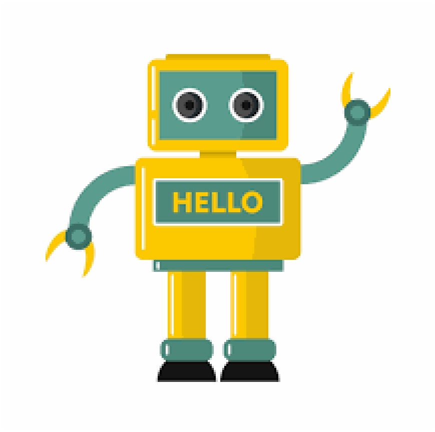
THURSDAY LEARNING HOUR

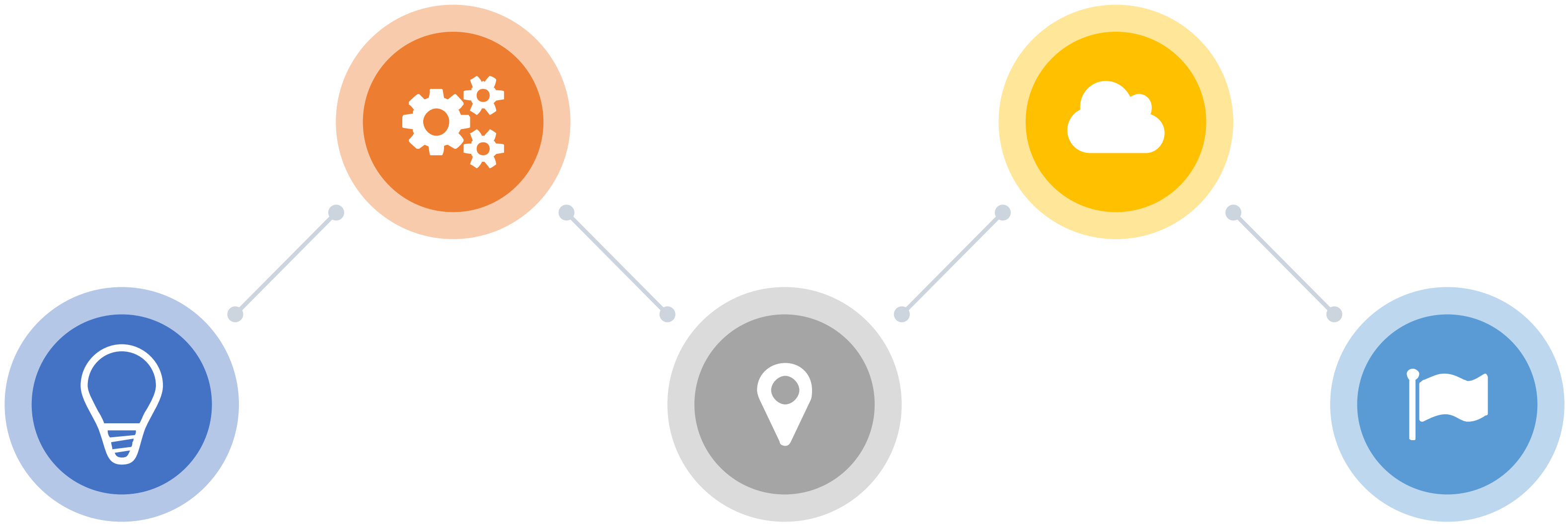
Computer Vision Using Deep Neural Networks



Prabakaran Chandran

Agenda

Computer Vision using Deep Neural Networks



Introduction

Deep learning Pipeline

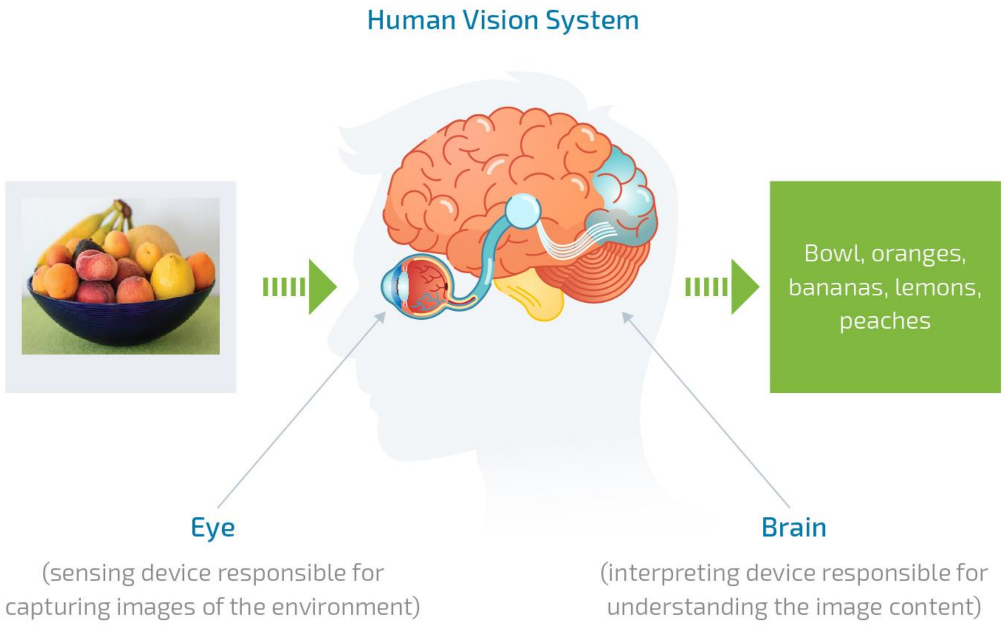
Convolutional Neural Networks

Image Classification

Transfer Learning

What is Computer Vision? :

- Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do



Human Vision System

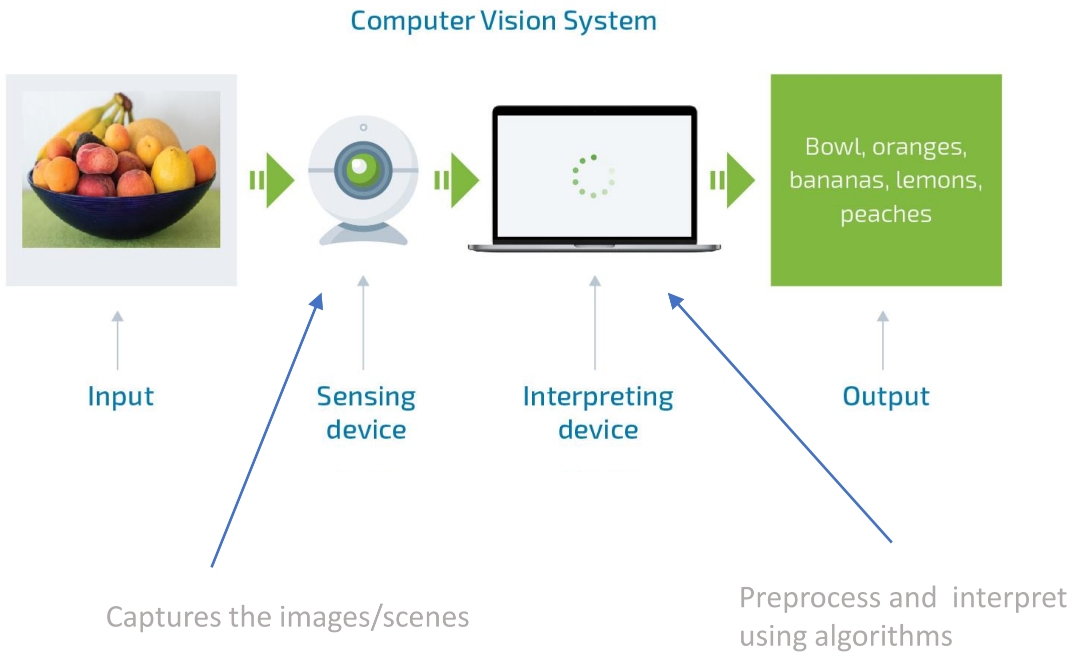
Bowl, oranges, bananas, lemons, peaches

Eye

(sensing device responsible for capturing images of the environment)

Brain

(interpreting device responsible for understanding the image content)



Computer Vision System

Bowl, oranges, bananas, lemons, peaches

Input

Sensing device

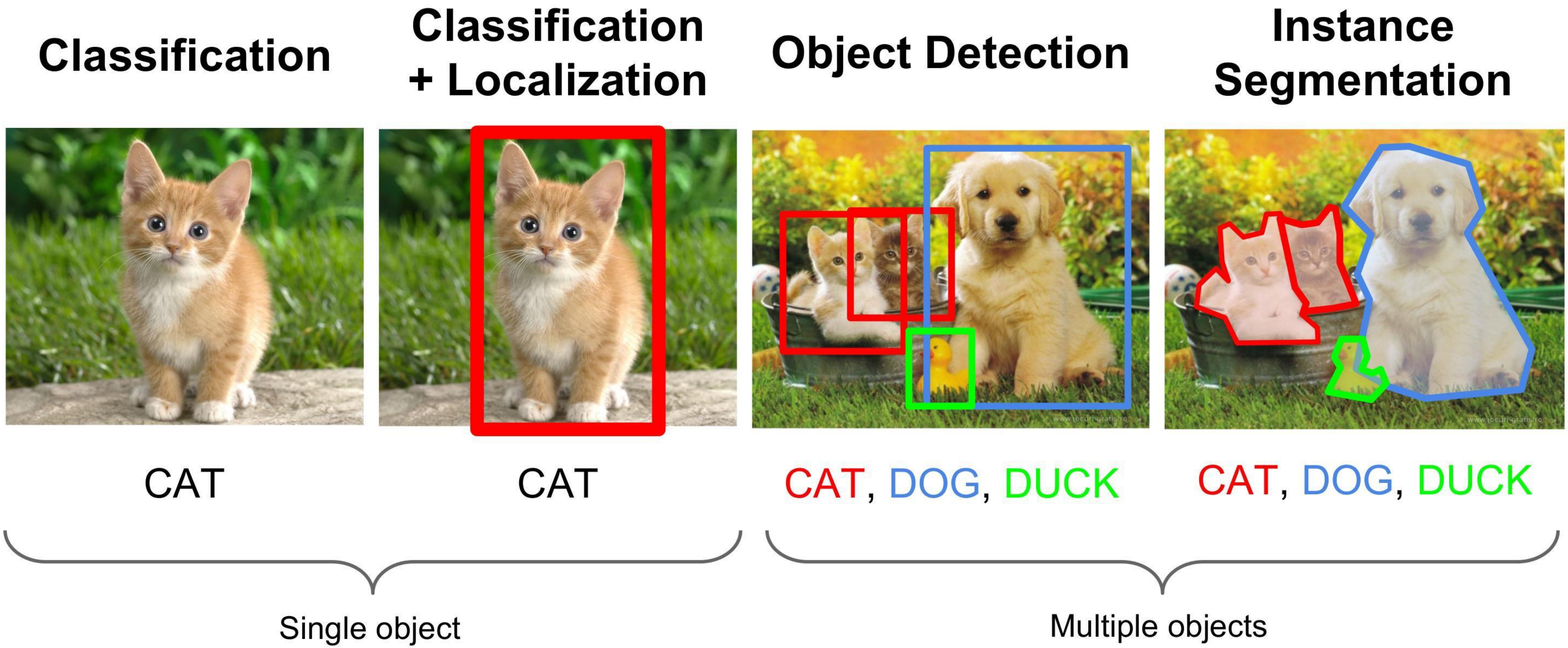
Interpreting device

Output

Captures the images/scenes

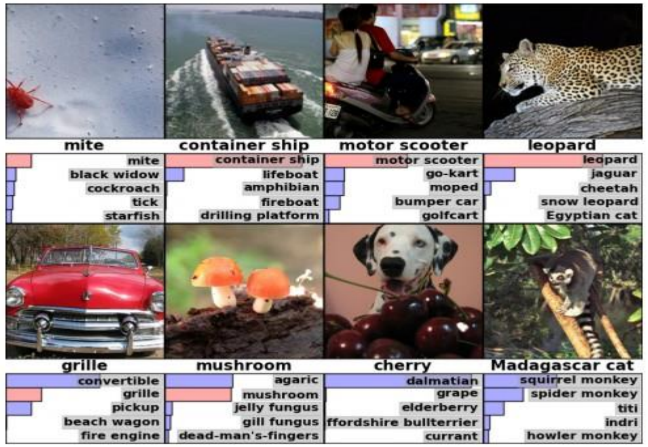
Preprocess and interpret using algorithms

Major Computer Vision tasks:

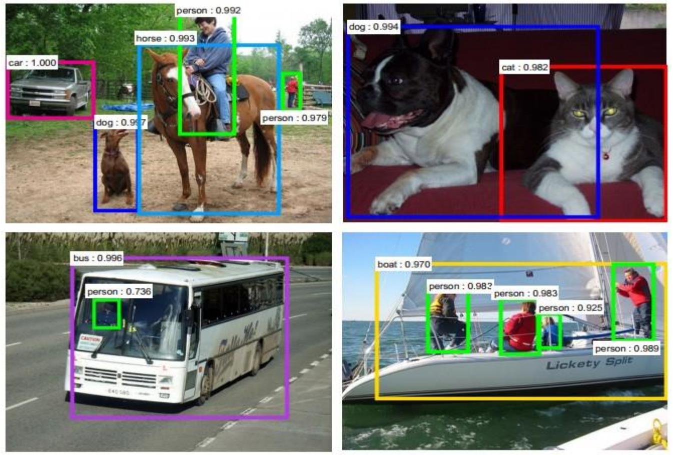


Computer Vision Tasks:

Classification



Detection



Segmentation

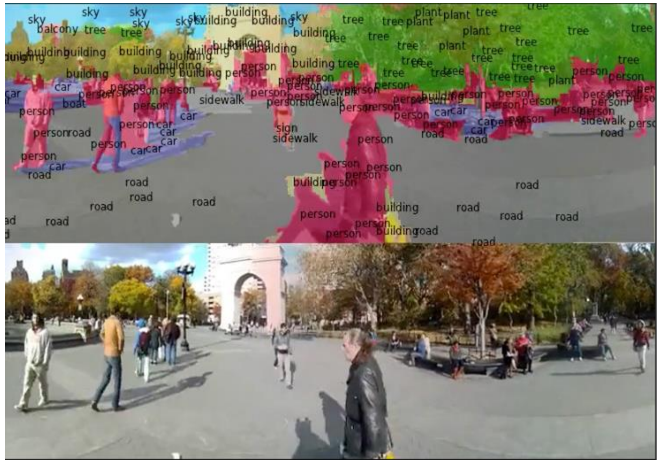
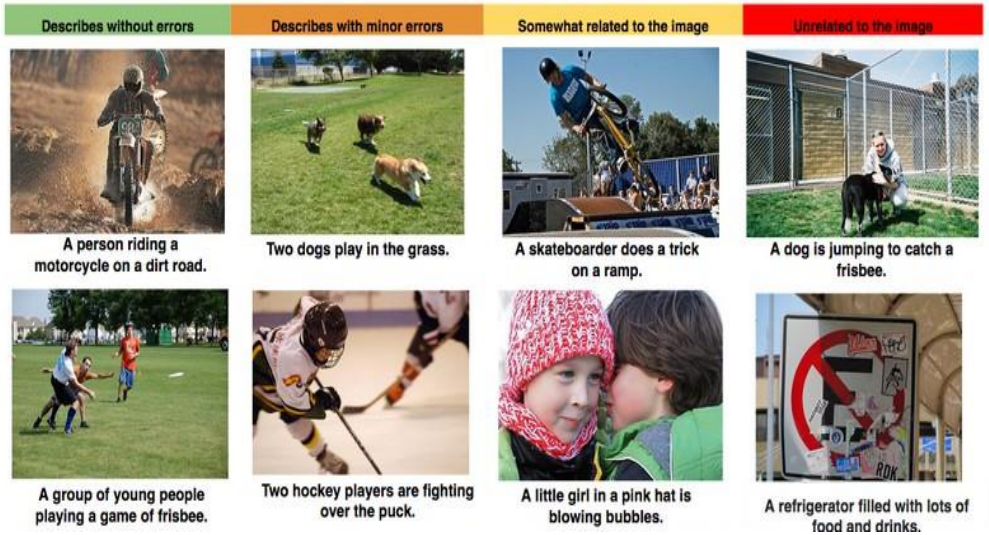


Image Captioning



A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

A skateboarder does a trick on a ramp.

A dog is jumping to catch a frisbee.

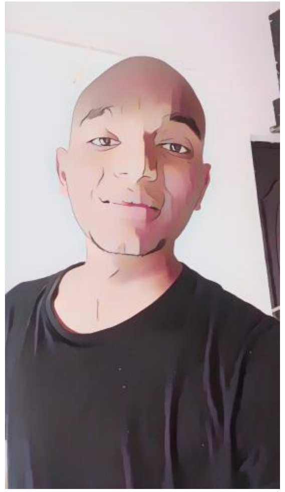
A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

A little girl in a pink hat is blowing bubbles.

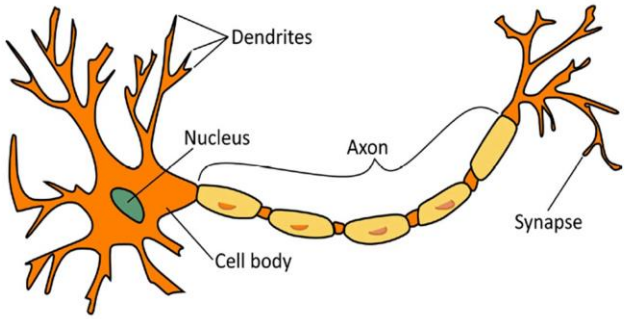
A refrigerator filled with lots of food and drinks.

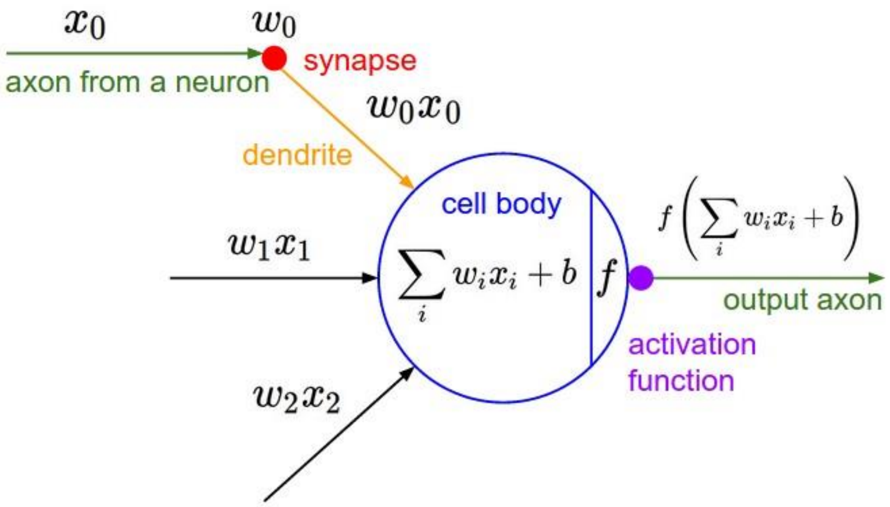
Image Cartooning

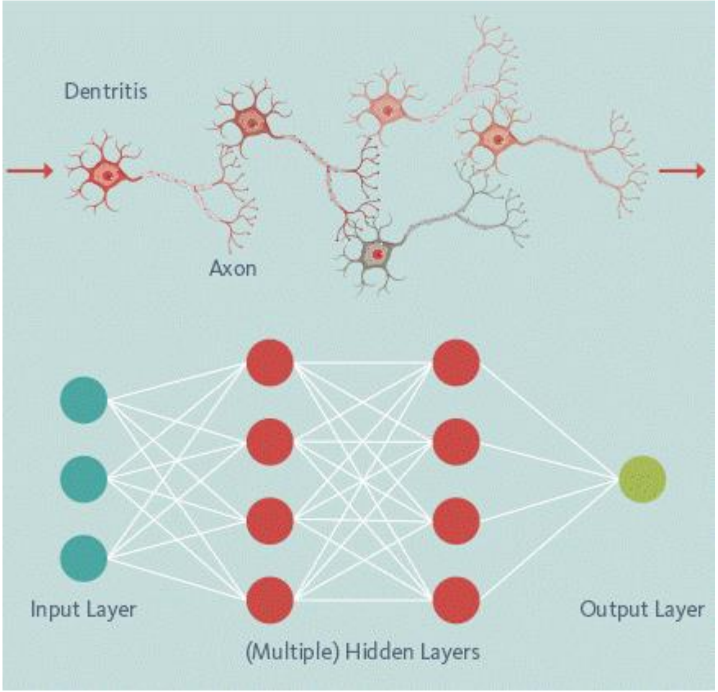


How can a Machine do Brain’s work ? – Artificial Neural Network

- Artificial Neural Networks or ANN is an information processing paradigm that is inspired by the way the biological nervous system such as brain process information
- It is composed of large number of highly interconnected processing elements(neurons) working in unison to solve a specific problem



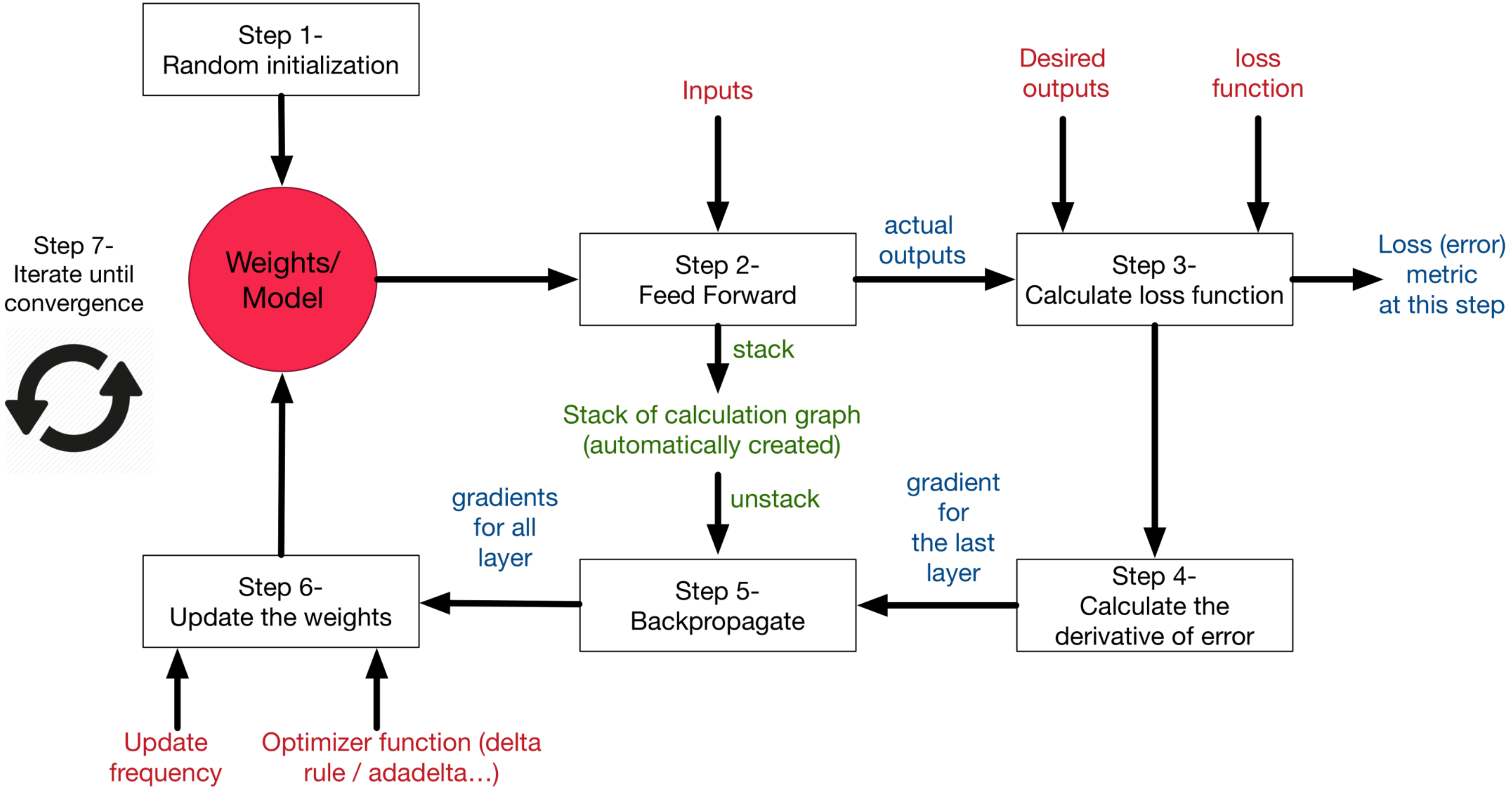




Neuron

Neural Network

How does a DNN learn things? – Fundamental Work flow



Step 1- Random initialization

Weights/ Model

Step 7- Iterate until convergence

Inputs

Step 2- Feed Forward

actual outputs

Desired outputs

loss function

Step 3- Calculate loss function

Loss (error) metric at this step

stack

Stack of calculation graph (automatically created)

unstack

Step 4- Calculate the derivative of error

gradient for the last layer

Step 5- Backpropagate

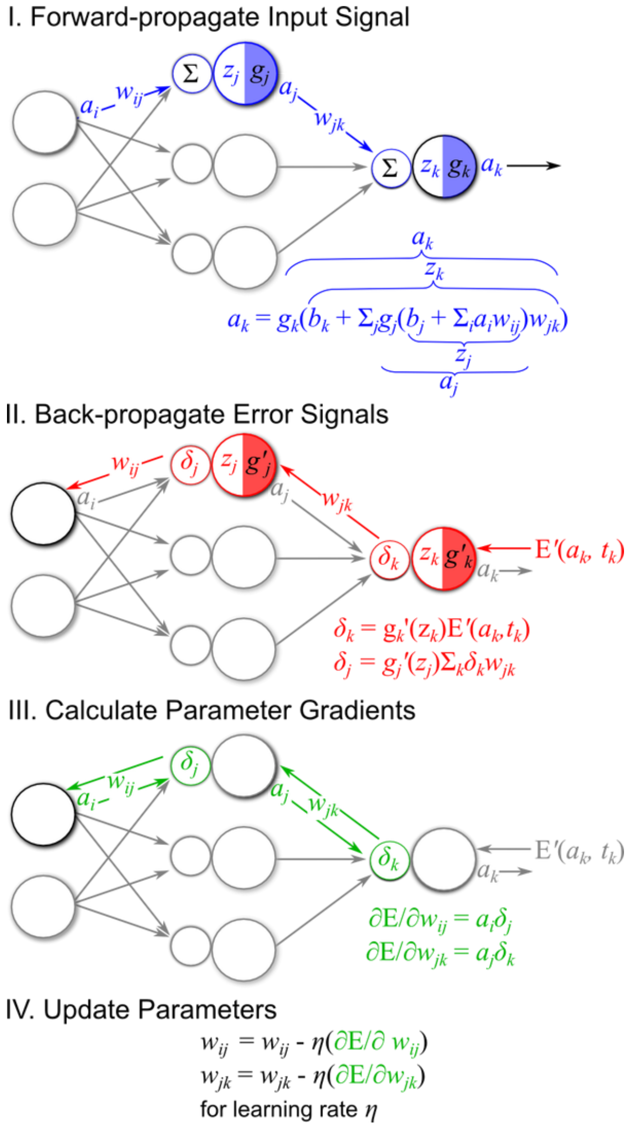
gradients for all layer

Step 6- Update the weights

Update frequency

Optimizer function (delta rule / adadelta…)

Fundamental Work flow and Key terms



I. Forward-propagate Input Signal

$$a_k = g_k(b_k + \Sigma_j g_j(b_j + \Sigma_i a_i w_{ij}) w_{jk})$$

II. Back-propagate Error Signals

$$\delta_k = g_k'(z_k)E'(a_k, t_k)$$
$$\delta_j = g_j'(z_j)\Sigma_k \delta_k w_{jk}$$

III. Calculate Parameter Gradients

$$\partial E/\partial w_{ij} = a_i \delta_j$$
$$\partial E/\partial w_{jk} = a_j \delta_k$$

IV. Update Parameters

$$w_{ij} = w_{ij} - \eta(\partial E/\partial\, w_{ij})$$
$$w_{jk} = w_{jk} - \eta(\partial E/\partial w_{jk})$$

for learning rate η

a_i *− inputs* , w_i *− weights are the actual learning paramerters*

Σ *− function that sums up input * weights*

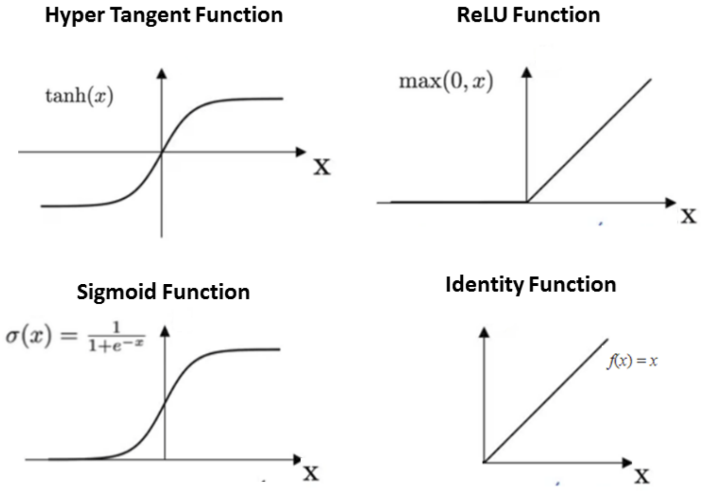
g_i, g_k *− activation function (Relu , softmax ,* tanh, *and sigmoid)*

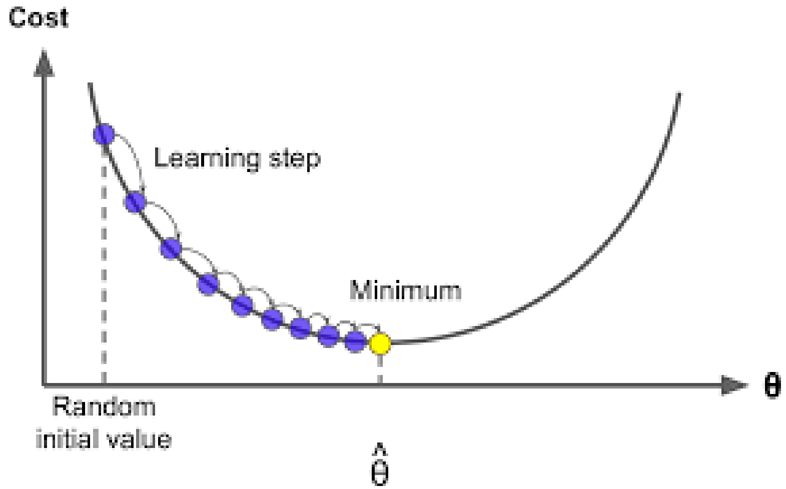
b_i *− bias term*

Forward pass calculates the output - a_k t_k *− target E or J − loss*

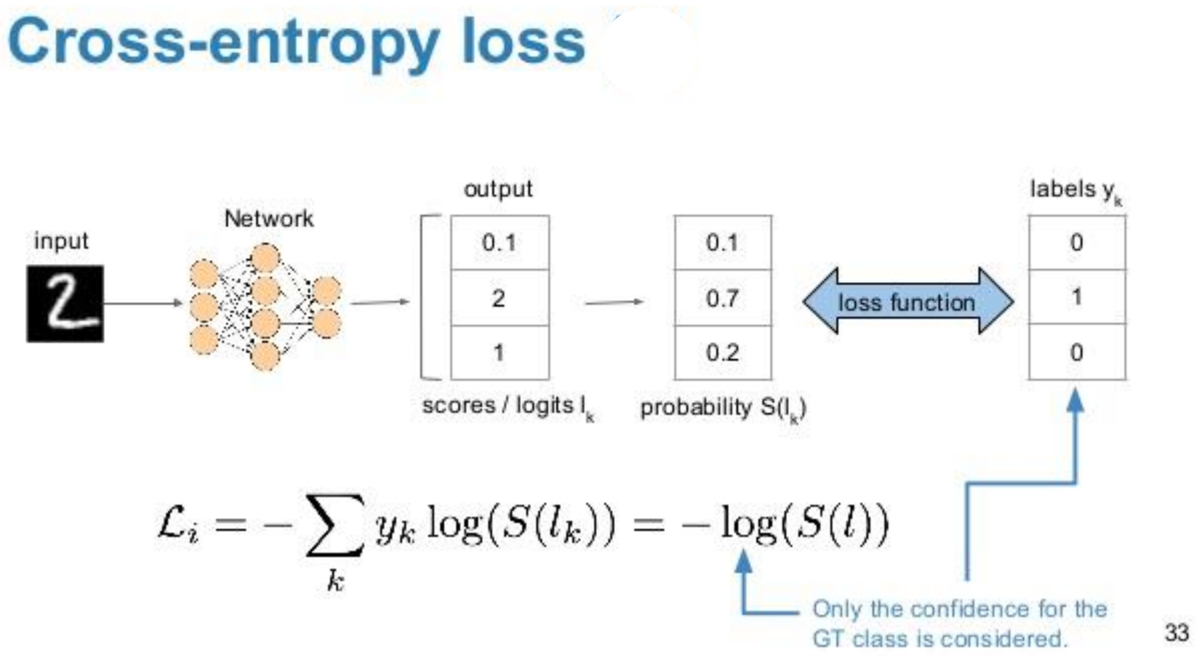
The aim of the back propagation is to calculate gradients to update the learning parameters which reduce the loss J

Loss : Negative log likelihood , Categorical Cross entropy , MSE





Fundamental Work flow – Loss function and Optimizers

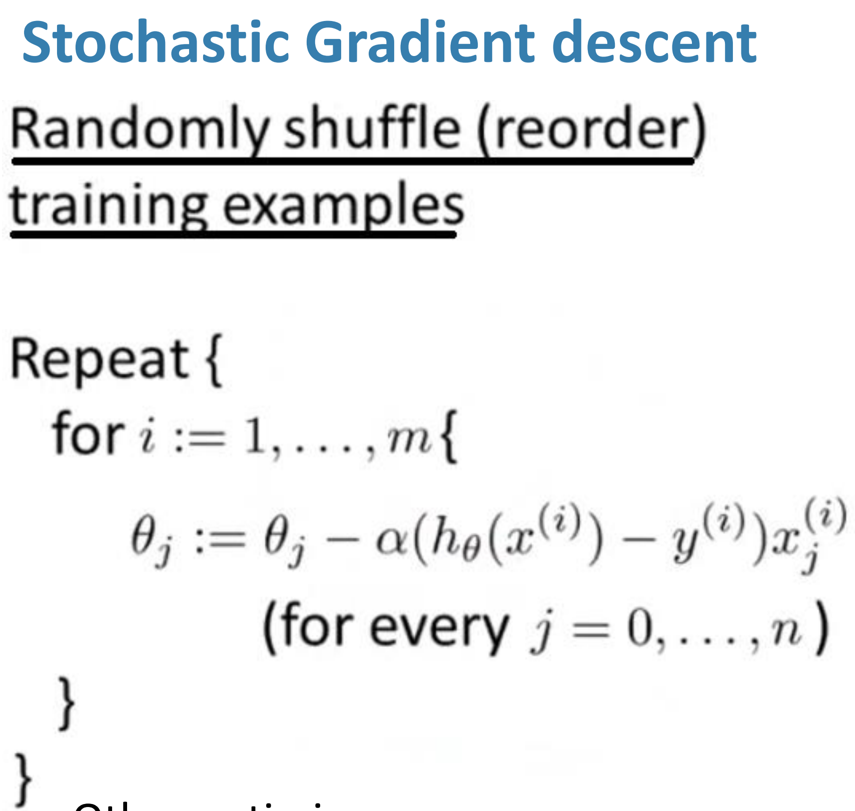


Cross-entropy loss

$$\mathcal{L}_i = -\sum_k y_k \log(S(l_k)) = -\log(S(l))$$

Only the confidence for the GT class is considered.

33



Stochastic Gradient descent

Randomly shuffle (reorder) training examples

```
Repeat {
  for $i := 1, \ldots, m${
      $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
          (for every $j = 0, \ldots, n$)
  }
}
```

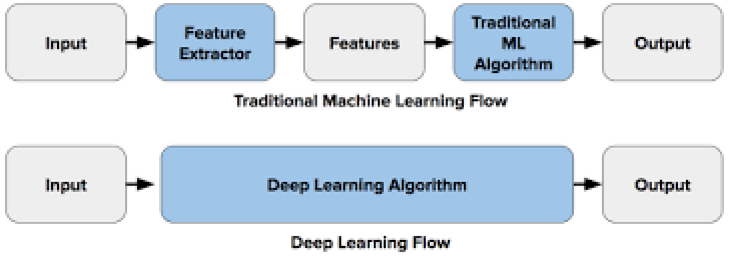
Other loss functions:

1. Negative log likelihood loss
2. Binary cross entropy
3. Mean square error
4. Mean absolute error
5. Hinge loss

Other optimizers:

1. SGD with momentum
2. Adagrad
3. AdaDelta
4. Adam
5. RMSprop

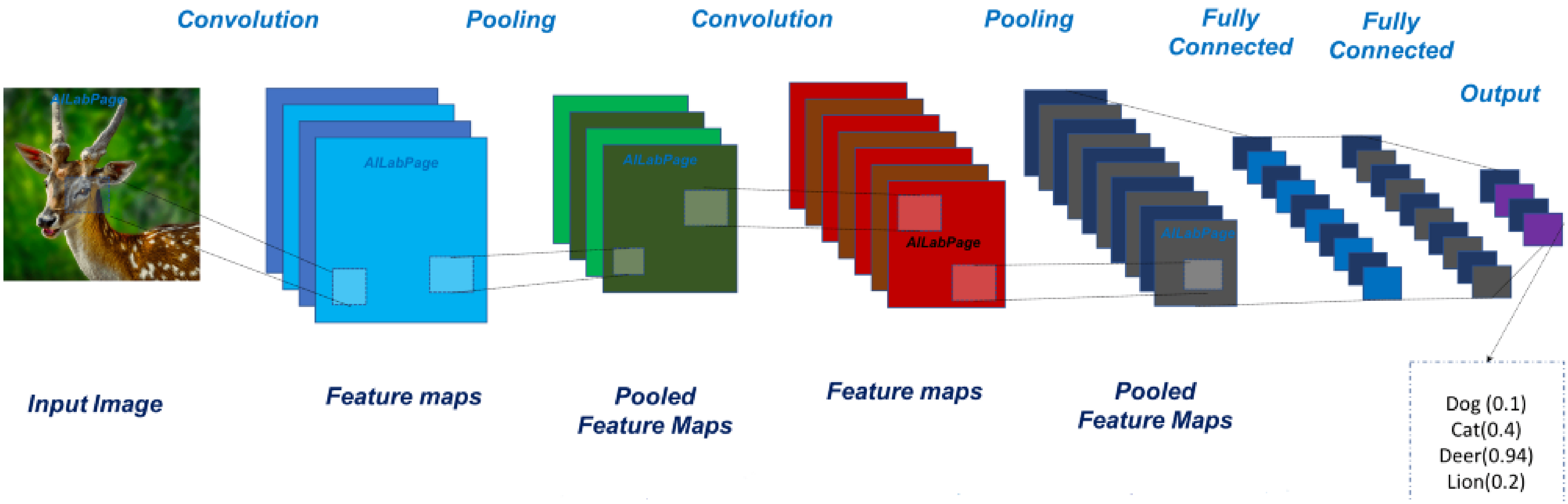
Computer Vision + Deep learning : Convolutional Neural Network.

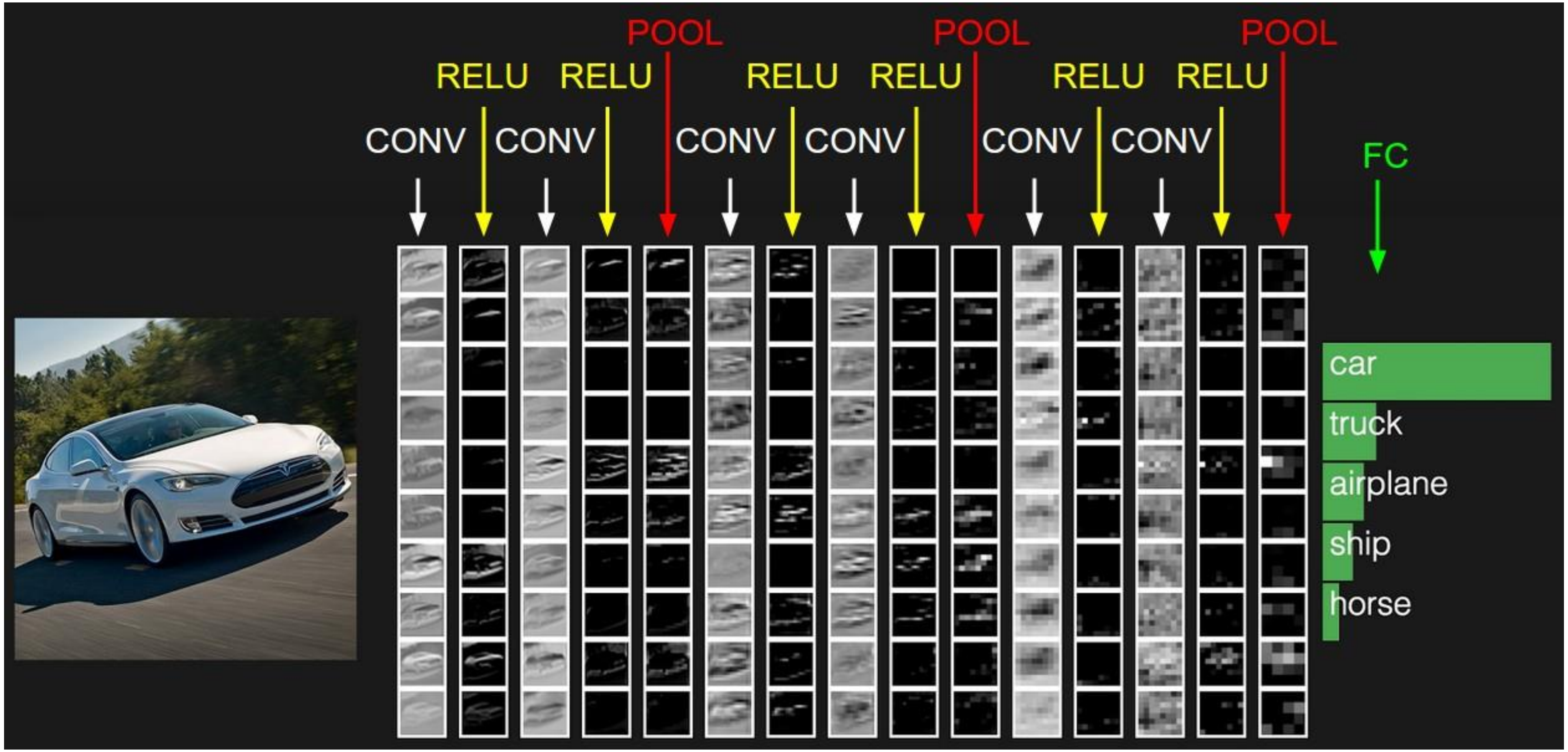


Traditional Machine Learning Flow

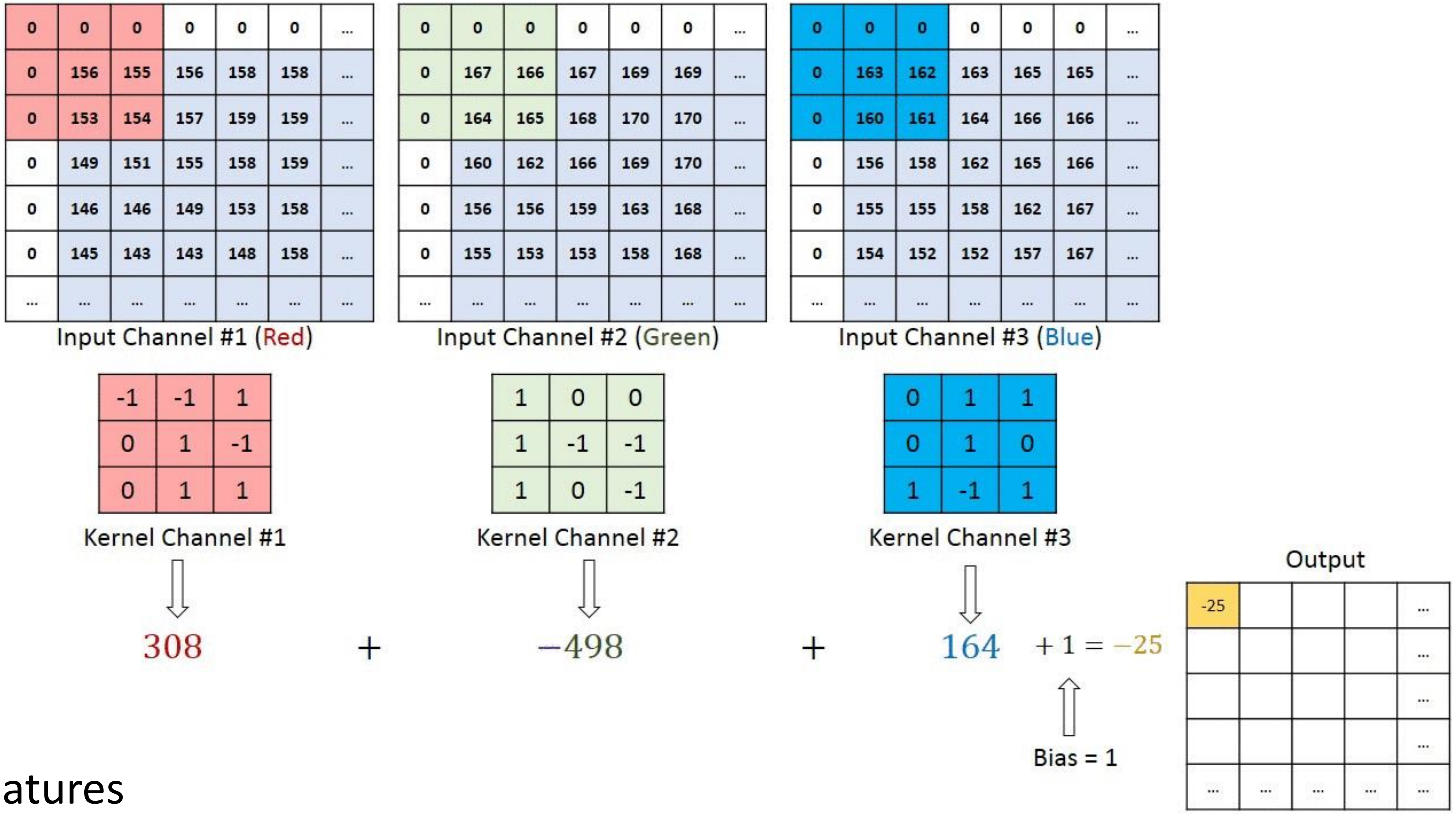
Deep Learning Flow

Traditional way of CV extracts features using some explicit tasks like wavelet transformation , image processing , but in Deep learning everything is being handled by the network itself.




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

Convolutional Neural Networks – different layers of CNN



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 \quad + \quad -498 \quad + \quad 164 \quad +1 = -25$$

Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Action :

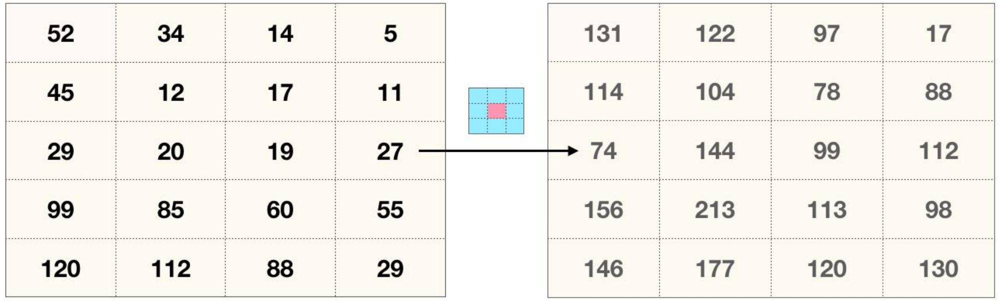
- Apply filters to extract features
- Filters are composed of small kernels, learned
- One bias per filter
- Apply activation function on every value of feature map

Parameters

- Number of Kernels , size of kernels
- Activation function ,striding , padding

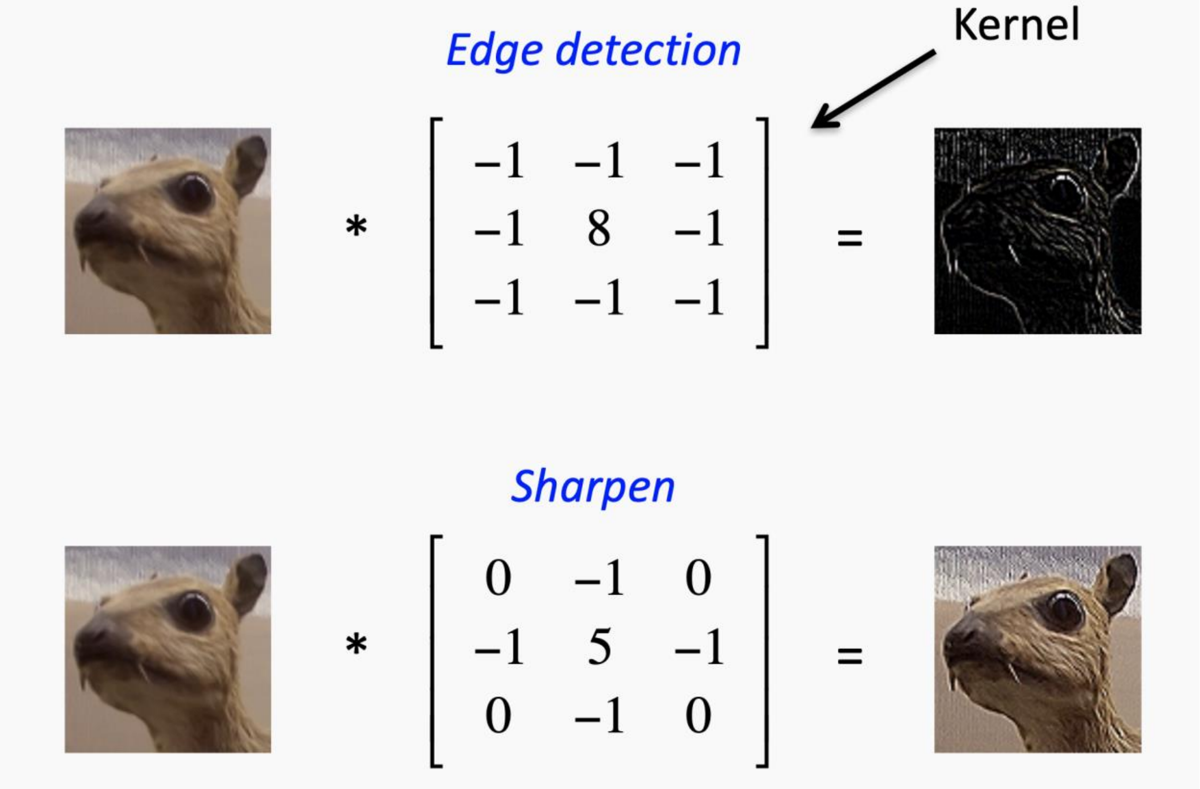
Convolutional Neural Networks – different layers of CNN

Before and after Convolution



52	34	14	5
45	12	17	11
29	20	19	27
99	85	60	55
120	112	88	29

131	122	97	17
114	104	78	88
74	144	99	112
156	213	113	98
146	177	120	130



Edge detection

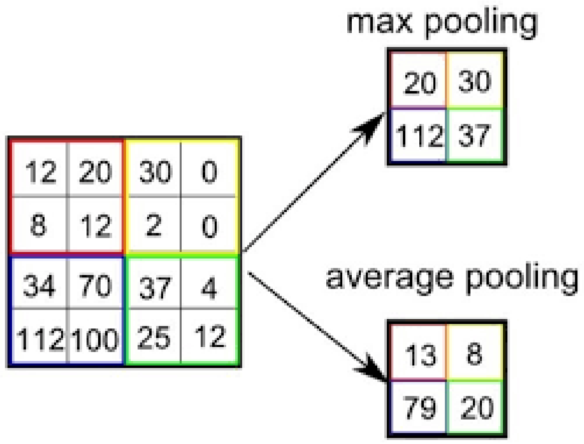
Kernel

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

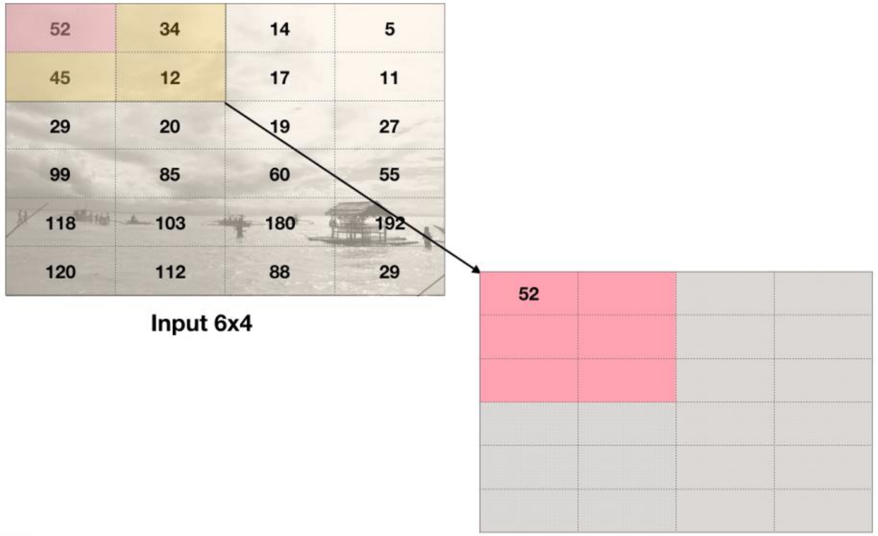
Sharpen

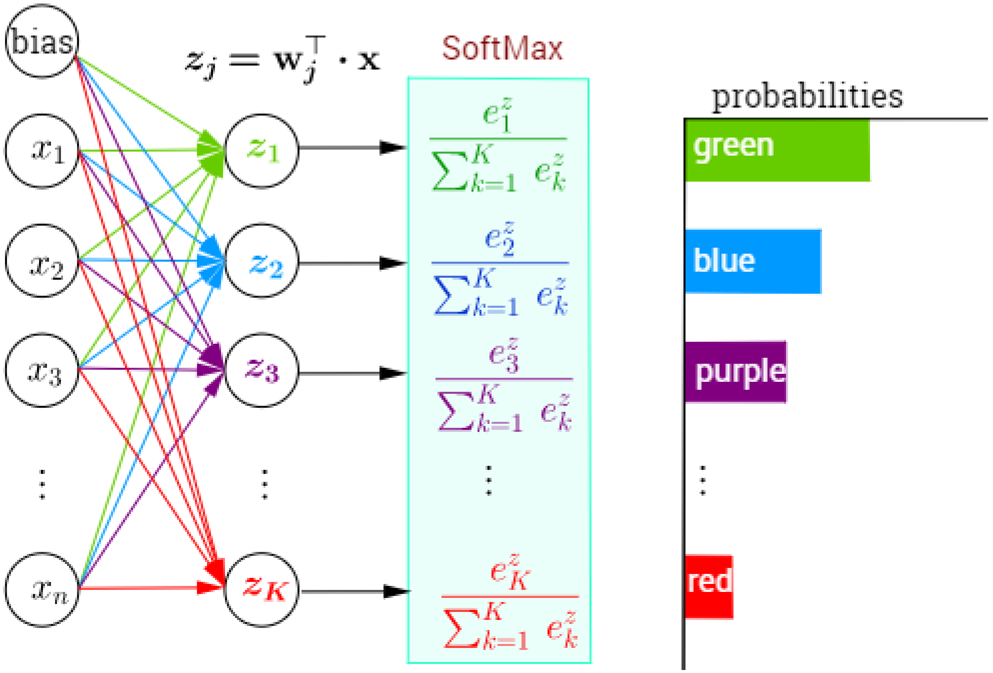
$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

Convolutional Neural Networks – different layers of CNN



Max Pooling




bias
x_1
x_2
x_3
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
z_K
SoftMax
$\frac{e^z_1}{\sum_{k=1}^K e^z_k}$
$\frac{e^z_2}{\sum_{k=1}^K e^z_k}$
$\frac{e^z_3}{\sum_{k=1}^K e^z_k}$
$\frac{e^z_K}{\sum_{k=1}^K e^z_k}$
probabilities
green
blue
purple
red

Pooling layers:

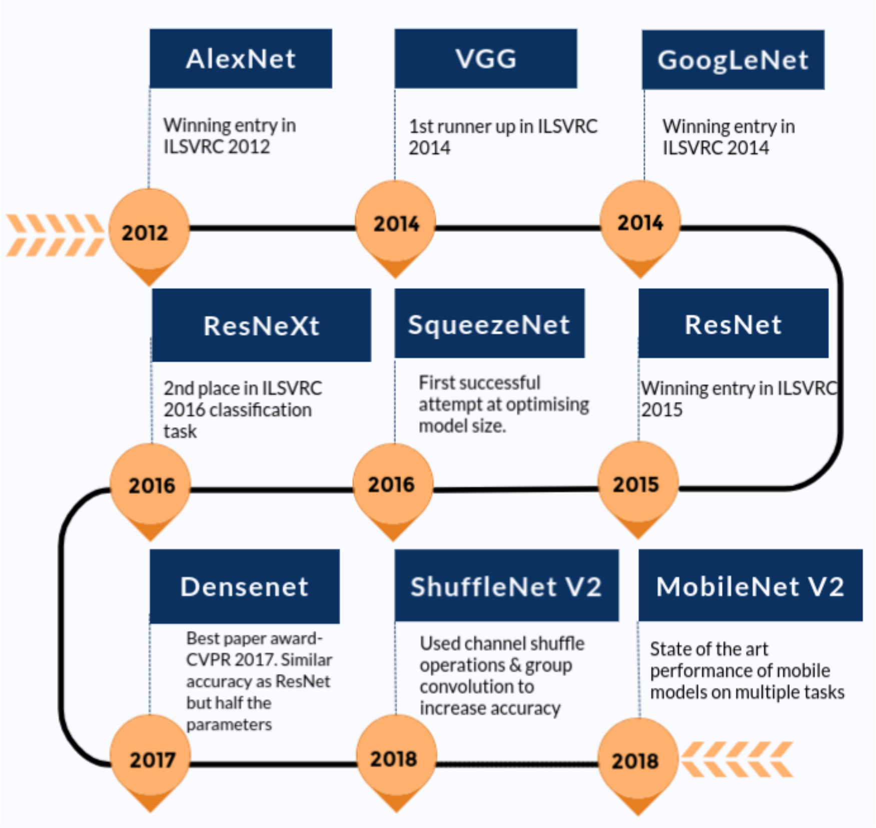
- Reduce Dimensionality
- Extract maximum of / average of a region
- Follow Sliding window approach

Fully connected layers:

- Aggregate information from final feature maps
- Flatten the feature maps for final classification
- Generate final classification with the use of Sigmoid/SoftMax

CNN for Image classification – ImageNet Competition

From ~2012 we see that Convolutional Neural Network (CNN) have become an important tool for object recognition



2012 AlexNet: Winning entry in ILSVRC 2012

2014 VGG: 1st runner up in ILSVRC 2014

2014 GoogLeNet: Winning entry in ILSVRC 2014

2015 ResNet: Winning entry in ILSVRC 2015

2016 SqueezeNet: First successful attempt at optimising model size.

2016 ResNeXt: 2nd place in ILSVRC 2016 classification task

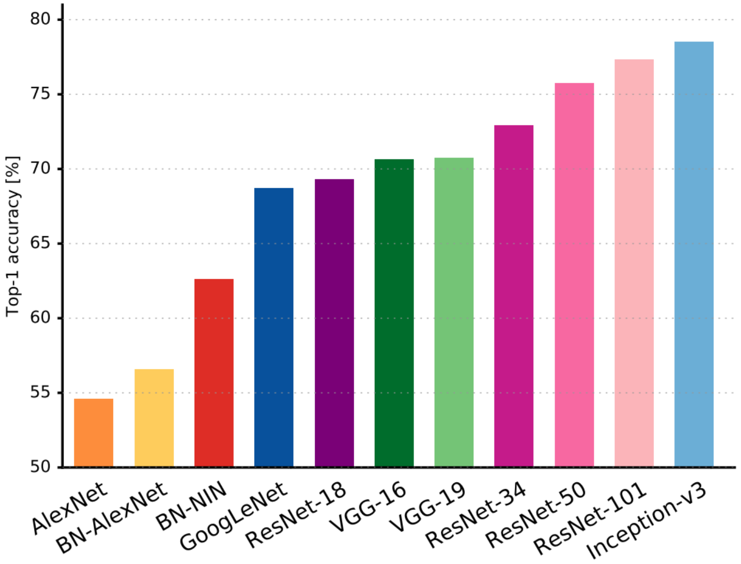
2017 Densenet: Best paper award- CVPR 2017. Similar accuracy as ResNet but half the parameters

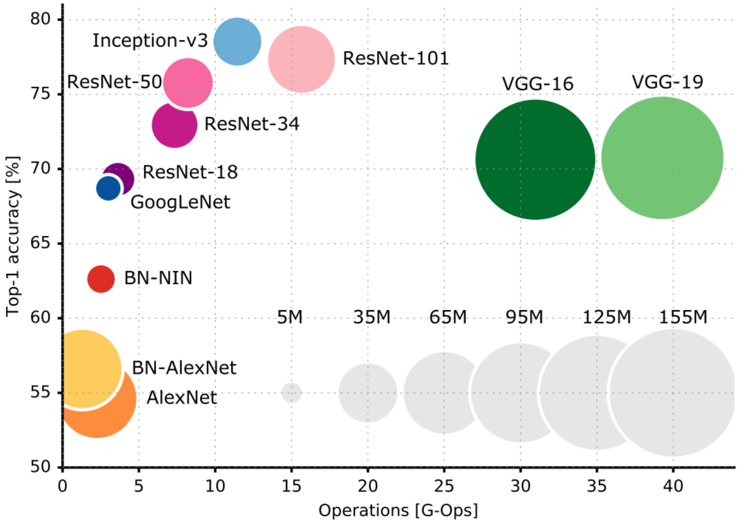
2018 ShuffleNet V2: Used channel shuffle operations & group convolution to increase accuracy

2018 MobileNet V2: State of the art performance of mobile models on multiple tasks

IMAGENET **Large Scale Visual Recognition Challenge**

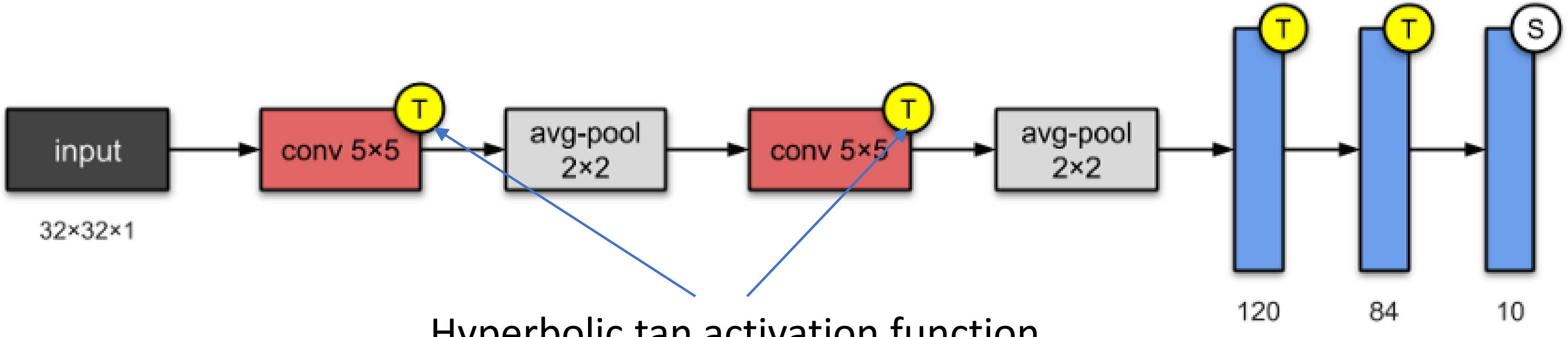
The Image Classification Challenge:
1,000 object classes
1,431,167 images





CNN for Image Classification : Various architectures for Image Classification

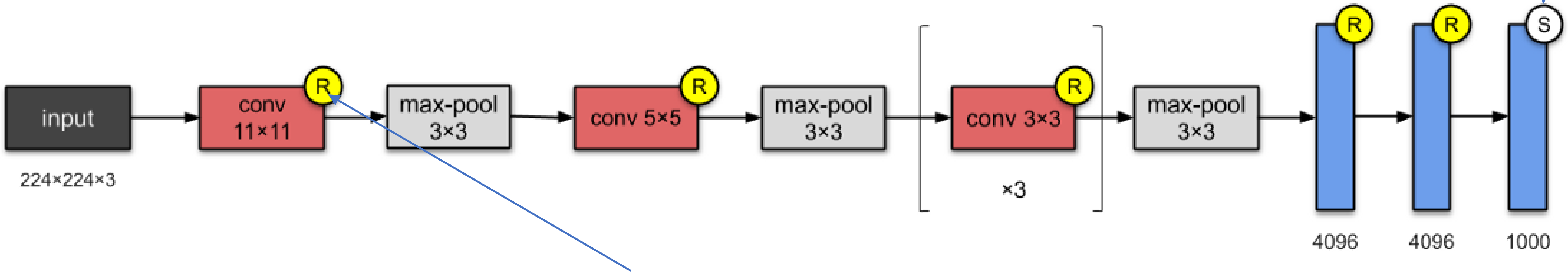
LeNet (1998)



SoftMax classifier

Hyperbolic tan activation function

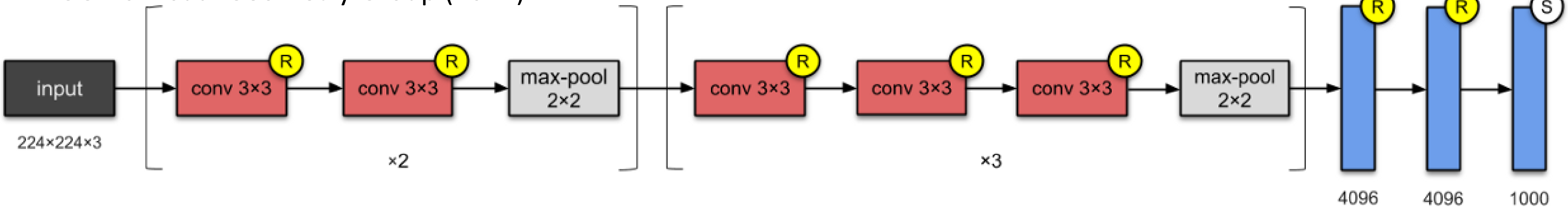
Alexnet (2012)



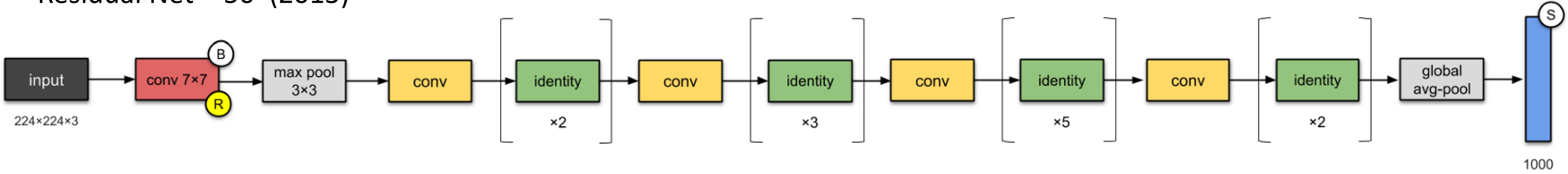
Rectified Linear unit

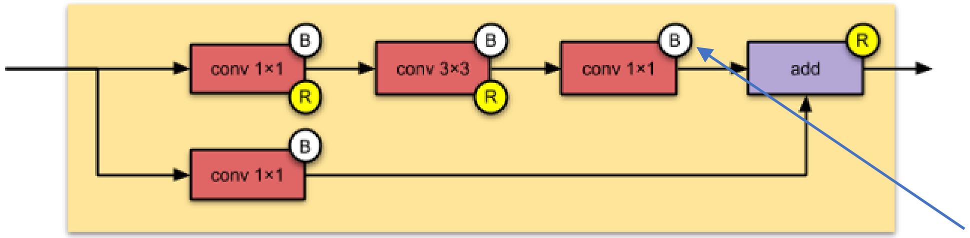
CNN for Image Classification : Various architectures for Image Classification

VGG-16 Visual Geometry Group (2014)



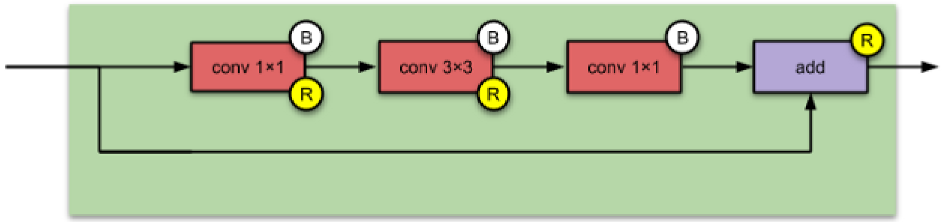
Residual Net – 50 (2015)





Conv block

Batch Normalization



Identity block

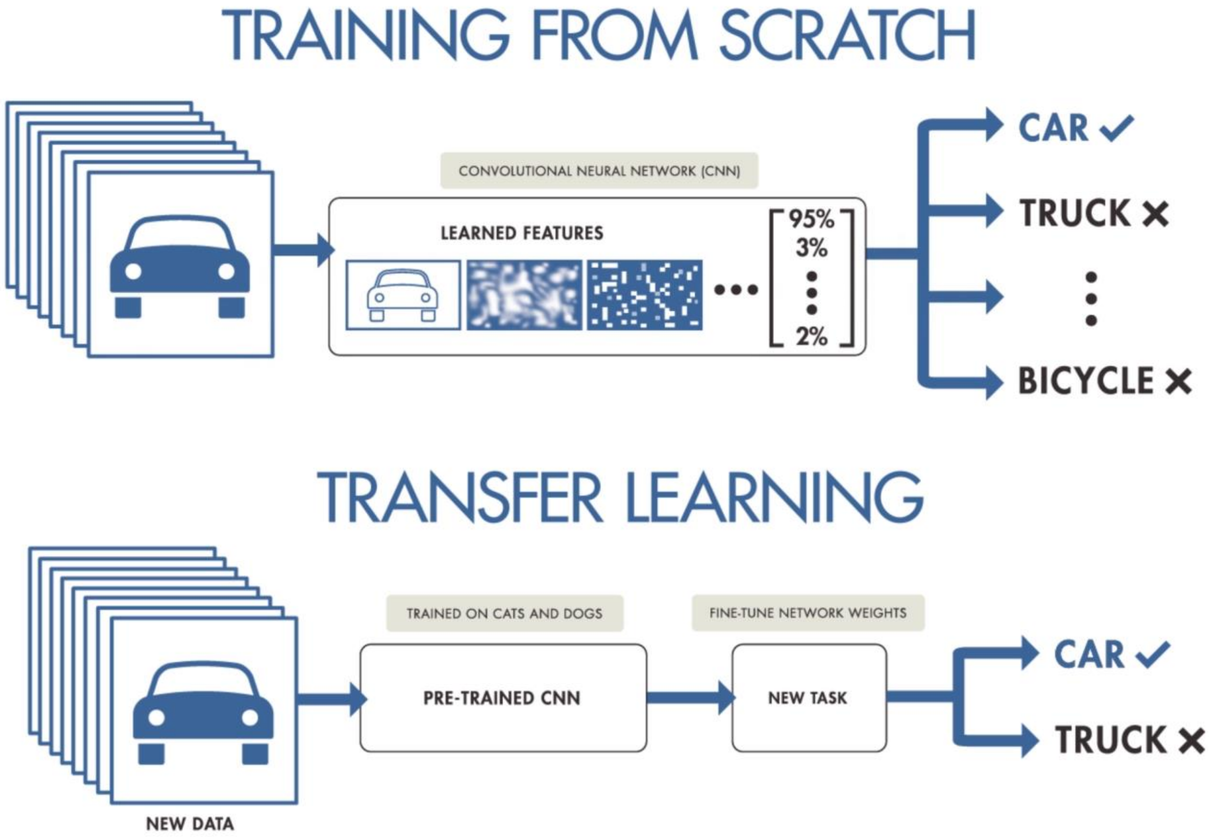
CNN for Image Classification : Transfer Learning

Transfer learning or fine tuning refers to training a network on a huge standard data set (e.g. ImageNet) and then re-tuning just the last few layers of the network for the required specific task

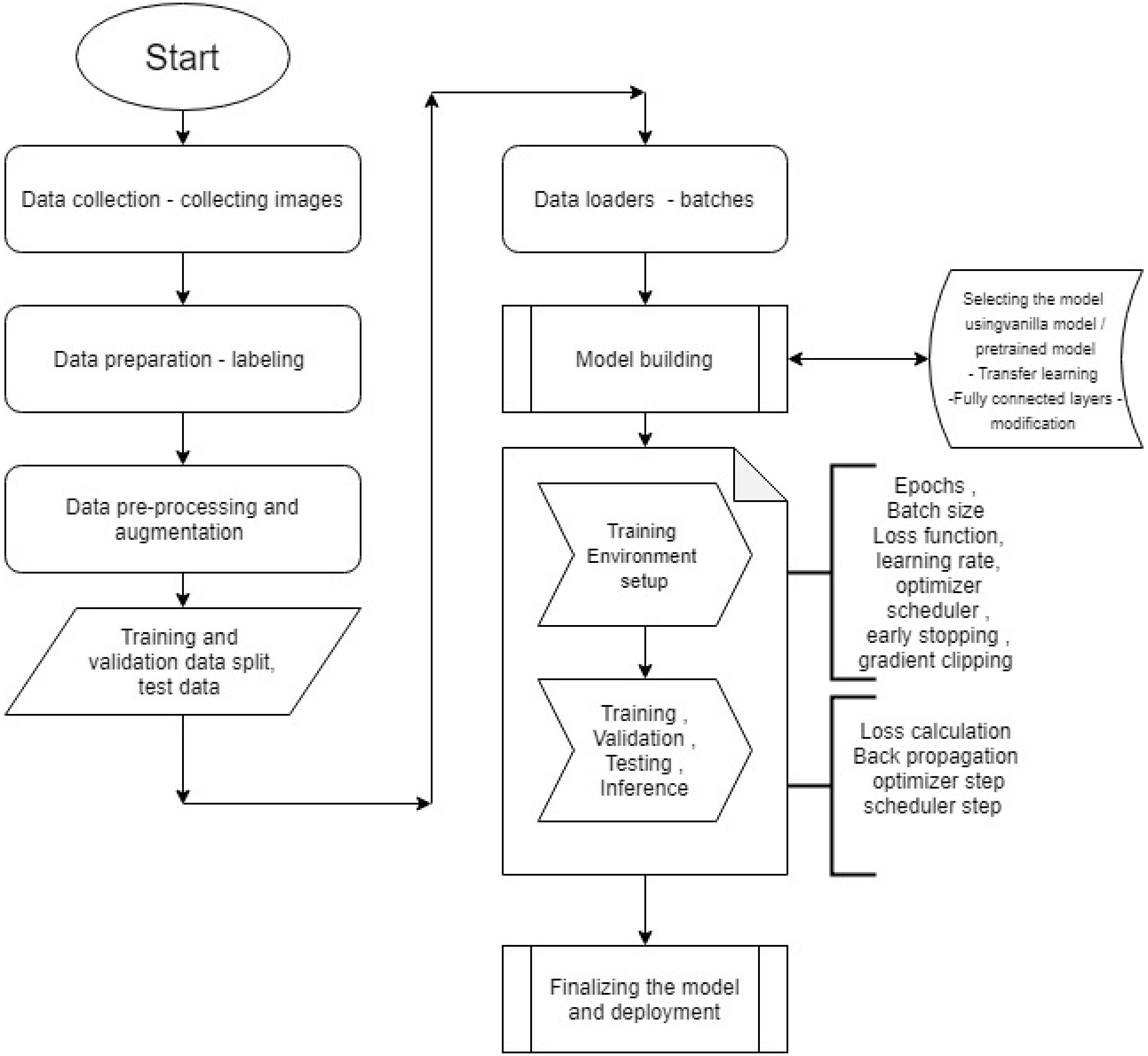
Pretrained models can be downloaded from the model zoos

Caffe2 Microsoft CNTK TensorFlow theano

Chainer PYTORCH dmlc mxnet K



A Deep learning and CNN based Image Classification pipeline



Start

Data collection - collecting images

Data preparation - labeling

Data pre-processing and augmentation

Training and validation data split, test data

Data loaders - batches

Model building

Selecting the model usingvanilla model / pretrained model
- Transfer learning
-Fully connected layers - modification

Training Environment setup

Epochs ,
Batch size
Loss function,
learning rate,
optimizer
scheduler ,
early stopping ,
gradient clipping

Training , Validation , Testing , Inference

Loss calculation
Back propagation
optimizer step
scheduler step

Finalizing the model and deployment

Appendix

Questions!