Now, when you actually train a network right, so one way to get a, one way to, a simple way to think about it is like this right. I mean this is a nice example which you will find in you know most probably lectures right if you have heard on ok. So it is like this right, so suppose so what you are trying to do is the following let us say right I mean I have got a few points ok like this ok, something like this ok. Now this is all that I know that means I know that I know that right I mean you know for some value of x right I mean there is some value of y for another value of x this is y this is all I this is all this is all the data I have ok. And through a network right what you are trying to do is so for example right so there is some g of x right which probably you know which rules these points right I mean so there is a true g of x which we do not know right. And what you are what a network is trying to do is it is trying to figure out it what how to best approximate that g of x ok.

But then the fact is g of x itself we do not know all that we know is we have sampled g of x at certain points right we know that g of x has takes its value at this place right so much at that place so much that is what we know. But then we do not know g of x in its entirety and or anything right. Therefore what can happen is right you can so when you have when you have you know a network which is trying to get there now it can do it can do right 3 things one is this ok what is called what is called and say under fitting ok. So for example it could be it could just probably plot a straight line alright just take it as a very you know very very naive sort of an example right.

So let us say that y hat simply does w x plus b right so here is x here is your y and you know y at some places and this is this is the simplest thing right which you can probably do and you know that you know that it is not really probably right you know a good thing to do because this is under fitting. Now you can have let us say the other way right could be that or maybe write on this itself I will draw. So the other way could be that could be that I go like that and then I fit something like that right and this could be right so this maybe I will write this as y hat is equal to w 2 x square plus w 1 x plus b right. So we are just looking at a scalar case imagine all this is happening in some high dimensional space right in reality. So now this is like you know a second order sort of right a polynomial fit and you know it seems to fit well.

Now this would what we will call is let us say appropriate fitting okay in a sense that right this we feel is probably more acceptable than the earlier one. Again we do not know what g of x is right but then we feel that right this probably is a kind of a better way to approximate the underlying you know function. Then a third could be what is called what is called you see over fitting right. So an over fitting would right would look like this for example, right I do this and then I do this okay that is an over fitting in the sense that right you have something like y hat I hope you are able to read this it is let us say b plus a very high right high degree sort of polynomial right i equal to 1 to 9 w i x to the power i okay. Now you just force yourself to fit as closely as possible right.

Now the problem is right this is called actually over fitting and you know this is a problem

that you will keep on encountering okay whenever you do whenever you train a network one of these things right will happen. And the ideal thing is to be able to get somewhere right where even if you make some errors with respect to the training example that is still okay right nobody is saying that you have to exactly match those values. The idea is that if I give you examples outside of it right you should be able to kind of write tell me something about what those values can be. See for example, right I mean you know see suppose I ask suppose I ask right what might be the value there okay for y okay. Now right in this case in the case of appropriate fitting right I have this whereas in the case of this you know whatever right this over fitting case it looks like it is actually 0 or something whereas in case of the line right I get there.

So, the idea is that see you train a network so that it can work on examples outside what it has seen right that is the whole idea that is called a generalizability right. You should generalize outside the set because what is the point if I have to show right everything if I have to show everything then why do I even need a network. So, idea is that you should be able to show enough number of samples which are rich enough right that is why this richness comes. So, if you just show some lame examples and then hope that you know if you give a new example and try to learn something fantastic it would not happen. So, lot depends upon what kind of examples you know expose it to and those examples should be such that they actually tend to they actually help this network figure out as to what probably is going on right.

And therefore, when examples come from outside of that training set it should be able to generalize well it should be able to predict values that are reasonable okay. Therefore, over fitting is bad under fitting is also bad. So, what really you are looking for is that appropriate fitting and here even though I have shown that you know it goes through all the points it does not have to okay. You do not have to meet that you do not have you do not need a training error that is absolutely 0 or anything you are okay with that you can make a little amount of error there is a small margin of error that you can tolerate. So, that so that you can probably accommodate examples coming from outside the training set you can accommodate them accommodate in the sense that you can you can actually predict reasonable values for them.

I mean there is this nice result okay sort of you know what is called what is called an Occam's razor. Let me write that I mean you know I am not going to talk about it much, but I like this you know because what it says is this Occam's razor. So, it is a kind of you know a principle that says that among let us say competing hypothesis that that explain known observations equally well one should choose the simplest one. I mean it is a very it is a very very elegant statement right. So, it says that right among competing hypothesis that explain known observations equally well choose the one right which is the simplest.

And you know this also goes back to some of those some of the relations right in math if you look at the Euler relation and all. The ones that are the simplest are the ones that

actually happen to be the most elegant also right most of the time. Right you may you read a paper if somebody writes a very very convoluted paper right and then solves a problem at the end of the day right you still feel that you feel a little hollow right you do not really get it. Whereas you know let us say this one a Kalman filter for example, if you read the Kalman filter you know so nice. And then after all it is a I mean the structure is simple, but then the problem that that that was solved was big right.

So, the elegance rise in its lies in the right simplicity. Therefore it is all it is never a good idea to kind of put a network you know with hundreds of layers you know you know when somehow do my task and all know right. The idea should be to have the bare minimum that you need in order to solve a problem right. Now here is where comes this notion of notion of a cost surface ok. Now the the point is right.

So, you have something like a cost landscape right whenever you talk about a loss see what what what are you trying to do. So, when you talk about a loss function right this is what you have at the end of the day right all that you have is this loss function right. You want to you want to be able to able to approximate something it could be a regression task it could be a classification task at the right end of the day. Now this loss function itself right nobody knows what kind of a landscape it has right. Because I mean if you look at that m dimensional some high dimensional space it has some kind of a landscape.

And what you are trying to do is right you are trying to try to try to minimize this loss function with respect to let us say ok. Now with respect to w and b right let me not write theta let us say w the weights and then b which is the bias right. There could be whatever it millions of these weights and then you know whatever it millions of these biases, but let us say right together if you if you call them as let us say theta then theta is that is that kind of space right where you are kind of say traversing. And what you have see ideally right I mean if you if you if you had your wish right you would have you would have like to have a cost function like that right. But that is actually never the case first of all right I mean first of first thing is that because of the nonlinearities that are all sitting there right after all this is the deep network with lots of nonlinearities you know in the you know in the in the layers right hidden layers.

And therefore, definitely one does not expect this cost function to be as simple as that right you do not expect to be except it to be a convex surface. Now, but then the interesting thing is it is not true that right it is non convex because you have a nonlinearity it is it is non convex right inherently. Even if I make all the activations linear let us say right I do not even have this you know the step activation and all everything is simply linear right. Even then you can show that a network right even if you force everything to be linear even then it does not convex. And that comes because of something called a weight symmetry you know what is called a weight symmetry feature.

What that really means is that I mean right this I leave it to you I want you to kind of right

think about this. So, suppose I have let us say 1, 2, 3 and then and then I have let us say 1, 2 which is which is my say this is a hidden layer. I will show for I will just I will just you know illustrate for one neuron the same applies I mean if you have multiple neurons at the output. See for example, right so you know that you know that right you have these weights right coming from this to this to this and then similarly you have these weights right. Now and similarly from let us say here to here.

Now if you swap the neurons and the weights right what this means is by which I mean the let us say let me just indicate this as 1, 2 and then 3 and this is your x 1 what is it x 2, x 3 right. Now if you swap the weights in the neurons that means right 2 goes to 1, 1 comes to 2 right that means you swap like this right and here of course, it is just 3 if you had more you would swap those also and then if you swap the weights ok that means right I mean you know what is kind of rate going from here to here will then go from here to here. So for example, right I mean see when 2 comes here then you see this weight is what we call this as w 2, 1 right. See when you have this x 1 to 2 node connecting to 2 we call this as w 2, 1 right. So, when 2 goes up there then this weight is what this w 1, 1 right then that weight will then be at w 2, 1 when you swap right.

So, when you swap the weights and when you swap the neurons and similarly right here this would have been w 3, 1 ok. So, instead of you have to swap the weights now. So, up will come w 3, 2 down will come w 3, 1. So, if you swap right if you swap the weights and the I mean neuron that means the bias will also get swapped ok everything will get swapped. When you swap it right you get the same output ok which actually means that, but then if you look at the look at look at in that space right what new set of weights you are looking at that is not the same as what you had what you had right originally.

So, if you look at each arm right what you had was let us say some weight it going now that weight is changing. So, it is like saying that right in a surface so, if you can think of this surface right I mean we nobody knows how that surface looks like, but then right you can think of it as something like that ok. Where this is the same I mean you have one weight configuration for which you hit the same minima you have another weight configuration for which again you hit the same loss you can have you can have right as many as many numbers I mean permutations right which you can do. And this is not the only symmetry that you have there are other symmetries also and I am saying right this is even if you have everything linear you do not have to bring in nonlinearities at all ok. So, that way a deep network the moment you talk about a deep network non-convexity is simply right inherent ok.

There is a it is not because you have nonlinearities and therefore, of course, throwing in those nonlinearities right makes it makes it more complex right. But inherently it can be shown that you know deep network you know will have there are other ways you know in which you can have you know these kind of you know this kind of non-convexity right. And therefore, therefore, right what happens is, but then at the end of the day end of the day

right we do not care about it because any of these is for us right we do not care because the loss is the same right. So, I do not I do not have this one a preference for this weights right over the over these or over this I do not because as far as the loss is concerned right it is the same right. And therefore, right looking for a global minimum and all is simply out ok.

So, maybe in your other courses right you might have looked at nice convex functions and then we talk about a global minimum and all right nothing here ok. We are all happy as long as and then the the the red beauty is that you know there is so much you know so much what do you call you know so much room to go around and be able to still get things such that they are acceptable right that is what that is what may that is what lends strength to these to these right you know deep networks ok. I will stop here.