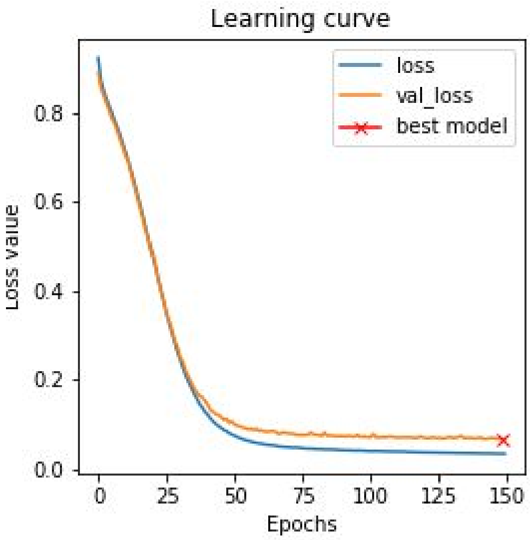
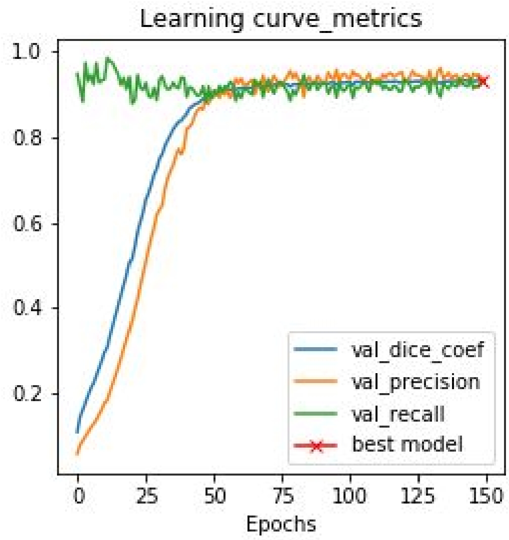
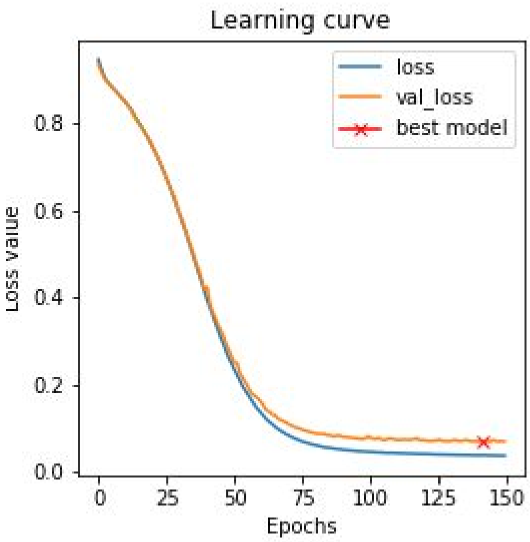
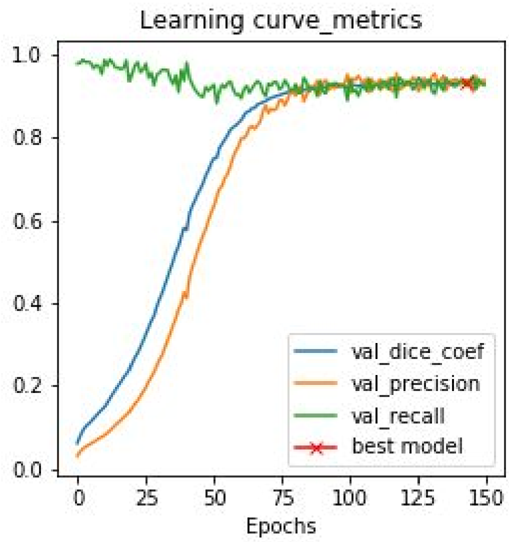
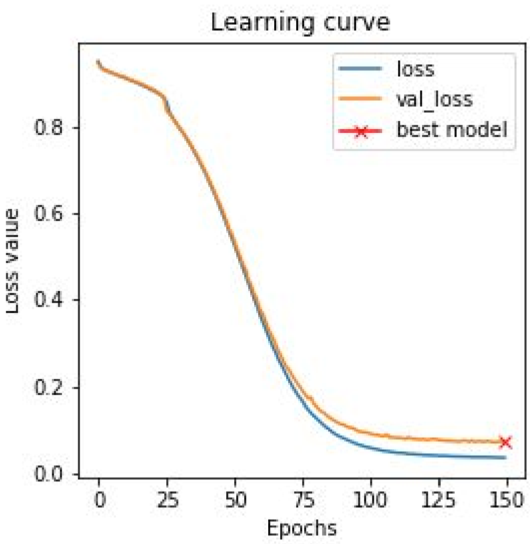
Task1
k=3:

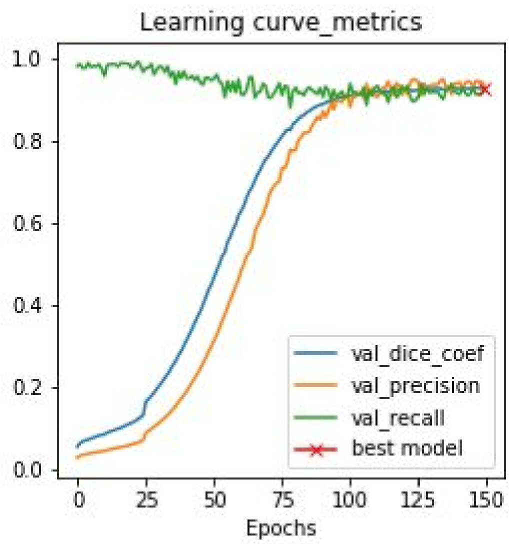

Learning curve
loss
val_loss
best model
Loss value
0.0
0.2
0.4
0.6
0.8
0
25
50
75
100
125
150
Epochs


Learning curve_metrics
val_dice_coef
val_precision
val_recall
best model
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
Epochs


Learning curve
loss
val_loss
best model
Loss value
0.0
0.2
0.4
0.6
0.8
0
25
50
75
100
125
150
Epochs


Learning curve_metrics
val_dice_coef
val_precision
val_recall
best model
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
Epochs

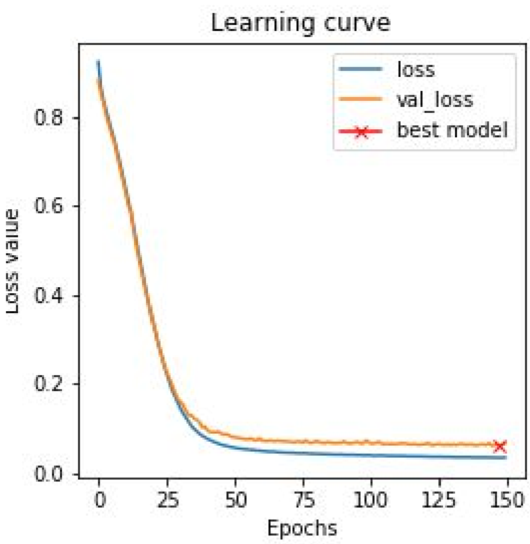

Learning curve
loss
val_loss
best model
Loss value
0.0
0.2
0.4
0.6
0.8
0
25
50
75
100
125
150
Epochs

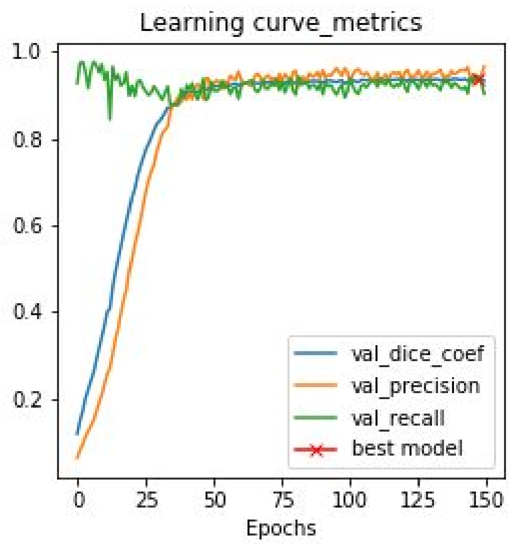

Learning curve_metrics
val_dice_coef
val_precision
val_recall
best model
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
Epochs

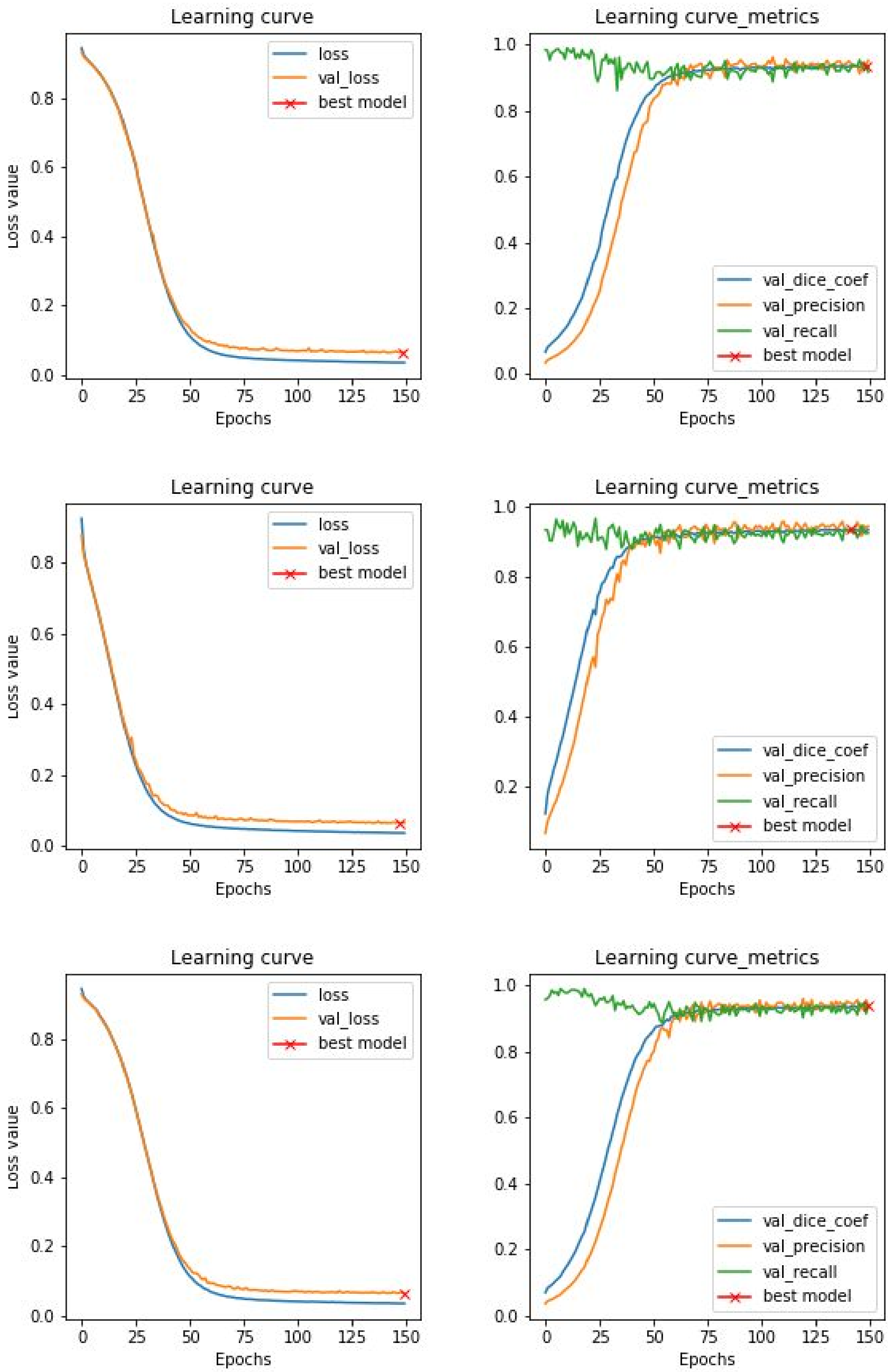
Fold	dice_coef	precision	recall
1	0.9320	0.9375	0.9272
2	0.9317	0.9296	0.9345
3	0.9285	0.9348	0.9230
average	0.9307	0.9340	0.9282

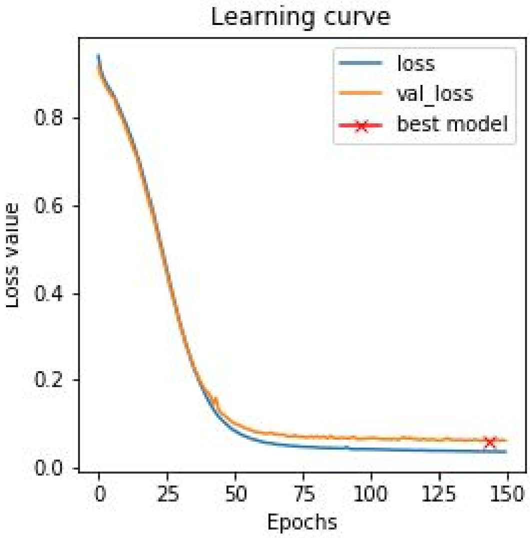
From the results we can see that the performance is not exactly the same across all folds. When we take the first fold as validation set, the model achieves highest accuracy. Actually for this dataset, the performance of each fold is not very different from each other. So if we have a dataset for which some folds have a different performance than others, I will shuffle the dataset until the performance doesn't vary too much across all folds.

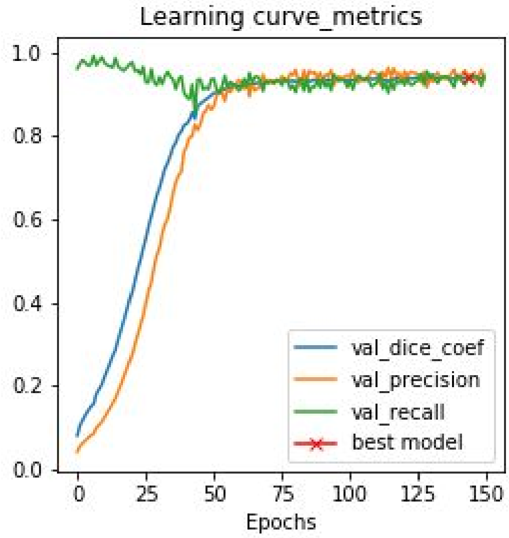
Bonus task
k=5:






Learning curve
loss
val_loss
best model
Loss value
Epochs
Learning curve_metrics
val_dice_coef
val_precision
val_recall
best model
Epochs
Learning curve
loss
val_loss
best model
Loss value
Epochs
Learning curve_metrics
val_dice_coef
val_precision
val_recall
best model
Epochs
Learning curve
loss
val_loss
best model
Loss value
Epochs
Learning curve_metrics
val_dice_coef
val_precision
val_recall
best model
Epochs

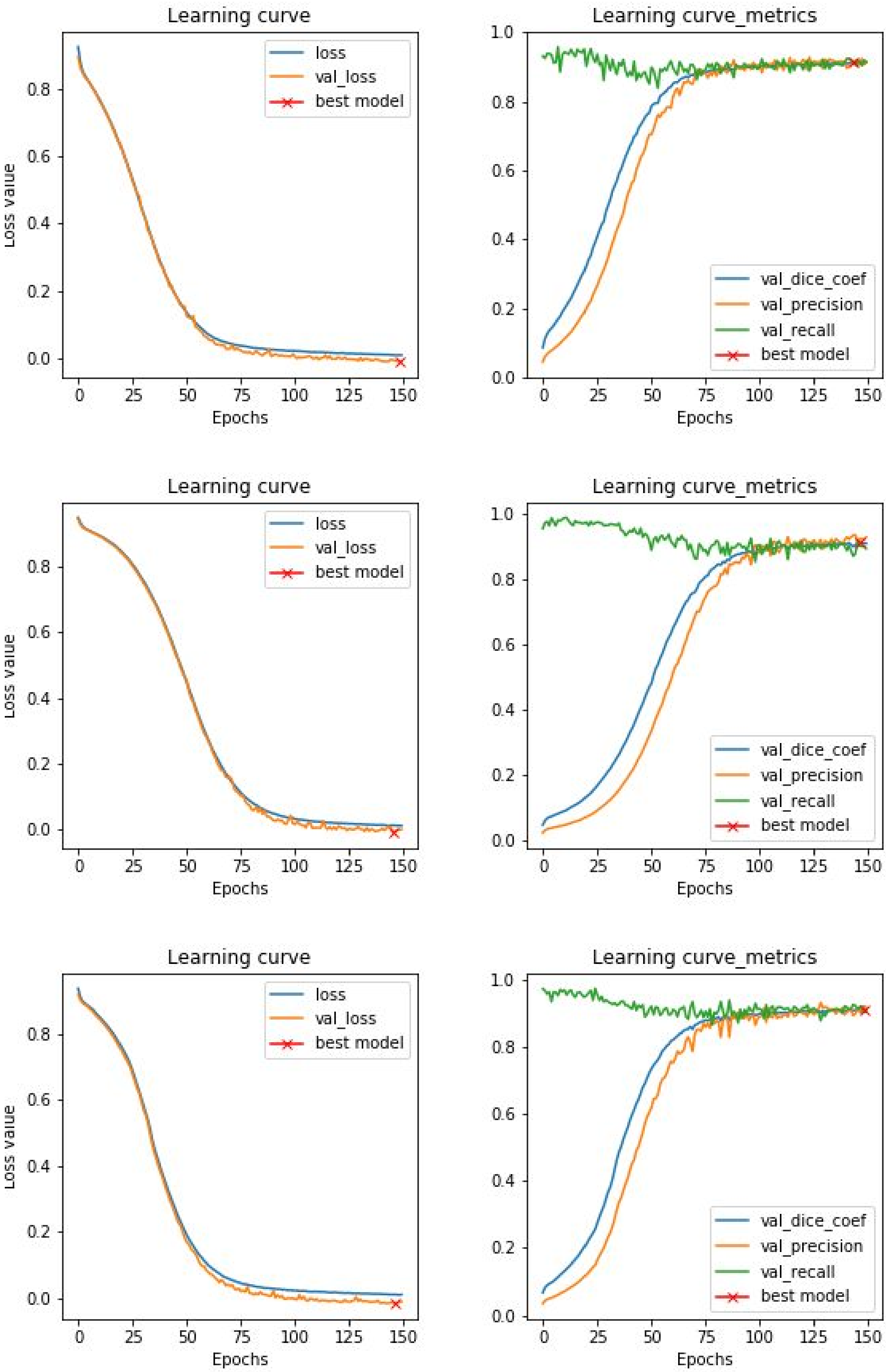




Fold	dice_coef	precision	recall
1	0.9371	0.9472	0.9277
2	0.9362	0.9391	0.9339
3	0.9364	0.9470	0.9266
4	0.9363	0.9389	0.9344
5	0.9403	0.9433	0.9377
average	0.9373	0.9431	0.9321

The average of accuracy, precision and recall are all higher when we increase the number of folds to 5, which means the model performs better.

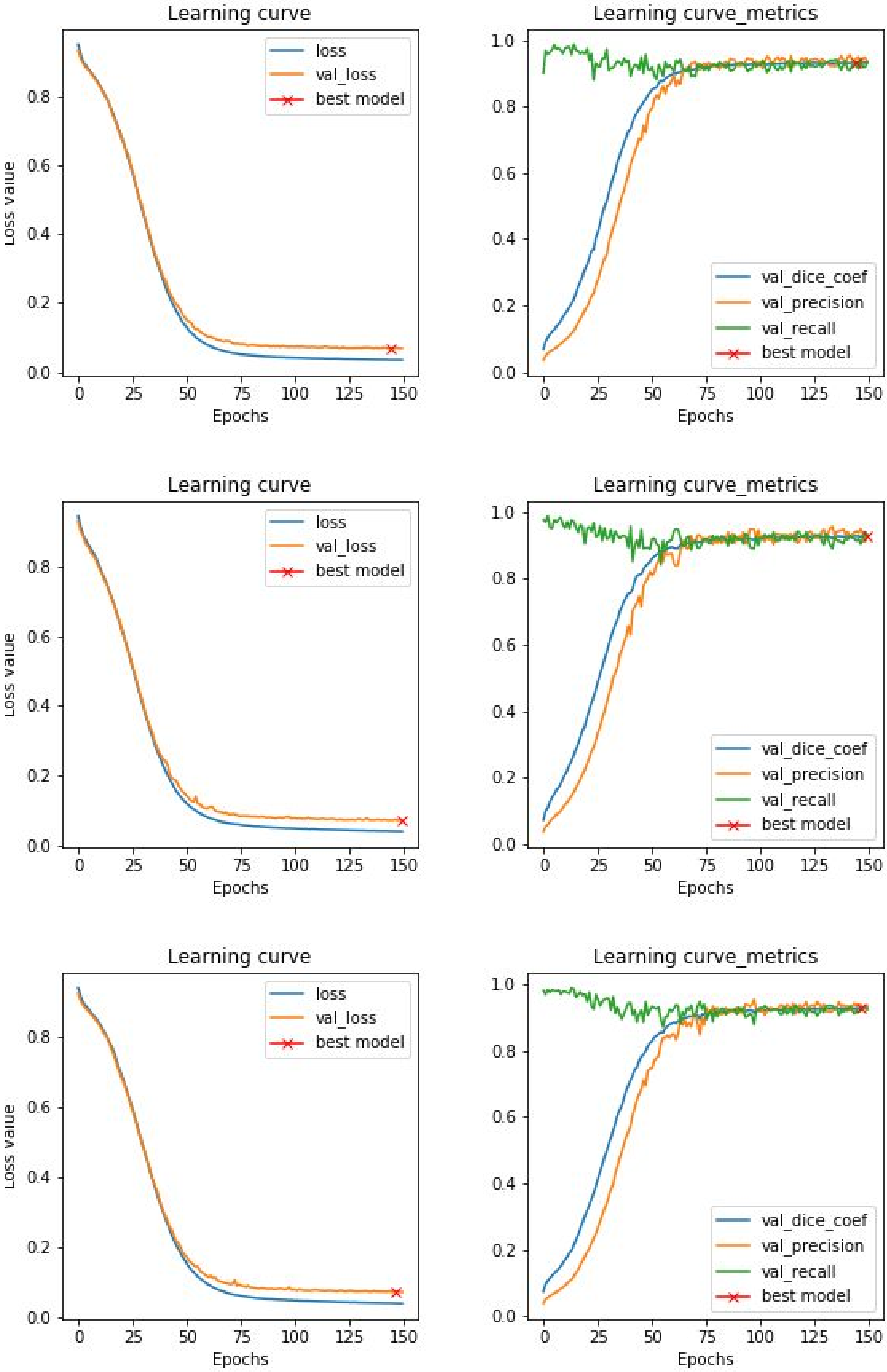
Task2
k=3:


Learning curve
Loss value
Epochs
loss
val_loss
best model
Learning curve_metrics
Epochs
val_dice_coef
val_precision
val_recall
best model
Learning curve
Loss value
Epochs
loss
val_loss
best model
Learning curve_metrics
Epochs
val_dice_coef
val_precision
val_recall
best model
Learning curve
Loss value
Epochs
loss
val_loss
best model
Learning curve_metrics
Epochs
val_dice_coef
val_precision
val_recall
best model

Fold	dice_coef	precision	recall
1	0.9117	0.9228	0.9030
2	0.9097	0.9225	0.8995
3	0.9109	0.9216	0.9032
average	0.9108	0.9223	0.9019

The dice coefficient is lower when we apply weighting to the loss function.We expected that the accuracy would be higher after applying the weighting.

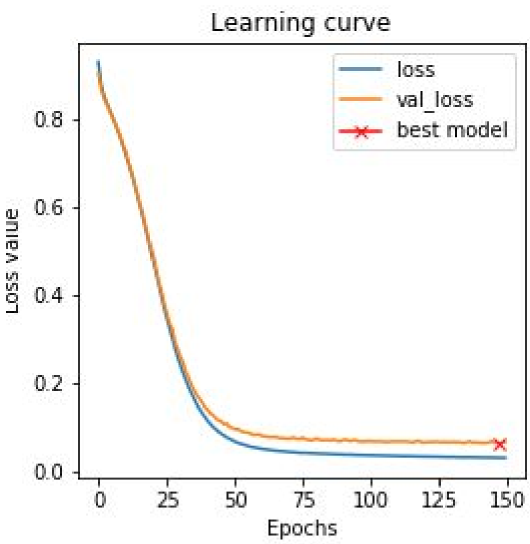
Task3
T=2 (i.e. k=3)

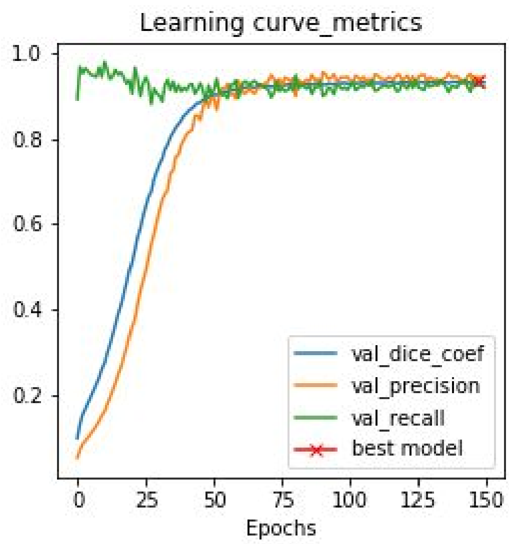

Learning curve
Loss value
Epochs
loss
val_loss
best model
Learning curve_metrics
Epochs
val_dice_coef
val_precision
val_recall
best model
Learning curve
Loss value
Epochs
loss
val_loss
best model
Learning curve_metrics
Epochs
val_dice_coef
val_precision
val_recall
best model
Learning curve
Loss value
Epochs
loss
val_loss
best model
Learning curve_metrics
Epochs
val_dice_coef
val_precision
val_recall
best model

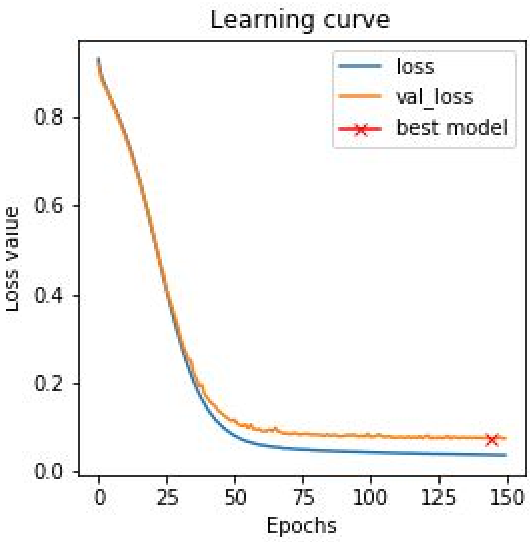
Cycle	dice_coef	precision	recall
0	0.9323	0.9375	0.9282
1	0.9280	0.9263	0.9305
2	0.9293	0.9391	0.9205
average	0.9299	0.9343	0.9264

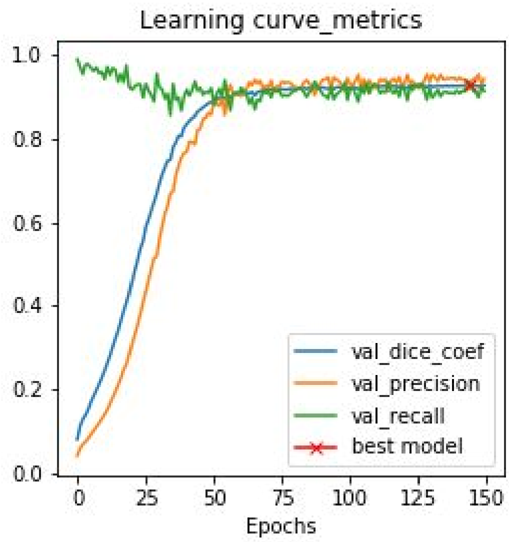
Compared to the results we got from the simple U-Net, autocontext layer doesn't improve the performance on segmentation task.

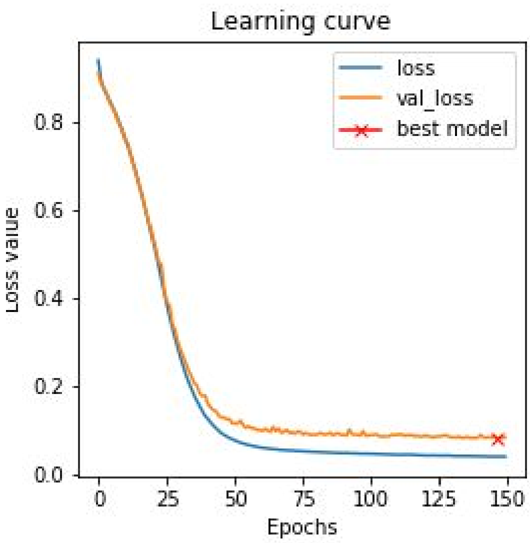
Bonus task:
T=4 (i.e. k=5)

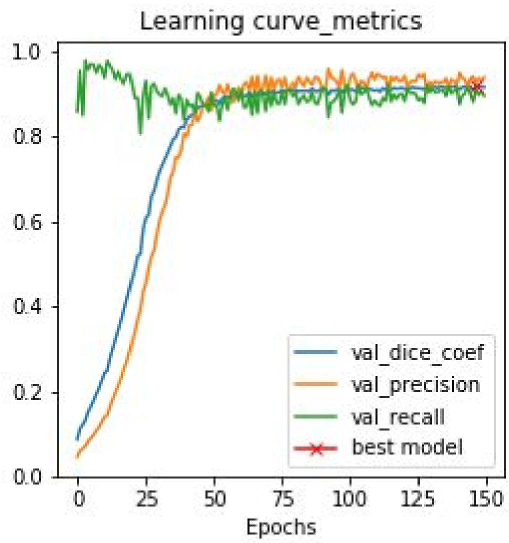


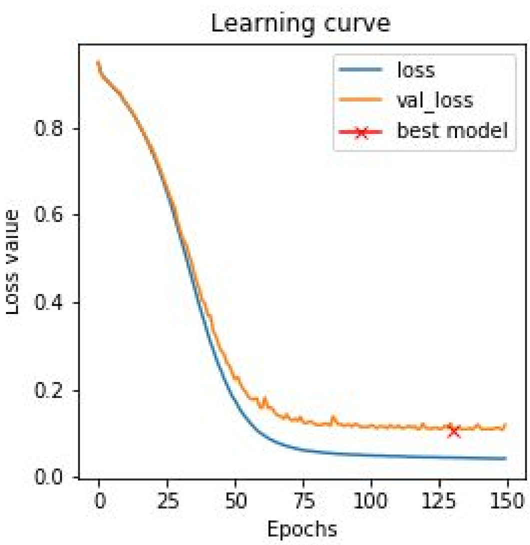


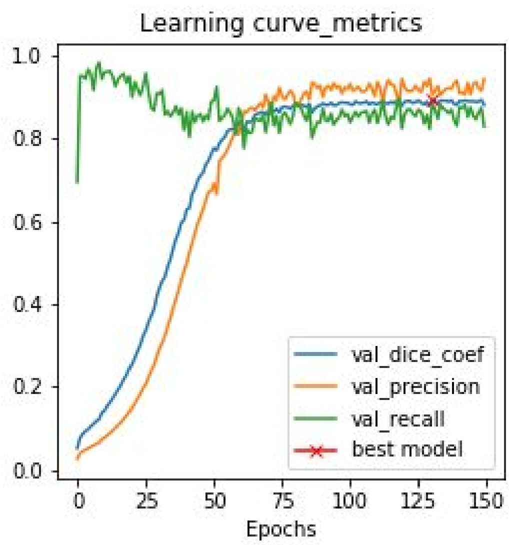












Cycle	dice_coef	precision	recall
0	0.9347	0.9439	0.9262
1	0.9266	0.9312	0.9231
2	0.9194	0.9284	0.9115
3	0.8924	0.9045	0.8827
4	N/A	N/A	N/A
average	0.9183	0.9270	0.9109

(For some reason the program stopped training when the last cycle almost finished, so we didn't get the results of last cycle. And we didn't run the last cycle again because I thought it was over so I deleted all the npy files. If we wanted to get the results of last cycle we had to

run run from the first cycle, which was too time-consuming. So we only show the results of first four cycles to you.)
The accuracy decreases when the the number of cycle increases. Also the average accuracy doesn’t improve compared to smaller number of cycles.

Final observations:
Higher number of folds of simple U-Net leds to the best performance on this dataset.