

# AlexNet

Leon Friedrich, INFM



**Elektrotechnik, Medizintechnik  
und Informatik**

# Agenda

---

- Convolution
- AlexNet
  - Why AlexNet?
  - Architecture
- Practical Part
  - Visualization of learned Kernels
  - Visualization of Convolution

# Convolution I

- Mathematical operation:  $*$
- 2D convolution in image processing:
  - Image  $I[u, v]$
  - Kernel  $K$  with size  $[2a + 1, 2b + 1]$
  - Output  $O[u, v]$

	-1	0	1
-1	1	2	3
0	4	5	6
1	7	8	9

$$O[u, v] = I[u, v] * K = \sum_{s=-a}^a \sum_{t=-b}^b I[u - s, v - t] K[s, t]$$

# Convolution II

$$O[u, v] = I[u, v] * K = \sum_{s=-a}^a \sum_{t=-b}^b I[u-s, v-t] K[s, t]$$

1	2	3
4	5	6
7	8	9

$K$

flip

9	8	7
6	5	4
3	2	1

9	8	7	0	0	0	0
6	5	4	0	0	0	0
3	2	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

# Convolution III

9	8	7	0	0	0	0
6	5	4	0	0	0	0
3	2	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

$$O[1,1] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 0$$

	0					

$O[u, v]$



# Convolution III

0	9	8	7	0	0	0
0	6	5	4	0	0	0
0	3	2	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

$$O[1,2] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 0$$

	0	0				

$O[u, v]$

# Convolution III

0	0	9	8	7	0	0
0	0	6	5	4	0	0
0	0	3	2	1	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

$$O[1,3] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 0$$

	0	0	0			

$O[u, v]$

# Convolution III

0	0	0	9	8	7	0
0	0	0	6	5	4	0
0	0	0	3	2	1	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

$$O[1,4] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 0$$

	0	0	0	0		

$O[u, v]$



# Convolution III

0	0	0	0	9	8	7
0	0	0	0	6	5	4
0	0	0	0	3	2	1
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

$$O[1,5] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 0$$

	0	0	0	0	0	

$O[u, v]$

# Convolution III

0	0	0	0	0	0	0
9	8	7	0	0	0	0
6	5	4	0	0	0	0
3	2	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$I[u, v]$

$$O[2,1] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 0$$

	0	0	0	0	0	
	0					

$O[u, v]$

# Convolution III

0	0	0	0	0	0	0
0	9	8	7	0	0	0
0	6	5	4	0	0	0
0	3	2	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

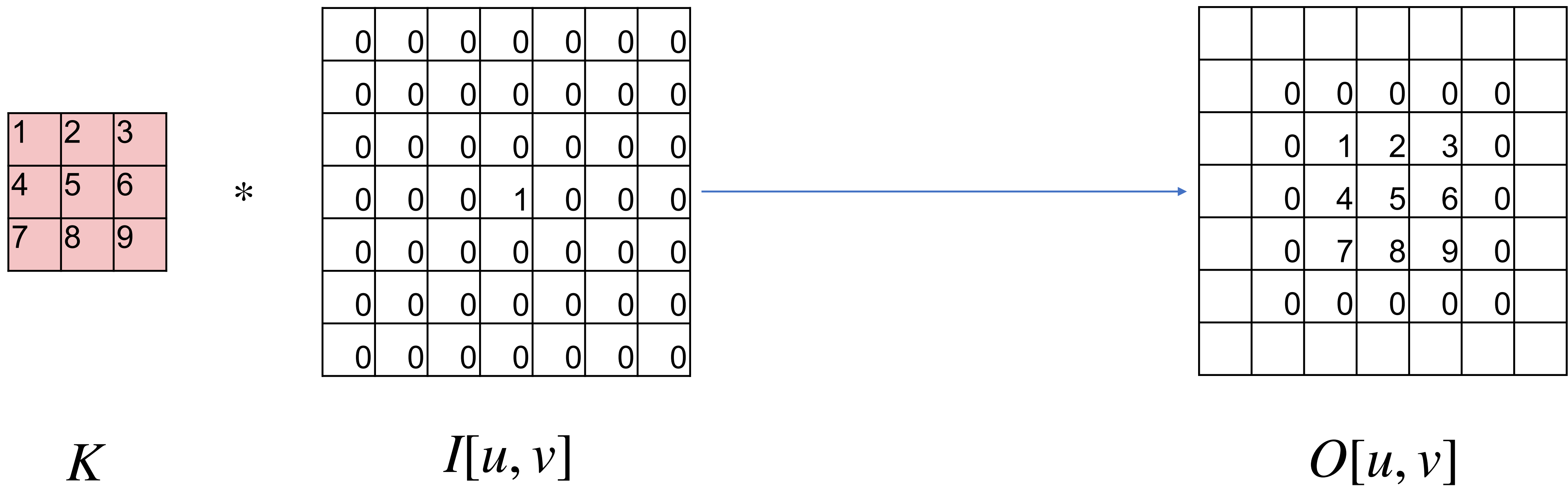
$I[u, v]$

$$O[2,2] = 9 \cdot 0 + 8 \cdot 0 + 7 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 4 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 1 = 1$$

	0	0	0	0	0	
	0	1				

$O[u, v]$

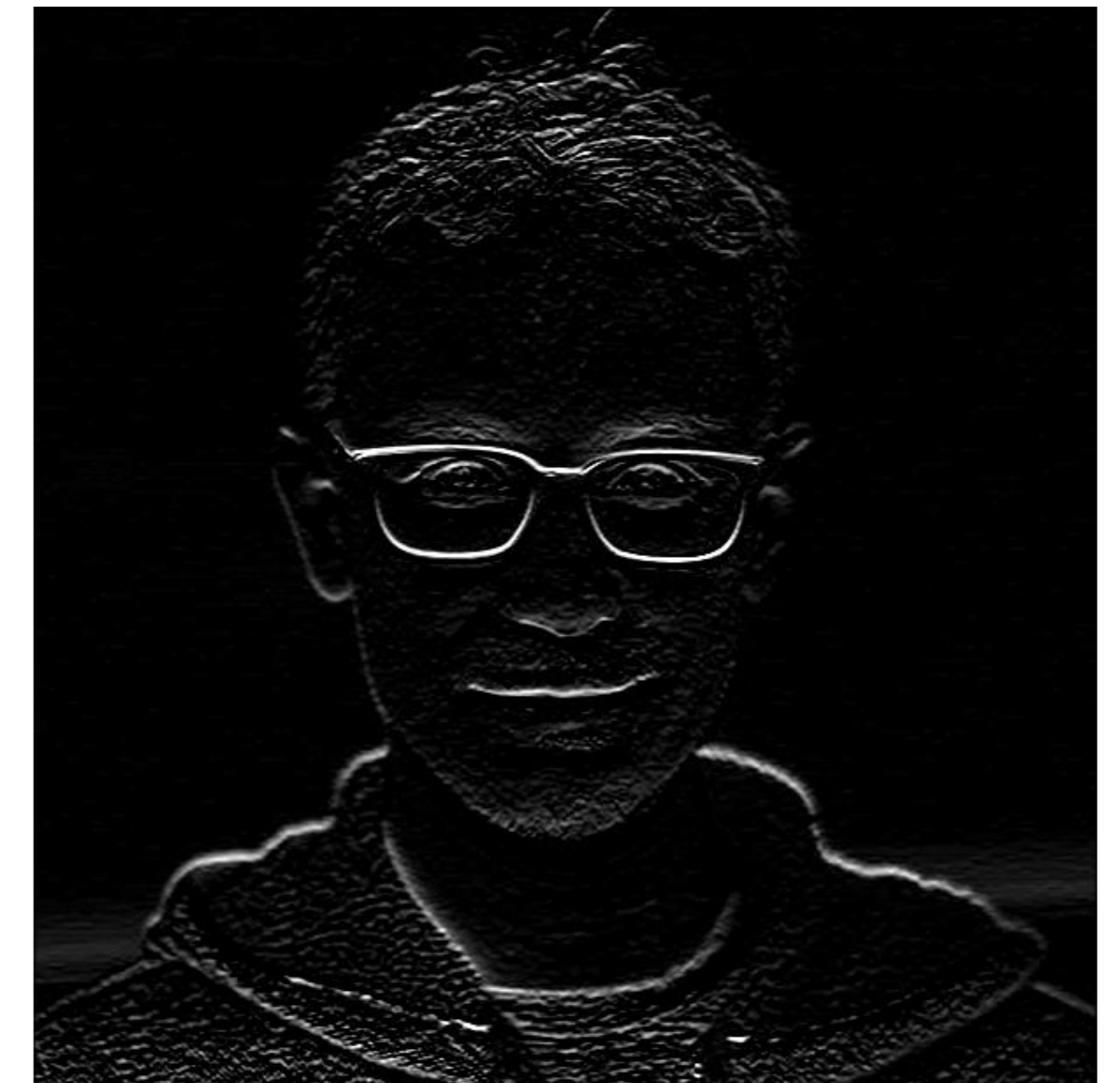
# Convolution IV



# Convolution V

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

\*



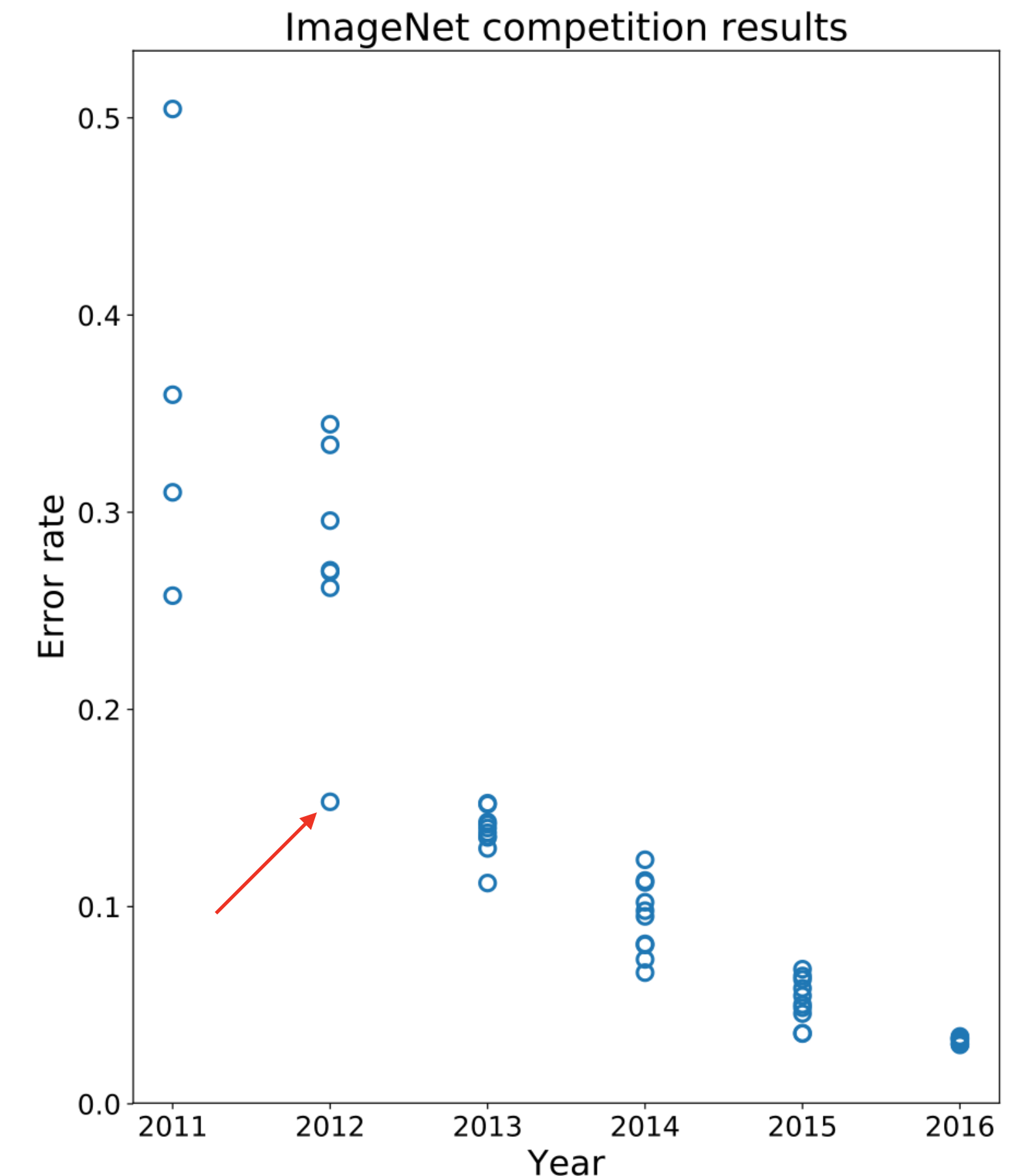
$K$

$I[u, v]$

$O[u, v]$

# AlexNet - Why AlexNet?

- Alex Krizhevsky et al. (2012)
- Winner ILSVRC-2012
  - Top-5 test error rate: 15.3%
- Key to success: Faster training



By Gkrusze - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=69750373>

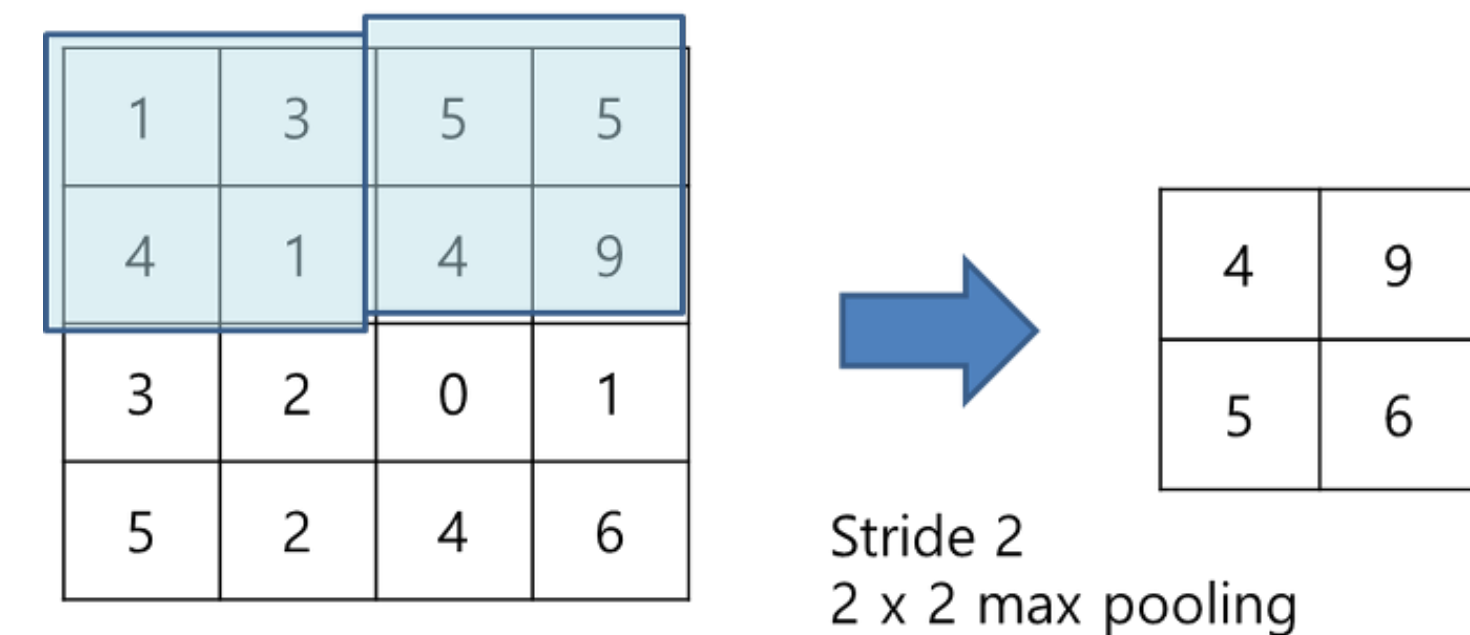




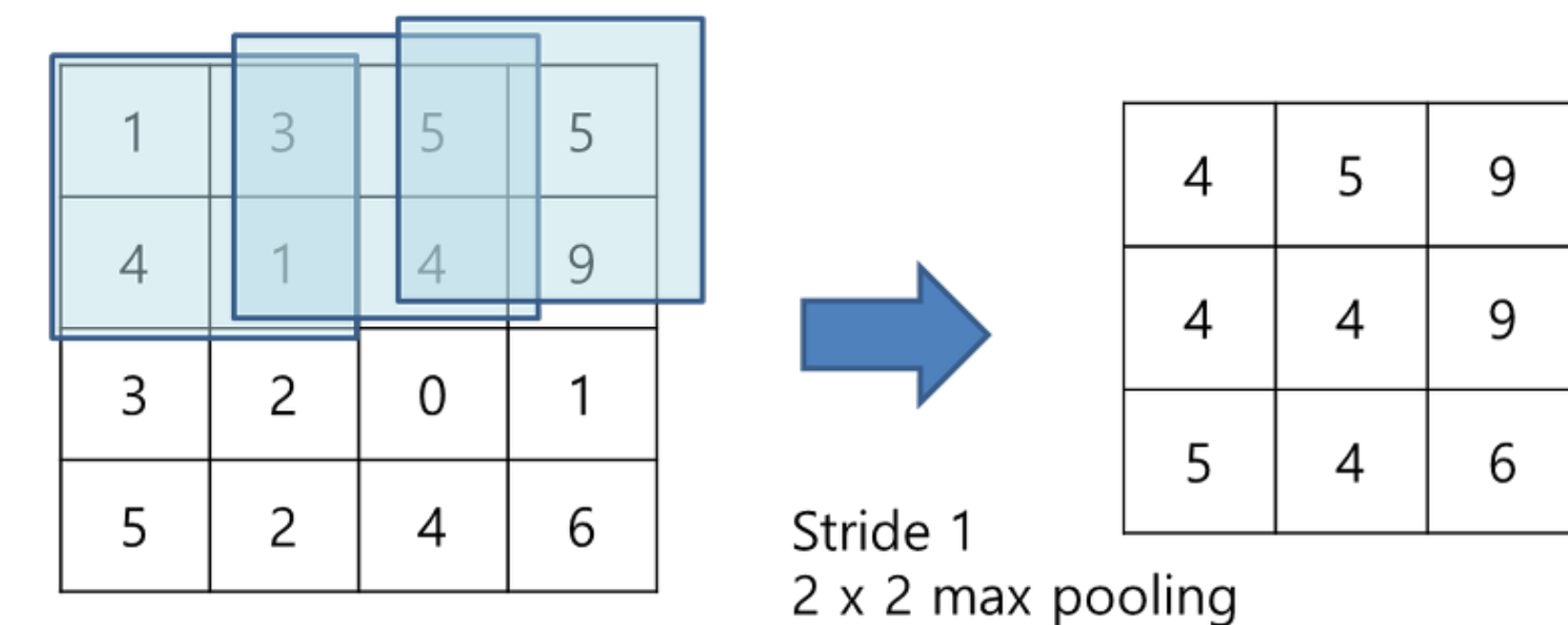
# AlexNet - Reduce Overfitting

- 60 Mio. parameters
- Data Augmentation
- Dropout
- Overlapping Pooling

Non-overlapping pooling



Overlapping pooling



Source: <https://bskyvision.com/421>

# Practical Part - AlexNet

## Manually implemented:

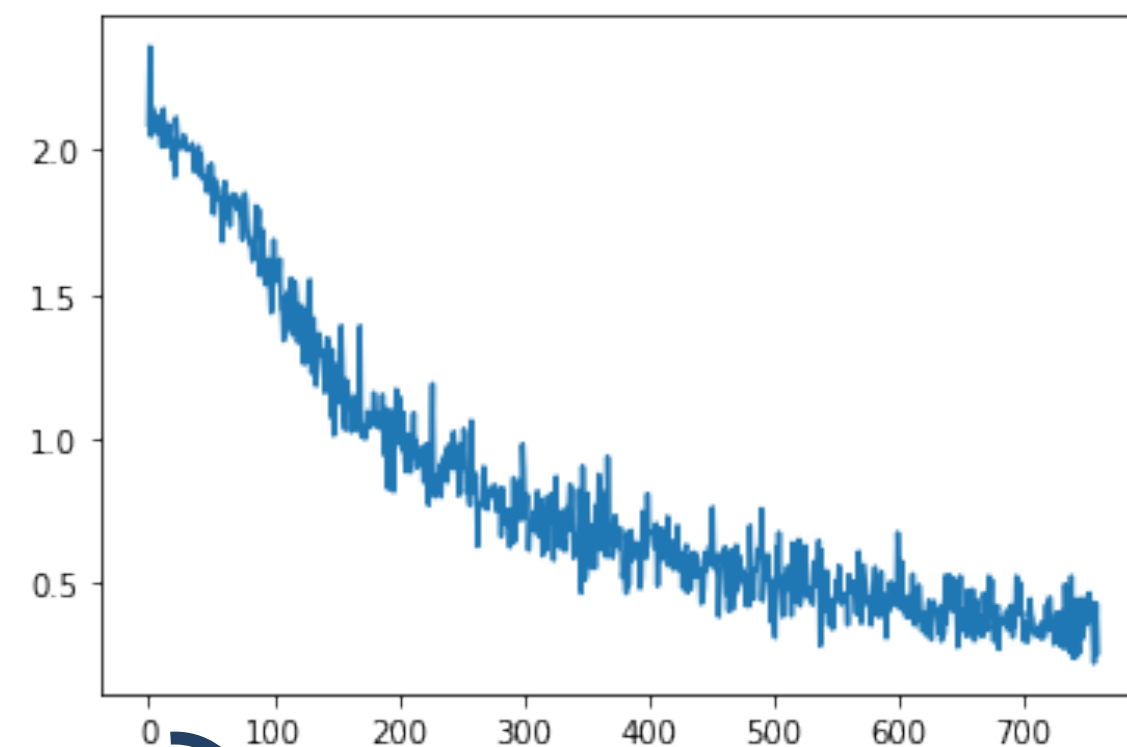
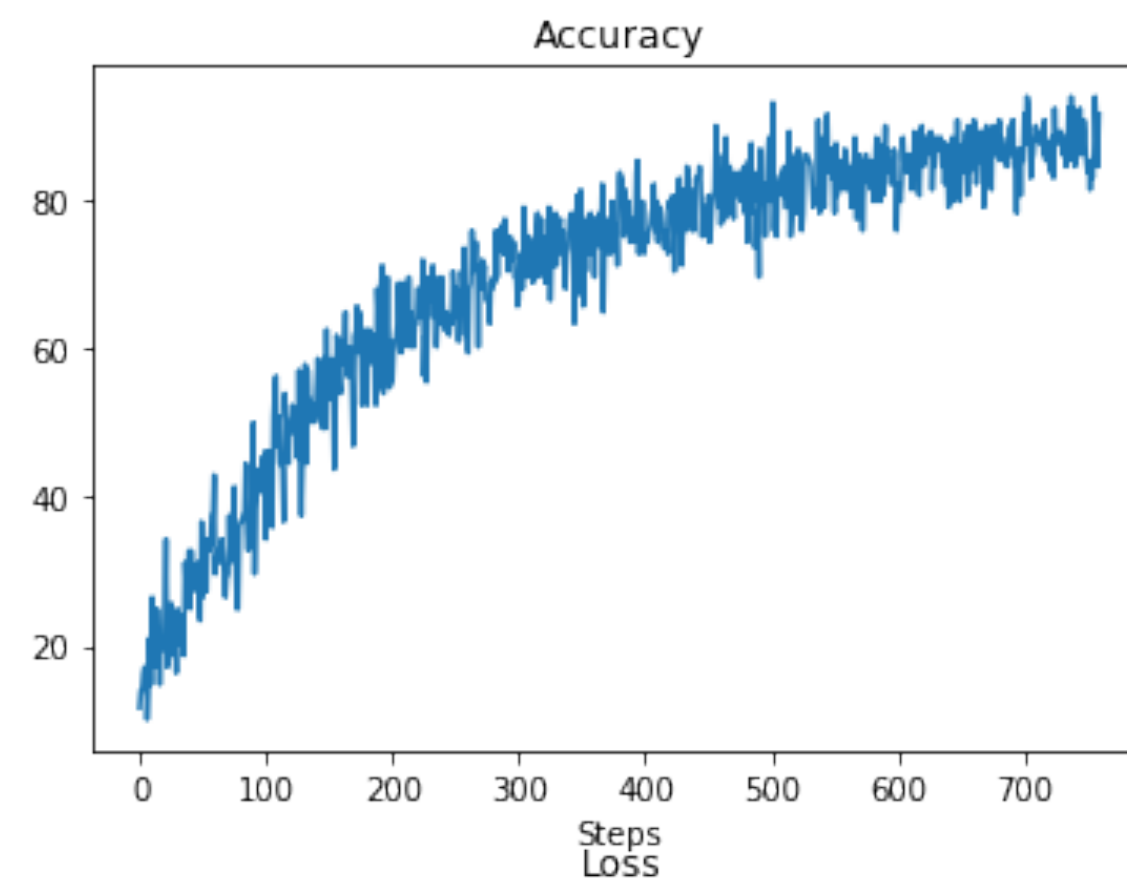
```
AlexNet(  
  (features): Sequential(  
    (0): Conv2d(3, 96, kernel_size=(11, 11), stride=(4, 4))  
    (1): ReLU()  
    (2): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=2)  
    (3): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (4): Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (5): ReLU()  
    (6): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=2)  
    (7): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (8): Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (9): ReLU()  
    (10): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (11): ReLU()  
    (12): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (13): ReLU()  
    (14): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (classifier): Sequential(  
    (0): Dropout(p=0.5, inplace=True)  
    (1): Linear(in_features=256, out_features=4096, bias=True)  
    (2): ReLU()  
    (3): Dropout(p=0.5, inplace=True)  
    (4): Linear(in_features=4096, out_features=4096, bias=True)  
    (5): ReLU()  
    (6): Linear(in_features=4096, out_features=8, bias=True)  
  )  
)
```

## Pytorch version:

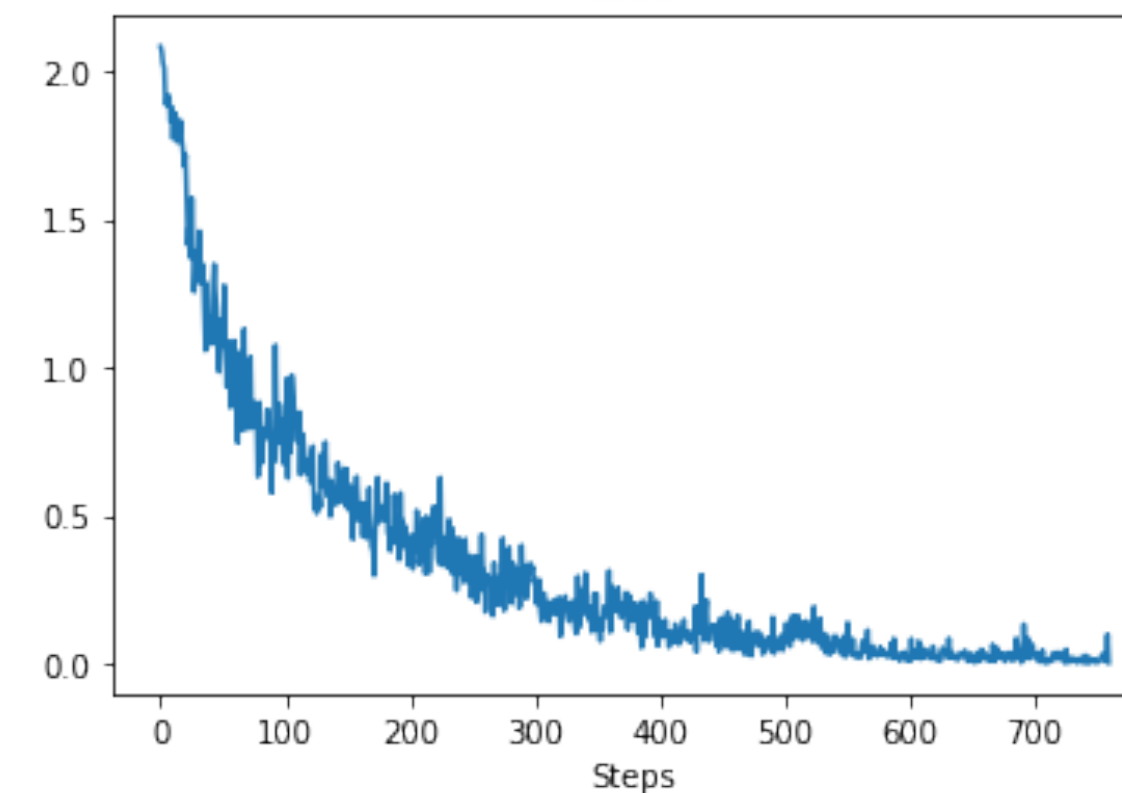
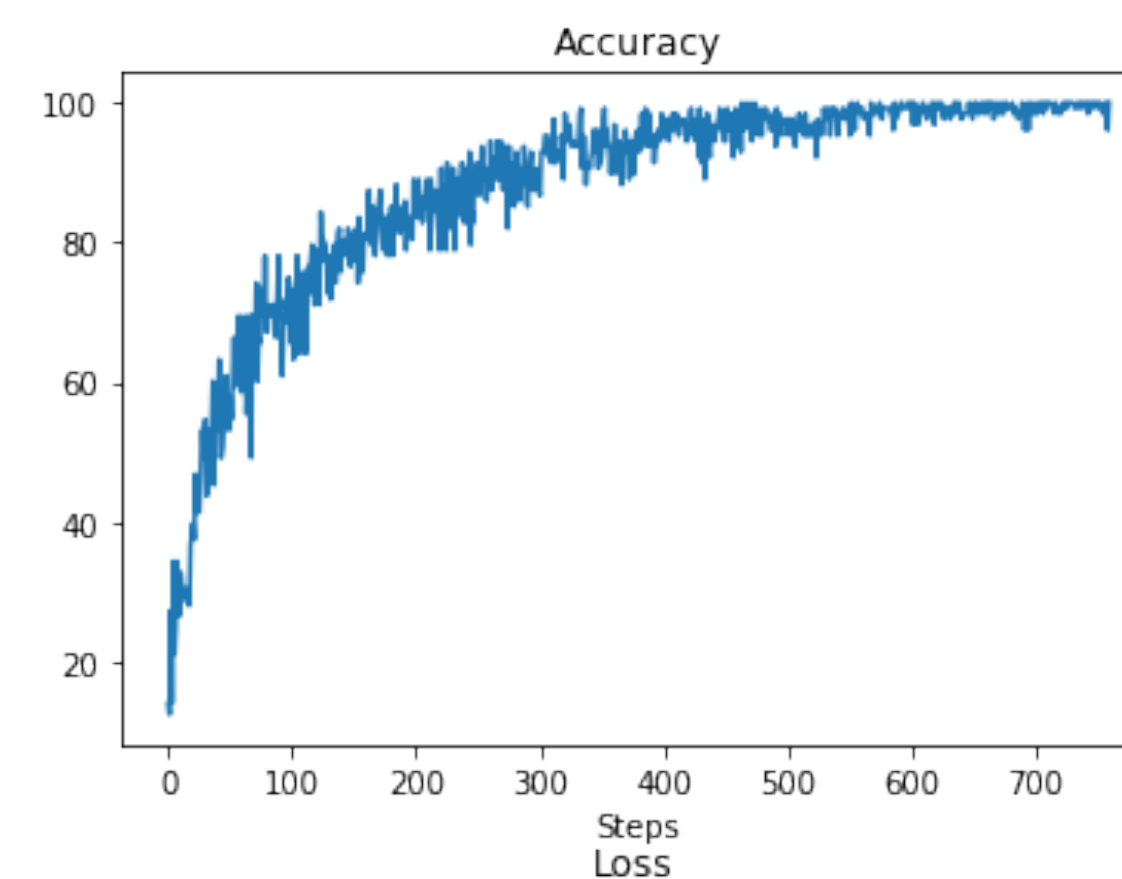
```
AlexNet(  
  (features): Sequential(  
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))  
    (1): ReLU(inplace=True)  
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (4): ReLU(inplace=True)  
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (7): ReLU(inplace=True)  
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (9): ReLU(inplace=True)  
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (11): ReLU(inplace=True)  
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))  
  (classifier): Sequential(  
    (0): Dropout(p=0.5, inplace=False)  
    (1): Linear(in_features=9216, out_features=4096, bias=True)  
    (2): ReLU(inplace=True)  
    (3): Dropout(p=0.5, inplace=False)  
    (4): Linear(in_features=4096, out_features=4096, bias=True)  
    (5): ReLU(inplace=True)  
    (6): Linear(in_features=4096, out_features=1000, bias=True)  
  )  
)
```

# Accuracy and Loss

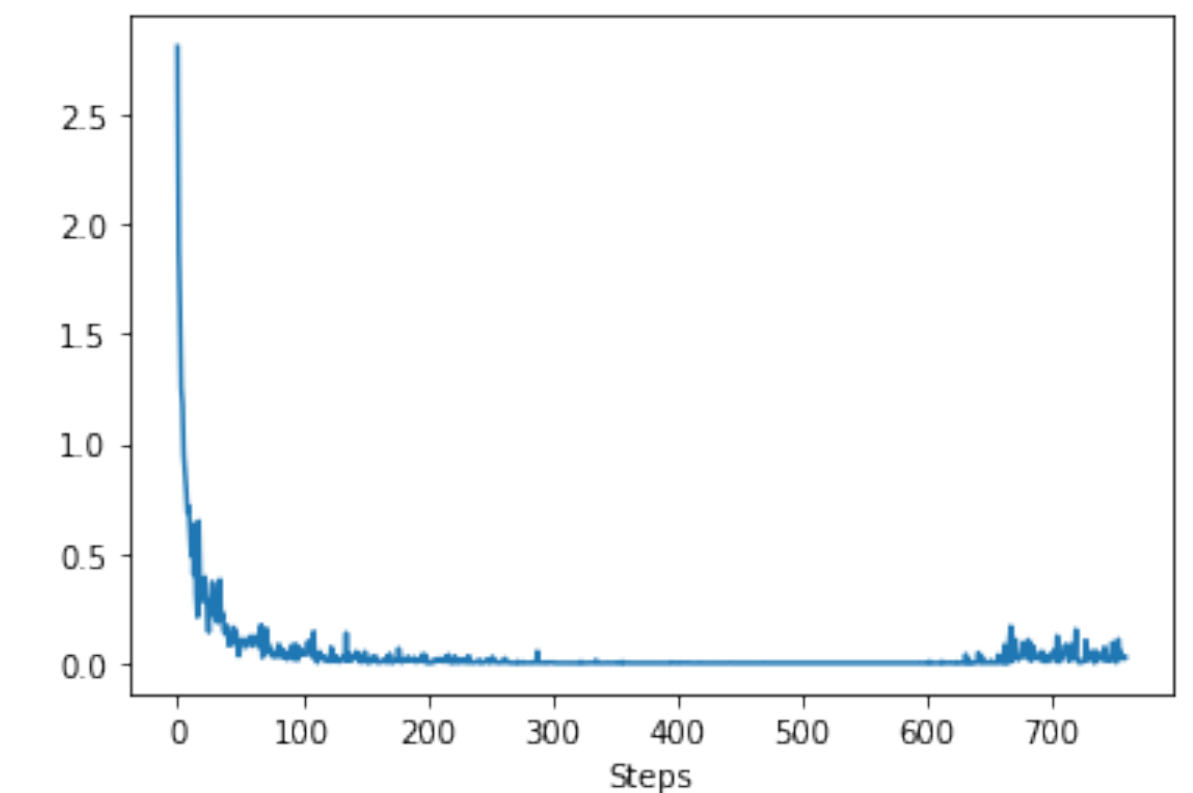
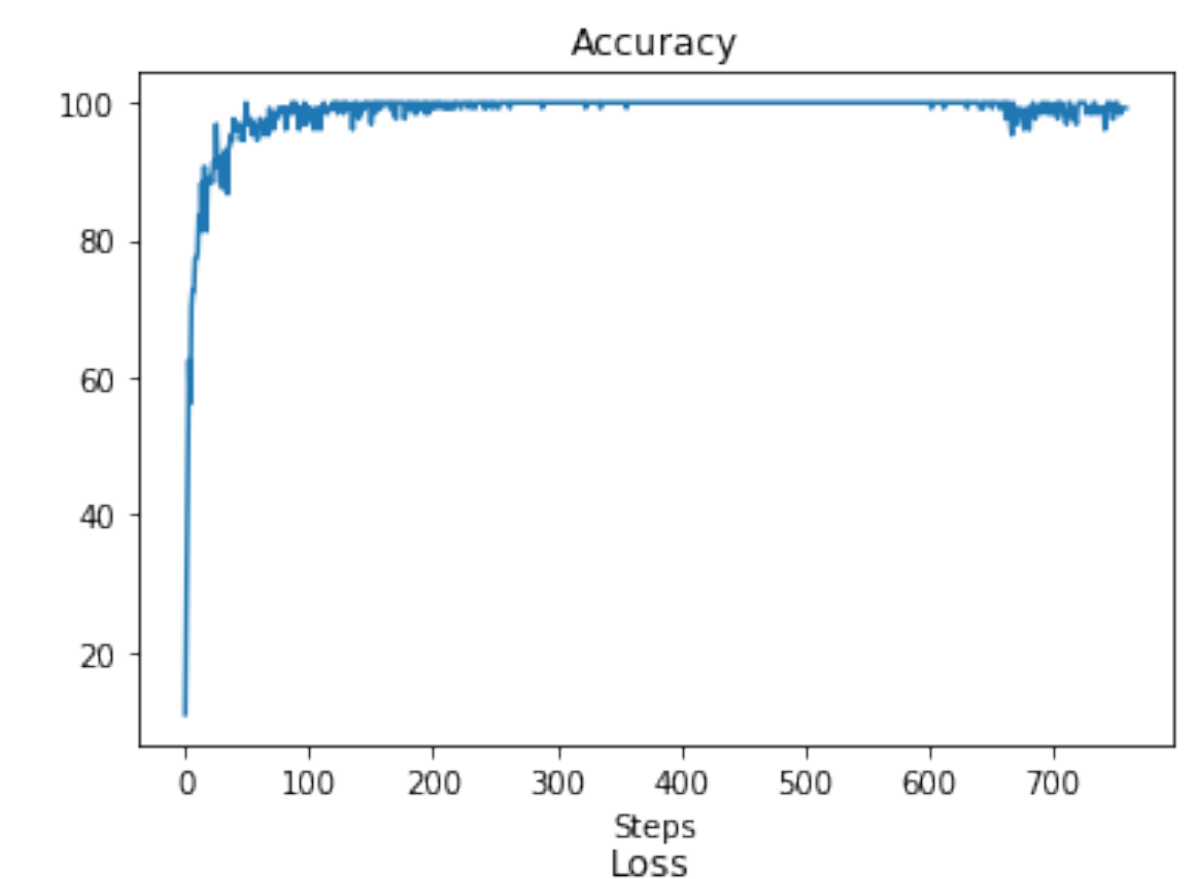
## Manual Implementation



## Pytorch not pretrained

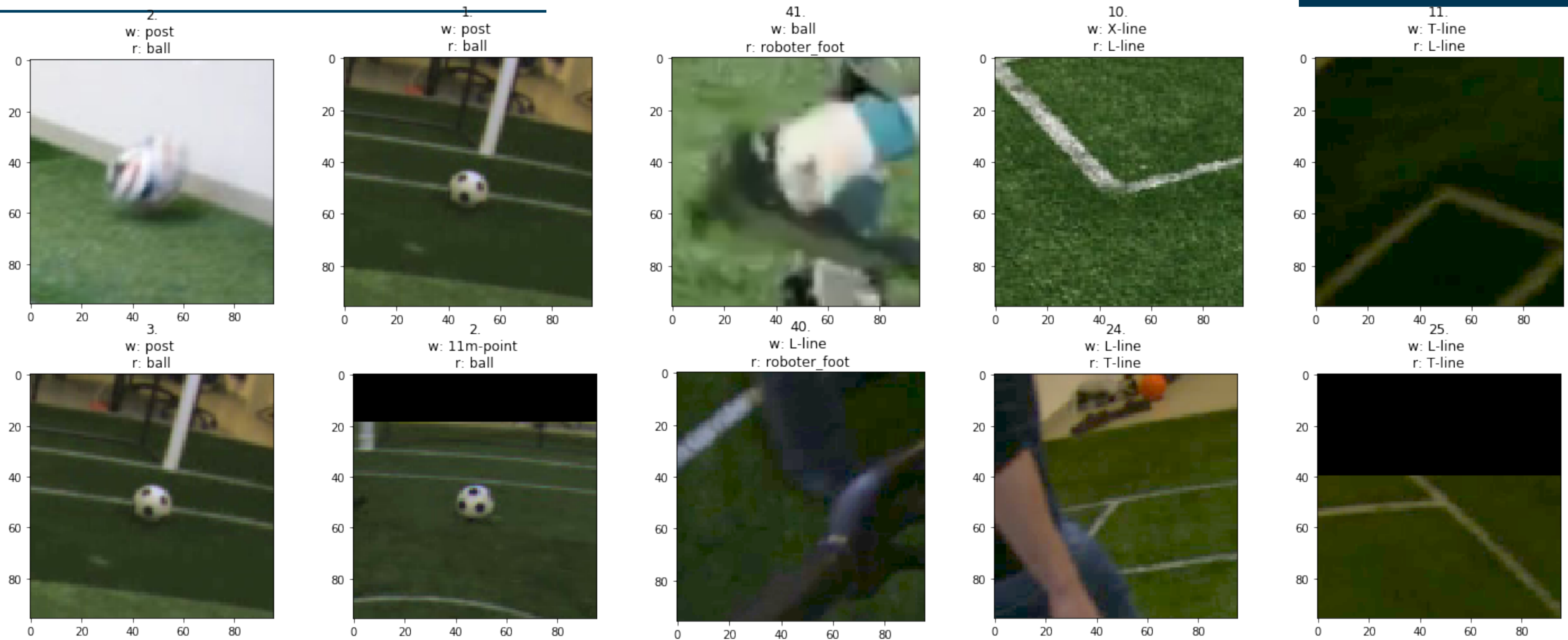


## Pytorch pretrained





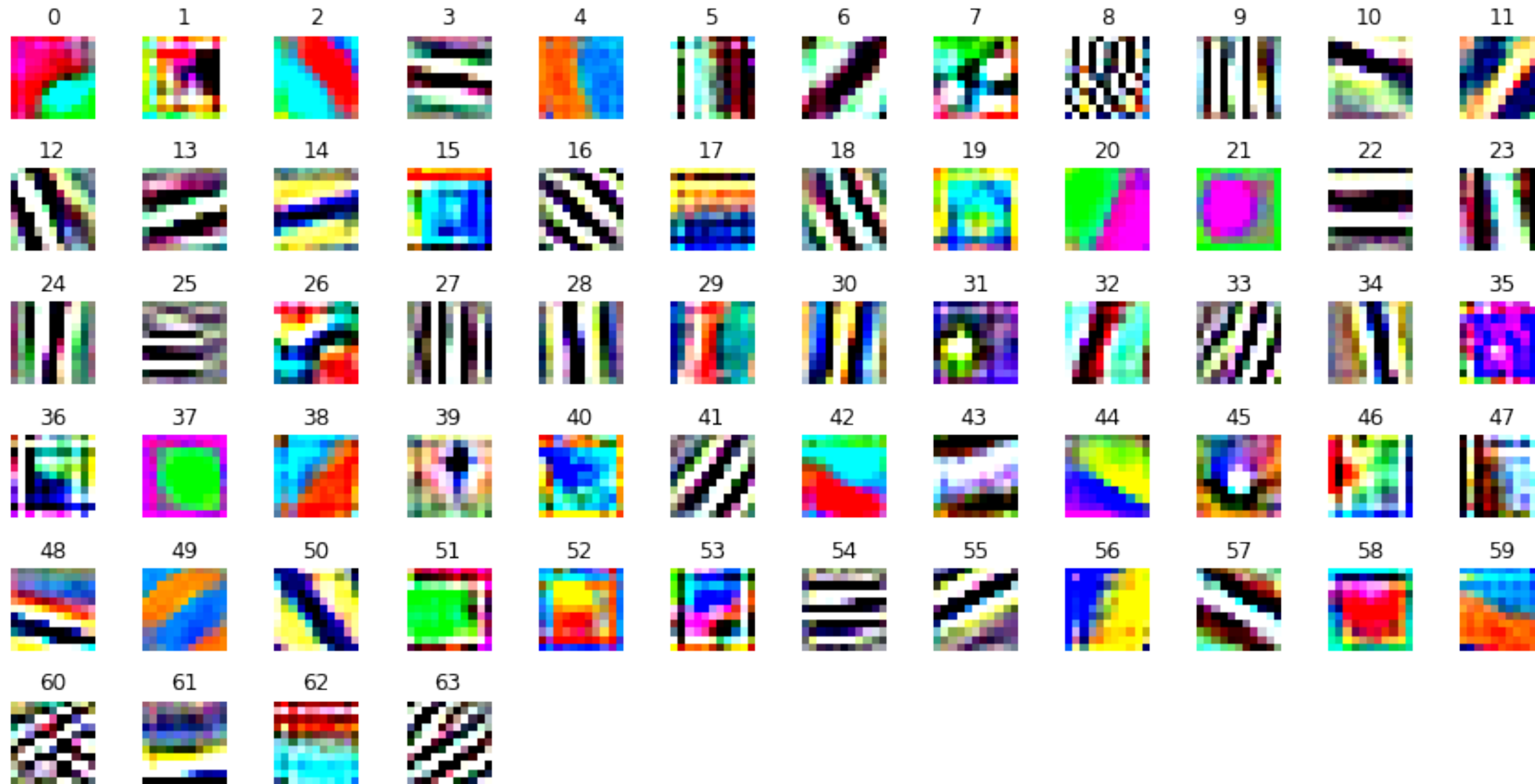
# Wrongly Classified Images



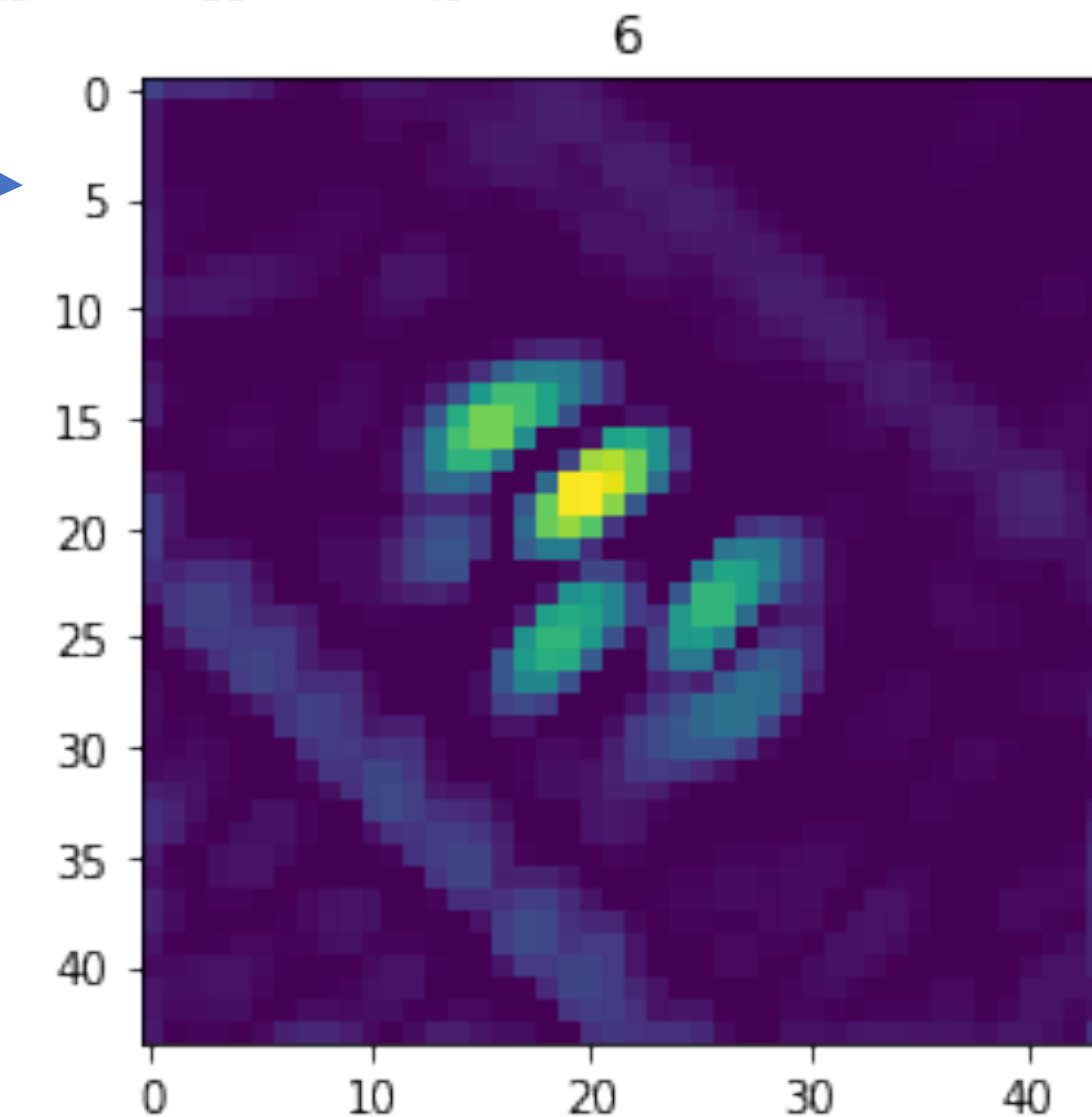
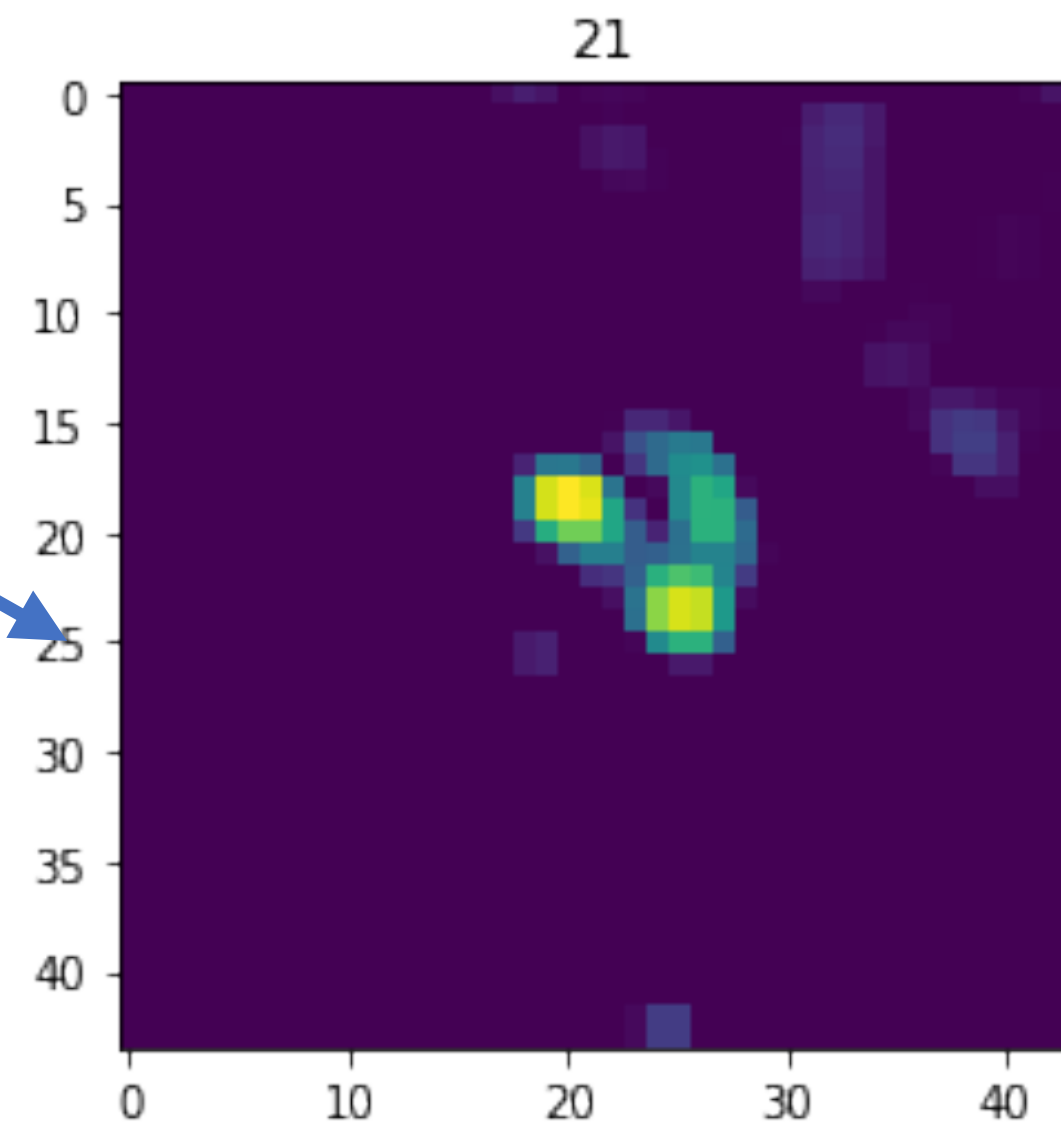
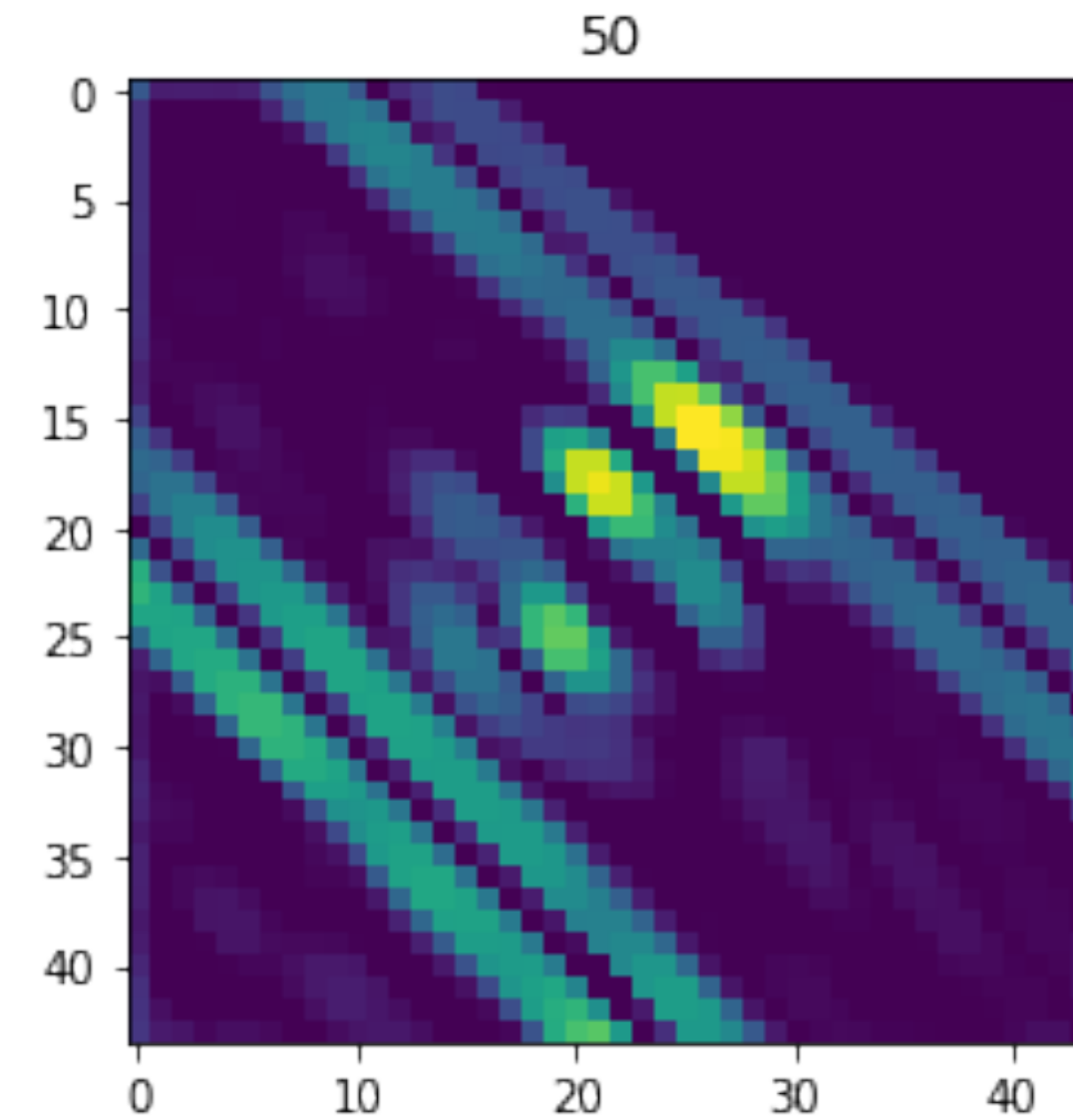
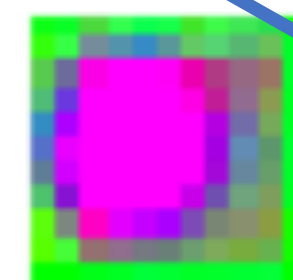
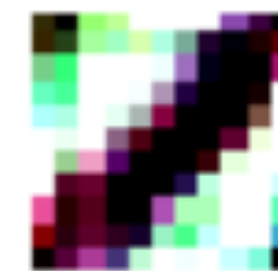
# Learned Kernels - Not Pretrained



# Learned Kernels - Pretrained



# Visualization of Convolution





# Thank you!

# Sources

---

## Paper:

<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

## Gitlab Repo with slides, notebook and paper:

<https://github.com/MumenRaida/SeminarDeepLearningAlexNet.git>

## Further sources:

<https://en.wikipedia.org/wiki/AlexNet>

<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

[https://machinelearningguru.com/computer\\_vision/basics/convolution/image\\_convolution\\_1.html](https://machinelearningguru.com/computer_vision/basics/convolution/image_convolution_1.html)

[https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))