Gnana Spandana Akumalla Serkan Arda Yilal Patrik Svensson

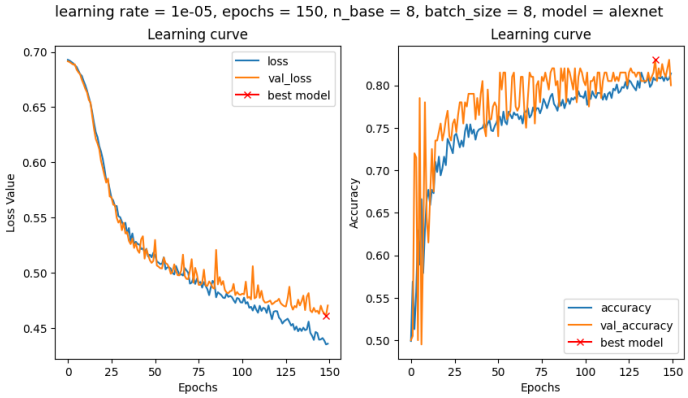
CM2003-Lab 3

Task 1 A) The implemented architecture of the VGG16 Model can be checked in **Models.py** file under the function called *vgg16_model*.

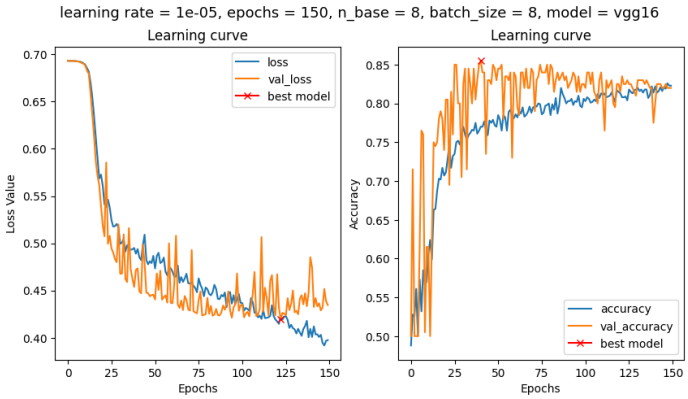
Task 1 B) In this specific task, we searched for optimum value of the learning rate that would lead a reliable classification performance over the validation set. The values that we checked for learning rate are [0.01, 0.001, 0.0001, 0.00005, 0.00001, 0.000001]. The optimum value chosen after the search was learning_rate = 0.00001 with a maximum validation accuracy of 0.85 while the others could not reach that level. The plots regarding the loss and accuracy change can be checked in **Lab3- Part 1.ipynb** file.

Task 1 C) In this task, we used the same settings mentioned in task 6C of Lab 2 with n_epoch = 150 and then trained AlexNet and VGG16 models. The loss and accuracy plots of both models are as follows:

- AlexNet:



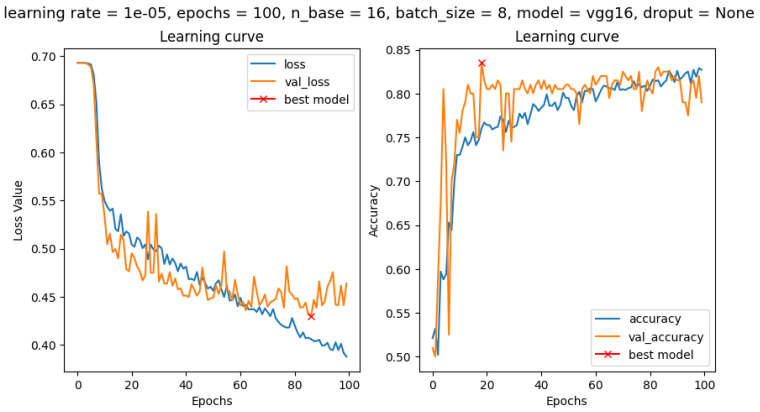
- VGG16:



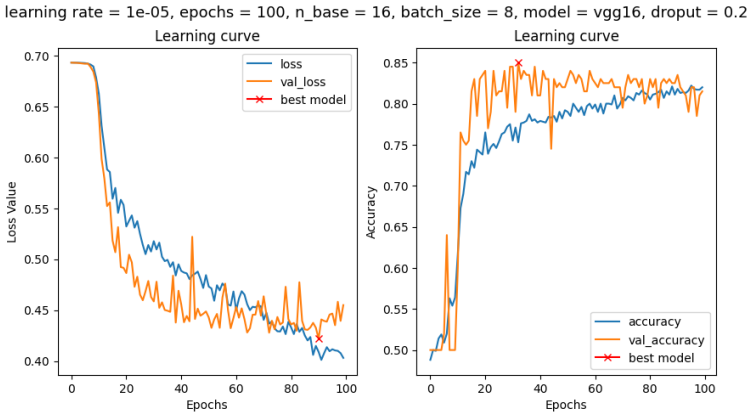
It can be seen that VGG16 model is able to get lower validation loss values in earlier epochs and also it is able to get higher validation accuracy than AlexNet did in earlier epochs. While the maximum validation accuracy of VGG 16 model is 0.85, it is around 0.82 for AlexNet.

Task 1 D) In this task, we updated the n_base parameter value of VGG 16 from 8 to 16 and we compared its performance without and with dropout rate of 0.2.

- Without dropout:



- With dropout rate = 0.2:



If we compare the results between n_base = 8 and n_base = 16 without dropout, we see that training loss of n_base = 16 reaches to similar levels of n_base = 8 much earlier and it causes some overfitting in the end as the validation accuracy is not as high as in n_base = 8. Also, validation loss seems to achieve lower levels with n_base = 8.

When it comes to comparison between without and with dropout rate of 0.2 for n_base = 16 model, we see that dropout helps model to achieve slightly better validation accuracy and slightly lower validation loss which might mean that dropout regularized our network.

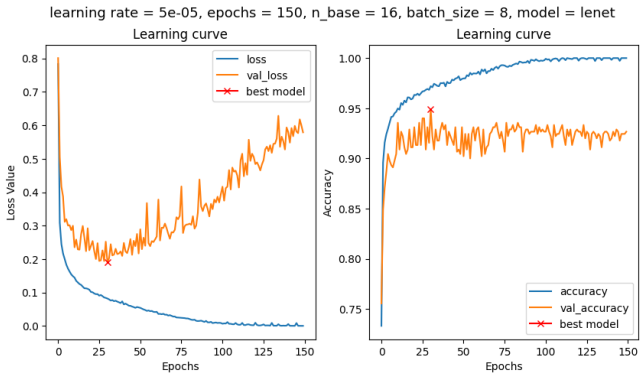
Task 1 E) If we compare the performance of LeNet, AlexNet, and VGG16 on skin cancer dataset so far, we can say that the best performing model was VGG16, then AlexNet, and lastly LeNet. The main difference among these models can be stated as the depth of the convolutional layers with a descending order of VGG16, AlexNet, and LeNet. When it comes to validation accuracy, the results in these labs so far shown that LeNet achieved a score around 0.75, AlexNet around 0.80, and VGG16 around 0.85. This might be because the deeper the network got; more useful features were possible to extract which in the end helps classification task. If we analyze validation loss, we are able to see if our model gets overtrained based on the existence of an increase in validation loss and a continuing decrease in training loss. By adding dropout layers, we can avoid this overtraining as the models get deeper and wider and more sensitive to overtraining.

Task 2) In this task, we trained VGG16 model on Bone dataset and searched for an optimal learning rate value among the values [0.001, 0.0001, 0.00005] for n_base = 8 and n_base = 16. The plots for the learning rates can be checked in **Lab3- Part 1.ipynb** file.
Based on the results, both models could not decrease validation loss with learning rate of 0.001 and as the learning rate got lower values, the models were able to decrease validation loss and increase validation accuracy. Although the optimal learning rate was chosen as 0.00005 for both models, the models got overtrained as validation loss started to increase around 40 epochs.

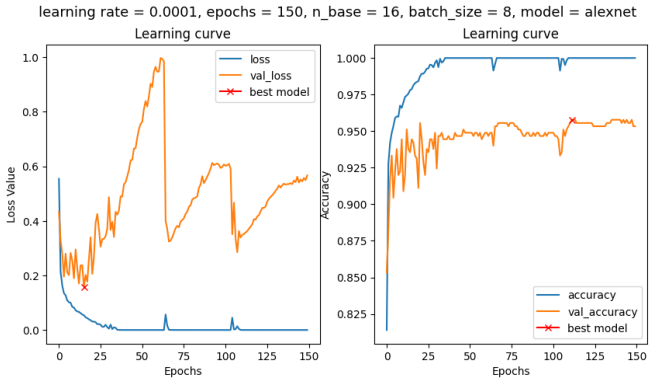
Task 3) The validation accuracy achieved on the dataset was 0.95 by n_base = 16 with learning rate = 0.00005. This shows that the model performed better on Bone dataset compared to skin dataset. The reason for that might be related to information that bone images have inside them compared to skin dataset and it might be easier to classify them based on their content. To make sure that the achieved results are reliable, different data shuffles can be used or different random seeds can be used on weight initialization of network so that we will make sure the results do not change based on our selection.

Task 4) In this task, we extended LeNet and AlexNet architecture to a multiclass classification problem in order to classify X-ray images of 9 different organs. A hyperparameter search was done among learning rate = [0.001, 0.0001, 0.00005] and n_base = [8, 16] for both models. In the end, best setting for LeNet was learning rate = 0.00005 and n_base = 16 while the best setting for AlexNet was learing rate = 0.0001 and n_base = 16 based on the lowest validation loss and highest validation accuracy achieved. The plots are as follows:

- LeNet:



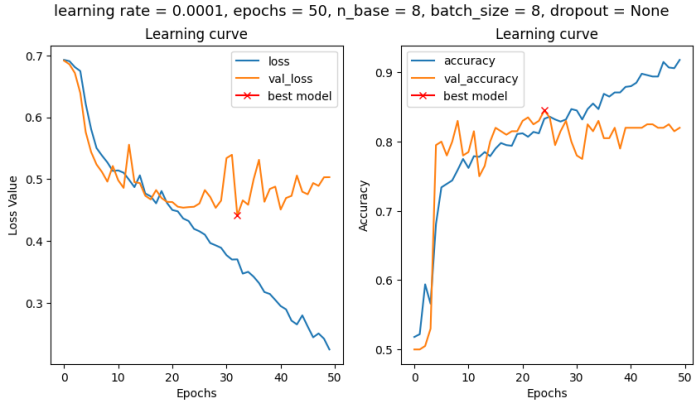
- AlexNet:



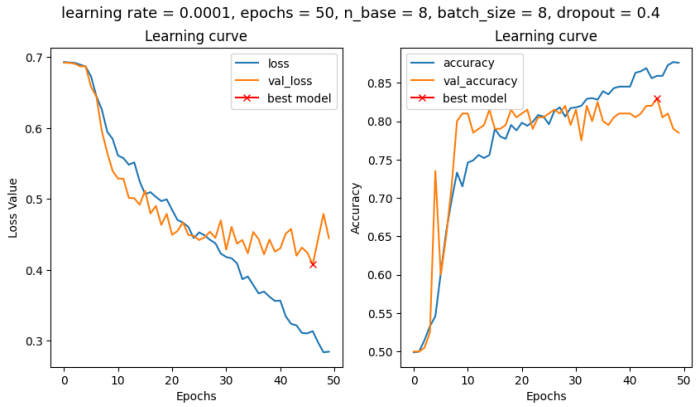
When we compare the results, their maximum validation accuracies were quite close with a score of 0.948 for LeNet and a score of 0.957 for AlexNet. Still, we can see that both models are getting overtrained based on the increase we can observe in validation loss. If we regularize them, even better performance might be achieved.

Task 5 A) In this task, we trained AlexNet on Skin dataset with 50 epochs without and with dropout rate of 0.4. The results are as follows:

- Without dropout:



- With dropout rate = 0.4:

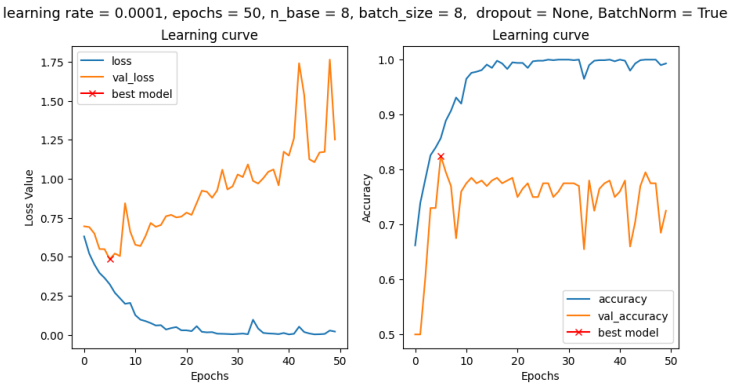


When we check the training and validation accuracy of no dropout case, we see that training accuracy increases up to 0.9 around epoch 50 while validation accuracy achieves a value around 0.84 around epoch 24. Also, validation loss stops decreasing around 20 epochs. After we add the dropout layers, we see that the stop of validation loss decrease is postponed to and epoch of 26 and the minimum validation loss achieved was lower than no dropout case. Still, the validation accuracy did not increase

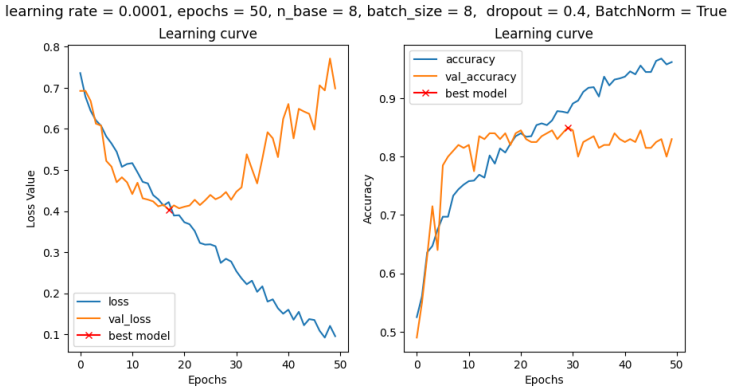
so much while it prevented model having very high training accuracy probably because of the regularization impact of it.

Task 5 B) This time, we will include batch normalization in our model and compare its effect on the cases without and with dropout rate of 0.4. The plots for the cases are as follows:

- Batch normalization without dropout:



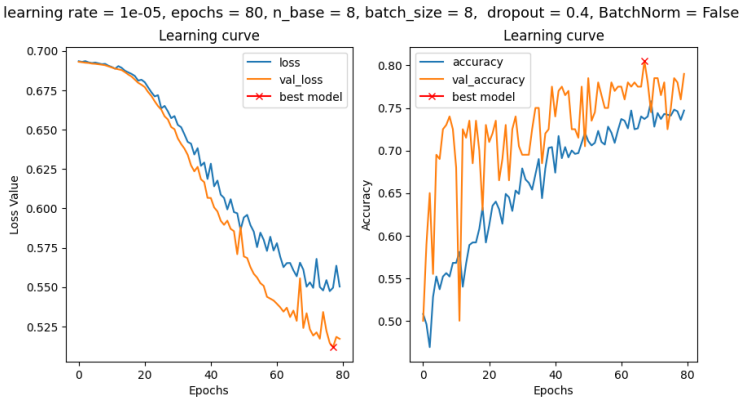
- Batch normalization with dropout rate = 0.4:



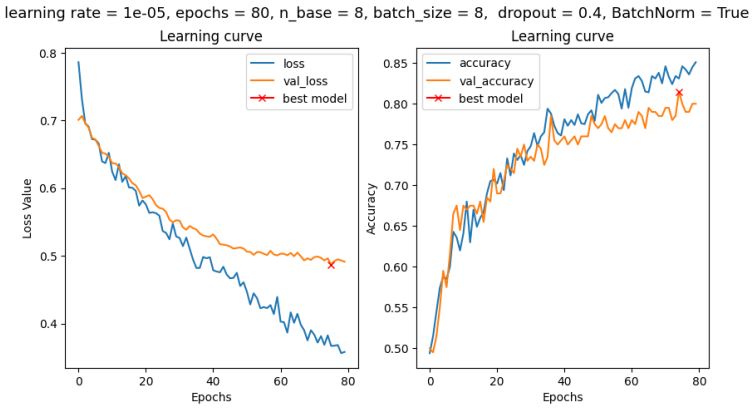
When we check the effect of batch normalization on no dropout cases, we see that adding batch normalization made model achieve the same training accuracy (0.9) around 10 epochs while the model without batch normalization achieved it around 50 epochs. When we check the training accuracy in the end of training of batch normalized model, it is almost 1. Based on these results, it seems like batch normalization caused model to get even more overtrained as validation loss starts to increase around 8 epochs. When it comes to the addition of dropout layer, it regularized the model and postponed the increase in validation loss. Based on the search we did, the problem with batch normalization might be related to batch size chosen for this task as it is known as it is not so effective on small batch sizes.

Task 5 C) In this task, we train the model this time with learning rate = 0.00001 and epochs = 80. This time, we will have two cases where we have no batch normalization but dropout rate = 0.4 and batch normalization with dropout rate = 0.4. The loss and accuracy plots of these models are as follows:

- Without batch normalization with dropout rate = 0.4:



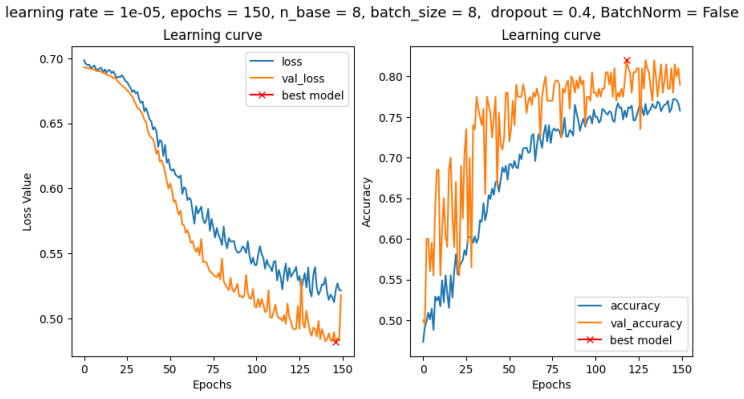
- With Batch normalization and dropout rate = 0.4:



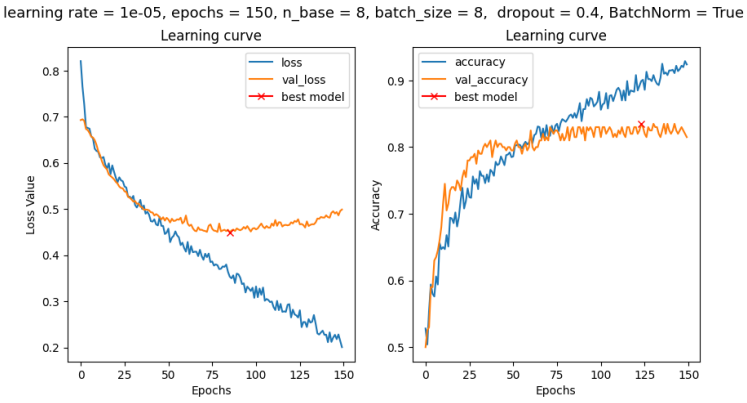
When we compare the validation accuracies of these models, we can see that without batch normalization, the model achieved a fluctuating progress with a maximum validation accuracy 0.8 while the batch normalized model had less fluctuation and achieved a maximum validation accuracy slightly higher than 0.8. If we check the loss progress, the validation loss actually had similar values in both cases while training loss decreased much faster in the case of batch normalization. This might show that although the model did not get overtrained, it learned the training data better with batch normalization.

Task 5 D) In this task, we replicate the things we did in previous task but train the models with 150 epochs instead of 80 epochs. The loss and accuracy plots are as follows:

- Without batch normalization with dropout rate = 0.4:



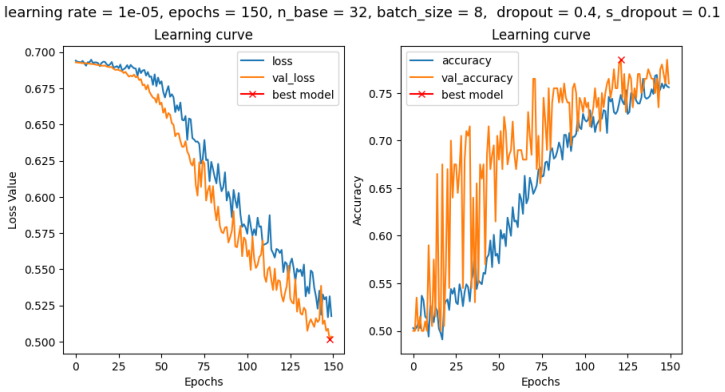
- With Batch normalization and dropout rate = 0.4:



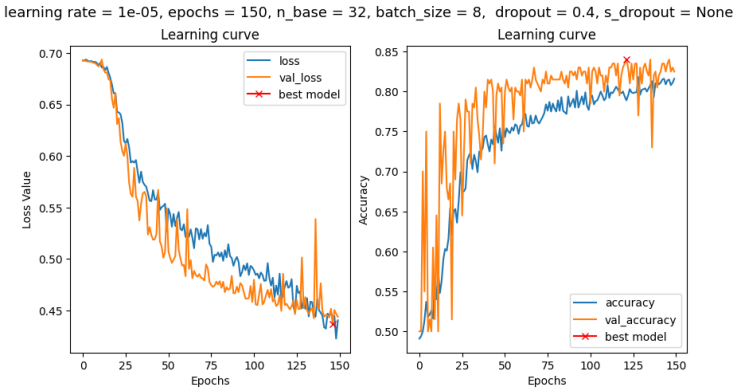
If we compare these models with previous models, we see that increasing the epoch made the batch normalized model to get trained even better on training data as it achieved the lowest training loss with a score of 0.2. On the other hand, after epoch 80-85, the validation loss slightly started to increase with batch normalized model while it is low in model without batch normalization. Hence, the model without batch normalization can be considered more generalized while both models had similar maximum validation accuracy although batch normalized model had higher training accuracy.

Task 6 A) For this task, we will use the same model we designed in previous tasks but with a base parameter of 32 and replacing batch normalization with spatial dropout rate of 0.1. We have two cases in this task where we have a model with spatial dropout rate = 0.1 and normal dropout rate = 0.4 and another model with no spatial dropout and with only normal dropout rate = 0.4. The accuracy and loss plots are as follows:

- With spatial dropout rate = 0.1 and dropout rate = 0.4:



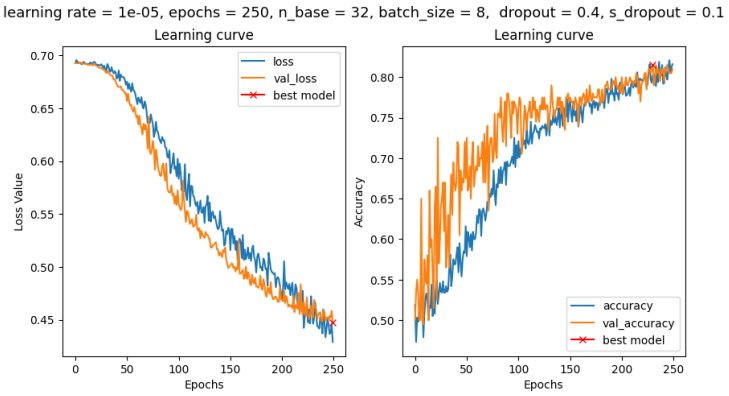
- Without spatial dropout, with dropout rate = 0.4:



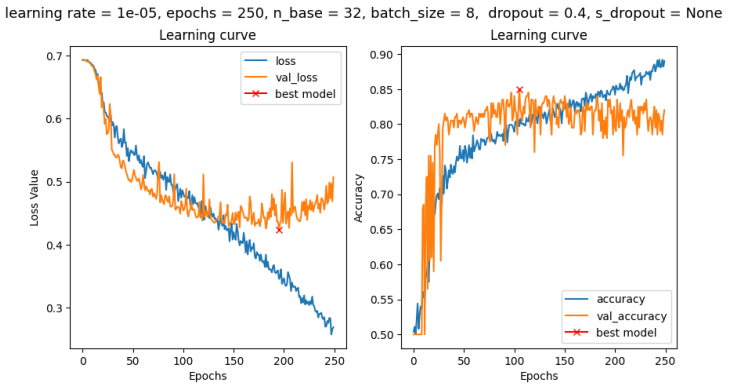
When we compare the accuracy and loss values of these models, we see that model with no spatial dropout layer achieves lower training and validation loss and higher training and validation accuracy compared to model with spatial dropout layers. Hence, we can say that model without spatial dropout layers converges faster. The reason for that might be because addition of spatial dropout layers might model become even more regularized which would prevent it from overfitting but as a drawback, it might converge slower than the model with no spatial dropout.

Task 6 B) In this task, we repeat the previous task with epochs = 250 instead of epochs = 150. The same models were used. The accuracy and loss plots are as follows:

- With spatial dropout rate = 0.1 and dropout rate = 0.4:



- Without spatial dropout, with dropout rate = 0.4:

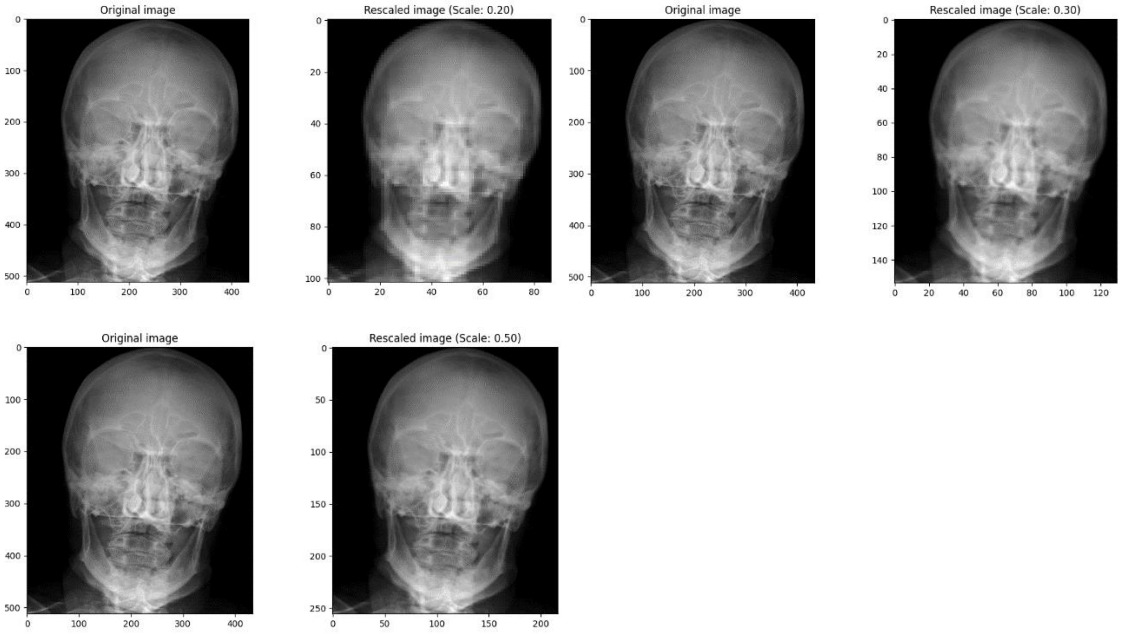


When we compare these models with the updated number of epochs, we see that model without spatial dropout layers starts to get overtrained after 200 epochs while the model with spatial dropout has a validation loss that keeps decreasing throughout the whole training. Although model with no spatial dropout layer achieves a higher validation accuracy in the whole training, the validation accuracy starts to decrease after some time. On the other and model with spatial dropout layers has an increasing validation accuracy during the whole training. In general, we can say that combining spatial and normal dropout layers provide a very effective regularization on training process. In case our models get overtrained so fast, spatial dropout layers can be included additional to normal dropout layers.

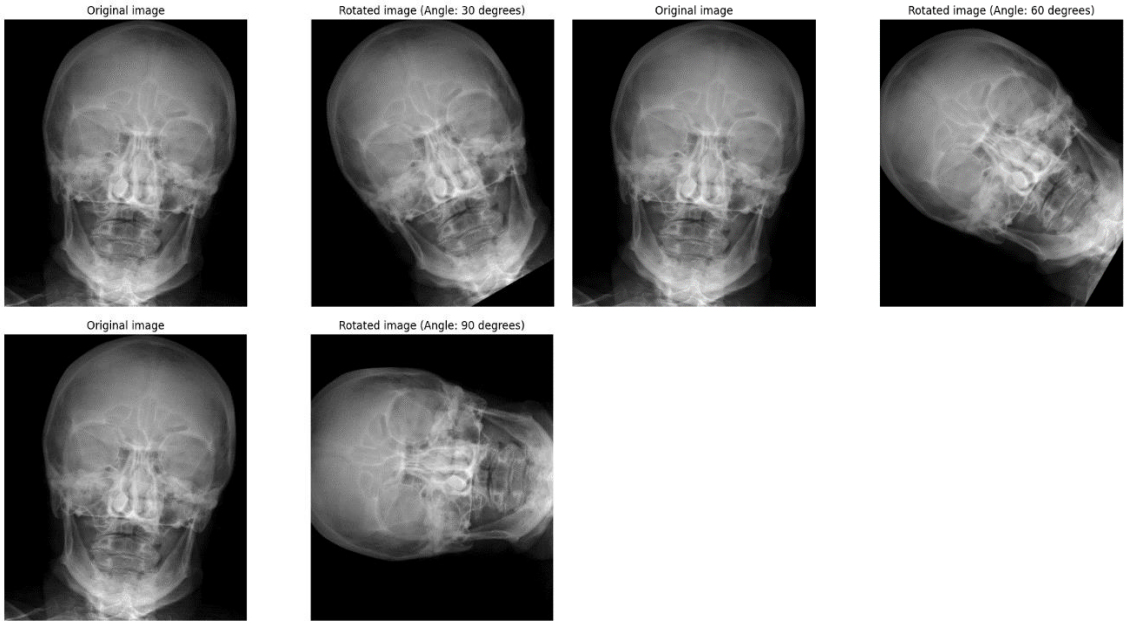
Task 7 A) In this task, we introduce data augmentation with operations such as geometrical and intensity transformations. These operations include scaling, rotation, horizontal/vertical flip and intensity rescaling. Their effect on the sample image are as follows:

Gnana Spandana Akumalla Serkan Arda Yilal Patrik Svensson

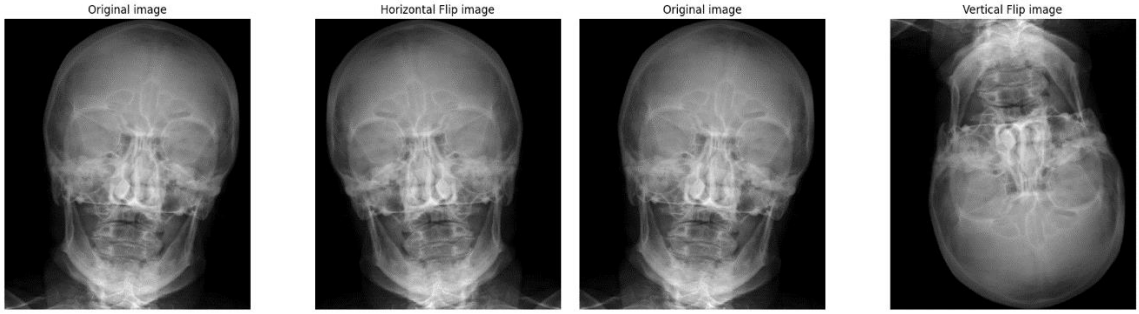
Rescaling:



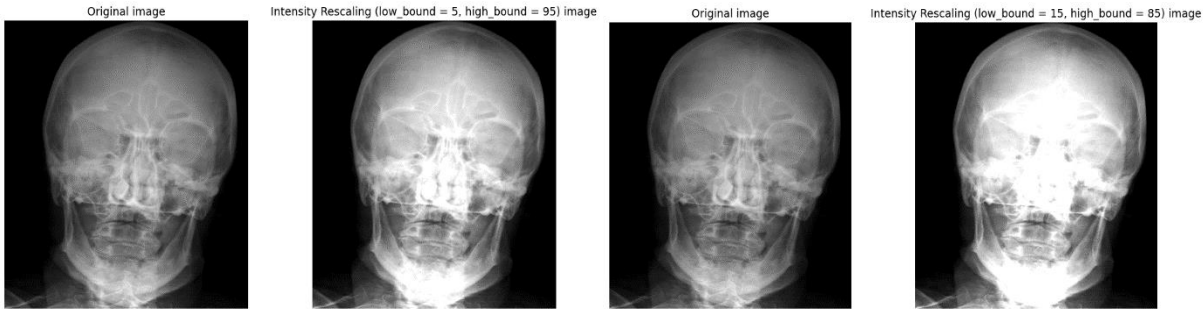
Rotation:



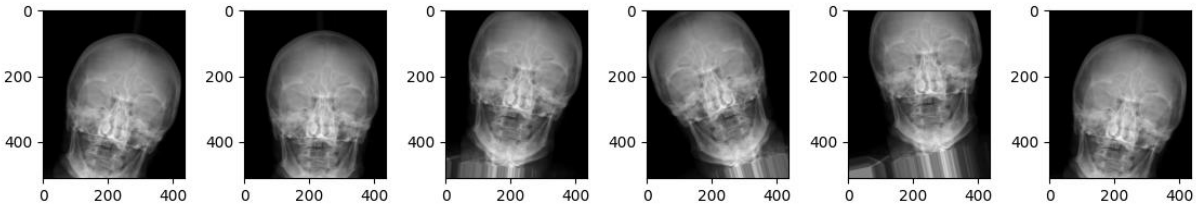
Horizontal/Vertical Flip:



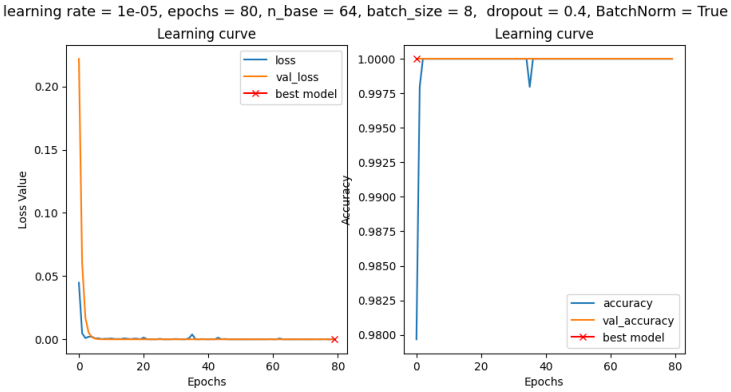
Intensity rescaling:



Task 7 B) The generator built with this task can be check in **Lab3- Part 3- Data Augmentation.ipynb** file.



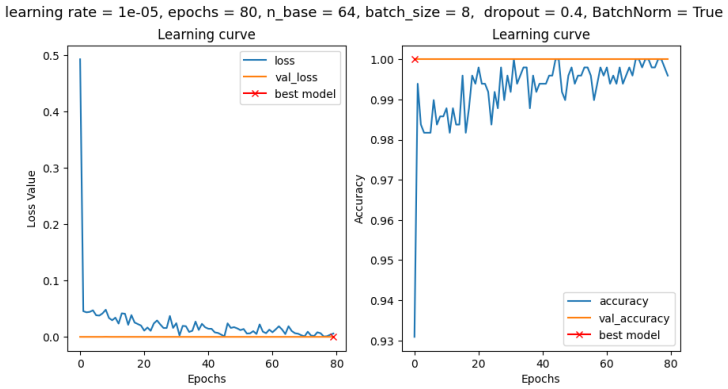
Task 8) In this task, we built a framework for training the models with augmenting the training data. By using the framework, we trained AlexNet with specified settings on skin images. The loss and accuracy plots of the model are as follows:



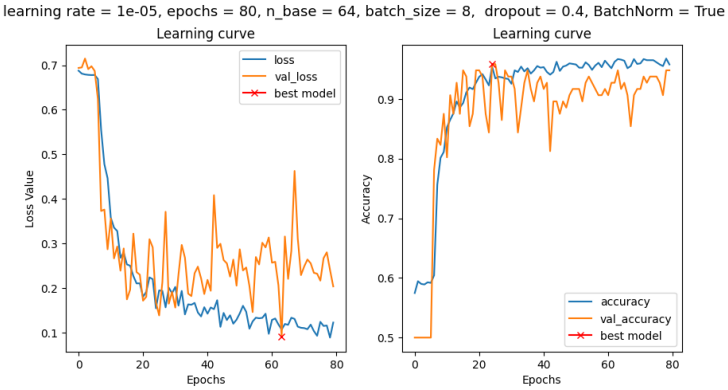
We see that adding a variety in the dataset helped model to achieve the best performance very fast. This might be because it led model to learn the features that would separate two classes much better and faster.

Task 9) In this last task, we trained VGG16 model instead of AlexNet on both skin and bone datasets. The loss and accuracy plots for VGG16 models trained on these datasets are as follows:

- VGG16 trained on Skin dataset:



- VGG 16 trained on Bone dataset:



We see that as in the AlexNet model, augmentation helped VGG16 model to be trained on the datasets much faster and better. But for bone dataset, we can see that its performance is not as good as it is with skin dataset.