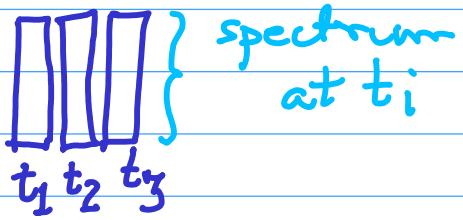
RNN: RECURRENT NEURAL NETWORK

RNN: modelling sequence data

- One snapshot of a ball cannot help us predict where it will go next, but a sequence of images will help us make a better prediction
- audio: raw waveform or spectrogram

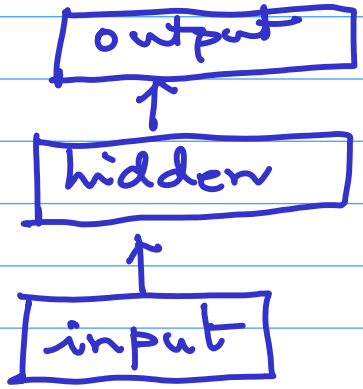


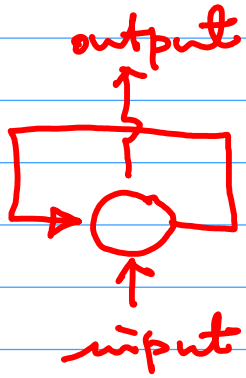
- Text: sequence of words/characters

We often learn sequences e.g.: alphabet A B C D ··· Z. Other way 'round? tough! Break it up into smaller parts e.g, start at F: after a few tries, we can get it right.

Model (in terms of the familiar feedforward neural network)

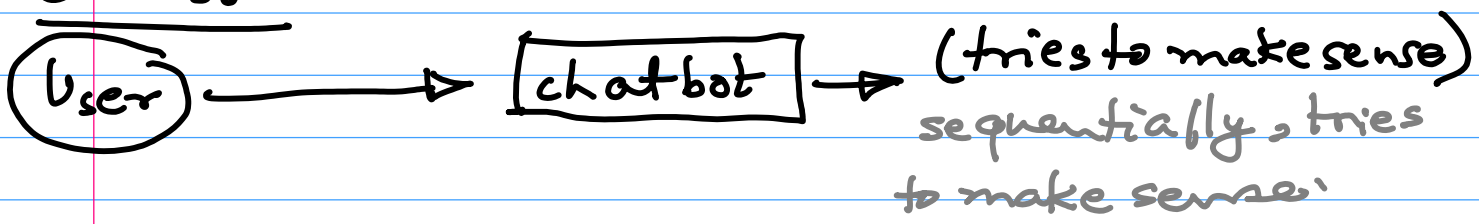
write this: top to bottom →





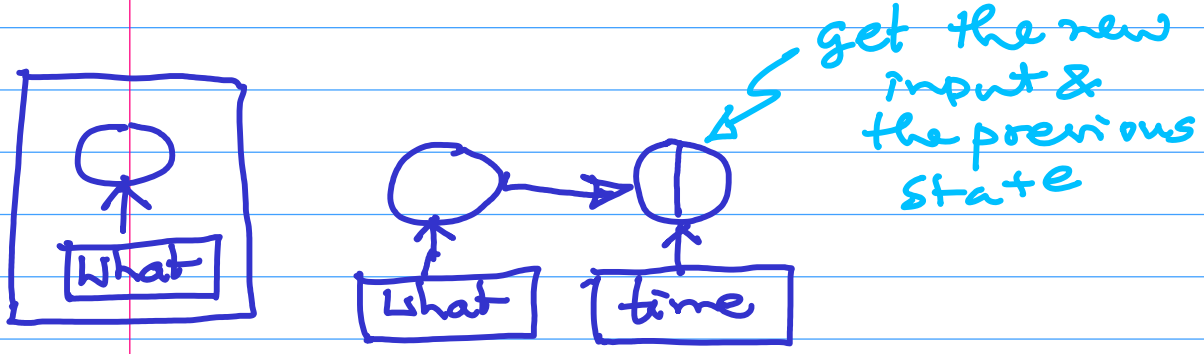
Add a loop in the feedforward neural network to make it use previous information

Chatbot:



"What time is it?"

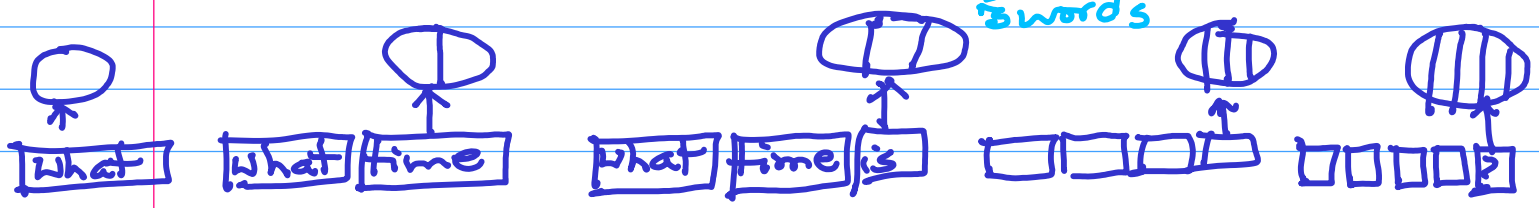
sequence of words



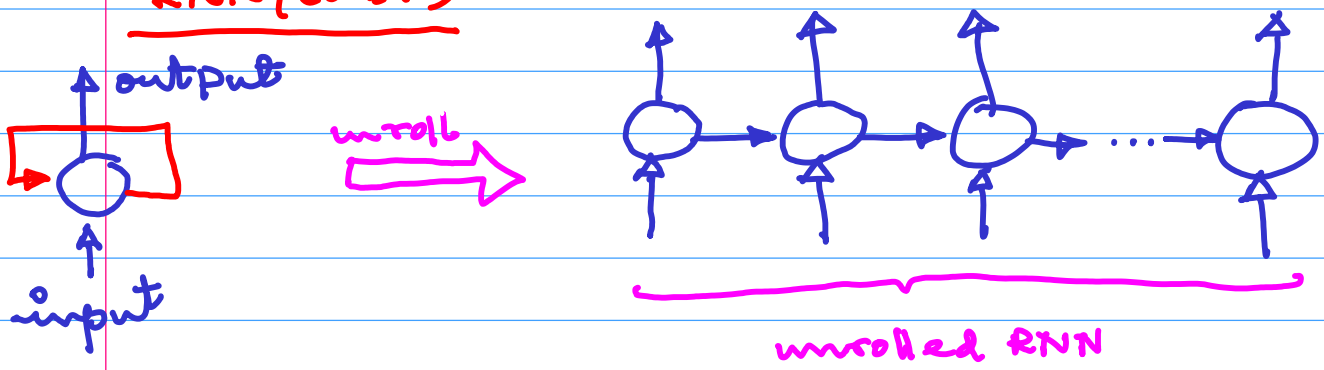
As the input comes in, feed the input and the previous hidden state → once there is no more input

We feed this to the output of the ordinary feed forward network

A representational illustration of the short-term memory problem in RNNs:—



RNN (contd.)



The effect: As the RNN processes more input, it has trouble retaining information from previous steps. Information from the word "what" is almost non-existent at the final step

Cause: the infamous vanishing gradient problem

input ← hidden layers ← output

In the propagation of the error from the output side towards the input, the gradient gets so small that the weights in the layers towards the input side (via the chain rule) do not change at all.

The gradient vanishes as we go from the output to the input side.

Interpretation for an RNN:

Think of each time step as a layer of a feedforward neural network. "Backpropagation through time"

'SHORT-TERM MEMORY' → causes early layers not to learn. In this example: there is a possibility that the words "what" and "time" are not considered at all, when trying to predict the user's intention. The network makes the best guess with "is it?"

Andrej Karpathy's notes:

* What does training with characters (for example) mean in the input and output?

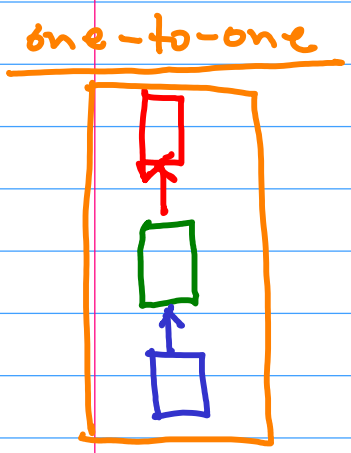
e.g., training with 4 characters h, e, l, o

→ "1-hot encoding" or "1-of-k-encoding"

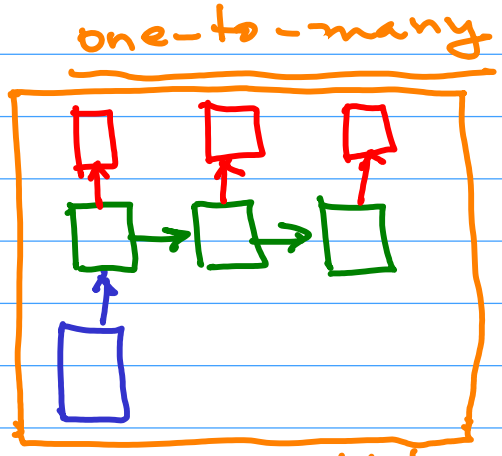
$$h: \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad e: \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad l: \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad o: \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

→ If the output is also expected to be a character, we will try to get similar values (approximate) from the outputs of a suitable neural work.

RNN uses: even if the data is not in the form of sequences, one can still formulate and train RNNs which process data sequentially.

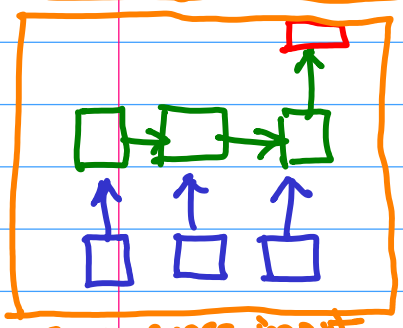


without RNN:
fixed sized input to a fixed-size output e.g., image classification



Sequence output
e.g. image captioning takes an image as input and gives as output: a sentence of words.

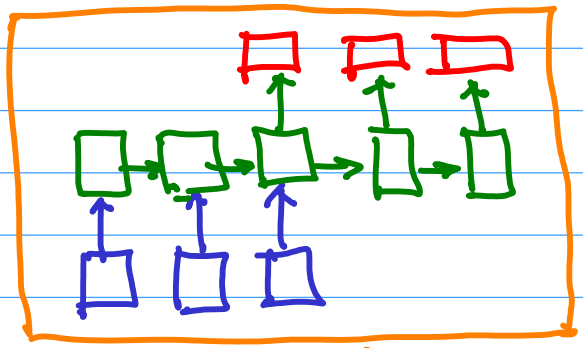
many-to-one



sequence input

e.g., sentiment analysis: given a sentence, it is classified as expressing positive or negative sentiment

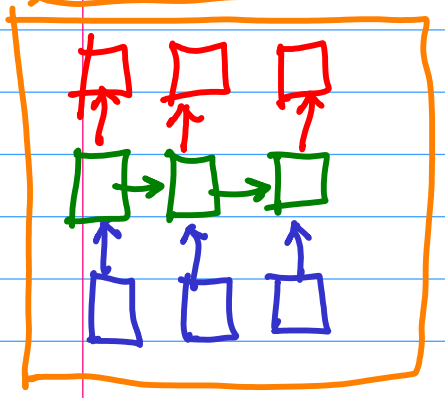
many-to-many



sequence i/p & sequence o/p

e.g., machine translation from English to French.

many-to-many



Synchronised sequence input and output

e.g., video classification: to label each video frame.

Basic Structure:

There are no pre-specified constraints on the lengths of the sequences because the recurrent transformation (hidden) is fixed, and can be applied as many times as we like.