Machine Learning in Healthcare

HW4- Theoretical Questions

1) 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 changed the model activation function from Relu to sigmoid. Relu is a function that gives 0 for negative inputs and returns the input for positive ones, while the sigmoid function returns values between zero and one. The sigmoid function has a smaller slope for larger values, which makes it harder to classify between closer values, which will have almost the same value. Also, the sigmoid function gives a dense representation of the data compared to the more sparse representation given to non-negative values by Relu. Overall, we should expect the model performance to decrease in the new model- as shown in our results.

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

   Increasing the number of epochs from 25 to 40 increased the model accuracy and decreased loss. This is because more epochs means passing the data through the network more times and optimizing the weights for each layer. However we can see that the change performance is relatively small, which could suggest that adding more epochs might not be effective at this point.

3) What are the advantages of the mini-batch vs. SGD?

   The mini-batches method uses multiple samples at a time to update the model parameters, while SGD introduces the samples to the model one by one. This means that using mini-batches allows implementation in vectorized form for faster computations and more efficient use of memory, while still having the main advantage of SGD which is frequently updating parameters. SGD updates the weights more frequently, and is faster than the mini-batch method. A major disadvantage of SGD is oscillations, which lead to a worse estimation of gradient than the mini-batches method.

4) Build the `new_a_model` again and add batch normalization layers. How does it impact your results?

   Introducing batch normalization to the model improved accuracy and reduced loss. Centering and scaling the data for each layer usually improves performance and convergence is faster.

5) Have a look at the model below and answer the following

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

There are 5 convoluted layers in the model, and a dense tail of 3 layers. The layers have 64, 128,128,256,256 layers respectively. The number of parameters will be lower than of a fully connected NN, because CNN is more efficient in weights and connections. This NN is performing kernel regularization at each of the convoluted layers.

6) 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?**

Cutting the number of filters in half produces a model with smaller loss and smaller accuracy.