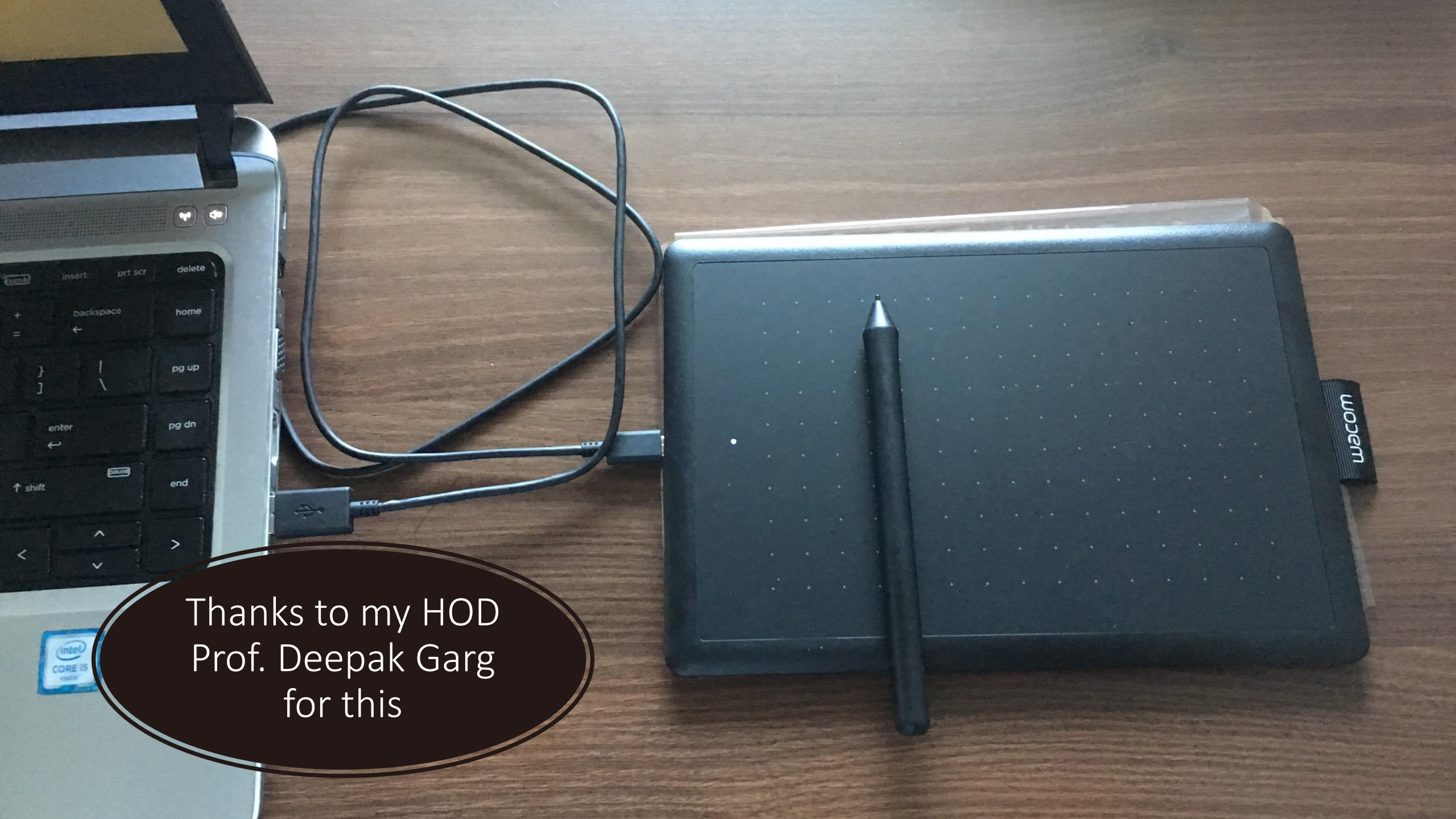
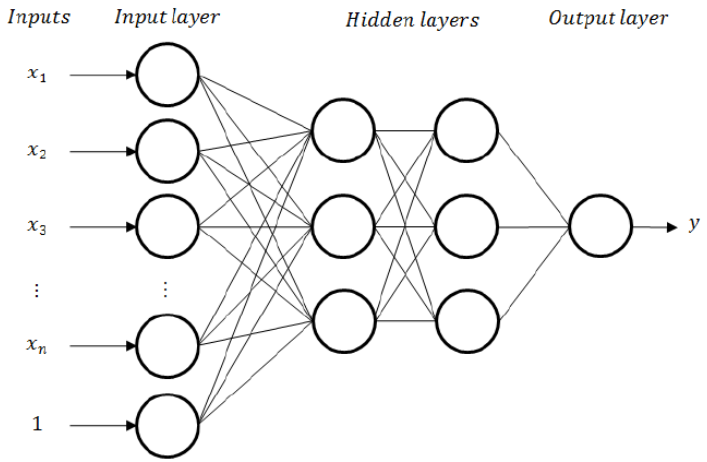
# Recurrent Neural Networks and Applications


insert
prt scr
delete
backspace
home
pg up
enter
pg dn
shift
end
wacom
Thanks to my HOD
Prof. Deepak Garg
for this

# Sentiment Analysis

- Let us try to classify following text as positive or negative
  - I like this phone – Positive
  - This phone is good – Positive
  - This phone is not okay – Negative
  - I do not like this phone because battery is not charging properly - Negative




Feed forward networks accept a fixed-sized vector as input !

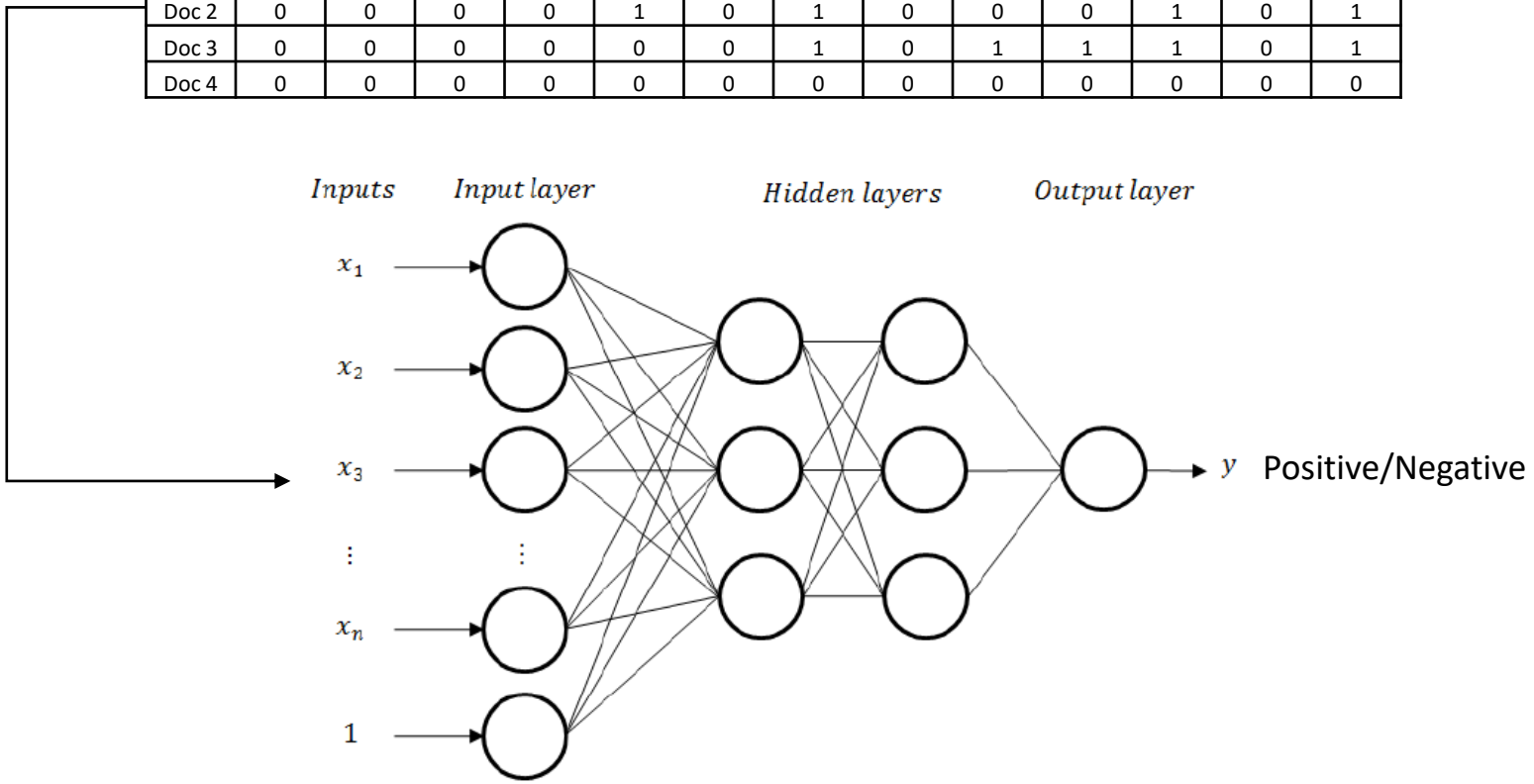
# Solution 1: Using Bag-of-Words

- Represent the text using Bag-of-Words
  - I like this phone
  - This phone is good
  - This phone is not okay
  - I do not like this phone because battery is not good

| | battery | because | charging | do | good | I | is | like | not | okay | phone | properly | this |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Doc 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| Doc 2 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Doc 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| Doc 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Applying ANN

| | battery | because | charging | do | good | I | is | like | not | okay | phone | properly | this |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Doc 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| Doc 2 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Doc 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| Doc 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



# Drawback of BoW

- Lets try to represent the following text using Bag-of-Words
  - This phone is no good - Negative
  - No this phone is good - Positive

| | good | is | no | phone | this |
|---|---|---|---|---|---|
| Doc 1 | 1 | 1 | 1 | 1 | 1 |
| Doc 2 | 1 | 1 | 1 | 1 | 1 |

- Feed Forward Neural Networks with Bag-of-words (BoW) model does not consider position of words in input !

# Word guessing game

- Guess the missing word in the follow sentence
  - I went to France last year, there the people speak the _____ language
- Lets find the missing word with the sentence (alphabetically sorted words)
  - Sorted Text: food I is it like Thai so
  - Original Text: I like Thai food it is so _____
  - Answer: I like Thai food it is so _____
- Feed Forward Neural Network will fail for guessing missed words
- Conclusion: Sequence matters !

# Other situations where sequence matters

- Stock price today will be more or less similar to yesterday's price
- Tomorrow's temperature will be close to todays' temperature

# Sequence Application Variation

- Audio Signal to Sequence - Speech Recognition
- Nothing to Sequence or Single Parameter to Sequence - Music Generation
- Sequence to Single Output - Sentiment Classification
- Sequence to Sequence - Machine Translation
- Video Frame Sequence to Output - Activity Recognition
- Sub-Sequence from a Sequence - Finding Specific Protein from a DNA Sequence
- Outlining Specific parts of a sequence - Name Entity Recognition

# Notation Understanding

X: Rama Conquered Ravana to install the virtue of dharma

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ …… $x^{<t>}$ …….. $x^{<9>}$

$T_x = 9$ (Length of training sequence: 9)

$x^{i<t>}$ : $t^{th}$ word of $i^{th}$ training sequence

Y: 1 0 1 0 …… 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ …… $y^{<t>}$ …….. $y^{<9>}$

$T_y = 9$ (Length of output sequence: 9)

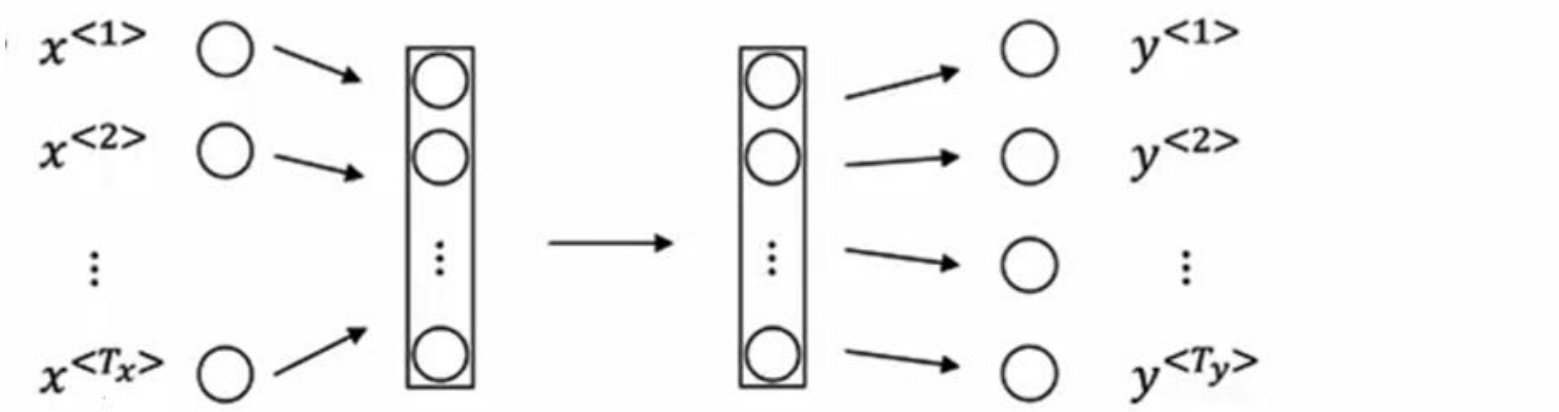
$y^{i<t>}$ : $t^{th}$ word of $i^{th}$ output sequence

# Representing words and one-hot encoding

X: Rama Conquered Ravana to install the virtue of dharma

| | | | |
|---|---|---|---|
| A | 1 | Rama | Ravana |
| : | | 0 | 0 |
| : | | 0 | 0 |
| Conquered | 329 | 0 | 0 |
| : | | 0 | 0 |
| : | | 0 | 0 |
| Install | 4521 | : | : |
| : | | : | : |
| : | | : | : |
| Rama | 7689 | 1 -7689 | : |
| : | | : | 1-7900 |
| Ravana | 7900 | : | : |
| : | | 0 | 0 |
| ZZZ | 10000 | 0 | 0 |

# Standard Neural Network Does not works out to give a good application for sequence models




Inputs, outputs can be different lengths in different examples.

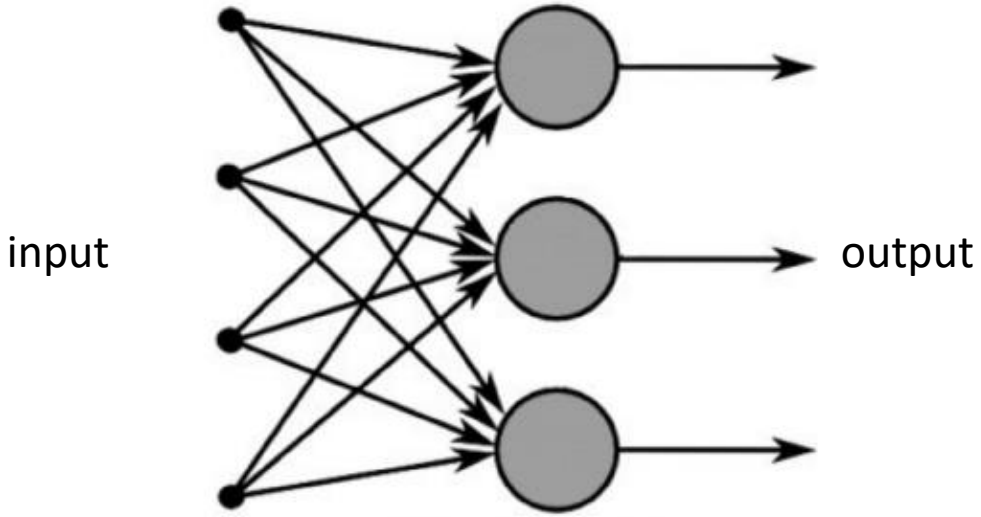
Doesn't share features learned across different positions of text.

- Feed forward networks accept a fixed-sized vector as input and produce a fixed-sized vector as output
- So, Feed forward networks cannot process sequential data containing variable length of data
- Feed forward networks does not consider sequence in the data

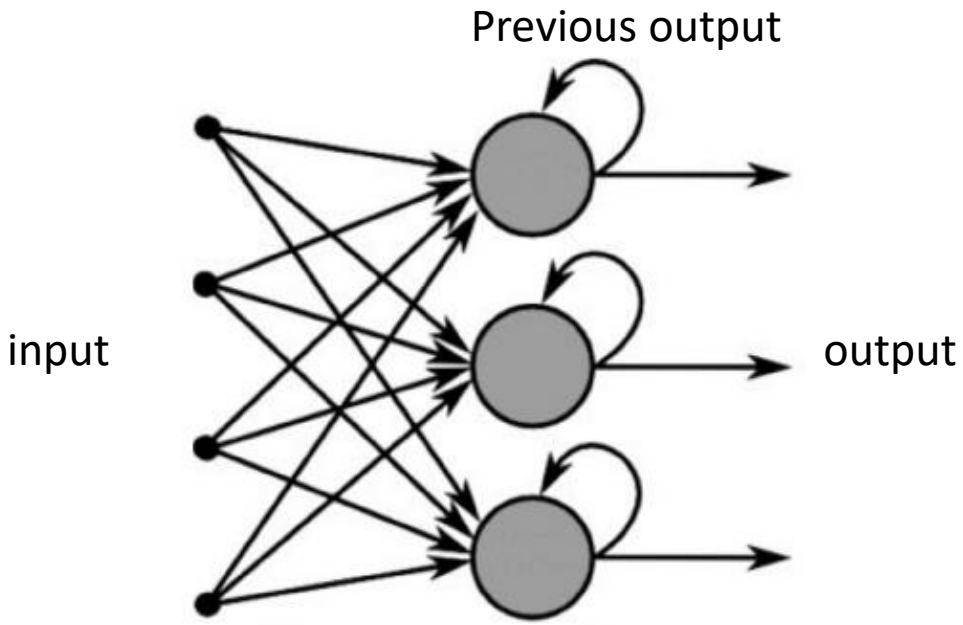
# Solution for Sequence Analysis - RNN

- Recurrent Neural Networks allow us to operate over sequences of vectors
- Recurrent, because previous output is also used with current input
- RNN also viewed as having a “memory”
- Unlike a traditional deep network, RNN shares same parameters across all steps
- Greatly reduces the total number of parameters we need to learn
- RNN is not a feed forward neural network as cycle is formed in hidden units

# NN vs RNN

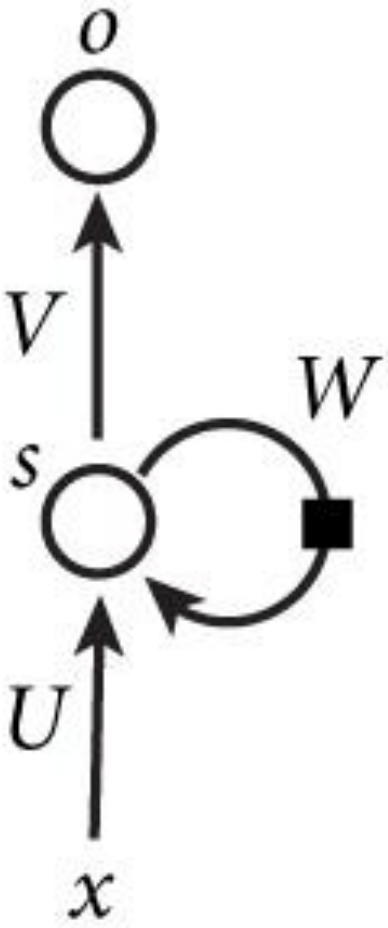



Feed-Forward Neural Network




Recurrent Neural Network

# Notations




x : Input

o : Output

s : state of the hidden unit

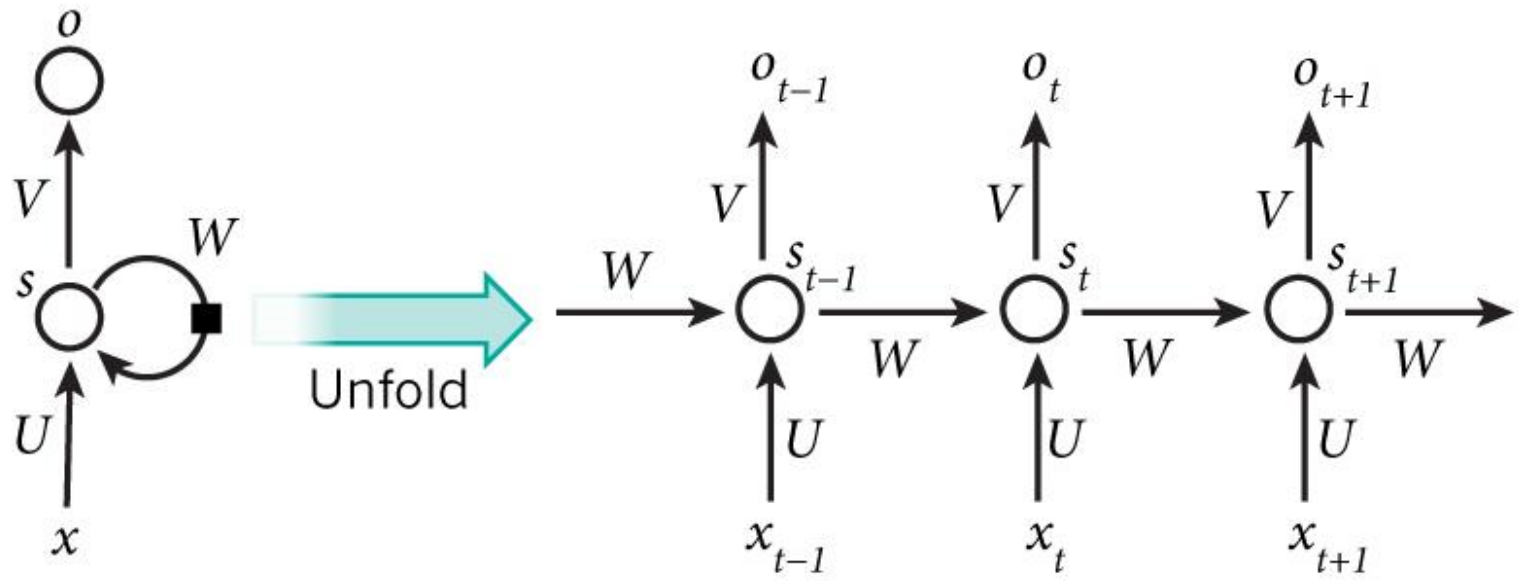
U, V and W : Weights to be learned

U : weights used for hidden state computation (from input)

V : weights used for output computation

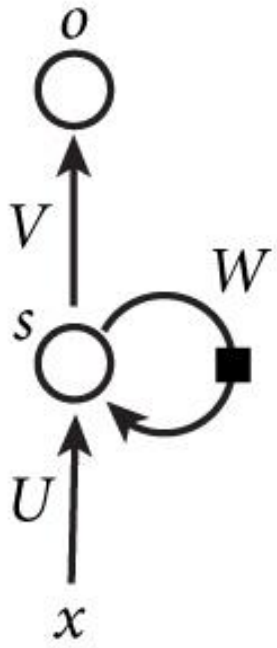
W : weights used for hidden state computation (from previous hidden state)

# Unrolled RNN with parameters



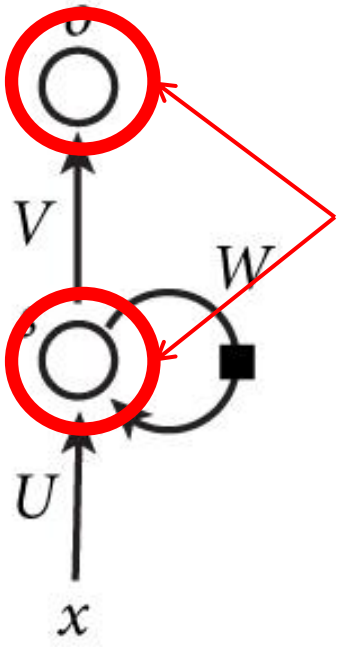

The recurrent network can be converted into a feed forward network by **unfolding over time**

# RNN Forward Pass

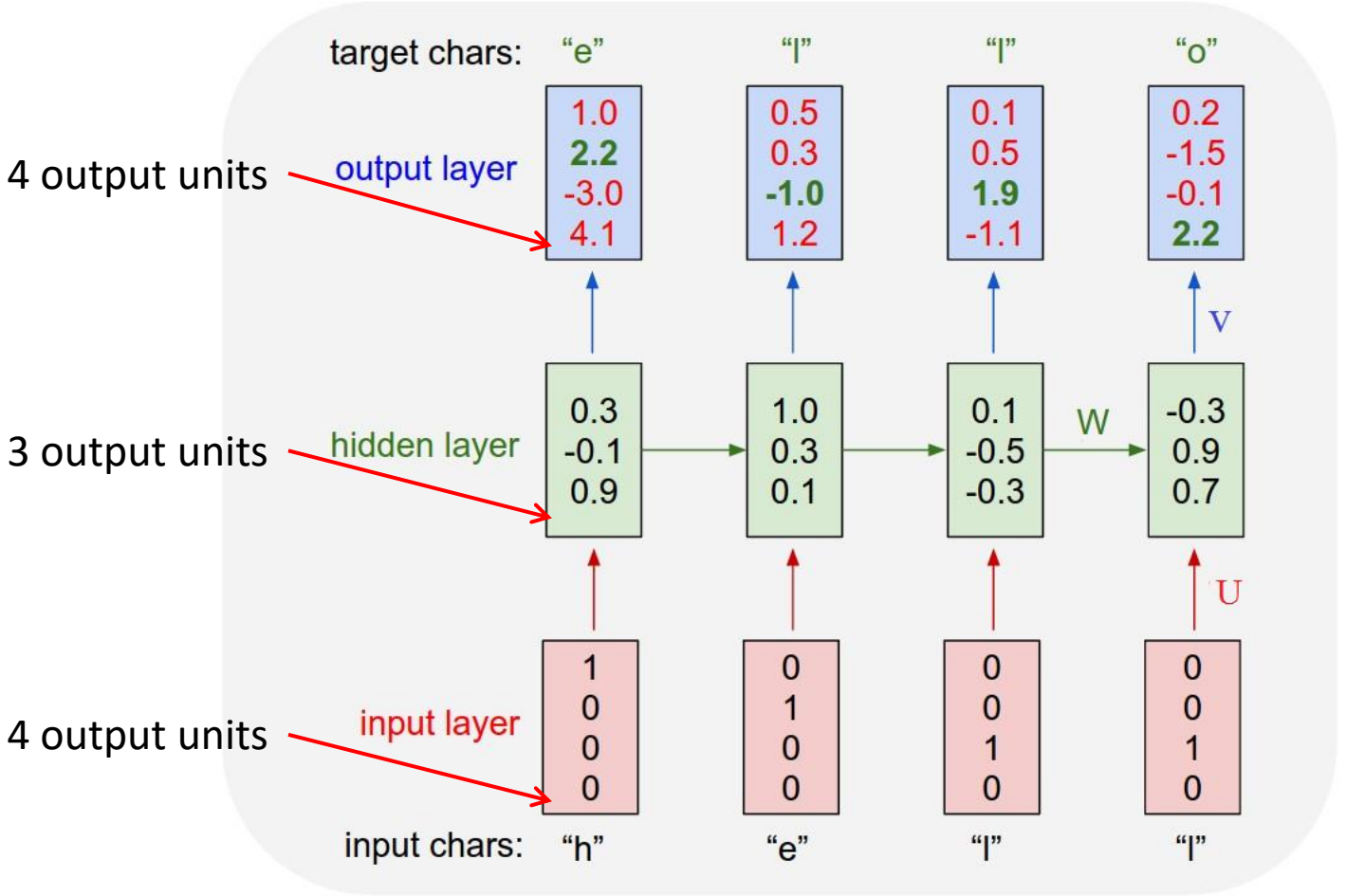



- Step 2: Current hidden state *s* at time *t* will be computed using
$$s(t) = f_h(Ux(t) + Ws(t\ -1)\ )$$
- Step 1: input *x* will be given at time *t*
- Step 3: Current output *o* at time *t* will be computed using
$$o(t) = f_o\big(Vs(t)\big)$$
- Note: Output will not necessarily be generated for every *t*. i.e. it depends upon application. Speech recognition RNN will output words instantly at every iteration. Opinion classification RNN will output label only at the end of sentence.

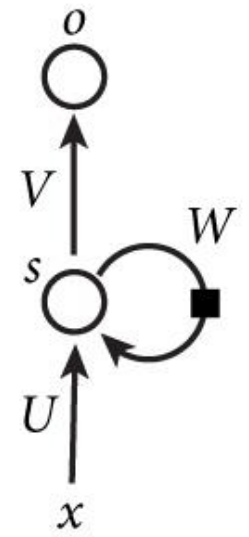
# Important Notes




These are not single units
There are single layers



# Forward Pass with Example




- The inputs are one hot encoded. Our entire vocabulary is {h,e,l,o} and hence we can easily one hot encode the inputs.

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 |
| h | e | l | l |

- Now the input neuron would transform the input to the hidden state using the weight U. We have randomly initialized the weights as a 3*4 matrix –

| U | | | |
|---|---|---|---|
| 0.287027 | 0.84606 | 0.572392 | 0.486813 |
| 0.902874 | 0.871522 | 0.691079 | 0.18998 |
| 0.537524 | 0.09224 | 0.558159 | 0.491528 |

# Step 1

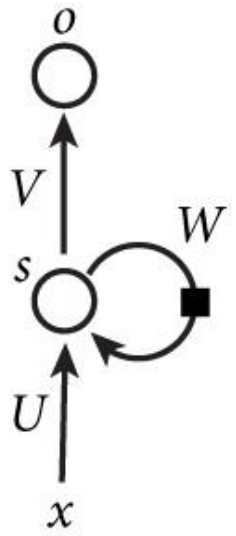
- Now for the letter “h”, for the the hidden state we would need $UX_t$. By matrix multiplication, we get it as

| U | | | |
|---|---|---|---|
| 0.287027 | 0.84606 | 0.572392 | 0.486813 |
| 0.902874 | 0.871522 | 0.691079 | 0.18998 |
| 0.537524 | 0.09224 | 0.558159 | 0.491528 |

| |
|---|
| 1 |
| 0 |
| 0 |
| 0 |
| h |



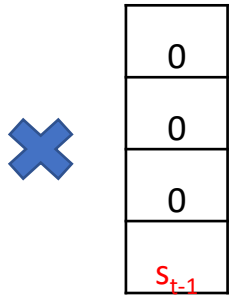
| |
|---|
| 0.287027 |
| 0.902874 |
| 0.537524 |

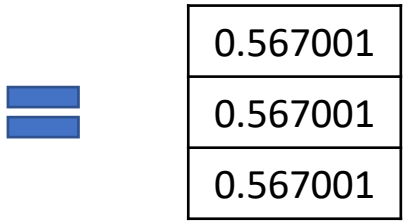


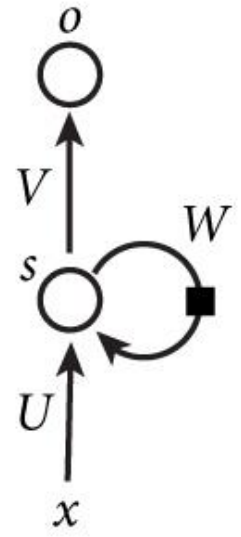
# Step 2

- Now moving to the recurrent neuron, we have *W* as the weight which is a 1*1 matrix as **0.427043** and the bias which is also a 1*1 matrix as **0.56700**
- For the letter “h”, the previous state is [0,0,0] since there is no letter prior to it.
- So to calculate -> $(W*s_{t-1}+bias)$

| W | bias |
|---|---|
| 0.427043 | 0.567001 |



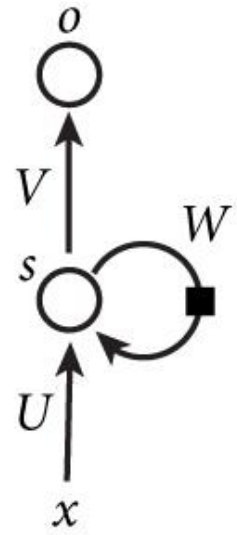
# Step 3

- Now we can get the current state as

$$s_t = \tanh(Ws_{t-1} + Ux_t)$$

- Since for h, there is no previous hidden state we apply the tanh function to this output and get the current state $s_t$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0.287027 | + | 0.567001 | = | tanh | 0.854028 | = | 0.693168 |
| 0.902874 | | 0.567001 | | | 1.469875 | | 0.899554 |
| 0.537524 | | 0.567001 | | | 1.104525 | | 0.802118 |



# Step 4

- Now we go on to the next state. “e” is now supplied to the network. The processed output of $s_t$, now becomes $s_{t-1}$, while the one hot encoded e, is $x_t$. Let’s now calculate the current state $s_t$.

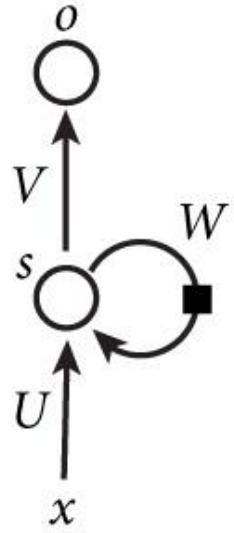
$$h_t = \tanh(Ws_{t-1} + Ux_t)$$

- $Ws_{t-1}$ +bias will be

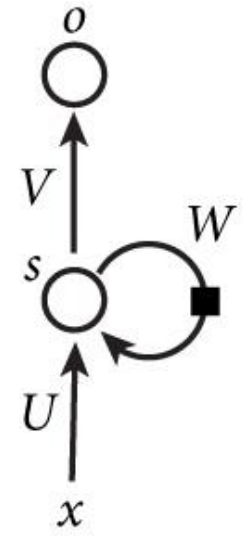
| | | | | | |
|---|---|---|---|---|---|
| 0.427043 | × | 0.693168<br>0.899554<br>0.802118 | + | 0.567001 | = | 0.863013<br>0.951149<br>0.90954 |

- $Ux_t$ will be

| U | | | | | | |
|---|---|---|---|---|---|---|
| 0.287027 | 0.84606 | 0.572392 | 0.486813 | × | 0<br>1<br>0<br>0<br>e | = | 0.84606<br>0.871522<br>0.09224 |
| 0.902874 | 0.871522 | 0.691079 | 0.18998 | | | | |
| 0.537524 | 0.09224 | 0.558159 | 0.491528 | | | | |



# Step 5




- Now calculating $s_t$ for the letter “e”,

$s_t$ = tanh

| | | |
|---|---|---|
| 0.863013 | + | 0.84606 |
| 0.951149 | | 0.871522 |
| 0.90954 | | 0.09224 |

= 

| |
|---|
| 0.93653372 |
| 0.94910403 |
| 0.76234056 |

- Now this would become $s_{t-1}$ for the next state and the recurrent neuron would use this along with the new character to predict the next one.

# Step 6

- At each state, the recurrent neural network would produce the output as well. Let's calculate $y_t$ for the letter e.

$$Y_t = Vs_t$$

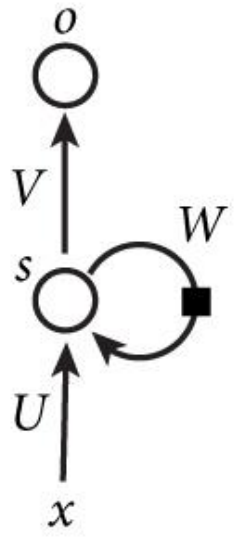
| V | | |
|---|---|---|
| 0.37168 | 0.974829459 | 0.830034886 |
| 0.39141 | 0.282585823 | 0.659835709 |
| 0.64985 | 0.09821557 | 0.334287084 |
| 0.91266 | 0.32581642 | 0.144630018 |

×

| $s_t$ |
|---|
| 0.93653372 |
| 0.94910403 |
| 0.76234056 |

=

| $y_t$ |
|---|
| 1.90607732 |
| 1.13779113 |
| 0.95666016 |
| 1.27422602 |



# Step 7

- The probability for a particular letter from the vocabulary can be calculated by applying the softmax function. so we shall have softmax($y_t$)

Classwise probabilities for next letter = softmax

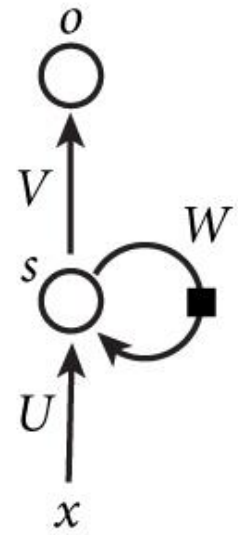
| $y_t$ |
|---|
| 1.90607732 |
| 1.13779113 |
| 0.95666016 |
| 1.27422602 |

=

| |
|---|
| 0.419748 |
| 0.194682 |
| 0.162429 |
| 0.223141 |

Letter h got high probability

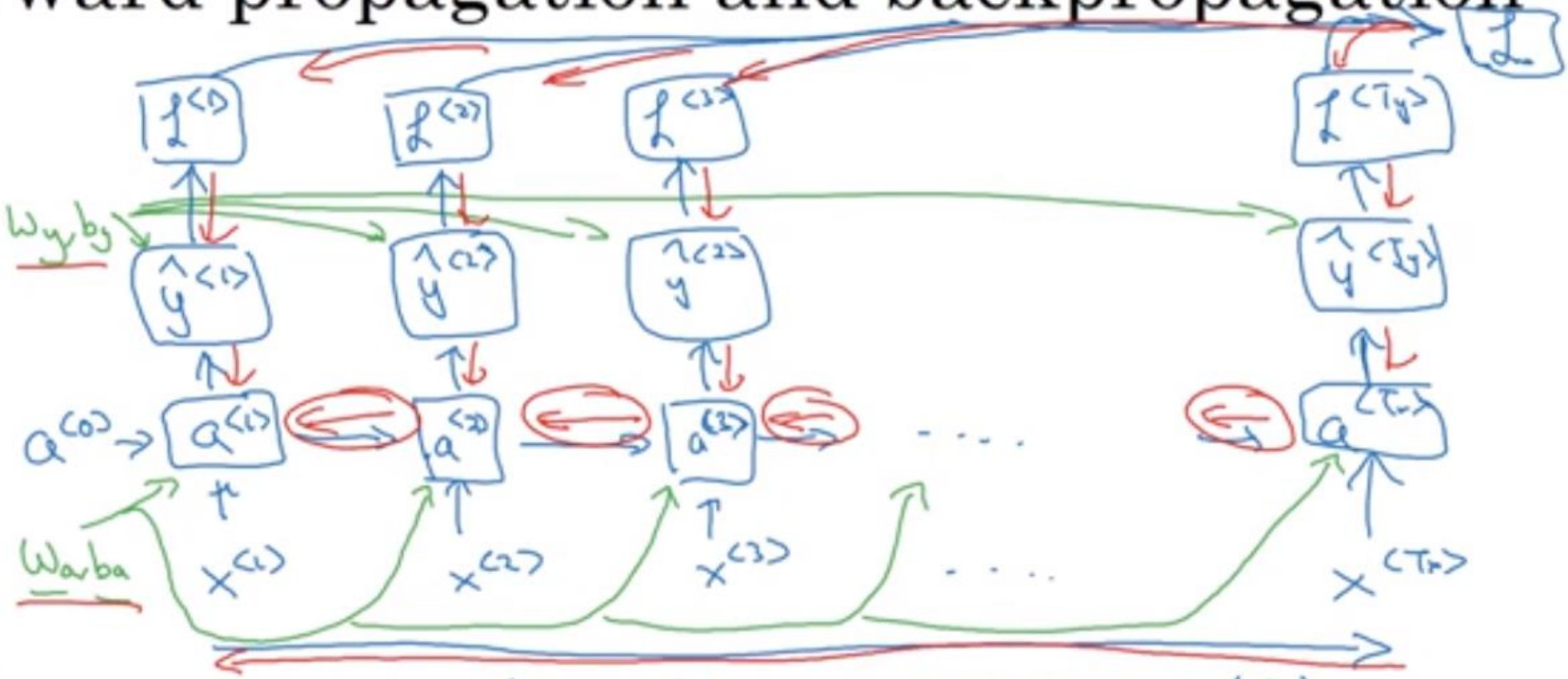
- If we convert these probabilities to understand the prediction, we see that the model says that the letter after “e” should be h, since the highest probability is for the letter “h”. Does this mean we have done something wrong? No, so here we have hardly trained the network. We have just shown it two letters. So it pretty much hasn’t learnt anything yet.
- Now the next BIG question that faces us is how does Back propagation work in case of a Recurrent Neural Network. How are the weights updated while there is a feedback loop?



# Back Propagation Through Time

- BPTT learning algorithm is an extension of standard backpropagation that performs gradients descent on an unfolded network.
- The gradient descent weight updates have contributions from each time step.
- The errors have to be back-propagated through time as well as through the network

# Forward propagation and backpropagation




$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>}\log\hat{y}^{<t>} - (1-y^{<t>})\log(1-\hat{y}^{<t>})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Backpropagation through time

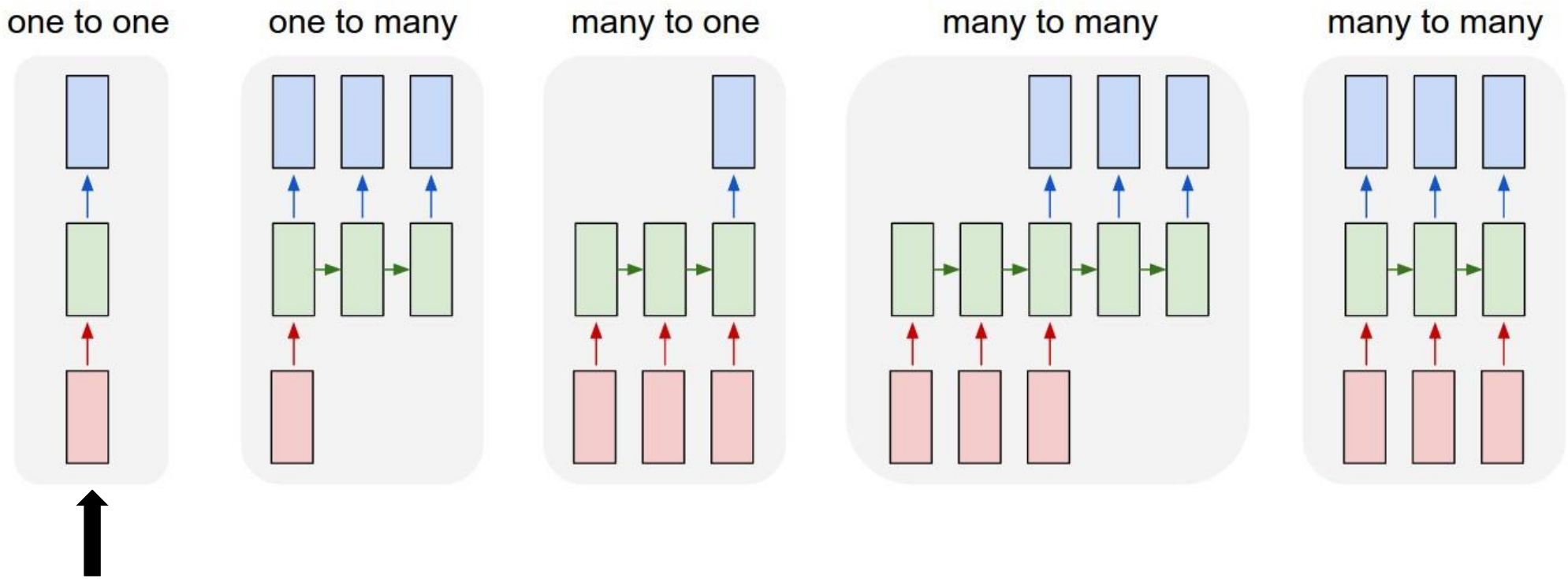
# RNN using Keras

- **</>**

- keras.layers.RNN(cell, return_sequences=False, return_state=False, go_backwards=False, stateful=False, unroll=False)

**Simple.**

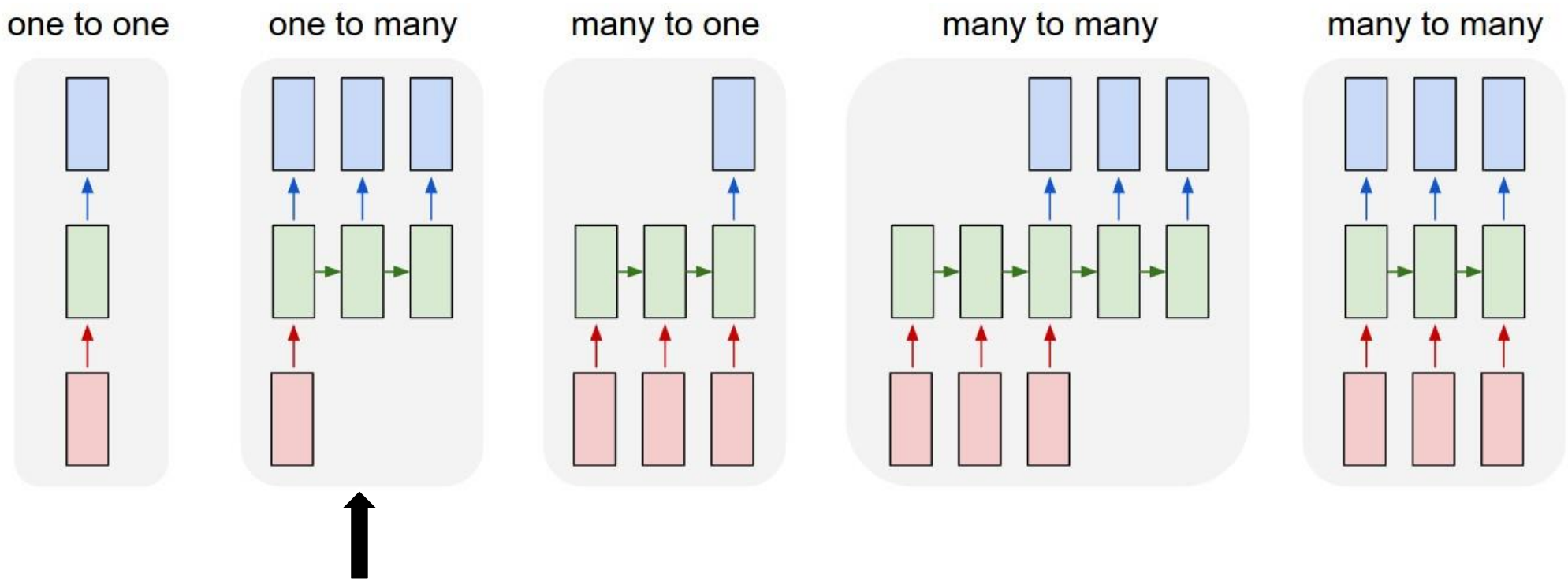
# RNN Variants




Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification)

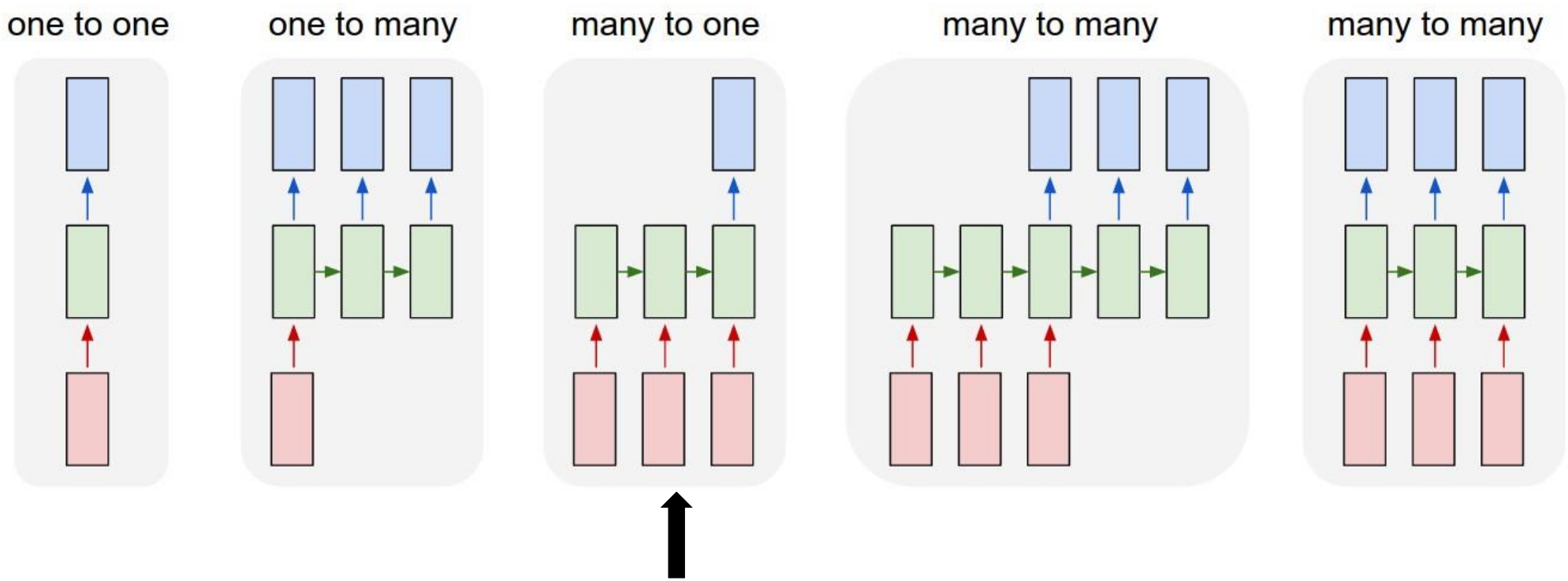
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

# RNN Variants



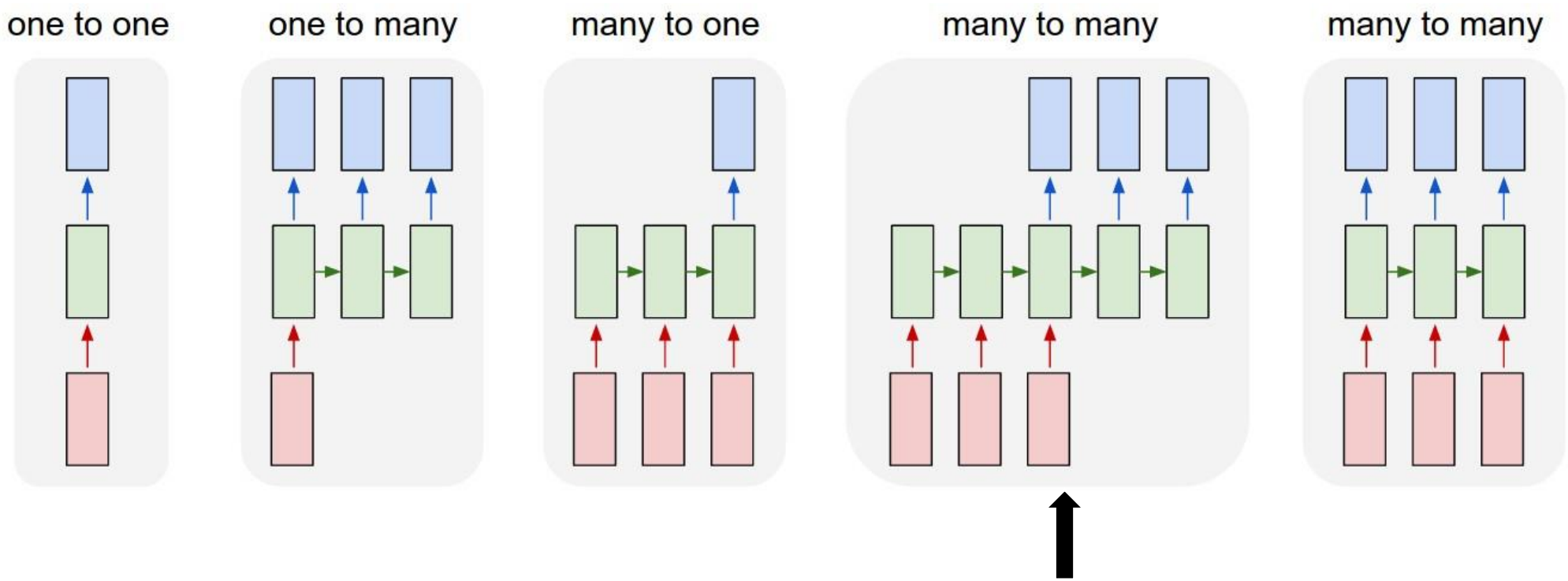

Sequence output (e.g. image captioning takes an image and outputs a sentence of words)

# RNN Variants



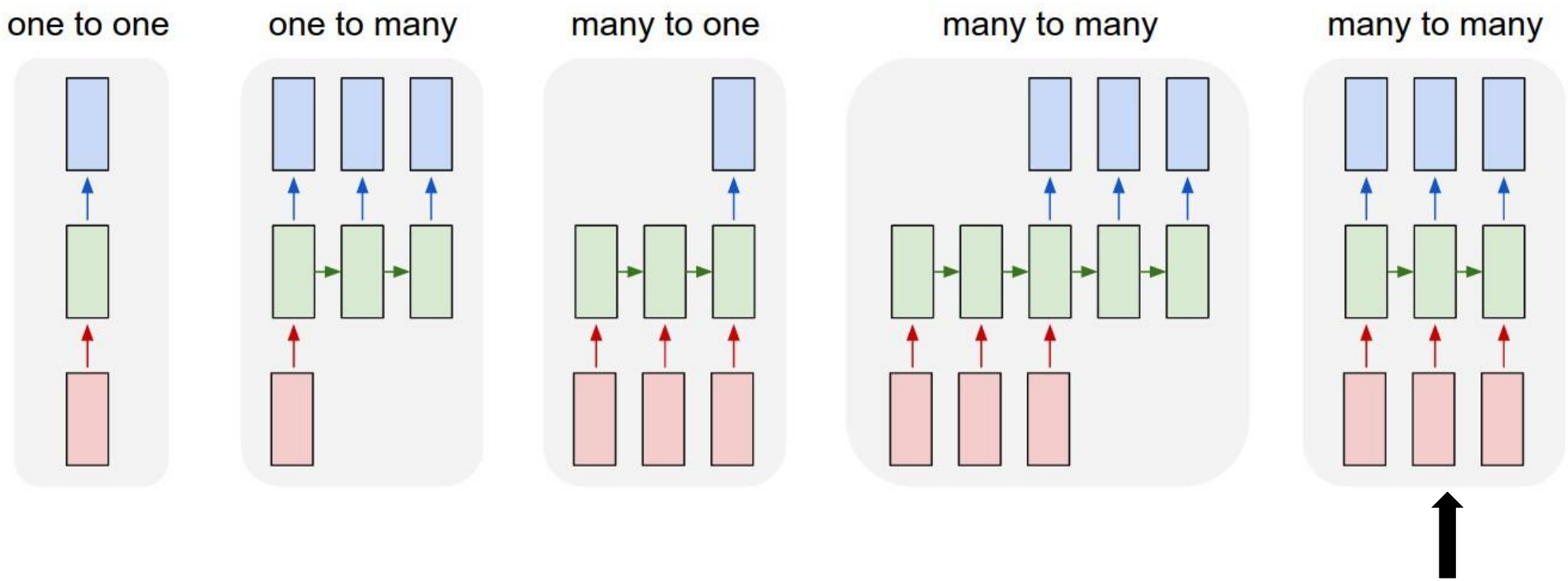

Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment)

# RNN Variants




Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French)

# RNN Variants



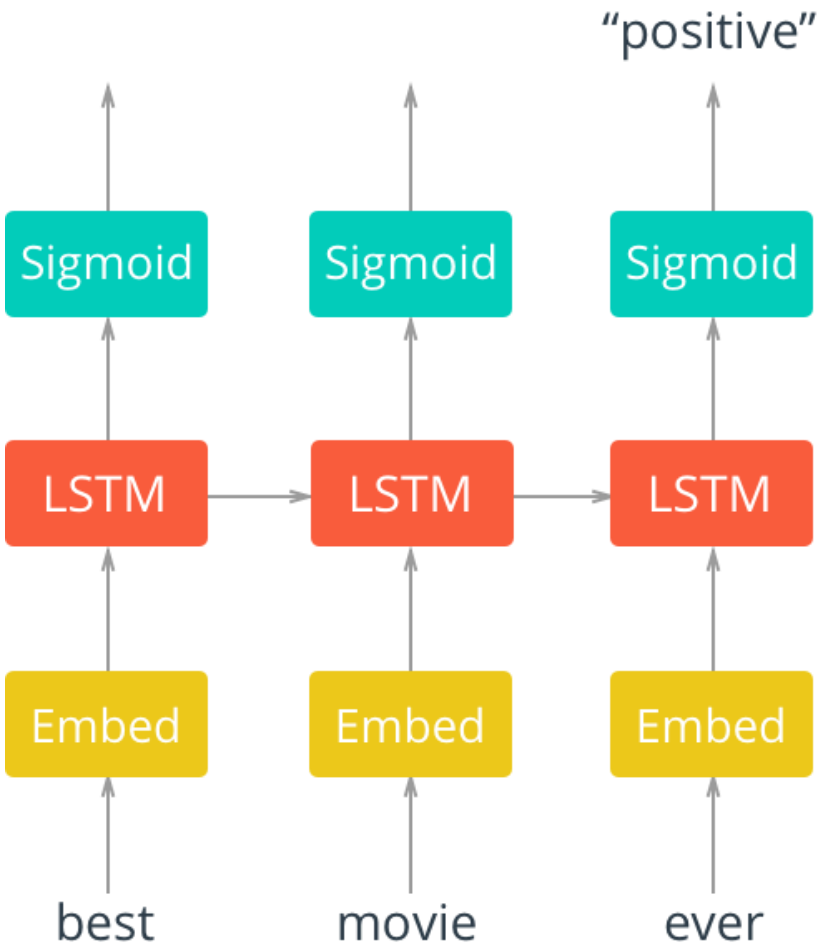

Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

# RNN Applications

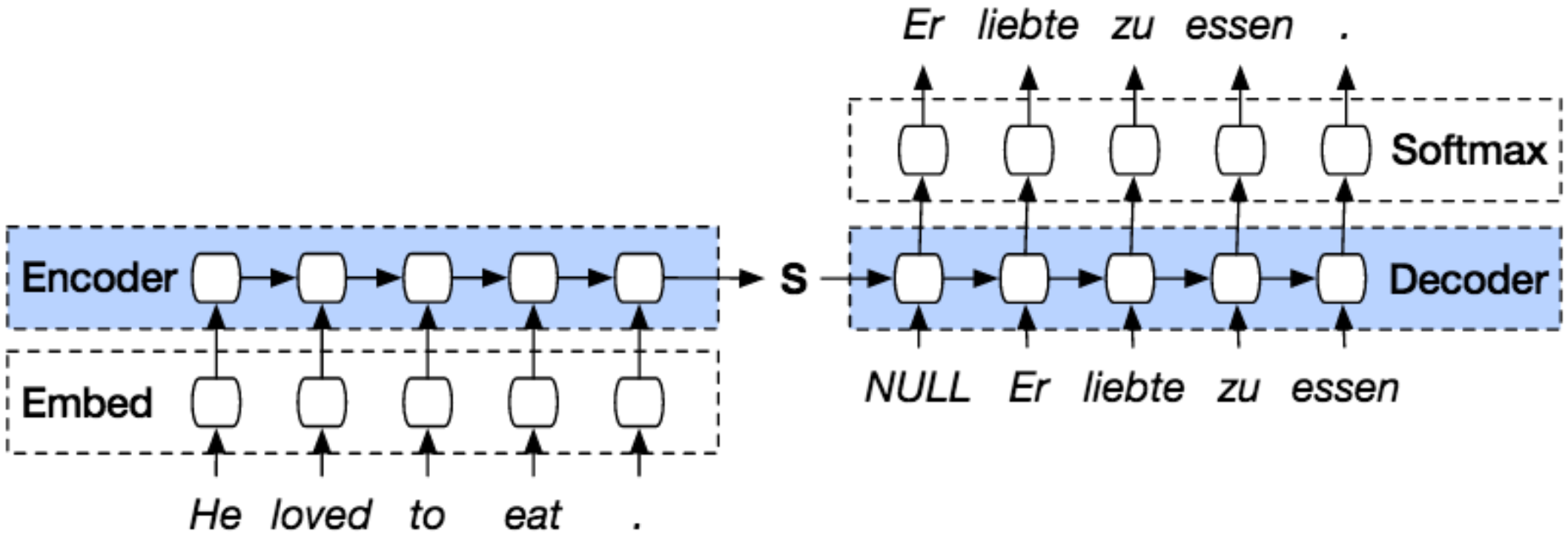
- Robot control
- Time series prediction
- Speech recognition
- Rhythm learning
- Music composition
- Grammar learning
- Handwriting recognition
- Human action recognition
- Protein Homology Detection
- Predicting subcellular localization of proteins
- Prediction tasks in the area of business process management
- Prediction in medical care pathways

Wherever you have Sequential Data !

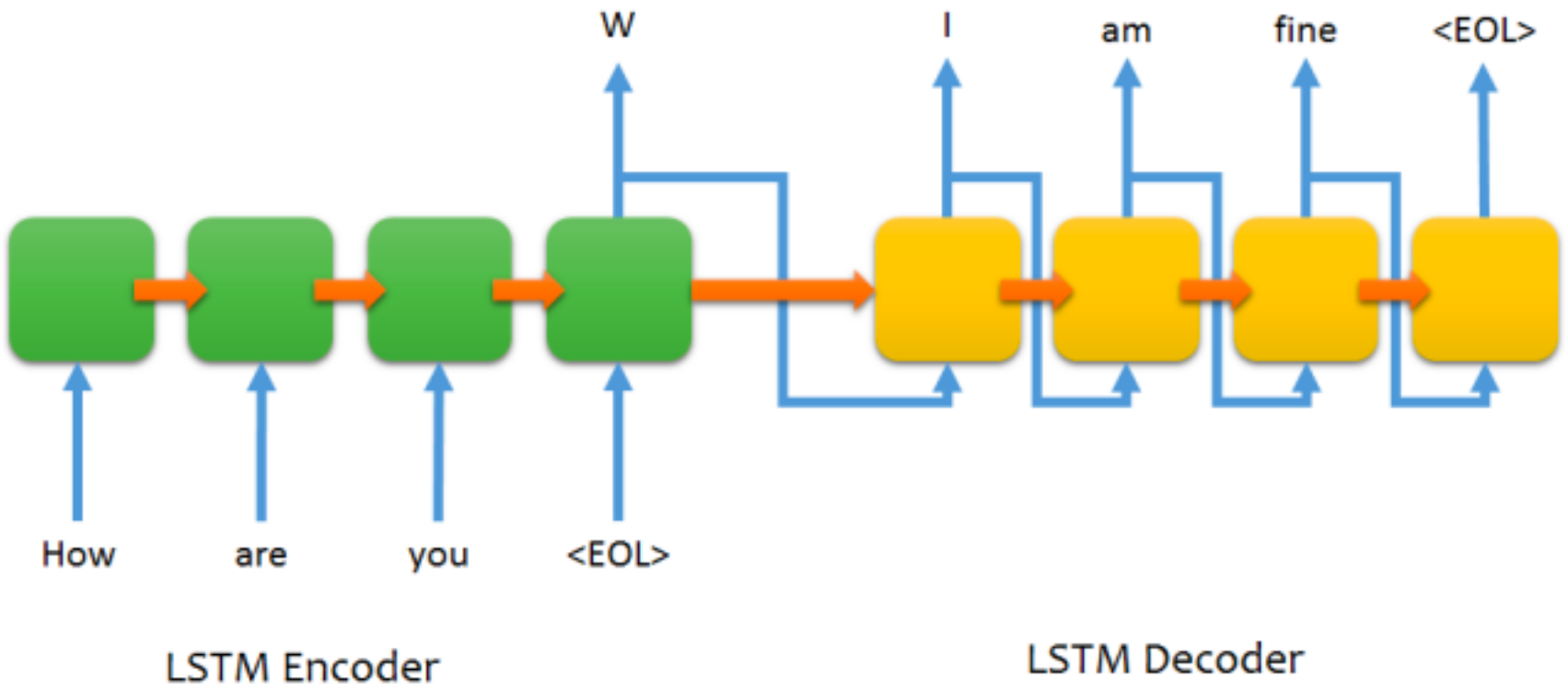
# Sentiment Classification



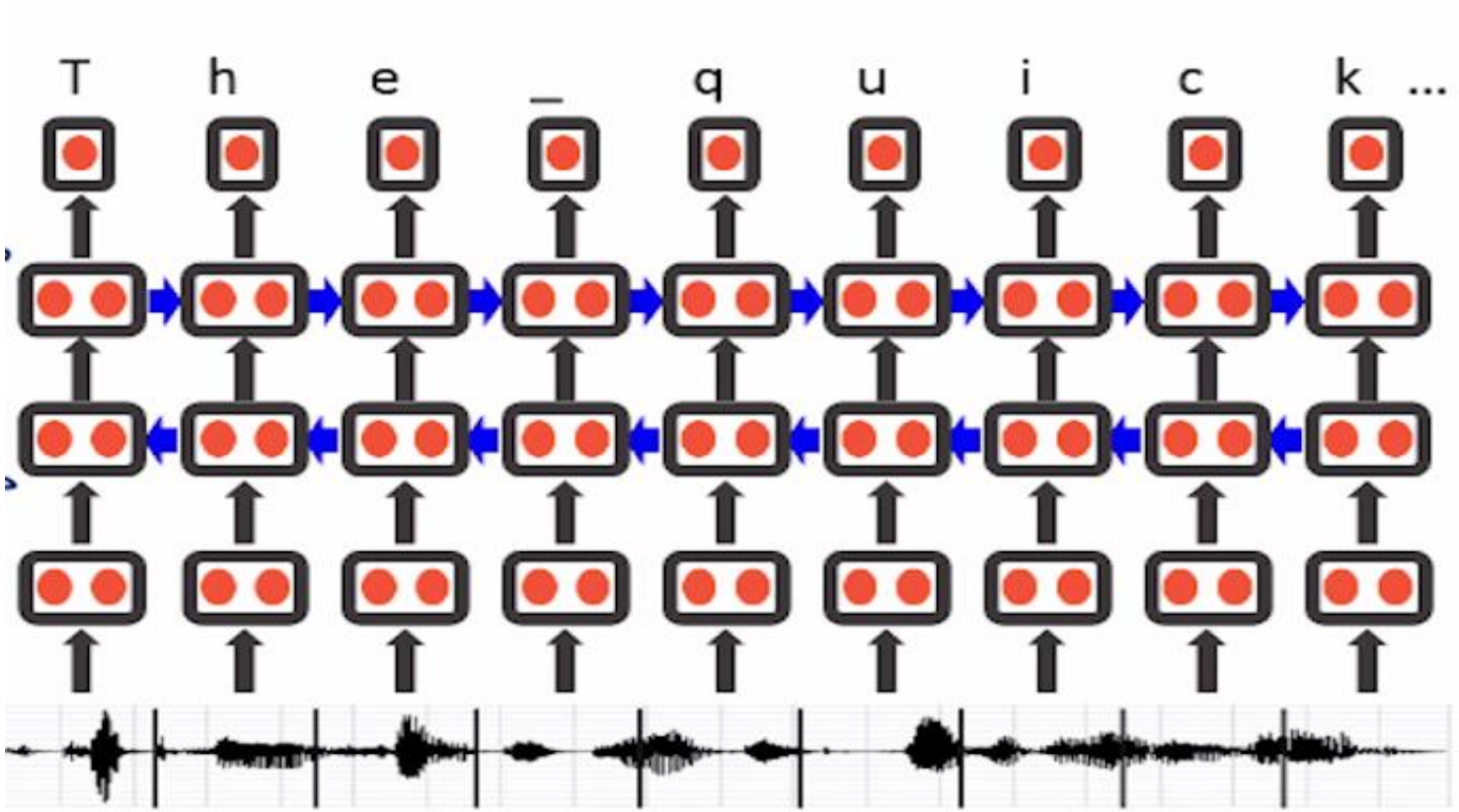
# Neural Machine Translation



# Sequence to Sequence chat model



# Baidu’s speech recognition using RNN



# Bidirectional RNNs

“He said, Teddy bears are on sale” and “He said, Teddy Roosevelt was a great President”.

In the above two sentences, when we are looking at the word “Teddy” and the previous two words “He said”, we might not be able to understand if the sentence refers to the President or Teddy bears.

Therefore, to resolve this ambiguity, we need to look ahead. This is what Bidirectional RNNs accomplish.