**Deep Learning – Assignment 1**

**Part 1:**

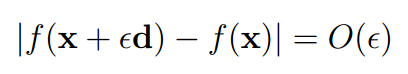
1. We created functions to compute the loss “soft-max regression” and its gradient with respect to and the biases ().

We then tested our implementation using a gradient test:

A comparison of a graph

Description automatically generated with medium confidence

The blue plot implements the formula:



While the orange plot implements the formula:

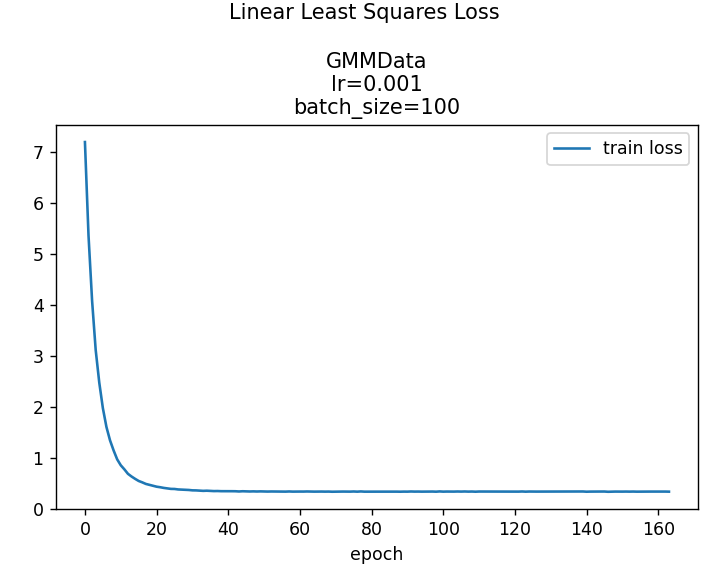


We can see that the test was successful since the blue plot behaves like while the orange plot behaves like .

For example, the blue plot starts at about while the orange plot starts at about .

1. We created functions for minimizing an objective function using SGD.

We then verified that our optimizer works on a small least squares example:



As can be seen, the loss function decreases as the number of epochs increases, just as we expected. In addition, as the number of epochs goes to infinity, the loss function tends to roughly 0.35.

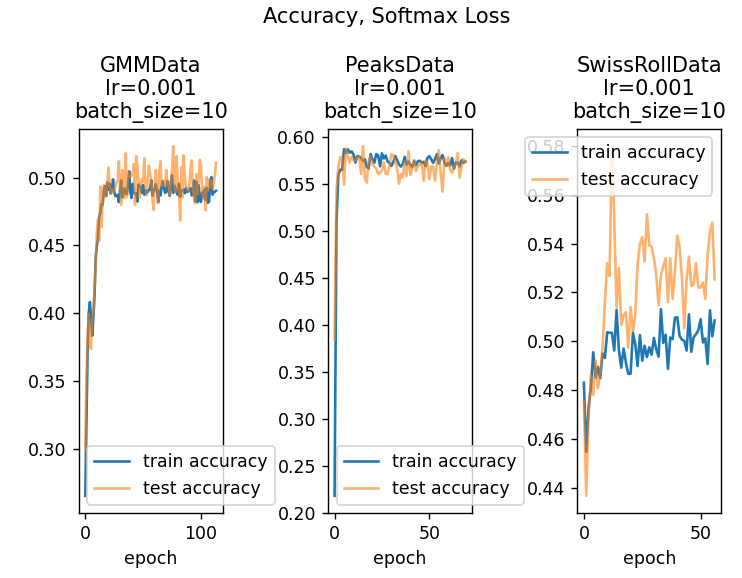
1. We ran our SGD minimization algorithm using the softmax function.

During the run, after each epoch, we checked the percentages of data classification – for both the training data and the test data.

We made sure to compute that metric only on random subsamples of the datasets.

We then tried multiple combinations of learning rates (0.01, 0.001, 0.0001) and batch sizes (10, 100, 1000).

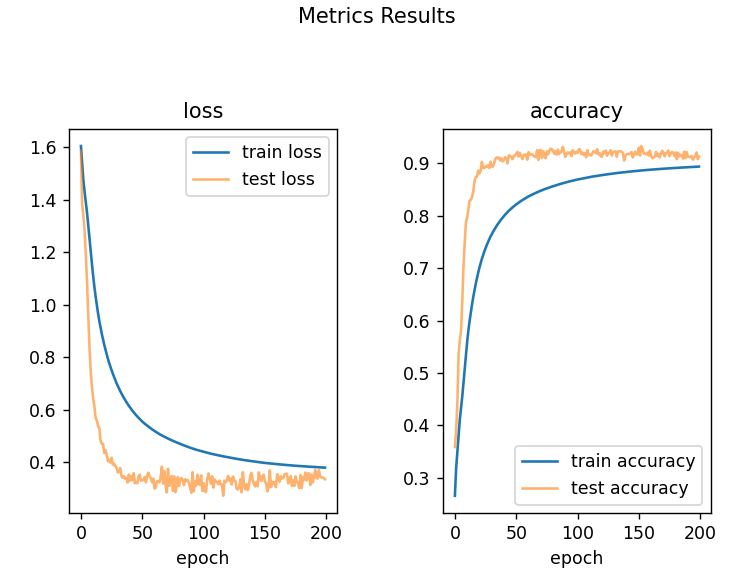
Here are the graphs of the best hyper-parameters that we’ve got:



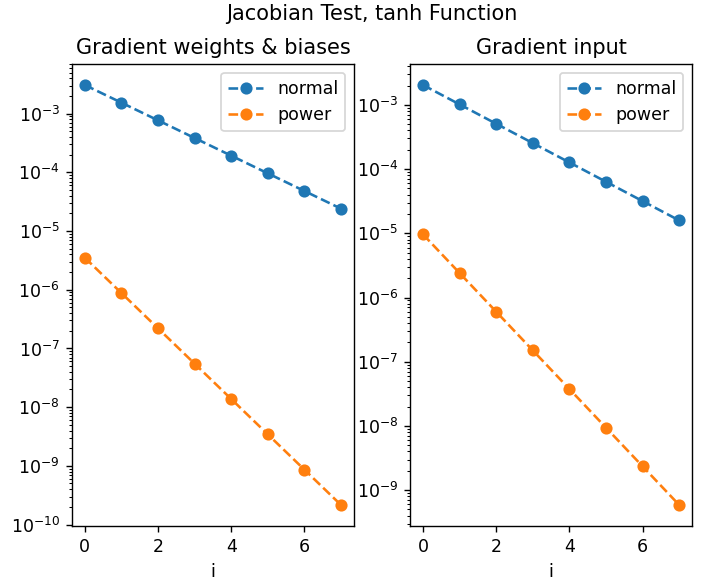
**Part 2:**

1. We wrote the code for the standard neural network, using tanh() as the activation function.

Here you can see the results of training a network with one inner layer:



We then made sure that our implementation is correct using a Jacobian test:

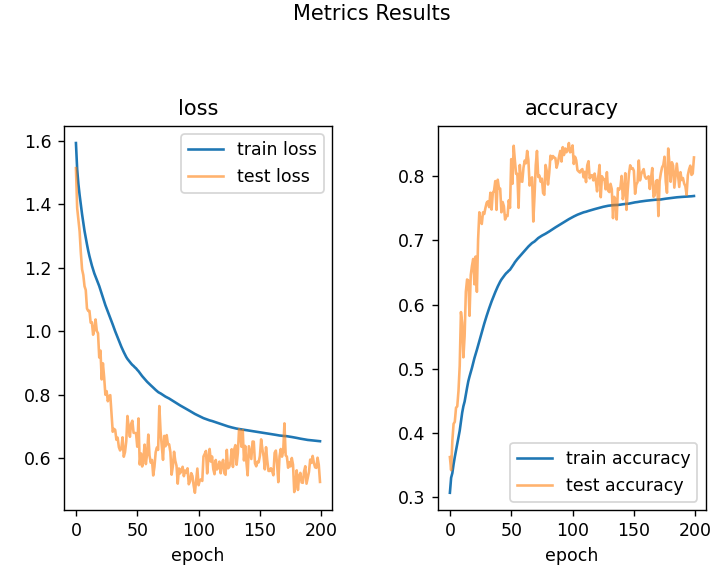


In addition, we performed a Jacobian transposed test and made sure that the result is near zero:

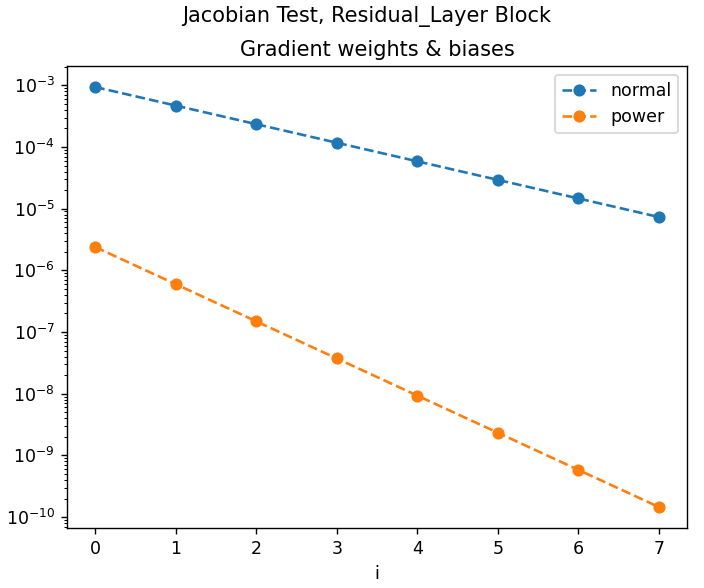


1. We then extended our network and added residual layers.

Here are the results of training a network with regular layer (tanh), residual layer (tanh) and softmax layer:



We then made sure that our implementation is correct using a Jacobian test:



In addition, we performed a Jacobian transposed test and made sure that the result is near zero:



1. After we’ve verified that all the individual parts are OK, we checked the gradient test of the whole network (layer, residual layer and softmax):

A graph of different sizes and colors

Description automatically generated with medium confidence

1. We ran our SGD minimization algorithm using the entire neural network.

During the run, after each epoch, we checked the percentages of data classification – for both the training data and the test data.

We made sure to compute that metric only on random subsamples of the datasets.

We then tried multiple combinations of learning rates (0.01, 0.001, 0.0001), batch sizes (10, 100, 1000), network depth (1,2,3) and layer types (tanh layer and residual layer).

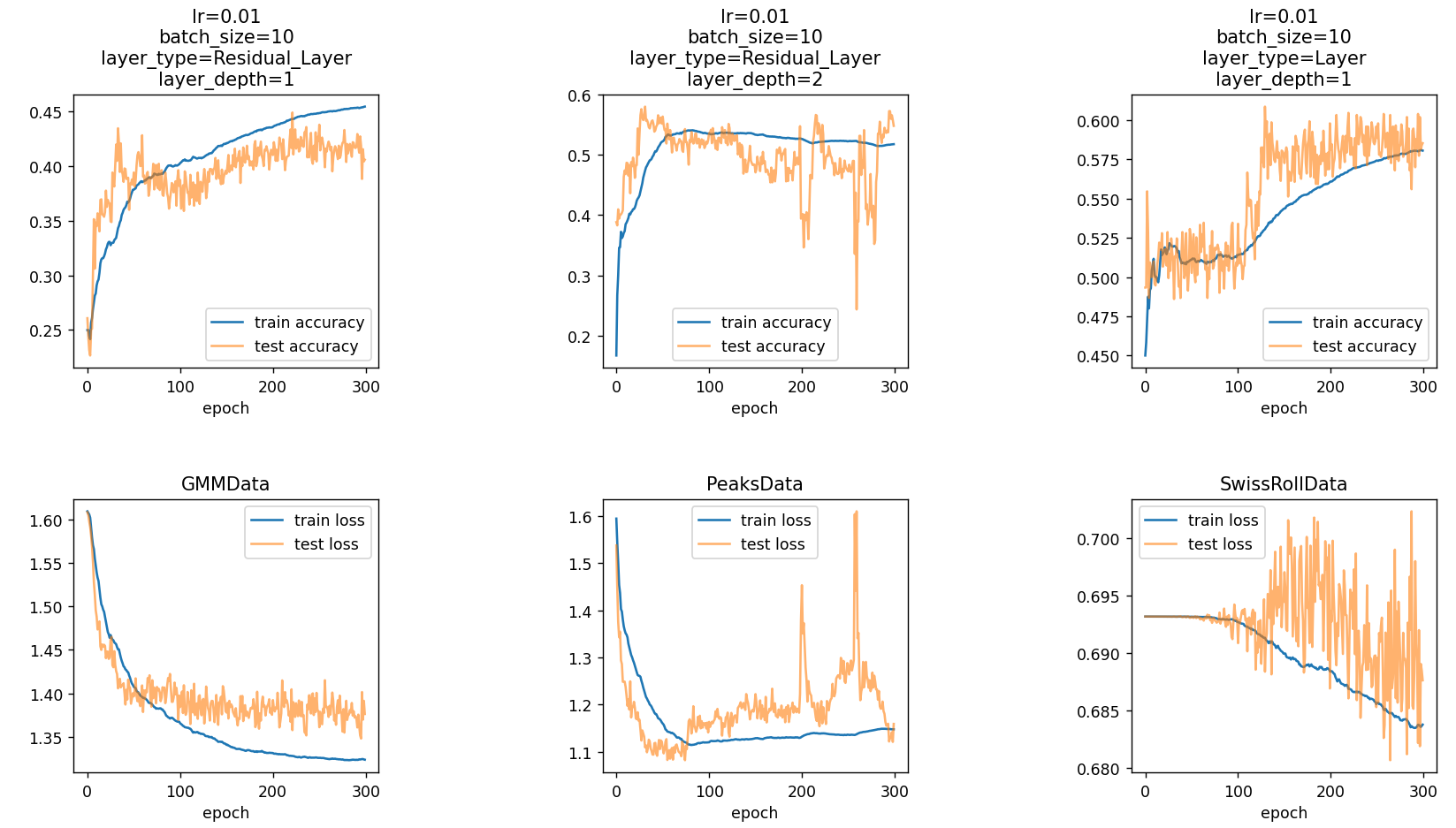
Here are the graphs of the best hyper-parameters that we’ve got for each data file:

A group of graphs with numbers

Description automatically generated

We can see that indeed a smaller batch size results that the network will have better performance. In addition, it managed to learn better with more simple architecture (lower depth), but maybe it can change if we would train the networks using more epochs.

1. Finally we repeated the previous section, only now we used 200 data points for training (sampled randomly).

Here are the new results:

As we can see, the test accuracy of the new networks are much lower now – for GMMData it decreased from around 0.9 to roughly 0.4, for PeaksData it decreased from around 0.85 to roughly 0.55, and for SwissRollData it decreased from around 0.85 to roughly 0.6.

In addition, the test loss of the new networks are also much higher now – for GMMData it increased from around 0.3 to roughly 1.4, for PeaksData it increased from around 0.6 to roughly 1.1, and for SwissRollData it increased from around 0.3 to roughly 0.7.

This is a result that we’ve expected for – since it trains on considerably less data, it makes sense that it will learn much less about the characteristics of the whole data and struggle with generalizing its behavior for examples that it hasn’t seen yet.