A Robust Classification-autoencoder to Defend Outliers and Adversaries*

Lijia Yu and Xiao-Shan Gao
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
University of Chinese Academy of Sciences

July 1, 2021

Abstract

In this paper, we present a robust classification-autoencoder (CAE) which has strong ability to recognize outliers and defend adversaries. The basic idea is to change the autoencoder from an unsupervised learning method into a classifier. The CAE is a modified autoencoder, where the encoder is used to compress samples with different labels into disjoint compression spaces and the decoder is used to recover a sample with a given label from the corresponding compression space. The encoder is used as a classifier and the decoder is used to decide whether the classification given by the encoder is correct by comparing the input sample with the output. Since adversary samples are seeming inevitable for the current DNN framework, we introduce the list classification based on CAE to defend adversaries, which outputs several labels and the corresponding samples recovered by the CAE. The CAE is evaluated using the MNIST dataset in great detail. It is shown that the CAE network can recognize almost all outliers and the list classification contains the correct label for almost all adversaries.

Keywords. Robust DNN, autoencoder, list classification, outlier, adversary sample.

1 Introduction

The deep neural network (DNN) [13] has become the most powerful machine learning method, which has been successfully applied in computer vision, natural language processing, and many other fields. One of the major problems of DNN is its robustness against noises, outliers, and adversaries. There exist vast literature on improving the robustness of DNNs [1, 3, 5, 34]. In this paper, we present a new approach by changing the autoencoder from an un-supervised learning model into a classifier.

1.1 Contribution

Intuitively, a neural network is robust if it has high accuracy at the input with noises. On the other hand, robustness can also be considered in the following more general sense: a neural network is robust if it can not only correctly classify samples containing noises but also has the ability to recognize outliers and to defend adversaries [1, 3, 5, 34].

Outlier detection is a key issue in the open-world classification problem [34, 24], where the inputs to the DNN are not necessarily satisfy the same distribution with the training set. On the contrary, the objects to

*This work is partially supported by NSFC grant No.11688101 and NKRDP grant No.2018YFA0306702.

be classified are usually consist of a low-dimensional subspace of the total input space to the DNN and the majorities of the input are outliers. To be more precise, let us consider a classification DNN $\mathcal{F} : \mathbb{I}^n \to \mathbb{L}$ for certain object $\mathbb{O} \subset \mathbb{I}^n$, where $\mathbb{I} = [0, 1]$ and $\mathbb{L} = \{0, 1, \ldots, o\}$ is the label set. For the MNIST benchmark dataset, $\mathbb{O}$ is the hand-written numbers, $\mathbb{L} = \{0, 1, \ldots, 9\}$, and $n = 784$. In general, $\mathbb{O}$ is considered to be a very low-dimensional subset of $\mathbb{I}^n$ and hence the majorities of elements in $\mathbb{I}^n$ are not in $\mathbb{O}$, which are called *outliers*. For each element $x \in \mathbb{I}^n$, $\mathcal{F}$ will give a label in $\mathbb{L}$ to x, which will be wrong with high probability if $\mathcal{F}$ is not specifically designed and trained to defend outliers.

A more subtle and difficult problem related to the robustness of DNN is the existence of adversaries [26], that is, it is possible to intentionally make little modification to an image in $\mathbb{O}$ such that human can still recognize the object clearly, but the DNN outputs a wrong label or even any given label. Existence of adversary samples makes the DNN vulnerable in safety-critical applications. Although many effective methods for training DNN to defend adversaries were proposed [1, 3], it was shown that adversaries seem still inevitable for current DNNs [2, 22].

In this paper, we present a DNN which has strong ability to recognize outliers and defend adversaries. The autoencoder is a powerful DNN to learn compressed features for data in high dimensional spaces. The basic idea of this paper is to change the autoencoder from an unsupervised learning model into a classification network. The encoder $\mathcal{E}$ of the autoencoder is used to compress the input images into a low-dimensional space $\mathbb{R}^m$ ($m \ll n$) such that the images with the same label are compressed into *compression space* of that label and images with different labels are compressed into disjoint subsets of $\mathbb{R}^m$. Furthermore, the images in $\mathbb{O}$ can be approximately recovered by the decoder $\mathcal{D}$ from their compression spaces. The encoder $\mathcal{E}$ is used both as a feature learner and a classifier, which is different from the usual classifiers in that several coordinates instead of one are used to represent and classify images for each label. The decoder $\mathcal{D}$ can be used to give the final classification by comparing the input image with the output.

The above network is called a classification-autoencoder (CAE). We prove that such a network exists in certain sense. Precisely, we prove that there exists an autoencoder such that the encoder compresses images with different labels into disjoint compression sets of of $\mathbb{R}^m$ for any m and the decoder approximately recovers the input image from its compression set with any given precision.

The CAE is evaluated using the MNIST dataset in great detail. A major advantage of the CAE network is that almost all outliers can be recognized. As an autoencoder, the outputs of a CAE are always like the object $\mathbb{O}$ to be classified. By definition, an outlier is an image which is not considered to be an element of $\mathbb{O}$, so an image is treated as an outlier if the input and the output are different.

The CAE also works well for adversaries in the following sense. For most adversaries of MNIST, the CAE can recognize them as problem images. Considering the fact that adversaries are seemly inevitable for neural networks with moderate complex structures [2, 22], a possible way to alleviate the problem is to give several answers instead of one. In a CAE, we can apply the decoder $\mathcal{D}$ to the compression space of each label to recover images and output the labels and corresponding images which are similar to the input image. This kind classification is called *list classification* and the output is a *classification list*, which tries to give uncertain but lossless classifications. Our experimental results show that for almost all adversaries, the classification list contains the correct label. A lossless classification is important in that, the classification does not miss important information, which can be used for further analysis.

1.2 Related work

The autoencoder is one of the most important neural networks for unsupervised learning [4], which has many improvements and applications. The autoencoder learns compressed features in a low-dimensional space for high-dimensional data with minimum reconstruction loss, while our CAE makes classification

at the same time of learning features. It is natural to use autoencoders for outlier detection due to its reconstruction property [35]. We improve this in two aspects. First, by compressing images with different labels into disjoint compression spaces, the robustness is increased and the classification can be given. Second, we use the compression spaces to introduce the list classification. The robustness of autoencoder was studied [28, 35, 20]. In principle, these methods can be applied to our CAE model to further improve the robustness.

A simple approach to recognize outliers is to introduce a new label representing outliers and add outlier samples [34]. The difficulty with this approach is that we need to add outlier samples and the distribution of outliers is usually too complex to model in high dimensional spaces. As a consequence, a network based on this approach works well for those outliers similar to those in the training set and works poorly for those outliers different from those in the training set, as shown by the experimental results in section 3.2 of this paper. Many approaches were proposed to detect outliers [34, 5, 24]. On the other hand, the CAE proposed in this paper is more natural to detect outliers, because the output of the CAE are assumed to be similar to the objects to be classified.

Effective methods were proposed to train more robust DNNs to defend adversaries [30]. In the adversary training method proposed by Madry et al, the loss function is a min-max optimization problem such that the maximal adversary in a small neighborhood of the training sample is minimized [17]. This approach can reduce the adversaries significantly [17, 33]. Another similar approach is to generate adversaries and add them to the training set [9]. The ensembler adversarial training [27] was introduced for CNN models, which can apply to large datasets such as ImageNet. A fast adversarial training algorithm was proposed, which improves the training efficiency by reusing the backward pass calculations [23]. A less direct approach to enhance the ability for the network to resist adversaries is to make the DNN more stable by introducing Lipschitz constant or $L_{2,\infty}$ regulations of each layer [6, 26, 32].

Increase the robustness of the network in general will increase its ability to defend adversaries and there exist quite a lot of work on the robustness of DNN [11, 34]. Adding noises to the training data is an effective way to increase the robustness [8, Sec.7.5]. The L_1 regulation and $L_{1,\infty}$ normalization are used to increase the robustness of DNNs [31]. Knowledge distilling is also used to enhance robustness and defend adversarial examples [11]. In [10, 16, 21, 29], methods to compute the robust regions of a DNN were given.

The rest of this paper is organized as follows. In section 2, we give the structure for $\mathcal{F}$ and prove the existence of the CAE. In section 3, we give experimental results and show that the CAE indeed improves robustness to resist outliers and adversaries. In section 4, we give the list classification algorithm and the experimental results for it to defend adversaries. In section 5, the proof for the existence of CAE is given. In section 6, conclusions are given

2 Structure of the classification-autoencoder

An autoencoder $\mathcal{F}$ is called a *classification autoencoder (CAE)*,

$$\begin{aligned} \mathcal{F} = \mathcal{D} \circ \mathcal{E} &: \mathbb{I}^n \to \mathbb{I}^n \\ \mathcal{E} &: \mathbb{I}^n \to \mathbb{R}^m \\ \mathcal{D} &: \mathbb{R}^m \to \mathbb{I}^n \end{aligned} \tag{1}$$

if it satisfies the following properties: $\mathcal{E}$ compresses the input images with different labels into disjoint subsets in $\mathbb{R}^m$ called *compression spaces* of the corresponding label and $\mathcal{D}$ will recover the input images to any given precision from the values of $\mathcal{E}$ in their compression spaces. In the rest of this section, we will give the structure of a CAE and prove its existence.

2.1 The classifier-encoder $\mathcal{E}$

The encoder $\mathcal{E}$ will be used as a classifier by trying to compress images with different labels into different subspaces of $\mathbb{R}^m$.

Let $\mathbb{L} = \{0, 1, \ldots, o\} \subset \mathbb{N}$ be the set of labels and $m = om_0$, where $m_0 \in \mathbb{N}_{>0}$ is a super-parameter to be defined by the user. We try to use $\mathcal{E}$ to compress an image with label l to the *compression space* of that label: the k-th coordinates of $\mathbb{R}^m$ for $k = lm_0 + 1, \ldots, m_0 l + m_0$. Define the *mask vectors* $M_l \in \mathbb{R}^m, l = 0, \ldots, o$ for the l-th compress space as follows

$$M_l[j] = \begin{cases} 1, & \text{when } lm_0 + 1 \leq j \leq lm_0 + m_0 \\ 0, & \text{otherwise.} \end{cases}$$

For $y \in \mathbb{R}^m$ and $l \in \mathbb{L}$, the projection weight of y to the l-th compression space is

$$W_l(y) = y \cdot M_l$$

where $\cdot$ is the inner product. In order to realize the goal of CAE, we define the following loss function for $\mathcal{E}$

$$\text{Loss}_1(x, l) = \text{Loss}_{\text{CE}}(A(x)/\gamma_1, l) \tag{2}$$

where $x \in \mathbb{R}^n$, $l \in \mathbb{L}$, $A(x) = \{W_0(\mathcal{E}(x)), \ldots, W_o(\mathcal{E}(x))\}$, CE stands for CrossEntropy, and γ_1 is a super parameter.

We define the *pseudo-label* of $x \in \mathbb{R}^n$ as follows

$$L(x) = \arg\max_{l \in \mathbb{L}} W_l(\mathcal{E}(x)) : \mathbb{R}^n \to \mathbb{L}, \tag{3}$$

that is, $L(x)$ is the label l such that $W_l(\mathcal{E}(x))$ achieves maximum value for all possible $l \in \mathbb{L}$.

An image is called a *problem image* if it is either an outlier or an adversary. We define the following *criterion function* $C(x) : \mathbb{R}^n \to \{0, 1\}$ to identify problems images:

$$C(x) = 1 \text{ if and only if } W_{l_x}(\mathcal{E}(x)) - W_j(\mathcal{E}(x)) > C_0 \tag{4}$$

for all $j \neq l_x$, where $l_x = L(x)$ and C_0 is a threshold super-parameter. If $C(x) = 0$, then x is considered to be a problem image. If $C(x) = 1$, then x is treated as a candidate with label $L(x)$ for further treatment with the decoder $\mathcal{D}$ to be defined in the next section.

Note that it is possible to choose M_l adaptively as follows: if the objects with label l are more complex, then M_l could be larger.

2.2 The decoder $\mathcal{D}$

The decoder $\mathcal{D} : \mathbb{R}^m \to \mathbb{R}^n$ in (1) tries to use the output $\mathcal{E}(x)$ of the network $\mathcal{E}$ to recover x from its compression subspace. We first define a projection function

$$p(y, l) = y \circ M_l : \mathbb{R}^m \times \mathbb{L} \to \mathbb{R}^m$$

where $\circ$ is the Hadamard (element-wise) product. For an $x \in \mathbb{I}^n$, the input to the network $\mathcal{D}$ is $p(\mathcal{E}(x), L(x))$ and the loss function for $\mathcal{D}$ is

$$\text{Loss}_2(x, l) = \text{Loss}_{\text{MSE}}(\mathcal{D}(p(\mathcal{E}(x), l)), x).$$

2.3 The whole network

The network $\mathcal{F}$ is certain composition of $\mathcal{E}$ and $\mathcal{D}$:

$$\mathcal{F}(x) = \mathcal{D}(p(\mathcal{E}(x), L(x))) \tag{5}$$

and the loss function for $\mathcal{F}$ is

$$\text{Loss}(x, l) = \text{Loss}_1(x, l) + \gamma_2 \text{Loss}_2(x, l)$$

where γ_1 and γ_2 are super-parameters.

Let Θ_1 and Θ_2 be respectively the parameter sets of $\mathcal{D}$ and $\mathcal{E}$ and $\Theta = \Theta_1 \cup \Theta_2$. The training algorithm for $\mathcal{F}$ is given below.

Algorithm 1 Training Network $\mathcal{F}$

Require:
The super parameters γ_1, γ_2, γ;
The training set S;
The initial value of Θ: Θ_1^0 and Θ_2^0

Ensure: The trained parameters $\widetilde{\Theta}_1$, $\widetilde{\Theta}_2$.
Input Θ_1^0 and Θ_2^0
Given a sample (x, l) in a training set.
For all l in $\{0, 1, \ldots, o\}$, do
$A(x) = \{W_0(\mathcal{E}(x)), W_1(\mathcal{E}(x)), \ldots, W_o(\mathcal{E}(x))\}$
End for.
$L_1(x, l) = \text{Loss}_{\text{CE}}(A(x)/\gamma_1, l)$
$L_2(x, l) = \text{Loss}_{\text{MSE}}(\mathcal{D}(p(\mathcal{E}(x), l)), x)$
$L(x, l) = L_1(x, l) + \gamma_2 L_2(x, l)$
Consider once gradient descent contains N samples. Let:
$L = \sum_{(x,l) \quad in \quad once \quad gradient \quad descent} L(x, l)$
Fixed all parameters of $\mathcal{D}$, gradient descent Θ_1 as:
$\Theta_1^{k+1} = \Theta_1^k - \gamma L_{\bigtriangledown \Theta_1^k}$
Fixed all parameters of $\mathcal{E}$, gradient descent Θ_2 as:
$\Theta_1^{k+1} = \Theta_2^k - \gamma L_{\bigtriangledown \Theta_2^k}$
γ is the step of gradient descent.
Output Θ_1^{k+1} and Θ_2^{k+1}

In what below, we will give a termination criterion for the algorithm. Since a sample (x, l) should satisfy $C(x) = 1$, we have $\mathcal{E}(x) \cdot M_l - \mathcal{E}(x) \cdot M_i \geq C_0$ for $i \neq l$. In this case,

$$L_1(x, l) = -\ln \frac{e^{\mathcal{E}(x) \cdot M_l / \gamma_1}}{\sum_{i=0}^{o} e^{\mathcal{E}(x) \cdot M_i / \gamma_1}} = -\ln \frac{1}{\sum_{i=0}^{o} e^{-(\mathcal{E}(x) \cdot M_l - \mathcal{E}(x) \cdot M_i)/\gamma_1}} \leq \ln\left(1 + 9e^{-C_0/\gamma_1}\right).$$

So when $L_1(x, l) \leq \ln\left(1 + 9e^{-C_0/\gamma_1}\right)$ for most (x, l) in the training set and $L(x, l)$ is small enough, we stop the algorithm.

2.4 The classification

We use the autoencoder $\mathcal{F}$ as a classifier as follows. By a *problem image*, we mean an outlier or an adversary sample.

Algorithm 2 CAE

The input: $x \in \mathbb{I}^n$, super parameters C_0, b_s, b_u.
The output: a label of x or "x is a problem image".

S1 Compute $\mathcal{F}(x)$ with (5), and $L(x)$ with (3), $C(x)$ with (4).

S2 If $C(x) = 0$, then output: "x is a problem image."

S3 If $||\mathcal{F}(x) - x|| \leq b_s$ for a given threshold $b_s \in \mathbb{R}_{>0}$, then output: "x has label $L(x)$."

S4 If $||\mathcal{F}(x) - x|| \geq b_u$ for a given threshold $b_u \in \mathbb{R}_{>0} (b_s < b_u)$, then output: "$x$ is a problem image."

S5 This step treats those x satisfying $b_s < ||\mathcal{F}(x) - x|| < b_u$.

S5.1 Let $d_l = ||\mathcal{D}(p(\mathcal{E}(x), l)) - x||$, for all $l \in \mathbb{L}$.

S5.2 If $d_{L(x)} \leq d_l$ for all $l \in \mathbb{L}$, then output: "x has label $L(x)$."

S5.3 Or output: "x is a problem image."

We explain the algorithm as follows. In Step S2, we eliminate the inputs which do not have obvious characteristic of the objects to be classified. In Step S3, when $\mathcal{F}(x)$ and x are similar enough, we think x belongs to $\mathbb{O}$ and its label is given. In Step S4, when x is not anything like $\mathcal{F}(x)$, we think x should be a problem image. In Step S5, we are not sure whether x is an element in $\mathbb{O}$ or a problem image. In this case, we give a refined analysis by computing all $\mathcal{D}(p(\mathcal{E}(x)), l)$ and checking if $\mathcal{D}(p(\mathcal{E}(x)), L(x))$ is more similar to x than $\mathcal{D}(p(\mathcal{E}(x)), l)$ for $l \neq L(x)$.

2.5 Existence of the network

The main idea of the network designed in section 2 is that images with the same label are mapped into a compression space and images with different labels are compressed into disjoint compression spaces. Furthermore, the images in $\mathbb{O}$ can be approximately recovered from their compression spaces. In this section, we prove that DNNs with such properties exist in certain sense.

Let $x \in \mathbb{R}^n$ and $r \in \mathbb{R}_{>0}$. When r is small enough, all images in

$$\mathbb{B}(x, r) = \{x + \eta \,|\, \eta \in \mathbb{R}^n, ||\eta|| < r\}$$

can be considered to have the same label with x. Therefore, the object $\mathbb{O}$ to be classified can be considered as bounded open sets in $\mathbb{R}^n$. This observation motivates the existence result given below. The proof of the theorem is given in the section 5.

Theorem 2.1. *Suppose that the objects to be classified are a finite number of disjoint bounded open sets $S_l \subset \mathbb{I}_0^n$ with labels $l \in \mathbb{L} = \{0, \ldots, o\}$. Then, for any $m \in \mathbb{N}_{>0}$ and for any $\epsilon > 0$, $\gamma > 0$, there exist DNNs $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathcal{D} : \mathbb{R}^m \to \mathbb{R}^n$ such that*

1. *Let $K = \{x \in \bigcup_{i=0}^{o} S_i \,|\, \mathcal{E}(x) \in \mathcal{E}(S_i) \cap \mathcal{E}(S_j)$ for some $i \neq j\}$. Then $\frac{V(K)}{\sum_i V(S_i)} < \epsilon$.*

2. *Let $\widehat{S}_l = \{x \in S_l \,|\, ||\mathcal{D}(\mathcal{E}(x)) - x|| < \gamma\}$ for $l \in \mathbb{L}$. Then $\frac{V(\widehat{S}_l)}{V(S_l)} > 1 - \epsilon$.*

3 Experimental results on classification

In this section, we will give experimental results based on MNIST dataset. The precise structures of the networks used in the experiments are given in the Appendix, and the super-parameters are $\gamma_1 = 50$ and $\gamma_2 = 1$, $C = 80$, $b_l = 0.04$, and $b_u = 0.09$.

3.1 Accuracy on the test set

The test set $\mathbb{T}$ of MNIST contains 10000 samples. Let $\text{Loss}(x, l) = ||x - y||_2^2/784$ and $\text{Loss}_B(x) = \text{Loss}(\mathcal{F}(x), x)$. Then we have: the average error for the samples in the test set is

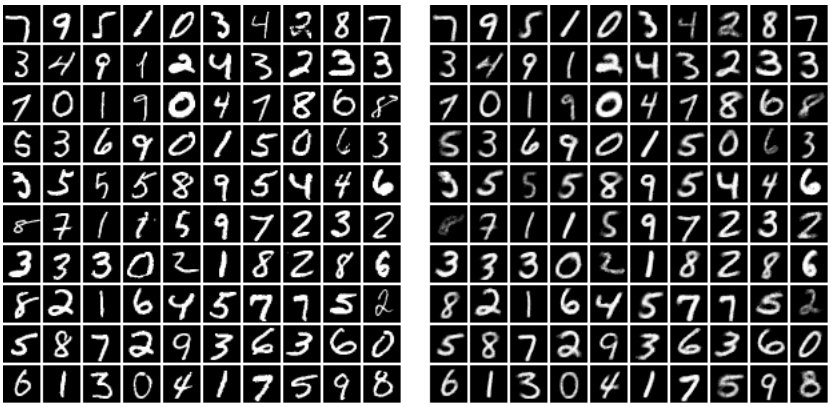
$$\frac{\sum_{(x,l)\in\mathbb{T}} \text{Loss}_B(x)}{10000} \approx 0.023.$$

78 of them are considered to be problem images by S2. We give the distribution of the Loss_B of 9922 non-problem images in Table 1.

<0.04	0.04-0.05	0.05-0.06	0.06-0.07	0.07-0.8	0.08-0.09	0.09-0.10	>0.10
9581(31)	220(8)	85(4)	26(5)	4(0)	2(0)	1(0)	3(2)

Table 1: Distribution of Loss_B. The number inside the bracket is the number of samples given the wrong label by $\mathcal{F}$

We can see that 9550 samples are given the correct labels by S3, and some of then are given in Figure 1. 4 samples are considered as problem images by S4, which are given in Figure 2. In Figure 3, we give the samples which are considered as problem images by S2. 31 samples are given the wrong labels by S3, which are given in Figure 4. 337 samples need step S5.



(a) (b)

Figure 1: Correct label by S3. (a): the input, (b): the corresponding output

(a) (b)

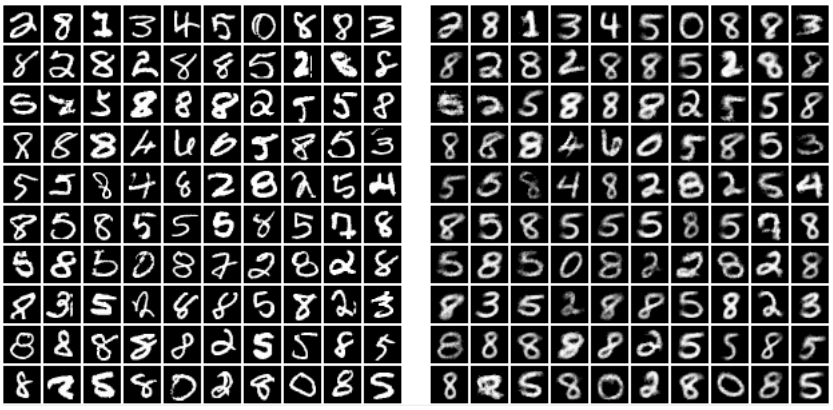
Figure 2: Problem images by S4. (a): the input, (b): the corresponding output

Figure 3: Problem images by S2

(a) (b)

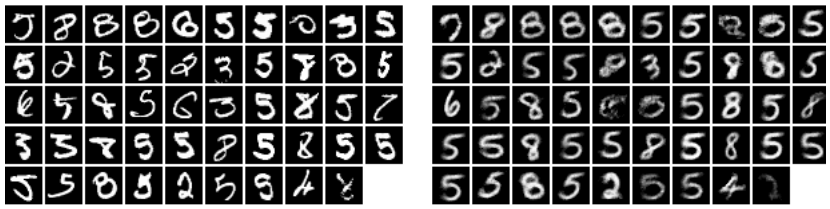
Figure 4: Wrong label by S3. (a): the input, (b): the corresponding output

Among the 337 samples need step S5, 280 of them are given the correct labels, 49 of them are considered problem images, and 8 of them are given the wrong labels, some of which are given in Figures 5, 6, and 7, respectively.



(a) (b)

Figure 5: Correct labels by S5. (a): the input, (b): the corresponding output



(a) (b)

Figure 6: Problem images by S5. (a): the input, (b): the corresponding output

(a) (b)

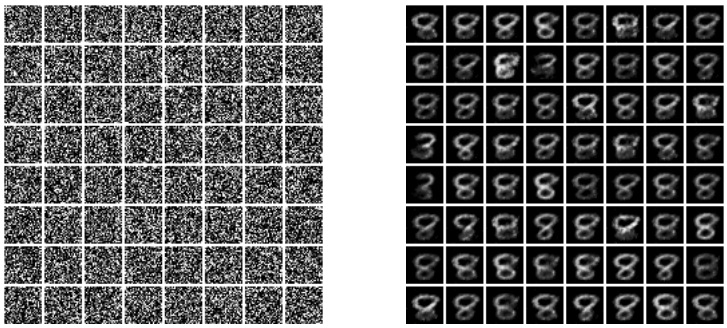
Figure 7: Wrong label by S5. (a): the input, (b): the corresponding output

To summarize, 9830(=9550+280) samples are given the correct labels, 39=(31+8) samples are given the wrong labels, and 131(=78+49+4) samples are considered as problem images.

3.2 Defend outliers

In this section, we use four types of outliers to check the ability of $\mathcal{F}$ on defending outliers.

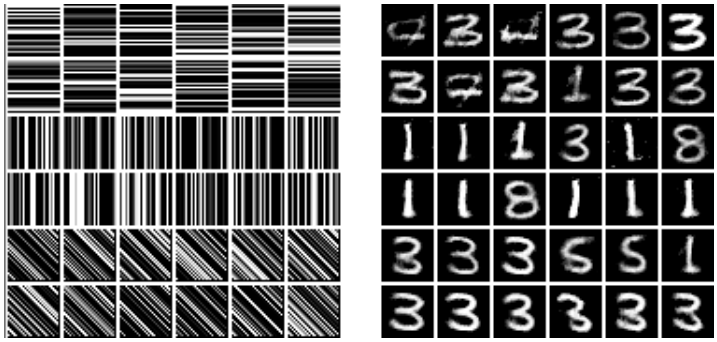
For the type 1 outliers, we use randomly generated noise images. We use three kinds of random noise images. Firstly, let $\mathcal{D}_n = \alpha\mathcal{N} + \beta\mathcal{U}$, where $\mathcal{N}$ is the normal distribution and $\mathcal{U}$ is the uniform distribution, $\alpha, \beta \in \mathbb{R}$. Let $\mathbb{S}_1 = \{x_{\text{noise}} \in \mathbb{R}^{28\times 28}\}$, where every coordinate of $x_{\text{noise}} \in S$ is generated by $\mathcal{D}_n$ (iid). We set $\alpha = \beta = 1/2$ in the experiment. The 64 type 1 outliers in Figure 8(a) are used as the input and the output $\mathcal{F}(x)$ for these inputs are given in 8(b). It can be seen that the outputs are quite different from the input and all type 1 samples are easily recognized as outliers by S4.



(a) (b)

Figure 8: Type 1.1 outliers. (a): Input, (b): output

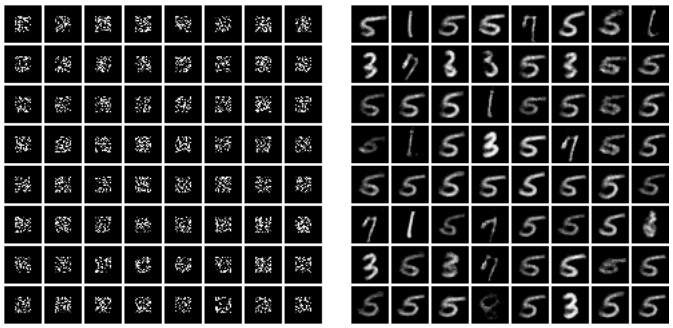
Secondly, we give some random samples with structures. They are created by randomly generating a row, a column, or the diagonal, and then by copying the row, column or the diagonal randomly to cover the matrix. We give 64 samples created in this way, and their outputs in Figure 9. All of them were recognized as problem images by S4.



(a) (b)

Figure 9: Type 1.2 outliers. (a): Input, (b): output

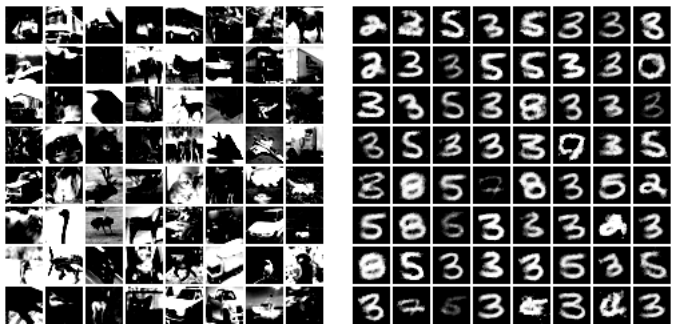
Thirdly, the middle 144 pixels of an image in $\mathbb{O}$ are given by the normal distribution $\mathcal{N}$ (iid). We give 64 samples created in this way and their outputs of $\mathcal{F}$ in Figure 10. It is easy to see that the input and the output are quite different and all of them were recognized as problem images by S4.



(a) (b)

Figure 10: Type 1.3 outliers. (a): Input, (b): output

Finally, we reduce the sizes of images in Cifar-10 to 28×28 and change the image from color to grey. These images might be animals or ships, but not numbers, so they are outliers to $\mathcal{F}$ trained with MNIST. We give 64 such samples in Figure 11.



(a) (b)

Figure 11: Type 1.4 outliers. (a): Input, (b): output

We compare our network with a network $\mathcal{H}$ which is trained with MNIST plus 60000 noise samples

with a new label -1, representing outliers. The noise samples, denoted as N_1, are generated by $\alpha\mathcal{N} + \beta\mathcal{U}$, where α is a random number in $(0, 1)$ and $\beta = 1 - \alpha$.

We compare $\mathcal{F}$ and $\mathcal{H}$ for four kinds of outliers given in Figures 8, 9, 10, and 11. It can be seen that our network can defend all kind of outliers, but $\mathcal{H}$ can only defend the samples similar to the random training samples.

Types of outliers	Our network $\mathcal{F}$	Network $\mathcal{H}$
Figure 8	100%	99%
Figure 9	100%	97%
Figure 10	100%	3%
Figure 11	100%	26%

Table 2: Percentages for $\mathcal{F}$ and $\mathcal{H}$ to recognize outliers

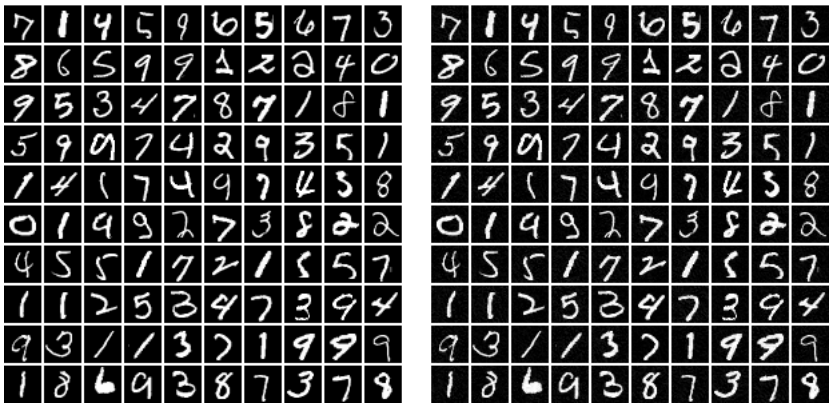
Finally, we compare the robustness of the two networks to defend outliers. Let X_n be a noise picture in N_1. For $\mathcal{H}$, we use gradient descent for X_n to make $L_{\text{CE}}(X_n, -1)$ smaller. For $\mathcal{F}$, we use gradient descent for X_n to make $L_{\text{MSE}}(\mathcal{D}(p(\mathcal{E}(X_n), L(X_n)) - X_n))$ smaller. Two types of outliers are generated: (1) each pixel of X_n is changed up to 0.1; and (2) each pixel of X_n is changed up to 0.2. The network $\mathcal{F}$ can still recognize 100% of them as outliers. $\mathcal{H}$ can only recognize 80% of samples (1) and 20% samples (2). So, we can see that $\mathcal{F}$ is very robust to recognize outliers.

To summarize, the network $\mathcal{F}$ can recognize almost all outliers and this is one of the main advantages of the network introduced in this paper.

3.3 Defend adversaries

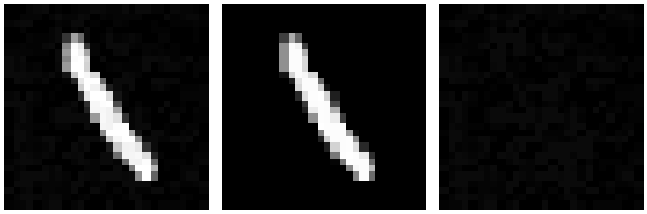
In this section, we check the ability of $\mathcal{F}$ to defend adversaries. For $\mathcal{F}$, $\mathcal{E}(x)$ is the network for classification, so we create adversaries for $\mathcal{E}(x)$. We use samples in the training set to generate 3 types of adversaries.

The type 1 of adversaries (L_{inf} adversary) are created with FGSM [9] and the change for each coordinate is ≤ 0.1. Some of type 1 adversaries are given in Figure 12. In Figure 13, we compare an enlarged adversary and the standard image.



(a) (b)

Figure 12: Type 1 adversaries. (a): standard samples, (b): adversary.



(a) (b) (c)

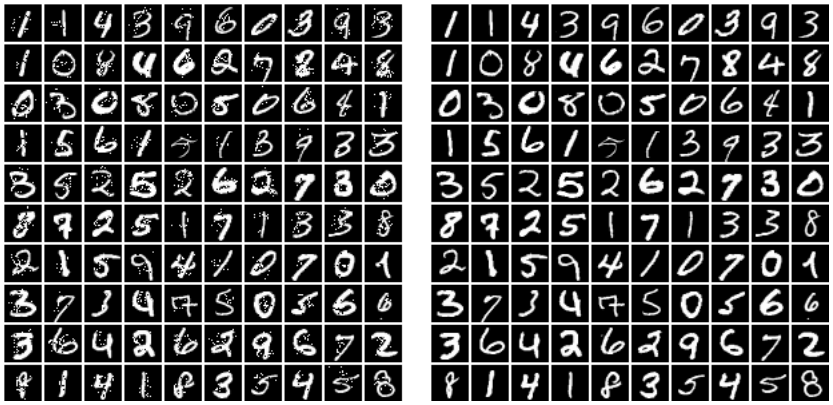
Figure 13: (a) adversaries, (b) standard sample, (c) the difference.

The creation rate for type 1 adversaries by $\mathcal{E}$ is about 35%. We use 1000 type 1 adversaries as input to $\mathcal{F}$. Among the 1000 type 1 adversaries, 733 samples are considered as problem images, 267 samples are given wrong labels. So, for about 26.7% of the type 1 adversaries, $\mathcal{F}$ gives wrong labels and 73.3% of them are considered as problem images. Therefore, for about $9.35\% = 35\% \cdot 26.7\%$ of the training samples. In Figure 14, we give some samples which are given the wrong labels.

(a) (b) (c)

Figure 14: (a): adversaries, (b): output, (c) output of $\mathcal{D}(g(\mathcal{E}(x), l_x))$, l_x is the label of x.

The type 2 adversaries (L_0 adversary) are obtained by changing 80 coordinates of a training sample. Some type 2 adversaries are given in Figure 15(a) and Figure 15(a) gives the corresponding standard images.



(a) (b)

Figure 15: Type 2 adversaries. (a): adversaries, (b) standard samples.

For the encoder $\mathcal{E}$, the creation rate for type 2 adversaries from the training set is about 96%, and we use 1000 adversaries as input to $\mathcal{F}$. Among the 1000 adversaries, 751 are considered as problem images and 249 are given the wrong labels. So, for about 24.9% of the type 2 adversaries, $\mathcal{F}$ gives the wrong labels and

75.1% of them are recognized as problem images by $\mathcal{F}$. Then, the adversary creation rate of the training set for $\mathcal{F}$ is $23.5\% = 96\% \cdot 24.9\%$. In Figure 16, we give some samples which are given the wrong labels.

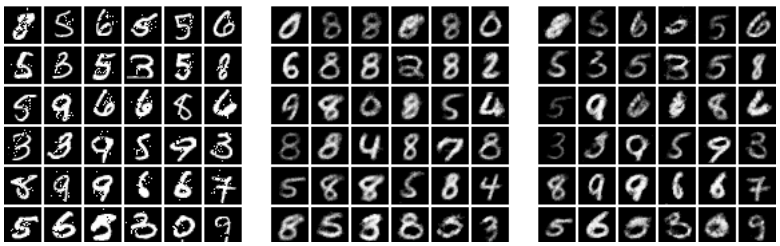


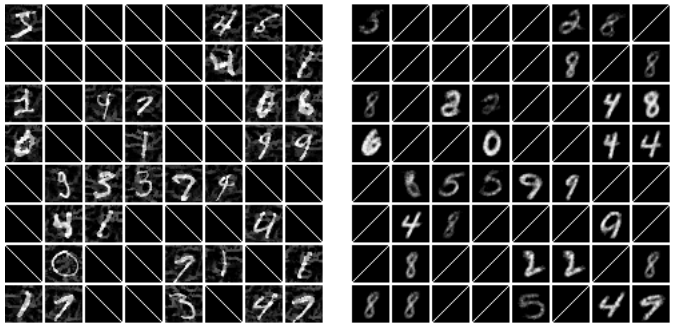
Figure 16: (a): adversaries, (b): output, (c) output of $\mathcal{D}(g(\mathcal{E}(x), l_x))$, l_x is the label of x.

The type 3 adversaries are called *strong adversaries* and are generated as follows. Let $\mathcal{C}$ be a standard CNN classification network. First, generate an adversary of $\mathcal{C}$ with FGSM such that each pixel changes ≤ 0.2. Second, for the adversary x generated in step 1 with l as the wrong label, use gradient descent on x to make $L_{\mathrm{CE}}(\mathcal{F}(x), l)$ bigger and the change for each pixel is ≤ 0.2.

Figure 17 contains 64 type 3 adversaries and their output by $\mathcal{F}$. Among these adversaries, 31 are seen as problem images (Figure18), 16 are given the correct labels (Figure 19), 17 are given the wrong labels (Figure20). We try 1000 adversaries of this type, and $\mathcal{F}$ gives wrong labels for about 37% of them.

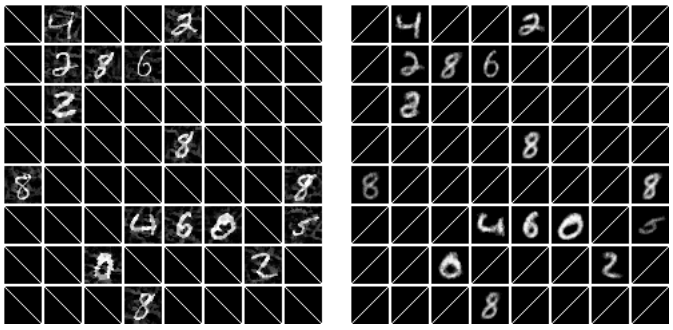
(a) (b)

Figure 17: Type 3 adversaries. (a) adversaries, (b) output from $\mathcal{F}$.



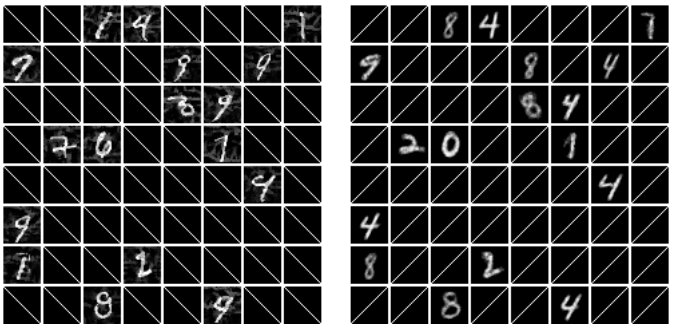
(a) (b)

Figure 18: Problem images.



(a) (b)

Figure 19: $\mathcal{F}$ gives correct labels.



(a) (b)

Figure 20: $\mathcal{F}$ gives wrong labels.

We will compare with the well-known min-max model of adversary training proposed in [17]. The network structure of the min-max model can be found the Appendix. For the type 1.i adversaries, we mean type 1 adversaries such that the change of each pixel is $\leq 0.i$ for $i = 1, 2, 3$. Similarly, type 2.i adversaries are type 2 adversaries by changing i pixels for $i = 40, 60, 80$. The results in Table 3 are the creation rates of adversaries from the training set.

Adversaries	$\mathcal{E}$	CAE	min-max model	LCAE	NLabels
1.1	38%	9%	6%	$< 1\%$	3.04
1.2	83%	21%	15%	$< 1\%$	3.10
1.3	96%	26%	55%	1%	3.13
2.40	85%	21%	55%	$< 1\%$	3.58
2.60	92%	24%	76%	1%	3.64
2.80	97%	29%	85%	2%	3.70

Table 3: Creation rates of adversaries

We have the following observations. (1) $\mathcal{F}$ is better than the min-max model for more difficult adversaries. The performance of $\mathcal{F}$ is more stable than that of the min-max model. (2) We should note that the outputs of $\mathcal{F}$ and min-max models are different. For these adversaries, $\mathcal{F}$ can recognize them as "problem images", but cannot give them the correct label. This fact can be seen from the values in the second column in Table 3, which means that for almost all samples, $\mathcal{E}$ and hence $\mathcal{F}$ cannot give the correct label. On

the other hand, the min-max model gives the correct labels for 45% of the type 1.3 adversaries. (3) Also note that $\mathcal{E}$ can be improved with the methods such as the min-max model to increase its ability to resist adversaries. That is, combining the network introduced in this paper and the min-max model, it is possible to gain more ability to resist adversaries.

Finally, we give the comparison results for three kinds of strong adversaries. They are generated as follows: first generate an adversary such that the change of each pixel is ≤ 0.2 and then generate strong adversaries by further changing each pixel up to $0, 0.1, 0.2$. In Table 4, the wrong accuracy rates are given. We can see that CAE gives lower wrong accuracies for strong adversaries.

Adversaries	CAE	min-max model	LCAE	NLabels
3 (0.2+0)	3%	2%	$<1\%$	2.87
3 (0.2+0.1)	12%	18%	1%	3.22
3 (0.2+0.2)	37%	51%	4%	3.94

Table 4: Wrong accuracy rates

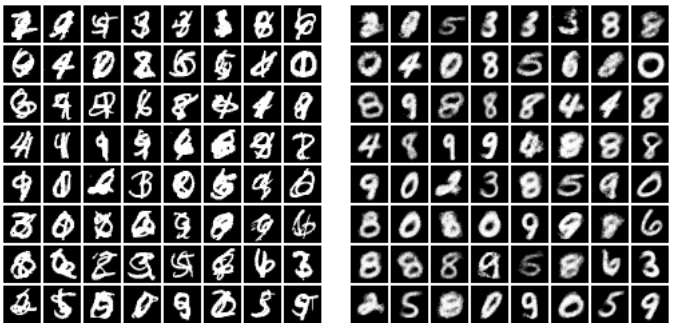
3.4 Outliers with certain characteristics of numbers

Besides the outliers and adversaries considered in sections 3.3 and 3.2, there exist images which are not numbers, but have certain characteristics of numbers. In this section, we give two classes of such samples.

3.4.1 Mixing two numbers

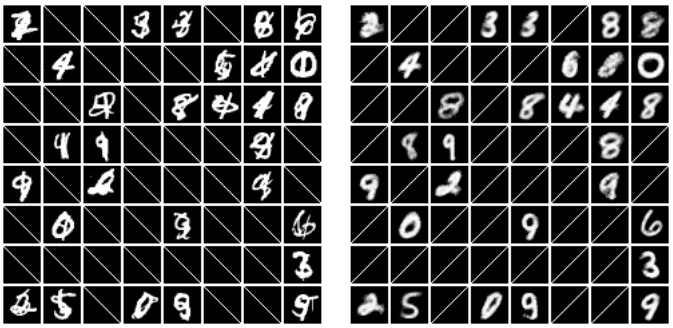
The outliers are obtained by adding two images from MNIST with different labels and 64 such samples and their outputs are given in Figure 21. 29 of them are given the wrong labels and 35 of them are considered as problem images, which are given in Figure 22 and Figure 23, respectively.

We have tried 1000 samples of this kind, and about 59% of them are recognized by $\mathcal{F}$ as problem images, and only 2% of them are recognized by $\mathcal{H}$ in section 3.2 as outliers.



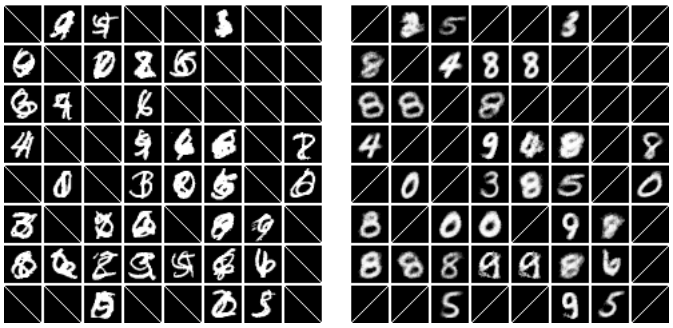
(a) (b)

Figure 21: Mixing two numbers. (a): Input, (b): output



(a) (b)

Figure 22: $\mathcal{F}$ gives the wrong labels.



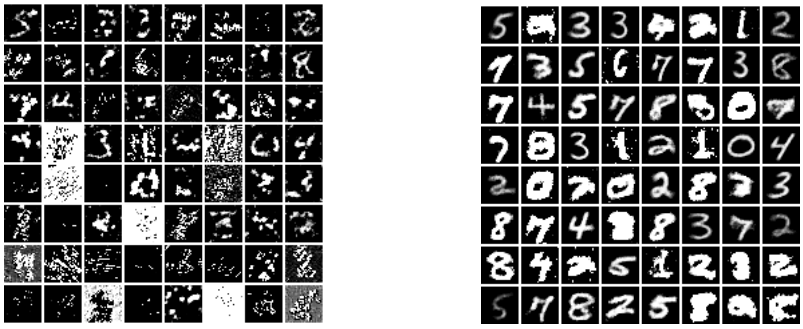
(a) (b)

Figure 23: $\mathcal{F}$ considers as problem images.

3.4.2 Adversary of outlier for $\mathcal{E}$

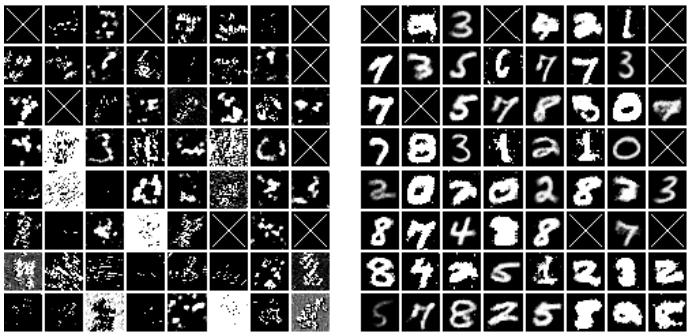
The adversaries of outliers are intentionally generated such that the encoder $\mathcal{E}$ thinks them as numbers with high probability, but they look quite different from numbers. Precisely, we select samples satisfying $|\mathcal{E}(x) \cdot V_{l_x} - \mathcal{E}(x) \cdot V_j| > 200$ for $j \neq l_x = L(x)$. Figure 24(a) gives 64 such outliers which will be used as the input. 56 of them are considered as problem images, which are given in Figure 25. 8 of them are considered as numbers, which are shown in Figures 26, respectively.

We try 500 samples of this type with CAE, and about 77% of them are seen as problem images by $\mathcal{F}$, but only 32% of them are recognized by $\mathcal{H}$ in section 3.2 as outliers. Thus, CAE has strong ability to recognize adversaries of outliers, that is, although the encoder thinks them as numbers, the decoder can be used to recognize them as outliers.



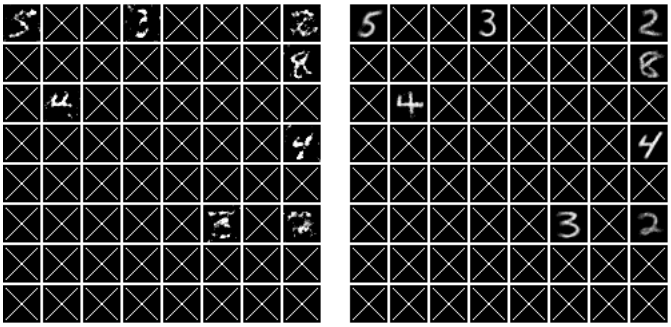
(a) (b)

Figure 24: Type 2 outliers. (a): Input, (b): output



(a) (b)

Figure 25: Problem images.



(a) (b)

Figure 26: 8 outliers are considered as numbers. (a): input, (b): output.

4 List classification to resist adversaries

It is widely believed [2, 22] that adversaries are inevitable for the current DNN framework. A possible way to alleviate the problem is to give several labels instead of one. In this section, we give such an approach based on the CAE introduced in section 2.

4.1 The list classification

The list classification algorithm is motivated by Figures 14 and 16, where the label $L(x)$ is wrong, but $\mathcal{D}(g(\mathcal{E}(x), l_x))$ gives a very close approximation to the input x. Therefore, we will compare x with $\mathcal{D}(g(\mathcal{E}(x), l))$ for all l and output those similar to x with "high probabilities". We first define a distance between two images.

Definition 4.1. *Let $x \in \mathbb{R}^n$, and $A(x)$ the average of all coordinates of x. For $a \in \mathbb{R}$, define $S(a) = 1$ if $a > 0$, and $S(a) = 0$ if $a \leq 0$. Define $\widehat{x} = (\widehat{x}_1, \ldots, \widehat{x}_n) \in \{0,1\}^n$, where $\widehat{x}_i = S(x_i - A(x))$. We call $\widehat{x}$ the standardization of x. Also define* $\text{Dis}(x, y) = \frac{A(x-y)}{A(x+y)}$ *for $x, y \in \mathbb{R}^n$.*

Suppose that $\mathcal{F}$ in (1) is trained. We do the image classification by giving several possible answers.

Algorithm 3 LCAE

The input: $x \in \mathbb{I}^n$, super-parameters B and D (in MNIST experiments, we choose B=14 and D=0.1).
The output: a list of labels and the corresponding output images of $\mathcal{F}$.

S1 Compute $\mathcal{F}(x)$ with (5), and $L(x)$ with (3).

S2 Let $z_l = \mathcal{D}(g(\mathcal{E}(x), l))$, for all $l \in \mathbb{L}$.

S3 Compute $L_l = ||z_l - x||$, $D_l = \text{Dis}(\widehat{z}_l, \widehat{x})$, and $E_l = D_l L_l$ for $l \in \mathbb{L}$.

S4 The probability for x to be an outlier is $P_{-1} = \frac{1-e^{-(BE_m)^2}}{1+e^{-(BE_m)^2}}$, where $E_m = \min_l E_l$.

S5 The probability for x to have label l is $P_l = (1 - P_{-1})\bar{P}_l$, where $\bar{P}_l = \frac{1/E_l}{\sum_j 1/E_j}$ and $l \in \mathbb{L}$.

S6 Output (l, z_l) if $P_l > D$ for $l \in \mathbb{L}$.

We give an illustrative example. In Figure 27, the images $x, z_l, \widehat{x}, \widehat{z}_l$ are given. In Table 5, the probabilities P_l are given. The probability of the picture to have label 1 is 88.80% and all other probabilities are smaller than 0.1.

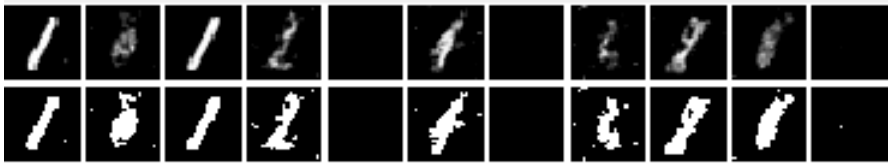


Figure 27: The first row is x and z_l in S2; the second row is $\widehat{x}$ and $\widehat{z}_l$.

$l =$	-1	0	1	2	3	4	5	6	7	8	9
$\text{Dis}(x, z_l)$		0.0473	0.0029	0.0290	0.0727	0.0266	0.0727	0.0515	0.0255	0.0357	0.0727
$\text{Dis}(\widehat{x}, \widehat{z}_l)$		0.3848	0.069	0.2759	1	0.2820	1	0.4951	0.2902	0.2773	1
E_l		0.0182	0.0002	0.0080	0.0727	0.0075	0.0727	0.0255	0.0074	0.0099	0.0727
$\bar{P}_l$		0.0098	0.8881	0.0222	0.0024	0.0237	0.0024	0.0070	0.0240	0.0179	0.0024
P_l	0.01%	0.98%	88.80%	2.22%	0.24%	2.37%	0.24%	0.70%	2.40%	1.79%	0.24%

Table 5: The probabilities for x to have label l.

4.2 Accuracy on the test set

We use LCAE to the test set of MNIST, which contains 10000 samples. The structure of the network $\mathcal{F}$ is given in the Appendix. The number of samples whose output contains the correct label is 9997. The number of samples whose output only contains the correct label is 5711. The number of samples whose output contains outlier is 1. In Table 6, we give the number of labels. The average number of labels in the output is 1.7228

Number of labels	1	2	3	4	5	6
Number of samples	5712	2122	1500	560	104	2

Table 6: Number of labels in the output

In Figure 28, we give the sample whose output contains outlier. In Figures 29, 30, 31, we give the three samples whose outputs do not contain the correct label. In Table 7, we give the probabilities of these figures.

Figure 28: Output contains outlier. (The first line is x and z_l in S2, the second is $\widehat{x}$ and $\widehat{z}_l$)

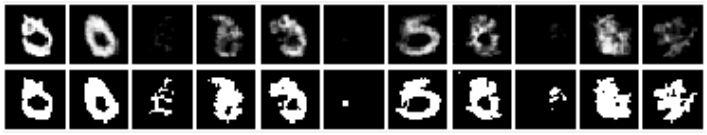


Figure 29: Output does not contain the correct label 5.

Figure 30: Output does not contain the correct label 5.

Figure 31: Output does not contain the correct label 5.

$l=$	-1	0	1	2	3	4	5	6	7	8	9
P_l	12.5%	13.1%	11.7%	12.9%	4.52%	4.20%	4.20%	9.39%	8.50%	11.7%	7.21%
P_l	0.25%	36.2%	1.91%	6.94%	14.7%	1.25%	8.27%	12.4%	1.34%	12.5%	4.08%
P_l	0.01%	0.63%	0.75%	0.43%	59.5%	0.43%	4.68%	0.43%	0.65%	2.64%	29.7%
P_l	0.01%	0.82%	0.96%	0.77%	76.6%	0.53%	7.42%	0.53%	0.56%	4.95%	6.77%

Table 7: The probabilities for the inputs in Figure 28, 29, 30, 31

4.3 Defend adversaries

We check the ability of the list classification model to defend adversaries for the three types of adversaries given in section 3.3. The creation rates of adversaries for LCAE are given in the column "LCAE" of Tables 3 and 4, and the numbers of labels in the outputs of LCAE are given in column "NLabels". From these tables, we can see that the list classification can almost always give the correct labels for various adversaries by outputting about 3.5 labels. Thus, the list classification can be considered to give an uncertain but lossless classification. In the rest of this section, we give some examples.

In the following figures, we give six adversaries and the outputs of LCAE. In Table 8, we give the probabilities for these adversaries.

Figure 32: This is a type 1.1 adversarial of 7; the correct label 7 is given. $P_7 = 27.24\%$. (The first row is x and z_l in S2, the second row is $\widehat{x}$ and $\widehat{z}_l$)

Figure 33: This is a type 1.2 adversarial of 5; the wrong label is 6; $P_5 = 7.22\%$.

Figure 34: This is a type 2.20 adversarial of 5; the wrong label is 9; $P_5 = 12.17\%$.

Figure 35: This is a type 2.80 adversarial of 5; the wrong label is 8; $P_5 = 7.18\%$.

Figure 36: This is a type 3 adversarial of 7; wrong label is 8; $P_7 = 14.97\%$.

Figure 37: This is a type 3 adversarial of 5; wrong label is 9; $P_5 = 9.03\%$.

$l=$	-1	0	1	2	3	4	5	6	7	8	9
P_l	0.74%	6.79%	3.45%	3.45%	3.67%	12.2%	7.34%	3.45%	27.2%	10.6%	20.9%
P_l	0.41%	2.58%	15.8%	4.29%	7.86%	0.89%	7.22%	31.4%	0.89%	24.6%	3.92%
P_l	3.32%	4.13%	3.89%	3.89%	4.02%	15.0%	12.1%	11.2%	3.89%	16.6	21.7%
P_l	2.80%	6.92%	3.26%	9.11%	3.03%	18.4%	7.18%	11.6%	2.67%	21.8%	13.1%
P_l	3.84%	8.69%	4.69%	14.1%	8.86%	9.76%	4.69%	4.69%	14.9%	15.5%	10.1%
P_l	3.03%	4.18%	9.92%	3.84%	3.88%	15.9%	9.03%	13.2%	3.84%	15.5%	17.7%

Table 8: Probabilities for the samples in Figures 32, 33, 34, 35, 36, 37.

4.4 Recognize outliers

In section 3.2, it is already shown that CAE can recognize most outliers. In this section, we give a more detailed analysis on the ability of $\mathcal{F}$ to defend outliers by giving the results of LCAE. In Table 9, we give the percentages of the samples whose P_{-1} is lager than 50% and 80%, respectively. From the table, we see that $P_{-1} > 50\%$ for all images. If a sample has $P_{-1} > 50\%$, it is almost certain to be an outlier.

Outlier	$P_{-1} > 50\%$	$P_{-1} > 80\%$
Figure 8	100%	100%
Figure 9	100%	100%
Figure 10	100%	64%
Figure 11	100%	98%

Table 9: The percentages of outliers which have probability bigger than 50% and 80%

In Figures 38 and 39, two outliers in Figures 10 and 11 are given, respectively. Their corresponding probabilities are given in Table 10.

Figure 38: An outlier from Figure 10 and the output.

Figure 39: An outlier in Figure 11 and the output.

$l=$	-1	0	1	2	3	4	5	6	7	8	9
P_l	79.5%	2.20%	2.44%	2.17%	2.20%	1.32%	2.73%	1.32%	2.45%	2.29%	1.36%
P_l	99.0%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%

Table 10: The probabilities for the images in Figures 38 and 39.

4.5 Outliers with certain characteristics of numbers

To apply LCAE to 1000 outliers in section 3.4.1, for only about 7% of the samples, $P_{-1} \geq 10\%$. This is easy to understand: since the images are obtained by adding two numbers, they have strong characteristics of numbers.

On the other hand, this class of outliers suggests another possible application of LCAE: find numbers from outliers which have certain characteristics of numbers. We use 1000 this kind of outliers as input to LCAE. For about 54% of these outliers, the two numbers used to form the images are found by LCAE, and for 99% of them, one of the two numbers is found. We give three examples in Figures 40, 41, 42, and the corresponding probabilities are given in Table 11.

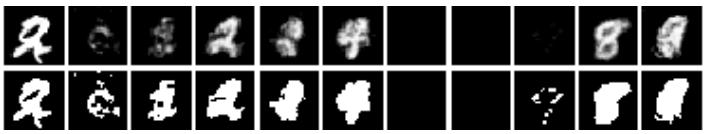


Figure 40: LCAE finds the two number 2 and 9 used to form the image.

Figure 41: LCAE finds one of the two number 4 and 5 used to form the image.

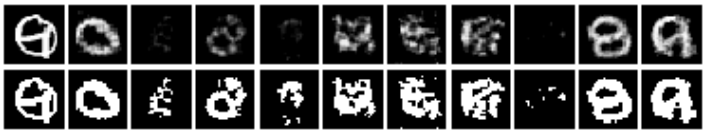


Figure 42: LCAE finds none of the two numbers 0 and 4 used to form the image.

$l=$	-1	0	1	2	3	4	5	6	7	8	9
P_l	3.07%	3.25%	7.39%	20.4%	11.4%	12.2%	1.73%	1.73%	2.14%	15.41%	21.1%
P_l	2.50%	3.31%	3.99%	9.02%	10.0%	12.3%	4.44%	1.94%	6.83%	21.9%	23.6%
P_l	3.34%	6.92%	3.02%	7.64%	3.46%	7.73%	5.61%	10.83%	2.79%	25.4%	23.2%

Table 11: The probabilities for the images in Figures 40, 41, 42.

For the adversary of outliers given in section 3.4.2, the performance of LCAE is similar to that of CAE. For 500 adversaries of outliers, we give the performance of LCAE in Table 12. That is, about 72% of them are seen as outliers. We also give two samples in Figures 43 and 44, and their corresponding probabilities are given in Table 44.

Outliers	$P_{-1} > 10\%$	$P_{-1} > 30\%$	$P_{-1} > 50\%$
4	72%	57%	22%

Table 12: The proportion of outlier that have probability bigger than 10%, 20%, 30%

Figure 43: The image is seen as an outlier: $P_{-1} > 50\%$.

Figure 44: The image is seen as one of the numbers: $1, 4, 7, 8, 9$.

$l=$	-1	0	1	2	3	4	5	6	7	8	9
P_l	52.5%	2.39%	2.39%	2.48%	4.15%	7.52%	5.85%	2.39%	5.90%	6.89%	7.47%
P_l	1.68%	1.46%	12.5%	3.01%	7.04%	24.1%	1.46%	1.46%	15.1%	13.1%	18.9%

Table 13: The probabilities for the images in Figures 43 and 44.

5 Proof of Theorem 2.1

We first consider the case where the objects to be classified consist of a single connected open set. Introduce the notations: $\mathbb{I}_0 = (0, 1)$ and $V(S)$ is the volume of S for $S \subset \mathbb{R}^n$.

Lemma 5.1. *Let S be a bounded and connected open set in $\mathbb{R}^n$ and A a bounded and connected open set in $\mathbb{R}^m$. For any $\epsilon, \gamma \in \mathbb{R}_{>0}$ and any $m \in \mathbb{N}_{>0}$, there exist functions $E : \mathbb{R}^n \to \mathbb{R}^m$ and $D : \mathbb{R}^m \to \mathbb{R}^n$, which are sectional-continuous functions with a finite number of continuous regions and satisfy*

0. *$E(x) = 0$ if $x \notin S$ and $D(y) = 0$ if $y \notin A$.*

1. *$V_1 = \{x \in S \subset \mathbb{R}^n \,|\, E(x) \in A\}$ satisfies $\frac{V(V_1)}{V(S)} > 1 - \epsilon$;*

2. $V_2 = \{y \in A \subset \mathbb{R}^m \,|\, D(y) \in S\}$ *satisfies* $\frac{V(V_2)}{V(A)} > 1 - \epsilon$*;*

3. $V_3 = \{x \in S \subset \mathbb{R}^n \,|\, ||x - D(E(x))|| < \gamma\}$ *satisfies* $\frac{V(V_3)}{V(S)} > 1 - \epsilon$.

Proof. Without loss of generality, assume $S \subset \mathbb{I}_0^n$ and $A \subset \mathbb{I}_0^m$. Let $k \in \mathbb{N}_{>0}$ and

$$C^{k,d_1,d_2,\ldots,d_n} = (\frac{d_1}{k}, \frac{d_1+1}{k}) \times (\frac{d_2}{k}, \frac{d_2+1}{k}) \cdots \times (\frac{d_n}{k}, \frac{d_n+1}{k}),$$

where $d_i \in \{0, 1, 2, \ldots, k-1\}$. Let

$$S_k = \{C^{k,d_1,d_2,\ldots,d_n} | C^{k,d_1,d_2,\ldots,d_n} \subset S\}.$$

It is easy to see that $V(S_k) = t_k/k^n$, where t_k is the number of cubes in S_k. When k becomes lager, $V(S_k)$ will increase and approach to $V(S)$. Then, we can choose a k such that

$$k > \frac{\sqrt{m}}{\gamma} \text{ and } \frac{V(S_k)}{V(S)} = \frac{\frac{t_k}{k^n}}{V(S)} > 1 - \epsilon. \tag{6}$$

For simplicity, let $S_k = \{C_i\}_{i=1}^{t_k}$. Let

$$A_k = \{a_i\}_{i=1}^{t_k} \tag{7}$$

be t_k distinct points in A. Define E and D as follows

$$\begin{aligned} E(x) &: \mathbb{R}^n \to \mathbb{R}^m, E(x) = a_i \text{ if } x \in C_i \text{ and } E(x) = 0 \text{ otherwise.} \\ D(y) &: \mathbb{R}^m \to \mathbb{R}^n, D(y) = c_i \text{ if } y \in A \text{ and } D(y) = 0 \text{ otherwise,} \end{aligned} \tag{8}$$

where c_i is the center of C_i and i is determined as follows: a_i is the point in A_k, which is nearest to y. It is clear that $E(x)$ is a constant function over each C_i and $D(x)$ is a constant function over each Voronoi polyhedron cell generated by A_k inside A.

We now prove that E and D satisfy the properties of the lemma. From the above construction, we have $V_1 = \cup_{i=1}^{t_k} C_i$ and $V(V_1) = V(S_k)$. Then property 1 follows from (6). It is easy to see $V_2 = A$. Then $\frac{V(V_2)}{V(A)} = 1$, and property 2 is proved. For $C_i \in S_k$, if $x \in C_i$, then $D(E(x))$ is the center of C_i, and hence $||D(E(x)) - x|| < \frac{\sqrt{m}}{k} < \gamma$ by (6). Then $V_3 = V_1$ and the lemma is proved. □

From the proof of Lemma 5.1, we have

Corollary 5.1. *Use the notations in Lemma 5.1. There exists a number t such that*

1. $V_1 = V_3 = \cup_{i=1}^{t} C_i$*, where $C_i \subset S$ are disjoint open cubes with center c_i.*

2. *Let $A_t = \{a_i\}_{i=1}^t$ be t distinct points inside A and $R_i, i = 1, \ldots, t$ the Voronoi polyhedron cells generated by A_t inside A and $a_i \in R_i$. Then $V_2 = A = \cup_{i=1}^t R_i$.*

3. $E(x) = a_i$ *for $x \in C_i$ and $E(x) = 0$ otherwise. $D(y) = c_i$ for $y \in R_i$ and $D(y) = 0$ otherwise.*

For a set $W \subset \mathbb{R}^n$ and $a \in \mathbb{R}_{>0}$, define $W^a = \{x \in W \,|\, \exists r > a, \text{ s.t. } \mathbb{B}(x, r) \subset W\}$.

Lemma 5.2. *Let $F : \mathbb{I}_0^n \to \mathbb{R}^m$ be a piece wise linear function with a finite number of linear regions. Then for any $\epsilon > 0$, $\gamma > 0$, $\alpha > 0$, there exists a DNN $\mathcal{F} : \mathbb{I}_0^n \to \mathbb{R}^m$ such that $|\mathcal{F}(x) - F(x)| < \epsilon$ for $x \in A^\gamma$, where A is any linear region of F. Moreover, $D = \{x \,|\, \mathcal{F}(x) - F(x)| > \epsilon\}$ satisfies $V(D) < \alpha$.*

Proof. It is easy to see that there exists a continuous function H which satisfies that $|F(x) - H(x)| = 0$ for $x \in A_\gamma$. Then the lemma follows from the universal approximation theorem of DNN [12, 15, 19]. □

We now prove the existence of the network defined in section 2. When r is small enough, all images in $\mathbb{B}(x, r)$ can be considered to have the same label with x. Therefore, the object $\mathbb{O}$ to be classified can be considered as a bounded open set in $\mathbb{R}^n$, and this motivates the existence results given below. We restate Theorem 2.1 below.

Theorem 5.1. *Suppose that the objects to be classified are a finite number of disjoint bounded open sets $S_l \subset \mathbb{I}_0^n$ with labels $l \in \mathbb{L} = \{0, \ldots, o\}$. Then, for any $m \in \mathbb{N}_{>0}$ and for any $\epsilon > 0$ and $\gamma > 0$, there exist DNNs $\mathcal{D} : \mathbb{R}^m \to \mathbb{R}^n$ and $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^m$ such that*

1. *Let $K = \{x \in \bigcup_{i=0}^o S_i \,|\, \mathcal{E}(x) \in \mathcal{E}(S_i) \cap \mathcal{E}(S_j)$ for some $i \neq j\}$. Then $\frac{V(K)}{\sum_i V(S_i)} < \epsilon$.*

2. *Let $\widehat{S}_l = \{x \in S_l \,|\, ||\mathcal{D}(\mathcal{E}(x)) - x|| < \gamma\}$ for $l \in \mathbb{L}$. Then $\frac{V(\widehat{S}_l)}{V(S_l)} > 1 - \epsilon$.*

Proof. Let $A_0 = \mathbb{I}_0^m$ and $A_j = \{y \,|\, y = z + 2j\mathbf{1}, z \in A_{j-1}\}$ for $j = 1, \ldots, o$, where $\mathbf{1} \in \mathbb{R}^m$ is the vector whose coordinates are 1. We first assume that each S_l is connected. By Lemma 5.1 and Corollary 5.1, for each $l \in \mathbb{L}$ and $\epsilon_1, \gamma_1 \in \mathbb{R}_{>0}$, there exist functions $E_l : \mathbb{R}^n \to \mathbb{R}^m$ and $D_l : \mathbb{R}^m \to \mathbb{R}^n$, which are sectional-continuous functions with a finite number of continuous regions and satisfy

(A1) $V_{l,1} = \cup_{i=1}^t C_{l,i}$, where $C_{l,i} \subset S_l$ are disjoint open cubes with center $c_{l,i}$. Furthermore, $\frac{V(V_{l,1})}{V(S_l)} > 1 - \epsilon_1$ and $||x - D_l(E_l(x))|| < \gamma_1$ for $x \in V_{l,1}$.

(A2) $U_l = \{a_{l,i}\}_{i=1}^t$ is a set of t distinct points inside A_l and $R_{l,i}, i = 1, \ldots, t$ the Voronoi polyhedron cells generated by U_l inside A_l and $a_{l,i} \in R_{l,i}$. Therefore, $A_l = \cup_{i=0}^o R_{l,i}$.

(A3) $E_l(x)$ and $D_l(y)$ are defined as follows: $E_l(x) = a_{l,i}$ for $x \in C_{l,i}$ and $E_l(x) = 0$ otherwise; $D_l(y) = c_{l,i}$ for $y \in R_{l,i}$ and $D_l(y) = 0$ otherwise.

Define E and D as follows: $E(x) = \sum_{i=0}^o E_i(x)$ and $D(y) = \sum_{i=0}^o D_i(y)$. Note that E is constant over $C_{l,i}$ and D is constant over $R_{l,j}$. We call $C_{l,i}$ and $R_{l,j}$ constant regions of E and D, respectively.

By Lemma 5.2, for any ϵ_2, ϵ_3, and β, there exist DNNs $\mathcal{E}$ and $\mathcal{D}$ such that

(B1) $||\mathcal{E}(x) - E(x)|| < \epsilon_2$, for $x \in \mathbb{I}_0^n \setminus S_{-1}$, where $S_{-1} \subset \mathbb{I}_0^n$, $V(S_{-1}) < \epsilon_3$, and $C_{l,i}^{\beta/4} \bigcap S_{-1} = \emptyset$ for any constant region $C_{l,i}$ of E.

(B2) $||\mathcal{D}(y) - D(y)|| < \epsilon_2$, for $y \in A \setminus S_{-2}$, where $A = \cup_{i=0}^o A_i$, $S_{-2} \subset A$, $V(S_{-2}) < \epsilon_3$, and $R_{l,i}^{\beta/4} \bigcap S_{-2} = \emptyset$ for any constant region $R_{l,i}$ of D.

Let β_l be the minimal of the distances between any pair of points in $E(S_l) = U_l = \{a_{l,i}\}_{i=1}^t$ and the distances between any point in $E(S_l)$ and the surface of A_l. Choose the parameters such that

(C0) $\beta < \beta_l$ for all $l \in \mathbb{L}$.

(C1) $\epsilon_2 \leq \beta/4$ and $\epsilon_2 \leq 1/2$.

(C2) $\epsilon_3 < \epsilon \sum_l V(S_l)$.

(C3) $\gamma_1 + \epsilon_2 < \gamma$, $\epsilon_1 + \frac{\epsilon_3}{V(S_l)} < \epsilon$ for all l.

We now prove that $\mathcal{E}$ and $\mathcal{D}$ satisfy the properties in the theorem. Let $x \in K$. Then $\mathcal{E}(x) \in \mathcal{E}(S_i) \cap \mathcal{E}(S_j)$ for $i \neq j$. By property (B1), if $x \in S_i/S_{-1}$ then $\mathcal{E}(x) \in D_i = (-\epsilon_2, 1+\epsilon_2)^m + 2i\mathbf{1}$. Similarly, if $x \in S_j/S_{-1}$ then $\mathcal{E}(x) \in D_j = (-\epsilon_2, 1+\epsilon_2)^m + 2j\mathbf{1}$. By (C1), we have $D_i \bigcap D_j = \emptyset$ if $i \neq j$. So $x \in K$ implies $x \in S_{-1}$, and hence $V(K) < \epsilon_3$ by (B1). Because of (C2), we have $\frac{V(K)}{\sum_l V(S_l)} < \epsilon$. The first property of the theorem is proved.

Let $x \in C_{l,i}/S_{-1}$. We will prove

$$||\mathcal{D}(\mathcal{E}(x)) - x|| \leq ||D(\mathcal{E}(x)) - x|| + \epsilon_2 \leq \gamma_1 + \epsilon_2 \leq \gamma. \tag{9}$$

The first inequality in (9) follows from $||\mathcal{D}(\mathcal{E}(x)) - D(\mathcal{E}(x))|| < \epsilon_2$ which will be proved below. By (A3), $E(x) = a_{l,i} \in R_{l,i}$ for $x \in C_{l,i}$. For any $x_0 \in \mathbb{B}(E(x), \beta/2)$, $E(x)$ is the nearest point of x_0 in $E(S_l)$ by (C0), and hence $\mathbb{B}(E(x), \beta/2) \subset R_{l,i}$ is a constant region of $D(y)$ by (A2). By (C1), $\mathbb{B}(E(x), \epsilon_2) \subset \mathbb{B}(E(x), \beta/2) \subset R_{l,i}$ and hence $\mathbb{B}(E(x), \epsilon_2) \subset R_{l,i}^{\beta/4}$. By (B1), $\mathcal{E}(x) \in \mathbb{B}(E(x), \epsilon_2)$ since $x \in C_{l,i}/S_{-1}$, hence $\mathcal{E}(x) \in R_{l,i}^{\beta/4}$. By (B2), $\mathcal{E}(x) \notin S_{-2}$ and hence $||\mathcal{D}(\mathcal{E}(x)) - D(\mathcal{E}(x))|| < \epsilon_2$. The first inequality in (9) is proved.

We now prove the second inequality in (9). We already proved $\mathcal{E}(x) \in \mathbb{B}(E(x), \epsilon_2) \subset R_{l,i}$ and hence $D(\mathcal{E}(x)) = D(E(x))$ by (A3). By (A1), $||D(\mathcal{E}(x)) - x|| = ||D(E(x)) - x|| < \gamma_1$ since $x \in C_{l,i} \subset V_{l,1}$. The last inequality in (9) follows from (C3).

By (9), $V_{l,1}/S_{-1} \subset \widehat{S}_l$. Finally, by (B1), (A1), and (C3), we have, $\frac{V(\widehat{S}_l)}{V(S_l)} \geq \frac{V(V_{l,1}/S_{-1})}{V(S_l)} > \frac{V(V_{l,1}) - \epsilon_3}{V(S_l)} > (1-\epsilon_1) - \frac{\epsilon_3}{V(S_l)} > 1 - \epsilon$. The theorem is proved.

In the above proof, each S_l is assumed to be connected. It is clear that the proof can be easily modified for the case such that different S_l have the same label. □

6 Conclusion

In this paper, we consider the problem of building robust DNNs to defend outliers and adversaries. In the open-world classification problem, the inputs to the DNN could be outliers which are irrelevant to the training set. On the contrary, the objects to be classified usually consist of a low-dimensional subspace of the total input space to the DNN and the majority of the input are outliers. So, the DNN need to recognize outliers in order to be used in open-world applications. A more subtle robust problem is that, for almost all DNNs with moderate complex structures, there exist adversary samples. Ability to defend adversaries is a necessary condition for the DNN to be used in safety-critical applications.

In this paper, we present a new neural network structure which is more robust to recognize outliers and to defend adversaries. The basic idea is to change the autoencoder from an un-supervised learning method to a classifier, where the encoder maps images with different labels into disjoint compression subspaces and the decoder recovers the image from its compression subspace. The newly introduced classification-autoencoder can recognize almost all outliers due to the fact that the output of the autoencoder is always similar the objects to be classified.

Since adversaries are seemly inevitable for the current DNNs, we introduce the list-classification based on the classification-autoencoder, which outputs several labels instead of one. With the list-classification, the outliers and the "nice" classified objects are firstly recognized, and for the adversaries and "fuzzy" objects, we output several labels and the corresponding images generated by the network. According to our experiments, the list-classification can give near lossless classification. The multi-output of the list classification could be used for further analysis, for example by other methods or even by a person.

Appendix. Structure of the network

We give the structure of the networks used in the experiments in sections 3 and 4.

The structure of $\mathcal{E}$ in (1):

Input layer: $N \times 1 \times 28 \times 28$, where N is steps of training.

Hidden layer 1: a convolution layer with kernel $1 \times 10 \times 3 \times 3$ with padding$= 1 \to$ do a batch normalization $\to$ do Relu $\to$ use max pooling with step=2.

Hidden layer 2: a convolution layer with kernel $10 \times 28 \times 3 \times 3$ with padding$= 1 \to$ do a batch normalization $\to$ do Relu $\to$ use max pooling with step=2.

Hidden layer 3: a convolution layer with kernel $28 \times 28 \times 3 \times 3$ with padding$= 1 \to$ do a batch normalization $\to$ do Relu $\to$ use max pooling with step=2.

Hidden layer 4: draw the output as $N \times 252 \to$ use a full connection with output size $N \times 168 \to$ do Relu.

Output layer: a full connection layer with output size $N \times 100 \to$ do Relu.

The structure of $\mathcal{D}$ in (1):

Input layer: $N \times 100$

Hidden layer 1: a full connection layer with output size $N \times 252 \to$ do Relu.

Hidden layer 2: a full connection layer with output size $N \times 600 \to$ do Relu.

Hidden layer 3: a full connection layer with output size $N \times 2700 \to$ do Relu.

Hidden layer 4: draw the output as $N \times 3 \times 30 \times 30 \to$ use convolution with kernel $3 \times 28 \times 3 \times 3$ with padding$= 0 \to$ do a batch normalization $\to$ do Relu.

Output layer: use convolution with kernel $28 \times 1 \times 1 \times 1$ with padding$= 0 \to$ do Relu.

The structure of the CNN is given below, which is used to train the min-max model in section 3.3 and to create the strong adversarial samples in section 3.3. The network $\mathcal{H}$ in section 3.2 is similar to this network, whose output layer is a full connection layer with output size $N \times 11$.

Input layer: $N \times 1 \times 28 \times 28$.

Hidden layer 1: a convolution layer with kernel $1 \times 32 \times 3 \times 3$ with padding$= 1 \to$ do Relu $\to$ use max pooling with step=2.

Hidden layer 2: a convolution layer with kernel $32 \times 64 \times 3 \times 3$ with padding$= 1 \to$ do Relu $\to$ use max pooling with step=2.

Hidden layer 3: a convolution layer with kernel $64 \times 64 \times 3 \times 3$ with padding$= 1 \to$ do Relu $\to$ use max pooling with step=2.

Hidden layer 4: draw the output as $N \times 576 \to$ use a full connection layer with output size $N \times 128 \to$ do Relu.

Output layer: a full connection layer with output size $N \times 10$.

References

[1] N. Akhtar and A. Mian, Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey, arXiv:1801.00553v3, 2018.

[2] A. Azulay and Y. Weiss, Why Do Deep Convolutional Networks Generalize so Poorly to Small Image Transformations? *Journal of Machine Learning Research*, 20, 1-25, 2019.

[3] T. Bai, J. Luo, J. Zhao, Recent Advances in Understanding Adversarial Robustness of Deep Neural Networks. arXiv:2011.01539, 2020.

[4] D.H. Ballard, Modular Learning in Neural Networks, *Proc. AAAI'87*, Vol. 1, 279-284, AAAI Press, 1987.

[5] R. Chalapathy and S. Chawla, Deep Learning for Anomaly Detection: A Survey. arXiv:1901.03407v2, 2019.

[6] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval Networks: Improving Robustness to Adversarial Examples, *Proc. ICML'2017*, 854-863, 2017.

[7] G. Cybenko, Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals and Systems, 2(4): 303-314, 1989.

[8] I.J. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.

[9] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and Harnessing Adversarial Examples. arXiv:1412.6572, 2014.

[10] M. Hein, M. Andriushchenko, Formal Guarantees on the Robustness of a Classifier Against Adversarial Manipulation. *Proc. NIPS*, 2266-2276, 2017.

[11] G. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network. arXiv:1503.02531, 2015.

[12] K. Hornik, Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4(2): 251-257, 1991.

[13] Y. LeCun, Y. Bengio, G. Hinton, Deep Learning, *Nature*, 521(7553), 436-444, 2015.

[14] N. Lei, D. An, Y. Guo, K. Su, S. Liu, Z. Luo, Z. Gu, A Geometric Understanding of Deep Learning. *Engineering*, 6(3), 361-374, 2020.

[15] M. Leshno, V.Ya. Lin, A. Pinkus, and S. Schocken, Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate any Function. *Neural Networks*, 6(6): 861-867, 1993.

[16] W. Lin, Z. Yang, X. Chen, Q, Zhao, X. Li, Z. Liu, J. He, Robustness Verification of Classification Deep Neural Networks via Linear Programming. *CVPR'2019*, 11418-11427, 2019.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083, 2017.

[18] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The Limitations of Deep Learning in Adversarial Settings. In 2016 IEEE European Symposium on Security and Privacy, 372-387, IEEE Press, 2016.

[19] A. Pinkus, Approximation Theory of the MLP Model in Neural Networks. *Acta Numerica*, 8: 143-195, 1999.

[20] Y. Qi, Y. Wang, X. Zheng, Z. Wu, Robust Feature Learning by Stacked Autoencoder with Maximum Correntropy Criterion, *ICASSP'2014*, 6716-6720, IEEE Press, 2014.

[21] A. Raghunathan, J. Steinhardt, P. Liang, Certified Defenses Against Adversarial Examples. ArXiv: 1801.09344, 2018.

[22] A. Shafahi, W.R. Huang, C. Studer, S. Feizi, T. Goldstein, Are Adversarial Examples Inevitable? arXiv:1809.02104, 2018.

[23] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L.S. Davis, G. Taylor, T. Goldstein, Adversarial Training for Free! ArXiv: 1904.12843, 2019.

[24] W.J. Scheirer, A. Rocha, A. Sapkota, T.E. Boult, Towards Open Set Recognition. *IEEE T. PAMI*, 36(7):1757-1772, 2013.

[25] M.H. Stone, The Generalized Weierstrass Approximation Theorem. *Mathematics Magazine*, 21(4): 167-184, 1948.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I.J. Goodfellow, R. Fergus, Intriguing Properties of Neural Networks. arXiv:1312.6199, 2013.

[27] F. Tramer, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble Adversarial Training: Attacks and Defenses. ArXiv: 1705.07204, 2017.

[28] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders. *Proc. ICML'08*, 1096-1103, ACM Press, 2008.

[29] E. Wong, J. Z. Kolter, Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope. ArXiv: 1711.00851, 2017.

[30] H. Xu, Y. Ma, H.C. Liu, D, Deb, H. Liu J.L. Tang, A.K. Jain, Adversarial Attacks and Defenses in Images, Graphs and Text: A Review, *International Journal of Automation and Computing*, 17(2), 151-178, 2020.

[31] Z. Yang, X. Wang, Y. Zheng, Sparse Deep Neural Networks Using $L_{1,\infty}$-Weight Normalization, Statistica Sinica, 2020, doi:10.5705/ss.202018.0468.

[32] L. Yu and X.S. Gao, Improve the Robustness and Accuracy of Deep Neural Network with $L_{2,\infty}$ Normalization. arXiv:2010.04912.

[33] D.H. Zhang, T.Y. Zhang, Y.P. Lu, Z.X. Zhu, B. Dong, You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle. ArXiv: 1905.00877, 2019.

[34] X.Y. Zhang, C.L. Liu, C.Y. Suen, Towards Robust Pattern Recognition: A Review. *Proc. of the IEEE*, 108(6), 894-922, 2020.

[35] C. Zhou and R.C. Paffenroth, Anomaly Detection with Robust Deep Autoencoders. *KDD'17*, 665-674, ACM Press, 2017.