Deep Metric Learning Using Negative Sampling Probability Annealing

Gábor Kertész

John von Neumann Faculty of Informatics, Obuda University, 1034 Budapest, Hungary; kertesz.gabor@nik.uni-obuda.hu

Abstract: Multiple studies have concluded that the selection of input samples is key for deep metric learning. For triplet networks, the selection of the anchor, positive, and negative pairs is referred to as triplet mining. The selection of the negatives is considered the be the most complicated task, due to a large number of possibilities. The goal is to select a negative that results in a positive triplet loss; however, there are multiple approaches for this—semi-hard negative mining or hardest mining are well-known in addition to random selection. Since its introduction, semi-hard mining was proven to outperform other negative mining techniques; however, in recent years, the selection of the so-called hardest negative has shown promising results in different experiments. This paper introduces a novel negative sampling solution based on dynamic policy switching, referred to as negative sampling probability annealing, which aims to exploit the positives of all approaches. Results are validated on an experimental synthetic dataset using cluster-analysis methods; finally, the discriminative abilities of trained models are measured on real-life data.

Keywords: negative sampling probability annealing; triplet mining; deep metric learning; triplet network

1. Introduction

Image-based instance re-identification, one- and few-shot learning, and image similarity analysis are popular fields of computer vision research, with applications in vehicle recognition [1,2] and tracking [3], and facial recognition and identification [4]. Solutions based on deep learning [5] use a dimensionality-reduction technique to transform observations into an embedded space where distance represents similarity—this is referred to as metric learning.

Over recent decades, multiple machine learning approaches have been studied, starting with the work of Bromley et al. who introduced the Siamese architecture [6] for signature verification. Recently, in the era of deep learning, applications in facial recognition drove researchers to reach peak performance; DeepFace [7], FaceNet [8], and OpenFace [9] represent deep-metric-learning-based solutions for facial identification, applying novel solutions, such as triplet loss and triplet mining.

The application of triplet loss—as an error function—as a replacement for the contrastive loss of the Siamese net redefines the structure of the problem as well. The Siamese net consists of two inputs, and loss is calculated based on the distance of the embedded vectors; for same-class elements, loss is proportional to distance, and the opposite is true for elements of different classes.

The training process differs from supervised learning where, in an epoch, every sample is used for training. When training a Siamese-architected network, applying every single sample would produce a total of $\frac{n(n-1)}{2}$ for $n$ (number) total training samples. In the case of deep learning, where a large number of training samples are used, the exponential growth of training steps is difficult to handle. Another problem in this scenario is the imbalance of positives and negatives: by increasing the number of classes, the number of pairs from different classes increases when compared with the number of same-class pairs.
The solution—of course—is to select both positive and negative pairs in the same number during training.

For triplet nets, three samples are used during training: a so-called anchor element used as the basis, along with a positive and a negative pair, formally given as \( T = \{a, p, n\} \). These elements—the triplet—are then used to calculate the loss using the triplet loss \[8\]:

\[
\mathcal{L} = \max(0, m + d^+ - d^-),
\]

where \( d^+ \) and \( d^- \) refer to the positive and negative distance, respectively, defined as

\[
d^+ = d(a, p), \quad d^- = d(a, n),
\]

\( d(\cdot, \cdot) \) representing the distance function, and \( m \) is the margin.

Similarly as above, a training that includes all possible triplets would be inefficient and computationally heavy, so triplets are selected such that the training is effective. Triplet mining is the method of selecting a batch of triplets where the training would result in a non-zero loss; of course, other policies are often taken into account. In theory, a high loss could be measured for a given \( a \) having the furthest \( p \) and the closest \( n \) to form as a triplet; however, experiments on real-life data show that the result is inflexible clusters and the classification performance of the models on unseen samples is low \[10,11\].

Therefore, other strategies for positive and negative sampling are often used. For selecting \( p \), so-called easy sampling is popular, where easy refers to the low computational cost of random selection. On the other hand, hard positive sampling refers to filtering and selecting a positive which is further than the other candidates.

For negative sampling, after \( a \) and \( p \) are known, different policies are available. The previously mentioned easy mining is a random selection; however, it is not a good choice: because of (1), if the selected \( n \) results in a \( d^- \) where \( d^- > d^+ + m \), then the resulting \( \mathcal{L} \) is zero. Therefore, negative sampling should always consider those \( n \) candidates, where \( \mathcal{L} \) is non-zero \[12\]. There are also choices on how to select negative pairs for the triplet. Random hard mining refers to randomly selecting \( n \) from candidates where the resulting loss is non-zero. Hardest mining is selecting \( n \) with the minimal \( d^- \), i.e., the closest one. Another interesting sampling technique is semi-hard mining, where \( n \) is randomly selected from those elements that result in non-zero loss, but are in the marginal distance.

Unfortunately, none of the above sampling techniques provide general, solid methods for deep metric learning; multiple approaches were investigated to improve performance.

**Related Work**

The paper on FaceNet introduced triplet loss \[8\], a similar solution presented in a paper by Hoffer et al. \[13\]; since these, multiple extensions, alterations, and optimization techniques, along with some interesting use cases were revealed.

Amos et al. showed \[9\] that triplet mining is useful, and can be performed efficiently. Online triplet mining—the method where triplets are selected right before the training step—is performing remarkably better than its counterpart (offline mining), where embeddings are unaffected by the last steps \[14\].

Wu et al. showed \[12\] that sampling matters, and it is even more significant than the selected loss function—or the hyperparameters of the selected function—in achieving high performance. The first sampling method was proposed in the original paper by Schroff et al. \[8\]: semi-hard negative sampling showed great potential in converging to optimal clustering. A similar approach (although for Siamese nets) was proposed by Simo-Serra et al. \[15\], where positive pairs and negative pairs were sorted by loss in descending order, and used for training accordingly. Harwood et al. \[16\] proposed an online mining method based on semi-hard sampling of negatives. Hermans et al. \[17\] showed that the use of hard positive sampling is effective in clustering same-class samples, although it is also demonstrated that the method is unstable and not necessarily applicable to all kinds of data. Recently, Xuan et al. \[18\] showed that hard negative sampling at early stages leads to
sticking in local minima, resulting in a suboptimal model. Kalantidis et al. [19] proposed a method for synthetic hard negative mixing to gain advantages of hard negative mining, without the drawbacks.

Different loss function alternatives were also proposed and analyzed: first and foremost, magnet loss [20] is a computationally hard approach with high memory costs continuously analyzing the clusters. Due to the frequent stops during the training process, the magnet loss method might show high performance, but it is quite inefficient. Wang et al. [21] introduced the ranked list loss to deal with sampling issues, and used data from non-trivial data points as well. Alternatives of the triplet loss, such as exponential triplet loss [22] and lossless triplet loss [23], have also been shown to be promising alternatives. It is also worth mentioning that a recent study showed that classical methods show similar performances to state-of-the-art loss functions [24].

A different viewpoint of the problem concentrates on the fact that metric learning is performed on Euclidean space; however, loss is based on non-Euclidean distances [25]. Novel solutions are based on a special type of Riemannian manifold, e.g., the Grassmann manifold or the Stiefel manifold [26]. Multiple studies have concluded that the application of metric learning in nonlinear structures shows promising results in face recognition [27–29].

In some cases—when applicable—pre-training or transfer learning applied to the backbone model is proven to be useful for deep metric learning applications [30–32]. Modern facial recognition solutions—e.g., SphereFace [33,34]—propose solutions based on angular marginal loss, which inspired multiple other works on the same topic, e.g., CosFace [35]. Another interesting application in visual sensors is for object tracking: recently, metric-learning-based trackers showed great performance [36,37]. The state-of-the-art results on applying the SiamRPN++ [38] tracker further fuel the research in deep metric learning.

In this paper, a method of cluster analysis is introduced; based on the observations, a novel negative sampling method is proposed and evaluated. The main contributions of the current study are:

1. Formal description and summary of negative sampling methods;
2. Cluster-analysis methods formally introduced and applied during different experiments on a synthetic dataset;
3. Introduction of the negative sampling probability annealing algorithm for negative selection, with evaluation of results on synthetic dataset as well as actual data; it can be concluded that—while the best-performing models behave similarly—in general, NSPA and random hard sampling seem to outperform other approaches.

The structure of the paper is as follows: in Section 2.1, the triplet network is introduced in formal terminology, including some of the sampling methods. In Section 2.2, the cluster analysis and the properties are presented. Section 2.3 describes the novel method on negative sampling. Section 3 describes the implementation and presents the results; finally, in Section 4, a discussion of the results is presented.

2. Methodology

In this section, inspiration and methods are presented in detail.

2.1. Triplet Network

As previously defined, a training triplet can be formalized as

\[ T = \{a, p, n\}, \]

\[ a, p, n \in X, \]

where \(a\) represents the anchor, \(p\) is the positive pair, and \(n\) is the negative pair, all coming from the dataset, \(X\). Function \(f(\cdot)\) represents the class of each sample as

\[ f : X \rightarrow Y, \]

\[ \forall x \in X, \exists y \in Y : f(x) = y, \]
where \( Y \) represents the set of classes. Based on this, the positive pair can be defined as

\[
\forall a \exists p : f(a) = f(p), \quad a \neq p.
\] (5)

Similarly, \( n \) can be given as

\[
\forall a \exists n : f(a) \neq f(n).
\] (6)

This case is—of course—an example of easy negative sampling: any randomly selected negative sample fits the criteria. Given \( n_H \) as a random hard negative, where the loss is non-zero:

\[
\forall a, p \exists n_H : f(a) \neq f(n), \quad d(a, n_H) < d(a, p) + m
\]
\[
\forall a, p, n_H \in X, \quad m > 0.
\] (7)

Semi-hard sampling can be similarly defined as

\[
\forall a, p \exists n_{SH} : f(a) \neq f(n), \quad d(a, p) < d(a, n_{SH}) < d(a, p) + m
\]
\[
\forall a, p, n_{SH} \in X, \quad m > 0.
\] (8)

Finally, the hardest negative is the one which maximizes the loss for a given \( a \) and \( p \):

\[
n_{H^*} = \{ \arg \min_{x \in X} \{ d(a, x) \} \}.
\] (9)

In the following, training is performed by selecting random anchors and applying easy positive mining—i.e., randomly selecting a positive pair, given as

\[
\forall a : a \in \mathbb{R}^X;
\]
\[
\forall p : p \in \mathbb{R}^X, f(a) = f(p).
\] (10)

On the negative sampling method, if random hard, semi-hard, or the hardest mining is applied, \( n \) is one of the following:

\[
n \in \{ n_H, n_{SH}, n_{H^*} \}.
\] (11)

Figure 1 presents triplets and negative sampling possibilities.

---

**Figure 1.** Two-dimensional representation of the embedded space. Gray cross in the middle marks the anchor element, \( a \), and a green point shows a positive pair, \( p \). \( d^+ \) positive distance is therefore defined and represented as a circle around \( a \). Marginal distance \((d^+ + m)\) is similarly shown as a dashed circle. The closest negative sample, which provides the highest loss, is represented with a star; other elements where \( d^- < d^+ \) are shown with squares, and triangles represent negatives where \( d^+ < d^- < d^+ + m \). Other positive and negative data points are represented with plus and minus signs, respectively.
2.2. Cluster Analysis

To observe the effects of different sampling methods, cluster properties are observed on a synthetic dataset during triplet-loss-based metric learning. The following properties were defined and implemented and measured during training, Section 3 presents the findings.

For cluster analysis [39,40], we define the centroid, $c_y$, of class, $y$, as

$$c_y = [c_y(1), c_y(2), \ldots, c_y(n)],$$

$$c_y(i) = \sum_{x_y \in X_y} x_y(i) / |X_y|,$$

(12)

where $n$ represents the dimensionality of the data points, $y$ stands for the selected class, and $X_y$ represents the set of elements of class $y$. Centroid, $c_y$, is given by selecting the averages of the values in each dimension.

One of the measured properties is the average distance of centroids, which simply represents

$$\forall y_1, y_2 \in [Y]^2,$$

$$\text{avgCentroidDistance} = \sum_{y_1, y_2 \in [Y]^2} d(c_{y_1}, c_{y_2}) / |[Y]^2|,$$

(13)

where $[Y]^2$ represents the set of all combinations with two elements of $y$ classes without repetition.

For each class, the radius of the cluster is given as the distance of the centroid and the furthest class member, formally described as

$$r_y = \max_{x_y \in X_y} \{d(x_y, c_y)\}.$$

(14)

Another measured property is the average cluster radius, which is given as

$$\forall y \in Y,$$

$$\text{avgClusterRadius} = \sum_{y \in Y} r_y / |Y|.$$

(15)

Figure 2 shows a sample for cluster centroid and the defined radius.

![Figure 2](image-url)

Figure 2. A visual representation of the centroid and the radius in two-dimensions. The centroid is the gravitational center of all data points of the same class, marked by a cross. The radius is given by selecting the maximal distance from the centroid to the data points, represented by a circle.

Based on this definition of a cluster, we can conclude that, in this case, all elements of the cluster will be inside the cluster, as $\forall x_y \in X_y : d(x_y, c_y) \leq r_y$. An interesting property is the number of other elements, which is given as
where \( m_y \) gives the number of negatives in cluster \( y \), and \( q_y \) gives the ratio of negatives in comparison to all elements in the cluster. Similarly, the proportion of negatives in marginal distance can be given based on margin \( m \) by having \( d(x, c_y) \leq r_y + m \) as the filter.

Further properties of the clusters can be given solely on the distances of data points, e.g.,

\[
\forall x_1, x_2 \in [X]^2, \\
\text{avgDistanceBetweenElements} = \sum_{x_1, x_2 \in [X]^2} d(x_1, x_2)/|[X]^2|,
\]

(17)
gives the average distance between any two elements. Similarly, the mean distance between positive elements can be given by measuring the distance between all same-class elements in all classes, \( y \), and returning with the average.

The maximal distance between any pair of elements of a given class can be formalized as

\[
\forall x_{y_1}, x_{y_2} \in [X_y]^2, \\
g(y) = \max\{d(x_{y_1}, x_{y_2})\}_{x_{y_1}, x_{y_2}}^{[X_y]^2}, \\
\text{avgDistanceOfFurthestPositives} = \sum_{y \in Y} g(y)/|Y|,
\]

(18)
where function \( g(\cdot) \) gives the distance of the furthest pair for a given class, \( y \).

An other interesting measurable property is the average distance of the closest negative for each data point. Formally given,

\[
h(x) = \min\{d(x, x^-)\}_{x \in [X]}^{X} |f(x) \neq f(x^-)|, \\
\text{avgClosestNegatives} = \sum_{x \in [X]} h(x)/|X|,
\]

(19)
where function \( h(\cdot) \) represents the distance to the closest different classed element for a given input \( x \in X \).

At the beginning of the training process, the clusters are not separable, and the elements are scattered in the embedded space. Therefore, the average radius, the average positive distance, the average distance of closest negatives, and the average distance of the furthest positives are not informative by themselves. These distances, in contrast to the average distance between all elements, are informative—i.e., we expect the positive distances to decrease compared with the dynamic of average distance. A normalized value of these properties are hereby defined, e.g.,

\[
\text{normDistanceOfFurthestPositives} = \frac{\text{avgDistanceOfFurthestPositives}}{\max\{\text{avgDistanceBetweenElements}, \epsilon\}},
\]

(20)
where the denominator is non-zero, thus the threshold \( \epsilon \), which is selected as \( 10^{-5} \).

Further details about implementation can be found in Section 3.

2.3. Negative Sampling Probability Annealing

By reviewing the relevant literature and analyzing the results of the cluster analysis on synthetic data (presented in detail in Section 4), a method of negative sampling policy switching is proposed.

The main ideas are as follows:

- Random hard sampling is efficient, and converges at the start;
- Semi-hard sampling performs better than the other methods;
Sampling the hardest negative has potential, but only in later phases. Therefore, as a naïve approach on sampling switching, random sampling should be used at the beginning, which should be followed by semi-hard mining, and finally hardest mining should be used. This approach would be unnecessarily rigid however, and the optimization of hyperparameters of phase boundaries seems to be key.

The evaluation of a random sampling policy-switching method showed [41] that both the drawbacks and benefits of each technique are combined in the results, while the method itself looks promising.

As a middle-ground between the abovementioned approaches, a probability-based method is proposed where probabilities of different sampling policies change during the training process.

Formally, let $H$ represent random hard negative sampling, $SH$ stand for semi-hard negative sampling, and $H^*$ stand for hardest negative selection. For a uniform random policy selection, the probabilities of using each method are given as

$$P(H) = \frac{1}{3}, \quad P(SH) = \frac{1}{3}, \quad P(H^*) = \frac{1}{3},$$

or by defining $P = [P(H), P(SH), P(H^*)]$

$$P = \begin{bmatrix} 1 & 1 & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}.$$  \hspace{1cm} (22)

In the proposed annealing method, $P = [1, 0, 0]$ initially, and occasionally the values are updated by increasing both $P(SH)$ and $P(H^*)$ with $step_{SH}$ and $step_{H^*}$, while having

$$\sum P(i) = 1,$$  \hspace{1cm} (23)

therefore continuously decreases the value of $P(H)$. Detailed behavior is explained in Algorithm 1.

The fundamental behavior is that the probability update method defined in Algorithm 1 is called during training repeatedly, with the actual value of probabilities referred to as $P$, predefined and fixed step sizes in the range of $[0; 1)$, expecting $step_{SH} < step_{H^*}$; and the maximal value of hardest sampling probability $H^*_{max} \in [0; 1)$.

---

**Algorithm 1** Negative Sampling Probability Annealing

```python
function NSPA(P, step_{SH}, step_{H^*}, H_{max})
  A ← [0, P[SH] + step_{SH}, P[H^*] + step_{H^*}]
  A[H^*] ← min A[H^*], H_{max}
  A[H] ← 1 - A[SH] + A[H^*]
  for all a ∈ A do
    a ← min{a, 1}
    a ← max{a, 0}
  end for
  if SUM(A) > 1 then
    A[ARGMAX(A)] ← A[ARGMAX(A)] - (SUM(A) - 1)
  end if
  return A
end function
```
The algorithm increases the probability of semi-hard and hardest mining until $P(H) = 0$, afterwards $P(H^*)$ monotonically increases while $P(SH)$ starts to decrease. A sample is shown on Figure 3.

Implementation and on-training use is detailed in Section 3.

![Figure 3](image_url)

**Figure 3.** Sample behavior of the proposed probability annealing algorithm for $step_{SH} = 0.1$, $step_{H*} = 0.01$ and $H_{max}^* = 0.5$. Initial $P$ is $[1, 0, 0]$ for random hard, semi-hard, and hardest negative sampling, respectively.

### 3. Results

This section presents the results of the synthetic cluster analysis and the results of applying the proposed NSPA method. In the second part of the section, performance results of NSPA applied on a real-life dataset are given. The source codes and materials are publicly available; please refer to Supplementary Materials.

#### 3.1. Synthetic Cluster Analysis

To understand the effect of different negative sampling methods on cluster properties, a synthetic set of data points was generated. Data points were generated from 7 classes, each sample represented in a 32-dimensional space. During generation, for each class, a random point was selected in the space, and points near that center were generated based on Gaussian normal distribution. For training, 5 selected classes and a total of 1000 samples were selected, and 100 samples of each class were moved to a validation subset, and were not used directly for training, only to evaluate model performance.

Training was implemented in Keras [42] and Tensorflow [43] using a slightly modified version of the EmbeddingNet framework [44,45]. The embedding network used in the triplet architecture is a simple 32–16–2 fully connected network with ReLU activations. Adam optimizer [46] was used with a learning rate set to $5 \cdot 10^{-4}$. No pretraining was applied. Mining was performed by selecting 20–20 samples from every 5 classes in each training step. A training iteration (or epoch) consisted of 50 training steps. Training concluded when loss measured on the validation set stopped improving for at least 25 epochs.

At every 10 epochs, embedded vectors of the validation dataset were collected and stored until training stopped.

A total of 20 training rounds were performed for each negative sampling method, resulting in a total of 60 logs of cluster information at different points of training. Figure A1 shows the results of the measurements, which were as expected: multiple studies reported that semi-hard negative sampling outperforms random hard sampling, and hardest sampling is suboptimal in every measured parameter.

Furthermore, the trainings showed that correlation was visible between the average distance between all elements, the average distance between the estimated centroids, and the average distance between the closest negatives. Similar behavior was observable in cases of calculating the average radius of clusters, the average distance of furthest positives, and the average distance between all positive pairs.
When evaluating the parameters normalized by the average element distance based on (20), it is visible (Figure A2) that the random negative sampling performs similarly to the semi-hard sampling, although with larger distances between the clusters.

When measuring performance, the percentage of negatives in a cluster and within marginal distance indicated the discriminative ability of the model. Results are visualized in Figure A3; it is clear that the hardest negative sampling struggled during the initial phases, and failed to improve.

The proposed NSPA method was implemented as a callback in Keras: at each epoch end, the probabilities were updated as defined in Algorithm 1. For the synthetic dataset, the values of \(step_{SH}, step_{H},\) and \(H_{max}\) were 0.1, 0.01, and 0.5, respectively.

Results of cluster analysis compared with the other methods are visualized in Figure 4. Further results are available in Appendix A.

Results on synthetic data are promising; NSPA performed similarly to semi-hard and random hard negative sampling, in some cases results are closer to hardest negative sampling.

To evaluate the performance, the proposed method was also applied to a benchmark dataset—NIST’19 [47].

![Graphs](image)

**Figure 4.** Results from applying the proposed NSPA method to synthetic data. (a) The average distance of cluster centroids; (b) the percent of different-class elements in the cluster; (c,d) display normalized results on the closest negative distance and the distance of the furthest positives, respectively. Dashed lines represent the mean values of different training experiments, while thick lines highlight representative records, which consist of average values including training length. Further results are available in the Appendix A, in Figure A4.

3.2. Discriminative Ability

The NIST dataset is a wider alternative to the well-known MNIST dataset, which is actually a subset of it; the MNIST contains 10 classes of handwritten digits, the NIST also contains samples of handwritten letters.
While the MNIST samples are 28 × 28 images, 60,000 samples in total, NIST are 128 × 128 images in a total of 62 classes, which contains a sum of 814,255 samples. Before applying the images, content highlighting preprocessing step was applied [23], and the resulting images were resized to 96 × 96.

The embedding neural network used was a simple convolutional neural network inspired by the VGG models [48]: after the 96 × 96 input, convolutional and pooling layers follow, and finally fully connected “dense” layers produced the output of 64 dimensions. The marginal distance was set to 0.5, a maximum number of 1500 objects were selected for training per class. Adam optimizer [46] was used with a learning rate set to $10^{-4}$. Early stopping was applied after validation loss failed to improve for a number of 25 epochs. An epoch consisted of 20 steps, where each step 10 classes were selected and for each class 10 samples were used for mining. NSPA was applied with originally proposed values of 0.1, 0.01, and 0.5 for $step_{SH}$, $step_{H^*}$, and $H^{\text{max}}$, respectively. Probability updates happened after every 5 epochs.

The measured losses during training are visualized on Figure 5. This metric shows very similar behavior on all mining methods in this experiment.

![Figure 5](image)

**Figure 5.** The improvement of the measured validation loss during training for different sampling methods. Each were tested in 5–5 training rounds; the best-performing models are selected here.

For evaluation of discriminative ability, the $n$-way one-shot classification accuracy [49,50] was calculated for the proposed method and the other baseline sampling techniques. Measurements were performed on the training and on the validation subset as well. $n$-way one-shot classification accuracy calculation was based on distance comparison of an anchor object and a total of $n$ objects, where $n - 1$ elements were from different classes and exactly one was from the same class. If the distance for the positive pair was minimal, classification was correct; in other cases it was incorrect. The number of correct classifications in measurements gave the one-shot classification accuracy. Whichever method was used, it needed to outperform random classification, which was $\frac{1}{n}$ for $n$ selected objects, i.e., 50% for two selected elements, and 25% for four samples. Table 1 presents the results of one-shot classification accuracy.

Each method was applied in training 5 times, resulting in a total of 20 trained models. Each models were evaluated using the one-shot classification accuracy, and the highest performances were selected in Table 1. Figure 6 visualizes the averages and standard deviations for classification performance of each method applied.
Table 1. One-shot classification accuracy on the NIST dataset when applying different methods, values are given in percentage (%). Classification accuracy was measured for \( n \)-way, referring to the number of selected classes. Values in bold are the best for each column. Random classification is the chance of randomly selecting the correct class; it is only shown for reference.

| Method                    | 1-Way | 2-Way | 3-Way | 4-Way | 5-Way | 6-Way | 7-Way | 8-Way | 9-Way | 10-Way |
|---------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
| Random classification     | 100.0 | 50.0  | 33.3  | 25.0  | 20.0  | 16.7  | 14.3  | 12.5  | 11.1  | 10.0   |
| Random hard negative mining| 100.0 | 96.5  | 93.5  | 92.4  | 88.6  | 85.8  | 84.1  | 83.8  | 81.2  | 80.5   |
| Semi-hard negative mining | 100.0 | 94.9  | 92.2  | 88.6  | 86.4  | 80.1  | 80.1  | 79.0  | 76.1  | 76.4   |
| Hardest negative mining   | 100.0 | 64.3  | 48.8  | 38.3  | 31.7  | 28.5  | 26.5  | 24.1  | 23.3  | 20.0   |
| NSPA                      | 100.0 | 94.8  | 92.9  | 90.0  | 88.1  | 86.6  | 83.2  | 83.0  | 82.5  | 80.7   |

Figure 6. One-shot accuracy for \( n \)-way classification using different negative sampling techniques compared to the proposed NSPA algorithm. Error bars visualize the standard deviation of the performance of different models, while the triangle indicates the top accuracy.

4. Discussion

In this paper, a cluster-analysis method was proposed to understand and measure embedded clustering dynamics. As an initial naïve approach, a synthetic set of multidimensional clusters were generated, and a triplet network with dimensionality-reduction goal was implemented. Different negative sampling methods were applied in multiple tests.

Results are as expected: semi-hard sampling outperforms other methods, with hardest sampling struggling at the initial phase of training. In the defined cluster properties, the average centroid distance and the average closest negative distance show high correlation: in both cases, the larger the distance, the better the performance.

In the case of the average distance between same-class elements—with average maximal positive distances and average cluster radii—a descending behavior was observed during training; however, in case of failed training rounds, the distances were short initially, and did not improve at any state. Thus, the values normalized with the average distance between elements provide further insight. Generally, we can conclude that a significantly reduced cluster radius has a negative effect on performance.

Clustering performance is measured by counting the number of different class elements inside a cluster, including marginal distance as well. Results show that most of the outliers are in case of hardest negative mining, where in some cases model was unable to fit, resulting to all negative embedded points appearing in all classes. It is also worth mentioning that the stability of the methods differs greatly, with the semi-hard sampling showing similar results on most training rounds.

When measuring the results for the proposed NSPA method, it is clear that its behavior is a mixture of all others, including average distances and performance as well.
It is interesting that, in later phases—if reached—behavior is following the pattern of hardest sampling.

After analysis on synthetic data, the NIST dataset was used to benchmark performance. Discriminative ability is defined by one-shot classification accuracy. Results show that the best models perform similarly; however, NSPA and random hard mining seem to outperform the others.

It can be concluded that the proposed NSPA method is promising, although different settings should be evaluated, including lower maximum probability for hardest sampling and different limitations for random and semi-hard probability.

5. Conclusions

In this paper, a novel negative sampling solution is presented. Results indicate that the approach is promising; however, further research on hyperparameter tuning [51] could improve the applicability. It is worth mentioning that some experiments with suboptimal parameter settings showed similar behavior as the worst-performing setups.

Future goals include adaptation to multiple graphical accelerators to improve efficiency [52]; current online mining methods are computationally expensive, parallel, or distributed execution is limited.

While the so-called magnet loss [20] is computationally complicated, its advantages are undeniable. The aim of future research is to develop a method between random-sampling and step-by-step cluster analysis.

It is also worth mentioning that application of transfer learning is showing excellent results in deep metric learning, where applicable. Results reported in this paper should be analyzed in context of pretraining and knowledge transfer as well.

Supplementary Materials: The source codes and materials are made publicly available at https://github.com/kerteszg/nspa.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The research was carried out with the support of the Ministry of Innovation and Technology from the National Research, Development and Innovation Fund, within the framework of the New National Excellence Program “ÚNKP-21-4”. On behalf of Project “Deep machine learning in parallel and distributed environment”, we thank for the usage of ELKH Cloud (https://science-cloud.hu/) that significantly helped us achieving the results published in this paper. The author is grateful to the members of the Applied Machine Learning Research Group of Obuda University John von Neumann Faculty of Informatics for constructive comments and suggestions.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A

In the figures, the transparent lines show the results of each training rounds (20 per sampling method), dashed lines represent the mean of values, and thick lines highlight representative records, which consist of average values including training length.
Figure A1. Cluster analysis results for random hard, semi-hard, and hardest negative mining. Sub-figures show (a) the dynamic of average element distance, (b) average centroid distance, (c) average distance of closest negatives, (d) average length of cluster radius, (e) the average distance between positive pairs, and (f) the average distance of furthest positives, respectively.
Figure A2. Values normalized by the average distance of elements for random hard, semi-hard and hardest negative mining. Subfigures show (a) the normalized dynamic of closest negative distance, (b) cluster radius length, (c) the distance between positive pairs and (d) the distance of furthest positives, respectively.

Figure A3. Clustering performance of random hard, negative mining. (a) The percentage of negatives in cluster compared with all other elements; (b) the same in marginal distance.
Figure A4. Cluster analysis results for the proposed NSPA method. Sub-figures show (a) the dynamic of the normalized closest negative distance, (b) cluster radius, (c) distance of furthest positive, (d) distance of all positives, and (e,f) the average percentage of non-positive pairs in clusters without and including margin, respectively.

References
1. Liu, X.; Liu, W.; Ma, H.; Fu, H. Large-scale vehicle re-identification in urban surveillance videos. In Proceedings of the 2016 IEEE international conference on multimedia and expo (ICME), Seattle, WA, USA, 11–15 July 2016; pp. 1–6.
2. Tadic, V.; Kiraly, Z.; Odry, P.; Trpovski, Z.; Loncar-Turukalo, T. Comparison of Gabor filter bank and fuzzified Gabor filter for license plate detection. Acta Polytech. Hung. 2020, 17, 1–21. [CrossRef]
3. Li, A.; Luo, L.; Tang, S. Real-time tracking of vehicles with siamese network and backward prediction. In Proceedings of the 2020 IEEE International Conference on Multimedia and Expo (ICME), London, UK, 6–10 July 2020; pp. 1–6.
4. Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face recognition systems: A survey. Sensors 2020, 20, 342. [CrossRef]
5. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436–444. [CrossRef] [PubMed]
6. Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. Signature verification using a “siamese” time delay neural network. In Proceedings of the Advances in Neural Information Processing Systems, 28 November–1 December 1994; pp. 737–744.

7. Taigman, Y.; Yang, M.; Ranzato, M.; Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1701–1708.

8. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 815–823.

9. Amos, B.; Ludwiczuk, B.; Satyanarayanan, M. Openface: A general-purpose face recognition library with mobile applications. CMU Sch. Comput. Sci. 2016, 6, 20.

10. Zhai, Y.; Guo, X.; Lu, Y.; Li, H. In defense of the classification loss for person re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–17 June 2019.

11. Yuan, H.; Stylianou, A.; Pless, R. Improved embeddings with easy positive triplet mining. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Village, CO, USA, 1–5 March 2020; pp. 2474–2482.

12. Wu, C.Y.; Mannatha, R.; Smola, A.J.; Krahenbuhl, P. Sampling matters in deep embedding learning. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2840–2848.

13. Hoffer, E.; Ailon, N. Deep metric learning using triplet network. In Proceedings of the International Workshop on Similarity-Based Pattern Recognition, Copenhagen, Denmark, 12–14 October 2015; pp. 84–92.

14. Sikaroudi, M.; Ghojogh, B.; Safarpour, A.; Karray, F.; Crowley, M.; Tizhoosh, H.R. Offline versus online triplet mining based on extreme distances of histopathology patches. In Proceedings of the International Symposium on Visual Computing, San Diego, CA, USA, 5–7 October 2020; pp. 333–345.

15. Simo-Serra, E.; Trulls, E.; Ferraz, L.; Kokkinos, I.; Fua, P.; Moreno-Noguer, F. Discriminative learning of deep convolutional feature point descriptors. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 118–126.

16. Harwood, B.; Kumar BG, V.; Carneiro, G.; Reid, I.; Drummond, T. Smart mining for deep metric learning. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2821–2829.

17. Hermans, A.; Beyer, L.; Leibe, B. In defense of the triplet loss for person re-identification. arXiv 2017, arXiv:1703.07737.

18. Xuan, H.; Stylianou, A.; Liu, X.; Pless, R. Hard negative examples are hard, but useful. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 126–142.

19. Kalantidis, Y.; Sarjiyidiz, M.B.; Pion, N.; Weinzaepfel, P.; Larlus, D. Hard negative mixing for contrastive learning. Adv. Neural Inf. Process. Syst. 2020, 33, 21798–21809.

20. Rippel, O.; Paluri, M.; Dollar, P.; Bourdev, D. Metric learning with adaptive density discrimination. arXiv 2015, arXiv:1511.05939.

21. Wang, X.; Hua, Y.; Kodirov, E.; Hu, G.; Garnier, R.; Robertson, N.M. Ranked list loss for deep metric learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5207–5216.

22. Urtans, E.; Nīkitenko, A.; Vecins, V. Exponential triplet loss. In Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis, 9–12 March 2020; pp. 152–158.

23. Kertész, G. Different triplet sampling techniques for lossless triplet loss on metric similarity learning. In Proceedings of the 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl’any, Slovakia, 21–23 January 2021; pp. 000449–000454.

24. Musgrave, K.; Belongie, S.; Lim, S.N. A metric learning reality check. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 681–699.

25. Hamm, J.; Lee, D.D. Grassmann discriminant analysis: a unifying view on subspace-based learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 376–383.

26. Hua, X.; Ono, Y.; Peng, L.; Xu, Y. Unsupervised Learning Discriminative MIG Detectors in Nonhomogeneous Clutter. IEEE Trans. Comput. 2022, 70, 4107–4120. [CrossRef]

27. Huang, Z.; Wang, R.; Shan, S.; Chen, X. Projection metric learning on Grassmann manifold with application to video based face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 140–149.

28. Wang, R.; Wu, X.J.; Kittler, J. Graph embedding multi-kernel metric learning for image set classification with Grassmannian manifold-valued features. IEEE Trans. Multimed. 2020, 23, 228–242. [CrossRef]

29. Dai, M.; Hang, H. Manifold Matching via Deep Metric Learning for Generative Modeling. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 6587–6597.

30. Wang, C.; Zhang, X.; Lan, X. How to train triplet networks with 100k identities? In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 1907–1915.

31. Li, C.; Ma, X.; Jiang, B.; Li, X.; Zhang, X.; Liu, X.; Cao, Y.; Kannan, A.; Zhu, Z. Deep speaker: An end-to-end neural speaker embedding system. arXiv 2017, arXiv:1705.02304.

32. Wang, M.; Deng, W. Deep face recognition: A survey. Neurocomputing 2021, 429, 215–244. [CrossRef]

33. Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; Song, L. Spherenace: Deep hypersphere embedding for face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 212–220.
34. Liu, W.; Wen, Y.; Raj, B.; Singh, R.; Weller, A. SphereFace Revived: Unifying Hyperspherical Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *22*, 7579. [CrossRef] [PubMed]

35. Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; Liu, W. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5265–5274.

36. Tao, R.; Gavves, E.; Smeulders, A.W. Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27–30 June 2016; pp. 1420–1429.

37. Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H. Fully-convolutional siamese networks for object tracking. In *Proceedings of the European Conference on Computer Vision*, Amsterdam, The Netherlands, 8–16 October 2016; pp. 850–865.

38. Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; Yan, J. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 15–20 June 2019; pp. 4282–4291.

39. Duran, B.S.; Odell, P.L. *Cluster Analysis: A Survey*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013; Volume 100.

40. Suárez, J.L.; García, S.; Herrera, F. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing* **2021**, *425*, 300–322. [CrossRef]

41. Kertész, G. Combining Negative Selection Techniques for Triplet Mining in Deep Metric Learning. In *Proceedings of the 2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*, Reykjavik, Iceland, 6–9 July 2022; pp. 000155–000160.

42. Chollet, F.; et al. *Keras: The Python Deep Learning Library*; Astrophysics Source Code Library: Houghton, MI, USA, 2018.

43. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th [USENIX] Symposium on Operating Systems Design and Implementation ([OSDI] 16)*, Savannah, GA, USA, 2–4 November 2016; pp. 265–283.

44. Yagfarov, R.; Ostankovich, V.; Akhmetzyanov, A. Traffic Sign Classification Using Embedding Learning Approach for Self-driving Cars. In *Human Interaction, Emerging Technologies and Future Applications II*, Proceedings of the 2nd International Conference on Human Interaction and Emerging Technologies: Future Applications (IHIT-El 2020), Lausanne, Switzerland, 23–25 April 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 180–184.

45. Kertész, G. Metric Embedding Learning on Multi-Directional Projections. *Algorithms* **2020**, *13*, 133. [CrossRef]

46. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

47. Grother, P.; Hanaoka, K. *NIST Special Database 19 Handprinted Forms and Characters*, 2nd ed.; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2016.

48. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.

49. Koch, G.; Zemel, R.; Salakhutdinov, R. *Siamese Neural Networks for One-Shot Image Recognition*; ICML Deep Learning Workshop: Lille, France, 2015; Volume 2.

50. Lake, B.M.; Salakhutdinov, R.R.; Tenenbaum, J. One-shot learning by inventing a compositional causal process. In *Advances in Neural Information Processing Systems 26*; Burges, C.J., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2013; pp. 2526–2534.

51. Ngoc, T.T.; Le Van Dai, C.M.T.; Thuyen, C.M. Support vector regression based on grid search method of hyperparameters for load forecasting. *Acta Polytech. Hung.* **2021**, *18*, 143–158. [CrossRef]

52. Szénási, S.; Felde, I. Using multiple graphics accelerators to solve the two-dimensional inverse heat conduction problem. *Comput. Methods Appl. Mech. Eng.* **2018**, *336*, 286–303. [CrossRef]