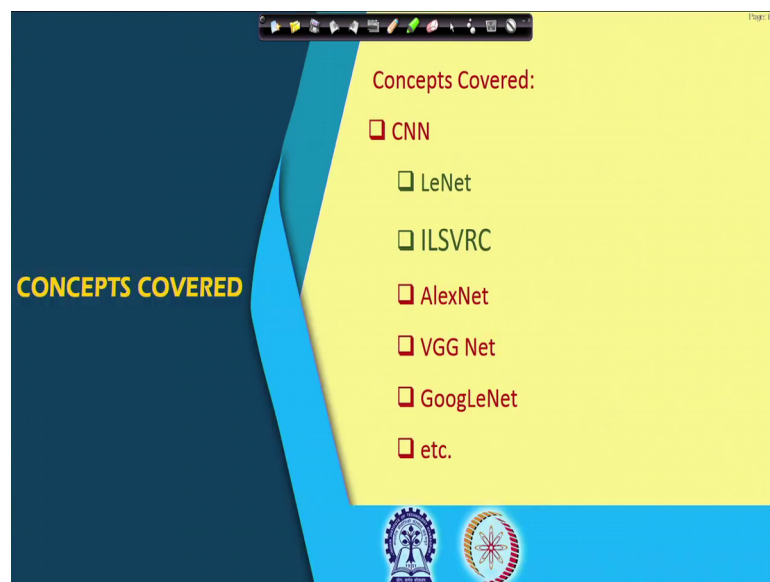


Deep Learning
Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 38
Popular CNN Architecture: AlexNet

Hello, welcome to the NPTEL online certification course on Deep Learning. So, in our previous lecture, we have started discussion on some of the popular convolutional neural network architectures. So, we have talked about LeNet 5 and we said that the LeNet 5 was a network which was used in earlier days by different banks for automation of recognition of the numerical values written in the bank cheques.

(Refer Slide Time: 01:05)



And then we had introduced one challenge which is image net large scale visual recognition challenge and we said that they were to the subsequent networks or the network models that we will discuss are from are basically the champions or very successful submissions to one of these visual recognition challenges. And this visual recognition challenge that actually encourages the researchers to compare and to monitor what is the progress of computer vision research across the globe.

(Refer Slide Time: 01:49)



And we have said that the image net database that contains more than a or few million image databases. And today we are going to discuss about a popular convolutional neural network model which is AlexNet and AlexNet was the winner of 2012 image net large scale visual recognition challenge.

(Refer Slide Time: 02:19)

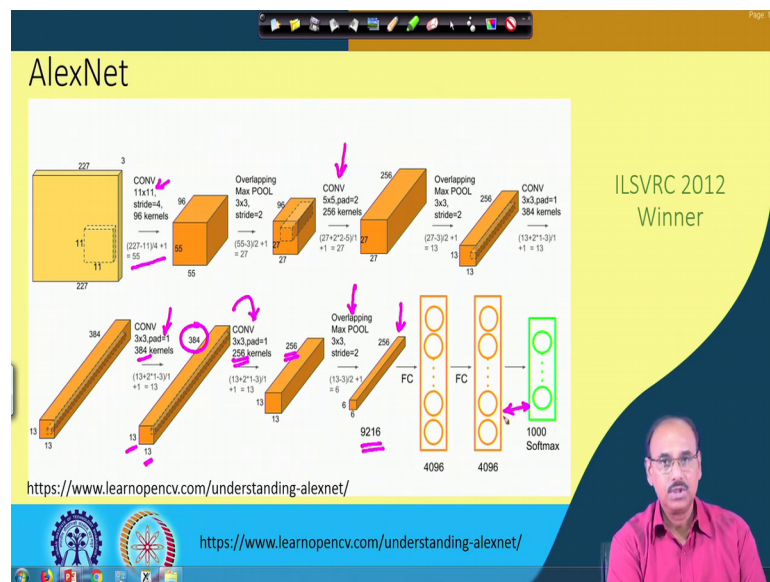


So, before we go to the architecture of AlexNet, let us see that what kind of images does this image net contain or what is the nature of the database. So, here what I have shown is just few samples from the image net database. And you find that there are images from

dogs, there are images from birds, there are images from horses, images from cars and all that. So, we said earlier that this image net database contains 1000 such categories of images and in every category there are thousands of images. So, in total the total number of images this image net database contains is around 15 million.

So, some of those images are used for the training purpose, some of those images are used for testing and validation purpose. So, this image net database is a huge database which is available to the computer vision researchers and they can use this image net database for designing, training and validation of the deep neural networks that the researchers design.

(Refer Slide Time: 03:37)



So, now let us come to what is this AlexNet, so as shown here the functional diagram of AlexNet network or convolutional network. You find that the AlexNet contains a number of convolutional layers it contains a number of pooling layers. So, pooling which is used in AlexNet is actually max pool operations. It also contains a number of fully connected nets, and the output layer which is a 1000 channel softmax layer. So, why this 1000 channels of max of max layer, we have said that the AlexNet database or the image net database, it contains 1000 different categories of images.

So, at the final output, we have one node for each of these 1000 categories. So, as there are a 1000 different categories of images. So, we have 1000 different nodes in the output layer, and this output layer is actually softmax layer. So, you find that the input to the

AlexNet is an RGB image. So, it is having three different channels, the red channel, green channel and blue channel.

Then the first convolution layer in the AlexNet that has convolution kernels of size 11 by 11. So, here you can find that you have convolution kernels, what the kernel size here 11 by 11 and which tied is going to 4 and it has 96 different kernels. So, as a result from the output the first convolutional layer output gives you 96 different channels because there are 96 different kernels. So, it contains so the output contains 96 different feature maps.

And each of the feature map contains features of size 55 by 55 and that comes from this calculation as every input is of size 227 by 227 and you have convolution kernels of size 11 by 11. So, as we said before in our previous class that I can compute what will be the feature size. So, the feature size will be less than be $227 - 227$ minus 11 divided by 4 plus 4 is the stride plus 1 which gives you 55. So, every feature map after the first convolution layer is of size 55 by 55. Then after this convolution layer you have the overlapping max pool layer, where the max pool max pooling is done over an window of 3 by 3 and stride is equal to 2.

So, here you find that because our max pooling window is of size 3 by 3, but stride is 2; that means, the max pooling will be done over overlapped windows. And after max pooling, you reduce the size of every feature map to size 27 by 27 and the number of feature channels that remains 96. Then you have the second convolution layer where the convolution kernel size is 5 by 5. And in this case you use a padding, so that the output of the convolution layer remains same as the input feature size.

So, the size of the feature maps generated by the second convolutional layer remains same as the input feature map size. And the number of kernels in this case is 256, so that means, from this convolution layer output you get 256 different channels or different feature maps. And every feature map is of size 27 by 27.

Then again you have an overlapping max pool layer where max pooling is again done over an window of size 3 by 3 and stride is equal to 2. So, again that means that max pooling is done over overlapping windows. And output of this becomes your 13 by 13 feature maps and the number of channels you have 256 because to max pooling you are not reducing the number of channels. After this you have three consecutive convolution layers.

So, the first convolution layer is having kernel size of 3 by 3 with padding equal to 1, and there are 384 kernels, so that gives you 13 by 13 feature maps and there are 384 feature maps which passes through the next convolution layer again the completion kernel size is 13 by 13 with padding equal to 1.

And this has 384 number of kernels, that means, output of this convolution layer will have again 384 number of channels on 384 number of feature maps. And every feature map is of size 13 by 13. It is because you find that you have given a padding equal to 1 for a 3 by 3 kernel size and that is the reason that your feature map size, the size of every feature map at the output of this convolution layer is remaining same as the size of the feature maps which are inputted to this convolution layer.

This again passes through another convolution layer where the convolution kernel size is again 3 by 3 padding equal to 1 that means output of this convolution layer will generate the feature maps, where the feature map size will remain the same that is 13 by 13. But in this case over here what AlexNet uses is 256 kennels, so that means, at the input of this convolution layer there are 384 channels and this 384 channels now gets converted to 256 channels or it generates 256 feature maps. And every feature map is of size 13 by 13.

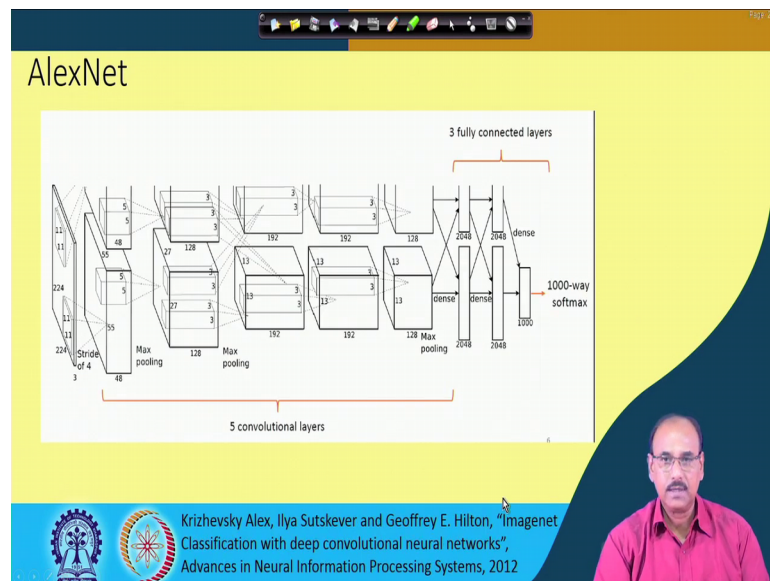
Followed by this is then next overlapping max pool layer, again the max pooling is done over window of 3 by 3 with stride equal to 2 and that gives you the output feature map. And the number of channels means the same which is 256 and the size of every feature map is 6 by 6. After this what you have is fully connected layers or which are same as your multilayer perceptrons that we have discussed earlier.

So, the first two fully connected layers have 4096 nodes each. So, you find that after output of this max pool layer we have 6 into 6 into 256 that is 9216 number of nodes or number of features ok. And each of them is connected to each of the nodes in this first fully connected layer. So, the number of connections or the number of parameters that we will have in this case is 9216 into 4096. And then from this first fully connected convolution layer every node of this first fully connected layer provides input to every node in the second fully connected layer.

So, here you have number of connections which are 4096 by 4096 because the number of nodes in the second fully connected layer is also 4096. And then we have the final layer

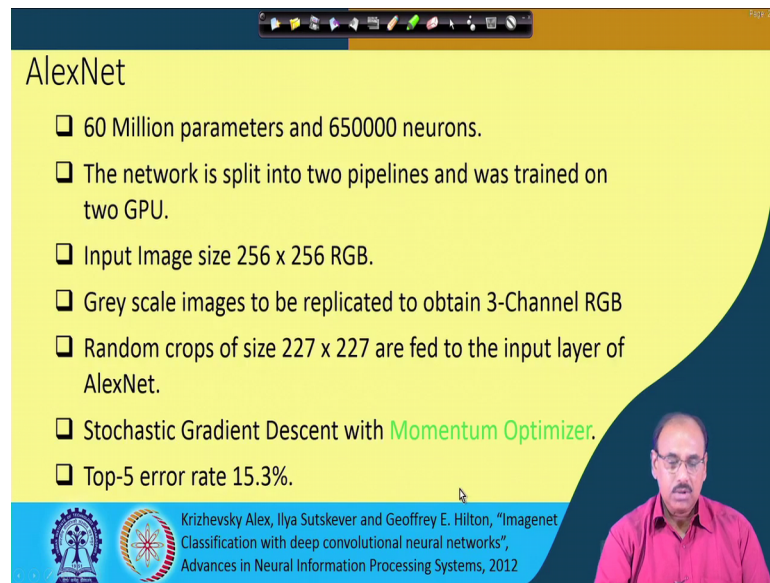
which is the output layer having 1000 softmax channels. So, this is a softmax layer. And the number of connections from this fully connected layer to this output layer that is again 4096 into 1000. So, this is overall the architecture of the AlexNet or the functional diagram of the AlexNet. And if you look at the architecture, the architecture looks something like this.

(Refer Slide Time: 12:11)



So, you find that the inter AlexNet is actually implemented in two channels ok. Half of the network is put in one channel and half of the network is put in the second channel. And because they are in two different channels, so that made it possible to train this network on two GPU cards.

(Refer Slide Time: 12:37)



AlexNet

- ❑ 60 Million parameters and 650000 neurons.
- ❑ The network is split into two pipelines and was trained on two GPU.
- ❑ Input Image size 256 x 256 RGB.
- ❑ Grey scale images to be replicated to obtain 3-Channel RGB
- ❑ Random crops of size 227 x 227 are fed to the input layer of AlexNet.
- ❑ Stochastic Gradient Descent with **Momentum Optimizer**.
- ❑ Top-5 error rate 15.3%.

Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, "Imagenet Classification with deep convolutional neural networks", Advances in Neural Information Processing Systems, 2012

So, coming to some specific features of this AlexNet, you will find that this AlexNet contains more than 60 million parameters and it has around 650000 neurons. And as we have just shown, the network is split into two pipelines which makes it possible to train the network on two GPUs. Of course, even then it took almost one week to train this network fully. The input image size actually is 256 by 256 in 3 channels that is red green and blue channels. And because AlexNet takes the color input in three different channels red green and blue.

So, even if we input a grey level image which is having just one channel, so that grey level images are to be replicated into 3-channels RGB, so that they can be accepted by AlexNet. And to fit the input as we have seen that the first over here the input to the AlexNet is 127 by 127 pixels, every channel is 127 by 127 pixels. So, when you fed the input for training purpose though our input actual input is 256 by 256 RGB image, but each such images RGB images were randomly cropped into size of 127 by 127 and they are fed to the input layer of the AlexNet.

For training of the AlexNet, the algorithm which is used and that is the algorithm which is used for by all the neural networks or all the deep neural networks which is back propagation learning as we said earlier and this back propagation learning actually implements stochastic gradient descent operation. And to make this gradient descent algorithm to be more efficient there is an optimizer which is called momentum optimizer.

So, this AlexNet training of the AlexNet uses this momentum optimizer techniques. So, it is stochastic gradient with momentum optimizer. And as you know that the stochastic gradient or the back propagation learning itself is an optimization technique. So, what does it try to optimize? It optimizes the loss at the output of the network that we have seen earlier right. And by optimization or minimization of the loss, you tend the network for your given applications.

And this momentum optimizer is another optimization technique, there are other optimization techniques also we will see that after few lectures. So, all these optimization techniques basically optimize this back propagation of gradient descent algorithm itself with the aim that your gradient descent algorithm becomes more efficient. And once the network was fully trained, the network gave a top 5 error rate of around 15.3 percent.

So, what is meant by this top-5 error rate. You find that you remember through our earlier discussion that as the AlexNet contains 1000 output nodes, every output corresponding to one of the image categories. So, when you test with a given input for which is the category is unknown, what the network will do, at every output node of the network that is the softmax layer, you get a score of the input that you are provided to the corresponding category, that means, every output or the score output that you get from the output softmax layer every output node gives you a score function to the corresponding category.

Now, if you arrange this scores, scores into say descending order of magnitude, then the first output becomes the score for the most preferred category, the second output score gives you the score for the second preferred category and so on. So, the top-5 error rate says that if I take the top-5 such categories to which the network has classified the input that we have provided, your actual class should be in one of those top 5 if it is not in any of those top-5 categories that indicates an error, so that is what is your top-5 error.

And in this case this AlexNet gives you a top-5 error rate of 15.3 percent. So, in all our subsequent discussions we will talk about top n error rates, so that should be cleared what is meant by your top n error rate or top-5 error rate.

(Refer Slide Time: 17:49)

The slide is titled "Vanishing Gradient Problem" and contains the following text:

- ❑ Uses ReLU activation instead of sigmoidal function.
- ❑ ReLU output is unbounded- uses Local Response Normalization (LRN).
- ❑ LRN carries out a normalization amplifying the excited neuron while dampening the surrounding neurons at the same time in a local neighbourhood.
- ❑ Encourage *Lateral Inhibition*: concept in neuro biology that indicates capacity of a neuron to reduce activity of its neighbours.

Hand-drawn diagrams in pink ink illustrate the ReLU function and LRN. The ReLU diagram shows a horizontal line at zero for negative inputs and a diagonal line for positive inputs. The LRN diagram shows a central peak with a shaded area around it, representing normalization.

At the bottom of the slide, there is a video inset of a man in a red shirt speaking. Below the video, the text reads: "Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, 'Imagenet Classification with deep convolutional neural networks', Advances in Neural Information Processing Systems, 2012".

So, these are the features of the AlexNet. Then earlier we also discussed about a problem called vanishing gradient problem. And we have seen that if your non-linear activation function is a sigmoidal function or a tan hyperbolic function, then it gives at risk of vanishing gradient, that means, in some cases as you are training the network with the gradient descent procedure, vanishing gradient means that the gradient of the error function may become too small, so that using that gradient, when you try to update the network parameters, the update becomes almost negligible because, the gradient itself is very small that is what is your vanishing gradient problem.

And why do you get this the vanishing gradient problem, it is because if you remember you remember that our sigmoid function was something like this all right. So, it gets saturated at 1, where your input parameter x becomes very high. Similarly, it saturates to 0, when the input parameter is very low on the negative side right, because over here your output saturates to 0, saturates to 1 and over here output saturates to 0.

So, if your argument is some somewhere over here, you try to take the gradient of the output with respect to argument, this gradient is almost 0. So, the same is true if your non-linear activation is sigmoidal function or the non-linear activation is tan hyperbolic function. Whereas, in case of ReLU, you remember from our earlier discussions that ReLU activation function is something like this, which is rectified linear limit.

So, on this side, your gradient never vanishes the gradient is almost count as a constant, so that is the advantage that you get using ReLU as an non-linear activation function. But it has other problem that unlike in case of the sigmoidal activation or tan hyperbolic activation where the activation output is limited is bounded; in case of ReLU the output is unbounded, as x increases the non-linear output the activation function also keeps on increasing, so that is the problem with the value if when ReLU is used as a non-linear activation function.

So, to avoid this problem, what AlexNet does is, it tries to normalize the output through a process which is known as local response normalization. So, this LRN or Local Response Normalization, it carries out carries out a normalization operation by amplifying the excited neurons while dampening the surrounding neurons at the same time in a local neighborhood.

And this particular operation is encouraged by from a phenomena known as lateral in inhibition concept in neurobiology which actually indicates the capacity of a neuron to reduce the activity of its neighbors. So, this AlexNet in order to deal with the problem of unbounded output, it goes for a normalization of the output before applying this non-linear activation function. So, that is how it avoids the unbounded problem or and at the same time it avoids the vanishing gradient problem.

(Refer Slide Time: 21:41)

Local Response Normalization (Inter-Channel)

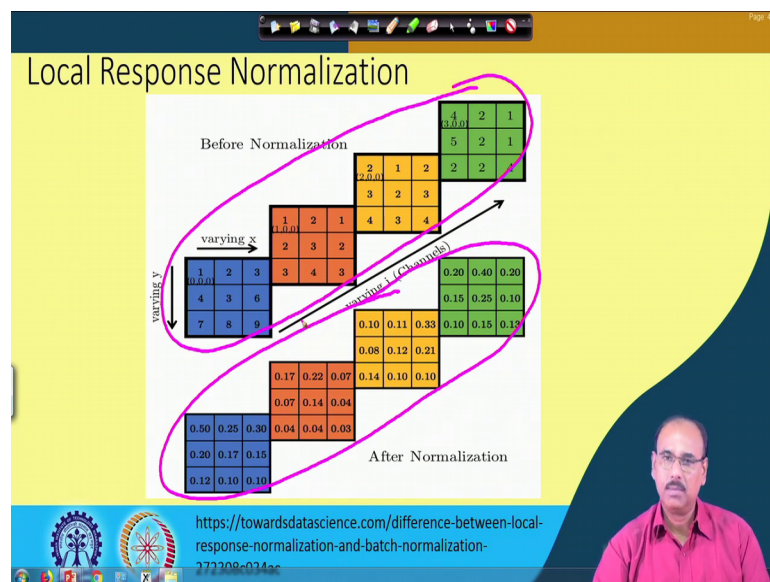
$$b_{x,y}^i = \frac{a_{x,y}^i}{\left(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a_{x,y}^j)^2 \right)^\beta}$$

So, this local response for normalization it can be done across the channels, also it can be done within a channel. So, when you do this local response normalization across the channel, this is what is known as inter channel normalization. And in case of inter channel normalization, the operation is done in this way suppose $i \times y$ is the response at location $x \ y$ in your feature map.

And the index i indicates the i th channel. So, $b_{i \ x \ y}$ is the output at location $x \ y$ in the i th channel and $a_{i \ x \ y}$ is the original value at location $x \ y$ in the i th channel. So, the way the normalization is done is by this, you normalize this $a_{i \ x \ y}$ by a factor which is given by k plus alpha times sum of $a_{j \ x \ y}$ where this j varies within neighboring channels.

So, j varies from maximum of 0 to i minus n by 2 to minimum of n minus 1 and i plus n by 2, that means, you take n by 2 number of channels before and n by 2 number of channels after. This max and min operations with 0 and n minus 1 as arguments is put to take care of the first channel feature channel and the last feature channel if there are total n number of feature channels. So, this is what is your local response normalization where the normalization is carried over subsequent channels.

(Refer Slide Time: 23:51)



So, the nature of output that you get using this local response normalization is something like this. So, here these are the feature values of different channels before normalization, and these are the feature values that you get after the normalization across the channels is carried out. So, here you find that when you use this normalization, your output actually

becomes minded, in fact, bounded in fact every output you find that after normalization becomes less than 1. So, this is one type of normalization, local response normalization which is inter channel LRN which is which can be used.

And the other kind of normalization that can be used is what is intra channel normalization. For the intra channel normalization is carried over the features within the same channel. So, here again if I want to apply this intra channel normalization in channel i , again you find that $b_{i \ x \ y}$ is the feature value after normalization at location $x \ y$ in the i th channel and $a_{i \ x \ y}$ is the feature value at the same location $x \ y$ in the i th channel before normalization.

So, here what you do is you normalize this feature value by again affected k plus alpha times take the summation of square of the feature values within neighborhood of $x \ y$. And again here the neighborhood size is say n by n by n and again max and min operations are performed to take care of the boundary features or features which are at the boundary of the feature frames ok.

Why k is used is to avoid division by 0. In case this particular summation term within this factor this becomes 0, k is a very small positive value, so k takes care of the fact that I do not encounter any situation of a division by 0. And you find that this k and alpha they are actually hyper parameters, so which has to be chosen by the designer. So, this is another kind of normalization which is intra channel normalization can be that can be performed for this local response normalization operation.

(Refer Slide Time: 26:25)

Local Response Normalization

Before Normalization					After Normalization				
1	3	2	4	2	0.03	0.07	0.03	0.07	0.04
2	4	3	5	1	0.05	0.07	0.03	0.07	0.02
2	3	2	1	3	0.05	0.04	0.02	0.01	0.06
1	3	5	3	2	0.03	0.04	0.05	0.03	0.04
2	3	4	5	2	0.09	0.05	0.04	0.06	0.05

<https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac>

And here again you find that the after normalization the kind of output that you get. So, here on the left hand side you have the feature values before normalization and on the right hand side, we have the feature values after normalization. And here again find you find that after normalization the feature values have been bounded ok. So, this is how using the local response normalization techniques, the AlexNet could take care of the unbounded output problem of rectified non- a rectified linear unit which is the non-linear activation function which is used in AlexNet.

(Refer Slide Time: 27:05)

Reducing Overfitting

- Train the network with different variants of the same image helps avoiding overfitting.
 - ❖ Generate additional data from existing data (Augmentation).
 - ❖ Data augmentation by mirroring.
 - ❖ Data Augmentation by random crops.
- Dropout Regularization.

Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, "ImageNet Classification with deep convolutional neural networks", Advances in Neural Information Processing Systems, 2012

The other problem in AlexNet you find is that as we said that the number of parameters in AlexNet is more than 60 million. Obviously, with such a huge number of parameters, there is always a risk that while you train this network, there will be a problem of over fitting. So, what is this problem of over fitting, over fitting means that the network will try to learn or it will perform very well on your training data set or it will try to memorize the training data set. But in the process it will not be able to code the general properties or the features of the input data.

So, as a result your training performance may be very high, whereas the performance during training during training or the generalization error may not be acceptable. So, to avoid this over fitting problem or to reduce this over fitting problem, during training additional data or augmented data training data was generated from the existing data. And this augmentation was done by data mirroring and the augmentation was also done by taking random crops from the input data.

So, you have different multiples of the input data right by which the networks was trained, so that the network can code the features or the descriptors the general descriptors of the input data. The other way this overfitting problem was reduced is by taking a regularization approach where the regularization was a drop out regularization.

(Refer Slide Time: 29:03)

Dropout

- Regularization Technique proposed by Srivastava et. al. in 2014.
- During training randomly selected neurons are dropped from the network (with probability 0.5) temporarily .
- Their activations are not passed to the downstream neurons in the forward pass.
- In the backward pass weight updates are not applied to these neurons.

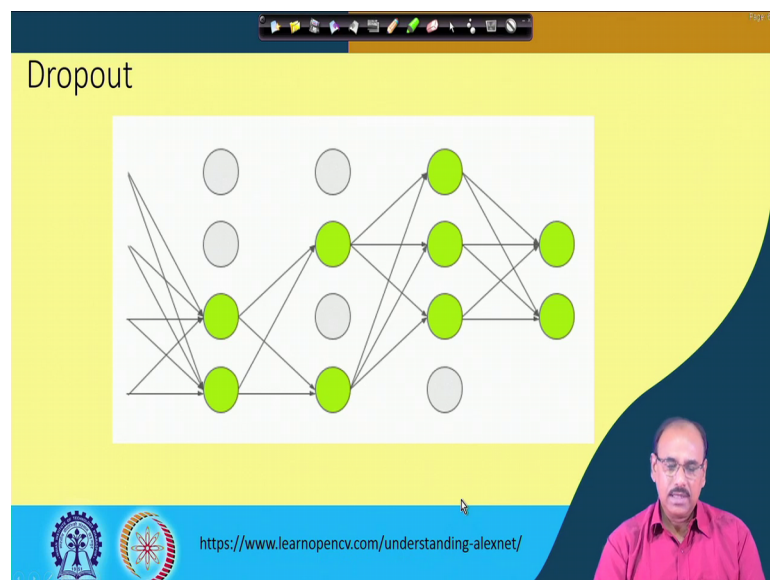
Srivastava Nitish et. al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" Journal of Machine Learning Research 15 (2014), 1929-1958

So, let us see what is this drop out regularization. This dropout regularization is a technique which was proposed by Srivastava et al in the year 2014. So, what is done in

this drop out regularization is that during training randomly selected neurons or randomly selected nodes which are selected with a probability of say 0.5, weight drop from the network. So, the probability that a node will exist in the network is 0.5 similarly the probability that it will be dropped out is also 0.5.

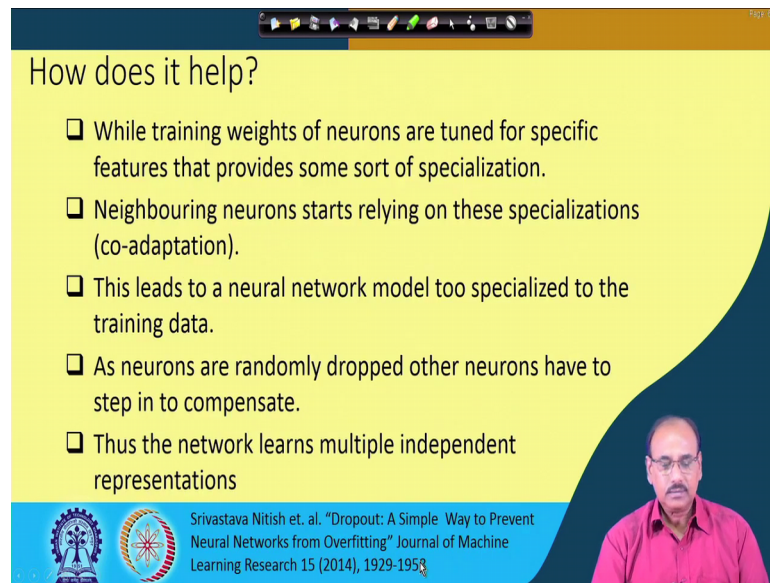
So, what does this drop out means that a node which is dropped out that node will not pass its output to the subsequent nodes or the nodes in the subsequent layers downstream. And at the same time, during backward pass for those nodes no updates of the input weights will also be taken place. So, that means, as if you are simply removing that particular node and the associated connections from the network during the training process.

(Refer Slide Time: 30:13)



So, how do you gain by this dropout?

(Refer Slide Time: 30:17)



How does it help?

- ❑ While training weights of neurons are tuned for specific features that provides some sort of specialization.
- ❑ Neighbouring neurons starts relying on these specializations (co-adaptation).
- ❑ This leads to a neural network model too specialized to the training data.
- ❑ As neurons are randomly dropped other neurons have to step in to compensate.
- ❑ Thus the network learns multiple independent representations

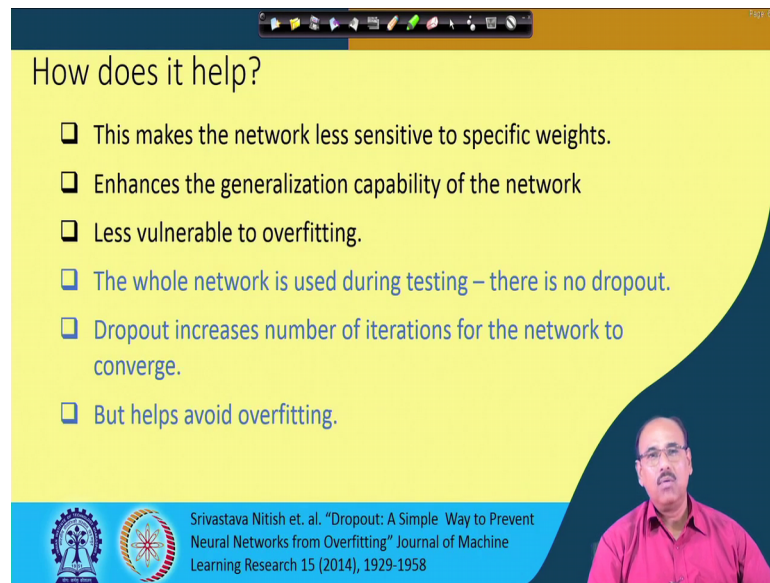
Srivastava Nitish et. al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" Journal of Machine Learning Research 15 (2014), 1929-1958

So, here you said that while training the weights of the neuron are tuned to some specific features that gives some sort of specialization to the neurons. And the neighboring neurons also starts relying on these specializations which is known as co-adaptation and that leads to a problem that neural network becomes too specialized to the training data.

And through dropout what you are doing is as you are removing some of the neurons at a random that means, we have to have other neurons to take over the charges of this dropped out neurons. So, as a result the neural network learns multiple independent representations and that is how it avoids the problem of overfitting and your neural network becomes more general.

So, this previous diagram shows that some of the neurons which had dropped out during the training process. So, the neurons which are white, they are dropped out at the during the training process. So, in different training epochs that neurons will be dropped out at random. So, it is not that the same neuron will be dropped out every time, so that is what is your probability that is decided by the (Refer Time: 31:39) of probability.

(Refer Slide Time: 31:41)



How does it help?

- ❑ This makes the network less sensitive to specific weights.
- ❑ Enhances the generalization capability of the network
- ❑ Less vulnerable to overfitting.
- ❑ The whole network is used during testing – there is no dropout.
- ❑ Dropout increases number of iterations for the network to converge.
- ❑ But helps avoid overfitting.

Srivastava Nitish et. al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" Journal of Machine Learning Research 15 (2014), 1929-1958

So, as a result what you gain is something like this the network becomes less sensitive to the specific weights and it enhances the generalization capability of the network. And as a result it becomes less vulnerable to the overfitting problem ok. So, this is what is done only during the training process. But during the testing the intra network is used that is no dropout at the time of testing. Obviously, while training as we are dropping out some of the neurons.

So, the number of iterations which will be required to train the network will be more, that means, your training time increases. But at the same time on the positive side what you have is that it helps overfitting problem, so that your network becomes more general, and it captures the salient features of the input data, it does not simply memorize the input data and that gives better performance the network gives better performance with such a type of training.

So, we will stop here today. In the next class, we will try to go through some other popular CNN architectures. And we will also discuss about some challenges that you face while training of the deep neural networks.

Thank you.