Assignment 3 - Report

Sivert Utne

March 4, 2022

Contents

Task 1

(a)

The convolution is visualized in Figure 1, the multiplications and additions are not written out. The explanation part visualizes with colors how the values from the flipped kernel, and the respective area in the original image is multiplied and then summed together.

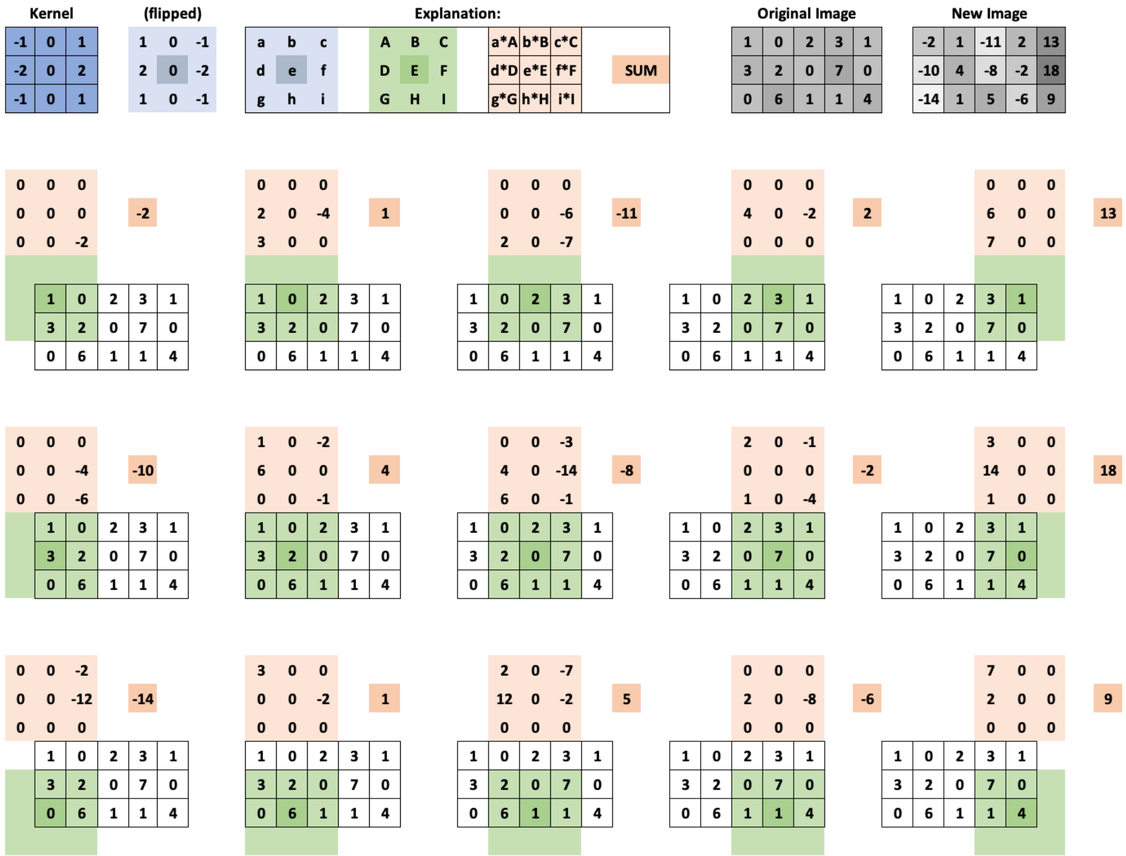

Figure 1: Convolution on an image with a sobel-kernel. All "pixels" outside the image are counted as 0. The resulting image is shown in the top-right corner

(b)

Out of the following:

(i) Convolutional Layer
(ii) Activation Function
(iii) Max Pooling

(iii) Max Pooling is the one that reduces the sensitivity to small translational variations in the output. This is because maxpooling will take the highest value of a range and return that as the new value, meaning with a MaxPooling kernel of size 3×3, if the highest value pixel in the image was located at the leftmost pixel of a kernel, and it was moved 2 pixels to the right, the output of that maxpooling kernel will be exactly the same. The same logic applies in all translational directions (up, down, left, right).

(c)

We have:

$$
\begin{aligned}
\text{kernel} = F_H \times F_W &\quad = 5 \times 5 \\
\text{stride} = S_H \times S_W &\quad = 1 \times 1 \\
\text{filters} &\quad = 6 \\
\text{Width out} = \text{Width in} \rightarrow W_1 = W_2 &\quad = W \\
\text{Height out} = \text{Height in} \rightarrow H_1 = H_2 &\quad = H
\end{aligned}
$$

To find the Padding in Width (P_W) and Height (P_H), we use equation 1 and 2 from "*A Brief Introduction to Convolutional Neural Networks*". This gives us:

$$
\begin{aligned}
W_2 &= \frac{W_1 - F_W + 2P_W}{S_W} + 1 \\
W &= \frac{W - F_W + 2P_W}{S_W} + 1 \\
S_W(W-1) &= W - F_W + 2P_W \\
S_W(W-1) - W + F_W &= 2P_W \\
P_W &= \frac{S_W(W-1) - W + F_W}{2} \\
P_W &= \frac{W - 1 - W + 5}{2} \\
\boldsymbol{P_W} &= \frac{4}{2} = \mathbf{2}
\end{aligned}
$$

and

$$
\begin{aligned}
P_H &= \frac{S_H(H-1) - H + F_H}{2} \\
P_H &= \frac{H - 1 - H + 5}{2} \\
\boldsymbol{P_H} &= \frac{4}{2} = \mathbf{2}
\end{aligned}
$$

We see that we should use a padding of $P_H \times P_W = \underline{\underline{2 \times 2}}$.

(d)

We know that:

$$\begin{aligned}
\text{size of original image} &= H_1 \times W_1 = 512 \times 512 \\
\text{number of layers} &= 2 \\
\text{spatial dimensions of the feature maps in the first layer} &= H_2 \times W_2 = 504 \times 504 \\
\text{number of feature maps in the first layer} &= 12
\end{aligned}$$

And we are told that we can assume:

$$\begin{aligned}
\text{kernels} &= N \times N && \text{where } N \text{ is odd} \\
\text{stride} &= S_H \times S_W && = 1 \times 1 \\
\text{padding} &= P_H \times P_W && = 0 \times 0
\end{aligned}$$

Again, using equation 1 and 2 from "*A Brief Introduction to Convolutional Neural Networks*", we get for the first layer that:

$$\begin{aligned}
W_2 &= \frac{W_1 - F_W + 2P_W}{S_W} + 1 \\
S_W(W_2 - 1) - W_1 - 2P_W &= -F_W \\
F_W &= -S_W(W_2 - 1) + W_1 + 2P_W \\
&= -1(504 - 1) + 512 + 2 * 0 \\
&= -504 + 1 + 512 \\
&= 9
\end{aligned}$$

and

$$\begin{aligned}
F_H &= -S_H(H_2 - 1) + H_1 + 2P_H \\
&= -1(504 - 1) + 512 + 2 * 0 \\
&= -504 + 1 + 512 \\
&= 9
\end{aligned}$$

In other words, we get a kernel size in layer 1 of $F_H \times F_W = \underline{\underline{\mathbf{9 \times 9}}}$

(e)

We have that:

$$\begin{aligned}
\text{Input} &= H_1 \times W_1 = 504 \times 504 \\
\text{Stride} &= S_H \times S_W = 2 \times 2 \\
\text{Subsampling Window Size} &= F_H \times F_W = 2 \times 2 \\
\text{Padding} &= P_H \times P_W = 0 \times 0
\end{aligned}$$

We get that the spatial dimensions of the pooled feature maps are:

$$\begin{gathered}
H_2 \times W_2 \\
\frac{H_1 - F_H + 2P_H}{S_H} + 1 \times \frac{W_1 - F_W + 2P_W}{S_W} + 1 \\
\frac{504 - 2 + 2 * 0}{2} + 1 \times \frac{504 - 2 + 2 * 0}{2} + 1 \\
\frac{502}{2} + 1 \times \frac{502}{2} + 1 \\
\mathbf{252} \times \mathbf{252}
\end{gathered}$$

(f)

We have that:

$$\begin{aligned}
\text{Input} &= H_1 \times W_1 = 252 \times 252 \\
\text{Kernel Size} &= F_H \times F_W = 3 \times 3 \\
\text{Stride} &= S_H \times S_W = 1 \times 1 \\
\text{Padding} &= P_H \times P_W = 0 \times 0
\end{aligned}$$

Which gives us a resulting dimension of the feature maps of:

$$\begin{gathered}
H_2 \times W_2 \\
\frac{H_1 - F_H + 2P_H}{S_H} + 1 \times \frac{W_1 - F_W + 2P_W}{S_W} + 1 \\
\frac{252 - 3 + 2 * 0}{1} + 1 \times \frac{252 - 3 + 2 * 0}{1} + 1 \\
249 + 1 \times 249 + 1 \\
\mathbf{250} \times \mathbf{250}
\end{gathered}$$

(g)

The image has dimensions 32×32 and is RGB $(32 \times 32 \times 3)$. Using the network outlined in Table 2 we get the following number of parameters for each layer:

Layer		Weights	Biases	Parameters
n	$F_H \times F_W \times C_1 \times C_2$		C_2	Weights + Biases
1	$5 \times 5 \times 3 \times 32$	2 400	32	2 432
2	$5 \times 5 \times 32 \times 64$	51 200	64	51 264
3	$5 \times 5 \times 64 \times 128$	204 800	128	204 928
4	$(4 * 4 * 128) \times 64$	131 072	64	131 136
5	64×10	640	10	650
$\sum_1^5 n$		390112	298	390 410

We see that the network has a total of **390 410** parameters.

Task 2

Table 2: Replica of Table 1 from the assignment. All *filters have size* 5 × 5 with *Padding=2* and *Stride=1*. The *MaxPool2D* layers has *kernel size* 2 × 2 and *Stride=2*.

Layer	Type	Filters / Hidden Units	Activation Function
1	Conv2D	32	ReLU
	MaxPool2D	-	-
2	Conv2D	64	ReLU
	MaxPool2D	-	-
3	Conv2D	128	ReLU
	MaxPool2D	-	-
	Flatten	-	-
4	Fully-Connected	64	ReLU
5	Fully-Connected	10	Softmax

Table 3: Hyperparameters for Model in Task 2.

Hyperparameter	value
Optimizer	SGD
Batch Size	64
Learning Rate	0.05
Early Stop Count	4
Epochs	10

(a)

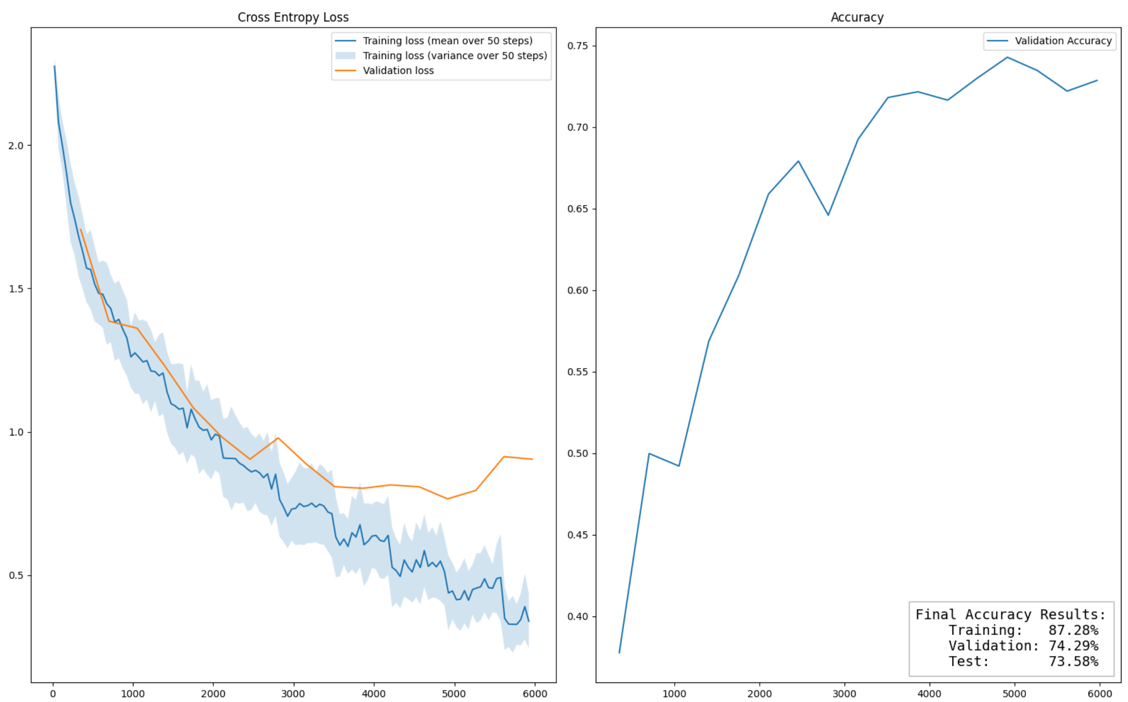

Figure 2: Training + Validation Loss and Validation Accuracy for CNN Model described in Table 2.

(b)

The final accuracies of the model[1] is written in Figure 2, and is also stated in Table 4.

Table 4: Final Accuracies of CNN model from Table 2.

Dataset	Accuracy
Train	87.28%
Validation	74.29%
Test	73.58%

[1] The Final Accuracies are after the ''best model" is loaded (model with lowest validation loss during training). This will be the case for all models. For the model described here that means the accuracies reported are from the model that has trained ≈ 5000 epochs.

Task 3

(a)

Model 1

The first model is an exact replica of the model from Table 2. The only differences are listed in Table 5.

Table 5: Hyperparameters for Model 1. Anything not listed are kept the same as they were in task 2 (Table 3), this is also the case for model 2:

Hyperparameter	value
Optimizer	Adam
Learning Rate	0.00049

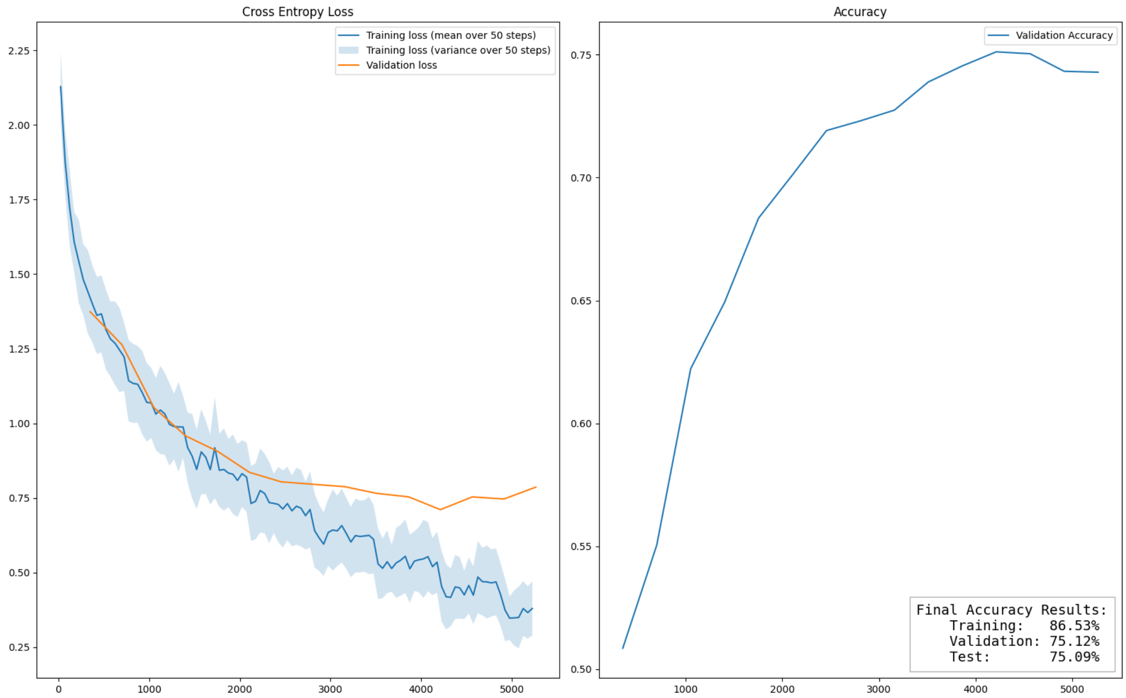

Figure 3: Training and Validation Loss + Training Accuracy for Model 1

Model 2

For my second model i wanted to increase the models complexity by adding more layers. In doing this i increased the learning rate to speed up the learning while also adding Dropout to two of the layers to ‘’force” the model to generalize. The Complete network is described in Table 6 and hyperparamters are listed in Table 7.

Table 6: Model 2. All *filters have size* 5×5 with *Padding=2* and *Stride=1*. The *MaxPool2D* layers has kernel size 2×2 and *Stride=2*.

Layer	Type	Filters / Hidden Units	Activation Function
1	Conv2D	32	ReLU
	MaxPool2D	-	-
2	Conv2D	64	ReLU
	MaxPool2D	-	-
3	Conv2D	128	ReLU
	MaxPool2D	-	-
	Flatten	-	-
	Dropout(0.4)	-	-
4	Fully-Connected	256	ReLU
5	Fully-Connected	128	ReLU
	Dropout(0.2)	-	-
6	Fully-Connected	64	ReLU
7	Fully-Connected	32	ReLU
8	Fully-Connected	10	Softmax

Table 7: Hyperparameters for Model 2.

Hyperparameter	value
Learning Rate	0.15

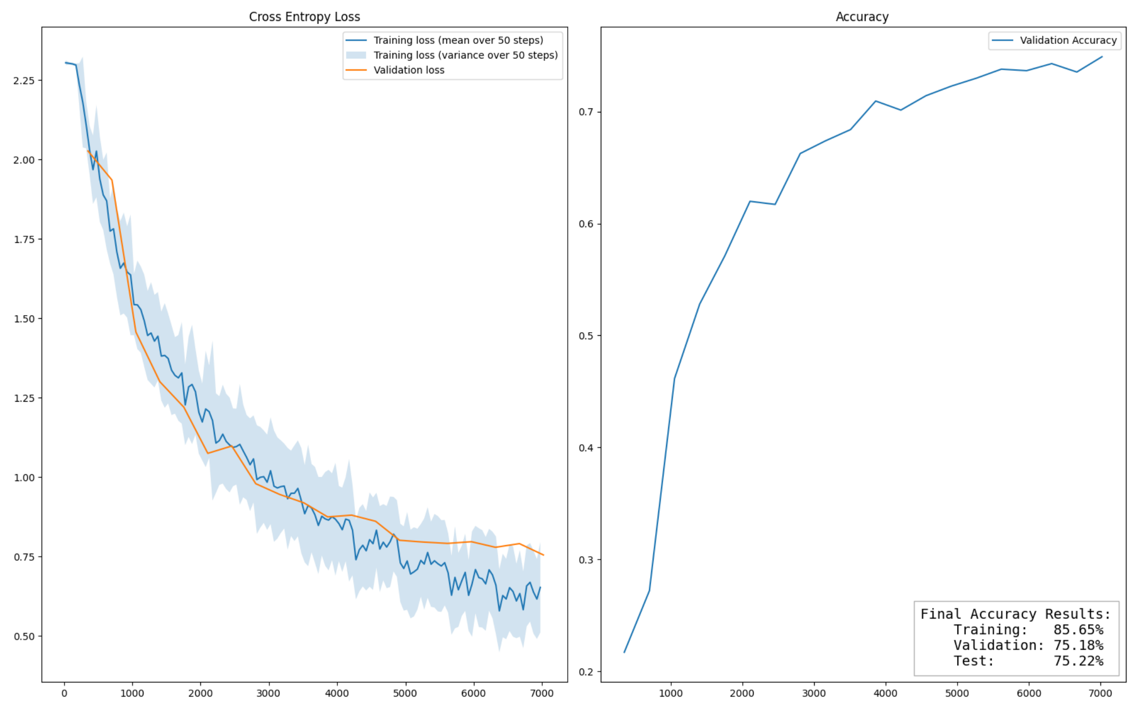

Figure 4: Training and Validation Loss + Training Accuracy for Model 2

(b)

The final results for the two models are listen in Table 8:

Table 8: Accuracies (in percent) for the two models described in Task 3a.

Dataset	Model 1 Accuracy	Model 2 Accuracy
Training	86.53%	85.65%
Validation	75.12%	75.18%
Test	75.09%	75.22%

From the table we that **Model 2** has the best results. Its training and validation loss, along with its validation accuracy is plotted in Figure 4.

(c)

I believe i got quite lucky with the improvements i wanted to test in such that the first things i tried worked well enough that i managed to squeeze them over 75% test accuracy. But that being said to get it over 80% in Task 3e i tried a few more improvements and here is the general impression i have of the positive effects of each of them (keeping in mind that too much of any of them can slo lead to negative effect, but using each of them in fitting amounts should give the effects listed):

Improvement	Effect
Adam Optimizer	Increased Learning rate
More Classification Layers	Able to learn distinguish between more features
More Feature Extraction Layers	Able to extract more features
Dropout	Prevent overfitting
Data Augmentation	Better generalization
Batch Normalization	Faster Training

(d)

Based on the limited experimentation i performed before both models were above 75% i found the biggest improvement after adding Dropout to Model 2.

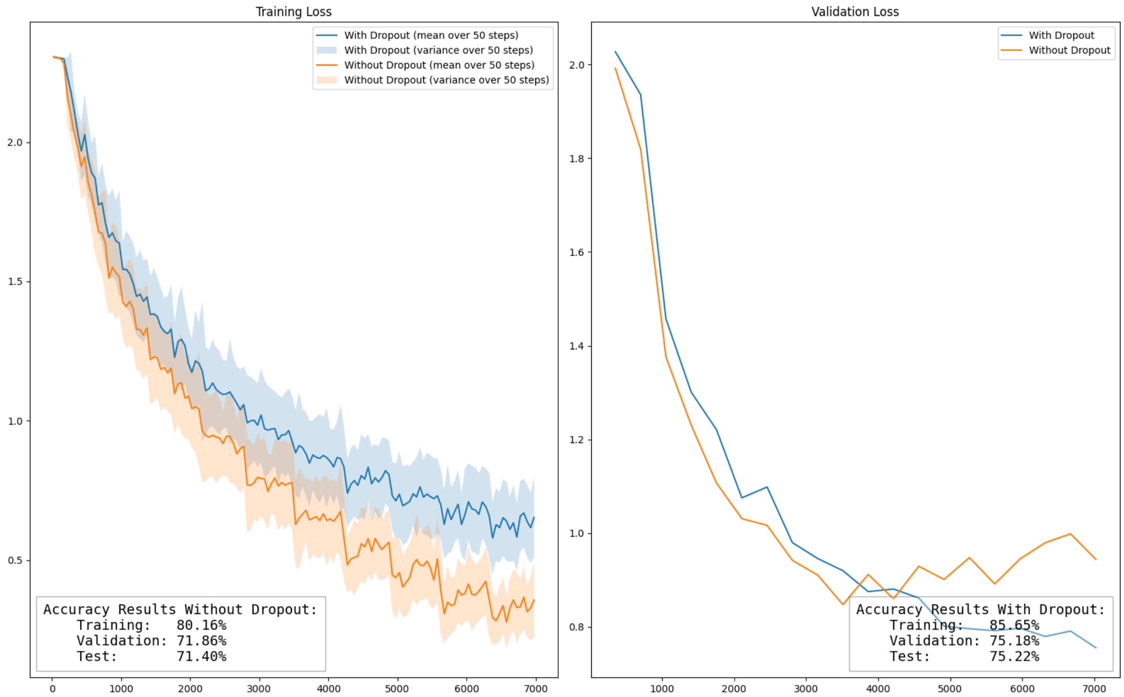

Figure 5: Comparison of the same model with and without Dropout

As we can see the model overfits much less and is able to achieve far better generalization.

(e)

After some more experimentation with improvements i ended up with this model:

Table 10: My Best Model. All *filters have size* 5×5 with *Padding=2* and *Stride=1*. The *MaxPool2D* layers has kernel size 2×2 and *Stride=2*.

Layer	Type	Filters / Hidden Units	Activation Function
1	Conv2D	32	ReLU
	BatchNorm2D	-	-
2	Conv2D	32	ReLU
	MaxPool2D	-	-
3	Conv2D	64	ReLU
	BatchNorm2D	-	-
4	Conv2D	64	ReLU
	MaxPool2D	-	-
5	Conv2D	128	ReLU
	BatchNorm2D	-	-
6	Conv2D	128	ReLU
	MaxPool2D	-	-
	Flatten	-	-
	Dropout(0.4)	-	-
7	Fully-Connected	128	ReLU
8	Fully-Connected	64	ReLU
	Dropout(0.2)	-	-
9	Fully-Connected	64	ReLU
10	Fully-Connected	10	Softmax

Table 11: Hyperparameters for my best model.

Hyperparameter	value
Optimizer	Adam
Batch Size	32
Learning Rate	0.0006

In addition to this i added **Data Augmentation** with:

- Random Horizontal Flip (50% probability)
- Random Rotations from -10° to 10°
- ColorJitter (0.5 for brightness, contrast, saturation and hue)

The results of the model are seen in Figure 6.

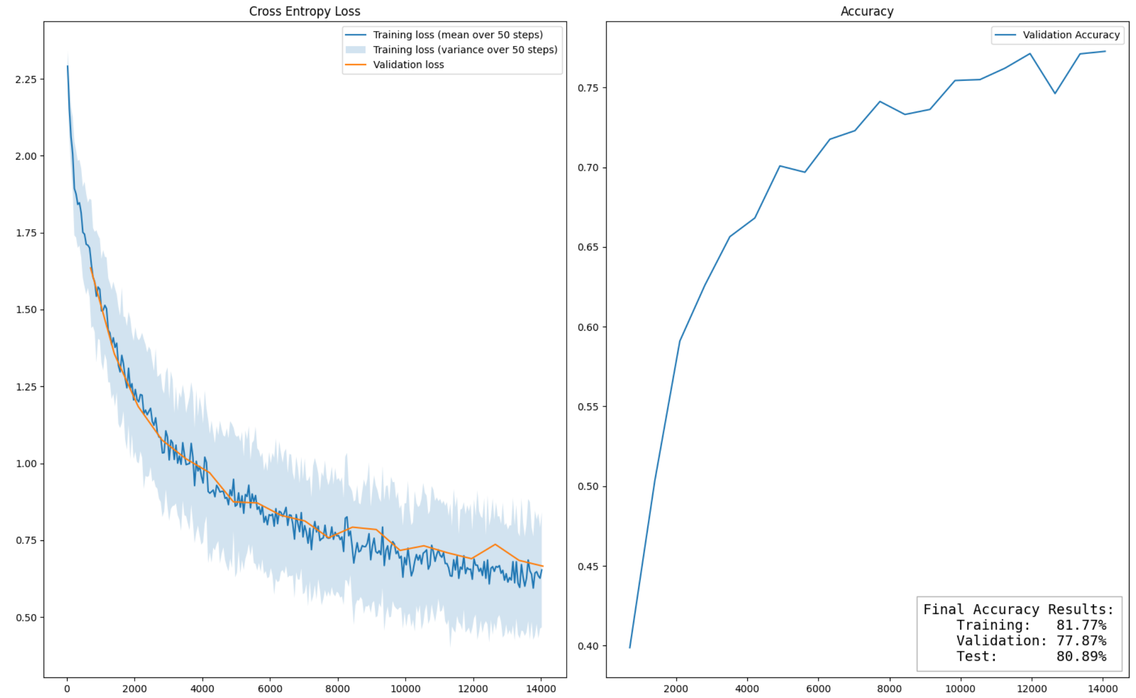

Figure 6: My Final Models Train and Validation Loss, along with its validation Accuracy. Test Accuracy is stated in lower right corner.

(f)

As we can see from the train and validation loss in Figure 6, the validation and training loss are pretty much equal at all times, this means the model doesn't show any signs of over/under-fitting. The model would be underfitting if the validation loss failed to keep up with the training loss, and overfitting if the validation loss had started to increase as the model overfits to the training data. This is also backed up by how the model has almost equal training and test accuracy.

Task 4

(a)

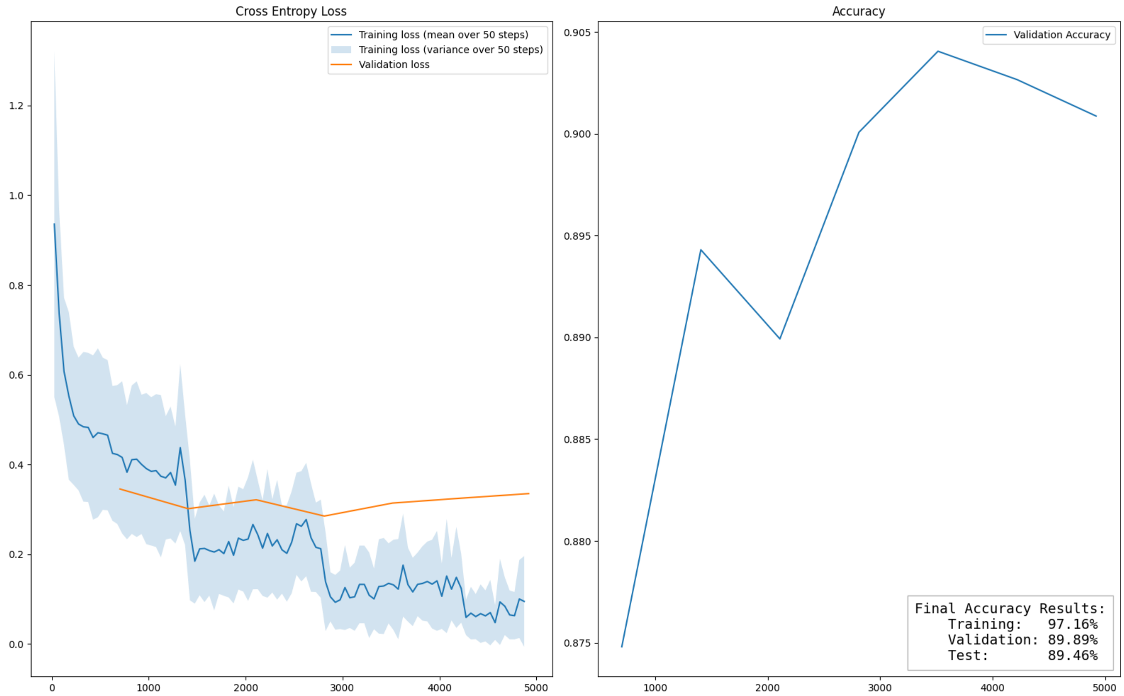

Figure 7: Training + Validation Loss and Validation Accuracy for ResNet18 Model. The Test Accuracy for the model with lowest validaton loss is listed in the bottom right.

The hyperparamters used by the model from Figure 7 are listed in Table 12.

Table 12: Hyperparameters used for training the ResNet18 model on CIFAR10.

	Value
Optimizer	Adam
Batch Size	32
Learning Rate	0.0005 (5×10^{-4})

The following transformationa were also applied to the images:

- Resize to 224×224
- Normalization with:
  - mean $= (0.485, 0.456, 0.406)$
  - std $= (0.229, 0.224, 0.225)$

(b)

Figure 8: zebra.jpg

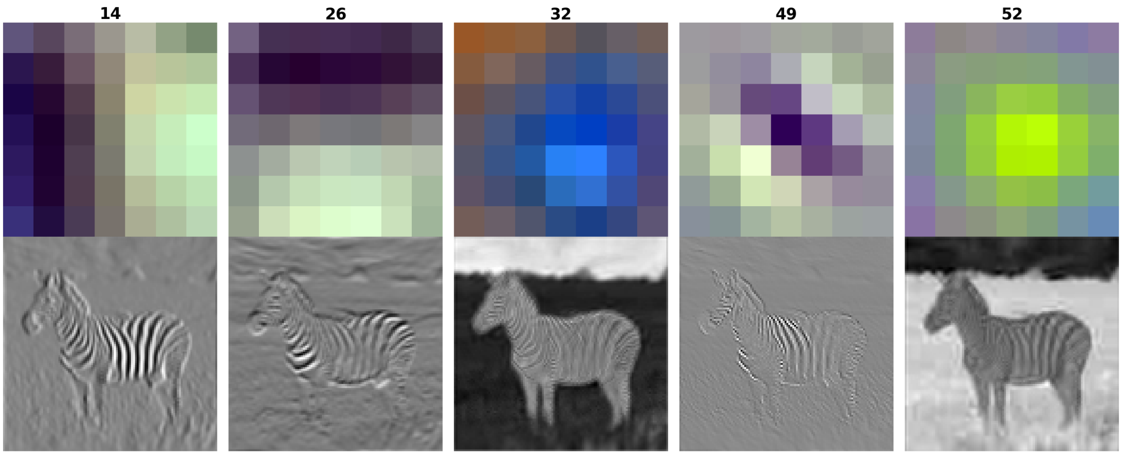

Figure 9: Visualization of filters and activations in ResNet18 on "zebra.jpg". Each column visualizes the 7×7 filter (top) of the first layer, and the corresponding grayscale activation (bottom). This is done on the following indices: [14, 26, 32, 49, 52]

In Figure 9 we see how the filters extract different features in the image. Each of the filters extracts the following features:

Filter	Feature	Example that has this feature
14	Vertical lines	Stripes and edges of the Zebra
26	Horizontal Lines	Stripes and edges of the Zebra
32	Blue Color	The Sky
49	Diagonal Lines (Down to the right)	Stripes and edges of the Zebra
52	Green Color	The Grass (yellow=green+red)

(c)

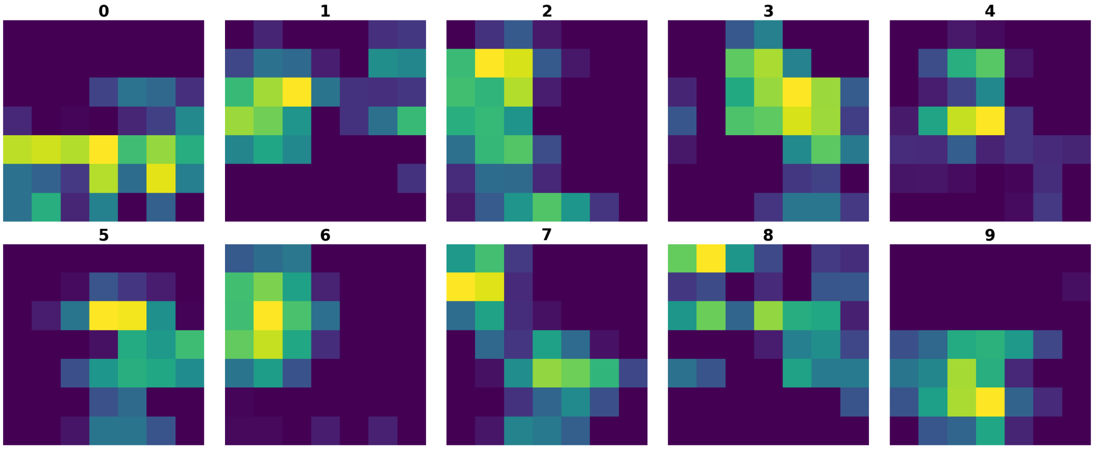

Figure 10: Visualization of activations in ResNet18 on “zebra.jpg” after passing through all but the last two convolutional layers. This is done for activations with indices 0-9

At this point in the model the activations no longer make much sense to us humans, but we can notice how all of the highest activations (yellow) are in what seems to be locations of the zebra. For instance:

- activation 0: The legs of the zebra
- activation 6: The head of the zebra