EXTREME TRIPLET LEARNING: EFFECTIVELY OPTIMIZING EASY POSITIVES AND HARD NEGATIVES

Anonymous authors
Paper under double-blind review

ABSTRACT

The Triplet Loss approach to Distance Metric Learning is defined by the strategy to select triplets and the loss function through which those triplets are optimized. During optimization, two especially important cases are easy positive and hard negative mining which consider, the closest example of the same and different classes. We characterize how triplets behave based during optimization as a function of these similarities, and highlight that these important cases have technical problems where standard gradient descent behaves poorly, pulling the negative example closer and/or pushing the positive example farther away. We derive an updated loss function that fixes these problems and shows improvements to the state of the art for CUB, CAR, SOP, In-Shop Clothes datasets.

1 INTRODUCTION

Deep metric learning optimizes an embedding function that maps semantically similar images to relatively nearby locations and maps semantically dissimilar images to distant locations. A number of approaches have been proposed for this problem (Schroff et al., 2015a; Sohn, 2016; Movshovitz-Attias et al., 2017; Song et al., 2016; Xuan et al., 2018; Kim et al., 2018; Ge, 2018). A common way to learn the mapping is to define a loss function based on triplets of images: an anchor image, a positive image from the same class, and a negative image from a different class. The loss penalizes cases where the anchor is mapped closer to the negative image than it is to the positive image.

One common variant of these functions (e.g. (Wang et al., 2018; Sohn, 2016)), uses a Deep Learning framework to map images to a feature vector, and computes similarity between normalized feature vectors based on the dot-product. This approach forces the features to lie on a hypersphere and has advantages of making the feature comparison intuitive and efficient.

In this work we explore standard implementations of these loss functions and show there are two problems. First, when the gradient of the loss function does not consider the normalization to a hypersphere, a large part of the gradient is lost when points are re-projected back to the sphere, especially in the easy-positive/hard-negative cases of triplets including nearby points. Second, when optimizing the parameters (the weights) of the network, when points are already mapped close together, it may be difficult to find gradient directions that effectively separate nearby images.

We give systematic derivation showing when and where these challenging triplets arise, and diagram the sets of triplets where standard gradient descent makes the loss increase. We find that this explains problems previously reported in the literature such as the difficulty in optimizing hard-negative triplets (Harwood et al., 2017). Furthermore, these problems are mostly introduced because the loss-function of the triplets is based on the differences between the anchor-positive and anchor-negative distances, so there is an equivalent effect of encouraging the positive image to be closer or the negative image to be further. We create a new loss function that breaks this symmetry and weights the importance of changing the anchor-positive and anchor-negative distances. Briefly, our main contributions are:

- A systematic characterization of triplet selection strategies and a visualization that highlights regions of bad gradient behavior.
- A simple modification to a standard loss function to fix bad gradient behavior.
- Improvements to current state of the art results across a range of datasets.

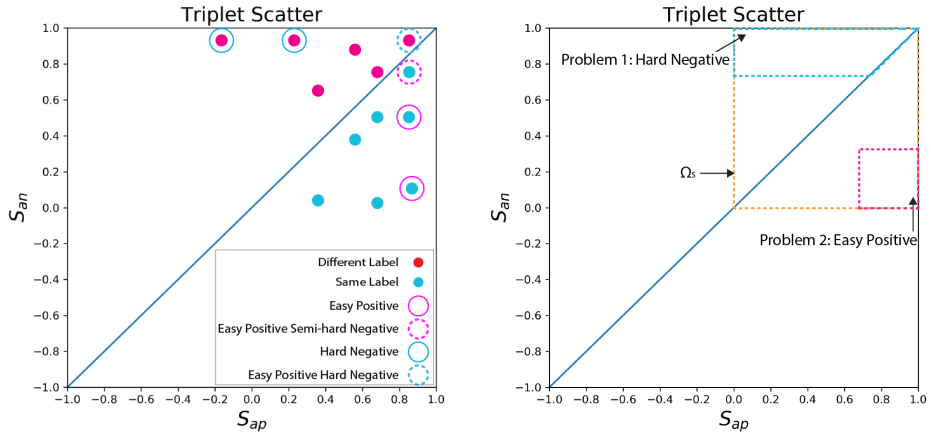

Figure 1: The triplet scatter diagram plots a triplet as a point defined by the Anchor-Positive similarity S_{ap} and the Anchor-Negative similarity S_{an}. (left) Points below the diagonal correspond to triplets that are "correct" in the sense that the anchor image could get the correct label because the same class example is closer than the different class example. Triplets along the top of the diagram are candidates for hard-negative mining, triplets along the right edge are candidates for easy-positive mining, and triplets mapped near but below the diagonal are candidates for semi-hard triplet mining. (right) Two important cases are the extremes of the hardest negatives and the easiest-positives.

2 Background

There is a large body of work in distance metric learning and they are leading with two main ideas. One of the idea is to increase the intra-class cluster density and keep the inter-class clusters as far as possible. Pairwise loss functions like contrastive divergence (Carreira-Perpinan & Hinton, 2005; Chopra et al., 2005; Radenović et al., 2016) directly optimize for this constraint, and "No fuss metric learning" (Movshovitz-Attias et al., 2017), implicitly optimizes for this constraint by assigning each class to different location and penalizes the failure of any example to go to its assigned location.

The other approach more directly reflects the fact that for image retrieval applications, it is not necessary for all elements of a class to be clustered, but instead that the distance to elements of the same class should be smaller than the distance to elements of different classes. Directly optimizing for this is based on triplets of images. The triplets are drawn from the training data and include an anchor image, a positive image from the same class, and a negative image from a different class.

Key questions for these approaches explore how to select the triplets that are used. Choices such as hard or semi-hard triplet mining (Schroff et al., 2015b; Simo-Serra et al., 2015; Wang et al., 2014) focus on triplets with negative examples that are closest (hard negative mining) or nearly as close to the anchor as positive images (semi-hard negative mining) and emphasize creating separations between classes in the embedding space. Recent work such as easy positive triplet mining (Xuan et al., 2019) selects the closest anchor-positive pairs and ensures that at least they are closer than the nearest negatives.

The next section introduces a diagram to systematically organize these triplet selection approaches, and to explore where different loss functions fail to improve the triplets.

3 Triplet Scatter Diagram

Triplet loss is trained with triplets of images, (x_a, x_p, x_n), where x_a is an anchor image, x_p is a positive image of the same class as the anchor, and x_n is a negative image of a different class. We consider a convolution neural network, $f(\cdot)$ that embeds the images on a unit hypersphere, $(\mathbf{f}(\mathbf{x_a}), \mathbf{f}(\mathbf{x_p}), \mathbf{f}(\mathbf{x_n}))$. We use $(\mathbf{f_a}, \mathbf{f_p}, \mathbf{f_n})$ to simplify the representation of the normalized feature vectors. When embedded on a hypersphere, the cosine similarity is a convenient metric to measure

the similarity of anchor-positive pair $S_{ap} = \mathbf{f_a^T f_p}$ and anchor-negative pair $S_{an} = \mathbf{f_a^T f_n}$, and this similarity is bounded in the range $[-1, 1]$.

The triplet scatter diagram is an approach to characterizing a given set of triplets. Figure 1 represents each triplet as a 2D point (S_{ap}, S_{an}), describing how similar the positive and negative images are to the anchor. This diagram is useful because the location on the diagram describes important features of the triplet:

- Triplets that are already in the correct configuration, where the similarity between anchor and positive is greater than the similarity between anchor and negative images are below the $S_{an} = S_{ap}$ diagonal. Dots representing triplets in the correct configuration are drawn in blue, dots where the negative is closer are drawn in red.
- Triplets that include an anchor and the most similar of the possible positive examples are the "Easy Positives" and are on the right side of the diagram because S_{ap} tends to be close to 1. We circle these with a red ring.
- Hard negatives are cases where the anchor is very similar to a negative example, so S_{an} is close to 1, depicted as red dots circled with a blue ring.
- Hard negatives are cases where the anchor is very similar to a negative example, so S_{an} is close to 1, depicted as red dots circled with a blue ring.
- One very selective mining strategy is "Easy-Positive, Semi-Hard Negative", where an anchor is matched with closest possible positive match, and a negative example which has a similar similarity. The blue dot circled with red dashed circle highlights one such example.
- Another selective mining strategy is "Easy-Positive, Hard Negative", which selects, for an anchor, the most similar positive and negative examples. The red circle surrounded by a blue dashed circle represents one such example.

During the later discussion, we may show a subset area of Ω, $\Omega_s = [0, 1] \times [0, 1]$, because it is rare that the hardest negative or positive pairs have a similarity less than 0.

Figure 1 (right) calls out two specific regions of points that we analyze in the next section; the extremes of hard negatives and easy positives, and the region that only includes positive similarities $\Omega_s = [0, 1] \times [0, 1]$, that includes nearly all triplets constructed with easy positives and hard negatives.

4 DIAGRAMMING WHY SOME TRIPLETS ARE HARD TO OPTIMIZE

The triplet scatter diagram offers the ability to understand when the gradient based optimization of the network parameters is effective and when it fails. The triplets are used to train a network whose loss function encourages the anchor to be more similar to its positive example (drawn from the same class) than to its negative example (drawn from a different class), encouraging S_{ap} to be greater than S_{an}. While there are several possible choices, we consider NCA (Goldberger et al., 2005) as the loss function, and denote this as L^{1st} to differentiate it from an updated loss function introduced later:

$$L^{1st}(\mathbf{f_a}, \mathbf{f_p}, \mathbf{f_n}) = -log\frac{\exp(S_{ap})}{\exp(S_{ap}) + \exp(S_{an})} \tag{1}$$

All of the following derivation can also be done for the triplet loss formulation used in (Schroff et al., 2015a); this has a very similar form and is derived in the Appendix.

The gradient of triplets loss $L^{1st}(\mathbf{f_a}, \mathbf{f_p}, \mathbf{f_n})$ can be decomposed into two parts: a single gradient with respect to feature vectors $\mathbf{f_a}$, $\mathbf{f_p}$, $\mathbf{f_n}$:

$$\Delta L = \frac{\partial L}{\partial \mathbf{f_a}}\Delta \mathbf{f_a} + \frac{\partial L}{\partial \mathbf{f_p}}\Delta \mathbf{f_p} + \frac{\partial L}{\partial \mathbf{f_n}}\Delta \mathbf{f_n}, \tag{2}$$

and subsequently being clear that these feature vectors respond to changes in the model parameters (the CNN network weights) θ:

$$\Delta L = \frac{\partial L}{\partial \mathbf{f_a}}\frac{\partial \mathbf{f_a}}{\partial \theta}\Delta\theta + \frac{\partial L}{\partial \mathbf{f_p}}\frac{\partial \mathbf{f_p}}{\partial \theta}\Delta\theta + \frac{\partial L}{\partial \mathbf{f_n}}\frac{\partial \mathbf{f_n}}{\partial \theta}\Delta\theta. \tag{3}$$

The gradient optimization only affects the feature embedding through variations in θ, but we first highlight problems with hypersphere embedding assuming that the optimization *could* directly affect the embedding location. To do this we derive the loss gradient with respect to the feature vector $\mathbf{f_a}$, $\mathbf{f_p}$, $\mathbf{f_n}$ and use this gradient to update the feature locations where should decrease the error:

$$\mathbf{f_p}^{new} = \mathbf{f_p} - \alpha \mathbf{g_p} = \mathbf{f_p} - \alpha \frac{\partial L}{\partial \mathbf{f_p}} = \mathbf{f_p} + \beta \mathbf{f_a} \tag{4}$$

$$\mathbf{f_n}^{new} = \mathbf{f_n} - \alpha \mathbf{g_n} = \mathbf{f_n} - \alpha \frac{\partial L}{\partial \mathbf{f_n}} = \mathbf{f_n} - \beta \mathbf{f_a} \tag{5}$$

$$\mathbf{f_a}^{new} = \mathbf{f_a} - \alpha \mathbf{g_a} = \mathbf{f_a} - \alpha \frac{\partial L}{\partial \mathbf{f_a}} = \mathbf{f_a} - \beta \mathbf{f_n} + \beta \mathbf{f_p} \tag{6}$$

where $\beta = \alpha \frac{\exp(S_{an})}{\exp(S_{ap}) + \exp(S_{an})}$ and α is the learning rate. This gradient update has a clear geometric meaning: the positive point $\mathbf{f_p}$ is encouraged to move along the direction of the vector $\mathbf{f_a}$; the negative point $\mathbf{f_n}$ is encouraged to move along the opposite direction of the vector $\mathbf{f_a}$; the anchor point $\mathbf{f_a}$ is encouraged to move along the direction of the sum of $\mathbf{f_p}$ and negative $\mathbf{f_n}$. All of these are weighted by the same weighting factor β. Then we can get the new similarity of anchor-positive and anchor-negative (The derivation is given in the Appendix):

$$S_{ap}^{new} = (1 + \beta^2) S_{ap} + 2\beta - \beta S_{pn} - \beta^2 S_{an} \tag{7}$$

$$S_{an}^{new} = (1 + \beta^2) S_{an} - 2\beta + \beta S_{pn} - \beta^2 S_{ap} \tag{8}$$

These gradients $\mathbf{g_a}$, $\mathbf{g_p}$, $\mathbf{g_n}$ have components that move them off the sphere, computing the cosine similarity requires that we compute the norm of $\mathbf{f_a}^{new}$, $\mathbf{f_p}^{new}$ and $\mathbf{f_n}^{new}$(the derivation for them is shown in Appendix). Given the norm of updated feature vector, we can calculate the similarity change after the gradient update.

$$\Delta S_{ap} = \frac{S_{ap}^{new}}{\|\mathbf{f_a}^{new}\| \|\mathbf{f_p}^{new}\|} - S_{ap} \tag{9}$$

$$\Delta S_{an} = \frac{S_{an}^{new}}{\|\mathbf{f_a}^{new}\| \|\mathbf{f_n}^{new}\|} - S_{an} \tag{10}$$

Because the gradient of the loss function does not consider this normalization, following the negative gradient can actually cause some triplets to push the anchor closer to the negative example or push the anchor away from the positive example, even assuming that you can directly push the anchor, positive and negative features vectors in any direction.

The direction in which $\mathbf{f_a}$, $\mathbf{f_p}$, and $\mathbf{f_n}$ move depends on the relative position of the $\mathbf{f_a}, \mathbf{f_p}, \mathbf{f_n}$ on the hypersphere. We use γ (fully defined in the Appendix) as a term to describe their relative orientation; when $\mathbf{f_a}$, $\mathbf{f_n}$, and $\mathbf{f_p}$ are close enough so that locally the hypersphere is a plane, then γ is the dot-product of normalized vector from $\mathbf{f_a}$ to $\mathbf{f_p}$ and $\mathbf{f_a}$ to $\mathbf{f_n}$. Therefore, if $\mathbf{f_p}, \mathbf{f_a}, \mathbf{f_n}$ are co-planer then $\gamma = 1$, and if moving from $\mathbf{f_a}$ to $\mathbf{f_p}$ is orthogonal to the direction from $\mathbf{f_a}$ to $\mathbf{f_n}$, then $\gamma = 0$. Given this description of the relative positions of the anchor, positive and negative points, Figure 2 shows calculations of the change in similarity between the anchor positive and anchor negative for $\gamma = 0.5$. There is an area along the right side of the ΔS_{ap} plot highlighting locations where the anchor and positive are pushed farther apart($\Delta S_{ap} < 0$), and along the top of the ΔS_{an} plot highlighting locations where the anchor and negative are pulled closer together($\Delta S_{an} > 0$). This behavior arises because the gradient is pushing the feature off the hypersphere and therefore, after normalization, the effect is lost.

This discussion so far considers the derivative of the loss as a function of the position of the feature vectors, but the optimization can only control the feature vectors based on the network parameters θ. Changes the θ are likely to affect **nearby** points in similar ways. For example, if there is a hard negative example with easy positive where the anchor is close to both the positive and the negative image, then changing θ to move the anchor closer to the positive is likely to pull the negative example along with it. We call this effect "entanglement" and propose a simple model to capture its effect on how the gradient update affects the similarities.

We use a scalar p and a factor $q = \sqrt{S_{ap} S_{an}}$ to quantify this entanglement. As for the factor q, when anchor, positive and negative are nearby to each other, both S_{ap} and S_{an} will be large and q will increase the entanglement effect; when positive or negative is far away to anchor, one of S_{ap}

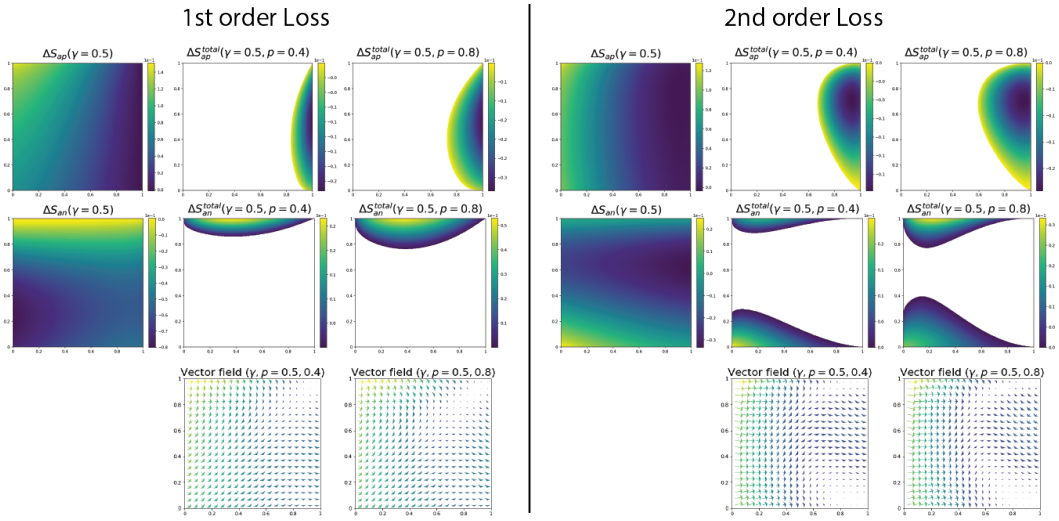

Figure 2: First two rows: numerical simulation for ΔS_{ap}, ΔS_{an}, ΔS_{ap}^{total}, ΔS_{an}^{total} in equation 9, 10, 11 and 12 of $\gamma = 0.5$ and $p = 0.4, 0.8$. Thrid row: Vector field of $\gamma = 0.5$ and $p = 0.4, 0.8$.

and S_{an} will be small and q will reduce the entanglement effect. The total similarity changes with entanglement will modelled as follows:

$$\Delta S_{ap}^{total} = \Delta S_{ap} + p\sqrt{S_{ap}S_{an}}\Delta S_{an} \tag{11}$$

$$\Delta S_{an}^{total} = \Delta S_{an} + p\sqrt{S_{ap}S_{an}}\Delta S_{ap} \tag{12}$$

Figure 2 shows (in color) problematic regions where this model of gradient entanglement indicates that anchor and positive images become less similar ($S_{ap} < 0$) and regions where the anchor negative images become more similar for different parameters of the entanglement ($S_{an} > 0$).

While figure 2 captures problematic regions on the scatter diagram, we can create a more complete description. The bottom row plots of figure2 shows the vector field on the scatter diagram, indicating that triplets move based on the gradient of their loss function. The figure shows several vector field plots with ΔS_{ap}^{total} and ΔS_{an}^{total} with $\gamma = 0.5$ and $p = 0.4, 0.8$ settings and indicates the 3 types of movement direction for dots on the triplet scatter. When $\Delta S_{ap} > 0$ and $\Delta S_{an} > 0$, the movement direction will point to up-right. When $\Delta S_{ap} < 0$ and $\Delta S_{an} < 0$, the movement direction will point to bottom-left. When $\Delta S_{ap} > 0$ and $\Delta S_{an} > 0$, the movement direction will point to bottom-right. In fact, the entanglement strength may be varied in different situation during the optimization. The exact value will not be discussed in this paper. We only use the entanglement phenomenon to demonstrate problems that happen optimizing triplets with Easy Positives and Hard Negatives.

Problem 1: Hard Negative Mining For a given anchor image, hard negative mining chooses the negative example that maximizes S_{an}. The vector field computed in the case of entanglement shows that most locations with large S_{an} (near the top of the plot) have vectors with an upward component, meaning the gradient update for a hard negative triplet will push the negative even closer to anchor. The result is that a network cannot effectively separate the negative pairs and tending to make all points close to each other. Initializing a fine-grained visualization tasks (e.g. CARS196) with a generic pre-trained network like ImageNet often creates a starting conditions where all points are mapped nearby to begin with, leading to an optimization failure where all triplets move towards (1,1) on the triplet scatter diagram and all images are mapped to the same feature location.

Problem 2: Easy Positive Mining For a given anchor image, easy positive mining chooses the positive example that maximizes S_{ap}. The vector field computed in the case of entanglement shows that most locations with large S_{ap} (near the right of the plot) have vectors with a strong downward component, meaning the gradient updates pushes the anchor and negative to be negative related, which leads over-training. A general idea for a pair of images to be different is their similarity to be zero, the negative similarity value still means the pair of images are a kind of 'related'. In the later

phase of the optimization, as the optimization proceeds the triplet scatter diagram will not effectively keep triplets close to the ideal (1,0) point.

5 WEIGHT GRADIENT BY SIMILARITY

Triplets that have problem 1 or problem 2 create gradients that move $\mathbf{f_a}$, $\mathbf{f_p}$, $\mathbf{f_n}$ in wrong directions. When the anchor is very close to either the positive or the negative points, the gradients defined by their interaction is largely lost because of the hypersphere normalization, and the remaining effective gradient is dominated by the entanglement.

Specifically, for problem 1, the anchor and the negative image are close together, and not effectively encouraged to move apart, and pull of the anchor image towards the positive pulls the negative image in the same direction. The fix for this problem is to emphasize more the part of gradient pushing the anchor and negative apart when S_{an} is close to 1.

For problem 2, the anchor and positive image are close together, and there is a more distant negative example. The effect of the distant negative example may push the anchor and positive example further apart. The fix for this problem is to emphasize decrease the weight of the gradient related to the distant negative example S_{ap} is close to 1.

A simple weighting strategy addresses both problems. We weight the $\mathbf{g_p}$ and $\mathbf{g_n}$ gradients with scalars $w_{ap} = 1 - S_{ap}$ and $w_{an} = S_{an}$ respectively. When a negative pair in a triplet is close but the positive in the triplet is relative distant to anchor, we want to emphasize pushing the negative pair apart, so we weight $\mathbf{g_p}$ by $1 - S_{ap}$. Similarly, when a positive pair in a triplet is close but the negative in the triplet is distant to anchor, we want to decrease the effect of the negative example, then we weight $\mathbf{g_n}$ with S_{an}. Our new gradients have the following form:

$$\alpha {\mathbf{g_p}}^w = -\beta_w w_{ap} \mathbf{f_a} \tag{13}$$

$$\alpha {\mathbf{g_n}}^w = \beta_w w_{an} \mathbf{f_a} \tag{14}$$

$$\alpha {\mathbf{g_a}}^w = \beta_w w_{an} \mathbf{f_n} - \beta_w w_{ap} \mathbf{f_p} \tag{15}$$

where $\beta_w = \alpha \frac{\exp(\frac{1}{2} S_{an}^2)}{\exp(S_{ap} - \frac{1}{2} S_{ap}^2) + \exp(\frac{1}{2} S_{an}^2)}$ These can be integrated to find a loss function with these gradients. We call this a 2nd-order triplet loss because in the exponent the similarities are squared.

$$L^{2nd}(\mathbf{f_a}, \mathbf{f_p}, \mathbf{f_n}) = -log \frac{\exp(S_{ap} - \frac{1}{2} S_{ap}^2)}{\exp(S_{ap} - \frac{1}{2} S_{ap}^2) + \exp(\frac{1}{2} S_{an}^2)}, \tag{16}$$

The calculation of the terms needed to solve for ΔS_{ap} or ΔS_{an} for the new loss function in the Appendix, and Figure 2 (the right column) shows that how vector field for the new loss function. In the area near (1,1), there is no longer the challenge that triplets are always moved towards (1,1) even when including the effects of entanglement. This helps overcome the problem that the optimization pushes all points towards the same place and we will show this effect in the experimental section.

6 EXPERIMENTAL RESULTS

We to run a set of experiments on CUB200 (Welinder et al., 2010), CAR196 (Krause et al., 2013), Stanford online products (Song et al., 2016) and In-shop cloth (Ziwei Liu & Tang, 2016) datasets. All tests are run on the PyTorch platform (Paszke et al., 2017), using ResNet18 and ResNet50 (He et al., 2016) architectures, pre-trained on ILSVRC 2012-CLS data (Russakovsky et al., 2015). Training images are re-sized to 256 by 256 pixels. We adopt a standard data augmentation scheme (random horizontal flip and random crops padded by 10 pixels on each side). For pre-processing, we normalize the images using the channel means and standard deviations. All networks are trained using stochastic gradient descent (SGD) with 40 epochs. We set initial learning rate 0.005 for CAR, SOP and In-shop cloth dataset and 0.0025 for CUB dataset, and divided by 10 after 20^{th} and 30^{th} epochs. The batch size is 128 and each batch of images contains n examples from c classes, randomly selected from the training data. Throughout the paper, we refer to the n examples per class as a group.

Comparing the performance of 1st order and 2nd order triplet loss **only requires changing one line of code** in a PyTorch implementation, substituting the loss from Equation 16 for the loss from

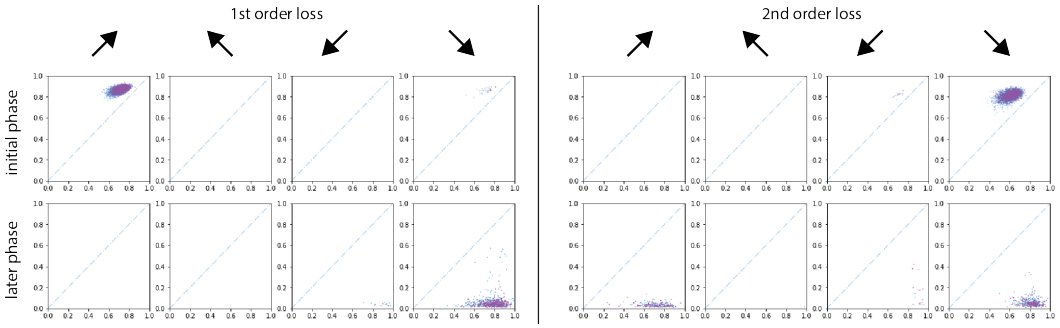

Figure 3: The locations of dots moves top-right, top-left, bottom-left and bottom-right.The blue color represents the initial phase iterations and the red color represents the later phase iterations

Equation 1. Then we calculate Recall@K as the measurement for retrieval quality. In the CUB, CAR and SOP datasets, both the query set and gallery set refer to the testing set. During the query process, the top-K retrieved images exclude the query image itself. In the In-Shop dataset, the query set and gallery set are predefined by the original paper.

6.1 Verification of vector field

We implement two variants from EPHN paper (Xuan et al., 2019), Hard-Positive with Hard-Negative(HPHN) to mimic the Problem 1 and Easy-Positive with Easy-Negative(EPEH) to mimic the Problem 2 with CAR dataset and set $n = 16$ so that there will be enough positive and negative for easy or hard selecting. Also, in the batch, we only allow each feature vector can only be in one triplet to prevent the affection among the triplet. Figure 3 shows two sets experiment results of training a network with 2 epochs(73 batches per epoch) at initial phase and later phase of the optimization. We set the learning rate very low(0.00001) so as to prevent the model updating too much in this experiment and better observe the problem happen in the wrong area.

We plot where the triplet dot move to all possible 4 direction(top-right, top-left, bottom-left and bottom-right). This result highly match our proposal model. 1) there is no top-left movement which is not possible in our model. 2) In the initial phase, most of the movement for the L^{1st} is top-right and for the L^{2nd} is bottom-right, which fit the our proposal fix to the vector field 3) In the later phase, most of the movement for the L^{1st} is bottom-right and for the L^{2nd} is bottom-right, bottom-left and top-right. 4Although the learning rate is small, there is still a patter change with the dot in early phase. L^{1st} move to top-right and L^{2nd} move to bottom-right.

6.2 Comparative Results for EPHN

Hard-Negative mining optimization problem depicts the specific cause of the challenges reported (for example (Harwood et al., 2017)) for optimizing with hard negative examples. Also, in the EPHN work (Xuan et al., 2019), when $n = 2$, there is only one positive example to choose from, so S_{ap} is more likely to be far away to 1. The negative is chosen as the closest negative, so S_{an} is more likely to be close to 1. This situation is similar to Problem 1. The L^{1st} loss leads to a singularity at the location (1,1).

In addition, we re-implement the test in EPHN work to plot the recall@1 accuracy versus n across CUB, CAR, SOP, and In-shop dataset. Figure 4 shows a clear gap of L^{1st} and L^{2nd} with EPHN mining. We hypothesize that the small number of examples in most classes in SOP leads to the inconsistent behavior we observe. Moreover, the figure shows Problem 1 happened on the CUB when $n = 2$ and CAR when $n = 2, 4$, but L^{2nd} loss perform well on with the same setting.

6.3 Comparison: EPHN vs EPSHN

Semi-hard negative mining samples triplets choosing the most similar negative where $S_{ap} > S_{an}$. It is therefore most affected by the Problem 2. We compare the result of easy positive and semi-hard

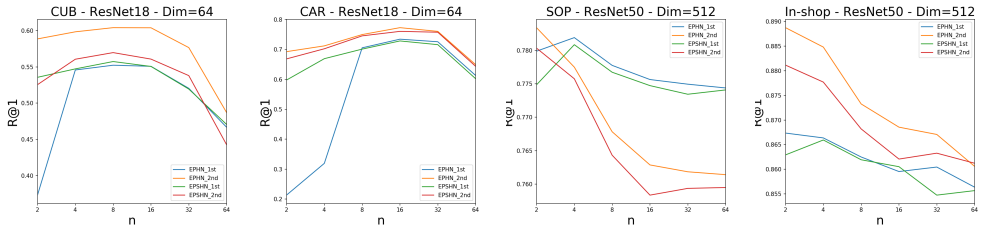

Figure 4: Recall@1 accuracy vs group size n on CUB, CAR, SOP and In-shop datasets with 1st/2nd order EPHN/EPSHN

Dataset	CUB			CAR			SOP			In-shop		
Method	R@1	R@2	R@4	R@1	R@2	R@4	R@1	R@10	R@100	R@1	R@10	R@20
HTL^{512}	57.1	68.8	78.7	81.4	88.0	92.7	74.8	88.3	94.8	-	-	-
ABE^{512}	60.6	71.5	79.8	85.2	90.5	94.0	76.3	88.4	94.8	87.3	96.7	97.9
DREML^{576}	63.9	75.0	83.1	**86.0**	**91.7**	**95.0**	-	-	-	-	-	-
FastAP^{512}	-	-	-	-	-	-	76.4	89.0	95.1	**90.9**	**97.7**	**98.5**
EPHN-1st^{512}	64.9	75.3	**83.5**	82.7	89.3	93.0	78.0	90.6	96.3	87.1	96.9	97.9
EPHN-2nd^{512}	**65.2**	**75.3**	82.9	83.2	89.2	93.2	**78.8**	**90.8**	**96.3**	89.0	96.9	97.8

Table 1: Retrieval Performance on the CUB, CAR, SOP and In-shop datasets comparing to the best reported results for more complex approaches and/or ensembles. All test are trained with ResNet50

negative mining (EPSHN) and easy-positive hard-negative mining (EPHN) for both 1st order and 2nd order loss functions on CUB, CAR, SOP and In-shop dataset. Figure 4 shows a another gap of L^{2nd} with EPHN and EPSHN on CUB, CAR and In-shop dataset for most choices of the group size n of elements per class in each batch. This result indicates that both Problem 1 and 2 are important for metric learning.

6.4 Comparing to the state of art

Finally we compare our results with current state-of-the-art embedding approaches, including more complex triplet loss approaches (Yuan et al., 2017; Ge, 2018; Cakir et al., 2019) and ensemble based approaches (Opitz et al., 2017; Kim et al., 2018; Xuan et al., 2018). Our embeddings are trained with ResNet50 and an output embedding size of 512. For CUB, CAR, SOP and In-shop, the optimal group size is 8,16,2 and 2. In Table 1, the 2nd order Easy Positive Hard Negative approach achieves a new record on the CUB and SOP dataset. On the CAR and In-shop dataset, our result is comparable to the ensemble methods.

7 Conclusion

This paper uses the triplet scatter diagram as a way to describe triplet selection strategies. The diagram offers the ability to characterize how triplets change as the Deep Metric Learning progresses, and we explore the behavior of the gradient descent optimization for the common case where points are normalized to a hypersphere. We find that important classes of triplets have an effective gradient forces negative examples closer to the anchor, or positive examples farther from the anchor, and situations that encourage triplets of images that are all similar to become even more similar. This explain previously observed bad behavior for hard-negative triplet mining. We suggest a simple modification to the desired gradients, and derive a loss function that gives those gradients. Experimentally we show that this improves the convergence for hard negative triplet selection strategies.

With this modification, we no longer observe challenges in optimization with the Easy-Positive Hard-Negative triplet mining strategies and show that easy-positive hard negative mining gives results that exceed or are competitive with state of the art approaches that including complicated network architectures and ensembles.

REFERENCES

Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *AISTATS*, volume 10, pp. 33–40, 2005.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 539–546, 2005.

Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proc. European Conference on Computer Vision (ECCV)*, September 2018.

Jacob Goldberger, Geoffrey E Hinton, Sam T. Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 17*, pp. 513–520. MIT Press, 2005. URL `http://papers.nips.cc/paper/2566-neighbourhood-components-analysis.pdf`.

Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2821–2829, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proc. European Conference on Computer Vision (ECCV)*, September 2018.

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.

Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proc. International Conference on Computer Vision (ICCV)*, Oct 2017.

Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier - boosting independent embeddings robustly. In *Proc. International Conference on Computer Vision (ICCV)*, Oct 2017.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision*, pp. 3–20. Springer, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

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

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015b.

Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 118–126, 2015.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 1857–1865. 2016. URL http://papers.nips.cc/paper/6200-improved-deep-metric-learning-with-multi-class-n-pair-loss-objective.pdf.

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

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. *CoRR*, abs/1801.09414, 2018. URL http://arxiv.org/abs/1801.09414.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1386–1393, 2014.

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.

Hong Xuan, Richard Souvenir, and Robert Pless. Deep randomized ensembles for metric learning. In *Proc. European Conference on Computer Vision (ECCV)*, September 2018.

Hong Xuan, Abby Stylianou, and Robert Pless. Improved embeddings with easy positive triplet mining. *arXiv preprint arXiv:1904.04370*, 2019.

Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proc. International Conference on Computer Vision (ICCV)*, Oct 2017.

Shi Qiu Xiaogang Wang Ziwei Liu, Ping Luo and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

A Appendix: Similarity after gradient updating for 1st order loss

The following derivation shows how to get S_{ap}^{new} and S_{an}^{new} in equation 7 and 8.

$$\begin{aligned} S_{ap}^{new} =& {\mathbf{f_a}^{new}}^\intercal {\mathbf{f_p}}^{new} \\ =&(1+\beta^2)\mathbf{f_a}^\intercal\mathbf{f_p} + \beta\mathbf{f_a}^\intercal\mathbf{f_a} + \beta\mathbf{f_p}^\intercal\mathbf{f_p} - \beta\mathbf{f_n}^\intercal\mathbf{f_p} - \beta^2\mathbf{f_n}^\intercal\mathbf{f_a} \\ =&(1+\beta^2)S_{ap} + 2\beta - \beta S_{pn} - \beta^2 S_{an} \end{aligned} \tag{17}$$

$$\begin{aligned} S_{an}^{new} =& {\mathbf{f_a}^{new}}^\intercal {\mathbf{f_n}}^{new} \\ =&(1+\beta^2)\mathbf{f_a}^\intercal\mathbf{f_n} - \beta\mathbf{f_a}^\intercal\mathbf{f_a} - \beta\mathbf{f_n}^\intercal\mathbf{f_n} + \beta\mathbf{f_p}^\intercal\mathbf{f_n} - \beta^2\mathbf{f_p}^\intercal\mathbf{f_a} \\ =&(1+\beta^2)S_{ap} - 2\beta + \beta S_{pn} - \beta^2 S_{ap} \end{aligned} \tag{18}$$

We construct two planes: P_{ap} spanned by $\mathbf{f_a}$ and $\mathbf{f_p}$, and P_{an} spanned by $\mathbf{f_a}$ and $\mathbf{f_n}$. On P_{ap}, $\mathbf{f_p}$ can be decomposed into two components: $\mathbf{f_p^{a\parallel}}$(the direction along $\mathbf{f_a}$) and $\mathbf{f_p^{a\perp}}$(the direction vertical to $\mathbf{f_a}$). On P_{an}, $\mathbf{f_n}$ can be decomposed into two components: $\mathbf{f_n^{a\parallel}}$(the direction along $\mathbf{f_a}$) and $\mathbf{f_n^{a\perp}}$(the direction vertical to $\mathbf{f_a}$). Then the S_{pn} should be:

$$S_{pn} = \mathbf{f_p}^\intercal\mathbf{f_n} = (\mathbf{f_p^{a\parallel}} + \mathbf{f_p^{a\perp}})(\mathbf{f_n^{a\parallel}} + \mathbf{f_n^{a\perp}}) = S_{ap}S_{an} + \gamma\sqrt{1-S_{ap}^2}\sqrt{1-S_{an}^2} \tag{19}$$

where $\gamma = \frac{\mathbf{f_p^{a\perp}}\mathbf{f_n^{a\perp}}}{\|\mathbf{f_p^{a\perp}}\|\|\mathbf{f_n^{a\perp}}\|}$ which represents the projection factor between P_{ap} and P_{an}

B APPENDIX: NORM OF UPDATED FEATURES FOR 1ST ORDER LOSS

The following derivation shows how to derive $\|\mathbf{f_a}^{new}\|$, $\|\mathbf{f_p}^{new}\|$ and $\|\mathbf{f_n}^{new}\|$ in equation 9 and 10. On P_{ap}, $\mathbf{g_p}$ can be decomposed into the direction along $\mathbf{f_p}$ and the direction vertical to $\mathbf{f_p}$. On P_{an}, $\mathbf{g_n}$ can be decomposed into the direction along $\mathbf{f_n}$ and the direction vertical to $\mathbf{f_n}$. Then,

$$\|\mathbf{f_p}^{new}\|^2 = (1+\beta S_{ap})^2 + \beta^2(1-S_{ap}^2) \tag{20}$$

$$\|\mathbf{f_n}^{new}\|^2 = (1-\beta S_{an})^2 + \beta^2(1-S_{an}^2) \tag{21}$$

On P_{ap}, $\mathbf{g_a}$ can be decomposed into 3 components: component in the plane and along $\mathbf{f_a}$, component in the plane and vertical $\mathbf{f_a}$, and component vertical to P_{ap}. Then,

$$\|\mathbf{f_a}^{new}\|^2 = (1+\beta S_{ap}-\beta S_{an})^2+(\beta\sqrt{1-S_{ap}^2}-\gamma\beta\sqrt{1-S_{an}^2})^2+(\beta\sqrt{1-\gamma^2}\sqrt{1-S_{an}^2})^2 \tag{22}$$

C APPENDIX: SIMILARITY AFTER GRADIENT UPDATING FOR 2ND ORDER LOSS

The following derivation shows how to get S_{ap}^{new} and S_{an}^{new} after gradient update with L^{2nd}

$$S_{ap}^{new} = (1+w_{ap}^2\beta_w^2)S_{ap} + 2w_{ap}\beta_w - w_{an}\beta_w S_{pn} - w_{ap}w_{an}\beta_w^2 S_{an} \tag{23}$$

$$S_{an}^{new} = (1+w_{an}^2\beta_w^2)S_{ap} - 2w_{an}\beta_w + w_{ap}\beta_w S_{pn} - w_{ap}w_{an}\beta_w^2 S_{ap} \tag{24}$$

D APPENDIX: NORM OF UPDATED FEATURES FOR 2ND ORDER LOSS

The following derivation shows how to derive $\|\mathbf{f_a}^{new}\|$, $\|\mathbf{f_p}^{new}\|$ and $\|\mathbf{f_n}^{new}\|$ after gradient update with L^{2nd}.

$$\|\mathbf{f_p}^{new}\|^2 = (1+w_{ap}\beta_w S_{ap})^2 + w_{ap}^2\beta_w^2(1-S_{ap}^2) \tag{25}$$

$$\|\mathbf{f_n}^{new}\|^2 = (1-w_{an}\beta_w S_{an})^2 + w_{an}^2\beta_w^2(1-S_{an}^2) \tag{26}$$

$$\begin{aligned}\|\mathbf{f_a}^{new}\|^2 &= (1+w_{ap}\beta_w S_{ap} - w_{an}\beta_w S_{an})^2 \\ &+(w_{ap}\beta_w\sqrt{1-S_{ap}^2} - \gamma w_{an}\beta_w\sqrt{1-S_{an}^2})^2 \\ &+(w_{an}\beta_w\sqrt{1-\gamma^2}\sqrt{1-S_{an}^2})^2\end{aligned} \tag{27}$$

E APPENDIX: GRADIENT UPDATES FOR VANILLA TRIPLET LOSS

The main paper uses NCA to define the loss. Another common vanilla triplet-loss function is derived to guarantee a specific margin. This can be expressed in terms of $\mathbf{f_a}$,$\mathbf{f_p}$,$\mathbf{f_n}$ as:

$$L = \max(\|\mathbf{f_a}-\mathbf{f_p}\|^2 - \|\mathbf{f_a}-\mathbf{f_n}\|^2 + \alpha, 0) = \max(D, 0) \tag{28}$$

$$\mathbf{g_p} = \frac{\partial L}{\partial \mathbf{f_p}} = \begin{cases} -\beta(\mathbf{f_a}-\mathbf{f_p}) & \text{if } D>0 \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

$$\mathbf{g_n} = \frac{\partial L}{\partial \mathbf{f_n}} = \begin{cases} \beta(\mathbf{f_a}-\mathbf{f_n}) & \text{if } D>0 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

$$\mathbf{g_a} = \frac{\partial L}{\partial \mathbf{f_a}} = \begin{cases} \beta(\mathbf{f_n}-\mathbf{f_p}) & \text{if } D>0 \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

where $D = (\|\mathbf{f_a}-\mathbf{f_p}\|^2 - \|\mathbf{f_a}-\mathbf{f_n}\|^2 + \alpha)$ and $\beta = 2$. For simplicity, in the following discussion, we set $D > 0$ for vanilla triplet loss. Then we can get the $\mathbf{f_a}^{new}$, $\mathbf{f_p}^{new}$ and $\mathbf{f_n}^{new}$ and their norm:

$$\mathbf{f_p}^{new} = \mathbf{f_p} + \beta(\mathbf{f_a}-\mathbf{f_p}) = (1-\beta)\mathbf{f_p} + \beta\mathbf{f_a} \tag{32}$$

$$\mathbf{f_n}^{new} = \mathbf{f_n} - \beta(\mathbf{f_a}-\mathbf{f_n}) = (1+\beta)\mathbf{f_n} - \beta\mathbf{f_a} \tag{33}$$

$$\mathbf{f_a}^{new} = \mathbf{f_a} - \beta\mathbf{f_n} + \beta\mathbf{f_p} \tag{34}$$

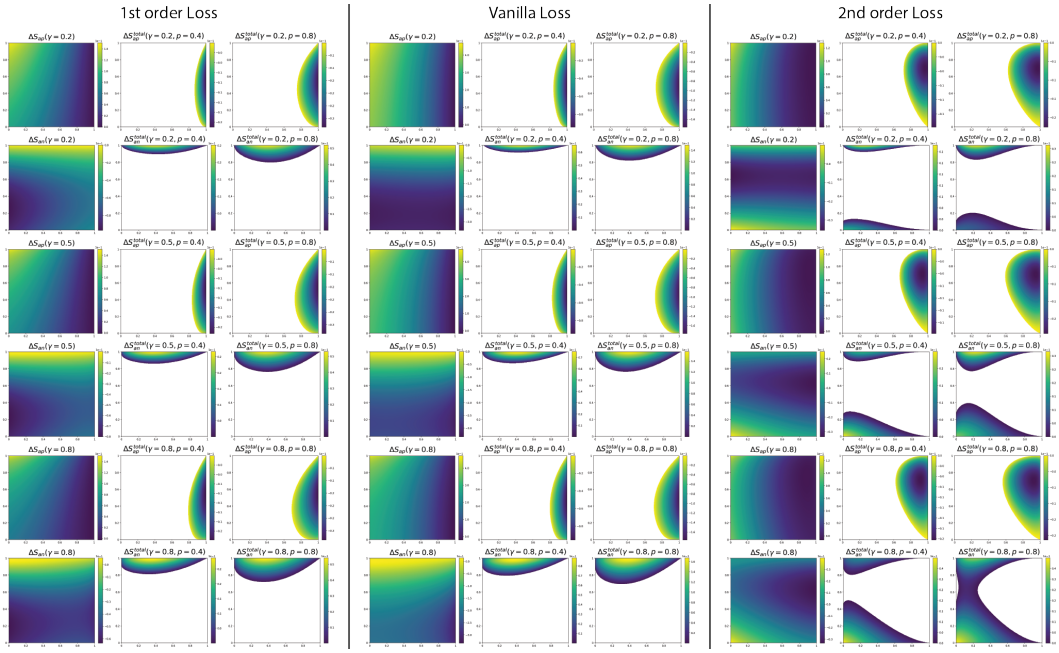

Figure 5: Numerical simulation for ΔS_{ap}, ΔS_{an}, ΔS_{ap}^{total} and ΔS_{an}^{total} change of L^{1st}, L^{v} and L^{2nd} with $\gamma = 0.2, 0.5, 0.8$ and $p = 0.4, 0.8$.

$$\|\mathbf{f_p}^{new}\|^2 = (1 - \beta + \beta S_{ap})^2 + \beta^2(1 - S_{ap}^2) \tag{35}$$

$$\|\mathbf{f_n}^{new}\|^2 = (1 + \beta - \beta S_{an})^2 + \beta^2(1 - S_{an}^2) \tag{36}$$

$$\|\mathbf{f_a}^{new}\|^2 = (1+\beta S_{ap}-\beta S_{an})^2+(\beta\sqrt{1 - S_{ap}^2}-\gamma\beta\sqrt{1 - S_{an}^2})^2+(\beta\sqrt{1-\gamma^2}\sqrt{1 - S_{an}^2})^2 \tag{37}$$

The updated similarity S_{ap}^{new} and S_{an}^{new} will be:

$$S_{ap}^{new} = (1 - \beta + \beta^2)S_{ap} + 2\beta - \beta^2 - \beta(1 - \beta)S_{pn} - \beta^2 S_{an} \tag{38}$$

$$S_{an}^{new} = (1 + \beta + \beta^2)S_{an} - 2\beta - \beta^2 + \beta(1 + \beta)S_{pn} - \beta^2 S_{ap} \tag{39}$$

Comparing to the equation 7 and 8, vanilla triplet behavior is similar to the triplet-NCA. And we simulate the ΔS_{ap} and ΔS_{an} with the vanilla triplet loss in figure 5.

We show results for EPHN for $n = 2$ and $n = 8$ on CAR dataset when the hard negative mining problem happens during the optimization. In both cases, all points are initially pushed towards the same location (1,1), which matches our predicted movement on the vector field of L^{1st} with entanglement in figure 2.

F Appendix: Numerical simulation for similarity change and their entanglement result for 1st order, vanilla and 2nd order loss

Figure 5 shows the numerical simulation ΔS_{ap} and ΔS_{an} with L^{1st}, L^{v} and L^{2nd}. These show that the problematic regions in terms are qualitatively similar for different values of p, the parameter in our model of entanglement.

G Appendix: EPHN for 1st and 2nd order loss

Figure 6 shows the training of the embedding function observed through the triplet scatter diagram, showing for each point the triplet containing the closest positive and the closest negative from whole

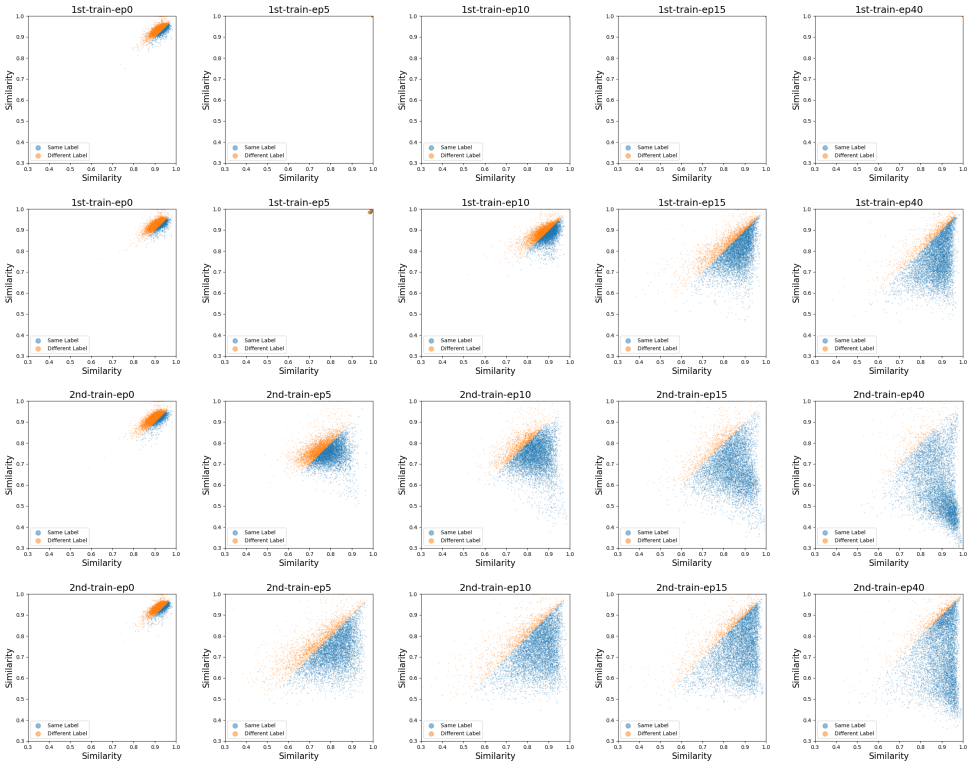

Figure 6: Triplet scatter plots after training epoch 0,5,10,15,40(epoch 0 represents the output of imagenet initialization). 1st row: 1st order EPHN with $n = 2$;2nd row: 1st order EPHN with $n = 8$;3rd row: 2nd order EPHN with $n = 2$;4th row: 2nd order EPHN with $n = 8$

dataset after each epoch training. We observe the dynamic movement to the singularity at the location (1,1) results for EPHN for $n = 2$. The L^{2nd} loss for EPHN never has this problem because the weighted gradient approach more effectively separates hard negative pairs early on in the optimization.

When $n = 8$, the easy-positive part of the triplet is easier because each can pick the closest of 7 neighbors from the same class instead of the randomly assigned same class element and S_ap will be closer to 1. Meanwhile, the negative is likely to be less similar because there are fewer other examples to choose from. Therefore optimization recovers because triplets are less likely to have large S_{an} and get trapped in the part of the vector field that lead to (1,1).