So, you had something like a x right belonging to let us say a set of this one p labels right I mean which is a positive examples. Then the idea is that your you see w transpose x right should be such that you want this to be kind of rate greater than or equal to 0 right and if it is if it is if x belongs to negative examples then ideally right you want this to be less than 0 right. And the idea was that I mean you kind of update it right. So, in such a manner that when this condition is not satisfied that means w transpose x is actually less than 0 then you make this update right you say that w new is equal to let us say w old whatever you had plus x right and for the other case right and so the idea is that right I mean you get effectively w transpose x plus x transpose x and since you since you apply this condition only when w transpose x is less than 0 that means you want to get a boost of that value right. So, that boosting happens right through this x transpose x which is actually greater than or equal to 0. Similarly when you have negative examples and you have actually greater than or equal to 0 that is when you apply the update rule you make it w minus x and which means that you it becomes w transpose x minus x transpose x which means that right you pull that number down right.

It is already greater than or equal to 0 you want it to be less than 0. So, you try to get a pull it down effectively by taking w transpose I mean w to be equal to w minus x. So, that the w new transpose x right becomes w transpose x minus x transpose x, x transpose x is always a number greater than or equal to 0 and therefore right you pull it down okay and then right eventual idea is that you get down get to a point where which then means that you are see cos alpha being equal to w transpose x by norm w into norm x right. So, the idea being that you are see w transpose x should eventually come out to be such that your so if your w transpose x goes up then your then which basically means that your cos alpha is going up that means alpha is coming down.

So, you sort of end up making acute angles with respect to the positive examples and then obtuse angles with respect to the negative examples right that is the idea and of course the convergence proof and all right is really beyond the scope of this. Okay now the rate universal approximation theorem right which we stated last time. So, what it is saying is so it is like saying that you know if I had examples x right and I have to actually write build a network such that I arrive at some let us say rate g of x whose analytical form I do not know because if it was analytical then it would be far more easy to solve that problem right. The idea is that you do not have an analytical solution but you have but you are kind of looking for a this one mapping that will take you from x to some ideal g of x and through the network right you are arriving some there is a guarantee that through the network right you can arrive at I mean all that that we stated last time that you have now one this one h l right hidden layer which rate if it contains sufficient number of neurons with whatever activations then you should be able to approximate g such that mod g of x minus f of x right. This is let us say less than a positive quantity small positive quantity let us say epsilon for all x time this is what this is what the rate universal approximation theorem says but then it does not say as to how many neurons you need and all that it just says that as long as you have this is sufficient number we should be able to approximate right I mean it does not say

what right I mean it does not give a representation it does not tell how what is the best way to achieve this right which is the reason why for example right if you find out deep networks right the very reason why we call them as deep is because they have several layers stacked right one after the other and actually the actual implementation happens through that and not really through one sort of a hidden layer right and that would have I do not know right millions of neurons or something.

So really what happens is right eventually people figure out that the way to actually achieve this because the even theorem per say does not say how to get there right and that has happened over years by let us say right people figuring out how to do this okay that I will just right come to come to that in a moment but before that let us just look at then what would be a structure of that kind of a network right. So what this means is that I have inputs let us say x1, x2 and all the way up to let us say xn okay so this is my input right that goes to my network and I have this one hidden layer okay according to UAT of course we are still at the UAT point we are not looking at a deep network in that sense but we are looking at a network right a neural network that can achieve the you know task at hand and what you are saying is right we got let us say right several of these of these neurons sitting here and each one of them is fully connected right I mean that is the whole idea. So each neuron will get all the past inputs from all the kind of right previous in this case all the right inputs. So if you look at the first neuron right it will get from here, it will also get from here and then right all the way here and then each one will have a weight right. So the weight is needed because sometimes right some of those inputs okay may not may be totally irrelevant to the output right and see for example right you can kind of right think of examples where let us say one of the inputs is not even right it does not even matter as to what that value what value it takes with respect to the output right.

So the weighting is really for that so this network should figure out along the way as to which ones really matter right for example you could have something like an output, output it could be the could be the right rating of let us say right a movie and then the input could be actor, director whatever right and then it could be that could be the right one of the one of those inputs could be simply right I mean you know rain or something right weather I mean how does that matter right you know with respect to movie. So these weighting is really happening so that you can actually so the network figures out right which ones really matter for us to be able to get to the output that we want and then the same thing would also happen to happen let us say with respect to the other neurons right. So you again have to do a connection from here, from here, from here and then the direction of course I am not showing but it is obvious right it is a kind of it is called a feed forward network right so you are kind of feeding forward all the information and then what happens when you have let us say an output layer again output right does not have to be a single value right. So this f of x or this g of x for that matter it could also be you know a vector valued function okay which is why I said I mean right I did not explicitly mention that but then it can also be vector valued. So really what you can have is something like this right so you have an output layer so this is your input layer let us call this IPL let us call this HL right hidden

layer and let us call this the output layer okay and again the output layer so you see each of these neurons here will get an input from all of the right previous you know previous output so which then means that this guy would connect to this, this guy would connect to this all the way, this guy would connect to this and then maybe right what will also happen is this guy will get input from all of them right and so on right.

And you can imagine that the right number of unknowns that you really have to solve for can really blow up right rather fast. For example I think of something like I mean if you have an input right which is let us say of a certain size right I mean you know if you have let us say m right m cross 1 right if that is the right I mean you know this one vector that is going in and let us say I have got some n cross 1 neurons right n number of right neurons and where let us say n is far greater than m typically right that would be the case and let us say my final layer which is output layer let us say I got some like let us say right p cross 1. Then you can imagine that in going from the input layer to hl right I mean you know hidden layer you are already looking at how many weights, how many weights do you have there that you have to actually calculate that is m into n right m into n number of weights and then the biases will be n number of n number of bias terms right because each neuron will have a bias. So from going from here to here right you are looking at m into n weights plus n number of these biases right because each neuron you also have to find the theta right for each one of them theta 1, theta 2 whatever right or right up to theta n. So you have that and then in going from here to here right you are looking at what is that n into p number of weights right and plus biases for the output neurons which is like p right.

So now given I mean if you have these numbers as large which is right typically the case right then you are looking at really learning that is why that is why when you kind of look at these deep networks right they end up you end up trying to solve for you know so many unknowns that is because I mean there is just so many weights out there right which you have to find out there is so many bias terms and and so on right and that is also the reason why let us say right when you when you when you handle images right this kind of an architecture is no longer the best because an image typically is like 512 by 512 and and you know and actually there is a spatial correlation and all therefore you know it is not a great idea to just stack it up as one single vector right what is called a lexicographical ordering. So you can just it is unwrap an image and put it as one long vector but that is not the way it is done because then you do not capture well you can expect the network to capture all that but that is not the way it is done right and you want some locality and all that that is why you have what is called a convolutional neural network right which comes later. Now this kind of right this this kind of an architecture right is called is called fully connected FC right fully connected I mean there is also something called convolutional fully convolutional networks. So that is typically called FCN so one should not confuse that that and this so when I say FC this means fully connected okay and yeah and here is your output by the way right y1 y2 all the way up to let us say yp sorry here yp okay. Now yeah so number of weights and biases right is done now then then why then right one could ask in fact right in the last class even after the class some people are asking then why do you then do a deep

network right so why do not you just solve the problem like this right why do not you just have one sort of hidden layer and try to compute all those weights and biases.

Now this is this sort of rate you know this is in a sense empirical and some of it is also because of the way that right one needs to have some kind of you know attractability right I mean attractability in the sense that when you train a network you should also know right which way to head because it is never very clear as to you know how to kind of do this and you also want some sort of an right explainability in the sense that we should be able to associate things that are happening right inside the network to things that you really want to happen. See for example right I mean you know just as you know inside a human system we have this idea that let us say right there are kind of right low level features right which we initially gather then maybe right they kind of then you know there is a pooling of those features in order to arrive at a certain higher level of features then there is probably another level of you know some sort of a pooling that happens you know which kind of brings those features together into a further whatever right whatever you mid level and then high level and then then okay then there is then you get some interpretations. Now similar to that right I mean here also we want so so right over the years what has happened is right people have actually figured out that that something like that right helps you actually achieve exactly the same task but at the same time right so the reasons right I will just write down and most of them would be obvious but I think it is just good to know so why we why we actually you know so instead of this what typically done what is typically done is in an implementation right what you would typically do is the following right so you will have you will have let us say X1 to Xn what is it Xm sorry okay so this should be m. So X1 to Xm okay is what you have and then what you have is typically a bunch of layers right I mean sometimes these number of layers could go up to 50, 100 and so on it depends on you know it can it can beyond a point that it can be you know intractable also but yeah but then typically right after a certain number of layers you may not even find any advantage but it is interesting that you know I mean I was in a I was in a talk right that one of the CVPRs and there was this guy who works in actually brains then he was I was asked to give a give an invited talk right and he said that you know that actually none of these architectures really mimics what the brain does okay. So even though we tend to think that you know this is like the human visual system and all but he says human visual system does not do a back propagation and all that and when the way we solve let me see one thing is that you can actually have an architecture that you think is more you know is more is more is tractable is more is a kind of convenient thing to handle but at the same time it should also know how to train it and so on right.

Now people have figured out how to how to train it and so on but then he says that that is not the way and nobody knows okay by the way he himself does not know how the but then he is sure knows for sure that this is not the way right things work inside our own system may be true right because very unlikely that that right we would do something like this but anyway right you know given that whatever has happened has happened right in this manner one can write or some of you might still want to think about you know this is the

best way to do it right I mean nobody is saying that this is the way to solve this problem but this is the way it has been solved right along the way and I think we are just trying to trying to sort of write interpret what has been done right we are not trying to say that right this is the best way to do things. So what is done is right so you have 1 layer, 2 layer okay I already wrote down right so you got multiple layers and then comes your output layer right what is that y12, y2 up to yp right now so the idea is that right I mean you know you might want to sort of write you know come up and so that is the way this is done and because of this because right anything more than 3 or 4 layers you would call it deep okay and deep network right so really that is what we mean by a deep network and this does exactly the same thing right same task that you want in the sense that irrespective of you know you can think of the problem that you are trying to solve itself could be a regression problem for example right you are going from an image to an image or something like that or it could also be like you know you are doing a classification problem where let us say right given an input image you want to say right to what class right you know that that from what class that image comes whatever be it right. Effectively the deep network is trying to solve right I mean either a regression problem or a classification problem right these are the 2 main problems that let us say right you know people deal with and yeah and the reason right for actually for sort of going from sort of transitioning from here to this is for the following reasons right I am just going to write down okay so I will just write down so it gives a compositional feature abstraction compositional feature abstraction feature abstraction right which basically means that you know see for example I mean if I call this as let us say right I mean something that is actually emulating some F1 sort of a functionality and then let us say this is the second guy F2 and then so on right. Now what really is happening suppose I call the input as some vector X right then what is happening is after the first layer right you have let us say F1 of X right and then F1 of X becomes the input for F2 and therefore F2 acts on is F1 of X right then you might have an F3 that is acting on F2 of F1 of X right so you got like F3 acting on F2 acting on right F1 acting on X right. So you have this kind of a compositional abstraction right which is actually going on and the idea is that this compositionality is somewhat mimicking a human system in the sense that like I said right like I have said several times earlier also that this sort of gives you a you know mimics the behavior of a human system but again to a certain point okay and you know each one tries to figure it out in their own ways you know which architecture works the best for a particular problem okay.

But I am saying this is just a you know general what you call right a general sense for how these things work okay so compositional feature abstraction is this one just as in the human visual system as in HVS which is human visual system human visual system. Then okay now UAT itself right does not UAT itself right limits itself okay you know so right I mean itself does not does not say how easy it is to learn right does not say how easy it is to learn this is representation or learn the mapping right. It just says that well you can do it right but then it does not say whether I doing it with just one sort of hidden layer is that the best way to achieve it or you know should we have multiple number of layers right effectively achieving the same thing but then right which one is easy to learn is not clear right then you

know it does not even say right you know write anything about that and the fact is that people have found that you know solving this there are there are say efficient ways of actually doing it okay. And or right it also it also right has I mean and then it does not say does not say how many how many neurons are needed does not say how many neurons are needed right. It only says that representation is possible representation of any g of x you can apply approximated by some may say f of x as close as possible right does not say how many neurons you need for a further task how many neurons you need.

And by the way right so this output layer that you have okay does not always mean that you need a non-linear activation there right but all this non-linear activation is all about this okay in fact what to say and even in the earlier example right where I think we did watch some or something we did right so even there I chose an activation that the output right but then you can show that right it is not even okay you do not really need that okay. So I mean you can have other ways of doing it also so and the other thing is right depending upon whether you are solving a classification problem or you know whether you are solving a regression problem right this output layer could have an activation function could turn out to be totally linear and so on right because again I mean you need a bunch of values that that that you want to work with right it does not have to be a probability right all the time okay. So, so therefore what you have in the output layer is activation depends but all of this this hidden layer thing right so the activation is all is all for that okay it does not explicitly say that you need something at the output you need a non-linear activations there then yeah each of y1 to y2 can take only 2 values right. Each of y1 to y2 can only take 2 values no no it is not like that right depends upon what kind of an activation function you have see for example if you have a step activation right like the one okay which we had then of course okay then you have some like either it can take a 0 or 1 it just fires that way but if you had something like a like a let us say sigmoid right it does not have to be yeah it can take it can take you know it is a real values in fact if you have if you have relu right it can take it does not have to be limited to anything between 0 and 1 right I mean you know relu right would look like that I mean so it is totally linear after 0 so it can take any value. So that is why I am saying we are not limiting ourselves to what is happening in y right what you put in y depends upon what task you want to solve okay.

Then there is one more thing right that you want to say abstraction then layered representation of features so from simple to complex right so you can also so this compositional feature is one aspect of it and then you know you can also think of it as some kind of a layered representation right layered in the sense that okay layered in the sense that right you typically believe that these features will go from being simple to complex okay so that is simple to complex right. So that means initial layers would be kind of would be learning some simple features and then as you go further right down there down the network right I mean what is being found is that you know typically that is where the complex features you know end up being and that is also the way our own our own kind of visual system works so it is like saying taking simple things which can easily identify and then and then right and then you know try to try to then you know go for more complex

features. And what is also been and then the other thing is right explainability okay explainability because that is kind of very very important right because if you just do something right and then you are not able to explain why it works right it is not good right I mean it is not enough if you simply show that and I am able to meet my target right I have built a network which can solve the problem but then you have no insights into why that is going on that is why you have what is what are called these ablation studies and all right I mean you know so right somebody wants to know what happens if this is not there what happens if that term is not there what happens if this layer is not there right if you remove it why do you really need it there and so on. So therefore this explainability right is again is again is again very very important I mean otherwise you just are just solving something like a black box right and therefore explainability matters and then the other thing right that that is more of an empirical evidence is that you in fact end up using fewer neurons than a single layer okay need I mean this is again okay this is all empirical empirical evidence right I mean again you know if you want I mean you should probably go and further read about these things but but what has been typically found is you need fewer neurons as compared to a single layer okay. So instead of having just one sort of hidden layer what has been found is in fact you end up end up using fewer number of neurons okay if you have a if you have a deeper network so this is again empirical evidence okay this is not this is not any there is no kind of a theoretical guarantee or anything just an empirical empirical evidence okay yeah.

So right which is the reason why when you see these networks right they actually end up being what they are okay like deep networks. Now the 2 main tasks right which one looks at is what is called a regression task and what is called a classification task okay. The regression examples are many for example you know okay now let me just it does not have to be just straight image based even though this course is about images and so on so regression one example of image could be like I have an image which is let us say noisy right and I want an image right which is actually clean I want an image to come out that I want a network which should take this and then right sort of suppress the noise and then it give me you cannot say remove the noise right that is not a good thing to say because you can never remove noise but you can maybe suppress the noise right and you know get out an image which looks far more clean that is like that is like a regression problem and similarly you can have several examples but even in 1D right I mean if you want to talk about because right now we are still you know talk and okay one more thing right that I that right before I forget to say right. So such a thing is called an MLP which is a which is a what is it called MLP? Multi? Multi Layer Perceptron. Multi Layer Perceptron right MLP Multi Layer Perceptron.

So you so you have all heard about this, this is called an MLP right multi layer right it is like right multiple layers of a network right so you called a multi layer sort of a perceptron and when you talk about convolutional neural networks they are called CNNs and so on right so those are those do not have this kind of right this kind of fully connected sort of you

know behavior okay. So from now on whenever we say MLP this is what we mean okay. So the idea is that right let me just go back to.