VARIATIONAL AUTOENCODERS FOR SEMI-SUPERVISED DEEP METRIC LEARNING

by

NATHAN SAFIR

(Under the Direction of Shannon Quinn)

ABSTRACT

Deep metric learning (DML) methods generally do not incorporate unlabelled data. We propose performing borrowing components of the variational autoencoder (VAE) methodology to extend DML methods to train on semi-supervised datasets. We experimentally evaluate atomic benefits to the performing DML on the VAE latent space such as the enhanced ability to train using unlabelled data and to induce bias given prior knowledge.

INDEX WORDS: Variational Autoencoders, Metric Learning, Deep Learning, Representation Learning, Generative Models

VARIATIONAL AUTOENCODERS FOR SEMI-SUPERVISED DEEP METRIC LEARNING

by

NATHAN SAFIR

B.S., University of Georgia, 2022

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2022

VARIATIONAL AUTOENCODERS FOR SEMI-SUPERVISED DEEP METRIC LEARNING

by

NATHAN SAFIR

Major Professor: Dr. Shannon Quinn

Committee: Dr. Frederick Maier
Dr. Sheng Li

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
May 2022

ACKNOWLEDGMENTS

I would like to thank my major advisor and research mentor Dr. Shannon Quinn for his mentorship, advice, patience, and material help offered throughout my research career as an undergraduate and graduate student. I would also like to thank Dr. Frederick Maier for his help in navigating the administrative processes of the Artificial Intelligence master's program, and Dr. Sheng Li for also serving on my committee and reviewing my thesis.

I would like to thank my labmates Meekail Zain, Curtis Godwin, Bella Humphrey, and Eric Miller and others who have worked in and outside the cilia team within the Quinn Research Group for the past several years for their help and encouragement. I would specifically like to thank Meekail Zain for his unwavering support and help throughout this research.

I would lastly like to thank my family for their support and words of encouragement throughout the thesis writing process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This thesis looks to explore how components of the variational autoencoder can improve the performance of deep metric learning tasks. Chapter two will review previous literature which researches similar or adjacent problems in semi-supervised metric learning and variational autoencoders. Chapters three and four will give a greater overview of metric learning and variational autoencoders, respectively. Chapter five will introduce a methodology to evaluate the claims we make about how components of the variational autoencoder can be used to extend deep metric learning models to train with unlabelled data. Chapter six will provide the results of our proposed methods, and chapter seven will discuss what we can learn from these results and how this line of research can be extended further. The remainder of this chapter gives a brief overview of the motivation and goals for this research.

§ 1.1 Motivation

Within the broader field of representation learning, metric learning is an area which looks to define a distance metric which is smaller between similar objects (such as objects of the same class) and larger between dissimilar objects. Oftentimes, a map is learned from inputs into a low-dimensional latent space where euclidean distance exhibits this relationship, encouraged by training said map against a loss (cost) function based on the euclidean distance between sets of similar and dissimilar objects in the latent space. Existing metric learning methods are generally unable to learn from unlabelled data, which is problematic because unlabelled data is often easier to obtain and is potentially informative.

§ 1.2 Objective

We take inspiration from variational autoencoders (VAEs), a generative representation learning architecture, for using unlabelled data to create accurate representations. Specifically, we look to evaluate three atomic claims that detail how pieces of the VAE architecture can create a better deep metric learning (DML) model on a semi-supervised dataset. From here, we can ascertain which specific qualities of how VAEs process unlabelled data are most helpful in modifying DML methods to train with semi-supervised datasets.

First, we propose that the autoencoder structure of the VAE helps the clustering of unlabelled points, as the reconstruction loss may help incorporate semantic information from unlabelled sources. Second, we claim that the structure of the VAE latent space, as it is confined by a prior distribution, can be used to induce bias in the latent space of a DML system. For instance, if we know a dataset contains N-many classes, creating a prior distribution that is a learnable mixture of N gaussians may help produce better representations. Third, we claim that performing DML on the latent space of the VAE so that the DML task can be jointly optimized with the VAE to incorporate unlabelled data may help produce better representations.

Each of the three claims will be evaluated experimentally. The claims will be evaluated by comparing a standard DML implementation to the same DML implementation

- jointly optimized with an autoencoder
- while structuring the latent space around a prior distribution using the VAE's KL-divergence loss term between the approximated posterior and prior
- jointly optimized with a VAE

Our primary contribution is evaluating these three claims. Our secondary contribution is presenting the results of the joint approaches for VAEs and DML for more recent metric losses that have not been jointly optimized with a VAE in previous literature.

CHAPTER 2

RELATED LITERATURE

The goal of this research is to investigate how components of the variational autoencoder can help the performance of deep metric learning in semi supervised tasks. We draw on previous literature to find not only prior attempts at this specific research goal but also work in adjacent research questions that proves insightful. In this review of the literature, we discuss previous related work in the areas of Semi-Supervised Metric Learning, Semi-supervised VAEs, and VAEs with Metric Losses.

§ 2.1 Semi-Supervised Metric Learning

There have been previous approaches to designing metric learning architectures which incorporate unlabelled data into the metric learning training regimen for semi-supervised datasets. One of the original approaches is the MPCK-MEANS algorithm proposed by Bilenko et al. (2004), which adds a penalty for placing labelled inputs in the same cluster which are of a different class or in different clusters if they are of the same class. This penalty is proportional to the metric distance between the pair of inputs. Baghshah and Shouraki (2009) also looks to impose similar constraints by introducing a loss term to preserve locally linear relationships between labelled and unlabelled data in the input space. Wang et al. (2013) also use a regularizer term to preserve the topology of the input space. Using VAEs, in a sense, draws on this theme: though there is not explicit term to enforce that the topology of the input space is preserved, a topology of the inputs is intended to be learned through a low-dimensional manifold in the latent space.

One more recent common general approach to this problem is to use the unlabelled data's proximity to the labelled data to estimate labels for unlabelled data, effectively transforming unlabelled data into labelled data. Dutta et al. (2021) propose a model which uses affinity propagation on a k-Nearest-Neighbors graph to label partitions of unlabelled data based on their closest neighbors in the latent space. Wu et al. (2020) also look to assign pseudo-labels to unlabelled data, but not through a graph-based approach. Instead, the proposed model looks to approximate "soft" pseudo-labels for unlabelled data from the metric learning similarity measure between the embedding of unlabelled data and the center of each input of each class of the labelled data.

§ 2.2 Semi-supervised VAEs

There have been previous attempts at incorporating labelled information into the VAE framework. As discussed before, the VAE training regimen does not incorporate training labels – creating a training regimen for the VAE which does learn from labels is not straightforward. An early solution proposed to this problem are Kingma and Welling's proposed M1 and M2 models (2014). The M1 model trains the VAE on data X without the labels Y to produce encodings Z and then trains a separate model on a supervised task with the data and labels pair $(Z; Y)$. The M1 model does not actually train the underlying VAE differently, so the authors propose an M2 model, which differs from the vanilla VAE in that there are two encoders which produce not only the latent vector z for each datapoint x but also a predicted label y', both of which the decoder receives as input. The classification task (i.e. the encoder's prediction y') is trained jointly with the regular VAE loss, as is consistent with the authors' new derivation of the VAE ELBO.

A more recent approach to the semi-supervised VAEs discourages producing an explicit label embedding within the latent space. Joy et al. (2020) propose a model which encodes several latent vectors $z_1; z_2; \dots z_n$ for n labelled characteristics of the image. For instance, if the dataset was over pictures of people, one characteristic may be if the person was smiling, if they were blonde, etc. For each characteristic c_i,

a classifier is trained to predict y_i from only the latent vector z_i. The authors argue that this is a superior training approach than creating explicit label embeddings with an encoder network as binary labels such as "smiling/not smiling" are oftentimes not actually binary (ex. a picture may show the subject slightly smiling or greatly smiling)

§ 2.3 VAEs with Metric Loss

Some approaches to incorporating labelled data into VAEs use a metric loss to govern the latent space more explicitly. Lin et al. (2018) model the intra-class invariance (i.e. the class-related information of a data point) and intra-class variance (i.e. the distinct features of a data point not unique to it's class) seperately. Like several other models in this section, this paper's proposed model incorporates a metric loss term for the latent vectors representing intra-class invariance and the latent vectors representing both intra-class invariance and intra-class variance.

Kulkarni et al. (2020) incorporate labelled information into the VAE methodology in two ways. First, a modified architecture called the CVAE is used in which the encoder and generator of the VAE is not only conditioned on the input X and latent vector z, respectively, but also on the label Y. The CVAE was introduced in previous papers (Sohn et al., 2015) (Dahmani et al., 2019). Second, the authors add a metric loss, specifically a multi-class N-pair loss (Sohn, 2016), in the overall loss function of the model. While it is unclear how the CVAE technique would be adapted in a semi-supervised setting, as there is not a label Y associated with each datapoint X, we also experiment with adding a (different) metric loss to the overall VAE loss function.

Most recently, Grosnit et al. (2021) leverage a new training algorithm for combining VAEs and DML for Bayesian Optimization and said algorithm using simple, contrastive, and triplet metric losses. We look to build on this literature by also testing a combined VAE DML architecture on more recent metric losses, albeit using a simpler training regimen.

Lastly, though the paper does not discuss VAEs, it is worthwhile to note Andresini et al.'s (2021) combined approach to metric learning (specifically triplet loss) and autoencoders. For a dataset with two classes, two autoencoders are trained on only one class, so triplets can be formed using an anchor point sampled from the dataset, the reconstruction of the autoencoder for the positive class, and the autoencoder for the triplet of the negative class. The authors claim that this method, along with other benefits, do not suffer the convergence problems of many triplet loss DML architectures as the triplets are not randomly sampled.

CHAPTER 3

METRIC LEARNING

§ 3.1 Metric Learning as Representation Learning

It is paramount for almost all tasks within the field of machine learning to build meaningful representations of data. For instance, building a representation of data is a necessary step for regression, classification, and clustering tasks. Multilayer perceptrons and other neural network models create successive representations of the input data before producing the output. Yoshua Bengio defines representation learning as "learning representations of the data that make it easier to extract useful information when building classifiers or other predictors" (2013).

Metric learning can thus be thought of as a type of representation learning. In metric learning, we attempt to learn an embedding space where similar samples' representations are close together (with respect to the ambient euclidean metric) and dissimilar samples representations' are far apart. Thus, the low-dimensional embeddings carry information in their relative distances between other embeddings in the latent space.

§ 3.2 Metric Learning Overview

Metric learning attempts to create representations for data by training against the similarity or dissimilarity of samples. In a more technical sense, there are two notable functions in DML systems. Function f is a neural network which maps the input data X to the latent points Z (i.e. $f : X \mapsto Z$, where is the

network parameters). Generally, Z exists in a space of much lower dimensionality than X (eg. X is a set of $28 \quad 28$ pixel pictures such that $X \quad R^{28 \ 28}$ and $Z \quad R^{10}$).

The function $D_f(x; y) = D(f(x); f(y))$ represents the distance between two inputs $x; y \ 2 \ X$. To create a useful embedding model f, we would like for f to produce large values of $D_f(x; y)$ when x and y are dissimilar and for f to produce small values of $D_f(x; y)$ when x and y are similar. In some cases, dissimilarity and similarity can refer to when inputs are of different and the same classes, respectively.

It is common for the Euclidean metric (i.e. the L_2 metric) to be used as a distance function in metric learning. The generalized L_p metric can be defined as follows, where $z_0; z1 \ 2 \ R^d$.

$$D_p(z_0; z_1) = jjz_0 \quad z_1 jj_p = (\sum_{i=1}^{} jz_{0_i} \quad z_{1_i} j^p)^{1=p}$$

If we have chosen f (a neural network) and the distance function D (the L_2 metric), the remaining component to be defined in a metric learning system is the loss function for training f. The following section provides a survey of the development of and differences between notable training objectives in metric learning, which for brevity we will refer to as *metric loss functions* or *metric losses*.

§ 3.3 Survey of Metric Loss Functions

§ 3.3.1 Contrastive Metric Losses

In cases where each input of a dataset has a class label, contrastive metric losses encourage samples $x_1; x_2$ to have a small value $D_f(x; y)$ if $y_1 = y_2$ and for $D_f(x; y)$ to have a large value if $y_1 \neq y_2$, where $y \ 2 \ Y$ is an object of the set of labels for X. In other words, contrastive metric losses encourage similar samples or samples of the same class to be close together or dissimilar samples or samples of different classes to be far apart.

One of the first contrastive metric losses is contrastive loss (Chopra et al., 2005). Contrastive loss minimizes the distance between a pair of points $f(x_1); f(x_2)$ if they are of the same class and maximizes the distance if the points are of different class by defining the following loss function.

$$L_{\text{contrastive}}(\ ; x_1; x_2; y_1; y_2) = y_1 = y_2 L_G(D_f((x_1; x_2)) + y_1 \neq y_2 L_I(D_f((x_1; x_2))$$

In this loss function, L_G refers to the loss function for "genuine pairs" of the same class, L_I refers to the loss function for "imposter pairs" of different classes, and 1_b is a piece-wise function that is 1 if b is true and 0 if b is false. One common instantiation of these loss functions is the following (Weng, 2021).

$$L_G(D) = D$$

$$L_I(D) = \max(0; \quad D)$$

In this instantiation, L_G is simply the identity function, or the raw distance, and L_I is the distance subtracted from a hyperparameter and constrained by a lower bound of 0. Intuitively, the function looks to minimize distance between embeddings in similar pairs and maximize distance between embeddings in dissimilar pairs.

Triplet loss is another simple contrastive metric loss and perhaps the most popular loss used in metric learning today. Triplet loss was originally proposed as follows (Schroff et al., 2015).

$$L_{\text{triplet}}(\ ; x_a; x_p; x_n) = jD(f(x_a); f(x_p)) \quad D(f(x_a); f(x_n)) + \quad j$$

In this loss objective, x_a is an anchor point, x_p is a positive sample (of the same class as the anchor point), and x_n is a negative sample (of a different class than the anchor point). These three points constitute a triplet in triplet loss. In the original paper, the authors note that "hard triplets", or triplets in which the distance between positive samples $D(f(x_a); f(x_p))$ exceeds or is within a certain bound of the

distance between negative samples $D(f\ (x_a); f\ (x_n))$ are especially crucial for training.

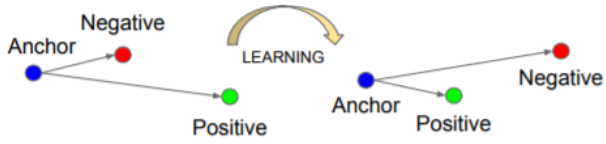

Figure 3.1: Diagram of the relative distances between triplet points. The distance between the anchor and the positive point should be minimized and the distance between the anchor and the negative point should be maximized. (Source: Schroff et al., 2015)

N-pair loss (Sohn, 2016) extends on triplet loss to include more than one negative sample for each anchor and positive sample pairing. The authors write that by only comparing the anchor against one negative sample per loss computation, it takes much longer for the loss to converge. N-pair loss is especially useful for datasets in which there is more than two classes, as it is often the case that there are more negative samples than positive samples for each point. Consider a tuple of $N + 1$ elements consisting of an anchor point x_a, a positive sample x_p, and $n \quad 1$ negative samples $x_n^1; x_n^2; :::; x_n^{N \ 1}$. We can then define the loss objective as follows.

$$L_{\text{N-pairs}}(\ ; x_a; x_p; x_n^1; x_n^2; :::; x_n^{N \ 1}) = log(1 + \sum_{i=1}^{N-1} \exp(f\ (x)^T f\ (x_n^i) \quad f\ (x)^T f\ (x_p))$$

The N-pair loss differs from triplet loss not only in the use of multiple negative samples but primarily in it's use of the inner-product in the place of a distance metric.

There are several general problems with contrastive approaches that have recently motivated research on alternative loss functions (Kha Vu, 2021). One of these issues is the computational complexity of sample mining. For most contrastive approaches (we will use triplet loss for this example) the most difficult triplets (in terms of minimizing the loss function) should be mined for the model to converge quickly. However, finding such triplets is often computationally expensive (Kha Vu, 2021). Another issue is that contrastive approaches do not necessarily pull together objects of the same class to the same region of the latent space (Kha Vu, 2021). Both of these issues are addressed by more recent metric loss functions.

§ 3.3.2 Modern Metric Losses

Modern metric loss functions address some of the shortcomings of traditional contrastive metric losses, such as supervised contrastive loss (Khosla et al., 2020). Though originally proposed for a self-supervised learning problem, further generalizes triplet and N-pair loss by using many positive and many negative samples for each loss computation, which the authors attribute as the reason the loss is able to converge quickly without mining for hard samples.

$$L_{out}^{sup} = \sum_{i2I} \frac{1}{jP(i)j} \sum_{p2P(i)} log(\frac{\exp(z_i \quad z_p=\ )}{\sum_{a2A(i)} \exp(z_i \quad z_a=\ )})$$

$$L_{in}^{sup} = \sum_{i2I} log(\frac{1}{jP(i)j} \sum_{p2P(i)} \frac{\exp(z_i \quad z_p=\ )}{\sum_{a2A(i)} \exp(z_i \quad z_a=\ )})$$

The two implementations proposed in the paper differ only in taking the logarithm outside or inside the summation. In both equations, $I = 1; :::; 2N$ is the set of indexes in the batch, $A(i) = I \; n \; fig$ is the set of indexes excluding i, $P(i) = fp \; 2 \; A(i) : y_p = y_i g$ is the set of positive indexes in the batch for a given index i excluding i itself, and $\quad 2 \; R^+$ is a temperature hyperparameter. Like the N-pair loss, the supervised contrastive loss is modeled around the softmax loss. Unlike the N-pair loss, the supervised contrastive loss can be computed once per batch and includes multiple *positive* samples in one computation.

The center loss is another important modern metric loss in solving some of the problems with contrasting learning. (Wen et al., 2016). To address the previously mentioned problem of latent points not self-organizing into regions based on class, the authors introduce a center loss L_C to encourage a point of class y_i to be close to the corresponding class center. The class center c_{y_i} for class y_i is a trainable parameter which updates with the overall loss function, which is stated below.

$$L_C = \frac{1}{2} \sum_{i=1}^{m} jjx_i \quad c_{y_i} jj_2$$

$$L_S = \sum_{i=1}^{n} g_{y_i}(x_i)$$

$$L = L_S + L_C$$

In the above equations, the center loss L_C is added to a softmax loss L_S in the overall loss function. g_c refers to the softmax classification probability for class c, and n is the number of elements in the batch.

Center loss can be viewed as one of the first steps in many of today's state of the art metric learning objectives. The SphereFace loss (Liu et al., 2017) is largely a modification of center loss's softmax function. SphereFace normalizes the class centers to be points on a hypersphere which gurantees consistency for inter-class variability. This is accomplished through normalizing of each row of the weight's matrix in the softmax function such that the norm of the row vector is 1. SphereFace additionally adds a margin term such that the decision boundary between classification of a point with softmax is more robust. Other State-of-the-Art metric losses, such as CosFace (H. Wang et al., 2018), ArcFace (Deng et al., 2019), and AdaCos (Zhang et al., 2019) are largely inspired by SphereFace and further explore the idea of a margin term for a decision boundary in angular projection (Kha Vu, 2021).

CHAPTER 4

VARIATIONAL AUTOENCODERS

§ 4.1 VAEs as Representation Learning

VAEs (Kingma and Welling, 2013) can be considered as a type of representation learning as VAEs also produce meaningful low-dimensional representations of input data. Both DML methods and VAEs are parameterized by neural networks which serve as a mapping from the input space to the latent dimensionality.

§ 4.2 VAEs as a Generative Model

Generative models are a class of models which attempt to approximate the distribution of input data $P(X)$. For instance, one may attempt to approximate the distribution of all pictures of dogs $P(X_{dog})$ with a generative model. If the generative model is successful, one can approximately sample from $P(X_{dog})$ through the generative model. However, the challenge in this approach is that many classes of inputs, including images, are very high-dimensional, and thus it is difficult to learn its distribution. For instance, a picture of a dog may be 256 pixels in both length and width and have 3 channels per pixel, in which case $X_{dog} \ 2 \ R^{256 \ 256 \ 3}$. The curse of dimensionality states that it is exceedingly difficult to learn a high-dimensional distribution, such as a probability distribution of dimensionality $256 \quad 256 \quad 3$, so generative models must use a work-around to contend with this dimensionality problem. VAEs, as we will discuss in the next section, define a low-dimensional latent variable z.

§ 4.3 The VAE Objective Function

To understand the derivation of the VAE objective function, it is first helpful to define the components of the VAE architecture in terms of probability theory. The encoder of the VAE attempts to estimate the posterior distribution $p(z|x)$ so we will refer to it as $q_{\phi}(z|x)$, where ϕ is the parameters of the encoder network. Likewise, we refer to the decoder network as $p_{\theta}(x|z)$, where θ is the parameters of the decoder network. $p(z)$ refers to a prior distribution that represents the prior beliefs about the latent variable's distribution (the choice of prior will be discussed in much greater detail in the following section).

The starting point for deriving the evidence lower bound (ELBO) is computing the KL divergence between the approximated posterior distribution $q_{\phi}(z|x)$ and the real distribution $p(z|x)$. To compute the KL divergence between any two distributions, we use the following formula (Odaibo, 2019).

$$D_{KL}(q(x)||p(x)) = \int q(x) \log\left(\frac{q(x)}{p(x)}\right) = \int q(x) \log\left(\frac{p(x)}{q(x)}\right)$$

Thus, the distribution between $q_{\phi}(z|x)$ and $p(z|x)$ is

$$D_{KL}(q_{\phi}(z|x)||p(z|x)) = \int q_{\phi}(z|x) \log\left(\frac{p(z|x)}{q_{\phi}(z|x)}\right) dz$$

Using Bayes Rule, we substitute the unknown real posterior distribution $p(z|x)$ with $\frac{p_{\theta}(x|z)p(z)}{p(x)}$

$$\begin{aligned}
D_{KL}(q_{\phi}(z|x)||p(z|x)) &= \int q_{\phi}(z|x) \log\left(\frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)p(x)}\right) dz \\
&= \int q_{\phi}(z|x)\left[\log\left(\frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)}\right) - \log(p(x))\right] dz \\
&= \int q_{\phi}(z|x) \log\left(\frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)}\right) dz + \int q_{\phi}(z|x) \log(p(x)) dz \\
&= \int q_{\phi}(z|x) \log\left(\frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)}\right) dz + \log(p(x)) dz
\end{aligned}$$

Because KL divergence is always non-negative, we know the above expression is greater than or equal to zero. From this inequality, we can derive a lower bound of likelihood.

$$\int q\ (zjx)\log(\frac{p\ (xjz)p(z)}{q\ (zjx)})dz + \log(p(x))dz \quad 0$$

$$\log(p(x))dz \quad \int q\ (zjx)\log(\frac{p\ (xjz)p(z)}{q\ (zjx)})dz$$

$$\log(p(x))dz \quad \int q\ (zjx)\log(\frac{p(z)}{q\ (zjx)})dz + \int q\ (zjx)\log(p\ (xjz))dz$$

$$\log(p(x))dz \quad D_{KL}(q\ (zjx)jjp(z)) + E_{q\ (zjx)}(\log(p\ (xjz)))$$

The right-hand side of the final inequality in the derivation is the ELBO which is used for the VAE loss function.

In addition to representing the lower-bound of the log-likelihood, the ELBO can also be viewed as two loss terms that have semantic meanings in training the VAE model even without considering their connection to probability theory. The KL-divergence term $D_{KL}(q\ (zjx)jjp(z))$ corresponds to the how close (as measured by KL divergence) the approximated posterior distribution is to a prior distribution. The reconstruction loss term $\log(p\ (xjz)$ can be shown to be equivalent to the L2 norm. We drop the expected value as z is sampled from the approximate posterior, allowing us to compute a monte carlo approximation over the distribution. The decoder distribution $p\ (xjz)$ has a constant variance of the identity matrix. The constants c_1 and c_2 can be ignored as they do not they do not impact the performance of the cost function.

$$
\begin{aligned}
\log(p\ (x|z)) &= \log(\not{p}\frac{1}{\sqrt{2}} e^{\ \frac{(x-\ )^2}{2\ ^2}}) \\
&= \log(\not{p}\frac{1}{\sqrt{2}}) \quad \log(\ ) + \log(e^{\ \frac{(x-\ )^2}{2\ ^2}}) \\
&= c_1 \quad \log(\ ) \quad \frac{(x\quad)^2}{2\ ^2} \\
&= c_1 \quad \frac{(x\quad)^2}{2} \\
&= c_1 \quad c_2(x\quad)^2 \\
&\quad (x\quad)^2
\end{aligned}
$$

§ 4.4 VAEs as Autoencoders

To supplement the probability theory shown above, the autoencoder is a helpful analog for understanding the implementation of a VAE. To define the relationship between the high-dimensional inputs and low-dimensional representations, VAEs use an autoencoder-style architecture. An autoencoder encodes an input into a low-dimensional representation and decode the representation to a reconstruction of the input. Both the encoder and the decoder are paramaterized by neural networks.

There are some key differences between the VAE and the autoencoder that are evident from the discussion in 3.3. First, while have defined the VAE's objective function as the ELBO (which happens to include a term that is equivalent to reconstruction loss), the autoencoder is generally trained using an expected reconstruction loss term (Bank et al., 2020). Another key difference between autoencoders and VAEs is that autoencoders map an input $x\ 2\ R^D$ to a vector $z\ 2\ R^d$ while the VAE maps an input x to the approximate posterior, which we paramaterize as a normal distribution with mean and log-variance $;\log(\ )\ 2\ R^d$, from which a latent point is randomly sampled.

§ 4.4.1 The VAE Prior Distribution and VampPrior

The prior distribution $p(z)$ in the KL divergence term is the unit gaussian $N(0; I)$ in most VAE implementations. Setting the prior as the unit gaussians has two distinct advantages. First, it is simple to

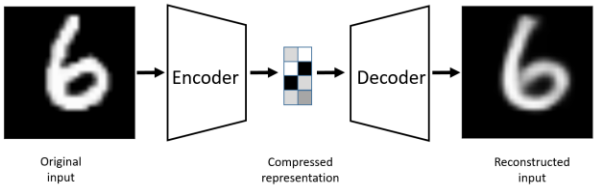

Figure 4.1: Diagram of autoencoders (Source: Bank et. al. 2020)

compute a closed-form KL divergence between two gaussians. Second, by choosing a prior distribution with a convex probability density function (PDF), we are less likely to choose points for which the decoder cannot provide quality reconstructions when interpolating between points in the high-density region of the distribution.

The unit gaussian is not an ideal prior in minimizing the ELBO; the ideal prior in minimizing the ELBO is the aggregated approximate posterior (Tomczak and Welling, 2018). In practice, this prior would cause overfitting (to say nothing of the computational cost of computing KL divergence with this prior), while on the other hand, the unit prior would certainly not overfit the embedding model but would likely overreguarlize it. As an alternative solution, Tomczak and welling propose the VampPrior, in which the prior is a dynamic mixture of gaussian components. For K learned pseudo-inputs $x_1; ...; x_K$, the posterior distribution is given by

$$p = \frac{1}{K} \sum_{k=1}^{K} q(z|x_k)$$

where is the union of the posterior parameters and the additional parameters for learning the pseudo-inputs (Tomczak and Welling, 2018).

CHAPTER 5

METHODOLOGY

We look to discover the potential of applying components of the VAE methodology to DML systems. We test this through presenting incremental modifications to the basic DML architecture. Each modified architecture corresponds to a claim about how a specific part of the VAE training regime and loss function may be adapted to assist the performance of a DML method for a semi-supervised dataset.

Algorithm 1: Base DML Training Routine

Input: Dataset dataset D, encoder network f, metric loss function m, learning rate , weights
Result: updated weights
for *batch* x, y *in* D **do**
  $z = f(x)$;
  $c = m(z; y)$;
  Compute $\frac{dc}{d}$ with backpropogation;
  $=$ $\frac{dc}{d}$;
end

§ 5.1 Claim 1: Benefits of Reconstruction Loss

We first look to evaluate the claim that adding a reconstruction loss to a DML system can improve the quality of clustering in the latent representations on a semi-supervised dataset. Reconstruction loss in and of itself enforces a similar semantic mapping onto the latent space as a metric loss, but can be computed without labelled data. In theory, we believe that the added constraint that the latent vector must be reconstructed to approximate the original output will train the spatial positioning to reflect semantic information. Following this reasoning, observations which share similar semantic information, specifi-

cally observations of the same class (even if not labelled as such), should intuitively be positioned nearby within the latent space. To test if this intuition occurs in practice, we evaluate if a DML model with an autoencoder structure and reconstruction loss (described in further detail below) will perform better than a plain DML model in terms of clustering quality. This will be especially evident for semi-supervised datasets in which the amount of labelled data is not feasible for solely supervised DML.

Given a semi-supervised dataset, we assume a standard DML system will use only the labelled data and train given a metric loss L_{metric} (see Algorithm 1). Our modified model DML Autoencoder will extend the DML model's training regime by adding a decoder network which takes the latent point z as input and produces an output $\hat{x}$. The loss function is then modified such that there is a supervised loss L_S which is identical to the metric loss L_{metric} and an unsupervised loss that is identical to the reconstruction loss L_U. Each epoch, the total loss alternates between the supervised and unsupervised loss, such $L = (1 \quad)L_S + \quad L_U$ on odd number epochs and $L = \quad L_U$ for even number epochs. is a hyperparameter which modulates the impact of the reconstruction loss on total loss for the DML autoencoder. The software tool used, Pytorch Lightning (Falcon and The PyTorch Lightning team, 2019), used to construct the models restricts not using all parameters in the computation of the loss for a given epoch; thus we have a semi-supervised stage consisting of the unsupervised and the supervised loss instead of solely a supervised stage, as the the supervised loss does not make use of the parameters in the decoder.

§ 5.2 Claim 2: Incorporating Inductive Bias with Prior

Say we are aware that a dataset has n classes. It may be useful to encourage that there are n clusters in the latent space of a DML model. This can be enforced by using a prior distribution containing n many Gaussians. As we wish to measure only the affect of inducing bias on the representation without adding any complexity to the model, the prior distribution will not be learnable (unlike VAE with VampPrior). By testing whether the classes of points in the latent space are organized along the prior components we can test whether bias can be induced using a prior to constrain the latent space of a DML. By testing

Algorithm 2: DML Autoencoder Training Routine

Input: Dataset dataset D, encoder network f, decoder network g, metric loss function m, learning rate , coefficient , weights

Result: updated weight

for *batch x, y in D* **do**

$z = f(x)$;

$\hat{x} = g(z)$;

if s **then**

$c = (1 \quad) \; m(z; y) + \quad MSE(\hat{x}; x)$;

else

$c = \quad MSE(\hat{x}; x)$;

end

Compute $\frac{dc}{d}$ with backpropogation;

$= \quad \frac{dc}{d}$;

end

whether clustering improves performance, we can evaluate whether this inductive bias is helpful.

Given a fully supervised dataset, we assume a standard DML system will use only the labelled data and train given a metric loss L_{metric}. Our modified model will extend the DML system's training regime by adding a KL divergence term to the loss which measures the difference between posterior distributions and a prior distribution. It should also be noted that, like the VAE encoder, we will map the input not to a latent point but to a latent distribution. The latent point is stochastically sampled from the latent distribution during training. Mapping the input to a distribution instead of a point will allow us to calculate the KL divergence.

The loss function is then modified such that the total loss L is equal to a weighted sum between the metric loss term L_{metric} and the KL divergence term L_{KL}. As is true in the previous section, the total loss alternates between the supervised and unsupervised loss, such $L = (1 \quad) L_S + \quad L_U$ on odd number epochs and $L = \quad L_U$ for even number epochs.

In practice, we will be evaluating a DML model with a unit prior and a DML model with a mixture of gaussians (GMM) prior. The latter model constructs the prior as a mixture of n gaussians – each the vertice of the unit (i.e. each side is 2 units long) hypercube in the latent space. The logvar of each component is set equal to one. Constructing the prior in this way is beneficial in that it is ensured that each component is evenly spaced within the latent space, but is limiting in that there must be exactly 2^d components in the GMM prior. Thus, to test, we will test a datset with 10 classes on the latent space dimensionality of 4, such that there are $2^4 = 16$ gaussian components in the GMM prior. Though the number of prior components is greater than the number of classes, the latent mapping may still exhibit the pattern of classes forming clusters around the prior components as the extra components may be made redundant.

The drawback of the decision to set the GMM components' means to the coordinates of the unit hypercube's vertices is that the manifold of the chosen dataset may not necessarily exist in 4 dimensions. Choosing gaussian components from a d-dimensional hypersphere in the latent space R^d would solve this issue, but there does not appear to be a solution for choosing n evenly spaced points spanning d dimensions on a d-dimensional hypersphere. KL Divergence is calculated with a monte carlo approximation for the GMM and analytically with the unit prior.

Algorithm 3: DML with Prior Training Routine

Input: Dataset dataset D, encoder network f, metric loss function m, learning rate ,
coefficient , prior distribution mean and log-variance $_p$, $_p$, weights
Result: updated weights
for *batch x, y in D* **do**
  ; = f (x);
  z N(;);
  if s **then**
    c = (1) $m(z; y)$ + $KL(z;$; ; $_p$; $_p$);
  else
    c = $KL(z;$; ; $_p$; $_p$);
  end
  Compute $\frac{dc}{d}$ with backpropogation;
  = $\frac{dc}{d}$;
end

Algorithm 4: Monte-Carlo KL Divergence Algorithm

Input: Latent variable z, approximate posterior distribution mean and variance , , prior distribution mean and log-variance $_p$, $_p$

Result: KL Divergence between distributions q and p

$P(z j \; ; \;) = \quad 0{:}5 \quad \frac{(z \quad)^2}{\exp}$;

$Q(z j \; _p; \; _p) = \quad 0{:}5 \quad \frac{(z \quad _p)^2}{\exp \; _p}$;

return $Q(z j \; _p; \; _p) \quad P(z j \; ; \;)$

§ 5:3 Claim 3: Jointly Optimizing DML with VAE

The third claim we look to evaluate is that given a semi-supervised dataset, optimizing a DML model jointly with a VAE on the VAE's latent space will produce superior clustering than the DML model individually. The intuition behind this approach is that DML methods can learn from only supervised data and VAE methods can learn from only unsupervised data; the proposed methodology will optimize both tasks simultaneously to learn from both supervised and unsupervised data.

The MetricVAE implementation we create jointly optimizes the VAE task and DML task on the VAE latent space. Across epochs, the MetricVAE model alternates between training only the unsupervised task L_U and the semi-supervised task $L_U + (1 \quad) \quad L_S$, like each of the other modified DML models. The actual implementation belies the pseudocode algorithm slightly as it uses the VAE with VampPrior model instead of the vanilla VAE.

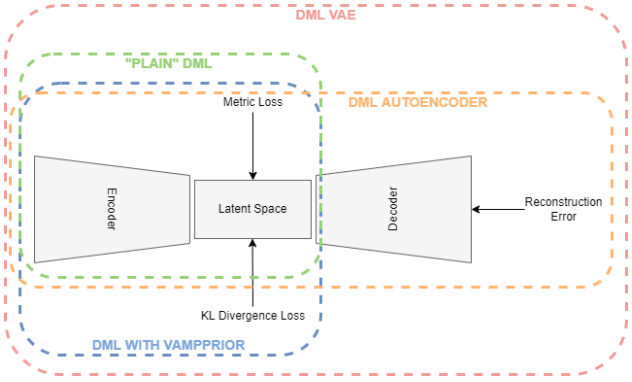

Figure 5.1: Comparison of Modified DML Architectures

Algorithm 5: DML VAE Training Routine

Input: Dataset dataset D, encoder network f, decoder network g, metric loss function m, learning rate , coefficient , coefficient , prior distribution mean and log-variance $_p$, $_p$, weights

Result: updated weights

for *batch* x, y *in* D **do**

  ; = $f(x)$;

  z $N(\ ;\ )$;

  $\hat{x} = g(z)$;

  $c_{vae} = MSE(x; x_{hat}) +$ $KL(z;\ ;\ ;\ _p$;

  if s **then**

    $c = (1\quad) \quad m(z; y) +$ c_{vae};

  else

    $c =$ c_{vae};

  end

  Compute $\frac{dc}{d}$ with backpropogation;

  = $\frac{dc}{d}$;

end

CHAPTER 6

RESULTS

§ 6.1 General Experimental Configuration

Each set of experiments shares a similar hyperparameter search space. Below we describe the hyperparameters that are included in the search space of each experiment. We also discuss the hardware used and the the evaluation method.

§ 6.1.1 Learning Rate (lr)

Through informal experimentation, we have found that the learning rate of 0.001 causes the models to converge consistently. The learning rate is thus set to 0.001 in each experiment.

§ 6.1.2 Latent Space Dimensionality (lsdim)

Latent space dimensionality refers to the dimensionality of the vector output of the encoder of a DML network or the dimensionality of the posterior distribution of a VAE (also the dimensionality of the latent space). When the latent space dimensionality is 2, we see the added benefit of creating plots of the latent representations (though we can accomplish this through using dimensionality reduction methods like tSNE for higher dimensionalities as well). Example values for this hyperparameter used in experiments are 2, 4, and 10.

§ 6.1.3 Alpha

Alpha ( ) is a hyperapameter which refers to the balance between the unsupervised and supervised losses of some of the modified DML models. More details about the role of in the model implementations are discussed in the methodology section of the model. Potential values for alpha are each between 0 (exclusive) and 1 (inclusive). We do not include 0 in this set as if is set to 0, the model is equivalent to the fully supervised plain DML model because the supervised loss would not be included. If is set to 1, then the model would train on only the unsupervised loss; for instance if the DML Autoencoder had set to 1, then the model would be equivalent to an autoencoder.

§ 6.1.4 Partial Labels Percentage (pl%)

The partial labels percentage hyperparameter refers to the percentage of the dataset that is labelled and thus the size of the partion of the dataset that can be used for labelled training. Of course, each of the datasets we use is fully labelled, so a partially labelled datset can be trivially constructed by ignoring some of the labels. As the sizes of the dataset vary, each percentage can refer to a different number of labelled samples. Values for the partial label percentage we use across experiments include 0.01, 0.1, 10, and 100 (with each value referring to the percentage).

§ 6.1.5 Datasets

Two datasets are used for evaluating the models. The first dataset is MNIST (LeCun and Cortes, 2010), a very popular dataset in machine learning containing greyscale images of handwritten digits. The second dataset we use is the organ OrganAMNIST dataset from MedMNIST v2 (Yang et al., 2021). This dataset contains 2D slices from computed tomography images from the Liver Tumor Segmentation Benchmark – the labels correspond to the classification of 11 different body organs. The decision to use a second dataset was motivated because the as the claims are tested over more datasets, the results supporting the claims become more generalizable. The decision to use the OrganAMNIST dataset specifically is motivated in part due to the the Quinn Research Group working on similar tasks for biomedical imaging (Zain et al., 2020). It is also motivated in part because OrganAMNIST is a more difficult dataset, at least for a the

classfication task, as the leading accuracy for MNIST is .9991 (An et al., 2020) while the leading accuracy for OrganAMNIST is .951 (Yang et al., 2021). The MNIST and OrganAMNIST datasets are similar in dimensionality (1 x 28 x 28), number of samples (60,000 and 58,850, respectively) and in that they are both greyscale.

§ 6.1.6 Hardware

Every experiment discussed was run on the Quinn Research Group's Rocinante server, which contains 4 NVIDIA GeForce RTX 2080 Ti GPUs. Using the Weights and Biases sweep API, we parallelize the experiments such that four experiments run simletaneously on one GPU each.

Figure 6.1: Sample images from the MNIST (left) and OrganAMNIST of MedMNIST (right) datasets

§ 6.1.7 Evaluation

We will evaluate the results by running each model on a test partition of data. We then take the latent points Z generated by the model and the corresponding labels Y. Three classifiers (sklearn's implementation of RandomForest, MLP, and kNN) each output predicted labels $\hat{Y}$ for the latent points. In most of the charts shown, however, we only include the kNN classification output due to space constraints and the lack of meaningful difference between the output for each classifier. We finally measure the quality of the predicted labels $\hat{Y}$ using the Adjusted Mutual Information Score (AMI) (Vinh et al., 2010) and accuracy (which is still helpful but is also easier to interpret in some cases). This scoring metric is common in research that looks to evaluate clustering performance (Zhu and Gao, 2021) (Emmons et al., 2016). We

will be using sklearn's implementation of AMI (Pedregosa et al., 2011). The performance of a classifier on the latent points intuitively can be used as a measure of quality of clustering. Each result shown in Table 6.1–6.2 is the average of four identical experiments performed on the same hyperparameter configuration. The results in Table 6.3–6.5 are randomly sampled on four identical experiments performed on the same hyperparameter configuration, and each result in Table 6.6 corresponds to one experiment.

§ 6.2 Claim 1: Benefits of Reconstruction Loss

In evaluating the first claim, we compare the performance of the plain DML model to the DML Autoencoder model. We do so by comparing the performance of the plain DML system and the DML Autoencoder across a search space containing the lsdim, alpha, and pl% hyperparameters and both datasets.

In table 6.1, we observe that for relatively small amounts of labelled samples (the partial labels percentages of 0.01 and 0.1 correspond to 6 and 60 labelled samples respectively), the DML Autoencoder severely outperforms the DML model. However, when the number of labelled samples increases (the partial labels percentage of 10 correspond to 6000 labelled samples respectively), the DML model significantly outperforms the DML Autoencoder. This trend is not too surprising, as when there is sufficient data to train unsupervised methods and insufficient data to train supervised method, as is the case for the 0.01 and 0.1 partial label percentages, the unsupervised method will likely perform better.

It is somewhat surprising is that the injection of labelled data does not appear to definitively improve the performance of the DML Autoencoder, as evidenced by the comparing $= 1$ to other values of < 1 that incorporate metric loss into the overall loss and as evidenced by the increasing percentage of labelled data having little effect on performance.

In observing the other hyperparameters, we see that for almost every model configuration, setting the latent space dimensionality to 10 dimensions (as opposed to 2 dimensions) causes a significant increase in performance. This is not surprising as the representational power of the latent space grows with its dimensionality. It is, however, difficult to assess the role of alpha in the DML Autoencoder and plain

Table 6.1: Comparison of the DML (left) and DML Autoencoder (right) models for the MNIST dataset. Bolded values indicate best performance for each partial labels percentage partition (pl%).

pl%	lsdim	knn acc	knn MI
0.01	2	**0.2618**	**0.1124**
	10	0.2610	0.1073
0.1	2	0.4256	0.2904
	10	**0.6556**	**0.4983**
10	2	0.7476	0.6200
	10	**0.9502**	**0.8838**

pl%	alpha	lsdim	knn acc	knn MI
0.01	0.25	2	0.3957	0.2815
		10	**0.8351**	**0.7172**
	0.5	2	0.3358	0.1932
		10	0.6688	0.5218
	0.75	2	0.3112	0.1835
		10	0.8323	0.7102
	1	2	0.4166	0.2923
		10	0.8319	0.7089
0.1	0.25	2	0.4349	0.3339
		10	0.8718	0.7616
	0.5	2	0.4033	0.3150
		10	**0.8798**	**0.7725**
	0.75	2	0.3812	0.2771
		10	0.7820	0.6399
	1	2	0.4202	0.3022
		10	0.8589	0.7463
10	0.25	2	0.1028	0
		10	0.8478	0.7258
	0.5	2	0.3465	0.2189
		10	0.7925	0.6606
	0.75	2	0.3536	0.2175
		10	0.8497	0.7373
	1	2	0.5137	0.3793
		10	**0.8570**	**0.7407**

DML models' performance. Changes in alpha can affect model performance very marginally or very significantly; it is difficult to find a pattern to infer how changing alpha affects model performance.

The data looks to show that the claim that adding a reconstruction loss to a DML system can improve the quality of clustering in the latent representations on a semi-supervised dataset when there are small amounts (roughly less than 100 samples) of labelled data *and* a sufficient quantity of unlabelled data. But an important caveat is that it is not convincing that the DML Autoencoder effectively combined the unsupervised and supervised losses to create a superior model, as a plain autoencoder (i.e. the DML Autoencoder with $= 1$) outperforms the DML for the partial labels percentage of or less than 0.1% and underperforms the DML for the partial labels percentage of 10%.

In the table for the MedMNIST dataset, we see very similar trends of the performance increasing greatly with increasing the latent space dimensionality, the DML performance improving greatly while the DML Autoencoder performance improves only somewhat with the amount of labelled data, and the alpha of the DML Autoencoder having a relatively small impact. Overall, it does not appear that medMNIST is a drastically more difficult dataset with regards to model performance (it is somewhat confounding why the DML performance on $pl\% = 0.01$ is much better on this dataset than on MNIST). The results on the MedMNIST dataset thus strengthen the above evaluation of claim 1.

§ 6.3 Claim 2: Incorporating Inductive Bias with Prior

In evaluating the second claim, we compare the performance of the plain DML model to the DML with a unit prior and a DML with a GMM prior. The DML prior with the GMM prior will have $2^2 = 4$ gaussian components when $lsdim = 2$ and $2^4 = 16$ components when $lsdim = 4$.

Our broad intention is to see if changing the shape (specifically the number of components) of the prior can induce bias by affecting the pattern of embeddings. We hypothesize that when the GMM prior contains n components and n is slightly greater than or equal to the number of classes, each class will cluster

Table 6.2: Comparison of the DML (left) and DML Autoencoder (right) models for the OrganAMNIST dataset. Bolded values indicate best performance for each partial labels percentage partition (pl%).

pl%	lsdim	knn acc	knn MI
0.01	2	0.3844	0.2637
	10	**0.6997**	**0.5529**
0.1	2	0.4323	0.3165
	10	**0.8309**	**0.7129**
10	2	0.4616	0.3614
	10	**0.8487**	**0.7339**

pl%	alpha	lsdim	knn acc	knn MI
0.01	0.25	2	0.3683	0.2685
		10	0.8060	0.6792
	0.5	2	0.3388	0.2146
		10	**0.8378**	**0.7207**
	0.75	2	0.3419	0.2183
		10	0.7568	0.5987
	1	2	0.3844	0.2637
		10	0.6997	0.5529
0.1	0.25	2	0.3717	0.2496
		10	0.8514	0.7381
	0.5	2	0.4314	0.2918
		10	0.8553	0.7419
	0.75	2	0.3896	0.2502
		10	**0.8597**	**0.7454**
	1	2	0.4323	0.3165
		10	0.8309	0.7129
10	0.25	2	0.4316	0.3354
		10	0.8560	0.7394
	0.5	2	0.3824	0.2404
		10	**0.8692**	**0.7589**
	0.75	2	0.5118	0.3891
		10	0.8250	0.7078
	1	2	0.4616	0.3614
		10	0.8487	0.7339

Table 6.3: Comparison of the DML model (left) and the DML with prior models with a unit gaussian prior (center) and GMM prior (right) models for the MNIST dataset.

pl%	lsdim	knn acc	knn MI
0.01	2	0.2603	0.1163
	4	0.4460	0.2944
0.1	2	0.4247	0.2916
	4	0.4277	0.3133
10	2	0.7450	0.6156
	4	0.9133	0.8167

pl%	alpha	lsdim	knn acc	knn MI
0.01	0.33	2	0.0989	0.0011
		4	0.09748	-0.0005
	0.66	2	0.0947	0.0008
		4	0.1082	0.001
	1	2	0.1064	0.0001
		4	0.1061	-0.0006
0.10	0.33	2	0.0965	0.0002
		4	0.1049	-0.0017
	0.66	2	0.0941	0.0008
		4	0.1073	0.0001
	1	2	0.1106	-0.001
		4	0.0941	0.0012
10	0.33	2	0.0998	-0.0008
		4	0.0944	-0.0007
	0.66	2	0.1058	-0.0009
		4	0.0941	0.0017
	1	2	0.0932	0
		4	0.1073	-0.0001

knn acc	knn MI
0.1049	0
0.0974	0.0022
0.0989	-0.0009
0.1025	0.0012
0.0968	0.0007
0.1109	0.0010
0.1055	-0.0011
0.0962	0.0019
0.0974	0.0001
0.1139	0
0.1037	0.0005
0.1082	0
0.0872	0.0008
0.0908	0.0003
0.1001	0.0013
0.0968	0.0006
0.0974	-0.0006
0.1058	- 0.0014

around one of the prior components. We will test this for the GMM prior with 16 components ($lsdim = 4$) as both the MNIST and MedMNIST datasets have 10 classes. We are unable to set the number of GMM components to 10 as our GMM sampling method only allows for the number of components to equal a power of 2 (see section 5.2). Our baseline models include a plain DML and a DML with a unit prior (the distribution $N(0; 1)$).

In the data, it is very evident that across both datasets, the DML models with any prior distribution all devolve to the null model (i.e. the classifier is no better than random selection). From the visualilzations of the latent embeddings, we see that the embedded data for the DML models with priors appears completely random (figure 6.2). In the case of the GMM prior, it also does not appear to take on the shape of the prior or reflect the number of components in the prior. This may be due to the training routine of the DML models. As the KL divergence loss, which can be said to "fit" the embeddings to the prior, trains

Table 6.4: Comparison of the DML model (left) and the DML with prior models with a unit gaussian prior (center) and GMM prior (right) models for the OrganAMNIST dataset.

pl%	lsdim	knn acc	knn MI	pl%	alpha	lsdim	knn acc	knn MI	knn acc	knn MI
0.01	2	0.3941	0.2651	0.01	0.33	2	0.0962	-0.0008	0.1016	-0.0009
						4	0.0905	0.0005	0.0983	0.0005
					0.66	2	0.1007	0	0.1043	-0.0006
	4	0.3791	0.2262			4	0.0998	0	0.1004	-0.0002
					1	2	0.1061	-0.0006	0.0935	0.0017
						4	0.1088	0.0013	0.1079	-0.0002
0.10	2	0.4286	0.3135	0.10	0.33	2	0.1064	0.0002	0.1079	-0.0005
						4	0.1031	0	0.1019	-0.0001
					0.66	2	0.1061	-0.0011	0.1085	0.0012
	4	0.2723	0.1519			4	0.1016	-0.0006	0.1049	-0.0009
					1	2	0.0959	-0.0005	0.0971	0.0007
						4	0.1058	-0.0005	0.0974	0.0006
10	2	0.4625	0.3653	10	0.33	2	0.0950	-0.003	0.1052	0.0005
						4	0.1034	0	0.0971	-0.0004
					0.66	2	0.1043	0	0.0965	0.0009
	4	0.9319	0.8490			4	0.1088	0.0007	0.09628	-0.0001
					1	2	0.0995	-0.0013	0.1112	0
						4	0.1055	0.0010	0.1073	0.0012

on alternating epochs with the supervised DML loss, it is possible that the two losses are not balanced correctly during the training process. From the discussed results, it is fair to state that adding a prior distribution to a DML model through training the model on the KL divergence between the prior and approximated posterior distributions on alternating epochs does is not an effective way to induce bias in the latent space.

§ 6.4 Claim 3: Jointly Optimizing DML with VAE

To evaluate the third claim, we compare the performance of DMLs to MetricVAEs (defined in the previous chapter) across several metric losses. We run experiments for triplet loss, supervised loss, and center loss DML and MetricVAE models. To judge whether the claim is true, we will assess whether the model performance improves for the MetricVAE over the DML for the same metric loss and other hyper parameters.

Table 6.5: Comparison of KNN Accuracy for DML and MetricVAE (MVAE) across metric losses for the MNIST dataset. The DML models do not take any alpha value, so their results are consistent across pl% and lsdim. The pl% value of 0.001 is not included as the supervised loss receives a NaN error.

				Triplet		Supervised		Center	
dataset	**pl%**	**lsdim**	**alpha**	**DML**	**MVAE**	**DML**	**MVAE**	**DML**	**MVAE**
MNIST	0.01	2	0.25	0.2603	0.1025	0.3167	0.1052	0.1889	0.1001
	0.01	2	0.50		0.1049		0.2699		0.1034
	0.01	2	0.75		0.1004		0.1055		0.1064
	0.01	2	1.00		0.1115		0.1022		0.1004
	0.01	10	0.25	0.2678	0.1028	0.5281	0.0947	0.5509	0.1064
	0.01	10	0.50		0.0923		0.2630		0.0995
	0.01	10	0.75		0.1037		0.1040		0.1076
	0.01	10	1.00		0.1067		0.1040		0.0932
	10	2	0.25	0.4625	0.1109	0.1103	0.1028	0.1682	0.0992
	10	2	0.50		0.0992		0.1073		0.10467
	10	2	0.75		0.1031		0.1022		0.1001
	10	2	1.00		0.0890		0.0920		0.1010
	10	10	0.25	0.8395	0.1025	0.9370	0.1010	0.46430	0.0986
	10	10	0.50		0.1025		0.0932		0.1073
	10	10	0.75		0.1058		0.1043		0.0947
	10	10	1.00		0.0950		0.1076		0.1013
OrganAMNIST	0.01	2	0.25	0.3941	0.1061	0.3608	0.0911	0.1967	0.1061
	0.01	2	0.50		0.0965		0.1019		0.1010
	0.01	2	0.75		0.0929		0.1076		0.1097
	0.01	2	1.00		0.1010		0.1055		0.1040
	0.01	10	0.25	0.6997	0.1175	0.4868	0.1058	0.5422	0.1121
	0.01	10	0.50		0.0929		0.1094		0.0989
	0.01	10	0.75		0.1091		0.1112		0.1040
	0.01	10	1.00		0.1091		0.1007		0.0911
	10	2	0.25	0.4625	0.0968	0.6694	0.1091	0.2228	0.1061
	10	2	0.50		0.0995		0.1067		0.0965
	10	2	0.75		0.1061		0.0959		0.1613
	10	2	1.00		0.1043		0.1679		0.1007
	10	10	0.25	0.8395	0.0980	0.9694	0.1091	0.5932	0.1094
	10	10	0.50		0.0980		0.1238		0.1019
	10	10	0.75		0.0974		0.1346		0.1148
	10	10	1.00		0.1025		0.1076		0.1088

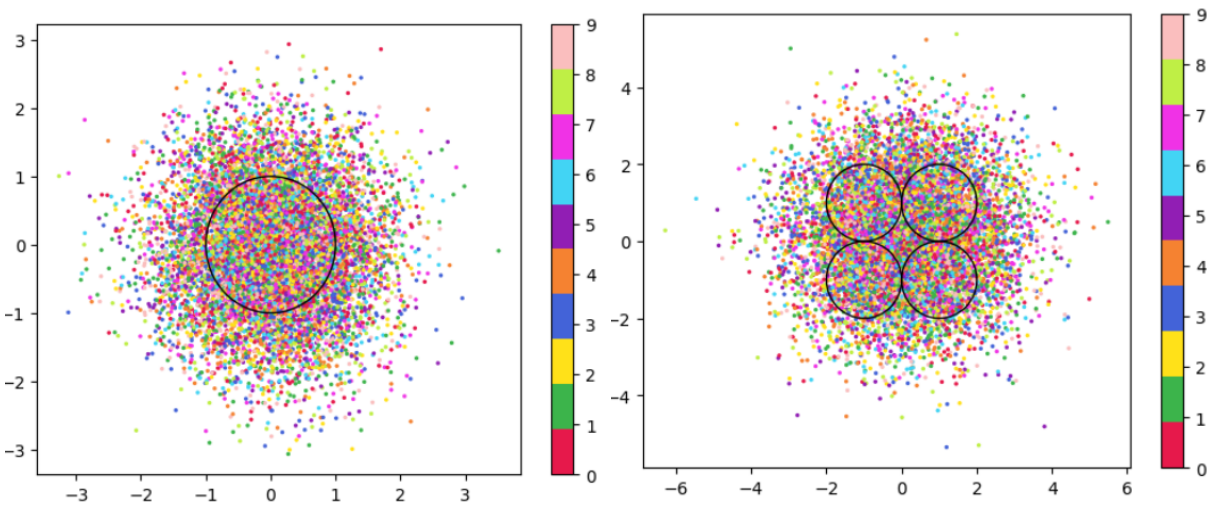

Figure 6.2: Comparison of latent spaces for DML with unit prior (left) and DML with GMM prior containing 4 components (right) for $lsdim = 2$ on OrganAMNIST dataset. The gaussian components are shown as black with the raidus equal to variance (1). There appears to be no evidence of the distinct gaussian components in the latent space on the right. It does appear that the unit prior may regularize the magnitude of the latent vectors.

From table 6.5, we see that the MVAE for each loss only occasionally trains to perform better than the null model. The corresponding DML model almost always performs much better. This is true across all percentages of labelled data, latent space dimensionality, and alpha values. As with claim 2, it is possible this is because the training routine of alternating between supervised loss (in this case, metric loss) and unsupervised (in this case, VAE loss) is not optimal for training the model. Further supporting this point, previously we have received the following results for testing a version of the MVAE model that is trained against both the supervised and unsupervised loss on each epoch, as shown in table 6.6. In these results, we see clearly that an alpha value of over zero (i.e. incorporating both the supervised metric loss into the overall MVAE loss function) can help improve performance especially among lower dimensionalities.

Given our analysis of the data, we see that incorporating the DML loss to the VAE is potentially helpful, but only when training the unsupervised and supervised losses jointly. Even in that case, it is unclear whether the MVAE performs better than the corresponding DML model even if it does perform better than the corresponding VAE model. Alternating between training the model on the unsupervised

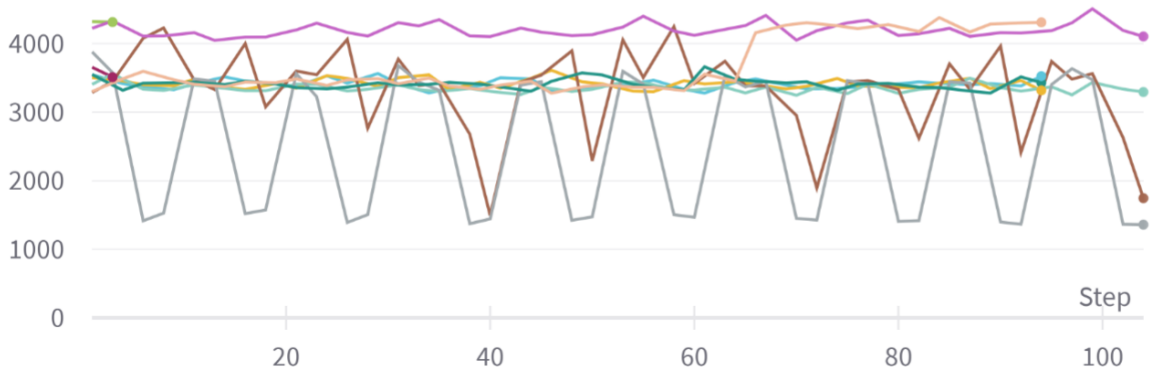

Figure 6.3: Graph of reconstruction loss (componenet of unsupervised loss) of MVAE across epochs. The unsupervised loss does not converge despite being trained on each epoch.

and supervised objectives is does not appear to be a viable training routine. From the data and our analysis, we are unable to confirm the claim that optimizing a DML model jointly with a VAE on the VAE's latent space will produce superior clustering than the DML model individually.

Table 6.6: Experiments performed on MVAE architecture across fully labelled MNIST dataset that trains on objective function $L = L_U + \quad L_S$ on fully supervised dataset. The best results for the classification accuracy on the MVAE embeddings in a given latent-dimensionality are bolded.

Architecture Parameters		**MVAE Clasif. Accuracy (%)**			**DML Clasif. Accuracy (%)**		
lsdim	**gamma**	**linear**	**rf**	**kNN**	**linear**	**rf**	**kNN**
2	0	57.8	65.2	64.6	95.0	94.5	95.0
	2	**75.0**	69.8	70.6			
	5	68.8	67.9	66.9			
	10	51.6	**75.4**	**74.2**			
5	0	81.3	83.9	84.5	98.1	97.9	97.9
	2	85.9	88.5	88.4			
	5	82.8	88.0	88.5			
	10	**89.1**	**92.0**	**92.4**			
10	0	85.9	92.1	92.7	98.5	97.7	98.0
	2	87.4	91.8	91.3			
	5	**98.4**	**92.8**	**93.4**			
	10	93.8	92.5	91.9			

CHAPTER 7

CONCLUSION AND FUTURE WORK

§ 7.1 Conclusion

In this work, we have set out to determine how DML can be extended for semi-supervised datasets by borrowing components of the variational autoencoder. We have formalized this approach through defining three specific claims. To evaluate each claim, we have created several variations of the DML model, such as the DML Autoencoder, DML with Unit/GMM Prior, and MVAE. We then tested the performance of the models across several semi-supervised partitions of two datasets, along with other configurations of hyperparameters.

We have determined from the analysis of our results, there is too much dissenting data to clearly accept any three of the claims. For claim 1, while the DML Autoencoder outperforms the DML for semi-supervised datasets with small amounts of labelled data, it's peformance is not consistently much better than that of a plain autoencoder which uses no labelled data. For claim 2, each of the DML models with an added prior performed extremely poorly, near or at the level of the null model. For claim 3, we see the same extremely poor performance from the MVAE models.

§ 7.2 Future Work

In the future, it would be worthwhile to evaluate these claims using a different training routine. We have stated previously that perhaps the extremely poor performance of the DML with a prior and MVAE mod-

els may be due to the training regimen of alternating on training against a supervised and unsupervised loss. Further research could look to develop or compare several different training regimens. One alternative would simply be to keep alternating between losses but at the level of each batch instead of each epoch. Another alternative, specifically for the MVAE, may be first training DML on labelled data, training a GMM on it's outputs, and then using the GMM as the prior distribution for the VAE. Grosnit et al. (2021) has defined a more complex training routines to balance the DML and unsupervised loss. If this line of research is pursued, it may be worthwhile to review the field of auxiliary task learning, in which a model trains against an additional task or tasks, to find a solution to how to optimize the training routine of the modified DML models.

Another potentially interesting avenue for future study is in investigating a fourth claim for a possible benefit to combining DML and VAE methodology: the ability to define a Riemannian metric on the latent space. Previous research has shown a Riemannian metric can be computed on the latent space of the VAE by computing the pull-back metric of the VAE's decoder function (Arvanitidis et al., 2020). Through the Riemannian metric we could calculate metric losses such as triplet loss with a geodesic instead of euclidean distance. The geodesic distance may be a more accurate representation of similarity in the latent space than euclidean distance as it accounts for the structure of the input data.

BIBLIOGRAPHY

An, S., Lee, M. J., Park, S., Yang, H., & So, J. (2020). An ensemble of simple convolutional neural network models for MNIST digit recognition. *CoRR*, *abs/2008.10400*. https://arxiv.org/abs/2008.10400

Andresini, G., Appice, A., & Malerba, D. (2021). Autoencoder-based deep metric learning for network intrusion detection. *Information Sciences*, *569*, 706–727. https://doi.org/https://doi.org/10.1016/j.ins.2021.05.016

Arvanitidis, G., Hauberg, S., & Schölkopf, B. (2020). Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*.

Baghshah, M. S., & Shouraki, S. B. (2009). Semi-supervised metric learning using pairwise constraints. *Twenty-First International Joint Conference on Artificial Intelligence*.

Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. *arXiv preprint arXiv:2003.05991*.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bilenko, M., Basu, S., & Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. *Proceedings of the twenty-first international conference on Machine learning*, 11.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 539–546.

Dahmani, S., Colotte, V., Girard, V., & Ouni, S. (2019). Conditional variational auto-encoder for text-driven expressive audiovisual speech synthesis. *INTERSPEECH 2019-20th Annual Conference of the International Speech Communication Association*.

Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4690–4699.

Dutta, U. K., Harandi, M., & Sekhar, C. C. (2021). Semi-supervised metric learning: A deep resurrection.

Emmons, S., Kobourov, S., Gallant, M., & Börner, K. (2016). Analysis of network clustering algorithms and cluster quality metrics at scale. *PloS one*, *11*(7), e0159161.

Falcon, W., & The PyTorch Lightning team. (2019). *PyTorch Lightning* (Version 1.4). https://doi.org/10.5281/zenodo.3828935

Grosnit, A., Tutunov, R., Maraval, A. M., Griffiths, R.-R., Cowen-Rivers, A. I., Yang, L., Zhu, L., Lyu, W., Chen, Z., Wang, J., et al. (2021). High-dimensional bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609*.

Joy, T., Schmon, S., Torr, P., Siddharth, N., & Rainforth, T. (2020). Capturing label characteristics in vaes. *International Conference on Learning Representations*.

Kha Vu, C. (2021). *Deep metric learning: A (long) survey*. https://hav4ik.github.io/articles/deep-metric-learning-survey

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., & Krishnan, D. (2020). Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes.

Kulkarni, A., Colotte, V., & Jouvet, D. (2020). Deep variational metric learning for transfer of expressivity in multispeaker text to speech. *International Conference on Statistical Language and Speech Processing*, 157–168.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. http://yann.lecun.com/exdb/mnist/

Lin, X., Duan, Y., Dong, Q., Lu, J., & Zhou, J. (2018). Deep variational metric learning. *Proceedings of the European Conference on Computer Vision (ECCV)*, 689–704.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 212–220.

Odaibo, S. (2019). Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv:1907.08956*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 815–823.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 1857–1865.

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, *28*, 3483–3491.

Tomczak, J., & Welling, M. (2018). Vae with a vampprior. *International Conference on Artificial Intelligence and Statistics*, 1214–1223.

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, *11*, 2837–2854.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., & Liu, W. (2018). Cosface: Large margin cosine loss for deep face recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5265–5274.

Wang, Q., Yuen, P. C., & Feng, G. (2013). Semi-supervised metric learning via topology preserving multiple semi-supervised assumptions. *Pattern Recognition*, *46*(9), 2576–2587.

Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – eccv 2016* (pp. 499–515). Springer International Publishing.

Weng, L. (2021). Contrastive representation learning. *lilianweng.github.io/lil-log*. https://lilianweng.github.io/lil-log/2021/05/31/contrastive-representation-learning.html

Wu, S., Feng, X., & Zhou, F. (2020). Metric learning by similarity network for deep semi-supervised learning. *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*, 995–1002.

Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., & Ni, B. (2021). Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*.

Zain, M., Rao, S., Safir, N., Wyner, Q., Humphrey, I., Eldridge, A., Li, C., AlAila, B., & Quinn, S. P. (2020). Towards an unsupervised spatiotemporal representation of cilia video using a modular generative pipeline.

Zhang, X., Zhao, R., Qiao, Y., Wang, X., & Li, H. (2019). Adacos: Adaptively scaling cosine logits for effectively learning deep face representations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10823–10832.

Zhu, Z., & Gao, Y. (2021). Finding cross-border collaborative centres in biopharma patent networks: A clustering comparison approach based on adjusted mutual information. *International Conference on Complex Networks and Their Applications*, 62–72.