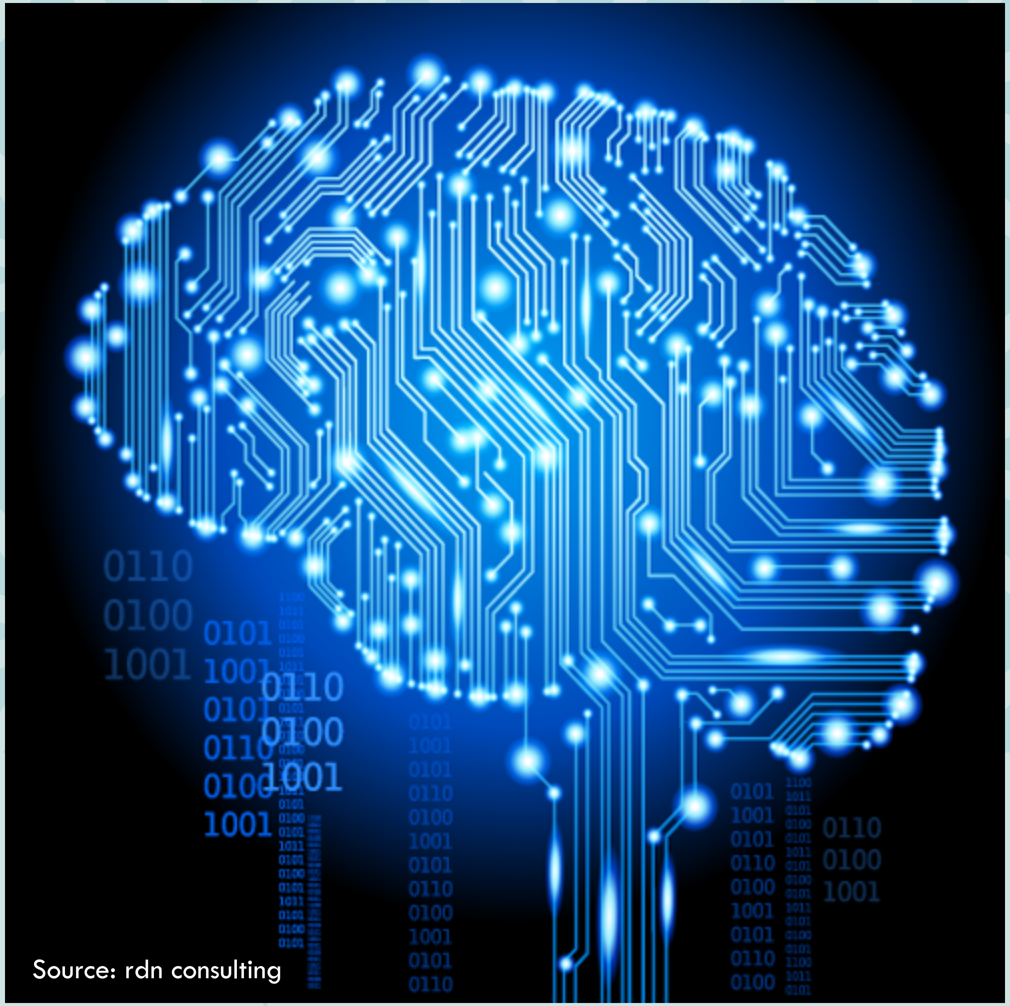
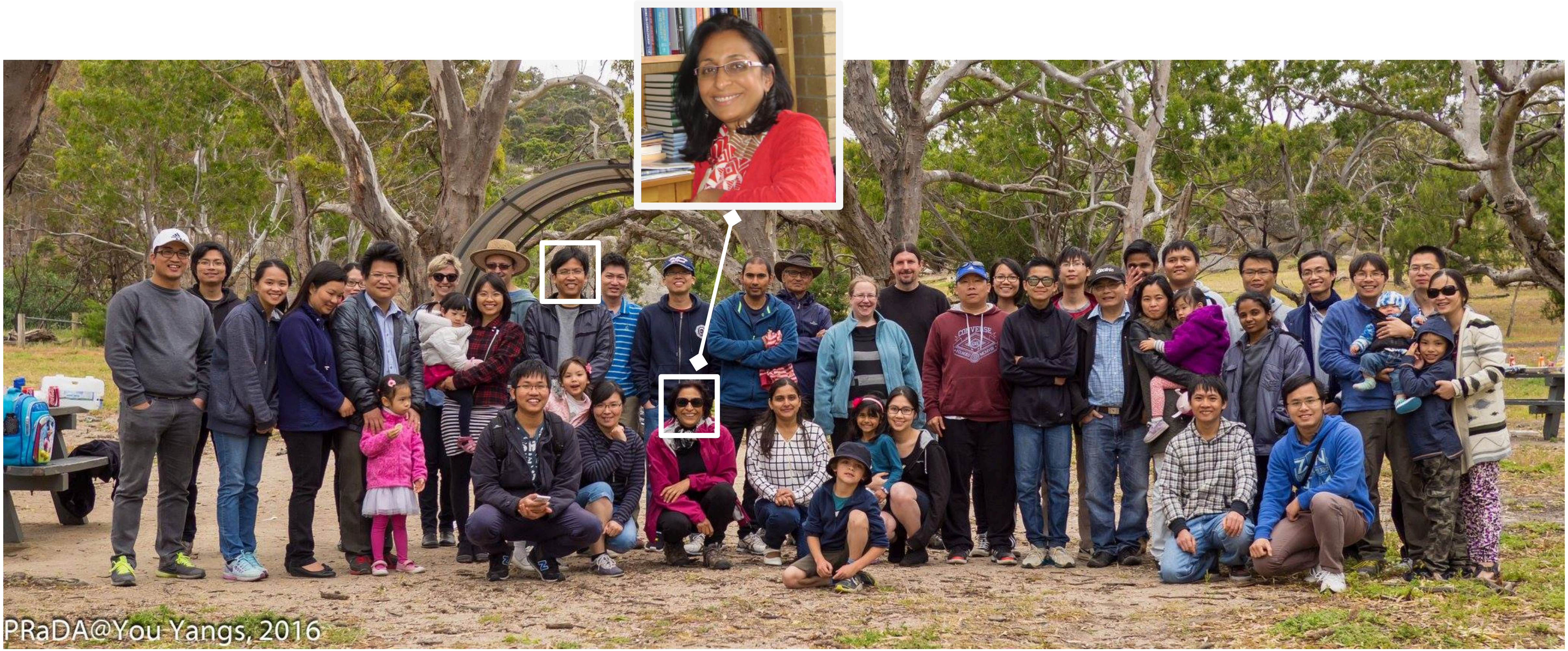
DEEP LEARNING & APPLICATIONS IN NON-COGNITIVE DOMAINS

Truyen Tran
Deakin University
truyen.tran@deakin.edu.au
prada-research.net/~truyen



AusDM'16, Canberra, Dec 7th 2016

PRADA @ DEAKIN, GEELONG CAMPUS

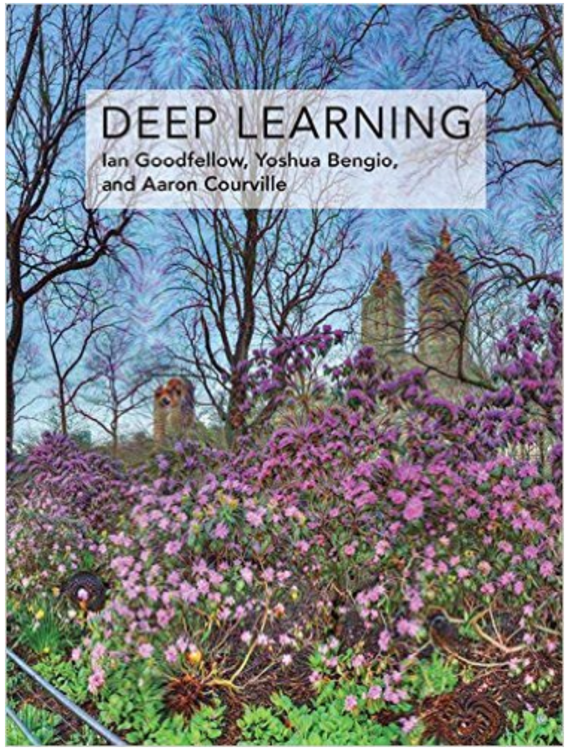


RESOURCES

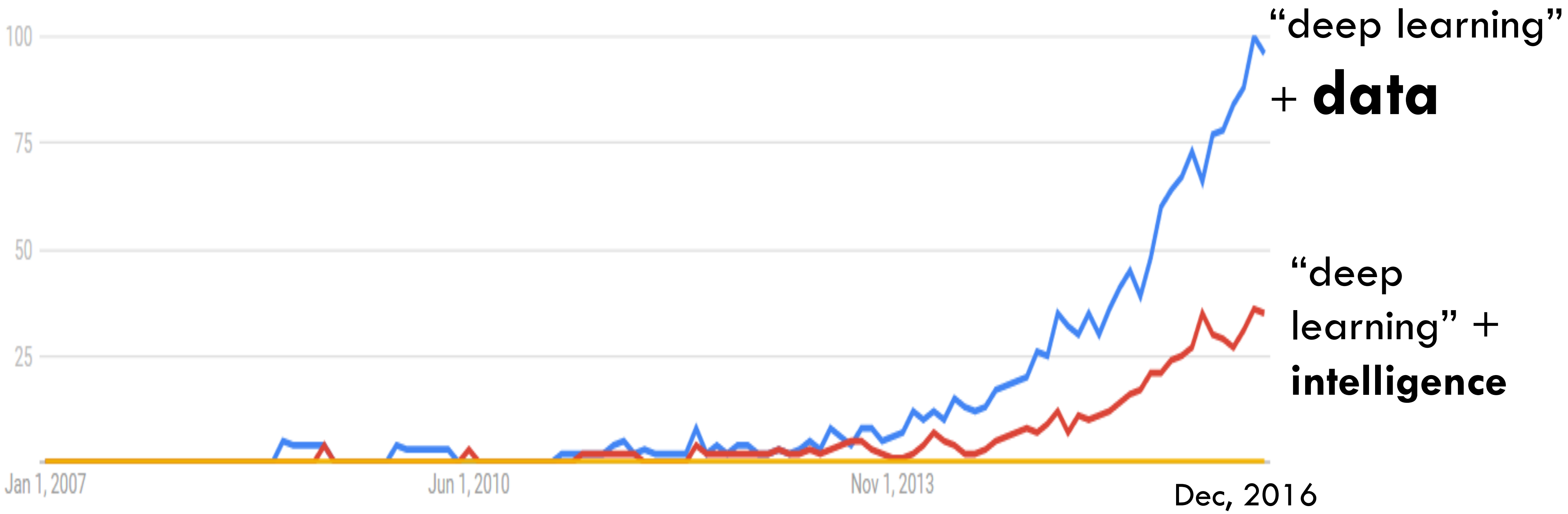
Tutorial page:

http://prada-research.net/~truyen/AusDM16-tute.html

Thanks to many people who have created beautiful graphics & open-source programming frameworks.



GOOGLE TRENDS

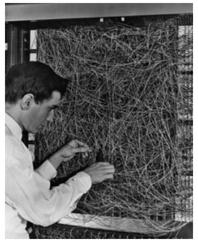


Yann LeCun

1988

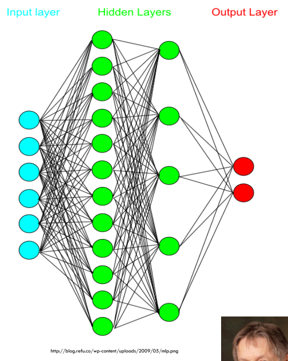
Geoff Hinton

2006



Rosenblatt's perceptron

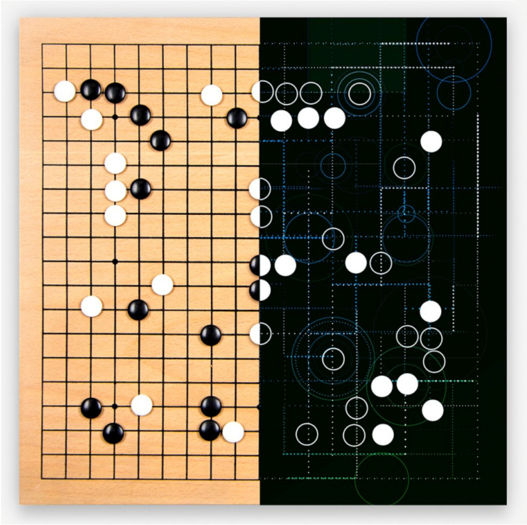
1958



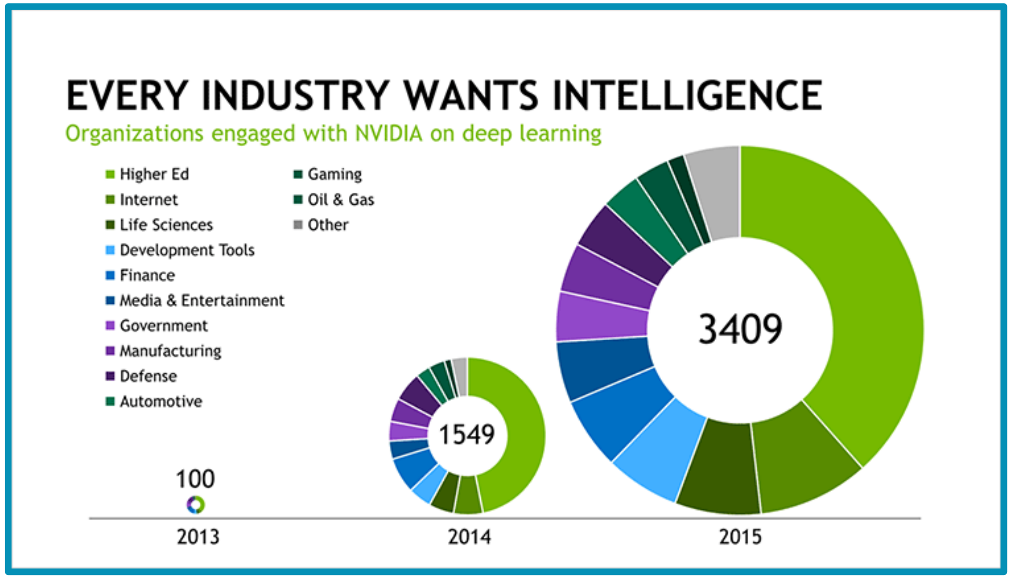
1986

2012

AlphaGo



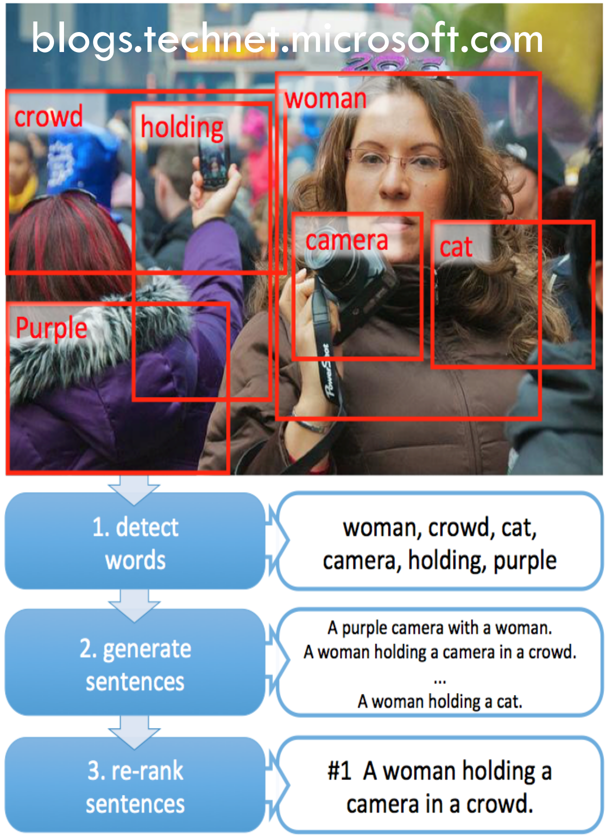
2016



DEEP LEARNING IN COGNITIVE DOMAINS

Google Translate

Where human can recognise, act or answer accurately within seconds



A → B

(ANDREW NG, BAIDU)

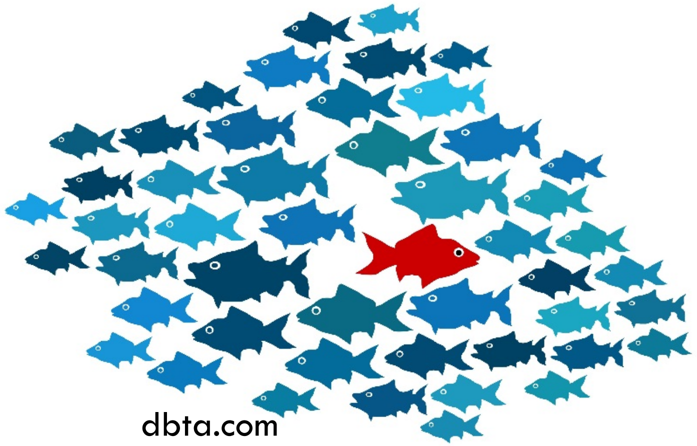
DEEP LEARNING IN NON-COGNITIVE DOMAINS

- **Where humans need extensive training to do well**
- Domains that demand transparency & interpretability.

… healthcare

… security

… genetics, foods, water …



AGENDA

Part I: Theory

Introduction to (mostly supervised) deep learning

Part II: Practice

Applying deep learning to non-cognitive domains

Part III: Advanced topics

Position yourself

PART I: THEORY
INTRODUCTION TO (MOSTLY SUPERVISED) DEEP LEARNING

Neural net as function approximation & feature detector

Three architectures: FFN → RNN → CNN

Bag of tricks: dropout → piece-wise linear units → skip-connections → adaptive stochastic gradient

PART II: PRACTICE
APPLYING DEEP LEARNING TO NON-COGNITIVE DOMAINS

Hand-on:

- Introducing programming frameworks (Theano, TensorFlow, Keras, Mxnet)

Domains how-to:

- Healthcare
- Software engineering
- Anomaly detection

PART III: ADVANCED TOPICS
POSITION YOURSELF

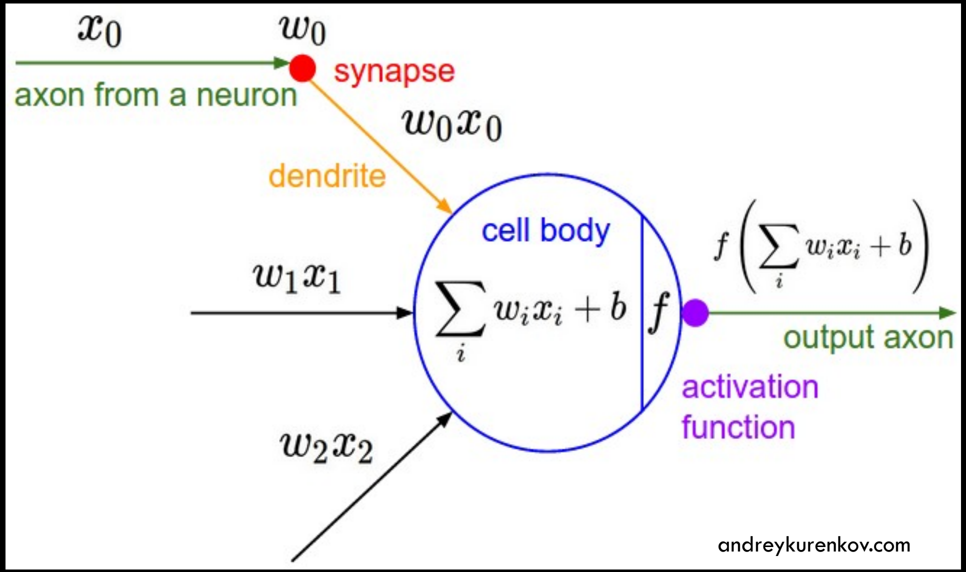
Unsupervised learning

Relational data & structured outputs

Memory & attention

How to position ourselves

PART I: THEORY

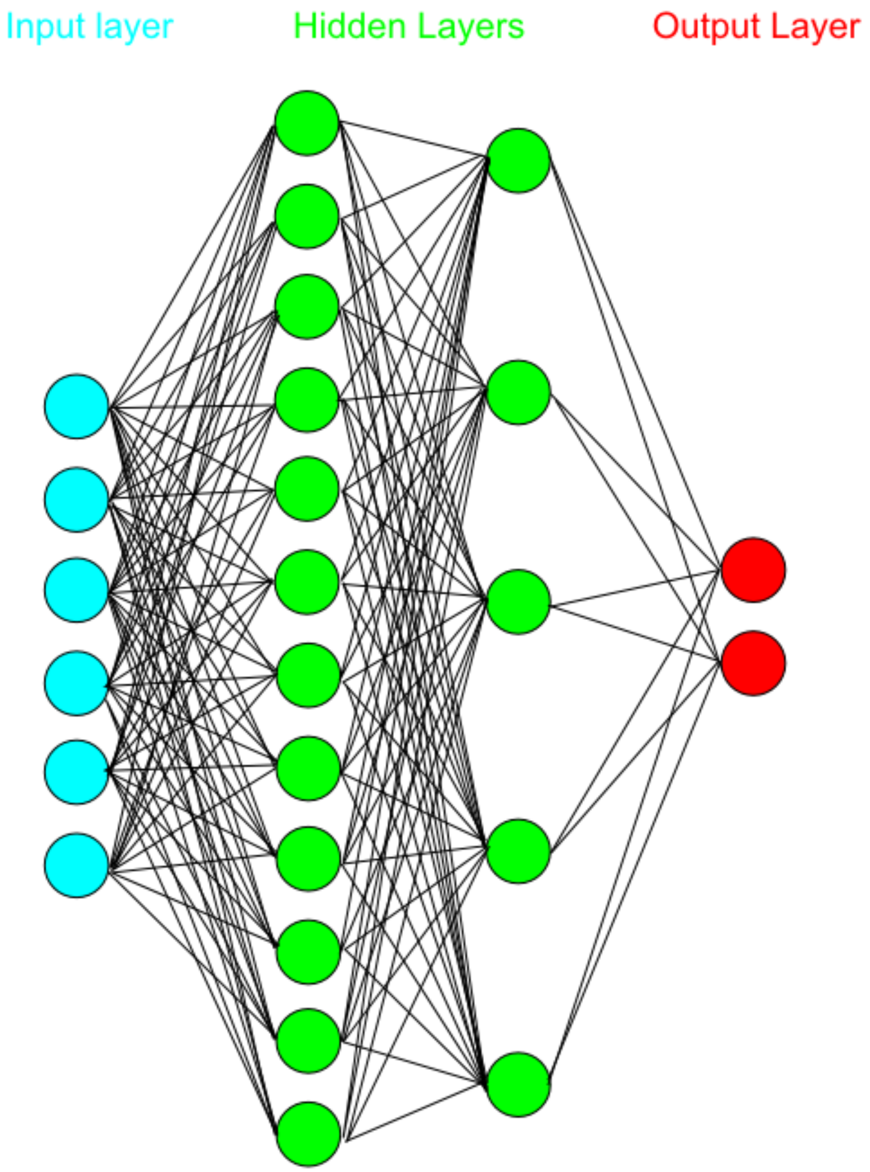


WHAT IS DEEP LEARNING?

Fast answer: multilayer perceptrons (aka deep neural networks) of the 1980s rebranded in 2006.

- But early nets go stuck at 1-2 hidden layers.

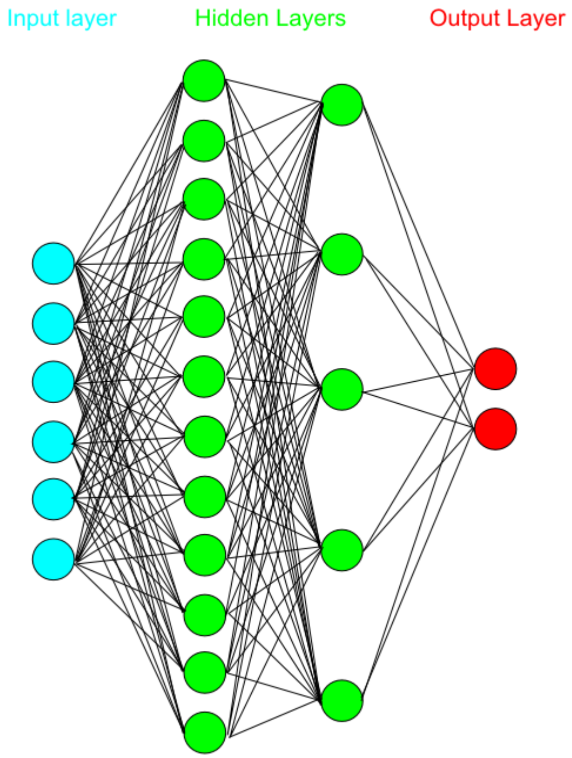
Slow answer: distributed representation, multiple steps of computation, modelling the compositionality of the world, a better prior, advances in **compute**, **data** & **optimization**, **neural architectures**, etc.



http://blog.refu.co/wp-content/uploads/2009/05/mlp.png

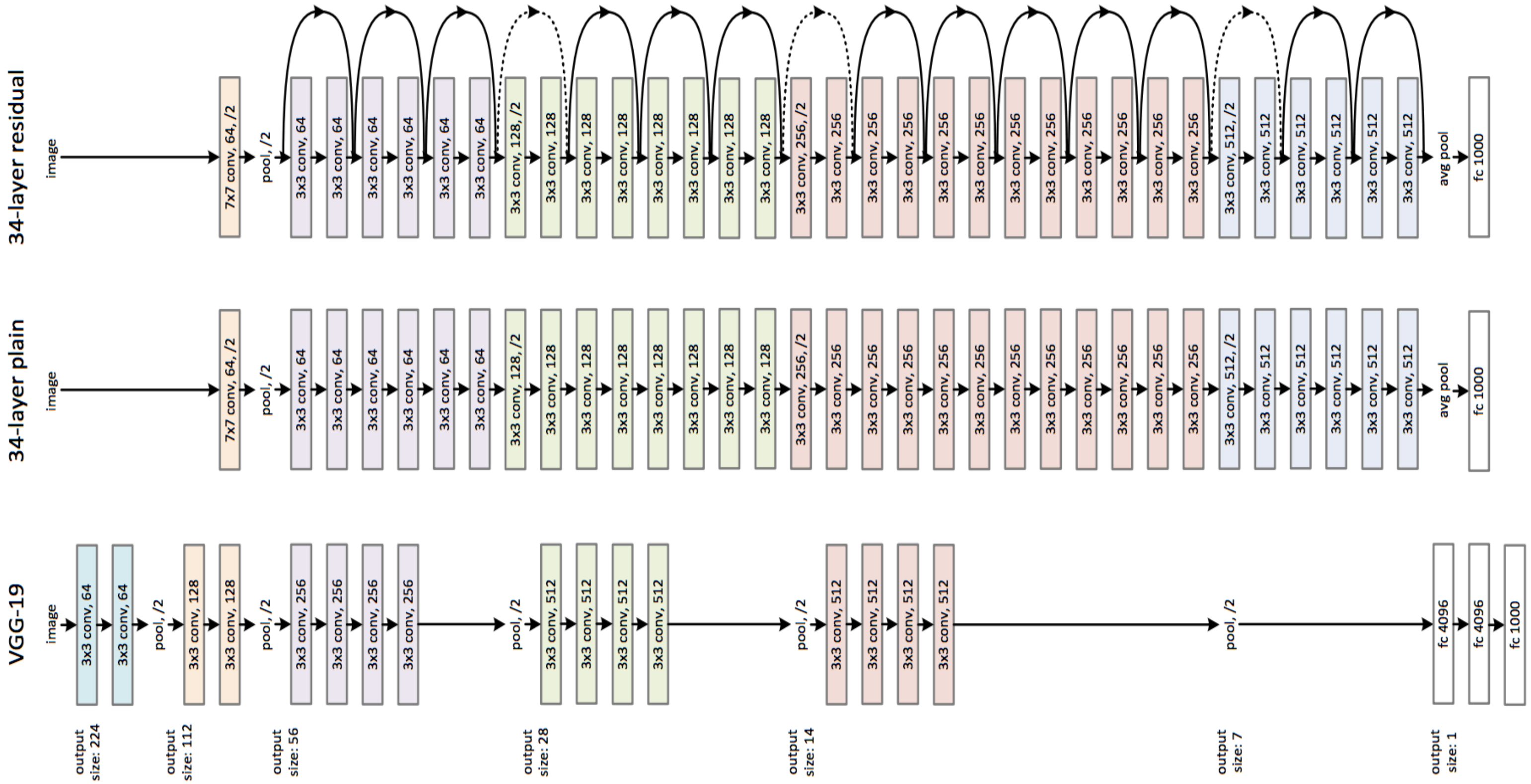
POWERFUL ARCHITECTURE: RESNET

1986



http://blog.refu.co/wp-content/uploads/2009/05/mlp.png

2015



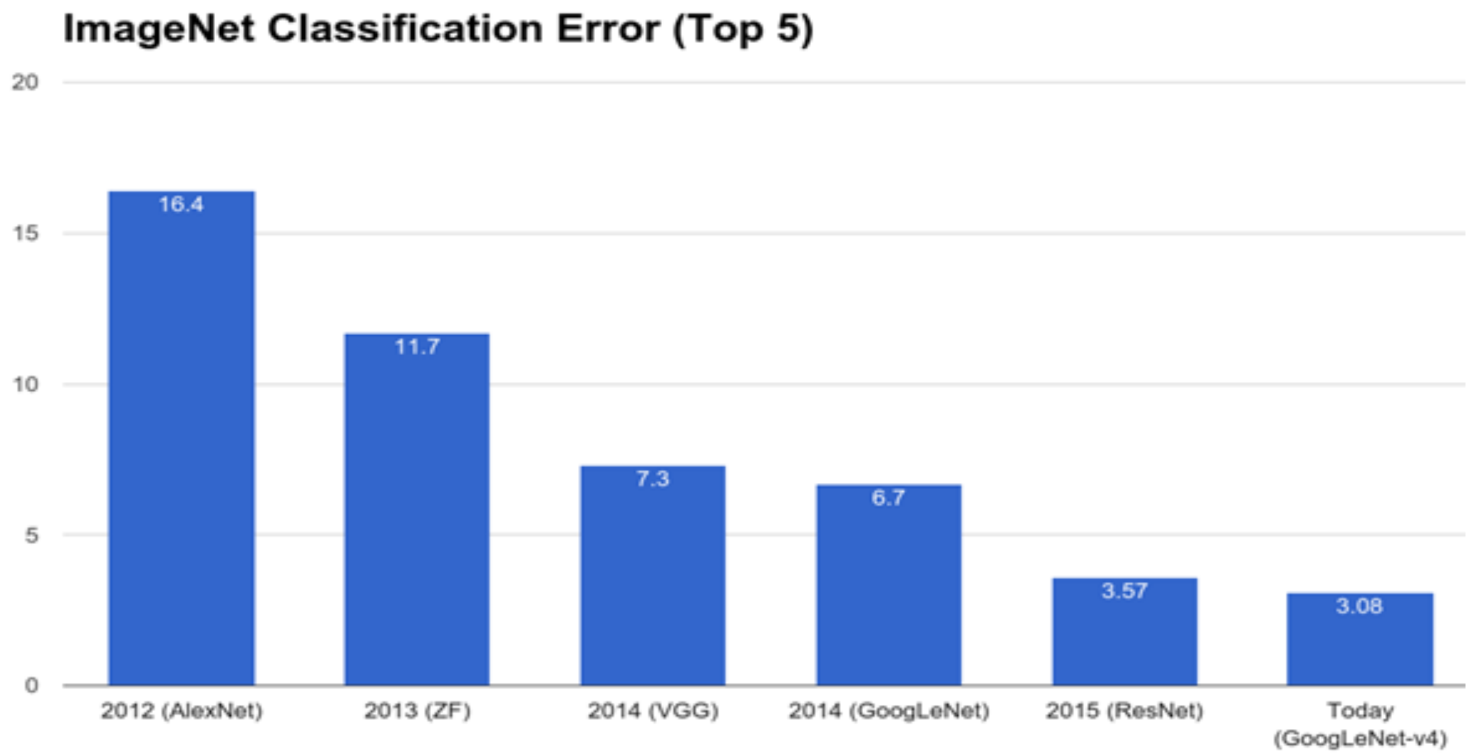
http://felixlaumon.github.io/2015/01/08/kaggle-right-whale.html

WHY DEEP LEARNING?

Because it works!
- Mostly performance-driven

But why does it work?
- Theory under-developed

Let's examine learning principles first.



https://www.quora.com/What-is-the-state-of-the-art-today-on-ImageNet-classification-In-other-words-has-anybody-beaten-Deep-Residual-Learning

RECAP: THE BEST OF MACHINE LEARNING

Strong/flexible priors:

- Good features → Feature engineering
- Data structure → HMM, CRF, MRF, Bayesian nets
- Model structure, VC-dimension, regularisation, sparsity → SVM, compressed sensing
- Manifold assumption, class/region separation → Metric + semi-supervised learning
- Factors of variation → PCA, ICA, FA

Uncertainty quantification: Bayesian, ensemble → RF, GBM

Sharing statistical strength: model reuse → transfer learning, domain adaption, multitask learning, lifelong learning

PRACTICAL REALISATION

More data

More GPUs

Bigger models

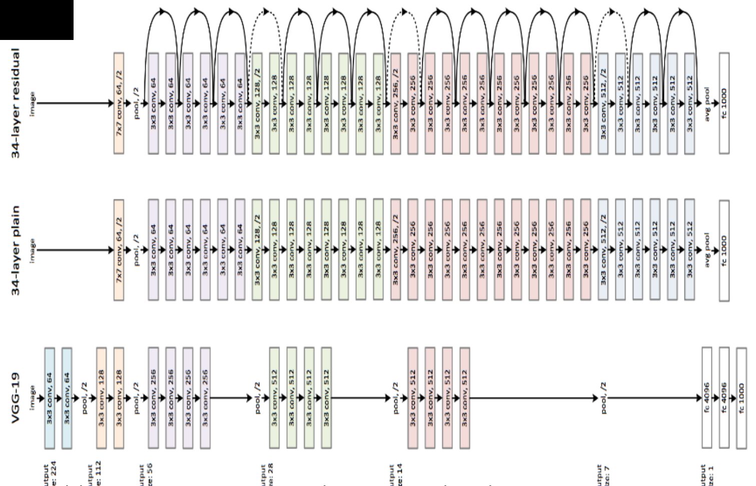
Better models

Faster iterations

Pushing the limit of patience (best models take 2-3 weeks to run)

A LOT OF NEW TRICKS

Data as new fuel



http://felixlaumon.github.io/2015/01/08/kaggle-right-whale.html

STARTING POINT: FEATURE LEARNING

In typical machine learning projects, 80-90% effort is on <u>feature engineering</u>

- A right feature representation doesn't need much work. Simple linear methods often work well.

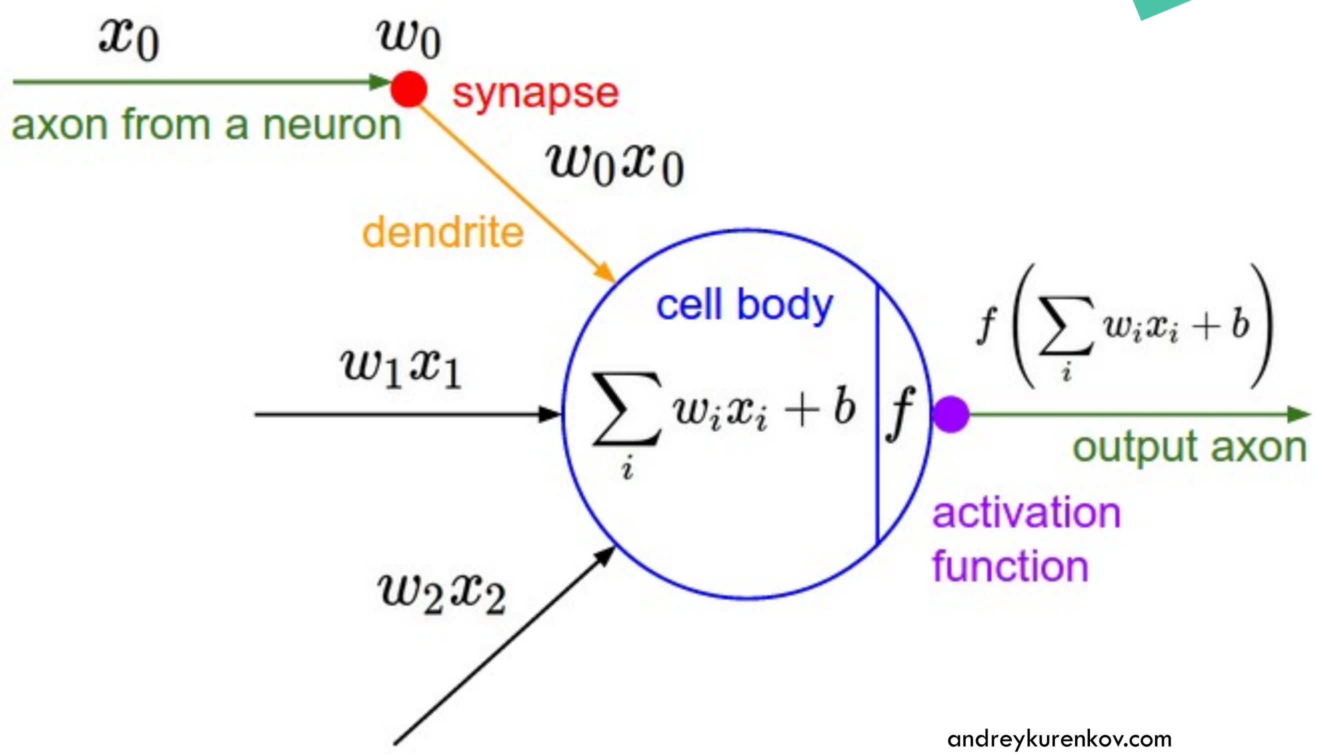
Text: BOW, n-gram, POS, topics, stemming, tf-idf, etc.

SW: token, LOC, API calls, #loops, developer reputation, team complexity, report readability, discussion length, etc.

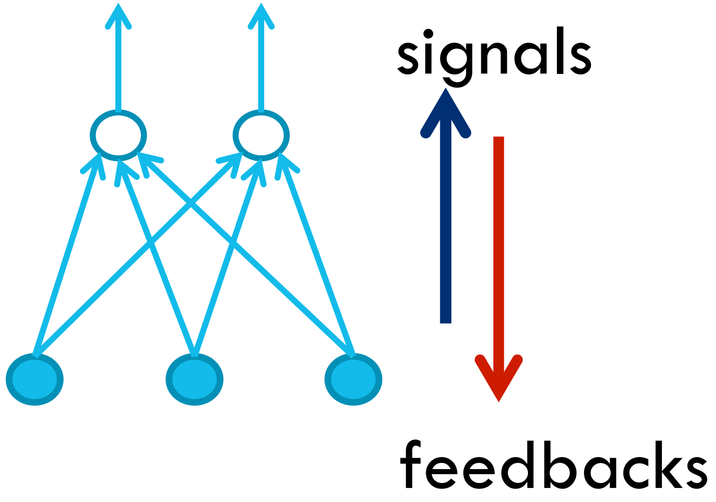
Try yourself on Kaggle.com!

THE BUILDING BLOCKS: FEATURE DETECTOR

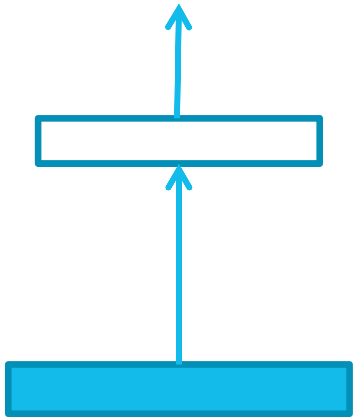
Integrate-and-fire neuron



andreykurenkov.com



Feature detector



Block representation

THREE MAIN ARCHITECTURES: FNN, RNN & CNN

Feed-forward (FFN): Function approximator (Vector to vector)

- Most existing ML/statistics fall into this category

Recurrent (RNN): Sequence to sequence

- Temporal, sequential. E.g., sentence, actions, DNA, EMR
- Program evaluation/execution. E.g., sort, traveling salesman problem

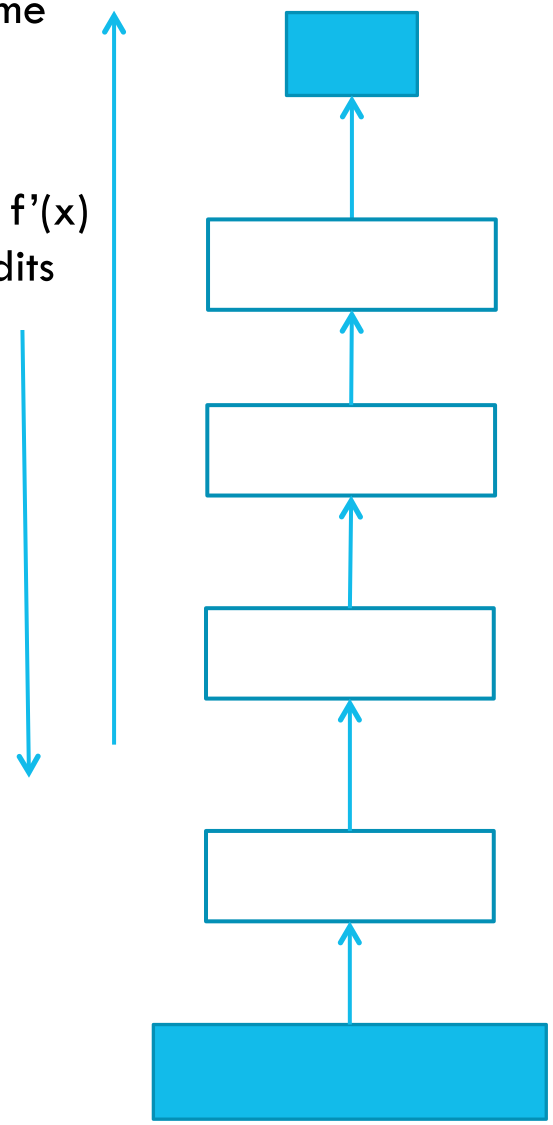
Convolutional (CNN): Image to vector/sequence/image

- In time: Speech, DNA, sentences
- In space: Image, video, relations

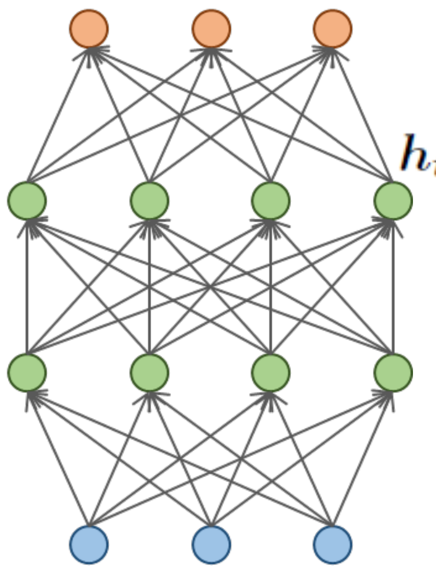
FEED-FORWARD NET (FFN)
VEC2VEC MAPPING

Forward pass: f(x) that carries units' contribution to outcome

Backward pass: f'(x) that assigns credits to units



$$P\left(\tilde{y}=i \mid \boldsymbol{x}\right)=\operatorname{softmax}\left(\boldsymbol{h}_{T+1}^{i}\right)=\frac{e^{h_{T+1}^{i}}}{\sum_{j} e^{h_{T+1}^{j}}}$$



$$\boldsymbol{h}_t=g\left(W_t \boldsymbol{h}_{t-1}+\boldsymbol{b}_t\right)$$

Problems with depth:

Multiple chained non-linearity steps

→ neural saturation (top units have little signal from the bottom)

→ vanishing gradient (bottom layers have much less training information from the label).

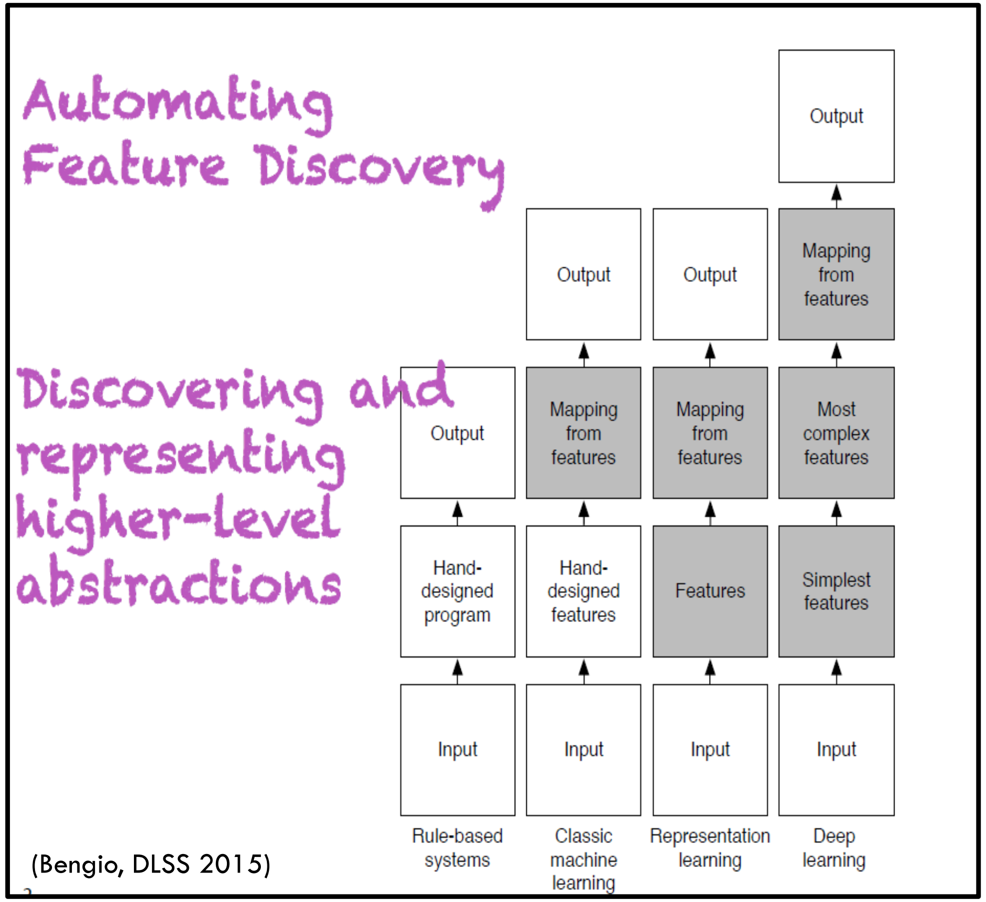
TWO VIEWS OF FNN

- The functional view: FNN as <u>nested composition</u> of functions

$$P\left(\tilde{y}=i \mid \boldsymbol{x}\right)=\operatorname{softmax}\left(\boldsymbol{h}_{T+1}^{i}\right)=\frac{e^{\boldsymbol{h}_{T+1}^{i}}}{\sum_{j} e^{\boldsymbol{h}_{T+1}^{j}}}$$

$$\boldsymbol{h}_{t}=g\left(W_{t} \boldsymbol{h}_{t-1}+\boldsymbol{b}_{t}\right)$$

- The representation view: FNN as <u>staged abstraction</u> of data



SOLVING PROBLEM OF VANISHING GRADIENTS

Principle: Enlarging the channel to pass feature and gradient

Method 1: Removing saturation with piecewise linear units

- Rectifier linear unit (ReLU)

Method 2: Explicit copying through gating & skip-connection

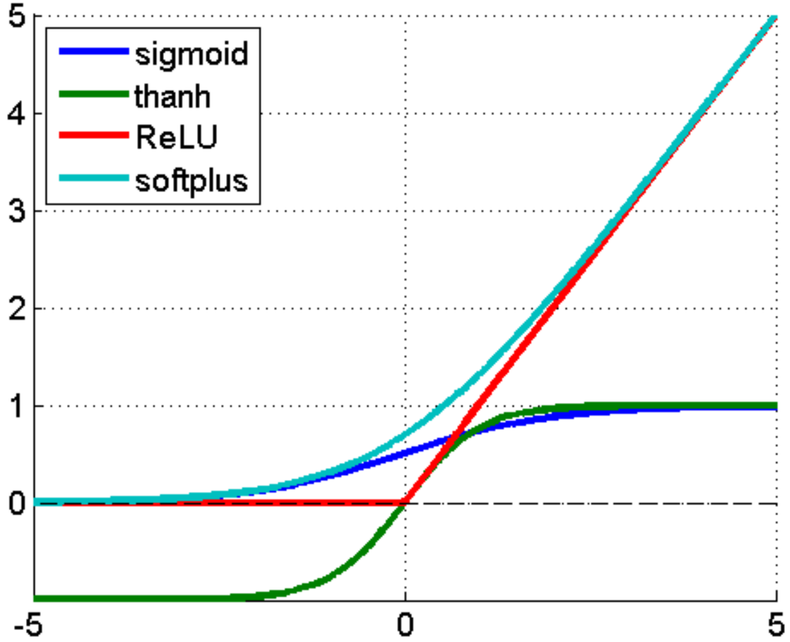
- Highway networks
- Skip-connection: ResNet

METHOD 1: RECTIFIER LINEAR TRANSFORMATION

The usual logistic and tanh transforms are saturated at infinity

The gradient approaches zero, making learning impossible

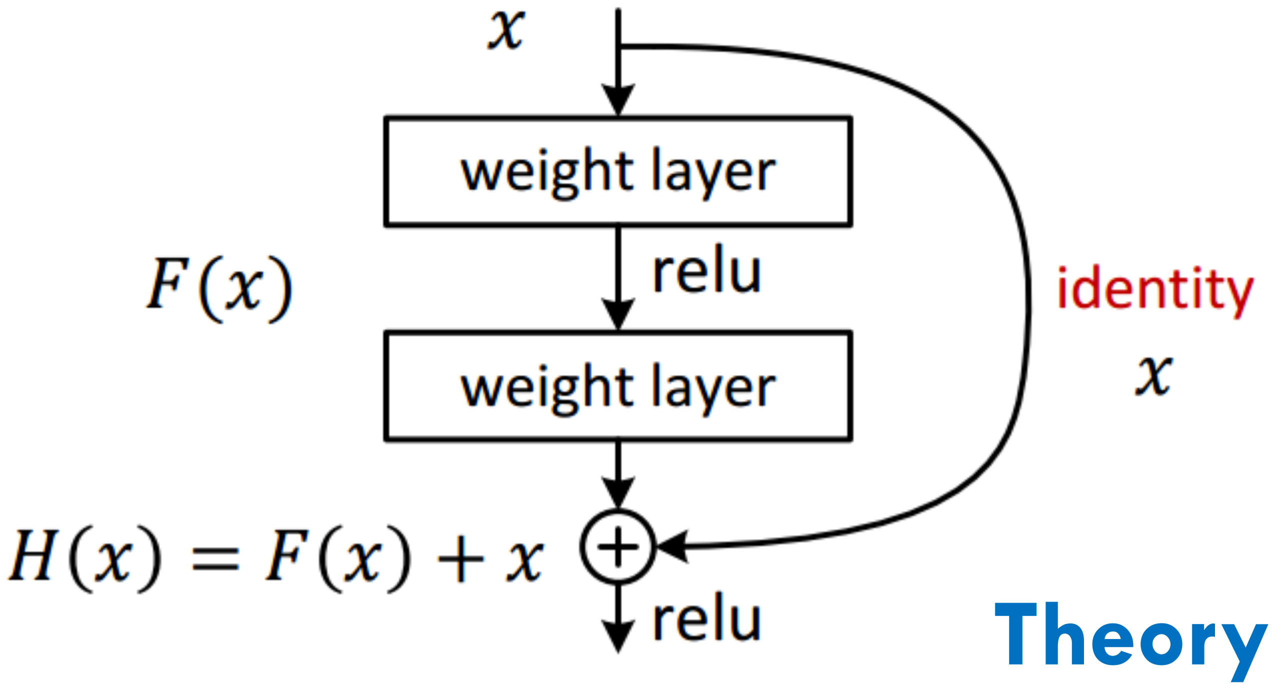
Rectifier linear function has constant gradier making information flows much better



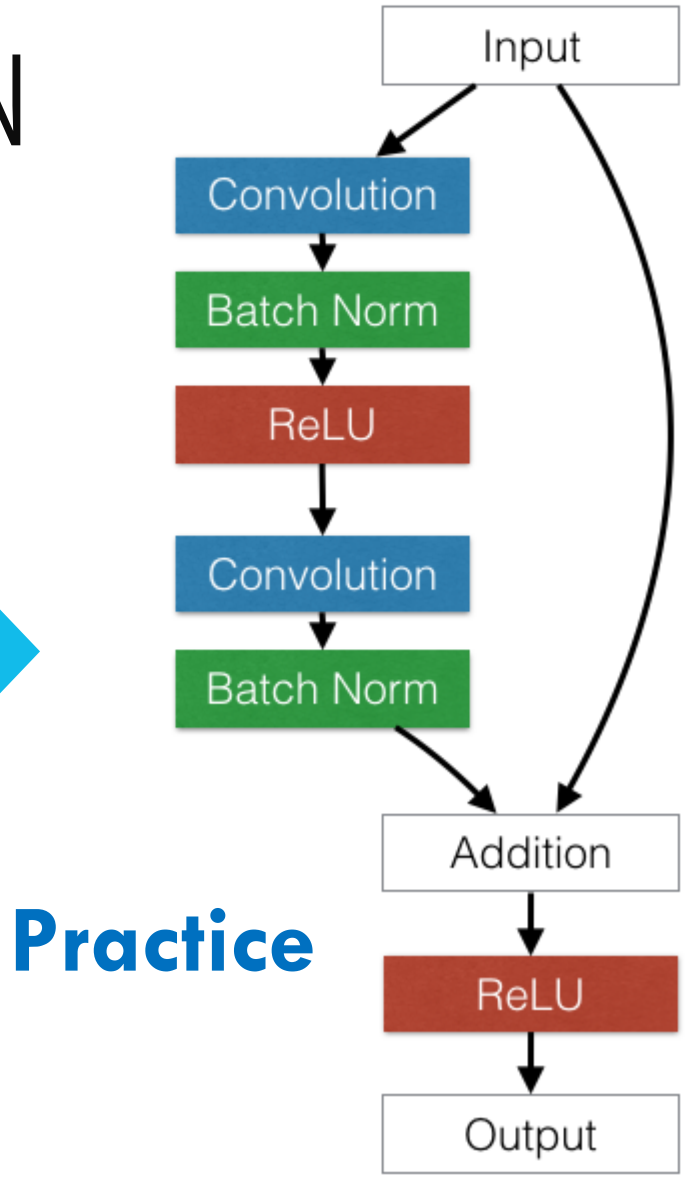
Source: https://imiloainf.wordpress.com/2013/11/06/rectifier-nonlinearities/

METHOD 2: SKIP-CONNECTION WITH RESIDUAL NET

- Residual net



http://qiita.com/supersaiakujin/items/935bbc9610d0f87607e8



http://torch.ch/blog/2016/02/04/resnets.html

THREE MAIN ARCHITECTURES: FNN, RNN & CNN

Feed-forward (FFN): Function approximator (Vector to vector)

- Most existing ML/statistics fall into this category

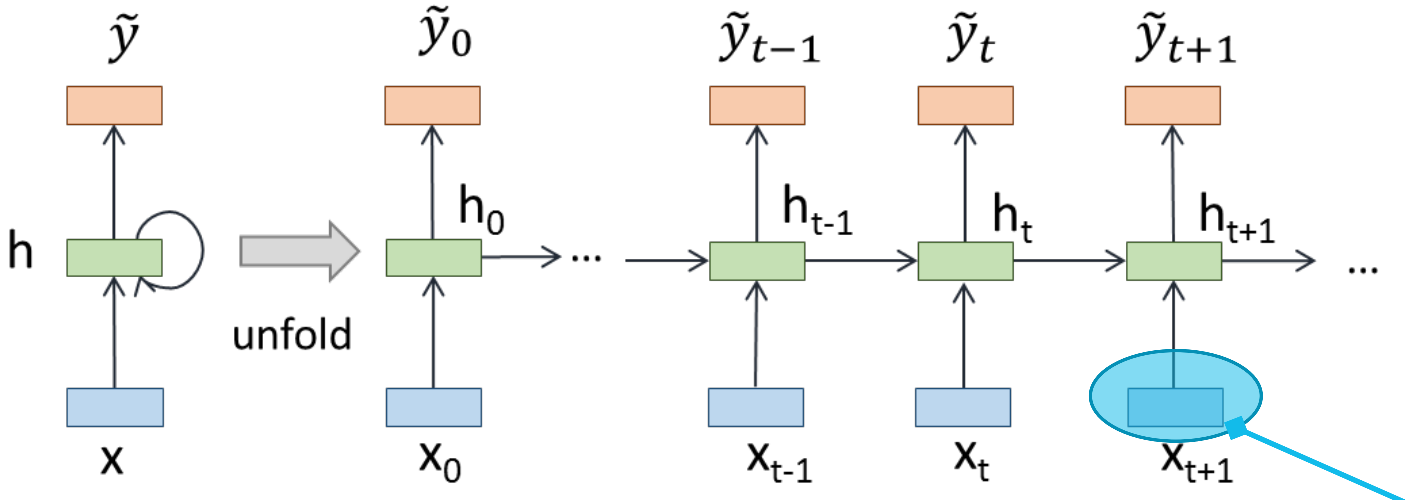
Recurrent (RNN): Sequence to sequence

- Temporal, sequential. E.g., sentence, actions, DNA, EMR
- Program evaluation/execution. E.g., sort, traveling salesman problem

Convolutional (CNN): Image to vector/sequence/image

- In time: Speech, DNA, sentences
- In space: Image, video, relations

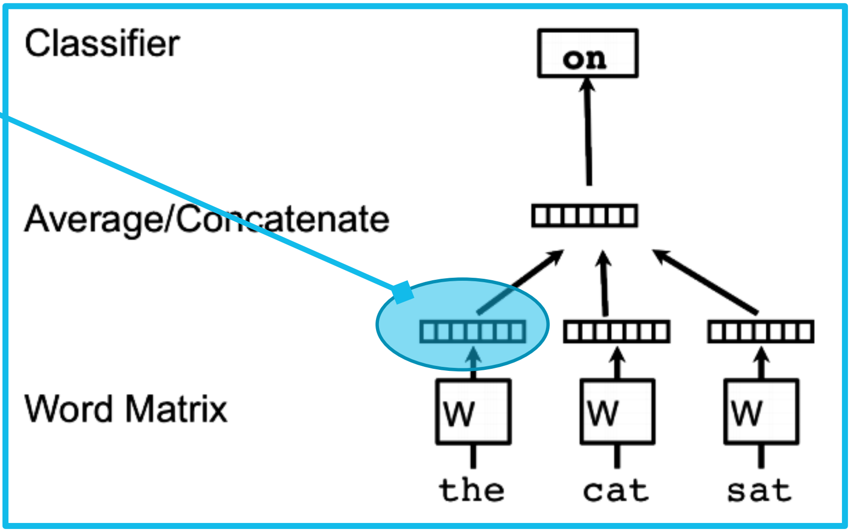
RECURRENT NET (RNN)



$$
\begin{aligned}
\boldsymbol{h}_t &= g\left(\boldsymbol{b} + W\boldsymbol{h}_{t-1} + U\boldsymbol{x}_t\right) \\
\boldsymbol{a}_t &= \boldsymbol{c} + V\boldsymbol{h}_t \\
P\left(\tilde{y}_t\right) &= \mathrm{f}_{prob}(\boldsymbol{a}_t)
\end{aligned}
$$

Example application: Language model which is essentially to predict the next word given previous words in the sentence.

Embedding is often used to convert a word into a vector, which can be initialized from **word2vec**.



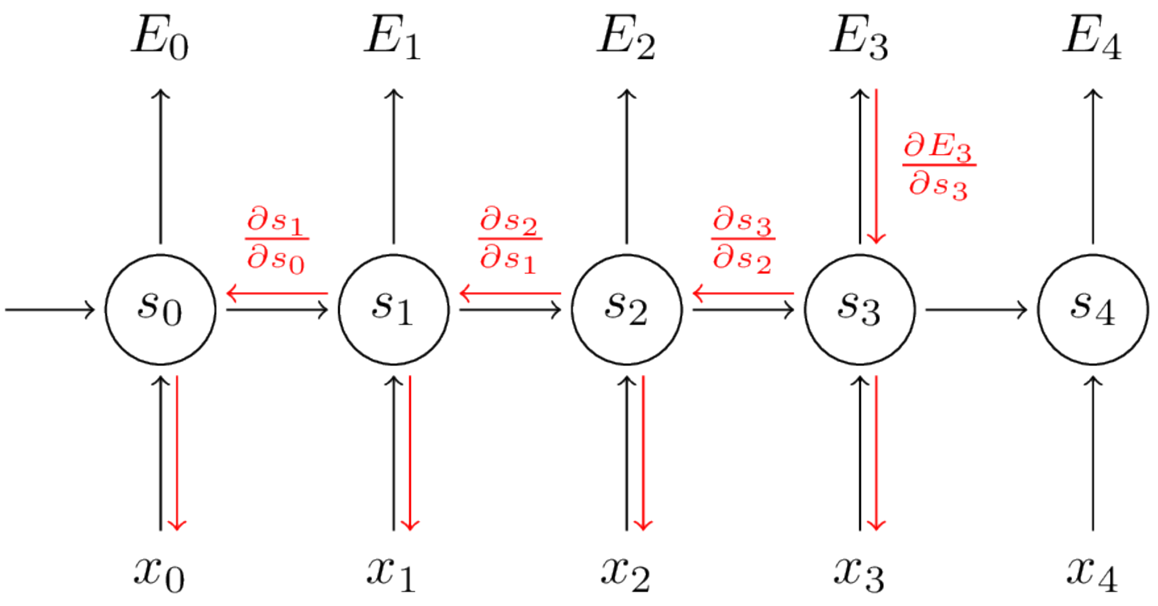
RNN IS POWERFUL BUT …

RNN is Turing-complete (Hava T. Siegelmann and Eduardo D. Sontag, 1991).

Brain can be thought of as a giant RNN over discrete time steps.

But training is very difficult for long sequences (more than 10 steps), due to:

- Vanishing gradient
- Exploding gradient



http://www.wildml.com/

SOLVING PROBLEM OF VANISHING/ EXPLODING GRADIENTS

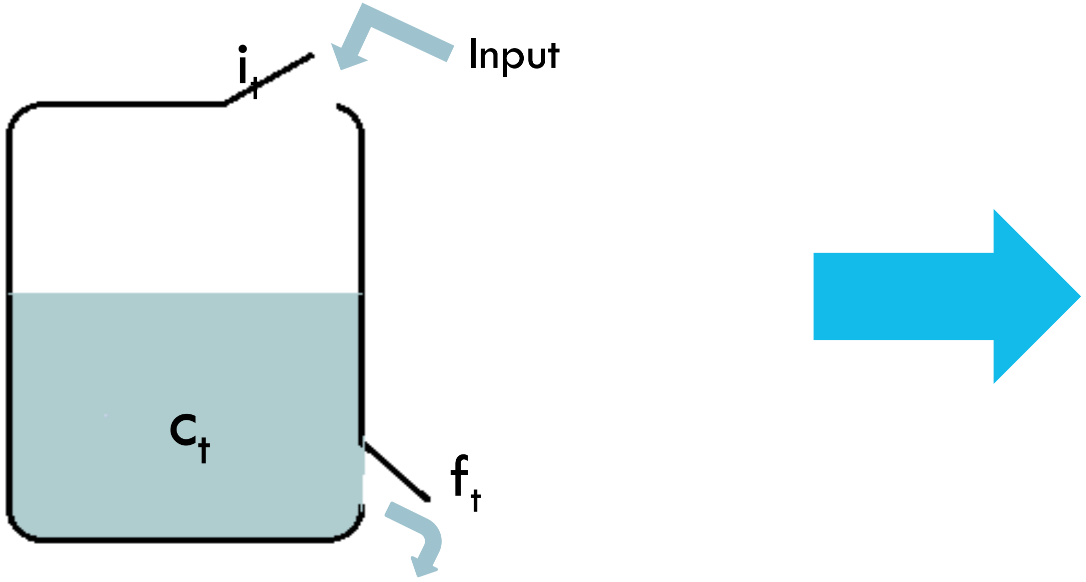
Trick 1: modifying the gradient, e.g., truncation for exploding gradient (not always works)

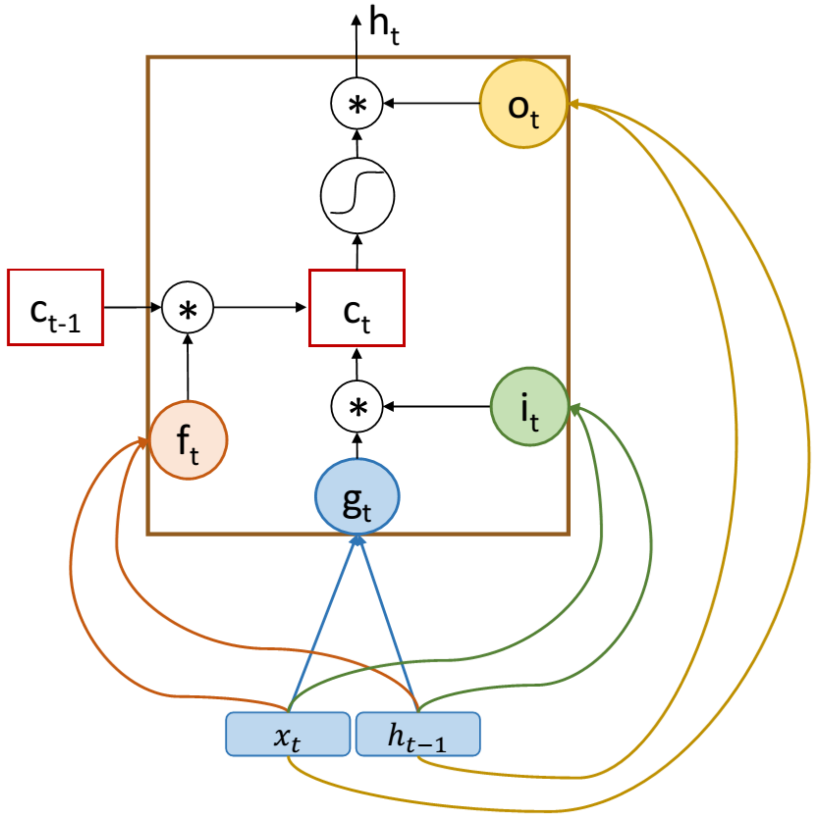
Trick 2: changing learning dynamics, e.g., adaptive gradient descent partly solves the problem (will see later)

Trick 3: modifying the information flow:

- Explicit copying through gating: Gated Recurrent Unit (GRU, 2014)
- Explicit memory: Long Short-Term Memory (LSTM, 1997)

LONG SHORT-TERM MEMORY (LSTM)





$$\boldsymbol{i}_t = \sigma\left(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right)$$
$$\boldsymbol{f}_t = \sigma\left(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right)$$
$$\boldsymbol{o}_t = \sigma\left(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right)$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t * \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * \tanh\left(W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right)$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{c}_t)$$

RNN: WHERE IT WORKS

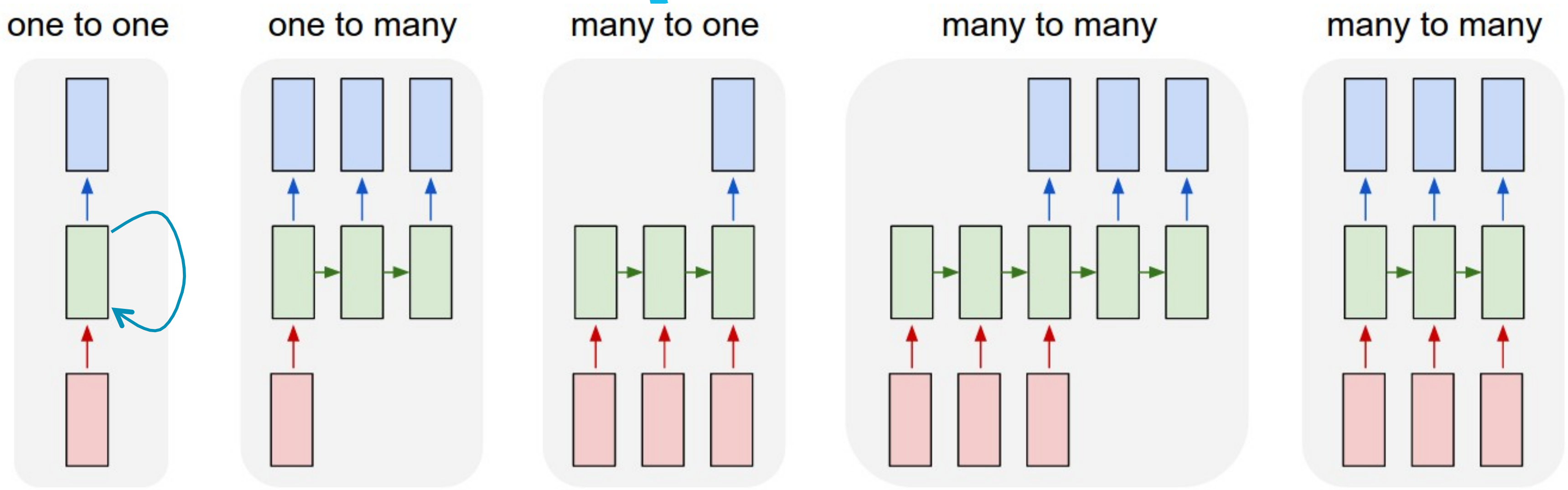
Classification

Image captioning

Sentence classification & embedding

Neural machine translation

Sequence labelling



Source: http://karpathy.github.io/assets/rnn/diags.jpeg

THREE MAIN ARCHITECTURES: FNN, RNN & CNN

Feed-forward (FFN): Function approximator (Vector to vector)

- Most existing ML/statistics fall into this category

Recurrent (RNN): Sequence to sequence

- Temporal, sequential. E.g., sentence, actions, DNA, EMR
- Program evaluation/execution. E.g., sort, traveling salesman problem

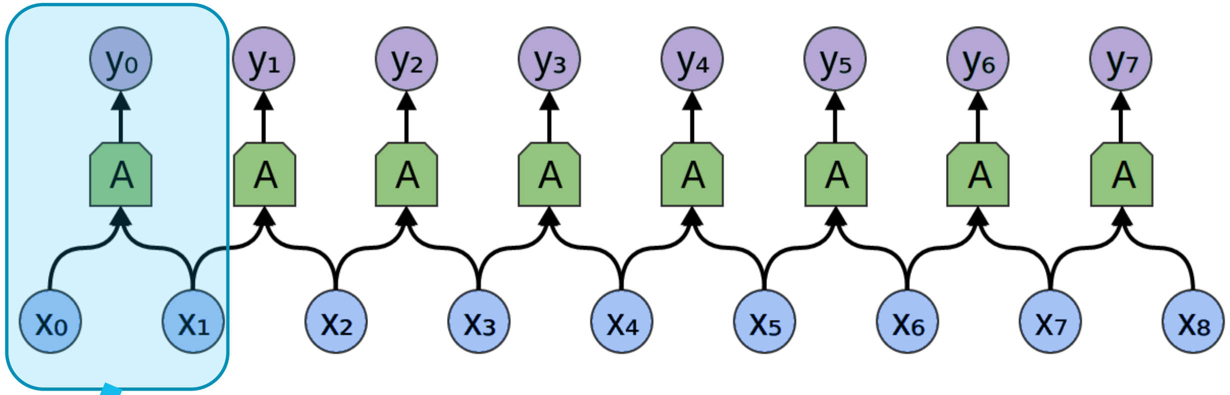
Convolutional (CNN): Image to vector/sequence/image

- In time: Speech, DNA, sentences
- In space: Image, video, relations

LEARNABLE CONVOLUTION AS FEATURE DETECTOR (TRANSLATION INVARIANCE)

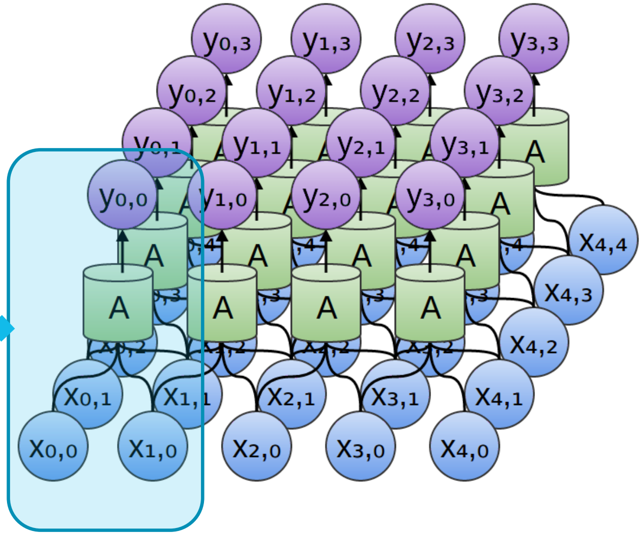
Learnable kernels

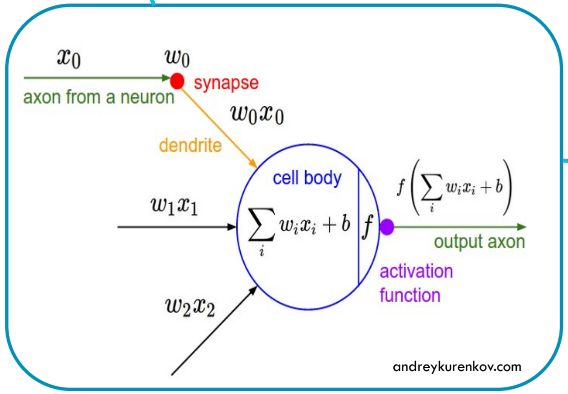
$$y_i = \sum_c \boxed{K(c)} x_{i+c}$$



http://colah.github.io/posts/2015-09-NN-Types-FP/

$$y_{ij} = \sum_{c,d} \boxed{K(c,d)} x_{i+c,j+d}$$

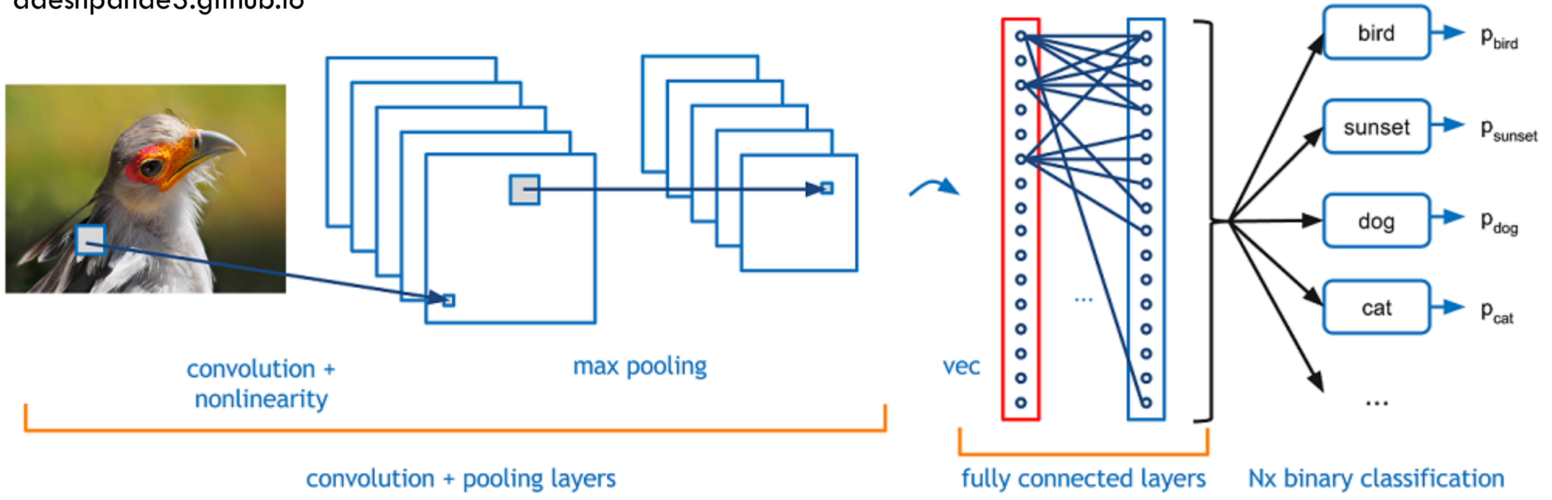




Feature detector, often many

CNN IS (CONVOLUTION → POOLING) REPEATED

adeshpande3.github.io



Design parameters:
- Padding
- Stride
- #Filters (maps)
- Filter size
- Pooling size
- #Layers
- Activation function

can be repeated N times - depth

$$F(x) = \text{NeuralNet}(\overbrace{\text{Pooling}(\text{Rectifier}(\text{Conv}(x)))}^{\text{can be repeated N times - depth}})$$

classifier max/mean nonlinearity feature detector

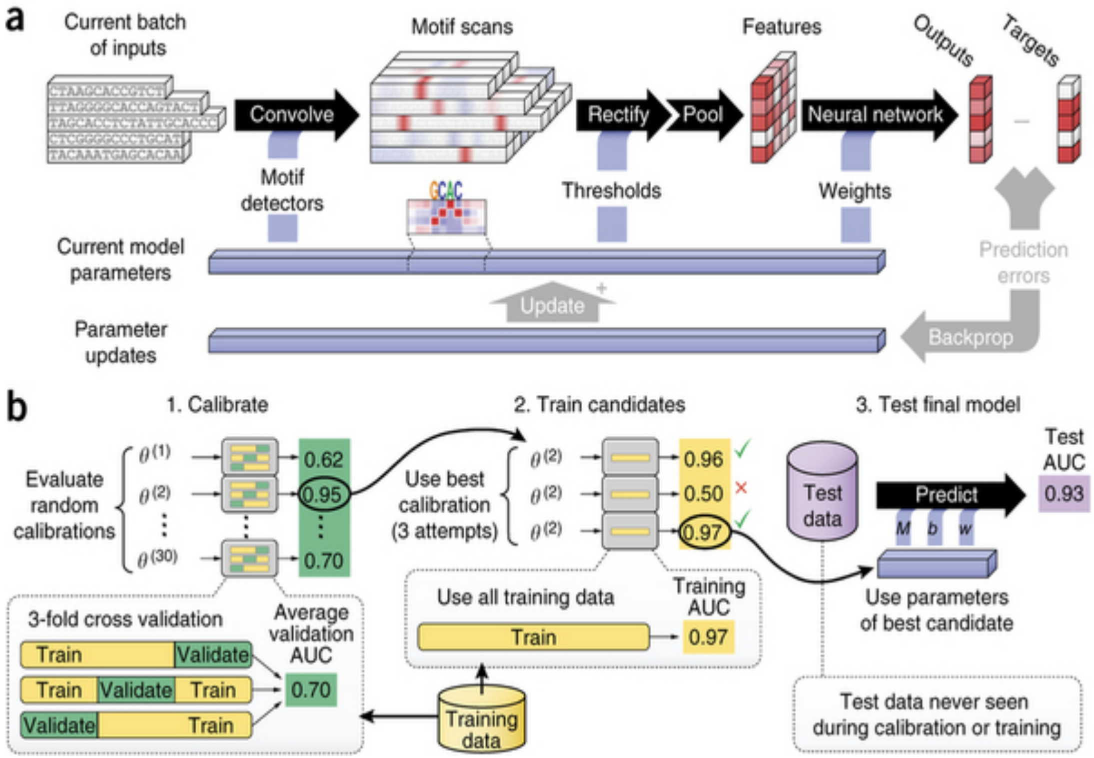
CNN: WHERE IT WORKS

Translation invariance: Image, video, repeated motifs

Examples:

- Sentence – a sequence of words (used on conjunction with word embedding)
- Sentence – a sequence of characters
- DNA sequence of {A,C,T,G}
- Relations (e.g., right-to, left-to, father-of, etc)

DeepBind, Nature 2015



http://www.nature.com/nbt/journal/v33/n8/full/nbt.3300.html

CURRENT WORK: COMBINATIONS OF {FFN,RNN,CNN}

Image classification: CNN + FFN

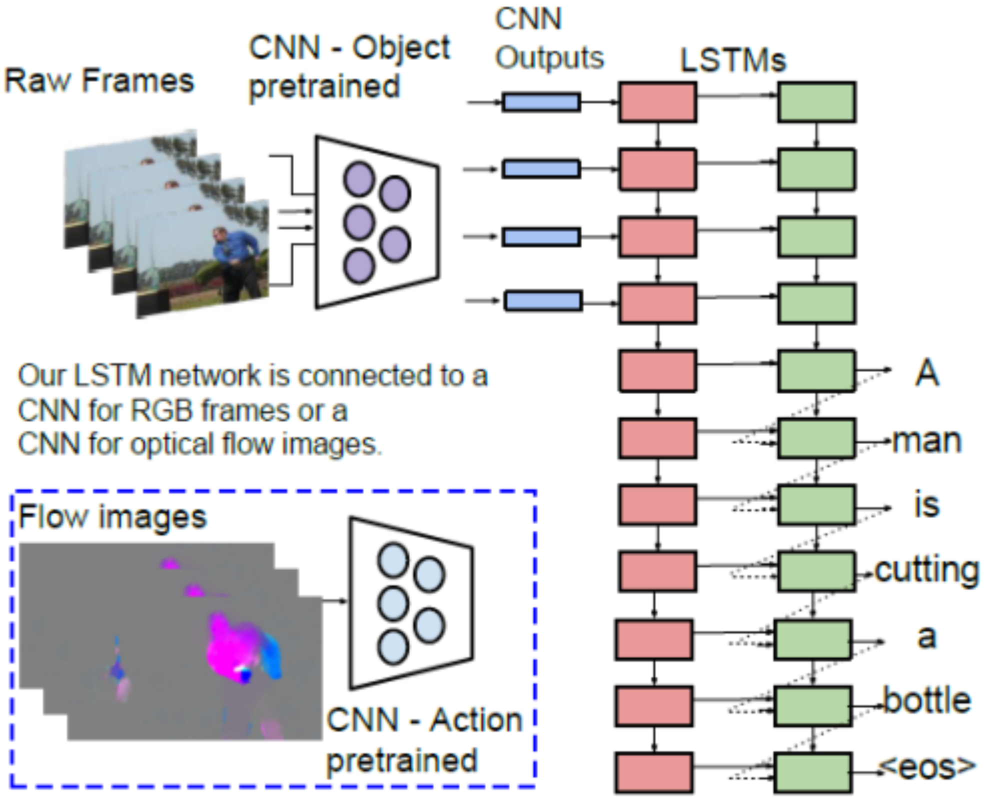
Video modelling: CNN + RNN

Image caption generation: CNN + RNN

Sentence classification: CNN + FFN

Sentence classification: RNN + FFN

Regular shapes (chain, tree, grid): CNN | RNN



https://vincentweisen.wordpress.com/2016/05/30/ammai-lecture-14-deep-learning-methods-for-image-captioning/

PRACTICAL REALISATION

More data

More GPUs

Bigger models

Better models

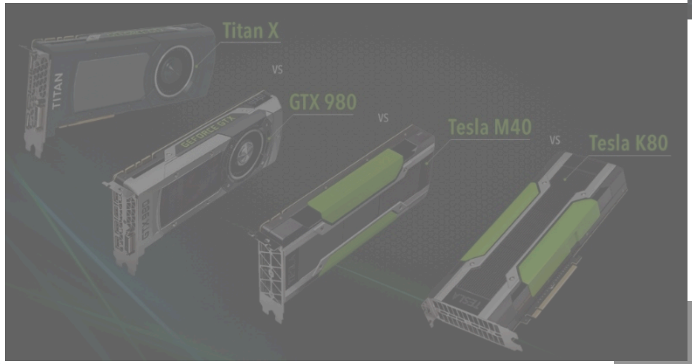
Faster iterations

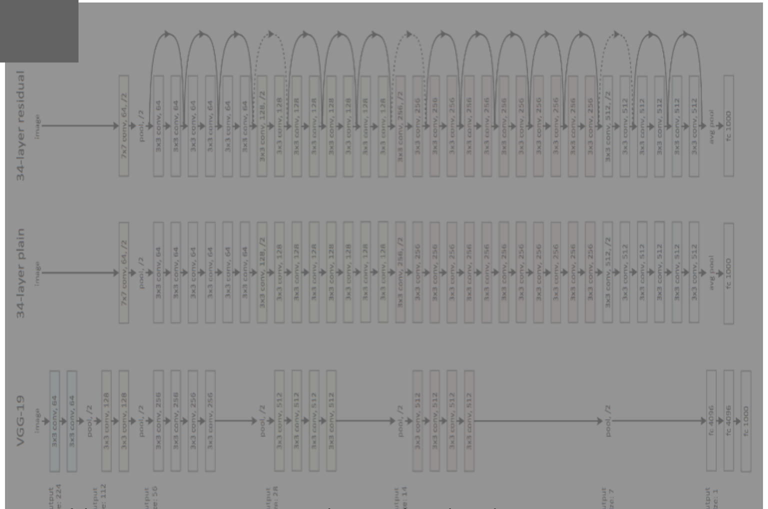
Pushing the limit of priors

Pushing the limit of patience (best models take 2-3 weeks to run)

A LOT OF NEW TRICKS

Data as new fuel





http://felixlaumon.github.io/2015/01/08/kaggle-right-whale.html

TWO ISSUES IN LEARNING

1. Slow learning and local traps
    - Very deep nets make gradients uninformative
    - Model uncertainty leads to rugged objective functions with exponentially many local minima
2. Data/model uncertainty and overfitting
    - Many models possible
    - Models are currently very big with hundreds of millions parameters
    - Deeper is more powerful, but more parameters.

SOLVING ISSUE OF SLOW LEARNING AND LOCAL TRAPS

Redesign model, e.g., using skip-connections

Sensible initialisation

Adaptive stochastic gradient descent

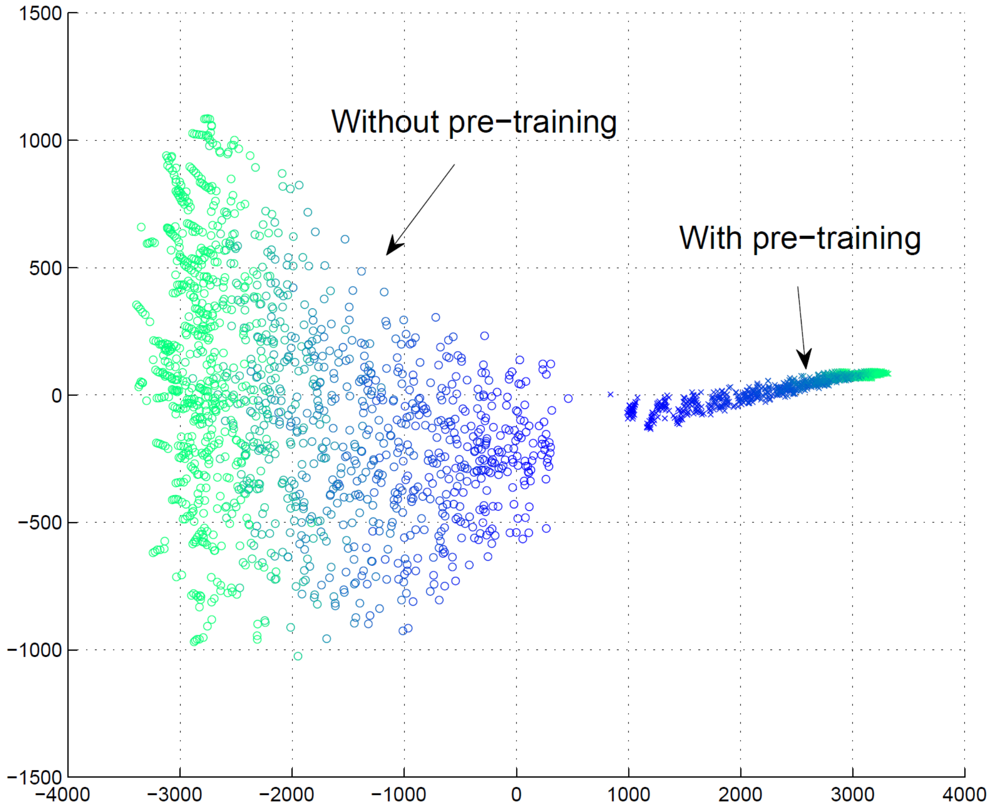
SENSIBLE INITIALISATION

Random Gaussian initialisation often works well

- TIP: Control the fan-in/fan-out norms

If not, use pre-training

- When no existing models: Unsupervised learning (e.g., word2vec, language models, autoencoders)
- Transfer from other models, e.g., popular in vision with AlexNet, Inception, ResNet, etc.



Source: (Erhan et al, JMLR'10, Fig 6)

STOCHASTIC GRADIENT DESCENT (SGD)

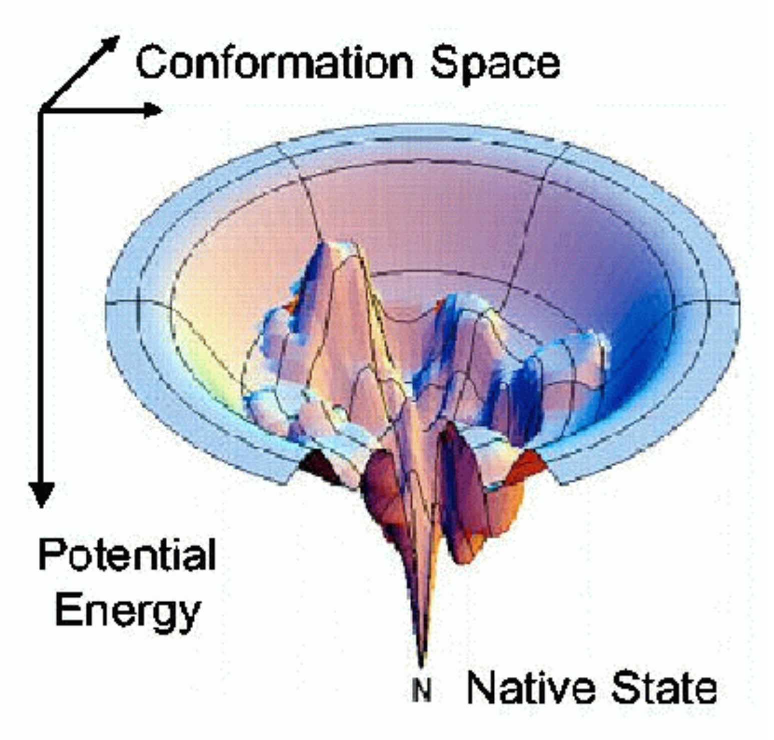
Using mini-batch to smooth out gradient

Use large enough learning rate to get over poor local minima

Periodically reduce the learning rate to land into a good local minima

It sounds like simulated annealing, but without proven global minima

Works well in practice since the energy landscape is a funnel



Source: https://charlesmartin14.wordpress.com/2015/03/25/why-does-deep-learning-work/

ADAPTIVE SGDS

Problems with SGD

- Poor gradient information, ill-conditioning, slow convergence rate
- Scheduling learning rate is an art
- Pathological curvature

Speed search: Exploiting local search direction with **momentum**

Rescaling gradient with **Adagrad**

A smoother version of Adagrad: **RMSProp** (usually good for RNNs)

All tricks combined: **Adam** (usually good for most jobs)

Adagrad (Duchi et al, 2011)

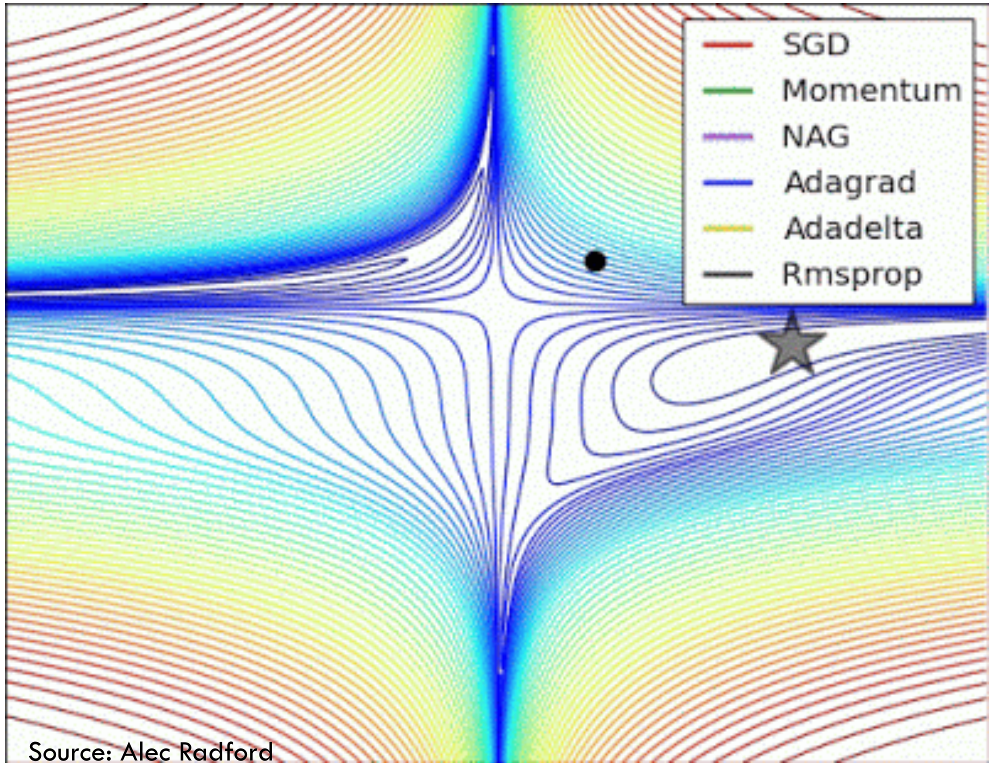
Init learning rate

Gradient

$$w_t \leftarrow w_{t-1} - \eta \frac{g_t}{\sqrt{\epsilon + \sum_{j=1}^{t-1} g_j * g_j}}$$

Smoothing factor

Previous gradients



SOLVING ISSUE OF DATA/MODEL UNCERTAINTY AND OVERFITTING

Dropouts as fast ensemble/Bayesian

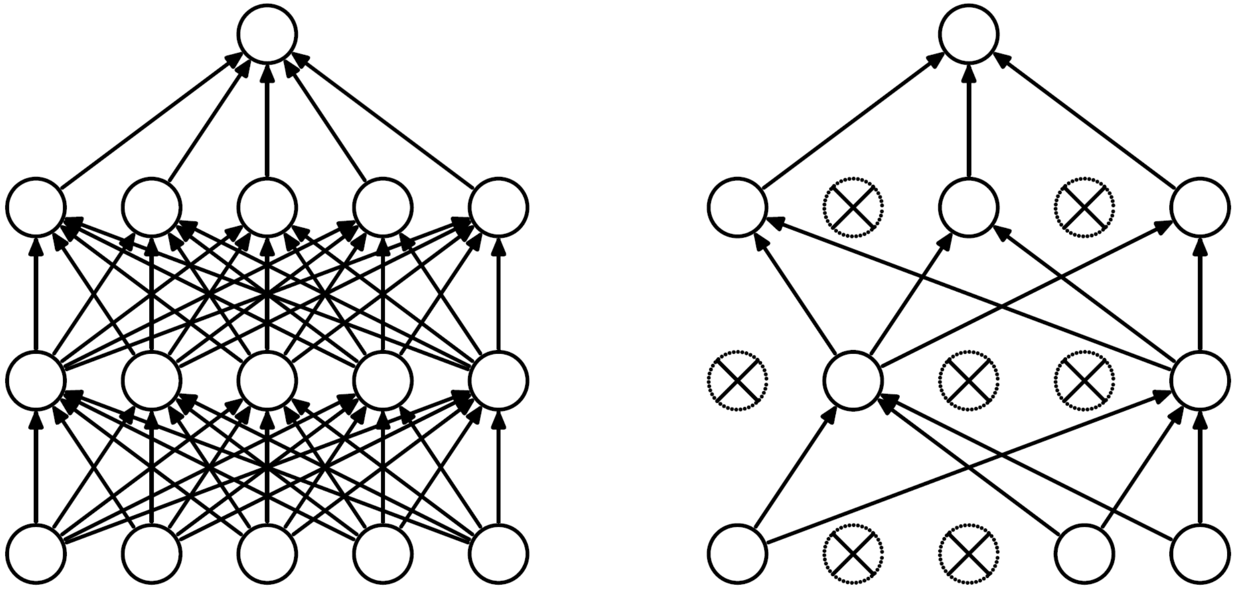
Max-norm: hidden units, channel and path

Dropout is known as the best trick in the past 10 years

DROPOUTS

A method to build ensemble of neural nets at zero cost

- For each param update, for each data point, randomly remove part of hidden units



(a) Standard Neural Net (b) After applying dropout.

Source: (Srivastava et al, JMLR 2014)

DROPOUTS AS ENSEMBLE

A method to build ensemble of neural nets at zero cost

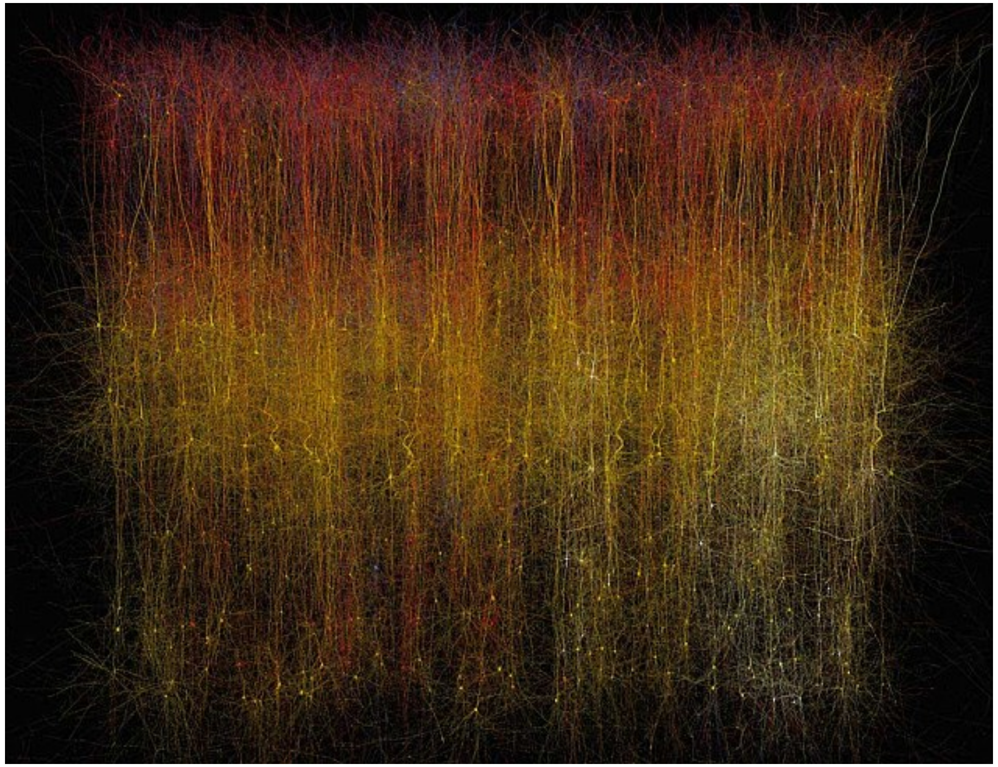
- There are $2\uparrow K$ such options for K units
- At the end, adjust the param with the same proportion

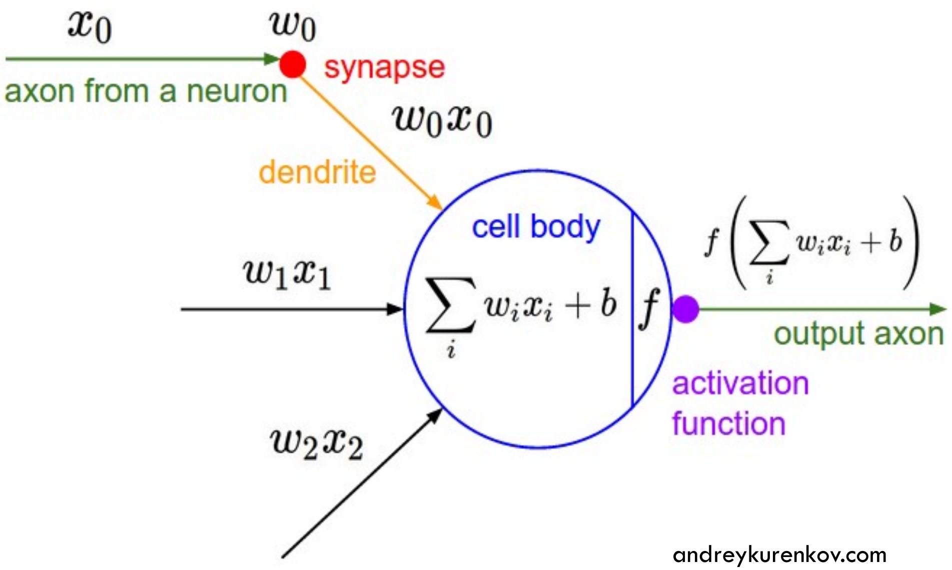
Only one model reported, but with the effect of ***n*** models, where ***n*** is number of data points.

Can be extended easily to:

- Drop features
- Drop connections
- Drop any components

PART I: WRAPPING UP





andreykurenkov.com

TWO MAJOR VIEWS OF "DEPTH" IN DEEP LEARNING

- [2006-2012] **Learning layered representations, from raw data to abstracted goal** (DBN, DBM, SDAE, GSN).
  - Typically 2-3 layers.
  - High hope for unsupervised learning. A conference set up for this: **ICLR**, starting in 2013.
  - **We will return in Part III.**
- [1991-1997] & [2012-2016] **Learning using multiple steps, from data to goal** (LSTM/GRU, NTM/DNC, N2N Mem, HWN, CLN).
  - Reach hundreds if not thousands layers.
  - Learning as credit-assignment.
  - Supervised learning won.
  - Unsupervised learning took a detour (VAE, GAN, NADE/MADE).

WHEN DOES DEEP LEARNING WORK?

Lots of data (e.g., millions)

Strong, clean training signals (e.g., when human can provide correct labels – cognitive domains).

- Andrew Ng of Baidu: When humans do well within sub-second.

Data structures are well-defined (e.g., image, speech, NLP, video)

Data is compositional (luckily, most data are like this)

The more primitive (raw) the data, the more benefit of using deep learning.

IN CASE YOU FORGOT, DEEP LEARNING MODELS ARE COMBINATIONS OF

Three models:

- FFN (layered vectors)
- RNN (recurrence for sequences)
- CNN (translation invariance + pooling)

→ Architecture engineering!

A bag of tricks:

- dropout
- piece-wise linear units (i.e., ReLU)
- adaptive stochastic gradient descent
- data augmentation
- skip-connections

END OF PART I