CS/ECE 407 Final Project: Homomorphic Encryption Applications in Machine Learning

This package is Jiazhi Chen and Yunxiao Song's final project for CS/ECE 407 Cryptography at the University of Illinois at Urbana-Champaign for Fall 2021.

Setting up the Environment

Our project uses the package TenSEAL. Installing it should be as simple as typing this command in a terminal:

$ pip install tenseal

Other libraries we used include pytorch, pandas, numpy, matplotlib, and torchvision. Users, please make sure you also have this packages installed. If not, you should be able to install them using this command:

$ pip install torch pandas numpy matplotlib torchvision

We have partitioned our codes into three python files, each representing a machine learning model that we experimented homomorphic encryption with.

Logistic Regression

All of our codes for logistic regression are in logistic.py. It takes in a dataset that has a categorical outcome (0 or 1), trains a logistic regression model on the plain data, evaluates the model on encrypted data, and eventually trains a logistic regression on the same data but in encrypted form. Currently, we are using the pima India dataset in pima_india_diabetes.csv. To run the program on a different dataset, one may edit logistic.py on line 25 to read a different .csv file. The file must satisfy these conditions: first, it is in .csv format and it is in the same directory as the program python file; second, it has a single column for outcome named "Outcome", and it should contain binary categorical values (0 or 1).

One might run the program by this command:

Logistic Regression

$ python3 logistic.py

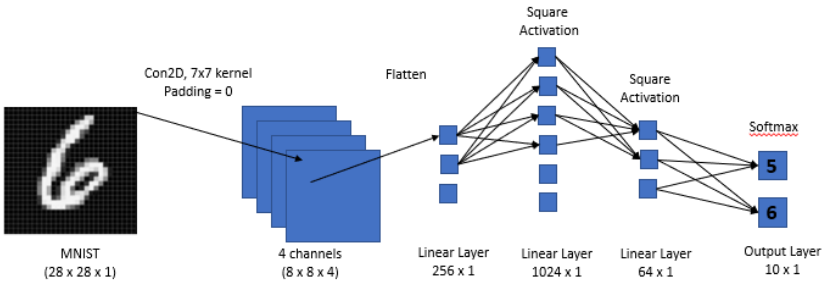
The program generates the outputs as we stated in our final report, for the logistic regression models. It first trains a logistic regression model on the plain data for 5 epochs and prints the accuracy. It then evaluates the accuracy on encrypted data and prints the accuracy. It eventually trains a logistic regression model on encrypted data and prints the final accuracy as well as the run time for each epoch.

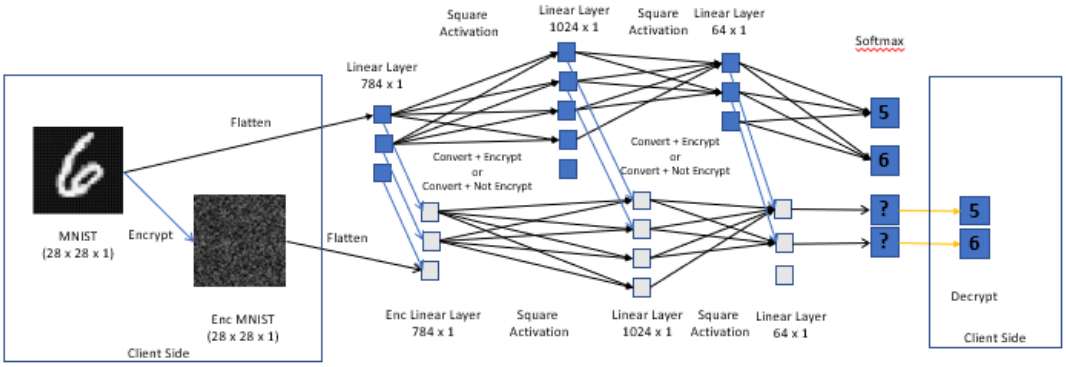
Be aware of the fact that the accuracies might slightly differ for each run. This is because in splitting the dataset, the 70% data points that go into the training set are sampled randomly. The accuracies won't differ too much.

Convolutional Neural Network

All of our codes for training and testing the CNN, FCNN baseline is in the following files, the data set will be automatically downloaded from the torch vision. The structure of the CNN model is as the following figure shows. We also included a figure which details the design of our model conversion between FCNN and EncFCNN. The conversion works the same way

between CNN and EncCNN. Furthermore, the Encrypted evaluation on deep learning is highly demanding and might trigger garbage collection mechanism in some environments. Therefore, we provided python notebook that can be run step by step as well.





One might train the model and see time measurement of CNN baseline by this command:

```
$ python3 CNN.py
# or just use notebook CNN.ipynb
```

One might check the Encrypted evaluation by simply call this command

```
$ python3 EncCNN.py
# or just use notebook EncCNN.ipynb
```

One might train the model and check the Encrypted evaluation of FCNN-EncFCNN by simply call this command.

```
$ python3 FCNN_EncFCNN.py
# or just use notebook FCNN_EncFCNN.ipynb
```

The program generates the training loss, testing accuracy, overall training time, testing time, and time used in propagating through each layer when evaluate on encrypted data. All results are ready for check in the notebooks. A copy of our CNN model is also included, the file name is mnist_cnn.pth for avoid repeat training.