HW4:

Matan Kachel – 205798507
Elinoy Faibish - 205738230

Part 1

Q: Change the activation functions to LeakyRelu or tanh or sigmoid. Name the new model *new_a_model*. Explain how it can affect the model.

A: The activation functions determine how the weighted sum of the input is transformed into an output from a node in a layer of the network. Usually, the activation functions introduce non-linearity into the model and allows the model to represent non-linearity in the data.

Relu is defined as:

$$relu(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

tanh defined as

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Main advantage of Relu over tanh and other activation function is that it less susceptible to the " vanishing gradient" problem. The tanh activation function has the problem of saturation in high and low value. This causes the derivative of the activation to shrink which impairs the learning process.

Q: Train the new model using 25 and 40 epochs. What difference does it makes in term of performance?

A: Our result was very similar between running the model with 25 epochs and running it with 40 epochs. Running model with more epochs can improve model performance, with more epochs the model "sees" the training data more times and can update the weight more time and improve performance. However, it can also cause overfitting when model start memorizing the training data and loss generalization ability. It can be the explanation to our result. Another explanation is that the test data not representative enough of the train data so the model has performance limit unrelated to the number of epoch and other hyperparameters.

Q: Build the *model_relu* again and run it with a batch size of 32 instead of 64. What are the advantages of the mini-batch vs. SGD?

A:

In SGD training we compute the gradient of the cost function for each training example separately. For each training sample the net parameters updated according to the computed gradient. Some main disadvantages of SGD are:
1. more sensitive to noise which might hinder getting to a global minimum.
2. It deals with only a single example at a time, so the advantage of vectorized operations gets lost.
3.It use all the resources for processing a single example at a time, so the frequent updates are computationally expensive.

In contrast mini-BGD: compute the gradient of the cost function for p training examples (with p>1). This has some major advantages over SGD:
1. It is easily fits in the memory and computationally efficient.
2. It has the advantage of vectorized operations
3. If stuck in local minimums, some noisy steps can lead the way out of them
4. It produces stable error gradients and convergence

Q: Build the *new_a_model* again and add batch normalization layers. How does it impact your results?

A:
normalization has the effect of stabilizing the learning process and dramatically reducing the number of training epochs and it also can be used as regularization to avoid overfitting of the model. Because the test data not representative enough of the train data so the model has performance limit unrelated to the batch normalization layers.

Part 2

Q:

- How many layers does it have?
- How many filter in each layer?
- Would the number of parmaters be similar to a fully connected NN?
- Is this specific NN performing regularization?

A:

a. 8 layers: 5 Convolutional, 3 fully connected.

b. There are 5 layers with filters:

- Conv2D_1 we have 64 filters.
- Conv2D_2 we have 128 filters.

- Conv2D_3 we have 128 filters.
- Conv2D_4 we have 256 filters.
- Conv2D_5 we have 256 filters.

c. The number of learned parameters will be lower in the CNN in comparisons to the Fully connected net. This is because the CNN net contains Max poll layer which reduce number of futures in every layer and as a result the number of learned parameters.

d. These net architectures contain two mode of regularization –

- **Kernel regularizer** – apply penalty to net weights in accordance with l2 regularization.
- **Dropout** - randomly sets input units to 0, this forces the net to process the data in number of parallel routs, in order to improve generalization ability of the net.

Q: Rebuild the function *get_net* to have as an input argument a list of number of filters in each layers, 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?

A:
Both the original CNN net and the modified CNN net (with half the number of filter) preform poorly on the given classification problem whit only 35-40% accuracy. To our understanding it is a problem of overfitting with train loss reduces and train accuracy improving with each epoch but without improving result on the test and validation dataset.