Deep Minimax Probability Machine

Lirong He
SMILE Lab, School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
lirong_he@std.uestc.edu.cn

Ziyi Guo
Cloud and Smart Industries Group
Tencent
Guangzhou, China
ziyiguo94@gmail.com

Kaizhu Huang
Department of EEE
Xi'an Jiaotong-Liverpool University
Suzhou, China
Kaizhu.Huang@xjtlu.edu.cn

Zenglin Xu[1,2]
[1]*SMILE Lab, School of Computer Science and Engineering*
University of Electronic Science and Technology of China
Chengdu, China
[2]*Center for Artificial Intelligence*
Peng Cheng Laboratory
Shenzhen, China
zenglin@gmail.com

***Abstract*—Deep neural networks enjoy a powerful representation and have proven effective in a number of applications. However, recent advances show that deep neural networks are vulnerable to adversarial attacks incurred by the so-called adversarial examples. Although the adversarial example is only slightly different from the input sample, the neural network classifies it as the wrong class. In order to alleviate this problem, we propose the Deep Minimax Probability Machine (DeepMPM), which applies MPM to deep neural networks in an end-to-end fashion. In a worst-case scenario, MPM tries to minimize an upper bound of misclassification probabilities, considering the global information (i.e., mean and covariance information of each class). DeepMPM can be more robust since it learns the worst-case bound on the probability of misclassification of future data. Experiments on two real-world datasets can achieve comparable classification performance with CNN, while can be more robust on adversarial attacks.**

***Index Terms*—deep neural networks, adversarial attacks, mimimax probability machine**

I. Introduction

Deep neural networks (DNNs) are adept at learning effective representation and have demonstrated significant success in a wide variety of applications, such as image classification [1], speech recognition [2], [3] and language translation [4], [5].

However, recent advances show that they are vulnerable to adversarial examples, which are augmented samples perturbed imperceptibly but able to mislead the predictions of the neural networks [6]–[8]. The existence of this issue prevents us from applying them to security-related applications, for example, self-driving cars [9].

It has attracted a lot of research interests in improving the adversarial robustness of deep neural networks. [10] suggests reducing the dimensionality of input data. [11] indicates that their model is robust to adversarial examples putting Gaussian processes on top of deep neural networks. Furthermore, [12] developed the semi-supervised version of adversarial training called Virtual Adversarial Training (VAT) where the output distribution was smoothed with a regularization term.

To alleviate such issues, we present an alternative model which equipts DNNs with the worst case misclassification bound provided by Minimax Probability Machine (MPM) [13]–[17], leading Deep Minimax Probability Machine (DeepMPM). In essence, we put the MPM at the top of a deep neural network, as shown in Figure 1. Through exploring a powerful theorem [18], MPM can obtain the upper bound on the probability of misclassification for future data, i.e., the worst-case accuracy and hence leads to a robust classifier. In contrast, for classification, the traditional DNNs only consider the information of single example since they use softmax classifier. Combining MPM with DNNs could inherit the good advantages of both MPM and DNNs where robustness and accuracy would be integrated. In details, DeepMPM could take advantage of global information by introducing the global statistics of data, i.e., the mean and covariance of data, control misclassification probabilities robustly in the worst case for future data, and do well in learning effective hidden representation. In other words, applying MPM to DNNs can make up for the inadequacy of DNNs and are more robust to adversarial examples. Specifically, instead of maximizing the likelihood of labels for data, we employ the objective function of MPM to promote our model to take into account global information. Importantly, we engage the Lagrangian multiplier to optimize the DeepMPM which enables an end-to-end training fashion.

In order to evaluate the performance of the proposed model, we perform evaluations on two benchmark datasets, MINST and CIFAR-10, in two tasks including classification tasks and robustness to adversarial examples. Experimental results have demonstrated the encouraging performance. In particular, the significant improvement over the state-of-the-art methods has been observed in recognizing adversarial examples.

2375-9259/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDMW.2019.00127

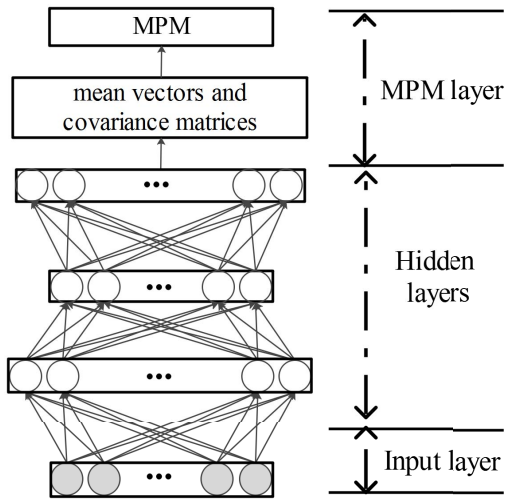

Fig. 1. The graphical illustration of the DeepMPM model. The mainly difference of the CNN and the DeepMPM is the final layer: one uses the Softmax function, the other employs the MPM layer. This causes the two models to be different in the optimization section as well. And the DeepMPM obtains the means and covariance matrices of two classes through the powerful representation of the DNN.

II. Related Work

It has been shown that deep neural networks are often vulnerable to adversarial examples. There is some literature on how to improve the adversarial robustness m from different perspectives. In the following, we present some literature on adversarial robustness.

The first aspect of work is based on adversarial training, specifically augmenting the training set with adversarial examples. It's said that adversarial training can improve the robustness of deep neural networks to adversarial examples and seems to hold the greatest promise. [19]–[25]. For instance, in order to enhance the robustness of the neural networks, [24] introduces adversarial training employing black-box attacks from similar networks. [25] demonstrates that if the perturbations calculated during training maximize the loss of the model, adversarial training can be made robust to white-box attacks. The two methods described above are only for the specified attack method.

The second aspect of work is based on defensive distillation (i.e. transfer knowledge of complex networks to smaller networks) [26], [27]. [26] employs knowledge of the network to train the networks to increase robustness. [27] extends the defensive distillation measure by addressing the numerical instabilities encountered in [26].

Meanwhile, there exists work to modify the input to improve the robustness [10], [28]. [10] points out that reducing the dimensionality of original data can improve the adversarial robustness of deep neural networks. [28] shows that training neural networks with the input gradient regularization can enhance robustness to transferred adversarial examples created to fake out all of the other models. In addition, the inherent characteristics of the model also affect robustness [29], [30]. [29] finds that the robustness can be improved by increasing the capacity of the model. When this is used together with adversarial training, the effect is more pronounced. [30] demonstrates that sparse nonlinear deep neural networks are more robust than their corresponding dense networks.

Deep neural networks can also combine the advantages of other methods to increase their own robustness. [11] puts the Gaussian process at the top of a DNN, achieving good classification and robustness. In this paper, we combine the MPM with deep neural networks for the sake of their complementary strengths in robustness learning.

III. Deep Minimax Probability Machine

In this section, we first provide an introduction to MPM. Further, we propose a novel model DeepMPM, which attempts to optimize an MPM optimization targeting in an end-to-end DNN fashion.

A. Minimax Probability Machine

Minimax Probability Machine (MPM) is proposed in [13]–[17], which tries to minimize the upper bound of the probability of misclassification of future data in a worst-case setting. No assumptions with respect to the data distribution are required in MPM, while those assumptions lack generality and are often invalid.

Let $\mathbf{x}$ and $\mathbf{y}$ denote random vectors with mean vectors and covariance matrices given by $\mathbf{x} \sim (\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ and $\mathbf{y} \sim (\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$ respectively in a binary classification problem, where $\mathbf{x}, \bar{\mathbf{x}}, \mathbf{y}, \bar{\mathbf{y}} \in \mathbb{R}^n$ and $\Sigma_{\mathbf{x}}, \Sigma_{\mathbf{y}} \in \mathbb{R}^{n\times n}$.

MPM seeks to determine the hyperplane $\mathbf{a}^\top \mathbf{z} = b$ ($\mathbf{a}, \mathbf{z} \in \mathbb{R}^n$ and $b \in \mathbb{R}$) which separates the two classes of data with maximal probability. The form of the MPM model is as follows:

$$\max_{\alpha,\mathbf{a},b} \alpha \quad \text{s.t. } \inf \Pr\{\mathbf{a}^\top \mathbf{x} \geq b\} \geq \alpha, \quad \inf \Pr\{\mathbf{a}^\top \mathbf{y} \leq b\} \geq \alpha. \tag{1}$$

where α denotes the worst-case accuracy of the future data. Through a powerful theorem due to Isii [18], as extended by [31], this finally can be transformed into a convex optimization problem, as follows,

$$\min_{\mathbf{a}} \sqrt{\mathbf{a}^\top \Sigma_{\mathbf{x}} \mathbf{a}} + \sqrt{\mathbf{a}^\top \Sigma_{\mathbf{y}} \mathbf{a}} \quad \text{s.t. } \mathbf{a}^\top (\bar{\mathbf{x}} - \bar{\mathbf{y}}) = 1, \tag{2}$$

or, equivalently

$$\min_{\mathbf{a}} \|\Sigma_{\mathbf{x}}^{1/2} \mathbf{a}\| + \|\Sigma_{\mathbf{y}}^{1/2} \mathbf{a}\| \quad \text{s.t. } \mathbf{a}^\top (\bar{\mathbf{x}} - \bar{\mathbf{y}}) = 1. \tag{3}$$

More specifically, this optimization problem is a second order cone program problem [32]. After obtaining the optimal solution a_*, b_*, for a new data point $\mathbf{z}$, if $a_* \mathbf{z} \geq b_*$, $\mathbf{z}$ is classified as the class $\mathbf{x}$, otherwise $\mathbf{z}$ belongs to the class $\mathbf{y}$. In the following sections we will introduce DeepMPM.

B. DeepMPM

It is known that deep neural networks offer a powerful representation mechanism, which could learn adaptive basis functions focusing on local useful information. At the same time, MPM can directly minimize the maximum probability of misclassification with mean vectors and covariance matrices of the data considering the global structural information.

Therefore, we combine the MPM with deep neural networks for the sake of their complementary strengths in the classification task and robustness learning, namely Deep Minimax Probability Machine (DeepMPM). Specifically, we can interpret our model as applying the MPM to the final hidden layer of a deep neural network, instead of using softmax, as shown in Figure 1.

Let $g(\mathbf{x}, \mathbf{w})$ denotes a nonlinear mapping given by a deep neural network, parametrized by weight $\mathbf{w}$. It can be said that through a neural network, we obtain effective representation for two classes of data, $g(\mathbf{x}, \mathbf{w})$ and $g(\mathbf{y}, \mathbf{w})$ respectively, making mean vectors and covariance matrices reliable. In details,

$$\mathbf{x} \sim (\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}) \rightarrow g(\mathbf{x}, \mathbf{w}) \sim (\overline{g(\mathbf{x}, \mathbf{w})}, \Sigma_{g(\mathbf{x}, \mathbf{w})}), \tag{4}$$

$$\mathbf{y} \sim (\bar{\mathbf{y}}, \Sigma_{\mathbf{y}}) \rightarrow g(\mathbf{y}, \mathbf{w}) \sim (\overline{g(\mathbf{y}, \mathbf{w})}, \Sigma_{g(\mathbf{y}, \mathbf{w})}), \tag{5}$$

where $\overline{g(\mathbf{x}, \mathbf{w})}, \overline{g(\mathbf{y}, \mathbf{w})}$ denote mean vectors of two classes of data respectively, and $\Sigma_{g(\mathbf{x}, \mathbf{w})}, \Sigma_{g(\mathbf{y}, \mathbf{w})}$ denote covariance matrices of two classes of data respectively. For simplicity, we omit the parameter $\mathbf{w}$ of $g(\cdot)$ in the following sections.

We desire a hyperplane $\mathbf{a}^\top g(\mathbf{z}) = b$ that separates the two classes of data points with maximal probability given the means and covariance matrices obtaining by a deep neural network. The formulation of our model is written as follows:

$$\begin{aligned} \max_{\alpha, \mathbf{a}, b} \alpha \quad \text{s.t.} \quad & \inf \Pr\{\mathbf{a}^\top g(\mathbf{x}) \geq b\} \geq \alpha, \\ & \inf \Pr\{\mathbf{a}^\top g(\mathbf{y}) \leq b\} \geq \alpha. \end{aligned} \tag{6}$$

With the powerful theorem used by the MPM [18], the optimized objective function of our model becomes,

$$\begin{aligned} \min_{\mathbf{a}} \ & \sqrt{\mathbf{a}^\top \Sigma_{g(\mathbf{x})} \mathbf{a}} + \sqrt{\mathbf{a}^\top \Sigma_{g(\mathbf{y})} \mathbf{a}}, \\ \text{s.t.} \ & \mathbf{a}^\top (\overline{g(\mathbf{x})} - \overline{g(\mathbf{y})}) = 1. \end{aligned} \tag{7}$$

In order to train our model in an end-to-end fashion, we employ the Lagrangian multiplier method to perform optimization. With the introduction of a Lagrange multiplier λ, we can minimize the objective function,

$$\mathcal{L} = \sqrt{\mathbf{a}^\top \Sigma_{g(\mathbf{x})} \mathbf{a}} + \sqrt{\mathbf{a}^\top \Sigma_{g(\mathbf{y})} \mathbf{a}} + \lambda(\mathbf{a}^\top (\overline{g(\mathbf{x})} - \overline{g(\mathbf{y})}) - 1). \tag{8}$$

We would like to underline that we jointly learn all our model parameters, $\phi = \{\mathbf{w}, \mathbf{a}\}$, including $\mathbf{w}$ the weights of the neural network, and $\mathbf{a}$ the parameter of the hyperplane for the MPM. In addition, we train DeepMPM with back propagation in an end-to-end fashion, using the chain rule to calculate derivatives about all the parameters. Particularly, the derivative of weight w is written as,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = & \frac{\partial \mathcal{L}}{\partial \Sigma_{g(\mathbf{x})}} \frac{\partial \Sigma_{g(\mathbf{x})}}{\partial g(\mathbf{x})} \frac{\partial g(\mathbf{x})}{\partial \mathbf{w}} + \frac{\partial \mathcal{L}}{\partial \Sigma_{g(\mathbf{y})}} \frac{\partial \Sigma_{g(\mathbf{y})}}{\partial g(\mathbf{y})} \frac{\partial g(\mathbf{y})}{\partial \mathbf{w}} \\ & + \frac{\partial \mathcal{L}}{\partial \overline{g(\mathbf{x})}} \frac{\partial \overline{g(\mathbf{x})}}{\partial g(\mathbf{x})} \frac{\partial g(\mathbf{x})}{\partial \mathbf{w}} + \frac{\partial \mathcal{L}}{\partial \overline{g(\mathbf{y})}} \frac{\partial \overline{g(\mathbf{y})}}{\partial g(\mathbf{y})} \frac{\partial g(\mathbf{y})}{\partial \mathbf{w}}. \end{aligned} \tag{9}$$

We apply the Nesterov momentum version of mini-batch SGD to optimize our model. Related methods have shown that mini-batch learning of distribution parameters (in detail, mean vectors and covariance matrices) is feasible if the batch size is adequately large [33], [34].

With the optimized parameters $\phi_* = \{\mathbf{w}_*, \mathbf{a}_*\}$, we can obtain b_* and α^* respectively, as follows,

$$b_* = \mathbf{a}_*^\top \overline{g(\mathbf{x})} - \frac{\sqrt{\mathbf{a}_*^\top \Sigma_{g(\mathbf{x})} \mathbf{a}_*}}{\sqrt{\mathbf{a}_*^\top \Sigma_{g(\mathbf{x})} \mathbf{a}_*} + \sqrt{\mathbf{a}_*^\top \Sigma_{g(\mathbf{w})} \mathbf{a}_*}}. \tag{10}$$

and

$$\alpha^* = \frac{1}{(\sqrt{\mathbf{a}^{*\top} \Sigma_{g(\mathbf{x})} \mathbf{a}^*} + \sqrt{\mathbf{a}^{*\top} \Sigma_{g(\mathbf{y})} \mathbf{a}^*})^2 + 1}. \tag{11}$$

α^* represents the worst-case accuracy of the future data. In general, traditional machine learning is fully data-driven, with the goal of maximizing the accuracy of the known data in the average sense, while our model is to maximize the accuracy of the future data in the worst sense, which is more robust.

The label of a test instance can be given by $\text{sign}(\mathbf{a}_*^\top g(\mathbf{z}, \mathbf{w}_*) - b_*)$: if the formulation is +1, $\mathbf{z}$ is classified as the class $\mathbf{x}$, otherwise $\mathbf{z}$ is classified as the class $\mathbf{y}$.

For multi-class classification, one can train a neural network with shared feature extraction and an integrated loss function via the one-vs-others or one-vs-one schemes, and then compare the α values for each class. Since the goal of this paper is to provide a seminal work of extending DNNs with MPM to evaluate its robustness, we leave this extension as the future work.

IV. Experiment

We evaluate our model across multiple scenarios on two datasets MNIST [35] and CIFAR-10 [36] respectively. Specifically, we evaluate the performance of DeepMPM on classification tasks and compare the accuracy of two datasets in the first place. Then we examine the robustness of our model under the attacks of adversarial examples.

All experiments are performed on a Linux machine with eight 4.0GHz CPU cores and 32GB RAM, one GTX 1080 Ti GPU card with 12GB RAM. We implemented DNNs based on Pytorch, a general deep learning platform.

A. Classification

In this subsection, we report the performance of our model on binary classification tasks. In particular, we test our model on MNIST and CIFAR-10 datasets. As MPM is designed to

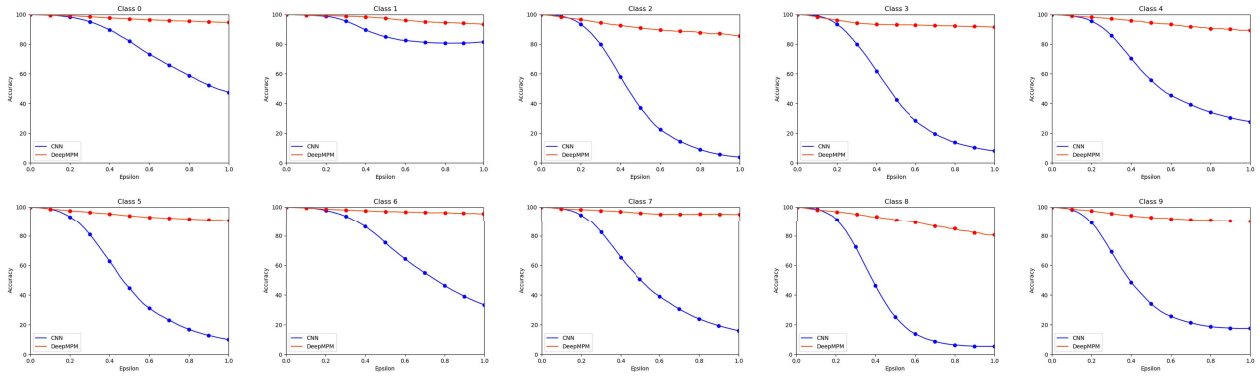

Fig. 2. The accuracy of ten binary classifiers on MNIST for FGSM attacks. The horizontal axis represents the size of ϵ. It's seen that the Accuracy of the DeepMPM decreases much more slowly with the size of adversarial perturbation for all ten classifiers. Thus, DeepMPM is more robust.

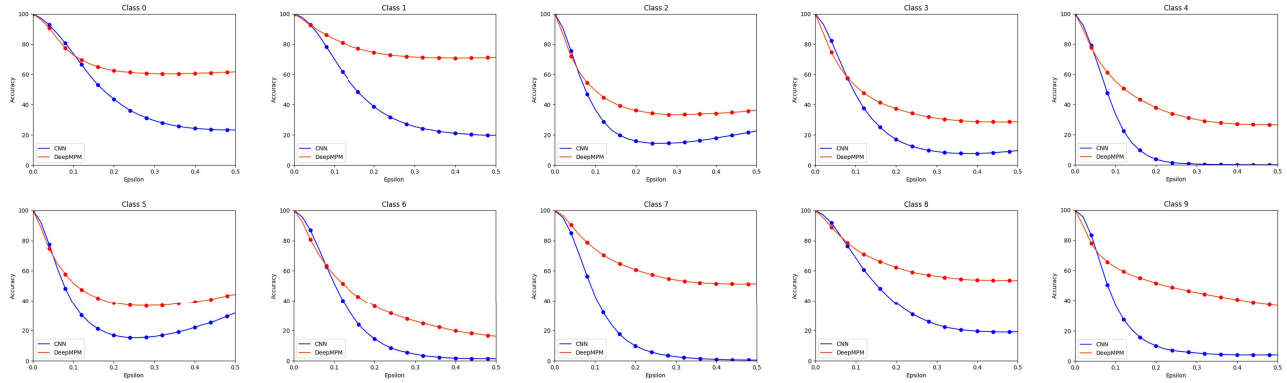

Fig. 3. The accuracy of ten binary classifiers from the FGSM attacks on CIFAR-10. DeepMPM's accuracy decreases significantly slower with size of adversarial perturbation.

solve binary classification problems, we train ten binary classifiers with CNN and DeepMPM. For each binary classifier, we use the one-vs-others approach instead of the one-vs-one approach.

For the experiments on MNIST, we exploit the network architecture in the Pytorch basic MNIST example[1] as the base CNN, which contains two convolutional layers and two fully connected layers, as shown in Table I.

CNN	DeepMPM
Conv. (5 by 5, 10 channels)	
Max pooling (2 by 2, padding is SAME)	
Conv. (5 by 5, 20 channels)	
Dropout	
Max pooling (2 by 2, padding is SAME)	
FC (to 50 units)	
FC (to 2 units)	MPM (to 1 unit)
Softmax	

TABLE I
THE CNN AND DEEPMPM ARCHITECTURES WE USE ON MNIST. THE FC IS THE ABBREVIATION OF THE FULLY CONNECTED LAYER.

[1]available at https://github.com/pytorch/examples/tree/0.4/mnist

While CIFAR-10 contains images of more complicated objects, we employ a deeper network. The base CNN architecture we use comes from [26] containing four convolutional layers and two fully connected layers, as shown in Table II.

CNN	DeepMPM
Conv. (3 by 3, 64 channels)	
Conv. (3 by 3, 64 channels)	
Max pooling (2 by 2, padding is SAME)	
Conv. (3 by 3, 128 channels)	
Conv. (3 by 3, 128 channels)	
Max pooling (2 by 2, padding is SAME	
FC (to 256 units)	
FC (to 2 units)	MPM (to 1 unit)
Softmax	

TABLE II
THE CNN AND DEEPMPM ARCHITECTURES WE USE ON CIFAR-10.

The same training set is used for all the experiments. Particularly, each model is trained end-to-end for 100 epochs, and the learning rate decay would be performed on the 50th and 80th epoch, with the decay factor as 0.1. Hyper-parameters for CNN models and MPM models are set the same except for the learning rate and momentum. We choose the small learning

rate and momentum empirically, since greater learning rates and larger momentums would produce Inf in gradient. Namely, the learning rate and momentum are set to $1e-2$ and 0.9 respectively for CNN models, while they are set to $1e-3$ and 0.5 for DeepMPM models.

The accuracy of two comparison models on 10 tasks of MNIST and CIFAR-10 are reported in Table III. DeepMPM performs better than CNN on 4 out of 10 tasks for MNIST, and the largest gap between our model and CNN on other 6 tasks is 0.11%. As for tasks on CIFAR-10, the average gap between the accuracy of the two models is 0.31%. From the results above, we could observe that our model could achieve comparable performance with CNN (the state-of-art method) on the ordinary classification tasks.

	MNIST		CIFAR-10	
	CNN	DeepMPM	CNN	DeepMPM
0	99.86	99.78	95.95	95.55
1	99.83	99.86	97.71	97.13
2	99.76	99.69	93.95	94.10
3	99.80	99.83	92.42	92.08
4	99.87	99.80	94.55	94.37
5	99.82	99.83	94.84	94.59
6	99.77	99.66	96.31	95.81
7	99.72	99.70	96.48	96.18
8	99.84	99.68	97.35	97.15
9	99.57	99.60	97.43	96.96

TABLE III
THE ACCURACY OF THE TEN BINARY CLASSIFICATION TASKS OF MNIST AND CIFAR-10 (%). IT SHOWS THAT THE DEEPMPM MODEL CAN ACHIEVE COMPARABLE PERFORMANCE WITH THE CNN MODEL.

B. Robustness to adversarial examples

There is a large amount of literature on producing adversarial examples with different methods [7], [8], [37], [38] since [6] pointed out neural networks are always vulnerable to adversarial examples. Adversarial examples are generated data which have minor differences from original data yet can mislead the deep networks to give the incorrect prediction. Adversarial attacks can be divided into two categories, targeted attacks and non targeted attacks respectively. Targeted attacks are those trying to find a minimal permutation that leads the classifier to the desired class prediction (target), while non-targeted attacks only care about the case that the classifier gives a different answer. In this paper, we focus on non-targeted attacks. It is noted that non-targeted could actually be viewed as targeted if classification tasks are all binary.

To validate the robustness of our model, we test the base CNN and our model on two attacks: the Fast Gradient Sign Method (FGSM) [7] and the L2 optimization attack of Carlini and Wagner [38].

1) The fast gradient sign method: The fast gradient sign method (FGSM) [7] is a one shot method for generating adversarial examples and can be considered as one special case of generalized gradient regularized method later developed in [8]. It tries to find the perturbation through the direction of the sign function of the gradient, which can be calculated by back-propagation. For a given image $\mathbf{x}$, the perturbation can be written as,

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x})), \tag{12}$$

where $J(\boldsymbol{\theta}, \mathbf{x})$ represents the model loss function with model parameters, ϵ is a small scalar value that controls the magnitude of the perturbation and $\operatorname{sign}(\cdot)$ denotes the sign function. Then the adversarial example $\mathbf{x}'$ is computed as: $\mathbf{x}' = \mathbf{x} + \boldsymbol{\eta}$.

For deep convolutional neural networks, the loss function is given as cross-entropy, namely the likelihood of the target class label. For our proposed DeepMPM model, the loss function is Equation (8) from the original MPM model. Attacks are performed on the models trained for classification tasks mentioned in the previous section.

To evaluate the robustness of our proposed model, we report the accuracy of classifiers on test sets of MNIST and CIFAR-10, with the magnitude factor ϵ ranges from small to large. For the MNIST experiment, ϵ ranges from 0 to 1, with step size 0.025. Meanwhile, in the CIFAR-10 experiment, we set the value of ϵ's from 0 to 0.5, and gradually increase it by 0.02.

		MNIST		CIFAR-10	
		CNN	DeepMPM	CNN	DeepMPM
0	CNN	0.00	94.09	0.00	89.71
	DeepMPM	99.99	89.50	98.16	0.00
1	CNN	0.00	99.66	0.00	95.35
	DeepMPM	99.47	88.00	98.73	0.05
2	CNN	0.01	97.96	0.00	77.81
	DeepMPM	99.98	78.95	96.15	0.01
3	CNN	0.00	98.74	0.00	77.81
	DeepMPM	99.95	86.64	97.58	0.00
4	CNN	0.13	98.50	0.00	85.65
	DeepMPM	99.91	86.90	97.83	0.00
5	CNN	0.00	98.88	0.00	89.48
	DeepMPM	99.93	89.00	97.28	0.00
6	CNN	0.00	95.25	0.00	87.19
	DeepMPM	99.97	89.93	97.96	0.01
7	CNN	0.00	99.65	0.00	95.40
	DeepMPM	99.92	88.59	94.17	0.11
8	CNN	0.00	91.00	0.00	89.86
	DeepMPM	99.89	88.14	98.40	0.03
9	CNN	0.00	97.85	0.00	66.30
	DeepMPM	99.89	87.93	97.77	0.22

TABLE IV
THE ACCURACY OF THE TEN BINARY CLASSIFICATION TASKS OF MNIST AND CIFAR-10 OVER L2 C&W ATTACK EXAMPLES(%).

We report the accuracy of the model under self-attack scenario in the first place, in Figure 2 (MNIST), and Figure 3 (CIFAR-10). Note that attacks are performed among examples which are originally classified correctly. For the sake of fairness, only examples that are correctly classified by both models are tested. Therefore, at the starting point where ϵ is equal to 0, the accuracy of both models are 100%. For MNIST, in Figure 2, it can be observed that the accuracy

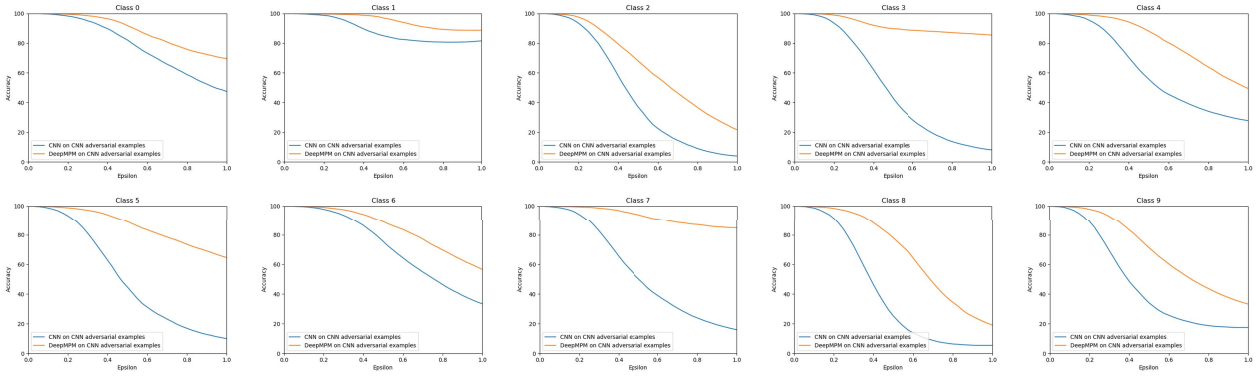

Fig. 4. The accuracy of both models on MNIST when applying both models in the adversarial examples generated by CNN for the FGSM attacks (%). With the adversarial examples generated by the CNN, although the DeepMPM performs worse than the case when applied to its own adversarial examples, its accuracy is much higher than that of the CNN.

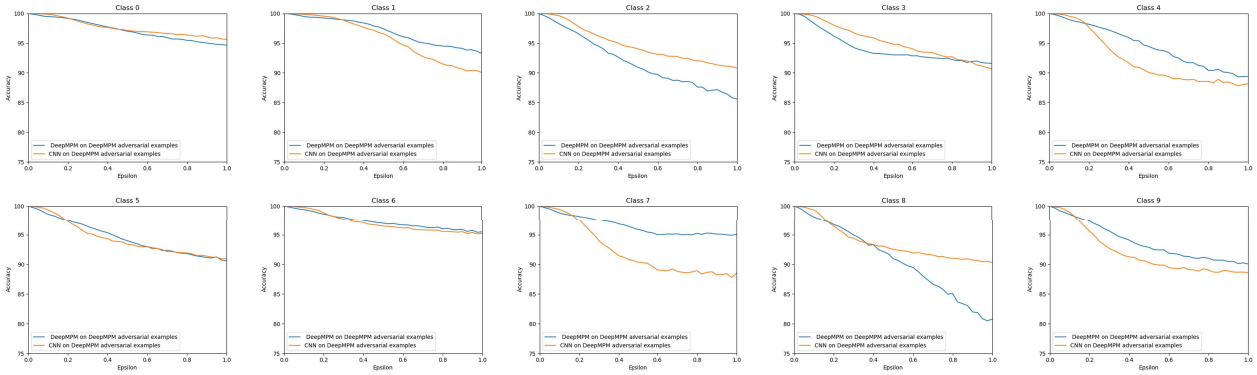

Fig. 5. The accuracy of both models on MNIST when applying both models in the adversarial examples generated by the DeepMPM for the FGSM attacks (%). In order to see the change more clearly, the value of the vertical axis starts from 75%. When attacking the CNN model on adversarial examples generated by the DeepMPM, the CNN has a great prediction on several binary classifications.

of DeepMPM remains high while that of the CNN model reduces significantly as ϵ increases. As for the harder CIFAR-10 tasks, in Figure 3, though the accuracy decreases are observed, DeepMPM still generates much better predictions than CNN. The above results demonstrate the great robustness of DeepMPM under FGSM attacks in the self-attack scenarios.

To illustrate the performance of the FSGM attack examples transferred between different models, we plot the accuracy of classifiers when both models are applied to adversarial examples generated by the CNN model, as shown in Figure 4. In this paper, We mainly present the experimental results on the MNIST dataset. For the experiments on MNIST we find that in most classifiers, although the DeepMPM model performs worse than the case when applied to its own adversarial examples, its accuracy is observed much higher than the accuracy of CNN. This indicates that the performance of DeepMPM is better than that of CNN.

We also show the accuracy of classifiers when two models are applied to adversarial examples generated by the DeepMPM model, shown in Figure 5. Similarly, We mainly present the experimental results on the MNIST dataset. The accuracy of CNN has increased a lot compared to applying to its own adversarial examples. And the accuracy of the two models is almost the same.

From the Figure 4 and Figure 5, it can be seen that the attack examples generated by the DeepMPM model are less aggressive, while the attack examples generated by the CNN model are universal.

2) L2 optimization attack of Carlini and Wagner: To further validate the robustness of our model, we perform attacks with the L2 optimization attack of Carlini and Wagner [38]. This method finds the perturbation by minimizing the following function,

$$\|\boldsymbol{\eta}\|_2^2 + cf(\mathbf{x}+\boldsymbol{\eta}). \tag{13}$$

where c is a constant chosen by the binary search method and $f(\cdot)$ we used is defined as below for non targeted attacks.

$$f(\mathbf{x}') = Z(\mathbf{x}')_l - \max\{Z(\mathbf{x}')_i : i \neq l\}. \tag{14}$$

$Z(\mathbf{x}')_i$ denotes the pre softmax predictions for the class i on image $\mathbf{x}'$, and l represents the correct class.

As no softmax function is carried out in our model, this attack might fail in generating adversarial examples directly from pre-softmax values like in CNN. It is known that we get the prediction by the value of $\mathbf{a}_*^\top g(\mathbf{z}, \mathbf{w}_*) - b_*$: if the value is greater than or equal to 0, the new point $\mathbf{z}$ is classified as the class $\mathbf{x}$, otherwise $\mathbf{z}$ is classified as the class $\mathbf{y}$. We found that the value of the equation $S(\mathbf{a}_*^\top g(\mathbf{z}, \mathbf{w}_*) - b_*)$ could represent the probability that a sample belongs to class $\mathbf{x}$ to some extent, where $S(\cdot)$ denotes the sigmoid function, and the probability that a sample belongs to class $\mathbf{y}$ can be obtained by the value of the equation $1 - S(\mathbf{a}_*^\top g(\mathbf{z}, \mathbf{w}_*) - b_*)$. Therefore, we reversely calculate the pre-softmax values with the probabilities, which is the same as done in [11]. Same as the implementation of [38], we actually optimize η in a transformed space.

Different from the FGSM attack, L2 optimization adversarial attack is an iterative attack which optimizes over an example for multiple times. Thus the optimized adversarial examples can easily reach rather a high attack success rate than FGSM generated examples.

In Table IV, we report the accuracy of models over the generated attack examples of L2 optimization of Carlini and Wagner. The row represents the targeted model of attacks while the column represents the model be tested on. As can be observed, DeepMPM can recognize attack examples with high confidence (87.36% averagely) under self-attack scenario for MNIST while CNN got nearly 0% accuracy. For Cifar-10, both models are perfectly attacked under self-attack examples, yet DeepMPM's accuracy suffers more on transfer attacks where examples of one model are tested on the other. (It's partly due to the universality of CNN targeted examples.)

We also report the differences in magnitudes of perturbation needed between DeepMPM and CNN, calculated by $L2_{MPM} - L2_{CNN}$ where $L2_{MPM}$ represents the distance between DeepMPM attack example and original data, and $L2_{CNN}$ as well. We only show the 4 classes of self-attacks on Cifar-10, as shown in Figure 6.

From Figure 6, we could see magnitudes of perturbation needed for DeepMPM is rather larger than that of CNN, meaning the difficulty of attacking DeepMPM is greater, proving the robustness of our model.

V. Conclusion

In order to alleviate the vulnerability of deep neural networks to adversarial attacks, we have proposed the Deep Minimax Probability Machine (DeepMPM), applying MPM to DNNs in an end-to-end fashion. Specifically, we put MPM on top of a deep neural network, and instead of maximizing the likelihood of labels for data, we employ the objective function of MPM. DeepMPM is more robust intuitively since it takes the global information into account and learns the worst-case bound on the probability of misclassification of future data. In practice, to evaluate the robustness of our proposed model, two groups of tasks are performed on our DeepMPM including classification and adversarial attacks on MNIST and Cifar-10. Experimental results on classification show DeepMPM have a competitive performance with CNN. The robustness of our model has also been fully demonstrated in two attacks, the FSGM and the L2 optimization attack of C&W.

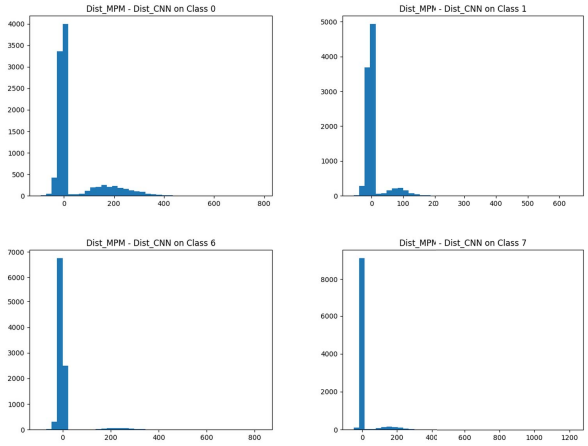

Fig. 6. The differences in magnitudes of perturbation needed between the DeepMPM and the CNN.

Acknowledgments

This paper was in part supported by Grants from the Natural Science Foundation of China(No. 61572111), and two Fundamental Research Funds for the Central Universities of China (Nos.ZYGX2016Z003, ZYGX2017KYQD177),

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] G. Saon, H.-K. J. Kuo, S. Rennie, and M. Picheny, "The ibm 2015 english conversational telephone speech recognition system," *arXiv preprint arXiv:1505.05899*, 2015.

[4] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1700–1709.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples (2014)," *arXiv preprint arXiv:1412.6572*.

[8] C. Lyu, K. Huang, and H.-N. Liang, "A unified gradient regularization family for adversarial examples," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 301–309.

[9] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[10] A. V. Maharaj, "Improving the adversarial robustness of convnets by reduction of input dimensionality," 2015.

[11] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani, "Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks," *arXiv preprint arXiv:1707.02476*, 2017.

[12] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[13] G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan, "Minimax probability machine," in *Advances in neural information processing systems*, 2002, pp. 801–807.

[14] K. Huang, H. Yang, I. King, M. R. Lyu, and L. Chan, "The minimum error minimax probability machine," *Journal of Machine Learning Research*, vol. 5, no. Oct, pp. 1253–1286, 2004.

[15] K. Huang, H. Yang, I. King, and M. R. Lyu, "Maxi–min margin machine: learning large margin classifiers locally and globally," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 260–272, 2008.

[16] ——, "Imbalanced learning with a biased minimax probability machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 4, pp. 913–923, 2006.

[17] Z. Xu, I. King, and M. R. Lyu, "Feature selection based on minimum error minimax probability machine," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 08, pp. 1279–1292, 2007.

[18] K. Isii, "On sharpness of tchebycheff-type inequalities," *Annals of the Institute of Statistical Mathematics*, vol. 14, no. 1, pp. 185–197, Dec 1962.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[20] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *arXiv preprint arXiv:1511.03034*, 2015.

[21] J. Ye, L. Wang, G. Li, D. Chen, S. Zhe, X. Chu, and Z. Xu, "Learning compact recurrent neural networks with block-term tensor decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9378–9387.

[22] Y. Pan, J. Xu, M. Wang, J. Ye, F. Wang, K. Bai, and Z. Xu, "Compressing recurrent neural networks with tensor ring for action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4683–4690.

[23] H. Liu, L. He, H. Bai, B. Dai, K. Bai, and Z. Xu, "Structured inference for recurrent hidden semi-markov model." in *IJCAI*, 2018, pp. 2447–2453.

[24] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[26] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "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.

[27] N. Papernot and P. McDaniel, "Extending defensive distillation," *arXiv preprint arXiv:1705.05264*, 2017.

[28] 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 Conference on Artificial Intelligence*, 2018.

[29] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[30] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse dnns with improved adversarial robustness," in *Advances in Neural Information Processing Systems*, 2018, pp. 240–249.

[31] D. Bertsimas and J. Sethuraman, "Moment problems and semidefinite optimization," in *Handbook of semidefinite programming*. Springer, 2000, pp. 469–509.

[32] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. Siam, 1994, vol. 13.

[33] W. Wang, R. Arora, K. Livescu, and J. Bilmes, "On deep multi-view representation learning," in *International Conference on Machine Learning*, 2015, pp. 1083–1092.

[34] W. Wang, R. Arora, K. Livescu, and J. A. Bilmes, "Unsupervised learning of acoustic features via deep canonical correlation analysis," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4590–4594.

[35] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[37] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[38] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.