Introduction to Recurrent Neural Networks (RNN)

Yulia Newton, Ph.D.

CS156, Introduction to Artificial Intelligence

San Jose State University

Spring 2021

What are RNNs?

- Recurrent neural networks (RNN) are artificial neural networks which form directed graph structures
- Good at processing variable length structures
  - E.g. speech recognition, natural language processing (NLP), handwriting recognition, sentiment analysis, machine translation (e.g. English to German), video activity recognition, image labeling, name entity recognition, etc.
- The idea is to take advantage of sequential information

- First appeared in the research literature in 1980's but gained popularity as a good model for natural language processing problems due to advances in computing power and methodology

Why are traditional ANNs not a good idea for sequential information?

- Traditional ANNs assume that all the inputs are independent of each other
- However, that is not the case for a lot of sequence information
  - E.g. sentence structure in written or spoken text
    - Prediction of the next word in the sentence is affected by words that came before
  - E.g. autocorrelated events
- “Recurrent” refers to performing the same task for each element of the sequence
- RNNs have a “memory” about what has been calculated up to this point

Sequential data requires special consideration

- Sequences and time series data require “remembering” the history of the previous inputs
  - Generally we mean recent history but depends on the problem

Example of an RNN

- Typically RNNs only keep the memory of a few sequence elements back
  - While theoretically possible to apply to any length of sequence, practical computing/complexity limitations apply

x_t - input at time point t
s_t - hidden state at time point t
o_t - output at time point t

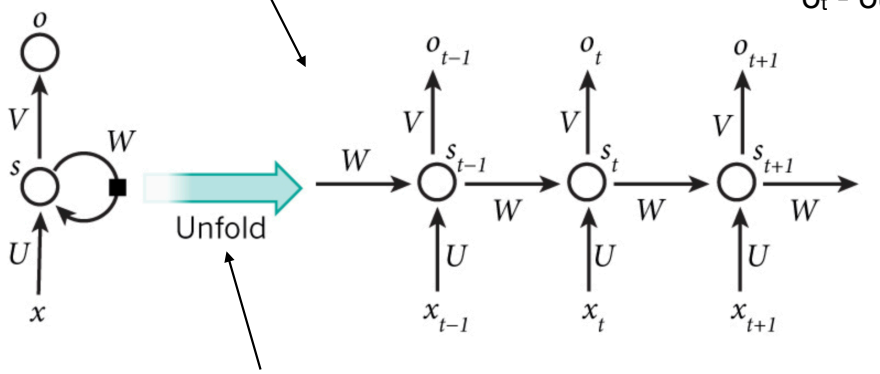
s_t is the "memory" node and is computed using an activation function (for non-linearity) based on the input x_t and weights of the previous time point t-1

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

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

ReLU or tanh are most commonly used activation functions here

Memory of this RNN is 3 elements long



Unrolling the RNN into FFANN

Output at time t is the prediction based on the memory about previous elements; e.g. if predicting the next word in the sentence then output is a vector of probabilities across the whole vocabulary

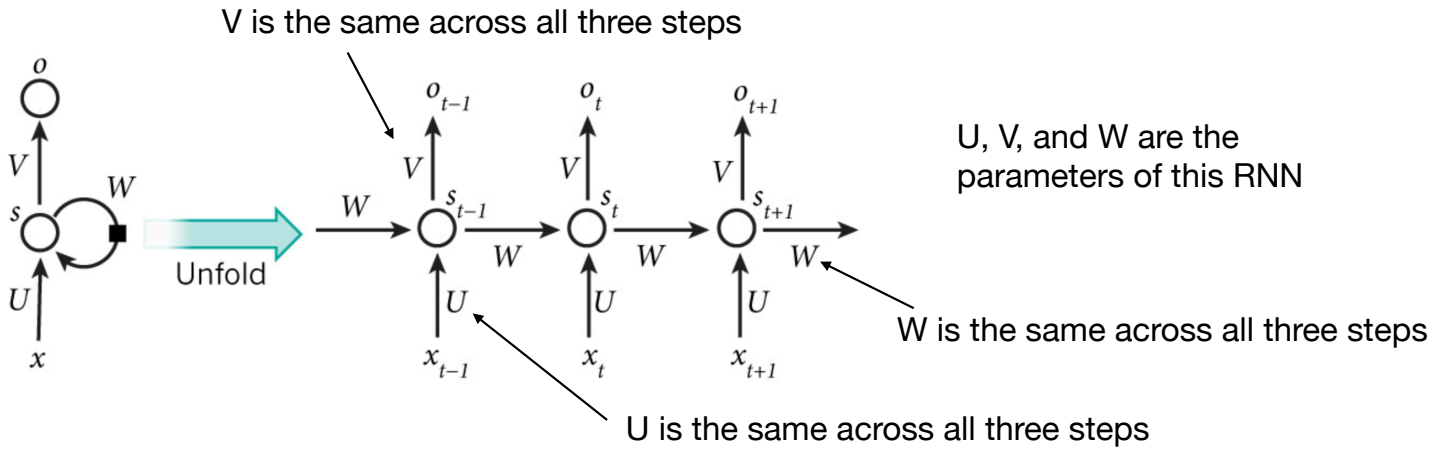
$$o_t = softmax\,(Vs_t)$$

"Neural Architectures for Relation Extraction Within and Across Sentence Boundaries in Natural Language Text", LeCun, Bengio, and G. Hinton 2015

RNN parameters

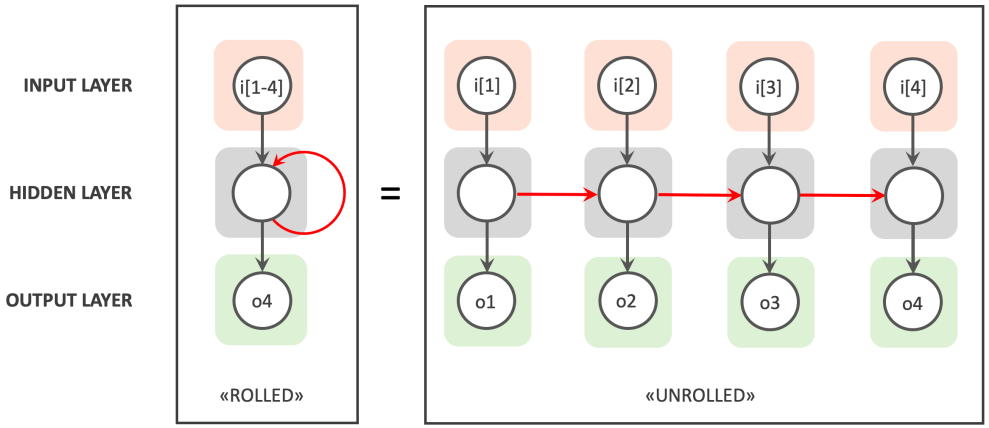
- In a traditional feed forward ANN parameters of each layer are different
- Parameters in each RNN step are the same
  - Perform the same task on different elements with the same parameters

RNN cells are parameterized by weight matrices U, V, W



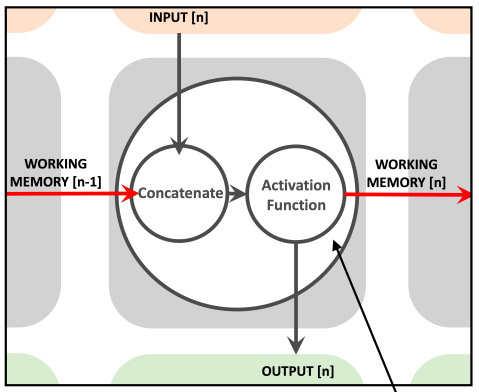
"Neural Architectures for Relation Extraction Within and Across Sentence Boundaries in Natural Language Text", LeCun, Bengio, and G. Hinton 2015

What is in the hidden state of an RNN cell?



https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

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



The input of the activation function is used to compute the output and is forwarded on to the next cell

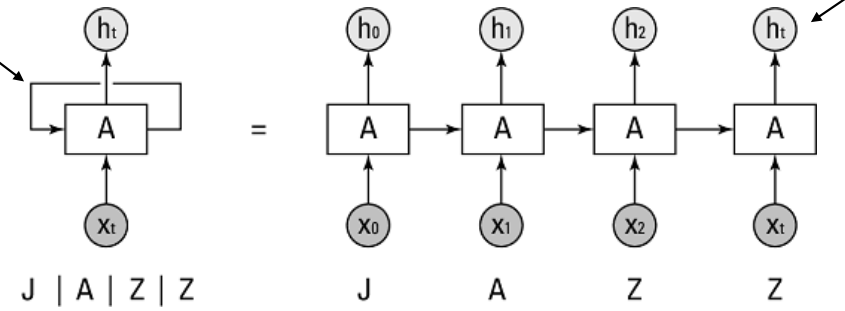
Outputs

- While each step can compute the output, it is not always necessary
- Depending on the task one may not care about intermediate outputs
- E.g. in sentiment analysis you only care about the final sentiment of the sentence
  - Sentiment analysis - NLP tasks that aim to understand the states of the conveyed information
    - E.g. is this customer review positive or negative?
    - Sometimes called “emotion AI” or “opinion mining”

Example: RNN that processes the word JAZZ

We can think about self-connection as conceptually a recursion

Sometimes you see h used to indicate output of an RNN cell

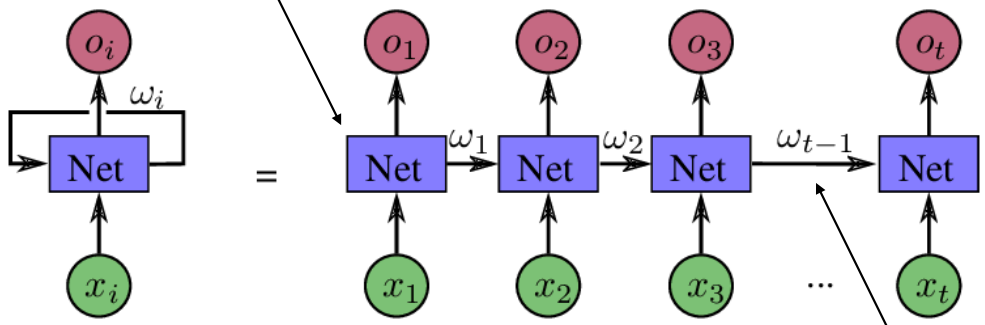


www.dummies.com

RNN cells are in fact neural networks

Each cell of RNN is typically a deep neural network

The state of each RNN cell is the sum of the weights learned from the inputs at this step and the weights from the previous step

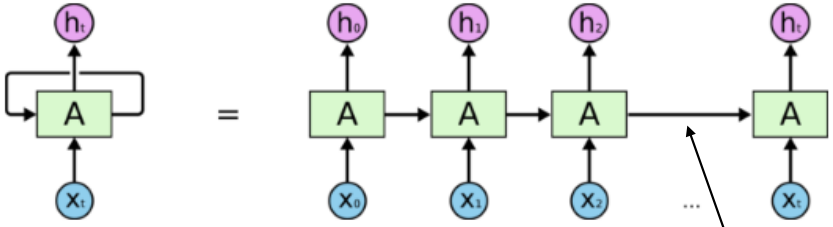


"Deep-learning in Mobile Robotics - from Perception to Control Systems: A Survey on Why and Why not" Lei Tai & Ming Liu 2016

Into each network at time point t goes the input at time point t and the weights from the previous step

RNNs are chained feed forward ANNs

RNNs can be thought of as many sequential copies of FFANNs chained together



An unrolled recurrent neural network.

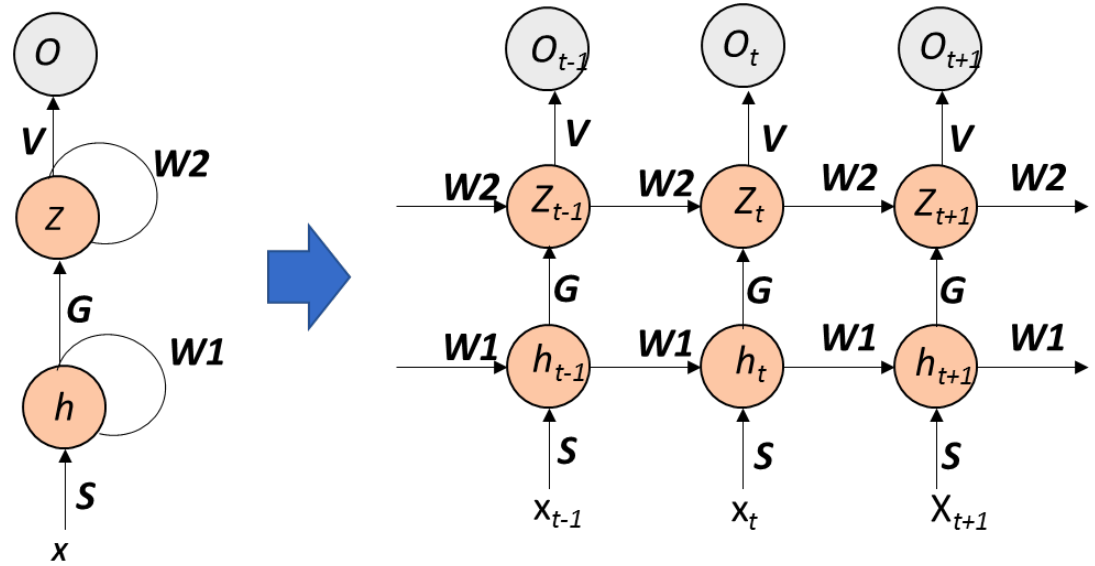
towardsdatascience.com

The state of each RNN cell is the sum of the weights learned from the inputs at this step and the weights from the previous step

Into each cell at time point t goes the input at time point t and the weights from the previous step

What is an RNN layer?

- A single RNA layer is a single rolled up RNN cell
  - The cell unrolls into a number of steps



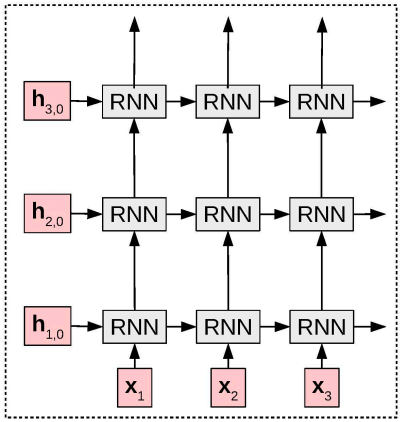
a) 2-layer Recurrent Neural Network (RNN)

b) Unfolded 2-layer Recurrent Neural Network (RNN)

"R Deep Learning Cookbook" PacktPublishing

Stacking RNNs

- Depending on your problem and the most appropriate ANN architecture for it, you may want/need to stack RNNs
- You can stack RNN cells horizontally or vertically to create more complex RNN structures

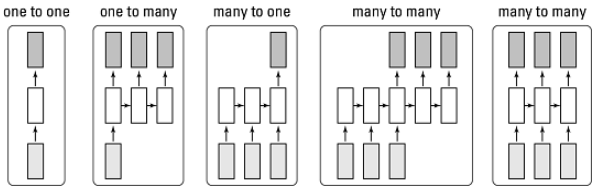


https://yiyibooks.cn/__src__/wizard/nmt-tut-neubig-2017_20180721165003_deleted/ch6.html

Various RNN input-output paradigms

- One-to-many
  - There is a single input and a sequence of outputs
  - E.g. automatic captioning of images
- Many-to-one
  - The input is a sequence and there is a single output
  - Classic example of RNN
    - E.g. sentiment analysis
- Many-to-many
  - The input is a sequence and the output is also a sequence
  - Often referred to as seq2seq
  - E.g. language sentence translation

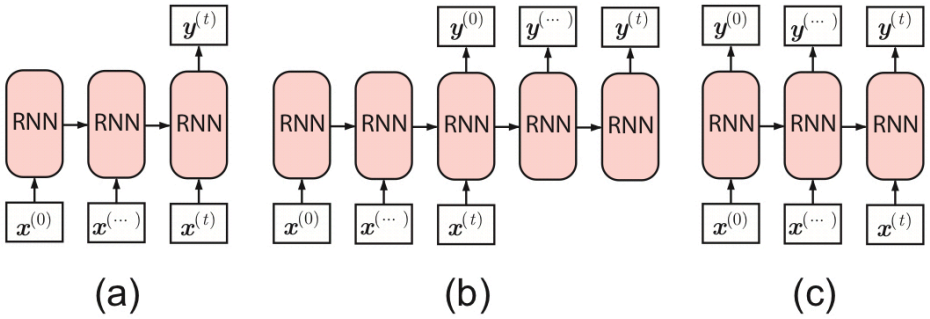
One-to-one is the traditional FFANN



www.dummies.com

Another look at RNN input-output paradigms

Many-to-one: Many-to-many: Many-to-many:



https://blog.acolyer.org/2019/02/25/understanding-hidden-memories-of-recurrent-neural-networks

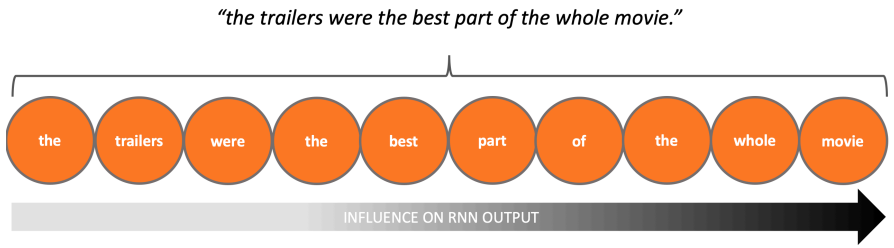
How are RNNs trained?

- RNN weight optimization is usually performed using gradient descent algorithms
- The RNN is unrolled during the learning process
- Back-propagation through time (BPTT) is a technique most often used in RNN learning
  - The algorithm sums the weight updates of accumulated errors for each element of the sequence
  - Then updates the weights using that sum of errors
  - Can encounter vanishing or exploding weights problem, so often used in combination with other algorithms

- Genetic algorithms have also been successfully applied to learning RNN parameters
  - Search heuristic approach, inspired by Charles Darwin's theory of evolution, often utilized in optimization problems

Limitations of RNNs

- While in theory RNNs can maintain memory of large sequences, memory of standard RNNs struggles to remember long-ago history
- Vanishing gradient problem - loss of information as updating weights through back-propagation using gradient descent in really deep ANNs or RNNs with large number of cells
  - RNNs are especially susceptible to vanishing and exploding gradient issues because multiple RNN cells use the same parameter



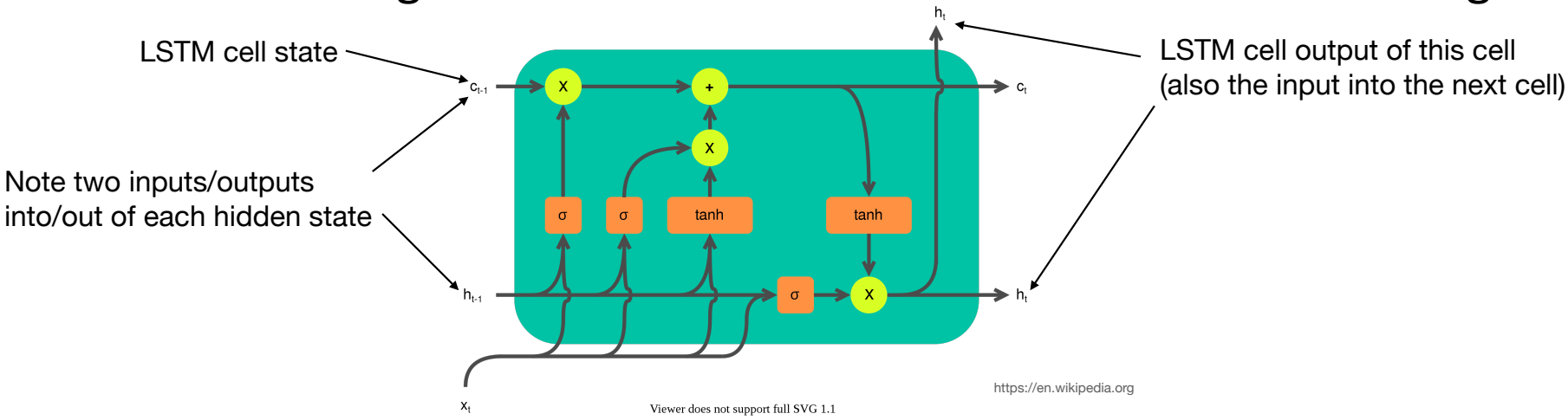
https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

What are LSTMs?

- Long Short-Term Memory (LSTM) are a type of recurrent neural networks
- Solve the issue of vanishing gradient descent
  - Prevents back-propagated error from vanishing or exploding
  - Error can propagate backwards through unlimited number of layers
- Most commonly used type of RNN at the moment
  - Differ from standard RNNs in the hidden state

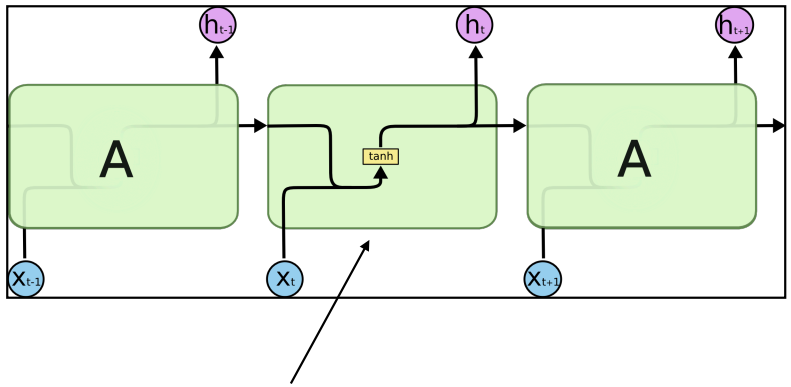
The main idea behind LSTM

- Cell state can be regulated by minor linear interactions
- Information is added to or removed from the cell state through gates
  - Gates are composed of sigmoid neural network layer and a point-wise multiplication
    - Sigmoid layer outputs a value in interval [0, 1]
    - This value regulates how much information should be let though



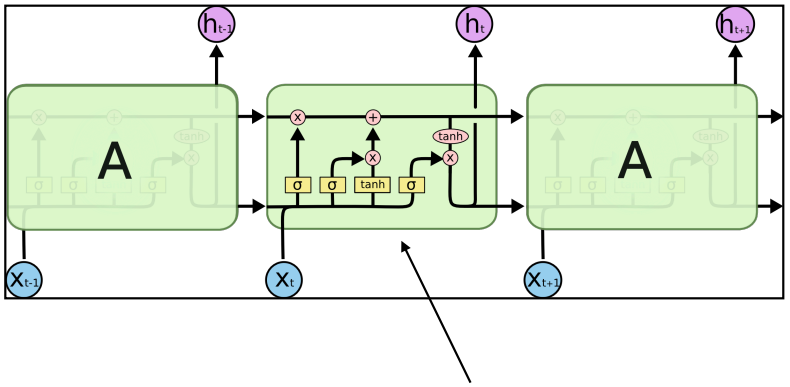
Standard RNN vs. LSTM

Standard RNN:

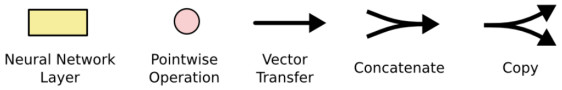


RNN hidden state

LSTM:

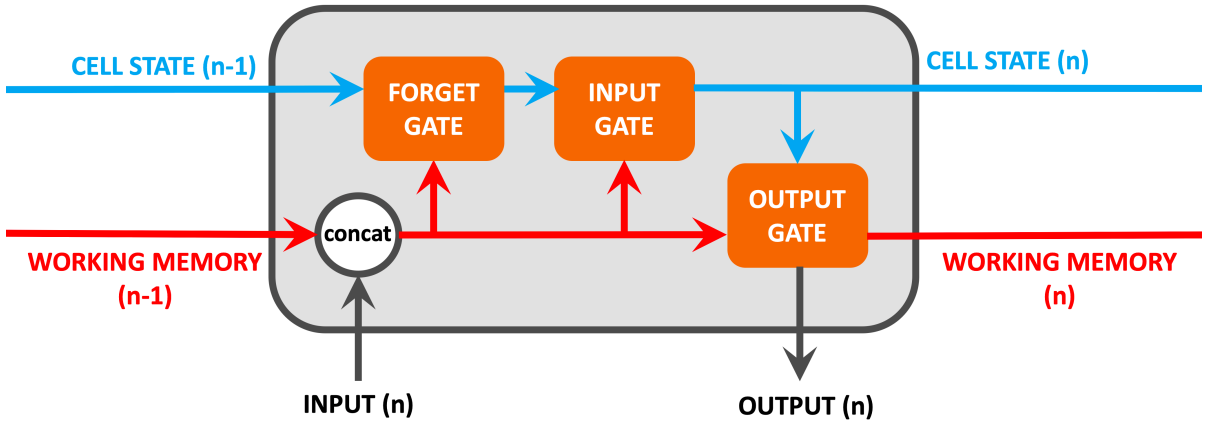


LSTM cell (also called "memory" cell)



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

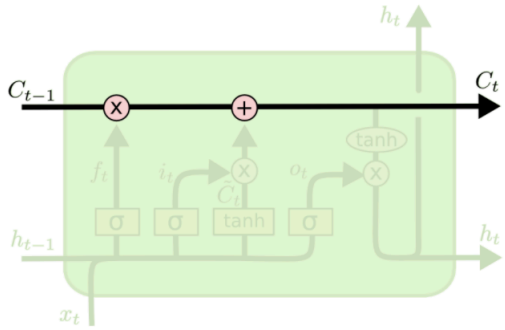
LSTM cells maintain both cell state and cell memory by utilizing various gates



https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

Step by step look at the hidden state of LSTM (*cell state*)

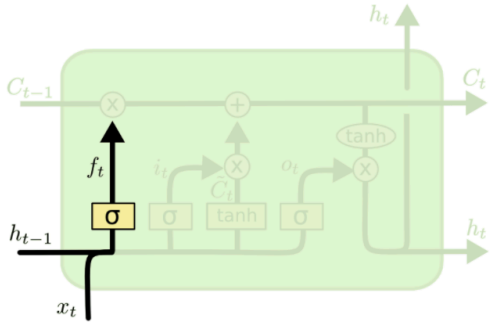
- Each LSTM cell has a state
- A state is the value that can be tweaked to carry on from the previous cell completely, in part, or not be carried at all
- Acts as a long-term memory
  - Able to persist information through as many iterations as needed
  - Important information from the early iterations is not lost for the later iterations



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

Step by step look at the hidden state of LSTM (*forget gate*)

- Forget gate layer "decides" whether the incoming information/state will propagate through this cell
  - "Throw away" or "remember" switch
  - Sigmoid function returns a value [0,1], where 0 means "forget" and 1 means "keep completely"
  - The old information (state from the previous cell) is updated by going through this gate and is either forgotten, kept completely, or kept partially

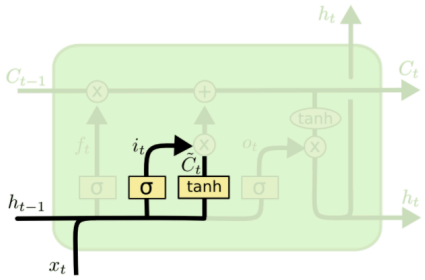


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

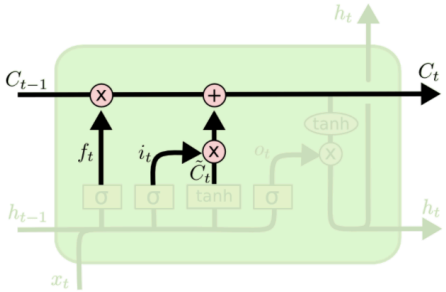
https://colah.github.io/posts/2015-08-Understanding-LSTMs

Step by step look at the hidden state of LSTM (*forget gate*)

- Input gate has two layers
  - Sometimes called “new information” gate
  - Sigmoid function layer “decides” which values are updated
    - Returns a value in [0, 1]
  - Tanh function layer “decides” what new information to add to the hidden state
    - Returns a value in [-1, 1]
  - Then updates the state flowing through the cell:
    - The “forget gate” already updated the previous state input into this cell
    - Additional updates from “new information” gate are added to the retained state after the “forget gate”



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

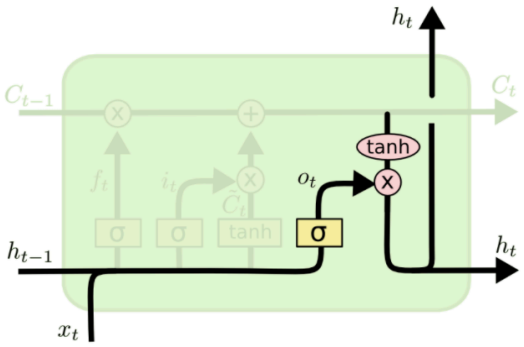


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

Step by step look at the hidden state of LSTM (*output gate*)

- Output gate layer “decides” what the output of this cell is and consists of two layers
  - Sigmoid layer “decides” how much of the cell information goes into the output
    - Returns a value [0, 1]
  - Tanh layer “decides” which parts of the cell information go into the output
    - Returns a value [-1, 1]

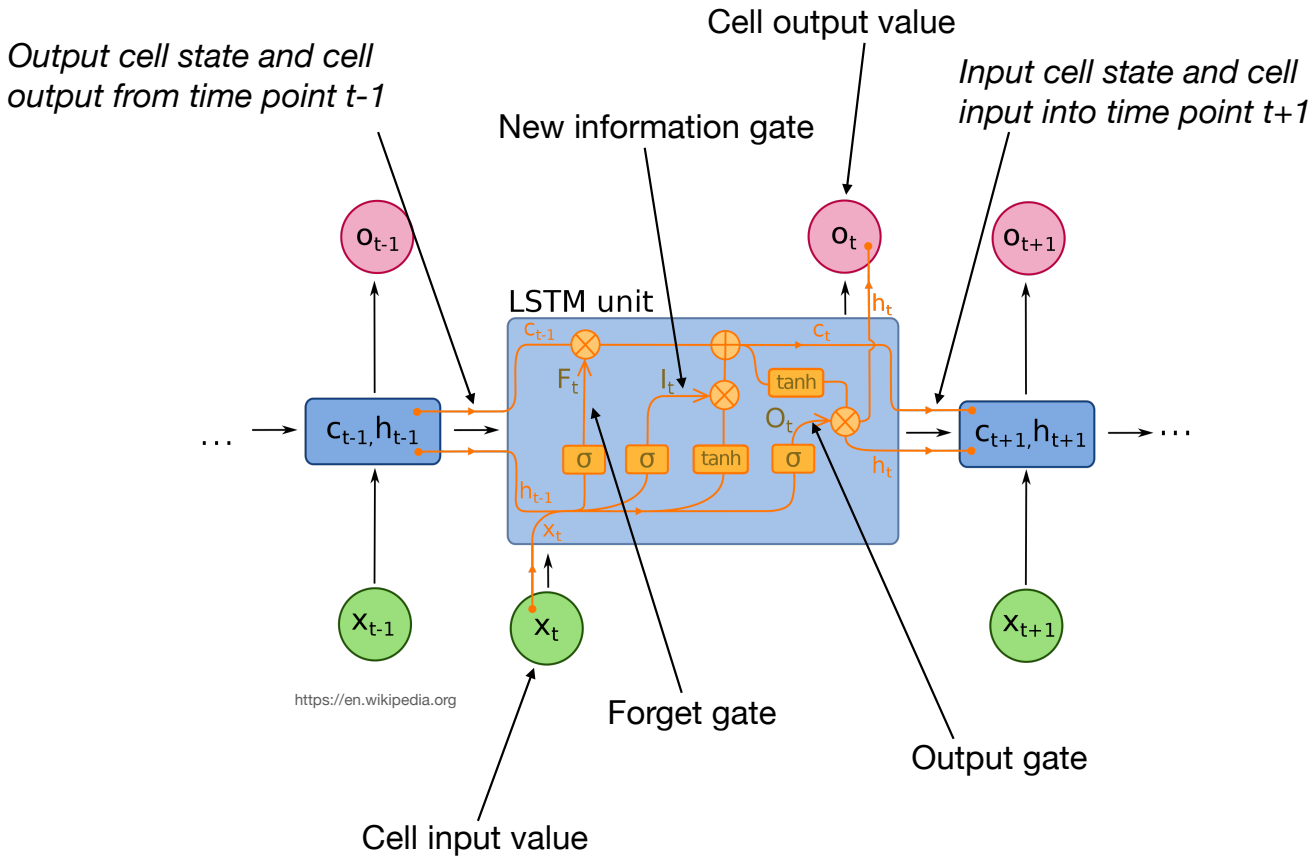


https://colah.github.io/posts/2015-08-Understanding-LSTMs

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Bringing it to the bird's eye view again



Interpretations of the gates inside an LSTM cell

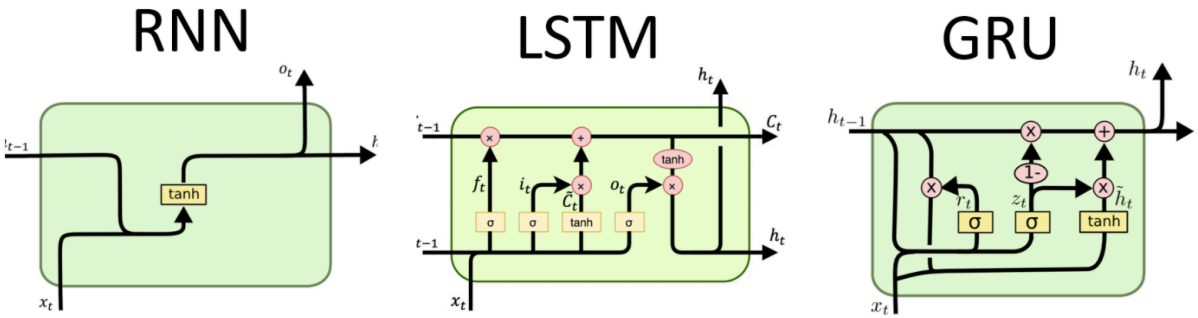
- *Forget gate* controls if the memory of the cell is reset (how much of the previous cell's state does it retain?)
- *Input gate* controls whether and with how much new information the cell state is updated
- Output gate controls how much of the information of the current cell is made visible and output from the cell

Output of each LSTM cell is based on

- *Cell state* - long term memory
- *Working memory* - short term memory from this iteration
- *Output value* - used for this iteration prediction as well as the input into the next cell

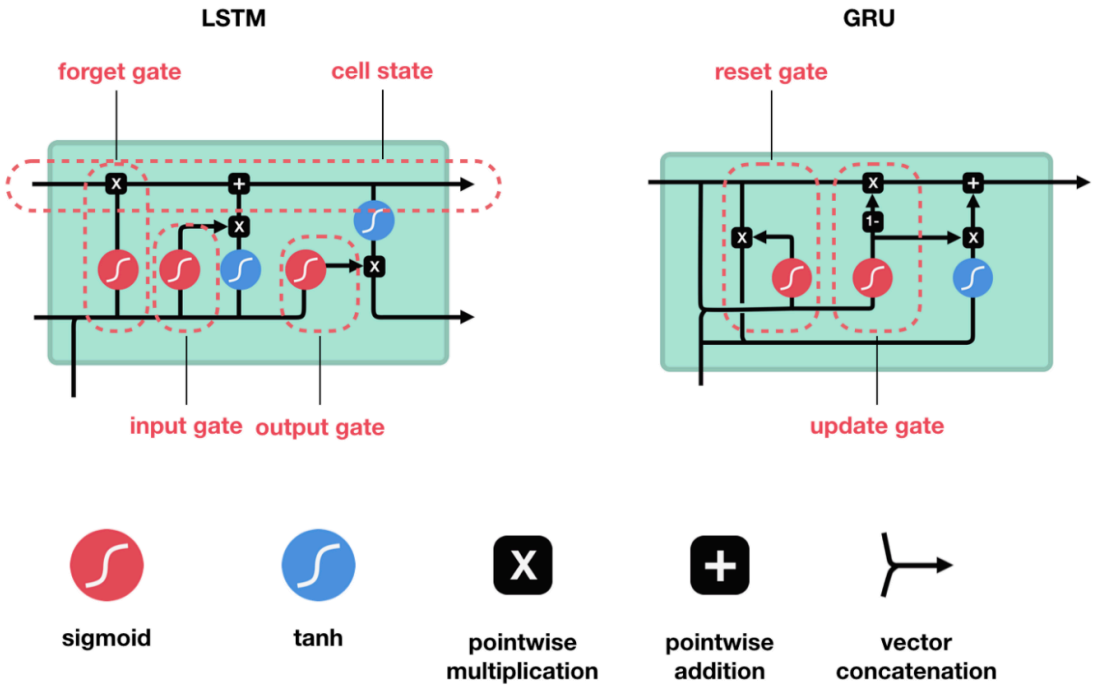
What are GRU networks?

- Gated recurrent unit (GRU) are another type of RNNs
- Similarly to LSTMs, GRUs have gates memory cells
  - Different gate configuration than LSTMs



http://dprogrammer.org/rnn-lstm-gru

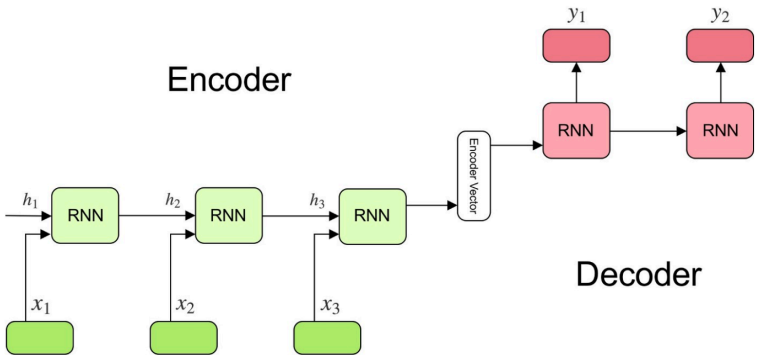
Detailed look at the GRU memory cell



towardsdatascience.com

Encoder-decoder RNNs

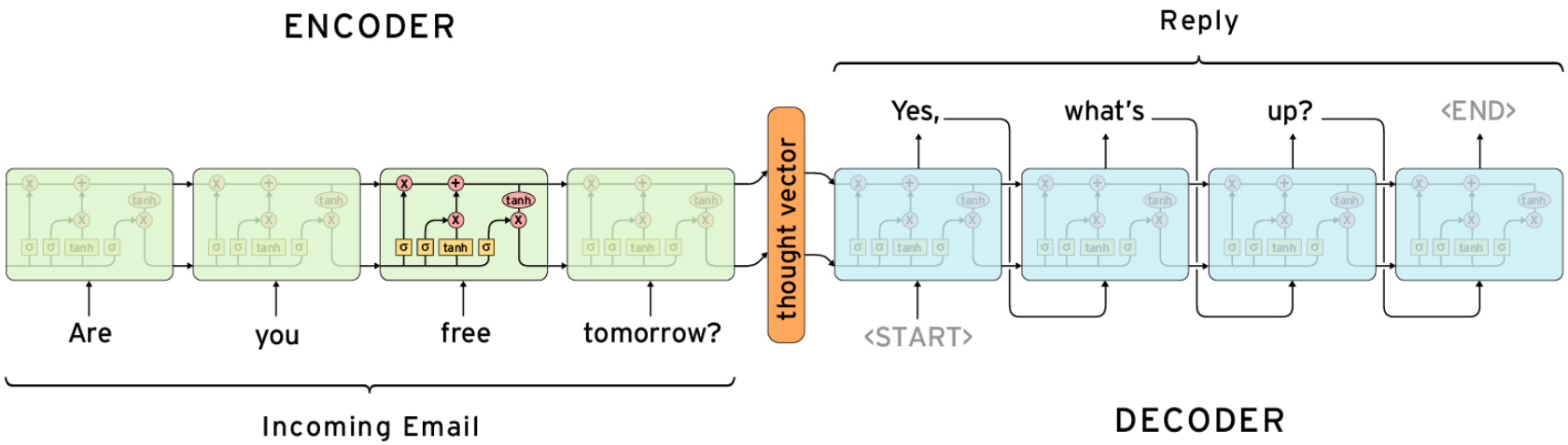
- Encoder-decoder RNNs are a common approach to solving machine translation and other seq2seq prediction problems
- E.g. translating from English to German
  - Input and output could be different lengths
  - We need to capture the meaning of the English sentence and translate that meaning into a German sentence, not translate word for word



Encoder-decoder sequence to sequence model

https://towardsdatascience.com

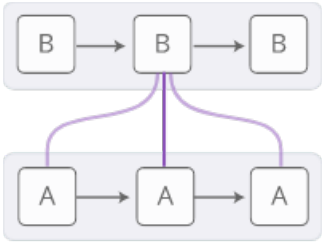
Example: response prediction model



www.gabormelli.com

Neural attention

- Attention techniques allow RNNs to focus on subsets of the information given to the neural network
- Attention interfaces/vectors are very useful in machine translation and seq2seq problems
  - Encoder-decoder RNNs
- Different attention is given to different subsets of inputs
- The attention can be made context-specific
- Attention can be used to interface with a network with repeating patterns in its output
- Attention vectors define connections between relevant pieces of content



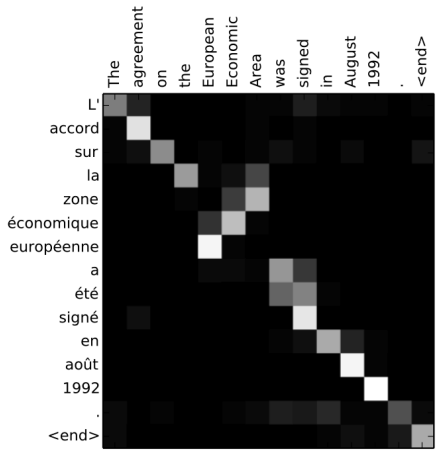
https://distill.pub/2016/augmented-rnns

Motivation for attention techniques

- When we are in the process of translating a sentence from one language to another, we are mostly paying attention to the current word we are on, not the entire sentence

Example: attention vector for two sentences

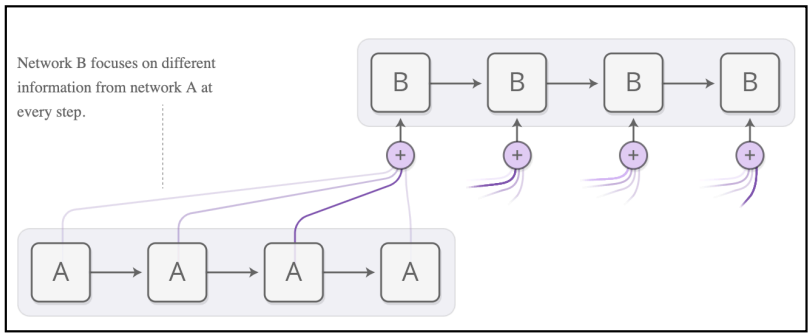
- “The agreement on the European economic areas was signed in August 1992”
- The matching matrix identifies relevant context
  - How meanings in one languages correlate to meanings in another languages

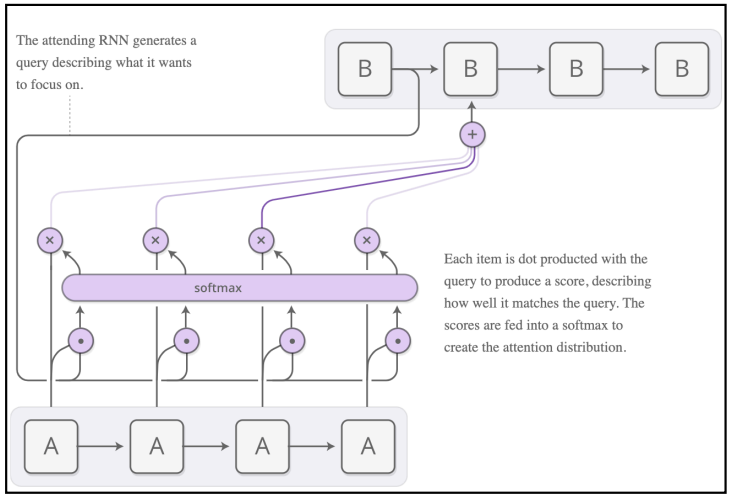


“NEURAL MACHINE TRANSLATION” Bahdanau et al. 2016

Attention vectors can be thought of as masks

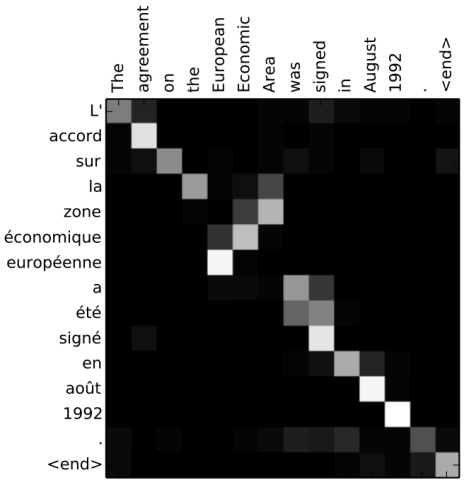
- Each attention vector defines which input features are relevant to the current output
- It is a filter or a mask on the input values to filter out irrelevant information



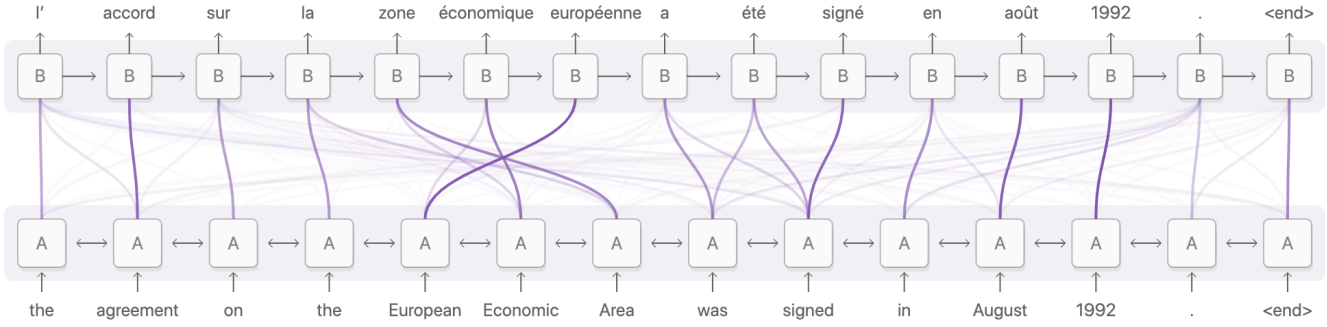


https://distill.pub/2016/augmented-rnns

Back to our example



https://distill.pub/2016/augmented-rnns



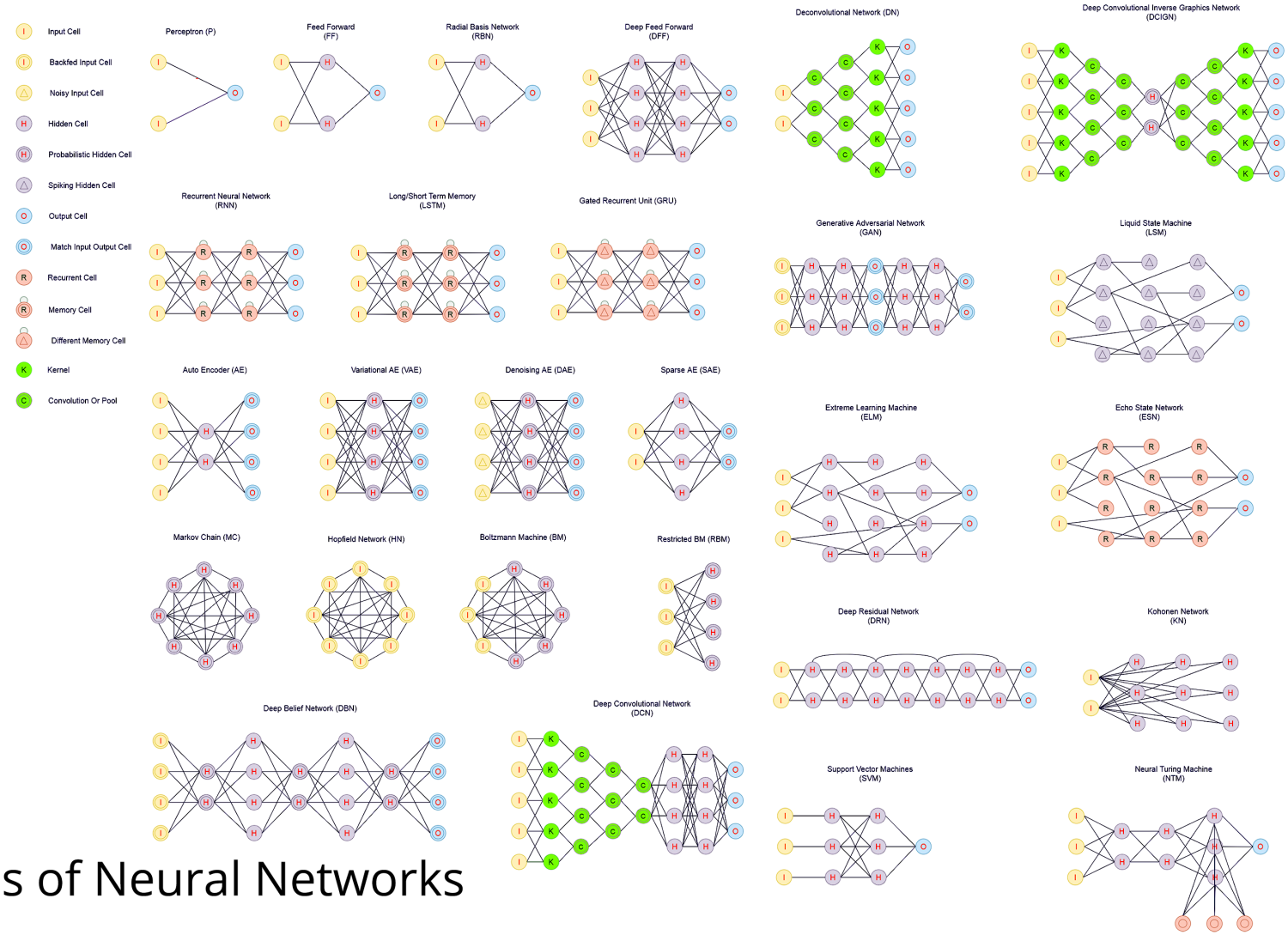
"NEURAL MACHINE TRANSLATION" Bahdanau et al. 2016

Useful resources about RNNs

- https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- https://www.youtube.com/watch?v=WCUNPb-5EYI
- https://www.youtube.com/watch?v=QciIcRxJvsM
- https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
- https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

Many many other variations of ANNs

- New literature is published all the time
- The only way to stay current is to keep an eye on the publications related to your problem of interest



medium.com

Main Types of Neural Networks

Let’s look at some code examples

- Classification of handwritten digits with RNNs
  - *RNN.MNIST.ipynb*

- Now let’s look at some fundamentals of Natural Language processing (NLP)