Recreation of KDD 2017 Paper: Adversary Resistant Deep Neural Networks with an Application to Malware Detection

Team 3

Rahul Shukla (652959493)

Scott Brewer (676075252)

Video Link: https://youtu.be/4b2aloQ9ThI

INTRODUCTION

For this report we are recreating portions of a paper submitted to KDD 2017 that discussed a method for generating adversarial data and then proposed a method to defend against this adversarial data. In the paper, they proposed a new adversary resistant technique that obstructs attackers from constructing impactful adversarial samples by randomly nullifying features within samples. To demonstrate the general applicability of the proposed method the paper conducted experiments with a malware dataset to investigate the specific issue of defense against malicious software evading detection DNNs and also conduct experiments using the MNIST and CIFAR-10 datasets, generally used in image recognition research, to show that the method generalized to other domains. For our re-creation, we were unable to obtain the malware dataset, so instead focused on recreating the data from MNIST and CIFAR-10.

Deep Neural Networks (DNN) are vulnerable to adversarial samples, a flaw that plagues most if not all statistical samples. Recent research has shown that those with ill intent can easily circumvent deep-learning powered malware detection by exploiting this flaw. If an adversary can infer the learning model underlying an application, examine feature/class importance, and identify the features that have the greatest significant impact on correct classification, the adversary can, with minimal effort, craft an adversarial sample – a synthetic example generated by slightly modifying a real example in order to “fool” the deep learning system to “believe” this modified sample belongs to an incorrect class with high confidence.

This flaw has been widely exploited to fool DNNs trained for image recognition. Further, depending on the degree to which the image is modified, the modified image could look similar enough to the original to avoid suspicion from a human observer. In this paper, they proposed a new technical approach that can be empirically and theoretically guaranteed to be effective for malware detection and, more importantly, resistant to adversarial manipulation. Specifically, they introduced a random feature nullification (RFN) while training the models and while testing them, effectively making the resultant DNN model non-deterministic. This non-deterministic nature manifests itself when attackers attempt to examine feature/class importance or when a DNN model takes input for classification. As such, there is a much lower probability that attackers could correctly identify critical features for target DNN models. Even if attackers could infer critical features and construct a reasonable adversarial sample, the non-deterministic nature introduced in the model's processing of input will significantly reduce the effectiveness of these adversarial samples.

The simple approach proposed in this work is beneficial for a variety of reasons. First, it increases the difficulty for attackers to exploit the vulnerabilities of DNNs. Second, this adversary resistant DNN maintains desirable classification performance while requiring only minimal modification to existing architectures. Third, this technique could theoretically guarantee the resistance of Deep Learning to adversarial samples. Lastly, while the paper is primarily motivated by the need to safeguard DNN models used in malware detection, the proposed technique is general and can be adopted to other applications where deep learning is popularly applied, such as image recognition.

BACKGROUND

Even though a well-trained model can recognize out of-sample patterns, a deep neural architecture can be easily be fooled by introducing perturbations to the input samples that are often indistinguishable to the human eye. These so-called "blind spots", or adversarial samples, exist because the input space of DNN is too broad to be fully explored. Given this natural restriction, an adversary can uncover specific data samples in the input space and bypass DNN models with carefully crafted inputs . More specifically, attackers can find the most powerful blind spots through effective optimization procedures. In practice, such as with a multi-class classification task, these optimized adversarial samples can cause a DNN model to misclassify them, often into classes that would be judged as entirely incorrect by a human observer - such as an image of a horse being classified as a frog post attack.

Furthermore, DNN models that share the same design goal, for example recognizing the same image set, all approximate a common highly complex, nonlinear function. Therefore, a relatively large fraction of adversarial examples generated from one trained DNN will be misclassified by other DNN models trained from the same data set but with different hyper-parameters. Given a target DNN, they refer to adversarial samples that are generated from other different DNN models but still maintain their attack efficacy against the target as cross-model adversarial samples.

Adversarial samples are generated by computing the derivative of the cost function with respect to the network's input variables. The gradient of any input sample represents a direction vector in this high-dimensional input space. Along this direction, any small change of this input sample will cause a DNN to generate a completely different prediction result. This direction is important since it represents the most effective way to degrade the performance of a DNN. Discovering this particular direction is done by passing the layer gradients from the output layer all the way back to the input layer via backpropagation. The gradient at the input may then be applied to the input samples to craft an adversarial example.

FAST GRADIENT SIGN METHOD (FGSM)

$$X_{Adversarial} = X + \varepsilon \, . \, sign\left(\nabla_X J(X, Y)\right),$$

$$where\ \varepsilon\ is\ small\ number\ and\ \nabla\ is\ the\ gradient\ of\ cost\ function\ with\ respect\ to\ X$$

In essence, FGSM is to add noise (not random noise) whose direction is the same as the gradient of the cost function with respect to the data. The noise is scaled by epsilon, which is usually constrained to be a small number via max norm.

RANDOM FEATURE NULLIFICATION

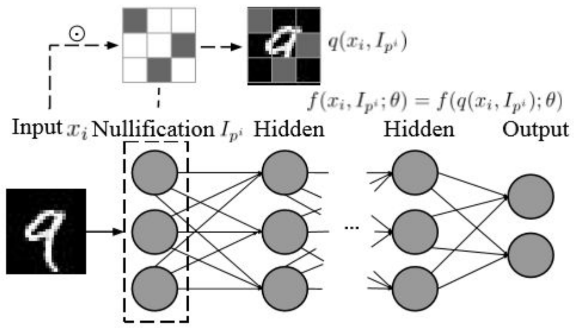


Figure 1 shows DNN equipped with Random Feature Nullification

Different from DNN, RFN introduces an additional layer between input layer and first hidden layer. This intermediate layer is stochastic, serving as a source of randomness during both training and testing phase. In particular, it randomly nullifies or masks the features within the input.

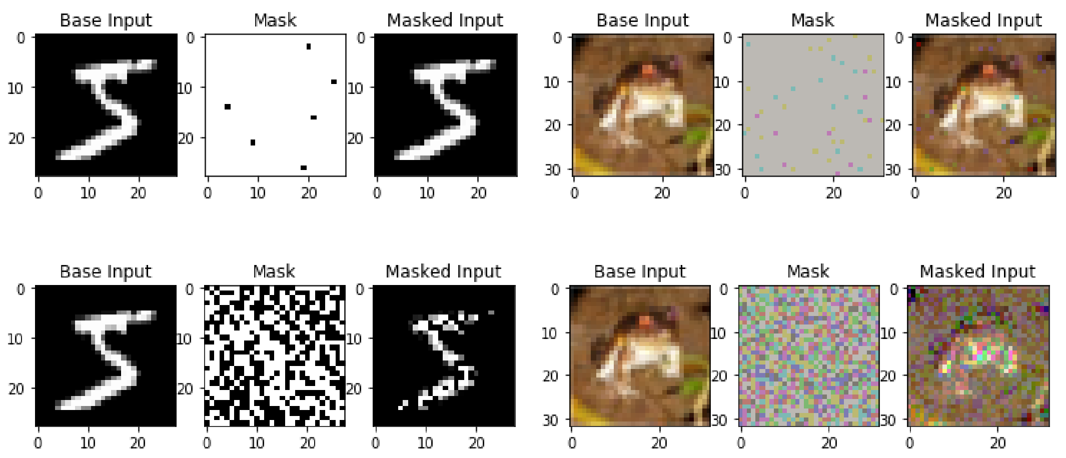


Figure 2. Examples of MNIST and CIFAR-10 images with 10% and 50% random feature nullification applied.

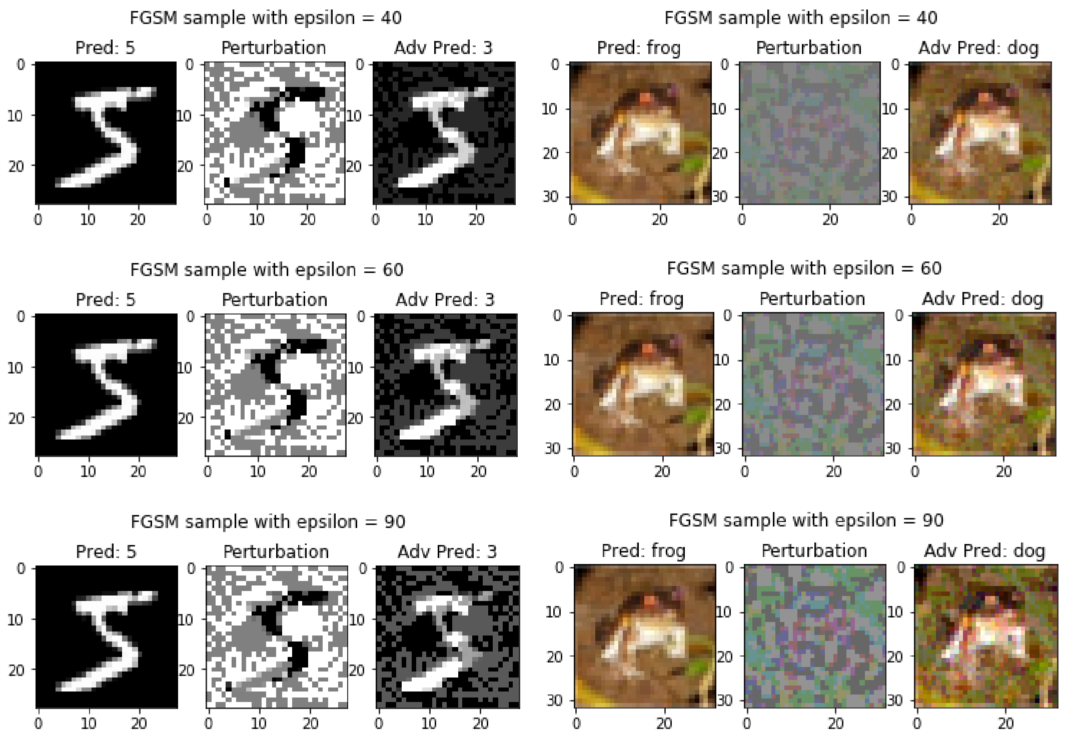


Figure 3. Examples of MNIST and CIFAR-10 images with a range of adversarial multipliers applied.

DATASET AND EXPERIMENTAL DESIGN

We were unable to obtain a malware dataset, so we focused on implementing the experimental design for the widely studied image recognition datasets: MNIST and

CIFAR-10. First, we built the DNN architectures described in the paper for a range of models: Standard, Dropout, Adversarial Training, RFN, Adversarial Training & RFN. MNIST models were all based on an identical base DNN architecture composed of four dense layers of 784 neurons. CIFAR-10 models were CNN based with a general architecture of two convolutions and a pool, followed by another 2 convolutions and a pool, followed by two dense layers. The number of filters and neurons varied slightly among these architectures, with no clear reasoning given in the paper for the variation. We tried running the same architecture across all models, but found that the accuracy of the Standard and Dropout models degraded at similar epochs to the other models using their architecture. This degradation is likely the reason behind the variation. As such, we used the same architectures described in the paper.

This range of models provides for comparison of several of the paper's hypotheses. The Standard model is entirely unregularized, so is expected to be very susceptible to adversarial attack. The Dropout model represents a common regularization method and is included for comparison to the proposed RFN method. The Adversarial Training model is included as another common approach to resist attack, as it's essentially a specialized data augmentation method. The RFN models are generated over a range of feature nullification rates from 10% to 50% with Figure 2 showing examples of images with nullified features. Finally, the Adversarial Training w/ RFN model is trained with adversarial images and with a set level random feature nullification.

Random feature nullification (RFN) was implemented via a wrapper class that extended pytorch nn.Module class. This class included a method to build a mask of zeros that was then multiplied element-wise by the training data to accomplish the feature nullification. The nullified training data was then sent to the models as normal for training. The number of nullified features was determined by a given mean and standard deviation used to draw from a normal distribution and then scaled up based on the size of the input.

The models were initially trained on train data and evaluated on validation data. These runs were used to verify the effectiveness of the hyperparameters presented in the paper. The only significant variation from the paper's hyperparameters resulting from these runs was the reduction in epochs from 50 to 20 for the CIFAR-10 models as the accuracies had leveled out at that point and the losses were starting to rise (see Appendix for epochs vs accuracy and loss). The model performance versus adversarial samples generated via the fast gradient sign method (FGSM) was also checked at this point.

Next the models were all trained on the entire training data set and then evaluated over a range of FGSM values for the multiplier of the gradient addition. Figure 3 shows examples of adversarial images. Evaluation was configured as usual for pytorch models with the addition of setting the RFN standard deviation parameter to zero and still including the RFN layer, effectively making the models non-deterministic. Altogether, 10 models were trained for each image data set with evaluation at three levels of FGSM.

RESULTS

As there is an infinite range of adversarial samples possible with FGSM, we initially experimented with levels of RFN over a single FGSM multiplier of 0.16 (or 40/255). Table 1 shows the results of these experiments. There is a clear correlation between increased levels of RFN and resistance to misclassification of adversarial samples. This resistance is accompanied by a decrease in model accuracy for non-adversarial samples. MNIST only loses ~1% of accuracy and gains ~48% resistance at 50% RFN, so the method is certainly worth implementing given that ratio performance degradation to gain in resistance. The conclusion is less clear for CIFAR-10, as it loses ~5% accuracy for a resistance gain of ~27%. These trends match those of the paper, but the magnitude of accuracy values that we obtained is consistently lower than those obtained in the paper, which is potentially concerning. The most likely explanation is due to their potential use of a different deep learning framework that had underlying assumptions that they didn't report in their summary of model architectures. Regardless, the resulting trends are promising enough that we continued with the next experiment using a mean of 0.5 to hopefully replicate the highest level of resistance.

	MNIST w/ RFN		CIFAR-10 w/ RFN	
		Resistance		Resistance
Expectation of nullification rates	Accuracy	0.16	Accuracy	0.16
0	0.9823	0.1949	0.7651	0.2654
0.1	0.981	0.3557	0.7274	0.3087
0.2	0.9805	0.428	0.7129	0.3745
0.3	0.978	0.5169	0.7148	0.4365
0.4	0.9785	0.5947	0.6931	0.4921
0.5	0.9731	0.6796	0.6803	0.534

Table 1. Accuracies of MNIST and CIFAR-10 with a range of random feature nullification values vs adversarial samples generated with a 0.16 multiple on the gradients.

In the next experiment we compared five different model architectures against three levels of FGSM adversarial attack and no attack. Table 2 shows the results of these experiments. As expected, the Standard models were strongly degraded by the attacks, with MNIST Standard dropping from 98% to 2%, which replicated the paper results. The CIFAR Standard exhibited the same overall lower accuracy magnitudes discussed before, but with better resistance than reported in paper. Both MNIST and CIFAR Dropout followed a similar pattern to the paper with the best no attack accuracies, but with poor resistance to FGSM attack. Therefore, our results confirm that regularization via dropout isn't sufficient to protect against this kind of attack. Adversarial Training (essentially specific augmentation) similarly didn't provide very effective resistance, again replicating the paper results.

Next, RFN showed the best resistance vs drop in accuracy for no attack, replicating the trends from the paper, but with much worse accuracy magnitudes, especially for MNIST RFN at attack level of 0.35 with our result of 25% vs the paper result of 61% being quite different given the similar no attack accuracy of 97%. Again, barring unreported assumptions, we replicated the architecture and methods, so have no clear explanation for this variation.

Finally, the Adversarial Training with RFN showed the first significant deviation in both trend and magnitude from the paper. The paper reported this combination as being the most effective with the highest resistance rates, but our MNIST was significantly worse than pure RFN and our CIFAR was slightly worse than our RFN.

	MNIST				CIFAR-10			
		Resistance (FGSM Rate)				Resistance (FGSM Rate)		
Learning Tech	Accuracy	0.16	0.24	0.35	Accuracy	0.16	0.24	0.35
Standard	0.9817	0.1678	0.0559	0.0227	0.4945	0.3107	0.3066	0.3041
Dropout	0.9821	0.1789	0.063	0.0208	0.7754	0.2741	0.2361	0.1978
Adv Training	0.9284	0.048	0.0408	0.0324	0.7	0.2451	0.2374	0.2221
RFN (mu=.5)	0.9731	0.6796	0.4348	0.2499	0.6803	0.534	0.4561	0.3809
Adv Training & RFN	0.9156	0.1978	0.0863	0.0606	0.6679	0.5418	0.4527	0.3607

Accuracies of a range of learning technologies for both MNIST and CIFAR vs adversarial samples generated over a range of gradient multipliers.

<u>CONCLUSIONS AND NEXT STEPS</u>

Overall we successfully replicated the trends from the paper showing the effectiveness of RFN at resisting misclassification due to FGSM adversarial attacks. We confirmed that current popular methods of regularization (dropout and data augmentation) were not effective at resisting attack and we also confirmed that RFN was very effective at resisting attack with an acceptable degradation of accuracy against non-adversarial samples.

The biggest variation in our findings from the paper was with the combination of RFN and Adversarial Training. The most likely reason for this variation was a misunderstanding of their methodology or an misimplementation of it, as there is no reason to expect the combination to perform so much worse given the improvements gained by the components separately.

For next steps, we want to review our Adversarial Training methodology to determine if there were issues and to reimplement it to triple check the paper's results for that combination.

We also want to continue efforts to extend the RFN method to other datasets and including text and pretrained image classification models such as IMDB dataset and Resnet18. We attempted to extend the method to these situations for this paper but were unable to fully develop the data pipelines to obtain meaningful results.

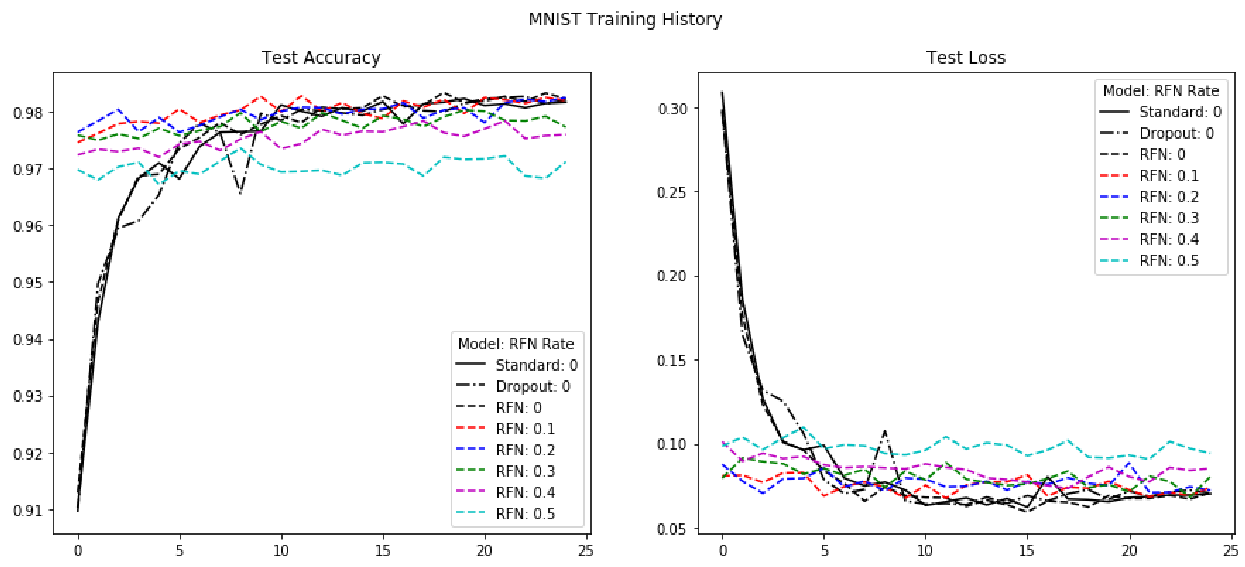
REFERENCES

[1] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G. Ororbia II , Xinyu Xing , C. Lee Giles , Xue Liu . 2017. Adversary Resistant Deep Neural Networks with an Application to Malware Detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, CA, Aug. 2017 (KDD'17)*, arXiv:1610.01239v4.

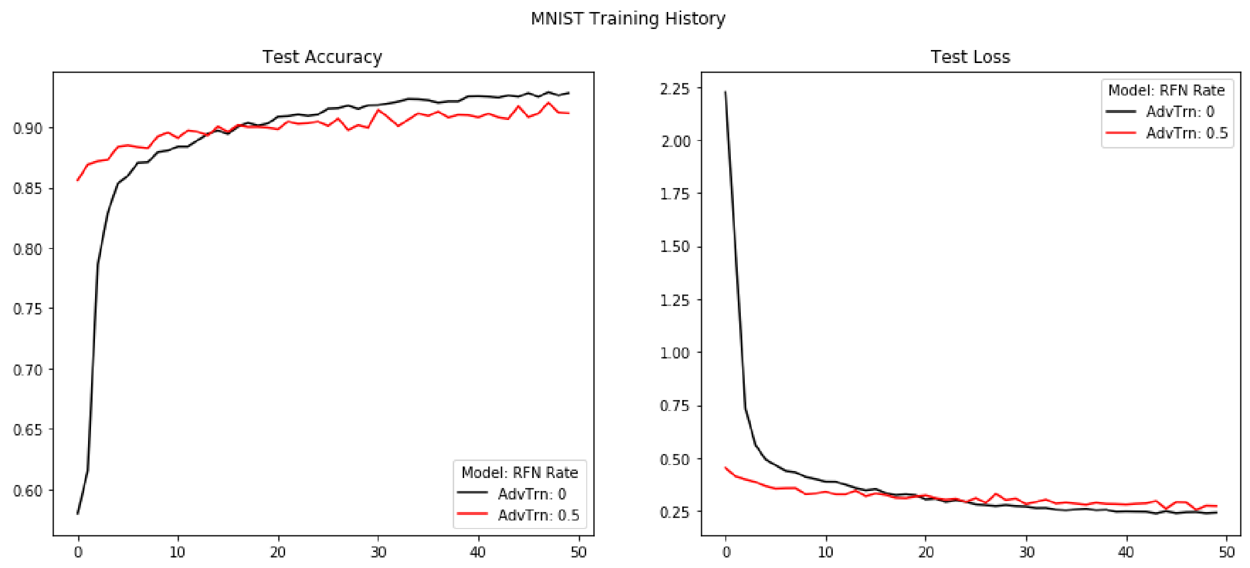
[2] Ian J.Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv:1412.6572.

APPENDIX

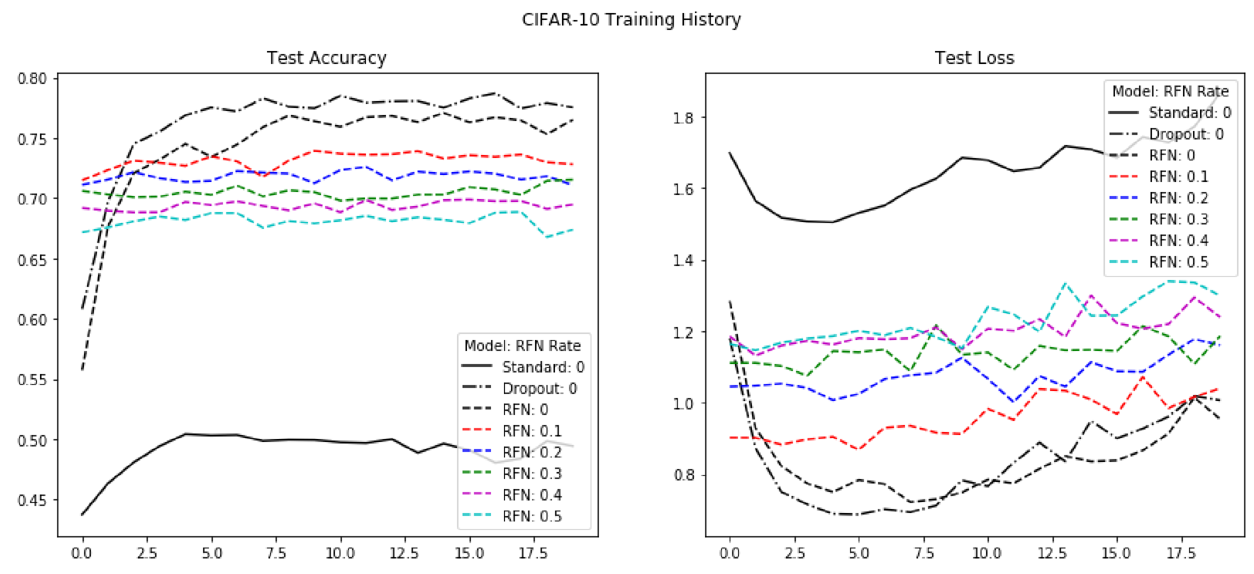
Training MNIST models Standard, Dropout, and RFN over range of epsilons:



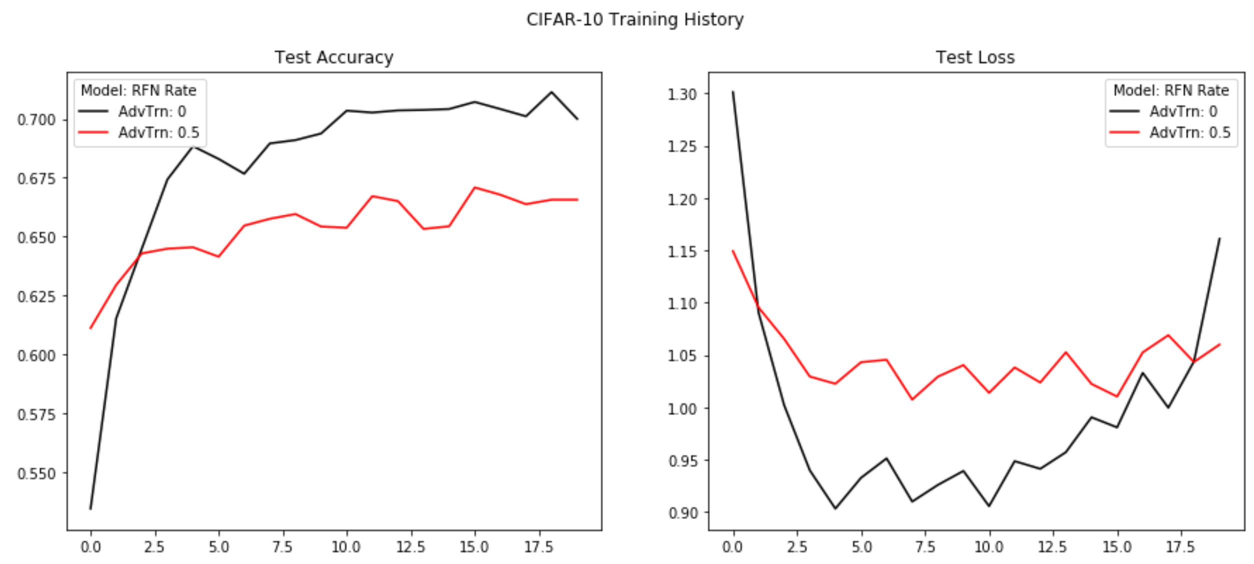
Training MNIST models Adversary Trained and Adversary Trained w/ RFN (mu=0.5) over range of epsilons:



Training CIFAR-10 models Standard, Dropout, and RFN over range of epsilons:



Training CIFAR-10 models Adversary Trained and Adversary Trained w/ RFN (mu=0.5) over range of epsilons


CIFAR-10 Training History
Test Accuracy
Test Loss
Model: RFN Rate
AdvTrn: 0
AdvTrn: 0.5