Towards Adversarial Purification using Denoising AutoEncoders

Dvij Kalaria, Aritra Hazra, and Partha Pratim Chakrabarti
Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur, INDIA

Abstract

With the rapid advancement and increased use of deep learning models in image identification, security becomes a major concern to their deployment in safety-critical systems. The deep learning architectures are often susceptible to adversarial attacks which are often obtained by making subtle perturbations to normal images, which are mostly imperceptible to humans, but can seriously confuse the state-of-the-art machine learning models. We propose a framework, named **APuDAE**, leveraging Denoising AutoEncoders (DAEs) to purify these samples by using them in an adaptive way and thus improve the classification accuracy of the target classifier networks. We also show how using DAEs adaptively instead directly, improves classification accuracy further and is more robust to the possibility of designing adaptive attacks to fool them. We demonstrate our results over MNIST, CIFAR-10, ImageNet dataset and show how our framework (**APuDAE**) provides comparable and in most cases better performance to the baseline methods in purifying adversaries.

1 Introduction and Related works

The phenomenal success of deep learning models in image identification and object detection has led to its wider adoption in diverse domains ranging from safety-critical systems, such as automotive and avionics (1) to healthcare like medical imaging, robot-assisted surgery, genomics etc. (2), to robotics and image forensics (3), etc. The performance of these deep learning architectures are often dictated by the volume of correctly labelled data used during its training phases. Recent works (4) (5) have shown that small and carefully chosen modifications (often in terms of noise) to the input data of a neural network classifier can cause the model to give incorrect labels. These adversarial perturbations are imperceptible to humans but however are able to convince the neural network in getting completely wrong results that too with very high confidence. Due to this, adversarial attacks may pose a serious threat to deploying deep learning models in real-world safety-critical applications. It is, therefore, imperative to devise efficient methods to thwart such attacks.

Adversarial attacks can be classified into whitebox and blackbox attacks. White-box attacks (6) assume access to the neural network weights and architecture, which are used for classification, and thereby specifically targeted to fool the neural network. Hence, they are more accurate than blackbox attacks (6) which do not assume access to the model parameters. Many recent works have proposed ways to defend these attacks. Methods for adversarial defense can be divided into 4 categories as – (i) Modifying the training dataset to train a robust classifier, popularly known as adversarial training (7; 8) , (ii) Block gradient calculation via changing the training procedure (9; 10; 11) , (iii) Detecting adversaries (12; 13; 14; 15; 16) and (iv) purifying adversaries (17; 18). Detecting adversaries only serves half the purpose, what follows logically is to purify the sample or revert it back to its pure form. Our main focus is defense mechanisms to purify input data that may have added adversarial perturbation. This can allow the mechanisms to effectively address any attacks. Previous works like MagNet (17), Defense-GAN (18) consider training a generative model to learn the data distribution closer to the training distribution and map an adversarial example to its corresponding clean example from the data distribution. We follow a similar technique but

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

instead of a generative model like VAE, we use a Denoising AutoEncoder (DAE). We train the DAE specifically to denoise noisy samples instead. Later, during testing, instead of directly passing the input sample which may contain the adversarial noise and treating the later example as a purified example, we use a novel way by adaptively decreasing the reconstruction error to derive the purified sample. This method has two advantages over former method – (i) Directly passing image through denoising autoencoder often blurs the image as it treats image features as noise as well leading to decreasing target classifier accuracy. Directly modifying the input image to reduce reconstruction error doesn't affect image quality. (ii) The method of purification is non differential so it wouldn't be possible to attack the defense mechanism easily unlike the former method. We observe a significant improvement in results when using DAEs adaptively in comparison to using it directly for MNIST, CIFAR-10 and IMAGENET.

In summary, the primary contributions made by our work are as follows.

(a) We propose a non-differentiable framework for Adversarial Purification using Denoising AutoEncoder (called **APuDAE**) based on DAE to adaptively purify adversarial attacks which cannot be easily attacked in conjunction in complete white box setting.
(b) We test our defense against strong grey-box attacks (attacks where the attacker has complete knowledge of the target classifier model but not the defense model) and attain better than the state-of-the-art baseline methods
(c) We devise possible adaptive attacks specifically designed to attack our method and show how our proposed method is robust to that (*Refer to appendix*).

2 Our Proposed APuDAE Framework

In this section, we present our proposed framework called *Adversarial Purification using Denoising AutoEncoders* (**APuDAE**) which shows how Denoising AutoEncoders (DAE), trained over a dataset of clean images, are capable of purifying adversaries.

2.1 Denoising AutoEncoders (DAE)

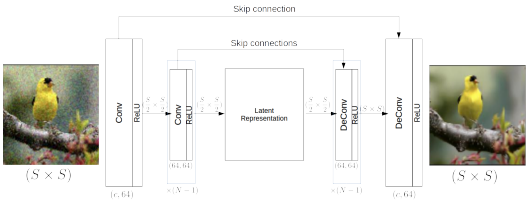


Figure 1: DAE Model Architecture for CIFAR-10 and ImageNet. c = Number of Channels in the Input Image ($c = 3, N = 15$ for CIIFAR-10 and $c = 1, N = 5$ for MNIST)

(a) airplane (b) truck (c) ship (d) airplane

Figure 2: Left to right : Input Image, Adversarial image, Directly purified image, Purified image by adaptive method. The sub-labels are the inferred classes from the target classifier network

Denoising AutoEncoder (DAE) is a variation of autoencoder in which the input is passed as a noisy image and the output is expected to be a clean image. The encoder learns to map noisy image to a latent representation corresponding to a clean image and decoder maps it back to the clean image. Training is carried out in unsupervised manner with the input image obtained by adding random gaussian noise with varying magnitudes and the output image is the clean image itself. The gaussian noise added is obtained with varying values of variance, σ. σ is chosen from a uniform distribution $[0, \sigma_{max}]$ so that the model is not biased to remove noise of specific magnitude. Formally, the loss functions is defined as given in Eqn 1 where the X_{noise} is the added noise, $F_{dae}(.)$ is the trained DAE which takes and outputs an image, mse is the mean square error loss. Skip connections are used to directly send the feature maps from an earlier layer to the later corresponding layer of the decoder to preserve the features. Network architecture for MNIST and imagenet dataset is given in Figure 1. The network architecture used for ImageNet is relatively more complex and is the same as proposed in (19), hence the readers are referred to it for more details.

$$
\begin{aligned}
&L = mse(F_{dae}(X_{inp}), X_{im}) \\
&\text{where, } X_{inp} = X_{im} + \sigma X_{noise}, \\
&\text{and } \sigma \sim U[0, \sigma_{max}], \;\; x_{noise} \sim \mathcal{N}(0, 1) \;\; (\forall x_{noise} \in X_{noise})
\end{aligned}
\tag{1}
$$

2.2 Determining Reconstruction Errors

Let X be the input image and X_{rcn} is the reconstructed image obtained from the trained DAE. We define the reconstruction error or the reconstruction distance as $\texttt{Recon}(X) = (X - X_{rcn})^2$. Two pertinent points to note here are:- 1) For clean test examples, the reconstruction error is bound to be less since the DAE is trained with unknown noise magnitude, hence DAE should ideally leave the input image unaffected. 2) For the adversarial examples, as they contain the adversarial noise, passing the adversarial image leads to removal of some part of the adversarial noise, hence leading to high reconstruction distance. We use this peculiar difference to our advantage by modifying the input image itself so that the reconstruction error reduces and leads to a clean image. We do this instead of directly taking the output image from DAE as the purified image i.e. $X_{pur} := X_{rcn}$. It has the following advantages over the later method :- 1) More image features are preserved 2) The purification function is non differentiable, hence it becomes difficult to craft adaptive attacks As an example, let the clean image be an image of a airplane and its slightly perturbed image fools the classifier network to believe it is a truck. Hence, the input to the DAE will be the slightly perturbed airplane image with the predicted class truck. Now, on passing the perturbed image directly through DAE, adversarial perturbations are removed but some image features are also perceived as noise and also removed. This leads to target classifier misclassifying the modified image as a ship (see Figure 2).

2.3 Variations in the adaptive algorithm

We use the following update rule for iteratively purifying the image :-

$$
\begin{aligned}
&X_{pur,0} = X_{adv} \\
&X_{pur,i+1} = X_{pur,i} - \alpha_i w_i \\
&\text{where, } w_i = \beta w_{i-1} + (1-\beta)\frac{\partial L(X_{pur,i}, purifier(X_{pur,i}))}{\partial X_{pur,i}} \\
&\text{for } i \in \{1, 2..., n\} \text{ and} \\
&L(X,Y) = \frac{|dist(X,Y) - \mu|}{\sigma},\ dist(X,Y) = (X-Y)^2
\end{aligned}
\tag{2}
$$

Based on this, we explore the following variations to the algorithm :-

(a) **Fixed no of iterations :** Set $\alpha_i = \alpha_{const}$, $n = n_{const}$ and $\beta = 0$.

(b) **Fixed no of iterations with ADAM for update :** Set $\alpha_i = \alpha_{const}$, $n = n_{const}$ and $\beta = \beta_{const}$. This helps to achieve the minima quickly hence improves the classification accuracy by a certain margin.

(c) **Variable learning rate based on Current Reconstruction Error :** Set $\alpha_i = \alpha(1 - e^{-(\frac{X_{pur,i}-\mu}{\sigma})^2})$. With this, the update is differentiated for adversaries and clean examples and hence achieves slightly better adversarial accuracy and less affected clean accuracy.

(d) **Set Target Distribution for Reconstruction Error :** Change $L(X,Y) = \frac{max(dist(X,Y)-\mu, 0)}{\sigma}$ where μ and σ are the mean and co-variances of the reconstructions errors in the training set. This helps essentially differentiating between clean and adversarial accuracy, hence clean accuracy remains nearly unaffected with this.

(e) **Set Target Distribution for Reconstruction Error with Modified Update Rule :** Drawback of the above method is that it tries to increase the reconstruction error (see Figure 3c) of the samples with less reconstruction error belonging to clean image set. To avoid this, we change the update rule by just modifying the loss function, $L(X,Y) = \frac{max(dist(X,Y)-\mu, 0)}{\sigma}$ which ultimately leads to no change for clean images with reconstruction error less than μ.

(f) **Adding random noise before each step** : Add $X_{pur,i} := X_{pur,i} + \gamma r_X, r_X\ \mathcal{N}(0, I_X)$ before the update step where γ is the amount of noise to be added.

(g) **Adding random transformation before the algorithm** : Change the initialization step as $X_{pur,0} = t(X_{adv}, f, \theta)$ where t is the transformation function which takes the resize factor f and θ as input.

3 Experimental Results

We present the esperimental results on MNIST, CIFAR-10 and ImageNet datasets for different white-box attacks. Additional results for varying values of hyperparameters like the no of iterations, n, noise γ added before each step, transformation parameters etc. are also reported in the **Appendix**.

3.1 MNIST Dataset

We present the comparison in results for MNIST dataset (20) with the 2 methods discussed earlier MagNet (17) and DefenseGAN (18)(see Table **??**). Results for adversarial training are generated in the same way as described in MagNet. All results are with $\epsilon = 0.1$ as used commonly in most works (18; 21). We use no. of iterations for purification, $n = 15$ and $\alpha = 0.01$.

Attack (ϵ=0.1)	**No Training**	**Adversarial training**	**DefenseGAN**	**MagNet**	**APuDAE** (Direct)	**APuDAE** (n=15,α=0.01)
Clean	0.974	0.822	0.969	0.95	0.973	0.973
FGSM	0.269	0.651	0.949	0.69	0.934	0.964
R-FGSM	0.269	0.651	0.945	0.65	0.921	0.962
PGD	0.1294	0.354	0.939	0.49	0.912	0.951
CW	0.	0.28	0.936	0.45	0.907	0.953

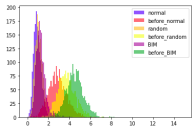
Table 1: Comparison of Results using **MNIST** Dataset. Values reported are classifier success rate (in fraction out of 1). **APuDAE (Direct)**: When the image directly obtained from DAE is used. **APuDAE**: When back propagation through reconstruction error is used for purification

3.2 CIFAR-10 Dataset

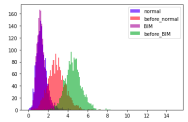
MNIST dataset is a simple dataset with easily identifiable number shapes which can be robustly learned by a neural network. It is therefore very easy to correct a perturbed image with a simple back-propagation method with fixed no. of iterations. For CIFAR-10 dataset, due to the inherent complexity in classifying objects, it is very easy to attack and thus simple defense with fixed no. of iterations is not enough to give a reasonably good purification result, hence we explore different variations in algorithm as tabulated in Table **??**.

Attack	**Clean**	**Random**	**FGSM**	**R-FGSM**	**BIM**	**CW**
No Defense	0.954	0.940	0.533	0.528	0.002	0
Adversarial Training	0.871	0.865	0.650	0.640	0.483	0.412
APuDAE (Direct)	0.790	0.793	0.661	0.655	0.367	0.351
APuDAE (A, n=12)	0.879	0.889	0.73	0.73	0.632	0.621
APuDAE (A, n=15)	0.853	0.875	0.76	0.747	0.694	0.683
APuDAE (A, n=18)	0.843	0.867	0.774	0.764	0.702	0.695
APuDAE(B)	0.872	0.883	0.778	0.772	0.706	0.698
APuDAE(C)	0.892	0.874	0.769	0.762	0.692	0.689
APuDAE(D)	0.904	0.889	0.783	0.767	0.684	0.679
APuDAE(E)	0.907	0.893	0.834	0.817	0.702	0.689
APuDAE(F)	0.908	0.893	0.818	0.811	0.729	0.708
APuDAE(G)	0.858	0.846	0.828	0.814	0.711	0.700

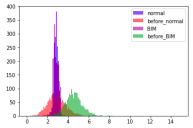
Figure 3: Comparison of Results for Different Variations on **CIFAR-10** dDataset. **Direct**: Direct usage of DAE, **A**: Fixed no. of iterations, **B**: Fixed no. of iterations with using ADAM optimizer, **C**: Variable learning rate based on the current reconstruction error, **D**: Set target distribution for reconstruction error, **E**: Set target distribution for reconstruction error with modified update rule, **F**: Add random noise at each update step, **G**: Add random transformation at the beginning



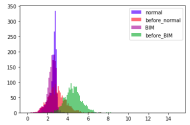
(a) Fixed no. of iterations



(b) Fixed no. of iterations with using ADAM optimizer



(c) Set target distribution for reconstruction error



(d) Set target distribution for reconstruction error with modified update rule

Figure 4: Reconstruction errors for different variations

3.3 ImageNet Dataset

ImageNet dataset is the high resolution dataset with cropped RGB images of size 256X256. For our experiments we use pretrained weights on ImageNet train corpus available from (22). For test set we use the 1000 set of images available for ILSVRC-12 challenge (23) which is commonly used by previous works (24) for evaluating adversarial attack and defense methods. The classification accuracy for ImageNet dataset comprising of 1000 classes is defined by the top-1 accuracy which means a prediction is correct if any of the top 1% of total classes or 10 class predictions in this case are correct. The value of $\epsilon = \frac{8}{255}$ and update step, $\alpha = \frac{1}{255}$ are chosen as standard values similar to ones used in previous literature (24)

Attack	Clean	Random	BIM $\left(\epsilon = \frac{5}{255}\right)$	BIM $\left(\epsilon = \frac{25}{255}\right)$
No defense	1.0	0.969	0.361	0.002
Adv. Training	0.912	0.901	0.641	0.454
APuDAE (Direct)	0.946	0.941	0.909	0.899
APuDAE (A,$n = 12$**)**	0.941	0.936	0.919	0.911
APuDAE (A,$n = 15$**)**	0.939	0.933	0.917	0.914
APuDAE (A,$n = 18$**)**	0.934	0.932	0.917	0.913
APuDAE (B,$n = 15$**)**	0.938	0.930	0.923	0.920
APuDAE (C,$n = 15$**)**	0.926	0.924	0.928	0.929

Table 2: Comparison of Results on Imagenet Dataset. **Direct**: Direct usage of DAE, **A**: No variation, **B**: With random noise, **C**: With random transformations. ϵ is the magnitude of adversarial perturbation and n is the no. of iterations of APuDAE for purification.

References

[1] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: Chances and challenges," in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, 2018, pp. 35–38.

[2] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[3] P. Yang, D. Baracchi, R. Ni, Y. Zhao, F. Argenti, and A. Piva, "A survey of deep learning-based source image forensics," *Journal of Imaging*, vol. 6, no. 3, p. 9, 2020.

[4] 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.

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

[6] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[8] A. Lamb, V. Verma, J. Kannala, and Y. Bengio, "Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy," *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019.

[9] F. Li, X. Du, and L. Zhang, "Adversarial attacks defense method based on multiple filtering and image rotation," *Discrete Dynamics in Nature and Society*, vol. 2022, pp. 1–11, 04 2022.

[10] R. Tran, D. Patrick, M. Geyer, and A. Fernandez, "Sad: Saliency-based defenses against adversarial examples," *ArXiv*, vol. abs/2003.04820, 2020.

[11] E. Raff, J. Sylvester, S. Forsyth, and M. McLean, "Barrage of random transforms for adversarially robust defense," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6521–6530.

[12] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," *arXiv preprint arXiv:1608.00530*, 2016.

[13] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5764–5772.

[14] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[15] R. Gao, F. Liu, J. Zhang, B. Han, T. Liu, G. Niu, and M. Sugiyama, "Maximum mean discrepancy test is aware of adversarial attacks," in *International Conference on Machine Learning*, 2021, pp. 3564–3575.

[16] S. Jha, U. Jang, S. Jha, and B. Jalaian, "Detecting adversarial examples using data manifolds," in *IEEE Military Communications Conference (MILCOM)*, 2018, pp. 547–552.

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

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

[19] S. Laine, T. Karras, J. Lehtinen, and T. Aila, "High-quality self-supervised deep image denoising," in *NeurIPS*, 2019.

[20] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ArXiv*, vol. abs/1706.06083, 2018.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[24] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2017.

[26] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE symposium on security and privacy (S&P)*, 2017, pp. 39–57.

[27] Y. Li, W. Jin, H. Xu, and J. Tang, "Deeprobust: A pytorch library for adversarial attacks and defenses," 2020.

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

Appendix

A Possible Counter Attacks

Based on the attack method, adaptive attacks can be developed to counter the attack. We study the 2 types of attacks in detail. For a detailed review on how to systematically design an adaptive counter attack based on the category of defense, readers are referred to (19).

A.1 Counter Attack A

The first adaptive attack possible is designed by approximating the transformation function i.e. the function of the output image obtained by modifying the input image through the update rule, by the differentiable function obtained by autoencoder end-to-end network. Intuitively this can be thought of as the output obtained by backpropagating through the reconstruction error loss between the input and purified output after passing through autoencoder purifier network. We observe (see Table 3) that on applying this counter attack, the accuracy for direct purifier output method drops drastically while ours method gives better robust results against this attack. Mathematically, we can express the attacked image, $X_{attacked}$ as follows where X_{cln} is the original clean image, n is the no. of iterations, $classifier(.)$ is the target classifier network, $J(.)$ is the cross entropy loss, $purifier(.)$ is the purifier autoencoder, α is the update rate and $\pi_{X,\epsilon}(.)$ function restricts value within $[X-\epsilon, X+\epsilon]$.

$$\begin{aligned} &X_{attacked,0} = X_{cln} \\ &X_{attacked,i+1} = \pi_{X_{cln},\epsilon}[X_{attacked,i}+ \\ &\qquad \alpha.sign(\frac{\partial J(classifier(X_{attacked,i}), y)}{\partial X_{attacked,i}})] \\ &\text{for } i \in \{1, 2..., n\} \end{aligned} \tag{3}$$

A.2 Counter Attack B

The second adaptive attack possible here is by modifying the loss function of the BIM attack by including new weighted term getting less reconstruction error from the purifier This way we attack both the classifier as well as purifier. The purifier method relies on reconstruction error as a measure to update the input to get less reconstruction error similar to clean images but if the attack is made with this consideration to fool the purifier method by giving similar reconstruction error to clean image while also fooling the classifier, the attack is successful as it bypasses the purifier method. For this attack, the attacked image seems to be attacked more at the edges as modifying those do not change the reconstruction error for purifier much. As observed from Table 3, the adaptive counter attack is successful to an extent but still performs considerably better than adversarial training. Mathematically, we can express the attacked image, $X_{attacked}$ as follows where X_{cln} is the original clean image, n is the no. of iterations, $classifier(.)$ is the target classifier network, $J(.)$ is the cross entropy loss, $purifier(.)$ is the purifier autoencoder, α is the update rate, β is the weighing factor for the reconstruction error in the combined loss function, and $\pi_{X,\epsilon}(.)$ function restricts value within $[X-\epsilon, X+\epsilon]$.

$$\begin{aligned} &X_{attacked,0} = X_{cln} \\ &X_{attacked,i+1} = \pi_{X_{cln},\epsilon}[X_{attacked,i} + \alpha sign(\\ &\qquad \frac{\partial}{\partial X_{attacked,i}}(J(X_{attacked,i}, y)+ \\ &\qquad \beta(X_{attacked,i} - purifier(X_{attacked,i}))^2)] \\ &\text{for } i \in \{1, 2..., n\} \end{aligned} \tag{4}$$

B Adversarial Attack Models and Methods

For a test example X, an attacking method tries to find a perturbation, ΔX such that $|\Delta X|_k \leq \epsilon_{atk}$ where ϵ_{atk} is the perturbation threshold and k is the appropriate order, generally selected ∞ so that the newly formed perturbed image, $X_{adv} = X + \Delta X$. Here, each pixel in the image is represented by the $\langle \mathtt{R}, \mathtt{G}, \mathtt{B} \rangle$ tuple, where $\mathtt{R}, \mathtt{G}, \mathtt{B} \in [0, 1]$. In this paper, we consider only white-box attacks, i.e. the attack methods which have access to the weights of the target classifier model.

B.1 Random Perturbation (RANDOM)

Random perturbations are simply unbiased random values added to each pixel ranging in between $-\epsilon_{atk}$ to ϵ_{atk}. Formally, the randomly perturbed image is given by,

$$X_{rand} = X + \mathcal{U}(-\epsilon_{atk}, \epsilon_{atk}) \tag{5}$$

Dataset	Counter Attack	Adversarial Training	APuDAE (Direct)	APuDAE (Best)
MNIST	A	0.354	0.799	0.891
	B	0.354	0.815	0.857
Cifar-10	A	0.483	0.199	0.680
	B	0.483	0.275	0.507
ImageNet	A	0.254	0.134	0.536
	B	0.254	0.312	0.407

Table 3: Comparison of Results for Counter Attacks. For adversarial training, results of plain BIM attack have been reported for comparison of performance on worst suspected possible attack with our method.

where, $\mathcal{U}(a, b)$ denote a continuous uniform distribution in the range $[a, b]$.

B.2 Fast Gradient Sign Method (FGSM)

Earlier work by (5) introduced the generation of malicious biased perturbations at each pixel of the input image in the direction of the loss gradient $\Delta_X L(X, y)$, where $L(X, y)$ is the loss function with which the target classifier model was trained. Formally, the adversarial examples with ϵ_{atk} are computed as :-

$$X_{adv} = X + \epsilon_{atk}.sign(\Delta_X L(X, y)) \tag{6}$$

B.3 Projected Gradient Descent (PGD) or BIM

Earlier works by (25) propose a simple variant of the FGSM method by applying it multiple times with a rather smaller step size than ϵ_{atk}. However, as we need the overall perturbation after all the iterations to be within ϵ_{atk}-ball of X, we clip the modified X at each step within the ϵ_{atk} ball with l_∞ norm.

$$X_{adv,0} = X, \tag{7a}$$

$$X_{adv,n+1} = \texttt{Clip}_X^{\epsilon_{atk}}\Big\{X_{adv,n} + \alpha.sign(\Delta_X L(X_{adv,n}, y))\Big\} \tag{7b}$$

Given α, we take the no. of iterations, n to be $\lfloor \frac{2\epsilon_{atk}}{\alpha} + 2 \rfloor$. This attacking method has also been named as Basic Iterative Method (BIM) in some works.

B.4 Carlini-Wagner (CW) Method

(26) proposed a more sophisticated way of generating adversarial examples by solving an optimization objective as shown in Equation 8. Value of c is chosen by an efficient binary search. We use the same parameters as set in (27) to make the attack.

$$X_{adv} = \texttt{Clip}_X^{\epsilon_{atk}}\Big\{\min_{\epsilon} \|\epsilon\|_2 + c.f(x + \epsilon)\Big\} \tag{8}$$

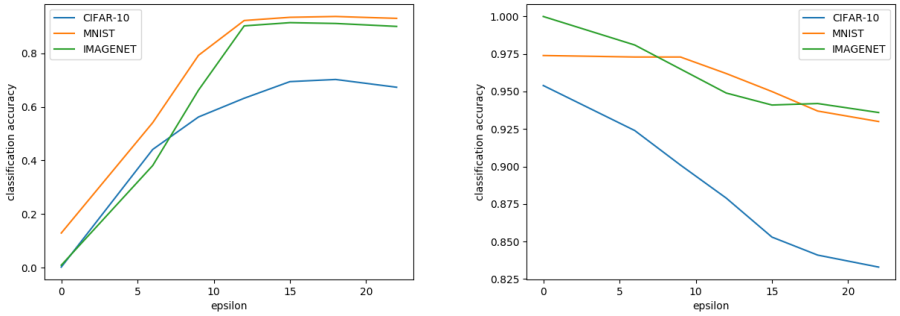
B.5 DeepFool method

DeepFool (28) is an even more sophisticated and efficient way of generating adversaries. It works by making the perturbation iteratively towards the decision boundary so as to achieve the adversary with minimum perturbation. We use the default parameters set in (27) to make the attack.

C Varying Number of Iterations for Adaptive Purification.

On varying the number of iterations for the proposed method as discussed in Algorithm **??**, we observe the following trend:

- Classification accuracy for adversaries increases up to a certain point with increasing iterations and then we observe a downward trend while the clean accuracy reduces or nearly stays constant at the beginning with increased no. of iterations n. This trend an be attributed to the fact that with increased no. of iterations there is more purification resulting to increased accuracy for adversaries as they are corrected.
- At the same time purification leads to destruction of some image features as well as they are perceived as noise. Hence, the clean accuracy decreases, and also for adversarial samples, the loss of image features overshadows the reduction of adversarial noise. However, it is empirically observed that the peak on adversarial accuracy occurs at around $n = \frac{1.5\epsilon}{\alpha}$ which can be interpreted as it takes about 1.5 times the correction effort to neutralize an attack of magnitude ϵ with step α.

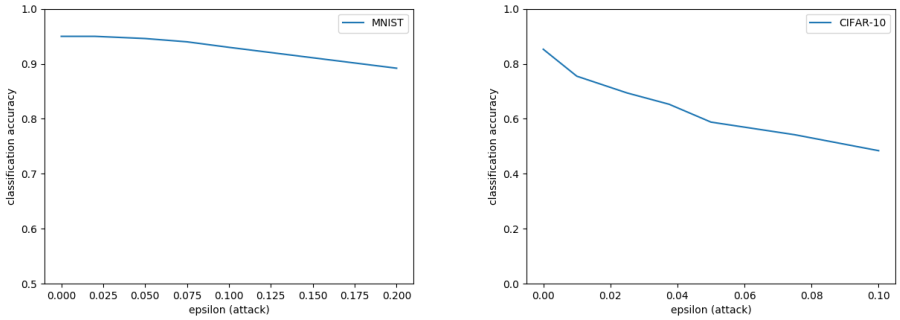


(a) BIM adversaries (b) Clean samples

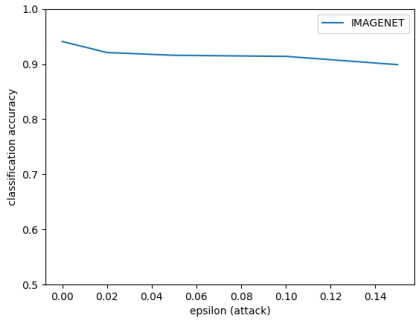
Figure 5: Classification accuracies after purification for different no. of iterations. $\epsilon = 0.1, 0.0314, 0.1$ for MNIST, CIFAR-10 and IMAGENET respectively.

D Varying Amount of Adversarial Perturbation (ϵ).

We experiment with larger values of ϵ to make a larger attack to test the robustness of our method against different values of ϵ. For fairness, we set no. of iterations, $n = \frac{1.5\epsilon}{\epsilon_{step}}$ to test robustness against different values of ϵ for BIM attack. As seen in Figure 6, we observe a downward trend for increased magnitude of attack as expected. This is because larger ϵ attack usually implies larger adversarial perturbation and thus hard to purify.



(a) MNIST (b) CIFAR-10

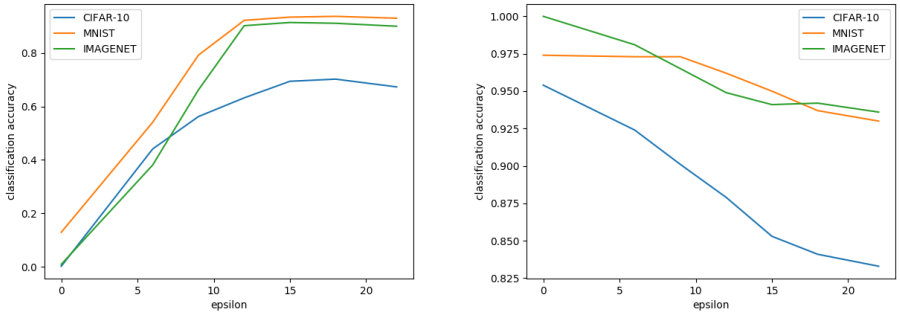


(c) IMAGENET

Figure 6: Classification accuracies after purification for adversaries with different magnitudes of attack.

E Varying Amount of Random Transformation Parameters at Every Step (ImageNet).

Further, we experiment with different values of random transformation parameters, rotation θ and resize factor f. As observed with $f = 1$ constant, on increasing θ, we first observed an upward trend as the rotation acts as a defense in the beginning (see Figure 7a) leading to increased accuracy but later it reduces. Similar trend is observed for different values of f for no rotation (see Figure 7b).



(a) Variable θ, fixed resize factor, $f = 1.1$ (b) No rotation, variable resize factor, f

Figure 7: Classification accuracies after purification for different transformation parameters.