Daniel Olshvang – 205842750
Omer Gottlieb - 302184064

Machine Learning – HW4

Part 1

Task 1:

Our results are as follows:
Loss: 0.82 . Accuracy: 64.57% .

Task 2:

We changed the activation function from ReLU to Tanh.
The purpose of the activation function is to enable the model to learn non-linear relationships in the data, by determining the output of the layer through the activation function.
ReLU is the simplest activation function, since if x>0 then the output is x, otherwise it is 0, so it requires very little computational work – making it faster to process.
However – ReLU has some limitations – negative values which are neglected can be of use, and in this case, they are all regarded as 0. In addition, the output of the function is (0,∞), unlike the output of Tanh or sigmoid which are finite, thus it can make the classification more challenging.
By changing the activation function to Tanh – this will probably result in longer computational time, but we might hit a better job in the classification – but this ofcourse depends on the data and the model.

Task 3:

By increasing the number of epochs, we increase the computational time on account of improving the performance of our model (on account that we didn't 'converge' and reach the ceiling of our model)
Our results are as follows:
For 25 Epochs: Loss: 0.83 . Accuracy: 65.71% .
For 40 Epochs: Loss: 0.85 . Accuracy: 66.86% .
We can see that the accuracy has improved, but not by that much, suggesting we are reaching the ceiling of our model, meaning – the results will be "as good as they get" while using this model.

Task 4:

Mini-batch gradient descent is basically a 'compromise' between Batch gradient descent and Stochastic gradient descent. Batch gradient descent involves the whole data set for each weight update, which takes a very long computational time for large data sets. SGD does a weight update for each training example. Mini-batch is updating the weights for every n training example, for example – 32.
The advantage of using the mini-batch gradient descent is that it does reach the local minimum (with very little fluctuations when comparing to SGD), unlike SGD which dances around the minimum since the weights are updated after every example.

Daniel Olshvang – 205842750
Omer Gottlieb - 302184064

Task 5:

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

Our results are as follows:
Without Batch Normalization: Loss: 0.83 . Accuracy: 68.57% .
With Batch Normalization: Loss: 0.9 . Accuracy: 66.86% .
We can see that the Batch normalization didn't help improve our results. This is because the quality of the data limits our classifications, and adding more layers and changes to our neural network might even reduce the quality of the model.

Part 2

Task 1:

How many layers does it have?

The Neural Network has a total of 8 layers (there are 24 layers if you count all the layers that do any processing operation, but they can be accounted to be part of the same hidden layer, leaving a total of 8 hidden layers), which consist of - 5 convolution 2D layers and 3 fully connected layers.

How many filters in each layer?

The filters are part of the Conv2D layers, and consist of:
(64, 128, 128, 256, 256) filters.

Would the number of parameters be similar to a fully connected NN?

No. The number of parameters in a fully connected NN would be much higher.
One of the points of the convolution layers is to reduce the number of parameters, because in fully connected NN we literally have "full connection" between the layers – meaning every neuron is fully connected to all neurons in the next layer.

Is this specific NN performing regularization?

Yes. We can see that every 2D convolution layer has a kernel regularization – L2.

Task 2:

Our results are as follows:
For the number of filters of (64,128,128,256,256): Loss: 8.2 . Accuracy: 28% .
For the number of filters of (32,64,64,128,128): Loss: 5.15 . Accuracy: 33% .
These results might indicate that we overfit the model by using a large number of convolutional filters, and by reducing their number we achieved better results – by generalizing the data.