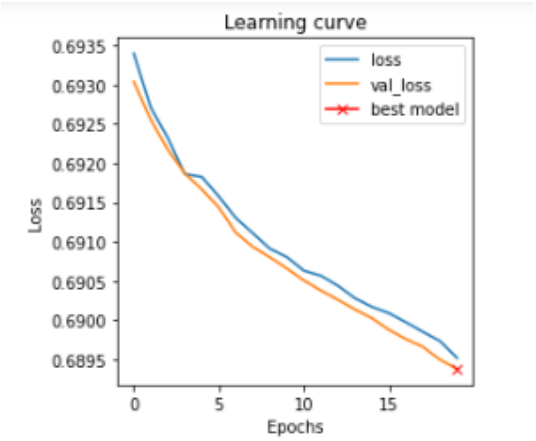
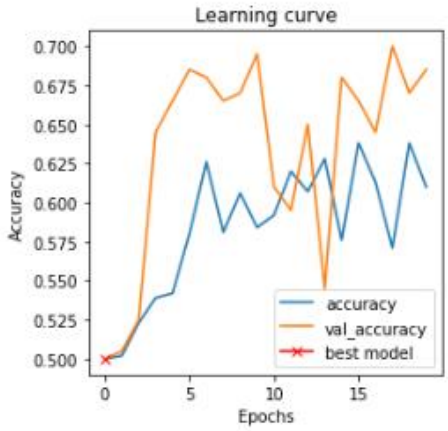
Gnana Spandana Akumalla Serkan Arda Yilal Patrik Svensson

CM2003-Lab 2

Task 5a (LeNet)

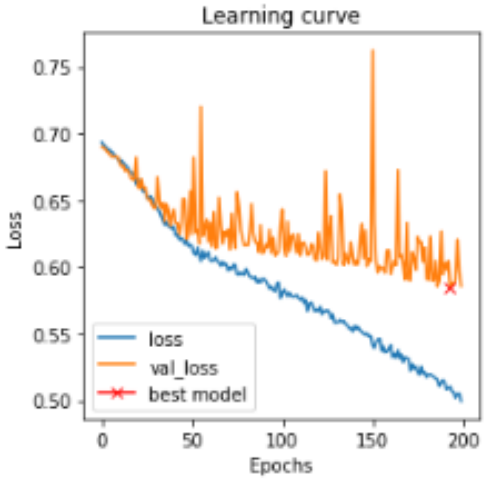
We can see that the loss curve is still decreasing fast at the 20th epoch. So increasing the number of epochs should generate more accurate models. We think it would be more reasonable to increase the epochs before making a decision from the produced model.

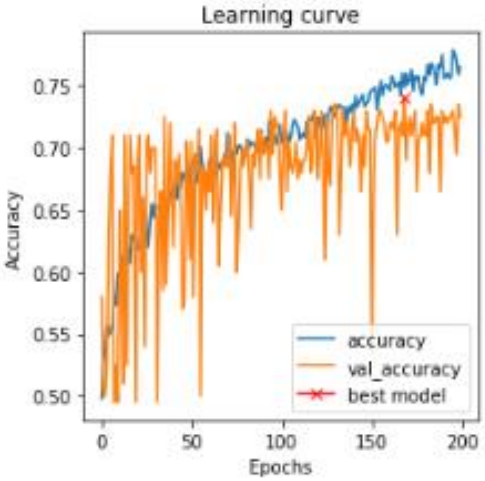




Task 5b

In the results we can see that it has time to train more and it reaches a higher accuracy.





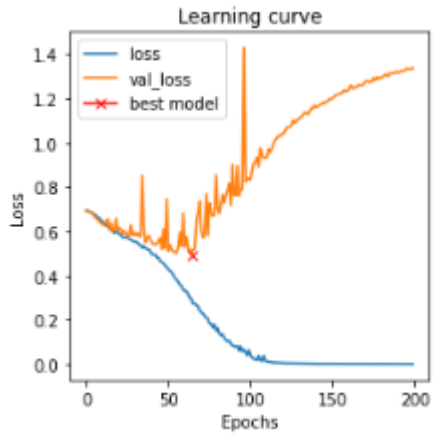
Task 5c,

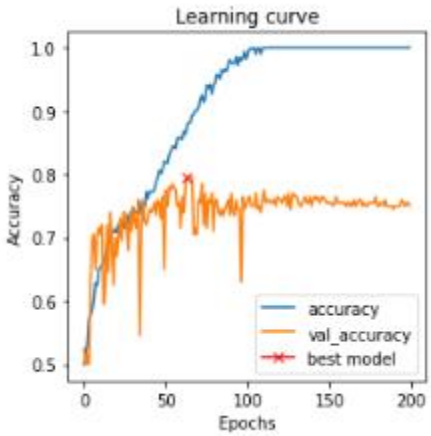
Keep all parameters from the last step except for the LR = 0.0001 . Run the experiment with new LR and interpret the results. How do you evaluate the generalization power of the model? What are the values of training and validation accuracies?

We can see that as the learning rate increases, it reaches higher levels of accuracy more quickly. In the case of LR=0.0001 the accuracy is consistently around 0.7 after approx. 80 epochs. In the case of LR=0.001 it reaches that level of accuracy after just a few epochs. In this case we also see that the model overfits much quicker. In the case of 5b it is possible to see that it starts to overfit because the accuracy for the training data is consistently higher than the validation data. Then when we increase the learning rate it overfits much quicker, reaching a training accuracy of 1. This behaviour continues when we increase the learning rate even higher, to 0.01.

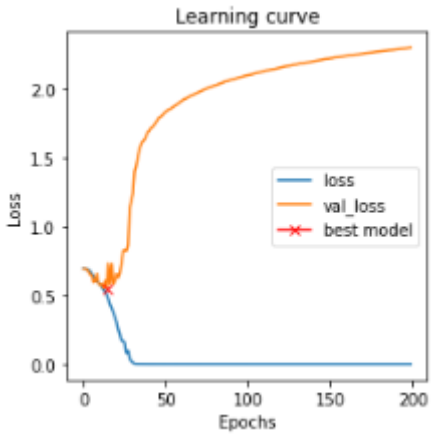
LR=0.0001 is the picture in 5b

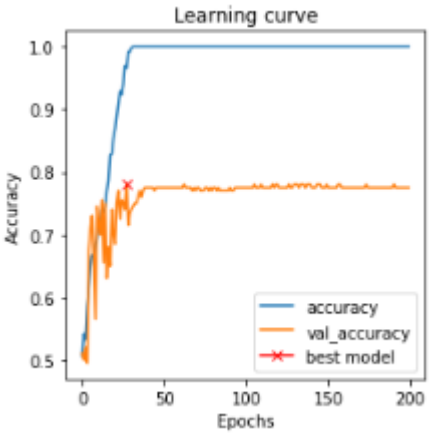
LR=0.001





LR=0.01





Task 5d,

What is the role of the first two convolutional layers?

The purpose of them is to pick up features from the input images. These features are entirely decided by the model, no human knowledge is given to the model regarding which features migt be useful.

Task 5e,

What is the role of the last two dense layers?

The dense layers at the end are a cheap way to learn non-linear combinations of the features. It is better than connecting the output of the convolutional layers (via flattening) directly to the output.

Task 5f,

What is the major difference between the LeNet and MLP?

It's a deeper network than the LeNet; the pipeline is a lot longer. This also means that it has many, many more parameters.

Task 5g,

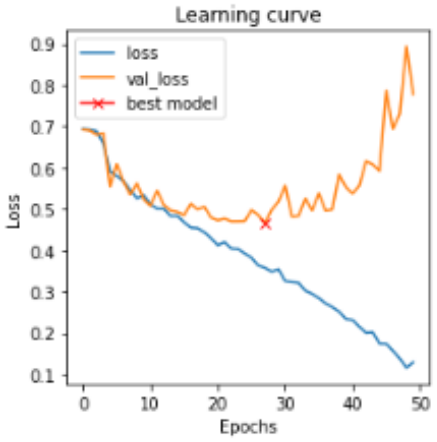
Look at the last layer of the network. How should we choose the number of neurons and the activation function of the last layer?

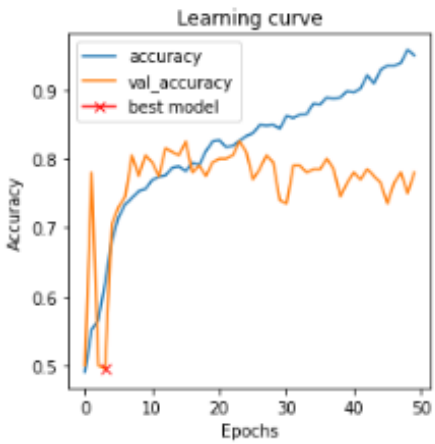
It depends on the number of classes that we have. If we want the model to classify between 5 different animals, then we will have 5 neurons in the last layer.

Task 6a (AlexNet)

Evaluate model performance. Batch=8, epochs=50, base=32, LR=0.0001

We can see that the model performs better than the LeNet-model. The AlexNet has an accuracy of almost 0.8 while the LeNet has an accuracy of around 0.7 (when estimating the performance after 50 epochs to make it comparable). It can also be noted that the AlexNet seems to be overfitting (as the accuracy is much higher on the training data (blue line) compared to the validation data (orange)). Therefore it should be possible to improve it even further.





Task 6b

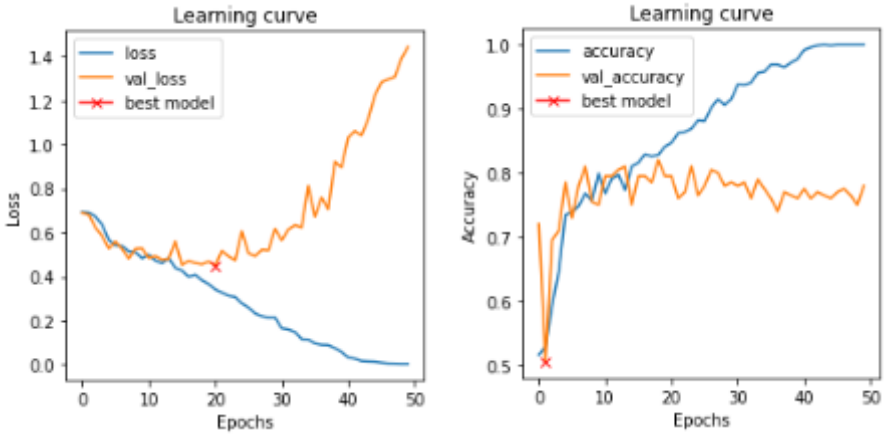
Base=16,8; epochs=50;added dropout;epochs=150. Interpret first results. What is the effect of the dropout layer? How do you explain the effect of increasing the number of epochs?

Using base 16 and no dropout seems to be a case of overfitting, as it performs very well on training data (even has an accuracy of 1 towards the end), while not as well on validation data.

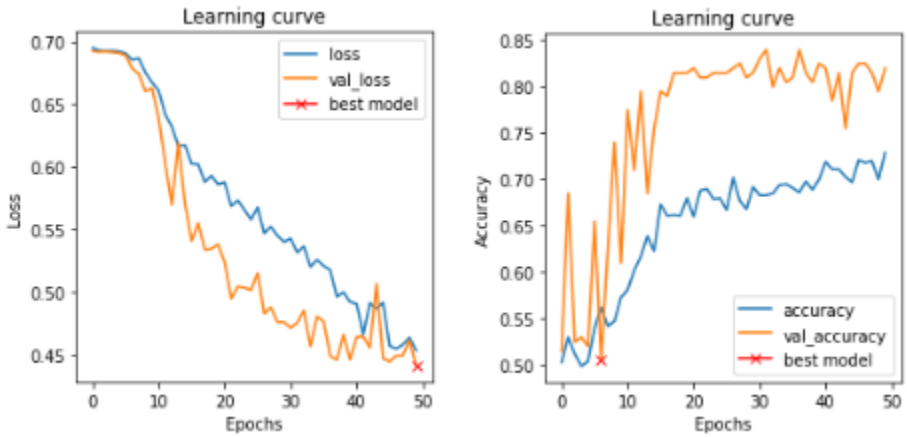
With the dropout layers included we can see that the model performs significantly worse on the training data, so it seems like the dropout layers combat overfitting. The validation accuracy is slightly better than in the last case, and the loss curve does also look more satisfactory.

With the dropout layer included but model trained over 150 epochs instead of 50, it seems to be overfitting again towards the end of the training. So it seems the dropout layers combat overfitting, but not enough when increasing the epochs. We believe this means that tuning the dropout layer should be done in relation to the number of epochs. If one is changed, then the other should be changed too.

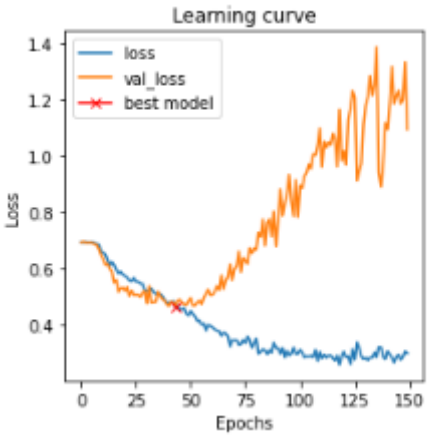
Base 16 w/o dropout, 50 epochs

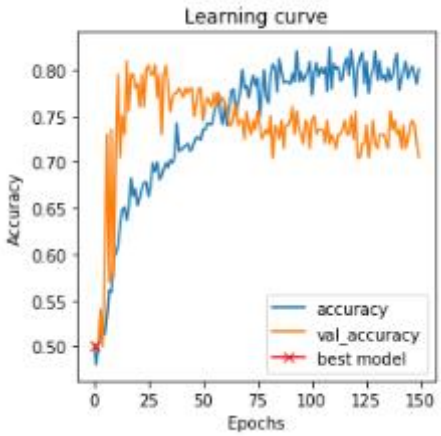


Base 8 w/ dropout, 50 epochs



Base 8 w/ dropout 150 epochs

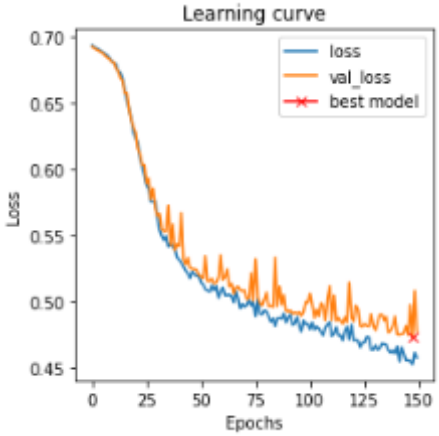


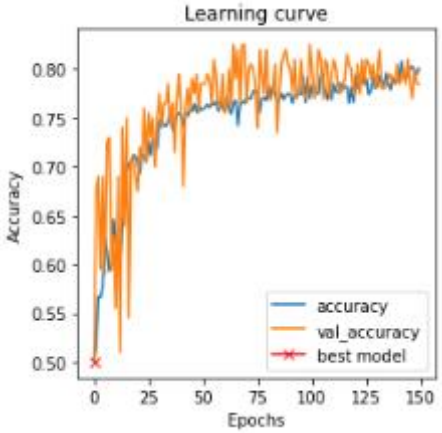


Task 6c *No dropout. Learning rate = 0.00001, base=8. How does LR change performance?*

A slower learning rate means it takes longer for the model to increase its performance. We can however note in this case that overfitting still occurs after increasing the number of epochs; with enough epochs, it seems it will overfit.

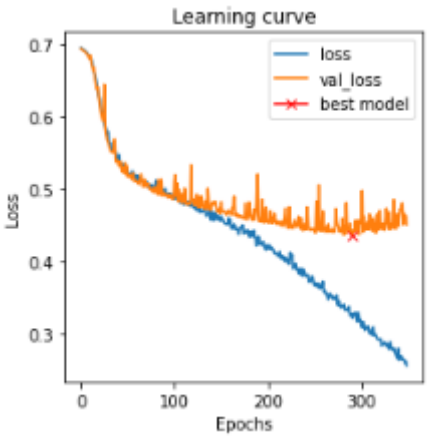
150 epochs

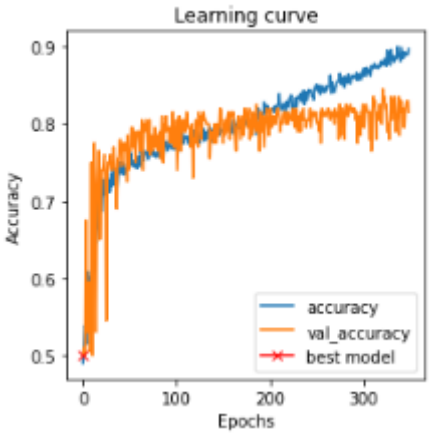




---

350 epochs





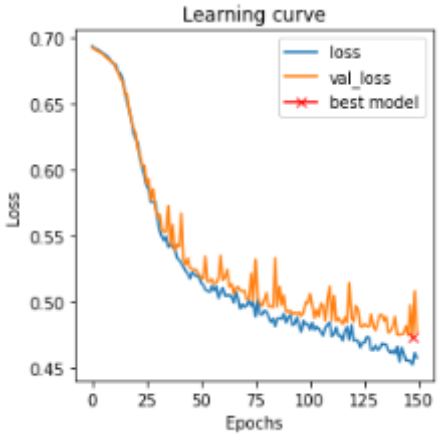
Task 6d *Batch size = 8,4,2. Do you see any significant differences in model performance when changing batch size?*

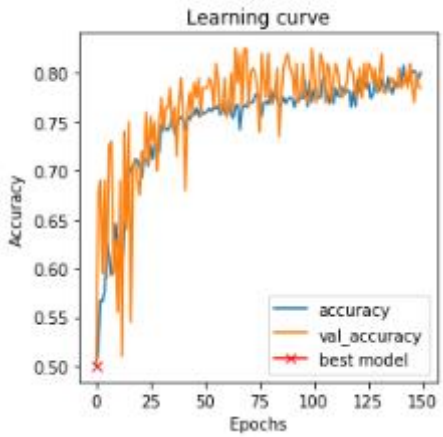
There doesn't seem to be a significant difference between the batch sizes regarding accuracy. The curve for the loss function does however behave a little differetly when the batch size is 2, which we cannot really manage to explain.

It's worth mentioning that every epoch took 1 seconds when the batch size was 4 and 3 seconds when the batch size was 2. So a smaller batch size means that it takes longer time.

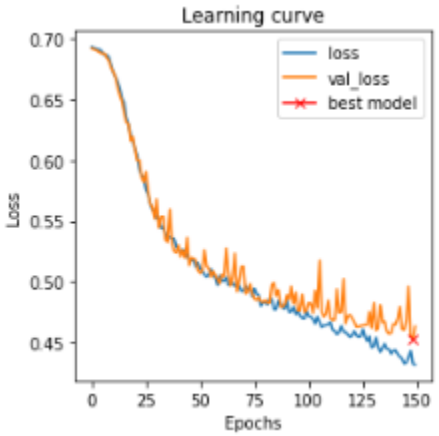
This can probably be explained by the fact that batch is a measure of how many calculations are done in parallel. A higher batch size means more calculations performed in parallel, but requires more memory. We think the end result should be the same, it's just a matter of how long time the process takes. Potential differences in the graphs below should therefore be attributed to the random nature of the initialization of the weights and biases and the shuffling.

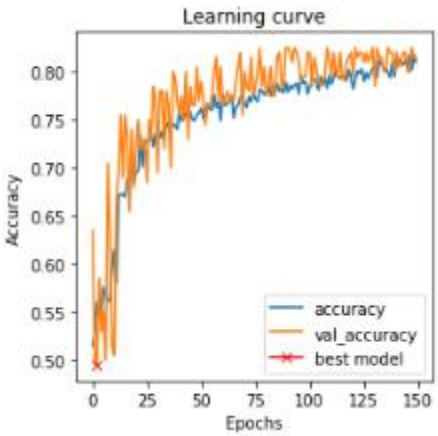
Batchsize 8



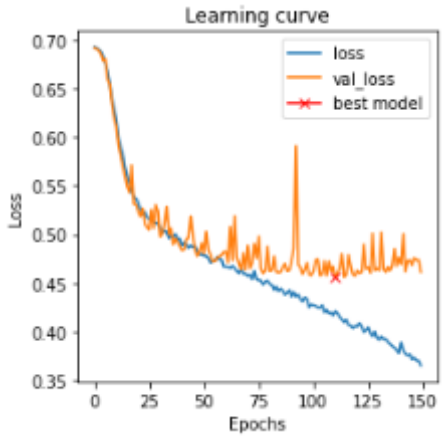


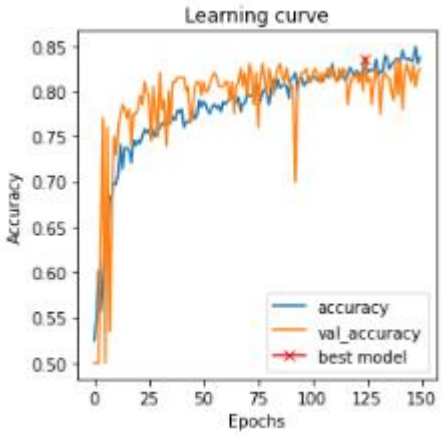
Batchsize 4





Batchsize 2





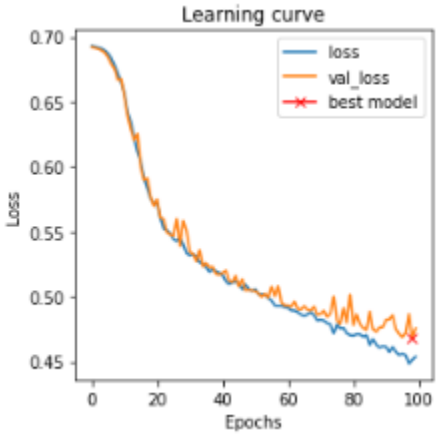
Task 6e, *By finding the optimum values of batch_size, learning rate, and base parameters, train the model for 100 epochs and make sure it is not overfitted. Report the classification accuracy of the model. Then, for this model, only change the optimizer algorithm from Adam to SGD and RMSprop and compare the observed results.*

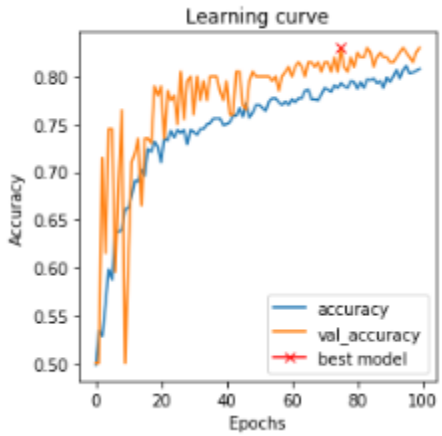
Optimum values: Batch_size=4, base=8, LR=0.00001. Classification accuracy = approx 0.82.

By comparing the three cases we can directly conclude that the SGD performs the worst. The graphs of the other two are quite similar but the SGD is very distinct. The graphs over the loss seem very peculiar as the validation loss is constantly lower. It's however worth noting that the y-axis is very "zoomed in" so they don't differ very much in absolute values.

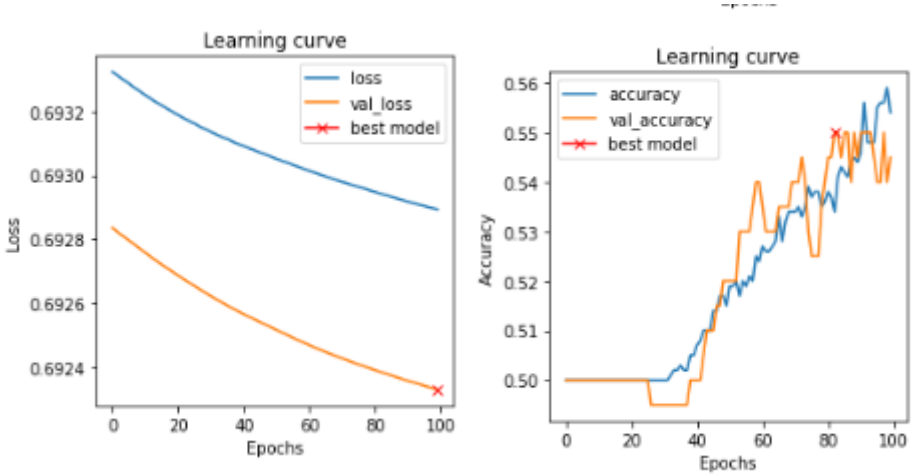
The Adam and RMSprop behave quite similarly and their accuracy are similar.

Optimal values + Adam

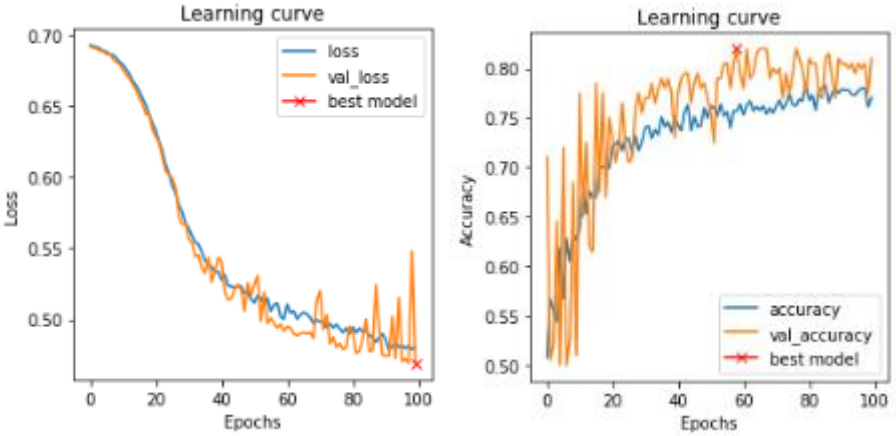




Optimal + SGD



Optimal + RMSprop



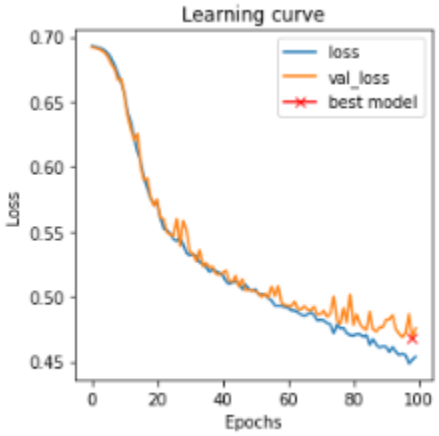
Task 6f,

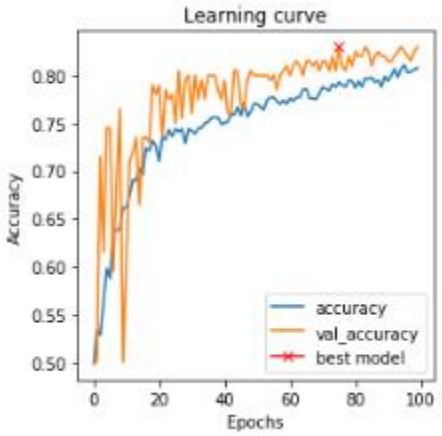
What is the major difference between the "BCE" and "hinge" in terms of model performance?

The most noticeable difference is that the adam has a lot higher accuracy than the hinge. The hinge also seem to stabilize quicker, while the adam keeps improving slightly over epochs.

Adam

Gnana Spandana Akumalla Serkan Arda Yilal Patrik Svensson





Hinge