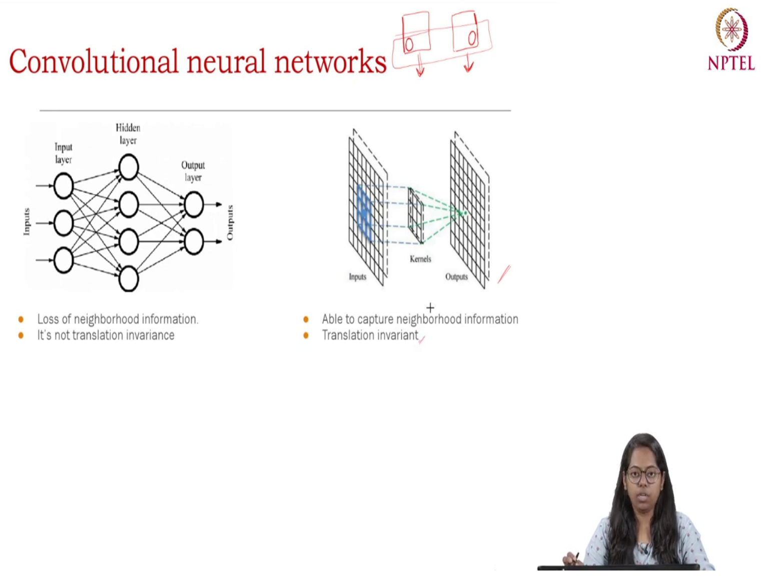
Social Network Analysis
Prof. Shivani Kumar
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Chapter - 09
Lecture - 03

Hello everyone, welcome back to Social Network Analysis and the introduction to deep learning. So, in the last few lectures, we saw the applications of deep learning, we saw the neurons that are basically the building blocks of deep learning architecture. We saw the perceptron, we saw feed forward networks and how to learn the weights and biases of that network using the back propagation algorithm.

In today's lecture, we will see two more architectures of the deep learning paradigm that as basically the Convolutional Neural Networks or the CNNs and the Recurrent Neural Networks or the RNNs. So, let us just get started.

(Refer Slide Time: 01:10)



Convolutional neural networks

- Loss of neighborhood information.
- It's not translation invariance

- Able to capture neighborhood information
- Translation invariant

So, first we will talk about convolutional neural networks. So, in the last lecture we saw something called feed forward neural network which basically constitutes different perceptrons in a hierarchical fashion right. But, in this structure, in this kind of architecture the neighborhood information is not captured, that is each data point it is treated as an independent point in the space which might be true for certain cases.

But say for example, in image classification task, we suppose we want to identify whether an image contains a dog or not. Then, in order to do that we must be aware, we cannot look at each pixel as a separate data entity. Since, the like a patch of the pixel, a collection of a pixel will be constituting the say the dog for in this example.

So, we want to look at a particular set of pixel or a particular neighborhood of a pixel, in order to identify or in order to achieve a task of identifying whether there is a dog in the picture or not.

Now, this kind of task, this kind of neighborhood capturing that is not possible in the traditional feed forward neural network. But, this kind of this kind of problem, it can be handled by something known as a convolutional neural network. See, this convolutional neural network it is able to capture the neighborhood information. We will see how this capturing happens.

Now, another thing that is interesting to note here is that a traditional feed forward network is not translation invariant. Now, what is translation invariant is basically say for instance in the example, we just said that whether an image contains a dog or not. Now, an image they can be like multiple images where in the first image the dog is present in the bottom left corner of the image right. And, now it can happen that you flip the same image vertically and now the dog is present in the bottom right corner of the image right.

Now, for these two image the output that we should get should be one, since the dog is present in both of these image. However, if we use a normal feed forward network, it will it might happen that the output for this and this image are different. Although, the two image are completely same, but just the second image is the flipped version of the first image, but still the it can happen the output can be different.

However, in the case of convolutional neural network, since we are using something known as kernels and filters over the image, there is it cannot happen that the output of such an instance is different. It if the network is trained properly, then for both the images the output should be true, that is yes the network does the image does contain a dog. So, that is why we say that Convolution Neural Network or CNNs, they are basically translation invariant.

(Refer Slide Time: 04:51)

Convolutional neural networks

Is there a flower in this image?

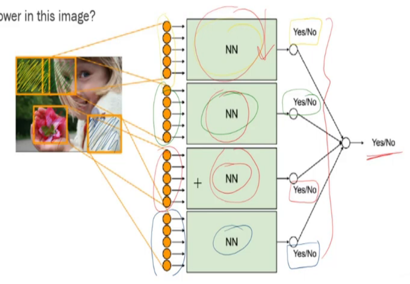
Now, moving forward, let us first just see like just as the example we just said that whether there is a dog in the image.

So, here is a very similar example that suppose we humans we want to identify whether there is a flower in this image or not. In order to do that, what we will do is that we will basically look at different segments of the images of this image right. And, we try to identify whether there is a flower in each of these segment or not. And, if there is a flower in any one of the segment, then we say that yes there is actually a flower present in this image.

(Refer Slide Time: 05:27)

Convolutional neural networks

Is there a flower in this image?



So, we want to mimic this similar or same behavior in a neural network correct. Now, how to do that? Now, for each of the segment that we have we want to apply some kind of filter over it, that is some kind of; some kind of transformation over it. So, that we can know that based on the output of this transformation or the based on the output of this filtered value that we have whether this output represents the pattern of a flower or not.

So, basically for each of the different segments that we have divided our figure into, the image into we apply a kind of filter over it right. We apply a kind of transformation over this segment. So, for example, the first segment here we see this segment, we apply this filter over it that is we multiply the values, the pixel values in this segment with say some real values, some a matrix of a real value and which the matrix which basically let us say identify the flower on a higher level.

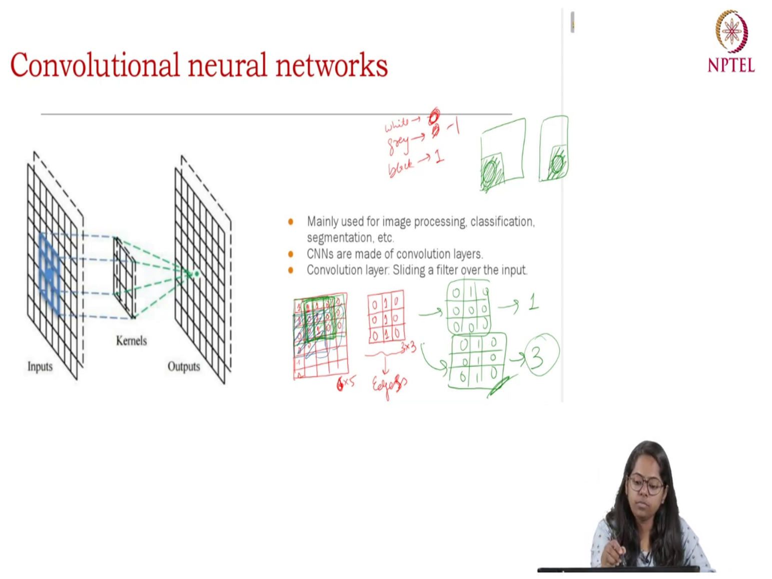
So, we will go into the depth of this right now. But yeah so, basically it if it identifies the matrix is made in such a way that identifies the flower whether the presence of a flower in the image. Then, we want to multiply this matrix to in over all the segments of the images. For example, this first segment here we multiply the matrix here and obtain some value here and say like check whether this particular matrix represents a pattern of a flower or not.

Then, again we have the second value here, we take this segment and we use these values, we calculate we have this filter over this particular segment, we multiply it. And, we see with the resultant matrix that we get represents the pattern of a flower or not. Then, again we have another segment here where we take this segment here, we multiply it with the with some weights, some filters and check whether this the resultant values represents the pattern of a flower or not.

Finally, we have another here where we multiply with the values, we have the filter and we check whether the there is a flower or not. Now, if any of these values are yes, then we can say that yes there is a flower in the image. Now, and how to do this? Now, this NNs here they are not like the feed forward network.

So, we are talking about filters and transformations. So, if we are talking about that, then it is necessary that we talk about convolution neural network. Since, these NNs here, these Neural Networks here these are basically convolutional neural network.

(Refer Slide Time: 08:43)


Convolutional neural networks

- Mainly used for image processing, classification, segmentation, etc.
- CNNs are made of convolution layers.
- Convolution layer: Sliding a filter over the input.

So, a Convolutional Neural Network or a CNN, it is basically used for image processing task like classification, segmentation and so on. And, this kind of a neural network it works on a 2D kind of an input and it this network is made up of convolution layers. Now, what is a convolution layer? Now, basically a convolution layer is when you slide a kernel over the input. So, a kernel is basically a 2D matrix which acts as a filter in order to identify the different patterns in the image right.

So, for example, suppose we have a grayscale image with some kind of a values of the grayscale pixel right. So, we have a matrix here, where let us say that if like we have some values here let us say that if the background is grey, we are getting a value of say 0. If it is white, we are getting a value of minus 1 and if it is black we are getting a value of 1. Then, we have some values here 1 1 0 0 minus 1 1 and 1 0 0 0 and so on.

And, what we do is we create a kernel or a filter of size 3 cross 3, now this image we saw it is of size 4 5 6, it is of size 6 cross 5 and what we do is we create a filter of 3 cross 3 here, such that it identifies edges. So, we have 1 1 1 in here, we say that; we say that we want to identify such pixels in our original image such that the right and the left side of that pixel is basically white. And ok so, for suppose 0 represents white and minus 1 represents grey and 1 represents black right.

So, the right and left is basically white and in the middle we have the black pixels. So, this might, this kind of filter might help us identify the edges in the figure right. So, if we are to multiply this kind of filter over here so, we have 1 1 1 1 and 0 0 0 0 0 0 here right. So, now, what we will do is we will take this filter and suppose we put this filter over this part here and we multiply. So, now, we do point wise multiplication. So, now, we will have the output as 0 0 0 1 0 0 and 0 0 0 right.

Now, based on this output like we will be getting we will slide this filter one at a time over here. So, for this second part, if we are to put this filter over this part then for this part the output will be something like 0 0 0 1 1 1 0 0 0 correct. Now, similarly we will have some values for like if we slide the filter one more to the right, will be we will have like this part to consider.

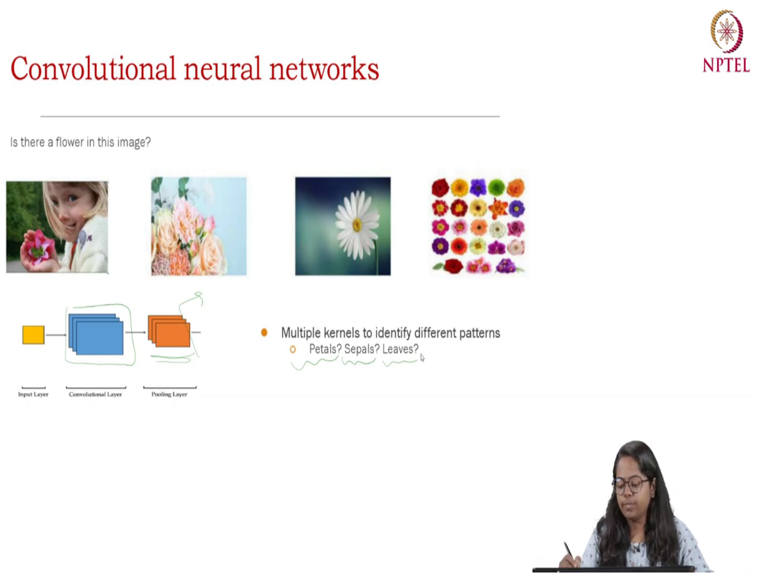
And, then we will slide the filter one to the bottom to like one step down, then we will have then we will have this part to consider right. Then again to the right, again to the right and then again to the bottom of it right. So, we will have some values for each of these filter mappings that we have correct. Now, we can perform something known as pooling in order to get one single value out of this filtered value.

So, here suppose we are doing just max pooling that we have that is we are trying to find out the maximum value of the resultant output. So, we have here 1 and or say in order to find the edges I think it would be better to do like summation kind of pooling. So, here the output is 1 that is we are just summing all of these and here it is 3, that we are summing all of these.

So, we can say that yeah this the higher the value, the denser the region is that is the more edges it represent, that is more there are more 1s in the middle column right. So, basically in a convolutional neural networks, we are trying to do this only. We are trying to create some kernels or some filters that we slide over the image in such a way; so, that different patterns are being identified and captured from different regions of the image.

So, for instance as we talked about the translation invariant things, even if the dog is present at the bottom left or the bottom right corner; whenever the filter that is a that is constructed to identify a dog is being is being like is passing over these pixel values, it will be giving us the same value even if the dog is present on the left or on the right. So, yeah so, this is how convolution network work with the help of these filters in order to identify different patterns.

(Refer Slide Time: 14:50)



Convolutional neural networks

Is there a flower in this image?

- Multiple kernels to identify different patterns
  - Petals? Sepals? Leaves?

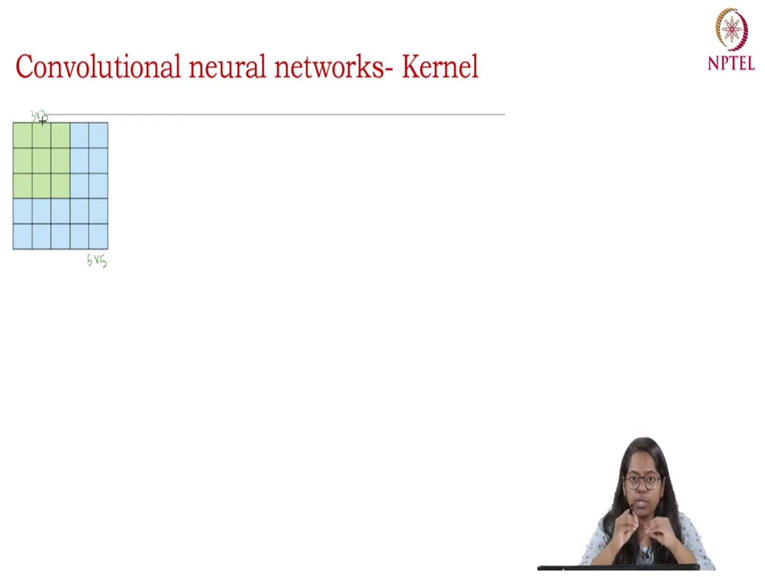
So, but we cannot like in the case as we saw in the feed forward neural network case, we cannot have a filter or we cannot have a single perceptron in the case of feed forward. And, in the case of convolution network, we cannot have a single filter which is able to identify complex objects like dog or flowers in this case. So, since the flower can be of multiple type, it can be of different types.

So, all of these figures they contain flowers, but each of the flowers there are of different kinds. It can be roses, it can be a sunflower, it can be a daisy. So, we must identify some basic patterns of a flower instead of identifying like complete flower in the image. So, what we might do, what we might want is to have multiple filters or multiple kernels passing over the image; so, that different attributes of the image is being captured.

For example, here we have multiple kernels to identify different patterns. Here, we have like maybe one kernel is used to identify petals in the image, one kernel is used to identify the sepals in the image and another is used to identify the leaves.

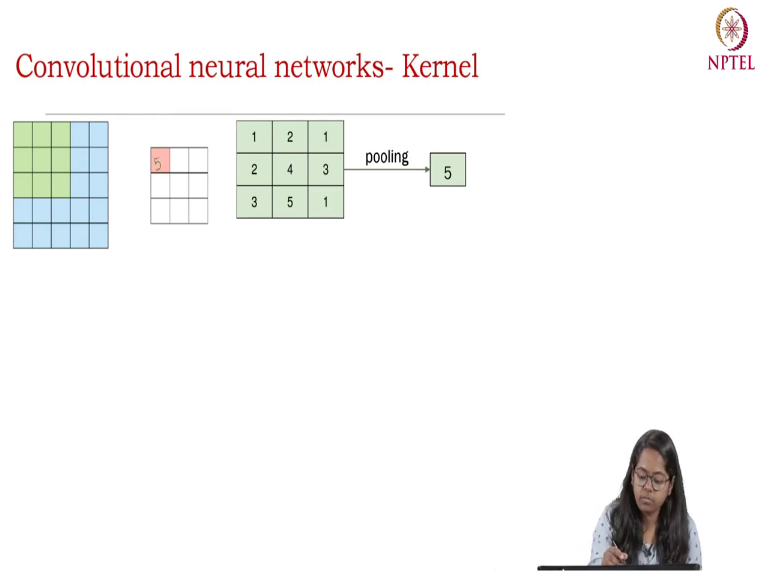
And, when we have these multiple kernels here, multiple convolution layers we will have multiple representations from the single input image which helps us to identify the different patterns in the image. And finally, these patterns will be used to identify whether the image contains the desired object or not, if it is an object classification task right.

(Refer Slide Time: 16:37)


Convolutional neural networks- Kernel

Now, how does as we already saw, but like let us just see again how does this convolution works. So, suppose the this blue matrix, it represents the image, the different pixels of a image, the different values of the pixel of a image. So, we have here like a 5 cross 5 image and this green represents the filter. So, we have a 3 cross 3 filter here which we are applying over this image.

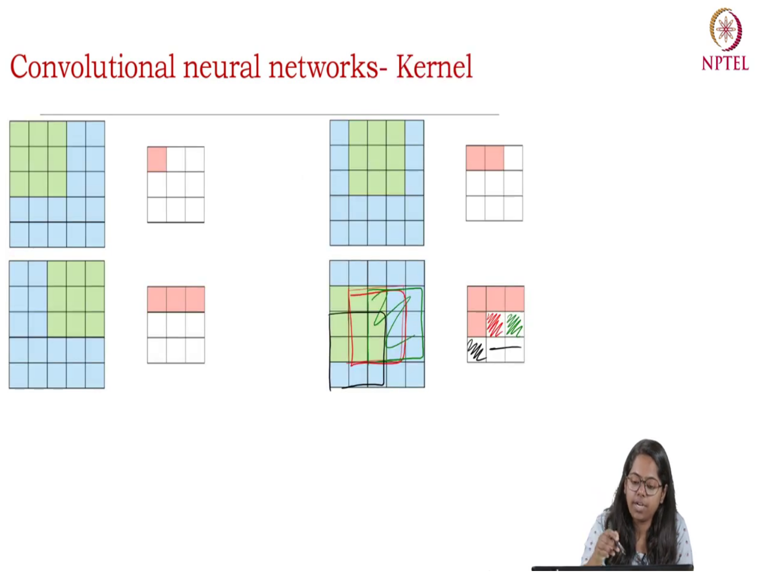
(Refer Slide Time: 17:14)


Convolutional neural networks- Kernel

Now, suppose when we do a point wise multiplication of this filter with the image; we are the result is this table right, this matrix that we have. So, now, what we want to do is in order to get a single value out of this feature map, basically this resultant matrix from the filter, we can perform pooling. So, we can either for max pooling, min pooling, average pooling, sum pooling.

So, here we just perform max pooling and just capture the maximum value, that we have in the matrix which is 5 in our case. So, this particular value will give us the value 5 and this value 5 will be a part of our final feature map that is here right. So, this is like the result that we are getting after passing the filter over the whole image. So, we put the value 5 here.

(Refer Slide Time: 18:11)



Convolutional neural networks- Kernel

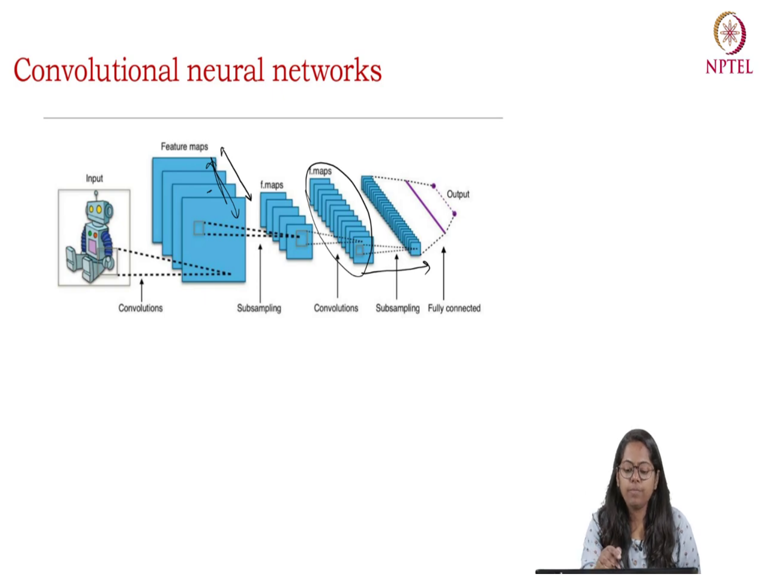
Then, we move forward, we slide this filter over the like one step to the right. So, that we can identify a similar pattern over this part of the image, if the pattern is there or not. We again do point wise multiplication, perform pooling and get the output and place it in the feature map at the appropriate position. So, this second position is also filled now which is basically the values that we get from this part of the image, the green highlighted part of the image.

Moving further, we move this filter again to the right and try to identify the pattern whether it is present in this part of the image or not and again do pooling and fill our filter map and so on, happens for the complete image right. So, if we move further then this filter will move to this part right, this part and then this part will be will capture this part of the feature map.

Then, the filter will move to further right and it will move on to this part which capture which will capture this part of the filter map. Then, again it will move on to one step down, it will come to this part, will it will capture this part of the feature map and so on. So, all of this feature map will be filled with the filtered value of the filters that we pass on the image right.

So, this is basically how on an theoretical and intuitive level, the convolution neural network works that is we are given the input as a 2D we are given a 2D input which can be an image. And, we have some filters or some kernels which identify different patterns in the input and based on these patterns, it generates a feature map of the input.

(Refer Slide Time: 20:15)



Convolutional neural networks

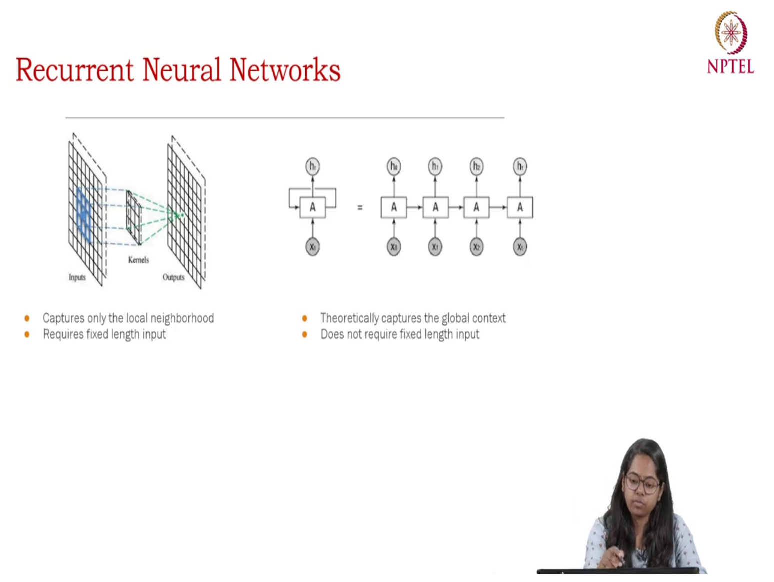
And, this feature map can then be further used to get more to being like we can apply different kernels or different filters over the over these feature maps or we can also just pass this feature map to a feed forward layer in order to perform a downstream task, say object classification.

Now, we are talking about like higher level object identification for example, like flowers, but as mentioned different features map feature map can learn different things. For example, in this like if you want to identify where is the hand of the robot, it can learn different things like edges and it one feature map can like maybe these feature map are represent a learning, where are the grey edges, where are the brown edges and so on.

And, the next feature map are basically trying to understand where are the fingers, if there is arm near it and so on. And, then higher based on the higher level of patterns that we are trying to identify the higher layers of the network will be able to identify such patterns right.

So, yeah so, this was for the convolutional neural network, it was a very brief overview, very broad overview of the structure of the architecture. And, in order to understand the maths and the nitty-gritties behind it, then you must go over some of the very interesting blogs and research papers that are present in this domain right.

(Refer Slide Time: 21:57)



Recurrent Neural Networks

- Captures only the local neighborhood
- Requires fixed length input

- Theoretically captures the global context
- Does not require fixed length input

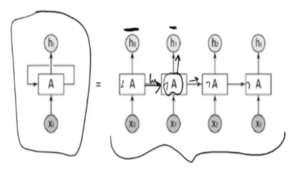
So, now, while convolution neural networks are they have proven effective for say image manipulation task, what happens if we are to like if the given input is basically is basically a sequential input; that is the input at a time step t depends upon the input at time step t minus 1 or time step t minus 2. Then, this convolution neural network might not prove beneficial; since it captures the local neighborhood of the input.

That is, if we are to provide a filter of size 3 cross 3, then it will then the convolutional layer will just capture the neighborhood of the dimension 3 cross 3 and not the global neighborhood of the whole input right. And, moreover the input length that the CNN require is of fixed size, since the convolution filter and the input it should be of like a matching size. So, that the filter can be applied to the whole input right.

So, now what we want to do is that we want to do two things, actually you want to do three things. First is that we want to handle sequential input, that is we want that we want to capture the time dependency between the inputs. Secondly, we want to capture the global context or the global neighborhood of a particular input and we want to handle an input which might not be of fixed length which can be of different lengths. In order to do that, we will see something known as recurrent neural networks.

(Refer Slide Time: 23:55)

Recurrent Neural Networks



- For sequential data
- Have the concept of memory
- Can handle inputs of varying length
- Learning: Backpropagation through time

So, basically a Recurrent Neural Network or RNN, it these kind of networks they are used for sequential data. They have a concept of something called as memory, that is they are able to remember what kind of inputs they have seen and then use that information in order to get the output for the next input right. So, they have the concept of something called as memory.

These inputs, these network they can handle inputs of varying length. Since, we I will get to it why it can handle and the learning that happens in these architecture, it is based on an algorithm known as back propagation through time. So, back propagation we already know and since it is a sequential data of the transformation in this kind of data is happening over time. So, here the algorithm is something known as back propagation through time.

But, we will not go into the depth of this algorithm and I will leave it to you guys to go into go and study this algorithm in detail. But, in this lecture we will just have an overview of how the RNN works. So, RNN can be thought of like a simple feed forward network, but it has a feedback loop that is if we just look at this left hand side of this particular slide.

We have this input x t which is being processed by this unit A which calculates like the which processes the input and calculates the hidden states and the activation functions over it just as we did in the feed forward network. And, then the output of this layer is being again passed on to the same layer and we also have something called as hidden state, we will see. So, this whole network, it can be unfolded in this fashion as can be seen at the right hand side.

So, it is easier to understand if we unfold this architecture. So, suppose at time step 0, we are passing the input x 0 to this whole architecture. And, this input x 0 it is being processed by this unit A and whatever the whatever information, it is learning at this time step it is being captured in the hidden state h 0. It is which can be passed on as an output as well as it is passed on to the next timestamp, where the input x 1 of the onetime stamp comes into play.

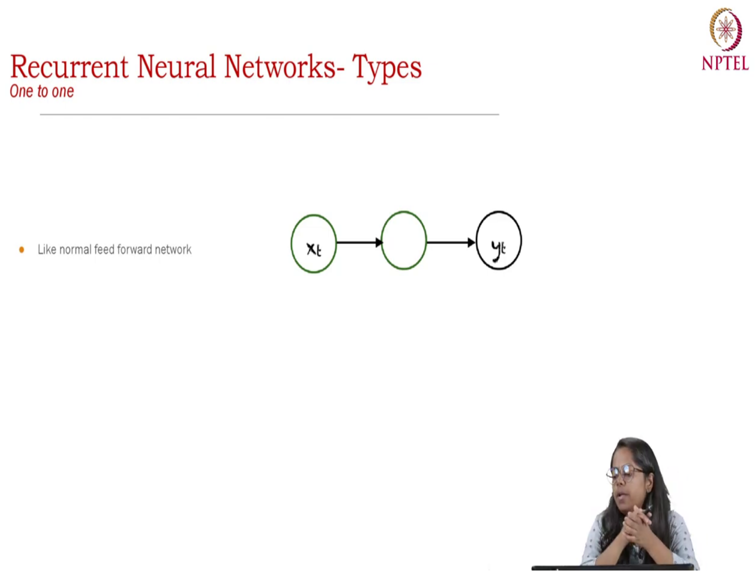
Then, this input is again passed on to the same unit A, that is the weights here of all these A are same because it is the same unit. It is just an unfolded way so, that we can better understand it, but the unit A is the same. So, here this x 1 is passed on this unit A and it also has some information from the previous time stamp, that is x 0, we have this h 0 here as well. So, based on these two information that this A is getting, this some modification will take place here, some processing will take place here.

And, it will result in the h 1 which can be directly considered as one of the outputs or can be passed on as a hidden state to the next timestamp right. This is how; this is how this whole RNN network looks like. But, you might be wondering what these inputs and outputs can be right. So, sequential data, the best example of sequential data is basically you know it is basically like a text right. So, for text if we are; if we are to read a text.

So, for example, if we are to read this last line that is learning back propagation through time. So, now, when we are on the word time here, we must not consider this word as standalone or independent. We must consider these three words that are preceding this word time so, that we can understand that what is the context the word time is present in order to understand the whole concept of the statement right.

So, this kind of sequential data where a particular word at a particular time stamp depends on the previous words, on the previous time stamp or the previous like a particular input for a particular time stamp depends on the previous time stamp inputs; this kind of data is handled by recurrent neural networks. So, now, this neural network, this RNN can be of different types based on the number of inputs and outputs that we have.

(Refer Slide Time: 29:12)



Recurrent Neural Networks- Types

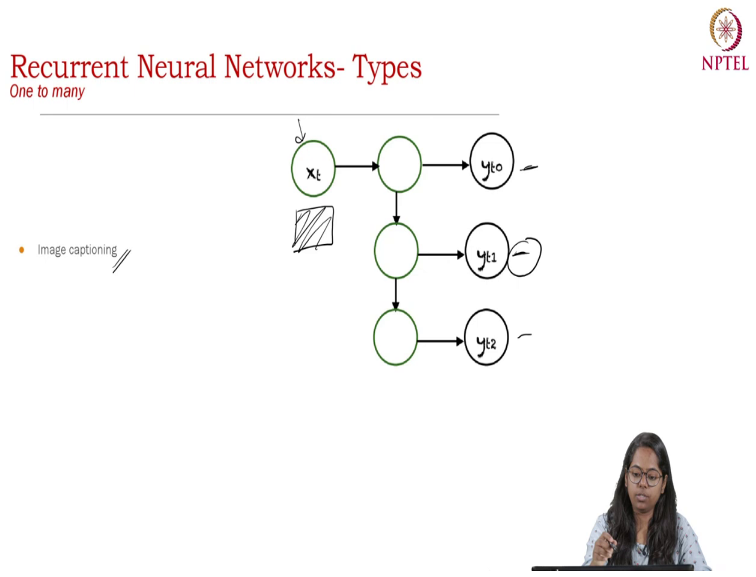
One to one

- Like normal feed forward network

So, for example, the first one can be one to one, that is it is basically normal feed forward network; we have one input and we want to produce a single output. So, for example, we have like an image classification task. We are given an image or not any image classification task, but a single data point classification task.

So, we had given a data point and we have to identify whether it will whether it is a positive data point and negative data point. Then, we can use simple feed forward network which is basically a one to one RNN.

(Refer Slide Time: 29:46)



Recurrent Neural Networks- Types

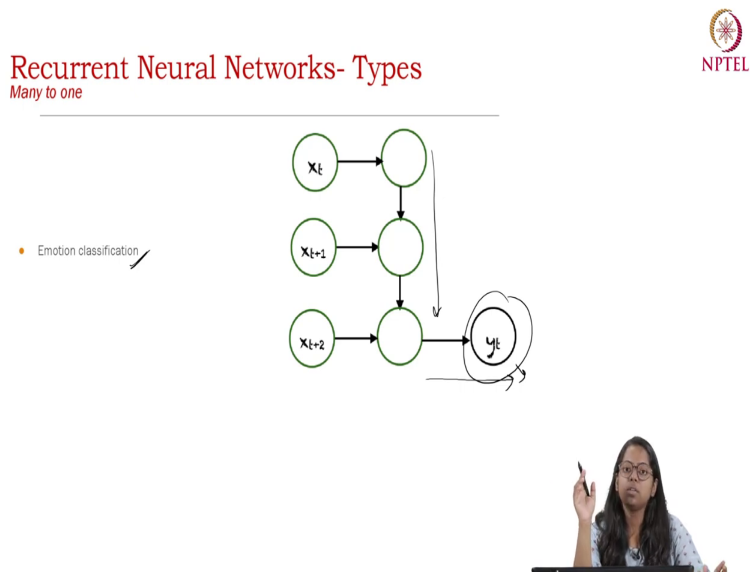
One to many

- Image captioning

Then, we have one to many RNN that is the input is a single object, but we are to generate an output with multiple elements. For example, a task can be of image captioning. So, an image is our single input, but we want to generate a caption such that the word that is present at t 1 time step depends on the word that is being generated at the t 0th time stamp right.

So, in order for the caption to be coherent, each word at the tth time step that is being generated must be aware of all the word that are already being generated from the t 0 to t minus 1th time stamp correct. So, in such a scenario, we can use a one to many kind of a structure.

(Refer Slide Time: 30:33)



Recurrent Neural Networks- Types

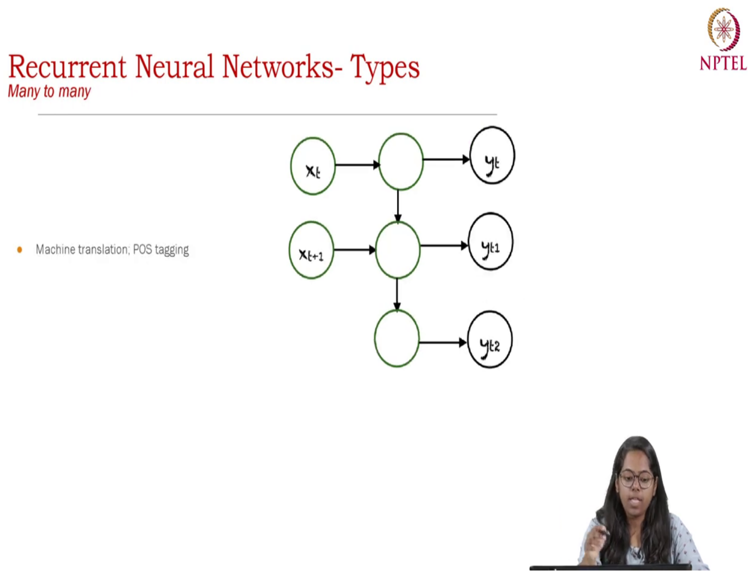
Many to one

- Emotion classification

Moving forward, we also have a many to one kind of a structure which is can which can be used for instance for an emotion classification type of an for a task, that is we have like suppose we are given a text or a tweet and we want to classify this tweet into one of the emotion. So, we are given a tweet as a sequence of words and we want to class we pass these we pass these tweet into say this RNN.

And, we are given a single output here, where we just identify the label of the emotion that is present in the tweet. So, this kind of a structure where the input contains multiple words in a sequence and we are to get just one output, this is known as a many to one structure.

(Refer Slide Time: 31:31)



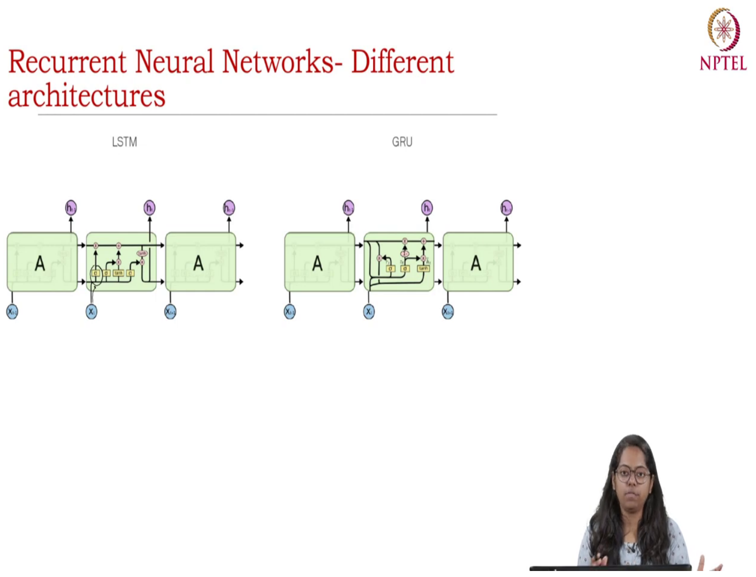
Recurrent Neural Networks- Types
Many to many

- Machine translation; POS tagging

Then, we have a many to many structure which where the input can be multiple words, if we are considering text and the output can also be multiple words. So, the one such instance can be machine translation, where we have multiple words like multiple English words, maybe as the input and multiple Hindi words as the output right.

Then, we can also have POS tagging, any other tagging and any many to many kind of in paradigm, that can be that such a network or many to many kind of RNN can be used in these type of problem statements.

(Refer Slide Time: 32:07)

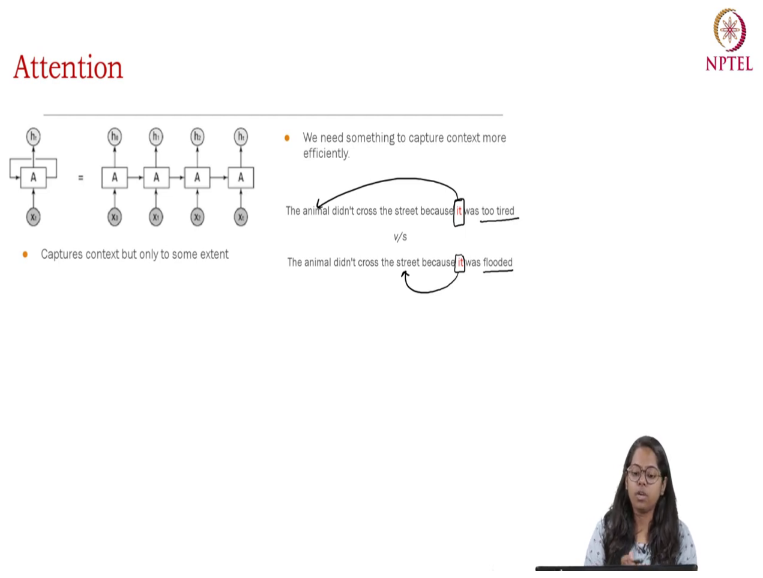


Recurrent Neural Networks- Different architectures

Now, although we will not go into the detail of the different type of architectures of this RNN, but we have some specialized architecture for RNN where called as a LSTM or GRU where. So, LSTM is short for long short Long Short Memory. And, here what we have is that we have multiple gates where a new information is being learned or stored and maybe the another kind of information the like the useless information it is being forgotten.

And, new essential information is being added to our knowledge base and then it is performing some kind of some kind of processing based on these different gates. Similarly, we have this GRU that is the Gate Recurrent Unit and we have another set of gates which perform in a different set of manner. And, these LSTMs and GRUs they are basically some different type of architectures of recurrent neural network which can be applied to different application based on the based on the type of output and the input, that we have right.

(Refer Slide Time: 33:29)



Attention

- Captures context but only to some extent
- We need something to capture context more efficiently.

The animal didn't cross the street because it was too tired

v/s

The animal didn't cross the street because it was flooded

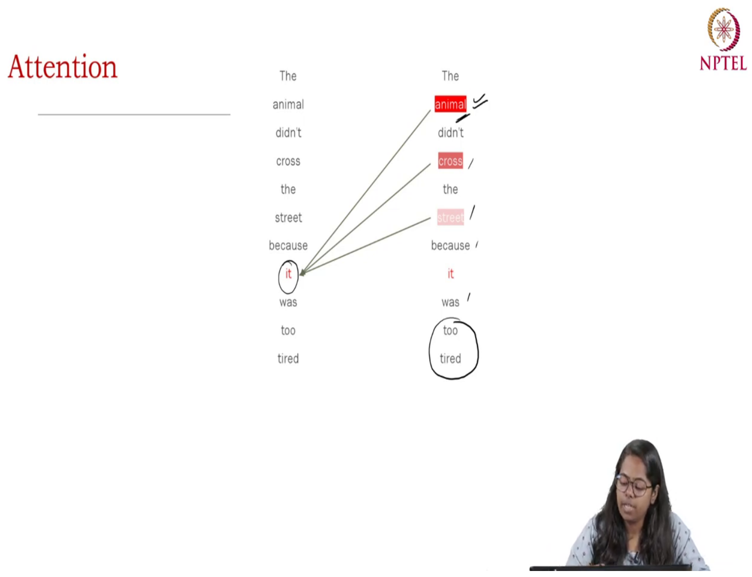
Now, we come to something known as attention. Now, we saw that in RNNs or in recurrent neural network, we want to capture context right. But, here the context is although the context being is being captured, but it is not being captured efficiently or we can say that all of the context that is present in our network is being given equal importance. But, it might happen then some part of the context is more important than some other part.

For instance, if we look at this statement, these two statements that the first statement is the animal did not cross the road because it was too tired versus the other statement is the animal did not cross the street because it was flooded. So, we see that in the first statement, since the last two words are too tired, we know that this particular word it refers to the animal; since the animal can be tired not the road whereas, if we see the next statement, then the word flooded is there in the statement.

So, the this word it, it corresponds to the word street and not animal here because animal cannot be flooded and street is flooded. But, if we are to use a normal RNN without any kind of attention or any kind of weight over it, then each of the words present in the context that is animal, cross, street will be given equal importance in order to when we are trying to get the representation of other word it.

But, we want the weight of the importance to be in such a way that animal is given more important in the first instance and street is given more important in the second instance. So, this is basically captured by this concept called attention.

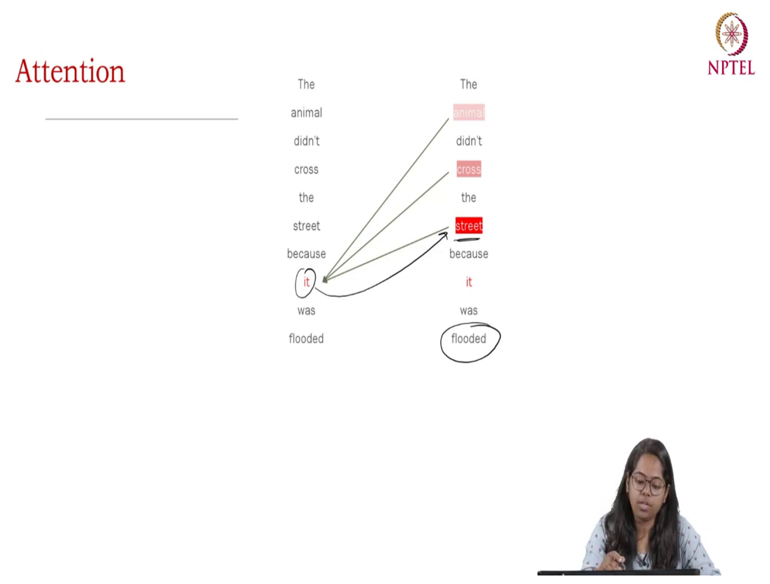
(Refer Slide Time: 35:34)



Attention

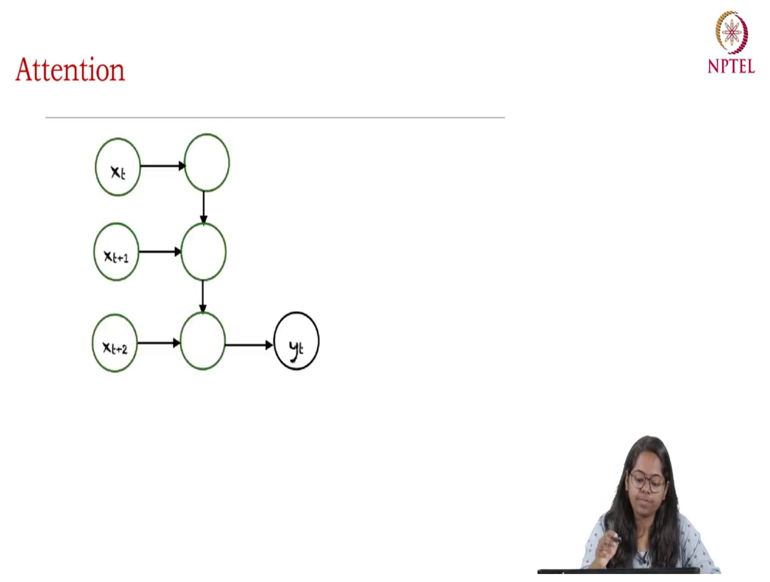
So, basically in the first case where the statement is the animal did not cross the street because it was too tired; we know that here since it is too tired here then the it refers to animal more right. This it refers to animals. So, computationally we can capture this by giving more weight to the term animal and less weight to the rest of the words right. So, this weighting is actually done by the attention mechanism that we have.

(Refer Slide Time: 36:07)


Attention

So, and in the second scenario, the animal did not cross the street because it was flooded. We know since we have flooded here, then it the word it should refer more to the word street here right. So, this kind of waiting it is done by this module of attention that we have. So, again not going into the mathematics of it just the intuition behind it. So, basically attention is something that is used to give weights to our context, that is how much attention we want to give to the different words that are present in the context.

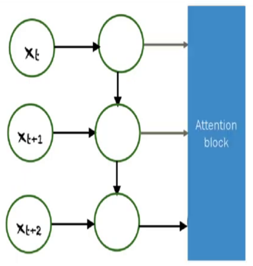
(Refer Slide Time: 36:44)


Attention

So, how can we use this attention mechanism? So, suppose we have a many to one architecture, let us say a sentiment classifier where we are given a tweet as the input and we want to classify the sentiment of it right; whether it is positive or negative.

(Refer Slide Time: 37:09)

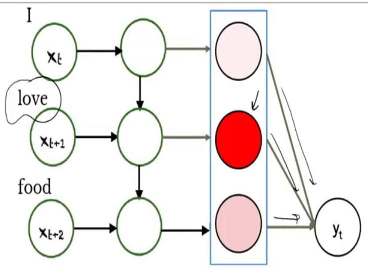
Attention



Now, instead of just performing a normal RNN architecture, that is shown in this figure; we can actually have an attention block in between, that is we pass our input to an RNN and we get some outputs from this RNN.

(Refer Slide Time: 37:19)

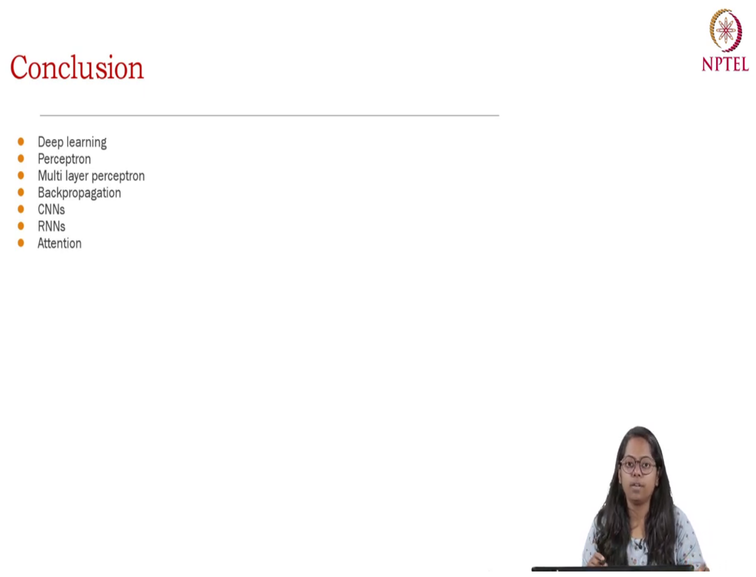
Attention



And, these outputs are then passed on to this attention block so, that we can identify how much weight do we have to give to each output in order to determine the sentiment class for the input right. So, for instance if the input is I love food, now here in order to determine the sentiment of the statement which we humans can immediately tell, that it is a positive sentiment, since it contains the word love.

Now, in for the compute to understand this, the attention mechanism what it will do is that it will give more weightage to the term love here. And, based on this more weightage here, it can identify the polarity of the sentence more efficiently right. It can identify that if it is a positive tweet, because it contains the word love.

(Refer Slide Time: 38:26)



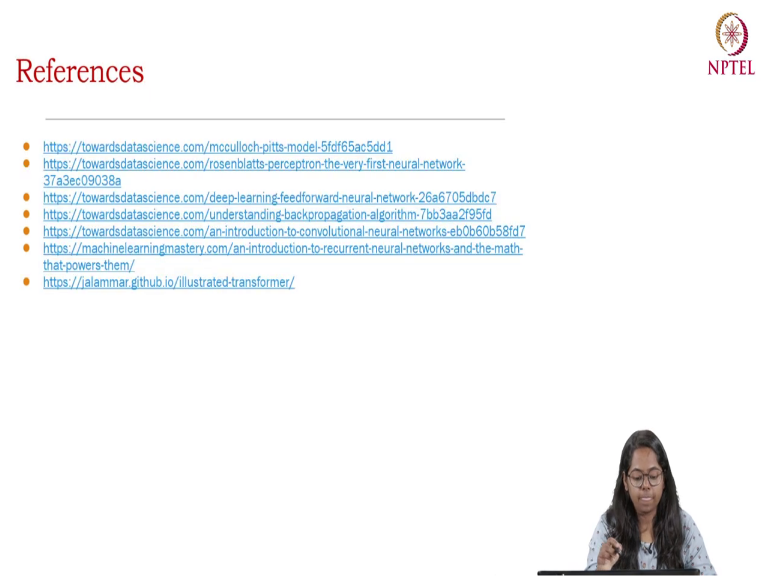
Conclusion

- Deep learning
- Perceptron
- Multi layer perceptron
- Backpropagation
- CNNs
- RNNs
- Attention

So, we saw like in all these different lectures over deep learning, we saw we like we got introduced to the to deep learning as a paradigm. Then, we saw the perceptron and how we can use it, you know multilayer perceptron fashion in a feed forward network. We saw the back propagation algorithm which is actually used to learn the different weights and biases of the feed forward network.

Then, we saw the convolutional neural network which are which is basically used in an like in a 2D kind of an input task like images and all. Then, we saw RNNs which are used for sequential tasks. And, lastly we got acquainted ourselves by the concept of attention which basically captures the different weights that we are to give to the input in order to capture the context in a in an completely efficient manner.

(Refer Slide Time: 39:21)



References

- https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1
- https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a
- https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7
- https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
- https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7
- https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/
- https://jalammar.github.io/illustrated-transformer/

So, these are some various references that you must see. So, these are the just the blogs, apart from these blogs one must also go through all the research papers that introduce these concepts the first time. And, then if you want to learn more about these concepts in a simpler or a overview kind of manner, then one can look at these blogs which are actually used to create these slides. So, that was all for the deep learning introduction of this social network analysis class.

Thank you.