2024

Contrastive Learning, with Application to Forensic Identification of Source

Cole Ryan Patten
South Dakota State University, colepatten@outlook.com

CONTRASTIVE LEARNING, WITH APPLICATION TO FORENSIC IDENTIFICATION OF SOURCE

BY

COLE RYAN PATTEN

A thesis submitted in partial fulfilment of the requirements for the

Master of Science

Major in Mathematics

Specialization in Statistics South Dakota State University

2024

THESIS ACCEPTANCE PAGE
Cole Patten

This thesis is approved as a creditable and independent investigation by a candidate for the master's degree and is acceptable for meeting the thesis requirements for this degree. Acceptance of this does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department.

Michael Puthawala
Advisor Date

Kurt Cogswell
Department Head Date

Nicole Lounsbery, PhD
Director, Graduate School Date

"Thinking is learning all over again to see, to be attentive, to focus consciousness; it is turning every idea and every image, in the manner of Proust, into a privileged moment."

–Albert Camus

"I want to learn more and more to see as beautiful what is necessary in things; then I shall be one of those who make things beautiful. Amor fati: let that be my love henceforth! I do not want to wage war against what is ugly. I do not want to accuse; I do not even want to accuse those who accuse. Looking away shall be my only negation. And all in all and on the whole: some day I wish to be only a Yes-sayer!"

–Friedrich Nietzsche

ACKNOWLEDGEMENTS

There is no way to issue enough credit to my advisor, Dr Michael Puthawala, for his guidance in completing my thesis. Under his supervision, I have been afforded an opportunity to research deep learning in an environment rich with knowledge and fostering to intellectual growth.

Encouragement from those close to me, especially my family and girlfriend, has been invaluable in the course of my master's degree. At times, their confidence in myself out measured my own and kept me committed to improving upon myself and my work.

Thank you to my friend, Amanda Seykora; father, Chane Patten; mother, Dawn Patten; and girlfriend, Faith Cox for their help editing this manuscript.

Thank you to Dr Kurt Cogswell and Dr Christopher Saunders, both wonderful mentors and friends. They have served a pivotal role in my education, encouraging me to attend SDSU for my bachelor's degree and afterward to stay in pursuit of my master's.

I have many friends who have enabled me to continually improve as a person. In specific, Jack Rabern, Nathan Tuttle, Carson Peery, and Payton Kleinsasser, my fellow co-founders of the Rabbit Readers Book Club at SDSU. Also, Kyle Greywall helped me realize how fun mathematics can be.

I would like to acknowledge the support I received from the National Science Foundation's (NSF) National Research Traineeship (NRT) grant number: 1828492.

It would not be possible to enumerate all of the people who share responsibility in making me the person I am today. To those not explicitly mentioned here, I recognize and appreciate your support.

Finally, the greatest thanks to my beautiful kitten, Stevie.

CONTENTS

ABSTRACT

CONTRASTIVE LEARNING, WITH APPLICATION TO FORENSIC IDENTIFICATION OF SOURCE

COLE RYAN PATTEN

2024

Forensic identification of source problems often fall under the category of verification problems, where recent advances in deep learning have been made by contrastive learning methods. Many forensic identification of source problems deal with a scarcity of data, an issue addressed by few-shot learning. In this work, we make specific what makes a neural network a contrastive network. We then consider the use of contrastive neural networks for few-shot learning classification problems and compare them to other statistical and deep learning methods. Our findings indicate similar performance between models trained by contrastive loss and models trained by cross-entropy loss. We also perform an ablation study to investigate the effects of different contrastive loss functions, metric functions, and margin values within contrastive learning. To test contrastive networks on real forensic data, we use the NBIDE cartridge casing dataset. Results are promising, as contrastive learning competed with older statistical methods while taking significantly less data preprocessing. Finally, we detail the desired invariance properties of embedding functions learned by contrastive networks in hopes that future work can enforce them through model architecture.

1 INTRODUCTION

Deep learning, the use of multi-layer neural networks to obtain meaningful representations of data, has made substantial advances within the last decade. The rate of progress has been so great that it has been scarcely possible for specialized fields of application to keep up with the latest technologies in deep learning. Because of this, time must often be taken to consider the best ways to implement deep learning in specific fields. Forensic science, defined as "the use of scientific methods or expertise to investigate crimes or examine evidence that might be presented in a court of law" [18], is one such field. The importance of accurately answering questions such as "Was this cartridge found at the crime scene fired from the suspect's firearm?" and "Were cartridge A and cartridge B fired from the same firearm?" cannot be understated.

Contrastive neural networks (contrastive networks) are a deep learning technique that was first proposed in Bromley et al. [2] (then called siamese networks) as a way to utilize deep learning techniques to quantify the similarity between two objects. The fact that contrastive networks learned similarities between objects allowed them to attend to a broader class of problems than extant deep learning techniques. Today, contrastive networks are among the state-of-the-art for computer vision tasks [20].

Contrastive networks operate by taking multiple objects as inputs and calculating some metric between the embeddings of the objects. For example, when trying to identify which firearm a cartridge casing was fired from, if the objects are expended cartridge casings, a contrastive network learns an embedding function where casings fired from different firearms are embedded further from each other than casings fired from the same firearm according to the metric.

The goal of this work is to address both the subject of contrastive networks in themselves and their application to forensic science. Through modern deep learning techniques, this work strives to embrace the scientific virtues of thoughtfulness and rigor

in the investigation of forensic evidence.

1.1 PRELIMINARY KNOWLEDGE

We will begin with some definitions.

When humans look at images of cats or dogs, we can tell the difference between the two. Unfortunately, the task is not such a simple one for a computer. Consider the task of programming a computer to distinguish between 200*200 pixel black-and-white images of cats and dogs.

Definition 1 (Object, Data space)**.**

An object, x, is an element of some space, $\mathcal{X}$, called the data space. A sample $S \subseteq \mathcal{Z}$ is defined to be a list of objects.

In our example, the data space $\mathcal{X} = [0, 255]^{200*200}$ and any object $x \in \mathcal{X}$ can be represented as an image. Only some of these images will be of cats and dogs, others may be horses or random static. Our sample S will only contain images of cats and dogs.

This manuscript always refers to objects as x_* and differentiates them with subscripts. Likewise, a data space is always referred to as $\mathcal{X}$ and samples as S.

To communicate the difference between objects, we need a notion of class.

Definition 2 (Class, Label)**.**

A class of objects shares some characteristic. A label y_* refers an object x_* to a class.

For a sample, S, the labels corresponding to the objects $x_* \in S$ are denoted y_*, and the set of labels for the sample S is the list $Y = \{y_* : x_* \in S\}$. This manuscript always denotes the label of an object x_* as y_*. The connection between the two is denoted by matching subscripts.

In our example, we have the class of cats and the class of dogs. Each object in our sample is labeled as either a cat or a dog. We want to define a method for a computer to

distinguish between cats and dogs. One naive –yet decent– method is the K-nearest neighbor classifier.

Definition 3 (Metric, Semimetric)**.**

For a set M, a function $d : M^2 \to \mathbb{R}$ is a metric on M, forming the metric space (M, d) if it exhibits the following four properties. For all $m_1, m_2, m_3 \in M$

1. Reflexive: $d(m_1, m_1) = 0$
2. Positive: $m_1 \neq m_2 \Rightarrow d(m_1, m_2) > 0$
3. Symmetric: $d(m_1, m_2) = d(m_2, m_1)$
4. Triangle Inequality: $d(m_1, m_2) \leq d(m_1, m_3) + d(m_3, m_2)$

A function d is a semimetric on M if it violates the triangle inequality, yet satisfies the other three properties. Then (M, d) is a semimetric space.

The word metric will sometimes be used generically in this manuscript to refer to either a metric or semimetric. In Section 3.2, the distinction between the two will be relevant and made explicit.

Distance in the usual sense, Euclidean distance, forms the metric space $(\mathbb{R}, d_{Euclidean})$ on the real number line. The angle between points on a circle, angular distance, forms a metric space $(\mathbb{S}^2, d_{angular})$ on the unit circle.

Definition 4 (K-Nearest Neighbor (KNN) Classification)**.**

Let x_u be a query object, and $x_1, \ldots, x_k$ be the K closest neighbors to x_u w.r.t. some metric. K-nearest neighbors assigns x_u the label that occurs the most out of the labels $y_1, \ldots, y_K$ of its K nearest neighbors.

By using a KNN classifier, we have a method that can be used for a computer to classify images as either a cat or a dog. This method relies on an assumption that within the data space, $\mathcal{X}$, images of cats are closer to other images of cats than to images of dogs,

and vice versa. For the KNN classifier to be effective on our images of cats and dogs, we need the images of cats to be closer to other images of cats than to images of dogs, and vice versa. If we had a function to map our images to a new space, where the images of cats are indeed closer to other cats than to dogs, then in this space the KNN classifier would be useful.

Definition 5 (Embedding Function, Parameters, Embedding Space, Embeddings)**.**

We call $f(\cdot;\theta) : \mathcal{X} \to \mathcal{Z}$ embedding function with parameters θ and $\mathcal{Z}$ the embedding space. An object $x_* \in \mathcal{X}$ has embedding $z_* = f(x_*;\theta)$.

The parameters, θ, of an embedding function are what determine how f embeds an object. If θ is changed, the value of the embeddings produced by f are changed. The parameters of a function are always denoted as θ.

In this manuscript, f refers to an embedding function in the sense described here. Note that our use of the word embedding differs from the use of the word in, for example, topology. An embedding space is always referred to as Z, and the embedding of an object, x_*, is denoted z_*, sharing its subscript.

If we were to possess an embedding function, f, which embedded images of cats within one area, and images of dogs within another distant area, then our KNN classifier would be able to accurately distinguish between images of the two animals. The purpose of this paper is to explore the field of contrastive learning, in which we attempt to "learn" the best f for a given problem and a chosen metric.

1.2 BACKGROUND

1.2.1 IDENTIFICATION OF SOURCE

Forensic identification of source (source ID) problems, within the field of forensic science, are concerned with a query object, x_u, and its unknown label, y_u. With the presence of a gallery sample, $S_{gallery} = \{x_1, \ldots, x_n\}$, and their known labels, $Y_{gallery} = \{y_1, \ldots, y_n\}$,

we can utilize statistical techniques to make a well-informed inference about the value of y_u.

Source ID problems can generally be distinguished into two classes, those of classification and those of verification.

Definition 6 (Classification, Verification)**.**

Classification problems assume the $S_{gallery}$ and their known labels $Y_{gallery}$ contain an exhaustive list of possible labels for x_u

- E.g., For a fired cartridge casing from one of five guns: which of the five guns fired the cartridge?

Verification problems relax the assumption of an exhaustive list of possible labels being present in the $S_{gallery}$.

1. E.g., For a fired cartridge casing and a specific gun: was this cartridge fired from this gun?

2. E.g., For a fired cartridge casing and five possible guns which fired it: which of the five guns, if any, fired the cartridge?

3. E.g., For two fired cartridge casings: were these cartridges fired by the same gun?

Classification problems are known as closed-set problems. Example 1 is a type of verification problem called a specific source problem [19, Sec. 5]. Example 2 is an open set problem, which in some ways generalizes both the closed set problem and the specific source problem. Example 3 is a common source problem [19, Sec. 4].

Because verification problems are generalizations of classification problems, methods addressing verification problems are more versatile than classification problems. Consider the verification problem "How similar is the cartridge casing found on the crime scene to cartridge casings fired from these ten firearms?" This can be easily specified to a

classification problem by selecting the firearm with the most similar cartridge casings to the query object. The classification form of this problem might be presented as, "Which of these ten firearms was the cartridge casing found on the crime scene fired from?"

1.2.2 DEEP LEARNING

Machine learning is the practice of using algorithms to learn how to perform a task on some training data, S_{train}, with the hope of generalizing to unseen data, S_{test}.

Supervised learning is a sub-field of machine learning where the training data, S_{train}, has known labels, Y_{train}. The labeled data can be leveraged to improve performance. This is as opposed to unsupervised learning, where labels, Y_{train}, are either unknown or irrelevant. Classification is an example of a supervised learning task, where for each object, we wish to predict a label for that object.

Definition 7 (Deep Learning)**.**

Deep learning is a sub-field of machine learning concerned with the use of multi-layered (deep) neural networks to learn a useful representation of data.

The archetypal deep neural network is a multi-layer perceptron (MLP) which is illustrated in Figure 1. In general, MLPs have construction

$$f(x;\theta) = \phi \circ f_L(\cdot;\theta_L) \circ \phi \circ f_{L-1}(\cdot;\theta_{L-1}) \circ ... \circ \phi \circ f_2(\cdot;\theta_2) \circ \phi \circ f_1(x;\theta_1), \qquad (1)$$

where the functions f_i, called the layers of the network, are affine. The function, ϕ, is a nonlinear function and is called an activation function because it acts element-wise upon its inputs. Neural networks of arbitrary width are proven to be universal approximators of continuous functions [11].

In deep learning, we want to train a network to perform a specific task. This learning process can be conceived as learning an embedding function, f, which serves as a feature extractor for objects. To measure the network's performance, we use the concept

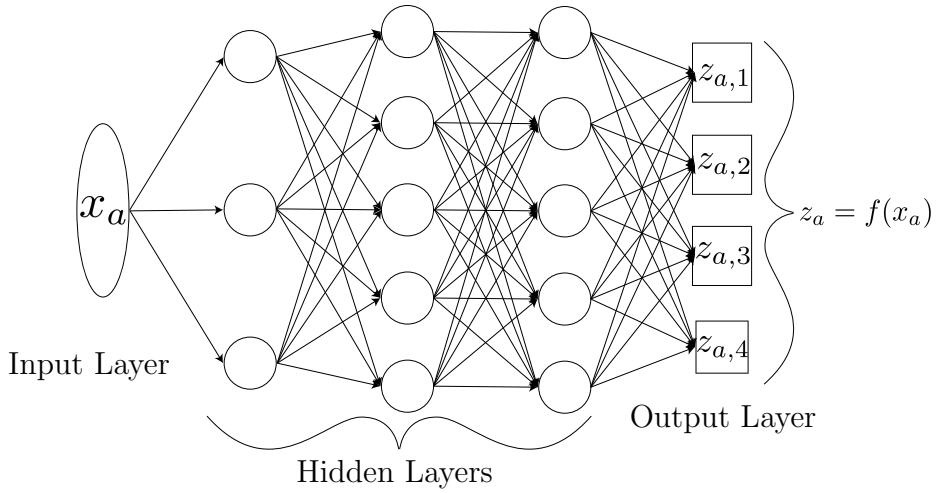

Figure 1: Depiction of a 4-layer MLP. Each column of arrows represents a component affine function f_i of the network, f (see Equation 1). The columns of circles are successive hidden representations of the data and the final column of squares is the final output vector. In a network trained by cross-entropy loss (Definition 9), an element $z_{a,i}$ of the output corresponds to the probability that x_a belongs to class i.

of a loss function.

Definition 8 (Loss Function)**.**

A loss function $\mathcal{L}(f(S;\theta))$ is a measure of the goodness of f. A low loss indicates f is a good embedding function.

In deep learning, the parameters, θ, of f are optimized to decrease the loss function with respect to a sample, S_{train}, called the training data. The process known as training refers to repeatedly passing objects through the network, computing the loss function, and updating the network's parameters to reduce the loss function.

The idea is that as the loss function is decreased, f will be better suited for performing some task. For this to be effective, the decreasing of the loss function must correspond with increased proficiency in the given task.

Image classification is a supervised learning task, and deep learning is typically used for classification by optimizing a network with respect to cross-entropy loss.

Definition 9 (Cross-Entropy Loss, Cross-Entropy Network)**.**

For a problem with C classes, an object $x_* \in \mathcal{X}$ belonging to class j has label $y_* \in Y \subseteq \mathbb{R}^C$ which is a unit vector along the j^{th} axis. The embedding space of the network is $\mathcal{Z} = \mathbb{R}^n$.

$$\mathcal{L}(S, Y; f) = -\sum_{x_i \in S} y_i \cdot \log z_i \tag{2}$$

Then the cross-entropy loss for a single object belonging to class c is the negative *log* of the j^{th} component of $z_* = f(x_*)$.

We call a neural network a cross-entropy network if it is trained by cross-entropy loss.

To train a cross-entropy network, our embedding space $\mathcal{Z}$ equals $\mathbb{R}^C$ where C is the number of unique classes to which an object may belong. Each unique class is assigned a point of distance one from the origin along an axis in $\mathbb{R}^{\mathbb{C}}$, which we will call its class point. We predict a query object x_u to belong to the class whose class point it is nearest. The cross-entropy loss measures how far an object's embedding is from its correct class point. The cross-entropy loss is then minimized where the predicted labels of a dataset are the correct labels.

In the previous cats and dogs example, to use a network trained by cross entropy our embedding space $\mathcal{Z}$ would equal $\mathbb{R}^2$ since we have two classes. Any object x_c which is an image of a cat would be assigned label $y_c = (0, 1)$ and any object x_d which is an image of a dog would be assigned label $y_d = (1, 0)$ (the assignment of cats to $(0, 1)$ and dogs to $(1, 0)$ is arbitrary). The cross-entropy loss measures how close the embeddings $z_c = f(x_c)$ of cats are to $(0, 1)$ and the embeddings $z_d = f(x_d)$ of dogs are to $(1, 0)$. (See figure 2).

Cross-entropy loss works well for the task of classification. Other loss functions are used for other tasks, but most are measured by the distance between a predicted label and a true label.

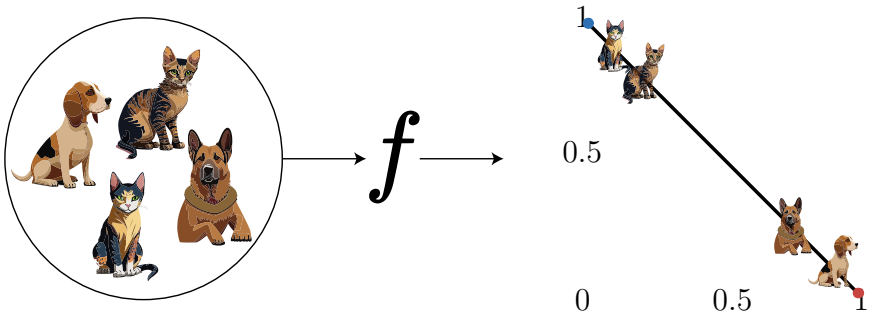

Figure 2: To use a cross-entropy network to classify an image as a cat or dog, we assign the points $(0, 1)$ and $(1, 0)$ to cats and dogs respectively. The elements of the embeddings must sum to 1. The cross-entropy loss measures how close the cat embeddings are to $(0, 1)$ and dog embeddings are to $(1, 0)$.

1.2.3 CONTRASTIVE LEARNING

Definition 10 (Contrastive Loss Functions, Contrastive Neural Networks, Contrastive Learning)**.**

A contrastive loss function measures the goodness of an embedding function f by its ability to embed objects of the same class close together and embed objects of different classes far apart with respect to some metric, d.

A contrastive neural network (contrastive network) is a neural network that is trained by optimizing a contrastive loss function.

Contrastive learning is the subfield of deep learning which uses contrastive networks.

The goal of contrastive learning is to learn the parameters for an embedding function, f, which maps objects from $\mathcal{X}$ to $\mathcal{Z}$ in such a way that a chosen d meaningfully demonstrates the similarity between two objects (see figure 3).

Networks using contrastive loss functions are fundamentally different from networks using loss functions such as cross-entropy loss. In the latter case, an object, x_*, is passed through the embedding function, f, and the loss is calculated as a function of a

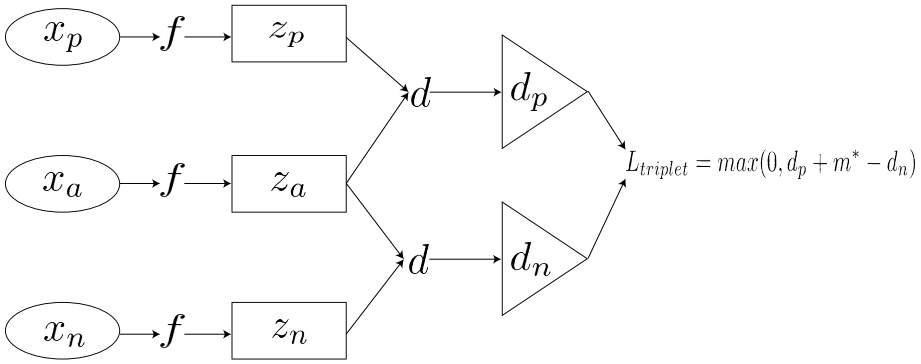

Figure 3: A contrastive network using the triplet loss function (Definition 15). Multiple objects (x_a, x_p, x_n) are passed through the same embedding function, f. Their embeddings (z_a, z_p, z_n) are contrasted by using a metric function, d, to measure the distances $d_p = d(z_a, z_p)$ and $d_n = d(z_a, z_n)$. The triplet loss decreases when d_p is decreased relative to d_n.

distance between an embedding $z_* = f(x_*)$ and its label, y_*. Loss is reduced by decreasing the distance between z_* and y_*. In contrastive networks, multiple objects simultaneously pass through the embedding function and then the distances between them are computed. The loss is reduced by decreasing the distance between objects of the same class and increasing the distance between objects of different classes.

We return to the cats and dogs example to illustrate what makes a neural network a contrastive network. Any object, x_c, which is an image of a cat would be assigned label $y_c =$ "cat" and any object, x_d, which is an image dog would be assigned label $y_d =$ "dog" (these labels are completely arbitrary. All that matters is that all cats share a label, all dogs share a label, and these labels are distinct). A contrastive network embeds multiple objects and computes the distances between these objects with a chosen metric function, d. Contrastive loss is calculated based on the closeness of objects which share a class, and the distance of objects of different classes.

Contrastive loss functions differ from cross-entropy loss because in order to use

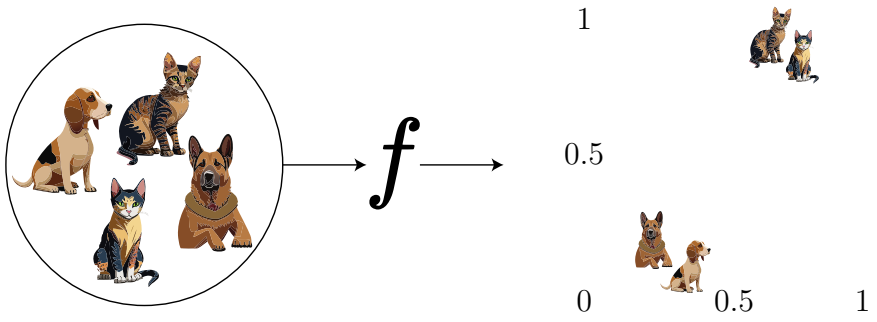

Figure 4: In contrastive networks, objects are embedded in such a way to keep objects within a class close together and objects in different classes far apart. Where in the embedding space $\mathcal{Z}$ the embeddings lie does not matter.

cross-entropy loss, we must first decide where in some embedding space we want each class to be mapped. To use cross-entropy loss in a network with some data, we must first perform one-hot encoding on the data, which effectively chooses a point in Euclidean space that objects of each class should map to. With contrastive loss, the contrastive network learns an embedding function where objects of each class should be close together and far apart from objects of different classes. Where, in the embedding space, particular classes are embedded is irrelevant, all that matters is that similar objects are clustered together. What makes contrastive loss functions special, is that they evaluate embeddings in this way, and not based on their distance from a predetermined, desired embedding.

Contrastive learning can be used to obtain a representation of a dataset wherein distances between objects accurately reflect their similarity. This representation can then be parleyed into further deep learning techniques, or traditional statistical methods, to solve a given problem. The use of contrastive learning for this purpose becomes attractive within the first steps of application, as machine learning techniques have long been favored over humans in the task of feature selection. Having an automated, mathematical procedure relieves doubt instilled by humans choosing what data is relevant, and deep

learning is heralded for its ability to detect complex –not humanly discoverable– patterns inherent to a dataset.

After training a contrastive network, there is no standard way to implement the newly obtained embedding function. Part of the appeal of contrastive networks is their agnosticism toward implementation. It could be used in standard statistical methods; such as K-nearest neighbors for classification, or hierarchical clustering to assess the similarity of different objects. For verification problems, appropriate positive or negative error rates can be chosen to set a threshold for classification based on the metric function. For forensic identification of source problems, the dissimilarities between objects could also be used in likelihood ratios. Further deep learning could also be implemented.

1.3 RELATED WORK

In this section, we review literature relevant to the theory and application of contrastive neural networks. One goal of this work is to tie together some scattered ideas within the field of contrastive learning. Among them are various methods of implementing the contrastive learning paradigm, different applications of contrastive networks, as well as general ideas within the field of deep learning that could be applied in contrastive learning.

The history of contrastive networks is not well documented, but we attempted to piece it together. Contrastive networks first appeared in Bromley et al. [2] under the name of "Siamese Networks". Pairwise contrastive loss was later defined in Chopra et al. [5]. In pairwise contrastive loss, an object is contrasted with one other object at a time. Triplet loss first appeared in Weinberger et al. [30]. In triplet loss, an object is contrasted with one object from the same class and one object from a different class at the same time. N-pair loss [22] was proposed as an extension of triplet loss in which an object is contrasted with one object from the same class and multiple objects from different classes. A variant of N-pair loss was later popularized in van den Oord et al. [26] [20, Sec. 2.3]. To our knowledge, SupCon loss was introduced in Khosla et al. [13] and can be seen as an

extension of N-pair loss which allowed an object to be contrasted with multiple objects from the same class and from different classes at once.

The lack of adequate labeled datasets deters supervised deep learning techniques from being deployed on some problems. Self-supervised learning, as proposed in De Sa [6], is a method for using prior knowledge to exploit data from multiple modalities to create pseudo-labels for unlabeled data. Contrastive networks have been used to learn a shared embedding space for objects from multiple modalities [15]. The self-supervised approach has been used in contrastive learning [26, 9] or unimodal data and more recently for multimodal data in Radford et al. [20] and Wang et al. [29]. In 2022, Yu et al. [32] achieved new state-of-the-art performance on the ImageNet data by building on the contrastive multimodal learning work by Radford et al. [20].

Small datasets can limit the effectiveness of machine learning techniques. The few-shot learning (FSL) problem is formally defined, and multiple approaches to FSL are given in Wang et al. [28]. Few-shot learning can be approached by using contrastive networks for self-supervised learning [9]. A unique approach to FSL, with global class representations, is given in Li et al. [14].

Contrastive learning can be made more efficient and effective by properly choosing objects to contrast, known as batch-mining. The effectiveness of batch-mining is demonstrated in Schroff et al. [21, Sec. 2] and Hermans et al. [10]. In recent scholarship, Mitrovic et al. [16] has shown that including the maximal amount of negative examples does not produce the best results. Furthermore, Tian et al. [25] explores how contrastive learning is possible without any negative examples at all.

Identifying relevant geometric priors for a deep learning problem can have a significant impact on sample complexity [1]. In Bronstein et al. [3], a procedure is outlined for implementing these geometric priors into deep learning architectures.

A major motivation for this work is the application of deep learning techniques to forensic identification of source problems. An encouraging example of the use of deep

learning for forensic identification of source is given in Mookiah et al. [17]. Forensic identification of source problems either fall under the category of classification or verification. Contrastive networks can be used to address both categories of problems, which is why we consider them for the task of forensic identification of source. The joint training of a neural network for classification (by cross-entropy loss) and verification (by contrastive loss) problems is demonstrated to be effective in Sun et al. [24]. There are nuances to properly posing a forensic identification of source problem, which are identified and made clear in Ommen and Saunders [19].

A detailed statistical analysis of the NBIDE data is given in Vorburger et al. [27].

Also mentioned in Hénaff et al. [9, Sec. 4.1] is that batch normalization appears to exert a negative effect on the model accuracy. We found this to be true for our studies.

1.4 ORGANIZATION

As an initial proof of concept, we will demonstrate a contrastive network's ability to learn a useful representation. Our experiments indicate that contrastive networks perform strongly at classification on small datasets; however, their use is not limited to the domain of classification. We provide some insight into alternative uses for contrastive networks.

To maximize the potential of contrastive networks, we will review different methods of constructing contrastive networks in an effort to find which produces the best results.

Within contrastive learning, there are multiple choices of loss function. This study examines three: pairwise contrastive loss, triplet loss, and supervised contrastive loss.

The relevant notion of "distance" is also a choice to be made, as any metric function will suffice. This study examines five metric functions: angular dissimilarity, cosine dissimilarity, Chebyshev distance, Euclidean distance, and arctangent dissimilarity.

In contrastive learning, it is common to not only stipulate that the distance between objects of different classes is greater than the distance between objects of the same class

but to stipulate that the former distance must be greater than the latter plus some additional distance, called the margin value, m^*. This study examines the effect of the choice of margin value on the training of a network.

Relevant to the broader field of deep learning is whether contrastive networks learn a representation of a dataset that possesses desirable invariance properties. For example, that embeddings of the same class are all within some distance from one another, and roughly the same distance from embeddings of different classes. This would affirm the usefulness of contrastive networks, as their implementation would effectively increase the size of a dataset.

To best use contrastive learning for forensic science, or any application, these foundational questions must be attended to. The goal of this work is to address them and to organize the various ideas within the field of contrastive learning.

2 PROOF OF CONCEPT

Contrastive networks are optimized only to learn an embedding function of a dataset. Therefore, they can be used for a variety of applications, especially considering they can be used for either supervised or self-supervised learning. If the goal of "...deep learning is to meaningfully transform data: in other words, to learn useful representations of the input data" [4, Sec. 1.1.3], then contrastive networks are easy to endorse because this is their only stated goal. Other networks, such as those utilizing cross-entropy loss, learn representations merely as a byproduct while synthesizing a classifier. This section advocates for contrastive networks by first illustrating their ability to learn desirable embeddings and then demonstrating their potential for classification problems and few-shot learning.

In this section and the proceeding Section 3, we use the MNIST dataset, illustrated in Figure 5, to display the effectiveness of contrastive learning, and study different methods of contrastive learning. The MNIST dataset is comprised of 70,000 $28 * 28$ pixel

images of handwritten digits. In both Sections 2 and 3 we use contrastive networks to learn an embedding function, $f : \mathbb{R}^{28*28} \to \mathbb{R}^{16}$, which maps the data to an embedding space where the KNN classifier can be effectively used.

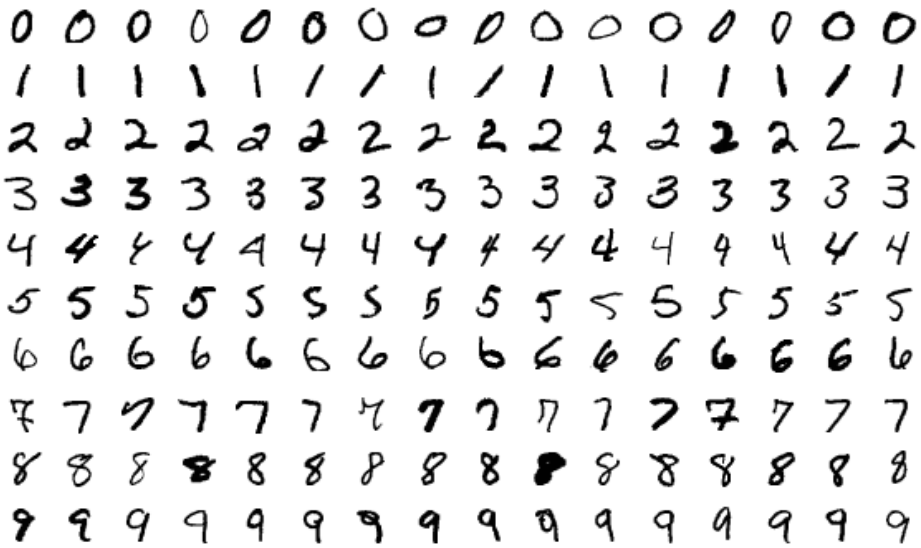

Figure 5: Examples of images in the MNIST dataset. Image sourced from Wikipedia contributors [31].

To display the general efficacy of contrastive networks, Figure 6 displays three 2-dimensional principle component analyses (PCAs) of the MNIST dataset, with different digits represented by different colors. We performed a 2-dimensional PCA of the raw dataset, meaning the MNIST data in its data space, $\mathbb{R}^{28*28}$. This process reduces the data down to $\mathbb{R}^2$, but in such a way that the maximum amount of variability between the data is retained in two dimensions. The first plot shows only a random sample of the data for clarity. In this plot, all of the classes are jumbled together, which makes it difficult to distinguish between classes.

To produce the second plot in Figure 6, a contrastive network was trained on thousands of MNIST images, to find an embedding function $f : \mathbb{R}^{28*28} \to \mathbb{R}^{16}$. A PCA was then performed on the embeddings of the training data, and a random sample of them is shown in the second plot. Here, there is a much higher degree of separation between objects of different classes. Given a random point from this plot, we could have a good

idea about which class it belongs to based on its location.

Deep learning practitioners must be cautious of overfitting the training data, meaning that the model learns to only be effective on the training data and will not generalize well to new data. However, in the third plot of Figure 6, we performed a 2-dimensional PCA on the validation data used to train our model. This plot shows that the model has not overfit and does generalize well, which is encouraging. Since we used thousands of MNIST images and it is not especially surprising that the network performs well; the inclusion of Figure 6 is purely to assure the reader that contrastive networks can produce their promised results.

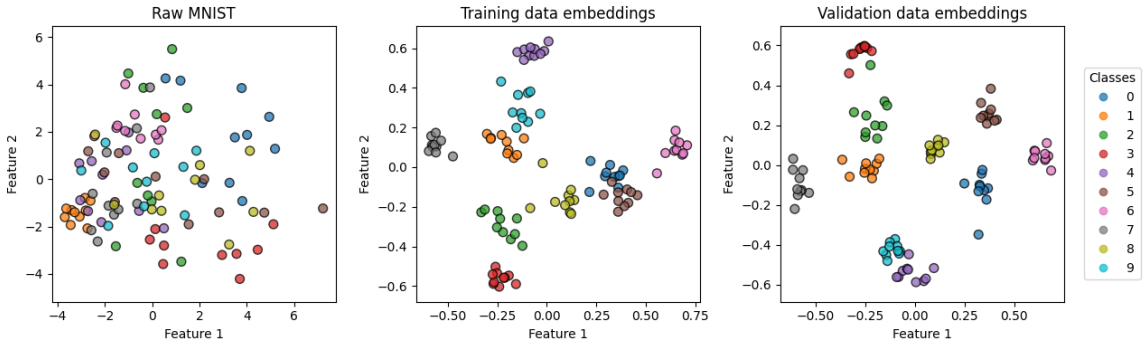

Figure 6: On the left is a PCA of the raw MNIST dataset. In the middle is a PCA of the training data embedded by learned f. On the right is a PCA of the validation data embedded by learned f. Different colors correspond to different digits, indicated by the legend on the right side of the figure.

To test the efficacy of contrastive networks in few-shot learning classification problems, we again used the MNIST dataset. We randomly sampled 5 images of each digit for training data, S_{train}, and 5 more for validation data, $S_{validation}$, resulting in 10 images of each digit to be used for the training process. In few-shot learning, this is referred to as a 10-shot 10-way classification problem. With our sample, we tested three different classification methods.

First, we trained a 1-nearest neighbor classifier on the training data, S_{train}, in the data space $\mathbb{R}^{28*28}$ (the 1-nearest neighbor classifier is the K-nearest neighbor classifier when $K = 1$, see Definition4). We also trained a cross-entropy network (Definition 9)

using S_{train} and stopped training when maximum accuracy on $S_{validation}$ was achieved. Lastly, we trained a contrastive network on S_{train}. Throughout the training we monitored our progress by training a 1-nearest neighbor classifier on the embeddings of the training data, $E_{train} = \{z_i : z_i = f(x_i), \forall x_i \in S_{train}\}$, and testing our accuracy on the embeddings of the validation data, $E_{validation} = \{z_i : z_i = f(x_i), \forall x_i \in S_{validation}\}$. We stopped training when maximum accuracy on $E_{validation}$ was achieved.

We then randomly sampled 1000 new MNIST images, S_{test}, and used them to test the classification accuracy of the 1-nearest neighbor classifier on the raw data, the cross-entropy network, and the 1-nearest neighbor classifier on their embeddings $E_{test} = \{z_i : z_i = f(x_i), \forall x_i \in S_{test}\}$ produced by the contrastive network.

This process was repeated 100 times, and the results are shown in the form of a boxplot in Figure 7. The 1-nearest neighbor classifier was outperformed by both deep learning techniques. The cross-entropy model achieved a slightly higher accuracy and lower standard deviation than the contrastive network.

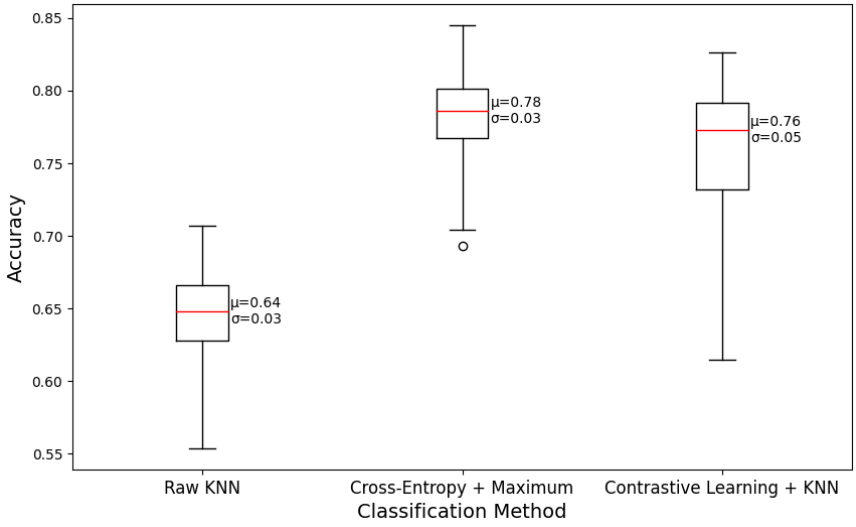

Figure 7: Boxplot of accuracies achieved by the 1-nearest neighbor classifier on S_{test}, by the cross-entropy network, and by a 1-nearest neighbor classifier on the embeddings E_{test}.

Classification tasks are not the only use of contrastive networks. Verification is perhaps the more natural way to implement the embedding function from contrastive

learning.

Dissimilarity scores, which measure the dissimilarity between objects, can be obtained by passing two objects through the embedding function and then taking the distance between them. We can then set a threshold of positive or negative classification based on the chosen error rate(s). In the setting of forensics for criminal prosecution, we might find a high false negative error rate is a worthy trade-off for a low false positive error rate, to avoid false incrimination.

This manuscript specifically studies contrastive networks for use in supervised learning. However, much ongoing research in contrastive learning focuses on unsupervised learning, where for the training dataset S_{train}, the labels Y_{train} are unknown. The essence of these methodologies is to consider each object $x_* \in S$ as in a class of its own. To use contrastive learning, objects are augmented to produce x_*^+. Contrastive learning is then used to embed each x_* and its augmentation(s) x_*^+ closer together. Studies [26, 9] have found that this often results in an embedding space where objects that have the same class are embedded closer together than objects from different classes.

For this research, a major motivating use case for contrastive networks was in attaining similarity scores to be used in score-based likelihood ratios.

Theorem 1 (Neyman-Pearson Lemma)**.** *Suppose we have a sample S, a simple null hypothesis, H_0, and a simple alternative hypothesis, H_a. Then for any given probability α of falsely rejecting H_0, the test with the highest probability of accurately rejecting H_0 is given by rejecting H_0 when $\frac{L(S|H_0)}{L(S|H_a)} < k$ for k determined by chosen α.*

Suppose we can use contrastive networks to learn an embedding function f with certain invariance properties (Section 4), namely that intra-class distances have the same distribution for all classes, and inter-class distances have the same distribution between all classes. Consider a specific source problem (see Definition 6) with query object x_u, objects from our suspected source $x_i \in S_{suspect}$, and objects from various other classes $S_{alternative}$. If our hypotheses are H_0: that our query object shares a class with objects in

$S_{suspect}$, and H_a : that x_u comes from a different class than the objects in $S_{suspect}$, then we can use our embedding function, f, to model the numerator of the likelihood ratio as the likelihood of distances between the embedding of our query object, $z_u = f(x_u)$, and embeddings, $E_{suspect} = \{z_i : z_i = f(x_i), \forall x_i \in S_{suspect}\}$, under an empirical distribution function given by distances between embeddings within $E_{suspect}$. We can model the denominator as the likelihood of distances between z_u and embeddings $z_i \in E_{suspect}$ under an empirical distribution function given by distances between embeddings in $E_{suspect}$ and embeddings in $E_{alternative} = \{z_i : z_i = f(x_i), \forall x_i \in S_{alternative}\}$.

In this way, the Neyman-Pearson Lemma can be used to attain, for any chosen false negative rate, the test with the highest true negative rate. For example, with a fingerprint found on a crime scene and a suspect, this method allows us to choose a probability of falsely acquitting the suspect, and retain the maximum probability of accurately acquitting them.

3 ABLATION STUDY

Different contrastive networks can be constructed by changing particular aspects of the network. Examples include the loss function, the metric function, and the margin value. In this section, we perform an ablation study, where we change one of these factors at a time while holding the other two constant. Through this experiment, we hope to gain a better understanding of the roles played by the choice of loss function, metric function, and margin value in contrastive learning.

3.1 LOSS FUNCTIONS

In deep learning, networks are trained by optimization with respect to a loss function. A loss function must be chosen concerning the task at hand. For example, a loss function used for a classification model will not apply regression.

Contrastive learning refers to the use of models that are trained with loss functions

of a general type, which take multiple inputs and "contrast" them.

Definition 11 (Positive Pair, Negative Pair)**.**

A positive pair is defined to be a pair of objects (x_a, x_b) such that $y_a = y_p$.

A negative pair is defined to be a pair of objects which is not a positive pair.

Note that if a pair of objects in a data space form a positive pair, then their embeddings in the embedding space also form a positive pair. The same is true for negative pairs.

Definition 12 (Positive Distance, Negative Distance)**.**

A positive distance is to be defined as $d(z_a, z_p)$ such that z_a and z_p, form a positive pair.

A negative distance is to be defined as $d(z_a, z_n)$ such that z_a and z_n, form a negative pair.

When we refer to positive distances and negative distances we will always be referring to the distances between the embeddings of a pair, not of the objects in data space.

A contrastive loss function calculates positive and negative distances then combines these values in some way to yield a loss. Cross-entropy loss differs from contrastive loss because it requires all classes to be pre-assigned to a point, and then calculates loss based on how far objects are from the point corresponding to their class. In some sense, contrastive loss functions allow the model to learn the embedding points of each class, acting as a generalization of cross-entropy loss.

Within the domain of contrastive loss, there are different contrastive loss functions. Understanding the differences between these loss functions and the results they produce will allow for more effective application of contrastive learning.

3.1.1 PAIRWISE CONTRASTIVE LOSS

The pairwise contrastive loss function is, in some sense, the most naive contrastive loss function. Pairwise contrastive loss was first introduced in Chopra et al. [5].

Definition 13 (Pairwise Contrastive Loss)**.**

We define the pairwise contrastive loss, denoted as $\mathcal{L}_{pair}$ as

$$\mathcal{L}_{pair}(S, f) = \sum_{x_a \in S} \left[\frac{1}{|P(x_a)|} \sum_{x_p \in P(x_a)} d(z_a, z_p) + \frac{1}{|N(x_a)|} \sum_{x_n \in N(x_a)} \max(m^* - d(z_a, z_n), 0) \right] \tag{3}$$

$$= \sum_{x_a \in S} \frac{1}{|P(x_a)|} \sum_{x_p \in P(x_a)} d(z_a, z_p) + \sum_{x_a \in S} \frac{1}{|N(x_a)|} \sum_{x_n \in N(x_a)} \max(m^* - d(z_a, z_n), 0) \tag{4}$$

where $P(x_a)$ is the set of objects in the same class as x_a, $N(x_a)$ is the set of objects in a different class than x_a. The margin value, m^*, is a hyperparameter that prevents certain undesired behaviors in training (see Section 3.3).

The second form of pairwise contrastive loss provides valuable intuition into what makes a neural network a contrastive network. In the first summation for Equation 4,

$$\sum_{x_a \in S} \frac{1}{|P(x_a)|} \sum_{x_p \in P(x_a)} d(z_a, z_p),$$

the average positive distance within the dataset, S, is calculated. This term decreases when the average positive distance decreases.

The second summation of Equation 4,

$$\sum_{x_a \in S} \frac{1}{|N(x_a)|} \sum_{x_n \in N(x_a)} \max(m^* - d(z_a, z_n), 0),$$

is slightly more complicated. We call the term $\max(m^* - d(z_a, z_n), 0)$ the hinge and it

operates as the hinge loss in support vector machines. By the inclusion of the hinge, the summation above decreases as negative distances increase, but only until they reach the margin value, m^*. The significance of the hinge will be further explored in Section 3.2.1.

As the number of classes and number of objects per class grows, the number of positive pairs will be eclipsed by the number of negative pairs. For example, with MNIST data there are 10 classes corresponding to the 10 digits. If we take S to be a sample comprised of two images from each class, then there are 10 positive pairs and 180 negative pairs, a ratio of 1/18. As we move the sample to 10 images from each class there are 450 positive pairs and 9000 negative pairs, a ratio of 1/20. Once we sample 1000 images of each digit we have 4,995,000 positive pairs and 900,000,000 negative pairs, a ratio of 1/180. This is not even the entire MNIST dataset.

This leads to a major drawback of pairwise contrastive loss. The network is rewarded more for moving all the negative embeddings apart than it is penalized for moving the positive pairs apart. Because of this, there is often a need to weigh the contributions of positive pairs and negative pairs to the loss function. Thus in Equation 3, the version of pairwise contrastive loss implemented in our study, an average loss is computed for positive pairs, and another for negative pairs, and then the two averages are added to create the pairwise contrastive loss. This way the model cannot arbitrarily increase all distances to decrease loss and must pay equally as much attention to reducing positive distances as to increasing positive distances.

3.1.2 TRIPLET LOSS

Triplet loss was first introduced in Weinberger et al. [30], although it was not termed so at that time. The triplet loss function takes a "triplet" of objects as input, hence the name.

Definition 14 (Triplet)**.**

A triplet, $T = (x_a, x_p, x_n)$, is composed of an anchor object x_a, a positive object x_p, and a negative object x_n in such a way that the anchor and positive

object share a class label, while the negative object has a different label.

Definition 15 (Triplet Loss Function)**.**

For the set T of all triplets formed by a sample S and its labels Y

$$\mathcal{L}_{triplet}(T; f) = \sum_{(x_a, x_p, z_n) \in T} max(0, d(z_a, z_p) + m^* - d(z_a, z_n)). \tag{5}$$

The presentation of this function is different from that of the pairwise contrastive loss function (Equation 3). Before, we shared the loss pairwise contrastive loss function with respect to a given object x_u. Here, for clarity, we show the triplet loss function with respect to a triplet.

A hinge term is again included in triplet loss, this time as

$$max(0, d(z_a, z_p) + m^* - d(z_a, z_n)).$$

The purpose of the hinge in triplet loss is the same as in pairwise contrastive loss; however, it is implemented slightly differently. The hinge in triplet loss allows the loss to decrease as negative distances increase, but only until the negative distance reaches the positive distance plus the margin.

Triplets are characterized by the relative magnitudes of the positive distances, the negative distances, and the margin. They fall into three different categories which are relevant for a process known as batch-mining. In batch-mining, before every epoch, the network selects which triplets to use for training. Furthermore, they help to make sense of the triplet loss function.

Definition 16 (Easy Triplets, Semihard Triplets, Hard Triplets)**.**

- Easy triplets are those such that the negative distance is greater than the positive plus the margin: $d(z_a, z_p) + m^* < d(z_a, z_n)$.

- Semi-hard triplets are triplets where the negative distance is greater than the positive distance, but less than the positive distance plus the margin: $d(z_a, z_p) < d(z_a, z_n) < d(z_a, z_p) + m^*$.
- Hard triplets are those in which the negative distance is less than the positive distance: $d(z_a, z_n) < d(z_a, z_p)$.

In the case of easy triplets, the loss is a constant 0, therefore the gradient will be a constant 0. Because of this, easy triplets can be considered a nuisance to training, and batch-mining techniques seek to avoid them.

Semi-hard triplets are sometimes, but not always sought after during batch-mining. An individual semi-hard triplet contributes a maximum of m^* to the loss and therefore has a non-zero but potentially less drastic effect on the gradient than hard triplets.

Hard triplets are always desired in batch-mining, and they are the exclusively relevant triplets in hard batch-mining. A hard triplet contributes a minimum of m^* to the loss and therefore has the potential to greatly affect the gradient. A dataset with no hard triplets is a dataset in which the 1-nearest-neighbor classifier will yield 100% accuracy. The hard-batch-mining may yield perfect performance.

Triplet loss has been implemented with various batch-mining techniques, which can produce different results. These techniques were not a focus of this study, but Schroff et al. [21] found that hard batch-mining can increase the effectiveness of triplet loss.

An advantage of triplet loss is that it avoids the problem of too many negative pairs by instead using triplets which ensure a positive and negative pair are always being considered together. Perhaps because of this property, triplet loss typically outperforms pairwise contrastive loss.

3.1.3 N-PAIRS LOSS

Over time the N-pairs, loss was introduced [22] as an extension of triplet loss. In N-pairs loss, the network trains on an N-pair instead of a triplet. An N-pair still consists of the

anchor and its positive counterpart, but instead of just one negative object, it contains N-1 negative objects. Therefore an N-pair is a generalization of a triplet, and a triplet is simply a 2-pair.

N-pair loss is not considered in this study. It is relevant to the reader because it has been found to outperform triplet loss [22], and it is relevant to the evolution of contrastive learning.

3.1.4 SUPCON LOSS

The last loss function considered in this study is the SupCon (short for Supervised Contrastive) loss. The SupCon loss function was first proposed in Khosla et al. [13].

Definition 17 (SupCon Loss)**.**

$$\mathcal{L}_{supcon}(S, f) = \sum_{x_a \in S} \frac{-1}{|P(x_a)|} \sum_{x_p \in P(x_a)} \log \frac{\exp(z_x \cdot z_p/\tau)}{\sum_{x_n \in N(x_a)} \exp(z_x \cdot z_n)/\tau)} \tag{6}$$

where τ is a hyperparameter called the temperature.

In our study, we kept $\tau = 0.1$ as per the findings of Khosla et al. [13, Sec. 4]. The metric, d, has been replaced by a dot product, which is a measure of similarity. Similarity is seen as the opposite of distance. By L_2 normalizing the embeddings, the dot product becomes the cosine similarity, which is 1 minus the cosine dissimilarity (see Equation 11). Other metric functions can be similarly adapted to product similarities instead of distances.

The SupCon loss function was conceived as an extension of the self-supervised contrastive loss used in van den Oord et al. [26] and Hénaff et al. [9]. However, the authors of Khosla et al. [13] point out that it can also be considered a generalization of the N-pairs loss. In SupCon loss, the network "contrasts" each embedding with all of its associated positive and negative embeddings. This is similar to how N-pairs loss extended the triplet loss to be used with multiple negative embeddings.

The SupCon loss function is very similar to the pairwise contrastive loss function. It is more similar in composition to pairwise contrastive loss (Equation 3) than to N-pair loss [22] or to triplet loss (Equation 5). Positive and negative distances are entangled in the latter two loss functions, while completely separable from one another in the former two. By separable we mean that the loss functions can be written in two distinct parts, the first only considering positive distances, and the second considering only negative distances.

In Equations 7 through 10, we demonstrate how to separate the positive and negative distances in SupCon loss.

$$L_{supcon}(x_a, f) = \frac{-1}{|P(x_a)|} \sum_{x_p \in P(x_a)} \log \frac{\exp(z_a \cdot z_p/\tau)}{\sum_{x_n \in N(x_a)} \exp(z_a \cdot z_n/\tau)} \tag{7}$$

$$= \frac{-1}{|P(x_a)|} \sum_{x_p \in P(x_a)} \left[\log\left(\exp(z_a \cdot z_p/\tau)\right) - \log \sum_{x_n \in N(x_a)} \exp(z_a \cdot z_n/\tau) \right] \tag{8}$$

$$= \frac{-1}{|P(x_a)|} \sum_{x_p \in P(x_a)} z_a \cdot z_p/\tau + \frac{1}{|P(x_a)|} \sum_{x_p \in P(x_a)} \log \sum_{x_n \in N(x)} \exp(z_a \cdot z_n/\tau) \tag{9}$$

$$= \log \sum_{x_n \in N(x_a)} \exp(z_a \cdot z_n/\tau) - \frac{1}{|P(x_a)|\,\tau} \sum_{x_p \in P(x_a)} z_a \cdot z_p \tag{10}$$

This formulation of the SupCon loss function looks very similar to the pairwise contrastive loss. In this form, the average positive similarity is subtracted from the weighted average of negative pair similarities. In pairwise contrastive loss, the average positive distance is added to the average negative distance.

Our early experiments indicated that triplet loss performed better than pairwise contrastive loss. The authors of Khosla et al. [13] consider SupCon loss as a generalization of triplet loss; however, the triplet loss that they consider is different from the one used in this study (the hinge term is absent there). The current portion of this study is dedicated to comparing the performance of different contrastive loss functions directly to one another.

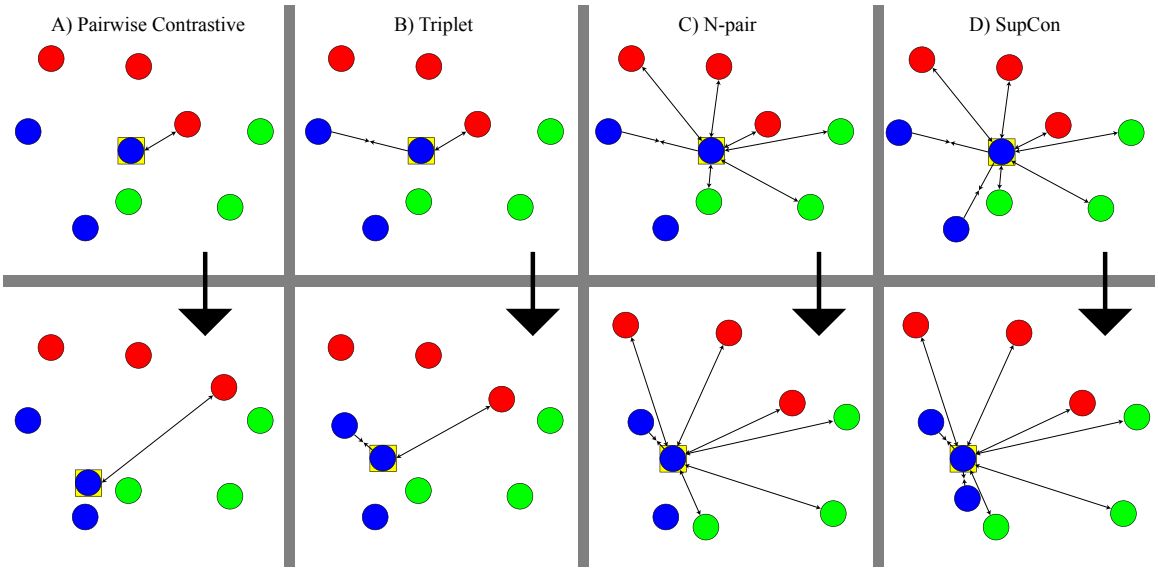

Figure 8: Demonstration of how different loss functions approach contrasting embeddings: A) Pairwise contrastive loss selects a red and a blue object then pushes them apart, B) Triplet loss selects two blue objects and moves them closer, but further from a red object, C) N-pair loss selects two blue objects and moves them closer together while moving them away from all red and green objects, D) SupCon loss selects all blue objects and moves them closer together while moving all the red and green objects away from them.

3.1.5 RESULTS

For each loss function, we trained 100 models. Each model was trained on a randomly generated training and validation dataset, each comprised of 10 images of each MNIST digit. Models were then evaluated by their accuracy with a 1-nearest-neighbor classifier on randomly generated testing data of 1000 MNIST images. Our study used the same model architecture for all three loss functions, and computed distances via cosine similarity.

In Figure 9, triplet loss performs better than both pairwise contrastive loss, and SupCon loss. In Section 3.1.4, we showed that the pairwise contrastive loss function and the SupCon loss function are more similar to one another than they are to triplet loss. This finding is supported by similar results obtained from models trained via pairwise contrastive loss and models trained via SupCon loss.

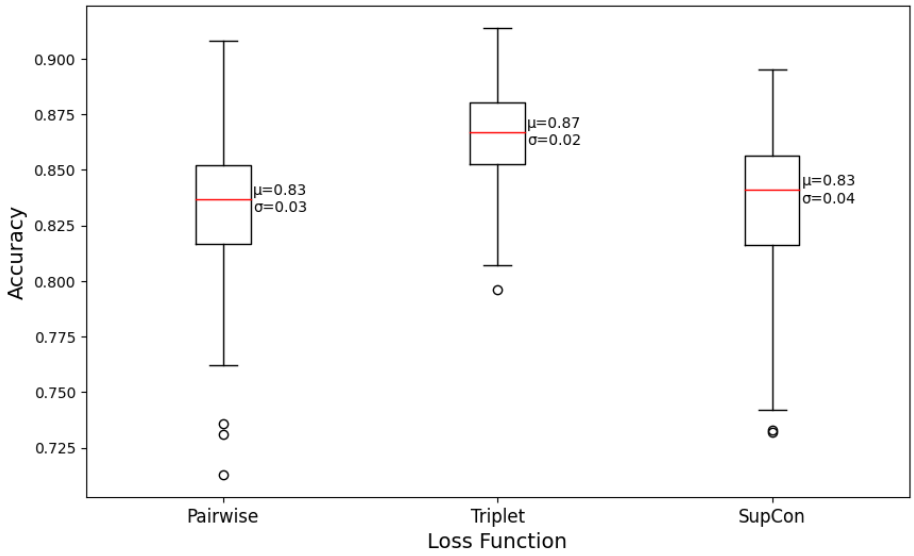

Figure 9: Boxplot depicting 1-nearest neighbor accuracies obtained by networks trained with three different contrastive loss functions. One hundred models were trained with each loss function. The mean and standard deviations of the accuracies from each model are displayed next to their corresponding boxplot. Triplet loss performed better than pairwise contrastive loss and SupCon loss. The latter two performed similarly, consistent with our derivation in Equation 7.

3.2 METRIC FUNCTIONS

In contrastive learning, we learn an embedding function, f, which maps our dataset from its native domain, $\mathcal{X}$, to an embedding space, $\mathcal{Z}$. We choose a metric function, d, on our embedding space which computes the distances between embedded objects $z_* = f(x_*) \in \mathcal{Z}$. We then optimize a contrastive loss function (Definition 1.2.3) with respect to the parameters of our embedding function, f, in hopes of decreasing positive distances and increasing negative distances computed by the metric, d.

3.2.1 DISTANCE AND DISSIMILARITY

Notably, some metric functions are bounded, while others are not. Therefore we separate metric functions into two distinct categories, distance functions and dissimilarity functions.

Definition 18 (Distance Function, Dissimilarity Function)**.**

A distance function is an unbounded metric function.

A dissimilarity function is a bounded metric function.

This notation is not a common convention, but adopting it will allow greater clarity in communicating our findings.

In contrastive learning, if a loss function, such as pairwise contrastive loss (Definition 13) is used in conjunction with a distance function, then the loss function can be infinitely decreased by infinitely increasing the distances between all embedded objects, including object from the same class. This is a problem because the purpose of contrastive learning is to render objects only from different classes distant while keeping objects from the same class close. The solution to this problem is to introduce a "hinge term".

We pointed out the hinge term in Section 3.1 when describing the pairwise contrastive loss (Definition 13) and triplet loss (Definition 15). The hinge term in both loss functions serves the purpose of setting a maximum attainable negative distance. For example, in pairwise contrastive loss, if the hinge term, $\max(m^* - d(z_a, z_n), 0)$, were removed in favor of using the unaltered negative distance, $-d(z_a, z_n)$, then the pairwise contrastive loss could be reduced by increasing negative distances infinitely instead of ever decreasing positive distances. By bounding the negative distance, the hinge term effectively renders any choice metric function as a dissimilarity function.

There are theoretical and philosophical justifications for the use of dissimilarity functions over distance functions. On the theoretical side, consider the use of a distance function in pairwise contrastive loss. The contrastive network could update the embedding function every epoch to increase the distance between every object –including positive pairs– while decreasing the loss function. The loss function would have no minimum, and the embedding function would practically be useless.

Philosophically, it makes little sense to allow embedded objects to become infinitely distant. In contrastive learning, our goal is to learn an embedding function that,

along with our metric function, allows us to characterize the similarity/dissimilarity between objects. There is an obvious case in which two objects are maximally similar, which is when they are the same. This is reflected in a metric function by the reflexive property $d(x, x) = 0$. The difference between using a distance function and a dissimilarity function is equivalent to the philosophical question of whether objects can be maximally similar, or whether, for a given object, increasingly dissimilar objects can be found ad infinitum.

It is the opinion of the author that there exists a notion of maximal dissimilarity, which corresponds to the use of a dissimilarity function. For example, in comparing images of handwritten digits from the MNIST dataset, any two images of different digits should be considered maximally dissimilar. Therefore, the use of dissimilarity functions, over distance functions, is rationally justified.

Different metric functions can and have been used in all types of deep learning, and indeed within contrastive learning. In this experiment, we analyzed the use of five metric functions for contrastive loss.

3.2.2 COSINE DISSIMILARITY

Cosine dissimilarity (usually referred to as cosine distance), is a semimetric function. It is popular within deep learning and has the property that it is easy to compute.

Definition 19 (Cosine Dissimilarity)**.**

$$d_{cosine}(z_1, z_2) = 1 - \frac{z_1 \cdot z_2}{\|z_1\|\|z_2\|}. \tag{11}$$

Cosine dissimilarity forms the semimetric space $(\mathbb{S}^n, d_{cosine})$.

The cosine dissimilarity is equivalent to one minus the cosine similarity of two vectors. One drawback to cosine distance is that fails to satisfy the triangle inequality, therefore it is not a formal metric function.

3.2.3 ANGULAR DISSIMILARITY

Definition 20 (Angular Dissimilarity)**.**

$$\begin{aligned} d_{angular}(z_1, z_2) &= \frac{\arccos(1 - d_{cos}(z_1, z_2))}{\pi} \\ &= \frac{\arccos\left(\frac{z_1 \cdot z_2}{\|z_1\|\|z_2\|}\right)}{\pi}. \end{aligned} \tag{12}$$

Angular dissimilarity forms the metric space $(\mathbb{S}^n, d_{angular})$

The angular dissimilarity (commonly referred to as angular distance) is the arccosine of the cosine similarity between two vectors, divided by π. The angular distance satisfies the triangle inequality which makes it a formal metric function.

3.2.4 CHEBYSHEV DISTANCE

The Chebyshev distance between two vectors is their greatest dimension-wise difference.

Definition 21 (Chebyshev Distance)**.**

$$d_{Chebyshev}(z_1, z_2) = \max_i(|z_{1,i} - z_{2,i}|) \tag{13}$$

where $z_{*,i}$ refers to the i^{th} element of the vector z_*.

The Chebyshev distance forms the metric space $(\mathbb{R}^n, d_{Chebyshev})$ and is some times written as $\|z_1 - z_2\|_\infty$.

3.2.5 EUCLIDEAN DISTANCE

Definition 22 (Euclidean Distance)**.**

$$d_{Euclidean}(z_1, z_2) = \sqrt{\sum_{i=1}(z_{1,i} - z_{2,i})^2} \tag{14}$$

where $z_{*,i}$ refers to the i^{th} element of the vector z_*.

The Euclidean distance forms the metric space $(\mathbb{R}^n, d_{Euclidean})$ and is usually times written as $\|z_1 - z_2\|_2$.

3.2.6 ARCTANGENT DISSIMILARITY

The arctangent dissimilarity is novel and created for experimental purposes. It is an extension of the Euclidean distance.

Definition 23 (Arctangent Dissimilarity)**.**

$$\begin{aligned} d_{arctan}(z_1, z_2) &= \frac{2}{\pi} \arctan(d_{Euclidean}(z_1, z_2)) \\ &= \frac{2}{\pi} \arctan\left(\sqrt{\sum_{i=1}^{n} (z_{1,i} - z_{2,i})^2}\right). \end{aligned} \tag{15}$$

The arctangent dissimilarity forms the metric space $(\mathbb{R}^n, d_{arctan})$ (see Remark 1).

The arctangent function is used to bound the Euclidean distance below $\frac{\pi}{2}$, which is then scaled by $\frac{2}{\pi}$ to bound the function on $[0, 1)$.

Remark 1. Suppose $a, b, c \in \mathbb{R}^+$ such that $a + b \geq c$. We want to show that $\arctan(a) + \arctan(b) \geq \arctan(c)$.

Proof. Suppose $a, b \in \mathbb{R}^+$. The derivative of the arctangent function,

$$\frac{d \arctan}{da} = \frac{1}{1 + a^2},$$

is positive for all $a \geq 0$. Therefore the arctangent function is strictly increasing for $a \geq 0$.

Define $g(a, b) = \arctan(a) + \arctan(b) - \arctan(a + b)$. Then

$$\frac{\partial g}{\partial a} = \frac{1}{1 + a^2} - \frac{1}{1 + (a + b)^2} \tag{16}$$

$$\geq 0 \tag{17}$$

and in the same manner $\frac{\partial g}{\partial b} \geq 0$.

Also, for $g(0,0) = \arctan(0) + \arctan(0) - \arctan(0+0) = 0$. Because $g(a,b)$ equals 0 at $a = b = 0$, and is nondecreasing as a and b increase, we have $\forall a, b \in \mathbb{R}^+$, $g(a,b) \geq 0$.

Now suppose $c \in \mathbb{R}^+$ such that $c \leq a + b$. We can show

$$\arctan(a) + \arctan(b) - \arctan(c) \geq \arctan(a) + \arctan(b) - \arctan(a+b)$$

because the arctangent function is a nondecreasing function. Then

$$\arctan(a) + \arctan(b) - \arctan(c) \geq g(a,b) \tag{18}$$

$$\geq 0 \tag{19}$$

$$\implies \arctan(a) + \arctan(b) \geq \arctan(c) \tag{20}$$

Therefore we have shown that if $a, b, c \in \mathbb{R}^+$ and $a + b \geq c$, then $\arctan(a) + \arctan(b) \geq \arctan(c)$. □

3.2.7 RESULTS

For each metric function, we trained 100 models, each on a randomly generated training dataset and validation dataset, both comprised of 10 images of each MNIST digit. Models were evaluated by their accuracy with a 1-nearest-neighbor classifier on randomly generated testing data of 1000 MNIST images. Our study used the same model architecture for all five metric functions, and all models were trained with the triplet loss function.

Figure 10 shows that most all metrics performed similarly except for the Chebyshev distance, which was slightly worse. We theorize that the choice of metric function has little to no effect on the accuracy of a model because the network is capable of learning a change of metric.

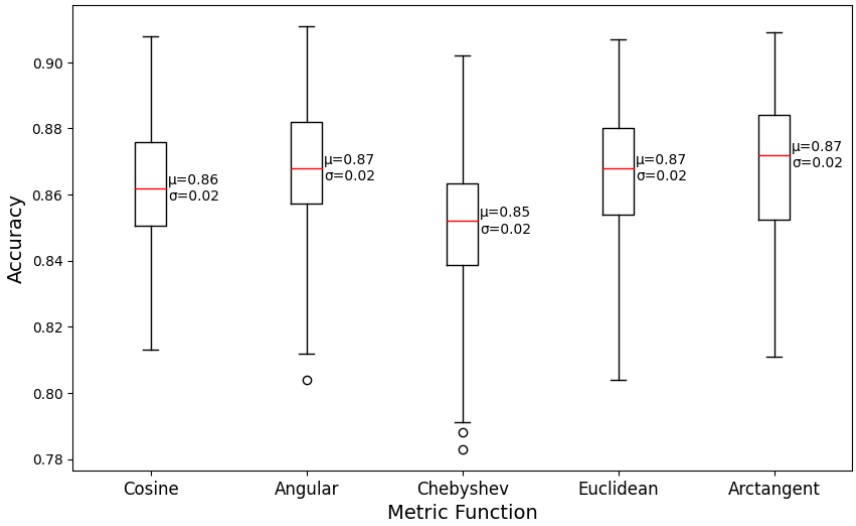

Figure 10: Boxplot depicting 1-nearest neighbor accuracies obtained by networks trained with five different metric functions. One hundred models were trained with each metric function. The mean and standard deviations of the accuracies from each metric function are displayed next to their corresponding boxplot. All metric functions performed similarly, with Chebyshev distance performing slightly worse than the other four metric functions.

Suppose we have a contrastive network $f_{base} : \mathcal{X} \to \mathcal{Z}_a$ which was trained on a metric $d_1 : \mathcal{Z}_a^2 \to \mathbb{R}$. Let $x_1, x_2 \in \mathcal{X}$ and denote their embeddings $z_1 = f_{base}(x_1)$ and $z_2 = f_{base}(x_2)$. Now, suppose we have a second metric, $d_2 : \mathcal{Z}_b^2 \to \mathbb{R}$. We theorize that it is possible to learn a neural network $f_{shift} : \mathcal{Z}_a \to \mathcal{Z}_b$ such that

$$d_1(z_1, z_2) \approx d_2(f_{shift}(z_1), f_{shift}(z_2)).$$

To us, this statement would imply that the choice of metric does not matter to the contrastive network, because it can learn to incorporate a change of metric into itself.

3.3 MARGIN VALUE

In Section 3.2.1, we explored what we called the hinge term, which is used in some contrastive loss functions to set a maximum negative distance. This is to prevent a contrastive network from increasing negative distance while neglecting positive distances.

The margin value m^* within the hinge term serves as this maximum negative distance (see Equations 13 and 15).

The implementation of the hinge term into contrastive loss functions can be straightforward, as in pairwise contrastive loss, where the maximum distance is set to m^*. In the case of triplet loss, implementation is slightly more nuanced, and the maximum negative distance is defined with respect to a triplet, as $d(x_a, x_p) + m^*$ (the positive distance plus the margin value) which will converge to m^* as the triplet loss converges to 0.

If a dissimilarity function is used as a metric, the use of a hinge is no longer necessary, as there is already a maximum distance. However, hinge terms are often used regardless.

The question addressed in this section is whether the specific value of the margin, m^* has any effect on model accuracy. To this end, we trained models with margin values ranging from 0.000 to 0.995 at increments of 0.005, for a total of 200 models. Each model was trained on randomly generated training and validation data, both comprised of 10 images of each MNIST digit. Models were evaluated by their accuracy with a 1-nearest-neighbor classifier on randomly generated testing data of 1000 MNIST images. Our study used the same model architecture for all five metric functions, and all models were trained with the triplet loss function and cosine dissimilarity.

Figure 11 indicates that the value of m^* does not have an obvious effect on the accuracy of the model. This is in some ways unsurprising, considering that for two models using different values of m^*, the calculated dissimilarities they produced can be scaled to the same range (see Figure 12).

When the dimension of the embedding space is small relative to the number of classes, the values of m^* will, in turn, have to be sufficiently small for the model to be capable of learning a function with a desirable property of specific-class invariance (see Definition 25). For example, if a dataset has 5 classes, and a model uses cosine

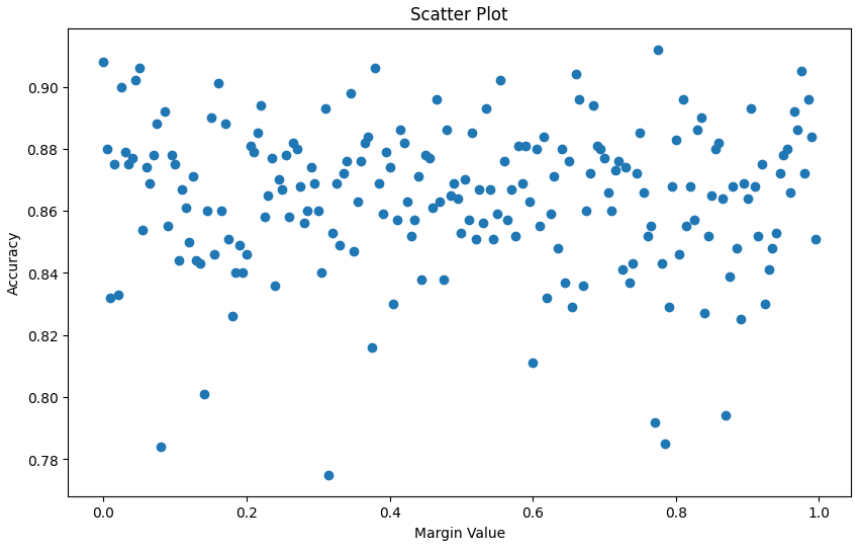

Figure 11: Scatterplot of model accuracies obtained by training contrastive networks on a multitude of margin values, spanning from 0.000 to 0.995

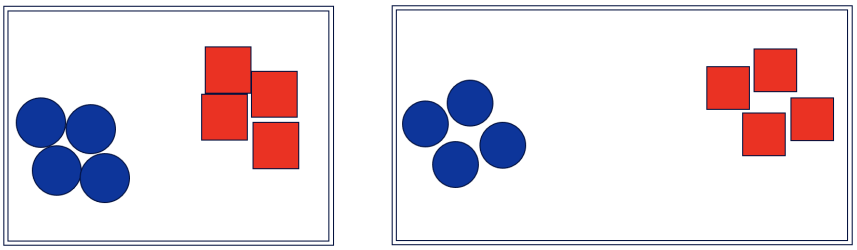

Figure 12: Examples of learned embeddings from contrastive networks with different margin values m^*. In this example, there are only two classes, blue circles and red squares. The network which produced the embeddings on the left had the smaller margin value and thus the two classes are not as distant as they are in the embeddings on the right. However, the two classes were already distinguishable in the embeddings on the left, and pushing them further apart does not provide any benefit.

dissimilarity in $\mathbb{R}^2$ with no margin value, there is simply not enough room in the embedding space to learn an embedding function, f, which negative pairs to be maximally dissimilar.

4 INVARIANCE PROPERTIES

The study of geometry has contributed to an unimaginably vast array of human achievements. From Einstein's general theory of relativity in physics to the use of perspective and proportions in the art of Leonardo da Vinci, geometry has been involved in many human activities. One of its most recent applications has been to deep learning, in a new field referred to as geometric deep learning. The principle object of importance in geometric deep learning is identifying the symmetries of a given dataset.

Deep learning can be informally defined as the use of deep neural networks to obtain an approximation, $\hat{f} : \mathcal{X} \to \mathcal{Z}$, of a target function, $f : \mathcal{X} \to \mathcal{Z}$, by optimization of a loss function $L(f(S), \hat{f}(S;\theta))$ with respect to the parameters $\theta \in \Theta$. The process of training the network can be understood as traversing a function space $\mathcal{F} = \{\hat{f}(\cdot;\theta) : \forall\theta \in \Theta\}$ in search of the best approximation $\hat{f}$ of f, using the training dataset, S_{train} as fuel for the journey (note the target function, f, may not be unique).

Deep neural networks are universal approximators of continuous functions [11], which makes the function space $\mathcal{F}$ massive, often too massive to traverse with given training data. Making progress in the face of this issue is the fundamental concern of few-shot learning. By leveraging symmetries, invariances, and equivariances, geometric deep learning provides a method to approach few-shot learning problems.

Definition 24 (Symmetry, Invariance, Equivariance)**.**

If an object $x \in \mathcal{X}$ possesses a property p, then a symmetry of x is to be defined as a transformation $g : \mathcal{X} \to \mathcal{X}$ such that $g(x)$ somehow preserves property p.

A function $f : \mathcal{X} \to \mathcal{Z}$ is invariant under symmetry g if $f(g(x)) = f(x)$.

A function $f : \mathcal{X} \to \mathcal{Z}$ is equivariant under symmetry g if $f(g(x)) = h(f(x))$ for some $h : \mathcal{Z} \to \mathcal{Z}$ which represents the action of g on a different space.

Suppose objects $x \in \mathcal{X}$ are images, and let g_{left} be the transformation of shifting an image leftward. Then if x is an image of a cat, the transformation g_{left} does not affect that the image is of a cat. Thus function, f, which classifies objects as a cat must be invariant to that symmetry of shifting.

For machine learning practitioners, large-scale image classification has historically been "notoriously difficult" [4, Sec 1.2.5]. The success of convolutional neural networks brought a massive performance increase in image classification among other areas of deep learning. The use of convolutional networks has geometric justifications: they are made of shift-equivariant convolutional layers and an eventual global pooling layer which makes them shift-invariant. A dense neural network could learn to become shift-invariant with masses of pixel-shifted data, but this wonderful solution is hard to find amidst the network's vast function space.

In summary, for deep-learning image classification, the target function, f, possesses the property of shift-invariance. We recognize this and restrict the function space –to convolutional networks– accordingly. When the function space is reduced, less data is needed to reach a sufficient approximator [1]. Until the recent success of transformers, almost all state-of-the-art image classification models were convolutional networks [7, 23].

For deep learning applications other than image classification, a dataset may possess symmetries under transformations other than shifts. The target function for any application should reflect the symmetries of its respective dataset through invariance or equivariance. The formal characterization of equivariance and invariance properties implicit in contrastive learning could lead to similarly large increases in model performance and adoption.

Suppose we want a contrastive network, $\hat{f} : \mathcal{X} \to \mathcal{Z}$, which can be used with a

dissimilarity function, $d : \mathcal{Z}^2 \to [0, m^*]$, to determine the dissimilarities between fingerprints. If we consider all pairs of fingerprints from the same person symmetric and all pairs of fingerprints from different people symmetric, then the target function, f, when used with d, possesses a couple of interesting invariance properties that we would want to be mimicked by our contrastive network.

Definition 25 (Specific-Object Invariance, Specific-Class Invariance)**.**

Let $\mathcal{C}$ be the set of all classes, then for all $C_i, C_j \in \mathcal{C}$ such that $C_i \neq C_j$, and $x_1, x_2 \in C_i$, $x_3 \in C_j$.

$$\text{Specific-Object Invariance (SOI): } d(z_1, z_2) = 0 \tag{21}$$

$$\text{Specific-Class Invariance (SCI): } d(z_1, z_3) = m \tag{22}$$

for some $m \in \mathbb{R}$.

Recall embeddings are denoted as $z_* = f(x_*)$.

Specifc-object invariance refers to the distance between embeddings from the same class being invariant under the transformation of swapping objects out for different objects within the same class. Specific-class invariance refers to the distances between embeddings from different classes being invariant under the transformation of selecting a different class.

In the fingerprint example, the distances between embeddings from our contrastive network, $\hat{f}$, would ideally possess both of these invariance properties. This means that any two fingerprints from the same person are embedded to the same point, and the distance between the embeddings of any fingerprints from any two different people is the same.

These properties also apply to the target function to be learned by cross-entropy loss. Recall the illustration of the embeddings of a cross-entropy network in Figure 2. In this example, the cross-entropy loss (Definition 9) is minimized when all images of cats, x_c, are mapped as $z_c = (0, 1)$ and all images of dogs, x_d, are embedded as $z_d = (1, 0)$.

Therefore the target function of a cross-entropy network satisfies these invariance properties with respect to Euclidean distance (Definition 22).

Specific-object invariance describes the property of the target function mapping all objects from the same class to the same point. Such a function would not be injective, and therefore limited in usefulness. We can instead aim for specific-object stability, which requires only that all objects from the same class be within a certain distance, c_0, from one another.

Definition 26 (Specific-Object Stability)**.**

Let $\mathcal{C}$ be the set of all classes, then for all $C_i \in \mathcal{C}$, for $x_1, x_2 \in C_i$, and $c_0 \in \mathbb{R}^+$.

$$\text{Specific-Object Stability (SOS): } 0 \leq d(z_1, z_2) \leq c_0. \tag{23}$$

Specific-class invariance describes the property of the target function which retains equal distance between the embeddings of any two objects from different classes. Likewise to the specific-object case, we propose to seek specific-class stability, rather than invariance.

Definition 27 (Specific-Class Stability)**.**

Let $\mathcal{C}$ be the set of all classes, then for all $C_i, C_j \in \mathcal{C}$ such that $C_i \neq C_j$, for $x_1 \in C_i$, $x_2 \in C_j$, and $c_1, m \in \mathbb{R}^+$.

$$\text{Specific-Class Stability (SCS): } m - c_1 \leq d(z_1, z_2) \leq m. \tag{24}$$

If we view our objects as signals, using a notion of stability instead of invariance can account for signal deformations [3, Sec. 3.3].

Describing the desired invariance/stability properties for contrastive networks is the first step in designing a network that enforces them. This is just as in the case of

convolutional networks being designed to enforce translation invariance. Unlike the case of convolutional networks, there is no obvious network architecture that possesses these properties.

Further analysis of these questions could lead to influential results for the field of deep learning, especially within the subset of few-shot learning. Because many forensic problems must grapple with small datasets, a greater understanding of these algorithms may prove unequivocally useful to forensic scientists.

5 APPLICATION TO FORENSIC BALLISTIC TOOLMARK DATA

5.1 NBIDE DATASET

The NBIDE (NIST Ballistics Imaging Database Evaluation) was created for the study Vorburger et al. [27]. Within the NBIDE dataset, there are 144 images of the backs of fired cartridge casings. When a gun fires a cartridge, it leaves impressions on the casing. The firing pin impression is caused by the firing pin striking the primer on the casing. The breech face impression is caused by the cartridge being forced back against the breech face of the gun. The ejector mark is caused by a gun's ejector mechanism which dispels a fired cartridge casing (see Figure 13 for reference).

The NBIDE study used twelve 9mm handguns. Each gun was one of three models; each model contributed four guns; and each gun fired twelve cartridges. The fired cartridges were one of four models, and each gun fired each model of cartridges three times. This amounts to a total of 144 cartridges fired, and thus 144 cartridges casings to analyze. However, the NBIDE study only analyzed the cartridges of 3 of the brands of cartridges, for a reduced total of 108 cartridge casings. To maintain a direct comparison, we omitted the same 36 cartridges from our study. Images were taken of the back of all fired cartridges, as well as 3-D topography scans. In this work, we focus on the images and do not analyze the scans.

In the NBIDE study, casing images were compared using the I-2D correlation.

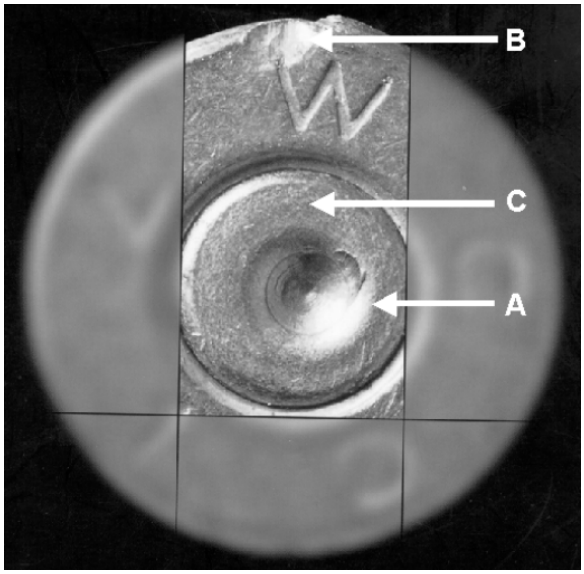

Figure 13: Image showing the A) firing pin impression, B) ejector mark, and C) breech face impression. Photo from Forensic Technology, Inc. [8].

Definition 28 (I-2D Correlation)**.**

The I-2D correlation between two images is computed by an Integrated Ballistic Identification System (IBIS) using BrassCatcher Software Version 3.4.5.

IBIS is a proprietary technology developed by Forensic Technology.

Before computation for the I-2D correlations between images of casings, the images were preprocessed into three regions corresponding to the three impressions on the casing (see Figure 13). The I-2D correlations between images were computed separately for each of the three regions of the images. The NBIDE study found the region containing the breech face impression to be the most powerful in identifying the handgun that left the impression.

In our study, we cropped the images down to square regions roughly encompassing both the breech face region and the firing pin region. Part of the motivation for applying deep learning to forensic identification of source problems is to reduce the amount of human preprocessing, instead delegating this task to the neural network –a purely mathematical procedure.

5.2 RESULTS

The NBIDE used a standard [27, Sec. 1] measure of performance for their method, called the "TopTen" accuracy. For each x_u in the set S of 108 analyzed cartridges, they computed the I-2D correlation between x_u and the remaining 107.

Definition 29 (TopTen Accuracy)**.**

For $x_u \in S$ with label y_u, let S_{top} be the set of the 10 $x_* \in S \setminus \{x_u\}$ which are most similar to x_u. For $x_i \in S_{top}$, let y_i be its class label.

$$\text{TopTen} = \frac{1}{|S|} \sum_{x_u \in S} \sum_{x_i \in S_{top}} \mathbf{1}_{y_u}(y_i) \tag{25}$$

where $\mathbf{1}_{y_u}$ is the indicator function of label y_u.

For example, in the NBIDE study using the I-2D correlation, the inner sum represents, for a given cartridge casing image, x_u, fired from gun, y_u, and the 10 cartridge casing images, S_{top}, which have the highest I-2D correlation with x_u, the number of $x_* \in S_{top}$ which were also fired from gun y_u. The TopTen accuracy is therefore the average across all $x_u \in S$.

For our tests, we built contrastive networks by using the DenseNet121 [12] architecture, pre-trained on the ImageNet database with its weights frozen. On top, we added a convolutional layer and a densely connected layer.

For each $x_u \in S$, we created a set S_{train} with 8 cartridges from each gun, and a set $S_{validation}$ with the remaining 11 cartridges. We then trained a contrastive network with triplet loss on S_{train}. At the end of every epoch we computed the average TopTen accuracy for $x_v \in S_{validation}$. We ended training when the average TopTen accuracy for the validation data stopped improving. Finally, we used the model to procure the 10 $x_* \in S_{train} \cup S_{validation}$ most similar to x_u. By repeating this process for each $x_u \in S$, we attained our average TopTen score.

Table 1: Table showing the "TopTen" and Top-1 accuracies of the three analyzed approaches. No Top-1 accuracy was provided for the I-2D correlation.

	I-2D Correlation	Contrastive	KNN
TopTen	5.57	4.28	2.11
Top-1	N/A	0.45	0.49

We also computed the Top-1 accuracy for our contrastive network, which is a more popular measure of accuracy in deep learning image classification. The Top-N accuracy of a model is the probability that the true class of y_u of x_u is one of the top N predictions of class by the model. Thus the Top-1 accuracy is a model's accuracy in correctly classifying an object.

We used the 1-nearest neighbor classifier to perform classification on the embeddings $z_* = f(x_*)$ obtained from our contrastive network. We also computed the 1-nearest neighbor classifier on the raw images for comparison. Future work can compare itself to these methods by the Top-1 accuracy.

The results of our experiment are summarized in Table 1. The I-2D correlation method outperformed the contrastive network, and both outperformed the 1NN classifier on the raw image data. Interestingly, the 1NN classifier slightly outperformed the contrastive network in Top-1 accuracy.

Although the contrastive network failed to outperform the I-2D correlation, the result is still promising for the technology. The first part of our contrastive network, DenseNet121, is pre-trained on the ImageNet dataset of images with classes such as "vending machine" and "bald eagle". DenseNet121 serves as a sort of preprocessing phase of feature extraction. It is likely the features that DenseNet121 selects are not well-suited for analyzing impressions on cartridge casings. The result obtained using DenseNet121 could probably be improved by instead using a network pre-trained on a larger corpus of cartridge casings besides the NBIDE data.

DenseNet121 also required the images to be compressed from $1500 * 1500$ pixels

(their size after cropping) to $224 * 224$ pixels. Because the casing impressions are very finely detailed, it may be that valuable information is lost during this compression. This further motivates the use of a novel pre-trained network, one trained on similar data and with a larger input size.

Furthermore, our analysis failed to account for rotations in the images. It is common to perform data augmentation in deep learning, especially in few-shot learning problems [28, Sec. 3]. Future work could improve this result by rotating the training images during training. Alternatively, as in Section 4, we could attempt to construct a network invariant to image rotations.

6 CONCLUSION

In this work, we focused on the use of contrastive networks for classification problems in forensic sciences. Our analysis showed that neural networks trained by cross-entropy loss slightly outperformed contrastive networks for few-shot classification.

To support our experiments, we found it necessary to unify some of the disparate terms used across the contrastive learning literature. We hope that this unification will be useful for future researchers in the field. We analyzed the individual effect on model accuracy of different contrastive loss functions, metric functions, and margin values. We found triplet loss to be the best-performing loss function, and that the metric function and margin value have little to no effect on accuracy.

The embedding functions learned from contrastive networks ideally exhibit certain invariance properties concerning the class of objects. We detailed these properties with the belief that further examination could provide insight to train contrastive networks more efficiently and with better results.

Because they are an auspicious method for few-shot learning and verification, contrastive networks can be used for many forensic identification of source problems. We found contrastive learning to be competitive with other statistical methods for the task of

identification of the source firearm for a fired cartridge casing. Furthermore, our methods spent significantly less time on data preprocessing and did not account for rotations of cartridge casing images. We also did not perform batch-mining in this study, which has been found to improve accuracies. We believe that by tending to these factors, future work may be able to achieve state-of-the-art performance.

REFERENCES

[1] Bietti, A., Venturi, L. and Bruna, J. [2021], 'On the sample complexity of learning under geometric stability', *Advances in neural information processing systems* **34**, 18673–18684.

[2] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. and Shah, R. [1993], Signature verification using a "siamese" time delay neural network, *in* 'Proceedings of the 6th International Conference on Neural Information Processing Systems', NIPS'93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 737â744.

[3] Bronstein, M. M., Bruna, J., Cohen, T. and Veličković, P. [2021], 'Geometric deep learning: Grids, groups, graphs, geodesics, and gauges', *arXiv preprint arXiv:2104.13478* .

[4] Chollet, F. [2021], *Deep learning with Python*, Simon and Schuster.

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

[6] De Sa, V. [1993], 'Learning classification with unlabeled data', *Advances in neural information processing systems* **6**.

[7] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. [2020], 'An image is worth 16x16 words: Transformers for image recognition at scale', *arXiv preprint arXiv:2010.11929* .

[8] Forensic Technology, Inc. [n.d.], 'Segmenting tool mark image reference files (tmirf) to identify crime guns more effectively', `http://www.forensictechnology.com/d4.html`. Accessed: 16 April 2007.

[9] Hénaff, O. J., Srinivas, A., Fauw, J. D., Razavi, A., Doersch, C., Eslami, S. M. A. and van den Oord, A. [2019], 'Data-efficient image recognition with contrastive predictive coding', *CoRR* **abs/1905.09272**.
URL: *http://arxiv.org/abs/1905.09272*

[10] Hermans, A., Beyer, L. and Leibe, B. [2017], 'In defense of the triplet loss for person re-identification', *arXiv preprint arXiv:1703.07737* .

[11] Hornik, K. [1991], 'Approximation capabilities of multilayer feedforward networks', *Neural networks* **4**(2), 251–257.

[12] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. [2017], Densely connected convolutional networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4700–4708.

[13] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C. and Krishnan, D. [2020], 'Supervised contrastive learning', *Advances in neural information processing systems* .

[14] Li, A., Luo, T., Xiang, T., Huang, W. and Wang, L. [2019], Few-shot learning with global class representations, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 9715–9724.

[15] Masci, J., Bronstein, M. M., Bronstein, A. M. and Schmidhuber, J. [2012], 'Multimodal similarity-preserving hashing', *CoRR* **abs/1207.1522**.
URL: *http://arxiv.org/abs/1207.1522*

[16] Mitrovic, J., McWilliams, B. and Rey, M. [2020], 'Less can be more in contrastive learning'.

[17] Mookiah, M. R. K., Puch-Solis, R. and Daeid, N. N. [2023], 'Identification of bullets fired from air guns using machine and deep learning methods', *Forensic science international* **349**, 111734.

[18] National Institute of Standards and Technology [n.d.], 'Forensic science'.
URL: *https://www.nist.gov/forensic-science*

[19] Ommen, D. M. and Saunders, C. P. [2018], 'Building a unified statistical framework for the forensic identification of source problems', *Law, Probability and Risk* **17**(2), 179–197.

[20] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. et al. [2021], Learning transferable visual models from natural language supervision, *in* 'International conference on machine learning', PMLR, pp. 8748–8763.

[21] Schroff, F., Kalenichenko, D. and Philbin, J. [2015], Facenet: A unified embedding for face recognition and clustering, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 815–823.

[22] Sohn, K. [2016], 'Improved deep metric learning with multi-class n-pair loss objective', *Advances in neural information processing systems* **29**.

[23] Srivastava, S. and Sharma, G. [2024], Omnivec: Learning robust representations with cross modal sharing, *in* 'Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision', pp. 1236–1248.

[24] Sun, Y., Chen, Y., Wang, X. and Tang, X. [2014], 'Deep learning face representation by joint identification-verification', *Advances in neural information processing systems* **27**.

[25] Tian, Y., Chen, X. and Ganguli, S. [2021], Understanding self-supervised learning dynamics without contrastive pairs, *in* 'International Conference on Machine Learning', PMLR, pp. 10268–10278.

[26] van den Oord, A., Li, Y. and Vinyals, O. [2018], 'Representation learning with contrastive predictive coding', *CoRR* $\mathbf{abs/1807.03748}$.
URL: *http://arxiv.org/abs/1807.03748*

[27] Vorburger, T. V., Yen, J. H., Bachrach, B., Renegar, T. B., Ma, L., Rhee, H.-G., Zheng, X. A., Song, J.-F. and Foreman, C. D. [2007], 'Surface topography analysis for a feasibility assessment of a national ballistics imaging database'.

[28] Wang, Y., Yao, Q., Kwok, J. T. and Ni, L. M. [2020], 'Generalizing from a few examples: A survey on few-shot learning', *ACM computing surveys (csur)* $\mathbf{53}$(3), 1–34.

[29] Wang, Z., Zhao, Y., Huang, H., Liu, J., Yin, A., Tang, L., Li, L., Wang, Y., Zhang, Z. and Zhao, Z. [2024], 'Connecting multi-modal contrastive representations', *Advances in Neural Information Processing Systems* $\mathbf{36}$.

[30] Weinberger, K. Q., Blitzer, J. and Saul, L. [2005], 'Distance metric learning for large margin nearest neighbor classification', *Advances in neural information processing systems* $\mathbf{18}$.

[31] Wikipedia contributors [2024], 'Mnist database — Wikipedia, the free encyclopedia'. [Online; accessed 23-April-2024].
URL:
https://en.wikipedia.org/w/index.php?title=MNIST$_d atabaseoldid = 1216945316$

[32] Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M. and Wu, Y. [2022], 'Coca: Contrastive captioners are image-text foundation models', *arXiv preprint arXiv:2205.01917* .