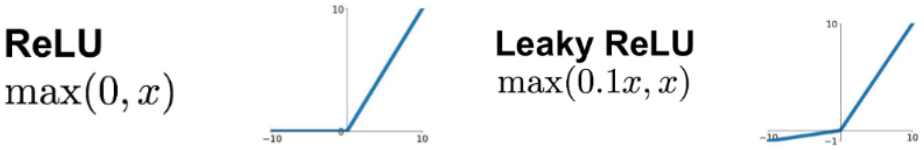
Ameer Lawen 316184456, Shaked Ron 305048142

HW4 – Theoretical Questions

Part 1: Fully connected layers

- ***Change the activation functions to LeakyRelu or tanh or sigmoid. Name the new model new_a_model. Explain how it can affect the model.***
  We chose to use the Leaky ReLU function as our new activation function, let us recall the differences between "standard" ReLU and the leaky one:



  In the ReLU function a positive value gets is linear match and a negative value gets zero, but in the Leaky ReLU one, while in the positive domain the result will be equal, in the negative domain the result is not zero. Leaky ReLU matches a small negative value to each negative input, defined by α (in the example above $\alpha = 0.1$). This setting allows the model to be a little bit more flexible, due to a broader dynamic range of post-activated perceptrons. This characteristic may influence model's convergence, in aspects of time and accuracy.
  In terms of Bias-Variance tradeoff, we can consider **ReLU** function as a forced "dropout" functions that "kills" some of the neurons. This may lead to a less over-fitted training (recall: overfitting means high variance and low bias) because there are less perceptron which contribute to the learning. **In this specific problem set, we get worse results using Leaky ReLU.**

- **Train the new model using 25 and 40 epochs. What difference does it make in term of performance?**
  With 25 epochs, Test loss is 0.786 Test accuracy is 65.7%
  With 40 epochs, Test loss is 0.782 Test accuracy is 66.9%
  There is no major improvement by using 40 epochs vs. 25 epochs as one might assume. These results suggest that the model has gained the majority of its performance after 25 epochs, the extra 15 epochs had no serious contribution to the learning process. This might be due to incompatibility of the model with the dataset, or due to low quality of the dataset.

- ***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?***
  There are lots of advantages of the mini-batch vs. SGD. For instance, mini-batch is less prone to noise due to the averaging of loss-function's gradient, thus is more stable and has a higher probability of converging to the absolute optimum and iterations are faster.

- **Build the `new_a_model` again and add batch normalization layers. How does it impact your results?**
  With batch normalization (50 epochs), Test loss is 0.944 Test accuracy is 67.4%.
  We see no serious improvement upon applying batch normalization, this might be due to the low quality of the images.

Part 2: Convolutional Neural Network (CNN)

- **Have a look at the model below and answer the following:**
  - **How many layers does it have?**
  - **How many filters in each layer?**
  - **Would the number of parameters be similar to a fully connected NN?**
  - **Is this specific NN performing regularization?**

  1) Let us count the layers:
     Input layer, 5 convolutional layers, 2 fully-connected layers and a "softmax" output layer (also fully connected). A total of 7 hidden layers between the input and output layers.
  2) In this first setup, the convolutional layers (5) have 64, 128, 128, 256 and 256 filters respectively.
  3) A CNN model usually uses less parameters than a fully-connected NN, the idea of using fixed-size filters that are being convolved with the data (to extract features) enables a faster learning process with less parameters.
  4) As can be seen in the code, these CNN layers perform an L2 regularization.

- **Now train the model with the number of filters reduced by half. What were the results?**
  With [64,128,128,256,256] filters, Test loss is 8.153 Test accuracy is 26.857%.
  With [32,64,64,128,128] filters, Test loss is 4.97 Test accuracy is 25.714%.

Ameer Lawen 316184456, Shaked Ron 305048142

Concluding Remarks:

We observed many different results while running this part of code again and again, in all of them performance was bad and did not excel chance level. We sought to look for the root of this issue, since all models we have tried during this assignment had not bad performance on the validation set but bad to very bad results on the test sets. We looked manually at the datasets and saw that the quality of the images is low and that there is a major difference in the pre-processing between validation and test sets.
Although the test set has good images in it, the down-sampling process and the use of input as small as 32X32 pixels compromise the results. We therefore suggest to use a larger input, which includes more information (pixels), to the different models, then it would be possible to compare the different approaches of learning.