Learning rates, activation functions, and optimization algorithms by JUSTINE Jean using chatGPT 01/12/2024

I. Deep MLP on MNIST Dataset

For this first tasks we were asked to create a neural network to train on the MNIST Dataset and achieve a 98% accuracy and to find the optimum learning rate and to plot the learning curve using TensorBoard.

To start, we import the data using tensorflow and we cleaned the data,we firstly scale the data by dividing it with 255, to have it between 0 and 1, then we flattened the images to have it in one dimension using the reshape function from numpy that allow us to manipulate the shape of the data. We also encoded the label in array using the “to categorical” function to encode them in form of array (e.g. the label 1 become [0 1 0 0 0 0 0 0 0 0] array).

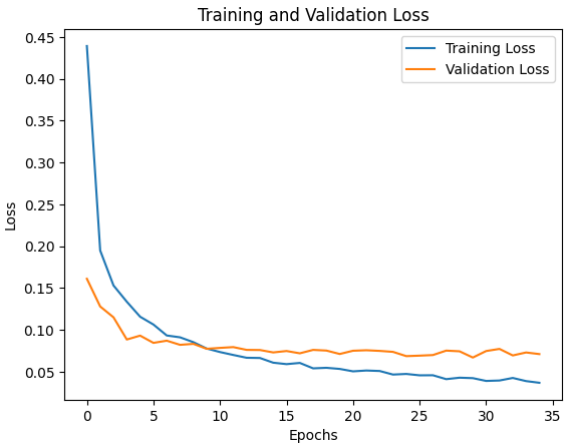
After that we create the neural network using the Sequential and dense function from tensorflow keras, we create two hidden layers, with Relu as activation and a dropout rate of 0.5 between each hidden layers to prevent overfitting as it was a problem that arose at one point. As input we put the size of the 1D images which is 28*28=784. And an output layer of 10 neurons with softmax activation to have the probability of each number to be the predicted.

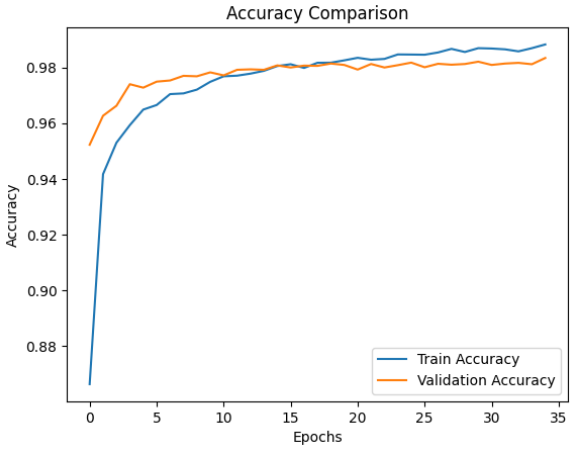
We also developed a callback, an early stopping, that stopped after 5 epochs with no improvement using early stopping function from tensorflow.

Then we compile the model with Adam as optimizer and categorical cross entropy for loss and accuracy as metrics.

Finally, we train it with the fit function, with a validation split of 0.2, 40 epochs and 128 batch size.

With all those first parameters, we achieve an accuracy of 98.33% without using a specific learning rates.



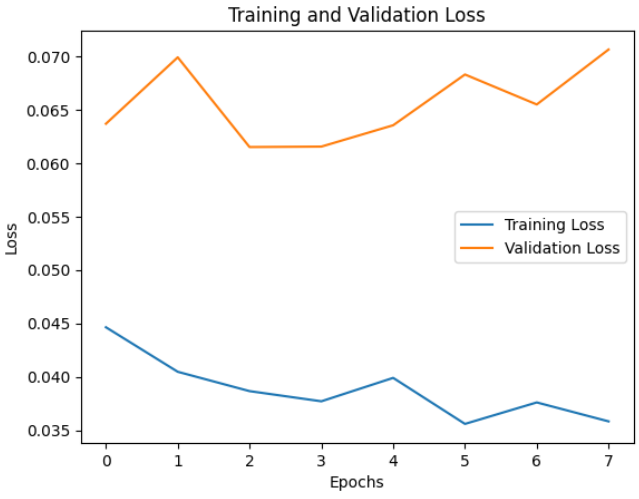


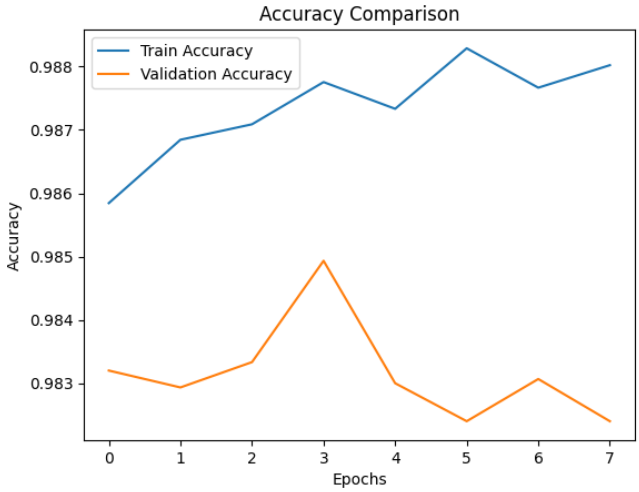
In those two figures we can see our model accuracy and loss compare to the validation set in function of the epochs, the model seems to perform well but a slight overfitting can be observed.

After that we tried changing the learning rate to have better results and we also used tensorboard to monitor our model.

I asked chatgpt to help me put the tensor board in action as I was not familiar with it basically we use the tensorboard function to create a callback that will register and monitor the learning of the model and will display, it using this command: “tensorboard --logdir logs/fit “in the terminal and open the local host in our navigator.

We tried to improve our model with searching for a learning rate, I found that 0.0001 was the best and could improve our accuracy of 0.1% as it was already really high mainly to the fact that it was the basic learning rate for the Adam optimizer I used.





The callback function stopped us as 7 epochs because the model did not improve much, there is still a very slight overfitting but we found a good learning rate and understood how to use tensorboard.

II. 100 LAYERS NEURAL NETWORK FOR FASHION MNIST

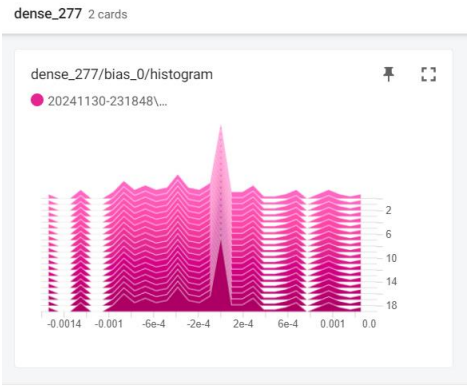
We were asked to build different neural network with 100 layers to train on the fashion MNIST with different activation function for each model.

Firstly, we import the data using tensor flow and we cleaned it exactly like the MNIST dataset in the first task.

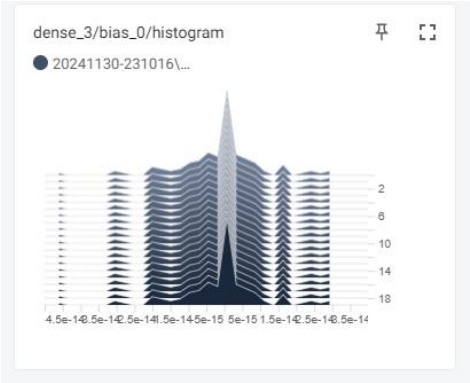
Then we create the Neural Network, we create the input layer like the first task and then we use a for loop to create 100 layers with each layers using 128 neurons and the relu activation at first, and finally the output layer with 10 neurons and the softmax activation.

We do those task multiple time for each different activation that were asked: Relu, selu, elu and sigmoid. We also used again the tensorboard function (we took the same code as the first task for that).

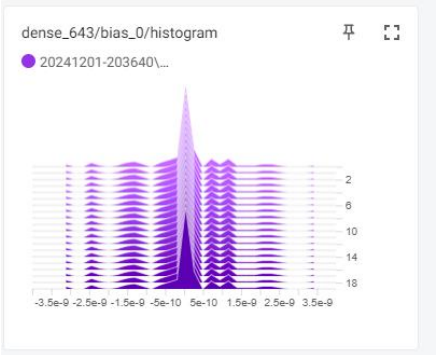
With tesnorBoard we could witness the vanishing gradient:



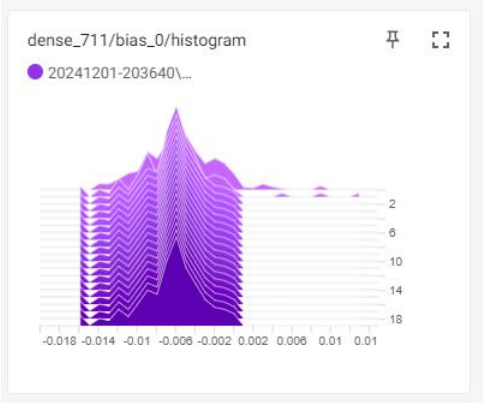
That figure is from the elu training were we can see a vanishing gradient for the 77 neurons layers of the bias.



We can witness the same thing but for sigmoid activation. And once again for selu:



But we could not find exploding gradient that stay even with the last activation : the relu function

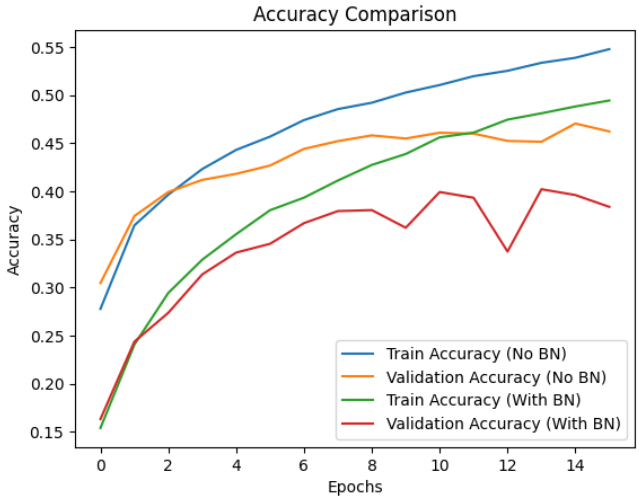


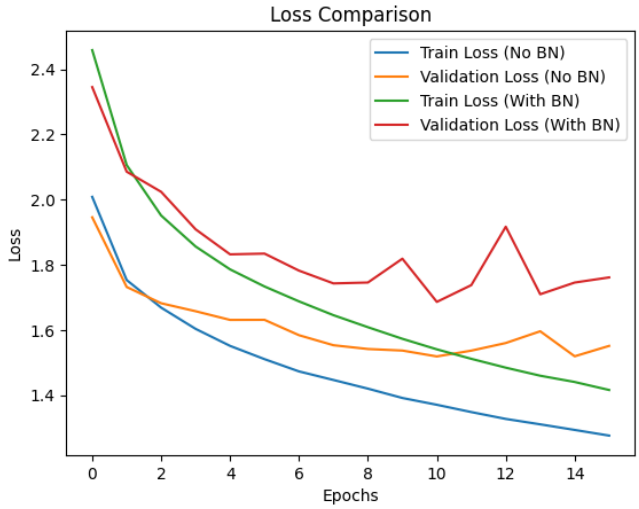
III. DNN ON CIFAR10 DATASET

Firstly, we as always import the dataset and clean it to use it on our models, for this dataset it's the same as the mnist or mnist fashion we divided it by 255 to scale it, but this dataset is not in greyscale image so we had to do instead of 32*32 for 32 pixels by 32 if it was in great scale image, we did 32*32*3, the 3 is for RGB pixel, we also encode the label like the MNHIST dataset.

I create the model like the second task, a first layer with 3072 input, 100 neurons Relu and He initialization, then with a for loop going through a 20 loop which each add a layers with the elu activation function and HE normal as initialization and 100 neurons each and we end with the last layer with 10 neurons and sigmoid function once again.

After that we compile the model with accuracy and categorical crossentropy once again with the Nadam optimizer we also created a callback with earlystopping like in the previous tasks. We train the model, with 30 epochs, 0.3 validation set 128 batch size and the early stopping with a patience of 5 and we iterate through a list of learning rates: learning_rates = [0.1,0.01,0.001,0.0001], to find the ideal one. After that we build the same model but with Batch normalization between each layers, and we obtained those curve :





We had poorer result with batch normalization, but it was much quicker and we needed less epochs to reach the maximum accuracy for the validation set. It could be improved with dropout to minimize overfitting especially with that many layers.

Lastly we train the model but with different optimizer to see the difference:

All the model has a pretty similar performance of 40% to 50% accuracy. But some were quicker than others like NAG, who were stable and converged quickly, AdaGrad was the most efficient with 51% accuracy, Nadam, was the fastest and was pretty stable too, SGD and RMSProp were the slowest and with the lowest accuracy. Adam was like always good and stable but not the quickest.

But I didn't search for the best learning rates, because it would have taken too long to train all the optimizer with a different learning rates, it was already long without so there is much to improve than those results.

To conclude we could experience a lot with different optimizer and see the influence of the number of hidden layers throughout this whole assignment.

I used chat GPT to help me plot the curve, help me with tensorboard set up, and also when I got errors and couldn't find the solution.