Laboratory Assignment 1

Task1) Run the above code and interpret the results. If you run the code several times, will you observe the same results? Why?

If we run the code several times we dont obtain the same results because everytime we are updating the weights and calculating ssd and due to this we get different predicted values every.

Increase the number of iterations starting from 10, 50, 100, 500, 2000 and compare the loss values.

The more iterations the closer the predicted values are to the true ones.

Task2) Repeat task2 for XOR operator. For the same number of iterations, which operator has lower loss values? Why?

The AND operator gives better predictions of the desired values with high number of iterations.

Increase the number of neurons in hidden layer (n_unit) from 2, 5, 10, 50. Does increasing the number of neurons improve the results? Why?

On increasing the number of neurons at a certain level, the network prediction is closer to the real values. However, with this it also increases the complexity therefore there's a point when it gets too difficult to perform a good prediction and the results get worse.

Task3) In the above code change the n_unit as 1, 2, 4, 16, 25, 50 and interpret the observed results.

It didn't really affect the prediction whatever number of units we use.

Task4) Develop a 4-Layers MLP with the following setting: 4 fully-connected layers with Base, Base//2 and Base//4 as the number of neurons at the first 3 layers and Relu activation function. For the last layer choose a proper number of neurons as well as activation function that fits the binary classification task.

How do you interpret the observed values of loss and accuracy? Set the learning rate parameters as 0.01, 0.001 and 0.0001 and try again. Which learning rate produced more stable results? Which learning rated yielded over-fitted results?

LR	Loss	Binary accuracy	Val loss	Val binary accuracy
0.01	0.612	0.672	0.609	0.69
0.001	0.59	0.68	0.61	0.68

0.0001	0.66	0.68	0.66	0.68

The higher LR the better results

Task5A) Set the following parameters: # of epochs = 20; batch size = 8; number of feature maps at the first convolutional layer = 32 and learning rate = 0.0001 and then run the experiment. What are the training and validation accuracies? What can you infer from the learning curves?

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.001	32	20	0.57	0.774	0.589	0.750

Task5B) Leave all the parameters from previous step unchanged except the # of epochs = 80. What is the major difference between the observed results and what you achieved from previous step?

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.001	32	80	0.24	0.96	0.67	0.79

If the plots are parallel to each other it shows the performance of the model is consistent. but if it starts to depart then it's probably a sign to stop training at an earlier epoch.

Task5C) Reduce the number of feature maps at the first convolutional layer to 8 and let the model run for 80 epochs. Compare your results with the steps 5B. How can you interpret the changes?

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.001	8	80	0.48	0.79	0.53	0.77

= Better

Task5D) Keep the setting of step 5C but increase the learning rate and set it as 0.01; What do you conclude from task5? How should you evaluate the generalization of your model?

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.01	8	80	0.0019	1	1.65	0.79

= Overfitting

To handle overfitting we should:

- Reduce the network's capacity by removing layers or reducing the number of elements in the hidden layers. However reducing too much will lead to underfitting which means network won't be able to learn patterns in the train data.
- Apply regularization.
- Use dropout layer.

Task 5E) What is the role of the first two convolutional layers?

The use of convolutional layers is to learn high-level features.

Task 5F) What is the role of the last two dense layers?

It is used to change the dimensions of your vector. Mathematically speaking, it applies a rotation, scaling, translation transform to your vector.

Task 5G) What is the major difference between the LeNet and MLP?

MLP has input, hidden and output layers. It uses non linear activation function. It depends on the training of back propagation. It can distinguish data that is not previously labeled. The major disadvantage is that it might require too many parameters and it becomes very dense. All the nodes are connected

LeNet is easier to be trained than MLP, this can also go more dense. It can take both matrix and vectors as inputs. All the nodes are not connected with other ones.

Task 5H) Look at the last layer of the network. How should we choose the number of neurons and activation function of last layer?

It depends on the number of outputs and labels.

Taks6A) Read the skin images with the size of 128*128 and train the AlexNet model with the following parameter: Batch_size = 8; Epochs = 50; Base = 32; learning rate = 0.0001; Evaluate the model performance.

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy

0.0001	32	50	0.0874	0.96	1.15	0.78

The graph is overfitted.

Taks6B) Change the Base parameter as 16 and 8 and try again.

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.0001	16	50	0.23	0.89	0.57	0.80

The overfitted is improved with the base 16.

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.0001	8	50	0.46	0.80	0.49	0.80

Now there is no overfitting.

Tax 6C) Set the Base = 8 and Batch size = 16. How do you interpret the observed results?

Higher batch size will require more memory space. Basically, it is the number of training samples in one forward/backward pass.

LR	Base	Epoch	Loss	Binary accuracy	Val loss	Val binary accuracy
0.0001	8	50	0.46	0.80	0.49	0.80

It is leading to overfitting.

Task6D) By finding the optimum values of batch_size, learning rate, and base parameters, train the model for 100 epochs and make sure it is not overfitted. Report the classification accuracy of the best model.

LR	Batch size	Base	Epoch

0.0001	8	8	100
0.0001	8	4	100
0.01	8	4	100 BEST (adam)
0.001	8	4	100 NO
0.001	16	4	100 NOOOO
0.01	8	8	100 NO

Then, for this model, only change the optimizer algorithm from Adam to SGD and RMSprop and compare the observed results.

SGD: better, exponentially decreasing.

RMSprop: bad, overfitting.

Task6E) "Binary cross entropy" is not the only loss function for a binary classification task. Run the code again by changing the optimizer into "hinge".

Hinge gives worse results.

Taks7) Read the skin images with the size of 128*128 and repeat the Task6D for the VGG16. For this exercise, you need to implement the model first. Use the framework from previous exercise and just replace the model function with your implementation.

For the skin images, compare the results you achieved from the 3 implemented models: LeNet, AlexNet, and VGG.

AlexNet

Task8) Repeat Task7 for the fractured bone image classification. Please note, in order to read the bone images, you need to apply some changes to the data loader:

AlexNet was the best performed model for the given bone images.

Task9) With the implemented models, which dataset could be better classified? Skin or bone images? How do you make sure that achieved results are reliable?

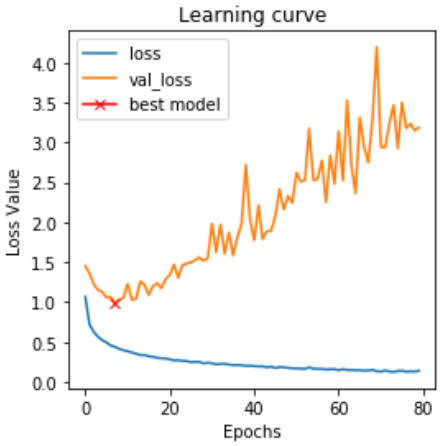
Skin dataset was best classified from the implemented models.

Bonus Task) In previous tasks, you experienced binary classification tasks by implementing three deep networks. In this bonus exercise, the data set includes X-ray images of 9 different organs. Therefore, you are expected to extend the implemented models into a multi-class classification task. Modify

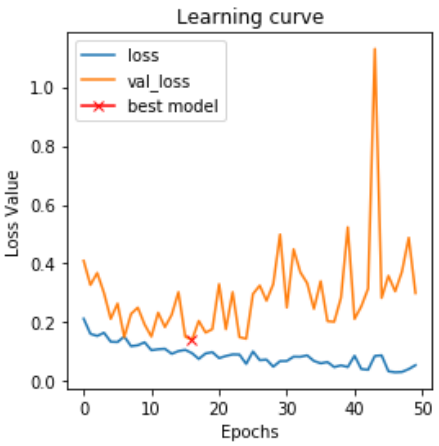
the data loader to properly load the images along with their class labels. Extend the LeNet, AlexNet, and VGG models for multi-class classification tasks. Tune the models by finding the optimum values for hyperparameters and compare the achieved results from the three models. Report the learning curves for both of the loss and accuracy values (for train and test data).

We did the bonus task and we implemented all the three networks on the given Xray images.

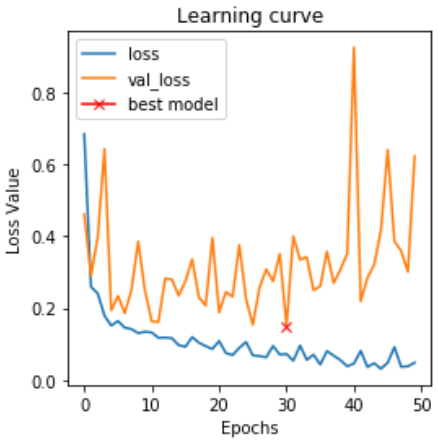
LeNet:



AlexNet:



VGG:



These were the plots we obtained after implementing the respective model with

loss: 0.1386 - accuracy: 0.9595 - val_loss: 3.1886 - val_accuracy: 0.5978 for LeNet

loss: 0.0542 - accuracy: 0.9822 - val_loss: 0.2991 - val_accuracy: 0.9489 for AlexNet

loss: 0.0479 - accuracy: 0.9824 - val_loss: 0.6237 - val_accuracy: 0.9422 for VGG

So the LeNet had overfitting and the rest two models performed better as compared to LeNet.