Advanced Learning Models -1 (CSA-CC-531)

Dr. Sudha S K

Dept. of Computer Science
University of Kerala

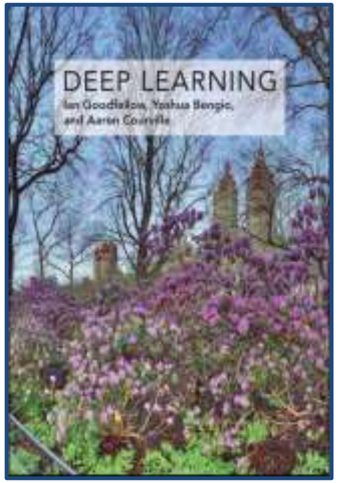
Syllabus

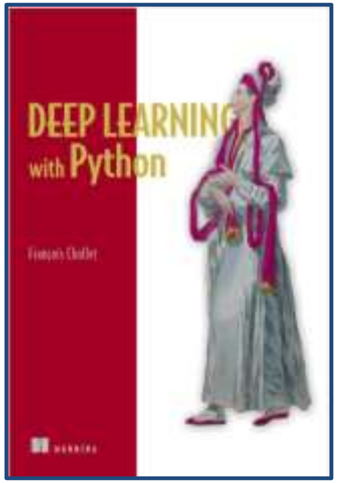
MODULE V: Deep architecture -Recurrent and Recursive networks, Bidirectional RNNs, Deep Recurrent Networks, Recursive Neural Networks, LSTM, GRU. Image captioning, word prediction. Deep Belief networks, Convolutional neural networks, Deep reinforcement learning, Geometric stability, Applications of deep learning.

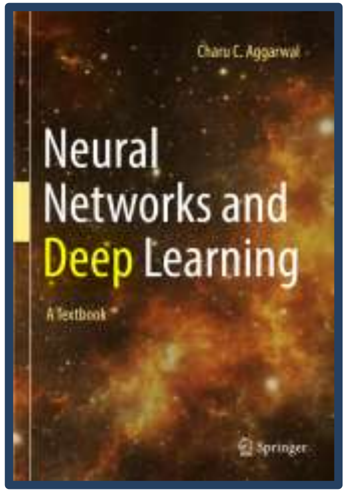
MODULE VI: TensorFlow - Implementing object classification and detection using CNN networks using any of deep libraries like Tensorflow, Keras, Caffe. Generative Networks: Auto encoders, Generative Models, GANs framework, GANs application, Variation auto encoders, DCGANS. Instance recognition, Category recognition, Context and scene understanding.

Recommended Textbooks

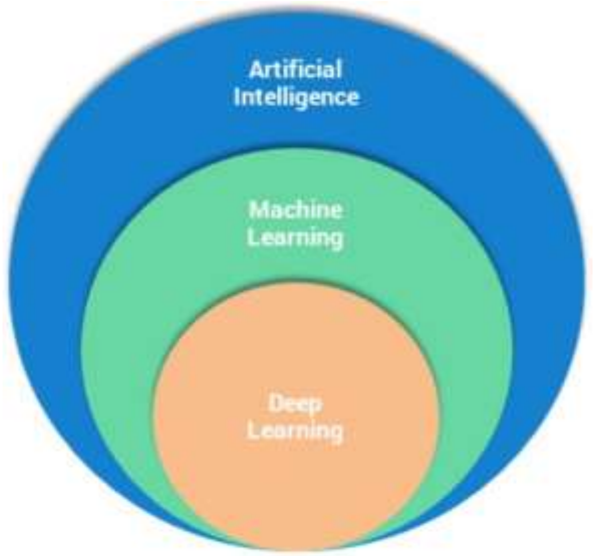
- “Deep Learning” by Goodfellow, Bengio, Courville
- “Deep Learning with Python” by François Chollet
- “Neural Networks and Deep Learning” by Aggarwal Charu







AI Vs ML Vs DL



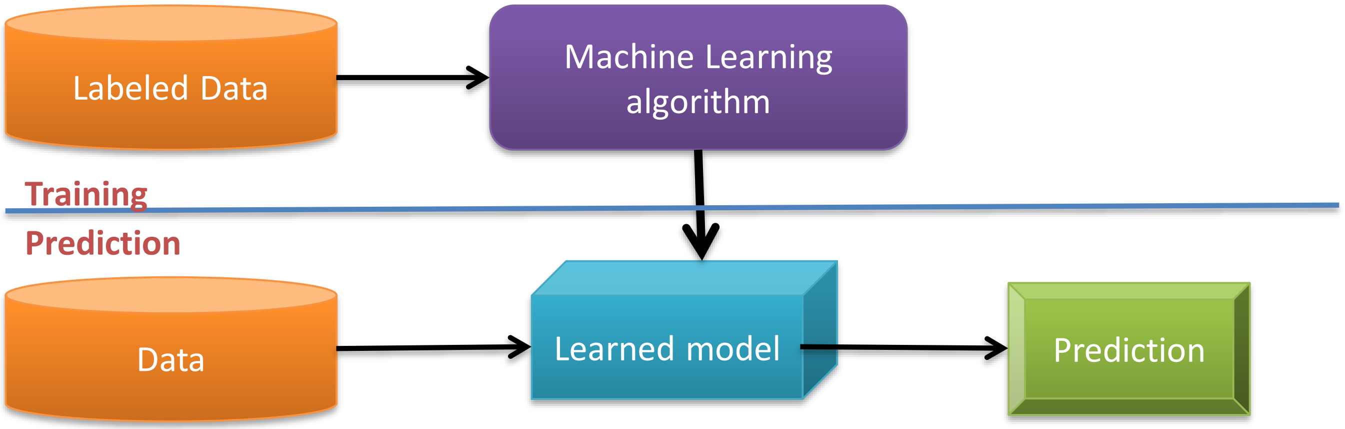
AI is the technique which enables machines to mimic human behavior.

ML is the subset of AI which use statistical methods to enable machines to improve with experience.

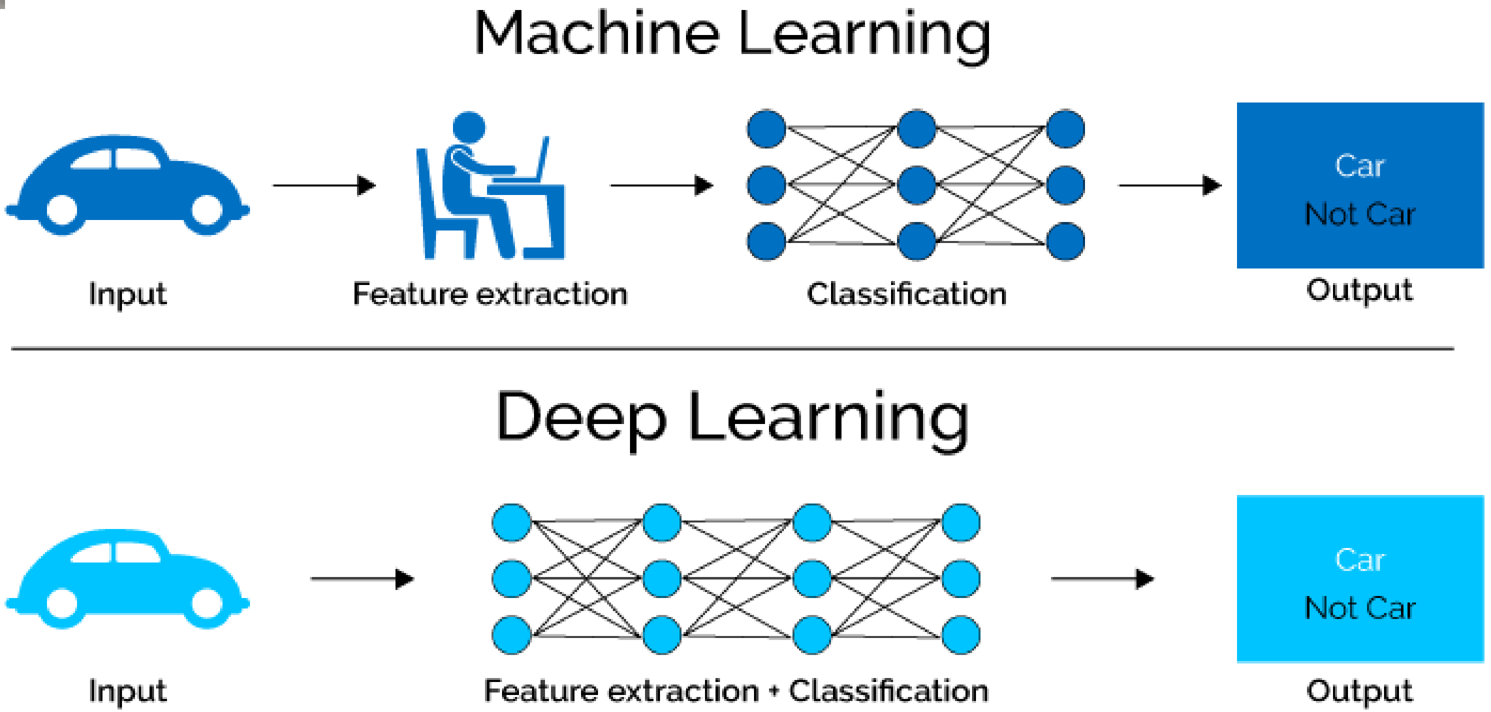
DL is the subset of ML which makes the computation of multi layer neural network feasible.

Machine Learning

Train computers to recognize patterns in data.



ML vs. Deep Learning

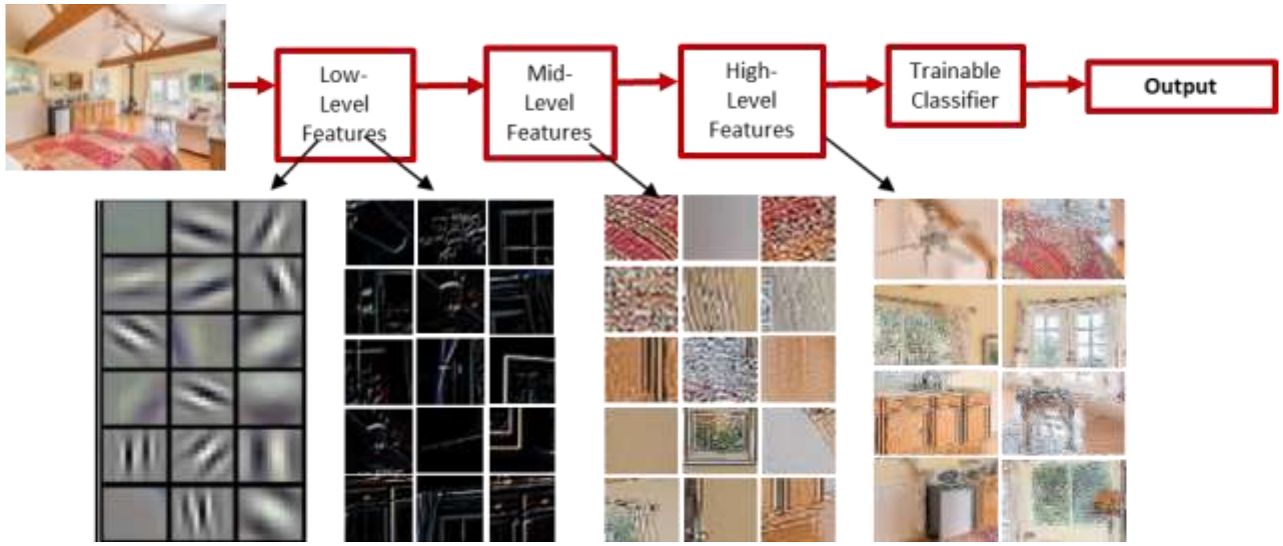


- Deep learning uses **multiple layers** for learning data representations
- No manual feature extraction

ML vs. Deep Learning

Eg: Hierarchical Features Extracted from Multiple Layers

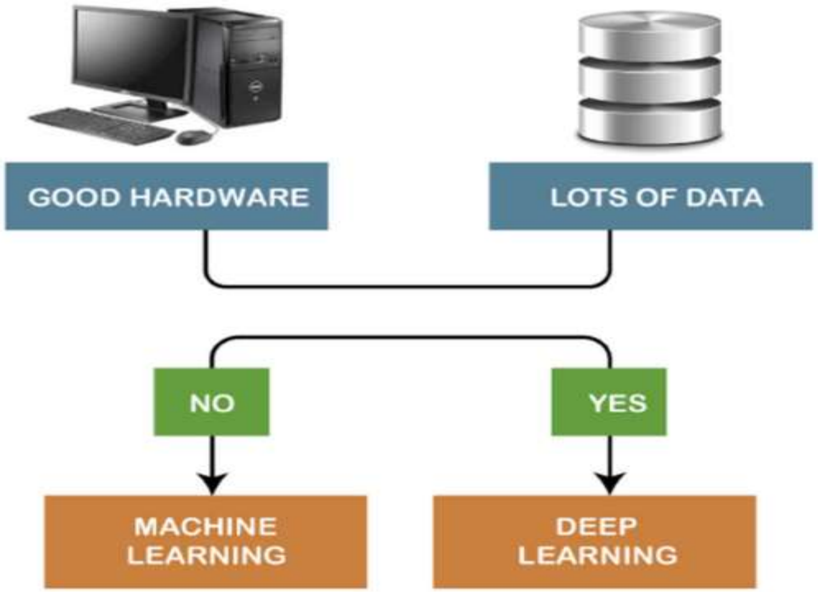
Input image pixels → Edges → Textures → Parts → Objects



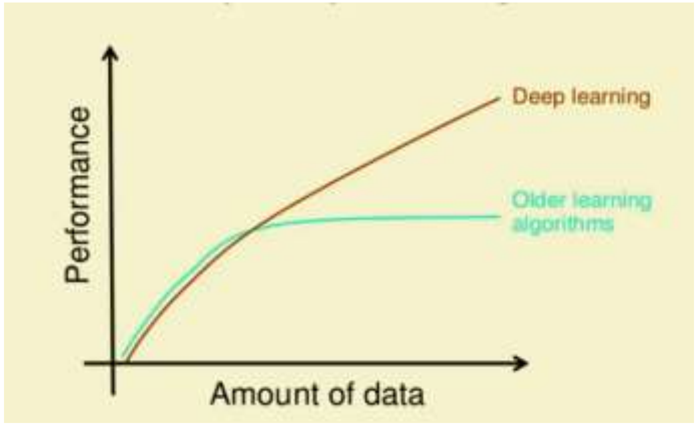
ML vs. Deep Learning

Machine Learning	Deep Learning
Apply **statistical algorithms** to learn the hidden patterns and relationships in the dataset.	Uses **artificial neural network** architecture to learn the hidden patterns and relationships in the dataset.
With **small data size**, traditional ML algorithms are preferable	Requires the **larger volume of data**
Better for the **less complex, low-label** task	**Highly complex** problems such as image classification, natural language processing, and speech recognition
Takes **less time to train** the model	Takes **more time to train** the model
Features are **manually** extracted	Features are **automatically** extracted
It can work on the **CPU** or requires less computing power	Requires a high-performance computer with **GPU**

Which one to Select ML or DL ?



Why DL over ML?



- ➢ Huge amount of data available – Bigdata
- ➢ Improved hardware architectures – GPU, TPU
- ➢ New software architectures

TensorFlow K Keras PyTorch DL4J Caffe

Deep Learning Applications

Computer Vision

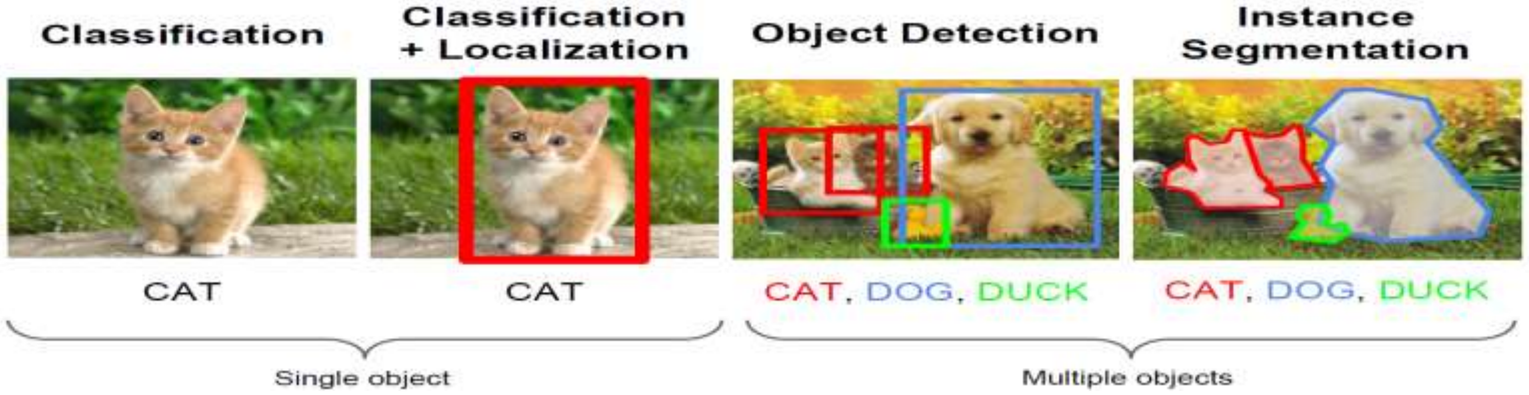
- Self-Driving Cars
- Automatic Image Caption Generation
- Object Detection and Recognition
- Image Classification
- Image Segmentation

Natural Language Processing

- Automatic Text Generation
- Language Translation
- Sentiment Analysis
- Speech Recognition

Reinforcement Learning

- Voice Controlled Assistance
- Game Playing
- Robotics

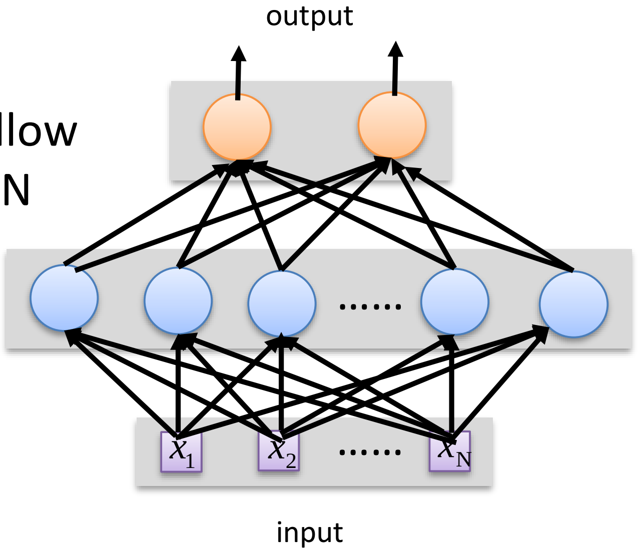


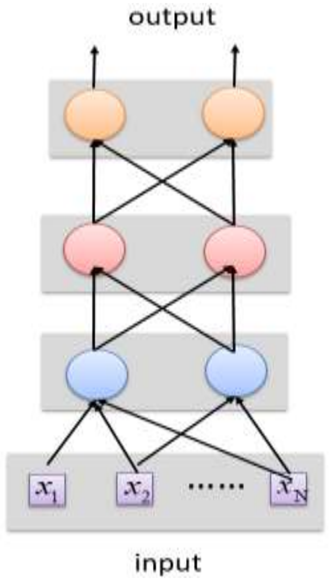
Neural Network Architectures

Shallow Neural Network - one hidden layer between the input and output

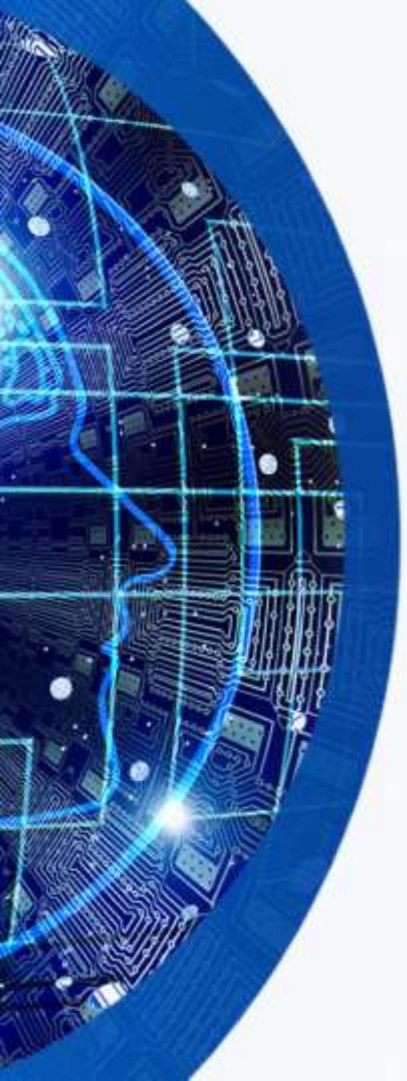
Deep Neural Networks - incorporates several numbers of hidden layers in between the input and output layers

Shallow NN





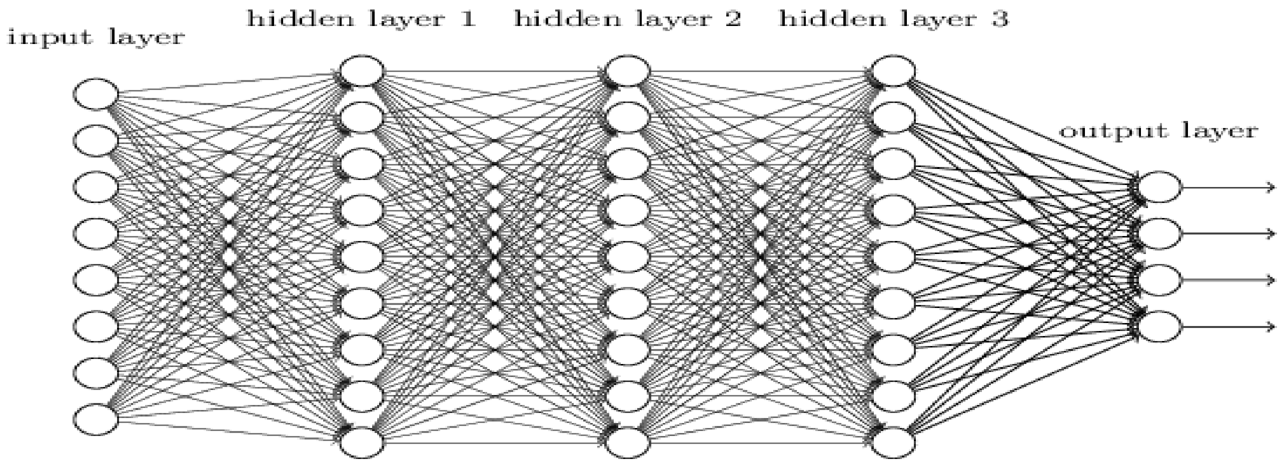
Deep NN



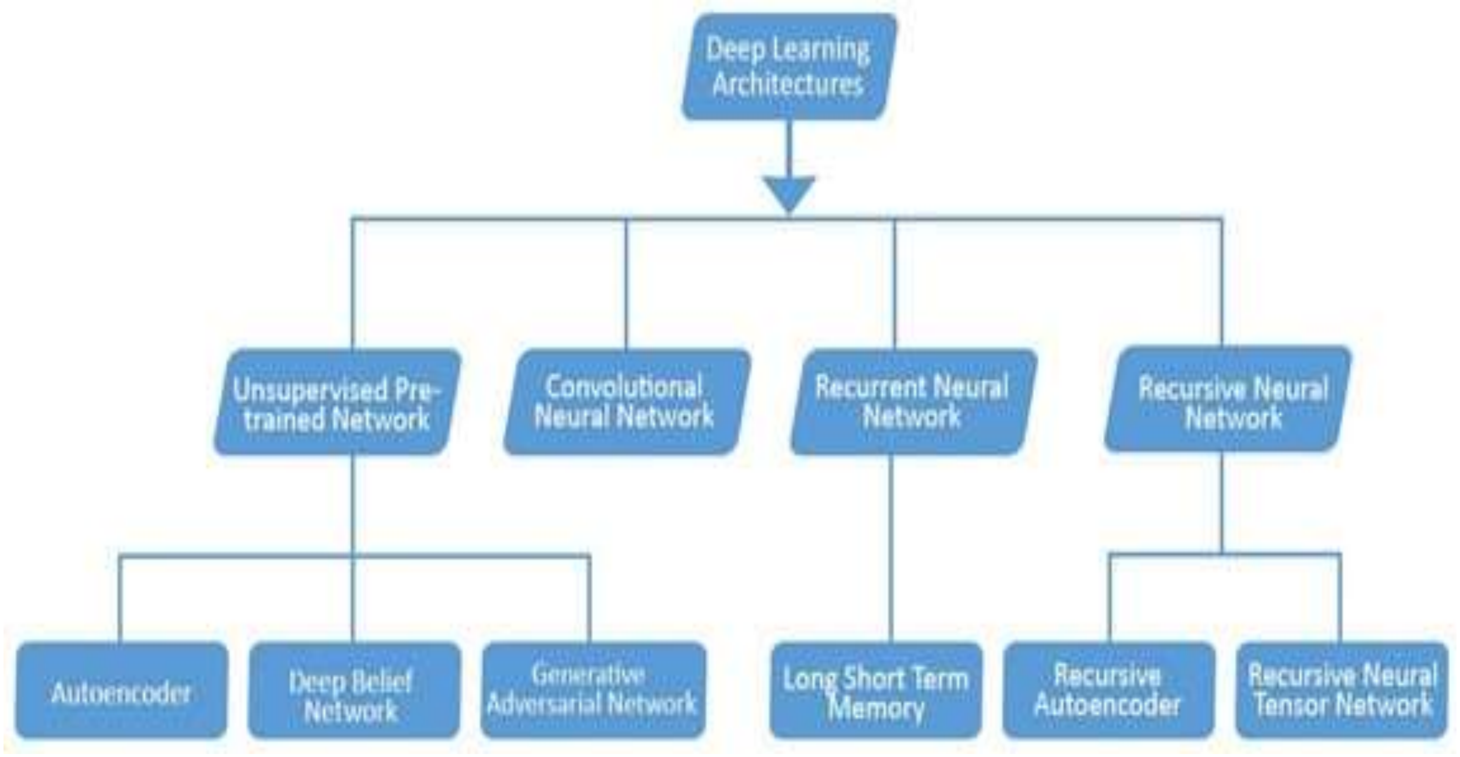
Introduction to DL

What is Deep Learning ?

Deep Learning is a **computer software** that mimics the network of neurons in a brain.
It is a subset of **machine learning** based on the concept of **artificial neural networks** with representation learning.
It is called deep learning because it makes use of **deep neural networks**.
The learning can be **supervised**, **semi-supervised** or **unsupervised**.



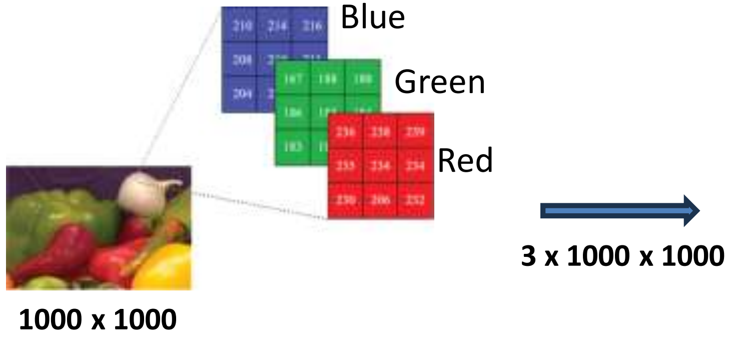
Deep Learning Architectures

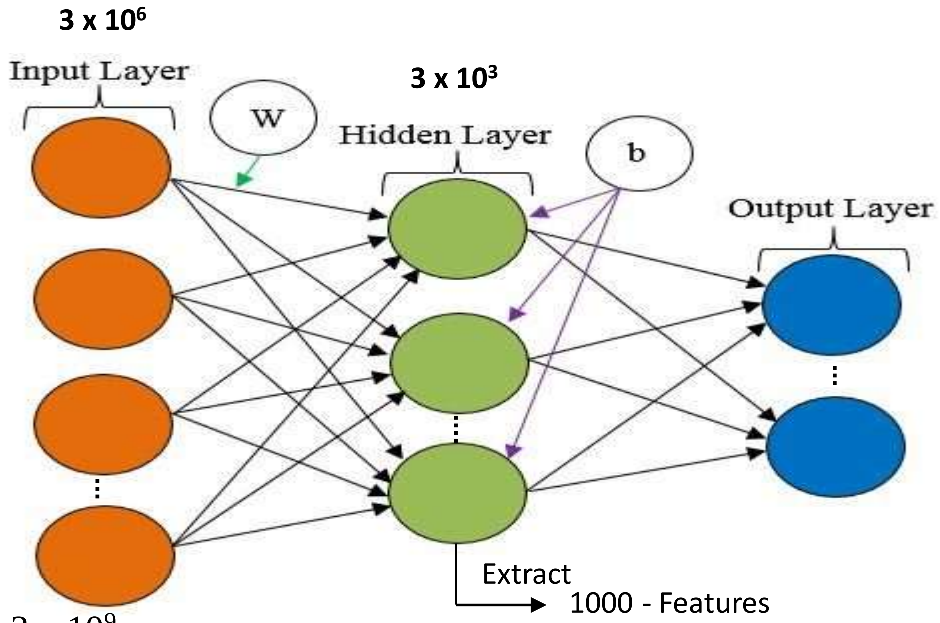

Deep Learning Architectures
Unsupervised Pre-trained Network
Convolutional Neural Network
Recurrent Neural Network
Recursive Neural Network
Autoencoder
Deep Belief Network
Generative Adversarial Network
Long Short Term Memory
Recursive Autoencoder
Recursive Neural Tensor Network

Convolutional Neural Networks (CNN/ConvNet)

Neural Networks are the building blocks of deep learning
CNN are built on top of regular neural network

Regular ANN Vs. CNN





- ➢ Then all the weights to train $= 3 \times 10^6 \times 10^3 = 3 \times 10^9$
- ➢ CPU/GPU cannot handle the load
- ➢ Training Time will increase – All leading to **overfitting**

Understanding Convolutional Neural Networks

- A CNN is a **feed-forward neural network.**
- CNN is the commonly used **DL architecture** designed for working with **two-dimensional image data**, although they can be used with **one-dimensional** and **three-dimensional** data. CNN consists of:

  (i) Convolutional Layer

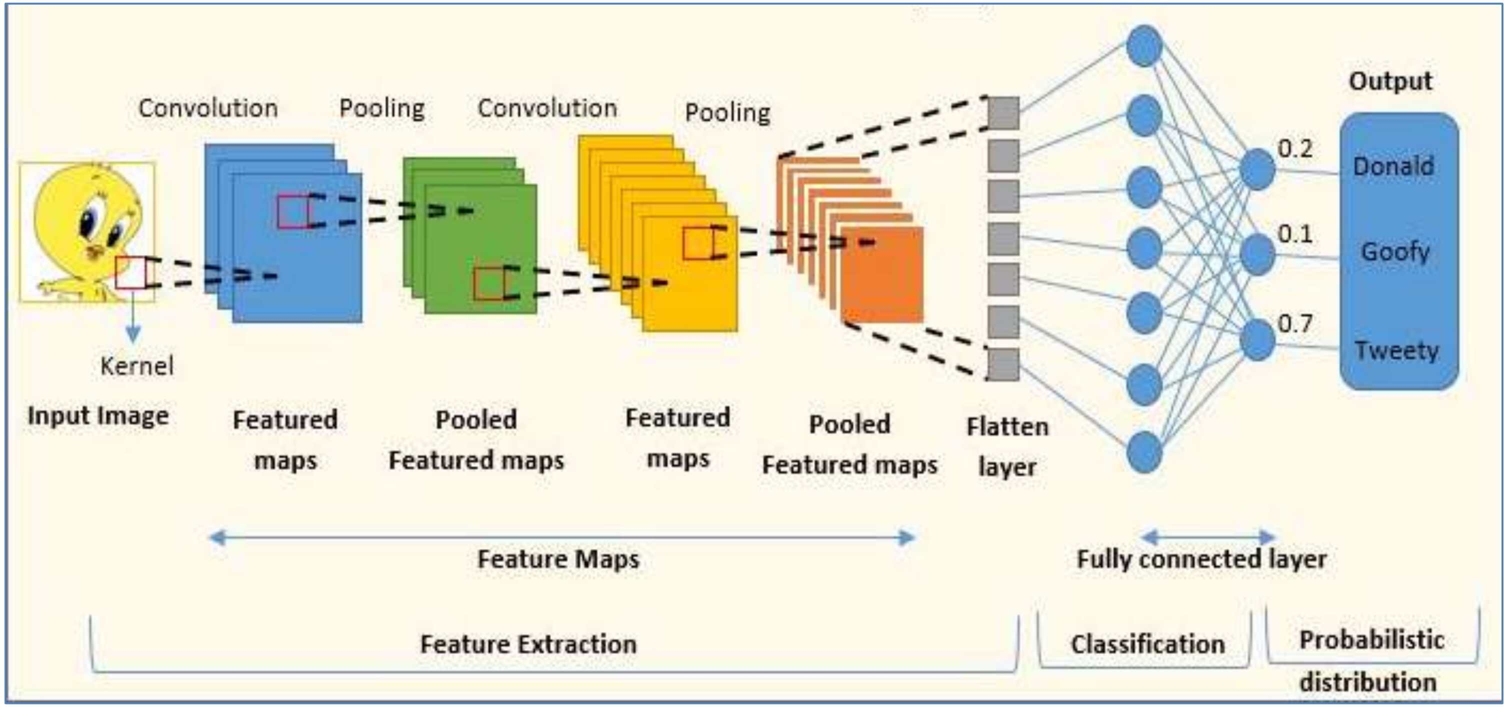
  (ii) Pooling Layer (POOL) and

  (iii) Dense Layer or Fully Connected (FC) Layer

  (iv) Activation Layer

Understanding CNNs Contd…

A Typical CNN



Understanding CNNs Contd…

Convolutional Layer

- The **basic unit** of the CNN architecture is a convolutional layer
- Convolutional layer performs an operation called a “**convolution**“ and computes the **convolution of input images** along with the NN **weights** and performs **feature extraction**
- Convolution operation is **linear**
- This layer’s main **processing parameters** are a group of learnable **filters or masks or kernels**, which generate the input feature maps

Understanding CNNs Contd…

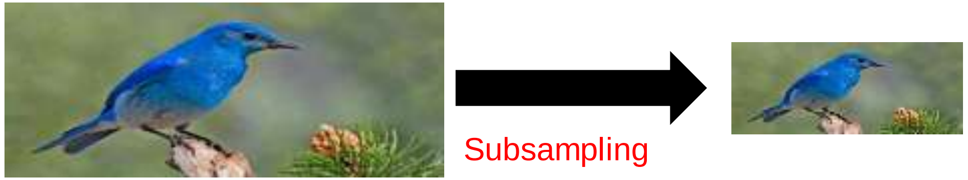
CNN Filters

- ➢ Multiple filters are there to **extract specific features** from input data.
- ➢ Filter values are **randomly initialized** to 0 or 1.
- ➢ Filters like Gaussian Blur, Prewitt Filter, Sobel Filter etc.. can also be used.
- ➢ Filters
  - ✓ in first layer detect horizontal, vertical, and diagonal **edges**
  - ✓ in the next layer detect **textures**
  - ✓ in the last layer detect **shapes/objects**

Understanding CNNs Contd…

Pooling Layer (Subsampling Layer)

- A CNN architecture can be built by **stacking** pooling and convolutional layers in an intervened manner.
- Using the pooling layer **reduces the spatial resolution** of the feature maps. The frequently used pooling methods are
  - Average Pooling
  - Max Pooling
- They are **spatially invariant** to input **distortions** and **translations** with less overfitting
- The resulting feature map from the pooling layer is **flattened**, which involves converting the feature map matrix into a single column

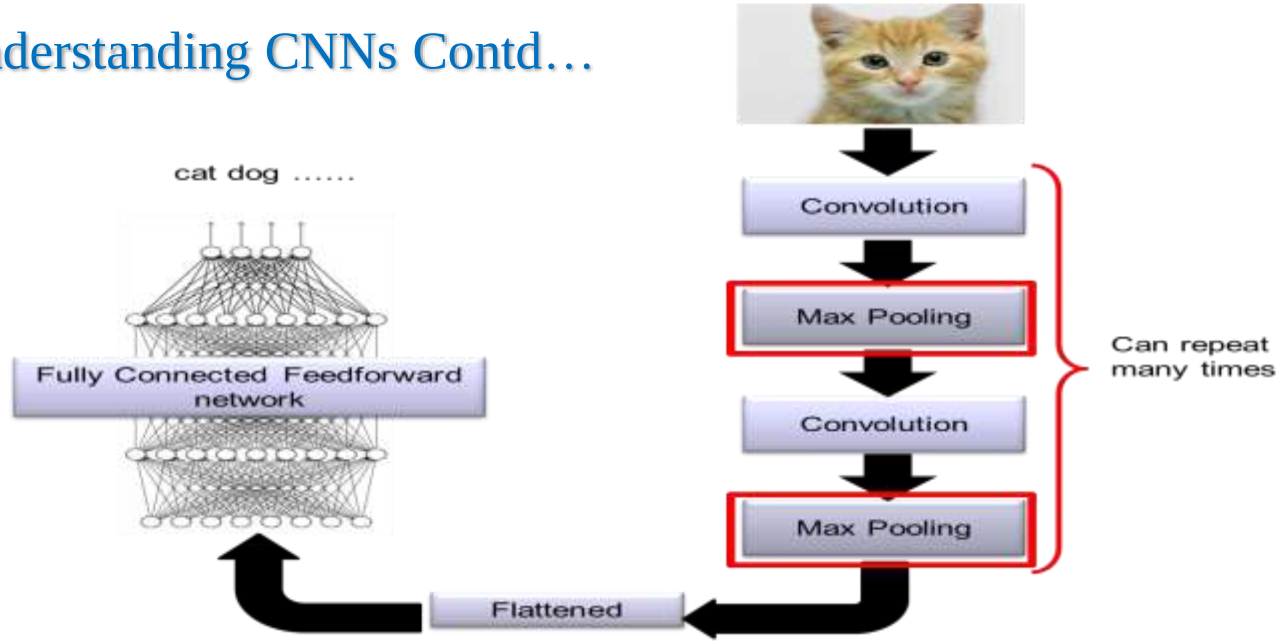


Understanding CNNs Contd…

Fully Connected Layer

- The flattened feature map from the pooling layer is given to dense FC layers and is utilized as the **final layer for the classification**.
- **More abstract feature** representations are extracted while moving through the entire network.
- The FC layers perform **high-level reasoning** and generate new features from the existing features.
- The **neurons** in the FC layer are **fully connected** to all the previous layers.
- Generally, this layer has the **most number of weights** with no sharing.
- The FC layer can thus **take time to train** as compared to other layers.

Understanding CNNs Contd…



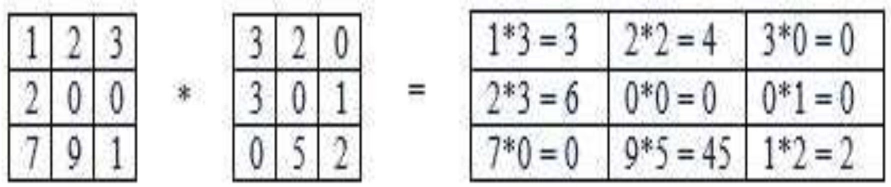
- Generally, there is **no hard boundary** existing between **Conv.layer-Pooling layer blocks**.
- Every layer is authenticated by few hyperparameters such as
  - i) the number of **filters**-to-learn,
  - ii) the **stride** between different windows (number of steps it moves)
  - iii) an optional **zero-padding** that controls the size of the output layer

Understanding CNNs Contd…

Convolutional Operation

- Mathematically, convolution is the **summation of the element-wise product** of two matrices.

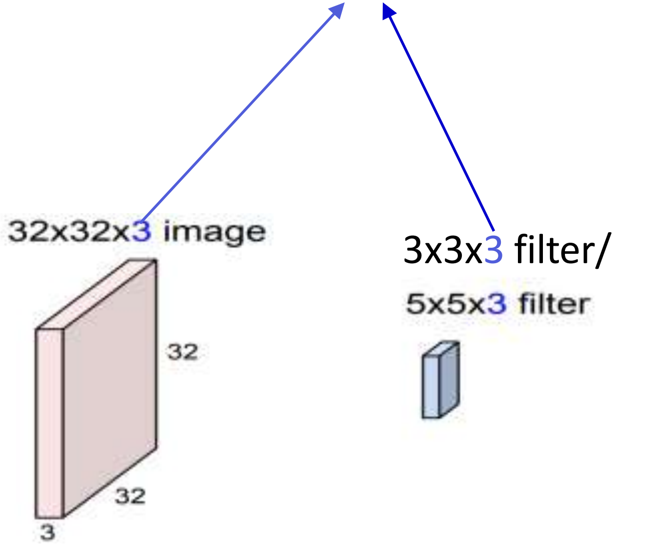
Filters always extend the full depth of the input volume



$$\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 2 & 0 & 0 \\ \hline 7 & 9 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 3 & 2 & 0 \\ \hline 3 & 0 & 1 \\ \hline 0 & 5 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 1*3=3 & 2*2=4 & 3*0=0 \\ \hline 2*3=6 & 0*0=0 & 0*1=0 \\ \hline 7*0=0 & 9*5=45 & 1*2=2 \\ \hline \end{array}$$

Input Filter Output

$3+4+0+6+0+0+0+45+2 = 60$

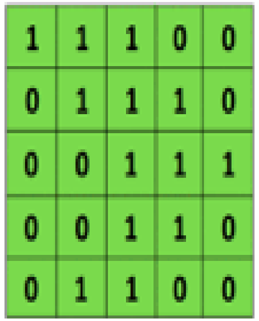


Input-Conv.Layer

Understanding CNNs Contd…

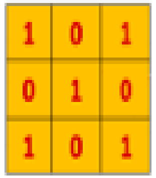
Convolutional Operation Eg:

- Place filter on the top left corner of image
- The filter **slides over** the image matrix
- Multiply filter values by pixel values, add the results
- Move filter to right one/two pixel at a time, and repeat this process **- Stride**
- When at top right corner, move filter down one pixel and repeat process
- Process ends when we get to bottom right corner of image



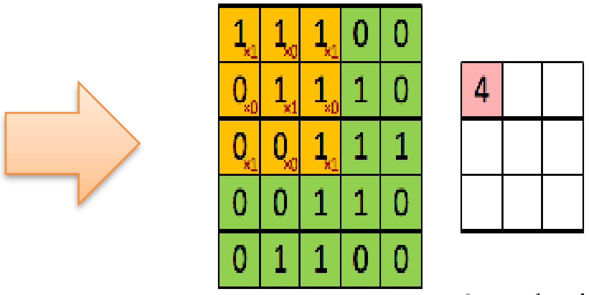
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input matrix



1	0	1
0	1	0
1	0	1

Convolutional 3x3 filter



$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

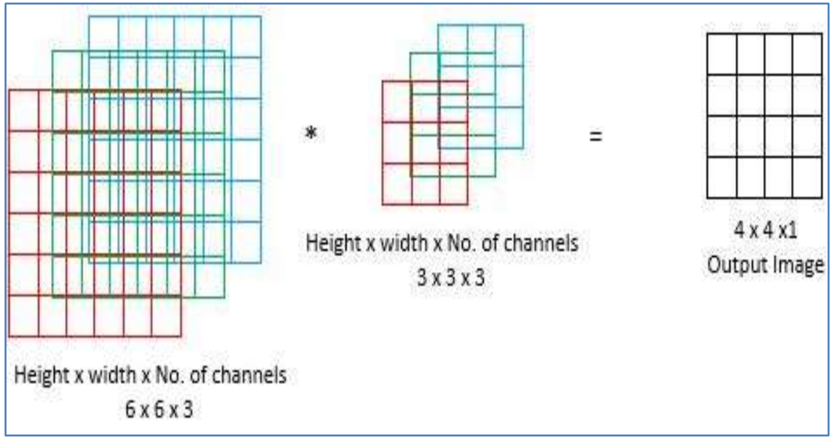
Convolved Feature

Understanding CNNs Contd…

Dimensions of the Convolved Output

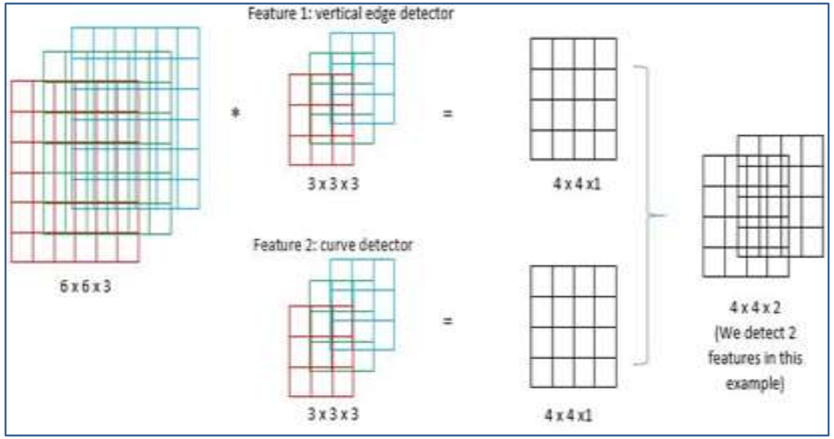
For an input image size **n x n** & filter size **f x f** ,
After convolution, the size of the output image is:

(**Size of input image – filter size + 1**)



*
=
Height x width x No. of channels
6 x 6 x 3
Height x width x No. of channels
3 x 3 x 3
4 x 4 x1
Output Image

Single Filter



Feature 1: vertical edge detector
*
=
6 x 6 x 3
3 x 3 x 3
4 x 4 x1
Feature 2: curve detector
=
3 x 3 x 3
4 x 4 x1
4 x 4 x 2
(We detect 2 features in this example)

Multiple Filter

Understanding CNNs Contd…

Convolution operator parameters

- Filter size
- Padding
- Stride
- Activation function

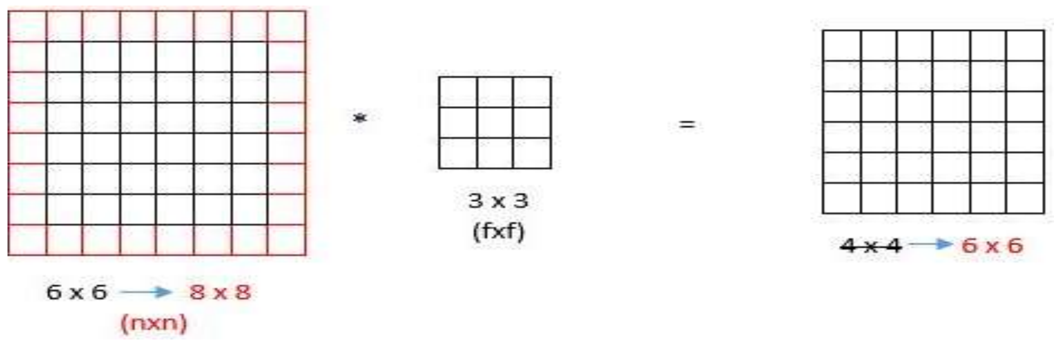
Filter size

- Filter size can be 5 x 5, 3 x 3, and so on
- **Larger filter** sizes should be **avoided**
  - ✓ As learning algorithm needs to learn filter values (weights)
- **Odd sized** filters are preferred to even sized filters
  - ✓ Nice geometric property of all input pixels being around output pixel

Understanding CNNs Contd…

Padding

- Every time we apply a convolution operator, the size of the image will **shrink**
- Lose a lot of information because of image shrinking after several convolution operation
- To keep the image size the **same**, we can use padding
  - ✓ We pad input in every direction with 0's before applying filter
  - ✓ If padding is 1 by 1, then we add 1 zero in every direction
  - ✓ If padding is 2 by 2, then we add 2 zeros in every direction, and so on



Understanding CNNs Contd…

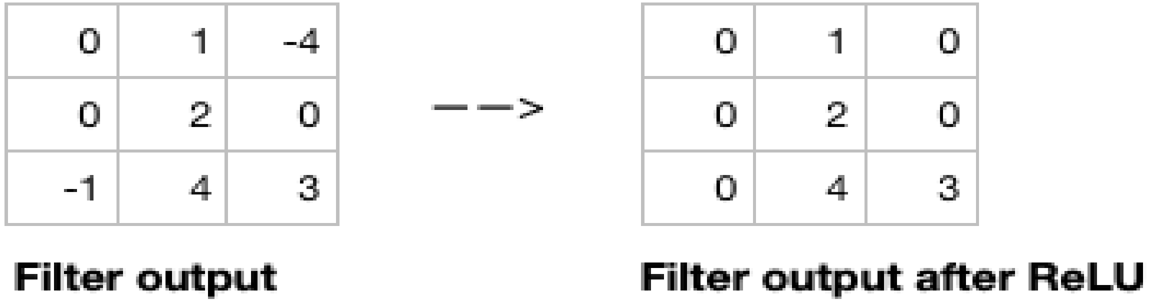
Stride

- The stride indicates the **number of pixels** by which the filter moves horizontally & vertically over the input image during convolution.
  - ✓ Stride 1: move filter one pixel to the right/down
  - ✓ Stride 2: move filter two pixels to the right/down
- Stride depends on what we expect in our output image.
- We prefer a smaller stride size if we expect **several fine-grained** features to reflect in our output.
- On the other hand, if we are only interested in the **macro-level** features, we choose a larger stride size.

Understanding CNNs Contd…

Activation Function

- After filter applied to whole image, apply activation function to output to introduce **non-linearity**
- Preferred activation function in CNN is **Rectified Linear Unit (ReLU)**
- ReLU leaves outputs with positive values as is, replaces negative values with 0



0	1	-4
0	2	0
-1	4	3

Filter output

$\dashrightarrow$

0	1	0
0	2	0
0	4	3

Filter output after ReLU

Understanding CNNs Contd…

Pooling Operation

- ➢ Pooling layer performs **down sampling** to reduce spatial dimensionality of input
- ➢ This **decreases** number of parameters
  - ✓ Reduces learning time/computation
  - ✓ Reduces likelihood of overfitting
- ➢ Two types
  - ✓ **Max** Pooling
  - ✓ **Average** Pooling
- ➢ Usually a **2 x 2 filter** with **stride 2** is preferred

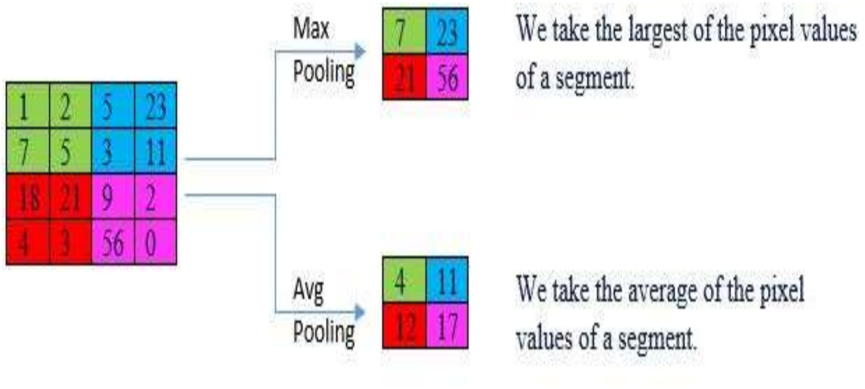
Understanding CNNs Contd…

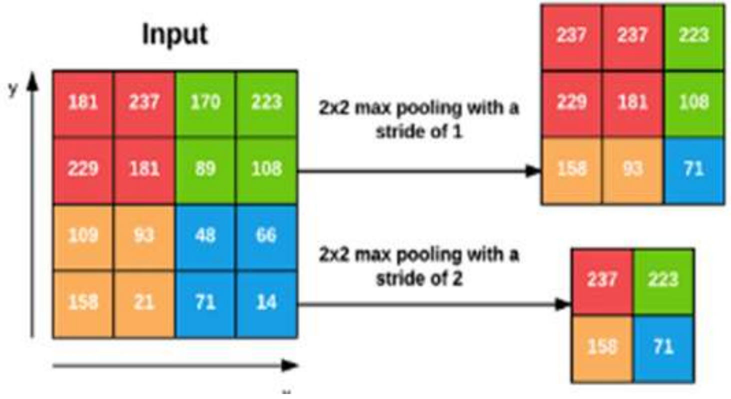
Max pooling

If any of the patches say something **firmly** about the **presence of a particular feature**, then the pooling layer counts that feature as ‘**detected**’.

Average pooling

If one patch says something very firmly, but the other ones **disagree**, the average pooling takes the average to find out.


1 2 5 23
7 5 3 11
18 21 9 2
4 3 56 0
Max Pooling
7 23
21 56
We take the largest of the pixel values of a segment.
Avg Pooling
4 11
12 17
We take the average of the pixel values of a segment.

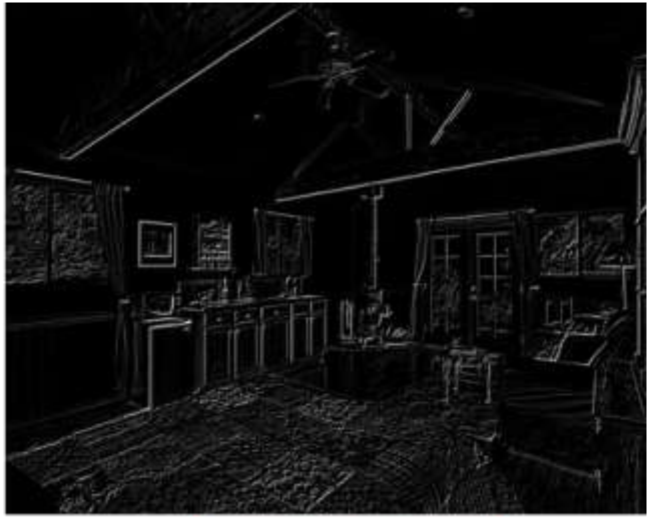


Understanding CNNs Contd…

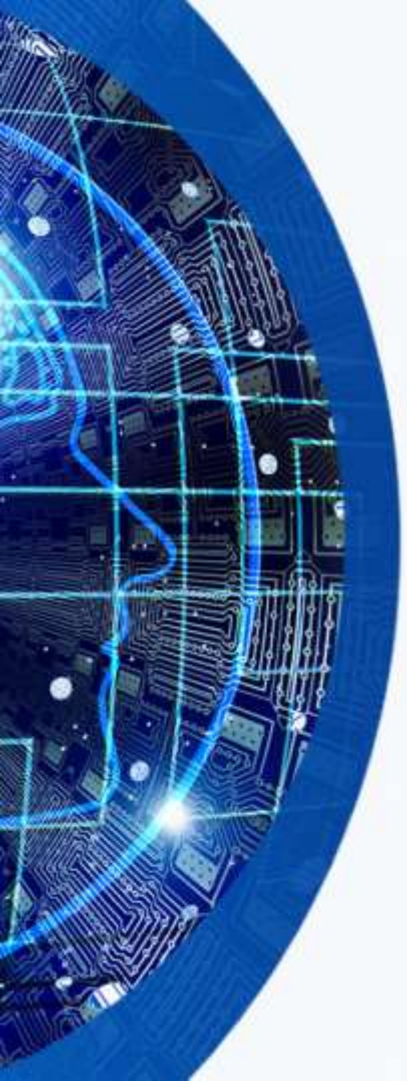
Eg: Convolution

Input Image

$\longrightarrow$



Convoluted Image



Training the CNN

Training CNNs

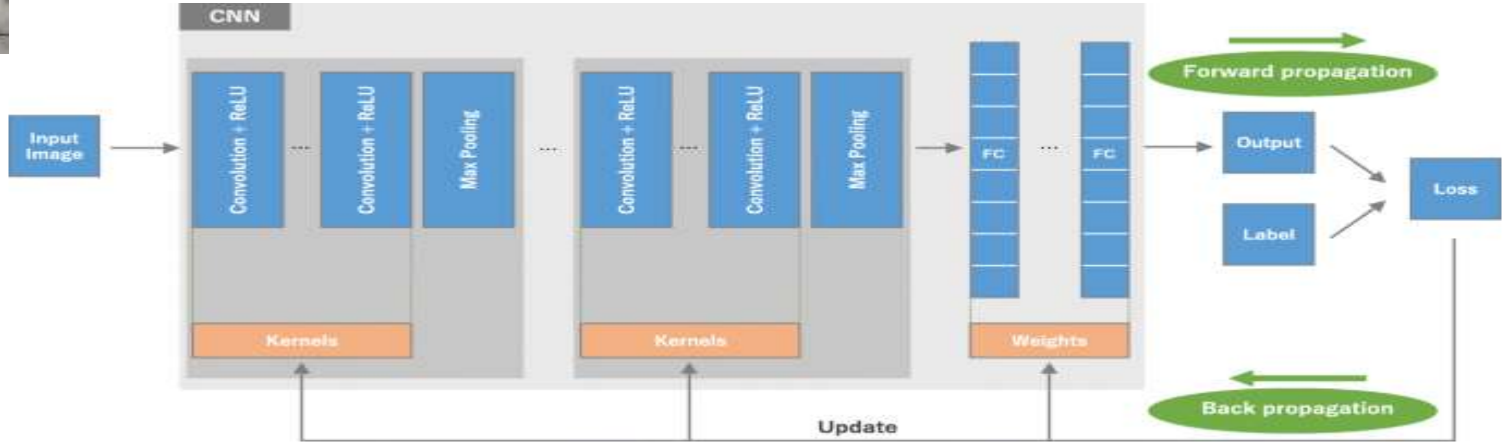
- The process of **adjusting the value of the weights / optimizing parameters such as kernels** is defined as **training** of the CNN.
- Firstly, the CNN **initiates** with the **random weights**.
- CNN is fed with training **dataset** with their corresponding **class labels.**

Steps

- ❑ Forward Propagation – Predicts the output using input image
- ❑ Back Propagation – Weight adjustment wrt loss

- This process of forward and backpropagation is **repeated until** the **loss value drops** below a previously defined value.
- Evaluated using **test** dataset

Training CNNs Contd…



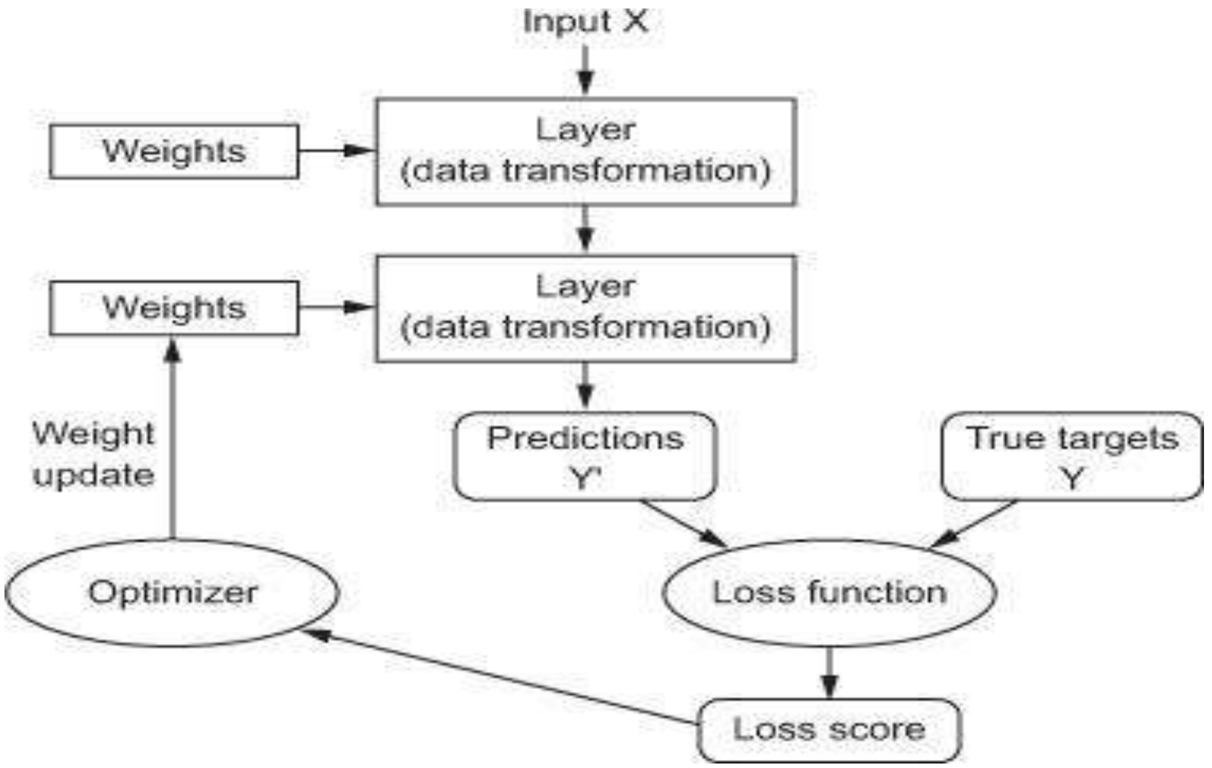
$$\overbrace{\text{Initialization->Forward Propagation->Computing Loss->Update Parameter => } \textbf{1 Epoch}}$$

Last Layer Activation Function

- ➢ The activation function applied to the last **FC** layer is usually **different** from the others.
- ➢ For a multiclass classification task a **Softmax function** is used.
- ➢ It normalizes **output** real values from the last FC layer to **target class probabilities**, where each value ranges between 0 and 1.

Training CNNs Contd…

Anatomy of a Deep Neural Network



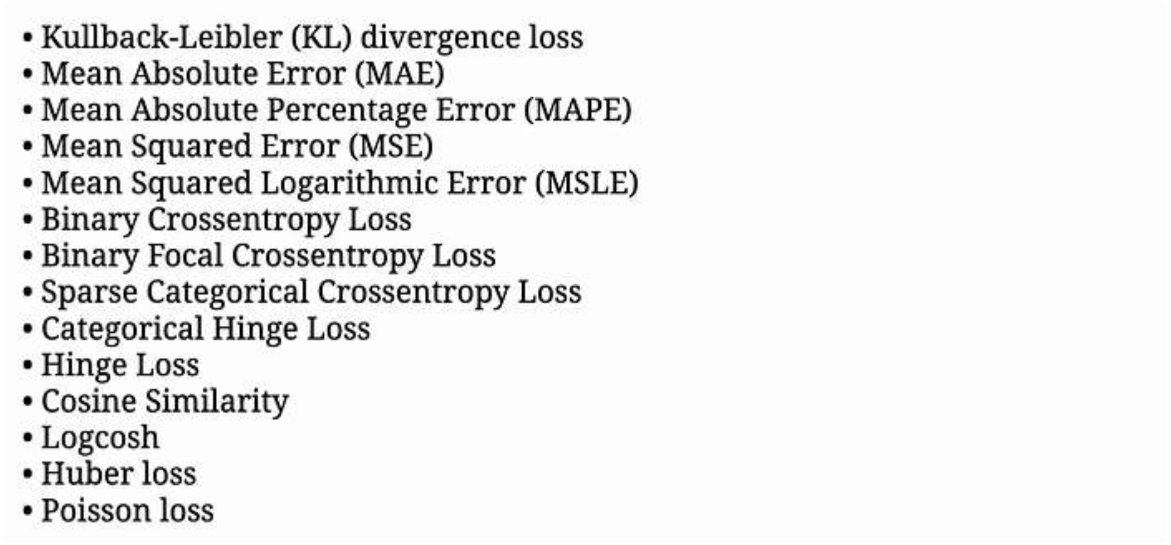
Training CNNs Contd…

Loss Functions

- A loss function, also referred to as a **cost function**, measures the **compatibility between output predictions** of the network through **forward propagation** and given **ground truth labels**.
- Aim is to **minimize** the loss
- Commonly used **types of** loss function
  - ✓ **Regression Loss Functions**
    - ❑ Mean Squared Error, Mean Absolute Error
  - ✓ **Classification Loss Functions**
    - ❑ Binary Cross-Entropy, Categorical Cross-Entropy

Training CNNs Contd…

In Tensorflow, the loss functions can be imported as function objects from the **tf.keras.losses** module. This module contains several built-in loss functions:



- Kullback-Leibler (KL) divergence loss
- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Mean Squared Error (MSE)
- Mean Squared Logarithmic Error (MSLE)
- Binary Crossentropy Loss
- Binary Focal Crossentropy Loss
- Sparse Categorical Crossentropy Loss
- Categorical Hinge Loss
- Hinge Loss
- Cosine Similarity
- Logcosh
- Huber loss
- Poisson loss

We can write **our own custom loss functions** to suit specific conditions.

Training CNNs Contd…

Optimization

- ➢ Most machine learning and deep learning algorithms involve some sort of **optimization**.
- ➢ Optimization refers to the process that **iteratively updates** the **learnable parameters**, i.e., **kernels** and **weights**, of the network so as to **minimize the loss**.
    - ✓ Gradient Descent

Training CNNs Contd…

Gradient Descent Optimization

- Most **basic** but **most used** optimization algorithm.
- **First-order** optimization algorithm which is dependent on the first order derivative of a **loss function**.
- It calculates that **which way the weights should be altered** so that the function can reach a **minima**.
    - ✓ gradient tells us **direction of greatest increase**, negative gradient gives us direction of **greatest decrease**

Training CNNs Contd…

Gradient Descent Optimization

Steps

- ✓ **Compute the gradient** (slope), the first order derivative of the function at that point.
- ✓ **Make a step (move) in the direction opposite to the gradient**, opposite direction of slope increase from the current point by alpha times the gradient at that point.
  - ❑ Step size - **Learning Rate**

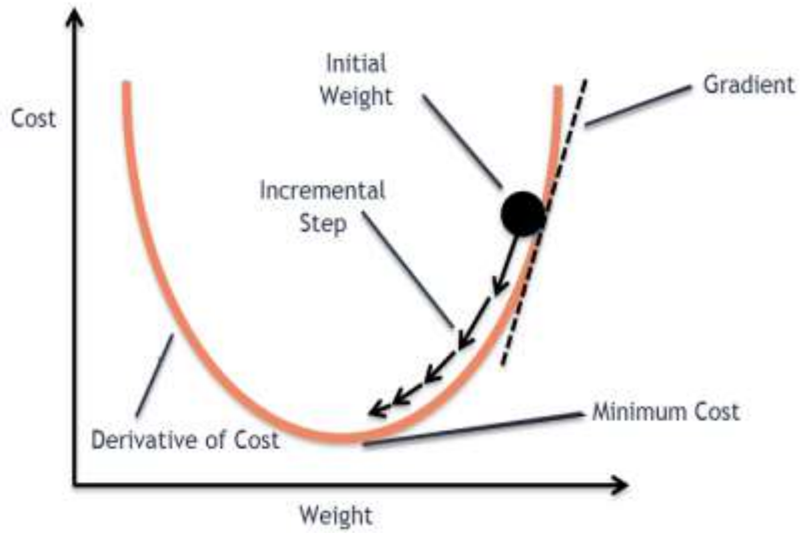
- pick a starting point (w)
- repeat until loss doesn't decrease in all dimensions:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_j = w_j - h\frac{d}{dw_j}loss(w)$$

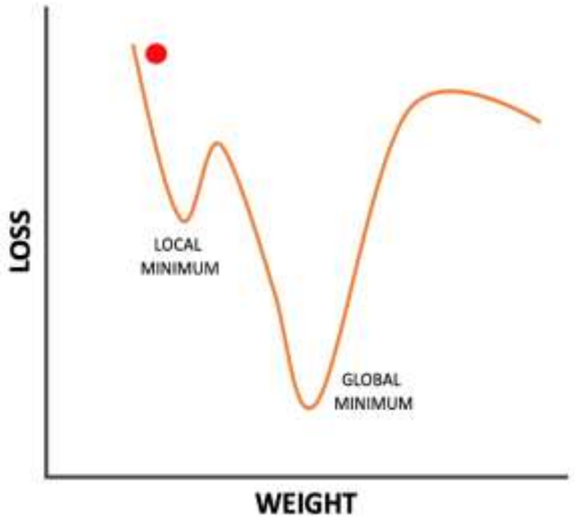
Training CNNs Contd…

Gradient Descent Disadvantages

- Due to **non convex nature of DL** we may trap at **local minima**.



Convex Function



Non Convex Function

Training CNNs Contd…

- How can we **avoid local minima** and always try and get the **optimized weights based on global minima**?
- Several **variants** of gradient descent optimization algorithms are available.
  - ✓ Stochastic Gradient Descent (SGD)
  - ✓ Batch Gradient Descent
  - ✓ Mini-batch Gradient Descent
  - ✓ Momentum-based Gradient Descent
  - ✓ RMSprop
  - ✓ Adam
  - ✓ Adagrad

Training CNNs Contd…

Stochastic Gradient Descent

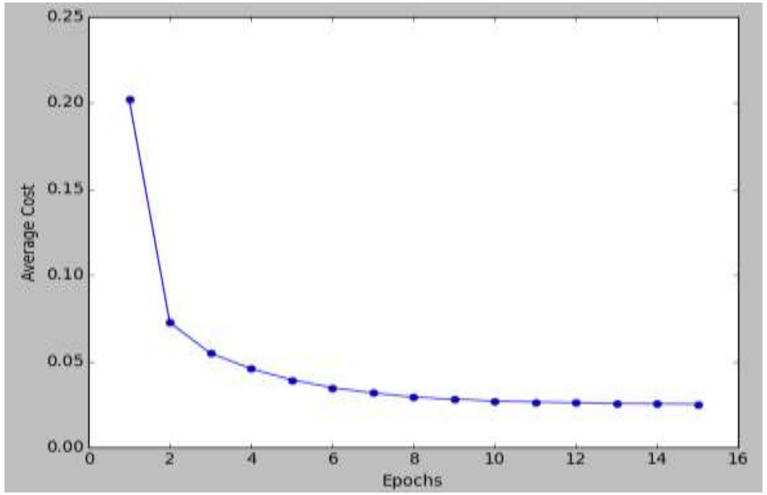
- It addresses the **computational inefficiency** of traditional Gradient Descent methods when dealing with **large datasets**
- In SGD, instead of using the entire dataset for each iteration, only a **single random training sample** is selected to calculate the gradient and update the model parameters wrt **learning rate**.
- **Shuffle** the training dataset to introduce **randomness**.

  For Eg: if the dataset contains **1000 rows** SGD will update the model parameters for **random samples** in one cycle.
- Converges in **less time**.
- Requires **less memory** as no need to store values of loss functions.

Training CNNs Contd…

Batch Gradient Descent

- Weights are changed after calculating gradient on the **whole dataset**.
- We take the **average of the gradients** of all the training samples and then use that mean gradient to update the parameters.
- So that's just one step of gradient descent in one **epoch**.
- If the dataset is **too large** then this may take **months to converge** to the minima.
- Requires **large memory** to calculate gradient on the whole dataset.



The cost keeps on decreasing over the epochs.

Training CNNs Contd…

Mini Batch Gradient Descent

- Mini-batch gradient is a variation of stochastic gradient descent where instead of single training example, **mini-batch of samples** is used.
- Mini batch gradient descent is widely used and **converges faster** and is **more stable**.
- **Batch size** can vary depending on the dataset.
- As we take a batch with different samples, it **reduces the noise** which is **variance of the weight updates** and that helps to have a **more stable** converge faster.

Training CNNs Contd…

Mini-batch size

- If the mini-batch size = **m**
  - It is a **batch gradient descent** where all the training examples are used in each iteration. It takes too much time per iteration.
- If the mini-batch size = **1**
  - It is called **stochastic gradient descent**, where each training example is its own mini-batch.
  - Since in every iteration we are taking just a single example, it can become extremely noisy and takes much more time to reach the global minima.
- If the mini-batch size is between **1 to m**
  - It is **mini-batch gradient descent**. The size of the mini-batch should not be too large or too small.
  - Based on memory requirements of the GPU or CPU hardware like **32, 64, 128, 256**, and so on.

Training CNNs Contd…

Adam (Adaptive Moment Estimation) Optimization

- The **Adam** optimization algorithm is an extension to **stochastic gradient descent**.
- SGD maintains a **single learning** rate (termed alpha) for all weight updates and the learning rate does not change during training. But in Adam a learning rate is maintained for **each network weight**.
- The authors describe Adam as **combining the advantages of two other extensions** of stochastic gradient descent, such as:
  - **Adaptive Gradient Algorithm** (AdaGrad) - maintains a **per-parameter learning rate** that improves performance on problems with sparse gradients (e.g. **natural language** and **computer vision** problems).
  - **Root Mean Square Propagation** (RMSProp) that also maintains per-parameter learning rates that are adapted based on the **average of magnitudes of the gradients** for the weight (e.g. how quickly it is changing). This means the algorithm does well on **online and non-stationary** problems (e.g. noisy).

Training CNNs Contd…

All types of Gradient Descent have some challenges:

1. Learning rate

The learning rate is the **size of the step** Gradient Descent takes all the way until it reaches the global minimum, and it directly impacts the performance of the algorithm.

- ✓ **Learning rate is too big**

  With big steps, Gradient Descent may never even reach the minimum and converge.

- ✓ **Learning rate is too small**

  With tiny steps at a time, the algorithm will eventually converge, but it might take a long time to do so.

Training CNNs Contd…

2. Vanishing/Exploding Gradient Problem

- What happens to the magnitude of the gradients as we backpropagate through many layers during training?
  - If the weights are **small**, the gradients shrink exponentially- **Vanishing Gradient**.
  - If the weights are big the gradients grow **exponentially** – **Exploding Gradient**.
- Typical feed-forward neural nets can cope with these exponential effects because they only have a **few hidden layers**.
- The vanishing gradient problem can **hinder the training** of deep neural networks.
- It **slows down** the learning process, leads to **poor convergence**, and **prevents** the network from effectively **capturing complex patterns** in the data.

Training CNNs Contd…

Reason for Vanishing/Exploding Gradients

- Certain activation functions, like the logistic function (**sigmoid**), have a very **huge difference** between the **variance** of their **inputs** and the **outputs**.

- In simpler words, they **shrink and transform** a **larger input space** into a **smaller output space** that lies between the range of **[0,1]**.

Training CNNs Contd…

How to know if our model is suffering from the Exploding/Vanishing gradient problem?

- ➢ Calculate **loss** and if **it is consistent** during epochs that means- Vanishing Gradient Problem.
- ➢ Draw the graphs between **weights and epochs** and if it is **constant** that means **weight has not changed** and hence Vanishing gradient problem.
- ➢ Large **error gradients accumulate** resulting in very large updates to model weights and the **loss value oscillate** during training –Exploding gradient

Training CNNs Contd…

How to overcome the vanishing/exploding gradient problem?

- Proper weight initialization
- Using non-saturating activation functions
- Batch Normalization
- Gradient Clipping
- Using Residual Networks (ResNets)

Training CNNs Contd…

1.Weight Initialization Techniques

- **Zero Initialization -** highly **ineffective** as neurons learn the **same features** during each iteration.
- **Random Initialization -** assigns random values- Overfitting, Vanishing Gradient Problem, Exploding Gradient Problem might occur

✓ Weight should not be same
✓ Weight should have variance

Training CNNs Contd…

He Initialization

- Works with **ReLU**
- Generate random numbers between two values – **size of *l*th layer** and size of ***l*-1th layer** – with some added **variance**

$$\textit{np.random.randn(size\_l, size\_l-1)} \times \sqrt{\frac{2}{size\_l-1}}$$

Xavier/Glorot Initialization

- Works with **Tanh**

$$\textit{np.random.randn(size\_l, size\_l-1)} \times \sqrt{\frac{1}{size\_l-1}}$$

$$\textit{np.random.randn(size\_l, size\_l-1)} \times \sqrt{\frac{2}{size\_l-1+size\_l}}$$

Training CNNs Contd…

2. Using non-saturating activation functions

Sigmoid & hyperbolic tangent activation functions – **saturate - ** where the **gradients become close to zero** for **large or small** inputs

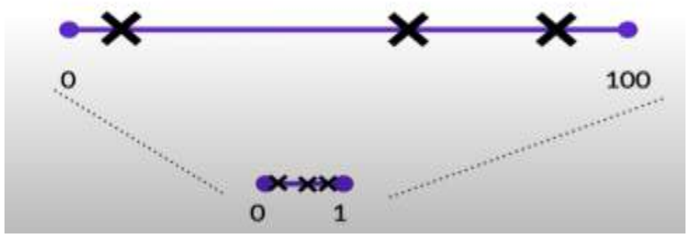
ReLU and its alternatives have a non-saturating activation function, which ensures that the gradients flow freely across the network.

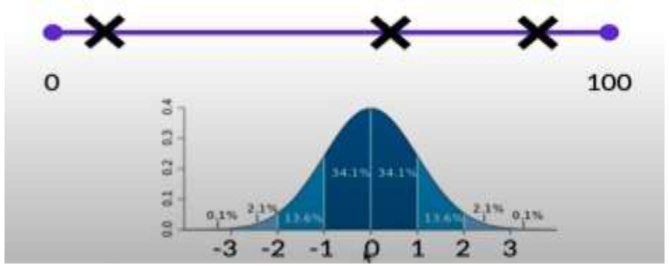
- ✓ Leaky ReLU (LReLU)
- ✓ Exponential Linear Unit (ELU)

Training CNNs Contd…

3. Batch Normalization

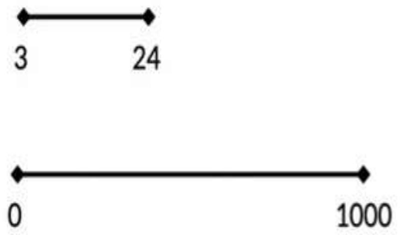
- ➢ **Normalization** – set the inputs to be between **0** and **1**
- ➢ **Standardization** – make mean **0** and variance (std-dev) **1** if put in a **distribution**
- ➢ Non normalized datasets cause **instability** to the model with
  - ✓ Imbalanced gradients
  - ✓ Reduced training speed

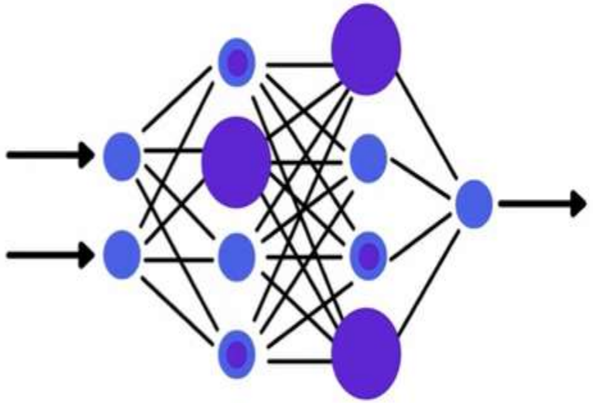




Training CNNs Contd…

Without Normalization





Vanishing/Exploding Gradients

Training CNNs Contd…

Batch normalization

- Batch normalization involves normalizing the output from activation function.

$$z = \frac{x - m}{s}$$

- Multiply the normalized output with some arbitrary parameter, g

$$z * g$$

- Add another arbitrary parameter, b

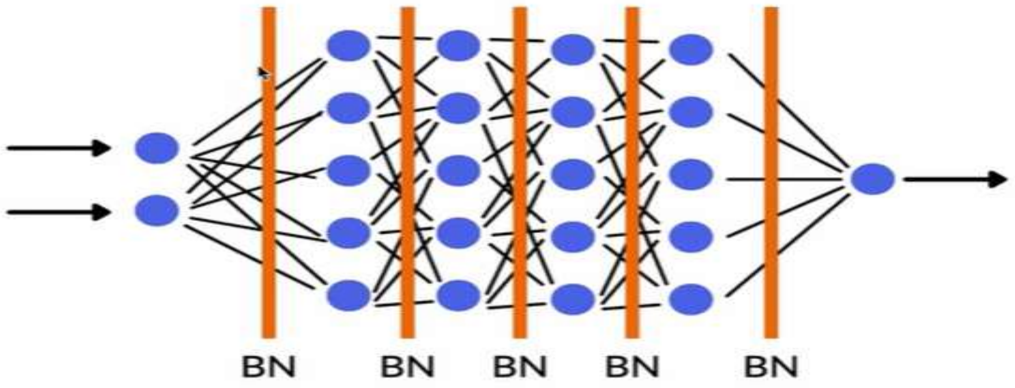
$$(z * g) + b$$

m - mean, s- std.dev

m, s, g & b are trainable and optimized during training

Training CNNs Contd…

- **Epochs** take longer time
- BN improves the **gradient flow** and allows **faster convergence**.
- BN is also **robust** to changes in hyperparameters and improves the stability of the training process.
- No need of separate standardization



Training CNNs Contd…

4. Gradient Clipping

Training CNNs Contd…

Overfitting and underfitting, generalization, regularization in DL

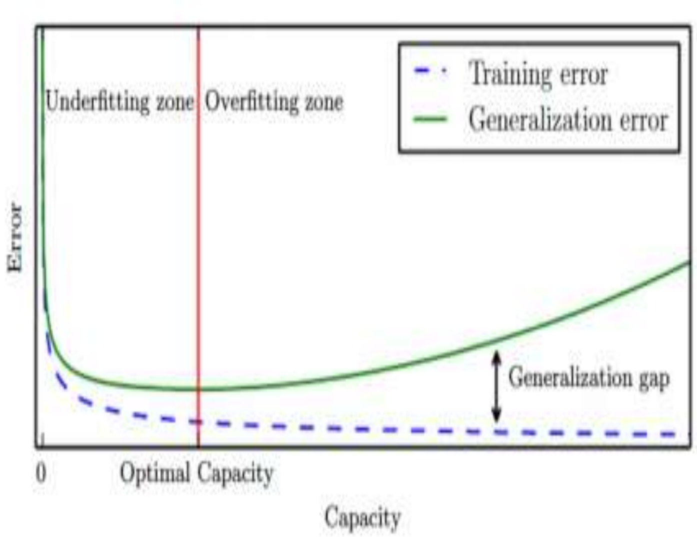
- When training a DL model, the model can be easily overfitted or under fitted.
- Models with **lots of parameters** can easily overfit
    - **Overfitting** is a phenomenon that occurs when a model is constrained to the training set and not able to perform well on unseen data.
    - **Underfitting** on the other hand is the case when our model is not able to learn even the basic patterns available in the dataset.

Training CNNs Contd…

Generalization: the quality of DL model is measured on new, unseen samples

Regularization:

- is a set of techniques used to ensure that a DL model can **generalize to new data** by **preventing overfitting**
- any modification to a **learning algorithm** to **reduce** its **generalization error** but not its **training error**
- reduce **generalization error** even at the expense of increasing **training error**



Training CNNs Contd…

The **regularizer** is a **penalty term** which depends on the hypothesis *h*

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing } \textit{too} \text{ well on training data}}$$

OR

$$\boxed{\sum_{i=1}^{n} \left(y_{act} - y_{pred}\right)^2 + \text{penalty}}$$

The most **common approaches** for regularization rely on statistical methods such as:

- ❑ Lasso regularization - Least Absolute Shrinkage and Selection Operator (L1 regularization)
- ❑ Ridge regularization (L2 regularization) and
- ❑ ElasticNet regularization, which combines both L1 and L2
- ❑ Dropout
- ❑ Data augmentation
- ❑ Early stopping

Training CNNs Contd…

L1 Regularization

- Introduce a **penalty term** into the model's **loss function** based on the **absolute values** of the model's parameters.
- Lasso **shrinks the less important feature’s coefficient to zero**; thus, removing some feature altogether.
- So, this works well for **feature selection** in case we have **high-dimensional data with huge number of features**.
- It enables one to choose a **subset of the most important attributes** and result in **sparse solutions** (i.e. lots of zero weights)
- More **robust to outliers**
- The **size of a penalty term** is controlled by a hyperparameter **lambda**, which regulates the L1 regularization's regularization strength.
- As **lambda rises**, more parameters will be **lowered to zero**, improving regularization.

$$\text{cost function} = \sum_{i=1}^{n} \left(y_{act} - y_{pred}\right)^2 + \lambda \cdot ||w||_1$$

Training CNNs Contd…

L2 Regularization

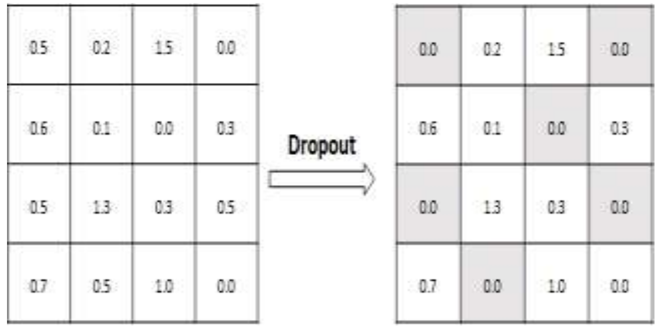
- Unlike L1 regularization, L2 regularization **does not induce sparsity**.
- Instead, it **shrinks** the weights **towards zero** without setting them exactly to zero.
- This results in a model that **considers all features** but reduces their overall impact on the final prediction.
- Ridge is **not robust to outliers** as square terms blow up the error differences of the outliers

$$\text{cost function} = \sum_{i=1}^{n} \left(y_{act} - y_{pred}\right)^2 + \lambda \cdot ||w||_2^2$$

Training CNNs Contd…

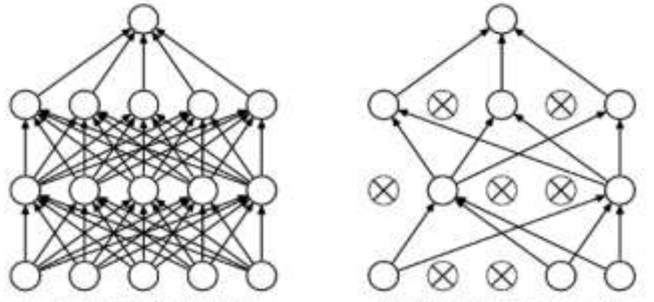
Dropout

- The dropout is a recently introduced technique used for regularization to reduce overfitting.
- Dropout can be employed in convolutional layers, POOL layers, or FC layers.
- Here, during training time, at each iteration, a neuron is **temporarily dropped out** or **disabled** with a **probability '*p*'** known as the **dropout-rate** hyperparameter.



0.5	0.2	1.5	0.0
0.6	0.1	0.0	0.3
0.5	1.3	0.3	0.5
0.7	0.5	1.0	0.0

0.0	0.2	1.5	0.0
0.6	0.1	0.0	0.3
0.0	1.3	0.3	0.0
0.7	0.0	1.0	0.0

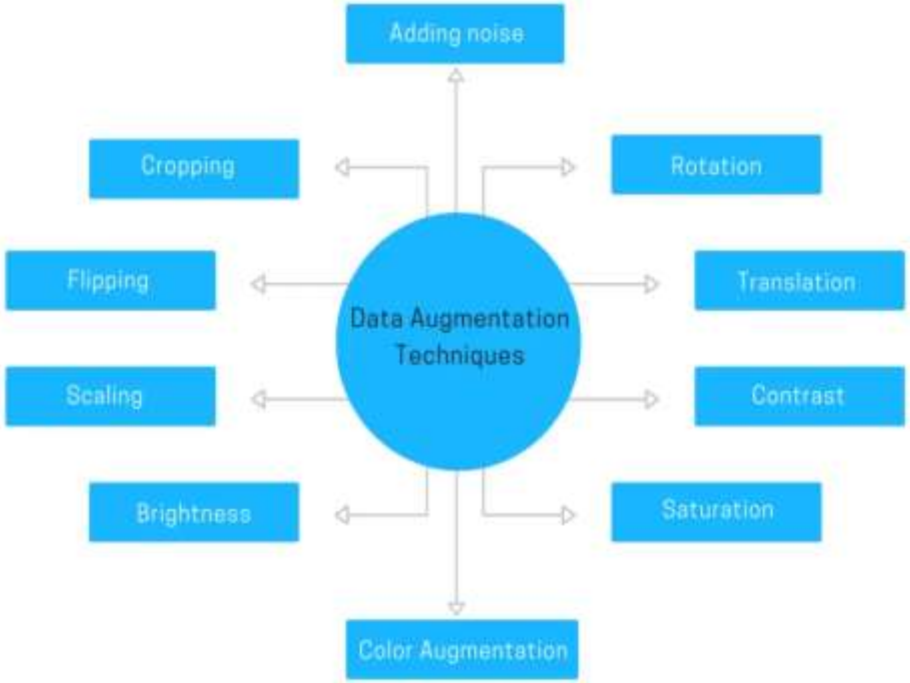


(a) Standard Neural Net (b) After applying dropout.

Training CNNs Contd…

Data Augmentation

- The best way to make a machine learning model **generalize better** is to **train it on more data**.



Training CNNs Contd…

- One must be careful **not to apply transformations** that would **change the correct class**.

(For example, optical character recognition tasks require recognizing the difference between "b" and "d" and the difference between "6" and "9," so horizontal flips and 180° rotations are not appropriate ways of augmenting).

Text Data

- ✓ Synonym replacement
- ✓ Text Substitution (rule-based, ML-based, mask-based and etc.)
- ✓ Random insertion
- ✓ Random swap
- ✓ Random deletion
- ✓ Word & sentence shuffling

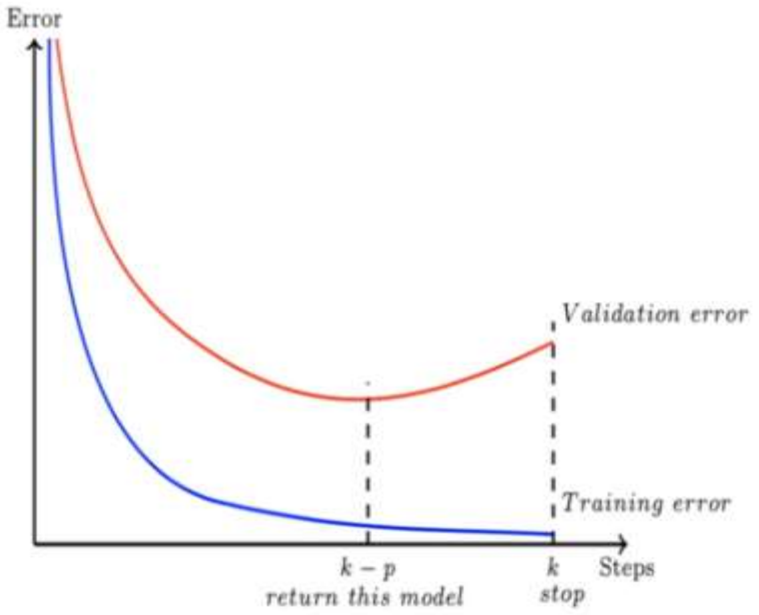
Audio data

- ✓ Cropping out a portion of data
- ✓ Noise injection,
- ✓ Shifting time,
- ✓ Speed tuning changing pitch,
- ✓ Mixing background noise and masking frequency

Training CNNs Contd…

Early Stopping

- When training models with sufficient representational capacity to overfit the task, we often observe that **training error decreases** steadily over time, while the error on the **validation set begins** to rise again or remaining the same for certain iterations, then there is no point in training the model further.
- This means we can obtain a model with better validation set error (and thus, hopefully better test set error) by **returning to the parameter** setting at the point in time with the **lowest validation set error**
- **Stop training when generalization error increases**

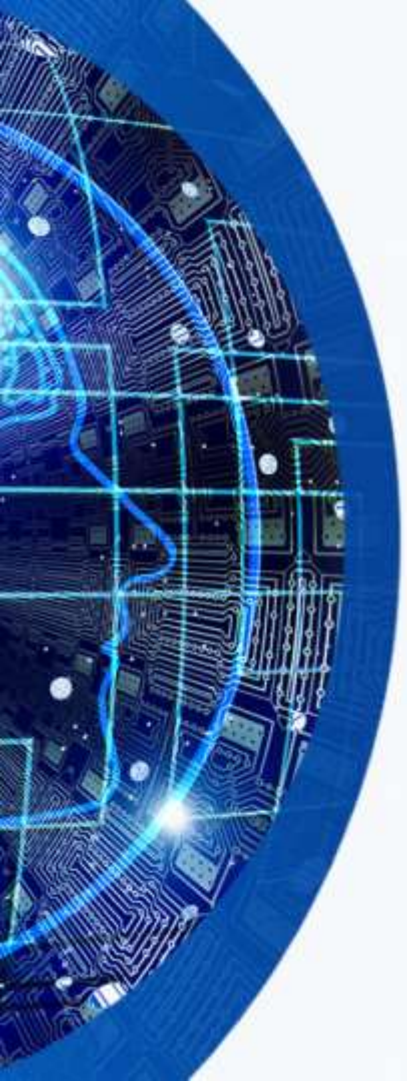


Training CNNs Contd…

How to Stop Training Early

There are three elements to using early stopping; they are:

- ➢ Monitoring model performance.
- ➢ Trigger to stop training.
  - ✓ Once a scheme for evaluating the model is selected, a trigger for stopping the training process must be chosen.
- ➢ The choice of model to use.



End of DL Part 1