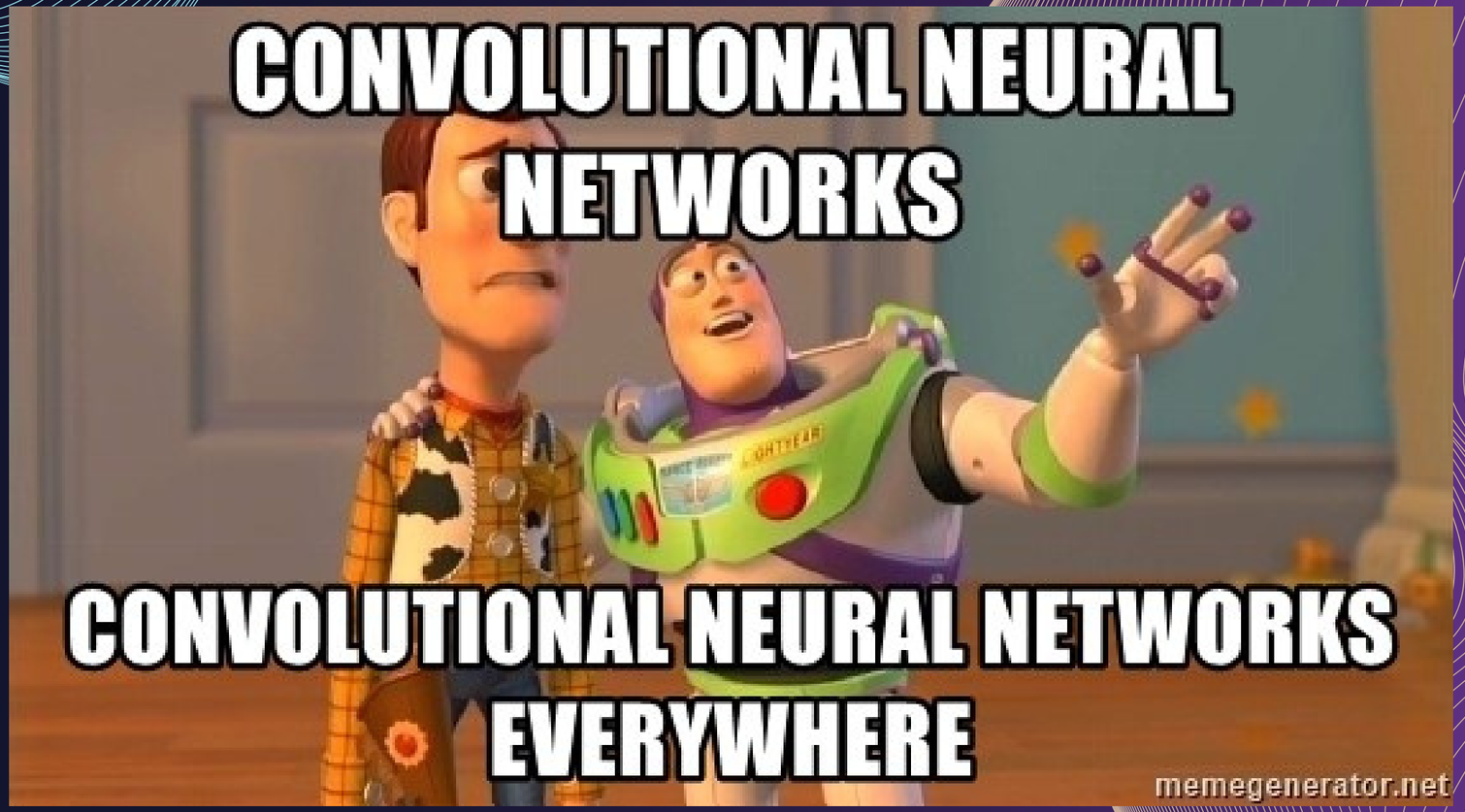
ANALYTICS CLUB X CVI CLUB

CFI SUMMER SCHOOL 2022

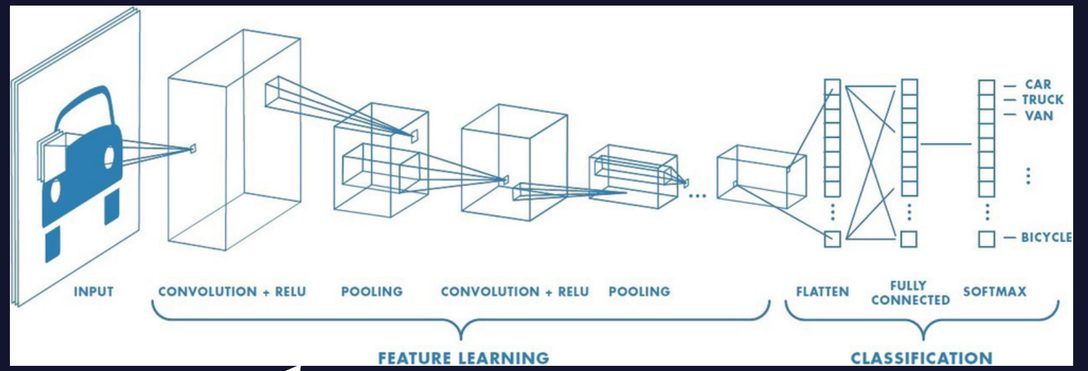


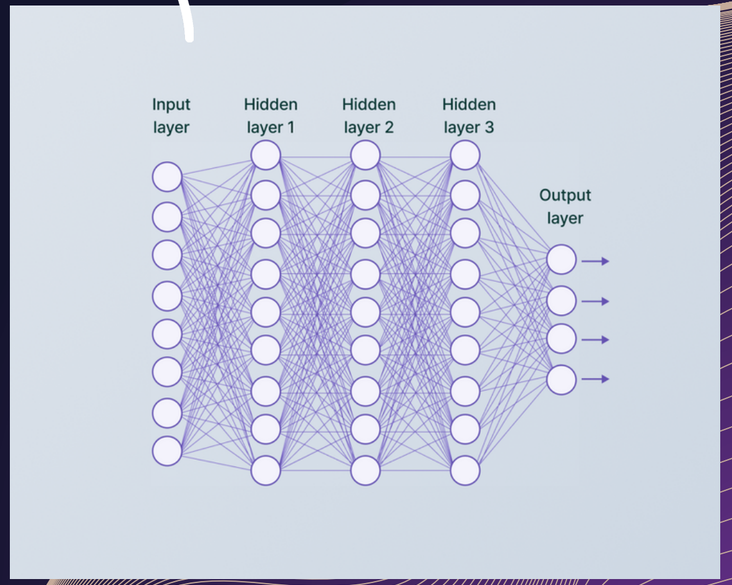
CONVOLUTIONAL NEURAL NETWORKS

TABLE OF CONTENTS

WHY CONVOLUTIONAL NEURAL NETWORKS?

- CNNs are particularly used in image processing.
- We could use the basic neural networks instead of CNNs, but it has been seen that CNNs are able to capture the spatial dependencies in an image.
- Spatial dependency is how a pixel is affected by the pixels surrounding it.
- A normal neural network may work as good as a CNN for basic tasks, but as the complexity of the task increases CNNs are increasingly preferred over neural networks as they are more efficient and reduce the number of parameters in the model drastically.





COMPONENTS OF A CONVOLUTIONAL NEURAL NETWORK

1. CONVOLUTIONAL LAYERS:

This involves performing convolution operation on the input image.

2. POOLING LAYERS:

These layers perform the task of retaining only useful features and discarding irrelavant ones.

3. FULLY CONNECTED LAYERS:

The output image after convolutions and pooling is flattened and passed into a fully connected neural network.

HOW ARE IMAGES REPRESENTED?

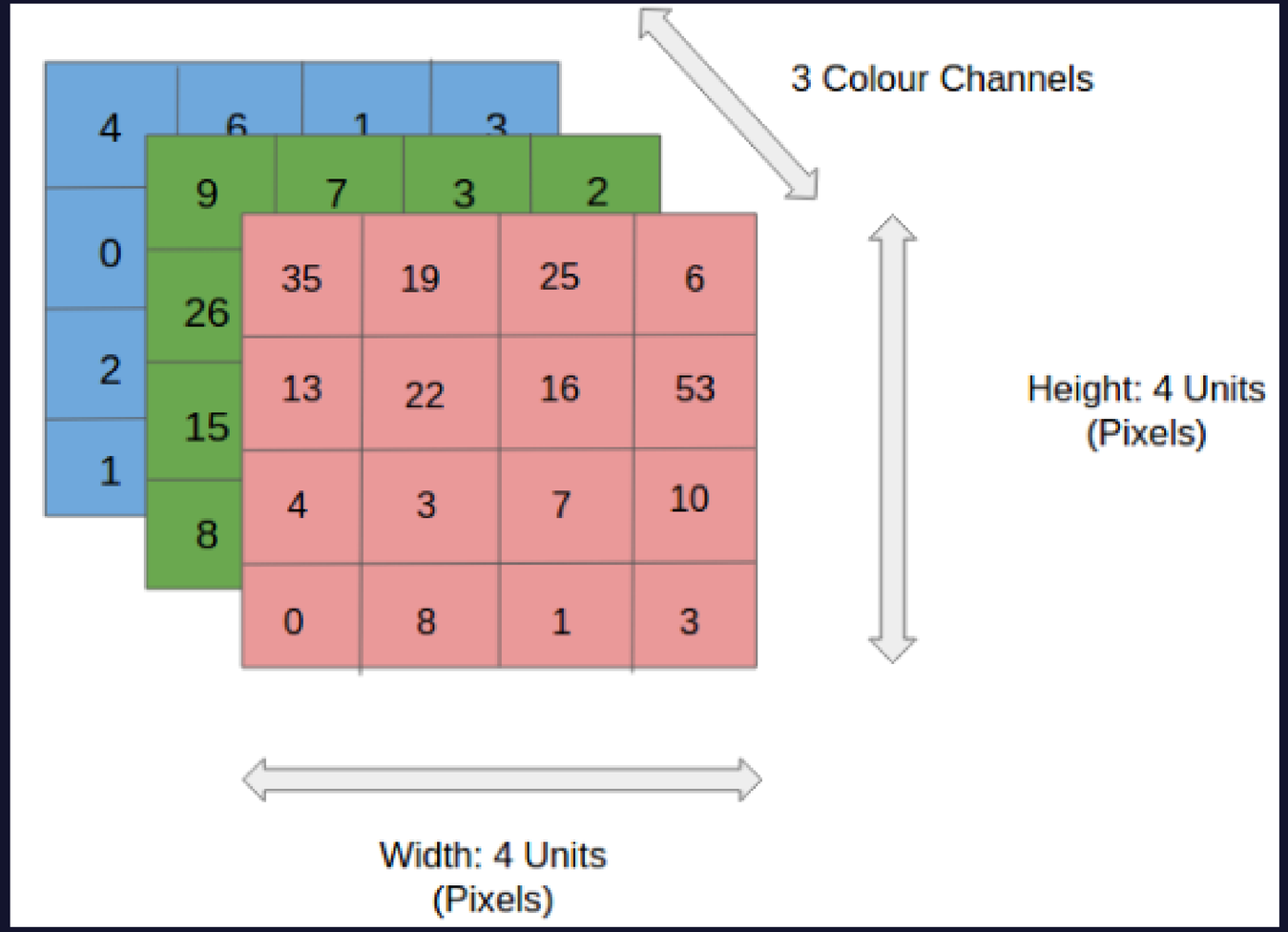
- Images are represented as matrices
- The size of the matrix and the number of channels depends on the resolution of the image and system used for representation.
- There are multiple systems used:

1. Grayscale: This is used for black and white images and each pixel is associated with a single number.

2. RGB system: Each pixel has a number associated with each of Red(R), Green(G) and Blue(B) channels.

4. HSV system: This is a circular system and stands for Hue(H), Saturation(S) and Value(V).

HOW ARE IMAGES REPRESENTED?



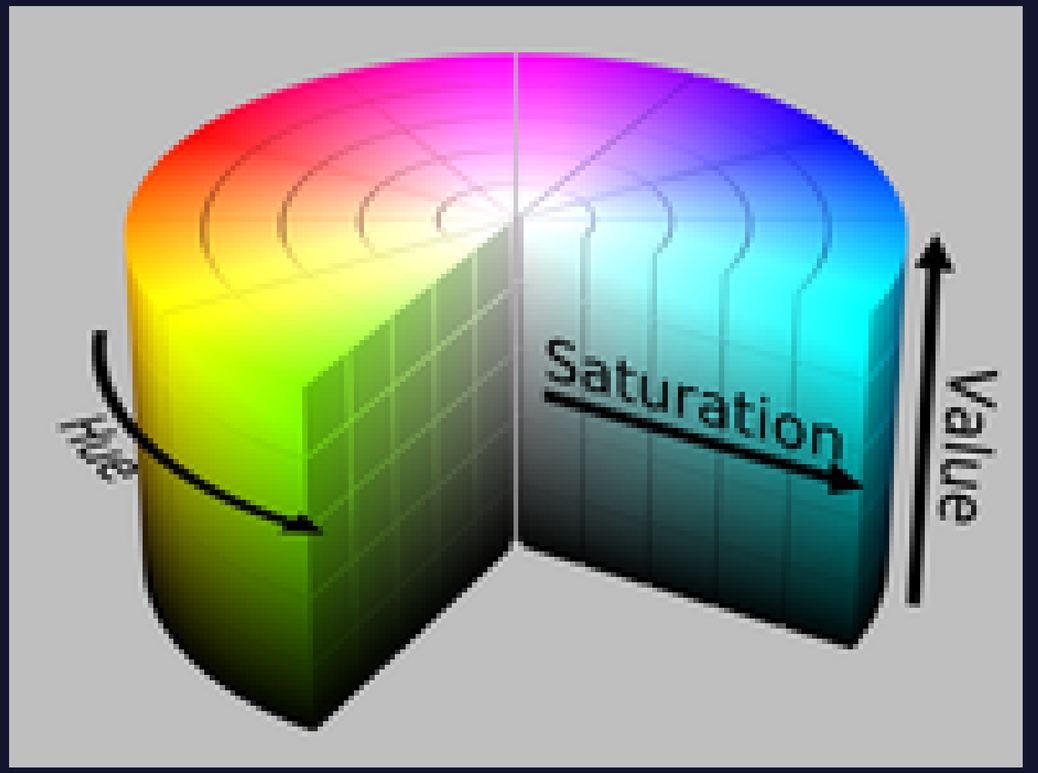


IMAGE PRE PROCESSING

- We perform some pre-processing steps on the input images.
    1. We resize images such that all images have the same size.
    2. We then convert the images from matrix form to tensors.
    3. We normalise the images such that the mean of the data is 0 and standard deviation is 1.
    4. We also perform batch normalisation during the training.

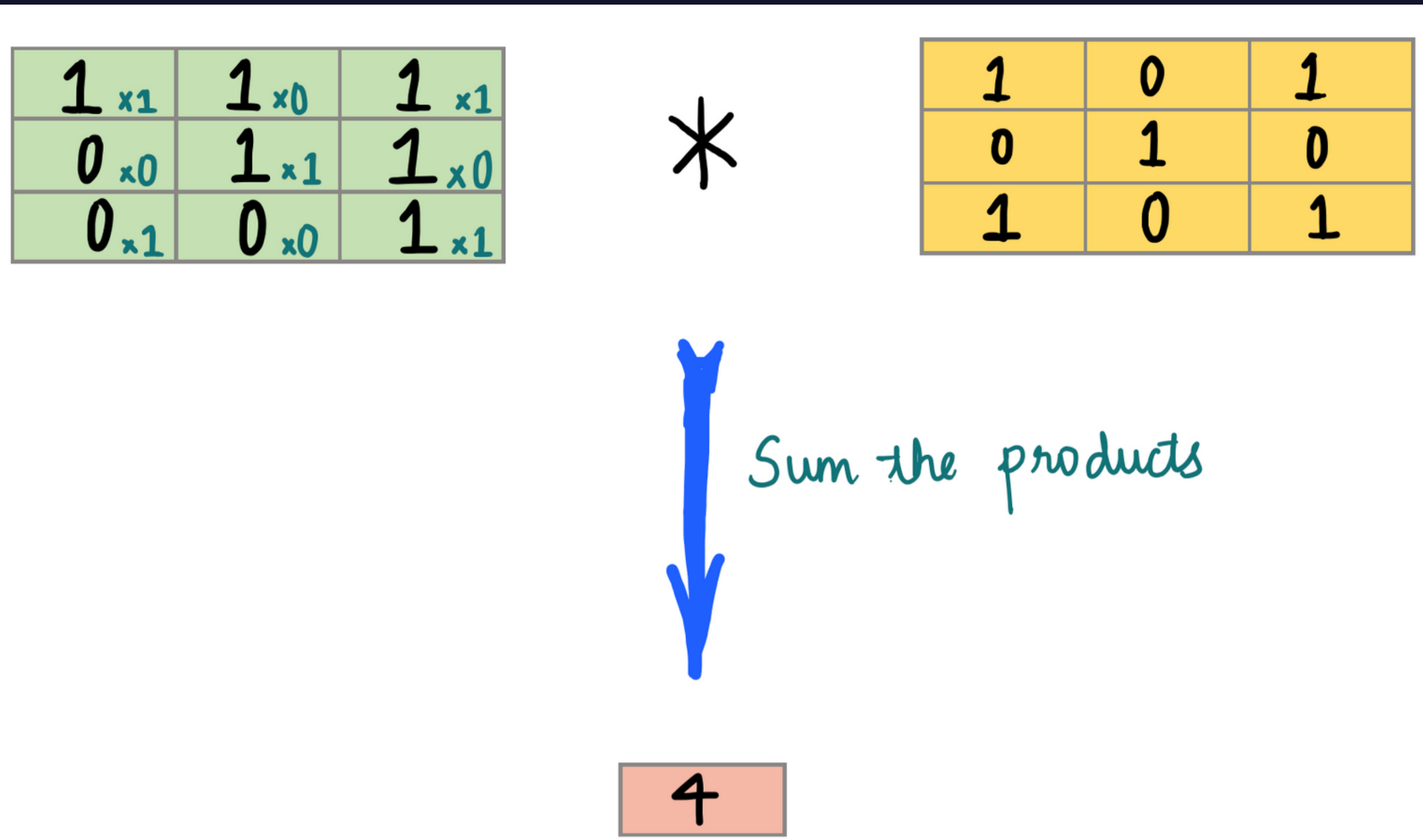
KERNELS AND FILTERS

- KERNELS:
  - A kernel is a matrix of weights that multiplied with the input matrix(image) to extract feature, a convolution.
  - The kernel shifts over the image and performs the convolution operation.
  - A kernel can also have a stride associated with it.
  - Stride is the number of pixels the kernel shifts over the input matrix in one step.
- FILTERS:
  - A collection of kernels is called a filter.

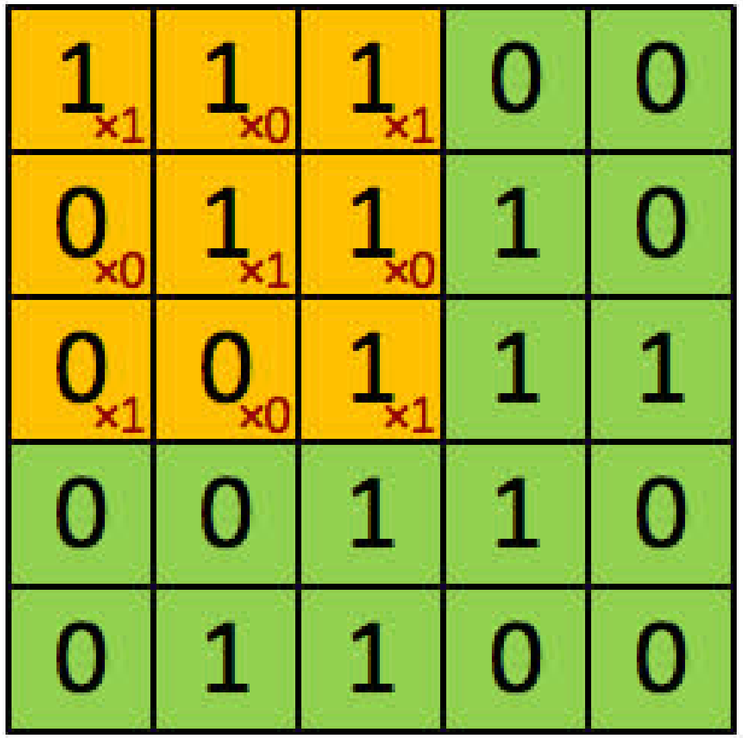
- OBJECTIVE OF CONVOLUTION OPERATION:
  - The role of ConvNets is to reduce large images into easier-to-process forms without losing important features.
  - This is the objective of the convolutional operation, to extract features from the input image, such as edges and colours.
- PARAMETERS INVOLVED IN A CONVOLUTION:

Kernel size, number of input channels, number of output channels and stride.

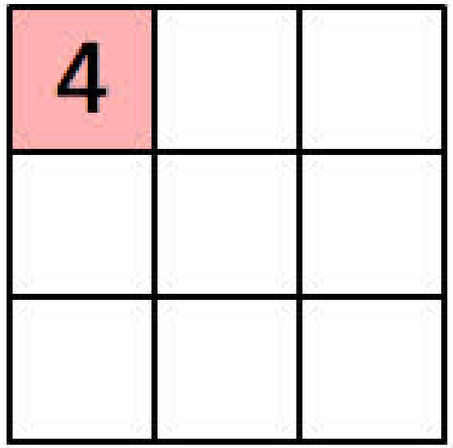
How to perform kernel operations?



How to perform kernel operations?



Image

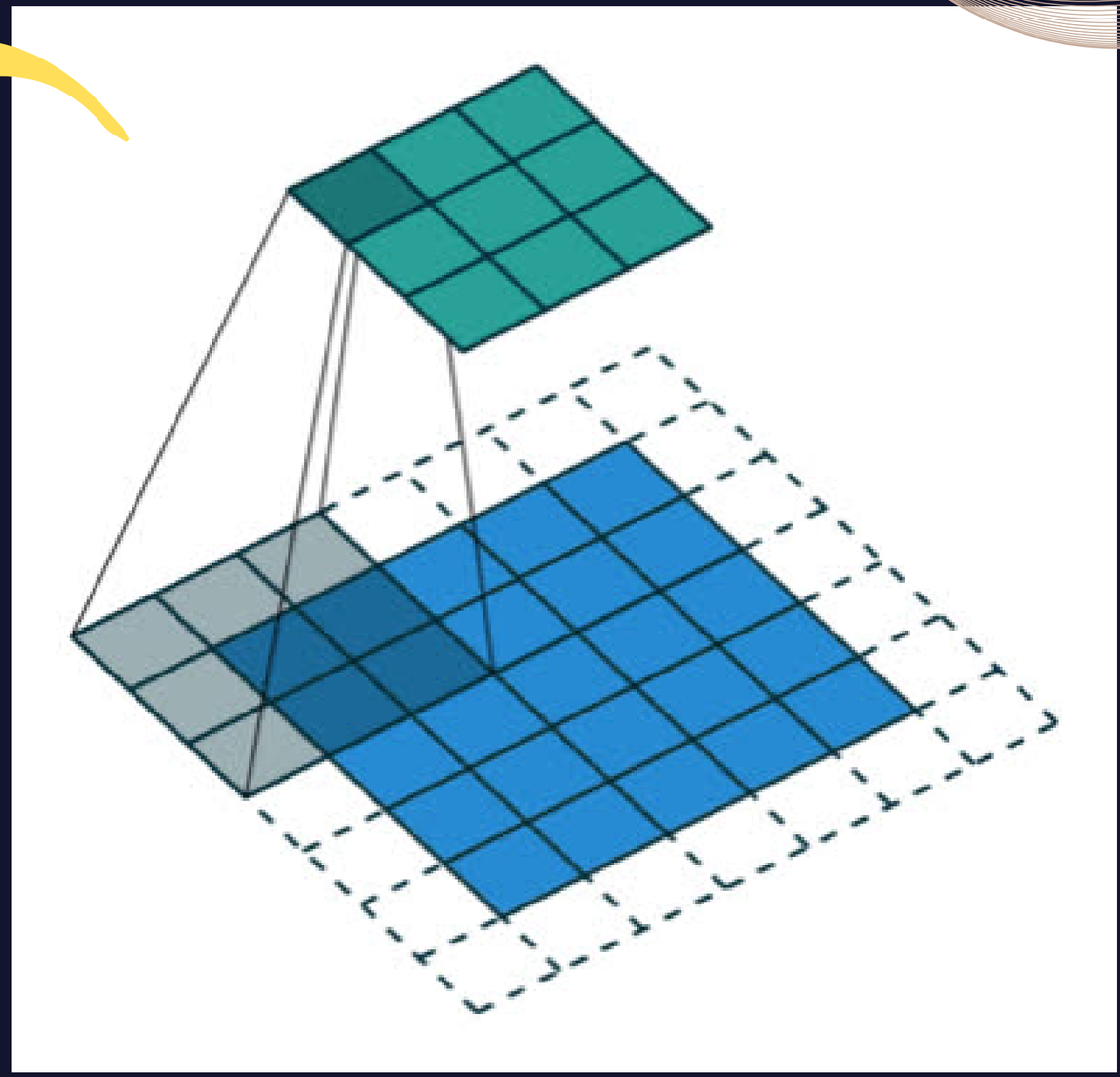


Convolved Feature

- Choose a submatrix whose size is same as that of the kernel.
- Perform a element-wise multiplicationon that submatrix.
- Now, shift the window by the stride and continue this process until the entire input matrix is traversed.

Here, the stride is 1

How to perform kernel operations?



Here, the stride is 2

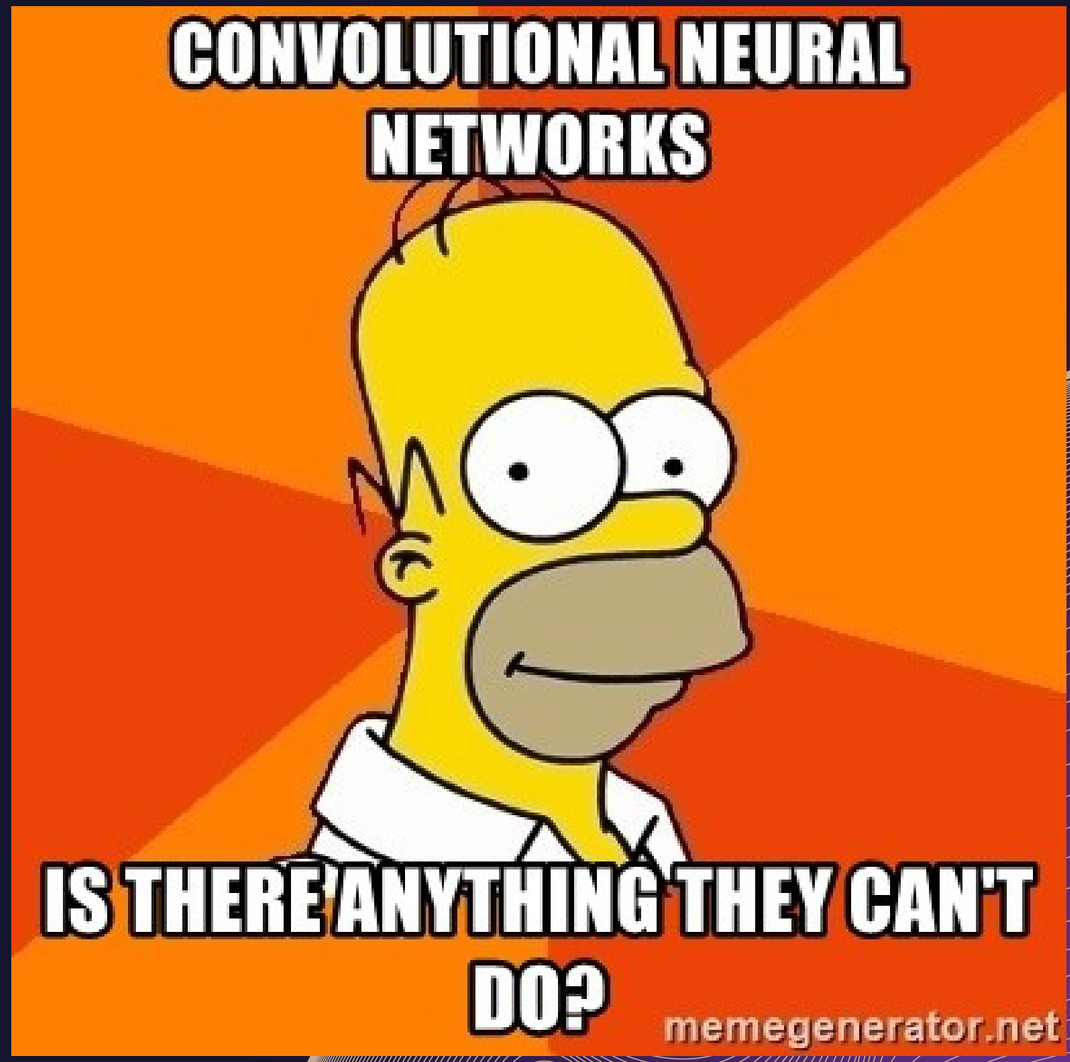
WHAT IS THE OUTPUT SIZE OF A CNN?

Suppose we have input image of size n x n and we perform convolution using kernels of size k x k, with stride s and padding p. Then the output will be of size

$$(n+2p-k)/s + 1$$

HOW DOES A CNN REDUCE NUMBER OF PARAMETERS?

- Suppose we had a input image of size 8x8 and 32 channels that we wish to convolve using 3x3 kernels and have 64 output channels.
- So, we need 64 kernels of size 3*3*32.
- So this convolutional layer will have 18432 learnable parameters.
- If we did this using a fully connected layer, then we will have 8*8*32 inputs and 6*6*64 output neurons.
- So we will have about 4.72 million learnable parameters.
- This drastic reduction in the number of neurons is because we are reusing the kernels over the entire image instead of having individual weights for each pixel.
- This allows faster training times and allows training on large data sets.



PADDING IN CONVOLUTIONAL NEURAL NETWORKS

WHAT HAPPENS WHEN WE PERFORM CONVOLUTIONS?

SIZE REDUCTION

- When we perform a kernel operation with a kernel of size f x f, on an image of size n x n, the output image has size (n-f+1) x (n-f+1).
- So, there is a reduction in the size of the image and this restricts us in the depth of the model that we can build.

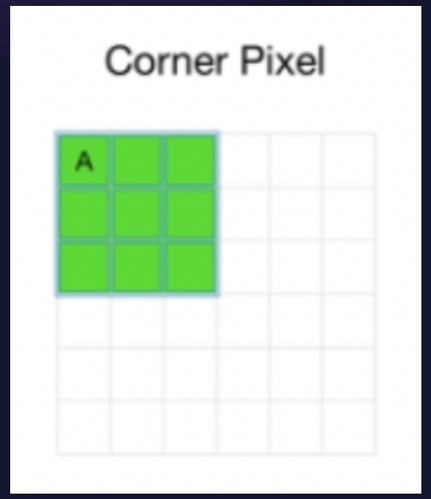
LOSS OF INFORMATION FROM EDGES

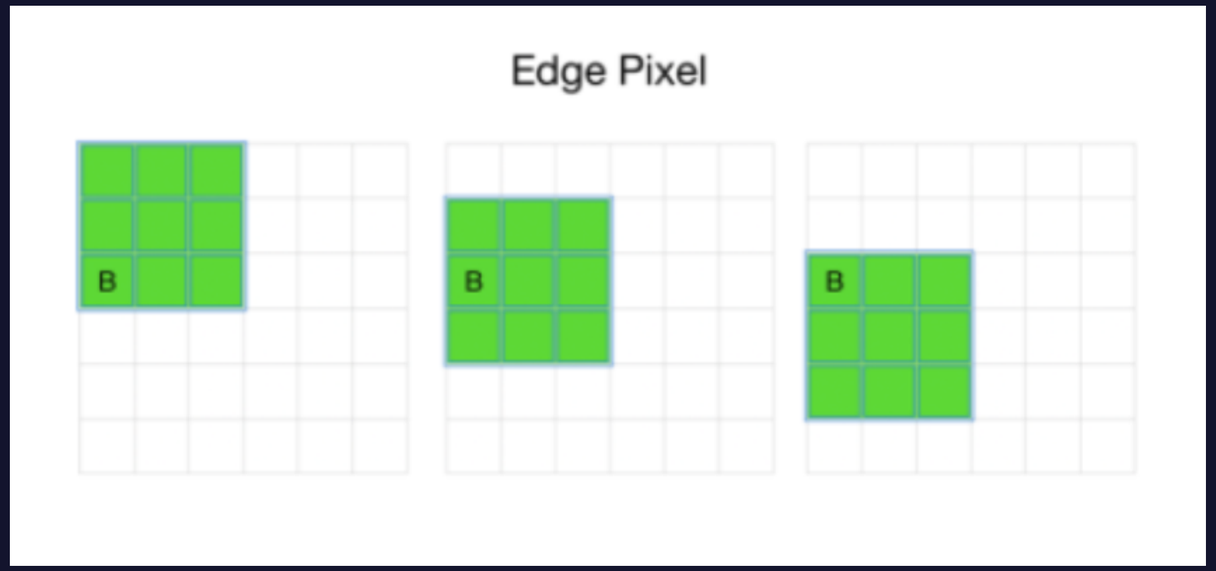
- When we perform a kernel operation the pixels at the edge of an image is not covered as much as a pixel that is not on the edge.
- For example, suppose a kernel of size 3 x 3.
- Then a corner pixel will be covered only once and an edge pixel will be covered 3 times.
- But a pixel in the middle will be covered 9 times.
- This leads to a loss of information at the edges

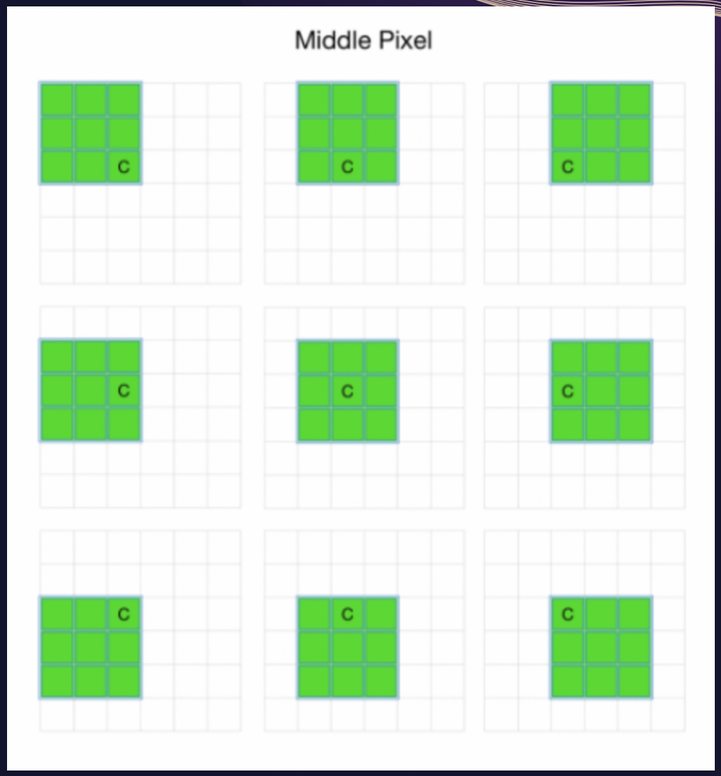
WE SOLVE THESE ISSUES USING PADDING

PADDING IN CONVOLUTIONAL NEURAL NETWORKS

WHAT HAPPENS WHEN WE PERFORM CONVOLUTIONS?





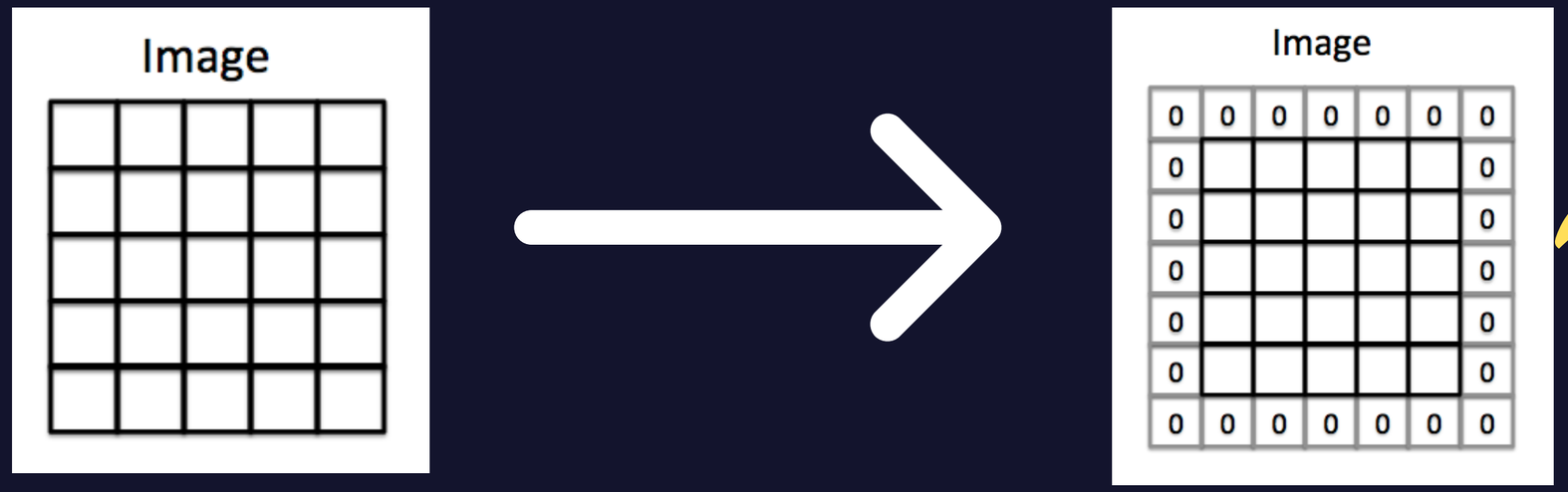


WE SOLVE THESE ISSUES USING PADDING

PADDING IN CONVOLUTIONAL NEURAL NETWORKS

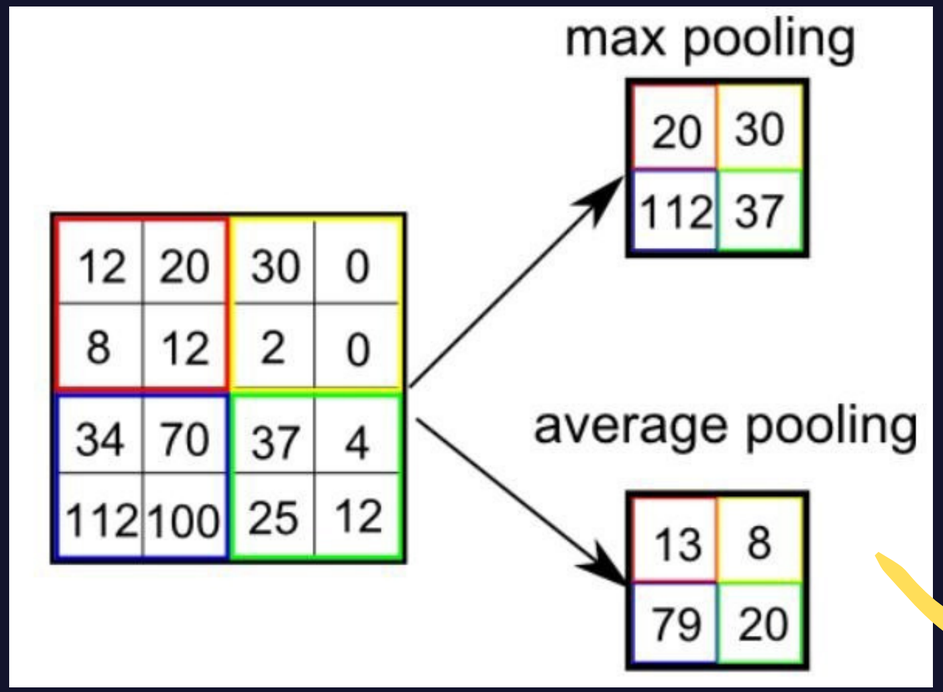
- In order to increase size of the image and to save the information present at the corners, we add padding layers to the image.
- Padding means to add an extra layer of pixels to the outer layer of the image.
- There are primarily two types of padding:

1. Valid Padding: There is no padding and the model uses only "valid" data.
2. Same Padding: We pad one extra layer of pixels to the image. So the output image has the same size as the input.
3. Full Padding: We pad such that each pixel is covered the same number of times by the convolution.



POOLING LAYERS

- We add pooling layers when we want to reduce the size of the image.
- These layers capture the dominant features of the image.
- There are two types of pooling: Maximum Pooling and Average Pooling.
- Maximum pooling returns the maximum value from the window while Average pooling returns the average of the values from the window.



Here, the window size is 2

BATCH NORMALISATION

- When we train the model, due to the large size of our model, the weights of some of the neurons can become too large.
- The gradients for back-propagation are calculated assuming that the weights are constant.
- But the weights change rapidly and hence the model is trying to chase a moving target.
- This is called Internal Covariate Shift, which is more formally defined as "the change in the distribution of network activations due to the change in network parameters during training".
- This will slow down the process of back propagation and hence increase training time.
- Batch Normalisation is used to solve this problem.
- We normalise the input to each layer(before or after activation) based on the following algorithm.

BATCH NORMALISATION

- Here, γ and β are learnable parameters that will get adjusted by the gradient descent algorithm.
- Note that now the mean of the input is β and the standard deviation of the input is γ.
- Batch Normalisation adds noise to the data and hence prevents overfitting.
- It also increases training efficiency.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

RELU AND FULLY CONNECTED LAYERS

- We use the ReLU activation function in CNNs so as to introduce non-linearity.
- We finally then flatten the output and feed it into a fully connected neural network to obtain classes and perform classification task.

IMPLEMENTATION USING PYTORCH

We used the torch.nn module for implementing a convolutional layer.

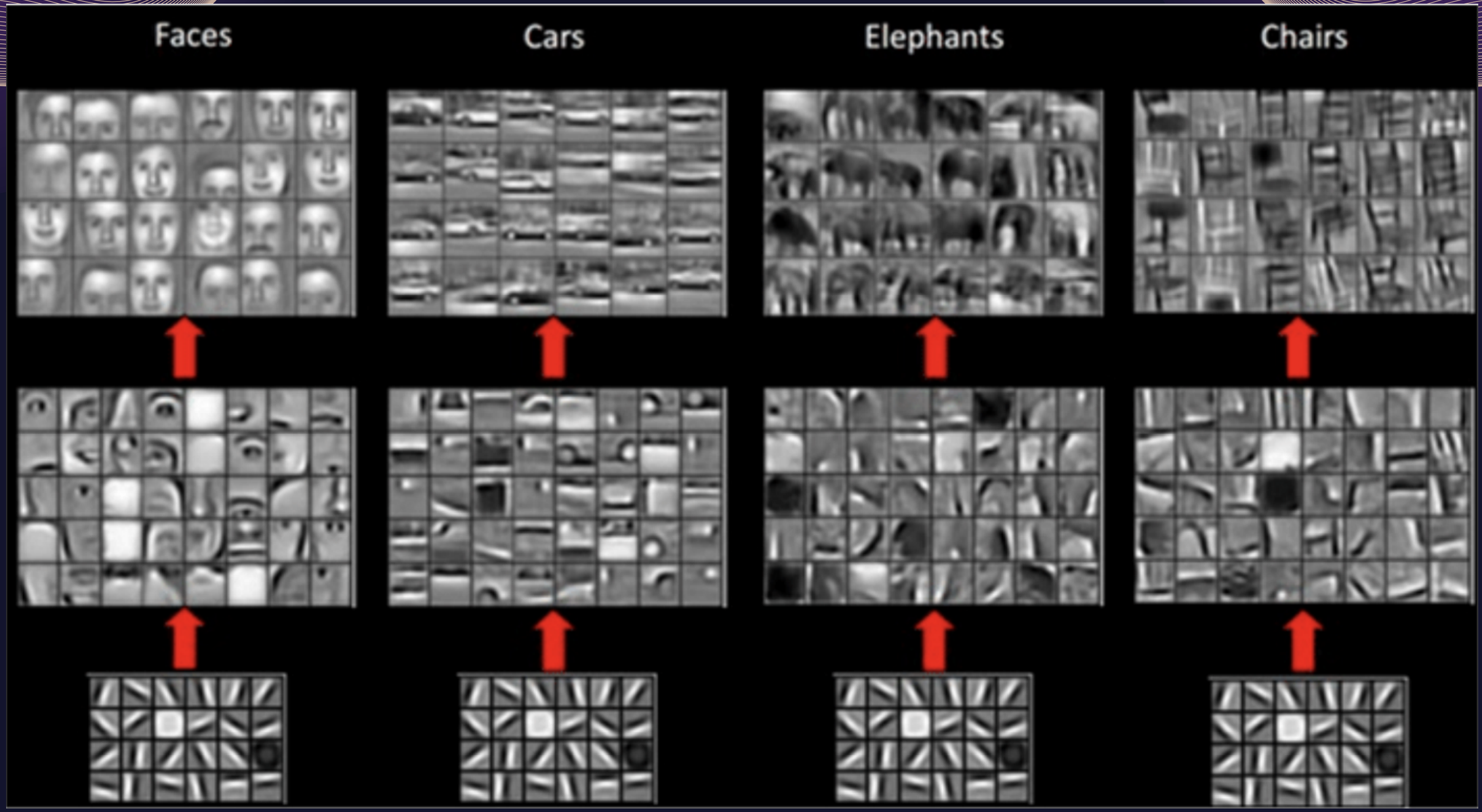
```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0,
dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)
```

- in_channels = number of input channels
- out_channels = number of output channels
- padding = 1 means we pad a layer of zeros around the image while padding = 0 means we do not pad the image.

WHAT DO CNN LAYERS LEARN?

- Each CNN layer learns filters are of increasing complexity.
- The initial layers learn basic feature detection filters like edge and corner detection.
- The middle layers learn filters that detect parts of objects.
- For faces for example, they might learn to detect eyes and noses.
- The last layers have higher representations.
- They learn to recognize full objects, in different shapes and positions.

WHAT DO CNN LAYERS LEARN?

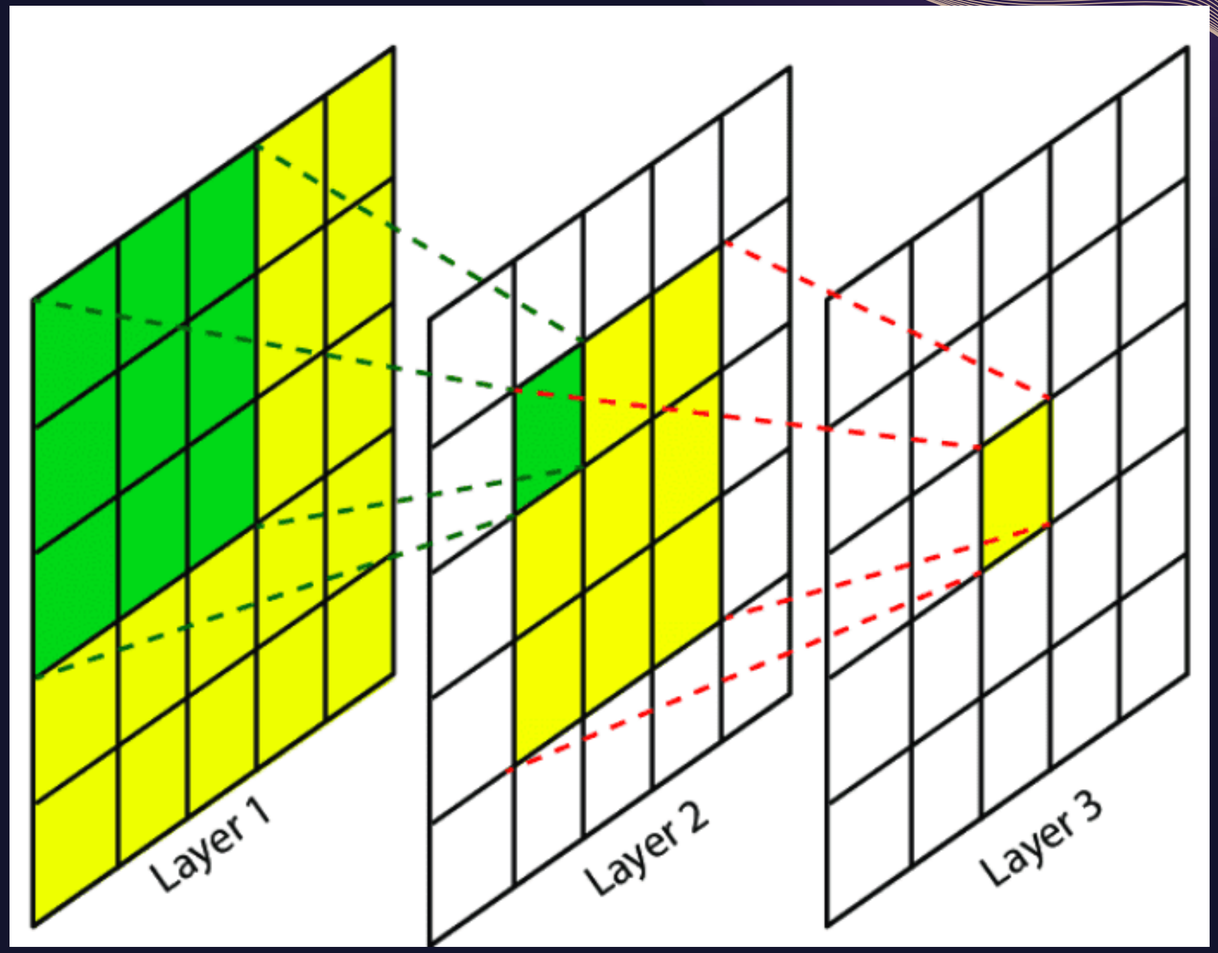


RECEPTIVE FIELDS

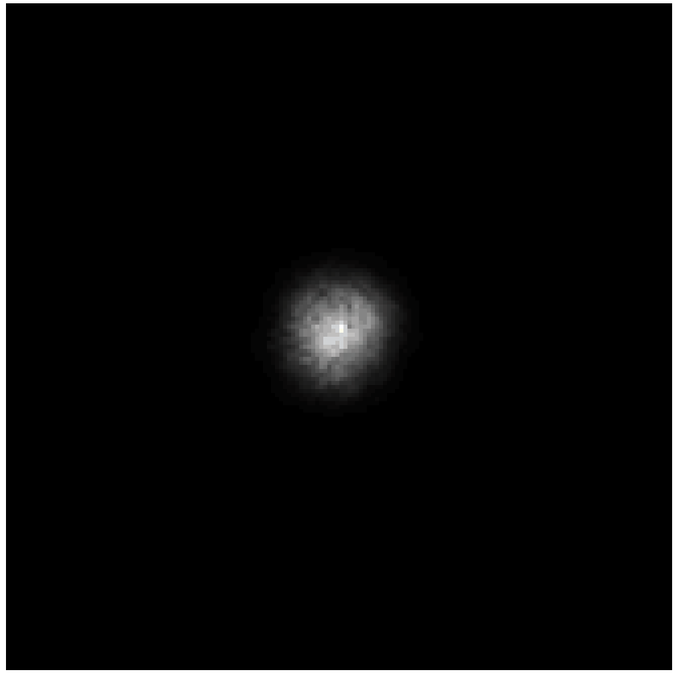
- For a given neuron/feature, the part of the input image that affects the given neuron.
- Receptive fields are important because they determine the model's ability to detect objects.
- If the receptive field is too small, then large objects will not be detected.

HOW TO INCREASE RECEPTIVE FIELD?

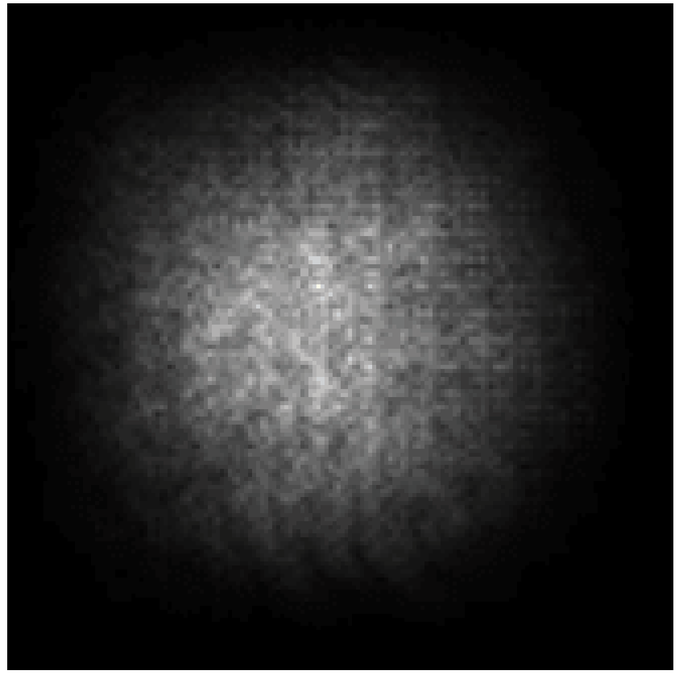
- Add more convolutional layers. Receptive field increases linearly.
- Increase stride.
- Add pooling layers.
- Perform dilated convolutions.



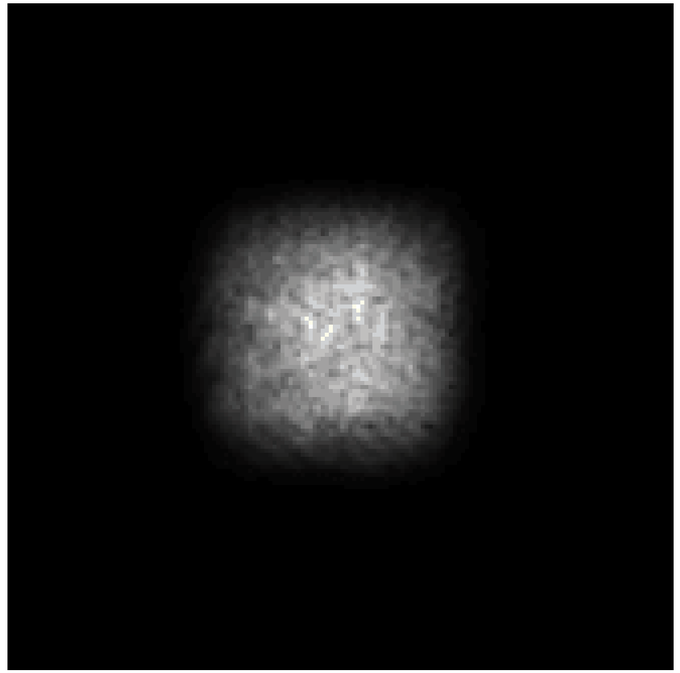
RECEPTIVE FIELDS



Conv-Only



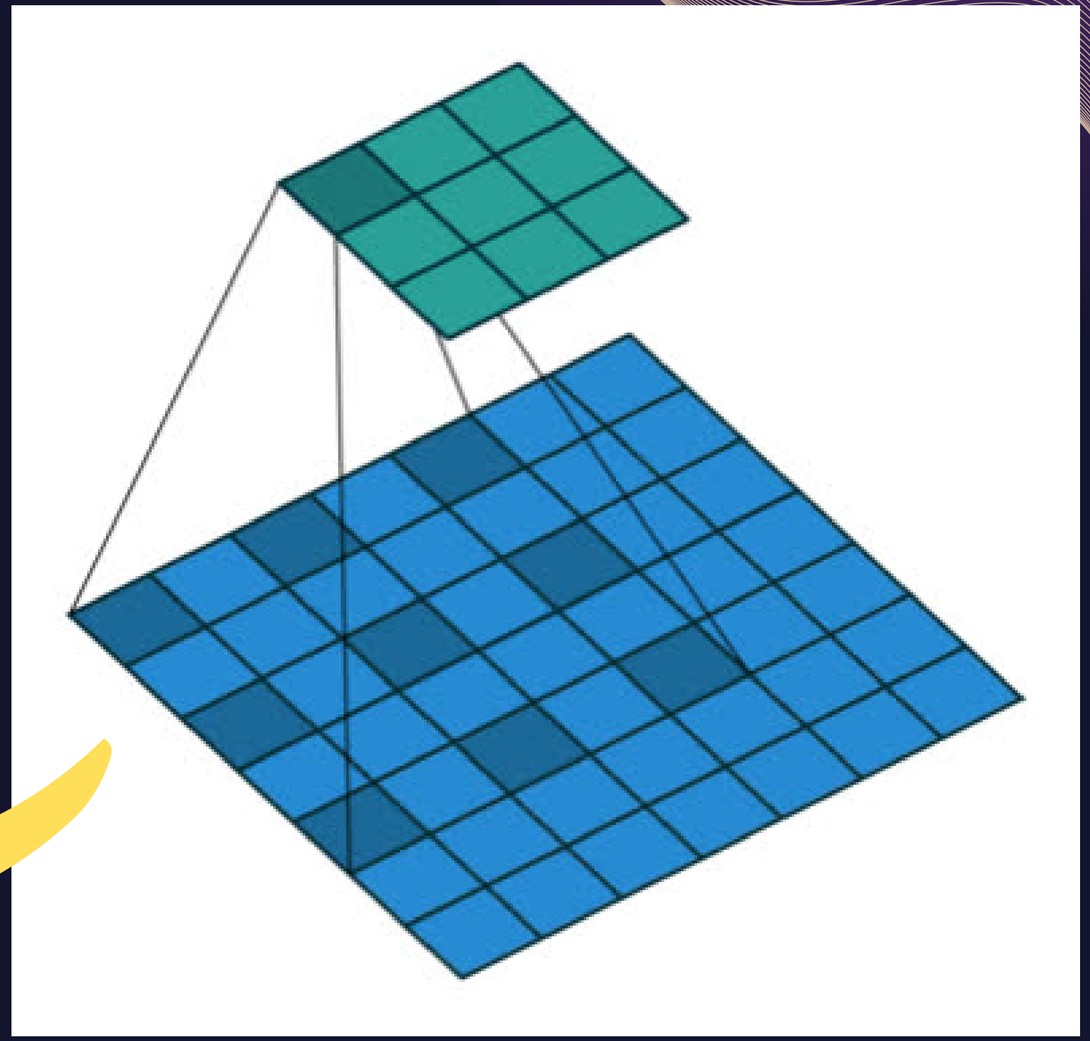
Subsample



Dilation

DILATED CONVOLUTIONS

- Dilations introduces "holes" in the kernels.
- So, there is a space between the elements of the kernel.
- A dilation of 1 is a normal convolutional operation and the kernel elements are consecutive.
- On the right is a convolution with dilation 2.



CNN ARCHITECTURE: VGG

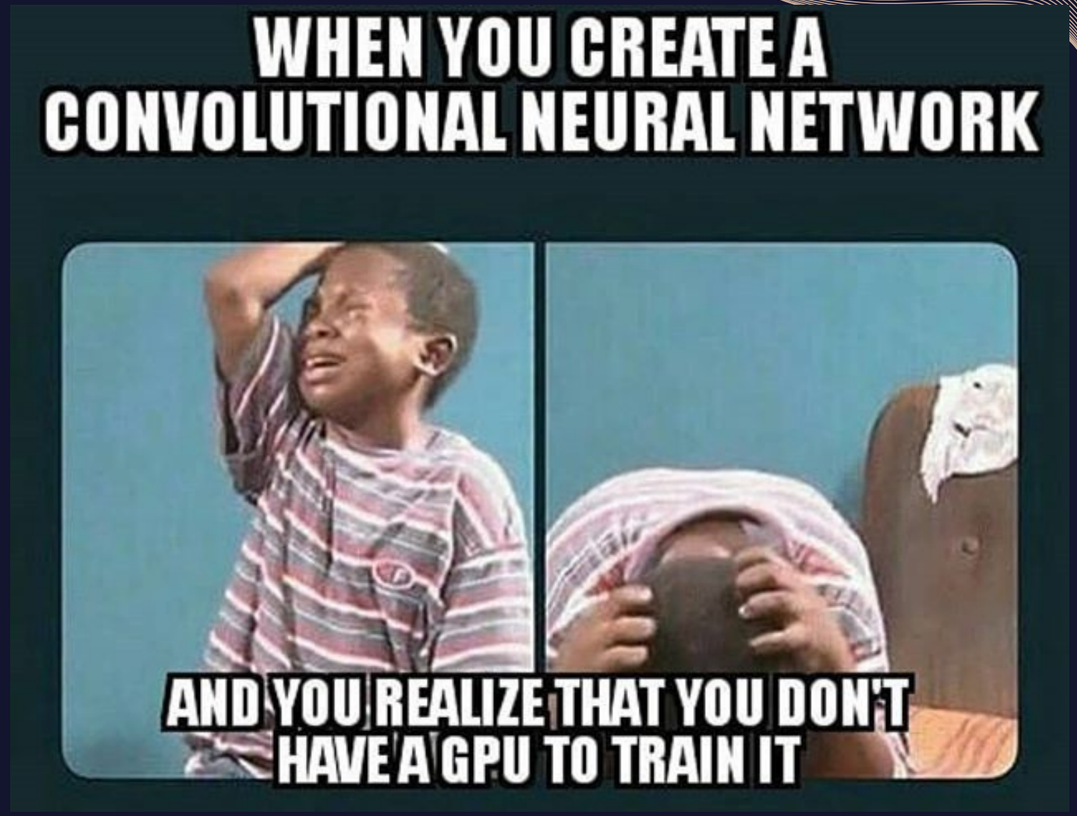
- There are several CNN based model architectures like LeNet, AlexNet, GoogleNet, VGGNet and ResNet.
- One of the most popular and effective models is the VGG(Visual Geometry Group) model.
- It is seen that deeper models perform better as they are able to capture more complex features present in the images.
- The largest VGG model is VGG19, which has 16 convolutional layers followed by 3 fully connected neural layers.

VGG ARCHITECTURES

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

HOW DO WE ACCELERATE TRAINING?

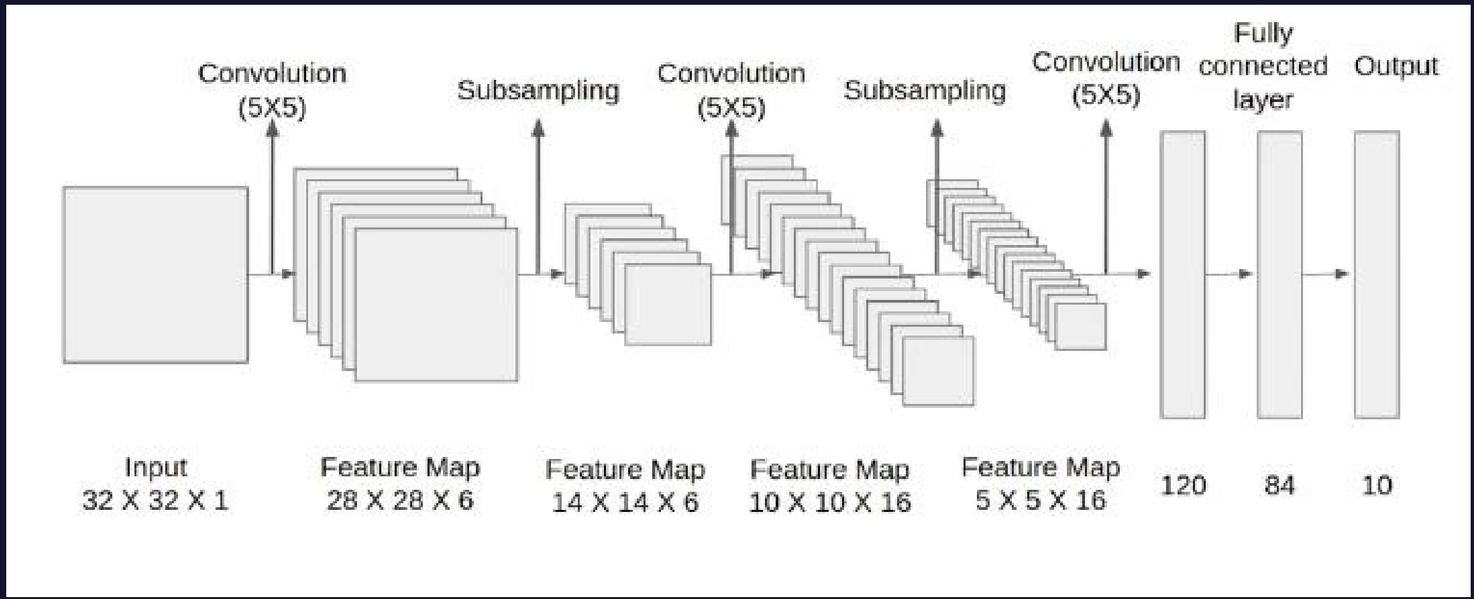
- The VGG19 model has 144 million learnable parameters.
- Training such robust models can take a lot of time even on extremely powerful CPUs.
- In order to accelerate the training process, we use Graphical Processing Unit or GPUs.
- PyTorch comes with a inbuilt functionality called CUDA, which allows us to use GPUs for training.



MINI-TASK

Try to implement another architecture, LeNet-5, which is used to recognise hand written digits.

Use the MNIST data set for training and testing which is a collection of 60,000 gray scale images of hand written digits, each of size 28 x 28.



SOME STUFF TO THINK ABOUT

1. What will happen if we keep increasing the depth of the model? Will its accuracy increase?
2. Is padding with zeroes the only possible form of padding? Are there any other methods of padding used?(See that nn.Conv2d has a parameter called padding_mode)


COMPUTER VISION PROBLEMS
ADD A CONVOLUTIONAL NEURAL NETWORK

THANK YOU