Is ReLU Adversarially Robust?

Korn Sooksatra [1] **Greg Hamerly** [1] **Pablo Rivas** [1]

Abstract

The efficacy of deep learning models has been called into question by the presence of adversarial examples. Addressing the vulnerability of deep learning models to adversarial examples is crucial for ensuring their continued development and deployment. In this work, we focus on the role of rectified linear unit (ReLU) activation functions in the generation of adversarial examples. ReLU functions are commonly used in deep learning models because they facilitate the training process. However, our empirical analysis demonstrates that ReLU functions are not robust against adversarial examples. We propose a modified version of the ReLU function, which improves robustness against adversarial examples. Our results are supported by an experiment, which confirms the effectiveness of our proposed modification. Additionally, we demonstrate that applying adversarial training to our customized model further enhances its robustness compared to a general model.

1. Introduction

One of deep learning models' significant challenges is their vulnerability to tiny, imperceptible perturbations embedded in their input, known as adversarial examples. An attacker (Goodfellow et al., 2014; Madry et al., 2017; Sooksatra & Rivas, 2022; 2021; Kurakin et al., 2016; Papernot et al., 2016b;a; 2017; Ilyas et al., 2018; Gong et al., 2017; Kos et al., 2018; Kurakin et al., 2018; Xiao et al., 2018; Hendrycks et al., 2021; Luo et al., 2018; Zhao et al., 2017; Croce & Hein, 2020) explicitly crafts these perturbations to cause the model to make a wrong prediction.

[*]Equal contribution [1]Department of Computer Science, Baylor University, Waco, TX. Correspondence to: Korn Sooksatra <korn_sooksatra1@baylor.edu>, Greg Hamerly <greg_hamerly@baylor.edu>, Pablo Rivas <pablo_rivas@baylor.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Several defense mechanisms (Goodfellow et al., 2014; Madry et al., 2017; Wong et al., 2020; Tramèr et al., 2017; Zhang et al., 2019; Wang et al., 2019; Shafahi et al., 2019; Rakin et al., 2018; Lin et al., 2019) have been proposed to address this issue, including adversarial training, input preprocessing, and network architecture modification. However, few have focused on fixing the general activation functions (such as Sigmoid, Tanh, and ReLU), which cause adversarial examples. ReLU activation functions are widely used in deep-learning models because they accelerate the training process and address the vanishing gradient problem. However, they also make the models vulnerable to adversarial examples, allowing many tiny perturbations to enlarge themselves over multiple layers in the models.

In this work, inspired by ReLU6 activation functions (Sandler et al., 2018), we propose a solution by experimenting with and customizing ReLU activation functions by capping them. Our results indicate that the models become more robust against adversarial examples when their max values are set. Additionally, we found that the models are even more robust when the max values are decreased. However, this technique is limited to small-scale datasets (such as MNIST (Deng, 2012)) as it does not address the vanishing gradient problem for medium and large-scale datasets (such as CIFAR10 (Krizhevsky et al., 2009) and Imagenet (Deng et al., 2009)), which require additional techniques. Our work aims to provide a new perspective on improving the robustness of deep-learning models and making them more trustworthy in practical scenarios.

This paper is organized as follows: Section 2 explores and describes the works that focus on improving adversarial robustness; Section 3 shows that the problem of ReLU functions and describes one way to customize them to mitigate the problem; Section 4 demonstrates how the size of a layer with customized ReLU functions affects the robustness; Section 5 demonstrates how the order of a layer with customized ReLU functions affects the robustness; Section 6 explains the experiment to support our claim regarding our customized functions and demonstrates its result; Section 7 discusses how to compute sensitivity map of an image to show that capping ReLU functions leads to robustness; Section 8 shows the effect of adversarial training on a model with our customized functions compared to the one with general functions; Section 9 concludes our work and describes

the limitation of our solution and future works.

2. Related Works

In recent years, a significant amount of research has been dedicated to addressing the vulnerability of deep-learning models to adversarial examples. A wide range of defensive mechanisms has been proposed to promote the robustness of these models, which can broadly be categorized into training and architectural solutions.

The training solution approach aims to improve the robustness of deep-learning models by proposing alternative ways to train a classifier. This can include techniques such as adversarial training, in which the model is trained on adversarial examples to improve its robustness against them.

On the other hand, the architectural solution seeks to alter various parts of the classifier's network to increase its robustness. This can include techniques such as network architecture modification and activation function customization. In this section, we will discuss the existing works within these two categories of solutions, providing an in-depth understanding of the current state of the art in adversarial robustness.

2.1. Adversarial Training

Goodfellow et al. (2014) proposed an adversarial robustness evaluation called Fast Gradient Sign Method (FGSM). Then, they augmented the adversarial examples from FGSM to the training dataset and retrained a classifier. As a result, the classifier was robust against FGSM. Later, in 2017, Madry et al. (2017) suggested using Projected Gradient Descent (PGD) (Madry et al., 2017) to estimate the inner maximization of adversarial training instead of FGSM because PGD was a stronger attack than FGSM. Also, a classifier retrained with adversarial examples from PGD could be robust against L_∞ attacks (e.g., FGSM and PGD) and L_2 attacks (e.g., Carlini and Wagner attack (Carlini & Wagner, 2017)). Tramèr et al. (2017) mentioned that the adversarial training relying on adversarial examples from one classifier does not provide robustness against black-box attacks. Therefore, they used adversarial examples from several classifiers to retrain a classifier to improve its robustness against black-box attacks. In 2019, Zhang et al. (2019) split the objective function for the training process into two terms. The first term was for accuracy, and the second was for robustness. Hence, their training scheme could find the tradeoff between accuracy and robustness. Wang et al. (2019) improved the adversarial training by emphasizing misclassified training samples. This method led to a more robust classifier than the one trained with adversarial training that did not differentiate the misclassified and correctly classified examples. Furthermore, Shafahi et al. (2019) utilized the gradients from the natural training to compute adversarial perturbations for adversarial training. However, these perturbations could be used only for FGSM. Hence, to make a classifier as robust as using PGD, they needed to train the classifier for much more epochs than general adversarial training. Therefore, in 2020, Wong et al. (2020) showed that FGSM was good enough to make a classifier robust; thus, we did not need to train the classifier for as many epochs as the work in (Shafahi et al., 2019) did.

2.2. Architectural solution

Only a few existing works proposed solutions by altering a classifier's architecture. In 2018, Rakin et al. (2018) quantized activation functions in a classifier to eliminate adversarial perturbations. They also showed that dynamic quantization could make the classifier more robust than fixing the thresholds of quantization. Later, in 2019, Lin et al. (2019) improved this approach by adding a regularization term that could indicate the Lipschitz constant. Nonetheless, minimizing the constant was intractable; hence, the term made the covariance matrix of each layer's weight close to the identity matrix. However, these quantization approaches negatively affected the training process due to the vanishing gradient problem.

To the best of our knowledge, our work is the first that customizes the activation functions for robustness and reduces the negative impact on the training process. For example, a previous work quantizes activation functions in a model to improve its robustness; however, this technique leaves only zero gradients in the activation functions and harms a gradient-based training process, such as gradient descent. Our work instead leaves some areas of the functions to be differentiable.

3. Flaws and Modification of ReLU Functions

ReLU activation functions have been widely used in deep-learning models due to their ability to accelerate the training process and address the vanishing gradient problem. Unlike Sigmoid and Tanh activation functions, ReLU activation functions have many spaces for gradient computation, making them more friendly to backpropagation. However, this property that makes ReLU functions worthwhile makes them weak in deep-learning models regarding adversarial examples. Because ReLU functions allow many tiny perturbations in inputs to enlarge themselves over the hidden layers, these tiny perturbations can result in a significant difference in the output layer, making the model vulnerable to adversarial examples.

3.1. Enlarged Perturbations

Fortunately, we found that capping ReLU activation functions can stop the perturbations from growing over the layers. Therefore, we construct an experiment to show the growing perturbations in the hidden layers for various max values. We use the MNIST dataset (Deng, 2012) and train its classifier consisting of three dense hidden layers whose sizes are 392, 196, and 98. After that, we utilize Projected Gradient Descent (PGD) attack on the classifier and the test dataset with the perturbation bound of $20/256$, step size of $2/256$ and the max iteration of 20. Then, we obtain adversarial examples. Next, we train other classifiers and cap different hidden layers (i.e., the first hidden layer (HL1), the second hidden layer (HL2), the third hidden layer (HL3) and all the hidden layers (HL123)). Also, we cap them with diverse max values (i.e., 0.01, 0.1, 1, 10, and 100). Figure 1 demonstrates that ReLU functions with high max values allow perturbations to become huge over the layers. On the other hand, capping them with low values can mitigate such an effect. Further, it is intuitive that the difference significantly goes down at the layer capped, as seen in the figure.

Although capping ReLU functions reduces the growth of perturbed values that may significantly alter the output, we found that when we set the max value to be very low, the classifier would underfit the dataset due to the vanishing gradient problem, as demonstrated in Figure 2. Also, capping all the hidden layers achieved a slightly lower performance than capping only one hidden layer, as seen in the figure when the max value is 0.1. Therefore, in this phenomenon, there is a tradeoff between the network's ability to be trained and its sensitivity to tiny perturbations.

3.2. Capped ReLU Function

We show that the capped ReLU function can effectively control the enlarged perturbations. This section shows the formal definition of the function.

A capped ReLU function is a general ReLU function capped with a value. Hence, we can formulate this function as

$$a(z, \beta) = \max(0, \min(z, \beta)) \tag{1}$$

where z is the function's input and β is a max value that caps the function. As seen in Figure 1a, reducing β can control the growing perturbations efficiently.

3.3. Sigmoid and Tanh Activation

Sigmoid and Tanh activation functions can be good candidates to provide adversarial robustness since their values have the highest and lowest values. However, the spaces from their minimum to maximum values are too wide to stop adversarial perturbations from enlarging themselves over layers. One way to narrow the gap is by multiplying their arguments with some constants. For example, let $\sigma(z)$ be a Sigmoid or Tanh function and z be the result after a layer before this activation. We can narrow it by multiplying its argument with a real value c, where $c > 1$. Hence, the function becomes $\sigma(c \cdot z)$. Figure 3 shows that increasing c can narrow the function.

Nevertheless, the parameters of the corresponding layer are a part of the computation of z. Thus, the function can become wide when the training reduces those parameters. Then, Sigmoid or Tanh activation functions cannot be used to eliminate adversarial perturbations, unlike ReLU functions.

4. Effect of Capped Layer's Size on Robustness

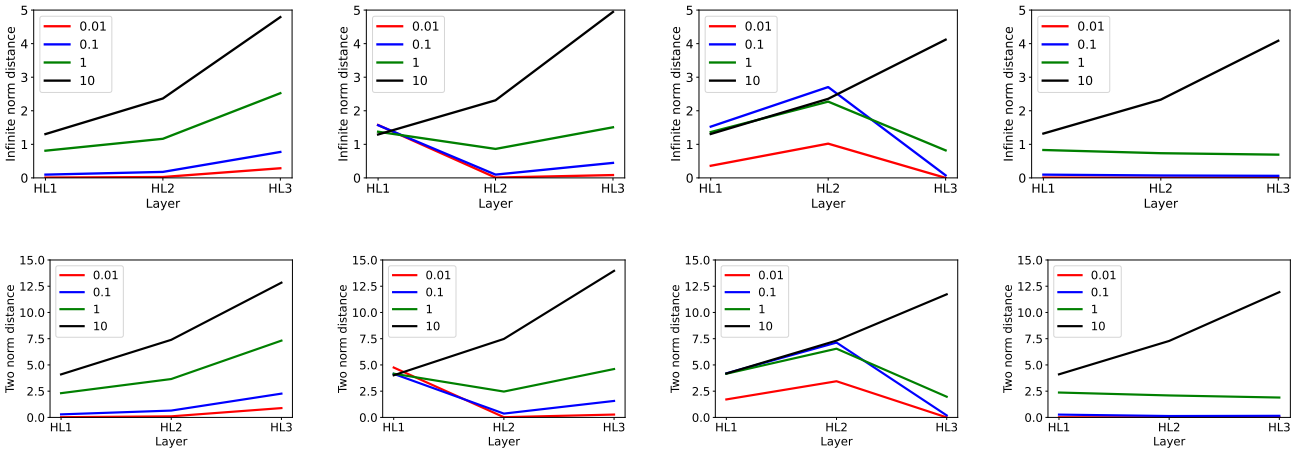
In this section, we want to know whether the width of the layer (narrow or wide) containing the capped neurons has an effect on robustness. Therefore, we conducted the following experiment.

4.1. Experimental Explanation and Setting

First, we trained a classifier with some layers capped with an initial max value to control the value after the ReLU function. Then, we evaluated this classifier in terms of accuracy on clean test samples (i.e., standard accuracy) and adversarial examples (i.e., robust accuracy) and the success rate of an attack. However, we cannot rely only on the initial max value because it may not make the classifier the most robust. Therefore, we evaluate this classifier with several max values (i.e., from 0.01 to 0.15 in our experiments) to determine the max value that promotes robustness and does not sacrifice much standard accuracy. We used the MNIST dataset for this experiment. Also, we created two kinds of classifiers: a "general" two-hidden-layer dense network and a "reversed" two-hidden-layer dense network. The former consists of an input layer, a 392-neuron layer with ReLU activation, a 196-neuron layer with ReLU activation, and the output layer with Softmax activation. In the latter, only the hidden layers are swapped. Further, the attack we used for this experiment is Projected Gradient Descent (PGD) (Madry et al., 2017) because it is one of the strong attacks and is widely used for adversarial robustness evaluation.

4.2. Results

Figure 4 shows the result of this experiment with the general network. It demonstrates that with the initial max values of 0.01 and 0.1, capping the second hidden layer surprisingly outperforms the first hidden layer and capping both the hidden layers. However, with the initial max value of 1, capping the second hidden layer underperforms the others when



(a) Cap the first hidden layer. (b) Cap the second hidden layer. (c) Cap the third hidden layer. (d) Cap all the hidden layers.

Figure 1: The distance between each hidden layer's outputs resulted from passing clean samples and adversarial examples. Note that the top row shows the L_∞ distance and the bottom row shows the L_2 distance.

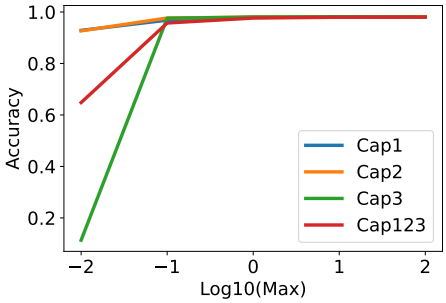


Figure 2: Accuracy achieved by classifiers with different capped hidden layers and max values on MNIST test dataset.

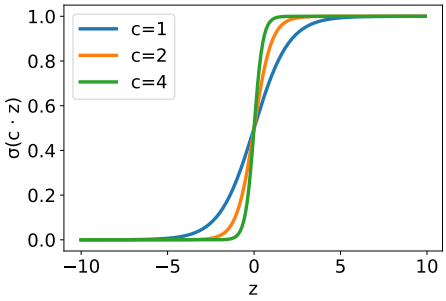


Figure 3: Sigmoid functions with different constant c.

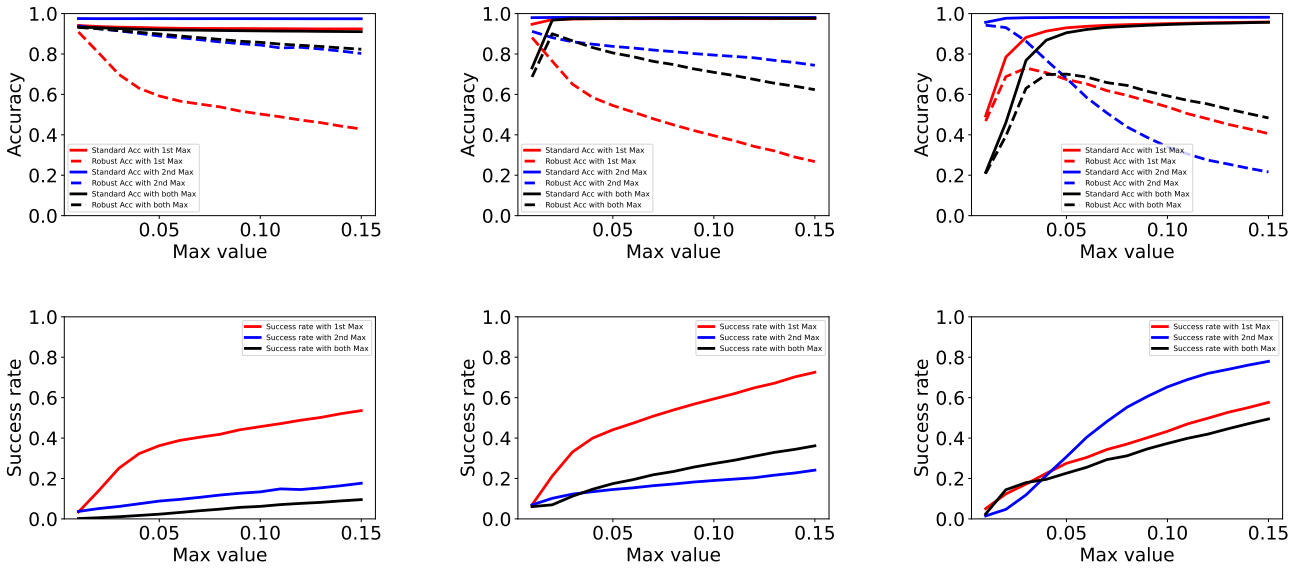
the max value is greater than 0.05 in terms of robustness. Nonetheless, the robustness of the classifier being capped at the second hidden layer with a max value less than 0.05 is higher than the others in all the max values. Therefore, capping the second hidden layer is the best solution for this general network. Because the second hidden layer has lower neurons than the first hidden layer, capping the small layer is better than capping the large layer. However, this result is derived from a specific network. Next, we experiment with the reversed network.

Figure 5 shows the results of the same experiment from the reversed network. Capping both the hidden layers outperforms the others in most cases. Also, capping the first hidden layer outperforms capping the second hidden layer in most cases. Since the first hidden layer contains fewer neurons than the second hidden layer, capping a small layer is better than capping a large layer in this case. We can summarize that capping a bottleneck layer would result in the most robustness.

5. Effect of Capped Layer's Order on Robustness

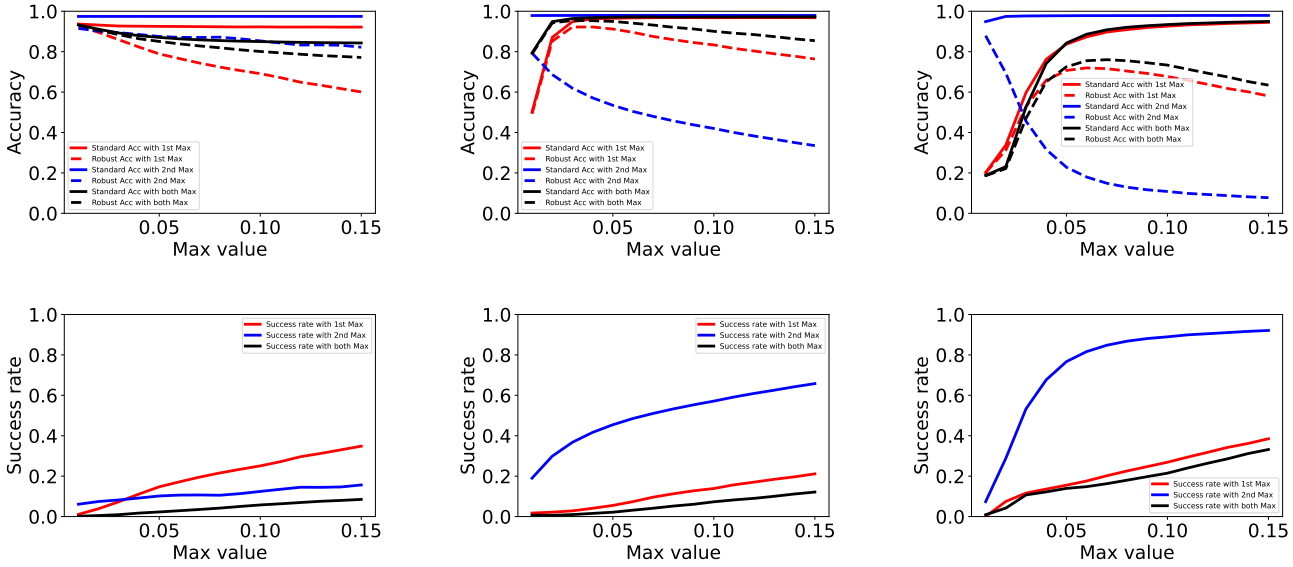
In this section, we want to know whether capping an early or deep layer in a classifier can provide the most robustness. We conducted the same experiment as in the previous section. However, we built another classifier consisting of the input layer, two 784-neuron hidden layers with ReLU activations, and the output layer with Softmax activation. Noticeably, the hidden layers' sizes are equal to see which layer affects the most in terms of robustness.

Figure 6 shows the results of this experiment. The classifier capped at the first hidden layer performs relatively better when the initial max value grows. Evidently, at the initial max value of 0.01, capping the second hidden layer is better than the first hidden layer. However, increasing the initial max value results in the opposite consequence. Therefore, capping the deep layer is recommended with a very low initial max value, and the early layer is preferred with a medium to high initial max value.



(a) Initial max = 0.01. (b) Initial max = 0.1. (c) Initial max = 1.

Figure 4: Standard accuracy, robust accuracy and success rate of the two-hidden-layer classifier according to PGD attack over a range of max values.



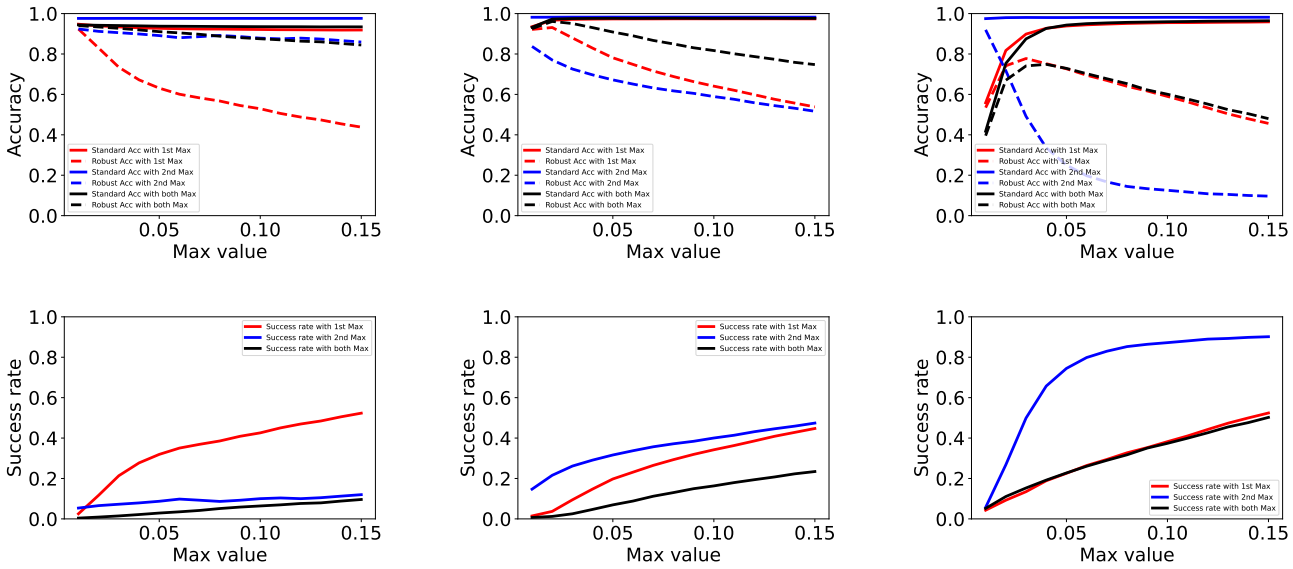
(a) Initial max = 0.01. (b) Initial max = 0.1. (c) Initial max = 1.

Figure 5: Standard accuracy, robust accuracy and success rate of the reversed two-hidden-layer classifier according to PGD attack over a range of max values.

6. Zero Gradient Experiment

In this experiment, we aim to provide further empirical evidence to support the validity of our previous results. To this end, we have modified the projected gradient descent (PGD) attack method to include a new stopping criterion, which we refer to as "zero gradients". Specifically, instead of terminating the attack once an adversarial example has been found, we continue the attack until the gradient of the objective function is zero. This modification allows us to assess the robustness of a targeted classifier in a more meaningful way.

We assume that a classifier is robust if the location where



(a) Initial max = 0.01. (b) Initial max = 0.1. (c) Initial max = 1.

Figure 6: Standard accuracy, robust accuracy and success rate of the equal two-hidden-layer classifier according to PGD attack over a range of max values.

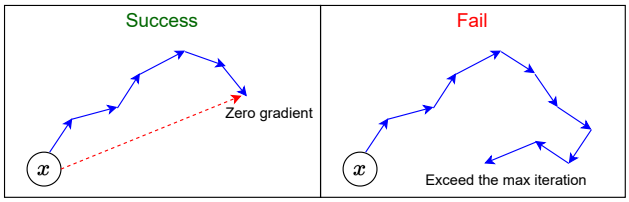


Figure 7: Examples of success and failure scenarios where a blue arrow is a gradient direction in each step of PGD attack, and the red dash arrow is the distance between sample x to the zero-gradient location.

the zero gradients are found is close to the original, clean input sample. This is because if an attacker encounters a zero gradient, they will no longer be able to perform any gradient-based attacks, such as the fast gradient sign method (FGSM) or PGD. To measure the distance between the clean sample and the location where the zero gradients are found, we use the Euclidean distance.

However, it should be noted that this method can fail if zero gradients are not found within the maximum number of iterations. Figure 7 illustrates examples of both successful and unsuccessful scenarios. To obtain a more comprehensive understanding of the robustness of the classifiers, we compute the average distance only from the test samples where zero gradients are found. We use the MNIST dataset, along with the classifiers from Section 4 and 5 for this experiment.

Figures 8, 9, and 10 show the results of the zero gradient experiment with the general, reversed and equal networks, respectively. Interestingly, we found that these results aligned with the previous experiments in Section 4 and 5. Therefore, we can conclude that the results in those sections are valid.

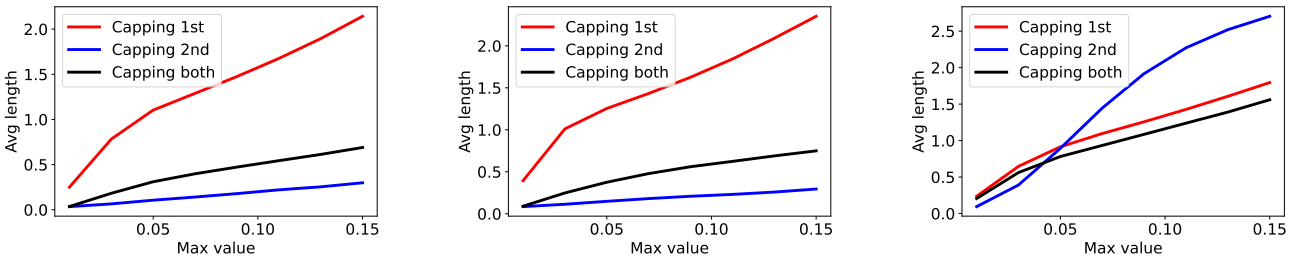
7. Capped-ReLU Classifier's Sensitivity Map

As discussed in (Sooksatra & Rivas, 2022), a pixel in an image vulnerable to adversarial attacks is sensitive to a slight change. It also proposed an equation to compute a sensitivity map to determine how much each pixel is susceptible to adversarial attacks. The equation is

$$\text{smap}(\boldsymbol{x}, Z) = \max\left(\mathbf{0}, \frac{\partial Z_t}{\partial \boldsymbol{x}} \cdot \sum_{c \neq t} \frac{\partial Z_c}{\partial \boldsymbol{x}}\right), \qquad (2)$$

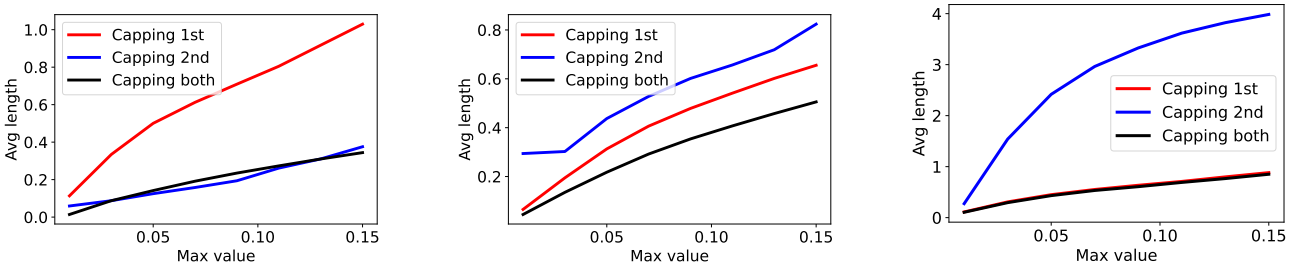
where $\boldsymbol{x}$ is an input, Z is a classifier whose output is before Softmax function, Z_i is the output of class i, t is the true class of $\boldsymbol{x}$ and $\mathbf{0}$ is a matrix of 0 whose size is the same as $\boldsymbol{x}$. We sum the map's values across all pixels to show that capping ReLU functions improves robustness.

We create a two-hidden-layer classifier and train it with several max values (i.e., 1, 0.1 and 0.01). Figure 11 shows the sensitivity map of digit five with the classifier. Essentially, the number of vulnerable pixels and the summation of the map decrease when the max value is reduced. Therefore, capping ReLU functions with low max values can improve the robustness.



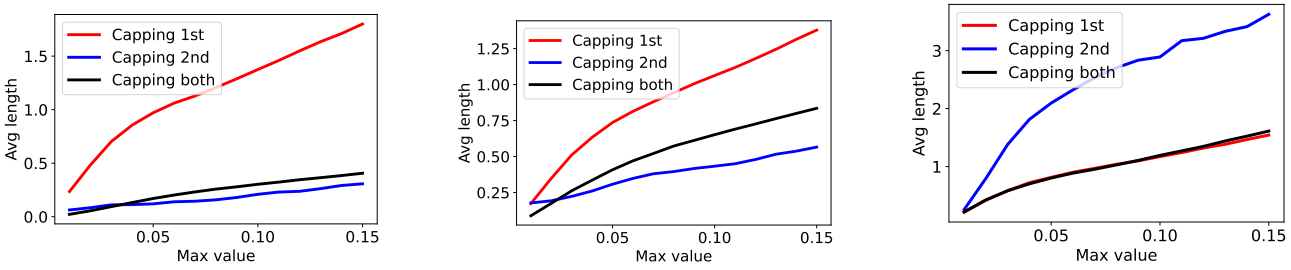
(a) Initial max = 0.01. (b) Initial max = 0.1. (c) Initial max = 1.

Figure 8: Average distance to zero gradients by PGD attack on a range of max values where the targets are general networks.



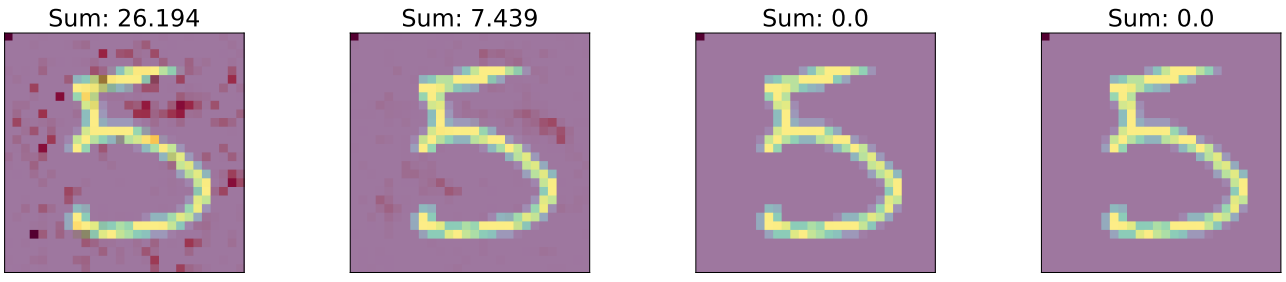
(a) Initial max = 0.01. (b) Initial max = 0.1. (c) Initial max = 1.

Figure 9: Average distance to zero gradients by PGD attack on a range of max values where the targets are reversed networks.



(a) Initial max = 0.01. (b) Initial max = 0.1. (c) Initial max = 1.

Figure 10: Average distance to zero gradients by PGD attack on a range of max values where the targets are equal networks.



(a) General ReLU. (b) Max val = 1. (c) Max val = 0.1. (d) Max val = 0.01.

Figure 11: Sensitivity map of digit five and the summation of the scores on the top. Note that the more red pixel is, the more sensitive pixel becomes. Also, the black pixel in the top left of the image is not included in the map. We use it as a maximum reference value to tune the value's range across all the images.

8. Capped ReLU with Adversarial Training

In this section, we aim to investigate the efficacy of applying adversarial training techniques to capped-ReLU classifiers to enhance their robustness beyond that of general classifiers that have undergone adversarial training. To accomplish this, we utilize two-hidden-layer neural networks as the base model and train them using clean test samples for a total of twenty epochs with the Adam optimizer as described in (Kingma & Ba, 2014) and a learning rate of 0.001. We only apply the ReLU function cap at the second hidden layer, as previous sections have demonstrated this to be the most effective location for such an operation.

Following the initial training phase, we then proceed to apply adversarial training to these networks through the use of either the Fast Gradient Sign Method (FGSM) or Projected Gradient Descent (PGD) for an additional ten epochs, with a perturbation bound of 0.1. Subsequently, we evaluate these networks' accuracy on clean test samples and samples that have been attacked using FGSM, PGD, and the Carlini and Wagner (CW) attack (Carlini & Wagner, 2017). For FGSM and PGD, we employ a perturbation bound of 0.1, a maximum iteration of 10 and a step size of 0.01. Additionally, in the case of the CW attack, we use a maximum iteration of 10000, a learning rate of 0.01, an initial balancing factor of 0.001, and 9 adjustments of the balancing factor.

Table 1: Accuracy of MNIST two-hidden-layer classifiers with general and capped ReLU activation functions on clean test samples and adversarial test samples generated by using FGSM, PGD and CW.

Max Val.	Adv. Training	Clean %	FGSM %	PGD %	CW(L_2) %
-	-	98.49	41.77	9.47	0.00
1.00	-	98.46	41.24	7.45	0.00
0.10	-	98.06	68.04	39.79	5.56
0.01	-	97.88	92.37	89.61	8.07
-	FGSM	98.26	91.44	85.12	0.19
1.00	FGSM	98.35	92.46	81.88	0.18
0.10	FGSM	98.18	93.00	90.37	3.50
0.01	FGSM	97.10	94.07	96.36	8.21
-	PGD	98.67	91.85	86.74	0.10
1.00	PGD	98.49	93.32	87.09	0.11
0.10	PGD	98.09	92.64	92.85	3.62
0.01	PGD	96.55	89.21	95.43	8.00

The configurations of the classifiers and their corresponding accuracy on both clean and adversarial test samples are presented in Table 1. The results reveal that by decreasing the maximum value, the robustness of the classifiers against attacks using FGSM, PGD, and CW can be improved without sacrificing a significant portion of standard accuracy. This is particularly evident when the models are retrained using FGSM, which results in similar performance and robustness to retraining using PGD, despite the latter taking much more time, as previously discussed in (Wong et al., 2020). Additionally, it is worth noting that although capping the ReLU function can improve robustness, the CW attack remains particularly effective, as it is not limited by any perturbation bound. Despite the success of the CW attack, we continue to see the trend that using a lower max value yields a more robust network. Therefore, a correctly customized classifier concerning its ReLU functions would ultimately be robust against CW. In this context, it is essential to note that static capping ReLU activation functions are a starting point for enhancing adversarial robustness by customizing architecture.

9. Conclusion

This work demonstrates that ReLU activation functions, despite their ability to increase the speed of the training process, are a significant contributor to adversarial attacks. Through experimentation, we have shown that small perturbations can rapidly grow over the layers in a classifier that utilizes general ReLU functions, ultimately leading to a change in the prediction at the output layer. However, by capping these functions with a maximum value, the growth of perturbations is controlled, improving the classifier's robustness. The zero gradient experiment and the sensitivity map support our findings. Additionally, we have established that applying adversarial training to classifiers that use capped ReLU functions can further improve robustness beyond that of classifiers that use general ReLU functions with adversarial training. These findings are of particular importance in the field of adversarial learning.

However, it is essential to note that the capping of ReLU functions also has a significant limitation. This technique creates a zero gradient area at the other end of the ReLU function, leading to a vanishing gradient problem in large architectures and negatively impacting the training process for large-scale image classifiers. This is particularly true for dense layers. On the other hand, capping ReLU functions in convolutional layers is less likely to affect the training process since it has a constant number of parameters in each filter (i.e., the filter's area times the number of input channels).

For future research, we aim to improve this approach so that classifiers can be more robust against a broader range of attacks (e.g., CW) and demonstrate scalability on large-scale classifiers. Additionally, as several large models can be imported into popular deep learning frameworks, we also aim to show that adding a layer at the beginning or end before the output layer with a customized ReLU function can

control adversarial perturbations and improve robustness as an alternative to directly customizing activation functions within the models.

References

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Gong, Z., Wang, W., and Ku, W.-S. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

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

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pp. 2137–2146. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kos, J., Fischer, I., and Song, D. Adversarial examples for generative models. In *2018 ieee security and privacy workshops (spw)*, pp. 36–42. IEEE, 2018.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kurakin, A., Goodfellow, I., Bengio, S., et al. Adversarial examples in the physical world, 2016.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pp. 99–112. Chapman and Hall/CRC, 2018.

Lin, J., Gan, C., and Han, S. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019.

Luo, B., Liu, Y., Wei, L., and Xu, Q. Towards imperceptible and robust adversarial example attacks against neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

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

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387. IEEE, 2016b.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Rakin, A. S., Yi, J., Gong, B., and Fan, D. Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions. *arXiv preprint arXiv:1807.06714*, 2018.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

Sooksatra, K. and Rivas, P. Enhancing adversarial examples on deep q networks with previous information. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 01–07. IEEE, 2021.

Sooksatra, K. and Rivas, P. Evaluation of adversarial attacks sensitivity of classifiers with occluded input data. *Neural Computing and Applications*, 34(20):17615–17632, 2022.

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

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

Zhao, Z., Dua, D., and Singh, S. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.