**IN 3063: Programming and Mathematics for AI**

**Tasks 3 & 4 Report**

**By Amir Harvey & Krzysztof Szymanski**

**Task 3 - Fashion-MNIST dataset**

In Task 3 we were asked to:

* Implement sigmoid and Relu layers (with forward and backward pass) (10%)
* Implement a softmax output layer (10%)
* Implement a fully parameterizable neural network (number and types of layers, number of units can be changed) (20%)
* Implement an optimizer (e.g. SGD or Adam) and a stopping criterion of your choosing (10%)
* Train your Neural Network using backpropagation (mainly using numpy) (20%)
* Evaluate different neural network architectures/parameters, present and discuss your results (30%)

A picture containing table

Description automatically generatedTo do this, first we needed to import all the libraries, and then the flattened csv image data from the Fashion-MNIST dataset.

Following this we converted the clothes categories into one-hot encoded vectors to prepare it for the training and testing, then we were able to successfully create the training and testing set and begin building our Neural Network.

After this we Initialised the Weight Matrices such as Sigmoid, Relu and Softmax for our forwards and backwards passes.

Then it was time to train our neural network using Backpropagation, this was done using Cross Entropy, to understand how the results we got deviate from the actual results. This is also where we implemented our forward pass, as well as defining our Derivatives.

Using past gradients, we were then able to calculate the present gradients of the data, which helped us calculate the accuracy of out neural network.

After this it was time to print out our accuracy in out training and testing dataset, where we found the accuracy improves very quickly within the first 10 tests, then stays at the 0.6 level the rest of the time, after this we generated a graph to graphically represent that data, which further shows this.

One of the things we did to meet the brief was implement the Sigmoid and Relu layers to go forwards and backwards, this is essential when dealing with a model where the output is predicting a probability, and this has been implemented and is working well.

Another thing we did to meet the brief was implement a softmax layer, this is the activation function, that is in the output layer of a neural network, and it predicts a multinomial probability distribution, which is exactly what we needed to do.

Another thing we did to meet the brief was implement a neural network, and although it wasn’t fully parameterizable, it was still able to predict probability, even though the number and types of layers as well as number of units can’t be changed.

Text

Description automatically generatedAnother thing we did to meet the brief was Train our Neural Network using backpropagation, this means the network trains by learning from its errors in the previous tests and this is one of the most efficient ways to work out accuracy as it learns from it mistakes and can insure it can do better, for example after around 10 tests, out average accuracy goes from around 10% to around 60% as seen here.

One way we improve the code would be to add some form of data visualisation to show the rapid increase in accuracy as more tests are done.

Another thing we tried to do, to meet the brief was using Adam to optimise our code, this is helpful because Adam combines the best features from the AdaGrad and RMSProp algorithms to give an optimization algorithm that can easily handle a wide range of gradients, like in our case. Although we couldn’t successfully implement this, we still realise its importance and tried out best to.

In this task we refenced code to help us with the neural network training using backpropagation, as well as converting the clothes categories to one-hot encoded vectors at the beginning, we believed this was the right choice to make as we had a desire to make our code as efficient as possible, and doing things like using one-hot seemed like the most efficient way to prepare the data as the pro of using one-hot is that you get a better prediction at the end of your algorithm because your data is better prepared. And as mentioned above using a neural network with backpropagation was also essential to ensure our data could have the most efficient testing accuracy.

In Conclusion the result of our code tells us that the implementation of our code was a success as the accuracy does improve as more tests are run, as it learns how to be more efficient in estimation the Fashion-MNIST.

**Task 4 - CIFAR-10 dataset**

In Task 4 we were asked to:

* Implement a neural network (using PyTorch) (20%)
* Propose improvements (e.g. Convolutional Neural Network, dropout, etc) (30%)
* Evaluate different parameters (20%)
* Present and discuss the results in the report (30%)

To do this we were required to use the CIFAR-10 dataset which consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images to create a large dataset for our neural network to work with, and a good test for our network, and a good test of our understanding of neural networks and implementation.

To do this, first we loaded the CIFAR-10 dataset using torchvision, then we had to convert the default PLI images to Tensor images.

Text

Description automatically generatedThis is when we made our Neural Network and thinking ahead to the Improvements part of the assignment, made it into a Convolutional Neural Network by changing it from taking 1 channel image to 3.

We then had to define our loss function and optimiser, this was archived by using a Classification Cross-Entropy loss, as well as SGD using momentum, which helped optimise our code.

After this we proceeded to train the network by looping over our data iterator and feeding the inputs into the network and optimise it, then we saved our newly trained network.

A picture containing text, screenshot, different, several

Description automatically generated

Once this was done, we could begin testing our network on out test data and see if two passes over the dataset is sufficient to increase its accuracy. This is done by predicting the class label of the images the network outputs and comparing it to the real output.

The initial test results were promising so we proceeded to test the whole dataset, which produced a 50-55% accuracy which is much better than the 10% it would have gotten if it randomly picks out of the 10 class labels. As you can see in this example the. neural network successfully detected 2 of the images (the cat at the beginning, and the plane at the end) but it also successy predicted that the two middle images would be 2 of the same image which is very impressive.

Another improvement we made as mentioned above was the implementation of the Convolutional Neural Network over a regular Neural Network, this was done because Convolutional Neural Network are specially designed for reading pixel values from Images and learn from it, which was perfect for our dataset choice, as our dataset is largely based on pixelated images.

We also evaluated to the different parameters by doing out testing in the code to see what the output would be, and this was another requirement that we have achieved as this has. Been explained earlier in our report.

In the future we could add even more features such as visual predictions so user can see exactly what the neural network was thinking without having to read any text, or in we could create a Convolutional Neural Network that could predict down to the individual picture, although this would be hard it would be possible to implement using our code as a good base.

In conclusion the result of our code tells us that our implementation of the Neural Network was a success as it has around 5 times more accuracy than a normal program would have when it comes to predicting images from the CIFAR-10 dataset, this means that the learning algorithm works, and we were able to do it using PyTorch as we were requested to, and. I believe with more training we. Could have made and testing we could have built an even more intelligent neural network, but it would have taken a long time to go execute the training code and would have used up a lot of RAM to do so. This is why we think. Our code is impressive because it is the perfect balance of Performance and Practicality.

**Reflection**

In these tasks we both contributed evenly, and work well together to produce, debug, and execute our neural network, as well as working well together on the repot. In terms of personal contributions for each task, Amir had a leading role in Task 4 and Kris had a leading role in task 3 however we both massive contributed on each, and Task 2 was a joint effort with Amir researching Outliers and Kris researching Nested Models.

**References**

Task 3 (50%) - Training using back propagation and converting the clothes categories to one-hot encoded vectors - <https://medium.com/swlh/estimating-the-fashion-mnist-using-simple-neural-network-fa867918c227>