Assignment 3

Gian Hug - András Izsó

March 4, 2022

Task 1

(a)

The image I is padded using zero-padding with $p = 1$. This ensures the output dimensions of 3×5. The output is the following:

2	-1	13	-1	-17
10	-4	8	2	-18
14	-1	-5	6	-9

(b)

(i) and (iii)

(c)

A padding of $p = 2$ should be used.

(d)

The kernel will have the size 9×9 due to

$$W_b = (W_a - F + 2P)/S + 1 = (512 - 9 + 0)/1 + 1 = 504$$

(e)

$$W_b = (W_a - F + 2P)/S + 1 = (504 - 2 + 0)/2 + 1 = 252$$

Thus the spatial dimension will be 252×252.

(f)

$$W_b = (W_a - F + 2P)/S + 1 = (252 - 3 + 0)/1 + 1 = 250$$

Thus the spatial dimension will be 250×250.

(g)

1. $\underbrace{(5 \cdot 5 \cdot 3 + 1)}_{\text{Single Filter with bias}} \cdot \underbrace{32}_{\text{count of filters}} = 2432$
2. $(5 \cdot 5 \cdot 3 + 1) \cdot 64 = 4864$
3. $(5 \cdot 5 \cdot 3 + 1) \cdot 128 = 9728$
4. After layer three the dimensions are $4 \times 4 \times 128$. The dimensions of the flattened output of the CNN are $2'048 \times 1$. Therefore layer 4 has $(2048 + 1) \cdot 64 = 131'136$ parameters.

5. $(64 + 1) \cdot 10 = 650$

6. Total: $148'810$ Parameters

Task 2

(a)

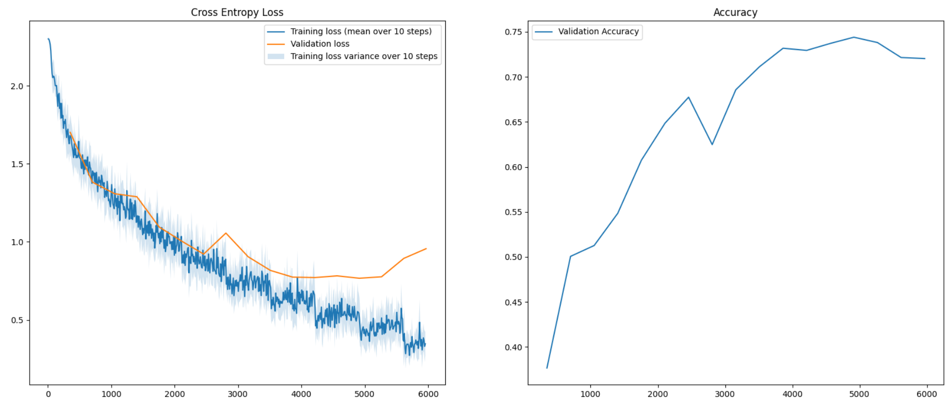

Figure 1: Plot of training for 2a)

(b)

To get the final accuracies the model with the lowest validation loss was loaded as the model using `load_best_checkpoint`. The model was set to inference model using `model.eval()` and then the accuracies were calculated over the whole data sets.

- Train Loss: 0.8677
- Final/Best Validation Accuracy: 0.7413
- Test Loss: 0.7422

Task 3

(a)

Layer	Layer Type	Number of Hidden Units/Filters	Activation Function
1	Conv2D	64	ReLu
1	Maxpool	-	-
1	BatchNorm	-	-
2	Conv2D	128	ReLu
2	Maxpool	-	-
2	BatchNorm	-	-
3	Conv2D	256	ReLu
3	Maxpool	-	-
3	BatchNorm	-	-
-	Flatten	-	-
4	Fully-Connected	64	ReLu
4	BatchNorm	-	-
5	Fully-Connected	10	SoftMax

Table 1: The architecture of the first network I reached >85% accuracy

In addition to the architecture in Table 1, I changed the optimizer to Adamax for faster convergence, and to make it work, I reduced the learning rate to $1e^{-3}$. The filter sizes, stride and padding remained the same (5×5, 1, 2). I also added $L2$ regularization and a 20% chance of dropout which showed minor increases in performance.

The architecture of the second network can be found at Table 3

(b)

Model	Train loss	Training accuracy	Validation accuracy	Test accuracy
1	1.2585	0.9433	0.7698	0.7742
2	0.0791	0.9782	0.820	0.8097

Table 2: The final results of the learning processes

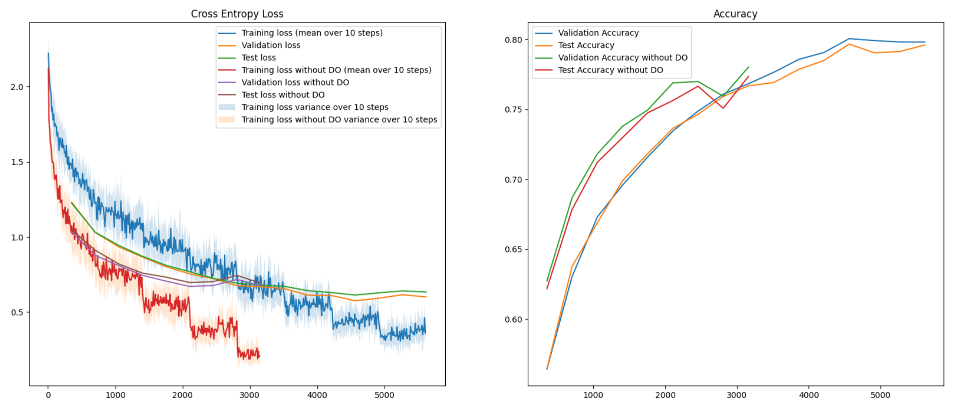

Figure 2: The results of our better model (no.2)

(c)

I saw improvement with increasing the filter numbers, introducing batch normalization, regularization and changing the optimizer to a more advanced one (Adamax). My first change was the optimizer (and with this the reduction of the learning rate) to possibly speed up the learning process. Adamax is a more sophisticated optimizer, which converges faster than Stochastic-Gradient-Descent. Then I tried changing the filter numbers, increasing them show an improvement. The introduction of batch normalization gave a big improvement, because it prevents the big differences between outputs of the activation functions, so certain nodes can't potentially suppress other ones. Regularization and dropout improved not the learning, but the accuracy on the test set, since these techniques opt for the creation of a more generalized result.

Changing the filter size yielded no improvement, but a bit of setback. I couldn't improve by data augmentation either. It's possible that I tried too harsh transformation, or would need more samples/other parameters changed to be able to learn. I didn't have the resources to dig deeper.

Changing pooling or striding didn't result in a noticeable change.

(d)

The biggest improvement resulted by the introduction of batch normalizations.

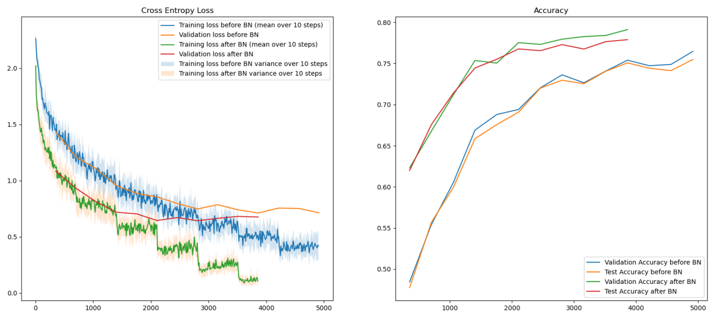

Figure 3:

(e)

Model 2 in Table 2

Layer	Layer Type	Number of Hidden Units/Filters	Activation Function
1	Conv2D	128	ReLu
1	Maxpool	-	-
1	BatchNorm	-	-
2	Conv2D	256	ReLu
2	Maxpool	-	-
2	BatchNorm	-	-
3	Conv2D	512	ReLu
3	Maxpool	-	-
3	BatchNorm	-	-
-	Flatten	-	-
4	Fully-Connected	64	ReLu
5	Dropout	-	-
5	Fully-Connected	10	SoftMax

Table 3: Model 2, Dropout with $p = 0.6$

- **Final test accuracy:** 80.97%

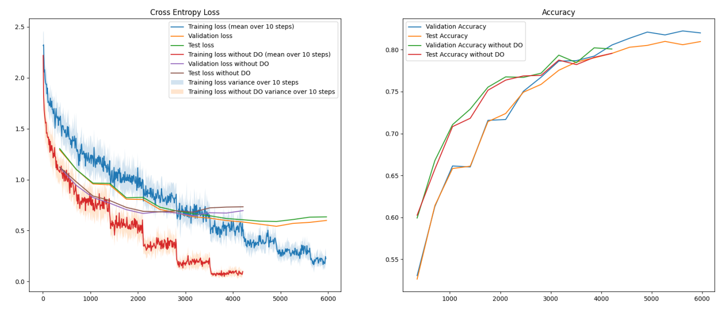

Figure 4: Model achieving over 80% test accuracy, Comparison with and without Dropout

(f)

According to Figure 4 training loss and validation loss look good and the early stopping does its job by preventing over fitting by aborting the learning after the validation loss stagnates. One indicator that it was about time to stop the learning process is the very high train accuracy (in inference mode, `model.eval()`) of over 97%. One would have to conduct further test to see if the model already did over fit but Figure 4 looks as if it did not over fit yet.

Task 4

(a)

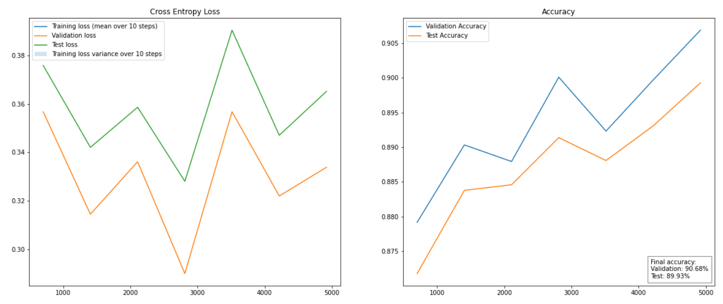

Figure 5: The results of the fine tuned Resnet18 model

I used the recommended settings:

- Optimizer: Adam
- Learning rate: $5 \cdot 10^{-4}$
- Batch size: 32

I didn't use any data augmentation (except the resizing to 224x244) since the model yielded the expected results. The final test accuracy was 89.93%.

(b)

In Figure 6 we can clearly see, that the first layer of the network tries to identify local features (for example edges). On the 3rd and 5th picture it also looks like as if the network is separating the subject from the background.

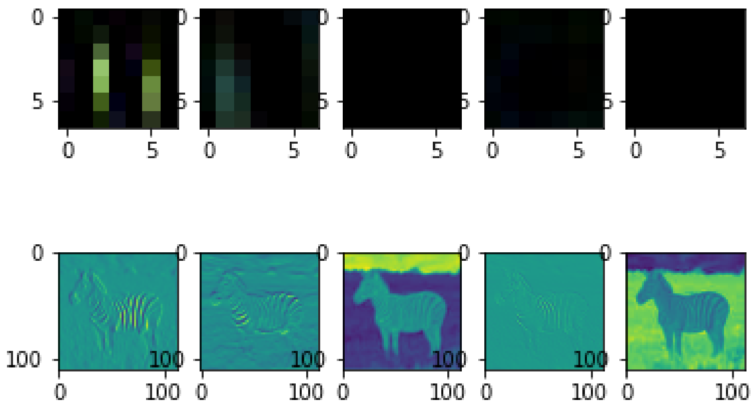

Figure 6:

(c)

In the last layer, as on Figure 7 it's visible, that the network is focusing more on structural elements, bigger areas, not just localized features. It is achieved by gradually downsampling the image, that's why the resolution is much worse, but this way each cell represents a bigger area of the original image.

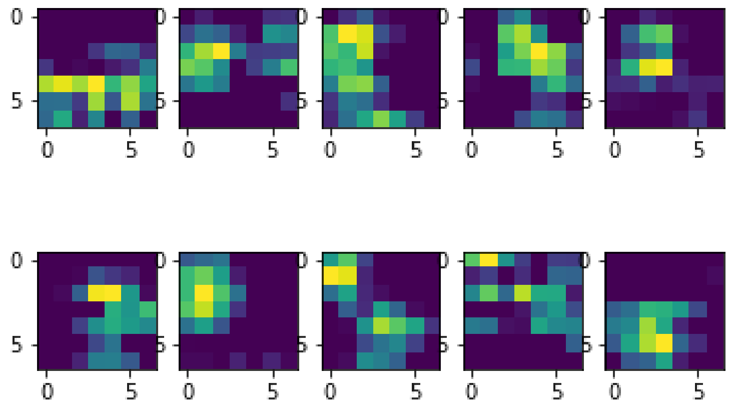

Figure 7: