12-2 Period

Introduction to Recurrent Neural Network

Contents

- Sequence Data
- Recurrent Neural Network (RNN)
- Application of RNN

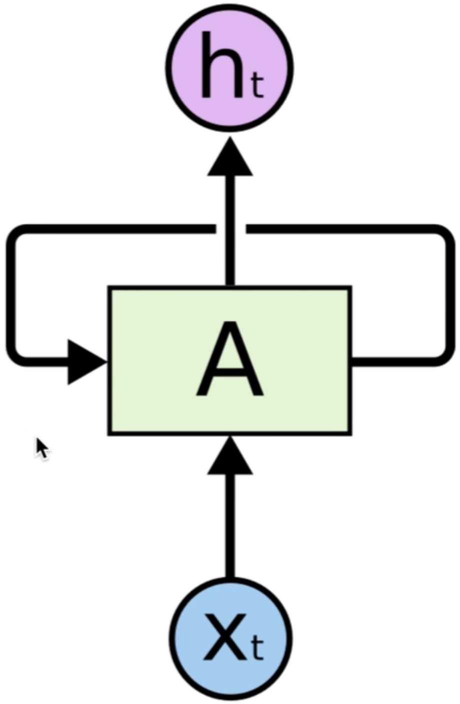
Sequence Data

Sequence data

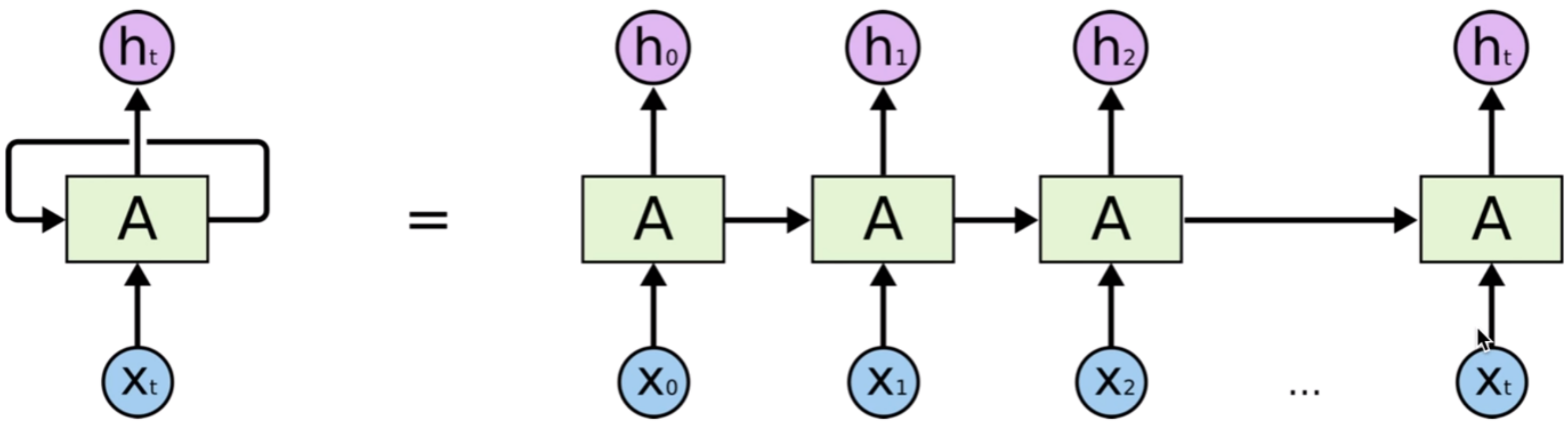
- We don't understand one word only
- We understand based on the previous words + this word. (time series)
- NN/CNN cannot do this

Sequence Data

- We don't understand one word only
- We understand based on the previous words + this word. (time series)
- NN/CNN cannot do this

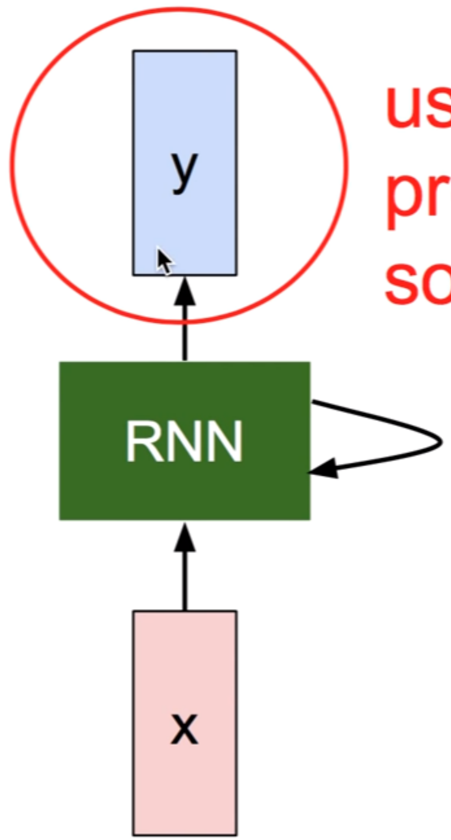


http://colah.github.io/posts/2015-08-Understanding-LSTMs/



h t
A
X t
=
h 0
h 1
h 2
h t
A
A
A
A
X 0
X 1
X 2
...
X t

Recurrent Neural Network



usually want to predict a vector at some time steps

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

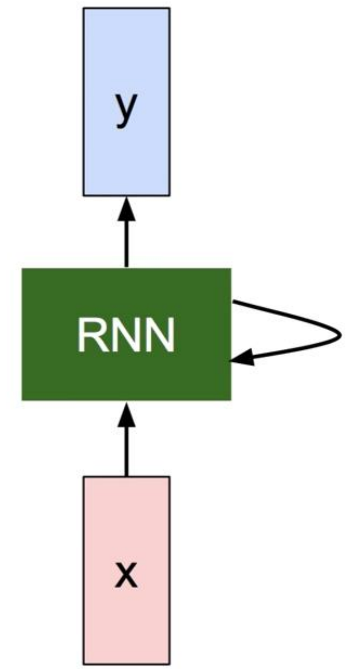
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

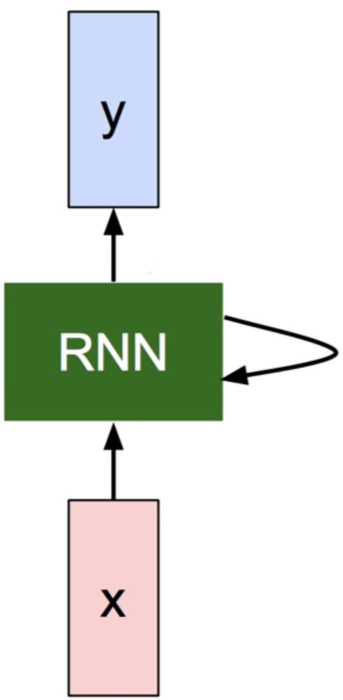
old state

input vector at some time step



(Vanilla) Recurrent Neural Network

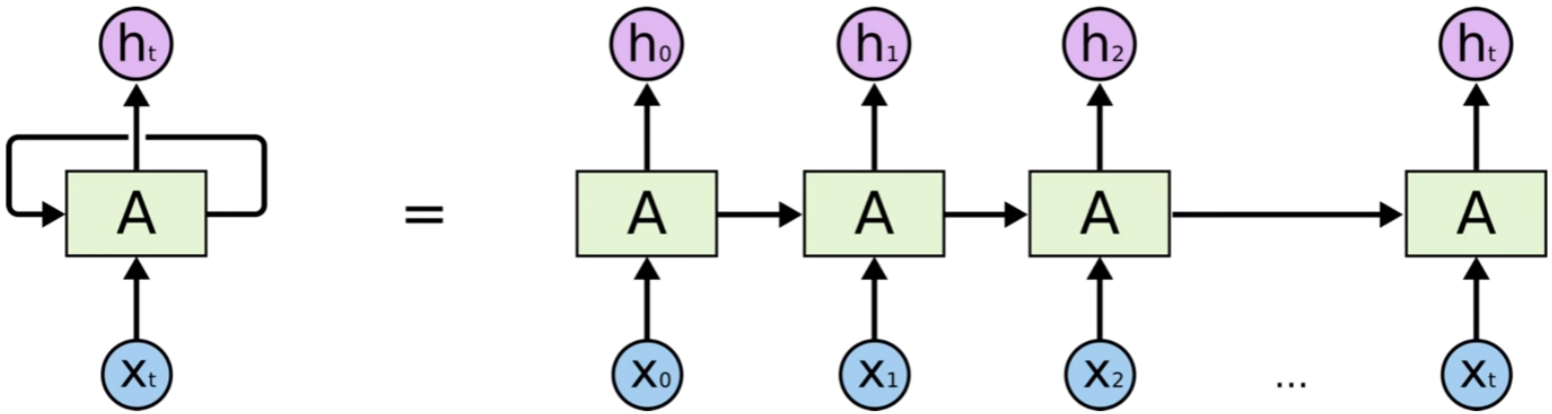
The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



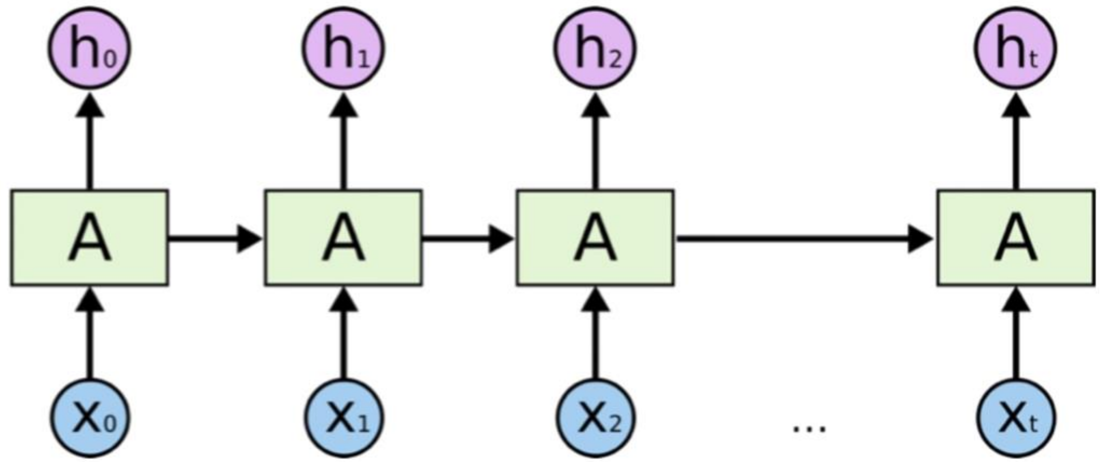
Notice: the same function and the same set of parameters are used at every time step.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

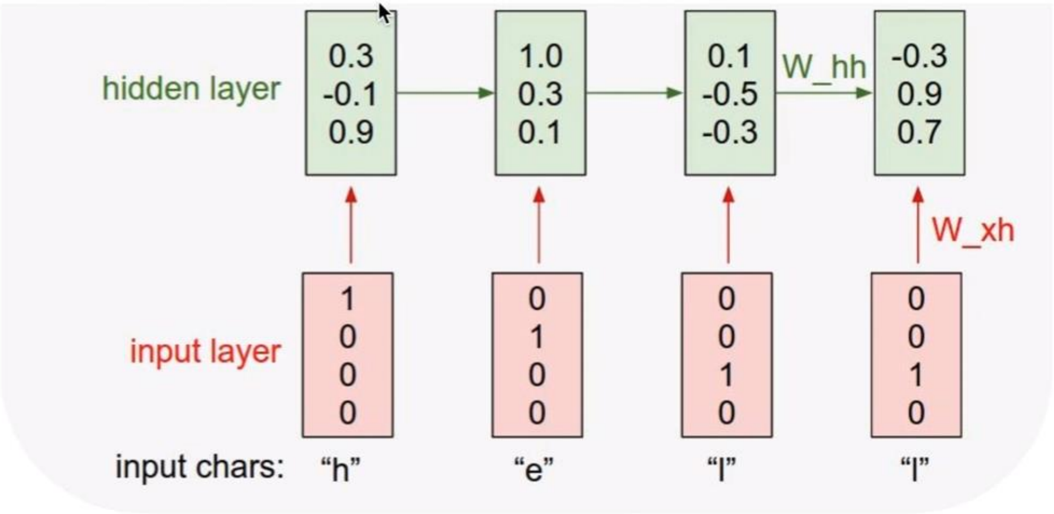


Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

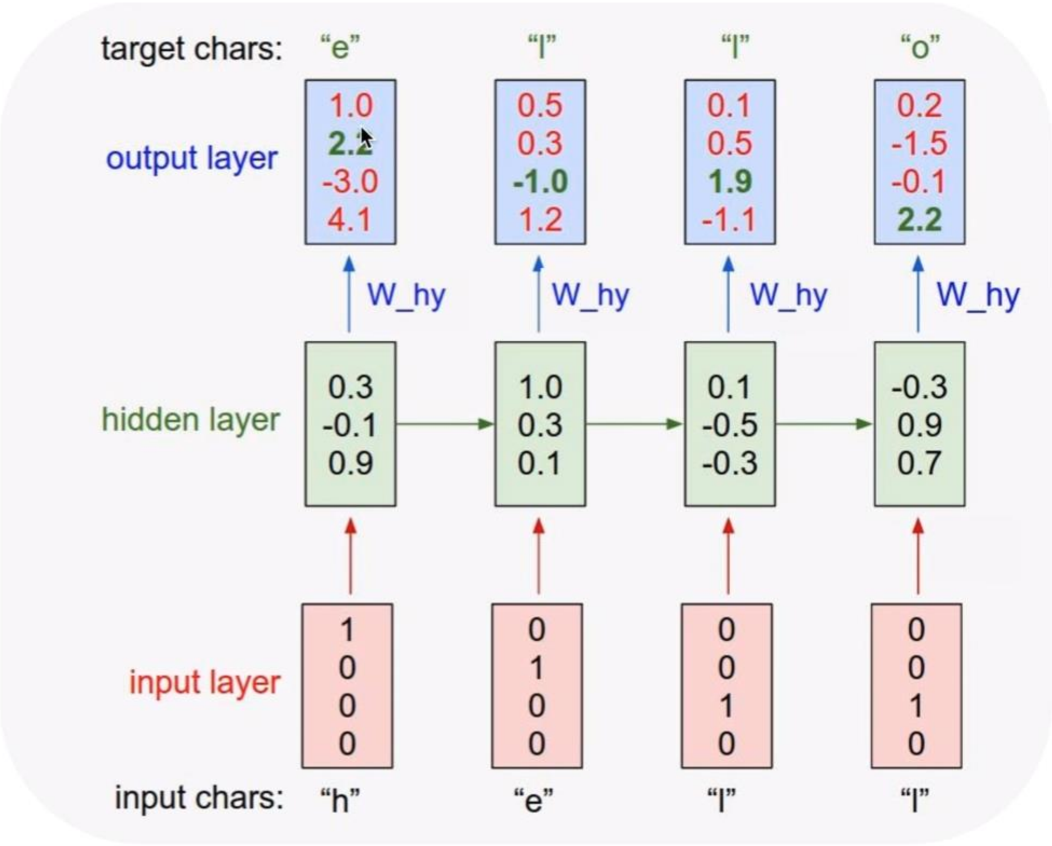


Character-level language model example

$$y_t = W_{hy} h_t$$

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



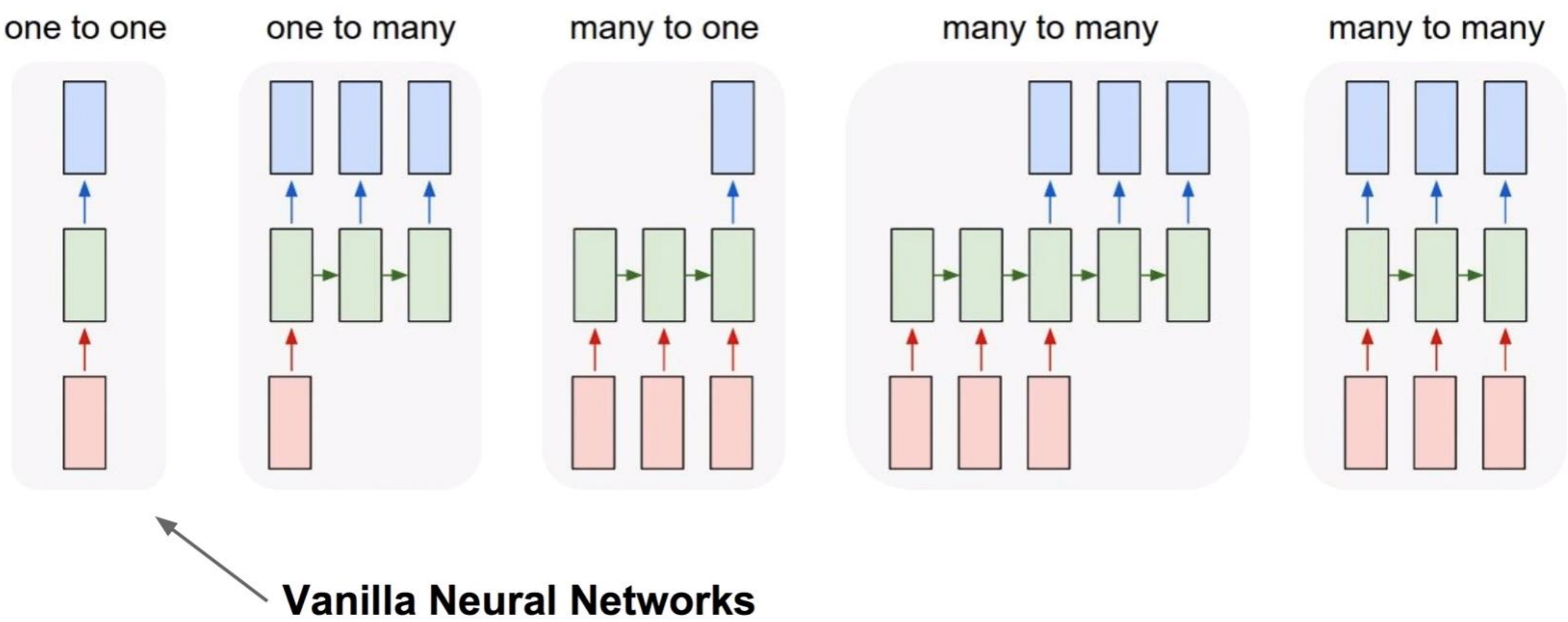
RNN applications

https://github.com/TensorFlowKR/awesome_tensorflow_implementations

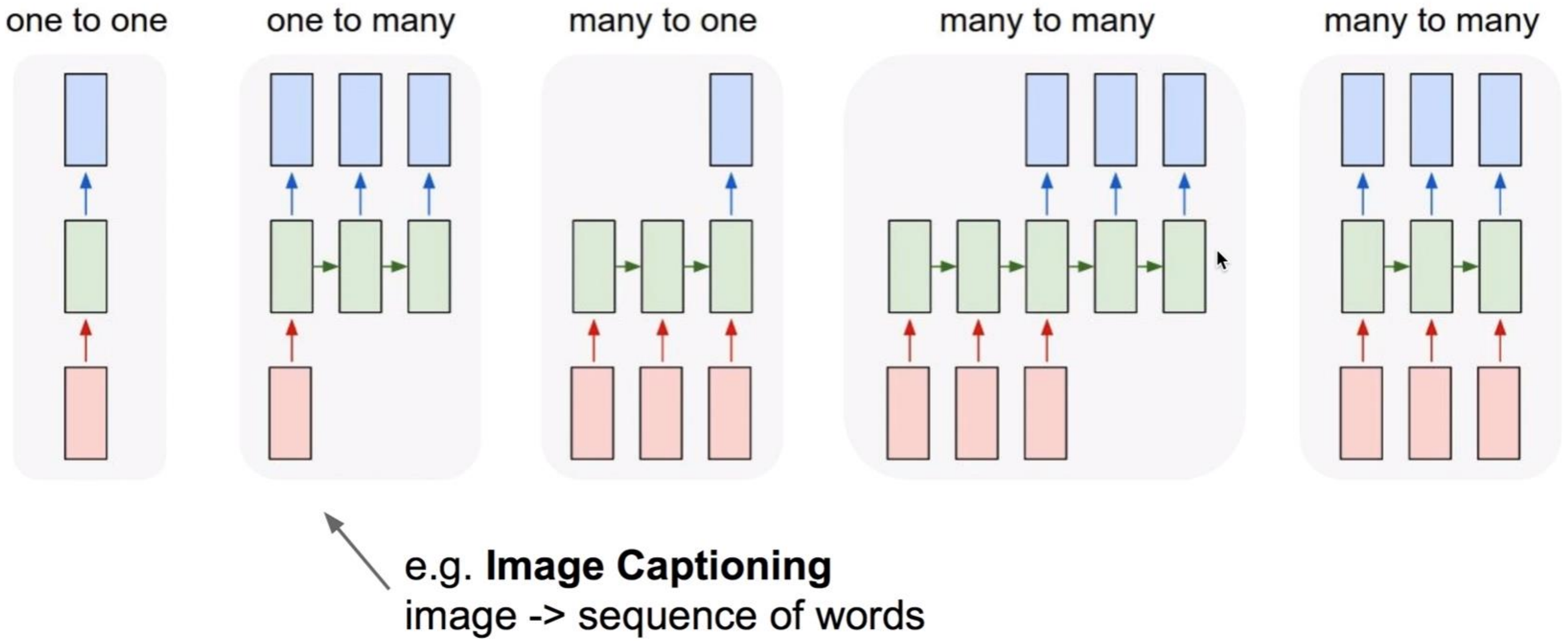
- Language Modeling
- Speech Recognition
- Machine Translation
- Conversation Modeling/Question Answering
- Image/Video Captioning
- Image/Music/Dance Generation

http://jiwonkim.org/awesome-rnn/

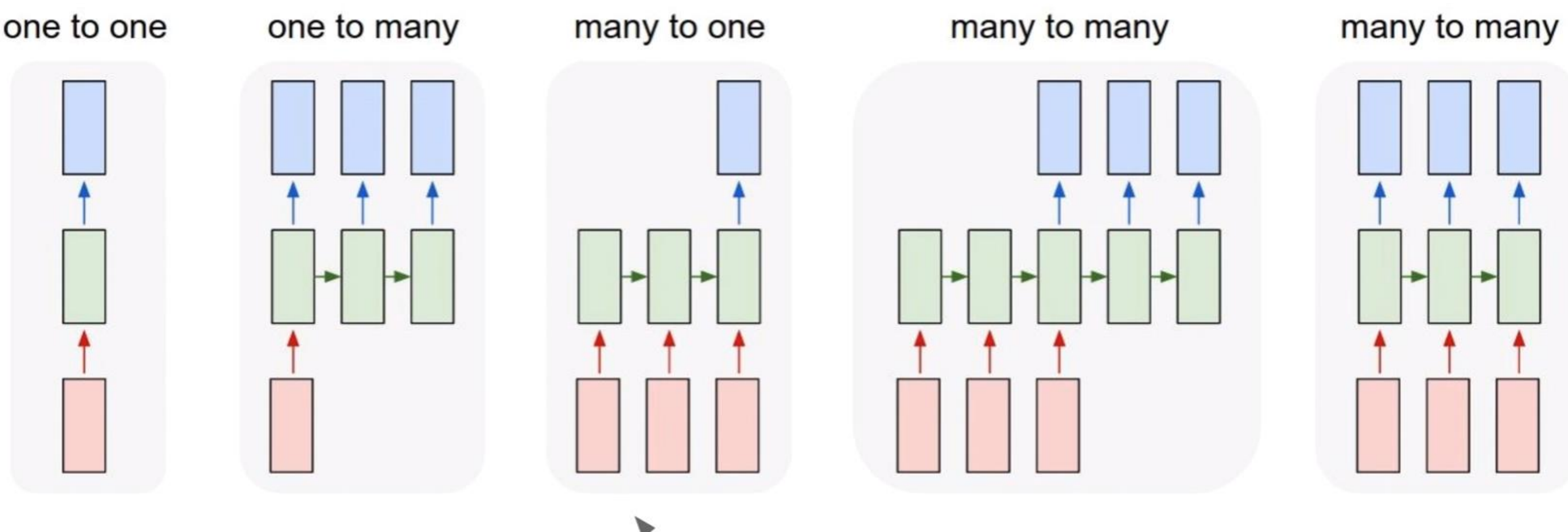
Recurrent Networks offer a lot of flexibility:



Recurrent Networks offer a lot of flexibility:

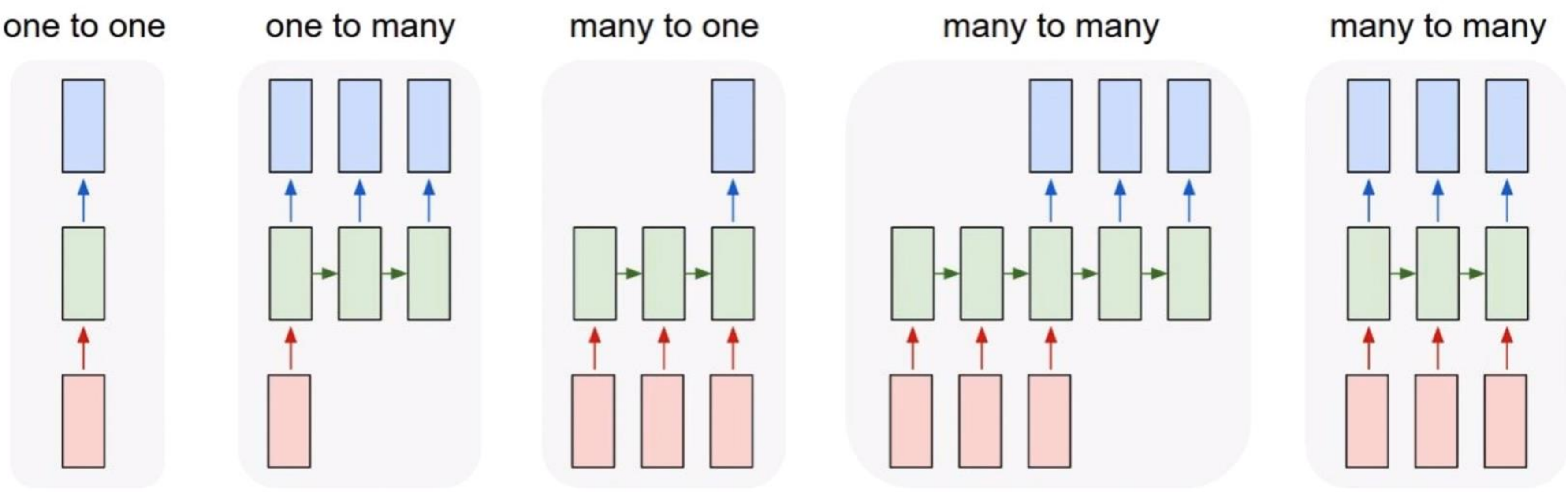


Recurrent Networks offer a lot of flexibility:



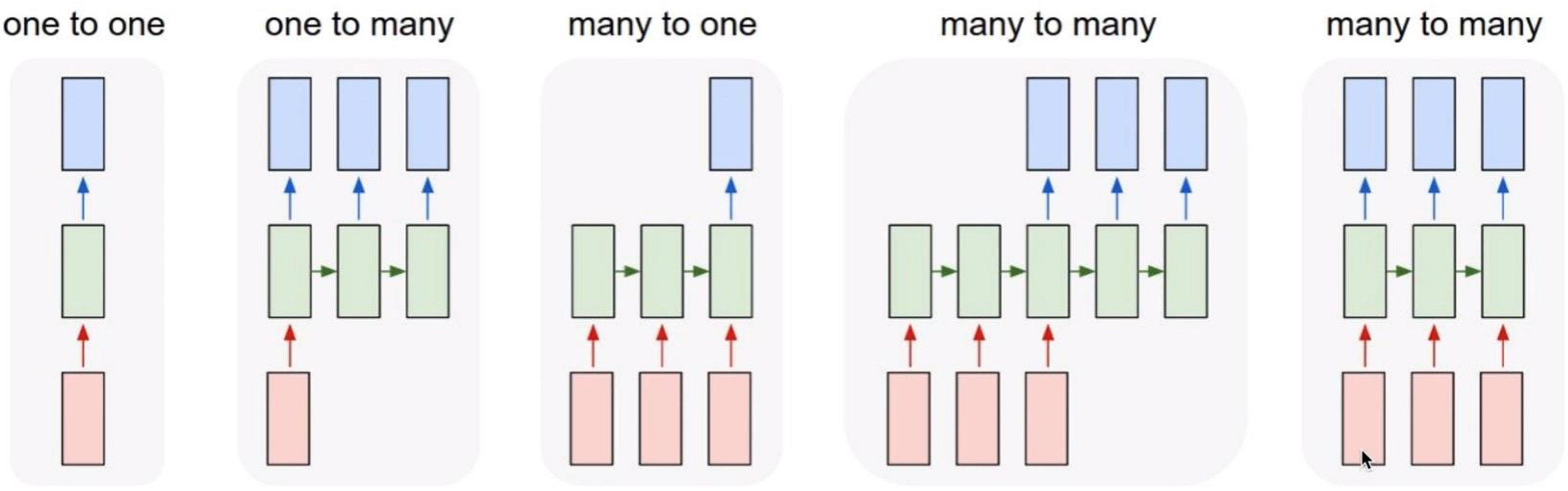
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Networks offer a lot of flexibility:



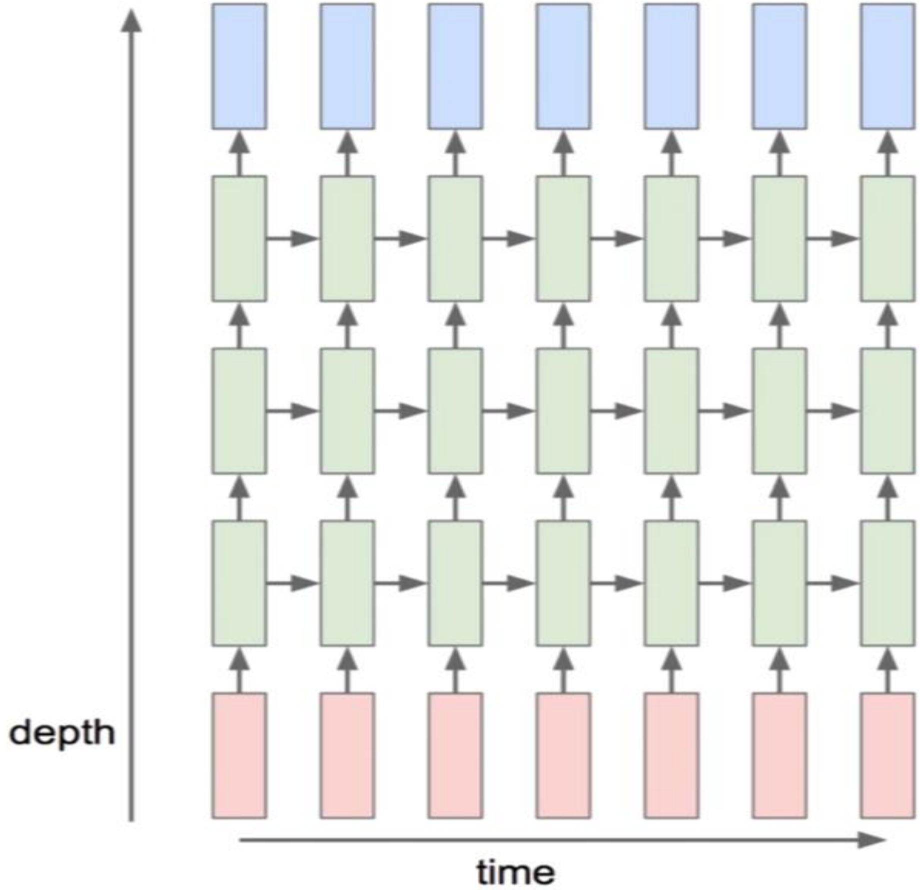
e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Networks offer a lot of flexibility:



e.g. **Video classification on frame level**

Multi-Layer RNN



Training RNNs is challenging

- Several advanced models
  - Long Short Term Memory (LSTM)
  - GRU by Cho et al. 2014

Paper: K. Cho, D. Bahdanau, etc, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Proceedings on Conference on Empirical Methods in Natural Language Processing 2014.

2 RNN Encoder–Decoder

2.1 Preliminary: Recurrent Neural Networks

A recurrent neural network (RNN) is a neural network that consists of a hidden state $\mathbf{h}$ and an optional output $\mathbf{y}$ which operates on a variable-length sequence $\mathbf{x} = (x_1, \ldots, x_T)$. At each time step t, the hidden state $\mathbf{h}_{\langle t \rangle}$ of the RNN is updated by

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right), \tag{1}$$

where f is a non-linear activation function. f may be as simple as an element-wise logistic sigmoid function and as complex as a long short-term memory (LSTM) unit (Hochreiter and Schmidhuber, 1997).

An RNN can learn a probability distribution over a sequence by being trained to predict the next symbol in a sequence. In that case, the output at each timestep t is the conditional distribution $p(x_t \mid x_{t-1}, \ldots, x_1)$. For example, a multinomial distribution (1-of-K coding) can be output using a softmax activation function

$$p(x_{t,j} = 1 \mid x_{t-1}, \ldots, x_1) = \frac{\exp\left(\mathbf{w}_j \mathbf{h}_{\langle t \rangle}\right)}{\sum_{j'=1}^{K} \exp\left(\mathbf{w}_{j'} \mathbf{h}_{\langle t \rangle}\right)}, \tag{2}$$

for all possible symbols $j = 1, \ldots, K$, where $\mathbf{w}_j$ are the rows of a weight matrix $\mathbf{W}$. By combining these probabilities, we can compute the probability of the sequence $\mathbf{x}$ using

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \ldots, x_1). \tag{3}$$

From this learned distribution, it is straightforward to sample a new sequence by iteratively sampling a symbol at each time step.

2.2 RNN Encoder–Decoder

In this paper, we propose a novel neural network architecture that learns to *encode* a variable-length sequence into a fixed-length vector representation and to *decode* a given fixed-length vector representation back into a variable-length sequence. From a probabilistic perspective, this new model is a general method to learn the conditional distribution over a variable-length sequence conditioned on yet another variable-length sequence, e.g. $p(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_T)$, where one should note that the input and output sequence lengths T and T' may differ.

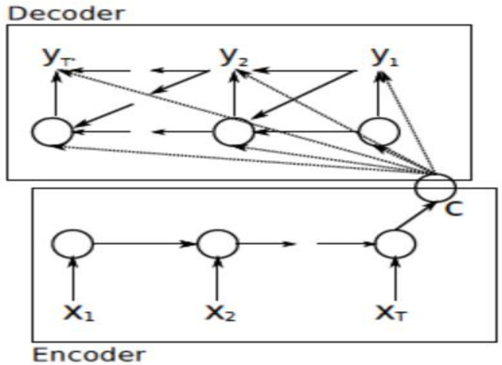


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

The encoder is an RNN that reads each symbol of an input sequence $\mathbf{x}$ sequentially. As it reads each symbol, the hidden state of the RNN changes according to Eq. (1). After reading the end of the sequence (marked by an end-of-sequence symbol), the hidden state of the RNN is a summary $\mathbf{c}$ of the whole input sequence.

The decoder of the proposed model is another RNN which is trained to *generate* the output sequence by predicting the next symbol y_t given the hidden state $\mathbf{h}_{\langle t \rangle}$. However, unlike the RNN described in Sec. 2.1, both y_t and $\mathbf{h}_{\langle t \rangle}$ are also conditioned on y_{t-1} and on the summary $\mathbf{c}$ of the input sequence. Hence, the hidden state of the decoder at time t is computed by,

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c}\right),$$

and similarly, the conditional distribution of the next symbol is

$$P(y_t | y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g\left(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c}\right).$$

for given activation functions f and g (the latter must produce valid probabilities, e.g. with a softmax).

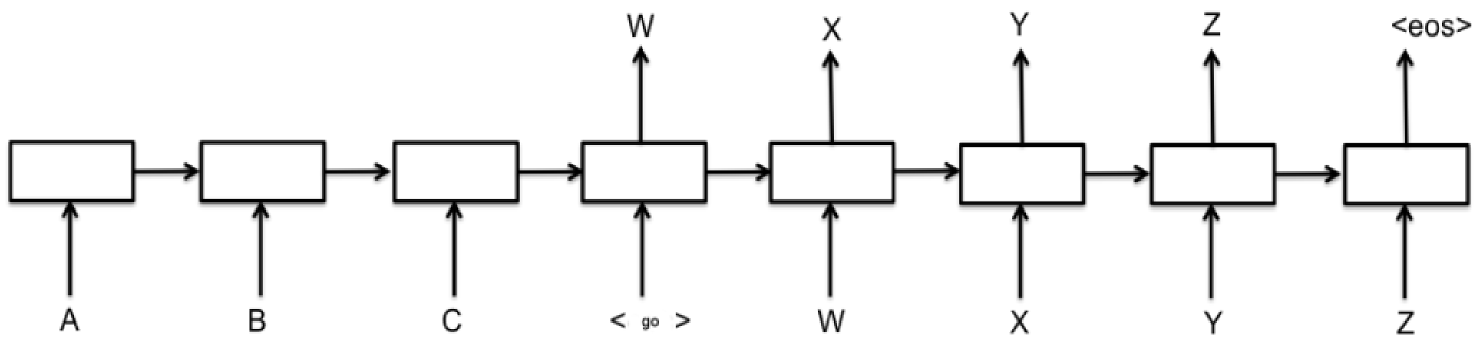
See Fig. 1 for a graphical depiction of the proposed model architecture.

The two components of the proposed *RNN Encoder–Decoder* are jointly trained to maximize the conditional log-likelihood

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{y}_n \mid \mathbf{x}_n), \tag{4}$$

Sequence-to-Sequence Models

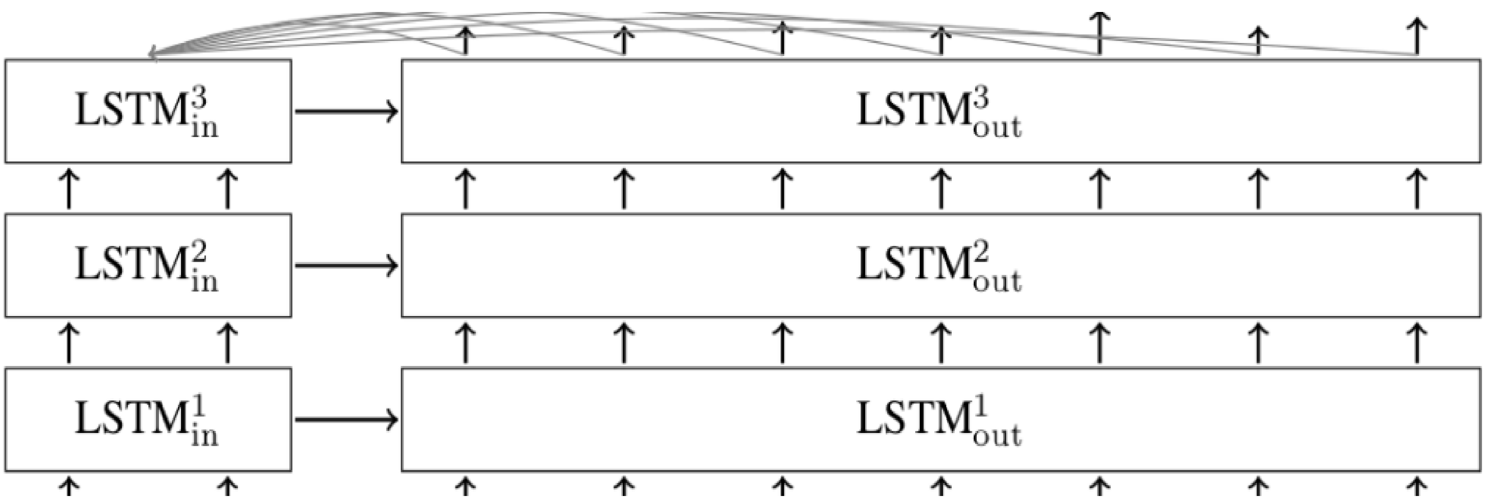
A basic sequence-to-sequence model, as introduced in Cho et al., 2014 (pdf), consists of two recurrent neural networks (RNNs): an *encoder* that processes the input and a *decoder* that generates the output. This basic architecture is depicted below.



Each box in the picture above represents a cell of the RNN, most commonly a GRU cell or an LSTM cell (see the RNN Tutorial for an explanation of those). Encoder and decoder can share weights or, as is more common, use a different set of parameters. Multi-layer cells have been successfully used in sequence-to-sequence models too, e.g. for translation Sutskever et al., 2014 (pdf).

Sequence-to-Sequence Models

In the basic model depicted above, every input has to be encoded into a fixed-size state vector, as that is the only thing passed to the decoder. To allow the decoder more direct access to the input, an *attention* mechanism was introduced in Bahdanau et al., 2014 (pdf). We will not go into the details of the attention mechanism (see the paper); suffice it to say that it allows the decoder to peek into the input at every decoding step. A multi-layer sequence-to-sequence network with LSTM cells and attention mechanism in the decoder looks like this.



Tensorflow Examples

- **Hello Example**
- **Learning Simple Character Sequence**
- **RNN Long Charter Sequence**
- **RNN Stock Prediction**
- **Chat Bot**