NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4265 - Computer Vision and Deep Learning

Assignment 3 Report

Authors:
Sander Aakerholt, Ludvig Brannsether Ellingsen

March, 2022

1 Introduction

This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet. If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways:

- Print the webpage (ctrl+P or cmd+P)
- Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File → Download as → PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc`.

2 Task 1

task 1a)

We use zero-padding to fill the edges of the image.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 3 & 1 & 0 \\ 0 & 3 & 2 & 0 & 7 & 0 & 0 \\ 0 & 0 & 6 & 1 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \circledast \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -1 & 11 & -2 & -13 \\ 10 & -4 & 8 & 2 & -18 \\ 14 & -1 & -4 & 6 & -9 \end{bmatrix} \tag{1}$$

task 1b)

The layer that reduces the sensitivity to translational variations is (iii) the Max Pooling layer.

task 1c)

With a Kernel size of 5x5 and a stride of 1, we need to add a padding of two on each side to not reduce image size through convolution.

$$P = (F - 1)/2 \tag{2}$$

Where P is the required layers of padding, F is the dimentionlality of the kernel, and for 2 to be valid, the stride must be 1.

Hence we have

$$P = (5 - 1)/2 = 2 \tag{3}$$

task 1d)

The image size has been reduced by 8 in each direction. In other words, you would need a padding of four on each side to prevent reduction in image size. Using eq. (2), we then find that the filter is of size 7x7.

task 1e)

The pooling the size in every direction. After the pooling is computed, the new image size 257x257.

task 1f)

As we use no padding, the size is reduced by one on every edge. The new size is therefore 255x255.

task 1g)

Layer	Number of parameters	Sum
1	5x5x32 + 32 + 2x2	836
2	5x5x64 + 64 + 2x2	1668
3	5x5x128 + 128 + 2x2	3332
Flatten	None	None
4	[4x4x128]x64 + 64	131136
5	64x10 + 10	650
Total		137622

Table 1: Parameter table

3 Task 2

3.1 Task 2a

See fig. 1.

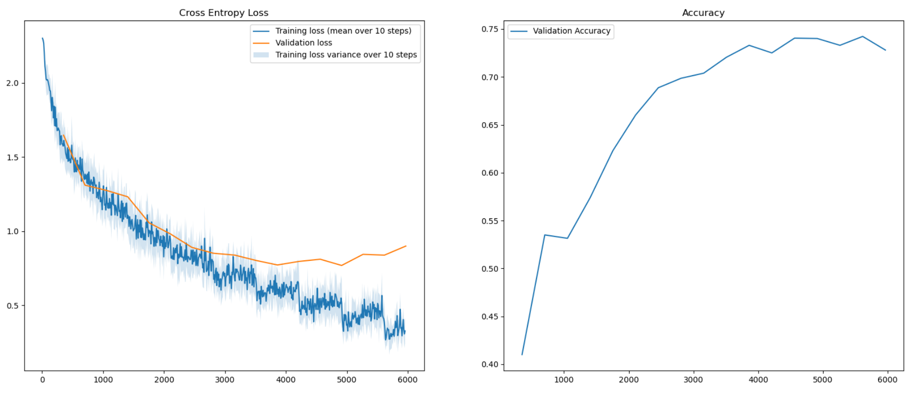

Figure 1: Task 2a: Train and validation loss

3.2 Task 2b)

Train accuracy: 0.8288584637268848
Validation accuracy: 0.7314082278481012
Test accuracy: 0.7280055732484076

4 Task 3

4.1 Task 3a)

See table 2:
Train accuracy: 0.8862242176386913
Validation accuracy: 0.778876582278481
Test accuracy: 0.7705015923566879

Train loss: 0.3479120516632902
Validation loss: 0.6408350607262382
Test loss: 0.6761114975069739

Layer	Layer type	Number of Hidden Units/Number of Filters	Activation function
1	Conv2d	32	LeakyReLU(slope=-0.1)
2	Conv2d	32	LeakyReLU(slope=-0.1)
3	MaxPool2d	-	-
4	BatchNorm2d	32	-
5	Conv2d	64	LeakyReLU(slope=-0.1)
6	Conv2d	64	LeakyReLU(slope=-0.1)
7	MaxPool2d	-	-
8	BatchNorm2d	32	-
9	Conv2d	128	LeakyReLU(slope=-0.1)
10	Conv2d	128	LeakyReLU(slope=-0.1)
11	MaxPool2d	-	-
12	BatchNorm2d	128	-
	Flatten	-	-
13	BatchNorm1d	2048	-
14	Fully-connected	1024	LeakyReLU(slope=-0.1)
15	Fully-connected	512	LeakyReLU(slope=-0.1)
16	Fully-connected	10	-

Table 2: First model: Learning rate = 1e-2. For all Conv2d layers we used K=3, S=1, P=1, for max pooling 2d layers we used K=2, S=2.

See table 3:
Train accuracy: 0.910472972972973 Validation accuracy: 0.783623417721519 Test accuracy: 0.7834394904458599

Train loss: 0.27312234366262284 Validation loss: 0.641265171426761 Test loss: 0.6547565146995957

4.2 Task 3b)

See table 4 and fig. 2:

4.3 Task 3c)

Experimenting with the learning rate and using batch normalization after max pooling resulted in the biggest increases in performance. Adding more layers and more and smaller filters in the feature extractor and classification sections also resulted in a performance boost, as well as that switching ReLU with LeakyReLU helped a bit. The batch normalization probably helped because it might help the model in generalizing as it normalizes the data from the layer it is placed after. Increasing the learning rate a bit makes the model take bigger steps when learning, if this is increased too much the model will preform badly as it will take too large steps, and miss important features

Layer	Layer type	Number of Hidden Units/Number of Filters	Activation function
1	Conv2d (K=3,S=1,P=1)	32	LeakyReLU(slope=-0.1)
2	Conv2d (K=5,S=1,P=2)	32	LeakyReLU(slope=-0.1)
3	MaxPool2d	-	-
4	Conv2d (K=3,S=1,P=1)	64	LeakyReLU(slope=-0.1)
5	Conv2d (K=5,S=1,P=2))	64	LeakyReLU(slope=-0.1)
6	MaxPool2d	-	-
7	Conv2d (K=3,S=1,P=1)	128	LeakyReLU(slope=-0.1)
8	Conv2d (K=5,S=1,P=2)	128	LeakyReLU(slope=-0.1)
9	MaxPool2d	-	-
	Flatten	-	-
4	BatchNorm1d	2048	-
4	Fully-connected	1024	LeakyReLU(slope=-0.1)
4	Fully-connected	512	LeakyReLU(slope=-0.1)
5	Fully-connected	10	-

Table 3: Second model: Learning rate = 1.5e-2. For max pooling 2d layers we used K=2, S=2.

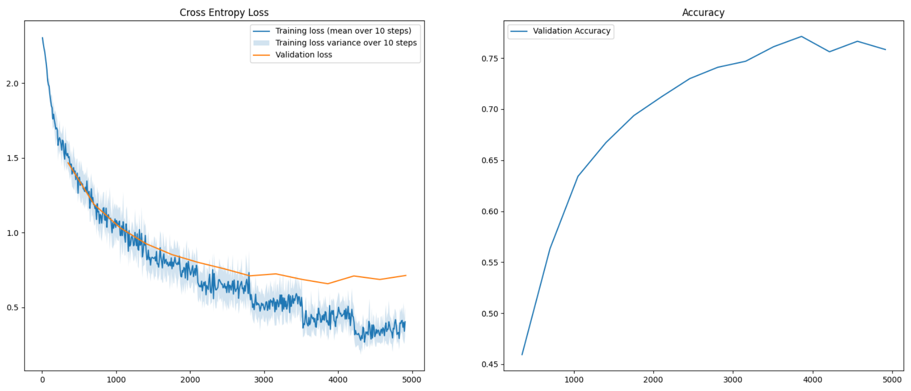

Figure 2: Task 3b: Train and validation loss

Model	Train loss	Train accuracy	Validation accuracy	Test accuracy
1	0.34791	0.8862	0.7788	0.7705
2	0.3336	0.9104	0.7836	0.7834

Table 4: Loss and accuracy

(under-fitting?). Deepening the network can help with recognizing more distinct features in the data, but making the network too deep can eventually result in over-fitting. Can be that the LeakyReLU helps a bit with generalization.

Things that we tested that gave worse results was increasing the filter and padding size, adding drop-out layers and placing the batch normalization layers in the wrong place. The dropout probably didn't do much good as the network was not doing any over-fitting of the data, so the model in itself was able to generalize well. Increasing the filter size gave worse results, as the details preserved by the filters would probably blur out details, which made it harder to classify the image. Not sure what was wrong with the batch norm, but it eventually made sense for us to put it after the max pooling layers.

4.4 Task 3d)

In the figure fig. 3 we see that with or without batch normalization in the right places the model was rubbish, but with batch norm it was preformed much better.

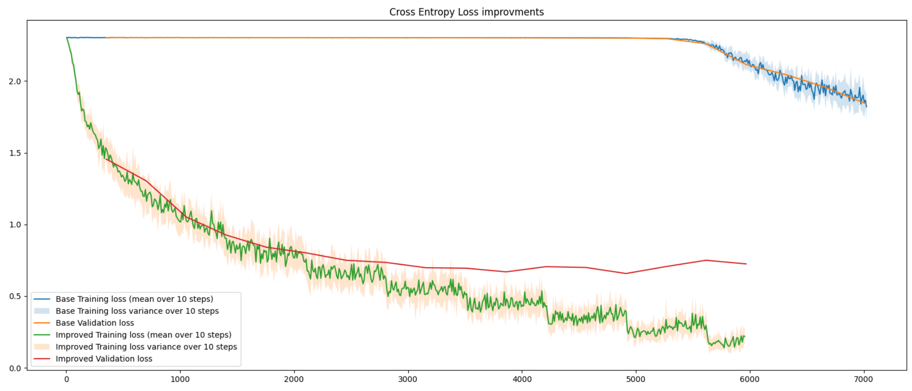

Figure 3: Task 3d: Batch normalization helped a lot.

4.5 Task 3e)

We decided to continue with model 1, as it preformed better at lower learning rates, hence it potentially had even more to gain from increasing the learning rate, and other changes.

By increasing the learning rate to 12e-2 both the previous models were hitting accuracy's around 80%, but not consistently. So by deepening the layers of the first model (see table table 5) we got the following results.

Train accuracy: 0.9383223684210527 – Train loss: 0.18932821468091113
Validation accuracy: 0.82179588607 – Validation loss: 0.5554459387365775
Test accuracy: 0.81906847133757972 – Test loss: 0.574605671843146

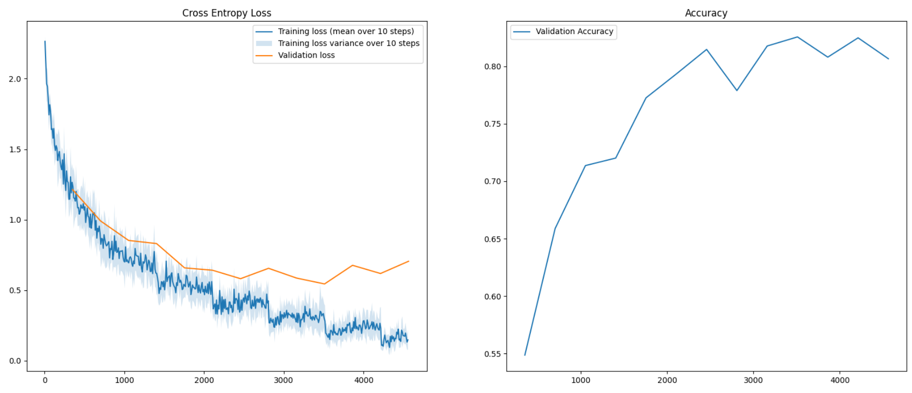

Figure 4: Task 3e: Figure of results from improved best model.

Layer	Layer type	Number of Hidden Units/Number of Filters	Activation function
1	Conv2d	32	LeakyReLU(slope=-0.1)
2	Conv2d	32	LeakyReLU(slope=-0.1)
3	MaxPool2d	-	-
4	BatchNorm2d	32	-
5	Conv2d	64	LeakyReLU(slope=-0.1)
6	Conv2d	64	LeakyReLU(slope=-0.1)
7	MaxPool2d	-	-
8	BatchNorm2d	32	-
9	Conv2d	128	LeakyReLU(slope=-0.1)
10	Conv2d	128	LeakyReLU(slope=-0.1)
11	MaxPool2d	-	-
12	BatchNorm2d	128	-
13	Conv2d	256	LeakyReLU(slope=-0.1)
14	Conv2d	256	LeakyReLU(slope=-0.1)
15	MaxPool2d	-	-
16	BatchNorm2d	256	-
17	Conv2d	512	LeakyReLU(slope=-0.1)
18	Conv2d	512	LeakyReLU(slope=-0.1)
19	MaxPool2d	-	-
20	BatchNorm2d	512	-
	Flatten	-	-
21	BatchNorm1d	2048	-
22	Fully-connected	1024	LeakyReLU(slope=-0.1)
23	Fully-connected	512	LeakyReLU(slope=-0.1)
24	Fully-connected	256	LeakyReLU(slope=-0.1)
25	Fully-connected	10	-

Table 5: First model: Learning rate = 1e-2. For all Conv2d layers we used K=3, S=1, P=1, for max pooling 2d layers we used K=2, S=2.

4.6 Task 3f)

In figure fig. 4 we see that the validation loss curve is starting to rise, which can indicate that the model did not generalize that well (indicating over-fitting), but early stopping kicked in before it got to far away.

5 Task 4

5.1 Task 4a)

FILL IN ANSWER and plot

5.2 Task 4b)

Visualize filters

5.3 Task 4c)

FILL IN ANSWER.