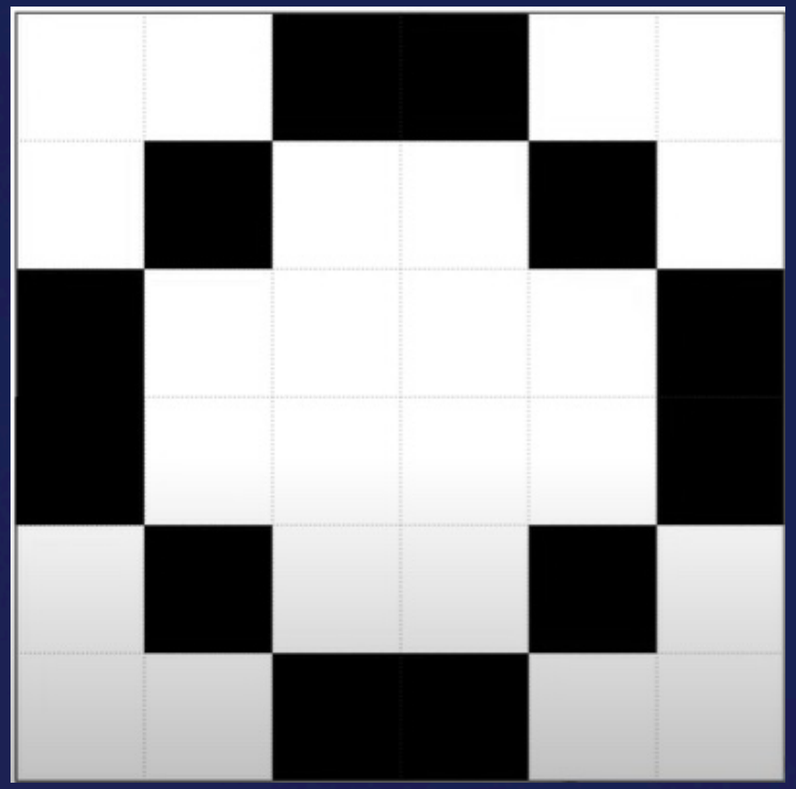
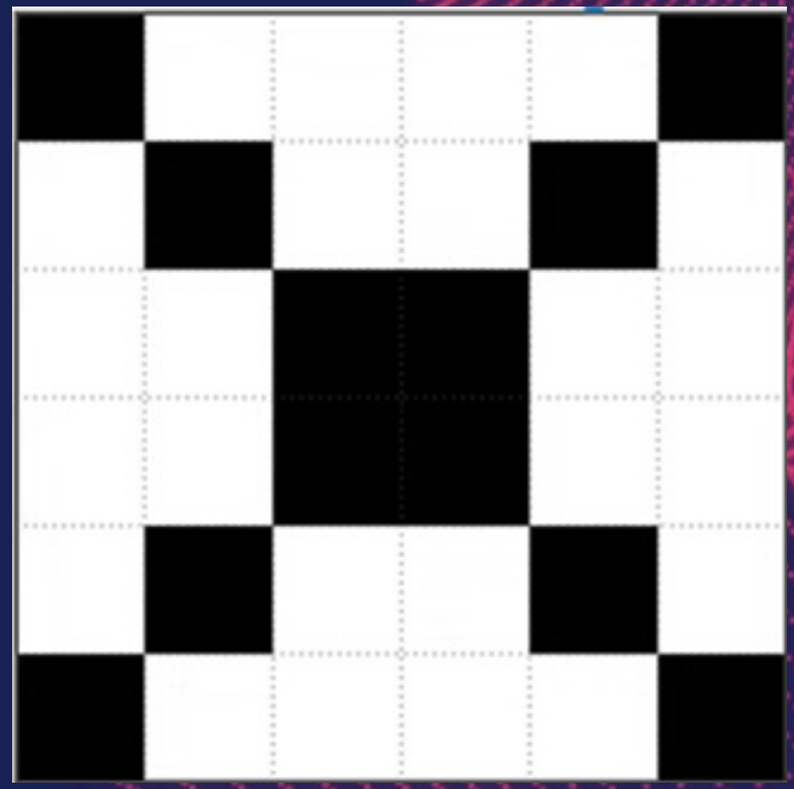
CONVOLUTION
NEURAL NETWORK

WHY DO WE NEED CNN?



6X6



6X6

36 NODES (INPUT LAYER)

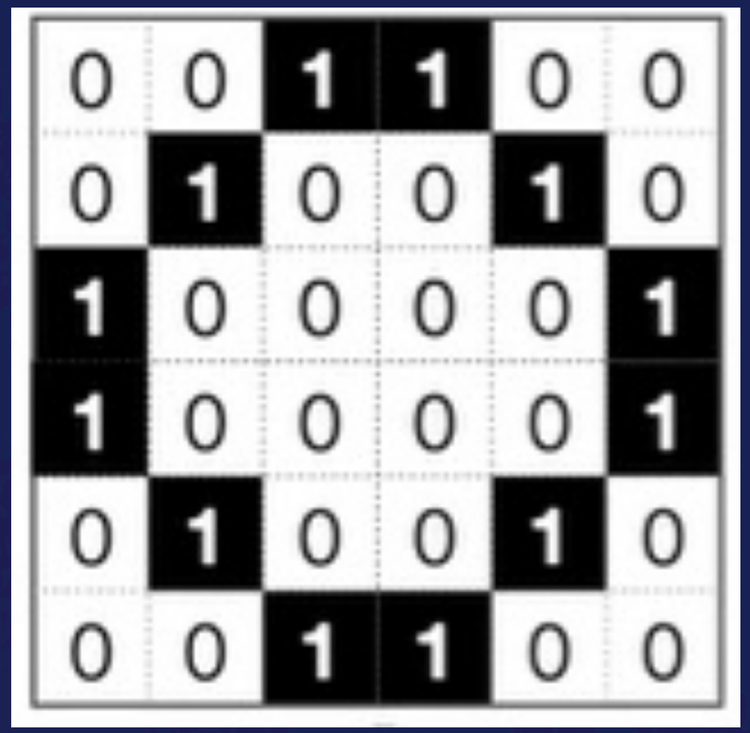
36 WEIGHTS

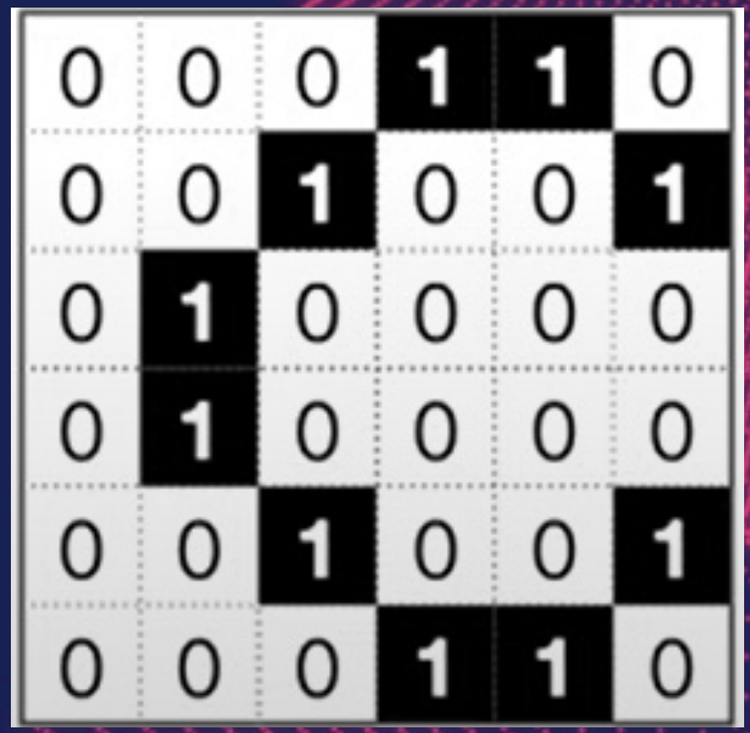
(PER NODE IN HIDDEN LAYER)

Each of the input node will be connected to each node of the hidden layer. If we assume a hidden layer of 256 nodes, we need 256*36 = 9216 weights !

Lets assume we are using the MNIST database , we have 256 input nodes. So if we want to make a hidden layer of 36 nodes , we once again need 9216 weights !

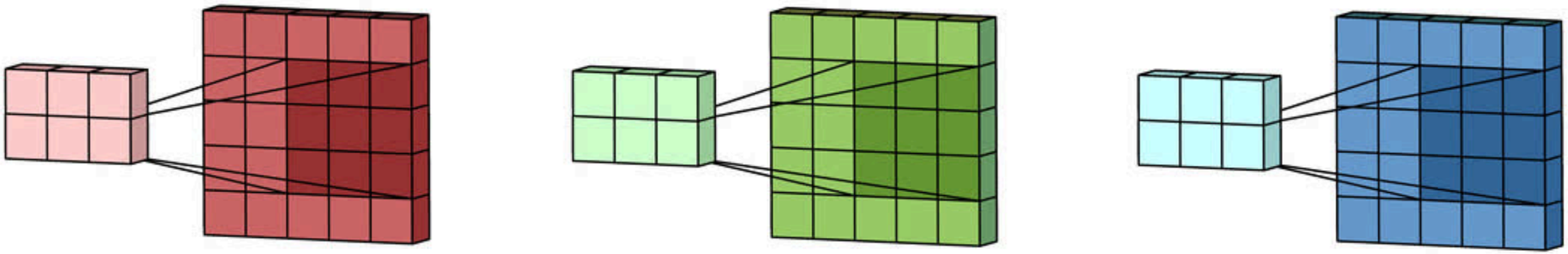
WHY DO WE NEED CNN?





AI CLUB

IMPORTANT TERMS



CONVOLUTION:

- Convolution between two functions in mathematics produces a third function expressing how the shape of one function is modified by other

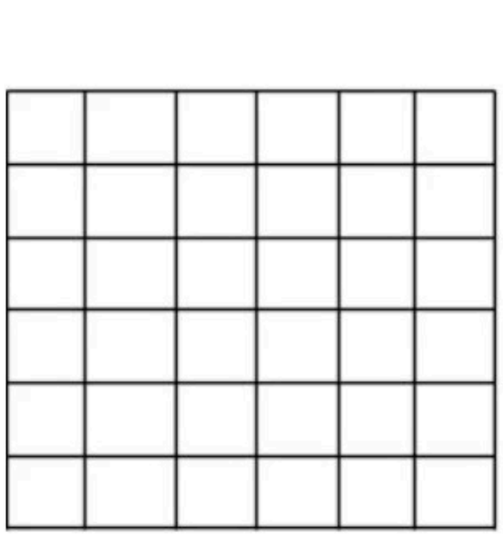
STRIDE:

- Stride defines by what step does to kernel move, for example stride of 1 makes kernel slide by one row/column at a time and stride of 2 moves kernel by 2 rows/columns.

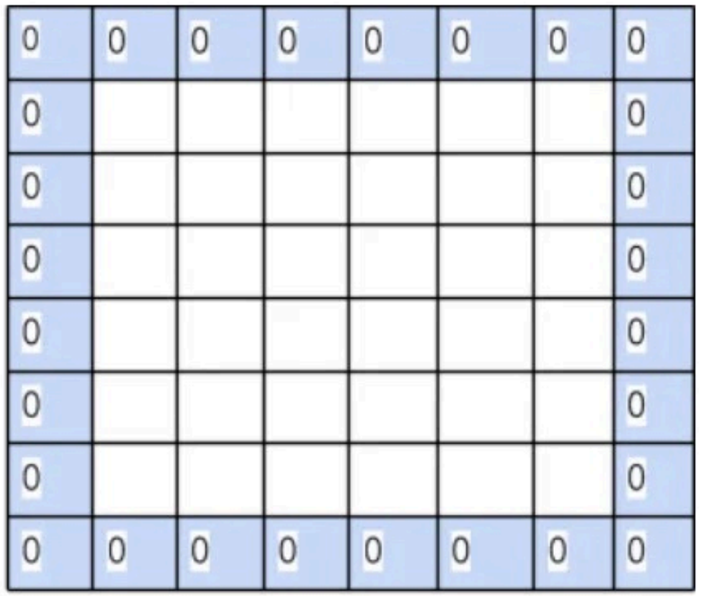
KERNEL:

- A kernel is a small 2D matrix whose contents are based upon the operations to be performed.

PADDING



6x6 image



6x6 image with 1 layer of zero padding

SOME KERNELS

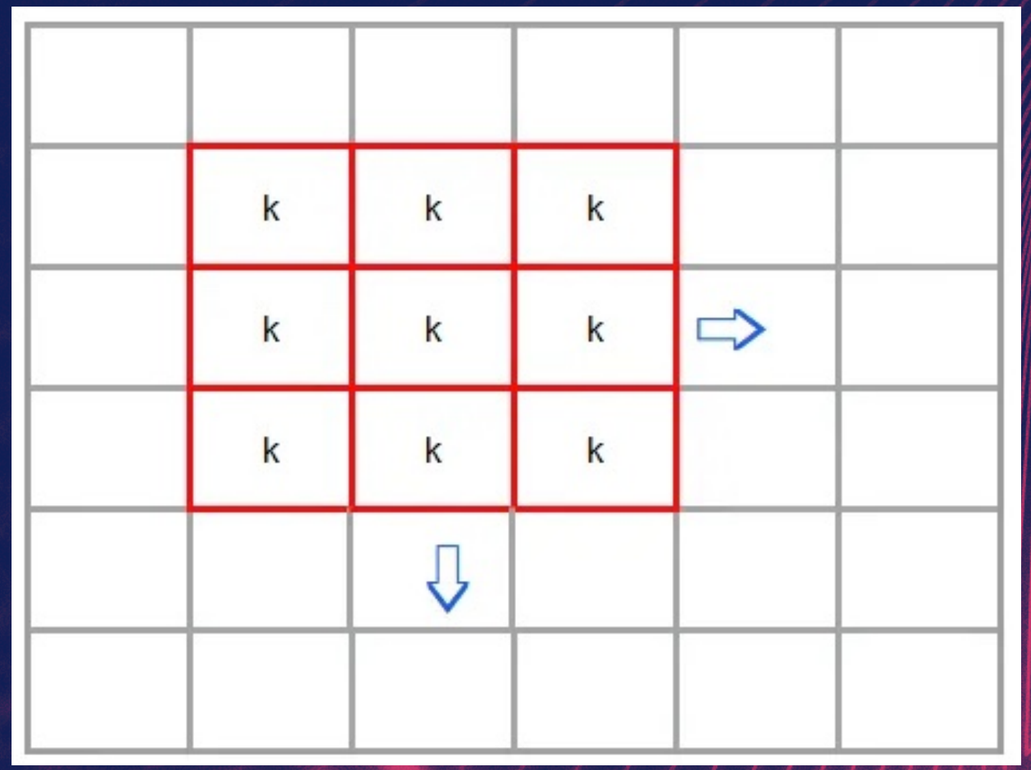
Original	*Gaussian Blur*	*Sharpen*	*Edge Detection*
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

SOME KERNEL OPERATIONS

slido.com

#3316354

KENREL MOVEMENT



EXAMPLES

Input Matrix

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

Kernel

0	-1	0
-1	5	-1
0	-1	0

⇒

Result

-45	12
22	10

CONVOLUTIONAL LAYERS

There is a simple formula which relates the dimensions of the output Activation Map, as a function of the input channel dimension, stride, padding and kernel size.

It is given as follows :

$$n_{out} = \frac{n_{in} + 2p - k}{s} + 1$$

Where:

N out = Activation Map's dimension

N in = input channel dimension

p= padding

s = stride

A CNN CONSISTS OF:

1)Convolutional Layer

2)Pooling Layer

3)Fully Connected Layer

CONVOLUTIONAL LAYER

1) Apply the kernel
2)Add bias
3)Pass the output through an activation layer.

CONVOLUTIONAL LAYER

Input Image

0	0	**1**	**1**	0	0
0	**1**	0	0	**1**	0
1	0	0	0	0	**1**
1	0	0	0	0	**1**
0	**1**	0	0	**1**	0
0	0	**1**	**1**	0	0

Filter (aka Kernel)

0	0	**1**
0	**1**	0
1	0	0

1	-1	-2	-1
-1	-2	-1	-2
-2	-1	-2	-1
-1	-2	-1	**1**

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	**1**

POOLING LAYER

Further reduce the dimensions by taking Mean/Max/Min of smaller group of cells.

POOLING LAYER

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	**1**

POOLING LAYER

Max Pooling

1	0
0	1

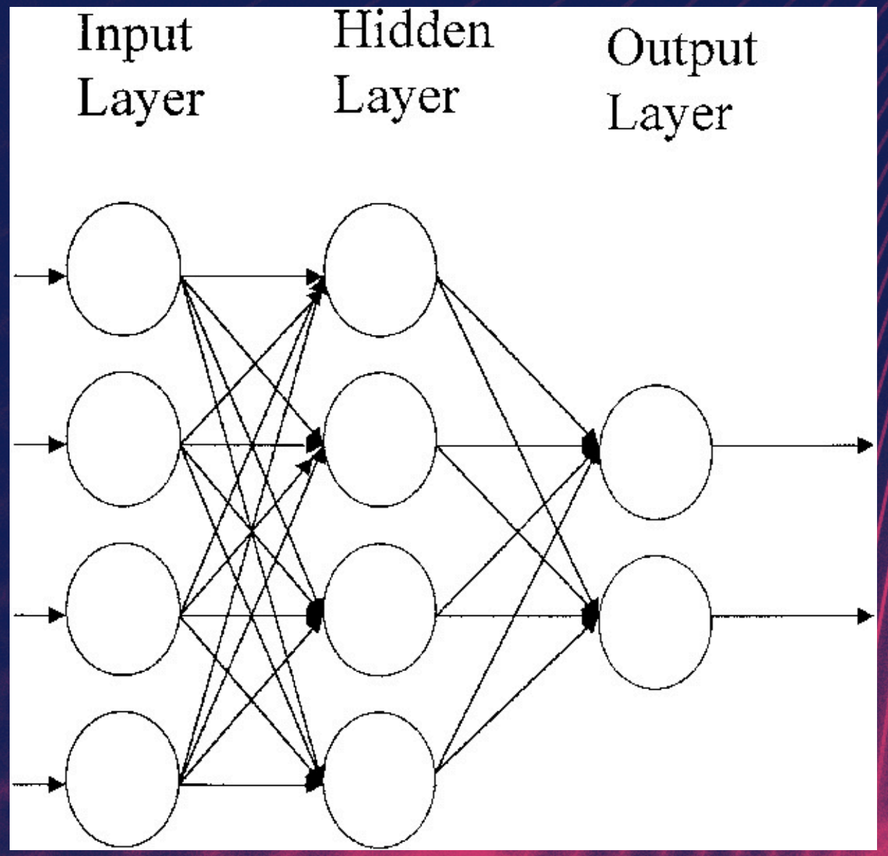
Min Pooling

0	0
0	0

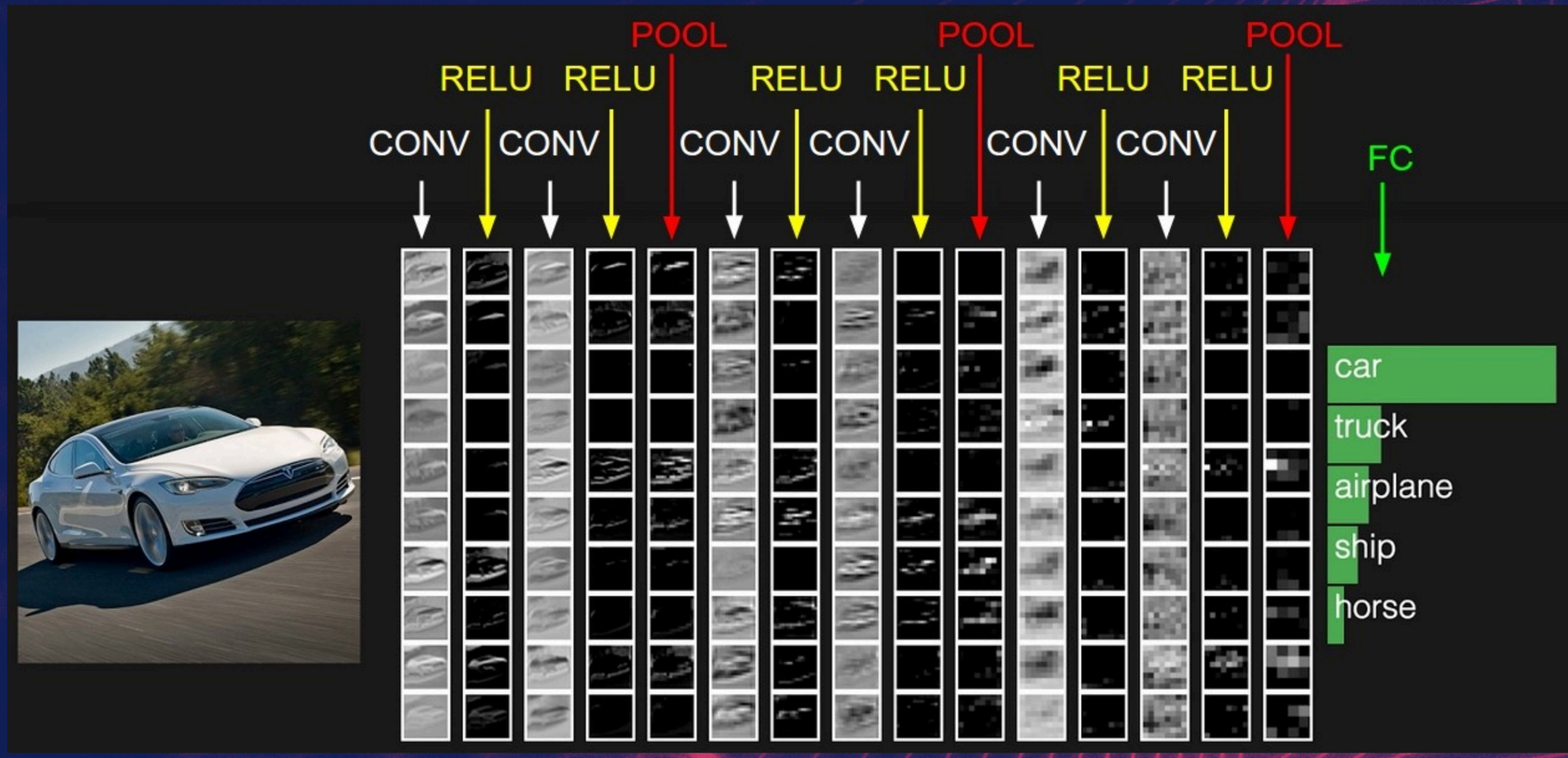
Mean Pooling

0.25	0
0	0.25

FULLY CONNECTED LAYER



ENTIRE CNN NETWORK

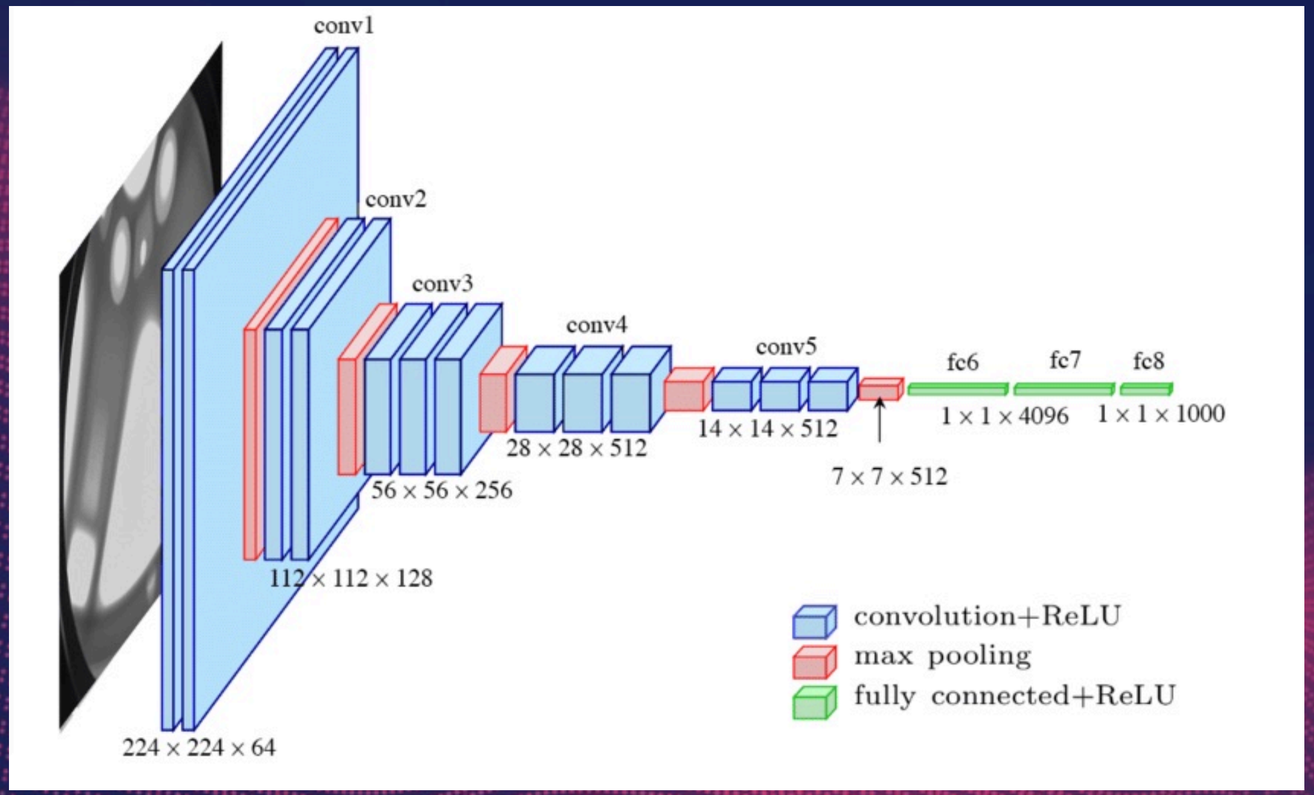


CODE IMPLEMENTATION

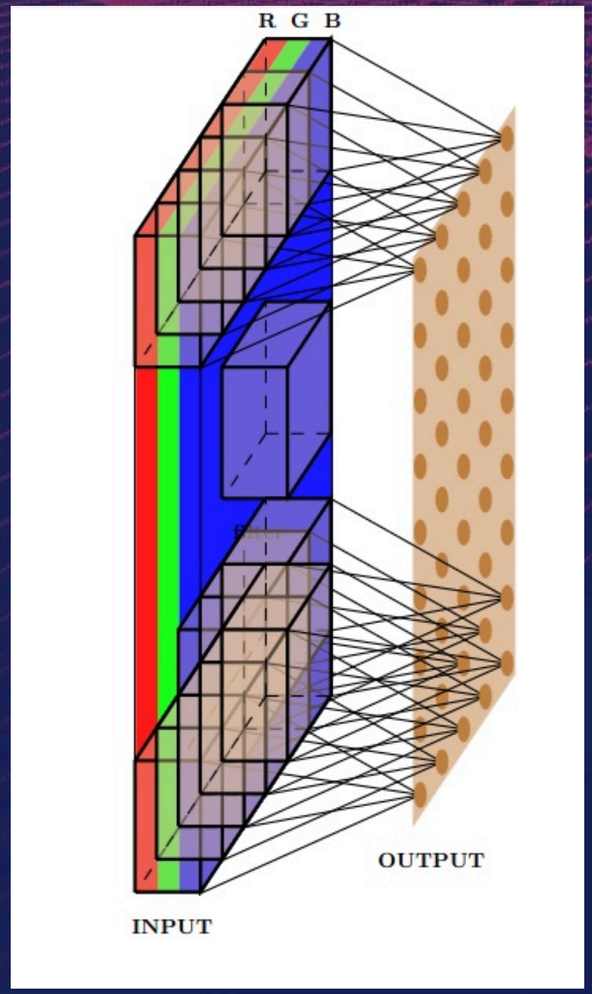
AI CLUB

VGG 19

VGG stands for Visual Geometry Group , belongs to Oxford university.
The aim of the VGG models was to understand, how the depth of the CNN models affect the accuracy when it comes to the tasks like image classification and image recognition.

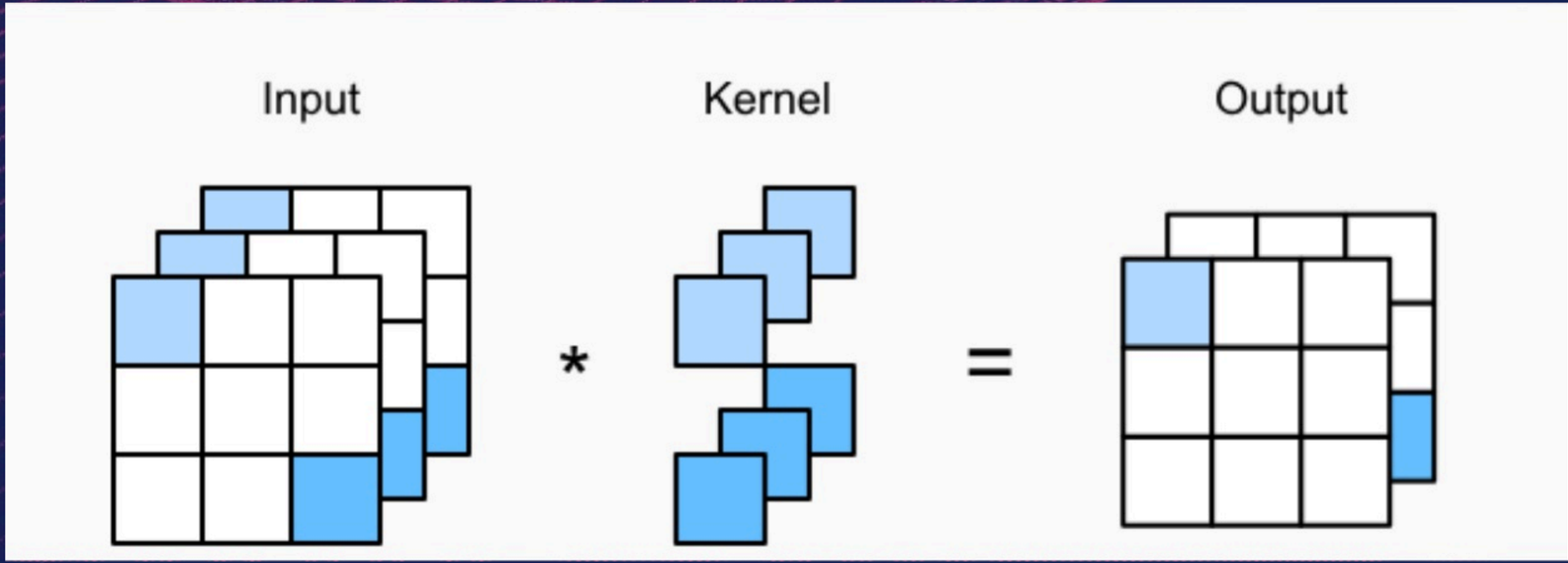


CONVOLUTION OF 3D INPUTS



Generally the filter that you convolve your 3D input image with, is of the same depth as of the input image. For example: for a RGB Input image the filter taken will be of depth 3 since the input image is of depth 3 (3 color channels Red, Green and Blue) .

CONVOLUTION OF 3D INPUTS



CONVOLUTION OF 3D INPUTS

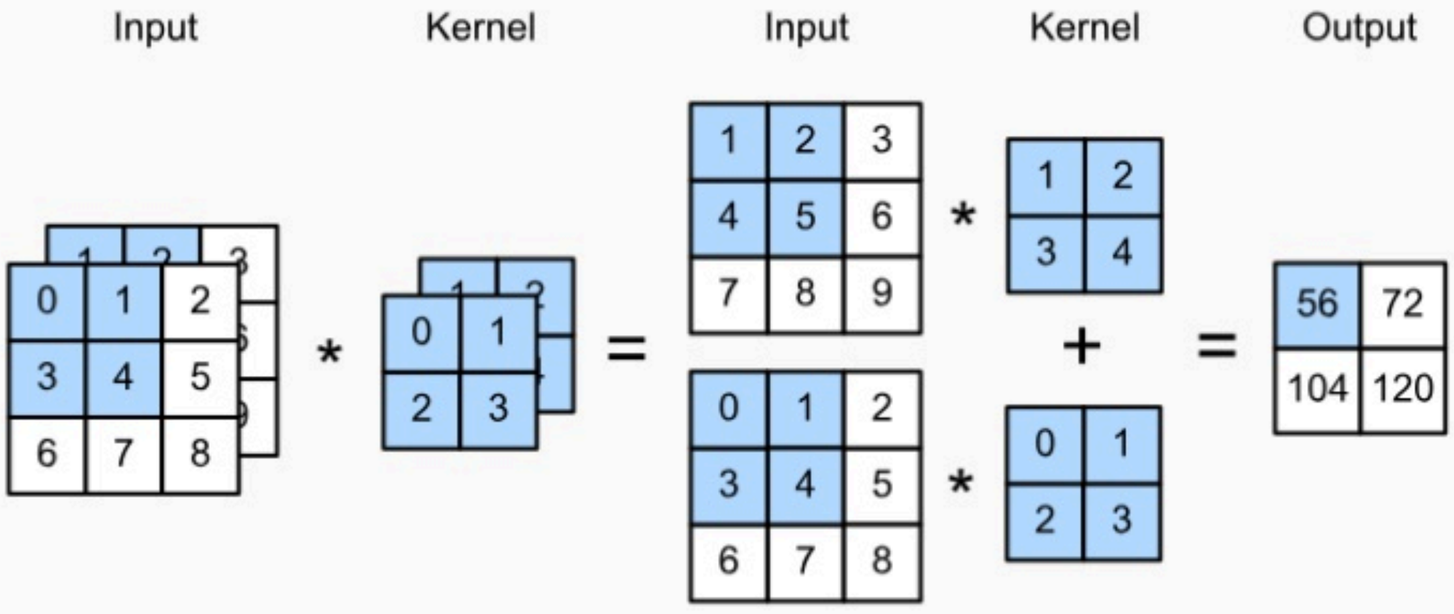
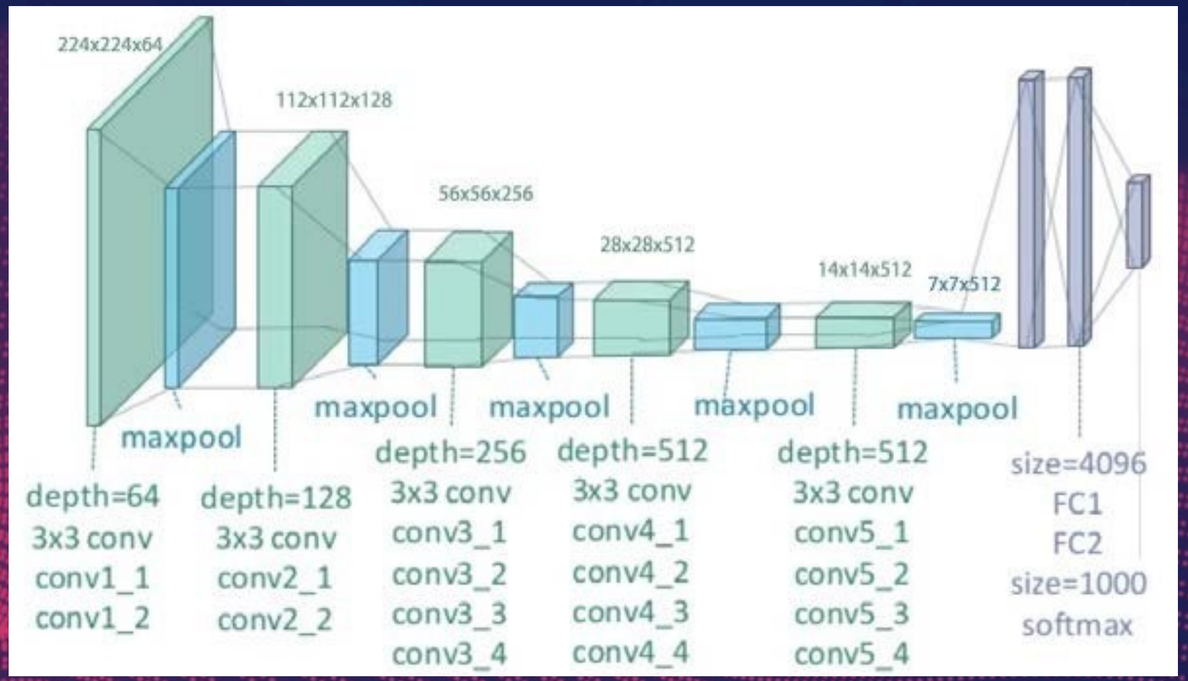


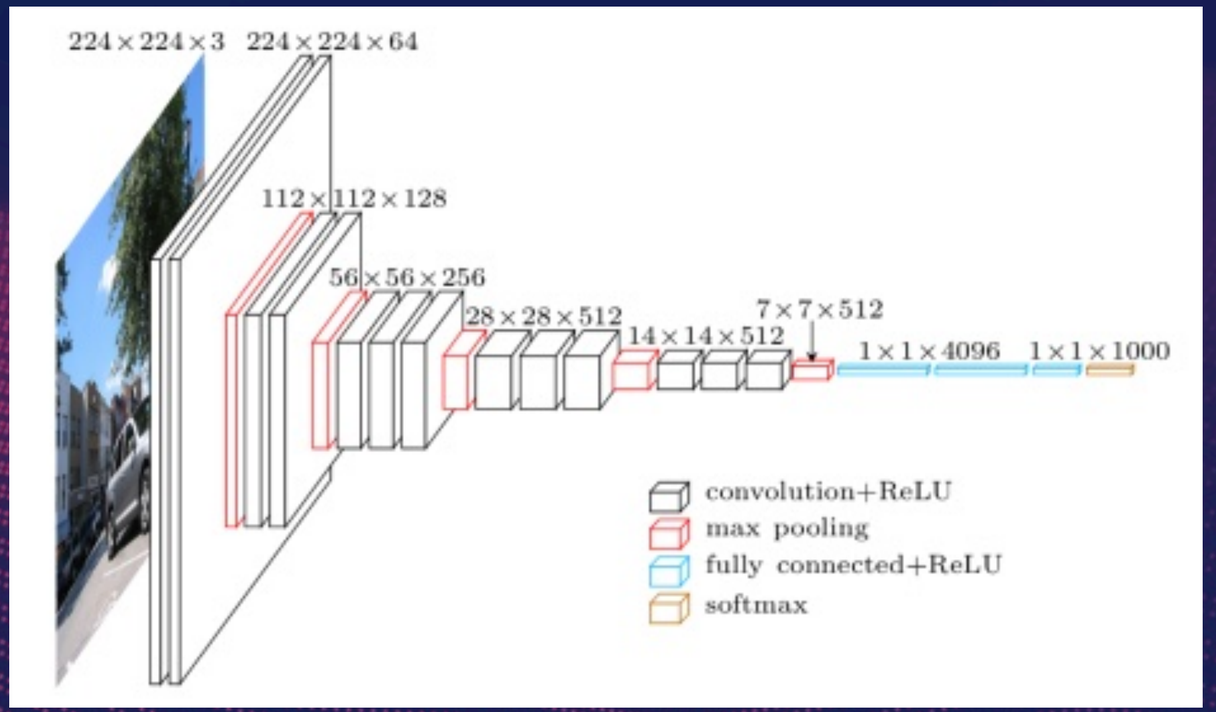
Fig. 7.4.1 Cross-correlation computation with two input channels.

CHARACTERISTICS OF VGG19 MODEL

- VGG19 consists of 16 convolutional layers and 3 fully connected layers, in total 19 layers
- The input to the VGG19 model is consdidered to be a 224 X 224 RGB image
- VGG19 uses 3x3 convolutional layers with stride=1 and padding=1 in order to maintain the dimensions of the feature maps.
- After each convolutional block, a 2x2 max-pooling layer with a stride of 2 is applied to reduce the spatial dimensions of the feature maps by half.
- After the series of convolutional and max-pooling layers, the resulting feature maps are flattened into a 1D vector and is given as input to the fully connected layer. The fully connected layer is followed by a softmax activation ehich gives the probability distribution.

VGG 19


224x224x64
112x112x128
56x56x256
28x28x512
14x14x512
7x7x512
maxpool
maxpool
maxpool
maxpool
maxpool
depth=64
3x3 conv
conv1_1
conv1_2
depth=128
3x3 conv
conv2_1
conv2_2
depth=256
3x3 conv
conv3_1
conv3_2
conv3_3
conv3_4
depth=512
3x3 conv
conv4_1
conv4_2
conv4_3
conv4_4
depth=512
3x3 conv
conv5_1
conv5_2
conv5_3
conv5_4
size=4096
FC1
FC2
size=1000
softmax


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

ACHIEVEMENTS OF VGG 19

- VGG19 was one of the top performers in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 with an error of 7.3%
- Had a better performance and accuracy than the former shallow models.
- VGG19's pre-trained weights on ImageNet have been widely used for transfer learning.
- Due to the high performance and wide range of applications in various domains like medical image analysis and facial recognition , VGG19 was set as a benchmark for the upcoming CNN architectures and designs.

slido.com

#1259016