Deep Learning

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

Lecture - 48
Batch Normalization – II

Hello welcome back to the NPTEL online certification course on Deep Learning. For last 2 lectures we are discussing about the normalization techniques or normalization of the feature vectors. What we have seen in the previous class or in the previous 2 classes are why do we need normalization and in the previous class we have talked about one particular normalization technique which is known as Batch Normalization.

So, let us quickly recapitulate what we have done in previous 2 classes. The basic purpose of normalization as we have discussed is to make your learning procedure more efficient, faster and more accurate. And while doing so, while you train a neural network or deep neural network; one of the problems which are faced while training is what is known as covariate shift and covariate shift particularly occurs because normally the kind of training that we employ is what is known as batch training or batch stochastic gradient descent.

And in this algorithm you take your training examples in batches where different batches are usually non intersecting and you train your network using these batches of the training examples one batch at a time. Now, what may happen is, as you have data in the batches which are disjoint the distribution or the data distribution in one batch may be quite different from the data distribution in another batch. And what we have seen earlier that the way a classifier learns is, the classifier basically learns the distribution of the data and based on the distribution of the data it tries to find out what is the boundary between 2 different distributions.

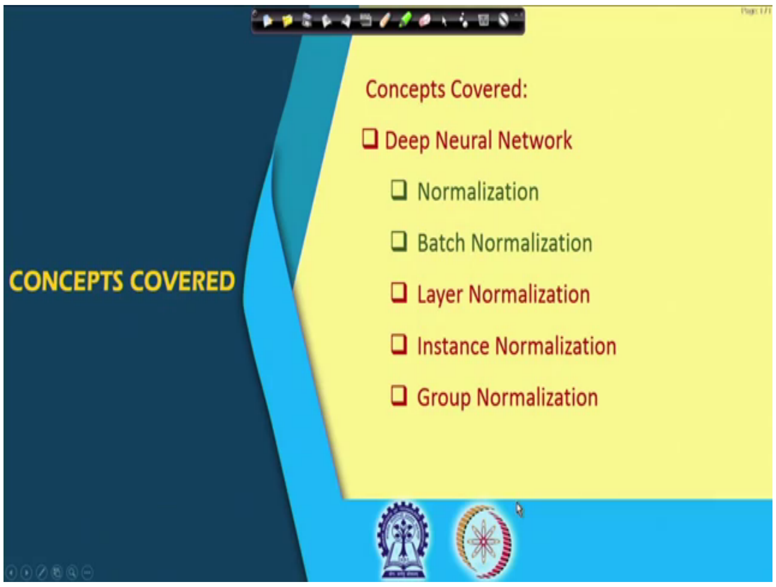
Say for example, if we say that we want to classify or we want to separate between say birds and flowers it is expected that the feature vectors corresponding to bird have one distribution whereas, the feature vectors corresponding to flowers have some other distribution and the classifier learns the boundary between these 2 data distributions. So, after the classifier is learnt if you find that one of the unknown data falls on one side which may be the side in which the distribution of the data belonging to the birds exists then that unknown data will be classified as a bird, if it falls on the other side then the unknown data will be classified as a flower.

And that is the case when you have a 2 class classifier, if you have multi class classifier say for example, I have birds, I have flowers, I have say deers, I have cars and all that. So, the feature vectors belonging to each of these different categories they will have different distributions in the feature space or in the vector space and in such cases what the classifiers learn is for all different classes the classifier learns something known as discriminating function.

So, for one class the distribution discriminating function may be g 1, in another class the discriminating function may be g 2, for another class it is g 3 and so on. So, it is expected that if the discriminating function g 1 is meant for say bird class, then given a data or example from the data distribution from the bird class g 1 output will be maximum among all other outputs of all other discriminating functions. So, as a result the unknown data will be classified to bird class.

And here again if the distribution of the data belonging to the same class changes, then learning the classifier or to have a particular discriminating function for different classes that will be very difficult. And as a result you are learning or training of the neural network or the classifier becomes very very slow.

(Refer Slide Time: 04:59)

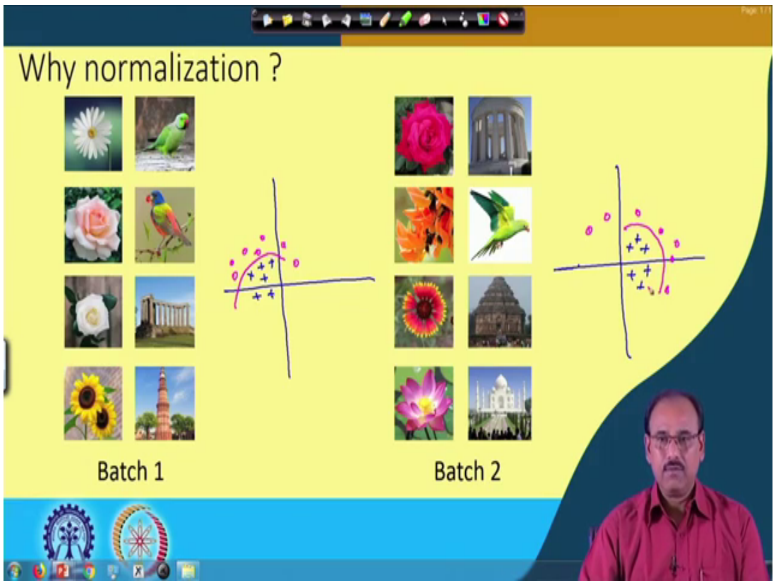

CONCEPTS COVERED

Concepts Covered:

- ❑ Deep Neural Network
  - ❑ Normalization
  - ❑ Batch Normalization
  - ❑ Layer Normalization
  - ❑ Instance Normalization
  - ❑ Group Normalization

So, what we have seen before is we have talked about batch normalization technique and today we are going to talk about the other normalization techniques like layered normalization, instance normalization and group normalization. And while talking about these normalization techniques we had taken a particular example say as given over here.

(Refer Slide Time: 05:18)



So, in this example what we have tried to do is, we have tried to discriminate between say flowers from all other classes. So, flower belongs to one class and the images from all other classes belong to another class. Now, find that we have taken 2 different batches of data or training samples batch 1 is on the left hand side and batch 2 is on the right hand side.

So, positive class in our case in this particular case is the flower another in batch 1 you find that this positive class that is the examples of flowers that we have they are mostly whitish in nature or they are not rich in color. Whereas, in batch 2 all these flowers they are very very rich in color.

So, when you represent them into the vector space then the vector representing the flowers in batch 1 distribution of those vectors and the vectors from for flowers in batch 2 the distribution of those 2 vectors these 2 distributions are likely to be quite different. So, as a result so, if I put it like this say take a feature space something like this another feature space over here, in the positive class in flowers say in this particular case all the positive class the features will be distributed like this. Whereas, in batch 2 all the positive class that is the colored features they will have distribution they might have distribution something like this.

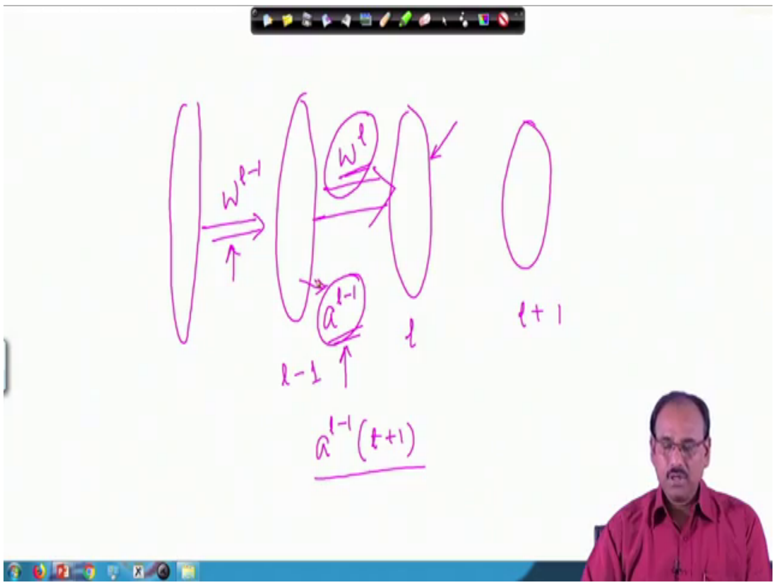
Now, on the other hand if you take examples of negative classes maybe negative classes will have distributions of this form, over here again the negative classes will have distributions of this form. So, in one case the classifier which discriminates flowers from all other classes that will they it will learn this boundary whereas, in this case it will learn a boundary something like this. So, you find that the boundaries which are learnt from the classifiers they are different in these 2 different cases.

So, when you are going for training in batch manner that is a for example, in batch gradient descent your classifiers will simply hop from one classifier to another classifier, because in different batches we are getting different sets of data and effectively that slows down the learning process or the rate of learning becomes very very slow. So, in order to avoid this you go for normalization techniques and in case of normalization techniques what is done is all the data are normalized to say 0 mean and unit variance or unit standard deviation.

So, if you do that, in that case you find that all the data because now they are normalized. So, when the classifier is the data while trying or while training while being trained it finds the distribution to be the same right. So, unlike over here what you have seen or the distribution changes, as the distribution remains the same your learning process becomes very very fast. And we have also said that this is not only applicable at the input layer because this raw data that we are shown we are showing this raw data is fed to a input layer.

This is also applicable to the hidden layers, there is not being in hidden layer if I consider the training of layer l, while when you train layer l it is based on the gradient of the error gradient which is propagated to layer l from the output layer as well as it also depends upon the activation output of the previous layer that is l minus first layer.

(Refer Slide Time: 09:34)



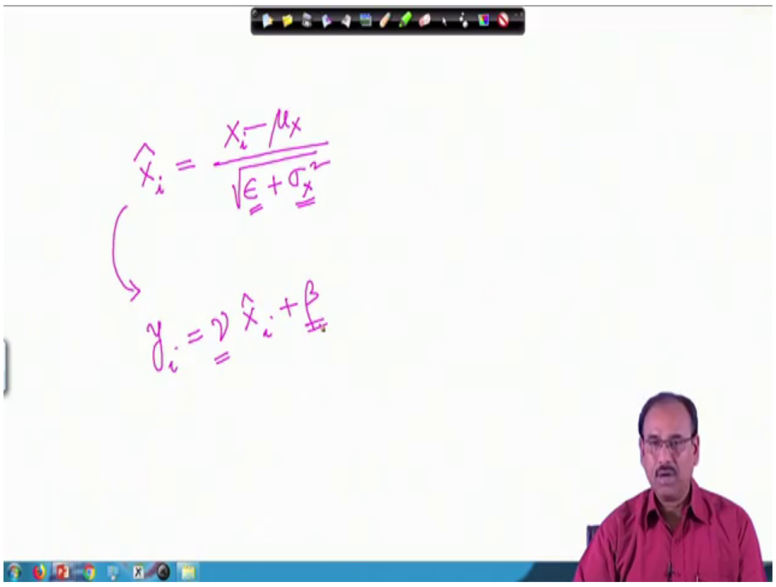
So, it is something like this if I put it in this form say I have a neural network having multiple number of layers. So, these are the different neurons, this is my l th layer, this is l plus first layer, this is l minus first layer. So, when you train this l th layer or when the weight vectors w l which are trained between the layer l minus 1 2 layer l this upgradation of this weight vectors depends upon the error which is propagated up to the l th layer as well as the activation output of l minus first layer so, which is l minus a l minus 1. Now, you find that when you are training or updating this w l at time instant t it depends upon what is the output at that time instant which is a l minus 1 activation from the previous layer.

Now, this does not this distribution of a l minus 1 again it is not static, because during the gradient descent procedure or back propagation learning you will also modify the weight vectors w l minus 1 that is the weight vectors between the layer l minus 2 to l minus 1. As a result next time when you get a l minus 1 at time instant say t plus 1 the distribution of this over the same set of data in the same batch may not remain the same, because this w l minus 1 is not same anymore it is changing.

So, as a result you find that training or for learning of w l again this is not stable this is going to be unstable because the distribution of the data based on which w l is being trained is been updated that distribution also varies over time. So, the same normalization technique that we have done for the input layer is also applicable in the hidden layers.

So, in that case this a l minus 1 that is the outputs from the l minus first layer they are to be normalized. So, for this normalization purpose what you do is you find out, what is the variance or standard deviation of the data and what is the mean of the data.

(Refer Slide Time: 11:54)



$$\hat{x}_i = \frac{x_i - \mu_x}{\sqrt{\underline{\underline{\epsilon}} + \underline{\underline{\sigma_x^2}}}}$$

$$y_i = \underline{\underline{\gamma}}\,\hat{x}_i + \underline{\underline{\beta}}$$

So, for any x, data x, x is normalized as x minus mu x, for mu x is the mean of the population divided by square root of epsilon plus sigma x square. What the sigma x is, the standard deviation mu x is the mean and epsilon is a very very small positive constant which is introduced to make your division stable, that is I do not come across any situation like division by 0.

So, this is how the normalization procedure is done and you find that as because of this normalization as the data distribution is now normalized. So, the training of every layer becomes independent of the training of other layers and at the same time because now your distribution becomes stable the classifier which is being trained that does not have to hop from one boundary to another boundary in different batches.

So, as a result the training becomes more stable, it becomes faster and the training of one layer becomes almost independent of the other layer, but when you do so, the other problem comes into picture. That is you find that the data from all different classes they are being normalized to be 0 mean and unit variance or unit standard deviation. So, if it is done in that case the data loses it is class belongingness, I mean you cannot discriminate the data belonging to different classes anymore, because the distribution of every data irrespective of from whichever class the data has come it has 0 mean and unit variance.

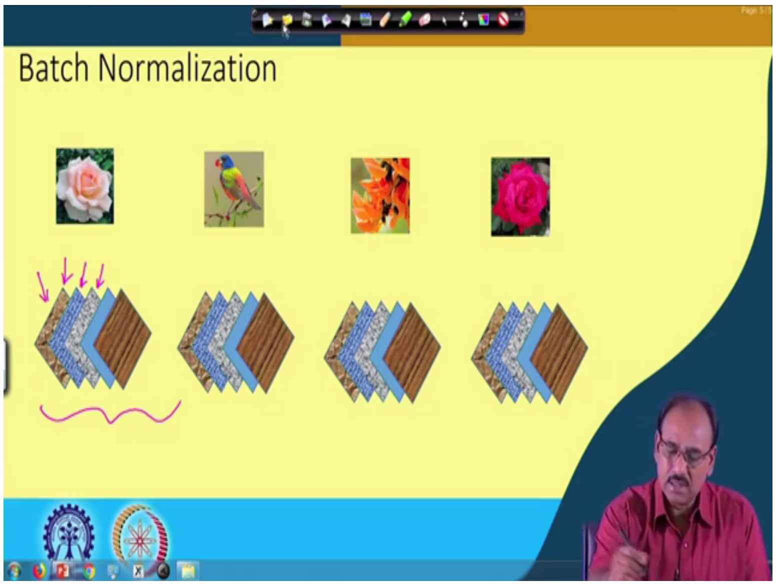
So, apart from this short of normalization the next operation that is done is what is known as a re-parameterization. What is re-parameterization? Suppose through this process I have or my normalized data which is x hat so, given an input data which is x i the normalized data is x i hat. So, using this x i hat you renormalize your data in the sense that from here I create y i which will be some say gamma times x i hat plus beta.

Where this gamma and beta they are actually tunable parameters and gamma becomes the new standard deviation and beta becomes the new mean and because these are tunable parameters so, the class belongingness of the data can be encoded within gamma and beta. And this gamma and beta are trained using the using similar back propagation algorithm along with the tuning of the parameters of your neural network.

So, along with updation of the weight vectors you also update this gamma and beta. So, that is how this gamma and beta are being trained and as a result you find that through this normalization what you are actually doing is, you are re projecting the data or in the other sense you are re-parameterizating the data distribution. And because of this re-parameterization what is ensured is, that even if there is covariate shift, but the shift of the data distribution belonging distribution of the data belonging to the same class this shift will be minimized ok.

The shift will not be much in very very small as a result the training of the neural network or the training process will be stable and it will be faster. So, what we have seen in our previous class is that we have talked about the batch normalization techniques.

(Refer Slide Time: 16:23)

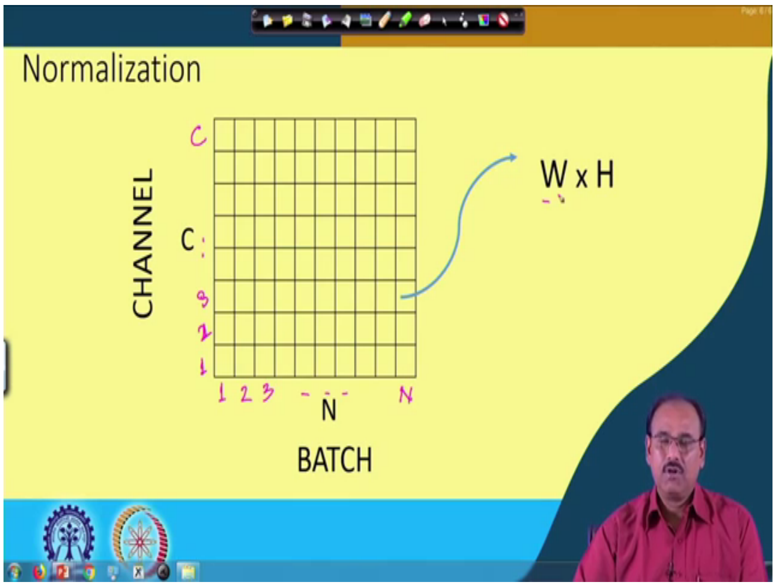


Batch Normalization

And in this batch normalization techniques what we have shown is the way this mean and standard deviation they are calculated. So, we have taken examples of the situation that suppose we have the data belonging to different classes in a particular batch and for all this data you are computing the feature maps.

So, here what is shown is that each of these textured outputs they are actually feature maps generated by different convolution kernels. And in batch normalization the way you compute the mean and the standard deviation or mean and the variance is as follows.

(Refer Slide Time: 17:15)



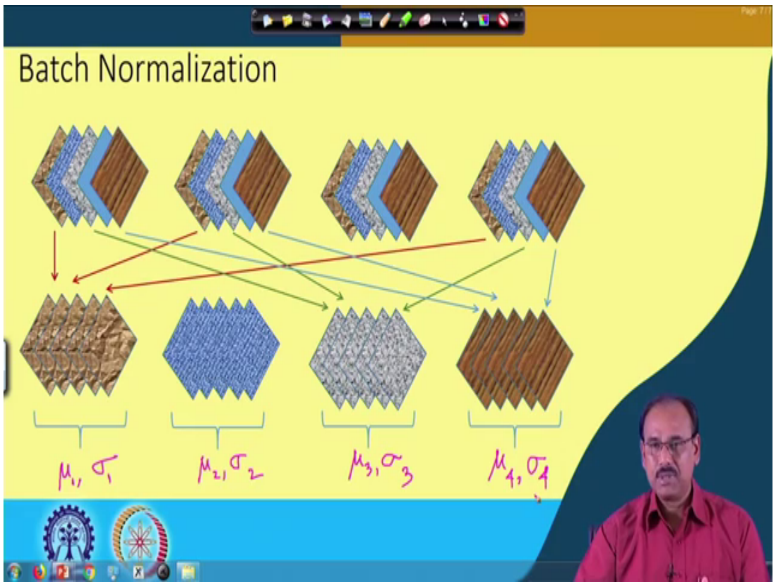
Normalization

So, I assume that in a batch say I have N number of training examples and suppose they are there are C number of convolution kernels. So, you have C number of channels being created from every example in batch N in N in every example in a batch having N number of examples.

So, these channels I can represent in the form of this matrix. So, what does this matrix mean? So, in this batch suppose this is my example 1, this is example 2, this is example 3 and so on and at the end I will have example N as there are N number of examples in this batch. And this is the channel number 1, channel number 2, channel number 3 and so on, this is channel number C every channel is the output of one convolution cannel.

And every channel I also assume is of size W by H. So, W is the width of the channel and H is the height of the channel. So, all these channels stack together that becomes a feature map which is fed to the next layer for training during the training operation and it is spread to the next layer for classification during the testing or inferencing operation.

(Refer Slide Time: 18:40)



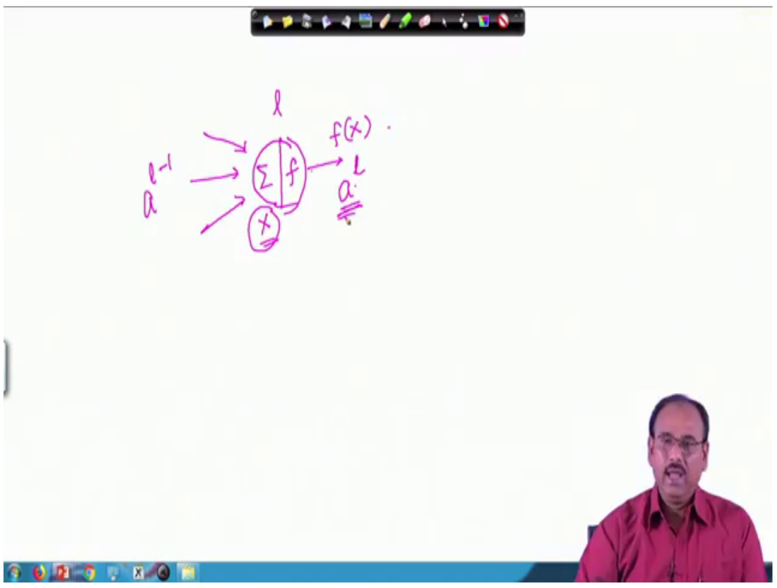
Batch Normalization

So, given this in batch normalization what you do is, you group the outputs of every channel right say for channel 1 you group the outputs of channel 1 for every example. Similarly, you group the outputs of channel 2 for every example, you group the outputs of channel 3 for every example and so on. So, these are the groups that you formed.

And once you have this grouping for every group you compute, what is the mean? And what is the standard deviation or variance? So, having this as channel 1 this will give you mu 1 this will give you sigma 1, if this is the output from channel 2 I will get mu 2 I will get sigma 2, from here I will get mu 3 I will get sigma 3, here will get mu 4 I will get sigma 4 and so on.

So, you find that you are grouping the outputs of individual channels together. So, all channel 1 are in one group, outputs of channel 2 are in another group, outputs of channel 3 are in another group and so on. And for every channel you compute the mean and standard deviation and these feature maps are being normalized with respect to these mean and standard deviation. Now, here again you remember one thing that when I go for normalization how I can normalize.

(Refer Slide Time: 20:19)

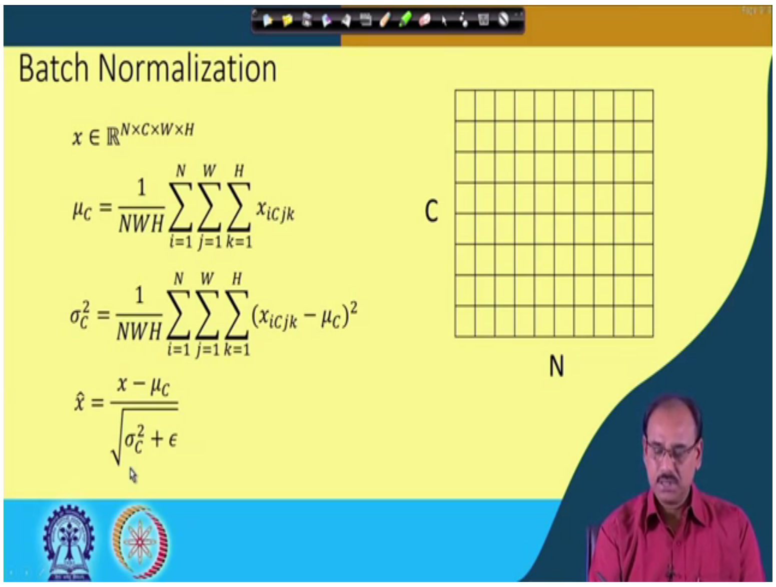


Take for example, as we have shown that every neuron in every layer consists of 2 parts; one part computes the weighted sum of all the outputs that it is receiving from the previous layer.

So, this is the l th layer, this is a activations of the previous layer that is a l minus 1. And the first part of the neuron computes the activations weighted sum of the activations that it receives from a previous layer and this output let us call it as x and the other part is the computation of activation of this x. So, this gives you output of this neuron is what is f x or I can say that this is activation of the l th layer.

Now, when I go for normalization you can normalize either a l or you can normalize x and what is more popular is normalization of x rather than normalization of a l or the activations of the l th layer right. So, when I discuss this normalization in our case we are assuming that this normalization is with respect to x right. So, here as we have shown that for each of the group of channels I have mean and standard deviation.

(Refer Slide Time: 21:42)



Batch Normalization

$x \in \mathbb{R}^{N\times C\times W\times H}$

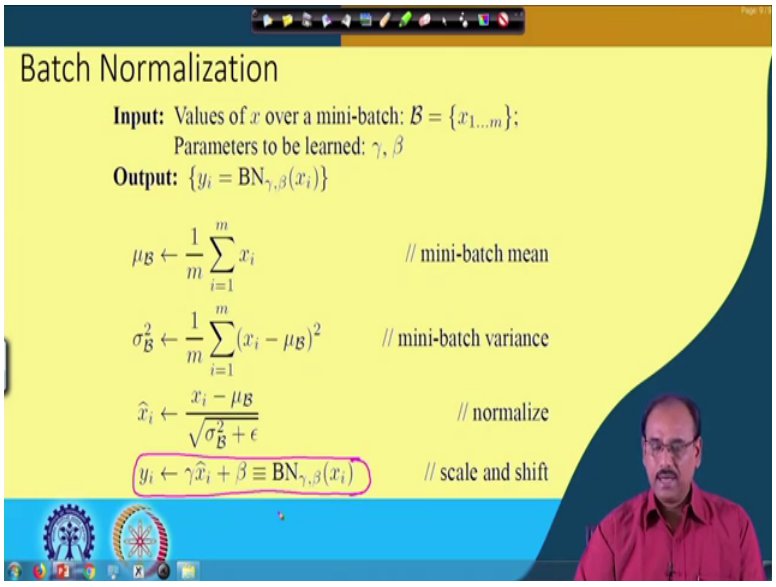
$$\mu_C = \frac{1}{NWH}\sum_{i=1}^{N}\sum_{j=1}^{W}\sum_{k=1}^{H} x_{iCjk}$$

$$\sigma_C^2 = \frac{1}{NWH}\sum_{i=1}^{N}\sum_{j=1}^{W}\sum_{k=1}^{H} (x_{iCjk} - \mu_C)^2$$

$$\hat{x} = \frac{x - \mu_C}{\sqrt{\sigma_C^2 + \epsilon}}$$

And for this mean and using this mean and standard deviation you normalize the outputs of the channels and this is what you get and if I put all these means for all the channels in the form of a vector or the variance of all the channels in the form of a vector I get the mean vectors, I get the variation variance vector or the standard deviation vector. And the normalization as we have just shown that normalization in this case is x hat will be x minus mu C upon square root of epsilon plus sigma C square. So, this is the normalization operation.

(Refer Slide Time: 22:23)



Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

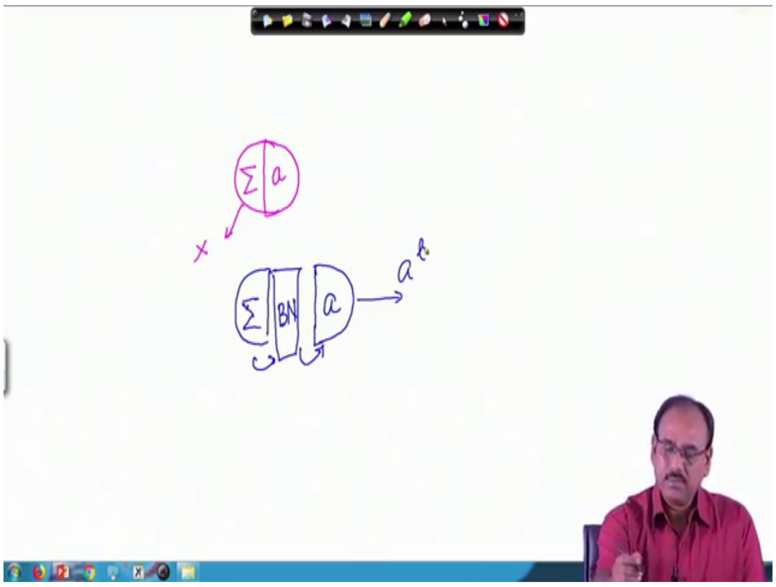
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$\boxed{y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i)} \qquad \text{// scale and shift}$$

And after that as we have said before that, what you are going for this re-parameterization. That is from every x i normalized training vector x i we create the batch normalized vector y i which is gamma x i hat plus beta and this is what is your batch normalized data.

(Refer Slide Time: 23:04)



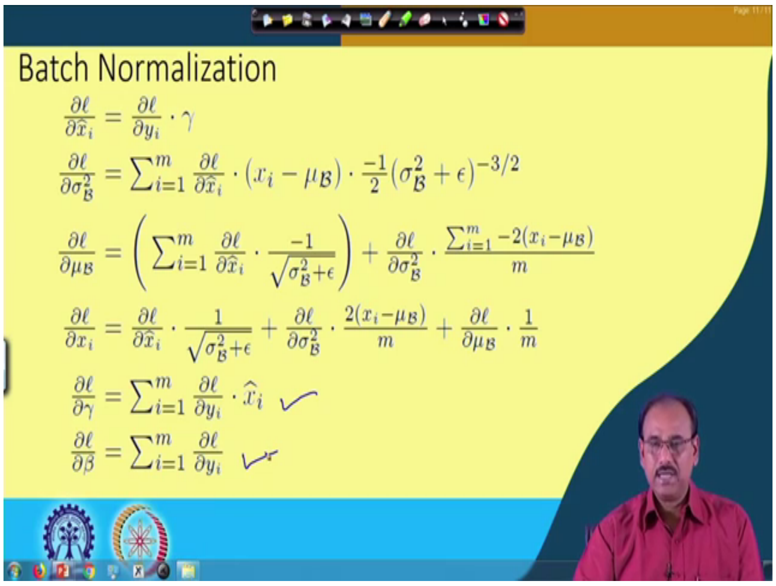
So, you find that what I need to do now is that for every layer or for every neuron as we had said that there is an aggregation aggregator called which collects the weighted sum of the inputs that it receives from the previous layer and there is an activation part and we are normalizing this aggregator part which is my x.

So, in other sense I can say that in between x and the activation. So, I have this portion which computes the weighted sum, I have this portion of the neuron that computes or the node which is gives you the activation and in between you introduce a layer which is in the batch normalization layer or BN. So, the output of this is batch normalized it is fed to the activation function layer activation layer and then you finally, get the output a l from the l th layer. So, this is how the batch normalization is to be done.

So, now, we said that when you talk about these normalizations, the different types of normalizations that you have that depends upon, how you compute the mean and sigma vectors alright. So, in case of batch normalization it is all the channels grouped together and for that channel you compute the mean and the standard deviation.

So, if you consider what is the dimensionality of this vector mu B or what is the dimensionality of this vector sigma B squared. If we have C number of channels then this vector mu B will have C number components, because for every channel I will have 1 mini. Similarly sigma B square will also have C number of components, because I will have one variance for every channel so; that means, I will have mu 1 I will have sigma 1, I will have mu 2 I have sigma 2 and since I have C number of channels I will have mu C I have sigma C and so on.

(Refer Slide Time: 25:32)



Batch Normalization

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

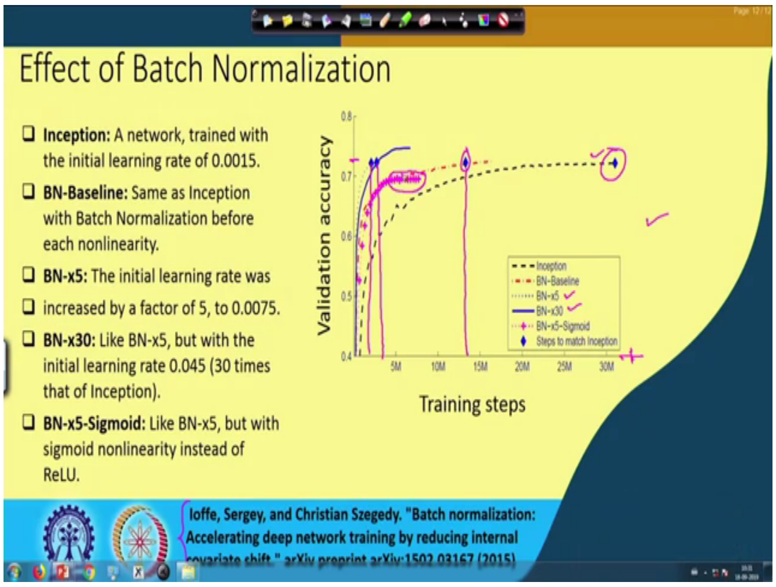
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i \quad \checkmark$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \quad \checkmark$$

And once you compute this then the way you go for updation or tuning of the mean and sigma, the tuning of gamma and beta which are the re-parameterization or the re-projected parameters will be again by gradient descent procedure. And for that you have to take that gradient of the loss function with respect to B with respect to gamma as given over here and you also have to take the gradient of the loss function with respect to beta as given over here. So, this is what is your batch normalization.

(Refer Slide Time: 26:18)



Effect of Batch Normalization

- **Inception:** A network, trained with the initial learning rate of 0.0015.
- **BN-Baseline:** Same as Inception with Batch Normalization before each nonlinearity.
- **BN-x5:** The initial learning rate was
- increased by a factor of 5, to 0.0075.
- **BN-x30:** Like BN-x5, but with the initial learning rate 0.045 (30 times that of Inception).
- **BN-x5-Sigmoid:** Like BN-x5, but with sigmoid nonlinearity instead of ReLU.

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

Now, let us talk about the other normalization procedure which is say and this is where what we have shown is what is the effect of batch normalization that you get. So, this is an experimental output which is given over here as given in this particular paper right and the experiment was done on a network known as inception and the initial learning rate of the network was given as 0.0015.

So, the experiments are done on the baseline with the same learning rate of 0.0015 without any batch normalization then it was done with batch normalization with initial learning rate being 5 times of the initial 0.0015. Then another experiment with learning rate initial learning rate which is 30 times of 0.0015 and then again the normalization was applied with an activation function which is not ReLU, but sigmoid.

So, as it has it is shown on the right hand side you find that this particular curve shows the performance of the validation accuracy without any normalization and you find that to have an accuracy of say more than 0.7 it is say 0. 72 or so somewhere over here the number of iterations which is required is more than 30 million. Whereas, when you go for batch normalization as is given over here with a learning rate initial learning rate of 0.0015, you find that the same validation accuracy is obtained within a number of iterations which is less than 15 million right.

Similarly over here this is the batch normalization with initial learning rate which is 5 times of 0.0015 you take much less number of iterations to achieve the same accuracy. Similarly, over here with an learning rate of 30 times of 0.0015, here again the number of iterations required is much less compared to the case when no normalization has been applied. Of course, this is the one which shows instead of ReLU if you use sigmoidal function as a non-linearity. So, over here you find that when you use sigmoidal function the performance I mean with sigmoidal function even your validation accuracy is much less.

So, that shows that batch normalization improves the performance or improves the learning rate you will learn faster as the covariate shift is minimized and also the different layers of the neural network are trained independently I mean they become almost independent of the training of other layers.

So, will stop here today in this lecture in subsequent lectures we shall talk about the other normalization techniques and as we have said that other normalization techniques are basically the way you compared the mean and the variance or the mean and the standard deviation.

Thank you.