Internship Interview – Coding Task

Oxford Research Group, FiveAI

February 29, 2020

1 Introduction

The aim of this coding task is to train a neural network on a simple machine learning dataset (namely MNIST), and then observe, explain and (in some cases) improve upon the performance it achieves on a number of modified variants of the MNIST test set. Note that we will be judging this task at least as much on the quality of the arguments you present as on the performance of your models, so please bear that in mind when deciding how to spend your time. You may use any programming language and any machine learning framework you like to tackle this task. Good luck!

2 Model Training

The first step is to load the MNIST training and test sets, and train the model. Some machine learning frameworks have utility functions that make loading MNIST easy – please feel free to use those if it makes your life easier. If your chosen framework is less helpful in this regard, you can instead download MNIST manually from the following website: `http://yann.lecun.com/exdb/mnist`.

The model you train should have eight fully-connected layers. The first, second and third hidden layers should have 500 units, the fourth and fifth layers should have 200 units, the sixth should have 100, the seventh 50, and finally, the eighth layer should have 10 units. There should be a ReLU layer after each fully-connected layer except the last one.

Task: Write code to define and train this model on MNIST. You should use cross-entropy for your loss function, but you are free to choose your own optimiser, learning rates and other training parameters. Train the model for (at most) 20 epochs. Verify that your trained model achieves at least 95% accuracy on the MNIST test set.

3 Model Analysis

Having trained the model, we now want to explore how it performs on some modified variants of the MNIST test set. We've provided these variants to you as simple `.npy` files, to make it easy to load them in using NumPy. (If you're using a language other than Python, please let us know, and we can provide you with the same data in `.csv` format.) To make it easier to switch between the clean test set and these variants, we have also included a file called `clean.npy` in the `.zip` file that contains the clean test set in the same format as the variants.

There are four variants in total, named `t1`, `t2`, `t3` and `t4`. Sample code to load in a variant can be found in the `.zip` file.

Task: Evaluate the model on each of the test set variants, and answer the following questions for each variant:

1. How has its accuracy changed compared to that on the normal test set?
2. If the accuracy has changed, how would you explain this change? If it hasn't, why not?
3. If the accuracy has decreased and it can be easily fixed, please demonstrate how (making limited changes to the model architecture and retraining is one possible way of doing this, but please avoid huge changes). If it can't be easily fixed, please explain why not.

Question: Why is there a qualitative difference between the way in which the model performs on `t1` and the way in which it performs on `t2`? Similarly, why is there a qualitative difference between the way in which it performs on `t3` and the way in which it performs on `t4`?

4 Model Generalisation

Task: Given everything you know, now train a single model that performs reasonably well on all variants. You may assume that at test time you know which image comes from which variant.