

Deep Learning for Visual Computing
Prof. Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 57
Recurrent Neural Networks and Long Short – Term Memory

Welcome. So, today in this lecture we are going to learn about one of these mechanisms in which you would be able to handle long sequence data. Now, as in the last lecture we had.

(Refer Slide Time: 00:23)



The slide features a header with the NPTEL logo and text: "NPTEL ONLINE CERTIFICATION COURSES" and "Indian Institute of Technology Kharagpur | Department of Electrical Engineering". The main title is "Recurrent Neural Networks and Long Short-Term Memory". Below the title, it lists "Dr. Debdoot Sheet" as the Assistant Professor, Department of Electrical Engineering, and Principal Investigator of the Kharagpur Learning, Imaging and Visualization Group at IIT Kharagpur. A QR code is located in the bottom right corner, and a URL "www.facweb.iitkgp.ernet.in/~debdoot/" is provided at the bottom.

Already understood that videos which are another major form of a visual data which we need to deal with in our day to day life, now videos are not just as simple as trying to deal with a image frames over there. And video is one of the major challenges which we would be facing down is that the whole sequence of frames which comes down over there, they are quite spatially in some way related to each other and a in a certain sequence gap over there.

And then if you are looking at the same pixel location over a period of time, then there would be certain distinct pattern which we would be able to see. Now, the question was; obviously, like is as in say for a one dimensional signal, where you can always do some sort of a moving average or a regression analysis kind of a stuff, then here also can we incorporate something of that same sort um. But, the challenge was definitely that the

total numbers of pixels which you would have at hand and multiplied by the total number of frames which would otherwise also come down over there.


Now, this volume of data is really large. And one of the mechanisms which we found out is that one simple way out may be that let us use some of these very simple networks which we had done till now. Something like your Google net, vgg net and these kind of networks; whereby you can extract out features on a frame by frame level and then these features are something which concisely describe your single frame. And then you have a set of features over a period of time and this is going to describe, what has happened down across frames in a video?

And now can we use all of these features in some sort of a temporal learning framework in order to do it. So, today what we are going to do is called as a recurrent neural network, and I am going to do one specific variant of it, which is called as long short term memory. Which is recently quite a popular one which has been gaining up its own pace, most of the common examples which we would see out over there are related to speech and a natural language.

So, if you have a text to text translation or a text debugging kind of problem. So, in fact; like when you were typing down a message over there, you would say that when you are typing an SMS on your phone even before you have finished a complete word, it start give giving you certain kind of a predictions over there. Now, these are some things which are generated by these kinds of recurrent models over there. So, based on what you have written down earlier, which may be total words and they are certain contexts over there. And based on, what part of the word some alphabets which you have put down over there that it is going to actually give you some sort of predictions.

Now, these are what come down over there? Now, what we are going to present you is more of a, can the same things be done on videos. So, can we use a similar kind of a mechanism for doing it? So, we are just going to extend out all of these understanding which we had from the field of a natural language processing on to understanding videos in terms of features as a space, for where the language is spoken for videos ok. So, without much delay, so, what I am going to do is?

(Refer Slide Time: 03:21)



NPTEL ONLINE
CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Organization

- Recurrent Neural Network (RNN)
- Long Short-Term Memory (LSTM)

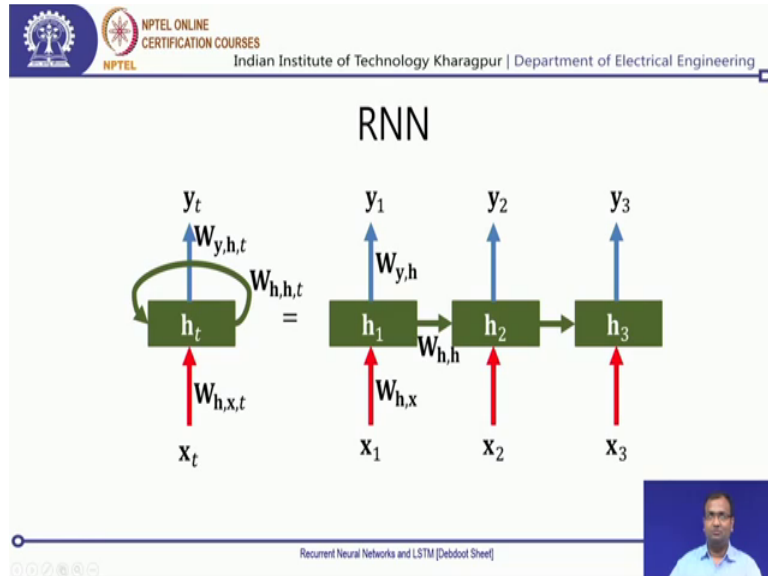
Recurrent Neural Networks and LSTM [Debbot Sheet] 2

Initially get you introduced to, what a Recurrent Neural Network is? And there is no hard and fast rule that these can be used only for videos. In fact, later down the week, I would be showing you certain examples of a more advanced ways in which RNNs are actually being used. So, they can even be used for completing an image or drawing an image, and there are multiple of these kinds of options over there.

So, we will not be restricting ourselves; however, today's lecture is just to get you the basics of it not much of an advance. So, we will be dealing with the pure basic math of, what goes down with an RNN? And one of the most popular variant is, what is called as Long Short-Term Memory, and this is credited to Schmidhuber. And, and so, we will just be going down through long short term memory in much more intricate details in terms of a signal flow graph representation.

So, nonetheless they are, still the standard neural network; however, the moment when we say that it is a sequence encapsulating network or a somewhere, where your time can also be encoded. So, this is what makes a fun point to learn down for this whole thing. Ok.

(Refer Slide Time: 04:28)



So, let us get into, what an RNN is? typically, Recurrent Neural Networks or RNN they are just a single straightforward way of representing these kind of sequence learning systems. Ok.

Now, typically how it is defined down is something of this sort that say x is my input over here and then I have a subscript t . So, if you look into this x , this is our tensor of some sort this is not a scalar value over there. So, this can be a multi dimensional thing. So, this can be a whole image. It can be just features extracted out of one single image. So, it can be a 1 d tensor, it can be a 2 d tensor, it can be a 3 d tensor and anything as you feel like, ok.

Now, that is 1 a tensor which represents only 1 9 stamp or one instance of time over there, and that is the time instant, over there. Now, associated with this one, you have a certain bit, which is called as W . Now, what you do over there is? W takes in something called as $W h$, ok, h is this hidden state over there $h t$, ok. Now, $H t$ will always be related down to $X t$ via this weight called as $W h x t$.

So, if you see over here, you would see that this weight is something which relates h , it also relates x and it also relates t . So, weight at a particular time instant also needs to change. So, that is that is another interesting aspect which you will have to keep in mind over there. So, this t is quite specific over here. Ok.

Now, from there you can produce a certain output over there, which is called as $Y t$, and the way this $Y t$ is generated is by another weight which relates down this h to y . Now,

one thing keeping in mind that since this is a sequence learning problem, so at a given instant, instance t is when, all of this mechanism is going to take place. Ok.

Now, this just does not end the whole process. Because what you have is, that you also have some sort of a feedback of this value of h going back to itself. Which means; that h at the state of h or these hidden layers over there at $t - 1$, is something which is going to influence my t -th level over there, itself. Ok.

So that is, but then it does not take itself directly over there, so the, it is not a unitary feedback mechanism. There is some sort of a feedback which goes down, so the state over here itself. So, instead of the output being related to the input, so y is not related down to x . But, what I have is, that the hidden state at the previous time instance is, what influences my hidden state and the current time instance over there? So, and the way it influences is via this weight W_{hh} .

Now, there is a simple way of representing and then looks quite nice and elegant, but there is also another version, which is called as a unrolled time unrolled representation for a Recurrent Neural Network. Now, what that would mean is; that if I have these observations x_1, x_2 and x_3 which are my set of inputs and say I have my outputs over there y_1, y_2 and y_3 , ok. Now, x_1, x_2 and x_3 can be it simple tensor, so I will take down this simple example from say writing down a particular word. So, I have written down say $h e l$ over here.

Now, what I would say is that; suppose I write down h then, which is my x_1 then, my y_1 should be e , which is; whenever I write down a particular character, let it predict what will be the next character which should come down over there? Now, I might not have written down the character over there, but it the network needs to predict itself. So, if I write down h so, x_1 will be h . Now, immediately let y_1 actually start predicting as e .

So, this is if you pull out your mobile phones over there and start typing a message you would see that, this is what will be happening down. You start if you are using an android phone or even a windows phone, most of these phones or an Apple iPhone as well they have this auto predictors and character mechanisms over there.

So, the moment you start writing down, it will start predicting on top you will get down three words or you might get down initially some alphabets over there, and then if you

select out the same alphabet it will start predicting out most closest words over there, ok. So, let this way by whole data set over there which I want to predict.

So, if h_1 is x_1 is h then, y_1 is e. Now, if x_1 is h and x_2 is e then, y_2 should be l. That is the next mandatory condition which you have. So, it is not just you will get down the value of a x_2 as e, but, you also need to have the value of x_1 as h, and x_2 as e then only you will be able to get down the value of y_2 which is l, ok.

Next is if you have x_1 as h, x_2 as e, x_3 as l then, y_3 should be l. So, these so, basically what I am trying to do is, write down the word hello over there, and I am just looking down at the first three characters and their prediction. So, these are the first three input characters and these are the, say after the first character whatever is the output character which comes down over there.

So, that is what we want to do. Now, if this network is trying to do the same thing then, t is equal to 1 is the state corresponding to this, at t equal to 2 this whole network will represent the state at x_2 , at t equal to 3 it will represent for x_3 and y_3 .

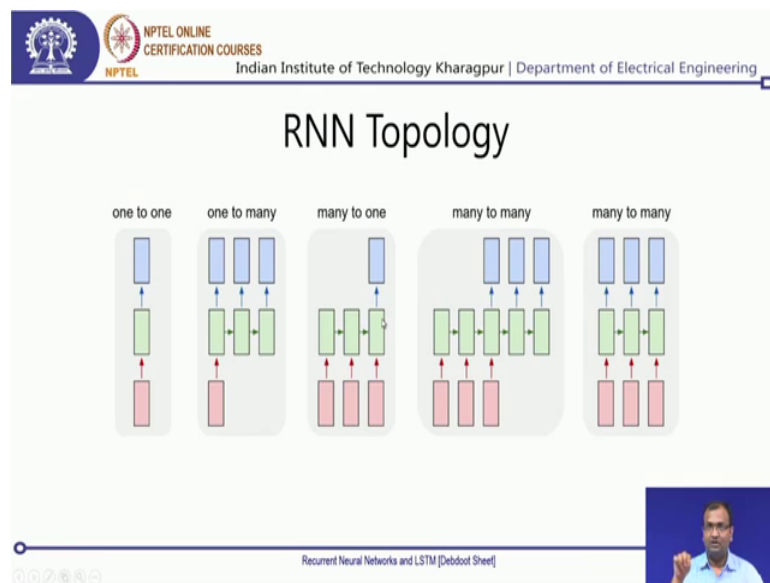
Now, this can be unrolled and written down something like this. So, I have my h over here which is my first character, which I put down ascii h via a certain weight it gets to this hidden state. Now, this is just the just a hidden layer over there. Something like our fully connected neurons, that you have your hidden layers over there, you do not exactly know, what is coming out of the hidden layer? But, it is a there is something happening. So, there are certain features which are formed over here, and then, via certain are the neuron connections it links to this next one.

Now, these weights over here can be fully connected neural networks, they can be convolutional connections this is up to you. So, we do not put a hard and fast rule over there in any way, ok. Now, what will happen is? That, this state over here, so what I said is that, if x_1 is h and x_2 is e then, y_1 is predicted y_2 is predicted as l. So, that would necessarily mean; that you have some sort of a connection between these two hidden states as well, and that is this weight of $W_{h,h}$ which is going to link it down.

So, you had this $W_{h,h}$ of t , this is that weight which links it down over there. And similarly, if you look and unroll it out, then this is what you would be getting down. Now, these weights are, what are the weights at t equal to 1? So, $W_{h,x}$ at t equal to 1,

W_{hh} at t equal to 1, W_{yh} at t equal to 1. Now, here also you will be having the weights, W_{hx} at t equal to 2, W_{hh} at t equal to 2 and W_{yh} at t equal to 2, here you will similarly have for t equal to 3, if you were unrolling it down further if t or this time duration over there or the sequence length, which we otherwise specified that the sigma length is larger, then this just keeps on continuing over there for the complete length of the sequence.

(Refer Slide Time: 11:24)



Now, let us get into some of these mechanisms in which it can predict. Now, these RNNs typically have multiple ways of being; one of them is called as a one to one prediction which is given one input over there one tensor input at a particular time stamp, it is going to predict; what the next timestamp is for? Or predict out certain variable on the other time stamping over there, then it can have one too many and this is something like the thing which you type down on your keyboard.

So, you might just type one character h and then it will say down multiple predictions over there. So, say I typed on h and then, it gives me a prediction hi , hello, happy three different word predictions over there. So, there can be a one too many prediction which comes down over there. They can also be a many to one prediction; which means that I have written down a complete word and maybe one of these alphabets is missing and then, it gives me only one prediction of one of these alphabets.

There can be many to many, which is something like; I write down a lot of these characters then or say; I write down one word it tries to predict the next word what will be coming down? So, that is a many to many. And you can also have another kind of a many many in which whatever I keep on writing on it provides me a whole sequence of description over there; so at sort of like thing over there. So, if I write down say a particular word say happy.. Then over there it might give me another word called as merry.

So, that is another many to many kind of a combination, which can take place over there. Now, these are typically the ways in which RNNS can relate. Now, if you look down into these kinds of modules you would see that, they are quite similar to certain kind of predictor filters which you have in your signal processing. So, here the like if you have already done a primary course on digital signal processing and I believe most of you have already done a course in digital signal processing before you take up these kinds of a subjects on the image processing.

So, over there you would have seen that, you have certain class of regressor filters or you have a autoregressive moving average filters, where the parameters of it changes. Then you have carbon like filters, where you are trying to predict the state variable based on, what the current input to a particular state over there is? Now, these are also kind of examples which are quite analogous to a recurrent neural network or an RNN. In fact, RNN is a larger family which encapsulate all of these smaller kind of behaviors over there based on, what kind of a topology you are following down out of these over there?

Now, nonetheless this can also be used for some of these crazy problems like; if you look into this many to many there was one of these interesting papers which will be discussing in the next class where, you put down a frame of a few sequences of videos into it, and then it tries to predict out, what will be the next few sequence of the video? Now, that is that is quite interesting because, say you are watching a movie and then suddenly, in between over there, you do not have a few frames available.

Now, can I synthesize a frame? The point is that, we have already finished our generative models and what; that means, is that through our a generative models an adversarial learning for generative models we have already known that, given you know, what is some sort of a distribution? And if you are able to model down your network in a way

that you can draw down a random sample from a distribution, you will be able to generate a whole image. And that is pretty much feasible and intractable.

Now, over there when we were doing it down, why cannot we do it in the same way? So, say that these hidden variables had some sort of a temporal pattern over there. Now, I do not know exactly what is the temporal pattern? Now, I keep on giving these a few sequence of input images over there, then it learns down the temporal patterns across the next few sequences which come up and then, you can actually synthesize. And that is what comes down in my RNN topology actually out of this many to many kind of a mapping.

So, there can be more like real interesting ones. In fact, when doing trying to do a video analytics problem, which will solve it in the next class over there, you can have a mechanism which can be something like this. Hold. Hello [FL]. Ok. Resume. Ok. So, over here since you have this kind of mechanism, so you can actually look into even synthesizing out your newer videos, which can come up really easily. Ok.

(Refer Slide Time: 16:34)

The slide is titled "Looking back at the Math" and features a diagram of a recurrent neural network cell. The diagram shows an input x_t (red arrow) entering a green box labeled h_t from the bottom. A red arrow labeled $W_{h,x,t}$ points from x_t to h_t . A blue arrow labeled y_t exits the box from the top, with a blue arrow labeled $W_{y,h,t}$ pointing from h_t to y_t . A green curved arrow labeled $W_{h,h,t}$ loops back from the top of the box to its bottom, representing a recurrent connection. To the right of the diagram are three equations:

$$h_t = f_{NL}(W_{h,x,t} \cdot x_t + W_{h,h,t} \cdot h_{t-1})$$
$$y_t = f_{NL}(W_{y,h,t} \cdot h_t)$$
$$\{W_{y,h}, W_{h,h}, W_{h,x}\}^* = \operatorname{argmin}(J(W))$$

The slide includes the NPTEL logo and text: "NPTEL ONLINE CERTIFICATION COURSES Indian Institute of Technology Kharagpur | Department of Electrical Engineering". At the bottom, it says "Recurrent Neural Networks and LSTM [Debbot Sheet]" and shows a small video feed of a man.

So, let us get into understanding the math behind all of this process. Now, the math is pretty simple laid forward and, what I have done over here is that, I am trying to use down the notations reuse the notations from over fully connected network definitions over there. So, if h is some sort of a hidden variable which you have over here, and x is the input which goes down over there, then via some sort of a network transformations.

So, if you have just if you are looking down at one single hidden layer over the one so, this h over there is just one single hidden layer.

So, you are just going to have these kinds of weighted connections from input output. So, that is something, which has this kind of a form. So, you have your input X_t which has a dot product with the weights over there, W of $h \times t$. And then, you sum up the other coefficient over there, which is; what is the self state feedback over here for the hidden variables.

So, that is the hidden variable at $t - 1$, which has a dot product with this W of $h \times t$ over here which is the hidden to hidden transformation state at the t th dimension. And then, for your output over there your y_t is something which depends only on h_t , it does not have any other dependency over here.

So,. So, the output is pretty late straight forward. And now, whatever you are going to learn down in case of your dependencies on the previous hidden state variable over there, then that is just being defined in terms of the equation which is the first equation over here. So, h_t is something which depends on h_{t-1} to a certain extent. So, since h_t is dependent on $t - 1$ h_{t-1} , which has and then, h_{t-1} is dependent on h_{t-2} , h_{t-2} is dependent on h_{t-3} .

So, in essence your h_t is dependent on that complete infinity series. So, since the origin of the problem, till the current state, it is it is something which is dependent over there. Ok. Now, what you would realize is that, that t equal to 0 is when you are starting. So, h_t equal to 0 is when I am starting. So, I definitely do not have h_{-1} available to me now.

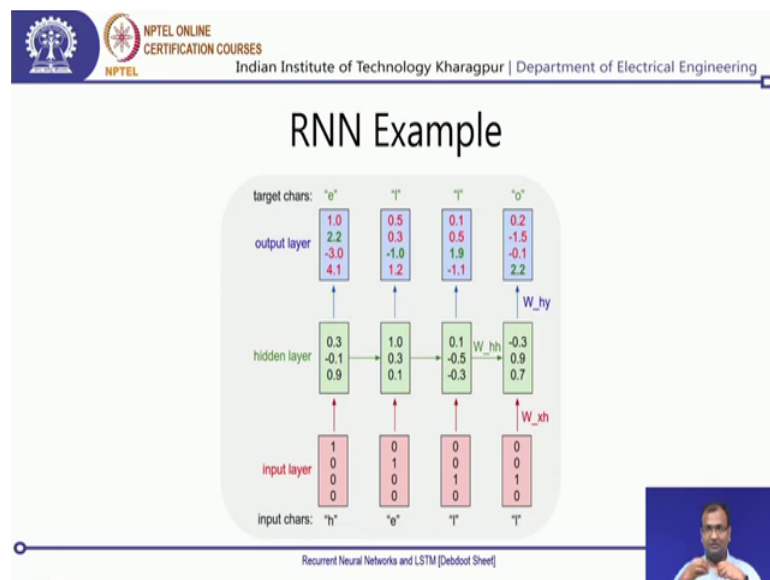
So, for all general purposes, what we do is? The moment where we start is where h_t equal to 0. And, we do not assume that, there is something which translates from h_{t-1} over there. So, there is no h_{t-1} , we are just starting at h_t equal to 0 and then we keep on continuing over there.

So, Y_t is something which is related down to h_t via this non-linearity over there. And then finally, what you have is during this learning process, what I am trying to do is? I will formulate a cost function. Now, this cost function can be as simple as Y_t and how good is the prediction over Y_t .

So, this can be a classification problem. Y_t can be a classification problem. Y_t can be a regression problem as well. And, and based on whatever cost functions I take over there, which are the same kind of a cost function, which we had used in the earlier cases for a classifications on our frame by frame basis.

So, if you had a classification cost classification kind of a task, you were using something like a binary cross entropy or a negative log likelihood criteria. In case; you had a regression problem you could pretty much use a mean square error or k l divergence or a j s divergence over there. So, we can use any of these in order to solve our weight optimization problem over here as well.

(Refer Slide Time: 19:34)



Now, if you get back to our RNN example over there. So, I had this example which i was picking up, which is of a trying to write down the word hello. Now, h is what comes down at t equal to 0 or the first timestamp over there.

Now, based on it, what it is trying to do is? This is the input, so if each of these neurons are over here something which represents just my ascii character over there, so and say the first one is corresponding to h. So, that is what goes in over there.

Then I have my hidden layer over there. So, there are just three hidden states just the hidden layer tensor size is 3 cross 1. And then, from there I generate an output state over there and this is again a 4 cross 1. So, now, in hello, what you have is, you have an h, you

have an e, there are two l, so that is just one unique character over there. And then, there is a o.

So, basically you have four unique characters which come down, which will have to timestamp over here. And that is the problem which it is doing. So, the your input over here is a food is a 4 cross 1 tensor because, there are just four unique characters and the based on whichever is firing up, it is that particular character which is going to get generated.

Now, when I write down just h, it is supposed to generate me e, if I write down h and e it is supposed to generate me l. So, you can see this, state transitions which are going down over here. Now, if I write down h e l it is supposed to generate me another l, if I write down h e l l it is supposed to generate me o.

So, this is the typical math, which gets solved out in this kind of a environment over here and then, based on whatever is your output which comes down is, what you are going to write down over here.. Now, that is laid out straightforward as to, how, how it is going to do! Now, you what, you would see down is that not always you are going to get down a correct prediction. Now, if you look down over here. So, I have my output layers and then, I have my target actual targets coming down.

Now, the first character over here is h, the second one is e, third is l, fourth is o, which it should come down. So, over here the prediction, maximum prediction is what comes down at 4.1, but that is o; however, my 2 is something which is at e. So, it means that; this network is not yet trained. So, we need to train it further. So, there will be an error which is generated.

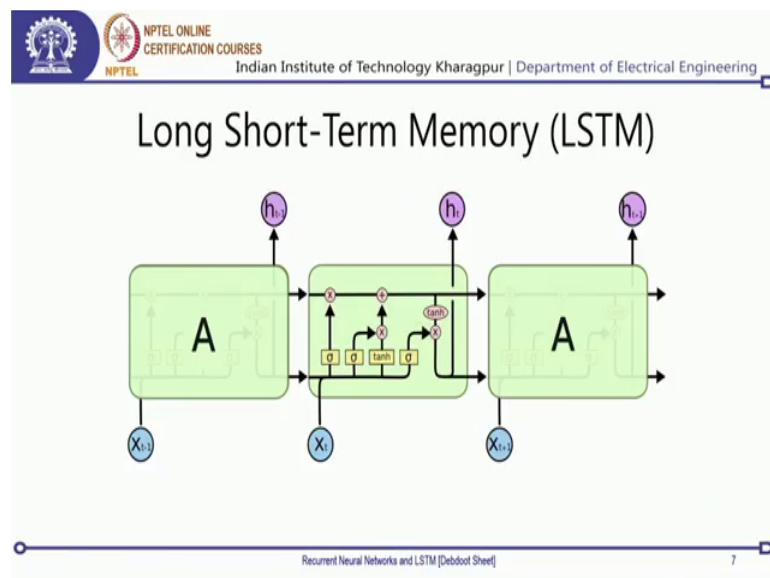
In the next which is h and e, I get down a predi it is supposed to get down a prediction of l; however, what i get down is, the maximum is at o. But, this is where the prediction should come down, and this is one of the least minus 1.

Now, when I write down three characters h e l, ok, then let us look down, what comes down? Now, the moment I have three characters written down, it is actually giving me a correct prediction over there of 1.9, which is at l. Then, if I write down four characters it gives me a correct prediction over here for the fifth character over there, which is o.

So, one thing which you can realize is that, if you have a sequence learning kind of a problem as an over here, then the larger is the sequence length, the better and better the prediction keeps on going down over there otherwise, it there can be chances of errors with a lower sequence length.

So, these are examples, which will be revising down in the next lecture. So, in the next lecture, when I will be actually going through one of these case studies before we start down doing our lab, so in one of the examples, where we were solving out there was a mechanism in which we could find out exactly, what should be an ideal sequence length to be used over there? So, that is with the RNN part over there.

(Refer Slide Time: 22:45)



Now, the next which comes down is that the kind of a Recurrent Neural Network which we had used is what is called as a LSTM or a Long Short-Term Memory.

Now, a Long Short-Term Memory has different kind of a mechanism in the intermediate layer and, how it connects down. So, your x 's which connects down to h . So, your h can again connect down to directly y over there via its own transformation function or even h can itself be the y . So, that is that is also possible over there.

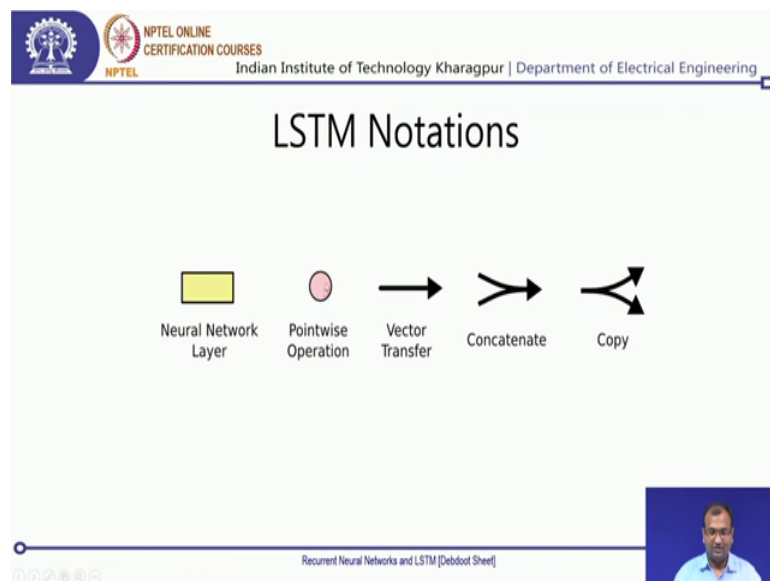
However, what we are trying to look over here is that, what is the inside connections over there? Because, in the earlier case what you saw is that, x is related to h via certain weights and, a h is related to itself from its previous state. So, h is something which is

depend h of t is dependent on x of t as well as, h of t minus 1. This was the thing. But, then the dependency in which h is dependent on h of t minus 1, was pretty straightforward over there.

Now, we do not have that kind of a straightforward mechanism in case of Long Short-Term Memory. And, one of the reasons is that typically if you are looking at long order relationships then, certain short order relationships actually do not come into matter and they do not play any role over there. So, these are things which we will need to really modify and do. So, what comes down over here is? That you have multiple number of gates. So, some of these gates allow you to pass down a certain information, some of these gates will not allow you to pass down certain information and then, this is how the whole mechanism comes down.

So, that is about the long short term memory which we will be doing now.

(Refer Slide Time: 24:13)



So, let us get into some of these notations which we have. Now, you will get you need to get used to certain of these new notations over there. So, whenever there is a yellow block, so that typically denotes some sort of a neural network layer coming down. If you have a pink operator over here or pink circle, so that is a point wise operations.

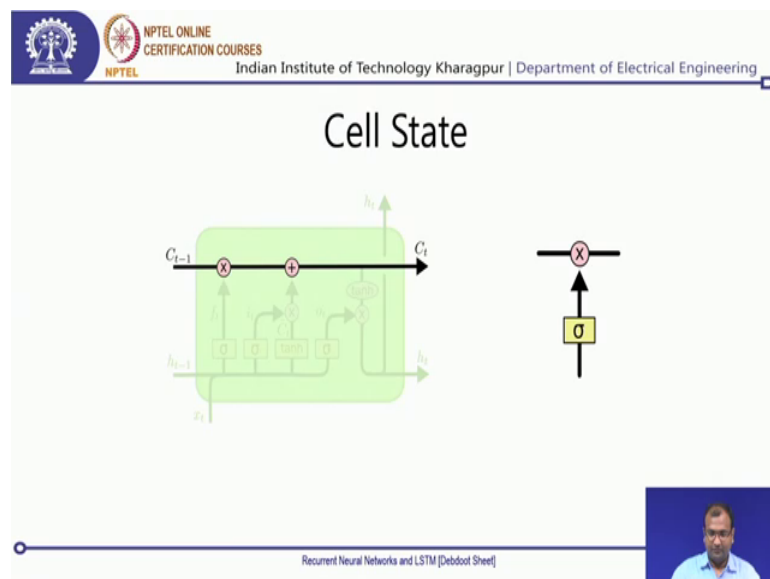
So, let us get back over there. So, these kinds of operations over here are some sort of point wise operations which will happen, which is each element on the tensor is going through one of these operations.

Now, whenever you have these yellow blocks so, there are basically some sort of neural connections available between them, ok. Now, if there is this sort of a straightforward array then, that is something called as a vector transfer or a tensor transfer. So, the same tensor is just flowing down through that one, there is no kind of an weighted a dot product or anything which happens. Now, you also have a concatenate operation, now this looks a bit different you have already studied down dense nets, where you have seen down how concatenate operators actually operate down.

So, you have the third channel over there or a multiple channels you basically specify one of these concatenation directions along with it will just concatenate. Now, here the concatenate decision direction is not the time direction, but concatenate along the feature length over there is what will happen. Then, if you have a branching out arrow over there, so that basically means that you are copying down through to two different paths, the same tensor is copied down to do different parts over there.

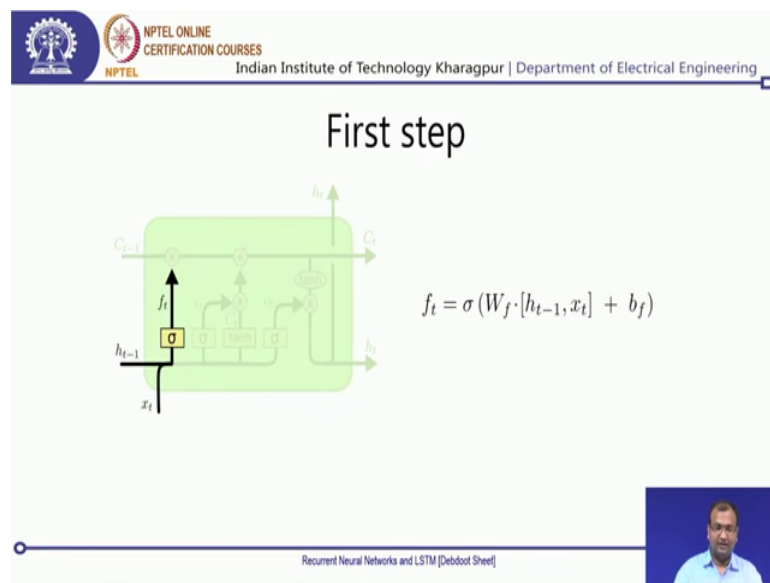
So, this is what your LSTM is actually made out of if you look back again.

(Refer Slide Time: 25:40)



So, now let us get into what comes down over there. Now, one of these things is what is called as a Cell State line. Now, Cell State line is something which will be indicative of, what is the current state? And, whether it is trying to preserve itself or modify itself. And, if you go through it, then you would find out that the cell state line is something which is quite unique and this will be indicating, what is the long order relationship as goes down within the whole network over there.

(Refer Slide Time: 26:10)



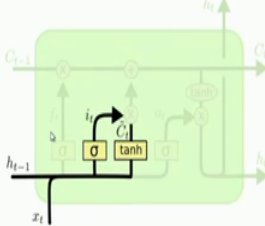
The next is when let us start with, what happens over there in the first case? Ok. So, I have my input, which is my x , I have my state from the previous one. So, my hidden state from the previous timestamp which is coming down of h of t minus 1. Now, if t is equal to 0, then I do not have an h of minus 1 available over there, so that is a completely zero. But, if I am there in one of these descents step points over there at x of t , then I have my hidden input coming down over there.

Now, what I do is that? I have a dot product operation and a sigmoid non-linearity which happens so, this sigma over here represents. And then I generate an output which is called as a F_t .

(Refer Slide Time: 26:48)

NPTEL ONLINE
CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Second step



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Recurrent Neural Networks and LSTM (Debbot Sheet)

Now, in the second step, what I do is that, I take my input over here and my hidden state over there and then I generate a dot the I generate two different terms basically. So, I generate something which is called as a I t and then, I generate something which is called as C hat t. Now, this C hat t and I t they are quite interestingly interesting points over there because, they will help me in actually figuring out, what will be the mechanism of whether this is a important information or this is a non important information.

Now, if you look down at these two transformations over there. So, I t is something which has a sigmoid non-linearity over there; which means that, it is going to pull down the information between 0 to 1, ok. Tan hyperbolic on the other end is something which is going to pull down the information from minus 1 to plus 1.

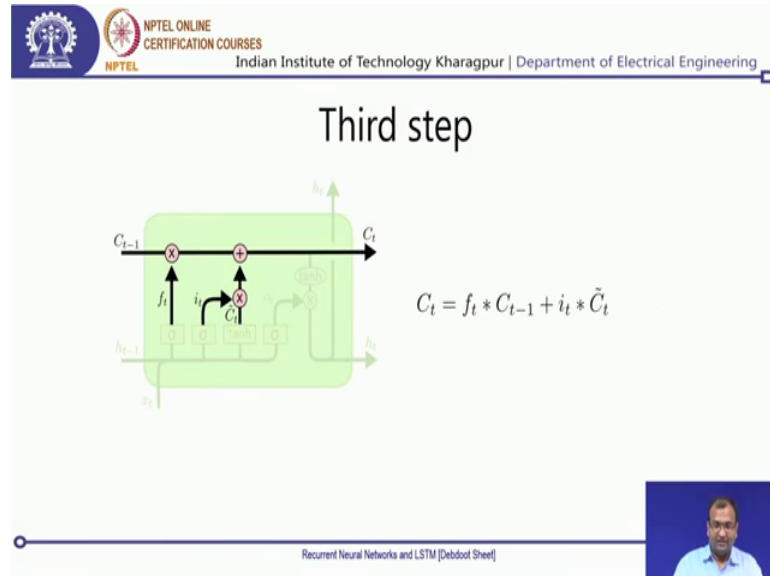
Now, since they have two different weights. So, there is W i weight which is associated with I, W c is a weight which is associated with C. Now, together you can actually get down three different states of value between minus 1 to plus 1.

So, while this is the sigmoid part is just going to accentuate and pull out your information over there; whether it is a 0 information or it is a non 0 information. On the other case, you have this c t which is going to change it twists the direction of the information.

And now, if you have this things together taken down and you do a dot product which you can see over here, and that dot product is something which is added down completely then you would figure out that, together this can actually modulate by

actually adding down and then, increasing the intensity of a certain tensor, or it can reduce down the intensity of a tensor as well.

(Refer Slide Time: 28:25)



So, let us get into the third step what comes down. So, far and the third step you would see, that you have this F_t coming down over there which we had done in the first step, and in the second step, I have this I_t and C_t . Now, I_t and C_t is something which comes down through a convolution operation over there, and then, or in fact; like over here it is basically a circular and dot product which is something similar to a circular convolution which happens.

Now, together what it would do is, either F_t can be multiplied by this C_t and C_t is the cell state from the previous state over there. Now, \hat{C}_t is the cell state which is generated, which is a predicted cell state for the current one, now that will be added down over here.

Now, this F_t is something which is going to add down a direct form of or keep on recursively cumulating some of the input and the previous hidden states over there. I_t and C_t on the other side, is going to add this information over here and, what that would lead is? That, if I need to forget certain variable then, it will just minus or subtract it out. Because, your \hat{C}_t has a tan hyperbolic non-linearity that can induce a negative over here, so if you have some positives coming down and a negative induced over here. So, it just forgets all of this. So, this acts as my forgetting input over there.

(Refer Slide Time: 29:36)

NPTEL ONLINE
CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Fourth step

$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Recurrent Neural Networks and LSTM [Debdoot Sheel]

Next. So, that was about like; if I am doing a sequence model, I need to remember certain things and I need to forget certain things. So, what should I be forgetting and what should I remember is what you get done? Then you have your fourth stage over here, which is just going to do a direct transfer of all of these information together and then, do a tan hyperbolic over that.

Now, finally, this is going to translate into sending my, hidden layer output over there as output from this one and, also taken my current cell state. So, taking my current state cell state and whatever input was coming down, which is my previous hidden state and my current state. So the, if you look back into a standard RNN, so you just have one single pipeline which is passing down between networks and that is just the hidden state over there.

But, here we are going to pass down two different factors; one is called as a cell state another is a hidden state. Now, hidden state you can put down in some way as the short term operation which it needs to remember. Hidden state is the long order relationships which it needs to remember and that is that comes down the name of a long short term memory. So, short term is what comes down through this H s. Long term is what comes down through this a cell state memory which, which keeps on passing down over there.


(Refer Slide Time: 30:46)

NPTEL ONLINE
CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Take Home Messages

- Ian Goodfellow, Aaron Courville, Yoshua Bengio, *Deep Learning*, MIT Press, 2016.
- Hochreiter, S., and Schmidhuber, J., "Long short-term memory", *Neural computation*, 9(8), 1735-1780, 1997.

Recurrent Neural Networks and LSTM [Debbot Sheet]



So, this is what comes down over there finally, in case of a LSTM over there.

Now, this was just a basic theory and if you want to read more about it then definitely follow down the book and you have the original paper by Schmidhuber and Hochreiter. So, you can go down through these papers. Now, in tomorrow's lecture, what I am going to do is? How do you use make use of all of these information together in order to design a video classification or a video activity direction framework is where, we get into more engineering details of where LSTM's are supposed to be plugged into our model for its complete use. So, till then stay tuned.

Thanks.