Gerard Castells and Mariona Carrasco

Deep Learning Methods for Medical Image Analysis (CM2003)

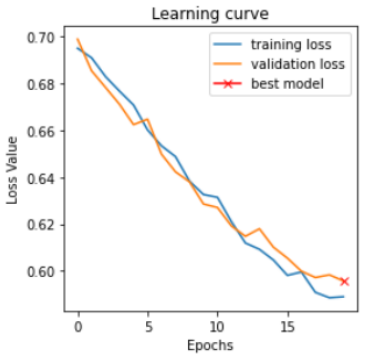
LAB 2

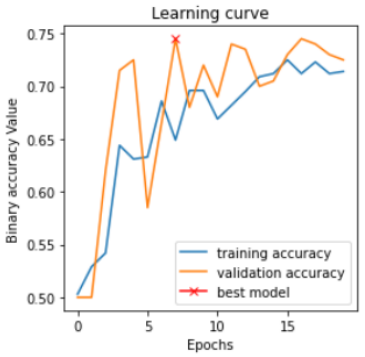
Task 5A

Set the following parameters: # of epochs = 20; batch size = 8; number of feature maps at the first convolutional layer = 32 and learning rate = 0.00001 and then run the experiment. What are the value training and validation accuracies? What can you infer from the learning curves? Is it reasonable to make a decision based on this set-up of the parameters?

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 128, 128, 1)]	0
conv2d_7 (Conv2D)	(None, 128, 128, 32)	320
max_pooling2d_5 (MaxPooling 2D)	(None, 64, 64, 32)	0
conv2d_8 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_6 (MaxPooling 2D)	(None, 32, 32, 64)	0
flatten_2 (Flatten)	(None, 65536)	0
dense_5 (Dense)	(None, 64)	4194368
dense_6 (Dense)	(None, 1)	65

Total params: 4,213,249
Trainable params: 4,213,249
Non-trainable params: 0

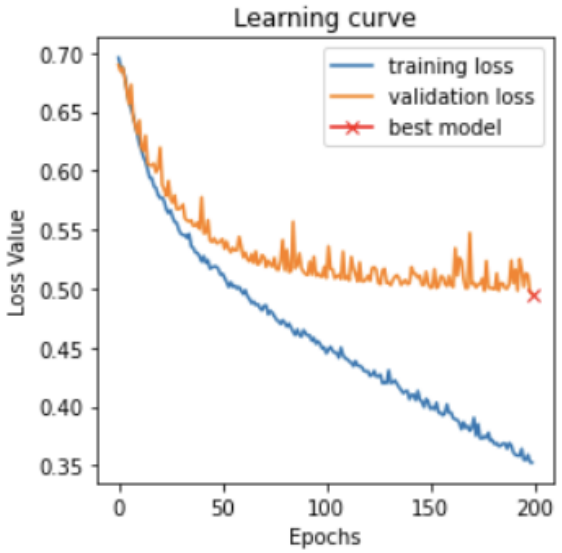


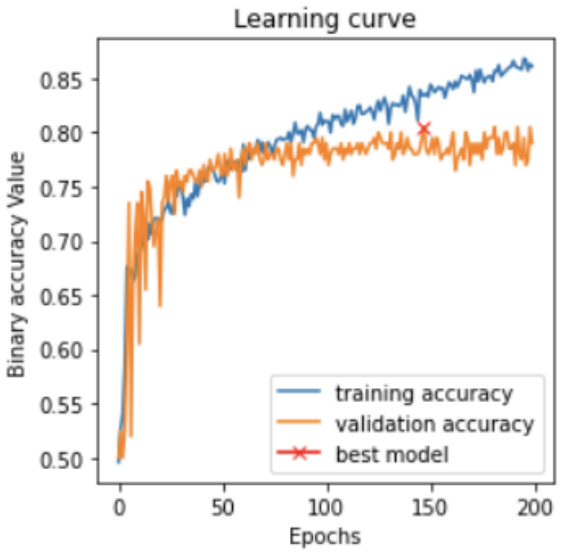


On the graphic, we can see that training and validation curves increase their binary accuracy with more epochs. It indicates that the model is learning from the data and generalizing well. To make a decision based on these learning curves, it should be more accurate and have less training and validation loss.

Task 5B

Leave all the parameters from the previous task unchanged except for the n_epoch = 200. Compare the results with task 5A.

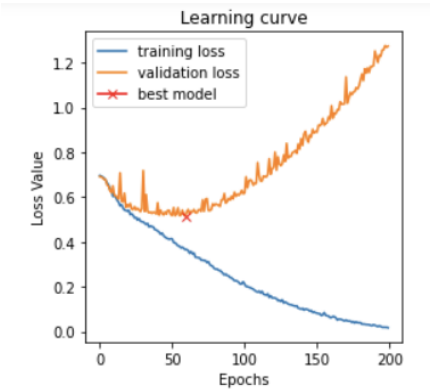


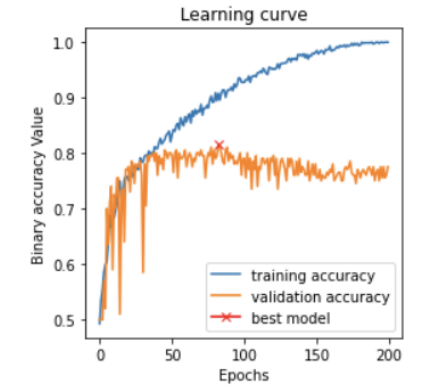


With more epochs, we can see that loss value is lower on training and validation because the model has learned better than on 5A. On the other hand, binary accuracy values get higher to help the model learn better representations from the data and potentially improve performance, with values around 0.8 compared to 0.7 in 5A. However, we can observe that around 100 epochs the validation performance obtained doesn't get much better.

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 could see in that case that overfitting occurs as long as there is a significant gap between training and validation accuracy. Overfitting occurs when a machine learning model learns to fit the training data too closely, capturing noise and idiosyncrasies in the training data rather than generalizable patterns. As a result, the model performs well on the training data (low training loss) but poorly on unseen data, such as the validation dataset. This is a common problem when training models, especially when using a learning rate that is too large.

Here's why a larger learning rate can contribute to overfitting:

- Quick Convergence: A larger learning rate can cause the model to converge quickly, which means it adapts its parameters rapidly to minimize the training loss. This can lead to a very low training loss, indicating that the model fits the training data well.
- Insensitive to Noise: With a large learning rate, the model may be more sensitive to noise and fluctuations in the training data. It can learn to memorize the training data rather than generalize from it, capturing both signal and noise.
- Limited Generalization: The model may not generalize well to unseen data because it has essentially "memorized" the training data and does not have the capacity to generalize to new, unseen examples.

Otherwise, about training and validation accuracy it occurs the same problem with a large learning rate. Training accuracy gets values increasing, but validation accuracy after a certain point gets stabilized because the model starts overfitting the training data.

Task 5D

What is the role of the first two convolutional layers?

The primary role of convolutional layers is to extract relevant features from the input data. In our code, the two convolutional layer uses a 3x3 kernel to convolve over the input image. This means it scans the input image with a small filter to detect patterns and features. Each convolutional filter in the layer learns to recognize specific features like edges, corners, textures, or more complex patterns.

Task 5E

What is the role of the last two dense layers?

These dense layers play important roles in the final stages of feature extraction and classification. They are responsible for aggregating the learned features from the earlier layers, introducing non-linearity, reducing dimensionality, and making the final classification decision.

Task 5F

What is the major difference between the LeNet and MLP?

LeNet and MLP (Multi-Layer Perceptron) are two distinct types of neural network architectures, and their major difference lies in their architectural design and purpose.

LeNet includes specialized layers like convolutional layers and max-pooling layers that are designed to capture spatial features and reduce dimensionality in grid-like data. However, MLP consists of fully connected layers, where each neuron in a layer is connected to every neuron in the next layer.

LeNet is a type of CNN designed for image processing tasks, while MLP is a more general-purpose neural network architecture used for a wide range of data types and tasks. LeNet's specialization in handling grid-like data, such as images, makes it particularly effective for image classification and related tasks.

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?

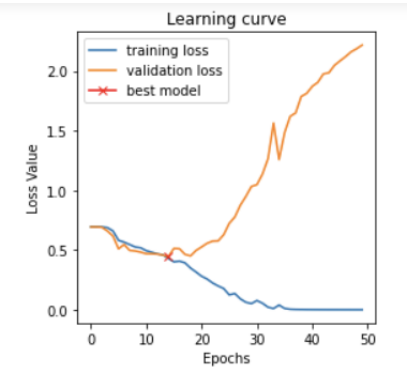
The choice of the number of neurons and the activation function in the last layer depends on the specific task you are trying to solve.

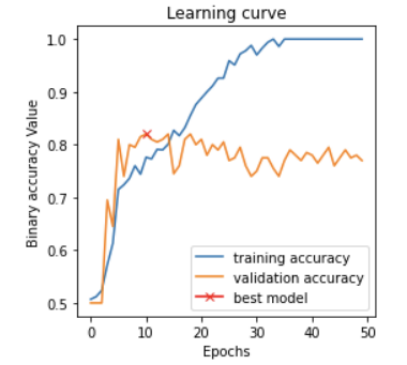
The last layer on our code is: Dense(1, activation='sigmoid'). So, It belongs to binary classification tasks, having a single neuron in the last layer is common. This single neuron can represent the probability of belonging to one class (positive) against another (negative). The sigmoid activation function ('sigmoid') is commonly used for binary classification in the last layer. It squashes the output into the range [0, 1], allowing you to interpret it as a probability. Values close to 0 indicate a prediction for the negative class, while values close to 1 indicate a prediction for the positive class.

However, we could also choose multi-class classification, where the number of neurons in the last layer should match the number of classes. Each neuron represents the probability of belonging to a specific class. The softmax activation function ('softmax') is commonly used in the last layer for multi-class classification. It converts the network's outputs into a probability distribution over the classes, ensuring that the predicted class probabilities sum to. Nevertheless, for regression tasks (where you are predicting a continuous numeric value), having a single neuron in the last layer is typical. The network directly predicts the numeric value. And no activation function is applied in the last layer for regression. The network's output is a continuous value.

Taks 6A

Read the skin images with the size of 128*128 and train the AlexNet model with the following parameter: batch size = 8; epochs = 50; n_base(Base) = 32; learning rate = 0.0001, and Adam as optimizer. Evaluate the model performance.





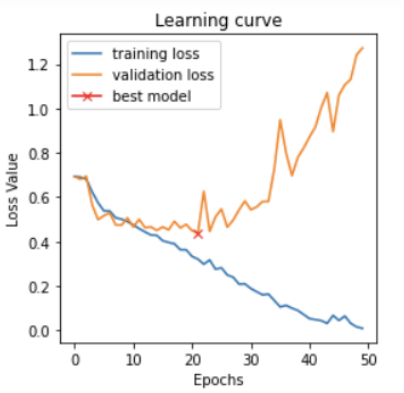
We could see in that case that overfitting occurs, it is a common issue in machine learning when a model becomes too specialized in fitting the training data, capturing noise and idiosyncrasies rather than general patterns. This leads to excellent performance on the training data but poor performance on unseen data like the validation set. This problem can occur with a large learning rate, where the model's training accuracy continues to increase while the validation accuracy plateaus, indicating overfitting.

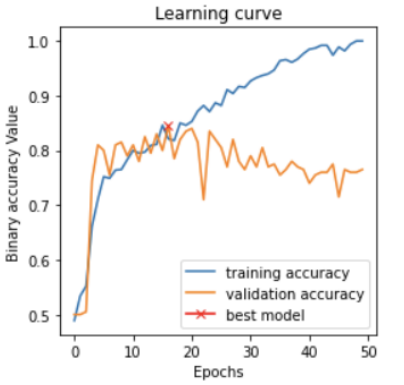
Careful monitoring of the training and validation metrics and adjusting hyperparameters like the learning rate is essential to mitigate overfitting and build more robust machine learning models.

Task 6B

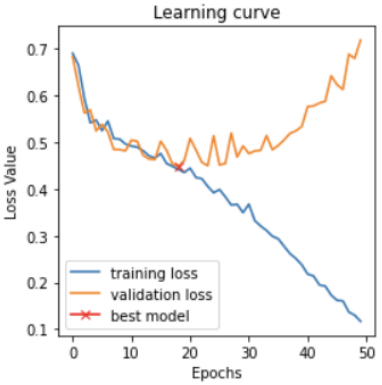
Change the n_base parameter as 16 and 8 and run the model for 50 epochs. How do you interpret the observe results? Now, with n_base = 8, after each of the dense layer add a "drop out layer" with a drop out rate of 0.4 and train the model for 50 epochs. What is the effect of the drop out layer? Increase the number of epochs to 150. How do you explain the effect of increasing the number of epochs?

n_base = 16:





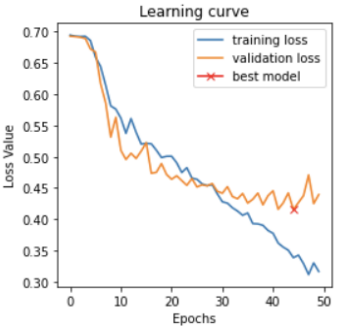
n_base = 8:



The value of n_base determines the number of learnable filters in each convolutional layer, and it often serves as a base value that you can adjust to control the complexity and capacity of your CNN model. By increasing or decreasing n_base, you can experiment with different network architectures and find the configuration that works best for your specific image classification task. In our case we could see that with 8 and 16 of the parameter n_base occurs overfitting. However, with n_base=8 the overfitting is less, so we conclude that it is better to use less learnable filters.

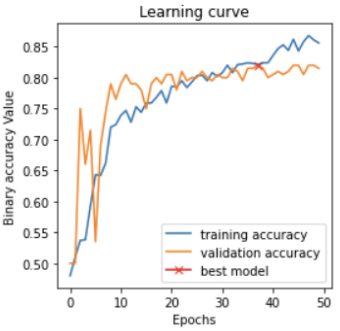
- Drop out layer, 50 epochs and n_base=8:

In addition, by adding a drop out layer we observe that the training accuracy is lower compared to a model without dropout, but the validation accuracy improves, indicating that dropout is helping to reduce overfitting and improve generalization.



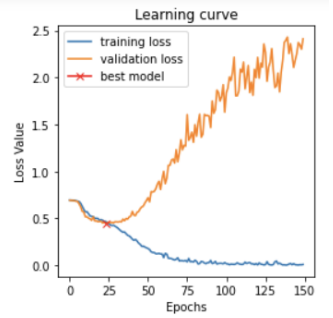
Dropout is a regularization technique that helps prevent overfitting. It does this by randomly setting a fraction of the input units to zero during each forward and backward pass.

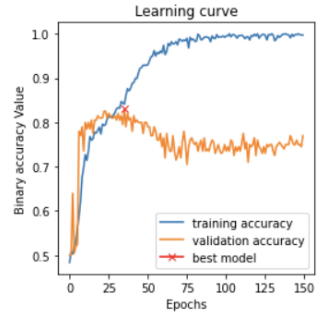
In this case, a dropout rate of 0.4 means that, on average, 40% of the units in the dropout layers will be turned off during training. Which can help prevent overfitting and improve the generalization of our model.



- Drop out layer, 150 epochs and n_base=8:

While more training epochs can lead to better performance on the training data, there's an increased risk of overfitting. Overfitting occurs when the model becomes too specialized in fitting the training data and starts to memorize noise and outliers.That is what happens with 150 epochs in our case.



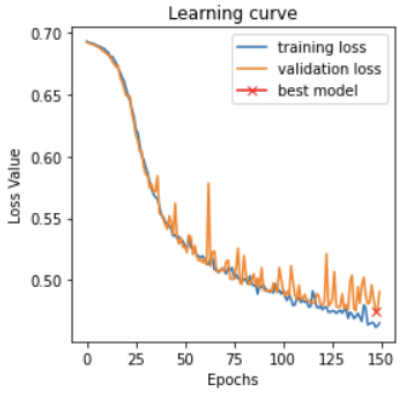


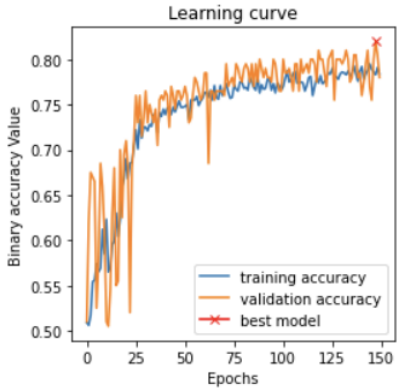
Task 6C

Remove the drop out layers, set the parameters n_base=8, learning_rate = 1e-5 and run the model for n_epochs =150,350 epochs. How changing the learning

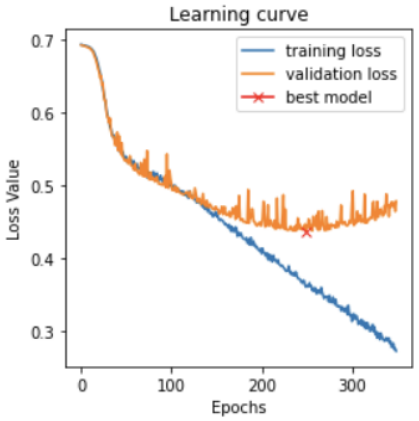
rate parameter affect model performance?

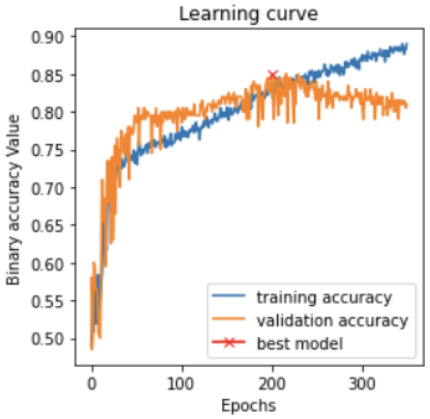
150 epochs:





350 epochs:



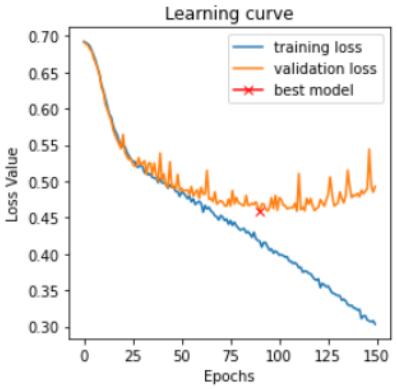


A smaller learning rate of 1e-5 means that the model will make smaller updates to its parameters during each training iteration. This typically results in slower convergence. It may take longer for the model to reach a good level of accuracy on the training data. In that case, we could observe that with fewer epochs (150) we obtain better results than with 350 epochs which converge with overfitting problems like before.

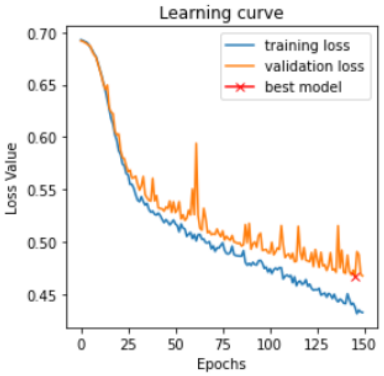
Taks 6D

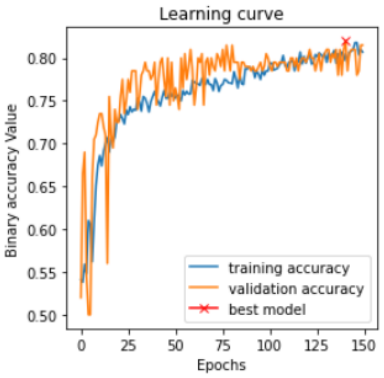
For the fix parameters of learning_rate = 1e-5, n_base=8, n_epochs = 150, change the batch size parameters as 2,4,8. Do you see any significant differences in model performance?

batch size=2

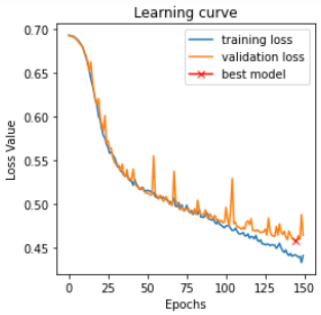


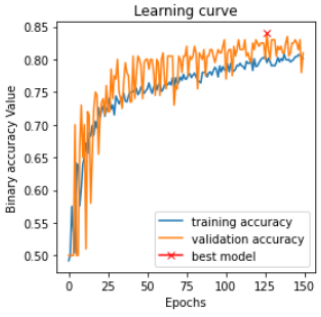
batch size=4





batch size=8





Changing the parameter of batch size which refers to the number of data samples (examples) that are processed together in each forward and backward pass during training. The batch size is a hyperparameter that you can adjust when training a neural network, and it can have an impact on the training dynamics, memory usage, and convergence speed of your model.

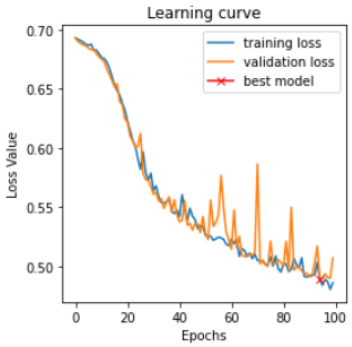
We conclude that a higher value of batch size gives better results so it is more computationally efficient. Because, the accuracy is higher and the loss lower than other batch sizes.

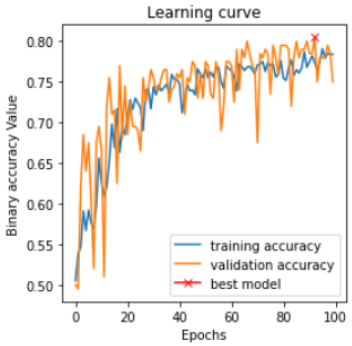
In contrast, with a lower batch size of 2 we could see that overfitting occurs which is not worth it for our training model.

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.

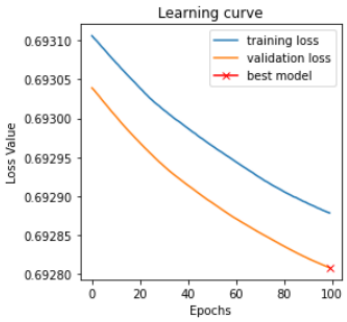
values → learning_rate = 1e-5, n_base=8, n_epochs = 100, batch_size=8

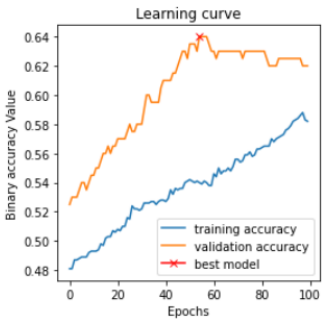




With the optimum values we could make sure that the model is not overfitted.

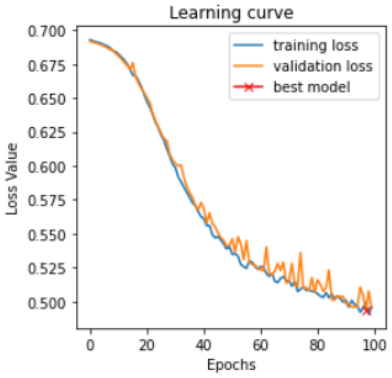
SGD

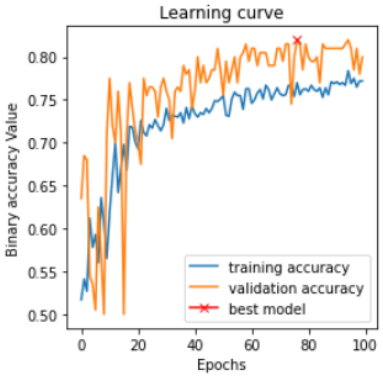




SGD is a classic optimization algorithm that updates the model's parameters based on the gradient of the loss function with respect to each parameter. SGD tends to converge more slowly than Adam.

RMSprop

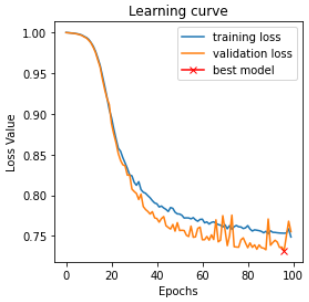


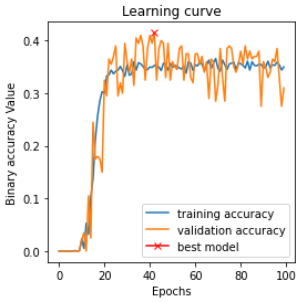


RMSprop is an adaptive learning rate optimization algorithm designed to mitigate some of the issues with SGD. RMSprop often converges faster and more reliably than plain SGD.

Task 6F

"Binary cross entropy (BCE)" is not the only loss function for a binary classification task. Run the code again by changing the loss func into "hinge" with Adam optimizer. What is the major difference between the "BCE" and "hinge" in terms of model performance? Please note, you need to have class labels as [-1, 1] therefore, you need to change the labels as : y_test[y_test == 0] = -1 y_train[y_train == 0] = -1.





Hinge is a loss function designed for binary classification tasks where the goal is to separate data points into one of two classes [-1, 1].

BCE focuses on probabilistic classification and encourages the model to provide calibrated probability estimates for each class. In contrast, hinge loss prioritizes margin-based classification and encourages the model to find a decision boundary that maximizes the margin between classes.