AI CLUB
IIT Madras

An intro to RNNS

AND OTHER RELATED TOPICS

Table of CONTENTS

RNN

Recurrent Neural Networks or RNNs are a class of neural networks that allow previous outputs to be used as inputs while having hidden states.

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations.

Why RNNs??

HOW IS IT BETTER?

HOW DOES A REGULAR NEURAL NETWORK WORK?

CONSIDER THE FOLLOWING SENTENCES

- YOU DON'T HAVE TO BE WELL TO TRAVEL RICH
- IF I'M BAD , YOU'RE MY DAD
- POLICY IS THE BEST HONESTY
- HEALTH IS INJURIOUS TO SMOKING
- IF OPPORTUNITY DOESN'T DOOR THEN BUILD A KNOCK

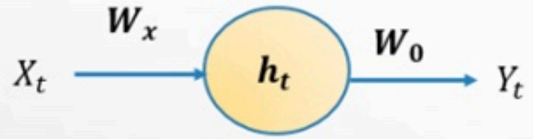
Why RNNs??

HOW IS IT BETTER?

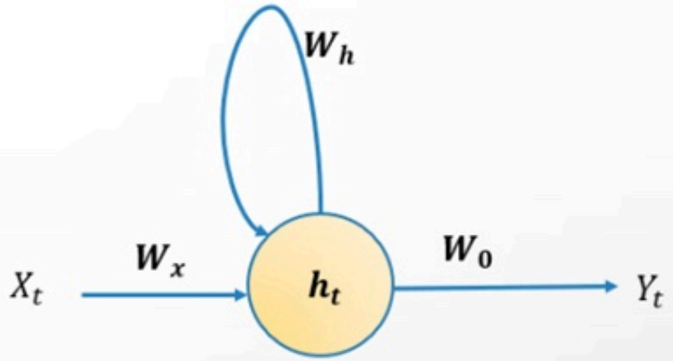
- SO YOU SEE A LITTLE JUMBLE IN THE WORDS MADE THE SENTENCE INCOHERENT .

- SO IF WE'RE TRYING TO USE SUCH DATA TO PREDICT ANY REASONABLE OUTPUT, WE NEED A NETWORK ,WHICH HAS ACCESS TO SOME PRIOR KNOWLEDGE ABOUT THE DATA TO COMPLETELY UNDERSTAND IT. THAT'S WHERE RECURRENT NEURAL NETWORKS COME TO RESCUE.

RNN WITH WEIGHTS



Regular Feedforward Network



Simple Recurrent unit

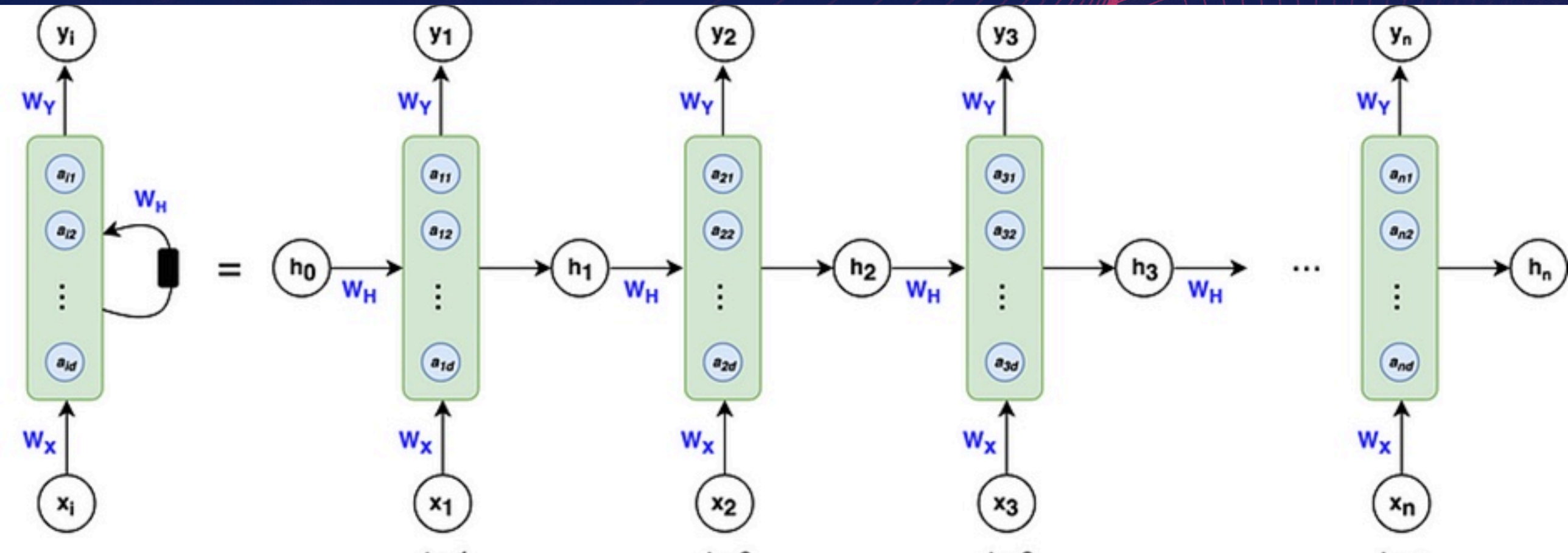
$$h_t = f\left(W_h^T h_{t-1} + W_x^T X_t + b_h \right)$$

$$Y_t = softmax\left(W_0^T h_t + b_0 \right)$$

(f=ReLU,tanh,sigmoid)

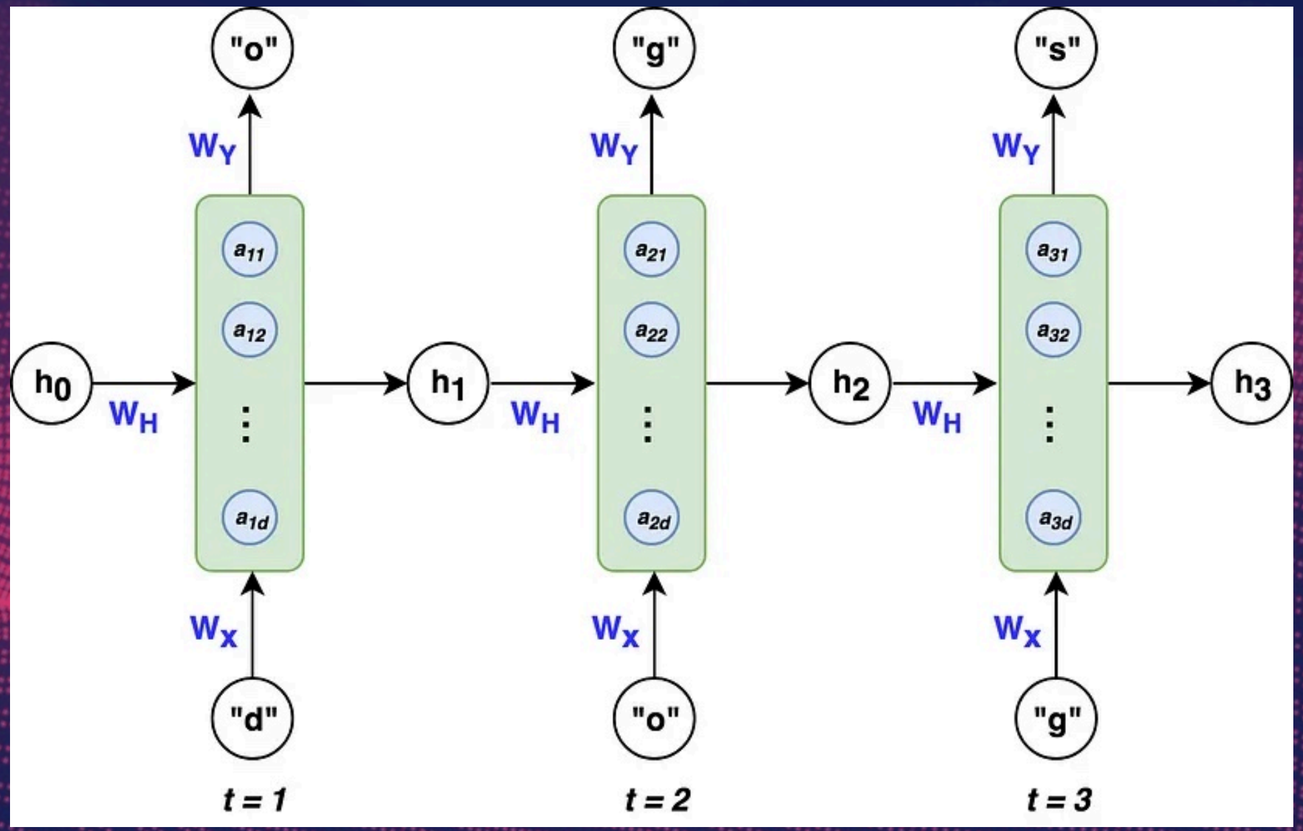
HERE THE HIDDEN STATE H_t TAKES INTO ACCOUNT BOTH THE INPUT SEQUENCE X_t AND THE PREVIOUS HIDDEN STATE H_t-1,AND GIVES THE CORRESPONDING OUTPUT Y_t.

ARCHITECTURE OF RNNS



AN EXAMPLE

TAKE THE WORD “DOGS,” WHERE WE WANT TO TRAIN AN RNN TO PREDICT THE LETTER “S” GIVEN THE LETTERS “D”-“O”-“G”. THE ARCHITECTURE ABOVE WOULD LOOK LIKE THE FOLLOWING:



HERE THE OUTPUTS ARE JUST THE IDEAL OUTPUTS WE DESIRE, FOR T=1 AND T=2 THE OUTPUTS CAN BE ANYTHING

FORWARD PROPAGATION

$$a_1 = \begin{pmatrix} W_{H,11} & W_{H12} & W_{H,13} \\ W_{H,21} & W_{H,22} & W_{H,23} \\ W_{H,31} & W_{H,32} & W_{H,33} \end{pmatrix} \begin{pmatrix} h_{0,1} \\ h_{0,2} \\ h_{0,3} \end{pmatrix} + \begin{pmatrix} W_{X,11} & W_{X,12} & W_{X,13} & W_{X,14} \\ W_{X,21} & W_{X,22} & W_{X,23} & W_{X,24} \\ W_{X,31} & W_{X,32} & W_{X,33} & W_{X,34} \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{1,4} \end{pmatrix} = \begin{pmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \end{pmatrix}$$

$$h_1 = tanh(\begin{pmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \end{pmatrix}) = \begin{pmatrix} h_{1,1} \\ h_{1,2} \\ h_{1,3} \end{pmatrix}$$

$$y_1 = softmax(\begin{pmatrix} W_{Y,11} & W_{Y,12} & W_{Y,13} \\ W_{Y,21} & W_{Y,22} & W_{Y,23} \\ W_{Y,31} & W_{Y,32} & W_{Y,33} \\ W_{Y,41} & W_{Y,42} & W_{Y,43} \end{pmatrix} \begin{pmatrix} h_{1,1} \\ h_{1,2} \\ h_{1,3} \end{pmatrix}) = \begin{pmatrix} y_{1,1} \\ y_{1,2} \\ y_{1,3} \\ y_{1,4} \end{pmatrix}$$

BACKWARD PROPAGATION

Function dependencies with respect to W_Y

$L_t = -y_t log(\hat{y}_t) \rightarrow \hat{y}_t = softmax(z_t) \rightarrow z_t = W_Y h_t$ **(MULTI CLASS CROSS ENTROPY LOSS)**

Chain rule with respect to W_Y

$$\frac{\partial L_t}{\partial W_Y} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial W_Y}$$

Gradient for W_Y **(AND SIMILARLY FOR WX AND WH AS WELL)**

$$\frac{\partial L_{total}}{\partial W_Y} = \frac{\partial L_1}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial z_1} \frac{\partial z_1}{\partial W_Y} + \frac{\partial L_2}{\partial \hat{y}_2} \frac{\partial \hat{y}_2}{\partial z_2} \frac{\partial z_2}{\partial W_Y} + \frac{\partial L_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial z_3} \frac{\partial z_3}{\partial W_Y} = \sum_{t=1}^{n} \frac{\partial L_t}{\partial W_Y}$$

EVERYTHING SEEMS FINE...

SO WHAT'S THE ISSUE?

HINT: THE ACTIVATION FUNCTION SIGMOID AND TANH HAVE THE SAME ISSUE

VANISHING GRADIENT ISSUE

The brown and black dog, which was playing with the cat, was a german shepherd.

(x_2) (x_4) (x_5) (x_{14}) (x_{15})

BROWN AND BLACK ARE FAR AWAY FROM THE END OF THE SENTENCE, SO THE MODEL WILL HAVE TROUBLE PREDICTING GERMAN AND SHEPHERD

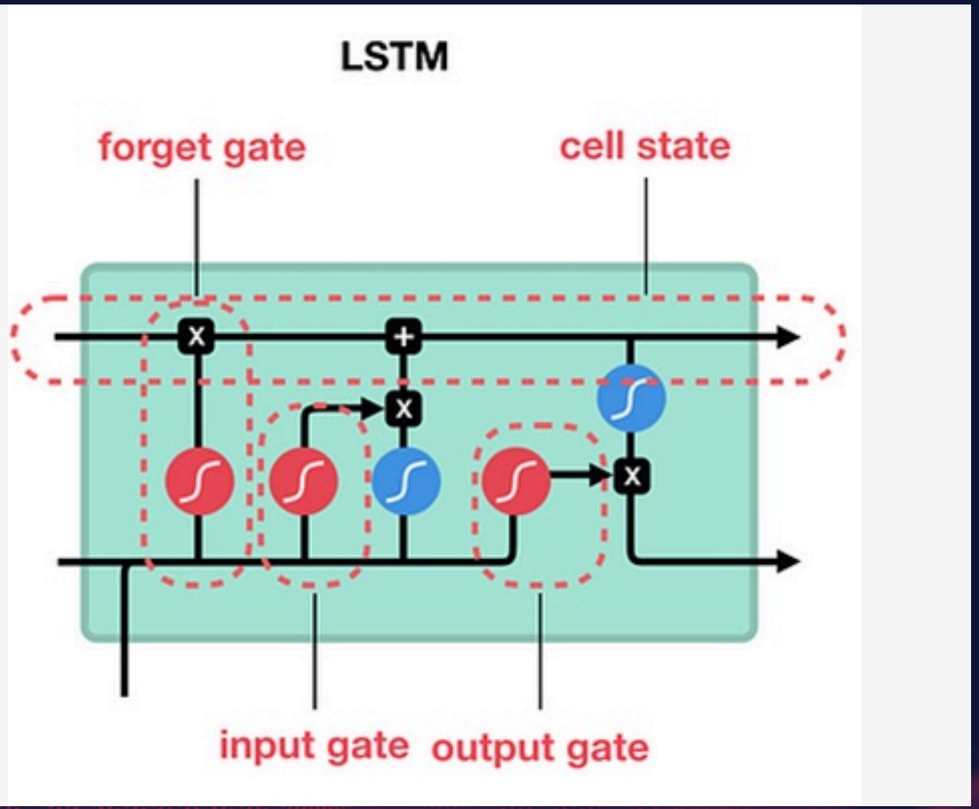
$$\frac{\partial L_{15}}{\partial \hat{y}_{15}} \frac{\partial \hat{y}_{15}}{\partial z_{15}} \frac{\partial z_{15}}{\partial h_{15}} \frac{\partial h_{15}}{\partial h_2} \frac{\partial h_2}{\partial W_x}$$

$$\frac{\partial h_{15}}{\partial h_2} = \frac{\partial h_{15}}{\partial h_{14}} \frac{\partial h_{14}}{\partial h_{13}} \cdots \frac{\partial h_2}{\partial h_1}$$

THESE CHAINS OF GRADIENTS ARE TROUBLESOME BECAUSE IF LESS THAN 1 THEY CAN CAUSE THE LOSS FROM THE WORD SHEPHERD WITH RESPECT TO THE WORD BROWN TO APPROACH 0,

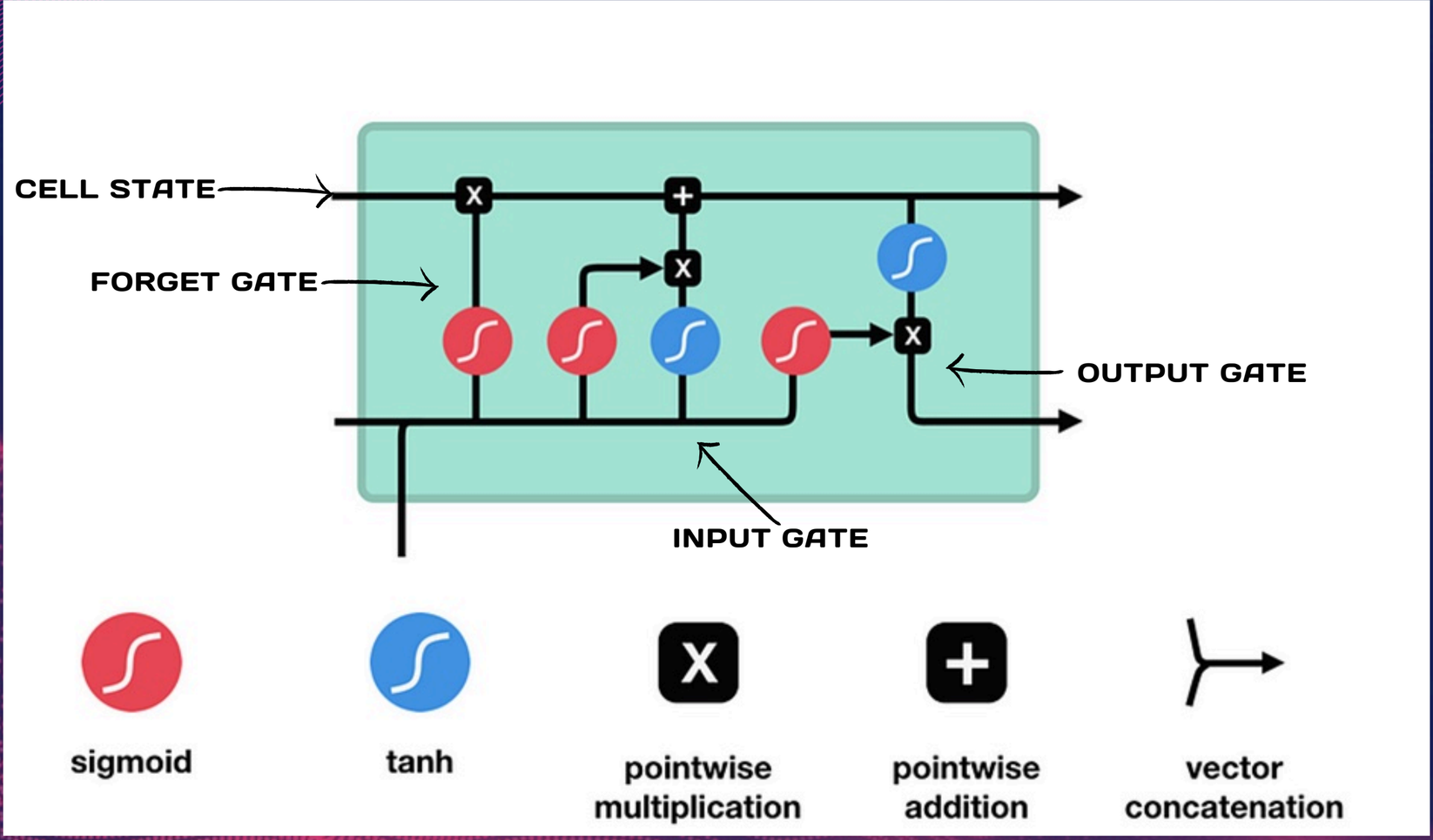
THE SOLUTION:

LSTMS



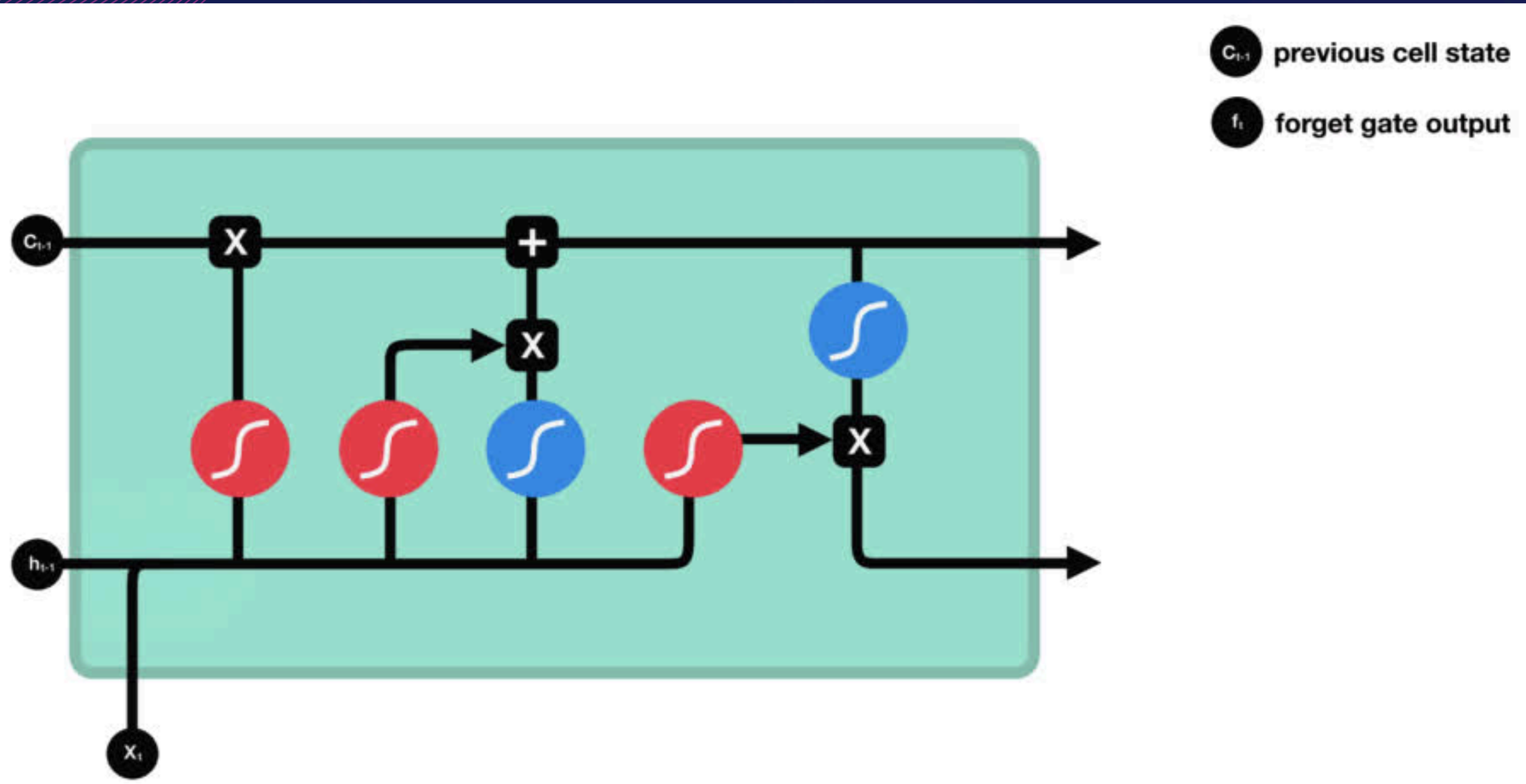
LSTMS

THE CORE CONCEPT OF LSTM'S ARE THE CELL STATE, AND IT'S VARIOUS GATES; THE FORGET GATE, INPUT GATE AND OUTPUT GATE. THE CELL STATE ACTS AS THE HIGHWAY BETWEEN THESE GATES FOR TRANSPORT OF NFORMATION ALONG THE LSTM UNIT.



FORGET GATE

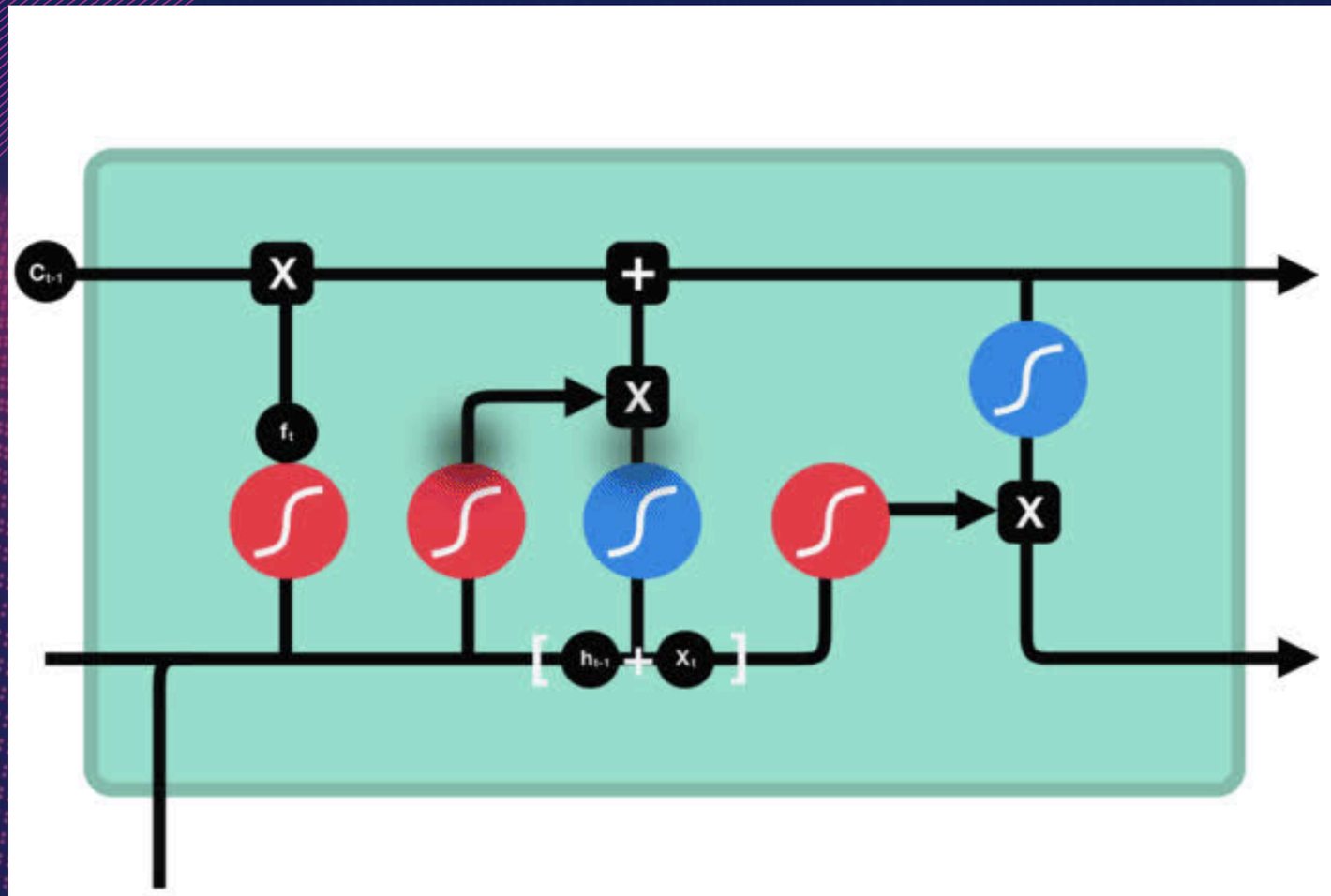
FIRST, WE HAVE THE FORGET GATE. THIS GATE DECIDES WHAT INFORMATION SHOULD BE THROWN AWAY OR KEPT. CONSISTS OF A SIGMOID FUNCTION WHICH ASSIGNS VALUES NEAR 0 TO INFORMATION THAT NEEDS TO BE FORGOTTEN AND VALUES NEAR 1 FOR INFORMATION TO BE REMEMBERED.



INPUT GATE

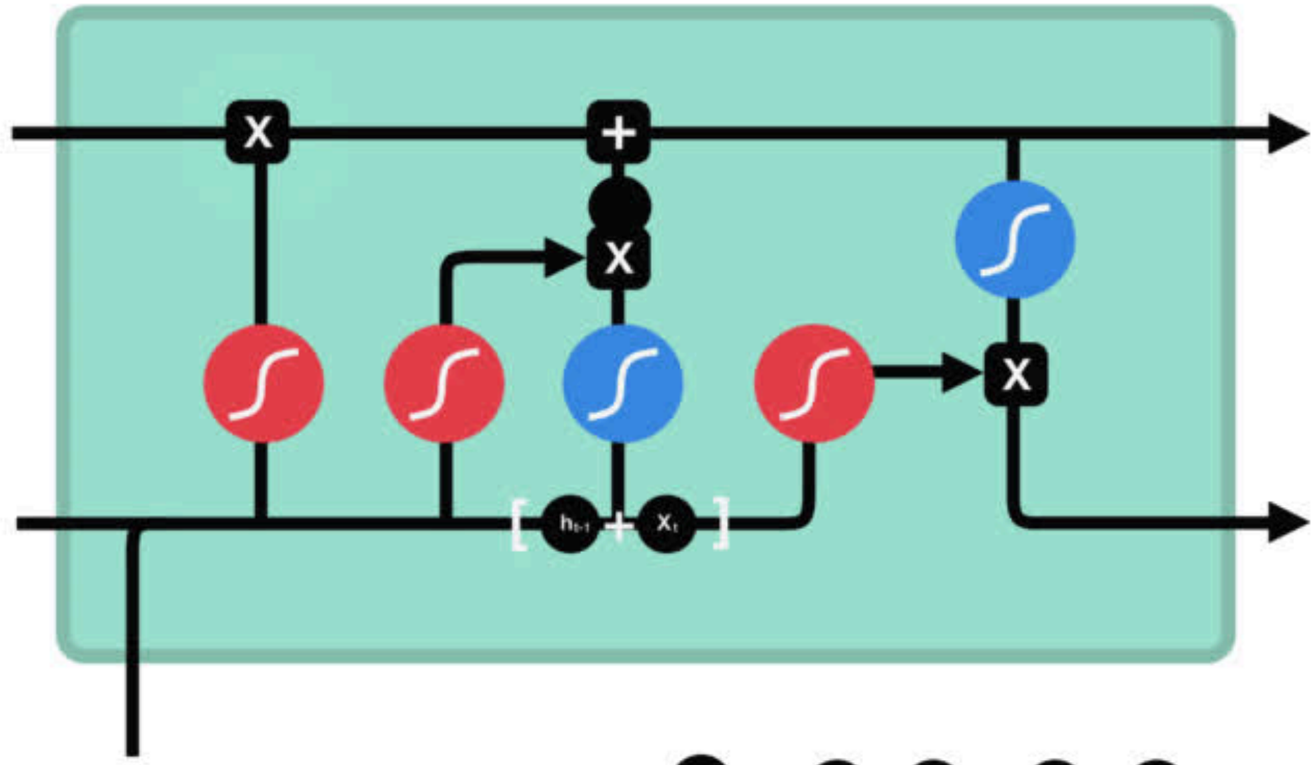
FIRST, WE PASS THE PREVIOUS HIDDEN STATE AND CURRENT INPUT INTO A SIGMOID FUNCTION. THAT DECIDES WHICH VALUES WILL BE UPDATED BY TRANSFORMING THE VALUES TO BE BETWEEN 0 AND 1. 0 MEANS NOT IMPORTANT, AND 1 MEANS IMPORTANT.

THE SAME HIDDEN STATE AND INPUT IS ALSO SENT TO A TANH FUNCTION TO DETERMINE WHETHER DATA IS ADDED OR EXTRACTED. THEN THE OUTPUT FROM BOTH THE FUNCTIONS ARE POINTWISE MULTIPLIED AND SENT TO THE CELL STATE.



CELL STATE

FIRST, THE CELL STATE GETS POINTWISE MULTIPLIED WITH THE FORGET VECTOR.
NEXT IT IS POINTWISE ADDED TO THE FINAL INPUT GATE CONTENT AND THEN SENT FORWARD.



C_{t-1} previous cell state

f_t forget gate output

i_t input gate output

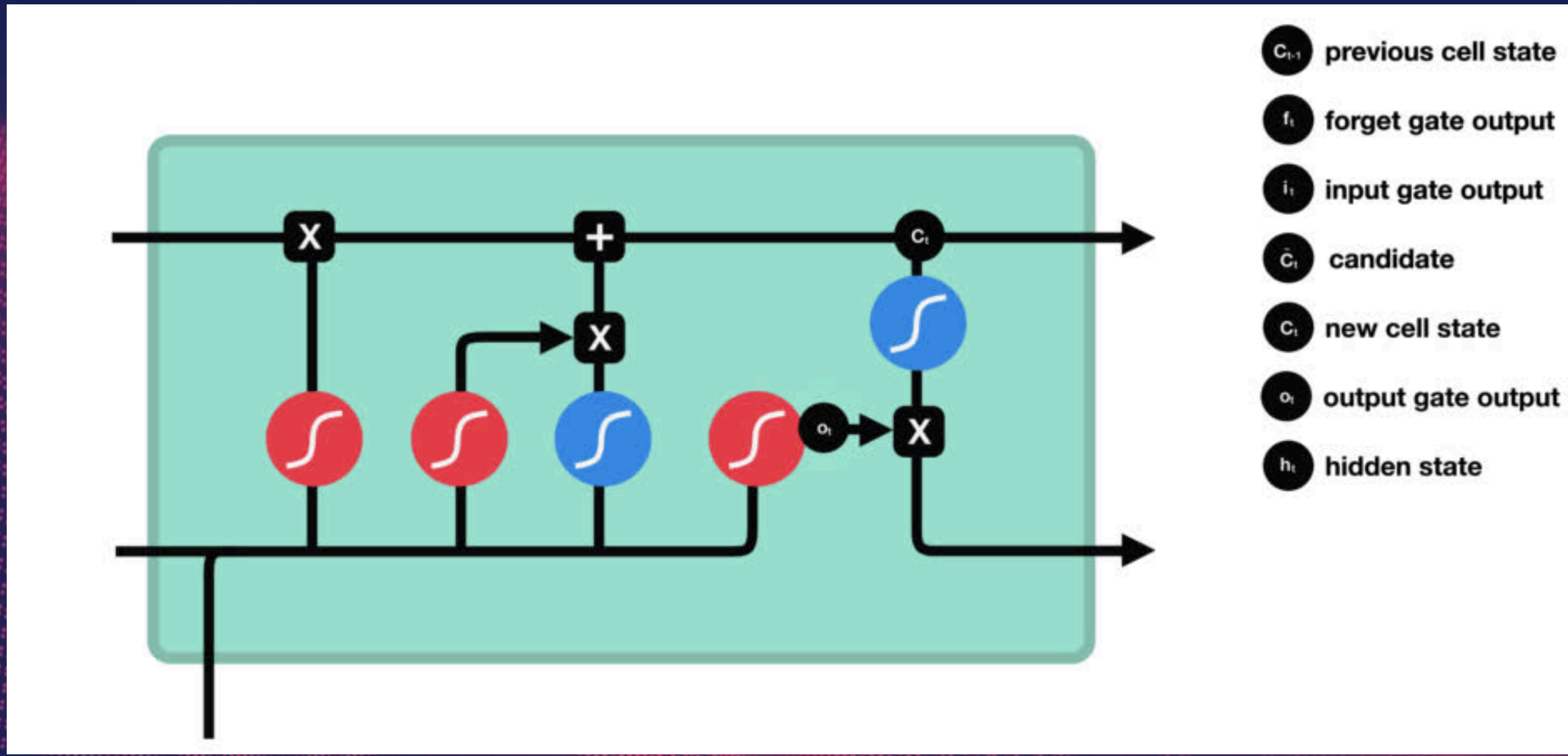
$\tilde{C}_t$ candidate

C_t new cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

OUTPUT GATE

THE OUTPUT GATE DECIDES WHAT THE NEXT HIDDEN STATE SHOULD BE. FIRST, WE PASS THE PREVIOUS HIDDEN STATE AND THE CURRENT INPUT INTO A SIGMOID FUNCTION.
THE NEWLY MODIFIED CELL STATE IS PASSED TO THE TANH FUNCTION. WHICH IS THEN MULTIPLIED WITH THE SIGMOID OUTPUT TO DECIDE WHAT INFORMATION THE HIDDEN STATE SHOULD CARRY. THE NEW CELL STATE AND THE NEW HIDDEN IS THEN CARRIED OVER TO THE NEXT TIME STEP.

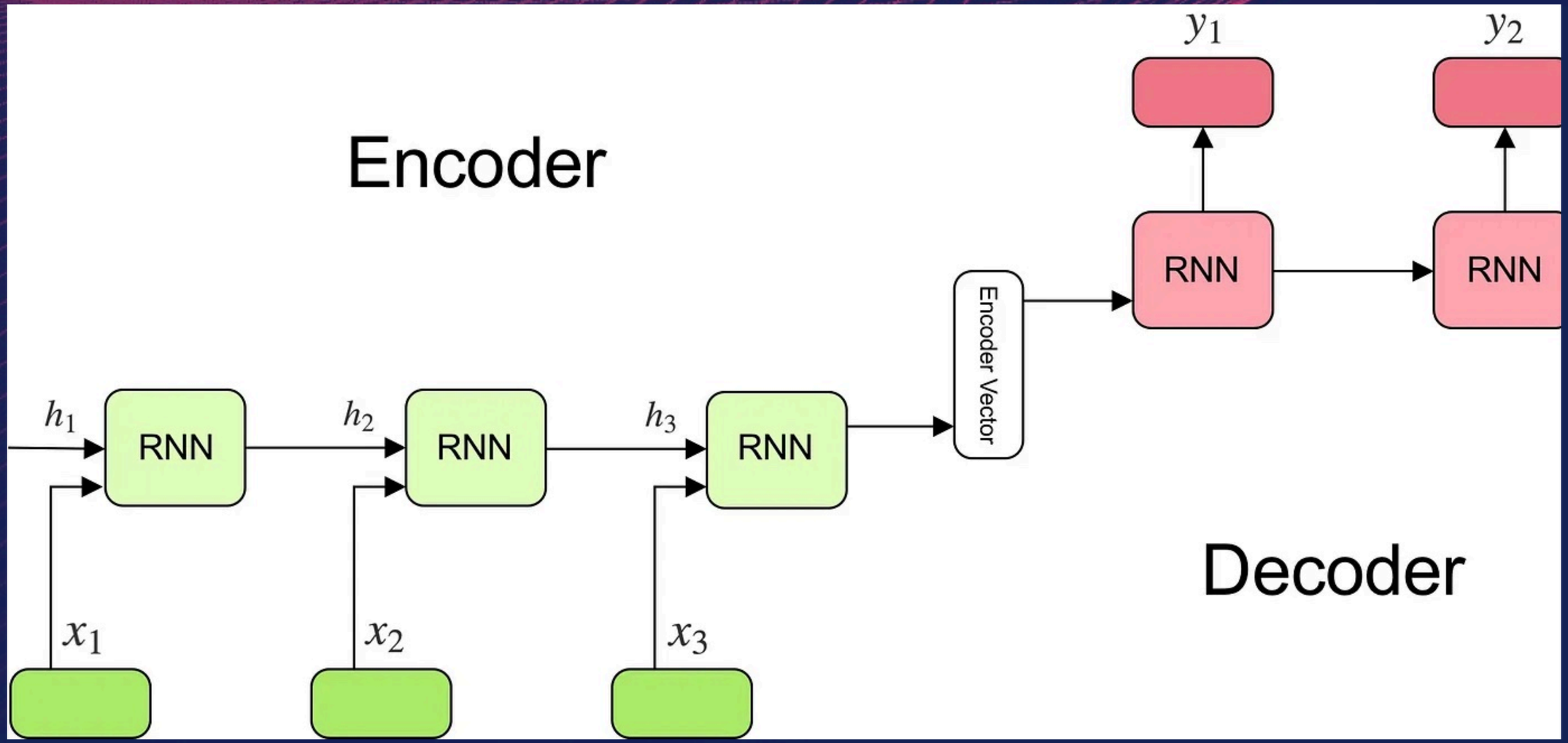


ARE EVEN THESE INADEQUATE?

WHAT IF THE OUTPUT AND INPUT VARY IN LENGTH? IF I WANT TO TRANSLATE HELLO TO CHINESE, IT WON'T NECESSARILY BE OF THE SAME LENGTH.

THAT'S WHERE SEQUENCE TO SEQUENCE MODELS STEP IN THAT TAKE INPUTS OF ANY SIZE, COMPRESS THEN INTO A SINGLE VECTOR THAT IS PROCESSSED TO GIVE OUTPUTS OF DIFFERENT LENGTHS.

SEQUENCE TO SEQUENCE MODELS



PARTS OF A SEQUENCE TO SEQUENCE MODEL

ENCODER

AN RNN MODEL WHICH ACCEPTS A SINGLE ELEMENT OF THE INPUT SEQUENCE, COLLECTS INFORMATION FOR THAT ELEMENT AND PROPAGATES IT FORWARD. GIVES RISE TO THE FINAL ENCODED VECTOR.

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

ENCODED VECTOR

THIS IS THE FINAL HIDDEN STATE PRODUCED FROM THE ENCODER PART OF THE MODEL.
IT ACTS AS THE INITIAL HIDDEN STATE OF THE DECODER PART OF THE MODEL.

DECODER

AN RNN MODEL WHICH PREDICTS AN OUTPUT Y_T AT A TIME STEP T. EACH TIME STEP ACCEPTS A HIDDEN STATE FROM THE PREVIOUS TIME STEP AND PRODUCES AN OUTPUT AS WELL AS ITS OWN HIDDEN STATE. THE OUTPUT Y_T AT TIME STEP T IS COMPUTED USING THE FORMULA:

$$y_t = softmax(W^S h_t)$$


THANK
YOU

ATTENDANCE QR CODE