Adversarial Machine Learning & Effectiveness of Attacks on Various Classifiers

Code and Video: https://github.com/mschroeder10/ML-final-project

TA: Xuyang Li

Team Members: Megan Schroeder and Ramsey Rosaa

Problem Description

Image classification is used in many applications, such as medical imaging, driverless cars, machine vision, and biometric identification. Not surprisingly it is also a target for adversarial attacks. An adversary may try to fool a medical imaging system in an attempt to alter a diagnosis. A driverless car could be tricked into slamming on the brakes by a seemingly innocuous image on a sticker on another vehicle. Systems using biometric identification, such as iris recognition, facial recognition, or fingerprint scans could be vulnerable to attack, creating potentially large security risks. Classifiers have become very powerful especially with the rise of neural networks.

However, they are not perfect, and can be fooled into false classifications. An example of this is that malicious algorithms will attempt to manipulate individual pixels to try and increase the score for a desired class. The pixels in the image are modified in a way to deceive a classification model while still appearing "normal" to a human. In many cases a human cannot distinguish between the original image and the attacked image. This is done by updating the image, as opposed to the parameter during gradient descent. This is just one example of an attack. We want to attempt to exploit various classifiers using attacks from the Cleverhans/Adversarial Robustness Toolbox (ART) on both linear and nonlinear classifiers to observe which ones are more robust against possible attacks.

Dataset & Analysis

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. This dataset is optimized for classification problems and will not require as much wrangling, feature selection, or other generalization techniques as other datasets might. Each image is 32 x 32 pixels and each pixel has RGB values between 0 and 225. These values are the features. The labels are integers between 0 and 9 that identify which of ten images each example is. There is an equal number (5000) of each of the ten classes in the training data and an equal number (1000) of each of the ten classes in the testing data.

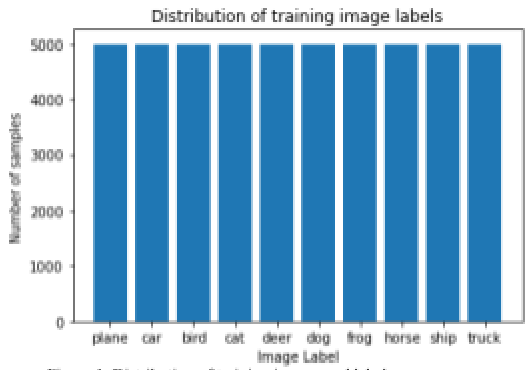


Figure 1. Distribution of training images and labels

Data Preprocessing
The original data requires a small amount of preprocessing. To use the sklearn models, we need to flatten each image from 32 x 32 x 32 to a vector. For the deep learning models, we can use the original format. For the label (ie, 'y') data, for the sklearn models we need this to be a 1D array of the numeric values identifying the class. For the ART modules, the y values need to be one-hot encoded. Therefore we create both types of X and y data.

Feature Selection
With each image flattened from 32 x 32 x 3 to a vector of length 3072, we effectively have 3072 features. Combining this with 50,000 training samples makes the models large and in some cases very slow. For the sklearn models, we used Principal Component Analysis (PCA) to reduce the number of features. We ran PCA with varying numbers of components, tested the results on a logistic regression model, and plotted how the accuracy varied with the PCA model. We determined that we could use approximately 100 components without noticeably reducing the accuracy of the model.

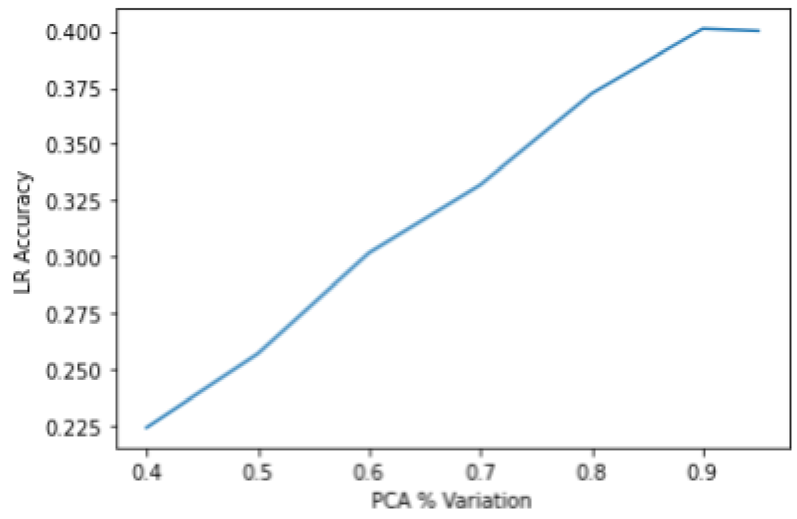


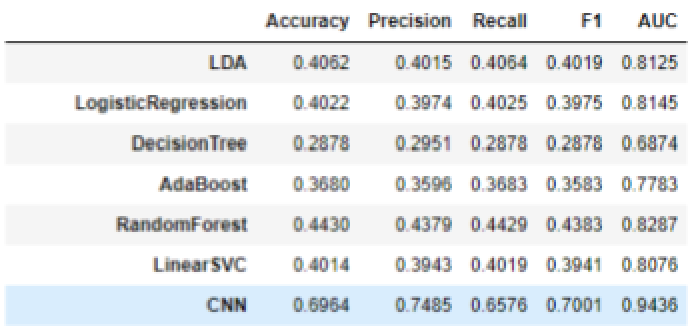
Figure 2. PCA component variation to model accuracy

For the deep learning model, we did not use feature reduction since that's somewhat built into the model.

Since our goal was to see how resilient different classification models are to adversarial images, we evaluated the baseline performance of several models against the CIFAR-10 dataset. We used GridSearchCV to select model parameters. We also evaluated a Convolutional Neural Network model using an example model from TensorFlow. (https://www.tensorflow.org/tutorials/images/cnn).

Approach & Methodology

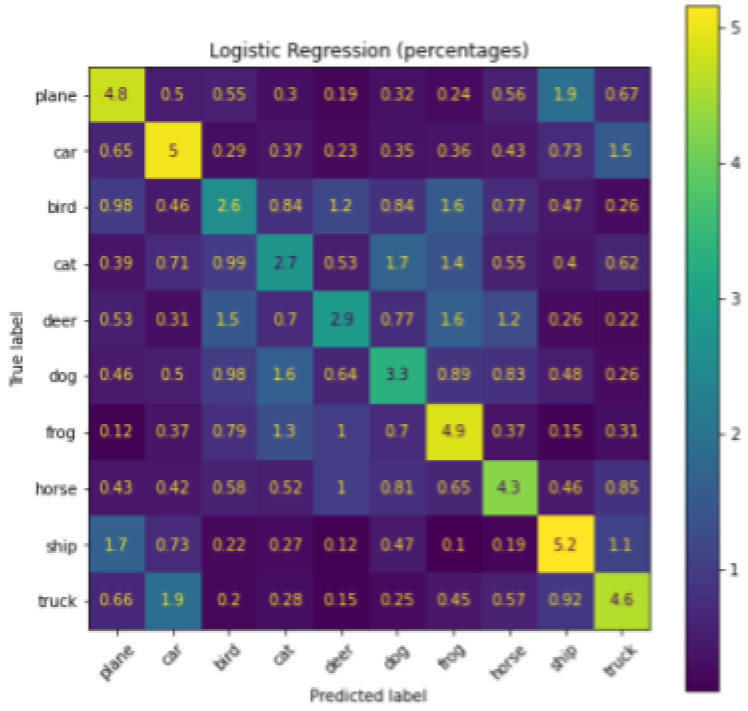
Machine Learning Models
We trained 7 models shown above on the normal benign data to get a baseline of how well each classifier performed. We then generated a few different attacks, ran the trained classifiers on the malicious images, and observed the results.



	Accuracy	Precision	Recall	F1	AUC
LDA	0.4062	0.4015	0.4064	0.4019	0.8125
LogisticRegression	0.4022	0.3974	0.4025	0.3975	0.8145
DecisionTree	0.2878	0.2951	0.2878	0.2878	0.6874
AdaBoost	0.3680	0.3596	0.3683	0.3583	0.7783
RandomForest	0.4430	0.4379	0.4429	0.4383	0.8287
LinearSVC	0.4014	0.3943	0.4019	0.3941	0.8076
CNN	0.6964	0.7485	0.6576	0.7001	0.9436

Figure 3. Initial metrics of all base models on benign data

We also plotted a confusion matrix for each classifier to get an idea of which classes had better or worse performance. This is an example for logistic regression.



	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
plane	4.8	0.5	0.55	0.3	0.19	0.32	0.24	0.56	1.9	0.67
car	0.65	5	0.29	0.37	0.23	0.35	0.36	0.43	0.73	1.5
bird	0.98	0.46	2.6	0.84	1.2	0.84	1.6	0.77	0.47	0.26
cat	0.39	0.71	0.99	2.7	0.53	1.7	1.4	0.55	0.4	0.62
deer	0.53	0.31	1.5	0.7	2.9	0.77	1.6	1.2	0.26	0.22
dog	0.46	0.5	0.98	1.6	0.64	3.3	0.89	0.83	0.48	0.26
frog	0.12	0.37	0.79	1.3	1	0.7	4.9	0.37	0.15	0.31
horse	0.43	0.42	0.58	0.52	1	0.81	0.65	4.3	0.46	0.85
ship	1.7	0.73	0.22	0.27	0.12	0.47	0.1	0.19	5.2	1.1
truck	0.66	1.9	0.2	0.28	0.15	0.25	0.45	0.57	0.92	4.6

Figure 4. Confusion matrix for predicted vs. true label for logistic regression classifier

We then plotted a multiclass ROC curve using a OneVsRestClassifier. This is an example using a logistic regression model.

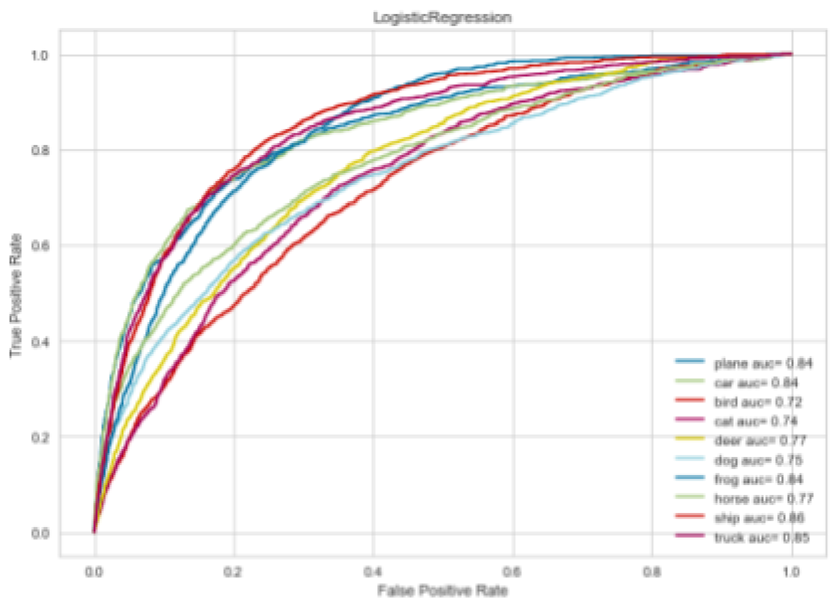


Figure 5. Multiclass ROC curve for base logistic regression classifier

There is some similarity to the confusion matrix. The five lower curves correspond to the five lowest diagonal entries in the confusion matrix. We noticed that the classifiers seemed to perform better on the vehicle-type classes, such as trucks and cars, whereas they seemed to perform worse on the animal-type categories.

Adversarial Attacks

We then tried adversarial attacks on the testing data using the Adversarial Robustness Toolbox (ART) toolbox. The ART library has many different evasion attacks. However most of these attacks are specific to a particular type of classifier. We initially tried classifier-specific attacks, such as DeepFool for neural networks or Fast Gradient Descent for Logistic Regression. However, we realized that these attacks were too specific, and did not work very well on other classifiers. We then decided to try a black-box evasion attack and a poisoning attack, because those were not classifier specific.

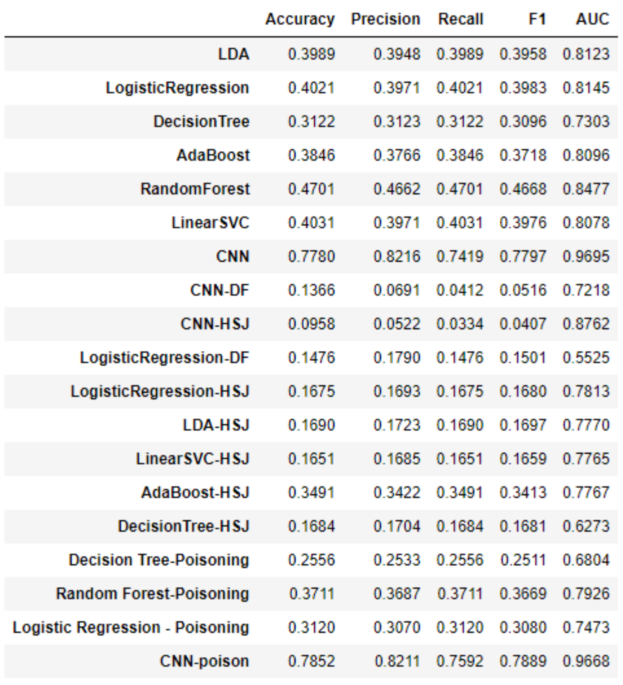
The first attack we tried was the DeepFool attack against neural networks. DeepFool sequentially adds perturbations into images to reduce the euclidean distance to the next closest hyperplane to trick the classifier into misclassifying an image. (https://towardsdatascience.com/deepfool-a-simple-and-accurate-method-to-fool-deep-neural-networks-17e0d0910ac0).

The black box evasion attack we chose was the HopSkipJump attack. We ran this attack on AdaBoost, Logistic Regression, Decision Tree, CNN, LDA, and Linear SVC. HopSkipJump is an attack that adds patches of perturbation, or noise, to images or videos. These adversarial patches cause classifiers to ignore other features in an image and misclassify it(https://arxiv.org/abs/1712.09665).

On top of the two evasion attacks, we also tried 2 backdoor poisoning attacks. The first simply changes pixel values in the corner of an image to a value of 1. This perturbation “poisons” the data our models rely on to train causing them to misidentify objects (https://arxiv.org/abs/1708.06733). We ran the poisoning backdoor on a Decision Tree, RandomForest, Logistic Regression, and CNN. The second attack was a clean label backdoor attack, and uses an already existing poisoning attack and a proxy classifier to determine what pixels may be changed without being clearly incorrect thus avoiding human detection (https://people.csail.mit.edu/madry/lab/cleanlabel.pdf). This second attack unfortunately proved to be much too memory intensive for our machines and google colab ran out of RAM almost immediately after attempting to run this attack.

Results

Overall, the various attacks appeared to be very effective. We initially found that the classifier-specific attacks worked much better when used on the classifier they were generated off of compared to when they were passed into different classifiers. This can be seen with the DeepFool attack in figure 6a. We also discovered that deep learning was by far the most effective method for image classification. All the non CNN classifiers had an initial accuracy between .3 and .4 with parameter optimization whereas the CNN had an initial accuracy of .778. Of the non-CNN classifiers, random forest had the best performance with an initial accuracy of .4701. The decision tree was by far the worst, with an accuracy of .3122.



	Accuracy	Precision	Recall	F1	AUC
LDA	0.3989	0.3948	0.3989	0.3958	0.8123
LogisticRegression	0.4021	0.3971	0.4021	0.3983	0.8145
DecisionTree	0.3122	0.3123	0.3122	0.3096	0.7303
AdaBoost	0.3846	0.3766	0.3846	0.3718	0.8096
RandomForest	0.4701	0.4662	0.4701	0.4668	0.8477
LinearSVC	0.4031	0.3971	0.4031	0.3976	0.8078
CNN	0.7780	0.8216	0.7419	0.7797	0.9695
CNN-DF	0.1366	0.0691	0.0412	0.0516	0.7218
CNN-HSJ	0.0958	0.0522	0.0334	0.0407	0.8762
LogisticRegression-DF	0.1476	0.1790	0.1476	0.1501	0.5525
LogisticRegression-HSJ	0.1675	0.1693	0.1675	0.1680	0.7813
LDA-HSJ	0.1690	0.1723	0.1690	0.1697	0.7770
LinearSVC-HSJ	0.1651	0.1685	0.1651	0.1659	0.7765
AdaBoost-HSJ	0.3491	0.3422	0.3491	0.3413	0.7767
DecisionTree-HSJ	0.1684	0.1704	0.1684	0.1681	0.6273
Decision Tree-Poisoning	0.2556	0.2533	0.2556	0.2511	0.6804
Random Forest-Poisoning	0.3711	0.3687	0.3711	0.3669	0.7926
Logistic Regression - Poisoning	0.3120	0.3070	0.3120	0.3080	0.7473
CNN-poison	0.7852	0.8211	0.7592	0.7889	0.9668

Figure 6a. Metrics for all base and attacked classifiers

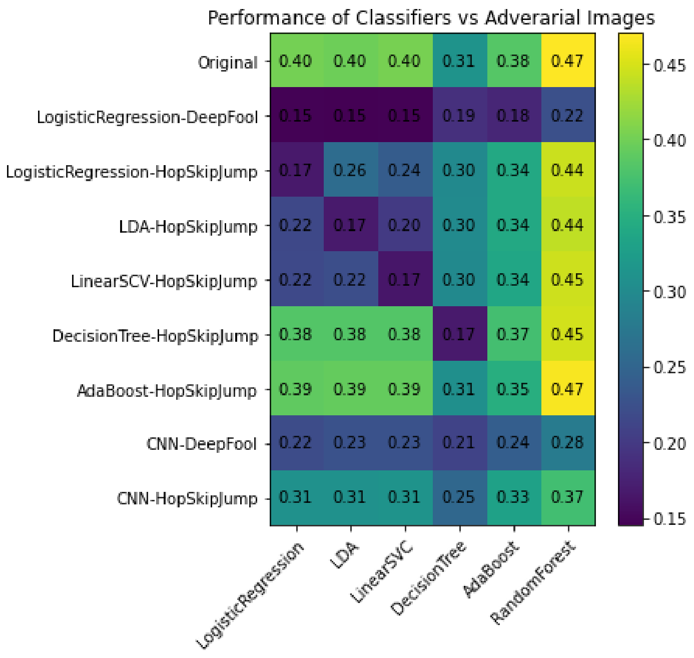


Figure 6b. Heatmap of performance of classifiers vs. adversarial images

Evasion Attacks

Our DeepFool attack was the first adversarial model we experimented with before realizing it was too classifier specific. DeepFool is an attack against neural networks, so the accuracy when run on the CNN was .1386, compared to when it ran on the logistic regression classifier with an accuracy of .1476. One interesting observation that can be seen in figure 8a is that a few of the DeepFool generated images were completely distorted.

HopSkipJump is not a classifier dependent attack, and works on a variety of linear, non linear, and deep learning classifiers. HSJ was the most effective on the CNN, dropping the accuracy to .0958. Logistic regression, LDA, linear SVC, and the decision tree all performed similarly, with accuracies around .16 to .17. These were still significant drops from the initial accuracy on benign data. AdaBoost seemed to be the most resilient to the HSJ images. The accuracy only dropped slightly, going from .3846 to .3491. The HopSkipJump generated images can be seen in figure 8b.

We generated a confusion matrix and multi-class ROC curve for each adversarial model to get a better sense of which labels were being misclassified the most. Figure 7a and 7b show an example confusion matrix and ROC curve for the HopSkipJump attack on the CNN.

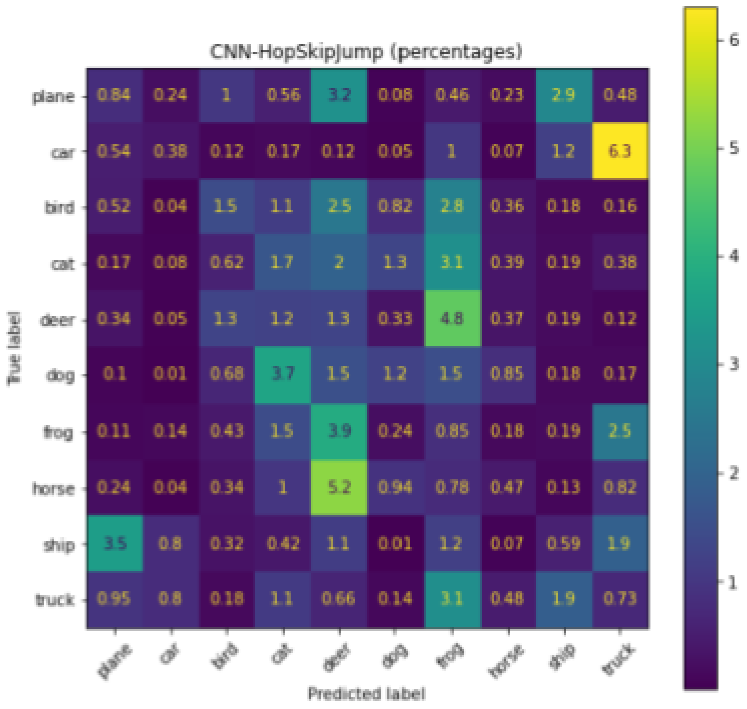


Figure 7a. Confusion matrix for CNN on HopSkipJump data

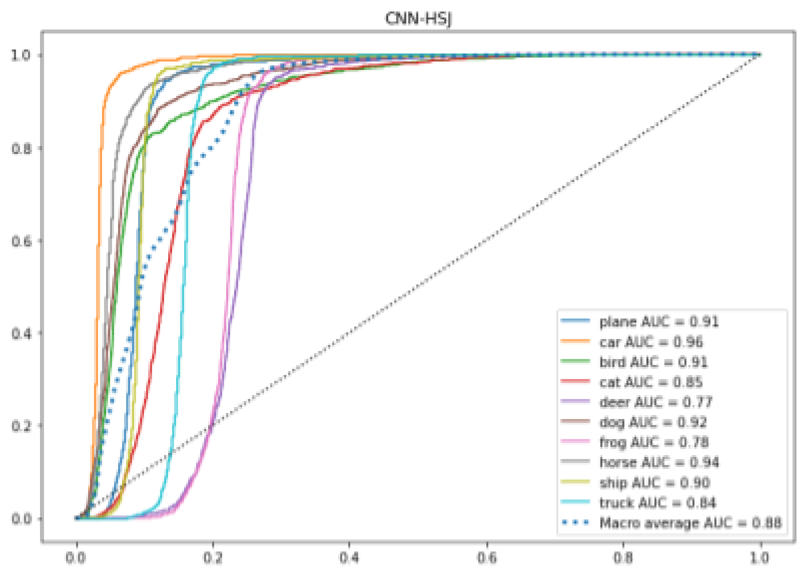


Figure 7b. Multiclass ROC for CNN on HopSkipJump data

Initially, the AUCs for the CNN on benign data were .98 or .99 for all classes (table 1, appendix). After evaluating the HSJ data, the AUC for all classes drops. The CNN still appeared to perform slightly better on vehicle-type classes compared to animal-type classes, with the exception of trucks, which had a much lower AUC of .84. The classifier misclassified cars as trucks, horses as deer, and planes as ships at a significantly higher rate.

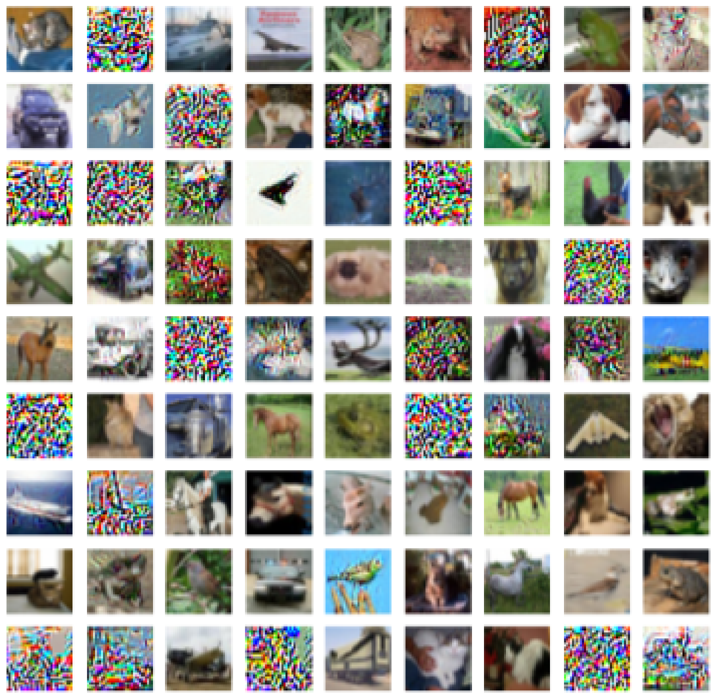


Figure 8a. DeepFool generated images

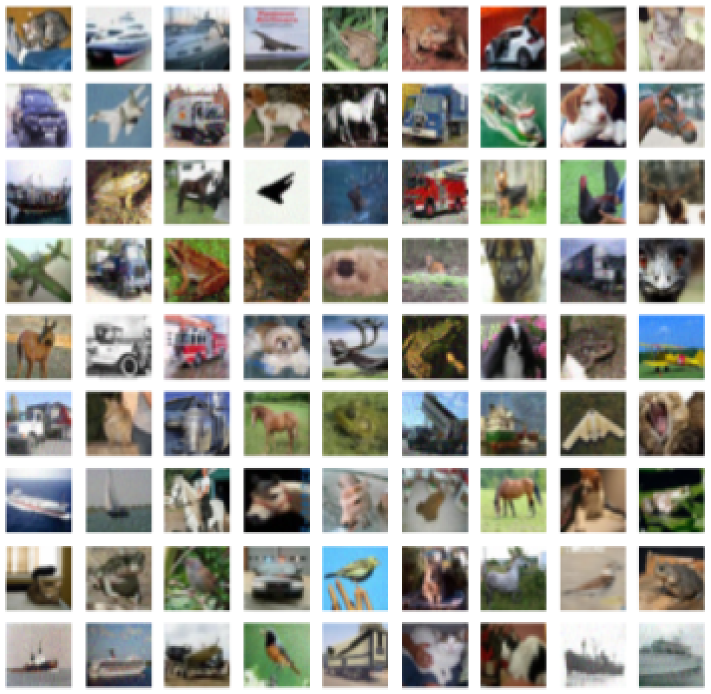


Figure 8b. Example HSJ images

Poisoning Attacks

Our backdoor poisoning attack was less effective than our evasion attacks, only reducing the accuracy of the Decision Tree, Random Forest, and Linear Regression by about 5-10%, and not affecting the CNN at all, however its changes to the images are much more subtle and less likely to be detected by a human monitoring the classifiers. Had we gotten the clean label to run with our limited machine specs the changes would be near undetectable. The poisoning attacks are less effective at attacking the classifiers but much less likely to be discovered, and as such are more likely to be successful.

We again generated a confusion matrix and multiclass ROC curve for each model using the poisoned data to get a better sense of which labels were being misclassified the most. Figures 9a and 9b show an example confusion matrix and ROC curve for the Decision Tree.

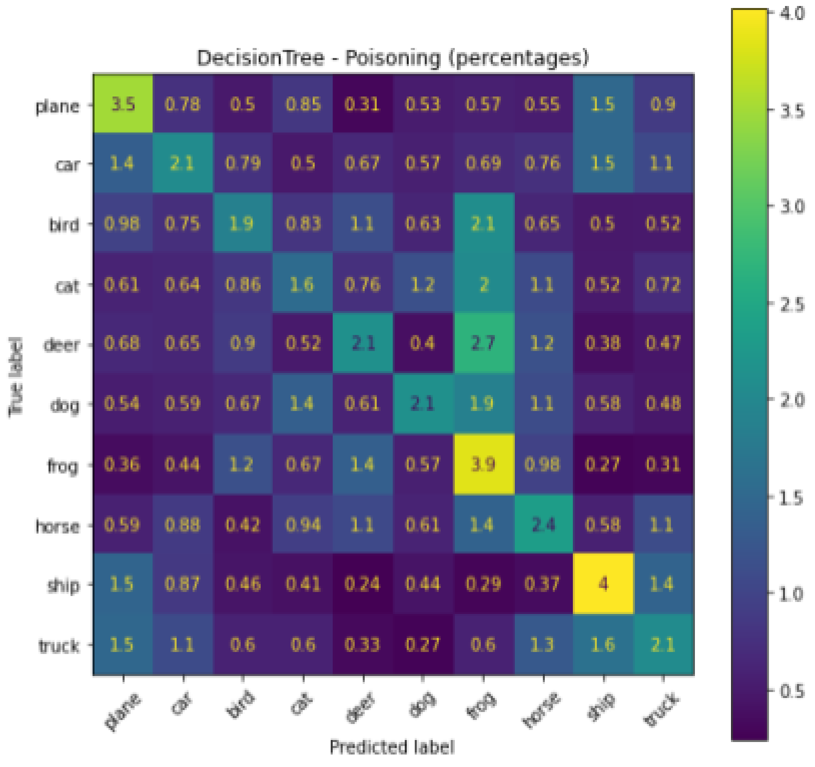


Figure 9a. Confusion Matrix for Decision Tree on poisoned data

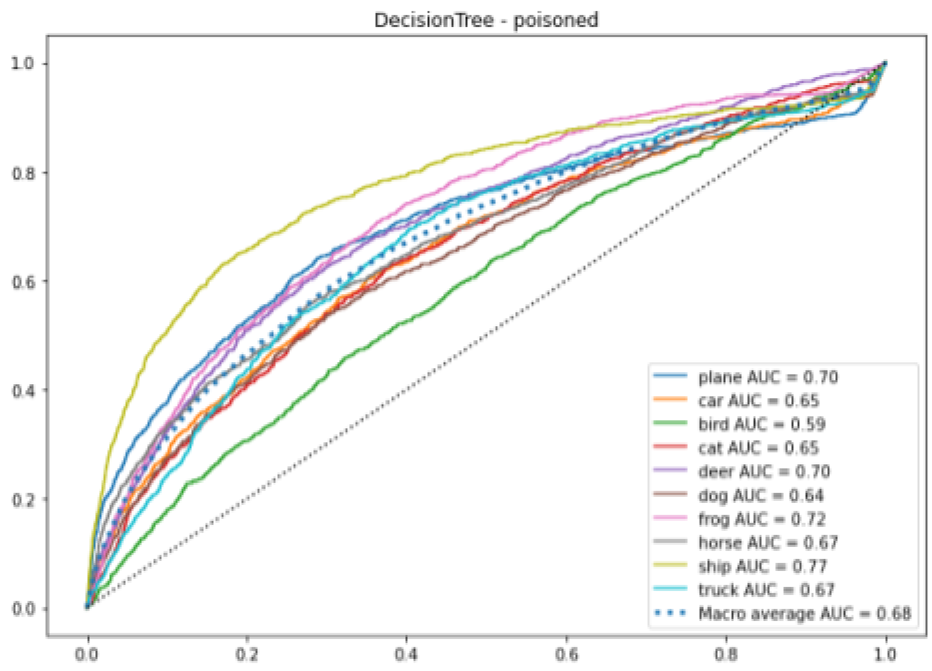


Figure 9b. Multiclass ROC for Decision Tree on poisoned data

Shown below in figure 10 are the images poisoned by our backdoor attack. If you look closely the images all have a small triangle of white pixels in the bottom right of the image. These easily missed white pixels are what skew the classification and lower the accuracy of models trained on this data.

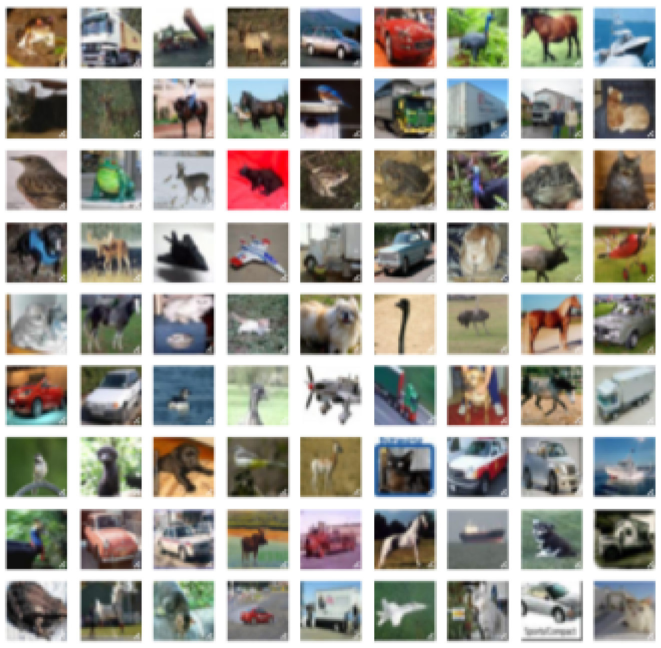


Figure 10. Images poisoned by the backdoor attack

Conclusion

Overall, adversarial machine learning seems to be an effective tool to attack classifiers and trick them into misclassifying data. Of the 3 different attacks that were tried, all of them had a substantial impact on accuracy for all classifiers. None of the classifiers with the exception of AdaBoost seemed to be particularly robust against the evasion attacks. However, AdaBoost performs so poorly in the first place, so it is hard to say how robust the classifier actually is. The CNN was the only classifier able to completely resist the poisoning attacks, however the effect on the other classifiers wasn't very drastic either.

In the future, we would hope to find ways to better generalize these attacks in order to get a more robust understanding of their effectiveness. There were models we weren't able to try with DeepFool or the poisoning attacks and as such we can't come to any firm conclusions on which model is the most resistant to attacks. However, as we've noted in this paper, many attacks are tailored towards certain types of classifiers. There doesn't exist a one size fits all attack and so of course there doesn't exist a classifier that resists all attacks best either. Like every other aspect of machine learning, it is our job to decide what classifier works best in which situation, or which attack.

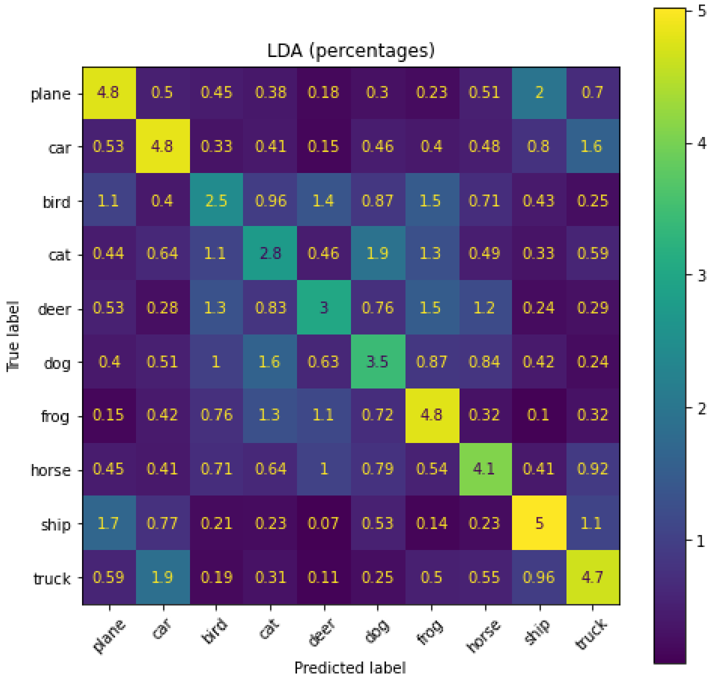
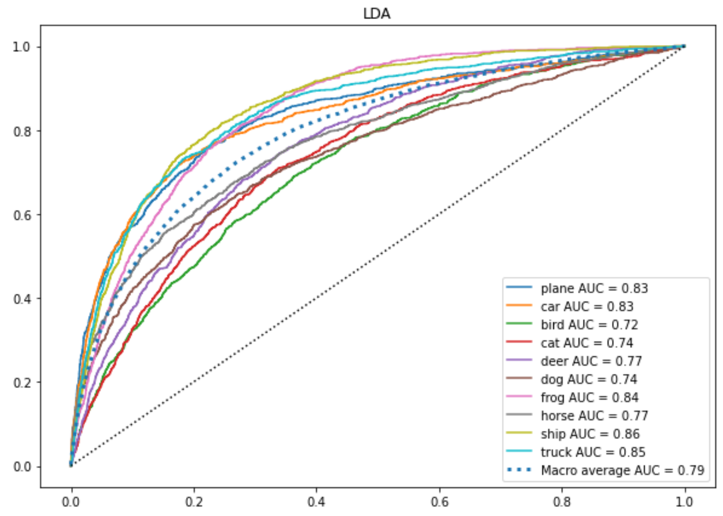
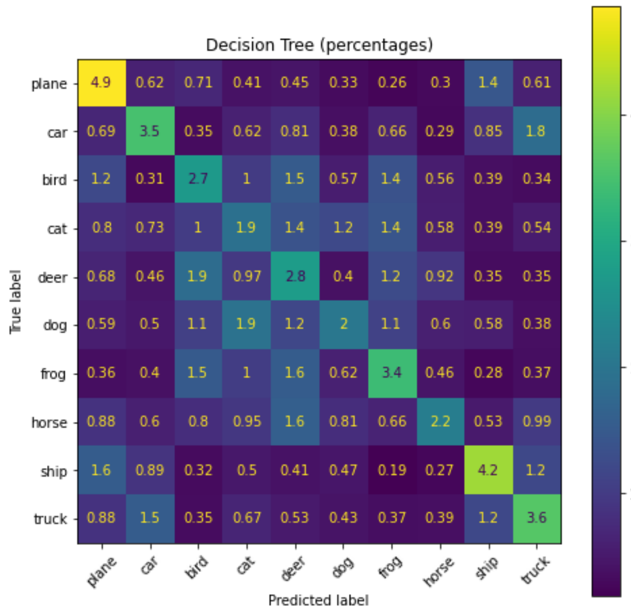
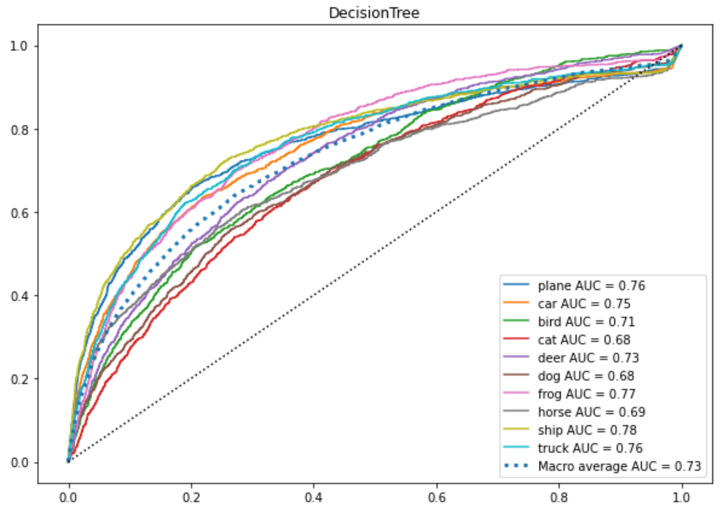
Contribution

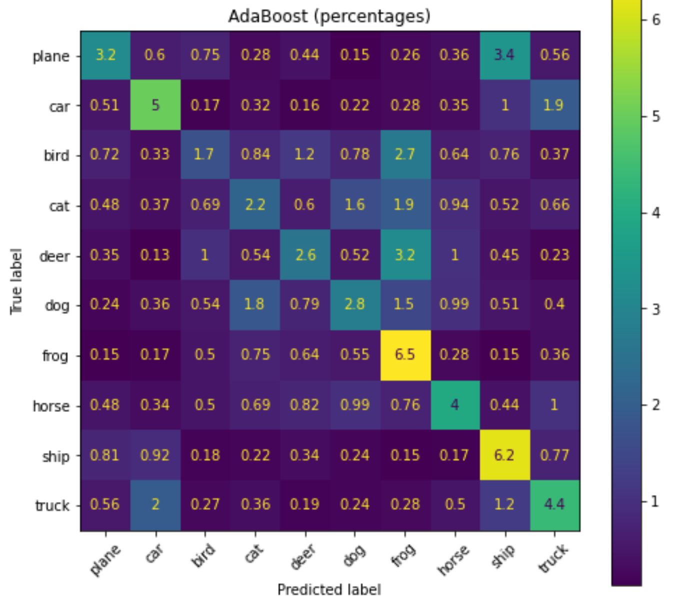
Megan: Preprocessing and evasion attacks

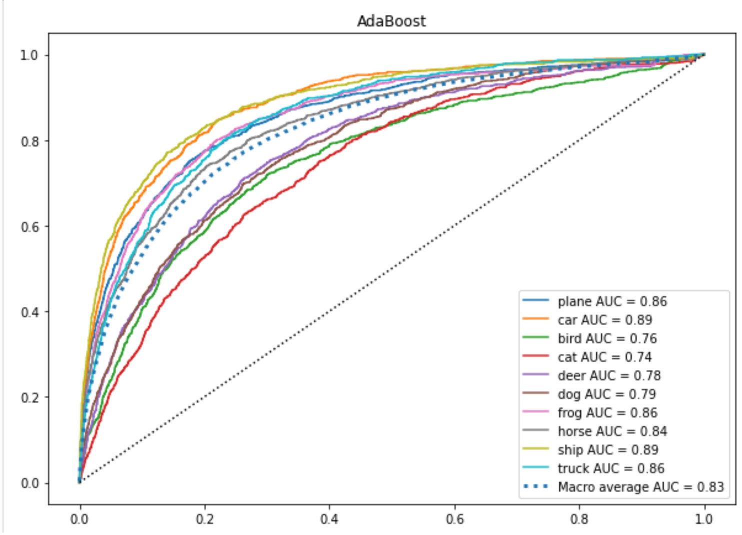
Ramsey: Poisoning attacks

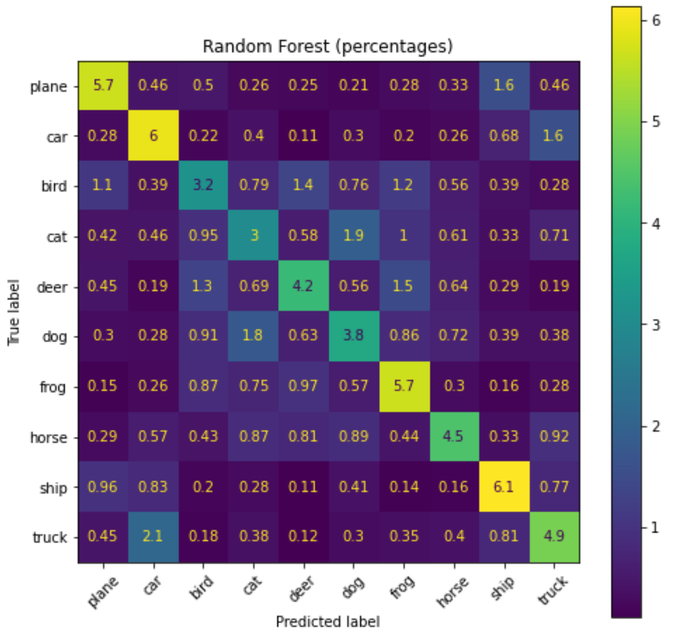
Appendix

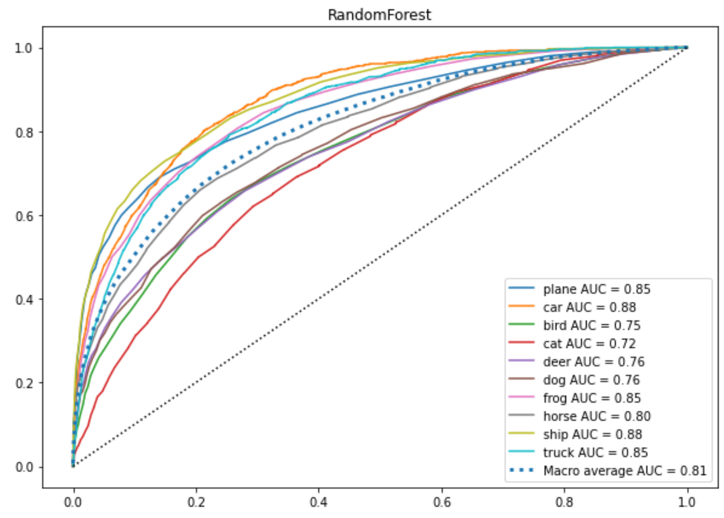
Figures can also be viewed in notebook.
Table 1. Classifiers on normal data

Confusion matrix	Multiclass ROC
	
	


AdaBoost (percentages)
True label
Predicted label
plane car bird cat deer dog frog horse ship truck
plane 3.2 0.6 0.75 0.28 0.44 0.15 0.26 0.36 3.4 0.56
car 0.51 5 0.17 0.32 0.16 0.22 0.28 0.35 1 1.9
bird 0.72 0.33 1.7 0.84 1.2 0.78 2.7 0.64 0.76 0.37
cat 0.48 0.37 0.69 2.2 0.6 1.6 1.9 0.94 0.52 0.66
deer 0.35 0.13 1 0.54 2.6 0.52 3.2 1 0.45 0.23
dog 0.24 0.36 0.54 1.8 0.79 2.8 1.5 0.99 0.51 0.4
frog 0.15 0.17 0.5 0.75 0.64 0.55 6.5 0.28 0.15 0.36
horse 0.48 0.34 0.5 0.69 0.82 0.99 0.76 4 0.44 1
ship 0.81 0.92 0.18 0.22 0.34 0.24 0.15 0.17 6.2 0.77
truck 0.56 2 0.27 0.36 0.19 0.24 0.28 0.5 1.2 4.4
1 2 3 4 5 6


AdaBoost
plane AUC = 0.86
car AUC = 0.89
bird AUC = 0.76
cat AUC = 0.74
deer AUC = 0.78
dog AUC = 0.79
frog AUC = 0.86
horse AUC = 0.84
ship AUC = 0.89
truck AUC = 0.86
Macro average AUC = 0.83
0.0 0.2 0.4 0.6 0.8 1.0


Random Forest (percentages)
True label
Predicted label
plane car bird cat deer dog frog horse ship truck
plane 5.7 0.46 0.5 0.26 0.25 0.21 0.28 0.33 1.6 0.46
car 0.28 6 0.22 0.4 0.11 0.3 0.2 0.26 0.68 1.6
bird 1.1 0.39 3.2 0.79 1.4 0.76 1.2 0.56 0.39 0.28
cat 0.42 0.46 0.95 3 0.58 1.9 1 0.61 0.33 0.71
deer 0.45 0.19 1.3 0.69 4.2 0.56 1.5 0.64 0.29 0.19
dog 0.3 0.28 0.91 1.8 0.63 3.8 0.86 0.72 0.39 0.38
frog 0.15 0.26 0.87 0.75 0.97 0.57 5.7 0.3 0.16 0.28
horse 0.29 0.57 0.43 0.87 0.81 0.89 0.44 4.5 0.33 0.92
ship 0.96 0.83 0.2 0.28 0.11 0.41 0.14 0.16 6.1 0.77
truck 0.45 2.1 0.18 0.38 0.12 0.3 0.35 0.4 0.81 4.9
1 2 3 4 5 6


RandomForest
plane AUC = 0.85
car AUC = 0.88
bird AUC = 0.75
cat AUC = 0.72
deer AUC = 0.76
dog AUC = 0.76
frog AUC = 0.85
horse AUC = 0.80
ship AUC = 0.88
truck AUC = 0.85
Macro average AUC = 0.81
0.0 0.2 0.4 0.6 0.8 1.0

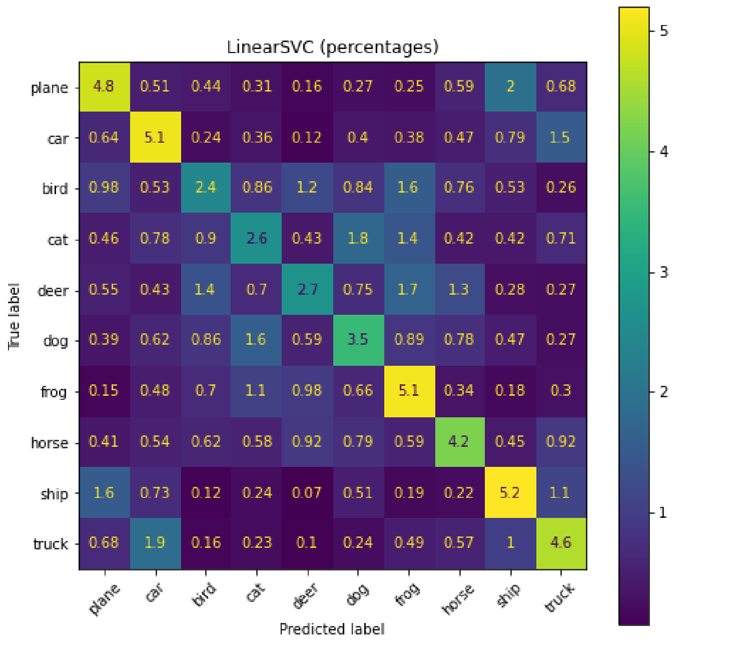
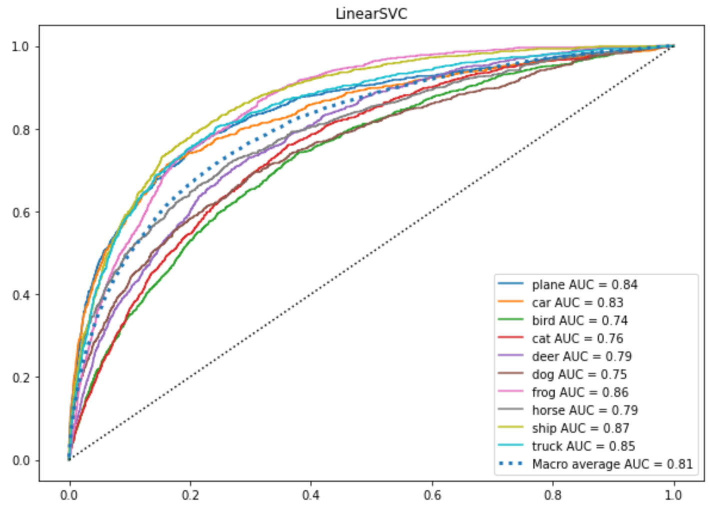
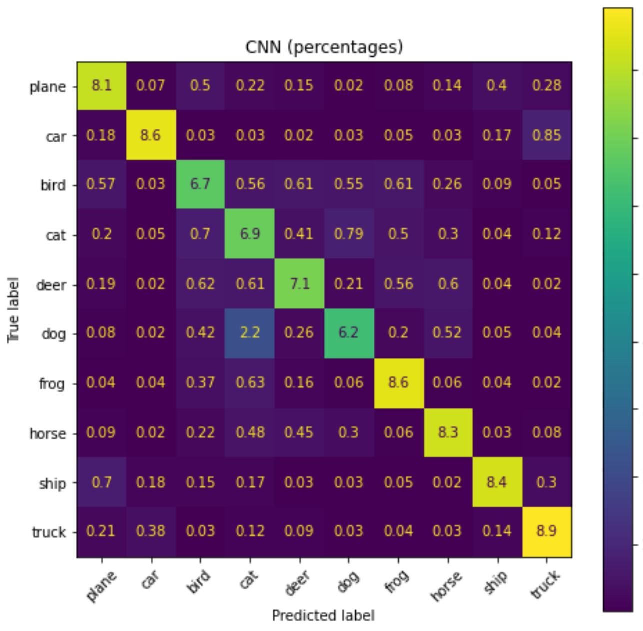
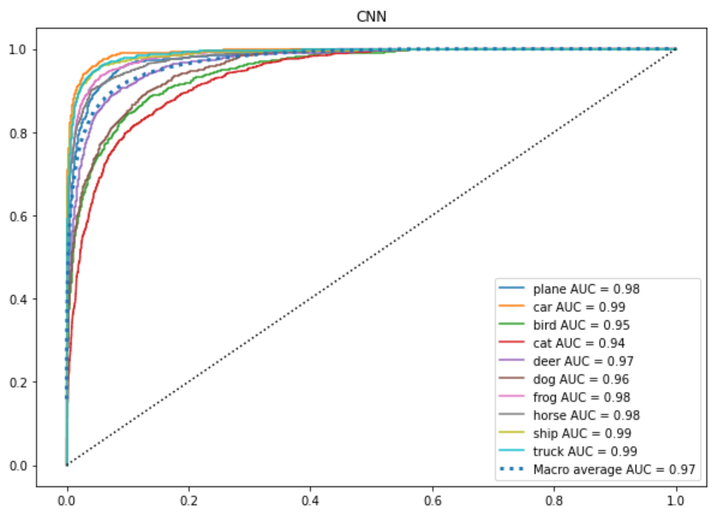
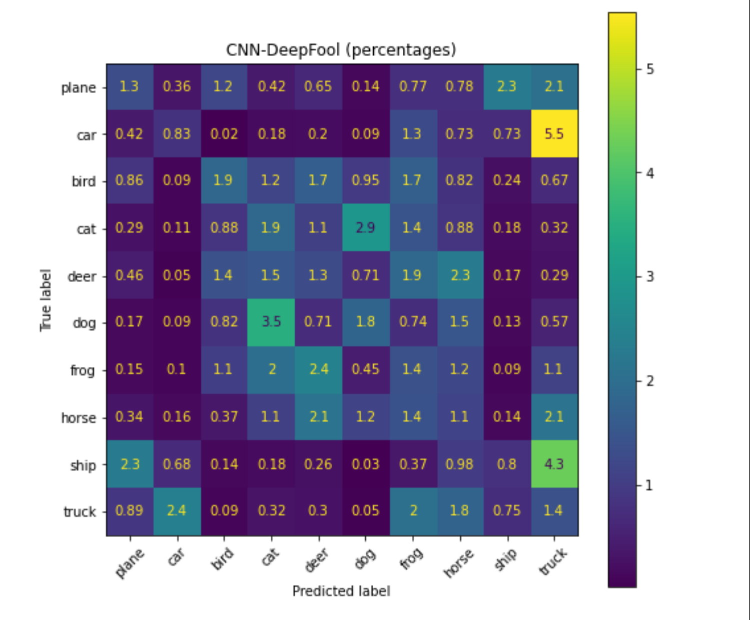
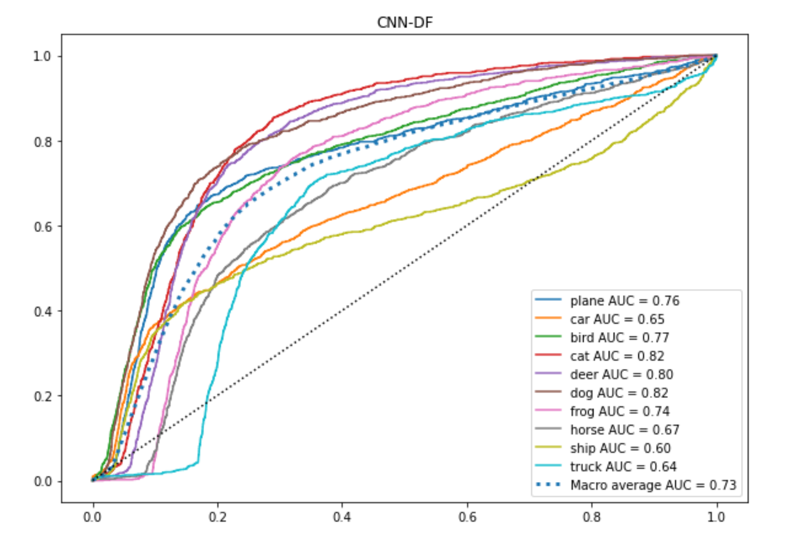
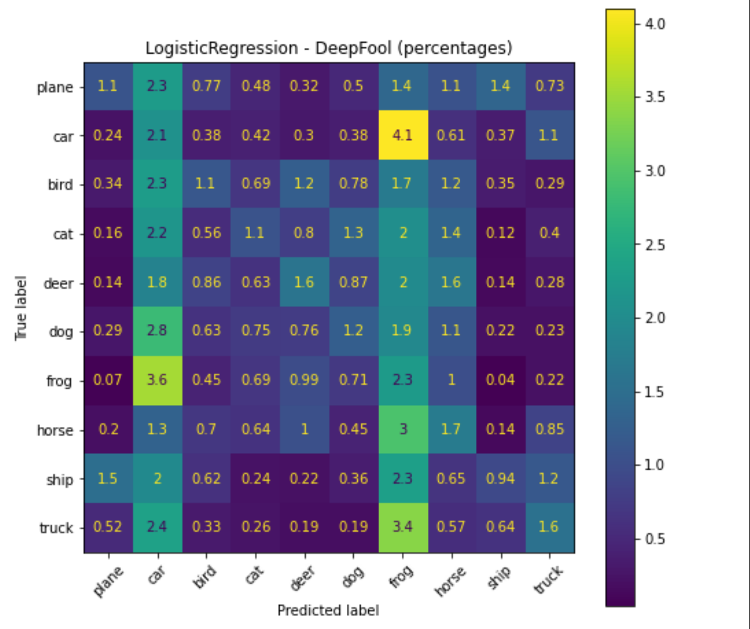
	
	

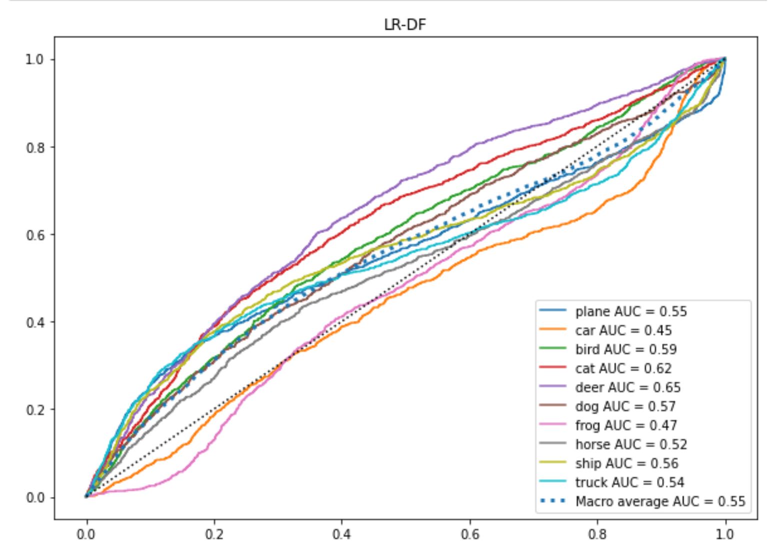
Table 2. Classifiers on adversarial data

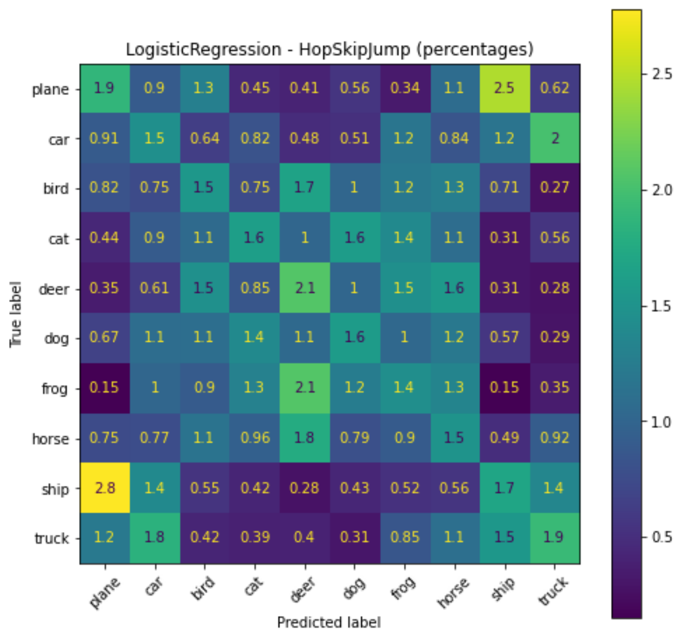
Confusion Matrix	Multiclass ROC

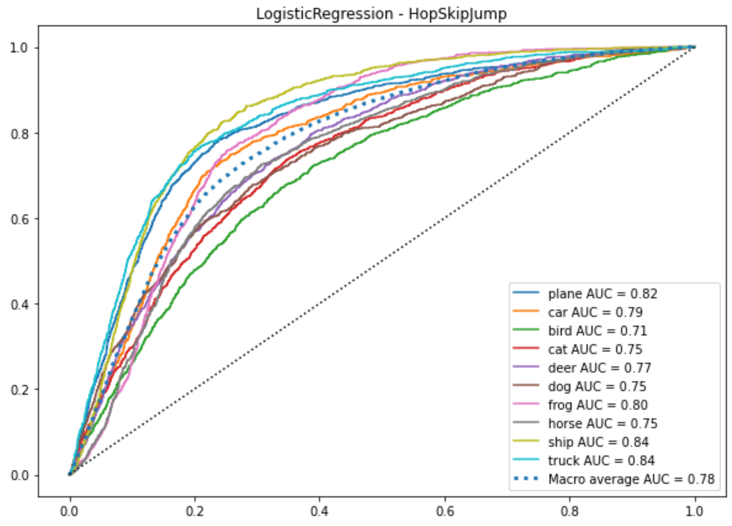

CNN-DeepFool (percentages)
True label
Predicted label
plane 1.3 0.36 1.2 0.42 0.65 0.14 0.77 0.78 2.3 2.1
car 0.42 0.83 0.02 0.18 0.2 0.09 1.3 0.73 0.73 5.5
bird 0.86 0.09 1.9 1.2 1.7 0.95 1.7 0.82 0.24 0.67
cat 0.29 0.11 0.88 1.9 1.1 2.9 1.4 0.88 0.18 0.32
deer 0.46 0.05 1.4 1.5 1.3 0.71 1.9 2.3 0.17 0.29
dog 0.17 0.09 0.82 3.5 0.71 1.8 0.74 1.5 0.13 0.57
frog 0.15 0.1 1.1 2 2.4 0.45 1.4 1.2 0.09 1.1
horse 0.34 0.16 0.37 1.1 2.1 1.2 1.4 1.1 0.14 2.1
ship 2.3 0.68 0.14 0.18 0.26 0.03 0.37 0.98 0.8 4.3
truck 0.89 2.4 0.09 0.32 0.3 0.05 2 1.8 0.75 1.4
plane car bird cat deer dog frog horse ship truck
1 2 3 4 5

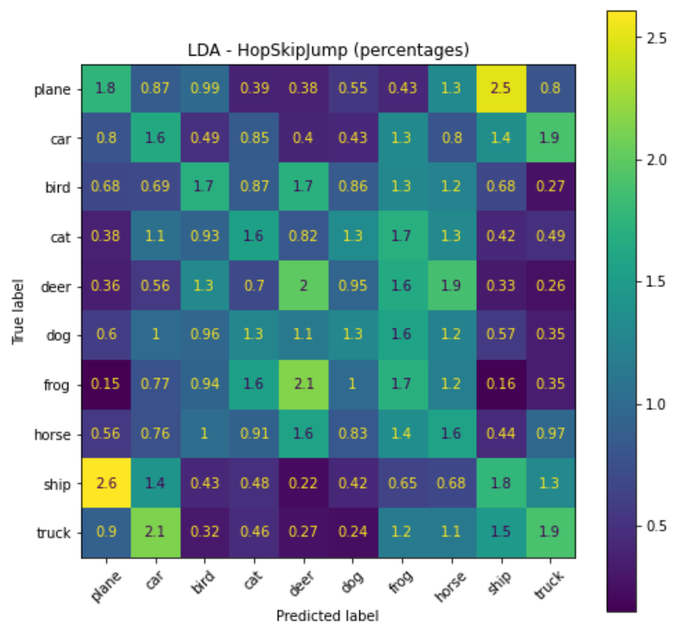

CNN-DF
plane AUC = 0.76
car AUC = 0.65
bird AUC = 0.77
cat AUC = 0.82
deer AUC = 0.80
dog AUC = 0.82
frog AUC = 0.74
horse AUC = 0.67
ship AUC = 0.60
truck AUC = 0.64
Macro average AUC = 0.73

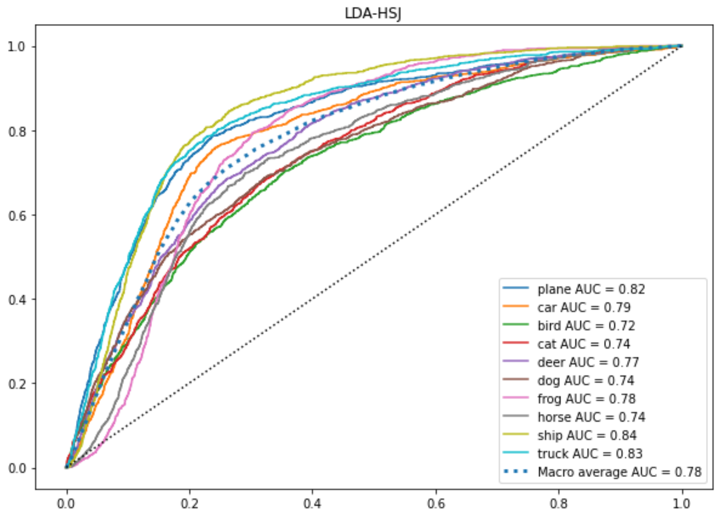

LogisticRegression - DeepFool (percentages)
True label
Predicted label
plane 1.1 2.3 0.77 0.48 0.32 0.5 1.4 1.1 1.4 0.73
car 0.24 2.1 0.38 0.42 0.3 0.38 4.1 0.61 0.37 1.1
bird 0.34 2.3 1.1 0.69 1.2 0.78 1.7 1.2 0.35 0.29
cat 0.16 2.2 0.56 1.1 0.8 1.3 2 1.4 0.12 0.4
deer 0.14 1.8 0.86 0.63 1.6 0.87 2 1.6 0.14 0.28
dog 0.29 2.8 0.63 0.75 0.76 1.2 1.9 1.1 0.22 0.23
frog 0.07 3.6 0.45 0.69 0.99 0.71 2.3 1 0.04 0.22
horse 0.2 1.3 0.7 0.64 1 0.45 3 1.7 0.14 0.85
ship 1.5 2 0.62 0.24 0.22 0.36 2.3 0.65 0.94 1.2
truck 0.52 2.4 0.33 0.26 0.19 0.19 3.4 0.57 0.64 1.6
plane car bird cat deer dog frog horse ship truck
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

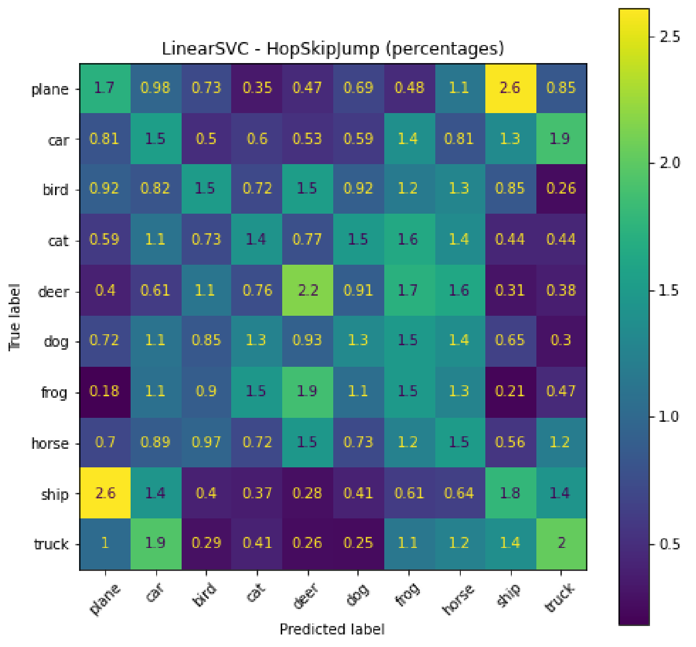

LR-DF
plane AUC = 0.55
car AUC = 0.45
bird AUC = 0.59
cat AUC = 0.62
deer AUC = 0.65
dog AUC = 0.57
frog AUC = 0.47
horse AUC = 0.52
ship AUC = 0.56
truck AUC = 0.54
Macro average AUC = 0.55

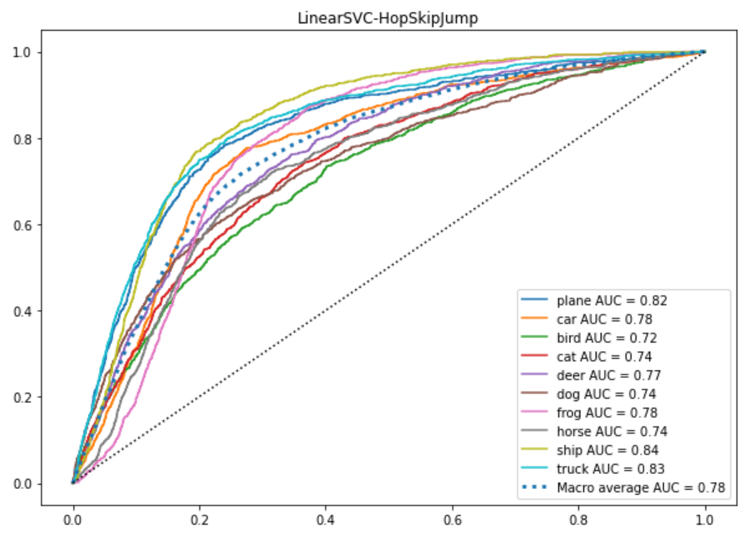

LogisticRegression - HopSkipJump (percentages)
plane 1.9 0.9 1.3 0.45 0.41 0.56 0.34 1.1 2.5 0.62
car 0.91 1.5 0.64 0.82 0.48 0.51 1.2 0.84 1.2 2
bird 0.82 0.75 1.5 0.75 1.7 1 1.2 1.3 0.71 0.27
cat 0.44 0.9 1.1 1.6 1 1.6 1.4 1.1 0.31 0.56
deer 0.35 0.61 1.5 0.85 2.1 1 1.5 1.6 0.31 0.28
dog 0.67 1.1 1.1 1.4 1.1 1.6 1 1.2 0.57 0.29
frog 0.15 1 0.9 1.3 2.1 1.2 1.4 1.3 0.15 0.35
horse 0.75 0.77 1.1 0.96 1.8 0.79 0.9 1.5 0.49 0.92
ship 2.8 1.4 0.55 0.42 0.28 0.43 0.52 0.56 1.7 1.4
truck 1.2 1.8 0.42 0.39 0.4 0.31 0.85 1.1 1.5 1.9
True label
plane car bird cat deer dog frog horse ship truck
Predicted label
2.5 2.0 1.5 1.0 0.5

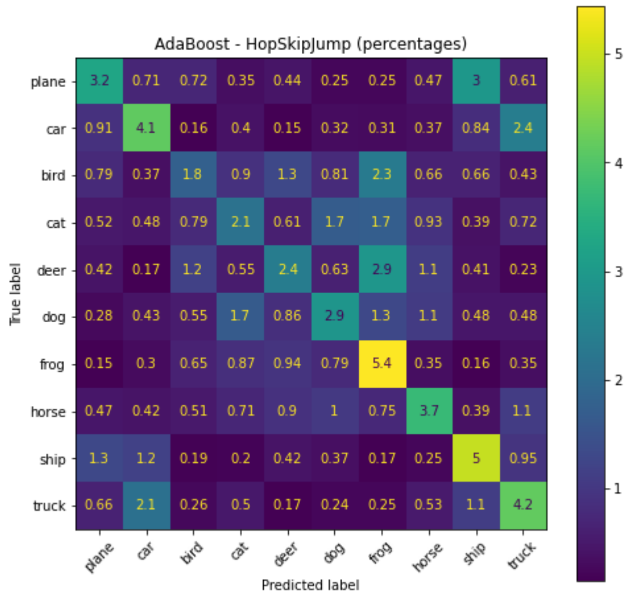

LogisticRegression - HopSkipJump
plane AUC = 0.82
car AUC = 0.79
bird AUC = 0.71
cat AUC = 0.75
deer AUC = 0.77
dog AUC = 0.75
frog AUC = 0.80
horse AUC = 0.75
ship AUC = 0.84
truck AUC = 0.84
Macro average AUC = 0.78
0.0 0.2 0.4 0.6 0.8 1.0

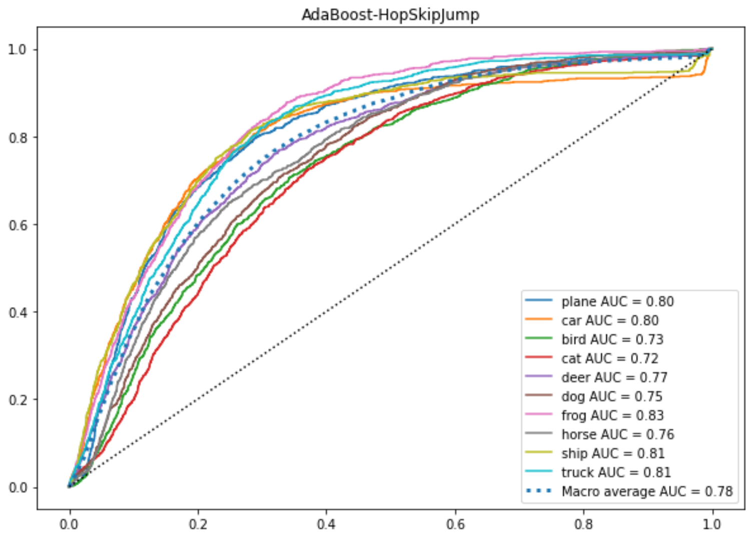

LDA - HopSkipJump (percentages)
plane 1.8 0.87 0.99 0.39 0.38 0.55 0.43 1.3 2.5 0.8
car 0.8 1.6 0.49 0.85 0.4 0.43 1.3 0.8 1.4 1.9
bird 0.68 0.69 1.7 0.87 1.7 0.86 1.3 1.2 0.68 0.27
cat 0.38 1.1 0.93 1.6 0.82 1.3 1.7 1.3 0.42 0.49
deer 0.36 0.56 1.3 0.7 2 0.95 1.6 1.9 0.33 0.26
dog 0.6 1 0.96 1.3 1.1 1.3 1.6 1.2 0.57 0.35
frog 0.15 0.77 0.94 1.6 2.1 1 1.7 1.2 0.16 0.35
horse 0.56 0.76 1 0.91 1.6 0.83 1.4 1.6 0.44 0.97
ship 2.6 1.4 0.43 0.48 0.22 0.42 0.65 0.68 1.8 1.3
truck 0.9 2.1 0.32 0.46 0.27 0.24 1.2 1.1 1.5 1.9
True label
plane car bird cat deer dog frog horse ship truck
Predicted label
2.5 2.0 1.5 1.0 0.5

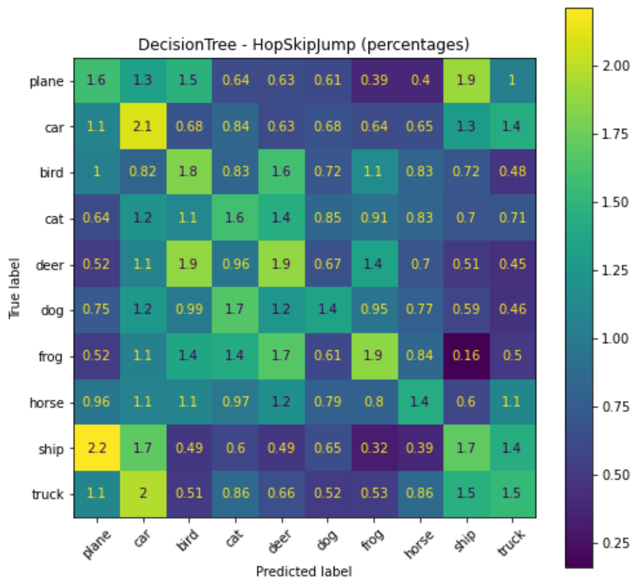

LDA-HSJ
plane AUC = 0.82
car AUC = 0.79
bird AUC = 0.72
cat AUC = 0.74
deer AUC = 0.77
dog AUC = 0.74
frog AUC = 0.78
horse AUC = 0.74
ship AUC = 0.84
truck AUC = 0.83
Macro average AUC = 0.78
0.0 0.2 0.4 0.6 0.8 1.0

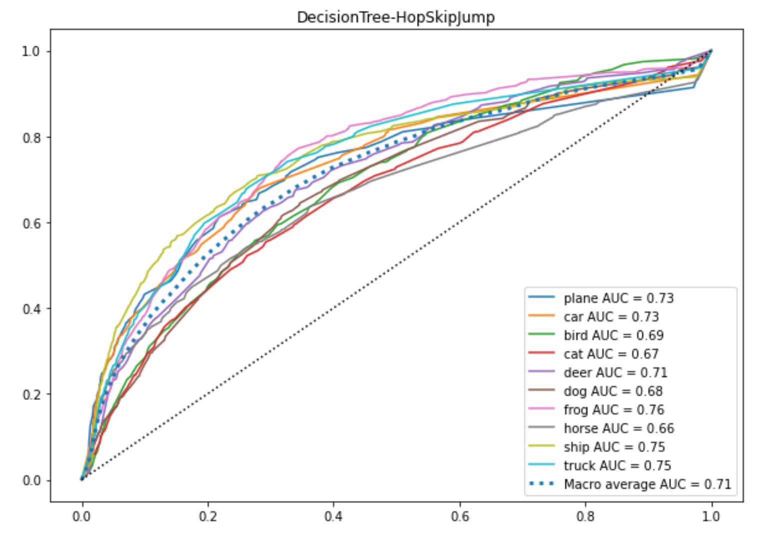

LinearSVC - HopSkipJump (percentages)
True label
Predicted label
plane car bird cat deer dog frog horse ship truck
plane 1.7 0.98 0.73 0.35 0.47 0.69 0.48 1.1 2.6 0.85
car 0.81 1.5 0.5 0.6 0.53 0.59 1.4 0.81 1.3 1.9
bird 0.92 0.82 1.5 0.72 1.5 0.92 1.2 1.3 0.85 0.26
cat 0.59 1.1 0.73 1.4 0.77 1.5 1.6 1.4 0.44 0.44
deer 0.4 0.61 1.1 0.76 2.2 0.91 1.7 1.6 0.31 0.38
dog 0.72 1.1 0.85 1.3 0.93 1.3 1.5 1.4 0.65 0.3
frog 0.18 1.1 0.9 1.5 1.9 1.1 1.5 1.3 0.21 0.47
horse 0.7 0.89 0.97 0.72 1.5 0.73 1.2 1.5 0.56 1.2
ship 2.6 1.4 0.4 0.37 0.28 0.41 0.61 0.64 1.8 1.4
truck 1 1.9 0.29 0.41 0.26 0.25 1.1 1.2 1.4 2
2.5 2.0 1.5 1.0 0.5

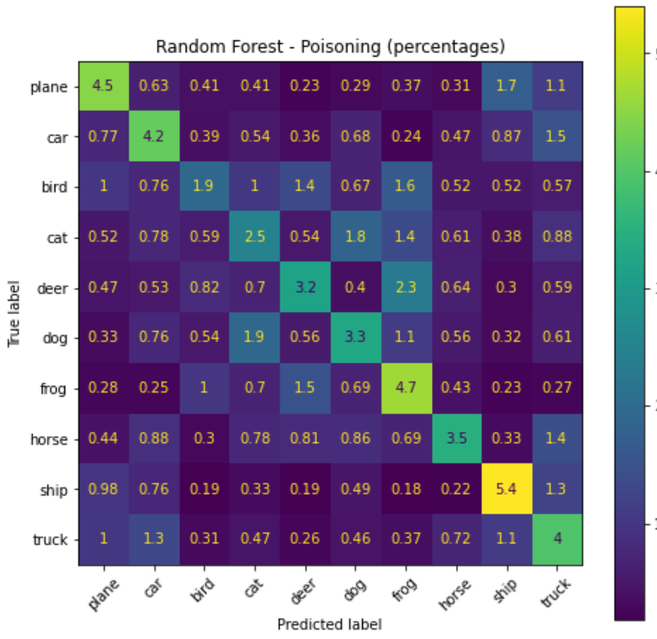

LinearSVC-HopSkipJump
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
plane AUC = 0.82
car AUC = 0.78
bird AUC = 0.72
cat AUC = 0.74
deer AUC = 0.77
dog AUC = 0.74
frog AUC = 0.78
horse AUC = 0.74
ship AUC = 0.84
truck AUC = 0.83
Macro average AUC = 0.78

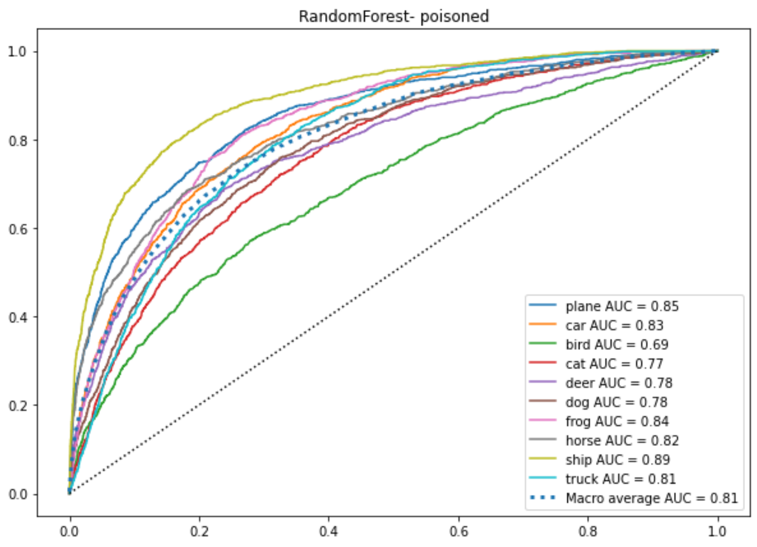

AdaBoost - HopSkipJump (percentages)
True label
Predicted label
plane car bird cat deer dog frog horse ship truck
plane 3.2 0.71 0.72 0.35 0.44 0.25 0.25 0.47 3 0.61
car 0.91 4.1 0.16 0.4 0.15 0.32 0.31 0.37 0.84 2.4
bird 0.79 0.37 1.8 0.9 1.3 0.81 2.3 0.66 0.66 0.43
cat 0.52 0.48 0.79 2.1 0.61 1.7 1.7 0.93 0.39 0.72
deer 0.42 0.17 1.2 0.55 2.4 0.63 2.9 1.1 0.41 0.23
dog 0.28 0.43 0.55 1.7 0.86 2.9 1.3 1.1 0.48 0.48
frog 0.15 0.3 0.65 0.87 0.94 0.79 5.4 0.35 0.16 0.35
horse 0.47 0.42 0.51 0.71 0.9 1 0.75 3.7 0.39 1.1
ship 1.3 1.2 0.19 0.2 0.42 0.37 0.17 0.25 5 0.95
truck 0.66 2.1 0.26 0.5 0.17 0.24 0.25 0.53 1.1 4.2
5 4 3 2 1

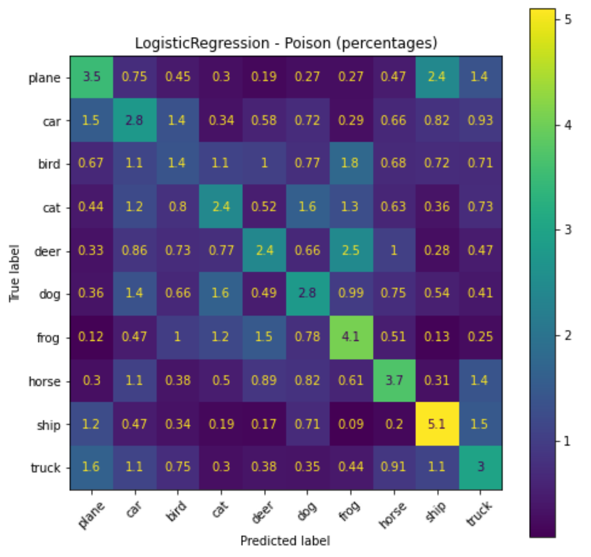

AdaBoost-HopSkipJump
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
plane AUC = 0.80
car AUC = 0.80
bird AUC = 0.73
cat AUC = 0.72
deer AUC = 0.77
dog AUC = 0.75
frog AUC = 0.83
horse AUC = 0.76
ship AUC = 0.81
truck AUC = 0.81
Macro average AUC = 0.78

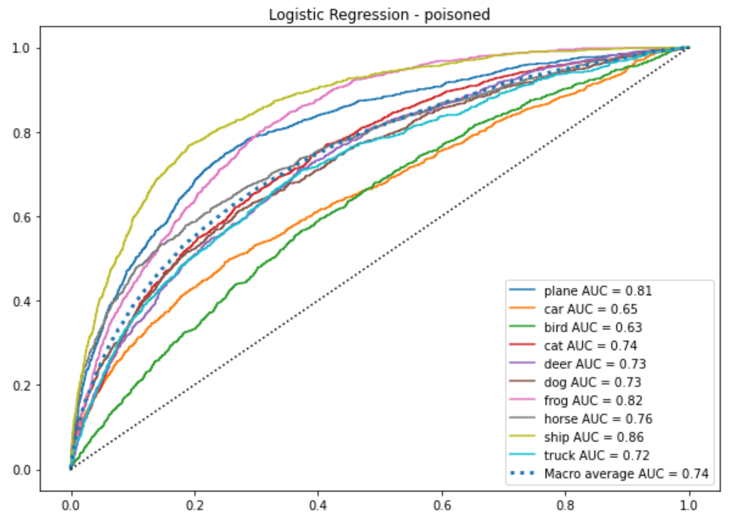

DecisionTree - HopSkipJump (percentages)
True label
Predicted label
plane
car
bird
cat
deer
dog
frog
horse
ship
truck
plane 1.6 1.3 1.5 0.64 0.63 0.61 0.39 0.4 1.9 1
car 1.1 2.1 0.68 0.84 0.63 0.68 0.64 0.65 1.3 1.4
bird 1 0.82 1.8 0.83 1.6 0.72 1.1 0.83 0.72 0.48
cat 0.64 1.2 1.1 1.6 1.4 0.85 0.91 0.83 0.7 0.71
deer 0.52 1.1 1.9 0.96 1.9 0.67 1.4 0.7 0.51 0.45
dog 0.75 1.2 0.99 1.7 1.2 1.4 0.95 0.77 0.59 0.46
frog 0.52 1.1 1.4 1.4 1.7 0.61 1.9 0.84 0.16 0.5
horse 0.96 1.1 1.1 0.97 1.2 0.79 0.8 1.4 0.6 1.1
ship 2.2 1.7 0.49 0.6 0.49 0.65 0.32 0.39 1.7 1.4
truck 1.1 2 0.51 0.86 0.66 0.52 0.53 0.86 1.5 1.5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25


DecisionTree-HopSkipJump
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
plane AUC = 0.73
car AUC = 0.73
bird AUC = 0.69
cat AUC = 0.67
deer AUC = 0.71
dog AUC = 0.68
frog AUC = 0.76
horse AUC = 0.66
ship AUC = 0.75
truck AUC = 0.75
Macro average AUC = 0.71


Random Forest - Poisoning (percentages)
True label
Predicted label
plane
car
bird
cat
deer
dog
frog
horse
ship
truck
plane 4.5 0.63 0.41 0.41 0.23 0.29 0.37 0.31 1.7 1.1
car 0.77 4.2 0.39 0.54 0.36 0.68 0.24 0.47 0.87 1.5
bird 1 0.76 1.9 1 1.4 0.67 1.6 0.52 0.52 0.57
cat 0.52 0.78 0.59 2.5 0.54 1.8 1.4 0.61 0.38 0.88
deer 0.47 0.53 0.82 0.7 3.2 0.4 2.3 0.64 0.3 0.59
dog 0.33 0.76 0.54 1.9 0.56 3.3 1.1 0.56 0.32 0.61
frog 0.28 0.25 1 0.7 1.5 0.69 4.7 0.43 0.23 0.27
horse 0.44 0.88 0.3 0.78 0.81 0.86 0.69 3.5 0.33 1.4
ship 0.98 0.76 0.19 0.33 0.19 0.49 0.18 0.22 5.4 1.3
truck 1 1.3 0.31 0.47 0.26 0.46 0.37 0.72 1.1 4
5
4
3
2
1


RandomForest- poisoned
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
plane AUC = 0.85
car AUC = 0.83
bird AUC = 0.69
cat AUC = 0.77
deer AUC = 0.78
dog AUC = 0.78
frog AUC = 0.84
horse AUC = 0.82
ship AUC = 0.89
truck AUC = 0.81
Macro average AUC = 0.81





References

General:

https://www.cs.toronto.edu/~kriz/cifar.html

https://pyimagesearch.com/2020/10/19/adversarial-images-and-attacks-with-keras-and-tensorflow/

https://viso.ai/deep-learning/adversarial-machine-learning/

https://towardsdatascience.com/evaluating-adversarial-examples-with-similarity-metrics-in-python-13acb9b5fa9f

https://medium.com/cltc-bulletin/adversarial-machine-learning-43b6de6aafdb

https://arxiv.org/abs/1712.09665

https://towardsdatascience.com/deepfool-a-simple-and-accurate-method-to-fool-deep-neural-networks-17e0d0910ac0

https://arxiv.org/abs/1708.06733

https://people.csail.mit.edu/madry/lab/cleanlabel.pdf

Packages/Libraries:
https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/evasion.html#hopskipjump-attack
https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/evasion.html#deepfool

https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html