DeeJanus Experiment

Experiments

Important Python files implemented for this experiment:

- ***work.py*** (obtain the 500 examples from training set at random from well classified ones and save them in janus_dataset_comparison.h5)
- ***model_create_Experiment.py*** (train a new tuned model using the adversial example created by DeepJanus and the original training set, save the model in the /models folder under name nnClassifierTunedX.h5)
- ***model_create_Experiment_Advanced.py*** (train a new tuned model using the adversial example created by PGD and the best of the tuned models, save the model in the /models folder under name nnClassifierTunedAdvanced.h5)
- ***model_create_Experiment_Final.py*** (train a new tuned model using the adversial example created by DeepJanus on the tuned model, save the model in the /models folder under name nnClassifierFineTunedX.h5)
- ***main.py*** (create the adversial examples based on the janus_dataset_comparison.h5)
- ***main2.py*** (create the adversial examples based on the janus_dataset_comparison.h5 for the tuned model)
- ***config.py*** (allow to alternate how the DeepJanus provide the adversial examples)
- ***adversarial_toolbox_evaluate.py*** (evaluate the robustness of each models using an attack from the adversarial toolbox such as PGD on the tuned models)
- ***adversarial_toolbox_evaluate_Advanced.py*** (evaluate the robustness of each models using an attack from the adversarial toolbox such as PGD on the final tuned models)

1. The implementation is working as intended for the MNIST dataset. In order to run DeepJanus locate the file "main.py" located in the path "deepJanus/DeepJanus-MNIST". Once launched, it produces a folder "runs" in which we can observe if the DeepJanus have managed to produce two members at the frontier of behaviors. Note: The accuracy for the base model is offset, offering a 84% of accuracy instead of the 99% stated.

2. In progress, but not necessary to have the project running and experimented.

3. The non-robust model cnnClassifier5.h5 can be attacked using the DeepJanus algorithm provided in this paper. In this experiment we are going to robust the model by using the members found by DeepJanus. In order to do so, we are going to use DeepJanus on 500 train-data in MNIST that have been adequately well defined by the model. We have implemented the file "work.py" located in the path "deepJanus/DeepJanus-MNIST", it allows to randomly takes 500 train-data from MNIST that will be used later in "main.py" for DeepJanus.
   In order to check the successfulness of DeepJanus, we are going to runs it once and evaluate how much time it has been successful.
   On the run using a population of 500 from the training data of MNIST we obtained for different seeds:
   - [1] 4/100 successful. Providing adversarial examples for labels 3,8 and 9.
   - [2] 7/100 successful. We have obtained adversarial example for labels of 2,3, 6 and 7.
   - [3] 11/100 successful. We have obtained adversarial example for labels of 7, 8 and 9.
   - [4] We have provided a last run in order to check the expansion possibilities of DeepJanus. 9/100 successful. The results does not differ much. It seems that based on the random seed selected from the training data, the DeepJanus manage to provide more adversarial examples.

4. We are now mixing the examples produced by DeepJanus such that we use the images which were miss-classified, checking if the expected label is different from the predicted label. We are extracting both the images and the expected

label and add it to the training data. The original training of the model is using 12 epochs, however here we are going to use only 5 epochs. The new training of the model using the DeepJanus examples can be found in the file "model_Create_Experiment.py". We are going to train 2 models using 10 DeepJanus runs. Giving us 3 models.

- cnnClassifierTuned1.h5
- cnnClassifierTuned2.h5
- cnnClassifierTuned3.h5

5. We are now performing some adversarial attacks (PGD) on the three different models, and checking the robustness of each of them, by predicting the model on original data, then with adversarial samples. Giving us the following robustness, for each of the classifiers respectively:
    - Accuracy on benign: 98.6% / Accuracy on attack: 49.2%
    - Accuracy on benign: 98.6% / Accuracy on attack: 40.400000000000006%
    - Accuracy on benign: 98.4% / Accuracy on attack: 67.4%
    - Accuracy on benign: 98.4% / Accuracy on attack: 34.4%

6. Based on the results obtained in 5., we are going to train the most robust model with the adversarial samples produced by the PGD attack, in order to obtain the best possible model capable to resist to adversarial attacks. As we can observe, we have chosen to retrain cnnClassifierTuned2.h5, and obtained the power of predictability of the model being 0.99%. The new model is named cnnClassifierTunedAdvanced.h5.

7. Once again, we are applying the DeepJanus attacks on the final robust model in order to obtain three even more robust models, respectively:

- cnnClassifierFineTuned1.h5
- cnnClassifierFineTuned2.h5
- cnnClassifierFineTuned3.h5

8. Finally, we are attacking once again with PGD the three final tuned models and check the robustness of each of them. We obtained respectively for each of the models the power of predictability:
    - Not concerned.
    - Accuracy on benign: 98.0% / Accuracy on attack: 60.199999999999996%
    - Accuracy on benign: 97.8% / Accuracy on attack: 58.599999999999994%

      Accuracy on benign: 98.2% / Accuracy on attack: 62.4%