HW4

Sagi Ben Itzhak and Shir Ricon

Part 1

Task 2

The activation function is responsible for transforming the sum of the weighted inputs of the node into an output value called the activation of the node, which can than be inserted to the next layer of the model. There are quite a few functions which can be used in neural networks for this purpose, among them are linear functions, sigmoid function, tanh function, and ReLu. Each function yields different node outputs which can affect the model and its' performance. In this task we decided to compare the ReLu function used in the previous task, and the sigmoid function. The ReLu function is a function that returns 0 for input values (the sum of the weighted inputs in this case) lower than 0, and returns the input itself for input values that are greater or equal to 0. The sigmoid activation function is an "S" shaped function that transforms the sum of the weighted inputs into a value between 0 and 1. Because this function is asymptotic, the sensitivity to changes in it's value is high around the mid-point (at x=0 it this case) but decreases as the absolute value of the input is higher. This results in saturation (very high inputs are all transforms to a value close to 1, while very low negative inputs are all transformed to a value close to 0) which can harm the learning process of the model and prevent the improvement of its' performance. This outcome does not occur for positive inputs in ReLu since it is linear in that range.

Task 3

Epoch is defined as a state in which our model has "seen" all our training set. In general, when we increase the number of epochs, our training process of the model will be longer which will result in better performance on our training set. As to the performance on the test set, it will get better with the increase of the number of epochs up to a certain point. At this point the performance of the test set can start decreasing as we increase the number of epochs. This outcome occurs due to overfitting (of the model to the training set). We can estimate the "sweet spot" of the number of epochs that will yield us the best performance by using a validation set in our training process and setting the number of epochs as a hyperparameter.

Task 4 – Mini-btaches

The main advantages of mini-batch over SGD:

- Using mini-batch is less computationally expensive compared to SGD since it updates the model after each batch of examples, as opposed to SGD which updates after each example in the dataset. As a result, using SGD takes significantly longer to train on large datasets.

- When using SGD, the frequent updates can lead to a noisy gradient signal which can result in a high variance of the model error between epochs (and cause the updated model parameters to jump around and not descend to the minima in a "smooth" manner). In some cases, this can make it hard for the model to settle on the minima. When using mini-batch this outcome does not usually occur since the update after a whole batch leads to a smoother descend to the minima.

Task 4 – Batch normalization

Using batch normalization slightly increased the performance of the model. This improvement occurred because the batch normalization standardizes the inputs to a layer for each mini-batch. By doing this, we are stabilizing the learning process of the model by reducing the change of the distribution of the input layers in the network, which in turn can reduce the number of training epochs needed.

Part 2

Task 1

- The model consists of 8 layers. 5 Convolutional layers and 3 fully connected layers.
- Number of filters in each convolutional layer:

Convolutional layer:	First	Second	Third	Forth	Fifth
# of filters	64	128	128	256	256

- When comparing the current network to an 8 layer fully connected network, the number of parameters of the later one will most likely be higher than the current network. This is because fully connected layers typically require a number of parameters that is larger compared to convolutional layers.
- Yes, there are l2 regularization on the weights in each convolutional layer. In addition, there is an optional "drop out" in all 8 layers which can be considered as regulation operation as well.

Task 2

Since the performances of both models on the training set were extremely low and the accuracy achieved for both models was similar, we trained the models on the training set numerous times. In all cases the accuracy achieved for both models was around 0.25 (in the range of 0.2-0.3), which in this case, where the distribution of the labels in the training set is equal for all 4 labels (~1620 examples for each label), means that the models performed similar to a naïve classifier. In addition, we could not recognize a preferred model since in some of the runs the initial model performed better, and in others the later one performed better. We are not completely sure about the reason for this outcome. We will state two possible explanations:

- The architecture of the networks is not suitable for this dataset, leading to an overall bad performance for both models and to a minor unrecognizable difference in performance when changing the number of filters between the models.
- The quality of the data (the images) is low, resulting in bad performance for both models.

References:

https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/