Convolutional Neural Network

Edge Detectors

Convolution

- The main building block of any CNN
- An operation to merge 2 sets of information
- Produces a "feature map"

Convolution

Convolution is performed using a convolutional filter or kernel on an input image.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

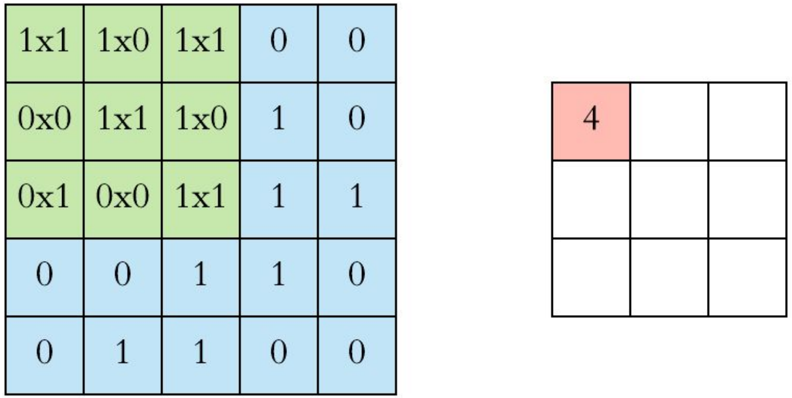
1	0	1
0	1	0
1	0	1

Filter / Kernel

This is a 3x3 convolution due to the filter shape.

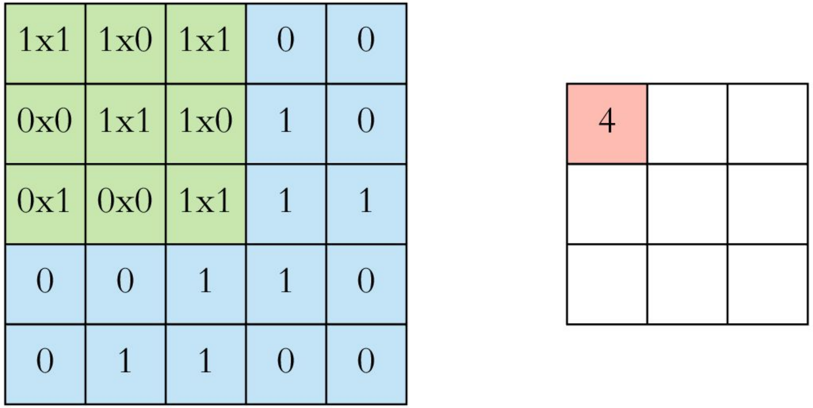
Convolution

- The results from convolution operations are aggregated into a feature map
- The kernel filter values **are the parameters** that are learned.



Convolution

- The operation is performed by sliding the filter over the input.
- Element wise multiplication and summation at every location
- The Green area is called the receptive field.



Input x Filter

Feature Map

Convolution

After sliding once

1	1x1	1x0	0x1	0
0	1x0	1x1	1x0	0
0	0x1	1x0	1x1	1
0	0	1	1	0
0	1	1	0	0

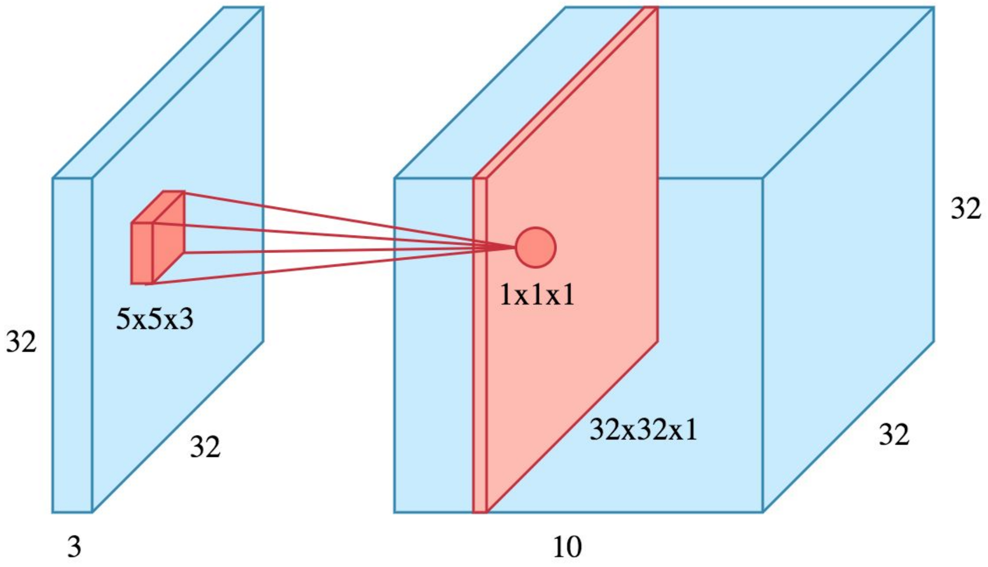
Input x Filter

4	3	

Feature Map

Convolution

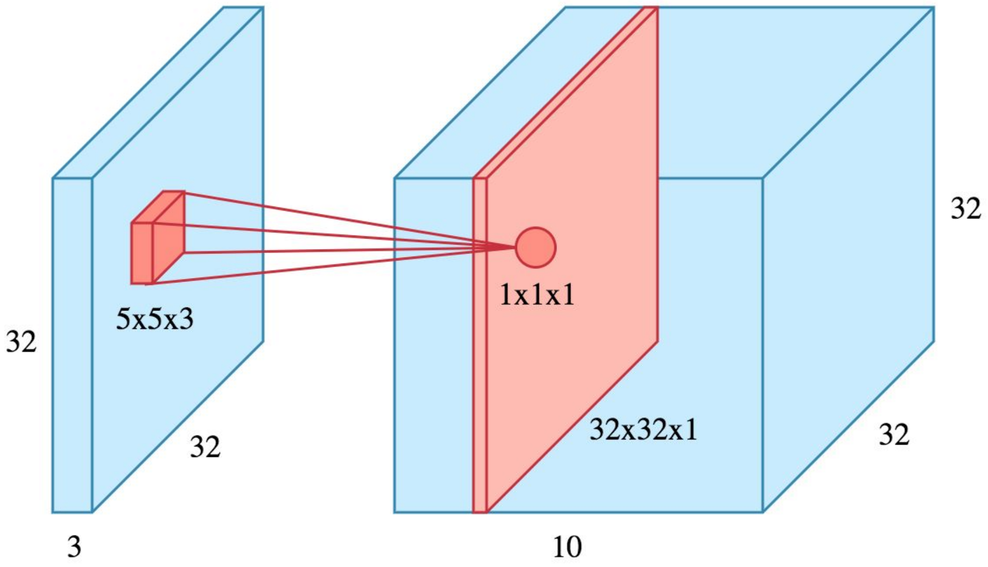
- In reality convolutions are performed in 3D.



DICE ANALYTICS

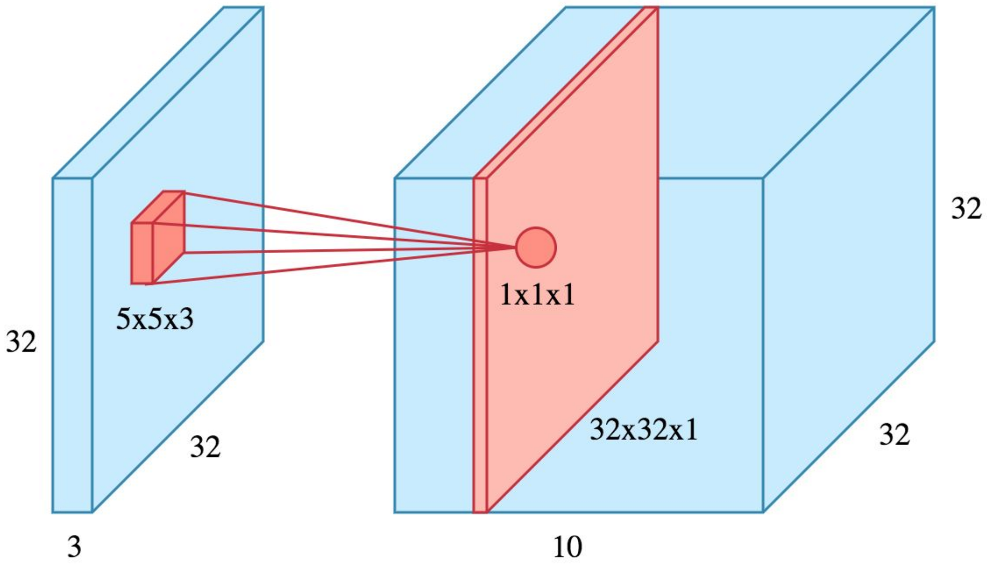
Convolution

- Multiple convolutions on an input generating distinct feature maps.
- All feature maps are stacked together to give the output of a convolution layer.



Convolution

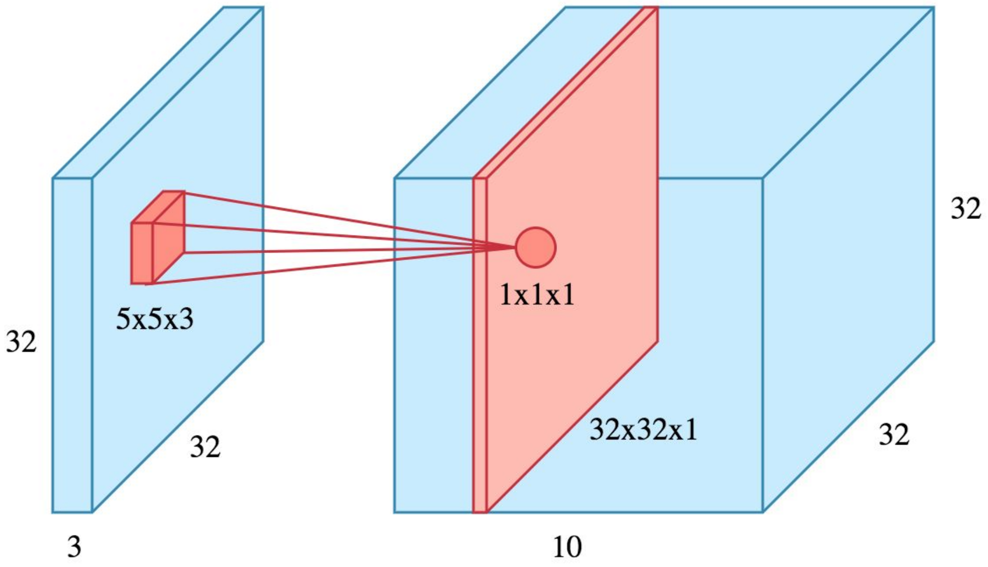
- Here, the input is a 32x32x3 (RGB) image
- The convolutional filter is 5x5x3 (matching the image depth)



DICE ANALYTICS

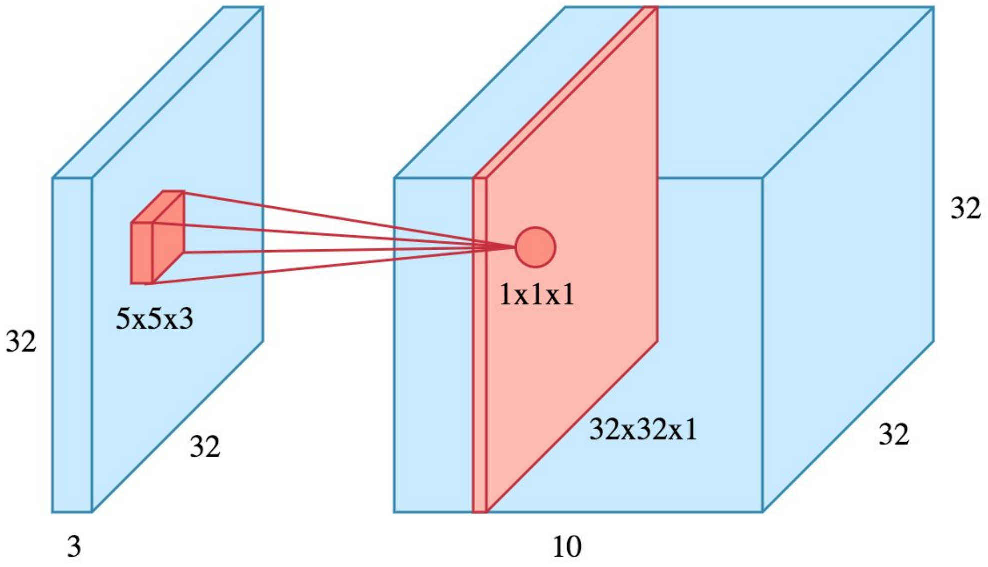
Convolution

- Here, the input is a 32x32x3 (RGB) image
- The convolutional filter is 5x5x3 (matching the image depth)
- The feature map is 32x32x1 (3D convolution also gives a scalar)



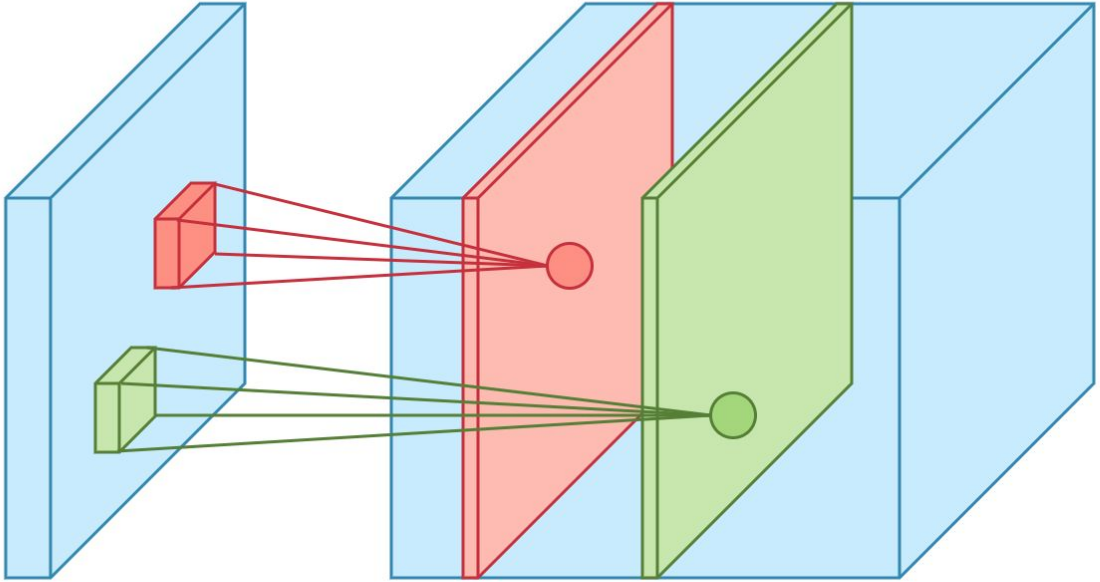
Convolution

- Here 10 different filters are used, resulting in 10 feature maps.
- Stacking these filter maps we get the final output of 32x32x10
- The feature map size is same as the image due to “padding”.



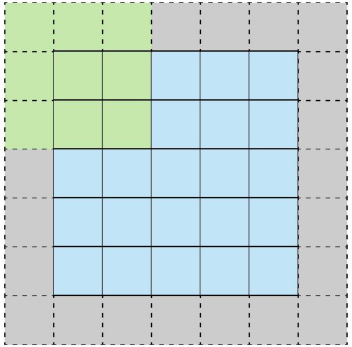
Convolution

Two different filters produce two different feature maps

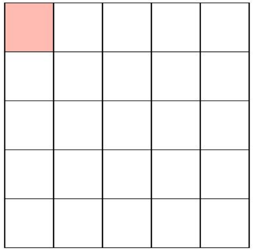


Padding

- The size of the feature map is smaller than the original input if the convolutions are contained in the input.
- Information in corner values is convolved only once without padding
- To maintain the dimensionality, padding is used.
  - The input is surrounded by zeros or edge values



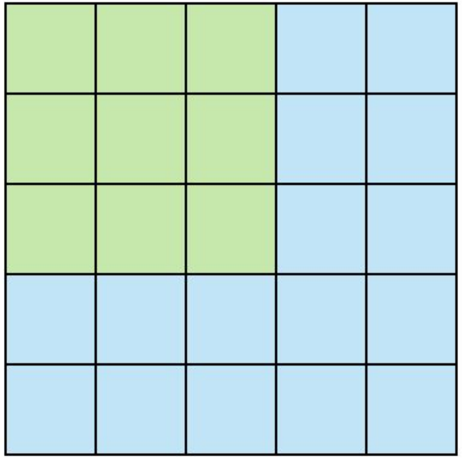
Stride 1 with Padding



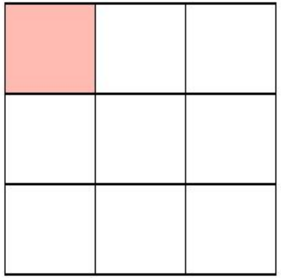
Feature Map

Stride

- An important hyper-parameter of the convolution layer.
- It specifies how much the convolution filter is to be moved at each step.
- By default, it is taken as 1.



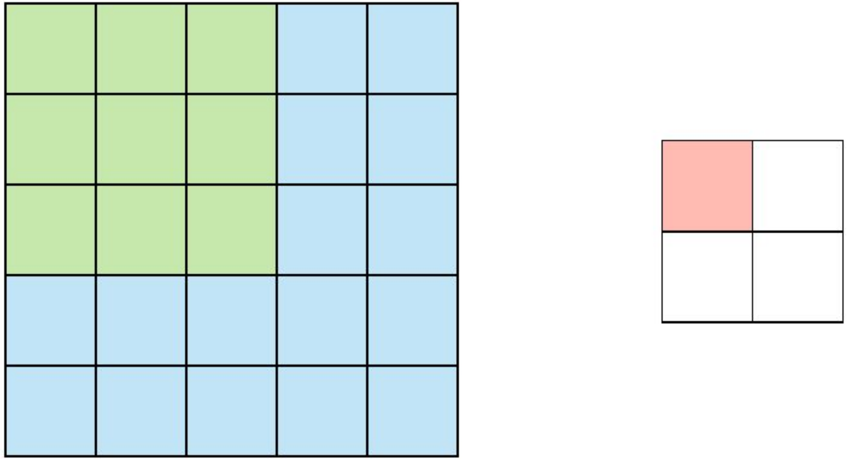
Stride 1



Feature Map

Stride

- Bigger strides can be taken to lessen the overlap between receptive fields.
- This makes the feature map smaller.



Stride 2 Feature Map

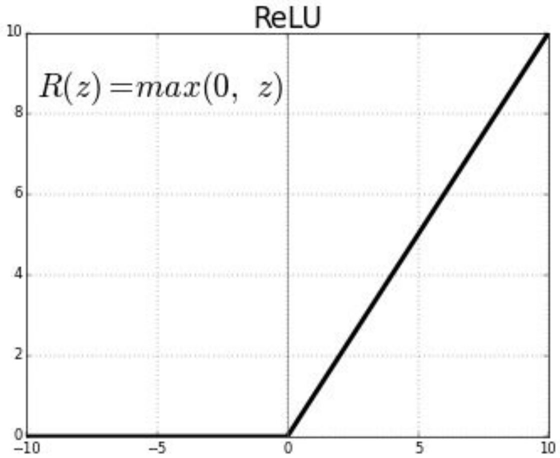
Feature map size

Stride and padding together with the size of filter decide the size of the feature map:

Dimension of output features = {(Dimension of input features + 2*padding – filter size) / stride size} +1

Non-linearity

- Non-linearities make a neural network powerful.
- The most commonly used non-linearity in CNN’s is the ReLU activation function.
  - All the negative values in feature maps are changed to zero.

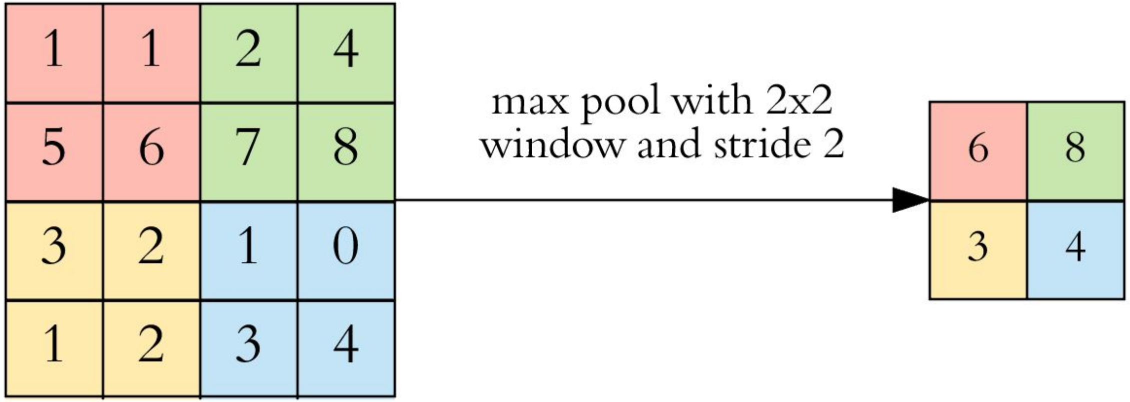


Pooling

- Pooling is performed after convolution to reduce dimensionality.
- The main use is to reduce the number of parameters (kernel filter weights)
  - This reduces both training time and overfitting.
- Pooling layers downsample each feature map independently meaning that the depth (number of feature maps) remains the same.
- The most common type of pooling is the max pooling
  - Taking the maximum value out of a “pooling window”
- Avg. pooling is another example
- Pooling layers have no parameters to train.

Pooling

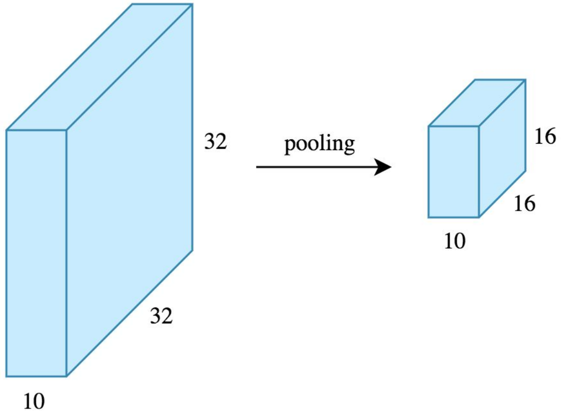
For example



Note that this window and stride configuration halves the size of the feature map while keeping the important information intact.

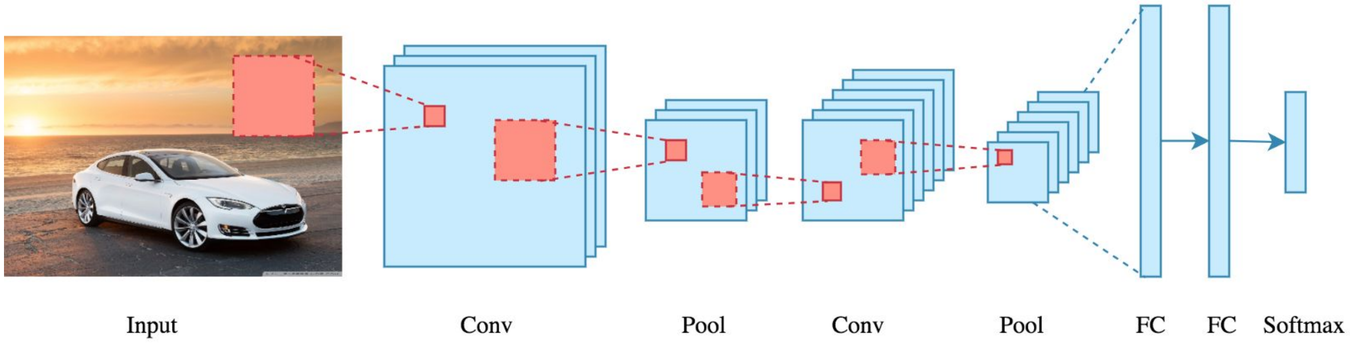
Pooling

- Using the same pooling window and stride sizes in a real use case
- By halving the height and the width, the number of weights in the next layer are reduced to 1/4 of the input.
- This reduction is quite significant considering the number of parameters.

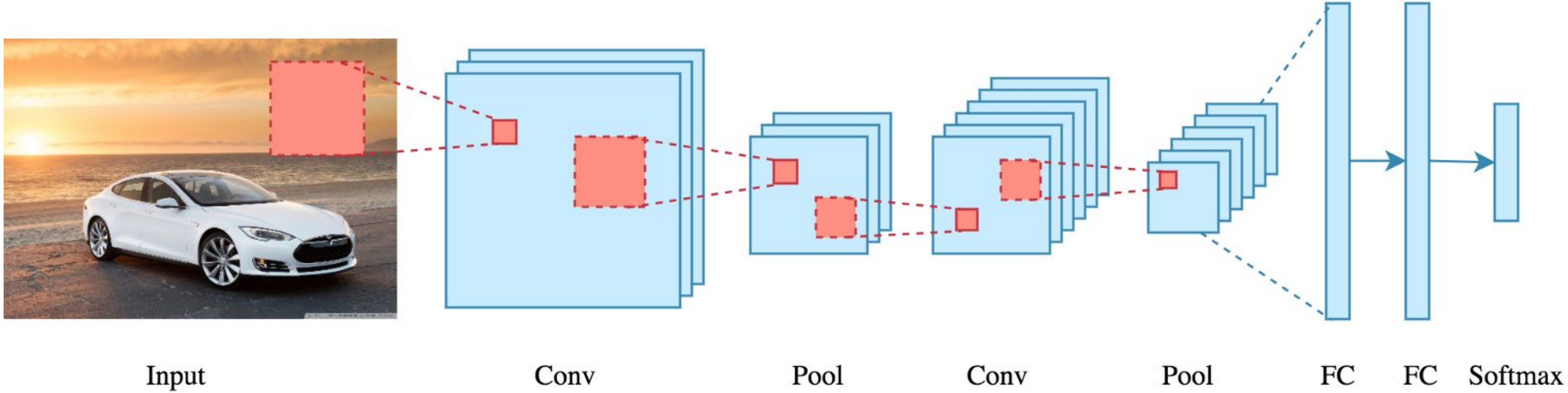


Fully connected layers

- The primary task of the convolution and pooling layers is to extract different features from input images.
- Once these features have been extracted, their respective presence can give an insight about what is in those images.
- The last 3D layer is arranged in a 1D form sequentially and connected to a simple feed-forward neural network through its weights.



Architecture

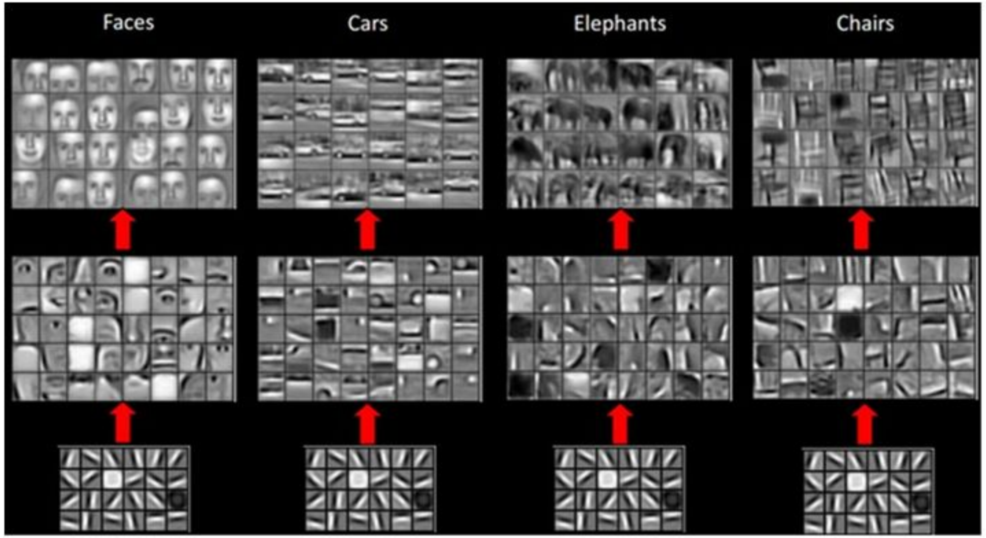


Hyper-parameters (review)

- There are six different architectural hyper-parameters, 4 linked with convolution layers and 2 with pooling layers.
- In convolution layer:
  - Filter size (e.g. 3x3)
  - Filter count
  - Stride
  - Padding
- In pooling layer:
  - Pooling window size
  - Pooling stride

Working

- Convolution layers are the main power house of any CNN model
- These layers learn complex features by building on top of each other.
- For example features such as two eyes, long ears, four legs, a short tail, etc.
- The fully connected layers then act as a classifier on top of these features.



Why Do We Need CNNs

- Spatial significance of data
- Computation limitation with large images on Dense networks
- Parameter Sharing: Same filter can be used all across image
- Sparsity of Connections: Each output only depends on a small number of inputs