Assignment 1

Assignment due: **02/15/2023 11:59 PM EST**

Late submission due: 02/22/2023 11:59 PM EST with 10% grade penalty

Submission after 02/22/2023 11:59 PM EST will not be accepted

Submission format: please submit your codes for Task 1 & 3 and a report in .pdf for all tasks. Do not include the dataset in your code submission.

Task 1 : MNIST Classification - 40%

After taking the deep learning practice class, please complete the **training**, **evaluation**, and **inference** for MNIST classification. Write a README.MD file to elaborate on how to set up a virtual environment and run your code. Write down your observations from the experiment and accuracy over the testing set in the report. An ideal accuracy should be above 97%. Some additional requirements:

1. Manage all the configurations properly in your codes, such as training epochs, batch size, learning rate, etc. Consider using the *argparse* package;

2. Split python files reasonably. A desirable project structure is not putting all code in a single file. Split your codes of model architecture, training, evaluation, inference, and some utility functions into different files so you can locate them easily;

3. Build the API in your codes as simple as possible. Try to disentangle different modules. Attach necessary comments in your codes.

Task 2 : Paper Reading - 30%

ResNet is a famous model in computer vision and deep learning. It has demonstrated a great capacity in image classification, object detection, and other vision tasks. Read the paper Deep Residual Learning for Image Recognition and answer the following questions:

1. How many types of ResNet have been proposed in the paper?

2. What is the fundamental difference between ResNet and VGG?

3. What is the input size of the images for ResNet-50?

Look into the implementation of ResNet in *PyTorch*. Try to call ResNet from the package *torchvision* in *PyTorch*. You can print the network out for its detailed architecture.

Task 3 : Deepface Exploration - 30%

Deepface is a lightweight face recognition and facial attribute analysis framework in Python. It is a hybrid face recognition framework wrapping state-of-the-art models. Install *Deepface* and try the following functions in your codes with your self-collected facial image data:

1. Face verification. This function verifies face pairs as the same person or different persons.

2. Facial attribute analysis. This function estimates the age, gender, facial expression, and race from an input facial image.

3. Face detection. This function detects the face from an image and returns the corresponding coordinates.

Write down your observations from the above functions in your report. Submit your codes with *Deepface* and your findings.