Project 5

The goal of this project is to gain some familiarity with the two most popular libraries for deep learning: TensorFlow and PyTorch. You'll begin by working through tutorials for both libraries, and then you'll apply those tools to classify larger sets of handwritten characters in English and Japanese.

Partners and Groups

You are required to work with your assigned partner on this project. You are responsible for ensuring that you and your partner both fully understand all aspects of your submission; you are expected to work collaboratively and take responsibility for each other's learning.

You will also be assigned a code-review partner after the deadline. To prepare for that code review, make sure you understand all of your code and that it is cleaned up to be readable by people outside your group. See the code review guidelines for more information.

TensorFlow and PyTorch Tutorials

TensorFlow and PyTorch are by far the two most widely-used libraries for deep learning. Both platforms are extremely powerful and feature-rich, and in general it's possible to do just about anything we might want using either library. However, TensorFlow and PyTorch have different philosophies on how to set up deep learning models that result in different strengths and weaknesses for various tasks. And so to give ourselves the widest range of options, it's worth making sure we're familiar with both of these tools.

To get up-to-speed on the basics, we'll start by following along with the standard tutorials for both libraries. In the starting-point code, you've been provided with two notebooks called `tensorflow_tutorial.ipynb` and `pytorch_tutorial.ipynb`, which you should use to follow along with these tutorials:

- TensorFlow: tensorflow.org/tutorials/keras/classification
- PyTorch: pytorch.org/tutorials/beginner/basics/intro.html (start with 1: Tensors)

Both of these tutorials build a model to classify the "fashion MNIST" dataset. Follow along with the tutorial, and as you go, make sure to try out variations on the code they provide to make sure you understand what it's doing. In principle, both tutorials come with notebooks you can download, but I recommend copying over the code blocks to our own notebooks individually to help ensure that you're thinking through each step of the process.

EMNIST and KMNIST

Once you've completed both tutorials, your main task for this assignment is to build models to classify the following data sets:

- EMNIST is an extension of the MNIST dataset to include alphabetic characters.
- KMNIST is a similar dataset based on Japanese Kuzushiji characters.

Both datasets can be downloaded from the links above, but to avoid massive duplication, I've downloaded these data sets for you to my public folder. If you run the terminal command

```
ls /home/DAVIDSON/brwiedenbeck/public
```

you should see sub-folders for `emnist` and `kmnist`. For each of these data sets, I recommend starting with the medium-difficulty version: respectively `EMNIST-Balanced` and `Kuzushiji-49`, and then later move on to the harder versions with more labels and unbalanced classes: `EMNIST-ByClass` and `Kuzushiji-Kanji`. See the respective websites for more information.

Your task is to build the best model you can for each of these problems, but you should tackle one in PyTorch and the other in TensorFlow. Which problem you solve using which library is up to you. Of note, when using the Launcher to create Jupyter notebooks, you have the environment options `TensorFlow`, `TensorFlow + GPU`, `Torch`, and `Torch + GPU`. The starting-point notebooks called `tensorflow_scratchwork.ipynb` and `pytorch_scratchwork.ipynb` have been created with the CPU versions of each library, and I recommend starting with these, but you are welcome to also create GPU notebooks when you get to the point of running experiments.

To get started, your first task is to load and transform the one of the data sets. This will include *at least* reading the appropriate file/directory format, reshaping inputs and re-encoding outputs appropriately for your neural network, normalizing any inputs that aren't properly scaled, and splitting off a validation set.

In general, your approach to training should be to play around with architectures and hyper-parameters to get *something* working, then figure out the limits within which your training mostly works, and then optimize within those ranges. For these problems, a great starting point to get something working is probably the same model sizes and settings you saw in the tutorial(s). Feel free to copy over code from the corresponding tutorial notebook to get started. But then you will need to explore the space of hyperparameters to see what you can improve. Feel free to try out lots of different options, but remember not to prematurely optimize: first determine whether something makes any significant difference before trying to optimize it.

Once you've optimized one model for EMNIST and one for KMNIST (one of which is in PyTorch and the other in TensorFlow), try tackling one of the harder instances. It's up to you whether to work on `EMNIST-ByClass` or `Kuzushiji-Kanji` and which library to use. Since both of these problems have imbalanced classes, you are strongly encouraged to look at metrics such as the confusion matrix that split out performance by class, rather than just overall accuracy. To achieve good performance on the less-common classes, you may need to look into training methods that over-weight or over-sample rare instances.

Your deliverables will be three Jupyter notebooks that explain and show off your best models, along with data and/or experiments demonstrating what else you tried and how you settled on your chosen architecture and hyperparameters. As with project 2, you should think of these notebooks as written presentations rather than code submissions. Remember that an important part of making a convincing presentation is to show both what worked well and where there's room for improvement.