# DunDi: Improving Robustness of Neural Networks using Distance Metric Learning

Lei Cui[1], Rongrong Xi[1], Zhiyu Hao[1], Xuehao Yu[2], and Lei Zhang[1]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]State Grid Information & Telecommunication Branch, Beijing, China
{cuilei, haozhiyu, xirongrong, zhanglei1}@iie.ac.cn

**Abstract.** The deep neural networks (DNNs), although highly accurate, are vulnerable to adversarial attacks. A slight perturbation applied to a sample may lead to misprediction of the DNN, even it is imperceptible to humans. This defect makes the DNN lack of robustness to malicious perturbations, and thus limits their usage in many safety-critical systems. To this end, we present DunDi, a metric learning based classification model, to provide the ability to defend adversarial attacks. The key idea behind DunDi is a metric learning model which is able to pull samples of the same label together meanwhile pushing samples of different labels away. Consequently, the distance between samples and model's boundary can be enlarged accordingly, so that significant perturbations are required to fool the model. Then, based on the distance comparison, we propose a two-step classification algorithm that performs efficiently for multi-class classification. DunDi can not only build and train a new customized model but also support the incorporation of the available pre-trained neural network models to take full advantage of their capabilities. The results show that DunDi is able to defend 94.39% and 88.91% of adversarial samples generated by four state-of-the-art adversarial attacks on the MNIST dataset and CIFAR-10 dataset, without hurting classification accuracy.

**Keywords:** Robustness · deep neural network · metric learning

## 1 Introduction

Over the past few years, the machine learning systems, especially the one cooperated with the deep neural network (DNN), have reached promising achievement in many fields including image recognition, speech translation, and policy decision [11]. Due to the advance, they are being widely deployed in many real-world applications such as self-driving, face authentication and medical diagnosis. Despite the success, however, recent studies have shown that the deep neural network models, although highly accurate, are vulnerable to adversarial attacks [4, 13, 16]. That is, the model can be fooled to mispredict a modified sample which is intentionally perturbed by the attacker, so-called adversarial sample. These adversarial samples can cause unexpected behaviour of DNNs,

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

and thus are problematic when DNNs are used in safety-critical systems. For example, an adversarial traffic sign may fool the self-driving system to determine a wrong policy and even worse cause car crash [14], and an adversarial human face will allow an attacker to pass the face authorization [21]. Therefore, to provide the ability to defend adversarial samples (or robustness in other literature), is one key factor in deploying DNNs in many safety-critical systems.

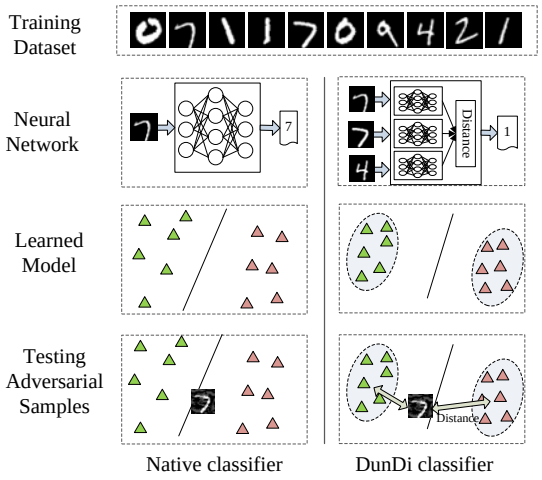



**Fig. 1.** Comparison of traditional model and DunDi model. DunDi employs the Triplet Network to train a model that puts similar samples together and enlarges the distance between different samples, which in turn enlarges the distance between samples and model's boundary. The testing samples will be classified exploiting the distance metric. As we can see, an adversarial sample can move across the boundary to lead to misprediction of the Native classifier, but it will be correctly classified by DunDi since it is closer to the samples of the proper label.

Several approaches have been proposed for defending adversarial attacks in recent years, such as adversarial training [3,22,23], defensive distillation [17] and deep contractive networks [7]. In addition to defense, some approaches tend to detect the adversarial samples and exclude them upon testing [12]. Unfortunately, some works show that these models are still vulnerable to adversarial samples [5]. Recent studies have shown that the DNNs are not robust mainly because the natural samples lie very close to the model's decision boundary [4,13]. Or in other words, the distance between the sample and boundary is short, so that the sample can easily move across the boundary by patching with slight perturbations. Inspired by this, we propose DunDi[1] to defend against adversarial samples by enlarging the distance between samples and boundary, so that a much larger

[1] DunDi is a kind of magic in ancient Eastern legends, and it can transform far to near or near to far, thus avoiding enemy attacks.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

magnitude of perturbations are required to move samples across the boundary, as shown in Figure 1. To achieve this, the key idea behind DunDi is a metric learning model which has been used in many fields [20]. Specifically, we employ a recently proposed metric learning model named as Triplet Network [8]. The Triplet Network takes as input three samples $\{x_a, x_p, x_n\}$ each time in which $x_a$ and $x_p$ are similar while $x_a$ and $x_n$ are different. Then, it learns to pull together the samples of the same label (i.e., reducing the distance between $x_a$ and $x_p$) meanwhile pushing away the samples of different labels (i.e., enlarging the distance between $x_a$ and $x_n$). Consequently, it enlarges the distance between samples and model's boundary, which in turn amplifies the required perturbations for crafting adversarial samples. Exploiting the trained metric model that provides distance computation between samples, we then design a two-step algorithm for efficient multi-class classification. DunDi can build a new model with customized DNN topology. In addition, it supports the incorporation of the pre-trained DNNs, to take full use of their advantages such as high accuracy.

We implement DunDi on Keras/Tensorflow platform. The experimental results show that DunDi is able to defend a significant fraction of adversarial samples. Specifically, it correctly predicts 94.39% and 88.91% of adversarial samples [5] respectively on the MNIST and CIFAR-10 dataset. To conclude, this paper makes the following contributions:

- First, we propose a metric learning based method to enlarge the distance between samples and model's boundary, which in turn amplifies the magnitude of malicious perturbations required to craft adversarial samples, for improving the robustness of DNN to adversarial attacks.
- Second, we design a two-step classification algorithm for classifying samples in a multi-class scenario efficiently based on the distance metric.
- Third, we implement DunDi and evaluate it on two datasets to show its effectiveness when suffering different types of adversarial attacks.

## 2 Background and Related Work

### 2.1 Adversarial Attacks

Adversarial attacks intentionally design a new sample by adding slight perturbations to a sample that is correctly classified by the model, so that the model misclassifies the new sample, so-called adversarial sample. Many recent works tend to carefully perturb samples for improving the attack success rate, including fast gradient sign method (FGSM), iterative FGSM (iFGSM), Jacobian-based saliency map algorithm (JSMA), projection method (DeepFool), etc. We will give a brief introduction to these methods below.

**Notations**: Let $x$ be a sample, $f$ be the trained classifier, $y$ be the predicted label of input $x$ (i.e., $y = f(x)$), and $y*$ be the true label of $x$. During the training stage, the model $f$ will be continuously updated for minimizing the value of loss function $L(y, y*)$ given a set of $x$ and $y*$.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

**FGSM**: FGSM [6] computes the gradient of the loss function $L$ with respect to a sample $x$, generates perturbations formed by the sign of the gradient multiplied by a parameter $\epsilon$, and finally crafts an adversarial sample $x'$ by adding perturbations to $x$, as shown in Equation 1. $\epsilon$ refers to the specified magnitude of perturbations. If it is small enough, the crafted adversarial sample and the original sample will be indistinguishable from the human view.

$$x' = x + \epsilon sign(\nabla_x L(y, y*)) \tag{1}$$

A later work, named as iterative FGSM or iFGSM, extends FGSM by applying FGSM multiple times with a finer perturbation. In each step, iFGSM clips pixel values of intermediate results of the previous step. The advantage over FGSM is that it achieves smaller distortion and higher attack success rate [9].

**JSMA**: JSMA is a targeted attack approach, i.e., it specifies a target label $y_t$ and tends to perturb $x$ so that the model misclassifies the crafted sample $x'$ to $y_t$ [15]. To achieve this, JSMA iteratively searches for pixels or pairs of pixels in $x$ to change such that the probability of the target label is increased and the probability of all other labels are decreased.

**DeepFool**: DeepFool is an untargeted attack algorithm based on the theory of projection to the closest separating hyperplane in classification [14], so that it can compute minimal perturbations to sufficiently fool the classifier.

### 2.2 Defenses

**Adversarial training**. This method augments the training dataset with a number of correctly labelled adversarial samples and then retrains the neural network model. With weights updating for minimizing the value of loss function, especially that of adversarial samples, the retrained neural network model will be more robust to adversarial samples [22, 23].

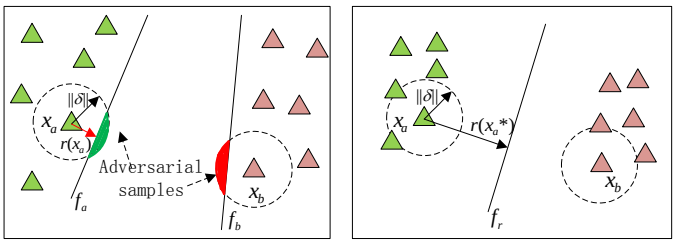
**Defensive distillation**. This strategy firstly trains a model using hard class labels. Then, exploiting the initial model's softmax probability outputs, it trains a second model. This method can smooth the model's surface in the directions would be exploited by attackers, making it difficult to discover gradients for crafting adversarial samples [17].

**Gradient masking**. Observed that most adversarial attack techniques generate adversarial samples using the gradients of the neural network, gradient masking is proposed to eliminate the gradients, so that the attackers fail to find appropriate directions that can be used for perturbing the samples [19, 24]. However, a recent study has shown that this method fails to work in many scenarios because the attacker can still discover gradients in a high dimension space

## 3 Design and Implementation of DunDi

In this section, we first formulate the robustness problem by the distance between samples and model's boundary and then present an overview of the proposed solution. Finally, we introduce two essential parts, i.e., how to enlarge the distance for improving robustness and how to classify samples with the distance metric.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11




(a) Models are not robust. (b) Model is more robust.

**Fig. 2.** Samples, boundaries, and adversarial samples. $f_a$ and $f_b$ are two accurate but not robust classifiers, while $f_r$ is more robust because the distance between samples and boundary is larger. The green and red regions denote the distribution of adversarial samples with respect to $x_a$ and $x_b$ respectively.

### 3.1 Enlarging Distance for Robustness

We first introduce some notations related to the robustness of neural networks [4], and then explain why increasing the distance would improve robustness.

**Perturbation**. Let $x$ be a sample, $f$ be the trained classification model. $\delta$ refers to the perturbations applied to $x$ for generating a new sample $x'$, i.e., $x' = x+\delta$. If the predicted label of $x'$ is different from that of $x$, i.e., $f(x) \neq f(x')$, then an adversarial sample $x'$ with respect to $\delta$ is successfully crafted.

**Magnitude of perturbation**. The magnitude of perturbations $\delta$ is denoted by $\| \delta \|_S$. It can be computed in many norms such as $\ell_1$, $\ell_2$ and $\ell_\infty$ in $S$ which denotes the space of admissible perturbations.

**Robustness of one sample**. The robustness of one sample $x$, denoted by $r(x)$, can be measured by the minimal perturbations changing the label of $x$, i.e., $r(x) = min \| \delta \|_S$ subject to $f(x+\delta) \neq f(x)$. The smaller value of $r(x)$ implies that it requires slight modifications at $x$ for changing its label, thereby enabling the perturbed samples difficult to be detected.

**Robustness over the classifier**. To measure the robustness over the classifier $f$, denoted by $R(f)$, we can simply compute the expectation of $r(x)$ over all the samples, i.e., $R(f) = \mathbb{E}_X(r(x))$.

Figure 2(a) illustrates the notations associated with two classes of samples and several classifiers. For a sample $x_a$, the dashed cycle shows the space of admissive perturbations, i.e., $S$. A large number of new samples can be generated by adding perturbations within the cycle to $x_a$. Given a classifier $f_a$, the robustness of $x_a$, denoted by $r(x_a)$ in Figure 2(a), refers to the shortest distance for moving $x_a$ across the boundary of $f_a$.

Recent DNNs have shown the ability to train accurate classifiers. However, high accuracy does not mean robustness. As shown in Figure 2(a), two classifiers $f_a$ and $f_b$ are accurate because they can correctly classify the samples. Unfortunately, they are not robust. Any sample $x'_a$ in the green region, which perturbs $x_a$ by adding perturbations with a magnitude greater than $r(x_a)$, can lead to misclassification of $f_a$. The same is true for sample $x_b$ and classifier $f_b$. The rea-

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

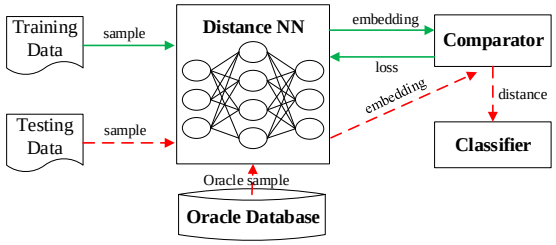



**Fig. 3.** Overview. The green arrow shows the training stage, while the red dashed arrow shows the testing stage.

son is that the sample lies too close to the classifier's boundary, so that it can be easily perturbed for moving across the boundary. Inspired by this, an accurate and robust classifier should keep the samples far away from the boundary for defending adversarial attacks. Figure 2(b) presents a more robust classifier $f_r$, i.e., a larger magnitude of perturbations (i.e., $r(x_a*)$) is required to fool the classifier due to a larger distance between $x_a$ and $f_r$. Such considerable perturbations would surpass the allowed perturbing space $S$, making the crafted sample be easily detected. This gives us a hint that enlarging the distance between samples and boundary would help to improve the robustness of the model.

### 3.2 Overview of DunDi

As stated above, the distance is one of the keys to improve the robustness of neural network models [4, 13]. To this end, we propose to enlarge the distance between samples and model's boundary with metric learning. Figure 3 presents a brief overview of DunDi on describing the process of training stage and testing stage. The critical component is a Distance Neural Network (Distance NN). It is responsible for producing an embedding, which is a numeric vector representing learned features for each input sample. The embeddings should satisfy two requirements. First, for samples of the same label, the distance of their embeddings should be small. Second, for samples of different labels, the distance of associated embeddings should be large. To achieve this, the Distance NN during the training stage takes as input samples and produces embeddings. Then, the Comparator computes the distance between embeddings, computes the loss between the expected distance (i.e., specified by the user) and the predicted distance (i.e., computed between generated embeddings), and then leverages the loss to update the weights of Distance NN with feedback propagation. Finally, with the aim to minimize the loss, the Distance NN will learn to generate embeddings following the two requirements.

For testing, DunDi maintains an Oracle Database storing a certain number of samples for each class in the dataset, which are considered as oracle samples for classification. Given a testing sample, the Classifier leverages Distance NN to produce the embedding, employs Comparator to compute the distance between

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

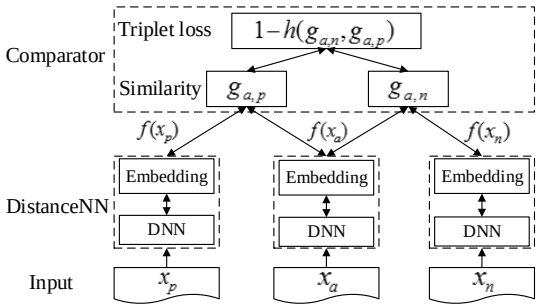



**Fig. 4.** Design of Distance NN and Comparator based on Triplet Network. $f$ denotes the embedding for each sample, $g$ denotes the similarity between two embeddings, and $1 - h$ denotes the triplet loss.

the testing sample and oracle samples, and finally employs the proposed multi-class classification algorithm to predict the label.

### 3.3 Enlarging Distance using Metric Learning

In DunDi, we enlarge the distance with metric learning (also known as similarity learning), which tends to learn from samples a similarity function that measures how similar two objects are. Specifically, we employ the idea of recently proposed Triplet Network to train the model [8]. The Triplet Network takes as input a tuple, which contains three samples along with a ranking of similarities between them. It can learn to compute similarities that are distributed in a wide range, i.e., [0,1], enabling the model to outperform other metric learning methods such as Siamese Network that only operates either 0 (same samples) or 1 (different samples) [2]. As shown in Figure 4, the input of Distance NN is a tuple denoted by $\{x_a, x_p, x_n\}$, in which $x_a$ (anchor sample) is more similar to $x_p$ (positive sample) than it is to $x_n$ (negative sample). DunDi aims to put the positive pair closer (i.e., $x_a$ and $x_p$) meanwhile pushing the negative one further (i.e., $x_a$ and $x_n$), for following the ranking of similarities provided in the input tuple. To achieve this, it computes the triplet loss of three input samples, which acts as the loss to be minimized during the training stage.

As shown in Figure 4, given the tuple $\{x_a, x_p, x_n\}$, the Distance NN produces an embedding for each sample, denoted by $f(x)$. With these embeddings, the Comparator calculates the similarity of each pair, denoted by $g(f(x_a), f(x_p))$ (or $g_{a,p}$ for short) for the positive pair and $g_{a,n}$ for the negative pair. Then, it computes the predicted discrepancy between $g_{a,p}$ and $g_{a,n}$, denoted by $h(g_{a,p}, g_{a,n})$, and finally computes the triplet loss by the difference between the expected discrepancy (specified by the user) and the predicted discrepancy. The following parts will describe how to generate the embedding, compute the similarity and compute the triplet loss in detail.

**Distance NN**. It is responsible for producing an embedding for an input sample. The embedding is a vector in a $n$-dimensional Euclidean space, repre-

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

senting the learned features of the sample. Distance NN contains two modules, the Embedding module and the DNN module, as illustrated in Figure 4. The Embedding module is composed of several fully connected layers, and it takes the weights of the last layer as the generated embedding. The DNN module can be the convolutional neural network or other DNNs. It is worth noting that the DNN module can incorporate the pre-trained DNN models to take full advantage of their capabilities. For a pre-trained DNN model, we simply remove its softmax layer and then connect it to the Embedding module to produce the embedding.

**Similarity computation**. There exist many methods to compute the similarity (or, in this case, the distance) between two embeddings, e.g., Euclidean, Manhattan and Cosine distance. In DunDi, we leverage the Euclidean distance to measure the similarity, i.e., $g_{a,p} = \sqrt[2]{(a_1 - p_1)^2 + (a_2 - p_2)^2 + ... + (a_n - p_n)^2}$.

**Loss function**. As stated before, the tuple $\{x_a,\ x_p,\ x_n\}$ implies that $x_a$ and $x_p$ are more similar than $x_a$ and $x_n$. For simplicity, we assign $x_a$ and $x_p$ with samples of the same label while $x_n$ a different label, so that the expected discrepancy, denoted by $h_e(g_{a,n}, g_{a,p})$, is 1. Let $h_p(g_{a,n}, g_{a,p})$ denote the predicted discrepancy, then the triplet loss value is computed by $1 - h_p(g_{a,n}, g_{a,p})$. The Comparator will return the loss back to the Distance NN for optimizing the weights of the neural network model. By iteratively optimization, the triplet loss will continue to decrease until convergence. We leverage the *sigmoid* function to compute the predicted discrepancy, i.e., $h_p = sigmoid(g_{a,n} - g_{a,p})$, so that its value as well the triplet loss can be mapped to a range of 0 to 1.

### 3.4 Multi-class Classification Algorithm

As for testing, it only requires a pair of samples each time, one of which is the testing sample and the other from the labelled dataset. We employ the Distance NN to produce embeddings and then computes the similarity between the two. Exploiting the similarities between the testing sample and other samples, we can classify the testing sample using the idea of nearest neighbour clustering, i.e., it will be assigned with the label of the closest class of samples.

Unfortunately, this approach will introduce significant time overhead, mainly because there exist multiple classes each of which has thousands of samples in the dataset for similarity computation. To mitigate this problem, instead of using the whole dataset, we use a small number ($n$) of labelled samples of each class as oracle samples. These samples are randomly selected to cover the diversity of samples. We then propose a two-step multi-class classification algorithm for reducing the computation cost, as shown in Algorithm 1. The first step attempts to exclude the most unlikely labels. It compares the testing sample with $m$ ($m < n$) oracle samples of each class (# 2-6) and selects the top $k$ labels showing the highest similarities (#7). Then, the second step compares the testing sample with $n$ samples of each of the selected $k$ labels (#9-13) and finally assigns it with the label with the highest similarity (#14). We empirically set the value of $n$, $m$, and $k$ to 20, 3, and 5 respectively, which achieves well trade-off between the accuracy and testing time.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

**Input:** *text_x*
**Output:** *predicted_label*
*selected_labels* ← {}, *similarities* ← {} ;
**for** *label* ∈ *all_labels* **do**
  *label_sim* ← 0;
  **for** *sample* ∈ *oracle_samples*[*label*][0 : *m*] **do**
    *sim* ← *sim*(*test_x*, *sample*), *label_sim* ← *label_sim* + *sim*;
  **end**
  *similarities* ← *similarities* ∪ *label_sim*;
**end**
*selected_labels* ← *argtop_k*(*similarities*);
*predicted_label* ← 0, *similarities* ← {} ;
**for** *label* ∈ *selected_labels* **do**
  *label_sim* ← 0 ;
  **for** *sample* ∈ *oracle_samples*[*label*][0 : *n*] **do**
    *sim* ← *sim*(*test_x*, *sample*), *label_sim* ← *label_sim* + *sim*;
  **end**
  *similarities* ← *similarities* ∪ *label_sim*;
**end**
*predicted_label* = *arg* max(*similarities*)

**Algorithm 1:** Two-step classification algorithm

### 3.5 Implementation

We implement DunDi using Keras on top of TensorFlow [1]. Within the paper, we focus on image classification, and thus we use the convolutional neural network as the DNN model in the Distance NN. Unlike traditional CNN topologies, which are configured with a softmax layer to help output a single label or a probability distribution over several labels, the CNN of DunDi has no softmax layer but is connected to the Embedding module. The Embedding module is a simple neural network composed of two fully-connected layers, and it produces a 128 length vector which is the final embedded representation of the input sample. As for the pre-trained CNN model, we firstly employ it to produce intermediate vectors and feed them into the Embedding module. Then we only update the weights of the Embedding module for minimizing the triplet loss value during training.

In Comparator, we compute the Euclidean similarity between two embeddings. Then, we compute the predicted discrepancy between the similarities of two pairs, i.e., positive pair and negative pair. Finally, we use the Sigmoid function to normalize the predicted discrepancy and compute the triplet loss by the difference between the expected discrepancy and predicted discrepancy. The triplet loss will be minimized during training, as mentioned before. In Classifier, we compare the testing sample and samples in the Oracle Database, and then employ the two-step classification algorithm for image classification.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11


(a) MNIST. (b) CIFAR-10.

**Fig. 5.** Adversarial samples of different attacks.

## 4 Evaluation

### 4.1 Experimental Setup

**Dataset.** We evaluate DunDi on MNIST and CIFAR-10 dataset. It is worth noting that in DunDi, the input upon training is a tuple of three images, in which the first two are of the same class (positive pair) and the third a different class (negative pair). Consider that there exist thousands of images in the dataset, we randomly select images of each class and pack them into tuples. Finally, we generate 360K tuples for training and use 10K original testing images for testing.

**Models.** As for MNIST, we use three CNN models of different architectures [10,18] as Native models for comparison. DunDi uses the same topology of these models as the DNN module and then retrains the entire Distance NN, named as DunDi_R (Retrained). In addition, it also incorporates the pre-trained models available in DeepXplore [18] and only trains the Embedding module, named as DunDi_P (Pre-trained). Similarly, for CIFAR-10, we employ the VGG-16 model to retrain a DunDi_R model, and we also incorporate a pre-trained VGG-16 model to build DunDi_P. We by default set the number of oracle images to 10.

**Adversarial samples.** We generate adversarial samples with the method proposed in [5], which supports four types of attacks, i.e., FGSM, JSMA, and two variants of iFGSM (iFGSM-a which stops iterating when misclassification is achieved, and iFGSM-b which runs for a certain number of iterations that is well beyond the average misclassification point). For each attack, we generate one adversarial sample with respect to each testing image and eventually get 40,000 adversarial images for each dataset. Figure 5 shows several adversarial samples generated by these attack approaches.

We run the experiments on one Tesla K40 GPU configured with 12GB of RAM. We train the model of DunDi_R with VGG-16 for 200 epochs and train others for 10 epochs. The following sections will report the experimental results.

### 4.2 Accuracy of Models

We first present the accuracy of different models. As can be seen in Table 1, for the MNIST dataset, the accuracy of the three native models is 98.78%, 98.99% and 99.05% respectively, which match the reported accuracy of previous works,

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

e.g., 98.3%, 98.9% and 99.05% respectively in [18]. The results of both retrained and pre-trained DunDi models are comparable with the accuracy of the native models. For example, the accuracy is 99.05%, 99.05% and 98.94% for Model3, DunDi_R3 and DunDi_P3 respectively when the number of oracle images is 20. These results imply that the proposed classification algorithm based on the distance metric is feasible. In addition, with the increasing number of oracle images, the accuracy increases as well. For example, the accuracy is 98.86%, 99.0% and 99.05% respectively when the number is 1, 10 and 20 for DunDi_R3. This is mainly because more oracle images are capable of representing the diversity of features for each class of images, so that the variance is reduced and the accuracy can be improved accordingly. The results on CIFAR-10 show similar results. It should be pointed out that the accuracy of DunDi_P is higher than that of Native Model, e.g., 93.1% against 89.7%. We suspect this is because the distance between images across different classes is much larger than that within the same class, so that the distance comparison can help to hide the variances between images of the same class, particularly for CIFAR-10 where the images exhibit large variances. We leave the further exploration as our future work.

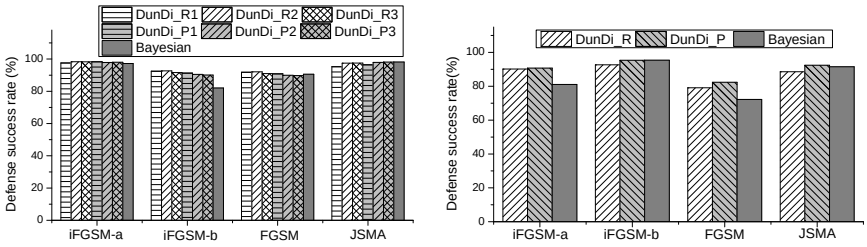
**Table 1.** Accuracy of different models. The Native Models are irrelevant to oracle images, so their results are only reported on column '1'.

| Dataset | Model Name | Number of oracle images | | |
|---|---|---|---|---|
| | | 1 | 10 | 20 |
| MNIST | Native Model1 | 98.78% | N/A | N/A |
| | Native Model2 | 98.99% | N/A | N/A |
| | Native Model3 | 99.05% | N/A | N/A |
| | DunDi_R1 | 97.88% | 98.53% | 98.65% |
| | DunDi_R2 | 98.59% | 98.96% | 98.98% |
| | DunDi_R3 | 98.86% | 99.0% | 99.05% |
| | DunDi_P1 | 98.79% | 98.96% | 99.04% |
| | DunDi_P2 | 98.07% | 98.59% | 98.71% |
| | DunDi_P3 | 98.5% | 98.82% | 98.94% |
| CIFAR-10 | Native Model | 89.7% | N/A | N/A |
| | DunDi_R | 91.16% | 91.37% | 91.53% |
| | DunDi_P | 93.1% | 93.28% | 93.32% |

### 4.3 Robustness of Models

We measure the robustness by the defense success rate, which refers to the number of correctly predicted adversarial samples to the total number of adversarial samples. Figure 6 compares the defense success rate of various DunDi models and a recently proposed method which employs Bayesian uncertainty estimates and density estimates to detect adversarial samples (named as Bayesian) [5]. As we can see, all the DunDi models achieve high defense success rate for the four types

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11




(a) MNIST. (b) CIFAR-10.

**Fig. 6.** Defense success rate of different models.

of adversarial attacks. For MNIST, the average defense success rate of the six DunDi models is 98.07%, 91.45%, 90.91% and 97.13% respectively for iFGSM-a, iFGSM-b, FGSM, and JSMA attack. As for CIFAR-10, the average value of DunDi models is 90.46%, 94.0%, 80.7% and 90.49% respectively. The accuracies are comparable with the results reported in the Bayesian approach. These results prove the effectiveness of the proposed DunDi approach which improves robustness by enlarging the distance between samples and model's boundary.

### 4.4 Varying Parameters

In this section, we show the effects on accuracy and robustness of varying configurations used in DunDi, including the size of training dataset, the number of oracle images [2]. For brevity, here we only report the results on MNIST.

**Varying dataset size.** Table 2 illustrates the accuracy and robustness on the training dataset of different sizes (or, in this case, the number of tuples). It can be seen that the accuracy increases with the increasing size of the dataset, e.g., it increases from 96.1% to 98.8% when the size increases from 18,000 to 360,000. The defense success rate, which denotes the average value of DunDi_R1 under four adversarial attacks, also shows an overall increasing trend. Thus, the model can achieve higher accuracy and robustness when training on more tuples.

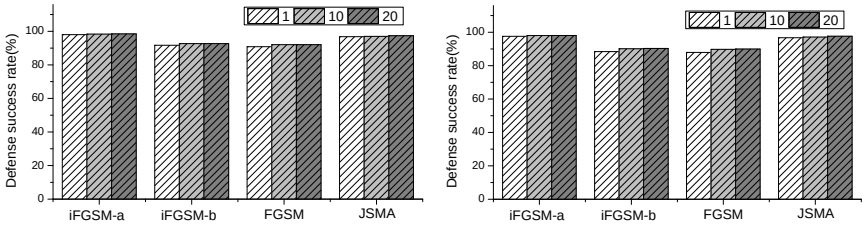
**Table 2.** Accuracy and defense success rate of DunDi_R1 with varying dataset size.

| Dataset Size | 18,000 | 36,000 | 180,000 | 360,000 |
|---|---|---|---|---|
| Accuracy | 96.1% | 96.6% | 98.0% | 98.8% |
| Defense success rate | 91.90% | 92.02% | 92.35% | 93.06% |

**Varying number of oracle images.** Figure 7 compares the robustness with varying number of oracle images. As expected, the defense success rate increases

[2] Here we only report the results of varying number of oracle images (i.e., $n$) because it plays a more significant role than $m$ and $k$ in the multi-class classification algorithm.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11




(a) Robustness of DunDi_R1. (b) Robustness of DunDi_P1.

**Fig. 7.** Robustness with varying number of oracle images.

with the increasing number of oracle images, mainly because more oracle image can capture the diversity of features. It is worth noting that the defense success rate is high even if there is only 1 oracle image for each label, e.g., the value of DunDi_P1 is 97.55%, 88.42%, 87.94% and 97.1% respectively for the four adversarial attacks, as shown in Figure 7(b). This implies that the model can learn representative features to characterize the distance between images.

## 5 Conclusions

The deep neural networks are vulnerable to adversarial attacks, which limits the usage of DNN in safety-critical systems. This paper presents DunDi to improve the robustness of neural networks. DunDi employs the metric learning approach to put the samples of the same class together and push the samples of different classes away, so that the distance between samples and model's boundary can be enlarged as well. The increase of distance will require a larger magnitude of perturbations for generating adversarial samples to move across the boundary, which yet will be easily detected. The experimental results show that DunDi can defend a significant fraction of adversarial samples without losing classification accuracy. In the future, we plan to extend DunDi with deeper CNN models such as ResNet-50 and evaluate it with real-world images of ImageNet.

## Acknowledgement

Thanks for the valuable comments from anonymous reviewers of ICCS2019 and researchers of the George Washington University. This work has been supported by the National Natural Science Foundation of China (grant no. 61602465 and 61601458), National Key Research and Development Program of China (grant no. 2016QY04W0804), and Beijing Natural Science Foundation (grant no. 4172069). Rongrong Xi is the corresponding author.

## References

1. Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11

2. Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546, 2005.
3. Lei Cui, Zhiyu Hao, Dongbin Wang, and Yongnan Li. Detecting adversarial samples using heatmap of neural networks. *AAAI-19 Workshop on Engineering Dependable and Secure Machine Learning*, 2019.
4. Alhussein Fawzi, Seyed Mohsen Moosavi Dezfooli, and Pascal Frossard. A geometric perspective on the robustness of deep networks. *IEEE Signal Processing Magazine*, 2017.
5. Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.
6. Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.
7. Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *arXiv preprint arXiv:1412.5068*, 2014.
8. Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92, 2015.
9. Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.
10. Yann LeCun. The mnist database of handwritten digits.
11. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
12. Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *ICLR*, 2017.
13. Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, pages 86–94, 2017.
14. Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016.
15. Nicolas Papernot, Patrick McDaniel, and et al. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P)*, pages 372–387, 2016.
16. Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, and et al. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint*, 2016.
17. Nicolas Papernot, Patrick McDaniel, Xi Wu, and et al. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP)*, pages 582–597, 2016.
18. Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *SOSP*, pages 1–18, 2017.
19. Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *arXiv preprint arXiv:1711.09404*, 2017.
20. Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.
21. Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*, 2017.
22. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, and et al. Intriguing properties of neural networks. In *ICLR*, 2014.
23. Florian Tramèr, Alexey Kurakin, Nicolas Papernot, and et al. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.
24. Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, and et al. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_11