**Weekly report**

Our model still doesn’t work, we spent the week unsuccessfully debugging the network but have ruled out many factors.

**A small dataset**

Partitioned our dataset of Chinese characters into a dataset of 10 characters and a dataset of 2 characters. The cross entropy was still reducing to ln(1/number of classes), meaning that the output was just a uniform distribution.

MNIST has ~60,000 training samples, meaning 6000 samples per character whereas we were using 100 per character. We incorporated more .gnt files to increase our datasets to 300 per character, with no success in accuracy.

**Data flow debug**

We considered that our data flow may be implemented incorrectly. We looked up the correct flow (shuffle the data, then batch it, then repeat it) and implemented this.

Then, we considered that maybe shuffling the data was causing the images and labels to not be matched up properly. We ruled this out by checking through images and labels.

We checked the tensorflow one hot vector implementation, but that works fine too.

Then we checked how the iterator works in tensorflow, in case we were skipping a batch when we were calling the next batch of images and labels, however by checking images/labels in the batches we found that it was working as intended.

We used two different optimizers (Gradient descent and the Adam optimizer) but neither worked.

Checked the data type we are feeding in, it is the same.

**MNIST database**

We downloaded the MNIST database and converted it to images and labels we could use.

When we input this into our database it initially worked, until we found out that our number of outputs was 2, not 10. In theory, a network is still valid if the number of outputs is less than the unique number of classes up to at least ln(number of classes)/ln(2), but we are not sure if this applies to a number of outputs as low as just 2.

We printed out and checked that the MNIST data we are feeding into our code is the same as in an example code. It is. Therefore there must be something wrong with the code, unless there is something we have overlooked with our data (unlikely).

There is a slight bug we have with the MNIST images that we can’t output them as .png or .jpeg but using an inbuilt function with the MNIST library outputs them in the command line and shows they are not corrupted, so we are not concerned about being unable to output them as images.

**Overfitting**

We tried overfitting: 12 samples of MNIST character with just 2 classes. This seems to get to some accuracy and then just settle such as 83% accuracy or 67% accuracy. Perhaps the learning rate is set too high here. Increasing sample size (with or without increasing batch size accordingly) leads to 0 accuracy once again.

**Further debug**

We have looked into what we expect for different learning rates, as well as basic checks on a neural network, and we have seemed to try most things.

There are a few things that we could add to improve our network, but these are extra things not things that could fundamentally leave us with 0 accuracy: stochastic gradient descent (shuffling batches), decaying learning rate, drop out layers.

We are at quite a dead end and not sure where to progress from here.

**Action points for the next week**

1. Rebuild the network from scratch, now that we know more about how tensorflow works.

2. With the rebuilt network, ensure that just the MNIST example works, with our saved MNIST data (rather than the MNIST class used in the tensorflow example).

3. From there, we can implement our Chinese character dataset.