12-2020

Automatic target recognition with deep metric learning.

Abdelhamid Bouzid
University of Louisville

AUTOMATIC TARGET RECOGNITION WITH DEEP METRIC LEARNING

By

Abdelhamid Bouzid
B.E. Polytechnic Engineer, Tunisia Polytechnic School, 2018

A Thesis
Submitted to the Faculty of the
J.B. Speed School of Engineering
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Computer Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2020

AUTOMATIC TARGET RECOGNITION WITH DEEP METRIC LEARNING

By

Abdelhamid Bouzid
B.E. Polytechnic Engineer, Tunisia Polytechnic School, 2018

A Thesis Approved On

8/3/2020

By the following Thesis Committee:

Hichem Frigui, Ph.D., Thesis Director

Olfa Nasraoui, Ph.D.

Zhang Xiang, Ph.D.

ACKNOWLEDGEMENTS

First of all, I would like to thank The University of Louisville for offering me this opportunity to have this memorable experience and to work in these encouraging conditions.

I would like also to express my sincere gratitude to my advisor Prof. Hichem Frigui for the continuous support of this research, for his patience, motivation, and immense knowledge. His guidance helped me throughout this research and in the writing of this report.

I would also like to thank Multi-media research laboratory (MRL) members for their support during my master. MRL members were a great support for my master thesis, offering me help and guiding me with valuable advices.

Last but not least, I would never forget the encouragement and the unconditional support I have always had from my parents, my siblings and my friends.

ABSTRACT

AUTOMATIC TARGET RECOGNITION WITH DEEP METRIC LEARNING

Abddelhamid Bouzid

August 03, 2020

An Automatic Target Recognizer (ATR) is a real or near-real time understanding system where its input (images, signals) are obtained from sensors and its output is the detected and recognized target. ATR is an important task in many civilian and military computer vision applications. The used sensors, such as infrared (IR) imagery, enlarge our knowledge of the surrounding environment, especially at night as they provide continuous surveillance. However, ATR based on IR faces major challenges such as meteorological conditions, scale and viewpoint invariance. In this thesis, we propose solutions that are based on Deep Metric Learning (DML). DML is a technique that has been recently proposed to learn a transformation to a representation space (embedding space) in end-to-end manner based on convolutional neural networks. We explore three distinct approaches. The first one, is based on optimizing a loss function based on a set of triplets [47]. The second one is based on a method that aims to capture the explicit distributions of the different classes in the transformation space [45]. The third method aims to learn a compact hyper-spherical embedding based on Von Mises-Fisher distribution [64]. For these methods, we propose strategies to select and update the constraints to reduce the intra-class variations and increase the inter-class variations. To validate, analyze and compare the three different DML approaches, we use a large real benchmark data that contain multiple target classes of military and civilian vehicles. These targets are captured at different viewing angles, different ranges, and different times of the day. We validate the effectiveness of these methods by evaluating their classification performance as well as analyzing the compactness of their learned features. We show that the three considered methods can learn models that achieve their objectives.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

An Automatic Target Recognizer (ATR) is a real or near-real time, understanding system, where its input data (images, signals) are obtained from sensors and its output is the detected and recognized target. ATR has been an important task in many civilian and military computer vision applications. Examples of civilian ATR applications include tracking pedestrian [16, 35] and tracking sports players [17]. Military ATR applications are more common and include object classification based on infrared images [46, 32] and buried land mine detection [42, 18]. Two sensors are commonly used to capture data for ATR: Synthetic Aperture Radar (SAR) and Infrared (IR). These sensors offer many benefits for military applications. They enlarge our knowledge of the surrounding environment, especially at night as they provide continuous surveillance. The ultimate goal of an ATR system is to improve the awareness of the surrounding environment by exploiting information captured by these sensors.

In general, an end-to-end ATR systems includes three main successive steps. The first one, object detection, identifies the spatial location and extent of the target. This is typically represented by a tight bounding box. The second step is a low-level classification in which the ATR system tries to identify and reject false alarms. High-level classification is the third step and it is the mainstay of ATR. In this step, the detected objects are classified into predefined categories of targets.

Several ATR algorithms have been proposed in the literature. Most of them follow a standard learning based approach. This approach consists of two main steps: (1) Feature extraction, and (2) Classification. The ultimate goal of the first step is to extract discriminative features so that the targets are separable in the features space. The extracted features affect all subsequent

steps of the ATR, and thus are a crucial component of the system. For this reason, there is a wide range of proposed algorithms to extract discriminative and reliable features [55, 53, 34, 33, 30]. The classification steps learns a function that maps the extracted features of an object to class labels. Examples of methods used for classification include K-Nearest Neighbor (KNN) [15], Support Vector Machine (SVM) [6], Neural Network (NN) [46] and others [12, 8].

In the last decade, deep learning has garnered enormous success in a variety of domains. It became one of the most common scientific research trends nowadays and it is growing exponentially. This new field of machine learning has been the most successful approach for many applications and has brought revolutionary advances in very large variety of applications, including image classification [13] and image embedding [23, 52, 26, 36]. These advances were the outcomes of many different techniques that have been proposed based on different categories of learning such as supervised, semi-supervised and unsupervised learning. Deep learning techniques are evolving at a fast pace and more robust and effective algorithms are continuously being proposed. These techniques have outperformed the traditional machine learning approaches based on features extraction followed by classification.

Convolution Neural Networks (CNN) [48] are a specific class of deep learning, mainly developed for 2D imagery data. CNN are characterized by the fact that they combine feature extraction and classification. CNN have been mainly applied to optical images, such RGB, due to the availability of large data sets with high resolution. Recently, CNN were used for ATR with IR images [46, 9, 43]. CNN aim to automatically process the image and perform both the detection and recognition tasks in order to alleviate as much as possible the human intervention. However, these ATR methods can be sensitive to meteorological conditions especially in challenging weather conditions and to sensor calibration. Other challenges may be due to: targets having different views, and different scales. This can be manifested in larger intra-class variations. Moreover, different targets, at different aspects angles, can appear similar. This can be manifested in small inter-class variations.

Metric Learning aims to learn a distance function that measures the similarity between objects in order to identify their categories. The goal of this approach is to reduce the distance

between objects from the same category while increasing the distance between objects from different categories. Metric Learning has been very important for many computer vision applications such as image classification [11, 57], visual search [59] and face recognition [27]. Traditional metric learning approaches can be divided into two main categories. The first one is based on linear transformation and have some advantages such as convex formulation. Examples of such approach include metric and kernel learning using linear transformation [29] and methods based on *Mahalanobis distance* [11, 60, 56]. These have some advantages such as convex formulation. However, they have limited ability in capturing nonlinear features. To address this limitation, a second category of metric learning, based on kernels [31] has been proposed. Kernel-based metric learning provides a solution to the features nonlinearity problem. However, it has two major issues: first, it is hard to choose a good kernel for learning any distance function. Second, these kernels have limited ability to capture the nonlinearity in the data set.

Deep learning has proven to be very effective in modeling nonlinearity in many computer vision applications [47, 51, 59, 61, 40]. Motivated by this fact, the synergy between deep learning and metric learning has been explored in a new research area referred to as deep metric learning (DML). In DML, the goal is to learn the distance function using deep architectures. This new approach has been adapted to many applications such as face identification [47], image classification and image retrieval [44, 64]. DML offers many advantages such as scalability to categorizing instances with millions of classes [47] and learning high abstract fine grained features [45].

The goal of this thesis is to investigate the applications of DML to ATR. We explore three distinct approaches. The first one, is based on optimizing a loss function based on a set of triplets that should or should not be assigned to the same class [47]. The second one is based on a method that aims to capture the explicit distributions of the different classes in the transformation space [45]. The third method aims to learn a compact hyper-spherical embedding based on Von Mises-Fisher distribution [64].

To validate, analyze and compare the three different deep metric learning approaches, we use the Defense Systems Information Analysis center (DSIAC) benchmark data set. This data contain multiple target classes of military and civilian vehicles. These targets are captured at

different viewing angles, different ranges, and different times of the day.

The rest of this thesis is organized as follows: we start by giving a background and an overview of related previous work in chapter 2. In chapter 3, we outline our adaptation of the three DML methods. In chapter 4, we present our experimental results and comparative analysis of the three methods. Finally, in chapter 5, we summarize our conclusions and outline potential future work.

CHAPTER 2

RELATED WORK

In this chapter, we outline the most relevant techniques, algorithms and related work in deep metric learning (DML) and automatic target recognition (ATR) based on Infrared images. In Section 1, we start by defining DML, provide basic background, and highlight its applications in computer vision. In Section 2, we focus on the two most common approaches in DML: the first one is based on contrastive loss and the second approach is based on triplet loss. In section 3, we highlight the major limitations of these two common approaches and outline new methods that were recently proposed to overcome them. Finally, in section 4, we review the basic ATR literature.

2.1 Deep metric learning

Metric Learning aims to learn a mapping, or embedding, that quantifies the similarity or dissimilarity between objects. The ultimate goal of metric learning is to reduce the distance between objects from the same category and increase the distance between objects from different categories. Figure 1 illustrates the main objective of metric learning using three classes (each class is shown with different color/symbol). In traditional machine learning, objects are first mapped to features that capture their salient characteristics. Next, metric learning methods are applied to improve their discrimination. For instance, in the area of image analysis, SIFT [38] and LBP [1] are two common feature extraction methods. However, These techniques have showed to generate image descriptors that are not discriminative enough for real word data sets, due to two main problems. First, the limited ability of the feature extraction techniques in capturing high level abstract features. Second, the mapping learning algorithm is not an end-to-end learning which limits the ability to improve the discrimination between different classes.

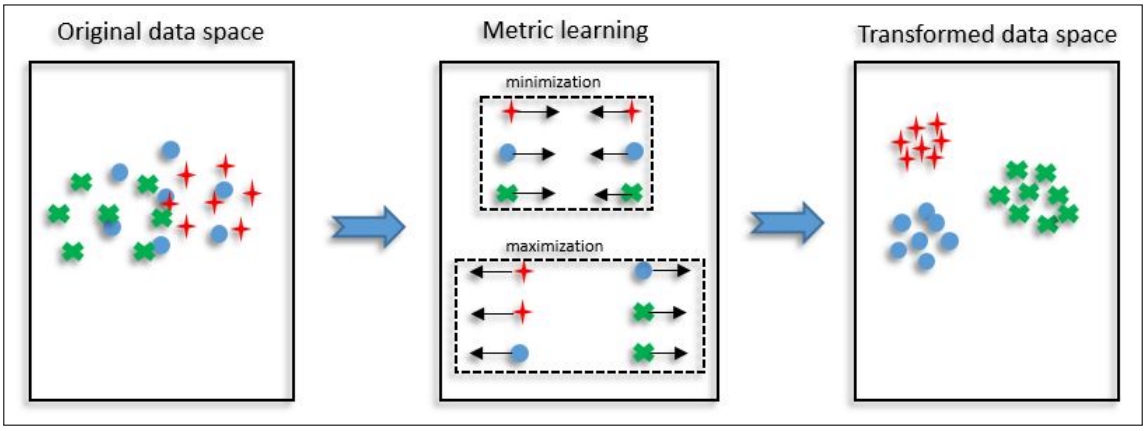

Figure 1: Illustration of metric learning for a small data with 3 classes. Samples from each class are displayed using different colors/symbols. The main objective of metric learning is to transform the features to a new space to minimize the intra-class distances and maximize the inter-class distances.

Deep learning has demonstrated to be very successful in learning high-level abstract representation of the data especially in computer vision because of its end-to-end learning mechanism. It also showed its high ability in learning the nonlinear characteristics of real-world data sets through its activation functions. Motivated by this fact and by the aforementioned issues for traditional metric leaning techniques, in recent years, deep learning and metric learning have been brought together to form a new research area, referred to as Deep Metric Learning (DML) [39]. Several studies have been conducted in the field of computer vision using this new technique [14, 10, 20]. In DML, the loss function takes advantage of the metric learning to take into account the distance between samples so that it minimizes the distance between samples from the same class and maximizes the distance between those from different classes [37, 25]. DML has contributed to many applications in computer vision such as face recognition [47], image similarity (retrieve similar images) [3], and image classification [64].

DML offers many benefits in visual understanding tasks. First, it is scalable to visual problems with very large number of classes. For example in [47], DML has been used in face identification and involved about 8 millions classes. Second, DML techniques allow the model to learn

more fine grained features, compared to standard CNNs with soft-max loss function [45]. Fine grained features are very important for visual tasks, such as, visual search [5], image retrieval [28, 44] and one shot learning [54].

To properly understand the general idea of DML approaches, we must first understand its global pipeline by going through the main steps. Most existing DML approaches follow the main steps illustrated in Figure 2 and outlined below.

1. **Sampling:** Sampling in machine learning can be done in two different ways depending on the learning paradigm: batch learning (Offline) or online learning. In particular, in DML and depending on the loss function, there are many proposed smart sampling strategies. These strategies are designed for different purposes. For example, sampling is used to speed up the convergence of methods that use loss functions that rely on a large number of tuples [47, 21, 44]. It has also been used to make the mini-batches more representative of the training data [45].

2. **Forward the mini-batch:** The second step is similar to the one in standard CNNs. After selecting a mini-batch using a sampling strategy, we forward it through the network layers (mapping the mini-batch to the transformation space). Formally, through this step the network acts as a function, f, that maps each sample I from the original space to an embedding space x, given the network parameters θ, i,e, $x = f(I, \theta)$.

3. **L2 Normalization:** this step normalizes the data in the mapped space to make the similarity invariant to the samples' magnitude.

4. **Loss function:** This step is the mainstay of DML approaches. It defines the distance function that will be learned, and thus, it affects the ability of the model to capture the relevant features. In fact, existing DML approaches are categorized based on their loss functions. In this step, the mini-batch loss is computed and based on its value, the back-propagation learning is performed.

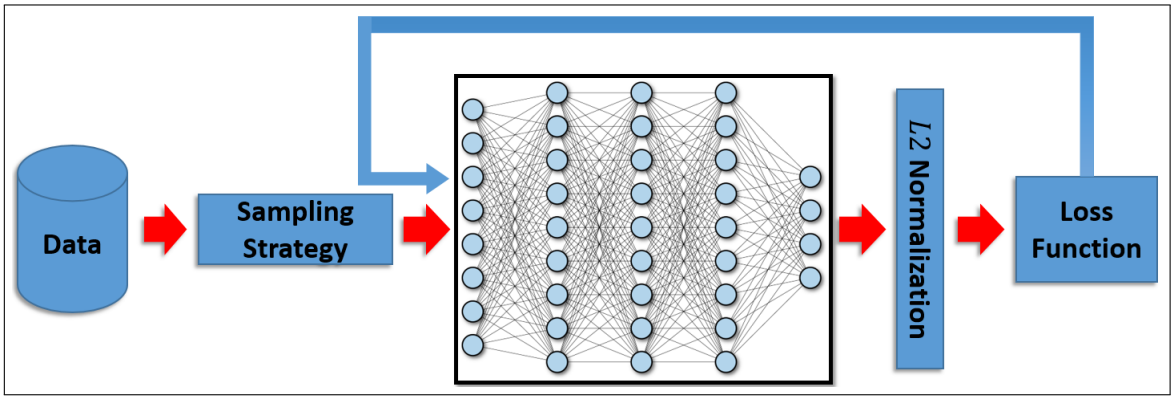

Figure 2: Illustration of the main steps involved in DML.

2.2 Deep metric learning based on contrastive loss

Contrastive loss [7] has been adapted to CNNs early in 2005 to learn a similarity metric for face verification applications. Following this initial work, many similar approaches have been developed and applied to various computer vision applications. For instance, in [3], DML with contrastive loss was used to retrieve images similar to a query image. Similarly, in [5] it was used for the purpose of visual search to identify objects in scenes and find stylistically similar ones.

2.2.1 Contrastive loss

The contrastive loss assumes that the training data consists of positive pairs and negative pairs. A positive pair has two instances, (I_q, I_p), that belong to the same class. On the other hand, a negative pair has two instances, (I_q, I_n), that belongs to different classes. Let (x_q, x_p, x_n) denote the embedding of the three objects (I_q, I_p, I_n) using the network. Then, the contrastive loss for just two pairs (one positive and one negative) is defined as:

$$L(\theta) = (1-y)\frac{1}{2}\{D(x_q, x_p)\}^2 + y\frac{1}{2}\{\max(0, \alpha - D(x_q, x_n))\}^2 \quad (1)$$

In (1), $y = 1$ if the pair is negative and $y = 0$ if it is positive. In the case where the pair is positive, the right-hand additive becomes equal to 0 and the network learns to reduce the distance between them. Similarly, if the images are from different classes, the distance between them should be maximized while keeping it smaller than a fixed threshold, α, to avoid over-fitting.

To train a network using contrastive loss, the mini-batch consists of a set of pairs (every pair is labeled as either positive or negative). The mini batch loss is the mean of all pairs losses. Figure 3 illustrates the goal of using the contrastive loss to adjust the features of the samples of a mini-batch.

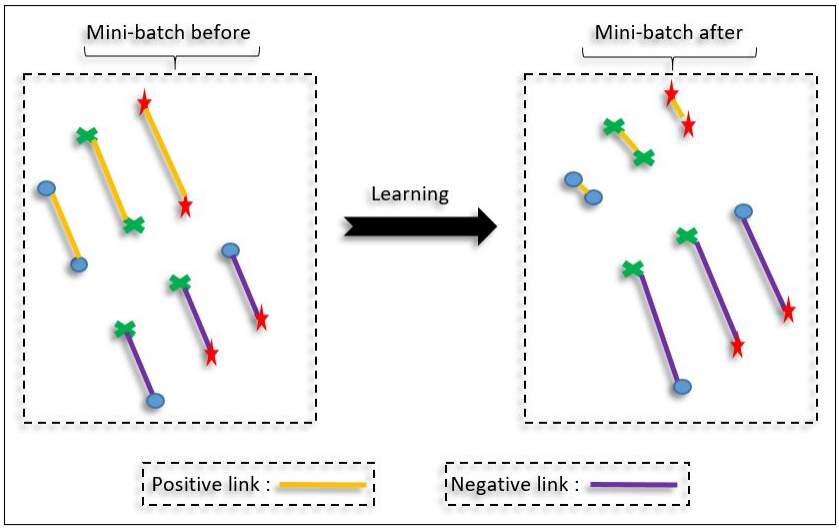

Figure 3: The objective of DML using the contrastive loss is to move positive samples closer to each-other and to move negative samples further from each-other. The sample mini-batch includes 3 positive and 3 negative pairs that are constructed from a small data that has 12 points that belong to 3 different classes. Samples from the same class are displayed using the same color/symbol.

2.2.2 Pair mining strategy selection

In pair mining, the training data is prepared as a set of pairs. Let τ as the set of all possible pairs $\{(I_q, I_p),(I_q, I_n)\}$ that can be generated from the training data set. For data with C classes, each with n_i image samples, the number of possible positive pairs that can be generated is $\sum_{n=1}^{C} \frac{n_i(n_i-1)}{2}$ and the number of possible negative pairs is $\sum_{n=1}^{C}(I-n_i)n_i$, where I is number of all images. For instance, if $I = 50,000$, $C = 10$ and $n_i = 1000$ samples per category, we end up with 500 million pairs [3]. Thus, training a network using all possible pairs for a medium or large data sets is not practical and may even be impossible. Moreover, most of the pairs may not contribute to the learning because they do not violate the loss function constraints. Therefore, sampling, where we identify a subset of relevant pairs is needed.

To identify and select relevant pairs for training, we first need to understand the intuition of easy and hard pairs that are either positive or negative. A positive pair of images is easy if the two images belongs to the same class and have a high degree of similarity using the current feature representation. Similarly, a negative pair is easy if the images belongs to different classes and have a low degree of similarity. On the other hand, a pair of positive images is hard if the two images belongs to the same class but the current feature representation leads to a low similarity. And a pair of negative images is hard if the two images have a high degree of similarity even though they belong to different classes. Formally, given an anchor x_i^a. we want to select the hardest positive x_i^p such that :

$$\arg\max_{x_i^p} \{D(x_i^a, x_i^p)\}^2 : \; hard\, positive, \tag{2}$$

similarly, we want to select the hardest negative x_i^n such that :

$$\arg\min_{x_i^n} \{D(x_i^a, x_i^n)\}^2 : \; hard\, negative, \tag{3}$$

To select a number of hard pairs we can repeat the arg max and arg min multiple times.

There have been many proposed Pair Mining strategies selection (PMSS) [3, 58]. These strategies are divided into two main categories: Online and Offline.

1. **Online:** A mini-batch of samples must be randomly selected under the condition of a minimum number of samples per class.
2. **Offline:** Generate and select the pairs after a fixed number of training steps using a subset or the whole training data set.

Typically, the choice between *online* and *offline* depends on the data set. The online option is the preferable choice for larger data sets since it can be computationally prohibitive to compute and sort the pair-wise distances between all samples. In addition, for classes with high intra-class variations, more pairs need to be identified and used, and this can be implemented more efficiently using online strategy.

2.3 Deep metric learning based on triplet loss

The triplet loss was introduced by Google in [47] for face recognition. In this work, the authors proposed a new approach to learn face embeddings using online triplet mining. The triplet loss was a way to learn good embeddings for each face. In the embedding space, faces from the same person should be close together and form well-separated clusters. This loss function has been used in many other studies in computer vision such as those in [24, 62, 22].

2.3.1 Triplet loss

Given a set of N triplets $(I_a^i, I_p^i, I_n^i),\ i = 1..N$, where for each triplet, I_a^i is a sample (referred to as anchor) that belongs to class C^i, I_p^i is a positive sample that belongs to the same class C^i, and I_n^i is a negative sample that belongs to any class other than C^i. The goal of DML using triplet loss is to map (I_a^i, I_p^i, I_n^i) to their embeddings (x_a^i, x_p^i, x_n^i) by minimizing the triplet loss:

$$L_i = \max\{\alpha + D(x_a^i, x_p^i)^2 - D(x_a^i, x_n^i)^2, 0\} \quad (4)$$

In (4), α is a margin.

The triplet loss for a mini-batch of size N is computed as the average of the N triplet losses, i.e,:

$$L = \frac{\sum_{i=1}^{N} L_i}{N} \quad (5)$$

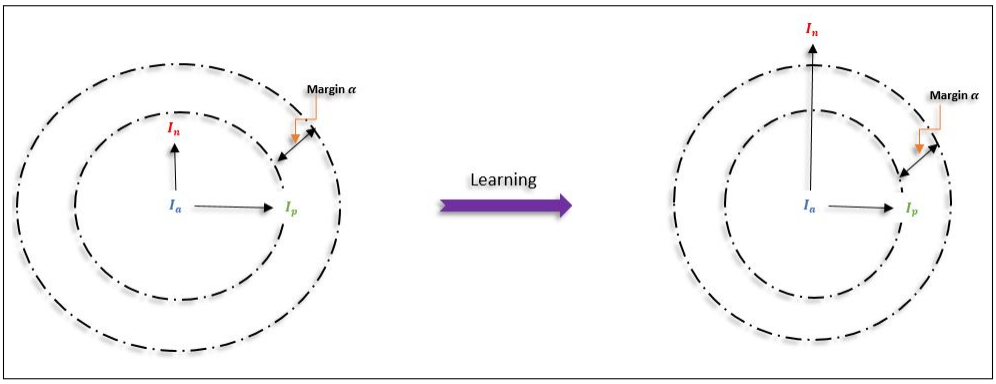

Figure 4: Illustration of the objective of learning using triplet loss.

Figure 4 illustrates learning using triplet loss, where the main goal is to move the negative sample further from the anchor to make its distance larger than the distance from the positive sample by at least a margin α. The idea behind the margin α is inspired from the margin used in SVM classifier [6], where we want to separate the classes by a margin α.

2.3.2 Triplet mining strategy selection

Similar to the pair mining strategy outlined in Section 2, for learning with a triplet loss, the training data consist of a set of triplet tuples. The number of all possible triplets that can be generated from a set of samples is extremely large even for a small data set. Thus, a strategy for triplet mining and selection is needed. This can be very similar to the one used for pairs. However, based on the definition of the loss, we define three categories of triplets:

1. **Easy triplet:** triplets which have a loss of 0 because they do not violate the loss function constraint, i.e

$$\alpha + D(x_a, x_p)^2 < D(x_a, x_n)^2 \tag{6}$$

2. **Semi-hard triplet:** triplets where the negative sample is not closer to the anchor than the positive sample, but is not far enough, i.e.,

$$D(x_a, x_p)^2 < D(x_a, x_n)^2 < D(x_a, x_p)^2 + \alpha \tag{7}$$

3. **Hard triplets:** triplets where the negative sample is closer to the anchor than the positive sample, i.e.,

$$D(x_a, x_n)^2 < D(x_a, x_p)^2 \tag{8}$$

These three categories are illustrated in Figure 5.

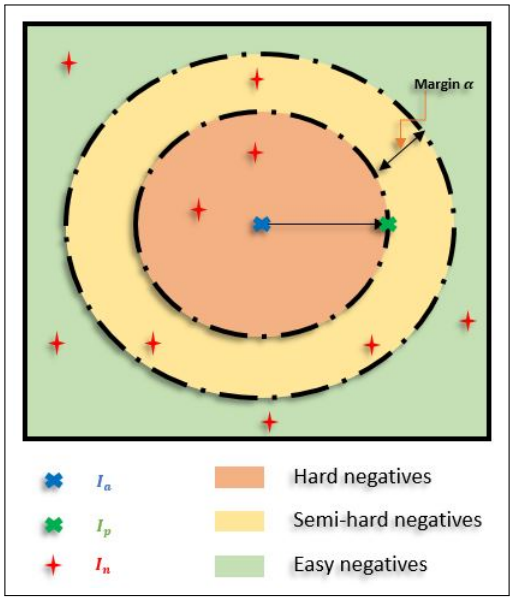

Figure 5: The three types of negatives, given an anchor I_a, and a positive sample I_p.

Two different approaches have been proposed to generate triplets [47]:

1. **Offline:** In this case, the triplets are generated offline at the beginning of each epoch or after few epochs, considering all the training data. First, we map the data using the last network checkpoint, then we select all (or a fixed random number) hard or semi-hard triplets and we continue the training process. Specifically, we generate a list of triplets (anchor, positive, negative), and partition it into batches of triplets of size B.

2. **Online:** This approach, initially proposed in [47], is based on selecting useful triplets as the training progresses, using a randomly selected mini-batch. Given a batch of B images, we first map all the B images, then we select only the semi-hard triplets. According to [47], selecting hard triplets can lead to a collapsed model.

2.4 Deep metric learning and recently proposed loss functions

The contrastive loss and triplet loss are the most common loss functions in DML. The main advantage of these two methods is that a complex deep learning architecture could be trained on a small data set. However, even for a data of a moderate size, the number of constrains that needs to be generated by these loss functions can be huge, making it hard for the network to learn and satisfy all of them. In addition, the positive and negative samples are defined by taking into consideration only the anchor and not any additional information that may exist among the positive and negative samples. This limitation is illustrated in figure 6. The above limitations can lead to learning a model that does not exploit the global space.

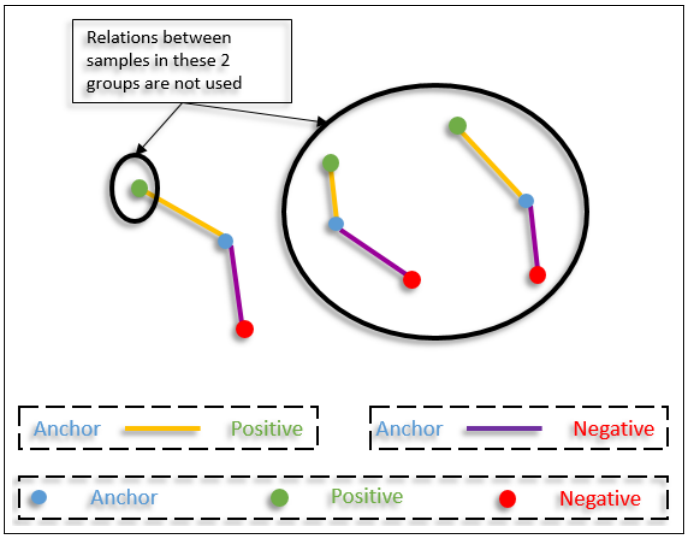

Figure 6: Illustration of the limitations of the triplet loss. Only distances indicated by (yellow and purple) lines are used for learning. Other available information (e.g between the two regions indicated by black circles) is not explored.

To overcome the above limitations, other approaches based on different loss functions have

been proposed. These approaches fall under a new sub-area in DML, called group loss, referring to the fact that they can explore similarities across all samples.

2.4.1 Augmenting the triplet loss with a global loss term

The idea of an additional global loss term was introduced in [22], motivated by the fact that the triplet loss cannot lead to a robust model most of the time, since it does not explore the global structure of the embedding space. In [22], the authors proposed adding a term to the loss function, that includes global batch information to better explore the embedding space structure. The global loss term is based on the assumption that the distance between all (anchor, positive) and (anchor, negative) pairs in a mini-batch follow Gaussian distributions. The goal of this loss term, as illustrated in figure 7, is to minimize the variance of the two distributions, and to push the mean of the (anchor, positive) distance distribution towards 0, and maximize the mean of the (anchor, negative) distance distributions.

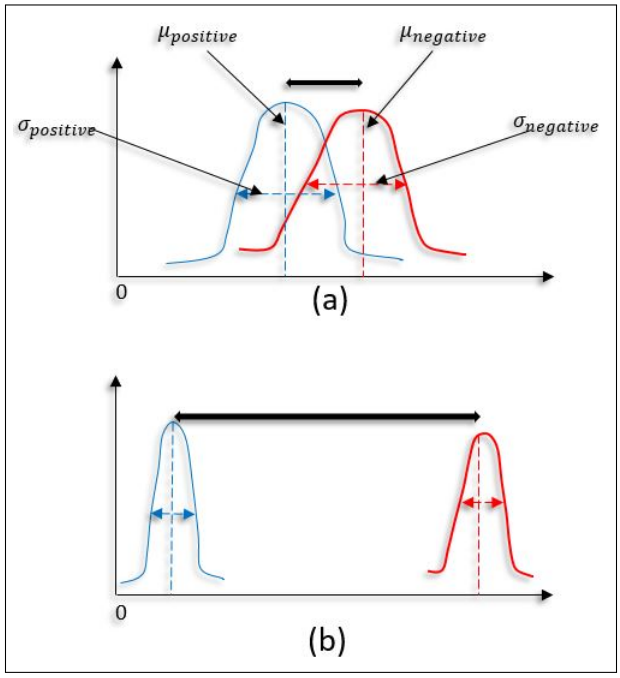

Figure 7: Illustration of the objective of incorporating a global loss term in the loss function. (a) distribution before learning. (b) distribution after learning.

The global loss term is defined as:

$$G = (\sigma^+)^2 + (\sigma^-)^2 + \lambda \max\{0, t + \mu^+ - \mu^-\}, \tag{9}$$

In (9), σ^+ and μ^+ are the standard deviation and mean of the distribution of the distances between positive pairs, σ^- and μ^- are the parameters of the distribution of the distances among negative pairs, t is a margin, and λ is a weighting term.

2.4.2 Lifted structured loss

The lifted structured loss was introduced in [44]. The basic idea is to explore information that exists among all sample pairs. Figure 8 illustrates how this loss function uses all samples in a mini-batch via pairwise distances using two samples from every class (here, we use 3 classes). Yellow lines show all possible positive pairs (i.e samples from the same class) in the randomly constructed mini-batch, where the dashed red lines shows all possible negative pairs (i.e samples from different classes). The combination of these distances can help the network to have faster and better converge.

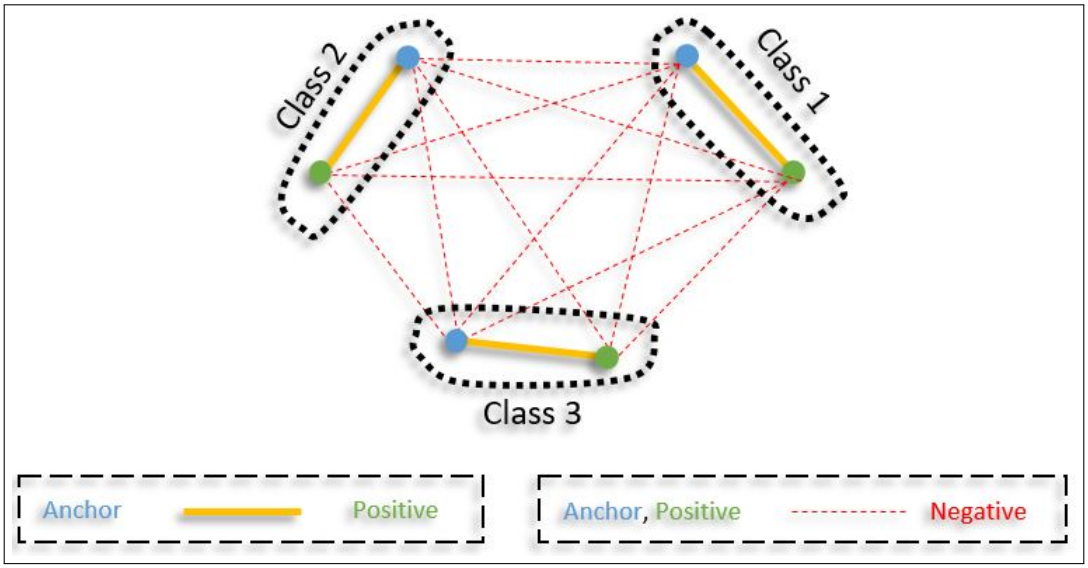

Figure 8: Illustration of the different distances involved in computing the lifted structured loss using 3 classes and 2 samples per class.

Algorithm 1 lifted structured loss

input : N: batch size, M: minimum number of samples per class

output: L: loss function value

- mini-batch = Select N random samples from the training data under the condition of a minimum number of images per any included class.
- Forward the mini-batch.
- We compute all possible positive pairs $P = \{(i, j); i, j \in c\}$, where $c \in C$.

for *every pair* $(i, j) \in P$ **do**
| Identify all negative pairs for both i and j, then compute its loss using (10).
end
Compute loss function value L using (11).

The lifted structure loss of a single positive pair (i, j) is defined as:

$$L_{i,j} = \max\left\{0, \log\left(\sum_{(i,k)\in N}\exp\{\alpha - D_{i,k}\} + \sum_{(j,l)\in N}\exp\{\alpha - D_{j,l}\}\right) + D_{i,j}\right\}^2, \quad (10)$$

Where $D_{i,k}$ is the euclidean distance between samples i and k, α is a margin, and N is the number of positive pairs. The loss of the whole mini-batch consists of averaging the loss of its P pairs, i,e:

$$L = \frac{-1}{2 \mid P \mid}\sum_{(i,j)\in P} L_{i,j} \quad (11)$$

Algorithm 1 describes the different steps used to compute the loss in (11).

2.4.3 N-pair loss

The global loss term and lifted structure loss are attempts to overcome the limitations of the triplet loss by exploiting the whole mini-batch information. However, these improvement were not sufficient for the model to fully explore the embedding space. In fact, these two loss functions were formulated based on the same online sampling of the triplet loss, which has two major problems: first, since the sampling is random, some mini-batches may not have sufficient number of triplets that violate the constraints defined by the loss function. This can lead to slow convergence. Second, the random sampling can also limit the embedding space exploration, as some samples may not be selected, or the selected samples may not reflect the distribution.

To overcome these limitations, a new loss function, named Multi-class N-pair loss was proposed in [50]. This loss function is similar to the lifted structure loss in the sense that it recruits multiple negative product examples while generating the loss term in a given mini-batch. The difference is that instead of random sampling, a new deterministic sampling technique is proposed.

The Multi-class N-pair loss is based on tuples of size $N + 1$, where N is the number of selected classes. First, 2 samples are selected from the same class. These will form the positive pair. Then, the remaining $N - 1$ samples are selected from different classes to form the negative pairs. Figure 9 illustrates the objective of learning with the N-pair loss function using a tuple of size 7. The anchor and positive are two samples from the same class and the 5 negative samples (red) are from different classes.

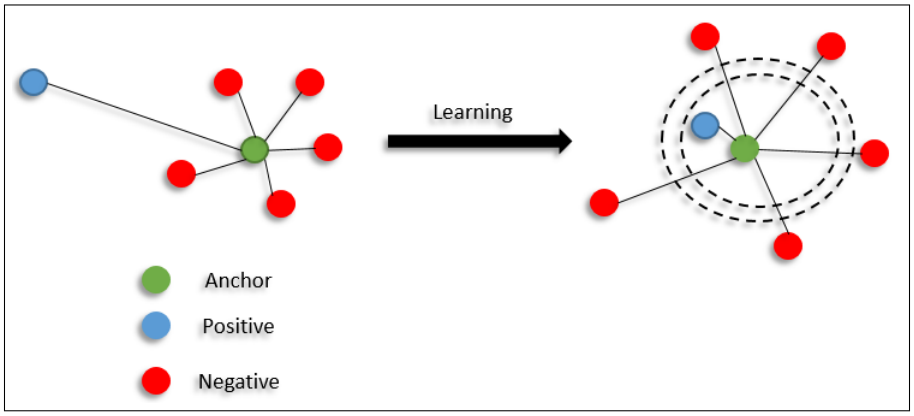

Figure 9: Illustration of the objective of learning with the N-pair Loss

Let M denote the number of tuples to be selected in a mini-batch. To construct a mini-batch, we select M tuples of size $N + 1$. Thus, a mini-batch will include $M * (N + 1)$ samples. In order to better represent the data, it is preferable to include as many from different classes as possible (i.e larger N for the tuples). To keep the size of the mini-batch within a reasonable range, the number of tuples per mini-batch M should be limited. In [50], this problem is solved using the hard mining algorithm outlined in algorithm 2:

Algorithm 2 N-pair Loss mining algorithm

input : N: tuple length, C: number of classes to be selected ($C >> N$)

output: mini-batch: $\{(x_i, x_i^+)\}_{i=1}^N$

- Select C random classes.
- Randomly select one of the C classes, c_1, and two samples from it (x_1, x_1^+).
- Select one random sample x_i from each of the remaining $C-1$ classes ($i = 2..C$).
- Using (x_1, x_1^+), greedily construct one tuple of size $N+1$ by adding the hardest negative sample x_i among $\{x_i\}_{i=2}^C$

for $i = 2$ *to* $\in N$ **do**
  select randomly another sample x_i^+ *from class* c_i.
end

Algorithm 2 will generate a mini-batch with $2N$ samples $\{(x_i, x_i^+)\}_{i=1}^N$ selected from N classes. The loss function over this mini-batch is defined as:

$$L = \frac{1}{N}\sum_{i=1}^{N}\log\left(1+\sum_{i\neq j}\exp\left((x_i^T x_j^+)-(x_i^T x_i^+)\right)\right). \tag{12}$$

2.4.4 Magnet loss

The previously reviewed DML approaches aim to project every class to a compact cluster far form the other classes by a defined margin. The tight cluster is obtained by enforcing similarity between samples from the same class, where only the class label information is considered for penalizing the similarity between samples. This kind of projection aims to improve the classification task. However, in real world data, classes can be multimodal due to large intra-class variations. In theory, the projection of these distributions of the same class into a single tight cluster limits the model ability in capturing the class diversity. This may also lead to converging to a collapsed

model with unreliable prediction credibility (correct prediction with low confidence while wrong prediction with high confidence). These issues are caused by the limited ability of these approaches in capturing the inter and intra class variations.

Motivated by the above issues, in [45] a new DML approach, based on a loss function named Magnet Loss was proposed. This loss strives to preserve the explicit distribution of the classes in the new embedding space. Its goal is to have a local discrimination by penalizing the overlap between the classes distributions. This is accomplished by penalizing the overlap between clusters from different classes using a margin α. Figure 12 illustrates the goal of learning using the magnet loss for 3 classes.

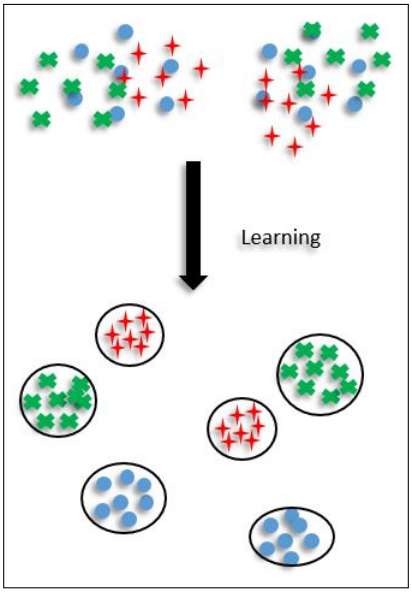

Figure 10: Illustration of the objective of learning using magnet loss with 3 classes. Here, each of the 3 classes is represented by two clusters.

Assume that the training data from every class, c, is partitioned into K clusters, and that

every cluster, $k = 1, .., K$, is represented by a Gaussian component with mean μ_k^c and standard deviation σ_k^c. The mini batch is constructed using algorithm 3.

Algorithm 3 Mini-batch selection for DML with Magnet Loss

input : M: number of clusters to be selected, D: number of samples to be selected from every cluster

output: mini-batch

1. Randomly select one cluster, c_i.
2. Select the $M - 1$ closest clusters to c_i.
3. Randomly select D samples from each of the M clusters.

Using Algorithm 3, we end up with a mini-batch that includes $M * D$ samples. The magnet loss function over this mini-batch is defined as [45]:

$$L = \frac{1}{MD} \sum_{m=1}^{M} \sum_{d=1}^{D} \max \left\{ 0, \log \frac{\exp\{-\frac{1}{2\sigma^2} \|x_d^m - \hat{\mu}_m\|_2^2 - \alpha\}}{\sum_{\hat{\mu}: C(\hat{\mu}) \neq C(\hat{\mu_m})} \exp\{-\frac{1}{2\sigma^2} \|x_d^m - \hat{\mu}\|_2^2\}} \right\}, \tag{13}$$

where $\hat{\mu_m} = \frac{1}{D} \sum_{d=1}^{D} x_d^m$ and $\hat{\sigma} = \frac{1}{MD-1} \sum_{m=1}^{M} \sum_{d=1}^{D} \|x_d^m - \hat{\mu}_m\|_2^2$ are the mean and the standard deviation of cluster m, estimated using the subset of sampled data, and α is the margin between the classes.

2.4.5 Von Mises-Fisher loss

Heretofore, we have reviewed several DML approaches, where every one of them tries to overcome one or several issues of its previous methods. Still, there are some issues that are not

solved yet. These can be categorized as:

- Most of the loss functions are based on preparing the data in formats of tuples (triplets, pairs or N-pairs), which is computationally expensive and time consuming.
- Results are obtained from a limited number of selected samples during the training phase (to form tuples or represent clusters). This may lead to learn a model that lacks robustness and global view of the embedding space.

To address the aforementioned issues, in [64] a new DML approach that uses a novel loss function named, Von Mises-Fisher (VMF), has been proposed. This loss function can be seen as the Gaussian distribution for a unit spherical data. In addition, a new efficient learning algorithm that does not require the preparation of the data in any specific format (Cluster or tuples) was proposed. This new learning algorithm is similar to the offline batch learning using CNNs with soft-max loss function. In this approach, the Euclidean distance is replaced with the cosine distance.

The VMF models each class by a Von Mises-Fisher Distribution, which is a probability distribution in directional statistics for spherical data, defined as:

$$f_p(x;\mu,\kappa) \quad = \quad Z_p(\kappa)\exp(\kappa\mu^T x). \tag{14}$$

In (14), $\|\mu\| = 1$ is the mean direction, κ is a concentration parameter, p is the space dimension, $Z_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)}$ is a normalization term and I_v is the modified Bessel function of the first kind with order v. The parameters of this distribution are estimated for N samples using:

$$\hat{\mu} \quad = \quad \frac{\sum_{i=1}^{N} x_i}{\|\sum_{i=1}^{N} x_i\|}, \tag{15}$$

and

$$\hat{\kappa} = \frac{\overline{R}(p - \overline{R}^2)}{1 - \overline{R}^2}. \tag{16}$$

In (16), $\overline{R} = \frac{\| \sum_{i=1}^{N} x_i \|}{N}$.

The learning problem is defined as the following. Given C classes, the goal is to learn C VMF distributions (a distribution for every class c) parameterized by $\{\kappa_i, \mu_i\}_{i=1}^{C}$. In this probability space, the normalized probability of a mapped sample x belongs to class c is:

$$P(c|x, \{\kappa_i, \mu_i\}_{i=1}^{C}) = \frac{Z_p(\kappa_c) \exp(\kappa_c \mu_c^T x)}{\sum_{i=1}^{C} Z_p(\kappa_i) \exp(\kappa_i^T \mu_i x)} \tag{17}$$

Equation (17) can be used to maximize the probability that the sample belongs to the true class and to minimize the probability that it belongs to other classes. Given a mini batch of N samples, the VMF maximizes.

$$P(Y|X, \Theta, \cup, \kappa) = \prod_{n=1}^{N} P(c|x, \{\kappa_i, \mu_i\}_{i=1}^{C}) \tag{18}$$

$$= \prod_{n=1}^{N} \frac{Z_p(\kappa_c) \exp(\kappa_c \mu_c^T x)}{\sum_{i=1}^{C} Z_p(\kappa_i) \exp(\kappa_i \mu_i^T x)}, \tag{19}$$

Where X and Y represent the training samples in the mini-batch and their labels, Θ contains the deep model parameters, and $\cup = \{\mu_i\}_{i=1}^{C}$, $\kappa = \{\kappa_i\}_{i=1}^{C}$. By applying the negative likelihood, and letting κ be a constant for all classes, (17) can be simplified to:

$$\arg\min_{\Theta, \cup} L = -\sum_{n=1}^{N} \log \left(\frac{\exp(\kappa \mu_c^T x)}{\sum_{i=1}^{C} \exp(\kappa \mu_i^T x)} \right) \tag{20}$$

In [64], the authors proposed optimizing (20) using an alternative learning algorithm, where they fix the mean directions and train the model for several iterations, then update the mean

directions using the training data set. The learning Algorithm is summarized in algorithm 4.

Algorithm 4 VMF learning algorithm

1. Initialize CNN parameters Θ.
2. Repeat:
   (a) Estimate mean directions using (15) and all the training data.
   (b) Train CNN for several iterations and update Θ.
3. Until convergence.

2.5 Automatic target recognition

2.5.1 ATR challenges

Automatic target recognition (ATR) is the task of recognizing targets using data obtained from sensors. In our work, we address the problem of ATR based on infrared images. Infrared imaging is a common sensor in military applications, where the goal is to detect and identify targets. These ATR systems are platforms that aim to quickly process incoming image data captured by sensors and accurately report the results within real-time. They consist of user (human) and operator (recognition algorithm), where the goal is to minimize the human intervention as much as possible. To address this problem, many effective ATR methods have been proposed [46, 9, 43]. These methods aim to automatically process the image and perform detection and recognition tasks in order to alleviate the human intervention as much as possible.

ATR methods suffer from major issues. First, they are not robust to meteorological conditions, especially in challenging weather, which highly affects the image quality. In such conditions, images of the same target may look very different. Second, sensor calibration is another factor that affects the image quality. The effect of these two issues is amplified for targets at further distance,

where objects are captured by only few pixels. Third, targets from the same class can have different views and different scales leading to large intra-class variations. Finally, different targets can appear similar at different aspect angles leading to small inter-class variations.

2.5.2 Deep learning for ATR

The ATR literature consists of a wide range of approaches that ranges from learning based to model based to CNN based architecture. In this work, we focus on deep learning as it proved to be the most effective approach [9].

Despite the tremendous success of CNN in visions tasks based on optical images, such as RGB, there are a few applications of CNN to thermal data [2, 19]. The main reason for this is the lack of publicly available large thermal data sets. These few CNN applications tried to address ATR problem issues and challenges with different ways. In 2016, [46] applied CNN to perform ATR on long-wave infrared images in an end-to-end learning manner. In this work the authors sought to create a long wave infrared target classifier based on CNN using a range of objects. Recently, [9] proposed a CNN architecture with a global average pooling layer (GAP). This CNN architecture was trained on simulated data and validated on real data using a range of targets. In this study, the authors conducted several experiments to stress and study the robustness of the proposed model especially with respect to the viewpoint invariance factor. Their architecture was based on a previous work [4] that claims that the last fully connected layer is the layer that mostly enforces the viewpoint invariance, which is the reason for choosing GAP. In 2019, [43] proposed CNN architectures for ATR. These CNNs are based on VGG architectures [49], which they claim that they achieve comparable results with state of the art methods.

Existing deep learning approaches to ATR using infrared images are based on standard traditional CNN architectures followed by fully connected layers with a classification layer (soft-max most of the time). In this thesis, we investigate the applications of DML to ATR. We explore three distinct approaches. The first one, is based on optimizing a loss function based on a set of triplets that should or should not be assigned to the same class [47]. The second one is based on a

method that aims to capture the explicit distributions of the different classes in the transformation space [45]. The third method aims to learn a compact hyper-spherical embedding based on Von Mises-Fisher distribution [64].

CHAPTER 3

ADPATATION AND APPLICATION OF DML TO ATR

In this chapter, we describe our adaptation of three DML approaches to ATR. The first one is based on the triplet loss that was outlined in Chapter 2.3. For this approach, we compare the original algorithm [47] and the extension that uses an additional global loss term [22]. The second method is based on the magnet loss described in Chapter 2.4.4. For this approach, first we present our adaptation of the magnet loss to ATR. Then, we illustrate the learned clusters that characterize different categories to capture the intra-class variations. The third method is based on the VMF loss that was outlined in chapter 2.4.5. For this method, we analyze the learned distributions of the classes. To analyze and compare these three methods, we designed several experiments to gauge a variety of metrics ranging from classification performance to clustering and visualization of the learned features. In the following, we first describe the data set used in our experiments and our processing. Then, we describe our adapted algorithms and analyze their performance.

3.1 Experimental data

3.1.1 Data Description

The data set used in our experiments was made available by the Defense Systems Information Analysis center (DSIAC) [1]. This data contain a large set of mid-wave infrared (MWIR) imagery collected by the US Army Night Vision and Electronic Sensors Directorate (NVESD). The targets within this data include: "person", eight military vehicle targets ("BTR70", "BRDM2", "BMP2", "T72", "ZSU23", "ZS3", "MTLB", "D20"), and two civilian vehicles ("PICKUP", "SUV"). These targets were imaged continuously (collected and stored as videos), at different aspect angles, at

[1] https://www.dsiac.org/

ranges varying from 1000 to 5000 meters with a step of 500 meters. The different aspect angles were accomplished by making the targets move in a circular path at a fixed speed for each range. Figure 11 illustrates the path traveled by one target for a sample video.

The data was collected during different times of the day. For our analysis, we quantize the time into 2 levels: day (from $7a.m$ to $5p.m$) and night. We also quantize the aspect ratio of the targets into 4 zones as illustrated in figure 11. Each frame has 512 rows and 620 columns. Within each frame, the target is included within a small region with a size that varies from [4 rows, 6 columns] to [30 rows, 80 columns]. Each video file is associated with a ground truth file that includes the target name and its bounding box within each frame. These truth files were used to extract image patches that include only the targets for training. They are also used to validate our results and score them.

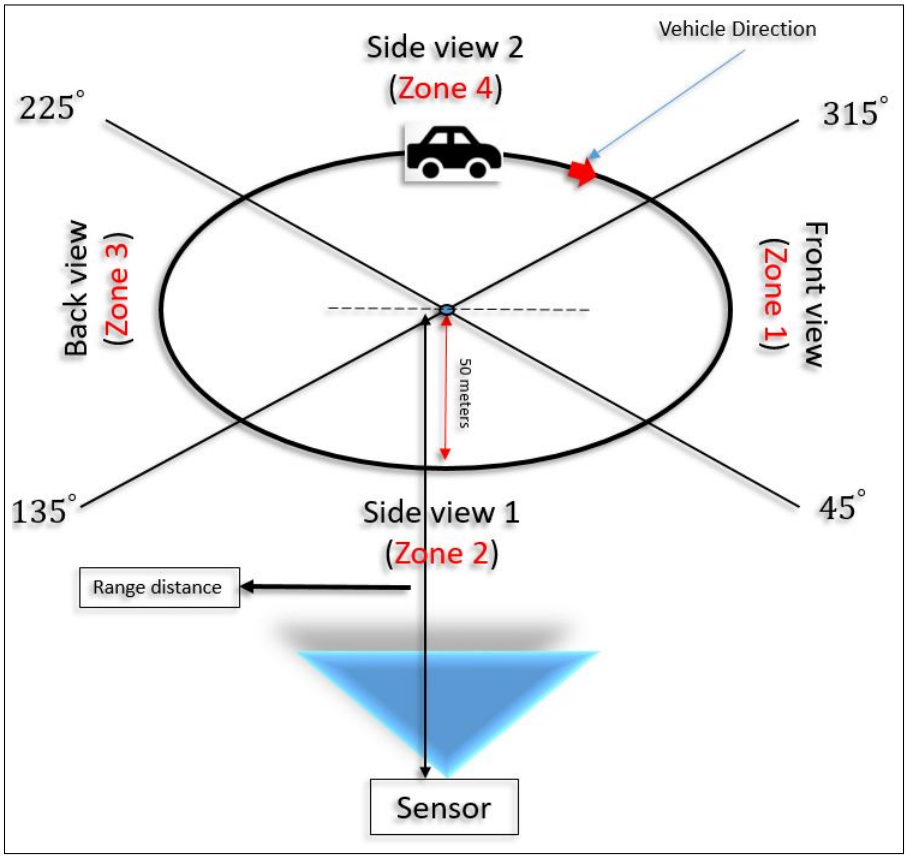

Figure 11: Illustration of the trajectory of moving target and quantization of the viewing angle into 4 zones.

In all our experiments, we only consider the civilian and military vehicles. We also consider only targets that are at ranges within 3500 meters. Targets at higher ranges are captured by very few pixels and cannot be identified. The inclusion of these target may confuse the training algorithms. For each target and at each range, two videos files are available: one during the day and one at night. Each video consist of 1800 frames that capture the motion of the target as illustrated in figure 11.

All methods are trained and tested under the same conditions. For training, we used all

ranges between 1000 and 3500 meters except 1500 meters which was reserved for testing. With this set up, for each target, we have 10 videos for training: 2 videos at each range (one during the day and one during the night). For testing, only 2 videos per target (one during the day and one during the night) were used.

The DSIAC data have high temporal resolution as it was recorded at a frame rate of $30Hz$ and the targets were moving at a relatively slow speed. Thus, successive frames have very similar content. Since The DML methods based on the magnet and triplet loss are computationally intensive, for these methods we sample the training data by considering only every 9^{th} frame. This results in reducing the training data from $180,000$ training samples ($18,000$ samples per target) to $20,000$ samples (2000 samples per target). The DML based on VMF is computationally efficient and no sampling is used to train this algorithm.

3.1.2 Data preprocessing

The pixel values of infrared images have a wide dynamic range and their distribution can have very long tails. To normalize these values to a $[0, 1]$ range and maintain the image contrast, we transform the pixel's values using the following two steps:

1. **Clipping :** To eliminate the long tails of the pixel's intensity distribution, we identify a lower threshold that corresponds to the 0.1 percentile of the distribution and an upper threshold that correspond to the 99.9 percentile. Pixel values below or above these thresholds are clipped to these values.

2. **Normalization :** The median and the Median of Absolute Deviation (MAD) are robust statistical measures that are more robust to outliers than the mean and standard deviation. For each frame, we compute these measures and normalize each of its pixel values, X, using:

$$X_{normalized} = \frac{X - median(X)}{MAD(X)}, \tag{21}$$

To avoid long tails that were not clipped in the first step, the normalized values are clipped to remain within the $[-5, 10]$ range.

After preprocessing each frame, image regions that include targets are cropped using the ground truth bounding box. Each of these images patches is resized to 32×32, then duplicated 3 times as an RGB image. Each target is then saved as a $32 \times 32 \times 3$ image.

3.2 Architecture of our approach to ATR using DML

Figure 12 illustrates the four main components of our DML adaptation to ATR. The first one, (a), is the data preparation. The goal of this component is to prepare the data in a specific format such as triplets, clusters or mean directions computation for the triplet, magnet or VMF loss, respectively. These data are updated by the learning algorithm after several iterations or epochs. The second component, (b), is the mini-batch sampling strategy in which an algorithm is used to sample mini-batches from the prepared data. The goal of this component is to sample mini-batches in a way that is consistent with the loss function. Notice that the learning algorithm and the mini-batch sampling can be combined under one single algorithm. The third component, (c), is the CNN architecture which is the embedding function in DML. In all of our experiments, we used a Wide Residual Network (WRN) [63], where the embedding space dimension $p = 128$. The last component, (d), is the loss function component that determines the learning objective.

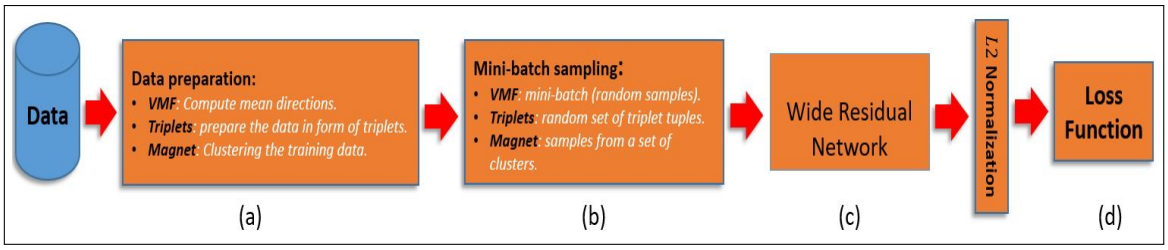

Figure 12: Different components of our proposed DML approach to ATR.

3.3 Adaptation of the triplet loss to ATR

3.3.1 Triplet Generation

Figure 13 illustrates the objective of the proposed sampling strategy to generate triplets. For a given anchor A, we check the labels of its KNN samples. We note a triplet as (A, N^{k_i}, P^{k_j}) where the N^{k_i} is the $(k_i)^{th}$ closest sample to the anchor A and has a different label, and P^{k_j} is the $(k_j)^{th}$ closest sample to the anchor A and has the same label. The triplet (A, N^{k_i}, P^{k_j}) is considered hard and selected for training if $k_i < k_j < K$. It is expected, after learning based on the triplet loss, that the positive samples will move closer to the anchor, while the negative samples are pushed away from it.

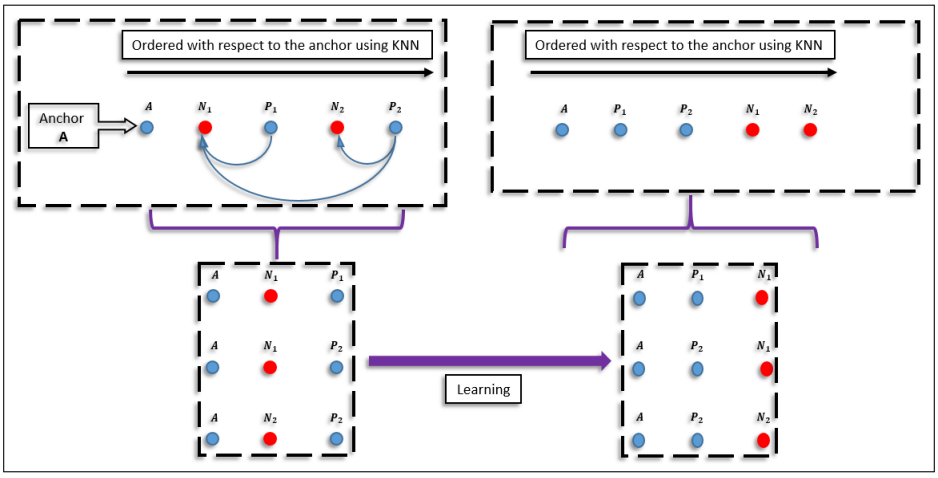

Figure 13: Illustration of the selection of triplets for the triplet loss method and the objective of this approach. A is the anchor, N^{k_i} and P^{k_j} are the nearest samples to A, where, N^{k_i} is a negative sample, that is the (k_i^{th}) closest to A and P^{k_j} is a positive sample at rank k_j to A. Initially $k_i < k_j < K$ the goal of learning is to have $k_j < k_i$ for the maximum number of triplets.

To generate a set of triplets for training, we propose an offline learning strategy based on KNN. Given the training data samples $\{X_1, .., X_N\}$, first, we process them by the CNN to map them to features. Second, we compute the pairwise distance between all the training features. Third, for each training sample X_i, we identify its K Nearest Neighbors, denoted X_i^{KNN}. Using X_i as an anchor and its X_i^{KNN}, we generate a set of triplets $X_i^{triplet}$ using all hard triplets as defined in (8). Algorithm 5 summarizes our proposed offline method for generating the triplets. This strategy is efficient for two main reasons. First, all the training samples contribute to learning. This leads to learn a model that has a better global view of the embedding space. Second, using only hard triplets generated from the KNN of each sample (anchor), limits the number of triplets that do not contribute to the learning and speeds up the convergence. Algorithm 6 outlines the different steps of our adaptation of the DML algorithm based on the triplet loss function.

Algorithm 5 Hard triplet generation based on KNN

---

input : K: number of nearest neighbors.

N: training samples $\{X_1, .., X_N\}$

output: $X^{triplet}$: Set of triplets.

1. Process the training data by the CNN to map them to features.
2. Compute the pairwise distance between all training features.
3. $X^{triplet} = \emptyset$.
4. **for** *every sample* X_i **do**
   - Identify the K nearest neighbors of X_i. Let X_i^{KNN} denote this set.
   - Using X_i as an anchor A and its X_i^{KNN}, generate a set of triplets $X_i^{triplet}$ using all hard triplets (A, N^{k_i}, P^{k_j}), where $k_i < k_j < K$.
   - $X^{triplet} = X^{triplet} \cup X_i^{triplet}$.

   end

Algorithm 6 DML for ATR based on triplet loss

input: B: batch size.

1. Initialize CNN parameters Θ with random weights.
2. Repeat until model converge:
   (a) Generate a set of triplets $X^{triplet}$ using algorithm 5.
   (b) Divide $X^{triplet}$ into mini batches of B triplets.
   (c) **for** *every mini-batch* **do**
   - Foreword the mini-batch through the network.
   - Compute the mini-batch loss using (5).
   - Update CNN weights based on the the mini-batch loss

   end

3.3.2 Evaluation strategy

We evaluate this method using KNN, where for a given query, we retrieve its nearest neighbors from the training data using their transformed features. We use the Euclidean distance for computing the distance between samples. The reason behind choosing KNN for evaluation is that it is consistent with the learning procedure and the sampling strategy we used it to generate triplets.

3.3.3 Parameters settings

Video sampling:	every 9^{th} frames
Triplet generation:	all violations within $K = 50$ of the nearest neighbors
Number of epochs:	1
The number of triplet in a mini-batch size:	500
Number of iterations:	1000
Learning rate:	10^{-3}
Margin:	$\alpha = 0.2$
Optimizer:	Adam
λ of Global loss term parameters:	1
Margin of Global loss term parameters:	0.01

Table 1: Triplet model training parameters

Table 1 contains the training parameters for the triplet method. In our experiments, we observed that after 1 epoch, which includes 1000 iterations, training did not improve the performance of the learned model. Thus, we fix the number of epochs to 1. For the triplet method, it is important to train with a large batch size. Thus, we fix the number of triplets in a mini-batch to 500. Our experimental analysis revealed that this number was sufficient for training. We set the margin $\alpha = 0.2$ as reported in the original paper [47]. For the global loss term, we used the same parameters as reported in [22].

3.3.4 Experimental results

We used KNN with $K = 15$ to compute the classification accuracy during the training phase and verify that the network is learning. We compute the training and testing accuracy at every iteration, after the forward and the backward passes of every mini-batch. Figure 14 shows the training and testing accuracy of the two considered models. The first one is trained based only

on the triplet loss (TL) while the second model is trained based on the triplet loss plus a global loss term (TL^{+G}). This plot illustrates the improvement induced by adding the global loss term. First, there is a significant improvement in terms of classification accuracy. Second, the testing accuracy of TL model is fluctuating from one mini-batch to another, which is an indication that the model is not robust and is passing through many sharp local minima. The addition of the global loss term resolves this issue. Third, the TL^{+G} model converges faster than TL as the number of iterations could be reduced by a factor of 4.

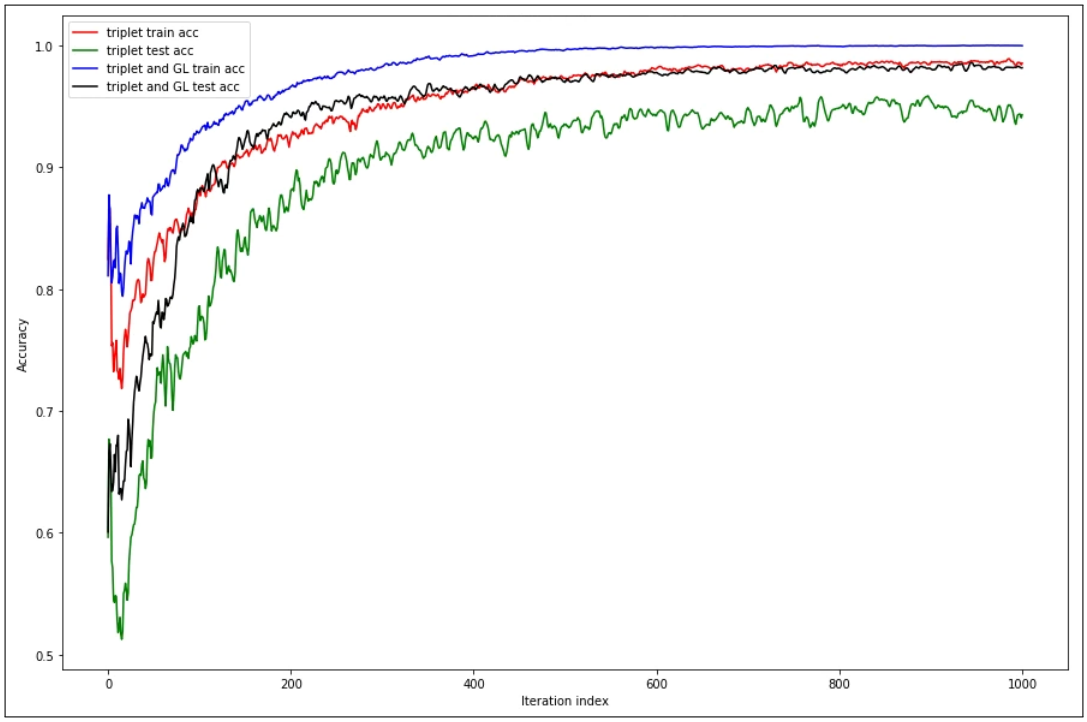

Figure 14: Classification accuracy using KNN (with $K = 15$) of the train and test data sets using TL and TL^{+G} models.

Our proposed triplet generation for this method is based on KNN as illustrated in figure 13. To validate this strategy, we designed an experiment where at every iteration, we compute

the number of unsatisfied triplets that can be generated by our algorithm, using models learned at different checkpoints. A triplet is unsatisfied if it violates the constraint defined by (6). Figure 15 displays the results of this experiment for both models (TL and TL^{+G}). The number of unsatisfied triplets decreases sharply during the first few iterations and remains constant after 400 iterations. This can be explained by the ability of both models to quickly learn most of the easy triplets. After 400 iterations, some of the hard triplets could not be satisfied by the TL model. By adding the global loss term, these hard triplets were satisfied by the TL^{+G} model decreases faster. This supports our claim about the advantages of adding the global loss term.

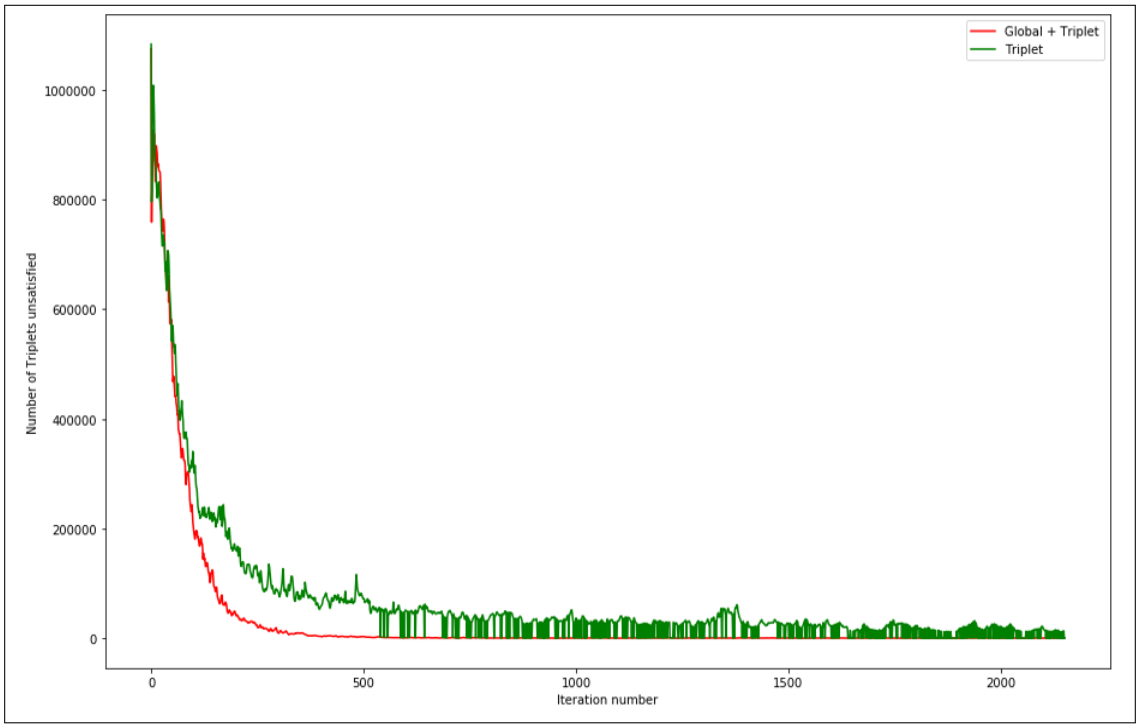

Figure 15: Number of unsatisfied triplets at every iteration during training of the TL and TL^{+G} models.

The overall decreasing in the number of unsatisfied triplets for both models is the results

of the change in the nearest neighbors for every sample, where more samples from the same class get closer to each other and negative samples get pushed away. Figure 16 illustrates what happened to a sample anchor and its nearest neighbors between two checkpoints. In this figure, labels of KNN that belong to the same class as the anchor are displayed in blue while labels of the negative samples are displayed in red. As it can be seen, after 10 iterations, 10 negative samples have disappeared from the top 15 neighbors and were replaced by positive samples.

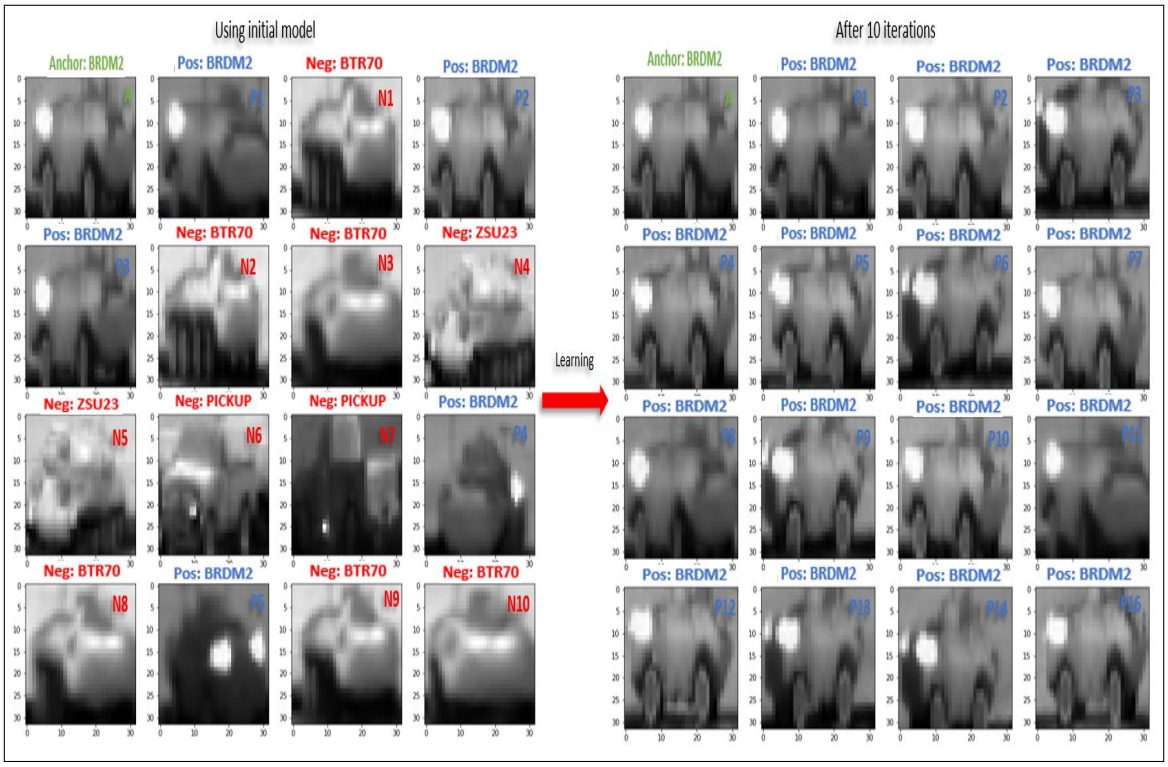

Figure 16: 15 Nearest neighbors of an anchor from the BRDM2 class using the initial TL model and the model after 10 iterations.

Another way to demonstrate that the TL^{+G} method based on our sampling strategy leads to learn better representation of the classes as we iterate, is to analyze the evolution of the learned

features in the embedding space during training. Since the learned features are high dimensional, we first project them to a lower dimensional space so that they can be visualized. Here, we used T-SNE [41], which was shown to be one of the most effective methods to project high dimensional data to a much lower space. Figure 17 illustrates the $2D$ projection of the training data at 4 different stages ((a): initial model, (b): after 100 iterations, (c) after 300 iterations and (d): after 999 iterations). It is clear that as we iterate, samples from the same class are getting closer to each other, while samples from different classes are pushed away. We notice also that a margin between all the classes is established as we iterate. This experiment shows that the TL^{+G} based on our proposed learning strategy leads to learn a model that achieved the objective of the DML based on triplet loss.

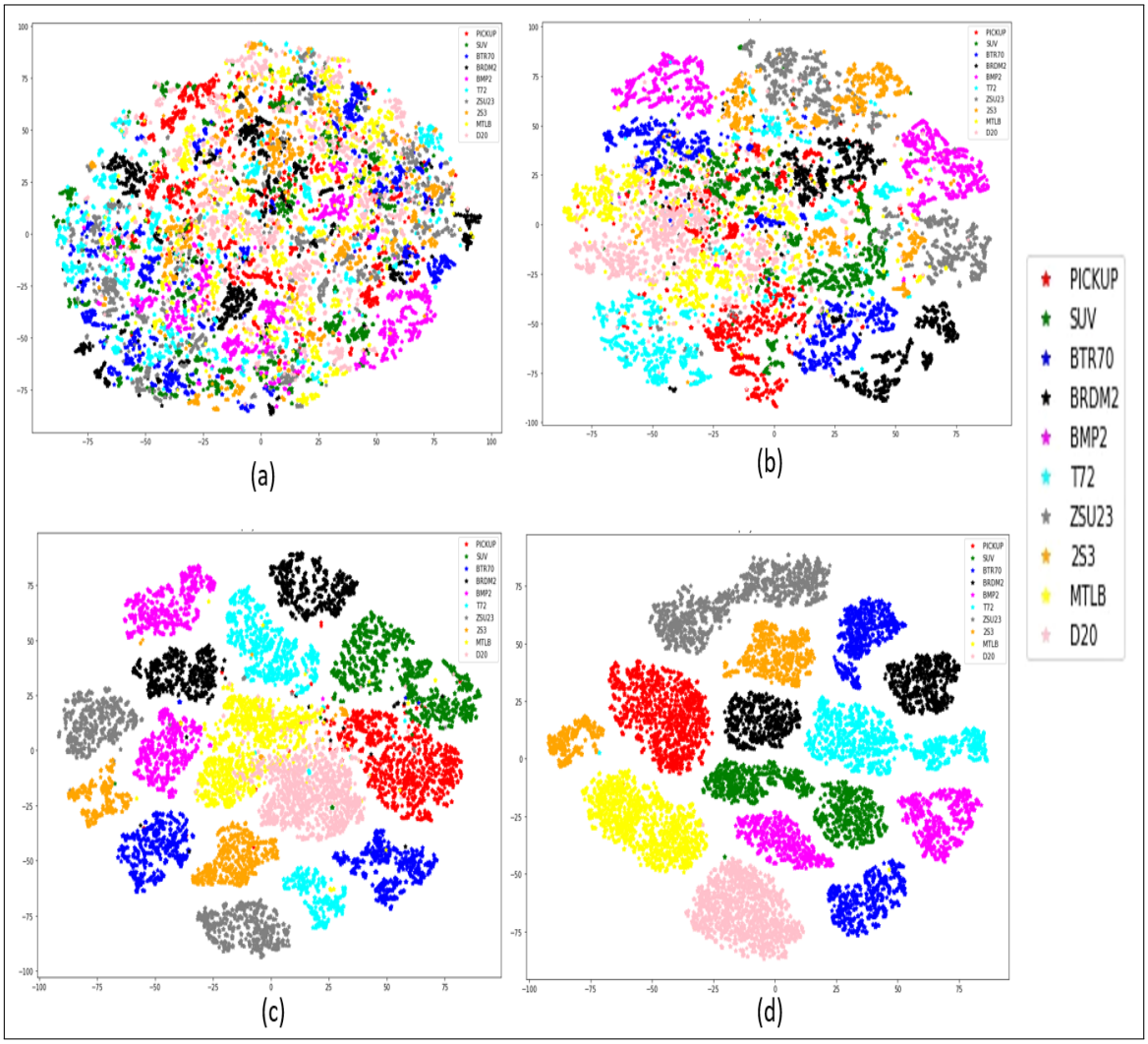

Figure 17: $2D$ T-SNE projection of the training data after: (a): initial model, (b): after 100 iterations, (c): after 300 iterations and (d): after 999 iterations of the training TL^{+G}.

To illustrate that the proposed TL^{+G} learns features that result in compact clusters in the high-dimensional space, we compute the pair-wise cosine similarity, for all the training data. Then, for each class, we compute a histogram of its intra-class similarity (similarity between all samples within the same class) and inter-class histograms (one for each different class). The results are

shown in figure 18 for each of the 10 targets. As it can be seen, the distribution plot of intra-class similarity for all classes are close to 1, indicating that the clusters are tight and most of the class samples are projected into a small area in the embedding space. The plots also show that there is a significant margin between every class and the other classes, indicating that there is no overlapping between the classes distributions.

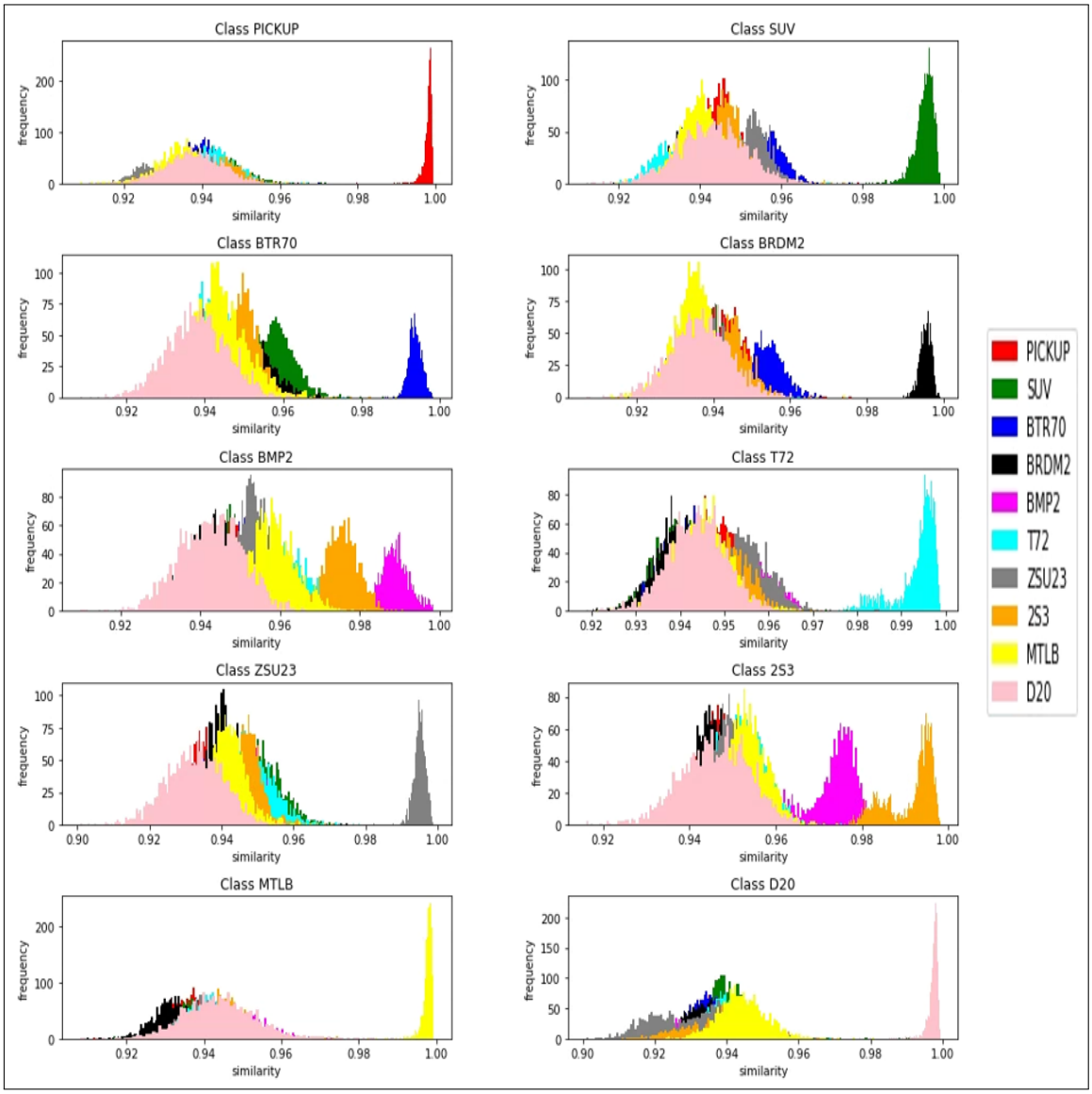

Figure 18: Histograms of inter and intra class similarities for features learned based on triplet loss plus global loss.

3.4 Adaptation of the magnet algorithm to ATR

We have adapted the DML with the magnet loss [45], described in section 2.4.4, to ATR using IR images. First, for selecting the $M-1$ nearest clusters of every anchor cluster from a class c (step 2 in algorithm 3), we assure that at least one third of them are negative (from classes different from c). This condition can help in reducing the inter-class similarities. Second, we ensure that all the clusters contribute to the learning by selecting different clusters as anchors for each mini-batch. After every epoch, we recluster the training data.

3.4.1 Evaluation strategy

The performance of this method is evaluated using two different strategies. The first one is based on the KNN as described in section 3.2.2. The second one is based on the evaluation metric used in [45]. Since this method aims to learn the clusters' representation of every class, its performance can be evaluated by the accuracy of the class label predicted using only the clusters centers rather than all training samples. Specifically, for a given test sample x_n, we identify its L nearest clusters. Let $\mu_1, ..., \mu_L$ denote these clusters centers and let $C(\mu_l)$ denote the class label of each cluster. The likelihood that the query belongs to a class c depends on the number of retrieved clusters that belong to the same class and on its distance to each one of them. Formally, the prediction label of a sample x_n is defined as:

$$c_n^* = \underset{c=1..C}{\arg\max} \frac{\sum_{\mu_l:C(\mu_l)=c} \exp(-\frac{1}{2\sigma^2}\|x_n - \mu_l\|_2^2)}{\sum_{l=1}^{L} \exp(-\frac{1}{2\sigma^2}\|x_n - \mu_l\|_2^2)}, \tag{22}$$

where σ is a running average of stochastic estimates $\hat{\sigma}$ computed during training.

3.4.2 Parameters setting

The training parameters for DML using magnet loss are listed in table 2. In our current implementation, the number of clusters needs to be fixed for each class. The optimal number is not critical, but if it is too small, then there is not enough number of prototypes to represent the intra-class variations. On the other hand, if the number of clusters is too large, outliers in the training data may be identified as representatives. This will affect the generalization of the network and slow down the testing phase. As a compromise, in our experiments, we fix the number of clusters per class to 15. Our experimental analysis revealed that this number was sufficient for all the classes within our data. The number of selected clusters per mini-batch, M, was set to 16, and the number of selected samples from every cluster, D, was set to 16. We set the margin $\alpha = 1$ as reported in the original paper [45]. The network was trained for 500 epochs.

Video sampling:	every 9^{th} frame
Number of epochs:	500
Learning rate:	10^{-4}
Number of clusters per class:	$K = 15$
Number of different cluster per mini-batch:	$M = 16$
Number of samples from every cluster:	$D = 8$
Margin:	$\alpha = 1$
Optimizer:	Adam

Table 2: Magnet model training parameters

3.4.3 Experimental results

To illustrate the ability of the model to learn, we report the classification accuracy during training using KNN with $K = 15$. Figure 19 plots the training and testing classification accuracies. We notice that the classification accuracy increases sharply during the first few epochs, then it slows

down drastically after few epochs. We also notice that the model based on the magnet loss converges much faster than the model based on TL^{+G} under the same conditions as shown in figure 19.

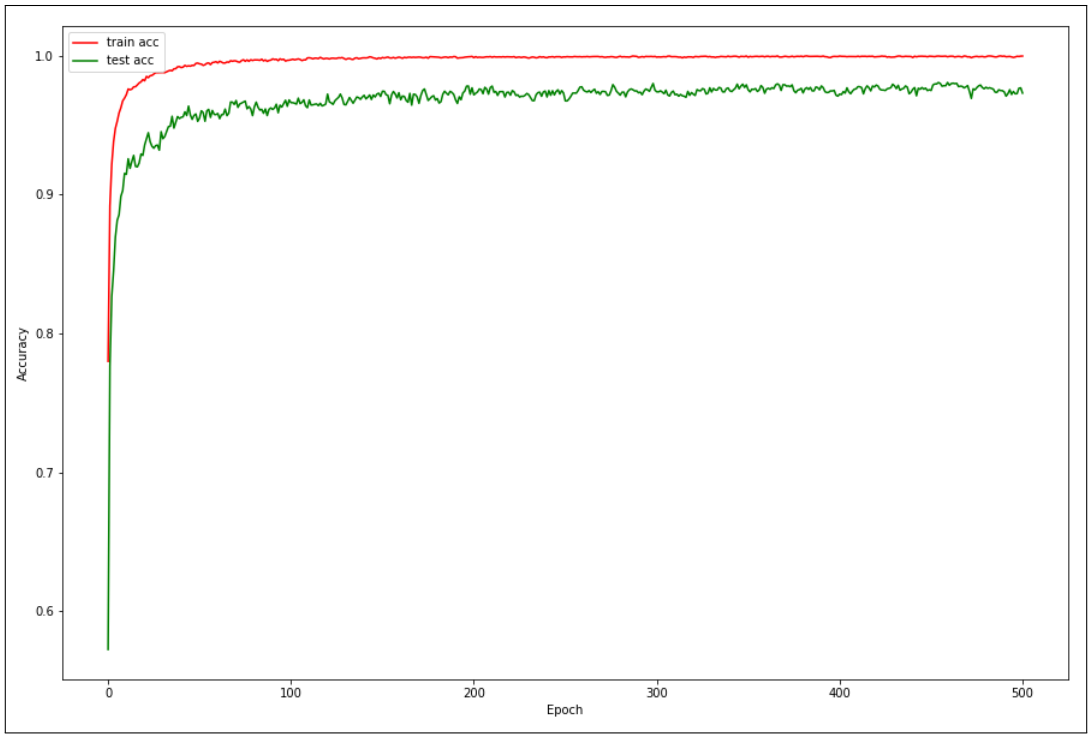

Figure 19: Classification accuracy using KNN of the train and test data sets using DML with magnet loss.

To illustrate the information captured by the learned clusters within each class, in figure 20, we display a $2D$ projection of the learned features obtained using T-SNE. To maintain clarity in these figures, we only show the 10 samples that are the closest to each learned prototype (i.e. cluster center). As it can be seen, in figure 20 (a), the data in the "BTR70" class has two main categories and could be represented by only two clusters. Since we are using 15 clusters, one category is represented by 6 small subcategories and the other one by 9 subcategories. In figure 20 (b),

the data in the "T72" class has two main categories: one compact category (Top right) represented by 3 clusters and one scattered category represented by the remaining 12 clusters. In figure 20 (c), the data in the "PICKUP" class has no apparent structure in the 2D projection space, and all 15 clusters were used to summarize different regions of the learned feature space.

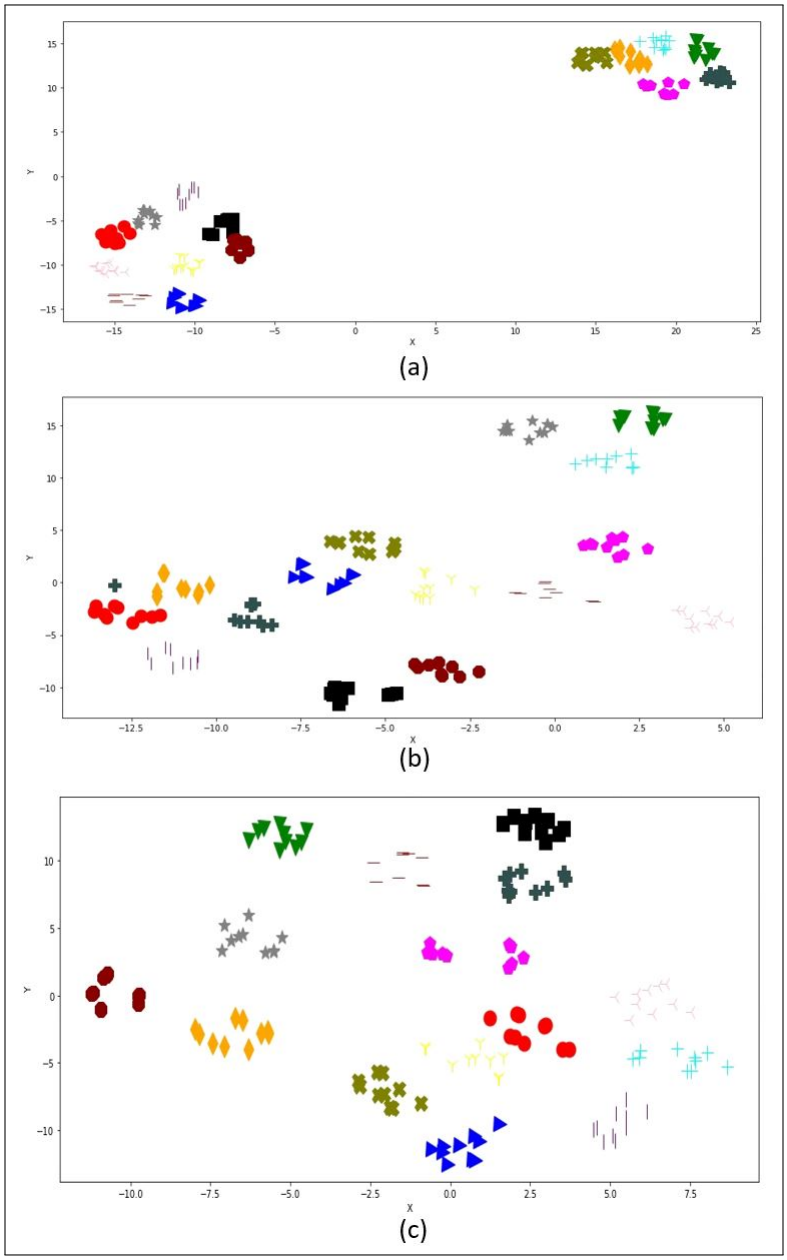

Figure 20: T-SNE projection of the 10 nearest samples to the centers of the 15 learned clusters for 3 different classes (a) "BTR70" class, (b) "T72" class and "PICKUP" class. Samples that belongs to the same cluster are displayed with the same color/symbol.

In figure 21 (a), to keep our analysis concise, we identify the 5 most distinct clusters (surrounded by a solid blue line). For each of these 5 cluster, we select its nearest image patches to its centers. Figure 21 (b) shows these selected image patches for the 5 clusters. As it can be seen, the 5 identified clusters can capture the within-class variation for the "PICKUP" target. For example, the blue triangle cluster includes samples that are captured during the day and imaged from a side view. Similarly, the gray star cluster includes samples that are captured during the night and imaged from a front view.

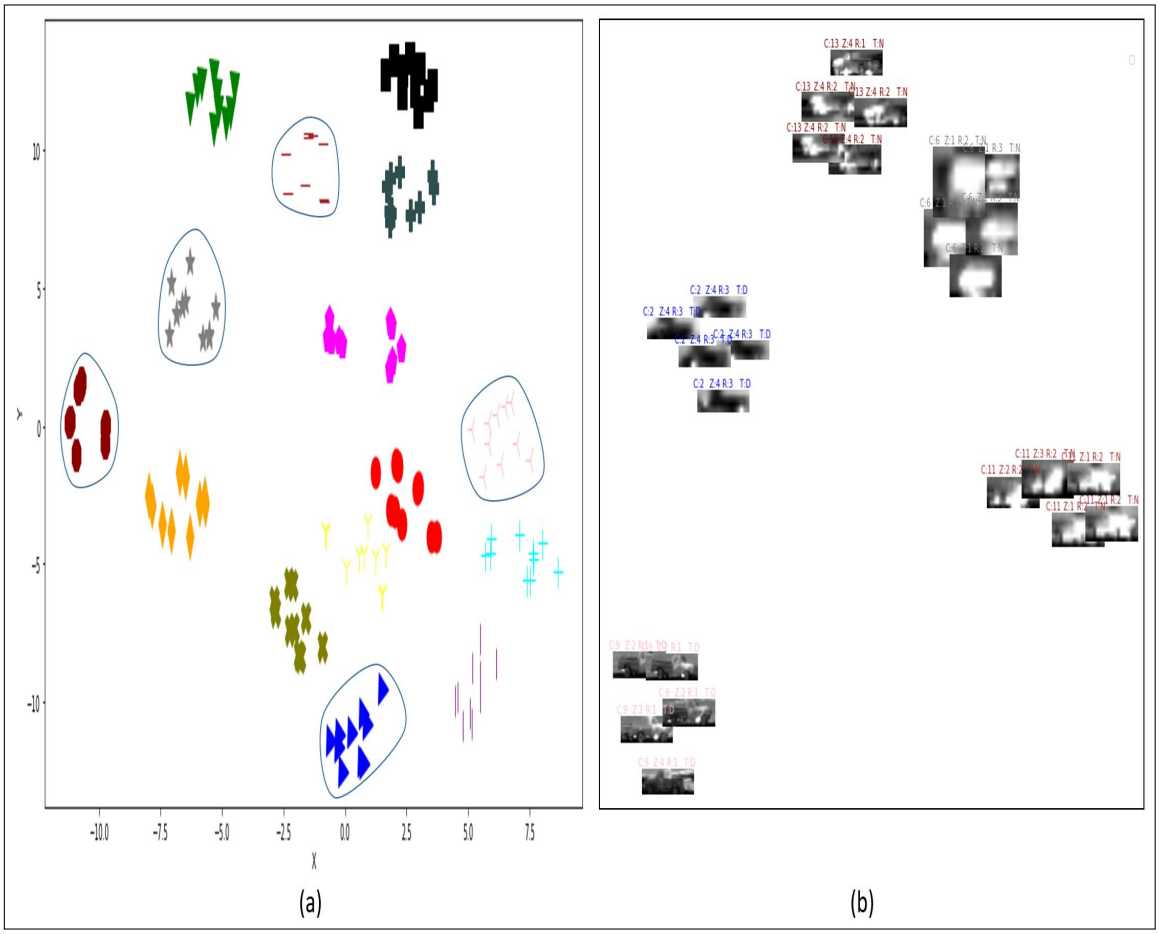

Figure 21: Illustration of the ability of the DML with the magnet loss to capture the intra-class variation for the "PICKUP" class. (a) the 5 most distinct clusters (surrounded by a solid blue line) among the 15 learned clusters. (b) the 5 most representative image patches. The image patch positions are based on the 2D T-SNE projection. Above each image patch we displayed the following information (C: cluster index, Z: Zone, R: range and T: is day or night (D/N)).

Next, we analyze the clusters of the "PICKUP" class further and show that they capture similarities with respect to some variations within the data. For each cluster, we analyze the similarity of all samples assigned to it, we group these into 3 histograms grouped according to (1)

day/night, (2) one of the 4 zones according to the view angle, and (3) range. In figure 22 (a), for each of the 5 clusters identified in figure 21 (a), we display the image patch of the nearest sample to the cluster center. In figure 22 (b)-(d) we display the 3 histograms of each cluster. For instance, the first cluster includes only samples collected during the day, that are captured mainly from side view 1 and at range 2500.

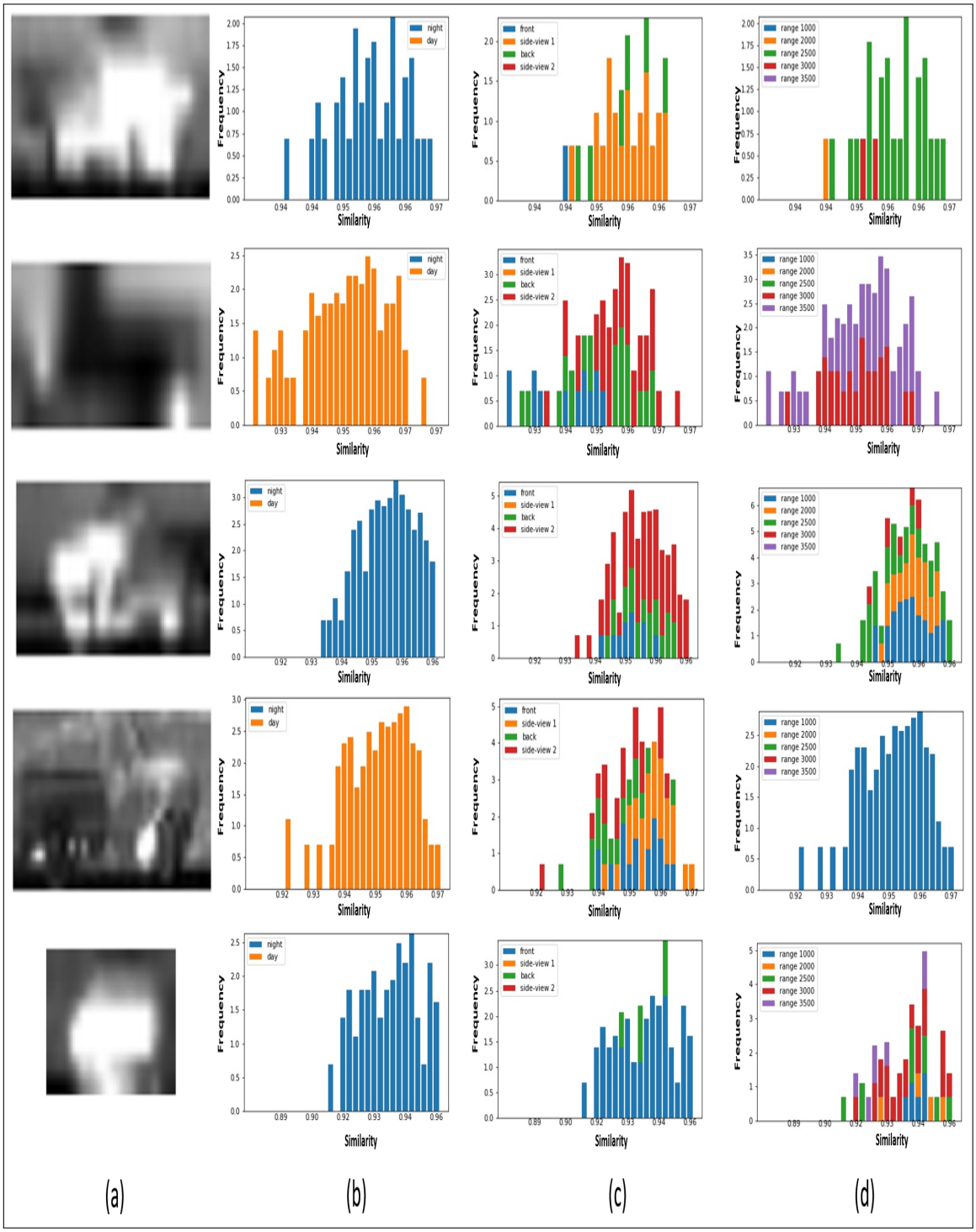

Figure 22: Analysis of the 5 identified distinct clusters for "PICKUP" class. (a) nearest image patch to the clusters center, (b) histogram accor ng to day/night, (c) histogram according to one of the 4 zones according to the view angle, and (d) histogram according to range.

Table 3 summarizes the content of the 15 clusters of the "PICKUP" target. As it can be seen, the majority of samples in each cluster are captured either during the day or night. In fact, 13 out of the 15 clusters have more than 90% of their samples that are imaged either during the day or night. Thus, we can claim that the day-night information is captured by the model. The zone information is partially captured in this class, where in several clusters the majority of samples have the same angle view. This means that most of the clusters are dominated by one zone.

		Time of the Day		Target Orientation				Target Range				
		night	day	front	side-view 1	back	side-view 2	1000	2000	2500	3000	3500
Cluster	No. samples	%	%	%	%	%	%	%	%	%	%	%
0	99	0.08	0.91	0.01	0.45	0.3	0.23	0.0	0.38	0.41	0.19	0.01
1	132	1	0	0.06	0.67	0.26	0.0	0.0	0.0	0.0	0.29	0.70
2	140	0.0	1	0.22	0.00	0.33	0.43	0.0	0.0	0.0	0.36	0.63
3	0.92	1	0	0.11	0.84	0.03	0.0	0.0	0.65	0.0	0.34	0.0
4	154	0.85	0.14	0.0	0.0	0.99	0.0	0.22	0.14	0.35	0.18	0.09
5	131	0.0	1	0.14	0.01	0.22	0.61	0.04	0.43	0.45	0.06	0.0
6	114	0.99	0.00	0.89	0.0	0.1	0.0	0.11	0.13	0.21	0.39	0.14
7	106	0.94	0.05	0.08	0.0	0.19	0.71	0.0	0.0	0.03	0.24	0.71
8	186	0.15	0.84	0.16	0.34	0.49	0.0	0.06	0.58	0.13	0.17	0.04
9	175	0.0	1	0.23	0.34	0.19	0.22	0.98	0.01	0.0	0.0	0.0
10	182	0.01	0.98	0.68	0.07	0.23	0.01	0.0	0.16	0.36	0.46	0.00
11	79	1	0.0	0.17	0.55	0.26	0.0	0.0	0.11	0.72	0.16	0.0
12	119	0.00	0.99	0.07	0.74	0.16	0.00	0.0	0.0	0.04	0.15	0.79
13	227	0.99	0.0	0.11	0.0	0.17	0.71	0.33	0.26	0.30	0.07	0.01
14	84	0.97	0.02	0.07	0.73	0.19	0.0	1	.0	0.0	0.0	0.0

Table 3: Statistics of the 15 learned clusters for the "PICKUP" class that capture its intra-class variations.

The same experiments were performed for the "T72" class. Figure 23 (a) shows the selected clusters and figure 23 (b) displays the selected image patches. We notice that there is a consistency in terms of captured information between the two targets. Both of them mainly capture the same information (day-night and angle view). This is supported by the results showed in

figure 24. However, there is a difference between the two targets in term of the clusters distribution versus the range. Table 4 summarizes the clustering results for the "T72" target. We notice that for the "PICKUP" target, 8 clusters have more than 60% of their samples that have the same range while only 3 clusters for the "T72" target. This means that the range information is partially captured for "PICKUP" target more than "T72".

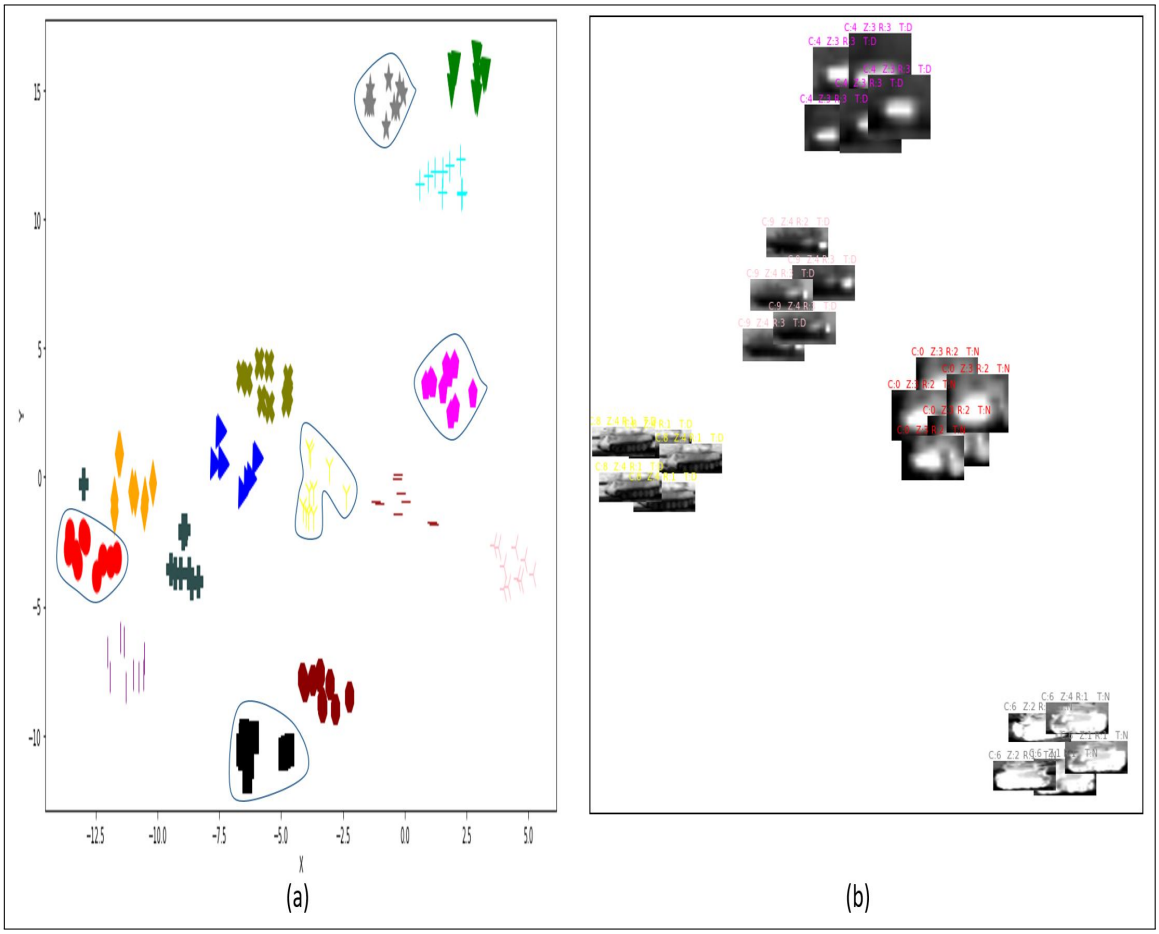

Figure 23: Illustration of the ability of the DML with the magnet loss to capture the intra-class variation for the "T72" class. (a) the 5 most distinct clusters (surrounded by a solid blue line) among the 15 learned clusters. (b) the 5 most representative image patches. The image patch positions are based on the 2D T-SNE projection. Above each image patch we displayed the following information (C: cluster index, Z: Zone, R: range and T: is day or night (D/N)).

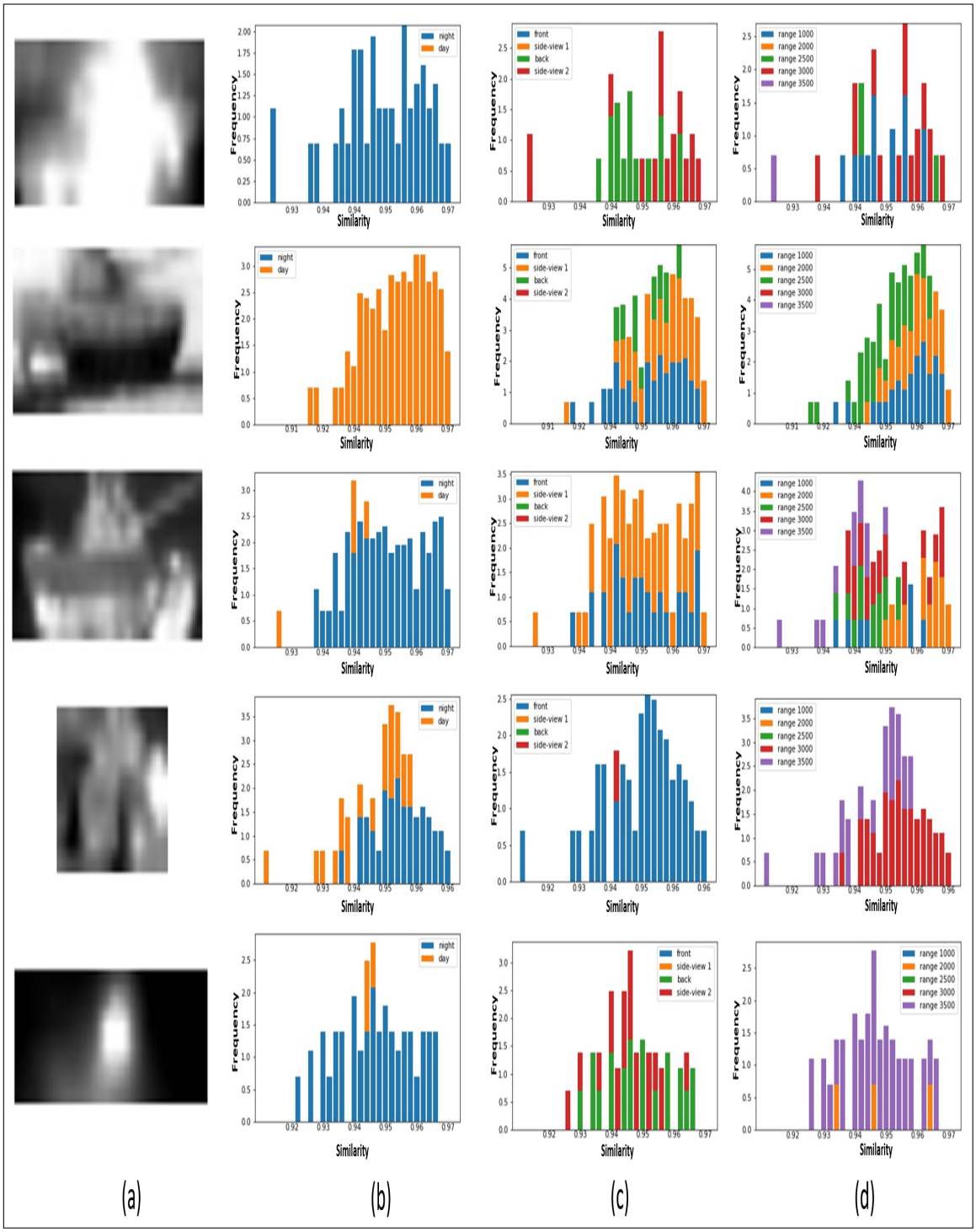

Figure 24: Analysis of the 5 identified distinct clusters for "T72" class. (a) nearest image patch to the clusters center, (b) histogram according to y/night, (c) histogram according to one of the 4 zones according to the view angle, and (d) histogram according to range.

		Time of the Day		Target Orientation				Target Range				
		night	day	front	side-view 1	back	side-view 2	1000	2000	2500	3000	3500
Cluster	No. samples	%	%	%	%	%	%	%	%	%	%	%
0	164	0.91	0.08	0.38	0.61	0.0	0.0	0.13	0.23	0.19	0.27	0.15
1	238	0.0	1	0.31	0.0	0.0	0.68	0.36	0.33	0.30	0.0	0.0
2	123	1	0	0.46	0.0	0.0	0.52	0.06	0.73	0.14	0.04	0.00
3	115	0.6	0.4	0.92	0.01	0.0	0.06	0.0	0.0	0.0	0.6	0.4
4	131	0.85	0.14	0.0	0.0	1	0.0	0.04	0.18	0.25	0.35	0.15
5	124	0.0	0.99	0.0	0.0	0.70	0.29	0.32	0.35	0.31	0.00	0.0
6	236	0	1	0.35	0.5	0.14	0.0	0.30	0.32	0.37	0.0	0.0
7	121	1	0	0.05	0.87	0.06	0.0	0.34	0.15	0.30	0.19	0.0
8	132	1	0	0.19	0.0	0.0	0.80	0.0	0.0	0.14	0.49	0.36
9	91	0.92	0.07	0.0	0.0	0.51	0.48	0.0	0.17	0.01	0.0	0.81
10	96	0.5	0.5	0.01	0.56	0.39	0.03	0.0	0.0	0.0	0.5	0.5
11	84	0.98	0.01	0.0	0.0	0.51	0.48	0.39	0.0	0.15	0.40	0.04
12	156	1	0	0.67	0.0	0.0	0.32	0.56	0.0	0.27	0.10	0.05
13	102	0.37	0.62	0.19	0.0	0.0	0.80	0.0	0.0	0.0	0.37	0.62
14	77	1	0	0.20	0.49	0.24	0.05	0.0	0.14	0.02	0.06	0.76

Table 4: Statistics of the 15 learned clusters for the "T72" class that capture its intra-class variations.

3.5 Adaptation of the VMF algorithm to ATR

For this method, we used the algorithm in [64] as is and kept all parameters to their default values.

3.5.1 Evaluation strategy

We evaluate the performance of this method using two different strategies. The first one is based on the KNN classifier as the previous methods. The second one is based on classifying the data based on its similarity to the learned mean direction vectors of all classes. We simply assign each test sample to the class of its nearest mean vector.

3.5.2 Parameters settings

The training parameters for VMF method are summarized in table 5. 300 epochs were sufficient to train the model. The mean directions are updated after each epoch. The concentration parameter κ characterizes the tightness of the distribution around the mean direction. Ideally, we want the learned VMF distribution for each class to be as tight as possible in the cost that the model is more likely to collapse. In our experiments, we found that $\kappa = 15$ gives the best results.

Video sampling:	every frame
Number of epochs:	300
Learning rate:	10^{-4}
Concentration:	$\kappa = 15$
Optimizer:	Adam

Table 5: VMF model training parameters

3.5.3 Experimental results

The training and testing classification accuracy using the mean directions are illustrated in figure 25. Notice that the model converges after few epochs. In fact, this model converges much faster than the previous two trained models. The KNN classification accuracy is very close to the accuracy using mean directions with difference less than 1%. The advantage of the mean vector classifier is that a test sample is classified by comparing the similarity to C mean vectors and not to all N training samples as the KNN.

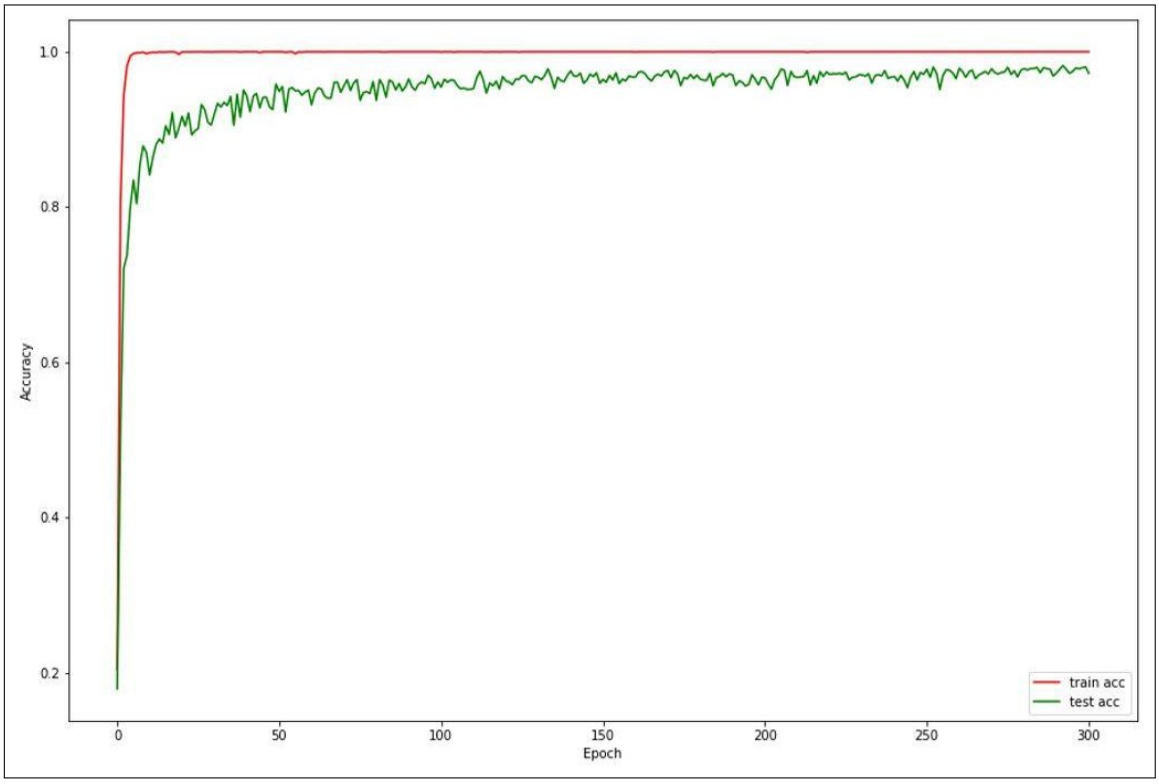

Figure 25: Classification accuracy using the mean direction vectors of the train and test data sets using VMF models.

The main objective of the VMF algorithm is to map features from each class to a tight cluster. These VMF clusters are represented by their mean vectors, which are used to discriminate between different classes. This can be validated by comparing the intra and inter class similarities for each class. Figure 26 displays these similarities where the cosine is used. In figure 26 (a), we show the results using the initial model, all training samples are randomly projected. As it can be seen, in figure 26 (b), for all classes the intra-class similarities are close to 1, indicating that the clusters are tight and most of the class samples are projected into a compact region in the embedding space. In addition, the plots show that there is a significant margin between every class and the other classes, indicating that there is no overlapping between the classes distributions.

Based on these results, we can claim that the model based on the VMF loss has achieved its goal on the training data by minimizing the intra-class distance and maximizing the inter-class distances.

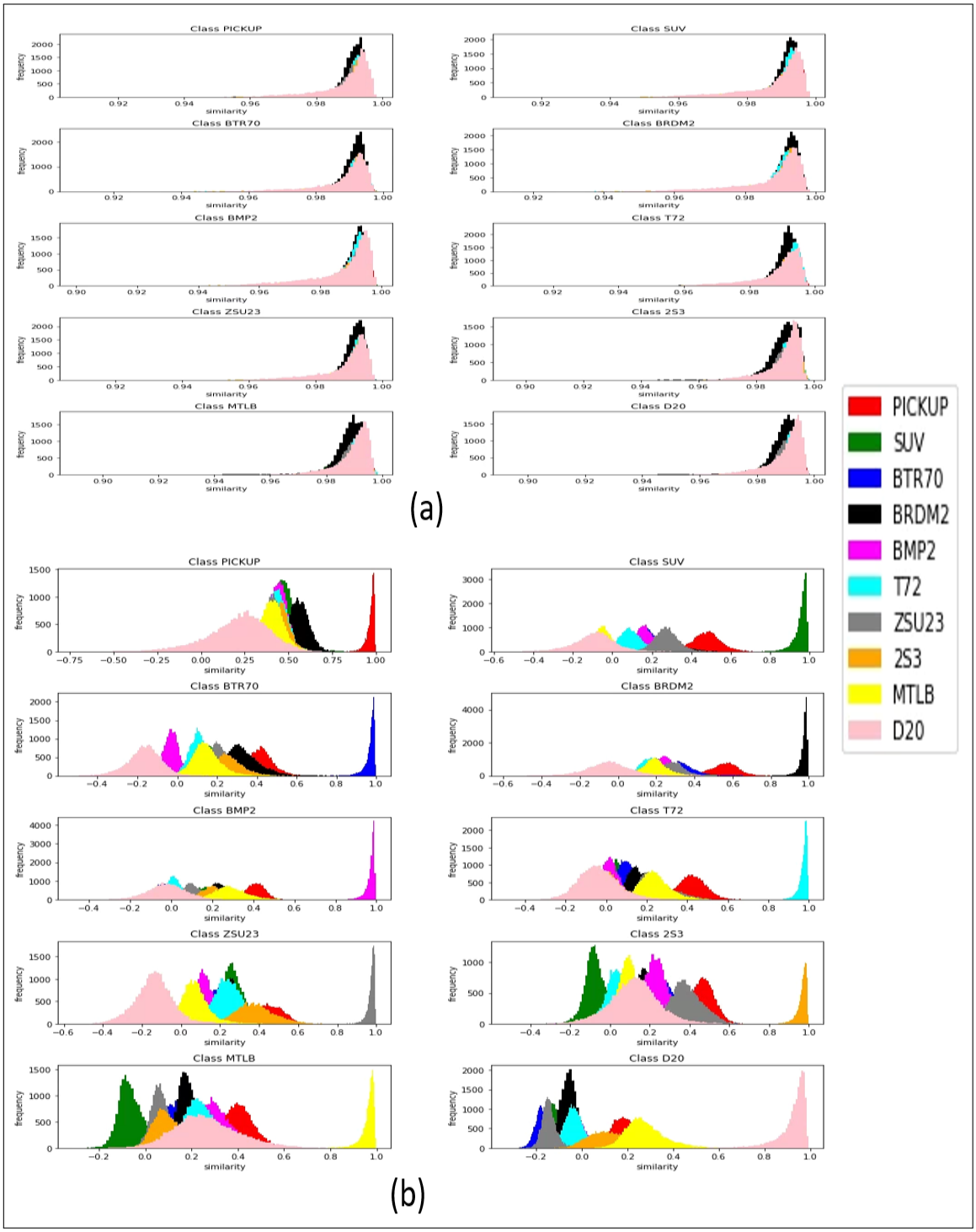

Figure 26: Histograms of inter and intra class similarities from features learned based on VMF distribution for the 10 classes using: (a) initial **model**, (b) learned model after 300 epochs.

3.5.4 Comparison between the 3 considered DML methods

Figure 27 displays the confusion matrices for the four methods. As it can be seen in the confusion matrix of TL model shown in figure 27 (d), "PICKUP" is confused with "SUV", which makes sense because they are two civilian cars that look similar, while "D20" is confused with "MTLB". The rest of the classes have good classification accuracy. These confusions vanished with VMF, magnet and TL^{+G} as they have good classification results for all the classes. These three methods make the same conclusion on all the classes. Notice that VMF have slightly better classification performance per class than the other methods.

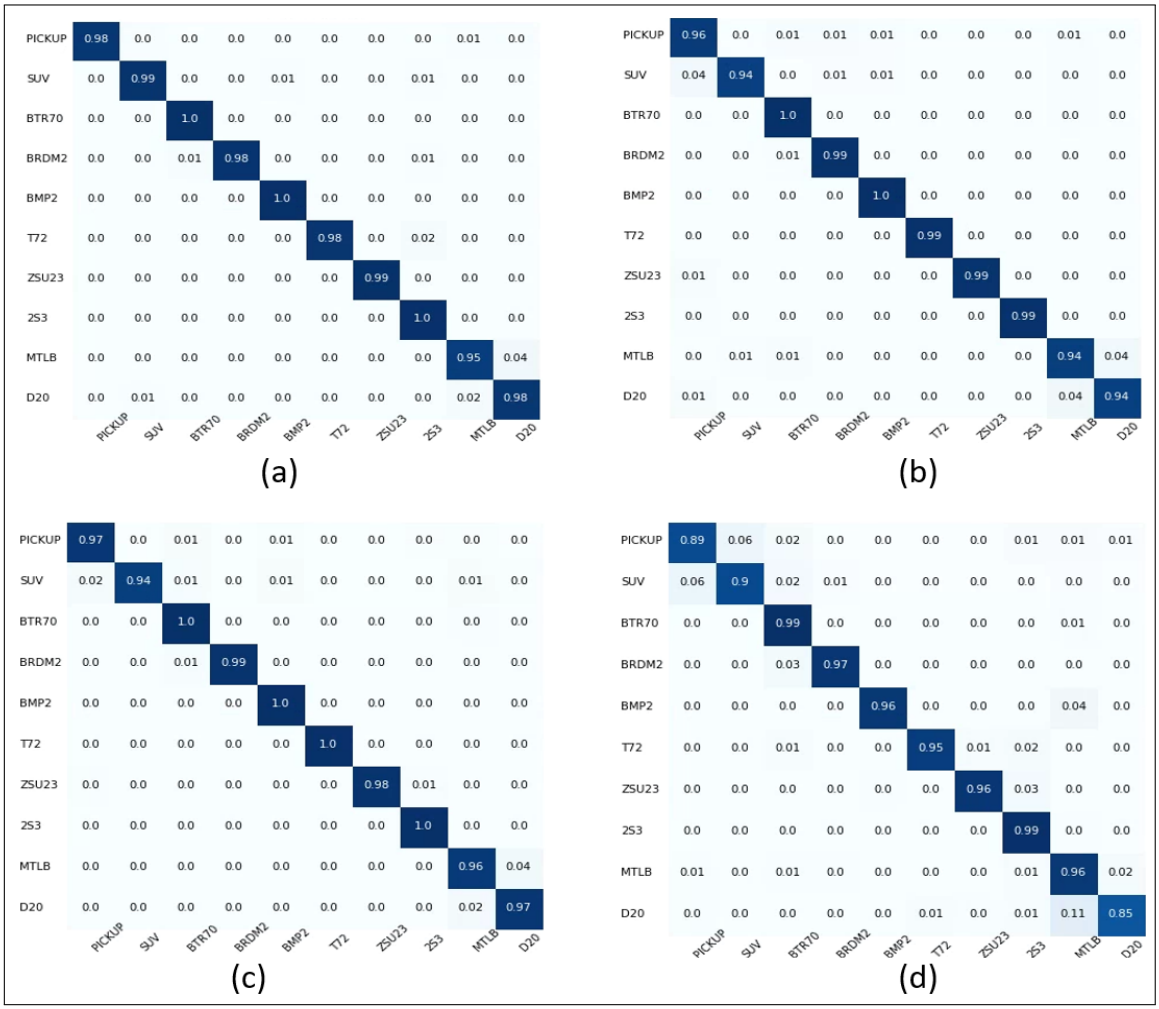

Figure 27: The Confusion matrices for the test data. (a) VMF confusion matrix, (b) magnet confusion matrix, (c) TL^{+G} confusion matrix, and (d) TL confusion matrix.

To compare the learning speed of the different methods in figure 28, we display the evolution of the classification performance during training. As it can be seen, the TL based methods are slower to converge compared to the other methods. The VMF method is the most efficient one and tends to converge after only few minuets.

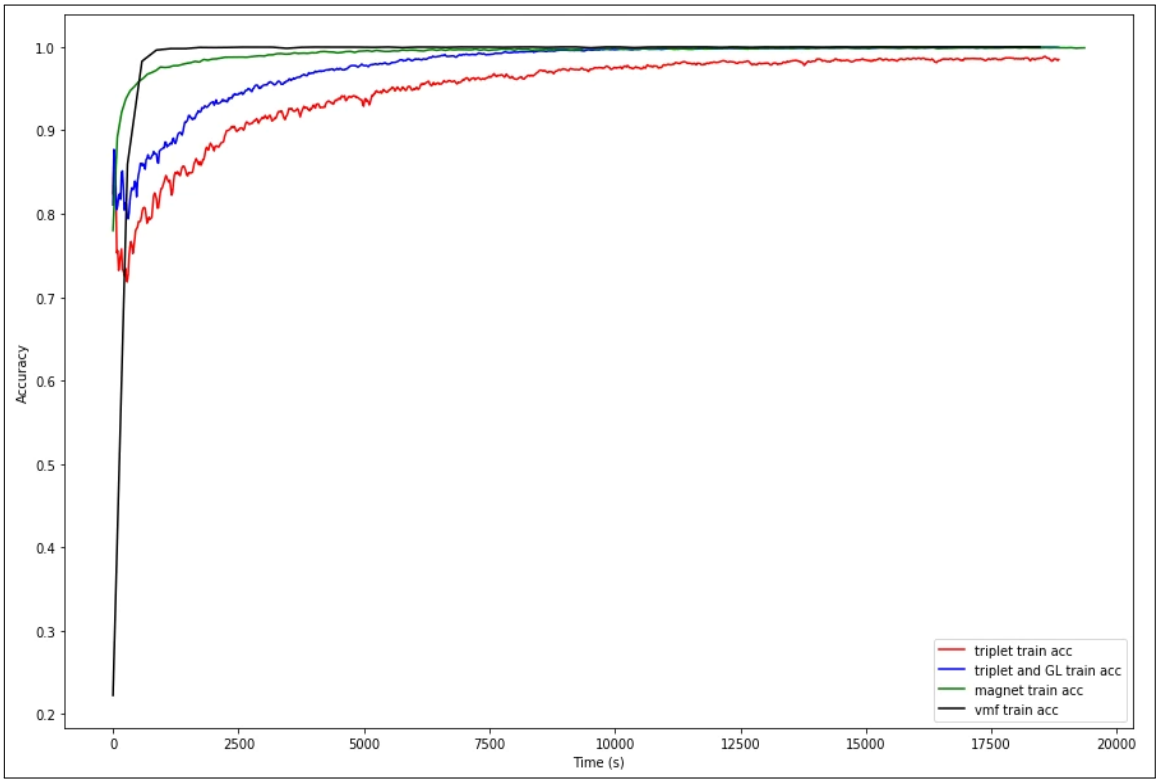

Figure 28: The evolution of the classification accuracy during training for the four methods versus time.

Another way to compare these methods is to compare the time needed to predict the test data. Table 6 displays the time needed by each method to predict all test data. As it can be seen, since the VMF makes predictions by comparing the test sample to only 10 means vectors, it is much more efficient than the other methods.

	Triplet	Triplet + Global loss term	Magnet	VMF
Used method for prediction	KNN	KNN	KNN	Mean directions method
Time to predict all test data set	49.47 s	49.47 s	49.49 s	0.10 s

Table 6: Comparison of the computational efficiency of the different DML methods.

CHAPTER 4

CONCLUSION

In this thesis, we investigated the application of DML to ATR based on infrared images. We reviewed the most common DML approaches, and we highlighted their advantages and disadvantages. The first one is based on the triplet loss. The second one is based on the magnet loss that aims to represent each class by multiple components to captures its intra-class variations. The third approach is based on a method that aims to learn a compact hyper-spherical embedding for each class based on Von Mises-Fisher distribution.

Using a large and diverse collection of infrared images, we showed that the three considered methods can learn models that achieve their objectives. We also validated the effectiveness of these methods by evaluating their classification performance as well as analyzing the compactness of their projection of the features. All methods showed good and comparable results in terms of classification performance, where they have very close classification accuracies. However, there is a significant difference in prediction time, where VMF is much faster than the two other methods.

REFERENCES

[1] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. "Face description with local binary patterns: Application to face recognition". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12 (2006), pp. 2037–2041.

[2] Aparna Akula et al. "Target recognition in infrared imagery using convolutional neural network". In: *Proceedings of International Conference on Computer Vision and Image Processing*. Springer. 2017, pp. 25–34.

[3] S Appalaraju and V Chaoji. "Image similarity using Deep CNN and Curriculum Learning. arXiv 2017". In: *arXiv preprint arXiv:1709.08761* ().

[4] Amr Bakry et al. "Digging deep into the layers of cnns: In search of how cnns achieve view invariance". In: *arXiv preprint arXiv:1508.01983* (2015).

[5] Sean Bell and Kavita Bala. "Learning visual similarity for product design with convolutional neural networks". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 98.

[6] David Casasent and Yu-Chiang Wang. "Automatic target recognition using new support vector machine". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 1. IEEE. 2005, pp. 84–89.

[7] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. "Learning a similarity metric discriminatively, with application to face verification". In: *CVPR (1)*. 2005, pp. 539–546.

[8] Lloyd G Clark and Vincent J Velten. "Image characterization for automatic target recognition algorithm evaluations". In: *Optical Engineering* 30.2 (1991), pp. 147–154.

[9] Antoine d'Acremont et al. "CNN-based target recognition and identification for infrared imaging in defense systems". In: *Sensors* 19.9 (2019), p. 2040.

[10] Guoxian Dai et al. "Deep correlated metric learning for sketch-based 3d shape retrieval". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[11] Jason V Davis et al. “Information-theoretic metric learning”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 209–216.

[12] Sandor Z Der and Rama Chellappa. “Probe-based automatic target recognition in infrared imagery”. In: *IEEE Transactions on Image Processing* 6.1 (1997), pp. 92–102.

[13] PN Druzhkov and VD Kustikova. “A survey of deep learning methods and software tools for image classification and object detection”. In: *Pattern Recognition and Image Analysis* 26.1 (2016), pp. 9–15.

[14] Yueqi Duan et al. “Deep localized metric learning”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.10 (2017), pp. 2644–2656.

[15] Ross S Eaton and Magnùs S Snorrason. “Searching for a Fast Alternative to KNN for Infrared ATR”. In: *Automatic Target Recognition XVII*. Vol. 6566. International Society for Optics and Photonics. 2007, 65660A.

[16] Antonio Fernández-Caballero, Marıa T López, and Juan Serrano-Cuerda. “Thermal-infrared pedestrian ROI extraction through thermal and motion information fusion”. In: *Sensors* 14.4 (2014), pp. 6666–6676.

[17] Rikke Gade and Thomas B Moeslund. “Thermal tracking of sports players”. In: *Sensors* 14.8 (2014), pp. 13679–13691.

[18] Paul D Gader, James M Keller, and Bruce N Nelson. “Recognition technology for the detection of buried land mines”. In: *IEEE transactions on fuzzy systems* 9.1 (2001), pp. 31–43.

[19] Erhan Gundogdu, Aykut Koc, and A Aydın Alatan. “Object classification in infrared images using deep representations”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 1066–1070.

[20] Erhan Gundogdu et al. “Deep distance metric learning for maritime vessel identification”. In: *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2017, pp. 1–4.

[21] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[22] Ben Harwood et al. “Smart mining for deep metric learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2821–2829.

[23] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[24] Elad Hoffer and Nir Ailon. “Deep metric learning using triplet network”. In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92.

[25] Elad Hoffer and Nir Ailon. “Semi-supervised deep learning by metric embedding”. In: *arXiv preprint arXiv:1611.01449* (2016).

[26] MD Zakir Hossain et al. “A comprehensive survey of deep learning for image captioning”. In: *ACM Computing Surveys (CSUR)* 51.6 (2019), pp. 1–36.

[27] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. “Discriminative deep metric learning for face verification in the wild”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1875–1882.

[28] Junshi Huang et al. “Cross-domain image retrieval with a dual attribute-aware ranking network”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1062–1070.

[29] Prateek Jain et al. “Metric and kernel learning using a linear transformation”. In: *Journal of machine Learning research* 13.Mar (2012), pp. 519–547.

[30] Luo Juan and Luo Gwon. “A comparison of sift, pca-sift and surf”. In: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 8.3 (2007), pp. 169–176.

[31] Dor Kedem et al. “Non-linear metric learning”. In: *Advances in neural information processing systems*. 2012, pp. 2573–2581.

[32] Mohammad Nazmul Alam Khan et al. “Automatic target recognition in infrared imagery using dense hog features and relevance grouping of vocabulary”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 293–298.

[33] Takumi Kobayashi. “BFO meets HOG: feature extraction based on histograms of oriented pdf gradients for image classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 747–754.

[34] Chulhee Lee and David A Landgrebe. "Decision boundary feature extraction for neural networks". In: *IEEE Transactions on Neural Networks* 8.1 (1997), pp. 75–83.

[35] Xin Li, Rui Guo, and Chao Chen. "Robust pedestrian tracking and recognition from FLIR video: A unified approach via sparse coding". In: *Sensors* 14.6 (2014), pp. 11245–11259.

[36] Geert Litjens et al. "A survey on deep learning in medical image analysis". In: *Medical image analysis* 42 (2017), pp. 60–88.

[37] Jingtuo Liu et al. "Targeting ultimate accuracy: Face recognition via deep embedding". In: *arXiv preprint arXiv:1506.07310* (2015).

[38] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[39] Jiwen Lu, Junlin Hu, and Jie Zhou. "Deep metric learning for visual understanding: An overview of recent advances". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 76–84.

[40] Jiwen Lu et al. "Multi-manifold deep metric learning for image set classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1137–1145.

[41] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[42] Mahdi Moalla et al. "Application of Convolutional and Recurrent Neural Networks for Buried Threat Detection Using Ground Penetrating Radar Data". In: *IEEE Transactions on Geoscience and Remote Sensing* (2020).

[43] Nasser M Nasrabadi. "Deeptarget: An automatic target recognition using deep convolutional neural networks". In: *IEEE Transactions on Aerospace and Electronic Systems* 55.6 (2019), pp. 2687–2697.

[44] Hyun Oh Song et al. "Deep metric learning via lifted structured feature embedding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4004–4012.

[45] Oren Rippel et al. "Metric learning with adaptive density discrimination". In: *arXiv preprint arXiv:1511.05939* (2015).

[46] Iain Rodger, Barry Connor, and Neil M Robertson. "Classifying objects in LWIR imagery via CNNs". In: *Electro-Optical and Infrared Systems: Technology and Applications XIII*. Vol. 9987. International Society for Optics and Photonics. 2016, 99870H.

[47] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[48] Patrice Y Simard, David Steinkraus, John C Platt, et al. "Best practices for convolutional neural networks applied to visual document analysis." In: *Icdar*. Vol. 3. 2003. 2003.

[49] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[50] Kihyuk Sohn. "Improved deep metric learning with multi-class n-pair loss objective". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1857–1865.

[51] Yi Sun et al. "Deep learning face representation by joint identification-verification". In: *Advances in neural information processing systems*. 2014, pp. 1988–1996.

[52] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[53] Bin Tian et al. "A study of cloud classification with neural networks using spectral and textural features". In: *IEEE transactions on neural networks* 10.1 (1999), pp. 138–151.

[54] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Advances in neural information processing systems*. 2016, pp. 3630–3638.

[55] Yue Wang et al. "Probabilistic principal component subspaces: a hierarchical finite mixture model for data visualization". In: *IEEE Transactions on Neural Networks* 11.3 (2000), pp. 625–636.

[56] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. "Distance metric learning for large margin nearest neighbor classification". In: *Advances in neural information processing systems*. 2006, pp. 1473–1480.

[57] Kilian Q Weinberger and Lawrence K Saul. "Distance metric learning for large margin nearest neighbor classification". In: *Journal of Machine Learning Research* 10.Feb (2009), pp. 207–244.

[58] Chao-Yuan Wu et al. “Sampling matters in deep embedding learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2840–2848.

[59] Pengcheng Wu et al. “Online multimodal deep similarity learning with application to image retrieval”. In: *Proceedings of the 21st ACM international conference on Multimedia*. 2013, pp. 153–162.

[60] Eric P Xing et al. “Distance metric learning with application to clustering with side-information”. In: *Advances in neural information processing systems*. 2003, pp. 521–528.

[61] Dong Yi et al. “Deep metric learning for person re-identification”. In: *2014 22nd International Conference on Pattern Recognition*. IEEE. 2014, pp. 34–39.

[62] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. “Hard-aware deeply cascaded embedding”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 814–823.

[63] Sergey Zagoruyko and Nikos Komodakis. “Wide residual networks”. In: *arXiv preprint arXiv:1605.07146* (2016).

[64] Xuefei Zhe, Shifeng Chen, and Hong Yan. “Directional statistics-based deep metric learning for image classification and retrieval”. In: *Pattern Recognition* 93 (2019), pp. 113–123.

Appendices

NOMENCLATURE

ATR	Automatic target recognition
CNN	Convolution Neural Networks
DDML	Deep distance metric learning
DML	Deep metric learning
FaceNet	Face Neural network
FGVC	Fine-grained Visual Categorization
KNN	K-Nearest Neighbor
ML	Machine Learning
OfPMS	Off-line Pair Mining Strategy
OnPMS	Online Pair Mining Strategy
SIFT	Scale-Invariant Feature Transform
UofL	University of Louisville
VMF	von Mises-Fishes

CURRICULUM VITAE

NAME: Abdelhamid Bouzid

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.S., Computer Science
August 2020
University of Louisville, *Louisville, Kentucky*

B.Eng., Polytechnic Engineering
June 2018
Tunisia Polytechnic School, *Tunis, Tunisia*

A.P.Chem., Associate Degree in Physics and Chemistry
June 2015
The Monastir Preparatory Engineering Institute, *Tunis, Monastir*

JOURNAL PUBLICATIONS:

- Moalla, M., Frigui, H., Karem, A. and Bouzid, A., 2020. Application of Convolutional and Recurrent Neural Networks for Buried Threat Detection Using Ground Penetrating Radar

Data. IEEE Transactions on Geoscience and Remote Sensing.

PROJECTS AND INTERNSHIPS:

1. Graduate Research Assistant in University of Louisville, January 2019 - Today
2. Graduation intern in University of Louisville , Mars 2018 - December 2018
3. Engineer intern in COSIM Laboratory , July 2017 - August 2017

HONORS AND AWARDS:

1. CECS Master of science Award, April 2020
2. Top 1% student in the National Exam for Admittance to Engineering Schools Tunisia: September 2015