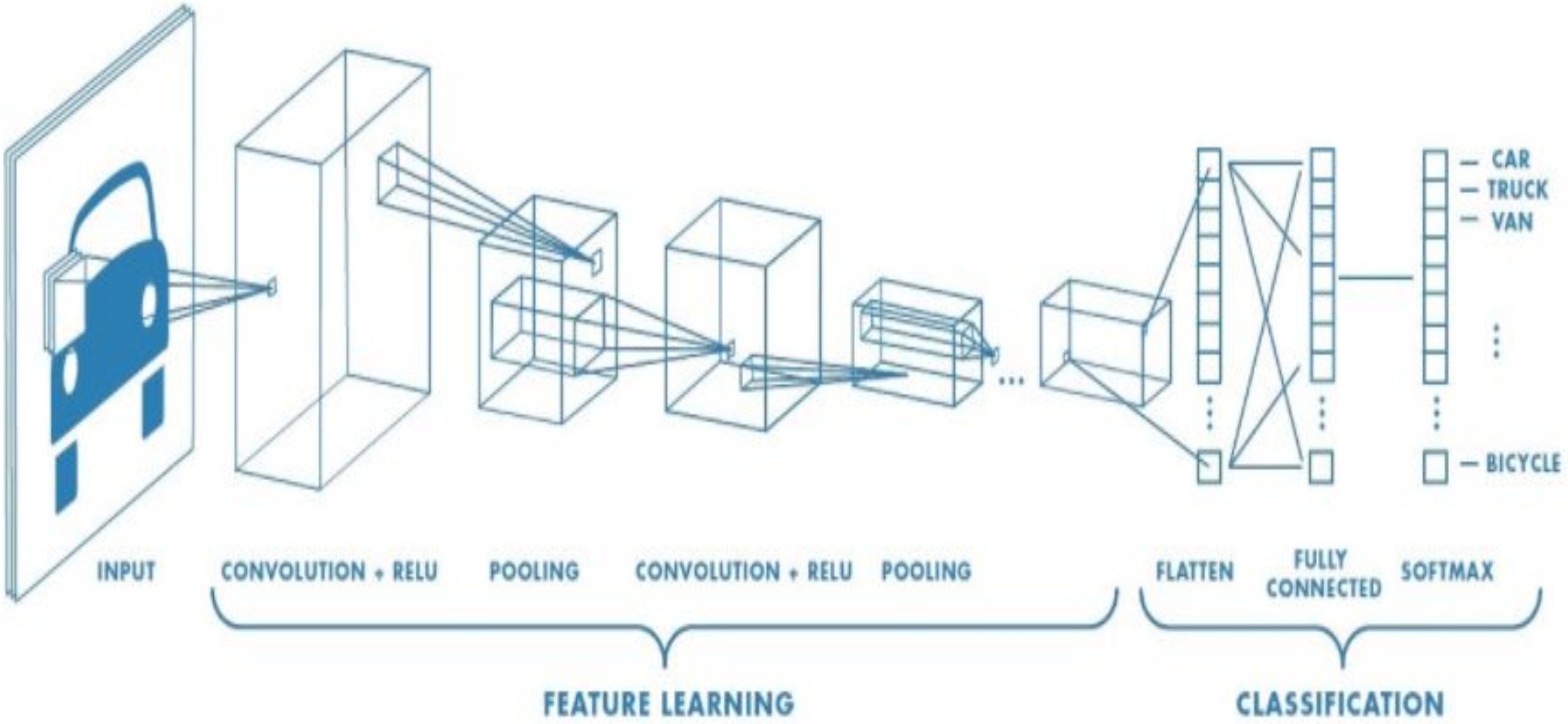
CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional neural network

- Also known as CNN or ConvNet
- A class of neural networks for processing data that has a grid-like topology, such as an images, videos etc.
- Used for object classification, object detection, face recognition etc.
- Humans use different layers of neurons for detecting different features of an image.
- Similarly, CNN uses multiple layers called **filters** on images to analyze image inputs.
- These layers are the **math layer**, **rectified linear unit layer**, and **fully connected layer**.
- These layers catches the pattern in the input, processes the data and deliver the output as an n-dimensional vector output



CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Math layer
(Convolution and pooling)

Convolution (filtering)

Filters are added using conv2D function

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

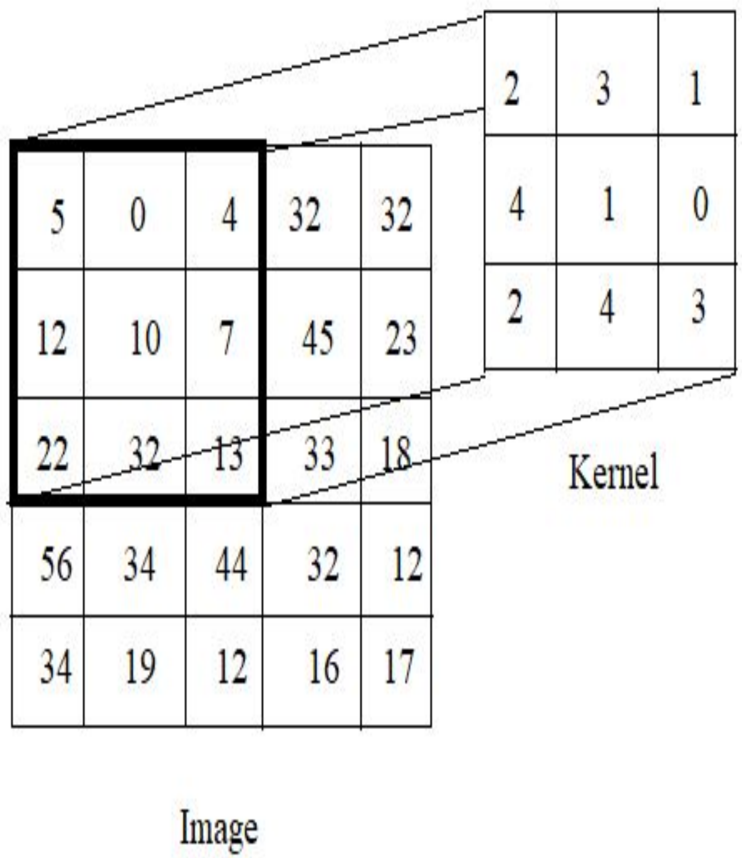
Image
5x5

2	3	1
4	1	0
2	4	3

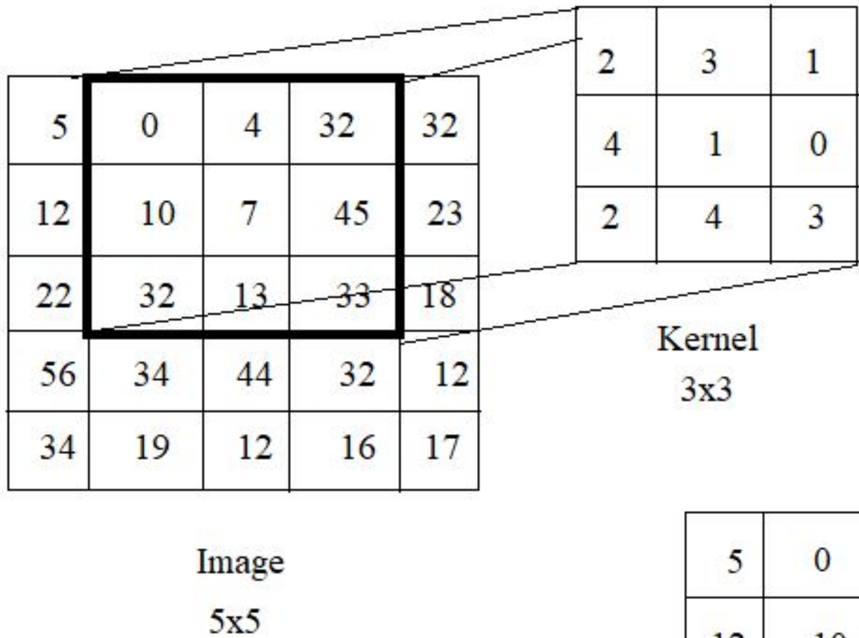
Kernel
3x3

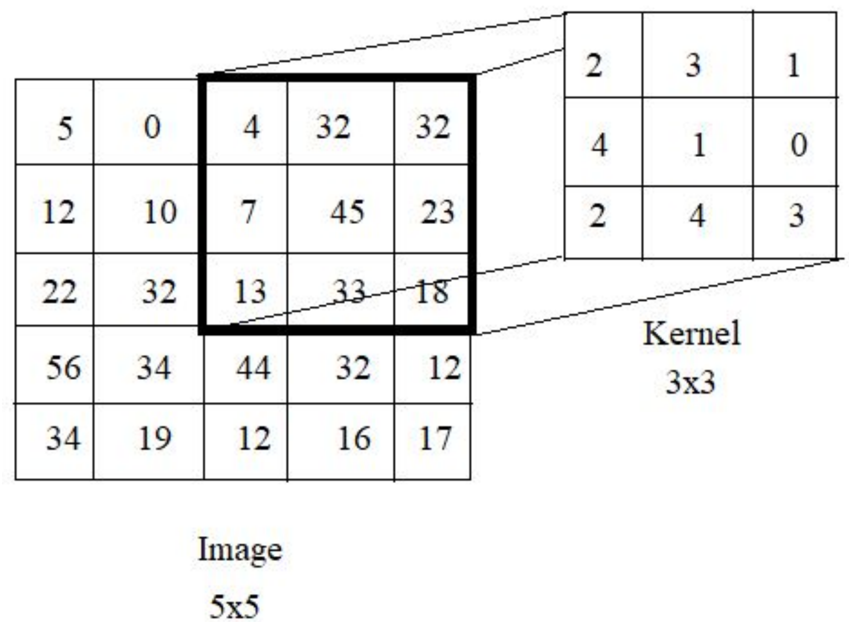
2	3	1
4	1	0
2	4	3

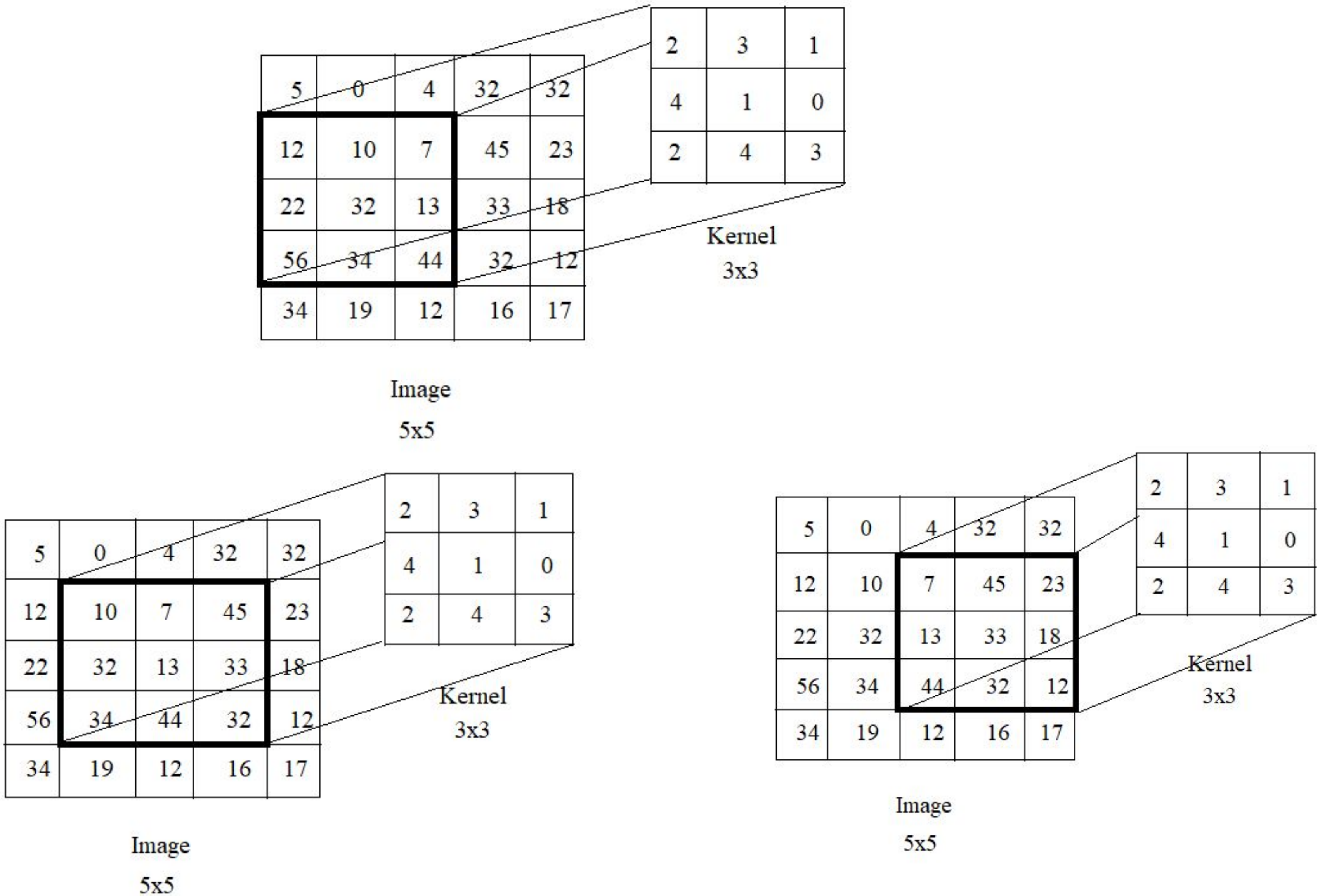
Filtered image
3x3

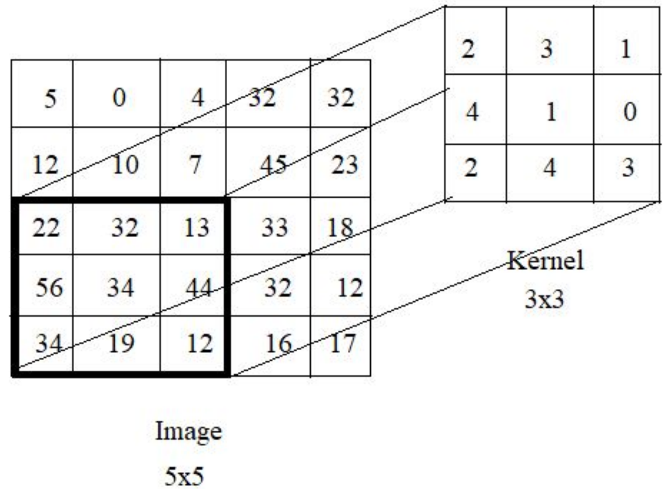


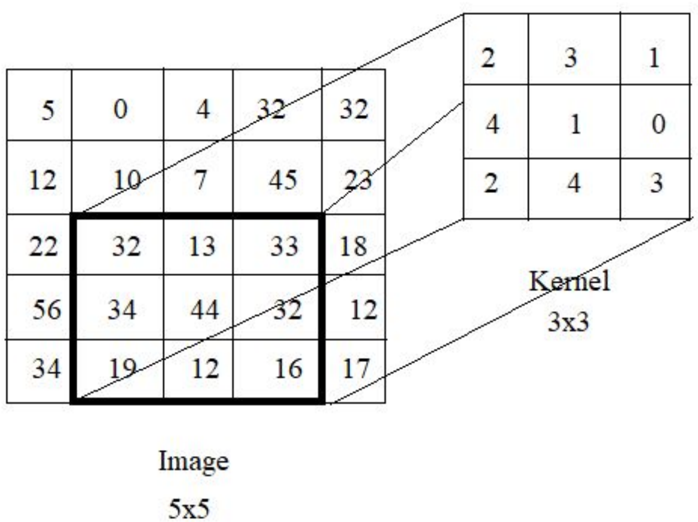
$(5x2)+(0x3)+(4x1)+(12x4)+(10x1)+(7x0)+(22x2)+(32x4)+(13x3) = 291$

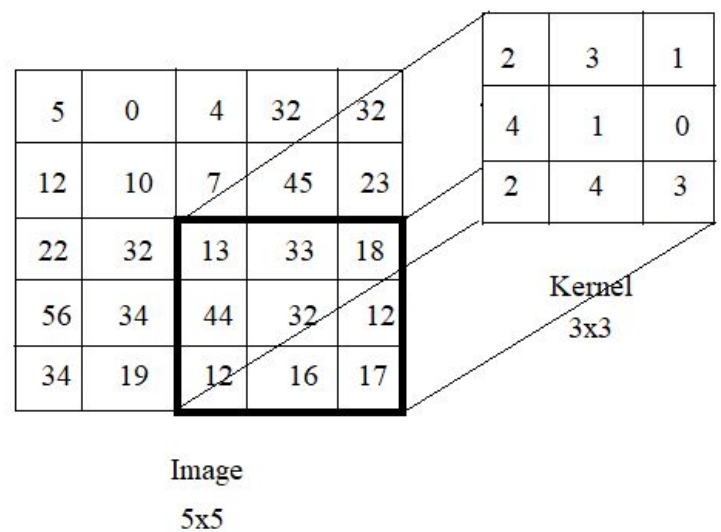

5 0 4 32 32
12 10 7 45 23
22 32 13 33 18
56 34 44 32 12
34 19 12 16 17
2 3 1
4 1 0
2 4 3
Kernel
3x3
Image
5x5


5 0 4 32 32
12 10 7 45 23
22 32 13 33 18
56 34 44 32 12
34 19 12 16 17
2 3 1
4 1 0
2 4 3
Kernel
3x3
Image
5x5


Kernel
3x3
Image
5x5
Kernel
3x3
Image
5x5
Kernel
3x3
Image
5x5


5 0 4 32 32
12 10 7 45 23
22 32 13 33 18
56 34 44 32 12
34 19 12 16 17
2 3 1
4 1 0
2 4 3
Kernel
3x3
Image
5x5


5 0 4 32 32
12 10 7 45 23
22 32 13 33 18
56 34 44 32 12
34 19 12 16 17
2 3 1
4 1 0
2 4 3
Kernel
3x3
Image
5x5


5 0 4 32 32
12 10 7 45 23
22 32 13 33 18
56 34 44 32 12
34 19 12 16 17
2 3 1
4 1 0
2 4 3
Kernel
3x3
Image
5x5

Resultant image (collections of receptive fields)

291	306	285
320	215	125
250	134	213

- Each of this receptive fields corresponds to a neuron.
- Thus the neuron is not connected to the entire input, but just to some section of the input.
- Multiple filters are added with varying values and the same process is done to look for a number of features thus generating the feature map.
- These filter weights get updated in multiple iterations just like the neuron weights in ANN.
- This is called convolution and hence the name **convolutional layer**.

If n is the size of the image and f is the size of the filter, then the size of the receptive field is

$\boldsymbol{n - f + 1}$ (without padding and striding)

Striding and padding

- **Stride** denotes the no of movement the filter makes in each step of convolution. By default, stride = 1.
- Set using strides = (int, int)

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

When stride =2

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

5	0	4	32	32
12	10	7	45	23
22	32	13	33	18
56	34	44	32	12
34	19	12	16	17

- Striding reduces the size of the output image further.

- **Padding** is done to retain the size of the original image to avoid information loss.
- Set using padding parameter; padding = valid or same
- padding = valid : no padding done. Allows reduction of features
- padding = same: retains the size of the image by padding zeros
- Padding alone will change the output size to **n-f+2p+1**
- Using striding and padding, the size of the convolved output will be **(n-f+2p)/s)+1**

  n is the input size

  f is the filter size

  p is the amount of padding

  s is the amount of striding

- The amount of padding done in same padding is

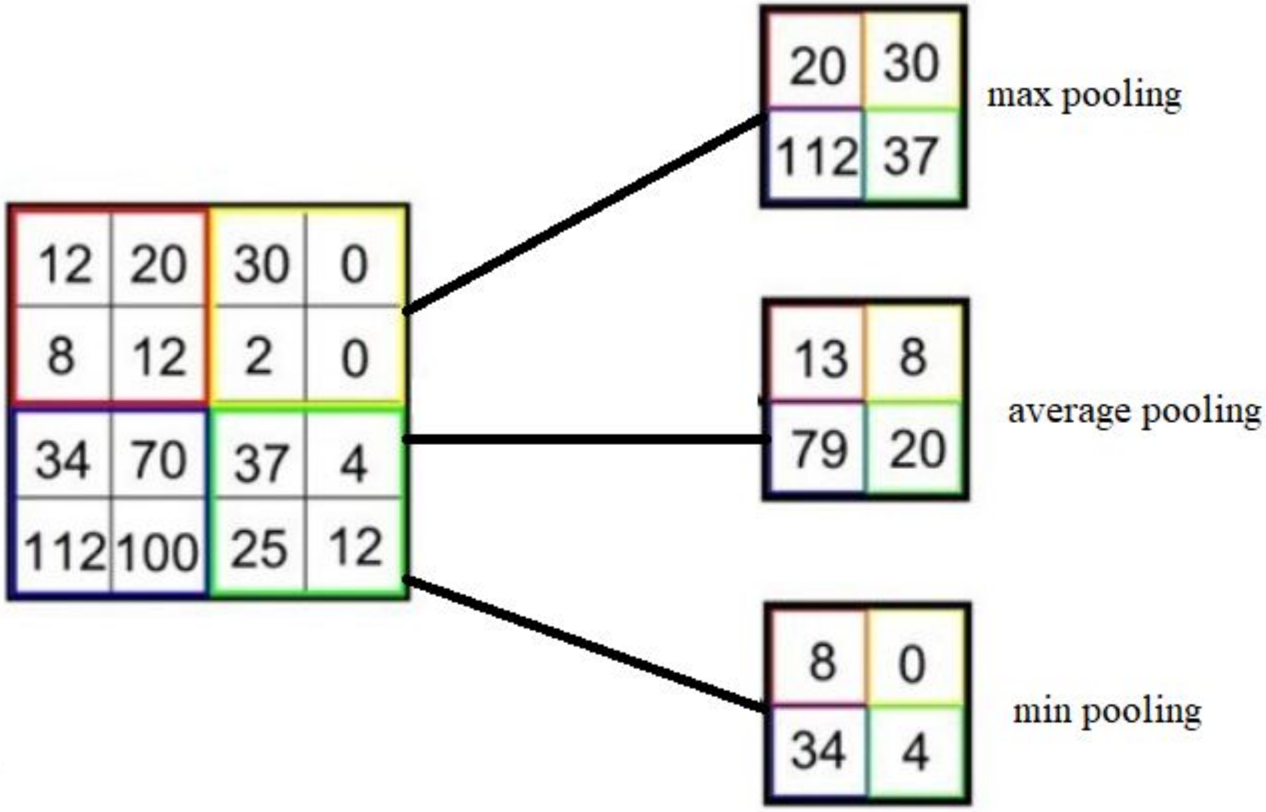
$$\mathbf{p = (f - 1) / 2}$$

0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

Pooling and types

- Pooling is done to reduce the dimensionality of the image.
- Reduces the amount of computation needed.
- Pooling layer reduces the no of pixels in the image obtained from the previous layer.
- Added after the convolutional layer.
- It summarizes the features in a region of the feature map generated by a convolutional layer.

- **Max pooling**: calculates maximum of each block of the feature map.
- **Average pooling**: calculates average of each block of the feature map.
- **Min pooling**: calculates minimum of each block of the feature map.



12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
20 30
112 37
max pooling
13 8
79 20
average pooling
8 0
34 4
min pooling

Fully connected layer (FCL)

- Convolutional layers provides a low dimensional, yet powerful, feature map.
- Last layer is a connected fully for learning non-linear combinations of these features.

Regularization

- To prevent over fitting

Drop out

- Usually done after pooling layers only (not necessarily)
- Can be added after conv2D layers too
- Applied to each element or cell within the feature map

  from tensorflow.keras.layers import Dropout

  model.add(Dropout(drop_out_ratio))
- Dropout entire feature map from the convolutional layers

  from tensorflow.keras.layers import SpatialDropout2D

  model.add(SpatialDropout2D(drop_out_ratio))

- Dropout values
  - Dropout value 1 : no dropout
  - Dropout value 0 : complete dropout (no output from the previous layer)
- Best dropout values between 0.5 and 0.8

Data Augmentation

- To increase the amount of data
- Adding slightly modified version of existing data
- Scaling, rotation (at 90 degrees), rotation (at finer angles), flipping, adding salt and pepper noise, changing lighting conditions etc.

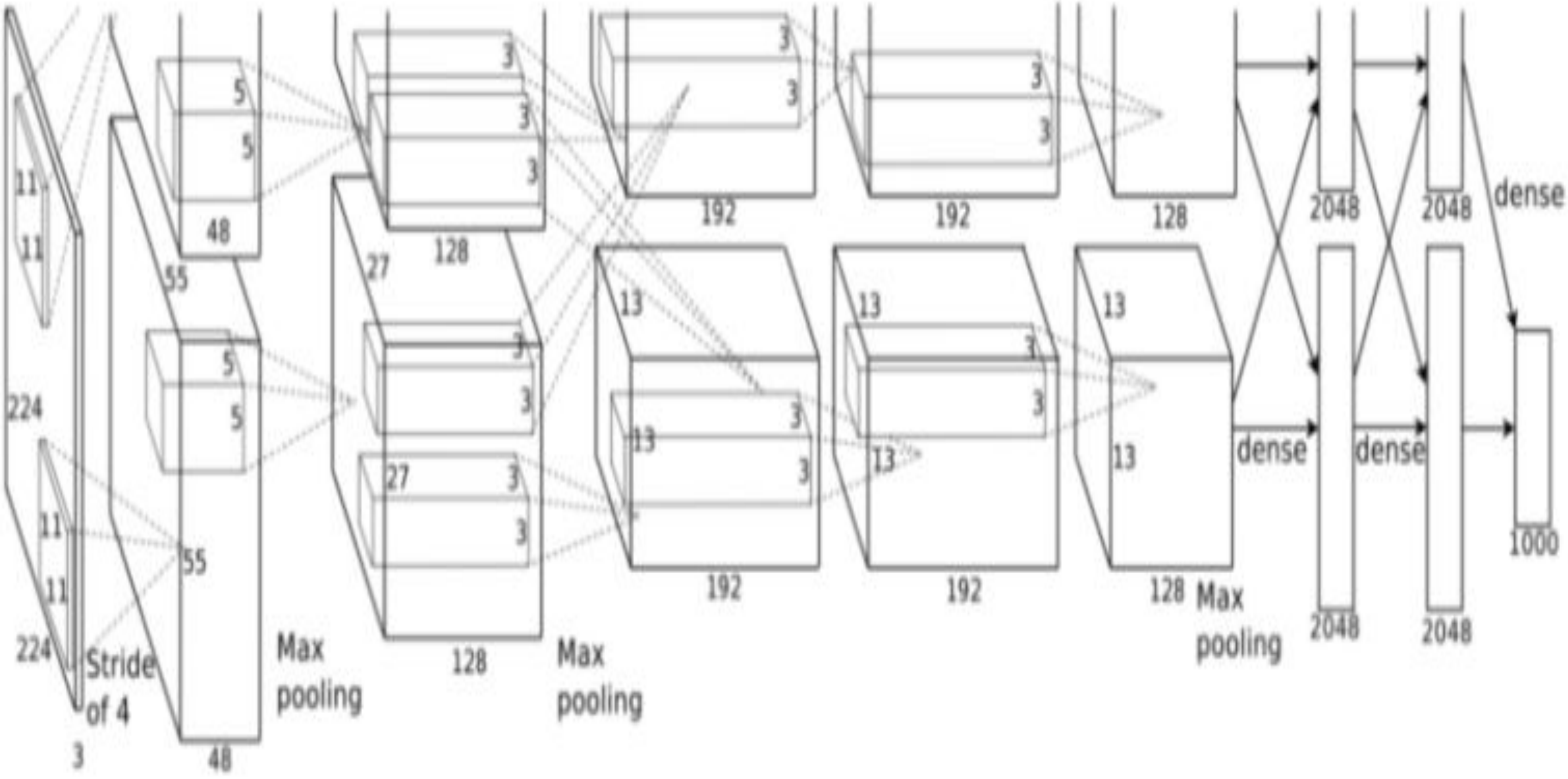
Transfer Learning
Different CNN architectures

- Transfer learning: using pre-trained models.
- Image classification eg; AlexNet, VGG network, ResNet, MobileNet etc

- Object detection eg: Fast R-CNN, Mask R-CNN, YOLO, SSD etc

AlexNet

- Designed by Alex Krizhevsky for the ImageNet Large Scale Visual Recognition Challenge which had an accuracy of 84.7%.
- 15 million high-resolution images labeled with 22 thousand classes from ImageNet
- The hidden layers consist of convolutional layers, pooling layers, fully connected layers, and normalization layers
- High performance of the model was due to its depth which is expensive computationally.
- GPUs solved the problem.



11
11
224
11
11
224
Stride of 4
3
55
5
5
48
55
5
5
48
Max pooling
3
3
3
27
128
3
3
27
3
3
128
Max pooling
3
3
13
13
3
3
192
192
3
3
13
13
3
3
192
192
13
13
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000

- A total of 8 layers; 5 convolutional layers and 3 fully connected layers

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-
Fully Connected 1	-	-	-	-	4096	ReLU
Dropout 2	rate = 0.5	-	-	-	4096	-
Fully Connected 2	-	-	-	-	4096	ReLU
Fully Connected 3	-	-	-	-	1000	Softmax

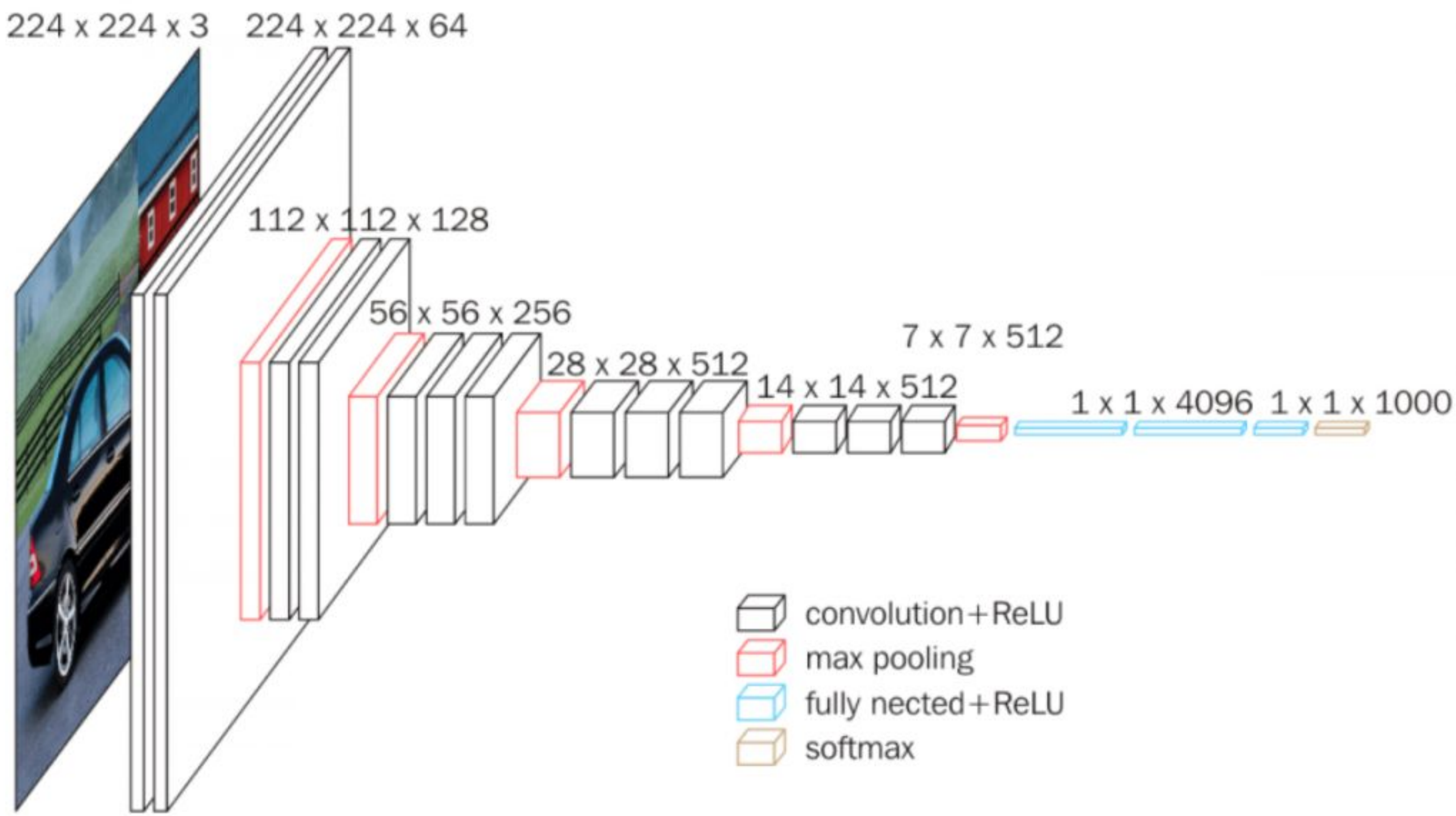
Features

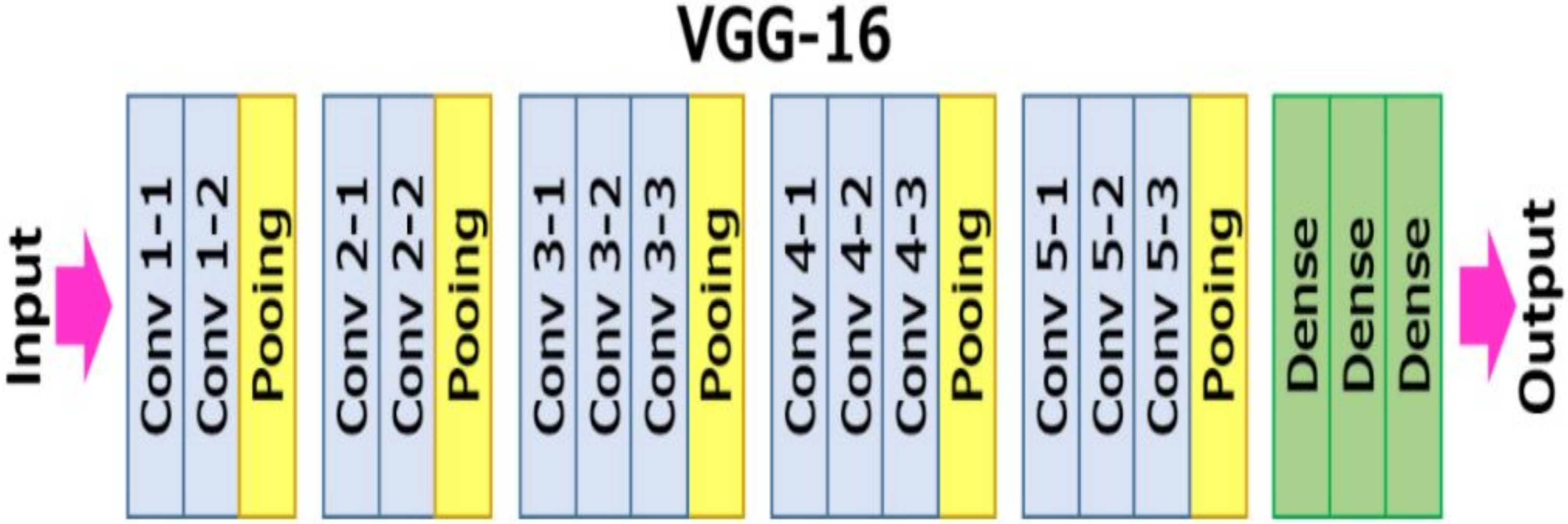
- ReLU activation function
  - several times faster than tanh and sigmoid- based networks
- Standardization
  - the value after the activation function has no range like the tanh and sigmoid functions, so a normalization will usually be done after ReLU
- Multiple GPUs
- Overfitting
  - Drop out
  - Data augmentation

VGG16 (OxfordNet)

- For classification and detection
- Proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"
- **VGG-Visual Geometry Group** (a group of researchers at Oxford who developed this architecture)
- 16 layers
- Accuracy of 92.7%
- Replaces large kernel-sized filters of AlexNet with multiple 3×3 kernel-sized filters one after another

Architecture




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

VGG16