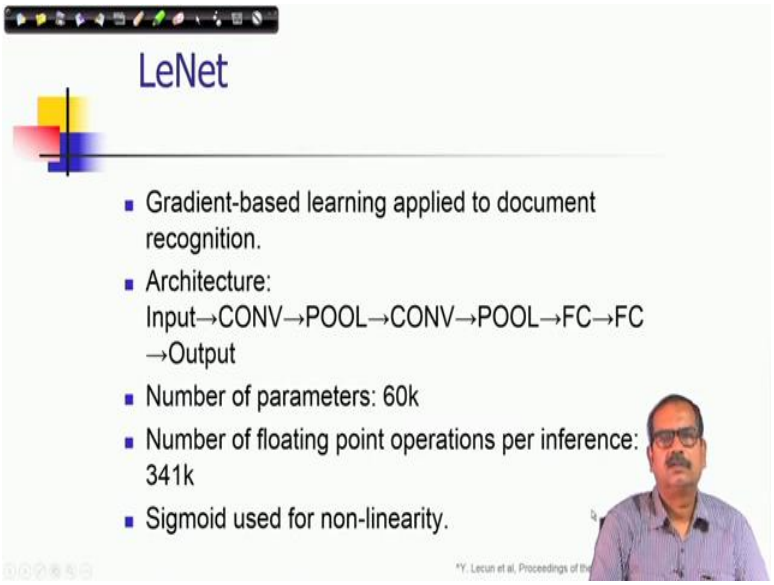
Computer Vision
Prof. Jayanta Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 57
Deep Neural Architecture and Applications Part – III

So, we are discussing about convolutional neural network and in this lecture I will provide some examples of CNN architectures or Convolutional Neural Networks which are used for different particularly for image and object classification.

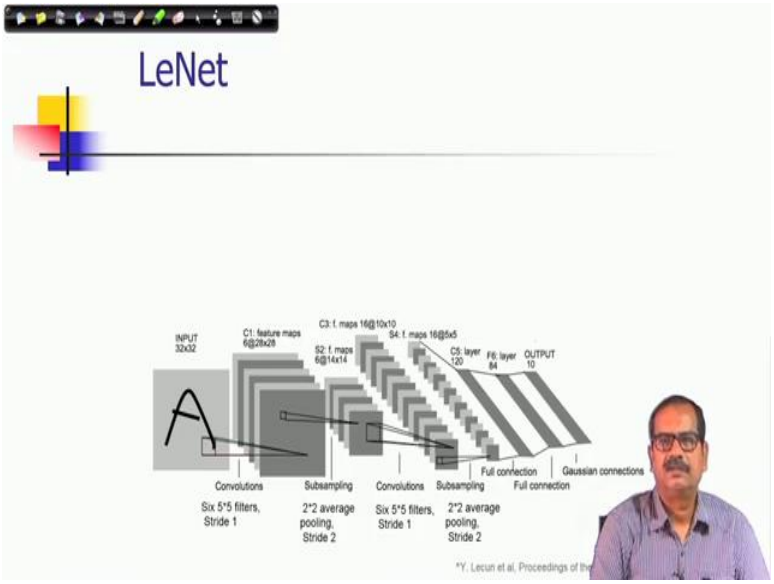
(Refer Slide Time: 00:32)



So, the first; the very first network which was introduced for as in the applications of image classifications in fact, in document image processing, document image processing or character recognition this is called LeNet. And, it is a gradient based learning applied to document recognition as you can see and this architecture uses two convolutional layers, there are two pooling layers, two fully connected layers.

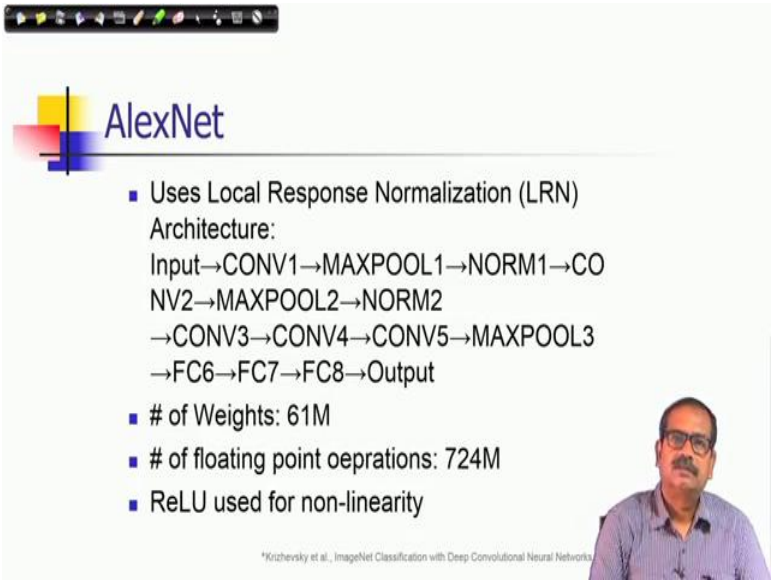
And, it was reported in 1998, it was quite long back and it is a quite a pioneering work in that sense. And, its number of parameters it was around 60,000 and number of floating point operations per inference was 341 k; that means, about 341000 and it uses sigmoid as a nonlinearity function.

(Refer Slide Time: 01:38)



So, this is a diagram which shows schematically how this architecture looks like. So, you can see that there are convolutional layers and then you have the sub sampling which means it has the pooling layers and as you discuss there are two convolutional layers, two pooling layers and there are two FC layers and finally, you get the output, out of this process.

(Refer Slide Time: 02:08)

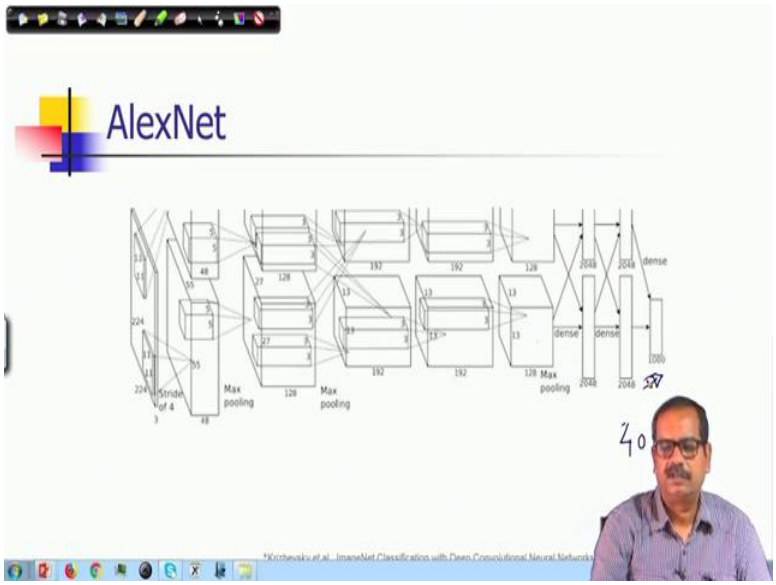


Then in 2012 in fact, the another this network is used and this has provided very good result for the image net classification and this is called AlexNet. So, in this network it

uses a normalization of the responses that local response normalization and its architecture it consists of 5 convolutional layers and you can have it has also you know 3 pooling layers. And, you can see that the first pooling layers are adjacent to their convolutional layers, but the last pooling layers it has it operates after processing through after getting the process data through third, fourth and fifth convolutional layers; that is what it is designed, that is how it is designed.

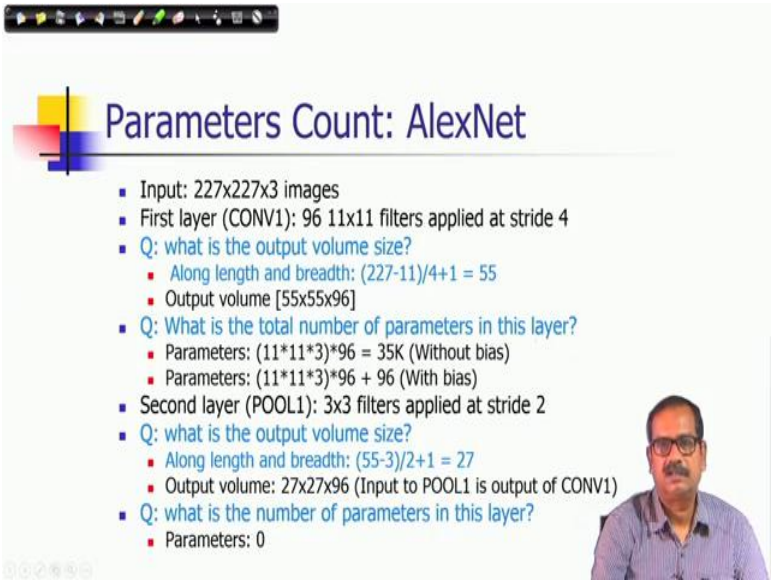
And, there are normalization operations after first and second convolution pooling layers and there are three fully connected layers from which the output is and then finally, there is an output layer. So, the number of weights in this case was 61 million and number of floating point operations is 724 million. So, it is a very large network compared to very primitive network of LeNet earlier what has been reported and here they have used ReLu as the non-linearity, that is one of the major change of non-linear functions in neural network.

(Refer Slide Time: 03:42)



So, a schematic diagram of AlexNet has been shown here, it also shows some of the information related to the size of the input say; it processes 224×224 inputs. It takes filter sizes of say 11×11 and then it uses max pooling of 5×5. Finally, you can see that the feature representation what you were waiting 4096 is the dimension of the feature representation and that is finally, classified for the 1000 notes. So, this is how this classifier is used.

(Refer Slide Time: 04:28)



So, if you would like to count the parameters of AlexNet let us consider how the parameters could be counted. See consider the input image size of $227 \times 227 \times 3$ images that is the input size, it was not 224. It is $227 \times 227 \times 3$ and the first layer convolution layer there are 96 filters each of size 11×11 and it is applied with a stride 4. So, the gap between two samples in the input pixels so, that stride should be 4. it should leave 3 samples and then the fourth sample it should operate with the convolution operation.

So, what should be the output volume size? So, as we know that it is related to with their dimension of input width and height and also the filter size. So, along length and breadth you can see that it has to be the $227 - 11$, 11 is the filter size and divided by 4. So, that is how the dimension and then you add with 1; so, because of the central you can add with 1. , this is the expression we have discussed earlier also. So, the 55 would be the size of the length and size of the breadth and your depth is the number of filters that is 96. So, the size should be $55 \times 55 \times 96$ so, that is that should be the output volume; it is $55 \times 55 \times 96$.

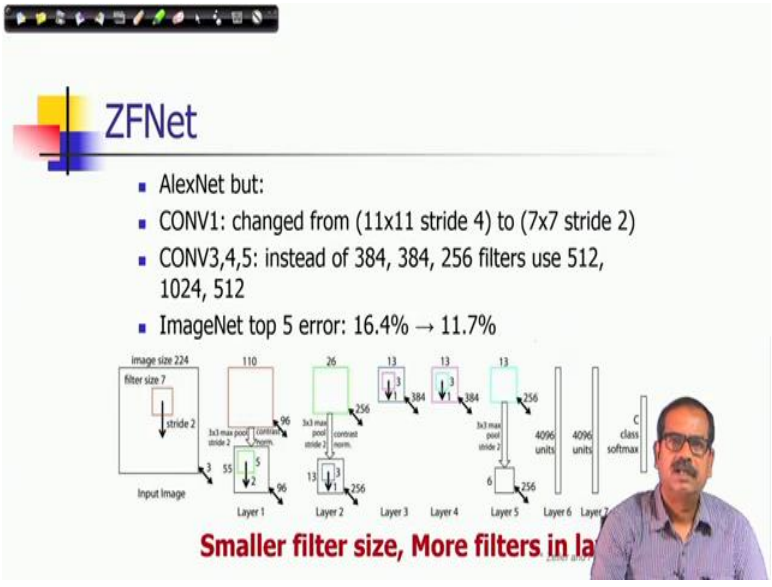
Similarly, what should be the number of parameters in this layer? So, the number of parameters involves the filter weights and also biases. So, since there are 11×11 filters that is a size $11 \times 11 \times 3$ there is a filter size. So, number of parameters will be $11 \times 11 \times 3$ for 1 filter and since there are 96 filters, it should be $96 \times 11 \times 11 \times 3$. There is a

total number of parameters that is without bias and if I consider each should have a bias; so, for each filter there would be one bias. So, you have to add another 96 biases.

So, it should be 90 this number which is around 35 K around 35 K or 35000, there is approximate number and that should be again plus 96. Then consider a second layer that is POOL 1, it has 3×3 filters and which is applied at stride 2. So, what should be the output volume size in this case? So, once again 55 minus filter size 3 divided by the stride 2 and plus 1 I.e., $\frac{55-3}{2}+1$. So, output the length and breadth should be 27. So, output volume would be $27\times27\times96$ because the input depth of the input was 96, it remains the same.

And, that is what is your output volume and, what is the number of parameters in this layer? And, as we mentioned that in pooling layer there is no parameter involved, you do not require any weight or bias. So, number of parameters would be 0. So, in this way you can count parameters by knowing the size of the filters and you can count the you can also determine the size of the output by knowing the size of the filter and size of the input and number of filters that is also important.

(Refer Slide Time: 08:35)

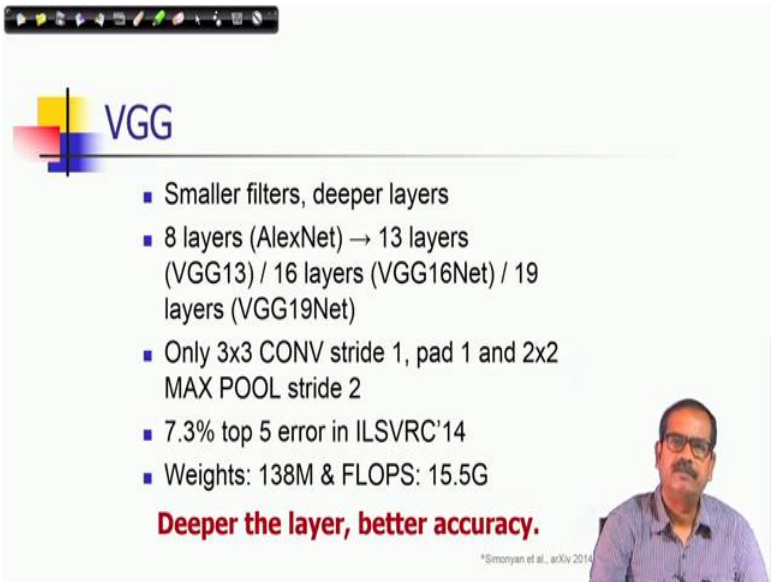


Other network that is also quite distinct that is ZF network, it is I am basically providing you some kind of historical evolution of CNNs. So, ZF network came from Alex network, the changes this modifications. So, what it does? It uses filters of smaller size.

So, earlier it was 11×11 and stride 4, in ZF network they use 7×7 stride 2. And, but what they do also they have used more number of filters say for example, in CONV 3, 4, 5 in AlexNet you have 384, 384 and 256 filters.

In ZF net you have 512, 1024 and 512 and it has improved the accuracy of the image classification for the ImageNet data set. So, this is one that is smaller the filter size it is smaller; in number filter size is smaller, but more filters in layer is used. A schematic diagram is shown here, it shows the size of the different processed output and also the size of filters and other things.

(Refer Slide Time: 09:57)



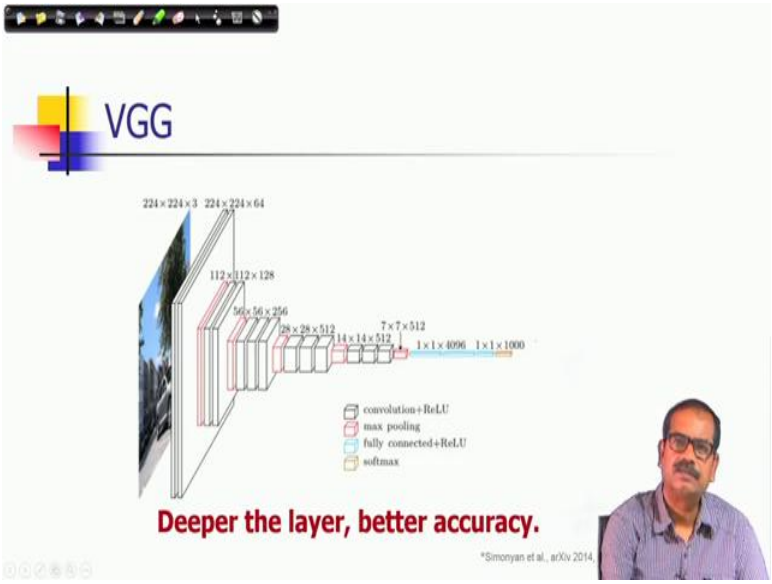
Next you know evolution of this CNN architecture, next stage was VGG and the here in this architecture you have a very, you have a good number of deep layers. So, the idea is that you should have smaller filters, smaller even then the ZF net and you should increase the number of layers. So, in ZF net you have smaller filters more number of filters, it is not only more number of filters also increase the more number of layers. And, by doing that they could increase the, performance of the image classification using the same data set whatever is referring.

So, in this case VGG net it uses different kinds of number of layers in architecture. There are 13 layers which is called VGG 13, there are 16 layers which is 16 or 19 layers which is called VGG 19. And, only 3×3 convolution filters are used with stride 1 pad 1 and

2×2 max pool with stride 2. So, the filters specification is very simple for VGG network. So, here the observation is the deeper the layer better was the accuracy.

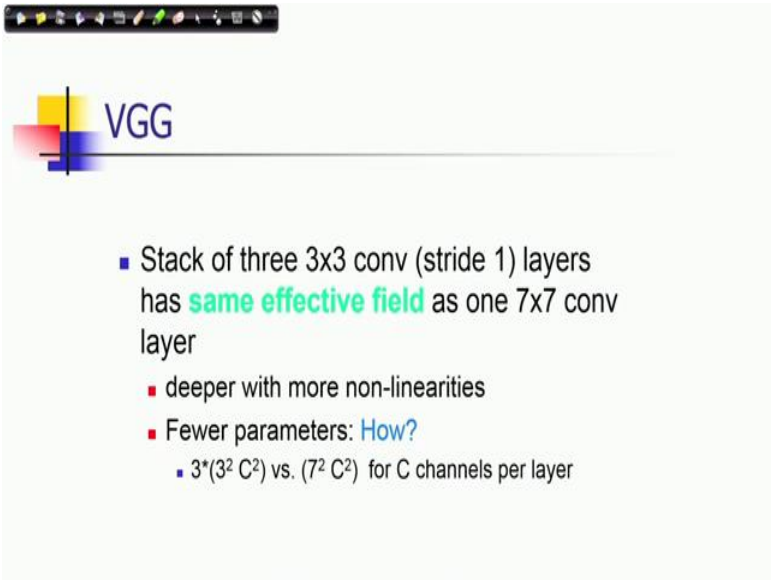
In this particular VGG network, suppose it is I am not sure which number it is in a typical VGG network; it is 138 million number of weights and number of floating point operations for each you know influencing it is about 15.2 billion floating point operations.

(Refer Slide Time: 11:48)



So, one example of VGG network, it has been shown here you can see that typical sizes of input which starts from $224 \times 224 \times 3$ and then it down samples still 7×7. And finally, the feature representation goes towards the $1 \times 1 \times 4096$ and then you use the soft max probabilities, you use the output layer 1000 class classification.

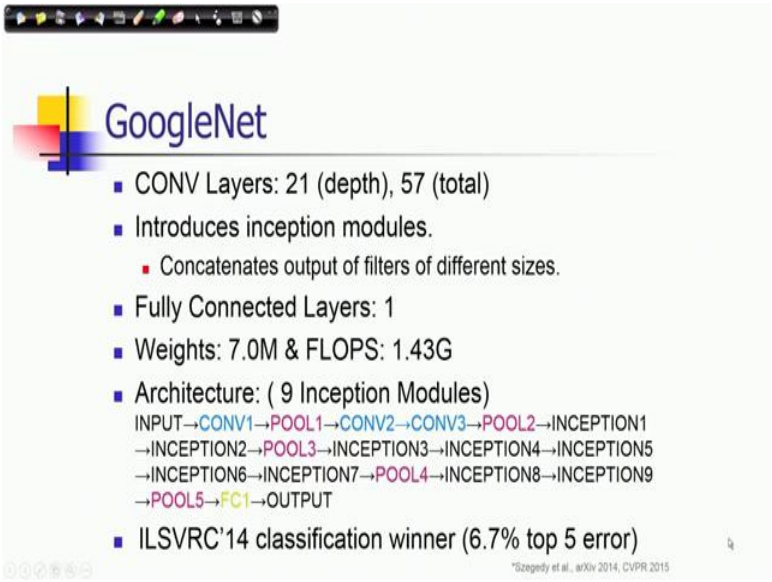
(Refer Slide Time: 12:21)



It has been observed in or it has been found in the VGG network that stack of 3×3 convolutional layers has the same effective field as one 7×7 convolutional layer. So, successively if you use 3×3 convolutional layer let me explain that. So, you have an image. So, apply 3×3 convolution with this image, then you produce another image, then you apply against 3×3 convolution. In this way successively you do it, the effect would be that same as if you have 7×7 convolution at a single stage and so this is one advantage.

So, that is why you have so many number of layers, but you are having the same effect and you have more nonlinearities introduced. And, that would give you the fewer parameters also, because you know if you are using this stack of three then you get fewer parameters. Like for this case you have say $3*(3^2C^2)$ that many number of parameters should be there; if you assume C channels per layer C is a constant whereas, for 7×7 filter it is (7^2C^2). So, that number is large compared to $3*3^2$ which is 27.

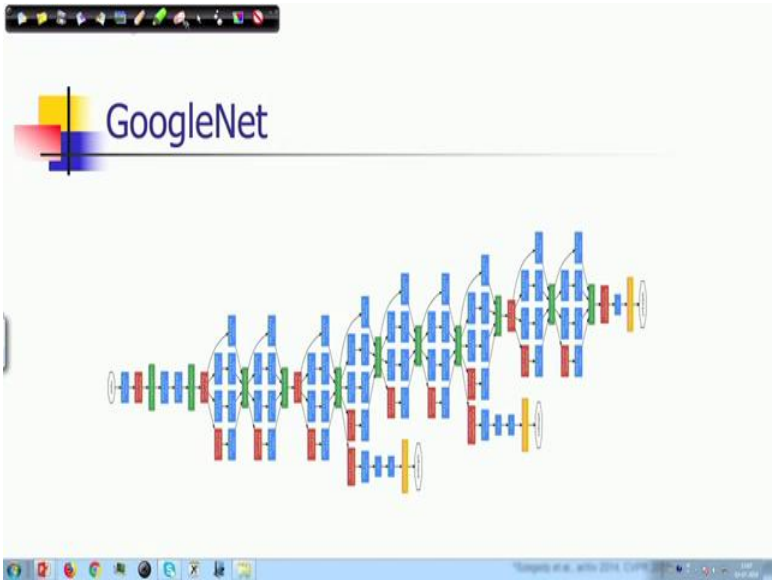
(Refer Slide Time: 13:57)



Then we came across GoogleNet and in GoogleNet you have more number of layers 21 depth and total 57, it introduces inception modules, that is the novelty in GoogleNet modifications. So, what it does instead of using a filter of same size in a particular layer, it concatenates output of filters of different sizes and that increases the power of the model. So, it has only one fully connected layer and number of weight is 7 million.

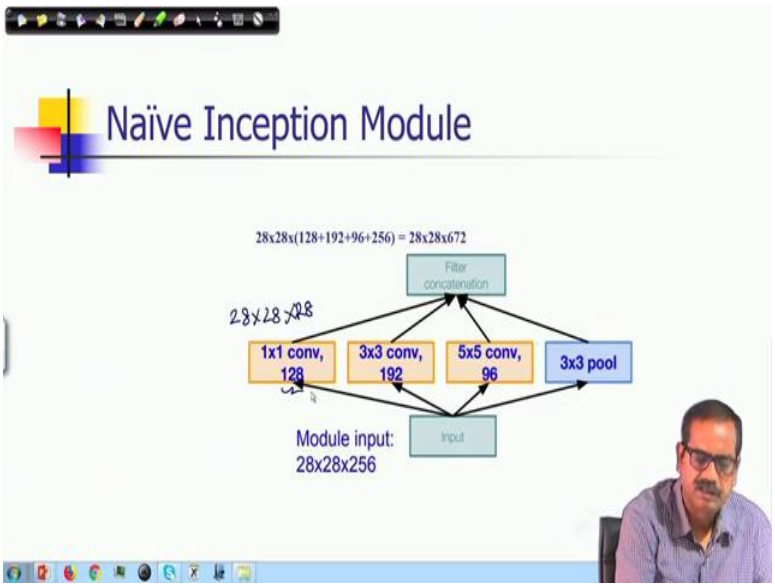
And, number of floating point operation is 1.43 billion and the architecture example architecture there are 9 inception modules as I mentioned. So, there are convolution, 3 convolution layers and there are 2 pooling layers followed by this convolutional layers, but after that you can see it is inception layers again there are pooling of inception layers. So, if I explain, if I give an example; so, GoogleNet also provides a very good performance 6.7 percent top 5 error.

(Refer Slide Time: 15:12)



So, this is a one short diagram for picture of GoogleNet how does it look and you see that these layers are inception layers. So, these layers are inception layers; so, it is using different sizes of filters and then it concatenate this filter.

(Refer Slide Time: 15:39)



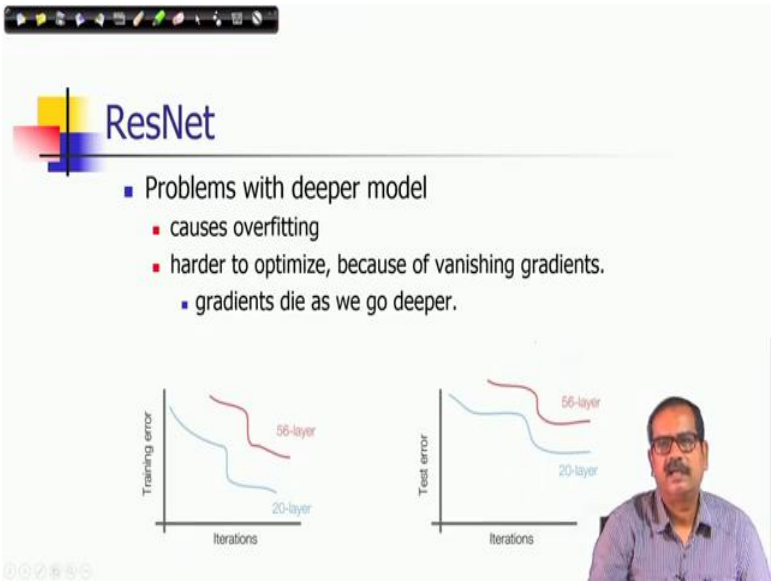
There is a enlarged diagram here. So, here it is shown that from the input there are filters of 1×1 . So, 1×1 means it is filtering along the depth only. So, your filter weights are along depth only, then there are 3×3 convolution filters, 5×5 filters and 3×3 pooling. And, number of filters with different sizes also they vary say 128 1×1 filters 192

3×3 filters 96 5×5 filters. And then of course, 3×3 pooling which does not require any number of filters which should give you the same depth value of the input.

So, finally, you can say that if I use 128 1×1 filters and your output would be $28\times28\times128$ that would be your output. Similarly for 3×3 convolution and here by performing 0 padding, you are keeping the input size same output size same as the input size; so, that is also done in the inception model. So, you have $28\times28\times192$ output from this model. So, this is giving $28\times28\times128$, because there are 128 filters this will give $28\times28\times192$, this will be giving $28\times28\times96$ because there is a 0 padding.

So, you are keeping input and output same size and this will give 3×3 pool, but there is no stride; so, this will also give $28\times28\times256$. So, total input size output size would be say 28 cross 28 cross and there are so many channels. So, this is how the inception model works, it increases the number of channels and sometimes it reduces the number of channels like here instead of 256, the number of channel becomes 128 here.

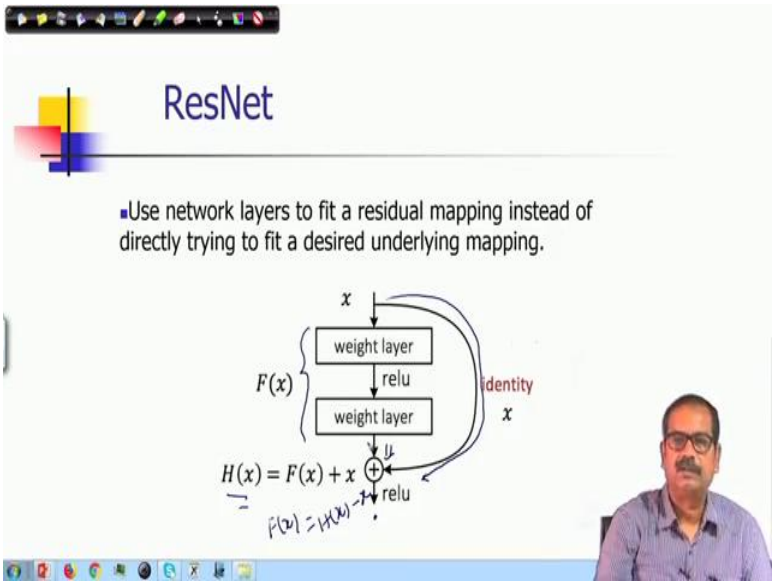
(Refer Slide Time: 17:54)



So, that was the you know modifications done in GoogleNet and it has found to be given good performance. Then the next kind of deeper model is the residual network or ResNet because the problem, if we increase the more number of layers; it has been observed that it causes overfitting and it is harder to optimize because the gradients they get managed. So, the since the managed for the vanishing gradients you cannot run the optimization, it drive soon as we go deeper.

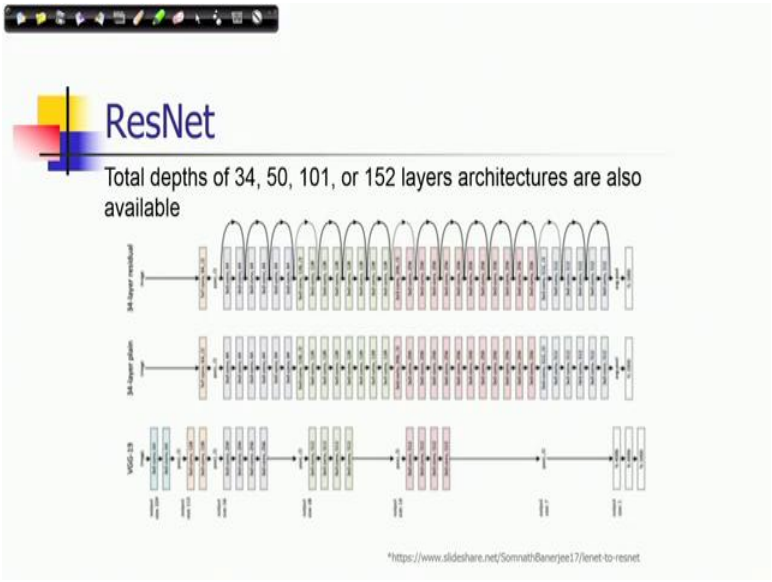
So, this ResNet we will discuss the architecture. So, this is the problem. So, you can have a good training error, but your tester would be quite high, if you increase the number of iterations, if you go by more number of layers. So, this is a normal CNN, this is CNN with less number of layers and this is the CNN with more number of layers. So, you will have this problem that your error will be higher.

(Refer Slide Time: 18:59)



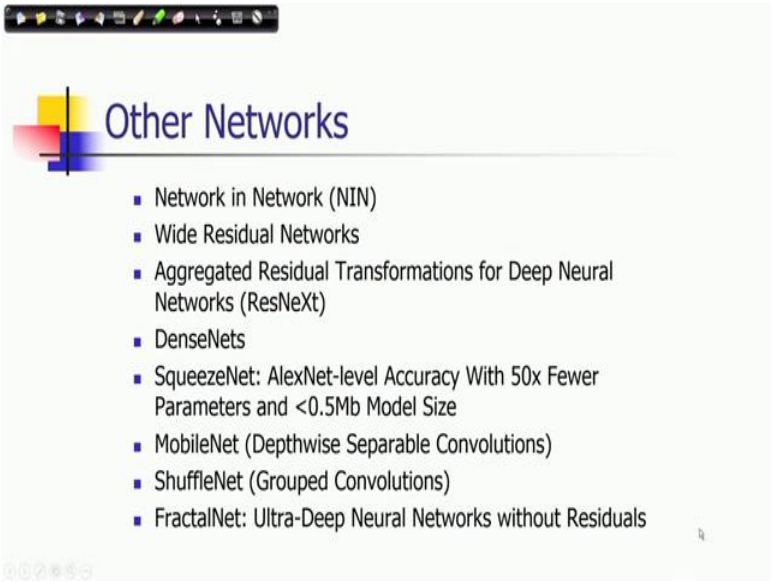
So, in the residual network or ResNet what we learned? We try to learn the residual errors. So, it fits the residual mapping instead of directly fitting the in fitting the data itself. So, as you can see in this unit that the learn; what you are learning; you are learning the residual part effects. Because, there is a skip connection here, it is called skip and it is added with this residual part and then only you are getting the output H x. So, what you are learning H(x). F(x) is H(x)-x which is the residual part that is what you are learning.

(Refer Slide Time: 19:48)



So, this is one schematic diagram where ResNet are shown, as you can see that the skip connections are there regularly and your number of layers could be very large. It could be more than 100 now and it has increased the performance of the classification to a great extent.

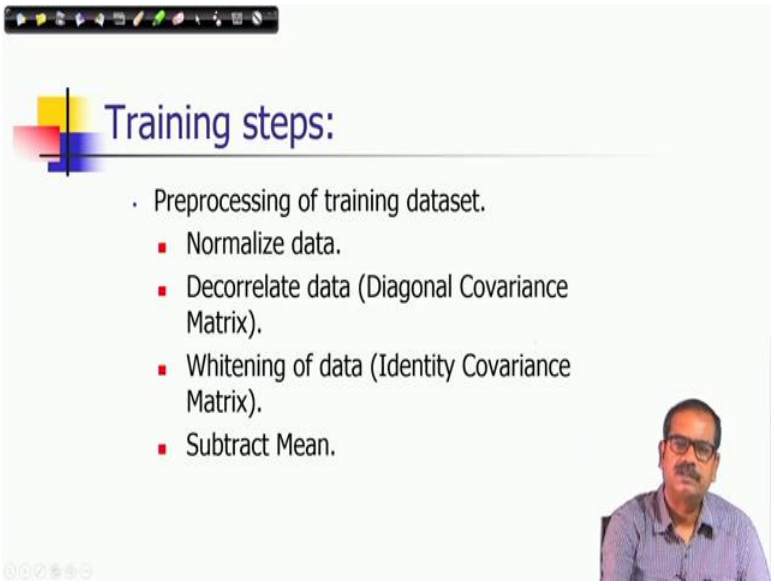
(Refer Slide Time: 20:08)



There are other examples of networks also like some list is provided here, I am not going to read all of them. So, you may go through these references. In particular let me point

out one network which is called MobileNet, it is used for low end applications and here all the filters are of size of 1×1, they are all separable convolutions.

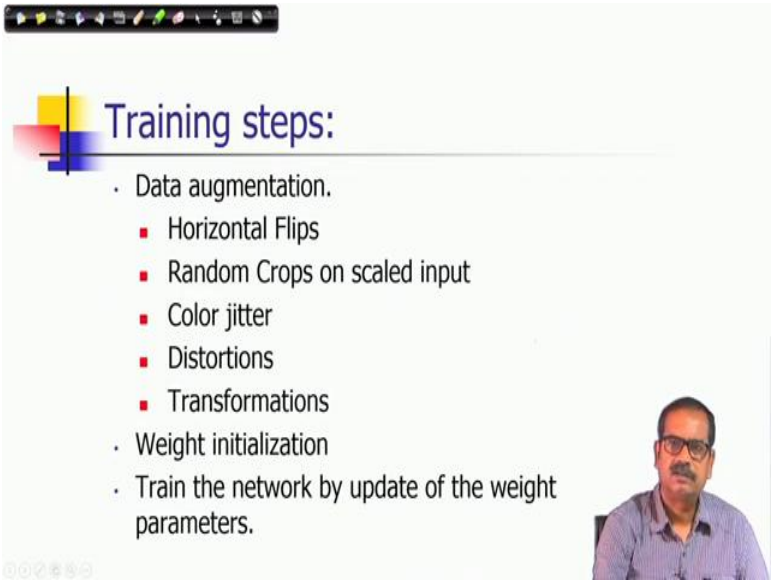
(Refer Slide Time: 20:36)



So, to there are few things we would like to discuss about training of this network. First thing is that we would like to pre-process the training data set, different kinds of pre-processing is required. We need to normalize the data with respect to the variances of the data that we have to normalize. Then decorrelation is also required which means diagonalization of the covariance matrix We know the decorrelation of data is principal component analysis, that we have learned and whitening of data is identity covariance matrix.

So, which means that during normalizations you have to divide it by the variances, you have to normalize the ranges and or you can subtract the mean. So, different kinds of pre-processing operations that may involve, while using these data for the purpose of computation.

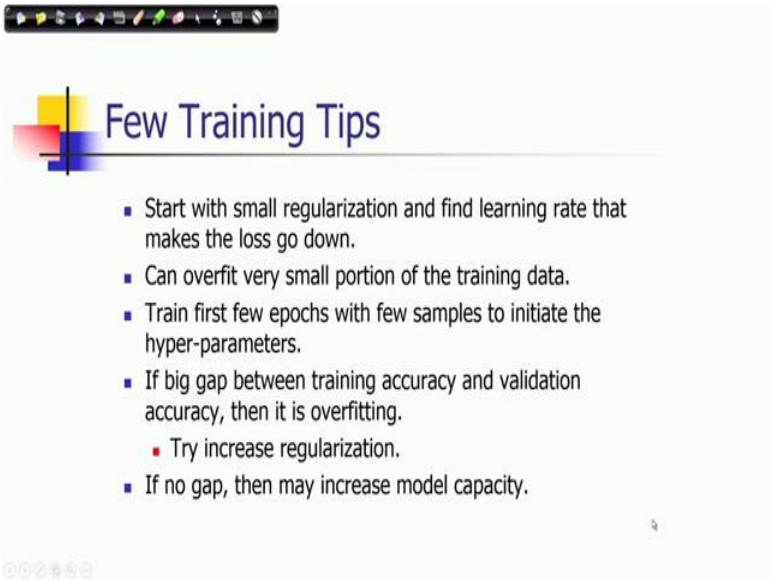
(Refer Slide Time: 21:34)



Then you may require to generalize your data set because, the data set what you have with level data set that may not capture the all variances of in the class. So, that is called data augmentation because by doing some simulation you can also generate some more data. So, like you can perform different kinds of geometric operations on data, images say horizontal flips, random crops on scaled input, color jitters.

You can add noise, you can add variances, add different kinds of color variations, distortions, then transformations. So, there are different kinds of simulations you can do on the given input data set and generate more input data set and use it for training. Then you need to initialize weight, there are different policies for initialization of weights. I will discuss that and train the network by updating the weight parameters.

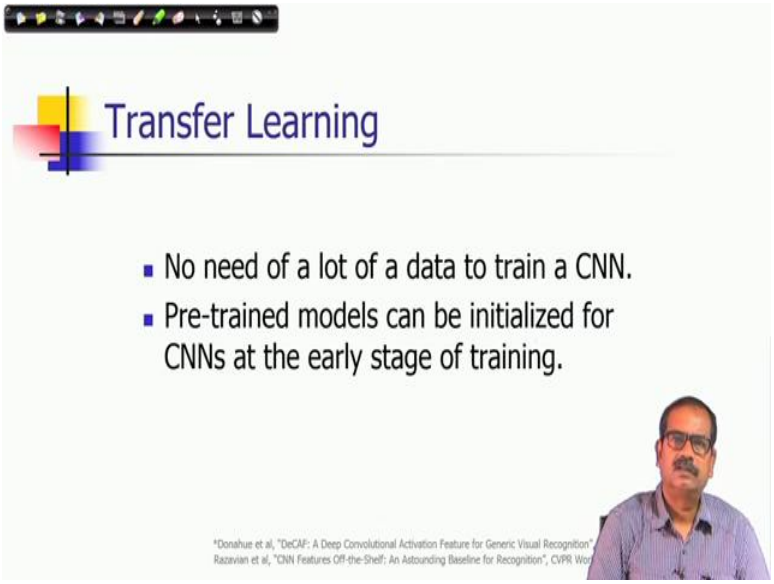
(Refer Slide Time: 22:44)



So, there are few training tips that is required that I would like to provide here. So, we should start with small regularization and find learning rate that makes the loss go down. So, regularization parameter; so, there is a loss function you have the regularization term and also the loss function due to the cross entropy term for example; so, it is a linear combination. So, start with the regularization component keeping it a small and also try to find out learning rate that makes loss go down. So, it can over; you can overfit very small portion of the training data and then train first few epochs with few samples to initiate the hyper parameters.

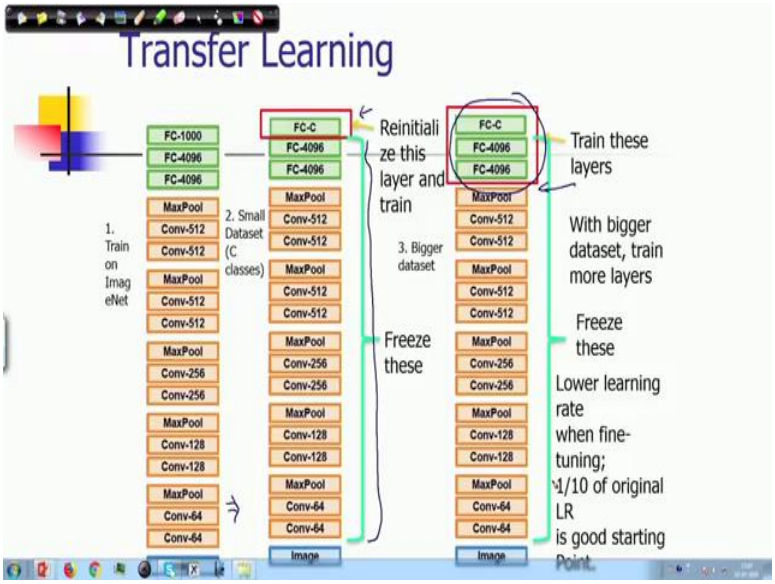
So, first you have to introspect how it is behaving and try to select those parameters which are giving you which is providing you the directions of convergence, good directions of convergence. So, those parameters you need to consider and because it is very time consuming; so, once you fix the parameters then again empirically you have to observe how it is behaving. So, these are certain tips by using it you can reduce your time of experimentations. So, if you find there is a big gap between training accuracy and validation accuracy then it is a overfitting case. So, what you should do? You should increase the regularization term in that case or if there is no gap then you may increase the model capacity.

(Refer Slide Time: 24:18)



So, one of the policy for initialization of the weights is called transfer learning and in this case even if you have a small data set, you can work with deep learning paradigm. You do not require a lot of data set to train CNN; usually deep learning requires lot of data set. Because, what you can do you can use a pre-trained network which has been trained using a very large data set for some applications. But deep, but use that feature representation in your own application. So, pre-trained models can be initialized for CNNs at the early stage of training.

(Refer Slide Time: 24:59)



So, the example here I will providing say for example, you train on ImageNet data set, a particular network. Here we have shown that there are few convolution networks and then fully connected networks given the image and it also shows number of channels etc. And, now in your small data set you fix this portion so, this is your feature representation. So, the whole feature representation is fixed so, you can freeze this and you can transfer the weight which has been learnt in this network, you can transfer it directly to this because your architecture is kept same.

So, that is a constraint you have to keep the same architecture; so, you transfer those weights and use these weights to learn the features. And, then only you train the output layer; that means, the final classifier you only train that part and you can reinitialize and train for the classification in your particular to your problems, what you are working on. If you have a large data set bigger data set, then you can have some other you can have other kind of policies. For example, in this case you can consider the all the fully connected layers to be trained and by keeping your future representations only at this point.

So, you can consider that and your lower; you have a, you can fine tune finally so, what you can do also that you can fine tune once you get a convergence some weights then you can again fine tune. So, you can train with your input data samples and using a lower learning rate and which is like one-tenth of the original learning rate; I mean that is a good starting point and then you know you can vary it.

So, these are you know different pragmatic policies and you know some of the intuitions, heuristics those are used while training a good deep neural network which takes lot of time and also lot of resources. And finally, a good network when it gives you a result that requires a lot of experimentations and it involves lot of experimentations you should understand that.

So, with this let me stop here and we will continue this discussion on deep neural architecture in my next lectures.

Thank you very much for your listening.