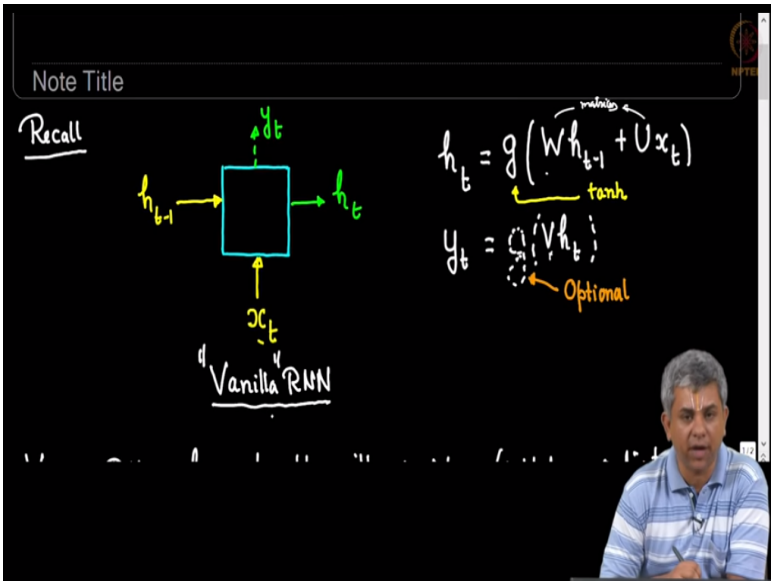
Machine Learning for Engineering and Science Application
Professor Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology Madras
RNN Architectures

(Refer Slide Time: 00:14)

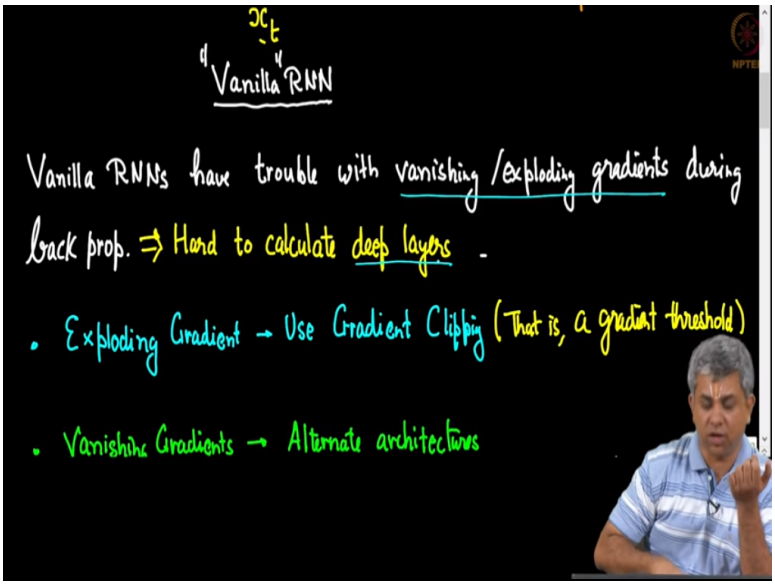


Welcome back in the last few videos we saw that vanilla ordinance can suffer from several problems such as exploding gradients as well as vanishing gradients what we will be doing in this video is to look at few alternate RNN architectures they are slightly more complicated than the plane architectures that we saw in the last few videos so just to recollect what we have done so far we looked at the plane or vanilla RNN as it is called it is a very simple idea you have your input coming from here and you also have a previous layer contributing from here.

So the input layer we call as XT and there is also an input coming from the previous layer which is HT minus one will T is the level at which we are looking and the output or the input to the next layer is HT and the output here is YT, now the formulation we saw was very similar to what we have been swinging whether with bitter CNN or ANN it is simply a linear combination in this case W and you are matrices weight matrices this linear combination followed by an non-linearity I had also mentioned that the typical non-linearity that we use within RN ends is a tannic layer, now the output which is optional you can take out the output at any point.

As we saw in the various combination of parents or various characterizations of RNN's that we saw in the previous videos you have an optional non-linearity might be simply linear or it could have some non-linearity there and time yet mate with another matrix multiplied by the output at HT, so this is typically what is defined as YT so this I shat we call a vanilla or a plane RNN.

(Refer Slide Time: 02:12)



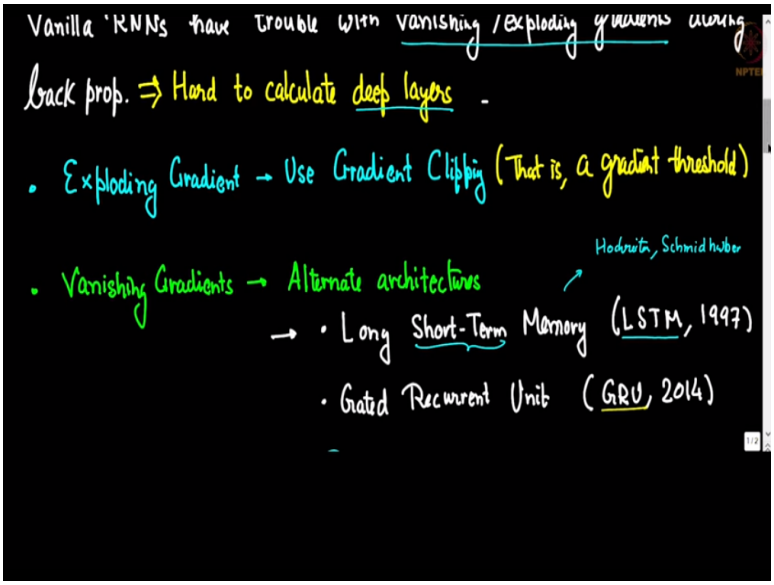
As we saw in the previous video vanilla RNN's have trouble with either vanishing or exploring gradients during back propagation why is that because repeated operation of this sort as we saw in the back propagation time video repeated operations of that sort can actually make the gradients either increase continuously or decrease continuously depending on the eigenvalues of these matrices.

So this make it hard to calculate deep layer and this difficulty goes back to the fact that machine have finite precision if you did not have finite precision even a vanishing gradient would not go exactly to zero and you will learn something but typically what happens is if the vanishing gradient become very small and smaller than your machine cutoff or machine epsilon you are not going to learn anything and you are learning algorithm is basically stuck when it uses some variation of gradient descent similarly with the exploding gradient it is possible that it might rise really high and then drop but that would not happen because you will always hit the maximum limit of machine largest numbers and will grow up unbounded.

So you can have various problems but in either case for all sort of practical algorithms vanishing as well as exploring gradient is a huge problem and you are unable to train deep layers what do we mean by deep layers lots and lots of deep layers so typical language task can use to 5200 layers and even engineering tasks might require that many at least a few of them in such case you will find it really-really impossible to rain this and that was situation till the 90.

So vanilla ordinance were not used very popularly for anything other than toy tasks the for an exploding gradient there is actually a method as we saw in the previous videos you can use gradient clipping that is you can hit a maximum size of the gradient and make sure that all gradient do not cross this threshold.

(Refer Slide Time: 04:14)



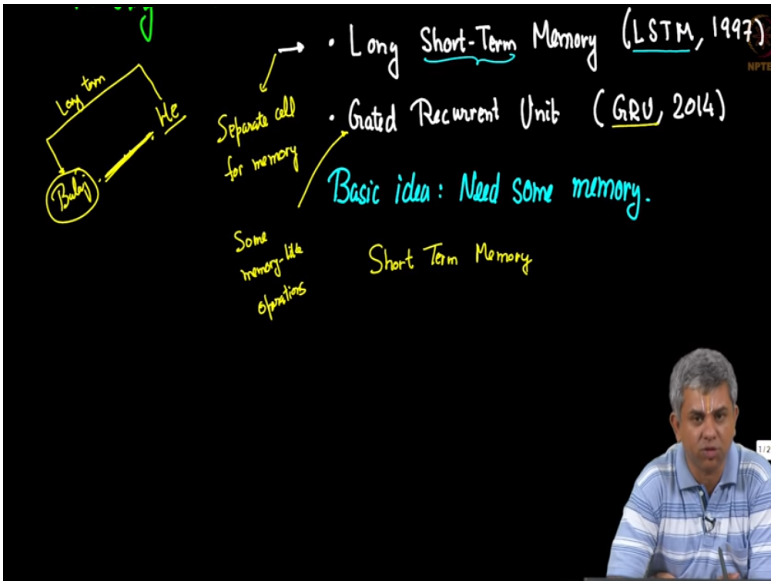
But vanishing gradients usually there is only one solution which is use an alternate architecture that is you tune this methodology of simply G of W HT plus UXT minus one and the earliest work on this and still very-very popular architecture is something called LSPM the humorously name long short term memory.

So the short term sit together and I will explain it shortly so long short term memory this is by writer and a very famous researcher in this field called professor SCHMID HUBER so this is in 1997, so SCHMID HUBER has worked extensively in this area and has well been very many other area also he has sparked phenomenal contributions in the field of machine learning recently

there is a slightly simplified version of this called the gated recurrent unit also called GRU for short.

LSDM and GRU for short and we will look at both this architectures in this video and coming video in order to see how is it that they will help the deep layers of training part I want to mention that it is recommended that you look at the original papers they are a little bit hard to read especially the LSDM paper is quite hard to read we will be simplifying both the explanation as well as the expressions that we will show here in general when you use RN ends typically you use either LSDM or GRU where you use still a matter of heart but we will give a few suggestions in the next few videos.

(Refer Slide Time: 06:01)



So what is the basic idea the basic idea with which LSDM and professor SCHMID HUBER had started was the idea human being do not simply sequential process something so if we say something like Balaji is taking class he talked about machine learning, so this sentence in order for he to workout you need long term dependency as we saw in the previous videos and this he relates to the person who is speaking and there is a lot of stuff in the middle.

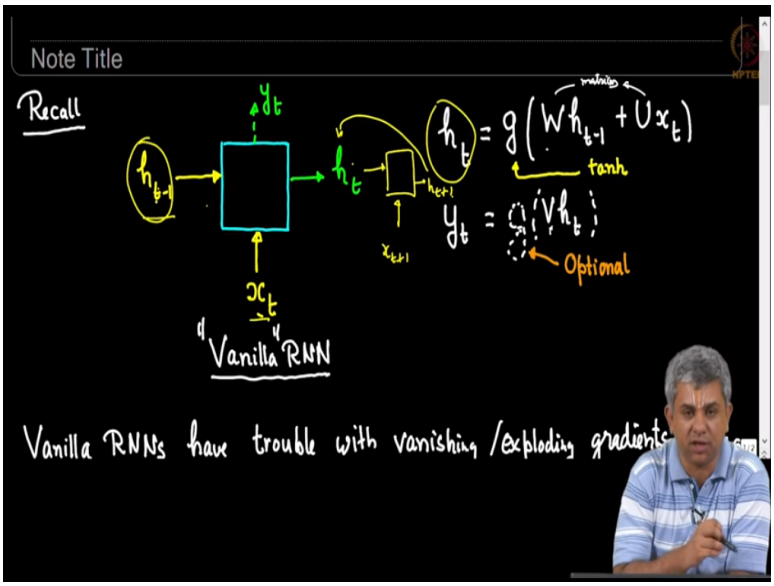
Now brain does this is using something called short term memory, now within the cognitive science as I understand it there is a lot of research on how many items we can keep and the general consensus based on paper of few decades ago is something seven items is something that

we can keep in our mind in our short-memory these are sort of our small computation were rambling so if I give you too many items or if I give you a long phone number or long number you cannot remember it but you can remember about six or seven items in a short term space which we can immediately access.

So what is also understood is the brain has two sorts of mechanism or at least we think that there are two sorts of mechanism there is something called short-term memory and long term memory, and the algorism that we have used here effective only has short term memory of one which is HD immediately the immediate value uses is simply that of XT and HD minus one but it does not store of the context where HD minus one came from this is a very rough example or this is a very rough explanation of what happens, so the basic idea here is to either have a separate cell.

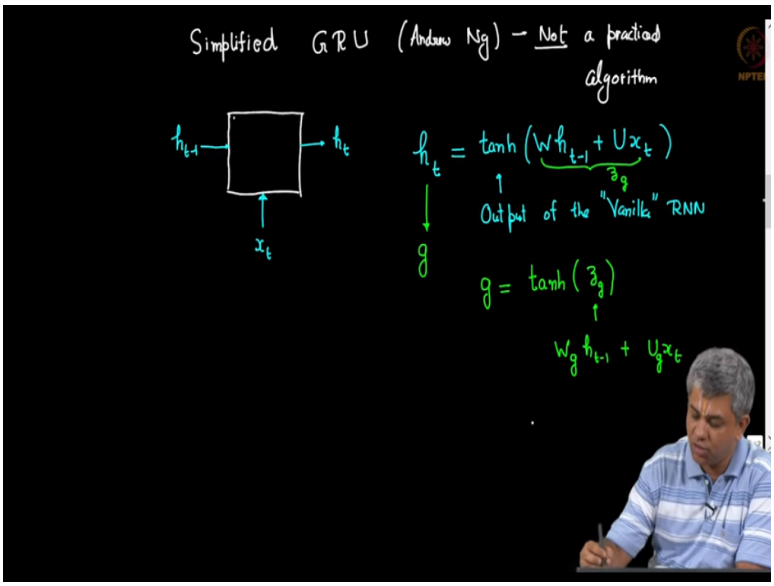
So this what Smith Huber did in LSDM which is to have a separate cell for memory and GRU also uses some memory like operations, so just to explain and I will show this mathematically shortly if you see here HT depends on HT minus one and XT and when I move on to the next unit HT plus one explicitly depends only on HT and it has completely forgotten HT minus one there is of course some portion of HT minus one hidden in HD but BT plus one in general has not using HT minus one explicitly.

(Refer Slide Time 09:03)



So what GRU and also LSDM do is trying to keep some portion of even prior computation store, some of you who have done hydrating methods might know something like relaxation schemes or successive over relaxation that also works on a similar principle as you will see in the formula that will be shown.

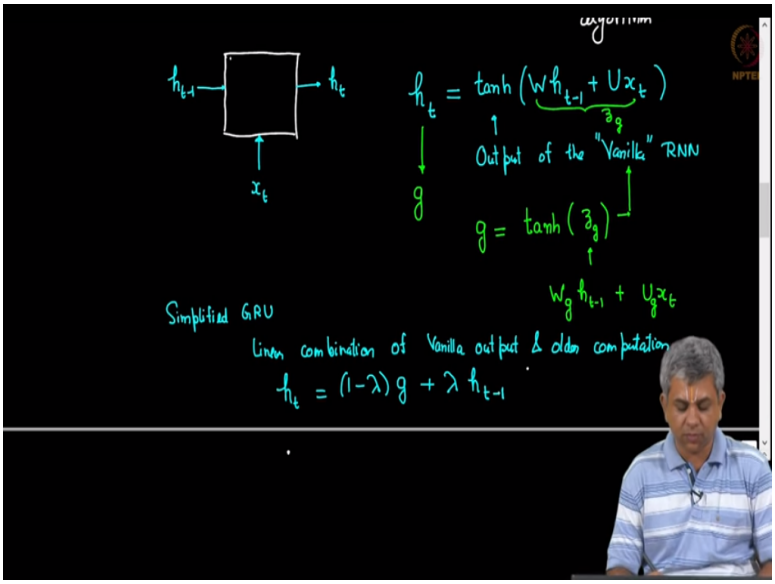
(Refer Slide Time 09:24)



So let us come to the idea of GRU, now in order to explain GRU we are going to use a slightly different version which we will called simplified GRU this is thanks to professor Andrew Lang just be clear this is not a practices algorithm so this is something we are using in order for explanation the actual thing in practice is GRU, I will be show you it is a slight modification of the GRU architecture okay so let us come back to our original vanilla RNN architecture we had XT, we have HT minus one HT comes out so we say I general that HT is tan H of W HT minus one plus UXT so this is the output, of the vanilla RNN.

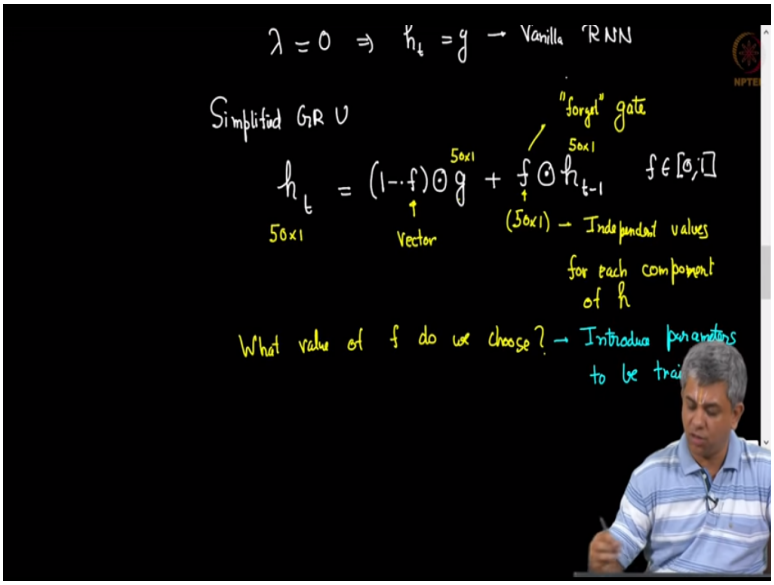
So simplified GRU does is labels this instead of calling it HD we will call it G, so G is equal to tan H of this term, we are going to label this something else well remember the linear combination is also always called Z, so we will call it ZG, where ZG is equal to W HT minus one plus UXT remember W and U are matrices and for a particular reason instead of calling them simply W and U we will called them WG and UG.

(Refer Slide Time 11:55)



So remember up until now nothing has changed this is simply the output of the vanilla RNN, so if this is G what is HT, so simplified GRU works in the following way we will take a linear combination of the vanilla output and older competition, so what does that look like it is looks like this HT is equal to one minus lambda time G plus lambda time HT minus one.

(Refer Slide Time 13:03)



So notice this, this is some scalar for now I am going to modify this shortly a linear combination this is our vanilla output and this is my older output how does this help what we are saying is I will not simply take, whatever I have got right now but I will also retain some portions of whatever of whatever competition I did before going back to the example that I use here, what we hope is in HT minus one some portion of this term Balaji is stored all this is very vague but much like CNN's all these people all this algorithm were made with several heuristics you will see that there will be a separates cell for memory in long short term memory but I wanted you to get the overall heuristic idea which is retain some portions of the old computation also add a new computation.

So when is this a nice linear combination, when lambda belong to is simply an interpolation when lambda is a number between zero and one, now this is not quite simplified GRU i am going to write the form shortly but remember what this functions like you can think of lambda as if it is a valve , the valve is like this when the valve is turned to one you only retain what was done before, so HP is simply equal to HT minus one it is pure memory, so lambda is equal to one you have HT equal to HT minus one this is pure and simple memory, no computation at all when lambda zero, then what happens HT is equal to G which is you get plane RNN.

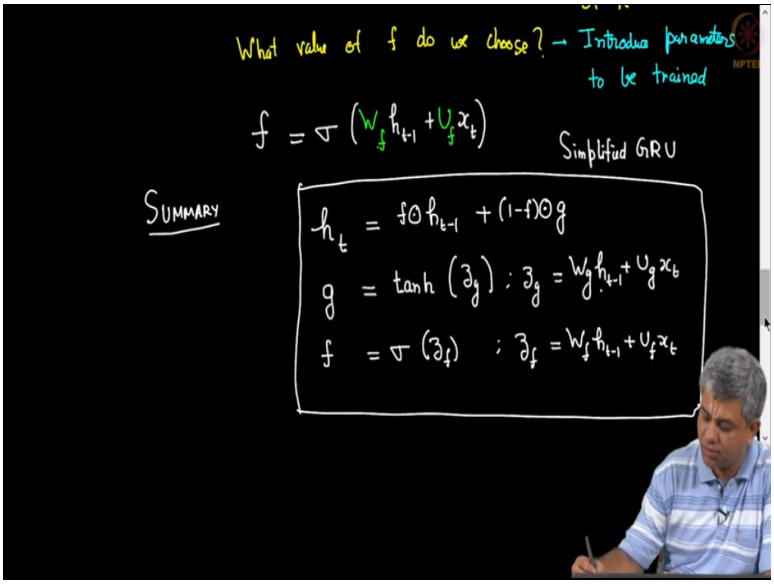
So you can think of simplified GRU effectively as a linear combination of just memorizing and not saying anything every word will mean the same thing if you are using a language task, or pure RNN, so it is a linear combination it is sort of an interpolation between a pure memory task and a pure computation task, so that is what simplified GRU does, so let us look at the actual simplified GRU exploration, it is like this HT is equal to one minus F Hadamard product G plus F Hadamard product HT minus one, so this F just lie a wall this F is called the forget gate, once again remember when F is One you will totally get memory and when F is zero you will get vanilla.

So now is the distinction between here and there lambda was a scalar here F is a vector just for the sake of clarity let us say HT was a 50 cross one vector let us say G was also a 50 cross one vector and HT minus one was also 50 cross one then in order for the Hadamard product to work F also has to be 50 cross one , now why would we take F to be a vector because it is possible that you might want to remember a few thing and forget a things within a vector , within the H vector

so what this gives us independent I will call them wolves for each component of H, that is all fine but what value of F do we choose and here we use the general principle of whatever we have been doing in neural networks which is we never really specify any component we let the algorithm chose it.

So this is introduce parameters to be trained it is the sane as ANN weights it is the same as CNN colonel weights similar to that we will assume F is full of parameters that ought to be trained but remember F also has to be between zero and on, why is that only then does the linear combination work out well there only then does it look like interpolation.

(Refer Slide Time 18:28)



So if I want F to be between zero and one I you remember from logistic regression that we have one function that always squeezes, any function into zero and one, so the same principle apply here , so we will keep F as a sigmoid of something, a sigmoid of what same we have linear combination of only two vector here at any place you have these two vectors HT, HT minus one and XT and you make them a linear combination of this I will keep another W another U but I wanted to distinguish it from the W and U I used here which was WG plus UG, so I will call it WF UF.

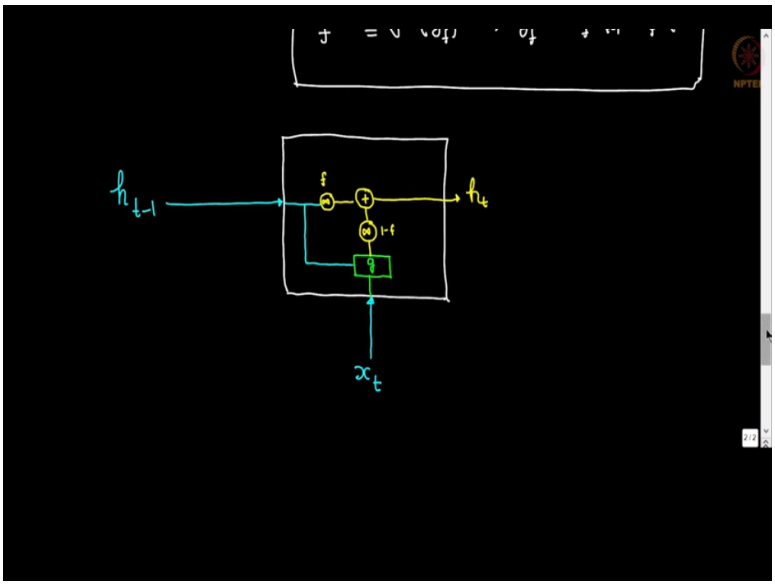
So you will see this theme repeated again and again and I hope that if and when you are in the position of making an algorithm of your own you remember that there are very simple things that

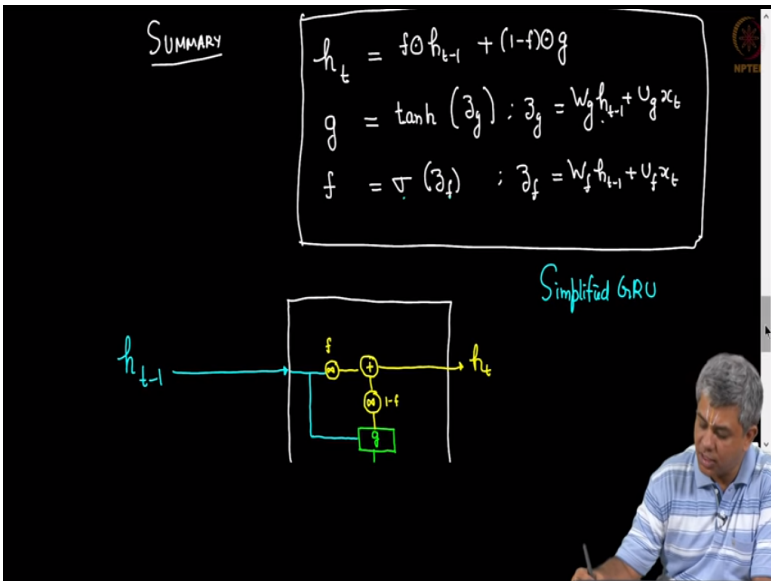
are going into any sort of neural network what are the basic ideas that we have seen, so far linear combination no-linearity, so here is a linear combination here is the non-linearity how did we decide on the non-linearity because we have some idea of what F should behave like F should behave like a wall, so I wanted between zero and one, so the non-linearity I will uses sigmoid.

So let me put it together and hopefully it will become clear then, so the summary is as follows I want to say that HT is a linear combination or some sort of linear combination of HT minus one and G where G itself is effectively the output of the vanilla RNN, we decided the G was tan H of WG HT minus one plus UG XT F was sigmoid of ZF where ZF is equal to WF HT minus one plus UF XT in order for us to see this symmetrically I will even erase this and call it tan H of ZG, where ZG is WG HT minus one plus UG XT, so this is essentially simplified GRU.

Now it is conventional for us to represent all these new architecture using a figure I am going to makes a figure of my own this might differ from the ones that are there online or wherever you find in other textbooks but this is just for a pictorial visualization of what is going on some people find it clearer to use this some people prefer to use simply the formulae that I have written before, so let us see this so what we have is as before we have HT minus one coming from the previous cell we have XT as the input.

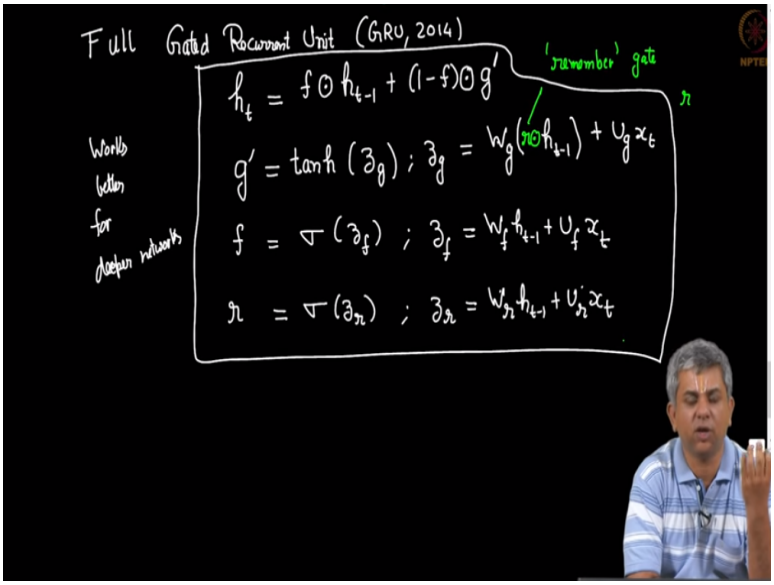
(Refer Slide Time: 22:32)





Now the two combine, so let us take this in the two combine to give I will call it G itself, so if you take tonnage of ZG you get G, now from G you go a little bit further there is our valve which is multiplied by one minus F similarly HT there is valve which is F go further add the two together and what comes out is HD this I is the straight forward figure expressing all this here remember even F is simply sigmoid of ZF where ZF is a linear combination, so all expressions that you will see within neural networks I have said this probably 100times through this course are simply one linear combination followed by a non-linearity that is also true for this simplified GRU but as I said before simplified GRU is not the algorithm is not in practice.

(Refer Slide Time: 24:16)



What is in practice is the usual gated recurrent unit it is a small variation of what we had in simplified GRU so this I will call the full gated recurrent unit, so the expression for that is very similar to that the one that we had before with some small variation s so this portion is this same F time HT minus one plus instead of one minus F times G which was the vanilla RNN and output this is a modified RNN in an output G prime and I will write the expression now G prime is tan H of ZG F is sigmoid of ZF instead of equal to I will put semicolon I guess that is more appropriate ZF is as before WF times HT minus one plus UF times XT ZG used to be WG times HT minus one plus UG times XT and now there is a small change to that.

There is another valve here is called the remember gate or the remember wall so instead of simply having HT minus one you add on yet another parameter there are HT minus one plus GT time HT are also should be between zero and one now what do we do about our it should be trivial by now for you to understand this it is simply sigmoid of some other thing called ZR where ZR is equal to WR HT minus one plus UR times XT , so is the full gated recurrent unit a few comment here, so what is it that we have achieved by doing this first thing we have introduce many more parameters remember in the usual vanilla RNN we only had two matrices W and you , now you have six sets of matrices WG UG, WF UF and WR UR.

So you have increase your number of parameters by a factor of three, we have and we will see in ISDM that it is a little bit more when you go to LSDM it is four times rather than just three times but for RU it is three times and if we go back to simplify GRU it was two times WG UG WF UF, so the fully grated recurrent unit gives you more parameters to play with but if it gave you more parameters to play with exactly the same result that it would not be useful, what happens is this is works much better for deeper networks, it does not mean that vanishing gradients are no longer going to occur it is just that you are able to train deeper network.

So just as a simple example for certain cases if you are able to do only ten layers with a normal vanilla and then you could go up to maybe 70 80 90layers with fully greater recurrent unit that is the first thing which is you use up some extra parameters but you do get better training you get better training the second thing is you have a little bit of non-linearity which is sitting in the system which tends to mean that you can get and train richer kind of architectures using the full GRU as of now for several applications LSDM is still the first choice, LSDM is something that

we will be covering in next video it will be a grief video because most of the inputs have already been covered here.

So LSDM is very-very similar to the full gated recurrent unit like I said we will have 4 times the parameters but even though it is older it is still more or less the industry standard GRU since it has fewer parameter is easier to train but it goes somewhere and between the Wayman RNN and LST, thank you.