Assignment 1 Report group 42

Task 1

task 1a)

Perform convolution with the flipped kernel:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Output position (row, column)	Calculation	Result
(1,1)	0-2+0-5	-7
(1,2)	4+0-4+4+0+0	4
(1,3)	2+0-6+5+0-7	-6
(1,4)	4+0-2+0+0+0	2
(1,5)	6+0+7+0	13
(2,1)	0-1+0-10+0-9	-20
(2,2)	2+0-2+8+0-0+3+0-1	10
(2,3)	1+0-3+10+0-14+9+0-1	2
(2,4)	2+0-1+0+0-0+1+0-4	-2
(2,5)	3+0+14+0+1+0	18
(3,1)	0-5+0-18	-23
(3,2)	4+0+0+6+0-2	8
(3,3)	5+0-7+18+0-2	14
(3,4)	0+0-0+2+0-8	-6
(3,5)	7+0+2+0	9

Which yields the following output:

$$\begin{bmatrix} -7 & 4 & -6 & 2 & 13 \\ -20 & 10 & 2 & -2 & 18 \\ -23 & 8 & 14 & -6 & 9 \end{bmatrix}$$

task 1b)

Max Pooling reduces the sensitivity to small translational variations in the input. Since the Max Pooling outputs are only sensitive to the largest value of the pooling area, a large node can shift around within the pooling area without changing the value of the Max Pooling output.

task 1c)

Use the fact that $H_2 = W_2$ and $H_2 = H_1 = H$:

$$\begin{aligned}
H_2 &= \frac{H_1 - F + 2P}{S} + 1 \\
H_2S &= H_1 - F + 2P + S \\
H_2 &= H_1 = H \\
2P &= H(S-1) - S + F \\
P &= \frac{H(S-1) - S + F}{2} \\
P &= \frac{H(1-1) - 1 + 7}{2} \\
P &= 3
\end{aligned}$$

task 1d)

$$\begin{aligned}
H_2 &= \frac{H_1 - F + 2P}{S} + 1 \\
H_2S &= H_1 - F + 2P + S \\
F &= H_1 + S(1 - H_2) + 2P = 512 + 1 - 508 = 5
\end{aligned}$$

The filter is supposed to be square, we then get $F = (5 \times 5)$.

task 1e)

$$\begin{aligned}
H_2 &= \frac{H_1 - F + 2P}{S} + 1 \\
H_2 &= \frac{508 - 2 + 2 * 0}{2} + 1 \\
H_2 &= 254
\end{aligned}$$

Square input, square output. We get (254×254).

task 1f)

$$\begin{aligned}
H_2 &= \frac{H_1 - F + 2P}{S} + 1 \\
H_2 &= \frac{254 - 3 + 2 * 0}{1} + 1 \\
H_2 &= 252
\end{aligned}$$

(252×252).

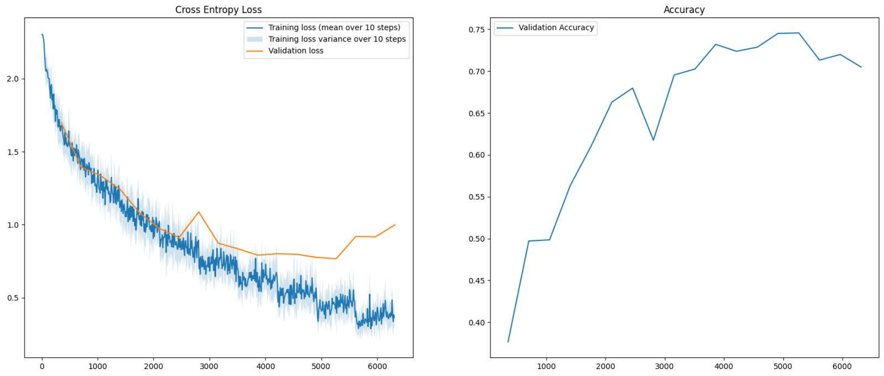
task 1g)

Filter parameters = $F_H \times F_W \times C_1 \times C_2$. Bias = C_2. Total parameters = Filter parameters + Bias.

Layer	Filter parameters	Bias	Total parameters
1	5x5x3x32 = 2400	32	2432
2	5x5x32x64 = 51200	64	51264
3	5x5x64x128 = 204800	128	204928
Flatten	4x4x128 = 2048		
4	2048x64 = 131072	64	131136
5	64*10	10	650
Sum			390410

Task 2

Task 2a)



Task 2b)

Epoch: 8, Batches per seconds: 32.67, Global step: 6318, Validation Loss: 1.00, Validation Accuracy: 0.705

Final accuracy/loss	Model
Train accuracy	0.8789
Validation accuracy	0.7164
Test accuracy	0.7272
Train loss	0.3460
Validation loss	0.9438
Test loss	0.8232

Task 3

Task 3a)

Chose to keep model 1 as in previous task given by table 1 in the assignment text. The architecture of model 2 is given by the following table. The filter size is changed to 3x3 with a padding of 1 and stride of 1. Data augmentation is applied to the training set by randomly flipping images horizontally.

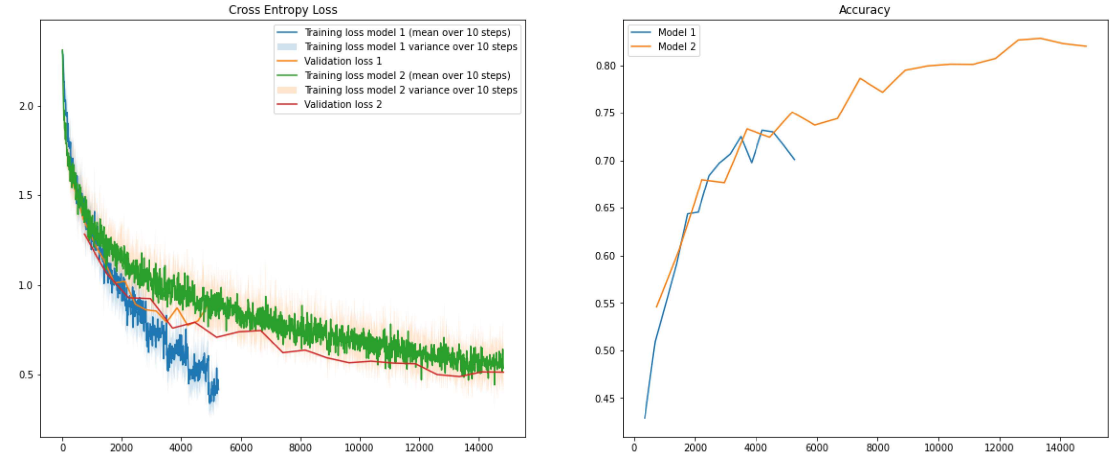
Layer	Layer Type	Number of Hidden Units / Number of Filters	Activation Function
1	Conv2D	32	ReLU
1	BatchNorm2d	-	-
1	MaxPool2D	-	-
2	Conv2D	64	ReLU
2	BatchNorm2d	-	-
2	MaxPool2D	-	-
3	Conv2D	128	ReLU
3	BatchNorm2d	-	-
3	MaxPool2D	-	-
	Flatten		
4	Fully-Connected	64	ReLU
	Dropout(p=0.5)	-	-
5	Fully-Connected	10	Softmax

Task 3b)

Training the second model yields the following final accuracy and loss:

Final accuracy/loss	Model 1	Model 2
Train accuracy	0.8330	0.8172
Validation accuracy	0.6996	0.7937
Test accuracy	0.7343	0.7952
Train loss	0.4785	0.5290
Validation loss	0.9483	0.5925
Test loss	0.8003	0.6104

Notice that the model trains for much longer, but generalizes better with improved test accuracy.

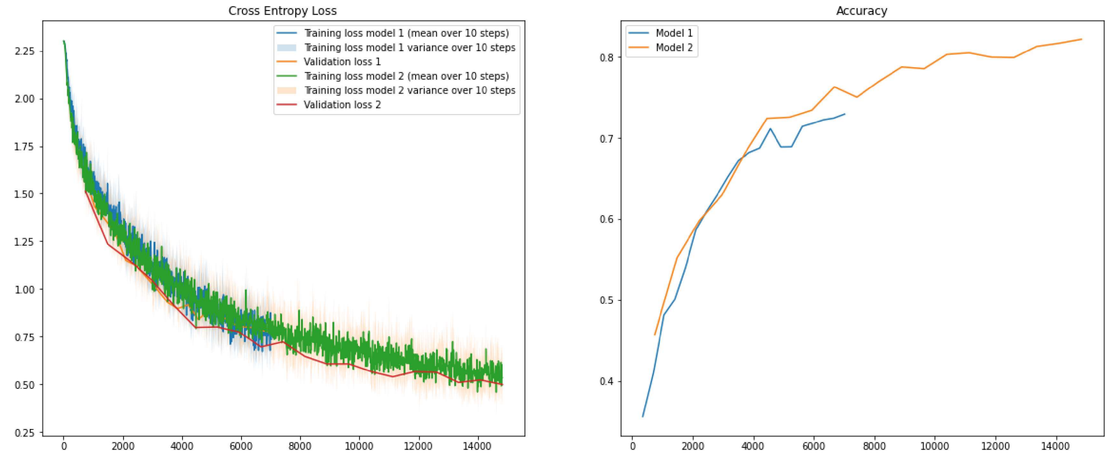


Task 3c)

- Changing filter size from 5 to 3: network trained a little faster, but no real improvement
- Adding data augmentation by flipping images horizontally gave a significant boost in accuracy. I think this is caused by the training data set being much larger.
- Adding dropout in the last layer yielded no results as far as I can tell.
- Adding batch normalization after every convolutional layer made the network train faster with high accuracy.
- Changing optimizer to Adam with a learning rate of 5e-4 gave worse results for some reason.

Task 3d)

I saw the best results by applying data augmentation. Below is a figure comparing the training of model 1 without data augmentation and model 2 with data augmentation. Model 2 takes longer to train, but in return the accuracy is far higher.



Task 3e)

As shown in the previous figure, the model with augmented training data reaches 80% validation accuracy within 10 epochs.

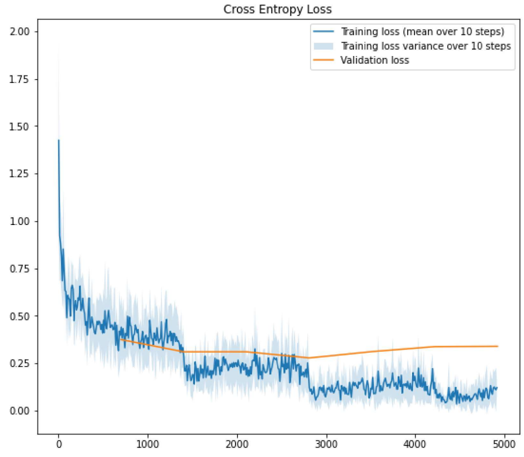
Epoch	Global step	Validation accuracy
5	8162	0.770
5	8904	0.788
6	9646	0.786
6	10388	0.803
7	11130	0.805
7	11872	0.800
8	12614	0.799
8	13356	0.813
9	14098	0.817
9	14840	0.822

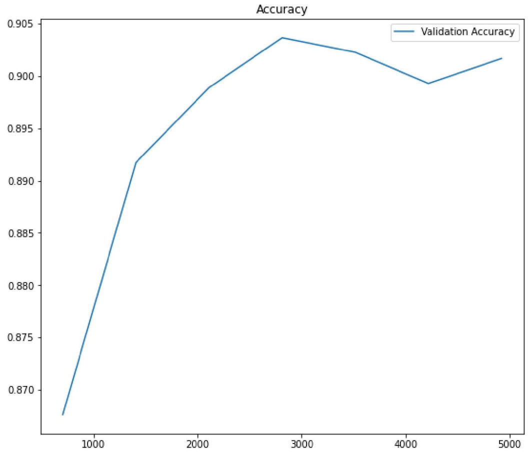
Task 3f)

Task 4

Task 4a)

Final test accuracy: 0.8970.


Cross Entropy Loss
Training loss (mean over 10 steps)
Training loss variance over 10 steps
Validation loss


Accuracy
Validation Accuracy