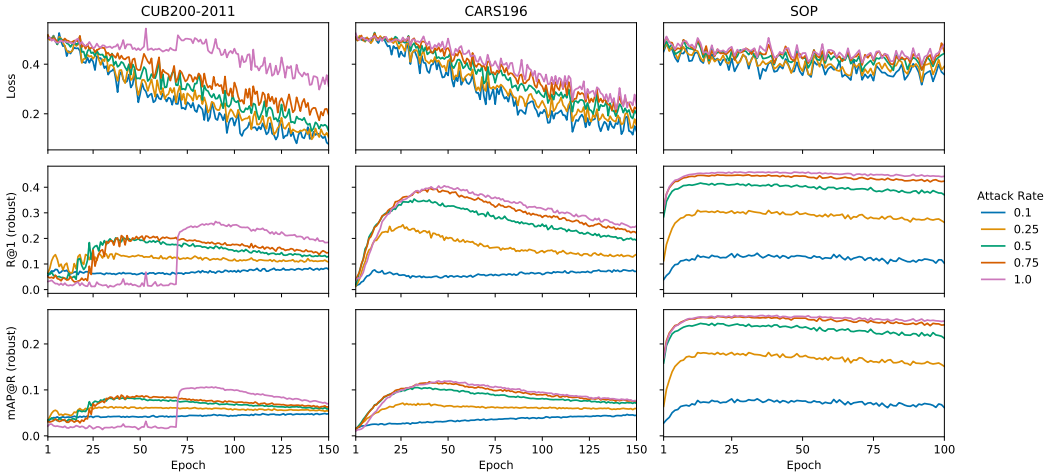

Figure 1: Contrastive loss training procedure (loss, R@1, and mAP@R) across attack rates ($P(\lambda = 1)$. Robust training process for contrastive loss across varying attack rates and datasets. $P(\lambda = 1) = 1$ for CUB200-2011 clearly show case the mentioned instabilities, by seen the fluctuation happening at approx. epoch 70.

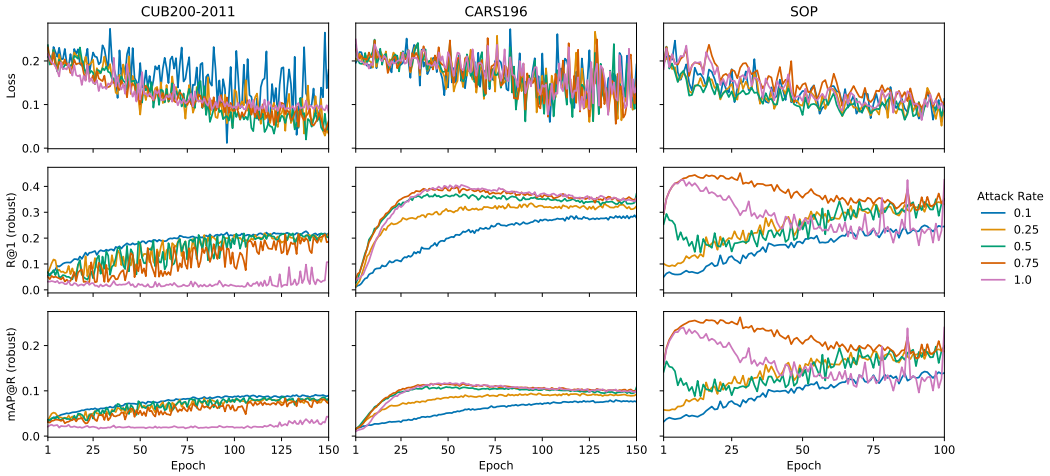

Figure 2: Triplet loss training procedure (loss, R@1, and mAP@R) across attack rates ($P(\lambda = 1)$. Robust training process for contrastive loss across varying attack rates and datasets. Smaller attack rates seem to be associated with more stability, with the cost of not necessarily reaching maximum obtainable robustness.

(R@1) or ability to rank (mAP@R). Thereby, these results to be inconclusive, in respect to the optimal choice for perturbation target.

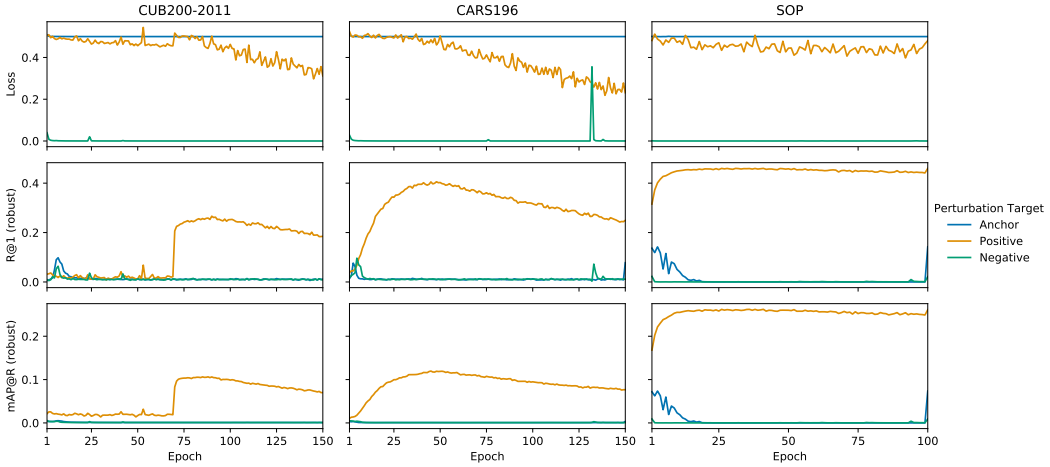

Figure 3: Contrastive loss training procedure (loss, R@1, and mAP@R) across perturbation targets for an attack rate of $P(\lambda = 1) = 1$. Losses across perturbation targets, other than positive, does not significantly decrease as training progress, suggesting that the training has reached a local minima.

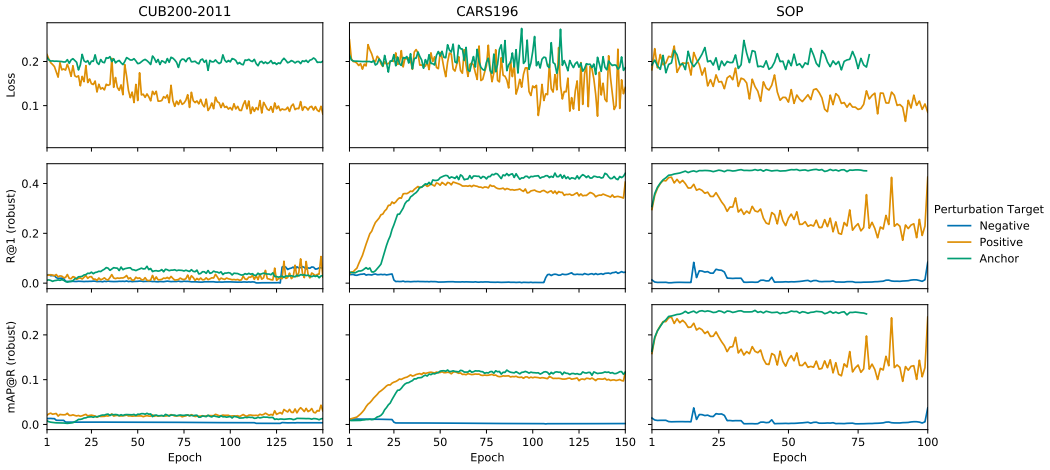

Figure 4: Triplet loss training procedure (loss, R@1, and mAP@R) across perturbation targets for an attack rate of $P(\lambda = 1) = 1$. SOP with the anchor point as perturbation target were unable to finalize training (reach 100 epochs) within the time of submission deadline, due to limitations of resource availability and the training time spanning more than three days.