# Unsupervised Metric Learning with Synthetic Examples

**Ujjal Kr Dutta,[†,1] Mehrtash Harandi,[*,2] C Chandra Sekhar[†,3]**
[†]Dept. of Computer Science and Eng., Indian Institute of Technology Madras, India
[*]Dept. of Electrical and Computer Systems Eng., Monash University, Australia
[1]ukd@cse.iitm.ac.in, [2]mehrtash.harandi@monash.edu, [3]chandra@cse.iitm.ac.in

## Abstract

Distance Metric Learning (DML) involves learning an embedding that brings similar examples closer while moving away dissimilar ones. Existing DML approaches make use of class labels to generate constraints for metric learning. In this paper, we address the less-studied problem of learning a metric in an unsupervised manner. We do not make use of class labels, but use unlabeled data to generate adversarial, synthetic constraints for learning a metric inducing embedding. Being a measure of uncertainty, we minimize the entropy of a conditional probability to learn the metric. Our stochastic formulation scales well to large datasets, and performs competitive to existing metric learning methods.

## Introduction

*"Are two given examples similar (or dissimilar)?"* This is a crucial question in problems like classification (Chen et al. 2018; Xie et al. 2018), clustering and retrieval (Duan et al. 2018; Wang et al. 2017), prevalent in machine learning and computer vision. **Distance Metric Learning** (DML) aims at learning a new embedding of the data such that similar examples are brought closer, while dissimilar examples are moved further apart. The importance of DML is even more crucial in challenging scenarios like *Zero-Shot Learning* (ZSL) (Xian et al. 2018; Oh Song et al. 2016), and *Fine-Grained Visual Categorization* (FGVC) (Qian et al. 2015). For instance, in the ZSL setting, where test examples belong to classes not seen during training, DML is a preferred choice over standard softmax-based classification models. This is because it aims at learning a *generic notion* of similarity among examples, as opposed to *class-specific concepts*.

To provide *weak-supervision* for learning a distance metric, DML approaches require *constraints*, that can be of various types: *pairs* (Chen et al. 2018; Harandi, Salzmann, and Hartley 2017; Xie et al. 2018; Koestinger et al. 2012; Davis et al. 2007), *triplets* (Shi, Bellet, and Sha 2014; Ye et al. 2017; Weinberger, Blitzer, and Saul 2006), *tuples* (Sohn 2016), or *batches* (Oh Song et al. 2016). Despite the success of state-of-the-art DML methods proposed in the recent years (Duan et al. 2018; Chen et al. 2018; Xie et al. 2018), they make use of class labels to generate such *constraints*. In this paper, we address the following question: *"How do we learn a metric, when there are no class labels present?"* This is an important problem, particularly in newer applications involving large datasets, where obtaining class labels may be infeasible (*e.g.*, invasive medical imaging techniques, 3D point cloud data, unsupervised domain adaptation etc.).

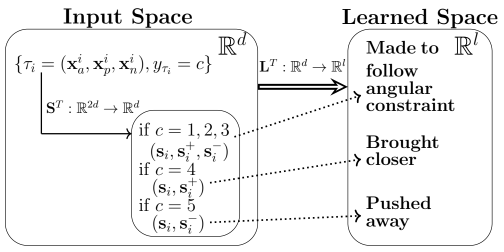
Input Space
Learned Space
$\mathbb{R}^d$
$\{\tau_i = (\mathbf{x}_a^i, \mathbf{x}_p^i, \mathbf{x}_n^i), y_{\tau_i} = c\}$
$\mathbf{S}^T : \mathbb{R}^{2d} \to \mathbb{R}^d$
if $c = 1, 2, 3$ $(\mathbf{s}_i, \mathbf{s}_i^+, \mathbf{s}_i^-)$
if $c = 4$ $(\mathbf{s}_i, \mathbf{s}_i^+)$
if $c = 5$ $(\mathbf{s}_i, \mathbf{s}_i^-)$
$\mathbf{L}^T : \mathbb{R}^d \to \mathbb{R}^l$
$\mathbb{R}^l$
Made to follow angular constraint
Brought closer
Pushed away

Figure 1: Illustration of our proposed approach SUML.

Classically, learning of embeddings in an unsupervised manner has been done in the context of *manifold learning* (Cox and Cox 2000; Tenenbaum, De Silva, and Langford 2000; He and Niyogi 2003; He et al. 2005), or *diffusion processes* (Donoser and Bischof 2013; Bai et al. 2017; Iscen et al. 2017) that capture the intrinsic manifold structure of the data. More recently, a few unsupervised feature learning approaches have been proposed (Dosovitskiy et al. 2016; Li et al. 2016; Caron et al. 2018; Wu et al. 2018). However, they ignore the basic *semantic relationships* among the examples and are not particularly designed for metric learning. (Iscen et al. 2018) proposed an approach to mine *hard* constraints for unsupervised metric learning, and is considered as a state-of-the-art.

In this paper, we propose a novel, unsupervised approach to learn an embedding that induces a metric (Bellet, Habrard, and Sebban 2015) in the learned space. We do not make use of class labels to learn the metric, but rather sample random triplets from the unlabeled data. As illustrated in Figure 1, for each random triplet $\tau_i$, we imagine an associated *hypothetical label* $y_{\tau_i}$ that represents all possible semantic permutations of examples $\boldsymbol{x}_a^i$, $\boldsymbol{x}_p^i$ and $\boldsymbol{x}_n^i$, present in $\tau_i$. Depending on a possible value of $y_{\tau_i}$, we generate synthetic triplets $(\boldsymbol{s}_i, \boldsymbol{s}_i^+, \boldsymbol{s}_i^-)$ or pairs $(\boldsymbol{s}_i, \boldsymbol{s}_i^+)$ (or $(\boldsymbol{s}_i, \boldsymbol{s}_i^-)$), which are finally used to learn an embedding, parameterized by $\boldsymbol{L} \in \mathbb{R}^{d \times l}$. This is achieved by minimizing the entropy of a conditional probability defined on $\tau_i$, considering all cases of *hypothetical labels*. The synthetic constraints are generated using a mapping parameterized by $\boldsymbol{S} \in \mathbb{R}^{2d \times d}$, such that they are *adversarial* to the metric learning process. Despite being unsupervised, our proposed approach performs competitive to existing DML approaches on various tasks.

## Proposed Approach

Given an unlabeled dataset $\mathcal{X} = \{\boldsymbol{r}_i\}_{i=1}^{|\mathcal{X}|}$, let $\boldsymbol{x}_i \in \mathbb{R}^d$ denote the descriptor of a raw example $\boldsymbol{r}_i$. Let $\boldsymbol{L} \in \mathbb{R}^{d \times l}$, $l \leq d$ be a matrix such that $\boldsymbol{L}^\top \boldsymbol{x}_i \in \mathbb{R}^l$, denotes the learned embedding of an example $\boldsymbol{x}_i \in \mathbb{R}^d$. As $\boldsymbol{L}\boldsymbol{L}^\top \succeq 0$, the matrix $\boldsymbol{L}$ induces a *pseudo* squared Mahalanobis distance metric: $\delta^2_{\boldsymbol{L}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{L}\boldsymbol{L}^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)$, for a pair of examples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$. To learn the parameter $\boldsymbol{L}$ of the metric, a popular form of *weak-supervision* is by providing a triplet constraint set (Shi, Bellet, and Sha 2014; Ye et al. 2017; Weinberger, Blitzer, and Saul 2006): $\mathcal{T}_{\text{labeled}} = \{(\boldsymbol{x}_i, \boldsymbol{x}_i^+, \boldsymbol{x}_i^-)\}_{i=1}^{|\mathcal{T}_{\text{labeled}}|}$, where $\boldsymbol{x}_i^+$ and $\boldsymbol{x}_i$ are from the same class, $\boldsymbol{x}_i^-$ and $\boldsymbol{x}_i$ are from different classes. Typically, $\boldsymbol{x}_i$ is called the *anchor*, $\boldsymbol{x}_i^+$ the *positive* and $\boldsymbol{x}_i^-$ the *negative*. The idea is to move anchor and positive closer, while moving away anchor and the negative (Figure 2).

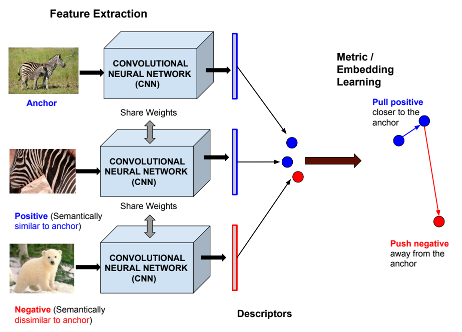
Feature Extraction
CONVOLUTIONAL NEURAL NETWORK (CNN)
Anchor
Share Weights
CONVOLUTIONAL NEURAL NETWORK (CNN)
Positive (Semantically similar to anchor)
Share Weights
CONVOLUTIONAL NEURAL NETWORK (CNN)
Negative (Semantically dissimilar to anchor)
Descriptors
Metric / Embedding Learning
Pull positive closer to the anchor
Push negative away from the anchor

Figure 2: Illustration of metric learning using *triplet constraints* (best viewed in color). The sample input images belong to the AwA2 dataset (Xian et al. 2018).

### Using synthetic examples for metric learning

In the absence of any class labels, we propose to randomly sample a triplet set: $\mathcal{T} = \{\tau_i = (\boldsymbol{x}_a^i, \boldsymbol{x}_p^i, \boldsymbol{x}_n^i)\}_{i=1}^{|\mathcal{T}|}$. Let $C(\boldsymbol{x}_a^i)$, $C(\boldsymbol{x}_p^i)$ and $C(\boldsymbol{x}_n^i)$ respectively denote the *actual semantic classes* of $\boldsymbol{x}_a^i$, $\boldsymbol{x}_p^i$ and $\boldsymbol{x}_n^i$, which are unknown to us. Hence, we cannot directly force to move $\boldsymbol{x}_a^i$ and $\boldsymbol{x}_p^i$ closer, while moving away $\boldsymbol{x}_a^i$ and $\boldsymbol{x}_n^i$, as done in supervised triplet based DML approaches. For this, we associate an imaginary *hypothetical label* $y_{\tau_i} = c$ to a triplet $\tau_i$. $c$ intuitively represents the possible semantic permutations of the examples in $\tau_i$, and can take values as follows:

**Case 1:** $(c = 1)$, if $C(\boldsymbol{x}_a^i) = C(\boldsymbol{x}_p^i) \neq C(\boldsymbol{x}_n^i)$.
**Case 2:** $(c = 2)$, if $C(\boldsymbol{x}_a^i) = C(\boldsymbol{x}_n^i) \neq C(\boldsymbol{x}_p^i)$.
**Case 3:** $(c = 3)$, if $C(\boldsymbol{x}_p^i) = C(\boldsymbol{x}_n^i) \neq C(\boldsymbol{x}_a^i)$.
**Case 4:** $(c = 4)$, if $C(\boldsymbol{x}_a^i) = C(\boldsymbol{x}_p^i) = C(\boldsymbol{x}_n^i)$.
**Case 5:** $(c = 5)$, if $C(\boldsymbol{x}_a^i) \neq C(\boldsymbol{x}_p^i) \neq C(\boldsymbol{x}_n^i)$.

Depending on the value of the *hypothetical label*, we generate *synthetic* constraints for metric learning. Specifically, for cases $c = 1, 2, 3$, we generate a synthetic triplet $(\boldsymbol{s}_i, \boldsymbol{s}_i^+, \boldsymbol{s}_i^-)$; for case $c = 4$, a *similar* synthetic pair $(\boldsymbol{s}_i, \boldsymbol{s}_i^+)$, and for case $c = 5$, a *dissimilar* synthetic pair $(\boldsymbol{s}_i, \boldsymbol{s}_i^-)$. In all cases, $\boldsymbol{s}_i, \boldsymbol{s}_i^+, \boldsymbol{s}_i^- \in \mathbb{R}^d$. By concatenating examples within a triplet as a single vector, and using an appropriate mapping, we can generate new representations of examples belonging to the same data distribution, for example, (Duan et al. 2018) generated *negatives* by a concatenation. Hence, we consider various forms of concatenation among the examples within the triplet to generate synthetic examples. The ways of concatenation depend on the *hypothetical label* $y_{\tau_i} = c$. We define our synthetic examples as follows: $\boldsymbol{s}_i = \boldsymbol{S}^\top \boldsymbol{c}_i$, $\boldsymbol{s}_i^+ = \boldsymbol{S}^\top \boldsymbol{c}_i^+$ and $\boldsymbol{s}_i^- = \boldsymbol{S}^\top \boldsymbol{c}_i^-$, where $\boldsymbol{S} \in \mathbb{R}^{2d \times d}$ is the parameter of the mapping used to generate synthetic examples. $\boldsymbol{s}_i$, $\boldsymbol{s}_i^+$ and $\boldsymbol{s}_i^-$ are called as the *synthetic anchor*, *synthetic positive* and *synthetic negative* respectively. $\boldsymbol{c}_i$, $\boldsymbol{c}_i^+$ and $\boldsymbol{c}_i^-$ are concatenations obtained using the examples within the triplet $\tau_i$, and can be defined as follows:

$$\textbf{Case 1: } (c=1),\ \boldsymbol{c}_i = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_a^i \end{bmatrix},\ \boldsymbol{c}_i^+ = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_p^i \end{bmatrix},\ \boldsymbol{c}_i^- = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_n^i \end{bmatrix}.$$

$$\textbf{Case 2: } (c=2),\ \boldsymbol{c}_i = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_a^i \end{bmatrix},\ \boldsymbol{c}_i^+ = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_n^i \end{bmatrix},\ \boldsymbol{c}_i^- = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_p^i \end{bmatrix}.$$

$$\textbf{Case 3: } (c=3),\ \boldsymbol{c}_i = \begin{bmatrix} \boldsymbol{x}_p^i \\ \boldsymbol{x}_p^i \end{bmatrix},\ \boldsymbol{c}_i^+ = \begin{bmatrix} \boldsymbol{x}_p^i \\ \boldsymbol{x}_n^i \end{bmatrix},\ \boldsymbol{c}_i^- = \begin{bmatrix} \boldsymbol{x}_p^i \\ \boldsymbol{x}_a^i \end{bmatrix}.$$

$$\textbf{Case 4: } (c=4),\ \boldsymbol{c}_i = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_p^i \end{bmatrix},\ \boldsymbol{c}_i^+ = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_n^i \end{bmatrix}.$$

$$\textbf{Case 5: } (c=5),\ \boldsymbol{c}_i = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_p^i \end{bmatrix},\ \boldsymbol{c}_i^- = \begin{bmatrix} \boldsymbol{x}_a^i \\ \boldsymbol{x}_n^i \end{bmatrix}.$$

### Idea of our method

We assume that there are two functions in our approach: a *synthetic loss* $s(\boldsymbol{S})$ that learns the parameter $\boldsymbol{S} \in \mathbb{R}^{2d \times d}$, and a *metric learning loss* $m(\boldsymbol{S}, \boldsymbol{L})$ that learns the parameter $\boldsymbol{L}$ using the synthetic examples generated by $\boldsymbol{S}$. Let us say we have two examples from *different* classes. Their distance should be usually *larger*, say, by a margin. If it is not the case (*i.e.*, *hard negatives*), and they are relatively closer, then it would be more beneficial to push them apart by $m(\boldsymbol{S}, \boldsymbol{L})$. On the other hand, $m(\boldsymbol{S}, \boldsymbol{L})$ would not benefit much if it tries to push away examples from different classes that are already well-separated. In fact, the *hardness* of constraints in DML is very crucial for the training convergence, and is well studied (Sohn 2016; Oh Song et al. 2016). Hence, we design the losses $s(\boldsymbol{S})$ and $m(\boldsymbol{S}, \boldsymbol{L})$ such that $s(\boldsymbol{S})$ synthesizes *tougher examples* that may violate what $m(\boldsymbol{S}, \boldsymbol{L})$ is trying to achieve. Essentially, by making them compete each other, we expect them to get better, and eventually learn a better metric.

## Motivation of our method

Our work is inspired by the spirit of Virtual Adversarial Training (VAT) (Miyato et al. 2018), a state-of-the-art approach for semi-supervised learning, where a tiny amount of perturbation is added to an unlabeled example. This perturbation is learned in such a way that the perturbed unlabeled example affects the objective function in the worst possible manner. Such a perturbed example is interpreted as an *adversarial example*. In our work, we do not learn a perturbation for adding to an example, but directly learn a mapping $\boldsymbol{S}^\top$ that generates the perturbed *synthetic* examples, which are *adversarial* to the metric learning process achieved by $m(\boldsymbol{S}, \boldsymbol{L})$.

## Justification of the concatenations

We provide a brief explanation on our choice of the above concatenations. For case $(c = 3)$, as we have $C(\boldsymbol{x}_p^i) = C(\boldsymbol{x}_n^i) \neq C(\boldsymbol{x}_a^i)$, the example $\boldsymbol{x}_p^i$ can be treated as the *anchor*, $\boldsymbol{x}_n^i$ can be treated as the *positive*, and $\boldsymbol{x}_a^i$ can be treated as the *negative*. Ideally, we expect that the descriptors of $\boldsymbol{x}_p^i$ and $\boldsymbol{x}_n^i$ should be *similar*, as they are from the same class. Also, descriptors of $\boldsymbol{x}_p^i$ and $\boldsymbol{x}_a^i$ should be *dissimilar*, as they are from different classes. Following the VAT principle, and under some perturbations, the combined representation of $\boldsymbol{x}_p^i$ and $\boldsymbol{x}_n^i$ could be mapped to a "better" *positive* representation of $\boldsymbol{x}_n^i$ with respect to the anchor $\boldsymbol{x}_p^i$.

We do not learn an explicit perturbation specific to an example, but hypothesize the existence of a global mapping $\boldsymbol{S}^\top$ that serves this purpose. Hence, we map the concatenation $\boldsymbol{c}_i^+$ to a "better" *synthetic positive* $\boldsymbol{s}_i^+$. Similarly, we hope to learn a "better" *synthetic negative* $\boldsymbol{s}_i^-$. In terms of the VAT principle, $\boldsymbol{s}_i^-$ can be interpreted as a "perturbed" version of $\boldsymbol{x}_a^i$, which poses as a more difficult negative with respect to the "perturbed" version of the anchor. For the *synthetic anchor*, we simply concatenate the current anchor with itself. Similar explanations hold for the cases $(c = 1, 2)$. For the case $(c = 4)$, where all three examples in $\tau_i$ are from the same class, we arbitrarily form a *synthetic anchor* and a *synthetic positive*, which are brought closer by the learned embedding. For the case $(c = 5)$, where all three examples in $\tau_i$ are from different classes, we arbitrarily form a *synthetic anchor* and a *synthetic negative*, which are moved away by the learned embedding.

## Synthetic loss and metric learning loss

**Cases 1, 2 and 3:** For $(c = 1, 2, 3)$, having obtained a synthetic triplet $(\boldsymbol{s}_i, \boldsymbol{s}_i^+, \boldsymbol{s}_i^-)$, we use the following *metric learning loss*: $m(\boldsymbol{S}, \boldsymbol{L}) = \log(1 + \exp(f_i^m))$, where $f_i^m = \delta^2(\boldsymbol{L}^\top \boldsymbol{s}_i, \boldsymbol{L}^\top \boldsymbol{s}_i^+) - 4\tan^2\alpha\, \delta^2(\boldsymbol{L}^\top \boldsymbol{s}_i^-, \boldsymbol{L}^\top \boldsymbol{s}_i^m)$, such that $\boldsymbol{s}_i^m = (\boldsymbol{s}_i + \boldsymbol{s}_i^+)/2$. Here, $\delta^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)$. $f_i^m$ intuitively optimizes an *angular constraint* (Wang et al. 2017) on the synthetic triplet, in order to learn the parameter $\boldsymbol{L}$ of the embedding. $\alpha > 0$ can be intuitively seen as an angle with respect to a triplet (Wang et al. 2017). The function $m(\boldsymbol{S}, \boldsymbol{L})$ essentially minimizes a smoothed version of the hinge loss $[f_i^m]_+ = \max(0, f_i^m)$. For generating "better" synthetic examples, we propose a *synthetic loss* $s(\boldsymbol{S})$ which aims at generating synthetic examples that are adversarial to $m(\boldsymbol{S}, \boldsymbol{L})$. $s(\boldsymbol{S})$ can be defined as follows: $s(\boldsymbol{S}) = \log(1 + \exp(f_i^s))$, where $f_i^s = 4\tan^2\alpha\, \delta^2(\boldsymbol{s}_i^-, \boldsymbol{s}_i^m) - \delta^2(\boldsymbol{s}_i - \boldsymbol{s}_i^+)$.

**Case 4:** For $(c = 4)$, we only consider a pair of synthetic examples, namely, a *synthetic anchor* $\boldsymbol{s}_i$ and a *synthetic positive* $\boldsymbol{s}_i^+$. The metric loss aims at minimizing the distance of the *similar* synthetic pair, and can be defined as: $m(\boldsymbol{S}, \boldsymbol{L}) = \delta^2(\boldsymbol{L}^\top \boldsymbol{s}_i, \boldsymbol{L}^\top \boldsymbol{s}_i^+)$. The adversarial synthetic loss can be defined as: $s(\boldsymbol{S}) = -\delta^2(\boldsymbol{s}_i, \boldsymbol{s}_i^+)$.

**Case 5:** For $(c = 5)$, we consider a pair consisting of a *synthetic anchor* $\boldsymbol{s}_i$ and a *synthetic negative* $\boldsymbol{s}_i^-$. The metric loss aims at maximizing the distance of the *dissimilar* synthetic pair, and can be defined as: $m(\boldsymbol{S}, \boldsymbol{L}) = -\delta^2(\boldsymbol{L}^\top \boldsymbol{s}_i, \boldsymbol{L}^\top \boldsymbol{s}_i^-)$. The adversarial synthetic loss can be defined as: $s(\boldsymbol{S}) = \delta^2(\boldsymbol{s}_i, \boldsymbol{s}_i^-)$.

## Optimization problem of the proposed approach

Having formulated the different possibilities of synthetic constraint generation, based on the semantic permutation of examples present in a triplet, we define the following loss: $l_c^i(\boldsymbol{S}, \boldsymbol{L}) = s_c^i(\boldsymbol{S}) + m_c^i(\boldsymbol{S}, \boldsymbol{L})$. Note that we add the subscript $c$ and superscript $i$ to account for the different combinations of values *hypothetical label* $y_{\tau_i}$ can take, for each triplet $\tau_i$. Given a randomly sampled triplet $\tau_i$, we now encode the loss $l_c^i(\boldsymbol{S}, \boldsymbol{L})$ as a probability, as follows: $p_c^i = p(y_{\tau_i} = c | \tau_i; \boldsymbol{S}, \boldsymbol{L}) = \frac{1}{1+\exp(l_c^i(\boldsymbol{S}, \boldsymbol{L}))}$; $\sum_c p_c^i = 1$. Intuitively, it represents the conditional probability parameterized by the distance metric. This probability can be used to compute the entropy, a measure of *uncertainty*. To learn our metric, we propose to minimize the randomness or *uncertainty* among all possible cases, leading to the following optimization problem:

$$\min_{\boldsymbol{S} \in \mathbb{R}^{2d \times d}, \boldsymbol{L} \in \mathbb{R}^{d \times l}} -\sum_i^{|\mathcal{T}|} \sum_c p_c^i \log p_c^i + \lambda \left\| \boldsymbol{L}^\top \boldsymbol{L} - \boldsymbol{I}_l \right\|_F^2 . \quad (1)$$

The term $\left\| \boldsymbol{L}^\top \boldsymbol{L} - \boldsymbol{I}_l \right\|_F^2$ in (1) is a regularizer added to enforce the orthogonality constraint $\boldsymbol{L}^\top \boldsymbol{L} = \boldsymbol{I}_l$, which avoids a model collapse and overfitting to training data. $\lambda > 0$ is a trade-off parameter. We call our proposed method as **Synthetic Unsupervised pseudo Metric Learning (SUML)**. It should be noted that synthetic examples are used only for the training process. After learning $\boldsymbol{L}$, one could directly compute the distance between two test examples $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^t$ as: $\delta_{\boldsymbol{L}}^2(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) = (\boldsymbol{x}_i^t - \boldsymbol{x}_j^t)^\top \boldsymbol{L}\boldsymbol{L}^\top (\boldsymbol{x}_i^t - \boldsymbol{x}_j^t)$. The gradient of the term $p_c^i \log p_c^i$ can be expressed as:

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{\Theta}} [p_c^i \log p_c^i] \\ &= \frac{-p_c^i (1 + \log p_c^i)}{1 + \exp(-l_c^i(\boldsymbol{S}, \boldsymbol{L}))} \left\{ \frac{\partial}{\partial \boldsymbol{\Theta}} s_c^i(\boldsymbol{S}) + \frac{\partial}{\partial \boldsymbol{\Theta}} m_c^i(\boldsymbol{S}, \boldsymbol{L}) \right\} . \end{aligned} \quad (2)$$

Here, $\boldsymbol{\Theta}$ could be any of $\boldsymbol{S}$ or $\boldsymbol{L}$. Let, $\boldsymbol{\delta}_{an} = \boldsymbol{c}_i - \boldsymbol{c}_i^-$, $\boldsymbol{\delta}_{ap} = \boldsymbol{c}_i - \boldsymbol{c}_i^+$, $\boldsymbol{\delta}_{nm} = \boldsymbol{c}_i^- - \boldsymbol{c}_i^m$. Here, $\boldsymbol{c}_i^m = (\boldsymbol{c}_i + \boldsymbol{c}_i^+)/2$.

We now define the gradients of the synthetic and metric losses corresponding to each case (we omit subscripts and superscripts for brevity):

**Cases 1, 2 and 3:**

$$\frac{\partial}{\partial \boldsymbol{S}}[s(\boldsymbol{S})] = \frac{2}{1+\exp(-f_i^s)}[4\tan^2\alpha\, \boldsymbol{\delta}_{nm}\boldsymbol{\delta}_{nm}^\top - \boldsymbol{\delta}_{ap}\boldsymbol{\delta}_{ap}^\top]\boldsymbol{S}$$

$$\frac{\partial}{\partial \boldsymbol{S}}[m(\boldsymbol{S},\boldsymbol{L})] = \frac{2}{1+\exp(-f_i^m)}[\boldsymbol{\delta}_{ap}\boldsymbol{\delta}_{ap}^\top - 4\tan^2\alpha\, \boldsymbol{\delta}_{nm}\boldsymbol{\delta}_{nm}^\top]\boldsymbol{S}\boldsymbol{L}\boldsymbol{L}^\top$$

$$\frac{\partial}{\partial \boldsymbol{L}}[m(\boldsymbol{S},\boldsymbol{L})] = \frac{2}{1+\exp(-f_i^m)}\boldsymbol{S}^\top[\boldsymbol{\delta}_{ap}\boldsymbol{\delta}_{ap}^\top - 4\tan^2\alpha\, \boldsymbol{\delta}_{nm}\boldsymbol{\delta}_{nm}^\top]\boldsymbol{S}\boldsymbol{L}.$$

**Case 4:**

$$\frac{\partial}{\partial \boldsymbol{S}}[s(\boldsymbol{S})] = -2\boldsymbol{\delta}_{ap}\boldsymbol{\delta}_{ap}^\top\boldsymbol{S}$$

$$\frac{\partial}{\partial \boldsymbol{S}}[m(\boldsymbol{S},\boldsymbol{L})] = 2\boldsymbol{\delta}_{ap}\boldsymbol{\delta}_{ap}^\top\boldsymbol{S}\boldsymbol{L}\boldsymbol{L}^\top$$

$$\frac{\partial}{\partial \boldsymbol{L}}[m(\boldsymbol{S},\boldsymbol{L})] = 2\boldsymbol{S}^\top\boldsymbol{\delta}_{ap}\boldsymbol{\delta}_{ap}^\top\boldsymbol{S}\boldsymbol{L}.$$

**Case 5:**

$$\frac{\partial}{\partial \boldsymbol{S}}[s(\boldsymbol{S})] = 2\boldsymbol{\delta}_{an}\boldsymbol{\delta}_{an}^\top\boldsymbol{S}$$

$$\frac{\partial}{\partial \boldsymbol{S}}[m(\boldsymbol{S},\boldsymbol{L})] = -2\boldsymbol{\delta}_{an}\boldsymbol{\delta}_{an}^\top\boldsymbol{S}\boldsymbol{L}\boldsymbol{L}^\top$$

$$\frac{\partial}{\partial \boldsymbol{L}}[m(\boldsymbol{S},\boldsymbol{L})] = -2\boldsymbol{S}^\top\boldsymbol{\delta}_{an}\boldsymbol{\delta}_{an}^\top\boldsymbol{S}\boldsymbol{L}.$$

**Stochastic Extension**: Let, $\mathbb{R}^d \ni \boldsymbol{x}_i = \boldsymbol{P}^\top z(\boldsymbol{r}_i;\theta_z)$ with $\boldsymbol{r}_i$ being the raw example, such that $\Phi = (\theta_z, \boldsymbol{P})$ denote the parameters of a network, where $z : \mathcal{X} \to \mathbb{R}^D$ with parameters $\theta_z$ provides a non-linear embedding of $\boldsymbol{r}_i$, and $\boldsymbol{P} \in \mathbb{R}^{D\times d}$ is the parametric matrix of a Fully-Connected (FC) layer at the end. Being fully differentiable by virtue of eq (2), our method can have a deep extension, wherein we can Back-Propagate (BP) the gradients by Stochastic Gradient Descent (SGD) and jointy learn $(\Phi, \boldsymbol{S}, \boldsymbol{L})$. To handle large-scale datasets, we propose a stochastic version of SUML, in Algorithm 1.

**Algorithm 1** stochastic SUML (sSUML)

```
Input: Unlabeled data $\mathcal{X}$, initial $\Phi$; $\lambda$ ,$\alpha$, $maxiter > 0$.
1: Initialize $\boldsymbol{S}_{\text{prev}}$, $\boldsymbol{L}_{\text{prev}}$ randomly.
2: while not converged do
3:     Randomly mine $\mathcal{T} = \{(\boldsymbol{x}_a^i, \boldsymbol{x}_p^i, \boldsymbol{x}_n^i)\}_{i=1}^{|\mathcal{T}|}$, using a sampled
       mini-batch.
4:     for $iter = 1$ to $maxiter$ do
5:         $[\boldsymbol{S}, \boldsymbol{L}] \leftarrow SUML(\boldsymbol{S}_{\text{prev}}, \boldsymbol{L}_{\text{prev}})$.
6:     end for
7:     $\boldsymbol{S}_{\text{prev}} = \boldsymbol{S}$; $\boldsymbol{L}_{\text{prev}} = \boldsymbol{L}$; Optionally BP to learn $\Phi$.
8: end while
9: return $\boldsymbol{L}_{\text{prev}}$
```

Table 1: Comparison of proposed method SUML against SOTA supervised DML approaches for the classification task in terms of classification accuracy (in %, higher the better).

| **Dataset** | | | **AT&T Faces** | | |
|---|---|---|---|---|---|
| **kNN** | **JDRML** | **LRGMML** | **AML** | **MDMLCLDD** | **SUML (Ours)** |
| k=1 | $90.42 \pm 3.93$ | $\mathbf{93.33 \pm 3.66}$ | $92.08 \pm 3.15$ | $91.17 \pm 2.55$ | $90.08 \pm 1.82$ |
| k=3 | $81.42 \pm 3.71$ | $\mathbf{88.75 \pm 5.02}$ | $84.00 \pm 3.09$ | $83.58 \pm 3.47$ | $81.17 \pm 2.01$ |
| k=10 | $64.83 \pm 3.84$ | $\mathbf{77.50 \pm 5.91}$ | $61.92 \pm 4.34$ | $64.33 \pm 5.18$ | $64.92 \pm 4.22$ |
| k=20 | $48.17 \pm 3.78$ | $\mathbf{55.42 \pm 4.97}$ | $47.67 \pm 4.19$ | $48.67 \pm 4.11$ | $48.58 \pm 5.12$ |
| **Dataset** | | | **COIL-20** | | |
| **kNN** | **JDRML** | **LRGMML** | **AML** | **MDMLCLDD** | **SUML (Ours)** |
| k=1 | $99.01 \pm 0.29$ | $\mathbf{99.24 \pm 0.35}$ | $98.73 \pm 0.49$ | $98.59 \pm 0.77$ | $98.96 \pm 0.55$ |
| k=3 | $97.67 \pm 0.77$ | $\mathbf{98.89 \pm 0.40}$ | $98.41 \pm 0.48$ | $97.78 \pm 0.99$ | $98.06 \pm 0.51$ |
| k=10 | $93.88 \pm 0.91$ | $\mathbf{96.95 \pm 0.65}$ | $95.33 \pm 1.14$ | $93.46 \pm 1.57$ | $94.57 \pm 1.38$ |
| k=20 | $90.21 \pm 1.51$ | $\mathbf{94.76 \pm 0.71}$ | $90.32 \pm 1.92$ | $89.17 \pm 2.05$ | $89.63 \pm 2.10$ |
| **Dataset** | | | **Isolet** | | |
| **kNN** | **JDRML** | **LRGMML** | **AML** | **MDMLCLDD** | **SUML (Ours)** |
| k=1 | $88.02 \pm 0.57$ | $\mathbf{89.44 \pm 0.50}$ | $87.89 \pm 0.74$ | $87.88 \pm 0.96$ | $87.78 \pm 1.01$ |
| k=3 | $86.60 \pm 0.72$ | $\mathbf{88.53 \pm 0.47}$ | $86.73 \pm 0.81$ | $86.58 \pm 0.81$ | $86.50 \pm 0.82$ |
| k=10 | $89.40 \pm 0.54$ | $\mathbf{90.46 \pm 0.47}$ | $89.50 \pm 0.84$ | $89.54 \pm 0.67$ | $88.99 \pm 0.57$ |
| k=20 | $89.59 \pm 0.65$ | $\mathbf{90.44 \pm 0.59}$ | $89.65 \pm 0.74$ | $89.62 \pm 0.59$ | $89.22 \pm 0.59$ |
| **Dataset** | | | **USPS** | | |
| **kNN** | **JDRML** | **LRGMML** | **AML** | **MDMLCLDD** | **SUML (Ours)** |
| k=1 | $96.99 \pm 0.23$ | $\mathbf{97.15 \pm 0.34}$ | $97.04 \pm 0.31$ | $97.05 \pm 0.34$ | $97.08 \pm 0.29$ |
| k=3 | $96.66 \pm 0.34$ | $96.79 \pm 0.41$ | $96.65 \pm 0.29$ | $\mathbf{96.82 \pm 0.24}$ | $96.80 \pm 0.36$ |
| k=10 | $95.76 \pm 0.26$ | $\mathbf{95.98 \pm 0.42}$ | $95.79 \pm 0.23$ | $95.77 \pm 0.25$ | $95.85 \pm 0.46$ |
| k=20 | $94.77 \pm 0.30$ | $\mathbf{94.99 \pm 0.49}$ | $94.77 \pm 0.35$ | $94.81 \pm 0.47$ | $94.84 \pm 0.29$ |
| **Labels** | | | **Yes** | | **No** |

## Time complexity analysis

We now present the computational time complexity of the major steps involved in our approach:

- **Cost Function:** The matrix multiplications require O($d^2|\mathcal{T}| + ld|\mathcal{T}|$). A matrix transpose and a matrix-vector product involved, each require O($l|\mathcal{T}|$). The cost computation requires O($|\mathcal{T}|$).
- **Gradients:** The gradient with respect to $\boldsymbol{L}$ requires O($d^2|\mathcal{T}| + d^2 l$). However, gradient with respect to $\boldsymbol{S}$ requires O($d^2|\mathcal{T}| + d^2 l + d^3$).
- **Parameter updates:** As search space of both our parameters can be constrained to the Euclidean space, the parameter update steps require simple additions that can be done in constant time.

The $O(d^3)$ term is due to multiplication of two $d\times d$ matrices arising in computation of gradients wrt $\boldsymbol{S}$. By ensuring $d < D$ in $\boldsymbol{P} \in \mathbb{R}^{D\times d}$, our method is scalable against dimensionality. Also, our approach is linear in terms of $|\mathcal{T}|$.

## Experiments

### Classification on benchmark datasets

We first compare the proposed SUML approach against the following recently proposed State-Of-The-Art (SOTA) *supervised* DML approaches that make use of class labels: JDRML (Harandi, Salzmann, and Hartley 2017), LRGMML (Bhutani et al. 2018), AML (Chen et al. 2018) and MDMLCLDD (Xie et al. 2018). We use the following benchmark datasets: AT&T Faces(Samaria and Harter 1994), COIL20 (Nene et al. 1996), Isolet (Lichman 2013) and USPS (Hull 1994). For each dataset, we perform 10 random splits with 70%-30% train-test ratio, and report the standard deviation along with the classification accuracies on the test data. For the supervised approaches, we learn

Table 2: Comparison of proposed method SUML against manifold learning approaches in terms of classification accuracy (in %, higher the better).

| Dataset | AT&T Faces | | | |
|---|---|---|---|---|
| **Method** | **k=1** | **k=3** | **k=10** | **k=20** |
| NPE | 83.67 ± 3.71 | 67.33 ± 5.12 | 45.42 ± 10.04 | 28.42 ± 10.31 |
| LPP | 84.92 ± 3.52 | 65.17 ± 3.88 | 41.42 ± 5.33 | 23.08 ± 4.43 |
| **SUML (Ours)** | **90.08 ± 1.82** | **81.17 ± 2.01** | **64.92 ± 4.22** | **48.58 ± 5.12** |
| **Dataset** | **COIL-20** | | | |
| NPE | 95.91 ± 0.91 | 94.00 ± 1.21 | 81.34 ± 1.24 | 52.29 ± 1.54 |
| LPP | 96.05 ± 0.86 | 93.95 ± 0.74 | 81.43 ± 1.15 | 48.59 ± 3.17 |
| **SUML (Ours)** | **98.96 ± 0.55** | **98.06 ± 0.55** | **94.57 ± 1.38** | **89.63 ± 1.38** |
| **Dataset** | **Isolet** | | | |
| NPE | 82.24 ± 1.38 | 80.47 ± 0.33 | 85.00 ± 0.42 | 84.80 ± 0.42 |
| LPP | 81.55 ± 0.87 | 79.85 ± 1.21 | 84.66 ± 0.92 | 84.53 ± 1.09 |
| **SUML (Ours)** | **87.78 ± 1.01** | **86.50 ± 1.01** | **88.99 ± 1.01** | **89.22 ± 0.59** |
| **Dataset** | **USPS** | | | |
| NPE | 91.19 ± 0.59 | 88.89 ± 0.66 | 81.41 ± 0.90 | 70.30 ± 1.09 |
| LPP | 90.91 ± 0.67 | 88.86 ± 0.74 | 80.51 ± 0.64 | 68.80 ± 0.73 |
| **SUML (Ours)** | **97.08 ± 0.29** | **96.80 ± 0.36** | **95.85 ± 0.46** | **94.84 ± 0.29** |

a distance metric using the labeled training data, and report the classification accuracies on the test data using the learned metric. For our approach, we use the same training data to learn a metric, but do not provide the class labels. The learned metric is used to classify the test data using kNN classifier (with varying values of k). Best results are shown in bold. As seen in Table 1, our method performs competitive despite being unsupervised. We also compare our method against unsupervised manifold learning techniques NPE (He et al. 2005) and LPP (He and Niyogi 2003), which easily get outperformed by SUML (Table 2).

Table 3: Comparison of proposed unsupervised method SUML against State-Of-The-Art (SOTA) DML approaches.

| Dataset | | JHMDB Dataset (Zero-shot scenario) | | | HMDB Dataset (Zero-shot scenario) | | |
|---|---|---|---|---|---|---|---|
| **Method** | **Class Labels** | **NMI** | **F** | **R@1** | **NMI** | **F** | **R@1** |
| JDRML | Yes | **70.68** | 63.33 | 84.68 | 48.77 | 28.80 | 72.81 |
| LRGMML | Yes | 69.06 | 61.33 | 86.29 | 49.54 | 29.57 | 72.72 |
| AML | Yes | 69.53 | 61.38 | **87.10** | **53.20** | **31.52** | **73.32** |
| MDMLCLDD | Yes | 69.72 | 62.28 | 87.09 | 51.17 | 30.27 | 73.26 |
| MOM | No | 65.21 | 57.12 | 80.91 | 41.36 | 24.39 | 67.14 |
| **SUML (Ours)** | No | 70.58 | **64.62** | 86.83 | 50.64 | 30.32 | 72.52 |
| **Dataset** | | **AwA2 Dataset + CUB (noise)** | | | **Fashion-MNIST Dataset +MNIST-Digit (noise)** | | |
| **Method** | **Class Labels** | **NMI** | **F** | **R@1** | **NMI** | **F** | **R@1** |
| JDRML | Yes | 83.19 | 78.73 | 93.85 | 59.75 | 47.69 | 77.40 |
| LRGMML | Yes | 82.25 | 77.59 | **94.80** | 59.44 | 49.47 | **78.60** |
| AML | Yes | 83.35 | 78.12 | 94.70 | **64.62** | **52.52** | 77.80 |
| MDMLCLDD | Yes | **84.26** | 78.76 | 94.60 | 63.88 | 50.20 | 77.70 |
| MOM | No | 68.11 | 62.24 | 87.05 | 53.71 | 42.24 | 73.10 |
| **SUML (Ours)** | No | 83.47 | **79.06** | 93.86 | 63.87 | 51.42 | 77.03 |

## Clustering and retrieval in zero-shot learning scenario, and in the presence of noise

For the task of Zero-Shot Learning (ZSL) we used two action recognition datasets: JHMDB (Jhuang et al. 2013) (classes 1-8 are for training and 14-21 are for testing) and HMDB (Kuehne et al. 2011) (classes 1-21 are for training and 32-51 are for testing). Due to the disjoint nature of training and testing classes, it corresponds to the ZSL scenario. The features were provided as part of (Cherian et al. 2018).

For DML in presence of noise, we used the AwA2 (Xian et al. 2018) and Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) datasets. For AwA2, the 10 largest classes were selected. From each chosen class, examples 1-200 are chosen for training, and examples 401-600 are for testing. Images from the first 10 classes of the Caltech-UCSD Birds 200 (CUB) dataset (Welinder et al. 2010) are added as noise. Features of both AwA2 and CUB were used as in (Xian et al. 2018). From the training split of Fashion-MNIST, the first 200 examples for each class are chosen. The first 200 examples of each class from the testing split are chosen. For inclusion to training data as noise, we add the first 150 training images from each class of the original MNIST dataset (LeCun et al. 1998), after adding random noise to each pixel.

For each dataset above, the size of the embedding is set as 64. We project the test data using the mapping learned from the train data. The test examples in the learned space are used to perform clustering and retrieval, to evaluate the compared approaches. As shown in Table 3, SUML performs competitive to the supervised SOTA approaches on all the four challenging datasets (best method is shown in bold, and second best is underlined). We also outperform the state-of-the-art unsupervised metric learning method MOM (Iscen et al. 2018). Additionally, in Table 4 we show that for the same four datasets, SUML outperforms the following classical DML approaches: NCA (Goldberger et al. 2005), ITML (Davis et al. 2007), KISSME (Koestinger et al. 2012), RVML (Perrot and Habrard 2015), SCML (Shi, Bellet, and Sha 2014) and DRIFT (Ye et al. 2017).

Table 4: Comparison of proposed unsupervised method SUML against classical supervised DML approaches.

| Dataset | | JHMDB Dataset (Zero-shot scenario) | | | HMDB Dataset (Zero-shot scenario) | | |
|---|---|---|---|---|---|---|---|
| **Method** | **Class Labels** | **NMI** | **F** | **R@1** | **NMI** | **F** | **R@1** |
| NCA | Yes | 63.39 | 50.43 | 76.08 | 36.79 | 21.18 | 63.23 |
| ITML | Yes | 68.82 | 58.26 | 85.48 | 46.43 | 27.66 | 71.95 |
| KISSME | Yes | 55.91 | 47.01 | 83.70 | 41.01 | 24.46 | 69.61 |
| RVML | Yes | 64.10 | 49.03 | 83.60 | 46.04 | 27.42 | 69.48 |
| SCML | Yes | 50.54 | 41.17 | 80.11 | 39.71 | 23.51 | 66.98 |
| DRIFT | Yes | 66.20 | 52.95 | 82.53 | 44.35 | 26.53 | 69.38 |
| **SUML (Ours)** | No | **70.58** | **64.62** | **86.83** | **50.64** | **30.32** | **72.52** |
| **Dataset** | | **AwA2 Dataset + CUB (noise)** | | | **Fashion-MNIST Dataset +MNIST-Digit (noise)** | | |
| **Method** | **Class Labels** | **NMI** | **F** | **R@1** | **NMI** | **F** | **R@1** |
| NCA | Yes | 54.71 | 41.60 | 85.05 | 50.06 | 40.17 | 67.90 |
| ITML | Yes | 73.72 | 66.15 | 92.15 | 49.09 | 37.95 | 74.60 |
| KISSME | Yes | 75.70 | 68.21 | 91.05 | 53.92 | 42.49 | 74.90 |
| RVML | Yes | 78.81 | 72.00 | 89.50 | 50.79 | 39.39 | 75.00 |
| SCML | Yes | 73.28 | 66.44 | 89.95 | 52.32 | 41.49 | 72.40 |
| DRIFT | Yes | 80.19 | 74.44 | 90.85 | 51.98 | 42.64 | 64.00 |
| **SUML (Ours)** | No | **83.47** | **79.06** | **93.86** | **63.87** | **51.42** | **77.03** |

## Hyperparameter tuning and Ablation studies

Tuning of hyperparameters in the unsupervised setting is not trivial. But, due to the simplicity of our model, we only have a single hyperparameter $\alpha > 0$ and a trade-off parameter $\lambda > 0$. $\alpha > 0$ being an angle, is fairly easy

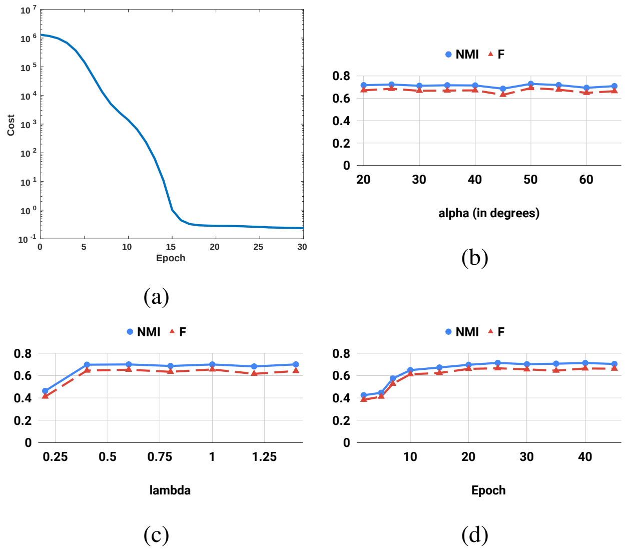

Figure 3: a) Convergence behaviour of SUML on the JHMDB dataset, b) Sensitivity towards $\alpha$, with respect to NMI and F values on the JHMDB test data, c) Sensitivity towards $\lambda$, with respect to NMI and F values on the JHMDB test data, d) NMI and F values on the JHMDB test data during progress of training of SUML.

to tune in the range $30°$ to $60°$, as suggested in (Wang et al. 2017). We do not tune it, but set it to $40°$ for all our experiments. We do not give too high weightage to the regularizer term, and hence set $\lambda$ to a low value of 0.5 for all our experiments. We observed that it can also be set to as low as $10^{-3}$. We used the Manopt toolbox (Boumal et al. 2014) to perform Riemannian Conjugate Gradient Descent (RCGD) to jointly learn $\boldsymbol{S}$ and $\boldsymbol{L}$. We keep all the default parameters of Manopt, and trained for a maximum of $num\_ep = 30$ epochs for all datasets. Our method converges quite fast, owing to the Riemannian product manifold based optimization (Figure 3a).

We fix $\lambda = 0.5$, $num\_ep = 30$ and observe the NMI and F-values on JHMDB test data with varying $\alpha$, and observe that SUML is robust to $\alpha$ (Figure 3b). However, upon fixing $\alpha = 40°, num\_ep = 30$ and varying $\lambda$, we observe that setting too low value of $\lambda$ decreases performance (Figure 3c). This shows the importance of the orthogonality constraint. Figure 3d shows that initially the performance of our method is poor (as we initialize $\boldsymbol{S}$ and $\boldsymbol{L}$ randomly), but gradually improves. This also shows that despite a random initialization, an optimal solution could be reached due to the product manifold based optimization.

## Experiments on large-scale data

We perform large-scale experiments to evaluate our stochastic method sSUML on the following datasets: STL-10 (Coates, Ng, and Lee 2011) and ImageNet (Russakovsky et al. 2015). Our stochastic method is again competitive with respect to the baselines (see Table 5). It should be noted that the performance of all the approaches are relatively poor in the STL-10 dataset. This is because we intentionally use the raw pixel values and reduce the dimensionality to 100 causing severe loss of information, to show that despite a poor initial representation, our method can perform competitive. For the ImageNet dataset, we followed the ImNet-2 protocol as in (Kodirov, Xiang, and Gong 2017). In both STL-10 and ImageNet, we fix $maxiter = 10, \alpha = 40°, \lambda = 0.5$ in Algorithm 1. Mini-batch size is 120, and embedding size is 64. Figure 4 shows convergence behaviour of our method on ImageNet.

Table 5: Comparison of the proposed stochastic SUML (sSUML) approach against various baselines, on STL-10 and ImageNet datasets.

| **Dataset** | **STL** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Method** | **Class Labels** | **Mini-batch based** | **NMI** | **R@1** | **R@2** | **R@4** | **R@8** |
| KISSME | Yes | No | 12.1 | 25.5 | 38.5 | 55.1 | 73.1 |
| JDRML | Yes | No | 14.9 | 30.2 | 44.9 | 61.3 | 77.5 |
| LRGMML | Yes | No | 15.2 | 30.4 | 45.5 | 61.8 | 77.6 |
| AML-stoch | Yes | Yes | 15.0 | 30.3 | 44.8 | 61.2 | 76.9 |
| MDMLCLDD | Yes | No | 15.1 | 30.1 | 44.7 | 61.1 | 77.0 |
| **sSUML (Ours)** | No | Yes | 14.3 | 29.4 | 44.1 | 60.5 | 76.7 |
| **Dataset** | **ImageNet** | | | | | | |
| KISSME | Yes | No | 56.5 | 39.4 | 49.2 | 58.8 | 68.1 |
| AML-stoch | Yes | Yes | 58.1 | 42.2 | 52.5 | 62.1 | 71.1 |
| **sSUML (Ours)** | No | Yes | 56.6 | 38.9 | 48.8 | 58.7 | 68.3 |

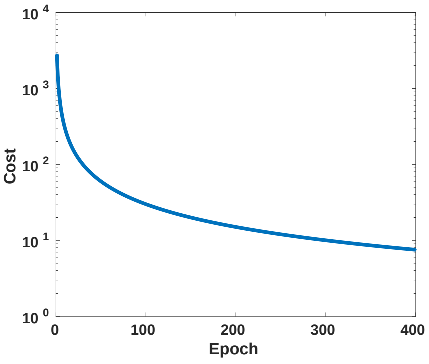

Figure 4: Convergence behaviour of our proposed stochastic method on the ImageNet dataset.

## Empirical runtime evaluation

The softmax-based metric learning method NCA takes 864s on the HMDB dataset, whereas ours take only 1.38s. Triplet-loss based DRIFT is 8.74 times slower than ours. ITML is 1.9 times slower than us. Faster, low-rank approach LRGMML (1.36s) is almost similar to ours. We require small mini-batch sizes (*e.g.*, 120). For each mini-batch, we only need about $10^3$ triplets, evaluating on which, takes very less time even on ImageNet (0.77 seconds). Even increasing number of triplets to $10^4$, we only require 6.6 seconds per mini-batch, which is quite less in practice. Due to the product manifold based optimization, our method converges quite fast, and we do not have problem scaling up.

## Clustering and retrieval for Fine-Grained Visual Categorization (FGVC)

We also compare our proposed stochastic method against the following recently proposed, deep unsupervised feature and metric learning methods: Exemplar (Dosovitskiy et al. 2016), NCE (Wu et al. 2018), DeepCluster (Caron et al. 2018) and MOM (Iscen et al. 2018). The following benchmark fine-grained datasets were used: CUB 200 (Welinder et al. 2010), Cars 196 (Krause et al. 2013) and Stanford Online Products (SOP) (Oh Song et al. 2016). GoogLeNet (Szegedy et al. 2015) pretrained on ImageNet (Russakovsky et al. 2015), has been used as the backbone CNN, using the MatConvNet (Vedaldi and Lenc 2015) tool. Recent metric learning approaches concluded their robustness towards embedding size in FGVC, and hence, following the evaluation protocol in (Wang et al. 2017; Iscen et al. 2018), we set the embedding size to 512, except for Cars, where we set it as 128. We fix $\alpha = 45°$ and $\lambda = 0.5$. We used mini-batch size of 120 and set $maxiter = 10$ in Algorithm 1. The results of comparison have been shown in Table 6. Our method performs the best in Cars and SOP datasets, and second best on CUB, where the MOM approach performs better than ours because of their careful mining strategy. We also report our own initial performance obtained using a random initialization. Despite a poor initialization, our method is capable of improving the performance.

Table 6: Comparison against deep unsupervised approaches on three benchmark fine-grained datasets.

| Dataset | **CUB 200** | | | | | |
|---|---|---|---|---|---|---|
| **Method** | **Class Labels** | **NMI** | **R@1** | **R@2** | **R@4** | **R@8** |
| Initial (Random) | No | 34.3 | 31.7 | 42.7 | 54.8 | 67.0 |
| Exemplar | No | 45.0 | 38.2 | 50.3 | 62.8 | 75.0 |
| NCE | No | 45.1 | 39.2 | 51.4 | 63.7 | 75.8 |
| DeepCluster | No | 53.0 | 42.9 | 54.1 | 65.6 | 76.2 |
| MOM | No | **55.0** | **45.3** | **57.8** | **68.6** | 78.4 |
| **sSUML(Ours)** | No | 53.4 | 43.5 | 56.2 | 68.3 | **79.1** |

| Dataset | **Cars 196** | | | | | |
|---|---|---|---|---|---|---|
| **Method** | **Class Labels** | **NMI** | **R@1** | **R@2** | **R@4** | **R@8** |
| Initial (Random) | No | 23.4 | 22.3 | 31.4 | 41.9 | 54.1 |
| Exemplar | No | 35.4 | 36.5 | 48.1 | 59.2 | 71.0 |
| NCE | No | 35.6 | 37.5 | 48.7 | 59.8 | 71.5 |
| DeepCluster | No | 38.5 | 32.6 | 43.8 | 57.0 | 69.5 |
| MOM | No | **38.6** | 35.5 | 48.2 | 60.6 | 72.4 |
| **sSUML(Ours)** | No | 37.6 | **42.0** | **54.3** | **66.0** | **77.2** |

| Dataset | **Stanford Online Products (SOP)** | | | | |
|---|---|---|---|---|---|
| **Method** | **Class Labels** | **NMI** | **R@1** | **R@10** | **R@100** |
| Initial (Random) | No | 79.7 | 25.4 | 35.6 | 48.6 |
| Exemplar | No | 85.0 | 45.0 | 60.3 | 75.2 |
| NCE | No | **85.8** | 46.6 | 62.3 | 76.8 |
| DeepCluster | No | 82.8 | 34.6 | 52.6 | 66.8 |
| MOM | No | 84.4 | 43.3 | 57.2 | 73.2 |
| **sSUML(Ours)** | No | 81.2 | **47.8** | **63.6** | **78.3** |

## Conclusion

An unsupervised metric learning approach has been proposed that learns a metric using synthetic constraints, which are obtained using randomly sampled triplets from unlabeled data. This is done by minimizing the entropy of a conditional probability over the triplets. The proposed method performs competitive to many state-of-the-art and classical metric learning approaches, despite not using class labels for training.

## References

Bai, S.; Zhou, Z.; Wang, J.; Bai, X.; Jan Latecki, L.; and Tian, Q. 2017. Ensemble diffusion for retrieval. In *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 774–783.

Bellet, A.; Habrard, A.; and Sebban, M. 2015. Metric learning synthesis lectures on artificial intelligence and machine learning. *Morgan & Claypool Publishers, San Rafael.*

Bhutani, M.; Jawanpuria, P.; Kasai, H.; and Mishra, B. 2018. Low-rank geometric mean metric learning. *arXiv preprint arXiv:1806.05454*.

Boumal, N.; Mishra, B.; Absil, P.-A.; and Sepulchre, R. 2014. Manopt, a Matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research* 15(1):1455–1459.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *Proc. of European Conference on Computer Vision (ECCV)*, 132–149.

Chen, S.; Gong, C.; Yang, J.; Li, X.; Wei, Y.; and Li, J. 2018. Adversarial metric learning. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Cherian, A.; Sra, S.; Gould, S.; and Hartley, R. 2018. Non-linear temporal subspace representations for activity recognition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2197–2206.

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223.

Cox, T. F., and Cox, M. A. 2000. *Multidimensional scaling*. CRC press.

Davis, J. V.; Kulis, B.; Jain, P.; Sra, S.; and Dhillon, I. S. 2007. Information-theoretic metric learning. In *Proc. of International Conference on Machine Learning (ICML)*, 209–216.

Donoser, M., and Bischof, H. 2013. Diffusion processes for retrieval revisited. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1320–1327.

Dosovitskiy, A.; Fischer, P.; Springenberg, J.; Riedmiller, M.; and Brox, T. 2016. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 38(9):1734–1747.

Duan, Y.; Zheng, W.; Lin, X.; Lu, J.; and Zhou, J. 2018. Deep adversarial metric learning. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goldberger, J.; Hinton, G. E.; Roweis, S. T.; and Salakhutdinov, R. R. 2005. Neighbourhood components analysis. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 513–520.

Harandi, M.; Salzmann, M.; and Hartley, R. 2017. Joint dimensionality reduction and metric learning: A geometric take. In *Proc. of International Conference on Machine Learning (ICML)*.

He, X., and Niyogi, P. 2003. Locality preserving projections. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 153–160.

He, X.; Cai, D.; Yan, S.; and Zhang, H.-J. 2005. Neighborhood preserving embedding. In *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 1208–1213.

Hull, J. J. 1994. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence* 16(5):550–554.

Iscen, A.; Tolias, G.; Avrithis, Y.; Furon, T.; and Chum, O. 2017. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2077–2086.

Iscen, A.; Tolias, G.; Avrithis, Y.; and Chum, O. 2018. Mining on manifolds: Metric learning without labels. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jhuang, H.; Gall, J.; Zuffi, S.; Schmid, C.; and Black, M. J. 2013. Towards understanding action recognition. In *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 3192–3199.

Kodirov, E.; Xiang, T.; and Gong, S. 2017. Semantic autoencoder for zero-shot learning. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3174–3183.

Koestinger, M.; Hirzer, M.; Wohlhart, P.; Roth, P. M.; and Bischof, H. 2012. Large scale metric learning from equivalence constraints. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2288–2295.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *Proc. of IEEE International Conference on Computer Vision Workshops (ICCVW)*, 554–561.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. HMDB: a large video database for human motion recognition. In *Proc. of IEEE International Conference on Computer Vision (ICCV)*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Li, D.; Hung, W.-C.; Huang, J.-B.; Wang, S.; Ahuja, N.; and Yang, M.-H. 2016. Unsupervised visual representation learning by graph-based consistent constraints. In *Proc. of European Conference on Computer Vision (ECCV)*, 678–694. Springer.

Lichman, M. 2013. UCI machine learning repository. *URL http://archive.ics.uci.edu/ml*.

Miyato, T.; Maeda, S.-i.; Ishii, S.; and Koyama, M. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Nene, S. A.; Nayar, S. K.; Murase, H.; et al. 1996. Columbia object image library (COIL-20). *Technical report CUCS-005-96*.

Oh Song, H.; Xiang, Y.; Jegelka, S.; and Savarese, S. 2016. Deep metric learning via lifted structured feature embedding. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4004–4012.

Perrot, M., and Habrard, A. 2015. Regressive virtual metric learning. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 1810–1818.

Qian, Q.; Jin, R.; Zhu, S.; and Lin, Y. 2015. Fine-grained visual categorization via multi-stage metric learning. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3716–3724.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision (IJCV)* 115(3):211–252.

Samaria, F. S., and Harter, A. C. 1994. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, 138–142. IEEE.

Shi, Y.; Bellet, A.; and Sha, F. 2014. Sparse compositional metric learning. In *Proc. of Association for the Advancement of Artificial Intelligence (AAAI)*, 2078–2084.

Sohn, K. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 1857–1865.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290(5500):2319–2323.

Vedaldi, A., and Lenc, K. 2015. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, 689–692. ACM.

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

Weinberger, K. Q.; Blitzer, J.; and Saul, L. 2006. Distance metric learning for large margin nearest neighbor classification. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 1473–1480.

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

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3733–3742.

Xian, Y.; Lampert, C. H.; Schiele, B.; and Akata, Z. 2018. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, P.; Wu, W.; Zhu, Y.; and Xing, E. P. 2018. Orthogonality-promoting distance metric learning: convex relaxation and theoretical analysis. In *Proc. of International Conference on Machine Learning (ICML)*.

Ye, H.-J.; Zhan, D.-C.; Si, X.-M.; and Jiang, Y. 2017. Learning mahalanobis distance metric: Considering instance disturbance helps. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, 3315–3321.