**Week -3**

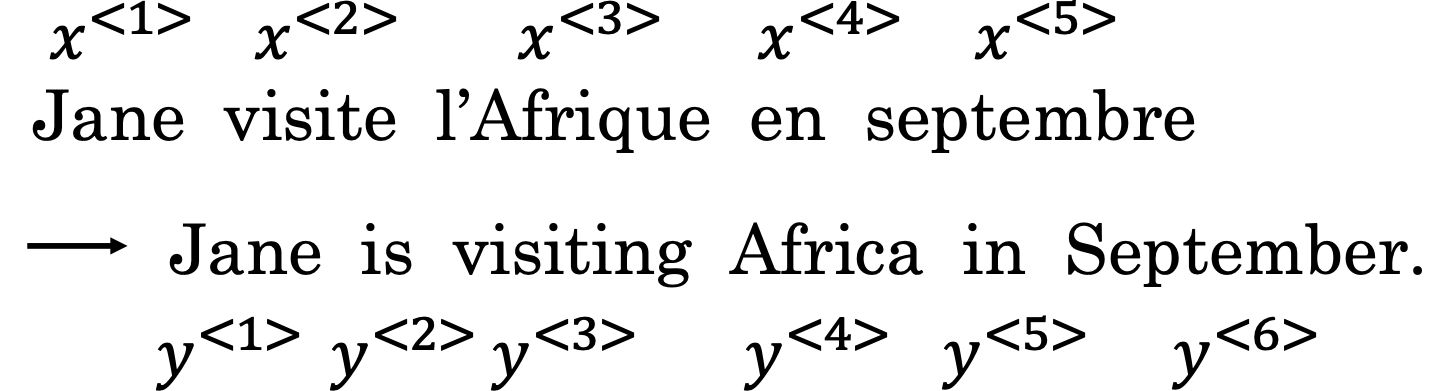
**SEQUENCE MODELS & ATTENTION MECHANISHM**

**\*Basic Models**

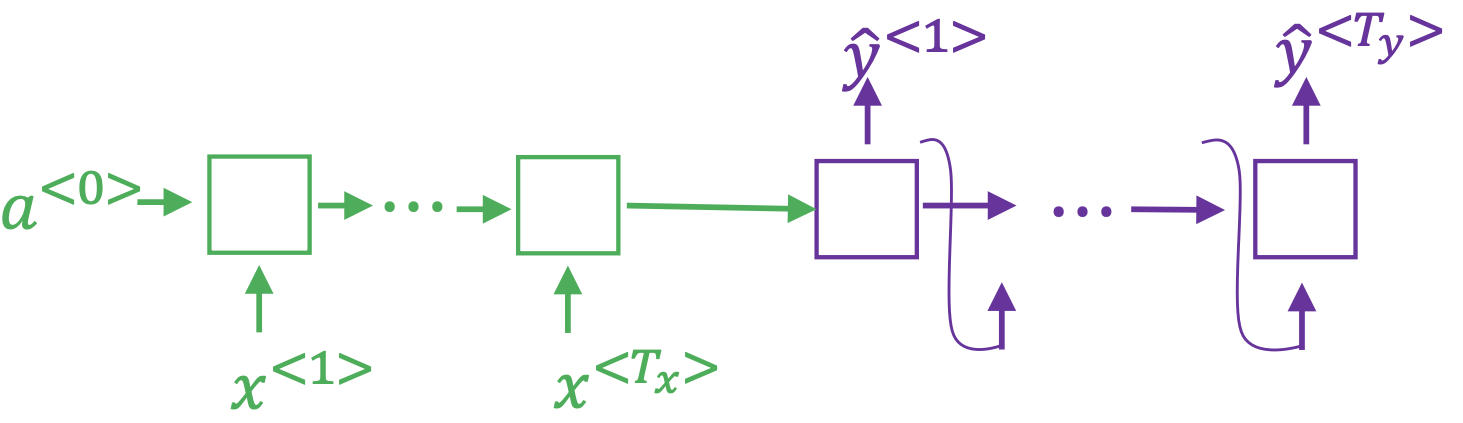
**1.Sequence to Sequence Model**

Original sentence = “Jane viste I’Afrique septembre”

English translation = “Jane is visiting Africa in September”



First we will train a network to input X and output y



After the inout sequence RNN outputs a vector that represents the entire input sequence

Then we build the decoder network(purple) that outputs the translation

This model is similar to-

**2.Image captioning**

For eg- given a image and we want it to be captioned as cat is on table.

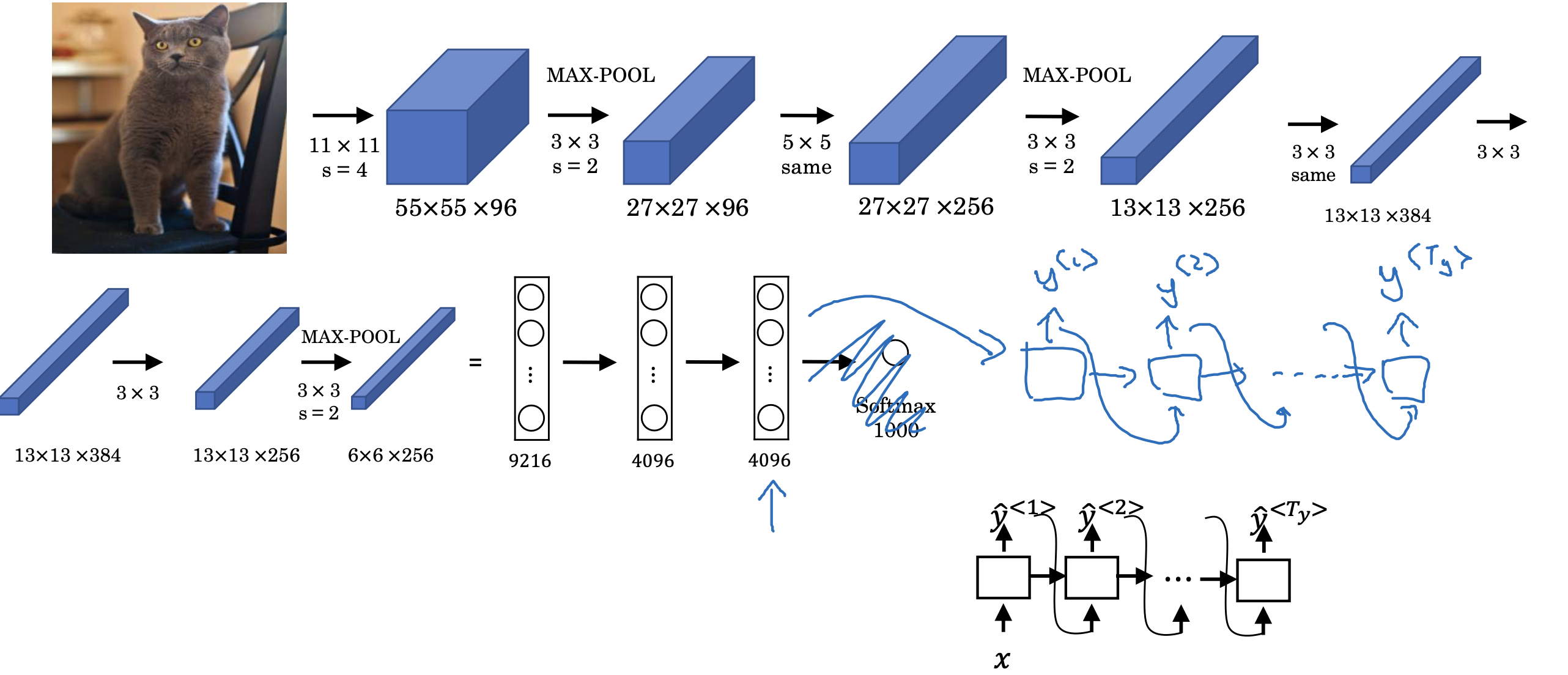
First we will use the Convnets to input the image to a known predefined conv net

And learn image encodings.

A pre-trained model like AlexNet gives a 4096 dimensional feature vector for the cat.

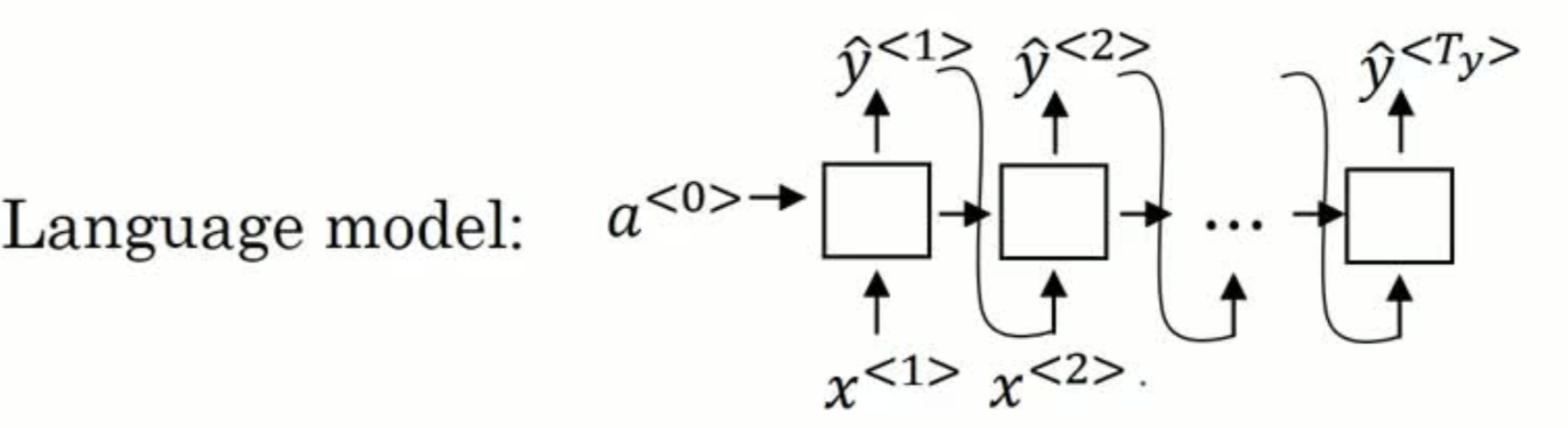
Then feed this vector to the RNN to generate the caption .

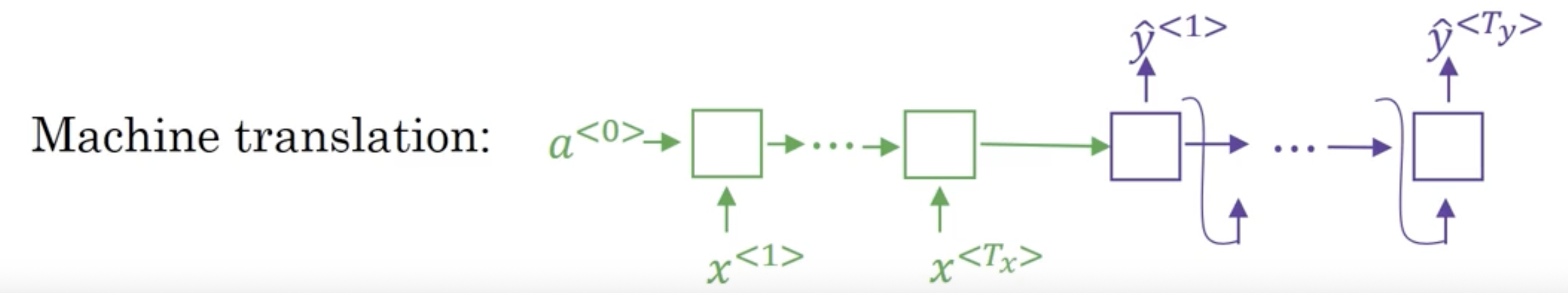
This model works for small captions.



**\*Picking the most likely sentence**

**Machine translation as building a conditional language model**





In language modelling - the model estimates the probability of sentence and generate sentence .

The x<1>,x<2>… etc are not important ,they could be vector of zeros and previous generated output .

In Machine translation-model has an encoder & decoder .

The translation model starts with encoded representation of input .

This is why , conditional language model (modelling the probability of output) , based on input.

To apply this model to translate the French into English , the model provides the various probability of generated English sentences as output .

As we don’t maximize the sample outputs at random we have to use an algorithm -

**Greedy search**

It picks the most likely first element and then 2nd so on..

For eg - lets consider 2 translation cases -

1)Jane is visiting Africa in September .

2)Jane is going to Africa in September.

Since going is more common word so greedy search results in optimal sentences and wont consider entire sequence.

And its impossible to operate all sentences so we need an approximate program to maximise the conditional probability.

**\*Beam Search**

Beam Search Basics:

Unlike Greedy Search, which picks the most likely word at each step, Beam Search considers multiple possibilities simultaneously.

B is the band width

B determines how many possibilities are considered.For ex-

B=3.

Procedure-

-First Word:

Evaluate the probability of each word in the vocabulary as the first word using an encoder-decoder model.

Keep the top

B words (e.g., "in", "Jane", "September").

-Second Word:

For each of the top

B first words, evaluate all possible second words.

Calculate the combined probability of the first and second words.

Keep the top

B combinations of the first and second words.

-Subsequent Words:

Repeat the process for subsequent words, each time considering the top

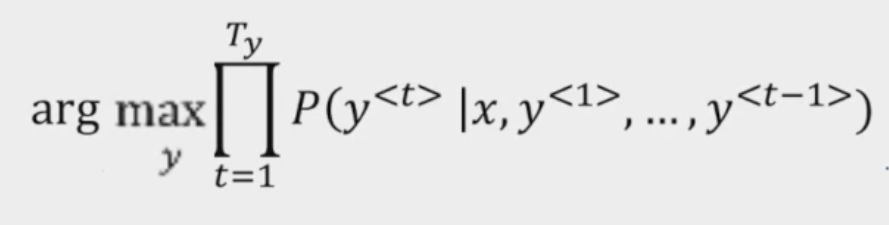
B sequences from the previous step and evaluating all possible next words.

Calculate and compare the combined probabilities of these sequences.

Continue this process until the end-of-sentence symbol is generated.

Greesy search is basically B=1

**\*Length Normalization**



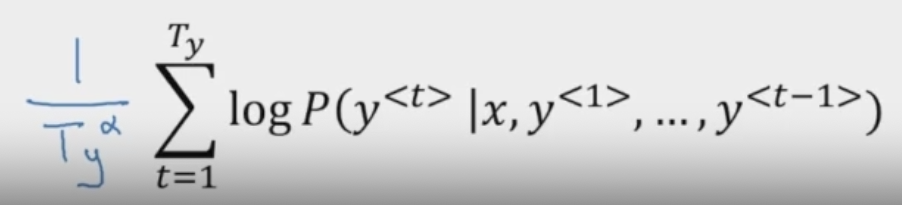
We have to maximize the probability.

But P(y<t>|x,……) is <1 since it is formed by multiplying a lot of small numbers so it causes floating representation rounding error.

So we’ll use logarithmic function as its monotonic and log P(y|x) will give same result as

P(y|x) as ame value of y maximizes both

Very short transition is preferred as in long sentences this log function will be very small.

Now lets normalize this-

This is average & reduces the penalty for outputting longer transitions.

Alpha is hyperparamter

Softer approach use - Ty(alpha)

alpha=1 —> completely normalized by length

alpha=0 —> no normalization

Beam width

Larger B = lot of possibilities (more options) —> better results

Small B = worse result (faster)

**\*Error Analysis**

Example, “Jane viste I’Afrique septembre”

Human, “Jane vists Africa in September”— y\*

Our dev set is human provided translation

Beam search, “Jane visited Africa last September”—y^

Our model has 2 components -

1.NN model-sequence to sequence model

2.beam search algorithm

Now we can make RNN compute the-

P(y\*|x) & P(y^|x) and see which is greater

CASE-1

P(y\*|x) > P(y^|x)

Beam at fault

CASE-2

P(y\*|x) <= P(y^|x)

RNN at fault

**\*Attention Model**

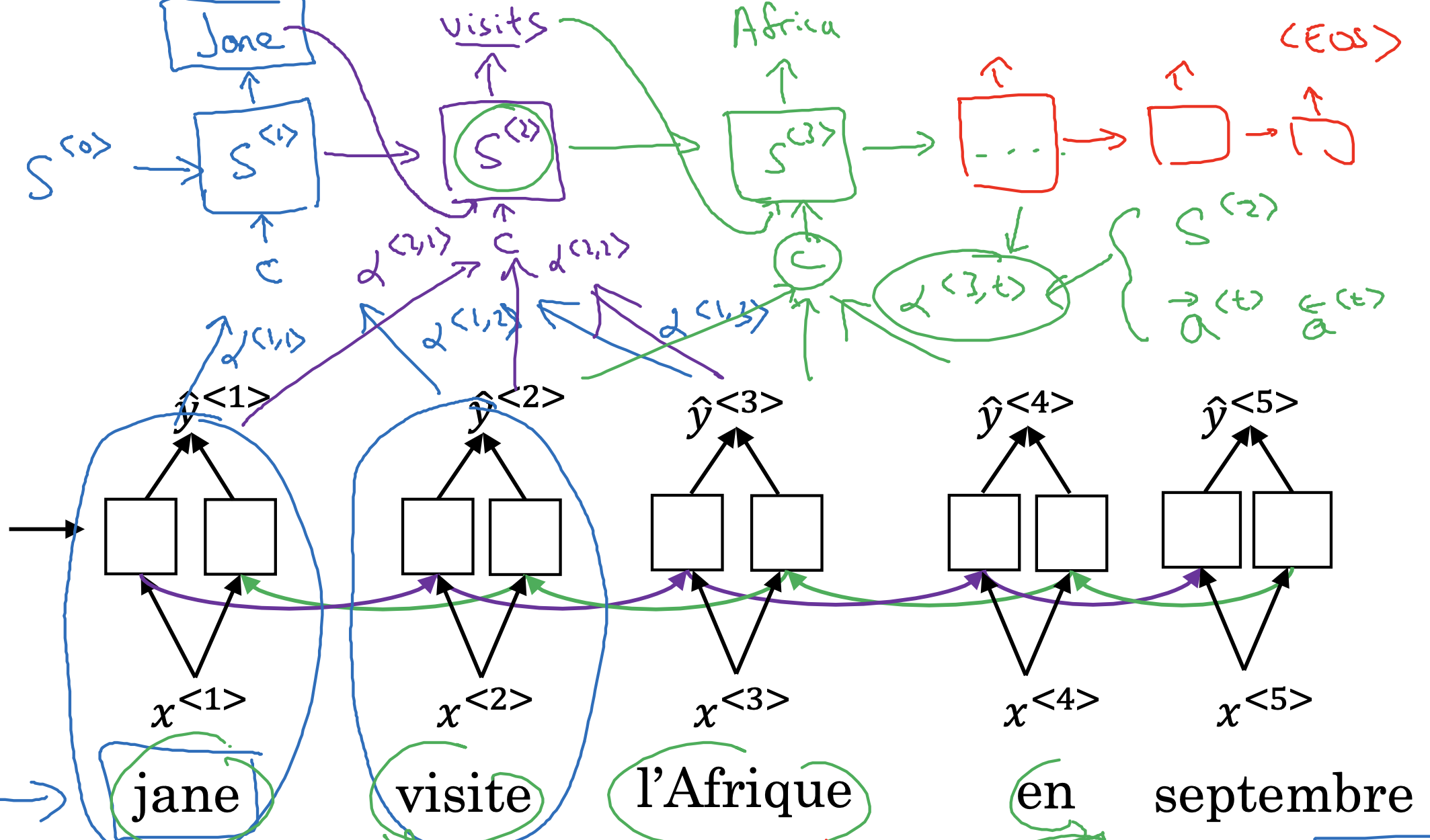
Given a very long sentence - we do have to read the whole sentence in one RNN & then memorise(activations) and then decode it to generate the English translation.

But,

Human translation will not read the French sentence as a whole instead it will read the parts of it an translate them.

Above encoder & decoder works well for short sentences (high blew score) & for longer one it goes down.

So we use attention model

Diagram from Ng’s notes is given below-

In above example -

Short sentence -Jane viste I’Afrique septembre

Bidirectional RNN - to capture features

To generate the first word (Jane ) we will be looking for it in the French sentence & near words , not looking at the end of the sentence.

So, attention model will be computing set of attention weighs & we use α<1,1> to denote we are generating the first word .

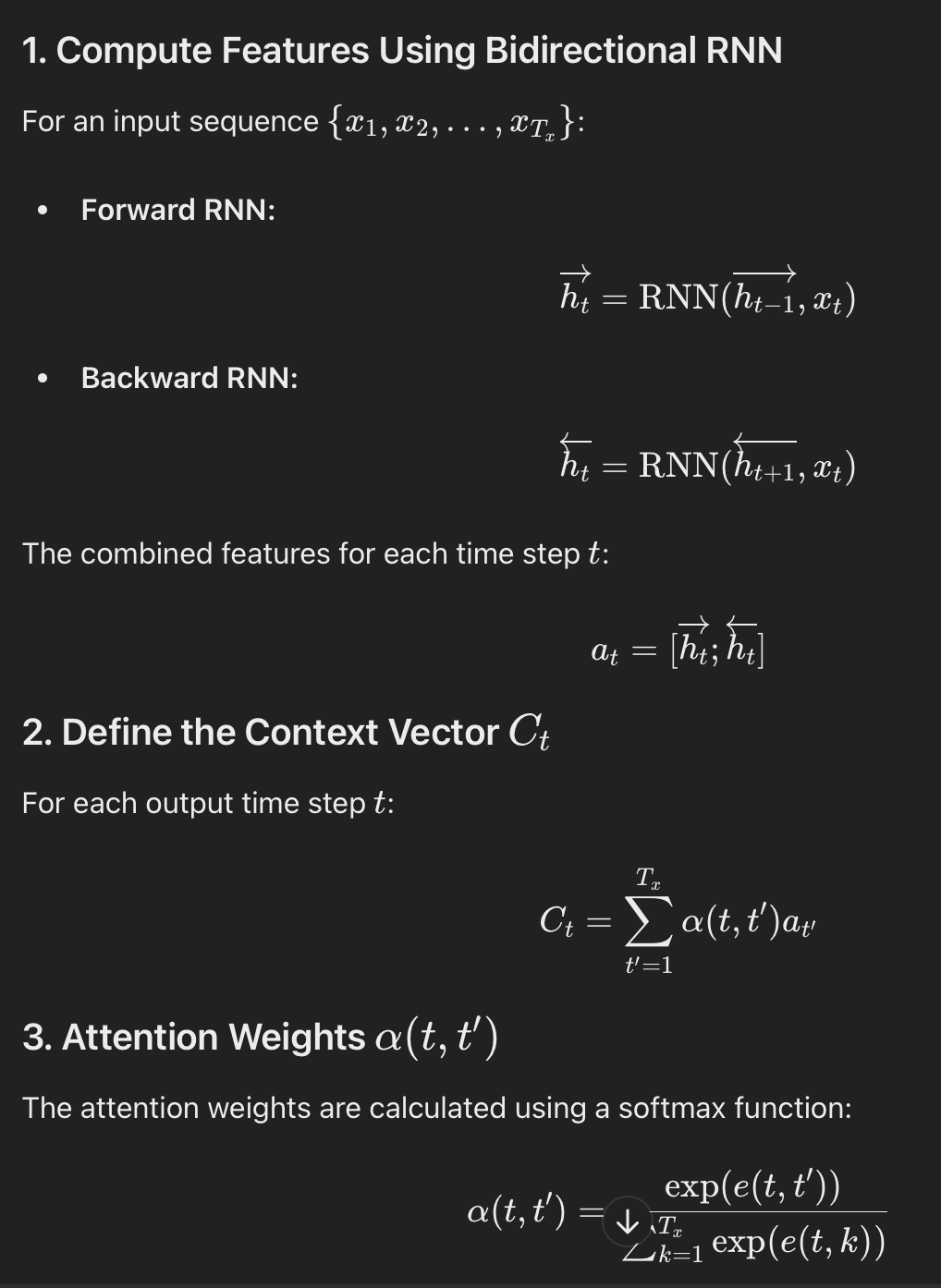
Then we’ll α<1,2> which is second attention weight and so on

Together all above attention weight will give C(context)

C is the input to the second unidirectional RNN network

Then we generate new sets of attention wrights to get second word and so on…

Here , attention model allows only a part of an inout sentence while it generating a translation.



**\*Speech recognition**

Speed recognition problem -

Audio clip -X

Output transcript -y

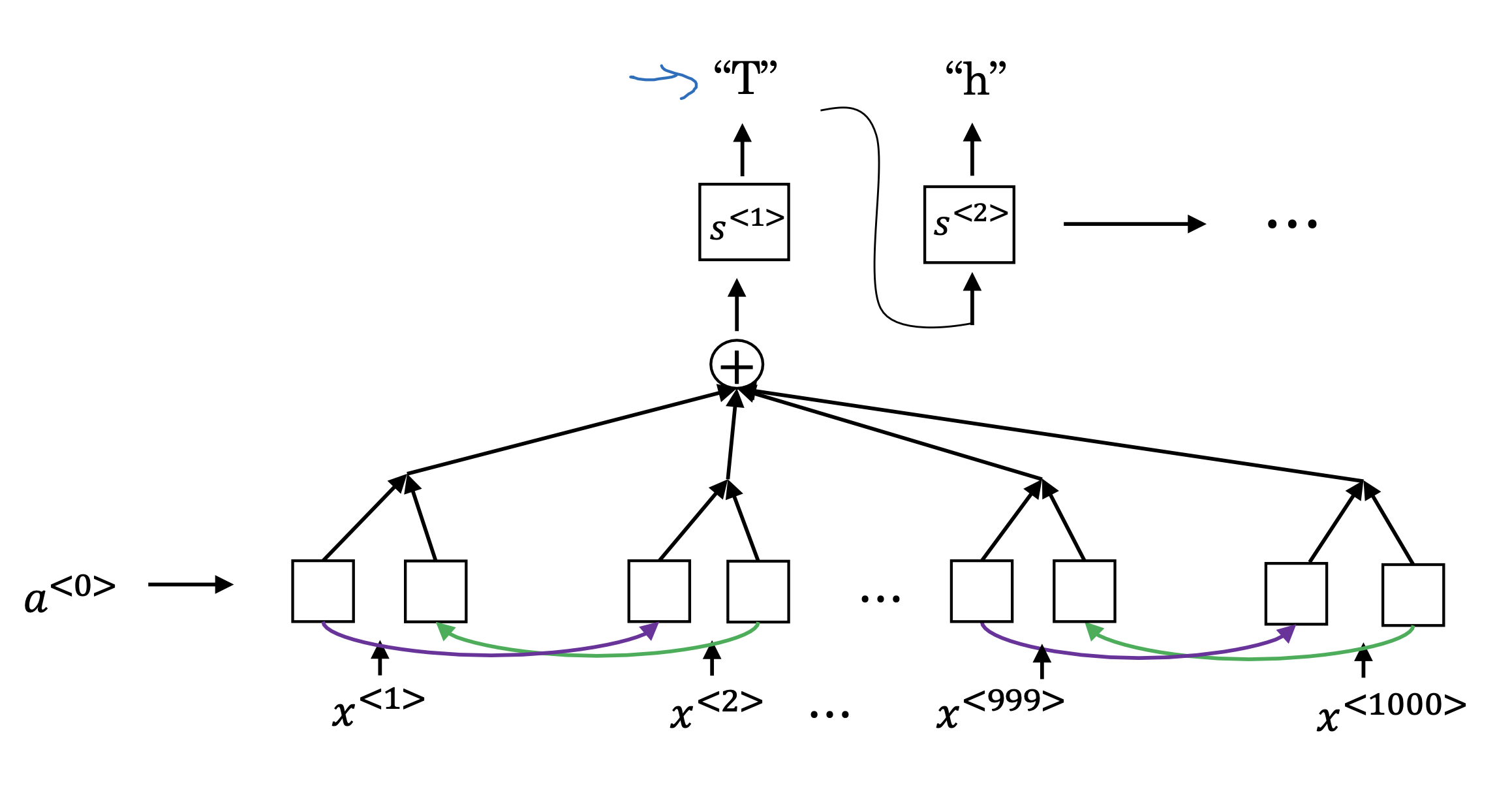
Job is to find transcript y from X

Basically we plot air pressure vs time graph

Or run audio clip on spectrogram

Attention model for speech recognition

In horizontal axis take the different time frames of audio input & then add attention model and try to find the output transcript .



**CTC( connectionist temporal classification ) cost for speech recognition**

Audio clip - “ the quick brown fox”

Like we have NN with same number of x’s & y’s

Usually in speech recognition no. of inputs > outputs

For ex- 10s audio clip would have 1000 inputs but output might not have 1000 alphabets

So, CTC cost function allows the RNN to generate an output like ttt\_\_h\_eee\_\_\_ \_\_\_qqq\_

There is a special charter called blank character denote by underscore

The basic rule of CTC cost function is to collapse the repeated elements .

So above output will change to——- the

This allows the network to have 1000 outputs by repeating characters .

**\*Trigger word detection**

Ex- google home ,Siri , Alexa etc

Using trigger word detection we can trigger all of them to do our tasks .

How it works ?

Take the audio commute it into spectrogram features that generate features x<1>,x<2>…

Then pass it to RNN and define target labels y.

As soon as someone finished saying hey Siri labels set to 1 by trains sets.

The problem here is it crates unbalanced trains set with more than 0’s then 1.

Solution is instead of setting a single time step output 1 we can make it output few 1’s for a period.

**Week-4**

**TRANSFORMERS**

As the complexity of sequence increases the model becomes more complex .

In RNN we had vanishing gradient problem which made hard to capture long range sequences so we moved to more complex models like GRU & LSTM

Transformers help to run a lot more computations for the entire sequence in parallel .

Motivation for transformer is -

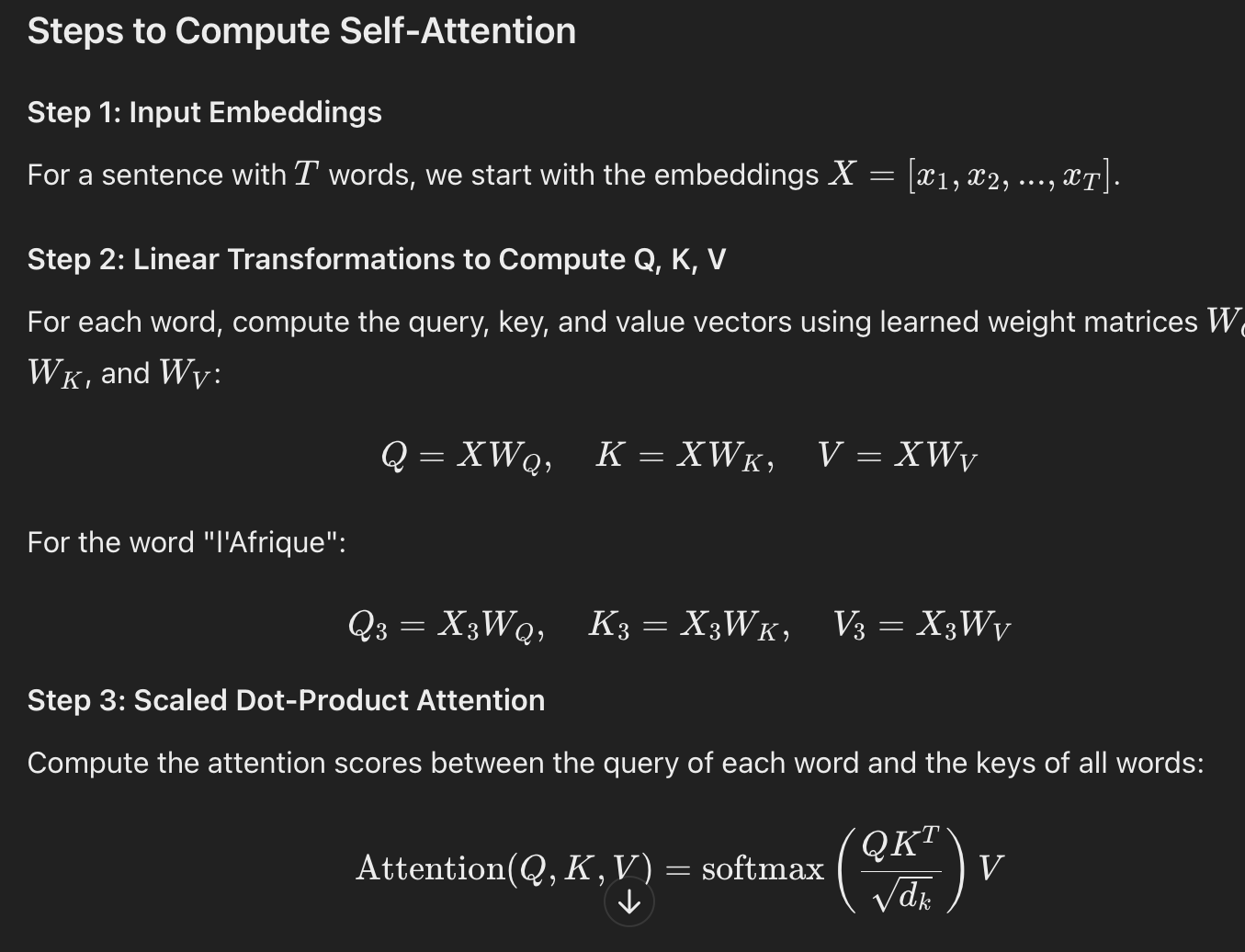
Attention model + CNN

Self attention

Multi head attention

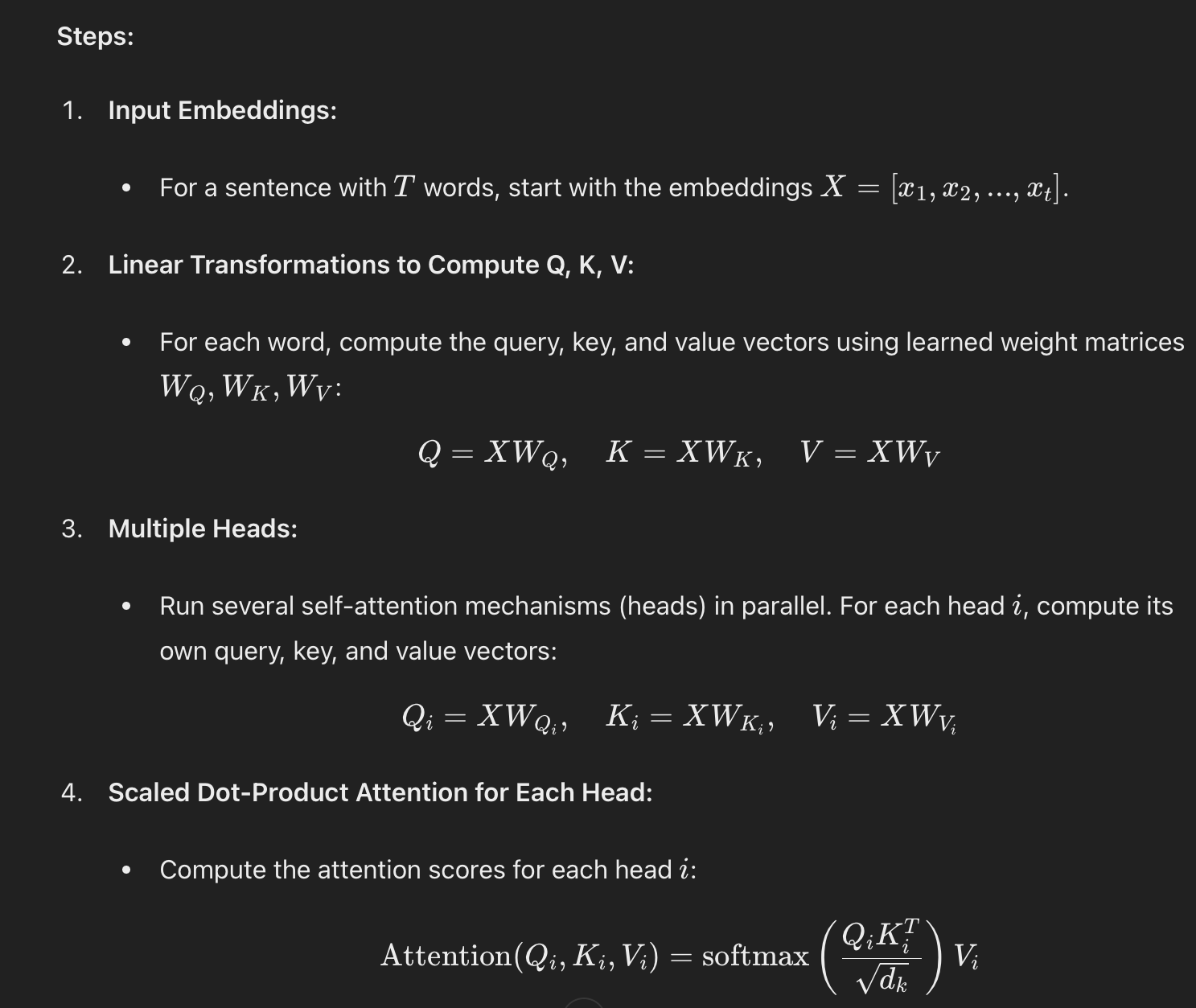
**\*Self attention**

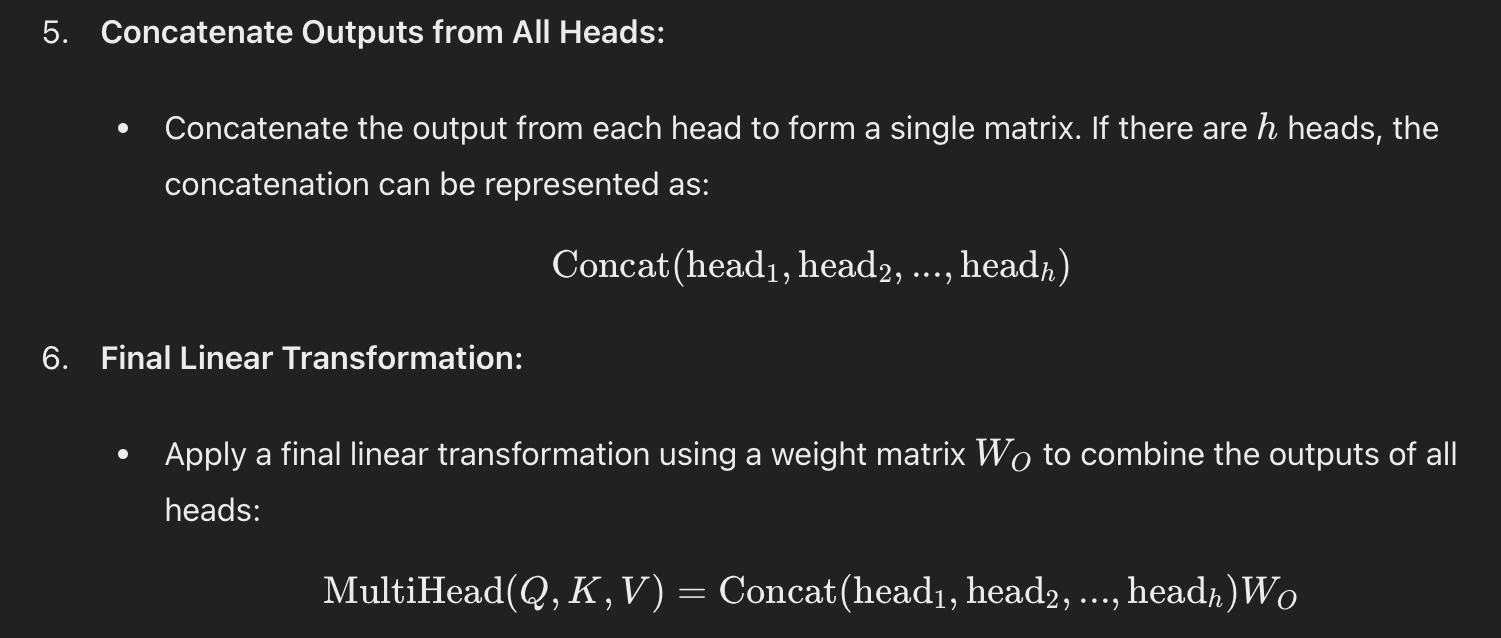
Self-attention allows each word in a sentence to be represented based on the context provided by all other words in the sentence.



**\*Multi head attention**

It is mainly set attention with a for loop to get multiple versions of above representations.





**\*Transformer Network**

Transformer network is given from Ng’s course

