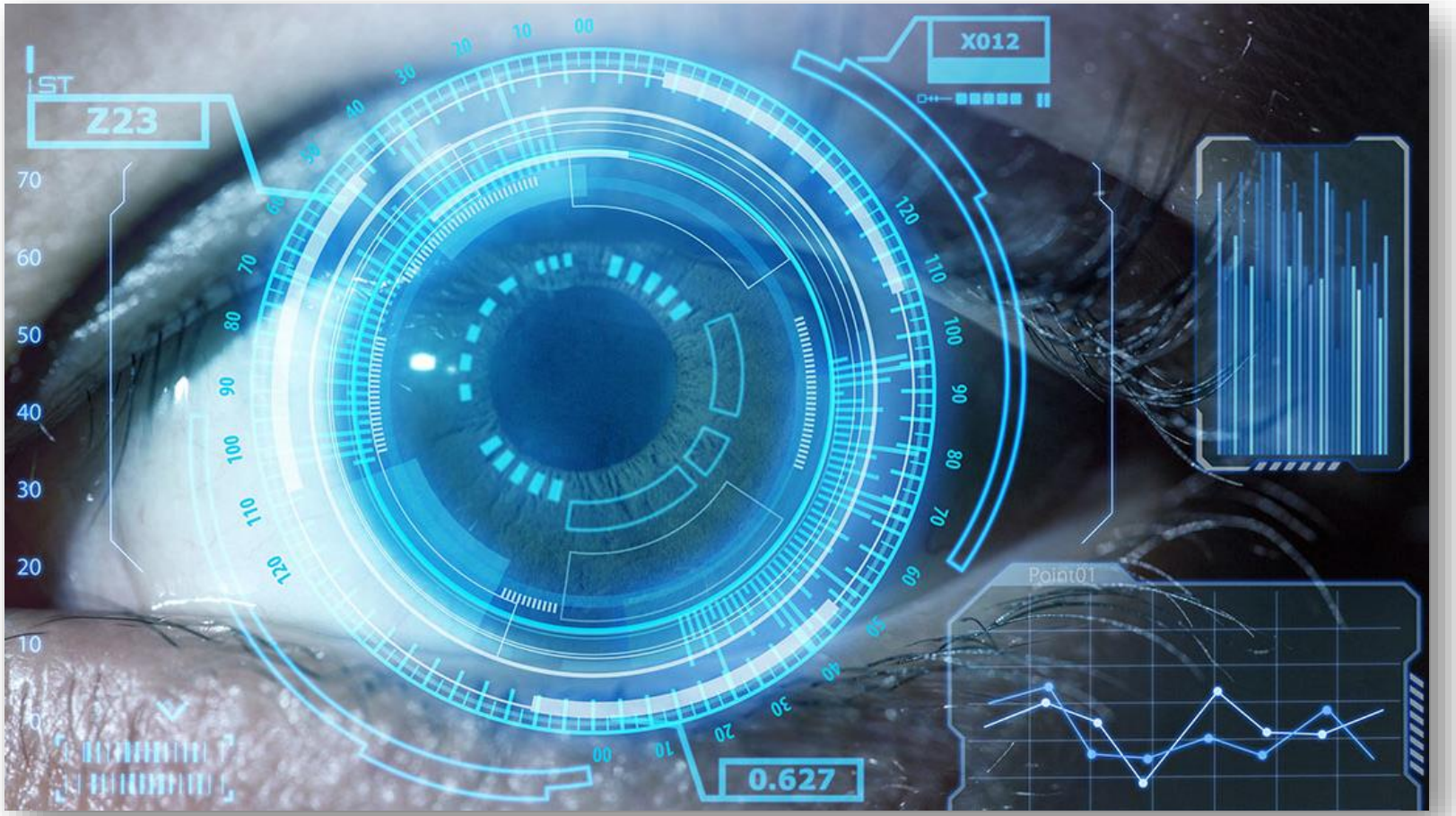
Deep Learning

Md. Jalil Piran, PhD
Asst. Professor
Computer Science and Engineering
Sejong University
Spring, 2021

Outline

- Introduction to Deep Learning (DL)
- The History of DL
- Programming Tools
- Artificial Neural Networks
- Convolutional Neural Networks

CONVOLUTIONAL NEURAL NETWORKS



Computer Vision

2011

- Google's Artificial Brain Learns to Find Cat Videos

Computer Vision

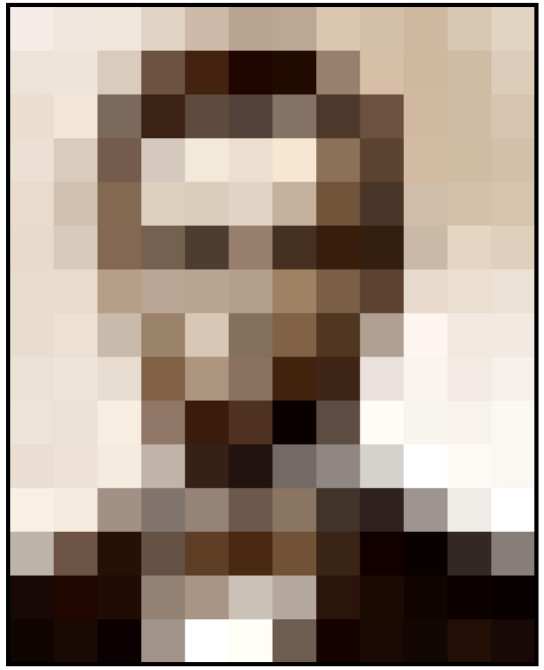
2014

- Clune and his then-PhD student Anh Nguyen at Cornell University

King Penguin **Starfish** **Baseball** **Electric Guitar**

Computer Vision

- **Images** and **numbers**!



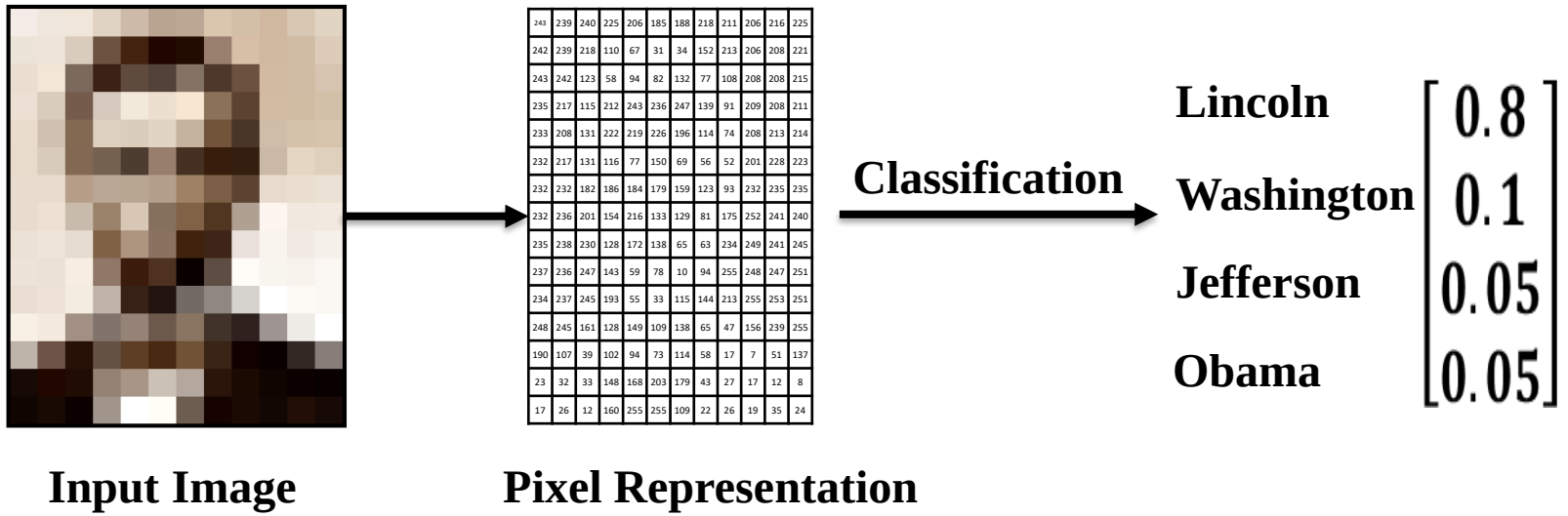
243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	[illegible]	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	[illegible]	[illegible]	51	137
[illegible]	[illegible]	[illegible]	148	168	203	179	43	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	160	255	255	109	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

An image is just a matrix of number [0,255]

Computer Vision

- Tasks in computer vision
  - **Regression**: output variable takes continuous value.
  - **Classification**: output variable takes class label. Can produce probability of belonging to a particular class.

- Which US president is in the image?



High level feature extraction

- Let's identify key features in each image category.

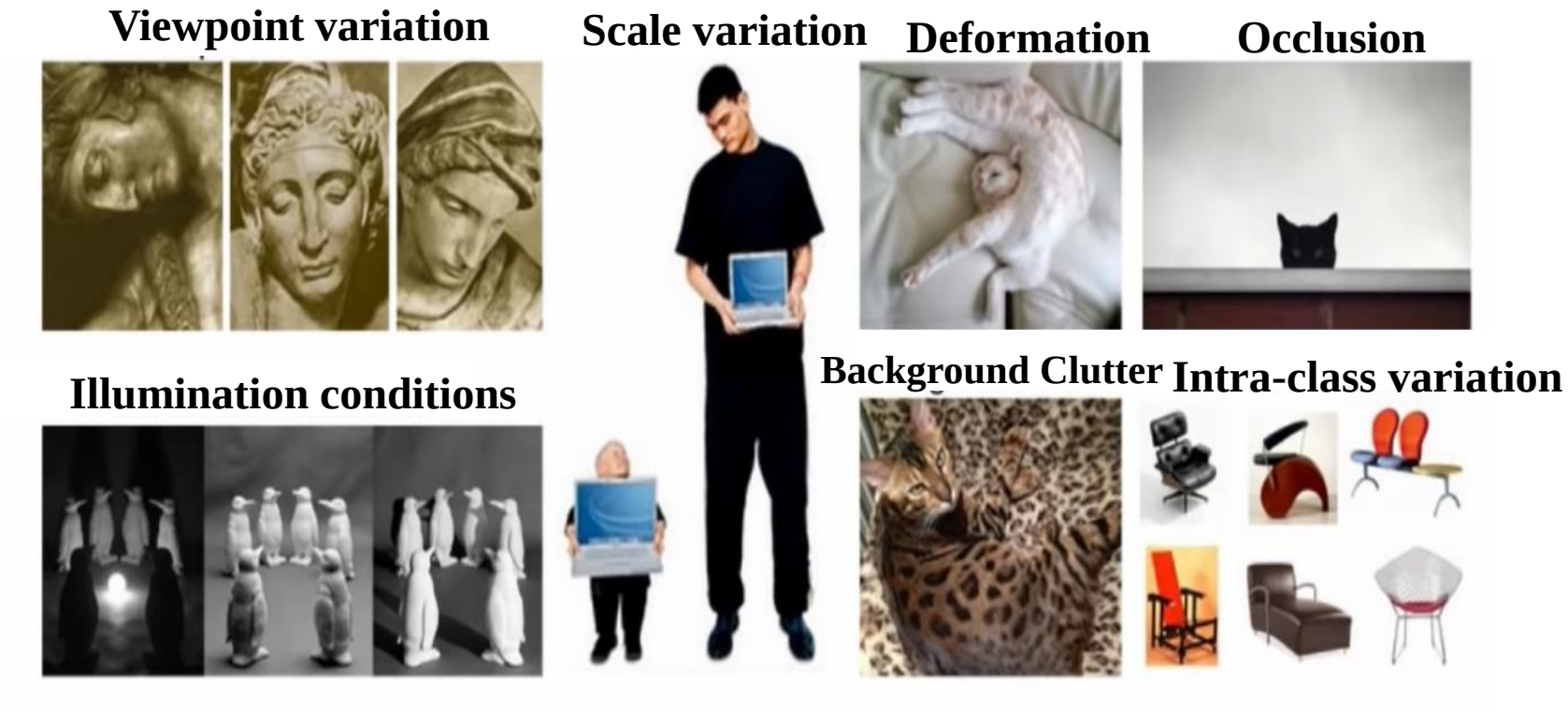
Nose,
Eyes,
Mouth

Wheels,
License Plate,

Headings,
Door Windows,
Steps

Computer Vision

- However, it is not easy always!

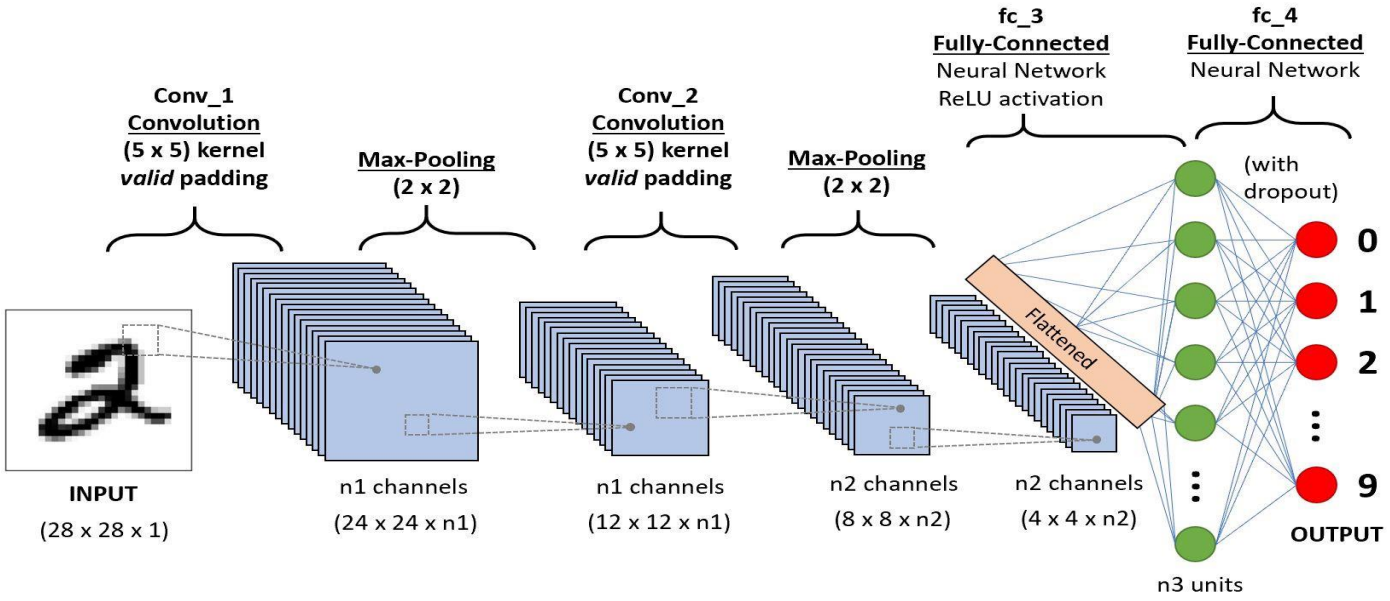


Recall ANN

1. Select a set of **data** randomly.
2. **Feedforwarding**
3. Compute the **cost function**
4. **Backpropatation**
5. Change the **weights** and **biases**

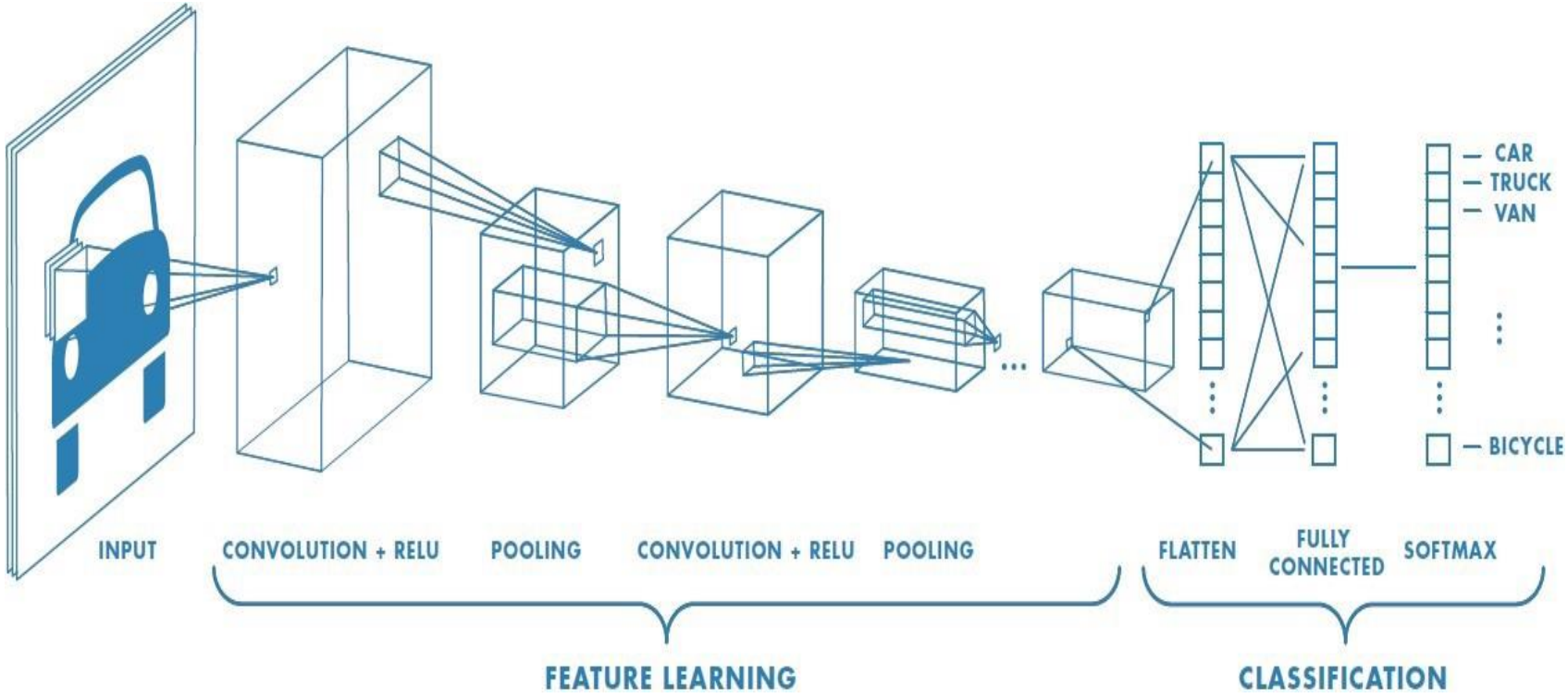
Convolutional Neural Network

- a.k.a **ConvNet or CNN**
- CNN was inspired by the organization of the **Visual Cortex**
- CNN is a **Deep Learning algorithm**
- CNN:
  - Takes in an input image,
  - Assign importance (learnable weights and biases) to various aspects/objects in the image
  - Differentiate one from the other.



Architecture

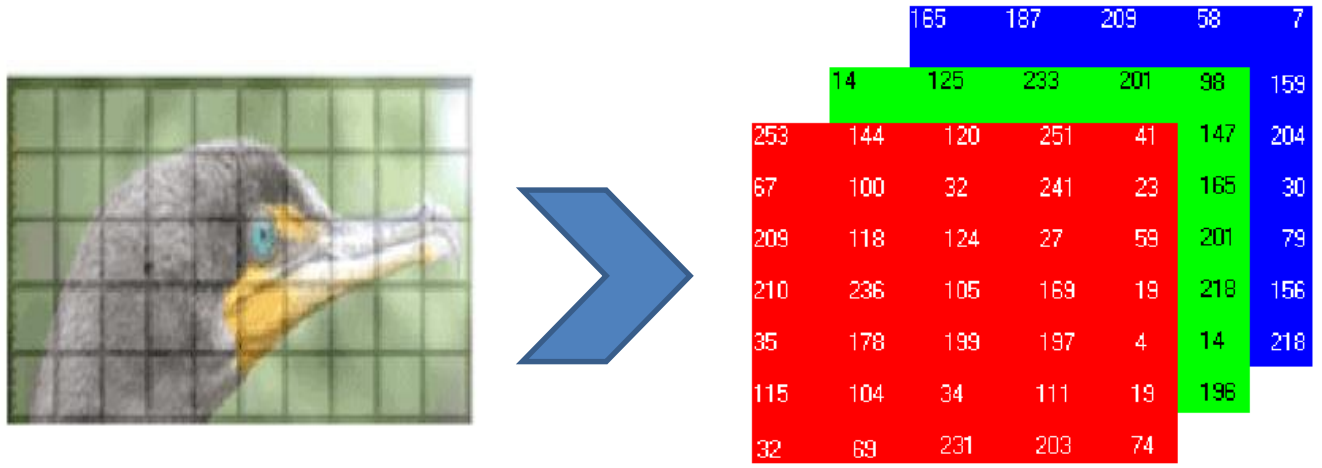
- Input layer
- Filter or Kernel, e.g. learned by DL.
- Convolutional Layers
- ReLU
- Pooling
- Padding
- Fully connected layers
- Output layer



Architecture

Input layer

- **Input:** a "$m \times m \times r$" image
- Color images: 3D mtrix:
  e.g. $1080 \times 1080 \times 3$ RGB



- Computationally intensive: 8K (7680×4320)

Convolution Layer

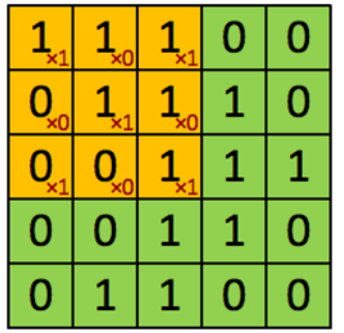
- Convolution layer:
  - The first layer to **extract features** from an input image.
- Convolution preserves the **relationship between pixels**
  - By learning image **features** using small squares of input data.
- It is a **mathematical** operation
  - Takes two inputs such as **image** matrix and a **filter** or **kernel**.
- Objective: to **extract the high-level features** such as edges, from the input image.

Architecture

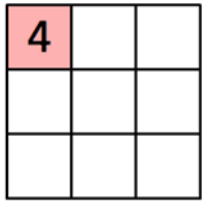
Convolution Layer

- **Green**: resembles the $5 \times 5 \times 1$ input image

- **Yellow**: Kernel/Filter, a $3 \times 3 \times 1$ matrix.
  - the kernel has the same **depth** as the input image



Image



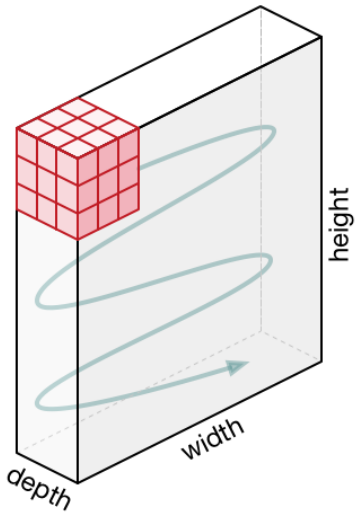
Convolved Feature

- **Stride Length** = 1, one step movement.

- Every time: a matrix multiplication between the kernel and the portion of the image over which the kernel is hovering.

1 0 1
0 1 0
1 0 1

Kernel / Filter

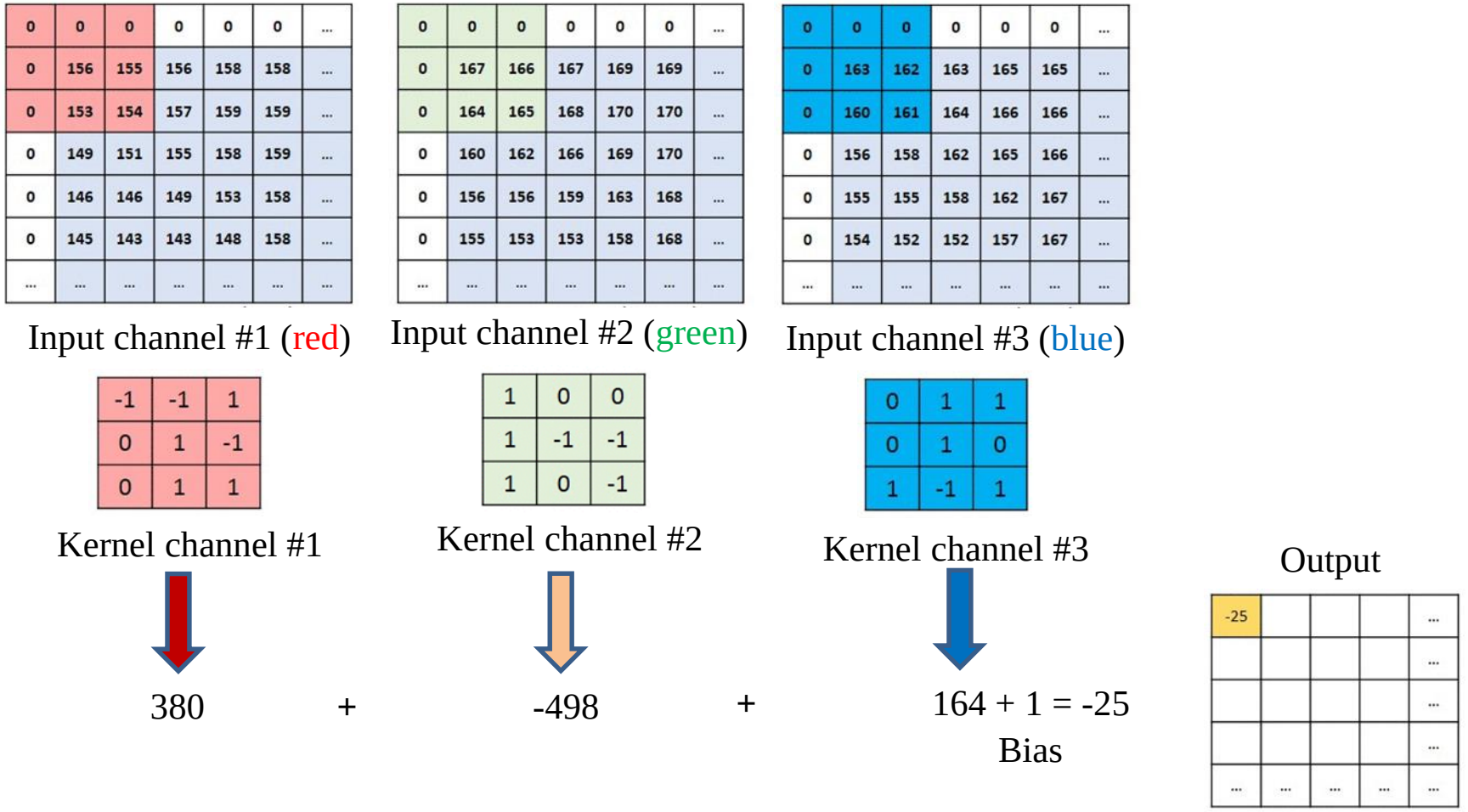


- **Convolution layers**; convolution two signals e.g,

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2].g[x-n_1,y-n_2]$$

Architecture

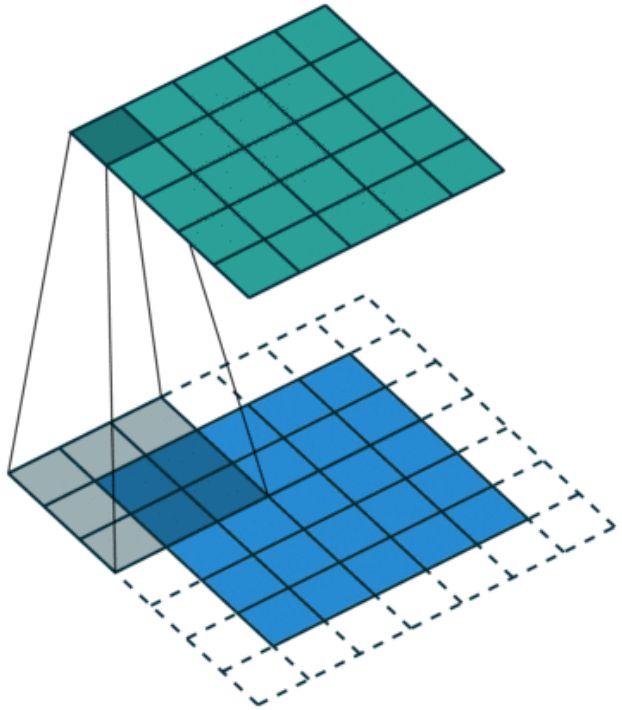
- The filter moves to the right with a certain **Stride Value** till it parses the complete width.



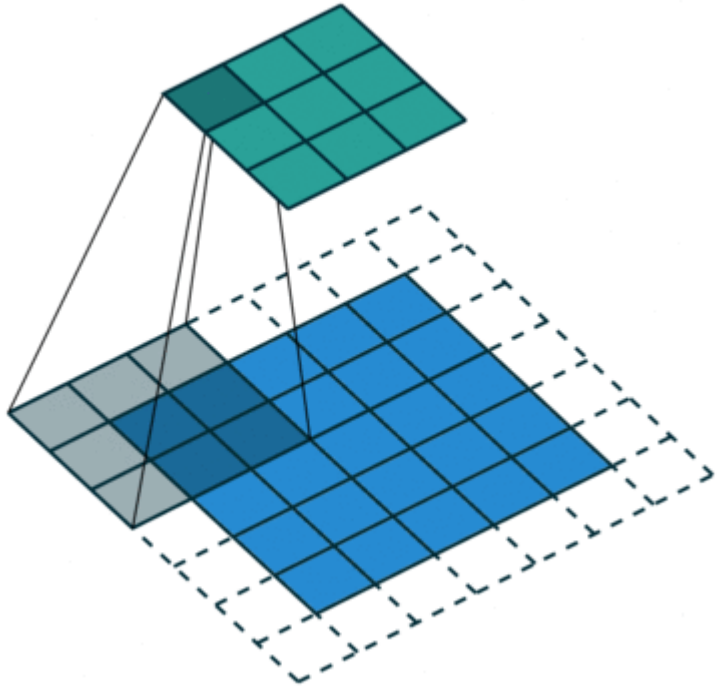
Architecture

Stride

- **Stride:** the amount by which the kernel is moved by as the kernel is passed over the image.



Convolution Operation with
Stride Length = 1

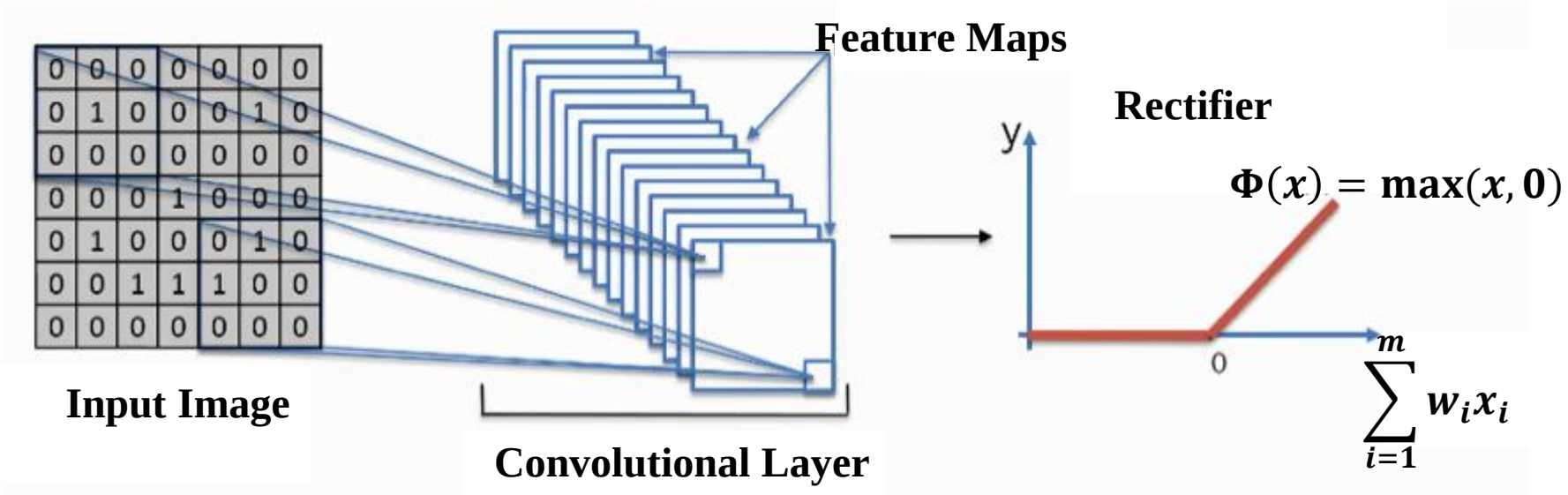


Convolution Operation with
Stride Length = **2**

Architecture

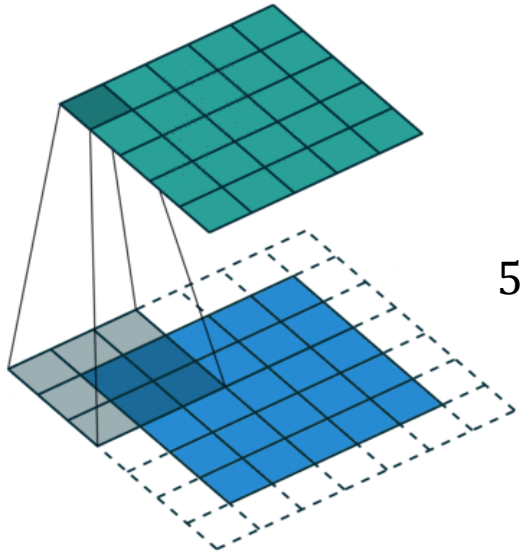
The Rectified Linear Unit (ReLU)

- A supplementary step to the convolution operation.
- Just changes all the negative activations to 0.
- Rectifier function: to increase the non-linearity in our images.



Padding

- Results:
  - the convolved feature is **reduced** in dimensionality as compared to the input,
  - the dimensionality is either **increased** or remains the **same**.
- **Valid Padding;** in case of the former,
- **Same Padding;** in the case of the latter.

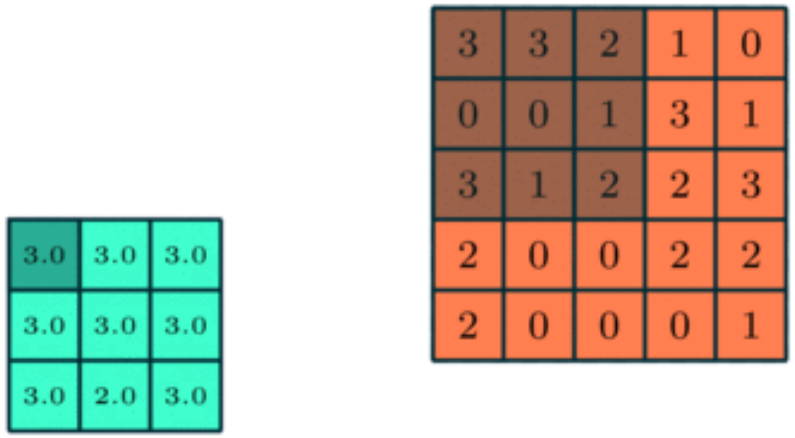


$5 \times 5 \times 1$ image is padded with 0s to create a $6 \times 6 \times 1$ image

Architecture

Pooling Layer

- Responsible for reducing the spatial size of the Convolved Feature.
- Goal: to **decrease the computational power required to process the data** through dimensionality reduction.

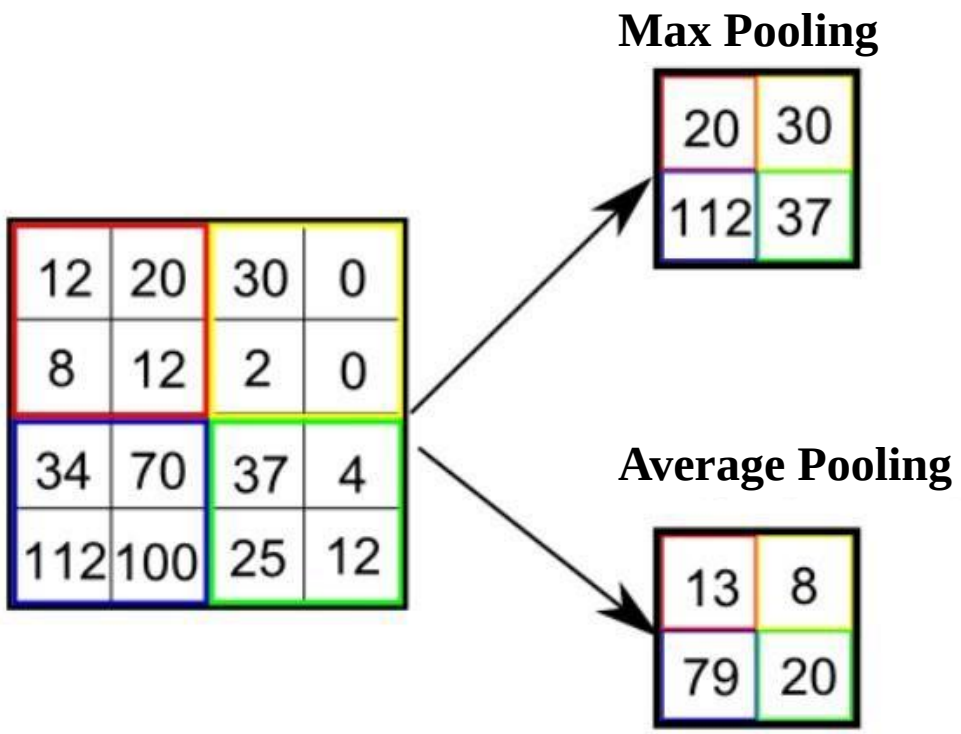


3 × 3 pooling over 5 × 5 convolved feature

Architecture

Pooling Layer

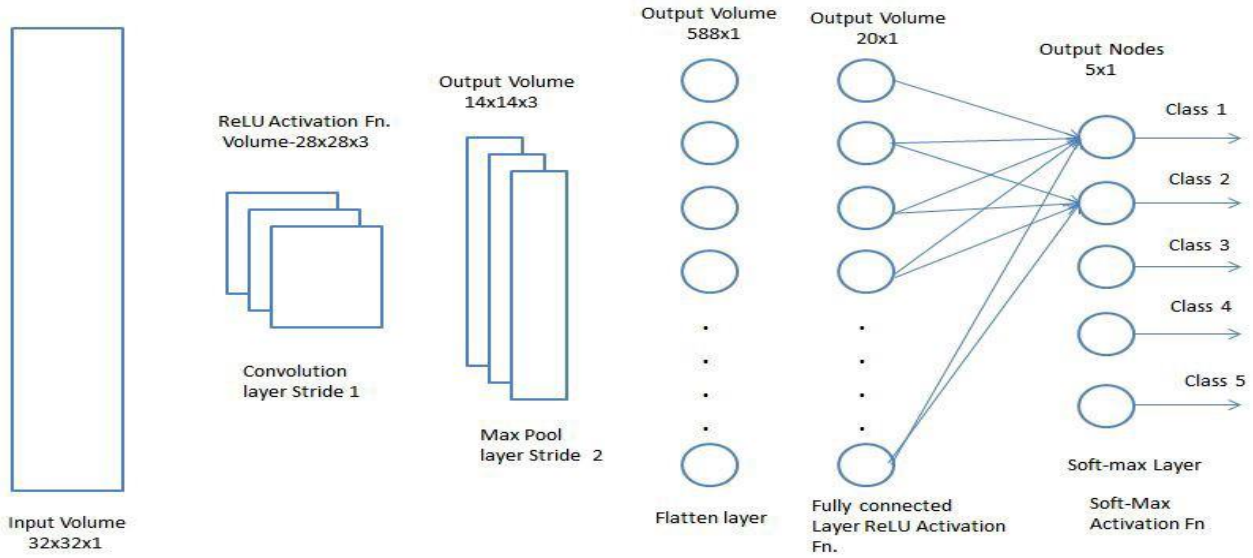
- Types:
  - **Max Pooling**
  - **Average Pooling.**



Architecture

Classification — Fully Connected Layer (FC Layer)

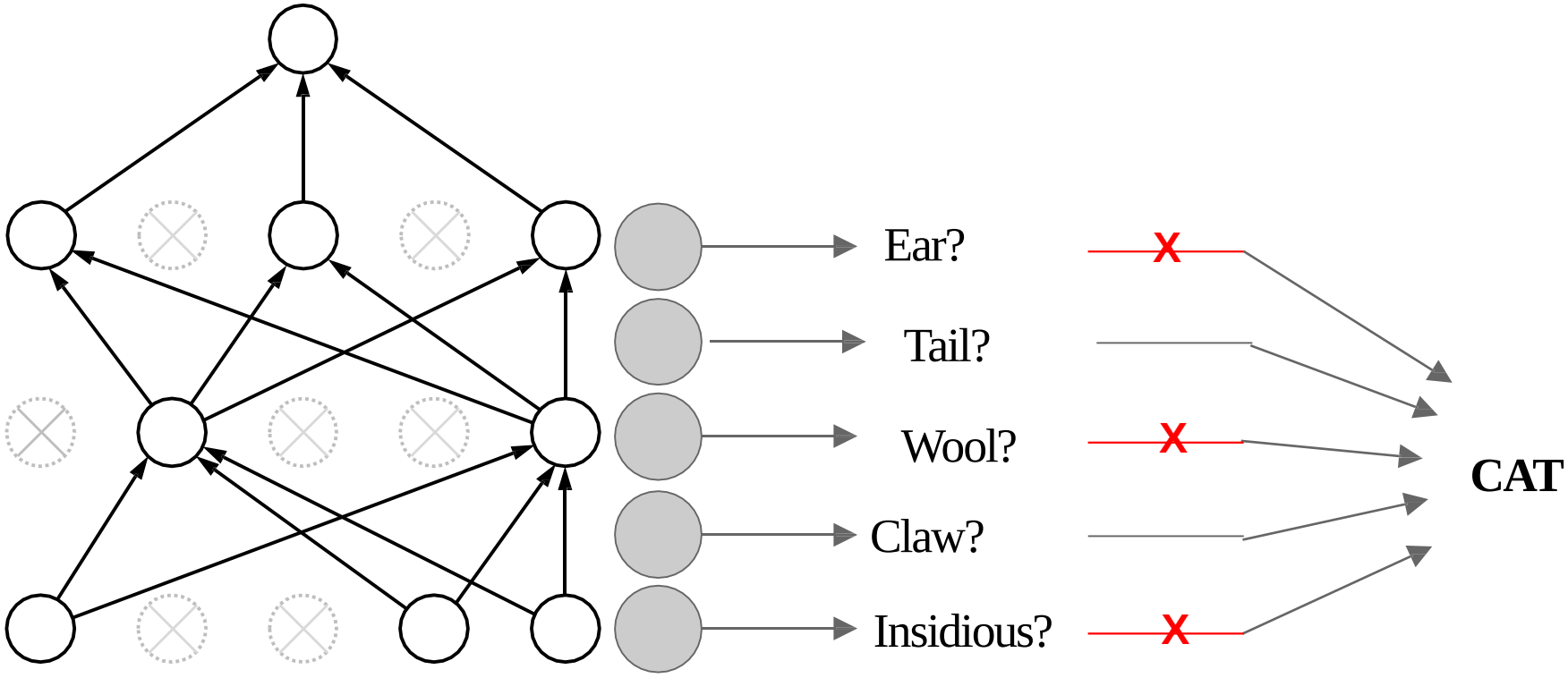
- FC layer: for learning non-linear combinations of the high-level features as represented by the output of the convolutional layer.
- Over a series of **epochs**, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.



Architecture

Dropout Layers

- Drops out a random set of activations in that layer by setting them to zero.



Architecture

- CNN Architecture:
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet

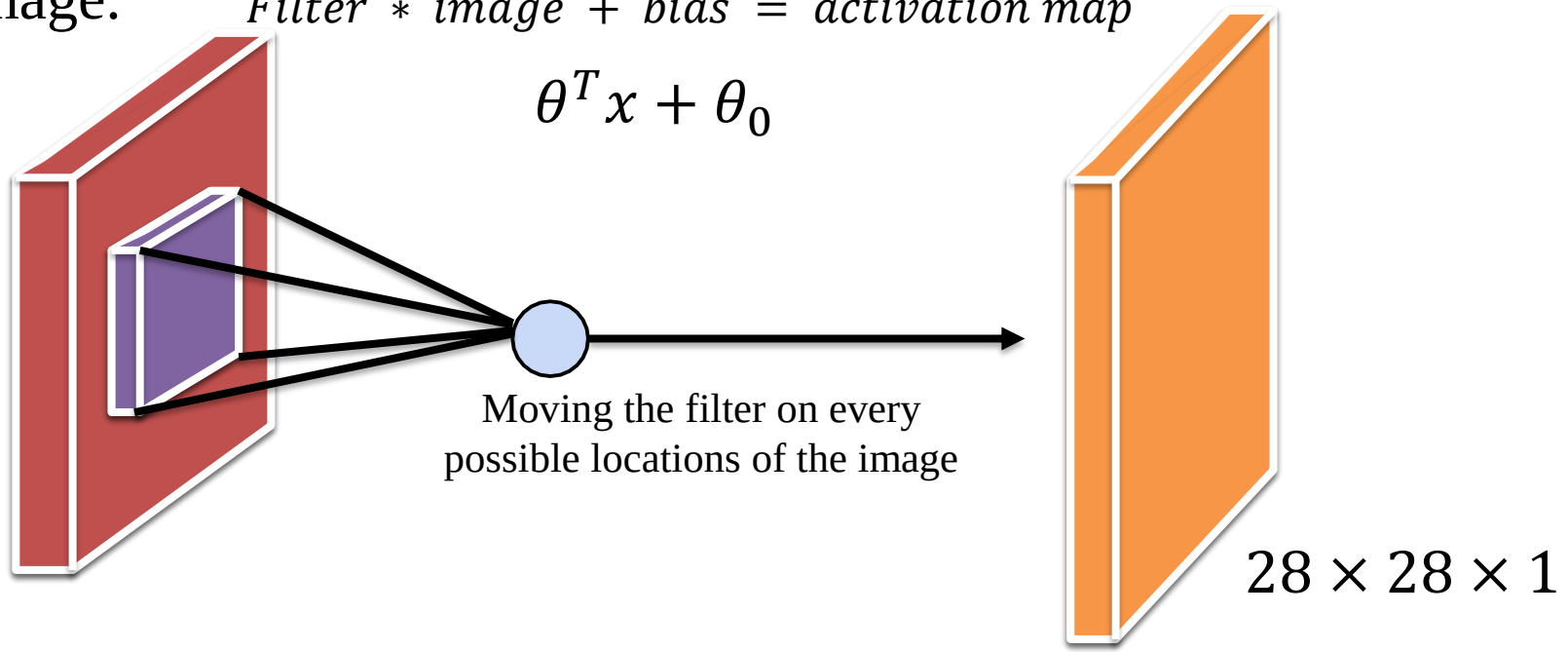
Convolution filter by image

- Put the filter
- The depth of the filter **equal to** the images.
- Calculate the **inner product** of the filter and a part of the image that is covered by the filter and store the result.
  - e.g. the similarity between the filter and image pixels, the higher similarity, the higher inner product.
- **Move** the filter by one pixel and calculate the inner production again.
- The results of all inner products form an matrix that is known as **Activation Map**.

- The result of the inner production:
  - Multiplication of two vectors with 75 dimensions + the **bias**
- Every multiplication result is stored in a **neuron**.
- So, the next layer is the **neurons** which are the results of the inner production of the filter and the image.

$$Filter * image + bias = activation\ map$$

$$\theta^T x + \theta_0$$



Edge Detection

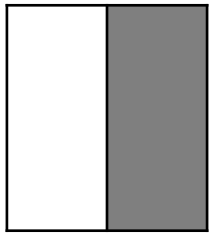
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

$*$

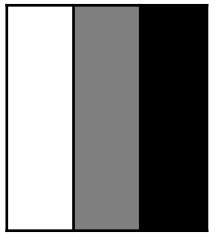
1	0	-1
1	0	-1
1	0	-1

$=$

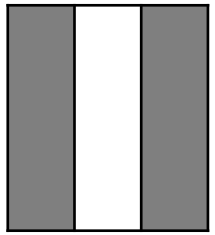
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



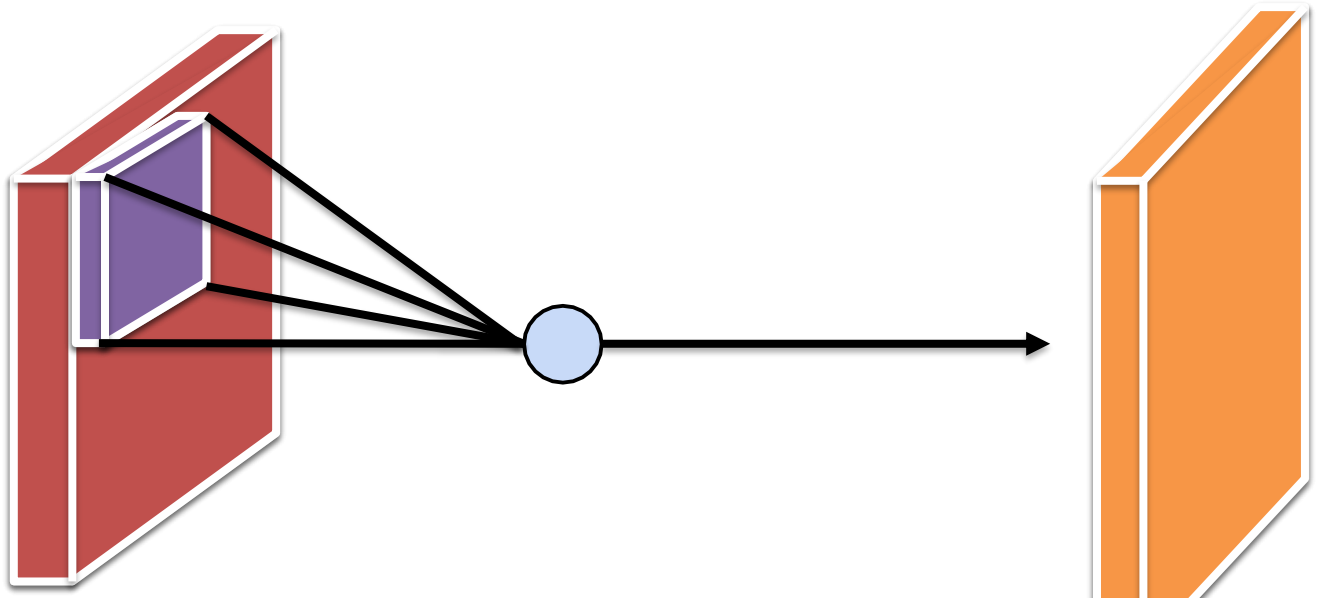
$\times$



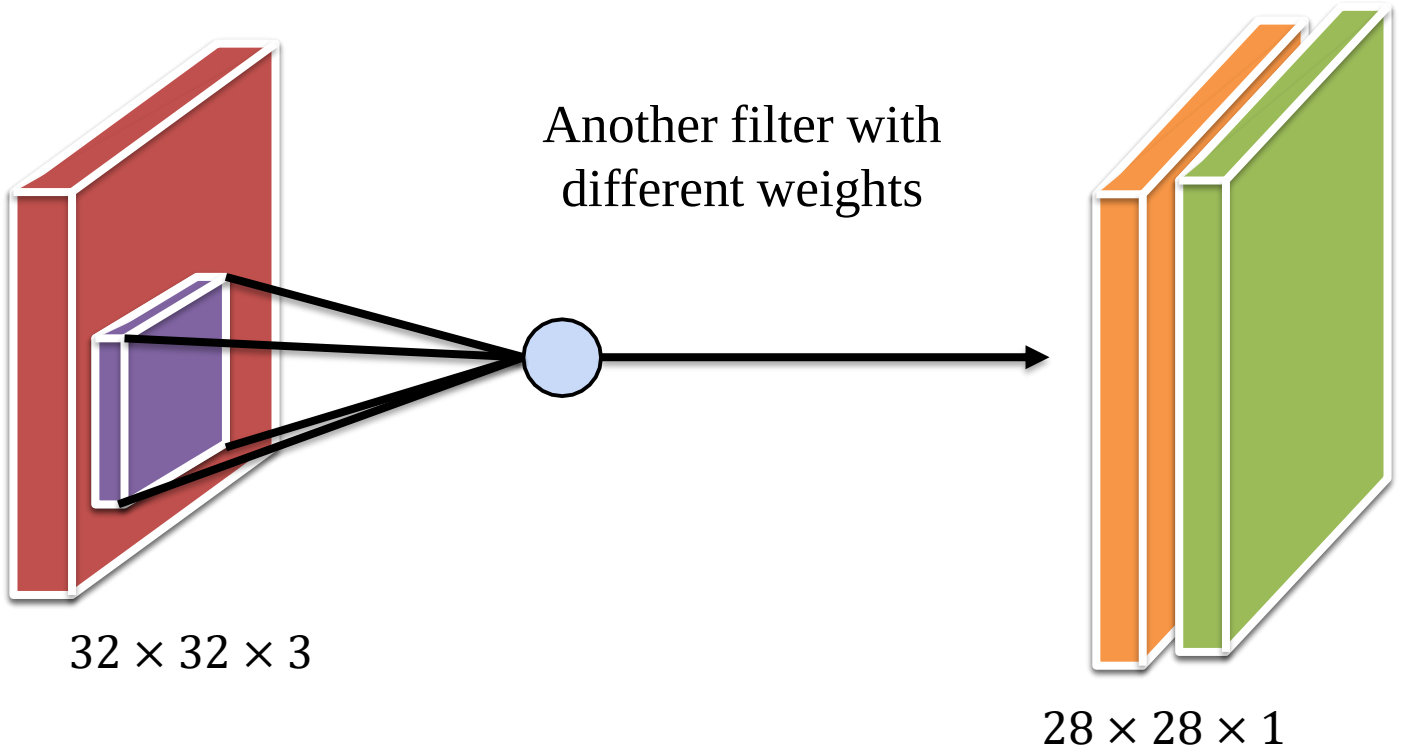
$=$



- Result: $28 \times 28 \times 1$ **dimension matrix**,
  - because our **filter** is $5 \times 5 \times 3$ and the **image** is $32 \times 32 \times 3$, so, 2 pixel from each part will be **ignored**

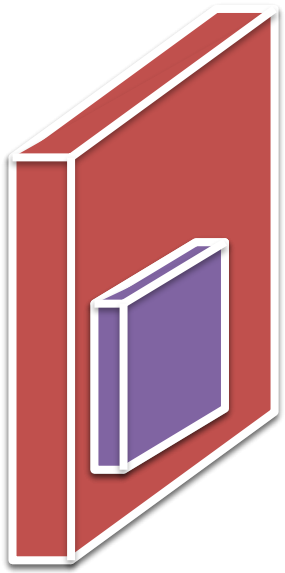


- Every time we may use **different filters** to **detect** different parts of the image.
- The result of convolutions of every filter by the images will be a **different activation map**.



Image

Activation maps



6 filter (5×5) -> 6 activation maps

Convolution layers

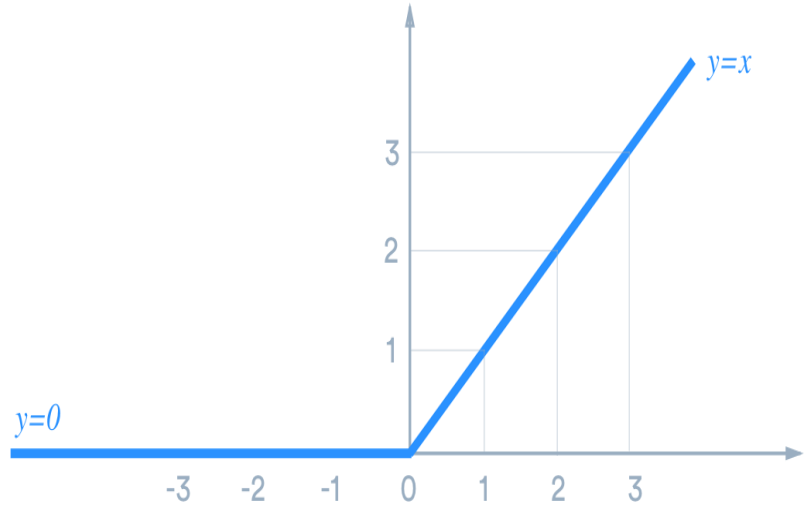
$32 \times 32 \times 3$

$28 \times 28 \times 1$

By putting all the activation maps together, we will have a new image in $28 \times 28 \times 6$ size

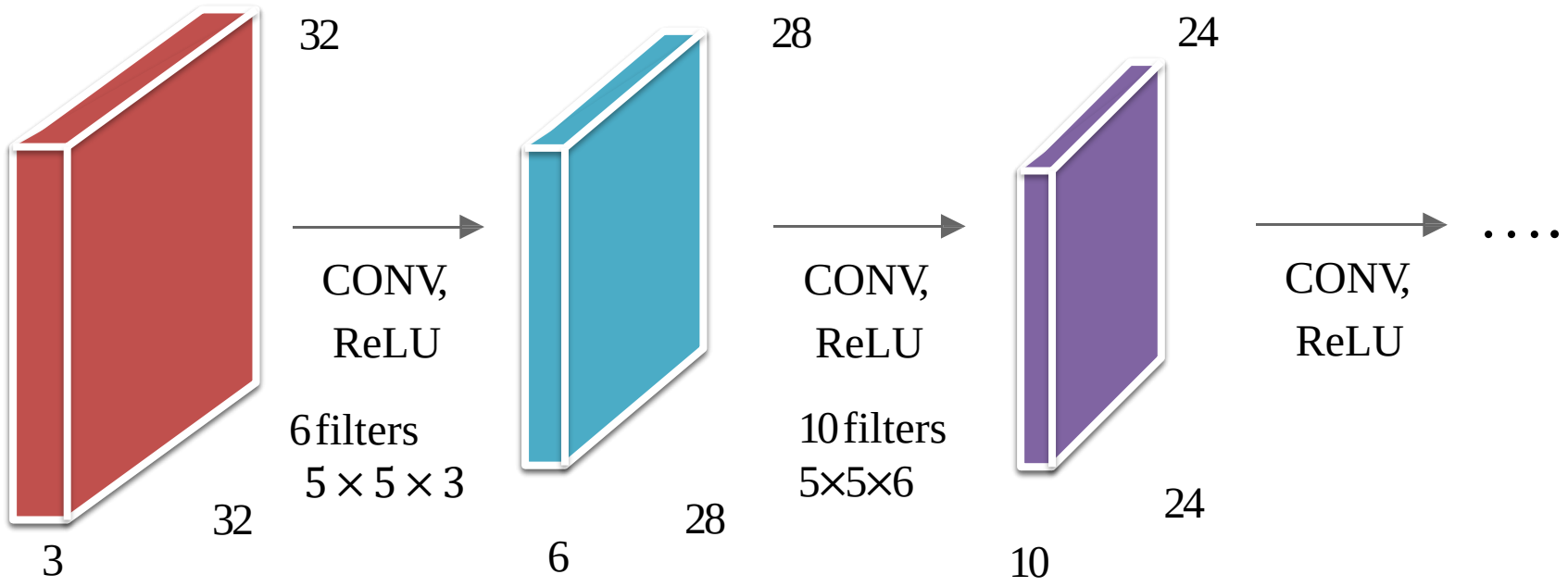
- The activation function: **Rectified Linear Unit (ReLU)**.

$$y = \max(0, x)$$

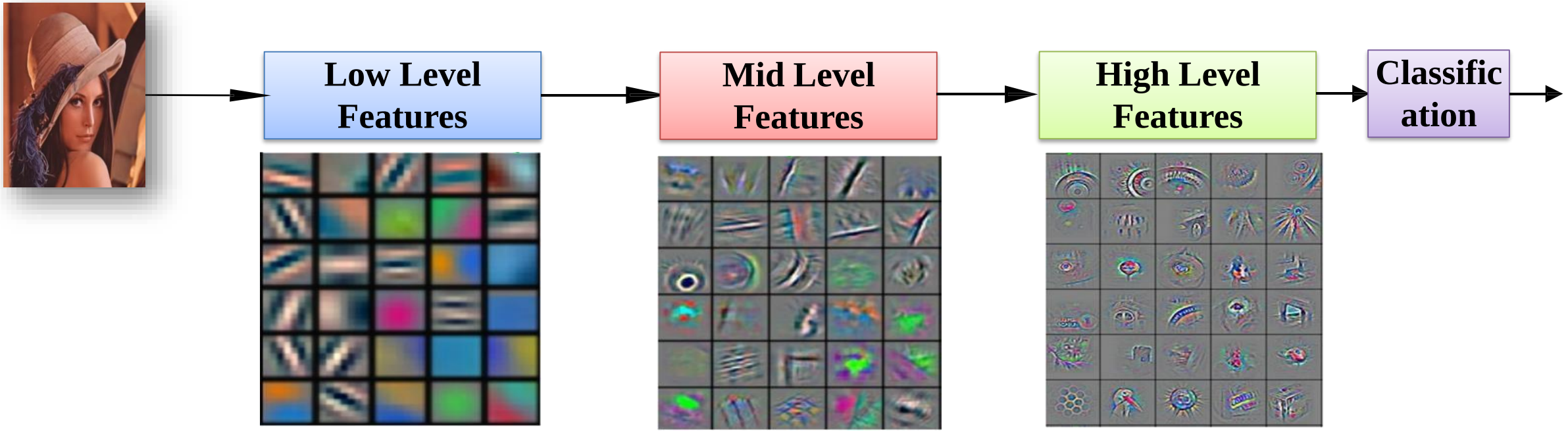


- **ReLU**
  - Common in CNNs.
  - Linear (identity) for all positive values, and zero for all negative values.
  - Cheap to compute as there is no complicated math.
  - Less time to train or run a model.
  - Converges faster.

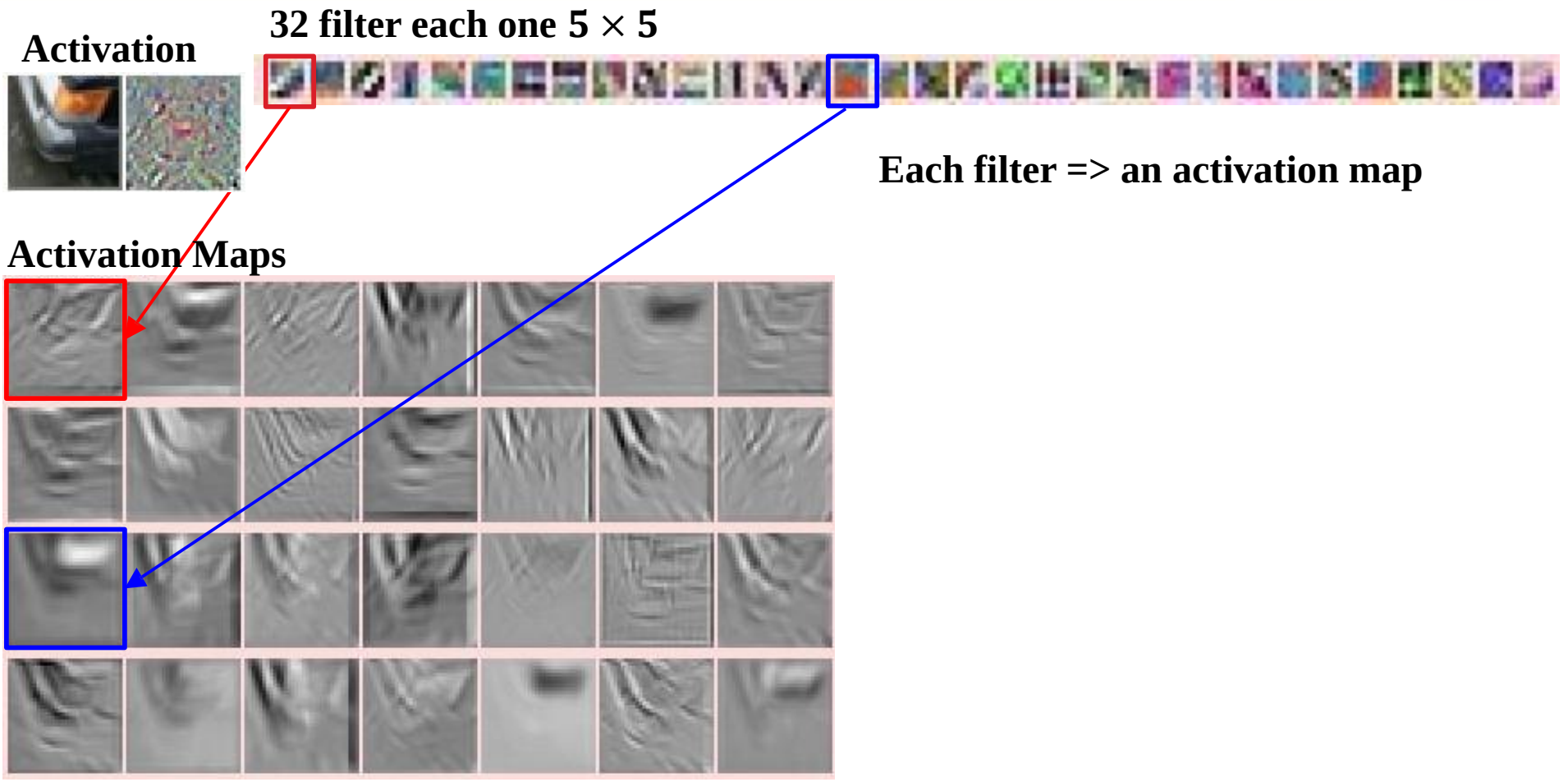
- CNN:
  - A **series of convolution layers**
  - Activation functions between them.
- Every step, the images get smaller and smaller!
  - Solution: **padding**



- **The rule of filters:**
  - Each filter extract some features from the image
  - e.g. a filter finds diagonal lines.
  - Filters are learnt by Deep learning process.



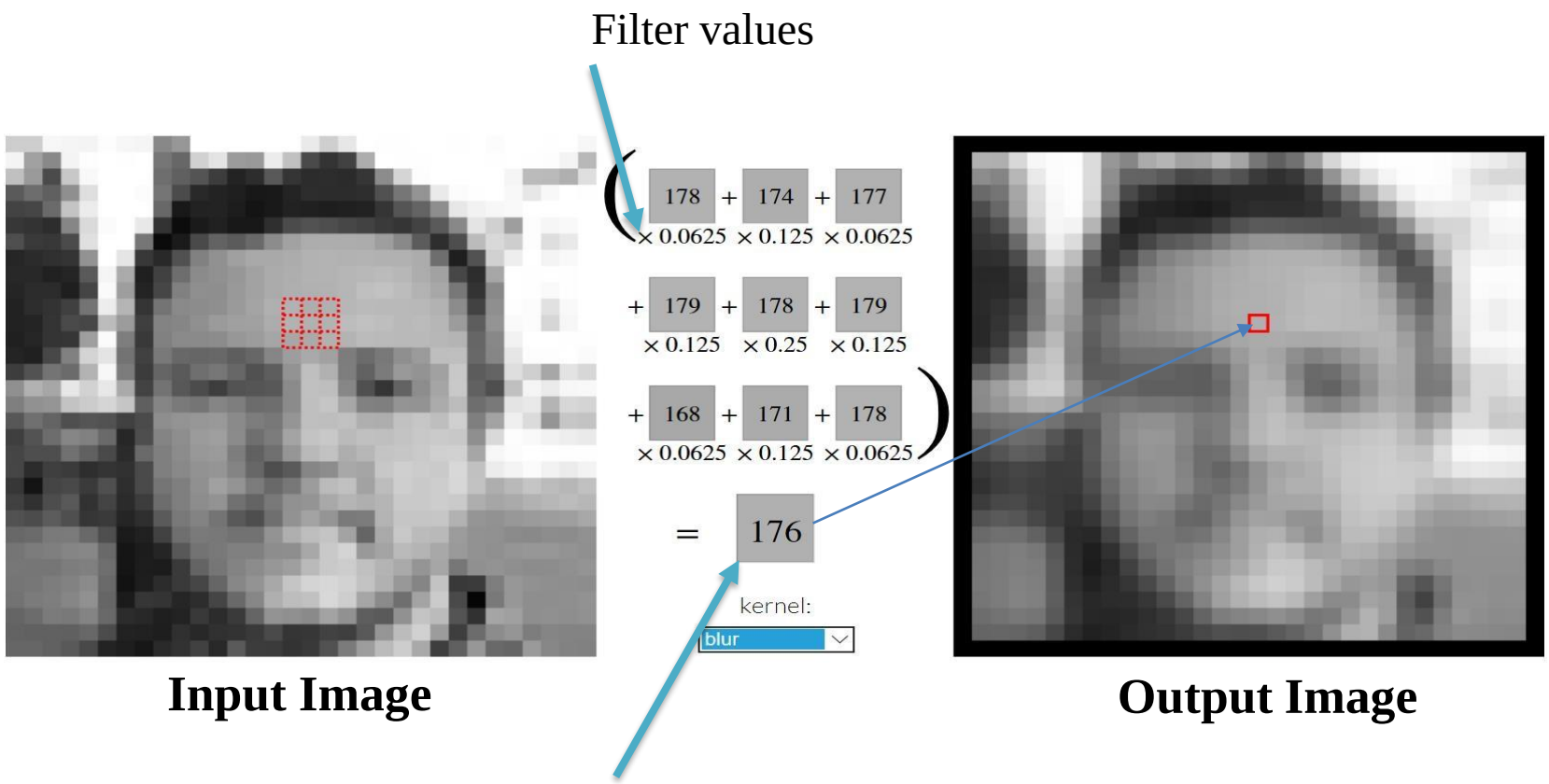
CNN



Convolution layers; convolution two signals e.g,

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2].g[x-n_1,y-n_2]$$

Image Kernels

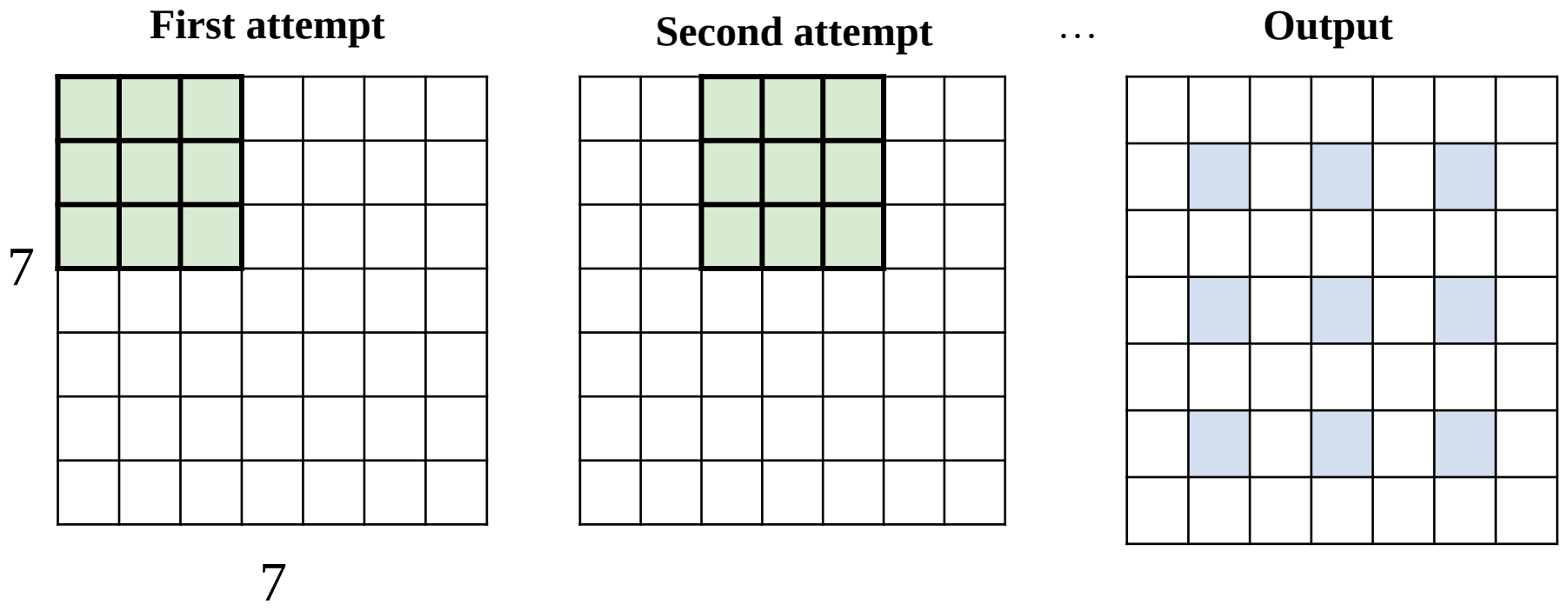


the representative of the 6 pixels in the output layer.

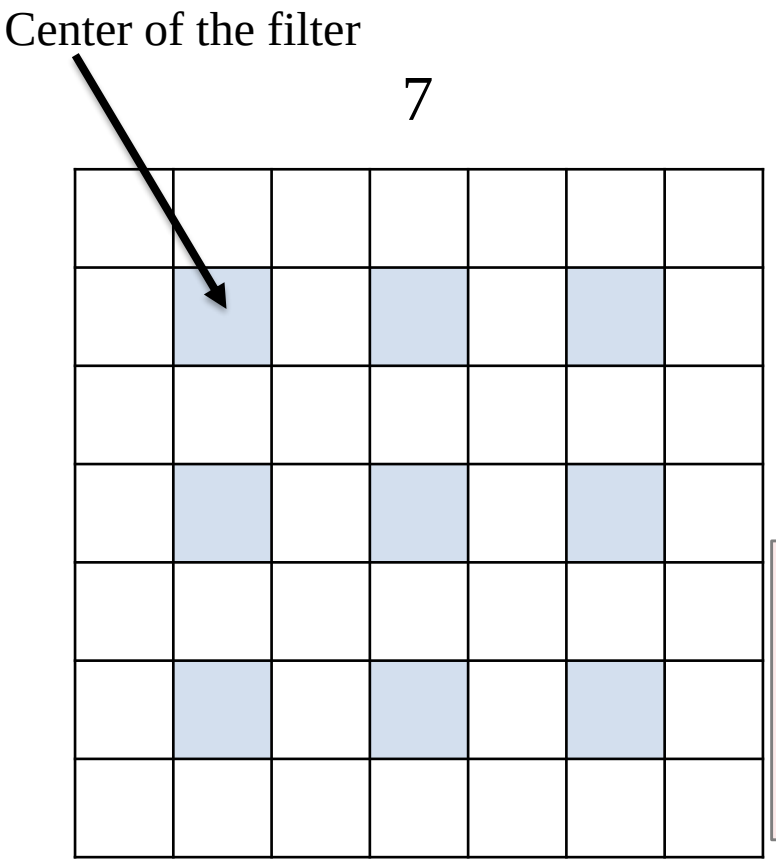
http://setosa.io/ev/image-kernels/

Consider;

- **Image** 7×7
- **Filter** 3×3
- **Step size** 2



- **Output:** a 3×3 image



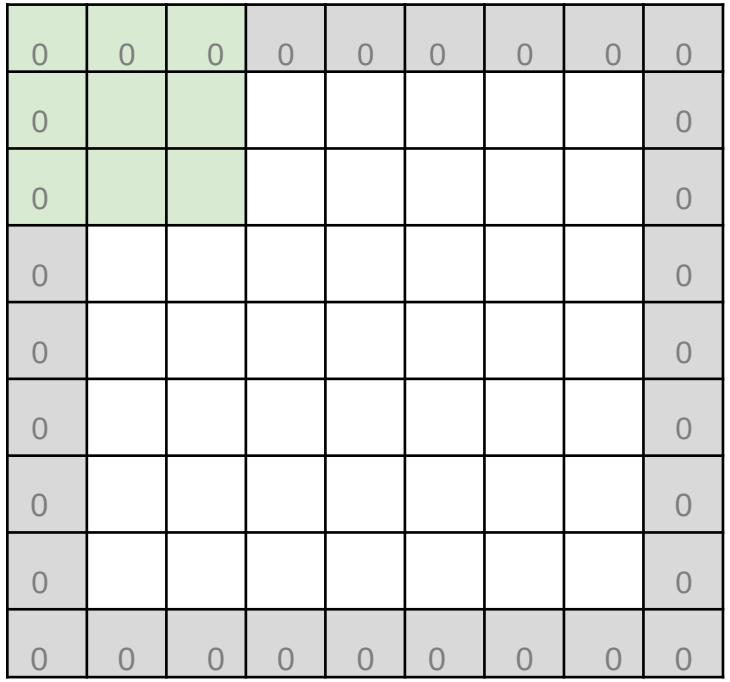
- Output image size

$$\frac{(N-F)}{stride} + 1$$

$N = 7, F = 3$

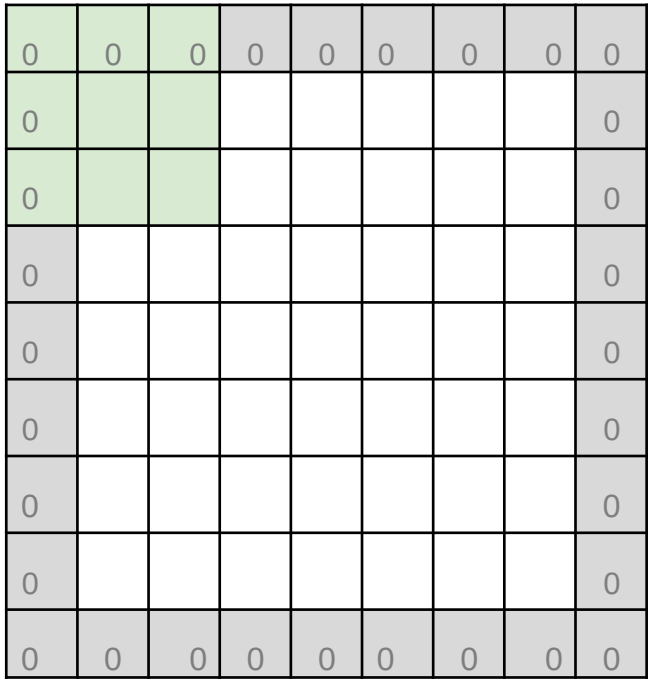
stride 1 => $(7 - 3) / 1 + 1 = 5$ OK
stride 2 => $(7 - 3) / 2 + 1 = 3$ OK
stride 3 => $(7 - 3) / 3 + 1 = 2.33$ NOT possible

- Apply **padding**
- Example: image 7×7
- Filter 3×3
- Stride 1
- We expand the borders by one pixel as Zeros
- Then, the size of output is 7×7

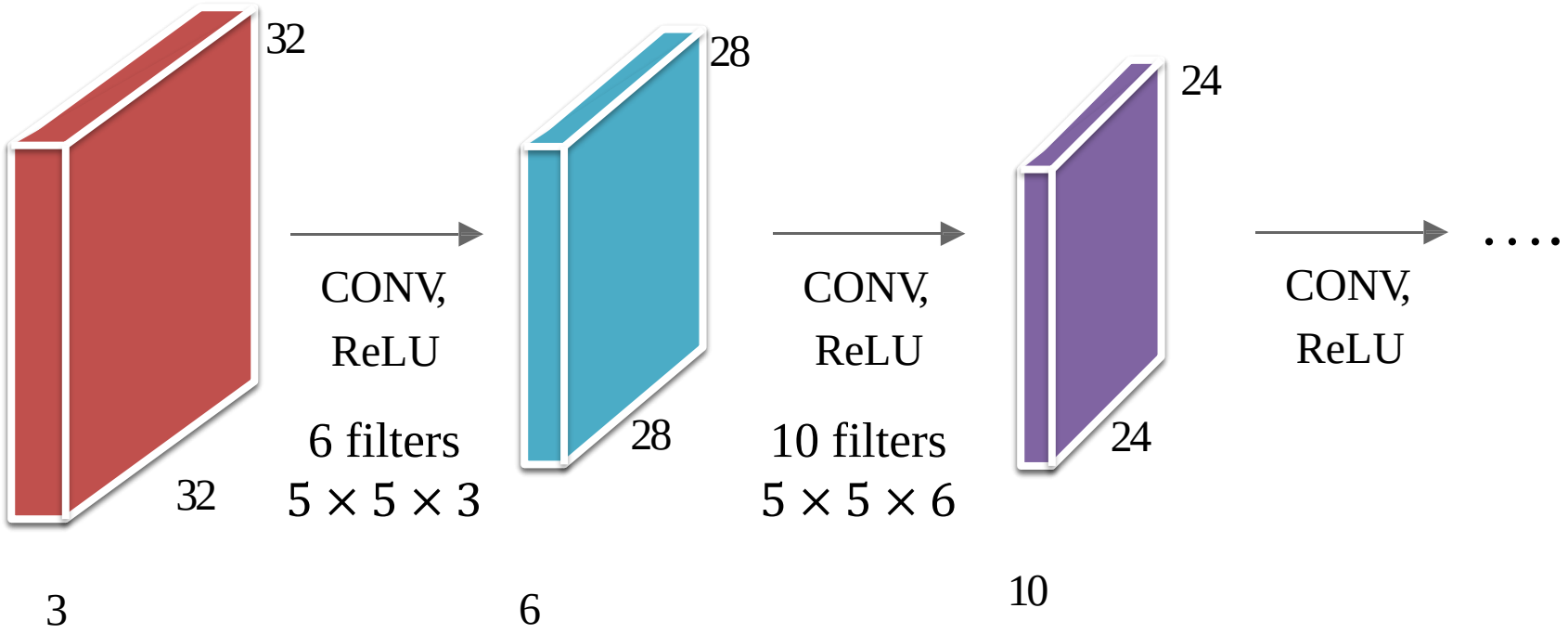


- In general, the convolution layers with 1 stride, the size of filter $F \times F$ and padding is $\frac{F-1}{2}$.

- Example)
  - Filter 3×3 => zero border one pixel
  - Filter 5×5 => zero border 2 pixels
  - Filter 7×7 => zero border 3 pixels



- **Padding:** a solution to overcome the issue of size reduction!

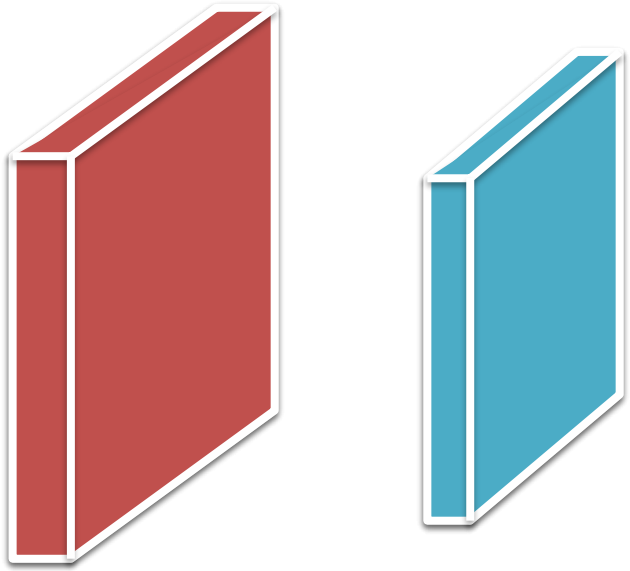


Convolution Layer

Example:

- Image size: $32 \times 32 \times 3$
- 10 filters, 5×5
- Padding 2 pixels
- Output size?

$$\frac{32 - 5 + 2 \times 2}{1} + 1 = 32$$



Size of each activation maps

- Number of parameters:

$$(each\ filter)\ 5 \times 5 \times 3 = 75\ (digit) + 1\ (bias) = 76$$
$$76 \times 10\ (filters) = 760$$

- If we want to use fully connected networks we have to have

$$3072 \times 10 = 30720$$

- So, **CNN has less number of parameters**.

Summary

Input: width, height, depth: $W_1 \times H_1 \times D_1$

We need 4 parameters:

- **Number of filters** K
- **Filter dimensions** $F \times F$
- **Stride** S
- **Padding** P

Output : $W_2 \times H_2 \times D_2$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

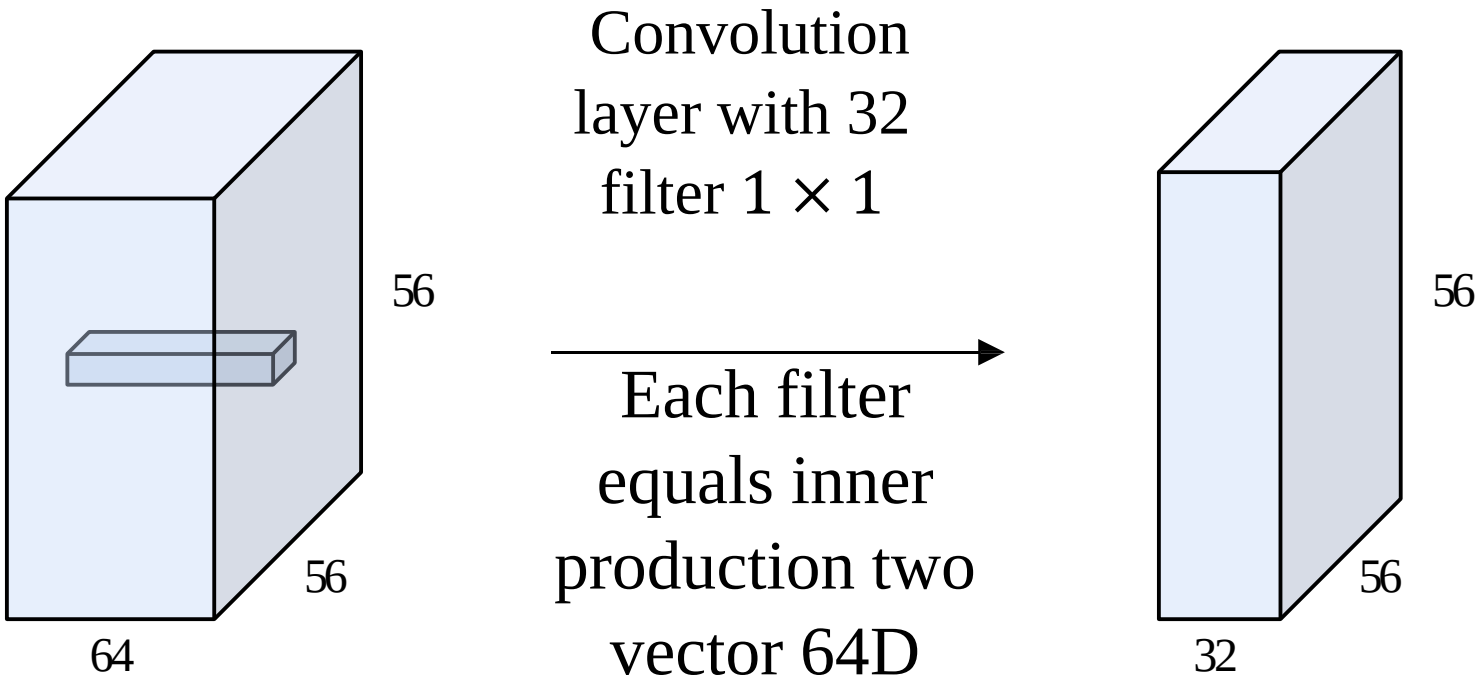
$$D_2 = K$$

Weights for each filter: $F \times F \times D_1$

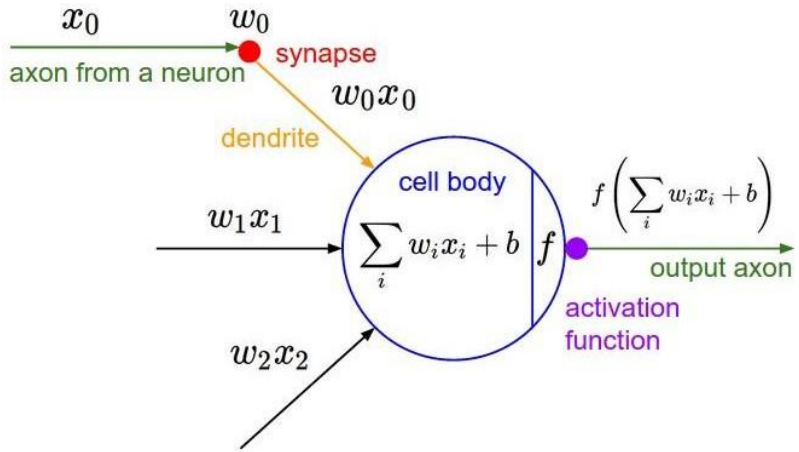
Total **weights**: $(F \times F \times D_1) \times K$

'K' **Bias**

Convolution with more depth



From Neuron perspective



A neuron with local links

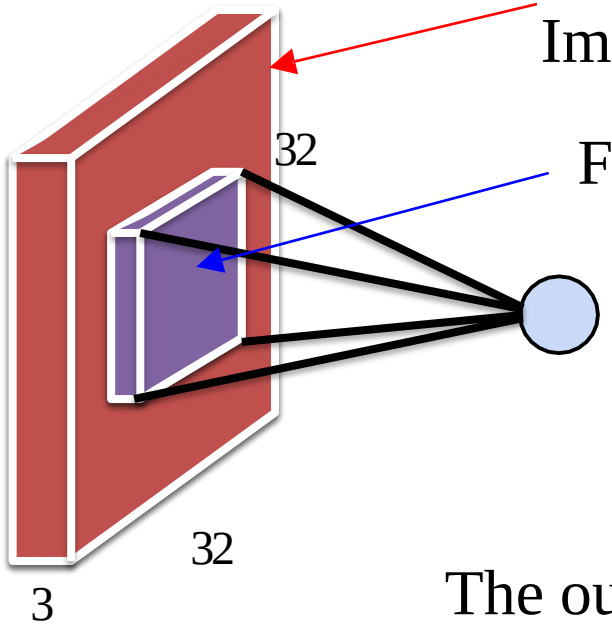
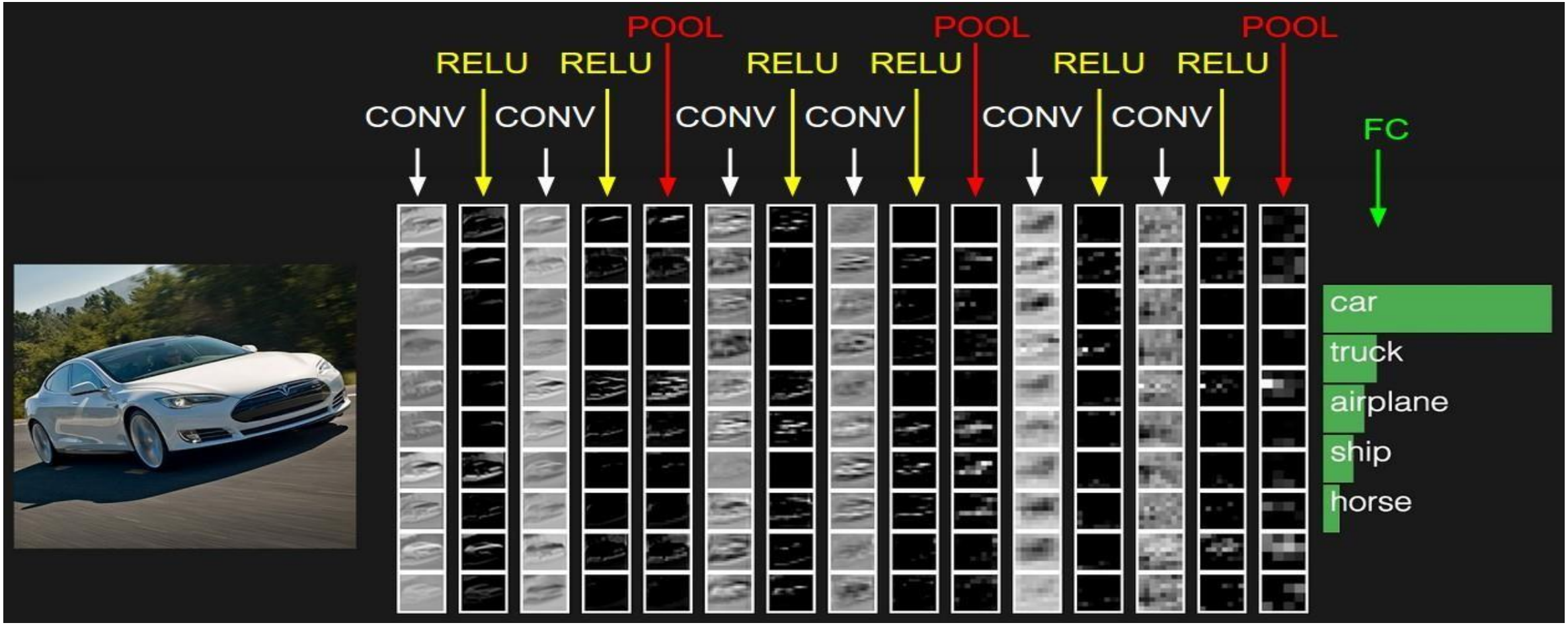


Image $32 \times 32 \times 3$

Filter $5 \times 5 \times 3$

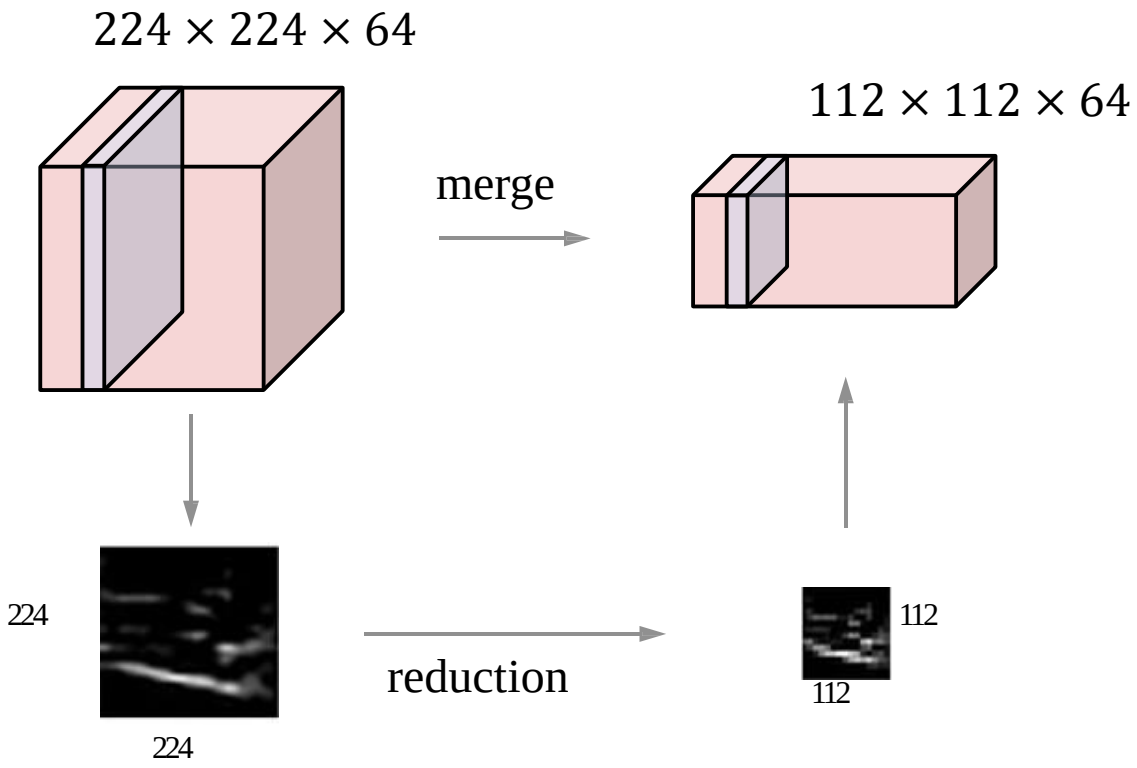
1 digit

The output of the inner production


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

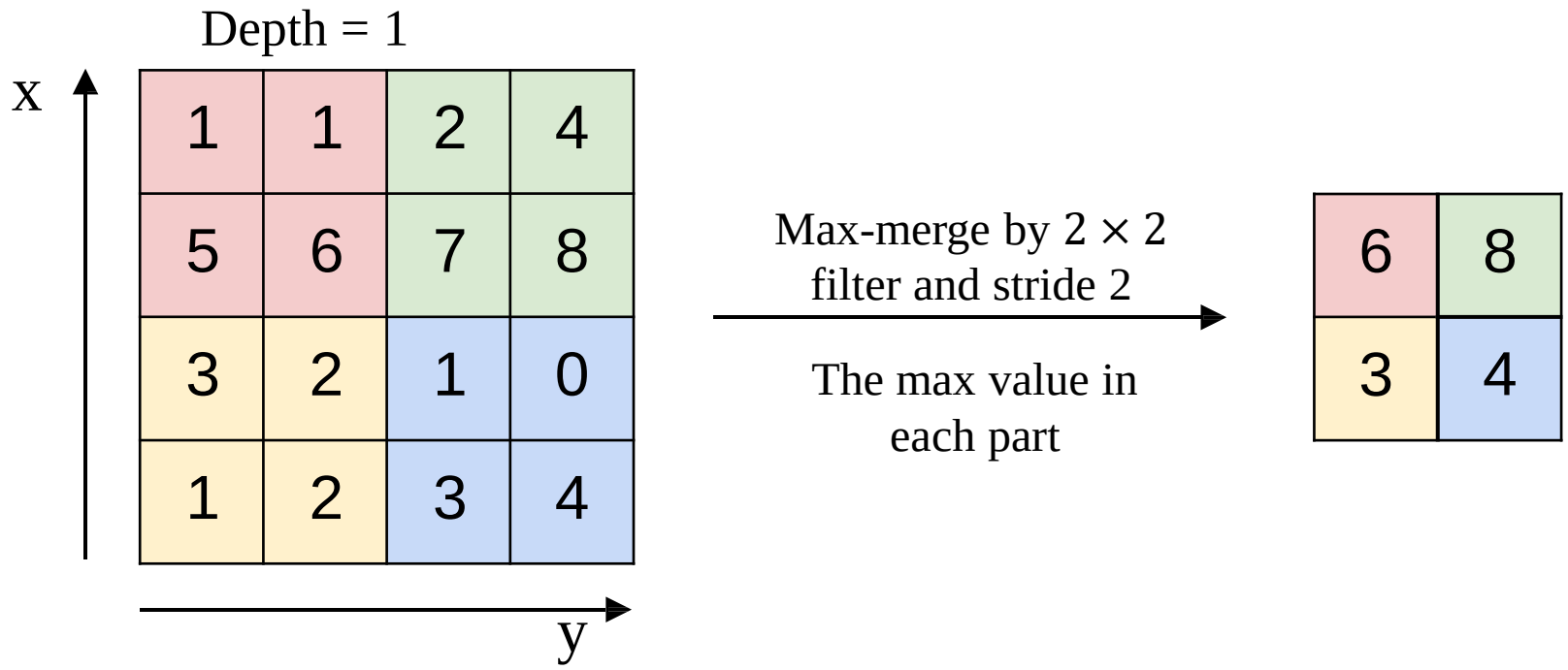
Pooling

- Merge layer: reduce the size of output
- Apply on each activation map separately



Pooling

- Merge layer: reduce the size of output
- Apply on each activation map separately



Merge layer

Input: $W_1 \times H_1 \times D_1$

We need 2 parameters:

- Filter dimensions $F \times F$
- Stride S

Output : $W_2 \times H_2 \times D_2$

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = (H_1 - F)/S + 1$$

$$D_2 = D_1$$

$$Number\ of\ parameters\ =\ 0$$

Padding is not recommended for the merge layers.

Sources

- M. Graph, Deep Learning with Python
- G. Bonaccorso, Mastering Machine Learning Algorithms
- U. Michelucci, Advanced Applied Deep Learning
- http://cs231n.stanford.edu/

Office: #432, Dayang AI Center, **Phone**: 010-8999-8586, **email**: *piran@sejong.ac.kr*