**ADVERSARIAL ATTACK ON NEURAL NETWORKS**

An´ıbal Guerrero Hern´andez; Ignacio Zu´n˜iga Mart´ınez Student, Group 5

Technical University of Munich Professorship of Multiscale Modeling of Fluid Materials

Boltzmannstraße 15, 85747 Garching

Lab course “Hands-on Deep Learning”

**Abstract: This report investigates the vulnerability of Convolutional Neu- ral Networks to adversarial attacks, using the MNIST dataset and compar- ing two architectures, LeNet and ResNet-18. The study finds that CNN’s dependency on linearity for ease of trainability makes them susceptible to adversarial attacks, highlighting the importance of considering the trade- off between adversarial resistance and ease of model training. The research also demonstrates the significance of adversarial examples in exposing algo- rithmic blind spots and the limitations of deep learning models. Adversarial training is shown to partially address the problem, but the study suggests that the model families used in the research may be intrinsically flawed. This motivates the development of optimization procedures that can train models with more locally stable behavior. Overall, this study highlights the importance of considering the robustness of the models and the need for further research in developing more resilient ones.**

1. **Academic Interest**

# Introduction

surprising number of machine learning models, including cutting-edge neural net- works, are susceptible to adversarial examples [1]. This reveals important algo- rithmic blind spots where these models classify examples that are only slightly different from correctly classified examples, incorrectly. Speculative theories have claimed that the very nonlinear nature of deep neural networks, in combination with inadequate model averaging and inadequate regularization of the solely supervised learning issue, is what causes these adversarial cases, which have remained a mystery. Adversarial examples are generated by linear behavior in high-dimensional spaces, and adversarial training can enhance further regularization beyond that provided by dropout’s regu- larization benefits [2]. Changing to nonlinear model families such as RBF networks

A

can significantly reduce the model’s vulnerability to these attacks. [1]

Current tensions lie between designing models that are easy to train and resistant to adversarial perturbation. The model’s training ease reflects its linearity, whereas its resistance to perturbation results from its non-linearity.

Through *Szegedy’s* extensive work on adversarial examples, it has been proven how modern machine learning techniques, even those with superior performance, do not learn the true underlying concepts that determine the correct output label [3]-[4]. In- stead, these models build a *Potemkin village* [5]. It was named for purported fake set- tlements erected at the direction of Russian minister *Grigori Aleksandrovich Potemkin* to fool Empress Catherine II during her visit to *Crimea* in *1787*. A Potemkin village

refers to a false construct devised to disguise a shortcoming or improve appearances (Fig. 1).

The distinction between a Potemkin village from a true improvement is almost imperceptible with naturally occurring data. However, adversarial examples reveal the true nature of the supposed model improvements, as one visits points in space that do not have a high probability in the data distribution. The discovery proved a disappointment, particularly in computer vision, as convolutional neural network features are utilized to approximate perceptual distance using Euclidean distances. However, this flaw can be fixed with adversarial training. In contrast, the cost of non-linearity comes at the expense of accuracy on clean inputs and computational costs.

## Linearity of adversarial examples

Limited computational power leads to limited computational precision. Therefore, some information will inevitably be lost. In the case of digital images, only 8 bits per pixel are used, disregarding all information below 1*/*255 of the dynamic range. Take an input *x* and an adversarial input *x*˜ = *x* + *η*, where *η* is a perturbation. If the perturbation is smaller than the precision of the features, the classifier will not be able to differentiate the two: *||η||∞ < ϵ*. Thus, leading the classifier to assign the same class, where *ϵ* is small enough to be disregarded by the sensor associated with the problem.

Consider the following product between a weight vector *ω* and an adversarial exam- ple, *x*:

*ωT x*˜ = *ωT x* + *ωT η* (1)

The adversarial perturbation causes the activation to grow by *ωT η*, which can be maximized when *η* is assigned as the sign of *ω*. As *||η||∞* does not grow with the dimensionality of the problem, many infinitesimal changes can be made to the input, which will add up to one large change to the output. As a result, a simple linear model can have adversarial examples if its input has sufficient dimensionality, leading to accidental steganography (Fig. 2).

Let *θ* be the parameters of a model, *x* the input, *y* the targets, and *J*(*θ, x, y*) the cost used to train the neural network. The cost function can be linearized around the current value of *θ*, obtaining an optimal max-norm constrained perturbation of *η* = *ϵsign* (*∇xJ*(*θ, x, y*)).

Referred to as the **Fast Gradient Sign Method (FGSM)** of generating adversar- ial examples, the required gradient can be computed efficiently using backpropagation.

# Methods

## Deep Learning Architecture

### LeNet Architecture

LeNet is a classic Convolutional Neural Network structure proposed by *LeCun et al.* in *1998* [6]. It consists of 7 layers (Fig. 3). The input is a grayscale image of size 28x28 pixels. The first 4 layers are - a convolutional layer C1 with 6 convolution kernels of 5*x*5 and a size of feature mapping of 28x28, followed by subsampling (pooling layer) S2 that outputs 6 feature graphs of size 14*x*14. Between each convolutional and subsampling layer, there is a sigmoid function, same as between the fully connected layers. Then, a convolutional layer C3 with 16 5x5 convolution kernels, and subsampling layer S4 of size 2x2 and 16 5x5 convolution kernels. Layer F5 is a fully connected convolution layer with 120 convolution kernels of size 5x5. Each cell is connected to the 5x5 neighborhood on all 16 feature graphs of S4. Layer F6 is another fully connected layer with 84 kernels, leading to the flattened output layer of 10 units. These correspond to the 10 classes in the MNIST dataset and employs a softmax activation function to output class probabilities.

### ResNet-18 Architecture

ResNet-18 (Fig. 4) is a much deeper and more complex CNN architecture than LeNet. It consists of 18 layers and uses residual blocks that allow for the training of very deep networks while avoiding the problem of *vanishing/exploding* gradients. Initially, the input to the Network is a color image of size 224x224 pixels and is passed through a series of convolutional layers with batch normalization and ReLU activation functions. Each block consists of 4 convolutional layers and a shortcut connection in between. The shortcut connection allows for the direct flow of information through the network. The output is passed through a global average pooling layer, which averages the feature maps across spatial dimensions. Finally, the output layer is passed through a fully connected layer with a softmax activation function to output class probabilities.

This architecture is well-suited for classifying more complex datasets like ImageNet, which contains images with much higher complexity and a larger number of classes than MNIST.

### Comparing Architectures

On one hand, LeNet uses a *tanh* activation function which is uncommon among modern architectures. This activation function has a higher computational cost than ReLU.

On the other hand, ResNet-18 uses batch normalization and dropout regularization, preventing overfitting during training. Batch normalization normalizes the output of each layer to have 0 mean and unit variance, while dropout randomly drops out units during training to prevent co-adaptation of neurons.

### Loss Function

Both architectures use the cross-entropy loss function, which is commonly used for classification tasks. It measures the difference between the predicted class probabilities and the true class labels and is optimized during training using *backpropagation*.

## Dataset

### MNIST

The first selected dataset is the well-known MNIST (Fig. 5) [7]. MNIST is a dataset composed of 60000 handwritten digit images for training and another 10000 handwrit- ten digits for testing. The images have a resolution of 28x28 pixels and only one color channel (black & white) and have been size-normalized and centered in a fixed-size image. MNIST is among the most common datasets for training Neural Networks on classification problems. The dataset has been split into a training set containing 59000 images and a validation set of 1000 images.

### CIFAR-10

The second selected dataset is CIFAR-10 [8] (Fig. 10). CIFAR-10 is a dataset com- posed of 50000 colored images for training and another 10000 images for testing. The images have a resolution of 32x32 pixels and only 3 color channels (RGB). The dataset has been split into a training set of 49000, a validation set of 1000, and a test set of 10000 images.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | plane | car | bird | cat | deer | dog | frog | horse | sheep | truck |
| Class ID | 0 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 |

Table 1: All CIFAR-10 classes and their corresponding label.

# Results

## Performance on MNIST Dataset

Before we attack each architecture, both are trained and the corresponding accuracy is measured.

|  |  |
| --- | --- |
| Architecture | Accuracy |
| LeNet | 0.981 |
| ResNet-18 | 0.991 |

Table 2: Accuracy of the two architectures prior to adversarial attacks.

## Results

### MNIST Dataset

Comparing the accuracy decrement caused by the attack on both networks, a counter- intuitive result shows off (Fig. 7). The ResNet-18 accuracy loss is higher for inter- mediate *ϵ* values than with LeNet’s architecture. However, for higher *ϵ* values, both accuracies tend to 0.

Usually, more complex CNNs tend to achieve better results due to their complexity and non-linearity. Anyhow, during an adversarial attack, this trend is reversed. A higher number of parameters learn patterns quicker, but will also make the model more vulnerable, as there are more parameters to be exploited by the attacker.

In this case, the ResNet-18 architecture is significantly more complex and deeper than LeNet, there are more opportunities to exploit the parameters and create ad- versarial examples. Meanwhile, LeNet’s simpler architecture difficults the attacker’s attempts to find perturbations. To be noted is that, whereas LeNet has been retrained with the MNIST dataset, ResNet-18 was pre-trained with ImageNet’s dataset.

As expected, when *ϵ* increases, the accuracy of both models tends to decrease. Larger *ϵ* values take larger steps in the direction that maximizes the loss. The trend in the curve (Fig. 7) is non-linear even though *ϵ* values are linearly spaced.

On the other hand, perturbations make adversarial attacks visible. These pertur- bations become visible, as soon as *ϵ ≥* 0*.*1, where image fidelity decreases noticeably. Therefore, the attacker must make a trade-off between accuracy degradation and per- ceptibility. Nevertheless, when considering (Fig. 8), humans are always capable of distinguishing the correct class, despite the additional noise.

Knowing the MNIST dataset is composed of gray-scale images with values between [0*,* 1], where 0 corresponds to black and 1 corresponds to white. Because the adversarial attack adds *η* to the value of the original input, the adversarial input images must be clipped, so its information lays bound to the former range.

### CIFAR-10 Dataset

Additionally, the ResNet-18 has been trained on the CIFAR-10 dataset. Looking at (Fig. 9), we can conclude that for colored images the network is more sensitive to attacks. The accuracy loss curve has a similar shape but higher losses appear for lower *ϵ* values.

Besides the network’s initial accuracy being lower, small *ϵ* values decrease accuracy significantly to less than 50%. This worsening in accuracy occurs because the dataset has more complex features. This results challenging to learn robust representations of the data. Therefore, the network is more susceptible to attacks. Additionally, the previously mentioned pre-training component is inherited, which can negatively affect the results.

Furthermore, colored images must be clipped to the training range. In CIFAR-10’s case, the channels have been normalized between [0*,* 1]. However, for other datasets, a [0*,* 255] range clipping may be necessary.

Some examples of misclassified images are represented in (Fig. 10). With the CIFAR-10 dataset, the attack becomes less noticeable, but more accuracy is lost. For *ϵ ≥* 0*.*03, the accuracy is *∼* 0*.*15, images are recognizable and background noise is still difficult to notice.

# Further Discussion

The vulnerability of CNNs has been proven with the former work. Assessing how the network can defend against a *FGSM* attack, the following pop-up as possible solutions:

* Include adversarial examples on training set - Not only will the adversarial ex- amples make the model more robust by providing further examples, but as the adversarial input increases compared to the original input, Eqn. 1 demonstrates that the loss function would increase. Therefore, the gradient of the loss func- tion increases, and the confidence with which the model predicts the outcome, increases correspondingly.
* Randomizing images - Randomization involves adding random noise to the input data during training or inference. By that, the attacker would need to overcome the added noise in addition to the underlying model. Randomization can be applied in different ways, such as adding noise to the input image, randomizing the weights of the neural network, or randomizing the decision boundaries of the network.
* Defensive distillation - Trains a *distilled* version of the CNN, where the output probabilities of the original network are used as soft targets for the distilled network. Then, it is trained by combining the original training data and the adversarial examples. It helps smooth out the decision boundary of the CNN, making it less susceptible to adversarial perturbations. However, some latest papers are opposed to defensive distillation performance against attacks [9].
* Gradient masking - Modifies the gradient of the loss function during the back- propagation of the CNN, preventing the attacker from computing the exact gra- dient needed to generate an effective adversarial example. This is achieved by adding noise to the gradient or by modifying the gradient in some other way.

# Conclusion

The vulnerability of CNNs has been demonstrated to occur due to the network’s de- pendency on linearity for ease of trainability. A deep trade-off between adversarial resistance and ease of model training must be considered. Furthermore, adversarial examples have proven to be of notorious importance in *Deep Learning*, as these net- works are volatile and show off algorithmic blindspots, as they do not understand the true nature of what is being observed.

On top of that, the vulnerability of networks, regardless of the complexity and depth of models against adversarial attacks, has been verified. Showing how not only older models but also more recent ones, are susceptible to adversarial attacks.

This work demonstrates how researchers are capable of partially correcting the prob- lem with adversarial training. However, one may conclude that the model families we use are intrinsically flawed. This would motivate the development of optimization procedures that are able to train models whose behavior is more locally stable.

# Contributions

An´ıbal Guerrero Hern´andez: Section I, Subsection II.A, Subsection III.A & Section IV. Ignacio Zu´n˜iga Mart´ınez: Subsection II.B, Subsection III.B, Section IV, & Section V.

# References

1. I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.
2. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
3. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Van- houcke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
4. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfel- low, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.
5. M. David-Fox, “The myth of the soviet potemkin village,” *On* [*http://www.*](http://www/) *histoire. ens. fr/IMG/file/Coeure/David-Fox% 20P otemkin% 20villages. pdf*, 2013.
6. Y. LeCun *et al.*, “Lenet-5, convolutional neural networks,” *URL:* [*http://yann.*](http://yann/) *lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.
7. Y. LeCun, “The mnist database of handwritten digits.”
8. V. N. Alex Krizhevsky and G. Hinton, “The cifar-10 dataset.”
9. N. Carlini and D. A. Wagner, “Defensive distillation is not robust to adversarial examples,” *ArXiv*, vol. abs/1607.04311, 2016.
10. **Introduction**

# Appendix - Images

This appendix addresses the corresponding images referred to in Section I.



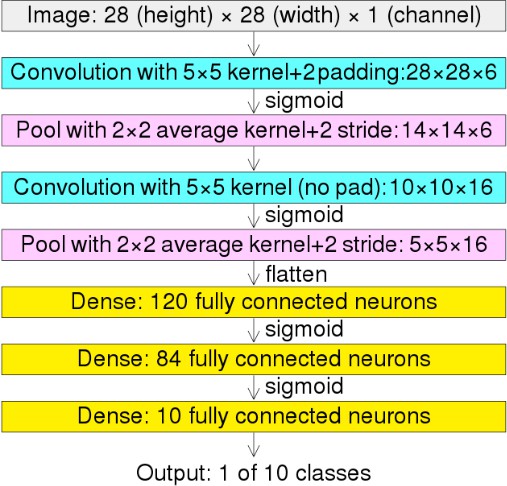
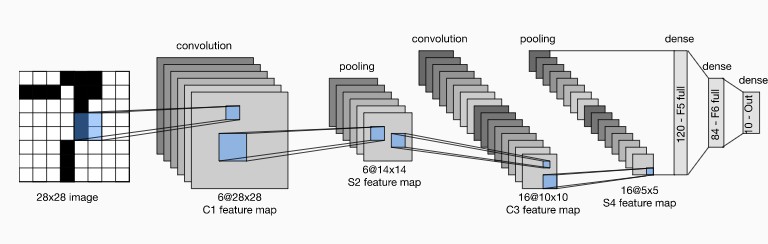
Figure 1: Example of a Potemkin village in reality.



Figure 2: Example of hiding one file within another through steganography.

## Architectures

This appendix addresses the corresponding images referred to in Subsection II.A.

* 1. LeNet’s CNN

layers.

* 1. Data flow in LeNet. The input is a handwritten digit. The output, a probability over 10 possible outcomes.

Figure 3: LeNet Architecture.

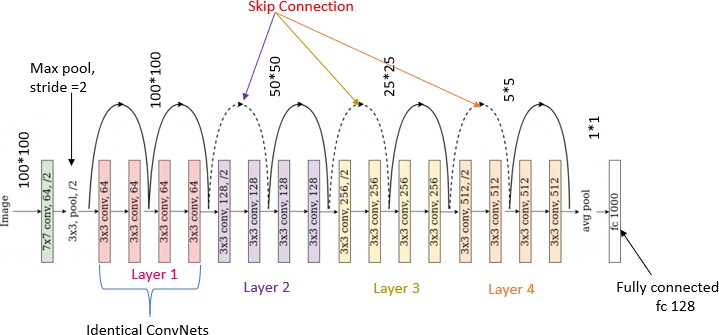


Figure 4: ResNet-18 Architecture.

## Datasets

This appendix addresses the corresponding images referred to in Section II.B.

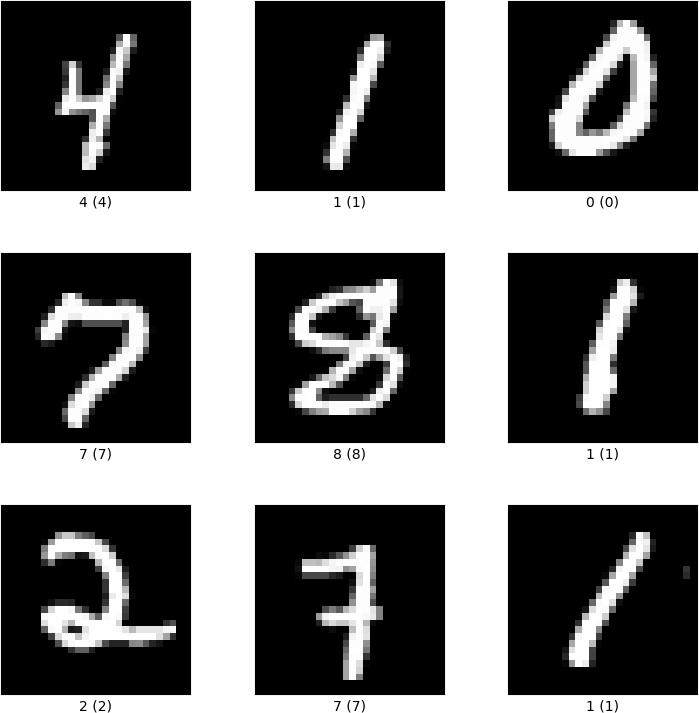


Figure 5: MNIST sample.

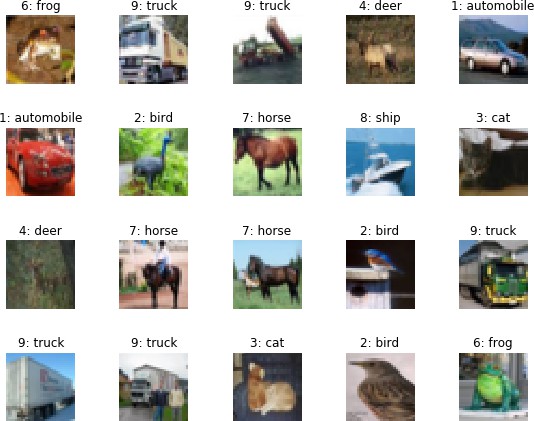


Figure 6: CIFAR-10 image samples and their corresponding labels.

## Results

This appendix addresses the corresponding images referred to in Section III.

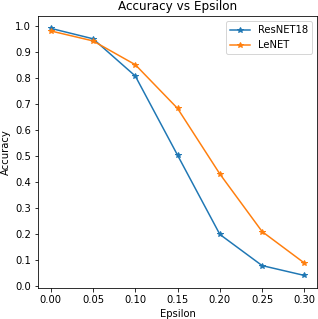


Figure 7: Accuracy vs epsilon for both models.

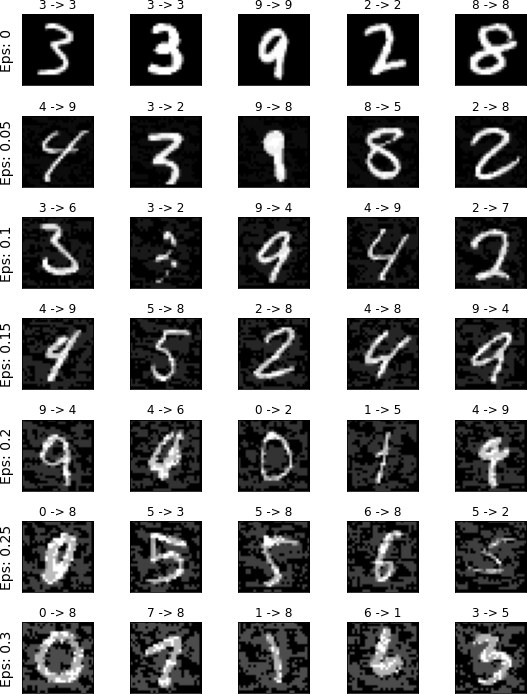


Figure 8: Missclassified image samples for different *ϵ* values (LeNet).

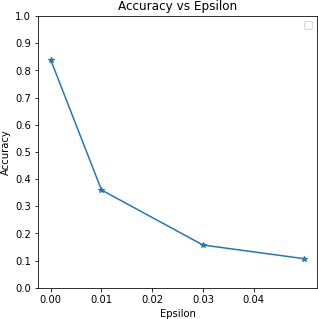


Figure 9: Accuracy vs *ϵ* for ResNet-18 on CIFAR-10.

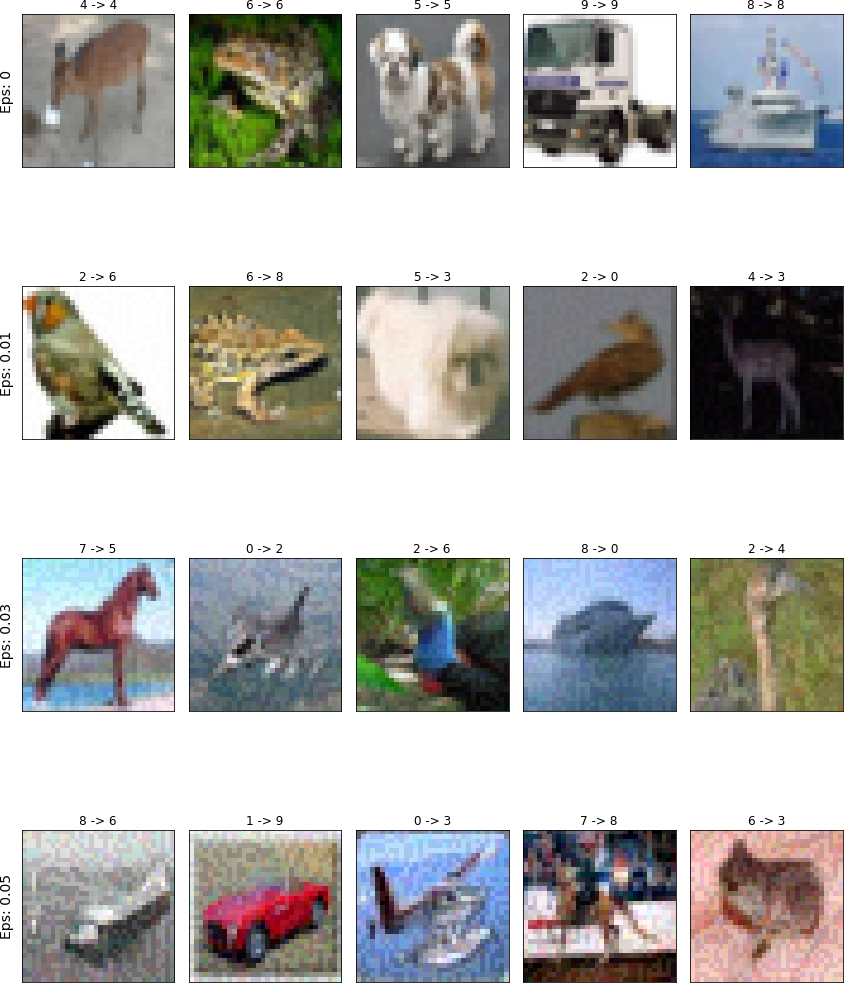


Figure 10: Sample of perturbed images of CIFAR-10 dataset.