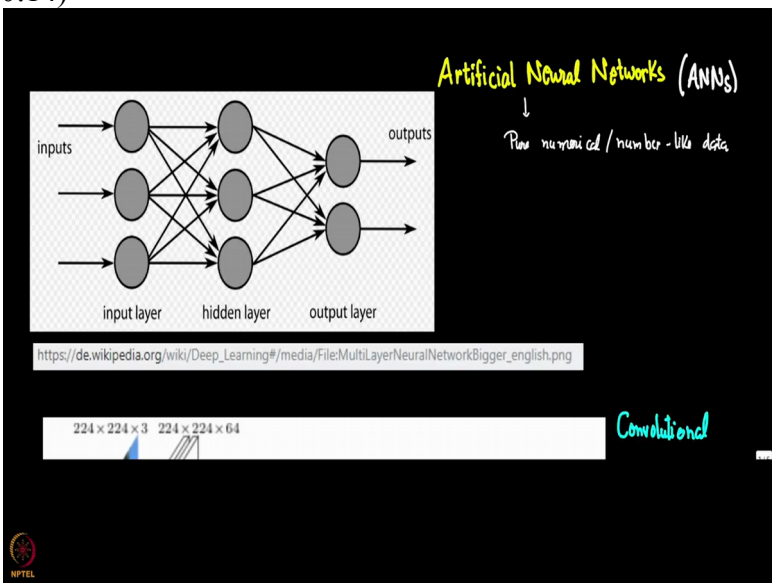
Machine Learning for Engineering and Science Applications
Dr. Balaji Srinivasan
Department of Engineering Design
Indian Institute of Technology Madras
Introduction to RNNs

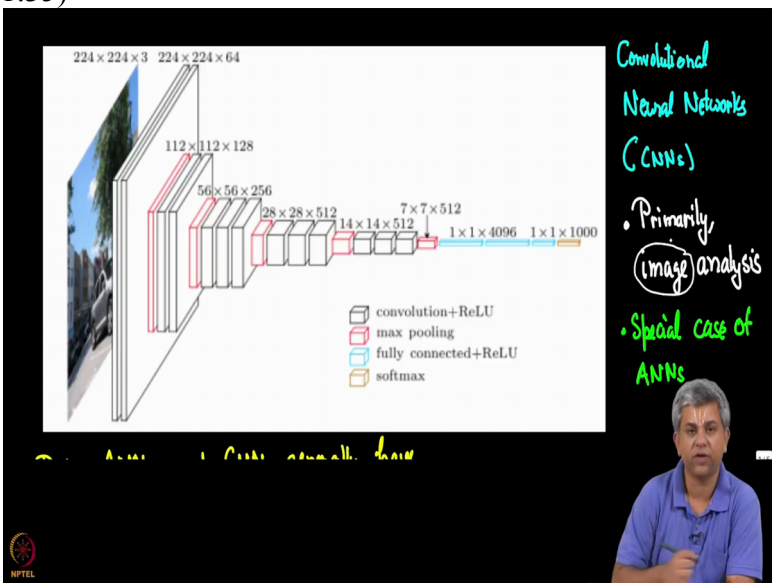
(Refer Slide Time 0:14)



Welcome back. This is the final module from this video on for the next few videos, this is the final module of the deep learning series that we have been doing so far. Next, we will do some conventional machine learning techniques that were you know non-deep learning techniques in the next week. So this final module is for what is called recurrent neural networks. So far, you have seen a couple of main techniques for deep loaning. One is for artificial neural networks, very heuristically speaking, this is good whenever we are dealing with your numerical or number like data.

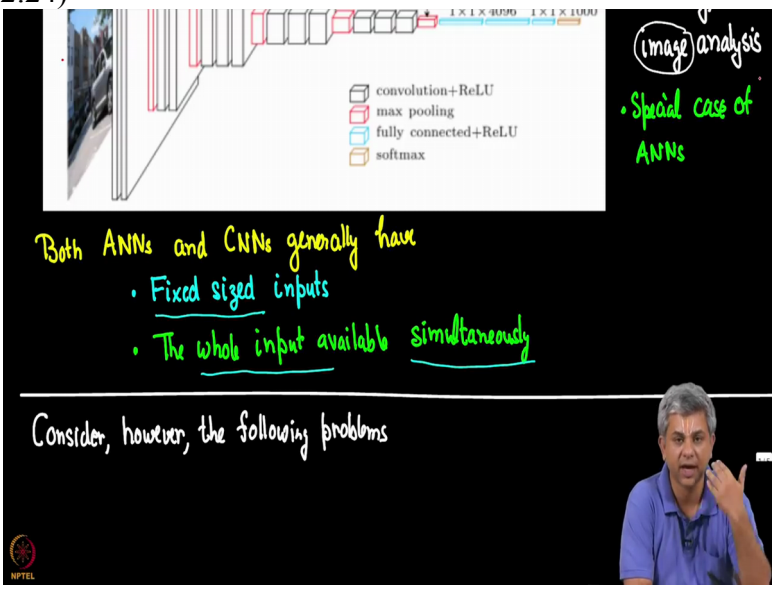
This is in the context of again I am saying this in the context simply of engineering or science problems. For example if you want, you know you have temperature and pressure somewhere and you want the density for a specific application, maybe you could, if you think that these are the only 2 variables in play, you could use artificial neural networks to predict something of that sort. Once again, in week 10 or so, we will actually see a whole variety of applications where this distinction will become clear. So that is for artificial neural networks.

(Refer Slide Time 1:35)



And you also saw this of course is Alex net we also saw convolutional neural network, CNNs. These are primarily for image like data, okay. So when you have data in the form of pictures, then we use CNNs. Once again, we will see applications of this. Dr Venapathy has shown you if you applications already in medical image analysis, we will show you a few more in proper engineering problems or science problems in a couple of weeks from now. So we also saw that CNNs are essentially a special case of artificial neural networks. Now, you have thought class of data or 3rd class of problems which is what RNNs deal best with okay.

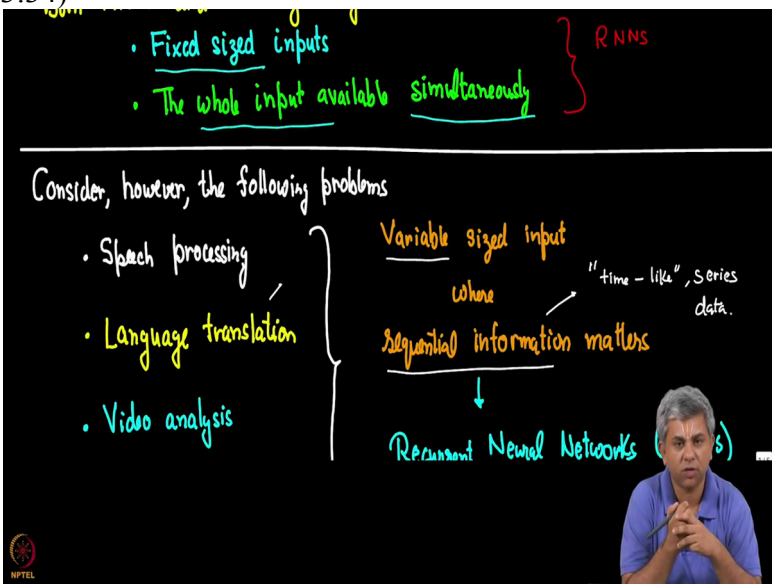
(Refer Slide Time 2:24)



Now ANNs and CNNs have a couple of lacuna, they have a few problems. The problem is that your inputs are of fixed size okay. So for example if I use temperature, pressure, that is what I am forced to use each time. Now there is a subtle point here in what fixed size means. We will come to that when we come to RNNs, both in this video and in the next okay. No more importantly you need that the whole input be available simultaneously. So you have to give the input for example the image here has to be given one shot.

You cannot later decide to give another image. So that image is given and you get an output. Okay. Now compare that with something like let us say making closed cautioning for what I am speaking. Okay. So you do not have typically and suppose you have online translator, that is as I am speaking, somebody is writing down what I am speaking or Alex I interprets your inputs as you speak okay. Or your car does think as you speak.

(Refer Slide Time 3:34)



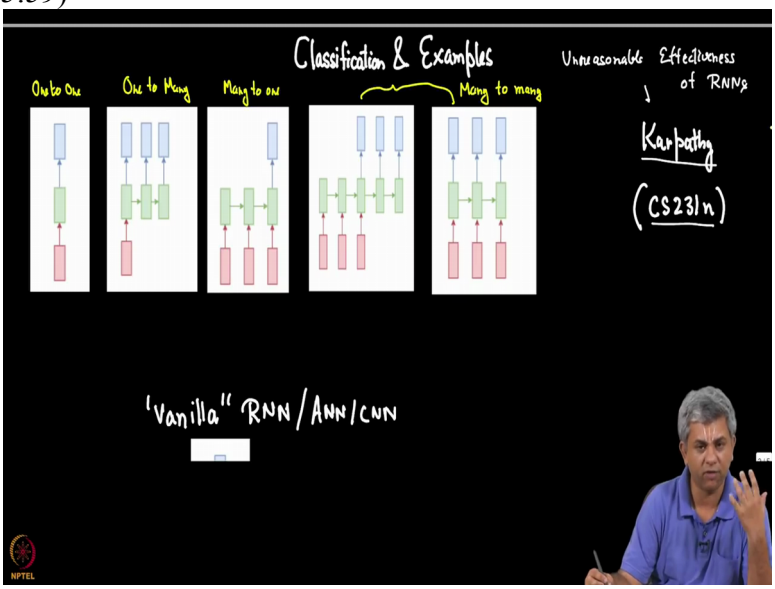
So something of that sort requires sequential processing is also some of the typical problems that are reduced in RNNs, RNNs R recurrent neural networks, involve typically speech processing. Okay. So like I said Amazon Echo, etc will be using an RNN somewhere within there. So speech recognition will be there. Language translation, for example Google translate, so that uses some form of RNNs, okay. Video analysis, etc, all these things where sequence matters, that kind of problem typically involves recurrent neural networks.

So the key ideas which are important in deciding whether RNN is an appropriate model to use or not is a variable sized input, okay and sequential information. Even if it is not quite clear what variable choice input is, it will become clear by the end of this video and the next video. But sequential information is something that is quite important. Now sequential information typically means in engineering speak, time like or time series like data. Now of course, RNNs are also used and currently the most popular use of RNNs is in language. Okay.

Or in what is called natural language processing. This involves series of words, like I said translating from one language to the other, Google has Google translate, all those applications use RNNs because words come and they are in a sequence also, sentences are not necessarily of a fixed size. Okay. You can have a short sentence, you can have a long sentence and you know that kind of you have to translate sentence between sentence.

Now these, we will not be discussing really language applications, that is not the purpose of this course. We are only going to basically discuss overall ideas behind RNN, behind RNN some of the architectures just like in CNN, you saw a few architectures, we will be discussing a few architectures within RNNs also and we will also show you you know what kind of tweaks you require for putting it into engineering problems, specifically actually in week 10 but I will show you one example this week also.

(Refer Slide Time 5:59)

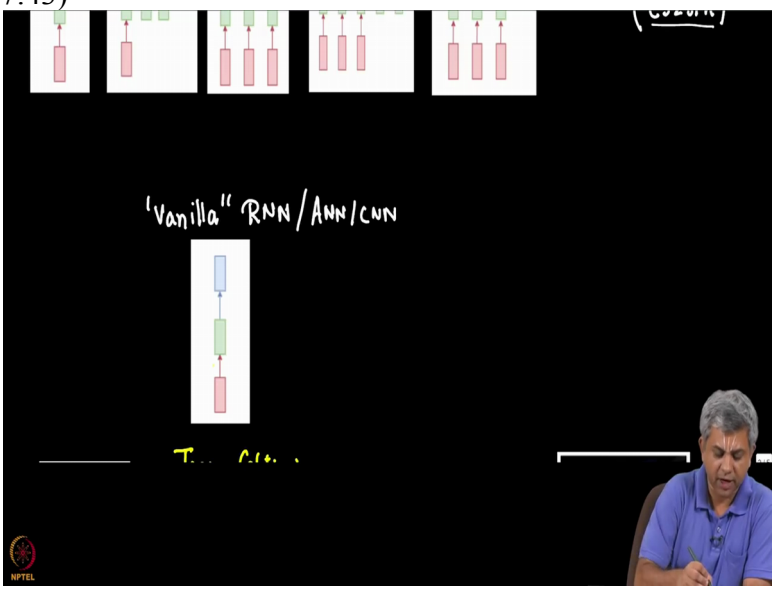


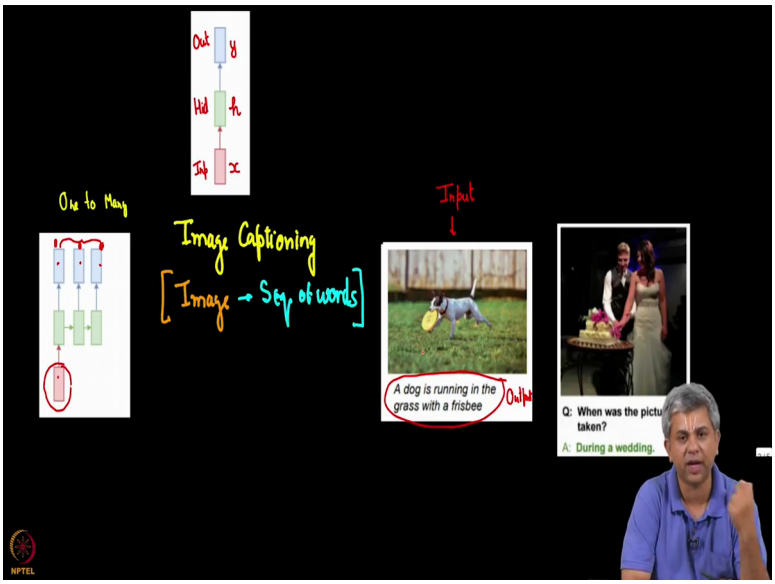
Andre Karpathy has written a very nice block, also there is a course called CS231n just called conventional neural networks but it does discuss basics of RNNs as well as CNNs. So this idea is actually borrowed from Andre Karpathy's. A very nice blog post which we will post on our website also. So this is called the unreasonable effectiveness of RNNs. So I would recommend that all of you take a look at it, it has some very nice language examples, we will not be using too many language examples at least within this course.

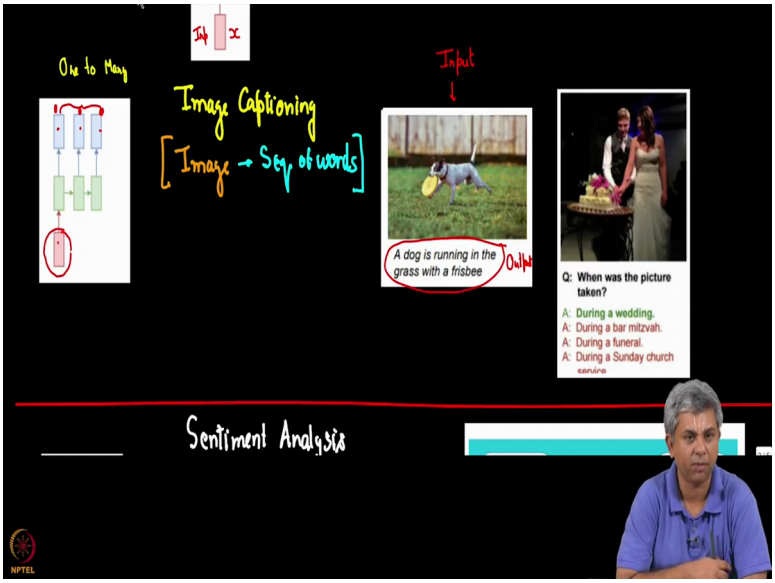
So but if you are interested, you can take a look at that. He also has posted his own Python code on github there. Okay. So this is not very standard classification but Karpathy has classified the types of RNN architectures that you will see into 5, so this is called one to one, this is called one to many, this is many to one and these 2 are many to many. Okay. Once again, this is not really classification that is there in the literature but I think it is a great sort of way of classifying it for introductory purposes.

So we will look at that. So let us look at some examples of where RNNs are used. Once again, very notionally and we will go into depth a little bit later in the next few videos, okay.

(Refer Slide Time 7:43)







So here, this red block here simply is our usual input, let us call it X, this green block here in the middle is our hidden layer and the blue block here is our output layer, okay. So as usual, input hidden, output. Now this one-to-one classification is basically our usual ANN, you can even think of it as a CNN in case there is only one or many, you can even think of this H as as if it is many layers but let us assume it is one. So it is not a particularly good CNN. But this is, let us assume this is simply a ANN with one single hidden layer and you have an output layer.

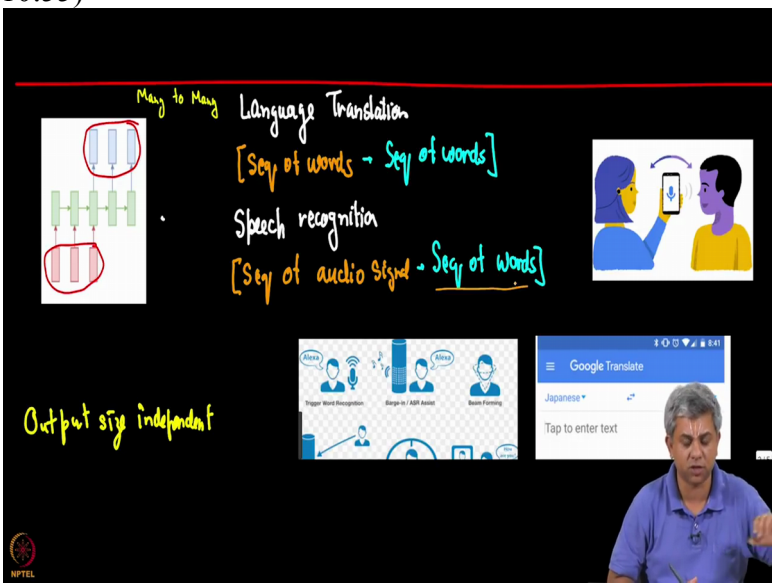
So whether it is RNN, ANN or CNN, as long as there is only one, you basically can say that all 3 are equivalent. So this would be what would be called a vanilla RNN or a simple RNN structure, okay. So up until so far, there is nothing impressive about nrl. So let me show some results, I must point out that I got these results of the web a little while back. I have kind of forgotten where I got them from. So if somebody can put it out, I will actually put up an explicit acknowledgement on the website. So please let me know.

So this one is a one to many classification and this is where RNNs kind of start differing from ANNs and CNNs. So what is happening here? You have a single input and you have got a bunch of outputs. Now what is that like? This is as if you have a single image which is the input to your problem and what the output here is a bunch of words. Now how does this magic happen? Again, we will see a little bit of it in the next few videos but the basic idea is very simple, as usual, it is just a mapping.

You somehow have to map one input here to a bunch of outputs. Now what each of these outputs means, we will see shortly. Okay. But let us assume this corresponds to word1, word2, word3 okay. Now you can see that just by seeing the image, there has been a fantastic output here saying a dog is running in the grass with a Frisbee. Okay. So that is remarkable. Of course, it is not simply a one to many RNN as I have shown here. There are many other pics going on and that is well beyond the content of this course to go into every detail.

But hopefully at the end of this course, you will be able to read papers like this, image captioning papers and be able to figure out what is actually going on, okay. And similarly, there is a question asked based on a figure and you are supposed to figure out you know which of these choices is correct and the fact that this picture was taken during a wedding. As you can probably figure out by what we have talked about so far, a lot of it depends on how you train. So training is a very very important part of how this kind of output can be gained. But a simple, this was just shown just to show you a simple example that there could be one input and many many outputs, okay.

(Refer Slide Time 10:55)



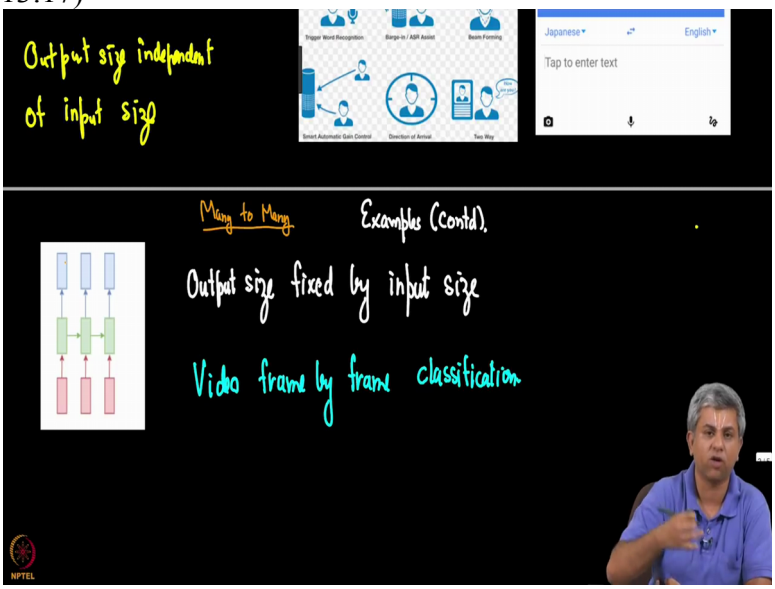
Then you could have a bunch of inputs, the opposite case which is many to one. You have whole bunch of inputs and one single output, what would that be like? That is like giving a sequence of words and trying to find out, this is called sentiment analysis, an example of this is just time to find out whether this is positive or negative. So if you read somebody's feedback report and somebody has written it in words and you simply want to figure out, is this a positive feedback or is this a negative feedback .

Some students at IIT Madras actually did a nice work of analysing Twitter feeds of many companies' stocks and trying to figure out whether the sentiment market, sentiment for this was positive or negative, this can be done automatically. Once again, if you use an RNN structure for this, it would be a many to one structure, a whole bunch of words with one output, is this positive or negative? You can do several things with this. The 3rd kind of past would be many to many and this itself splits into 2.

A many to many is such that you have a whole bunch of inputs, you have lots of outputs but the outputs need not come simultaneously with the inputs, further the outputs need not have the same size as the input, okay. For example if I change, if I translate one language to the other, suppose whatever I am speaking is changed into Tamil, the number of words in my English speech need not be the same as the number of words in my Tamil speech okay. So similarly, you can see Google translate also works, it is not as if it is a one-to-one map.

It is just that some bunch of words are given and then after that some other bunch of words comes out as the output and that would be a language translation task. Similarly, speech recognition. Why would that be a many to many task? One of course, when I speak, there is this length of my audio signal. The length of my audio signal lead not be the same as the length of the number of words that I am using okay. So this is also a many to many task where the input size is not the same as the output size, need not be the same, okay not generally be the same as the output size.

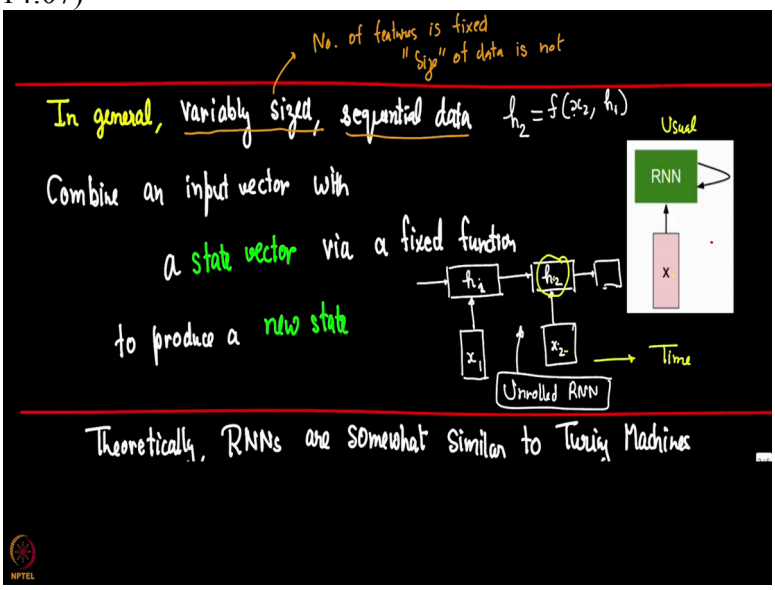
(Refer Slide Time 13:17)



Now there is a final many to many task and in fact we will show you one example of this in the very next video. , the many to many task here is where input size is the same as the output size, okay. So for example, a frame by frame classification of a video and I will show you another example which would be trying to figure out what a person is doing based on sensor data on that person and you want to know what that person is doing every second.

So whatever input you have, you have a corresponding output. At least you have equal number of inputs and outputs in such cases. So all these are I would say traditional examples of what RNNs are used for, okay.

(Refer Slide Time 14:07)



Our interest of course is to try and find out engineering examples but before that, let me just show you the rough structure of an RNN, I just showed it as a box. So let us get into slightly more detail within this video about what is done. So the general idea is variably sized, sequential data. What does variably sized mean? Okay. So variably sized means that even though number of features is fixed, the size of data is not.

So this might be slightly confusing to you, it is best explained in terms of a couple of examples. So we will see that later on, just keep this at the back of your mind. Okay. What does variable size mean? And we will see this both in this video as well as in the next video. Now what is the basic idea of a RNN? Same as most ANNs which is you combine an input vector, okay. You get an input vector and you get an output vector except in this case you have a slightly different architecture.

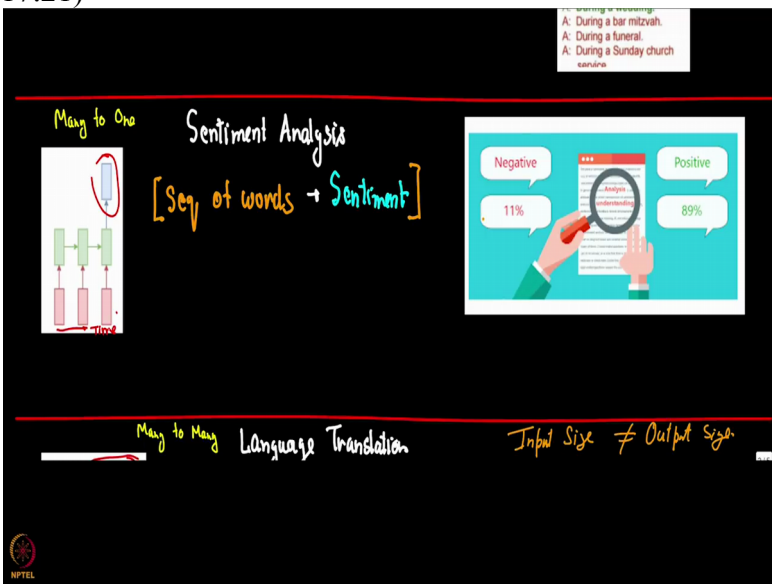
It works like this. You will see this in the pictures that I have drawn so far. You have one input vector here, you have an output vector, a hidden vector here and we will have another hidden vector here, so on and so forth. Now this hidden vector, let us say if I take H2 is a function of 2 things. It is a function of X2, it is also a function of H1 okay. And you will see the next video how this can be tremendously useful. In fact later on in this video itself, I will talk about this.

So this is a function of 2 variables, X2 as well as H1. So usually this, many people find this very confusing. This is a usual representation. X, RNN and it loops into itself, you will understand

why this is so, very very shortly okay. But most of us including Professor Andre do not prefer this kind of picture, we prefer this picture which is called an unrolled RNN. So I would recommend that at least for thinking purposes, you think of (())(16:49) in this way but if somebody uses this, you should be able to understand it.

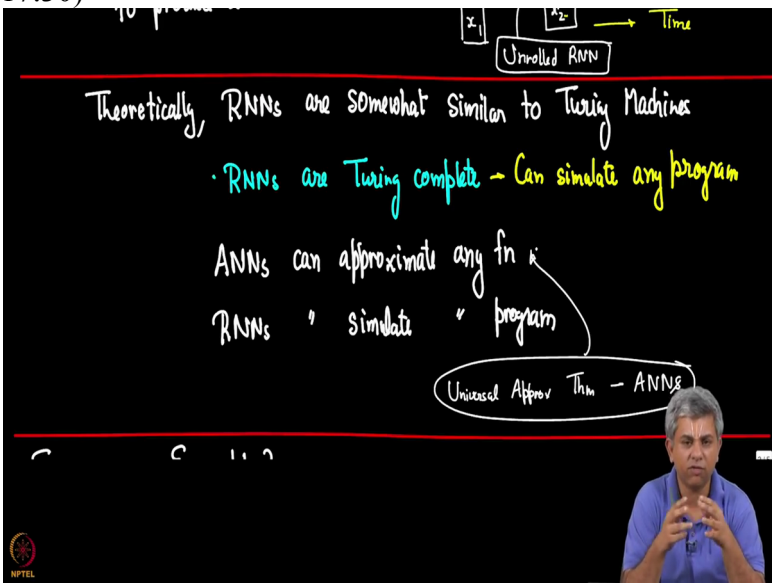
So I will show this a little bit more in this video. Okay. So remember this. You have one hidden vector, it depends on the previous hidden vector as well as the input vector at that particular time. This axis, even though sometimes I will abuse notation and call this number of layers, it is actually it is a stand in for time.

(Refer Slide Time 17:21)



So as I keep on moving here, in all these pictures much you can think about it as if you are moving in time okay. okay. okay.

(Refer Slide Time 17:30)

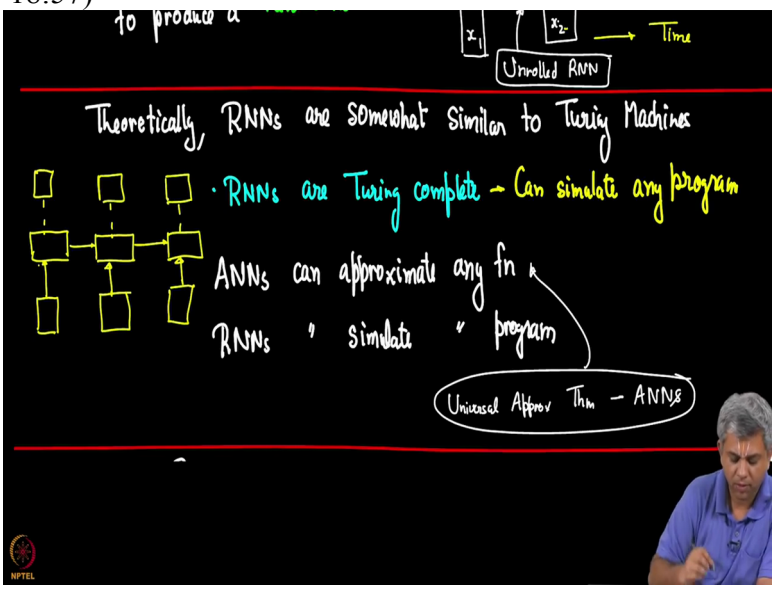


So here is some theoretical detail. I will discuss this and then I will come back to how this function works, okay. So these are if somebody is from a computer science background, theoretically remember that we had universal approximation theorem for artificial neural networks. That says that I can take a large enough ANN which will approximate any function that I would like okay. Now RNNs have a similar property, in that they can simulate or they can come close to approximating any program. Okay.

So any computer program can be sort of simulated because remember, for a computer program all your inputs might not be available simultaneously, okay. So you might say something now or you might give some data now and after some time, the computer asks something else and you give new data at each point, okay. So RNNs are supposed to simulate a generic function which can move and it can take inputs at varying time instants. Okay.

So that is the speciality of RNNs. So once again theoretically we know that RNNs are what are called Turing complete, that is they can simulate any program okay. After this brief not, just sort of a theoretical note, we will move on to a practical thing. Okay.

(Refer Slide Time: 18:57)



So let us come here. Let us come to the structure of an RNN. Let us take a simple example and explain it via diagram. Okay so remember the structure. Okay.

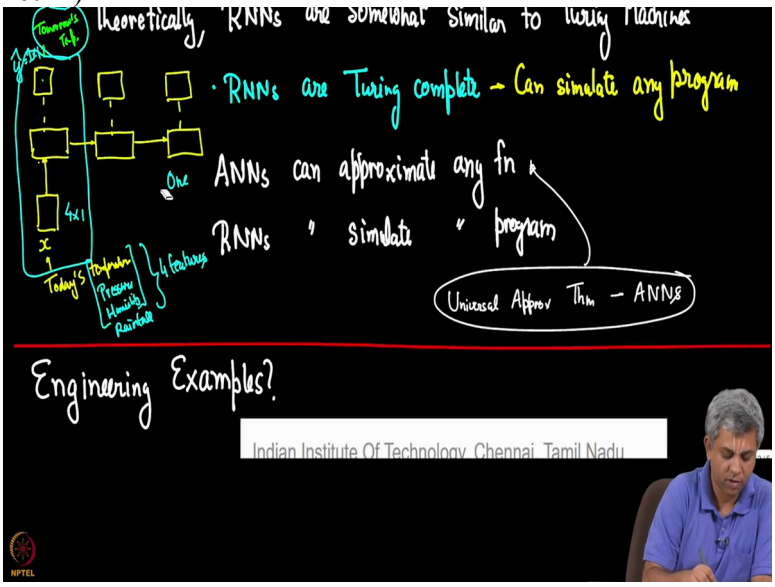
(Refer Slide Time 19:31)

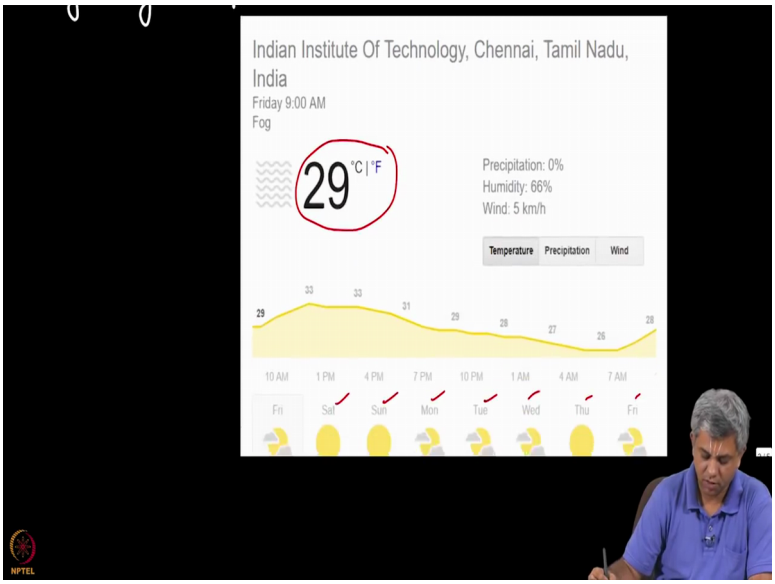
So let me show you a problem and we will come back to this picture. Okay. So let us say I want to find out today's temperature. I go on Google and indeed Google gives me today's temperature or yesterday's temperature as it might be okay. Now not only does it gives that, it gives you a few other things. It gives you tomorrow, day after, the day after that, et cetera, et cetera. Now how does it do it? Now of course, they are not using RNNs here, I want to be very clear and even

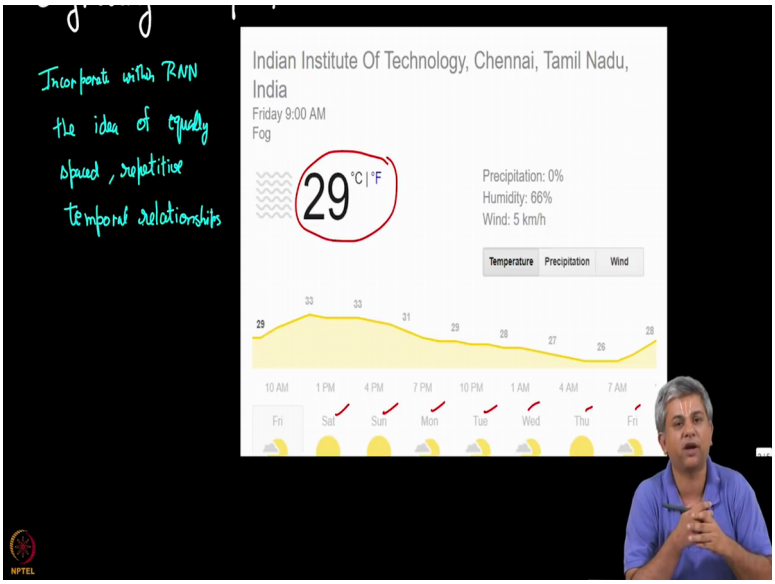
though I am using weather prediction as an example or temperature prediction as an example, it is a very simple example with which we can kind of understand RNN ideas but I do not think that Google is actually using any such thing.

It is probably using very very traditional tools. RNNs have not yet become very good enough to actually do weather predictions as of now, as far as my knowledge goes. Okay.

(Refer Slide Time 20:24)







But let us come back to this picture here okay. So let us say I make a simple prediction, I will erase this for a short while and then I will reintroduce this. So let us say I have today's weather, this X here denotes let us say today's temperature and this is my RNN structure, my RNN structure works this way. This here will predict let us say tomorrow's temperature. Now X just to be a little bit more realistic, apart from today's temperature, I give today's temperature, today's pressure distribution of you know various fields, let us say today's humidity and today's rainfall, let us see all these are my inputs.

Let us say I am taking 4 features. Okay. And tomorrow's temperature is my sole output. So Y hat is a scalar 1 cross 1. X hat or X is a vector which is 4 cross 1. Okay. So let us say I have a neural network, this is that neural network. Very simple neural network that is trained in the following way. You give today's temperature, pressure at a particular place or a particular height or something, humidity, rainfall and then you predict tomorrow's temperature. This would be a one-to-one prediction and that is just an ANN, it is not an RNN at all, okay

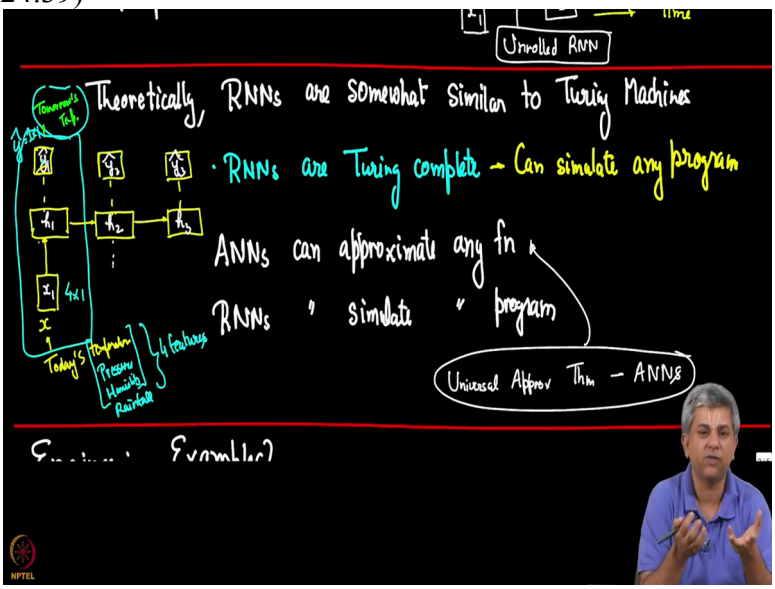
But suppose we want to do this task. Not only do I want to predict Saturday, I also want to predict Sunday, okay, from just today's temperature and today's pressure, etc, find it, okay. So we are not doing anything else. Now can we exploit something? That the interval between Friday and Saturday is the same as the interval between Saturday and Sunday. So if I say that this output was some function of X, this output should be somewhat similar, okay in its relationship to this output.

Now that is kind of encoded within an RNN. So what do you do? You recognise that somehow the relationship here is similar to the relationship here and from Sunday if I want Monday, that relationship is also similar. Then again Monday to Tuesday, so on and so forth which is what is being done. Of course what will happen if you do this in practice is if you have a smaller error at some place, it will kind of propagate and it will grow but this is the basic idea behind an RNN which is you try to incorporate within the RNN, the idea of equally spaced, repetitive temporal relationships which is that you want the same relationship between H2 and H1 as is there between H3 and H2 as is there between H4 and H3, so on and so forth, okay.

So this is a very simple idea. When I write it in a formula, you will see that maybe all this explanation was actually completely unnecessary. Okay. Now before I go forth and all this expression, even though you can keep it at the back of your mind for intuition, it is not necessary that deep learning researchers will agree with me that this is the basic intuition behind RNNs okay. This works very well for scientific and engineering problems. For language problems, it is actually quite amazing that it works but that works because of the Turing complete property. Okay.

So please keep this at the back of your mind, I am just saying this for you to build your intuition specially if you are from an engineering or science background. Okay.

(Refer Slide Time 24:39)



So now that I have shown this picture, I have one H1 here, X1 here and let us say Y1 hat here. I have H2 here. Remember, I have no corresponding H2. H2 would be the temperature and pressure and humidity and rainfall tomorrow which I do not have because I am asking for all this data today. So I put the search on Friday and I want, on Friday I want Google for Saturday, Sunday, Monday, Tuesday, Wednesday, et cetera. So I cannot really get X back. Okay. But H's information is somewhat encoded within H. okay.

Even though it is after a nonlinearity, etc, it is encoded within H. So I want to extract from that H you know what could possibly be the temperature tomorrow. Similarly, I will reuse this H and try to predict the temperature day after tomorrow. So how do we do this mathematically? Sort of anti-climatically mathematical it is very very simple, okay.

(Refer Slide Time 25:46)

Mathematically, we will write it this way. In any box you have the following going on. You have HT, sometimes you have XT and sometimes you do not. You have HT minus 1 coming from the previous instant of time and you will see later that sometimes you take out an output and sometimes you do not, okay. Now HT as I wrote before is a function we will call this variously FW, G, et cetera. Function of HT minus 1 and XT. Now what is the most general function we typically use within neural network?

It is very simple. We take linear combination followed by non-linearity, always that. Now typically, in RNNs we usually use Tanh for the nonlinearity in the hidden layers, okay. So in this

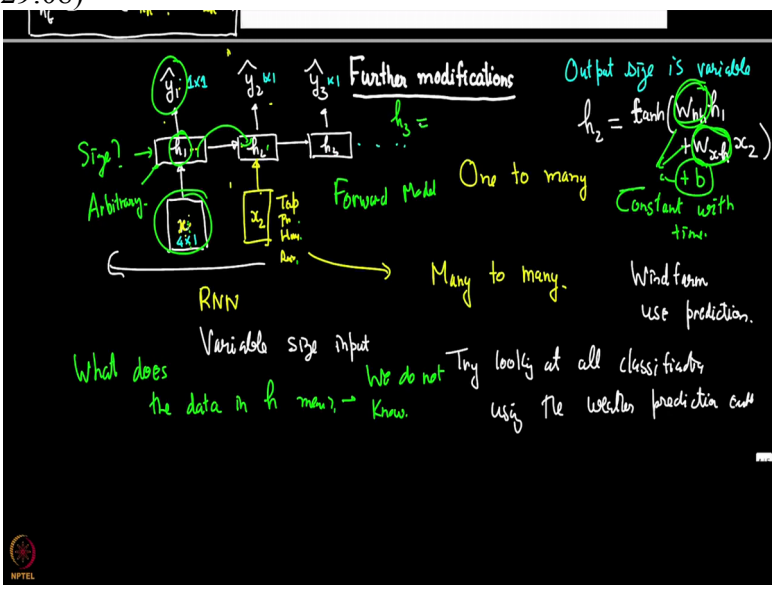
case, this will be Tanh and we need a linear combination of H and X. So there will be some weight matrix which we will multiply H and some other weight matrix which we will multiply in X. So these 2 weight matrices general will be different. Not only that, they will also have different sizes as you will see very shortly.

Now this weight matrix, we will call HH because it takes an H and gives out an H and this weight matrix, we will call XH because it takes an X and gives out H okay. So this is the general formula for the hidden layer of an RNN, some people will replace this tanh by FW or by G. okay. Now what about this YT hat? YT hat is equal to some function of HT. Now in some cases, it simply makes sense for this function to be a linear function. In some cases, it makes sense for the function to be a non-linear function.

Also it depends on whether you are doing a regression task or a classification task. If it is a classification task and let us say it is a binary classification task, then J will become a Sigma 8. If it is a multiclass classification task, you will use a soft max. And I will show you one example in the next video where we will have a fully connected layer with a full nonlinearity followed by a soft max. So all this can be, it can be linear, non-linear depends on what you want to do. Okay.

So in something like the task that we have chosen, we would probably make this into a simple you know if it is a single output, they will have like a Sigma 8 followed by a linear layer, something of that sort. Okay.

(Refer Slide Time 29:08)



So let us do some further modifications on this basic example. So the example I started with was I have X, X goes into H1 which depending on how far I want to predict in the future, goes into H2, H3, et cetera. Let me erase this, I will come back to this later and I take out temperatures. This is a classic one too many case. Okay so this is the case that we just saw. Let me put some numbers just for you to get a little bit of clarity. So let us say X is 4 cross 1, okay. We already know that Y1 is 1 cross 1, I am only predicting the temperature.

X took more thing. If you find pressure strange, you can look at events speed or something on that sort. So let us say we have some 4 features that we are taking in as input and we are predicting tomorrow's, day after tomorrow's temperature, so on and so forth, okay. How many? As many as we want. So here itself, you see the variation in sequential length. Not only can the input be of varied length, the output also can be of varied length because you can predict forever as against an ANN or a CNN which will be able to predict accurately or not, its output size is also fixed okay.

So here in output size is variable. So that is another property of RNNs okay. Now let us see how input size can also be variable. Okay. So let us say I made this prediction. It took today's temperature, wind speed, pressure, whatever, I gave some humidity and let us say precipitation. So I took these 4, I predict today's tomorrow's temperature, okay. Now tomorrow came and I

actually remeasured this data. Took that day's temperature, I think I call it pressure, humidity, and rainfall and then I made this.

Now why cannot I make simply this prediction? I would like to reuse this old data. Okay. So yesterday's data also tells me something, today's data also tells me something, okay. So now instead of making this an independent ANN, I actually reincorporate this data by using this hidden layer input and this would actually become a RNN. Now I can use this and I will get slightly modified vice now. So now if I include this, this becomes a many to many prediction. Okay.

So many to many prediction could be, I have maybe 2 to 3 days data and I predict 2 to 3 days' data later. In fact recently, Google had claimed actually they have not put up the full paper online as yet that they have been able to predict an example of the RNN prediction could be wind farm usage prediction. So this is fairly recent, about a week ago which would make it around March 8 or 9 of this year they made this and they were able to predict you know how much their load would go up or come down and that would almost certainly be an RNN prediction.

What would they use? So they would use historical data of all the usage so far. So you can see now you will have a variable size input. A variable size input would be as and when data keeps on coming, you can keep on adding this new data and your answers will change. Now an example of this in fact, a good example of this is when you start typing in some search terms within Google or nowadays even within Gmail, when you start typing something, as you change what you are typing, its prediction will change for what you are doing okay.

In fact even cellphones do that. I am not sure which of these use an RNN but I am fairly certain that Google is using currently some version of some RNN sitting there with a combination of traditional AI techniques but that would be an example of a variable size input. As you give more, it gives you better or different prediction. You cannot do this with traditional ANNs or CNNs. Okay. So an example of a many to many thing would be, I give many day's data and you give out future production of many days temperature. Okay. So that would be many to many.

And I would recommend that try looking at all classifications using this temperature example and you will see that it fits in very well. okay. You will see that you are able to find an analogy

and that is good for people from an engineering or science background because we are not going to do, that is a language task in this module at least in this course. Okay. Before I end this video, I want to give you some idea of the numbers that will actually exist in this.

So let us take H2. H2 will be F of let us say tanh of WHH H1 plus WXH X2 plus in general we should I add a biased unit, I will talk about this in the next video. Now you need to think about what is the size of this H. okay. X was size 4 cross 1, Y was size 1 cross 1. What about the size of H? The size of H is like the size of any hidden layer of a neural network, this is arbitrary. Okay. So you can give 100 neurons, you can give 200 neurons, how many ever you want. Obviously you want to give a smaller number of neurons and make sure that your prediction is fairly good.

Now one question I am pretty certain will come, it would have come for you naturally even within ANNs or CNNs. What does H mean or what does the data in H mean? So we have a meaning for this or we ascribe some meaning for X which was temperature, let us say humidity, rainfall and pressure. Y was tomorrow's temperature. What does H mean? Actually we do not know. So what do we do? We simply always remember that this is simply the forward model. In a forward model, we simply postulate, we simply guess for a relationship between X and Y hat and this is the relationship that we are guessing okay.

We guess this relationship and we just calculate about these Ws are, that is all, that is all we do. And whatever those Ws turn out to be, H has no further meaning other than that. As we have said a few times during the course, the meaning of this, trying to interpret this is an open research problem. So nobody knows this, so why should this H go to this H? All we can say is somehow the information behind X is being stored in H and you want to reuse it. okay. Since you do not explicitly know the pressure, humidity and you know rainfall you sort of guess that somehow that information is hidden in H and I will reuse it. okay.

Now one final very important point which is what gives the power to RNNs is that this WHH and WXH and B, these things are constant with time. What does that mean? If I write down H3, H3 is tanh of WHH H2 plus WXH X3, you will notice that there is no X3 in which case this gets wiped out, plus B, the biased unit. Now this WHH is the same as this WHH. So you do not change weight matrices as you go forward okay. So you do not change them at all because that would mean a tremendous number of parameters for RNNs.

The power of RNNs or the reason why they are useful is that you can use one WHH, one WXH and get done, okay. So that is what gives RNNs hour. In the next video, we will be seeing a short example, a sort of a more detailed example than this one, the temperature one and we will show it to you in MATLAB and hopefully things will become a little bit clearer, you will also get a little bit clearer about what variable size sequential data means, thank you.