

4a. Convolutional Neural Networks

Never Stand Still

Faculty of Engineering

COMP9444 Week 4a

Sonit Singh

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales, Sydney, Australia

sonit.singh@unsw.edu.au

Learning Outcomes

By the end of this module, you will:

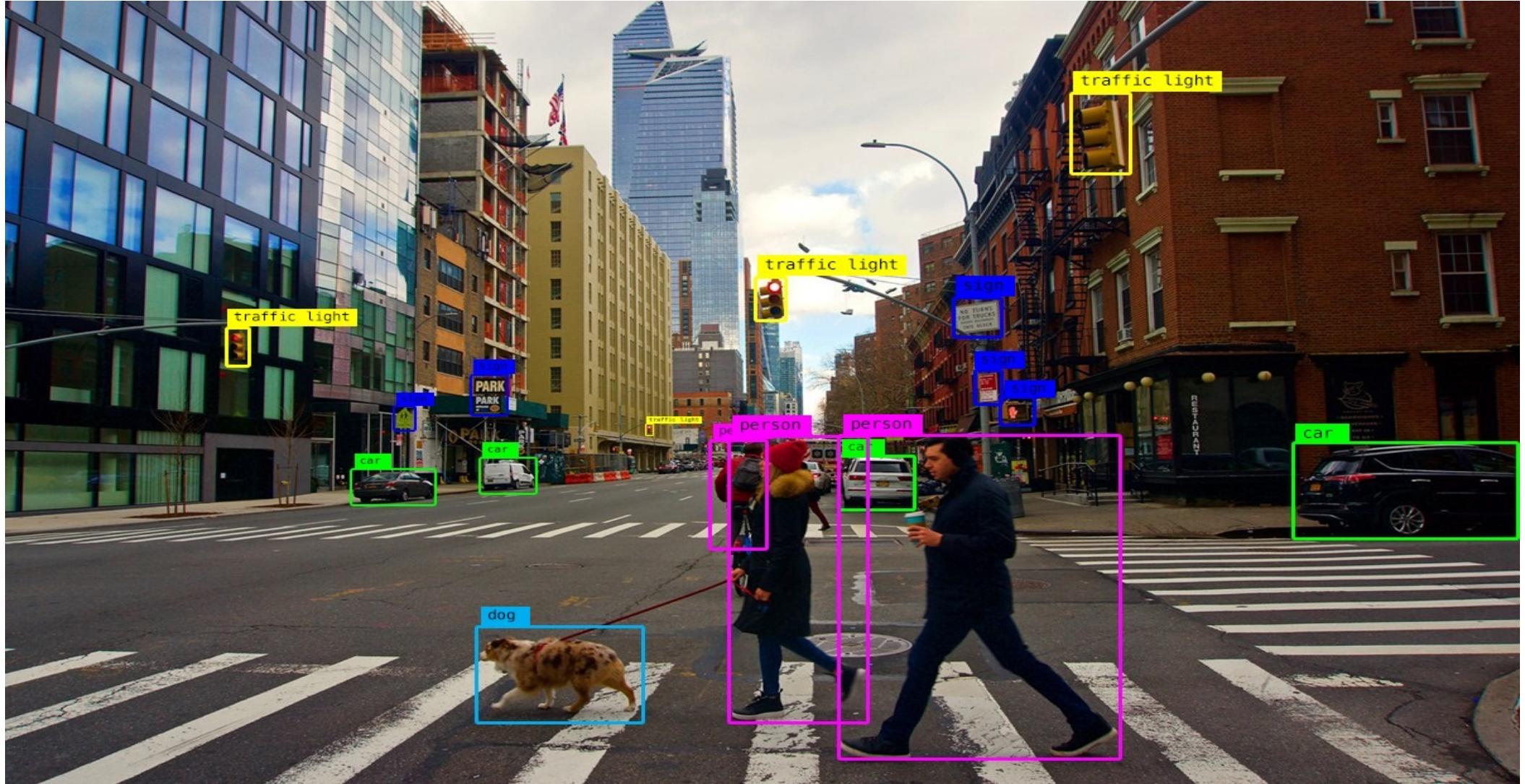
- learn “Why we need CNNs for Images/Videos?”
- learn building blocks of a typical CNN architecture
- describe convolutional network, including convolution operator, stride, padding, max pooling
- Identify the limitations of 2-layer neural networks
- explain the problem of vanishing or exploding gradients, and how it can be solved using techniques such as weight initialization, batch normalization, skip connections, dense blocks
- design a customized CNN model using building blocks
- learn how we can apply CNNs to solve various computer vision problems

Agenda

- What is Computer Vision?
- What are various applications of Computer Vision?
- What are Convolution Neural Networks (CNNs)?
- What are typical building blocks of a CNN?
- How can we arrange these building blocks to design a customized CNN architecture?
- Walkthrough of LeNet

Computer Vision

- Enabling machines to process, represent, understand, and generate visual data



How computers see images?

- ## ➤ A matrix of numbers



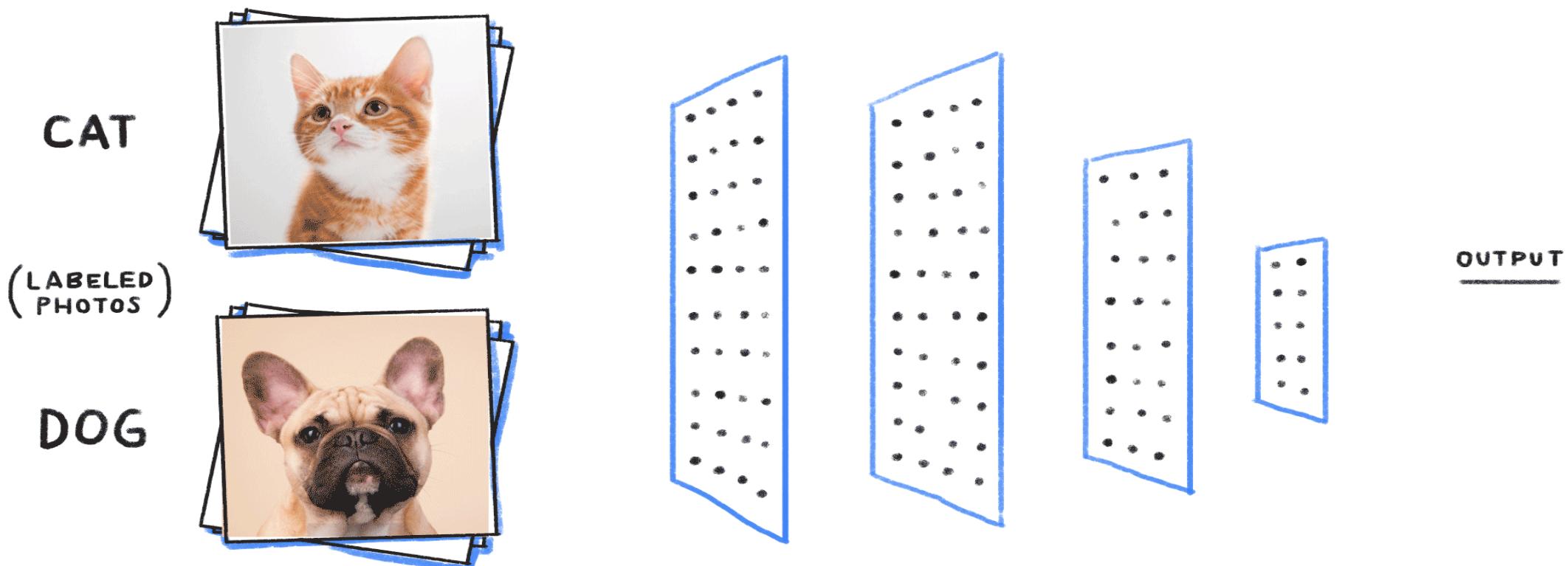
What Computer Sees

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	377	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	45
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	123	252	255	248	144	6	0
0	13	111	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5117	251	255	241	255	247	255	241	162	17	0	7	0	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	65	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	120	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5117	251	255	241	255	247	255	241	162	17	0	7	0	
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	249	255	240	255	129	0	5	
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

CV Applications

➤ Image Classification



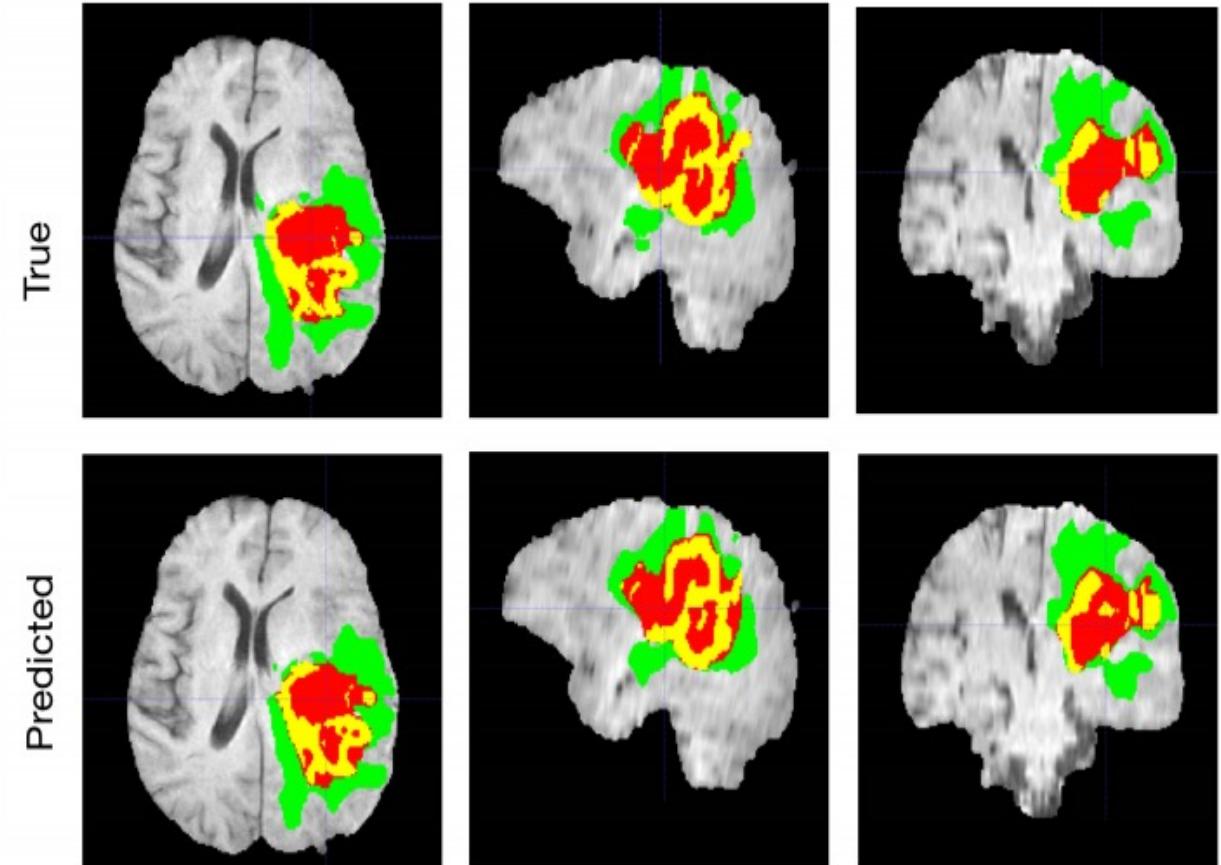
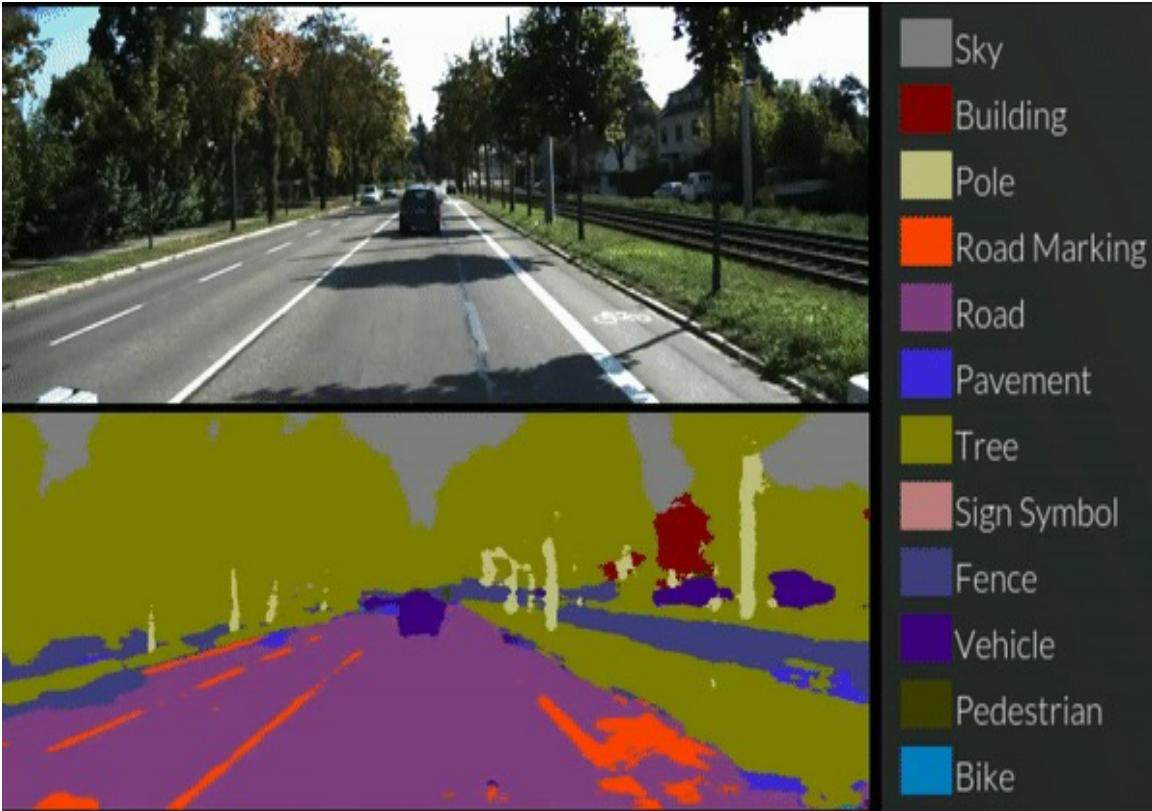
CV Applications

➤ Object Detection

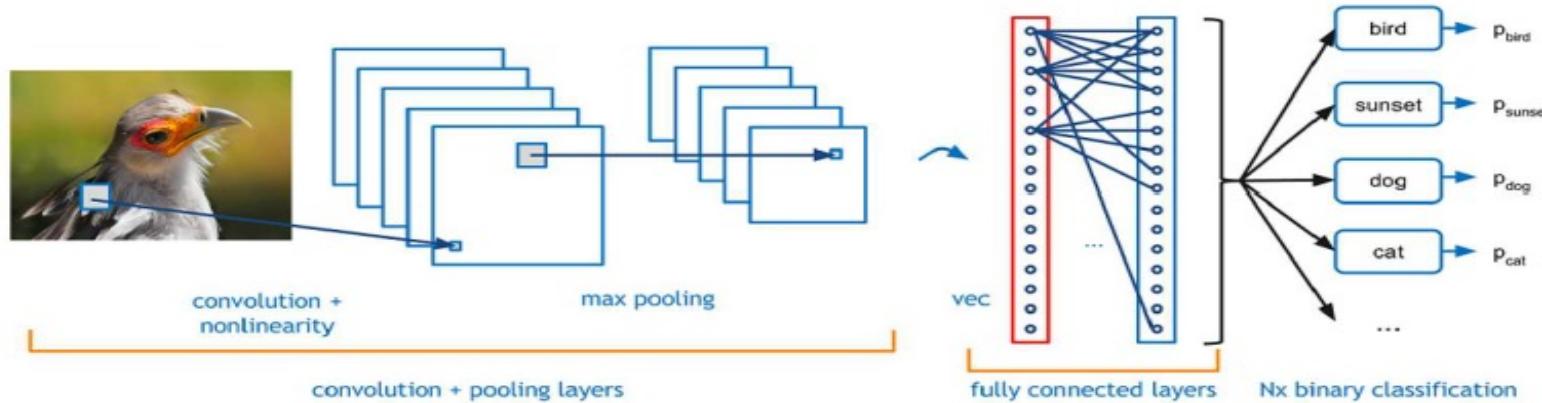


CV Applications

➤ Segmentation

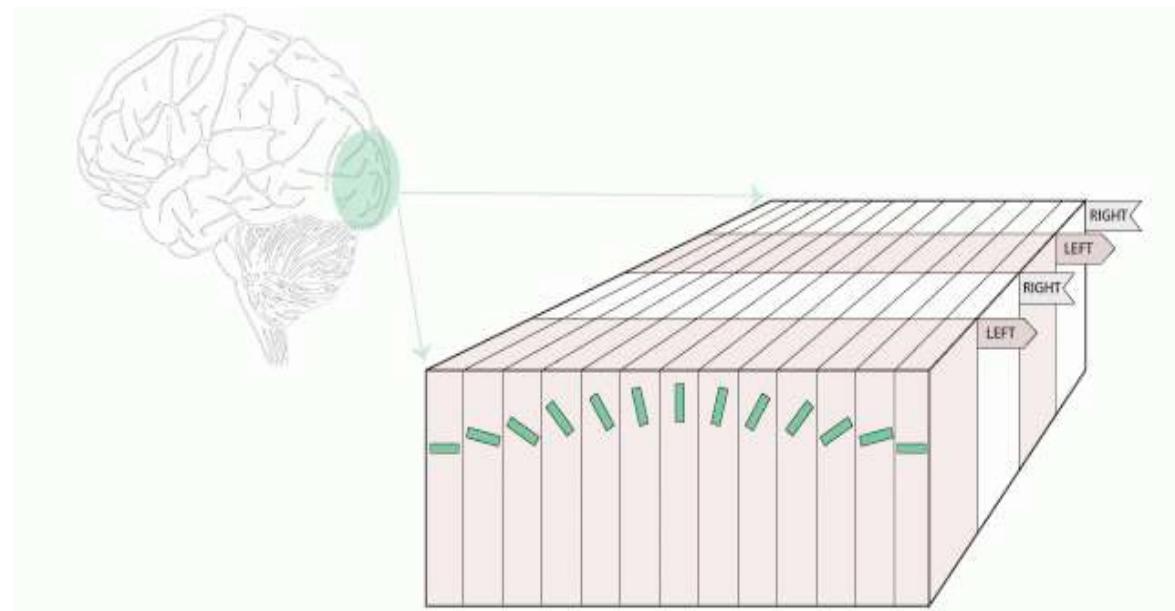
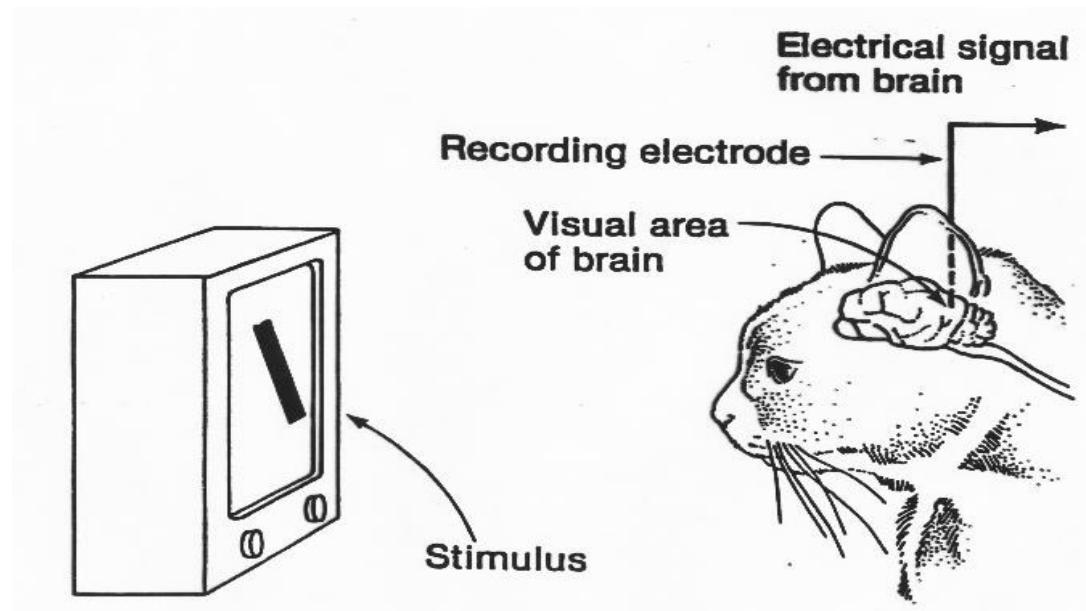


Convolutional Networks



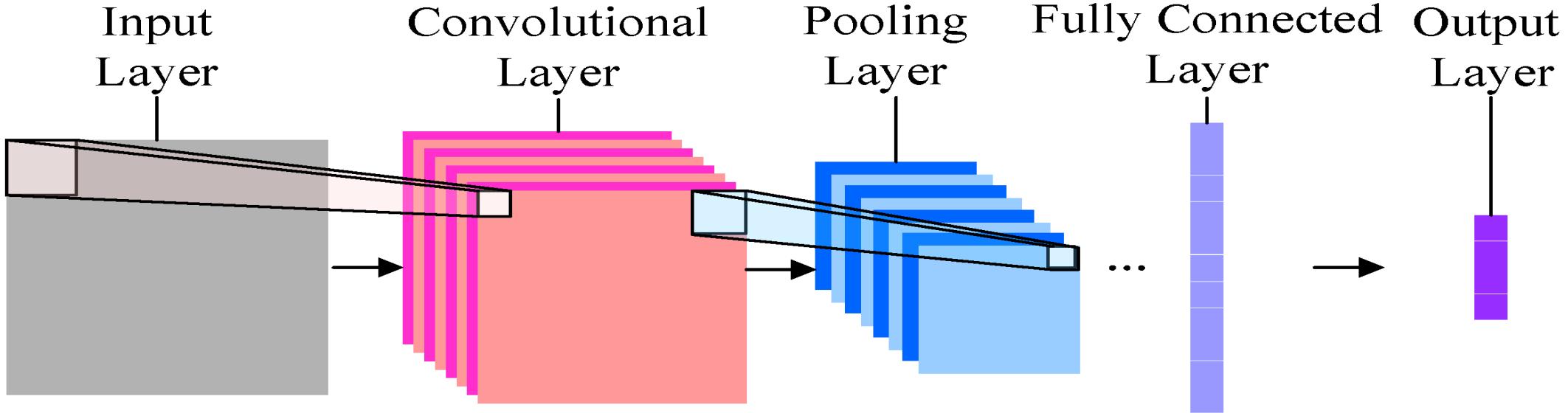
- Suppose we want to classify an image as a bird, sunset, dog, cat, etc.
- If we can identify features such as feather, eye, or beak which provide useful information in one part of the image, then those features are likely to also be relevant in another part of the image.
- We can exploit this regularity by using a convolution layer which applies the same weights to different parts of the image.

Hubel and Weisel – Visual Cortex



- cells in the visual cortex respond to lines at different angles
- Cells in V2 respond to more sophisticated visual features
- Convolutional Neural Networks are inspired by this neuroanatomy
- CNN's can now be simulated with massive parallelism, using GPU's

Convolutional Network Components



- **Convolutional layers:** extract shift-invariant features from the previous layer
- **Subsampling or pooling layers:** combine the activations of multiple units from the previous layer into one unit
- **Fully connected layers:** collect spatially diffuse information
- **Output layer:** choose between classes

MNIST Handwritten Digit Dataset



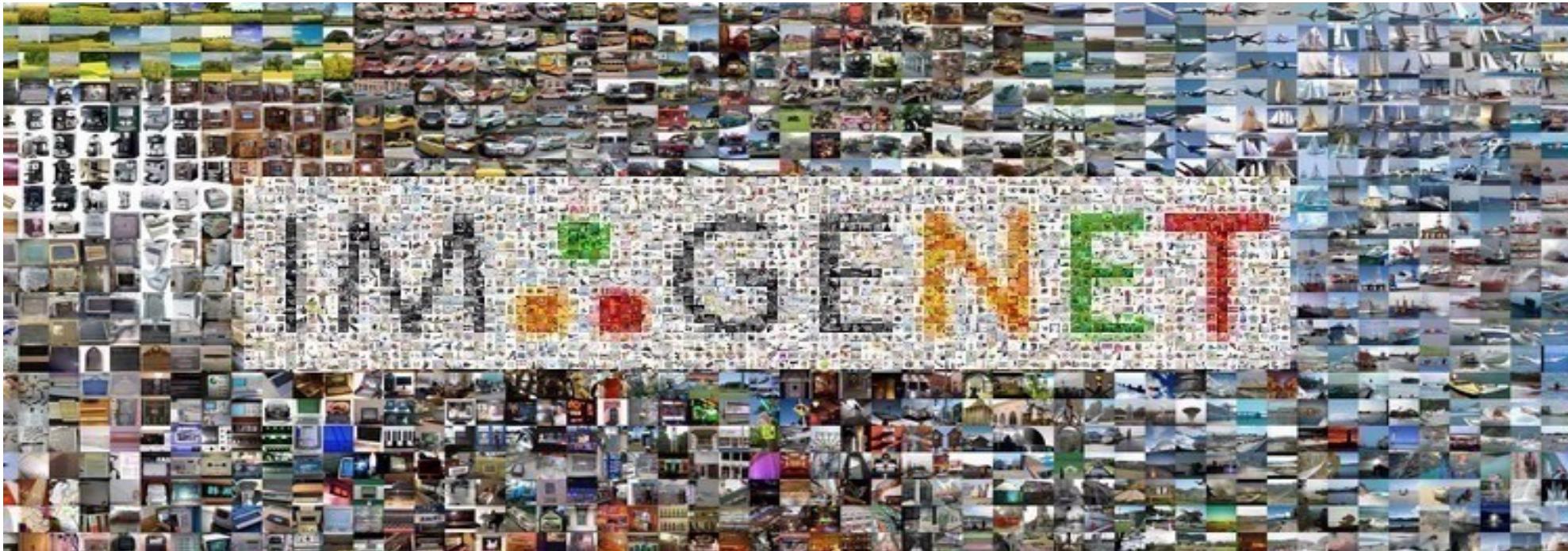
- black and white images, resolution 28 x 28
- 60,000 images
- 10 classes (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

CIFAR Image Dataset



- color images, resolution 32 x 32
- 50,000 images
- 10 classes

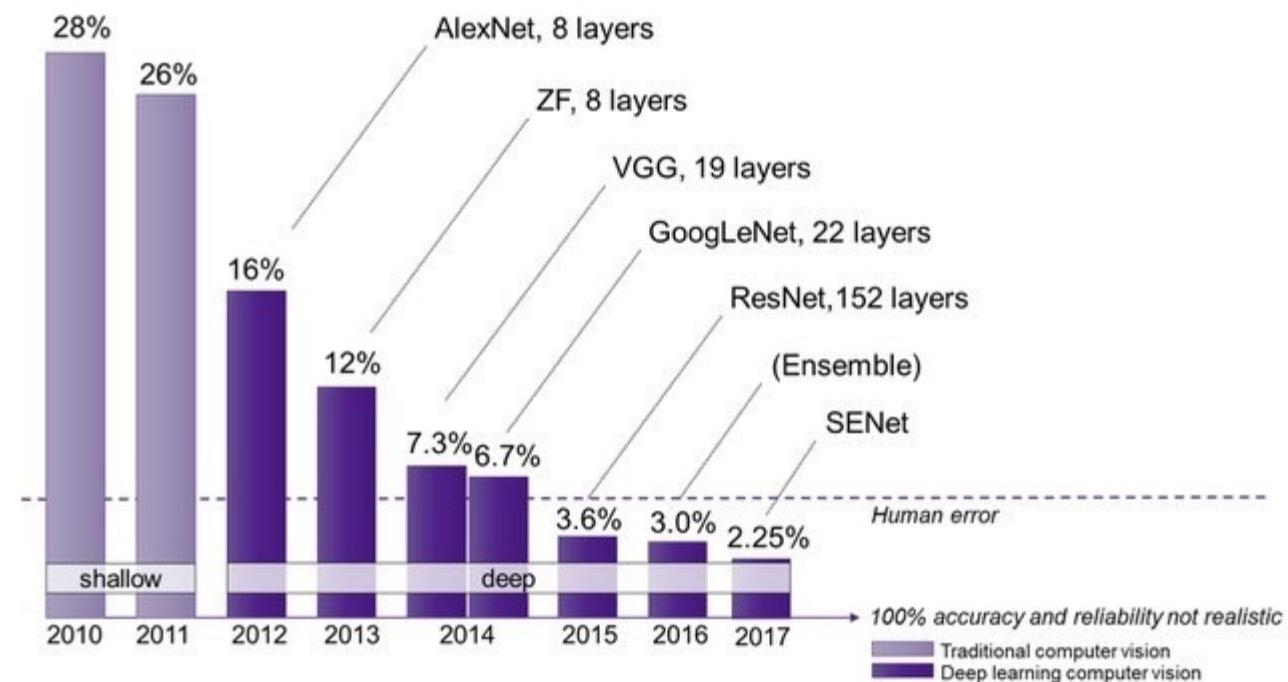
ImageNet LSVRC Dataset



- color images, resolution 227 x 227
- 1.2 million images
- 1000 classes

Convolutional Neural Networks (CNNs)

- A class of deep neural networks suitable for processing 2D/3D data. For e.g., Images and Videos
- CNNs can capture high-level representation of images/videos which can be used for end-tasks such as classification, object detection, segmentation, etc.
- A range of CNNs improving over the years



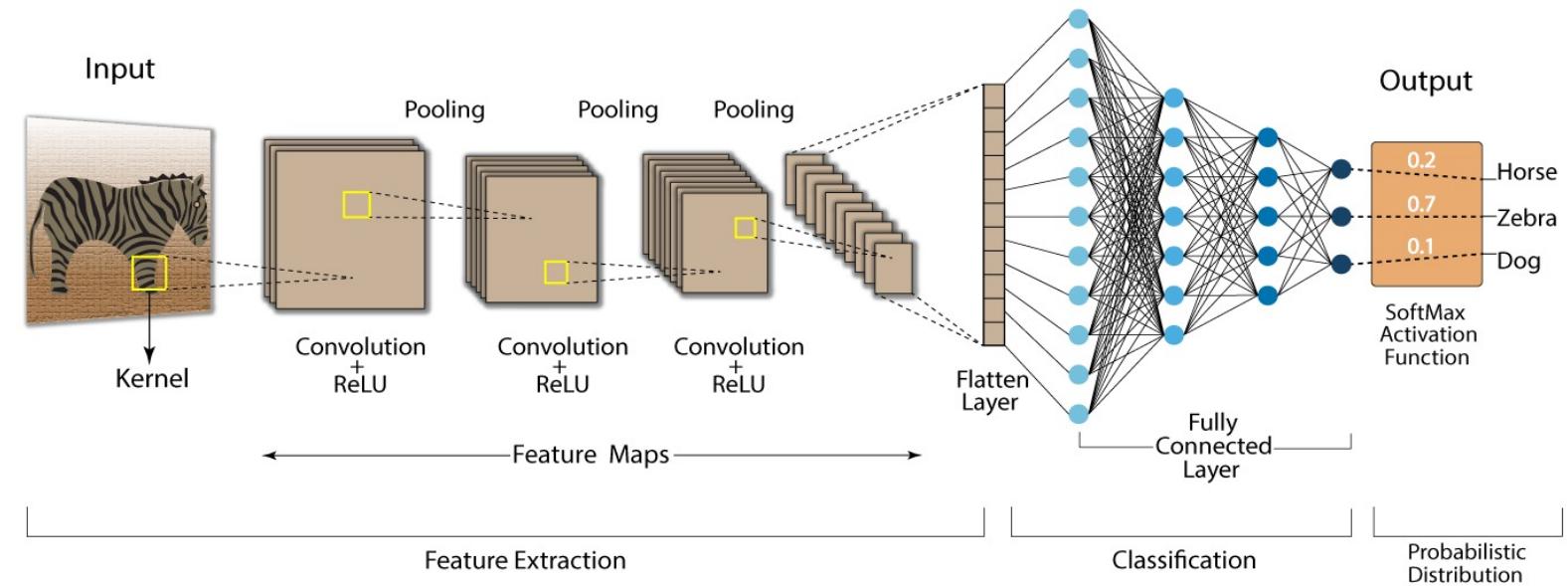
Why CNNs?

- Regular Neural Networks (ANNs) don't scale well with increasing dimensions
 - Image of dimensions 8×8 : Considering color image having 3 channels, the input dimensions of the neural network would be $3 \times 8 \times 8 = 192$ units (**Manageable**)
 - Image of dimensions 500×500 : The input dimensions of the neural network would be $3 \times 500 \times 500 = 750,000$ units (**Complexity and the number of parameters increasing exponentially -> unmanageable and inefficient**)
- CNNs leverage important ideas:
 - **sparse interaction** : Making kernel smaller than the input e.g., an image, we can detect meaningful information that is much smaller than thousands or millions of pixels. This means storing fewer parameters requiring less memory.
 - **parameter sharing** : In CNNs we have weight sharing. For getting output, weights applied to one input are the same as the weight applied elsewhere.
 - **equivariant representation** : Due to weight sharing, CNN have a property of equivariance to translation, which means if we change the input in a way, the output will also get changed in the same way.

CNN Architecture

- A typical CNN architecture consists of the following layers:

- Convolution layer
- ReLU layer (non-linearity)
- Pooling layer
- Flattening
- Fully-connected layer
- Output layer



- There can be multiple steps of convolution followed by pooling, before reaching the fully connected layers.

Convolution Operator

Continuous convolution

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da$$

Discrete convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

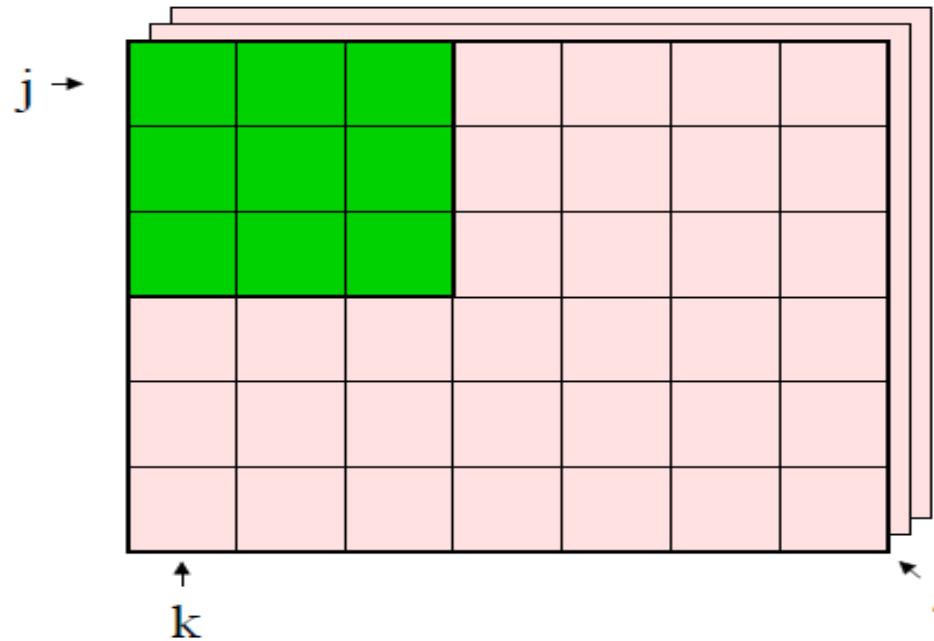
Two-dimensional convolution

$$S(j, k) = (K * I)(j, k) = \sum_m \sum_n K(m, n)I(j+m, k+n)$$

Note: Theoreticians sometimes write $I(j-m, k-n)$ so that the operator is commutative.

But, computationally, it is easier to write it with a plus sign.

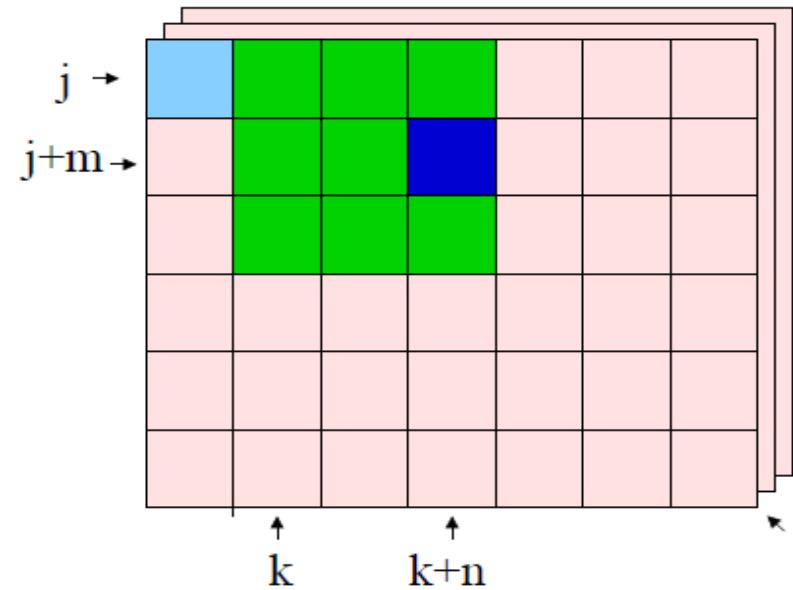
Convolutional Neural Networks



Assume the original image is $J \times K$, with L channels.

We apply an $M \times N$ “filter” to these inputs to compute one hidden unit in the convolution layer.
In this example, $J = 6$, $K = 7$, $L = 3$, $M = 3$, $N = 3$

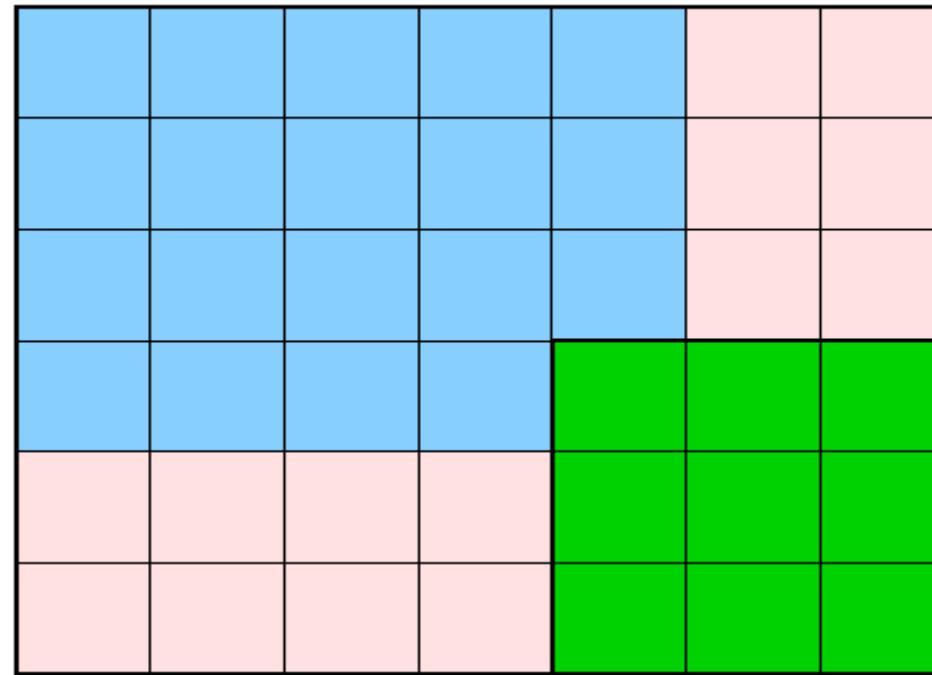
Convolutional Neural Networks



$$Z_{j,k}^i = g(b^i + \sum_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K_{l,m,n}^i V_{j+m, k+n}^l)$$

The same weights are applied to the next $M \times N$ block of inputs, to compute the next hidden unit in the convolution layer (“weight sharing”).

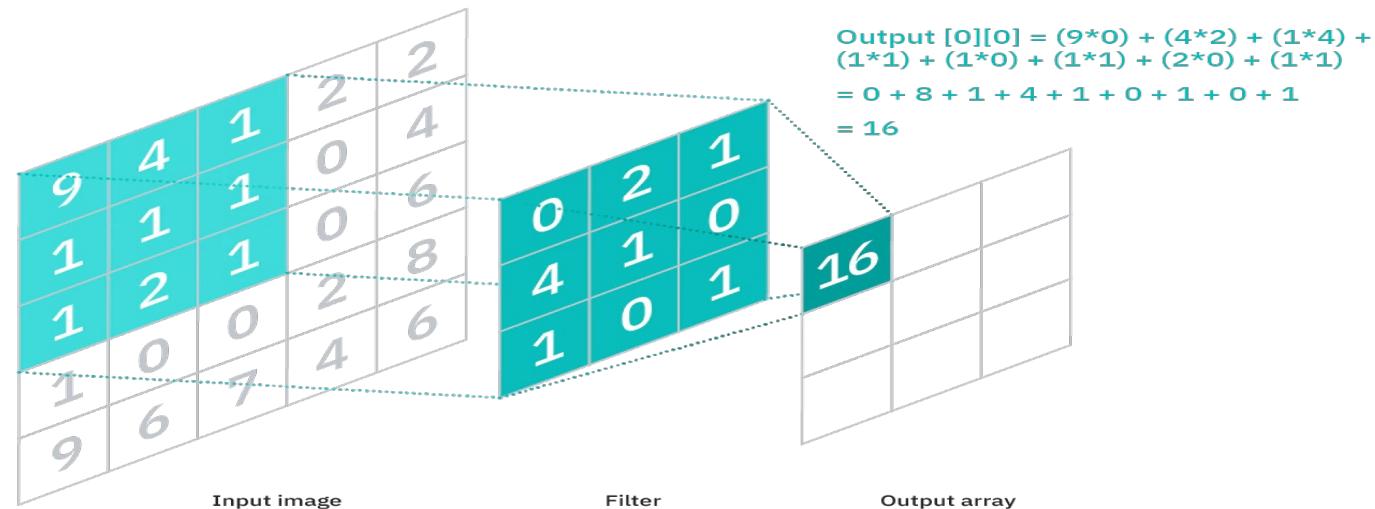
Convolutional Neural Networks



If the original image size is $J \times K$ and the filter size is $M \times N$, the convolution layer will be $(J + 1 - M) \times (K + 1 - N)$

Convolution Layer

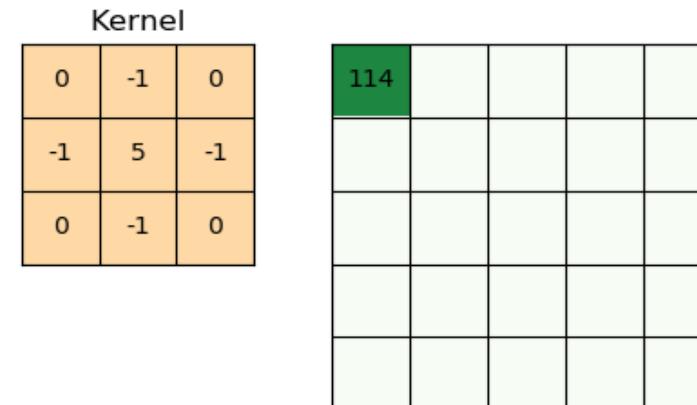
- The convolution layer detect features or visual features in images such as edges, lines, etc.
- The kernel (also, known as filter or feature detector) move across the receptive field of the image and extracts important features
- The filter carries out a convolution operation which is an element-wise product and sum between two matrices
- Given output value in the feature map does not have to connect to each pixel in the input image, it only needs to connect to the receptive filed, where the filter is being applied. This characteristic is described as “local connectivity”



Convolution in action

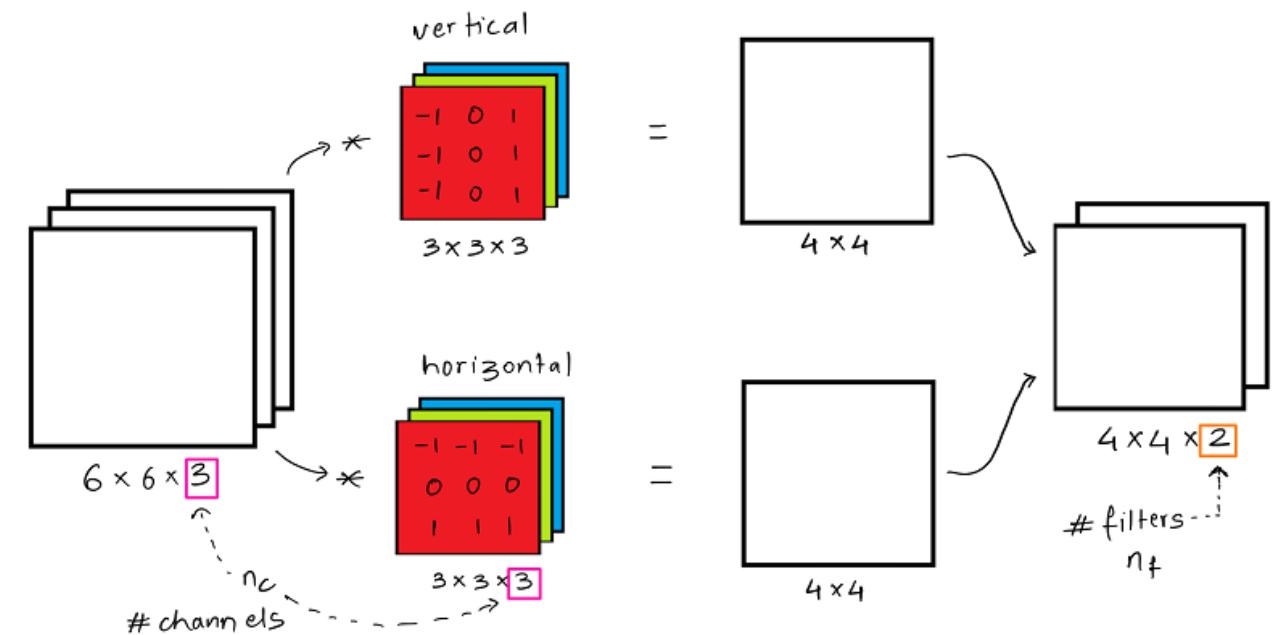
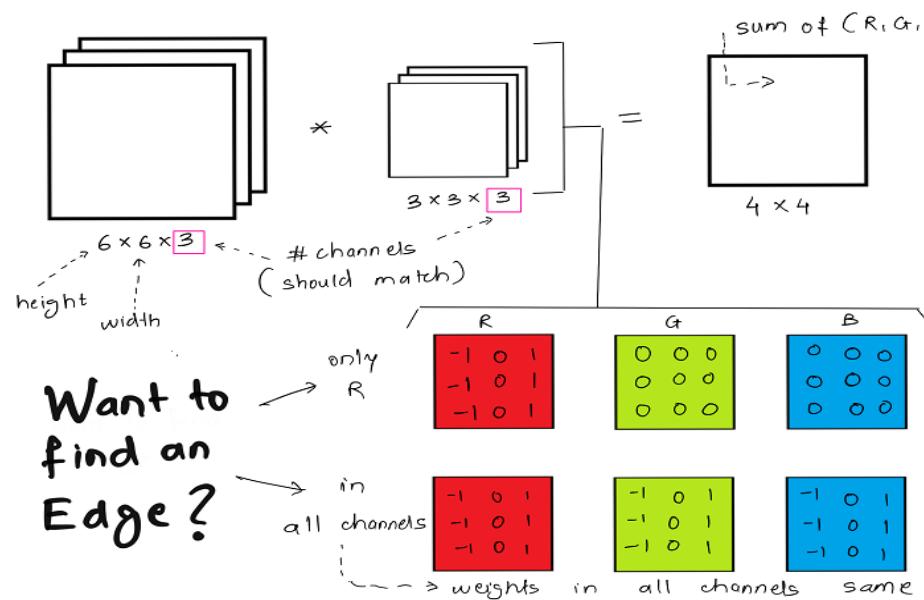
- We need multiple number of filters (feature detectors) to detect different curves/edges or features in the image
- The result of convolving filter over the entire image is an output matrix called feature maps or convolved features that stores the convolutions of the filter over various parts of the image
- N filters produces N feature maps

0	0	0	0	0	0	0	0
0	60	113	56	139	85	0	0
0	73	121	54	84	128	0	0
0	131	99	70	129	127	0	0
0	80	57	115	69	134	0	0
0	104	126	123	95	130	0	0
0	0	0	0	0	0	0	0



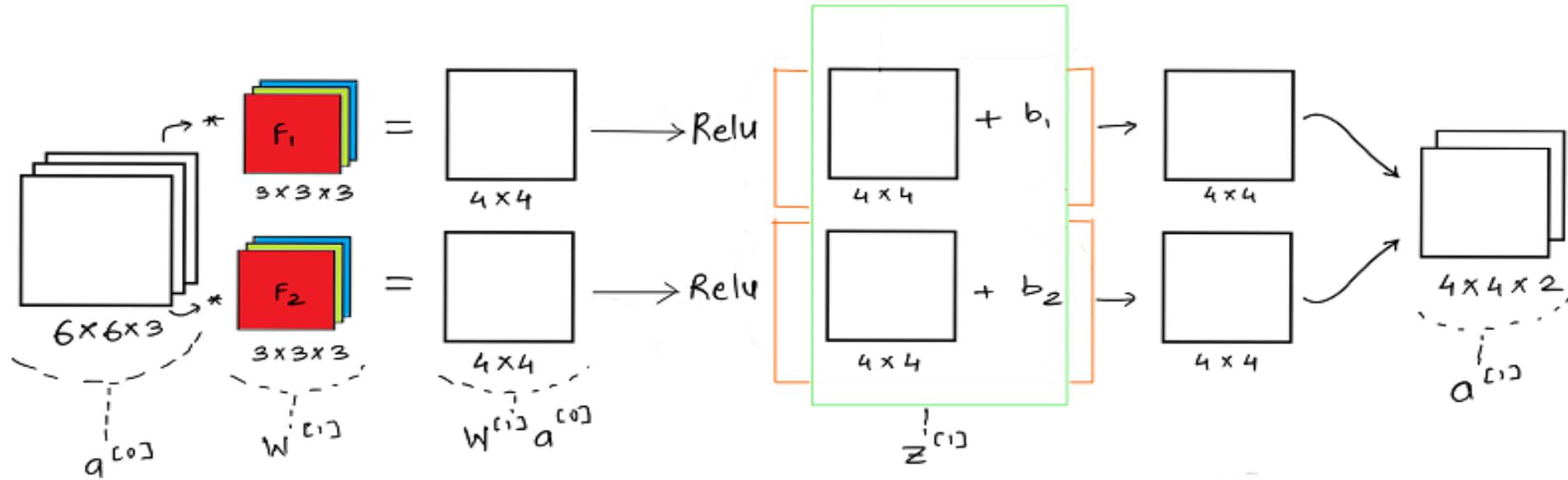
Convolution over color images (3 channels)

- If the image dimension is $N \times N \times \# \text{channels}$, so the filter would also be of dimension $f \times f \times \# \text{channels}$
- For extracting different features from an image, we need multiple filters



Convolution Layer

- One layer of Convolution over color image, having 2 filters



In any given neural network

ReLU function

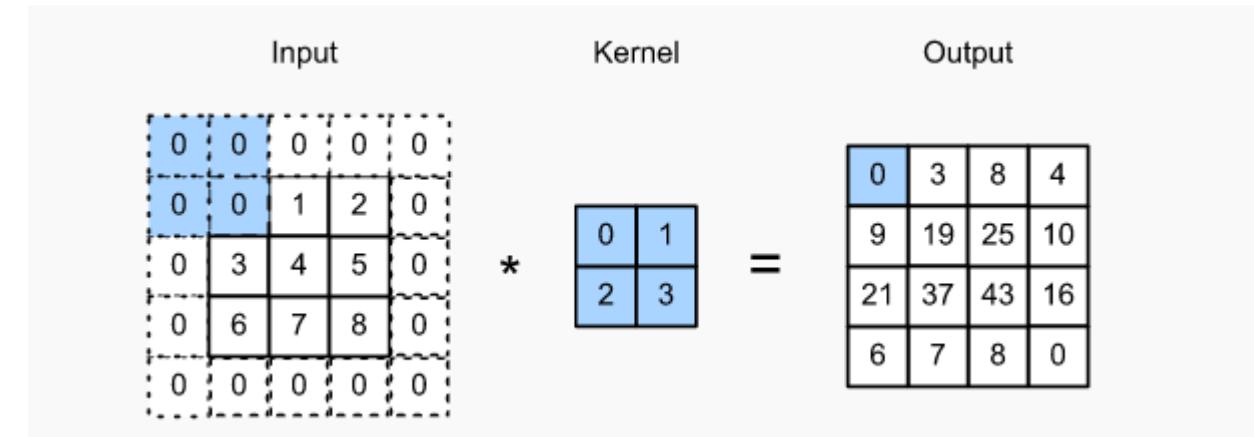
$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

Convolution Layer

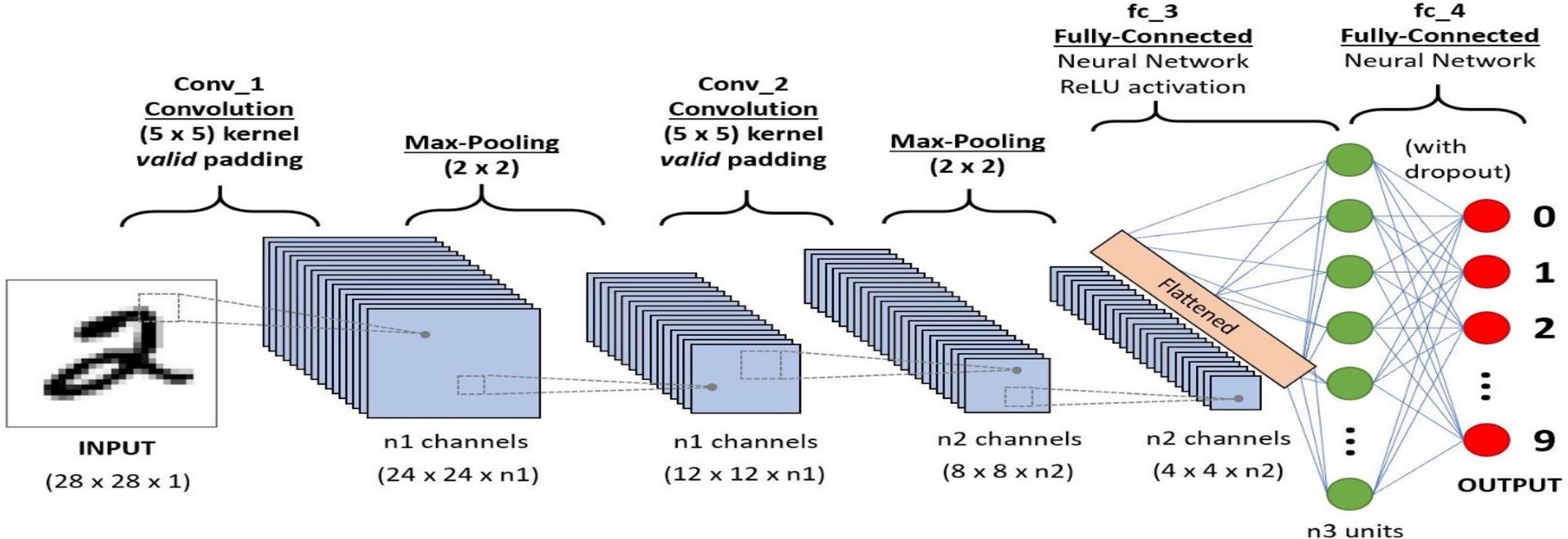
- Three hyperparameters affect the volume size of the output:
 - **Number of filters:** N distinct filters would yield N different feature maps
 - **Stride:** The distance or the number of pixels, that the kernel moves over the input matrix.
 - **Padding:** Refers to the number of pixels added to an image when it is being processed by the kernel of a CNN. Zero padding, which set elements to zero which fall outside of the input matrix, is usually used when the filters do not fit the input image.
- If we have an image of size $W \times W \times D$, a kernel of spatial size F , and stride S , then the size of output volume can be determined by the formula. The output volume is of size $W_{out} \times W_{out} \times D$

$$W_{out} = \frac{W - F}{S} + 1$$



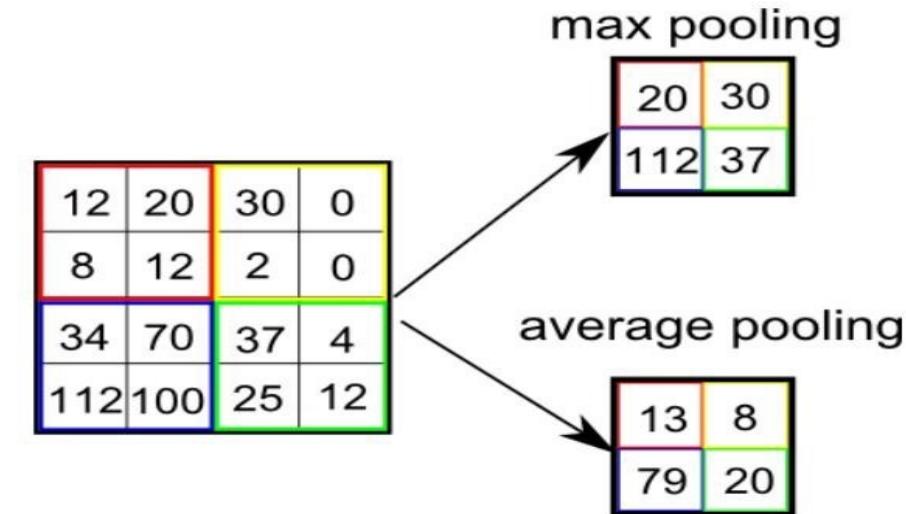
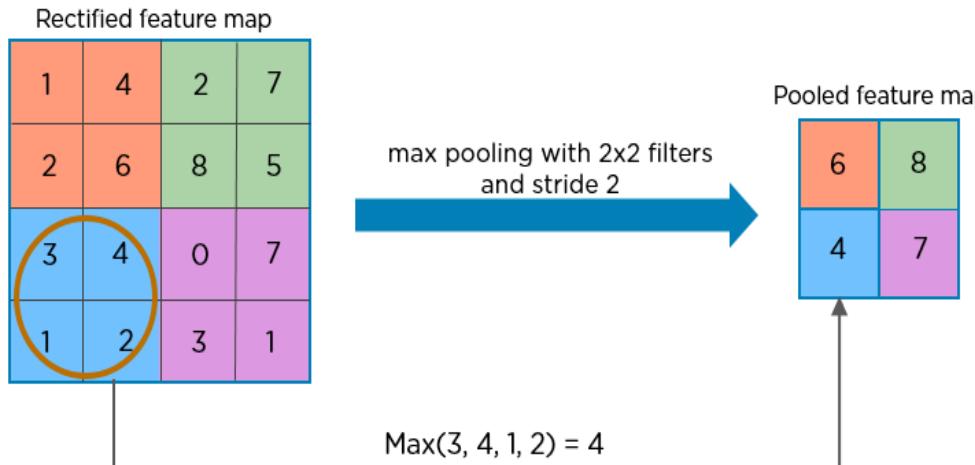
Convolution Layer

- Let's get numbers right



Subsampling or Pooling Layer

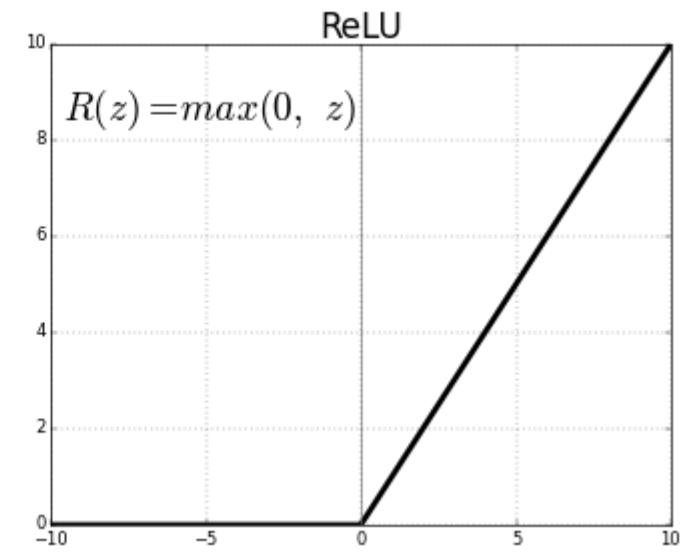
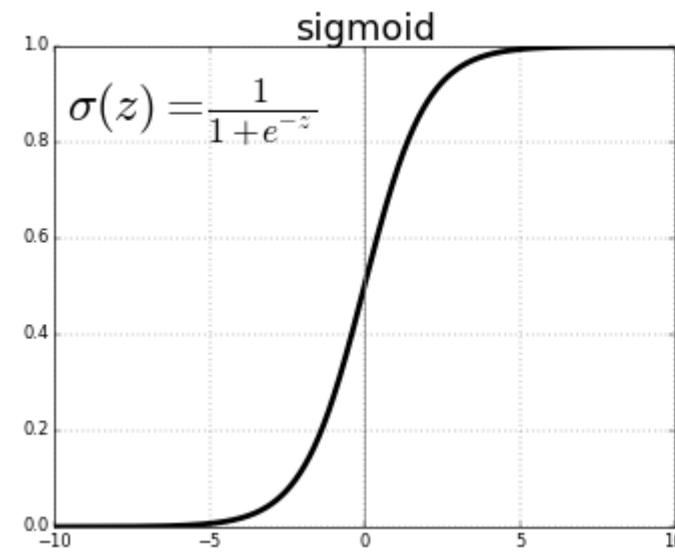
- Pooling is a down-sampling operation that reduces the dimensionality of the feature map.
- Each ReLU feature map pass through the pooling layer to generate a pooled feature map.
- Two of the important pooling operations are:
 - Max pooling: As the filter moves across the input, it selects the pixel with the maximum value to send to the output array
 - Average pooling: As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.



- Although there is information loss due to pooling, it helps to reduce complexity, improve efficiency, and limit risk of overfitting.

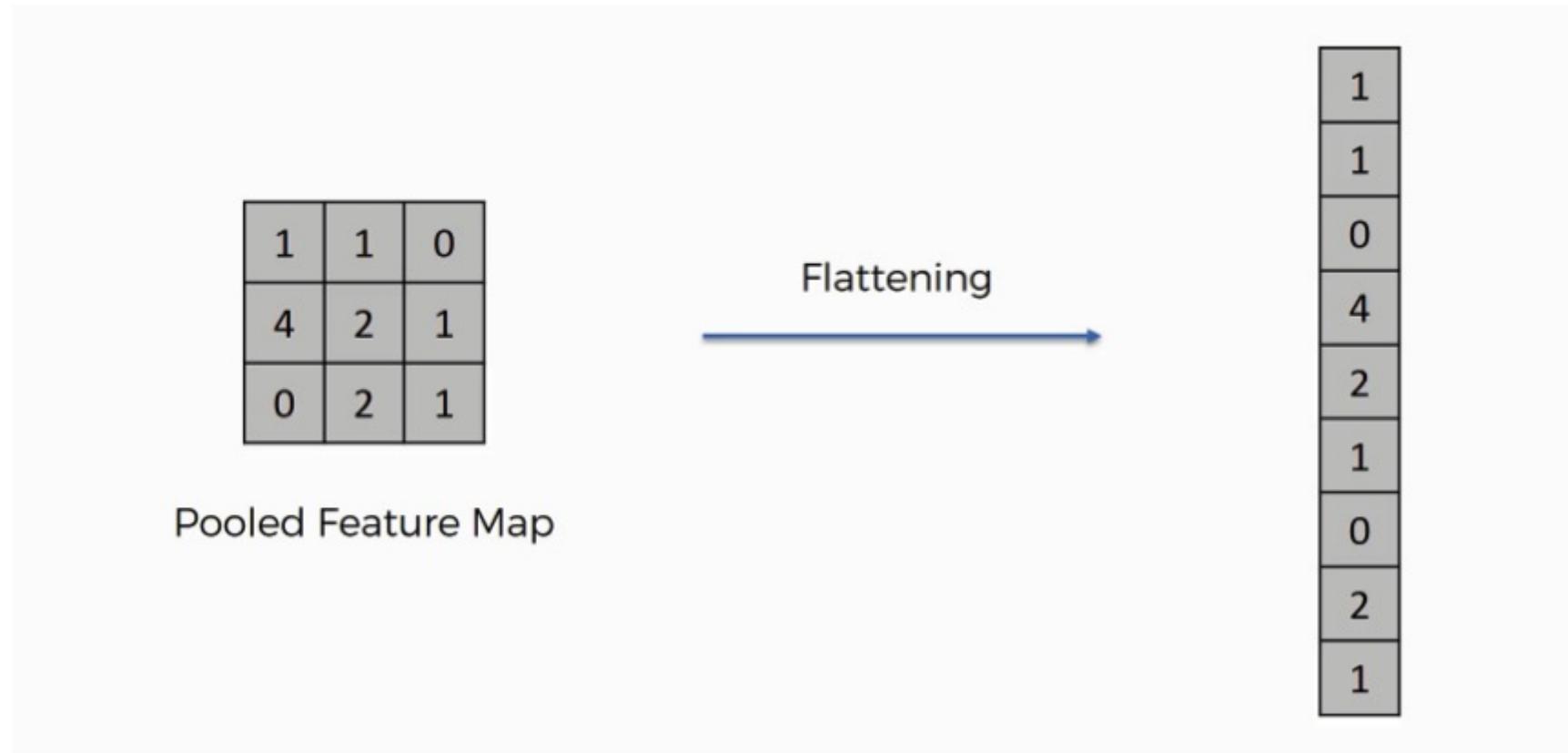
Non-linearity layers (ReLU)

- Since convolution is a linear operation, non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map.
- Main non-linear operators are:
 - Sigmoid
 - Tanh
 - ReLU
- ReLU is more reliable and accelerates the model convergence



Flattening

- After multiple convolution layers and pooling operations, the 3D representation of the image is converted into a feature vector that is passed into a multi-layer perception (MLP)

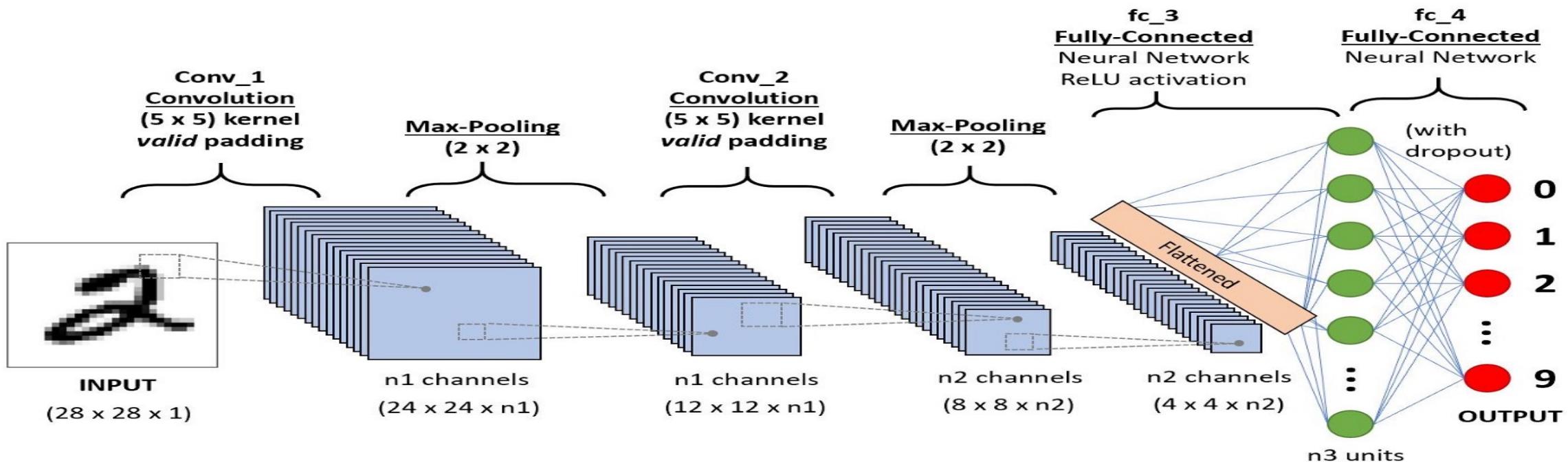


Fully-Connected Layer(s)

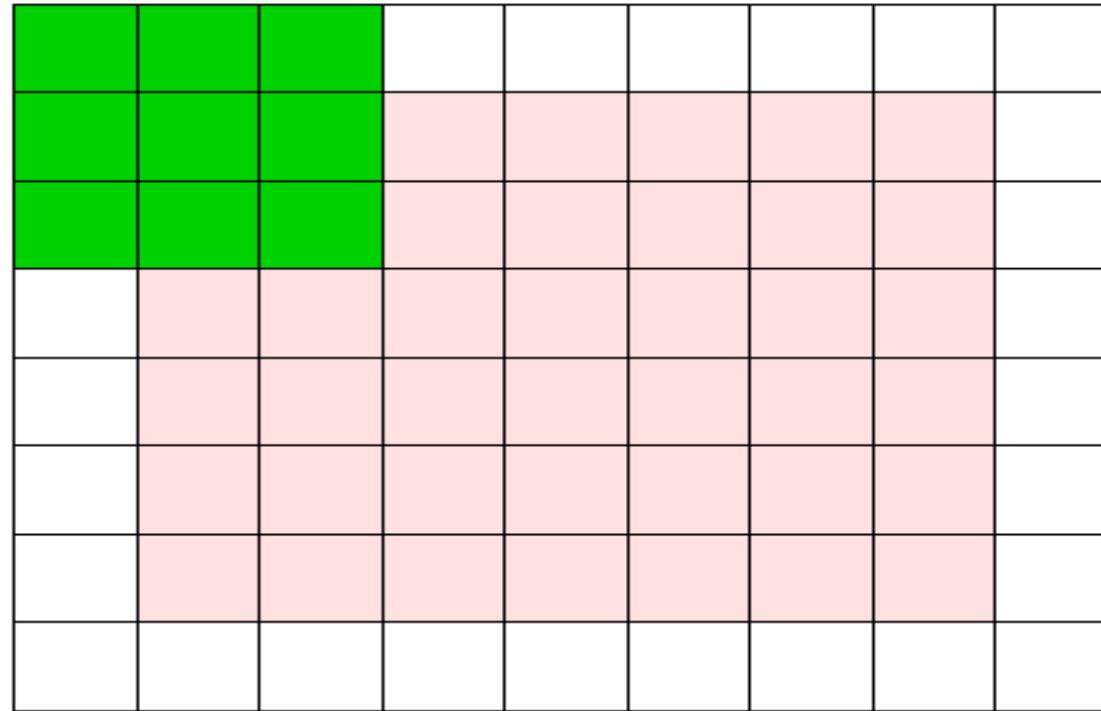
- The FC layer comprises the weights and biases together with the neurons and is used to connect the neurons between two separate layers.
- In FC layer, each node in the output layer connects directly to a node in the previous layer.
- The flattened feature map is passed through the FC layer(s)

Output Layer

- The output layer produces the probability of each class given the input image
- This is the last layer containing the same number of neurons as the number of classes in the dataset
- The output of this layer passes through the Softmax activation function to normalize the output to have probability sum to one

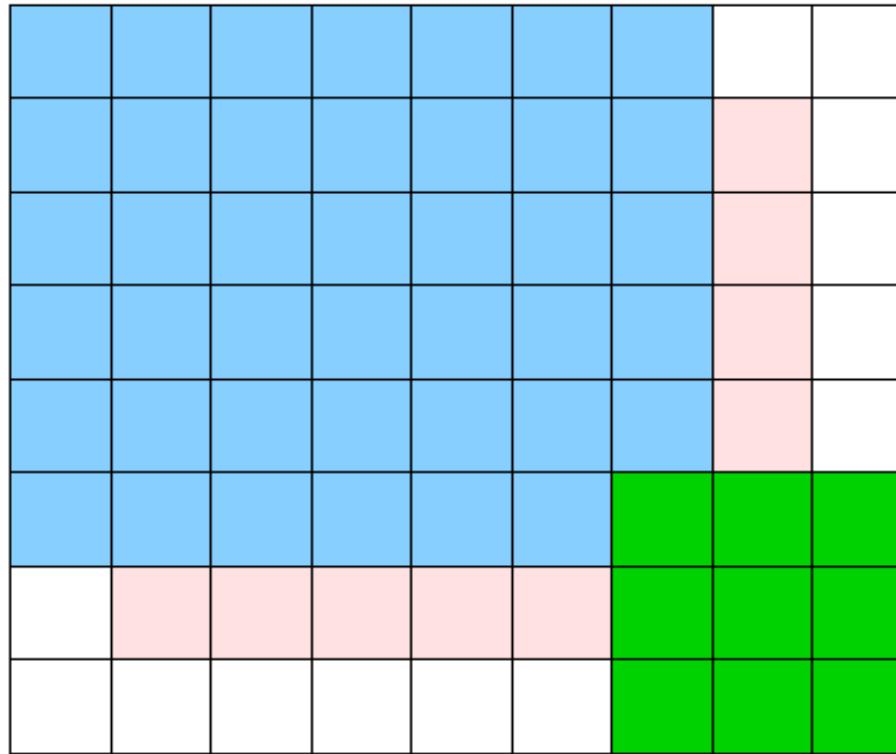


Convolution with Zero Padding



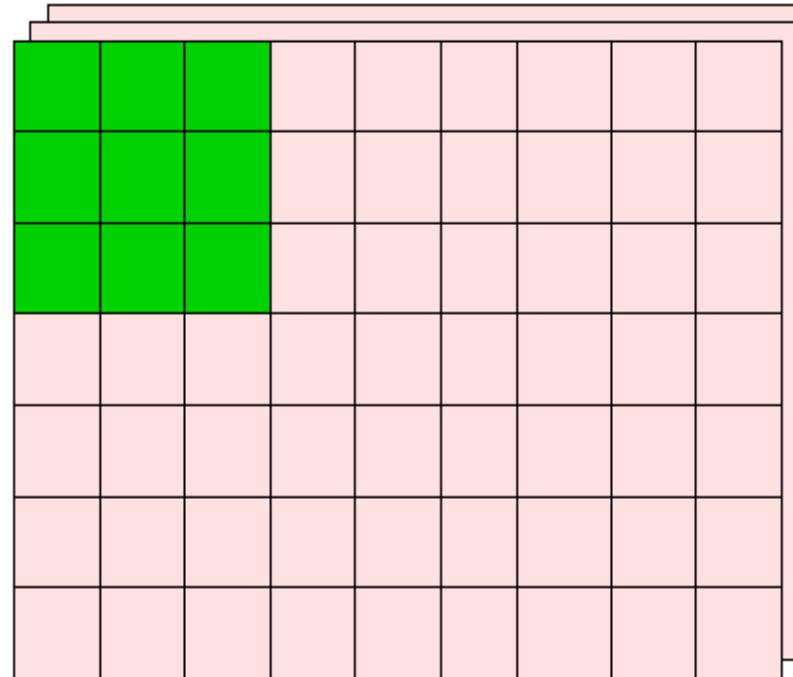
- Sometimes, we treat the off-edge inputs as zero (or some other value).
- This is known as “Zero-Padding”

Convolution with Zero Padding

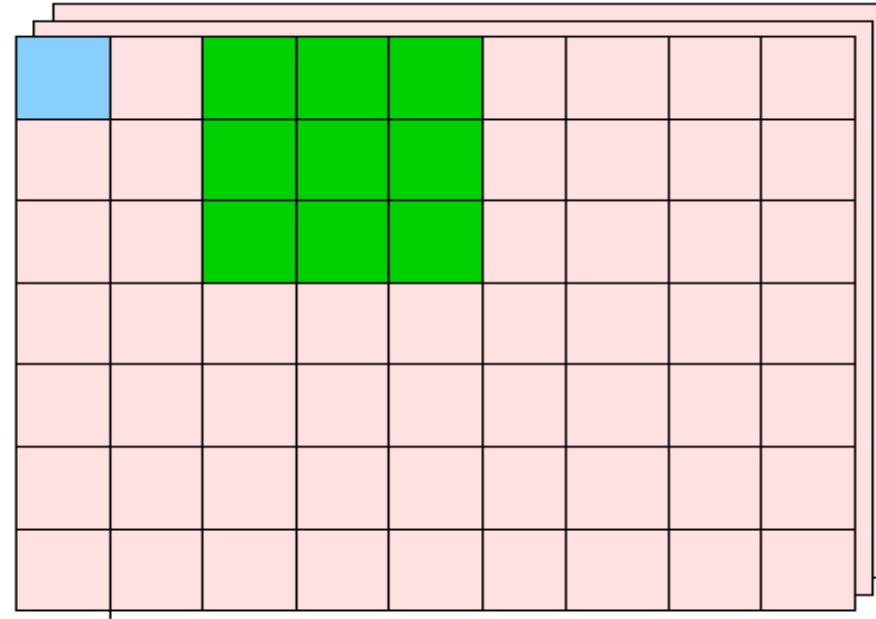


- With “Zero-Padding”, the convolution layer is the same size as the original image (or the previous layer).

Stride



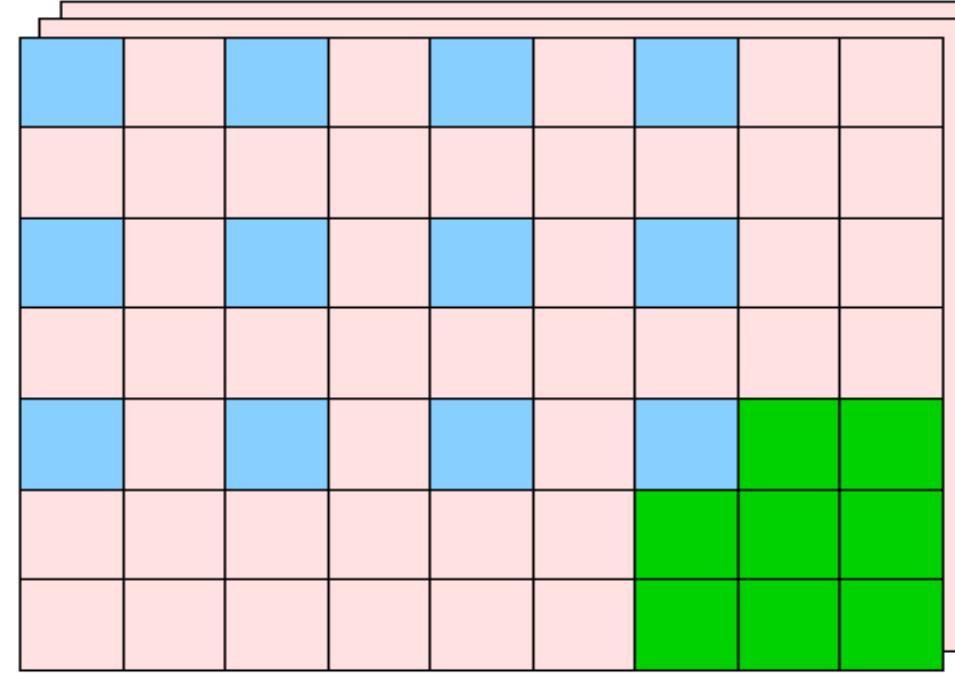
- Assume the original image is $J \times K$, with L channels.
- We again apply an $M \times N$ filter, but this time with a “stride” of $s > 1$
- In this example, $J = 7$, $K = 9$, $L = 3$, $M = 3$, $N = 3$, $s = 2$



$$Z_{j,k}^i = g\left(b^i + \sum_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K_{l,m,n}^i V_{j+m, k+n}^l\right)$$

- The same formula is used, but j and k are now incremented by s each time.
- The number of free parameters is $1 + L \times M \times N$

Stride Dimensions

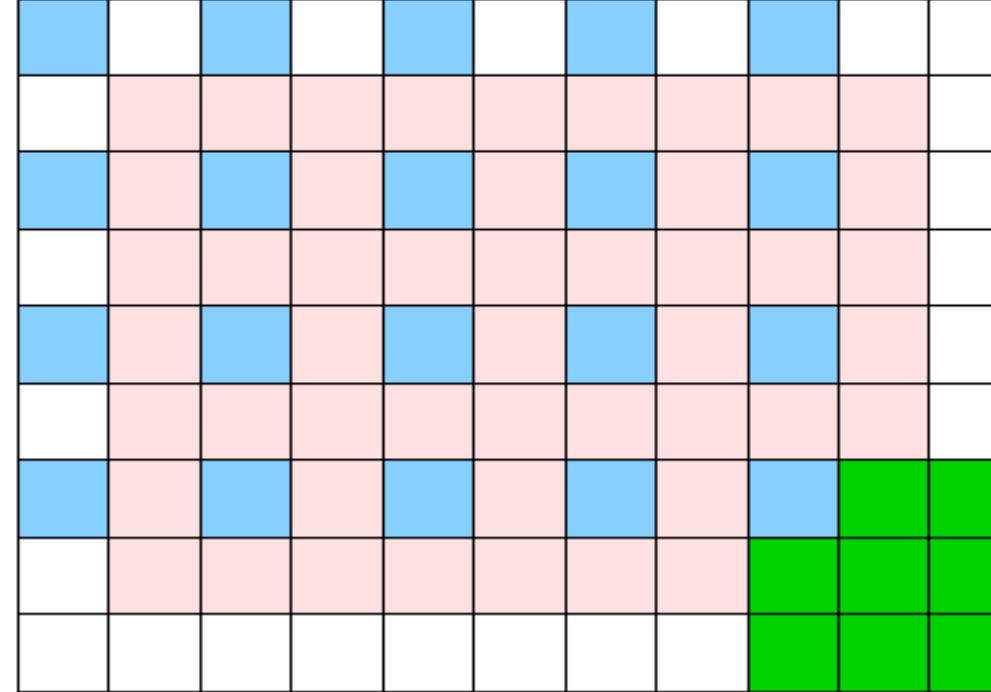


j takes on the values $0, s, 2s, \dots, (J - M)$

k takes on the values $0, s, 2s, \dots, (K - N)$

The next layer is $(1 + (J - M) / s) \times (1 + (K - N) / s)$

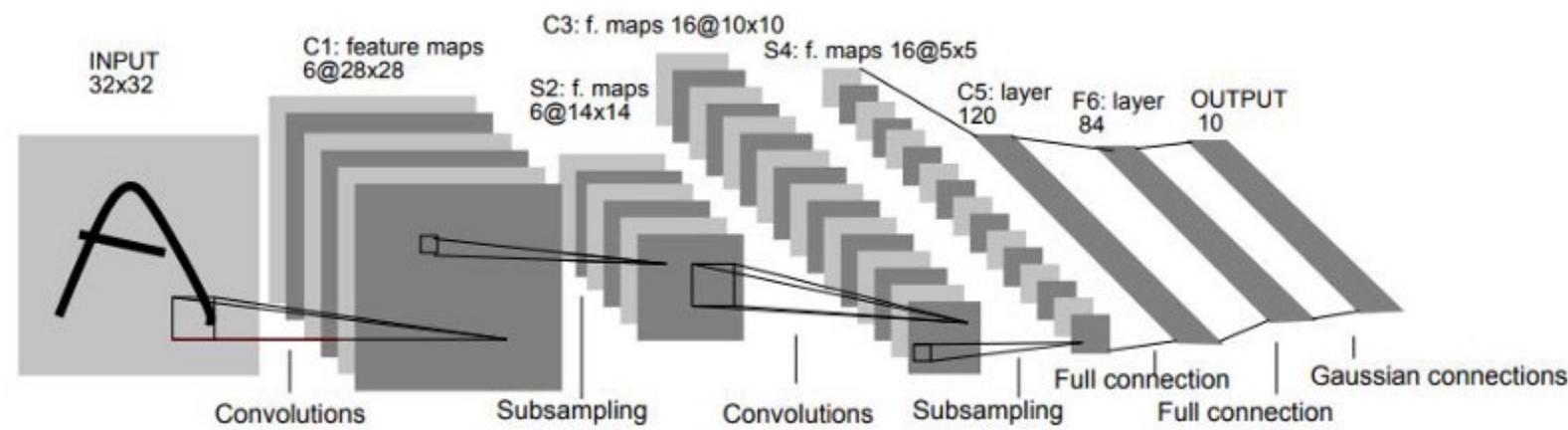
Stride with Zero Padding



- When combined with zero-padding of width P ,
 - j takes on the values $0, s, 2s, \dots, (J + 2P - M)$
 - k takes on the values $0, s, 2s, \dots, (K + 2P - N)$
 - The next layer is $(1 + (J + 2P - M) / s) \times (1 + (K + 2P - N) / s)$

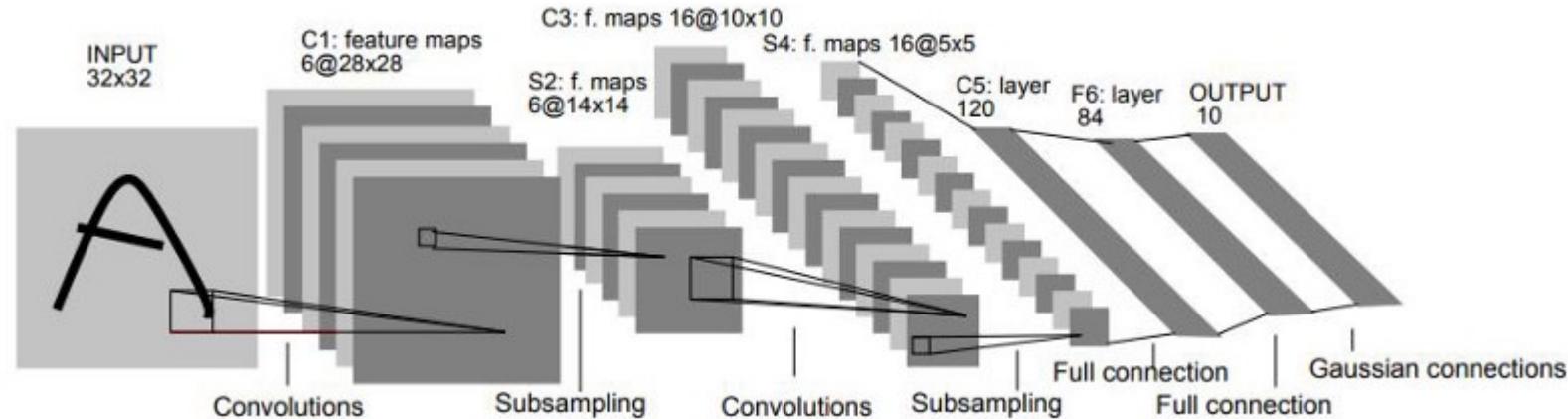
A walk-through of LeNet

- LeNet contains 2 convolutional layers, 2 pooling layers, 2 fully-connected layers, and an output layer



A walk-through of LeNet

- Convolution layer

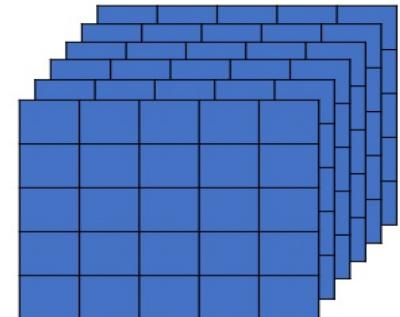


Convolutional Layer



$32 \times 32 \times 1$

Filters: 6
Filter Size: 5x5
Stride: 1
Padding: 0

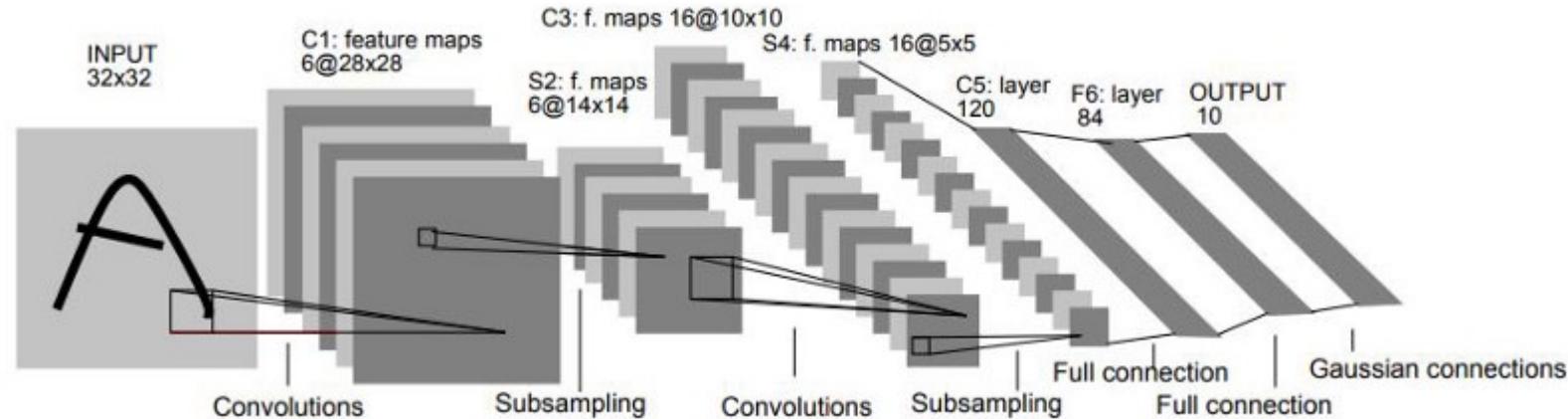


$28 \times 28 \times 6$



A walk-through of LeNet

- Pooling layer



Pooling Layer

Filters: 6

Filter Size: 2×2

Stride: 2

Padding: 0



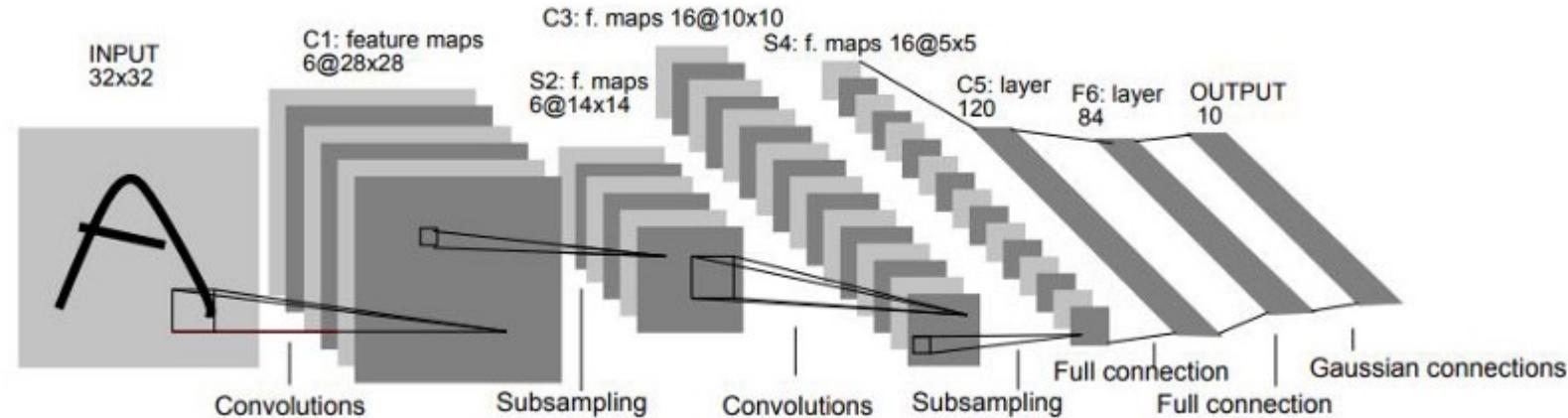
$28 \times 28 \times 6$



$14 \times 14 \times 6$

A walk-through of LeNet

- Convolution layer



Convolutional Layer

Filters: 16

Filter Size: 5x5

Stride: 1

Padding: 0



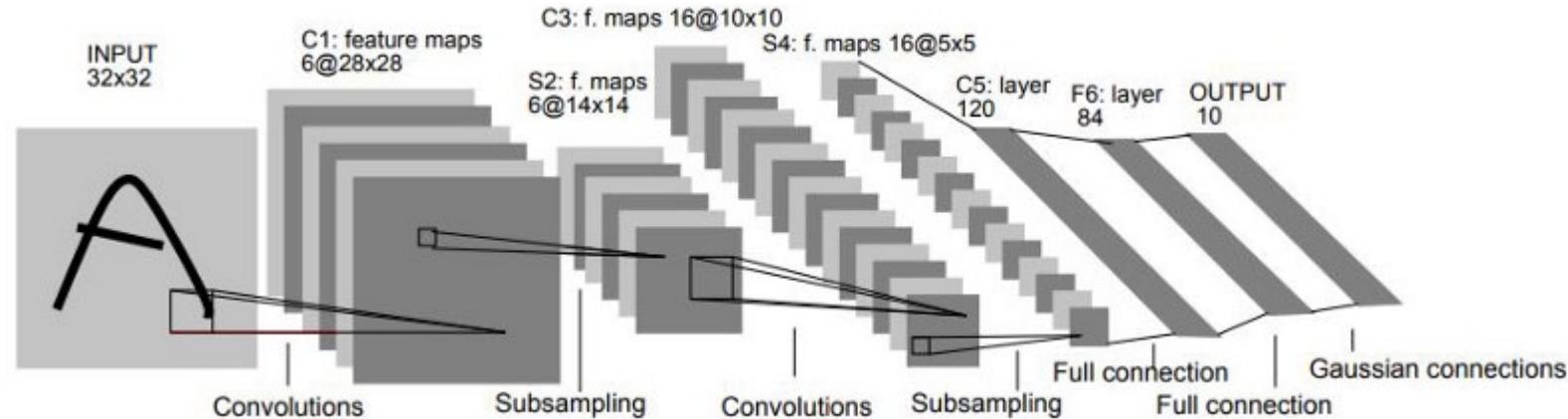
$14 \times 14 \times 6$



$10 \times 10 \times 16$

A walk-through of LeNet

- Pooling layer



Pooling Layer

Filters: 16
Filter Size: 2×2
Stride: 2
Padding: 0



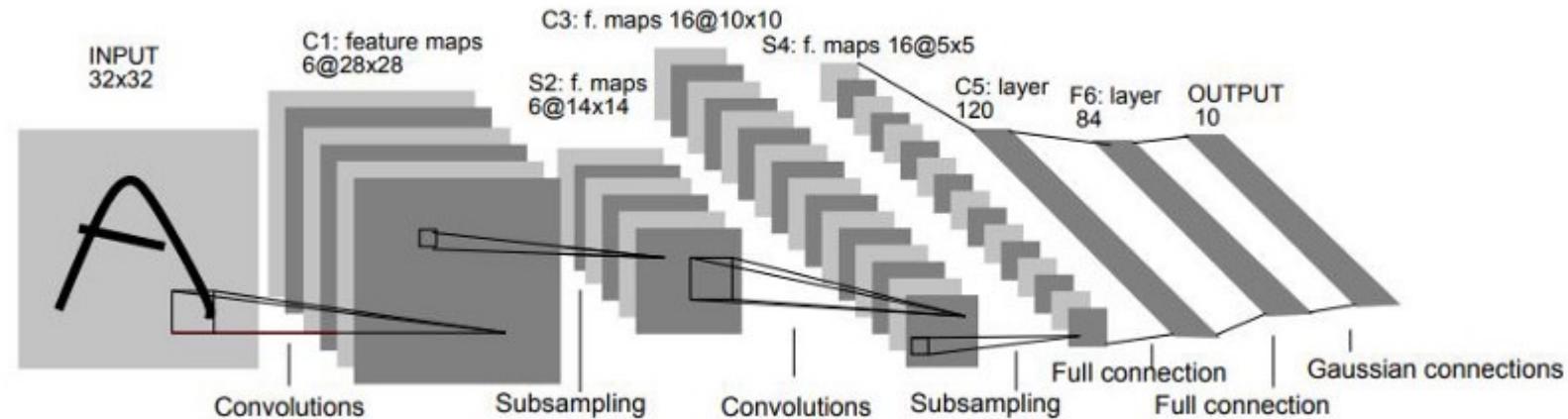
$10 \times 10 \times 16$



$5 \times 5 \times 16$

A walk-through of LeNet

- Flattening + FC layers
- $5 \times 5 \times 16 = 400$ neurons



Fully Connected Layer



$5 \times 5 \times 16$



Fully Connected Layer



120



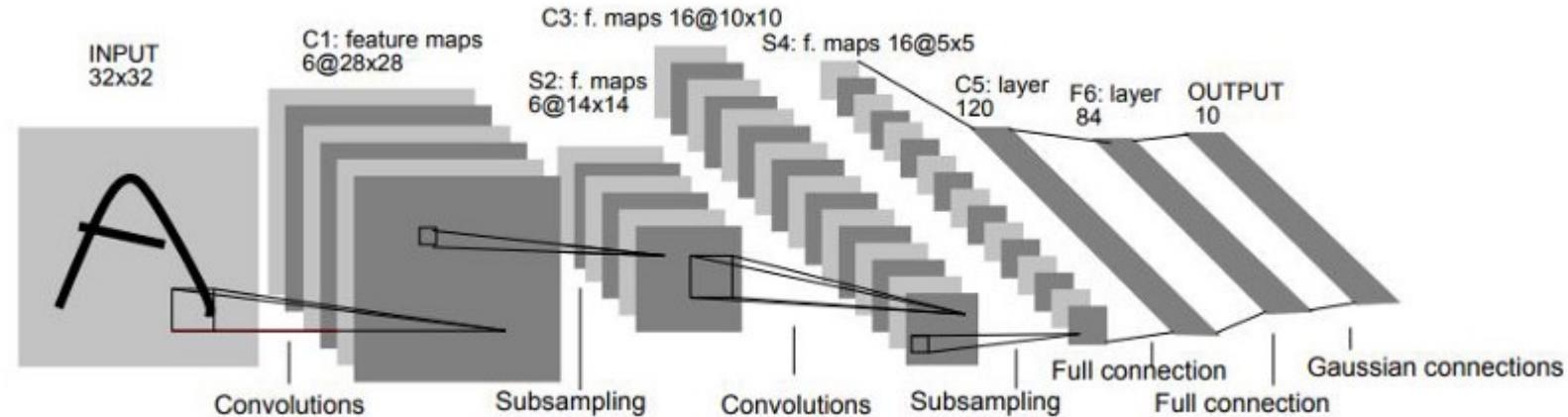
120



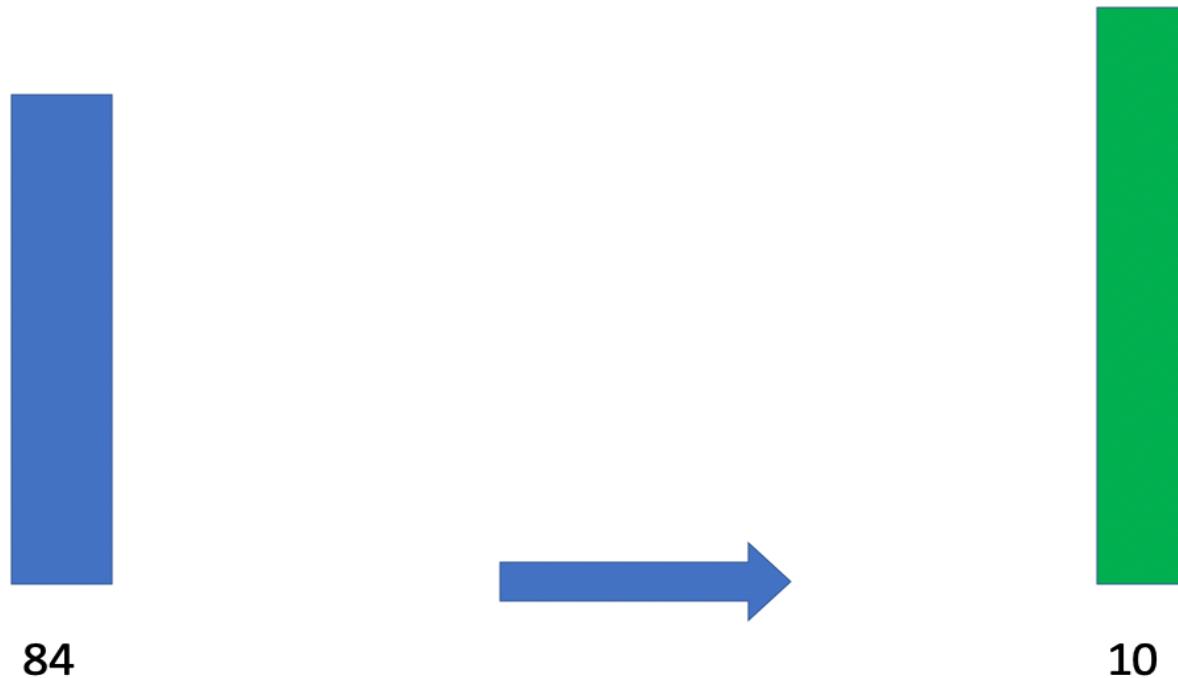
84

A walk-through of LeNet

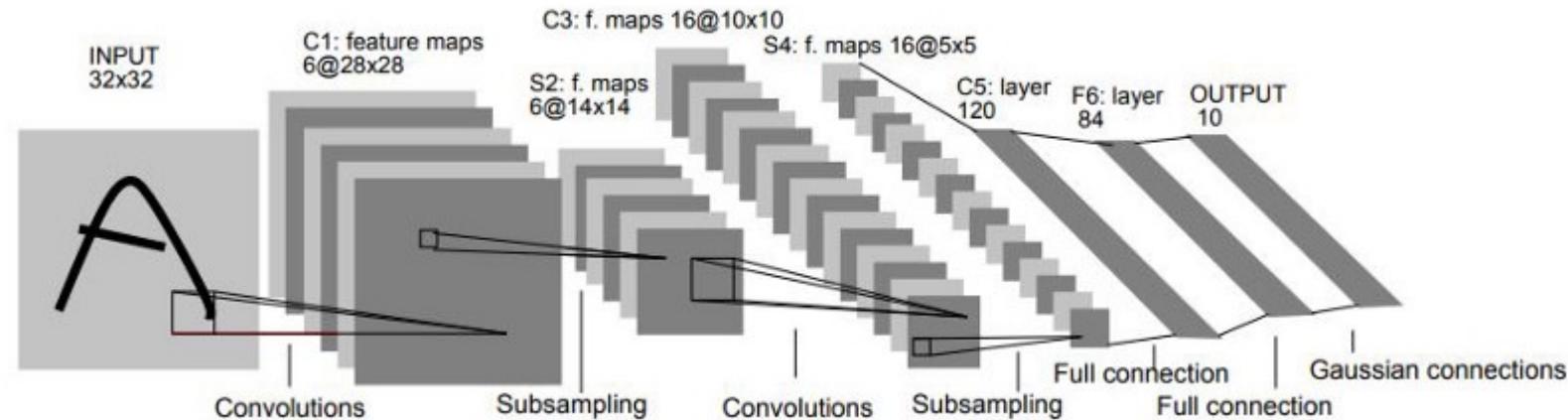
- Output layer
- MNIST dataset (10 digits)



Output Layer



Example: LeNet



For example, in the first convolutional layer of LeNet (greyscale images),
 $J = K = 32, M = N = 5$.

The width of the next layer is

$$J + 1 - M = 32 + 1 - 5 = 28$$

Question: If there are 6 filters in this layer, compute the number of:

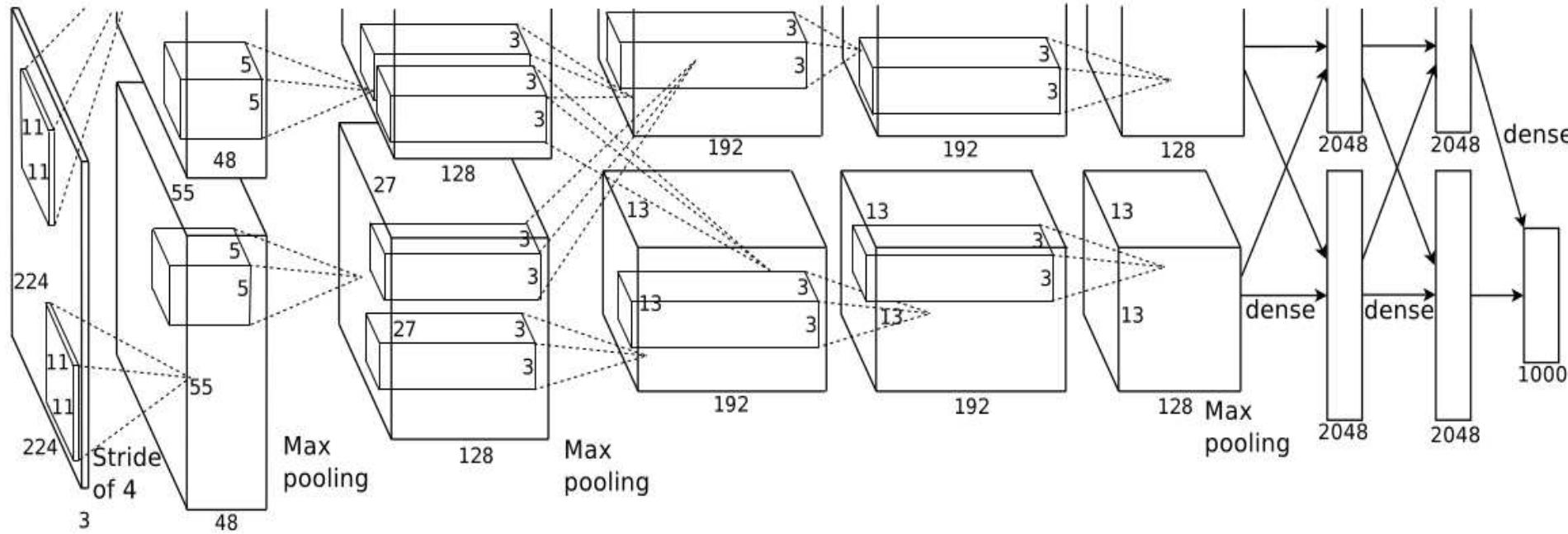
weights per filter? $1 + 5 \times 5 \times 1 = 26$

neurons? $28 \times 28 \times 6 = 4704$

connections? $28 \times 28 \times 6 \times 26 = 122304$

independent parameters? $6 \times 26 = 156$

AlexNet (2012)



- 5 convolutional layers + 3 fully-connected layers
- max-pooling with overlapping stride
- softmax with 1000 classes
- 2 parallel GPUs which interact only at certain layers

Example: AlexNet Conv Layer 1

- In the first convolutional layer of AlexNet,

$$J = K = 224, P = 2, M = N = 11, s = 4$$

The width of the next layer is

$$1 + (J + 2P - M) / s = 1 + (224 + 2 \times 2 - 11) / 4 = 55$$

Question: If there are 96 filters in this layer, compute the number of:

weights per neuron?

neurons?

connections?

independent parameters?

Overlapping Pooling

- In the previous layer is $J \times K$, and max pooling is applied with width F and stride s , the size of the next layer will be

$$(1 + (J - F) / s) \times (1 + (K - F) / s)$$

Question: If max pooling with width 3 and stride 2 is applied to the features of size 55×55 in the first convolutional layer of AlexNet, what is the size of the next layer?

Answer:

Question: How many independent parameters does this add to the model?

Answer:

Overlapping Pooling

- In the previous layer is $J \times K$, and max pooling is applied with width F and stride s , the size of the next layer will be

$$(1 + (J - F) / s) \times (1 + (K - F) / s)$$

Question: If max pooling with width 3 and stride 2 is applied to the features of size 55×55 in the first convolutional layer of AlexNet, what is the size of the next layer?

Answer: $1 + (55 - 3) / 2 = 27$

Question: How many independent parameters does this add to the model?

Answer: None! (no weights to be learned, just computing max)

How to train CNNs

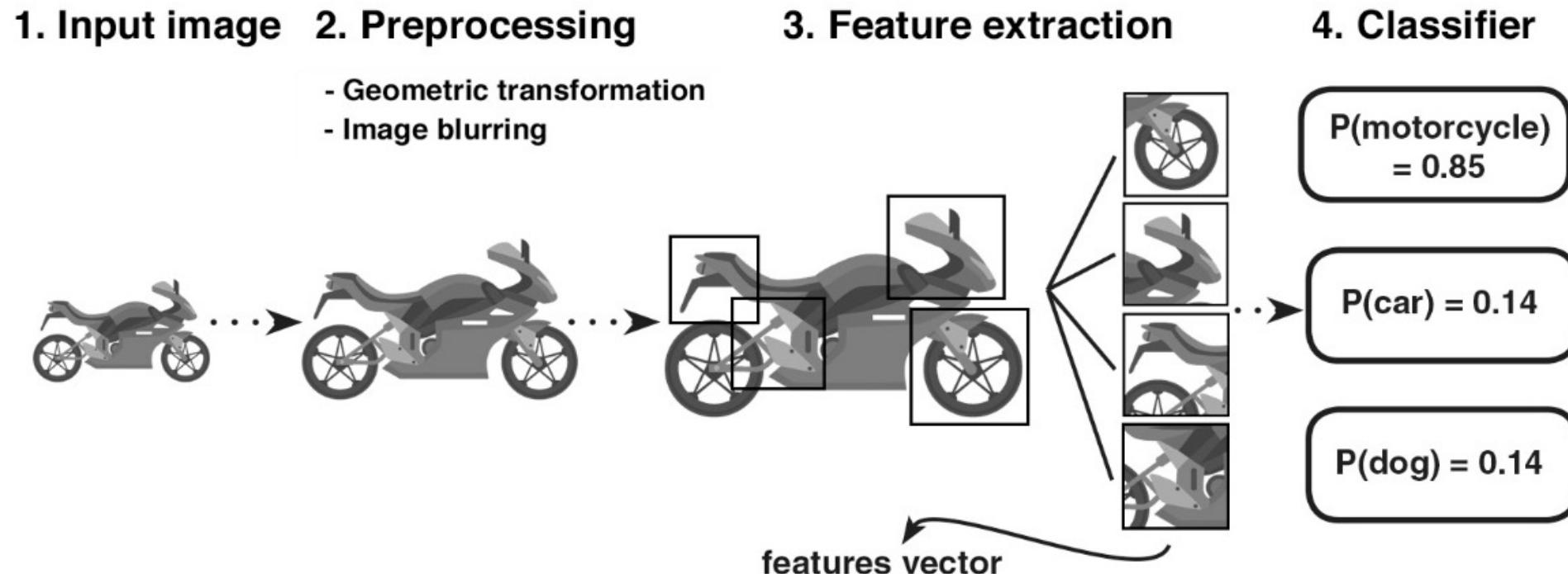
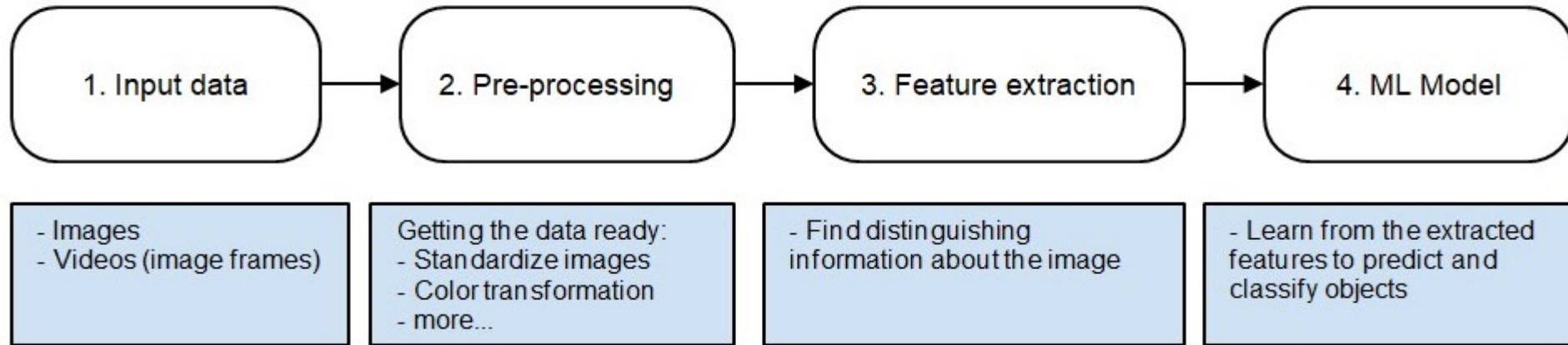
- A loss function is used to compute the model's prediction accuracy from the outputs
- Commonly used: Categorical cross-entropy loss function

$$L_{\text{CE}} = - \sum_{i=1}^n t_i \log(p_i), \text{ for } n \text{ classes,}$$

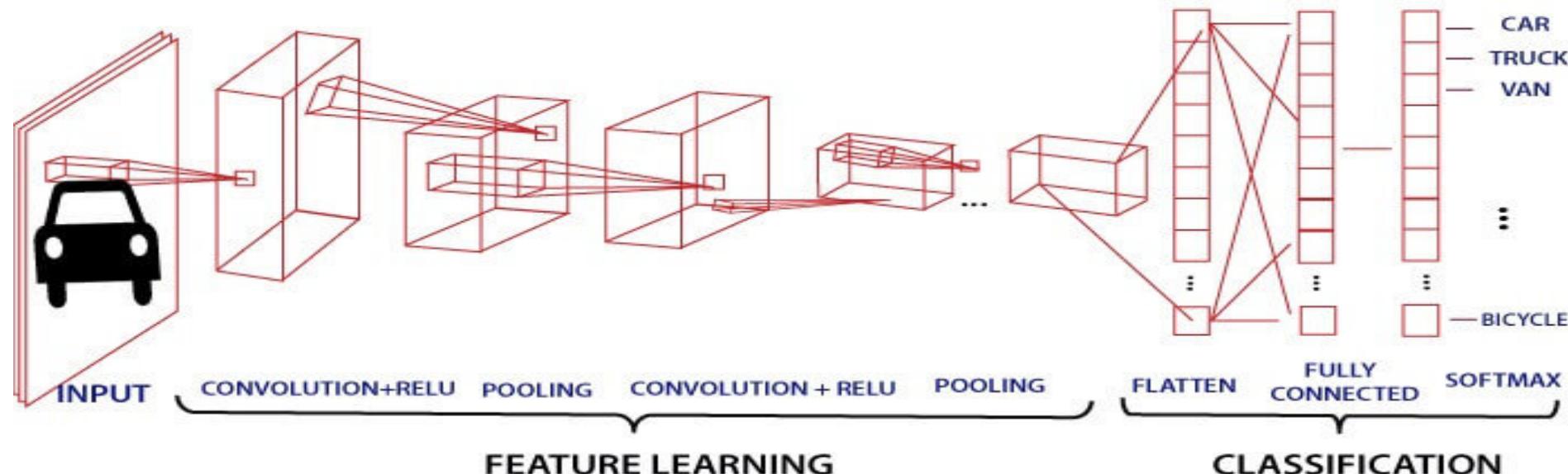
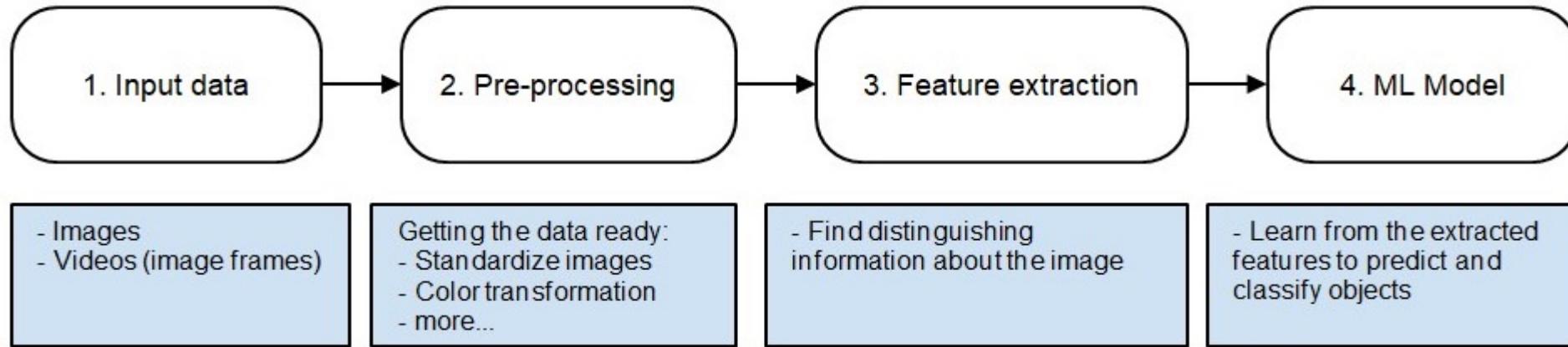
where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

- The training objective is to minimize this loss
- The loss guides the backpropagation process to train the CNN model
- Gradient descent methods, such as Stochastic Gradient Descent (SGD) and the Adam optimizer, are commonly used algorithms for optimization

Traditional CV pipeline vs CNNs

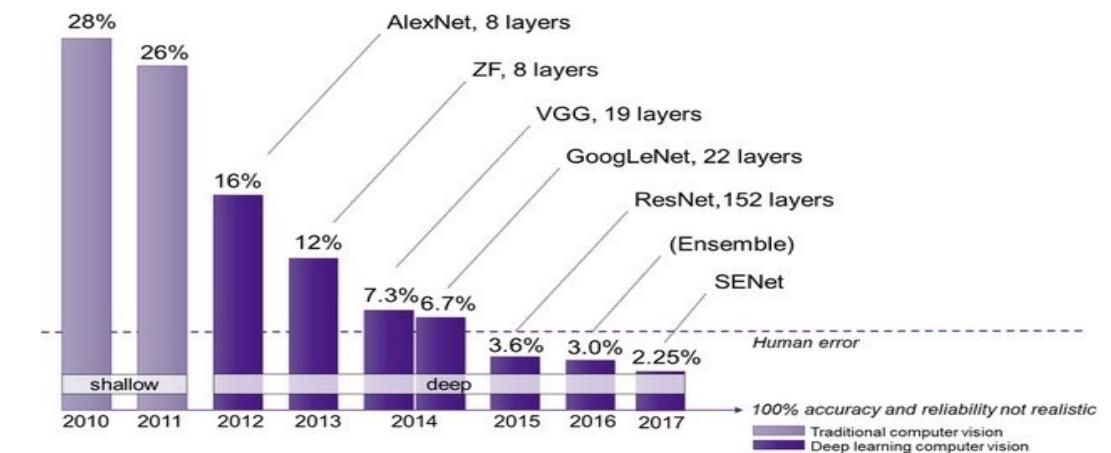
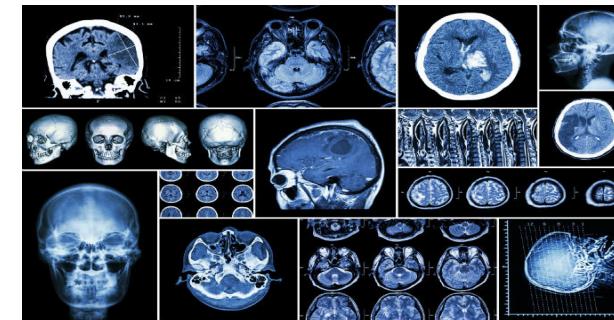


Traditional CV pipeline vs CNNs



Why DL becoming so popular?

- Increasing availability of large-scale annotated datasets
- Improvement in computing power
 - Rise of Graphics Processing Units (GPUs)
- Advancements in algorithms
 - Better and bigger models
 - new insights and improved techniques to train models
- Open-source codebase + DL Frameworks



Deep Learning = convergence of data, models, and computing power

Important readings

- Understanding and calculating model parameters (weights)?

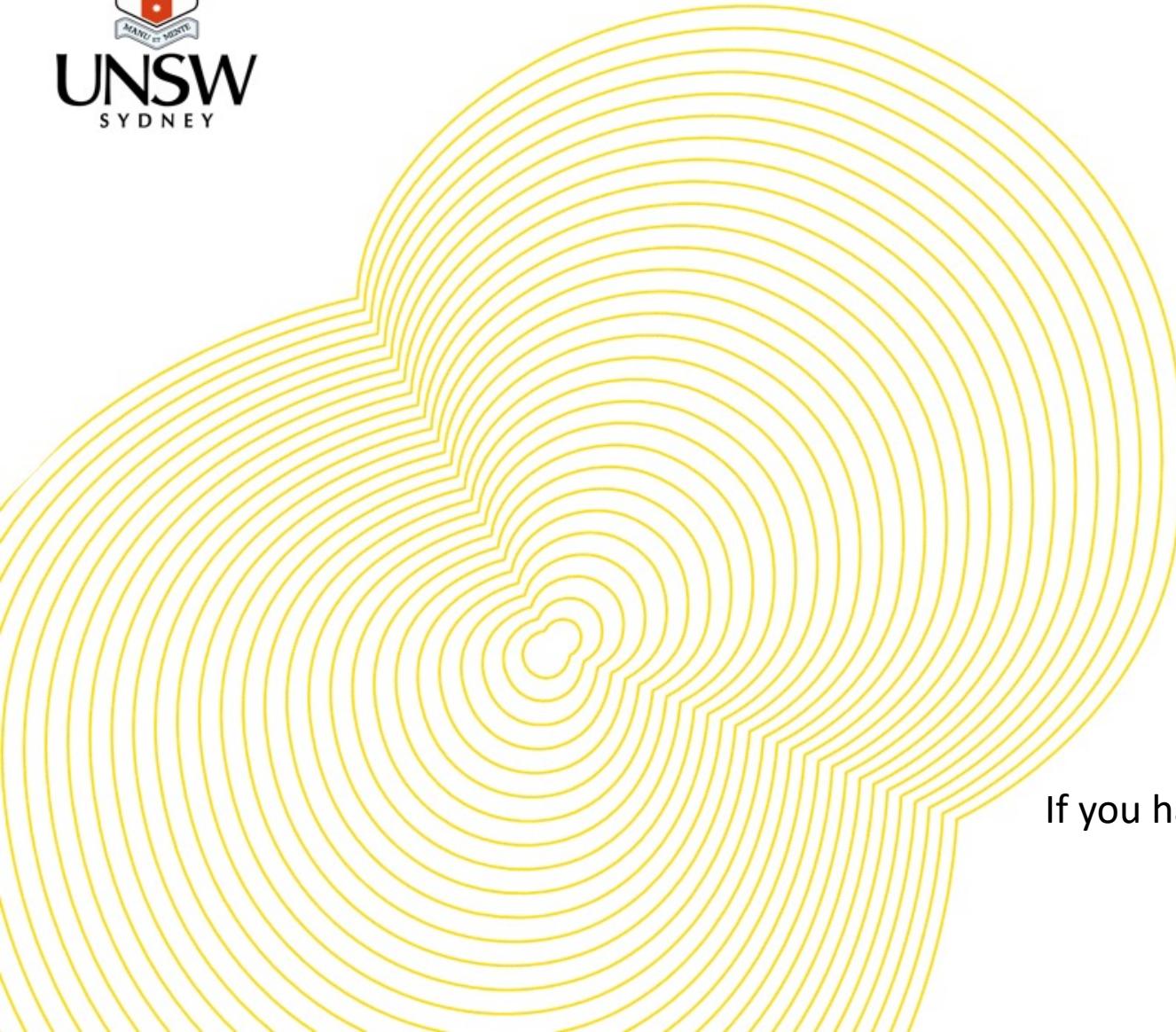
<https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d>

<https://medium.com/@iamvarman/how-to-calculate-the-number-of-parameters-in-the-cnn-5bd55364d7ca>

<https://androidkt.com/calculating-number-parameters-pytorch-model/>



UNSW
SYDNEY



Questions?

If you have any questions, post it on the Ed forum