Project Report : CS 4803/7643 Spring 2020
[Re] Enhancing Adversarial Defense by k-Winners-Take-All

Ilya Golod
Georgia Institute of Technology
igolod@gatech.edu

Baran Usluel
Georgia Institute of Technology
baran@gatech.edu

Abstract

The goal of our project was to demonstrate the reproducibility of the ICLR 2020 paper "Enhancing Adversarial Defense by k-Winners-Take-All" (Xiao et al.*) [3]. We have shown that usage of the k-Winners-Take-All (k-WTA) activation function instead of ReLU does indeed significantly increase a model's robustness against white-box adversarial attacks even without adversarial training, without a significant decline in its accuracy. We have applied the method proposed in [3] to two models including one that was not featured in the original paper, as well as multiple adversarial attacks. In all these experiments, models with k-WTA activation outperformed the conventional ones, thus demonstrating the method's reproducibility. Furthermore, we have discovered a significant overhead with using k-WTA compared to ReLU in training and prediction time not mentioned in the original work, and propose a workaround to address this limitation.*

1. Introduction/Background/Motivation

In [3], Xiao *et al*. have shown how a simple change of activation function can significantly improve a deep neural network's robustness to white-box adversarial attacks – i.e. techniques targeted at fooling a model by feeding it an image generated by an algorithm knowing the weights to which the model is trained. As [3] was an open issue for the next ICLR Reproducibility challenge, we decided to reproduce its results and test the robustness of the method presented in the paper over a larger variety of models than presented in the original.

Using conventional activation functions such as ReLU, modern neural networks achieve high accuracy on a wide variety of tasks. However, many of these well-performing models are still quite vulnerable to adversarial attacks which can be used to make a model produce a very unrealistic output for a seemingly regular input sample with imperceptible modifications. This is a major security risk for applications such as face recognition authorization and autonomous driving. Currently, one of the most common method to make models more robust to such attacks is adversarial training [1] – i.e. using machine-generated adversarial samples as a part of the training data.

If the method presented in [3] is proven reproducible, it can potentially become a very feasible alternative or complement to adversarial training. In comparison with the latter, just using k-WTA activation function constitutes less overhead in terms of both development and training, as it only requires a small change to already existing models, while resulting in a notable improvement in robustness.

To test the reproducibility of this method we decided to use the CIFAR-10 image classification dataset which was also used by Xiao *et al*. This dataset consists of 60000 32x32 color images in 10 classes, split into 50000 training and 10000 test images.

2. Approach

The core of our approach was to substitute the conventional activation functions in popular deep learning models with k-WTA and to measure their subsequent performance and robustness to adversarial attacks.

2.1. k-Winners-Take-All Activation Function

k-WTA function is defined as the following:

$$\phi_k(\mathbf{y})_j = \begin{cases} y_j, y_j \in \{k \text{ largest elements of } \mathbf{y}\} \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

This operator retains the k largest values of an $N \times 1$ input vector and sets all other values to zero. This is illustrated with a comparison to ReLU in Figure 1.

As can be seen, the function comes with its hyperparameter k. The same way it is done in [3], not to define it for each layer of our models separately, we will replace it with another hyper parameter $\gamma \in (0, 1)$ called sparsity ratio. By our definition of γ, k for each layer will be determined in the following way: $k = \lfloor \gamma N \rfloor$, where N is the dimension of the layer.

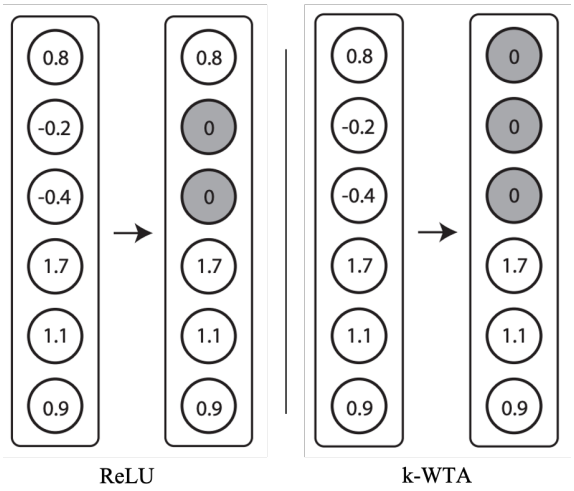

Figure 1: Comparison of ReLU and k-WTA activations

2.2. Outline of the Approach

Our idea for the experiment was to first reimplement the k-WTA function and alter the tested models (see Sec. 3 for the list) by replacing the original activation function used in them with k-WTA. In particular, we also wanted to include the models that were not covered in [3], as it seemed a logical way to prove the robustness of the method. After that, we wanted to retrain the models with the k-WTA activation functions in them. Finally, we wanted to subject the models to a number of adversarial attacks such as vanilla gradient descent, projected gradient descent, Deepfool, etc.

The idea was then to compare the performance of the altered models to the original ones on the same datasets. By examining the resulting accuracies on both the adversarial and normal testing sets, we then wanted to confirm the effectiveness of the method. The goal that we set as the positive experimental outcome is us seeing a visible increase of accuracy on the adversarial samples without a significant decrease in accuracy on normal samples. Another secondary goal of our work was to determine whether we can find any limitations, problems or pitfalls that were not found or unmentioned in the original study.

While working on this project we did not encounter major problems, but we did have some unexpected findings that lead us to further analysis that was beyond the scope of our initial proposal. Additionally, the training and analysis for each model took longer than anticipated, so we had to use fewer different model architectures than planned.

3. Experiments and Results

3.1. Frameworks and Data

All experiments are conducted with the PyTorch deep learning framework and Google Colaboratory computational environment, using GPU acceleration.

We made use of the CIFAR-10 dataset, one of the several datasets tested in [3], for our investigation. Xiao *et al*. tested k-WTA with three models: ResNet18, DenseNet and Wide ResNet. Due to constraints, we chose to focus our reproducibility experiments on ResNet, given its popularity in the literature. We additionally considered AlexNet, to test whether we would get similar findings using k-WTA with a previously unexplored model architecture.

The standard pre-trained implementations of the ResNet18 and AlexNet architectures from the Torchvision package were used. Since these networks were originally trained for the ImageNet dataset, we modified the final layer to output 10 features and fine-tuned the models on the CIFAR-10 dataset.

3.2. Training Models

We took the trained models, replaced all of the ReLU activation layers with k-WTA using a sparsity $\gamma = 0.2$ (referred to as k-WTA-0.2) and trained the models again for several epochs. Finally we took the k-WTA-0.2 models and further decreased the sparsity in the activations to produce k-WTA-0.1 and fine-tuned the models again. This follows the concept of incremental training proposed by Xiao *et al*., based on the reasoning that networks with a lower sparsity ratio will be less expressive and harder to train, but ideally more robust to adversarial attacks.

The key decision that we made here was not to train our models from scratch, but to use the weights from the previously pre-trained ReLU-based models. Our assumption was that due to a certain degree of similarity between ReLU and k-WTA, the optimal weights for the same models with different activation functions should be relatively close to each other. This idea allows us to decrease the time required for training dramatically, as the number of epochs that it takes for k-WTA models to converge from ReLU weights is much lower than the convergence time from random weights. As shown in our results section 3.5.1, our assumption was later confirmed.

We used the same implementation of the k-Winners-Take-All forward pass function as the original paper to minimize differences in our reproduction of the results.

Unlike most examples in which the CIFAR-10 dataset is used, the original paper did not normalize the input images, so we followed suit. Our training dataset was preprocessed with random cropping and random horizontal flipping, to replicate the same training environment as Xiao *et al*. Finally, we additionally scaled up the images to 224x224

since the Torchvision pre-trained model architectures expect this input dimensionality. We used a batch size of 16 in the data loaders, for computational purposes.

All of our models used the Cross Entropy loss function and were trained with Stochastic Gradient Descent using a learning rate initially set to 0.01 and then manually adjusted to 0.001, and a momentum of 0.9. We also had to introduce gradient clipping with a max norm of 0.5 to combat exploding gradients. We did not notice any significant overfitting during the training process.

3.3. Robustness to Adversarial Attacks

In [3], Xiao *et al*. propose that the rationale of the k-WTA activation function is that it destroys network gradients due to its densely piecewise continuous nature, which are relied upon in white-box attacks. They investigated the robustness of k-WTA models to several white-box attacks, including but not limited to Projected Gradient Descent (PGD) and Deepfool. We will similarly look at these two attacks along with a vanilla gradient descent attack.

The PGD and Deepfool attacks will be run using the Foolbox third-party toolkit [2] with the same parameters as [3]. Attacks use the l_∞ metric, a perturbation $\epsilon = 0.031$ and inputs bounded in the range $[0, 1]$. PGD is run for 40 steps with random starting and a relative step size of 0.003. Deepfool is run for 20 steps with a candidate sub-sampling parameter of 10.

The third attack type is our implementation of a targeted vanilla gradient ascent (VGA) attack, as set forth in Problem Set 2 of CS4803/7643. It uses 20 maximum steps and a step size of 10. To calculate aggregate statistics, we run this attack multiple times on each input image, targeting each of the 9 classes different from the ground truth class.

Due to computational constraints, each of these attacks was run on only the first 20 mini-batches (of size 16) of images from the test set.

The standard test accuracy (A_{std}) and robustness accuracy findings for the ResNet18 models are found in Table 1, and for AlexNet in Table 2. The robustness accuracy metric was defined as the accuracy of the models' predictions to the ground truth labels after the inputs undergo attack perturbations. The best accuracies across the models are emphasized in bold.

Activation	A_{std}	VGA	PGD	Deepfool
ReLU	**92.9%**	0.0%	0.0%	0.0%
k-WTA-0.1	85.7%	**7.2%**	**50.3%**	**48.8%**
k-WTA-0.2	91.5%	6.5%	16.6%	47.2%

Table 1: ResNet18 Experiment Results

Activation	A_{std}	VGA	PGD	Deepfool
ReLU	87.7%	0.1%	8.8%	0.0%
k-WTA-0.1	**88.4%**	**18.1%**	**27.8%**	**3.1%**
k-WTA-0.2	**88.4%**	0.9%	11.6%	0.0%

Table 2: AlexNet Experiment Results

3.4. Reproducibility Confirmation

From the results above, we can see that using k-WTA activation functions instead of ReLU indeed provides a significant increase in robustness against adversarial attacks for the models. Just as in the original paper, the increases differ from model to model, and the numbers are very similar to the ones presented in [3] for the corresponding attacks. Furthermore, for the PGD attack we were able to get a significantly larger increase in robustness than in the original paper (50.3% vs 13.3%). It is difficult to say what exactly caused such a change, since we used exactly the same hyperparameters as in [3] to confirm its reproducibility. One of the possible sources of this increase in performance is the differences in the training processes.

We can also see a confirmation of transferability of the method to models different from those used in the original paper (AlexNet, in particular). Even though the exact robustness values for the same attacks vary between the models we used, it is clear that there is a noticeable increase in robustness with usage of k-WTA in comparison to ReLU activation functions.

3.5. Findings beyond Reproducibility Confirmation

3.5.1 A More Efficient Way to Train k-WTA models

While training the models, we confirmed our hypothesis about the similarity of weights between the ReLU and k-WTA. By initializing the k-WTA-based models with the weights taken from the pre-trained ReLU-based ones, we were able to achieve a very large reduction in training required in comparison with the original paper. In [3] it took Xiao *et al*. **80** epochs to train the k-WTA-based ResNet18. We, on the other hand, spent only **6** epochs on training it starting with the weights from the ReLU version, while achieving results very similar to the original.

We did not dive deeply into the formal confirmation of this similarity of weights as it seemed somewhat beyond the scope of this project, but the intuition behind it is the following. The general structure of these activation functions (ReLU and k-WTA) are quite similar to one another, and they can often produce very similar outputs for certain layers. From our several tests on this particular matter, we found that even a k-WTA-based AlexNet which was initialized with unaltered pre-trained ReLU weights is able to pro-

duce around 50% test accuracy on 10 classes, which is far beyond random chance.

3.5.2 k-WTA Training Overhead

Another result beyond the reproduction of the original paper itself came from the fact that we noticed that the time it takes for a k-WTA-based model's epoch of training to run appears to be somewhat larger than the one of a ReLU-based model. To confirm this suspicion, we decided to run training of ResNet18 for 10 epochs with each of the activation functions (ReLU and k-WTA) and measure the average time it took one epoch of training to complete. To get a full picture of the models' performance, we also measured the time taken by just forward passes by timing the test accuracy evaluation loop which performs 10,000 passes. The results turned out to be quite interesting and can be seen in the Table 3.

Activation	Training Time per Epoch	Test Time
ReLU	381 sec	10 sec
k-WTA-0.2	735 sec	42 sec

Table 3: ResNet18 Benchmarking Results: Average time in seconds to train 50,000 samples and evaluate 10,000 samples.

It can be seen from these results, that k-WTA-based models are trained almost 2 times as slow as their ReLU-based counterparts. Furthermore, there is an even larger difference between the models' respective times to evaluate the test set. A ReLU-based model's forward pass appears to be more than 4 times faster than the k-WTA based one's. It seemed as quite an important finding to us, as this overhead was not mentioned in the original paper, but can itself be seen as a noticeable shortcoming of usage of k-WTA as an activation function.

3.5.3 Workaround for k-WTA Training Overhead

Fortunately, our finding regarding the similarity of ReLU and k-WTA weights turns out to be itself a workaround to the slowdown problem described above. Models can be pre-trained with ReLU activation functions and only afterwards trained for just several epochs with the k-WTA activation functions in place—which can be seen as a kind of fine tuning—to increase any deep model's robustness to adversarial attacks. According to our metrics provided in the Tables 1, 2, this approach yields accuracies similar to the ones demonstrated in [3], while requiring a significantly less overhead in terms of the training time.

4. Conclusion

Through the experiments described above we were able to reproduce and confirm the results acquired by Xiao *et al.* in [3]. We found that replacing ReLU activation functions with k-WTA in existing models can indeed considerably increase its robustness to adversarial attacks without causing a significant decrease in the model's accuracy. We have also confirmed that Xiao *et al.*'s method works with models beyond the ones examined in the orginial paper (i.e. AlexNet).

Furthermore, beyond the scope of reproducing the original paper, we have found that the optimal weights of a ReLU-based model tend to be significantly closer to the optimal weights of a similar k-WTA-based model than randomly set weights. This allows to use the weights of a ReLU-based pre-trained model as the starting point for training of a k-WTA-based model. This has a very significant positive impact on the convergence time of the training process. Therefore, by using this technique, one can decrease their training overhead associated with a transition to k-WTA dramatically, thus effectively increasing the robustness of a model without spending too much time on the additional training required.

Lastly, we found a slight drawback of usage of k-WTA in comparison to ReLU functions that was not mentioned in [3]. The per-epoch training time of k-WTA-based models happens to be around 2 times as large as the time of ReLU-based ones. This makes k-WTA models train somewhat slower than conventional ones. With this in mind, we propose that the optimal usage of the k-WTA activation function seems to be to train a ReLU-based model first, as it takes significantly less time to converge, and only after it, to replace the activations with k-WTA and finish the training with a small number of epochs.

5. Work Division

Team members' contribution information as required in the assignment can be found in Table 4 on the last page of this PDF.

References

[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1

[2] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017. 3

[3] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by kw inners-take-all, 2020. 1, 2, 3, 4

Student Name	Contributed Aspects	Details
Ilya Golod	Implementation, Analysis, Report Writing	Implemented the necessary models, analyzed the results of the experiments.
Baran Usluel	Implementation, Experimentation, Report Writing	Trained the models, implemented and ran the attacks, ran the experiments.

Table 4: Contributions of team members.