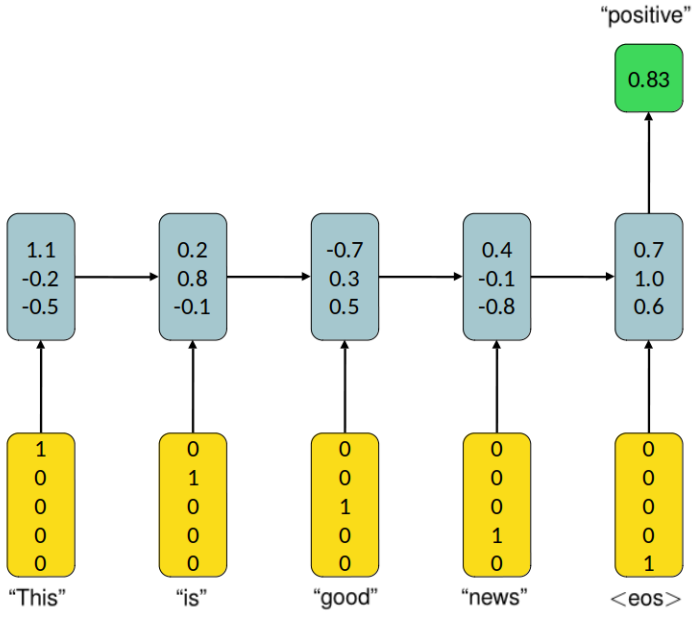
Deep Learning

Recurrent Neural Networks - Introduction



Learning goals

- Why do we need them?
- How do they work?
- Computational Graph o Recurrent Networks

Motivation

MOTIVATION FOR RECURRENT NETWORKS

- The two types of neural network architectures that we've seen so far are fully-connected networks and CNNs.
- Their input layers have a fixed size and (typically) only handle fixed-length inputs.
- The primary reason: if we vary the size of the input layer, we would also have to vary the number of learnable weights in the network.
- This in particular relates to **sequence data** such as time-series, audio and text.
- **Recurrent Neural Networks (RNNs)** is a class of architectures that allows varying input lengths and properly accounts for the ordering in sequence data.

RNNS - INTRODUCTION

- Suppose we have some text data and our task is to analyse the *sentiment* in the text.
- For example, given an input sentence, such as "This is good news.", the network has to classify it as either 'positive' or 'negative'.
- We would like to train a simple neural network (such as the one below) to perform the task.

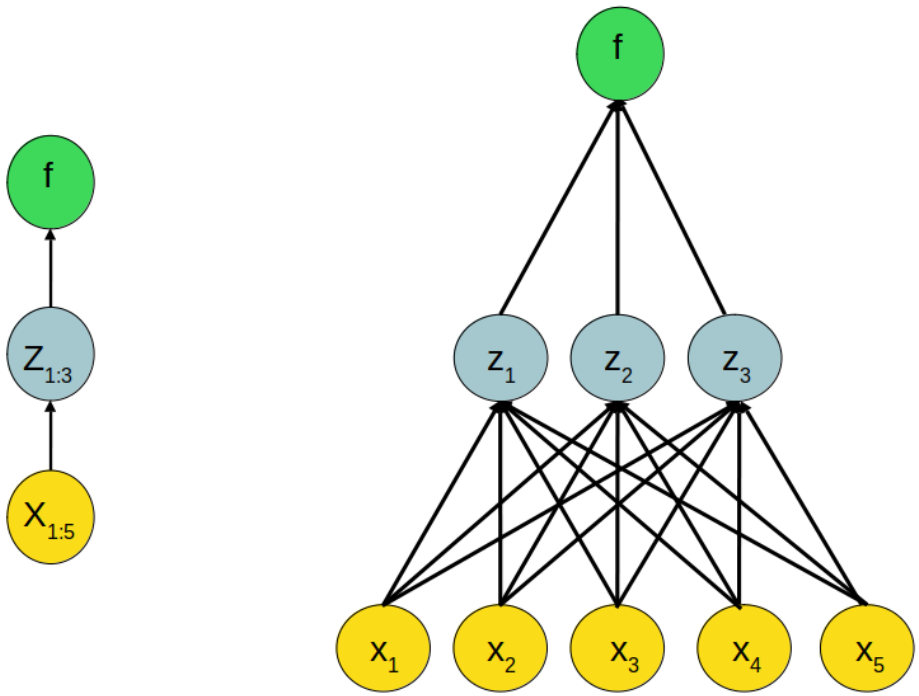


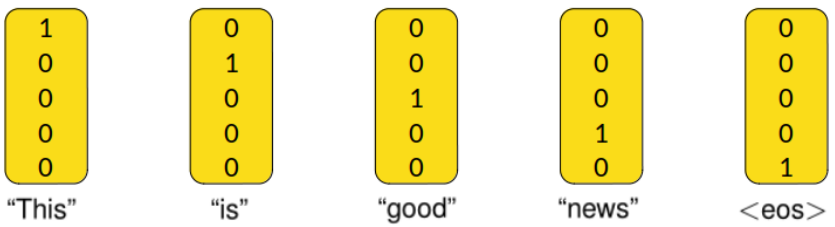
Figure: Two equivalent visualizations of a dense net with a single hidden layer, where the left is more abstract showing the network on a layer point-of-view.

RNNS - INTRODUCTION

- Because sentences can be of varying lengths, we need to modify the dense net architecture to handle such a scenario.
- One approach is to draw inspiration from the way a human reads a sentence; that is, one word at a time.
- An important cognitive mechanism that makes this possible is "**short-term memory**".
- As we read a sentence from beginning to end, we retain some information about the words that we have already read and use this information to understand the meaning of the entire sentence.
- Therefore, in order to feed the words in a sentence sequentially to a neural network, we need to give it the ability to retain some information about past inputs.

RNNS - INTRODUCTION

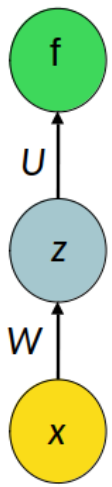
- When words in a sentence are fed to the network one at a time, the inputs are no longer independent. It is much more likely that the word "good" is followed by "morning" rather than "plastic". Hence, we also need to model this **(long-term) dependency**.
- Each word must still be encoded as a fixed-length vector because the size of the input layer will remain fixed.
- Here, for the sake of the visualization, each word is represented as a 'one-hot coded' vector of length 5. (<eos> = 'end of sequence')

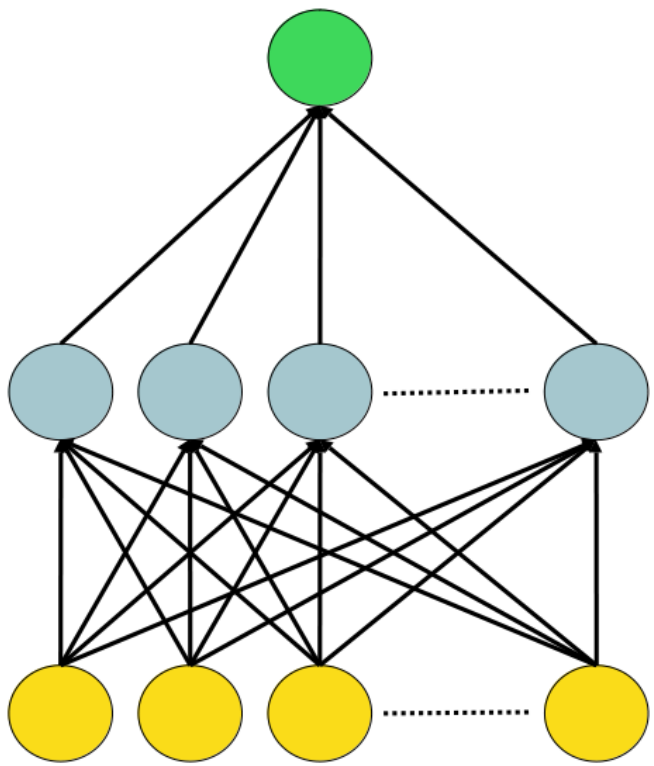


While this is one option to represent words in a network, the standard approach are word embeddings (more on this later).

RNNS - INTRODUCTION

- Our goal is to feed the words to the network sequentially in discrete time-steps.
- A regular dense neural network with a single hidden layer only has two sets of weights: 'input-to-hidden' weights **W** and 'hidden-to-output' weights **U**.





RNNS - INTRODUCTION

- In order to enable the network to retain information about past inputs, we introduce an **additional set of weights V**, from the hidden neurons at time-step t to the hidden neurons at time-step $t + 1$.
- Having this additional set of weights makes the activations of the hidden layer depend on **both** the current input and the activations for the *previous* input.

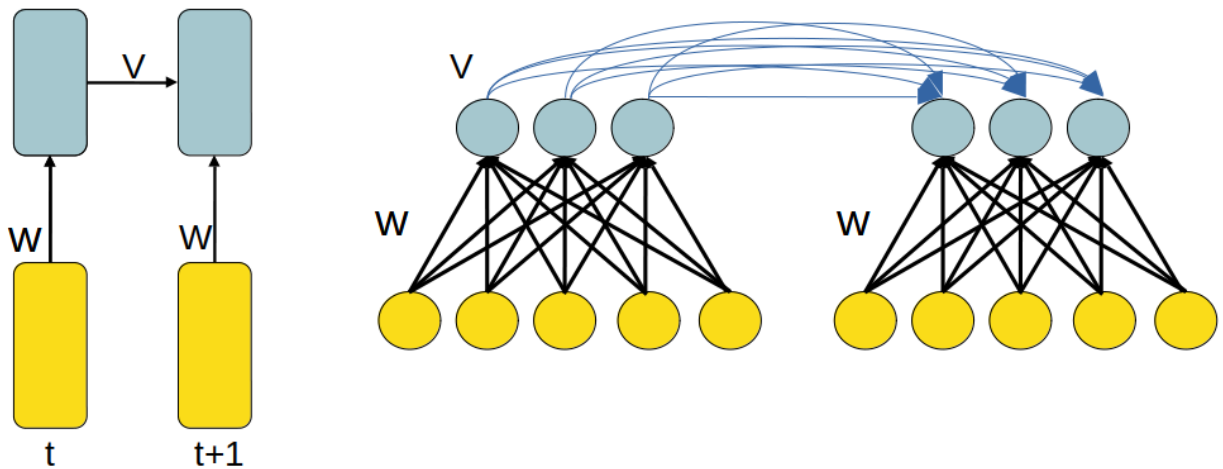


Figure: Input-to-hidden weights **W** and **hidden-to-hidden** weights **V**. The hidden-to-output weights **U** are not shown for better readability.

RNNS - INTRODUCTION

- With this additional set of hidden-to-hidden weights $\mathbf{V}$, the network is now a Recurrent Neural Network (RNN).
- In a regular feed-forward network, the activations of the hidden layer are only computed using the input-hidden weights $\mathbf{W}$ (and bias $\mathbf{b}$).

$$\mathbf{z} = \sigma(\mathbf{W}^\top \mathbf{x} + \mathbf{b})$$

- In an RNN, the activations of the hidden layer (at time-step t) are computed using *both* the input-to-hidden weights W and the hidden-to-hidden weights $\mathbf{V}$.

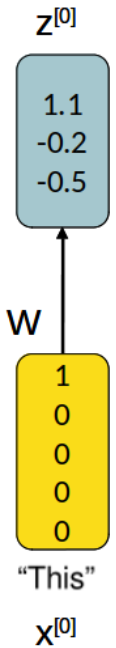
$$\mathbf{z}^{[t]} = \sigma(\mathbf{V}^\top \mathbf{z}^{[t-1]} + \mathbf{W}^\top \mathbf{x}^{[t]} + \mathbf{b})$$

- The vector $\mathbf{z}^{[t]}$ represents the short-term memory of the RNN because it is a function of the current input $\mathbf{x}^{[t]}$ and the activations $\mathbf{z}^{[t-1]}$ of the previous time-step.
- Therefore, by recurrence, it contains a "summary" of *all* previous inputs.

Examples

APPLICATION EXAMPLE - SENTIMENT ANALYSIS

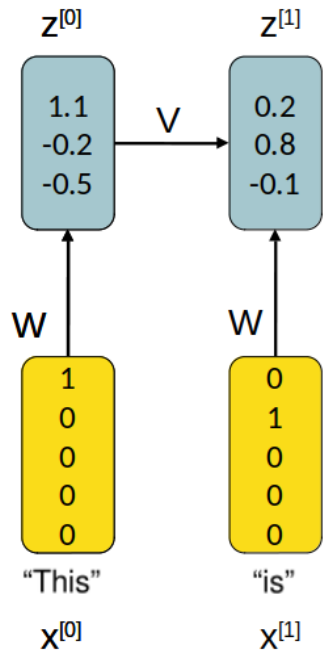
- At $t = 0$, we feed the word "This" to the network and obtain $\mathbf{z}^{[0]}$.
- $\mathbf{z}^{[0]} = \sigma(\mathbf{W}^\top \mathbf{x}^{[0]} + \mathbf{b})$



Because this is the very first input, there is no past state (or, equivalently, the state is initialized to 0).

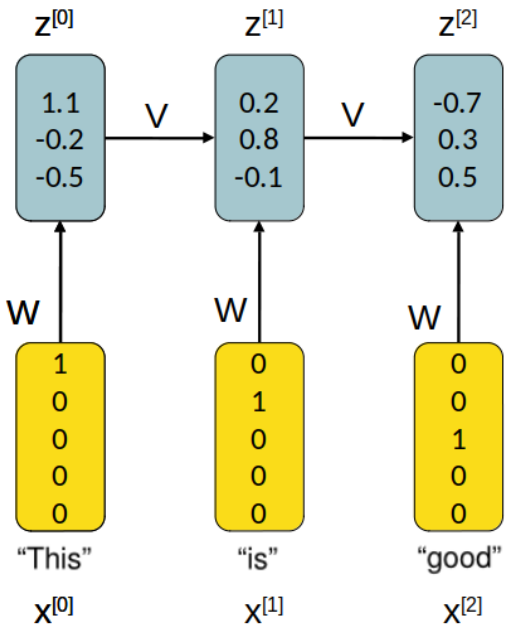
APPLICATION EXAMPLE - SENTIMENT ANALYSIS

- At $t = 1$, we feed the second word to the network to obtain $\mathbf{z}^{[1]}$.
- $\mathbf{z}^{[1]} = \sigma(\mathbf{V}^\top \mathbf{z}^{[0]} + \mathbf{W}^\top \mathbf{x}^{[1]} + \mathbf{b})$



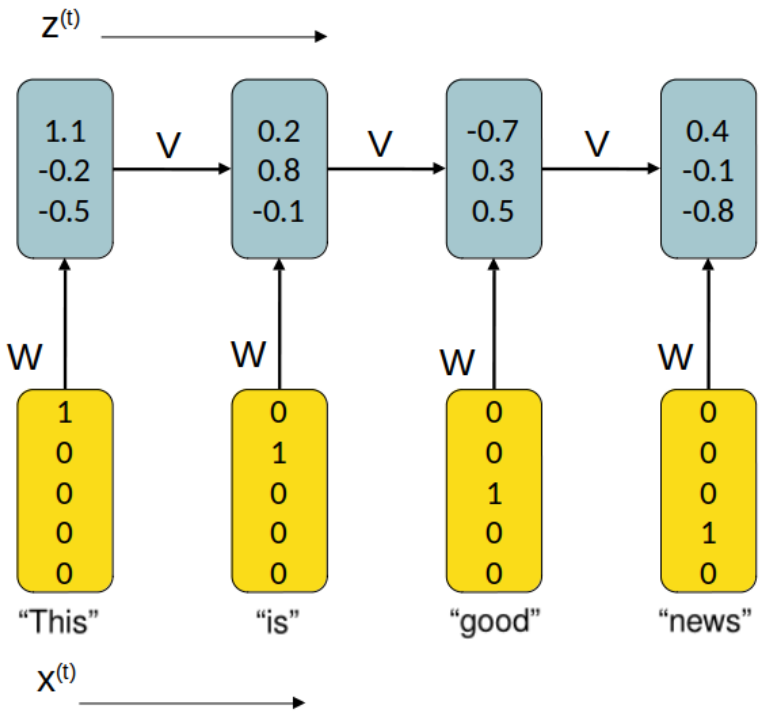
APPLICATION EXAMPLE - SENTIMENT ANALYSIS

- At $t = 2$, we feed the next word in the sentence.
- $\mathbf{z}^{[2]} = \sigma(\mathbf{V}^\top \mathbf{z}^{[1]} + \mathbf{W}^\top \mathbf{x}^{[2]} + \mathbf{b})$



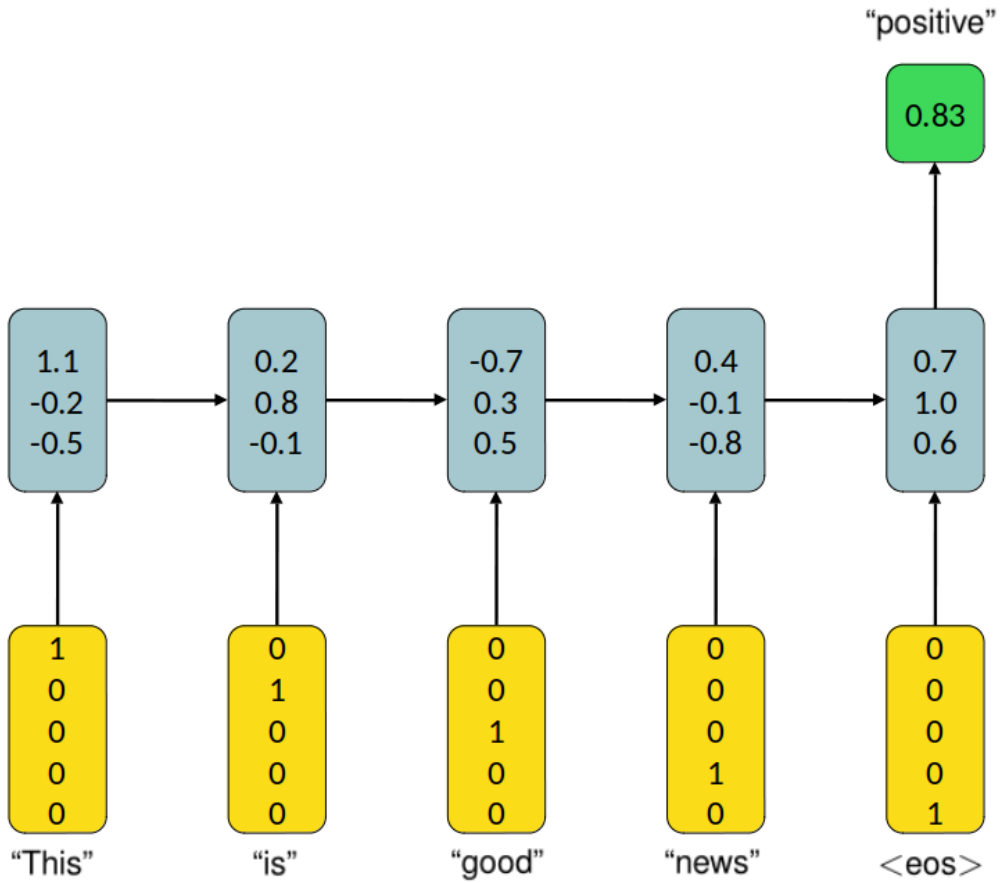
APPLICATION EXAMPLE - SENTIMENT ANALYSIS

- At $t = 3$, we feed the next word ("news") in the sentence.
- $\mathbf{z}^{[3]} = \sigma(\mathbf{V}^\top \mathbf{z}^{[2]} + \mathbf{W}^\top \mathbf{x}^{[3]} + \mathbf{b})$



APPLICATION EXAMPLE - SENTIMENT ANALYSIS

- Once the entire input sequence has been processed, the prediction of the network can be generated by feeding the activations of the final time-step to the output neuron(s).
- $f = \sigma(\mathbf{U}^\top \mathbf{z}^{[4]} + c)$, where c is the bias of the output neuron.



PARAMETER SHARING

- This way, the network can process the sentence one word at a time and the length of the network can vary based on the length of the sequence.
- It is important to note that no matter how long the input sequence is, the matrices **W** and **V** are the same in every time-step. This is another example of **parameter sharing**.
- Therefore, the number of weights in the network is independent of the length of the input sequence.

RNNS - USE CASE SPECIFIC ARCHITECTURES

RNNs are very versatile. They can be applied to a wide range of tasks.

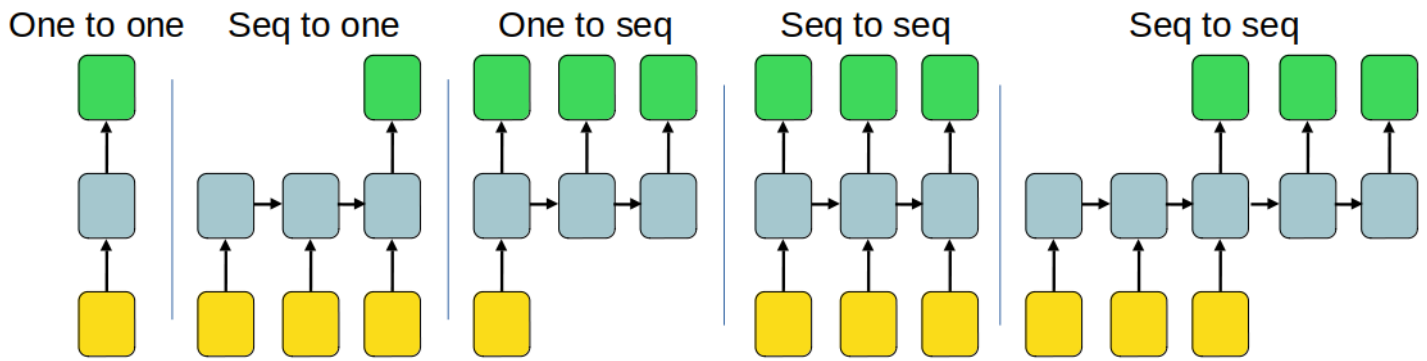


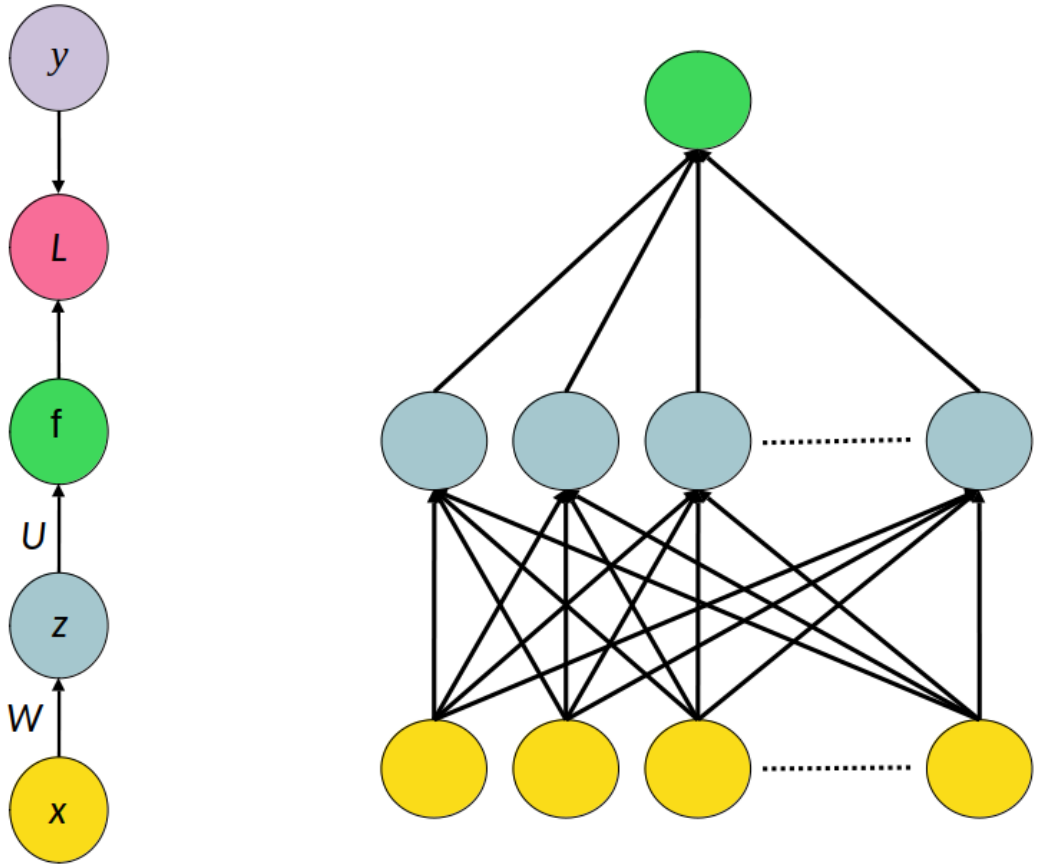
Figure: RNNs can be used in tasks that involve multiple inputs and/or multiple outputs.

Examples:

- Sequence-to-One: Sentiment analysis, document classification.
- One-to-Sequence: Image captioning.
- Sequence-to-Sequence: Language modelling, machine translation, time-series prediction.

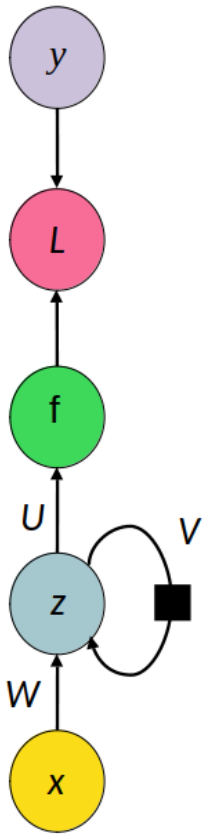
Computational Graph

RNNS - COMPUTATIONAL GRAPH



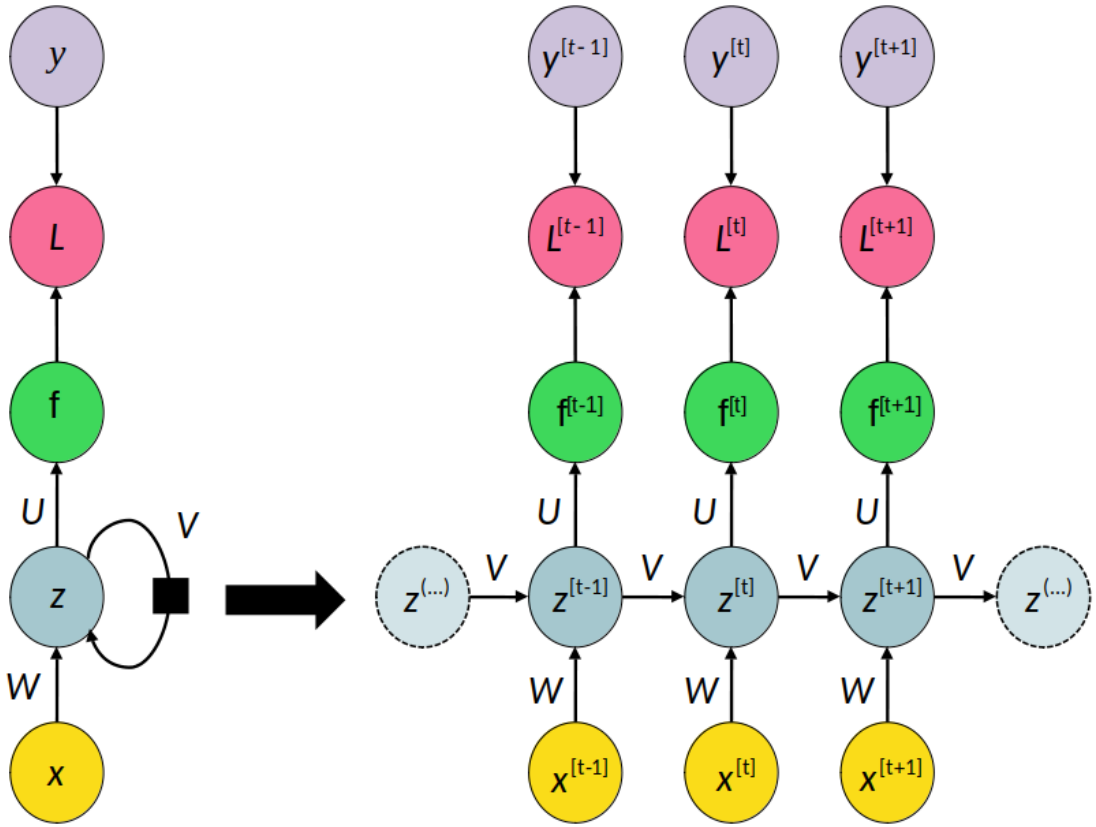
- On the left is an abstract representation of the computational graph for the network on the right. A loss function L measures how far each output f is from the corresponding training target y.

RNNS - COMPUTATIONAL GRAPH



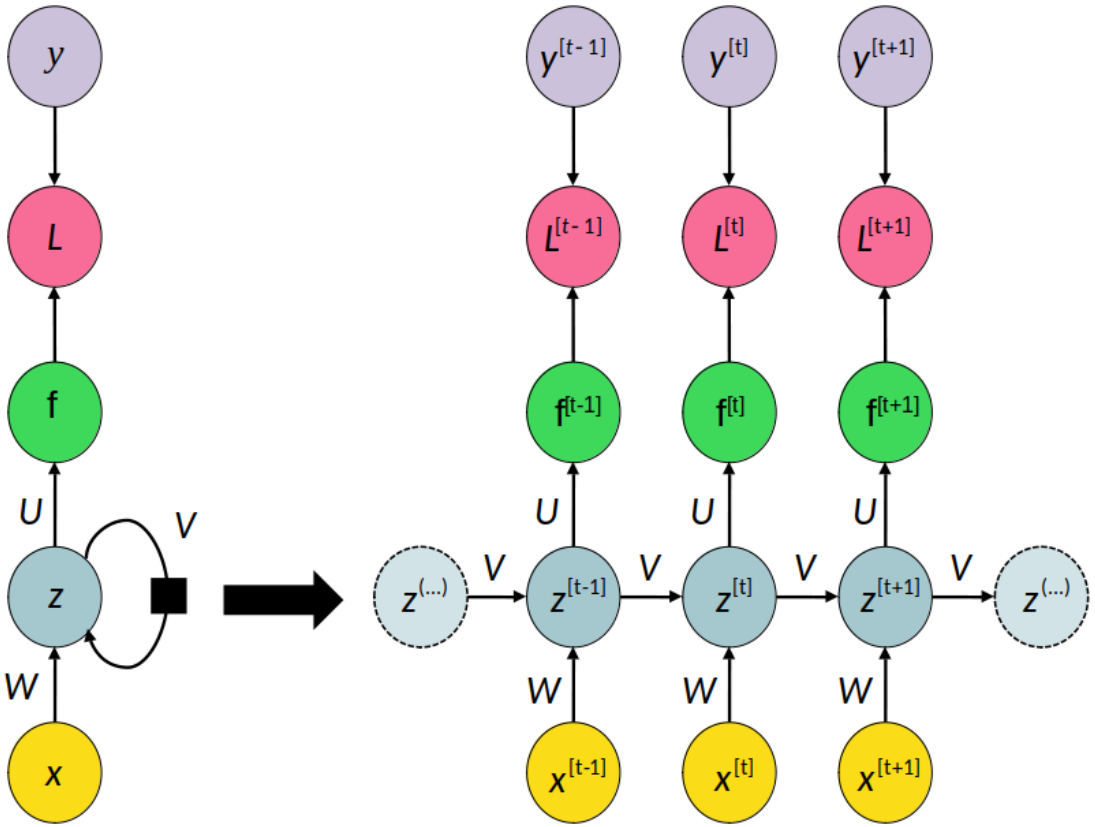
- A helpful way to think of an RNN is as multiple copies of the same network, each passing a message to a successor.
- RNNs are networks with loops, allowing information to persist.

RNNS - COMPUTATIONAL GRAPH



- Things might become more clear if we unfold the architecture.
- We call $\mathbf{z}^{[t]}$ the *state* of the system at time t.
- Tthe state contains information about the whole past sequence.

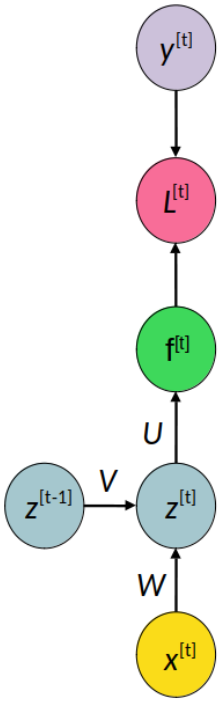
RNNS - COMPUTATIONAL GRAPH



- We went from

$$
\begin{aligned}
f &= \tau(c + \mathbf{U}^\top \sigma(\mathbf{b} + \mathbf{W}^\top \mathbf{x})) \text{ for the dense net, to} \\
f^{[t]} &= \tau(c + \mathbf{U}^\top \sigma(\mathbf{b} + \mathbf{V}^\top \mathbf{z}^{[t-1]} + \mathbf{W}^\top \mathbf{x}^{[t]})) \text{ for the RNN.}
\end{aligned}
$$

RNNS - COMPUTATIONAL GRAPH



- A potential computational graph for time-step t with

$$f^{[t]} = \tau(c + \mathbf{U}^\top \sigma(\mathbf{b} + \mathbf{V}^\top \mathbf{z}^{[t-1]} + \mathbf{W}^\top \mathbf{x}^{[t]}))$$

RECURRENT OUTPUT-HIDDEN CONNECTIONS

Recurrent connections do not need to map from hidden to hidden neurons!

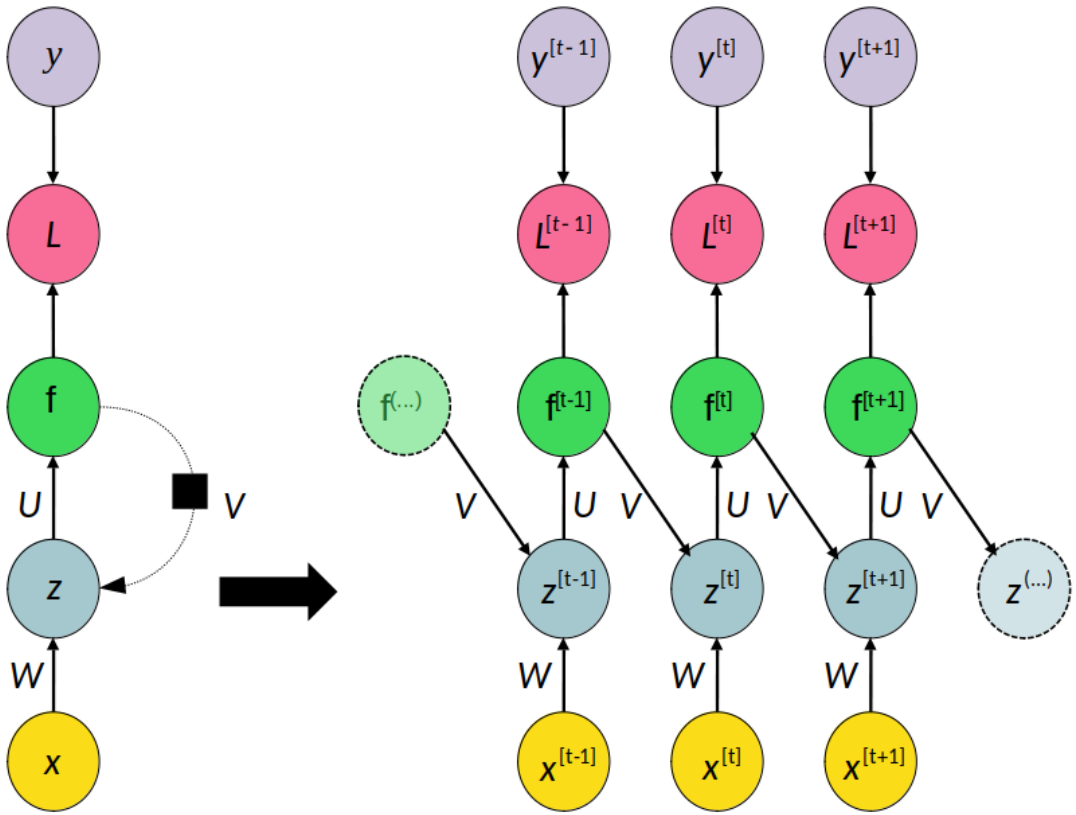


Figure: RNN with feedback connection from the output to the hidden layer. The RNN is only allowed to send f to future time points and, hence, $z^{[t-1]}$ is connected to $z^{[t]}$ only indirectly, via the predictions $f^{[t-1]}$.

SEQ-TO-ONE MAPPINGS

RNNs do not need to produce an output at each time step. Often only one output is produced after processing the whole sequence.

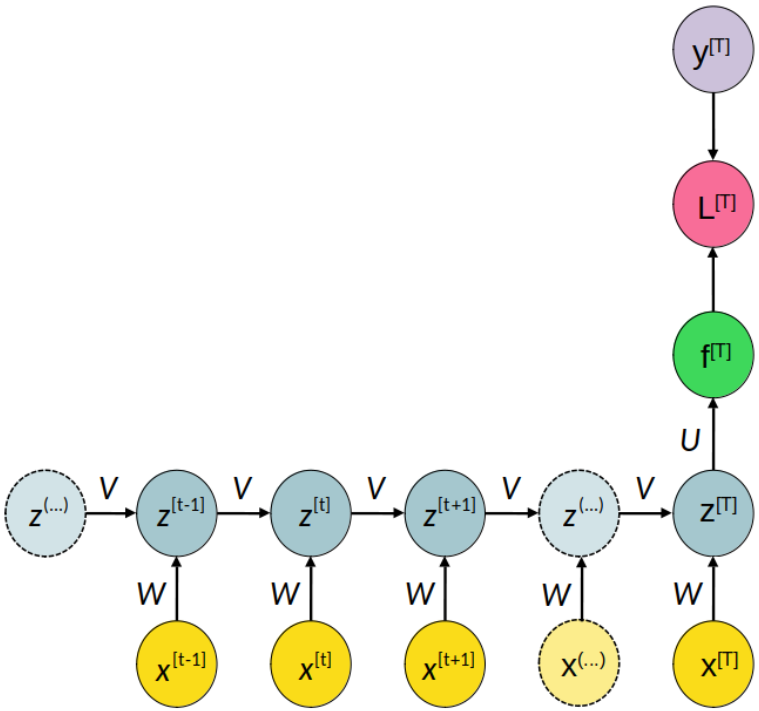


Figure: Time-unfolded recurrent neural network with a single output at the end of the sequence. Such a network can be used to summarize a sequence and produce a fixed size representation.