HW4 –

Iyar Hadad – 206172793

Carmit Mordechai - 315822577

PART 1

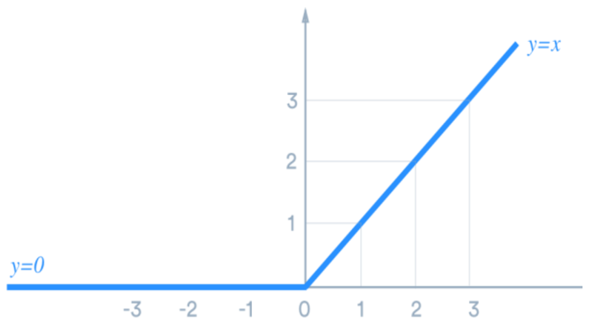
Task 1: NN with fully connected layers

Code.

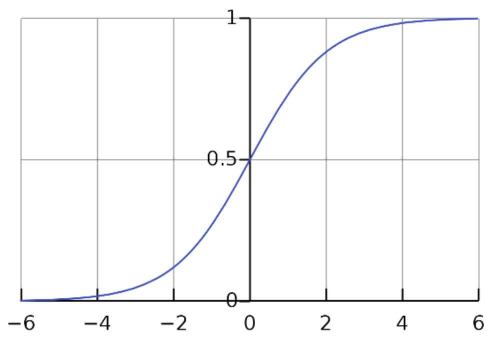
Task 2: Activation functions

Explanation of how changing of the activation function from Relu to Sigmoid affects the model.

ReLU function



Sigmoid function



By changing the activation functions from ReLU to sigmoid, we get two differences:

1. Values lower than zero will get a probability of classification, but less than 50%.
2. Values higher than zero will get a probability of classification between 50-100%, but the classification is not linear at this interval. Therefore, the difference is significant in small values, but in high values there is minor influence, the function tends rapidly to zero or to one.

Advantages:

1. Sigmoid is not a divergent function activation, it converges to finite values, whereas ReLU (in ReLU the derivative is constant so there is no mechanism that constrains the output of the neuron)

2. Sigmoid considers values below zero. If too many activations get below zero then most of the units(neurons) in network with ReLU will simply output zero, in other words, die and thereby prohibiting learning.

Disadvantages:

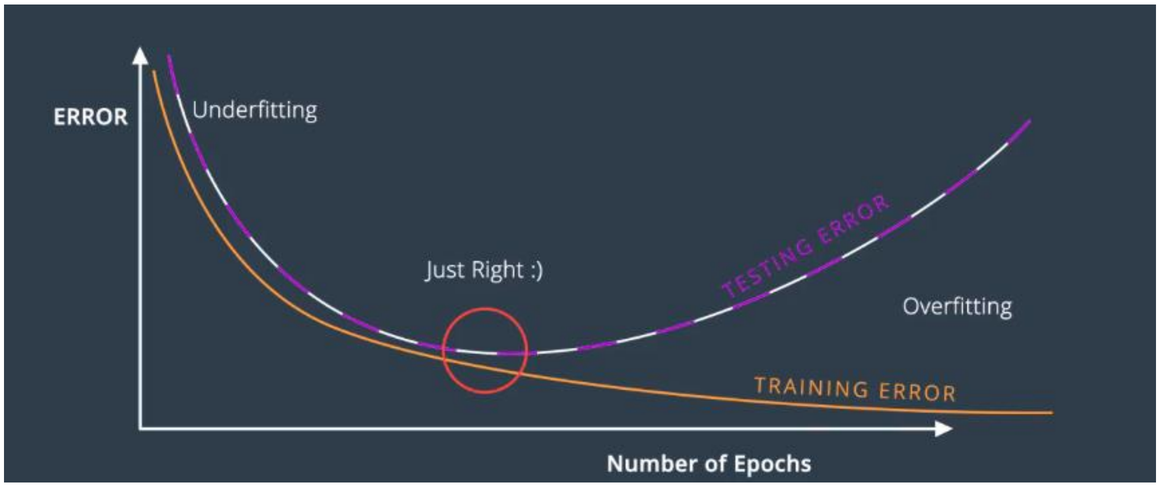
1. The mathematic operation in ReLU is more simple
2. Relu tend to show better convergence performance
3. Relu not vanishing gradient like the sigmoid which its derivative is growing

Task 3: Number of epochs.

What difference does it makes in term of performance between a model using 25 and 40 epochs?

Epoch is one iteration over all the data. There is a tradeoff we need to find in order to choose epochs number.

Small value of epochs could not perform a good learning and enough fitting. On one hand, low number of epochs the model does not have enough data to learn. On the other hand, high number of epochs could be unnecessary, prolongs the running time and even causes overfitting and therefore a bad performance of the model.



We got:

25 epochs-

Test Loss is 1.12

Test Accuracy is 50.86 %

40 epochs-

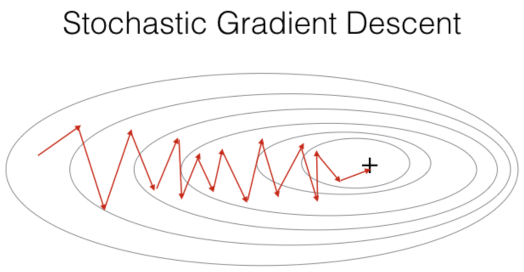
Test Loss is 0.94

Test Accuracy is 62.86 %

We can see 40 epochs performed better than 25, but still we didn't get a good result so maybe increase the epochs number is essential.

Task 4: Mini-batches.
What are the advantages of the mini-batch vs. SGD?*

Stochastic Gradient Descent- SGD means that we update the weights (W) after one example, which equivalent to a mini-batch that contains one batch.



Advantages:

We update the gradient (W) after only one example therefore, SGD converges faster. For instance, if we have dataset of 2 million examples, then just to take one step – an epoch, the model will have to calculate the sum of the gradient of all the 2 million examples.
As much as the dataset is larger, it updates to the parameters more frequently, so on short time it gives us a good aspect of the model performance.

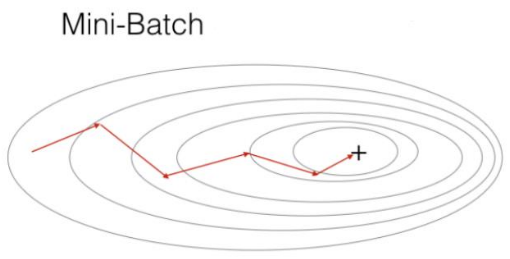
Disadvantages:

Since SGD considers just one example at a time so that the update of the gradient is noisy and will never reach the minima, but it will keep dancing around a local minimum.

In addition, when the process is noisy it causes the model parameters and in turn the model error to increase, so that high variance can cause overfitting.

Moreover, updating the model so frequently is computationally expensive.

Mini-batch- splits the training dataset into small batches and afterwards calculates the gradient. It is a process that finds a threshold between the robustness of stochastic gradient descent and the batch gradient descent.



Advantages:

The update of the gradient using mini-batch reduces the variance and avoid overfitting.

As much as the number of batches is smaller the computational resource per epoch is smaller, so it goes faster. Mini-batch provides a computationally more efficient process than SGD.

Disadvantages:

As much as the number of batches is higher it computationally complicated which means an expensive process of long iterations.

Task 5: Batch normalization.

How does batch normalization impact your results?

batch normalization is a standardization technique which calculates the mean and standard deviation of the input of a layer per mini-batch. It affects the training model by reducing the training steps - number of epochs, which provides faster training.

It reacts like a regularization that reduces the error like an activation function.

In addition, Batch Normalization has a beneficial impact on the updating gradients (W) through the NN, by reducing the dependence of gradients on the scale of the parameters or of their initial values. Consequently, we can increase the learning rates without the risk of divergence.

PART 2

Task 1: 2D CNN.

- How many layers does it have?
  After the input data, there are 5 layers of convolution computation, and 3 dense layers, and each one is attached to processing operations such normalization, activation, pooling, flatten etc. Sum of layers is 8.

- How many filter in each layer?
  From the first convolution layer to the fifth – 64, 128, 128, 256, 256.

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

  No, the number of parameters is much less than fully connected NN, an advantage of CNN over fully connected NN.

- Is this specific NN performing regularization?

Yes, we have two ways that indicate regularization:

1) Using of regularization type L2 which updates the cost function in Machine Learning in general, and specifically in NN it causes the values of weights to decrease.
2) Dropout – Specifically regularization technique in NN. At every iteration, it randomly selects some nodes and removes them from the calculation of the error so that it updates the weights according to different set of nodes and this provides variety set of outputs.

Task 2: Number of filters**.**

Rebuild the function get_net to have as an input argument a list of number of filters in each layer, i.e., for the CNN defined above the input should have been [64, 128, 128, 256, 256]. Now train the model with the number of filters reduced by half. What were the results?

The number of filters in each convolution layer, will affect the dimensions number we will learn on the data. By defining more filters, more weights are being learned and more features get weight in the classification. On the other hand, as the number of filters increases the running time increases as well, so if the results of the model aren't much effective it better stay with smaller number but still effective.

We got:

Fiters- [64, 128, 128, 256, 256].

Result- test loss, test acc: [7.8841415132795065, 0.3257143]

Filters-[32, 64, 64, 128, 128].

Result - test loss, test acc: [4.177604424612863, 0.26285714]

We can see, by reducing filters number we got smaller ACC but anyway in both cases the results aren't good. We prefer the 28% over the 25%, but there are more changes that needed to be done in the model to perform better results. For instance, add more layers or more epochs, different activation functions…