Introduction

- A novel language which just consists of three geometrical signs (squares, circles and triangles) was recently found. There is a huge interest in discretizing handwritten images of this language. You should solve this task by developing a CNN model. It gets handwritten 4x4 pixel images as an input and should assign each image to one of the three sign classes. The architecture of the network is visible in the table below. Always use zero padding. The filter in the convolutional layer has the size 3x3 with a padding and a stride of 1. The max pooling layer has a 2x2 filter with a stride of two and no padding. The prediction is performed using Softmax and the loss is calculated using Cross Entropy.

Layer	Layer Type	Number of Filters/Number of hidden units	Activation Function
1	Conv2D	1	ReLU
1	MaxPool2D	-	-
Flatten	-	-	-
2	Fully-Connected	3 units for the three output classes	Softmax

Task 1: Compute the image sizes after the convolution and the max pooling.

Solution:

Use the following formula to compute the image size after the convolution:

$$\mathrm{W_{out}} = \frac{W_{in} - F + 2 * P}{S} + 1$$

W_{in}: Input size of the image, W_{out}: Output size of the image, F: size of the filter, P: Padding, S: Stride.

Image size after the convolution:

$$\mathrm{W_{out}} = \frac{4 - 3 + 2 * 1}{1} + 1 = \mathbf{4}$$

The output after the convolution has the size **4x4 pixels**.

Image size after the max pooling:

$$\mathrm{W_{out}} = \frac{4 - 2 + 2 * 0}{2} + 1 = 2$$

The output after the max pooling has the size **2x2 pixels**.

Task 2: Compute a forward path through the convolutional and max pooling layer.

The input image I= | 3 | 2 | 3 | 4 |
|---|---|---|---|
| 4 | 0 | 1 | 4 |
| 3 | 1 | 2 | 3 |
| 4 | 3 | 4 | 4 |

and the filter B= | 0 | 1 | -2 |
|---|---|---|
| 0 | 1 | 0 |
| -1 | 0 | 2 |

are given.

Some parts of the output after the convolution have already been processed so that only the missing values need to be calculated. Preliminary convolution:

	0	11	3
5		1	
13	3	0	3
5		0	7

Solution: Perform the image convolution with zero padding and a stride of one. The first entry is exemplary calculated but the procedure is the same for the other values: $out_{conv}(1,1) = 0 * 0 + 0 * 1 + 0 * -2 + 0 * 0 + 3 * 1 + 2 * 0 + 0 * -1 + 4 * 0 + 0 * 2 = 3$

$Out_{conv} =$

3	0	11	3
5	-3	1	6
13	3	0	3
5	0	0	7

Apply ReLU →

3	0	11	3
5	0	1	6
13	3	0	3
5	0	0	7

Apply max pooling →

Output after the first layer:

5	11
13	7

Task 3: Compute the predicted output of the network (use the result from the previous task.). The weight matrix and the bias of the fully connected layer are: $\mathrm{w} = \begin{bmatrix} 0.5 & 0.2 & -0.1 & 0.3 \\ -0.4 & -0.2 & 0 & 0.1 \\ 0 & 0.2 & 0.6 & -0.3 \end{bmatrix}$, $\mathrm{b} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$. The output of the first neuron corresponds to the circle class (k=1), of the second neuron to the triangle class (k=2) and of the third neuron to the squares class (k=3). To which class will the network assign the image ?

Solution 3.1: The flattened output a_1 from the first convolutional layer is $\begin{bmatrix} 5 \\ 11 \\ 13 \\ 7 \end{bmatrix}$. The output $z_2 = w * a_1 + b$ can now be calculated:

$$z_2 = \begin{bmatrix} 0.5 & 0.2 & -0.1 & 0.3 \\ -0.4 & -0.2 & 0 & 0.1 \\ 0 & 0.2 & 0.6 & -0.3 \end{bmatrix} \begin{bmatrix} 5 \\ 11 \\ 13 \\ 7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 6.5 \\ -3.5 \\ 9.9 \end{bmatrix}$$

Next slide

Solution3.2:

The predicted outputs $\hat{y}_k$ of the network are calculated using Softmax.

The denominator is :

$$denom = \sum_{k=1}^{3} e^{z_k} = e^{6.5} + e^{-3.5} + e^{9.9} \approx 20595.54$$

The prediction for the different classes are:

$$\hat{y}_1 = \frac{e^{6.5}}{denom} \approx 0.0323$$

$$\hat{y}_2 = \frac{e^{-3.5}}{denom} \approx 1.467 * 10^{-6}$$

$$\hat{y}_3 = \frac{e^{9.9}}{denom} \approx 0.9677$$

The network will assign the image to the class of the squares because $\hat{y}_3$ has the highest prediction value and the network seems pretty sure about it because $\hat{y}_3$ has a very high value.

Task 4: Compute the loss for the prediction. Use the Cross Entropy Loss. Compute the backward pass through the fully connected layer and update the weights (you do not need to consider the bias). The image is from the squares class and use a learning rate of $\alpha = 0.1$.

Solution 4.1:

The Cross entropy Loss C is: $-\ln(\hat{y}_3) \approx 0{,}0328$

The gradient for the weights of the fully connected layer can be calculated using $\frac{\partial C(w)}{\partial w_{kj}} = -x_j(y_k - \hat{y}_k)$.

The gradient of the weights is:

$$\frac{\partial C(w)}{\partial w} = -\begin{bmatrix}5 & 11 & 13 & 7\end{bmatrix}\begin{bmatrix}0 - 0.0323 \\ 0 - 1.467 * 10^{-6} \\ 1 - 0.9677\end{bmatrix} \approx \begin{bmatrix}0.1615 & 0.3552 & 0.4198 & 0.2261 \\ 7.331 * 10^{-6} & 1.612 * 10^{-5} & 1.906 * 10^{-5} & 1.026 * 10^{-5} \\ -0.1615 & -0.3553 & -0.4199 & -0.2261\end{bmatrix}$$

The updated weights W' can be calculated as follows:

$$W' = W - \alpha\frac{\partial C(w)}{\partial w} = \begin{bmatrix}0.5 & 0.2 & -0.1 & 0.3 \\ -0.4 & -0.2 & 0 & 0.1 \\ 0 & 0.2 & 0.6 & -0.3\end{bmatrix} \text{-0.1*} \begin{bmatrix}0.1615 & 0.3552 & 0.4198 & 0.2261 \\ 7.331 * 10^{-6} & 1.612 * 10^{-5} & 1.906 * 10^{-5} & 1.026 * 10^{-5} \\ -0.1615 & -0.3553 & -0.4199 & -0.2261\end{bmatrix} \approx$$

$$\begin{bmatrix}0.4839 & 0.1645 & -0.142 & 0.2774 \\ -0.4000 & -0.2000 & -1.906 * 10^{-6} & 0.100 \\ 1.6148 * 10^{-2} & 0.2355 & 0.642 & -0.2774\end{bmatrix}$$

Task 5: Another dataset consists of only 3x3 images with the same classes. For this network a different architecture is used. It is depicted in the table below. The filter in the convolutional layer has the size 2x2 with no padding and a stride of 1. The image *X* was forwarded into the network and the filter *F* was currently used (see below). The gradient flowing back from the fully connected layer to the convolution layer is also given (see below). Update the values of the filter using the learning rate $\alpha = 0.1$ and the matrix convolution operation. Additionally compute the gradient that would flow back from this convolutional layer to a previous one using the transposed matrix convolution.

$X =$ | 0 | 4 | 0 |
|---|---|---|
| 4 | 0 | 3 |
| 1 | 4 | 0 |

, $F =$ | 0 | 1 |
|---|---|
| -1 | 1 |

, $\frac{\partial L}{\partial O} =$ | -0.1 | -0.3 |
|---|---|
| 0.1 | 0.2 |

Layer	Layer Type	Number of Filters/Number of hidden units	Activation Function
1	Conv2D	1	-
Flatten	-	-	-
2	Fully-Connected	3	Softmax

Solution 5.1: The gradients $\frac{\partial L}{\partial F}$ and $\frac{\partial L}{\partial X}$ need to be calculated. The basis for the calculation are given by the formulas $\frac{\partial L}{\partial F_i} = \sum_{k=1}^{4} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$ and $\frac{\partial L}{\partial X_i} = \sum_{k=1}^{4} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial X_i}$. The structure of the used matrices is adapted from the lecture slides (See second last slide for further information).

Next slide

The matrix $M = \begin{bmatrix} -0.1 & -0.3 & 0 & 0.1 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & -0.1 & -0.3 & 0 & 0.1 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.1 & -0.3 & 0 & 0.1 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & -0.1 & -0.3 & 0 & 0.1 & 0.2 \end{bmatrix}$, which consists of the loss gradient from the next layer and the matrix $\hat{X}^T = [0 \quad 4 \quad 0 \quad 4 \quad 0 \quad 3 \quad 1 \quad 4 \quad 0]$, which consists of the flattened input X are used for the matrix convolution for the computation of the filter gradient.

Solution 5.2: Now the matrix convolution can be performed

$M * \hat{X} = \begin{bmatrix} -0.8 \\ 0.2 \\ 0.5 \\ -0.5 \end{bmatrix}$. Thus the filter gradient is $\frac{\partial L}{\partial F} =$ | -0.8 | 0.2 |
|---|---|
| 0.5 | -0.5 |

. The updated Filter $F' = F - \alpha \frac{\partial L}{\partial F}$ can be finally calculated and is | 0 | 1 |
|---|---|
| -1 | 1 |

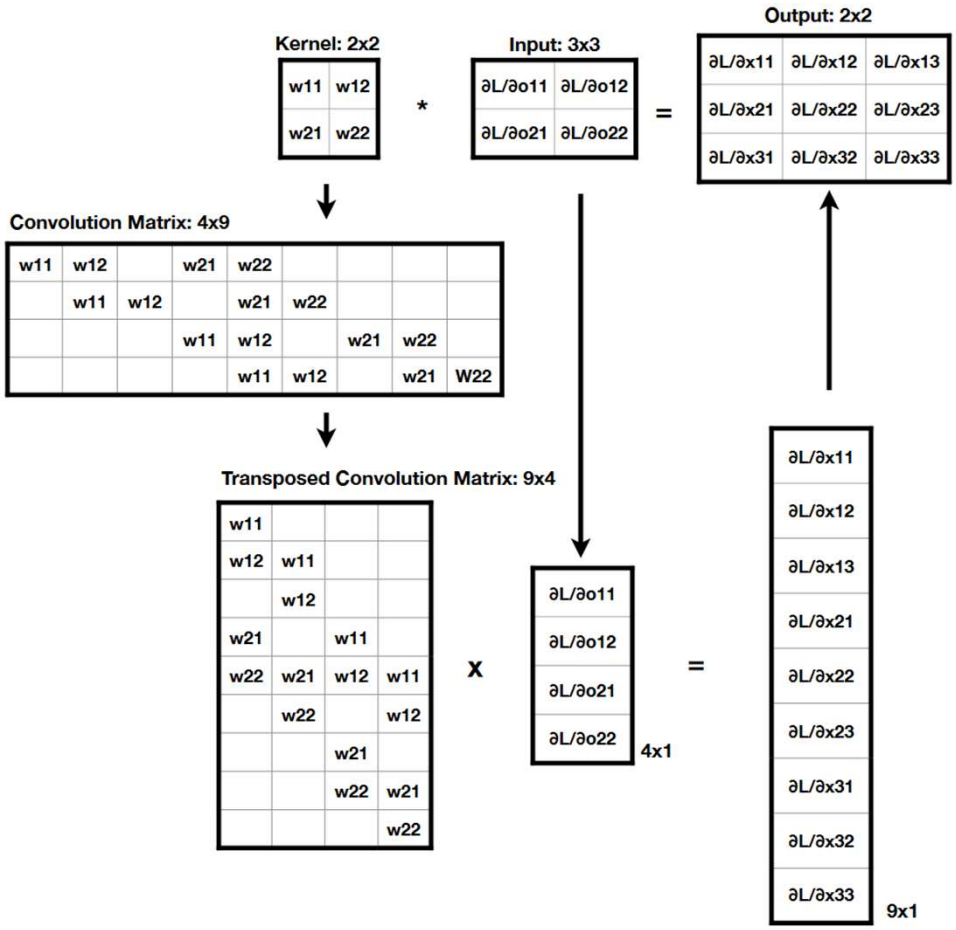
$-0.1*$ | -0.8 | 0.2 |
|---|---|
| 0.5 | -0.5 |

$=$ | 0.08 | 0.98 |
|---|---|
| -1.05 | 1.05 |

The next task is to compute the transposed matrix convolution for the gradient that would flow to the previous layer. The filter F is converted into the transposed convolution matrix $M_F = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and the gradient from the next layer is flattened into $\mathrm{G} = \begin{bmatrix} -0.1 \\ -0.3 \\ 0.1 \\ 0.2 \end{bmatrix}$. The gradient for the previous layer can be calculated $\frac{\partial L}{\partial X} = M_F * G = \begin{bmatrix} 0 \\ -0.1 \\ -0.3 \\ 0.1 \\ 0.3 \\ -0.1 \\ -0.1 \\ -0.1 \\ 0.2 \end{bmatrix}$. The gradient can be rewritten into the image format:

0	-0.1	-0.3
0.1	0.3	-0.1
-0.1	-0.1	0.2

Additional Information Task 5



The image on the left side is from the slide 49 of the presentation TDT4265-2022_06_Conv-Viz. It describes how the transposed convolution is conducted to compute the gradient that would flow backwards to a previous layer. The transposed convolution in Task 5 was conducted based on this matrix in the image. The regular matrix convolution for the gradient of the filter was also adapted from this presentation. It would have also been possible to change the dimensions of the filter and the input. But this would have required to deduce the new matrices. The deduction would have taken to much time in this giving task which should only take a few minutes. The deduction would have been conducted using the formulas which were mentioned in solution 5.1.

Inspiration

- Inspiration was found in the lecture slides of the course Computer Vision and Deep Learning and from the following blog post
- How does Backpropagation work in a CNN? | Medium