Neural Nets for NLP

Vsevolod Dyomkin

prj-nlp-2020

Limitations of “Classic” NLP

* categorical (1-hot) features
* extra large feature spaces
* UNK problems
* complicated feature engineering
* difficult domain adaptation
* need for markovization in sequence models
* what else?

Neural Nets to the Rescue

RBM CNN

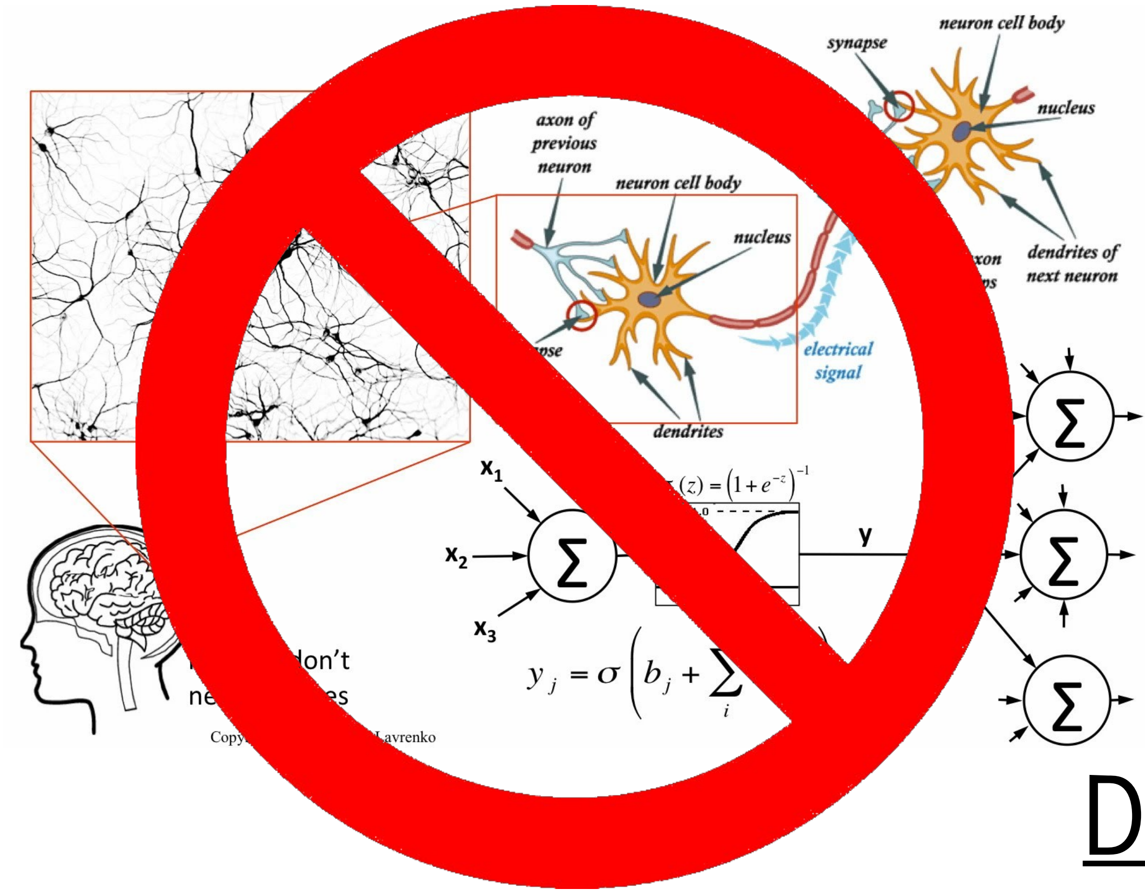
MLP DBN SOM

SNN RNN

Hopfield GAN

VAE Capsule

Terminology



Deep Learning?

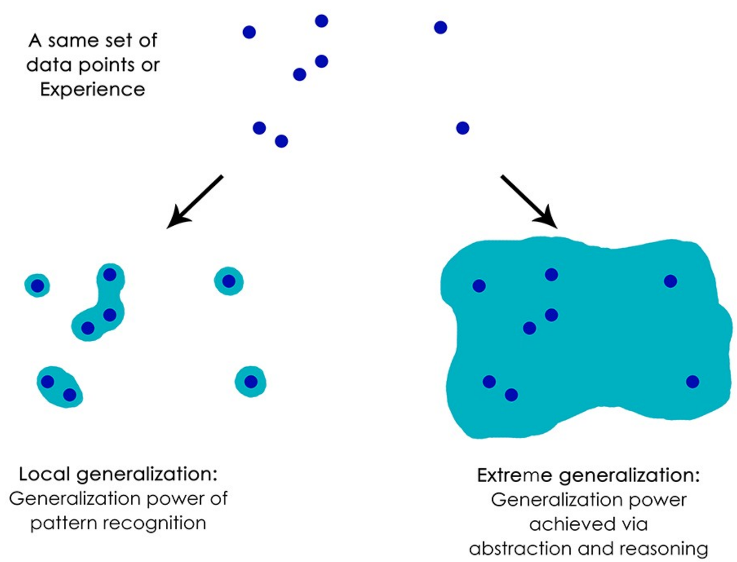
https://blog.keras.io/the-limitations-of-deep-learning.html

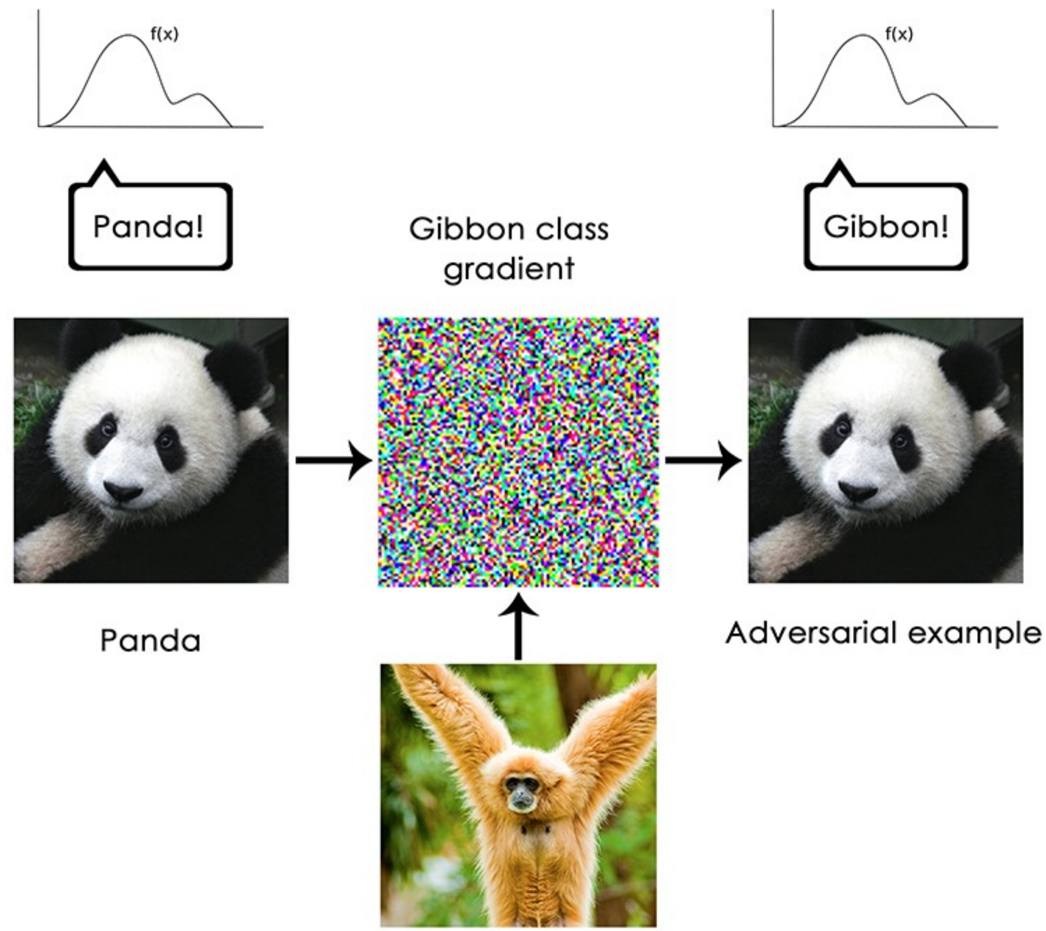
A Geometric View of DL

In deep learning, everything is a vector, i.e. everything is a point in a geometric space. Model inputs (it could be text, images, etc) and targets are first ”vectorized”, i.e. turned into some initial input vector space and target vector space. Each layer in a deep learning model operates one simple geometric transformation on the data that goes through it. Together, the chain of layers of the model forms one very complex geometric transformation, broken down into a series of simple ones. This complex transformation attempts to map the input space into the target space, one point at a time. This transformation is parameterized by the weights of the layers, which are iteratively updated based on how well the model is currently performing. A key characteristic of this geometric transformation is that it must be differentiable, which is required in order for us to be able to learn its parameters via gradient descent. Intuitively, this means that the geometric morphing from inputs to outputs must be smooth and continuous—a significant constraint.

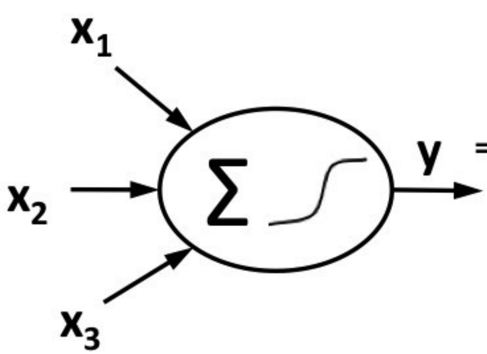
The Limitations of DL

The boy is holding a baseball bat.





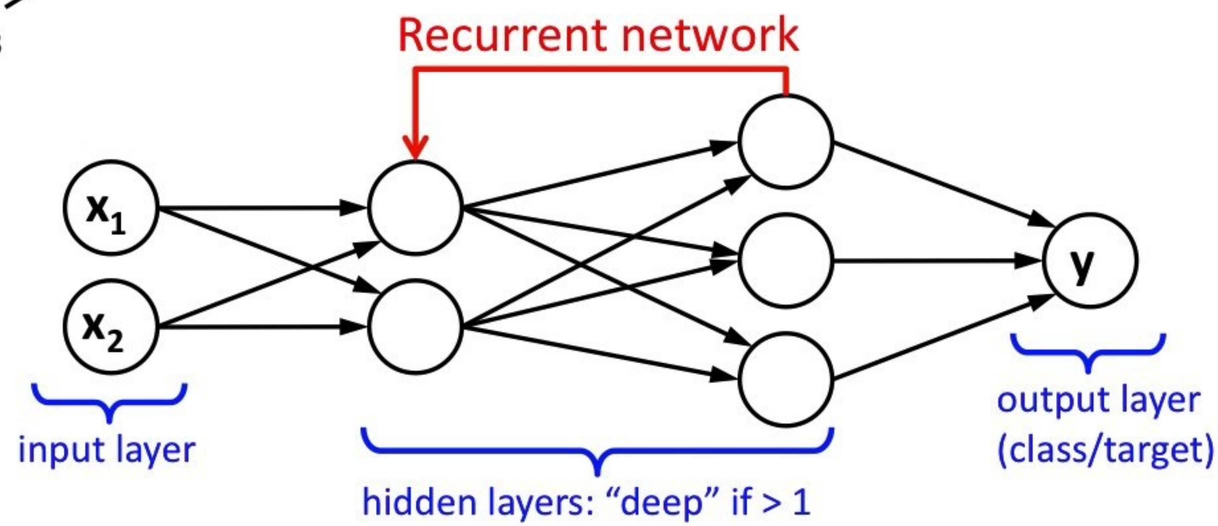
Types of Neural Networks



$$y = \sigma\left(b + \sum_i w_{ij} x_i\right)$$

Single neuron: perceptron,

linear / logistic regression

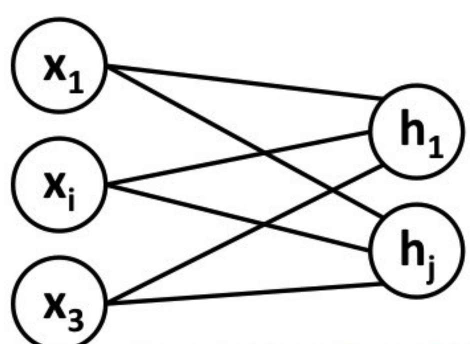


Feed-forward network

(no cycles) -- non-linear classification & regression

P (input | hidden)

$$\sigma\left(\beta_i + \sum_j \boxed{w_{ij}} h_j\right) =$$



P (hidden | input)

$$= \sigma\left(b_j + \sum_i \boxed{w_{ij}} x_i\right)$$

same set of weights

Symmetric (RBM)

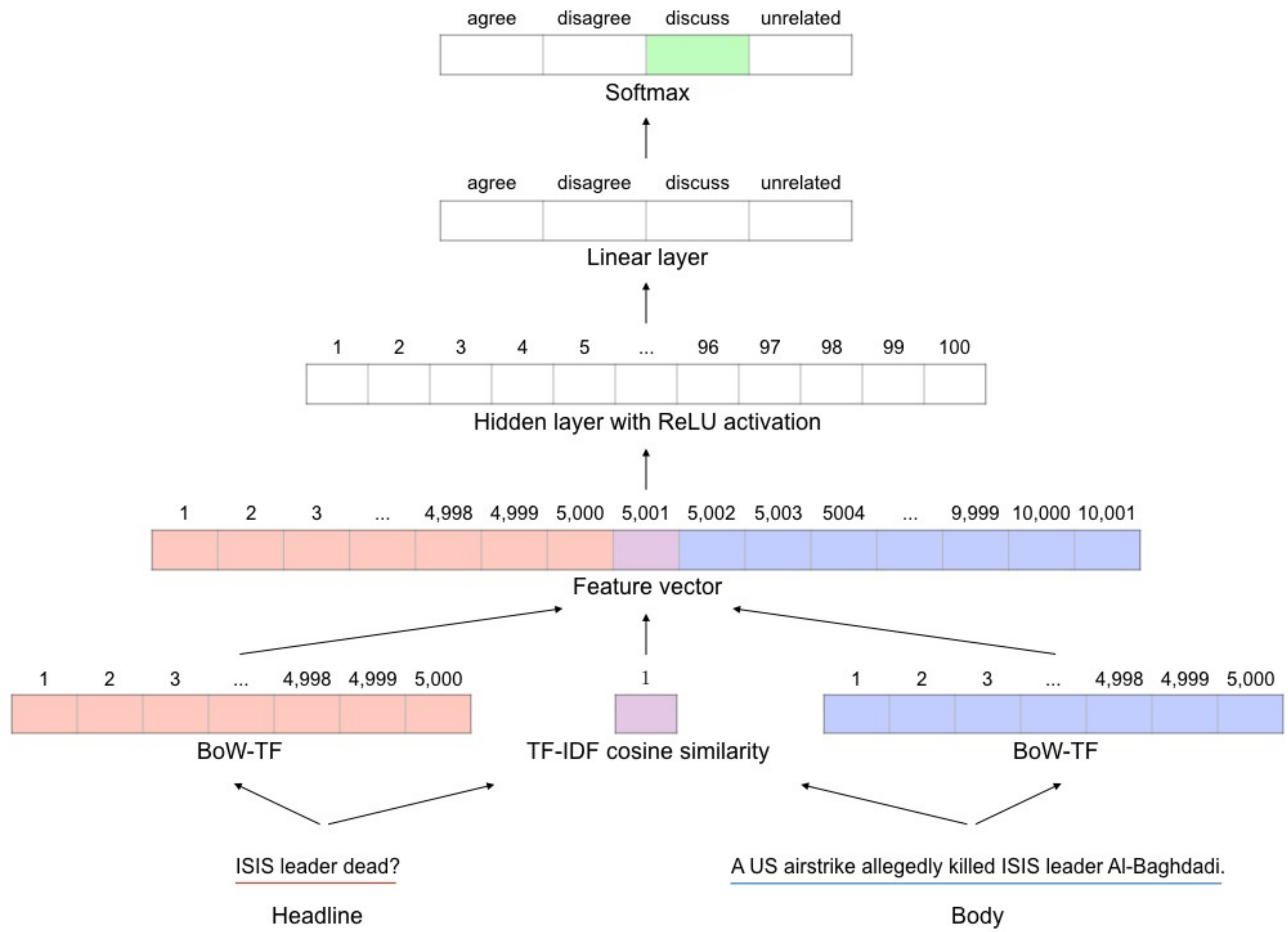
unsupervised, trained to maximize likelihood of input data

a mixture model

~~MLPs~~ FNNs

* computational graph
* composed of various layers
* backprop for learning
* GD for optimization

Example: FNC-1



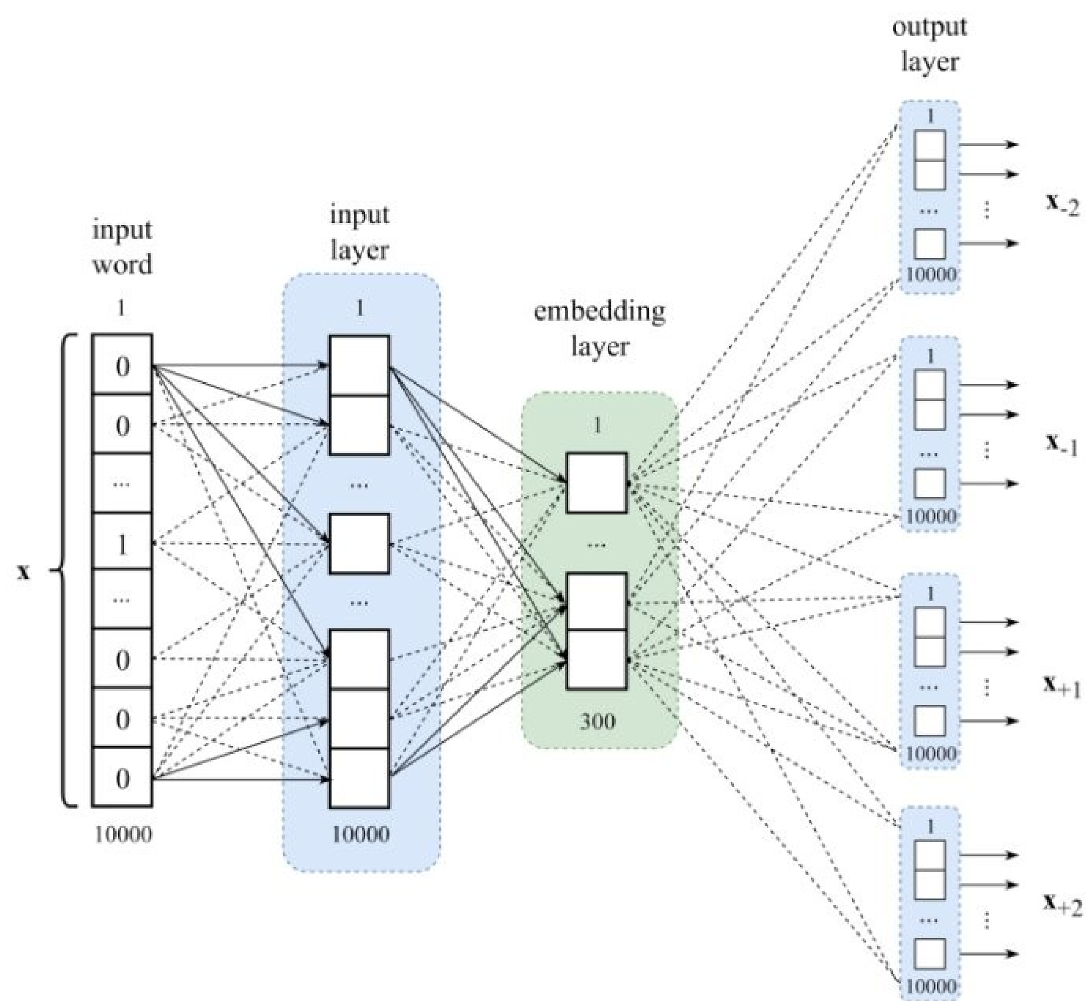
https://github.com/uclmr/fakenewschallenge

Layers

* input
* fully-connected
* convolutional
* non-linearities
* regularization
* output

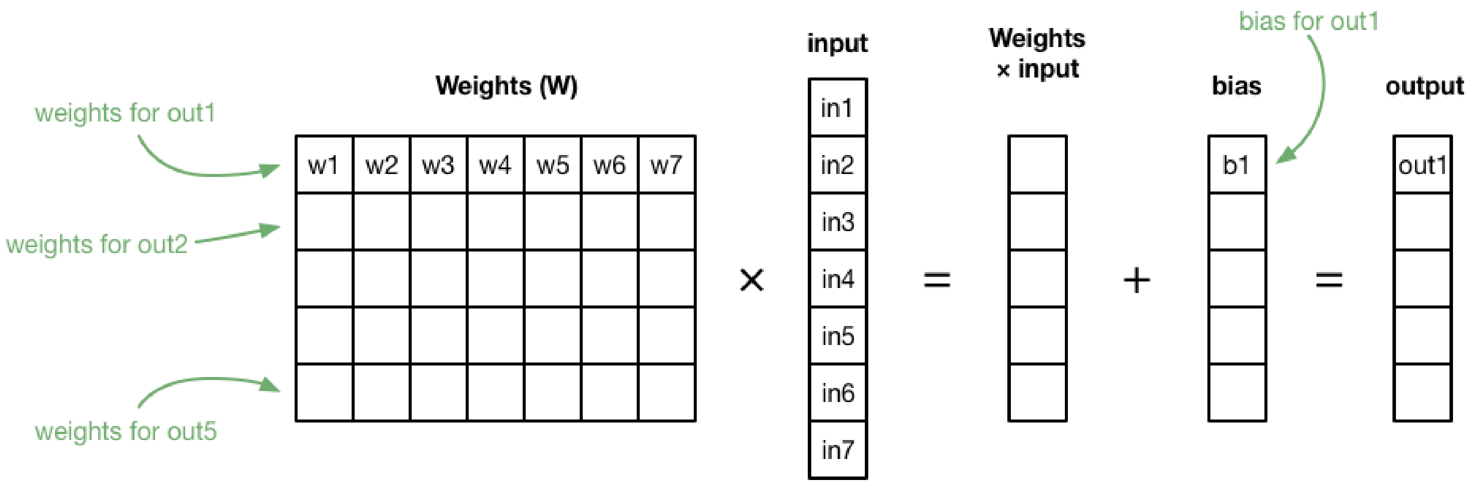
Input Layers

* 1-hot
* embedding
* mixed

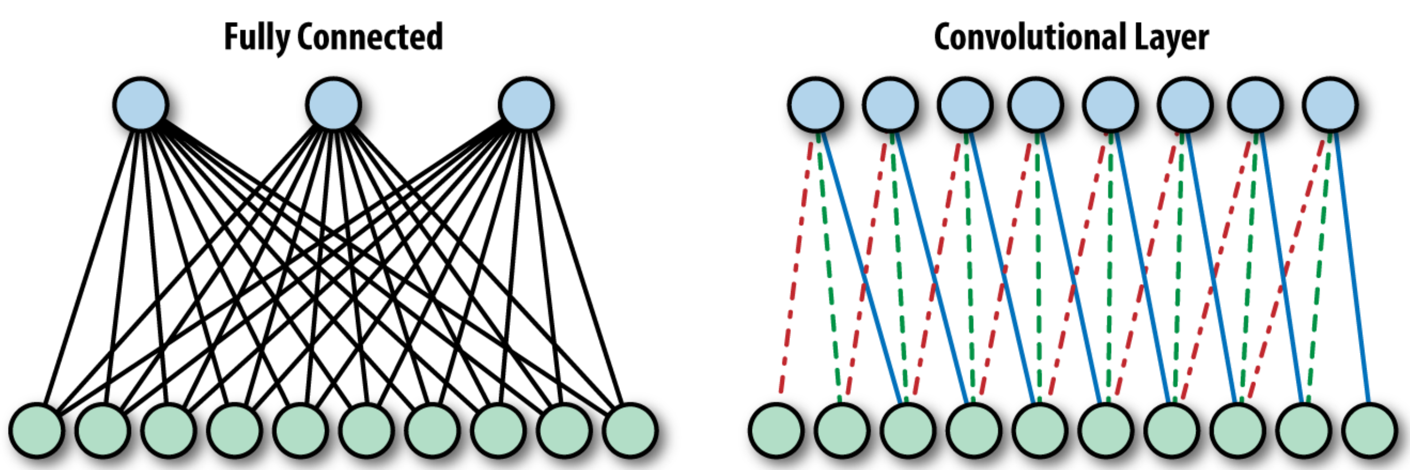


Fully-Connected Layers

* linear transformation

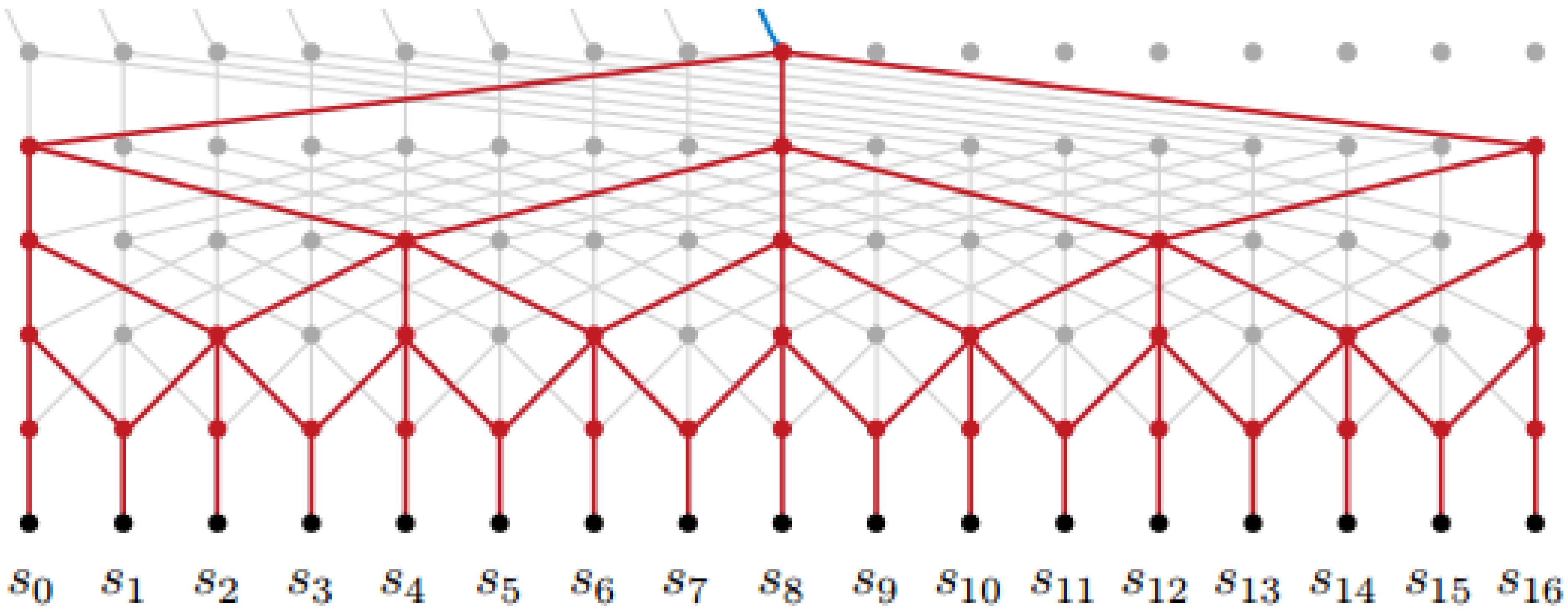


output = f(Weights × input + bias)



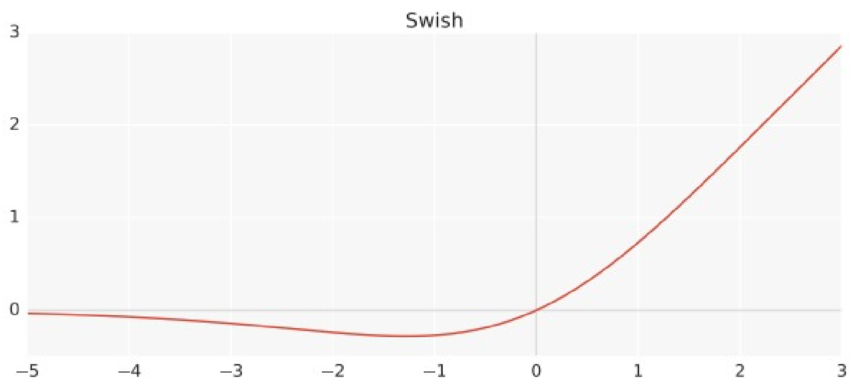
Convolutional Layers

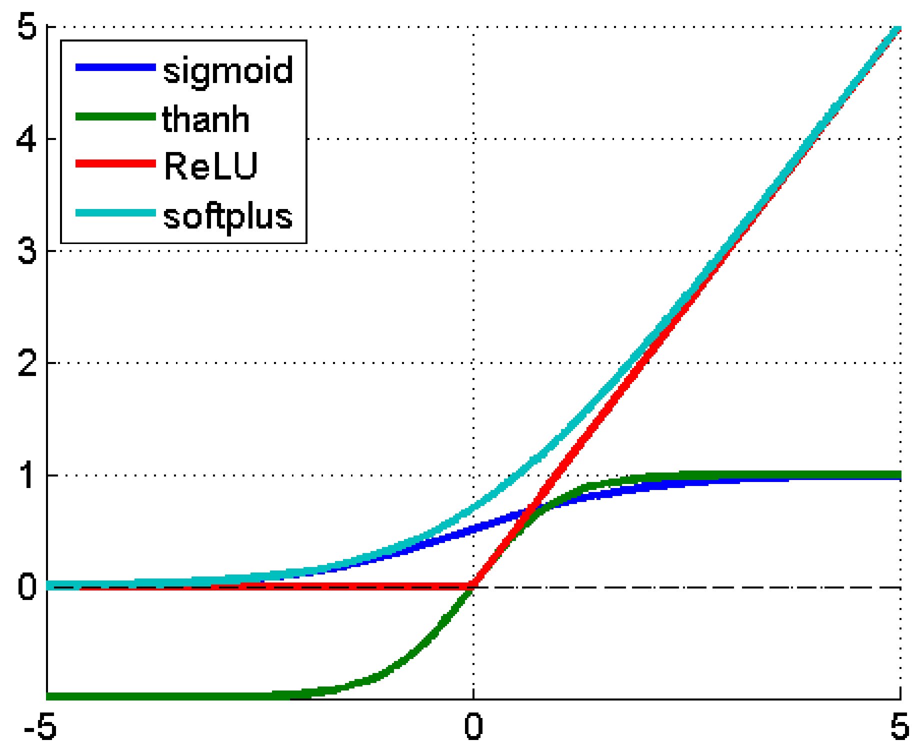
* apply mask
+ pooling (max, mean,…)



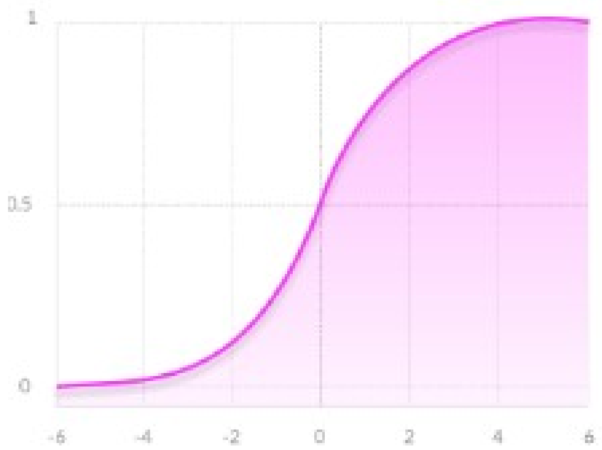
Nonlinearities

* sigmoid/logistic
* tanh
* ReLU/SeLU/ELU/leakyReLU/…
* softplus
* swish





* maxout



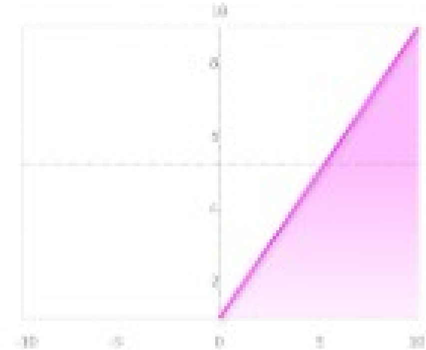
SIGMOID / LOGISTIC

ADVANTAGES

- **Smooth gradient**, preventing "jumps" in output values.
- **Output values bound** between 0 and 1, normalizing the output of each neuron.
- **Clear predictions**—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.

DISADVANTAGES

- **Vanishing gradient**—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- **Outputs not zero centered**.
- **Computationally expensive**



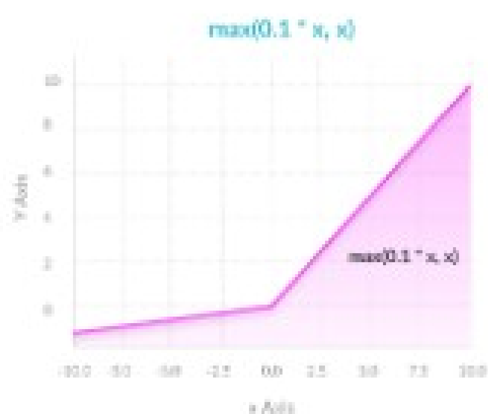
RELU (RECTIFIED LINEAR UNIT)

ADVANTAGES

- **Computationally efficient**—allows the network to converge very quickly
- **Non-linear**—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation

DISADVANTAGES

- **The Dying ReLU problem**—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.



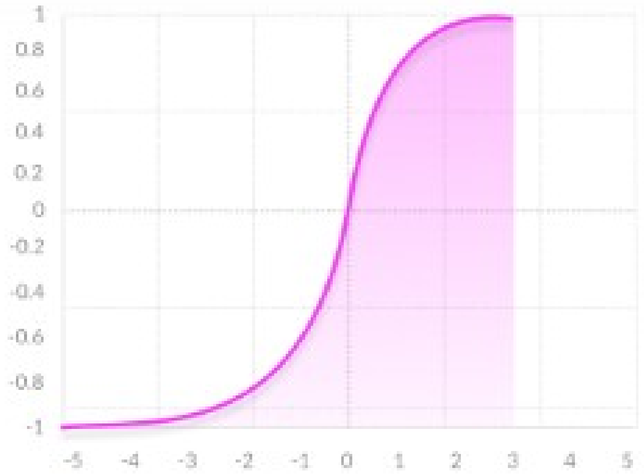
LEAKY RELU

ADVANTAGES

- **Prevents dying ReLU problem**—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values
- Otherwise like ReLU

DISADVANTAGES

- **Results not consistent**—leaky ReLU does not provide consistent predictions for negative input values.



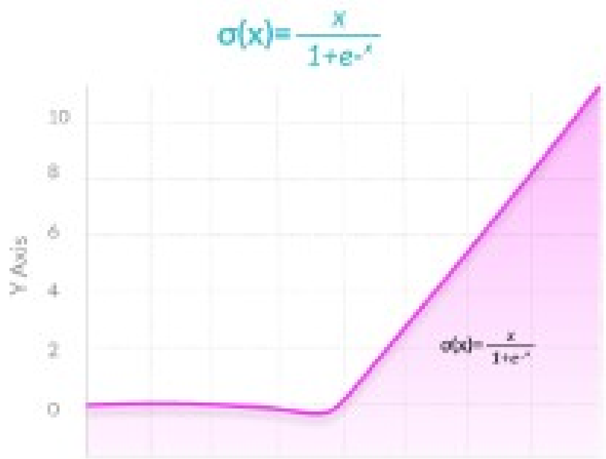
TANH / HYPERBOLIC TANGENT

ADVANTAGES

- **Zero centered**—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.
- Otherwise like the Sigmoid function.

DISADVANTAGES

- Like the Sigmoid function

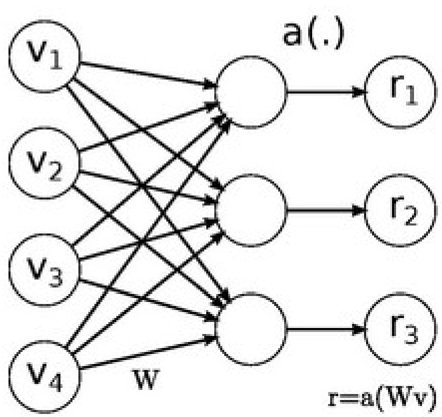


SWISH

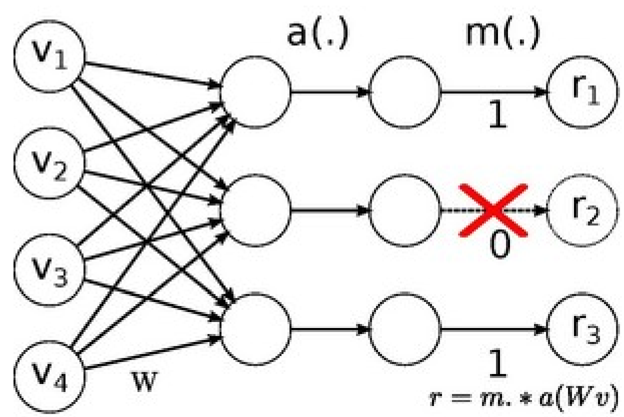
Swish is a new, self-gated activation function discovered by researchers at Google. According to their paper, it performs better than ReLU with a similar level of computational efficiency. In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher.

Regularization Layers

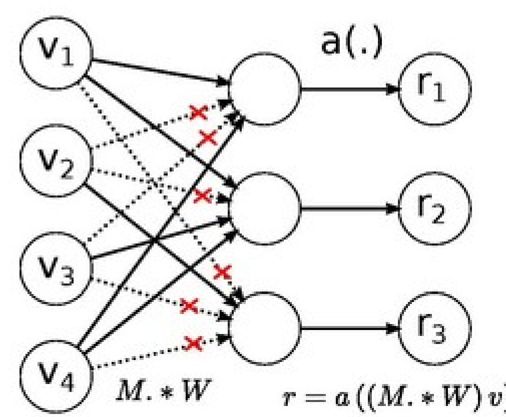
* nonlinearity regularization
* dropout
* dropconnect



No-Drop Network



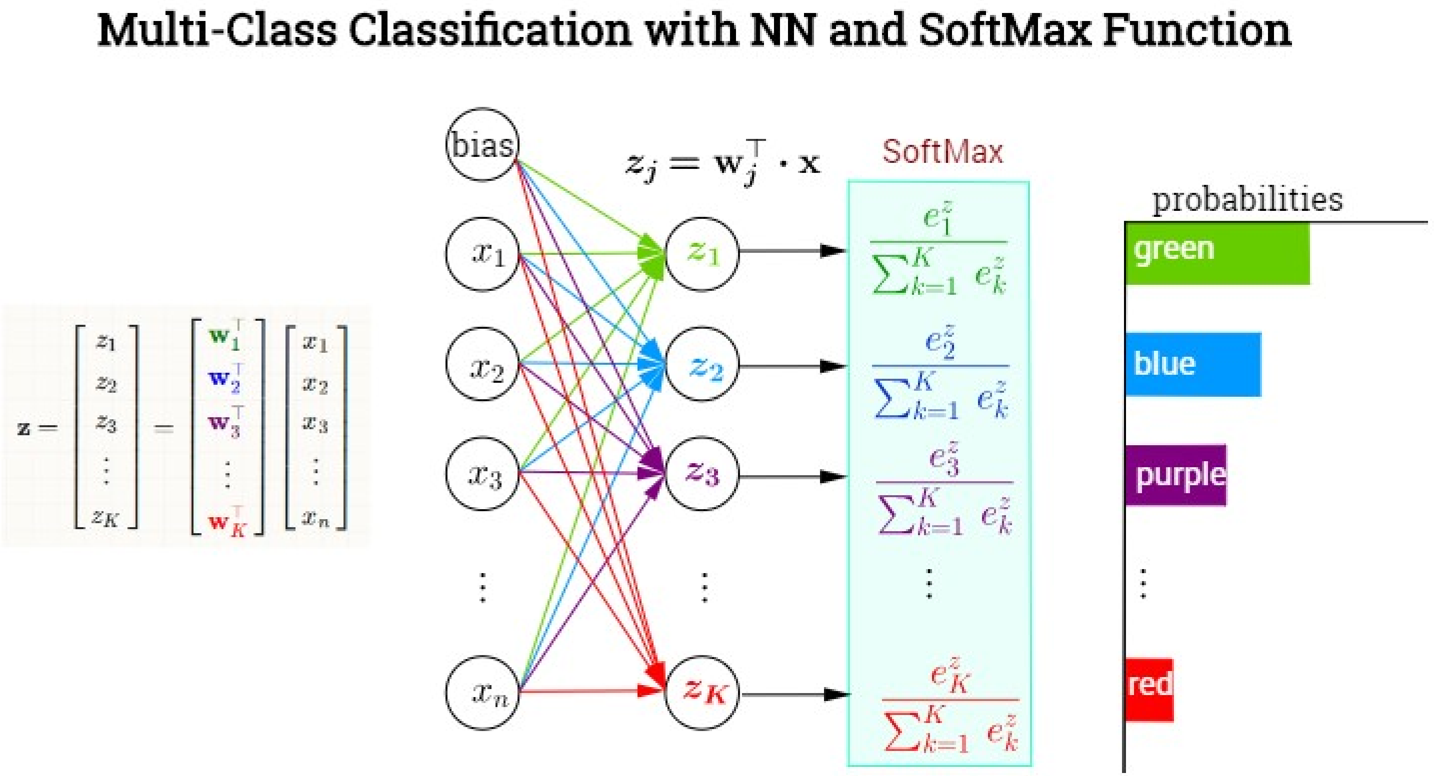
DropOut Network



DropConnect Network

Output Layers

* softmax/hierarchical softmax



Loss Functions

* maximum likelihood estimation (MLE) — Cross Entropy (Log loss)

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

* max margin objective - Hinge loss

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \max(0, m - y^{(i)} \cdot \hat{y}^{(i)})$$

* KL-divergence loss

$$\begin{aligned}\mathcal{L} &= \frac{1}{n}\sum_{i=1}^{n} \mathcal{D}_{KL}(y^{(i)}||\hat{y}^{(i)}) \\ &= \frac{1}{n}\sum_{i=1}^{n} \left[y^{(i)} \cdot \log\left(\frac{y^{(i)}}{\hat{y}^{(i)}}\right)\right] \\ &= \underbrace{\frac{1}{n}\sum_{i=1}^{n} \left(y^{(i)} \cdot \log(y^{(i)})\right)}_{entropy} \underbrace{- \frac{1}{n}\sum_{i=1}^{n} \left(y^{(i)} \cdot \log(\hat{y}^{(i)})\right)}_{cross-entropy}\end{aligned}$$

https://isaacchanghau.github.io/post/loss_functions/

Backprop

* efficient way to compute derivatives (using DP)
* automatic & symbolic differentiation

https://colah.github.io/posts/2015-08-Backprop/

Optimization Algorithm

* gradient descent
* SGD (+minibatch)
* Momentum
* Adagrad/Adadelta/...
* Adam

https://ruder.io/optimizing-gradient-descent/

Example: FNN for Adjective Ordering

Input: noun & 2 adjectives
       (embeddings)
Hidden: 4 ReLU FC-layes
Output: Sigmoid (Softmax)

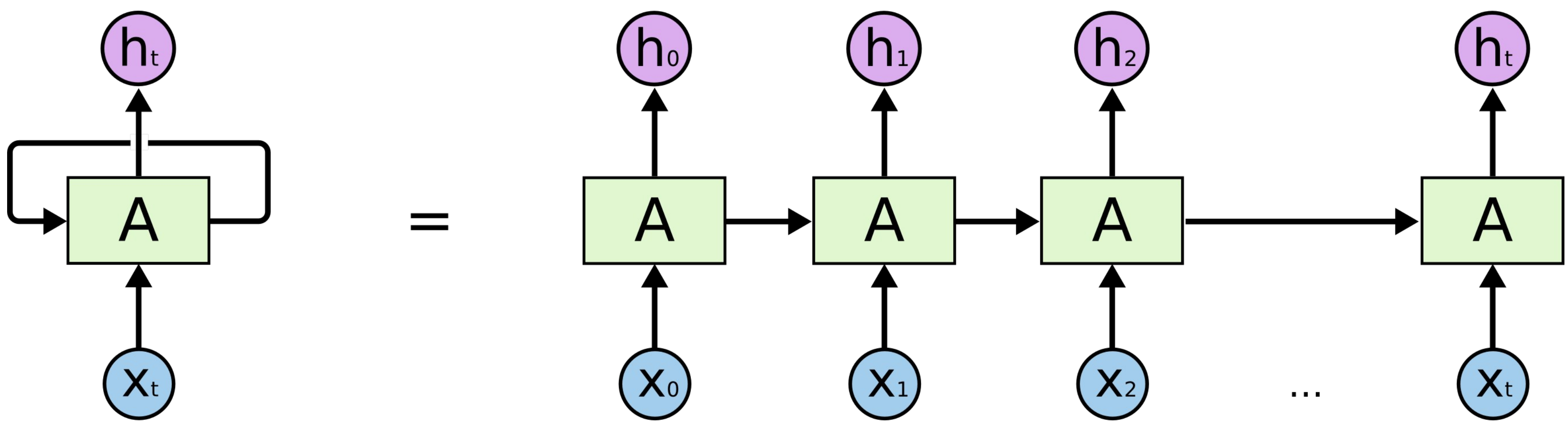
Run with both variants of order and select the better score

FNNs Recap

+ nonlinear, flexible
+ efficient training
+ allows to use embeddings
- fixed input
- limited context

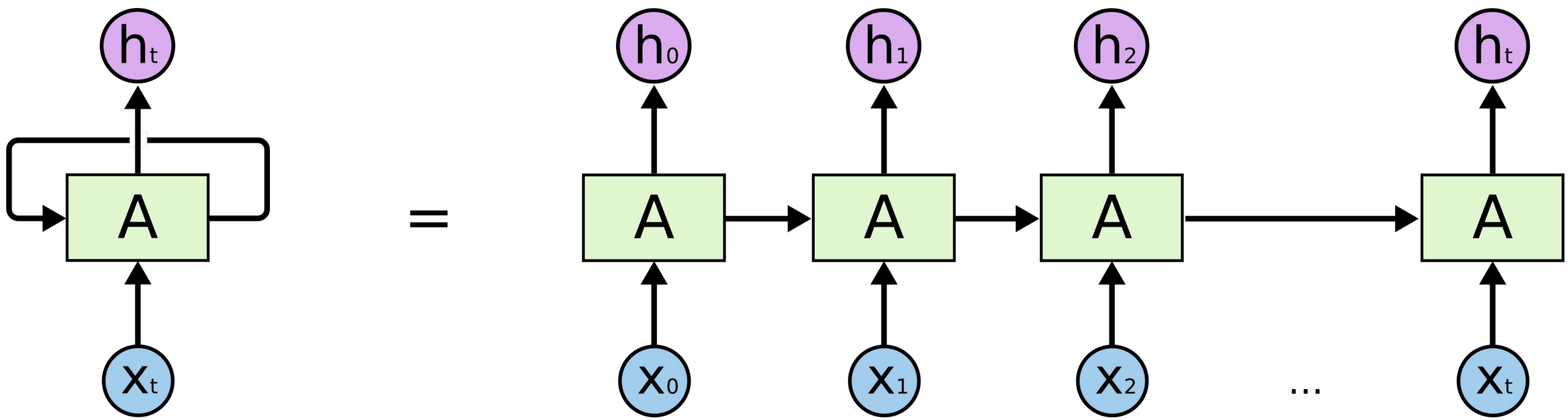
RNNs to the Rescue

* add previous state to input
* backpropagate through time (truncated - BPTT)



RNNs to the Rescue

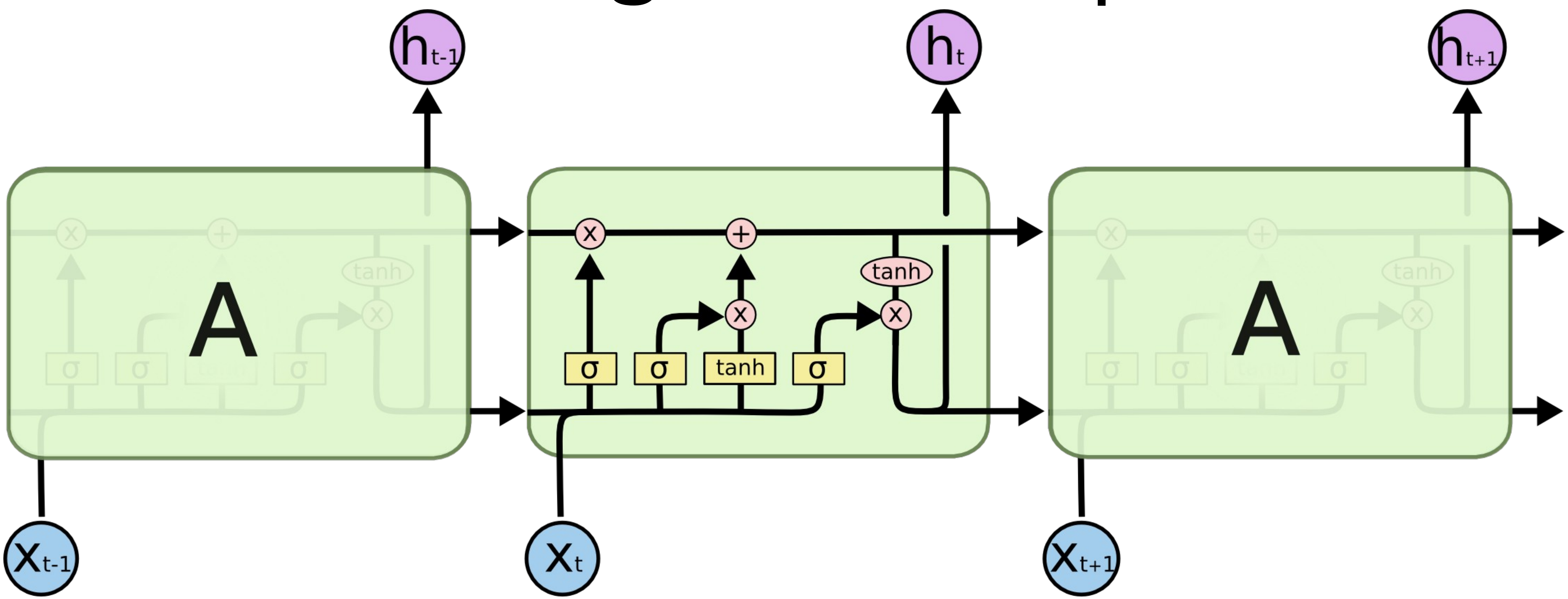
* add previous state to input
* backpropagate through time



* not so easy:
  - vanishing gradients
  - exploding gradients

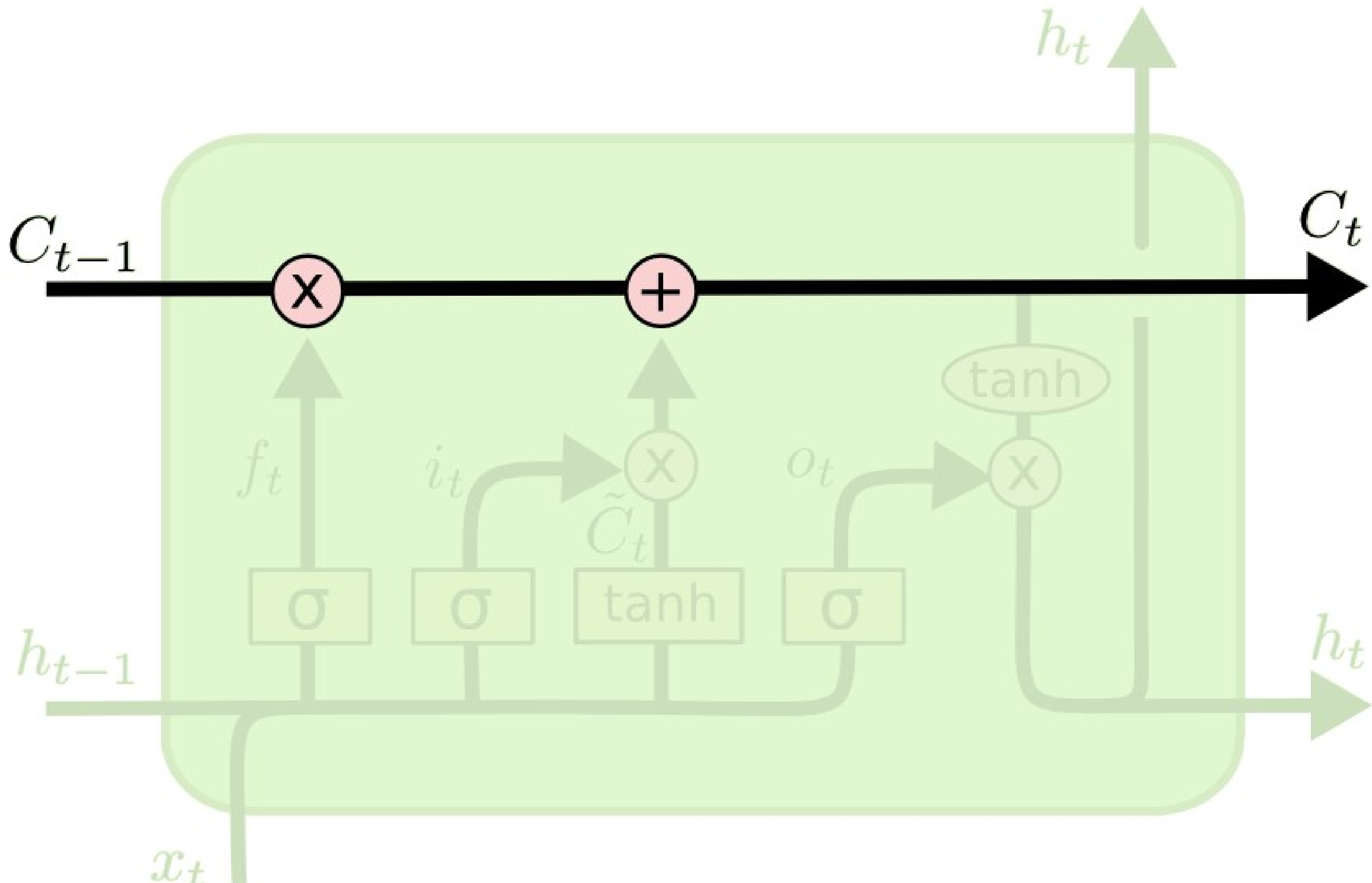
LSTM

specifically designed to remember long-term dependencies

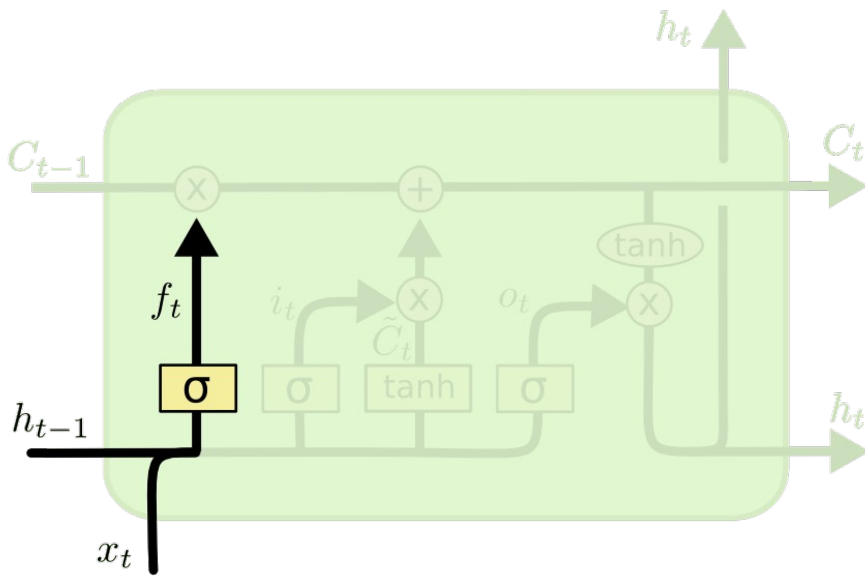


https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Cell State

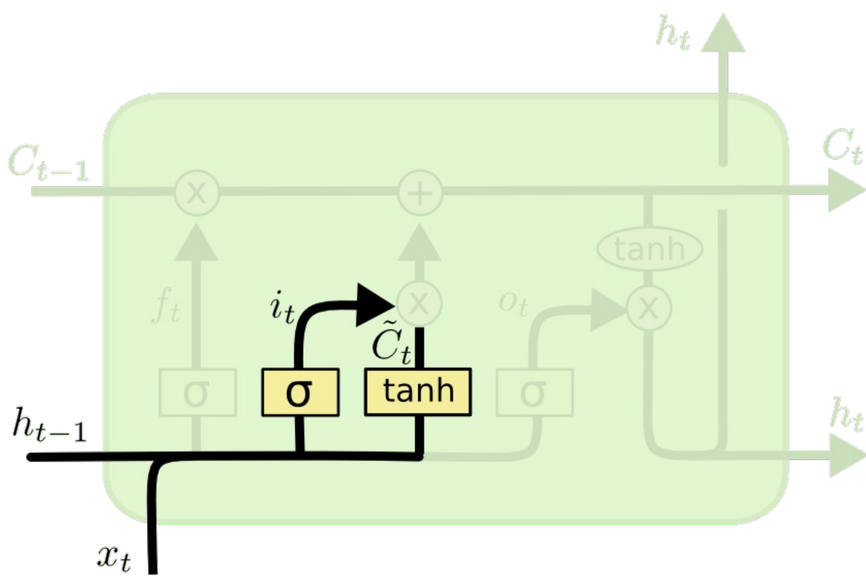


LSTM Forget Gate



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

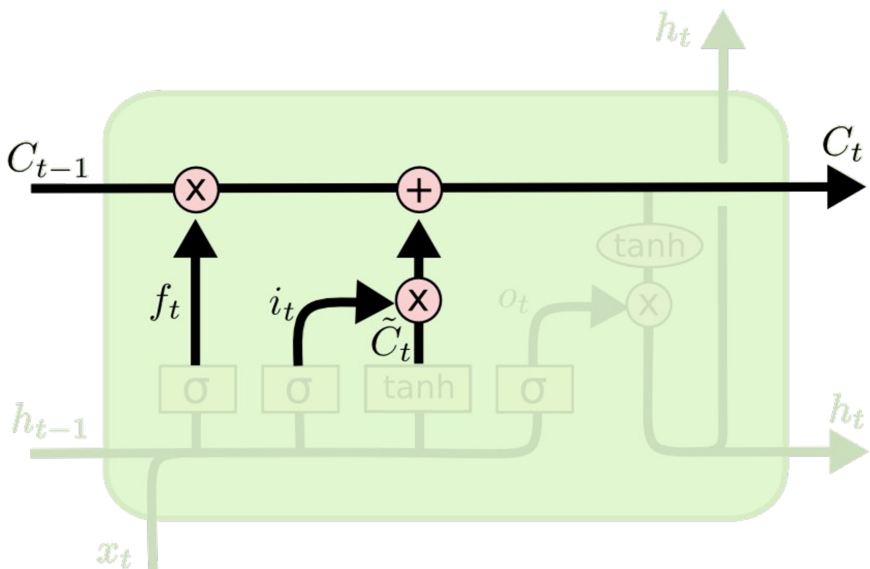
LSTM Remember Gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

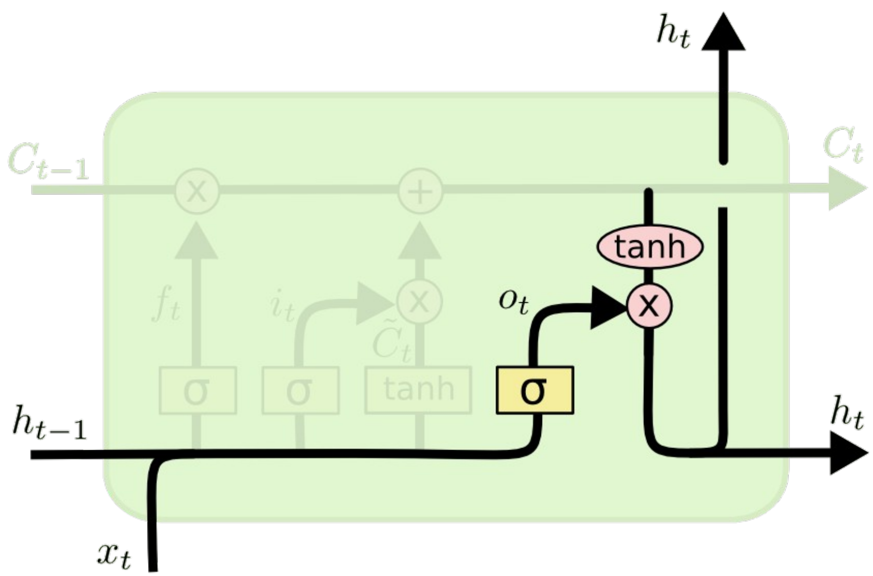
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM State Update



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

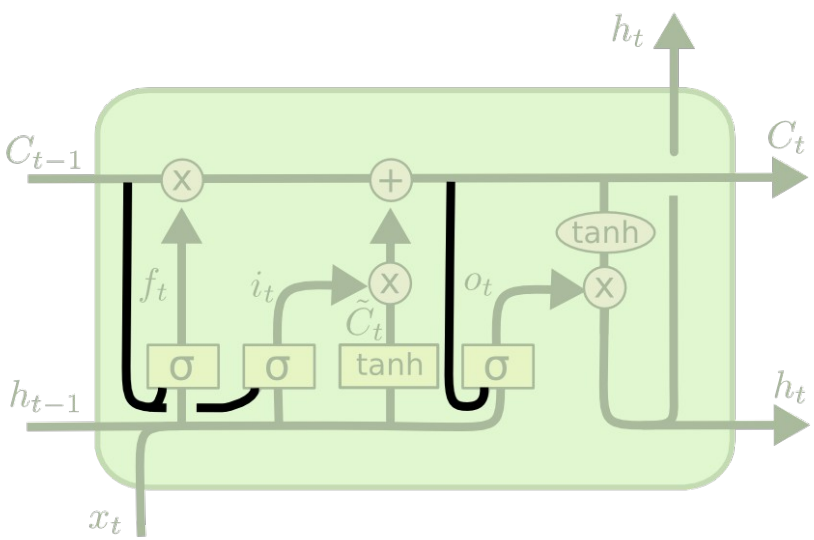
LSTM Output



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

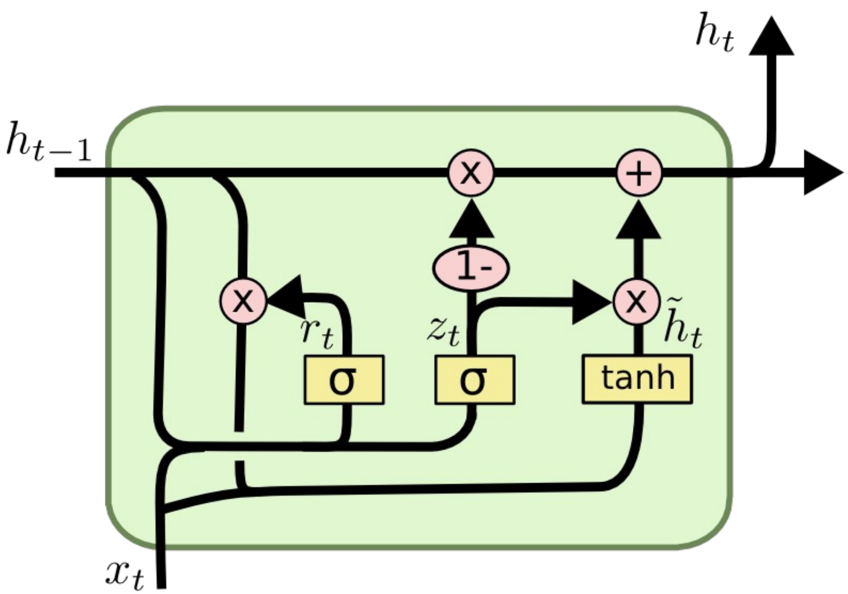
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM Peephole Connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \;+\; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \;+\; b_o\right)$$

Gated Recurrent Unit (GRU)



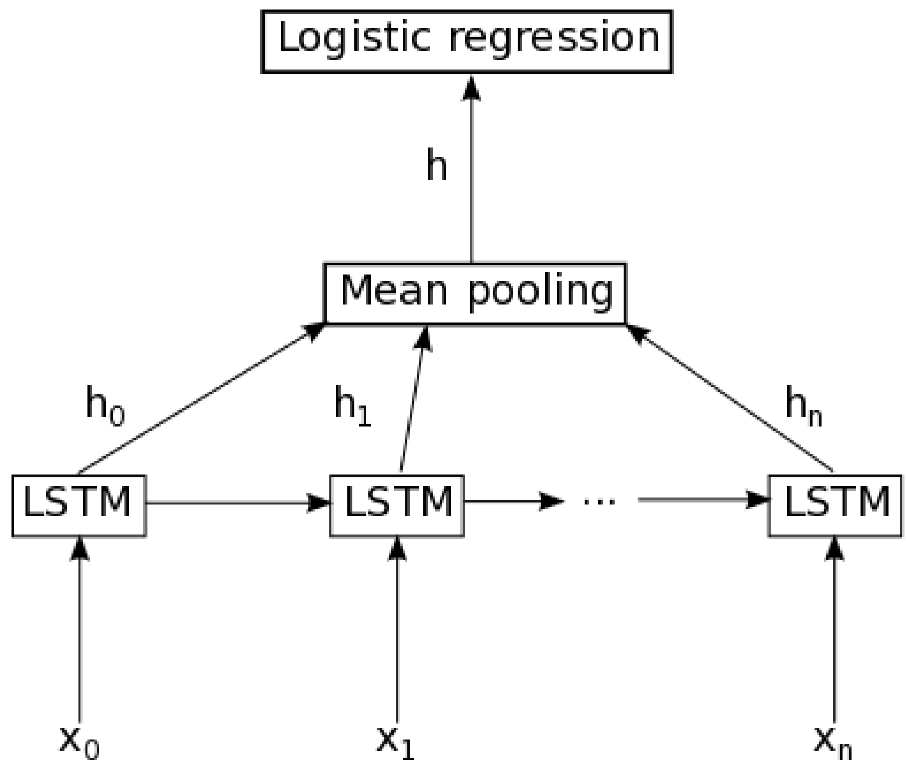
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

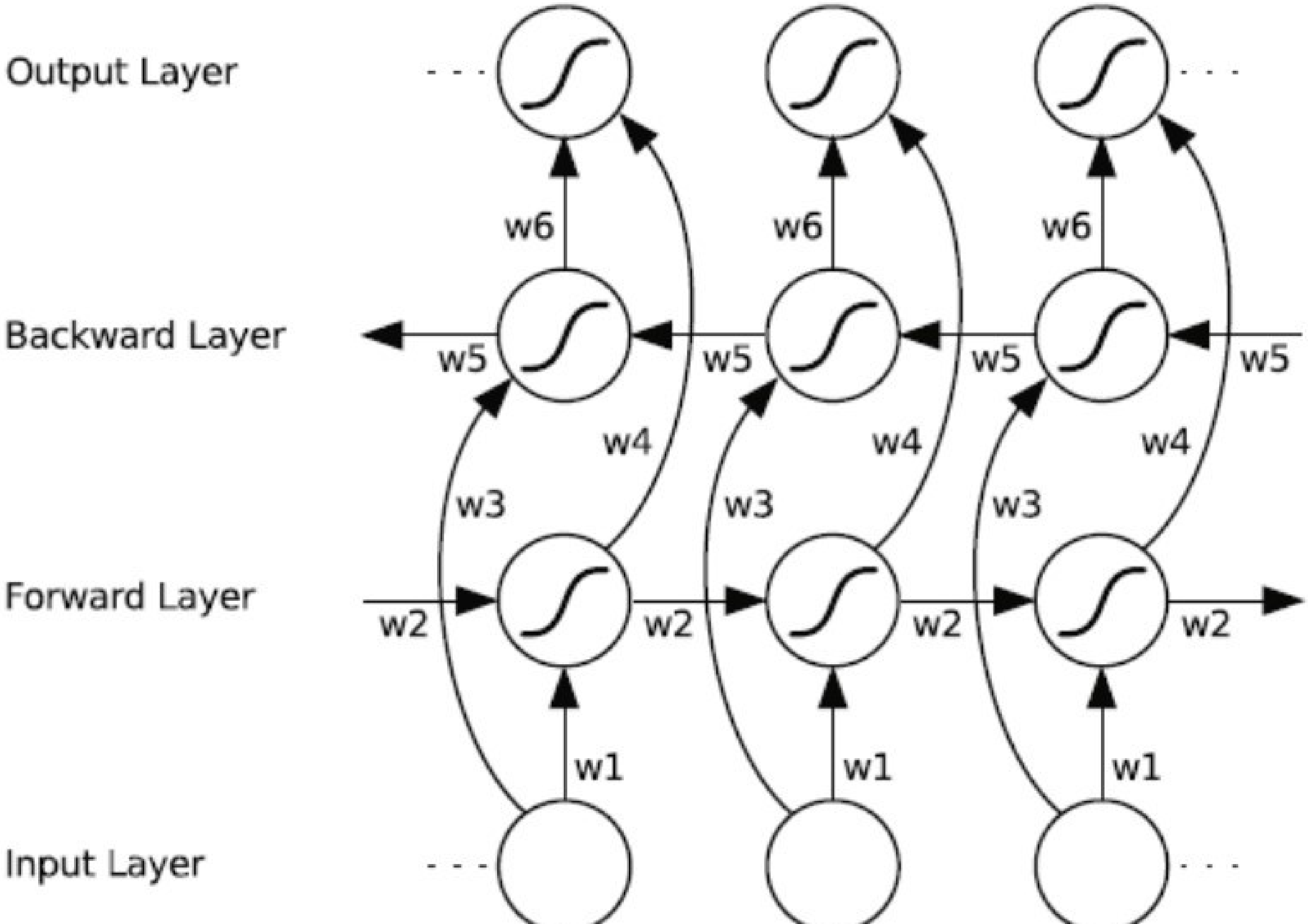
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM Example: Sentiment Analysis



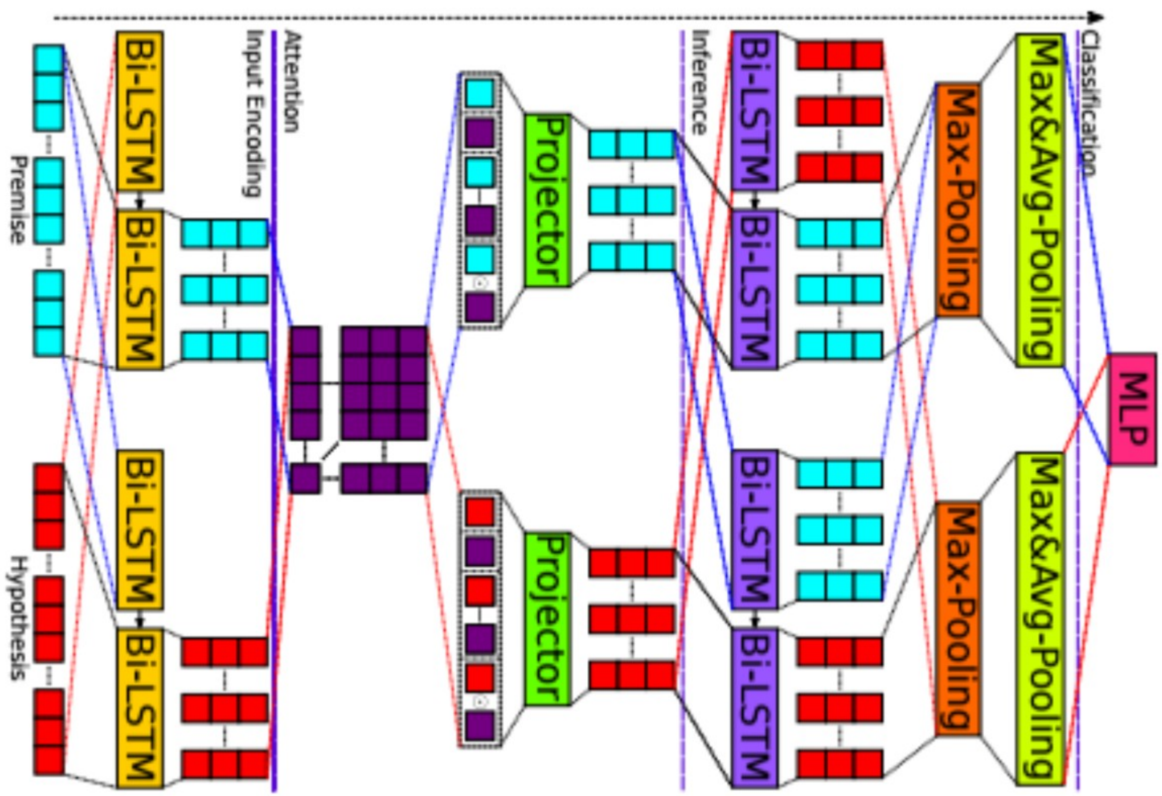
http://deeplearning.net/tutorial/lstm.html

BiLSTM



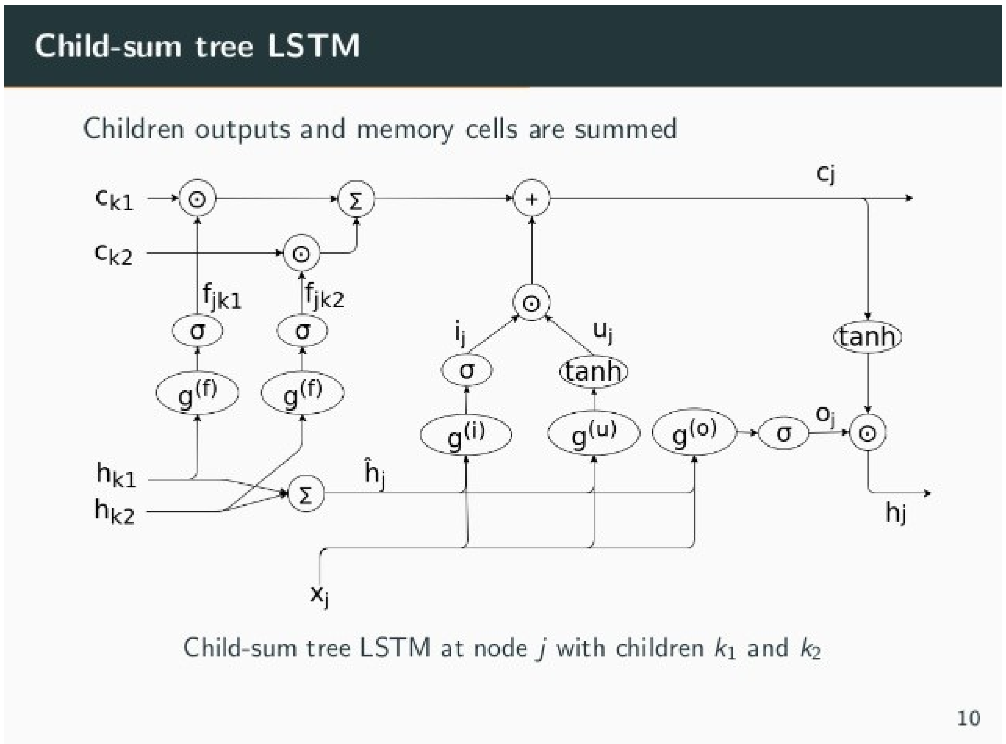
Output Layer
Backward Layer
Forward Layer
Input Layer
w6
w6
w6
w5
w5
w5
w5
w4
w4
w4
w3
w3
w3
w2
w2
w2
w2
w1
w1
w1

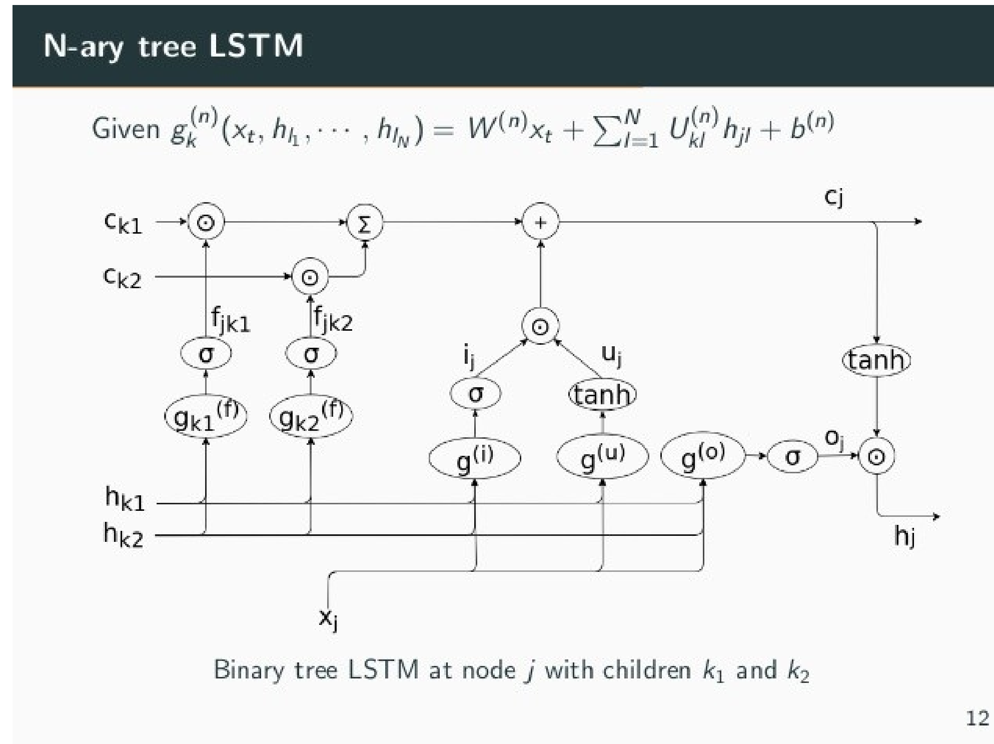
BiLSTM Example: Machine Reading



https://arxiv.org/pdf/1802.05577.pdf

TreeLSTM





https://www.slideshare.net/tuvistavie/tree-lstm

TreeLSTM Example

Semantic relatedness

Task

Predict similarity score in $[1, K]$ between two sentences

Method

Similarity between sentences L and R annotated with score $\in [1, 5]$

- Produce representations h_L and h_R
- Compute distance h_+ and angle $h_\times$ between h_L and h_R
- Compute score using fully connected NN

$$h_s = \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right)$$

$$\hat{p}_\theta = \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right)$$

$$\hat{y} = r^T \hat{p}_\theta \qquad r = [1, 2, 3, 4, 5]$$

- Error is computed using KL-divergence

LSTM Deficiencies

* computation not parallelizable
* neuron interpretability

Improvements/alternatives:

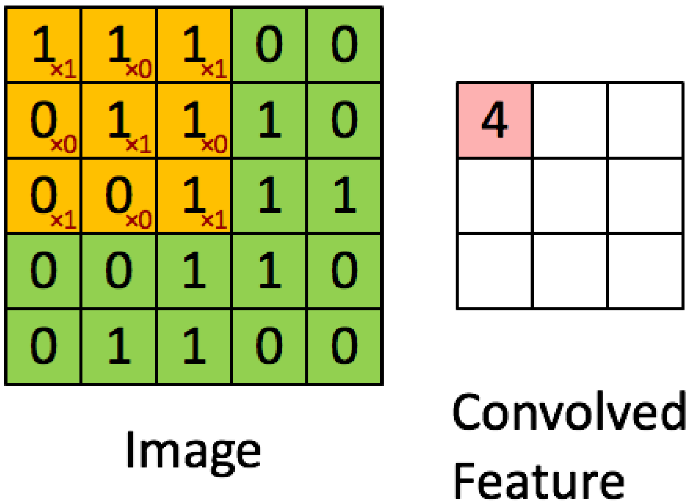
* RAN (Recurrent Additive Network)
  https://arxiv.org/pdf/1705.07393.pdf
* Janet (just the forget gate)
  https://arxiv.org/pdf/1804.04849.pdf
* QRNN (Quasi-recurrent Neural Network)
  https://goo.gl/NUx7VC

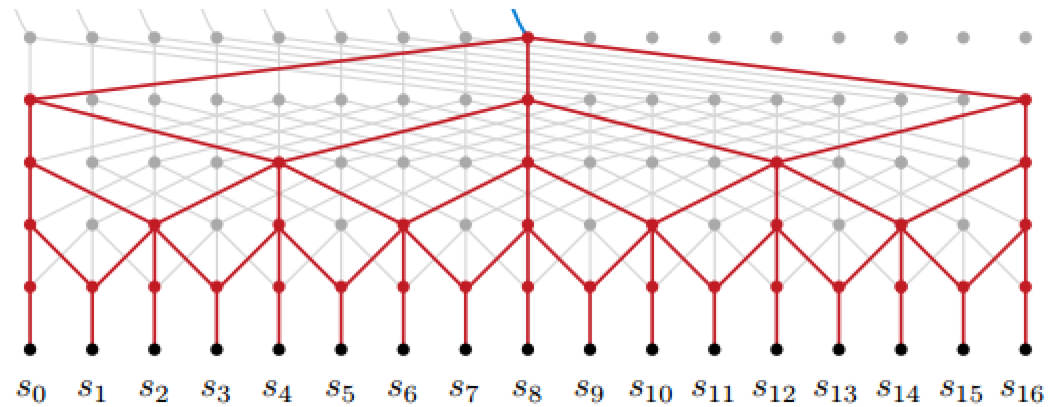
BiLSTM+attention as SOTA

“Basically, if you want to do an NLP task, no matter what it is, what you should do is throw your data into a Bi-directional long-short term memory network, and augment its information flow with the attention mechanism.”
— Chris Manning

https://twitter.com/mayurbhangale/status/988332845708886016

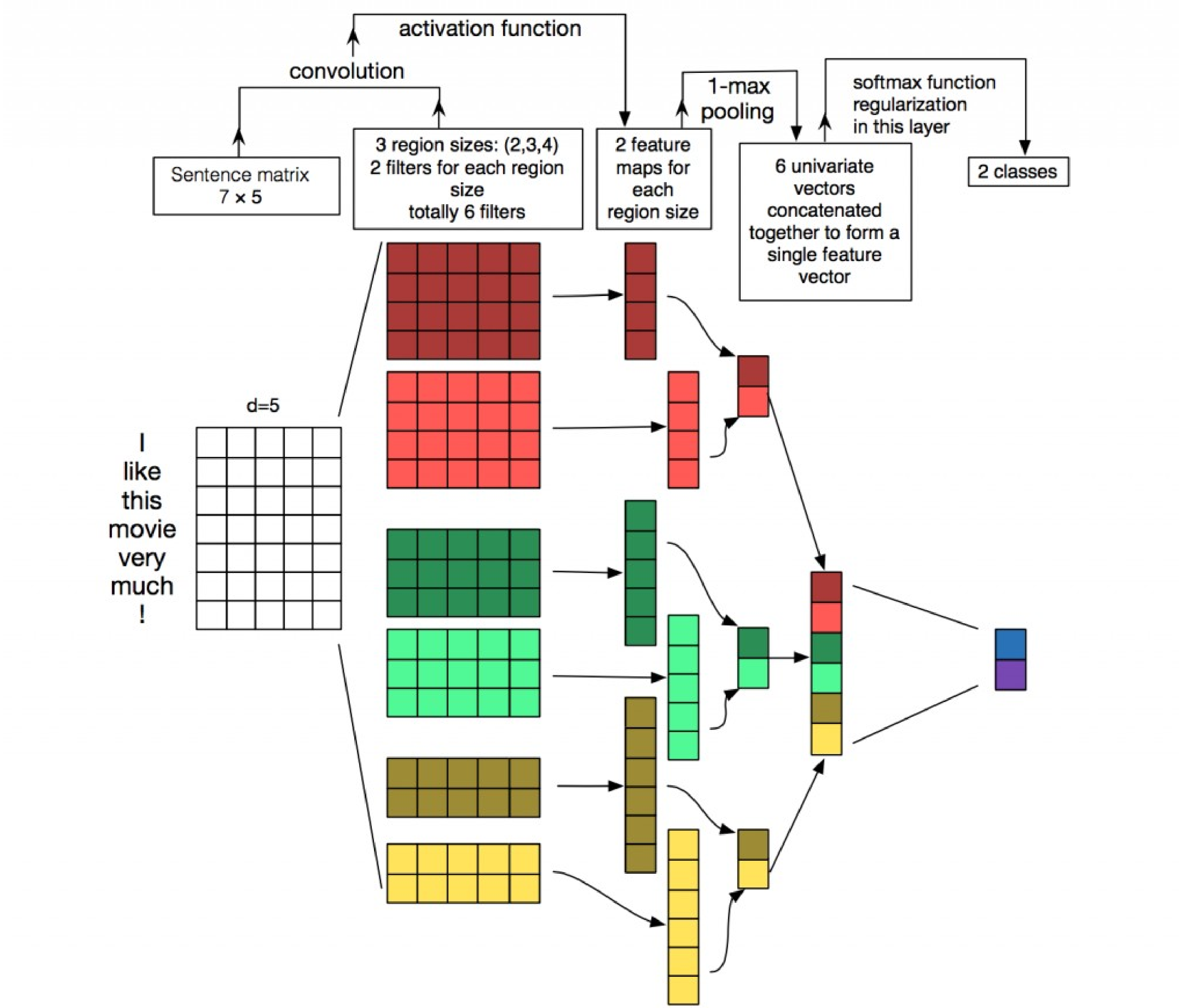
Convolutional Neural Networks (CNNs)





* convolutions can do the same as RNNs but faster
* any part of a sentence can influence the semantics of a word
  So we want our network to see the entire input at once
* getting that big a receptive can make gradients vanish
  and our networks fail
* we can solve the vanishing gradient problem
  with DenseNets or Dilated Convolutions
* use “deconvolutions” to generate arbitrarily long outputs

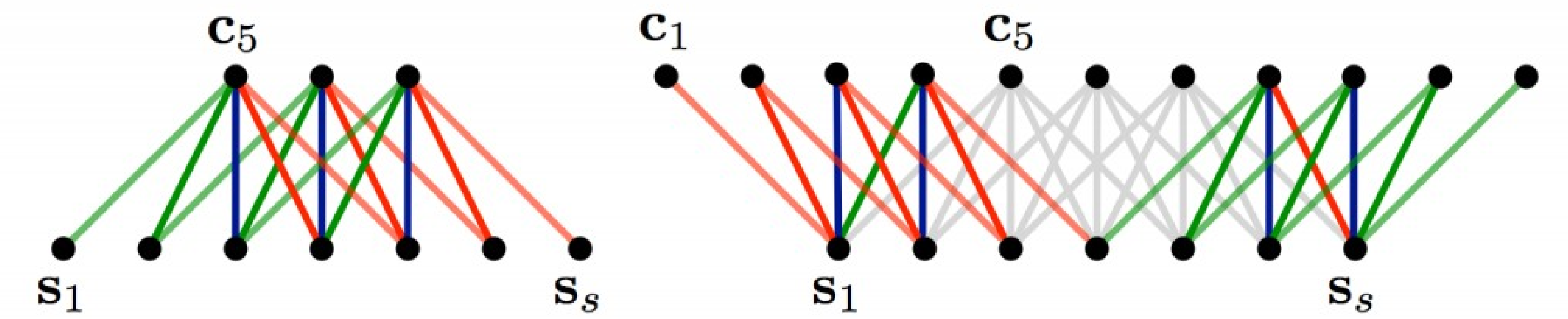
https://medium.com/@TalPerry/convolutional-methods-for-text-d5260fd5675f



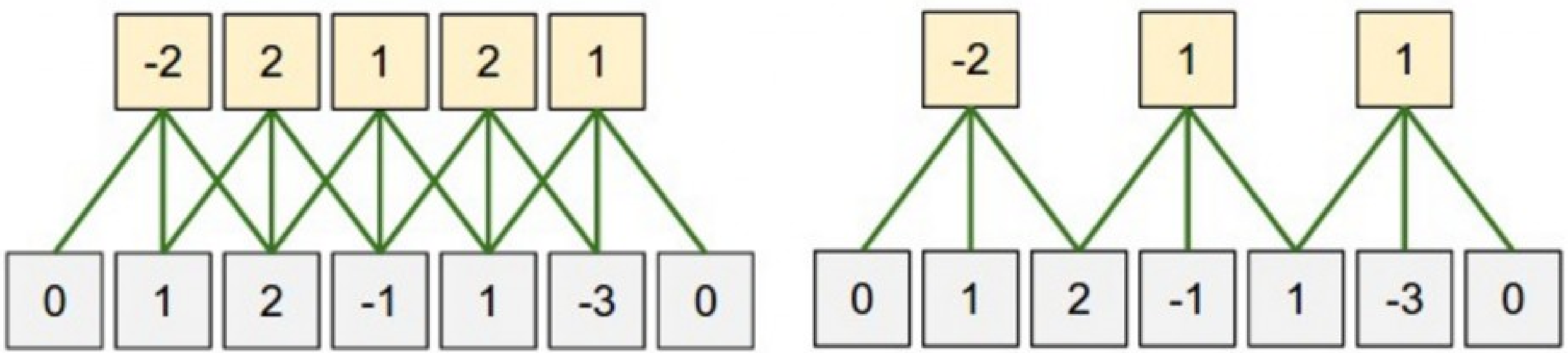
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

CNN Parameters

Wide vs narrow convolutions:



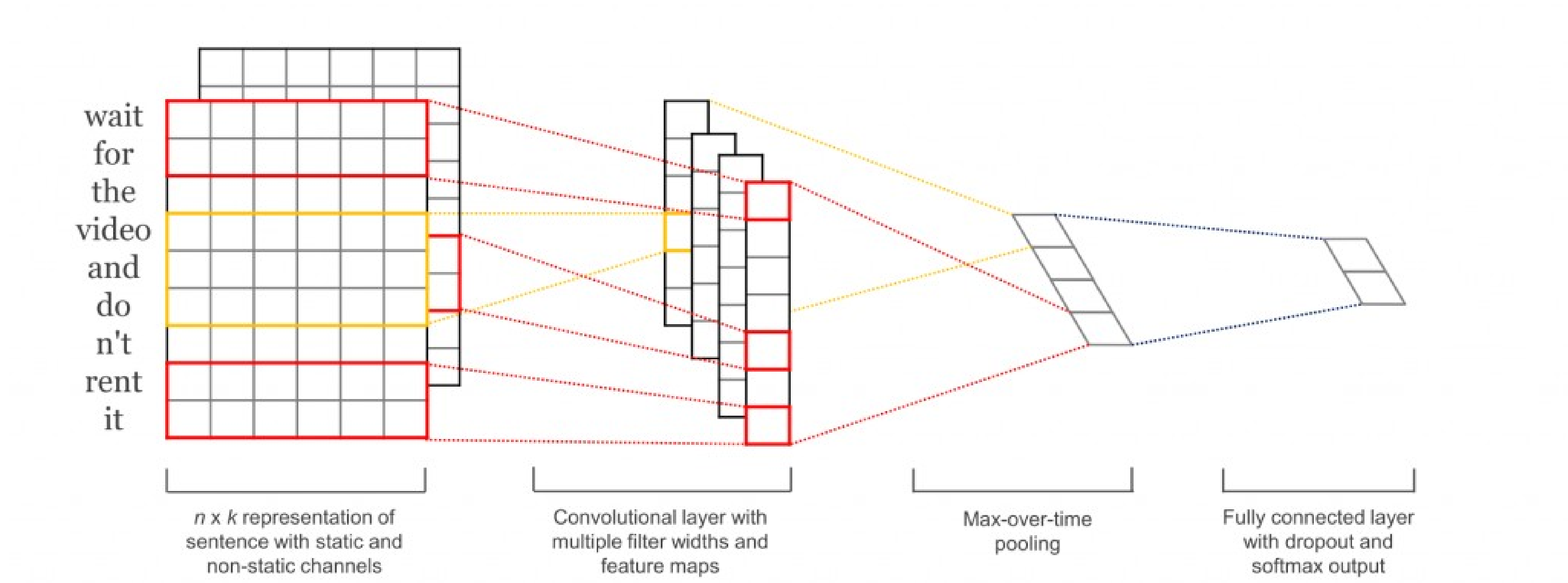
Strides:



Pooling:
- max
- mean

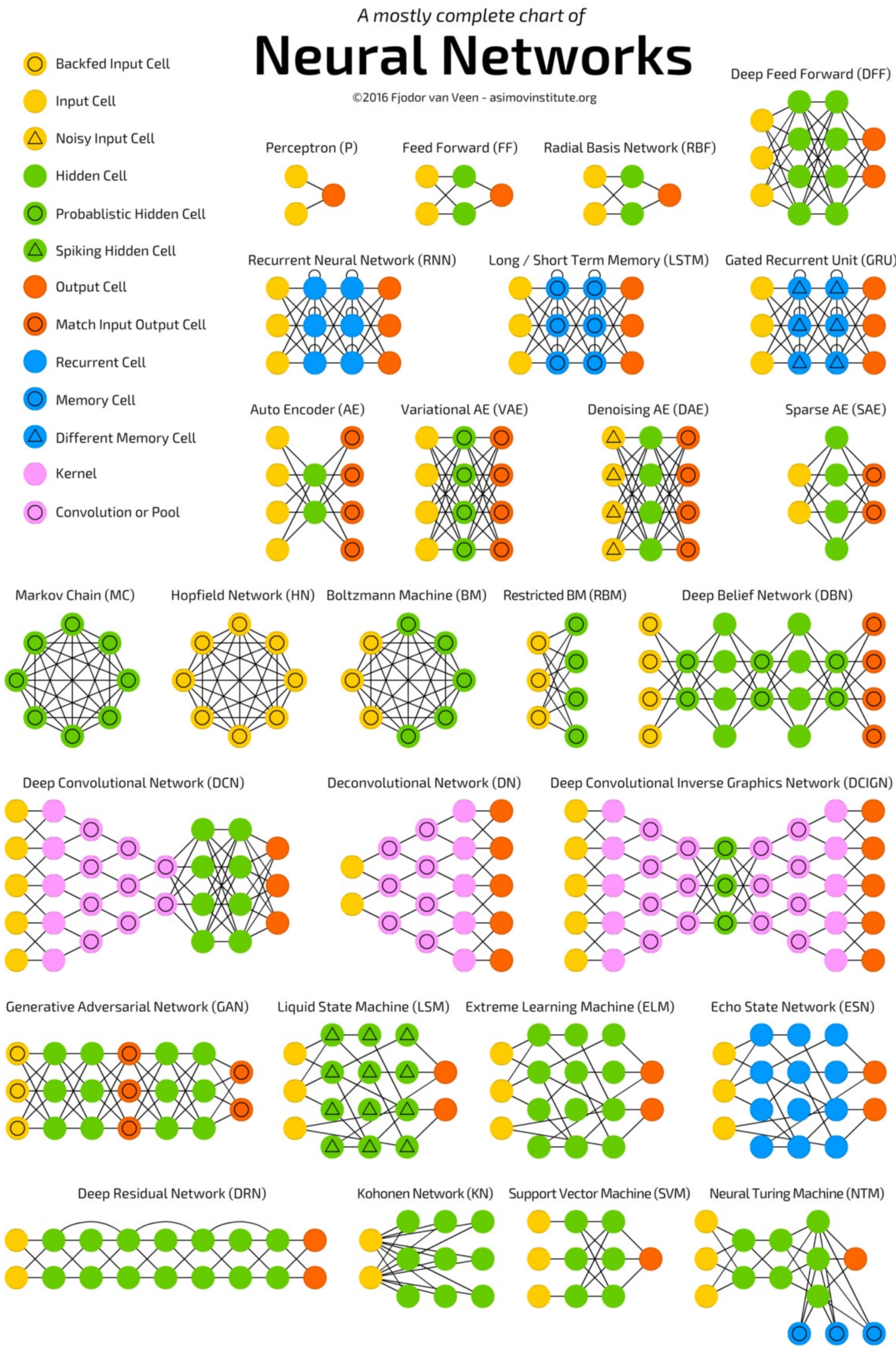
CNN Example

Classification (sentiment et al.)



CNNs Pros & Cons

+ fast (and furious)
+ don’t forget
+ view the whole input at once
+ can reuse a lot of tech from CV
- fixed input size
  (although RNNs, in fact, also suffer from it)
- harder to apply to sequence-based tasks
- non-generative


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Read More

Neural Nets for NLP:
http://cs231n.github.io
https://hackernoon.com/the-unreasonable-ineffectiveness-of-deep-learning-in-nlu-e4b4ce3a0da0
https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
https://blackboxnlp.github.io/

Nonlinearities:
https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9
https://medium.com/@jaiyamsharma/experiments-with-swish-activation-function-on-mnist-dataset-fc89a8c79ff7
https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464
http://building-babylon.net/2017/08/01/hierarchical-softmax/

Read More x2

Backprop & gradient descent:

https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

http://ruder.io/optimizing-gradient-descent/

https://openreview.net/pdf?id=ryQu7f-RZ

https://fosterelli.co/executing-gradient-descent-on-the-earth

https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94

RNN & LSTM:

https://deeplearning4j.org/lstm.html

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

https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9