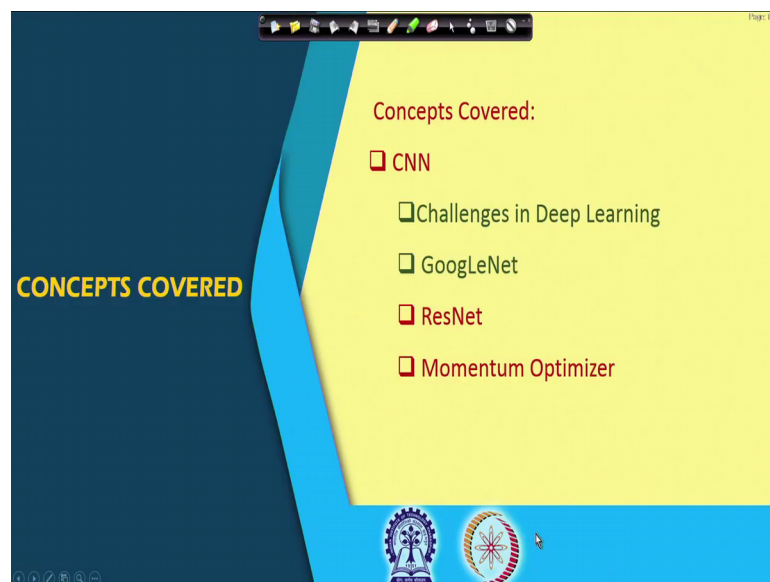


**Deep Learning**  
**Prof. Prabir Kumar Biswas**  
**Department of Electronics and Electrical Communication Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 42**  
**ResNet, Optimisers: Momentum Optimiser**

Hello, welcome to the NPTEL online certification course on Deep Learning. For last few lectures, we are talking about some of the popular convolutional neural network models. And in the last class, we have talked about a particular model which tries to deal with a very, very challenging problem faced in the deep neural network which is the vanishing gradient problem.

(Refer Slide Time: 01:01)



So, the problem that we have talked about that is the vanishing gradient problem. We have seen the architecture of a network known as GoogLeNet and we have seen how GoogLeNet tries to alleviate the problem of vanishing gradient. In today's lecture, we will talk about another network architecture which is known as residual network or ResNet, and we will also see that how in case of residual network the vanishing gradient problem is addressed.

And after that we will also talk about another challenge which is faced in the deep neural network that is how to choose the parameters or the step size when you update the

network parameters or the network weights. So, today we will particularly talk about one particular approach to solve such a problem which is known as momentum optimizer.

(Refer Slide Time: 02:03)

**Challenges**

- ❑ Deep learning is data hungry.
- ❑ Overfitting or lack of generalization.
- ❑ Vanishing/Exploding Gradient Problem.
- ❑ Appropriate Learning Rate.
- ❑ Covariate Shift.
- ❑ Effective training.

18

So, let us see just briefly what we have done in the previous lecture. So, the problem did that we tried to address this is the vanishing or exploding gradient problem and a particular network architecture we discussed is GoogLeNet, and we have seen how this vanishing product gradient problem is addressed in Google network.

(Refer Slide Time: 02:27)

**GoogLeNet**

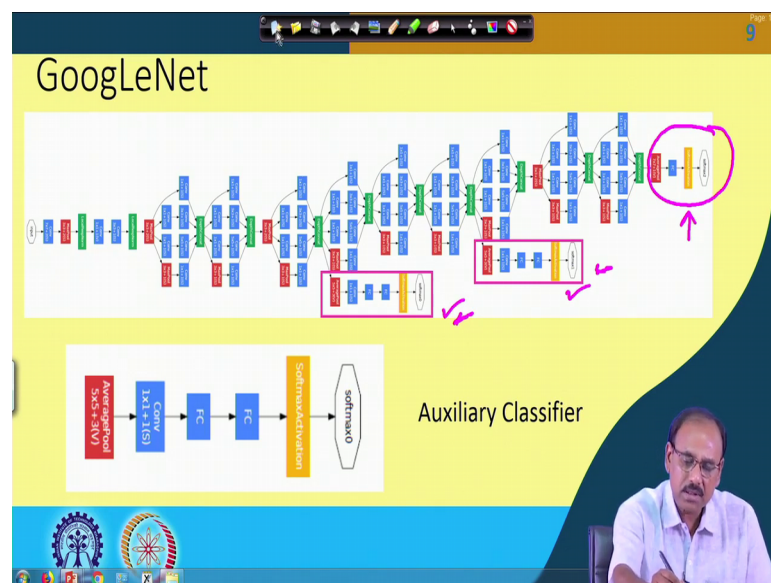
**Inception Module**

13

So, in GoogLeNet the architecture is something like this. So, as we have discussed in the previous class, the GoogLeNet consists of 9 inception modules, and every inception module computes the features at different scales starting from starting with the convolution kernels of size 1 by 1 to convolution kernels of size 5 by 5. So, all these different features at different scales are computed by inception module in GoogLeNet simultaneously.

And the outputs of each of these feature extractor modules or outputs of each of these convolution kernels; they are stacked one after another to give you the feature map from a particular inception module. And then this feature map is passed onto the next inception module as an input for processing. So, these are the basic elements of the GoogLeNet.

(Refer Slide Time: 03:39)



And the way the GoogLeNet addresses the problem of vanishing gradient is by making use of some auxiliary classifiers. So, as we have discussed in the previous class, we have one classifier over here which is actually the main classifier. And we have two more classifiers over here which are auxiliary classifiers. These auxiliary classifiers take the outputs from the inception modules which are in the middle of the GoogLeNet, and exploit the power of the features or the discriminative power of the features which are computed by the inception modules in the middle. And because these

are also classifiers, so they also produce some loss function. So, both these auxiliary classifiers at this level and this level they also compute some loss function.

So, when you have compute the final loss function, these two loss functions after scaling by a factor of three is added to the final loss function which is computed over here. And this inter loss function is used for back propagation learning that is the gradient of loss function with the parameters with respect to the parameters are passed in the backward direction for parameter updation. So, as you are computing the loss functions in the middle of the GoogLeNet, so the problem of vanishing gradient is believed to be addressed by using this auxiliary classifiers which are in the middle of GoogLeNets.

So, with this brief introduction of what we have done in the previous class, now let us try to see the other network that we are going to discuss today is the residual network or ResNet. And let us see that how this ResNet also addresses the problem of vanishing gradient.

(Refer Slide Time: 05:43)

ResNet

- ❑ Core idea is: introduction of Skip Connection/ Identity Shortcut Connection that skips one or more layers.
- ❑ Stacking layers should not degrade performance compared to its shallow counterpart.
- ❑ Weight layer learns  $F(x)=H(x)-x$

Diagram illustrating the ResNet residual block structure:

The input  $x$  is split into two paths. The first path goes through a weight layer, a ReLU activation, and another weight layer to produce  $F(x)$ . The second path is an identity shortcut that bypasses the layers. The outputs  $F(x)$  and  $x$  are added together ( $F(x) + x$ ), and the result is passed through a ReLU activation to produce the final output  $H(x)$ .

<https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>

So, the core idea in the ResNet is, it introduces skip connection or identity shortcut. So, the purpose of this skip connection or identity shortcut is that it skips one or more parameter layers or weight layers, and feeds the input and directly bypassing say one or more layers to a layer ahead. So, the connection that is given over here that suppose I have this is the output from one layer and following this I have two more layers, a one layer over here and another layer over here.



So, this activation from the previous layer is directly moved to two layers ahead and added over here to be passed on to the next layers. And this is done in addition to the regular path of the information flow in the forward path which flows through these two weight layers as well. So, this skip connection is the core contribution of the core idea in the skip connection or ResNet network and it is believed that stacking of the layers this should not degrade the performance when you compare the performance with respect to its shallow counterpart.

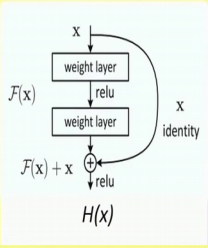
Because, we have discussed in the previous class that in many cases it has been found experimentally that the performance of the deep neural network which is believed to give better performance compared to the shallow network for the same job, but which is not always true, and this happens because of the vanishing gradient problem. So, because of this skip connection, as the skip connections are introduced, so now, it is expected that stacking of layers or as we have more number of layers in our deep neural network, the performance should not be worse than its shallow counterpart.

And while doing so why it is ResNet, if you look at these two layers over here what these layers learn. If I assume that output over here is my  $Hx$ , so  $Hx$  is nothing but  $x$ , but plus  $Fx$ , where  $Fx$  is the activations which are implemented these two layers which are skipped, so  $Hx$  will be basically  $Fx$  plus  $x$ . So, what is  $Fx$  in this case  $Fx$  has to be  $Hx$  minus  $x$ . So, this is the residual and that is what is learned by the layers which are actually skipped and that is the reason this is what is known as skip connection or residual network.

(Refer Slide Time: 08:41)

ResNet

- ❑ By stacking identity mappings the resultant deep network should give at least same performance as its shallow counterpart.
- ❑ Deeper network should not give higher training error than shallow network.
- ❑ During learning the gradient can flow to any earlier network through shortcut connections alleviating vanishing gradient problem.



$x$

weight layer

relu

weight layer


$F(x)$

$x$  identity

$F(x) + x$

relu

$H(x)$



So, given that so as we have already said that by stacking this identity mappings or giving the skip connections, the resulting classifier or the resulting network that we have that should not perform worse than its shallow counterpart or it should give at least to the same performance as its shallow counterpart. And as a result, it is also expected that such deep networks should not give higher training rate compared to the shallow network.

And during learning, when we have to pass the gradient or the error gradient from the output side towards the beginning of the network, such gradients can also flow from any layer to any layer before to the shortcut, shortcut connections. So, from any layer to any of the earlier networks, any layer in the any earlier layer in the networks from which we had the shortcut connection or we had the skip connection, the gradient can directly flow to that connection as well.

(Refer Slide Time: 09:54)

ResNet

Forward flow:

$$a^l = f(W^{l-1,l} \cdot a^{l-1} + b^l + W^{l-2,l} \cdot a^{l-2})$$
$$= f(Z^l + W^{l-2,l} \cdot a^{l-2})$$

$a^l = f(Z^l + a^{l-2})$  if same dimension

The diagram shows three layers: Layer l-2, Layer l-1, and Layer l. An input arrow enters Layer l-2. From Layer l-2, one path goes down to Layer l-1, and another path goes down to Layer l. From Layer l-1, an arrow goes down to Layer l. From Layer l-2, an arrow goes down to Layer l. The weight between Layer l-1 and Layer l is labeled W^{l-1,l}. The weight between Layer l-2 and Layer l is labeled W^{l-2,l}. The output of Layer l is labeled a^l.

Page 2

Logos of institutions and a URL: <https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>

So, as seen over here so whatever gradient is available at this particular level, so whatever gradient is available there at this particular level, this gradient can directly flow to the layer beyond this, so that is the advantage of the ResNet or skip connection that the gradient need not flow through each and every layer in your deep neural network, and that is how it can avoid the problem of vanishing gradient.

(Refer Slide Time: 10:35)

ResNet

The diagram shows a 34-layer ResNet architecture. It starts with an input layer, followed by a series of residual blocks. Each residual block has skip connections that bypass the block and add the input to the output. The diagram shows the flow of information through the network, with skip connections from the input layer to various layers. The output is labeled as 'output'.

Page 3

Logos of institutions and a URL: <https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>

So, now given this let us see that how the information flows through this residual neural network. So, this diagram shows a 34 layer ResNet. So, here you find that all these

bypass connections which are shown by the curved arrows. So, they represent all the skip connections which skip a number of layers in between to feed the information layers ahead. So, the way this ResNet forwards the information or as the information flows in the forward direction is given by this. If I assume that this layer is at layer  $l$ , the previous layer is at layer  $l - 1$ , and the layer previous to that is layer  $l - 2$ .

Then the information forward information which is available to the input of layer  $l$  is basically a function of the activation at layer  $l - 1$  plus 1 activation at the output of layer  $l - 1$  minus 1 plus the activation at the output of layer  $l - 2$  and that is what the skip connection gives. This activation and this activation they are added together and that gives the input to layer set layer  $l$ . And the output of layer  $l$  will be a non-linear function of this total input.

So, if I assume that the input to layer  $l - 1$  is  $a_{l-1}$  which is basically sorry the output the activation of layer  $l - 1$  is  $a_{l-1}$  and the activation output of layer  $l - 2$  is  $a_{l-2}$ , and the connection weights from layer  $l - 1$  to layer  $l$  is given by  $W_{l-1, l}$ , and the connection weights from layer  $l - 2$  to layer  $l$  is given by  $W_{l-2, l}$ , in that case the activation at the output of layer  $l$  which is  $a_l$  is given by a non-linear function of  $W_{l-1, l} a_{l-1} + W_{l-2, l} a_{l-2}$ .

Where,  $a_{l-1}$  is the output or the output activation of layer  $l - 1$  plus  $b_l$   $b_l$  is assumed to be the bias at the input of layer  $l$  plus  $W_{l-2, l} a_{l-2}$ , where  $a_{l-2}$  is the activation at the output of layer  $l - 2$ . And that can be written as it is a non-linear function of  $Z_l + W_{l-2, l} a_{l-2}$ , where  $Z_l$  is nothing but this term which is the activation at the output of layer  $l - 1$ . So, this is what is  $Z_l$ .

So, I can compute the activation at the output of layer  $l$  as a function of  $Z_l$  plus  $a_{l-1}$  in case  $a_{l-1}$  and  $Z_l$  sorry  $a_{l-2}$  and  $Z_l$  they are of same dimension. But if  $a_{l-2}$  and  $Z_l$  they are of different dimension, then I have to have a mapping by making use of a weight matrix which is  $W_{l-2, l}$ . So, if there is identity, in that case I have  $Z_l + a_{l-2}$  that passes through the non-linear activation of layer  $l$ , and that gives me the final output  $a_l$ . And this is how the information flows in the forward direction through a residual network.

So now, let us see that how does this residual network perform the back propagation learning or how this skip connections help to alleviate the problem of vanishing gradient.

(Refer Slide Time: 15:01)

ResNet

Backward Propagation:

$$\nabla W^{l-1,l} = -a^{l-1} \cdot \delta^l \quad \text{normal path}$$
$$\nabla W^{l-2,l} = -a^{l-2} \cdot \delta^l \quad \text{skip path}$$

If the skip path has fixed weights, identity matrix, then they are not updated.

The diagram illustrates the backward propagation in a ResNet block. It shows three layers: Layer  $\ell-2$ , Layer  $\ell-1$ , and Layer  $\ell$ . A skip connection bypasses Layer  $\ell-1$  and goes directly from Layer  $\ell$  to Layer  $\ell-2$ . The normal path flows from Layer  $\ell$  to Layer  $\ell-1$  and then to Layer  $\ell-2$ . The skip path flows from Layer  $\ell$  to Layer  $\ell-2$ . Gradients are shown flowing backward from Layer  $\ell-2$  through both paths. The weights  $W^{l-1,l}$  and  $W^{l-2,l}$  are labeled on the connections. The input to Layer  $\ell$  is  $x$ .

So, while back propagation, I can have the gradient to propagate backward from layer  $l$  to layer  $l-1$  following the usual link. This gradient can also flow from layer  $l$  to layer  $l-2$  following the skip connection. So, the back propagation, I have two gradients, one is the update of the weights of  $W^{l-1,l}$  that is the update over here that is given by  $\delta W^{l-1,l}$ , which is  $-a^{l-1}$  times  $\delta^l$  and that is what close to the normal path.

And, when you use the skip path that is for updating these weights  $W^{l-2,l}$ , the gradient flow becomes the  $\delta W^{l-2,l}$  is  $-a^{l-2}$  as  $a^{l-2}$  is the output activation of layer  $l-2$  times  $\delta^l$ . So, this is what follows the skip path. And of course, if the skip path has got fixed weight, in that case this weight updation of the skip path is not required anymore, because in that case it becomes an identity matrix. And what is  $\delta^l$ ,  $\delta^l$  is the propagated error gradient at layer  $l$ . So,  $\delta^l$  is over here right.

So, you find that when this back propagation has to be carried out further at this location, you have the error propagation from both paths; one is the normal path the usual path and other one is to the skip paths. And it is skip path which helps in avoiding the problem of vanishing gradient, because from the previous layers the error gradient, the gradient is directly passed to the other layers in the forward direction in the backward direction. So,



this is how the vanishing gradient problem can be tackled using this residual network or ResNet.

So, given this as we have discussed about two different architectures, one is the GoogLeNet and other one is the ResNet. And we have seen that how in these two architectures the vanishing gradient problem has been addressed. In case of GoogLeNet, the vanishing gradient problem has been addressed using the auxiliary class effects, whereas in case of ResNet the vanishing gradient problem has been addressed by using the skip connections or the residual network part ok.

Now, given this, now let us talk about the other problem, the other challenge that you face in deep neural network that is during learning or during training of the network, how do you choose the appropriate learning rate or how do you choose that what should be the weight upgradation step size that is what is your learning rate. So, let us see how these problems can be tackled in, or what are the different algorithms available for tackling this particular problem.

(Refer Slide Time: 18:49)

The slide is titled "Gradient Descent Challenges" and is presented on a yellow background. It contains a list of challenges under the heading "Challenges of Mini-batch Gradient Descent". To the right of the text is a graph showing a loss function  $L(W)$  on the vertical axis and weights  $W$  on the horizontal axis. The graph illustrates the process of gradient descent with several steps, showing how the loss decreases as the weights are updated. The graph also shows a local minimum and a global minimum, with the loss function being non-convex. The graph is annotated with  $W^{(1)}$ ,  $W^{(2)}$ , and  $W^{(3)}$  on the horizontal axis, and  $L(W)$  on the vertical axis. A small inset video shows a man speaking.

Gradient Descent Challenges

Challenges of Mini-batch Gradient Descent

- Choice of Proper Learning Rate:
  - Too small a learning rate leads to slow convergence.
  - A large learning rate may lead to oscillation around the minima or may even diverge.

So, what we will discuss today is one of the optimization approaches for optimizing gradient descent problem optimizing the gradient descent operation and the algorithm that we will talk about is what is known as momentum optimizer. So, let us see what that momentum optimizer is. Now, before that let us see that what are the challenges in the gradient descent problem. So, in gradient descent problem as we have said the first

challenge, the important challenge is the choice of proper learning rate or weight update steps.

If we choose the learning rate to be very small or weight update steps to be very small, then your convergence or the learning rate is very, very slow the convergence is very slow. At the same time, if the weight update step is very large, then that may lead to oscillation around the minima of the loss function or in some cases it may even diverge instead of converging.

So, let us see how does that can happen. For simplicity, I am assuming a loss function in one dimension say I have a loss function  $L$  which is a function of a scalar parameter  $W$ . And I want to update the value of  $W$  in such a way that the loss value of the loss should be minimized, and this is what we have already discussed in one of our earlier lectures. So, what I am going to do is just a short of recapitulation. So, to initialize what you do is your initialization is done at random say for example I have initialization somewhere over here. So,  $W$  is chosen somewhere over here.

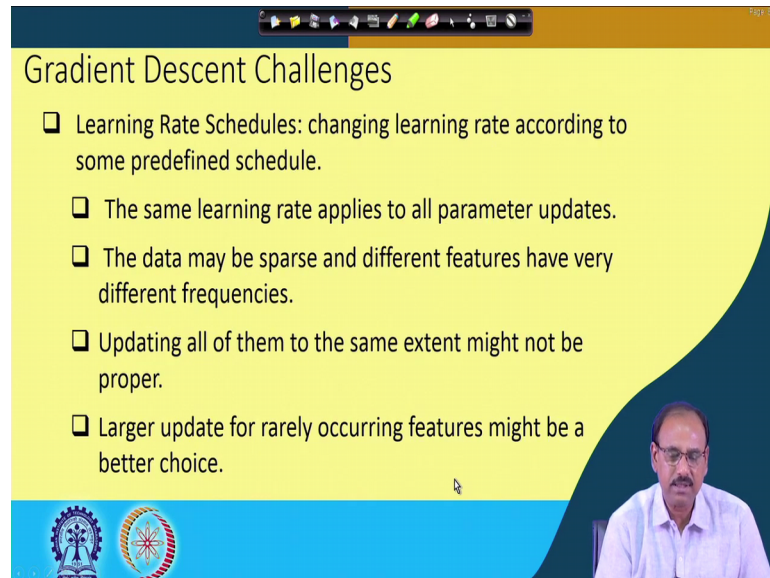
So, for update what you do is you take the gradient of  $L$  at this location, and you update  $W$  in the direction of negative gradient that means, suppose this was my  $W_0$ , and to have that  $W_1$  because the gradient direction is this, I take a step in the negative direction, so I come over  $W_2$  or  $W_1$  which is somewhere over here that is my next estimate of  $W$  or the weight value. So, here you find that as you are estimating the  $W$  over here, the value of the loss function that is deduced, now the value of loss function is this. So, my step of update was this much.

If this step size is very small, then I will slowly move towards the error minimum somewhere over here. So, the number of iterations may be quite large if my update step size is very small. If the update step size is large that is instead of  $W_1$  being here suppose that  $W_1$  comes somewhere over here if the update step size is very large. And because of this you find that your loss value straight way jumps from here to here, and in the process you are bypassing your minimum loss location.

Of course, gradually we will come over here, but there will be an oscillation. It may so happen that if the step size is very large, in that case I may even go over here it can go like this, and in such cases instead of converging to the minimum loss location you are

actually diverging. So, you find that choice of proper step or proper learning rate is very very important. So, this is one of the challenges of the gradient descent problem.

(Refer Slide Time: 22:45)



The image shows a presentation slide with a yellow background and a dark blue header. The title is 'Gradient Descent Challenges'. Below the title is a list of five bullet points, each preceded by a square checkbox. In the bottom right corner of the slide, there is a small video inset showing a man with glasses and a white shirt speaking. At the bottom of the slide, there are two circular logos: one on the left with a gear and a tree, and one on the right with a sun-like pattern.

### Gradient Descent Challenges

- Learning Rate Schedules: changing learning rate according to some predefined schedule.
- The same learning rate applies to all parameter updates.
- The data may be sparse and different features have very different frequencies.
- Updating all of them to the same extent might not be proper.
- Larger update for rarely occurring features might be a better choice.

The other challenge is that I can have a pre scheduled of learning rate that is how the learning rate will change over the iterations and that can be scheduled before and so that is what is pre scheduling. But the problem in that case is that the same learning rate has to be applied to all the parameters which might not be a wise choice, because our data may be sparse, and not only that the different features may have different distributions or different frequencies.

So, if it is so, then updating all these parameters by the same extent or using the same step size may not be proper. So, what I need to do is, depending upon the frequency a larger update I can use for rarely opening features, whereas for those features which are frequently occurring I can have a smaller update size. So, this is another problem in gradient descent optimization techniques.

(Refer Slide Time: 23:57)

The slide is titled "Gradient Descent Challenges" and is set against a yellow background. It contains three bullet points: "Avoiding getting trapped in suboptimal local minima.", "Difficulty arises in from saddle points, i.e. points where one dimension slopes up and another slopes down.", and "These saddle points are usually surrounded by a plateau of the same error, which makes it hard for SGD to escape, as the gradient is close to zero in all dimensions." To the right of the text is a 3D surface plot of a saddle point, showing a surface that curves up in one direction and down in another. At the bottom right of the slide is a small video inset showing a man with glasses and a mustache, wearing a light blue shirt, speaking. The slide also features a navigation toolbar at the top and logos at the bottom left.

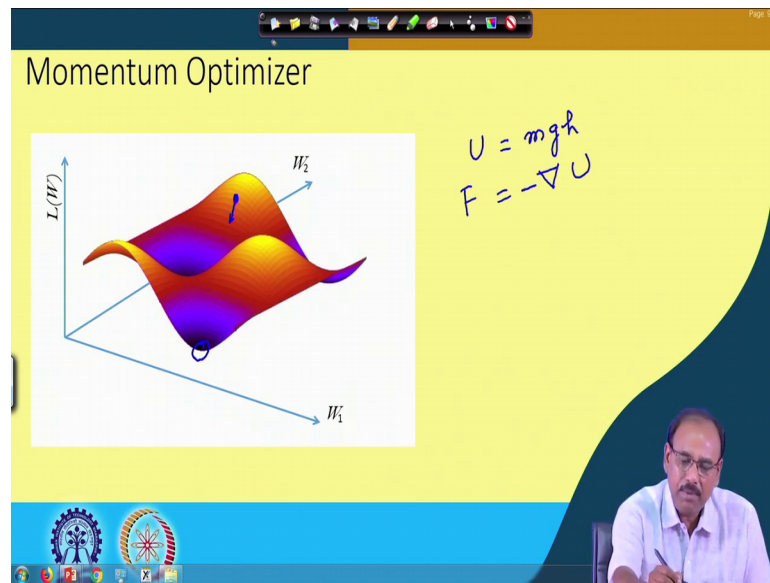
Gradient Descent Challenges

- ❑ Avoiding getting trapped in suboptimal local minima.
- ❑ Difficulty arises in from saddle points, i.e. points where one dimension slopes up and another slopes down.
- ❑ These saddle points are usually surrounded by a plateau of the same error, which makes it hard for SGD to escape, as the gradient is close to zero in all dimensions.

The other problem which is also very, very challenging is that my algorithm may be tracked in suboptimal local minima, particularly, if the error surface has a saddle point something as shown in this diagram say as over here. So, what are the saddle points? Saddle points are that surface that point on the surface where in one dimension the slope rises whereas in other dimension the slope goes down. So, it is over here in this direction the slope is rising, whereas in this direction the slope is falling. So, this is what a saddle point.

And the challenge comes because of the fact that this saddle points are usually surrounded by plateau of the same error over which the gradient values negligible. And when it is so, then the gradient descent algorithm or stochastic gradient descent algorithm find it very difficult to come out of this problem. So, these are the different challenges that we can have in gradient descent. So, now, let us see that how we can tackle this problem.

(Refer Slide Time: 25:12)



What I can do is, I can assume say this is the surface given by the loss function. And for simplicity I am assuming that my parameters are two-dimensional parameters or the weights are two dimensional weights having  $W_1$  and  $W_2$ . And the way you start your gradient descent operation is that you assume that initially the parameters or the weights are say initialized somewhere over here, say initial is the weights somewhere over here. And then what you do is you compute the gradient at this location and the direction of gradient is such that it falls along the valley and finally, it should come to the minimum location.

So, I can assume that I place a ball at this location with initial velocity 0 and that ball will have a potential energy which is given by  $u$  is equal to  $mgh$ . And the force which acts on this ball is say  $F$  which is nothing but gradient of this potential energy with negative. So, minus grad  $u$  is the force acting on this ball. And under the action of this force the ball falls along the surface, and while falling on the surface it gains the momentum. So, while falling it will drop down to it is expected that it will drop down to this minimum point on this surface ok.

The algorithm is fine if the surface is a well behaved surface or it has equal curvature in all the directions. But the problem arises, if we find the surface is not well behaved or the curvature in one dimension in one direction is more than the curvature in another direction, and that is where this momentum optimization technique which is nothing but



an approach for making your gradient descent more efficient. So, these are the cases where the momentum optimization technique will help in faster convergence of the gradient descent algorithms. So, let me stop here today. In our next class, we will talk about this momentum optimization.

Thank you.