Adversarial Robustness of Deep Metric Learning

Final Project Report

Abstract

Adversarial attacks have the capability of perturbing an input on a minute level, which subsequently causes a trained model to severely misclassify. It is therefore an intriguing and vital branch of research within the field to understand how to create models which have better defense against these mechanisms. In this study, we compare the adversarial robustness of models trained with Deep Learning and Deep Metric Learning methodologies, employing a range of differing parameters; such as, the loss function, chosen optimisation algorithm, distance metric, amongst other key variables. The datasets employed were MNIST, Fashion-MNIST and CIFAR-10. Following the implementation and analysis of results, the study finds that Deep Metric Learning is superior to Deep Learning in creating models with adversarial robustness.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 15,000 words.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The creation of Machine Learning models that are robust against adversarial attacks is of vital importance. Adversarial examples pose significant implications to a model's security, reliability and performance within various domains such as autonomous vehicles and the medical field. Given a set of image data and an image x belonging to class c_1 where a trained model assigns x to c_1, an adversarial attack can produce a perturbed sample $x+\delta$ which the model assigns to all classes but c_1. This provides a perturbation so minute that the human eye cannot differentiate between x and $x + \delta$, as illustrated in Figure 1.1. [2, 57].

(a) No Perturbation

(b) Perturbed Sample

Figure 1.1: Normal and Perturbed Image of Frog, PGD Attack with $\epsilon = 0.01$

Subsequently, there is a potential to create significant harmful implications. For example, within the context of a self-driving car, there are plausible devastating consequences if the system fails to recognize important components on a road. Within the medicinal field, [25] explored how adversarial attacks can lead to significant erroneous diagnoses.

To give an example, Figure 1.2 displays an image of a horse taken from the CIFAR-10

dataset, alongside its perturbed representation found with a PGD attack with $\epsilon = 2 \times 10^{-3}$. A trained model classifies the benign sample as a horse with a confidence of 98.49%, whereas the perturbed image is classified as a truck with a confidence of 99.54%. Perturbations thus can impact models negatively, therefore it is important to holistically evaluate methods to prevent substandard classification and to investigate factors which can improve a model's robustness.

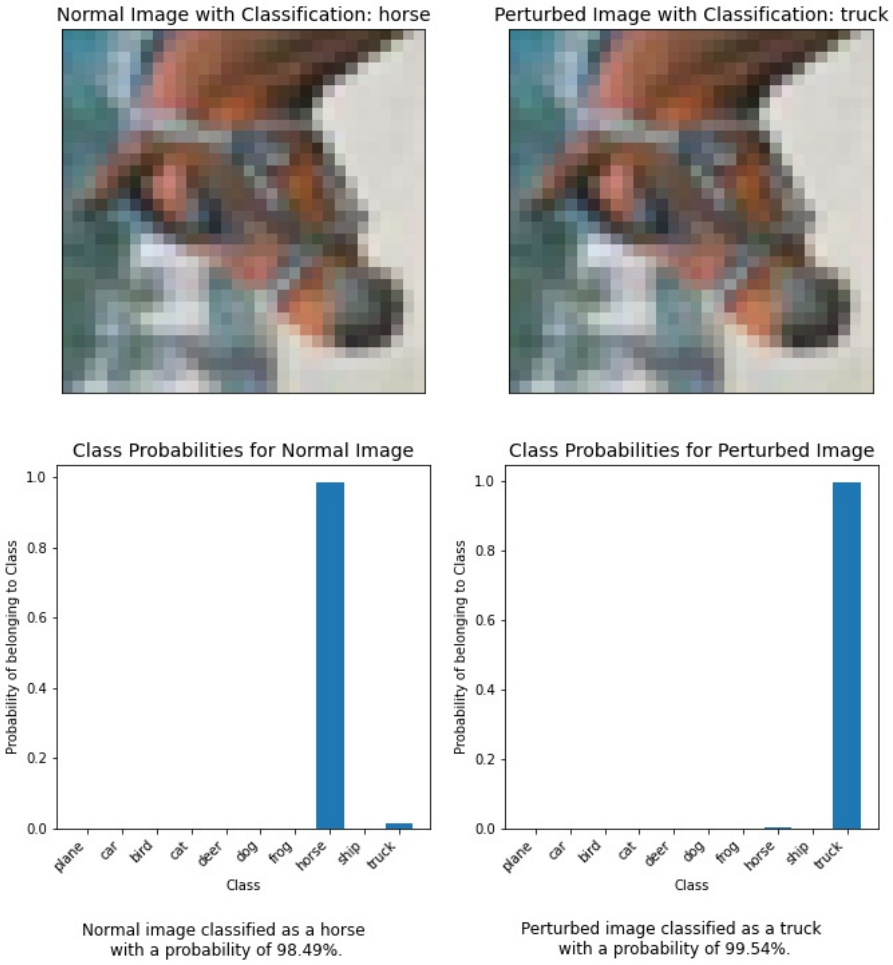


Figure 1.2: Normal and Perturbed Image Classification from CIFAR-10 Dataset with $\epsilon = 2 \times 10^{-3}$

The aim of this study was to determine whether models constructed with Deep Metric Learning (DML) are more robust to adversarial examples in contrast to those using standard Deep Learning (DL) methods. This was accomplished by deploying differing adversarial attacks (defined within the Background Chapter) on a large number of trained models. Once attacked, performance analysis of said models was undertaken. A multiplicity of loss functions

were utilised to train each model in order to investigate and identify the optimal model architecture for adversarial robustness. Network topologies were configured based on the previously determined state-of-the-art architectures for Convolutional Neural Networks (CNN). However, it is important to note that the present study was focused on network robustness rather than finding the most optimal architecture. Once the methodology of the study was concluded, it became feasible to analyse their results and subsequently conclude which method between DL and DML provides superior robustness. To the best of my knowledge, [71] is the closest-related investigation to the research question of this study. The study concluded that DML models lead to increased adversarial accuracy, and so given this result, it was an aim of this study to try and re-create the outcome that models trained using DML are more robust than those employing DL.

During the assessment of robustness of various architectures, large bodies of data were required to train and test the models. The datasets utilized were MNIST, Fashion-MNIST and CIFAR-10, all which are commonly employed within previously published research. Further descriptions are provided within the Background Chapter.

1.1 Report Structure

To begin, the present report demarcates numerous key definitions relevant to the aim of this study within the Background chapter, outlining these with reference to previously published literature which reinforces and expands on said definitions. The following concepts are examined: Deep Learning, Metric Learning, Deep Metric Learning, Adversarial Attacks and a number of other relevant subjects. An explanation of the methodology and implementation ensues, where many key elements of experiment design are discussed. Results and related analysis are then scrutinized to create ample evidence for the conclusion, which follows this section. This is followed by a list of citations.

Chapter 2

Background

The Background Chapter defines the differing concepts within this study, drawing upon definitions and examples given in pertinent previously published literature, and describes their relevance to the aim of this research. The means in which the described components are employed in this study is subsequently described in Chapter 3, Configuration and Implementation. The following sub-sections of the Background examine Deep Learning, Metric Learning and Deep Metric Learning, as well as Adversarial Attacks and conclude with a review of the used data and their pivotal role within research. For each of the sections within the background, there are associated sub-sections which aim to develop a deeper understanding of each concept. The creation of this chapter allowed for the formulation of the experiment and its methodology as this study attempted to further understanding regarding the adversarial robustness of Deep Learning and Deep Metric Learning models.

2.1 Deep Learning

2.1.1 Introduction to Artificial Neural Networks

Deep Learning, a prominent sub-topic within the field of Machine Learning, inspired by the biological structure of a human brain, employs artificial neural networks (ANNs) to perform a particular task. ANNs are made up of a user-specified number of layers, composed of varying numbers of nodes, where subsequent layers are linked with weighted connections thus communicating signals which carry out the designated task [83]. Such networks can be applied in a supervised, semi-supervised and unsupervised learning setting, rendering them adaptable to a plethora of problems. For any continuous function $y = f(\mathbf{x})$ neural networks can approximate

its parameters, thus defining the network a universal approximator where the aim of the classification process is to estimate $\omega = f(\mathbf{x})$ [20, 92]. Each model contains an input, which receives the sample to be classified, and an output layer which designates the class the sample belongs to. In between this, there are a number of hidden layers and nodes whose task is to classify the input and communicate this to the output [47]. The input and output layers are deemed visible, since they liaise with the external environment, whereas the hidden layers are hidden; they process inputs and output this to other nodes in ensuing layers [75]. It is important to note that generally, a neural network is deemed 'Deep' once it has exceeded three layers within its structure, or when there is more than one hidden layer [1]. These deep networks have become a standard part of research, and were used to design the models in this study.

2.1.2 Training a Network

The question arises, how does a neural network learn? Neural networks are comprised of nodes, where each node within the network accommodates properties which enable computation to occur [85]. Each node contains a transfer or net function, and an activation function. Given a node j in layer k, the transfer function is assigned the task of multiplying the outputs of the nodes in layer $k-1$ with the associated weights connecting those nodes to j. These products are summed together, and fed into the current node, which is the output of the transfer function [23]:

$$net(j) = \sum_{i=1}^{d} x_j w_{ji} + w_{j0}$$

Here i is the index of nodes in the input layer, j the index of nodes in the hidden layer, and w_{ji} indicates the connecting weights between these nodes. Following this, the output of the transfer function is fed into the activation function which determines the final output of j [87]:

$$y_j = f(net_j)$$

y_j travels further into the network and a similar operation occurs at nodes in ensuing layers. There exists a variety of available activation functions, most notably the Sigmoid, Hyperbolic Tangent and Rectified Linear Activations. The Sigmoid Activation Function is given as [67]:

$$S(x) = \frac{1}{1 + e^{-x}}$$

the Hyperbolic Tangent as [113]:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

and the Rectified Linear Activation Function as [105]:

$$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

Depending on the chosen activation function, the output of each node will change and with this computation complete, the input has passed through a given node of the network. In some literature the transfer function is incorporated directly into the activation function, but it is important to note the sequence of operations in each node is similar.

In addition to this, neural networks are trained with loss functions with the aim of minimising the model's error [11]. There exists numerous loss functions to select when designing networks and are a fundamental part of the model's success. The loss calculated during training is the output of the loss function, and is a clear indication whether the neural network is improving or not. The loss is a key variable to examine when constructing a neural network and choosing the number of epochs, or runs through the data, to train for. When the loss reaches a minimum, the network generally has attained its foremost performance, and further training may result in overfitting [76]. This occurs when the model becomes too accustomed to the training examples, thus performing exceptionally well on said examples, but displays poor performance when exposed to new, previously unseen inputs [43]. As the given objective of the study is classification, Cross-Entropy is a common loss function used when constructing Deep Learning networks. Cross-Entropy loss is defined [60]:

$$H(p, q) = -\sum_{i=1}^{N} p_i \log q_i$$

where p_i is the target classification and q_i is the softmax probability of the i^{th} class. Inputs into the loss function include the model's output and the true label associated with a particular sample. Whilst alternatives have been analysed [28], Deep Neural Network architectures commonly employ backpropagation. Model parameters are adjusted in correspondence to the loss, where weights are updated using the backpropagation method and a chosen optimisation algorithm such as Stochastic Gradient Descent. This is an algorithm which calculates the gra-

dient of the error with respect to each weight at hand, and uses the calculated value to update each weight respectively [30]. By iteratively updating the weights, the network learns how to classify feature vectors with a greater accuracy. However, networks with many layers may suffer with certain issues, such as the vanishing or exploding gradient problem, which causes weight updates that are too little or too large respectively [32].

2.1.3 Optimisation Algorithms

As alluded to previously, weights in a neural network are optimized using an optimisation algorithm on the gradients which are fed through the model with backpropagation, rendering this an important step in model training. The present study employed Stochastic Gradient Descent (SGD) and Adam as the chosen optimisation algorithms. SGD performs an update based on each input fed into the network [5]. As the algorithm does not pass over all samples in the training set for one iteration, and instead examines one instance at a time, it is computationally inexpensive which prompts quickened training times. SGD has the following update rule [10]:

$$w_{t+1} = w - \eta \nabla_w J(w; x^{(i)}; y^{(i)})$$

where η is the learning rate, and $x^{(i)}$ is a sample with the associated label $y^{(i)}$. As first introduced in [37], Adam functions well when faced with bias-correction, and due to this fact is favoured by researchers. It appeared propitious as by default it is an adaptive algorithm, and therefore adjust the learning rate during training. When the loss decreases, the learning rate increases, and in contrast, when the loss increases, this causes the learning rate to decrease. The update rule for Adam is the following [37]:

$$w_{t+1} = w - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m_t}$$

where η is the learning rate, $\epsilon = 1 \times 10^{-8}$ and $\hat{v_t}$ and $\hat{m_t}$ are equal to [37]:

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

which are subsequently broken down into [37]:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

where v_t is the decaying average of past squared gradients (exponential) otherwise described as the second moment (uncentered variance), and m_t is the decaying average of past gradients (also exponential), otherwise known as the first moment, the mean with respect to the gradients. The update rule makes use of $\hat{v_t}$ and $\hat{m_t}$ as these are estimates with bias correction implemented, thus giving Adam its beneficial properties. [37] suggest that optimal default values are as follows:

$$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$$

2.1.4 Choosing a Network Topology

Training cannot occur without choosing an appropriate network topology or design, rendering this choice a significant obstacle within Deep Learning research. Given the data and objective at hand, the type of neural network model changes, as well as its corresponding architecture. Some important variables to consider are the learning rate, loss function, number of layers and nodes for the network, amongst a number of other vital characteristics. Some examples of network structures include Multi-layer Perceptron Models [91], Competitive Networks [15], Recurrent Neural Networks [115], Negative Feedback Networks [74] as well as many other variations, all of which are selected to perform tasks of contrasting nature. When training a model, a dataset is split into training and test sub-datasets otherwise known as training and test sets (although variations exist, such as the introduction of a validation set), the model trains on the training set, and is subsequently tested on the test set [97]. The results are then measured with a range of standard accuracy metrics. These metrics allow for models to be improved or altered if their results are not up to the standard of the user.

A common form of deep neural network architecture is the Convolutional Neural Network (CNN), a type of network which is heavily employed in the present study. A CNN is simple to identify; it is a model in which the transfer function belonging to at least one layer is computed by using Convolution [3]. Image data can be represented as a matrix of pixel values, therefore can be subject to the process of Convolution. This operation occurs by applying a convolution kernel, otherwise described as a filter, onto each pixel within the input matrix. A mathematical computation transpires between the image and convolution matrices [99], where each pixel in the input is multiplied by the corresponding pixel of the convolution kernel, and their results are summed to produce the output value. This operation is illustrated in Figure 2.1.

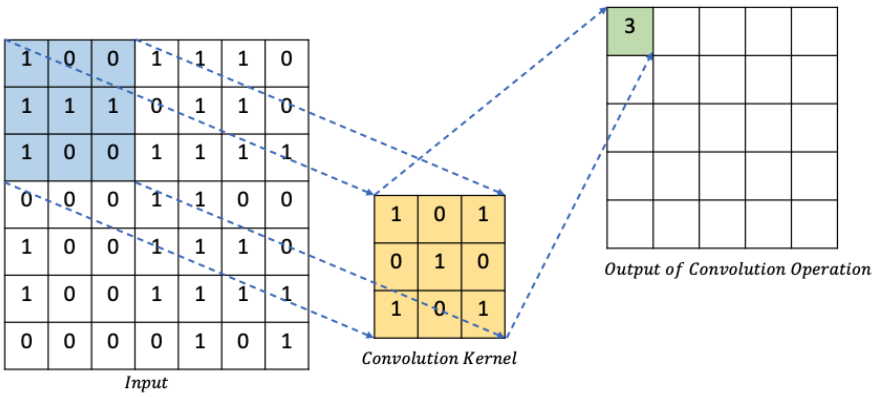


Figure 2.1: Example of Convolution Operation

The aim of Convolution is to capture demarcating properties of the inputs or enable efficient feature extraction, which allows for classes within the data to become distinguishable [108]. The properties, or the weighting coefficients of the convolution kernel are learnt through the training process with backpropagation [53]. Convolution carries with it certain parameters such as the stride and padding, both which contribute to how the computation behaves and the resultant output. The stride controls for how the convolution kernel moves over the input matrix; if the stride is equal to one, than the filter is shifted one pixel at a time, if it is two than it shifted two pixels at a time and so on [114]. Padding relates to augmenting the input by appending zeros on the border of its matrix representation [68]. This can be beneficial when the user wishes to maintain an output of a similar dimensionality as the input. If this is not specified, the dimensionality of the output following a standard convolution operation will be lower than the input.

Pooling layers are a further common component of CNN architectures and have the capability of improving a model's accuracy when the test data possesses a multitude of transformations and shifts present. Particular benefits arise when including this layer as a model becomes more robust to the locality of patterns within the data. Given an input array, the process functions by replacing a part of the array made up of a combination of values with a single value [94]. Various types of pooling exist; Average Pooling functions by taking an average of a part of the input, which is then used as the value outputted by the Pooling layer. Max Pooling works by taking the maximum value within the sliced input, and outputs this largest value [111]. Traditionally, Pooling layers have a stride greater than 1, meaning that the dimensionality of the output will be less than the input to the layer, therefore it is also referred at times as "down-sampling" or "sub-sampling". However, if this parameter is altered and made smaller,

the operation which occurs is referred to as "Global" Pooling.

2.1.5 The Applications of Convolutional Neural Networks

CNNs are optimal for the application of image recognition as they are motivated by identifying trends and patterns within an input but as described in [3], they are not dependent on where in the image the object is presented, rendering them advantageous. It gives importance to certain components of the image sample, which differentiates the properties of the class the image belongs to with regards to other classes in the data. CNNs work optimally with large datasets due to their capability to reduce an image's size but maintain important aspects of the samples [3, 44]. Once convolution has been applied to the input array, an activation function is applied to its result, which is a required hyper-parameter from the user. [48] designed a CNN model with the aim of classifying lung image patches with interstitial lung disease, showing its applicability in a medical application.[72] designed a CNN model to classify hyperspectral imagery, improving on previously set computational time and accuracy. [45] used a CNN-based approach to build LeNet, a pioneering model designed to recognize handwritten digits. [39] developed AlexNet, a further CNN model which employed techniques such as dropout in order to reduce possibilities of overfitting, a concept within Machine Learning in general, and is vital to avoid. This is when the model fits extremely well on the training set but fails to perform on unseen instances.

Apart from image data, CNNs are adaptable and can handle inputs such as text, audio, and video signifying that the type of data being classified can vary widely. [38] examined text classification using a hierarchical approach of Deep Learning models, and achieved stronger results than the previously set benchmark. When the number of classes within multi-class problems rises above a particular threshold, traditional Deep Learning models perform worse on hierarchical document classification whereas [38]'s approach solves this with their HDLTex model of creating DL approaches for each level of the document hierarchy. [52] was the first study to use Deep Learning for Extreme Multi-label Text Classification. The goal of this procedure was to assign relevant class labels to the document, where the number of classes is voluminous, ranging from hundreds of thousands to millions. Using Precision@K and NDCG@K (Normalized Discounted Cumulative Gain) as metrics, [52]'s XML-CNN model outperformed all previously set benchmarks by over 10%. [46] was one of the first studies to investigate the use of ANNs trained on unlabeled audio data to classify various audio samples, and found that using Convolutional Deep Belief Networks is effective in audio classification. Similarly, [31] investigated

various CNN architectures to classify a dataset of 70 million instances comprised of over 30,000 class labels. With the aid of various metrics such as AUC (Area-under-Curve), d-prime and mAP (mean Average Precision), [31] concluded that using CNNs allowed for improved audio classification, ameliorating previously set benchmarks. [62] built a model which could tune various hyperparameters of neural networks and alter their topologies resulting in accuracies comparable to the most optimal human designs. Similarly to this, [7] employed various reinforcement learning techniques in order to generate optimal CNN topologies, and achieved results which rival state-of-the-art hand-crafted models.

2.2 Metric Learning

Metric Learning involves measuring distances between samples in a dataset, and performing classification depending on these measured distances. The training that occurs within this domain is the algorithm aims to learn a new metric, which minimizes the distance between feature vectors belonging to the same class and in contrast, maximizing the distances between instance belonging to dissimilar classes [8]. Within Machine Learning, the k-Nearest Neighbours algorithm follows this logic; feature vectors are classified based on the majority of points closest to them belonging to a specific class, and inter-class distances are maximized in order for classification to perform with maximal accuracy [51]. To give an example, as mentioned in [100], processes in the field of Computer Vision benefit from Metric Learning as it aims to find measures of similarity between pairs of images "that preserve desired distance structure" [100]. This has positive implications as it can improve concepts such as image search, when data is split across a large number of classes [100]. Metric Learning uses a distance metric to compute the distances between instances such as the Euclidean, Cosine Similarity or Mahalanobis distances. The Euclidean distance between points x and y is [101]:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

whereas Cosine Similarity between vectors A and B is [69]:

$$\text{Cosine Similarity}(A, B) = \frac{A \bullet B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The Mahalanobis distance between vectors u and v is found with the following [61]:

$$d(u, v) = \sqrt{(u - v)V^{-1}(u - v)^T}$$

Similar to this, there exists a plethora of other distance metrics which are chosen based on their applicability to a particular problem.

2.3 Deep Metric Learning

2.3.1 Introduction

The topic of Deep Metric Learning (DML) is a recent advancement within the field of Deep Learning which fuses the methodologies of Deep and Metric Learning together. DML draws upon the objective of identifying similarities between instances as applied within Metric Learning. As described in [58], DML employs a chosen distance metric which allows for a transformation into a new feature space to occur. [71] articulated that Deep Metric Learning creates models which in turn project outputs onto a new embedding space, which clusters similar instances, and increases distances between dissimilar instances. Further to this,[55] identified DML as a process of learning a non-linear mapping f (function) to project instances into a new embedding or feature space with the utilization of deep neural network architecture and employs the standard characteristics of deep neural networks such as weights and biases. Based on these descriptions, it can be deduced that a set of data is presented in an environment with superior discriminative qualities, signifying that instances can be distinguished with more ease. This new space contains features of a lower dimension than the input [109]. Training is completed with the use of various metric loss functions, an indispensable feature which separates DML from standard Deep Learning methods. When a pair of inputs carrying a similar label are presented to the network during training, the loss is minimised whereas when instances from dissimilar classes are given, the loss is maximised. This training method allows for the model to generalise efficiently, even if transformations occur, suggesting why this method could be strong against adversarial examples.

2.3.2 Classification Using Deep Metric Learning

The classification algorithm for DML models differs from DL, as DML pertains to distance learning. DML allows the model to retrieve instances which are close to the input from the

embedding space. Alternatively, the model works by returning the closest related samples to the query image fed into the network. By doing so, the model can recognise which class the input closely belongs to based on the discriminative embeddings of the input and the training samples.

Based on the neighbours retrieved, the classification is determined using k-Nearest Neighbours (k-NN), a clustering algorithm with the aim of grouping together samples belonging to the same class. This algorithm requires a measure of distance, and whilst the standard measurement to use is the Euclidean distance, variations exist such as Cosine Similarity, Manhattan and Jaccard distances. Given a query sample and by fixing k to a particular value, for example k= 5, the 5 closest samples with the smallest distance to the query are retrieved. These 5 neighbours then have their class labels analysed. Following this, a process of voting occurs. Suppose 3 of the neighbours belong to c_1, and 2 neighbours belong to c_2, the query image is assigned the label c_1 as this is the more popular class in the 5 nearest neighbours. A number of assumptions are made regarding the employed distance metric [21]. Firstly, non-negativity must be present signifying that $d(x, y) \geq 0$. Further to this point, if $d(x, y) = 0$ than x must be equal to y. This is the identity assumption within the algorithm. There must also be symmetry present, and can be expressed as $d(x, y) = d(y, x)$. Finally, the algorithm must adhere to the Triangle Inequality which is expressed as $d(x, y) + d(y, z) \geq d(x, z)$ [19].

DML can be advantageous in cases when there are many classes and a small amount of data to train on. Embeddings can still be produced with proper sampling strategies, meaning that models trained with DML differentiate strongly without the necessary large data quantities of Deep Learning [58]. Having said this, the particular Loss Functions used within DML will now be described in detail.

2.3.3 Loss Functions Used in Training

A common loss function employed is Contrastive Loss, and will now be broken down. Assume that the distance taken between samples X_1 and X_2 is the Euclidean distance, than this can be represented as:

$$D(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_2$$

Given this, Contrastive loss is described with [29]:

$$L(W) = \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i)$$

which can be broken down into:

$$L(W,(Y,X_1,X_2)^i) = (1-Y)L_S(D_W^i) + YL_D(D_W^i)$$

where $(Y,X_1,X_2)^i)$ is a pair of samples i associated with label Y (whether they belong to the same class or not). L_S represents a partial loss function which is implemented for a pair of similar samples belonging to the same class, and L_D is for a pair of unalike samples, belonging to different classes. The parameter P represents the total number of possible pairs presented in the training data. This loss function's responsibility is that if the two samples, X_1 and X_2, are from the same class, then the loss is small. If they are from dissimilar classes, the loss becomes large, causing further adjustment [49].

A further function often used is Triplet Loss, which relies on triplets of data points at a time. Triplet loss analyses the distances found between an anchor, which can be any random data point, as well as a positive and negative example, where the positive example has the same label as the anchor, and the negative one a dissimilar label. The goal of this loss function is to minimize the distance $D(x_i^a, x_i^p)$ between the anchor x_i^a and positive point x_i^p whilst rendering the distance $D(x_i^a, x_i^n)$ between the anchor and the negative point larger than $D(x_i^a, x_i^p) + \alpha$ where α is a specified distance margin [26]. As described in [84], the minimised equation for the triplet loss function is the following:

$$L = \sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+$$

Here N is the cardinality of the training set. This loss function has been used in various applications, for example, [22] employed triplet loss for person re-identification, where the distance between dissimilar images was maximised.

Ranking Loss, introduced and defined by [81], is a further loss function used in DML. When minimised, the network is configured to assign scores to each class outcome, and subsequently assigns the input to the class with the highest score. Further to this, the network assigns lower scores to classes that the input does not belong to. [81] defines Ranking Loss as:

$$L = log(1 + exp(\gamma(m^+ - s_\theta(x)_{y^+})) + log(1 + exp(\gamma(m^- + s_\theta(x)_{c^-}))$$

where $s_\theta(x)_{y^+}$ and $s_\theta(x)_{c^-}$ are the scores for for the correct and incorrect class labels respectively. γ is scaling factor, which aids in disciplining prediction errors, and m^+ and m^- are

margins.

Angular Loss is a methodology explored and pioneered by [100]. As described, various loss functions focused on employing a distance metric do not perform well with scale change in various classes. The primary advantage of applying angular loss is the network handles varied levels of intra-class variation with ease, as opposed to conventional triplet loss. Angular loss works by finding a minimum of the angle at the negative point, whilst remaining invariant to scale due to its inherent properties. By focusing on the angle at the negative point and placing an upper-bound limit on its value, this drives the negative point away from the resulting anchor and positive cluster pair. [117] investigated employing angular loss in DML applications and extending this with the use of a spherical embedding constraint. Their investigation into topics such as contrastive self-supervised learning resulted in DML models with optimal performance.

2.3.4 Siamese and Triplet Networks

As identified in [55], the most common forms of neural networks used in Deep Metric Learning are Siamese Networks and Triplet Networks, with contrastive and triplet loss functions implemented respectively. Siamese networks are represented by two parallel networks which have similar topologies, weights and other hyperparameters. Their aim is to compare and contrast the two inputs, and calculate a loss dependent on the labels of these two inputs. When the labels of the inputs are similar, the loss is minimised, and in the case of inputs with dissimilar labels, the loss is maximised. Triplet networks, introduced by [33], are inherently slightly different in nature and use three inputs instead of two during training. An anchor x is fed into the system along with a positive input, x^+, and a negative input, x^-. The output of this operation is the L_2 distance between the embeddings of x and x^+, and x and x^- [33]. The objective of the Triplet network structure is to minimise the distance between x and x^+, and maximise the distance between x and x^-. An example of the Triplet network structure is shown in Figure 2.2.

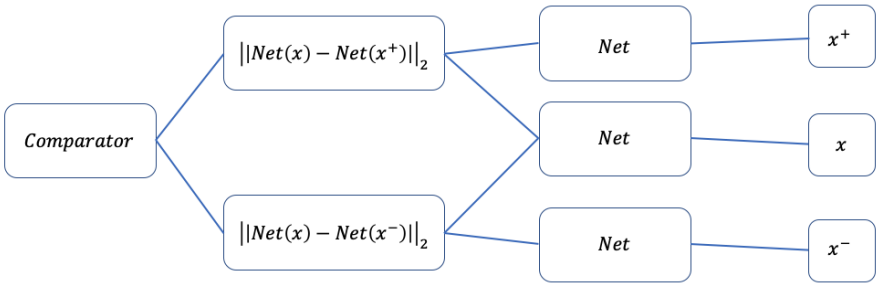


Figure 2.2: Structure of Triplet Network [33]

2.3.5 Applications of Deep Metric Learning

Deep Metric Learning is applicable in a variety of contexts in tasks such as image retrieval [70] and object detection in video footage [9]. In the case of [33], the L_2 or Euclidean distance was used where the method aimed to identify a function $F(x)$ where similarity between instances x and x' is $S(x, x') = \|F(x) - F(x')\|_2$. [110] employed DML for person re-identification purposes. With the aid of a Siamese CNN structure, the study achieved superior results to previous Deep Learning models, suggesting that a DML approach was more optimal and applicable in this setting. Another field where DML prospered was speech recognition applications. [49] used a similar Siamese Network approach with contrastive loss in order to classify speech inputs into different categories of emotion. [118] investigated the use of DML in a medical setting, specifically for analysing chest radiograph images during the COVID-19 pandemic to achieve a superior comprehension of patient diagnosis and treatment. The study discovered that a DML approach increased the accuracy of such systems, further indicating the advantages, applicability and adaptability of DML approaches.

2.3.6 Mining

It is vital to note the various sampling techniques used for input instances, as this can be an influential factor on model performance. Further to this, an optimal sampling strategy can lead to quickened training times, therefore in the context of this study, it is an important factor to discuss and is described further in the Configuration and Implementation Chapter.

2.4 Metrics to Measure Model Performance

Within research, model evaluation is performed by using a variety of metrics. In the context of deep learning, classification accuracy has been a popular method of assessing model

performance, and is defined as:

$$\text{Classification Accuracy} = \frac{\text{correctly predicted samples}}{\text{total predictions}} \times 100$$

Further to this, it is important to examine performance pertaining to a specific class as well. This can be achieved with Precision and Recall which are defined as [98]:

$$\text{Precision} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})}$$

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

To further this explanation, a true positive sample is when model correctly predicts an input to belong to the positive class when the sample has a previously determined positive label. A false positive is when the model predicts the input to belong to the positive class, whereas the input actually has the negative label attached to it. Precision and Recall can be furthered and combined to create a joined metric known as the F1-score which is defined as [16]:

$$\frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}$$

Accurate evaluation of Deep Metric Learning models requires a variety of informative metrics. DML is associated with establishing a distance between samples during the training process, with inputs from the same class projected to a similar location in feature space, and inputs belonging to differing classes extended to contrasting locations. The required metrics focus on the relationships of the projected inputs with their nearest neighbours in order to gauge the distances and locations of the samples. With this being said, popular metrics used within research of this field are Recall@K and Precision@K [36]. Recall@K examines the relationships that samples in feature space possess with their nearest neighbours, and construct a method of understanding sample locations in said space. Precision@K further illustrates relationships between neighbouring samples in feature space. K specifies the number of nearest neighbours for each sample to examine. In the case where $K = 1$, each sample examines its one nearest neighbour. If the neighbour belongs to the same class, than the Precision@1 measurement is set to 1 for that sample.

2.5 Adversarial Attacks

With the increased implementation of Deep Learning methodologies in various environments becoming more prominent, architects must ensure that such models remain secure, in particular their ability to differentiate adversarial examples from true samples. An adversarial example can be denoted as $x + \delta$, where δ is the perturbation and x the original sample. Whilst x and $x + \delta$ appear similar, an accurate neural network may still severely misclassify $x + \delta$ even though x is correctly classified with confidence. Such perturbations can be generated with an adversarial attack. [80] adds that adversarial attacks are implemented by perturbing inputs whilst remaining undetectable by human features. As described in [95], it is the process of manipulating an input in order to confuse the trained neural network thus resulting in severe misclassification. It takes an input x, a member of class c_1, and finds x^{adv}, an adversarial example analagous to x, only the network does not assign the sample to c_1 anymore. These adversarial attacks can be implemented by non-randomly altering the features of the inputs slightly which leads to significant misclassification rates for trained models [80, 95].

Adversarial attacks can be split into three categories; white-box, grey-box and black box attacks [66], all which accommodate differing properties. White-box attacks are knowledgeable of the various parameters of the model they are attacking, and use gradients to create the adversarial samples. Grey-box attacks relate to creating a generative model which attempts to create the best possible adversarial samples, and only has knowledge of the model that they are attacking during training [66].

White-box, grey-box and black-box attacks can be split into further sub-categories which relate to the assigned class of the input. The first is the non-targeted adversarial attack, which creates a perturbation of the input wherein the network then outputs a randomly chosen class, with the exception of the input's true class given that the attack is trying to confuse the network. The second identified sub-category is the targeted adversarial attack, which modifies the input and forces the network to classify the example to a distinct class specified by the attack.

Finally, the third area of focus within literature regarding adversarial attack is the defense against these attacks and how to prevent them having a significant effect on a classifier. The following sections describe a variety of adversarial attacks and their methodologies, and a number of said attacks were used in the method of the present study.

Szegedy's Limited-Memory BFGS (L-BFGS) Attack, is the first adversarial attack algorithm introduced with the aim of perturbing images to confuse CNN models [106]. [95] introduced that the adversarial example x' has the objective min $\|x - x'\|_2^2$ so that x' is classified to a target

t. [95] computes this with the use of a loss function with the following objective:

$$\min c\|x - x'\|_2^2 + L(\theta, x', t) \quad s.t \quad x' \in [0,1]^m$$

The left side of the function ensures analogy between x and x'. The right side of the equation ensures that the adversary perturbes x with a minor loss value to target label t causing the classifier to be erroneous, and misclassify x'. The perturbation for each sample has the following target [79]:

$$p(x) = \min_{\delta} d(x, x+\delta) \quad s.t \quad x+\delta$$

However, as mentioned in [79], it is extremely difficult to find the minimum adversarial perturbation, and so other attack methods may be more beneficial to try and achieve this objective.

One of the proposed attack algorithms is the Fast Gradient Sign Method (FGSM), and analagous to the definition in [63], it determines an adversarial example as

$$x^{adv} = x + \epsilon(\nabla_x L(\theta, x, y^{true}))$$

where $L(\theta, x, y)$ is the chosen loss function and $\theta \in R^p$. [65] successfully used FGSM to generate adversarial examples on several different datasets in order to demonstrate how trivial it can be to fool a deep neural network. In contrast to this, [14] suggests that FGSM is an inadequate attack algorithm for computing the robustness of an ANN model and recommends to avoid it entirely. Such a recommendation is made since there exists a multitude of defenses which accomplish cogent accuracies. To explore these contrasting findings, the present study employs FGSM as one of the chosen adversarial attacks.

The Basic Iterative Method (BIM) attack is a continuation of FGSM. It is iterative, and attempts to find the edge of the ϵ max-norm ball of the input, and is therefore an aggressive attack technique. This means that it aims to find a significant perturbation of the input x. As defined in [42] and taking notation from [63], BIM generates

$$x_0^{adv} = x, x_{N+1}^{adv} = Clip_{x,\epsilon}\{x_N^{adv} + \alpha(\nabla_x L(\theta, x_N^{adv}, y^{true}))\}$$

[42] chose $\min(\epsilon + 4, 1.25\epsilon)$ for the number of iterations.

Projected Gradient Descent (PGD) employs a similar methodology as to BIM. However, subtle differences exist. PGD begins by configuring a random perturbation from a uniform distribution, and runs iterations until the optimal adversarial sample is created [35]. By doing

so, the attack is found to be more aggressive, and is therefore an applicable attack in the goal of the present study.

As is the case with L-BFGS, the Carlini-Wagner Attack Algorithm aims to find the perturbation of sample x with the least amount of distortion possible. The author solves

$$\min c\|x - x'\|_2^2 + c \cdot (\max_{i \neq t} Z(x')_i - Z(x')_t)^+$$

with the goal of assigning the largest score to the target t which causes the model to misclassify x'. As stated in [106], the use of the margin loss instead of cross entropy as in L-BFGS, ensures that the distance between x' and x is minimal, which is aligned with the overall goal of the algorithm.

The Jacobian-based Saliency Map Attack involves computing the Jacobian matrix of a score function F. This attack iteratively changes each pixel value in a sample x, to find the one which has most impact on the output of the model at hand. [73] employed the Jacobian matrix

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \{\frac{\partial F_j(x)}{\partial x_i}\}_{i \times j}$$

which displays the alteration of $F(x)$ when x is adjusted. With a target t in mind, the adversarial attack changes the pixel values of x which causes $F_t(x)$ to alter $\sum_{j \neq t} F_j(x)$.

As is apparent in previously published literature, there exists a large number of varying adversarial attack possibilities. To improve robustness, [18, 27, 42, 57] all suggest that the best defense mechanism to adversarial attacks is adversarial training, where the found perturbed samples are introduced into the model's training set.

2.6 Datasets

Selecting data to analyse the research question in the present study is of the upmost importance, and so it is vital to understand what previous research has employed for analysis. [29] performed initial experiments on a sample from the MNIST dataset, a popular database within the Computer Vision field consisting of images of handwritten digits. An example from each class of the MNIST dataset is shown in Figure 2.3.

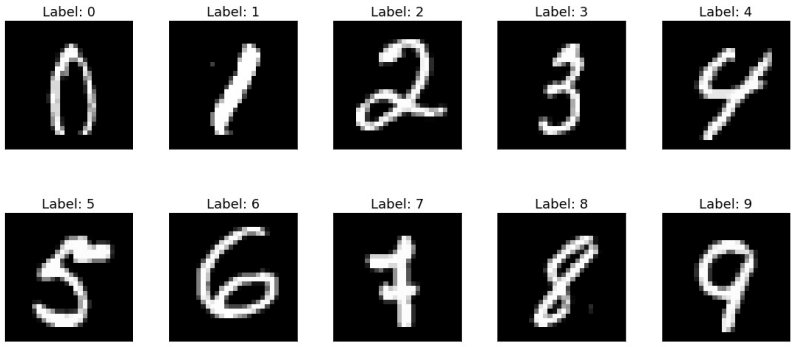


Figure 2.3: Example of Digits in MNIST Dataset

[88] analysed various methods to improve CNN performance including data augmentation and applied their experimental solutions on MNIST. At the time, their results were superior, and their study resulted in being valuable to improving CNN capabilities. Similarly, [96] performed data augmentation on MNIST, employing techniques such as translations, centering, rotations, and elastic deformation in order to train a CNN model which subsequently generalised better on unseen examples. [17] also employed such techniques for creating various CNN models, and achieved leading results on MNIST. [33] investigated using Deep Metric Learning with a Triplet Network structure, and analysed their method on MNIST, achieving high research-standard results. With this being said, studies in Deep Learning [86] and Deep Metric Learning have used the MNIST database for their experiments, therefore testing on this dataset was appropriate for the aims of this study.

Fashion-MNIST is a commonly used dataset within research [104] and is similar in nature to MNIST but differs in what the ten classes of the dataset represents. For example, some similarities include that images in both datasets share a common black background, are the same size and belong to one of ten classes. Examples of digits pertaining to each class within the Fashion-MNIST dataset are displayed in Figure 2.4.

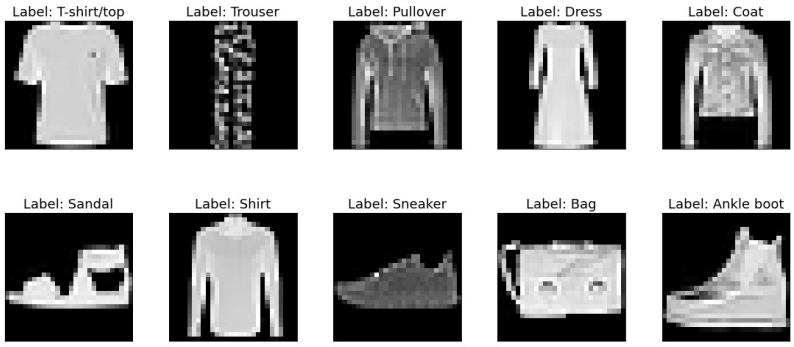


Figure 2.4: Example of Items of Clothing in Fashion-MNIST Dataset

CIFAR-10 is a dataset consisting of labelled colour images belonging to ten different classes and has been widely analysed in previously published literature. Examples of each class of this dataset are shown in Figure 2.5.

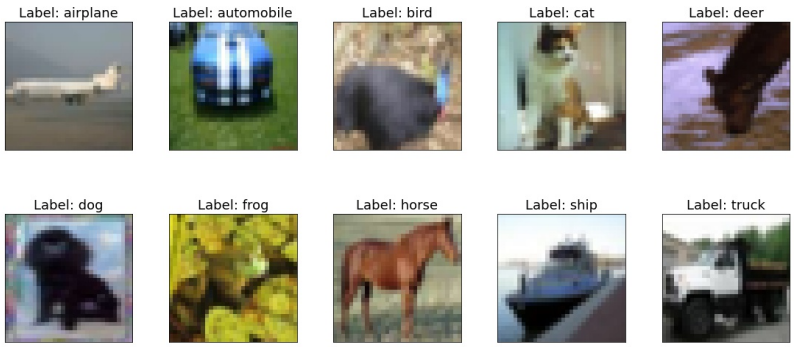


Figure 2.5: Example of Classes in CIFAR-10 Dataset

[39] applied CNN topologies to achieve state-of-the-art performance at the time of publication, improving any accuracy measures published prior to the study. [7] studied their reinforcement learning-based approach for CNN creation on CIFAR-10, allowing them to test their model's accuracy. Recurrent CNNs studied by [4] and [50] similarly used CIFAR-10 in order to assess their approaches. [112] used CIFAR-10 to investigate DML approaches and specifically altering the distance metric, as opposed to solely using CNNs for image classification tasks. By using a Signal-to-Ratio as a similarity measurement and replacing the Euclidean distance metric, the study's authors were able to generate more discriminative features in their models and concluded that using such an approach achieved exceptional results. This methodology was tested on CIFAR-10 amongst other databases, and used a variety of loss functions such

as the contrastive, triplet, lifted and N-pair losses. Having established this, it is evident that CIFAR-10 remains a key database within research and remains a valuable tool for researchers to evaluate their models on.

A further standardized dataset used commonly in research applications is the open-source ImageNet database, a vast collection of images annotated by researchers. Although this dataset was not investigated for analysis in the present study, it is important to provide context due to its significant impact on varying fields of research. [40] investigated deep CNN's whilst training on the ImageNet database for the LSVRC-2010 contest, and achieved unprecedented results at the time, resulting in the pioneering AlexNet model. Furthering this study, [78] investigated altering various parameters of AlexNet training on ImageNet in order to achieve stronger results. More specifically, the study investigated weight binarization as well as the intermediate representations of the CNN model. This was done by looking at Binary-Weight Networks, where weights are assessed as binary values, and by introducing the concept of XNOR-Networks, where weights and inputs to convolutional layers are similarly valued with binary values. [89] investigated regularization techniques to reduce overfitting and attempted differing methods on ImageNet to make conclusions. The study determined that Dropout works better than DropConnect, but requires larger model sizes for improved performance. The authors suggest that new methods such as Data Augmentation, Standout or Stochastic Pooling amongst others may prove effective in improving classification accuracy. [64] investigated optimal CNN topologies and hyperparameter tuning and tested their methods on ImageNet. The study analysed changes in activation function choice, pooling variants, network width, classifier design, image pre-processing as well as setting an optimal learning rate, batch size amongst other factors. The study concluded that the use of 128×128 pixel images is adequate to infer deductions regarding optimal network structure and significantly improves computation speed rather than using 224×224 pixel images. Further to this, Deep Metric Learning has been profoundly dependent on models trained with ImageNet. [70] DML models used convolutional layers pre-trained on ImageNet in order to achieve their results. Furthermore, a plethora of studies have investigated DML methods where their initial network parameters were pretrained on ImageNet [12, 77, 102, 106].

Chapter 3

Configuration and Implementation

3.1 Configuring CNN architectures

As mentioned in the introduction, the models examined in this study were Convolutional Neural Networks, a practicable ANN that provides advantageous capabilities in image classification tasks. Configuring their structures was a question of vital importance, and as mentioned in [90], examining and resolving the various trade-offs which occur was a particularly arduous task. By doing so, an improved understanding of the influential forces in a network were analysed, allowing for a decision on which network entities to retain or omit to occur. With this being said, networks for each dataset were configured in differing ways. A number of hyperparameters were altered throughout the training process to achieve superior results and heightened performance. As suggested in previously published literature, there exists optimal strategies implemented by researchers which generally lead to a positive impact on network capabilities. To confidently conclude which method between DL and DML is more advantageous for creating models with superior adversarial robustness, a significant quantity of networks with changed hyperparameters were created for analysis. By attacking these models with varied attack strategies, and by observing whether varying hyperparameters such as the loss function could help mitigate against said attacks, the optimal strategy could then be determined. It is of significant importance to regard that the primary aim of this study was not to create models with the highest accuracies, but to determine which structures are more robust to adversarial examples. Therefore, an extended duration of time was utilised constructing models

with different parameters rather than consistently updating network structures to improve classification accuracy. The process of creating the network topologies for the various datasets will now be described and explained.

3.2 Creating the Network Structures

3.2.1 When to stop training?

A further question arose during experiment configuration: for what duration should each model be trained for? When model training time is too short, the potential risk of model weights not being configured to their optimal values and underfitting rises. This means the model may not perform to its potential. In contrast, if the model is subjected to elongated training times, the risk of overfitting increases, subsequently causing model performance to decrease significantly. A number of metrics were used in the implementation to improve comprehension regarding training duration of each model.

An established technique to determine the optimal number of epochs to train a model for is early stopping; which aids in decreasing the risk of overfitting [54]. This revolves around examining both training and validation errors of a particular epoch as training occurs, and analyzing the behaviour of these variables throughout this process. As expected, both training and validation errors decrease as the number of epochs increases. However, at a certain point, the training error will continue to decrease, whereas the validation error will begin to plateau, and may rise. As described in [54, 82], when this occurs, training should be terminated as it indicates that weight adjustments within ensuing epochs do not contribute positively to model performance. Furthermore, early stopping possesses the advantage of reducing a model's training time as it is only trained for as many epochs as required. However, whilst it appears that there exists a trivial solution for determining when to stop training with early stopping, as pointed out in [76], the stopping point is not always easily identified as the validation error curve may display a multitude of local minima. Therefore, it is important to understand the trade-off that occurs between training for longer periods and the increased risk of overfitting, or stopping too early and having the model underfit. Within this study, the curves of the training and validation errors were analysed and a stopping point for training was determined by examining their behaviours. MNIST and Fashion-MNIST models were trained for 40 epochs, whereas CIFAR-10 models were trained for 50.

3.2.2 Activations

As described in the Background Chapter, each layer of the configured networks contains an activation function. The Rectified Linear Activation Function, otherwise shortened to ReL, was chosen to be used. Each node within the network that employs the function is known as a ReLU, the Rectified Linear Activation Unit. To reiterate, if the output of a node is positive, then it remains unchanged. However, if the output is negative, then following the application of ReL, the output becomes 0. ReL possesses certain benefits such as allowing weight adjustments to occur more consistently. This signifies that the gradient moves fluidly through the network, which can aid in resolving issues such as the Vanishing Gradient problem, where weight adjustments are too minute, and therefore cause sub-optimal performance.

3.2.3 Loss Functions

In order to ascertain which methodology created more robust models to adversarial examples, a large number of models were configured. Neural networks employ loss functions during training to adjust weights, therefore varying this parameter was key to the investigative properties of this study. A multitude of loss functions exist, as mentioned within the background. For the present study, the DL models used Cross-Entropy loss and DML models employed Contrastive and Triplet losses.

3.2.4 Learning Rate, Optimisation Algorithms, Momentum and Weight Decay

A model's designated learning rate determines the degree of change to apply to the weights given the error. An increased learning rate results in weight changes that are too large, meaning that the training may converge too quickly, and the optimal weights may not be learnt resulting in unstable learning. On the other hand, with a learning rate that is too small, the training time may take too long to converge and is computationally expensive. It can also lead to training which gets stuck, and may not find the optimal parameters. The learning rate η was set to be 1×10^{-3}.

For the employed Optimisation Algorithms, the models in this study used Stochastic Gradient Descent and Adam, which are delineated in the Background Chapter.

An issue which can arise during training is that the model may not find the global minimum. The minimum guarantees that the error is minimised and that the model's weights are set to

their optimal values. A hyperparameter known as Momentum can improve a model's capability of avoiding falling into local minima and can also lead to improved training times [6]. It is a component which is added to the update rule, and aids in finding a global minimum. Given that this may lead to improved accuracies, the decision was made to include this parameter in the various model designs. It is important to note that a trade-off exists in setting this parameter; if the momentum is too high than the model may overshoot the global minimum and not find the optimal weights. On the other hand, a momentum value which is too low could lead to a model which is not able to leave a local minima if it is stuck. Further to this, the learning rate must be small if the momentum term is significantly sizeable. If this is not apparent, the model may not find the global minimum of the validation error. Momentum for when Stochastic Gradient Descent was used as the optimisation algorithm was set to be $= 0.9$, as this appeared to be an advantageous parameter in previous studies [33]. This further improved the baseline accuracies of the employed models. In the case of when Adam was as the optimizer, the momentum is learnt through training and therefore was not set prior to training commencing.

Weight decay, a regularization technique, was a further hyperparameter introduced into the model design and was included to avoid the models' risk of overfitting [41]. It adds a penalty value to the loss function, and can also prevent the model from succumbing to the exploding gradient problem faced in neural network training. By keeping the weight values to a minimum and preventing them from becoming too large, the network can remain stable. Weight decay was set to be $= 4 \times 10^{-4}$, a parameter found to be beneficial in Deep Metric Learning research [71].

3.2.5 Mining Strategy

In contrast to the training methods employed in Deep Learning processes, the loss associated with one sample within Deep Metric Learning is interlinked with other data points, hence the importance of distances. Having said this, efficient sampling strategies of the data is significant in creating an optimal training environment for the various DML models. For example, when using Triplet Loss in training, it is necessary to have an anchor image, along with positive and negative samples to compute loss. With a substandard sampling strategy for these examples, the network will not achieve the highest possible results but with the right input mining strategies, this can be improved. The DML models in this study employed a Multi-Similarity Mining strategy [103], which allowed for quick and efficient model training.

3.2.6 Batch Normalisation

A concern encountered when training Neural Network architectures is that the distribution of inputs in layer k will vary depending on the learnt parameters in $k-1$, the previous layer. As a result, learning is slow and takes longer to converge. The method employed to resolve this is to normalise the outputs of $k-1$ in order for the inputs of k to possess a mean close to 0, and variance equal to 1. Batch Normalisation is defined as [34]:

$$BN(x) = \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}} \times \gamma + \beta$$

where $E(x)$ is the mean value of x, β and γ are parameters learnt during training with the help of backpropagation, and $Var(x)$ is the variance of x. Both $E(x)$ and $Var(x)$ are learnt from the current batch of data fed into the network during training. The hyperparameter ϵ is incorporated to ensure that division occurs only by non-zero values and was set to be $\epsilon = 1 \times 10^{-}5$. For MNIST and Fashion-MNIST, networks were configured to have a Batch Normalisation layer after the ReL activation function was applied to the Convolution layer. In the case of CIFAR-10, Batch Normalisation was applied prior to the activation in the Convolution layers.

3.2.7 Dropout

A method to avoid overfitting is the incorporation of dropout, a technique which imitates the training of various network topologies at once. During learning, arbitrary nodes are 'dropped out' of the training process, simulating that layers are diversified with a dissimilar number of said nodes [93]. By having random dropouts, the network is forced to adapt to a more obstreperous training process which in turn prevents overfitting and renders a model more robust. Dropout can be used on any of the hidden layers in a network, and has a probability hyperparemeter p, controlled by the user. For example with $p = 0.3$ and samples taken from a Bernoulli distribution, an element is dropped out or not. Dropout is particularly effective in deep architectures with a multitude of layers as they are prone to rapidly overfitting, therefore rendering itself useful to the models of this study. The Dropout parameter p was set as $\{p|0.1 \leq p \leq 0.5\}$ where the optimal p was determined using grid search, which appeared to regularise learning.

3.2.8 Models for each Dataset

Network topologies for each dataset were standardised to allow for a comparison between DL and DML model robustness. For example, when training on MNIST, both DL and DML models would carry a similar network topology, where hyperparameters such as learning rate were the same, and the only difference being the choice of loss function. A standardised method of comparing models between the two approaches allowed for a fair and in-depth analysis after results were obtained.

Having described the various layers and parameters employed, the attention turns to the concrete model structures for each dataset. Studies in the past have successfully used blocks of Convolutional layers [45] along with Dropout to create models of superior performance on MNIST, Fashion-MNIST and CIFAR-10. Having been inspired by this, a similar strategy was employed. Convolution layers were included in all of the models, and were imperative to the success of the networks' ability to distinguish important features in the image data. Furthermore, by employing batch normalisation often, the learning rate did not have to be initialized at a minute value. After each subsequent block, a dropout layer was introduced and the ReL activation function was used at each layer.

For MNIST and Fashion-MNIST, six Convolution layers were included in total, with kernel sizes ranging from 3×3 and 5×5. The padding was kept consistent at 1, whereas the Stride varied between values of 1 and 2. These parameters were configured during exploration, and were found to be optimal for MNIST and Fashion-MNIST. A common addition to a CNN's architecture is the Pooling layer, which aids in making a network more robust to positional features in an input. However, in the case of MNIST and Fashion-MNIST, the test data did not contain any major shifts or translations, and therefore it was decided to omit Pooling from the networks, as the convolutional layers were efficient enough at detecting important features. The following diagram (Figure 3.1) shows the architecture employed for MNIST and Fashion-MNIST.

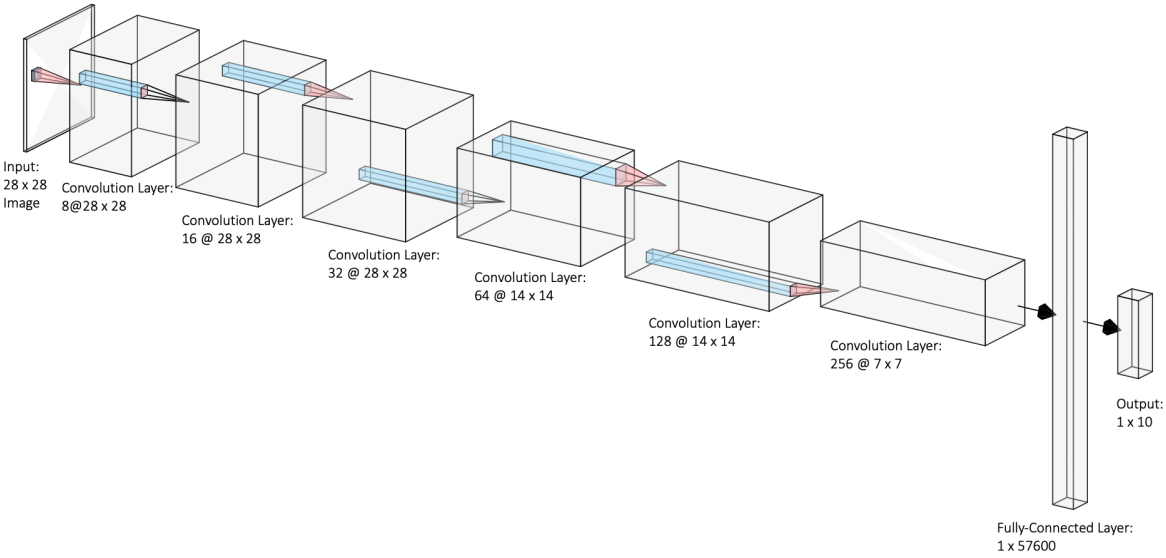


Figure 3.1: MNIST and Fashion-MNIST CNN Architecture

However, for CIFAR-10, the inclusion of a Max Pooling layer improved the models' baseline classification accuracies. Therefore, the networks were comprised of three convolution blocks; each containing a Convolution layer with kernel sizes set to 3×3, Batch Normalisation followed with an ReLU activation function, a further Convolution layer with ReLU, and a Max Pooling layer. This was followed by a fully-connected block; comprised of a Dropout layer with $p = 0.1$, two Linear layers with ReLU, followed by Dropout ($p = 0.1$) and a final Linear layer. The learning rate was configured to be $\eta = 1 \times 10^{-3}$. Figure 3.2 illustrates the architecture employed for analysis on CIFAR-10.

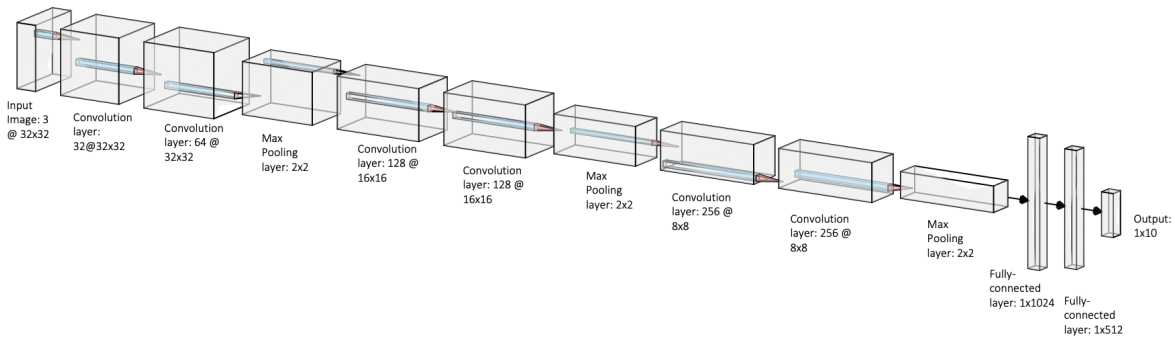


Figure 3.2: CIFAR-10 CNN Architecture

Whilst models for MNIST and Fashion-MNIST were similar due to the fact that the input samples were of a comparable size and structure, the case was not the same for the CIFAR-10

training process. Input images were not grayscale, and the features within the training samples were drastically more complicated as images were larger and contained an increased number of features. Some of these features were varied backgrounds, colours and shifts amongst other variables. This meant that the goal of achieving a strong baseline accuracy proved arduous in comparison with models associated with the MNIST and Fashion-MNIST datasets.

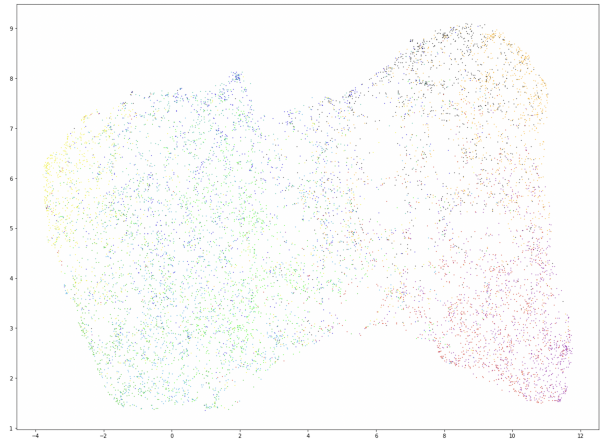
3.2.9 Increasing Embedding Dimensionality for Deep Metric Learning Models

An embedding network was employed to configure a higher dimensionality of the embeddings during training. As mentioned in [116], embeddings with an increased dimension lead to improved model performance. The trade-off present is that embeddings with a dimension of a significantly large value result in slower training times, therefore choosing the right dimensionality is a key question to refrain from pipelines that are too computationally expensive. For this study, this process was configured so that the dimensionality of the embeddings was equal to 64.

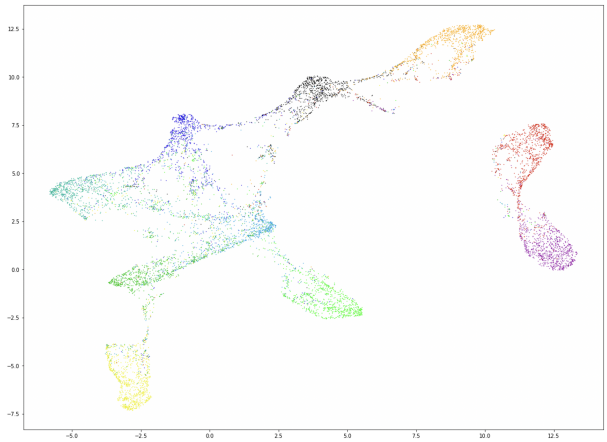
3.2.10 Classification Using Deep Metric Learning

As described in the background section, the classification process of Deep Metric Learning relies on the k-Nearest Neighbours in the training set of each testing image. The decision was made to experiment with various k values to get an in-depth understanding of how the classification accuracies change based on the set k value. The k values experimented with were $k = \{1, 3, 5\}$. When $k = 3$ and $k = 5$, if a draw occurred between classes during voting, then the input was deemed a misclassification. Furthermore, an important variable in the retrieval of samples is the distance metric employed, therefore a number of distance metrics were analysed. These were Cosine Similarity and the Euclidean Distance.

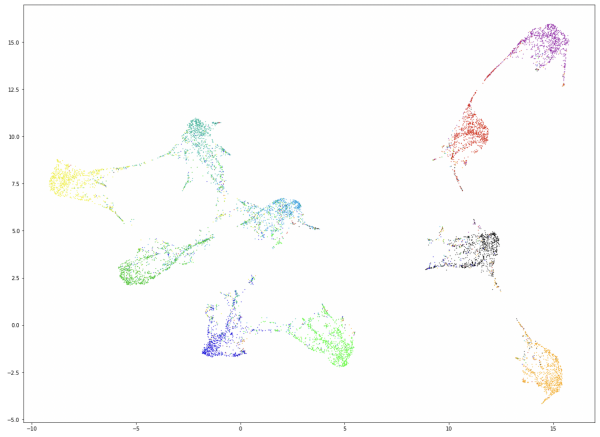
As shown in Figure 3.3, the projected embeddings of a trained model on CIFAR-10 with Deep Metric Learning are displayed. This visual representation of feature space conveys how the model projects the various inputs as training occurs. Each coloured cluster represents a class in the dataset. As shown in the diagram, certain clusters have more discriminative properties than others, which means that they are projected further from any of the other clusters. On the other hand, certain classes carry properties which in turn mean that they are quite similar to another class. This is represented by the coloured clusters which are close together, and where overlap is present.



(a) 1 Epoch



(b) 25 Epochs



(c) 50 Epochs

Figure 3.3: Changing Feature Space as Training Occurs: Deep Metric Learning model on CIFAR-10 with Triplet Loss, SGD as Optimisation Algorithm, Cosine Similarity as Distance metric.

3.3 Comparing Deep Learning and Deep Metric Learning

3.3.1 Comparison Metrics

As highlighted in the background section of this study, DL and DML have varying metrics with regards to measuring their performance as DL studies predominantly employ accuracy, and DML utilizes measurements such as Precision@K and Recall@K.

The decision was made to employ classification accuracy as the evaluation metric of each model as it is easily interpreted and leads to an immediate understanding of model robustness towards adversarial examples. In the case of DL, this was calculated by testing each model on a test set of images, and was calculated with

$$\frac{\text{Number of correctly classified samples}}{\text{Total number of samples classified}} \times 100$$

. For DML, the nearest neighbours of the test images were retrieved in the training set. If the k-Nearest Neighbours returned a classification similar to that of the test image, than the classification was deemed correct. Once this was done for all of the test images, the classification accuracy was calculated as

$$\frac{\text{k-NNs returning similar label to test image}}{\text{Total number of samples classified}} \times 100$$

3.4 Configuring the Adversarial Attacks

An important aspect in administering adversarial attacks efficiently and in an analytical fashion is to ensure that their parameters are controlled. For example, controlling the ϵ parameter within these attacks subjected the model to varying strengths of adversarial attack. The generated perturbations reflected the chosen ϵ value; as ϵ increased, so did the strength of the perturbation. In order to align the method with the goals of the study, attacks had to be strong as to generate perturbations which would confuse the trained models, allowing for a more global comparison of DL and DML.

The adversarial attacks employed in the experiment were the Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and Basic Iterative Method (BIM), all of which have been used extensively in previously published literature [106, 13, 71, 35]. Choosing the correct ϵ values to use for each adversarial attack was a question of vital importance, and therefore references to previously published literature are made. For all of the attacks, ϵ represented

the maximum l_∞ norm for the perturbed components. As used in [24] and in [56], FGSM and PGD attacks were configured so that $\epsilon = \{0.01, 0.02\}$. Similar ϵ values were employed for the BIM attacks. As employed in [107], the FGSM attacks were configured so that $\epsilon = 14$, and was therefore an appropriate ϵ to use for each of the attacks.

Chapter 4

Legal, Social, Ethical and Professional Issues

There exists a range of issues surrounding adversarial attacks and their potentially severe implications in differing contexts. These will now be broken down into Legal, Social, Ethical and Professional issues.

4.1 Legal Issues

An important issue to discuss relating to this study is the legality of the subject at hand; in particular model robustness, and how models can be prone to adversarial inputs. Without proper legislation, such systems may not be able to function properly due to the potential risk they cause. For example, if a driver-less car, an automated drone, or a missile defense system make a catastrophic error in response to an adversarial input, the following questions arise: who should be help accountable? What are the repercussions? How is this dealt with from a legal perspective? Governments must introduce legislation relating to model robustness in order to appropriately tackle these issues. A further point is that criminal activity will progress along with technology advancement, therefore the need to continue to invest in the advancement of technology within the legal framework will enable it to continue to build rather than go underground.

4.2 Social Issues

Systems without the necessary robustness can cause a wide range of issues within society. There is potential risk for a lack of trust from the general public to systems prone to adversarial attacks. Given this lack of trust, there can possibly be lessened adoption rates in differing contexts. This leads to slow progress and change within industry, which is ultimately unfavourable for the scientific community.

4.3 Ethical Issues

A number of ethical issues may arise when dealing with adversarial attacks and the robustness of Machine Learning models. The health and safety of systems in various domains is a vital entity. Furthermore, issues arise in the domain of privacy and confidentiality. Adversarial attacks, especially those with targeted properties, may harm these systems to the point of breaking any previously set confidentiality agreements. Furthermore, adversarial attacks can be employed in malicious procedures. Given a facial recognition software in a judicial context, someone may wrongfully be implicated for a crime due to a targeted adversarial attack. This also gives rise to the issue of discrimination, which can also be integrated into malicious activity, therefore it is imperative to ensure models are built with superior adversarial robustness.

4.4 Professional Issues

It is also imperative to mention that models prone to adversarial inputs or do not possess the necessary robustness, can lead to problems within the professional world. For example, a worrisome effect of the introduction of perturbed medical imagery for diagnosis of a particular disease or condition is that a patient may not receive the care they need. As examined in [24, 56] medical images are susceptible to various types of adversarial attacks with differing degrees of success which lead to significant misclassification. The identification of this potential issue in the medical community is a significant step forward, however training a model to be robust to adversarial inputs is of the upmost importance to resolve any inherent issues with diagnosis. Further to this, from a more business-like perspective, potential investors may have less confidence in Machine Learning systems, leading to a lack of interest from the business community. Moreover, governments may slow down the adoption and investment of said systems, even if the research has been complete, as there is an increased risk of models not being robust

enough. This leads to lost business opportunities, thus reducing competitiveness within varying industries, ultimately slowing down the advancement of Machine Learning technologies.

Chapter 5

Results/Evaluation

To reiterate, the aim of the present study was to identify which methodology, Deep Learning or Deep Metric Learning, can create models with superior adversarial robustness. To obtain results, the following pipeline was created. A training set was used to train a model and an adversarial attack was carried out on the samples in the test set. These adversarial examples were then fed into the trained model and predictions were recorded. If the model's prediction of the adversarial input matched the true label of the test sample than it was deemed correct, and if not then it was incorrect deeming the adversarial attack was successful. Having obtained the number of correct classifications, the classification accuracy of each model was deduced. Each model had various hyper-parameters such as the loss function, optimisation algorithm and learning rate amongst others which were tuned prior to training in order to analyse a variety of cases within DL and DML. Figures 5.1 through to 5.12 illustrate the results for the Deep Learning and Deep Metric Learning models respectively.

5.1 Evaluating Deep Learning Models

The results of the Deep Learning models follow a similar trend for MNIST, Fashion-MNIST and CIFAR-10, all of which can be found in Table 5.1. As the ϵ value increases with each adversarial attack, the classification accuracies of the models decrease. When $\epsilon = 14$, all of the PGD adversarial attacks were successful in perturbing the test samples and reducing the classification accuracies of each model to 0.

Table 5.1: Deep Learning Model Results

			MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples					
Cross-entropy	SGD	-	99.10	90.50	81.50
	Adam	-	99.10	90.60	77.40
$l_\infty(\epsilon = 0.01)$					
Cross-entropy	SGD	PGD	**98.53**	63.35	39.95
		BIM	98.49	59.98	41.80
		FGSM	98.52	63.99	25.83
	Adam	PGD	98.25	42.87	38.24
		BIM	98.23	38.42	38.99
		FGSM	98.30	50.53	30.25
$l_\infty(\epsilon = 0.02)$					
Cross-entropy	SGD	PGD	97.70	36.18	29.05
		BIM	97.61	32.10	33.14
		FGSM	97.75	46.71	14.00
	Adam	PGD	96.77	8.78	24.68
		BIM	96.56	7.00	38.99
		FGSM	97.20	28.61	14.36
$l_\infty(\epsilon = 14)$					
Cross-entropy	SGD	PGD	0.00	0.00	0.00
		BIM	0.00	0.02	0.00
		FGSM	7.49	4.66	0.00
	Adam	PGD	0.00	0.00	0.00
		BIM	1.02	0.21	0.00
		FGSM	12.30	3.86	0.00

5.2 Evaluating Deep Metric Learning Models

For the majority of trained models, as the ϵ parameter in each adversarial attack grew, the accuracies diminished.

It was found that altering k when retrieving the nearest neighbours did not affect the classification accuracy significantly. The accuracies followed similar trends as ϵ increased for each adversarial attack, but did not change by a sigificant amount when k was altered.

In the case of CIFAR-10, models trained with Contrastive Loss produced results of a superior quality than those which employed Triplet Loss. This was a consistent trend across all k values, suggesting that Contrastive Loss is more appropriate for adversarial robustness for this particular dataset. Having said this, when examining the results associated with MNIST, the opposite result was found. Models employing Triplet Loss were significantly more robust when $\epsilon = 14$ (the harshest perturbation), in contrast to models trained with the Contrastive Loss Function.

Across all of the trained models, the FGSM attack was the least successful where models

produced results with the highest robustness to adversarial inputs generated with said attack. Upon observation of the results, the most successful attack was found to be PGD, especially when $\epsilon = 0.02$ or $\epsilon = 14$. The BIM attack was found to be somewhere between the success rate of FGSM and PGD, and could comfortably be ranked as the second most successful adversarial attack in the experiment.

5.3 Comparison of Deep Learning and Deep Metric Learning Methods

The attention now turns to the following research question: Which methodology produced the more robust models, on what data and what were its parameters? To facilitate the most optimal comparison between DL and DML, each of the three datasets used must be analysed in isolation as their baseline accuracies are not similar, and it would be impractical to suggest that one method is better if the accuracies are taken from the testing of a different dataset. Figures X through X illustrate the achieved accuracies of both DL and DML models on MNIST, Fashion-MNIST and CIFAR-10. Each figure depicts the results following the PGD, BIM and FGSM adversarial attacks with varying ϵ values. The DL models are represented by the purple columns, whereas DML is illustrated through the green columns. The general trend that was identified is that models trained with DML were more robust to those trained with DL methodologies. As observed in these figures, the columns representing DML are consistently higher than those pertaining to DL, and this is consistent across all ϵ values. This is particularly apparent in the figures relating to Fashion-MNIST and CIFAR-10. A small number of exceptions were found contrasting the overall general trend. When trained with Triplet Loss and Adam, the DML model performed worse at $\epsilon = 0.01$ for the BIM adversarial attack. Further to this, when trained with Triplet Loss and SGD, the DML model had lower accuracies at $\epsilon = 0.01$ and $\epsilon = 0.02$ for PGD and BIM. However, these were the only significant exceptions found, therefore it can be concluded that in general, the DML models were more robust to adversarial examples. The results for each of the trained models are found in Tables 5.2 through to 5.7.

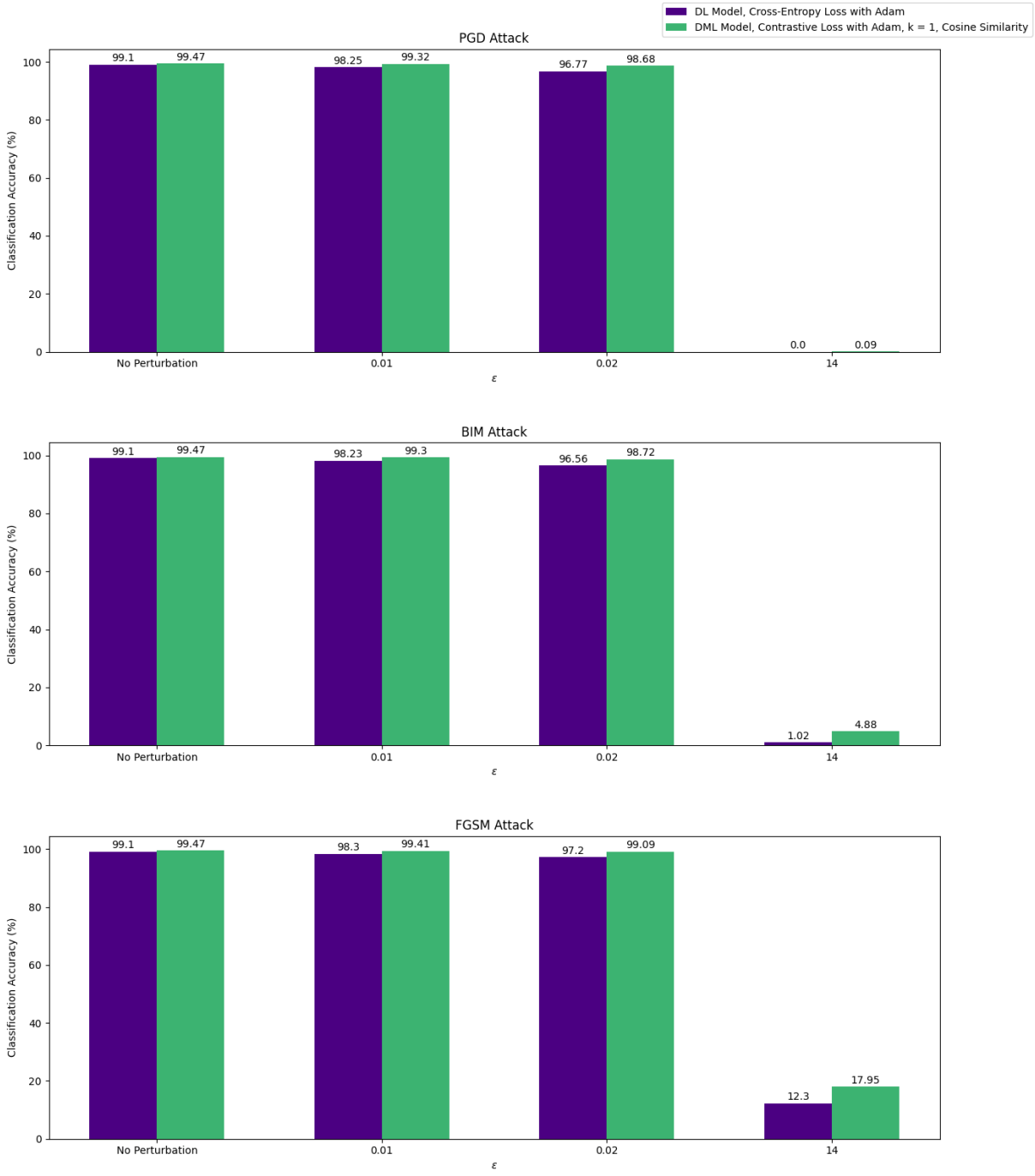


Figure 5.1: MNIST: DL (Cross-Entropy) vs DML (Contrastive Loss, k = 1, Cosine Similarity) - trained with Adam Optimisation Algorithm. DL results are in purple, and DML in green.

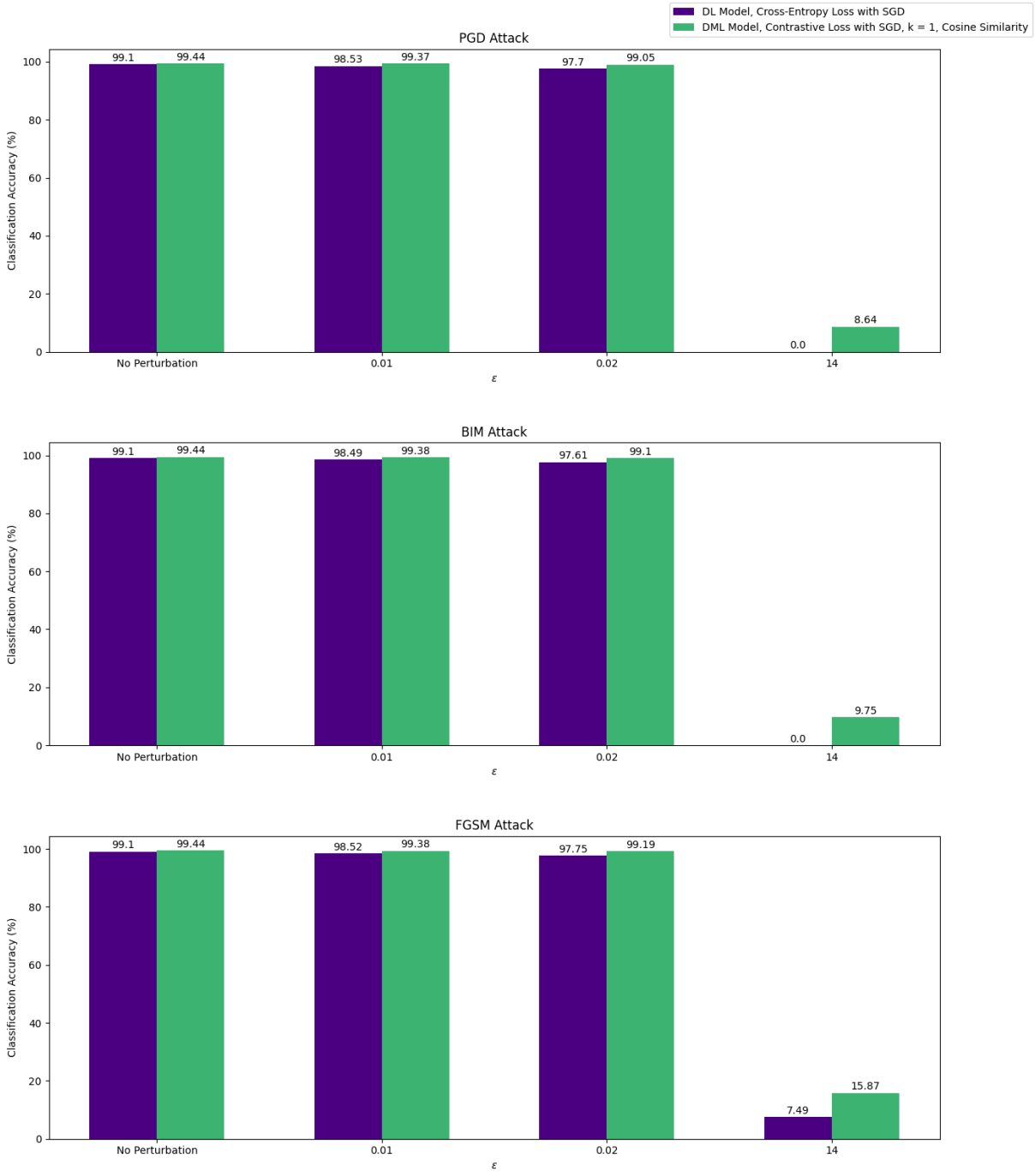


Figure 5.2: MNIST: DL (Cross-Entropy) vs DML (Contrastive Loss, k = 1, Cosine Similarity) - trained with SGD Optimisation Algorithm. DL results are in purple, and DML in green.

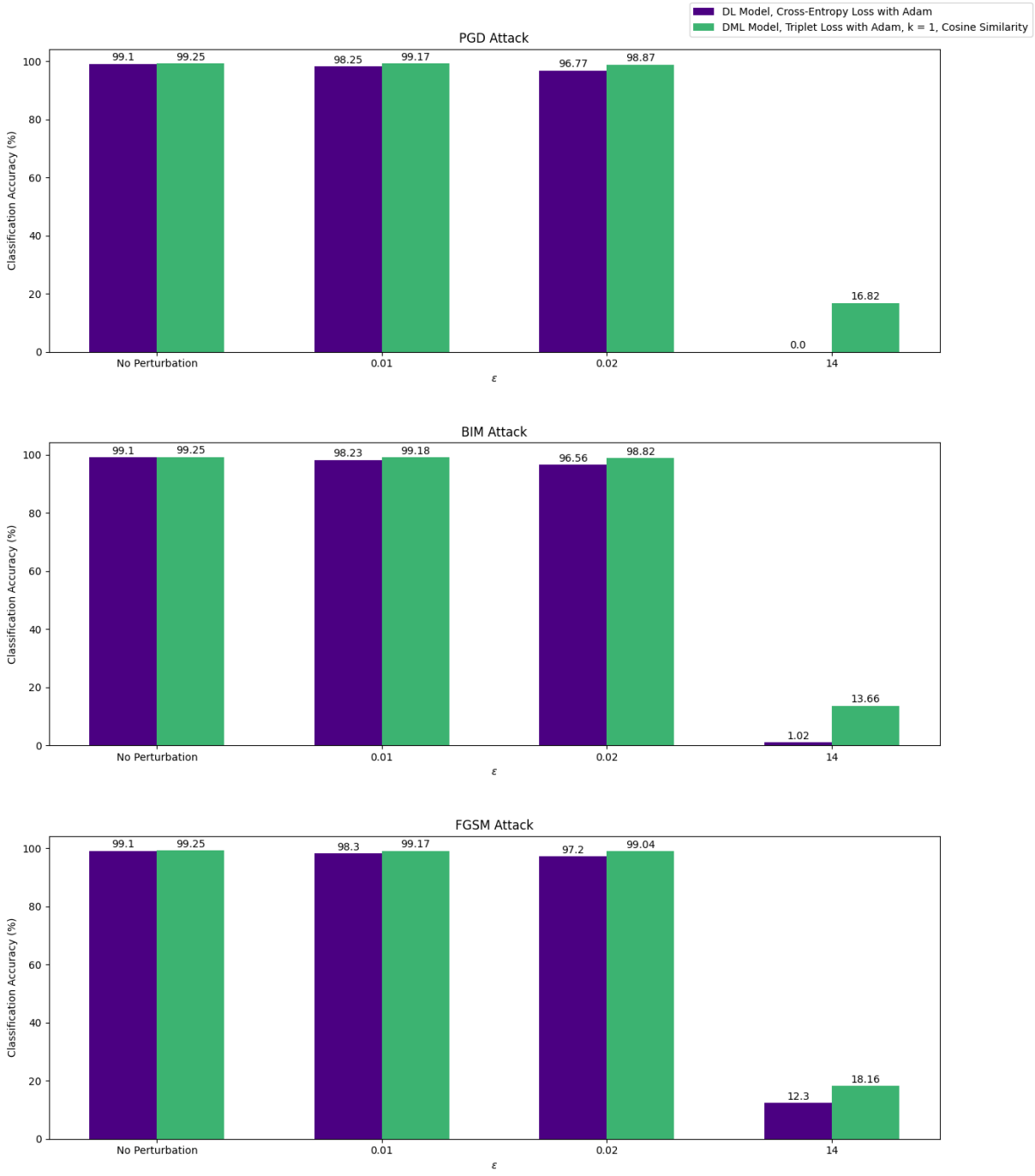


Figure 5.3: MNIST: DL (Cross-Entropy) vs DML (Triplet Loss, k = 1, Cosine Similarity) - trained with Adam Optimisation Algorithm. DL results are in purple, and DML in green.

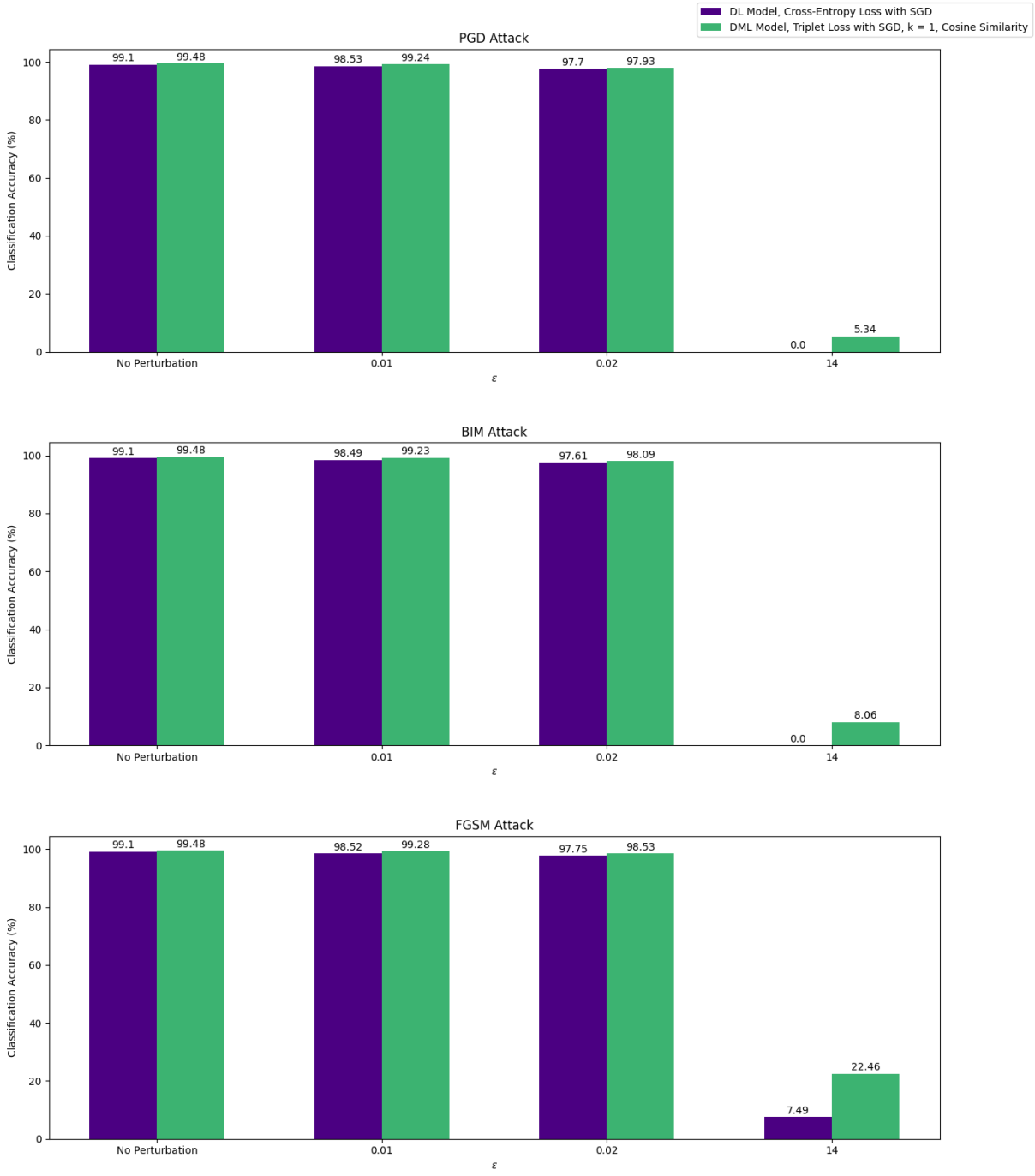


Figure 5.4: MNIST: DL (Cross-Entropy) vs DML (Triplet Loss, k = 1, Cosine Similarity) - trained with SGD Optimisation Algorithm. DL results are in purple, and DML in green.

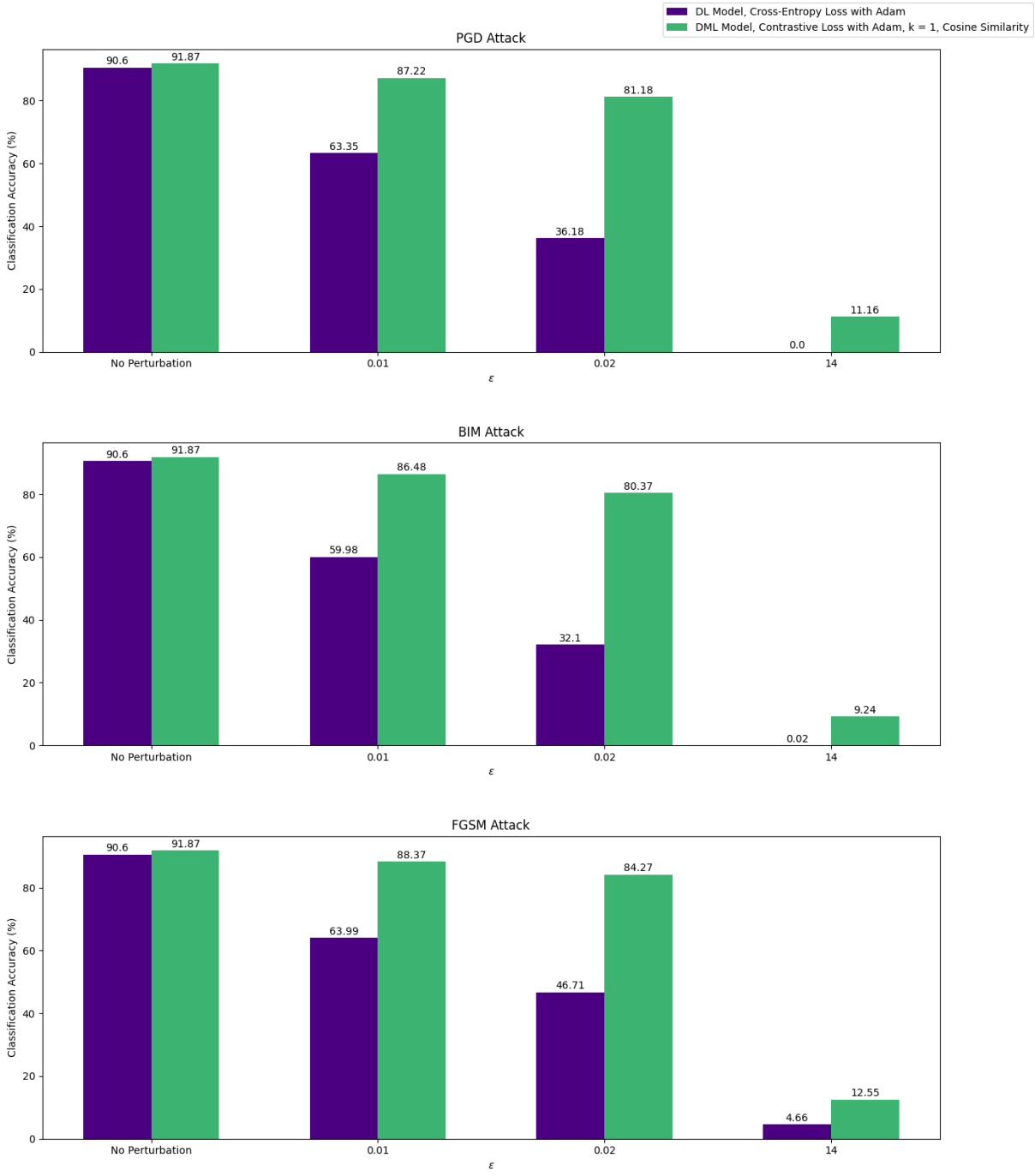


Figure 5.5: Fashion-MNIST: DL (Cross-Entropy) vs DML (Contrastive Loss, k = 1, Cosine Similarity) - trained with Adam Optimisation Algorithm. DL results are in purple, and DML in green.

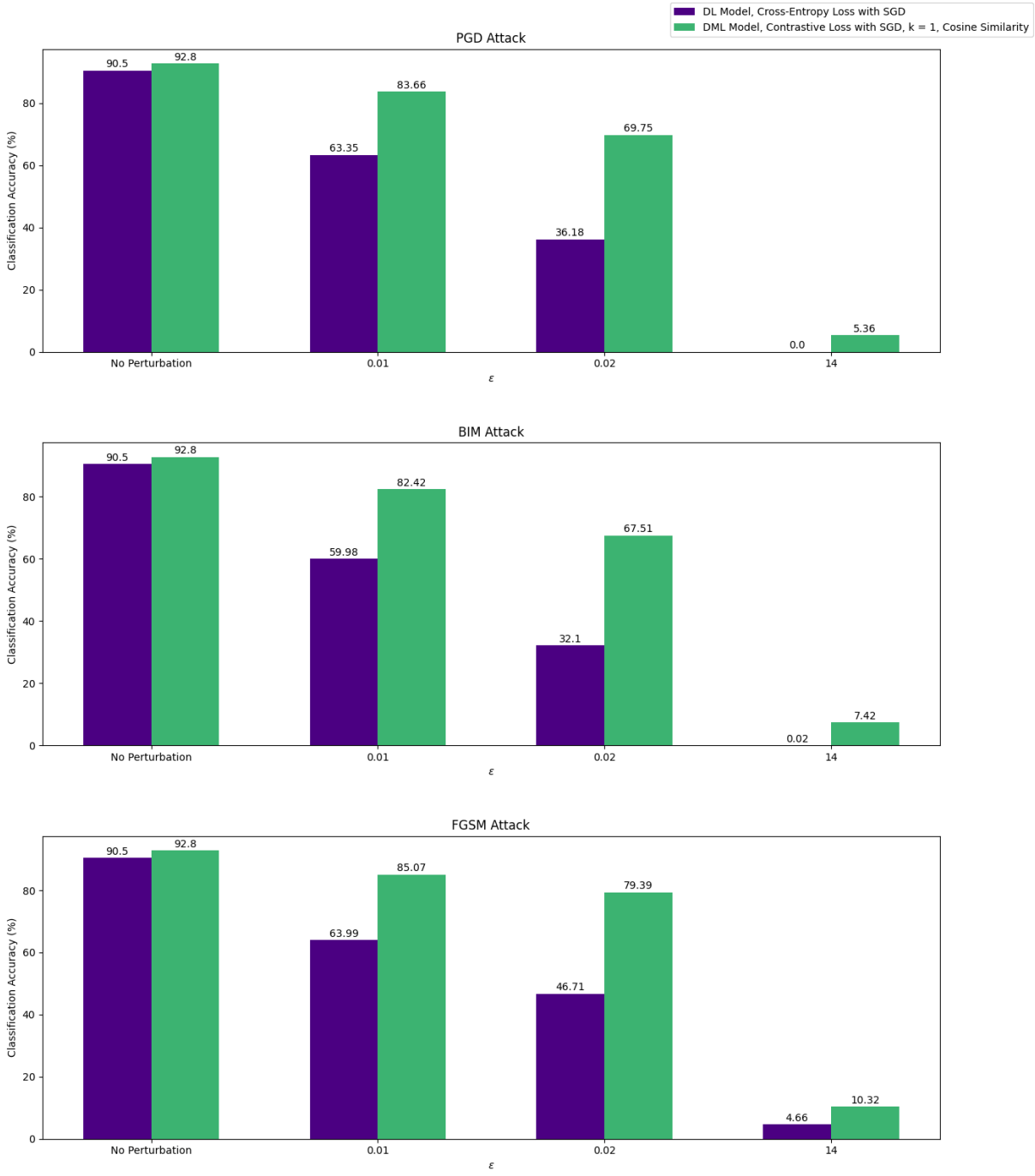


Figure 5.6: Fashion-MNIST: DL (Cross-Entropy) vs DML (Contrastive Loss, k = 1, Cosine Similarity) - trained with SGD Optimisation Algorithm. DL results are in purple, and DML in green.

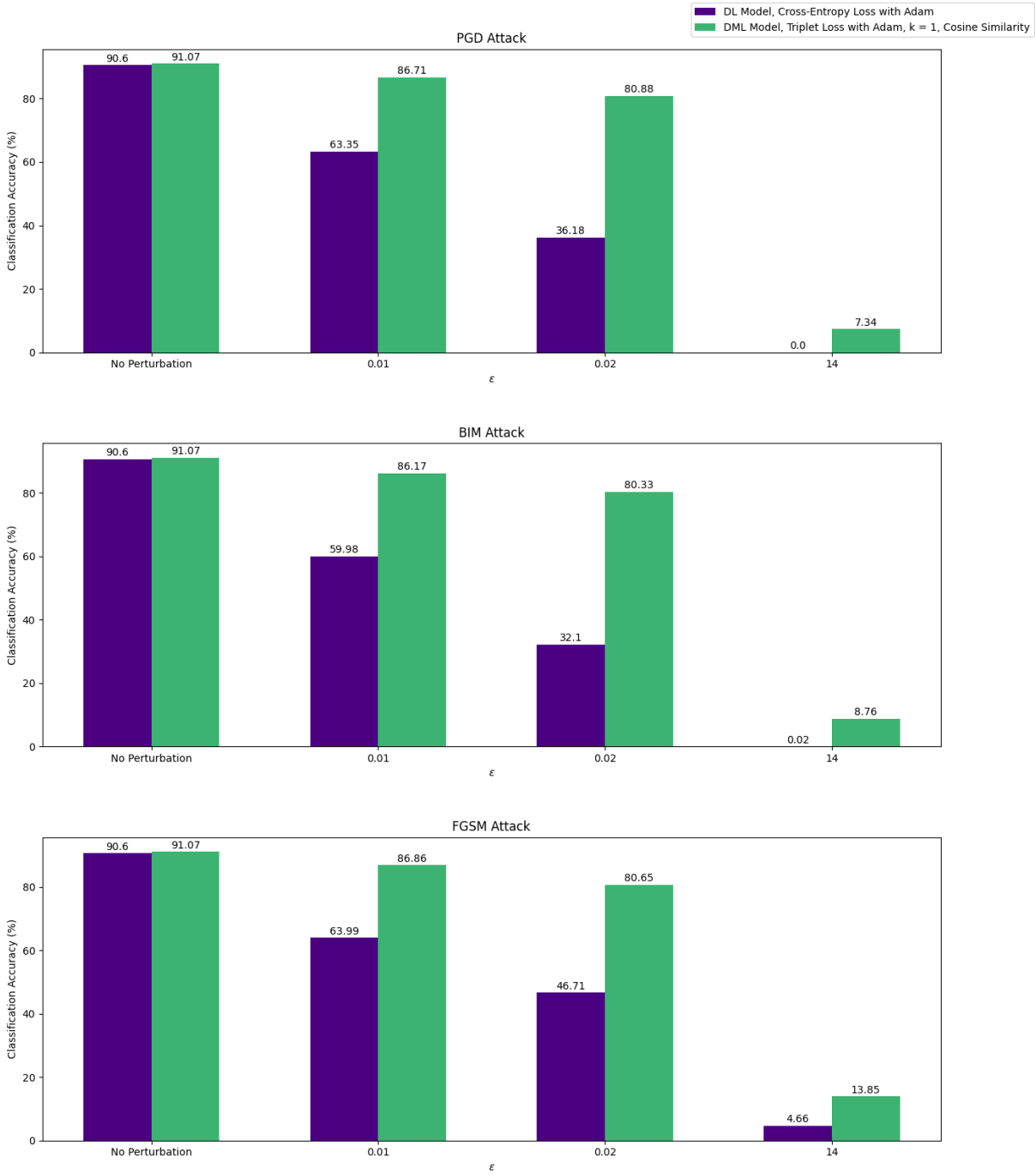


Figure 5.7: Fashion-MNIST: DL (Cross-Entropy) vs DML (Triplet Loss, k = 1, Cosine Similarity) - trained with Adam Optimisation Algorithm. DL results are in purple, and DML in green.

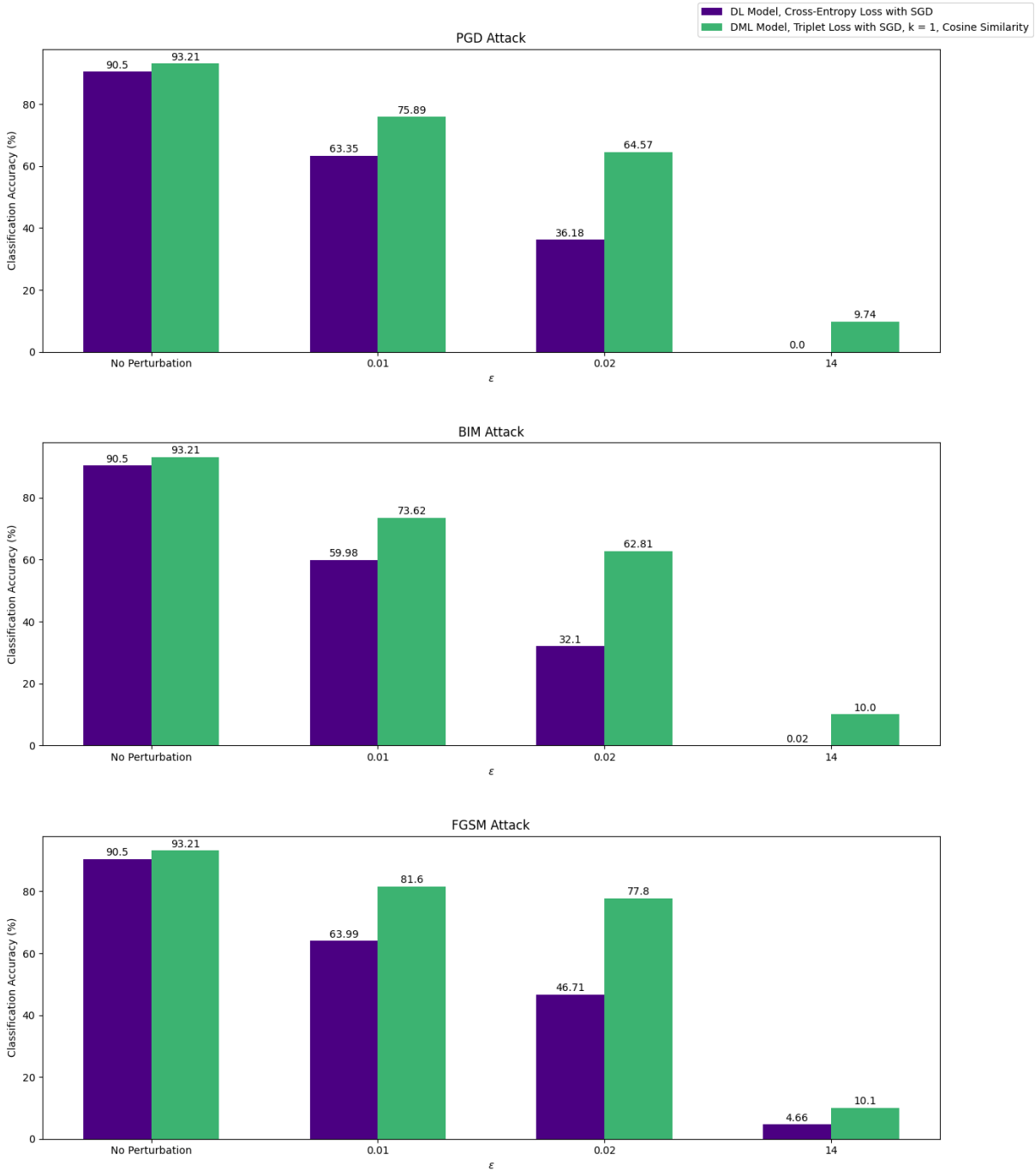


Figure 5.8: Fashion-MNIST: DL (Cross-Entropy) vs DML (Triplet Loss, k = 1, Cosine Similarity) - trained with SGD Optimisation Algorithm. DL results are in purple, and DML in green.

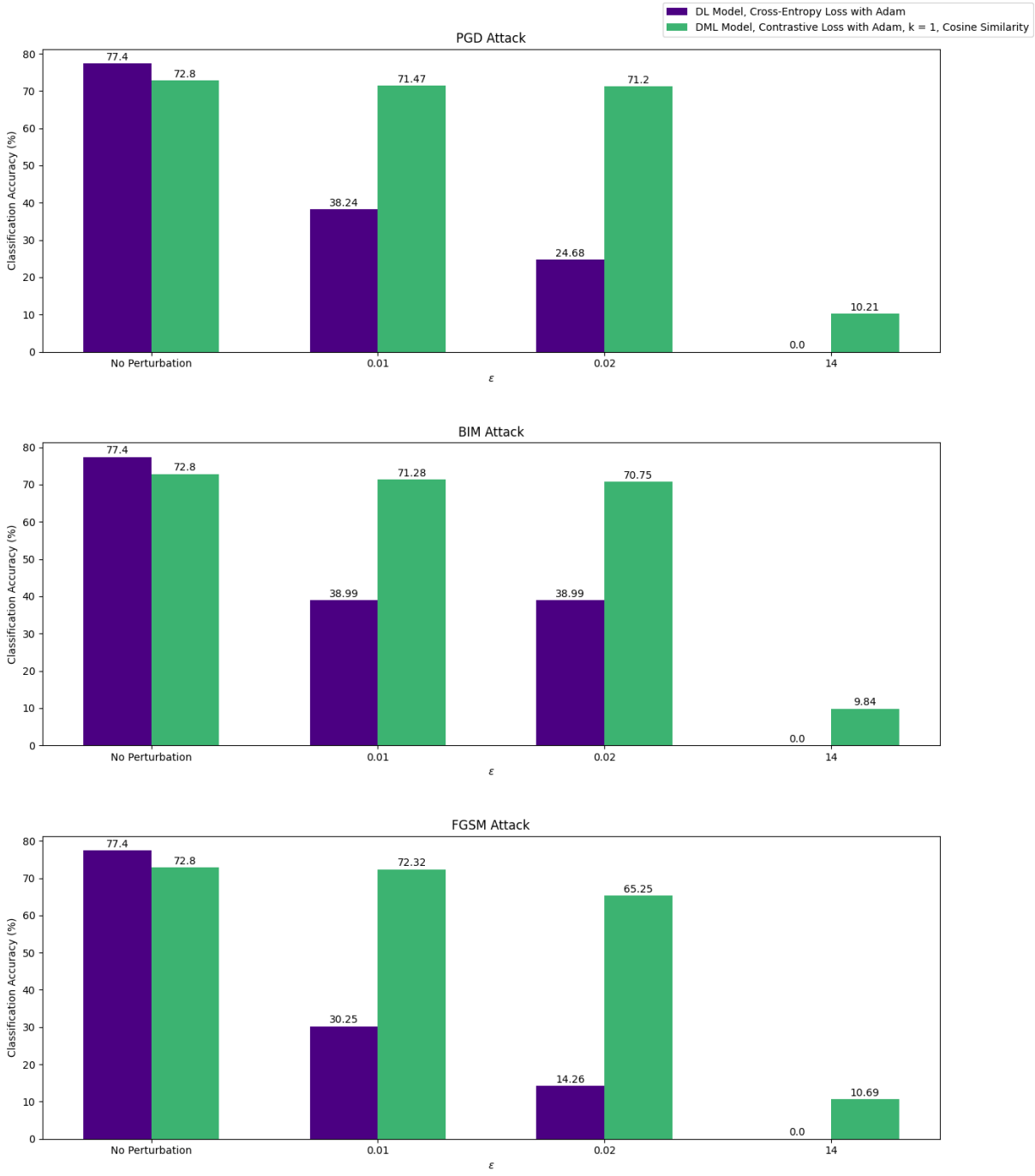


Figure 5.9: CIFAR-10: DL (Cross-Entropy) vs DML (Contrastive Loss, k = 1, Cosine Similarity) - trained with Adam Optimisation Algorithm. DL results are in purple, and DML in green.

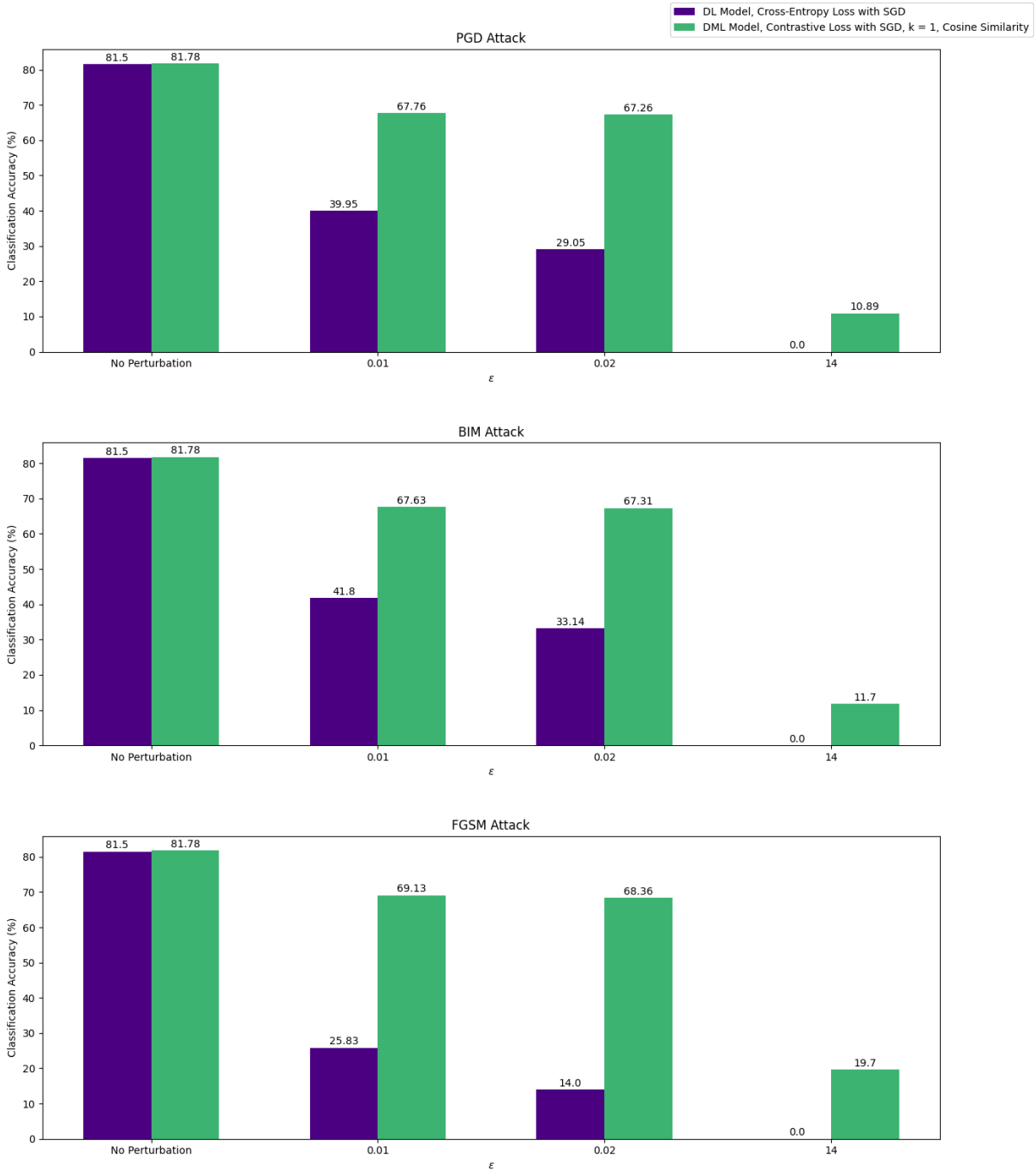


Figure 5.10: CIFAR-10: DL (Cross-Entropy) vs DML (Contrastive Loss, k = 1, Cosine Similarity) - trained with SGD Optimisation Algorithm. DL results are in purple, and DML in green.

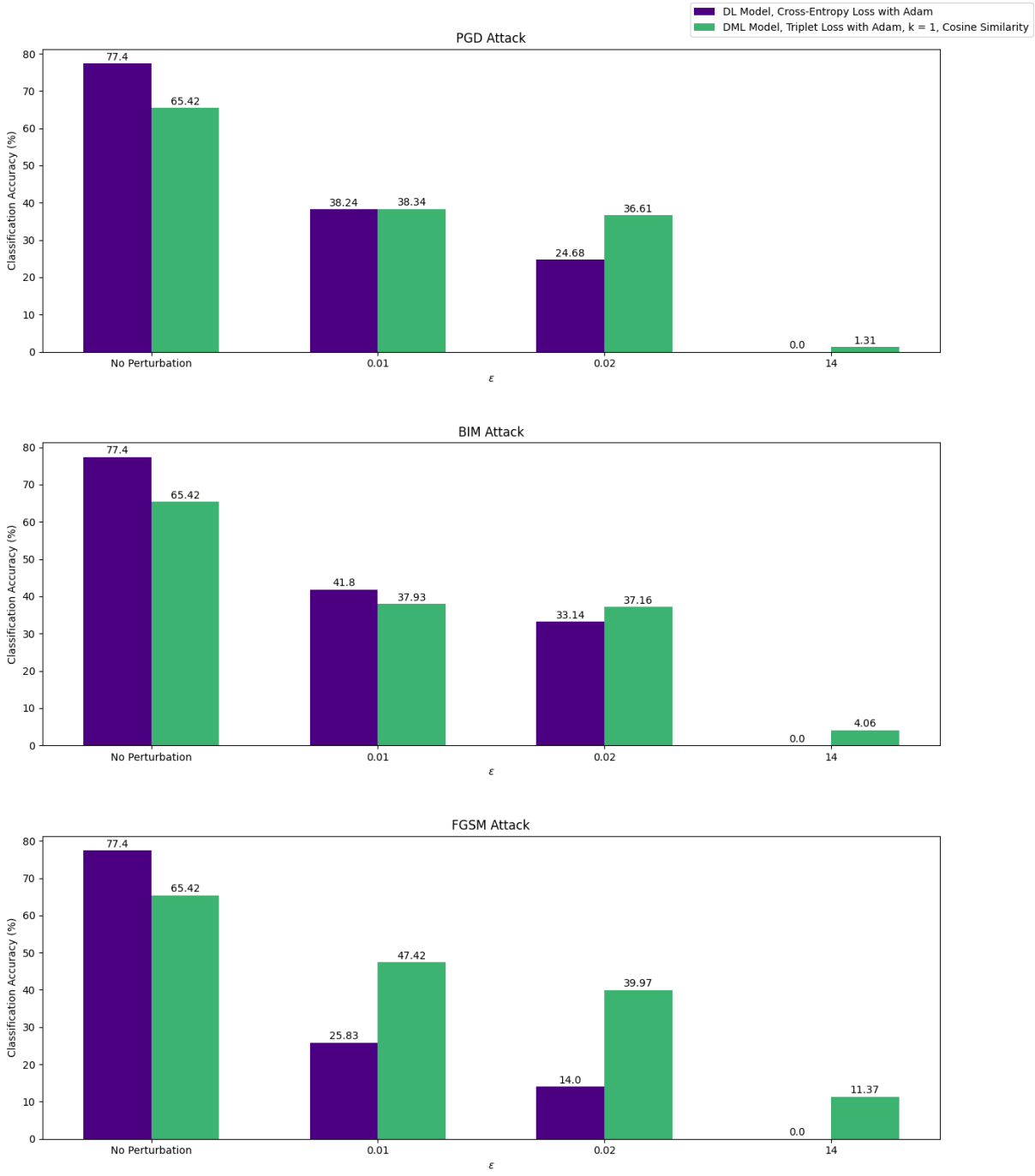


Figure 5.11: CIFAR-10: DL (Cross-Entropy) vs DML (Triplet Loss, k = 1, Cosine Similarity) - trained with Adam Optimisation Algorithm. DL results are in purple, and DML in green.

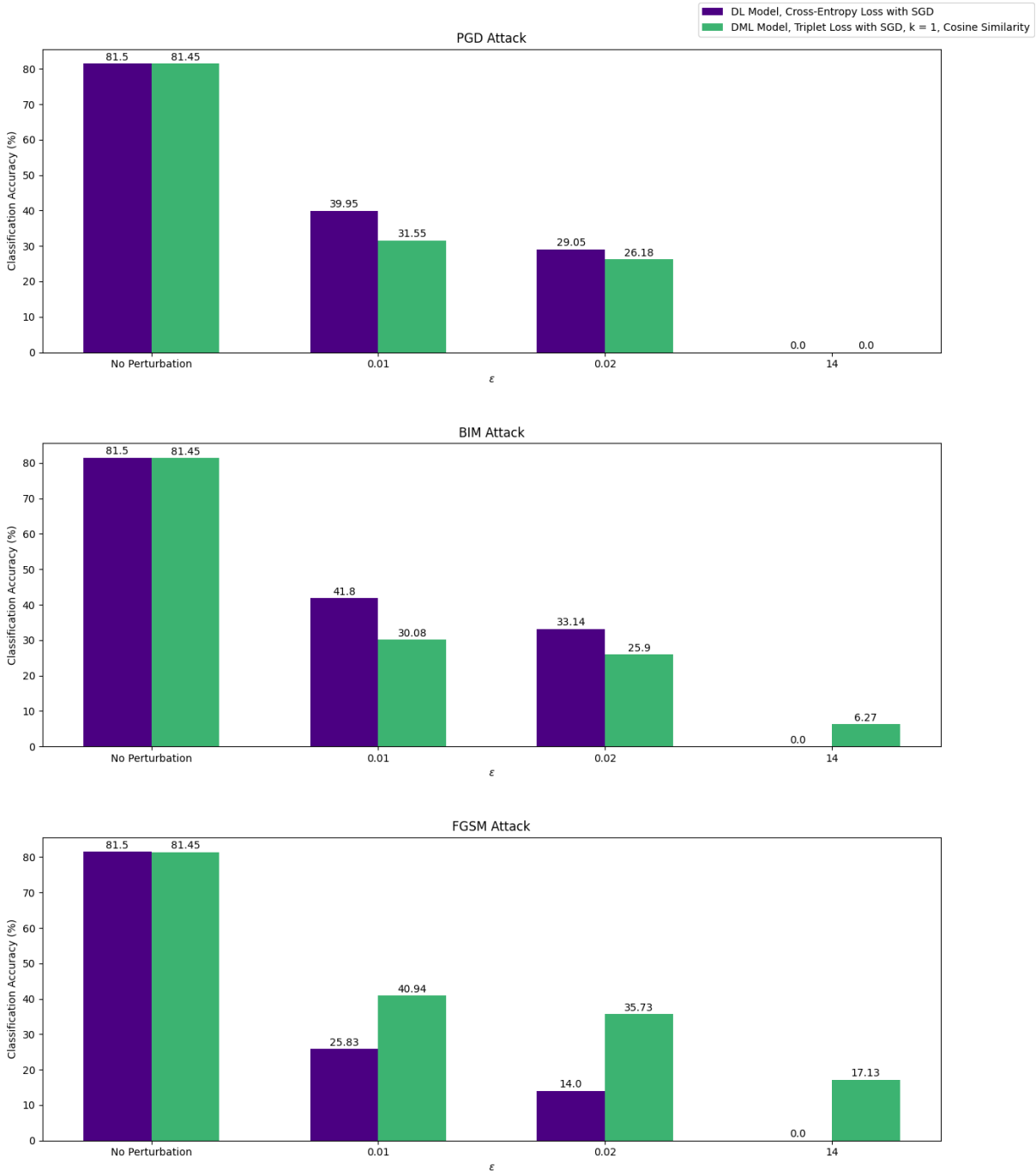


Figure 5.12: CIFAR-10: DL (Cross-Entropy) vs DML (Triplet Loss, k = 1, Cosine Similarity) - trained with SGD Optimisation Algorithm. DL results are in purple, and DML in green.

The effects of using a changed optimisation algorithm varied throughout the results. When looking at all three datasets, the general trend was that Adam produced results that were superior to those trained with SGD by a small margin. However, in certain cases, models trained with SGD were better at recognising perturbations generated with the FGSM attack, but this was not always the case. Therefore, it was not possible to suggest which optimisation

algorithm was more beneficial to create models with adversarial robustness against the FGSM attack.

A general property identified through the results is that not only is DML better at creating models with improved adversarial robustness, but this methodology also created models with a better baseline accuracy on benign samples. This suggests that DML is superior at creating classification models on image data in comparison to DL methodologies.

Table 5.2: Deep Metric Learning Model Results: Contrastive Loss, $k = 1$

				MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Distance	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples						
Contrastive	SGD	Cosine	-	99.44	92.80	81.78
		Euclidean	-	99.44	92.80	81.79
	Adam	Cosine	-	99.47	91.87	72.80
		Euclidean	-	99.47	91.87	72.80
$l_\infty(\epsilon = 0.01)$						
Contrastive	SGD	Cosine	PGD	99.37	83.66	67.76
			BIM	99.38	82.42	67.63
			FGSM	99.38	85.07	69.13
		Euclidean	PGD	99.37	83.74	67.89
			BIM	99.38	82.42	97.73
			FGSM	99.38	85.07	69.13
	Adam	Cosine	PGD	99.32	87.22	71.47
			BIM	99.30	86.48	71.28
			FGSM	99.41	88.37	72.32
		Euclidean	PGD	99.32	87.24	71.51
			BIM	99.30	86.48	71.28
			FGSM	99.41	88.37	72.32
$l_\infty(\epsilon = 0.02)$						
Contrastive	SGD	Cosine	PGD	99.05	69.75	67.26
			BIM	99.10	67.51	67.31
			FGSM	99.19	79.39	68.36
		Euclidean	PGD	99.08	69.15	67.34
			BIM	99.10	67.51	67.31
			FGSM	99.19	79.39	68.36
	Adam	Cosine	PGD	98.68	81.18	71.20
			BIM	98.72	80.37	70.75
			FGSM	99.09	84.27	65.25
		Euclidean	PGD	98.72	81.36	71.12
			BIM	98.72	80.37	70.75
			FGSM	99.09	84.27	65.25
$l_\infty(\epsilon = 14)$						
Contrastive	SGD	Cosine	PGD	8.64	5.36	10.89
			BIM	9.75	7.42	11.70
			FGSM	15.87	10.32	19.70
		Euclidean	PGD	8.62	5.19	10.85
			BIM	9.75	7.42	11.70
			FGSM	15.87	10.32	19.70
	Adam	Cosine	PGD	0.09	11.16	10.21
			BIM	4.88	9.24	9.84
			FGSM	17.95	12.55	10.69
		Euclidean	PGD	0.13	11.03	10.20
			BIM	4.88	9.25	9.84
			FGSM	17.95	12.55	10.69

Table 5.3: Deep Metric Learning Model Results: Contrastive Loss, $k = 3$

				MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Distance	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples						
Contrastive	SGD	Cosine	-	99.50	93.20	81.94
		Euclidean	-	99.50	93.20	81.94
	Adam	Cosine	-	99.49	92.34	74.14
		Euclidean	-	99.49	92.34	74.14
$l_\infty(\epsilon = 0.01)$						
Contrastive	SGD	Cosine	PGD	99.41	83.51	67.76
			BIM	99.40	82.19	67.59
			FGSM	99.41	85.16	69.13
		Euclidean	PGD	99.41	83.69	67.87
			BIM	99.40	82.19	67.59
			FGSM	99.41	85.16	69.13
	Adam	Cosine	PGD	99.36	87.87	71.67
			BIM	99.36	87.30	71.56
			FGSM	99.41	88.99	72.52
		Euclidean	PGD	99.36	87.73	71.68
			BIM	99.36	87.30	71.56
			FGSM	99.41	88.99	72.52
$l_\infty(\epsilon = 0.02)$						
Contrastive	SGD	Cosine	PGD	99.04	69.75	67.27
			BIM	98.99	67.86	67.31
			FGSM	99.23	79.40	68.32
		Euclidean	PGD	99.02	69.51	67.36
			BIM	98.99	67.86	67.30
			FGSM	99.23	79.40	68.32
	Adam	Cosine	PGD	98.91	82.32	71.38
			BIM	98.87	81.33	70.95
			FGSM	99.15	85.17	65.60
		Euclidean	PGD	98.90	82.20	71.17
			BIM	98.87	81.33	70.95
			FGSM	99.15	85.17	64.60
$l_\infty(\epsilon = 14)$						
Contrastive	SGD	Cosine	PGD	8.82	9.67	10.91
			BIM	9.74	5.94	11.70
			FGSM	15.49	10.14	19.69
		Euclidean	PGD	8.71	9.54	10.89
			BIM	9.74	5.94	11.70
			FGSM	15.49	10.14	19.69
	Adam	Cosine	PGD	0.06	10.50	10.00
			BIM	5.02	9.24	9.99
			FGSM	18.16	11.96	10.64
		Euclidean	PGD	0.09	10.38	10.02
			BIM	5.02	9.24	9.99
			FGSM	18.16	11.96	10.64

Table 5.4: Deep Metric Learning Model Results: Contrastive Loss, $k = 5$

				MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Distance	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples						
Contrastive	SGD	Cosine	-	99.47	93.05	81.87
		Euclidean	-	99.47	93.05	81.87
	Adam	Cosine	-	99.47	91.89	71.34
		Euclidean	-	99.47	91.89	71.34
$l_\infty(\epsilon = 0.01)$						
Contrastive	SGD	Cosine	PGD	99.36	83.51	67.76
			BIM	99.36	81.94	67.58
			FGSM	99.36	85.18	69.06
		Euclidean	PGD	99.37	83.69	67.85
			BIM	99.36	81.94	67.58
			FGSM	99.36	85.18	69.06
	Adam	Cosine	PGD	99.35	87.59	71.49
			BIM	99.36	86.55	71.42
			FGSM	99.37	88.61	71.69
		Euclidean	PGD	99.36	87.25	71.57
			BIM	99.36	86.55	71.42
			FGSM	99.37	88.61	71.69
$l_\infty(\epsilon = 0.02)$						
Contrastive	SGD	Cosine	PGD	99.09	69.77	67.29
			BIM	99.00	68.07	67.31
			FGSM	99.23	79.29	68.36
		Euclidean	PGD	99.08	69.41	67.36
			BIM	99.00	68.07	67.31
			FGSM	99.23	79.29	68.36
	Adam	Cosine	PGD	98.89	80.94	71.24
			BIM	98.83	80.14	70.85
			FGSM	99.12	84.26	64.21
		Euclidean	PGD	98.86	81.11	71.08
			BIM	98.83	80.14	70.85
			FGSM	99.12	84.26	64.21
$l_\infty(\epsilon = 14)$						
Contrastive	SGD	Cosine	PGD	8.89	7.76	10.92
			BIM	9.74	5.03	11.66
			FGSM	15.22	10.11	19.67
		Euclidean	PGD	8.85	7.88	10.90
			BIM	9.74	5.03	11.66
			FGSM	15.22	10.11	19.67
	Adam	Cosine	PGD	0.06	9.54	10.00
			BIM	4.70	8.83	9.96
			FGSM	17.27	10.81	10.37
		Euclidean	PGD	0.08	9.48	10.02
			BIM	4.70	8.83	9.96
			FGSM	17.27	10.81	10.37

Table 5.5: Deep Metric Learning Model Results: Triplet Loss, $k = 1$

				MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Distance	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples						
Triplet	SGD	Cosine	-	99.48	93.21	81.45
		Euclidean	-	99.48	93.21	81.45
	Adam	Cosine	-	99.25	91.07	65.42
		Euclidean	-	99.25	91.07	65.42
$l_\infty(\epsilon = 0.01)$						
Triplet	SGD	Cosine	PGD	99.24	75.89	31.55
			BIM	99.23	73.62	30.08
			FGSM	99.28	81.60	40.94
		Euclidean	PGD	99.25	75.75	31.45
			BIM	99.23	73.62	30.08
			FGSM	99.28	81.60	40.94
	Adam	Cosine	PGD	99.17	86.71	38.34
			BIM	99.18	86.17	37.93
			FGSM	99.17	86.86	47.42
		Euclidean	PGD	99.16	86.82	38.25
			BIM	99.18	86.17	37.93
			FGSM	99.17	86.86	47.42
$l_\infty(\epsilon = 0.02)$						
Triplet	SGD	Cosine	PGD	97.93	64.57	26.18
			BIM	98.09	62.81	25.90
			FGSM	98.53	77.80	35.73
		Euclidean	PGD	97.82	64.65	26.13
			BIM	98.09	62.81	25.90
			FGSM	98.53	77.80	35.73
	Adam	Cosine	PGD	98.87	80.88	36.61
			BIM	98.82	80.33	37.16
			FGSM	99.04	80.65	39.97
		Euclidean	PGD	98.86	81.01	36.44
			BIM	98.82	80.33	37.16
			FGSM	99.04	80.65	39.97
$l_\infty(\epsilon = 14)$						
Triplet	SGD	Cosine	PGD	5.34	9.74	0.00
			BIM	8.06	10.00	6.27
			FGSM	22.46	10.10	17.13
		Euclidean	PGD	5.43	9.77	0.01
			BIM	8.06	10.00	6.29
			FGSM	22.46	10.10	17.13
	Adam	Cosine	PGD	16.82	7.34	1.31
			BIM	13.66	8.76	4.06
			FGSM	18.16	13.85	11.37
		Euclidean	PGD	16.79	6.81	1.13
			BIM	13.66	8.76	4.06
			FGSM	18.16	13.85	11.37

Table 5.6: Deep Metric Learning Model Results: Triplet Loss, $k = 3$

				MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Distance	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples						
Triplet	SGD	Cosine	-	99.45	93.27	81.44
		Euclidean	-	99.45	93.27	81.44
	Adam	Cosine	-	99.40	92.04	65.56
		Euclidean	-	99.40	92.04	65.56
$l_\infty(\epsilon = 0.01)$						
Triplet	SGD	Cosine	PGD	99.25	75.75	31.48
			BIM	99.26	73.44	30.03
			FGSM	99.27	81.79	40.92
		Euclidean	PGD	99.26	75.66	31.45
			BIM	99.26	73.44	30.03
			FGSM	99.27	81.79	40.92
	Adam	Cosine	PGD	99.31	87.16	38.60
			BIM	99.31	86.67	38.17
			FGSM	99.30	87.62	47.23
		Euclidean	PGD	99.32	87.23	38.44
			BIM	99.31	86.67	38.17
			FGSM	99.30	87.62	47.23
$l_\infty(\epsilon = 0.02)$						
Triplet	SGD	Cosine	PGD	97.89	64.65	26.18
			BIM	97.99	62.65	25.93
			FGSM	98.59	77.66	35.64
		Euclidean	PGD	97.88	64.74	26.12
			BIM	97.99	62.95	25.92
			FGSM	98.59	77.66	35.64
	Adam	Cosine	PGD	99.05	81.80	36.70
			BIM	99.02	81.41	37.20
			FGSM	99.12	81.85	39.80
		Euclidean	PGD	99.03	81.06	36.60
			BIM	99.02	81.41	37.20
			FGSM	99.12	81.85	39.80
$l_\infty(\epsilon = 14)$						
Triplet	SGD	Cosine	PGD	7.92	9.76	0.00
			BIM	7.91	10.00	6.27
			FGSM	22.50	10.05	17.13
		Euclidean	PGD	7.48	9.78	0.01
			BIM	7.97	10.00	6.28
			FGSM	22.50	10.05	16.89
	Adam	Cosine	PGD	16.72	7.23	1.40
			BIM	13.86	6.76	1.64
			FGSM	17.99	13.85	9.48
		Euclidean	PGD	16.74	6.67	1.38
			BIM	13.86	6.76	1.63
			FGSM	17.99	12.65	9.48

Table 5.7: Deep Metric Learning Model Results: Triplet Loss, $k = 5$

				MNIST	**Fashion-MNIST**	**CIFAR-10**
Loss Function	Optimizer	Distance	Adversarial Attack	Classification Accuracy (%)		
No Pertubation: Benign Samples						
Triplet	SGD	Cosine	-	99.50	93.24	81.24
		Euclidean	-	99.50	93.24	81.24
	Adam	Cosine	-	99.37	91.68	60.50
		Euclidean	-	99.37	91.68	60.50
$l_\infty(\epsilon = 0.01)$						
Triplet	SGD	Cosine	PGD	99.26	75.60	31.18
			BIM	99.25	73.33	29.78
			FGSM	99.25	81.71	40.68
		Euclidean	PGD	99.26	75.54	31.09
			BIM	99.25	73.33	29.78
			FGSM	99.25	81.71	40.68
	Adam	Cosine	PGD	99.29	86.51	38.26
			BIM	99.28	86.03	37.78
			FGSM	99.28	87.03	44.76
		Euclidean	PGD	99.27	86.72	38.11
			BIM	99.28	86.03	37.78
			FGSM	99.28	87.03	44.76
$l_\infty(\epsilon = 0.02)$						
Triplet	SGD	Cosine	PGD	97.79	63.81	26.09
			BIM	97.94	62.28	25.71
			FGSM	98.58	77.55	35.67
		Euclidean	PGD	97.76	63.94	26.22
			BIM	97.94	62.28	25.71
			FGSM	98.58	77.55	35.67
	Adam	Cosine	PGD	99.09	81.13	36.48
			BIM	99.00	80.68	36.98
			FGSM	99.10	81.10	39.80
		Euclidean	PGD	99.04	81.06	36.36
			BIM	99.00	80.68	36.98
			FGSM	99.10	81.10	36.49
$l_\infty(\epsilon = 14)$						
Triplet	SGD	Cosine	PGD	7.21	9.77	0.00
			BIM	7.67	10.00	6.13
			FGSM	22.44	10.06	16.73
		Euclidean	PGD	6.80	9.78	0.00
			BIM	7.71	10.00	6.14
			FGSM	22.44	10.06	16.73
	Adam	Cosine	PGD	16.44	6.85	1.32
			BIM	14.12	5.48	4.06
			FGSM	17.65	10.36	7.09
		Euclidean	PGD	16.21	6.11	1.31
			BIM	14.12	5.48	0.78
			FGSM	17.65	10.36	7.09

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The present study aimed to find the advantageous methodology between Deep Learning and Deep Metric Learning in creating models which are more robust to adversarial examples. CNN models with varying parameters were configured for the MNIST, Fashion-MNIST and CIFAR-10 datasets, and were used in both a DL and DML training capacity. Following the study's implementation and analysis of results, it was concluded that Deep Metric Learning, with the help of various metric losses, is superior to Deep Learning in creating robust models to adversarial examples.

6.2 Implications

This study allowed for an in-depth exploration into many of the different parameters within Deep Learning, and subsequently allowed for an introduction into the rising field of Deep Metric Learning. This sanctioned an intriguing literature review, one which inspired a number of procedures within the implementation of this study. The capabilities surrounding the embeddings onto a new feature space with a measure of distance within DML allowed to better understand how samples are interrelated, as well as which classes in data are more or less different. I believe that the results show that Deep Metric Learning is a strong contender to be an optimal strategy within Machine Learning applications for sample retrieval, especially when presented with adversarial examples. This conclusion is made based on the results obtained for the three datasets used; specifically, the accuracies achieved for three different adversarial attacks with

varying ϵ values, as well as the unperturbed test samples.

6.3 Limitations

It is vital to reiterate that this study was focused on identifying the advantageous methodology between DL and DML in constructing adversarially robust models and not constructing models with a superior base classification accuracy. Having said this, by spending more time on constructing networks and finding the optimal topologies, the base accuracies can be improved in further research.

To the best of my knowledge, this is the first study which directly compares DL and DML methodologies in creating adversarially robust models. With this being said, it was problematic in finding a base to compare to the findings to, as this did not exist. Many studies examined DL and DML in isolation and improved accuracies in several examples [71, 59, 80] with regards to adversarial inputs but the comparison was never made. Whilst a thorough comparison between DL and DML has been shown, it would have been beneficial to examine this using more data, where a larger number of classes and inputs exist to assess model performance. By doing so, the capabilities of each model are tested further and sub-par performance can be exposed. This is also suggested in the following section, how to improve the current study.

6.4 Furthering this study

Whilst a multitude of models were created and analysed in the present study, there are many factors and variables that can be continuously altered in the search for creating robust models using DL and DML methodologies. One vital component within DML is the choice of loss function during training. A large number of said losses exist, and therefore it would be advantageous to analyse how employing each of these losses in training may have an effect on the model's robustness. This may also help to answer the question of the most optimal loss function to use in DML to train a model to be robust to adversarial inputs.

As mentioned throughout this study, the creation of DL and DML models involves a significant number of hyperparameters, as well as building an appropriate network structure.

Many studies in previously published literature have examined improving the adversarial robustness of a model by including adversarial examples within the training set [71]. As this was an advantageous method for improving accuracy, the methodology employed in the current study can be furthered by including batches of perturbed samples in the training process.

Arguably one of the most significant parameters in training a DL or DML model is the choice of learning rate as this has an effect on weight updates thus influencing the resultant accuracies. An experiment designed to compute the optimal learning rate would be effective and constructive, and would aid in further studies in network designs. Other similar parameters of this nature would be entities such as weight decay and momentum, therefore experimenting further with these variables would expose more about their influences on model performances in an adversarial setting.

Whilst the present study examined three popular datasets within research, there exists a multitude of other datasets which can be used to test model performance. Further to this, this study only looks at image data whereas CNNs can be applied in a number of settings such as text analysis and audio identification. To further compare DL and DML, it would be beneficial to use datasets of a different type to images to see if the conclusion of this study would be similar for other types of data.

Furthermore, in the context of DML, evaluating models with a variety of further metrics such as Precision@K, Recall@K as well as mAP@R, Normalised Mutual Information and Adjusted Mutual Information would facilitate a greater understanding of model performance. The use of more than one metric in comparing models can prove that some models work better in certain contexts; for example, changing the k value when using Precision@K may expose intriguing properties relating to a model's capability of retrieving similar instances in a dataset. For each class in the dataset, it would be beneficial to look at its performance on adversarial inputs in isolation. This could better the understanding of which classes in a particular dataset are more susceptible to adversarial attacks, and may focus the efforts of the researcher in a more optimal manner.

Citations

[1] U Rajendra Acharya et al. “A deep convolutional neural network model to classify heartbeats”. In: *Computers in biology and medicine* 89 (2017), pp. 389–396.

[2] Naveed Akhtar and Ajmal Mian. “Threat of adversarial attacks on deep learning in computer vision: A survey”. In: *Ieee Access* 6 (2018), pp. 14410–14430.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a convolutional neural network”. In: *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017* 2018-Janua.April 2018 (2018), pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[4] Md Zahangir Alom et al. “Inception recurrent convolutional neural network for object recognition”. In: *arXiv preprint arXiv:1704.07709* (2017).

[5] Shun-ichi Amari. “Backpropagation and stochastic gradient descent method”. In: *Neurocomputing* 5.4-5 (1993), pp. 185–196.

[6] Nii O Attoh-Okine. “Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance”. In: *Advances in engineering software* 30.4 (1999), pp. 291–302.

[7] Bowen Baker et al. “Designing neural network architectures using reinforcement learning”. In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2017), pp. 1–18. arXiv: 1611.02167.

[8] Aurélien Bellet, Amaury Habrard, and Marc Sebban. “Metric learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 9.1 (2015), pp. 1–151.

[9] Luca Bertinetto et al. “Fully-convolutional siamese networks for object tracking”. In: *European conference on computer vision*. Springer. 2016, pp. 850–865.

[10] Léon Bottou. “Stochastic gradient descent tricks”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[11] Léon Bottou et al. “Stochastic gradient learning in neural networks”. In: *Proceedings of Neuro-Nımes* 91.8 (1991), p. 12.

[12] Rui Cao et al. “Enhancing remote sensing image retrieval using a triplet deep metric learning network”. In: *International Journal of Remote Sensing* 41.2 (2020), pp. 740–751.

[13] Nicholas Carlini and David Wagner. “Towards evaluating the robustness of neural networks”. In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[14] Nicholas Carlini et al. “On Evaluating Adversarial Robustness”. In: *arXiv* (2019), pp. 1–24. ISSN: 23318422. arXiv: 1902.06705.

[15] Ruey-Maw Chen and Yueh-Min Huang. “Competitive neural network to solve scheduling problems”. In: *Neurocomputing* 37.1-4 (2001), pp. 177–196.

[16] Davide Chicco and Giuseppe Jurman. “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”. In: *BMC genomics* 21.1 (2020), pp. 1–13.

[17] Dan Claudiu Cireşan et al. “Convolutional neural network committees for handwritten character classification”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* 10 (2011), pp. 1135–1139. ISSN: 15205363. DOI: 10.1109/ICDAR.2011.229.

[18] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. “Certified adversarial robustness via randomized smoothing”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1310–1320.

[19] Padraig Cunningham and Sarah Jane Delany. “k-Nearest neighbour classifiers: (with Python examples)”. In: *arXiv preprint arXiv:2004.04523* (2020).

[20] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[21] Fredrik A Dahl. “Convergence of random k-nearest-neighbour imputation”. In: *Computational Statistics & Data Analysis* 51.12 (2007), pp. 5913–5917.

[22] Shengyong Ding et al. “Deep feature learning with relative distance comparison for person re-identification”. In: *Pattern Recognition* 48.10 (2015), pp. 2993–3003. ISSN: 00313203. DOI: 10.1016/j.patcog.2015.04.005. arXiv: 1512.03622.

[23] Mohammad Dorofki et al. “Comparison of artificial neural network transfer functions abilities to simulate extreme runoff data”. In: *International Proceedings of Chemical, Biological and Environmental Engineering* 33 (2012), pp. 39–44.

[24] Samuel G Finlayson et al. “Adversarial attacks against medical deep learning systems”. In: *arXiv preprint arXiv:1804.05296* (2018).

[25] Samuel G Finlayson et al. “Adversarial attacks on medical machine learning”. In: *Science* 363.6433 (2019), pp. 1287–1289.

[26] Weifeng Ge. “Deep metric learning with hierarchical triplet loss”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 269–285.

[27] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[28] Daniel Greenfeld and Uri Shalit. “Robust learning with the hilbert-schmidt independence criterion”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3759–3768.

[29] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2 (2006), pp. 1735–1742. ISSN: 10636919. DOI: 10.1109/CVPR.2006.100.

[30] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[31] Shawn Hershey et al. “CNN architectures for large-scale audio classification”. In: *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2017, pp. 131–135.

[32] Sepp Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[33] Elad Hoffer and Nir Ailon. “Deep metric learning using triplet network”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9370.1271 (2015), pp. 84–92. ISSN: 16113349. DOI: 10.1007/978-3-319-24261-3_7. arXiv: 1412.6622.

[34] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[35] Yan Jiang et al. “Project Gradient Descent Adversarial Attack against Multisource Remote Sensing Image Scene Classification”. In: *Security and Communication Networks* 2021 (2021).

[36] Wonsik Kim et al. “Attention-based ensemble for deep metric learning”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 736–751.

[37] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[38] Kamran Kowsari et al. “Hdltex: Hierarchical deep learning for text classification”. In: *2017 16th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2017, pp. 364–371.

[39] Alex Krizhevsky and G Hinton. “Convolutional deep belief networks on cifar-10”. In: *Unpublished manuscript* (2010), pp. 1–9. URL: `http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:Convolutional+Deep+Belief+Networks+on+CIFAR-10%7B%5C#%7D0`.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[41] Anders Krogh and John A Hertz. “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems*. 1992, pp. 950–957.

[42] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. *Adversarial examples in the physical world*. 2016.

[43] Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. “Lessons in neural network training: Overfitting may be harder than expected”. In: *AAAI/IAAI*. Citeseer. 1997, pp. 540–545.

[44] Steve Lawrence et al. “Face recognition: A convolutional neural-network approach”. In: *IEEE transactions on neural networks* 8.1 (1997), pp. 98–113.

[45] Yann LeCun et al. “Comparison of learning algorithms for handwritten digit recognition”. In: *International conference on artificial neural networks*. Vol. 60. Perth, Australia. 1995, pp. 53–60.

[46] Honglak Lee et al. “Unsupervised feature learning for audio classification using convolutional deep belief networks”. In: *Advances in neural information processing systems* 22 (2009), pp. 1096–1104.

[47] Ki Bum Lee, Sejune Cheon, and Chang Ouk Kim. “A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes”. In: *IEEE Transactions on Semiconductor Manufacturing* 30.2 (2017), pp. 135–142.

[48] Qing Li et al. “Medical image classification with convolutional neural network”. In: *2014 13th international conference on control automation robotics & vision (ICARCV)*. IEEE. 2014, pp. 844–848.

[49] Zheng Lian et al. “Speech Emotion Recognition via Contrastive Loss under Siamese Networks”. In: *arXiv* (2019). ISSN: 23318422.

[50] Ming Liang and Xiaolin Hu. “Recurrent convolutional neural network for object recognition”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 07-12-June.Figure 1 (2015), pp. 3367–3375. ISSN: 10636919. DOI: 10.1109/CVPR.2015.7298958.

[51] Yihua Liao and V Rao Vemuri. “Use of k-nearest neighbor classifier for intrusion detection”. In: *Computers & security* 21.5 (2002), pp. 439–448.

[52] Jingzhou Liu et al. “Deep learning for extreme multi-label text classification”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2017, pp. 115–124.

[53] S-CB Lo et al. “Artificial convolution neural network techniques and applications for lung nodule detection”. In: *IEEE transactions on medical imaging* 14.4 (1995), pp. 711–718.

[54] Aleksander Lodwich, Yves Rangoni, and Thomas Breuel. “Evaluation of robustness and performance of early stopping rules with multi layer perceptrons”. In: *2009 international joint conference on Neural Networks*. IEEE. 2009, pp. 1877–1884.

[55] Jiwen Lu, Junlin Hu, and Jie Zhou. “Deep metric learning for visual understanding: An overview of recent advances”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 76–84. ISSN: 10535888. DOI: 10.1109/MSP.2017.2732900.

[56] Xingjun Ma et al. “Understanding adversarial attacks on deep learning based medical image analysis systems”. In: *Pattern Recognition* 110 (2021), p. 107332.

[57] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[58] Mahmut Kaya and Hasan Sakir Bilge. “Deep Metric Learning : A Survey”. In: *Symmetry* 11.9 (2019), p. 1066.

[59] Chengzhi Mao et al. “Metric learning for adversarial robustness”. In: *arXiv* NeurIPS (2019). ISSN: 23318422. arXiv: 1909.00900.

[60] Manuel Martinez and Rainer Stiefelhagen. “Taming the cross entropy loss”. In: *German Conference on Pattern Recognition*. Springer. 2018, pp. 628–637.

[61] Goeffrey J McLachlan. “Mahalanobis distance”. In: *Resonance* 4.6 (1999), pp. 20–26.

[62] Risto Miikkulainen et al. “Evolving deep neural networks”. In: *Artificial Intelligence in the Age of Neural Networks and Brain Computing* (2018), pp. 293–312. DOI: 10.1016/B978-0-12-815480-9.00015-3. arXiv: 1703.00548.

[63] Md Ashraful Alam Milton. “Evaluation of momentum diverse input iterative fast gradient sign method (M-DI2-FGSM) based attack method on MCS 2018 adversarial attacks on black box face recognition system”. In: *arXiv preprint arXiv:1806.08970* (2018).

[64] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. “Systematic evaluation of CNN advances on the ImageNet”. In: (2016). DOI: 10.1016/j.cviu.2017.05.007. arXiv: 1606.02228. URL: http://arxiv.org/abs/1606.02228%7B%5C%%7D0Ahttp://dx.doi.org/10.1016/j.cviu.2017.05.007.

[65] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[66] Seyed-Mohsen Moosavi-Dezfooli, Ashish Shrivastava, and Oncel Tuzel. “Divide, denoise, and defend against adversarial attacks”. In: *arXiv preprint arXiv:1802.06806* (2018).

[67] Sridhar Narayan. “The generalized sigmoid activation function: Competitive supervised learning”. In: *Information sciences* 99.1-2 (1997), pp. 69–82.

[68] Anh-Duc Nguyen et al. “Distribution padding in convolutional neural networks”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 4275–4279.

[69] Hieu V Nguyen and Li Bai. “Cosine similarity metric learning for face verification”. In: *Asian conference on computer vision*. Springer. 2010, pp. 709–720.

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

[71] Thomas Kobber Panum et al. “Exploring Adversarial Robustness of Deep Metric Learning”. In: Dml (2021). arXiv: 2102.07265. URL: http://arxiv.org/abs/2102.07265.

[72] Mercedes E Paoletti et al. “A new deep convolutional neural network for fast hyperspectral image classification”. In: *ISPRS journal of photogrammetry and remote sensing* 145 (2018), pp. 120–147.

[73] Nicolas Papernot et al. “The limitations of deep learning in adversarial settings”. In: *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[74] MD Plumbley. “Information processing in negative feedback neural networks”. In: *Network: Computation in Neural Systems* 7.2 (1996), p. 301.

[75] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. “Aspect extraction for opinion mining with a deep convolutional neural network”. In: *Knowledge-Based Systems* 108 (2016), pp. 42–49.

[76] Lutz Prechelt. “Early stopping-but when?” In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[77] Qi Qian et al. “Softtriple loss: Deep metric learning without triplet sampling”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6450–6458.

[78] Mohammad Rastegari et al. “XNOR-net: Imagenet classification using binary convolutional neural networks”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9908 LNCS (2016), pp. 525–542. ISSN: 16113349. DOI: 10.1007/978-3-319-46493-0_32. arXiv: 1603.05279.

[79] Jonas Rauber, Wieland Brendel, and Matthias Bethge. “Foolbox: A Python toolbox to benchmark the robustness of machine learning models”. In: *arXiv* (2017). ISSN: 23318422. arXiv: 1707.04131.

[80] Kui Ren et al. “Adversarial Attacks and Defenses in Deep Learning”. In: *Engineering* 6.3 (2020), pp. 346–360. ISSN: 20958099. DOI: 10.1016/j.eng.2019.12.012. URL: https://doi.org/10.1016/j.eng.2019.12.012.

[81] Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. “Classifying relations by ranking with convolutional neural networks”. In: *arXiv preprint arXiv:1504.06580* (2015).

[82] Warren S Sarle et al. “Stopped training and other remedies for overfitting”. In: *Computing science and statistics* (1996), pp. 352–360.

[83] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[84] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 07-12-June (2015), pp. 815–823. ISSN: 10636919. DOI: 10.1109/CVPR.2015.7298682. arXiv: 1503.03832.

[85] Yi Shang and Benjamin W Wah. “Global optimization for neural network training”. In: *Computer* 29.3 (1996), pp. 45–54.

[86] Neha Sharma, Vibhor Jain, and Anju Mishra. “An analysis of convolutional neural networks for image classification”. In: *Procedia computer science* 132 (2018), pp. 377–384.

[87] P Sibi, S Allwyn Jones, and P Siddarth. “Analysis of different activation functions using back propagation neural networks”. In: *Journal of theoretical and applied information technology* 47.3 (2013), pp. 1264–1268.

[88] Patrice Y Simard, Dave Steinkraus, and John C Platt. “0176_689_Patrice_P.Pdf”. In: *Microsoft Research* Icdar (2003), pp. 1–6.

[89] Evgeny A Smirnov, Denis M Timoshenko, and Serge N Andrianov. “Comparison of regularization methods for imagenet classification with deep convolutional neural networks”. In: *Aasri Procedia* 6 (2014), pp. 89–94.

[90] Leslie N Smith and Nicholay Topin. “Deep convolutional neural network design patterns”. In: *arXiv preprint arXiv:1611.00847* (2016).

[91] Jayshril S Sonawane and DR Patil. “Prediction of heart disease using multilayer perceptron neural network”. In: *International conference on information communication and embedded systems (ICICES2014)*. IEEE. 2014, pp. 1–6.

[92] Sho Sonoda and Noboru Murata. “Neural network with unbounded activation functions is universal approximator”. In: *Applied and Computational Harmonic Analysis* 43.2 (2017), pp. 233–268.

[93] Nitish Srivastava. “Improving neural networks with dropout”. In: *University of Toronto* 182.566 (2013), p. 7.

[94] Manli Sun et al. “Learning pooling for convolutional neural network”. In: *Neurocomputing* 224 (2017), pp. 96–104.

[95] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings* (2014), pp. 1–10. arXiv: 1312.6199.

[96] Siham Tabik et al. “A snapshot of image Pre-Processing for convolutional neural networks: Case study of MNIST”. In: *International Journal of Computational Intelligence Systems* 10.1 (2017), pp. 555–568. ISSN: 18756883. DOI: 10.2991/ijcis.2017.10.1.38.

[97] S Tamari, JHM Wösten, and JC Ruiz-Suárez. “Testing an artificial neural network for predicting soil hydraulic conductivity”. In: *Soil Science Society of America Journal* 60.6 (1996), pp. 1732–1741.

[98] Luis Torgo and Rita Ribeiro. “Precision and recall for regression”. In: *International Conference on Discovery Science*. Springer. 2009, pp. 332–346.

[99] Boukaye Boubacar Traore, Bernard Kamsu-Foguem, and Fana Tangara. “Deep convolution neural network for image recognition”. In: *Ecological Informatics* 48 (2018), pp. 257–268.

[100] Jian Wang et al. “Deep metric learning with angular loss”. In: *arXiv* (2017). ISSN: 23318422.

[101] Liwei Wang, Yan Zhang, and Jufu Feng. “On the Euclidean distance of images”. In: *IEEE transactions on pattern analysis and machine intelligence* 27.8 (2005), pp. 1334–1339.

[102] Xinshao Wang et al. “Ranked list loss for deep metric learning”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June (2019), pp. 5202–5211. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00535. arXiv: 1903.03238.

[103] Xun Wang et al. “Multi-similarity loss with general pair weighting for deep metric learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5022–5030.

[104] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

[105] Bing Xu et al. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[106] Han Xu et al. "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review". In: *International Journal of Automation and Computing* 17.2 (2020), pp. 151–178. ISSN: 17518520. DOI: 10.1007/s11633-019-1211-x. arXiv: 1909.08072.

[107] Jin Xu. "Generate Adversarial Examples by Nesterov-momentum Iterative Fast Gradient Sign Method". In: *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE. 2020, pp. 244–249.

[108] Kewen Yan et al. "Face recognition based on convolution neural network". In: *2017 36th Chinese Control Conference (CCC)*. IEEE. 2017, pp. 4077–4081.

[109] Xun Yang, Peicheng Zhou, and Meng Wang. "Person reidentification via structural deep metric learning". In: *IEEE transactions on neural networks and learning systems* 30.10 (2018), pp. 2987–2998.

[110] Dong Yi et al. "Deep metric learning for person re-identification". In: *Proceedings - International Conference on Pattern Recognition* 11.4 (2014), pp. 34–39. ISSN: 10514651. DOI: 10.1109/ICPR.2014.16. arXiv: arXiv:1407.4979v1.

[111] Dingjun Yu et al. "Mixed pooling for convolutional neural networks". In: *International conference on rough sets and knowledge technology*. Springer. 2014, pp. 364–375.

[112] Tongtong Yuan et al. "Signal-to-noise ratio: A robust distance metric for deep metric learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4815–4824.

[113] Babak Zamanlooy and Mitra Mirhassani. "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.1 (2013), pp. 39–48.

[114] Luiz Zaniolo and Oge Marques. "On the use of variable stride in convolutional neural networks". In: *Multimedia Tools and Applications* 79.19 (2020), pp. 13581–13598.

[115] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization". In: *arXiv preprint arXiv:1409.2329* (2014).

[116] Andrew Zhai and Hao-Yu Wu. "Classification is a strong baseline for deep metric learning". In: *arXiv preprint arXiv:1811.12649* (2018).

[117] Dingyi Zhang, Yingming Li, and Zhongfei Zhang. "Deep metric learning with spherical embedding". In: *arXiv* 2.1 (2020), pp. 1–20. ISSN: 23318422. arXiv: 2011.02785.

[118] Aoxiao Zhong et al. "Deep metric learning-based image retrieval system for chest radiograph and its clinical applications in COVID-19". In: *Medical Image Analysis* 70 (2021), p. 101993.