Lab 7 – Introduction to PyTorch and Deep Learning

Prior to commencing with the tasks, ensure that you:

1. Synchronise the time on the Pi, either by connecting to a different network, i.e. a Hotspot, or by using: `sudo date -s 'YYYY-MM-DD hh:mm:ss`
2. Clone the Lab7 Repository from GitHub Classroom **into the ee347 folder**, using this link. Open the ee347 folder in VSCode and work from there. **Do not move the .venv folder.**
3. Commit and Push to GitHub after each task. Each task should be completed in the individual taskX.py scripts provided. Ensure model/data files are included in your .gitignore or you may have difficulties pushing to GitHub.

Tasks

1. Edit task1.py to replicate the output shown below:

```
Tensor before manipulation
tensor([[[[0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 0, 0]]]], dtype=torch.int32)
Tensor after manipulation
tensor([[[[1, 1, 1, 1, 1, 1, 1, 1],
          [1, 0, 0, 0, 0, 0, 0, 1],
          [1, 0, 0, 0, 0, 0, 0, 1],
          [1, 0, 0, 0, 0, 0, 0, 1],
          [1, 0, 0, 0, 0, 0, 0, 1],
          [1, 0, 0, 0, 0, 0, 0, 1],
          [1, 0, 0, 0, 0, 0, 0, 1],
          [1, 1, 1, 1, 1, 1, 1, 1]]]], dtype=torch.int32)
```

2. Multiply the two tensors provided in task2.py together. Print the output to the terminal.
3. Reshape the tensor provided in task3.py to the shape: (3, 2). Print the output to the terminal.
4. Use OpenCV to view random sample images from the MNIST dataset. The code to download and load the dataset is provided in task4.py
5. Run the train script provided in task5.py. Adjust the batch size to minimise training time.
6. Adjust the training script to save the model when training is completed, so that training can continue from the same point, or the model can be deployed. See the **PyTorch Documentation**.
7. Adjust the training script to allow for multiple epoch training.

8. Adjust the training script to save the ‘best’ model, along with the current model. The best model should be the model with the **lowest train loss** and the current model should be the model at the end of the last complete epoch.
9. After each epoch, save a matplotlib graph of the test and train loss across all epochs as an image. Overwrite the image at the end of each epoch to simulate a ‘live’ graph. Train for 3 epochs and save the graph output.
10. Deploy your model to a simple application. Your application should load random samples from the MNIST testset, show them to the user, along with your model’s prediction. Your application should also show the ground truth value of the sample, along with indicating if the prediction was correct or not.