Homework Assignment 2

(Programming Category)

Student Name:______________________

Student Session: cs6220-A

You are given 2 types of programming problems. You only need to choose one of the 2 problems as your second homework. Each of the two problems has multiple options, you are required to choose only one option. Feel free to choose any of your favorite programming languages, such as Java, C, Perl, Python, and so forth.

You are given three types of programming problems in the second homework assignment.

- **Problem 1** is on understanding new security vulnerabilities in using well-trained deep learning models. (Pages 1-5)
- **Program 2** is on performance tuning of ML model training with either shallow or deep learning package from a popular ML library. (Pages 6-11)

Post Date: on Monday of Week 4
Due Date: midnight on Friday of Week 5

Problem 1. Understanding new security vulnerabilities in deep learning.

This problem has two options: security vulnerabilities in using well-trained DNN models for image classification (option 1) and for object detection models (option 2).

For each option, we design the hands-on learning in three stages to complement the lectures on this topic:

(i) You are provided with a web-based API platform to learn visually how adversaries may attack a well-trained DNN model to make it fail miserably.

(ii) You are provided with an open-source attack package and the pre-trained model(s) on benchmark dataset(s) to launch attacks and report your experience.

(iii) You are asked to report your learning experience in two aspects: the experience with the API visual learning (quiz 1) and the experience with the hands-on attack code (quiz 2).

Option 1: Understanding security vulnerabilities of using a well-trained DNN models for image classification

Requirements:

1) Visit the URL here: https://security4ai-vlab.herokuapp.com/, choose "Image Classification" to be the "Task", and learn how the attacks are happening through visualization of attack examples.

   Hint (how): first choose a dataset provided (MNIST, CIFAR-10), and the pre-trained model on the dataset. Click Confirm. Then select an attack algorithm from the drop-down menu with a total of six attack algorithms (e.g., FGSM), you will see each row has three images, the original, the test result with accuracy printed and the attacked image with adversarial perturbation, which leads to error in classification for the same model. Each row is a new test image.

   Hint (what you can learn): (i) You can learn adverse effects of different attacks by choosing a test image in the left panel and comparing visual results of using different attack methods to attack this same test example. (ii) You can also study whether test images from different classes may have different degree of vulnerabilities under the same attack.

2) Fill in the questionnaire at URL https://gatech.instructure.com/courses/201470/quizzes/305033 (or log into our course on Canvas under quizzes and click Quiz 1), which summarize your learning experience so far.

3) Visit the URL: https://git-disl.github.io/GTAttackPod/ to fork the attack code. You can also click the link from the GTAttackPod link in https://security4ai-vlab.herokuapp.com/.

4) Choose one of the datasets provided under GTAttackPod (MNIST, CIFAR-10), and the pre-trained model on the dataset.

5) Then select two attack algorithms from the set of attack algorithms provided in GTAttackPod (e.g., FGSM and JSMA), and run them on the chosen DNN image classifier trained on the chosen dataset (say CIFAR-10). Collect the average test accuracy over the total tested inputs under each of the attack algorithms.

6) Once you have done step 5), you can choose two of the following three options to measure the adverse effects of the adversarial examples you have generated in step 5).

   a. **Adverse effect on different depths of CNNs.** Choose another two or more pre-trained models on CIFAR-10 with very different DNN layers in terms of depths (ranging from 10s to 100s for example), using the attack algorithm to examine whether adversarial examples are sensitive to deeper NNs. Report your results.

b. **Test transferability of your generated adversarial examples** on a pre-trained CIFAR-10 (say trained by DenseNet 40) or MNIST (say trained by CNN-7) model, ideally trained using different backbone algorithms. Using 100 test adversarial examples to produce average transferability of your attack algorithm to this second CIFAR-10 or MNIST model.

c. **Divergence of attack effects.** There are two types of divergence we talked about in class. The first divergence is on instances of the same class and on two different models. For the adversarial examples generated for the same class, show their divergence in terms of attack effect (recall lecture 7). For example, how a random (untargeted) attack to digit 0 distributed over the remaining 9 different digit classes. The second divergence is on different models. For instance, attack to digit 4 under model 1 will have one type of distribution and under model 2 will have a very different distribution. You can use 100 adversarial examples you generated for each model to show such distribution difference. You can also use transferability to generate the adverse effect on the second model and show the different distribution under attack transferability.

7) **Optional:** If you have time and prefer to try something new, then here is an option (not mandatory).
GTAttackPod only includes the implementations of the list of attack algorithms on MNIST and CIFAR-10 under the respective pre-trained model. You are encouraged to modify the attack code so you can attack other pretrained models on new datasets (say ImageNet or CheXay) or other pre-trained CIFAR-10 or MNIST models from different DL vendors, report your experiences and provide your modified attack code.

8) Fill in the questionnaire in URL https://gatech.instructure.com/courses/201470/quizzes/305047 (or log into our course on Canvas under quizzes and click Quiz 2), to summarize your learning experience by combining the API visual learning and hands-on attack code learning.

Input analysis:

1) Provide a summary of your pre-trained models and datasets. For each dataset, provide 10 example inputs under five different classes, 2 per class.
2) Provide a summary of the two attack algorithms of your choice.
3) Provide the attack examples you generated for the 10 examples you listed

in 1).

Output analysis:

1) Include a note that you have provided your answer to the questionnaire posted as quiz 1 on Canvas (under quizzes), refer requirement 2) above.
2) Provide test accuracy measurement and average test time per example of the two well trained models under no attack.
3) Compare the two models under attacks with the two models under no attack on test accuracy and time.
4) Plot the results from 1) + 2) into a table for easy comparison.
5) Provide 10 visual examples for both datasets to illustrate the adverse effects of the attacks: clean input image, the amount of perturbation added by attack algorithm via iterative learning (ideally show visual results under 2-3 iterations), and the final input image under attack to compare with the clean image without adversarial perturbation.
6) Include a note that you have provided your answer to the questionnaire posted as quiz 2 on Canvas (under quizzes), refer requirement 8) above. This questionnaire summarizes your learning experience by combining lecture, the API visual learning and hand-on attack code programming experience.
7) Provide additional analysis and comments on the learning experience with this problem (optional).

Option 2: Understanding security vulnerabilities of using a well-trained DNN models for object detection

Requirements:

1) Visit the URL here: https://security4ai-vlab.herokuapp.com/, choose "Object Detection" to be the "Task" and learn how the attacks are happening through visualization of attack examples.

   Hint (How): first choose a dataset (VOC), and choose one of the five pretrained models on VOC, Click Confirm. Then select an attack algorithm from the drop-down menu with a total of six attack algorithms (e.g., TOG vanishing, TOG mislabeling, etc.), you will see each row has three images, the original input, the test result under no attack with accuracy printed and the test result under attack, the attack image is injected with adversarial perturbation on pixels, which leads to error in object detection (object vanishing, object fabrication, object mislabeling). Each row is a new test image with middle under no attack and right under attack. You can view different attacks and compare visual results across different attacks. You can also select different pre-trained models on VOC and identify difference in attack effect, learning different vulnerabilities for different pre-trained models on the same VOC dataset.

Hint (what you can learn): (i) You can compare adverse effects of different attacks by choosing a test image in the left panel and comparing visual results under different attacks to the same model. (ii) You can also compare the adverse effect of one attack method on different pre-trained models on VOC by selecting a couple of test examples on the left panel to learn that different models may have different degree of vulnerabilities under the same attack.

2) Fill in the questionnaire at URL https://gatech.instructure.com/courses/201470/quizzes/305033 (or log into our course on Canvas under quizzes and click Quiz 1), which summarize your learning experience so far.

3) Visit the URL here: https://git-disl.github.io/TOG/ to fork the attack code for TOG.

4) The dataset is VOC and there are several pretrained model on VOC: YOLOv3 (Darknet53), YOLOv3 (MobileNetV1), SSD300 (VGG16), SSD512 (VGG16), and Faster R-CNN (VGG16). All of them are trained on PASCAL VOC. You are asked to choose two of them to report your hands-on experience.

5) Run the two attack algorithms from TOG family of attacks (say TOG vanishing, TOG Mislabeling) on each of your chosen pre-trained detection model with a test dataset (say 100 test images). Collect model average test accuracy under attack to compare with no attack results.

6) Once you have done step 5), you can choose two of the following three options to measure the adverse effects of the adversarial examples you have generated in step 5).

    a. **Test transferability of your generated adversarial examples** on other pretrained object detection models on VOC dataset, ideally trained using different backbone algorithms. Using 100 test adversarial examples to produce average transferability of the attack algorithm you used (say TOG mislabeling or TOG general).
    b. **Divergence of attack effects on objects of the same class.** There are two types of divergence we talked about in class. The first divergence is on instances of the same class and on two different models. For the adversarial examples generated for the same class, show their divergence in terms of attack effect (recall lecture 7). For example, how a random (untargeted) attack to people objects of the person class is distributed over test examples containing person.
    c. **Divergence of attack effects on two different models.** The second divergence is on different models. For instance, attack to

person objects under model 1 will have one type of distribution and the same attack to the same test set under model 2 will have a very different distribution. You can use 100 adversarial examples you generated for each model to show such distribution difference. Of course, you can also use transferability to generate the adverse effect on the second model and show the different distribution under attack transferability.

7) **Optional:** TOG attacks are implemented on five pre-trained detection models on VOC. You are encouraged to modify the attack code so you can attack other pretrained models on new datasets such as COCO, INRIA, or other pre-trained VOC models from different DL vendors, report your experiences and provide your modified attack code.

8) Fill in the questionnaire in URL https://gatech.instructure.com/courses/201470/quizzes/305047 (or log into our course on Canvas under quizzes and click Quiz 2), to summarize your learning experience by combining the API visual learning and hands-on attack code learning.

Input analysis:

1) Provide a summary of your pre-trained models and datasets. For each dataset, provide 10 example inputs under five different classes, 2 per class.
2) Provide a summary of the two attack algorithms of your choice.
3) Provide the attack examples you generated for the 10 examples you listed in 1).

Output analysis:

1) Include a note that you have provided your answer to the questionnaire posted as quiz 1 on Canvas (under quizzes), refer requirement 2) above.

2) Provide test accuracy measurement and average test time per example of the two well trained models under no attack.

3) Compare the two models under attacks with the two models under no attack on test accuracy and time.

4) Plot the results from 1) + 2) into a table for easy comparison.

5) Provide 10 visual examples for both datasets to illustrate the adverse effects of the attacks: clean input image, the amount of perturbation added by attack algorithm via iterative learning (ideally show visual results under 2-3 iterations), and the final input image under attack to compare with the clean image without adversarial perturbation.

6) Fill in the second questionnaire in URL https://gatech.instructure.com/courses/201470/quizzes/305047 (or log into our course on Canvas under quizzes and click Quiz 2), to summarize your learning experience by combining the API visual learning and hand-on attack code learning. Include this questionnaire in your HW2 deliverable.

7) Provide additional analysis and comments on the learning experience with this problem (optional).

Problem 2. Understanding Performance Tunning for Training High Accuracy Deep or Shallow Models

In this problem, you are given 2 options. In each option, you are asked to train multiple models using different hyperparameter configurations (option 1) or using different dataset complexity plus hyperparameter configurations (option 2).

Problem 1.1 Performance Tunning for Deep or Shallow Learning

You are asked to train 5 models for the same learning task and dataset, each uses one configuration of the hyperparameters: the default configuration from the package you download, and the four different configurations you are asked to change and each of the four should have only one hyperparameter different to the baseline to help with the analysis on performance tuning results. If you use CPU without GPU card, then create a smaller dataset. Here are the steps:

(1) Choose your favorite ML framework, say TensorFlow, Scikit Learn, PyTorch, etc.. In the rest of the requirement, we will use TensorFlow as an example.

(2) After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

(3) Choose a dataset and a machine learning (ML) algorithm to train a baseline ML model using the default hyperparameters, say CNN3 MNIST, which use a 3 layer CNN to train a 10-class neural network classifier on MNIST dataset. Given MNIST dataset has 60K images. You may choose a subset of them, say 6000 total, evenly distributed for 10 classes, say 600 per class.

(4) You are asked to show the default settings of the hyperparameters of the TensorFlow CNN algorithm you used to train a k-class classification model, such as the CNN layers used, the #kernels (weight filters) used, the feature map sizes, the minibatch size, the #iterations (#epochs), the loss optimizer (e.g., SGD, Adam, etc.)

(5) Train your model using the algorithm and the dataset you have selected. Report the output of the trained K-class model, including

- the storage size of the model in MB,
- the test accuracy (use a small portion of the training data as training validation),
- the training time and
- the test time.
- In addition, you may want to show the confusion matrix for all K classes. For K>10, you may choose to show a selected subset of classes. [Hint: see lecture 6 on confusion matrix]

(6) **Outlier Test Scenario**. When performing testing on a k-class classifier you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by using 10 or more images that do not belong to the dataset of your learning task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars which will be the out of distribution data. Perform outlier test on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set in your report.

(7) For each of the 10 outlier queries, and run each of the outlier tests at least three times and report your results, including
- The results from the 3 times of the same outlier query: show if they return the same prediction result or not;
- Make an attempt to elaborate on why the results from 3 times of test using the same test example may result in very different probability vectors.

Hint: There are two ways to perform the above tests: (i) select 3 different pre-trained models on the same dataset, run each of the outlier tests against three models and analyze your results on the above two bullets. (ii) perform the same outlier query against the same model. Report your finding. Ideally compare with case (i). Elaborate your comparison results and your thought on the results.

(8) Now choose at least two different types of hyperparameters (say different learning rate functions and different #kernel filers) and for each type, select two settings (say fixed learning rate with two different values, plus 10 kernel filers v.s. 5 kernel filers) and use the same training dataset (say MNIST) to train 4 different models for the same learning task (e.g., MNIST with 10 classes), each using one changed hyperparameter configuration, and compare the performance of these 4 newly trained models with the baseline model you have trained in step (1)-(7) in
    - training time,
    - test accuracy,
    - test time,
    - the storage size of the trained model

(9) Analyze the comparison results based on the confusion matrix and the different hyperparameter configurations you have used in the 5 models you have trained.

Hints: Hyperparameters include (i) different #kernels (weight filters) used, (ii) different learning rate function or settings, say different value for a constant learning rate; (iii) different convergence condition, e.g., #iterations (or #epoches), (iv) other hyperparameters of your own choice, including different training algorithm: say you used LeNet MNIST and now change to another algorithm, such as ResNet32, ResNet64, MobiNet, VGG16, …

Deliverable:

(1) provide URL of your open source code package and the dataset download.
(2) Screen shots of your execution process/environments
(3) **Input Analysis**: *Use a table to report your training configuration parameters*:
    a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB).
    b. choose and show 5 sample images per class for all K classes in the dataset.
    c. the training v.s. testing data split ratio and size used in your CNN training.
    d. You are asked to record the structure and default settings of the neural network (NN) algorithm you chose to train your k-class classifier, such as LeNet, or ResNet, or DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters (kernels) and the size of kernel (weight filter) for each layer of your CNN, the min-batch size, #epochs/#iterations (convergence), in a table for the configuration you use to train your four CNN classifiers.

e. You are asked to report the default #iterations used for training convergence.

(4) **Output Analysis:** *Report the performance comparison and analysis of each of your 5 CNN classifiers:*

a. You are asked to provide a table to compare the 5 models you have trained in terms of training time, training accuracy, testing time and testing accuracy.
b. You are asked to record the trained model size in MB in the above table for all 5 models.
c. You are asked to make at least three observations from your experimental comparison of the 5 models that you have trained on the same dataset under 5 different hyperparameter configurations (including the baseline model using the default hyperparameters).

Problem 1.2 Understanding Model Training for different dataset complexity

This assignment provides you with experience in tuning deep learning model configurations for training a deep neural network model on the same type of learning task, such as image classification, NLP, object detection, word embedding, graph embedding, etc.. You want to compare the accuracy and training time and test time on on two different types of benchmark datasets with varying complexity (e.g., color or grey images, color density, or different K classes, say CIFAR10 v.s. CIFAR100. If you are familiar with other DNN model training than image classifier, you are encouraged to use your favorite one.

Please refer to the Problem 1.1 on Software download, Dataset preparation, including training/test data partition.

Goal. The goal of this problem option is to get you to learn how different complexity and variations of datasets may impact on the performance of a deep neural network trained model even Choose a type of learning task, such as image classification, NLP, object detection, word embedding, graph embedding, etc..

In the following, we use image classifier as an example to outline the requirement.

You only need to work with an off-shell deep learning software framework such as TensorFlow, PyTorch or one of your favorite DL framework. It does not require you to write or modify any program code. Your job is simply to use TensorFlow to generate a total of N (N>=3) different CNN classifiers, one per dataset. Hence, you are asked to find three datasets for this option: say MNIST, CIFAR-10, LFW. Then compare their training and testing performance (time and accuracy).

For a beginner, you are recommended to train a binary classification model. Feel free to create a classification task with k classes (k=2, 10, 100 for example), depending on your machine capacity for running the TensorFlow training.

Deliverable: You are asked to train a total of 6 models and make the comparison. First, you train 2 models using datasets of 2 different complexity. Then choose the best performing classifier as the default baseline and choose to change the setting of one hyperparameter. You should consider two different hyperparameters to vary, such as two different learning rate settings, and two different settings of the kernel filers, either different # of kernel filers or different size of kernel filers (5x5 or 10x10). This will allow you to produce 4 alternative models to compare the accuracy and time results with this baseline default hyperparameter configuration from the download package, and provide analysis of your results. Here are the steps for this option:

Requirements: You are asked to vary a set of dataset-specific parameters in your training phase and to make observations and comparisons:

(1) Preparing at least two Datasets of different complexity:
   a. the input datasets in terms of varying classes. The number of images per class should be no smaller than 100.
   b. the input datasets in terms of varying resolutions (at least two different resolutions, e.g., 28x28, 64x64)
   c. or hybrid of both

   You may name your datasets as dataset-resolution-#class. D100-10, D1000-10, D-28x28-10, D-64x64-100.
   Hint: make sure you divide your dataset into training and testing.

(2) For each of two datasets from the same domain (say color images from ImageNet, e.g., cat v.s. dog), you are asked to use TensorFlow to train a different CNN classifiers and to make comparison on their training time, accuracy and test time and accuracy. Also include data complexity specification.

(3) **Outlier Test Scenario**. When performing testing on each of the N (>=3) classifiers you have trained, in addition to use the test dataset from the same collection, you are asked to perform an outlier test by creating 10 images that are outside the binary classification task. For example, if you choose dog and cat as your binary classification task, then create 10 photos of your favorite actor/actress, or favorite cars). Perform outlier test on your classifiers and report your results in a table titled outlier test. Also include 5 examples of your outlier test set in your report. Suggestion: For each of these 10 outlier photos, perform each query at least 3 times and report whether the results are the same or different and analyze why.

(4) Choose the model that has the highest test accuracy among the two models you have trained. For this classifier and its training dataset, you are asked to choose two of the following subtasks in the deliverable of Problem 1.2.
    a. Choose two different learning rate functions or different learning rate value for fixed learning rate, for example, for training 2 models, say MNIST-LR-fix, MNIST-LR-NStep.
    b. Choose different # of kernel filers (initial weights) used for training 2 models, say MNIST#10filers, MNIST-#5filters.
    c. Choose two different kernel filer sizes, say 5x5 or 10x10 … say MNIST-5x5Kernel, MNIST-10x10Kernel.
    d. For this CNN classifier and its training dataset, you are asked to change the default setting of #epoch to be 5x and 10x larger. Report your comparison.
       Note: if your training dataset has B mini-batches, then you will need about #iterations = #epochs x B.

    Compare the 4 models generated by changing hyperparameters with your chosen baseline mode in three tables: (i) table 1 with training time, test accuracy, test time; (ii) table 2 with different hyperparameter settings, (iii) table 3 with confusion matrix for at least 3 classes on TP, TN, FP, FN, Precision, Recall, F1 score.

Deliverables:

(1) Provide URL of your open source code packages
(2) Example screen shots of your execution process/environments
(3) Input Analysis: *Use a table to report your setup parameters for* ***each*** *training model built*:
    a. the input dataset (size, resolution, storage size in KB or MB per image, storage size of dataset in MB or GB). the training v.s. testing data split ratio and the size used in your CNN training. (Hint: If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)
    b. Choose 5 sample images and show each in their different dataset complexity versions.
    c. You are asked to record the default neural network (NN) model, such as LeNet, ResNet, Densenet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for your training and testing experiments. Also the same should be reported on the four models you trained for comparison (see step (4) above)
    d. Option: You are asked to report the error threshold used in the TensorFlow default configuration for backward propagation in each step. Usually it is documented for MNIST and CIFAR-10.

(4) **Output Analysis:** *For the set of 6 models you have trained, and report the performance comparison and analysis:*

(a) You are asked to provide a table (table 1) to compare the two classifiers (each is trained from one dataset configuration), in terms of training time, training accuracy, testing time and testing accuracy.

(b) For the best performing mode in (1), you are asked to train 4 different models with different hyperparameters. For example, choose two different types of hyperparameters (e.g., #filers, filer size), each with two different settings (10 kernel filers v.s. 5 kernel filers, and 5x5 filer size v.s. 10x10 filer size). Compare these 4 models with the baseline in table 2, with hyperparameter setting variation and training time, testing time and testing accuracy.

(c) You are asked to record the trained model size in MB for each of the 6 models into a table (table 3)

(d) You are asked to make at least three observations from your experimental comparison of the 6 models. You are encouraged to use confusion matrix and hyperparameter configuration differences for this analysis.