NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4265 - Computer Vision and Deep Learning

# Assignment 3

Kolbjørn Kelly
Sander Endresen

5th March 2021

## Contents

# 1 Task 1 - Theory

## 1.1 1a) Spatial Convolution

We are given a 3 × 5 image and a 3 × 3 kernel shown in figure 1. To ensure that the final image is of dimensions 3 × 5, two rows and two columns was added, with a step size of 1.

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 1 |
| 3 | 2 | 0 | 7 | 0 |
| 0 | 6 | 1 | 1 | 4 |

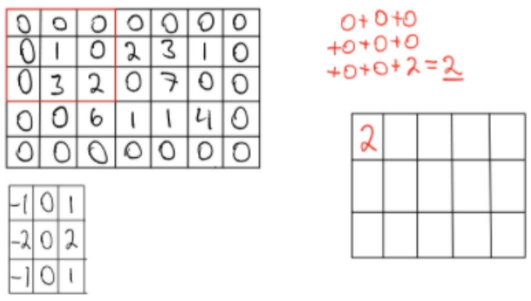
(1a) 3 × 5 image

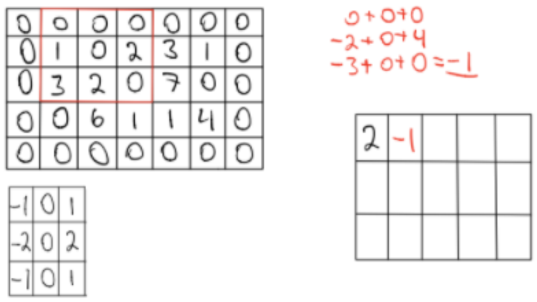
| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

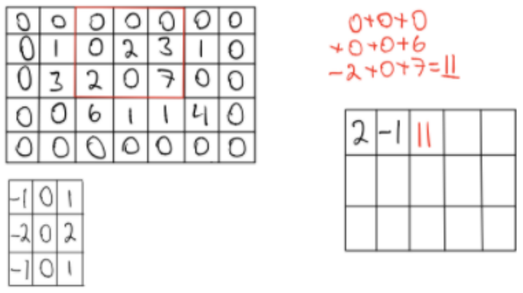
(1b) 3 × 3 kernel

Figure 1: 3 × 5 image and 3 × 3 kernel for spatial convolution

The following calculations have been done by hand:







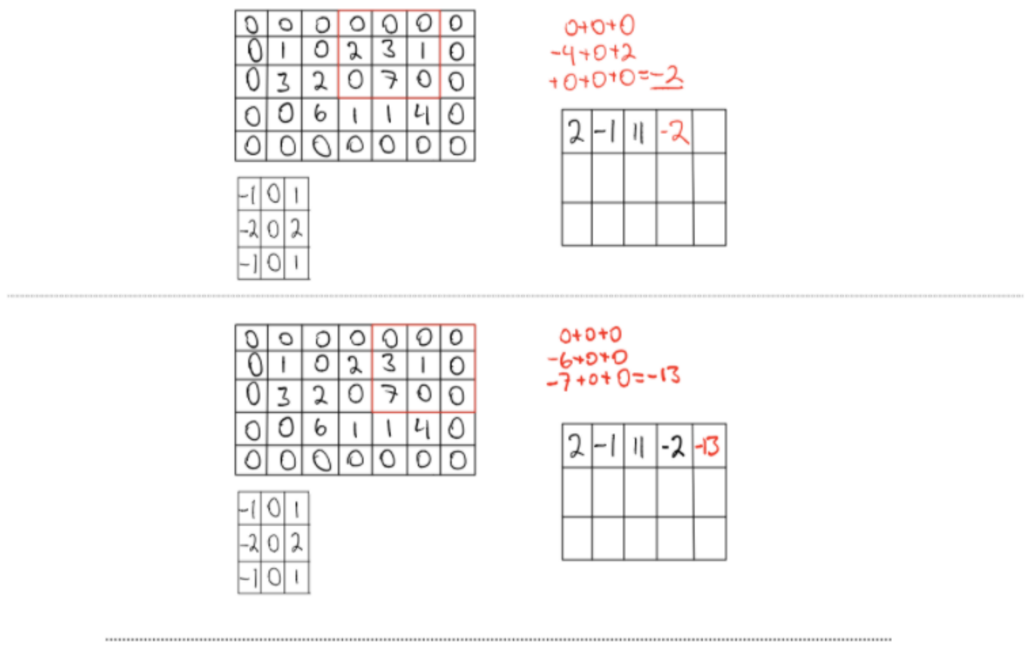


Figure 2: Convolution of image (1a) with kernel (1b) by hand

By moving one row down and repeating this for the upcoming rows, one ends up with the 3 × 5 convolved image shown in 3.

| | | | | |
|---|---|---|---|---|
| 2 | -1 | 11 | -2 | -13 |
| 10 | -4 | 8 | 2 | -18 |
| 14 | -1 | -5 | 6 | -9 |

Figure 3: Resulting convolved 3 × 5 image

## 1.2 1b) Spatial Translation Invariance

(i) As the kernel ”moves” over the image, it clearly reduces the sensitivity to translational variations.

(ii) The activation functions are connected to certain input nodes, hence it does not reduce sensitivity to translational variations.

(iii) Max Pooling is mainly used to reduce the number of parameters in the network, and does not reduce translational sensitivity on its own.

## 1.3 1c) Padding for single convolutional layer

We have the following relationships shown in (1) and (2):

$$\mathrm{W}_2 = \frac{\mathrm{W}_1 - \mathrm{F}_\mathrm{W} + 2\mathrm{P}_\mathrm{W}}{\mathrm{S}_\mathrm{W}} + 1 \tag{1}$$

with values as follows: W = width, $P_W$ = padding width, $S_W$ = stride width, $F_W$ = kernel width.

$$H_2 = \frac{H_1 - F_H + 2P_H}{S_H} + 1 \tag{2}$$

where H = height, $P_H$ = padding height, $S_H$ = stride height, $F_H$ = kernel height.

We now have $S_W = S_H = 1$ and $F_W = F_H = 5$. We want to find $P_W$ and $P_H$ such that $W_1 = W_2$ and $H_1 = H_2$. By using the relationships in (1) and (2) we get:

$$\begin{aligned} W_2 &= W_1 - 5 + 2P_W + 1 \\ 2P_W &= 5 - 1 \iff P_W = 2 \end{aligned}$$

which gives a padding width of 2. To get the padding height we calculate

$$\begin{aligned} H_2 &= H_1 - 5 + 2P_H + 1 \\ 2P_H &= 5 - 1 \iff P_H = 2 \end{aligned}$$

and get a padding height of 2 as well. This means that the amount of padding that should be added is two rows and two columns.

## 1.4 1d) Spatial dimensions of kernels

We now consider a CNN with inputs as 512 × 512 RGB color images in two convolutional layers, yielding the structure 512 × 512 × 3. Then the first layer will be 504 × 504 × 12, and we have $S_H = S_W = 1$ and $F_H = F_W = n = 2i + 1$. To find $F_H$ and $F_W$, we simply do:

$$\begin{aligned} 504 &= 512 - F_W + 1 \\ F_W &= 512 - 504 + 1 = 9 \end{aligned}$$

$$\begin{aligned} 504 &= 512 - F_H + 1 \\ F_H &= 512 - 504 + 1 = 9 \end{aligned}$$

## 1.5 1e) Pooled feature maps in the first layer

We now have $H_1 = W_1 = 504$, $S_W = S_H = 2$ and $F_W = F_H = 2$. This gives us the width and height as follows:

$$\begin{aligned} W_2 &= \frac{504 - 2}{2} + 1 = 252 \\ H_2 &= \frac{504 - 2}{2} + 1 = 252 \end{aligned}$$

## 1.6 1f) Feature maps in the second layer

As $F_{W2} = F_{H2} = 3$ and $S_{W2} = S_{H2} = 1$ we get

$$\begin{aligned} W_3 &= 252 - 3 + 1 = 250 \\ H_3 &= 252 - 3 + 1 = 250 \end{aligned}$$

### 1.7 1g)

To compute the number of parameters in the convolutional layers, the formula

$$params = F_H \times F_W \times C_{in} \times C_{out} + C_{out} \tag{3}$$

is used. Employing this on all convolutional layers gives

$$\begin{aligned}
params_1 &= 5 * 5 * 3 * 32 + 32 = 2432 \\
params_2 &= 5 * 5 * 32 * 64 + 64 = 51264 \\
params_3 &= 5 * 5 * 64 * 128 + 128 = 204928
\end{aligned}$$

For the fully connected layers, we use the formula

$$params = inputs \times outputs + bias. \tag{4}$$

To, get the number of inputs to the first fully connected layer, $H_3$ and $W_3$ must be calculated. Using formula 1 and 2, we get

$$\begin{aligned}
H_1' &= W_1' = W_0 - F_W + 2P_W + 1 = 32 - 5 + 4 + 1 = 32 \\
H_1 &= W_1 = \frac{W_1' - F_{WP}}{S_{WP}} + 1 = \frac{32-2}{2} = 16 \\
H_2' &= W_2' = W_1 - 5 + 4 + 1 = 16 \\
H_2 &= W_2 = \frac{16-2}{2} + 1 = 8 \\
H_3' &= W_3' W_2 - 5 + 4 + 1 = 8 \\
H_3 &= W_3 = \frac{8-2}{2} + 1 = 4
\end{aligned}$$

That yields

$$\begin{aligned}
params_4 &= 4 * 4 * 128 * 64 + 64 = 131136 \\
params_5 &= 64 * 10 + 10 = 650
\end{aligned}$$

Summing up all the parameters, we get a total of 390 410 parameters in the network.

## 2 Task 2 - Convolutional Neural Networks

### 2.1 2a)

The training and validation loss over the training period is shown in figure 4.

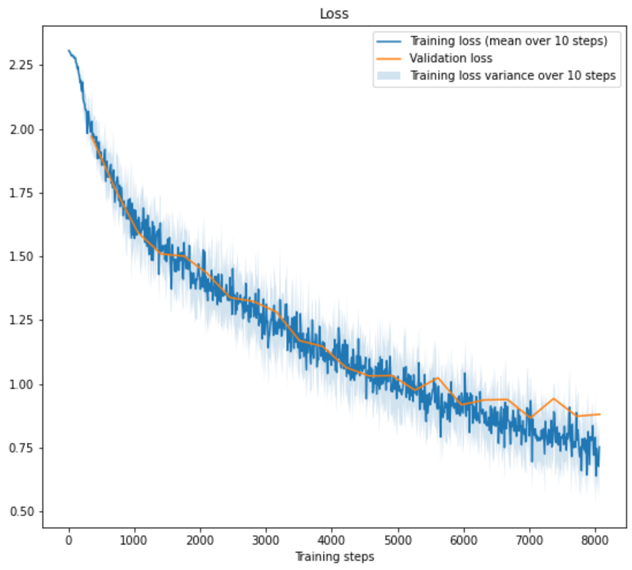


Figure 4: 2a) Training and validation loss

### 2.2 2b)

The final results over the entire respective dataset were the following

| | Train | Validation | Test |
|---|---|---|---|
| Accuracy | 0.745 | 0.693 | 0.69 |
| Loss | 0.72 | 0.87 | 0.88 |

## 3 Task 3 - Deep Convolutional Network for Image Classification

### 3.1 3a Design of two Convolutional Neural Network for the CIFAR-10

Initially, both the following models were built on the model from the previous task. No architectural features were changed, other than the ones explicitly specified.

#### 3.1.1 Model X

For the first model, the architecture from task 2 was kept unchanged. Hence, it was the following:

| heightLayer | Layer Type | Number of hidden units / Filters | Activation Function |
|---|---|---|---|
| 1 | Conv2D | 32 | ReLU |
| 1 | MaxPool2D | - | - |
| 2 | Conv2D | 64 | ReLU |
| 2 | MaxPool2D | - | - |
| 3 | Conv2D | 128 | ReLU |
| 3 | MaxPool2D | - | - |
| | Flatten | - | - |
| 4 | Fully-Connected | 64 | ReLU |
| 5 | Fully-Connected | 10 | Softmax |

Also, the filter size (5 × 5), padding (2), stride (1) and batch size (64) was kept unchanged, as well as the optimizer (SGD).

The only technique applied to this model was data-augmentation. The data was augmented by the following transforms

- RandomCrop() was added to crop an image at a random location
- RandomHorizontalFlip(p) to horizontally flip an image with a probability of p.
- ColorJitter() to randomly change the brightness, saturation and contrast of an image.
- RandomGrayScale(p) to convert an image to grayscale with a probability of p

Employing the function RandomApply(p), these transformation where applied with a probability of p. Note that the use of this function introduces nested probabilities. Accordingly a shared value of $p = 0.5$ was used.

Lastly, the learning rate was set to $1.5e - 2$.

The model did not early stop during the 10 epochs.

#### 3.1.2 Model Y

For the second model, the filter size (5 × 5), padding (2), stride (1) and batch size (64) was kept unchanged. The architecture was only altered by adding spatial batch normalization after the convolutional layers, yielding

| heightLayer | Layer Type | Number of hidden units / Filters | Activation Function |
|---|---|---|---|
| 1 | Conv2D | 32 | ReLU |
| 1 | MaxPool2D | - | - |
| 2 | Conv2D | 64 | ReLU |
| 2 | MaxPool2D | - | - |
| 3 | Conv2D | 128 | ReLU |
| 3 | MaxPool2D | - | - |
|  | BatchNorm2D | - | - |
|  | Flatten | - | - |
| 4 | Fully-Connected | 64 | ReLU |
| 5 | Fully-Connected | 10 | Softmax |

Other changes made were

- Added dropout with a probability of $p = 0.25$
- Used ADAM-optimizer
- Learning rate = $1.5e - 3$

The model early stopped after 6 epochs.

### 3.2 3b)

The results are shown in the following table. The models performs similar, but we chose to continue with the model performing the best on the test set, namely model Y. The validation performance of this model is shown in figure 5.

| Model X / Model Y | Train | Validation | Test |
|---|---|---|---|
| Accuracy | 0.89 / 0.91 | 0.82 / 0.768 | 0.756 / 0.76 |
| Loss | 0.33 / 0.27 | 0.53 / 0.72 | 0.78 / 0.77 |

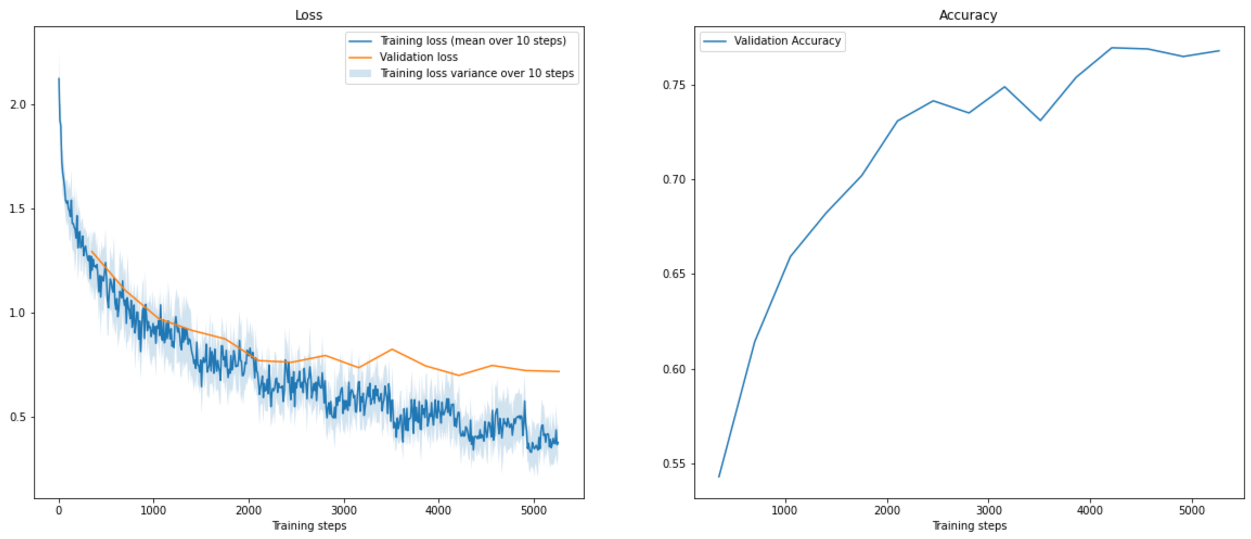


Figure 5: 3b) Validation data for model Y

## 3.3 3c) Methods discussion

Augmenting the data set was, without a doubt, the most effective technique. Augmenting the data yields a larger data set for the network to train on by changing aspects on the given data, in this case effectively doubling the training set. In this case, it is clear by the improvement in test accuracy that the data set was altered enough to be useful.

A technique that yielded little to no results was changing the number of filters in the convolutional part of the network. The following was tried

- $num\_filters = 64$
  - Validation: loss=0.95, accuracy=0.658
- $num\_filters = 16$
  - Validation: loss=1.02, accuracy=0.637

Here, $num\_filters$ indicate the number of filters for the first layer. The number of filters for the following layers were scaled by the same factor as earlier.

These results indicate that the initial choice of filters were suitable for the given problem. In theory, a greater number of filters should be able to capture higher abstractions from the input data. The reason $num\_filters = 16$ yielded poor results can be that this network simply is not able to catch the complexity of the image shapes. The reason $num\_filters = 64$ yielded poor results can be that this network caught to much noise from the images. This aspect is typically why the number of filters increase through the convolutional layers.

## 3.4 3d)

A comparison between the base model and model X (only data augmentation was applied here) is shown in figure 6. Note that the learning rate was also adjusted, as we found this necessary to emphasize the impact of the data augmentation. Also note that data augmentation increases the data set, thus also the number of training steps, as seen by figure 6.

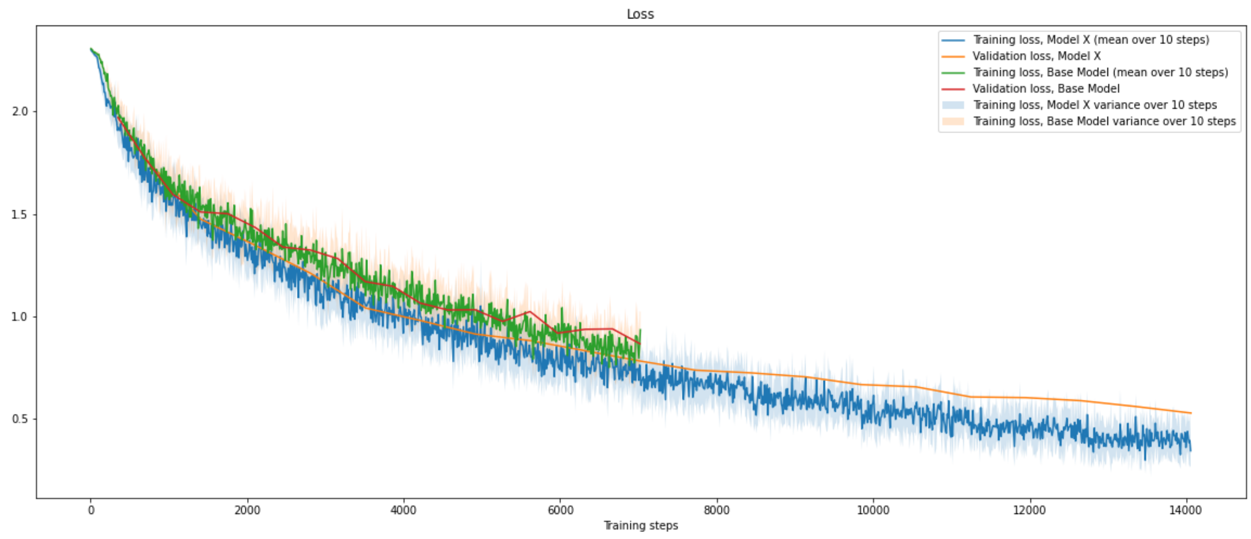


Figure 6: Models with and without data augmentation

### 3.5 3e) Improved model

To improve on model Y, the data augmentation used on model X was employed. In addition, some minor tweaks were made

- Learning rate was set to $1e - 3$
- Dropout probability was set to $p = 0.7$

The model did not early stop, and it produced the following results over the entire data set

| | Train | Validation | Test |
|---|---|---|---|
| Accuracy | 0.884 | 0.842 | 0.804 |
| Loss | 0.35 | 0.47 | 0.58 |

The evolution of the validation accuracy over training is shown in figure 7.

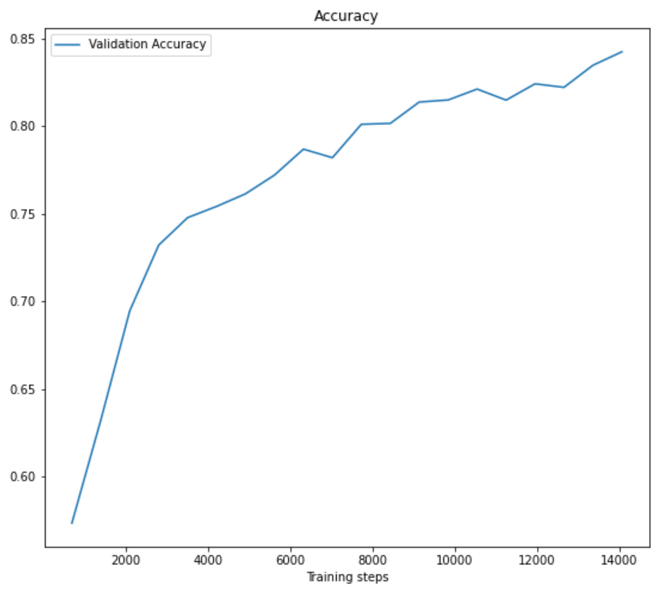


Figure 7: 3e) Validation Accuracy for the final model

### 3.6 3f) Overfitting on Improved Model

The performance on training data being better than the performance on the validation data which, in turn, is better than the performance on the test set, does indicate some degree of overfitting. In other words, it indicates that the model thinks it performs better than it actually does.

## 4 Task 4 - Transfer Learning with ResNet

### 4.1 4a) Transfer learning with Resnet18

As suggested by the assignment text, the following parameters was used

- Learning rate = $5e - 4$
- Batch size = 32
- Optimizer = ADAM

Furthermore, the data augmentation from the previous task was employed.A plot of training and validation loss is shown in figure 8.

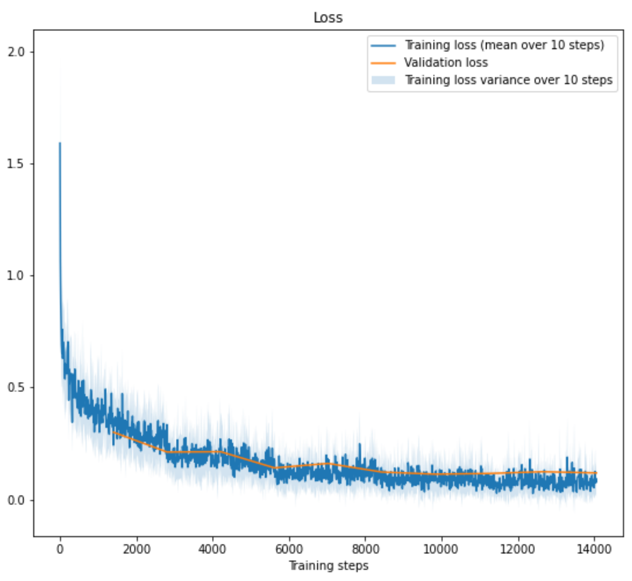


Figure 8: 4a) Validation loss using ResNet18 transfer learning

The model early stopped after epoch 5, and gave the following results

| | Train | Validation | Test |
|---|---|---|---|
| Accuracy | 0.984 | 0.963 | 0.909 |
| Loss | 0.05 | 0.12 | 0.35 |

## 4.2 4b) Visualizing filters

The weight visualization is shown in figure 9.

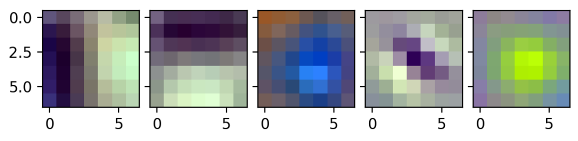


Figure 9: 4b) Visualizing the weights

The activation visualization is shown in figure 10. Clearly, the first convolutional layer recognizes relatively simple shapes present in the image. This correlates well with the idea that a networks recognizes more abstract features as the convolutional depth increases.

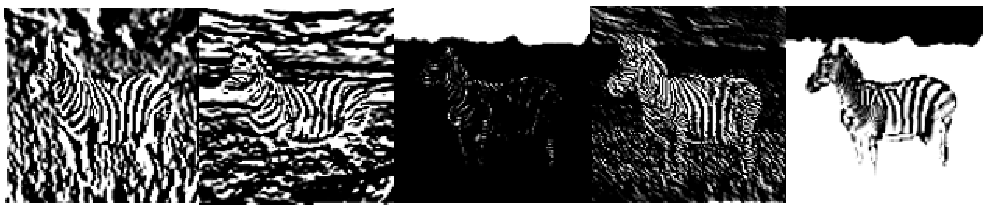


Figure 10: 4b) Visualizing the activations

## 4.3 4c) Visualizing the ten first filters from the last convolutional layer

The activations from the convolutional layers are visualized in figure 11. As expected, we see more abstract shapes as the depth increases. The first images are clearly perceptible by a human, but deeper into the network the model recognizes shapes that are simply too abstract for a human to understand.

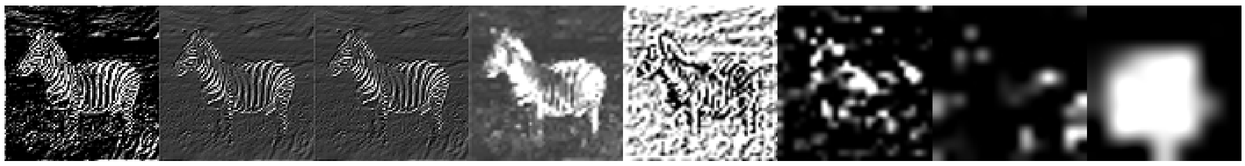


Figure 11: 4c) Visualizing the activations from the convolutional layers