Computer Vision and Image Processing - Fundamentals and Applications

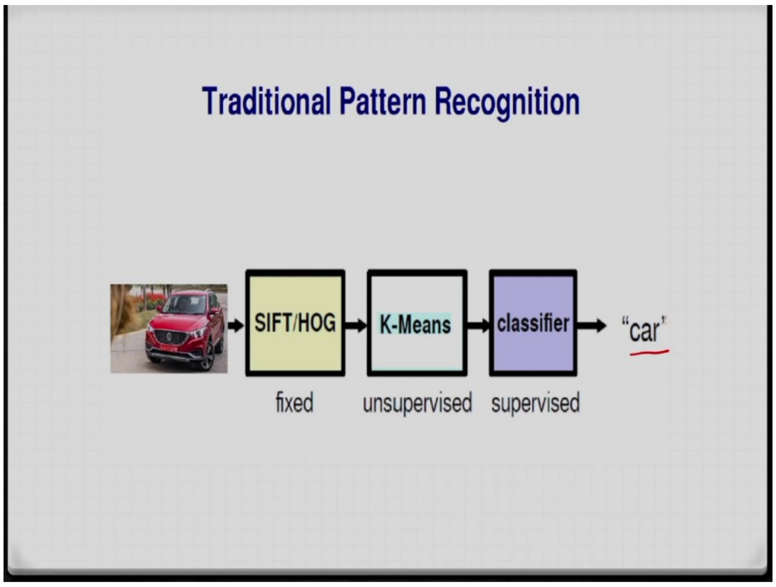
Professor Doctor M.K Bhuyan
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

Lecture - 37

Introduction to Deep Learning

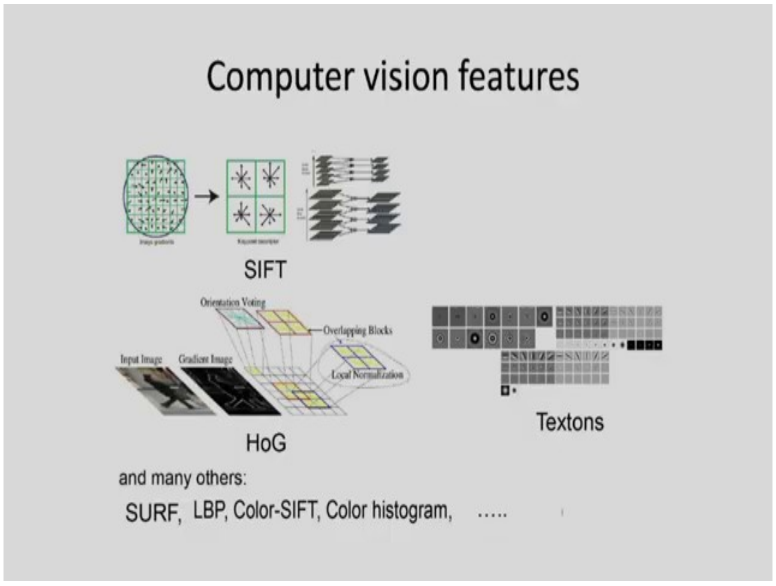
Welcome to NPTEL MOOCS course on Computer Vision and Image Processing, Fundamentals and Applications. I have been discussing the concept of artificial neural network. Today I am going to discuss the concept of deep learning techniques, it would not be possible to discuss all the deep learning techniques in this computer vision course. So, that is why I will briefly introduce the concept of deep learning, mainly the convolution neural networks and also auto encoders. How is deep learning different from the traditional pattern recognition system that I am going to explain. So, let us see the difference between the traditional pattern classification system and the deep learning techniques.

(Refer Slide Time: 01:25)



So, you can see here in this block diagram, I have shown 1 traditional pattern recognition system, you can see the input is the image 1 image I am considering and for classification or maybe for recognition of objects, I have to extract features from the image. So, in this case I am considering the features maybe SIFT features or maybe the HOG features I can consider and after this I can apply some unsupervised or supervised classification techniques for object recognition. So that means, the object is a car that is available in the image that I can recognize, this is the traditional pattern recognition system.

(Refer Slide Time: 2:11)



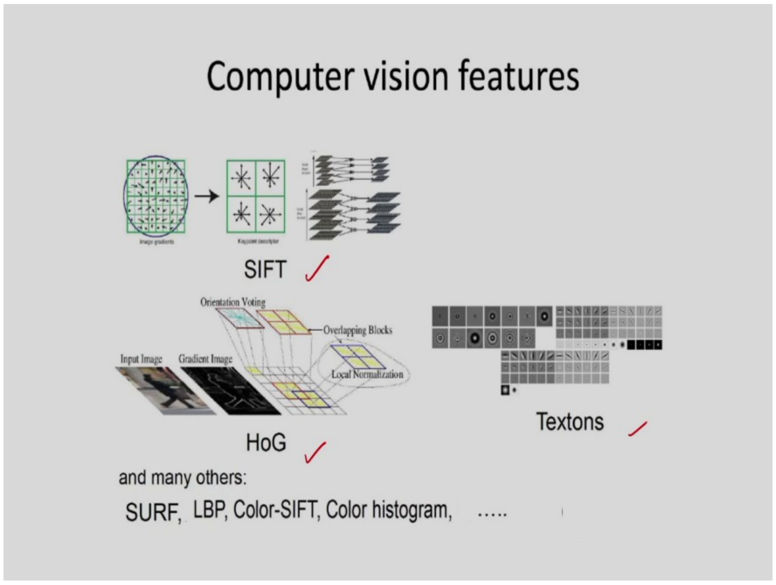
Computer vision features

and many others:

SURF, LBP, Color-SIFT, Color histogram, ...

And already I have discussed the image features. So, we can consider SIFT features or maybe you can consider HOG features, or maybe we can consider Textons features or maybe the features like SURF, the local binary pattern, the Colour-SIFT, Colour histograms so all these features we can consider for image classification, object recognition.

(Refer Slide Time: 2:36)



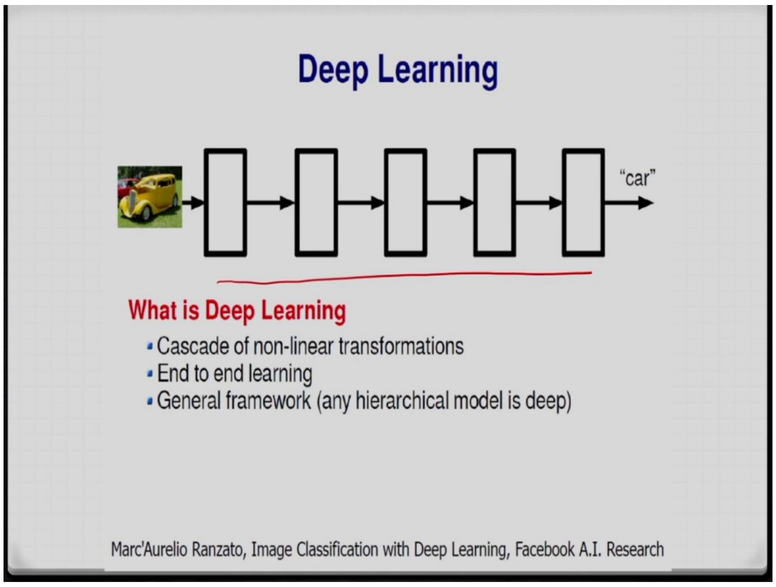
Computer vision features

and many others:

SURF, LBP, Color-SIFT, Color histogram, ...

So, in this case also I have shown 1 traditional pattern recognition system. So, you can see the first 1 is the input, input is the image, after this we can extract the features, these are the handcrafted features. So, maybe the features, the low-level features, the edges, we can consider the boundaries, we can consider the sift, we can consider HOG features, we can consider. So these features we can consider for image recognition, image classification, object recognition and after this we can apply the pattern recognition algorithms like support vector machine or maybe some other supervised or unsupervised techniques for object detection object classification. So, this is a typical traditional pattern recognition system.

(Refer Slide Time: 3:29)


Deep Learning

What is Deep Learning

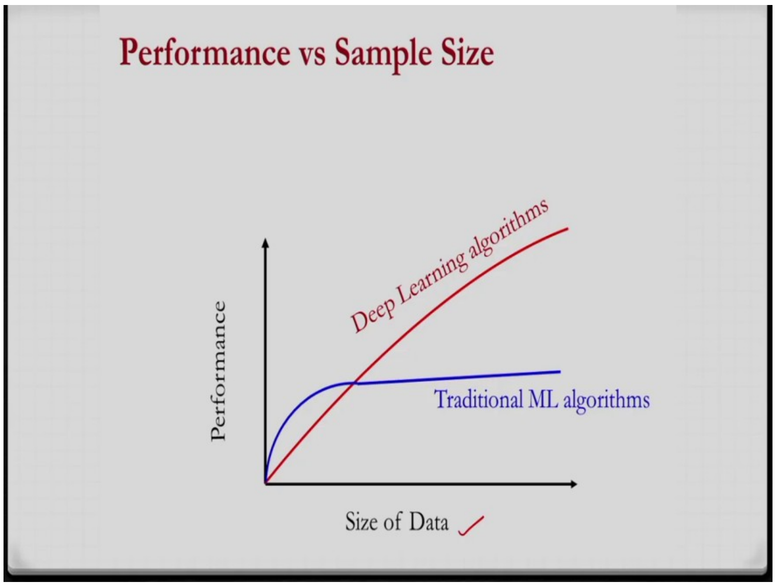
- Cascade of non-linear transformations
- End to end learning
- General framework (any hierarchical model is deep)

Marc'Aurelio Ranzato, Image Classification with Deep Learning, Facebook A.I. Research

And now, this is the structure of the deep learning, in this case you can see the deep learning is nothing but that cascade of nonlinear transformations. So, you can see, here is a cascade of nonlinear transformations and it is nothing but end to end learning. And in this case, we consider hierarchical model that we consider. In this case why it is called a deep because we want to extract more and more information from the input data.

So, suppose in this case I am considering 1 image. So, I want to extract more and more information from the image, maybe some low-level information we can consider or maybe the mid level information or the high-level information that is the features we can consider for image recognition for image classification. So, that is why it is called a deep learning approach because I want to consider more and more information from the input image and it is the end-to-end learning.

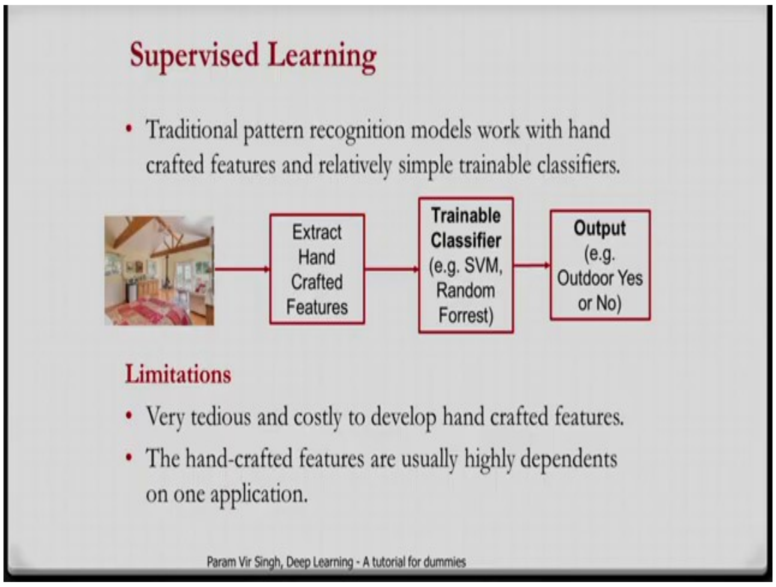
(Refer Slide Time: 4:34)



Performance vs Sample Size

And if you can see the performance of the deep learning algorithms and the traditional machine learning algorithms. Here you can see that we are considering the size of the training data, we are considering the size of the training data and also, we are considering the performance. If we have a large amount of training data, then you can see the performance of the deep learning algorithms are much better as compared to traditional machine learning algorithms, but if I consider a small amount of training samples are available suppose, then the traditional machine learning algorithms will be sufficient for a particular pattern classification problem or maybe the pattern recognition problem. So, you can see the performance versus sample size, sample size means number of training samples available for a particular classification problem.

(Refer Slide Time: 5:30)



Supervised Learning

- Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.

Extract Hand Crafted Features
Trainable Classifier (e.g. SVM, Random Forrest)
Output (e.g. Outdoor Yes or No)

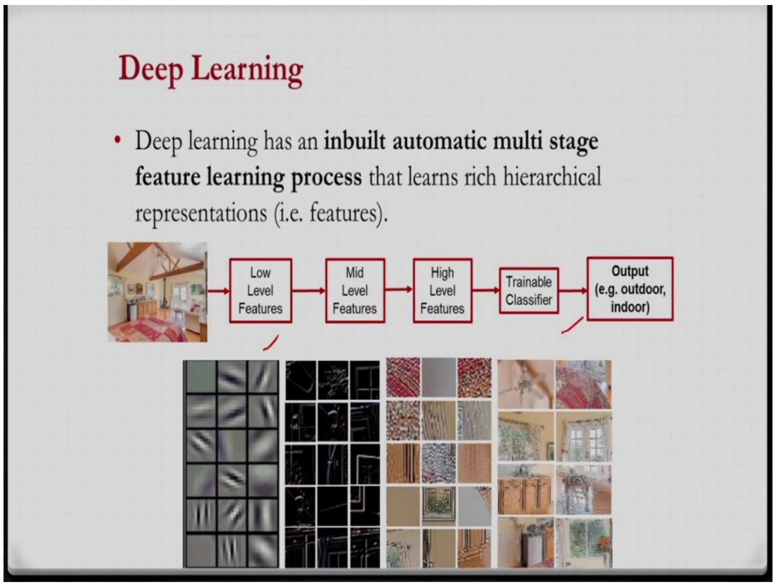
Limitations

- Very tedious and costly to develop hand crafted features.
- The hand-crafted features are usually highly dependents on one application.

Param Vir Singh, Deep Learning - A tutorial for dummies

So now, I am considering 1 supervised learning technique that is a traditional pattern recognition model I am considering. So, input image I am considering, after this we have to extract encrypted features and after this we are considering the classifiers maybe something like the support vector machine random forest. So, these types of classifiers we can consider and output is object recognition. So, outdoor scene that means, yes, if it is not the outdoor scene it is no, so that is the image classification. So, this is a block diagram for traditional pattern recognition system.

(Refer Slide Time: 6:10)

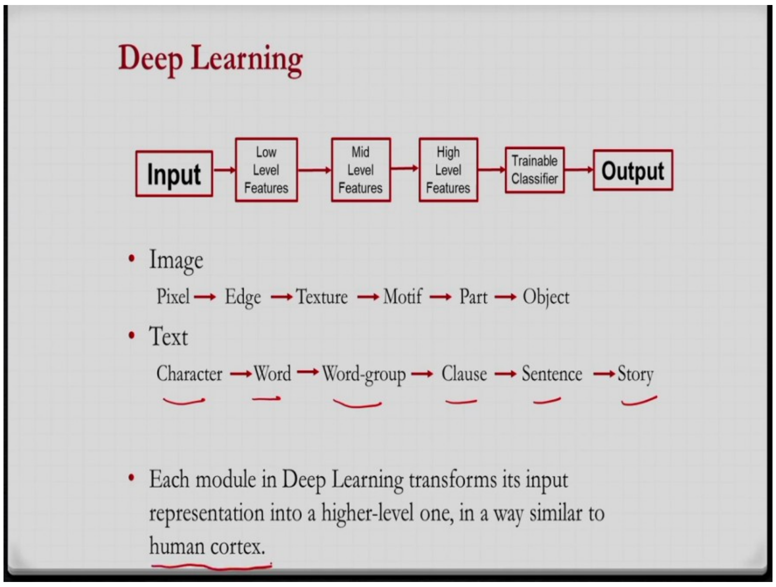


Deep Learning

- Deep learning has an **inbuilt automatic multi stage feature learning process** that learns rich hierarchical representations (i.e. features).

Now, I am considering that deep learning method. In this deep learning method already I have mentioned this is nothing but the cascade of nonlinear transformations, in this figure you can see I am considering 1 image and from this image I can extract some low level features or we can extract the mid level features and after this we can extract the high level features. So, maximum information I want to extract from the input image that is why it is called a deep learning then more and more information I want to extract from the input data. And after this, we can consider the classifier for object recognition for object classification or maybe the image recognition or image classification. So, you can see the distinction between the traditional pattern recognition system and the deep learning system.

(Refer Slide Time: 6:55)



Deep Learning

$$\begin{tikzcd}\boxed{\text{Input}} \arrow[r] & \boxed{\text{Low Level Features}} \arrow[r] & \boxed{\text{Mid Level Features}} \arrow[r] & \boxed{\text{High Level Features}} \arrow[r] & \boxed{\text{Trainable Classifier}} \arrow[r] & \boxed{\text{Output}}\end{tikzcd}$$

- Image

  Pixel → Edge → Texture → Motif → Part → Object

- Text

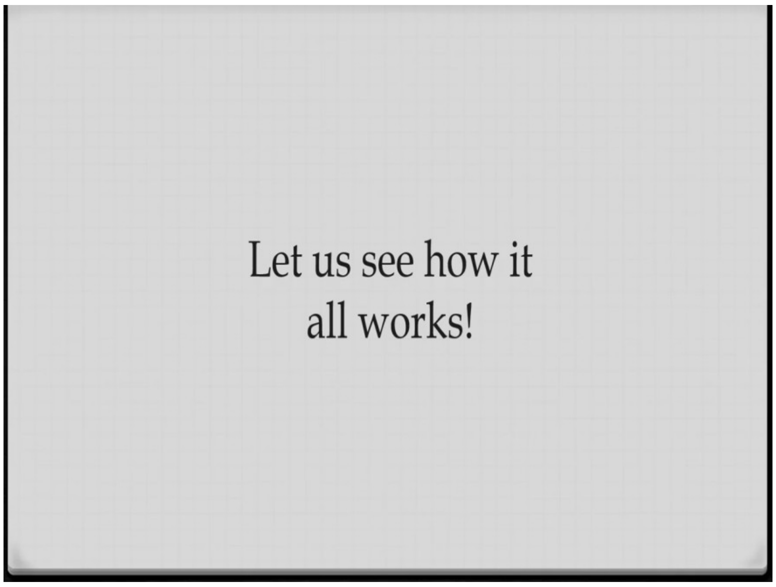
  Character → Word → Word-group → Clause → Sentence → Story

- Each module in Deep Learning transforms its input representation into a higher-level one, in a way similar to human cortex.

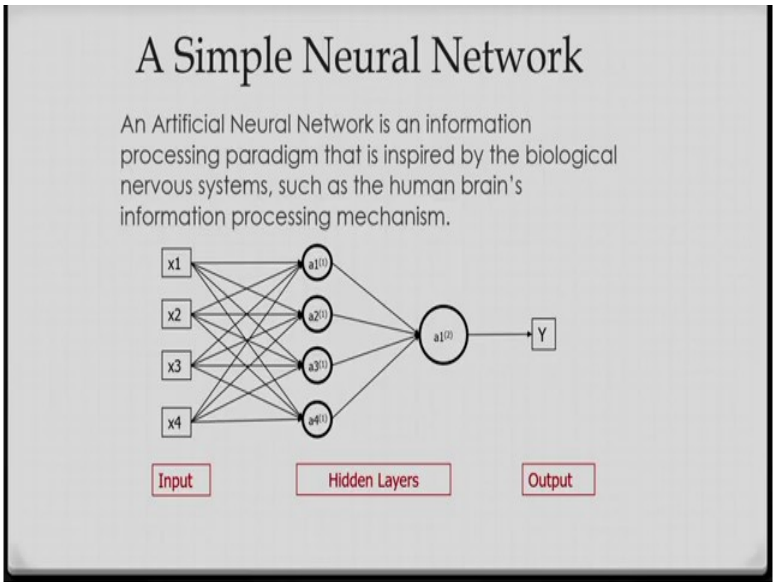
So, in the deep learning techniques, so already I have mentioned so we have to extract low level features, mid level features, high level features. So, in case of the emails if I consider, maybe the pixels values or maybe the edges, I can consider as the lower-level features. For the mid level features, we can consider the texture or maybe the motif and for the high-level features, we can consider the part the part of an object. So, these type of features we can consider as high-level features, that is the part of the object or maybe the object as a whole that we can consider for image recognition image classification.

In case of the text recognition, what we can consider, the low-level information, the low level features, maybe the characters, the word we can consider, for mid level features we can consider word-group or maybe the clause we can consider, and if I consider the high level features, we can consider sentence or maybe the story we can consider as high level features. And one important point is, if I consider this approach that is the deep learning approach, that is the hierarchical approach that is very similar to the human cortex. That means what we are doing, the deep learning transforms its input representation into a high level 1 and that is very similar to human cortex.

(Refer Slide Time: 8:33)



Let us see how it all works!



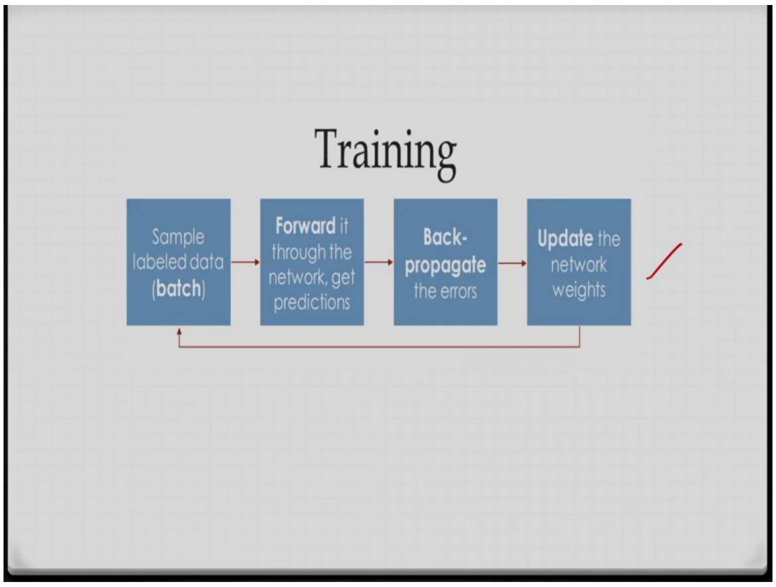
A Simple Neural Network

An Artificial Neural Network is an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.

Let us see the basic concept of the deep learning so, how it works. So first, I will explain the simple artificial neural networks and what are the main disadvantages of the artificial neural network and why we need deep learning systems, why we need deep learning techniques. So, in case of the artificial neural network, already I have explained that we have the inputs that is the input layers and we have the output, output is why I am showing here and also the hidden layers are available.

And you can see the nodes corresponding to the hidden layers, a 1, a 2, a 3, a 4, these are the nodes corresponding to the hidden layer. And if you see in the output, we have the layer the only 1 node is available. So, I am considering this artificial neural network. So, you can see the inputs x 1, x 2, x 3, x 4 these are the inputs. And corresponding to the hidden layer my nodes are a 1, a 2, a 3, a 4, and from a 1 2 I will be getting the output. So, this is a simple artificial neural network.

(Refer Slide Time: 9:58)


Training

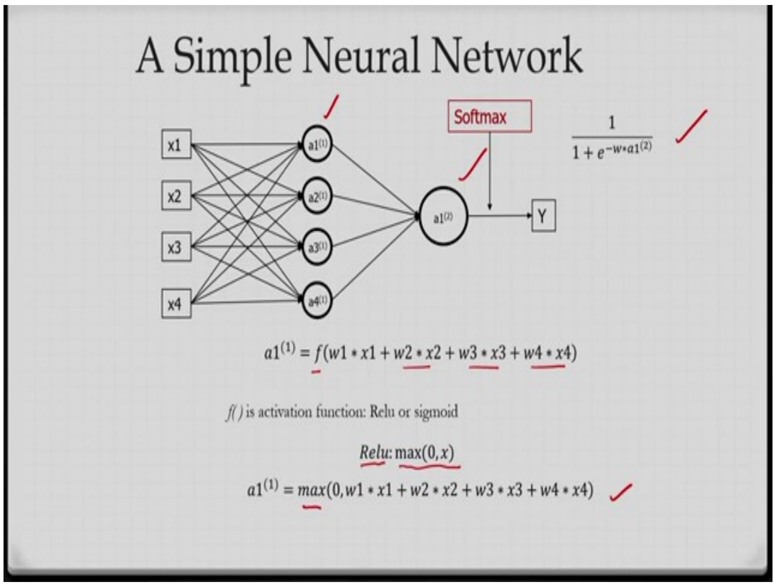
Sample labeled data (batch)
Forward it through the network, get predictions
Back-propagate the errors
Update the network weights

And the training procedure already I have explained in case of the artificial neural network and that is the supervised learning. So first, the sample level data we are considering so that is why it is a supervised learning, after this considering this sample level data we are uploading it through the network and get predictions, that is the meaning is I want to calculate the output of the artificial neural network.

And already we know the desired output. So, the difference between the desired output and the actual output that is the error. So, errors should be back propagated to the input, so that we can update the weights of the artificial neural networks. So, that back propagation technique already I have explained in artificial neural network. So, this is the concept of the training of artificial neural network.

So, we have to consider sample level data, after this we have to determine the output of the artificial neural network and we can determine the error the error is nothing but the desired output minus actual output and we have to minimize this error, so that the error is back propagated to the input and for this we have to update the weights of the artificial neural network that means, we are reducing the error in the output. And based on this we are adjusting the weights of the artificial neural network and that is the training procedure that is a supervised training procedure of the artificial neural network.

(Refer Slide Time: 11:20)



A Simple Neural Network

$$a1^{(1)} = f(w1*x1 + w2*x2 + w3*x3 + w4*x4)$$

$f()$ is activation function: Relu or sigmoid

Relu: $\max(0,x)$

$$a1^{(1)} = max(0, w1*x1 + w2*x2 + w3*x3 + w4*x4)$$

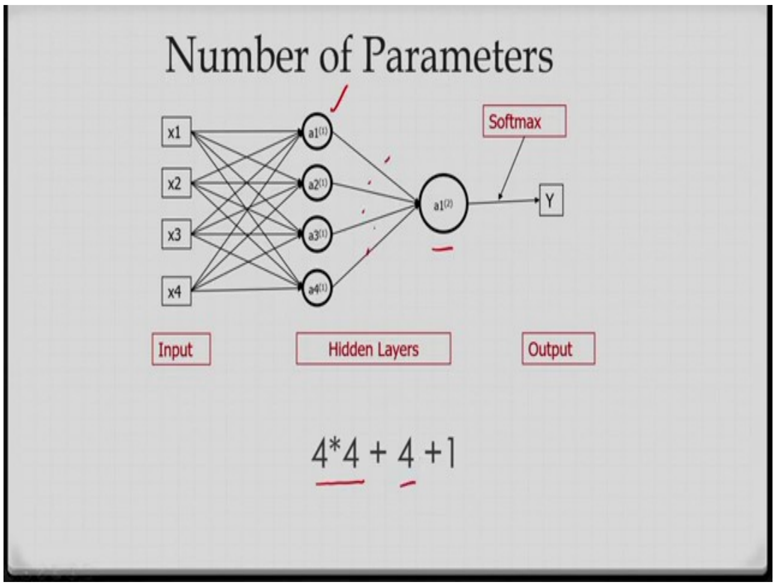
So, now I am considering a simple neural network here you can see. So, I am considering 4 inputs x 1, x 2, x 3, x 4 and one note I am considering a 1 that is corresponding to the hidden layer, a 1 one I am considering you can see, so my input is x 1, x 2, x 3, x 4 and weights are w 1, w 2, w 3, w 4. So, what will be the response in a 1 that is the neuron is a 1. So, in this case you can determine the response.

So, response will be you can see w 1 is multiplied with x 1, w 2 is multiplied with x 2, w 3 is multiplied with x 3, w 4 is multiplied which x 4 and after this I am considering the artificial function in my artificial neural network plus I explained the activation function that is a sigmoid function we have considered.

Now, I am considering another activation function that is called Relu rectified linear unit I am considering, and you can see this function that maximum between 0 and x it is considered. So, this concept of relu I will be discussing in this class. So, we can find the response of this. So, if I consider this Max function instead of the sigmoid function, I can determine the output in the neuron that is the response in the neuron a 1 1 I can determine that I can determine. Corresponding to the input, the input is x 1, x 2, x 3, x 4. And I am considering the connected weights are w 1, w 2, w 3, w 4.

Similarly, for all other hidden nodes a 2, a 3, a 4, I can determine the response and finally, I can determine the response in the output neuron that is a 1 2 I can also determine and finally, I can consider one function that is the softmax function I can consider and that is the function is something like this, the soft max function so it is an activation function. So, that scales numbers into probabilities. So, I will be getting the probabilities value, the probabilities lies between 0 and 1 and it is used for multi class classification. So, the soft max function is an activation function that scales numbers into probabilities. So, I will be getting the probabilities for each and every classes, if I consider multiple classes. Suppose for class 1, the probability is 0.2, for class 2 probability is 0.3 like this, I will be getting the probabilities.

(Refer Slide Time: 14:10)

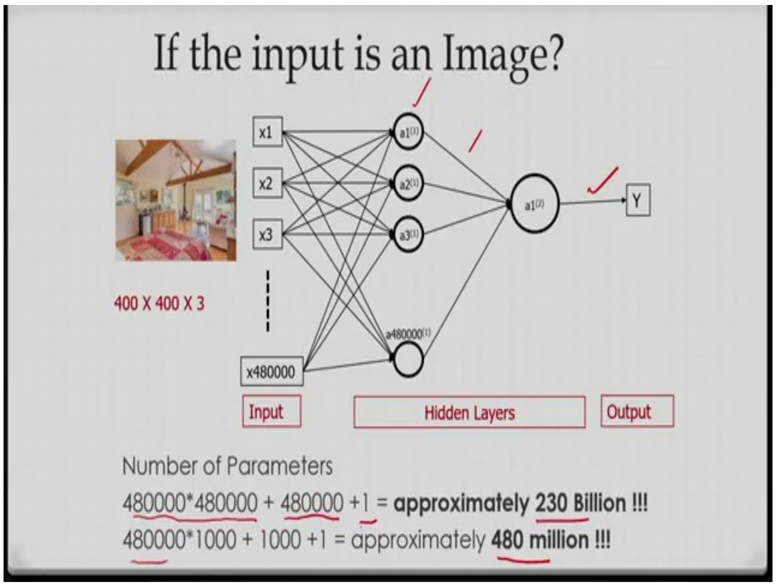


Number of Parameters

$4*4 + 4 + 1$

Now, in case of the artificial neural network if you see the number of parameters. So, you can see here, the input I am considering x 1, x 2, x 3, x 4 and that means, 4 input nodes I am considering and 4 hidden nodes I am considering, the 4 hidden nodes are a 1 1, a 2 1, a 3 1, a 4 1 that is a second layer I am considering. And you can see, corresponding to this, I have the 4 cross 4 connections, 4 into 4 connections. After this if I consider the second layer, the second layer is a 1 2. So, corresponding to the second layer, I have 4 connections, you can see the connections 1 2 3 4, four connections and finally I have the output for output I have 1 connection so that means, you can see how many connections I have in this simple artificial neural network.

(Refer Slide Time: 15:20)



If the input is an Image?

Number of Parameters

480000*480000 + 480000 +1 = **approximately 230 Billion !!!**

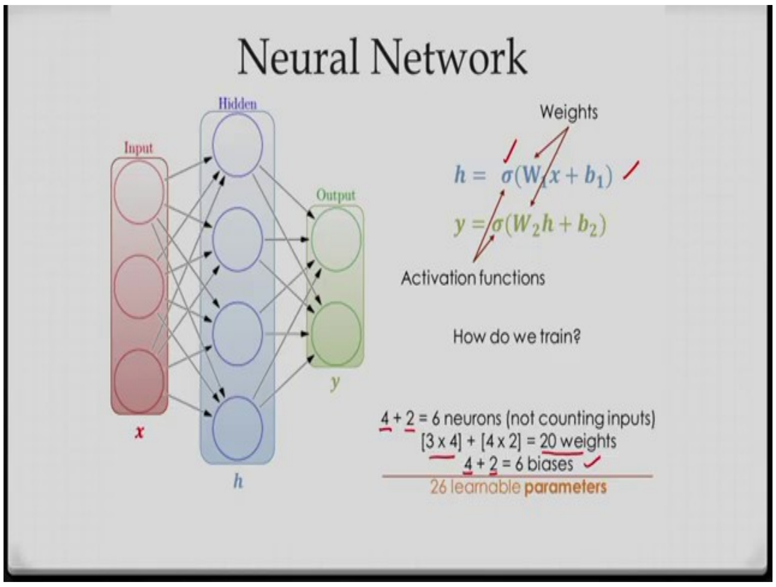
480000*1000 + 1000 +1 = approximately **480 million !!!**

And suppose if I consider 1 input image, the dimension is 400 × 400 × 3 that is 3 means, I am considering the RGB image. So, corresponding to this you can see, so number of parameters. So, you can see the serial number of parameters. So, this 400 × 400 × 3 that is nothing but 4,80,000 input nodes. So, how many nodes are available. So, x 1, x 2, x 3 up to x 4,80,000 number of nodes I am considering input nodes. After this I am considering the hidden layer, suppose the hidden layer also has 4,80,000 nodes, and if I consider this network, so number of parameters.

So, if you see the connections, the input connection and the first hidden layer connection, so 4,80,000 into 4,80,000 will be this and after this if I consider the second connection, so how σmany connections between the second layer and the third layer. So, it is again 4,80,000 plus 1 terminal that is the connection between the second hidden layer and the output so, 1 terminal. So, approximately it is 230 billion parameters, you can see. And if I consider suppose 1000 nodes corresponding to the first hidden layer so corresponding to this you can see we have 480 million parameters approximately, you can see the number of parameters in the artificial neural network.

So, it is too high so that is why it is very difficult to train an artificial neural network because we have to consider so many connections and so many weights we have to consider. So, that is why it is very difficult to consider or it is very difficult to train a particular artificial neural network for image classification for image recognition. The number of parameters are too high.

(Refer Slide Time: 17:35)


Neural Network

Weights

$h = \sigma(W_1 x + b_1)$

$y = \sigma(W_2 h + b_2)$

Activation functions

How do we train?

4 + 2 = 6 neurons (not counting inputs)

[3 x 4] + [4 x 2] = 20 weights

4 + 2 = 6 biases

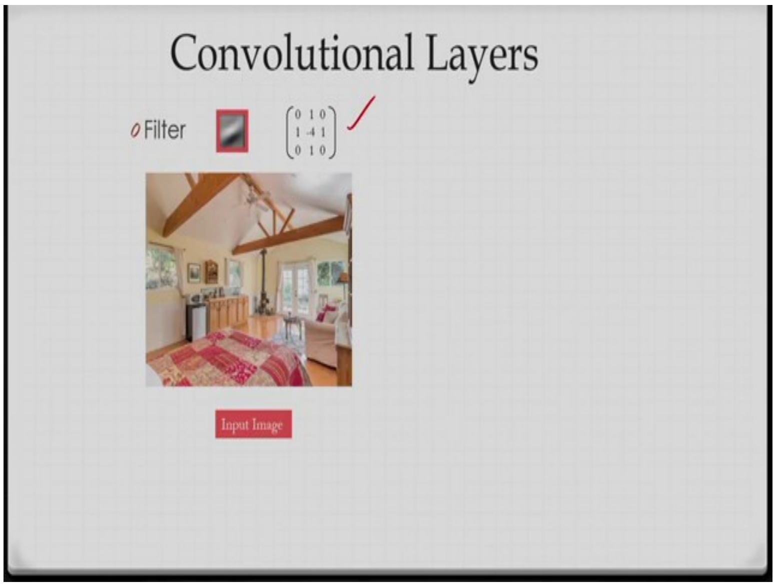
26 learnable parameters

Again, I am considering another example here you can see, I am considering input layer, hidden layer and the output layer. In the input layer we have 3 nodes that I am not considering now, and you can see in the hidden layer we have 4 nodes and in the output layer we have 2 nodes that means 2 neurons. So, corresponding to hidden layer if I consider sigma is the activation function so what will be the output in the hidden layer, it is nothing but σ ¿+ b_1¿. So, b_1 I am considering it is the biased terminal I am considering. And similarly, if I consider the response in the output nodes then in this case again I am considering the activation function sigma that is a sigmoid function I am considering.

So, $W_2 x$, W_2 is the weight plusb_2, b_2 is the bias I am considering. Now, in this case how do you train this artificial neural network. So, if I do not consider input nodes then how many neurons will be there. So, in the hidden layer we have 4 neurons and in the output, we have 2 neurons so total 6 neurons, and if you see the interconnections, that is the weight mainly the interconnections means the weights so how many weights? 3 into 4 corresponding to the connections between input and the hidden layer, and after this 4 into 2 that is the connection between the hidden layer and the output layer. So, total 20 widths we have to consider.

And if I consider the bias, suppose for the hidden layer we have 4 bias terminals. So, then in this case we will be getting 6 biases we are considering. So that means, we have 26 learnable parameters in this neural network that is too high. So, how to resolve this problem by considering the deep architecture the deep networks? So, for this I will discuss the concept of the convolution neural network. So, what is the convolution neural network and how the number of learnable parameters can be reduced.

(Refer Slide Time: 19:37)



Convolutional Layers

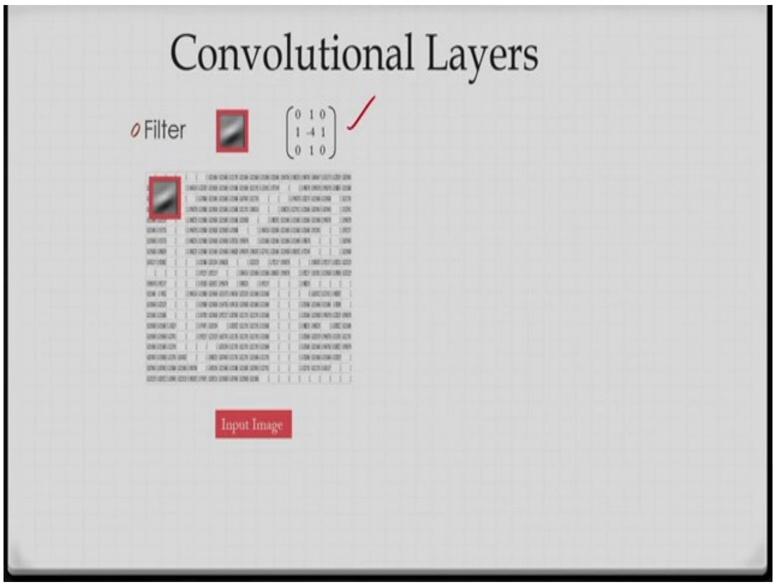
- Filter

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

So, for this I am considering one convolution layer, you can see I am considering the input image and I am considering one filter that is the convolution filter I am considering and here it is I am considering the Laplacian filter. So, what I can consider corresponding to this input image, I can determine the convolution between the filter and the image. So, I have the input image, I can determine the convolution between the input image and the filter.

So, for this I have to move the filter over the image and I have to determine the convolution outputs. So, for each and every pixels, I have to do the convolution. So, I have to place the max, I have to place the filter over the image and I have to determine the convolution. So, for all the pixels of the image I have to determine the convolution.

(Refer Slide Time: 20:34)



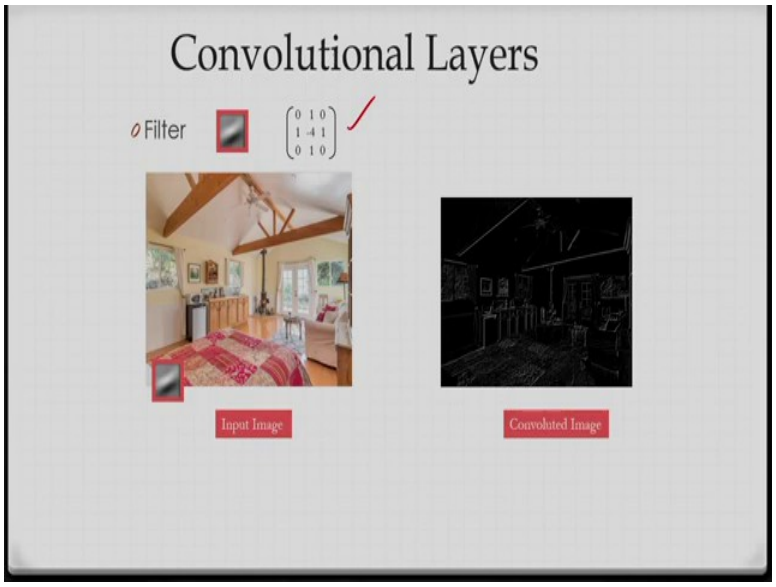
Convolutional Layers

Filter

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

So, you can see suppose this is the image, the pixels value I am considering and I have to place the marks over the image, I have to place the filter over the image and after this I have to determine the convolution and for all the pixels, I have to do the convolution. So I am showing it again so like this, you can see I am doing the convolutions.

(Refer Slide Time: 20:58)



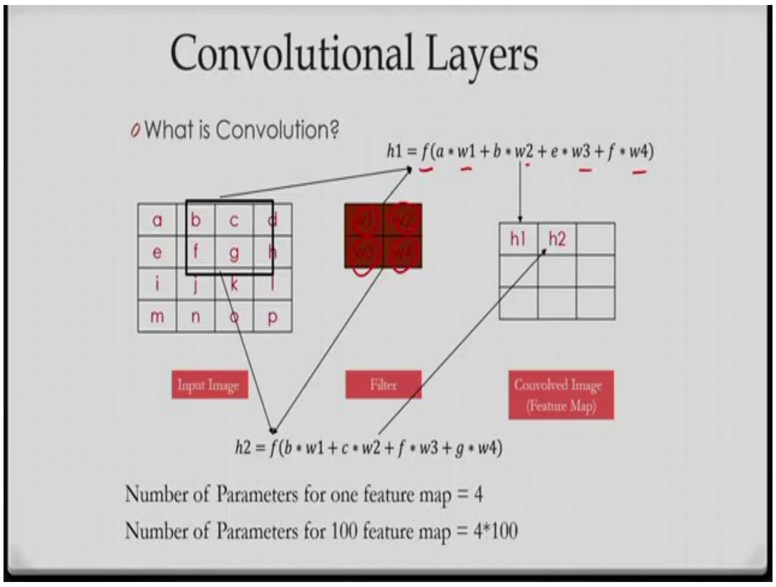
Convolutional Layers

Filter

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

And after this I am getting the convolve image since I am considering the high pass filter that is the Laplacian filter, I am considering so I will be getting the edges and the boundaries in the convolve image. So, this is my convolve image so in the convolve image, I will be getting the edges and the boundaries.

(Refer Slide Time: 21:17)



Convolutional Layers

- What is Convolution?

$h1 = f(a*w1 + b*w2 + e*w3 + f*w4)$

$h2 = f(b*w1 + c*w2 + f*w3 + g*w4)$

Number of Parameters for one feature map = 4

Number of Parameters for 100 feature map = 4*100

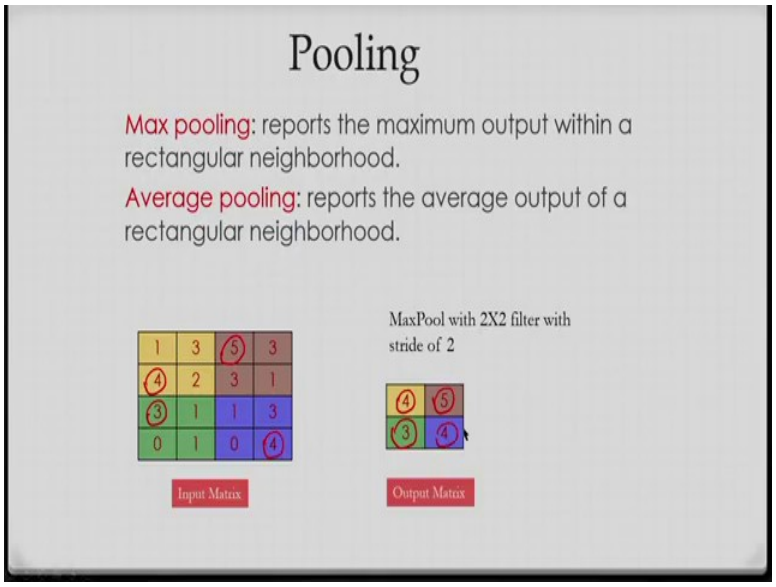
So, let us see what is convolution. So, suppose it is my input image I am considering so it is a 4 by 4 image a, b, c, d, e, f, g, h, i j, k l, m, n, o, p, these are the pixels I am considering, the pixel values are a b c d like this. And I am considering the filter, the filter weights are w 1, w 2, w 3, w 4. So, I am considering here w 1, w 2, w 3, w 4 like this. So, it is w 3 and w 4 so this filter I am considering, after this I am determining the convolution. So, how to determine the convolution you can see. So, I have to place the max over the image and I have to determine the convolution. So, just I am determining the convolution here you can see.

So, corresponding to that filter, I am just multiplying the pixels value with the mask value the mask means the filter value. So, weight is W 1 that is a into W 1 plus b into W 2 plus e into W 3 plus f into W 4 I am considering, and also I am considering the activation function, the activation function is f. So, corresponding to this I will be getting the output, the output is h 1 corresponding to the convolve image. And similarly, if I move the mask to the next pixel, then corresponding to this also I can determine the response of the mask. So, you can see I can determine h 2 just like the previous one so, I can determine h 2 and I will be getting the below h 2 here.

So, like this I have to do the convolution and I will be getting the feature map because I am extracting the features from the image. So, if I consider this is the filter and this filter can extract some important features from the image. So, that is why I am getting the feature map corresponding to their particular filter. And in this case, you can see the number of parameters for 1 feature map it will be 4, because I am only considering the filter weights W 1, W 2, W 3, W 4 so that is why the number of parameters for 1 feature map will be only 4, but just you can see, you can do the comparison between the simple artificial neural network and this case.

So, by using the convolution operation, I can reduce the number of parameters you can see the principle behind the convolution layers. So that means the number of parameters, the number of learnable parameters we can reduce. So, 4 number of parameters for 100 feature maps will be only 4 into 100. So, that is very least as compared to the earlier cases in case of the artificial neural networks.

(Refer Slide Time: 24:13)



Pooling

Max pooling: reports the maximum output within a rectangular neighborhood.

Average pooling: reports the average output of a rectangular neighborhood.

MaxPool with 2X2 filter with stride of 2

1	3	5	3
4	2	3	1
3	1	1	3
0	1	0	4

Input Matrix

4	5
3	4

Output Matrix

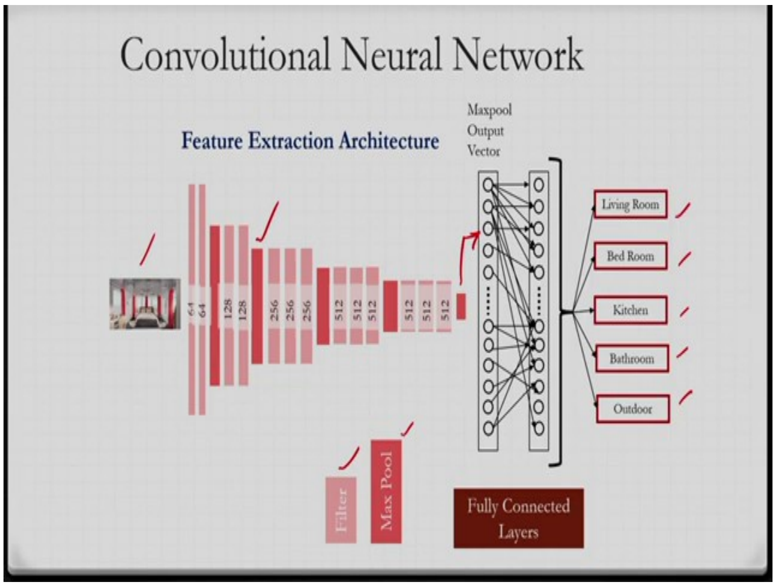
After this again I want to reduce the size of the feature map by the concept of the pooling. So, these 2 pooling techniques I can apply there are many techniques but mainly one is the max pooling, another one is the average pooling we can consider. So, suppose this is my input matrix and what is the max pooling, the maximum output within a rectangular neighbourhood.

So, suppose if I consider 2 × 2 neighbourhood, that means, 2 number of rows and 2 number of columns that is 2 × 2 neighbourhood I am considering. So, maximum output within a rectangular neighbourhood we have to determine. In case of the average pooling, what we have to do, average output of a rectangular neighbourhood we have to determine.

Now, I am only considering the max pooling. So, corresponding to this if you see here the corresponding to this input matrix, the output matrix will be 4 5 3 4 because corresponding to the person neighbourhood that is a 2 × 2 neighbourhood the maximum value is 4. So, that is why I am considering 4 here and corresponding to the second neighbourhood so if you see the maximum value, the maximum value is 5 so 5 I am considering here, corresponding to the third 2 ×2 neighbourhood I am considering the maximum value the maximum value is 3 so 3 I am considering here, and corresponding to the last 2 by 2 neighbourhood.

So, the maximum value is 4 so that is why I am considering 4 so that is the output matrix after max pooling. So, that means, I am reducing the size of the feature map. So, feature map is obtained by the process of the convolution and after this we can consider a pooling the max pooling we can consider. So that the size of the feature map can be reduced.

(Refer Slide Time: 26:08)

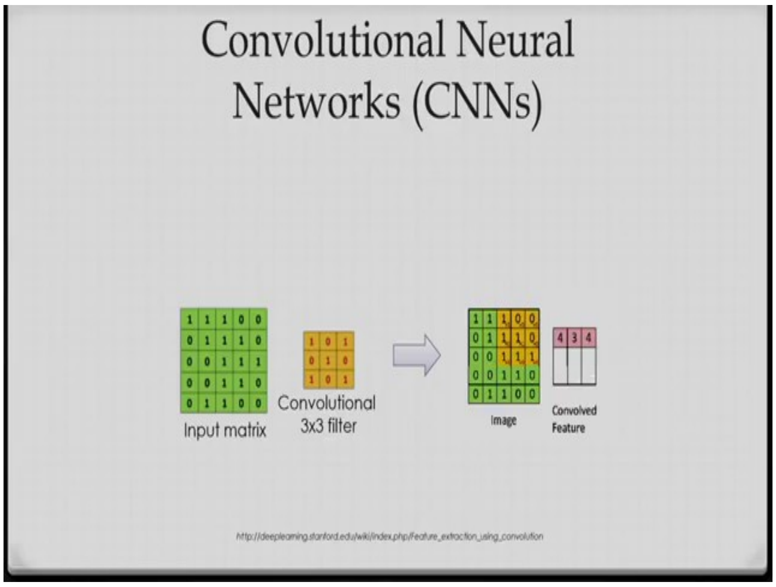


So, here I am showing the structure of convolution neural network one structure. So, here you see I am only considering the filters and the max pool this I am considering. So, here you can see corresponding to this input image, I am considering number of filters in the convolution layer. So first this represents the filter and this represents the max pool. So, first I am considering 64 filters for the convolution layer, after this again another convolution layer, after this I am considering 1 max pool layer, after this I am again considering 128 filter convolution layer, after this again another convolution layer after this again the max pooling layer you can see so like this I can consider the cascade of the convolution layer and the max pool layer we can consider.

And finally, I am getting the max pool layer I am getting this one and that is converted into 1 d vector and after this I am considering 1 fully connected network for classification. So, after the fully connected networks I will be getting the outputs, output mainly the classification of the objects present in an image. So, you can see I am considering the cascade of these layers, one is a convolution layer, another 1 is a max pool layer, I am considering the cascade of this because already I have explained the deep learning means, the cascade of nonlinear transformations and it is nothing but end to end learning.

So, fort learning we can use the back propagation technique so we can adjust the parameters of these layers by the process of the backpropagation learning and you can see here also final max pool layer that is converted into 1 d vector and after this we can consider the fully connected layers. And you can see so what will be the output of the fully connected layers, I will be getting the classes. So, different classes I will be getting that is the object recognition I am considering; living room, bedroom, kitchen, bathroom, outdoor that is the image classification I am considering based on the input, the input is the image I am considering, this is the concept of the convolution neural network.

(Refer Slide Time: 28:31)

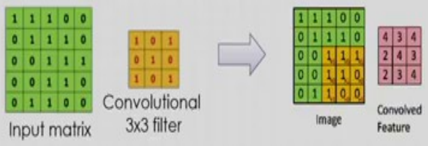


Convolutional Neural Networks (CNNs)

Input matrix
Convolutional 3x3 filter
Image
Convolved Feature
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

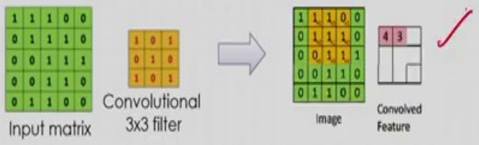
$$\begin{bmatrix}1&1&1&0&0\\0&1&1&1&0\\0&0&1&1&1\\0&0&1&1&0\\0&1&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix} \Rightarrow \begin{bmatrix}1&1&1_{\times1}&0_{\times0}&0_{\times1}\\0&1&1_{\times0}&1_{\times1}&0_{\times0}\\0&0&1_{\times1}&1_{\times0}&1_{\times1}\\0&0&1&1&0\\0&1&1&0&0\end{bmatrix} \quad \begin{bmatrix}4&3&4\\ & & \\ & & \end{bmatrix}$$

Convolutional Neural Networks (CNNs)



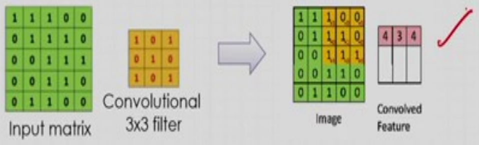
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolutional Neural Networks (CNNs)

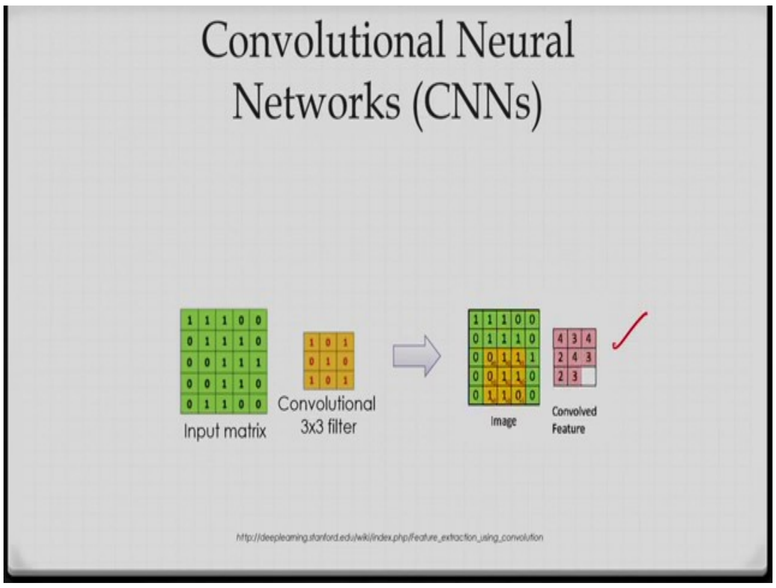


http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolutional Neural Networks (CNNs)

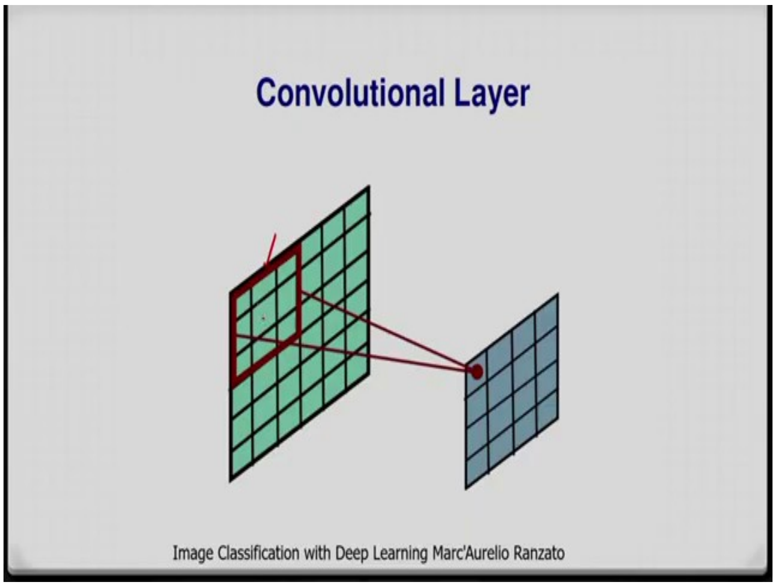


http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution


Convolutional Neural Networks (CNNs)
Input matrix
Convolutional 3x3 filter
Image
Convolved Feature
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

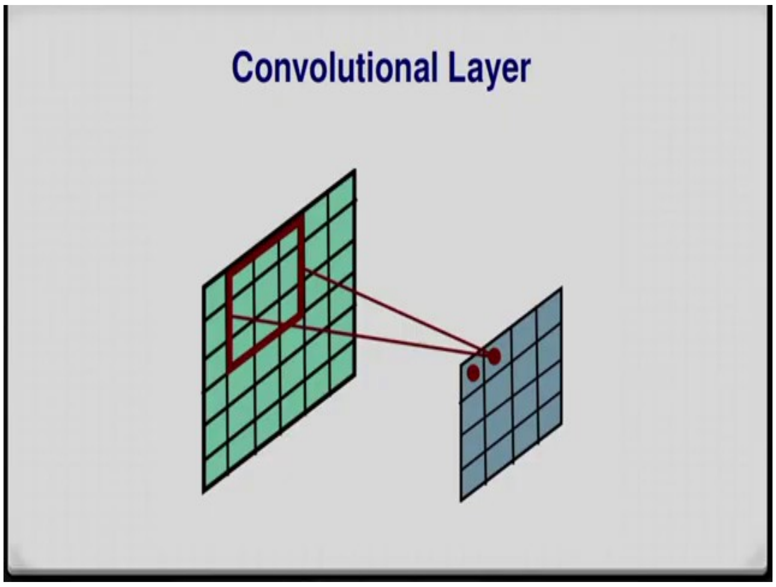
And in this illustration, I am showing again the concept of the convolution. So, I am considering the input metrics and I am considering the convolution 3 by 3 filter and after this you can see the process of the convolution. So, after convolution I will be getting the feature map so, you can see I am getting the feature map here.

(Refer Slide Time: 28:59)

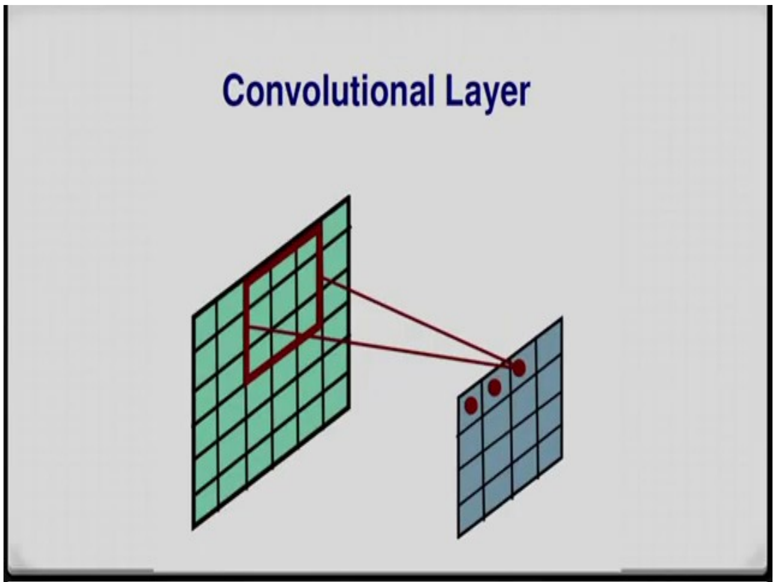

Convolutional Layer
Image Classification with Deep Learning Marc'Aurelio Ranzato

The next again I am showing the concept of the convolution. So, I am considering the filter, suppose this is my filter, so 1 filter I am considering, so I have to place the filter over the image and after this I have to determine the convolution, I have to determine the response of the filter. So, corresponding to the centre pixel, I am putting or I am placing the filter over the image and I have to determine the output that is the convolution output I have to determine.

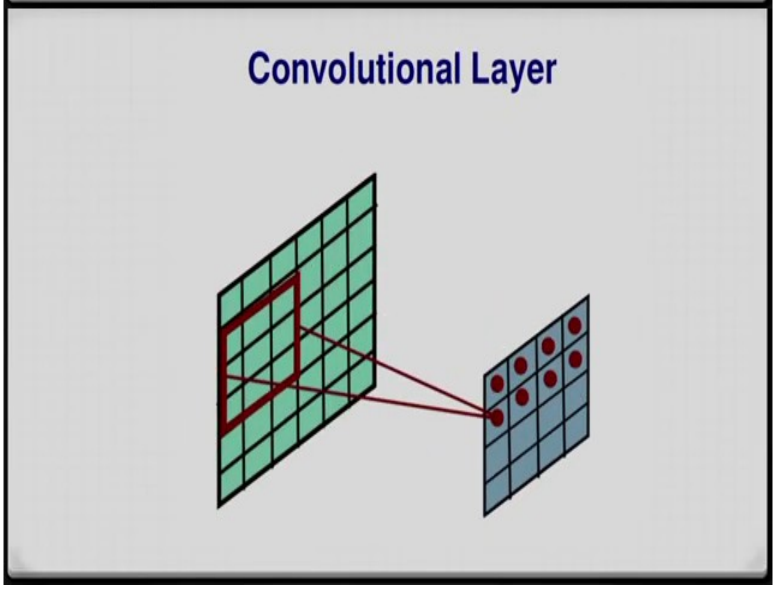
(Refer Slide Time: 29:25)



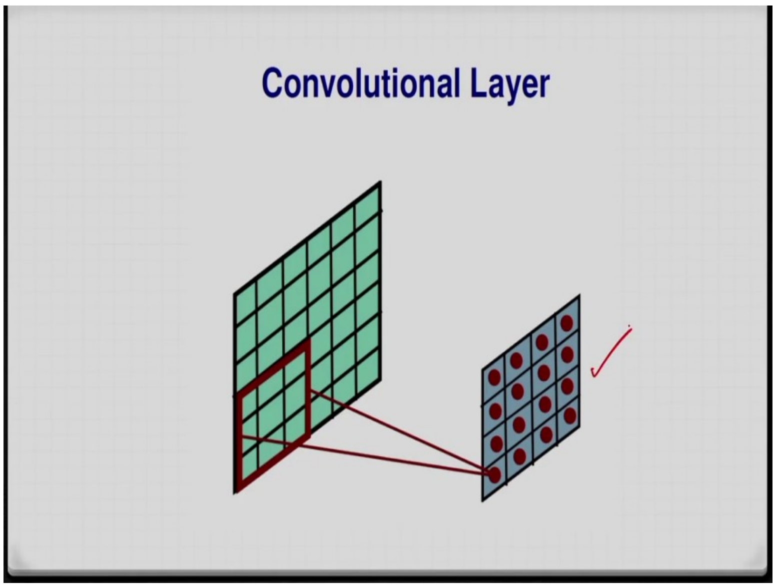
Convolutional Layer



Convolutional Layer

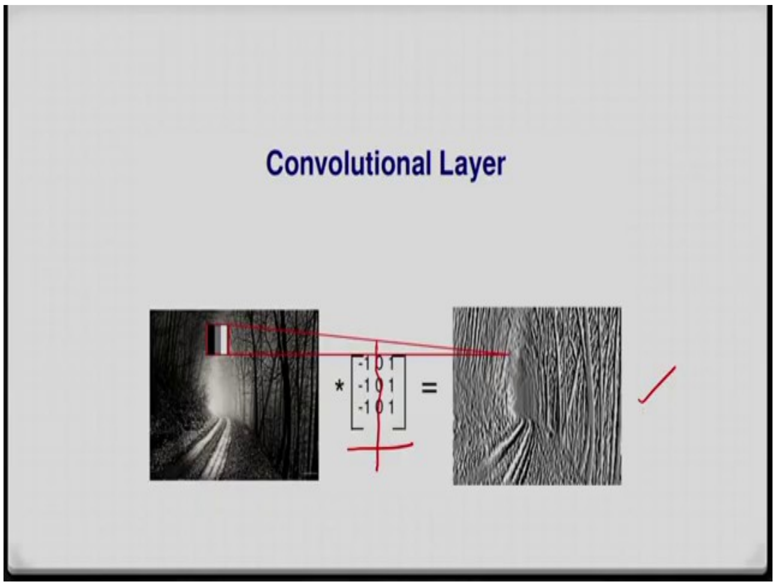


Convolutional Layer


Convolutional Layer

And like this, I have to shift the filter like this for, all the pixels of the image I have to do this, you can see just I am doing the convolution. So, this is the concept of the convolution how to determine the convolutions by using the filter. And I am getting the feature map. So, you can see I am getting the feature map here so this is my feature map after the convolution.

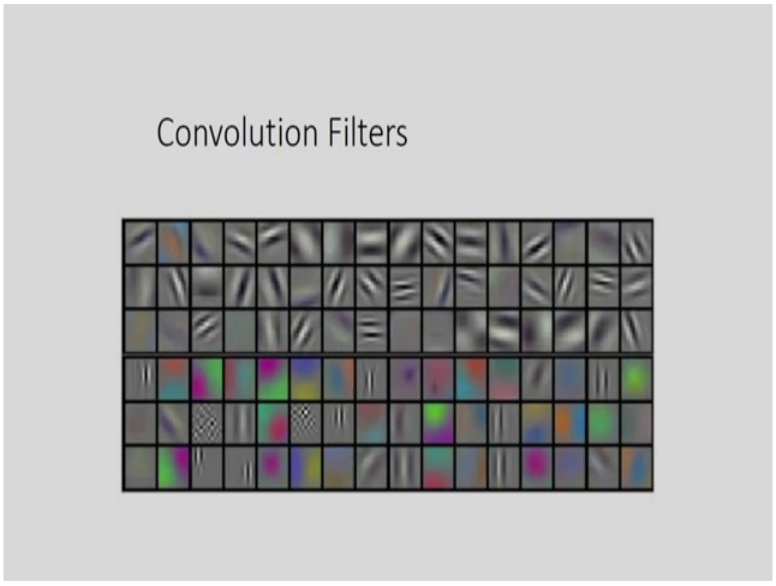
(Refer Slide Time: 29:53)


Convolutional Layer

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

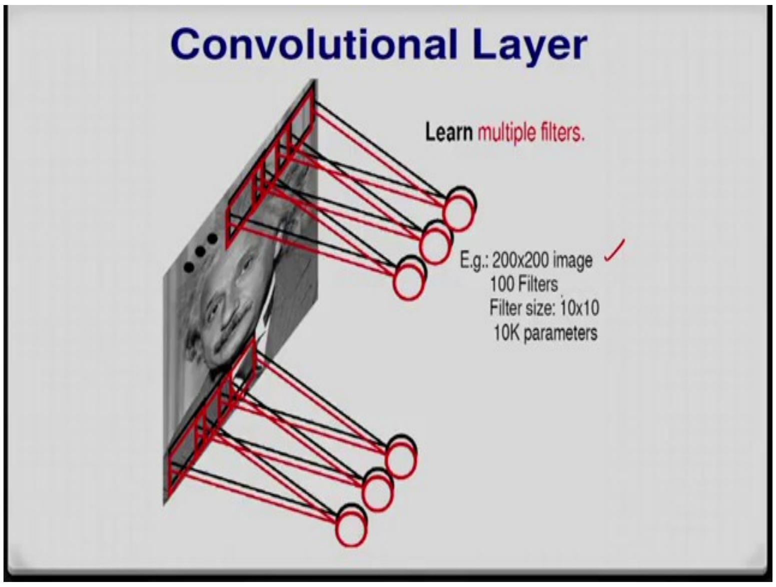
And you can see here, if I consider this filter, if I consider this filter and if I do the convolution with the help of this filter, then I will be getting the vertical edges I will be getting because this is nothing but the high pass filter, if I consider because you can see the response, here response is 0 0 0 so that means, I will be getting the vertical lines by considering this filter that is the convolution filter.

(Refer Slide Time: 30:32)


Convolution Filters

And like this, I am giving some examples of the convolution filters to extract important features from the input image. So, from the input image I can extract important information, important features of the images, the low-level features, the mid level features, and the high level features we can extract by considering the convolution filters.

(Refer Slide Time: 30:50)

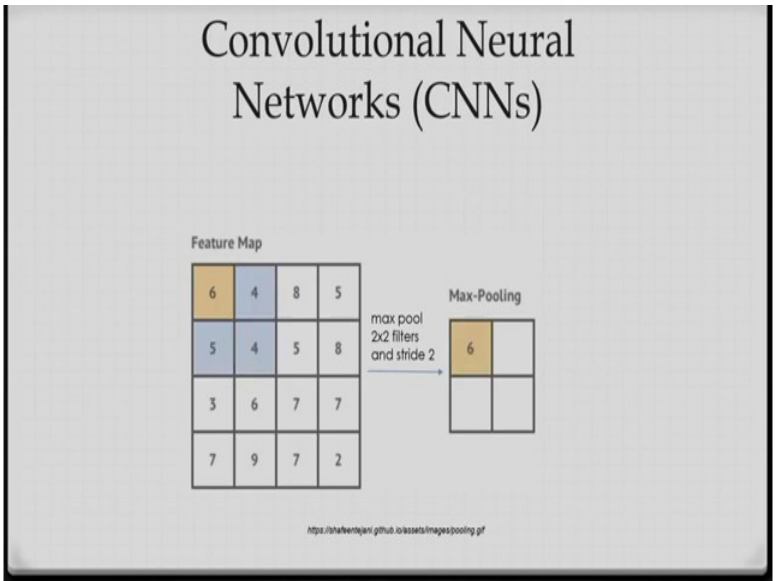

Convolutional Layer

Learn multiple filters.

E.g.: 200x200 image ✓
100 Filters
Filter size: 10x10
10K parameters

And here again, I am showing the concept of the convolution layer. So, we can consider multiple filters instead of considering only one filter, we can consider multiple filters. So, in this case, I am considering 200 into 200 image and maybe we can consider 100 filters and the filter size will be 10 into 10 then corresponding to this I may have 10k parameters, so we have to do the convolutions.

(Refer Slide Time: 31:19)



Convolutional Neural Networks (CNNs)

Feature Map

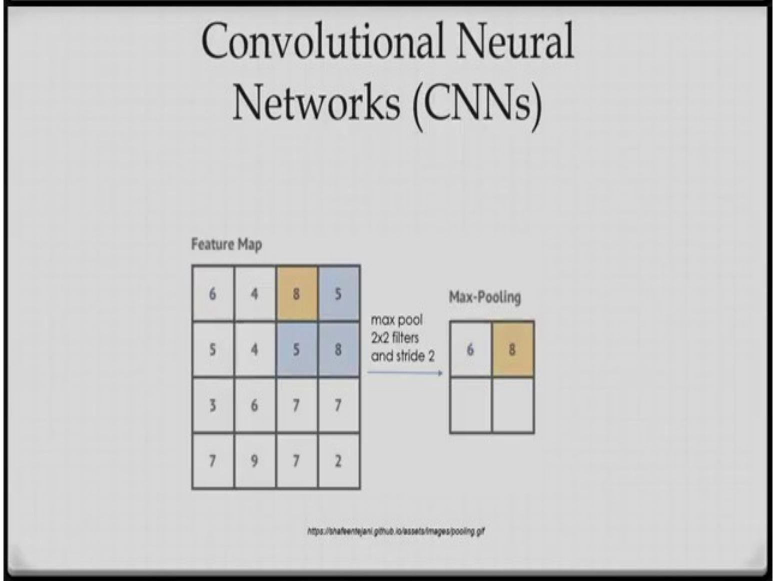
6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

max pool 2x2 filters and stride 2

Max-Pooling

6	

https://shafeentejani.github.io/assets/images/pooling.gif



Convolutional Neural Networks (CNNs)

Feature Map

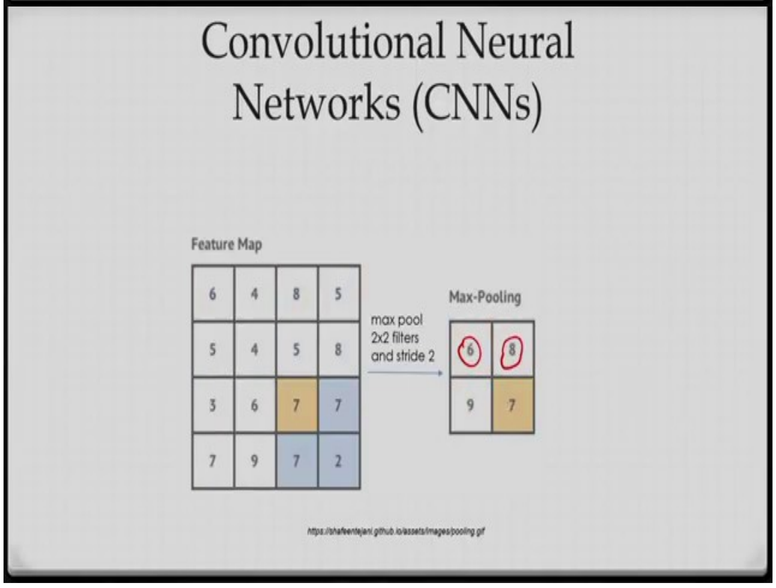
6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

max pool 2x2 filters and stride 2

Max-Pooling

6	8

https://shafeentejani.github.io/assets/images/pooling.gif



Convolutional Neural Networks (CNNs)

Feature Map

6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

max pool 2x2 filters and stride 2

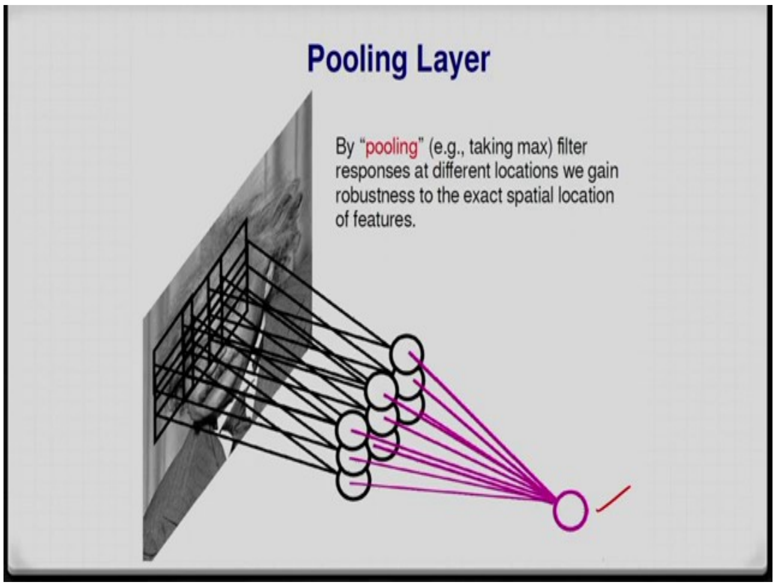
Max-Pooling

6	8
9	7

https://shafeentejani.github.io/assets/images/pooling.gif

After getting the feature map we have to apply the max pooling technique, the max pooling technique already I have explained. So, you can see corresponding to the 2 × 2 neighbourhood the first neighbourhood that maximum value is 6, corresponding to the second neighbourhood the maximum value is 8 and like this, I am considering that max pooling operations.

(Refer Slide Time: 31:40)

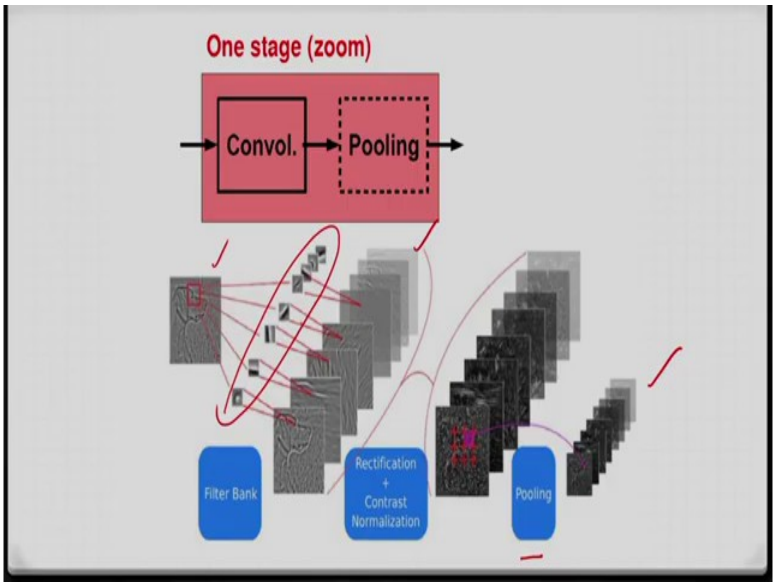


Pooling Layer

By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

So, what is the importance of the pooling layer. So, in this example, you can see let us assume that filter is an eye detector. So, I want to detect the eye of this face that is present in the image. So, how can we make the detection robust to the exact location of the eye. So, first I have to do the convolution so that we can extract some features, after this I am applying the max pooling, you can see the pooling I am applying so that we can locate the location of the eye because max pooling means I am taking the maximum value I am considering. So, that means, the exact spatial locations of the feature we can determine by considering the pooling operation.

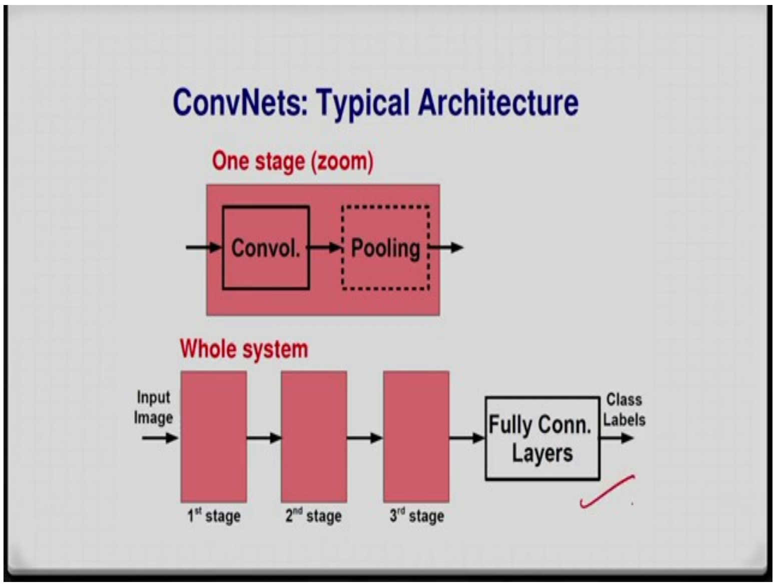
(Refer Slide Time: 32:29)



So finally, I want to say that 1 so you can see here, so if I consider 1 stages of this architecture, so I have the convolution layer and after this I have the pooling layer. So, you can see corresponding to suppose this image I am applying number of filters, you can see the number of filters like this so that is the filter bank, after this I am doing the convolution. So, I will be getting the feature map.

So, you can see so these are the feature maps, after this we can consider the activation function, the activation function is Relu I can consider, and after this we can apply the pooling operation. We can apply the max pooling operation or maybe the average pooling operations and after applying the pooling operations, I will be getting the outputs so, output will be this. So, this is 1 stage considering convolution layer and the pooling layer.

(Refer Slide Time: 33:30)

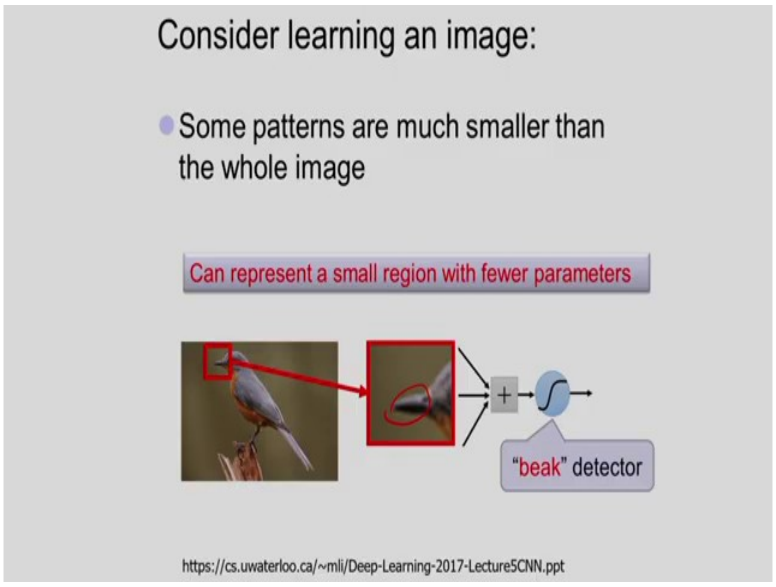

ConvNets: Typical Architecture

One stage (zoom)

Whole system

And if I consider a typical architecture of the CNN the convolution neural networks, so you can see we may have number of stages. So, maybe the first is maybe the convolution pooling again convolution pooling convolution pooling. So, we may have number of stages and finally, we have the fully connected layers, and output will be the class levels corresponding to the input image. So, this is a typical architecture of the convolution neural network.

(Refer Slide Time: 33:59)


Consider learning an image:

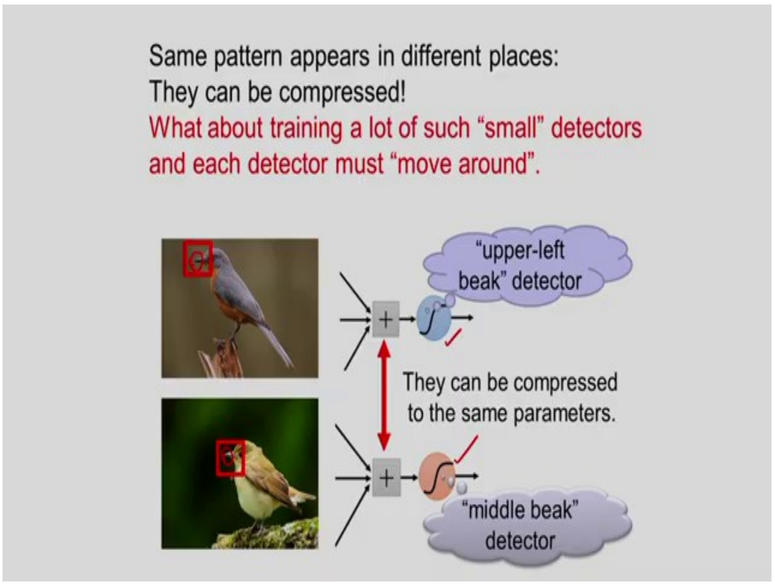
- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters

https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture5CNN.ppt

Now, let us consider learning of an image. So, suppose in this example, I am considering the beak detector that means, I want to determine and this portion of an image this portion of the bird, and in this case the sum patterns are much smaller than the whole image and can represent a small region with few parameters. So, how to represent that portion by considering few parameters.

(Refer Slide Time: 34:28)

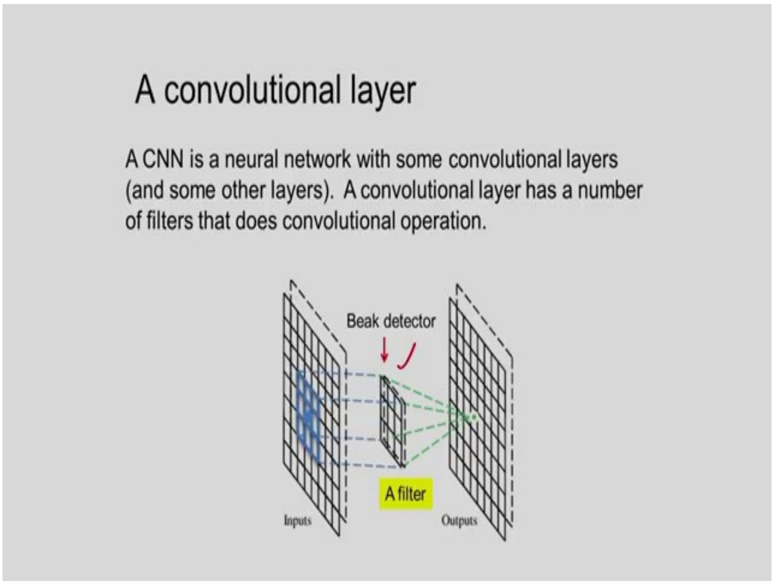


Same pattern appears in different places:
They can be compressed!
What about training a lot of such "small" detectors and each detector must "move around".

"upper-left beak" detector
They can be compressed to the same parameters.
"middle beak" detector

So, here you can see if I consider this portion, this portion is appearing here in the second image, in the second image in the different positions. So, how to represent this so if I consider suppose this one that is the upper left beak detector and if I consider this one that is a middle beak detector. Now, these two can be combined because these two can be compressed to the same parameters because this pattern is appearing in another position of the image. So, that is why if I consider the upper left beak detector and the middle beak detector, they can be compressed to the same parameters, because they appear in different places of the image.

(Refer Slide Time: 35:19)

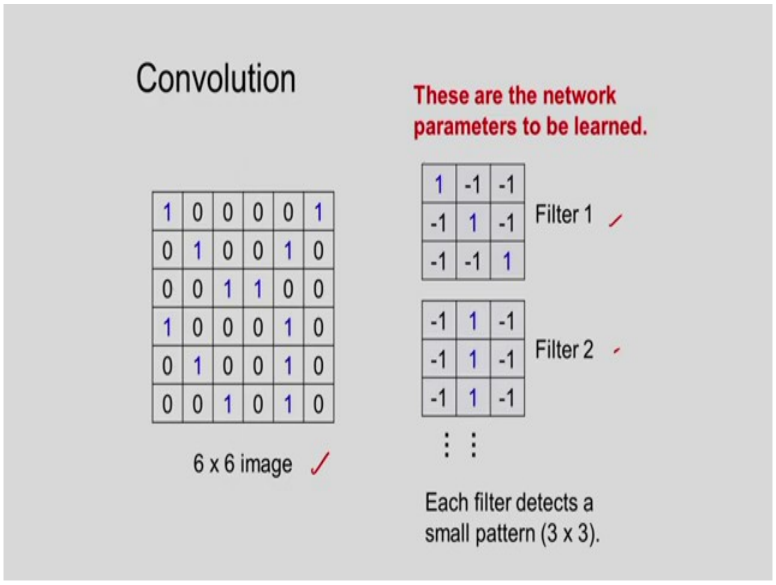


A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.

So, that means we can consider a filter for beak detector. So, a filter I am considering to detect the beak that is a beak detector and that is nothing but the convolution layer. So, we have to do the convolution and I will be getting the feature map.

(Refer Slide Time: 35:33)



Convolution

These are the network parameters to be learned.

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

6 x 6 image

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

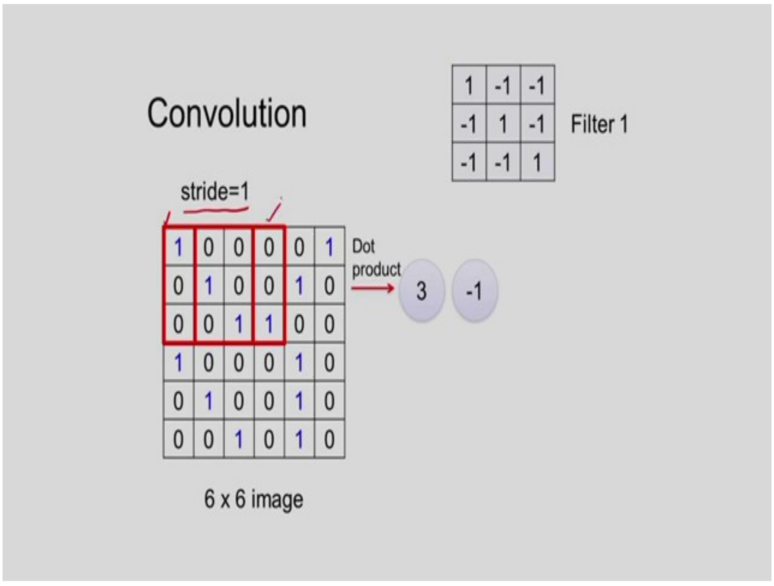
-1	1	-1
-1	1	-1
-1	1	-1

⋮ ⋮

Each filter detects a small pattern (3 x 3).

So, suppose I am considering the 6 by 6 image, the image is like this and I am considering suppose filter 1, filter 2, filter 3, like this and each filter can detect a small pattern, the pattern is suppose 3 × 3 size. So, the filter size is 3 × 3 size and the image size is 6 × 6 and we have to consider a network. So, we have to learn the parameters of the network so for this we are considering the filters, the filter 1, filter 2 and number of filters I am considering and each filter can detect a small pattern, the pattern maybe the 3 × 3 pattern. So, let us see about this convolution procedure.

(Refer Slide Time: 36:26)

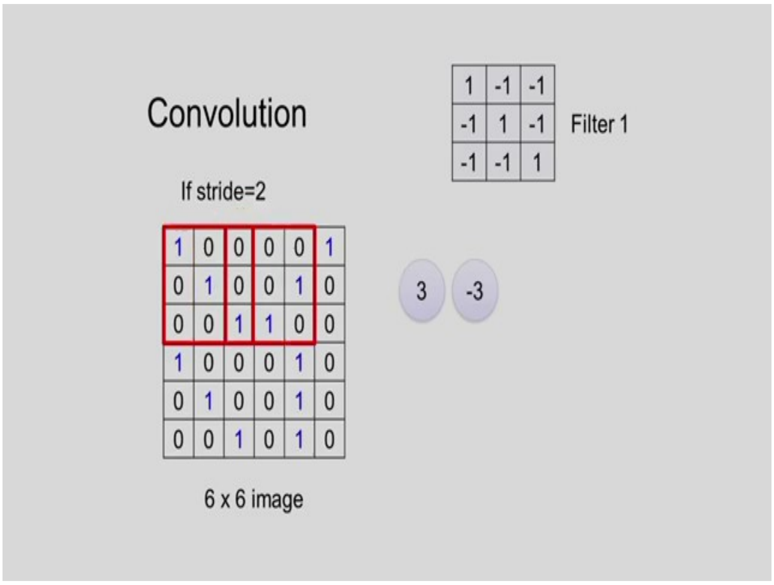


$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

So, you can see if I apply the convolution between the filter 1 and the image, so I will be getting the output something like this $3-1$ and I am considering the stride 1 that means the filter marks it is moved over the image by only 1 pixel. So, that is why I am considering the stride 1. So, if I consider the first position of the filter here and if I consider the second position of the filter here that means, in the second position, the filter is moved by 1 pixel, by only 1 pixel it is moved so that is why it is called a stride 1.

(Refer Slide Time: 37:05)

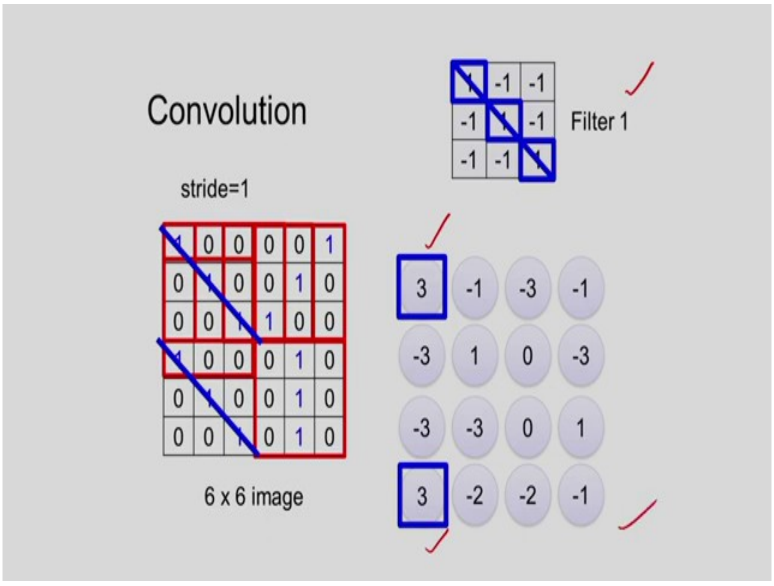


$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

And if I consider a stride 2 that means the filter marks is moved by 2 pixels. So, if I consider a higher value of the stride, then in this case I can reduce the size of the feature map.

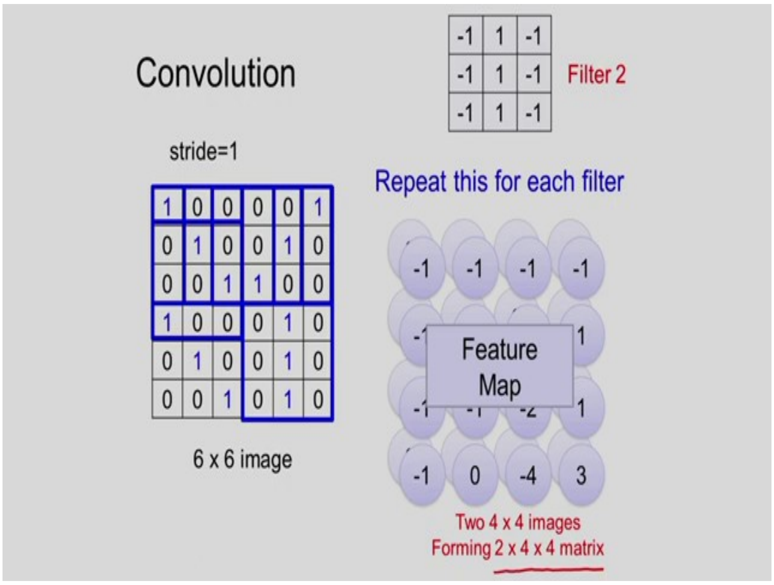
(Refer Slide Time: 37:21)



And corresponding to this you can see I am getting the corresponding stride 1, I am getting the feature map something like this. And you can see corresponding to this filter, the filter 1 that can determine diagonal lines that means, what is the feature which can extract by the filter 1, the filter 1 can extract the diagonal lines. So, that is why you can see the response, the response is 3 corresponding to the diagonal lines. Here also you can see the response 3 corresponding to the diagonal lines present in an image.

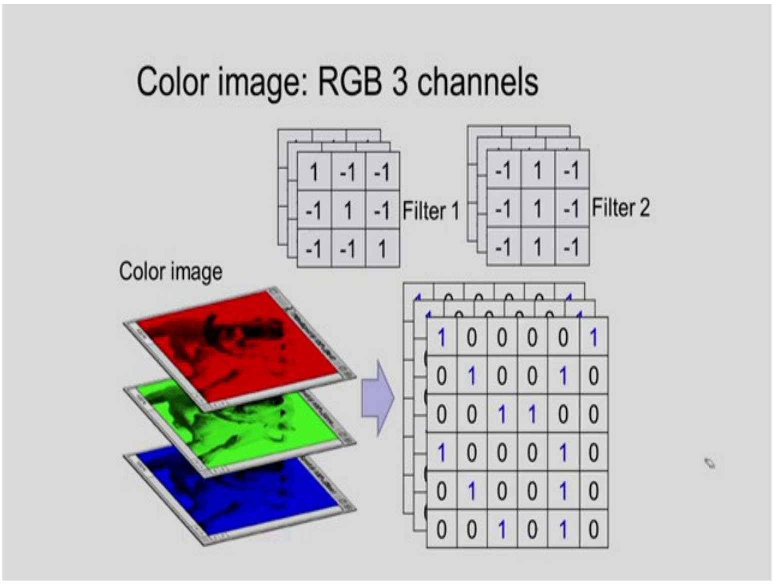
So, by using the filter 1, I can extract the information of the diagonal line present in an image. So, that filter I can consider filter 1 and you can see after convolution I will be getting the feature map a feature map will be something like this.

(Refer Slide Time: 38:15)



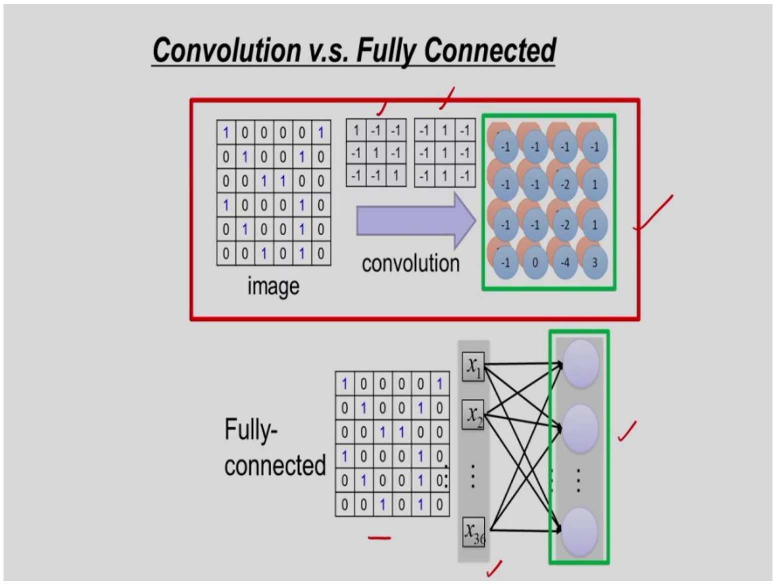
And after this we can consider filter 2 and we have to repeat this for each and every filters, we have to do this procedure. So, corresponding to filter 2 also I will be getting the feature map that means I will be getting 2 × 4 ×4 matrix that is a feature map. So, feature maps obtained by considering filter 1 and filter 2, and in this example, I am considering stride 1. So, you can consider strike 2 also like this you can consider.

(Refer Slide Time: 38:40)



And if I consider a colour image. Suppose in colour image, we have 3 channels, R channel, G Channel and the B Chanel. corresponding to these 3 channels, we can determine the feature maps by considering filter 1, filter 2, filter 3 like this we can consider and we can do the convolution between the image and the filters. So, for this we have to consider R channel, G channel and the Blue Chanel separately. So, we have to do the convolution between the R channel and the filter, convolution between the G Channel and the filter, convolution between the Blue channel and the filter, we can do the convolution like this and we will be getting the feature map.

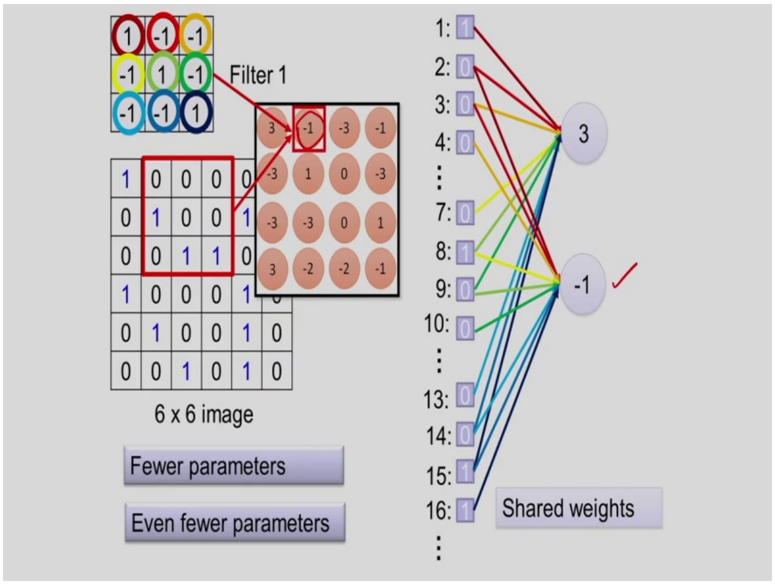
(Refer Slide Time: 39:24)



And in this case, you can see the difference between the convolution and the fully connected because in case of the fully connected network that is available in the artificial neural network, the number of parameters are too high. So, that is why by the process of the convolution, we can reduce the number of parameters. So, by using the convolution operation, you can see I can reduce the number of parameters. So, here you can see I am applying the convolution between the image and the filters. So, I am considering these filters, and I am getting the feature map like this.

But if I consider suppose this image, that is 6 × 6 image, if I consider a fully connected neural network, so, that means the input nodes, we have 36 nodes and after this we have the hidden layer and after this, we have the interconnections between the input layer and the hidden layer. So, there is the fully connected neural network. So, we need many many parameters in case of the artificial neural network. But because of the convolution operation, we can reduce the number of parameters.

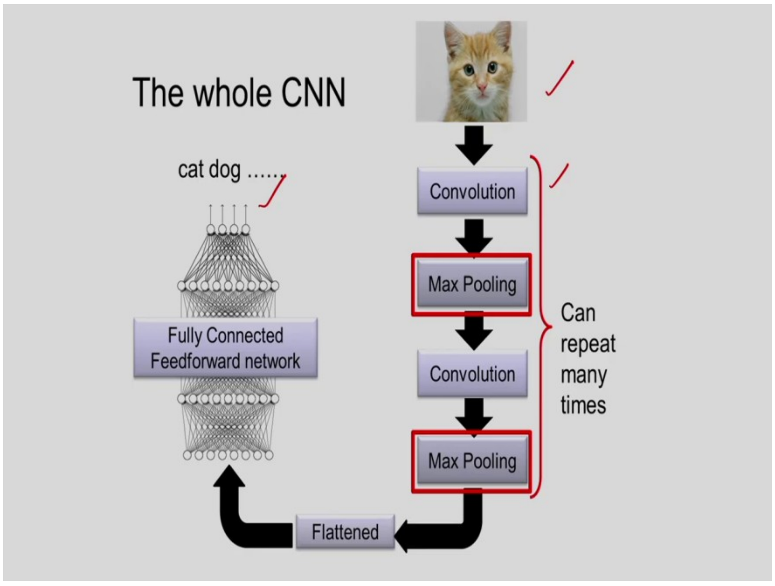
(Refer Slide Time: 40:34)



So, here you can see, I am applying the convolution and by using the convolution I will be getting few parameters only as compared to fully connected networks. So, in this case, you can see I will be getting 3 here, that 3 that what is the 3, the 3 is nothing but the result of the convolution between the image and the filter 1. So, that portion of the image I am considering and that is a convolution between that portion of the image and the filter 1, so, output is 3. So, that means corresponding to suppose this pixel centered pixel I will be getting the value the value is suppose 3.

So, to get 3, so, we are doing the connections like this. So, only connect to 9 inputs and it is not fully connected. So, that means, I am reducing the number of connections, number of parameters. In the second case, you can see if I do the sharing of the weights, then in this case, I can reduce the number of parameters. So, corresponding to the second output, the second output is minus 1, second output is minus 1 I can share the weights of the layer you can see. And by this process, I can reduce the number of parameters. So, sharing of the weights is possible. So, by sharing the weights, we can reduce the number of parameters, that is not possible in case of the fully connected neural networks. So, that means the convolution helps in reducing number of parameters of the network.

(Refer Slide Time: 42:12)

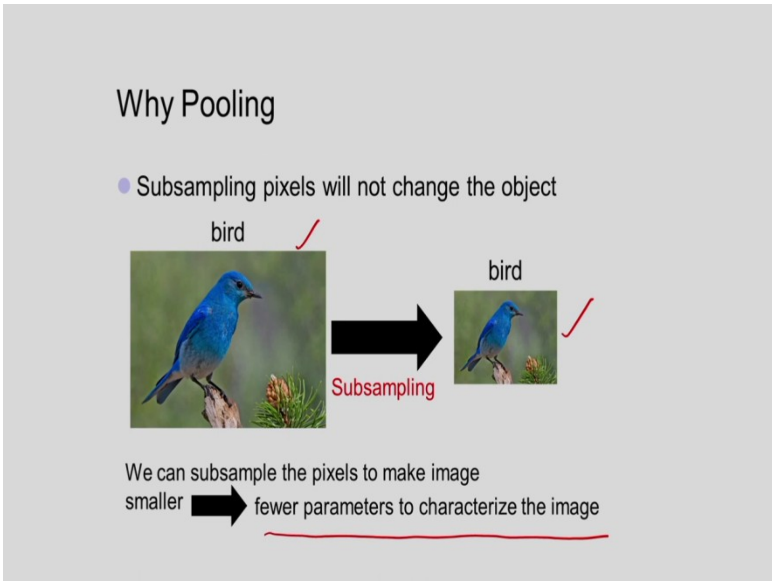


The whole CNN

So, the entire CNN, that is the whole CNN will be like this. So, my input is the image, after this I am doing the convolution, after this I am applying the max pooling. And again, I am doing the convolution and after this again I am applying the max pooling and that can be repeated many times the convolution, max pooling, convolution, max pooling, that can be repeated many times. After this the output of the max pooling and that is converted into 1 D vector, that is the flat end.

The output of the max pooling that can be converted into 1 D vector that is the flat end , after this we can consider a fully connected feed forward network and after this I will be getting the outputs. So, output means the number of classes I will be getting, that is the object detection we can do, whether it is a cat or a dog and that we can determine by considering this CNN, the convolution neural network.

(Refer Slide Time: 43:09)



Why Pooling

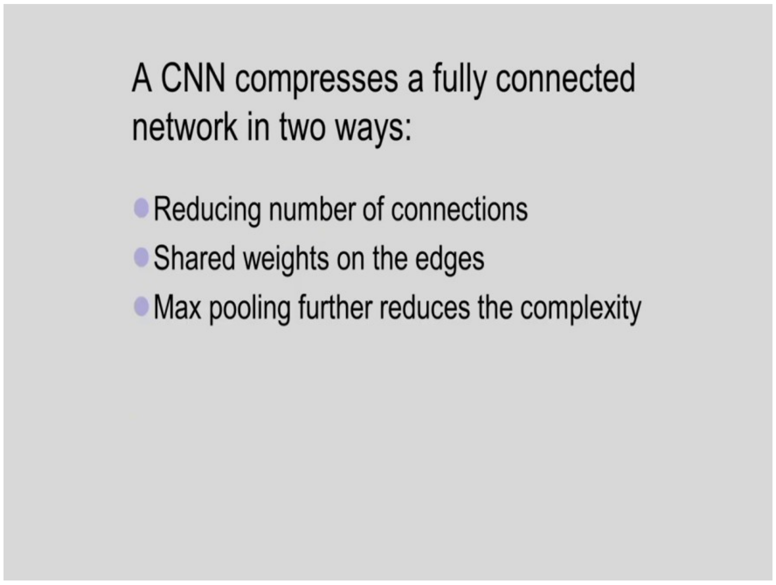
- Subsampling pixels will not change the object

bird
bird
Subsampling

We can subsample the pixels to make image smaller → fewer parameters to characterize the image

And what is the pooling operation actually, why it is important because by using the pooling operation, I can reduce the size of the feature map. So, that means, I can reduce the number of parameters. And one important point is sub sampling pixels will not change the object. So, that means, you can see in this image I am considering this is the bird and after this I am doing the sub sampling, but the object will remain same. So, that means, sub sampling will not change the object, we can sub sample the pixels to make image smaller. So, that means, if I apply this max pooling operation, so, what will be the advantage, the advantage will be pure parameters to characterize the image. So, we will have only few parameters to represent a particular image.

(Refer Slide Time: 44:04)

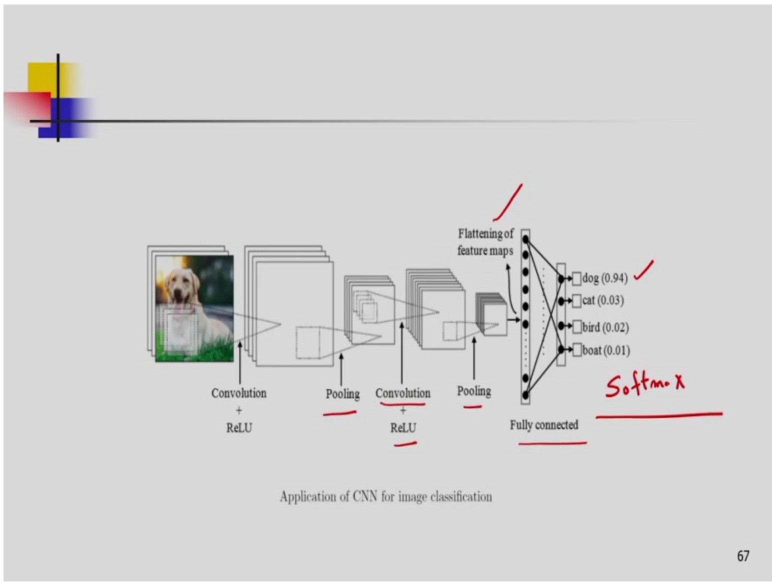


A CNN compresses a fully connected network in two ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

So, that means a CNN compress a fully connected network in two ways. The first one is reducing number of connections, that is one important point and shared weights on the edges. So, we can share that weights and also we can apply that max pooling operations. So, like this we can consider.

(Refer Slide Time: 44:24)



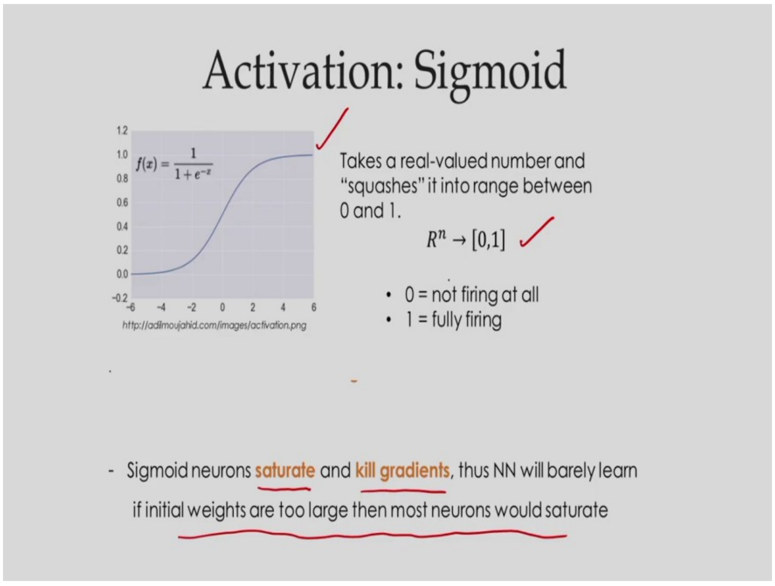
Application of CNN for image classification

67

So, in this figure I am showing the same concept again. So, convolution I am considering and the activation function I am considering the ReLU function I am considering and after this you can see pooling operation I have shown again I have shown the convolution operation and the ReLU. Again, I am considering the pooling operations and finally, the flattening of the feature maps we have done and after this we have the fully connected neural network, that is the feed forward network.

And corresponding to this you can see the outputs the dog 0.94, Cat 0.03, bird 0.02, that is the probabilities and both 0.01. So these probabilities I am considering that these probabilities I am obtaining by considering the function, the function is the softmax. That already I have explained. So this is an activation function that scales numbers into probabilities. So softmax function I am considering so I am getting the probabilities, the probability of dog is 0.94, the probability of cat is 0.03 like this I am considering. So, this is 1 application of CNN for image classification.

(Refer Slide Time: 45:42)



Activation: Sigmoid

Takes a real-valued number and "squashes" it into range between 0 and 1.

$R^n \rightarrow [0,1]$

- 0 = not firing at all
- 1 = fully firing

- Sigmoid neurons **saturate** and **kill gradients**, thus NN will barely learn

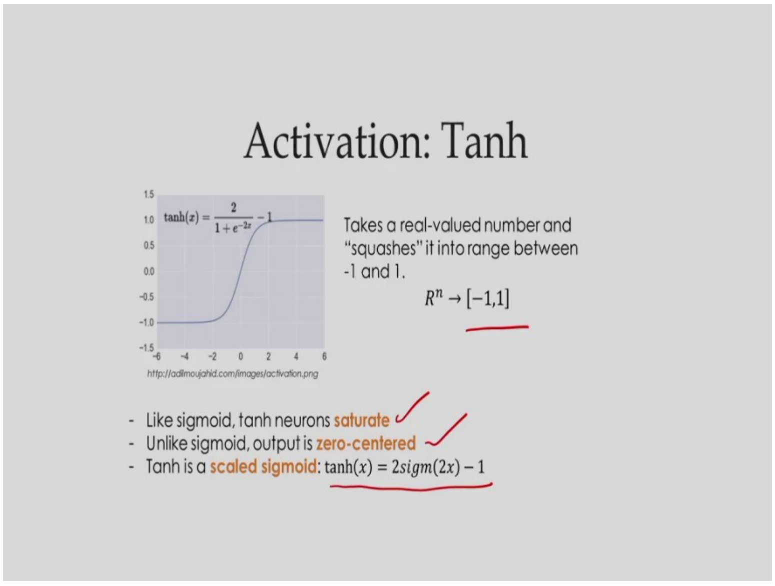
  if initial weights are too large then most neurons would saturate

$f(x) = \frac{1}{1+e^{-x}}$
http://adilmoujahid.com/images/activation.png

And I want to again highlight the concept of the activation function, the sigmoid function, already I have explained in my class of artificial neural network. So, you can see this is the sigmoid function$f(x)=\frac{1}{1+e^{-x}}$ takes a real valued number and squashes it into a range between 0 and 1. So, you can see the range is 0 and 1 and one problem of the sigmoid function is that sigmoid neurons saturate and kill gradients.

That is the main problem of the sigmoid function, the saturation problem and the kills gradient. Because we have to train the network based on the stochastic gradient descent algorithm, and that already I have explained that is the gradient descent algorithm. So, if I consider suppose the stochastic gradient descent algorithm, so, you can see, I am not explaining this concept, the stochastic gradient descent algorithm. So, if I apply the sigmoid function, so, it will saturate and kills the gradient during the training and if the initial weights are too large, then the most of the neurons will be saturated. So, that is the main problem of the sigmoid activation function.

(Refer Slide Time: 47:03)



Activation: Tanh

Takes a real-valued number and "squashes" it into range between -1 and 1.

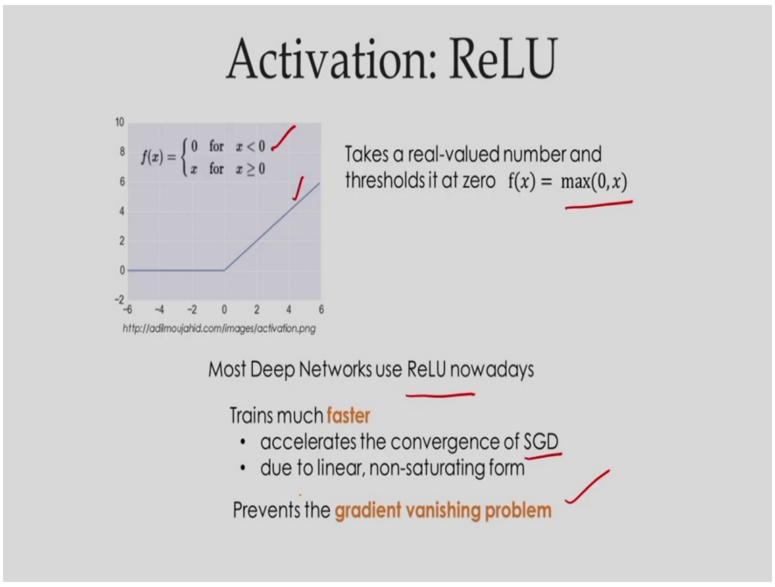
$R^n \rightarrow [-1,1]$

- Like sigmoid, tanh neurons saturate
- Unlike sigmoid, output is zero-centered
- Tanh is a scaled sigmoid: $\tanh(x) = 2sigm(2x) - 1$

$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$
http://adilmoujahid.com/images/activation.png

Another activation function is tanh. So, that is that takes a real value number and squashes it into range between minus 1 and 1. So, output range is minus 1 and 1. So, that is the tanh function. But again the problem is the situation. But one advantage is this output is 0 centered. And you can see the tanh can be represented in terms of the scale sigmoid function. So, tanh can be represented by the sigmoid function, but the problem is again the saturation problem.

(Refer Slide Time: 47:45)



Activation: ReLU

Takes a real-valued number and thresholds it at zero $f(x) = \max(0,x)$

Most Deep Networks use ReLU nowadays

Trains much faster
- accelerates the convergence of SGD
- due to linear, non-saturating form

Prevents the gradient vanishing problem

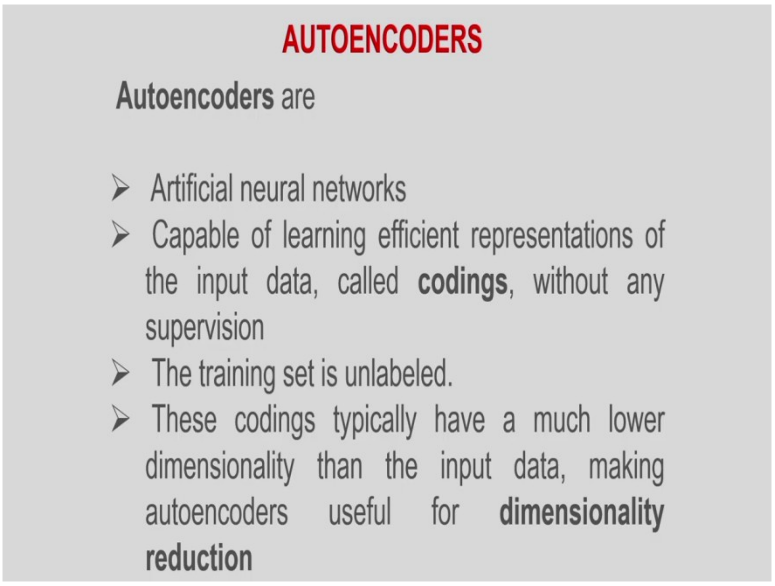
$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
http://adilmoujahid.com/images/activation.png

So, that is why in most of the deep networks in convolution neural networks, we used the ReLU function that is the ReLU function is x = 0 for x ¿0 and it is = x for x ≥0. So, it takes a real value number and thresholds it at 0 f x is equal to maximum of 0 and x. So, you can see it is a linear function. So, in case of ReLU, that is the rectified linear unit you can see the trains much faster, the training is faster and accelerates the convergence of the stochastic gradient descent.

So, that means the convergence will be fast and due to linear nature the saturation problem will be over. Due to the linear characteristics, the saturation problem will be eliminated that means, it prevents that gradient vanishing problem because of the linear characteristics, because it is a linear function. So, that concept I have not explained.

So, if you are interested about this Stochastic gradient descent algorithm, so you can see the concept of the stochastic gradient descent algorithm as SGD, that you can see and also the concept of the situation that is the gradient vanishing problem you can see. So, based on this principle, most of the deep networks used the ReLU function. So, this is about the convolution neural networks.

(Refer Slide Time: 49:25)



AUTOENCODERS

Autoencoders are

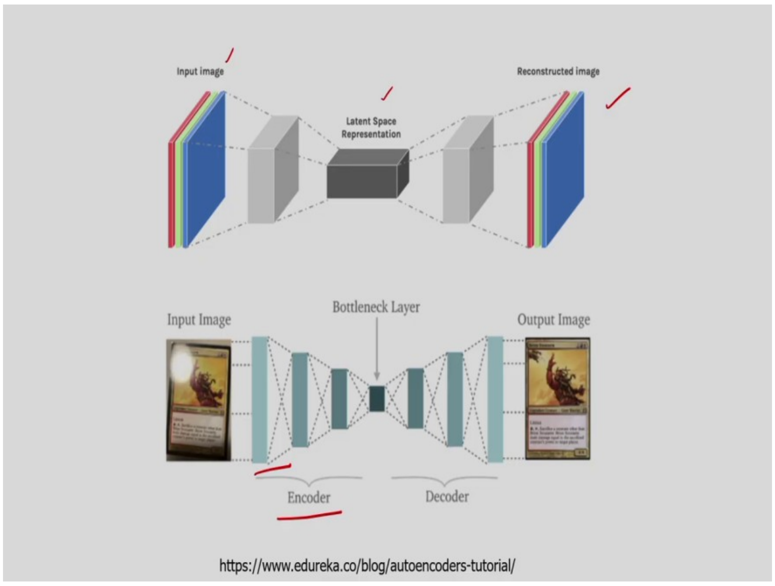
- Artificial neural networks
- Capable of learning efficient representations of the input data, called **codings**, without any supervision
- The training set is unlabeled.
- These codings typically have a much lower dimensionality than the input data, making autoencoders useful for **dimensionality reduction**

Now, I will discuss the concept of auto encoders. So, what is auto encoders? auto encoders are artificial neural networks and auto encoder neural network is an unsupervised machine learning algorithm that applies backpropagation setting the target values to be equal to the inputs. Also, auto encoders are used to reduce the size of the inputs into smaller representations. And suppose, if anyone needs the original data, they can reconstruct it from the compressed data.

Here you can see the auto encoders are artificial neural networks and capable of learning efficient representation of the input data and that is called codings without any supervision.

So, that is why it is called unsupervised machine learning algorithm. The training set is unlabeled and these coding typically have a much smaller dimensionality than the input data making auto encoders useful for dimensionality reduction. So, it is used to reduce the dimension of the input data.

(Refer Slide Time: 50:37)



So, in my next slide you can see the structure of the auto encoder. So, I am showing one input image, after this I am showing the latent space representation of the input image that is nothing but efficient representation of the input image and that is compressed. So, the dimension is reduced, you can see here. So, from the input image I am considering the latent space representation. So, dimension of the input image is reduced and that is nothing but efficient representation of the input image.

And after this from that compress images or maybe the latent space representation, I want to reconstruct the original image. So, this is the reconstructed image. In the second figure, I am considering one input image, maybe the noisy image and after this I am considering one auto encoder. So, this auto encoder compresses the input image. So, for this I am considering the encoder. So, I am getting the latent space representation, after this from this representation, that is from the compressed data, I can reconstruct the output image.

So, what is the function of the encoder? That is used for efficient representation of the input data. So, that means in a auto encoder, I have input layer, I have a bottleneck layer that is nothing but the hidden layer and after this I have the output layer. So, mainly I am considering 3 layers, one is the input layer, the first one is the input layer, after this the bottleneck layer and after this I am considering the output layer. So, it has encoder and decoder.

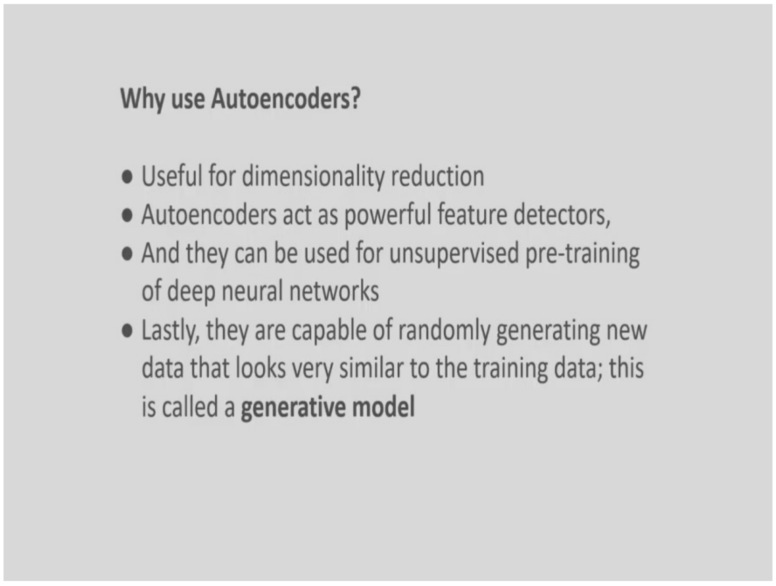
Now, in this case, since I am doing the dimensionality reduction, then what is the difference between auto encoder and a PCA. So, why auto encoders are preferred over PCA, the principal component analysis. So, one point I can mention here, an autoencoder can learn nonlinear transformation with a nonlinear activation function and multiple layers, that is one important point. Also, it is more efficient to learn several layers with an autoencoder rather than learn one huge transformation with PCA, because if I considered a PCA, I have to consider a huge transformation.

But in case of the auto encoder, I can consider a number of layers for efficient representation of data. Also, I can make use of the pre trained layers from another model to apply transferred learning to enhance the encoder and the decoder, that is also another advantage of the auto encoder over PCA. So, what is encoder? Encoder means the part of the network compress the input into a latent space representation.

So, in the figure you can see, I have shown one encoder. So, this part of the network compress the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduce dimension. A compressed image is the distorted version of the original image. So, I am doing the compression by considering the encoder. After this I am getting the code. So, this part of the network represents the compressed input which is fed into the decoder.

So, I am getting the bottleneck layer that is nothing but the latent space representation and after this I have the decoder. So, this layer decodes the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation. So, I can mention one property of the auto encoder, the auto encoder shall only be able to compress the data similar to what they have been trained on. The decompressed output will be degraded compared to the original input. And that is also one important point, that is the decompressed output will be degraded compared to the original input. So, this is about the auto encoder.

(Refer Slide Time: 55:24)

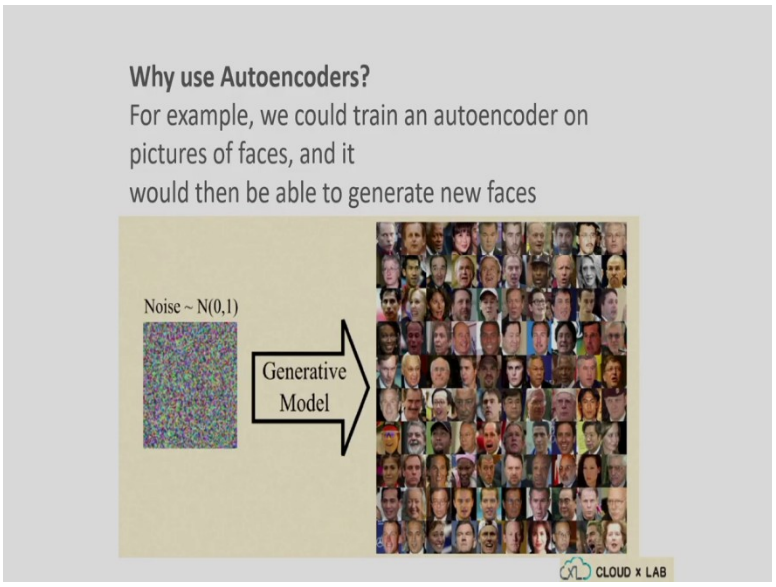


Why use Autoencoders?

- Useful for dimensionality reduction
- Autoencoders act as powerful feature detectors,
- And they can be used for unsupervised pre-training of deep neural networks
- Lastly, they are capable of randomly generating new data that looks very similar to the training data; this is called a **generative model**

Next, why we use auto encoders. So, use for dimensionality reduction, so, we can reduce the dimensionality of the input image. So, that is why we are considering the auto encoder. And with the help of autoencoders I can extract important features from the input image. So, that is also one important point, that is with the auto encoders I can extract important features of the image and they can be used for unsupervised pre training of deep neural networks. So, that is also another advantage. And one important point is they are capable of randomly generating new data that looks very similar to the training data and that is called generative model. So, I can give one example of the generative model in my next slide.

(Refer Slide Time: 56:17)

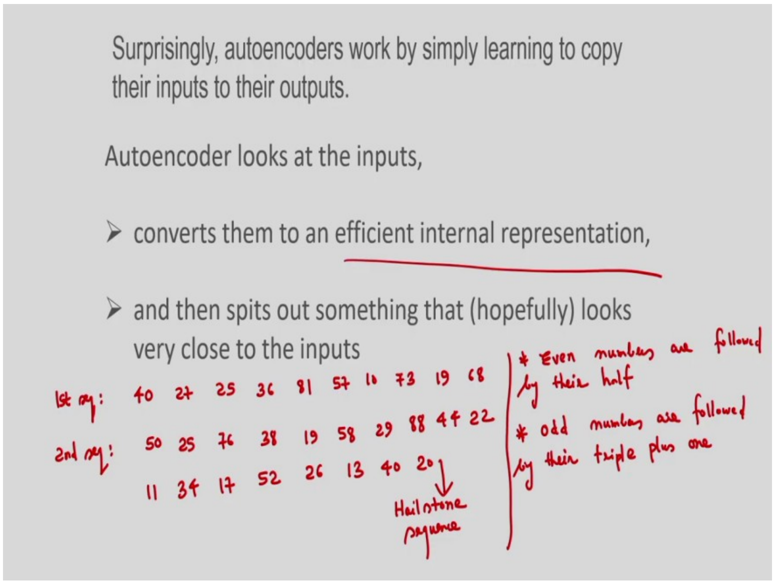


Why use Autoencoders?

For example, we could train an autoencoder on pictures of faces, and it would then be able to generate new faces

So, here you can see. For example, we could train an auto encoder on pixels of faces and it would be able to generate new faces. So, that is the generative model. So, here you can see, I am considering the auto encoder and we can train the auto encoder on pixels of faces. And suppose, I am considering these images. So, this auto encoder will be able to generate new faces that I have shown in this figure. So, that is one application of auto encoder.

(Refer Slide Time: 56:56)



Surprisingly, autoencoders work by simply learning to copy their inputs to their outputs.

Autoencoder looks at the inputs,

- converts them to an efficient internal representation,
- and then spits out something that (hopefully) looks very close to the inputs

1st seq: 40 27 25 36 81 57 10 73 19 68

2nd seq: 50 25 76 38 19 58 29 88 44 22 11 34 17 52 26 13 40 20

Hailstone sequence

* Even numbers are followed by their half
* odd numbers are followed by their triple plus one

Now, surprisingly, auto encoders work by simply learning to copy their inputs to their output. Then, what is the importance of auto encoder? The importance is there that auto encoder looks at the input and converts them to an efficient internal representation, that is efficient internal representation of data, input data. And then spits out something that looks very closer to the inputs. That means, it can reconstruct the inputs which looks very close to the original inputs.

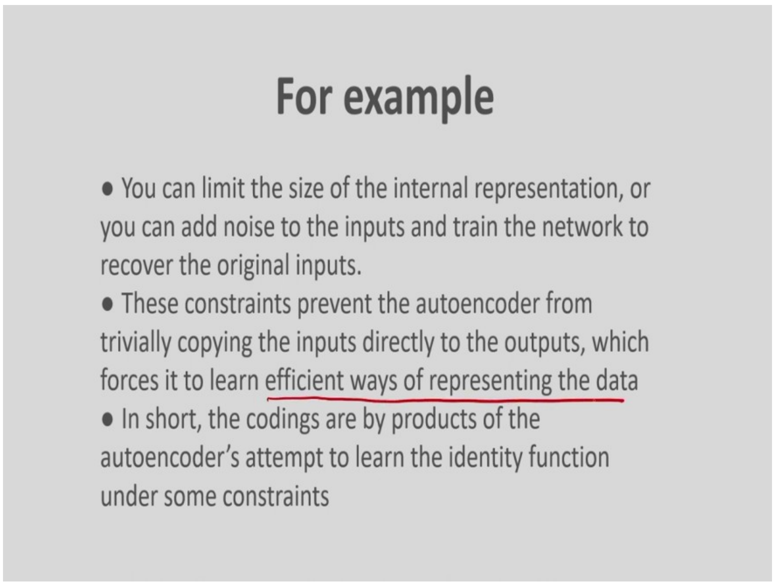
What is the meaning of the efficient internal representation? So, I can give one example, the power sequences 40, suppose 27, 25, 36, 81, 57, 10, 73, 90, 68. So, this is my first sequence, first sequence I am considering. And let us consider another sequence, the second sequence. So, second sequence is suppose 50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, and 20 suppose. So, I am considering two sequences, the first sequence is 40 27 like this and second sequence is 50 25 76. So, like this I am considering.

The question is which one is the easiest to memorize. So, first sequence is the short sequence, the second sequence is the long sequence. At the first glance, it would seem that the first sequence should be easier, since, it is much shorter as compared to the second sequence. But actually, it is not. So, if I considered the second sequence, two simple rules I can formulate. The first rule is, number 1 rule is even numbers are followed by their half. So, this is my first rule.

Second Rule is odd numbers are followed by their triple plus 1. So, I have these two rules to represent the second sequence. This second sequence is a very famous sequence, the name of the sequence is the Hailstone sequence. It is called the Hailstone sequence. So, if I follow these two rules, it is easy to memorize the second sequence, as we have to remember first number and the length of the sequence. So, by using these two rules, I can generate the second sequence.

But in the first sequence, there are no rules. So, that means, in the second sequence, the one important point is existence of a pattern in the second sequence, that is nothing but the efficient internal representation. That is one example, why auto encoders can be used for efficient internal representation of data.

(Refer Slide Time: 61:02)



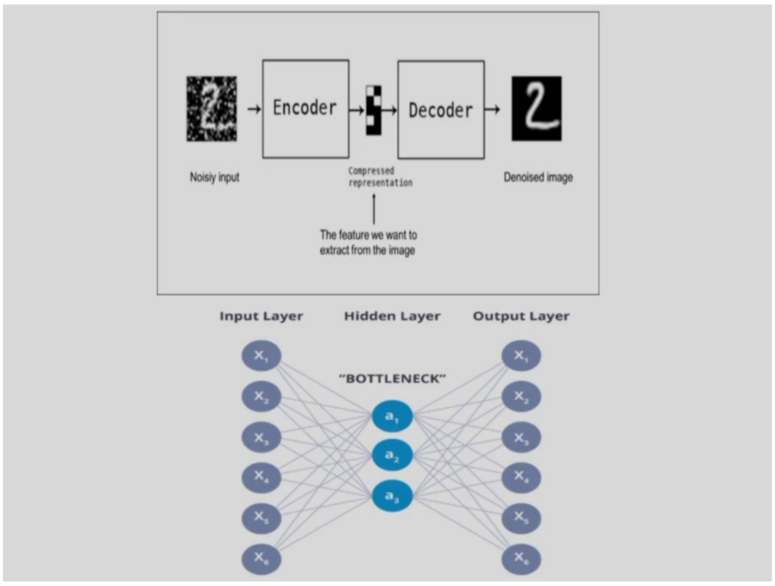
For example

- You can limit the size of the internal representation, or you can add noise to the inputs and train the network to recover the original inputs.
- These constraints prevent the autoencoder from trivially copying the inputs directly to the outputs, which forces it to learn efficient ways of representing the data
- In short, the codings are by products of the autoencoder's attempt to learn the identity function under some constraints

So, here you can see, you can limit the size of the internal representations or you can add noise to the inputs and train the network to recover the original inputs. So, we can limit the size of the internal representation, that means, we can compress the input data more or we can add noise to the inputs and train the network to recover the original inputs. That means, the auto encoder can be used for removing noise. That means, the noise is eliminated and we can reconstruct the original input image.

These constraints prevent the auto encoder from copying the input directly to the outputs, which forces it to learn efficient ways of representing data. So, that concept already I have mentioned, that is efficient data representation by auto encoder. In short, the codings are by products of the auto encoders.

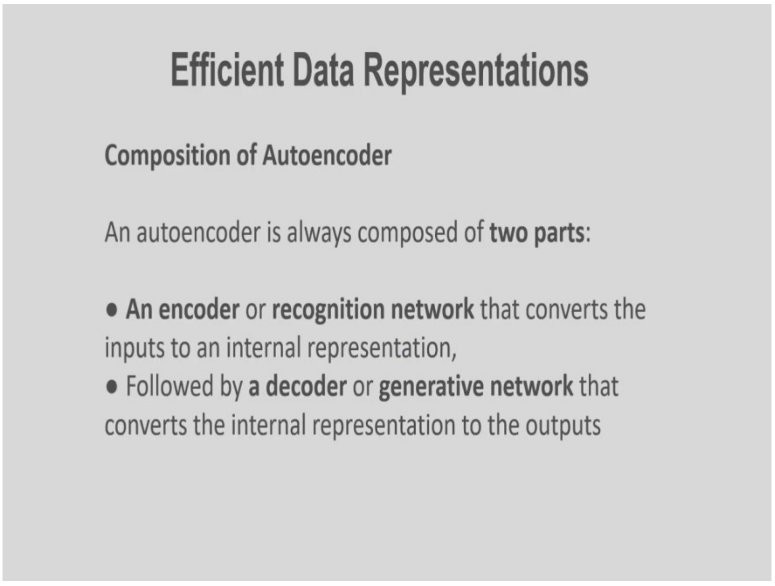
(Refer Slide Time: 62:04)



So, in this figure I have shown, in the first figure I am considering one noisy image and after this I am considering the auto encoder, auto encoder you can see the encoder I am considering after this I am considering the bottleneck layer, that is nothing but the hidden layer. After this I am considering the decoder and I am considering the denoised image. So, denoised image is reconstructed. The input is the noisy input image.

The input seen by the auto encoder is not the raw image, the raw input, but stochastically the corrupted version. So, that means a denoising auto encoder is trained to reconstruct the original input from the noisy version. That is the concept of the denoising Auto encoder. So, input is the noisy input and after this I am considering the efficient representation of the input image, the auto encoder can learn this and after this we can reconstruct the image by removing the noises. In the second case also I am showing the structure of the auto encoder. So, you can see I have the input layer, hidden layer and the output layer.

(Refer Slide Time: 63:24)



Efficient Data Representations

Composition of Autoencoder

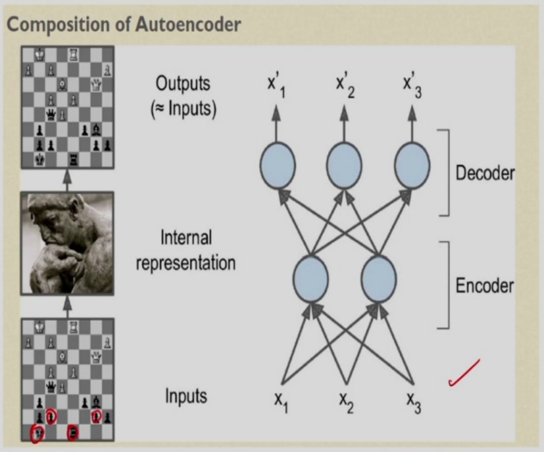
An autoencoder is always composed of **two parts**:

- **An encoder** or **recognition network** that converts the inputs to an internal representation,
- Followed by **a decoder** or **generative network** that converts the internal representation to the outputs

So, composition of auto encoder. So, first is I need encoder order recognition network that converts the input to an internal representation. So, the first component is an encoder. After this the next component is either a decoder or it is called generative network that converts the internal representation to the outputs. So, that means I have one encoder and another one is a decoder in auto encoder. So, encoder is for compressing the input data and after this a decoder or a generative network which can convert internal representation to the outputs.

(Refer Slide Time: 64:12)

Efficient Data Representations



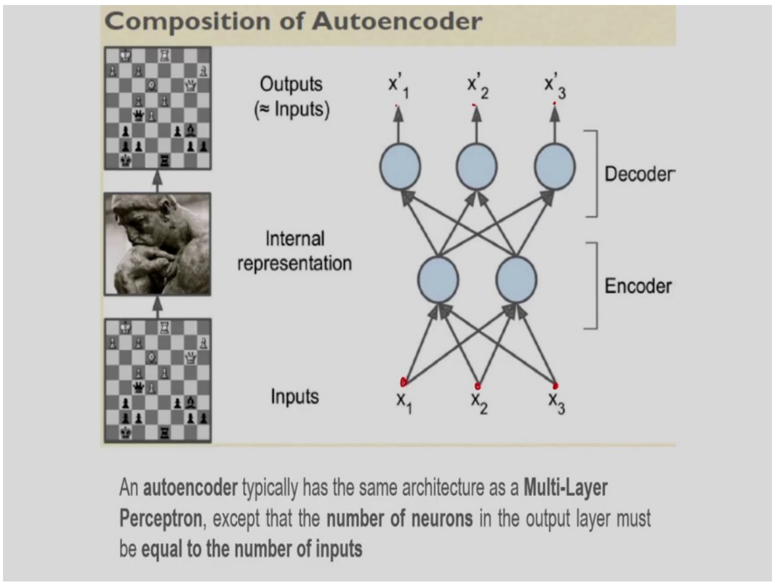
https://discuss.cloudxlab.com/c/course-discussions/deep-learning

So, here you can see I am showing one example of efficient data representation. I am considering one input image that is nothing but the chessboard I am showing. After this the auto encoder can create the internal representation of the input image. After this it can reconstruct the input image. So, output image, it is approximately equal to inputs. So, for this I can give one example. Suppose, I want to remember the position of the pieces industries board.

So, these are my pieces. So, expert chess players can easily memorize the position of all the pieces of in a game by looking at the board just for 5 to 6 seconds, but it is very difficult for most of the people. So, that is a task that most people would find impossible. So, one important point is that pieces are placed in the realistic positions from one actual game and they are not placed randomly.

So, what is the method basically they are doing, these expert chess players, they can learn the patterns of the input image or input data. So, by looking at the chessboard, they can see the patterns and based on these patterns, they can actually represents input data most efficiently. And after this, they can also reconstruct the original patterns. A pattern means, I am considering the position of the pieces in the chess board. So, that is the concept of efficient data representation. So, you can see, I have the inputs, after this I am considering internal representation, after this, I can reconstruct back the original inputs.

(Refer Slide Time: 66:11)

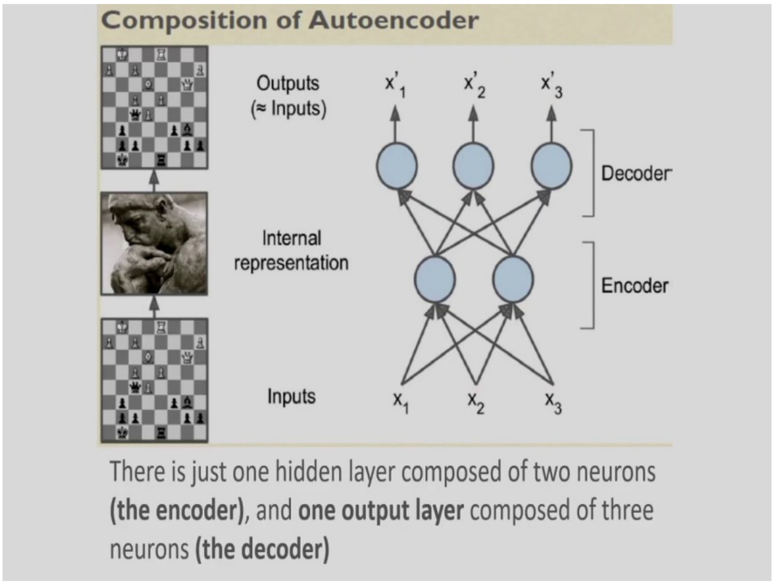


Composition of Autoencoder

An **autoencoder** typically has the same architecture as a **Multi-Layer Perceptron**, except that the **number of neurons** in the output layer must be **equal to the number of inputs**

And the composition of auto encoders, so auto encoders typically have the same architecture as a multi layer perception, except that the number of neurons in the output layer must be equal to the number of inputs. So, in this figure you can see, I have 3 inputs x 1, x 2, x 3, similarly, I have 3 outputs x 1 dash, x2 dash and x3 dash. So, that means number of neurons in the output layer must be equal to the number of inputs in a simple autoencoder.

(Refer Slide Time: 66:49)

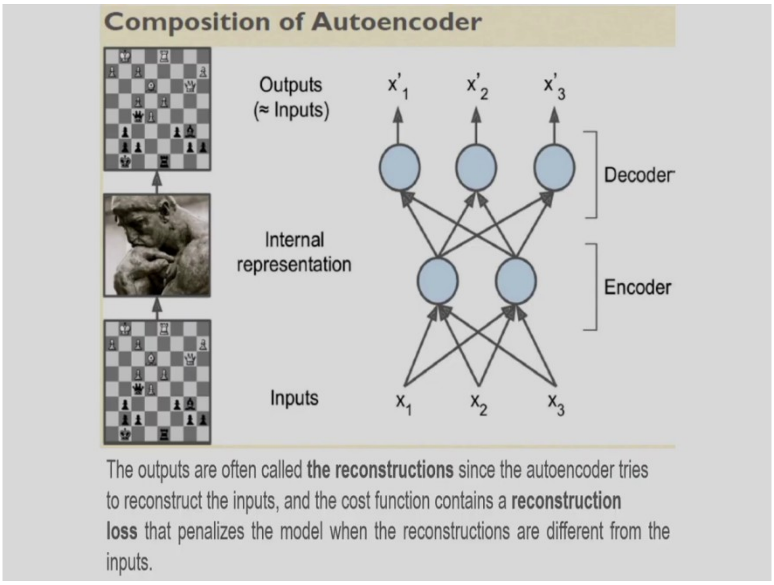


Composition of Autoencoder

There is just one hidden layer composed of two neurons **(the encoder)**, and **one output layer** composed of three neurons **(the decoder)**

And you can see there is just one hidden layer composed of 2 neurons, that is the encoder and one output layer composed of 3 neurons, that is the decoder. So, that is I am showing the encoder, another one is the decoder.

(Refer Slide Time: 67:07)

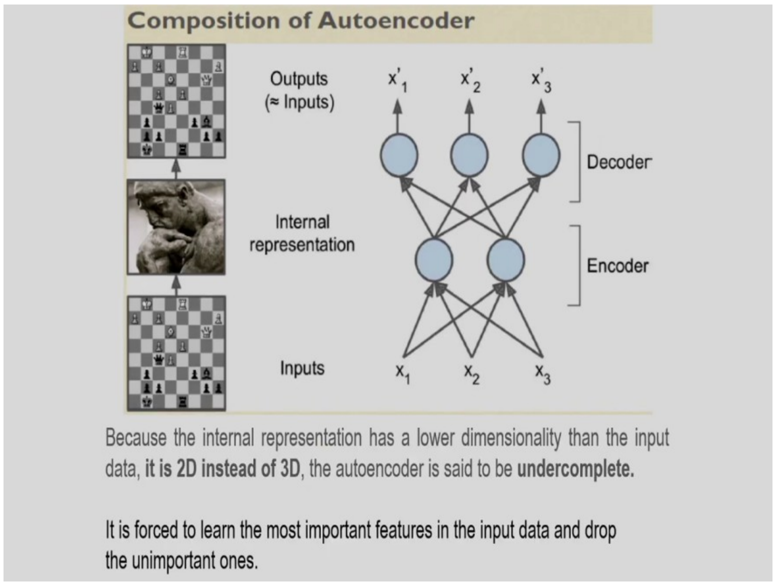


Composition of Autoencoder

The outputs are often called **the reconstructions** since the autoencoder tries to reconstruct the inputs, and the cost function contains a **reconstruction loss** that penalizes the model when the reconstructions are different from the inputs.

The outputs are called the reconstructions, since the auto encoders tries to reconstruct the inputs, and the cost function contains a reconstruction loss, that penalize the model when the reconstructions are different from the inputs. So, because the decoder I am considering, that is used for the reconstruction of the input data, so for is what I am considering I am considering a loss function. If the output reconstructed data is not similar or not same as that of the inputs, then in this case, the loss function will be more, the loss will be more and based on this loss, I can train the auto encoder.

(Refer Slide Time: 67:51)



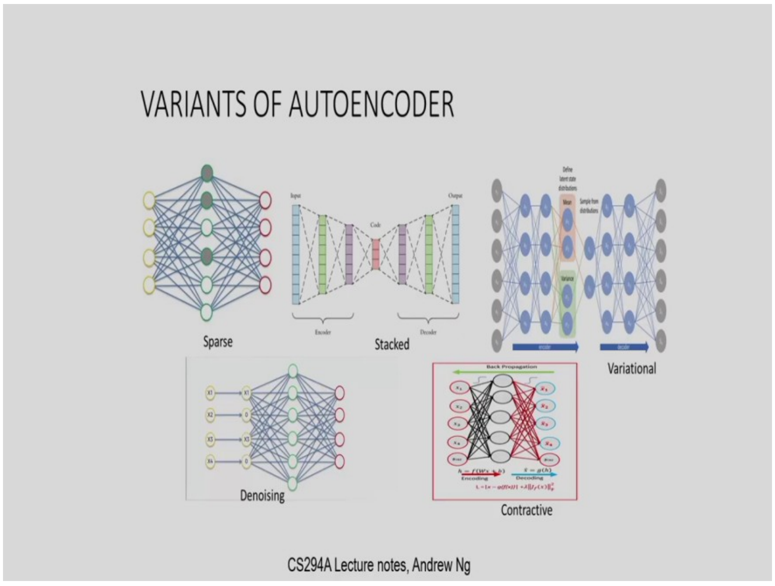
Composition of Autoencoder

Because the internal representation has a lower dimensionality than the input data, **it is 2D instead of 3D**, the autoencoder is said to be **undercomplete.**

It is forced to learn the most important features in the input data and drop the unimportant ones.

And here you can see the internal representation has a lower dimensionality than the input. So, that is why if I consider the bottleneck layer, that is the internal representation, it is 2D instead of 3D, because, I am considering the dimensionality reduction in the bottleneck layer. That is why auto encoder is said to be undercomplete, because I am reducing the dimensionality of the input data. So, that is the concept of the encoder. So, it is forced to learn the most important features in the input data and drop the unimportant ones. So, that is nothing but efficient representation of the input data.

(Refer Slide Time: 68:40)



VARIANTS OF AUTOENCODER

CS294A Lecture notes, Andrew Ng

And there are many variants of auto encoder. I am not going to discuss all these auto encoders, one is the sparse auto encoder, another one is just stacked auto encoder, one is the variational auto encoder, one is the denoising auto encoder and one is the contractive auto encoder. So, if you are interested then you can read some research papers regarding these auto encoders, different types of auto encoders. So, I am not going to discuss about these auto encoders. But the main concept of the auto encoder I have already explained.

So, in this class I discussed the concept of the convolution neural networks, what are the disadvantages of the artificial neural networks. So, how to reduce number of parameters by considering convolution layer, that concept I have discussed. After this I discussed the concept of the pooling layer. And after this I discussed the concept of the fully connected layer. Also, I discussed about activation functions, one is the sigmoid function, another one is the tanh function and another only that ReLU activation function.

After this briefly I discussed the concept of the auto encoder. I am not discussing in detail because it is not possible to discuss all the deep learning techniques in this computer vision course. So, I hope you can understand the basic concept of the deep learning and also the convolution neural networks and the auto encoder. Let me stop here today. Thank you.