Gerard Castells and Mariona Carrasco

Deep Learning Methods for Medical Image Analysis (CM2003)

LAB 3

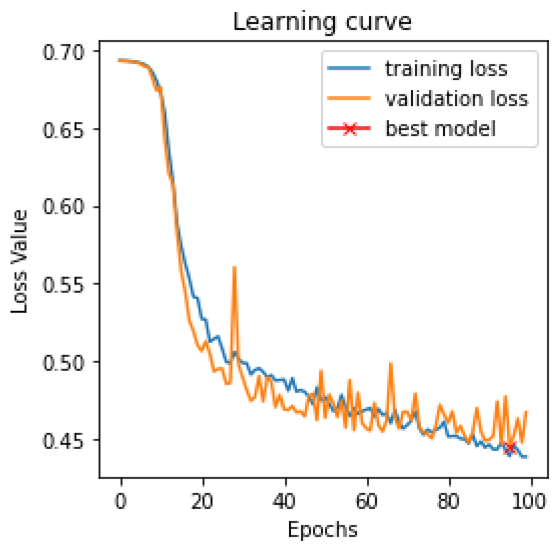
Task 1A

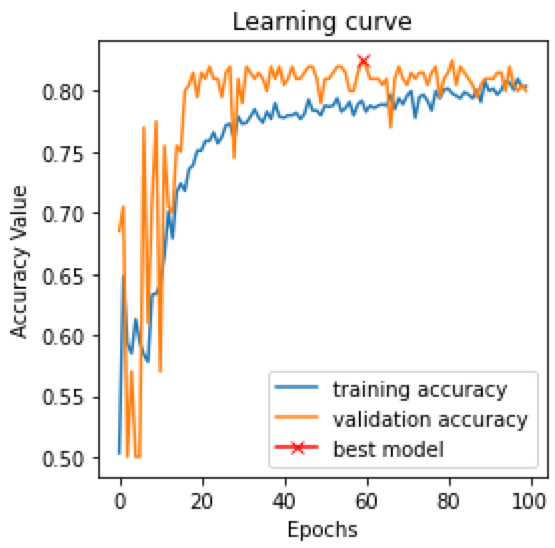
The purpose of this task is to implement an architecture from scratch. You have to implement the architecture of the VGG16 model, as shown in the following figure. Please note 3×3 represents the size of the convolutional kernels, "Base" shows the number of feature maps, "pool/2" indicates the max-pooling operator, and "fc 64" means fully connected (dense) layer with 64 neurons.

Done in the code

Task 1B

Load the Skin data set with the size of 128*128. For the following fixed parameters, compile the model and find the optimum value of the learning rate that will lead to reliable classification performance over the validation set. What is the proper learning rate? n_epoch = 100, batch_s = 8, n_base = 8





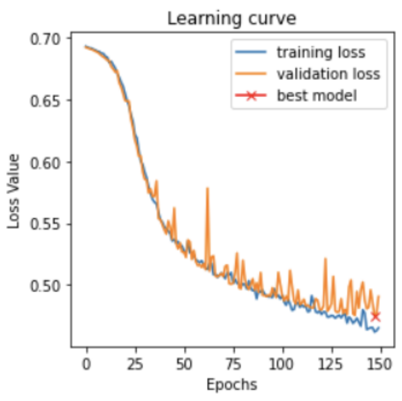
The proper learning rate is 1e-5 as we can see with the learning curve.

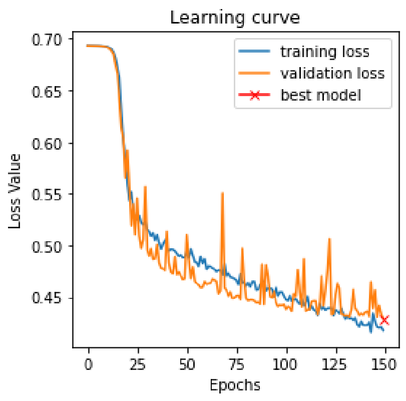
Task 1C

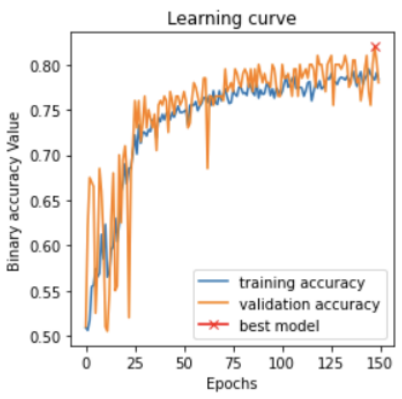
For the same setting as task 6C (only for n_epoch=150) compare the classification accuracy over the validation set between the AlexNet and VGG16 models. How do you justify the observed results?

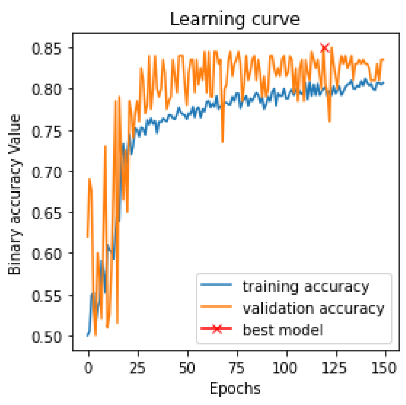
AlexNet

VGG16









We can see that the performance with VGG16 model is better than with AlexNet, with loss value around 0.45 and higher accuracy (0.85).

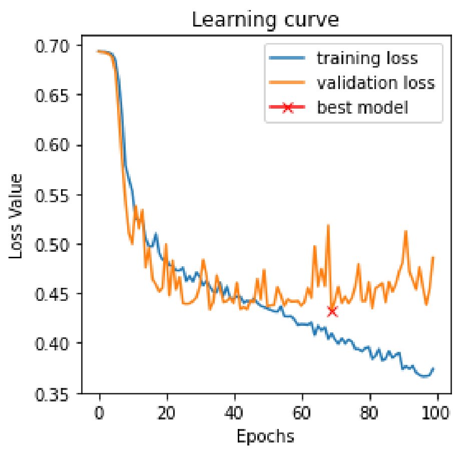
The fact that the VGG16 architecture works better is due to the following reasons. VGG16 has a deeper architecture with 16 weight layers (13 convolutional and 3 fully connected) compared to AlexNet's 8 weight layers (5 convolutional and 3 fully connected). Deeper networks often have more capacity to learn complex features and representations, which can be advantageous when dealing with intricate patterns in data. This increased depth in the network results in a larger number of model parameters. Having more parameters allows the network to learn a richer set of features, potentially improving its ability to discriminate between classes.

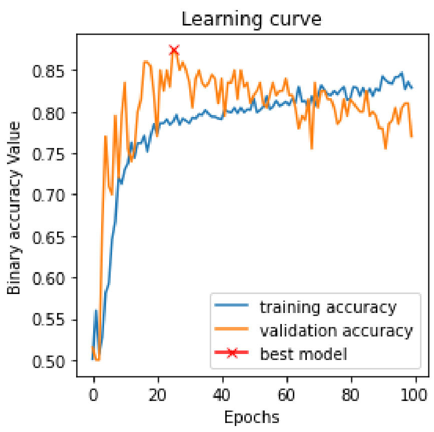
Also, VGG16 uses smaller 3x3 convolutional filters throughout its architecture, whereas AlexNet initially uses larger 11x11 and 5x5 filters. Smaller filters are more effective at capturing fine-grained details and patterns in the data, which can be beneficial for many image classification tasks.

Task 1D

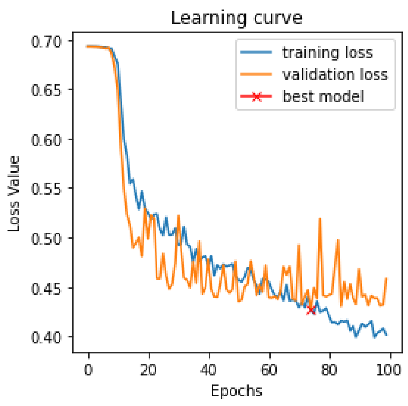
Change the parameter n_base to 16. Train the model for 100 epochs with LR = 1e-5 and batch size of 8. Does increasing the number of feature maps affect model performance? Then, add a dropout layer with a rate of 0.2 for each of the dense layer. What was the effect of adding the dropout layer?

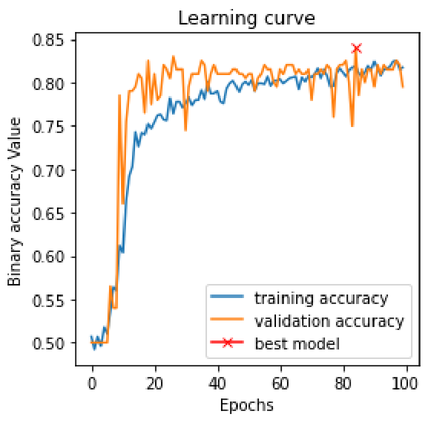
n_base=16





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, with n_base=8 we don't have overfitting, so we conclude that it is better to use less learnable filters because increasing the number of feature maps affects the model performance worse.





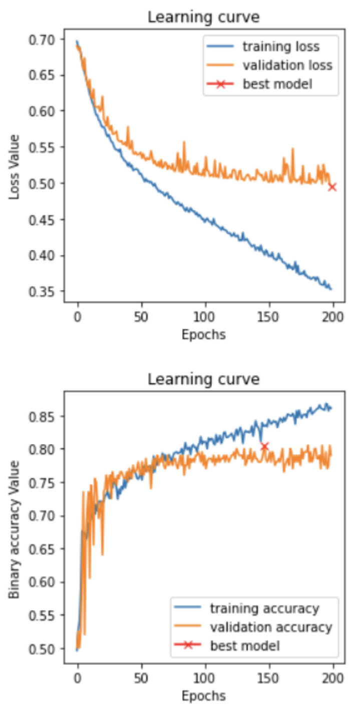
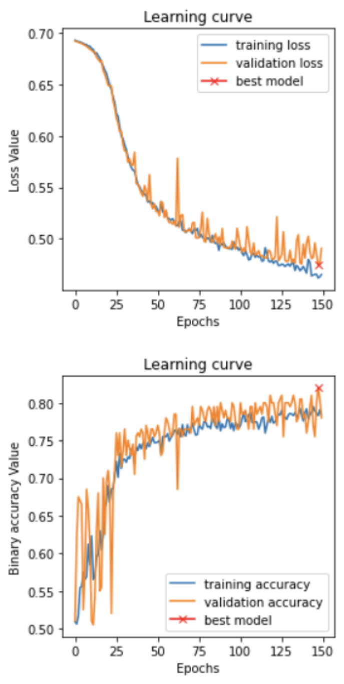
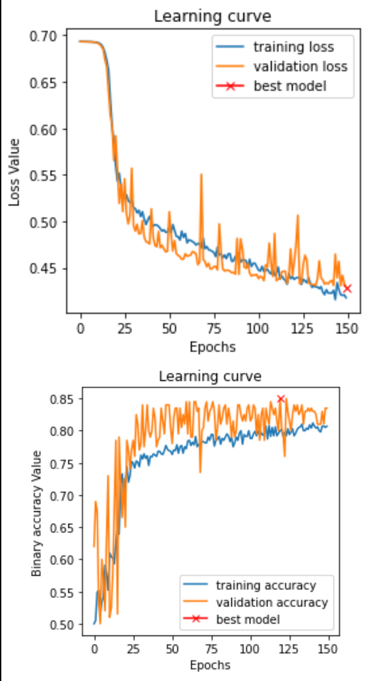
In addition, by adding a drop out layer we observe that the training accuracy is lower compared to a model without dropout, but the validation accuracy improves, indicating that dropout is helping to reduce overfitting and improve generalization. Dropout is a regularization technique that helps prevent 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 0.2 means that, on average, 20% 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.

Task 1E

So far, you classified the Skin cancer dataset with 3 different models named as LeNet, AlexNet as well VGG16. In general, what is the difference between these three models? Which of them yields more accurate classification results? Why? To evaluate the model performance, how do you assess the loss values? How can you prevent the model training from overfitting?

On the table we can see the 3 different models.

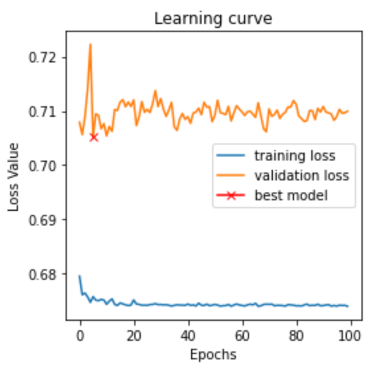
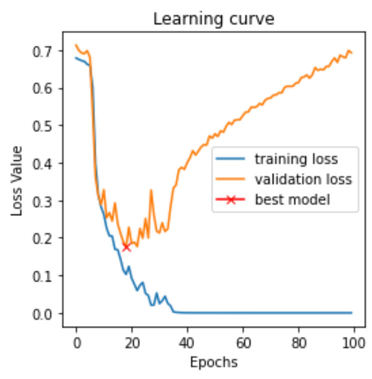
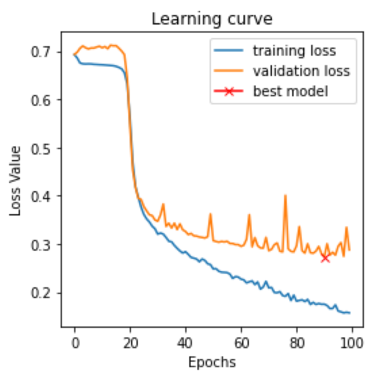
LeNet	AlexNet	VGG16
		

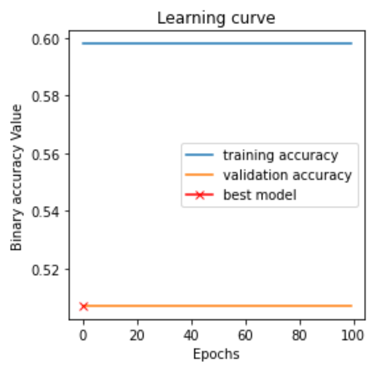
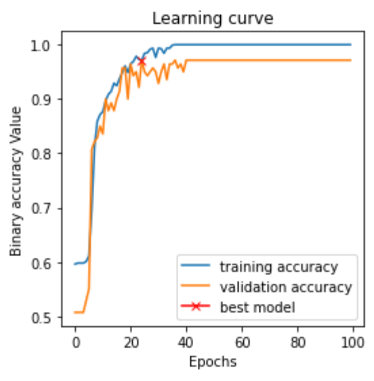
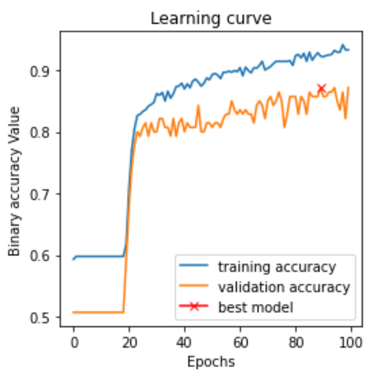
The main difference between these 3 models is the number of convolutional layers. LeNet is made of 2 convolutional layers, AlexNet has 5 and VGG16 has 13. This makes that the bigger the number of convolutional layers is, more parameters are taken into account for the classification task. In consequence, VGG16 has more accurate results and less loss value than the other 2 models.
Loss values are reduced as the number of parameters used in the task increases. That's because the overall performance of the architecture is improved which results in better training.

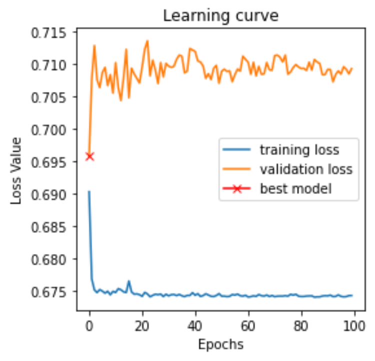
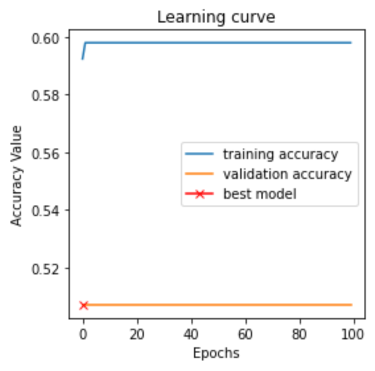
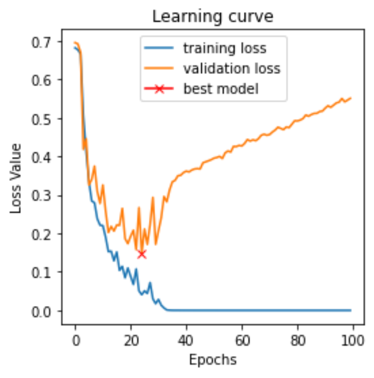
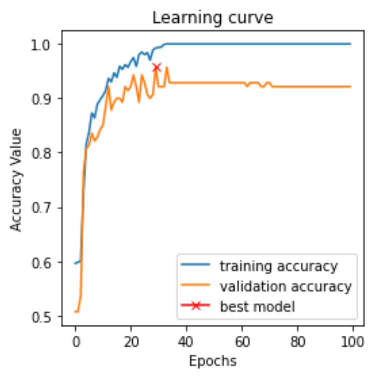
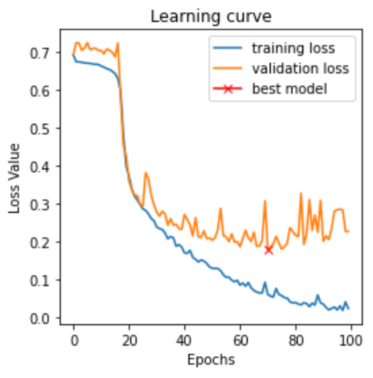
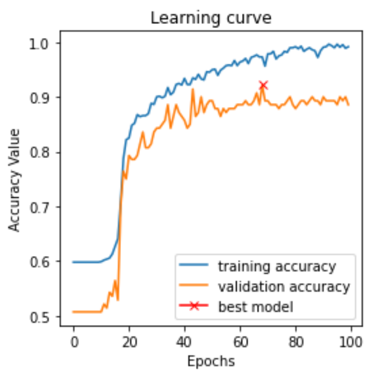
In order to prevent the model training from overfitting and improve the generalization of our model we should put a dropout layer with a specific rate for each of the dense layers.

Task 2

Train the VGG16 model developed in Task 1 with the following parameters to classify two types of bone fractures from the "Bone" dataset. Please note the size of the Bone images is quite large, so that it would take a longer time to read and load all the images. data_path = '.../Lab1/Bone/' img_w, img_h = 128, 128 replace the pattern1 and pattern2 in gen_label function with 'AFF' and 'NFF' Loss function: BCE, Optimizer: Adam, n_base: 8 & 16, n_epoch: 100, batch_s = 8 Choose an appropriate learning rate for the task, a suggestion is to try values in the range 1e-3, 1e-4, 1e-5 and plot loss plots to evaluate.

learning rate=1e-3 n_base=8	learning rate=1e-4 n_base=8	learning rate=1e-5 n_base=8
		

learning rate=1e-3 n_base=16	learning rate=1e-4 n_base=16	learning rate=1e-5 n_base=16
		

We observed that the learning rate of 1e-5 is the one that gives us the best results, because it produces the least overfitting. With learning rates of 1e-3 and 1e-4 we can observe that the validation loss is noticeably higher than the training loss. With learning rate bigger than 1e-5 we can observe that noticeable overfitting appears.

Comparing the performance of the model with a learning rate of 1e-5 and 8 and 16 as a base number, we can observe that the learning curves follow a similar pattern but with better accuracy and loss values when having n_base=16. This means that the model needs a bigger number of learnable filters in each convolutional layer to adjust the model network. It can be due to the fact that Bone images are bigger, so they require more parameters in order to be properly classified.

Task 3

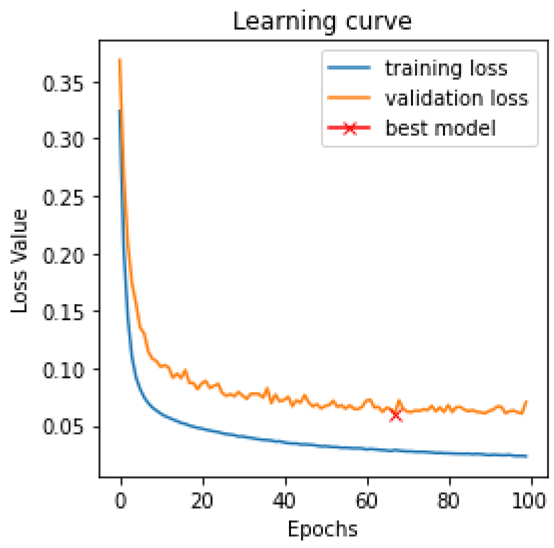
With the implemented VGG models, which of the Skin/Bone image sets classified more accurately? Why? How do you make sure that achieved results are reliable?

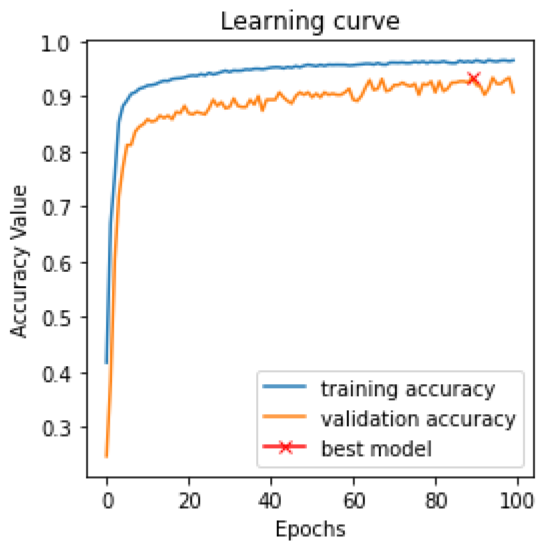
VGG models is more accurate with bone images because we obtain less loss values and higher accuracy

Task 4

In previous exercises, you conducted some experiments with binary classification tasks by implementing three deep networks named as LeNet, AlexNet, VGG. In this task, 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 load the images along with their class labels properly. Extend the LeNet and AlexNet models for multi class classification tasks. Tune these two models by finding the optimum values of hyperparameters to get the best performance for each of the models and, then, compare the observed results between the two models. Report the learning curves for both of the loss and accuracy values (for train and test data). Data path is: '.../Lab1/X_ray/'.

LeNet model: base=16, batch_size=8, epochs=100, learning_rate=1e-5.

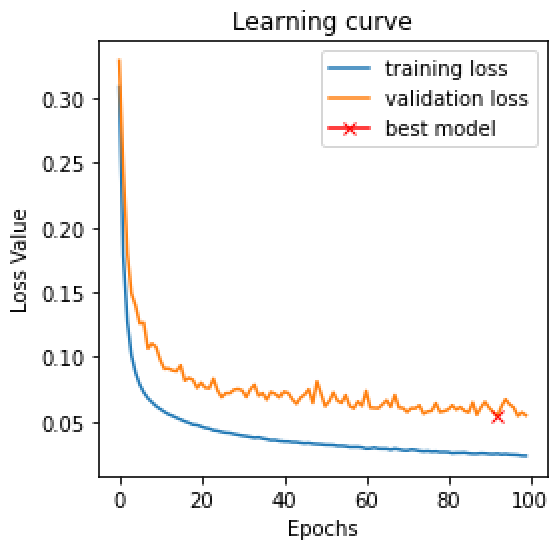


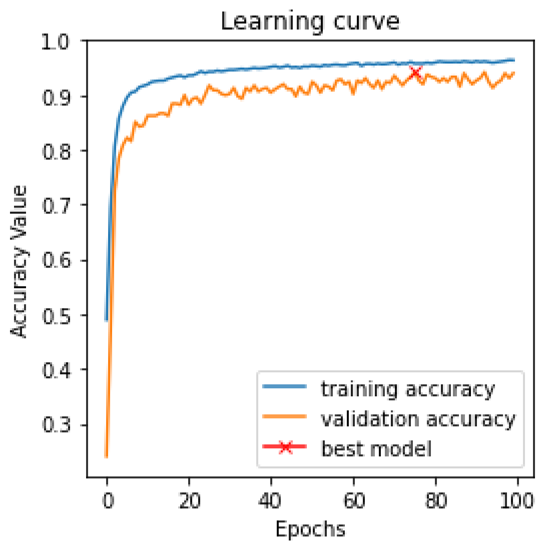


By choosing a base number of 16, we get to have 1.053.705 parameters which help us in the classification task as long as the database is big and there are 9 different output options. Using 8 as the batch size, we prevent it from overfitting as not too many training images are taken into account in each step. The learning rate of 1e-5 gives a nice result avoiding overfitting while converging fast enough.

We can see that the LeNet model offers an accuracy of around 0.92 and loss values of around 0.07, which can be considered as satisfying.

AlexNet model: base=16, batch_size=8, epochs=100, learning_rate=1e-5.



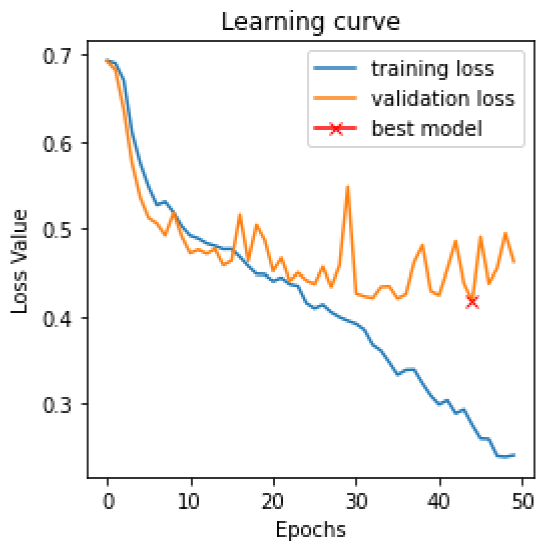


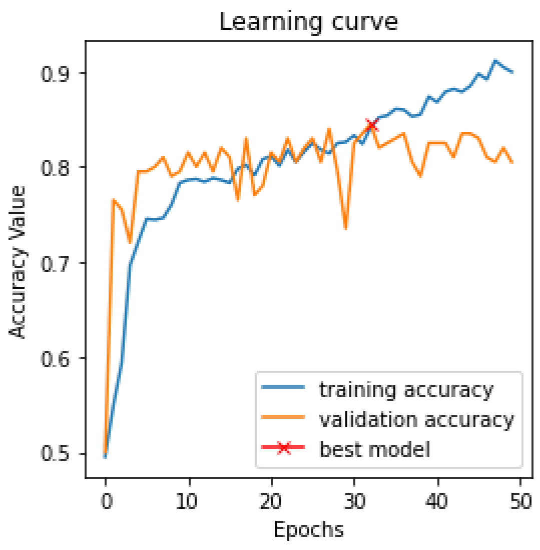
final val loss= 0.07, final val acc= 0.93

Regularization Techniques

Task 5a

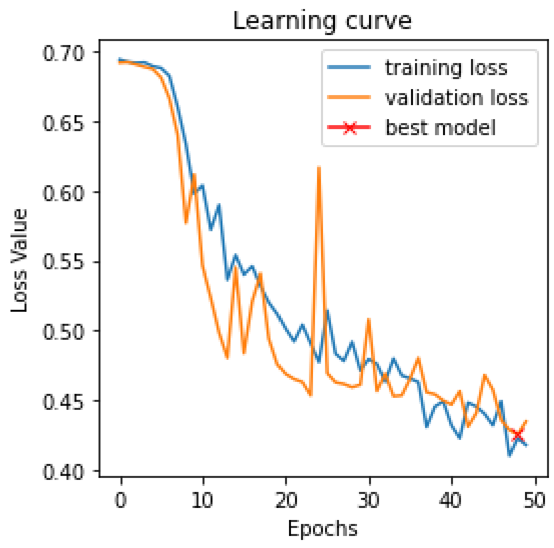
Employ the AlexNet model with the following architecture: five convolutional blocks [with feature maps of the size of base, base*2, base*4, base*4, base*2 where base=8], followed by three dense layers with 64, 64, and 1 neuron, respectively. Add three max-pooling layers after 1st, 2nd, and 5th convolutional blocks. Set the following parameters: learning rate=0.0001, batch size=8, 'relu' as activation function, 'Adam' as optimizer, and image size=(128,128,1). Train this model for 50 epochs on skin images. What are the values of the train and validation accuracy? How do you interpret the learning curves?

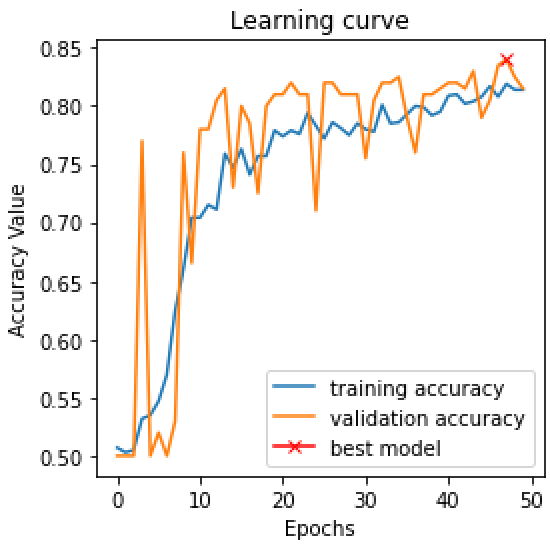




We can observe that validation loss values are high (>0.4) and that overfitting occurs. In relation to the accuracy values, we observe that it gets up to more than 0.8, but the uncorrelation between training and validation is still noticeable.

Add two drop out layers after the first two dense layers with the dropout rate of 0.4 and repeat the experiments and compare the results. What is the effect of adding drop out layers?



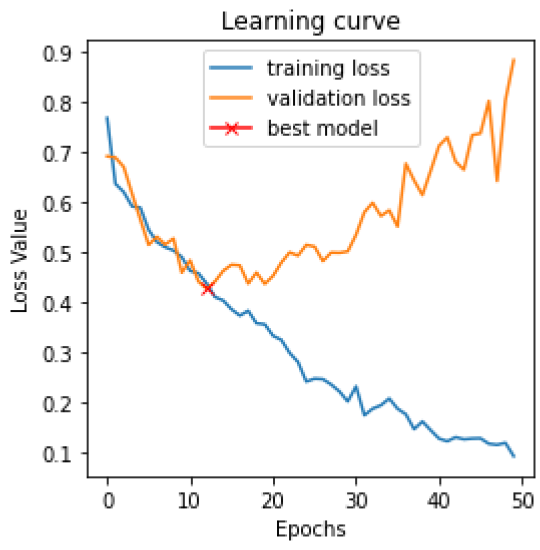


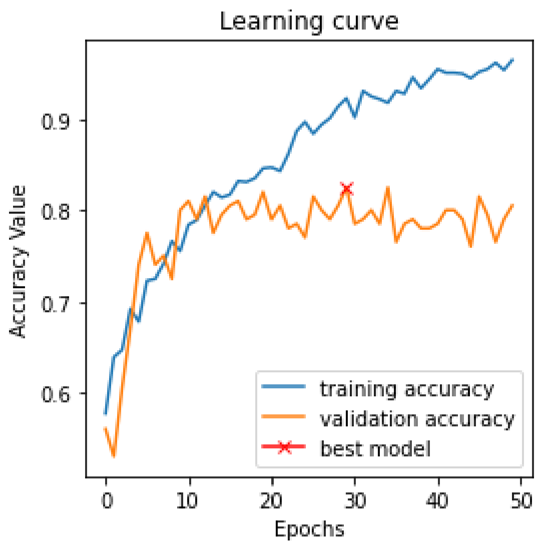
By adding the dropout layers, the accuracy and loss values stay in a similar value, but we can see that overfitting is corrected. We can observe that the validation values follow an up and down parameter which is related to the way the dropout layer performs. In this case, a dropout rate of 0.4 means 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.

Task 5b

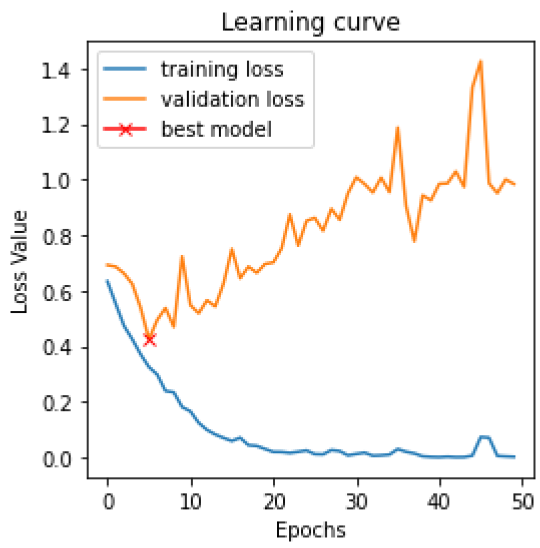
With the same model and same settings, now insert a batch normalization layer at each convolutional block (right after convolution layer and before activation function). At which epoch do you observe the same training accuracy as task (a)? What is the value of final training accuracy? What is the effect of the batch normalization layer? Similar to task (a), do this task with and without drop out layers.

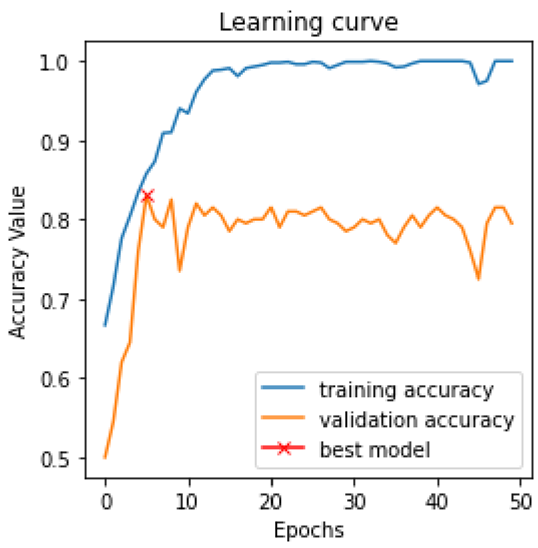
- With drop out layer:





- Without drop out layers



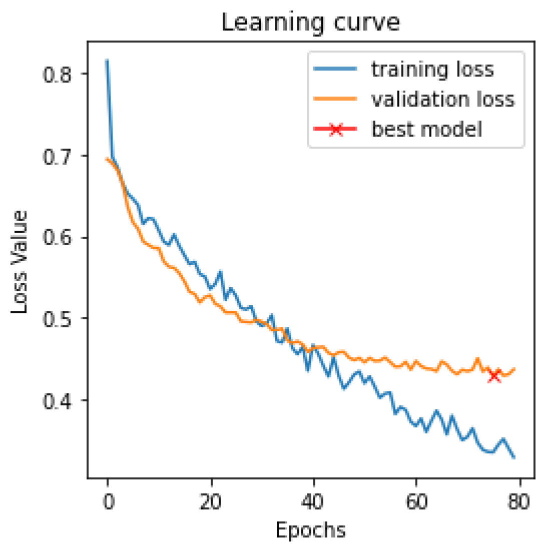


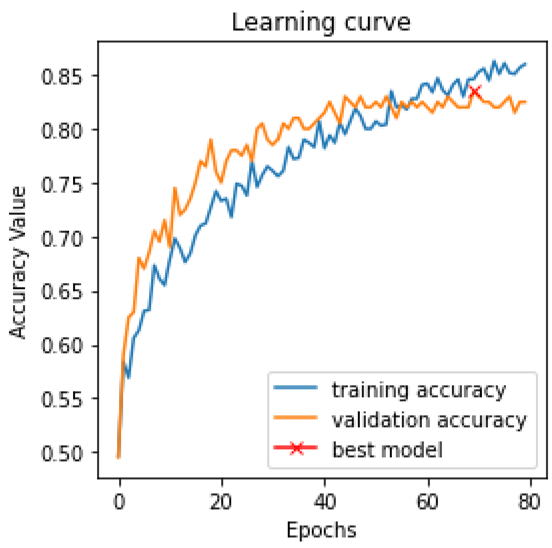
We could observe the same training accuracy as task 5a with less epochs, because batch normalization layer is implemented now and it stabilizes and accelerates the training of deep neural networks. In that case, the value of the final training accuracy is around 1.
However, with the drop out layers it occurs overfitting around 15 epochs but the learning curve follows a more regular pattern. In contrast, without drop out layers, training curves of with few epochs around 5 the model starts overfitting the training data.

Task 5c

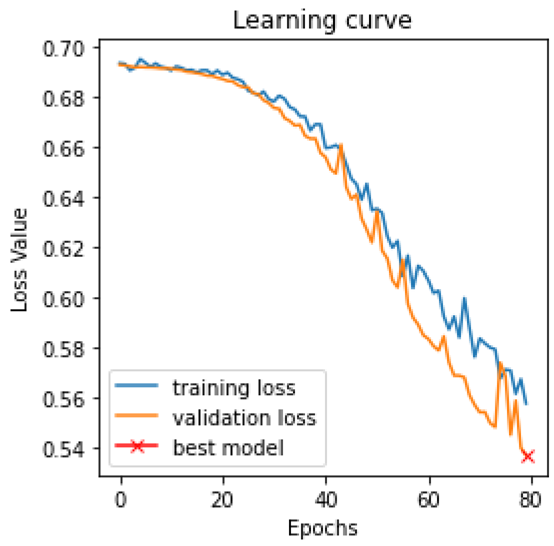
Train again the same model with precisely the same parameters except learning rate = 0.00001 and epochs = 80 with and without batch normalization layers (in both cases, use the drop out layers). Focus on validation loss & accuracy. Which model resulted in higher validation accuracy? How do you explain the effect of batch normalization?

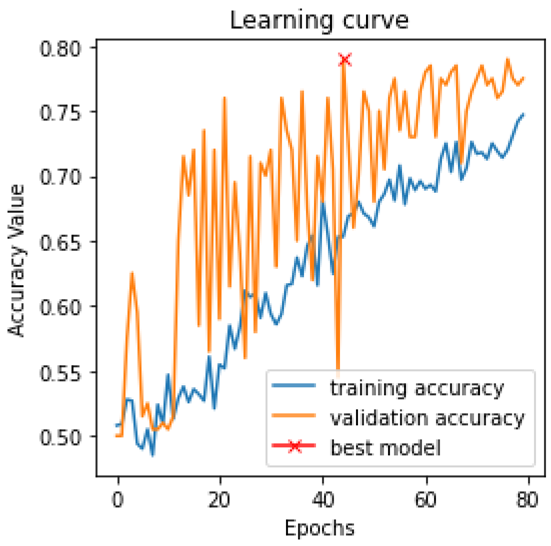
- With batch normalization





- Without batch normalization



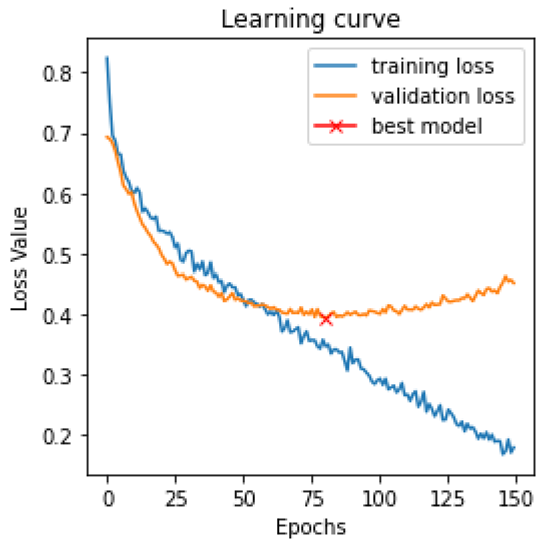


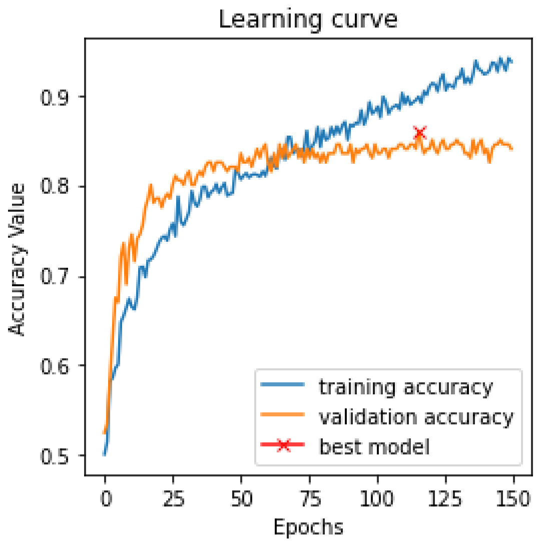
With the batch normalization layers we observed higher validation accuracy around 0.82. In contrast, without batch normalization we only observed 0.77 validation accuracy.
However, the learning curve with batch normalization occurs overfitting, it follows a more regular pattern. To conclude, the appearance of overfitting in the training curve does not necessarily imply that the model is not generalizing well. In many cases, it's a sign that the model has learned the training data more deeply and is capable of fitting to the training set more closely.

Task 5d

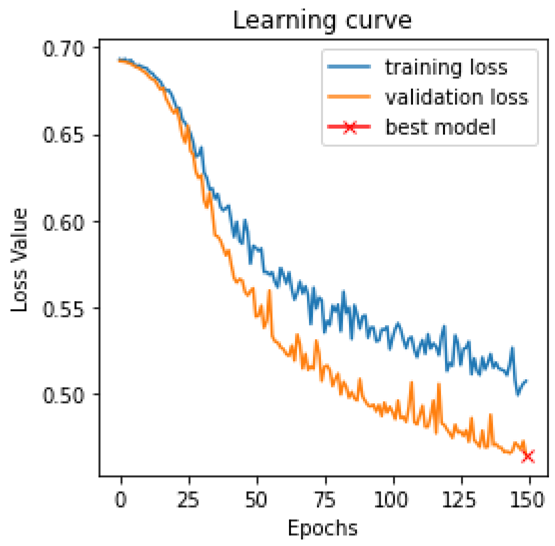
Keep the settings from the task1c unchanged and train the model for 150 epochs with and without batch normalization layers (in both cases, use the drop out layers). Which model yields more accurate results? Which of them has more generalization power?

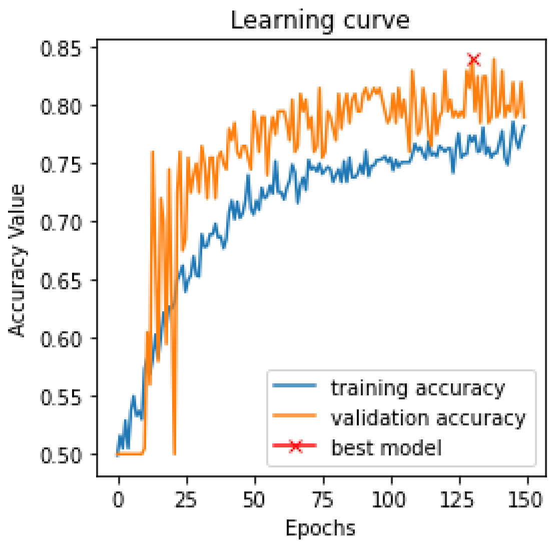
-With batch normalization





- Without batch normalization





Implementing the batch normalization layers the learning curve follows a more regular pattern, even though overfitting occurs from around 75 epochs. Since this point loss value starts to increase and accuracy doesn't get better. The best accuracy and loss values are 0.4 and 0.83, respectively.

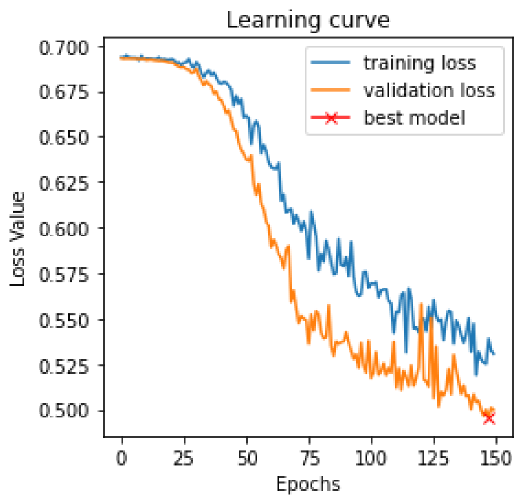
When performing without the batch normalization layers, overfitting does not occur, as long as the performance follows an increasing tendency, obtaining better values to more epochs done. Validation loss values finish in 0.46 and accuracy in 0.83, which is very similar to without the batch normalization layer.

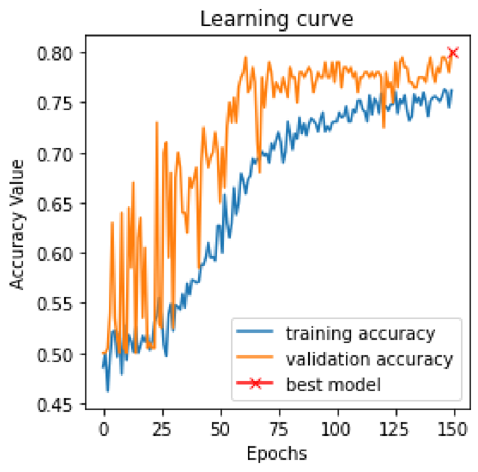
We can conclude, observing the results, that applying the batch normalization layers helps the model to generalize better, as long as the learning curve for the validation data gets better values than the training one.

Task 6a

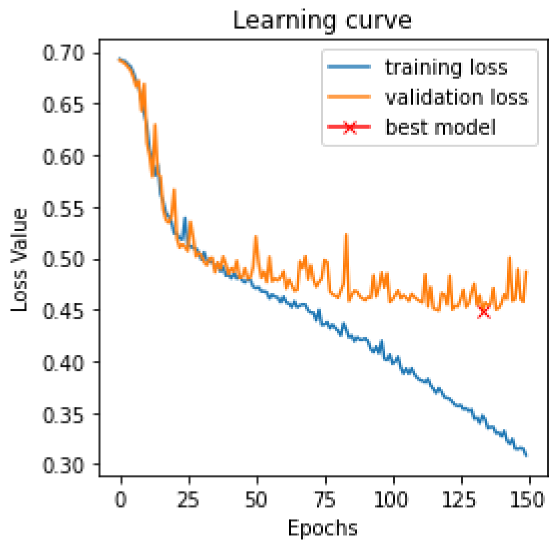
Use the same model as task1d but set the "base" parameter as 32 and replace the batch normalization layers with spatial dropout layers at each convolutional blocks (after activation function, and before max-pooling). Set the dropout rate of spatial drop out layers as 0.1 and the rate of 0.4 for the normal drop out layers after the first two fully connected layers. Then let the model runs for 150 epochs with LR=0.00001. Save the loss and accuracy values for the validation data. Then, run the same model with the same settings but remove all the spatial drop out layers. Which of them converges faster? Why?

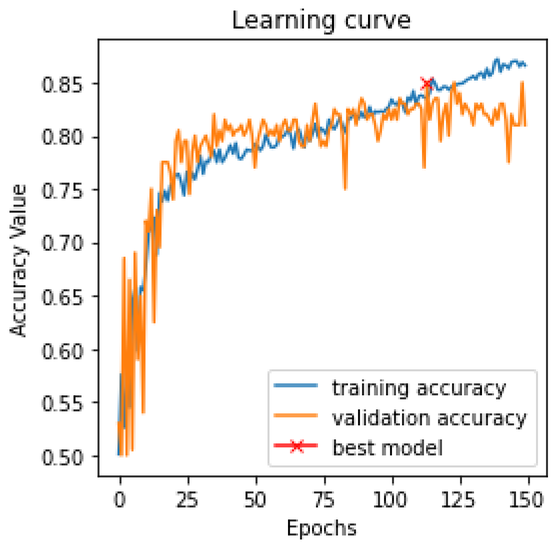
- With spatial dropout





- Without spatial dropout



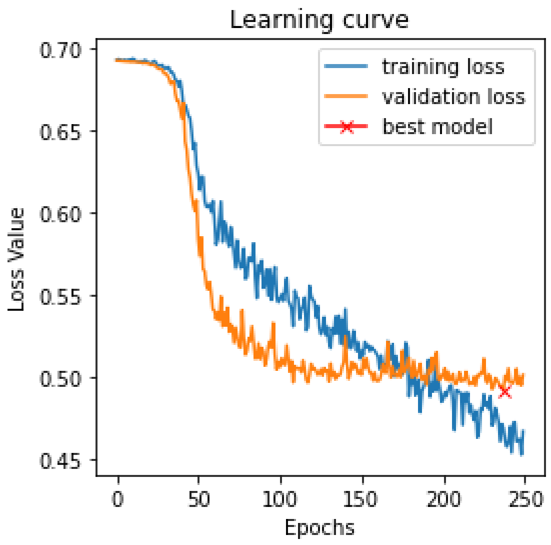


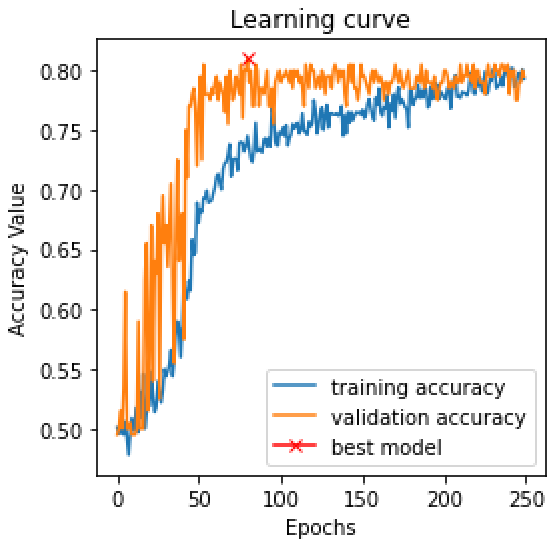
As we can observe with the learning curves, when we have the spatial dropout layer it converges faster. This occurs because spatial dropout layers deactivate entire feature maps for a given input example. This can help the network retain more spatial information, as important patterns or features aren't completely lost in the dropout process. So, the network is forced to rely on a wider variety of features during training, potentially making it more robust and less prone to overfitting. Consequently, it may require fewer training epochs to converge to a good solution and can lead to faster training times.

Task 6b

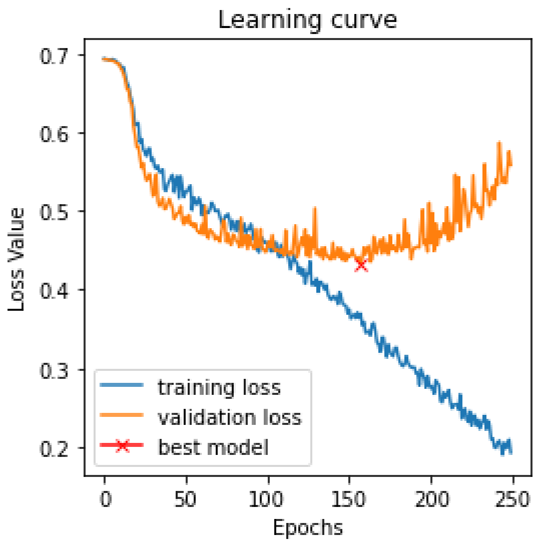
Repeat the task2a for 250 epochs with and without spatial dropout layers. In general, discuss how the drop out technique (spatial and normal one) would help the learning procedure.

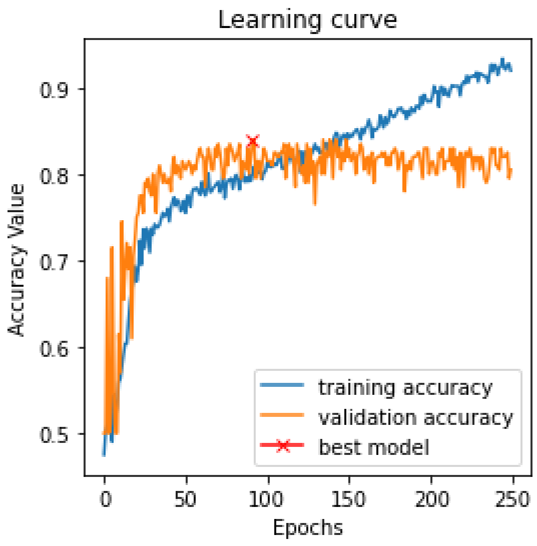
- With spatial dropout





- Without spatial dropout





We observe that when the spatial dropout layer is applied the model converges fast to its best performance values and then it doesn’t improve. Validation loss ends up in a value around 0.5 and accuracy around 0.8.

When spatial dropout is used, the model generalizes better as the validation values are more similar to the training ones. Without spatial dropout overfitting occurs.

Data Augmentation

Task 7a

Read the following set of codes and find out how they work. Then, change the following parameters: scale_factor, Angle, low/high bounds to see their effects.

After making these parameter changes, we can observe how each data augmentation operation is affected, and see different results for rescaling, rotation, and intensity rescaling based on our chosen values. This experimentation helps to understand how these transformations impact the appearance of the image data.

- Scale Factor → A smaller value will shrink the image, while a larger value will enlarge it.
- Angle → Modify the rotation to a different angle in degrees.
- Low/high bounds → Modify the intensity rescaling with different percentiles to control the range of intensity values used.

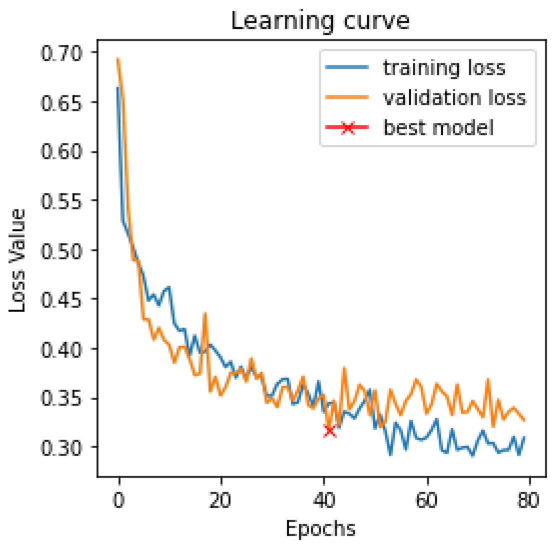
Task 7b

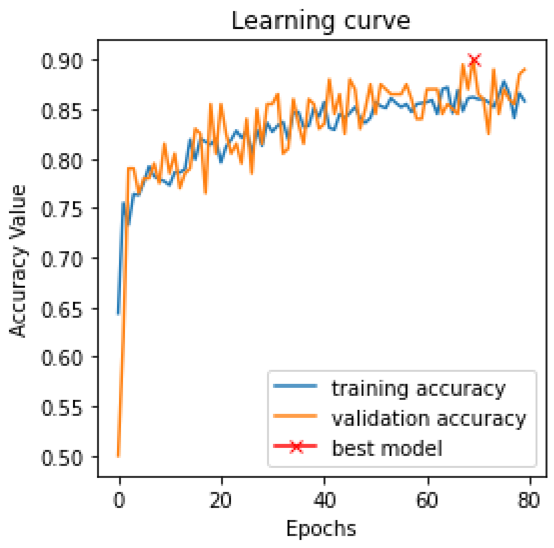
A practical way to perform the data augmentation technique is to develop a generator. The following code is an example of how you can generate augmented images randomly with a TensorFlow built-in generator.

The provided code demonstrates how to use the Keras ImageDataGenerator to perform data augmentation on an input image. Data augmentation is a technique used to artificially increase the size of your training dataset by applying random transformations to the original images to improve the generalization of our deep learning model.

Task 8

Develop a framework for training the models with augmenting the training data as:
The augmentations that are going to be applied over the training data include: rotation_range=10, width_shift_range=0.1, height_shift_range=0.1, rescale=1./255, and horizontal_flip=True.
However, for validation data, you just need to rescale=1./255 the images.
Use the AlexNet model with the batch normalization layers, and drop out layers for the first two dense layers(rate=0.4). Set the “base” parameter as 64, and assign 128 neurons for the first dense layer and 64 for the second one. Set the optimizer=Adam, LR=0.00001, batch-size=8, and train the model on skin images for 80 epochs. How the data augmentation impact model training? Why?





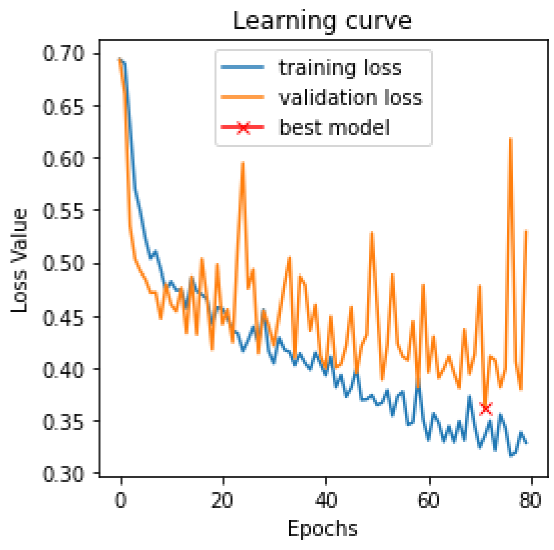
Data augmentation is a technique used in deep learning to impact positively on model training by creating variations of the training data through random transformations.
It increases the effective training data size by generating diverse data samples, improves model generalization by exposing it to a broader range of data patterns and variations. Acts as a form of regularization, reducing overfitting by encouraging the model to focus on essential features. Enhances the model's robustness to real-world variability, making it better prepared for unseen data. And also contributes to stable training by introducing variability.
To sum up, it is a crucial technique for training deep learning models, especially when the available training data is limited.

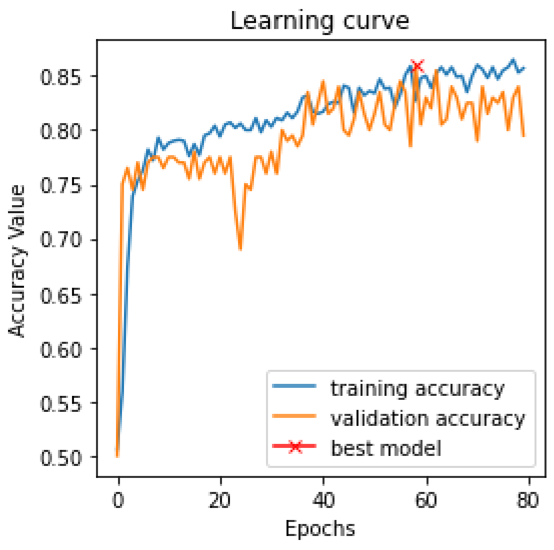
Task 9

Repeat task6 for VGG model for both skin and bone data set.

We use the VGG model with dropout layers and the same parameters as task 8.

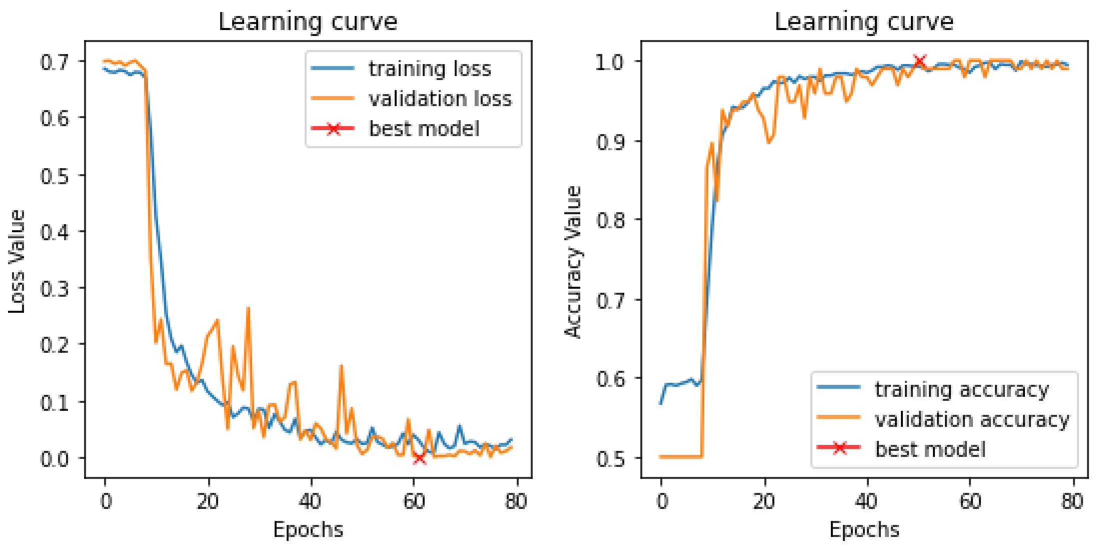
- Skin dataset:
    - 1000 training images
    - 200 validation images





For the skin dataset, using the VGG16 model we observe good accuracy and loss values without the presence of overfitting.

- Bone dataset:
    - 1112 training images
    - 96 validation images



We can observe that the best performance occurs with 60 epochs on the bone dataset with a loss value of 1.4526e-05 and an accuracy value of 1.