Code-Bridged Classifier (CBC): A Low or Negative Overhead Defense for Making a CNN Classifier Robust Against Adversarial Attacks

Farnaz Behnia, Ali Mirzaeian, Mohammad Sabokrou, Saj Manoj, Tinoosh Mohsenin,
Khaled N. Khasawneh, Liang Zhao, Houman Homayoun, Avesta Sasan
fbehnia@gmu.edu, amirzaei@gmu.edu, sabokro@ipm.ir, spudukot@gmu.edu, tinoosh@umbc.edu,
kkhasawn@gmu.edu, lzhao9@gmu.edu, hhomayoun@ucdavis.edu, asasan@gmu.edu

***Abstract*—In this paper, we propose Code-Bridged Classifier (CBC), a framework for making a Convolutional Neural Network (CNNs) robust against adversarial attacks without increasing or even by decreasing the overall models' computational complexity. More specifically, we propose a stacked encoder-convolutional model, in which the input image is first encoded by the encoder module of a denoising auto-encoder, and then the resulting latent representation (without being decoded) is fed to a reduced complexity CNN for image classification. We illustrate that this network not only is more robust to adversarial examples but also has a significantly lower computational complexity when compared to the prior art defenses.**

I. Introduction

Deep learning is the foundation for many of today's applications, such as computer vision, natural language processing, and speech recognition. After AlexNet [1] made a breakthrough in 2012 by significantly outperforming other object detection solutions, and winning the ISLVRC competition [2], CNNs gained a well-deserved popularity for computer vision applications. This energized the research community to architect models capable of achieving higher accuracy (that led to development of many higher accuracy models including GoogleNet [3] and ResNet [4]), increased the demand and research for hardware platforms capable of fast execution of these models [5, 6], and created a demand for lower complexity models [7–9] capable of reaching high levels of accuracy.

Even though the evolution in their model structure and the improvement in their accuracy have been very promising in recent years, it is illustrated that convolutional neural networks are prone to adversarial attacks through simple perturbation of their input images [10–13]. The algorithms proposed by [10–13] have demonstrated how easily the normal images can be perturbed with adding a small noise in order to fool neural networks. The main idea is to add a noise vector containing small values to the original image in the opposite or same direction of the gradient calculated by the target network to produce adversarial samples [10, 11].

The wide-spread adoption of CNNs in various applications and their unresolved vulnerability to adversarial samples has raised many safety and security concerns and has motivated a new wave of deep learning research. To defend against adversarial attacks, the concept of adversarial training was proposed in [10] and was further refined and explored in [11, 12]. Adversarial training is a data augmentation technique in which by generating a large number of adversarial samples and including them with correct labels in the training set, the robustness of network against adversarial attacks improves. Training an adversarial classifier to determine if the input is normal or adversarial and using autoencoder (AE) to remove the input image noise before classification are some of the other approaches taken by [10] and [14]. Finally, [15] utilizes distillation as a defense method against adversarial attacks in which a network with a similar size to the original network is trained in a way that it hides the gradients between the softmax layer and its predecessor.

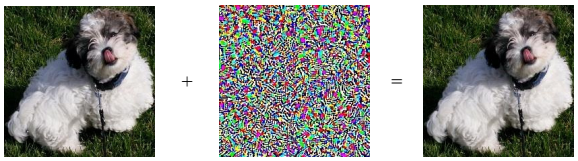

Fig. 1. The FGSM attack is used to add adversarial noise to the original image. The adversarial perturbation remains imperceptible to the human eyes but causes the neural network to misclassify the input image.

In this work, we combine denoising and classification into a single solution and propose the code-bridged classifier (CBC). We illustrate that CBC is 1) more robust against adversarial attacks compared to a similar CNN solution that is protected by a denoising AE, and has substantially less computational complexity compared to such models.

II. Background and Related Work

The vulnerability of deep neural networks to adversarial examples was first investigated in [14]. Since this early work, many new algorithms for generating adversarial examples, and a verity of solutions for defending against these attacks are proposed. Following is a summary of the attack and defense models related to our proposed solution:

A. Attack Models

Many effective attacks have been introduced in the literature. Some of the most notable attacks include Fast Gradient Sign Method (fgsm) [10], Basic Iterative Method [11], Momentum Iterative Method [12], DeepFool [13] and Carlini Wagner [16], the description of each method are as follows.

1) FGSM attack: in [10], a simple method is suggested to add a small perturbation to the input to make an adversarial image. The adversarial image is obtained by:

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

in which x is the input image, y is the correct label, θ is the network parameters, and J is the loss function. ϵ defines the magnitude of the noise. The larger the ϵ, the larger the possibility of misclassification. Figure 1 illustrates how such adversarial perturbation can change the classifier's prediction.

2) Basic Iterative Method (BIM) attack [11]: Also known as Iterative-FGSM attack, BIM attack is iterating over the FGSM attack, increasing the effectiveness of the attack. The BIM attack can be expressed as:

$$x'_0 = x,\ x'_n = x'_{n-1} + \epsilon sign(\nabla_x J(\theta, x_{n-1}, y)) \quad (2)$$

3) Momentum Iterative attack [12]: In the Momentum Iterative attack, the momentum is also considered when calculating the adversary perturbation, and is expressed as:

$$g_0 = 0,\ g_n = \mu g_{n-1} \frac{\nabla_x J(\theta, x_{n-1}, y)}{||\nabla_x J(\theta, x_{n-1}, y)||_1}$$
$$x'_n = x'_{n-1} + \epsilon sign(g_n) \quad (3)$$

in which μ is the momentum, and $||\nabla_x J(\theta, x_{n-1}, y)||_1$ is the L_1 norm of the gradient.

4) Deepfool [13]: The Deepfool attack is formulated such that it can find adversarial examples that are more similar to the original ones. It assumes that neural networks are completely linear and classes are distinctively separated by hyper-planes. With these assumptions, it suggests an optimal solution to find adversarial examples. However, because neural networks are nonlinear, the step for finding the solution is repeated. We refer to [13] for details of the algorithm.

5) Carlini & Wagner (CW) [16]: Finding adversarial examples in the CW attack is an iterative process that is conducted against multiple defense strategies. The CW attack uses Adam optimizer and a specific loss function to find adversarial examples that are less distorted than other attacks. For this reason, the CW attack is much slower. Adversarial examples can be generated by employing L_0 , L_2 and L_∞ norms. The objective function in CW attack consider an auxiliary variable w and is defined as:

$$\delta_i = \frac{1}{2}(tanh(w_i) + 1) - x_i \quad (4)$$

Then if we consider the L_2 norm, this perturbation is optimized with respect to w:

$$min_w ||\delta||_2^2 + c.f(\delta + x) \quad (5)$$

in which function f is defined as follows:

$$f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa) \quad (6)$$

in the above equation, $Z(x')$ is the pre-softmax output for class i, the parameter t represents the target class, and κ is the parameter for controlling the confidence of misclassification.

B. Transferability of Adversarial Examples

All previously described attacks are carried out in a white-box setting in which the attacker knows the architecture, hyperparameters, and trained weights of the target classifier as well as the existing defense mechanism (if any). It is very hard to defend against white-box attacks because the attacker can always use the information she has to produce new and working adversarial inputs. However, adversarial attacks can be considered in two other settings: Gray Box and Black Box attacks. In gray box attacks, the attacker knows the architecture but doesn't have access to the parameters, the defense mechanism. In black-box setting the attacker does not know the architecture, the parameters, and the defense method.

Unfortunately, it has been shown that adversarial examples generalize well across different models. In [14] it was shown that many of the adversarial examples that are generated for (and are misclassified by) the original network are also misclassified on a different network that is trained from scratch with different hyperparameters or using disjoint training sets.

The findings of [14] are confirmed by the following works, as in [17], universal perturbations are successfully found that not only generalize across images but also generalize across deep neural networks. These perturbations can be added to all images and the generated adversarial example is transferable across different models. The work in [18, 19] show that adversarial examples that can cause a model to misclassify, can have the same influence on another model that is trained for the same task. Therefore, an attacker can train her dummy model to generate the same output, craft/generate adversarial images on her model, and rely on the transferability of the adversarial examples, being confident that there is a high chance for the target classifier to be fooled. We argue that our proposed solution can effectively defend black-box attacks.

C. Defenses

Several works have investigated defense mechanisms against adversarial attacks. In [10], adversarial training is proposed to enhance the robustness of the model. In [20, 21] autoencoders are employed to remove the adversarial perturbation and reconstruct a clean input. In [15] distillation is used to hide the gradients of the network from the attacker. Other approaches are also used as a defense mechanism [22–24]. In this section, we explore the ideas for defending against adversarial examples.

1) Adversarial Training: The basic idea of the adversarial training [10] is to train a robust classifier via adding many adversarial examples (that are generated using different attacks) to the training dataset [12, 13, 25]. The problem with this approach is that it can only make the network robust against known (and trained for) attacks for generating adversarial examples. It also increases the training time significantly.

2) Defensive Distillation: In [15] distilling was originally proposed to train a smaller student model from a larger teacher model with the objective that the smaller network predicts the probability of the bigger network. The distillation technique takes advantage of the fact that a probability vector contains more information than only class labels, hence, it is a more effective mean for training a smaller network. For defensive distillation, the second network is the same size as the first network [15]. The main idea is to hide the gradients between the pre-softmax and softmax layers to make the attacker's job more difficult. However, it was illustrated in [16] that this defense can be beaten by using the pre-softmax layer outputs in the attack algorithm and/or choosing a different loss function.

3) Gradient Regularization: Input gradient regularization was fist introduced by [26] to improve the generalization of training in neural networks by a double backpropagation method. [15] mentions the double backpropagation as a defense and [22] evaluate the effectiveness of this idea to train a more robust neural network. This approach intends to make

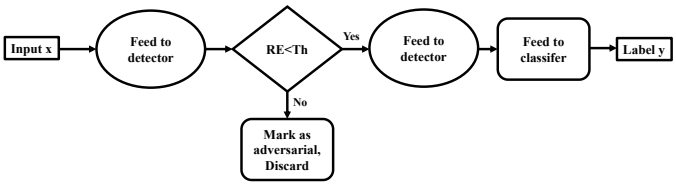

Fig. 2. Magnet defense in [21] is a two stage defense: the first stage tries to detect the adversarial examples. The images that pass the first stage are denoised using an AutoEncoder in the second stage and fed to the classifier.

sure that if there is a small change input, the change in KL divergence between the predictions and the labels also will be small. However, this approach is sub-optimal because of the blindness of the gradient regulation.

4) Adversarial Detection: Another approach taken to make neural networks more robust is to detect adversarial examples before feeding to the network[23, 27]. [23] tries to find a decision boundary to separate adversarial and clean inputs. [27] deploys the fact that the perturbation of pixel values by adversarial attack alters the dependence between pixels. By modeling the differences between adjacent pixels in natural images, deviations due to adversarial attacks can be detected.

5) Autoencoders: [20] analyzes the use of normal and denoising autoencoders as a defense method. Autoencoders are neural networks that code the input and then try to reconstruct the original image as their output. [21], as illustrated in Fig. 2, uses a two-level module and uses autoencoders to detect and reform adversarial images before feeding to the target classifier. However, this method may change the clean images and also add a computational overhead to the whole defense-classifier module. To improve the method introduced in [21], [28] presents an efficient auto-encoder with a new loss function which is learned to preserve the local neighborhood structure on the data manifold.

III. Problem Statement

An abstract view of a typical Auto-Encoder (AE) and Denoising Auto-Encoder (DAE) is depicted in Fig. 3. An AE is comprised of two main components: 1) The *encoder*, $\varphi(X)$, that extracts the corresponding latent space for input X, and 2) the *decoder*, $\zeta(\varphi(X))$, that reconstructs a representation of the input image from its compressed latest space representation. Ideally, the decoder can generate the exact inputs sample from the latent space, and the relation between the input and output of an AE can be expressed as $\zeta(\varphi(X)) = X$. However, in reality, the output of an AE is to some extent different from the input. This difference is known as reconstruction error and is defined as $E_R = |\zeta(\varphi(X)) - X|$ [29]. When training an AE, the objective is E_R.

A DAE is similar to AE, however, it is trained using a different training process. As illustrated in Fig. 3.b the input space of DAE are the noisy input samples, $X + \epsilon$, and their corresponding latent space is generated by $\varphi(X + \epsilon)$. Unlike AE (in which the E_R is defined as the difference between the input and output of AE), the E_R of DAE is defined as $E_R = |\zeta(\varphi(X+\epsilon)) - X|$ [29]. In other words, the reconstruction error is the difference between the output of decoder $\zeta(\varphi(X + \epsilon))$ and the clean input samples. An ideal DAE removes the noise ϵ from the noisy input and generates the clean sample X.

This refining property of DAEs, make them an appealing defense mechanism against adversarial examples. More precisely, by placing one or more DAEs at the input of a classifier, the added adversarial perturbations are removed and a refined input is fed into the subsequent classifier. The effectiveness of this approach highly depends on the extent of which the underlying DAE is close to an ideal DAE (in which the DAE completely refines the perturbed input). Although a well-trained DAE refines the perturbed input to some extent, it also imposes a reconstruction noises to it. As an example, assume that ϵ in Fig. 3.b is zero. This means the input X is a clean image. In this case the output is $X + E_R$. If the size of E_R is large enough, it can move the input X over the classifier's decision boundary. This, as illustrated in Fig. 4, will result in predicting the input X as a X^* class member. In this scenario, DAE not only fails to defend against adversarial examples, but also generates noise that could lead to the misclassification of the clean input images.

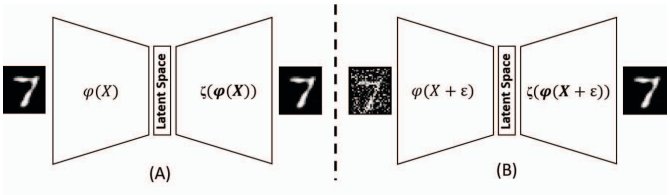

Fig. 3. An abstract view of a) a typical Auto-Encoder, and b) a Denoising Autoencoders. Two major components of both structures are 1) Encoder, $\varphi(.)$, which extracts the latent space of sample inputs 2) Decoder, $\zeta(.)$, which reconstructs sample inputs from the latent space.

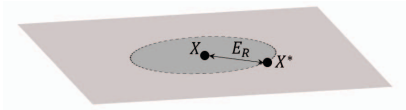

Fig. 4. Reconstruction error (E_R) of a decoder can also result in misclassification if the features extracted for the reconstructed image X^* are pushed outside of the classifier's learnt decision boundary.

The other problem of using AE or DAE as a pre-processing unit to refine the image and combat adversarial attacks is their added computational complexity. Adding an autoencoder as a pre-processor to a CNN increases 1) the energy consumed per classification, 2) the latency of each classification and 3the number of parameters of the overall model.

In the following section, we propose a novel solution for protecting the model against adversarial attacks that addresses both the computational complexity problem and the reconstruction error issue of using an AE as a pre-processor.

IV. Proposed Method

Using DAEs to refine perturbed input samples before feeding them into the classifier is a typical defense mechanism against adversarial examples [20, 21]. A general view of such defense is illustrated in Fig. 5.(top). In this figure, $\varphi(.), \zeta(.)$ are the decoder and encoder of DAE, respectively, $\varphi'(.)$ represents the first few CONV layers of the CNN classifier, and $C(.)$, represents the later CONV stages. In this defense, the DAE and CNN are separately trained. The DAE is trained to minimize the reconstruction error, while the CNN is trained to reduce the pre-determined loss function (e.g. $L1$ or $L2$ loss). An improved version of such defense is when the training is done serially, where in the first stage, the DAE is trained, and then the CNN classifier is trained using the output of DAE as input sample. Note that the 2nd solution tends to get

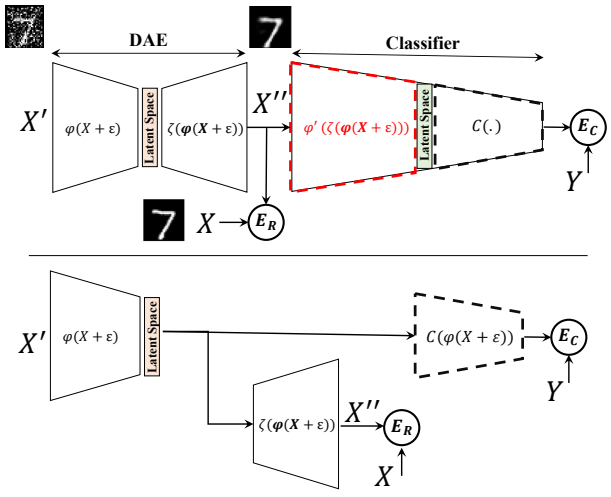

Fig. 5. (Top) the defense proposed in [21] where a DAE filter the noise in the input image before feeding it to a classifier. (Bottom) the CBC model in which the decoder of DAE and the first few conv layers of the base classifier are removed. Note that the decoder in CBC is only used for training the CBC, and is removed after training (for evaluation). In this figure X, X', X'' are respectively the clean input sample, noisy input sample and the output of DAE. The Y is the corresponding ground truth, and E_R and E_C are reconstruction error and classification error respectively.

a higher classification accuracy. Regardless of the choice of training addition of a DAE to the CNN classifier adds to its complexity. Aside from added computational complexity, the problem with this defense mechanism is that AEs could act as a double agent: on one hand refining the adversarial examples is an effective means to remove the adversarial perturbation (noise) from the input image and is a valid defense mechanism, but on the other hand, its reconstruction error, E_R, could force misclassification of clean input images. For correcting the behavior of the DAE, we propose the concept of Code Bridge Classifiers (CBC), aiming to 1) eliminating the impact of reconstruction error of the underlying DAE, and 2) reducing the computational complexity of the combined DAE and classifier to widen its applicability.

Fig. 5.(bottom), illustrates our proposed solution where the encoder $\varphi(.)$ of a trained DAE and a part of original CNN ($C(.)$) are combined to form a hybrid yet compressed model. In this model, the decoder $\zeta(.)$ of DAE, and the first few CONV layers of the CNN model, $\varphi'(.)$, are eliminated. In CBC $\zeta(.)$ and $\varphi'(.)$ are eliminated with the intuition that they act as an Auto Decoder (AD). As opposed to AE, the AD translates the latest space to an image and back to another latent space (intermediate representation of the image in the CNN captured by output channels of $\varphi'(.)$). This is, however, problematic because 1) the decoder $\zeta(.)$ is not ideal and it introduces reconstruction error to the refined image, 2) decoding and encoding (the first few CONV layers act as an encoder) of the image only translates the image from one latest space to another without adding any information to it. This is when such code translation (latest space to latent space) could be eliminated and the code at the output of $\varphi(.)$ could be directly used for classification. This allows us to eliminate the useless AD (the decoder $\zeta(.)$ and first few conv layers of original CNN that act as an encoder) and not only reduce the computational complexity the overall model but also improves the accuracy of the model by eliminating the noise related to image reconstruction of the decoder $\zeta(.)$.

TABLE I
ARCHITECTURE OF THE FASHIONMNIST CLASSIFIERS

	Base		DAE-CNN		CBC	
	Type	Size	Type	Size	Type	Size
Defense			Conv.ReLU	$4 \times 4 \times 16$	Conv.ReLU	$4 \times 4 \times 16$
			Conv.ReLU	$4 \times 4 \times 48$	Conv.ReLU	$4 \times 4 \times 48$
			ConvTran.ReLU	$4 \times 4 \times 48$		
			ConvTran.ReLU	$4 \times 4 \times 16$		
CNN	Conv.ReLU	$3 \times 3 \times 32$	Conv.ReLU	$3 \times 3 \times 32$	Conv.ReLU	$3 \times 3 \times 64$
	Conv.ReLU	$3 \times 3 \times 32$	Conv.ReLU	$3 \times 3 \times 32$	Conv.ReLU	$3 \times 3 \times 64$
	Max Pool	2×2	Max Pool	2×2	FC.ReLU	4096×200
	Conv.ReLU	$3 \times 3 \times 64$	Conv.ReLU	$3 \times 3 \times 64$	FC.ReLU	200×200
	Conv.ReLU	$3 \times 3 \times 64$	Conv.ReLU	$3 \times 3 \times 64$	Softmax	10
	FC.ReLU	4096×200	FC.ReLU	4096×200		
	FC.ReLU	200×200	FC.ReLU	200×200		
	Softmax	10	Softmax	10		

The training process for the CBC is serial: We first train a DAE, the encoder section of the model is separated. Then the trained decoder is paired with a smaller CNN compare to that of the original model. One way to build a smaller model is to remove the first few CONV layers of the original model and adjust the width of the DAE and the partial CNN to match the filter sizes. The rule of thumb for the elimination of the layers is to remove as many CONV layers equal to those in the encoder of AE. The next step is to train the partial CNN while fixing the values of the decoder, allowing the propagation to only alter the weights in the classifier $C(.)$.

V. IMPLEMENTATION DETAILS

In this section, we investigate the effectiveness of our proposed solution against adversarial examples prepared for FashionMNIST [30] and CIFAR-10 [31] datasets. To be able to compare our work with previous work in [21], we build our CBC solution on top of the CNN models that are described as *Base* in tables I and II. In these tables, the DAE columns represent the solution proposed in [21], in which a full auto-encoder pre-process the input to the CNN model and finally the columns CBC described the modified model corresponding to our proposed solution. The DAE as described in tables I and II includes 2 convolutional layers for encoding, and 2 convolutional transpose layers for decoding. The input is a clean image, and the output is an image of the same size generated by the autoencoder.

To build the CBC classifier, we stacked the trained encoder of the DAE with an altered version of the target classifier in which some of the CONV layers are removed. The trade-off on the number of layers to be removed is discussed in the next section. Considering that the encoder quickly reduces the size of the input image to a compressed latent space representation, the CNN following the latent space is not wide. For this reason, we also remove the max-pooling layers making sure that the number of parameters of the CBC classifier, when it reaches the softmax layer is equal to that of the base architecture. In our implementation, all the attacks and models are implemented using PyTorch [32] framework. To train the models we only use clean samples, and freeze the weights of the encoder part and train the remaining layers. Training parameters of the target classifier and the proposed architecture are listed in table III. We evaluated our proposed solutions against the FGSM [10], Iterative [11], DeepFool [13], and Carlini Wagner [16] adversarial attacks.

VI. EXPERIMENTAL RESULTS

By adopting the training flow described in Table IV, the top-1 accuracy of the base classifiers (in Tables I and II) that

TABLE II
ARCHITECTURE OF THE CIFAR-10 CLASSIFIERS

	Base		DAE-CNN		CBC	
	Type	Size	Type	Size	Type	Size
Defense			Conv.ReLU	$4 \times 4 \times 48$	Conv.ReLU	$4 \times 4 \times 48$
			Conv.ReLU	$4 \times 4 \times 72$	Conv.ReLU	$4 \times 4 \times 72$
			ConvTran.ReLU	$4 \times 4 \times 72$		
			ConvTran.ReLU	$4 \times 4 \times 48$		
CNN	Conv.ReLU	$3 \times 3 \times 96$	Conv.ReLU	$3 \times 3 \times 96$	Conv.ReLU	$3 \times 3 \times 96$
	Conv.ReLU	$3 \times 3 \times 96$	Conv.ReLU	$3 \times 3 \times 96$	Conv.ReLU	$3 \times 3 \times 192$
	Conv.ReLU	$3 \times 3 \times 96$	Conv.ReLU	$3 \times 3 \times 96$	Conv.ReLU	$3 \times 3 \times 192$
	Max Pool	2×2	Max Pool	2×2	Conv.ReLU	$3 \times 3 \times 192$
	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$3 \times 3 \times 192$
	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$3 \times 3 \times 192$
	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$1 \times 1 \times 192$
	Max Pool	2×2	Max Pool	2×2	Conv.ReLU	$1 \times 1 \times 192$
	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$3 \times 3 \times 192$	Conv.ReLU	$1 \times 1 \times 192$
	Conv.ReLU	$1 \times 1 \times 192$	Conv.ReLU	$1 \times 1 \times 192$	Avg Pool	
	Conv.ReLU	$1 \times 1 \times 192$	Conv.ReLU	$1 \times 1 \times 192$	Softmax	10
	Avg Pool		Avg Pool			
	Softmax	10	Softmax	10		

TABLE III
TRAINING PARAMETERS

Dataset	Optimization Method	Learning Rate	Batch Size	Epochs
FashionMNIST	Adam	0.001	128	50
CIFAR-10	Adam	0.0001	128	150

we trained for FashionMNIST and CIFAR10 are 95.1% 90.% respectively. For the evaluation purpose, we trained denoising autoencoders with different noise values for both Datasets. The structure of the DAEs are shown in Table I and II. The reconstruction error for DAEs was arround 0.24 and 0.54 for FashionMNIST and CIFAR10 datasets respectively.

A. Selecting altered CNN architecture:

As discussed previously, the removal of the decoder of the DAE should be paired with removing the first few CONV layers from the base CNN and training the proceeding CONV layers to use the code (latent space) that is generated by the encoder as input. The number of layers to be removed was determined by a sweeping experiment in which the accuracy of the resulting model and its robustness against various attacks was assessed. Figure 6 shows the accuracy of CBC networks when the number of convolutional layers in the altered classifier is reduced compared to the base classifier. The experiment is repeated for both MNIST and CIFAR datasets, and the robustness of each model against CW [16], Deepfool [13], and FGSM [10] with $\epsilon = 0.5$ is assessed. As illustrated in Fig. 6, the models remain insensitive to the removal of some of the first few layers (2 in MNIST, and 5 in CIFAR) with negligible ($\tilde{1}\%$) change in the accuracy by complete removal of each CONV layer until they reach a tipping point. The MNIST model, for being a smaller model, reaches that tipping point when 2 CONV layers are removed, whereas the CIFAR model (for being a larger model) is slightly impacted even after 5 CONV layers are removed.

B. CBC accuracy and comparison with prior art

Fig. 7 captures the result of our simulation, in which the robustness and the accuracy of the base CNN, the solution in [21] in which the DAE refines the input for base CNN model, and our proposed CBC are compared. For the CNN protected with DAE, we provide two sets of results: 1) DAE-CNN Model accuracy: DAE and CNN are separately trained and paired together; 2) Retrained-DAC-CNN model accuracy: The CNN is incrementally trained using the refined images produced at the output of the DAE (denoted by Retraind-DAE-CNN). The comparison is done for the classification of both original and adversarial images. Results for FGSM, DeepFool, and CW adversarial attacks are reported. For completeness, we

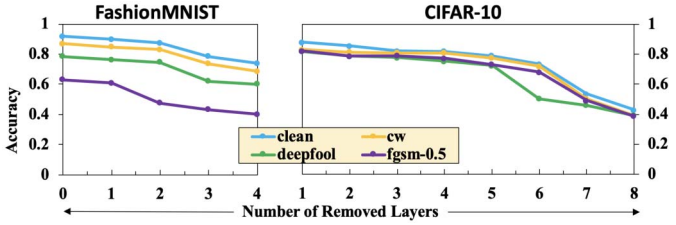

Fig. 6. The change in the accuracy of the CBC model for a) FashionMNIST and b) CIFAR-10 classification with respect to the number of removed CONV layers from the base CNN model.

have captured the robustness of each solution when the DAE is trained with different noise values.

As illustrated in Fig. 7, the base model is very sensitive to adversarial examples, and the accuracy of the network in presence of adversarial examples (depending on the attack type) drops from over 90% to the range of 0% to 20%. The DAE-CNN model also performs very poorly even for benign images. This is because of the reconstruction error introduced by the decoder which severely affects the accuracy and the ability of the base CNN model. The Retrained-DAE-CNN model (representing the solution in [21]) performs well in classifying benign images and also exhibits robustness against adversarial images. As illustrated, the robustness improves when it is paired with a DAE that is trained with high noise. The best solution, however, is the CBC solution: regardless of the type of the attack, type of the benchmark, and the noise of DAE, the CBC model outperforms other solutions in both classification accuracy of the benign images and also robustness against adversarial examples. This clearly illustrates that the CBC model by eliminating the reconstruction error is a far more robust solution than DAE protected CNN models.

C. Reduction in model size and computational complexity

In a CBC model, the DAE's decoder and the first few CONV layers of the base CNN model are removed. Hence, a CBC model has a significantly smaller flop count (computational complexity). Table IV captures the number of model Parameters and the Flop count for each of the CBC classifiers which are described in Tables I and II. Note that the majority of computation in a CNN model is related to its CONV layers, while a CONV layer has a small number of parameters. Hence, removing a few CONV layers may result in a small reduction in the number of parameters, but the reduction in the FLOP count of the CBC models is quite significant. As reported in Table IV, in the FashionMNIST model, the flop count has reduced by 1.8x and 2.8X compared to the base and DAE protected model, while the parameter count is respectively reduced by 0.37% and 2.69% . This saving is more significant for the CIFAR-10 CBC model, where its computational complexity has reduced 3.1x and 3.3x compared to the Base and DAE protected model respectively, while the number of parameters is respectively reduced by 5.8% and 13.4%. Reduction in the flop count of the CBC model, as illustrated in table IV also reduces the model's execution time. The execution time reported in table IV is the execution time of each model over the validation set of each (FashionMNIST and CIFAR-10) dataset when the model is executed using Dell PowerEdge R720 with Intel Xeon E5-2670 (16 core CPUs) processors. As reported in table IV, the execution time of the

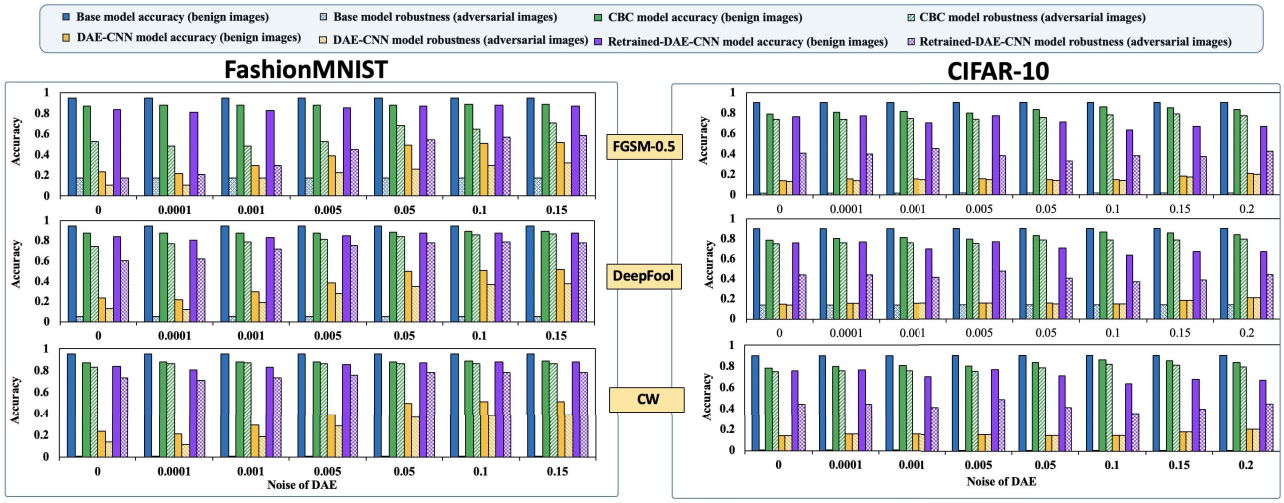

Fig. 7. Comparing the accuracy of the base CNN model, the DAE protected CNN model (with and without retraining), and the CBC model when classifying bening images and adverserial images generated by different attack models: (left): FashionMNIST models, (right): CIFAR-10 models.

TABLE IV
OF THE NUMBER OF PARAMETERS, COMPUTATIONAL COMPLEXITY AND EXECUTION TIME OF CBC AND THE BASE MODEL WITH AE AND WITHOUT AE PROTECTION.

Dataset	Model	Flops	Parameters	Execution time
FashionMNIST	Base CNN	9.08 MMac	926.6 K	463.4 s
	AE-CNN[21]	14.3 MMac	951.81 K	562.3 s
	CBC	5.04 MMac	926.25 K	293.7s
CIFAR-10	Base CNN	0.59 GMac	1.37 M	1673.0 s
	AE-CNN [21]	0.63 GMac	1.49 M	1749.7 s
	CBC	0.19 GMac	1.29 M	1191.6 s

CBC is even less than the base CNN. Note that the CBC also results in processing unit energy reduction proportional to the reduction in the flop count. Hence, the CBC, not only resist against adversarial attacks, but (for being significantly smaller than the base model) also reduces the execution time, and energy consumed for classification.

VII. Conclusion

In this paper, we propose the Code-Bridged Classifier (CBC) as a novel and extremely efficient mean of defense against adversarial learning attacks. The resiliency and complexity reduction of CBC is the result of directly using the code generated by the encoder of a DAE for classification. For this purpose, at the training phase, a decoder is instantiated in parallel with the model to tune the denoising encoder by computing and back-propagating the image reconstruction error. At the same time, the code is used for classification using a lightweight classifier. Hence, the encoder is trained for both feature extraction (contributing to the depth of classifier and low-level feature extraction) and denoising. The parallel decoder is then removed when the model is fully trained. This allows the CBC to achieve high accuracy by avoiding the reconstruction error of the DAE's decoder, while reducing the computational complexity of the overall model by eliminating the decoder and few CONV layers from the trained model.

References

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012.
[2] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *proc. of the IEEE conf. on computer vision and pattern recognition*, 2015, pp. 1–9.
[4] K. He *et al.*, "Deep residual learning for image recognition," in *proc. of the IEEE conf. on computer vision and pattern recognition*, 2016, pp. 770–778.
[5] A. Mirzaeian *et al.*, "Tcd-npe: A re-configurable and efficient neural processing engine, powered by novel temporal-carry-deferring macs," in *2020 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Jan 2020.
[6] A. Mirzaeian *et al.*, "Nesta: Hamming weight compression-based neural proc. engine," in *Proceedings of the 25th Asia and South Pacific Design Automation Conference*, 2020.
[7] K. Neshatpour *et al.*, "ICNN: An iterative implementation of convolutional neural networks to enable energy and computational complexity aware dynamic approximation," in *2018 Design, Automation & Test in Europe Conf. & Exhibition (DATE)*. IEEE, 2018, pp. 551–556.
[8] K. Neshatpour *et al.*, "Icnn: The iterative convolutional neural network," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 6, Dec. 2019.
[9] K. Neshatpour *et al.*, "Exploiting energy-accuracy trade-off through contextual awareness in multi-stage convolutional neural networks," in *20th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2019, pp. 265–270.
[10] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[11] A. Kurakin *et al.*, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.
[12] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.
[13] S.-M. Moosavi-Dezfooli *et al.*, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conf. on computer vision and pattern recognition*, 2016, pp. 2574–2582.
[14] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[15] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.
[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.
[17] S.-M. Moosavi-Dezfooli *et al.*, "Universal adversarial perturbations," in *Proceedings of the IEEE Conf. on computer vision and pattern recognition*, 2017.
[18] N. Papernot *et al.*, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.
[19] N. Papernot, P. McDaniel *et al.*, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conf. on computer and communications security*. ACM, 2017, pp. 506–519.
[20] I.-T. Chen and B. Sirkeci-Mergen, "A comparative study of autoencoders against adversarial attacks," in *Proceedings of the Int. Conf. on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer , 2018.
[21] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conf. on Computer and Communications Security*. ACM, 2017, pp. 135–147.
[22] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Thirty-second AAAI Conf. on artificial intelligence*, 2018.
[23] T. Pang *et al.*, "Towards robust detection of adversarial examples," in *Advances in Neural Information Processing Systems*, 2018, pp. 4579–4589.
[24] X. Wang *et al.*, "Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses," in *Proceedings of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI'19)*, 2019.
[25] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
[26] H. Drucker and Y. Le Cun, "Improving generalization performance using double backpropagation," *IEEE Trans. on Neural Networks*, vol. 3, no. 6, 1992.
[27] J. Liu *et al.*, "Detection based defense against adversarial examples from the steganalysis point of view," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 4825–4834.
[28] M. Sabokrou *et al.*, "Self-supervised representation learning via neighborhood-relational encoding," in *ICCV*, 2019, pp. 8010–8019.
[29] P. Vincent *et al.*, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th Int. Conf. on Machine Learning*, ser. ICML '08, 2008, pp. 1096–1103.
[30] H. Xiao *et al.* (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.
[31] A. Krizhevsky *et al.*, "Cifar-10 (canadian institute for advanced research)."
[32] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.