CS463 Survey Paper: Defense against adversarial attacks

Yuguang WEI
yuguang2@illinois.edu
2019.4.15

Abstract

Deep networks have achieved impressive results across a variety of important tasks. However a known weakness is a failure to perform well when evaluated on data which differ from the training distribution, even if these differences are very small, as is the case with adversarial examples. Distillation and fortified network as two famous defensive mechanism are presented at this here. I focus on the architecture and statistical intuition in my survey. I would like to demonstrate the insight of each strategy and indicate the merits and weakness of each one.

1.Introduction

Deep Learning has been demonstrated to perform exceptionally well on several categories of machine learning problems, notably input classification.(A. Krizhevsky and Hinton 2012)These Deep Neural Networks (DNNs) efficiently learn highly accurate models from a large corpus of training samples, and thereafter classify unseen samples with great accuracy.However, there will always be adversarial samples which cause misclassification of the DNNs.Such attacks can seriously undermine the security of the system supported by the DNN, sometimes with devastating consequences. For example,autonomous vehicles can be crashed, illicit or illegal content can bypass content filters, or biometric authentication systems can be manipulated to allow improper access.(Barreno et al. 2006) So,the establish and development of defensive mechanism are essential for the robustness and security consideration of the applications. In this section, I would like to introduce two state of art defensive mechanism,distillation and fortified network, which are embedded with existing DNN model to improve the robustness and defensive ability against deliberately crafted adversarial examples.(Warde-Farley and Bengio 2016)
Firstly, the strategy of crafting deliberate samples are demonstrated as background to give a sense of what kind of attack we should defense and detect.
Secondly, defensive distillation model are presented as a extend version of distillation which was initially motivated by embedding DNN into resource restrained devices, such as smart phone.(Yuan et al. 2014) The insight of this mechanism is reducing sensitivity of adversarial gradients, which is straight forward derived from crafting procedures. The defensive performance and influence of accuracy are discussed as well. Finally, I demonstrate the evaluation of this approach on MNIST, CIFAR10 datasets.

As following, the Fortified Networks consists of using denoising auto encoders to decorate the hidden layers of the original network. use "decoration" in the Pythonic sense that it can be applied to any function and extend its behavior without explicitly modifying it. The intuition behind the fortification of hidden layers is discussed and lay out some of the method's salient properties. Finally, evaluation of our this approach on MNIST, CIFAR10 datasets is shown and compared with other contemporary techiniques just as that in distillation.

2. Craft Adversarial samples

The adversarial sample is crafted in two processes:

- Direction sensitivity estimation
- Perturbation selection

Basically, It is proposed to discuss two famous strategies which are both capable of producing effective adversarial samples in this part.

The first step considers sample X, a M-dimensional input. The goal here is to find the dimensions of X that will produce the expected adversarial behavior with the smallest perturbation. To achieve this, the adversary must evaluate the sensitivity of the trained DNN model F to changes made to input components of X.(Papernot et al. 2016b) Building such a knowledge of the network sensitivity can be done in several ways. Goodfellow et al.(I. J. Goodfellow and Szegedy 2015) introduced the fast sign gradient method that computes the gradient of the cost function with respect to the input of the neural network. Finding sensitivities is then achieved by applying the cost function to inputs labeled using adversarial target labels. Papernot raised (Papernot et al. 2016a) a different approach and introduced the forward derivative, which is the Jacobian of F, thus directly providing gradients of the output components with respect to each input component.

The second step considers to determine a minimum perturbation from most likely dimensions, which can result in misclassification. Goodfellow et al. (I. J. Goodfellow and Szegedy 2015) proposed to introduce a small amplitude perturbation in all dimensions after determining the sensitivity gradient. The advantage of this method is that the magnitude of the perturbation is small in Euclidean space and the generation process is simple. In contrast, Papernot et al.(Papernot et al. 2016a) take a different approach and follow a more complex process involving saliency maps to only select a limited number of input dimensions to perturb. Saliency maps assign values to combinations of input dimensions indicating whether they will contribute to the adversarial goal or not if perturbed. This effectively diminishes the number of input features perturbed to craft samples. The parameters which determine the amplitude of the perturbation are fixe in both approaches and these are also reply on the input nature, for instance images, malware,sound etc.(Papernot et al. 2016b)

The craft of adversarial samples could be described as:

$$argmin|\delta X|s.t.F(x+\delta X)=Y^*$$

, where $Y^*!=Y$. The network structure of this process is shown as Figure 1

3.Distillation

3.1 Distillation Defense

Distillation techniques for DNN was first motivated for reducing the size of DNN which can be implemented in constrained resources, such as smart phone. This strategy propose to train the second neural net with same input samples but different labels captured from origin network.(Hinton, Vinyals, and Dean 2015) The distinguishing is that the probability vector,referred as the output of the softmax function, is feed into a smaller size of neural network as the label of second neural net. Based on kullback-Liber divergence theorem, these two probability would like to converge finally.

It is mentioned that the temperature parameters plays an central role in the distillation process.

$$F(x)=[\frac{e^{\frac{z_i(x)}{T}}}{\sum_{j=0}^{N-1}e^{\frac{z_j(x)}{T}}}]_i$$

The temperature T squeezes the probability distribution of the output vector. A large value of T forces the logits close to be zero, as a result, the output vector converge $\frac{1}{N}$ for all categories. The effect of the temperature would be discussed in next section.(Faghri et al. 2017)

Our intuition is that knowledge extracted by distillation, in the form of probability vectors, and transferred in smaller networks to maintain accuracy comparable with those of larger networks can also be beneficial to improving generalization capabilities of DNNs outside of their training data set and therefore enhances their resilience to perturbations.The methodology for defensive distillation is the same as general distillation except for the second DNN is the same as the first one. (Sermanet et al. 2013)

Following procedures describe how to train distilled model:

- Given this training set , we train a deep neural network F with a softmax output layer at temperature T.
- Form a new training set, by consider samples of the form. Instead using hard labels, the origin probability believe logits are used as labels
- Using the new training set to train another DNN model with the same neural network architecture as F, and the temperature of the softmax layer remains T. This new model is referred to as the distilled model and noted as F^d

3.2 Distillation Effect Analysis

3.2.1Impact of the Sensitivity In this section, the analysis and derivation here aim to provide an intuition of how distillation at high temperatures improves smoothness of the distilled model F^d compared to model F,thus reducing its sensitivity to small input variations. The derivative of $F(X)$ to each X_j can be calculated as:

$$\begin{aligned}\left.\frac{\partial F_i(X)}{\partial X_j}\right|_T =&\frac{\partial}{\partial X_j}\left(\frac{e^{z_i/T}}{\sum_{l=0}^{N-1}e^{z_l/T}}\right)\\ =&\frac{1}{g^2(X)}\left(\frac{\partial e^{z_i(X)/T}}{\partial X_j}g(X)-e^{z_i(X)/T}\frac{\partial g(X)}{\partial X_j}\right)\\ =&\frac{1}{g^2(X)}\frac{e^{z_i/T}}{T}\left(\sum_{l=0}^{N-1}\frac{\partial z_i}{\partial X_j}e^{z_l/T}-\sum_{l=0}^{N-1}\frac{\partial z_l}{\partial X_j}e^{z_l/T}\right)\\ =&\frac{1}{T}\frac{e^{z_i/T}}{g^2(X)}\left(\sum_{l=0}^{N-1}\left(\frac{\partial z_i}{\partial X_j}-\frac{\partial z_l}{\partial X_j}\right)e^{z_l/T}\right)\end{aligned}$$

From the equation, it can be seen that the gradient of $F(x)$ with respect to the input is inversely proportional to the amplitude of temperature(T).Recalling from the process of generating adversarial samples, the gradient is crucial to craft the adversarial samples. If the adversarial gradient is smooth and negligent, the amplitude of the perturbation is required to be significant in order to achieve the same effects. Thus, the sensitivity of the DNNs to adversarial samples was reduced and slight modifications or perturbation of samples would not lead to misclassification in the distilled model F^d.

3.2.2Impact of the Generality Another interesting point is that why F^d distilled model is stilled necessary in the presence of the parameter T appears in the first model F. This particular architecture shown in Figure 2 improves the generality and reducing the overfitting of the previous model significantly. It can learn the structural similarity between different objects. For instance, suppose we are given a sample X of some hand-written 7 but that the writing is so bad that the 7 looks like a 1. Assume a model F assigns probability F7(X) = 0.6 on 7 and probability F1(X) = 0.4 on 1, when given sample X as an input. This indicates that 7s and 1s look similar and intuitively allows a model to learn the structural similarity between the two digits. Note that model F^d should theoretically eventually converge towards F. Indeed, locally at each point $(X, F(X))$, the optimal solution is for model F^d to be such that $F^d(X) = F(X)$. However, empirically this is not the case because training algorithms approximate the solution of the training optimization problem, which is often non-linear and non-convex. Furthermore, training algorithms only have access to a finite number of samples. Thus, we do observe empirically a better behavior in adversarial settings from model F^d than model F.(Papernot et al. 2016b)

3.3 Evaluations and Results

In this section, I would like to show some results of distilled model. First, the defensive effect to adversarial samples is

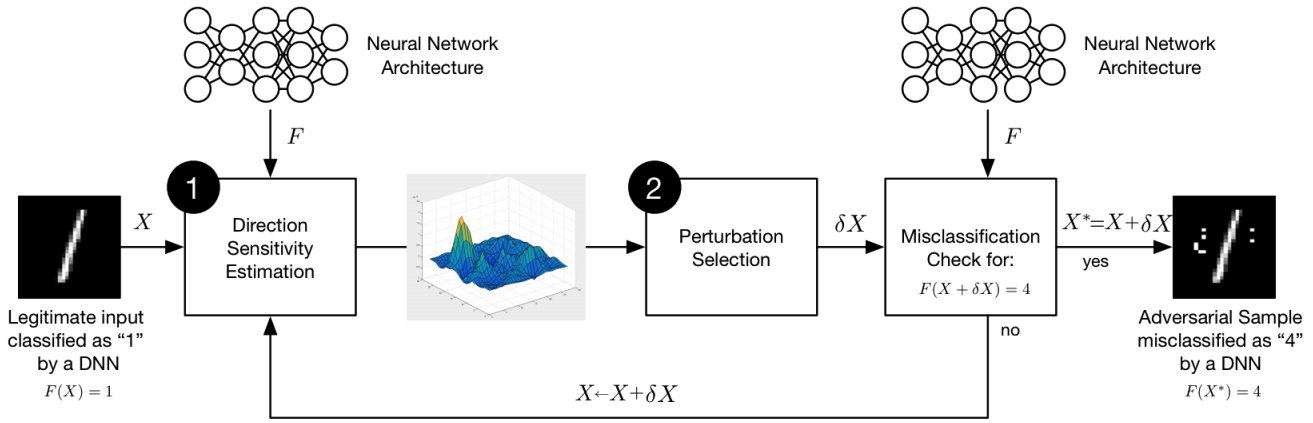

Figure 1: network for generating adversarial samples(Papernot et al. 2016b)

shown in below graph.

From this graph, it can be seen that the successful rate of adversarial samples dropped dramatically with the increase of temperature. Another issue is that whether the distilled techniques would influence the performance of neural network.The comparison of these two version is also shown as following.(Papernot et al. 2016b)

It can be observed that the accuracy for MNIST slightly decreased by 0.5 percent in average but that of CIFAR10 is slightly increased by 0.5 percent in average for temperature from 0 to 50. It can be inferred that distilled model even capable of improving the accuracy of models when the target scenario involves more complex features just like CIFAR 10 compared with MNIST(CIFAR10 involves RGB images of 10 categories and MNIST is gray image of hand wirten digits 0 to 9). It can be explanied by the model F^d reduce the effects of overfitting and improve resilience of the model, which is obvious in the complex scenario. It is also noticed that the accuracy decrease for both datasets when temperature is 100. There is a trade-off between the accuracy and defensive ability, which should be considered and determined by discretion for each scenario.

4.Fortified Networks

4.1 Fortified Networks Structure

Denoising autoencoders (DAEs) are neural networks which take a noisy version of an input (for example, an image) and are trained to predict the noiseless version of that input.The DAE is the core component of fortified network.It is also argued that inserting the DAE after the hidden layer instead of raw input layers facilitates learning, while preventing the adversarial sampels.(Lamb et al. 2018)

The DAE is described as

$$h_k{}^{decode} = D_k(E_k(h_k + n_k))$$

For the training purpose, aside from the original classification loss, L_C, we also include the classification loss from the adversarial objective, $Lc(ye)$ and we introduce a dual objective for the DAEs.

- **Reconstruction loss.** For a mini-batch of N clean examples, $x^{(1)}, \ldots, x^{(N)}$, each hidden layer $h_k^{(1)}, \ldots, h_k^{(N)}$ is fed into a DAE loss, similar to (3):

$$\mathcal{L}_{rec,k} = \frac{1}{N} \sum_{n=1}^{N} \left\| D_k \left(E_k \left(h_k^{(n)} + n_k \right) \right) - h_k^{(n)} \right\|_2^2.$$

- **Adversarial loss.** We use some adversarial training method to produce the perturbed version of the mini-batch, $\widetilde{x}^{(1)}, \ldots, \widetilde{x}^{(N)}$, where $\widetilde{x}^{(i)}$ is a small perturbation of $x^{(i)}$ which is designed to make the network produce the wrong answer. The corresponding hidden layer $\widetilde{h}_k^{(1)}, \ldots, \widetilde{h}_k^{(N)}$ (using the perturbed rather than original input) is fed into a similar DAE loss:

$$\mathcal{L}_{adv,k} = \frac{1}{N} \sum_{n=1}^{N} \left\| D_k \left(E_k \left(\widetilde{h}_k^{(n)} + n_k \right) \right) - h_k^{(n)} \right\|_2^2,$$

  where we note that the target reconstruction for denoising is the clean version of the hidden layer, without noise and without adversarial perturbation.

To build a fortified network, we can apply this fortification process to some or all the layers. The final objective used for training the fortified network includes the classification loss and all reconstruction and adversarial losses:

$$L = L_c(y) + L_c(y_{adv}) + \lambda_{rec} \sum_k L_{rec,k} + \lambda_{adv} \sum_k L_{adv,k}$$

The purpose of first two items in the loss functions are general traning loss with respect to label. The last two items are customized for optimizing reconstruction error and adversarial difference respect to the hidden layer unit.

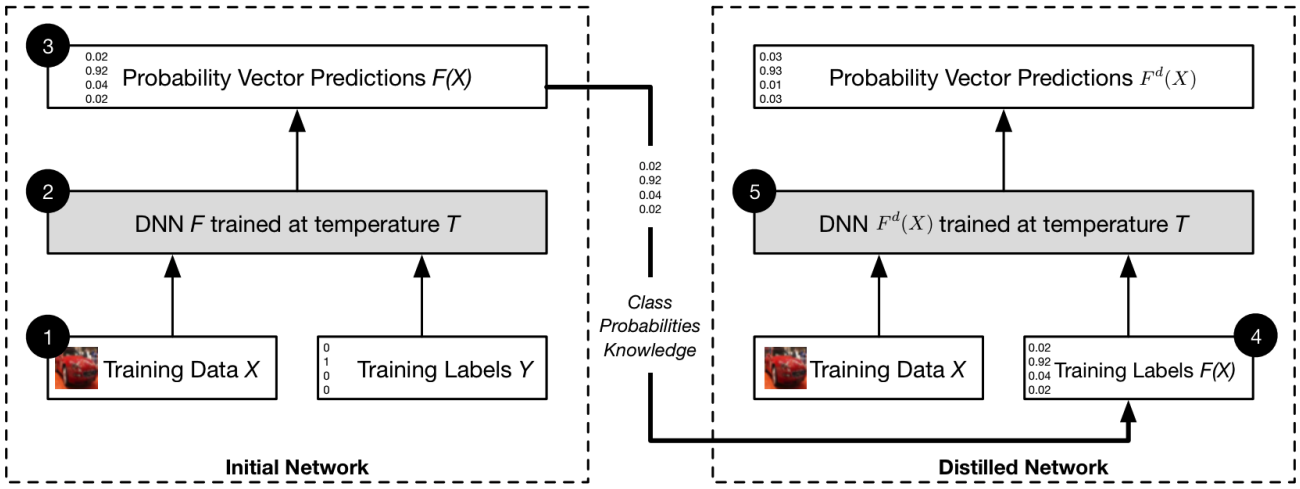

Figure 2: An overview of our defense mechanism based on a transfer of knowledge contained in probability vectors through distillation: We first train an initial network F on data X with a softmax temperature of T. We then use the probability vector F(X), which includes additional knowledge about classes compared to a class label, predicted by network F to train a distilled network Fd at temperature T on the same data X(Papernot et al. 2016b)

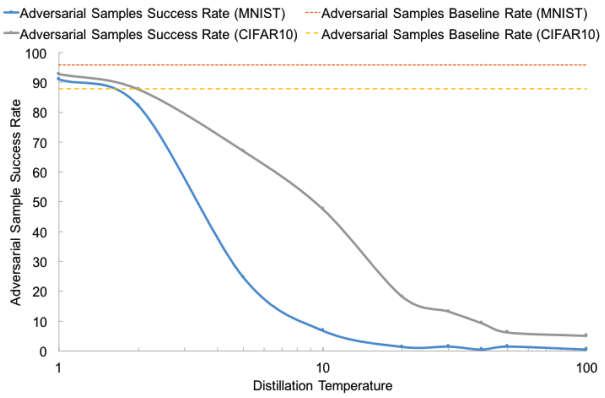

Figure 3: An exploration of the temperature parameter space: for 900 targets against the MNIST and CIFAR10 based models and several distillation temperature

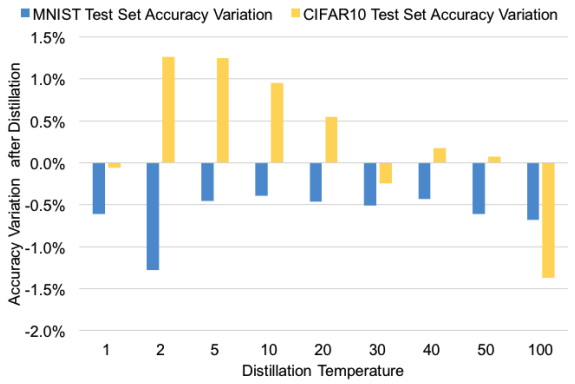

Figure 4: An exploration of the temperature parameter space: for 900 targets against the MNIST and CIFAR10 based models and several distillation temperature

4.2 Fortified Networks Evaluations

In this section, I would like to show some results of fortified network.Two particular experiments are chosen from the paper(**?**) in order to compare with that of distillation process later and these experiments also test the defense of adversarial samples on MNIST and CIFAR10 respectively.It is also mentioned that the fortified network is only used in one hidden layer

Table 1 Accuracies against white-box MNIST attacks with FGSM(Fast gradient sign method),where the model is a convolutional net. We used the standard FGSM attack parameters with 0.3

Model	FGSM
Adv. Train (Madry et al., 2017)	95.60
Adv. Train (Jacob Buckman, 2018)	96.17
Adv. Train (ours)	96.36
Adv. Train No-Rec (ours)	96.47
Quantized (Jacob Buckman, 2018)	96.29
One-Hot (Jacob Buckman, 2018)	96.22
Thermometer (Jacob Buckman, 2018)	95.84
Our Approaches	
Fortified Network - Conv, w/o $\mathcal{L}_{adv}$	96.46
Fortified Network - Conv	**97.97**

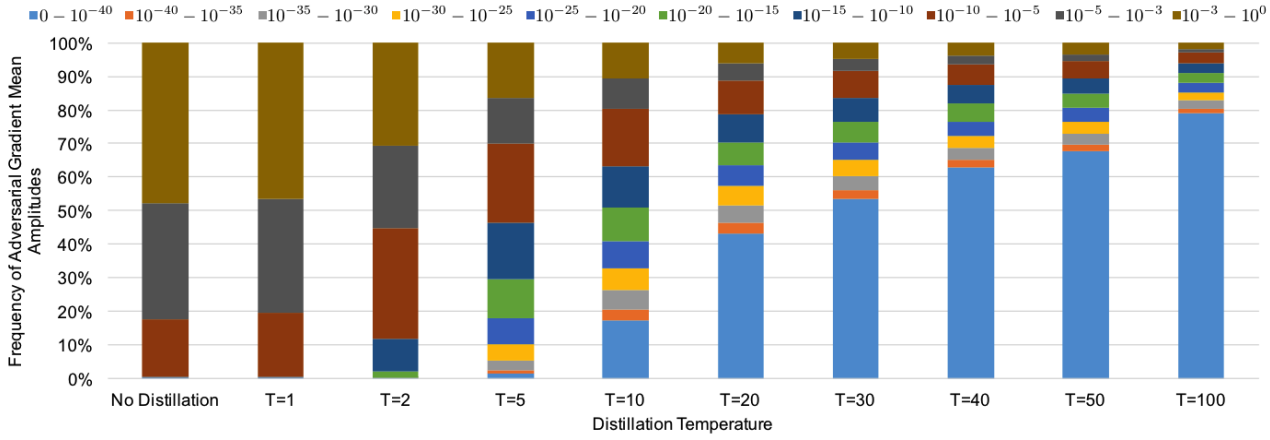

Figure 5: An exploration of the impact of temperature on the amplitude of adversarial gradients We illustrate how adversarial gradients vanish as distillation is performed at higher temperatures. Indeed, for each temperature considered, we draw the repartition of samples in each of the 10 ranges of mean adversarial gradient amplitudes associated with a distinct color. This data was collected using all 10, 000 samples from the CIFAR10 test set on the corresponding DNN model(Papernot et al. 2016b)

From the table 1, it can be seen that the fortified network has better performance in fortifying adversarial samples compared to other techniques and it even performs better when the adversarial samples are introduced into training process as illustrated structure in previous section.(Sharif et al. 2017)

Table 2 Accuracies against white-box CIFAR attacks with FGSM sing the standard, where each model is a convnet.

Model	FGSM
Baseline Adv. Train (ours)	79.34
Quantized (Buckman)	53.53
One-Hot (Buckman)	68.76
Thermometer (Buckman)	80.97
Fortified Networks (ours)	
Autoencoder on input space with loss in hidden states	79.77
Autoencoder in hidden space	**81.80**

From the table2(Lamb et al. 2018), it can be observed that Fortified has better performance than other techniques as well. It can be inferred that the improvement of fortified network is due to the introduce of adversarial in training process.

4.3 Fortified Networks Analysis

From the figure 6, It can be observed that the hidden layer has more obvious and characteristic distribution in spatial space and convexity as well.The key motivation behind Fortified Networks is that directions which point off the data manifold are easier to identify in an abstract space with simpler statistical structure, making it easier to map adversarial examples back to the projected data manifold.(Madry et al. 2017) So, the fortified network is placed in the hidden layer instead of after the input layer directly.(Buckman et al. 2018) Off-manifold signaling The reconstruction losses act as a re-liable signal for detecting off-manifold examples This is a particularly useful property in practice: not only can we provide more robust classification results, we can also sense and suggest to the analyst or system when the original example is either adversarial or from a significantly different distribution.(Gilmer et al. 2018) There is still one remaining problems when and where to use fortified layers. It has argued that advantages to placing fortified layers in the hidden states instead of the input space but the question of where exactly fortified layers need to be placed remains unanswered. Is it just the final hidden layer? Is it every hidden layer? We outline two important considerations regarding this issue:

1. In the higher-level hidden layers, it is much easier for the network to identify points which are off of the manifold or close to the margin.

2. At the same time, the higher level hidden layers may already look like points that are not adversarial due to the effect of the adversarial perturbations in the earlier layers.

Given these opposing objectives, we argue for the inclusion of multiple fortified layers across the network

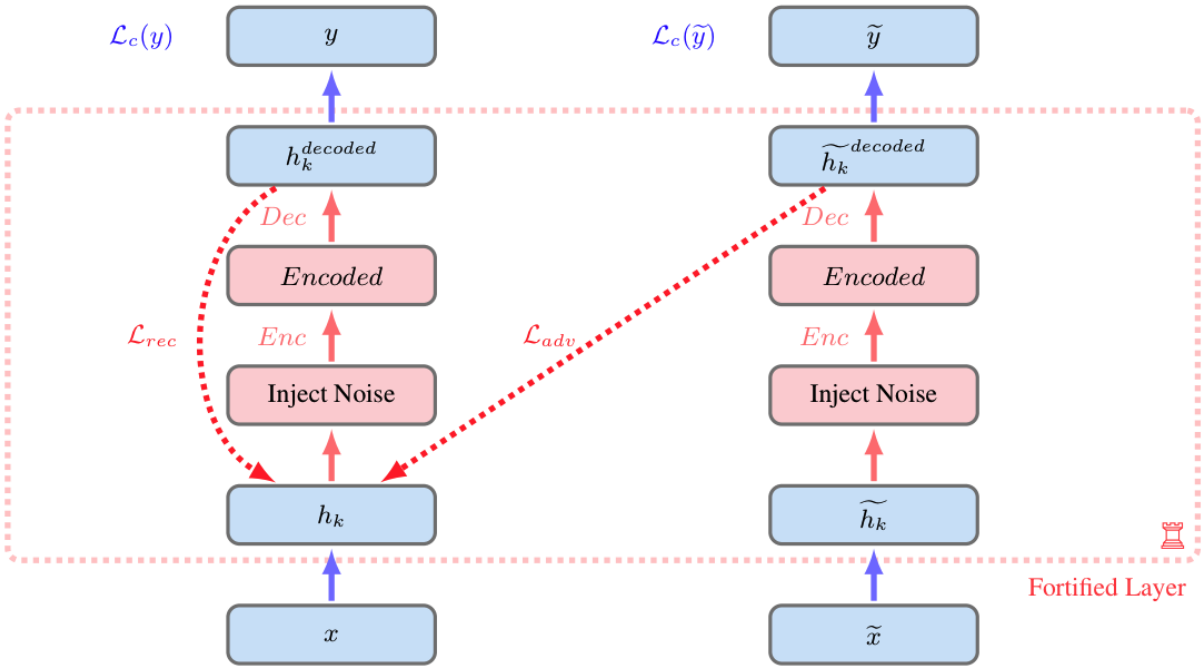

Figure 6: Diagram illustrating a one-layer fortified network. A network is evaluated with a data sample x and its corresponding adversarial example x_adv. Hidden units h_k and h_kadv are corrupted with noise, encoded with the encoder Enc, and decoded with the decoder Dec. The autoencoder (denoted by the red color) is trained to reconstruct the hidden unit h_k that corresponds to the clean input. Dotted lines are two reconstruction costs: for a benign (L_rec) and adversarial examples (L_adv). Note that a layer can be fortified at any position in the network.(Lamb et al. 2018)

Figure 8 demonstrates a particular difference of manifold between the visible space(input state) and latent space(hidden state). A small variation or perturbations would result in total different classification. Obviously, hidden layer is more suitable for serving statistical correction of adversarial samples.(Lamb et al. 2018)

5.Conclusion

Protecting against adversarial examples could be of paramount importance in mission-critical applications. Distillation model and fortified networks, two representative methods for improving robustness and defensive abilities to adversarial samples of existing deep neural networks. In this distillation part, I have introduced the use of distillation, a technique previously used to reduce DNN dimensionality, as a defense against adversarial perturbations. Defensive distillation is fornally defined and evaluated it on standard DNN architectures. Using elements of learning theory, we analytically showed how distillation impacts models learned by deep neural network architectures during training. In the Fortified network part, I have introduced the typical architecture of fortified networks and emphasized the insight of the behind it, applying on the hidden states instead of input states. The difference of manifold in hidden layer(latent) and input layer(visible) are explicitly demonstrates by graphs. The construction of network and definition of the cost function totally reply on DAE and learning theory, which try to optimize the purpose of model statistically.(Xiao et al. 2018)

- Distillation Model: Apply the probability Vector as the labels for the same training samples and attempt to converge the second model to the first model into which temperature parameter is integrated. Distillation Model reduce the sensitivity of adversarial gradients and improve the robustness as well. It is mentioned that there is a trade off between the temperature and accuracy. The high temperature facilitates defensive ability but deteriorates the accuracy of the model due to overwhelming smoothness in softmax function.
- Fortified Networks: fortifying an existing network entails introducing DAEs between the hidden layers of the network. Furthermore, the DAE reconstruction error at test time is a reliable signal of distribution shift, that is examples unlike those encountered during training. High error can signify either adversarial attacks or significant domain shift; both are important cases for the analyst or system to be aware of.

Results Comparison and Clarification: The empirical findings in paper (Papernot et al. 2016b) show that defensive distillation can significantly reduce thes uccessfulness of attacks against DNNs. It reduces the success of

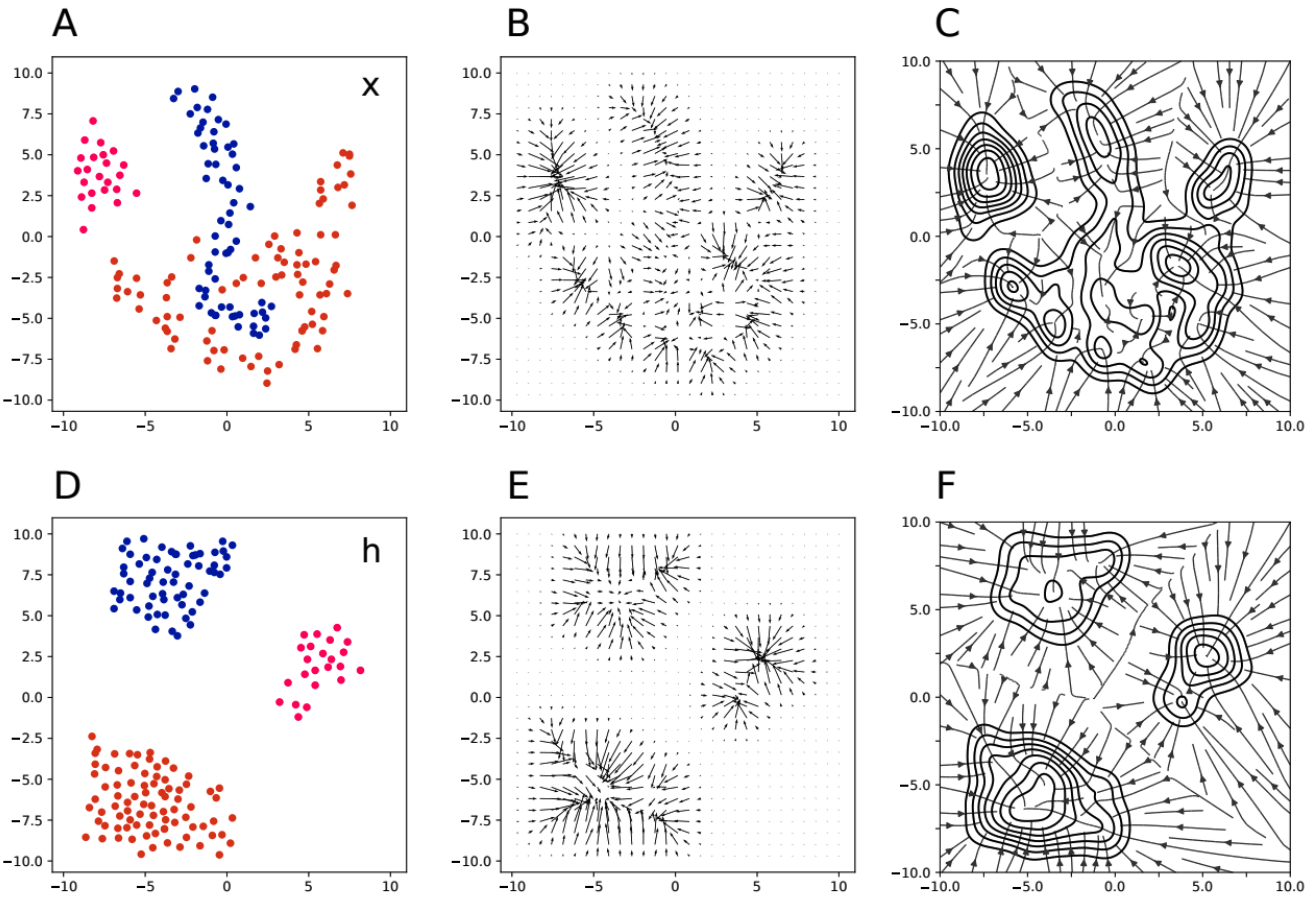

Figure 7: Illustration of the autoencoder dynamics in the input space (top) and in abstract hidden space (bottom). The leftmost panels show data points from three different classes, the middle panels show vector fields describing the autoencoder dynamics, and the rightmost panels show a number of resulting trajectories and basins of attraction. The key motivation behind Fortified Networks is that directions which point off the data manifold are easier to identify in an abstract space with simpler statistical structure, making it easier to map adversarial examples back to the projected data manifold.(Lamb et al. 2018)

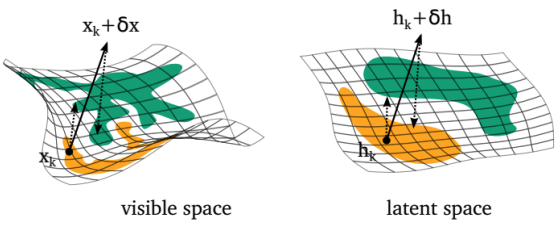

Figure 8: An illustration of the process of mapping back to the manifold in the visible space (left) and the hidden space (right). The shaded regions represent the areas in the space which are occupied by data points from a given class (they do not represent decision boundaries).

adversarial sample crafting to rates smaller than 0.5% on the MNIST dataset and smaller than 5% on the CIFAR10 dataset while maintaining the accuracy rates of the original DNNs.Surprisingly, distillation is simple to implement and introduces very little overhead during training. Hence, this work lays out a new foundation for securing systems based on deep learning. The empirical experimental result shows fortified networks on defenses for adversarial attacks on MNIST and CIFAR 10 improves compared with other state of art techniques. Since the DNNs and the way of crafting adversarial samples are different from distilled model which we just discussed above, there would draw an conclusion that which one performs better here. However, it certainly concluded that fortified networks has better performance in defending adversarial samples in realistic circumstances compared to most existing methods.

Limitations and Trade-off: For the distilled model, there is a trade-off between the high accuracy and defensive capability. High temperature would reduce the sensitivity of adversarial samples while decrease statistically concentration on

particular category, resulting deterioration in accuracy. Additionally, distilled model only works on classification network due to the usage of probability vector.So, it should be discretion and cautious to select suitable temperature parameters in order to optimize the performance in realistic scenario.
For fortified network, the cost of the proposed method is the extended training time due to the search for an adversarial example and training the auto encoder, which in not the case in distilled model.The added cost of the fortified layers over adversarial training by itself is relatively small, and is also much easier and simpler than training a full generative model (such as a GANs) in the input space.(Lamb et al. 2018) Layer fortification typically involves smaller DAEs thatrequire less computation. At the same time, fortified networks have only been shown to improve robustness when used alongside adversarial training, which is expensive for iterative attacks

Future work: For distillation, investigate the impact of distillation on other DNN models and adversarial sample crafting algorithms. One notable endeavor is to extend this approach outside of the scope of classification to other deep learning tasks. This is not trivial as it requires finding a substitute for probability vectors used in defensive distillation with similar properties. Lastly, we will explore different definitions of robustness that measure other aspects of DNN resilience to adversarial perturbations.
For fortified network, two distinct cost function L_rec and L_adv are defined to optimize the overall performance. The existence of on-manifold adversarial samples in hidden state indicates that only DAE is not sufficient to prevent adversarial samples.In the other other hand, there definitely some adversarial samples are capable of evading from the defensive mechanism, that is the essence of another function L_adv. However, it also lead to potential risk in this model. Overfitting of against particular adversarial strategy would cause less resilience to emerging adversarial crafts and burden the training computation cost as well.(Papernot et al. 2016c) Additionally, it is not necessary to put the fortified network all over the DNNs for some particular model, such as ResNet(Two layers are good enough)(Lamb et al. 2018). So, how to determine the most efficient layers in which fortified network is places and reduce the training cost is challenging direction.

References

A. Krizhevsky, I. S., and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks in advances in neural information processing systems, 2012, pp. 1097–1105.

Barreno, M.; Nelson, B.; Sears, R.; Joseph, A. D.; and Tygar, J. D. 2006. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 16–25. ACM.

Buckman, J.; Roy, A.; Raffel, C.; and Goodfellow, I. 2018. Thermometer encoding: One hot way to resist adversarial examples.

Faghri, I. G. R. F. F.; Yi-Lin, A. M. K. H.; Abhibhav, J. A. K. R. S.; Papernot, G. Y.-C. L. N.; and Carlini, N. 2017. cleverhans v2. 0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*.

Gilmer, J.; Metz, L.; Faghri, F.; Schoenholz, S. S.; Raghu, M.; Wattenberg, M.; and Goodfellow, I. 2018. Adversarial spheres. *arXiv preprint arXiv:1801.02774*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

I. J. Goodfellow, J. S., and Szegedy, C. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations. Computational and Biological Learning Society*.

Lamb, A.; Binas, J.; Goyal, A.; Serdyuk, D.; Subramanian, S.; Mitliagkas, I.; and Bengio, Y. 2018. Fortified networks: Improving the robustness of deep networks by modeling the manifold of hidden representations. *arXiv preprint arXiv:1804.02485*.

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

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swam, A. 2016a. The limitations of deep learning in adversarial settings.

Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; and Swami, A. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks.

Papernot, N.; McDaniel, P.; Sinha, A.; and Wellman, M. 2016c. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*.

Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; and LeCun, Y. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Sharif, M.; Bhagavatula, S.; Bauer, L.; and Reiter, M. K. 2017. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*.

Warde-Farley, D., and Bengio, Y. 2016. Improving generative adversarial networks with denoising feature matching.

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

Yuan, Z.; Lu, Y.; Wang, Z.; and Xue, Y. 2014. Droid-sec: deep learning in android malware detection. In *ACM SIGCOMM Computer Communication Review*, volume 44, 371–372. ACM.