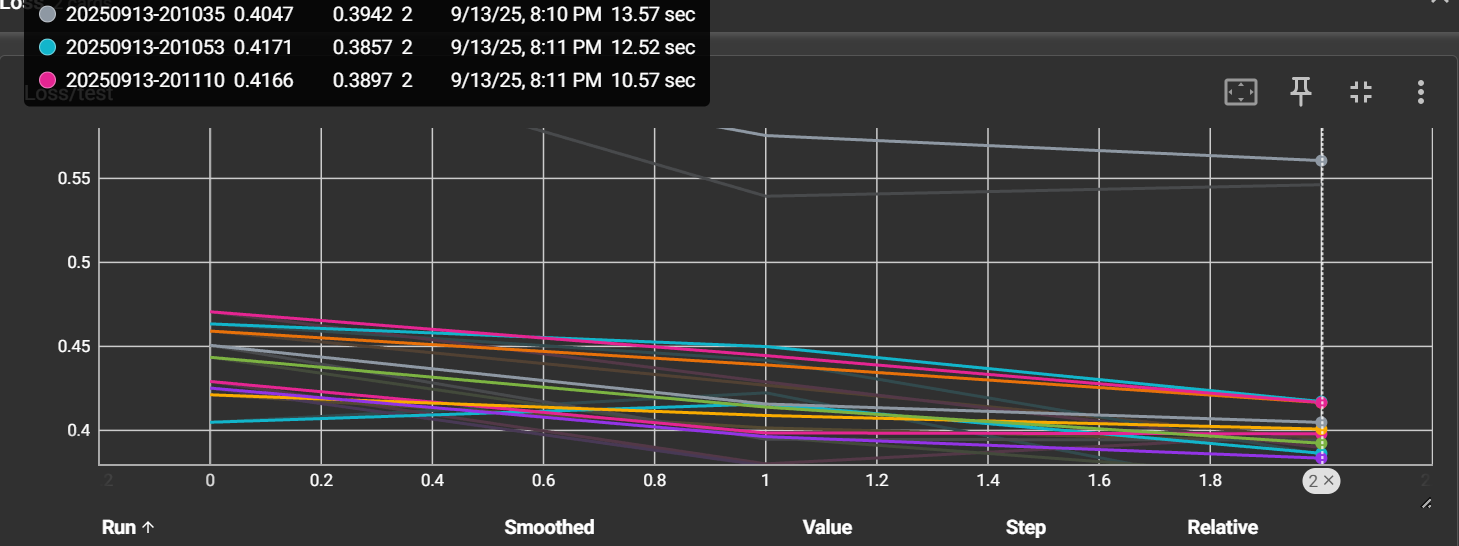
General remarks

Accuracy does not go up and down in a straight line. Because of the gradient descent.

The number of epochs is important

The size of the learning step is important.

Test 0



In the first run the 5th combination scored best with value 0.3711 time 13,03 128,128 units.

Test 1

**5 epochs**

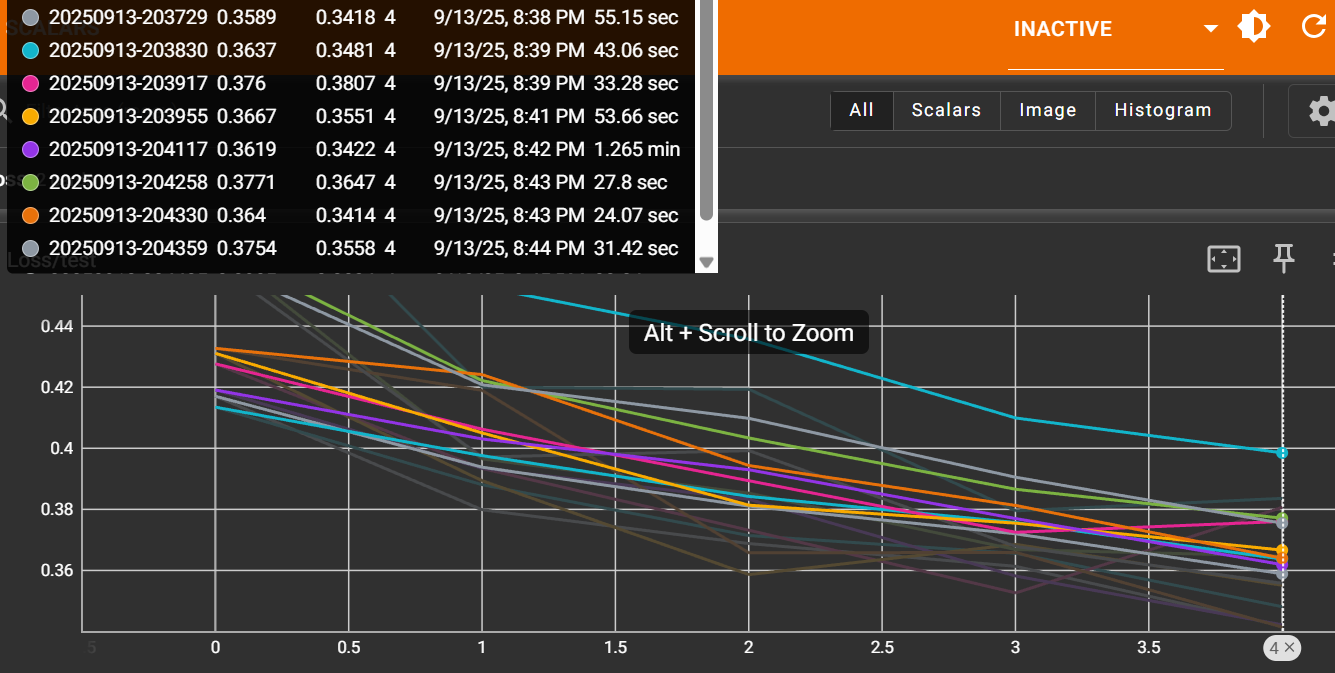
Theory

The 128,128 will still be the best.

The time will not be 2,5 as big but exponentially more

The score will only slightly drop.

Outcome



256,256 was the best. With score 0.8821

**Theory with 10 epochs**

256,256 will be best

Outcome:

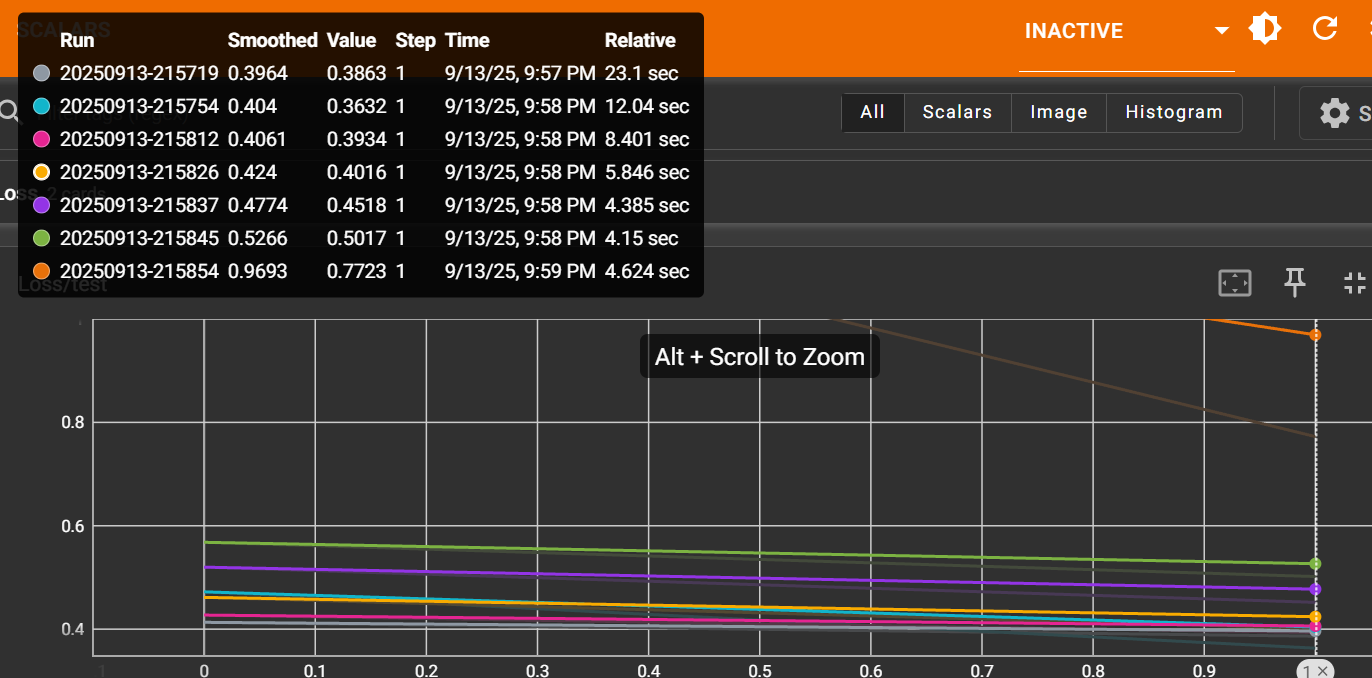
64,256 was best with 0.8830

There is only a slight difference in accuracy. Time took way longer with 1,30 min.

There were even some scores which got worse over time.

Change, add hidden layer, with again half the units. Started at 1024, moving down with 2 epochs.

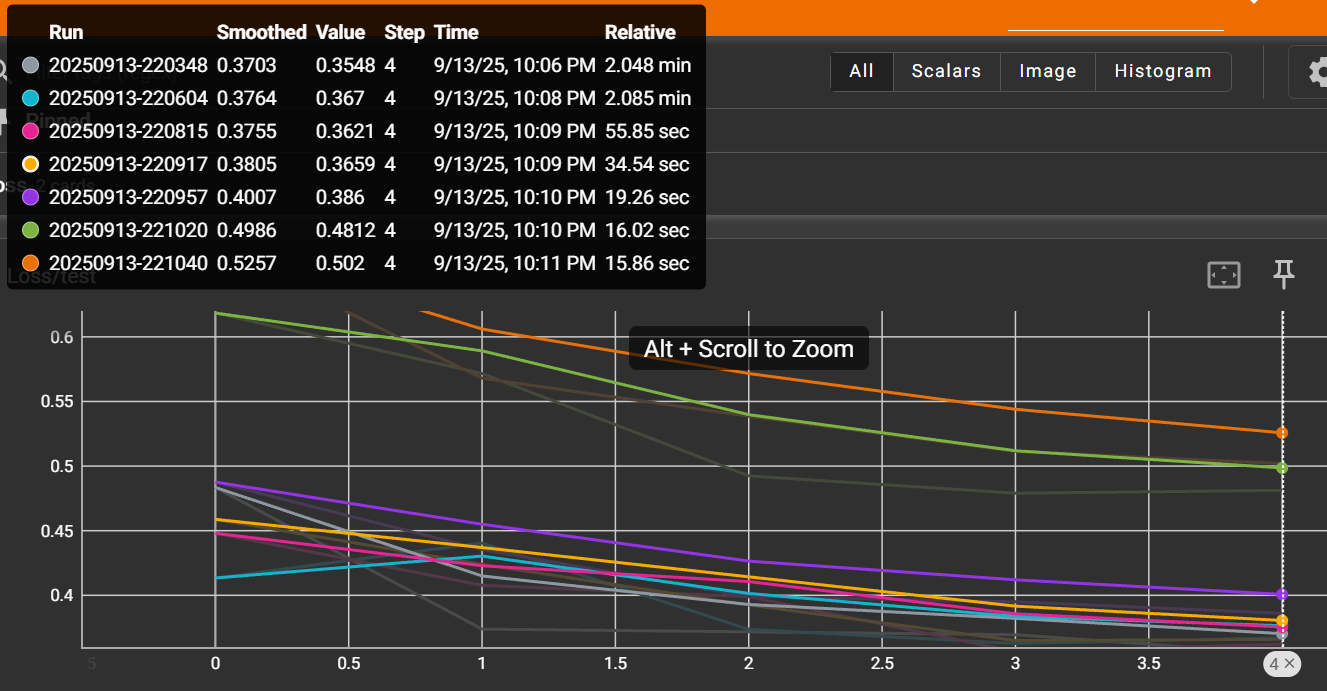
I expect because of the image size that the first setup 1024,512,256 works best. At least the number of inputs matches the pixel number. Because of this I expect the outcom to be better than the first setup. But worse than the 10 epoch setup



Outcome. The second seup was best. 512,256,128. Accuracy of 0.8678. Because of 2 epochs very fast.

**Theory:** Change to 5 epochs

I'm expecting the same setup to be fastests.



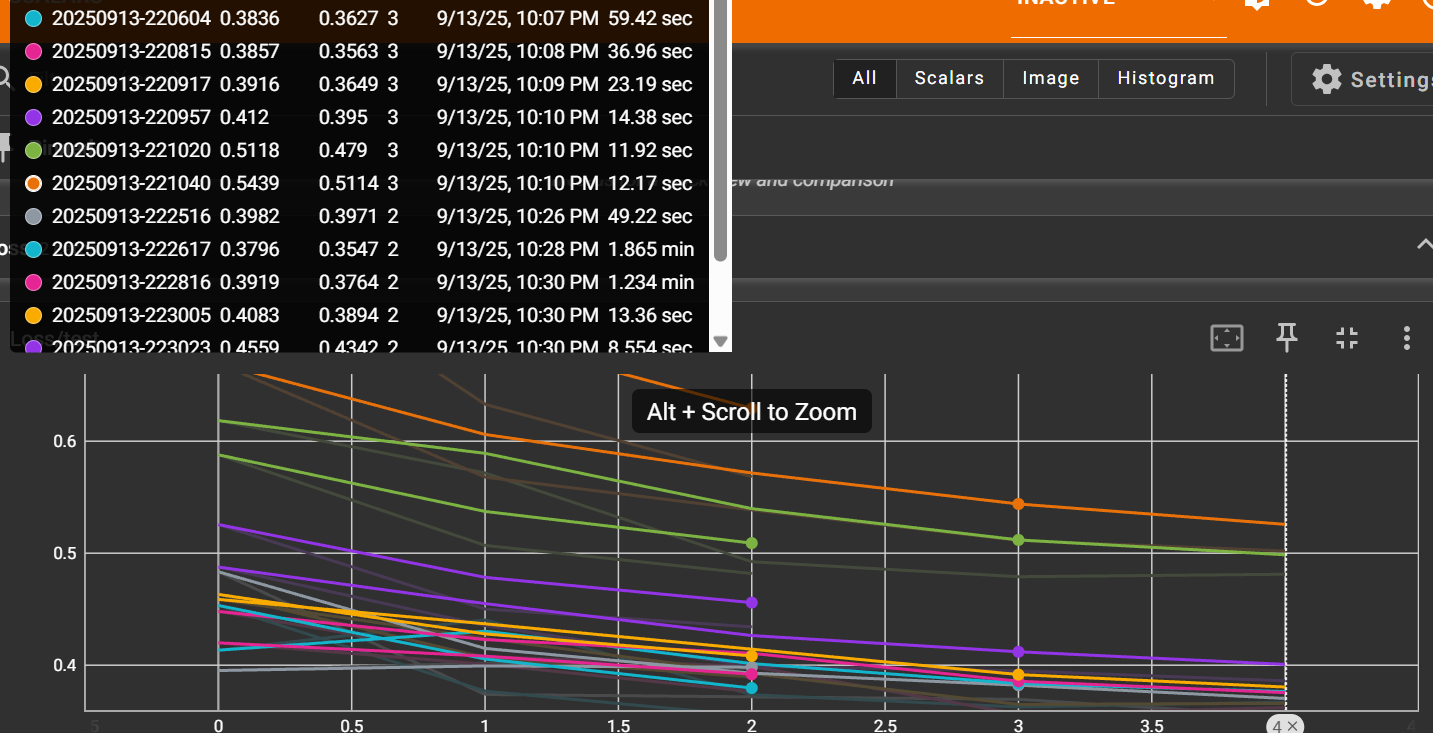
**Outcome:**

The 1024,512,256 and 512,256,128 are more or less the same. After that the models function worse. The number of epochs seem to let the first 2 function more or less the same.

**Theory:**

Change the learning rate to smaller and the number of epochs to 3

I'm expecting it to make all the models perform worse



Outcome:

The second model 512,256,128 still performed best 0.8711. A number of models early stopped. The final accuracy came close

Change: change the batch size to 32 instead of 64

I would say that it becomes more accurate

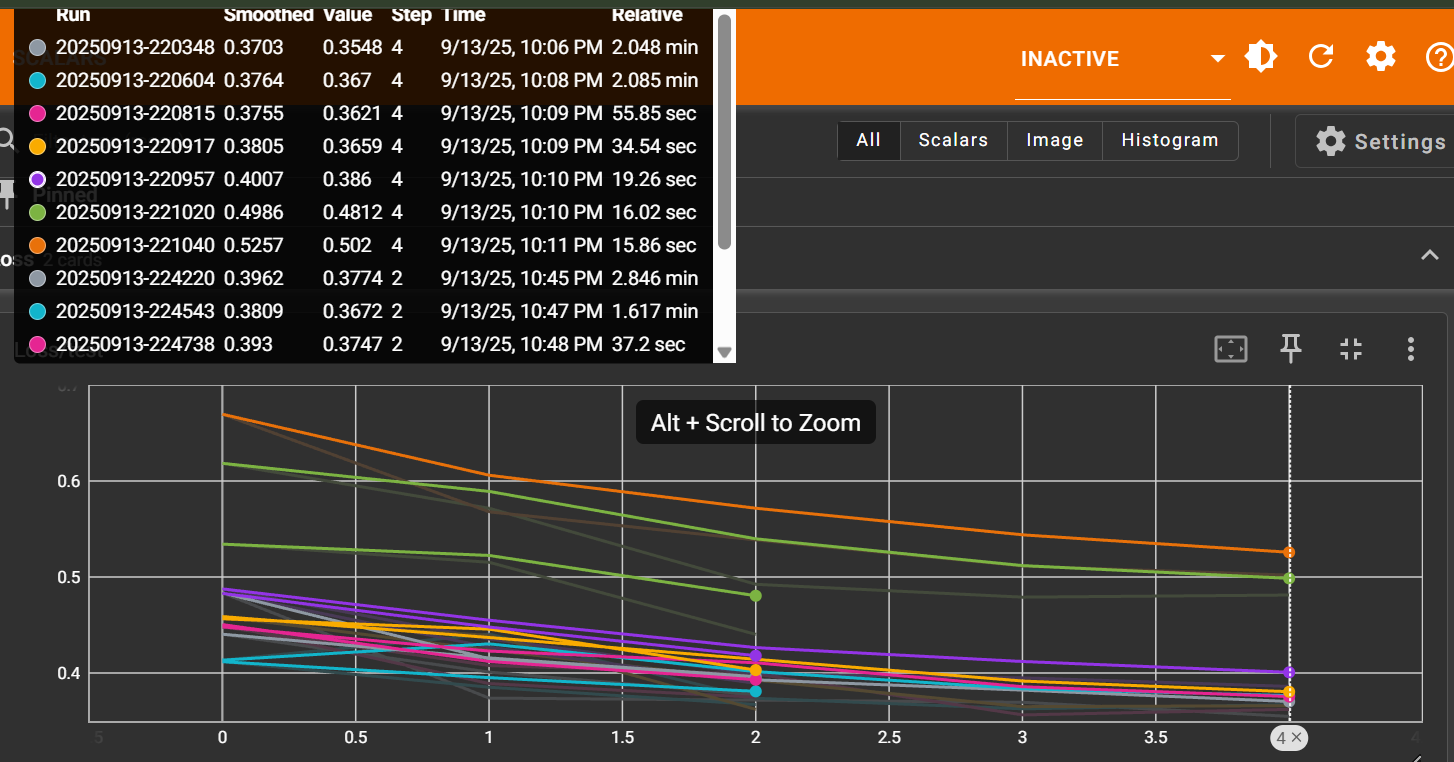
Outcome: The accuracy seems the same 0.8711 time see below picture

Change batchsize to 128.

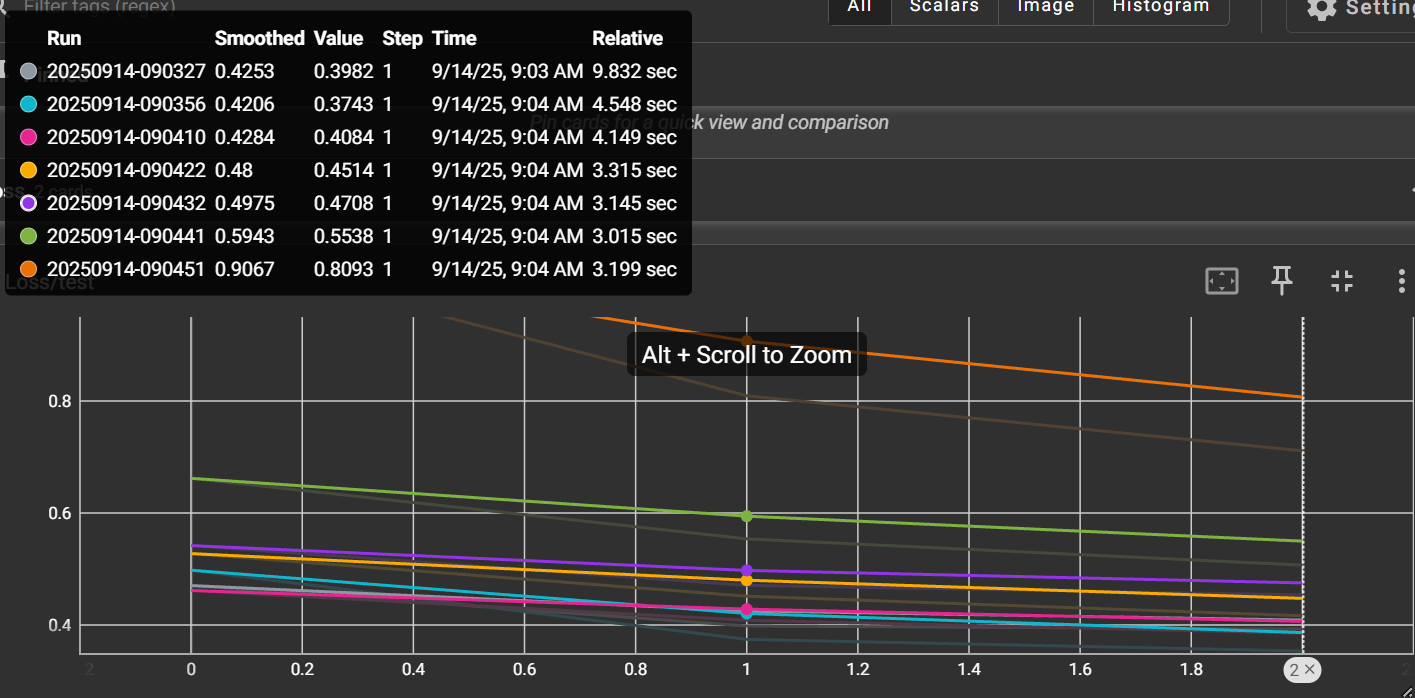
Accuracy will stay the same. Depending on hardware it will be slower or faster. Either there is enough RAM or not, if not slower HDD is going to be used.

Outcome:

The accuracy dropped, the training was much faster. This is logical because there are less steps.



**Change:** Change the loss function from crossentropy to MSE.Maybe this changes the accuracy for a bit, but not much.



Outcome: accuracy 0.8726 Crossentropy,