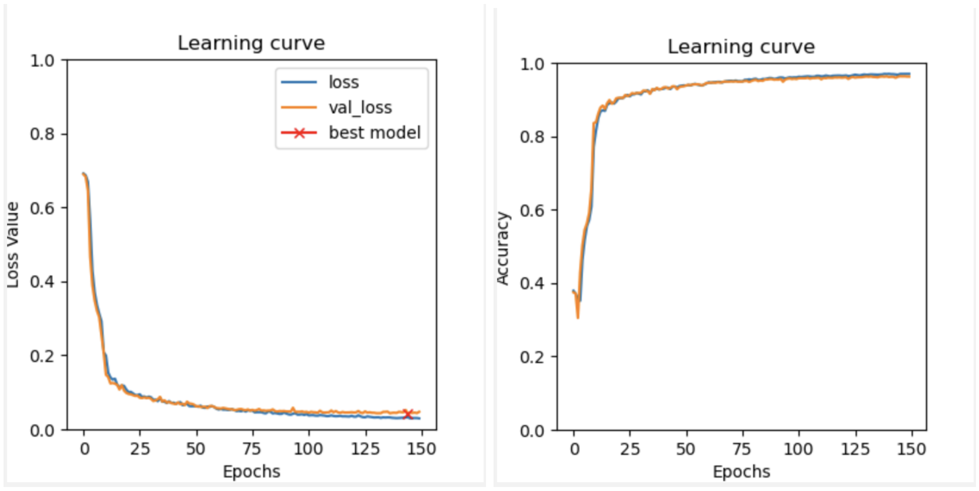
Task 1:

1a) Result of Unet on X-Ray images with cross-entropy-loss:

Training DSC: 0.97

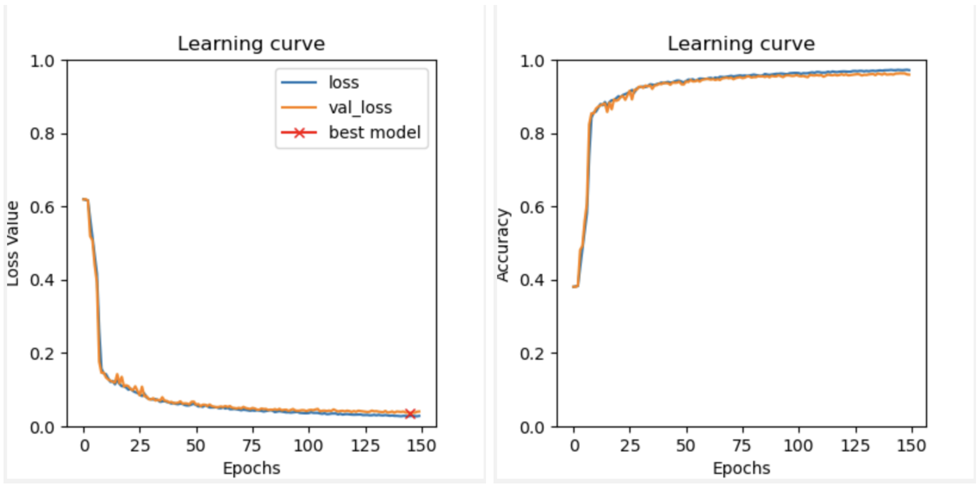
Validation DSC: 0.96



1b) Result of Unet on X-Ray Images with dice-loss:

Training DSC: 0.97

Validation DSC: 0.96



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?

The performance of the model using the dice-loss is the same as the model that incorporates the CE-Loss. The model converges in both cases and no overfitting can be seen in the learning curves.

For more challenging segmentation tasks, a difference is expected. Let's say we have large volumes with very detailed structures on the edges, the dice score tends to be very good, even if the edges are not segmented very finely, as the majority of pixels lie within that volume. Thus, the dice loss would be unable to account for that precision at the boundaries of two surfaces. As the cross-entropy loss is calculated per pixel, It would punish these imperfectly segmented edges more heavily. Thus, the CE loss (or a combination of both) would be more useful.

If the labels in the challenging segmentation task are unbalanced, the dice score might be more useful. (see next question)

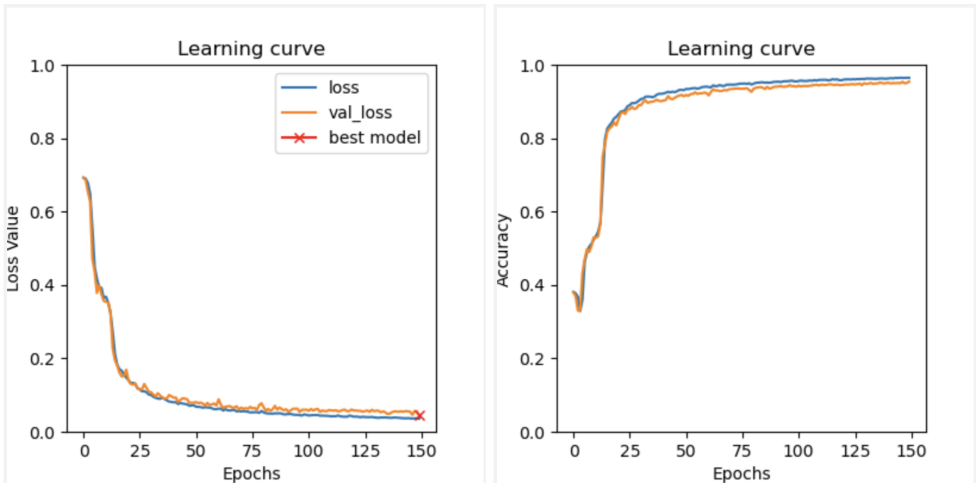
Dealing with cases in which the size of the target to be segmented would be much smaller than the image (imbalance between the background and foreground). Which loss function would you choose? Discuss it.
If there are substantially more background pixels than foreground pixels, the dice-loss would be the preferred option. Since the CE-loss is calculated in a pixel-wise manner, for sparse labels the wrongly segmented pixels would be punished equally, regardless of their distance to the target region. Dice loss on the other hand only focuses on the intersection of regions in the segmentation maps.

e)

Does it affect model performance? What about the learning curves?

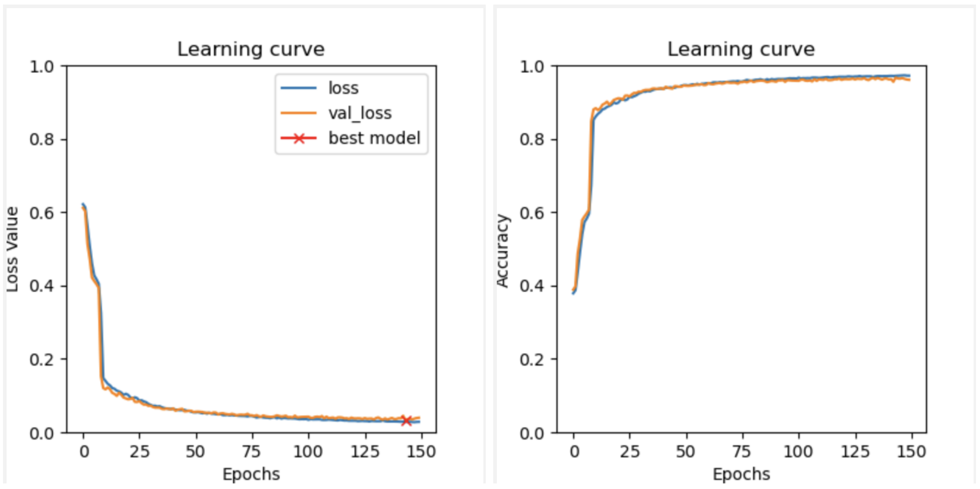
Unet with dropout 0.2 and CE-loss:

Training DSC: 0.96
Validation DSC: 0.95



Unet with dropout 0.2 and dice-loss:

Training DSC: 0.97
Validation DSC: 0.96



Does it affect model performance? What about the learning curves?

Adding the spatial dropout doesn't substantially change the models' performance in either case. Usually, dropout can help with overfitting. Since we already have a high validation dice score and no overfitting problem, it was expected, that it won't affect the result.

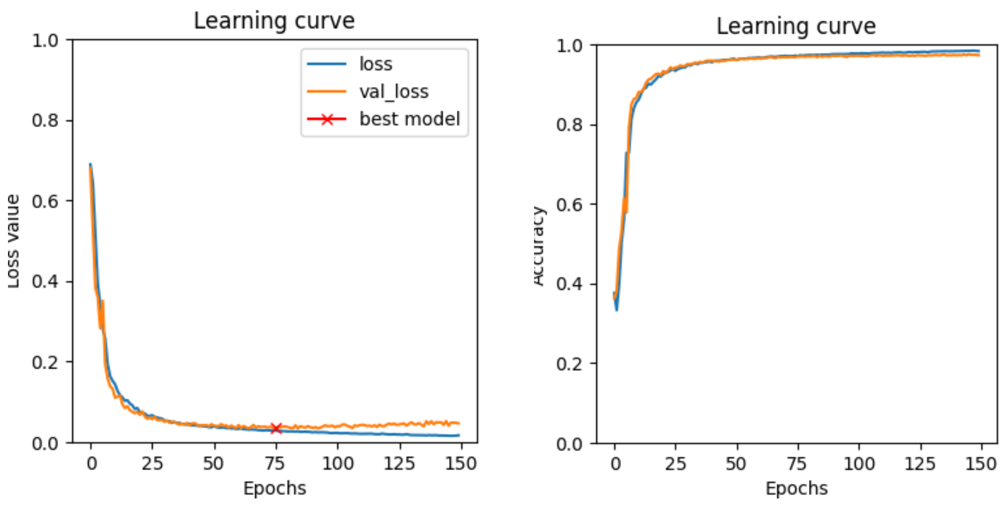
The learning curves are very steep in the first epochs and there is a step in the learning curves.

1d)

Unet with dropout 0.2, base=32 Binary Cross-Entropy loss:

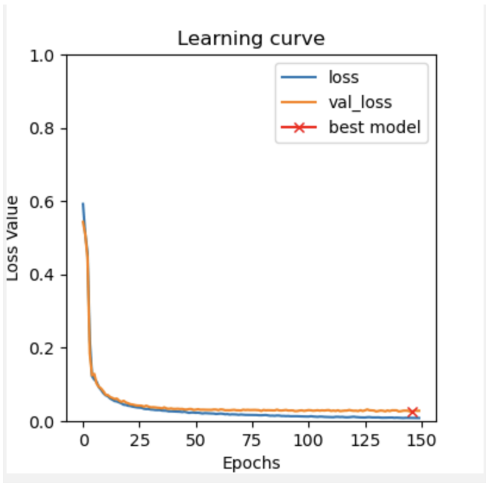
Training DSC: 0.98

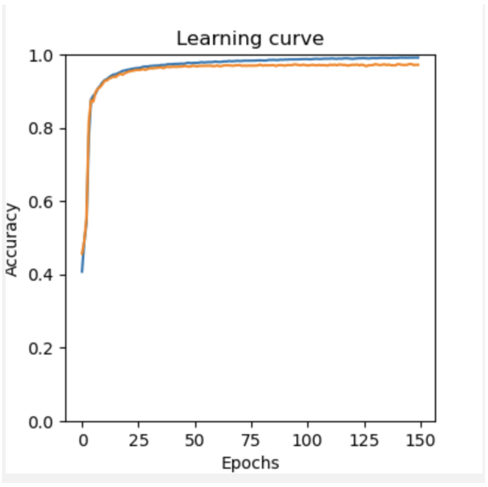
Validation DSC: 0.97



Unet with dropout 0.2, base=32 dice loss:

Training DSC: 0.99
Validation DSC: 0.97





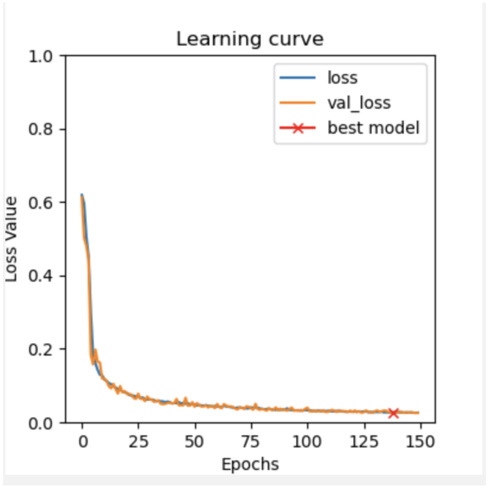
Repeat tasks 1c and evaluate the results.

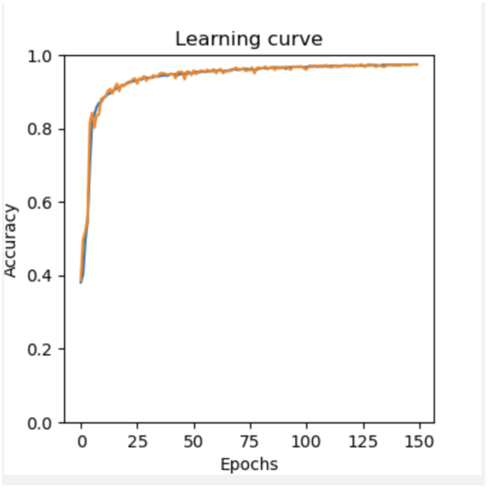
The training dice score is slightly higher when using a higher number of feature maps. The validation dice also increased slightly. The time for convergence seems to be similar and in both cases, the learning curves look smooth.

1e) **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?**

Model using Dice loss:

Training DSC: 0.97
Validation DSC: 0.97

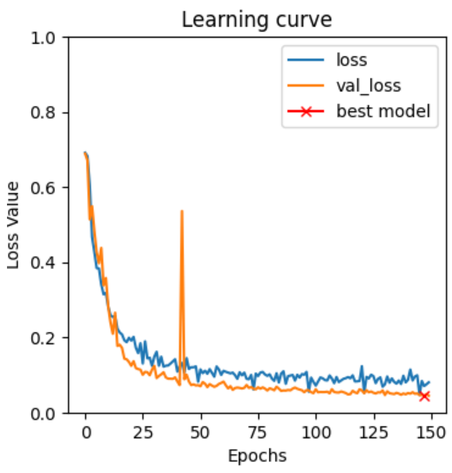


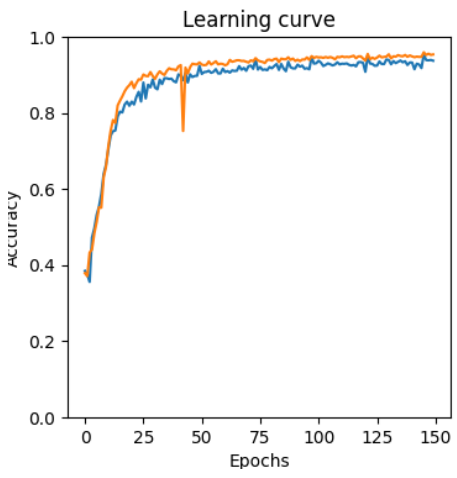


Model using Binary Cross-Entropy:

Training DSC: 0.94

Validation DSC: 0.95



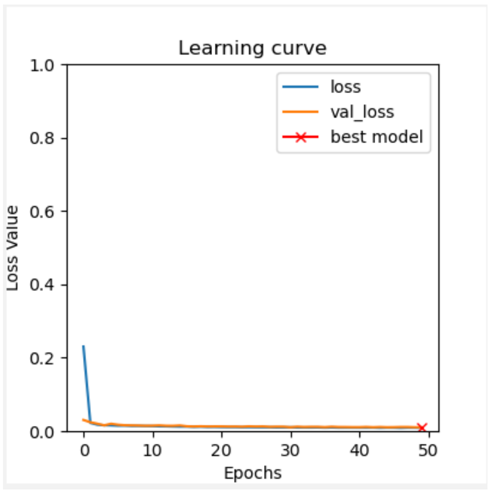


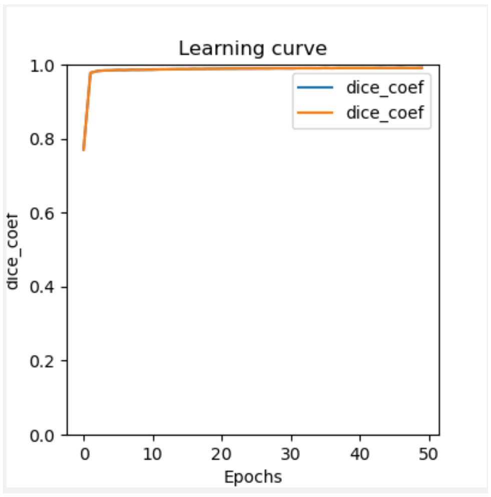
Using data augmentation resulted in a validation DSC that is higher or equal to the training DSC, thus, the generalization power of the model seems to have increased. However, the absolute DSC is slightly lower, suggesting a decrease in overall model performance.

2a)

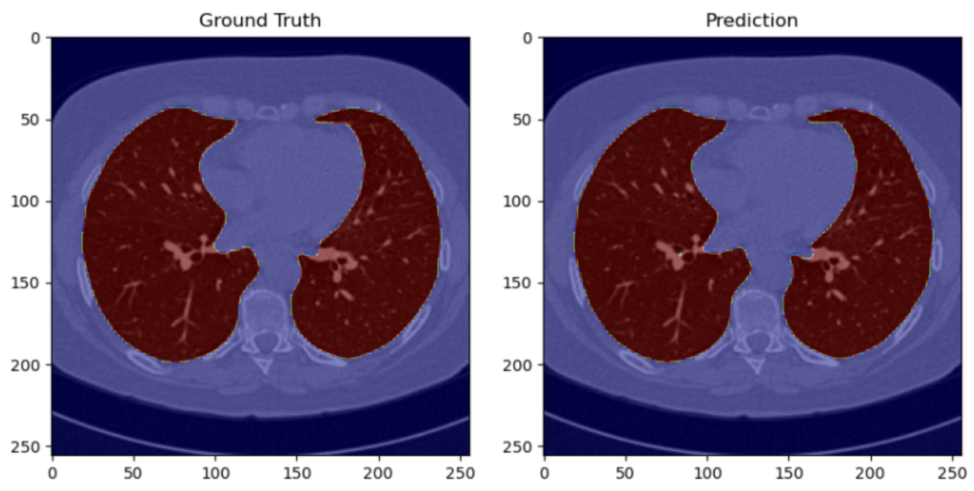
Training DSC: 0.991

Validation DSC: 0.99





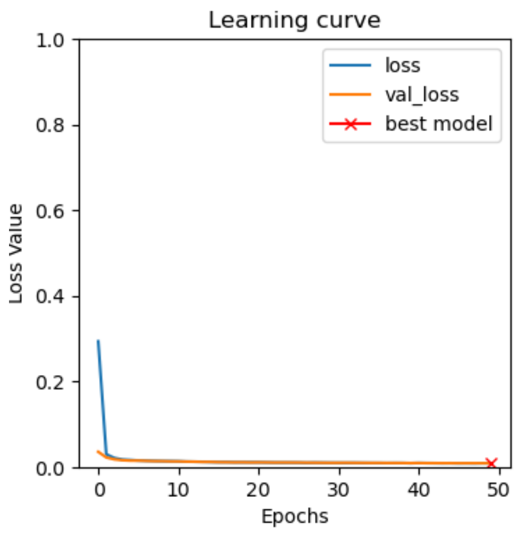
Sample prediction for a random image from the validation set:

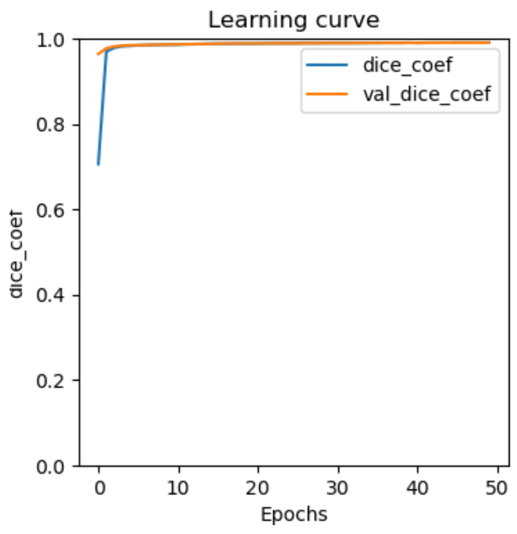


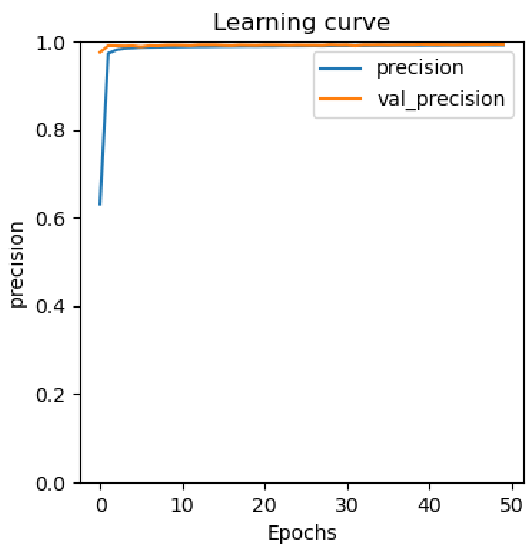
The segmentation is almost perfect. Note, that the data split was not performed patient-wise, thus slices from the same patient could be in the training set. Therefore, the validation performance will be lower on completely unseen patients.

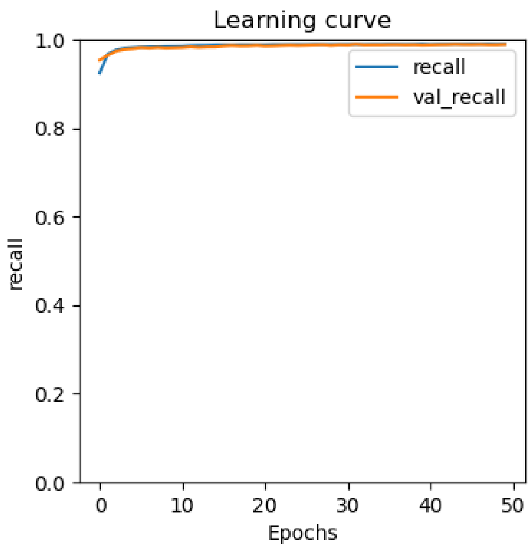
2a) corrected with dropout - 50 epochs

Training DSC: 0.992
Validation DSC: 0.992
precision
precision 0.9919797778129578
val_precision 0.993267297744751
recall
recall 0.9895551204681396
val_recall 0.9883333444595337



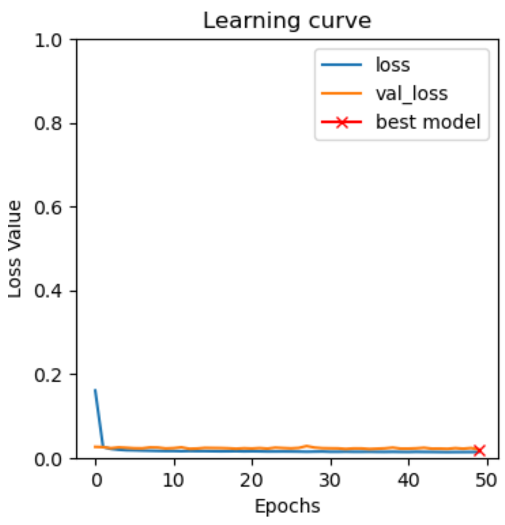


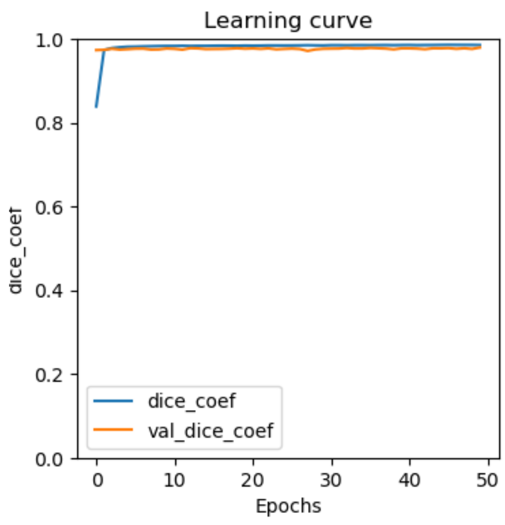


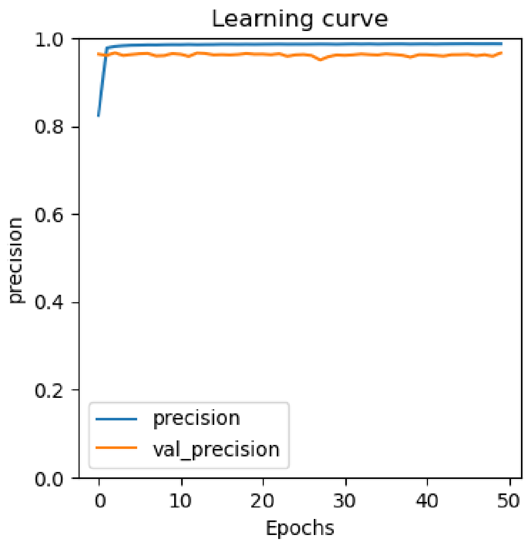


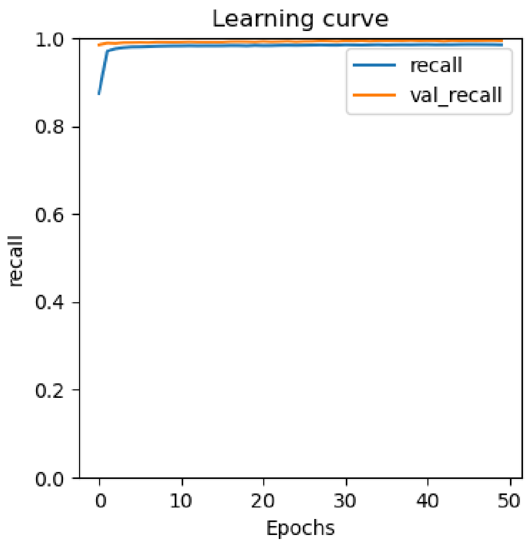
2b)

	DSC	precision	recall
Training	0.985	0.986	0.984
Validation	0.979	0.966	0.994

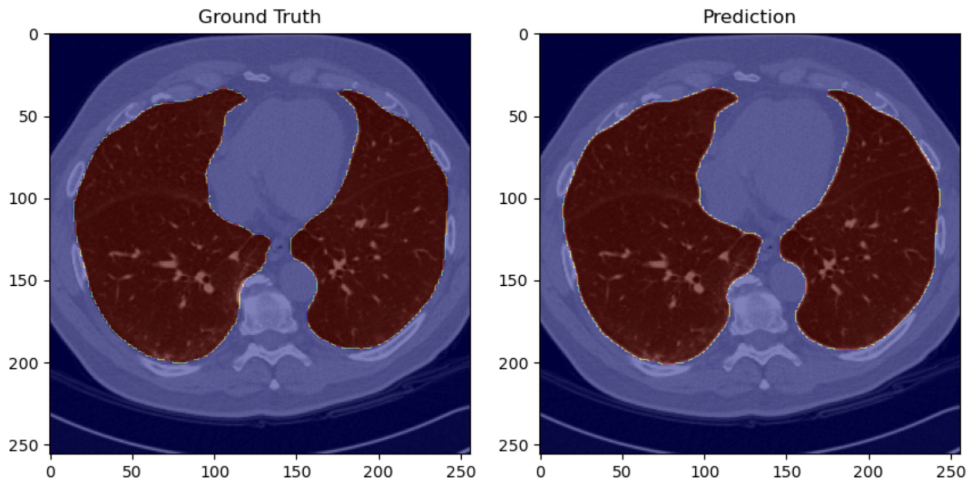








Sample segmentation from validation dataset:



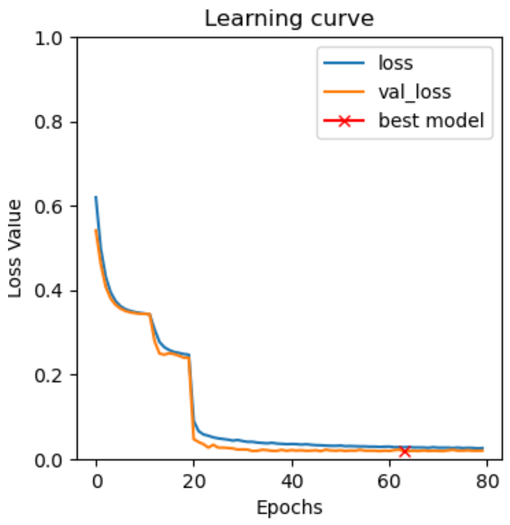
In general, what can be inferred from precision and recall metrics?

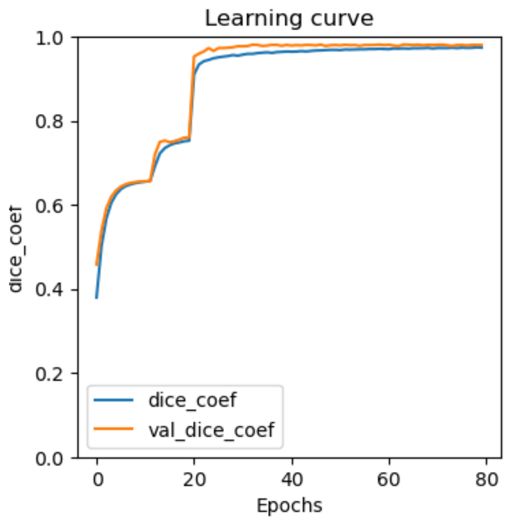
Recall tells us how many pixels of the lung have been correctly segmented relative to all true lung pixels.

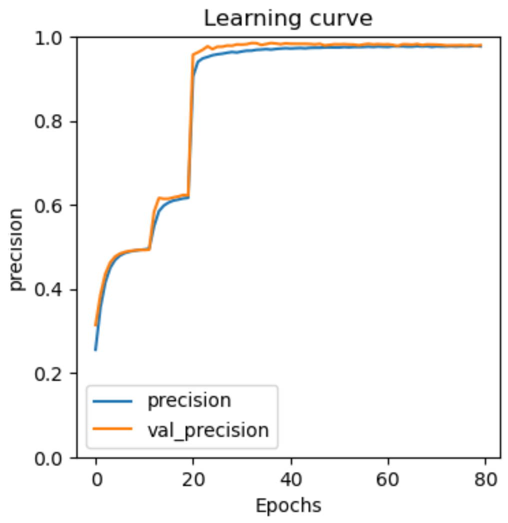
Precision tells us the number of pixels that have been classified correctly as lung relative to the number of all pixels that have been classified as lung.

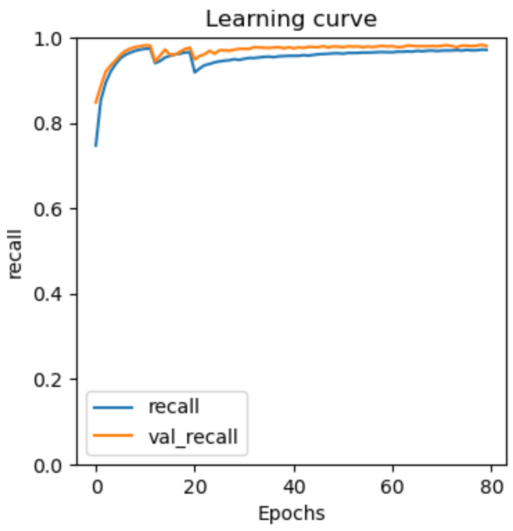
Task 3)

	DSC	precision	recall
Training	0.974	0.977	0.971
Validation	0.98	0.98	0.98


Learning curve
loss
val_loss
best model
Loss Value
Epochs


Learning curve
dice_coef
val_dice_coef
dice_coef
Epochs


Learning curve
precision
val_precision
precision
Epochs


Learning curve
recall
val_recall
recall
Epochs