Denoising and Verification Cross-Layer Ensemble Against Black-box Adversarial Attacks

Ka-Ho Chow, Wenqi Wei, Yanzhao Wu, Ling Liu
School of Computer Science
Georgia Institute of Technology
Atlanta, GA, USA 30332

Abstract—**Deep neural networks (DNNs) have demonstrated impressive performance on many challenging machine learning tasks. However, DNNs are vulnerable to adversarial inputs generated by adding maliciously crafted perturbations to the benign inputs. As a growing number of attacks have been reported to generate adversarial inputs of varying sophistication, the defense-attack arms race has been accelerated. In this paper, we present MODEF, a cross-layer model diversity ensemble framework. MODEF intelligently combines unsupervised model denoising ensemble with supervised model verification ensemble by quantifying model diversity, aiming to boost the robustness of the target model against adversarial examples. Evaluated using eleven representative attacks on popular benchmark datasets, we show that MODEF achieves remarkable defense success rates, compared with existing defense methods, and provides a superior capability of repairing adversarial inputs and making correct predictions with high accuracy in the presence of black-box attacks.**

Index Terms—**adversarial deep learning, ensemble defense, ensemble diversity, robustness**

I. INTRODUCTION

The recent advances in deep neural networks (DNNs) have powered numerous applications in different domains due to their outstanding performance compared to traditional machine learning techniques. However, it has been shown that DNNs can be easily fooled by adversarial inputs [1] and become a double-edged sword as their vulnerability to adversarial attacks has posed serious threats to many security-critical applications, such as biometric authentication and autonomous driving. As a number of defenses are being proposed, more attacks of varying sophistication have been put forward, accelerating the defense-attack arms race. Some even argue that designing new attacks requires much less efforts than developing effective defenses. Thus, improving the robustness and defensibility against adversarial attacks is crucial.

Adversarial examples are generated by maliciously perturbing benign examples sent to the target DNN model through querying its prediction API, aiming to fool and mislead the target model to misclassify by producing incorrect predictions randomly (untargeted attack) or purposefully (targeted attack). Given a D-dimensional benign example $\boldsymbol{x} \in \mathbb{R}^D$ and a K-class classification target model $\text{TM} : \mathbb{R}^D \rightarrow \mathbb{R}^K$ such that the prediction is given as $C_{\boldsymbol{x}} = \arg\max_{1 \leq i \leq K} \text{TM}_i(\boldsymbol{x})$, the generation process of the adversarial example $\boldsymbol{x}'$ can be formulated as

$$\min ||\boldsymbol{x} - \boldsymbol{x}'||_p \quad s.t. \ C_{\boldsymbol{x}'} = y^*, \ C_{\boldsymbol{x}'} \neq C_{\boldsymbol{x}}, \quad (1)$$

where p is the distance metric and y^* denotes the target class label for targeted attacks and any incorrect class label for untargeted attacks. For image inputs, the distance metric p can be 0, 2, or ∞ representing L_0, L_2, and L_∞ norms where L_0 norm counts the number of pixels of $\boldsymbol{x}$ that are changed, L_2 norm is the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$, and L_∞ norm denotes the maximum change to any pixel of $\boldsymbol{x}$.

Related Work. Existing defense proposals are mostly attack-dependent and can be broadly divided into three categories. *Adversarial training* counters known attacks by retraining the target model using adversarial examples generated by each attack algorithm [2]–[4]. *Input transformation* defenses apply noise reduction techniques to input data to reduce the sensitivity of the target model to small changes due to adversarial perturbations. However, to distinguish adversarial examples from benign ones, they rely on finding a dataset-specific or attack-specific threshold [5]–[7]. *Gradient masking* defenses aim to harden the process of generating adversarial examples by hiding the gradient information from attackers, such as distillation training techniques [8]. But they either significantly reduce the benign accuracy of the target model or are vulnerable to attack transferability [9], [10].

In this paper, we present MODEF, a cross-layer MOdel Diversity Ensemble Framework by integrating unsupervised model denoising ensemble with supervised model verification ensemble, aiming to protect the target model against adversarial attacks. From a manifold learning perspective [11], natural high-dimensional data concentrate close to a nonlinear low-dimensional manifold, and adversarial attacks can be considered as a malicious process to drag benign examples away from the manifold where they concentrate. MODEF first exploits denoising autoencoders to learn such manifold and to map adversarial examples back to their corresponding benign form. Different from prior works using denoising autoencoders as an input preprocessing-based defense [12], [13], MODEF leverages multiple denoisers to boost defensibility by joint force and quantifies model diversity to enable strategic teaming of denoisers with diverse denoising effects. To further improve the robustness and repair those adversarial examples escaped from the denoising autoencoders, the denoised example is sent to our second layer of model verification ensemble, which exploits the weak spots of attack transferability using a team of failure-independent models, for prediction verification and repairing. We further enhance the defensibility of MODEF

with defense structure randomization by creating a pool of models and enabling strategic randomized ensemble teaming. By promoting such uncertainty at runtime, it allows MODEF to combat adversarial attacks with higher robustness.

In summary, MODEF advances existing works from three perspectives. First, it provides three model diversity ensemble defenses against adversarial examples: the model denoising ensemble, the model verification ensemble, and the denoising-verification cross-layer ensemble, each improves the previous layer by empowering the target model with higher robustness. Second, MODEF is by design attack-independent and can generalize well over attack algorithms. MODEF does not use dataset-specific or attack-specific magic parameters, such as the detection thresholds, to distinguish adversarial examples from benign ones. Furthermore, MODEF enables strategic defense structure randomization to improve its defensibility against adversarial examples and hardens black-box attacks [9]. Extensive experiments on popular benchmark datasets with eleven representative attacks are used to validate that MODEF can significantly improve the robustness of a target DNN model against adversarial attacks.

The remainder of this paper is organized as follows. We first briefly review the representative attacks and the benchmark datasets used in this paper in Section II. Then, we introduce the Model Denoising Ensemble Defense in Section III and the Model Verification Ensemble Defense and the Denoising-Verification Cross-Layer Ensemble Defense in Section IV, followed by experimental evaluation in Section V. We conclude the paper in Section VI.

II. ADVERSARIAL EXAMPLES

Adversarial examples are generated over their corresponding benign examples by adding maliciously crafted perturbations that can fool the target model to misclassify during the model prediction (testing) phase. For the attack threat model, if the adversary can generate adversarial examples by only accessing the prediction API of the target model with no knowledge of the target model training process, we call such attack the black-box attack. In this paper, we focus on defense strategies against black-box attacks.

We generate adversarial examples using seven representative attack algorithms: FGSM [1], BIM [3], CW_0, CW_2, CW_∞ [10], DeepFool [14], and JSMA [15] which cover a wide spectrum of techniques including both L_∞, L_2, and L_0 distortions. We build those attacks on top of EvadeML-Zoo [6] using the same set of hyperparameters. For each targeted attack, we study two attack targets representing two ends of the targeted attack spectrum: the most-likely (ML) attack class in the prediction vector (i.e., $y^* = \arg\max_{1 \leq i \leq K, i \neq C_x} \text{TM}_i(x)$) and the least-likely (LL) attack class (i.e., $y^* = \arg\min_{1 \leq i \leq K} \text{TM}_i(x)$). Thus we have a total of eleven different attacks.

Table I summarizes the evaluation of the attacks for two popular benchmark datasets: MNIST and CIFAR-10. MNIST consists of $70,000$ gray-scale images of ten handwritten digits, each image is $28 \times 28 \times 1$ in size with $60,000$ images for

TABLE I: Evaluation of the attacks on MNIST and CIFAR-10.

Configuration			Cost (s)	ASR	MR	Prediction Confidence	Distortion		
	Attack	Mode					L_∞	L_2	L_0
MNIST L_∞	FGSM	UA	0.003	0.46	0.46	0.9475	0.302	5.931	0.563
	BIM		0.01	0.92	0.92	0.9982	0.302	4.819	0.522
	CW$_\infty$	ML	57.2	1.00	1.00	0.9999	0.226	3.235	0.416
		LL	50.1	1.00	1.00	0.9998	0.279	4.655	0.507
L_2	CW$_2$	ML	0.3	1.00	1.00	0.9999	0.603	2.151	0.443
		LL	0.3	1.00	1.00	0.9999	0.733	3.207	0.458
L_0	CW$_0$	ML	65.1	1.00	1.00	0.9999	0.983	3.667	0.029
		LL	61.8	1.00	1.00	0.9999	0.995	5.122	0.061
	JSMA	ML	0.6	0.93	0.93	0.7137	1.000	3.728	0.031
		LL	0.7	0.43	0.53	0.6253	1.000	5.555	0.063
CIFAR-10 L_∞	FGSM	UA	0.1	0.86	0.86	0.9694	0.016	0.864	0.998
	BIM		0.4	0.92	0.92	0.9872	0.008	0.367	0.994
	CW$_\infty$	ML	186.0	1.00	1.00	0.9890	0.009	0.341	0.960
		LL	197.6	1.00	1.00	0.9751	0.014	0.522	0.994
L_2	DF	UA	0.5	1.00	1.00	0.8382	0.028	0.028	0.993
	CW$_2$	ML	6.1	1.00	1.00	0.9866	0.024	0.204	0.641
		LL	7.0	1.00	1.00	0.9730	0.041	0.353	0.847
L_0	CW$_0$	ML	459.4	1.00	1.00	0.9898	0.581	1.547	0.010
		LL	461.2	1.00	1.00	0.9764	0.694	2.517	0.024
	JSMA	ML	7.5	1.00	1.00	0.5326	0.843	3.681	0.046
		LL	12.0	0.98	1.00	0.3984	0.904	5.563	0.098

training and the remaining $10,000$ images for testing. CIFAR-10 consists of $60,000$ colorful images of ten classes, each is $32 \times 32 \times 3$ in size. Similarly, for CIFAR-10, $50,000$ images are used for training with the remaining $10,000$ images for testing. For MNIST, the target model is a seven-layer CNN [10] with an accuracy of 0.9943. For CIFAR-10, the target model is a DenseNet [16] with an accuracy of 0.9484. The first 100 testing examples (10 per class) that are correctly classified by the target model are selected to generate adversarial examples. We exclude DeepFool (DF) from MNIST because the generated images are unrecognizable to humans.

Attack success rate (ASR) measures the percentage of successful adversarial examples over all attacked inputs while misclassification rate (MR) is defined as the percentage of misclassified adversarial examples over all attacked inputs. For untargeted attacks (UA), MR is the same as ASR, but for targeted attacks, MR may be higher than ASR because an adversarial example may fail the targeted attack but still cause misclassification, resulting in a wrong prediction output other than the true class. Most adversarial attacks evaluated produce a high ASR and MR with high prediction confidence, meaning that the target model being attacked predicts adversarial examples to be a wrong class with a high probability.

We utilize the following metrics to evaluate defensibility: *Prevention Success Rate (PSR)* measures the percentage of the adversarial examples that are repaired and correctly classified by the target model under defense. With the benign testing set, PSR denotes the benign accuracy. *Detection Success Rate (TSR)* computes the percentage of adversarial examples that could not be repaired but are correctly flagged as the attack example by the defense. For the benign testing set, TSR is the benign false positive rate, i.e., the percentage of the benign examples being flagged as adversarial over the total number of benign examples. *Defense Success Rate (DSR)* is the percentage of adversarial examples that are either repaired or detected (DSR = PSR + TSR). *False Positive Rate (FP)* measures the percentage of the adversarial examples that can be correctly classified (repaired) but are flagged as adversarial when all inputs are adversarial examples. For the
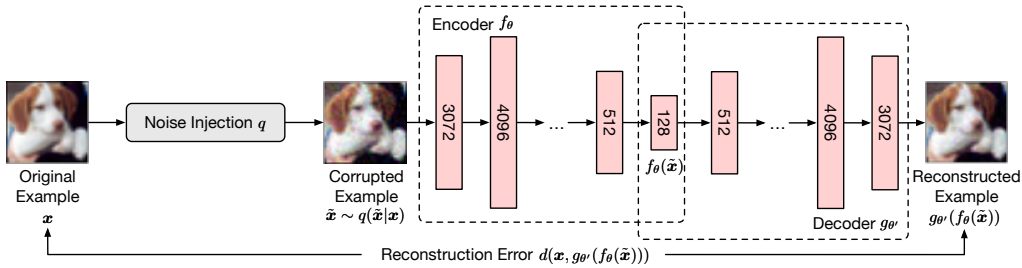
Fig. 1: The training process of a denoising autoencoder with an example from CIFAR-10.

benign testing set, it represents the percentage of the correctly classified benign examples being flagged as adversarial. All evaluations on benign examples are based on the entire testing set with $10,000$ images.

III. MODEL DENOISING ENSEMBLE DEFENSE

We design the model denoising ensemble defense as the first perimeter of defense to prevent adversarial misclassification. The defense structure consists of multiple DNN denoisers, each performs noise reduction via unsupervised learning using one specific denoising autoencoder, aiming to remove adversarial perturbations as much as possible by joint force.

A. Denoising Autoencoders

Denoising autoencoders are initially proposed as a way to extract and compose robust features which can be utilized to replace and optimize the random initialization and bootstrap the efficiency of training deep neural networks [17]. From a manifold learning perspective [11], natural high-dimensional data concentrate close to a nonlinear low-dimensional manifold. Adversarial attacks can be considered as a malicious process to drag benign examples away from the manifold where they concentrate. DNN denoising training is to learn a function to map a corrupted example, likely to be outside and farther from the manifold, back to its uncorrupted form. Thus, we can utilize denoising autoencoders, trained with uniformly corrupted examples, to reverse the adversarial perturbation process such that the adverse effect can be removed. Several efforts [18], [19] have been proposed to perform image denoising tasks with different DNN models and different design of autoencoder structures and hyperparameters. For image data, a denoising autoencoder takes an input image, transforms it into a noisy version, and feeds the noisy image to the autoencoder to perform latent space projection and then reconstructs the image with the goal of generating a clean version of the original image as the output. Figure 1 illustrates, by example, the key components of training a denoising autoencoder.

Let $\boldsymbol{x}$ be an example from the training set and $\tilde{\boldsymbol{x}}$ be the version corrupted by a stochastic noise mapping q such that $\tilde{\boldsymbol{x}} \sim q(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ [20]. The encoder is an L_f-layer neural network, which projects the corrupted example $\tilde{\boldsymbol{x}}$ from a high-dimensional image space to a low-dimensional latent feature space, producing its latent representation $f_\theta(\tilde{\boldsymbol{x}}) = f^{L_f}(\cdots(f^2(f^1(\tilde{\boldsymbol{x}};\theta_1);\theta_2));\theta_{L_f})$ where f^i is the operation at the i-th encoding layer (e.g., convolution) with weights θ_i, f_θ and $\theta = (\theta_1,...,\theta_{L_f})$ can be viewed as the composite function

of the encoder and the weights respectively. The decoder, an L_g-layer neural network, then restores the spatial structure of $f_\theta(\tilde{\boldsymbol{x}})$ by mapping its latent representation back to the original image feature space and produces the reconstructed example $g_{\theta'}(f_\theta(\tilde{\boldsymbol{x}})) = g^{L_g}(\cdots(g^2(g^1(f_\theta(\tilde{\boldsymbol{x}});\theta'_1);\theta'_2));\theta'_{L_g})$ where g^i is the operation at the i-th decoding layer (e.g., deconvolution) with weights θ'_i, $g_{\theta'}$ and $\theta' = (\theta'_1,...,\theta'_{L_g})$ can be viewed as the composite function of the decoder and the weights respectively. The multilayer encoder and decoder constitute a deep denoising autoencoder. Given N training examples $\{\boldsymbol{x}_1,...,\boldsymbol{x}_N\}$, the denoising autoencoder is trained by backpropagation to minimize the reconstruction loss:

$$
\begin{aligned}
&\mathcal{L}(\theta,\theta';\{\boldsymbol{x}_i\}_{i=1}^N, d, q, \{f^i\}_{i=1}^{L_f}, \{g^i\}_{i=1}^{L_g}, \lambda) \\
&= \frac{1}{N}\sum_{i=1}^N d(\boldsymbol{x}_i, g_{\theta'}(f_\theta(\tilde{\boldsymbol{x}}_i))) + \frac{\lambda}{2}(||\theta||_{\mathrm{F}}^2 + ||\theta'||_{\mathrm{F}}^2),
\end{aligned}
\tag{2}
$$

where d is a distance function and λ is a regularization hyperparameter penalizing the Frobenius norm of θ and θ'. Given a query example $\boldsymbol{x}$ at runtime, the denoising autoencoder produces a denoised version $\mathcal{D}(\boldsymbol{x}) = g_{\theta'}(f_\theta(\boldsymbol{x}))$ with fixed weights (θ, θ').

B. Strategic Teaming of Multiple DNN Denoisers

Recall in Figure 1 that the noise injection function q of a denoising autoencoder transforms the original input $\boldsymbol{x}$ to a noisy version of $\boldsymbol{x}$, denoted by $\tilde{\boldsymbol{x}}$. Clearly, different data corruption methods produce different versions of input $\boldsymbol{x}$ and result in denoisers that exhibit different denoising effects. Figure 2 visualizes two testing examples from MNIST and CIFAR-10 and the corresponding adversarial examples generated by six attack methods (the 1st row), their denoised versions produced by two denoisers trained with Gaussian noise (the 2nd row) and salt-and-pepper noise (the 3rd row) respectively. The predicted class label and confidence by the target model are presented for each case. On the first row, it shows that adversarial attacks successfully fool the target model under all six attacks for the examples from both datasets. On the second row, Gaussian denoiser successfully remove malicious perturbations from four out of six attacks for MNIST example and three out of six attacks for CIFAR-10 example, showing the robustness improvement of the target model even with single DNN denoiser. On the third row, the salt-and-pepper denoiser successfully repaired five out of six attacks for CIFAR-10 example and three out of six attacks for MNIST example. These examples deliver two important messages: First, exploiting denoising autoencoders trained
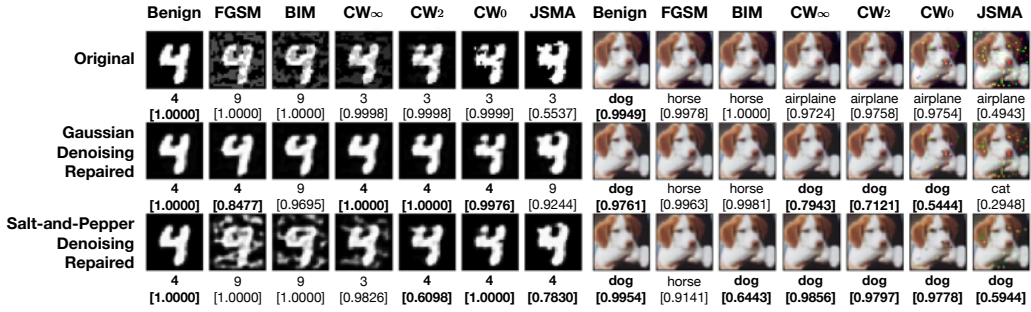
Fig. 2: The visualization of denoising effects by two denoising autoencoders on MNIST (left) and CIFAR-10 (right).

	Benign	FGSM	BIM	CW$_\infty$	CW$_2$	CW$_0$	JSMA	Benign	FGSM	BIM	CW$_\infty$	CW$_2$	CW$_0$	JSMA
Original	4 [1.0000]	9 [1.0000]	9 [1.0000]	3 [0.9998]	3 [0.9998]	3 [0.9999]	3 [0.5537]	dog [0.9949]	horse [0.9978]	horse [1.0000]	airplaine [0.9724]	airplane [0.9758]	airplane [0.9754]	airplane [0.4943]
Gaussian Denoising Repaired	4 [1.0000]	4 [0.8477]	9 [0.9695]	4 [1.0000]	4 [1.0000]	4 [0.9976]	9 [0.9244]	dog [0.9761]	horse [0.9963]	horse [0.9981]	dog [0.7943]	dog [0.7121]	dog [0.5444]	cat [0.2948]
Salt-and-Pepper Denoising Repaired	4 [1.0000]	9 [1.0000]	9 [1.0000]	3 [0.9826]	4 [0.6098]	4 [1.0000]	4 [0.7830]	dog [0.9954]	horse [0.9141]	dog [0.6443]	dog [0.9856]	dog [0.9797]	dog [0.9778]	dog [0.5944]

with uniformly corrupted examples can remove meticulously crafted adversarial perturbations and restore the classification capability of the target model. Second, no single denoiser is effective across all attacks and each of them is good at removing some types of noise but not the others. This motivates us to employ a team of diverse denoisers, instead of relying on a single denoiser, such that an adversarial example can be denoised from different perspectives.

In our model denoising ensemble defense, we first construct a set of base DNN denoisers by exploiting different approaches to generate denoisers. For example, we could use different input transformation techniques to perform the input corruption process q (e.g., Gaussian noise, salt-and-pepper noise, and masking noise) such that the optimization in Equation 2 attempts to learn different functions to reverse the corruption [18]. Alternatively, we can also create different denoisers by altering the model structure (i.e., $\{f^i\}_{i=1}^{L_f}$ and $\{g^i\}_{i=1}^{L_g}$), or the hyperparameter settings (e.g., λ, training epochs, and random seed for weight initialization), because different ways of generating denoisers can have different effects with respect to the manifold and the convergence of the DNN learning [21]–[23]. The third approach to generate different denoisers is to use different optimization objectives, such as changing the distance function d from a simple per-pixel loss to a perceptual loss [24], or using an advanced regularization such as a sparsity constraint [25].

Upon obtaining a set of base denoisers, we next define ensemble diversity metric to identify and select diverse ensembles such that each of the ensemble members has high benign accuracy and the ensemble teaming has high diversity in terms of failure-independence and low negative example correlation. In our first prototype of MODEF, we use the kappa coefficient (κ) [26] as a pairwise metric to quantify the diversity of an ensemble by measuring each pair of its member denoisers from a statistical perspective. The top-ranked ensemble teams by kappa diversity from the set of base denoisers will be chosen as the pool of our candidate ensembles.

Kappa Diversity Metric. Let N^- denote the number of testing examples that the denoised versions are labeled as different classes by the target model, N_{ij}^- denote the number of instances in the testing set that the target model labels one denoised version as class i and the other as class j. The kappa

metric can be computed by

$$\kappa = \frac{\frac{1}{N^-}\sum_{i=1}^{K} N_{ii}^- - \sum_{i=1}^{K}\left(\frac{N_{i*}^-}{N^-} - \frac{N_{*i}^-}{N^-}\right)}{1 - \sum_{i=1}^{K}\left(\frac{N_{i*}^-}{N^-} - \frac{N_{*i}^-}{N^-}\right)}. \quad (3)$$

Based on the kappa value of each pair of base denoisers, we enumerate and rank all possible ensemble teaming combinations according to their average kappa values and maintain a "κ-ranked list" of denoising ensembles with the low average kappa values below a system-supplied threshold to ensure the ensemble used by MODEF has high denoising diversity.

Denoising Ensemble Selection and Output Decision. We select one denoising ensemble team from the κ-ranked list of denoising ensembles at runtime. For the chosen denoising ensemble of size Z_D, we send the query (test) example x to each ensemble member and obtain Z_D denoised versions of x. There are several ways to produce the ensemble output accordingly. In the first prototype of MODEF, we use a majority voting method. When the target model makes a consistent prediction on a majority of the denoised versions, we randomly choose one of the majority members and uses its denoised version as the ensemble output. Otherwise, the query example is flagged as adversarial.

Figure 2 provides an illustrative example. First, the benign case refers to the target model without attacks (the 1st column for MNIST and the 8th column for CIRAR-10). The 2nd to the 7th columns correspond to six attacks for MNIST and the 9th to the 14th correspond to six attacks for CIFAR-10. The first row shows that the attacks to the two original test examples (digit 4 in MNIST and dog in CIFAR-10) are successful with high confidence. However, when we employ the two denoisers (see the 2nd row and the 3rd row), the six adversarial attacks may fail for both test examples with either Gaussian denoiser or salt-and-pepper denoiser. For MNIST digit 4, the denoising ensemble will survive under CW_0 and CW_2. For CIFAR-10 dog example, the denoising ensemble will survive under all three CW attacks. This example also illustrates that some attack algorithms, such as FGSM, BIM and JSMA, can escape from the denoising ensemble defense. One may argue that the reconstruction vectors from a well-trained denoising autoencoder form a vector field which points in the direction of the data manifold. Yet, this may not hold for the examples distant from the manifold as they are rarely sampled during training [27]. Thus, a more comprehensive solution approach should be developed. This motivates us to propose the output

model verification ensemble (Section IV-A) and to integrate the input denoising and output verification to formulate a cross-layer model ensemble defense (Section IV-B).

IV. DENOISING AND VERIFICATION CO-DEFENSE

In this section, we first describe the model verification ensemble as an alternative defense method and then present our denoising-verification cross-layer ensemble defense framework, which takes the model denoising ensemble as the front-end defense and then combines it with the model verification ensemble as the back-end defense.

A. Model Verification Ensemble Defense

One of the intriguing properties of adversarial examples is the attack transferability [28]. The main idea of model verification ensemble is to break adversarial attacks by exploiting the weak spots of adversarial transferability. It is known that two diverse models trained on the same dataset for the same learning task, their decision boundaries can be quite different even if they obtain a similar training accuracy. Hence, an adversarial example generated by one attack algorithm through one way of perturbing a benign input may successfully fool the target model but may not succeed in fooling other models trained on the same dataset, especially when such models are diverse with respect to negative examples and thus are failure-independent. By creating model ensembles with high diversity as output verifiers for the target model, we argue that (1) the model verification ensembles can improve the robustness of the target model against adversarial examples, and (2) the model verification ensembles are complementary and alternative to denoising ensembles, and thus a cross-layer diversity ensemble defense that integrates input denoising ensemble with output verification ensemble provides greater potential for higher robustness, because the adversarial examples are unlikely to be transferable to all of the member models of the cross-layer ensemble.

Similar to the creation of denoising ensembles, we perform two steps to create verification ensembles. First, we construct a set of base models as the verifiers for examining and repairing the target model prediction output, each is trained on the same dataset for the same task as the target model. Several techniques can be used to produce the base candidate models, such as varying neural network structures [23], training hyperparameters, or performing data augmentation [29]. One can also conduct snapshot learning [22] to obtain a set of model verifiers efficiently in a single run of model training. We employ two pruning criteria to generate model diversity ensembles. The first filter is to use the test accuracy to select only those base candidate models with the test accuracy similar to that of the target model. For MNIST and CIFAR-10, we collect numerous pre-trained DNN models from the public domain, with high training and test accuracy under no attack scenarios. The second filter is to select those model ensemble teams that have high model diversity with respect to high failure-independence and low negative/error correlation. In the first prototype of MODEF, we utilize the kappa coefficient metric to quantify the disagreement between each pair of base

TABLE II: The ten model verifiers for each dataset with their prediction accuracy on benign examples.

Model	MNIST		CIFAR-10	
	Name	Benign Accuracy	Name	Benign Accuracy
V_1	CNN-5	0.9919	CNN-10	0.9062
V_2	CNN-4	0.9861	ResNet20	0.9205
V_3	CNN-6	0.9917	ResNet32	0.9313
V_4	LeNet	0.9880	ResNet50	0.9312
V_5	MLP	0.9761	ResNet56	0.9419
V_6	MobileNet	0.9934	ResNet110	0.9419
V_7	MobileNet-v2	0.9939	ResNet152	0.9392
V_8	PNASNet	0.9950	ResNext29	0.9725
V_9	SVM	0.9832	VGG	0.9359
V_{10}	VGG	0.9946	WideResNet28	0.9712

candidate models. Let N^- be the number of testing examples the base candidate models produce inconsistent predictions and N_{ij}^- be the number of instances in the testing set, which are labeled as class i by one model and class j by the other in Equation 3. We then enumerate and rank all possible ensemble teaming combinations with a minimum team size of 3 and we compute the average kappa value for each ensemble team examined and select a κ-ranked list of model verification ensembles with low average kappa values using the system-defined kappa-diversity threshold. Any model ensemble from this chosen κ-ranked list will have its average kappa value below the given threshold and thus high model diversity.

Table II gives the set of ten base models for MNIST and CIFAR-10 respectively, and all ten models have similar benign accuracy under no attack scenarios. Table III illustrates the model verification ensemble method using the same two query examples: digit 4 from MNIST and a color image of dog from CIFAR-10. The 2nd row shows how the TM (target model) responds to the query example under benign (no attack) and under the six different attacks for MNIST (columns 2-8) and for CIFAR-10 (columns 9-14). When no defense protection to the target model, all adversarial examples successfully fool the target model for both digit 4 of MNIST and dog image of CIFAR-10. Interestingly, even with the adversarial example transferability, some of these adversarial examples are correctly classified by some of the ten base model verifiers (denoted by $V_1, ..., V_{10}$).

Given a query example x, a model verification ensemble of size $Z_\mathcal{V}$ is selected randomly at runtime from the κ-ranked list of ensembles as the team of verifiers for the target model, each member, denoted by V_i, takes the query example as the input and produces a prediction probability distribution, $V_i(x) : \mathbb{R}^D \rightarrow \mathbb{R}^K$, over the output space. Then, the MODEF-approved prediction output $\hat{y}$ will be produced based on a soft-voting scheme. In addition to the unweighted majority voting (recall the denoising ensemble in Section III), another example of such voting scheme could be weighted averaging based on the confidence of the prediction made by each verifier in the chosen verification ensemble:

$$\hat{y} = \arg\max_{1 \le j \le K} \frac{1}{Z_\mathcal{V}} \sum_{i=1}^{Z_\mathcal{V}} V_{i,j}(x), \quad (4)$$

where $V_{i,j}(x)$ denotes the confidence of x being from class j predicted by verifier V_i. The 2nd row from the bottom of Table III shows the results of using model verification en-

TABLE III: Predicted class label and confidence of each verifier on benign or adversarial examples. The last two rows show the predictions by model verification ensemble (Section IV-A) and denoising-verification cross-layer ensemble (Section IV-B).

	MNIST							CIFAR-10						
	Benign	FGSM	BIM	CW_∞	CW_2	CW_0	JSMA	Benign	FGSM	BIM	CW_∞	CW_2	CW_0	JSMA
TM	4 [1.0000]	9 [1.0000]	9 [1.0000]	3 [0.9998]	3 [0.9998]	3 [0.9999]	3 [0.5537]	dog [0.9949]	horse [0.9978]	horse [1.0000]	airplane [0.9724]	airplane [0.9758]	airplane [0.9754]	airplane [0.4943]
V_1	4 [0.9029]	9 [0.3287]	9 [0.6249]	9 [0.2235]	4 [0.6314]	4 [0.7399]	9 [0.5120]	dog [0.9530]	dog [0.4945]	dog [0.7772]	dog [0.8809]	dog [0.7491]	dog [0.9023]	truck [0.8392]
V_2	4 [1.0000]	9 [1.0000]	9 [1.0000]	4 [0.9925]	4 [1.0000]	4 [0.9997]	4 [0.9934]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [0.5663]
V_3	4 [1.0000]	9 [0.9994]	9 [1.0000]	4 [0.9999]	4 [1.0000]	4 [1.0000]	9 [0.9525]	dog [0.9999]	horse [0.8076]	horse [0.5280]	dog [0.9934]	dog [0.9996]	dog [0.9996]	cat [1.0000]
V_4	4 [1.0000]	9 [0.9686]	9 [0.8475]	4 [0.8105]	4 [0.9985]	4 [0.9886]	4 [0.9640]	dog [1.0000]	dog [0.7023]	dog [0.8187]	dog [0.9998]	dog [1.0000]	dog [1.0000]	cat [0.9584]
V_5	4 [1.0000]	9 [0.8796]	9 [1.0000]	9 [0.5978]	4 [0.9974]	4 [0.9939]	9 [0.9130]	dog [0.9956]	dog [0.9484]	dog [0.9881]	dog [0.9961]	dog [0.9959]	dog [0.7752]	cat [0.7850]
V_6	4 [0.9999]	4 [0.8277]	8 [0.9915]	4 [0.9949]	4 [0.9995]	4 [0.9939]	4 [0.9994]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [1.0000]	dog [0.9233]
V_7	4 [1.0000]	9 [0.6400]	9 [0.7800]	3 [0.5576]	3 [0.9908]	4 [0.4516]	4 [0.9903]	dog [1.0000]	horse [0.9994]	dog [0.8934]	dog [0.9958]	dog [0.9998]	horse [0.8080]	frog [0.9975]
V_8	4 [0.9997]	9 [0.4434]	9 [0.5201]	8 [0.8267]	4 [0.8710]	3 [0.7317]	4 [0.9998]	dog [0.8284]	dog [0.6055]	dog [0.7699]	dog [0.8572]	dog [0.8565]	dog [0.6601]	dog [0.5119]
V_9	4 [0.9948]	9 [0.5192]	9 [0.6377]	4 [0.8961]	4 [0.9885]	4 [0.9612]	4 [0.3684]	dog [0.9996]	dog [0.9987]	dog [0.9995]	dog [0.9996]	dog [0.9994]	dog [0.9975]	cat [0.9879]
V_{10}	4 [0.9997]	9 [0.9166]	9 [0.9994]	4 [0.3515]	4 [0.5573]	2 [0.4868]	9 [0.6144]	dog [0.9990]	dog [0.9985]	dog [0.9990]	dog [0.9989]	dog [0.9988]	dog [0.9989]	dog [0.9869]
Model Verification Ensemble	4 [0.9982]	9 [0.4663]	9 [0.5459]	4 [0.6303]	4 [0.9952]	4 [0.9830]	4 [0.4559]	dog [0.9775]	dog [0.6748]	dog [0.8246]	dog [0.9722]	dog [0.9599]	dog [0.8334]	cat [0.3731]
Denoising-Verification Cross-Layer Ensemble	4 [1.0000]	4 [0.9825]	4 [0.8452]	4 [0.9994]	4 [0.9996]	4 [0.9996]	4 [0.9586]	dog [0.9844]	dog [0.9576]	dog [0.9775]	dog [0.9809]	dog [0.9798]	dog [0.9742]	dog [0.9306]

semble defense with the 3-model team V_5,V_6,V_9 for MNIST and the 10-model team $V_1, ..., V_{10}$ for CIFAR-10. Without using the model denoising ensemble, we observe that for these two query inputs (digit 4 and dog), most of the adversarial examples can be correctly classified, including JSMA and CW_∞ attacks on the MNIST example, the FGSM and BIM attacks on the CIFAR-10 example, which failed by both the denoising ensemble and any of the two denoisers in Figure 2. However, the verification ensemble selected from the ten base models still fails to repair the adversarial examples generated by FGSM and BIM for digit 4 of MNIST and adversarial example generated by JSMA for the dog example of CIFAR-10. We are motivated to combine denoising ensemble with verification ensemble, which has a high probability to outperform denoising ensemble or verification ensemble alone.

B. Denoising-Verification Cross-Layer Ensemble

There are a number of ways that we can create cross-layer ensembles. Due to the space constraint, we below describe two approaches representing two ends of the spectrum. The first approach sends an ensemble voted denoising output to the model verification ensemble. The second approach sends to the model verification ensemble all denoised versions of the query input produced by every member of the denoising ensemble. Let the back-end defense to conduct the cross-layer ensemble. This approach ensures that no errors from the front-end defense will be propagated to the back-end defense phase, at the cost of sending every denoised version. Other solutions could be in between of these two spectrums, such as sending only those voted denoising outputs that are ranked higher than the system-defined consensus threshold.

One-to-Many Denoising-Verification Cross-Layer Ensemble. In the front-end defense phase, MODEF will employ denoising ensemble to produce one transformed input example by removing adversarial noises through unsupervised denoising and multiple model denoising consensus, such as unweighted majority voting, simple averaging or weighted averaging. In the back-end defense phase, it takes the voted denoised version of the query example, performs the target model prediction first and then performs the model verification ensemble over the target prediction outcome. This process serves two purposes: (1) It aims to ensure the correctness of the target model prediction outcome by repairing the prediction error through the cross-layer ensemble. (2) It also provides the detection capability to flag those adversarial examples that escape from or cannot be repaired by the cross-layer model ensemble defense framework.

Many-to-Many Denoising-Verification Cross-Layer Ensemble. When the denoising ensemble as the front-end defense chooses not to filter out any denoised versions of the query input. The back-end defense phase will need to consider each denoised version. In this case, we may choose to produce one cross-layer verified prediction outcome for each denoised version and then use an ensemble ranking algorithm, such as unweighted majority voting, simple averaging or weighted averaging, to determine the MODEF defense verified prediction outcome for the target model. Alternatively, we could also pool all the cross-layer verification results for all denoised versions together and run the ensemble consensus algorithm to rank and select the top-1 or top-k prediction outcomes.

Concretely, given a query example x at runtime, we first exercise the front-end defense by employing a model denoising

ensemble of size $Z_\mathcal{D}$ to produce $Z_\mathcal{D}$ denoised versions of $\boldsymbol{x}$, denoted by $\mathcal{D}_1(\boldsymbol{x}), ...\mathcal{D}_{Z_D}(\boldsymbol{x})$. Then, we select a model verification ensemble team of size $Z_\mathcal{V}$ from the κ-ranked list of model verification ensembles. For each of the denoised versions, we will produce $Z_\mathcal{D}$ probability distributions $P_1(\boldsymbol{x}), ..., P_{Z_\mathcal{D}}(\boldsymbol{x})$ where $P_k(\boldsymbol{x}) = \frac{1}{Z_\mathcal{V}} \sum_{i=1}^{Z_\mathcal{V}} V_i(\mathcal{D}_k(\boldsymbol{x}))$. We select the one, denoted as the c-th denoiser, producing the most confident prediction and take the corresponding predicted class label as the final defense-approved prediction. Similar to the one-to-many approach, detection capability can be offered to flag irreparable adversarial examples.

Table III shows the result of the cross-layer ensemble approach in the last row using the above many-to-many cross-layer ensemble defense. Compared with model verification ensemble only defense (the 2nd row from the bottom), clearly, the denoising-verification cross-layer ensemble can successfully verify and repair all the adversarial examples. In this experiment, the cross-layer ensemble defense method uses the two denoisers in Figure 2.

V. EXPERIMENTAL EVALUATION

A. Experimental Setup

We evaluate MODEF on two popular benchmark datasets: MNIST and CIFAR-10. Each of them is associated with one denoiser trained with Gaussian noise and another trained with salt-and-pepper noise. Their neural architectures are reported in Table IV where we use [kernel width]×[output channel] to denote parameters of convolution layers and [kernel width] to denote parameters of average pooling and upsampling layers. The volume of Gaussian noise is set to be 0.3 for MNIST and 0.01 for CIFAR-10 while the ratio of pixels randomly modified by the salt-and-pepper corruption process is set to be 0.1 for both datasets. All denoisers are trained with Adam optimizer with a learning rate of 0.001 and a batch size of 256. Euclidean distance is employed to measure the difference between two examples (i.e., the function d in Equation 2) while the regularization hyperparameter λ is set to be 10^{-9}. The number of training epochs for MNIST is 100 and that

for CIFAR-10 is 400. For each dataset, a set of ten model verifiers is collected from the public domain. As reported in Table II, each of them provides a competitive accuracy on benign examples as the target model. In our experimental studies, we consider a fixed team of DNN denoisers and demonstrate different teaming options through model verifiers. All experiments were run on Google Colab operated under Ubuntu 18.04.2 LTS with an Intel Xeon CPU (two cores @2.3 GHz), an NVIDIA Tesla K80 (12 GB) GPU, and 13 GB RAM.

B. Comparing MODEF with Existing Representative Defenses

This set of experiments compares the MODEF cross-layer ensemble defense with three representative defense methods: adversarial training [30], defensive distillation [8], and ensemble transformation of input examples [31]. We select the ensemble team to be V_5, V_6, V_9 for MNIST and $V_1, ..., V_{10}$ for CIFAR-10 from the κ-ranked list based on their model diversity. They are exploited by default in the following studies unless further specified. Table V reports the results in benign accuracy under no attack and DSR under ten attacks for MNIST and CIFAR-10 introduced in Section II. Clearly, MODEF outperforms existing schemes for all attacks on CIFAR-10 and nine out of ten attacks on MNIST. The one exception on MNIST is the FGSM attack, under which MODEF has a DSR of 0.89, slightly lower than the DSR of 0.91 by adversarial training. However, adversarial training is attack-dependent and does not generalize over attack algorithms. Indeed, existing defense methods tend to perform inconsistently across different attacks. For instance, defensive distillation reaches a high DSR of 0.90 under the CW_2 LL attack on CIFAR-10 but achieves only a low DSR of 0.47 under JSMA LL. This can also be observed in their high standard deviation of DSRs over attacks. In contrast, MODEF achieves a competitive benign accuracy with a high and stable DSR, showing that it can generalize well across all attacks.

C. Comparison of Three MODEF Ensemble Defenses

Given that MODEF outperforms existing defenses, the next set of experiments was conducted to compare the performance of three MODEF ensemble approaches and understand how denoising ensemble and verification ensemble complement one another in delivering higher robustness for the target model in the cross-layer defense. Table VI reports the results with upper wide table for MNIST and lower wide table for CIFAR-10.

Denoising Ensemble Defense. We have shown in Section II that most of the attacks successfully fool the target model by 100% attack success rate or by reducing its test accuracy significantly to zero. Table VI shows three important

TABLE IV: The neural network structures of DNN denoisers adopted in MNIST and CIFAR-10 experiments.

MNIST			CIFAR-10	
Layer	**Settings**		**Layer**	**Settings**
Convolution + Sigmoid	$3 \times 3 \times 3$		Convolution + Sigmoid	$3 \times 3 \times 3$
Average Pooling	2×2		Convolution + Sigmoid	$3 \times 3 \times 3$
Convolution + Sigmoid	$3 \times 3 \times 3$		Convolution + Sigmoid	$3 \times 3 \times 1$
Convolution + Sigmoid	$3 \times 3 \times 3$			
Upsampling	2×2			
Convolution + Sigmoid	$3 \times 3 \times 3$			
Convolution + Sigmoid	$3 \times 3 \times 1$			

TABLE V: Comparing the MODEF cross-layer ensemble defense with three representative defenses on benign accuracy and DSR.

	Defense	Benign Accuracy	FGSM UA	BIM	CW_∞ ML	CW_∞ LL	CW_2 ML	CW_2 LL	CW_0 ML	CW_0 LL	JSMA ML	JSMA LL	Average	Std
MNIST	MODEF	0.9855	0.89	**0.86**	**0.99**	**0.97**	**0.98**	**0.95**	**0.94**	**0.92**	**0.98**	**0.90**	**0.94**	**0.04**
	Adversarial Training	**0.9884**	**0.91**	0.81	0.97	0.88	0.92	0.84	0.67	0.64	0.73	0.69	0.81	0.12
	Defensive Distillation	0.9784	0.68	0.57	0.91	0.85	0.91	0.84	0.78	0.72	0.85	0.75	0.79	0.11
	Ensemble Transformation	0.9820	0.60	0.22	0.64	0.51	0.37	0.33	0.21	0.21	0.57	0.64	0.43	0.18
CIFAR-10	MODEF	**0.9631**	**0.91**	**0.96**	**0.98**	**0.98**	**0.98**	**0.98**	**0.94**	**0.93**	**0.92**	**0.80**	**0.94**	**0.06**
	Adversarial Training	0.8790	0.64	0.58	0.68	0.77	0.75	0.79	0.44	0.48	0.50	0.45	0.61	0.14
	Defensive Distillation	0.9118	0.60	0.65	0.79	0.88	0.86	0.90	0.60	0.69	0.70	0.47	0.71	0.14
	Ensemble Transformation	0.8014	0.23	0.40	0.56	0.61	0.57	0.61	0.19	0.34	0.45	0.41	0.44	0.15

TABLE VI: Comparing defensibility of the three MODEF on MNIST and CIFAR-10.

MNIST		Denoiser (Gaussian Noise)				Denoiser (Salt-and-Pepper Noise)				Model Denoising Ensemble Defense				Model Verification Ensemble Defense				Denoising-Verification Cross-Layer Ensemble Defense			
		PSR	TSR	DSR	FP	PSR	TSR	DSR	FP	PSR	TSR	DSR	FP	PSR	TSR	DSR	FP	PSR	TSR	DSR	FP
Benign		**0.9943**	0.00	0.9943	0.00	**0.9943**	0.00	0.9943	0.00	0.9911	0.0038	0.9949	0.0024	0.9888	0.00	0.9888	0.00	0.9855	0.0035	0.9890	0.0030
FGSM	UA	0.82	0.00	0.82	0.00	0.40	0.00	0.40	0.00	0.40	0.43	0.83	0.28	0.71	0.00	0.71	0.00	0.87	0.02	**0.89**	0.00
BIM		0.71	0.00	0.71	0.00	0.06	0.00	0.06	0.00	0.06	0.65	0.71	0.17	0.78	0.00	0.78	0.00	0.84	0.02	**0.86**	0.00
CW$_\infty$	ML	0.94	0.00	0.94	0.00	0.03	0.00	0.03	0.00	0.02	0.93	0.95	0.55	0.93	0.00	0.93	0.00	0.99	0.00	**0.99**	0.00
	LL	0.99	0.00	**0.99**	0.00	0.02	0.00	0.02	0.00	0.02	0.97	**0.99**	0.62	0.85	0.00	0.85	0.00	0.97	0.00	0.97	0.00
CW$_2$	ML	0.89	0.00	0.89	0.00	0.14	0.00	0.14	0.00	0.10	0.83	0.93	0.44	0.96	0.00	0.96	0.00	0.98	0.00	**0.98**	0.00
	LL	0.83	0.00	0.83	0.00	0.12	0.00	0.12	0.00	0.12	0.83	**0.95**	0.50	0.94	0.00	0.94	0.00	0.95	0.00	**0.95**	0.00
CW$_0$	ML	0.28	0.00	0.28	0.00	0.63	0.00	0.63	0.00	0.26	0.40	0.66	0.18	0.94	0.00	**0.94**	0.00	0.94	0.00	**0.94**	0.00
	LL	0.41	0.00	0.41	0.00	0.68	0.00	0.68	0.00	0.41	0.41	0.82	0.15	0.89	0.00	0.89	0.00	0.92	0.00	**0.92**	0.00
JSMA	ML	0.82	0.00	0.82	0.00	0.94	0.00	0.94	0.00	0.80	0.16	0.96	0.12	0.98	0.00	**0.98**	0.00	0.98	0.00	**0.98**	0.00
	LL	0.65	0.00	0.65	0.00	0.81	0.00	0.81	0.00	0.64	0.22	0.86	0.14	0.88	0.00	0.88	0.00	0.90	0.00	**0.90**	0.00
Average		0.73	0.00	0.73	0.00	0.38	0.00	0.38	0.00	0.28	0.58	0.86	0.32	0.89	0.00	0.89	0.00	0.93	0.00	**0.94**	0.00
Std		0.23	0.00	0.23	0.00	0.35	0.00	0.35	0.00	0.26	0.28	0.11	0.18	0.09	0.00	0.09	0.00	0.05	0.01	**0.04**	0.00

CIFAR-10		PSR	TSR	DSR	FP	PSR	TSR	DSR	FP	PSR	TSR	DSR	FP	PSR	TSR	DSR	FP	PSR	TSR	DSR	FP
Benign		0.9304	0.00	0.9304	0.00	0.8835	0.00	0.8835	0.00	0.8772	0.0656	0.9428	0.0369	0.9608	0.00	0.9608	0.00	**0.9631**	0.0002	0.9633	0.0002
FGSM	UA	0.19	0.00	0.19	0.00	0.34	0.00	0.34	0.00	0.17	0.24	0.41	0.12	0.85	0.00	0.85	0.00	0.83	0.08	**0.91**	0.00
BIM		0.22	0.00	0.22	0.00	0.50	0.00	0.50	0.00	0.21	0.33	0.54	0.13	0.92	0.00	0.92	0.00	0.95	0.01	**0.96**	0.00
CW$_\infty$	ML	0.21	0.00	0.21	0.00	0.67	0.00	0.67	0.00	0.21	0.47	0.68	0.22	0.95	0.00	0.95	0.00	0.98	0.00	**0.98**	0.00
	LL	0.83	0.00	0.83	0.00	0.85	0.00	0.85	0.00	0.77	0.17	0.94	0.11	0.98	0.00	**0.98**	0.00	0.98	0.00	**0.98**	0.00
DF	UA	0.64	0.00	0.64	0.00	0.78	0.00	0.78	0.00	0.59	0.25	0.84	0.15	0.97	0.00	0.97	0.00	0.98	0.00	**0.98**	0.00
CW$_2$	ML	0.42	0.00	0.42	0.00	0.75	0.00	0.75	0.00	0.42	0.34	0.76	0.22	0.95	0.00	0.95	0.00	0.98	0.00	**0.98**	0.00
	LL	0.87	0.00	0.87	0.00	0.90	0.00	0.90	0.00	0.82	0.14	0.96	0.09	0.97	0.00	0.97	0.00	0.98	0.00	**0.98**	0.00
CW$_0$	ML	0.03	0.00	0.03	0.00	0.33	0.00	0.33	0.00	0.03	0.33	0.36	0.10	0.88	0.00	0.88	0.00	0.86	0.08	**0.94**	0.00
	LL	0.21	0.00	0.21	0.00	0.70	0.00	0.70	0.00	0.18	0.76	**0.94**	0.34	0.91	0.00	0.91	0.00	0.93	0.00	0.93	0.00
JSMA	ML	0.40	0.00	0.40	0.00	0.77	0.00	0.77	0.00	0.39	0.45	0.84	0.31	0.83	0.00	0.83	0.00	0.92	0.00	**0.92**	0.00
	LL	0.32	0.00	0.32	0.00	0.71	0.00	0.71	0.00	0.31	0.58	**0.89**	0.38	0.54	0.00	0.54	0.00	0.79	0.01	0.80	0.00
Average		0.39	0.00	0.39	0.00	0.66	0.00	0.66	0.00	0.37	0.37	0.74	0.20	0.89	0.00	0.89	0.00	0.93	0.02	**0.94**	0.00
Std		0.27	0.00	0.27	0.00	0.19	0.00	0.19	0.00	0.25	0.18	0.21	0.10	0.12	0.00	0.12	0.00	0.07	0.03	**0.05**	0.00

results: First, by employing a DNN denoiser, either trained with Gaussian or salt-and-pepper noise, one can improve the robustness of the target model, as shown in the columns under "Denoiser (Gaussian Noise)" and "Denoiser (Salt-and-Pepper Noise)". Second, even though some attacks are mitigated with a high DSR, the effectiveness of one denoiser varies wildly across different attacks and datasets. For instance, the Gaussian denoiser on MNIST achieves a high DSR of 0.99 in defending the CW$_\infty$ LL attack but only gets a low DSR of 0.28 under the CW$_0$ ML attack. In fact, we have performed experiments on multiple denoisers, and we found consistently that no DNN denoiser can remove adversarial perturbations from all eleven attacks examined and each denoiser fails to generalize well over all attacks. Third, the DSR of model denoising ensemble is consistently better than using one denoiser as shown in the columns under "Model Denoising Ensemble Defense". This validates our argument that a robust defense method should not rely on a single defense strategy, such as one denoiser as adopted in prior works [12], [13] but employ a denoising ensemble. Also, a simple denoising ensemble as those advocated in [5] may not outperform a diversity optimized denoising ensemble, as an ensemble team of diverse denoisers offers different denoising effects and hence generalizes better over different adversarial attacks.

Verification Ensemble and Cross-Layer Ensemble Defense. Table VI reports the defensibility of model verification ensemble under the column "Model Verification Ensemble Defense". Compared with denoising ensemble, although verification ensemble achieves a higher average DSR over all eleven attacks in both datasets (i.e., 0.89 v.s. 0.86 for MNIST and 0.89 v.s. 0.74 for CIFAR-10 shown on their respective "Average" row in Table VI), it performs much worse than denoising ensemble under the JSMA LL attack (0.54 v.s. 0.89) for CIFAR-10 and CW$_\infty$ LL attack (0.85 v.s. 0.99) for MNIST. The model verification ensemble is hence a competitive alternative to the denoising ensemble, since neither of them is the winner over all attacks on both datasets, and yet each of them is the winner for some attacks. Compared with denoising or verification ensemble only defense, the cross-layer ensemble defense consistently outperforms across almost all attacks with a remarkably small standard deviation of DSRs (see the columns of "Denoising-Verification Cross-Layer Ensemble Defense" in Table VI). In other words, the cross-layer ensemble can generalize well over the eleven attacks examined.

We make another interesting observation that although the model verification ensemble and the cross-layer ensemble may have slightly lower benign accuracy (0.9888 and 0.9855) than the target model on MNIST (0.9943) under no attack, their benign accuracy on CIFAR-10 are 0.9608 and 0.9631, which are higher than the target model accuracy of 0.9484. This is consistent with the theoretical proof in [32] that ensemble can reduce uncorrelated errors. For example, if we have 5 completely independent classifiers and we use unweighted majority voting as a consensus algorithm, by the probability formula, with accuracy of 70% for each classifier, we have 83.7% ensemble accuracy with majority voting and if we have 101 such classifiers, we reach 99.9% ensemble accuracy by majority voting. Equivalently, let δ be the classification accuracy of each member in the ensemble E of size Z, the accuracy can be boosted to $P(E \geq \lceil Z/2 \rceil) = \sum_{i=\lceil Z/2 \rceil}^{Z} \binom{Z}{i} \delta^i (1-\delta)^{Z-i}$.

In MODEF, we consider three options to create an ensemble team of base models: (1) randomly selecting a team from the entire pool of ensemble combinations (**Rand**); (2) randomly

TABLE VII: The pairwise kappa value of model verifiers for MNIST.

	TM	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
TM		0.79	0.65	0.78	0.66	0.37	0.79	0.78	0.81	0.58	0.78
V_1			0.67	0.81	0.65	0.47	0.77	0.77	0.76	0.69	0.75
V_2				0.67	0.50	0.39	0.62	0.62	0.62	0.56	0.60
V_3					0.64	0.48	0.76	0.77	0.75	0.66	0.74
V_4						0.25	0.69	0.71	0.67	0.47	0.71
V_5							0.35	0.36	0.36	0.43	0.31
V_6								0.84	0.83	0.56	0.82
V_7									0.86	0.56	0.84
V_8										0.54	0.82
V_9											0.54
V_{10}											

selecting an ensemble team from the κ-ranked list (**Randκ**), which filters out those ensemble team combos that have high kappa values above the system-defined threshold (due to the space, the factors impact on the decision for the threshold setting is omitted here); and (3) selecting the ensemble team from the κ-ranked list with the highest prediction accuracy (**Bestκ**). The κ-ranked list is built on the kappa value of each pair of model verifiers. Different levels of disagreement occur between a pair of model verifiers. Table VII gives an example of the pairwise kappa values of the model verifiers on MNIST where the pair of V_4 and V_5 has a low kappa value of 0.25, showing their high prediction disagreement diversity (high failure-independence and low error correlation).

To understand the effectiveness of three different ensemble teaming options, we report in Table VIII the defensibility of the ensembles randomly sampled according to the above definitions. We make three observations. First, the target model suffers different levels of failure under various attacks. In comparison, each of the model verifiers consistently provides better robustness against most of the attacks than that of the target model. For instance, each verifier has higher DSRs under CW attacks, JSMA attacks, and BIM attacks and comparable performance under FGSM. Second, verification ensemble defense has better robustness compared to each of the individual verifiers. For each verification ensemble, its average DSR is higher than that of individual ensemble member. Third, using an ensemble team from the κ-ranked list performs better in many cases than random ensemble selection because the ensembles in the κ-ranked list have low kappa value and thus high model diversity with respect to failure-independence or error correlation. In addition to the above observations, the cross-layer ensembles significantly outperform their corresponding verification ensembles with the same team and performs much better than individual ensemble member. One example is the Randκ team with V_3, V_4, V_{10} having an average DSR over all attacks of 0.72, 0.62, and 0.48 respectively while the cross-layer ensemble method makes a significant improvement to 0.89, demonstrating the complementary property of denoising and verification ensembles.

The above shows that all three MODEF ensembles improve the test accuracy of the target model under eleven attacks on the two benchmark datasets. This indicates that ensembles with good diversity hold great potential for improving robustness of DNN models in the presence of adversarial inputs. We notice that verification ensemble offers a stronger ability to repair adversarial examples with correct predictions while denoising ensemble tends to be more robust by flagging and rejecting adversarial examples. The cross-layer ensemble advocates high robustness with a high DSR by achieving a high PSR, which

TABLE VIII: Prediction accuracy of the target model on MNIST, ten base model verifiers, three different teamings for model verification ensemble and denoising-verification cross-layer ensemble.

Model		Benign Accuracy	FGSM	BIM	CW$_\infty$ ML	CW$_\infty$ LL	CW$_2$ ML	CW$_2$ LL	CW$_0$ ML	CW$_0$ LL	JSMA ML	JSMA LL	Average	Std
TM: Target Model		0.9943	0.54	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.47	0.12	0.21
V_1: CNN-5		0.9919	0.58	0.27	0.59	0.43	0.74	0.78	0.72	0.72	0.90	0.81	0.65	0.19
V_2: CNN-4		0.9861	0.65	0.44	0.85	0.68	0.85	0.74	0.60	0.57	0.76	0.63	0.68	0.13
V_3: CNN-6		0.9917	0.65	0.37	0.81	0.71	0.82	0.89	0.73	0.65	0.86	0.73	0.72	0.15
V_4: LeNet		0.9880	0.49	0.44	0.72	0.31	0.72	0.63	0.70	0.57	0.86	0.74	0.62	0.17
V_5: MLP		0.9761	0.51	0.44	0.74	0.61	0.88	0.87	0.83	0.81	0.85	0.74	0.73	0.16
V_6: MobileNet		0.9934	0.19	0.18	0.28	0.16	0.49	0.36	0.76	0.55	0.95	0.87	0.48	0.30
V_7: MobileNet-v2		0.9939	0.22	0.23	0.36	0.11	0.34	0.14	0.70	0.44	0.96	0.86	0.44	0.30
V_8: PNASNet		0.9950	0.15	0.15	0.19	0.08	0.20	0.10	0.68	0.43	0.94	0.80	0.37	0.32
V_9: SVM		0.9832	0.67	0.72	0.98	0.92	**0.98**	**0.97**	0.93	0.86	0.96	0.80	0.88	0.11
V_{10}: VGG		0.9946	0.40	0.32	0.59	0.03	0.49	0.35	0.62	0.32	0.89	0.80	0.48	0.25
Model Verification Ensemble	Rand = V_1,V_2,V_4,V_7,V_{10}	**0.9951**	0.61	0.39	0.78	0.51	0.76	0.69	0.79	0.68	0.94	0.80	0.70	0.16
	Randκ = V_3,V_4,V_{10}	0.9919	0.63	0.42	0.84	0.65	0.88	0.88	0.80	0.77	0.92	0.80	0.76	0.15
	Bestκ = V_5,V_6,V_9	0.9888	0.71	0.78	0.93	0.85	0.96	0.94	**0.94**	0.89	**0.98**	0.88	0.89	0.09
Denoising-Verification Cross-Layer Ensemble	Rand = V_1,V_2,V_4,V_7,V_{10}	0.9912	**0.90**	0.76	0.95	**0.98**	0.86	0.81	0.83	0.84	0.95	0.85	0.87	0.07
	Randκ = V_3,V_4,V_{10}	0.9873	0.84	0.71	0.98	0.94	0.96	0.92	0.88	0.88	0.95	0.88	0.89	0.08
	Bestκ = V_5,V_6,V_9	0.9855	0.89	**0.86**	**0.99**	0.97	**0.98**	0.95	**0.94**	**0.92**	**0.98**	**0.90**	**0.94**	**0.04**

TABLE IX: Transferability of adversarial examples generated over the target model.

		TM	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	Model Verification Ensemble	Denoising-Verification Cross-Layer Ensemble
FGSM	UA	1.00	0.72	0.63	0.74	0.74	0.67	0.76	0.76	0.83	0.57	0.70	0.63	**0.24**
BIM		1.00	0.79	0.61	0.68	0.61	0.61	0.83	0.78	0.86	0.30	0.71	0.24	**0.15**
CW$_\infty$	ML	1.00	0.34	0.13	0.19	0.21	0.14	0.21	0.35	0.36	0.02	0.23	0.06	**0.01**
	LL	1.00	0.28	0.06	0.10	0.15	0.07	0.04	0.30	0.16	0.00	0.15	0.03	**0.00**
CW$_2$	ML	1.00	0.24	0.10	0.18	0.22	0.08	0.35	0.55	0.61	0.02	0.39	0.04	**0.01**
	LL	1.00	0.03	0.04	0.01	0.06	0.02	0.07	0.45	0.45	0.00	0.21	0.02	**0.00**
CW$_0$	ML	1.00	0.25	0.22	0.24	0.27	0.11	0.20	0.20	0.25	0.03	0.34	0.05	**0.03**
	LL	1.00	0.06	0.12	0.14	0.06	0.02	0.11	0.16	0.19	0.01	0.16	**0.00**	**0.00**
JSMA	ML	1.00	0.06	0.13	0.13	0.10	0.08	0.05	0.02	0.02	0.01	0.08	**0.01**	**0.01**
	LL	1.00	0.07	0.19	0.16	0.05	0.09	0.00	0.00	0.00	0.02	0.00	0.02	**0.00**
Average		1.00	0.28	0.22	0.26	0.25	0.19	0.26	0.36	0.37	0.10	0.30	0.10	**0.05**
Std		0.00	0.27	0.22	0.25	0.24	0.24	0.30	0.28	0.31	0.19	0.24	0.20	**0.08**

is in stark contrast to existing detection-only methods [6].

To gain a better understanding of how the transferability of adversarial examples behaves under MODEF, we measure the attack transferability in ASR and report the results in Table IX. Due to the space constraint, only the results on MNIST are presented. Given that the adversarial examples are generated over the target model (TM), the TM column shows the transferability to be 1.00. Those adversarial examples may not be consistently transferable to other independently trained models. For instance, the adversarial examples generated by the CW_2 ML attack can be effectively transferred to V_7 and V_8 with a transferability of 0.55 and 0.61 respectively but achieve only a low transferability of 0.10 for V_2, 0.08 for V_5, and 0.02 for V_9. Such divergence in transferability of adversarial examples is the main motivation of MODEF to exploit the weak spots of attack transferability using the denoising ensemble, the verification ensemble and the denoising-verification cross-layer ensemble as effective defense methods. As shown in the last two columns in Table IX, the verification ensemble for MNIST has a very low transferability under all attacks. With the cross-layer ensemble that combines the denoising ensemble and verification ensemble, its attack transferability becomes further reduced.

VI. CONCLUSION

We have presented MODEF, a diversity ensemble defense framework against adversarial deception in deep learning. Driven by quantifying ensemble diversity based on classification failure-independence, MODEF provides three diversity ensemble methods: (1) the model denoising ensemble, leveraging multiple diverse denoising autoencoders to remove adversarial perturbations, (2) the model verification ensemble that exploits the weak spots of attack transferability to verify and repair prediction output of the target model, and (3) the denoising-verification cross-layer ensemble, which can guard input and output of the target model through intelligently combining input denoising ensemble and output verification ensemble. Furthermore, MODEF is by design attack-independent and can generalize over different attacks. Due to the space constraint, we could not include additional experimental results on the effectiveness of MODEF under new attack algorithms, including PGD [2].

ACKNOWLEDGEMENT

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[3] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[4] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *ICLR*, 2018.

[5] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *CCS*, 2017.

[6] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *NDSS*, 2018.

[7] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," in *ICLR*, 2018.

[8] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *S&P*, 2016.

[9] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ASIACCS*, 2017.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *S&P*, 2017.

[11] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, 2009.

[12] R. Sahay, R. Mahfuz, and A. El Gamal, "Combatting adversarial attacks through denoising and dimensionality reduction: A cascaded autoencoder approach," in *CISS*, 2019.

[13] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *CVPR*, 2018.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, 2016.

[15] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *EuroS&P*, 2016.

[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[17] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008.

[18] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *NIPS*, 2012.

[19] F. Agostinelli, M. R. Anderson, and H. Lee, "Adaptive multi-column deep neural networks with application to robust image denoising," in *NIPS*, 2013.

[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *JMLR*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[21] W. Wei, L. Liu, S. Truex, L. Yu, and M. E. Gursoy, "Adversarial examples in deep learning: Characterization and divergence," *arXiv preprint arXiv:1807.00051*, 2018.

[22] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," in *ICLR*, 2017.

[23] Y. Wu, L. Liu, C. Pu, W. Cao, S. Sahin, W. Wei, and Q. Zhang, "A comparative measurement study of deep learning as a service framework," *arXiv preprint arXiv:1810.12210*, 2018.

[24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[25] K. Cho, "Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images," in *ICML*, 2013.

[26] M. L. McHugh, "Interrater reliability: the kappa statistic," *Biochemia medica: Biochemia medica*, vol. 22, no. 3, 2012.

[27] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *JMLR*, vol. 15, no. 1, 2014.

[28] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[31] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *ICLR*, 2018.

[32] T. Holloway, "Introduction to ensemble learning," 2007.