Rapport 3

Task 1

a)

img: zero padding

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 3 & 1 & 0 \\ 0 & 3 & 2 & 0 & 7 & 0 & 0 \\ 0 & 0 & 6 & 1 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

kernel:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & -1 & 11 & -2 & -13 \\ 10 & -4 & 8 & 2 & -18 \\ 14 & -1 & -5 & 6 & -9 \end{bmatrix}$$

* added zero padding so the result is 3x5 and we can perform "same convolution"

row column

$1.1 = 0+0+0 + 0+0+0 + 0+0+2\cdot 1 = 2$
$2.1 = 0+0+0 + 0+0+0 + 0+2\cdot 2+1\cdot 6 = 10$
$3.1 = 0+0+0 + 0+0+0 + 2\cdot 1+2\cdot 6+0 = 14$

$1.2 = 0-2-3 + 0+0+0 + 0+4+0 = -1$
$2.2 = -1-6+0 + 0+0+0 + 2+0+1 = -4$
$3.2 = -3+0+0 + 0+0+0 + 0+2+0 = -1$

$1.3 = 0+0-2+0+0+0+0+6+7 = 11$
$2.3 = 0-4-6+0+0+0+3+14+1 = 8$
$3.3 = -2-12+0+0+0+0+7+2+0 = -5$

$1.4 = 0-4+0+0+0+0+0+2+0 = -2$
$2.4 = -2+0-1+0+0+0+1+0+4 = 2$
$3.4 = 0-2+0+0+0+0+0+8+0 = 6$

$1.5 = 0-6-7+0+0+0+0+0+0 = -13$
$2.5 = -3-14-1+0+0+0+0+0+0 = -18$
$3.5 = -7-2+0+0+0+0+0+0 = -9$

1b

Max polling

1c

Padding = 1

1d)

$$[512 \times 512] \xrightarrow{conv1} [504 \times 504] \rightarrow \dots$$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \quad \Rightarrow \quad F = W_1 - W_2 = 512 - 504 + 1 = \underline{\underline{9}}$$

1e)

$$W_2 = \frac{W_1 - F}{S} + 1 \quad \Rightarrow \quad W_2 = \frac{504 - 2}{2} + 1 = 252$$

$$[504 \times 504] \xrightarrow{pol1} \underline{\underline{[252 \times 252]}}$$

1f)

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 = \frac{252 - 3}{1} + 1 = 250$$

$$[252 \times 252] \xrightarrow{conv2} \underline{\underline{[250 \times 250]}}$$

1g)

conv: $P = 2 \quad S = 1 \quad F = 5$ pool: $F = 2 \quad S = 2$

$$[32 \times 32 \times 3] \xrightarrow{conv1} [32 \times 32 \times 32] \xrightarrow{pool1} [16 \times 16 \times 32] \xrightarrow{conv2}$$
$$[16 \times 16 \times 64] \xrightarrow{pool2} [8 \times 8 \times 64] \xrightarrow{conv3+pool3} [4 \times 4 \times 128]$$

Layer 1 : 32 filters $\times (5 \times 5 + 1$ bias$) = 832$

Layer 2 : 64 filters $\times (5 \times 5 + 1$ bias$) = 1664$

Layer 3 : 128 filters $\times (5 \times 5 + 1$ bias$) = 3329$

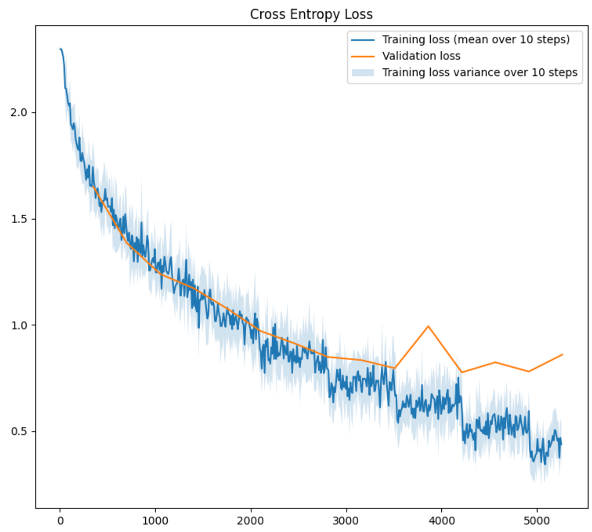
Layer 4 : $(4 \cdot 4 \cdot 128) \times 64$ weights $+ 64$ bias $= 131136$

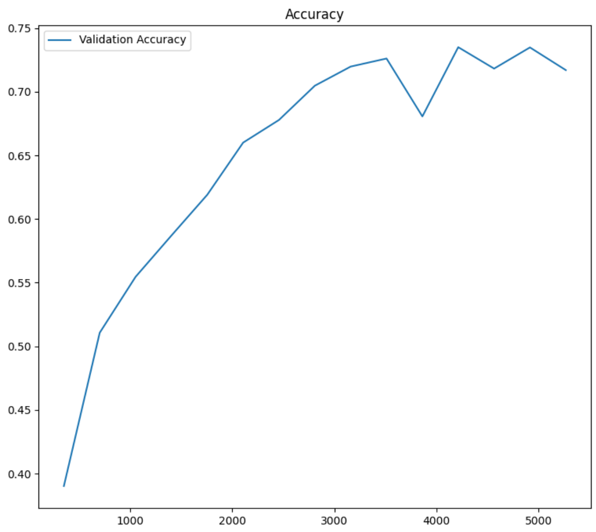
Layer 5 : $64 \cdot 10$ weights $+ 10$ bias $= 650$ +

137 611

Task 2

2a





2b

- Train loss: 0.46, Train accuracy: 0.84
- Validation loss: 0.77, Validation accuracy: 0.74
- Test loss: 0.80, Test accuracy: 0.73

Task 3

3a

My idea here was to make one network that uses more convolutional layer and few fully connected layers, in this model tweaking the architecture should be enough to achieve 75% accuracy. The other network should have more complex fully connected layer, and maybe fewer convolution layers, so I could use other methods to improve the performance.

Model 1

LAYER	LAYER TYPE	HIDDEN UNITS/FILTERS	ACTIVATION FUNCTION
1	Conv2d	128	ReLU
2	Conv2d	256	ReLU
2	MaxPool2d		
3	Conv2d	512	ReLU
4	Conv2d	512	ReLU
4	MaxPool2d		
	Flatten		
5	Linear	64	ReLU
6	Linear	64	ReLU

7	Linear	10	Softmax

- Data augmentation: RandomRotation, RandomGrayscale
- Optimizer: SGD, no momentum, no Regularization
- Batch size: 64
- Learning rate: 5e-2

Model 2

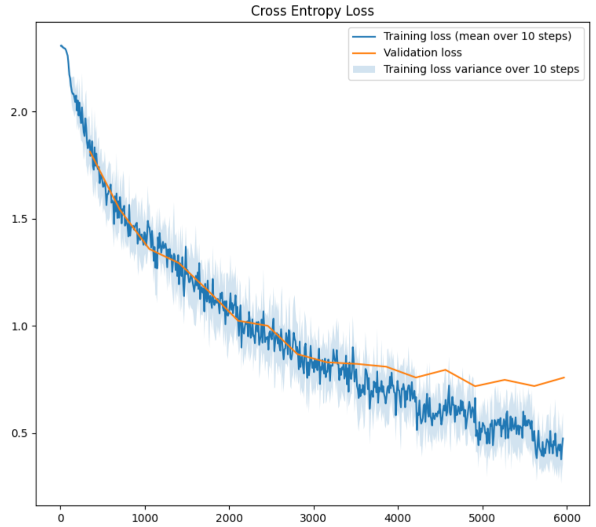
LAYER	LAYER TYPE	HIDDEN UNITS/FILTERS	ACTIVATION FUNCTION
1	Conv2d	150	ReLU
1	MaxPool2d		
1	BatchNorm2d		
2	Conv2d	256	ReLU
2	MaxPool2d		
2	BatchNorm2d		
	Flatten		
	BatchNorm1d		
3	Linear	128	ReLU
3	Dropout	P=0.1	
4	Linear	128	ReLU
4	Dropout	P=0.1	
5	Linear	32	ReLU
6	Linear	10	Softmax

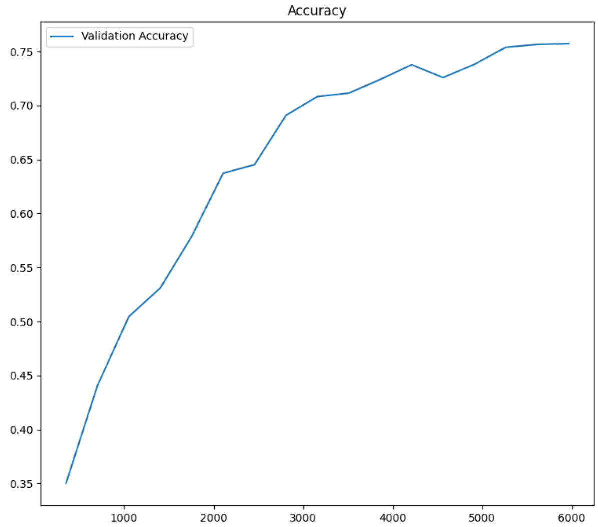
- Data augmentation: None
- Optimizer: Adagrad
- Batch size: 64
- Learning rate: 0.001 (standard adagard lr)

3b

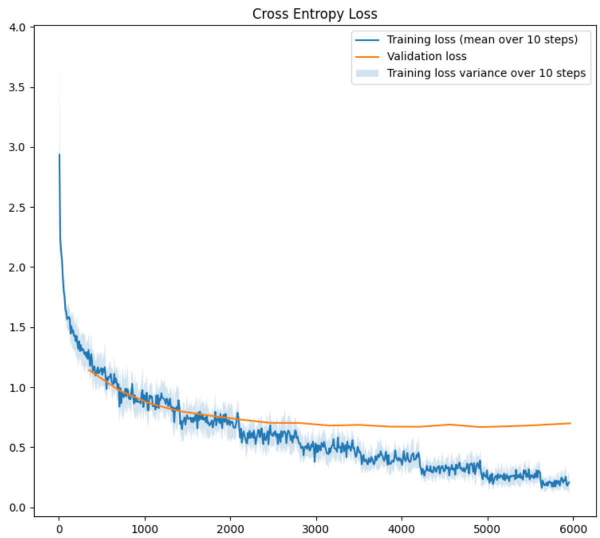
	MODEL1	MODEL2
TRAIN LOSS	0.35	0.35
TRAIN ACCURACY	0.88	0.88
VALIDATION ACCURACY	0.76	0.76
TEST ACCURACY	0.78	0.75

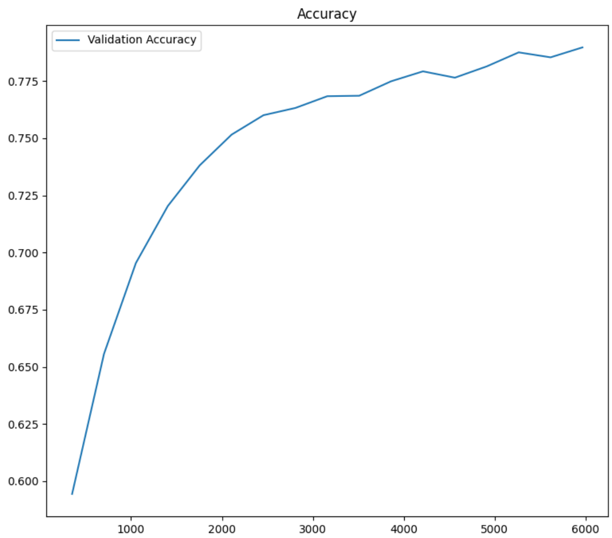
Model 1





Model 2





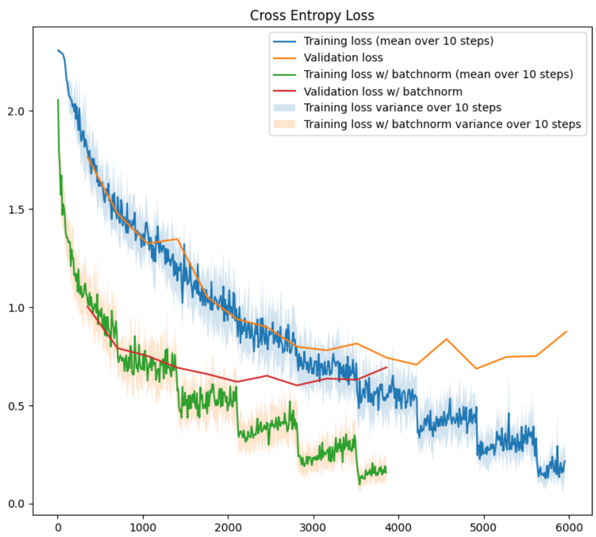
3c

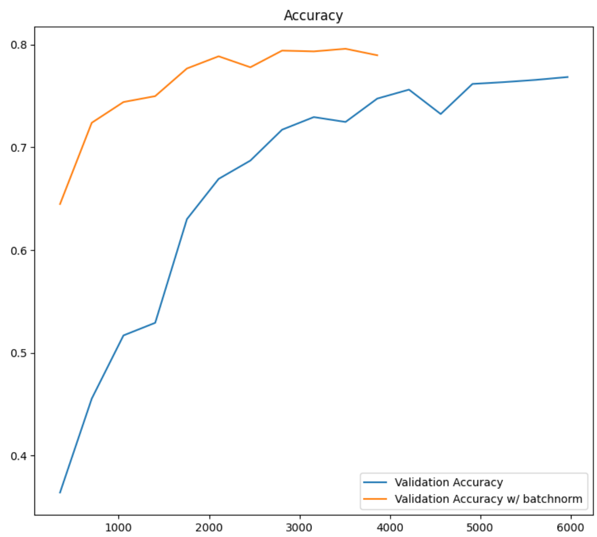
- Increasing number of convolution layers generally improved the performance, increasing complexity of the fully connected layers had little improvement. This means that feature extraction is the part that does the main work in the image recognition. However, after some point the performance didn't get any better after adding more filters and convolution layer. This might be because the model was complex enough for the problem.

- Since we can not maxpool images infinitely because the size will get too small, I decided to keep two maxPool layers. The network gave best results when the maxPool layers was spread out in the layer (for instance with two conv layers in between).
- Batch normalization improved both the end accuracy and reduces the training time needed. It's an expected effect and it had great effect on the performance.
- Adding momentum had small improvement, adding L2 weight normalization made the performance worse. Here it would be necessary to tune the hyperparameters. It was much easier and more effective to add dropout to eliminate some overfitting.
- By making the network more complex, we might need to update the learning rate. It can be quite difficult to find optimal learning rate when using SGD, the quite easy solution was to change to an adaptive learning rate method. The method that gave best results for my network was Adagrad and improved the performance specially when the Dropouts were added.
- Dropout on fully connected layer had quite a big impact on reducing the overfitting. With dropout the validation loss and training loss were almost identical. It increased however the learning time, which is to be expected after adding extra operations.
- ReLU has been much better activation function compared to tanh and sigmoid. ReLU minimizes problem with small derivatives far away from 0, which we saw was a problem in assignment 1 and 2. Other functions like ELU and LekingReLU had very similar or slightly worse performance then ReLU.
- Reducing the filter size from 5 to 3 had a little improvement in the performance. Small filter might be better at finding features in smaller images like ours.
- Adding data augmentation resulted in better test accuracy even though the final validation and training accuracy decreased. Reason for this might be that both validation data and training data got augmented (ideally only training data should be augmented), but it resulted in more general model that performs better on unseen data.
- Decreasing batch size had little effect on the accuracy, but increased the runtime. Increasing the batch size significantly had negative effect on the final accuracy.

3d

Here we can see comparison of network similar to model2 from previous task with and without batch normalization. As we can see the training is much faster and gives better results.



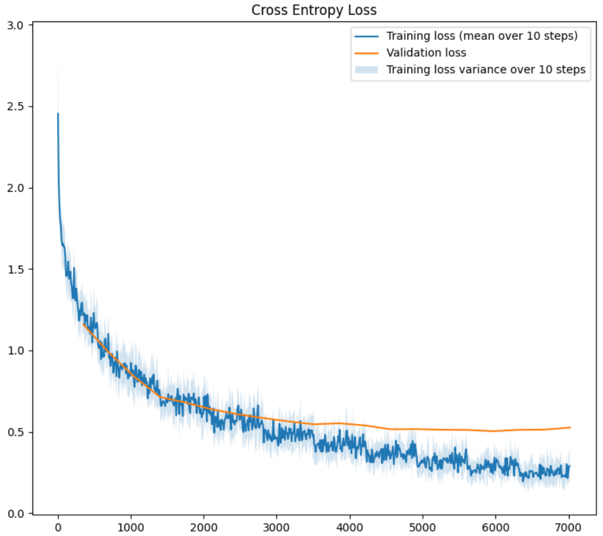


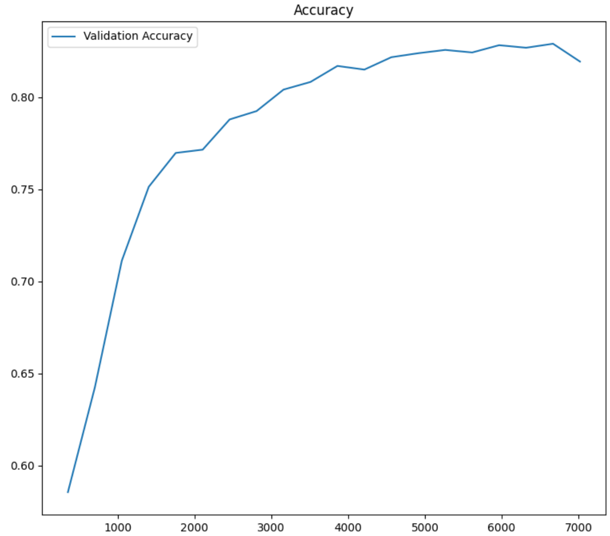
3e

LAYER	LAYER TYPE	HIDDEN UNITS/FILTERS	ACTIVATION FUNCTION
1	Conv2d	128	ReLU
1	BatchNorm2d		
2	Conv2d	128	ReLU
2	BatchNorm2d		
2	MaxPool2d		
3	Conv2d	128	ReLU
3	BatchNorm2d		
4	Conv2d	118	ReLU
4	BatchNorm2d		
4	MaxPool2d		
5	Conv2d	128	ReLU
5	BatchNorm2d		
6	Conv2d	118	ReLU
6	BatchNorm2d		
	Flatten		
	BatchNorm1d		
7	Linear	256	ReLU
7	BatchNorm1d		
7	Dropout	P=0.4	
8	Linear	10	Softmax

Final results:

- Train loss: 0.26, Train accuracy: 0.91
- Validation loss: 0.58, Validation accuracy: 0.80
- Test loss: 0.53, Test accuracy: 0.83



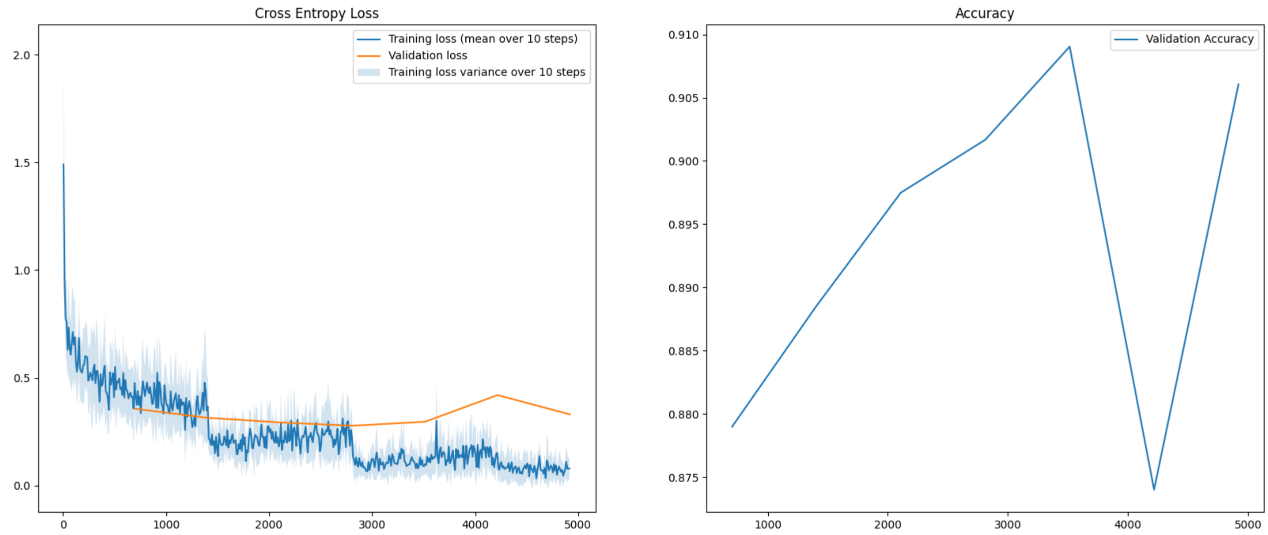


3f

We can see that the final model is still overfitting a bit, since validation loss and training loss is not same.

Task 4

4a



The accuracy plot has quite small interval on y-axis whis make it seem like big variations

Final results:

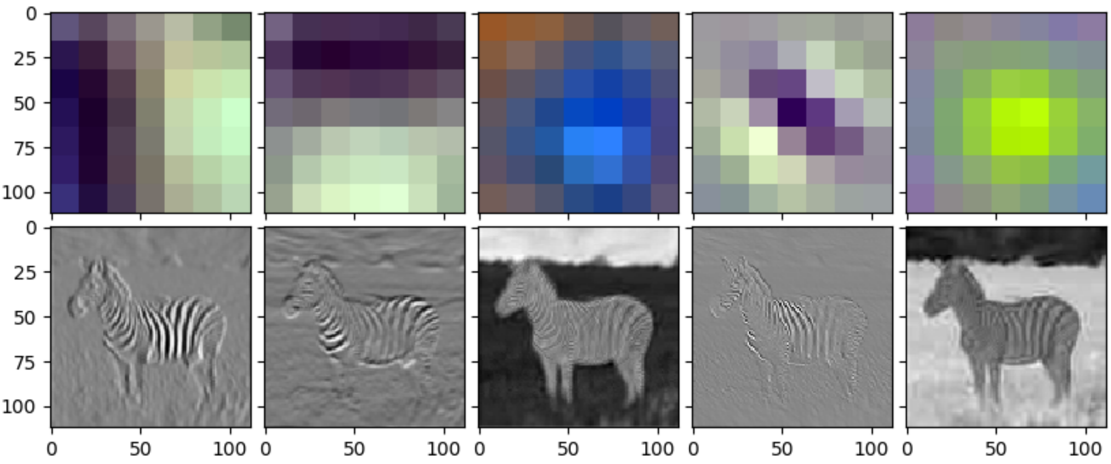
- Train loss: 0.13, Train accuracy: 0.96
- Validation loss: 0.33, Validation accuracy: 0.89
- Test loss: 0.37, Test accuracy: 0.88

Hyperparameters:

- batch_size = 32
- learning_rate = 5e-4
- optimizer: Adam
- data augmentation: resize and normalization

4b

We can clearly see that different layers extract different features of the image. For instance, filter 3 and 5 in the figure below extracts pretty well the outline shape of the zebra, these two filters are also almost negatives of each other. Filter 1 extracts vertical lines and filter 2 horizontal lines, which is easily visible on the zebra.



4c

The last layers contain more low-level features, like part of a line. It's no longer possible to recognize the image features by eye. Found feature have much higher contrast, clearer what feature has been extracted. It is to be expected when an image has been passed through a lot of convolution layers.