TDT4265 - Computer Vision and Deep Learning

Vegard Iversen, Sebastian Skogen Raa

Mar 2022

1a

Q: Perform spatial convolution by hand on the image in Figure 1a using the kernel in Figure 1b. The convolved image should be 3×5. You are free to choose how you handle boundary conditions, and state how you handle them in the report.

A:

Table 1: Rotated cw sobel kernel by π.

1	0	-1
2	0	-2
1	0	-1

Table 2: Image with padding

0	0	0	0	0	0	0
0	1	0	2	3	1	0
0	3	2	0	7	0	0
0	0	6	1	1	4	0
0	0	0	0	0	0	0

First row (using image axis)

$$\begin{aligned}[0,0] =& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\ & 0\cdot -2 + 1\cdot 0 + 0\cdot 2 + \\ & 0\cdot -1 + 3\cdot 0 + 2\cdot 1 + \\ =& 2\end{aligned}$$

$$\begin{aligned}[0,1] =& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\ & 1\cdot -2 + 0\cdot 0 + 2\cdot 2 + \\ & 3\cdot -1 + 2\cdot 0 + 0\cdot 1 + \\ =& -1\end{aligned}$$

$$\begin{aligned}[0,2] =& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\ & 0\cdot -2 + 2\cdot 0 + 3\cdot 2 + \\ & 2\cdot -1 + 0\cdot 0 + 7\cdot 1 + \\ =& 11\end{aligned}$$

$$\begin{aligned}[0,3] =& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\ & 2\cdot -2 + 3\cdot 0 + 1\cdot 2 + \\ & 0\cdot -1 + 7\cdot 0 + 0\cdot 1 + \\ =& -2\end{aligned}$$

$$\begin{aligned}[0,4] =& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\ & 3\cdot -2 + 1\cdot 0 + 0\cdot 2 + \\ & 7\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\ =& -13\end{aligned}$$

Second row

$$\begin{aligned}[1,0] =& 0\cdot -1 + 1\cdot 0 + 0\cdot 1 + \\ & 0\cdot -2 + 3\cdot 0 + 2\cdot 2 + \\ & 0\cdot -1 + 0\cdot 0 + 6\cdot 1 + \\ =& 10\end{aligned}$$

$$\begin{aligned}[1,1] =& 1\cdot -1 + 0\cdot 0 + 2\cdot 1 + \\ & 3\cdot -2 + 2\cdot 0 + 0\cdot 2 + \\ & 0\cdot -1 + 6\cdot 0 + 1\cdot 1 + \\ =& -4\end{aligned}$$

$$\begin{aligned}[1,2] =& 0\cdot -1 + 2\cdot 0 + 3\cdot 1 + \\ & 2\cdot -2 + 0\cdot 0 + 7\cdot 2 + \\ & 6\cdot -1 + 1\cdot 0 + 1\cdot 1 + \\ =& 8\end{aligned}$$

$$\begin{aligned}[1,3] =& 2\cdot -1 + 3\cdot 0 + 1\cdot 1 + \\ & 0\cdot -2 + 7\cdot 0 + 0\cdot 2 + \\ & 1\cdot -1 + 1\cdot 0 + 4\cdot 1 + \\ =& 2\end{aligned}$$

$$\begin{aligned}[1,4] =& 3\cdot -1 + 1\cdot 0 + 0\cdot 1 + \\ & 7\cdot -2 + 0\cdot 0 + 0\cdot 2 + \\ & 1\cdot -1 + 4\cdot 0 + 0\cdot 1 + \\ =& -18\end{aligned}$$

Third row

$$\begin{aligned}
[2,0] =& 0\cdot -1 + 3\cdot 0 + 2\cdot 1 + \\
& 0\cdot -2 + 0\cdot 0 + 6\cdot 2 + \\
& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\
=& 14
\end{aligned}$$

$$\begin{aligned}
[2,1] =& 3\cdot -1 + 2\cdot 0 + 0\cdot 1 + \\
& 0\cdot -2 + 6\cdot 0 + 1\cdot 2 + \\
& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\
=& -1
\end{aligned}$$

$$\begin{aligned}
[2,2] =& 2\cdot -1 + 0\cdot 0 + 7\cdot 1 + \\
& 6\cdot -2 + 1\cdot 0 + 1\cdot 2 + \\
& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\
=& -5
\end{aligned}$$

$$\begin{aligned}
[2,3] =& 0\cdot -1 + 7\cdot 0 + 0\cdot 1 + \\
& 1\cdot -2 + 1\cdot 0 + 4\cdot 2 + \\
& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\
=& 6
\end{aligned}$$

$$\begin{aligned}
[2,4] =& 7\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\
& 1\cdot -2 + 4\cdot 0 + 0\cdot 2 + \\
& 0\cdot -1 + 0\cdot 0 + 0\cdot 1 + \\
=& -9
\end{aligned}$$

Result:

Table 3: Final result after filtering with Sobel kernel

2	-1	11	-2	-13
10	4	8	2	-18
14	-1	-5	6	-9

1b

Q:
CNNs are known to be invariant to small translational shifts in the input image (for example a 1-pixel shift). Of the following layers, which layer reduces the sensitivity to translationial variations in the input?

A:
The convolutional layer does this at it learns filters to detect features in the images independently of how the image is translated. This comes as a consequence of he translationally-invariant structure of images.

1c

Q:

Given a single convolutional layer with a stride of 1, kernel size of 5 × 5, and 6 filters. If you want the output shape (Height × Width) of the convolutional layer to be equal to the input image, how much padding should you use on each side?

A:

Because of symmetry we can use either width or height for the computation, we use width W:

$$\begin{aligned} P &= \frac{F + SW - S - W}{2} \\ &= \frac{5 + 1 \cdot W - 1 - W}{2} \\ &= \frac{4}{2} \\ &= 2 \end{aligned} \tag{1}$$

1d

Q:

Given a single convolutional layer with a stride of 1, kernel size of 5 × 5, and 6 filters. If you want the output shape (Height × Width) of the convolutional layer to be equal to the input image, how much padding should you use on each side?

$$\begin{aligned} F &= 2P - W_2 S + W_1 + S \\ &= 0 - 504 + 512 + 1 \\ &= 9 \end{aligned} \tag{2}$$

1e

Q:

If subsampling is done after the first convolutional layer, using neighborhoods of size 2 × 2, with a stride of 2, what are the spatial dimensions of the pooled feature maps in the first pooling layer? Give the answer as (Height) × (Width).

A:

$$\begin{aligned} W_2 &= \frac{W_1 - F}{S} + 1 \\ &= \frac{5042}{2} + 1 \\ &= 252 \end{aligned} \tag{3}$$

1f

Q:

The spatial dimensions of the convolution kernels in the second layer are 3 × 3. Assuming no padding and a stride of 1, what are the sizes of the feature maps in the second layer? Give the answer as

(Height) $\times$ (Width).
A:

$$\begin{aligned} W_2 &= (W_1 F + 2P)/S + 1 \\ W_2 &= (2523 + 2 \cdot 0)/1 + 1 \\ W_2 &= 250 \end{aligned} \tag{4}$$

1g

Q:
Table 1 shows a simple CNN. How many parameters are there in the network? In this network, the number of parameters is the number of weights + the number of biases. Assume the network takes in an RGB image as input and the image is square with a width of 32.
A:
Number of parameters in conv layer:

$$Params_{Conv} = ((height \cdot width \cdot depth_{previous}) + biases) \cdot depth_{current}) \tag{5}$$

Here height and width is that of the filter size/receptive field size. No added parameters for pooling. We calculate params for all 3 convolutional layers:

$$\begin{aligned} Params_{Conv1} &= ((5 \cdot 5 \cdot 3) + 1) \cdot 32) \\ &= 2432 \end{aligned} \tag{6}$$

$$\begin{aligned} Params_{Conv2} &= ((5 \cdot 5 \cdot 32) + 1) \cdot 64) \\ &= 51264 \end{aligned} \tag{7}$$

$$\begin{aligned} Params_{Conv3} &= ((5 \cdot 5 \cdot 64) + 1) \cdot 128) \\ &= 204928 \end{aligned} \tag{8}$$

We need to find the size of the feature map out that goes into layer 4, we do this by using the same equation as in Equation 3. Doing this 3 times an we end up with:

$$\begin{aligned} W_2 &= (W_1 F)/S + 1 \\ &= (82)/2 + 1 \\ &= 4 \end{aligned} \tag{9}$$

We can then find the parameter count for the fully connected layers:

$$\begin{aligned} Params_{FC1} &= (n_{in} + n_{bias}) \cdot n_{out}) \\ &= (128 \cdot 4 \cdot 4 + 1) \cdot 64) \\ &= 131136 \end{aligned} \tag{10}$$

$$\begin{aligned} Params_{FC2} &= (n_{in} + n_{bias}) \cdot n_{out}) \\ &= (64 + 1) \cdot 10) \\ &= 650 \end{aligned} \tag{11}$$

Adding all params up:

$$\begin{aligned} Params_{total} &= 2432 + 51264 + 204928 + 131136 + 650 \\ Params_{total} &= 390410 \end{aligned} \tag{12}$$

2a

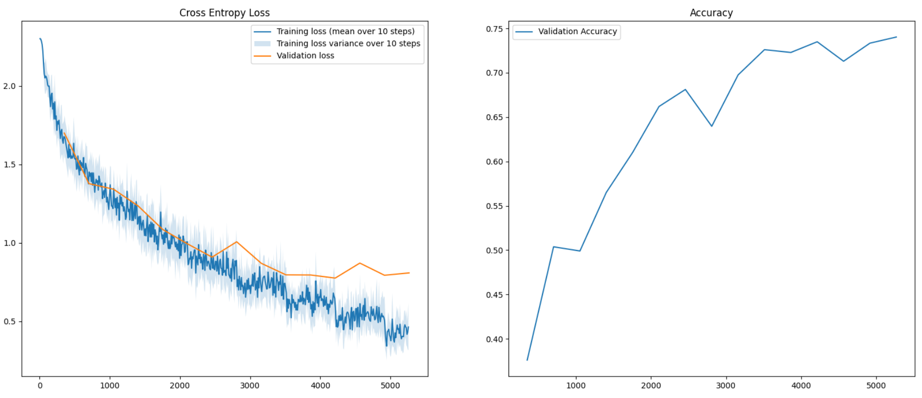

Figure 1: Accuracy plot

2c

Notes: Using a 3x3 filter did not give any worse result than 5x5, but mb some more overfitting. 11x11 kernel does not give worse acc but a bit higher val loss. so overfitting

3a

I did many different model for fun. These models have the same architecture as task2 but change other things.

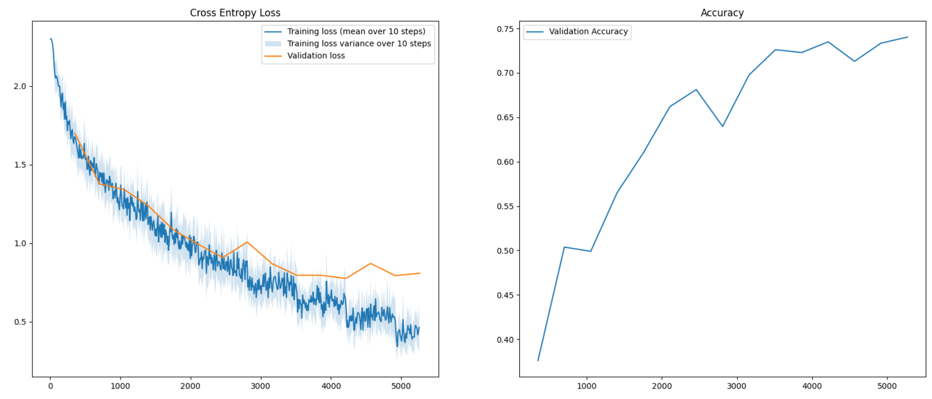

Figure 2: Base model from task 2

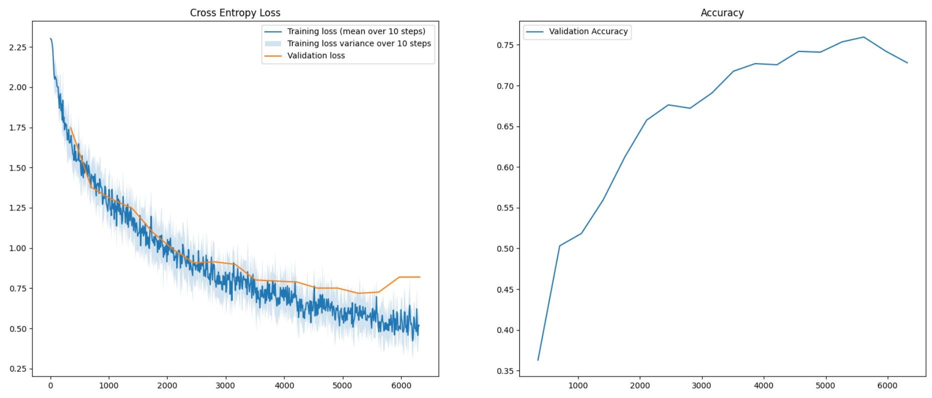

Figure 3: Augmentation 1

in data augmentation1 have we added a randomHorizontal flip with probability of 0.5 (50%) The reason this transform can be useful is that a image would still be valid after a horizontal flip. Can give more variation in the data.

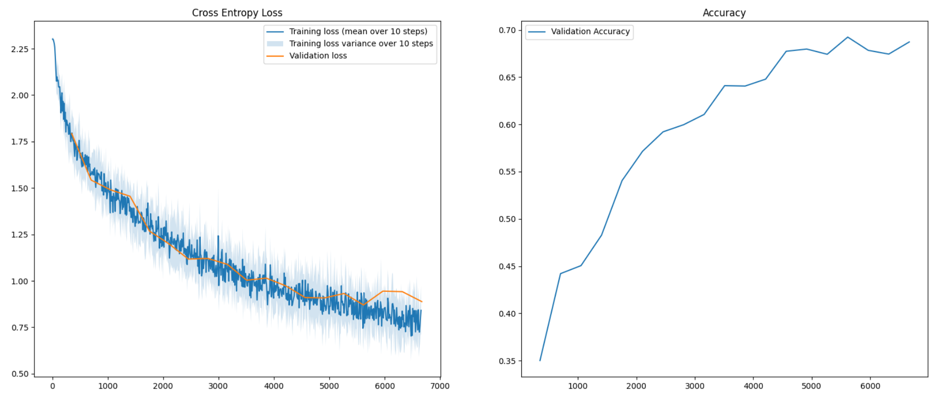

Figure 4: Augmentation 2

in data augmentation2 have we added a randomHorizontal flip with probability of 0.5 (50%), and a RandomPerspective(0.5,0.5). Same argument as over.

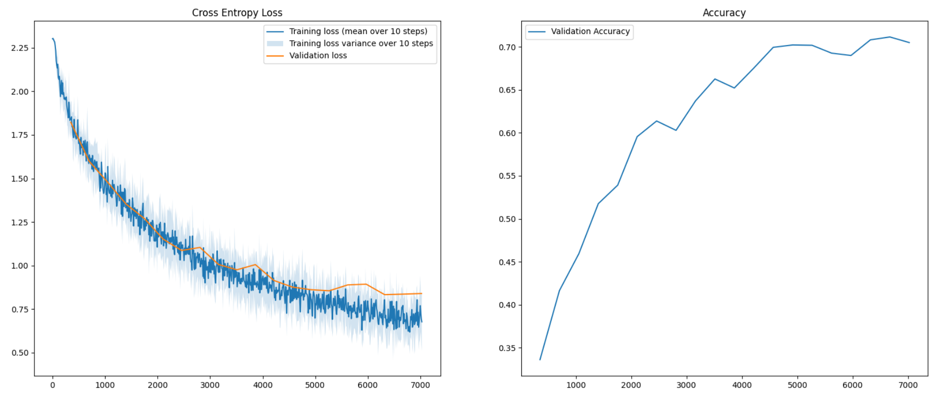

Figure 5: Augmentation 3

in data augmentation3 have we added a randomHorizontal flip with probability of 0.5 (50%) and RandomAdjustsharpness(2)(gives sharper images). same argument as the first.

One of the goals for data augmentation is to get more varied data, but also we would like to get more data. The method we used earlier did not give more data, it just changed part of the dataset. Now we have loaded in the dataset twice and used transforms on one of them and no other transform other than normalizing. then we concatenate them so we have basically double our data. A weakness here is that we can get duplicates which can create memorization/overfit. In the models under we have used this technique. We will only adress the one dataset using the transform

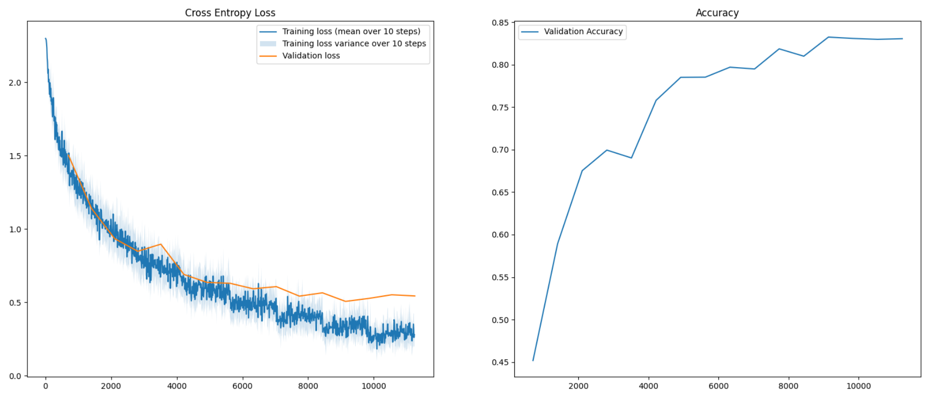

Figure 6: Augmentation 4

in data augmentation4 have we added a randomHorizontal flip with probability of 0.5 (50%) (would maybe think that a horisontal flip of all gave better result but testing showed otherwise).

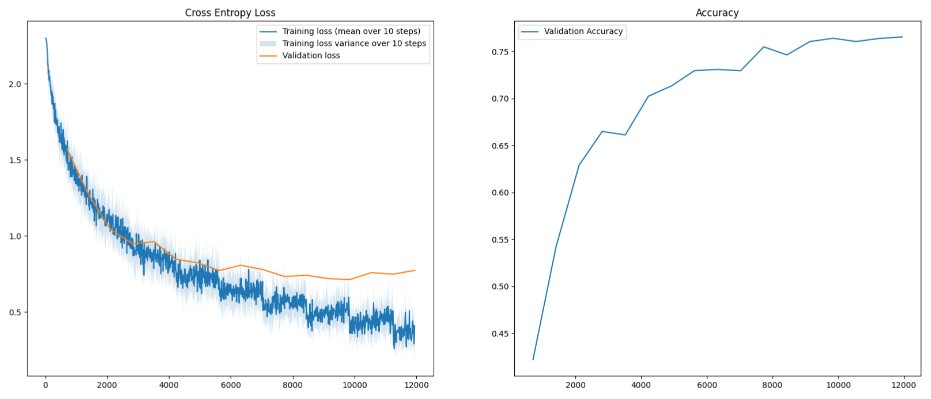

Figure 7: Augmentation5

in data augmentation5 have we added a randomHorizontal flip with probability of 1 (100%) here we did not hit early stop, so tried with more epochs :

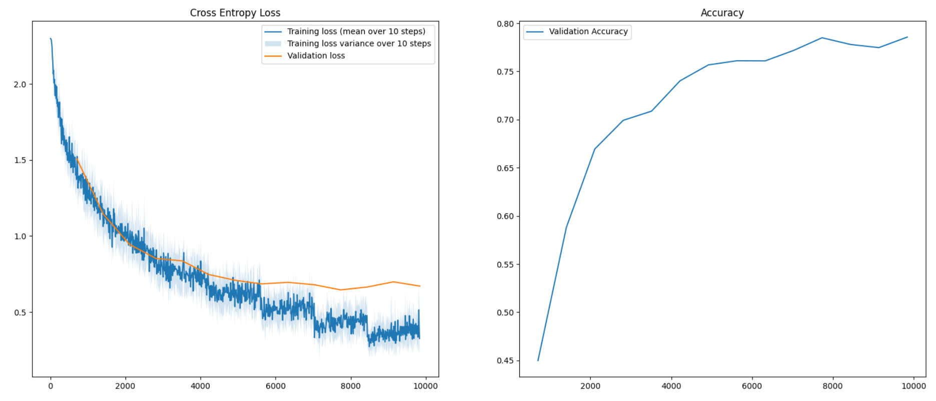

Figure 8: Augmentation5v2

We can see a decent increase in validation accuracy, but also a small increase in the validation loss. This could there seem to overfit more.

The models under have increased epoch as it did not seem to converge with 10 epochs. increased to 25 and early stop to 10.

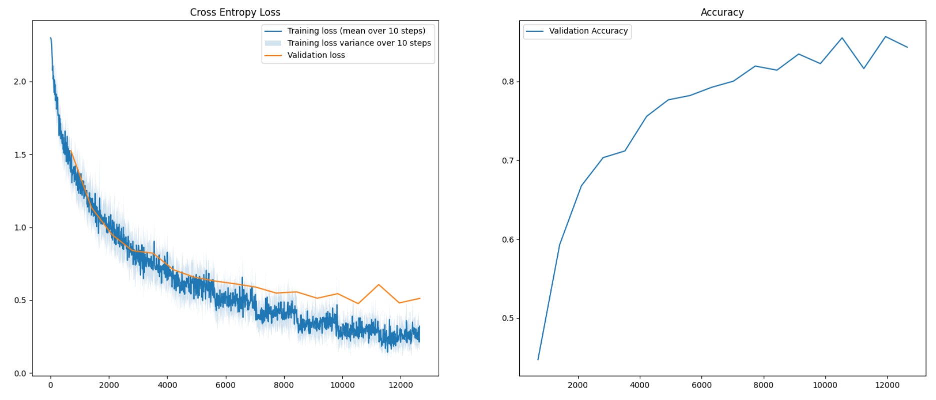

Figure 9: Augmentation6

in data augmentation6 have we added a randomHorizontal flip with probability of 0.5 (50%) and a randomgrayscale(0.1)

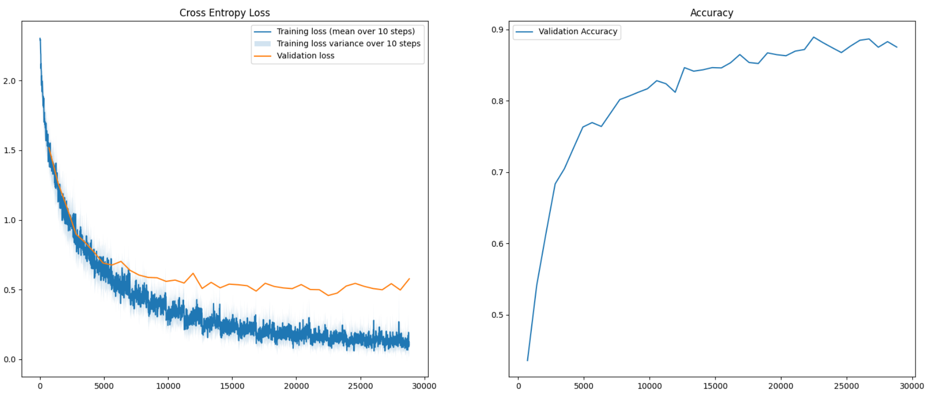

Figure 10: Augmentation7

in data augmentation6 have we added a randomHorizontal flip with probability of 0.5 (50%), Randomgrayscale(0.1) and RandomAutocontrast(0.2).

This data augmentation the best accuracy. But at the same time more indication of overfitting. From now I will use the transformation in this data augmentation as base, and try and change other parameters to improve the model.

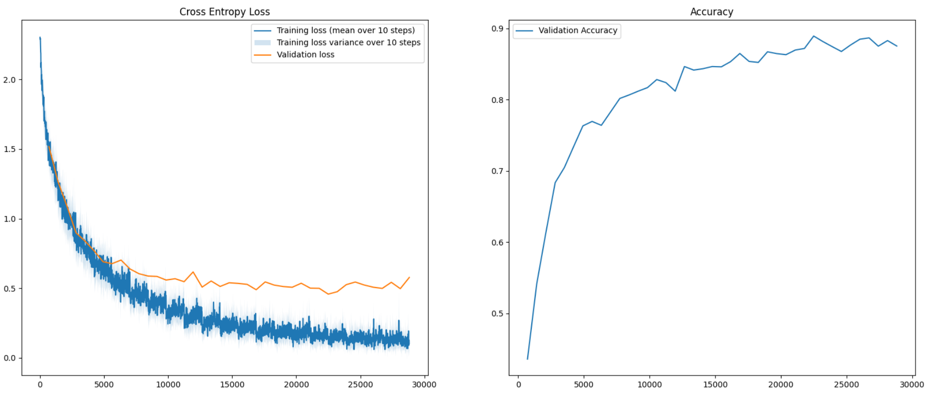

Figure 11: Augmentation7

Table 4: Description of modelv2 with the data augmentations over.

Layer	Layer Type	Number of Hidden Units	Activation Function
1	Conv2D	32	ReLU
1	MaxPool2D	-	-
2	Conv2D	64	ReLU
2	MaxPool2D	-	-
3	Conv2D	64	ReLU
3	MaxPool2D	-	-
4	Conv2D	128	ReLU
4	MaxPool2d	-	-
	Flatten	-	-
5	Fully-Connected	64	ReLU
6	Fully-Connected	10	Softmax

Here the maxpool stride is 2 and size is [2,2]. In the conv layer the stride is 1, and padding is 2. With a filter size = 5. Modelv2 plot:

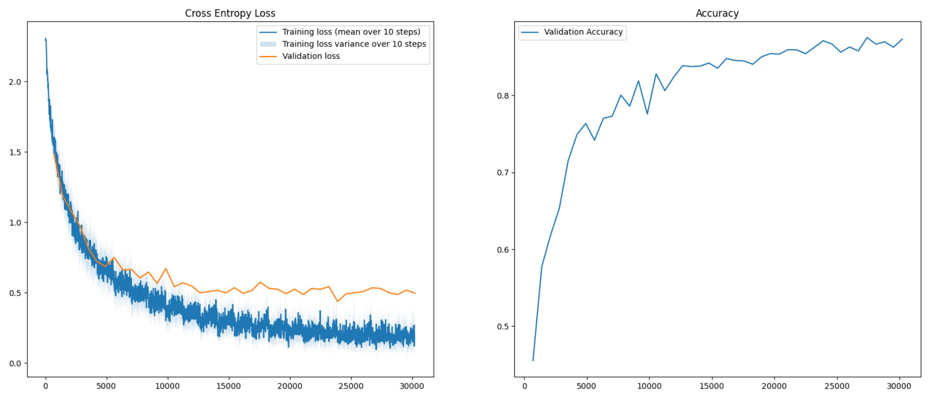

Figure 12: Augmentation7 modelv2

Table 5: Description of modelv3 with the data augmentations over.

Layer	Layer Type	Number of Hidden Units	Activation Function
1	Conv2D	32	ReLU
1	MaxPool2D	-	-
2	Conv2D	64	ReLU
2	MaxPool2D	-	-
3	Conv2D	256	ReLU
3	MaxPool2D	-	-
4	Conv2D	128	ReLU
4	MaxPool2d	-	-
	Flatten	-	-
5	Fully-Connected	64	ReLU
6	Fully-Connected	10	Softmax

Here the maxpool stride is 2 and size is [2,2]. In the conv layer the stride is 1, and padding is 2. With a filter size = 5. Modelv3 plot:

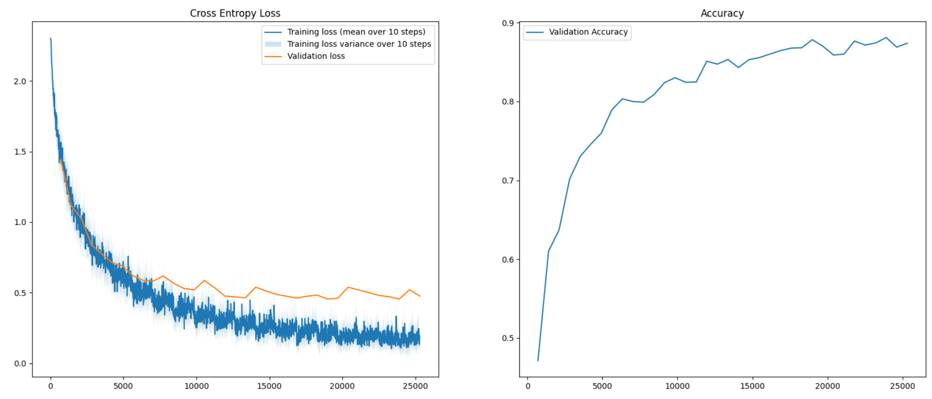

Figure 13: Augmentation7 modelv3

Table 6: Description of modelv4 with the data augmentations over.

Layer	Layer Type	Number of Hidden Units	Activation Function
1	Conv2D	32	ReLU
1	Dropout(0.2)	-	-
1	MaxPool2D	-	-
2	Conv2D	64	ReLU
2	Dropout(0.2)	-	-
2	MaxPool2D	-	-
3	Conv2D	256	ReLU
3	MaxPool2D	-	-
4	Conv2D	128	ReLU
4	MaxPool2d	-	-
	Flatten	-	-
5	Fully-Connected	64	ReLU
6	Fully-Connected	10	Softmax

Here the maxpool stride is 2 and size is [2,2]. In the conv layer the stride is 1, and padding is 2. With a filter size = 5. Modelv4 plot:

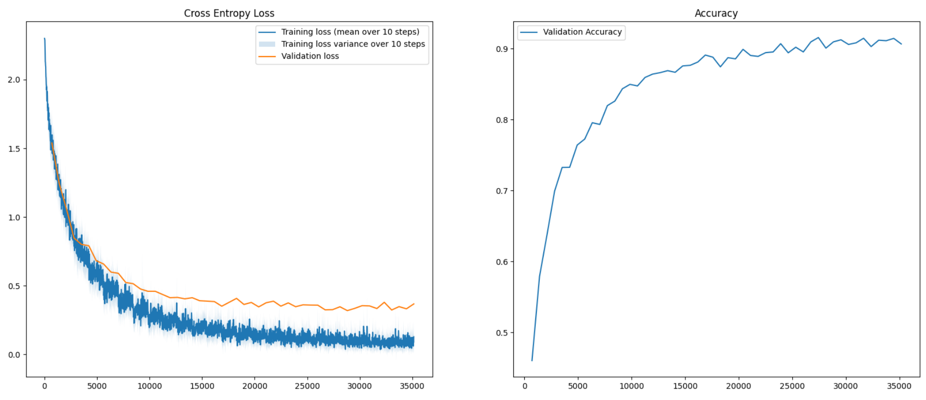

Figure 14: Augmentation7 modelv4

Table 7: Description of modelv5 with the data augmentations over.

Layer	**Layer Type**	**Number of Hidden Units**	**Activation Function**
1	Conv2D	32	ReLU
1	Dropout(0.2)	-	-
1	MaxPool2D	-	-
2	Conv2D	64	ReLU
2	Dropout(0.2)	-	-
2	MaxPool2D	-	-
3	Conv2D	256	ReLU
3	MaxPool2D	-	-
4	Conv2D	128	ReLU
4	MaxPool2d	-	-
	Flatten	-	-
5	Fully-Connected	64	ReLU
6	Fully-Connected	10	Softmax

Here the maxpool stride is 2 and size is [2,2]. In the conv layer the stride is 1, and padding is 2. With a filter size = 5. Now we have used Adam as optimizer with a $3e^{-4}$ learning rate Modelv5 plot:

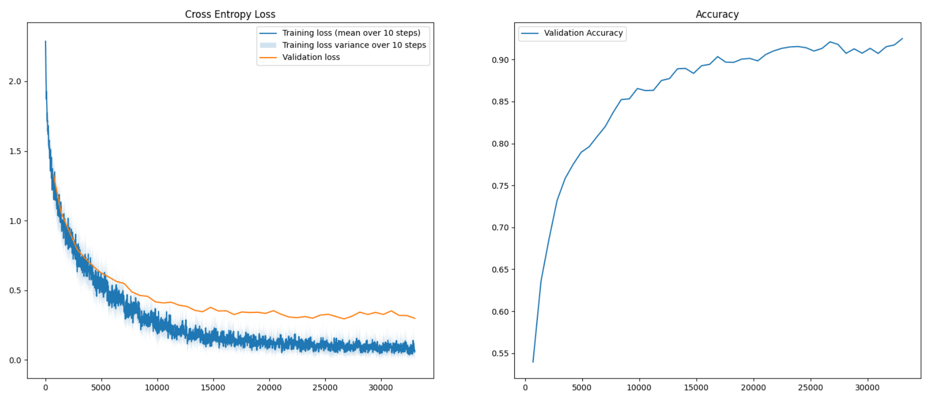

Figure 15: Augmentation7 modelv5

As we see with Adam optimizer the model learns much quicker.

Table 8: Description of modelv6 with the data augmentations over.

Layer	**Layer Type**	**Number of Hidden Units**	**Activation Function**
1	Conv2D	32	ReLU
1	BatchNorm2d	-	-
1	Dropout(0.2)	-	-
1	MaxPool2D	-	-
2	Conv2D	64	ReLU
2	BatchNorm2d	-	-
2	Dropout(0.2)	-	-
2	MaxPool2D	-	-
3	Conv2D	256	ReLU
3	BatchNorm2d	-	-
3	MaxPool2D	-	-
4	Conv2D	128	ReLU
4	BatchNorm2d	-	-
4	MaxPool2d	-	-
	Flatten	-	-
5	Fully-Connected	64	ReLU
5	BatchNorm1d	-	-
6	Fully-Connected	64	ReLU
6	BatchNorm1d	-	-
7	Fully-Connected	10	Softmax

Here the maxpool stride is 2 and size is [2,2]. In the conv layer the stride is 1, and padding is 2. With a filter size = 5. Now we have used Adam as optimizer with a $3e^{-4}$ learning rate Modelv6 plot:

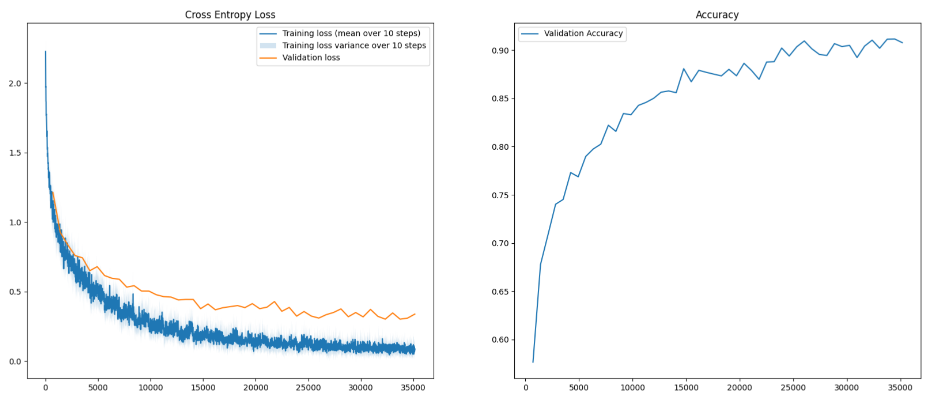

Figure 16: Augmentation7 modelv6

We have now used modelv6 as base for the next changes, which are kernel sizes.

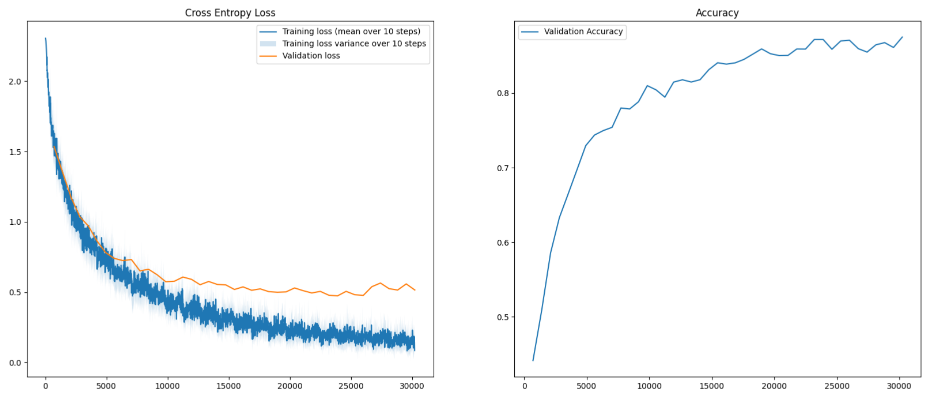

Figure 17: Augmentation7 modelv6 and kernel size 3

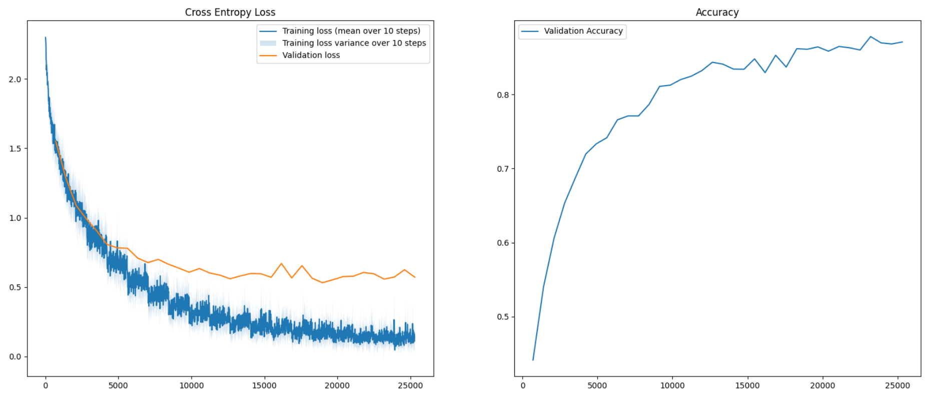

Figure 18: Augmentation7 modelv6 and kernel size 11

Here we can see that there is not much difference between the different kernel size, especially between 3 and 5. With kernel size of 11 the validation loss is a bit higher which can mean that it overfits more. This could be because with the bigger kernel you loose out of important details.

3b

Table 9: Final train loss, final train accuracy, final validation accuracy and average test accuracy for the two networks.

Model	**Train loss**	**Train accuracy**	**Validation accuracy**	**Validation loss**
Modelv6(adam)	0.092	97.0 %	91.0 %	0.31 %
Modelv6(sgd)	0.069	97.7 %	91.4 %	0.31 %

c

The biggest leap in preformance was when we increased the dataset size with augmented images. The reason for the improvement is that more data usally gives better results. As some of the data would be duplicated we saw an increase in overfitting also, but minimal compared to the accuracy. We could also see that changing optimizer changed the learning rate substantially. Adam learned much quicker than SGD, but SGD did generilze a bit more, which corresponds to research on this field. The bigger network also gave better results, did not test that many possibilities but the base network was pretty small. As seen in the plots from task 3a, we tested alot of different combination of transform, and some worked and some made it worse. The transformed that worked are kept in the best model (modelv6).

d

The method that gave the largest amount of improvement are discussed above. The plot for this can be seen in figure 1 to figure 9.

e

this is done above. NOTE: I know we have "cheated" a bit by increasing the number of epochs and early stop. But when looking at the graphs with 10 epochs it did not seem to converge yet.

f

There are some signs of overfitting. As the difference in validation loss and training loss are increased. Some of the reasons why happens are discussed above.

NOTE: I know we have "cheated" a

4a

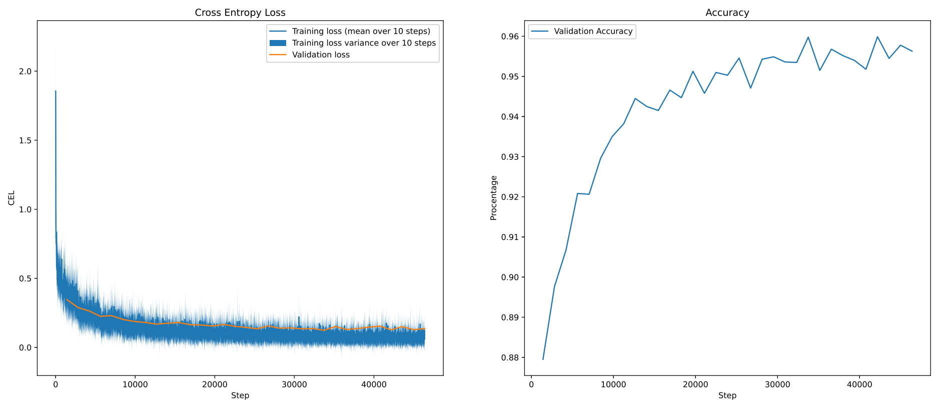

Figure 19: Zebra image activation in first convolution layer using adam. Stoped at Epoch: 11, Batches per seconds: 22.68, Global step: 33744, Validation Loss: 0.12, Validation Accuracy: 0.960

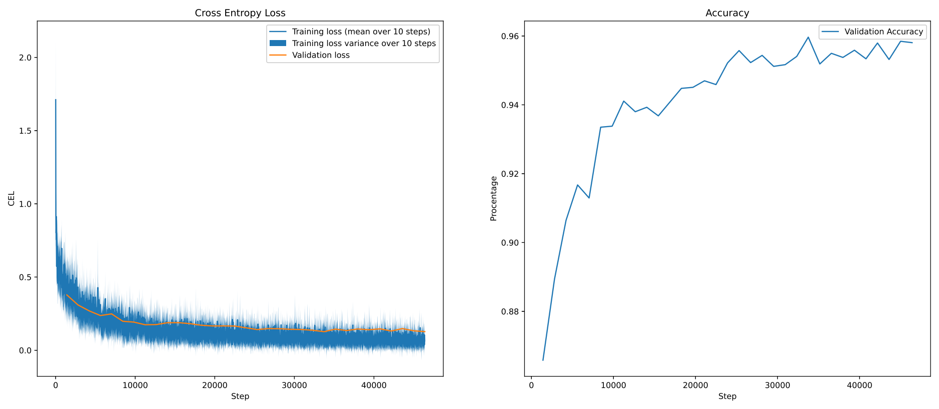

Figure 20: Zebra image activation in first convolution layer using sgd. Stoped at Epoch: 14, Batches per seconds: 22.75, Global step: 42180, Validation Loss: 0.13, Validation Accuracy: 0.958

4b

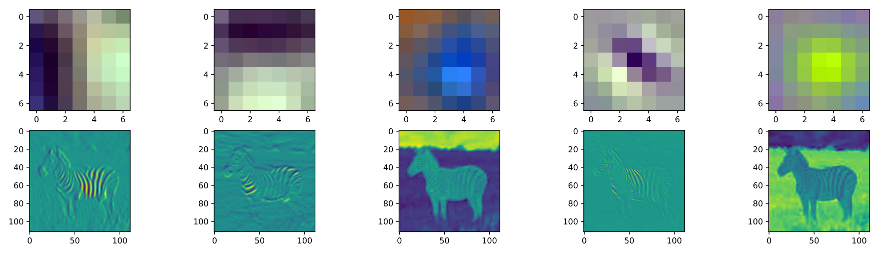

Figure 21: Zebra image activation and filters in first convolution layer.

4c

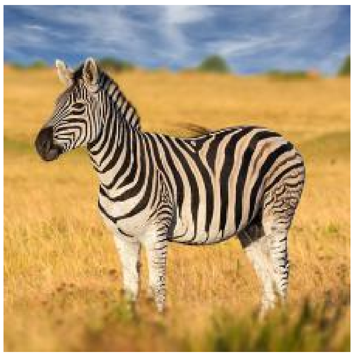

Figure 22: Zebra image for comparison.

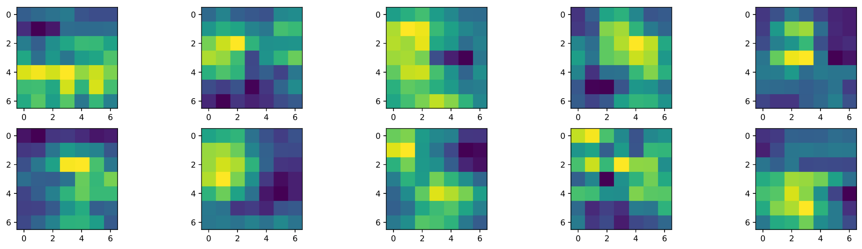

Figure 23: Zebra image activation in last convolution layer.

Comparing the image seen in Figure 22 to the the corresponding activation filters in Figure 23 we can see what areas of the image that has the most unique / interesting features by their different "hotspots". Looking at the locations of these there seem to be an emphasize on the center (covered with the zebras stripes) and the upper left corner with the head. This makes sense as it is also the first features a human would look at to correctly identify the zebra.