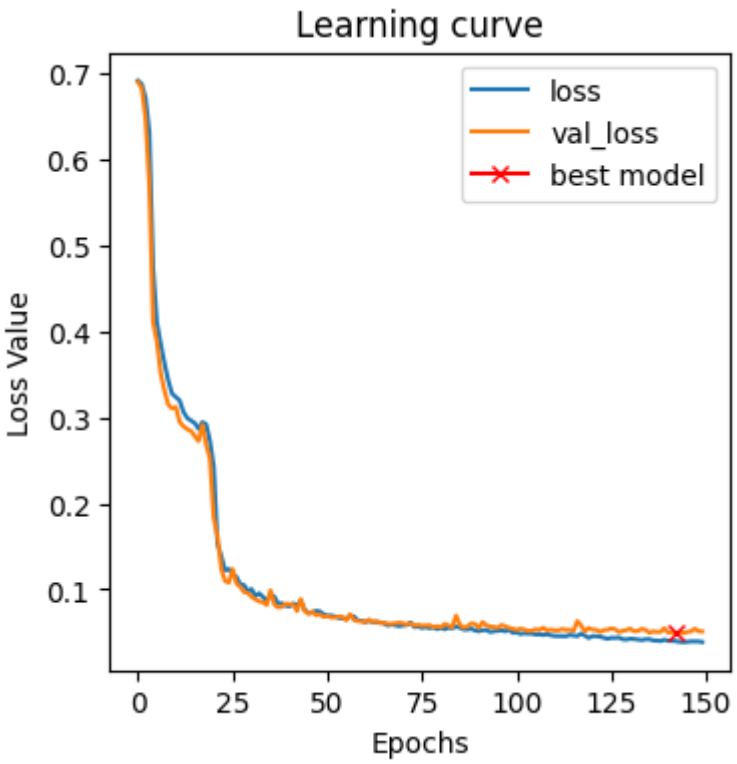
Lab 4

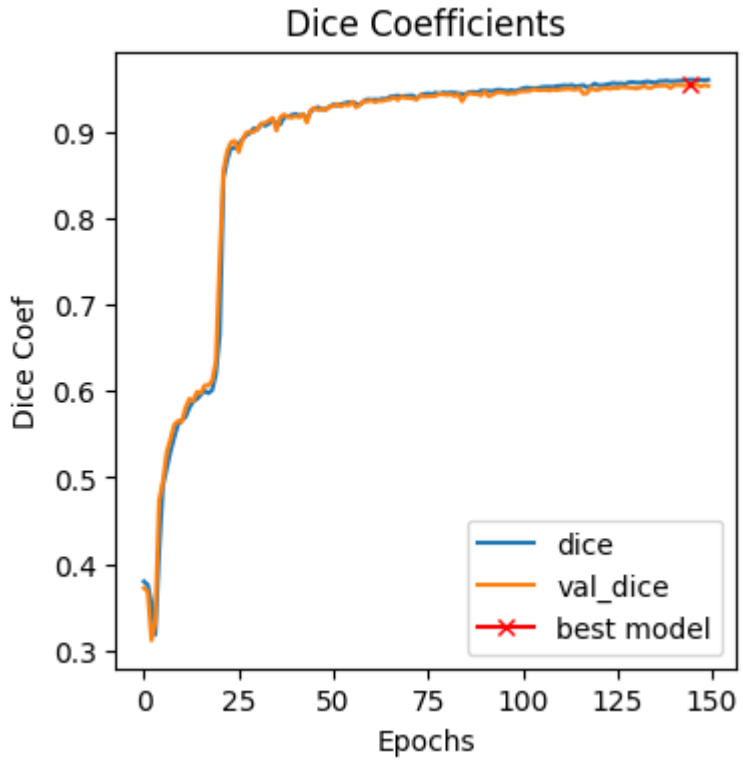
Task1a)

Instantiate your pipeline with the following parameters: # of filters at first layer (base)=8; image size=256; batch size=8; learning rate = 0.0001; dropout=Fase; batch normalization=false; Metric=Dice Coefficient; no data augmentation; and loss function=binary cross-entropy. Reading the 'X_ray' images (images and masks), randomly shuffle them and assign 80 percent of them for training and the rest of 20 percent for validation. Train the model for 150 epochs and report your final results. Please remember that you have to normalize the intensity range of the loaded images. Moreover, the segmentation masks are saved as 8-bit jpg images so that the background is labeled as 0, and the foreground is labeled as 255. Segmenting the lung regions within the X-ray images can be considered as a binary segmentation task, so that you need to normalize the loaded segmentation masks as well.

Lung segmentation in chest X-ray images:

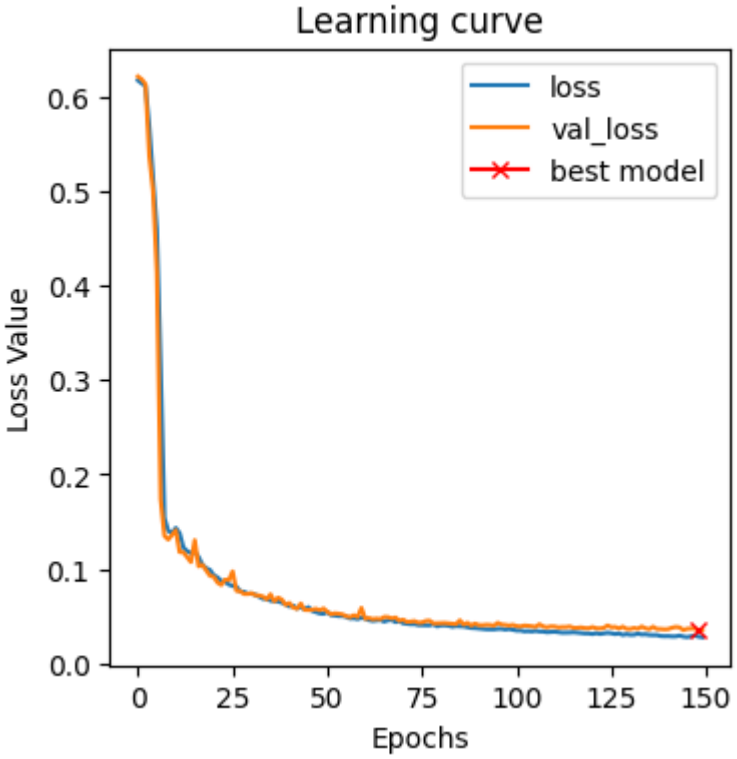
drop out = False , loss =BCE , n_base = 8, batch_norm = False

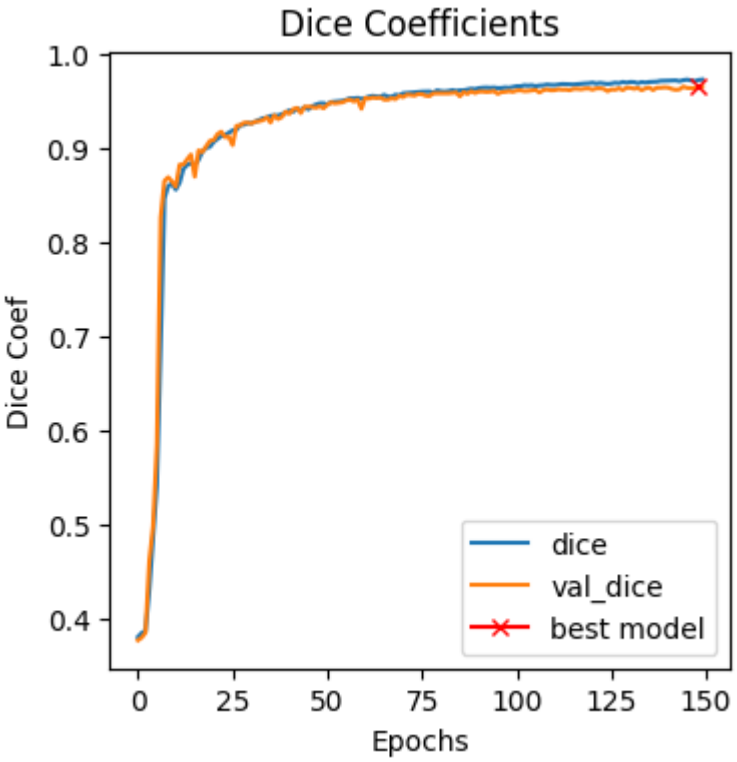




Task 1b) Is there any difference between model performance over validation set when you changed the loss functions? Do you expect to observe similar results when you deal with more challenging segmentation tasks? Dealing with cases in which the size of the target to be segmented would be much smaller than the image (imbalance between the background and foreground). Which loss function would you choose? Discuss it.
drop out = False , loss = dice_coef, n_base = 8, batch_norm = False

Answer:
From the results, performance of the model with dice loss shows smoother iteratial curves compared to the one with BCE.
When dealing with more challenging tasks, the performance might be different.
When the size of the target is much smaller, dice loss will be a better choice.
Since BCE calculates the difference of information between predicted images and true images. It treats positive and negative results equally, which means it works better with images of equal distribution. When there is big imbalance, the results from BCE will be less plausible.

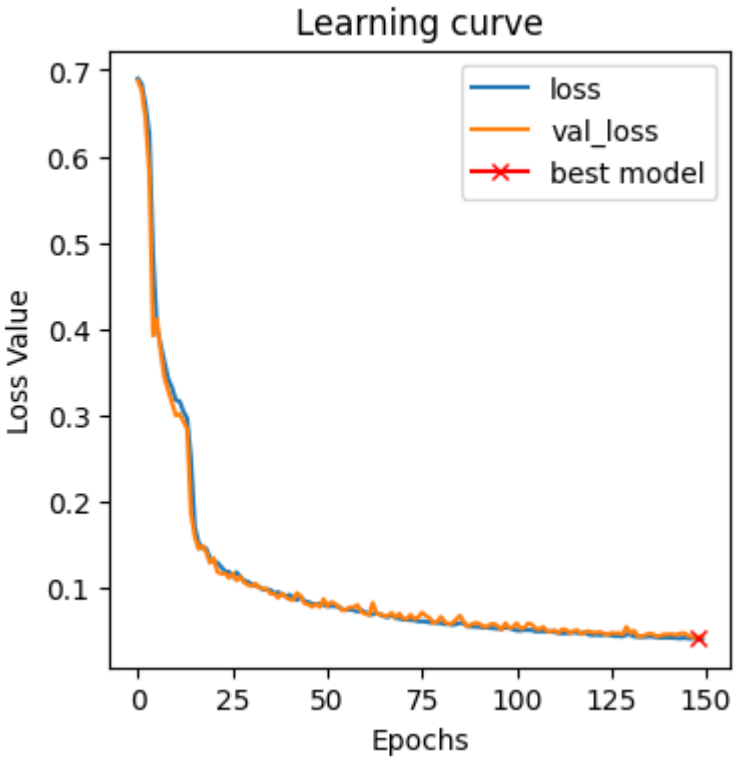


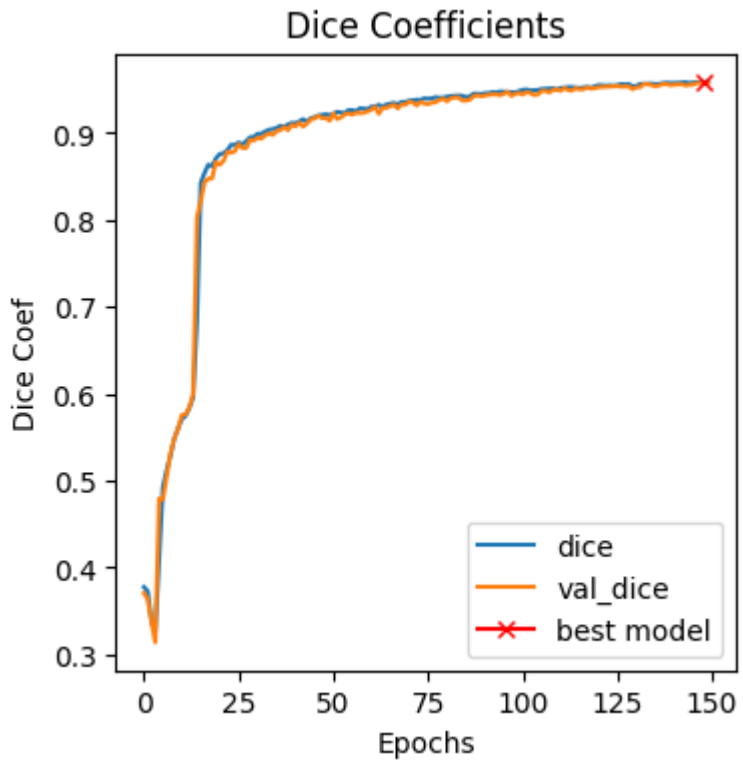


Task1c) Repeat the tasks 1a and 1b by including the dropout layer (rate=0.2). **Does it affect model performance? What about the learning curves?**

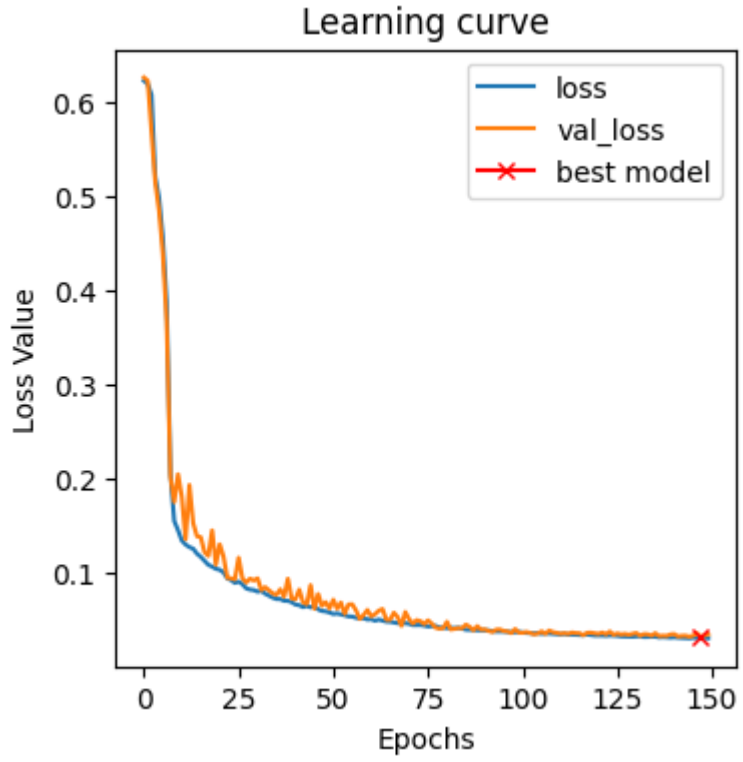
Adding dropout layers into the networks, the training curve and validation curve are closer for models with both BCE and Dice as loss function. We conclude that the models are forced to learn more from the features with the dropout layer. The curves from the model with BCE are smoother than before. Within 150 epochs, the performance of models is not stable yet. Only from the shapes and characters of curves, we conclude dropout layers improve performance in training and validation.

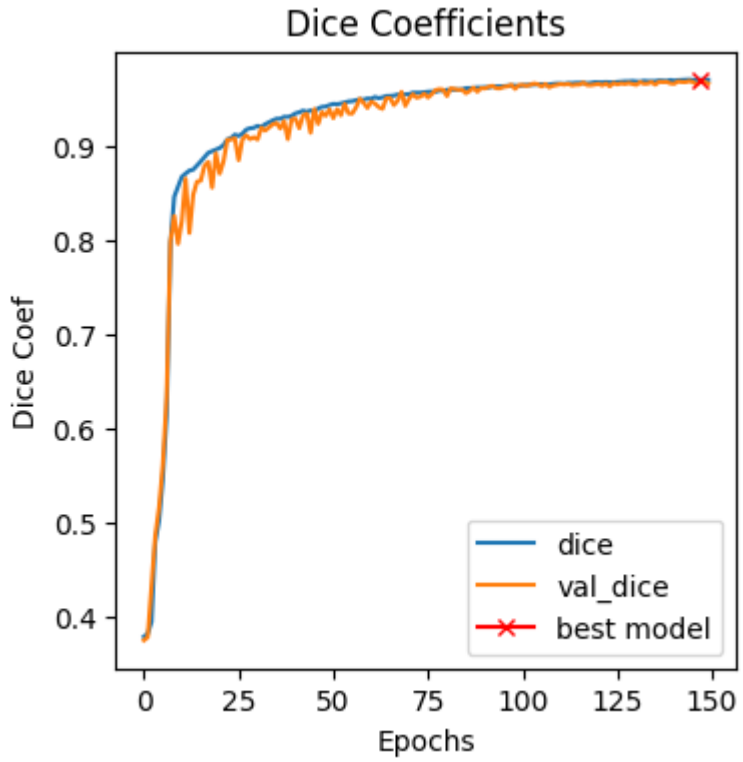
drop out = True , loss = BCE, n_base = 8, batch_norm = False





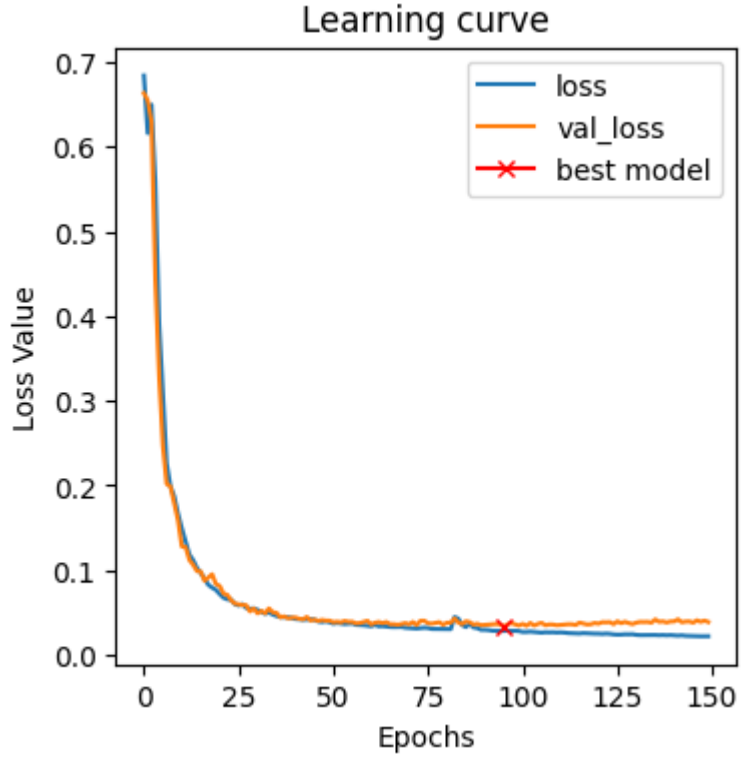
drop out = True , loss = dice_coef, n_base = 8, batch_norm = False

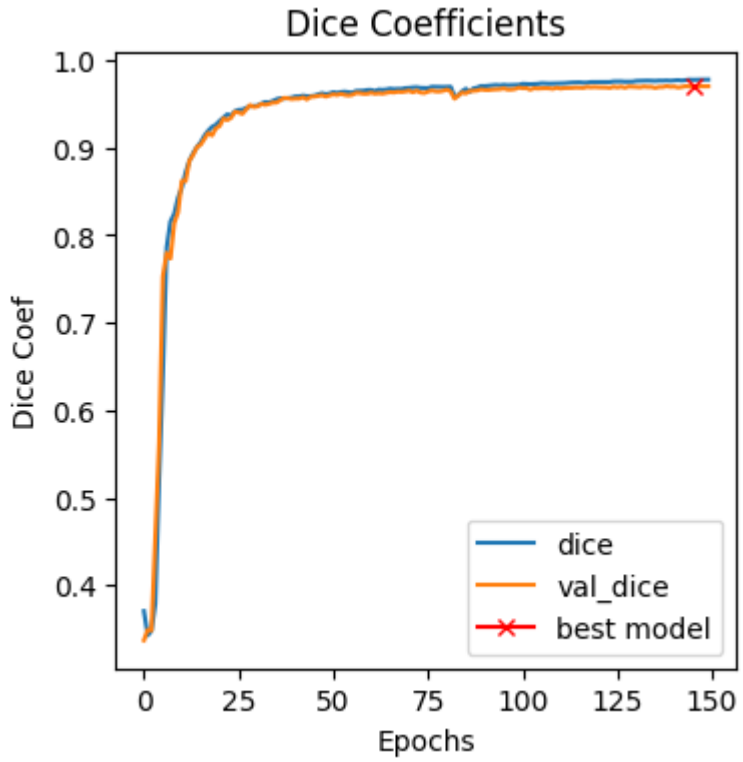




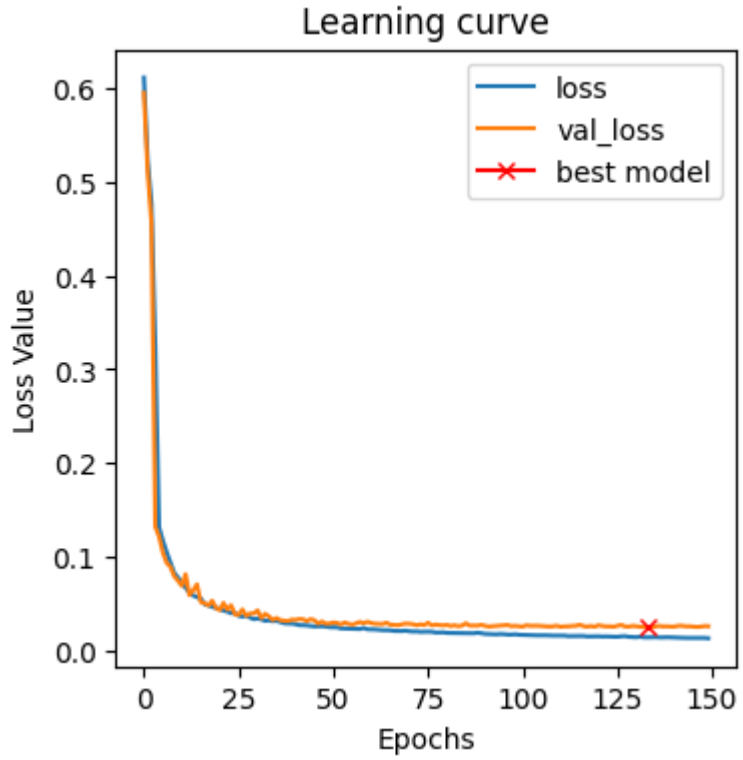
Task1d) Increase the model capacity by setting base=32. Repeat tasks 1c and evaluate the results.

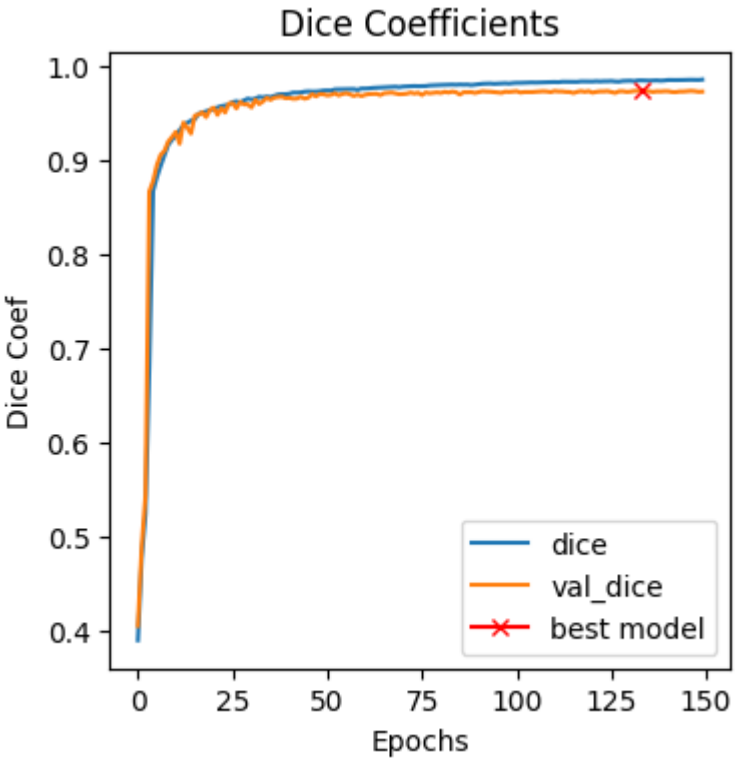
drop out = True , loss = BCE, n_base = 32, batch_norm = False





drop out = True , loss = dice coef , n_base = 32, batch_norm = False



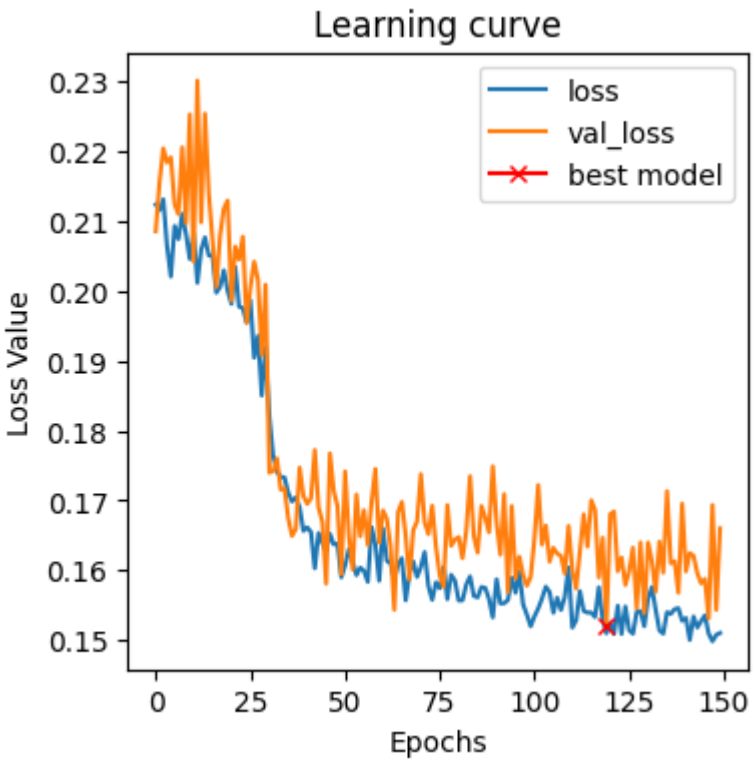

Dice Coefficients
Dice Coef
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
25
50
75
100
125
150
dice
val_dice
best model

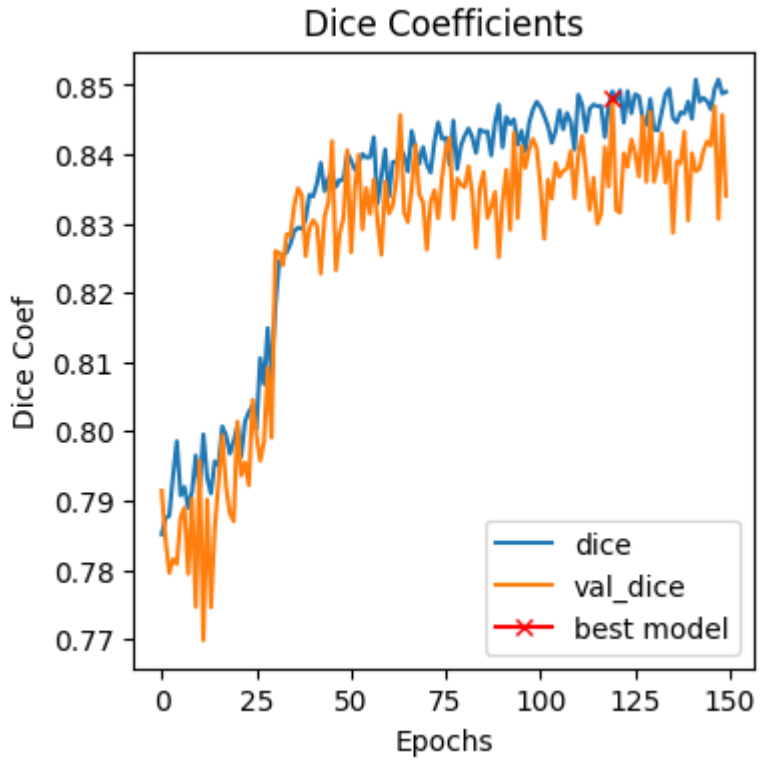
Task1e) Repeat the task 1c with setting the base=16, batch norm=True, and train the model by applying augmentation technique on both images and masks: rotation range=10; width and height shift ranges=0.1; zoom range = 0.2, and horizontal flip. Could image augmentation improve the generalization power of the model?

Answer:

With data augmentation, the model would learn from more images with various targets. From the figures, it shows that the performance of the model is more dynamic locally because the input data is of higher variation. Generally, the generalisation power of the model will improve with various data. From the figures, two learning curves are still not converging. So the final performance of models are unknown.

drop out = True , loss = dice coef , n_base = 16 , batch_norm = True





Task2a) Lung segmentation in CT images: Reuse the same pipeline of task1 for segmentation of lung regions within the “CT” dataset. Please note the left and right lungs are labeled with different values in the segmentation masks. So you need to binarize the loaded masks for a binary segmentation task. Use the following parameters: Base=8, batch norm=ture, image_size=256, train_percentage=0.8, LR=1e-4, loss=dice, dropout=True(0.2),batch size=8 and augmentation: false. Please note as the number of images in this data set is quite large (>8000 images), the computational time will be a bit longer so that you can train the model for only 50 epochs.

Image size 256*256

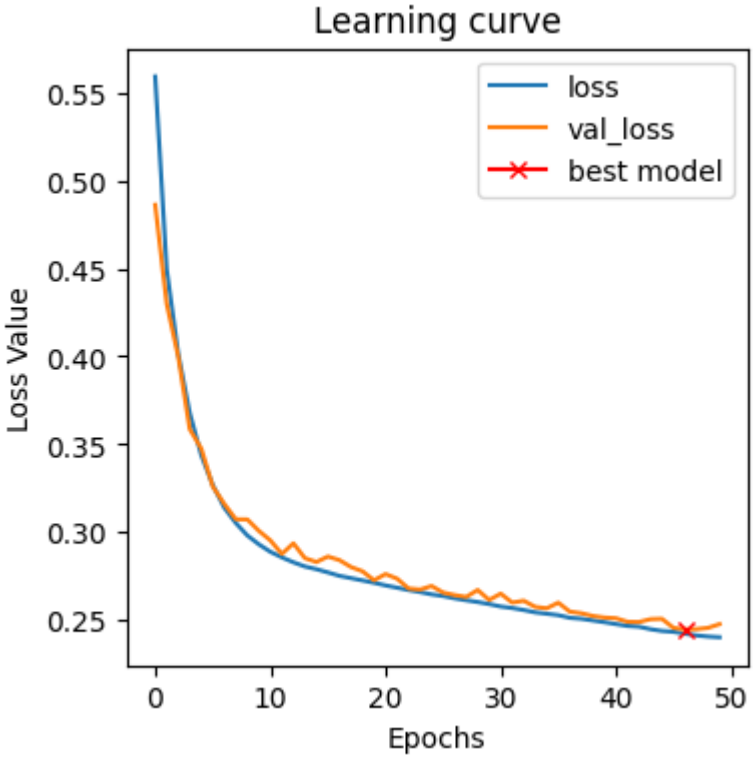
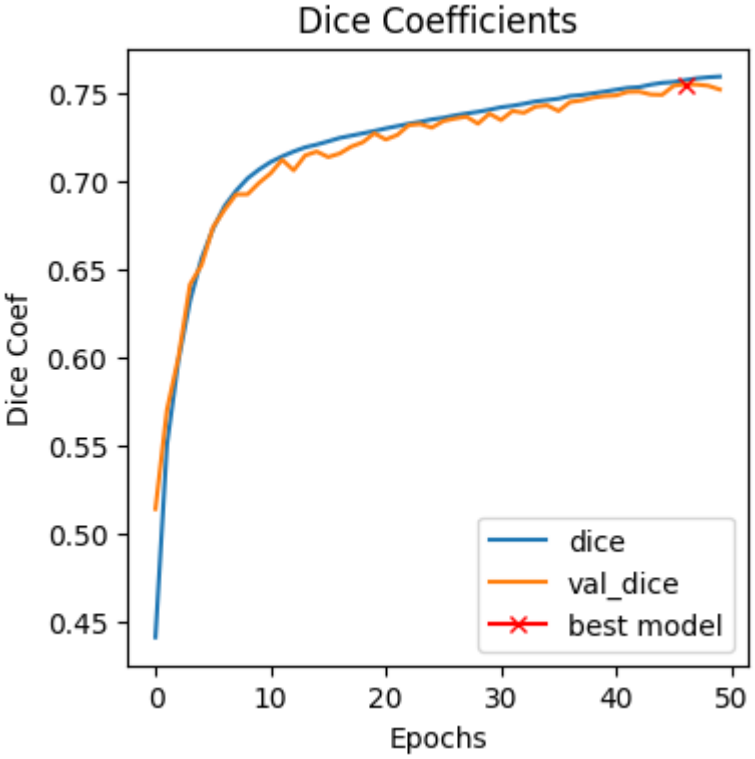


Image size 256*256



Task 2b

Evaluating the segmentation performance is often done by using Dice coefficient; however, it is quite essential to assess the rate of false positives and false negatives too. Implement “precision” and “recall” metrics as well, and use these three metrics in your model (Metric = [dice_coef, precision, recall]) to monitor the model performance. Repeat the task 2a by including data augmentation technique (similar to task1e) and train the model for 50 epochs and interpret the observed values of the three metrics over the validation set. **In general, what can be inferred from precision and recall metrics?**

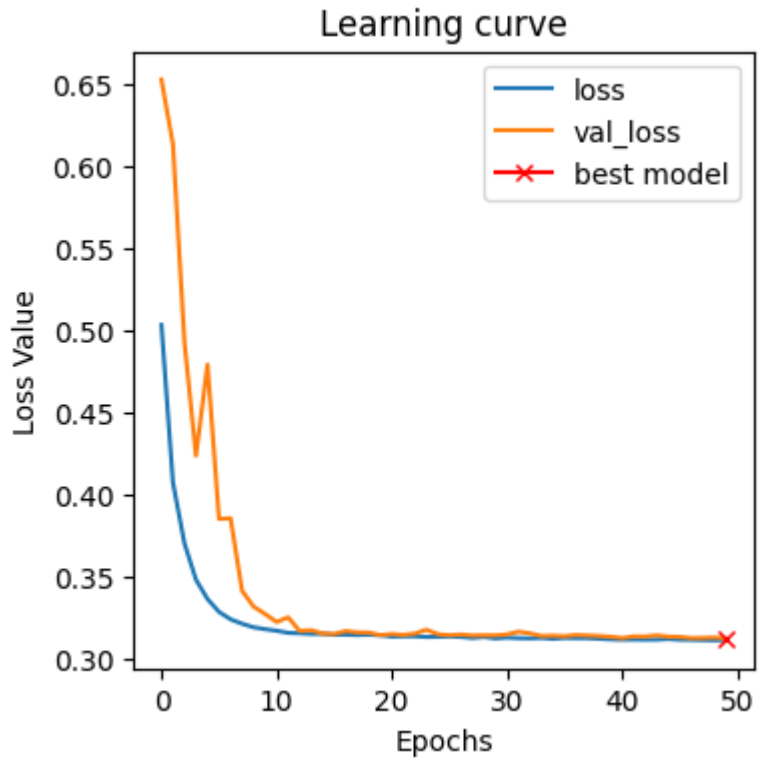
Answer:

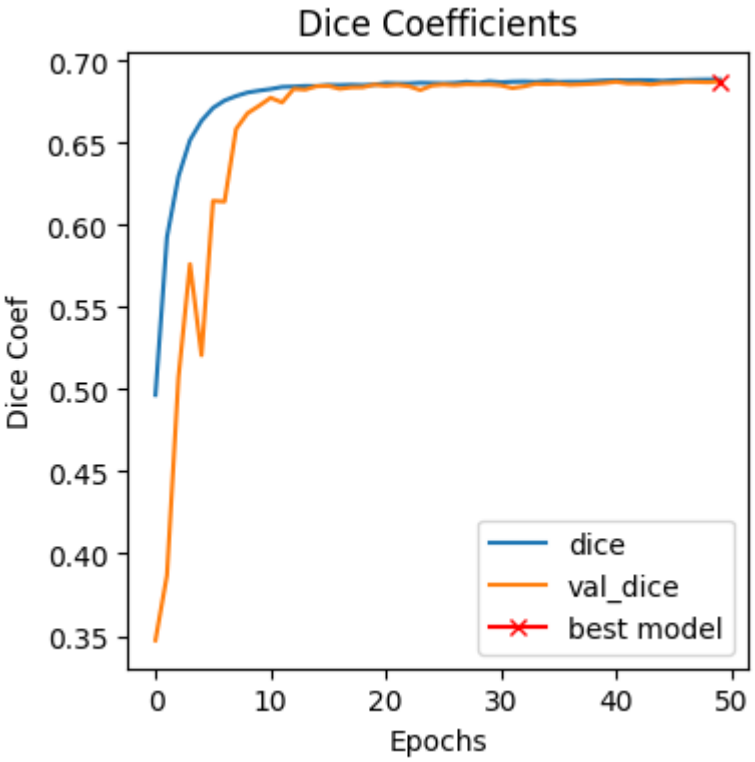
Precision = tp / (tp + fp)

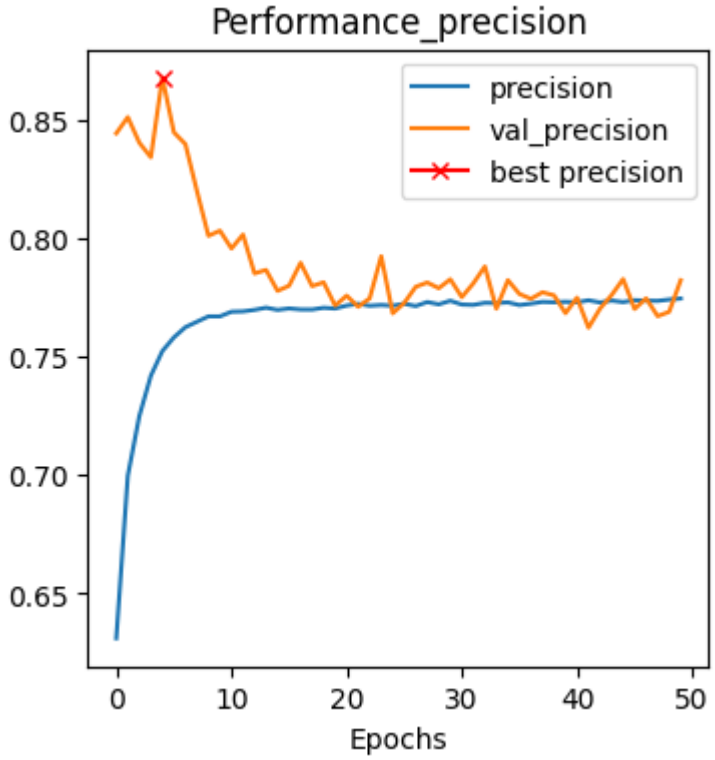
Recall = tp / (tp + fn)

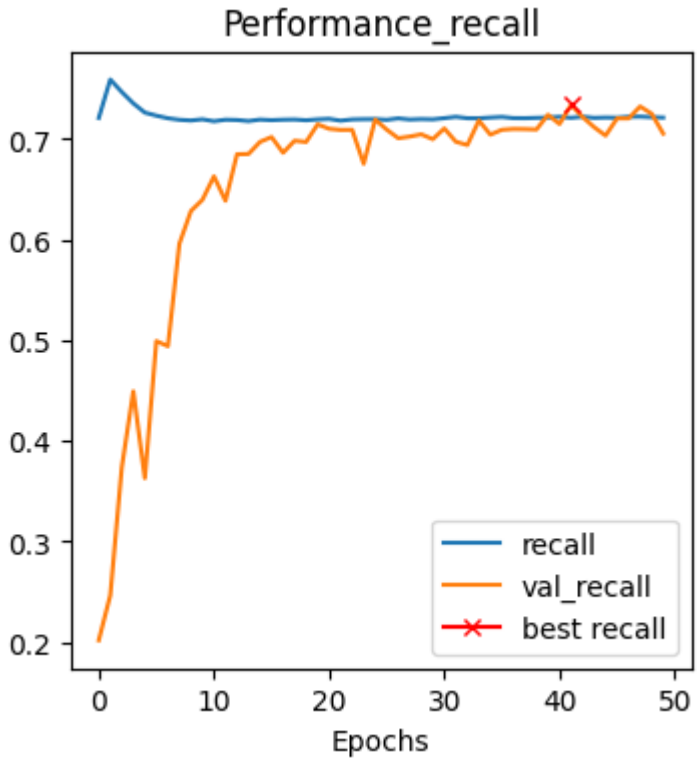
Precision is related to the positive predictions, and how many of these positive predictions are really positive. Having high precision means there are not many positive predictions that are wrongfully predicted, and we can trust that when the model has made a positive prediction that the model is indeed correct.

Recall is related to when the real label is positive. High recall means the model seldom predicts negative when it is supposed to be positive. For instance, if the patient has a tumour, we can trust that the model will pick up on this and give a positive output.

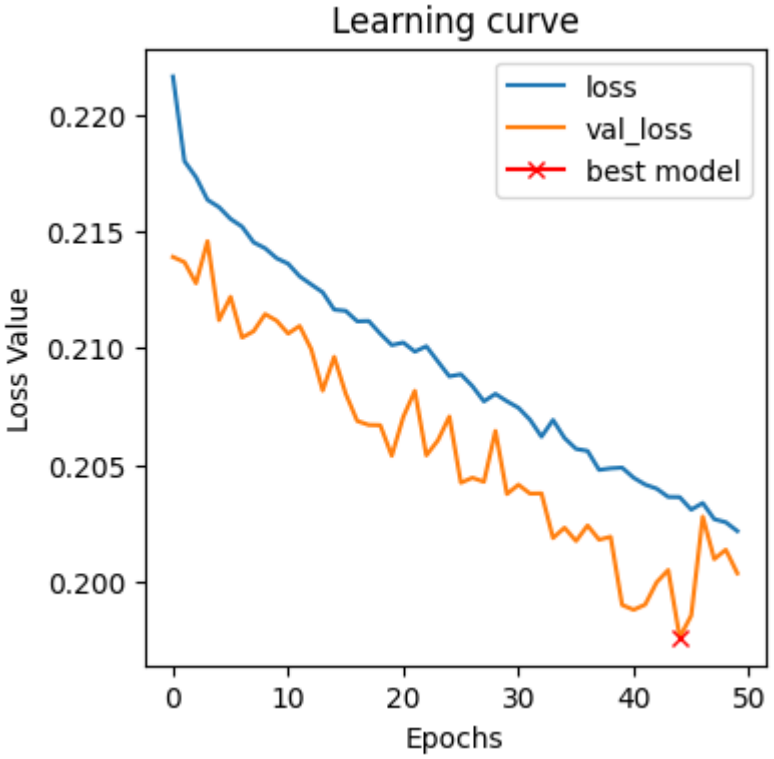


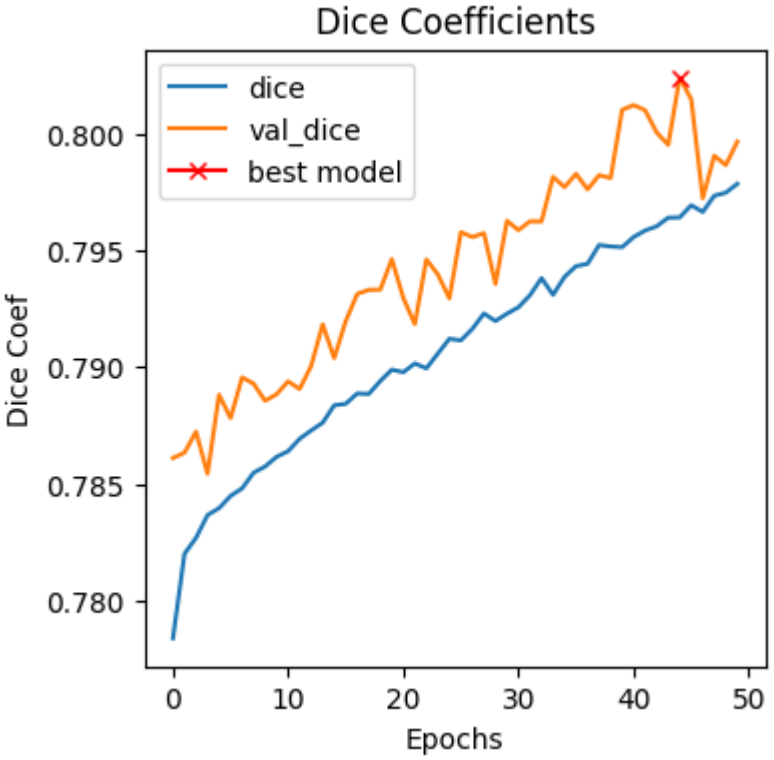

Dice Coefficients
Dice Coef
Epochs
dice
val_dice
best model
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
10
20
30
40
50

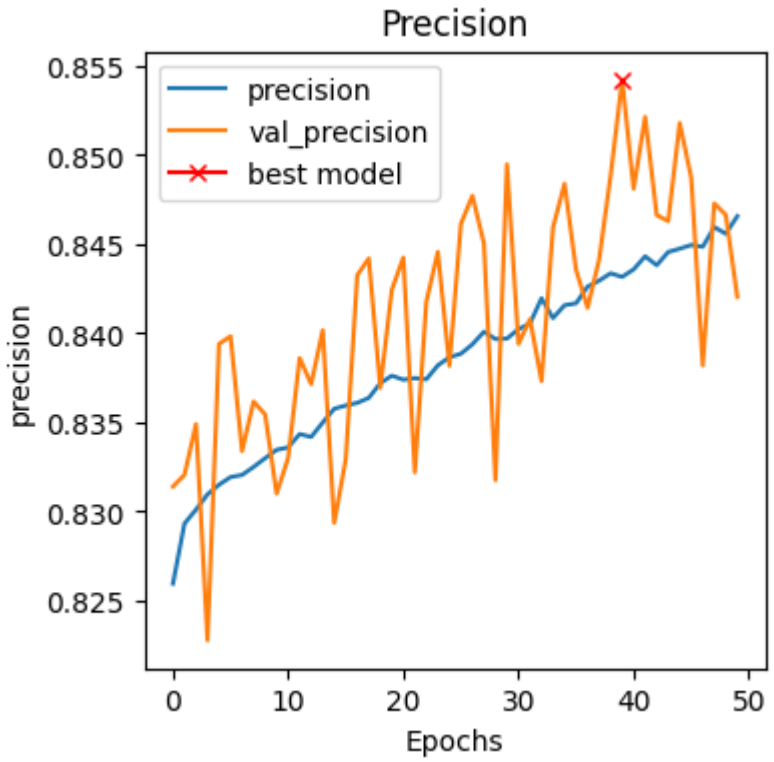

Performance_precision
precision
val_precision
best precision
Epochs
0.85
0.80
0.75
0.70
0.65
0
10
20
30
40
50

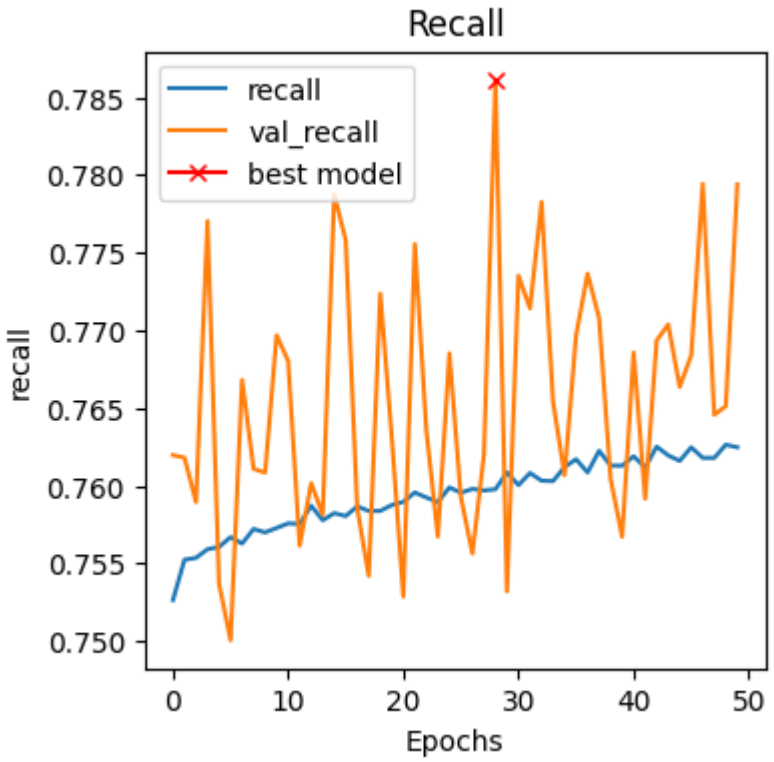


With augmentation


Learning curve
loss
val_loss
best model
Loss Value
0.220
0.215
0.210
0.205
0.200
0
10
20
30
40
50
Epochs


Dice Coefficients
dice
val_dice
best model
Dice Coef
0.800
0.795
0.790
0.785
0.780
0
10
20
30
40
50
Epochs

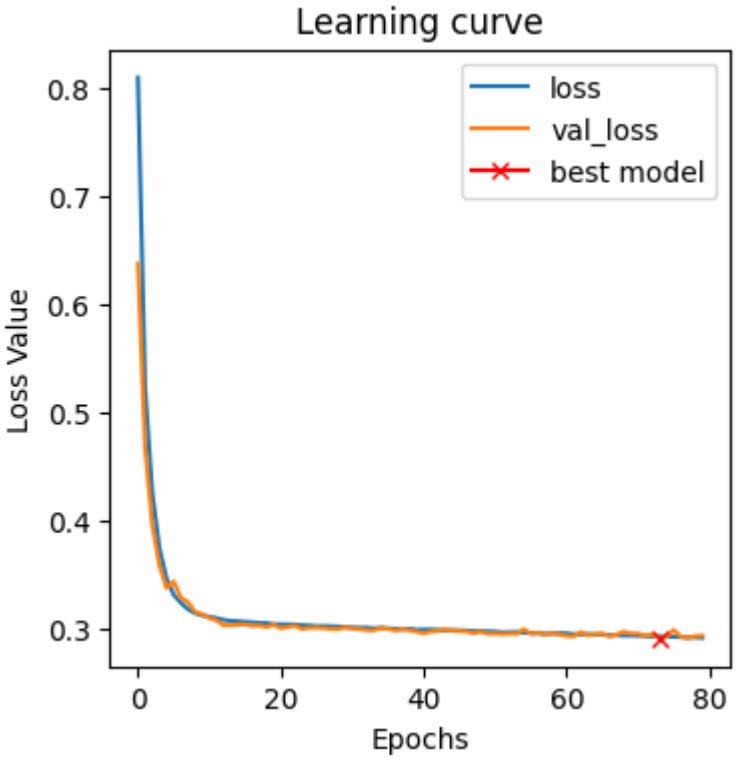


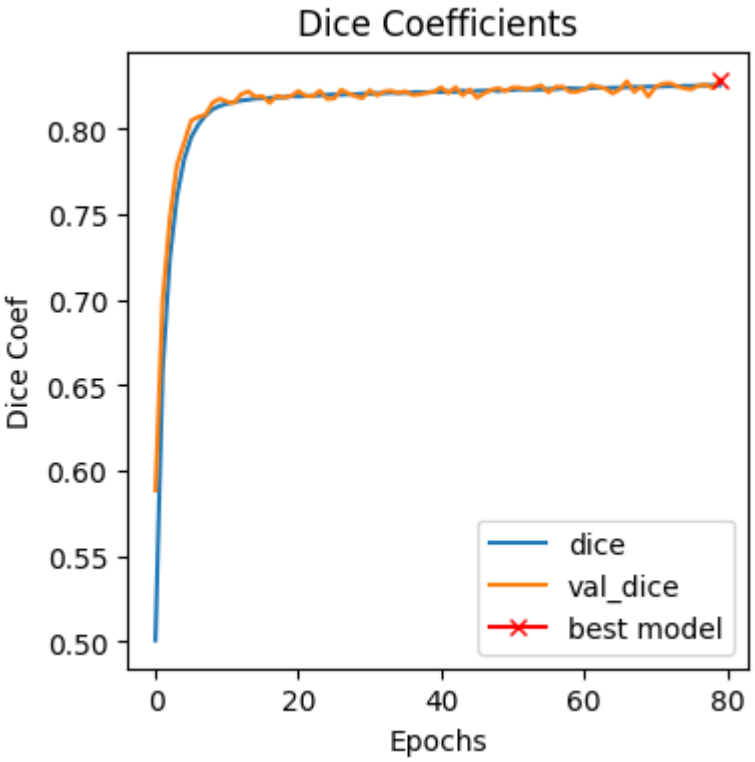


Task3) As it is already explained, the labels of the left and right lungs in the CT masks are not the same. We can consider the two lungs as two different organs and perform a multi-organ segmentation. Modify your model and adapt it for a multi-organ segmentation task and segment the left and right lungs separately in one framework. Set

the image_size:256, batch norm:true, LR:1e-4, dropout=True(0.2), augmentation:true and n_epoch=80. Choose a proper loss function for multi-organ segmentation and modify the segmentation mask labels to match the problem.

Model with data augmentation and multi-label masks.
Implementation please check in code files.




Dice Coefficients
Dice Coef
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
20
40
60
80
Epochs
dice
val_dice
best model