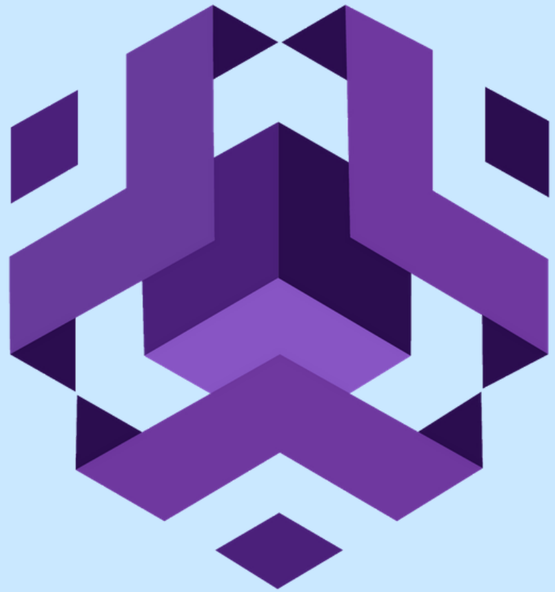
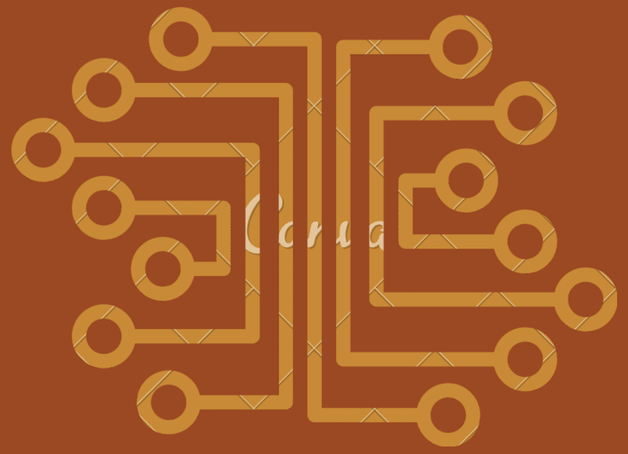
NEURES

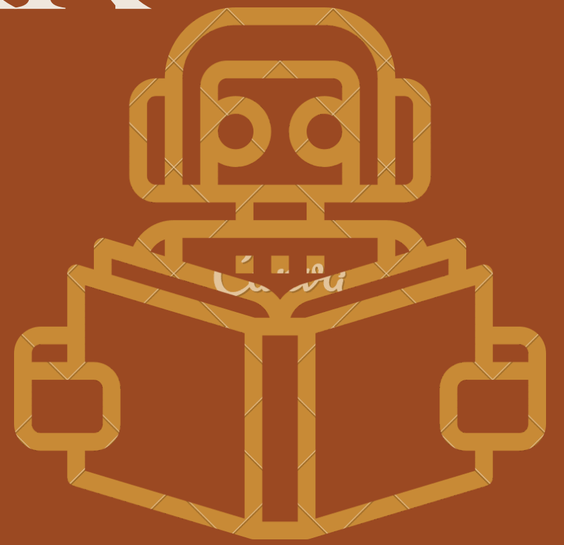
FROM DATA SCIENCE COMMUNITY SRM

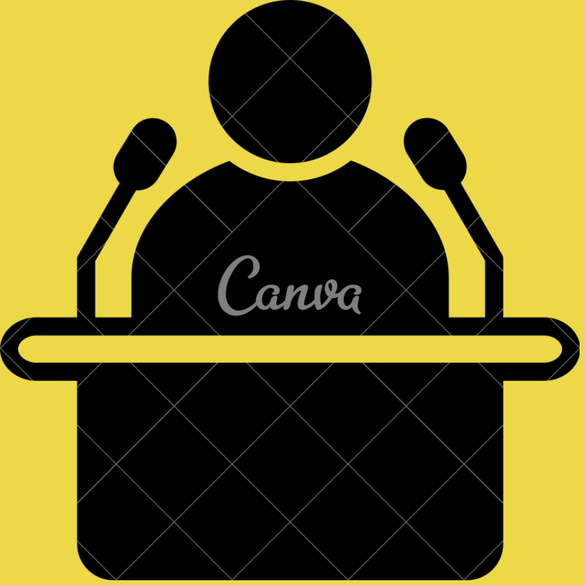


DAY 3

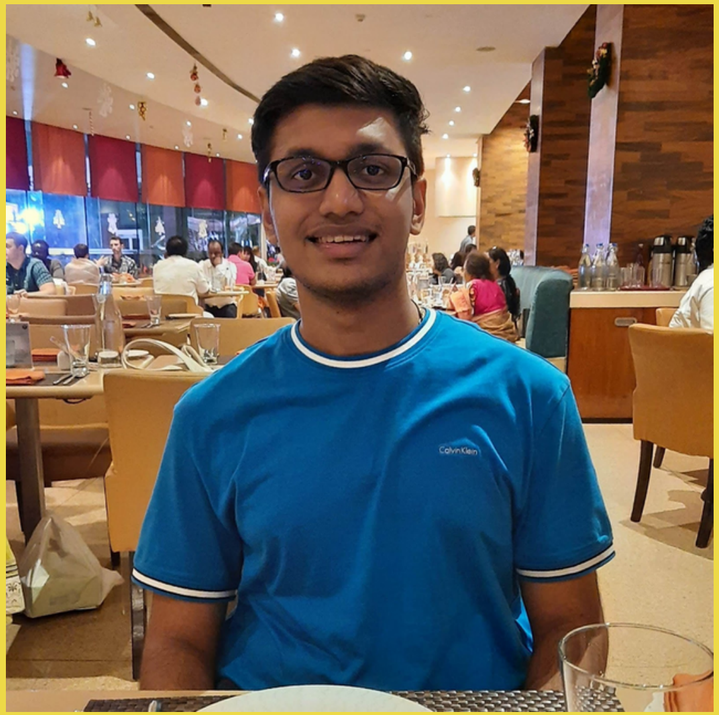


CONVOLUTIONAL NEURAL NETWORK





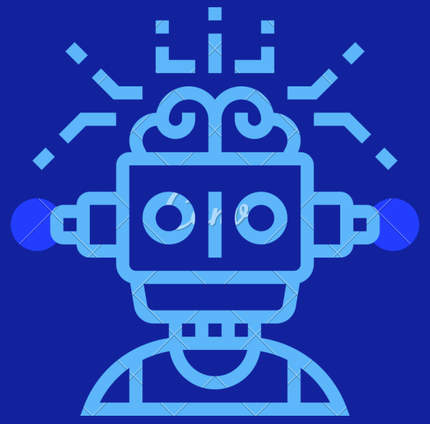
SPEAKERS



Tejas Chintala

Harshit Aggarwal

AGENDA



INTRODUCTION TO CNN

IMPORTANCE, USAGE

OPERATIONS

KERNELS, PADDING, STRIDES

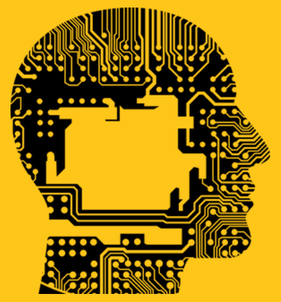
LAYERS

DISCUSSION OF THE ARCHITECTURE OF THE CNN

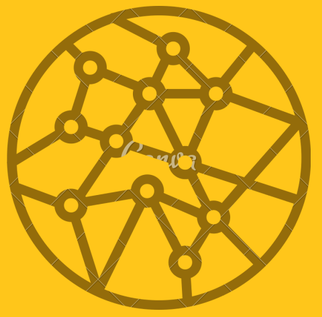
HANDS ON SESSION

GOOGLE COLAB

HERE WE GO!!

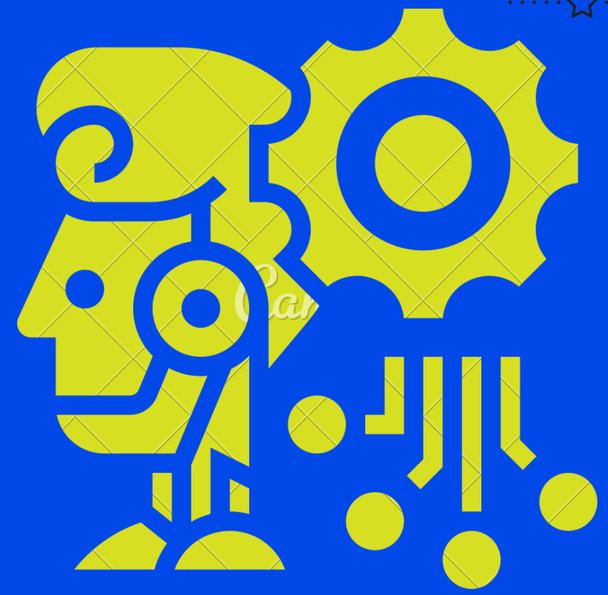


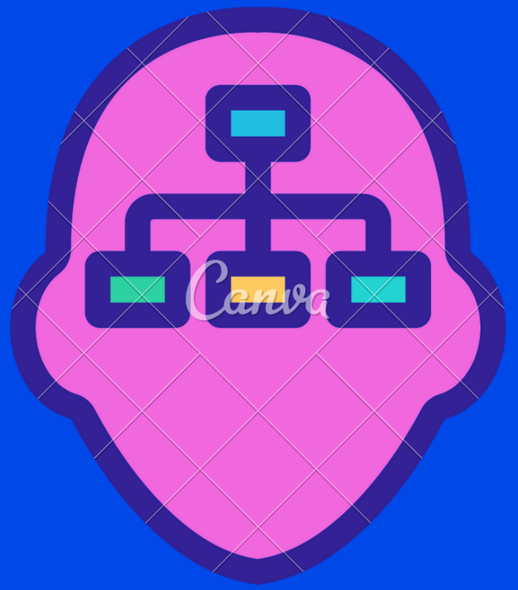
WHAT IS CNN?

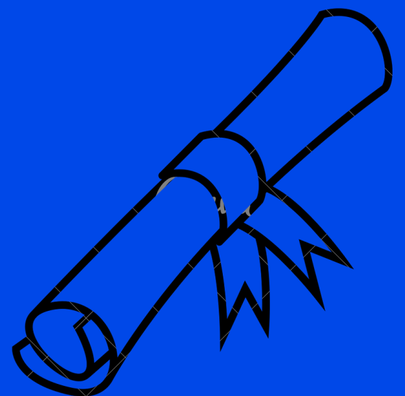


CNN's are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images.Their applications range from image and video recognition, image classification, medical image analysis, computer vision, and natural language processing.

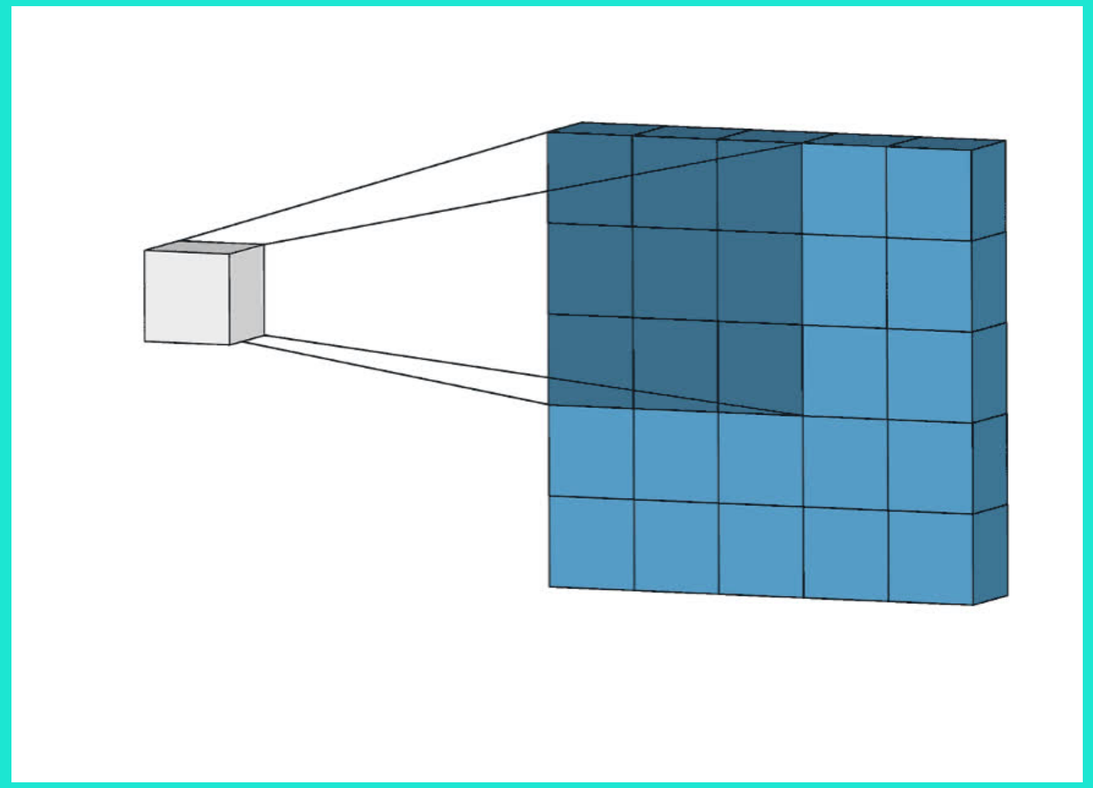
The term ‘Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other.



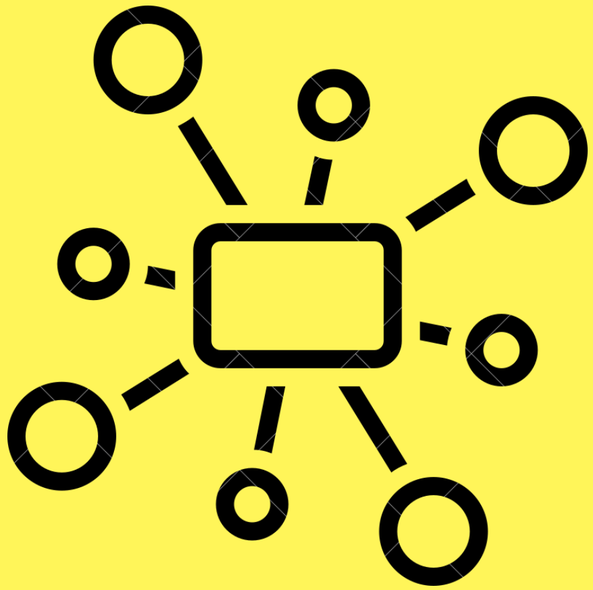




WHY DO WE USE CNN?

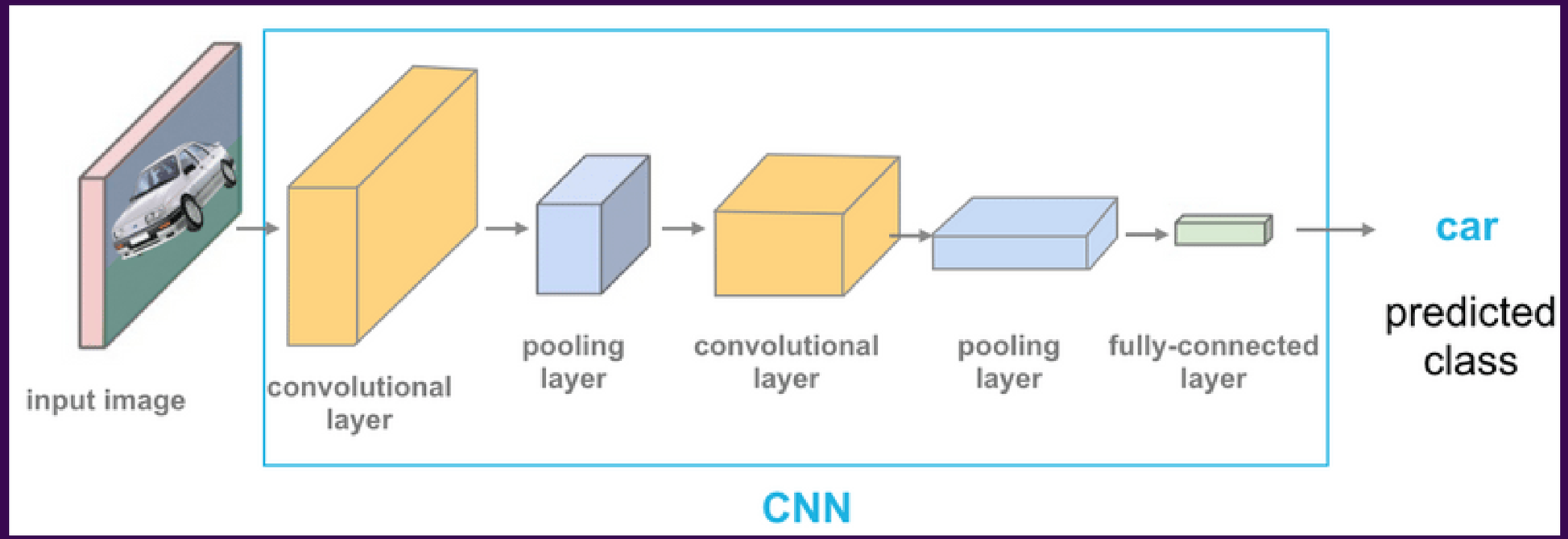


WHY DO WE USE CNN?



We know that Neural Networks are good at Image Recognition. Now if we consider an image recognition task and the image is of Large pixels then the number of parameters for a Neural Network increases. This makes Neural networks slow and consumes a lot of computational power.

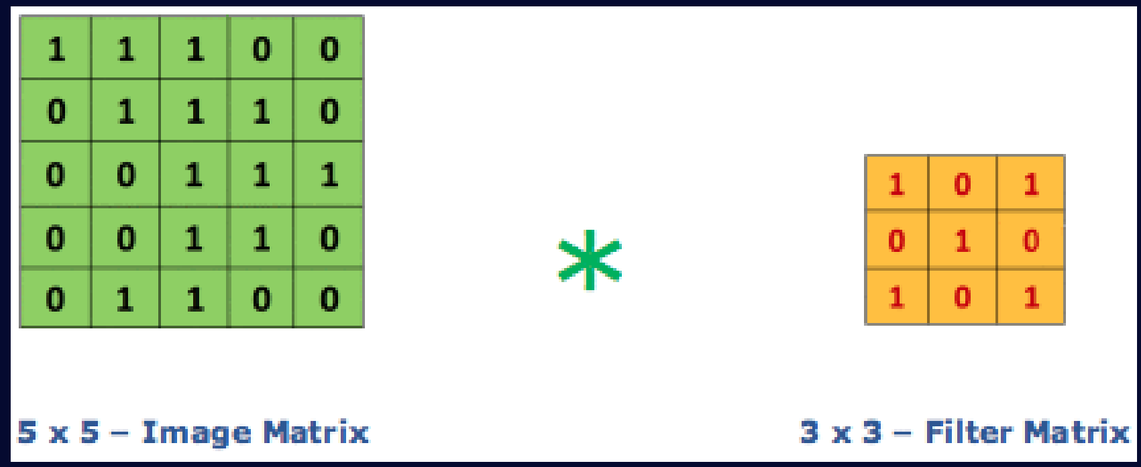
For Ex: If we process a 64x64x3 size image, then we will be getting 12288 input parameters, but if the image is a high resolution with 1000x1000x3, then it has 3 million input parameters to process. This takes lot of time and computational power.

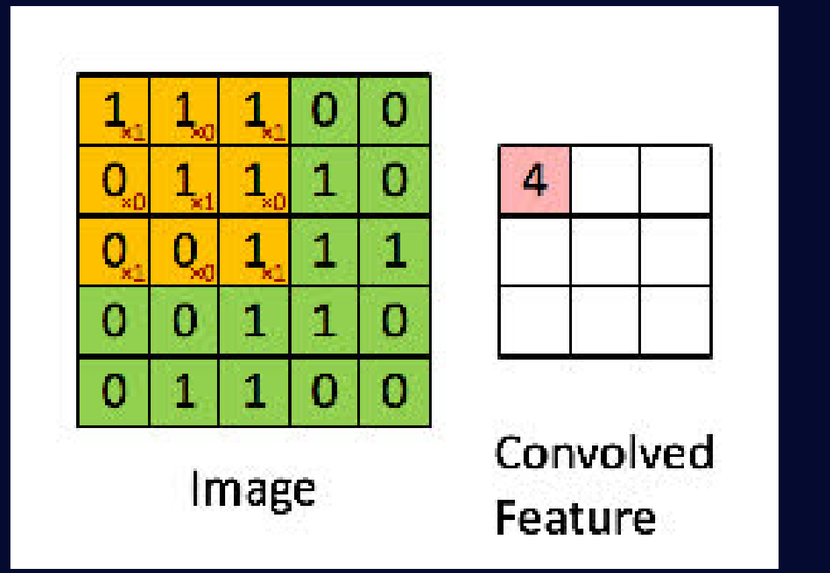

input image
convolutional layer
pooling layer
convolutional layer
pooling layer
fully-connected layer
car
predicted class
CNN

2D CONVOLUTION

The kernel “slides” over the 2D input data, performing an element-wise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

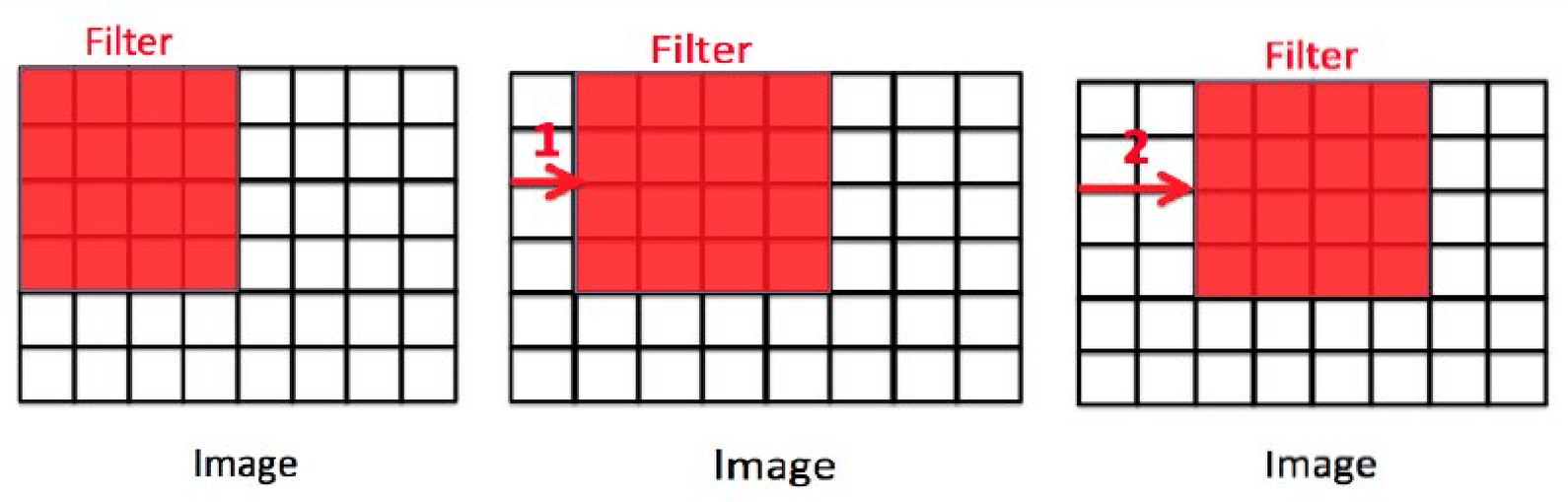
The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essential, the weighted sums (with the weights being the values of the kernel itself) of the input features located roughly in the same location of the output pixel on the input layer.





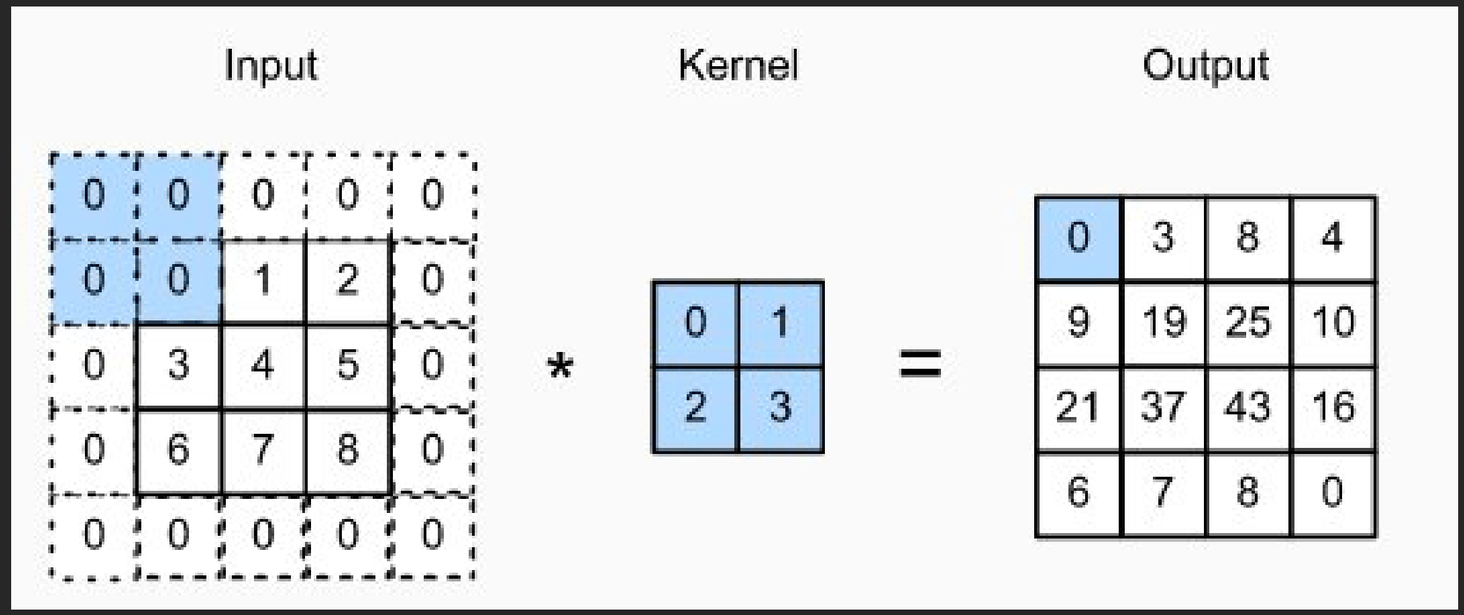
STRIDING

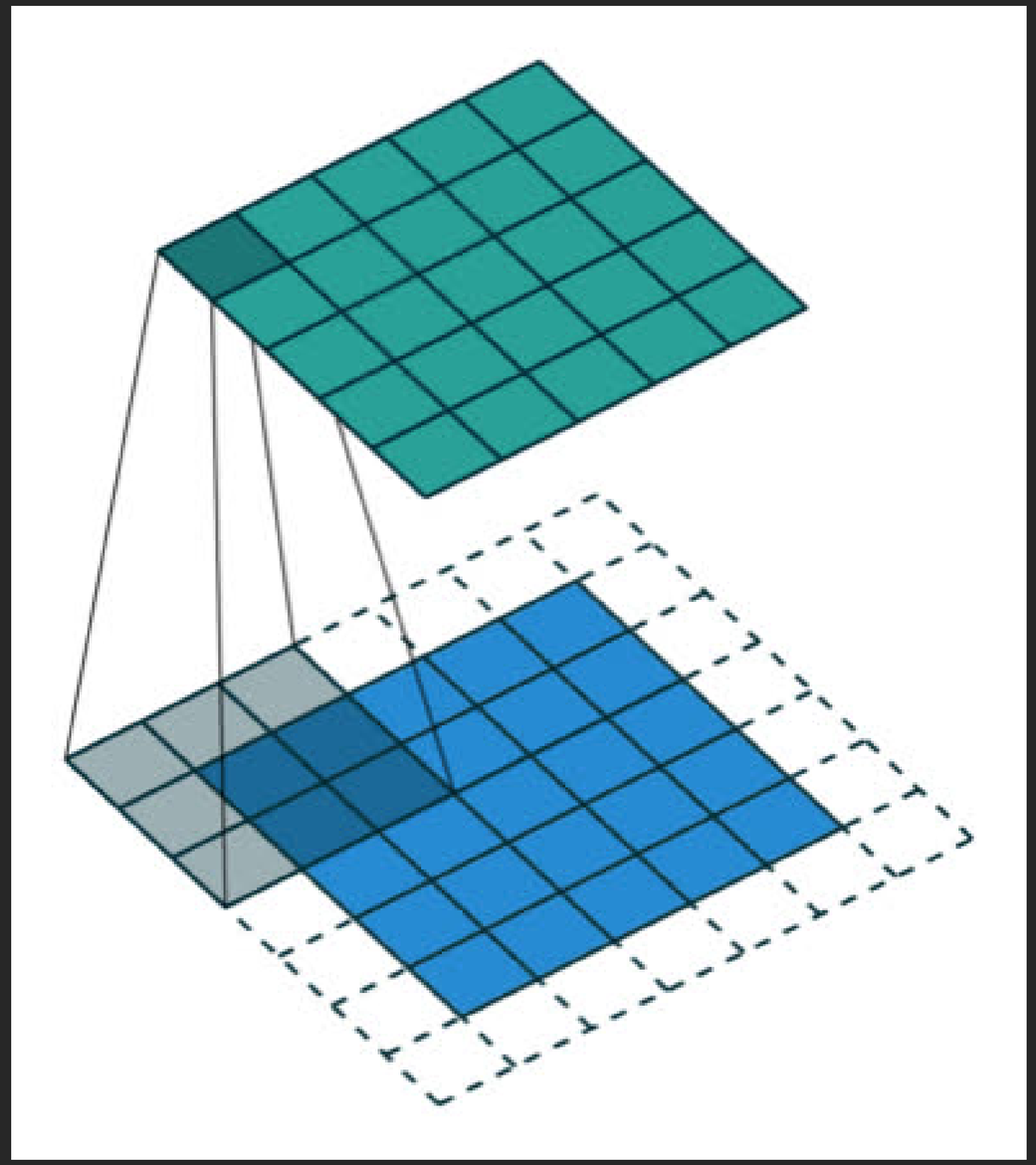
Stride is the number of pixels shifts over the input matrix. The idea of the stride is to skip some of the slide locations of the kernel. A stride of 1 means to pick slides a pixel apart, so basically every single slide, acting as a standard convolution. A stride of 2 means picking slides 2 pixels apart, skipping every other slide in the process.

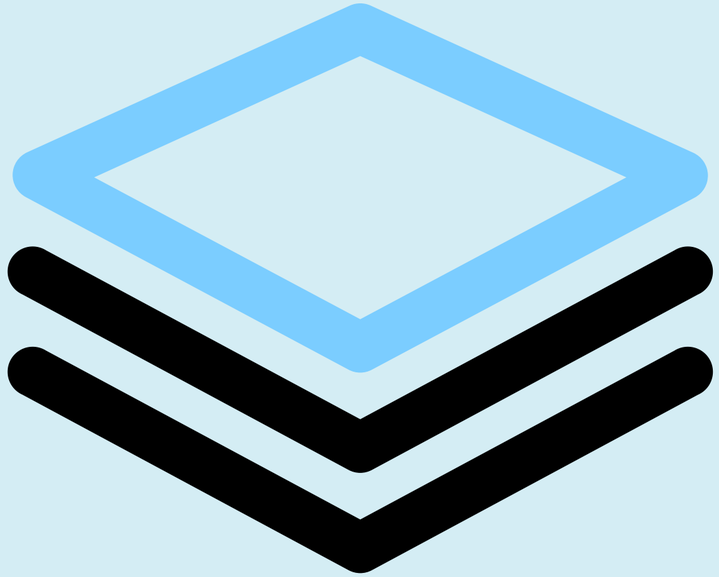


PADDING

PADDING PRESERVES THE SIZE OF THE ORIGINAL IMAGE. PADDING
DOES SOMETHING PRETTY CLEVER TO SOLVE THIS: PAD THE EDGES WITH EXTRA,
“FAKE” PIXELS (USUALLY OF VALUE 0, HENCE THE OFT-USED TERM “ZERO PADDINGS”).
THIS WAY, THE KERNEL WHEN SLIDING CAN ALLOW THE ORIGINAL EDGE PIXELS TO BE AT ITS CENTER,
WHILE EXTENDING INTO THE FAKE PIXELS BEYOND THE EDGE, PRODUCING AN OUTPUT THE SAME SIZE AS THE INPUT.







DIFFERENT LAYERS OF CNN

INPUT LAYER

CONVO LAYER

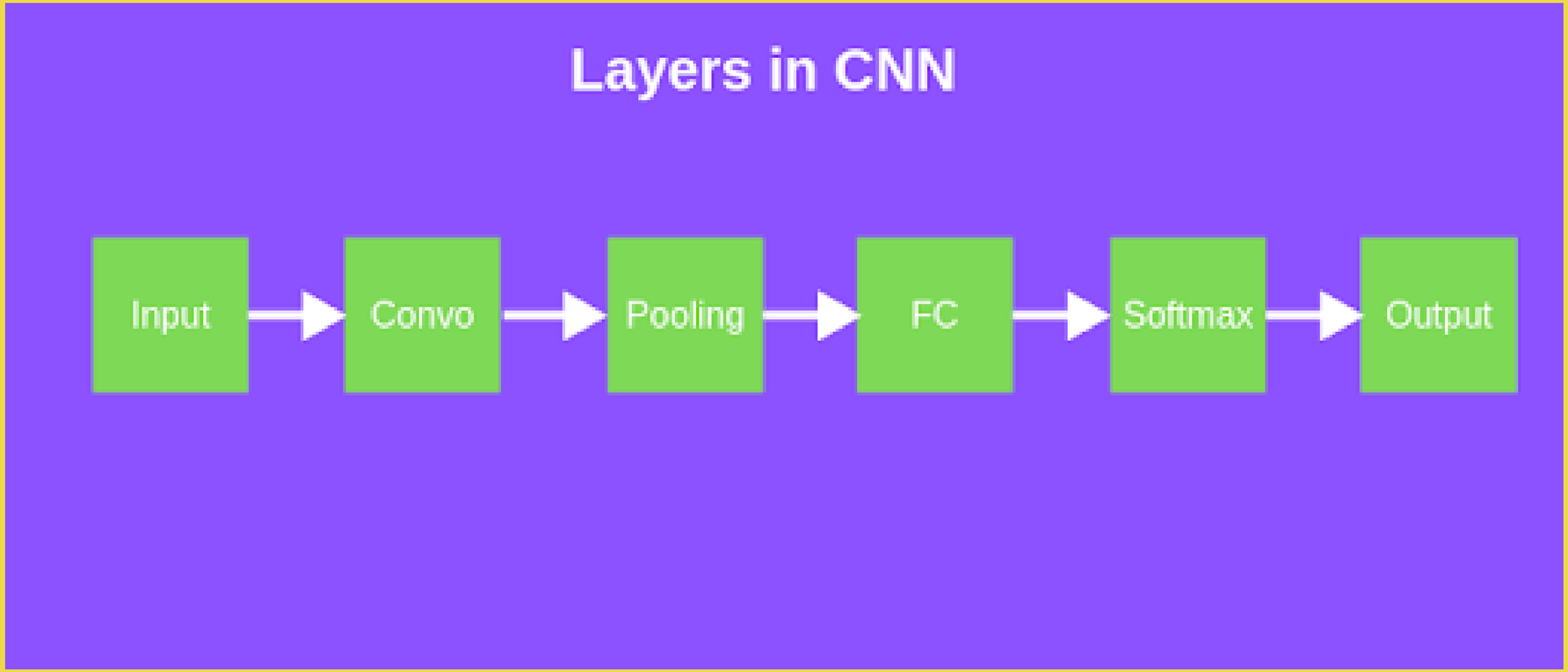
(CONVO + ReLu)

POOLING LAYER

FULLY CONNECTED LAYER (FC)

DROUPOUT LAYER

OUTPUT LAYER


Layers in CNN
Input
Convo
Pooling
FC
Softmax
Output

INPUT LAYER

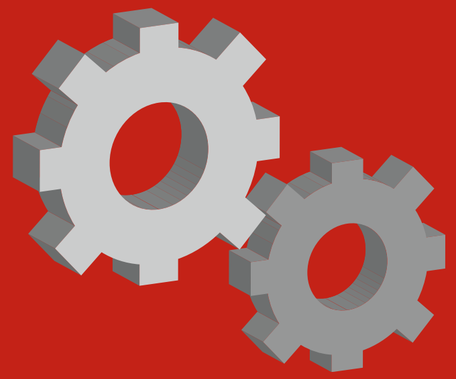
INPUT IMAGE

Input layer in CNN should contain image data. Image data is represented by three dimensional matrix, reshaping it into a single column.For eg: image of dimension 28x28 = 784, we need to convert it into 784 x 1 before feeding into input. If you have ‘m’ training examples then dimension of input will be (784,m).

Code Snippet:

```
INPUTS = KERAS.INPUT(SHAPE=INPUT_SHAPE)
```

CONVO LAYER

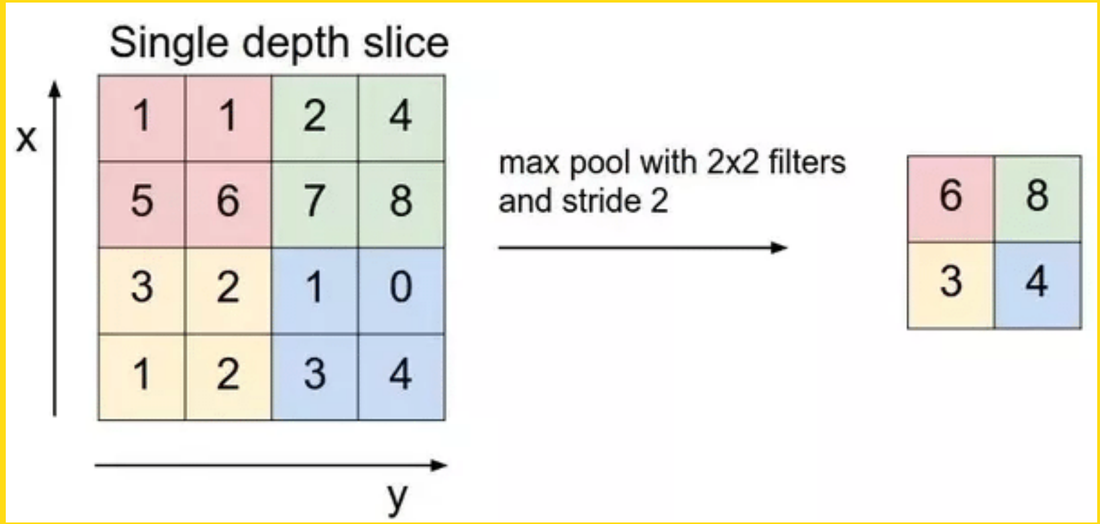


This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

Code Snippet:

```
LAYERS.CONV2D(32, 3, STRIDES=2, PADDING="SAME")
```

POOLING LAYER

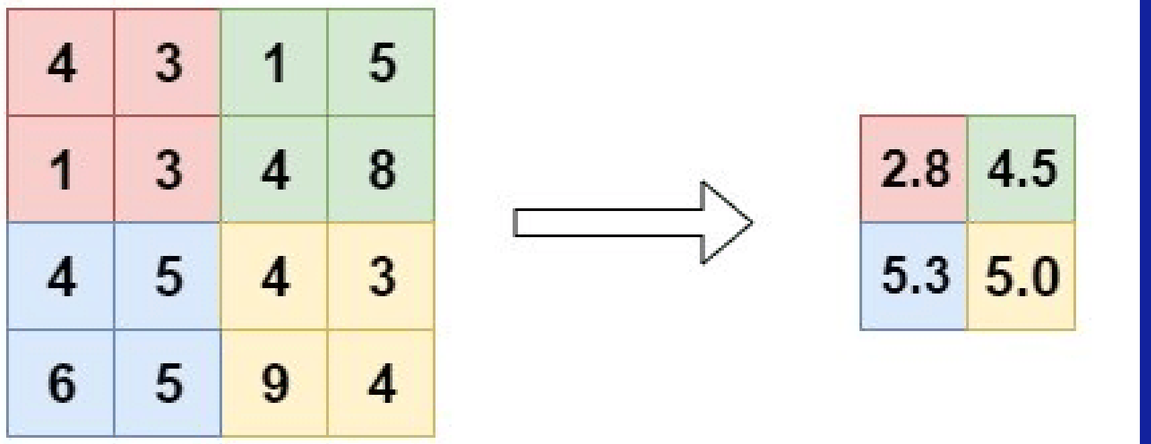


The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs.

- Max Pooling, the largest element is taken from feature map.
- Average Pooling calculates the average of the elements in a predefined sized Image section.

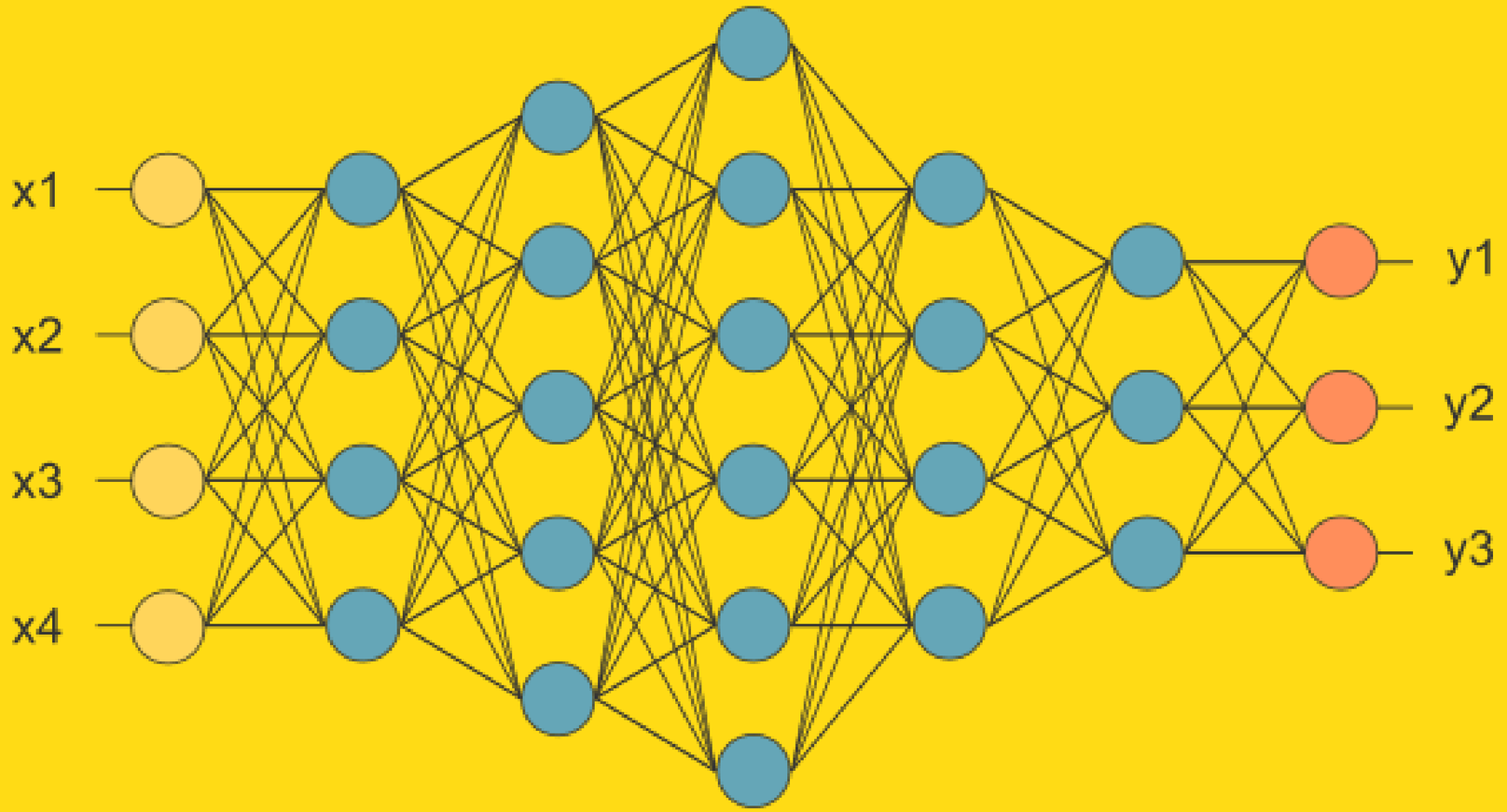
4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4

Avg([4, 3, 1, 3]) = 2.75



FULLY CONNECTED LAYER

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

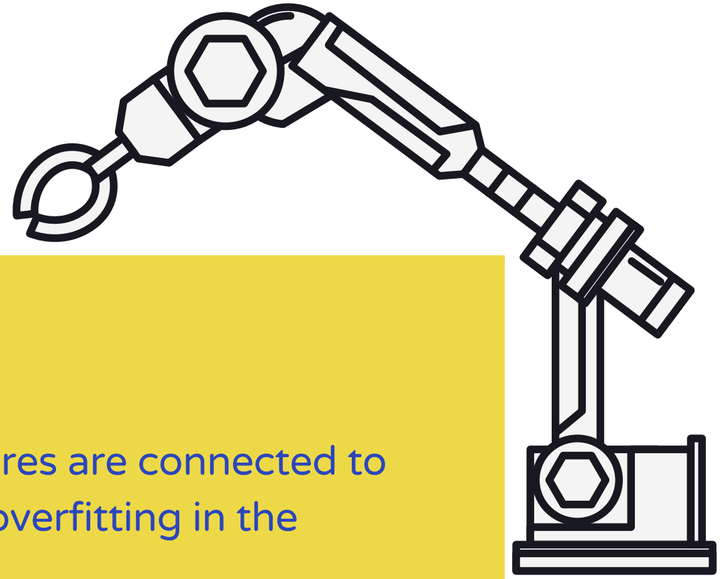


INPUT

HIDDEN LAYER

OUTPUT

DROPOUT LAYER



Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset.

To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during the training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network..

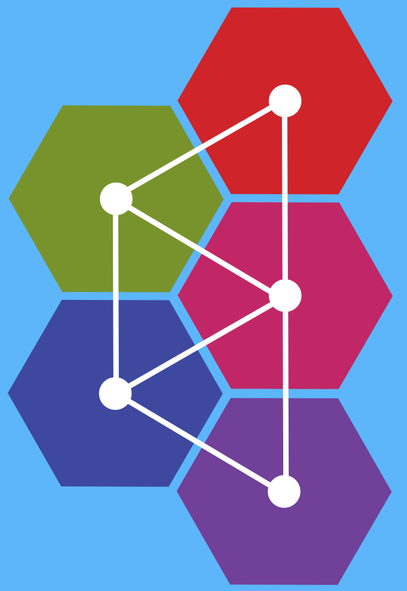
Code Snippet:

```
LAYERS.DROPOUT(0.5)
```

ACTIVATION FUNCTIONS

They are used to learn and approximate any kind of continuous and complex relationship between variables of the network.
It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage.

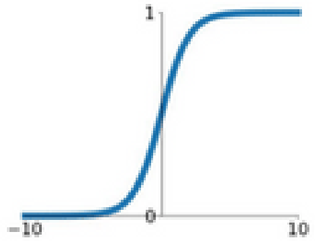
1. ReLu function (Rectified Linear Unit)
2. Softmax, tanH
3. Sigmoid



Activation Functions

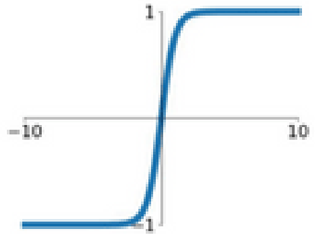
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



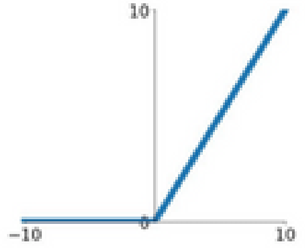
tanh

$$\tanh(x)$$



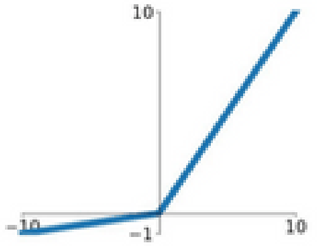
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

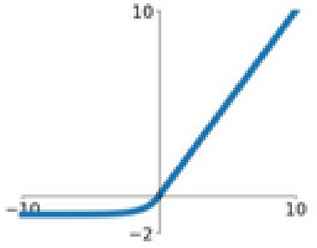


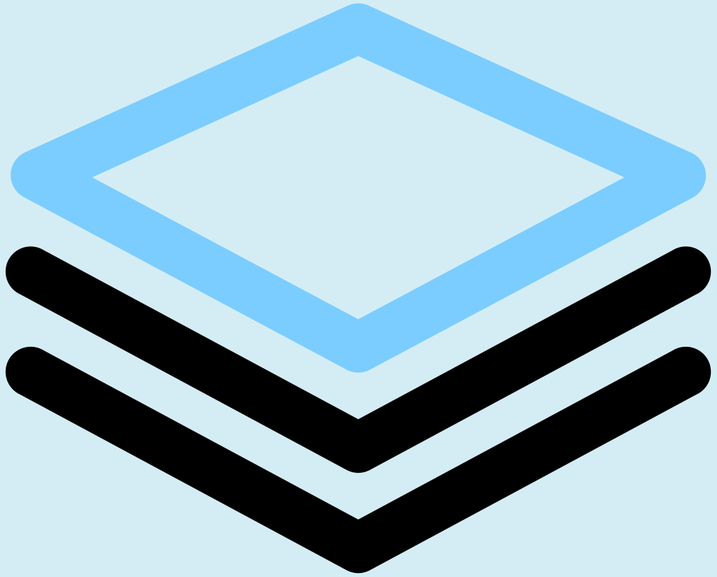
Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

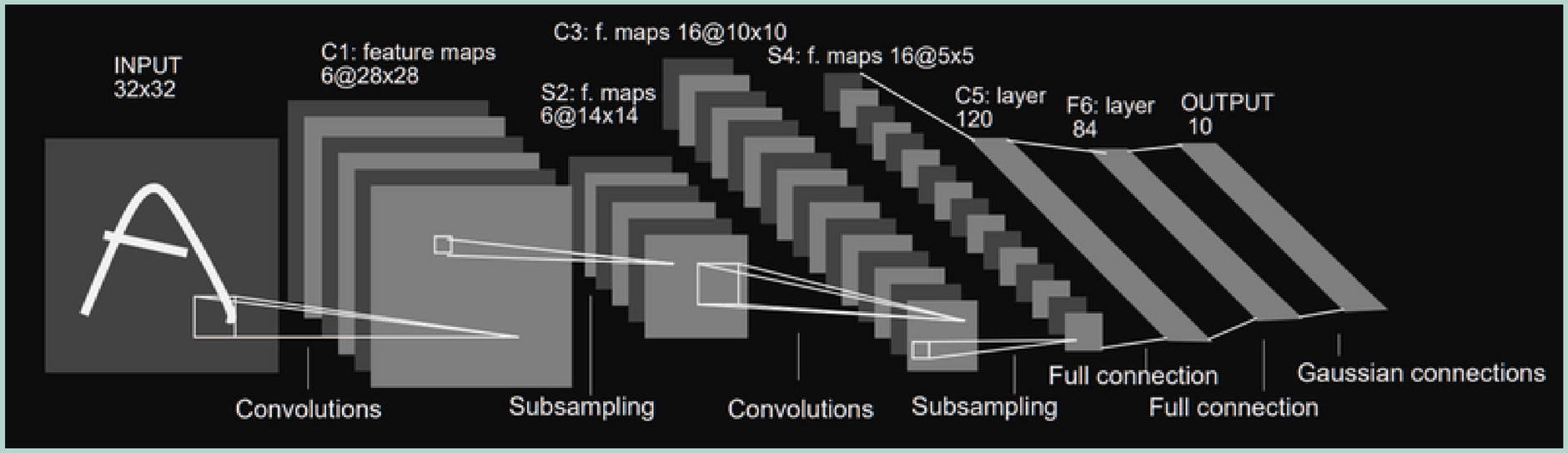
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



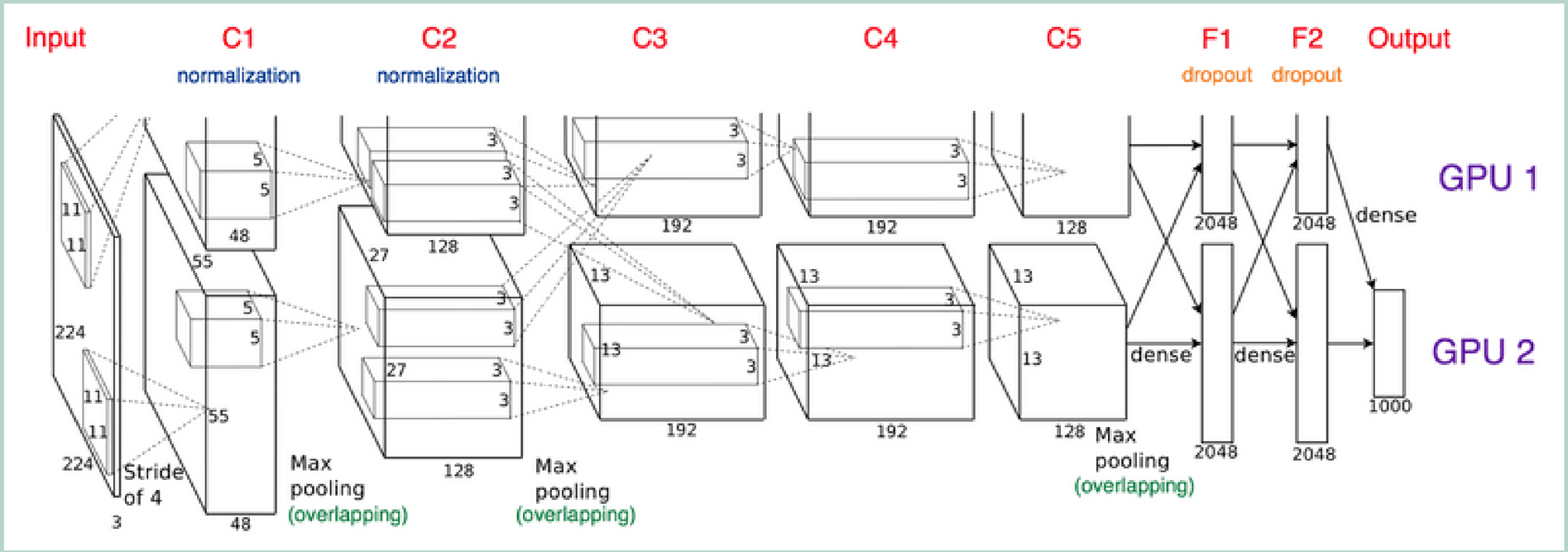


TYPES OF CNN

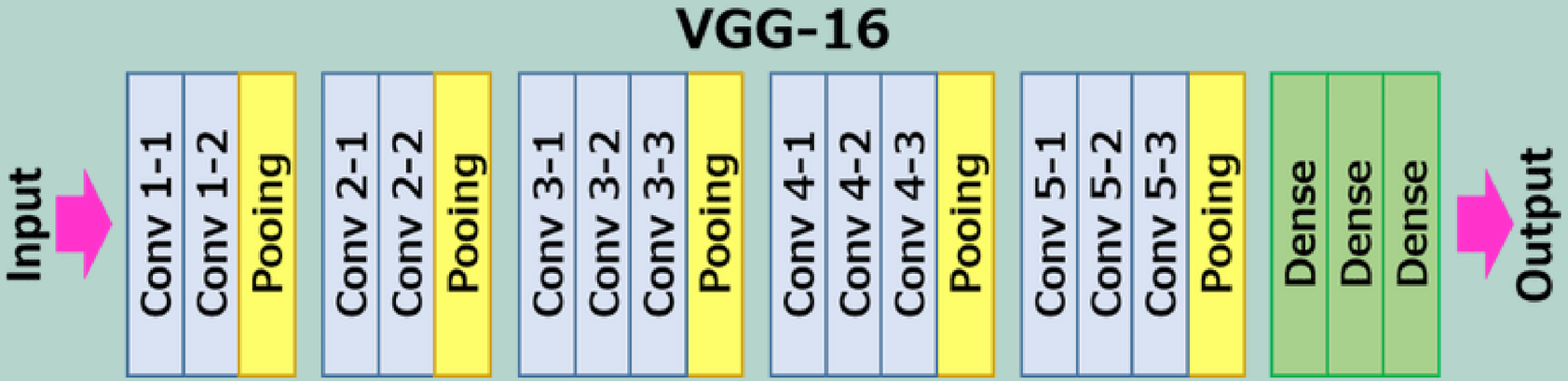
LE NET


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

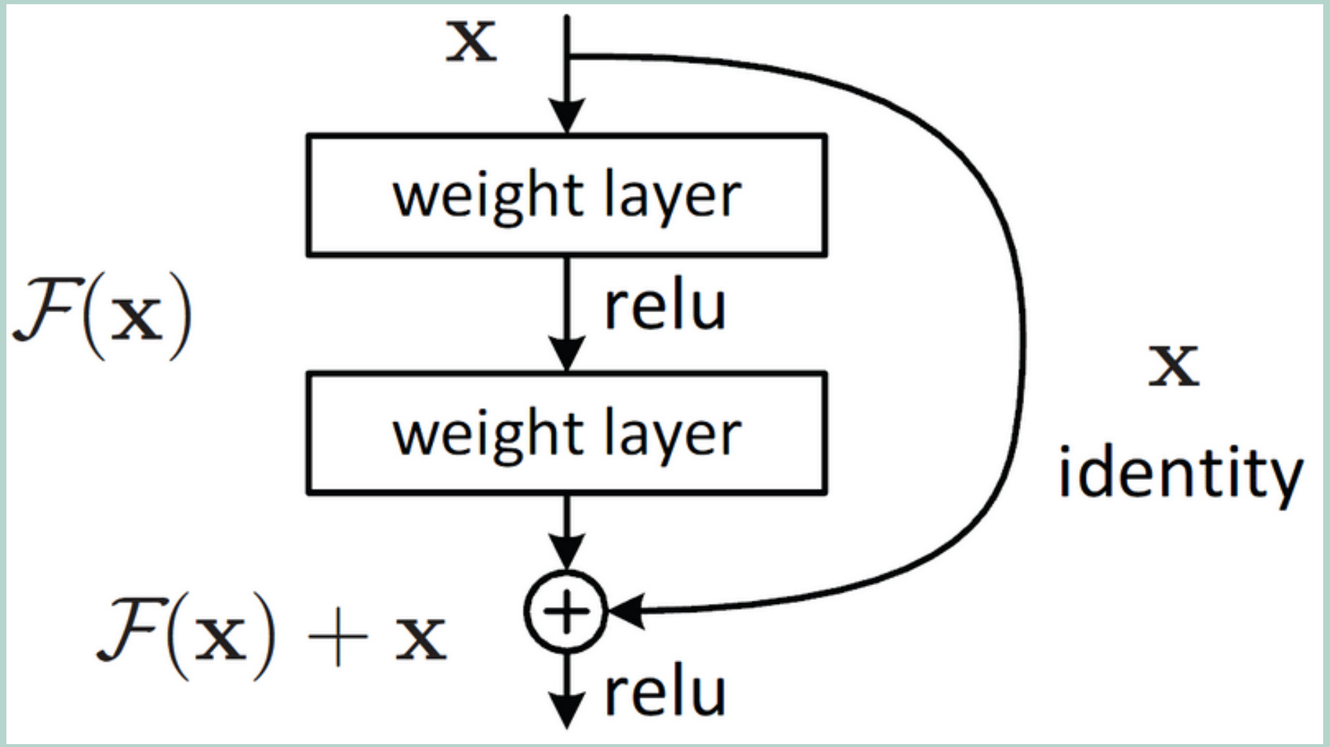
ALEXNET


Input
C1
normalization
C2
normalization
C3
C4
C5
F1
dropout
F2
dropout
Output
GPU 1
GPU 2
224
224
3
11
11
Stride
of 4
55
48
5
5
48
Max
pooling
(overlapping)
27
128
128
3
3
3
Max
pooling
(overlapping)
13
13
192
192
192
128
Max
pooling
(overlapping)
dense
dense
dense
2048
2048
2048
2048
1000

VGGNET16


VGG-16
Input
Conv 1-1
Conv 1-2
Pooing
Conv 2-1
Conv 2-2
Pooing
Conv 3-1
Conv 3-2
Conv 3-3
Pooing
Conv 4-1
Conv 4-2
Conv 4-3
Pooing
Conv 5-1
Conv 5-2
Conv 5-3
Pooing
Dense
Dense
Dense
Output

RESNET


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu


LEASE STAND

Let's Connect

DATASCIENCECOMMUNITY@GMAIL.COM

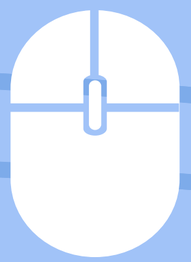
DSCOMMUNITY_SRM

DSCOMMUNITY_SRM

DATA SCIENCE COMMUNITY SRM

DATA SCIENCE COMMUNITY SRM

Hands-On-Session



Thank you

YOU ROCK!

DATA SCIENCE
COMMUNITY SRM

Proudly presenting our **Sponsors** for **NeuRes**

SASHIDO

Give My Certificate

BADElोग.IN

aws

repl.it

Tesseract Lights

SHAPE.AI 2020

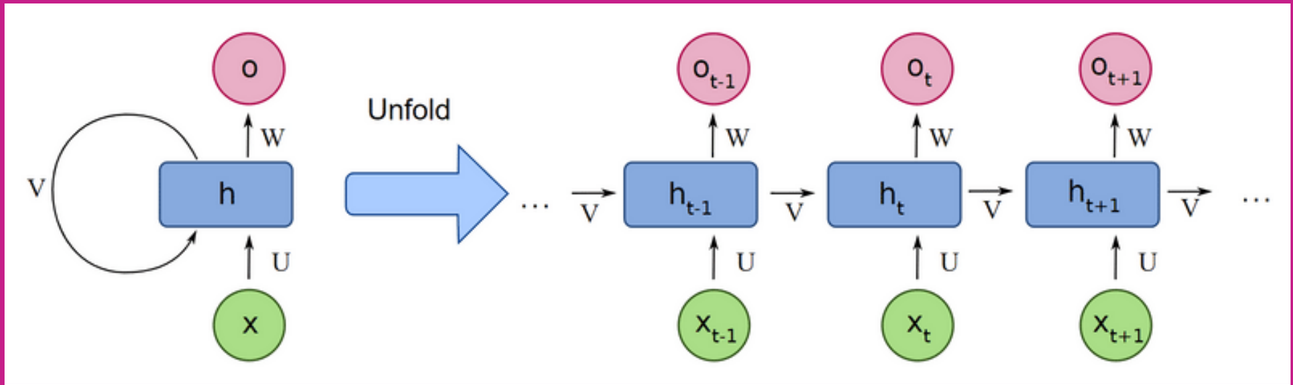
taskade

balsamiq

JOIN US

DAY 4:

RNN AND SEQUENCE MODELS



ON 22ND MARCH

ITS QUIZ TIME!