

Deep Learning for Computer Vision
Professor Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture - 56
Attention in Vision Models: An Introduction



(Refer Slide Time: 00:14)



The slide features the NPTEL logo on the left and the IIT-H logo on the right. The main title is "Attention Models in Vision: An Introduction" in red text, with "Deep Learning for Computer Vision" above it. Below the title, the speaker's name "Vineeth N Balasubramanian" and affiliation "Department of Computer Science and Engineering, Indian Institute of Technology, Hyderabad" are listed. A small video inset of the speaker is in the bottom left corner. A footer bar contains the text "Vineeth N B. (IIT-H)", "59.1 Attention Models in Vision", and "1 / 27".

Having discussed RNNs last week, we will now move to a topic, which is very contemporary in terms of trying to address some of the technical features of what RNN brings to Deep Learning which is attention models.

(Refer Slide Time: 00:34)





Review

Question

What do you think will happen if you train a model on normal videos and do inference on a reversed video?

Depends on the application/task. May work for certain tasks, say differentiating walking versus jumping, but may not for recognizing a tennis forehand.

An interesting problem in this context: Finding the arrow of time, see Wei et al, Learning and Using the Arrow of Time, CVPR 2018





Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 2 / 27

Before we go into attention models, let us discuss the question that we left behind, which was, what do you think will happen? If you train a model on normal videos and do inference on a reversed video, hope you had a chance to think about this. It depends on the application or task. For certain activities, say maybe let us say you want to differentiate walking from jumping, it could work to a certain extent, even if you tested it on a reversed video.

However, for certain other activities, see a sports action such as a Tennis forehand, this may not be that trivial. An interesting related problem in this context is known as finding the arrow of time. There are a few interesting papers in this direction, where the task at hand is to find out whether the video is forward or backward. This can be trivial in some cases, but this can get complex in some cases. If you are interested, please read this paper known as Learning and Using the Arrow of Time if you would like to know more.


(Refer Slide Time: 01:59)



Review

- RNNs can be used to efficiently model sequential data
- RNNs use Backpropagation through time (BPTT) approach as training method
- RNNs suffer from vanishing & exploding gradients problems
- Gradient clipping can be used to control exploding gradient
- LSTM/ GRU units use gates to help mitigate the vanishing gradients problem

But...is this modeling sufficient?



Vineeth N B. (IIT-H) S9.1 Attention Models in Vision 3 / 27

So far with RNNs, we saw that RNNs can be used to efficiently model sequential data. RNNs use backpropagation through time as the training method. RNNs, unfortunately, suffer from vanishing and exploding gradients problems. To handle the exploding gradient problem, one can use gradient clipping, and to handle the various vanishing gradients problems one can use RNN variants, such as LSTMs or GRUs, which was good.

We saw how to use these for handling sequential learning problems. But the question we asked now is this sufficient? Are there tasks when RNNs may not be able to solve the problem? Let us find more about this.

(Refer Slide Time: 02:58)

RNN Tasks

Image Captioning



A woman is throwing a frisbee in a park

Neural Machine Translation

India got its independence from the British
↓
भारत को अंग्रेजों से आजादी मिली
↓
L'accord sur l'Espace économique européen a été signé en août 1992
↓
The agreement on the European Economic Area was signed in August 1992



Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 4 / 27

Let us consider a couple of popular tasks where RNNs may be useful. One is the task of image captioning. Given an image, one has to generate a sequence of words to make a caption that describes the activity or the scene in the image. Another example where RNNs are extremely useful is the task of Neural Machine Translation or what is known as an NMT. It is also what you see on your translate apps that you may be used where you try to, you have a sentence given in a particular language, and then you have to give the equivalent sentence in a different language. Both of these are RNN tasks.

(Refer Slide Time: 03:51)

Encoder-Decoder Modeling

Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 5 / 27

A standard approach to handling such tasks is given any input your input could be video, could be an image, could be audio, or could be text, you first pass these inputs through an encoder network which gets you a representation of that input which we call the Context Vector. Given this context vector, you pass this through a decoder network which gives you your final output text. These are known as Encoder-Decoder models, and they are extensively used in such a context.

(Refer Slide Time: 04:37)

Autoencoders

- An **autoencoder neural network** is an unsupervised learning model that applies backpropagation, setting target values to be equal to inputs themselves, i.e. $y_i = x_i$
- Learns a function $f_{W,b}(x) \approx x$; in other words, learn an approximation to identity function, output \hat{x} close to x
- Loss function? Mean Squared Error
$$\mathcal{L} = \|x - \hat{x}\|_2^2$$

Credit: Andrew Ng, CS294A, Stanford Univ

Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 6 / 27

Now let us take a brief detour to understand encoder-decoder models a bit more. The standard name for such encoder-decoder models is known as the Auto Encoder. Although in this case, it says that the decoder is trying to encode the input itself and that is the reason why this is called an autoencoder. Not all encoder-decoder models need to be autoencoders. However, the conceptual framework of encoder-decoder models comes from autoencoders, which is why we are discussing this briefly before we come back to encoder-decoder models.

And An autoencoder is a neural network architecture, where you have an input vector, you have a network which we call the encoder network. And then you have a concept vector or we also call that the bottleneck layer, which is a representation of the input, and then you have a decoder layer or a network that outputs a certain vector. In an autoencoder, we try to set the target value to the input themselves.

So you are asking the network to predict the input itself. So what are we trying to learn here, we are trying to learn a function f parametrized by some weights and bias wb . $f(x) = x$. Rather we are trying to learn the identity function itself and predict an output \hat{x} , which is close to x .

So how do you learn such a network using backpropagation? What kind of loss function would you use? It would be a mean squared error, where you are trying to measure the error between x and \hat{x} , which is the reconstruction of the autoencoder. Then you can learn the weights in the network using backpropagation as with any other feed-forward neural network.

(Refer Slide Time: 06:56)

Deep Autoencoders

Input Output

Encoder Decoder

Code

Both encoder and decoder can have many layers too - in standard deep autoencoders, the architecture in encoder is often mirrored in decoder (not always so though)

Credit: Arden Dertat, TowardsDataScience

Vineth N B. (IIT-H) 59.1 Attention Models in Vision 7 / 27

Now, the encoder and the decoder need not be just one layer, you could have several layers in the encoder. Similarly, a several layers in the decoder in the autoencoder setting traditionally, the decoder is a mirror architecture of the encoder. So have if you have a set of layers in the encoder with a certain number of dimensions, number of hidden nodes in each of these layers.

Then the decoder mirrors the same architecture the other way, to ensure that you can get an output, which is of the same dimension as the input. That is when you can measure the mean squared error between the reconstruction and the input. However, while this is the case for an autoencoder, not all encoder-decoder models need to have such architectures, you can have a different architecture for an encoder, and a different architecture for a decoder, depending on what task you are trying to solve.

(Refer Slide Time: 08:00)

Denoising Autoencoders

- Variant of autoencoder, where input is perturbed with noise (e.g. Gaussian), but network is asked to predict original input without noise
- Loss function? Mean Squared Error between output and original uncorrupted input



Credit: Kumar et al, Static hand gesture recognition using stacked Denoising Sparse Autoencoders, IC3 2014

Vineeth N B (IIT-H) 59.1 Attention Models in Vision 8 / 27

Just to understand a variant of the autoencoder, a popular one is known as the Denoising Autoencoder. In a denoising Autoencoder, you have your input data, you intentionally corrupt your input vector, for example, you can add something like a Gaussian noise and you would get a set of values \hat{x}_1 to \hat{x}_n , so those are your corrupted input values. Now, you pass this through your encoder, you get a representation than a decoder and you finally try to reconstruct the original input itself. What is the loss function here?

The loss function here would again be a mean squared error, this time it would be the mean squared error between your output and the original uncorrupted input. What are we trying to do here? We are trying to ensure that the autoencoder can generalize well, tomorrow at the end of training rather so that even if there was some noise in the input, the autoencoder would be able to recover your original data.

(Refer Slide Time: 09:18)




More on Autoencoders

Why should the hidden layers be smaller in size than input layer?

- Autoencoder (AE) with hidden layer with lesser dimension than input layer called **undercomplete AE** \Rightarrow AE learns a lower-dimensional representation on suitable manifold of input data
- Autoencoder (AE) with hidden layer with higher dimension than input layer called **overcomplete AE** \Rightarrow AE could learn trivial solutions, by copying input!

Are autoencoders then dimensionality reduction methods?

- Yes, indeed; undercomplete AEs are dimensionality reduction methods
- Do you see connections to PCA? **Homework!**



Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 9 / 27

With that, Introduction to Autoencoders, let us ask one question. In all the architectures that we saw so far, with autoencoders, we sorted the hidden layers that were always smaller in size in dimension when compared to the input layer. Is this always necessary? Can you go larger? Autoencoders where the hidden layers have a lesser dimension than the input layer are called complete autoencoders.

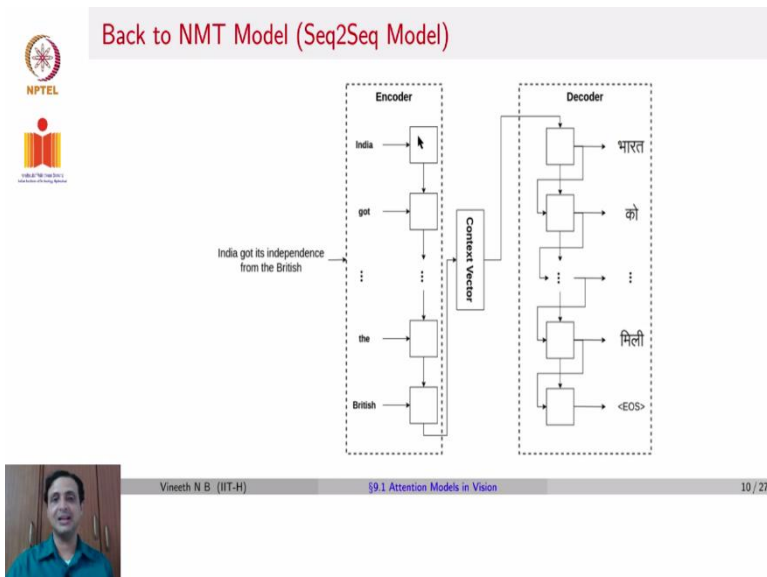
So you can say that such auto-encoders learn a lower-dimensional representation on a suitable manifold of input data. From which if you use the decoder; you can reconstruct back your original input. On the other hand, if you had an autoencoder architecture, where the hidden layer dimension is larger than your input, you would call such an auto-encoder an over-complete autoencoder.

While technically this is possible, the limitation here is that the autoencoder could blindly copy certain inputs to the certain dimensions of that hidden layer which is larger in size and still be able to reconstruct, which means such an over complete autoencoder can learn trivial solutions, which do not really give you useful performance, they may simply memorize all the inputs and just copy inputs back to the output layer.

Then the question is are all auto encoders also dimensionality reduction methods? Assuming we are talking about under complete autoencoders? Partially yes, largely speaking, autoencoders can be used as dimensionality reduction techniques. A follow-up question then is then, can an

autoencoder be considered similar to principal component analysis, which is a popular dimensionality reduction method? The answer is actually yes, again. But I am going to leave this for you as homework to work out the connection between autoencoders and PCA.

(Refer Slide Time: 11:54)



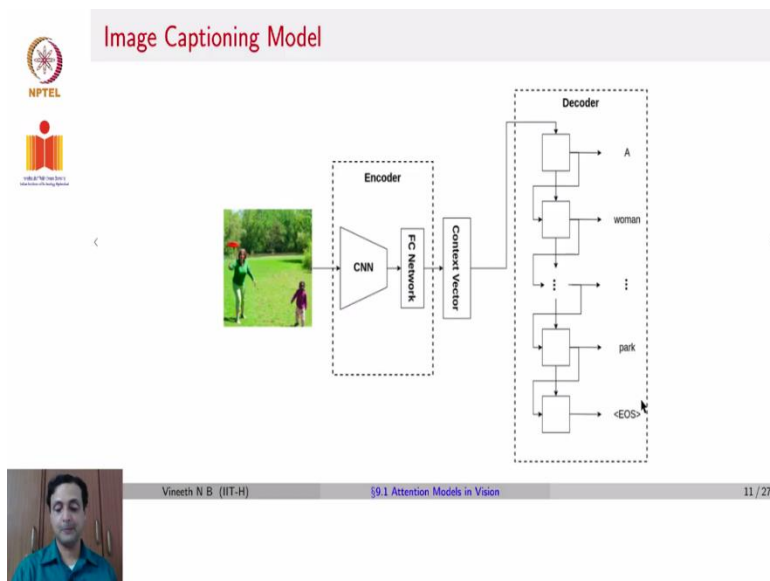
Let us not come back to what we were talking about, which was one of the tasks of RNNs which is Neural Machine Translation, or NMT. These kinds of encoder-decoder models are also called Sequence to Sequence models, especially when you have an input to be a sequence and an output also to be a sequence. So if you had an input sentence, which says India got its independence from the British.

Let us say now that we want to translate this English sentence to Hindi, what you do know is you would have an encoder network, which would be a Recurrent neural network and RNN where each word of your input sentence is given at one-time step of the RNN. And the final output of the RNN would be what we call a Context vector. And this context vector is fed into a decoder RNN, which gives you the output, which says Bharat Ko.

The rest of the sentence, Mili, and then you have an end-of-sentence token. This is what we saw as a many-to-many RNN last week. Why are not we giving output at each time step of the encoder RNN? For the machine translation task, if you recall, the recommended architecture, we said that it is wiser to read the full sentence, and then start giving the output of the translated sentence.

Why so? Because different languages have different grammar and sentence constructions. So it may not be correct for the first word in English to be the first word in Hindi, or the Hindi sentence may not exactly follow the same sequence of words in English, because of grammar, grammatical regulations. So that is the reason why in machine translation tasks, you generally have a reading of the entire input sentence, you get a context vector, and then you start giving the entire output in the translated output.

(Refer Slide Time: 14:28)



Similarly, if you considered the image captioning task, you would have an image. And in this case, your encoder would be a CNN followed by say, a fully connected network, out of which you get a representation or a context vector. And this context vector goes to a decoder, which outputs the caption, A woman dot, dot dot, say in the park end of the sentence.

(Refer Slide Time: 14:57)

NPTEL

What is the problem?

A hidden state at time step t (h_t) is a compressed form of all previous inputs (x_1, x_2, \dots, x_t)



Vineeth N B (IIT-H)

§9.1 Attention Models in Vision

12 / 27

What is the problem? This seems to be working well, is there a problem at all? Let us analyze this a bit more closely. So in an RNN, the hidden states are responsible for storing relevant input information in RNNs. So, you could say that the hidden state at time step t or h_t is a compressed form of all previous inputs. That hidden state represents some information from all the previous inputs, which is required for processing in that state, as well as future states.

(Refer Slide Time: 15:41)

NPTEL

What is the problem?

After meeting a comrade at the last post station but one before Moscow, Denisov had drunk three bottles of wine with him and, despite the jolting ruts across the snow-covered road, did not once wake up on the way to Moscow, but lay at the bottom of the sleigh beside Rostov, who grew more and more impatient the nearer they got to Moscow. <EOS>

Excerpts from the work of Leo Tolstoy (~60 words)

But what if input is very long? Can h_T encode all information without forgetting? Information bottleneck!



Vineeth N B (IIT-H)

§9.1 Attention Models in Vision

12 / 27

Now, let us consider a longer sequence. If you considered language processing, and a large paragraph, if your input is very long, can your ht the hidden state at any time step encode all this information? Not really, you may be faced with the information bottleneck problem in this kind of context.

(Refer Slide Time: 16:07)

What is the problem?

NPTEL

... to reach the official residency of Prime Minister Nawaz Sharif.

... die offizielle Residenz des Premierministers Nawaz Sharif zu erreichen.

English to German

to reach = zu erreichen

Can we guarantee that words seen at earlier input time steps be reproduced in later time steps of output?

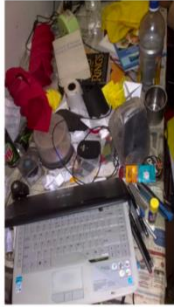
Vineeth N B (IIT-H) 59.1 Attention Models in Vision 12 / 27

So, if you considered a sentence such as this one here, which has to be translated to German, can we guarantee that words seen at earlier time steps be reproduced. at later time steps. Remember, when you go from a language such as English, to a language such as German, the position of the verbs the nouns may all change and to reproduce this one may have to get a word much earlier in the sentence in English, which may follow much later in, say the German language. Is this possible? Unfortunately, RNNs do not work that well, when you have such long sequences.

(Refer Slide Time: 16:57)

NPTEL
National Programme on Technology Enhanced Learning

What is the problem?




Visual Question Answering (VQA)
(To be discussed later)

Relevant information in a cluttered image
should also be preserved

Question: What is the name of the book?
Answer: The name of the book is Lord of the Rings.

Credit: Bharath Kishore, Flickr CC License

Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 13 / 27



Similarly, even if you had Image Captioning and related problems, such as visual question answering, which we will see later, so if you had this image that we saw at the very beginning of this course, and if we ask the question, what is the name of the book? The expected answer is the name of the book is Lord of the Rings. The relevant information in a cluttered image may also need to be preserved. In case there are follow-up questions with a dialog.

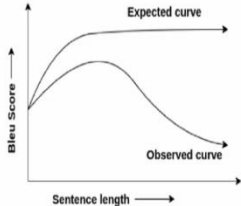
(Refer Slide Time: 17:34)

NPTEL
National Programme on Technology Enhanced Learning


Failure of Encoder-Decoder Modeling

BLEU score:

- Stands for Bilingual Evaluation Understudy
- Metric for evaluating quality of machine translated text
- Can be used for other language tasks (like Image captioning, VQA, etc)
- For more information, see [BLEU score, Wikipedia](#)



Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 14 / 27



So a statistical way of understanding this is through what is known as the BLEU score. BLEU score is a common performance metric used in NLP Natural Language Processing, BLEU stands for Bilingual Evaluation Understudy. It is a metric for evaluating the quality of the machine-translated text. It is also used for other tasks such as image captioning, visual question answering, so on and so forth.

And when one looks at the BLEU score, one observes that as the sentence length increases, then while the expected curve is that you should get a high BLEU score after a certain sequence length, unfortunately, as the sentence length goes further beyond a threshold, the BLEU score starts falling which means using such encoder-decoder models where encoders are RNNs decoders are also RNNs starts failing in these cases, when the sequences are long by nature. If you would like to know more about BLEU, you can see this entry in Wikipedia for more information.

(Refer Slide Time: 18:57)

The slide is titled "Solution?" in red text at the top left. On the left side, there are logos for NPTEL and IIT-H. The central image shows two hands cupped together, holding the word "ATTENTION" in white capital letters. Above the hands, the text "I FOUND WHAT YOU WERE LOOKING FOR" is written in white capital letters. At the bottom left, there is a small portrait of the presenter, Vineeth N B. The footer contains the text "Vineeth N B (IIT-H)", "59.1 Attention Models in Vision", and "15 / 27".

So, what, what is the solution to this problem? The solution which is extensively used today is what is known as Attention, which is going to be the focus of this week's lectures.

(Refer Slide Time: 19:13)

Attention: Intuition



How do you answer this?



What is the boy doing?

Pay attention to the relevant artifacts





Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 16 / 27

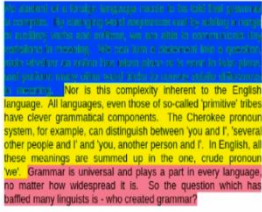
So what is this Attention? Intuitively speaking, given an image, if we had to ask the question, what is this boy doing? The human way of doing this would be you first identify the artifacts in the image. You pay attention to the relevant artifacts, in this case, the boy and what activity the boy is associated with.

(Refer Slide Time: 19:42)


Attention: Intuition



Similarly,



Summary




Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 16 / 27

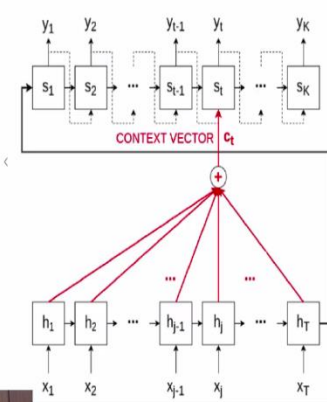
Similarly, if you had an entire paragraph, and you had to summarize, you would probably look at certain parts of the paragraph and write them out in a summarized form. So paying attention to

parts of inputs, beat images, or beat long sequences, like the text is an important way of how humans process data.

(Refer Slide Time: 20:12)




Attention Mechanism: Temporal Data



- Given an encoder, with h_j as hidden state at time-step j , and decoder, with s_t as hidden state at time-step t
- Attention mechanism creates shortcut connections between context vector (c_t) and the entire source input (X)
- Decoder hidden state at time t (s_t) given by:

$$s_t = f(s_{t-1}, y_{t-1}, \mathbf{A})$$



Vineeth N B (IIT-H)
§9.1 Attention Models in Vision
17 / 27

So let us now see this in a Sequence learning problem in the traditional encoder-decoder model setting. So this is once again, the many to many RNN setting, similar to what we saw for Neural Machine Translation. So you have your inputs, then you have a context vector that comes out at the end of the inputs, that context vector is fed to a decoder RNN, which gives you the outputs Y_1 to Y_k .

Now let us assume that head j 's are the hidden states of the encoder and S_j 's are the hidden states of the decoder. So what does Attention do? Attention suggests that, instead of directly outputting h_T , which is the last hidden state, to your decoder RNN, we instead have a context vector, which relies on all of the hidden states from the input. This creates a shortcut connection between this context vector c_t and the entire source input x .

How would you learn this context vector? We will see there are multiple different ways. So given this context vector, the decoder hidden state S_t is given by some function $f(S_{t-1})$, the previous hidden state in the decoder y_{t-1} , the output of the previous time step in the decoder could be given as input to the next time step, as well as the context vector C_t .

(Refer Slide Time: 22:05)

Attention Mechanism: Temporal Data

Context vector (c_t) given by:

$$c_t = \sum_{j=1}^T \alpha_{t,j} h_j$$

$\alpha_{t,j}$ gives degree of alignment between s_{t-1} and h_j :

$$\alpha_{t,j} = \frac{\exp(\text{score}(s_{t-1}, h_j))}{\sum_{j'=1}^T \exp(\text{score}(s_{t-1}, h_{j'}))}$$



Vineeth N B. (IIT-H) 19.1 Attention Models in Vision 18 / 27

And what is this context vector? This context vector is given by C_t , which is overall the time steps in your encoder, RNN, $\alpha_{t,j} h_j$. So it is a weighted combination of all of your hidden state representations in your encoder RNN. How do you find $\alpha_{t,j}$? How do you find those weights of the different inputs? A standard framework for doing this is $\alpha_{t,j}$ can be obtained as a softmax over some scoring function that captures the score between S_{t-1} and each of the hidden states in your encoder.

So S_{t-1} gives us a current context of the output. So we try to understand what is the alignment of the current context in the output with each of the inputs and accordingly pay attention to specific parts of the inputs? Now there is an open question, how do you compute this score of S_{t-1} with each of the h_j 's in the encoder RNN. Once we have a way of computing that score, we can take a softmax over h_j with respect to all of the h_j 's.

So we will do this for each of the hidden states h_j 's in the encoder RNN. And using that, we can compute your $\alpha_{t,j}$'s. And using $\alpha_{t,j}$'s, we can compute the context vector. Once you get the context vector, you would give the corresponding context vector as input to each time step of the decoder RNN. How do you compute this score?



(Refer Slide Time: 24:05)



Alignment Scores

Name	Alignment Score Function
Content-based Attention	$\text{score}(s_t, h_i) = \text{cosine}(s_t, h_i)$
Additive Attention	$\text{score}(s_t, h_i) = v_a^T \tanh(W_a[s_t; h_i])$
Location-Based Attention	$\alpha_{t,j} = \text{softmax}(W_a s_t)$
General Attention	$\text{score}(s_t, h_i) = s_t^T W_a h_i$
Dot-Product Attention	$\text{score}(s_t, h_i) = s_t^T h_i$
Scaled Dot-Product Attention	$\text{score}(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

Credit: Lilian Weng, Attention? Attention!, Github Blog



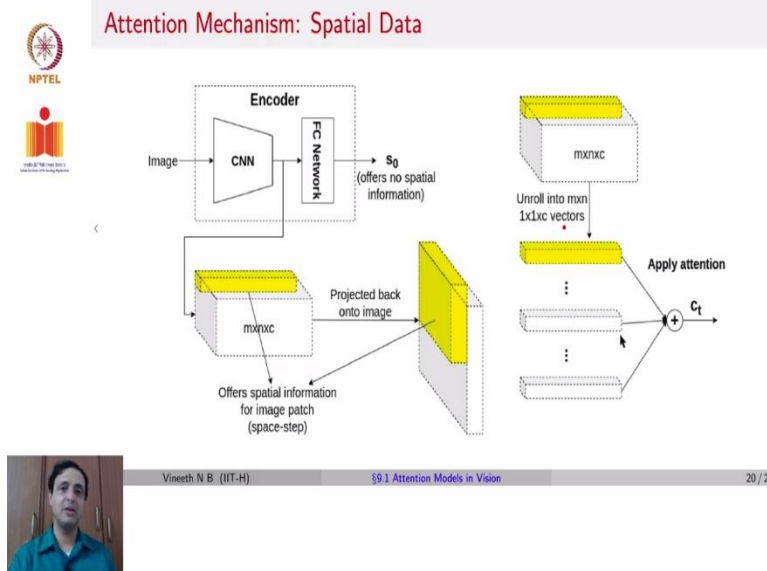
Vineeth N B (IIT-H) 59.1 Attention Models in Vision 19 / 27

There are a few different approaches in literature at this time. We will review many of them over the lectures this week. But to give you a summary, you could have content-based attention which tries to look at s_t and h_i . So each a particular hidden state in your decoder RNN s_t and a particular hidden, hidden state in an encoder, RNN h_i as cosine similarity between the two. That is one way of measuring the score.

You could also learn weights to compute this alignment. So you can take s_t and h_i learn a set of weights, take a tan h and use another vector to get the score. So this is a learning procedure to get your final score. One could also get $\alpha_{t,j}$ as a softmax over a learned set of weights, W and s_t again. One could also use a more general framework, where you have $s_t^T * h_i$, which is similar to cosine, which will give you a dot product.

But you also have a learned set of weights in between, which tells you how to compare the two vectors s_t and h_i . Remember, any W here is learned by the network to compute the score. Or you could simply use just a dot product by itself, which would be similar to your content-based attention, the cosine and the dot product would give similar values. Or there is a variant known as the Scaled dot product Attention where you use the dot product between the two vectors s_t and a h_i . But scale it by root n , which tells you the number of inputs that you have.

(Refer Slide Time: 26:06)



What about Spatial data? So, we saw how it is done for temporal data where you had a sequence to sequence RNN a many to many RNN What if you had an image captioning task if you had spatial data. So, in this case, your image would give you a certain representation s_0 out of the encoder network. Unfortunately, when you use a fully connected layer, after the CNN, you lose spatial information in that vector.

So, instead of using the fully connected layer, we typically take the output of the convolution layers themselves, which would give you a certain volume, which let us say is $m \times n \times c$. Now, we know that if you considered one specific patch of this volume $m \times n \times c$, we know that you can trace that back to a particular patch of the original image which was passed through a CNN.

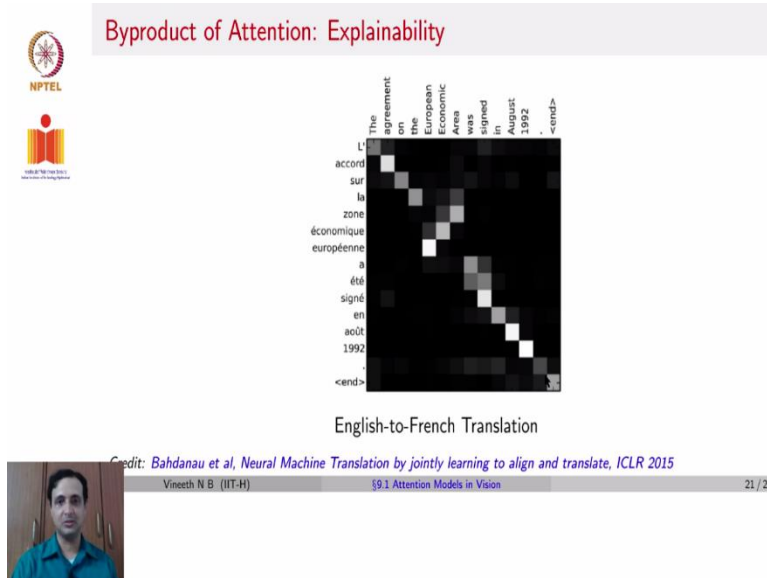
So you know the output feature map say a con five feature map, if you looked at one particular path part of that depth volume, you would get a certain patch in the input image. Now, this gives you Spatial information. So, what can we do we take this feature map that we get at the output of a certain convolution layer, we can unroll them into $1 \times 1 \times c$ vectors. So, you ideally have $m \times n \times c$.

So you can unroll this into c different vectors. And then you can apply attention to get a context vector. In what way is this useful, this context vector, now can be understood as paying attention to certain parts of the image while giving the output because each of these bands, each of these

sub-volumes here highlighted in yellow is certain parts of the input image and one could now understand the same weighted attention concept.

The alignment part of it could be implemented very similarly to what we saw on the previous slide. But now, this represents different parts of the input image.

(Refer Slide Time: 28:35)



Another use of performing Attention is it gives you the explainability of the final model. Why so how so? If you have said a machine translation task, you know, that when you generated a certain output word from a decoder, RNN, your attention model or your context vector tells you which part of the input you looked at, while predicting that word as the output, and that automatically tells you which words in your input sequence corresponded to when or a word in your output.

So in this case, you can see that this particular sequence here European Economic Area, depended on Zone Economic European, so that is also highlighted by these white patches here. So white means a higher dependence. Black means no dependence. And looking at this heat map gives you an understanding of how the model translated from one language to another.

(Refer Slide Time: 29:55)

Byproduct of Attention: Explainability

Credit: Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 22 / 27


What about images image captioning task In this case too you can use the same idea, given an image, if the model is generating a caption, you can see that the model generates each word of the caption by looking at certain parts of the image. For example, when it says A it seems to be looking at a particular part of the image, when it says A woman, it seems to be looking at a certain part of the image, while the other object is also in relevance.

And if you keep going, you see when it says the word throwing, it seems to be focusing on the woman part of the image. And if you see the word, Frisbee, it seems to focus on the Frisbee in the image. And if you see the word Park, it seems to be focusing on everything other than the woman and the child. This gives you an understanding and trusts the model is looking at the right things while giving a caption as output.

(Refer Slide Time: 31:03)

Modes of Attention

Hard vs Soft Attention:



What is the boy doing? What is the boy doing?

- Single position is chosen for full alignment (1.0)
- All positions get partial alignment weights (0-1)
- Position-choosing is stochastic and hence non-differentiable
- Deterministic and hence differentiable, as no position-choosing

Vineeth N B (IIT-H) §9.1 Attention Models in Vision 23 / 27


What are the kinds of attention one can have, you could consider having Hard versus Soft attention. What do these mean? in hard attention, you choose one part of the image as the only focus for giving a certain output, let us say image captioning, you look at only one patch of the image to be able to give a word as an output. So, this choice of a position could end up becoming a stochastic sampling problem.

And hence, one may not be able to backpropagate suit through such a hard attention problem, because that stochastic stamp sampling step could be nondifferentiable. We will see this in more detail in the next lecture. On the other hand, one could have Soft attention, where you do not choose a single part of the image, but you simply assign weights to every part of the image. In this case, you are only going to have a newer image, where each part of the image has a certain weight. In this case, your output turns out to be deterministic, differentiable. And hence, you can use such an approach along with standard backpropagation.

(Refer Slide Time: 32:30)

Modes of Attention

Global vs Local Attention:



What is the boy doing?

What is the boy doing?

All input positions are chosen for attention

Neighbourhood of aligned position is chosen for attention

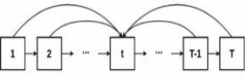
Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 24 / 27

Another categorization of Attention is Global versus Local attention. In global attention, all the input positions are chosen for attention whereas in local attention, maybe only a neighborhood window around the object of interest or the area of interest is chosen for Attention.

(Refer Slide Time: 32:55)

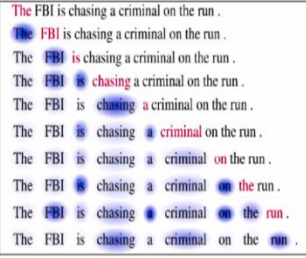
Modes of Attention

Self-Attention:



Also known as **intra** attention

Used extensively in advanced attention models (will see more soon)



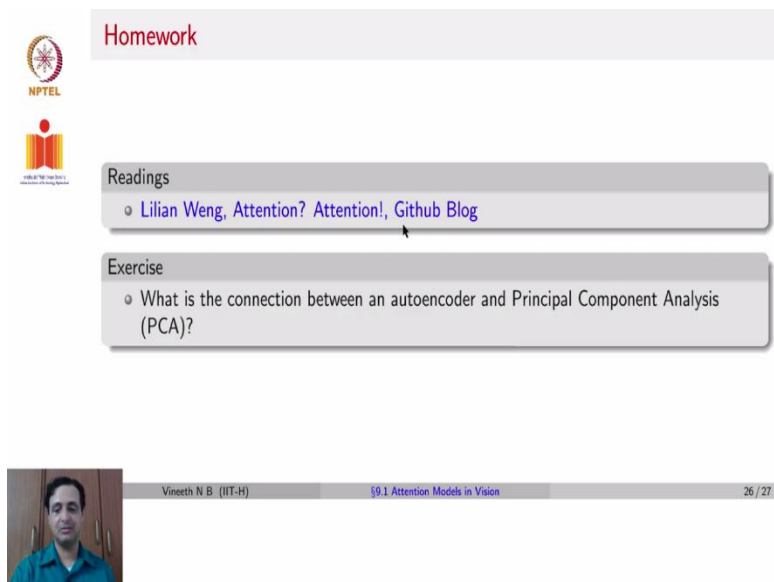
Credit: Cheng et al, Long Short-Term Memory-Networks for Machine Reading, ACL 2016

Vineeth N B. (IIT-H) 59.1 Attention Models in Vision 25 / 27

A third kind, which is very popular today is known as Self-attention where the attention is not with respect to when decoder RNN with respect to the encoder, or an output RNN with respect to parts of an image, but is of attention of a part of a sequence with respect to another part of the same

sequence. This is known as Self Attention or Intra Attention. And we will see this in more detail in a later lecture this week.

(Refer Slide Time: 33:32)





The screenshot shows a presentation slide with a light gray background. On the left side, there are two logos: the NPTEL logo (a circular emblem with a book and a lamp) and the IIT Bombay logo (a stylized book icon). The main content is organized into three sections:

- Homework**: A red heading at the top of the first section.
- Readings**: A gray heading for the second section, containing a single bullet point: [Lilian Weng, Attention? Attention!, Github Blog](#). A mouse cursor is pointing at the link.
- Exercise**: A gray heading for the third section, containing a single bullet point: "What is the connection between an autoencoder and Principal Component Analysis (PCA)?"






At the bottom of the slide, there is a small video thumbnail of a man in a blue shirt. To the right of the thumbnail, the text "Vineeth N B (IIT-H)" is visible. Further right, the slide title "§9.1 Attention Models in Vision" is displayed, and on the far right, the slide number "26 / 27" is shown.


Your homework for this lecture is to read this excellent blog by Lillian Wang known as Attention? Attention!, it is a blog on Github. And one question that we left behind, which is, is there a connection between an Autoencoder and Principal Component Analysis? Think about it and we will discuss this in the next lecture.

(Refer Slide Time: 34:01)



References

-  Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).
-  Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015).
-  Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International conference on machine learning*. 2015, pp. 2048–2057.
-  Jianpeng Cheng, Li Dong, and Mirella Lapata. "Long short-term memory-networks for machine reading". In: *arXiv preprint arXiv:1601.06733* (2016).
-  Lilian Weng. *Attention? Attention!* 2018. URL: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>.



Vineeth N B. (IIT-H) [59.1 Attention Models in Vision](#) 27 / 27

References.