TDT4265 Assignment 3

Fredrik Veidahl Aagaard

February 2019

1 Convolutional Neural Networks

a)

The CNN was implemented using a filter of size 5x5 and a padding of 2. The max pool operation uses a kernel size of 2x2 and a stride of 2. The hyper parameters for the training is given by Table 1, and the architecture of the CNN is given by Table 2.

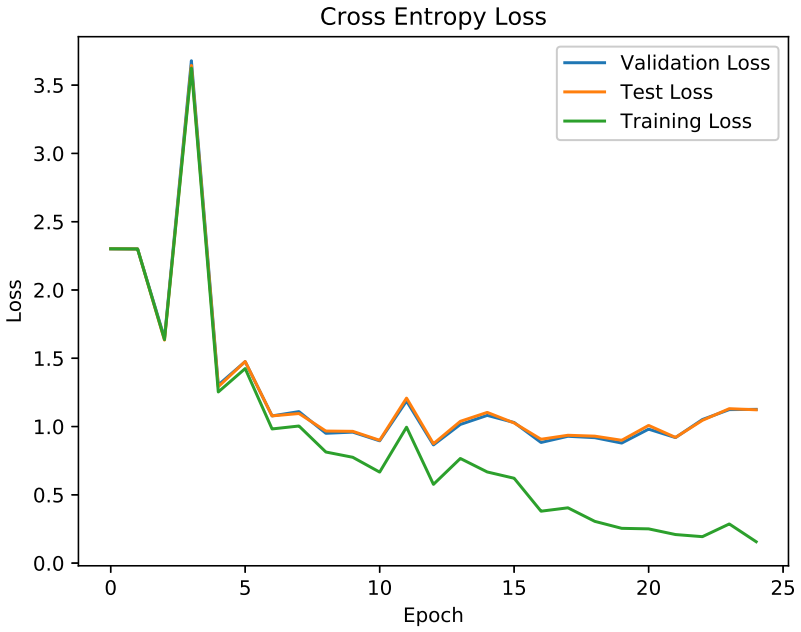

Figure 1: Cross Entropy Loss for CNN

Epochs	Learning Rate	Batch Size
100	0.05	64

Table 1: Hyperparameters

Layer Type	Number of filters	Activation Function
Conv2D	32	ReLU
MaxPool2D	-	-
Conv2D	64	ReLU
MaxPool2D	-	-
Conv2D	128	ReLU
MaxPool2D	-	-
Flatten	-	-
Dense	64	ReLU
Dense	10	Softmax

Table 2: Architecture of CNN

b)

The final training, validation, and test accuracy for the CNN implemented in a) is 94.6%, 73.5%, 73.2%, respectively.

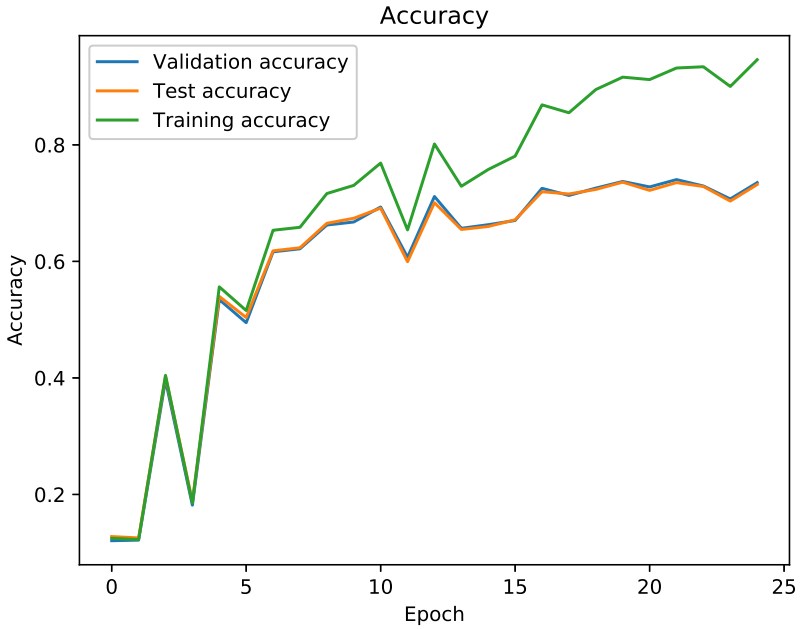

Figure 2: Accuracy for CNN

c)

We can use the relationship

$$F_H \times F_W \times C_{in} \times C_{out} + C_{out} = N \tag{1}$$

where F_H, F_W, C_{in}, C_{out}, F_H, N is the filter height, filter width, number of input channels, number of output channels and number of parameters respectively. We use this equation to calculate the number of parameters in each of the layers given by Table 2. Thus the number of parameters in the CNN is given by

Conv2D 1:	$5 \times 5 \times 3 \times 32 + 32$	$=$	2432
Conv2D 2:	$5 \times 5 \times 32 \times 64 + 64$	$=$	51264
Conv2D 3:	$5 \times 5 \times 64 \times 128 + 128$	$=$	204928
Dense 1:	$5 \times 5 \times 128 \times 64 + 64$	$=$	204864
Dense 2:	$5 \times 5 \times 64 \times 10 + 10$	$=$	16010
Total:		$=$	479498

Thus there are a total of 479498 parameters in the network given by Table 2.

2 Deep Convolutional Network for Image Classification

a)

Network 1

When creating the first network I tried to change as little as possible to get it above 75% accuracy on the test set within 10 epochs. I added batch normalization to each layer as I assumed this would help a little with the effiecieny, since the different features get normalized. I also added dropout to the last layer of the 'feature extractor'.

Doing only this I was able to get the network to about 76% accuracy on the test set after 8 epochs.

The hyperparameters for training Network 1 is given in Table 3 and the architecture of the network is given in Table 4.

Epochs	Learning Rate	Batch Size
10	0.05	64

Table 3: Hyperparameters for Network 1

Layer Type	Number of filters	Activation Function
Conv2D	32	ReLU
BatchNorm2D	-	-
MaxPool2D	-	-
Conv2D	64	ReLU
BatchNorm2D	-	-
MaxPool2D	-	-
Conv2D	128	ReLU
BatchNorm2D	-	-
MaxPool2D	-	-
Dropout	-	-
Flatten	-	-
Dense	64	ReLU
Dense	10	Softmax

Table 4: Architecture of Network 1

Network 2

When creating the second network I tried to implement more of these 'tricks'. The architecture of the CNN was changed to have more layers. Batch normalization and dropout was used in this network aswell, but here I used dropout after each group of layers. The filter size was reduced to 3x3, from 5x5, since this is the most popular shape, and the padding was reduced to 1 instead of 2. Xavier Initialization was used to initialize the convolutional weights. Adam was used to optimize the parameters instead of Stochastic Gradient Descent(SGD). PReLU was used as activation function instead of ReLU in the classifier layer.

By implementing these changes I was able to get the network above 75% accuracy on the test set after 5 epochs, and it was able to reach above 80% within 10 epochs.

The hyperparameters for training Network 2 is given in Table 5 and the architecture of the network is given in Table 6.

Epochs	Learning Rate	Batch Size
10	0.005	32

Table 5: Hyperparameters for Network 2

Layer Type	Number of filters	Activation Function
Conv2D	32	ReLU
BatchNorm2D	-	-
Conv2D	32	ReLU
BatchNorm2D	-	-
MaxPool2D	-	-
Dropout	-	-
Conv2D	64	ReLU
BatchNorm2D	-	-
Conv2D	64	ReLU
BatchNorm2D	-	-
MaxPool2D	-	-
Dropout	-	-
Conv2D	128	ReLU
BatchNorm2D	-	-
Conv2D	128	ReLU
BatchNorm2D	-	-
MaxPool2D	-	-
Dropout	-	-
Flatten	-	-
Dense	64	PReLU
Dense	10	Softmax

Table 6: Architecture of Network 2

b)

	Network 1	Network 2
Final Training Loss	0.317	0.360
Final Training Accuracy	89.2%	87.4%
Final Test Loss	0.678	0.519
Final Test Accuracy	77.6%	82.3%
Final Validation Loss	0.641	0.513
Final Validation Accuracy	78.2%	84.3%

Table 7: Final loss and accuracy for the different sets

It is interesting to observe that the final values for the training set is worse in Network 2 compared to Network 1, but for the two other sets it is a lot better.

c)

The best model is Network 2 and will be the one we continue to study

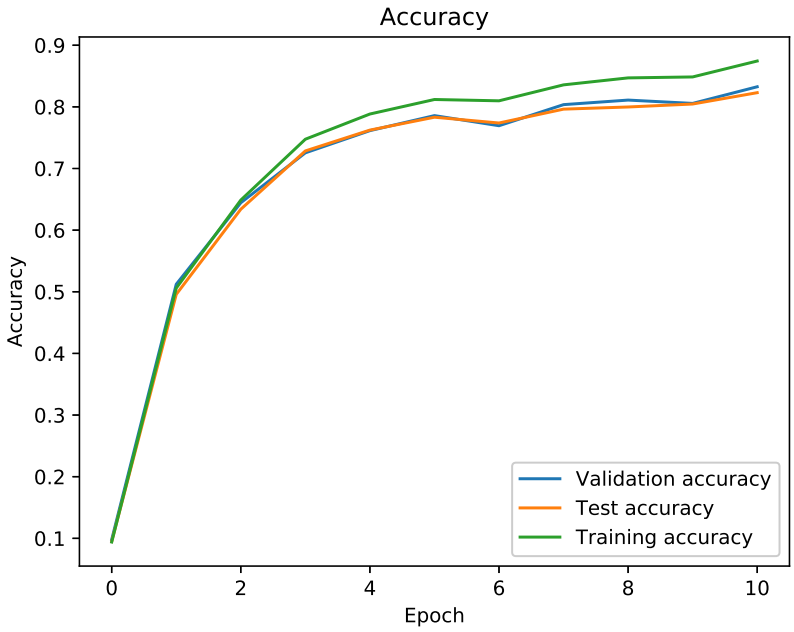

Figure 3: Accuracy for Network 2

d)

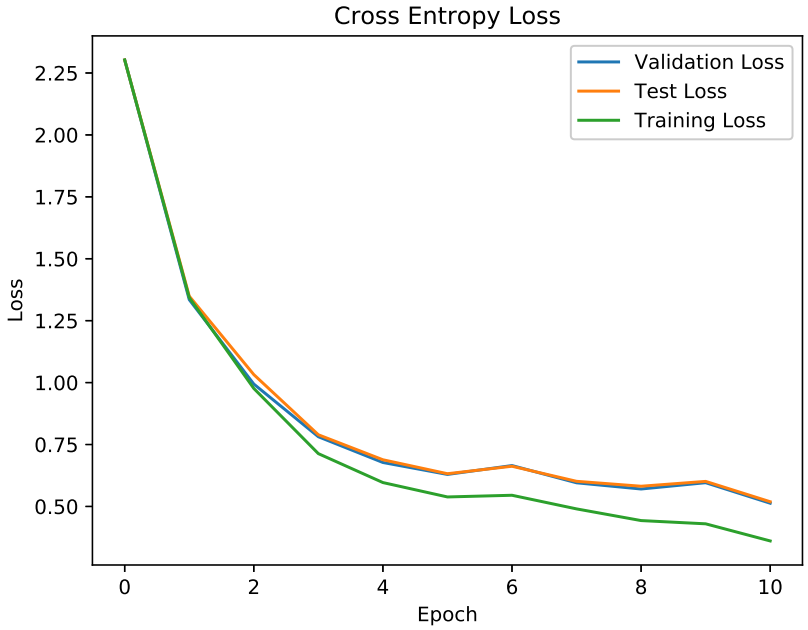

Figure 4: Cross Entropy Loss for Network 2

e)

When implementing the different tricks I saw the most improvement when initializing the weights using Xavier Initialization and when using Adam optimizer instead of SGD. It is important to properly initialize the weights before training starts. Xavier makes sure that the weights are not too large or small such that the signal doesn't shrink or blow up to become useless as it passes through the layers. I think the Adam optimizer gives better results compared to SGD because the algorithm uses some other tricks, like momentum, and in this case it works better than SGD (which may not be true for all cases).

I tried some different structures that didn't work as well as the one I ended up with in Network 2. An example was to have fewer layers in the two last feature extractor layers. I think the network would then be less efficient at discovering the features in these layers.

3 Transfer Learning with ResNet18

a)

I implemented ResNet18 using the recommended hyperparameters given in Table 8 and Adam optimizer.

Epochs	Learning Rate	Batch Size
10	0.0005	32

Table 8: Hyperparameters for ResNet18

b)

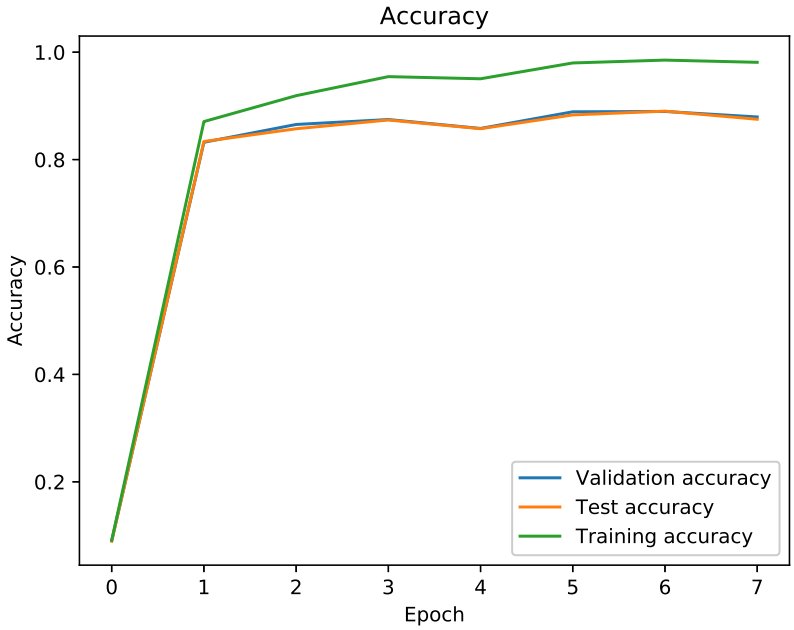

Figure 5: Accuracy for ResNet18

c)

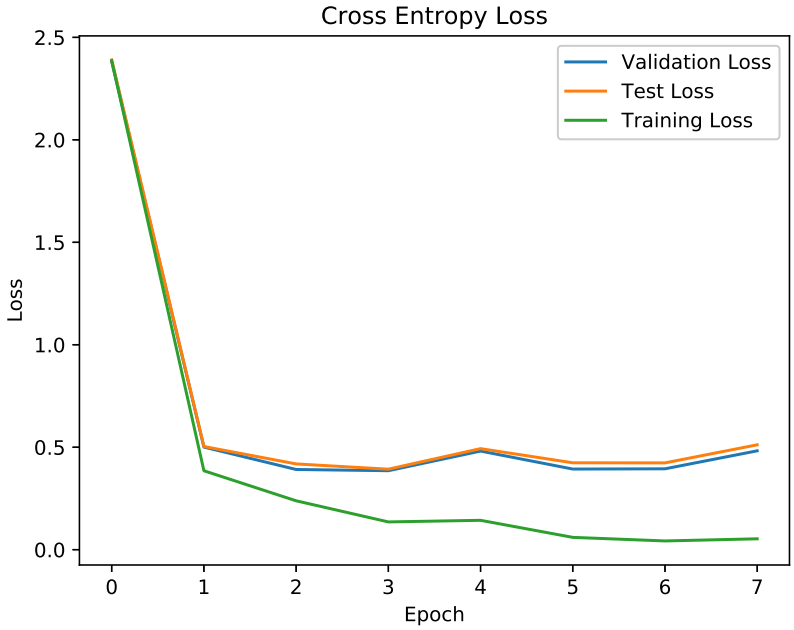

Figure 6: Cross Entropy Loss for ResNet18

d)

We can see from Figure 7 that ResNet18 obiously trains much faster than my best network (Network 2). We see that it converges after about 4 epochs, and starts to show tendencies of overfitting which is why it early stops after 7 epochs. The validation loss of Network 2 starts to approach that of ResNet18, but the training loss is much worse.

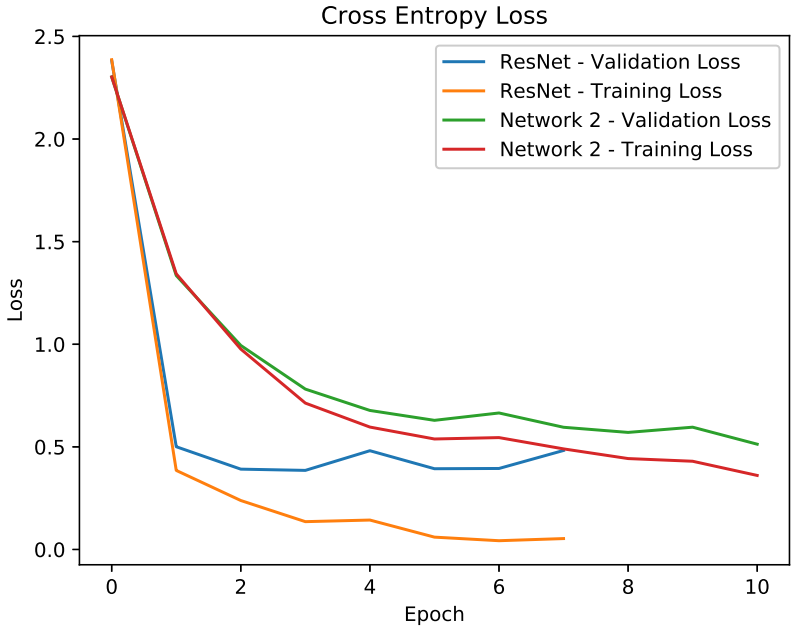

Figure 7: Cross Entropy Loss for ResNet18 vs. my best network (Network 2)

e)

Figure 10 shows the filters in the first convolutional layer after Figure 8 has been passed through the first layer. We see that the different filters extract different features. Some look to e.g. find edges on the bottom of the frog, find edges on top of the frog, find a slight outline of the whole frog, find the 2 large eyes on top of its head. And it seems that most of theses filters have a specific "job" to do.

Figure 8: Original image of a frog

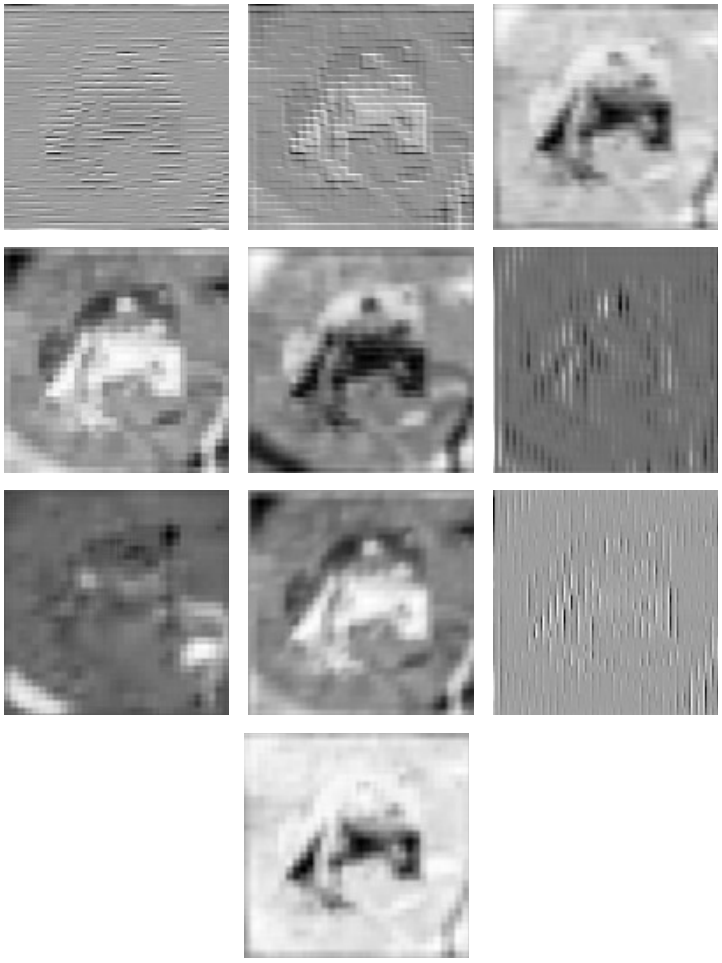

Figure 9: 10 filters in the first convolutional layer

f)

Figure 10 shows the filters in the last convolutional layer after Figure 8 has been passed through the network. It is a little harder to see what these filters are 'searching' for, and get an intuitive feel for it, but the idea is still there just on a deeper level.

Figure 10: 10 filters in the last convolutional layer

g)

Figure 11 represent the weights in the first convolutional layer. These weights shows which parts of the image is looked at when trying to classify objects in the image. When there is e.g. a large red area, that area is more weighted towards something red being there.

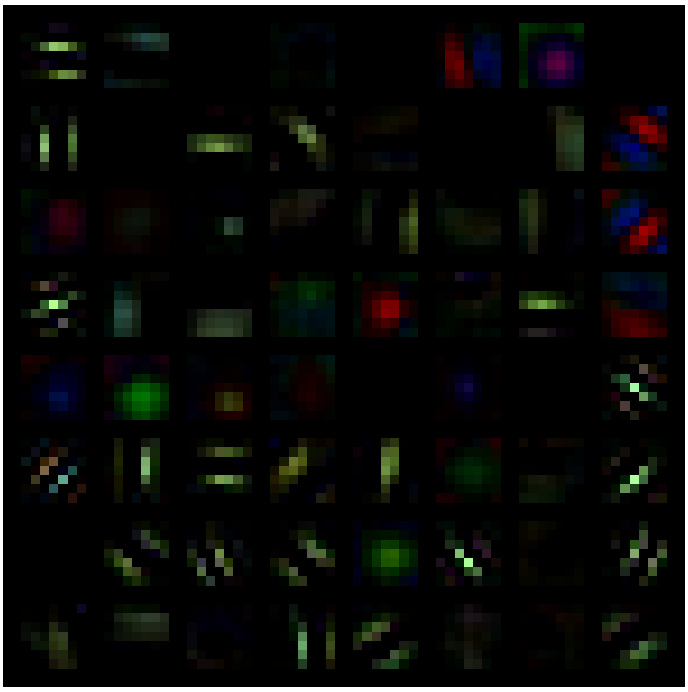

Figure 11: Weights in the first convolutional layer