Deep Learning

Elisa Ricci
University of Perugia

Outline

- Introduction to Deep Learning
- Brief history of Neural Networks
- Deep Learning
    - Recent technical advances
    - Models: Autoencoders, CNN, RNN
- Open problems
- Deep Learning Hands-on

The AI Revolution

FORTUNE SUBSCRIBE

Illustration by Justin Metz

SEPTEMBER 28, 2016, 5:00 PM EDT

WHY DEEP LEARNING IS SUDDENLY CHANGING YOUR LIFE

Decades-old discoveries are now
and will soon transfo

Over the past four years, readers have doubt
a wide range of everyday technologies.

Most obviously, the speech-recognition func

A survival guide for the coming AI revolution

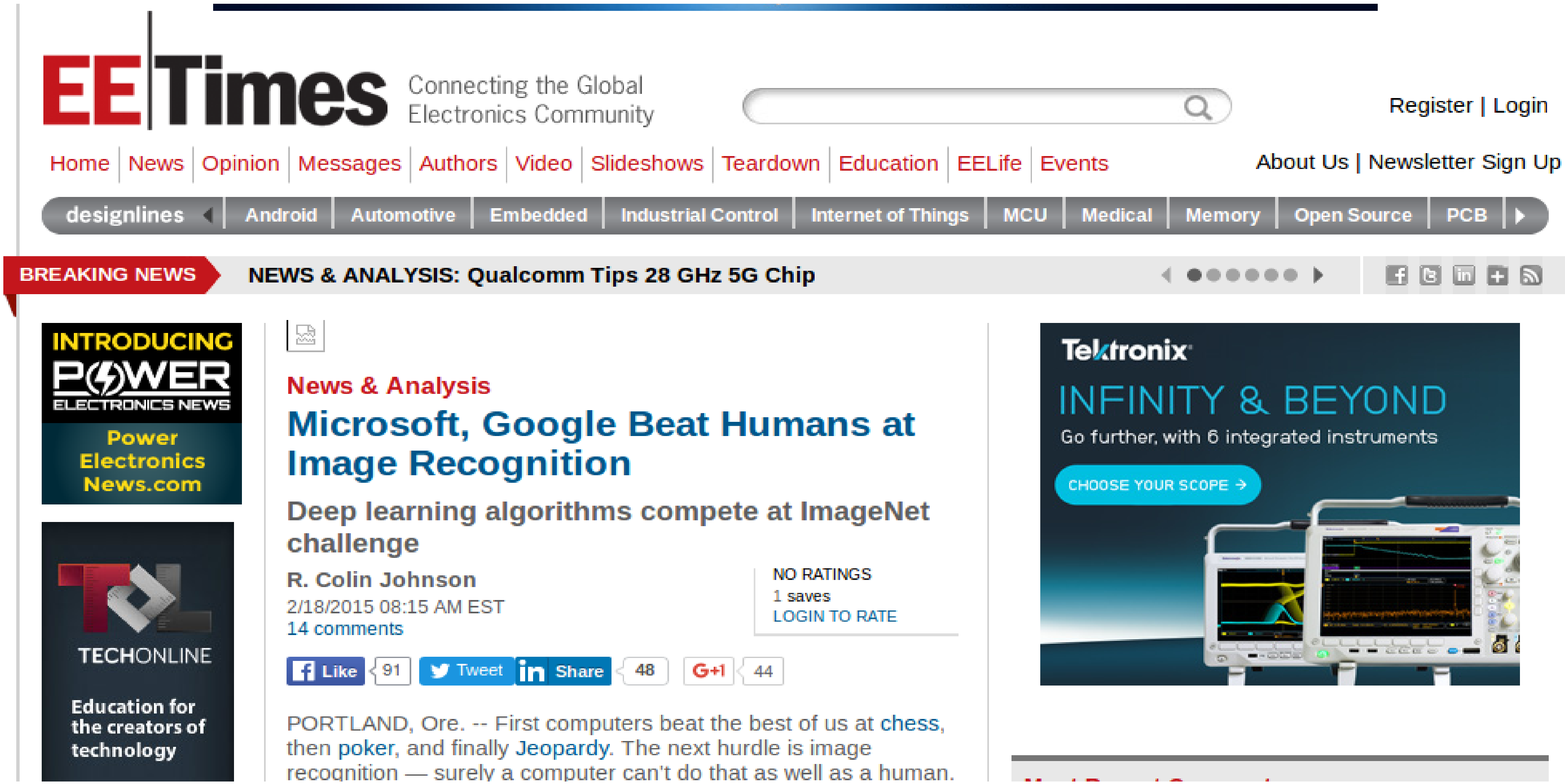
By Natalie Rens, Juxi Leitner | Mar 03, 2017

This article first appeared on The Conversation.

If the popular media isto be believed, artificial intelligence is coming to steal your job and threaten life as we know it. If we do not prepare now, we may face a future where AI runs free and dominates humans in society.

The AI revolution is indeed underway. To ensure you are prepared to make it through the

Computer Vision



EE Times Connecting the Global Electronics Community

Register | Login

Home | News | Opinion | Messages | Authors | Video | Slideshows | Teardown | Education | EELife | Events

About Us | Newsletter Sign Up

designlines | Android | Automotive | Embedded | Industrial Control | Internet of Things | MCU | Medical | Memory | Open Source | PCB

BREAKING NEWS NEWS & ANALYSIS: Qualcomm Tips 28 GHz 5G Chip

News & Analysis

Microsoft, Google Beat Humans at Image Recognition

Deep learning algorithms compete at ImageNet challenge

R. Colin Johnson
2/18/2015 08:15 AM EST
14 comments

NO RATINGS
1 saves
LOGIN TO RATE

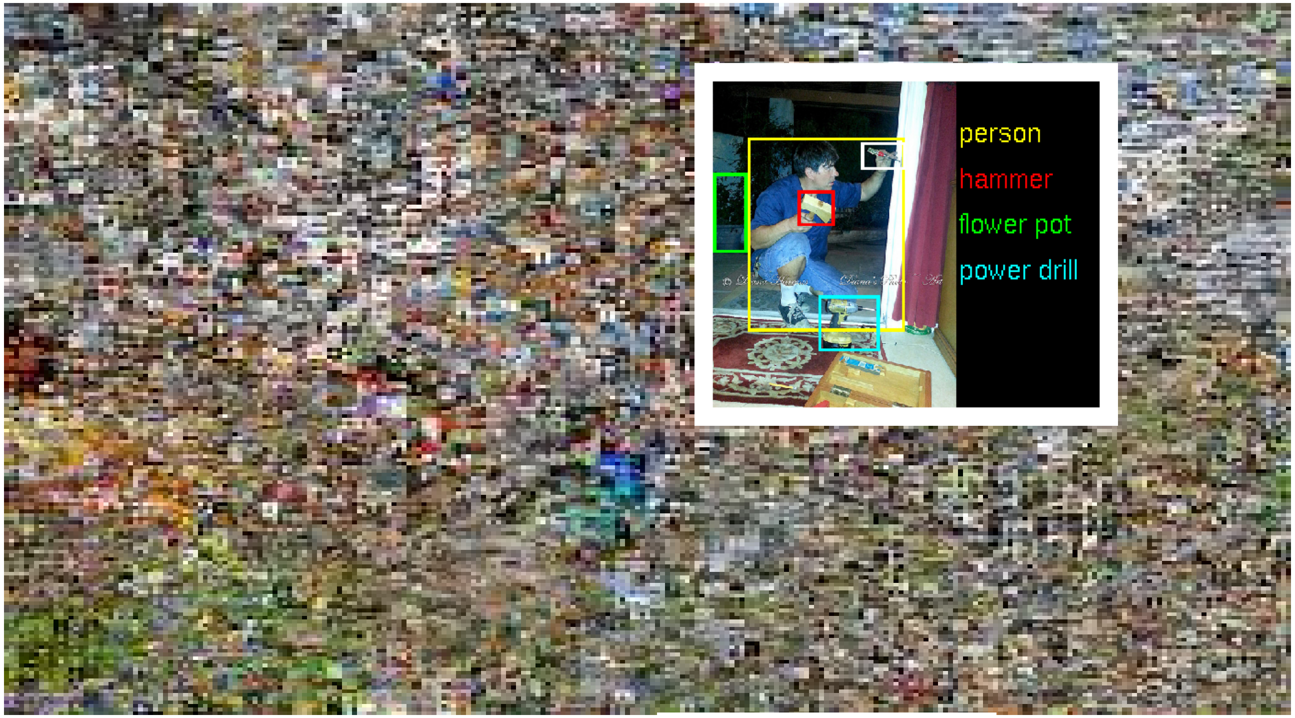
Like 91 | Tweet | Share 48 | G+1 44

PORTLAND, Ore. -- First computers beat the best of us at chess, then poker, and finally Jeopardy. The next hurdle is image recognition — surely a computer can't do that as well as a human.

Computer Vision

- Amazing progresses in the last few years with Convolutional Neural Networks (CNNs).

1.4M images, 1K categories

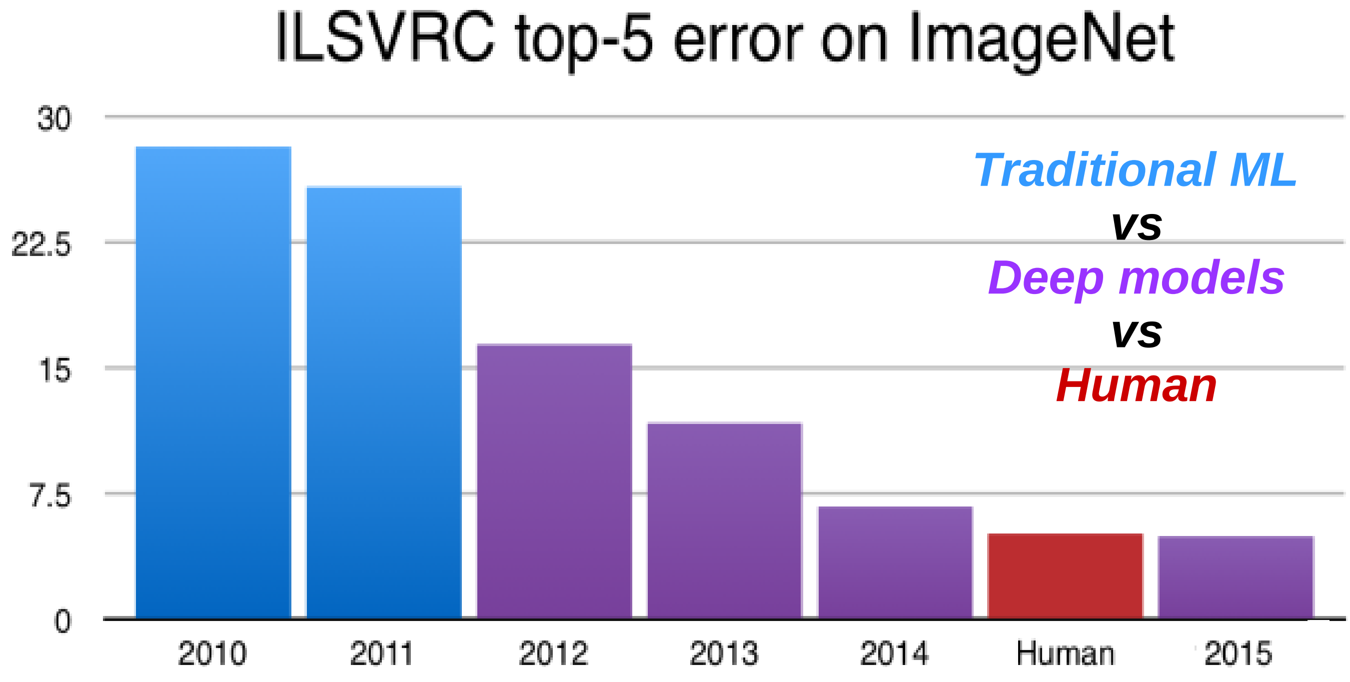


IMAGENET (2009)

Computer Vision

- Amazing progresses in the last few years with Convolutional Neural Networks (CNNs).

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)



Speech Processing

- Machine translation.

Rick Rashid in **Tianjin, China**, October, 25, 2012

A voice recognition program translated a speech given by
Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese.

Speech Processing

- Text To Speech WaveNet.

Home Research Applied News & Blog About Us Careers

WaveNet: A Generative Model for Raw Audio

This post presents **WaveNet**, a deep generative model of raw audio waveforms. We show that WaveNets are able to generate speech which mimics any human voice and which sounds more natural than the best existing Text-to-Speech systems, reducing the gap with human performance by over 50%.

We also demonstrate that the same network can be used to synthesize other audio signals such as music, and present some striking samples of automatically generated piano pieces.

Talking Machines

Allowing people to converse with machines is a long-standing dream of human-computer interaction. The ability of computers to understand natural speech has been revolutionised in the last few years by the application of deep neural networks (e.g., **Google Voice Search**). However, generating speech with computers — a process usually referred to as **speech synthesis** or text-to-speech (TTS) — is still largely based on so-called **concatenative TTS**, where a very large database of short speech fragments are recorded from a single speaker and then recombined to form complete utterances. This makes it difficult to modify the voice (for example switching to a different speaker, or altering the emphasis or emotion of their speech) without recording a whole

Creativity

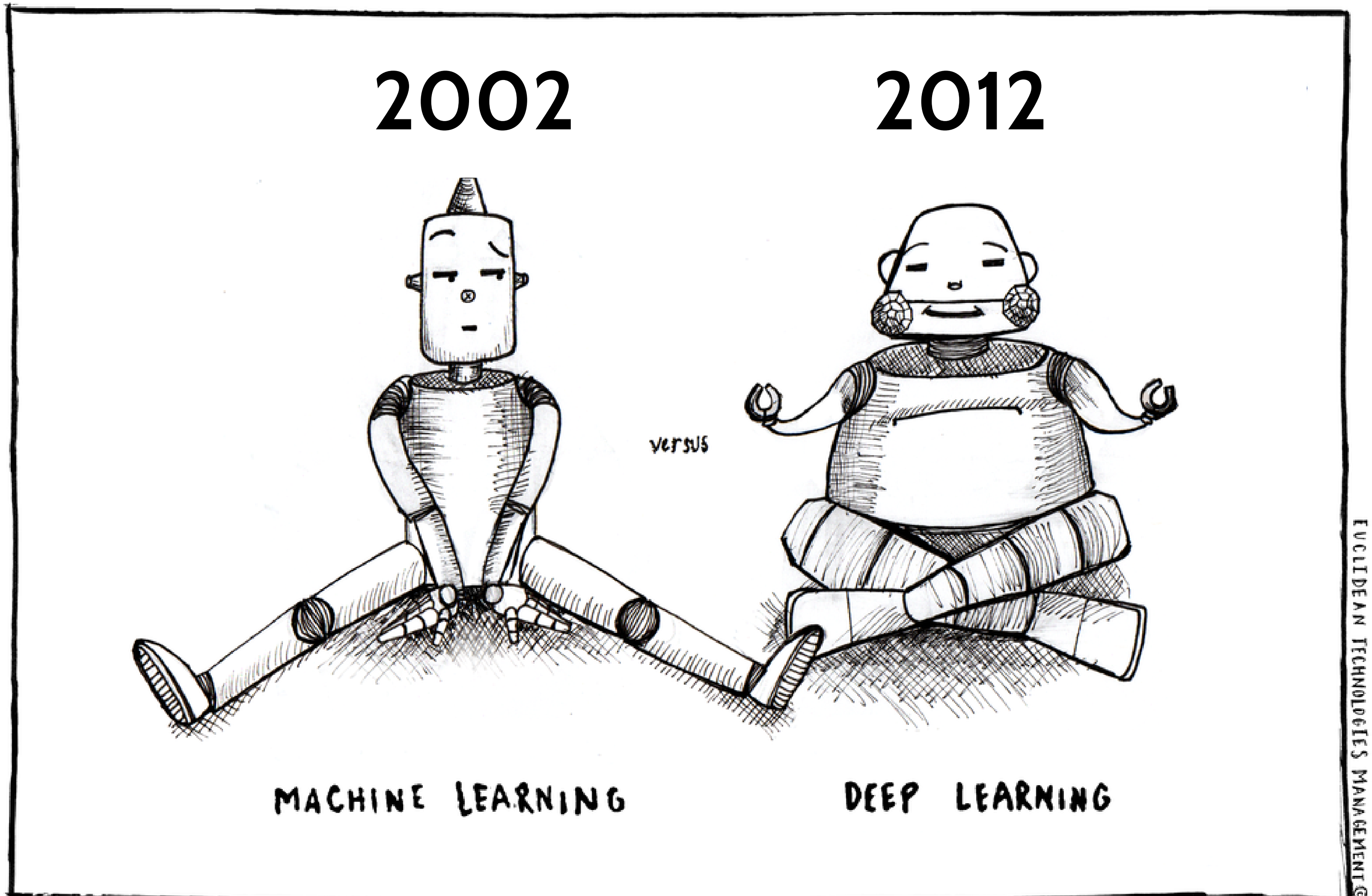
- Style Transfer.

[Ruder *et al.*]

Creativity

- Generate Donald Trump's Twitter eruptions.

Machine Learning vs Deep Learning



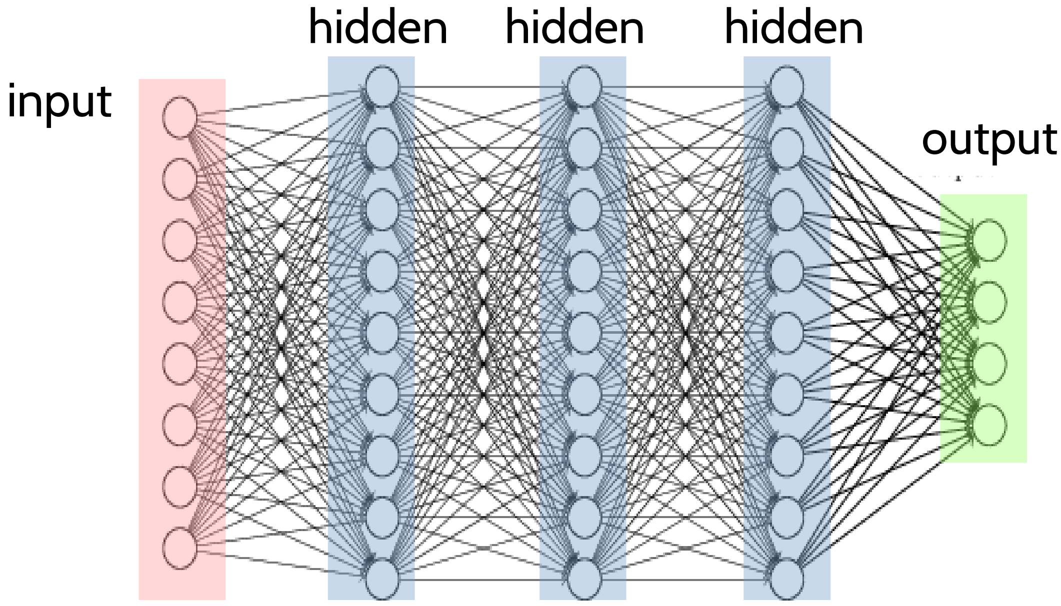
Deep Learning

- **What is deep learning**?
- **Why is it generally better** than traditional ML methods on image, speech and certain other types of data?

Deep Learning

- **What is deep learning**?

*Deep Learning means using a **neural network** with **several layers of nodes** between input and output*



More formally

- A family of **parametric** models which learn **non-linear hierarchical** representations:

$$a_L(\mathbf{x};\boldsymbol{\Theta}) = h_L(h_{L-1}(\ldots(h_1(\mathbf{x},\boldsymbol{\theta}_1),\boldsymbol{\theta}_{L-1}),\boldsymbol{\theta}_L)$$

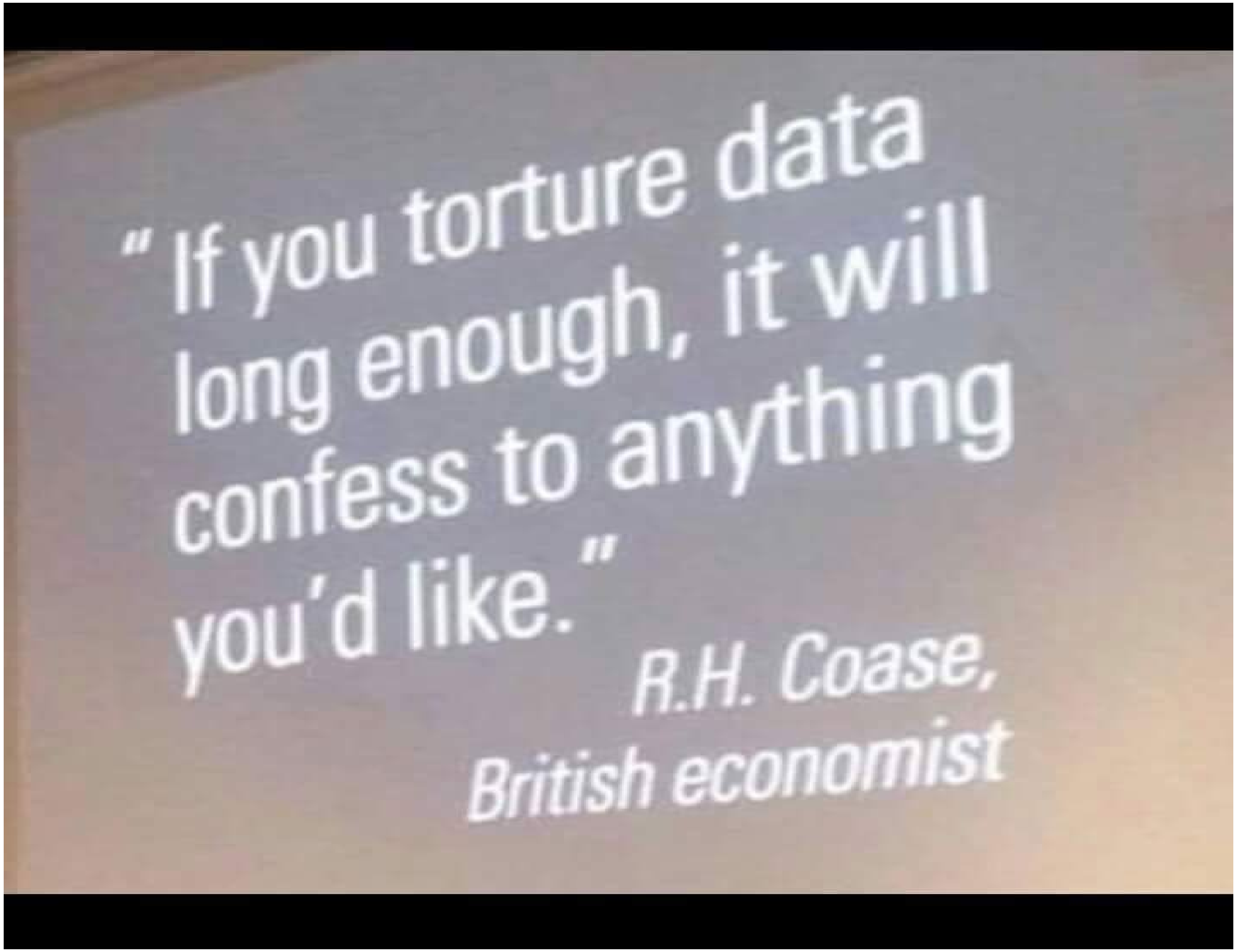
input

parameters
of the network

non-linear
activation

parameters
of layer *L*

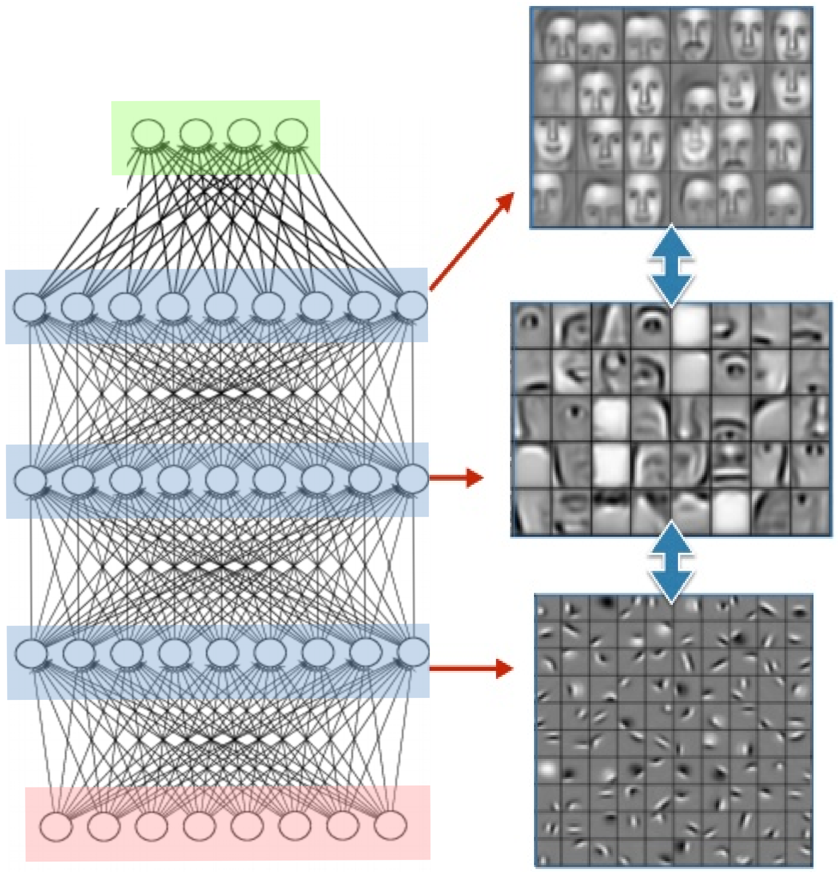
… and informally



Deep Learning

- **Why is it generally better** than other ML methods on image, speech and certain other types of data?

The series of layers between input and output compute ***relevant features automatically*** *in a series of stages, just as our brains seem to.*



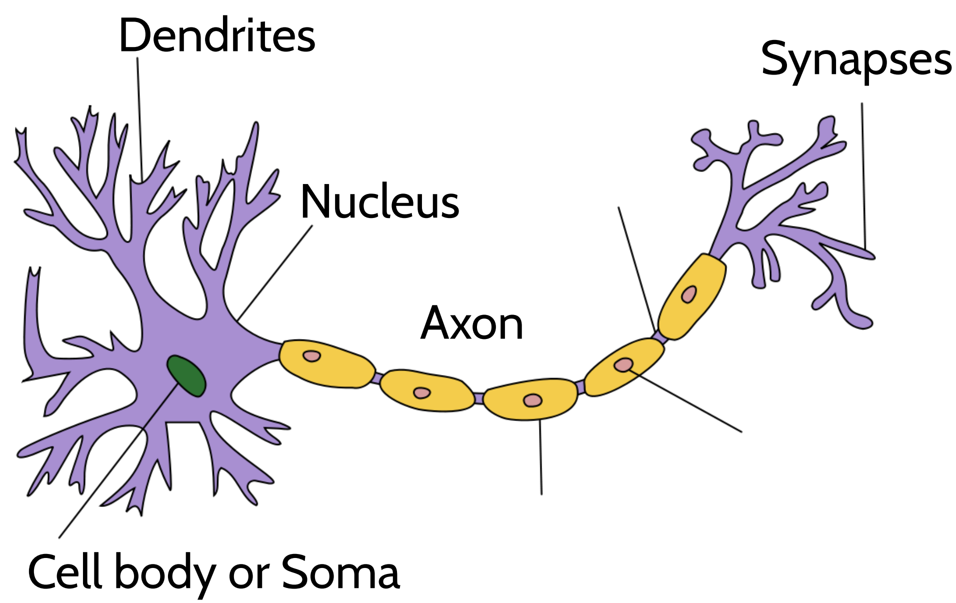
Deep Learning

...but neural networks have been around for 25 years...

So, what is new?

Biological neuron

- A neuron has
  - Branching input (**dendrites**)
  - Branching output (the **axon**)

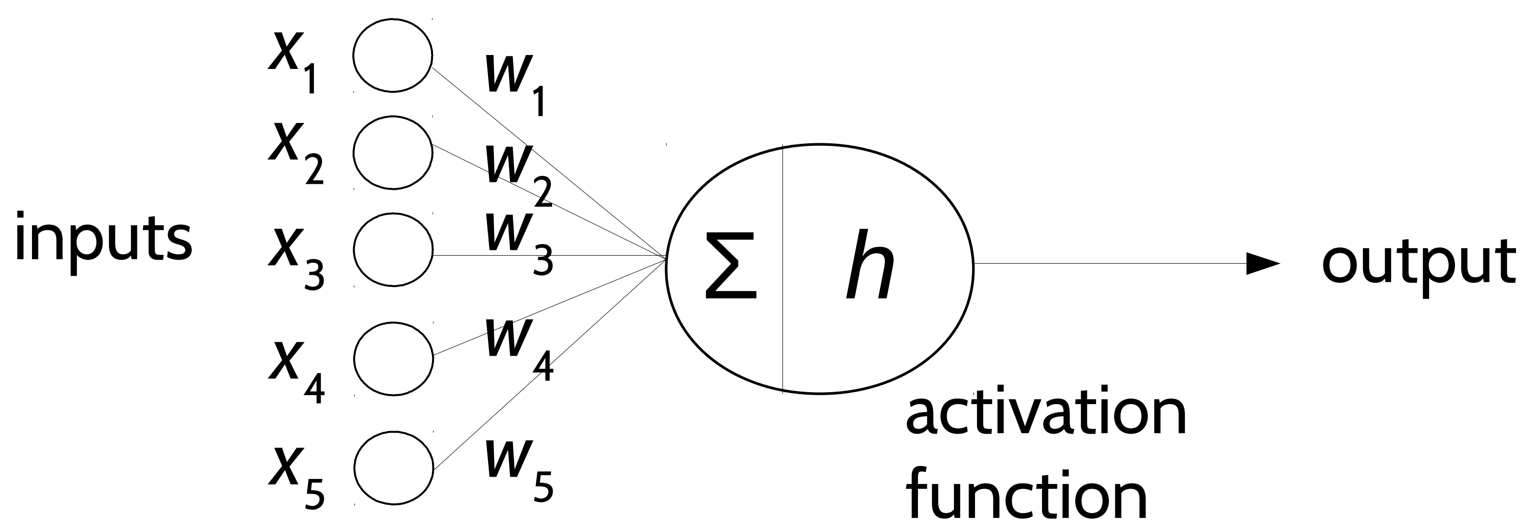


- Information moves from the dendrites to the axon via the **cell body**

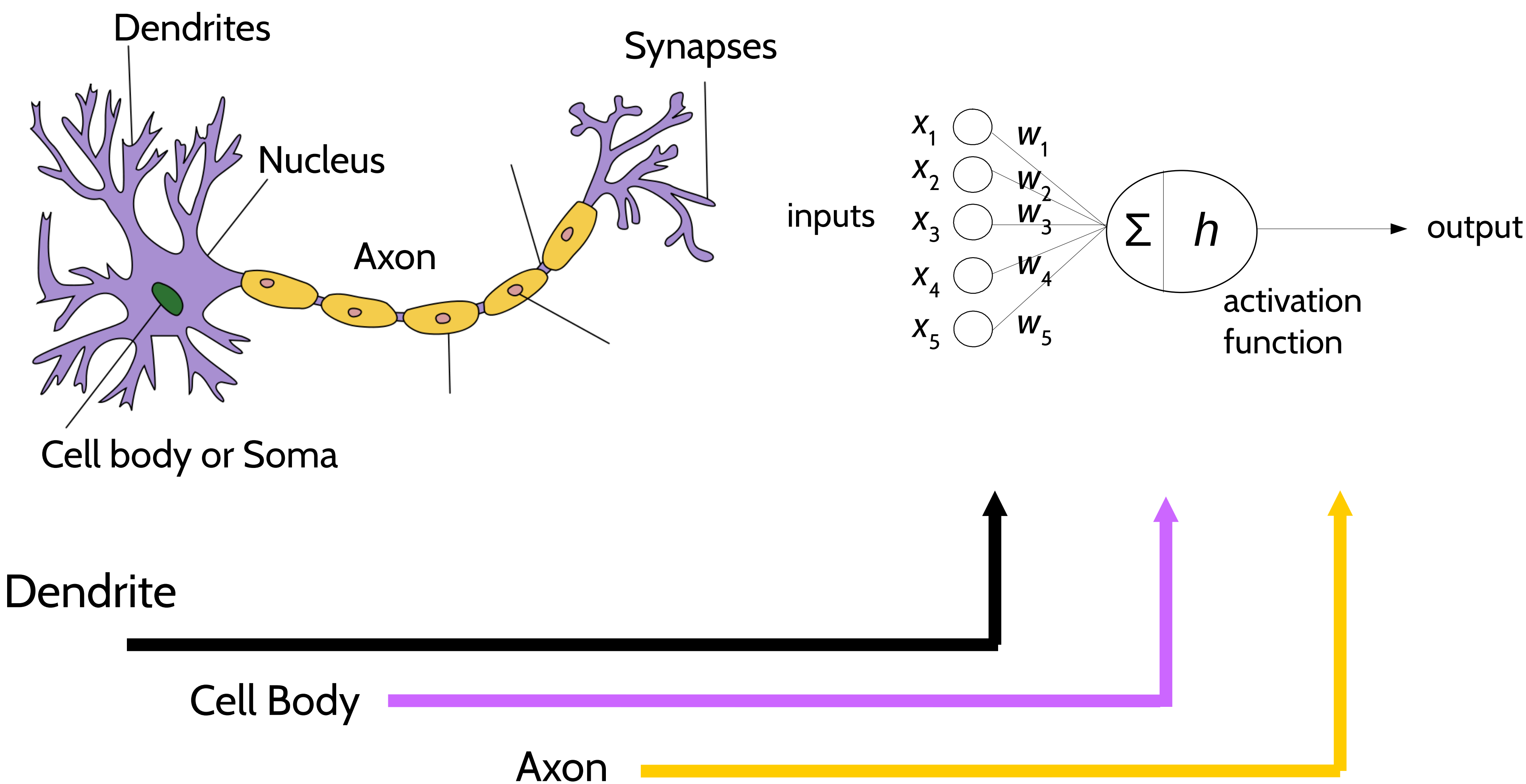
- Axon connects to dendrites via **synapses**
  - Synapses vary in strength
  - Synapses may be excitatory or inhibitory

Perceptron

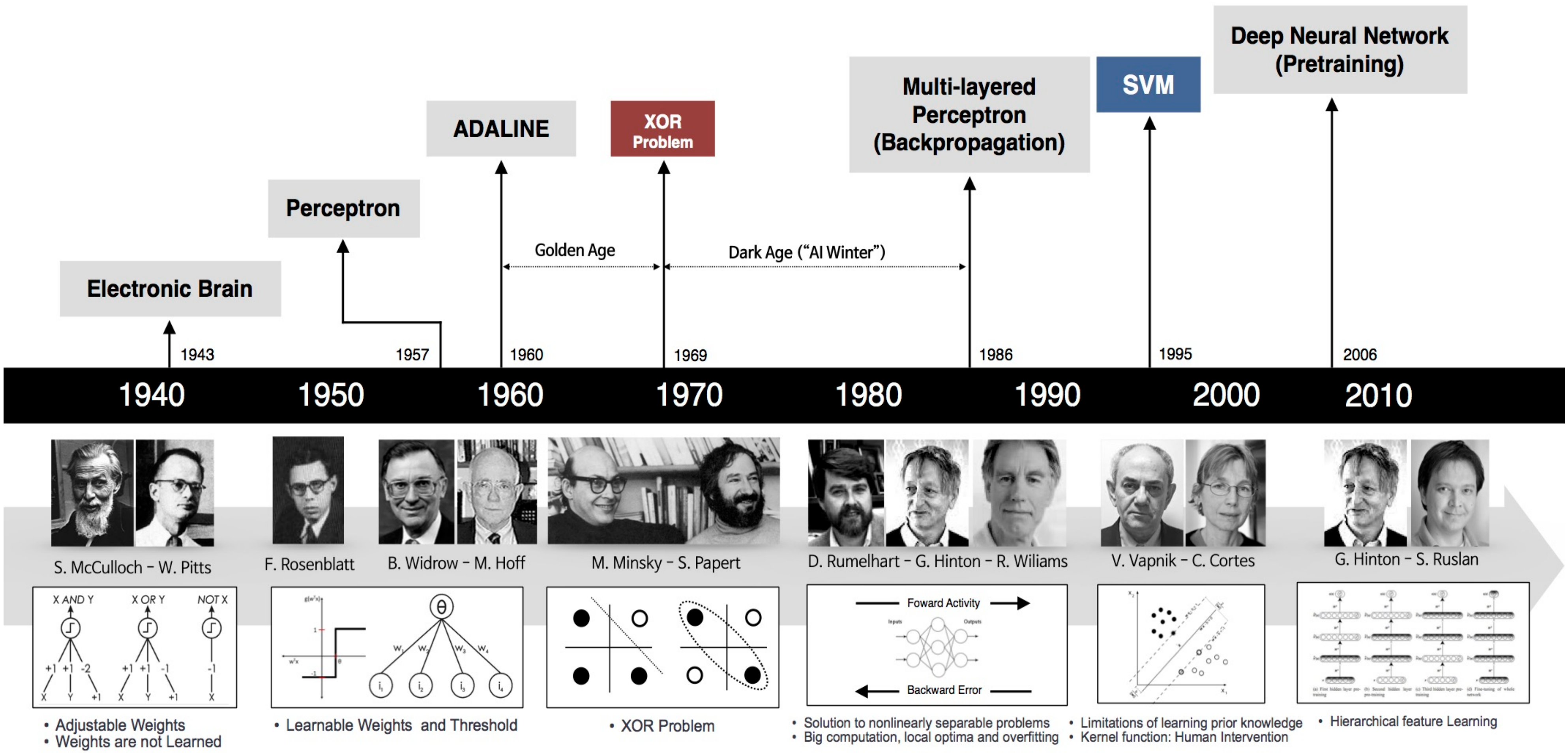
An Artificial Neuron (Perceptron) is a non-linear parameterized function with restricted output range



Biological neuron and Perceptron



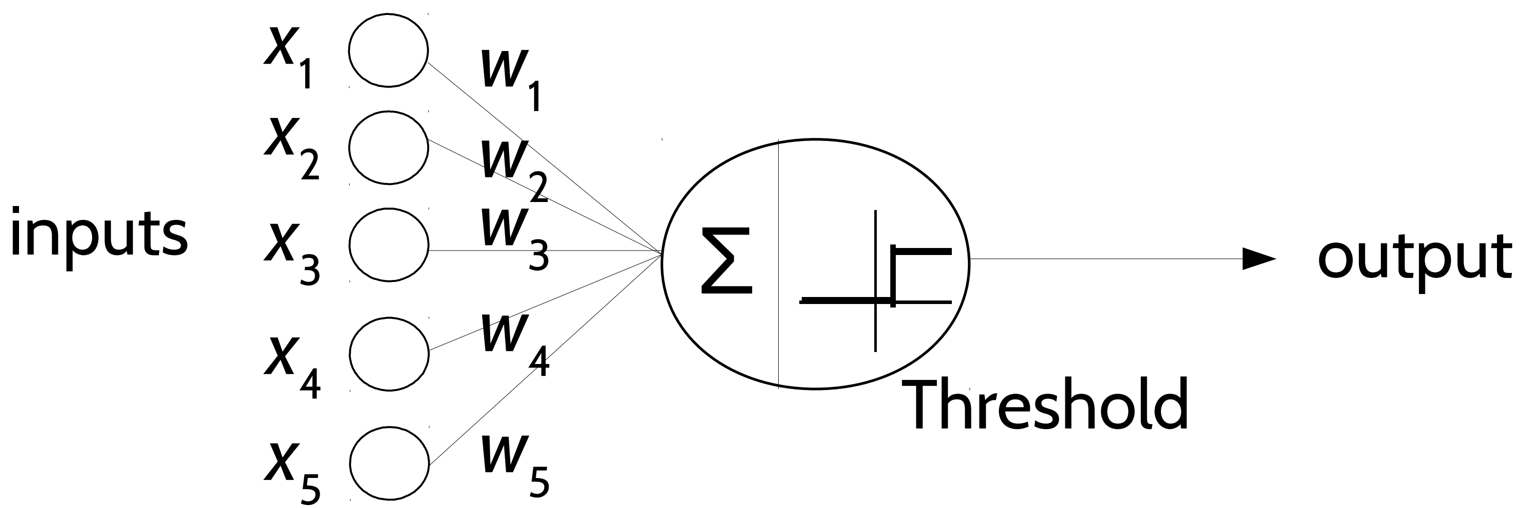
Brief History of Neural Networks



[VUNO]

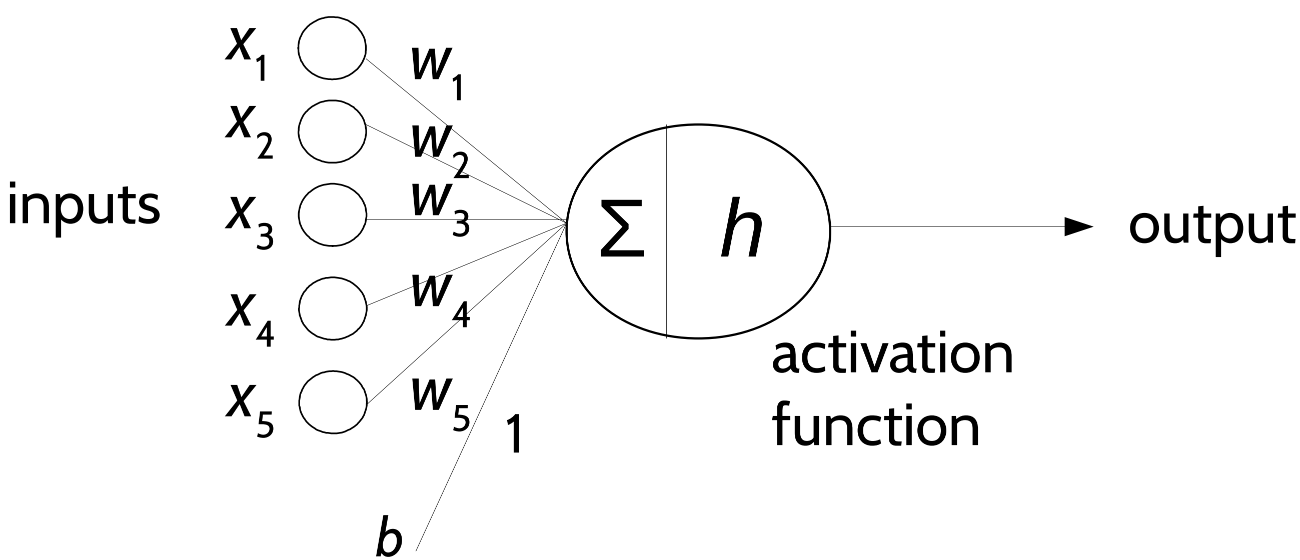
1943 – McCulloch & Pitts Model

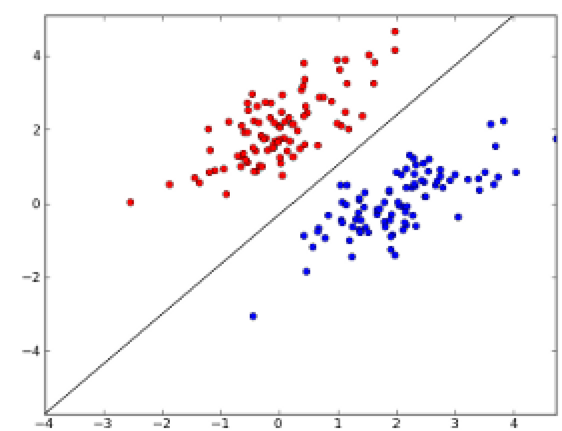
- Early model of artificial neuron
- Generates a binary output
- The weights values are fixed



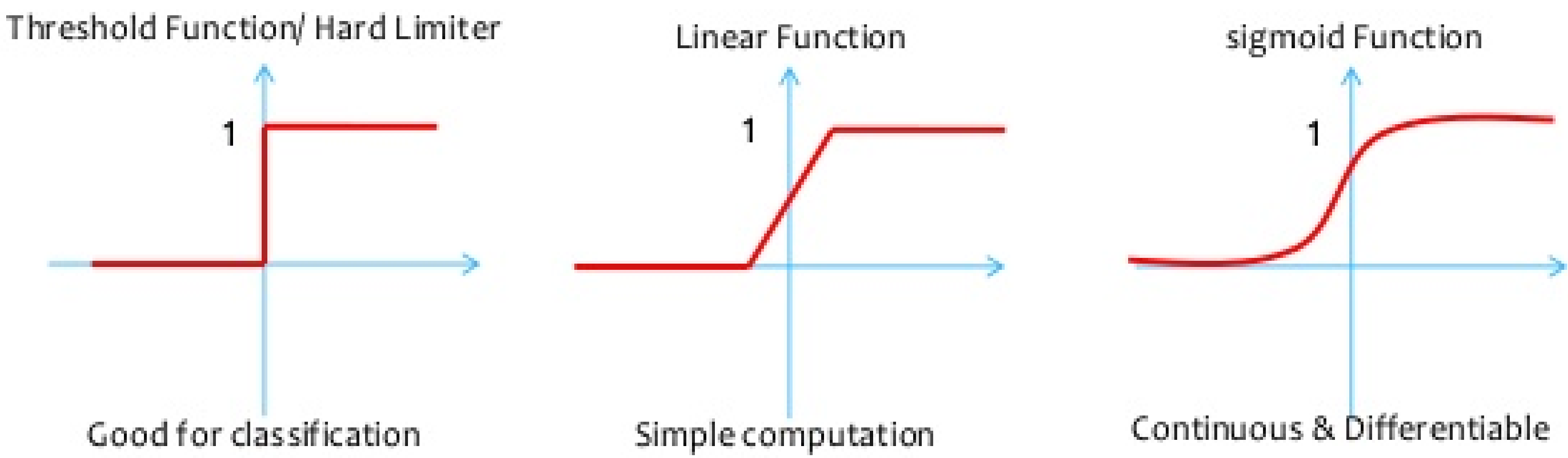
1958 – Perceptron by Rosemblatt

- Perceptron as a machine for linear classification
- Main idea: Learn the weigths and consider bias.
  - One weight per input
  - Multiply weights with respective inputs and add bias
  - If result larger than threshold return 1, otherwise 0





Activation functions



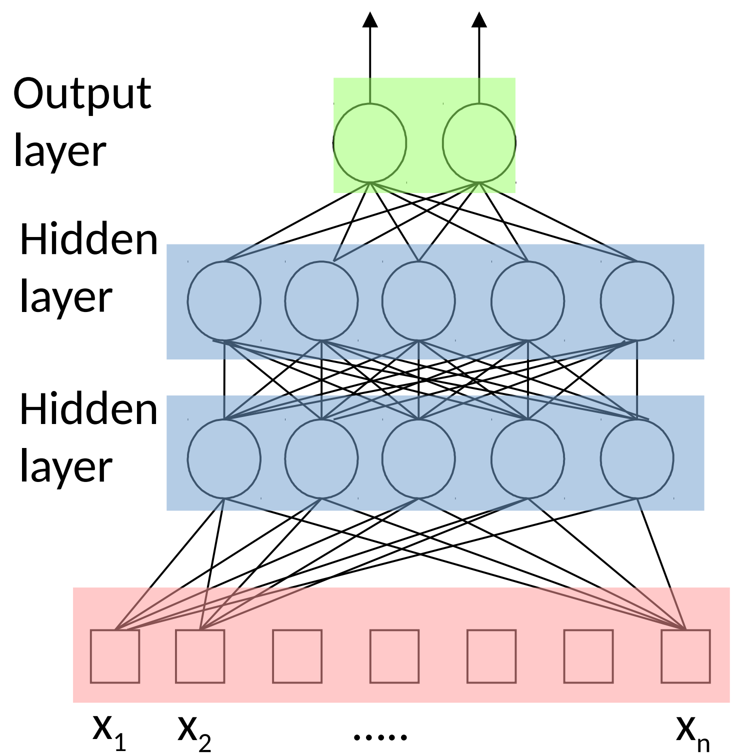
$$a = \sigma(x) = \frac{1}{1 + e^{-x}}$$

First AI winter

- The exclusive or (XOR) cannot be solved by perceptrons
- Neural models cannot be applied to complex tasks

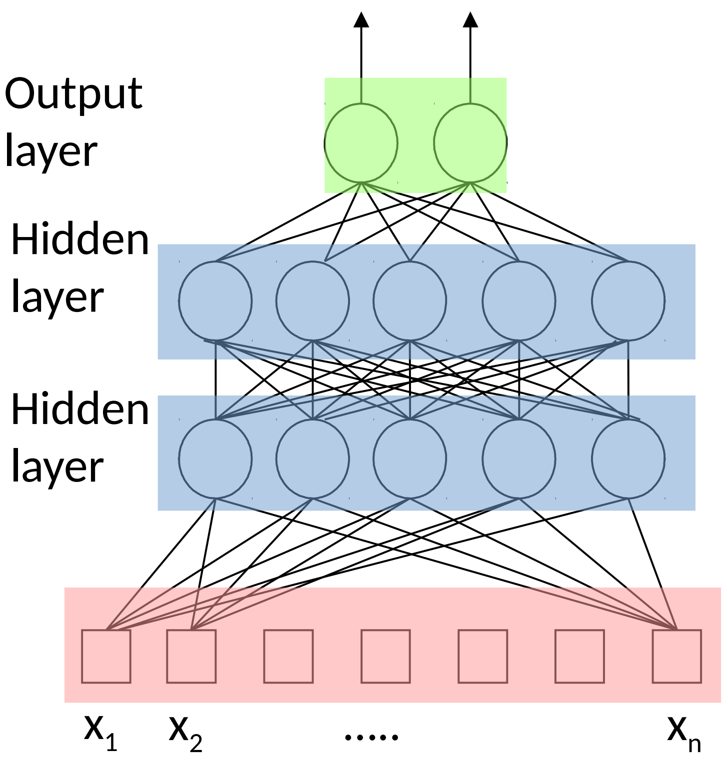
First AI winter

- But, can XOR be solved by neural networks?
  - Multi-layer perceptrons (MLP) can solve XOR
  - Few years later Minsky built such MLP



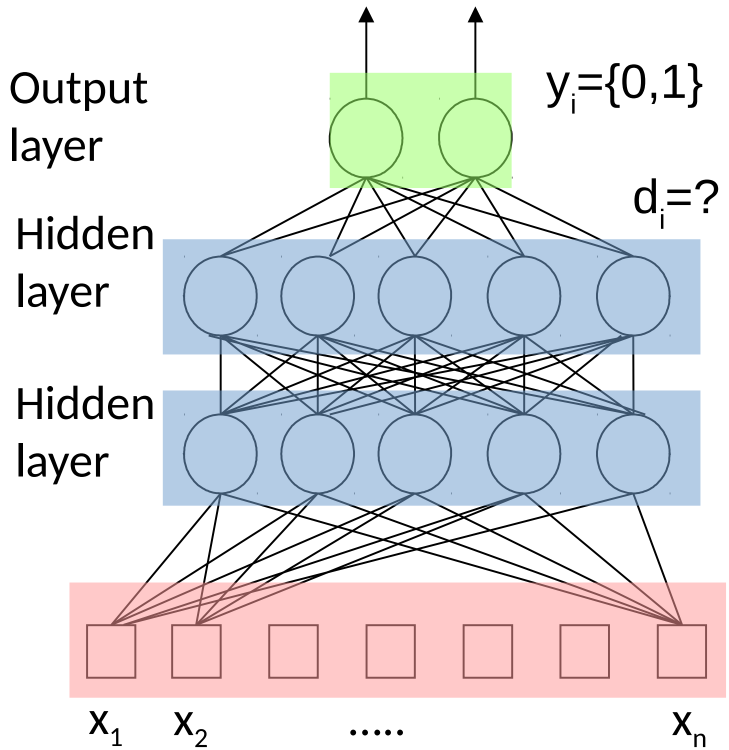
Multi-layer feed forward Neural Network

- Main idea:
  - Densely connect artificial neurons to realize compositions of non-linear functions
- The information is propagated from the inputs to the outputs
  - Directed Acyclic Graph (DAG)
- Tasks: Classification, Regression
- The input data are n dimensional, usually the feature vectors



First AI winter

- How to train a MLP?
  - Rosenblatt's algorithm not applicable, as it expects to know the desired target.
  - For hidden layers we cannot know the desired target



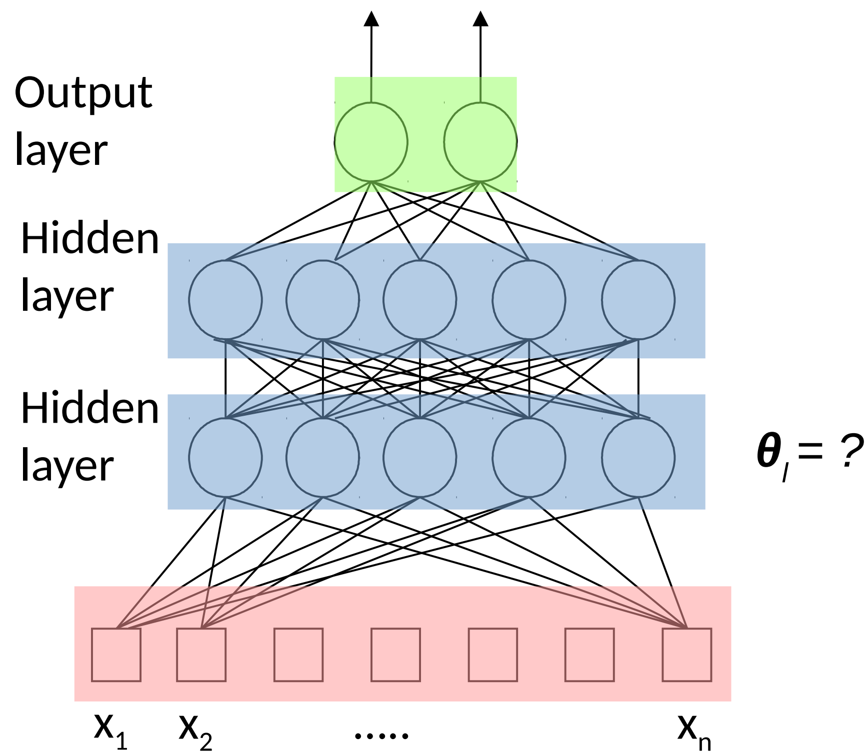
1986 – Backpropagation

- Backpropagation revitalize the field
- Learning MLP for complicated functions can be solved
- Efficient algorithm which processes “large” training sets
- Allowed for complicated neural network architectures
- Today backpropagation is still at the core of neural network training

Werbos (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University.
Rumelhart, Hintont, Williams (1986). Learning representations by back-propagating errors. Nature

Backpropagation

Learning is the process of modifying the weights of each layer $\boldsymbol{\theta}_l$ in order to produce a network that performs some function:



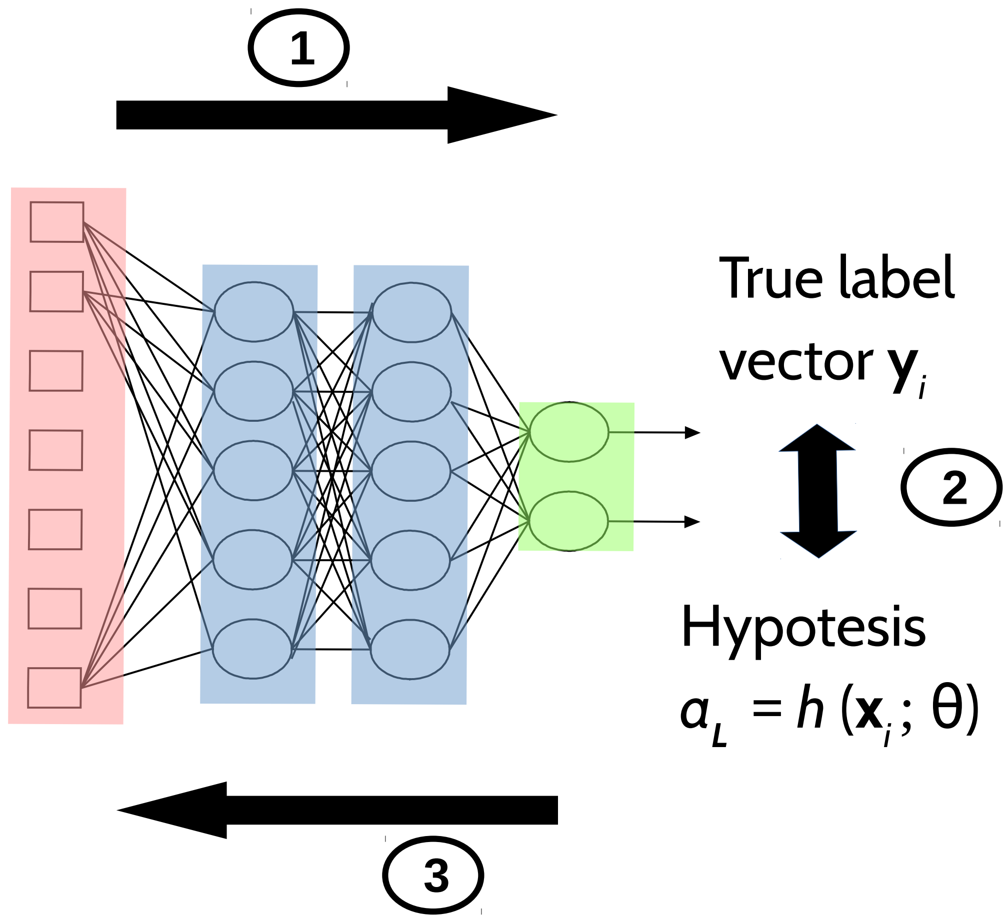
Backpropagation

- Preliminary steps:
  - Collect/acquire a training set {*X*, *Y*}
  - Define model and initialize randomly weights.

- Given the training set find the weights:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \sum_{\mathbf{x}_i, y_i} \ell\left(y_i, a_L(\mathbf{x}_i; \boldsymbol{\Theta})\right)$$

Backpropagation



1) ***Forward propagation:*** sum inputs, produce activations, feed-forward
2) ***Error estimation***.
3) ***Back propagate*** the error signal and used it to update weights

Backpropagation

Randomly initialize the initial weights

While error is too large

(1) For <u>each training sample</u> (presented in random order)

Apply the inputs to the network

Calculate the output for every neuron from the input layer, through the hidden layers, to the output layer

(2) Calculate the error at the outputs

(3) Use the output error to compute error signals for previous layers

Use the error signals to compute weight adjustments

Apply the weight adjustments

Periodically evaluate the network performance

Backpropagation

- Optimization with gradient descent:

$$\boldsymbol{\Theta}^{t+1} = \boldsymbol{\Theta}^{t} - \eta_t \nabla_{\Theta} \mathcal{L}$$

- The most important component is how to compute the gradient
- The backward computations of network return the gradient

Backpropagation

Forward

$$a_l = h_l(x_l) \quad \text{and} \quad x_{l+1} = a_l$$

Backward

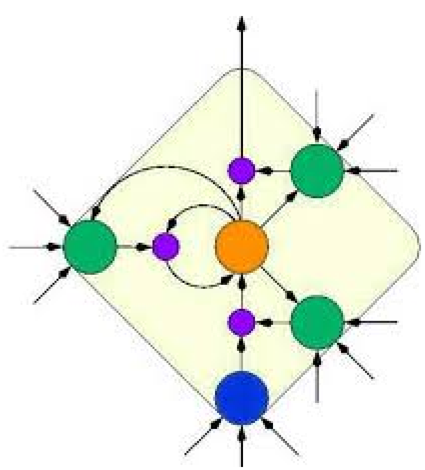
$$\frac{\partial \mathcal{L}}{\partial a_l} = \left(\frac{\partial a_{l+1}}{\partial x_{l+1}}\right)^T \cdot \frac{\partial \mathcal{L}}{\partial a_{l+1}} \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial \theta_l} = \frac{\partial a_l}{\partial \theta_l} \cdot \left(\frac{\partial \mathcal{L}}{\partial a_l}\right)^T$$

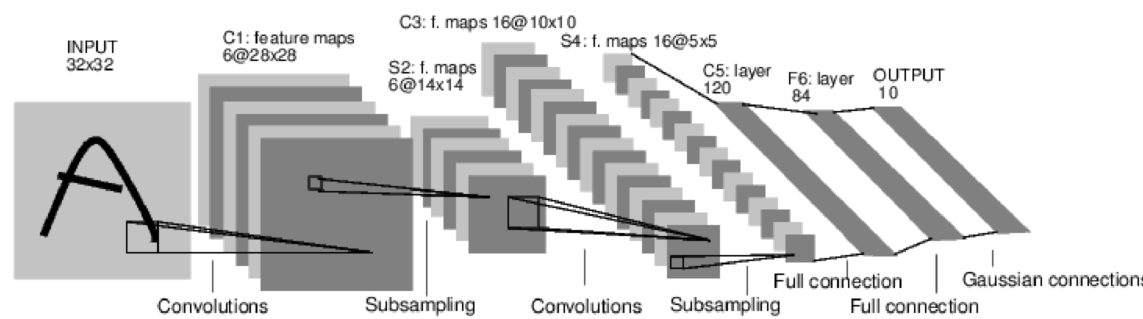
Recursive rule: Previous layer

Current layer

1990s - CNN and LSTM

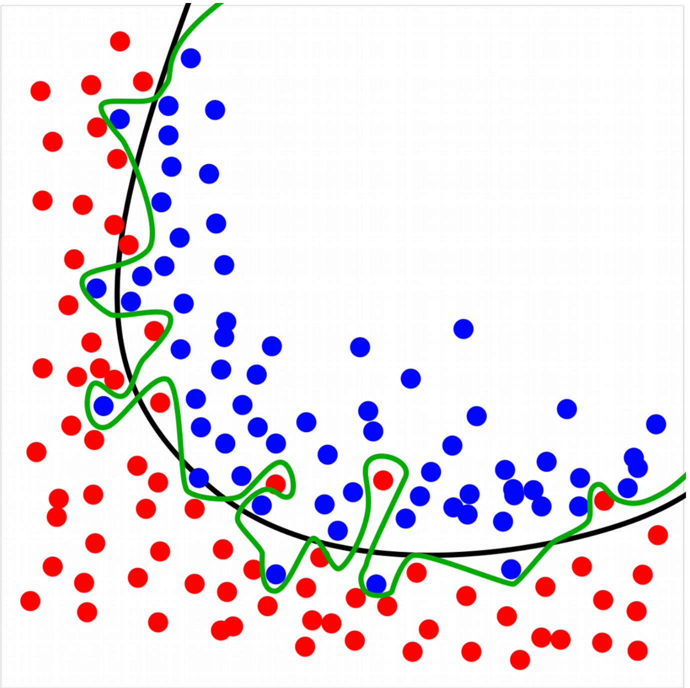
- Important advances in the field:
  - Backpropagation
  - Recurrent Long-Short Term Memory Networks (Schmidhuber, 1997)
  - Convolutional Neural Networks: OCR solved before 2000s (LeNet, 1998).





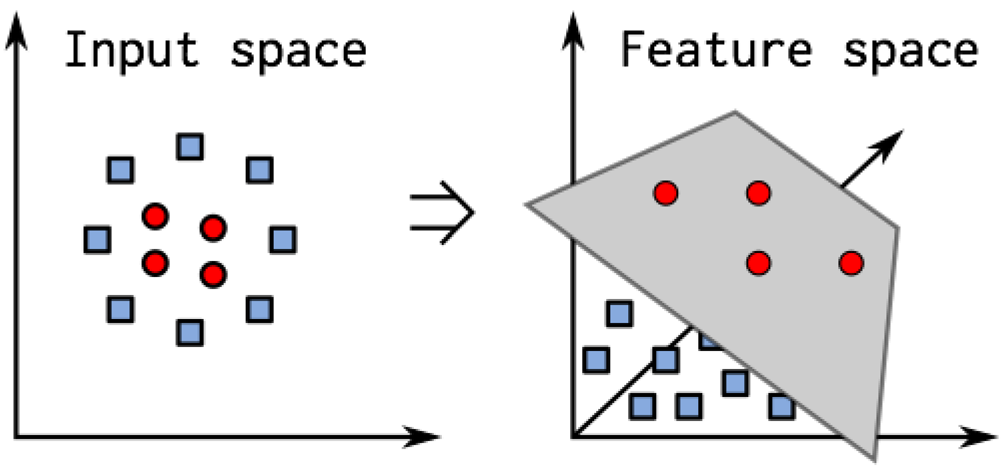
Second AI winter

- NN cannot exploit many layers
  - Overfitting
  - Vanishing gradient (with NN training you need to multiply several small numbers → they become smaller and smaller)
- Lack of processing power (no GPUs)
- Lack of data (no large annotated datasets) ◦



Second AI winter

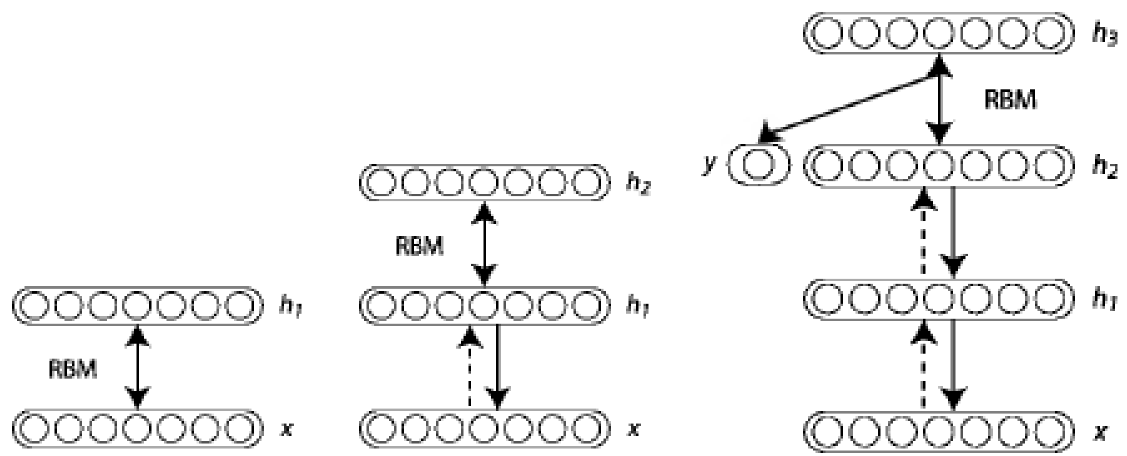
- Kernel Machines (e.g. SVMs) suddenly become very popular
  - Similar accuracies than NN in the same tasks
  - Much fewer heuristics and parameters
  - Nice proofs on generalization



The believers

2006 - Learning deep belief nets

- Clever way of initializing network weights:
  - Train each layer one by one with unsupervised training (using contrastive divergence)
  - Much better than random values
- Fine-tune weigths with a round of supervised learning just as is normal for neural nets
- State of the art performance on MNIST dataset

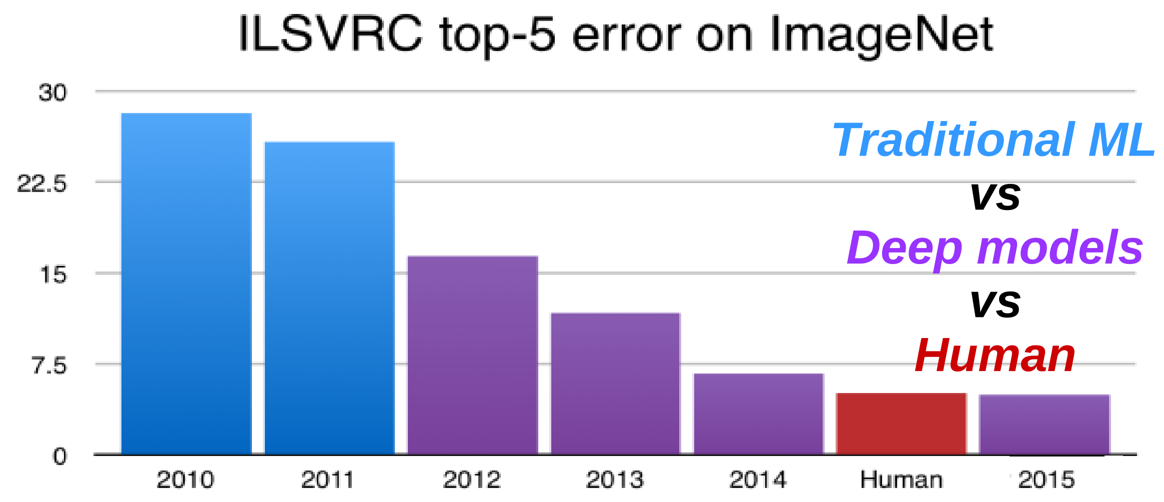


(a) Train RBM for $\mathbf{x}$ (b) Train RBM for $\mathbf{h}^1$ (c) Train RBM for $\mathbf{h}^2$ and y

[Hinton *et al.*]

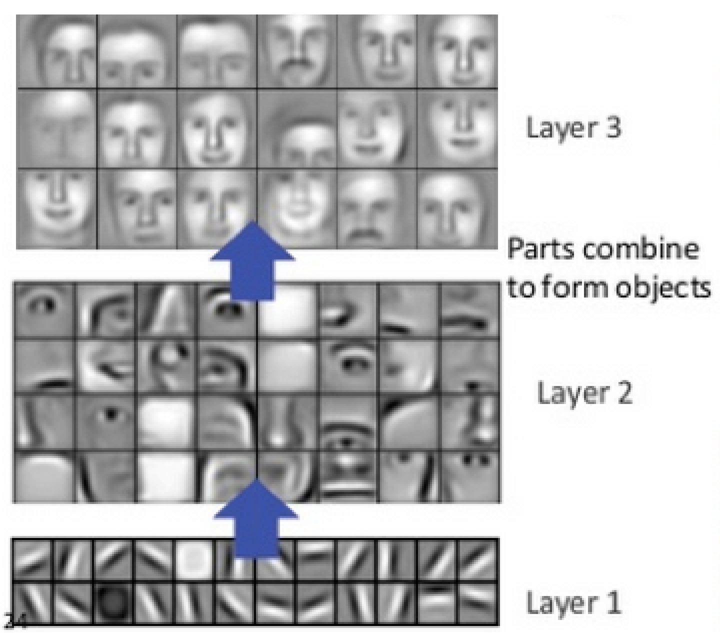
2012 - AlexNet

- Hinton' s group implemented a CNN similar to LeNet [LeCun1998] but...
  - Trained on Imagenet with two GPUs
  - With some technical improvements (ReLU, dropout, data augmentation)



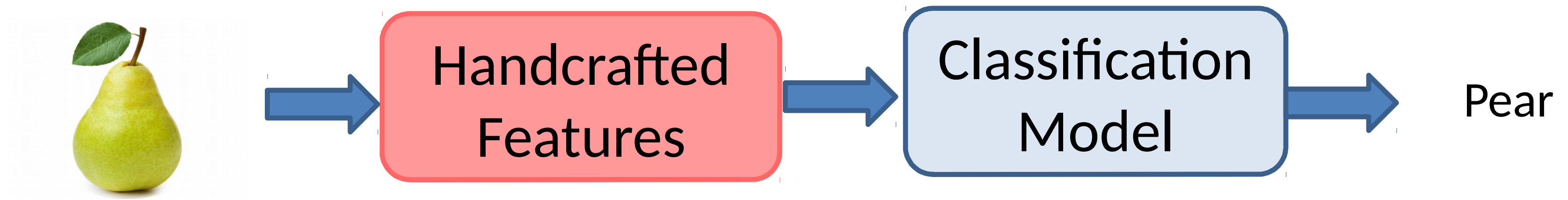
Why so powerful?

- Build an improved feature space
  - First layer learns first order features (e.g. edges…)
  - Subsequent layers learns higher order features (combinations of first layer features, combinations of edges, etc.)
  - Final layer of transformed features are fed into supervised layer(s)

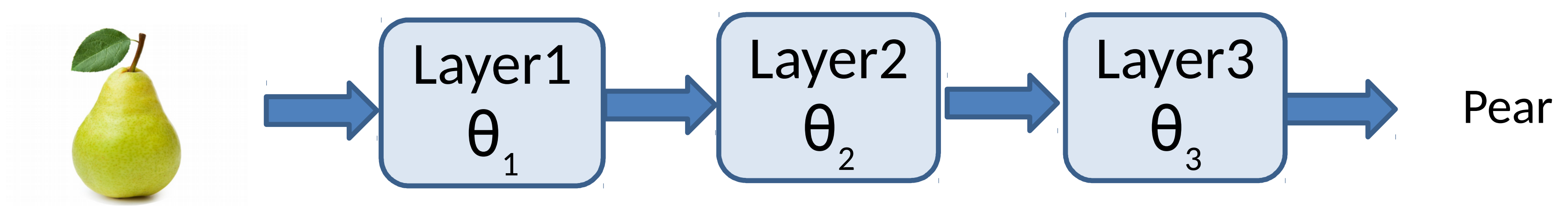


Learning hierarchical representations

- Traditional framework



- Deep Learning

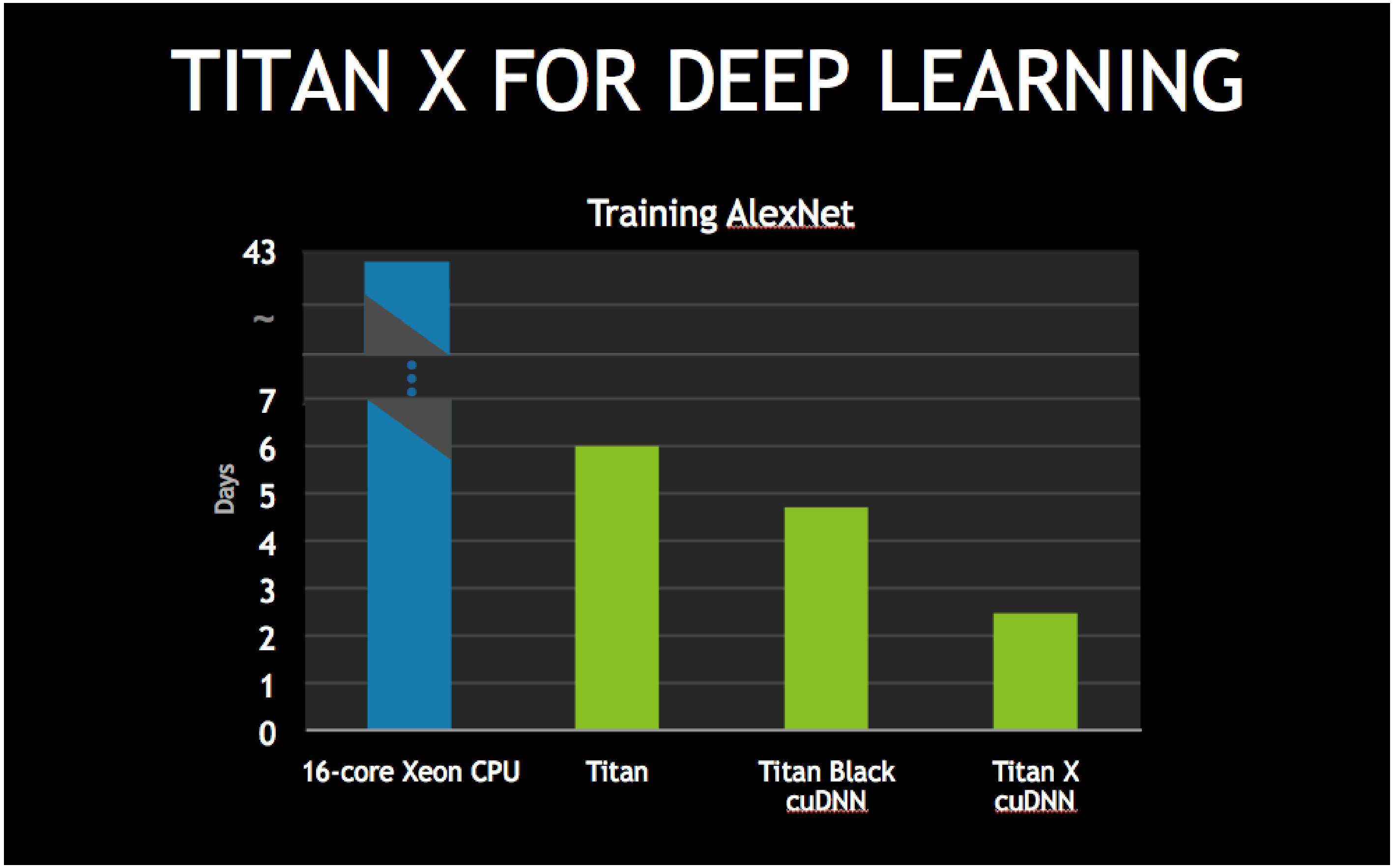


Why Deep Learning now?

- Three main factors:
  - Better hardware
  - Big data
  - Technical advances:
    - Layer-wise pretraining
    - Optimization (e.g. Adam, batch normalization)
    - Regularization (e.g. dropout)

    ....

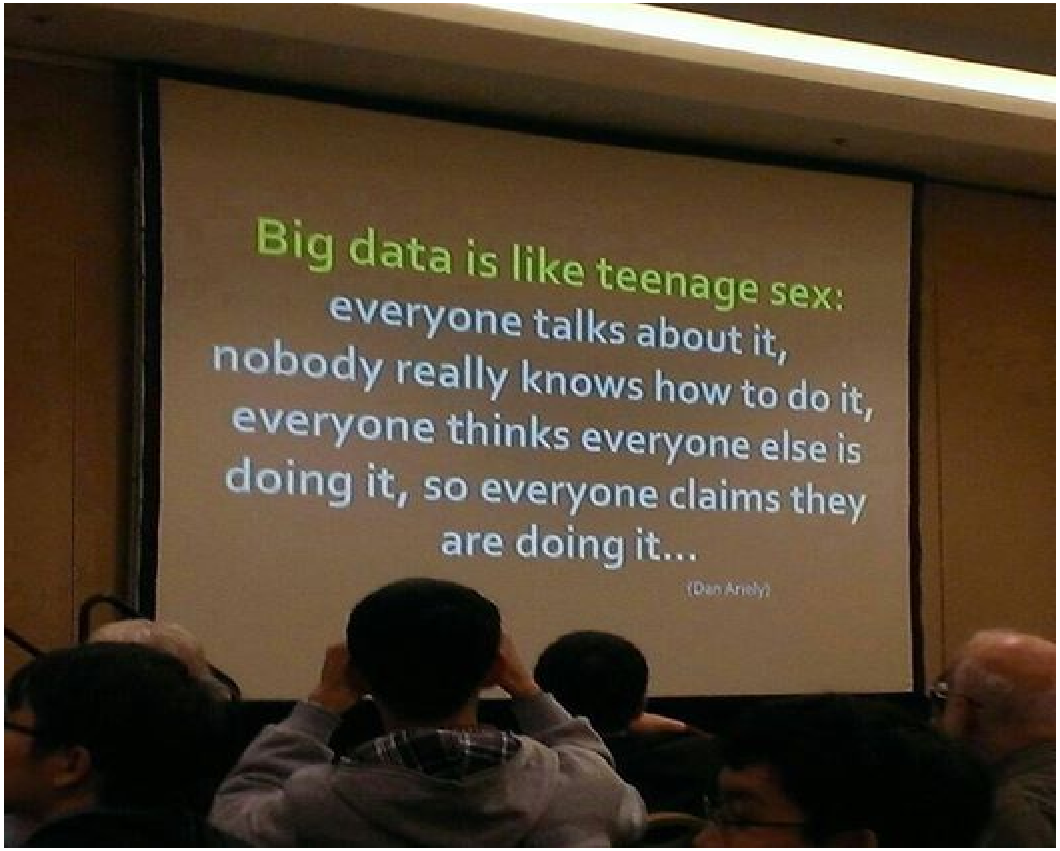
GPUs



NVIDIA Blog

Big Data

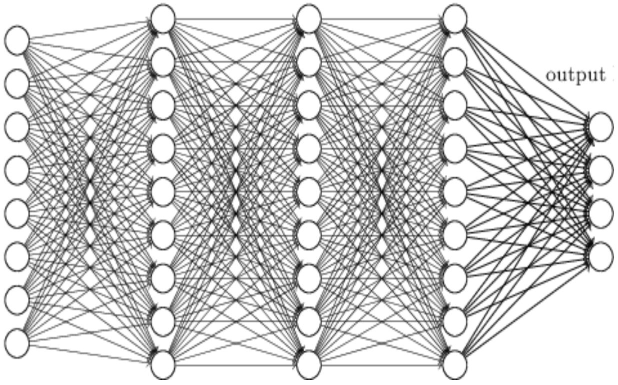
- Large **fully annotated** datasets



Advances with Deep Learning

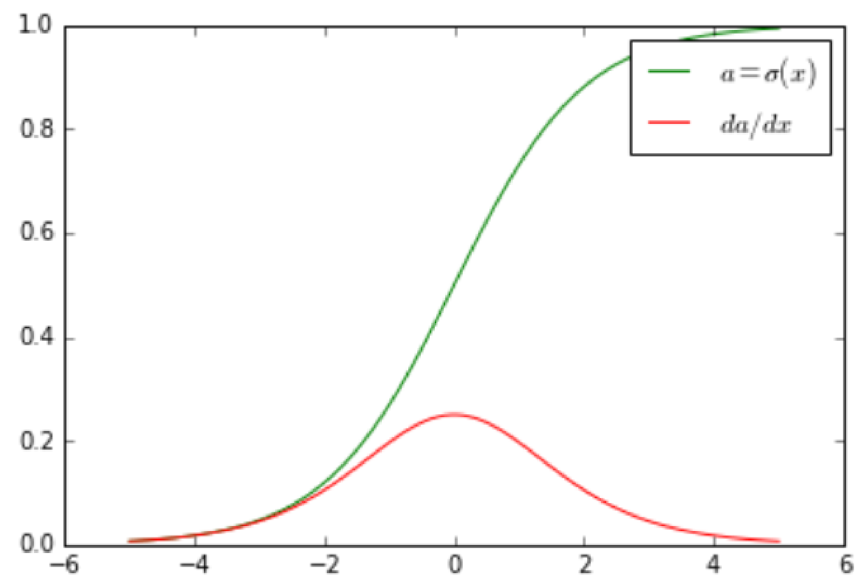
- Better:

  - Activation functions (RELU)
  - Training schemes
  - Weights initialization
  - Address overfitting (dropout)
  - Normalization between layers
  - Residual deep learning
  - ....



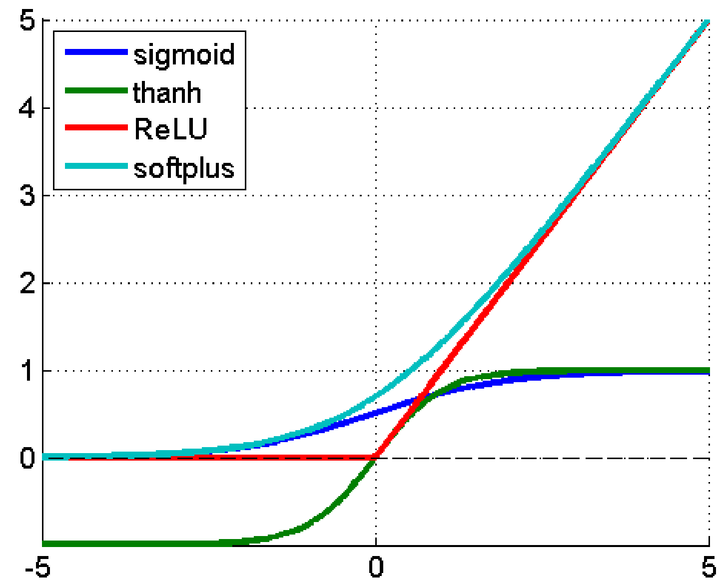
Sigmoid activations

- Positive facts:
  - Output can be interpreted as probability
  - Output bounded in [0,1]
- Negative facts
  - Always multiply with <1, gradients can be small
  - The gradients at the tails is flat to 0, almost no weights updates



Rectified Linear Units

$$f(x) = \max(0, x)$$



- More efficient gradient propagation:
  (derivative is 0 or constant)
- More efficient computation:
  (only comparison, addition and multiplication).
- Sparse activation: e.g. in a randomly initialized networks, only about 50% of hidden units are activated (having a non-zero output)
- Lots of variations have been proposed recently.

Losses

- ***Sum-squared error (L2) loss*** gradient seeks the maximum likelihood hypothesis under the assumption that the training data can be modeled by Normally distributed noise added to the target function value.
- Fine for regression but less natural for classification.
- For *classification* problems it is advantageous and increasingly popular to use the ***softmax*** activation function, just at the output layer, with the ***cross-entropy loss*** function.

Softmax and Cross Entropy

- Softmax: softens 1 of *k* targets to mimic a probability vector for each output.

$$softmax(x_k) = \frac{e^{x_k}}{\sum_j e^{x_j}}$$

Softmax and Cross Entropy

- Cross entropy loss: most popular classification losses for classifiers that output probabilities:

$$\ell(y, a) = -\sum_k y^k \log a^k$$

- Generalization of logistic regression for more than two outputs.
- These new loss and activation functions helps avoid gradient saturation.

Stochastic Gradient Descent (SGD)

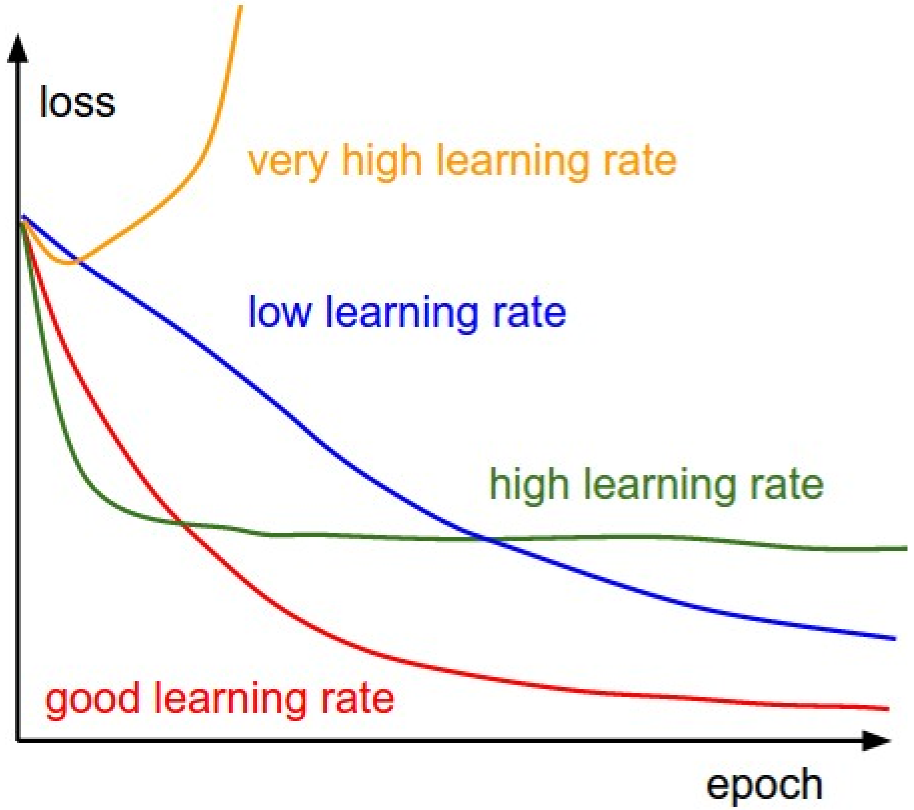
- Use mini-batch **sampled** in the dataset for gradient estimate.

$$\Theta^{t+1} = \Theta^{t} - \frac{\eta_t}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\Theta} \mathcal{L}_i$$

- Sometimes helps to escape from local minima
- Noisy gradients act as regularization
- Also suitable for datasets that change over time
- Variance of gradients increases when batch size decreases
- Not clear how many sample per batch

Learning rate

- Great impact on learning performance

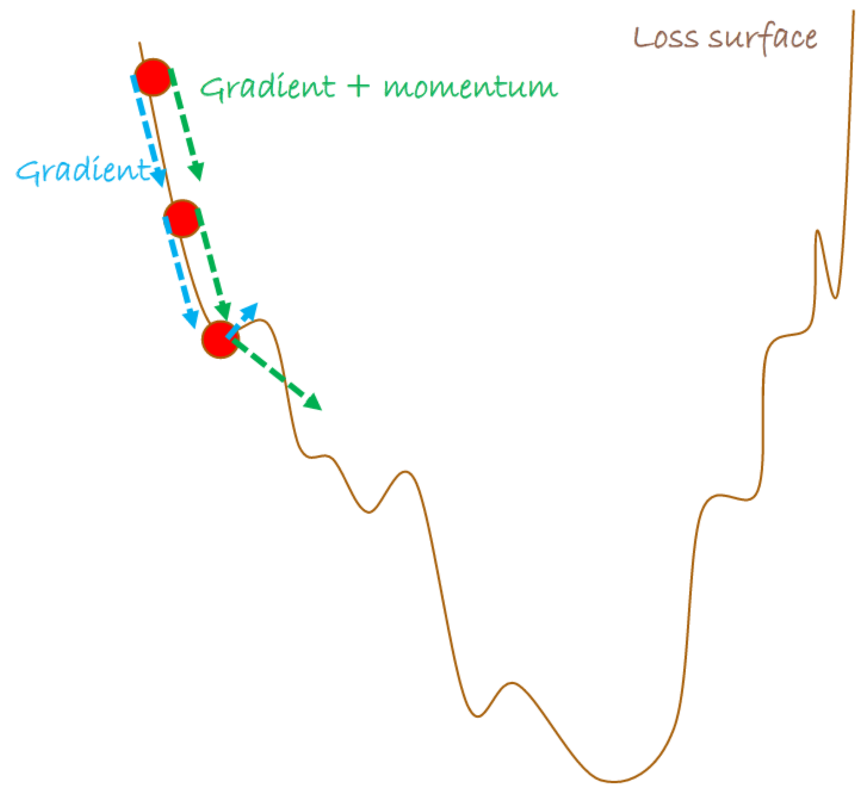


Momentum

- Gradient updates with momentum

$$\hat{\mathbf{\Theta}}^t = \gamma \mathbf{\Theta}^t + \frac{\eta_t}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\Theta} \mathcal{L}_i$$

$$\mathbf{\Theta}^{t+1} = \mathbf{\Theta}^t - \hat{\mathbf{\Theta}}^t$$



- Prevent gradient switching all the time
- Faster and more robust convergence

Adaptive Learning

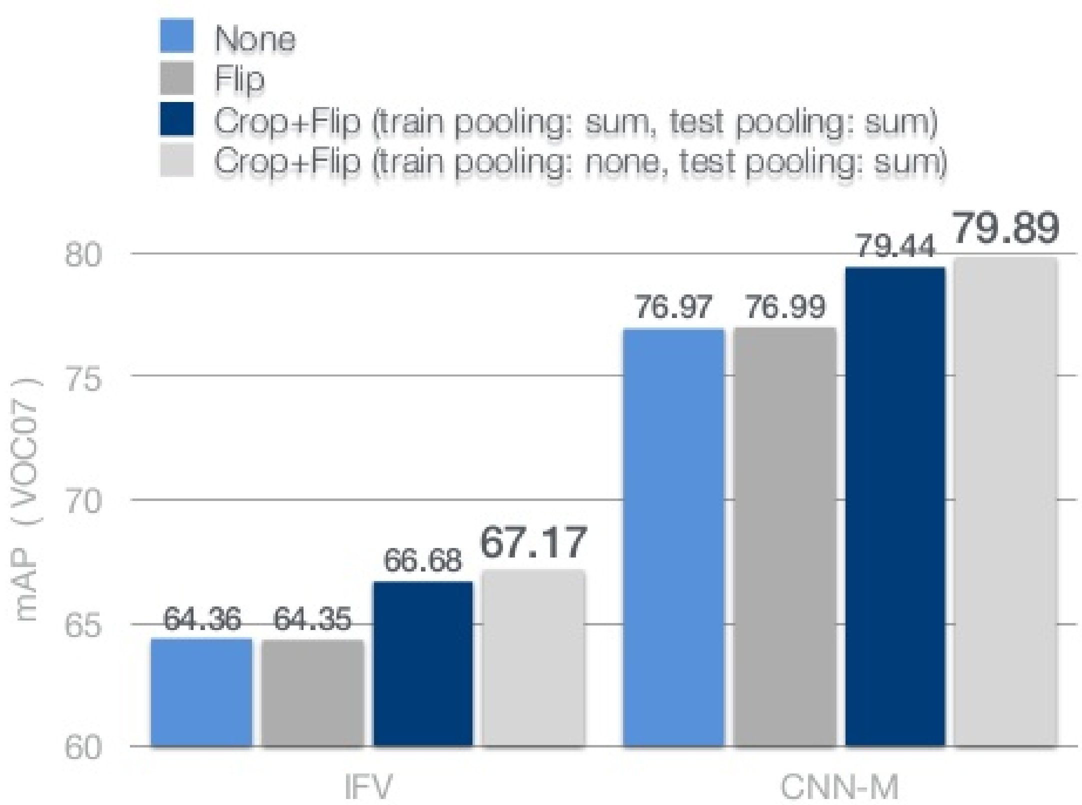
- Popular schemes
  - **Nesterov Momentum** – Calculate point you would go to if using normal momentum. Then, compute gradient at that point. Do normal update using *that* gradient and momentum.
  - Rprop – Resilient BP, if gradient sign inverts, decrease its individual learning rates, else increase it.
  - Adagrad – Scale learning rates inversely proportional to sqrt(sum(historical values)), such that learning rates with smaller derivatives are decreased less
  - RMSprop – Adagrad but uses exponentially weighted moving average, older updates basically forgotten
  - **Adam (Adaptive moments)**: Momentum terms on both gradient and squared gradient (1^{st} and 2^{nd} moments) – update based on both

Data augmentation

- Simple preprocessing makes the difference (e.g. image flipping, scaling)

Data augmentation

- Simple preprocessing makes the difference (e.g. image flipping, scaling)



Weights initialization

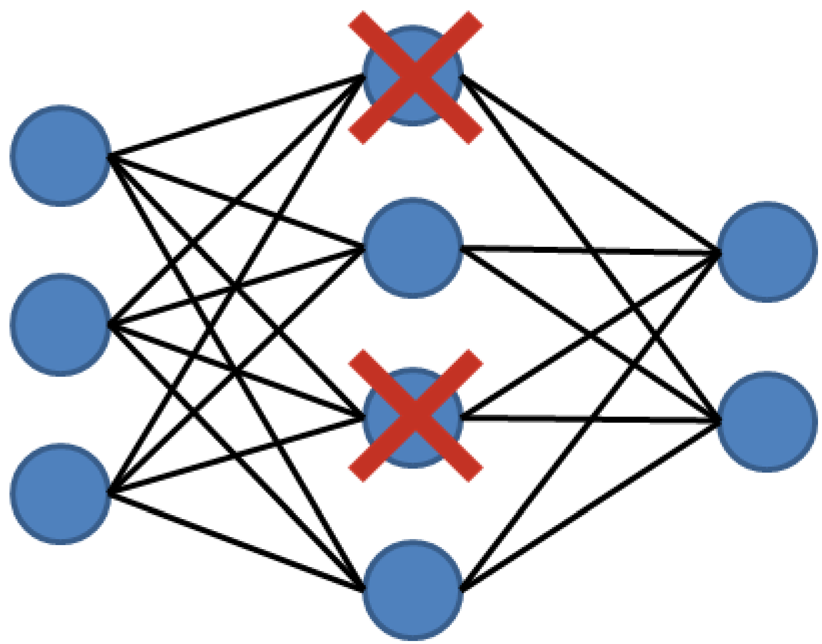
- Initialization depends on chosen non-linearities and data normalization
- Initial weights are important to find a good balance among layers and which learns well across all layers.
- Common is to select initial weights from a uniform distribution between:

  [-*c*/root(node fan-in), *c*/root(node fan-in)] (*c* = 1 Xavier, *c* = 2 He)

- Can do Gaussian distribution with above as variances
- Lots of other variations and current work

Regularization - Dropout

- For each instance drop a node (hidden or input) and its connections with probability *p* and train
- Final net just has all averaged weights (actually scaled by 1-*p*)
- As if ensembling 2^n different network substructures



Batch Normalization

- To maintain learning balance renormalize activations at each layer
- Obtain zero-mean and unit variance inputs: re-normalize the activation/net values at each input dimension *k* at each layer

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

- Want mean and variance of that activation for the entire data set. Approximate the empirical mean and variance over a mini-batch of instances and then normalize the activation.

[Ioffe and Szegedy, 2015]

Batch Normalization

- Then scale and shift the normalized activation with two learnable weights per input, γ and β, to attain the final batch normalization for that activation:

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}$$

- BN advantages:
  - Allows larger learning rates
  - Improves gradient flow
  - Reduces dependence on initialization

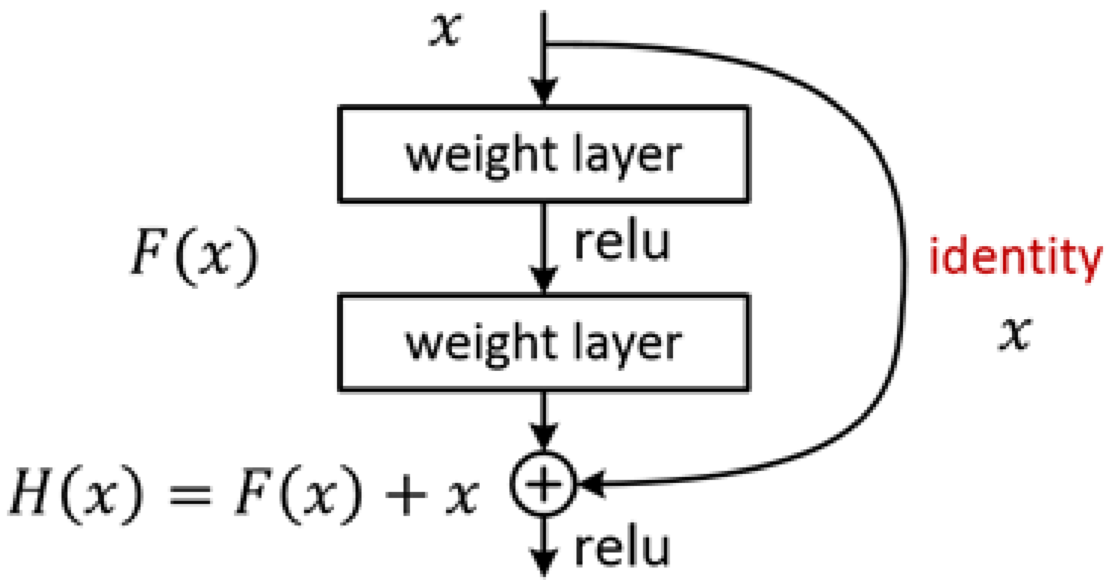
[Ioffe and Szegedy, 2015]

Deep Residual Learning

- Residual Nets
- 2015 ILSVRC winner
- A CNN with hundreds of layers
- Uses Batch Normalization extensively
- Learns the residual mapping with respect to the identity
- Simple concept which tends to make the function to be learned ***simpler*** across depth

Deep Residual Learning

- *F* is a residual mapping of the desired function *H* with respect to identity
- If the optimal mapping close to identity, small fluctuations.

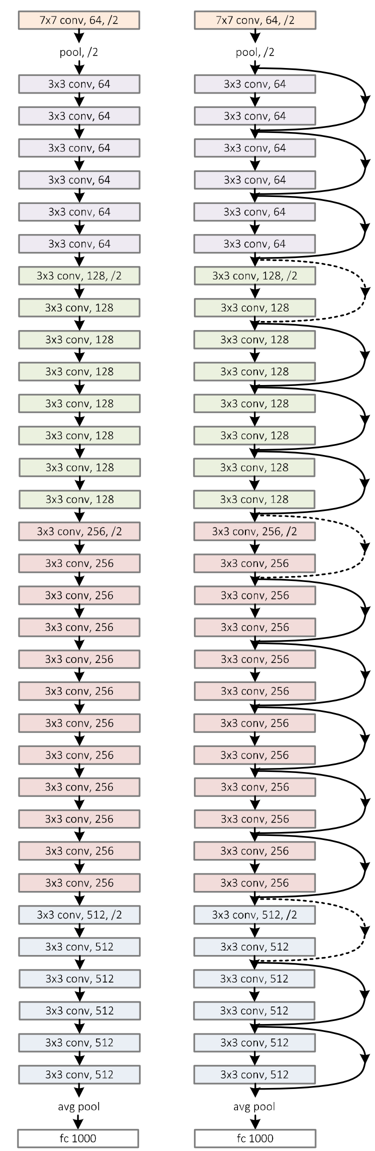


[He]

Deep Residual Learning

- Very simple design but deep

plain net



ResNet

·)

References

[Andrychowicz2016] Andrychowicz, Denil Gomez, Hoffman, Pfau, Schaul, de Freitas, Learning to learn by gradient descent by gradient descent, arXiv, 2016

[He2015] He, Zhang, Ren, Sun. Delving Deep into Rectifiers: Surpassing Human Level Performance on ImageNet Classification, ICCV, 2015

[Ioffe2015] Ioffe, Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv, 2015

[Kingma2014] Kingma, Ba. Adam: A Method for Stochastic Optimization, arXiv, 2014

[Srivastava2014] Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR, 2014

[Sutskever2013] Sutskever, Martens, Dahl, Hinton. On the importance of initialization and momentum in deep learning, JMLR, 2013

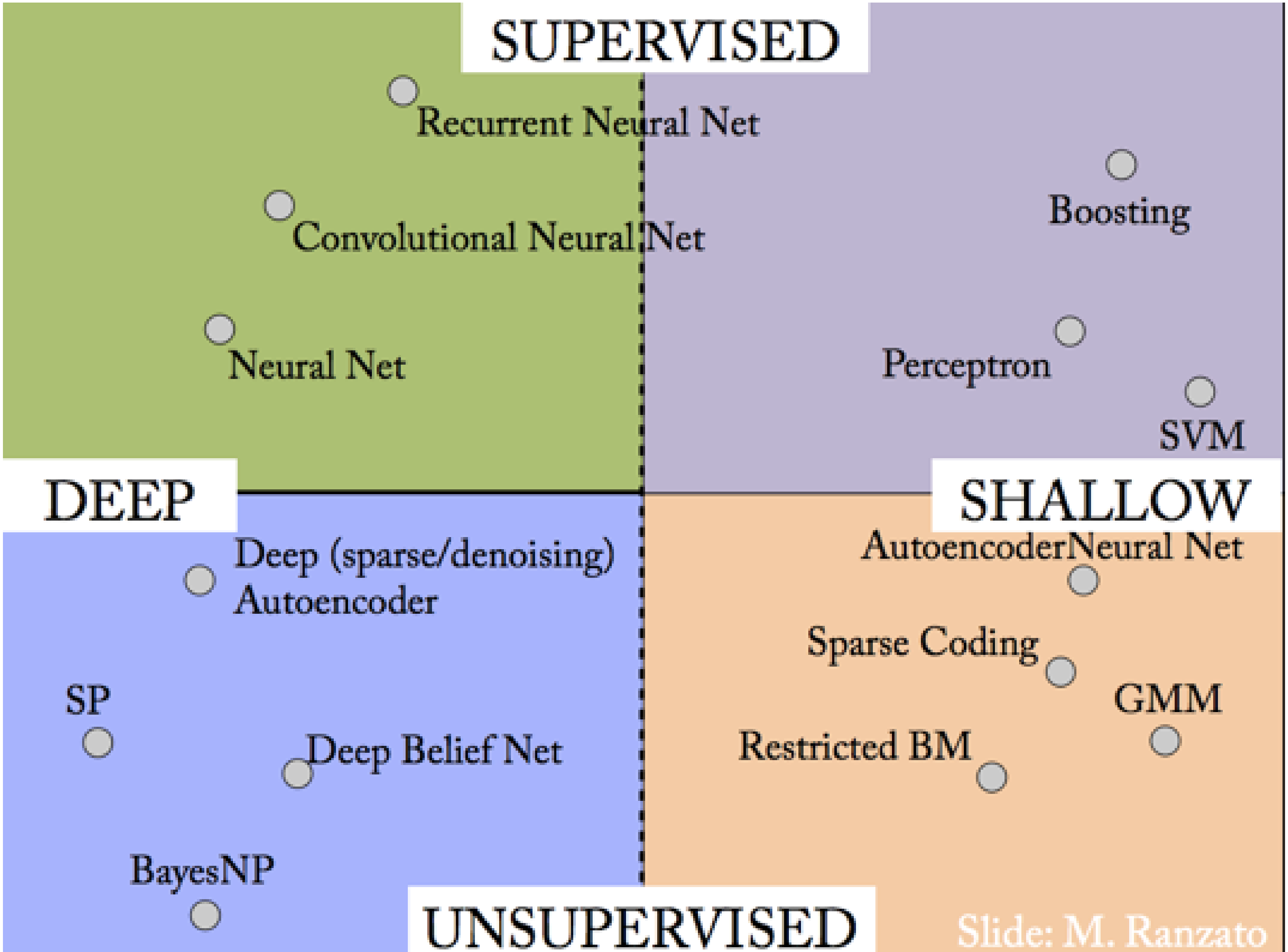
[Bengio2012] Bengio, Practical recommendations for gradient-based training of deep architectures, ArXiv, 2012

[Krizhevsky2012] Krizhevsky, Hinton. ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012

[Duchi2011] Duchi, Hazan, Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, JMLR, 2011

[Glorot2010] Glorot, Bengio. Understanding the difficulty of training deep feedforward neural networks, JJMLR, 2010

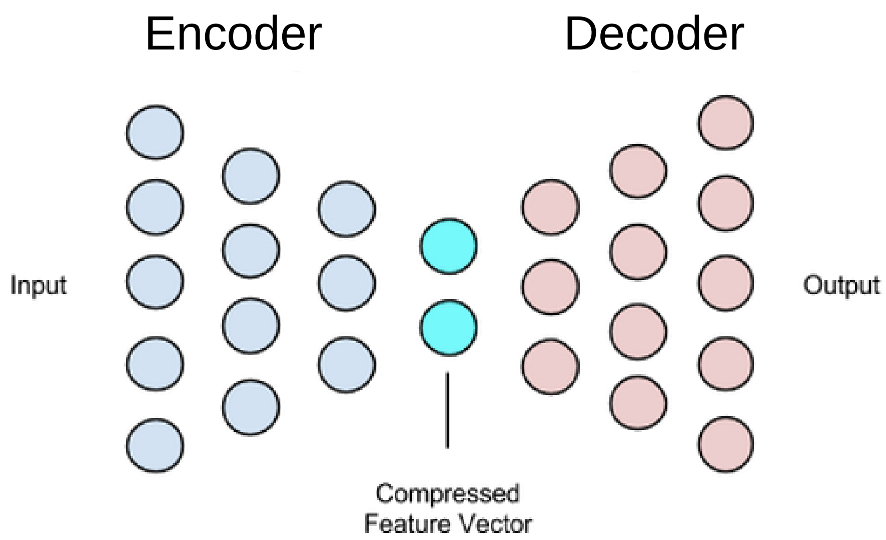
Deep Learning Models


SUPERVISED
Recurrent Neural Net
Boosting
Convolutional Neural Net
Neural Net
Perceptron
SVM
DEEP
SHALLOW
Deep (sparse/denoising)
Autoencoder
AutoencoderNeural Net
Sparse Coding
GMM
SP
Deep Belief Net
Restricted BM
BayesNP
UNSUPERVISED
Slide: M. Ranzato

Deep Learning Models

Deep Belief Networks and ***Autoencoders:***

unsupervised learning, employs layer-wise training to initialize each layer and capture multiple levels of representation simultaneously.



Hinton, G. E, Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527-1554.

Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks, Advances in Neural Information Processing Systems 19

Autoencoders

- The *auto encoder* idea is motivated by the concept of a good representation.
  - For example, for a classifier, a good representation can be defined as one that will yield a better performing classifier.
- An *encoder* is a deterministic mapping ***f*** that transforms an input vector ***x*** into hidden representation ***y***
  - Parameters in ***f:*** weight matrix W and bias ***b*** (an offset vector)
- A *decoder* maps back the hidden representation ***y*** to the reconstructed input ***z*** via ***g***.
- ***Auto encoding:*** compare the reconstructed input ***z*** to the original input ***x*** and try to minimize the reconstruction error.

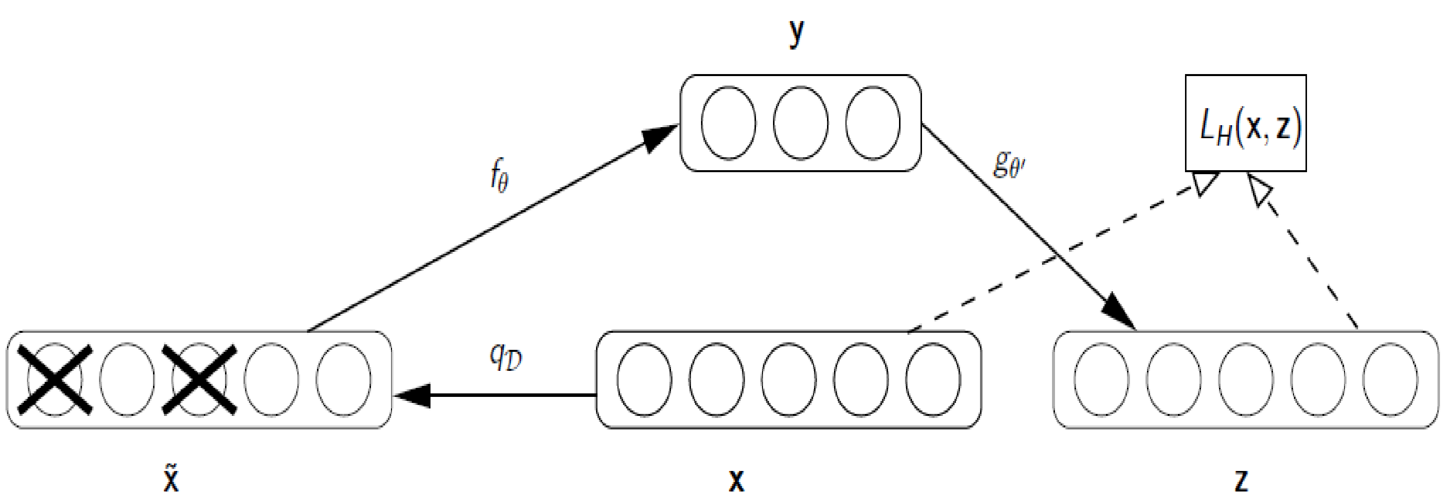
Denoising Autoencoders

- Vincent et al. (2010), "*a **good representation** is one that can be obtained **robustly** from a **corrupted input** and that will be useful for **recovering** the corresponding **clean input**.*"
  - The higher level representations are relatively stable and robust to input corruption.

- In denoising auto encoders, the partially *corrupted* output is cleaned (de-noised).

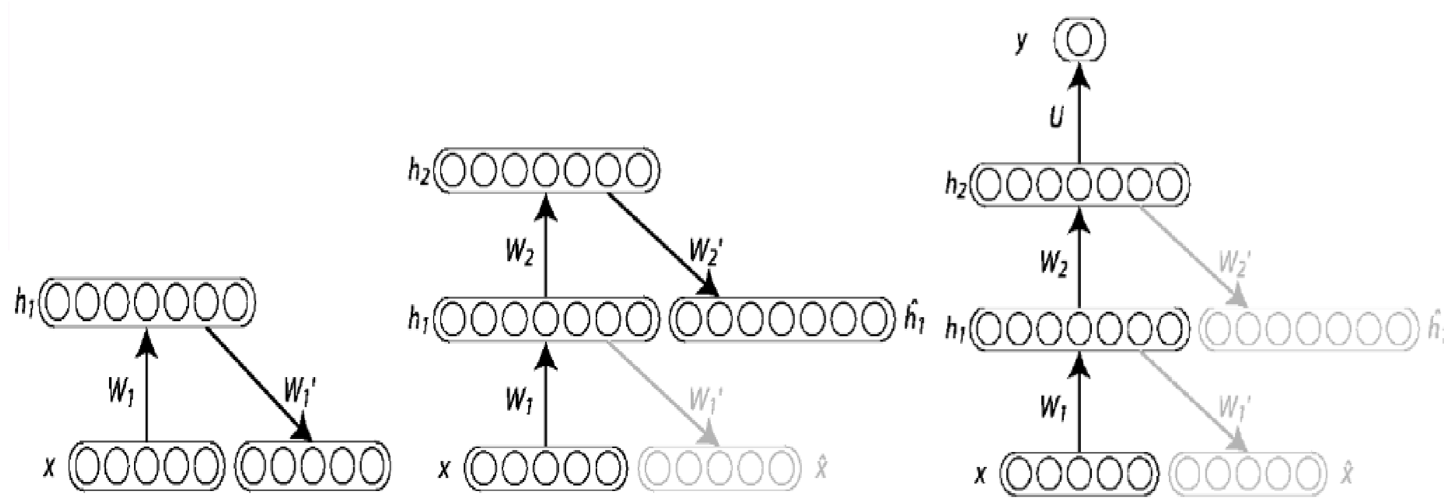
Denoising Autoencoders

1. Clean input is partially corrupted through a stochastic mapping.
2. The corrupted input passes through a basic auto encoder and is mapped to a hidden representation.
3. From this hidden representation, we can reconstruct *z*.
4. Minimize the reconstruction error (cross-entropy or squared error loss.



Stacked Denoising Autoencoders

- Deep architecture: auto encoders stack one on top of another.
- Once the encoding function of the first DAE is learned and used to reconstruct the corrupted input, we can train the second level.
- Once the SDAE is trained, its output can be used as the input to a supervised learning algorithm such as support vector machine classifier or a multi-class logistic regression.



Structured Data

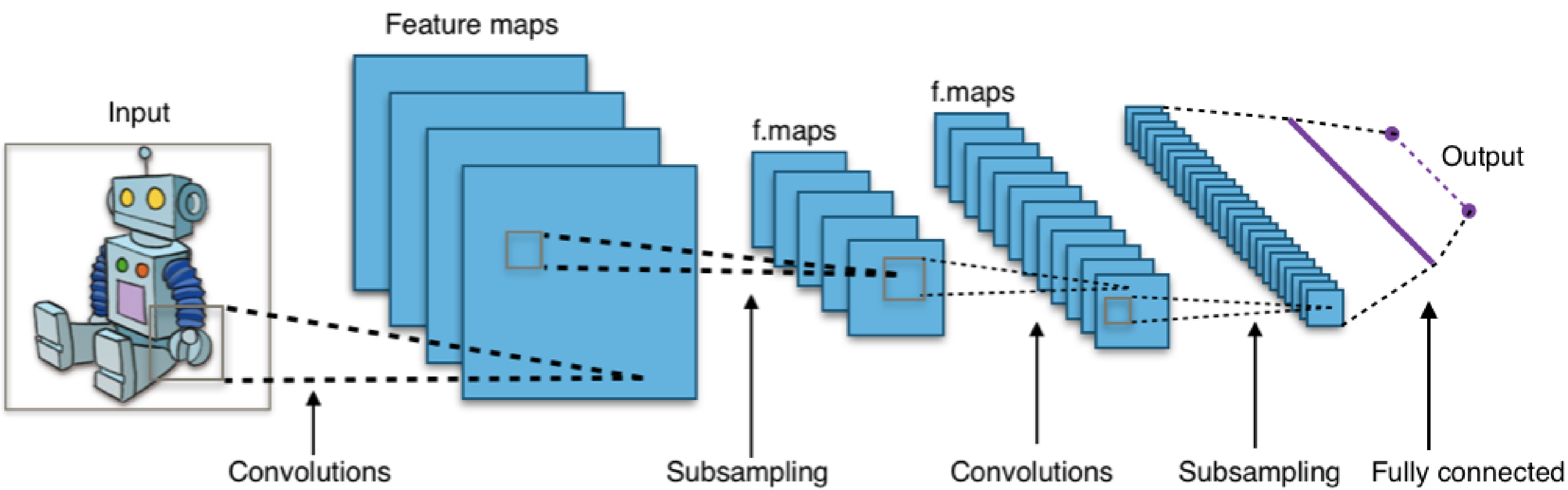
- Some applications naturally deal with an input space which is locally structured – spatial or temporal
- Images, language, etc. vs arbitrary input features
- Deep Learning extremely powerful in this case.

Tomorrow, and tomorrow, and tomorrow; creeps in this petty pace from day to day, until the last syll-able of recorded time. And all our yesterdays have lighted fools the way to dusty

Deep Learning Models

Convolutional Neural Networks:

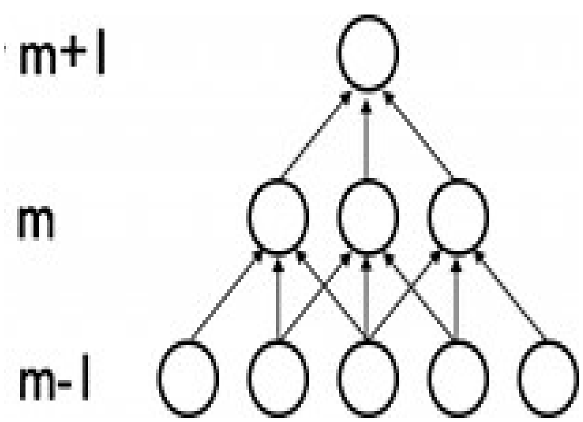
organizes neurons based on animal's visual cortex system, which allows for learning patterns at both local level and global level.

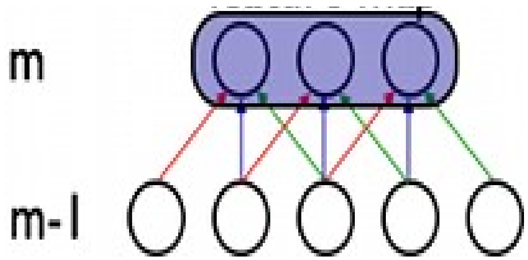


Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

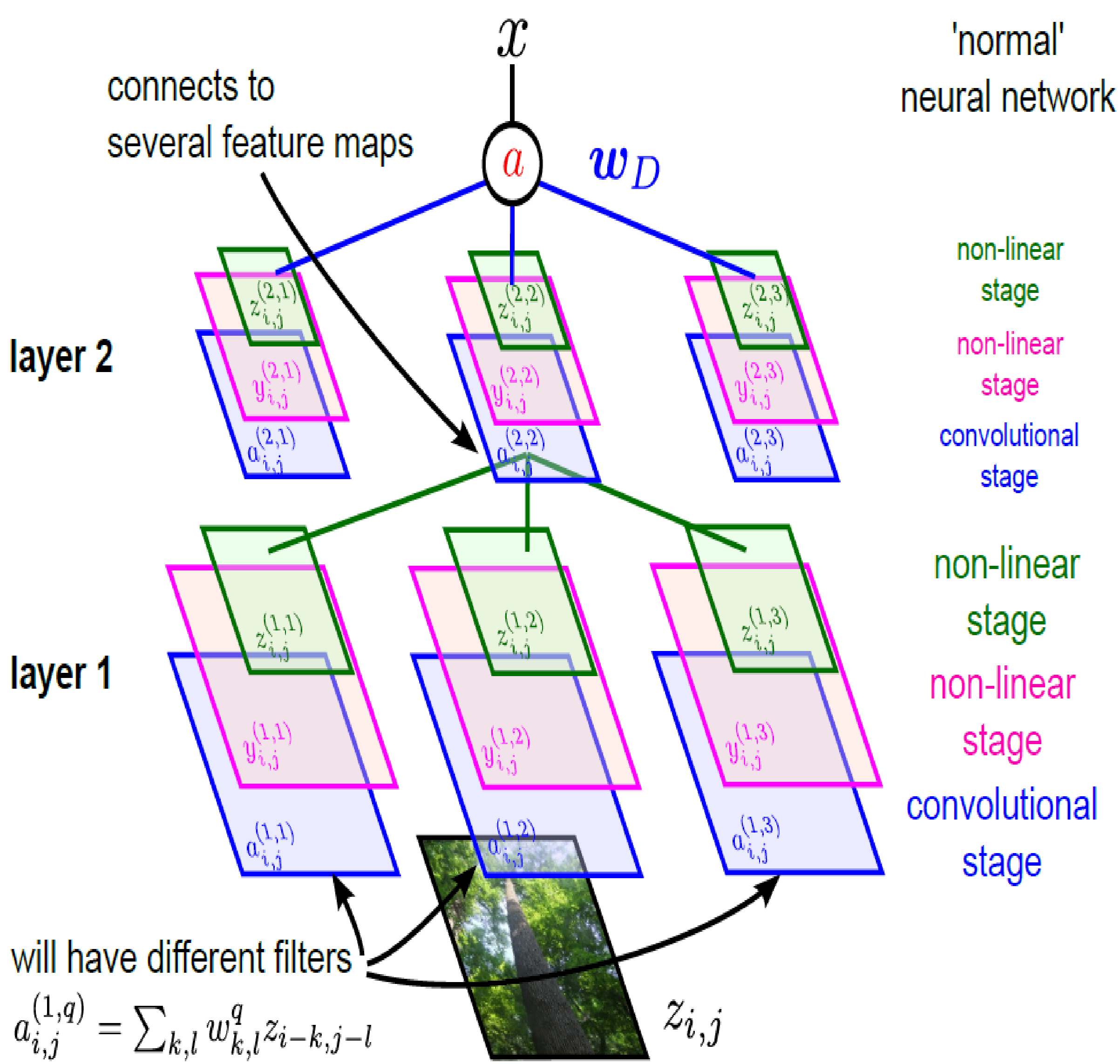
Convolutional Neural Networks

- CNN: a multi-layer neural network:
  - With **Local connectivity**:
    - Neurons in a layer are only connected to a small region of the layer before it
  - **Sharing** weight parameters across spatial positions:
    - Learning shift-invariant filter kernels
    - Reducing the number of parameters



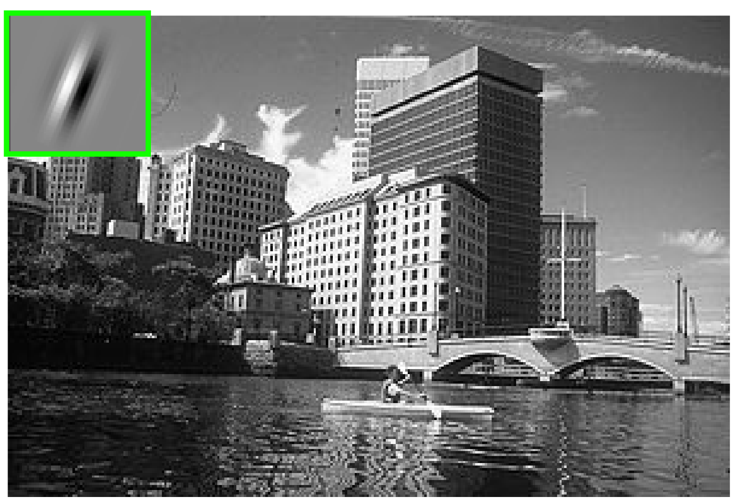


CNN Architecture

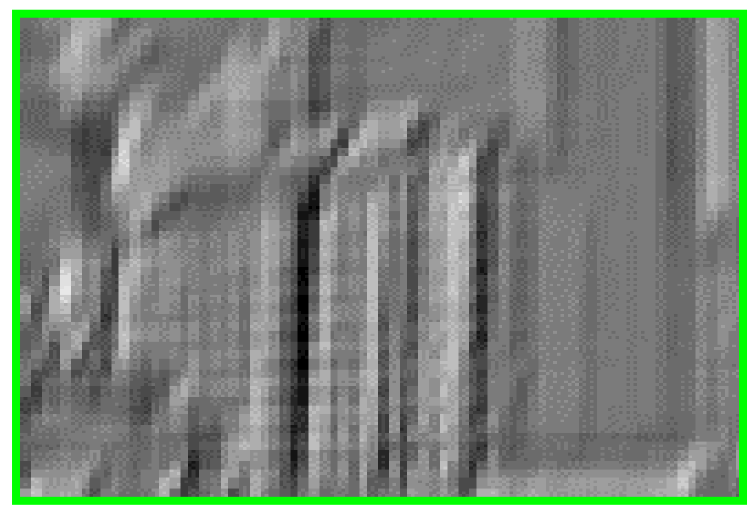


Convolutional Neural Networks

Input Image → Convolution (Learned) → Non-linearity → Spatial pooling → Normalize → Feature maps



Input



Feature Activation Map

Convolutional Neural Networks

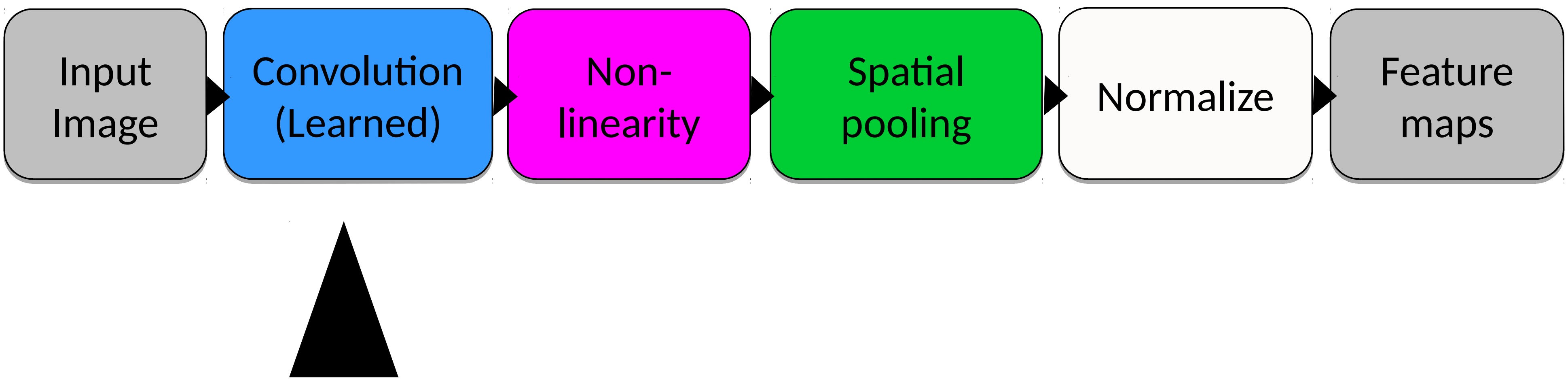
Input Image → Convolution (Learned) → Non-linearity → Spatial pooling → Normalize → Feature maps

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

Shared weights

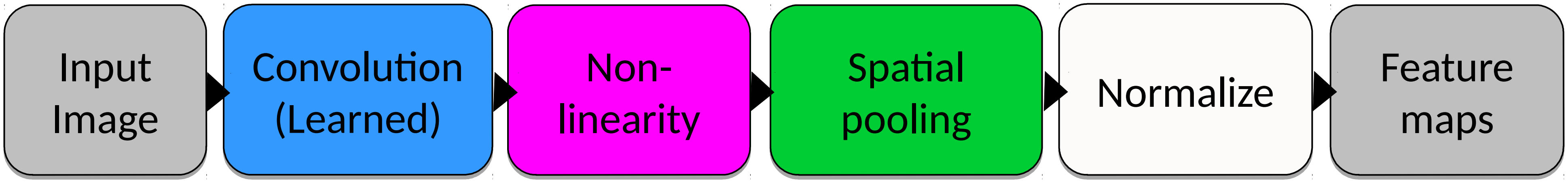
Each image sub-region yields a feature map, representing its feature.

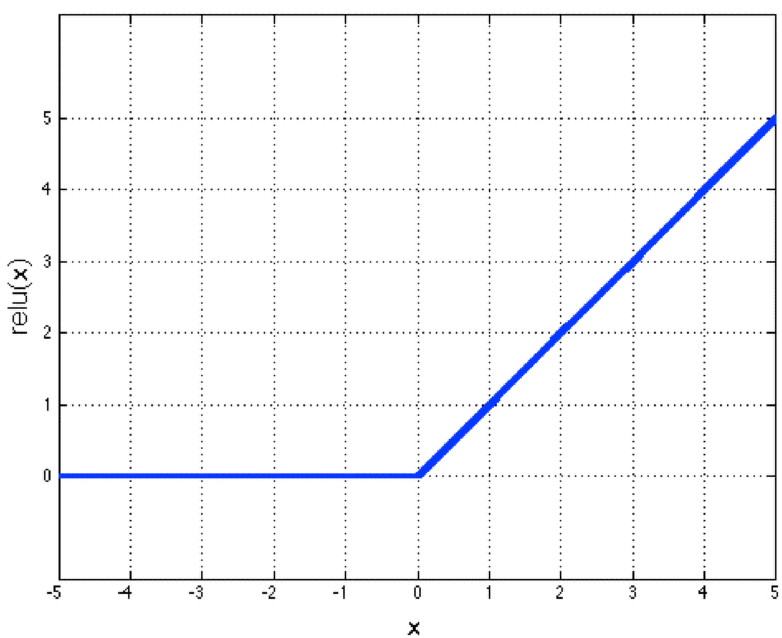
Convolutional Neural Networks



Convolutional filters are learned in a supervised manner by back-propagating classification error

Convolutional Neural Networks

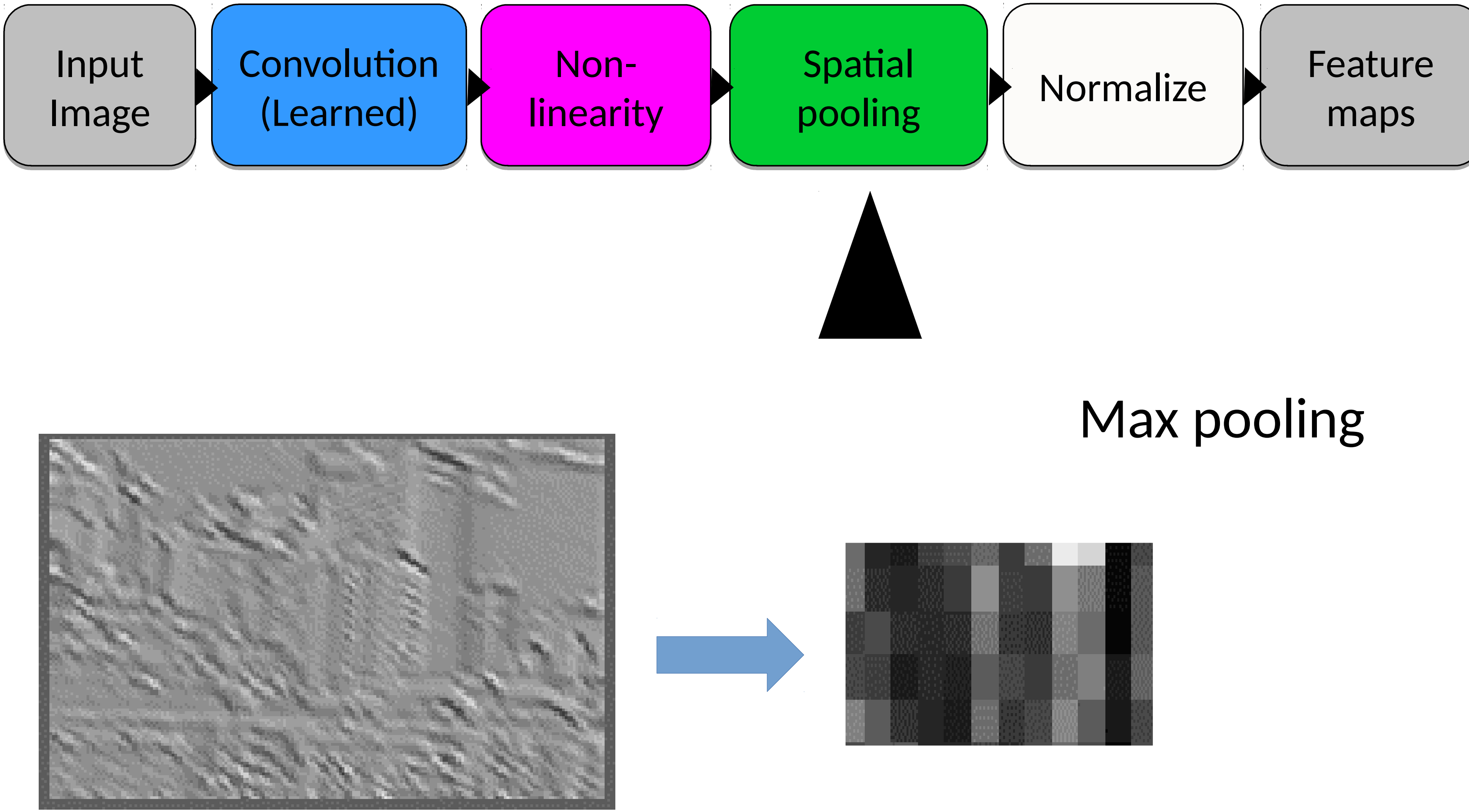




$$y_{i,j} = f(a_{i,j})$$

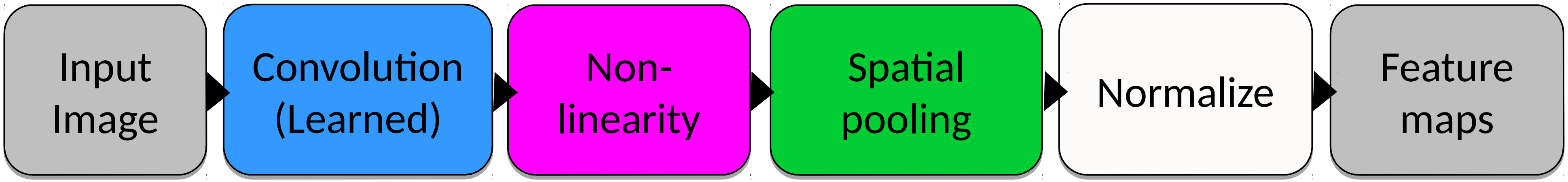
Non-linearity:
e.g. Rectified Linear Unit (ReLU)

Convolutional Neural Networks



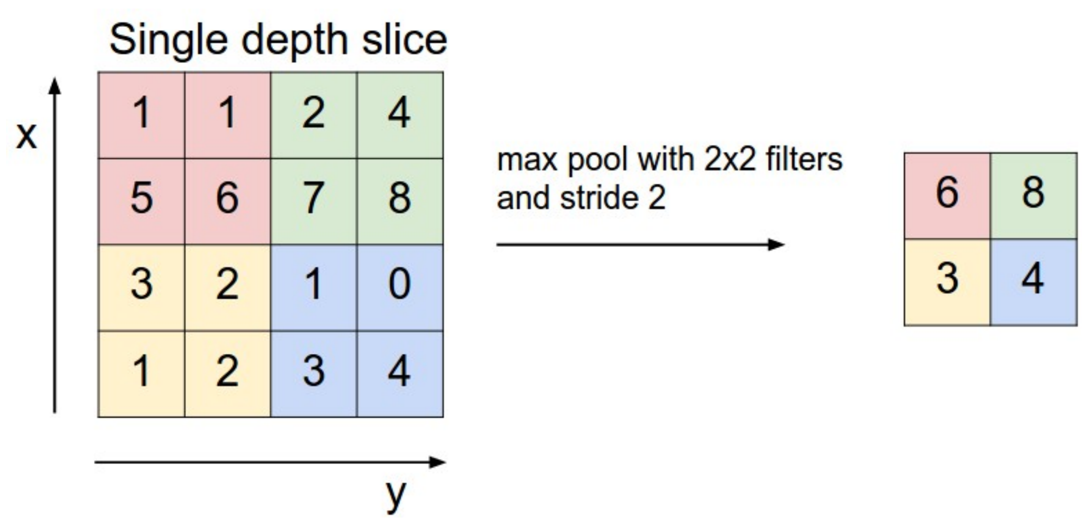
A non-linear down-sampling, to provide *translation invariance*

Convolutional Neural Networks

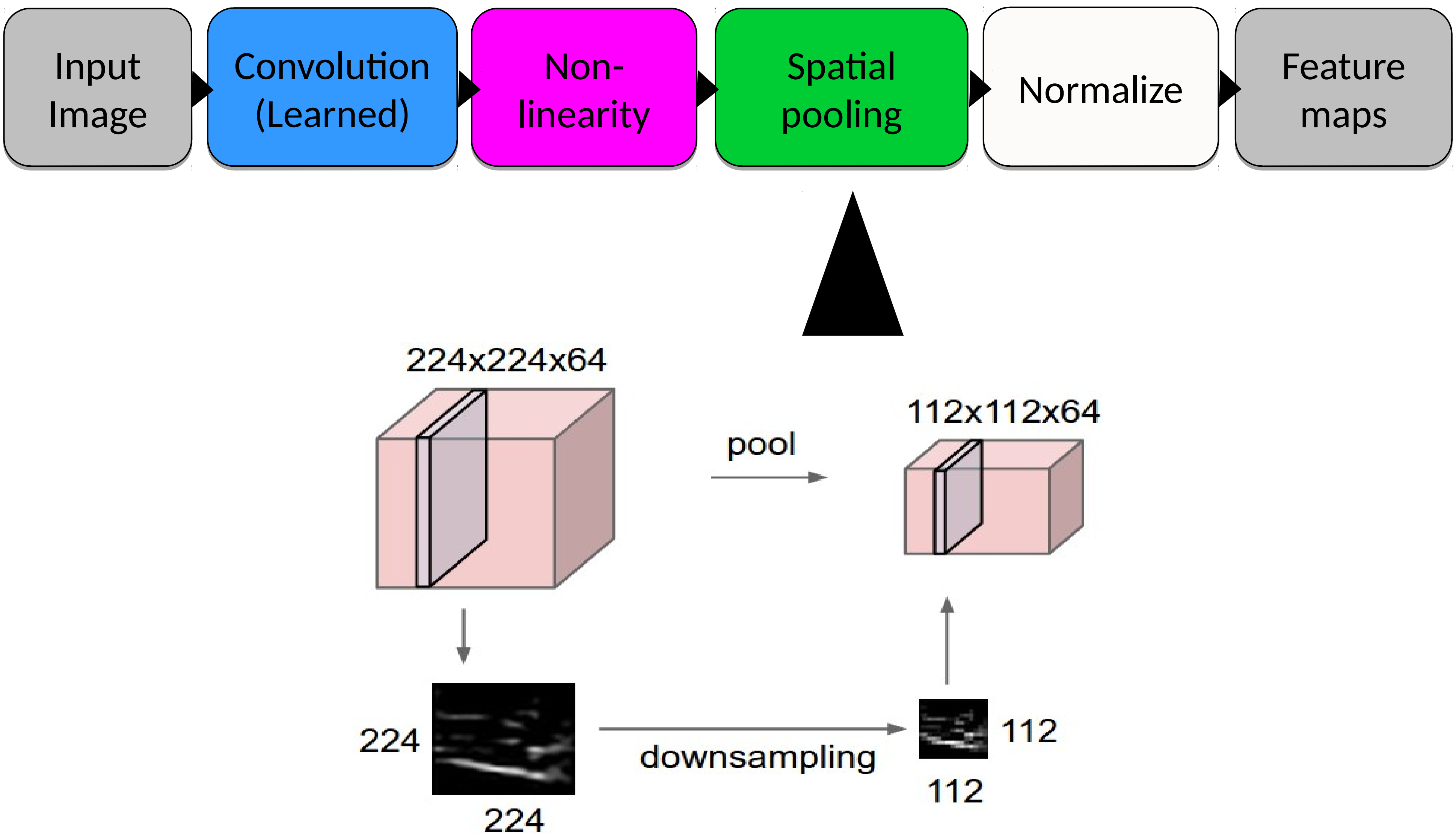


$$x_{i,j} = \max_{|k|<\tau, |l|<\tau} y_{i-k,j-l}$$

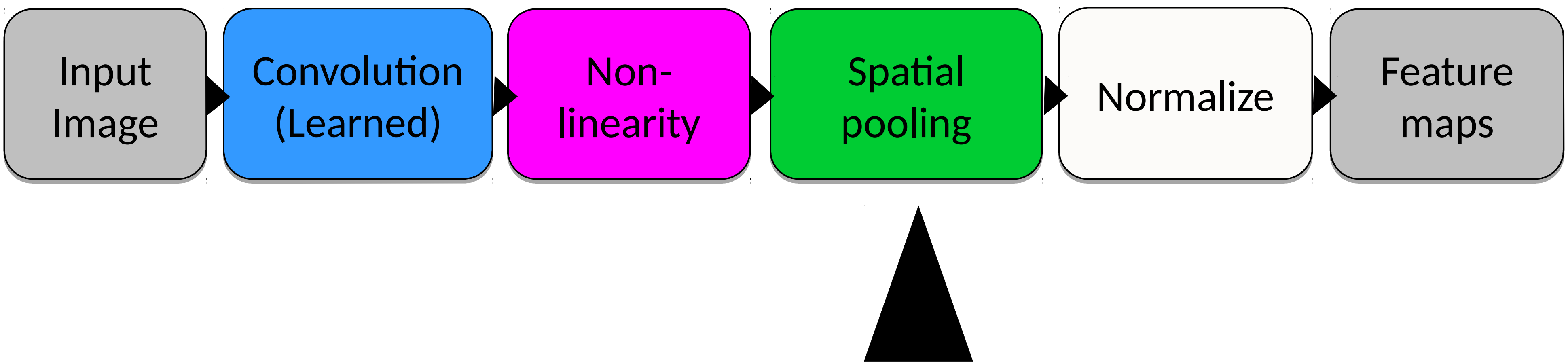
mean or subsample also used



Convolutional Neural Networks

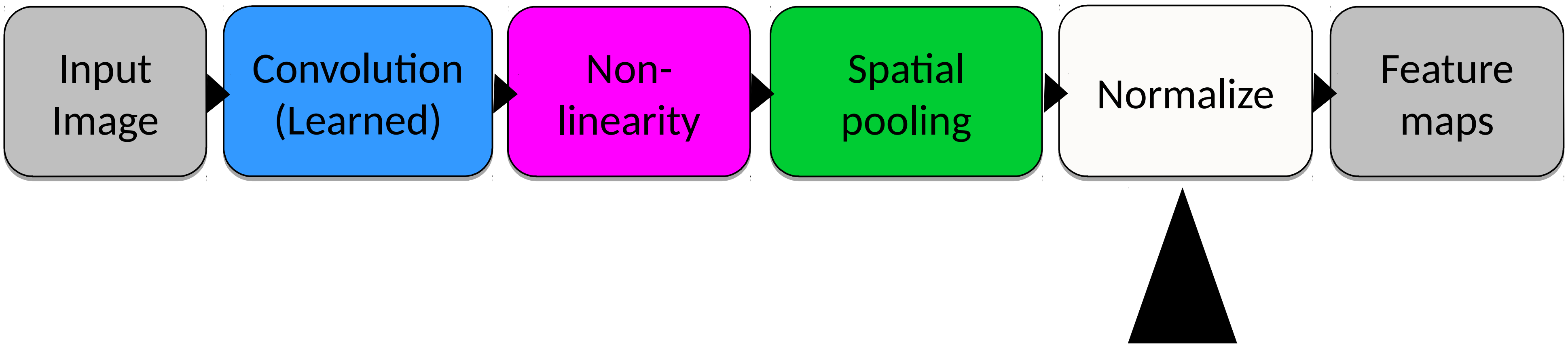


Convolutional Neural Networks



- By progressively reducing the spatial size of the representation we reduce the amount of parameters and computation in the network, and also control overfitting.

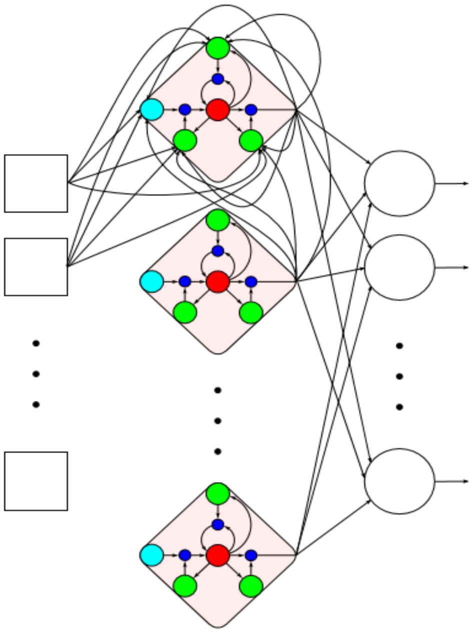
Convolutional Neural Networks



Deep Learning Models

Recurrent Neural Networks:

connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior.



Hochreiter, S, Schmidhuber (1997) Long Short-Term Memory, Neural Computation, 9(8):1735–1780, 1997

RNNs for sequences

Standard Neural Networks (and also CNN):

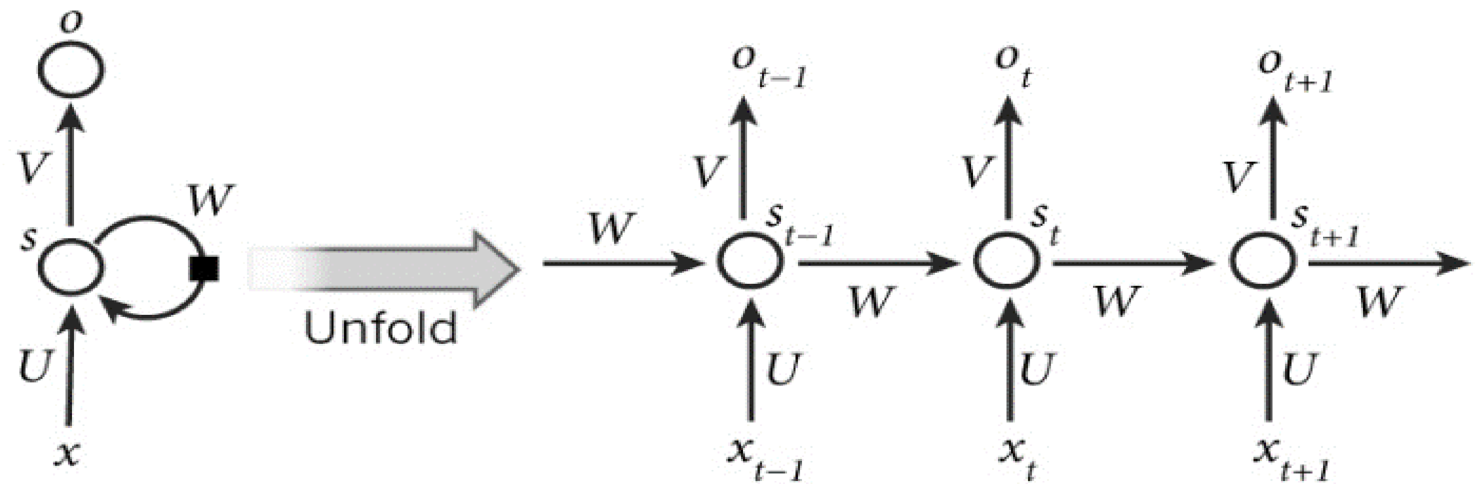
- Only accepted a fixed-size vector/matrix as input (e.g., an image) and produce a fixed-size vector as output (e.g., probabilities of different classes).
- These models use a fixed amount of computational steps (e.g. the number of layers in the model).

Recurrent Neural Networks are unique as they allow us to operate over sequences of vectors.

Sequences in the input, the output, or in the most general case both.

Recurrent Neural Networks

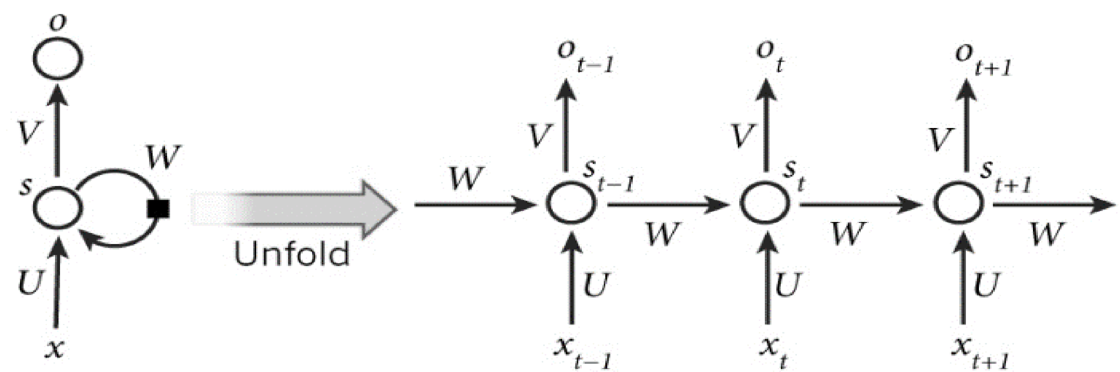
- An unrolled RNN (in time) can be considered as a deep neural network with indefinitely many layers:



Recurrent Neural Networks

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

$$y = g(Vs_t)$$



X_t : input at time

S_t : hidden state at time (memory of the network).

f: is an activation function (e.g, sigmoid, ReLU).

U, V, W: network parameters (unlike a feedforward neural network, an RNN shares the same parameters across all time steps).

g: activation function for the output layer (typically a softmax function).

y: the output of the network at time

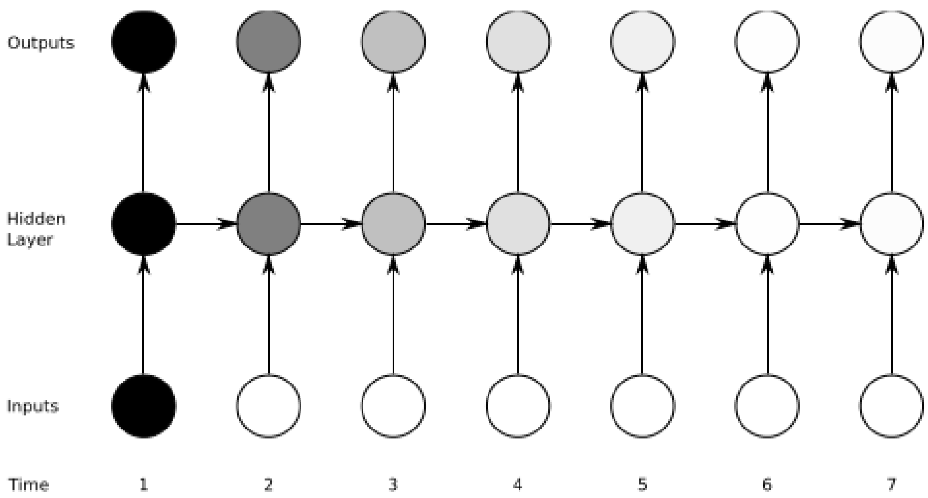
Back Propagation Through Time

- The backpropagation algorithm can be extended to BPTT by unfolding RNN in time and stacking identical copies of the RNN.
- As the parameters that are supposed to be learned (*U*, *V* and *W*) are ***shared*** by all time steps in the network, the gradient at each output depends, not only on the calculations of the current time step, but also the previous time steps.
- A common choice for the loss function is the cross-entropy loss.

Vanishing gradient

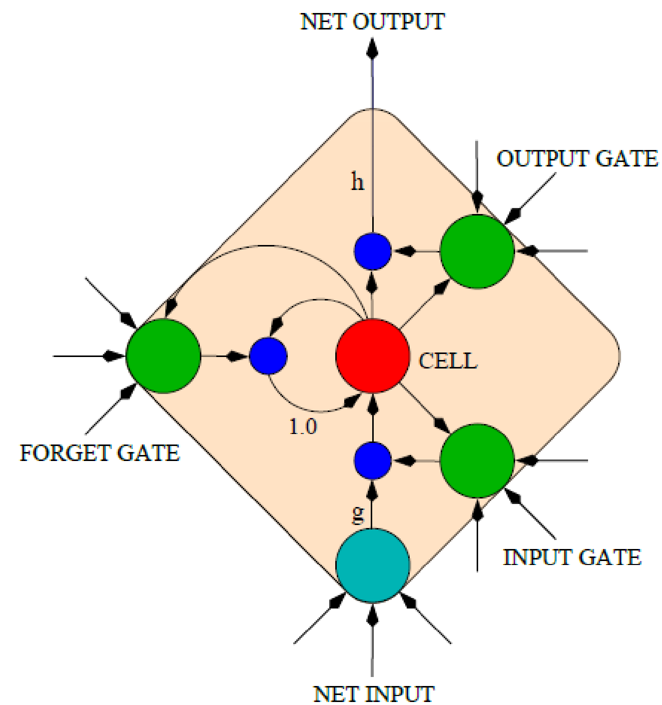
- Definition: The influence of a given input on the hidden layer, and therefore on the network output, either decays or grows exponentially as it propagates through an RNN.
- In practice, the range of contextual information that standard RNNs can access are limited to approximately 10 time steps between the relevant input and target events.

Solution: LSTM networks.



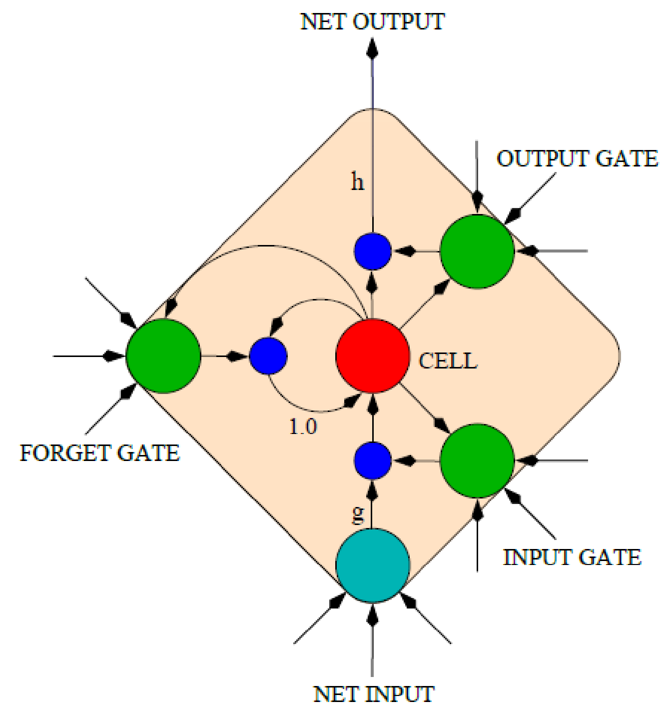
Long Short Term Memory (LSTM)

- An LSTM is a special kind of RNN architecture, capable of learning long-term dependencies.
- An LSTM can learn to bridge time intervals in excess of 1000 steps.
- This is achieved by multiplicative ***gate units*** that learn to open and close access to the constant error flow.



Long Short Term Memory (LSTM)

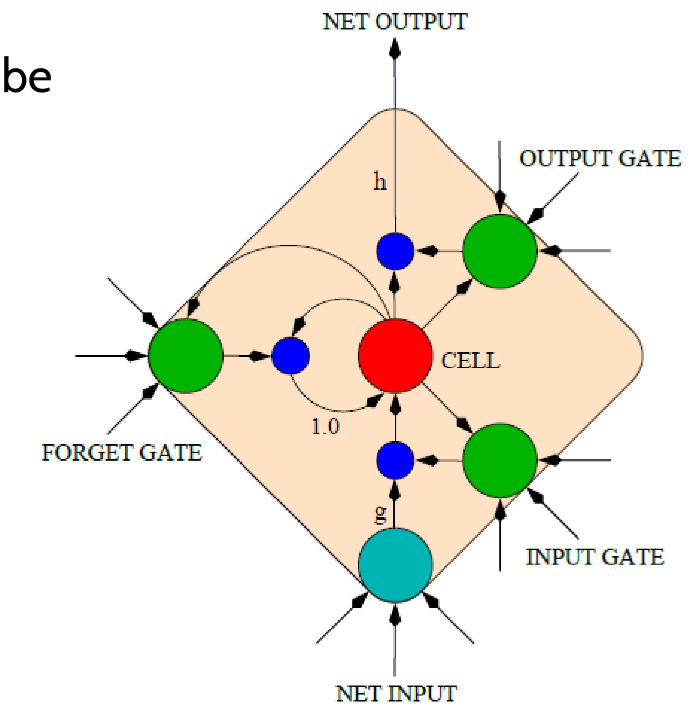
- LSTM networks introduce a new structure called a ***memory cell***.
- Each memory cell contains four main elements:
  - Input gate
  - Forget gate
  - Output gate
  - Neuron with a self-recurrent
- These gates allow the cells to keep and access information over long periods of time.



LSTM Memory Cell

Long Short Term Memory (LSTM)

- ***i*** : input gate, how much of the new information will be let through the memory cell.
- ***f*** : forget gate, responsible for information should be thrown away from memory cell.
- ***o*** : output gate, how much of the information will be passed to expose to the next time step.
- ***g*** : self-recurrent which is equal to standard RNN
- c_t: internal memory of the memory cell
- s_t: hidden state
- y : final output

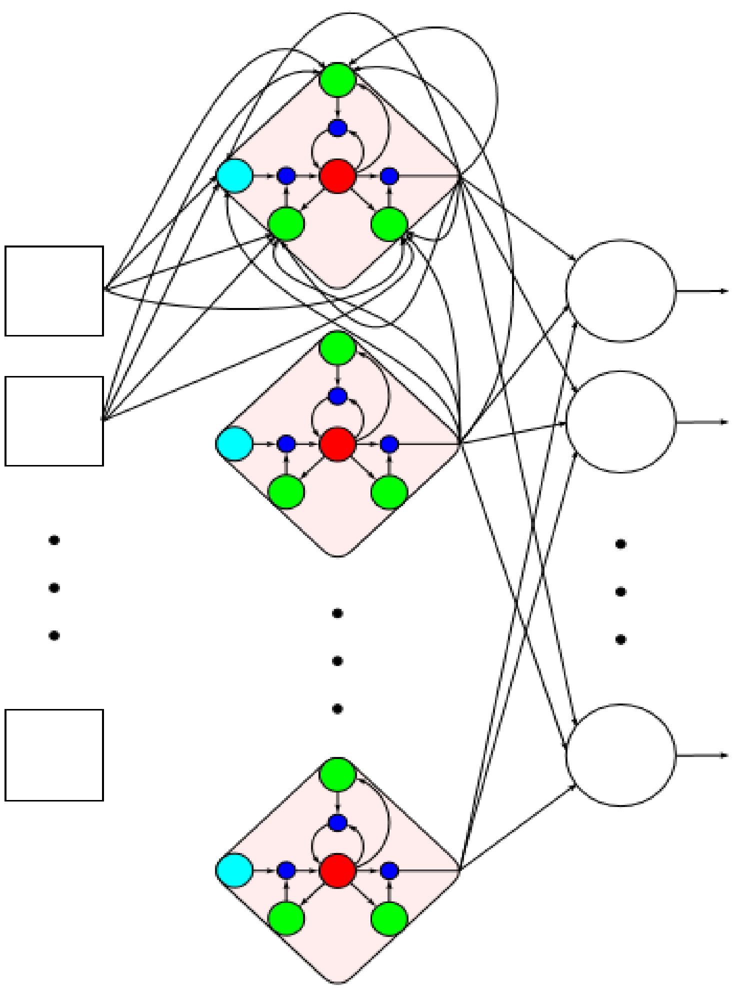


Long Short Term Memory (LSTM)

- ***i*** : input gate, how much of the new information will be let through the memory cell.
- ***f*** : forget gate, responsible for information should be thrown away from memory cell.
- ***o*** : output gate, how much of the information will be passed to expose to the next time step.
- ***g*** : self-recurrent which is equal to standard RNN
- c_t: internal memory of the memory cell
- s_t: hidden state
- y : final output

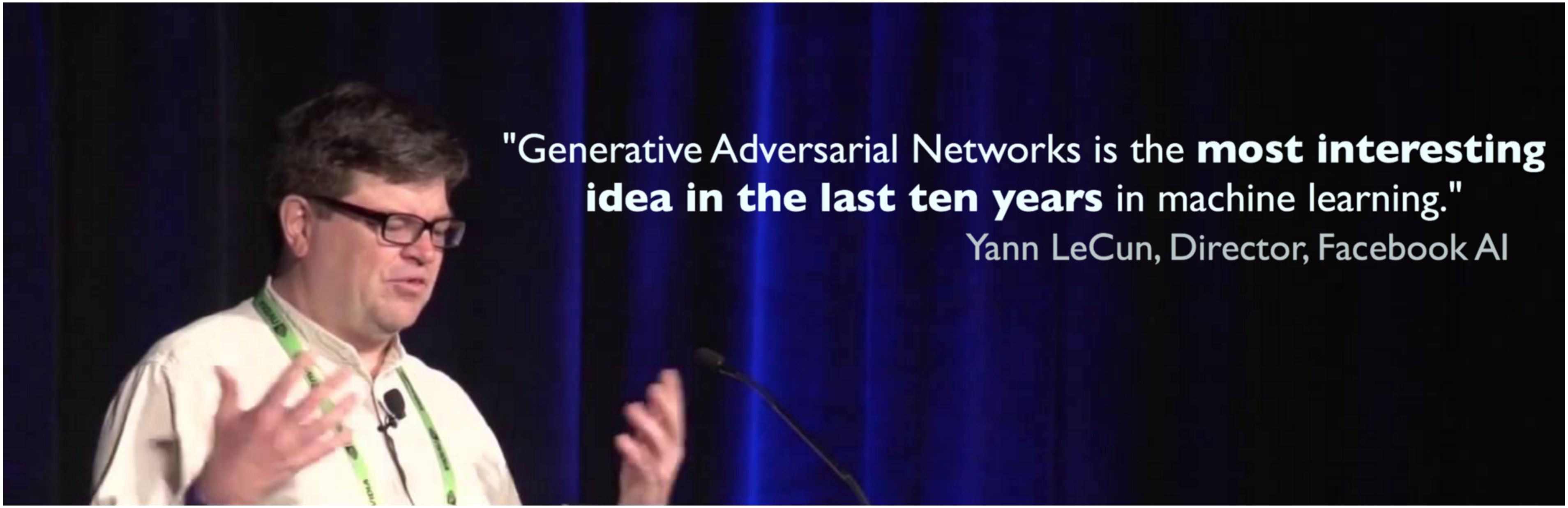
- $i = \sigma\left(x_t U^i + s_{t-1} W^i\right)$
- $f = \sigma\left(x_t U^f + s_{t-1} W^f\right)$
- $o = \sigma(x_t U^o + s_{t-1} W^o)$
- $g = \tanh(x_t U^g + s_{t-1} W^g)$
- $c_t = c_{t-1} \circ f + g \circ i$
- $s_t = \tanh(c_t) \circ o$
- $y = softmax(V s_t)$

Long Short Term Memory (LSTM)



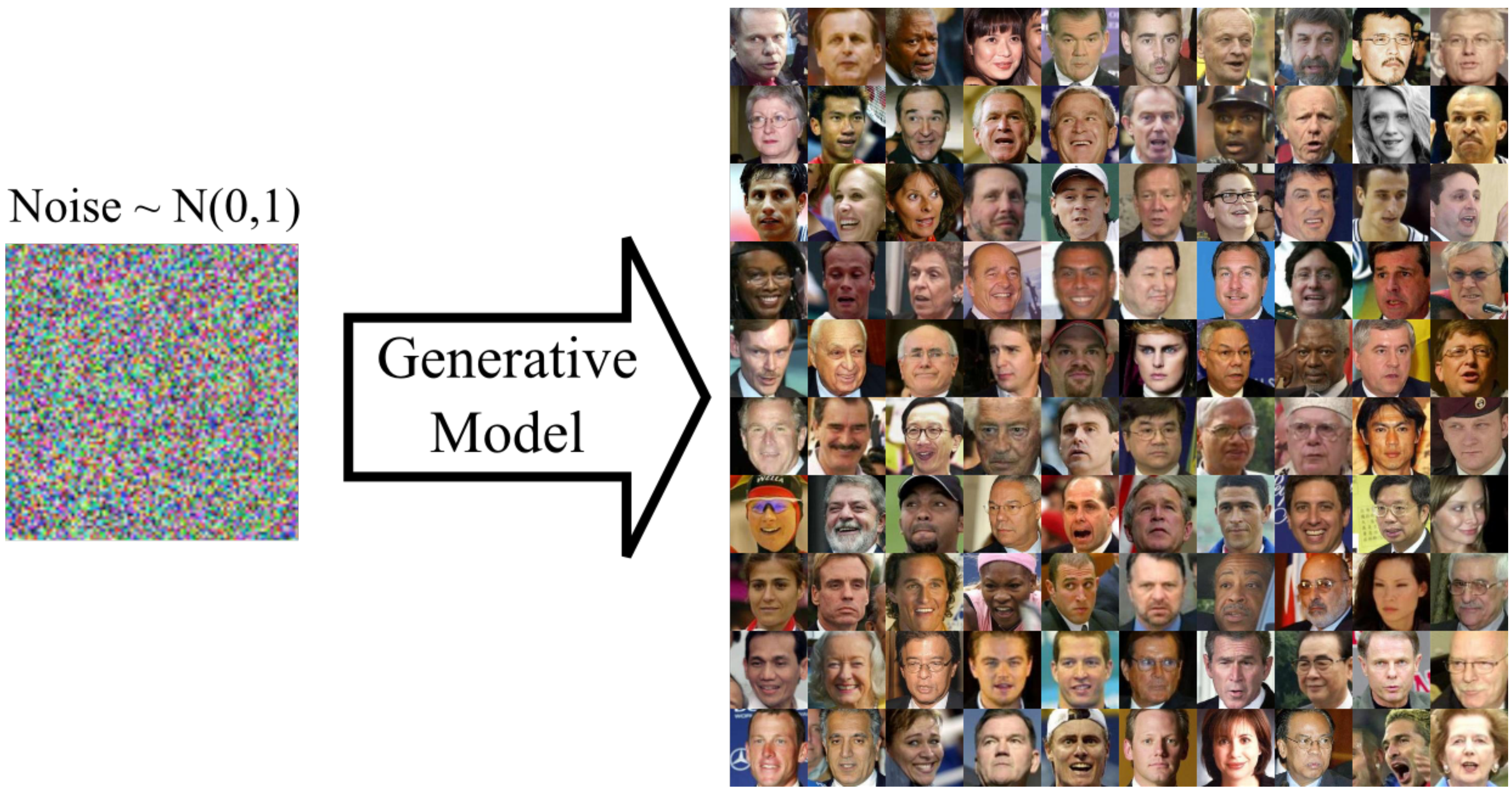
The future of Deep Learning

New models


"Generative Adversarial Networks is the most interesting idea in the last ten years in machine learning."
Yann LeCun, Director, Facebook AI

Deep Generative Models

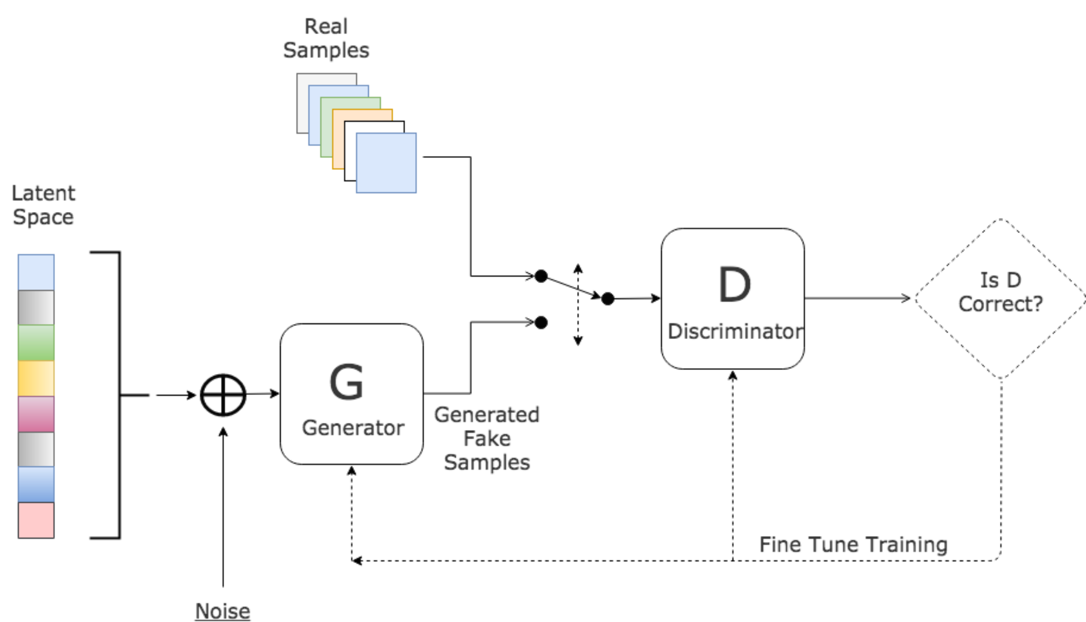
- Lots of research on generative models to create probabilistic models of training data with ability to generate new images, sentences, etc.



Deep Generative Models

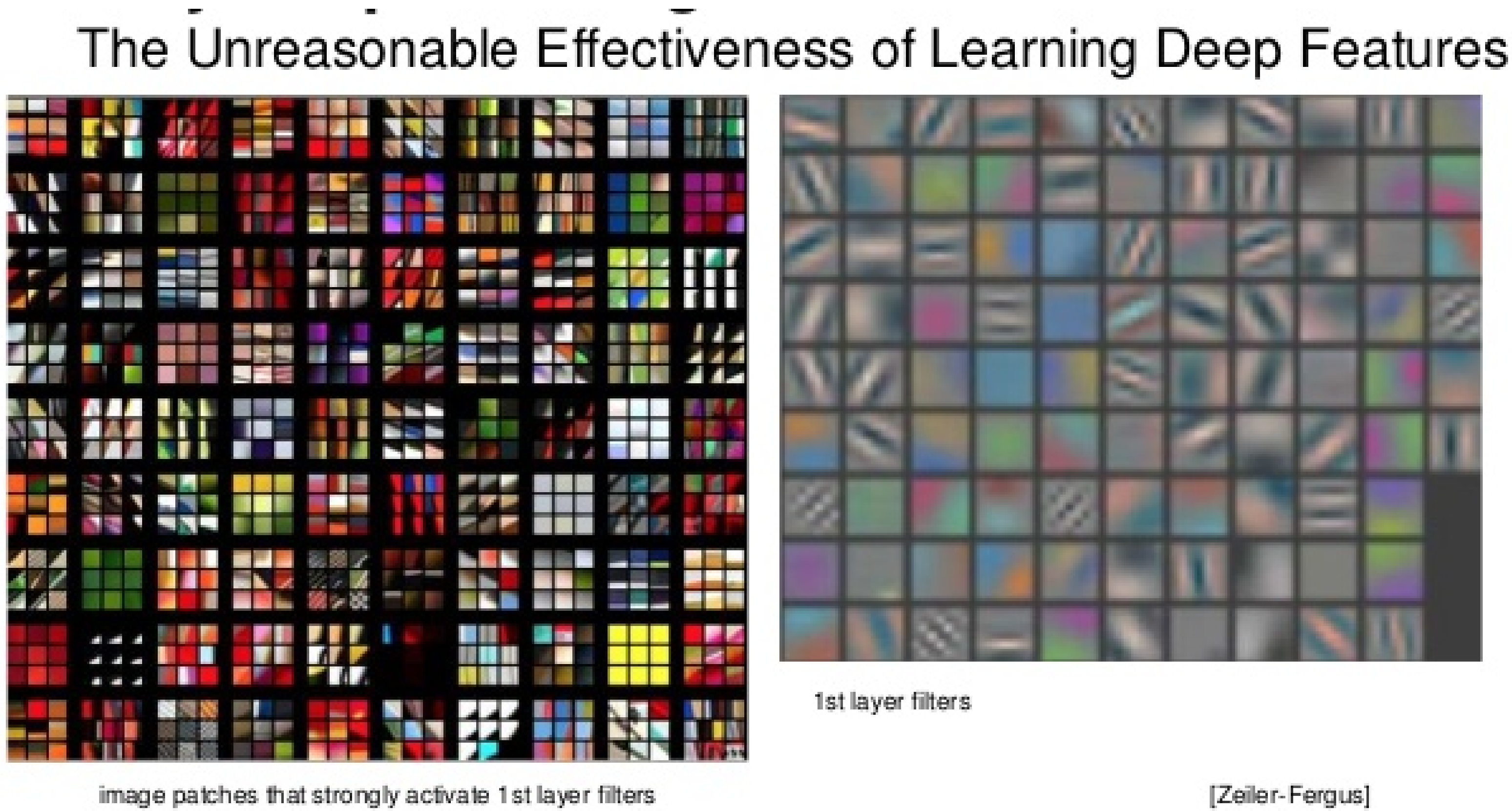
Generative Adversarial Networks (GANs)

- Generator net produces samples *x* close to training samples
- Discriminator net (adversary) must differentiate between samples from the generative net and the training set
- Use error feedback to improve task of both nets, until discriminator can no longer distinguish, then can discard discriminator net – increasingly difficult for humans to distinguish



Open Issues

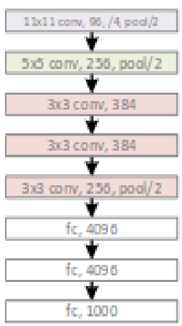
Why it works?



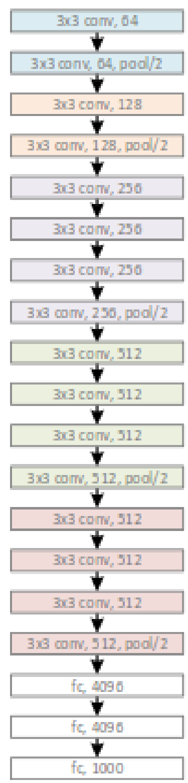
Open Issues

Scale: larger and larger nets...

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

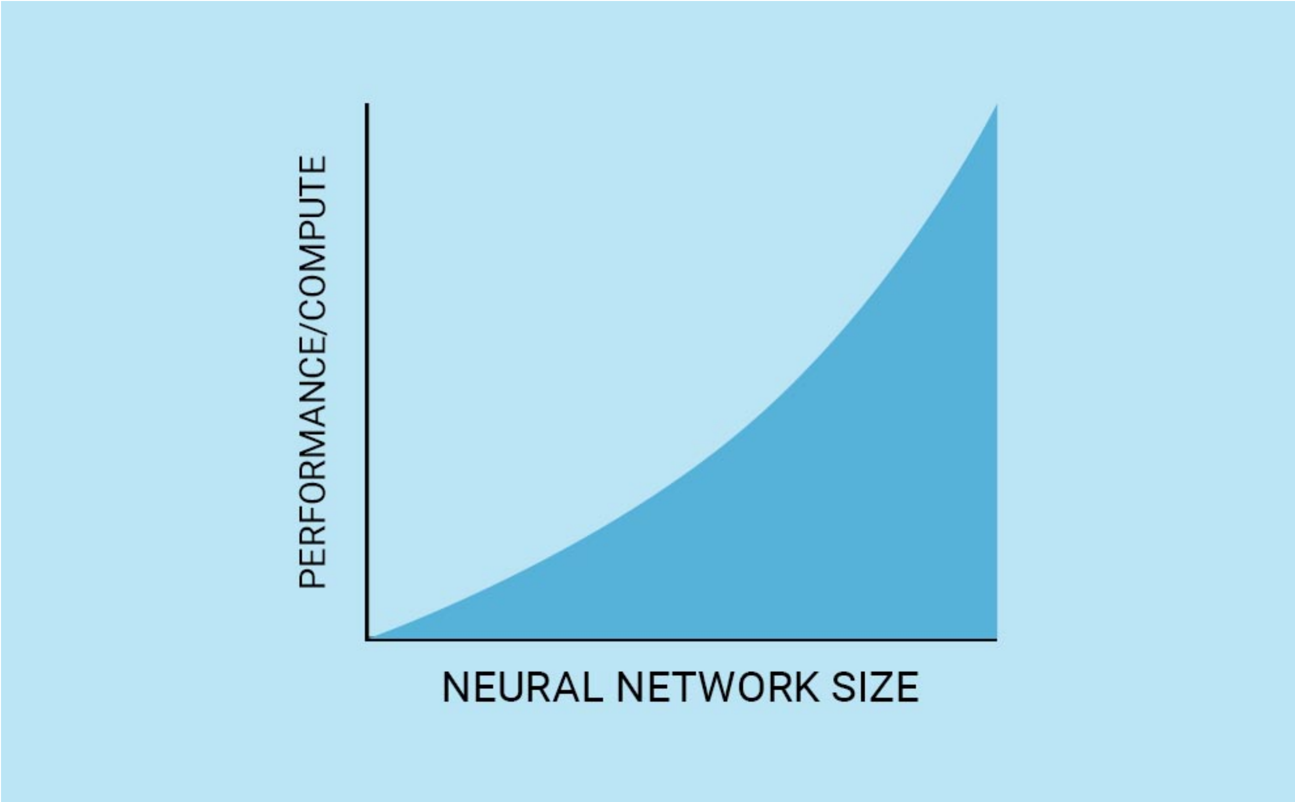


GoogleNet, 22 layers
(ILSVRC 2014)

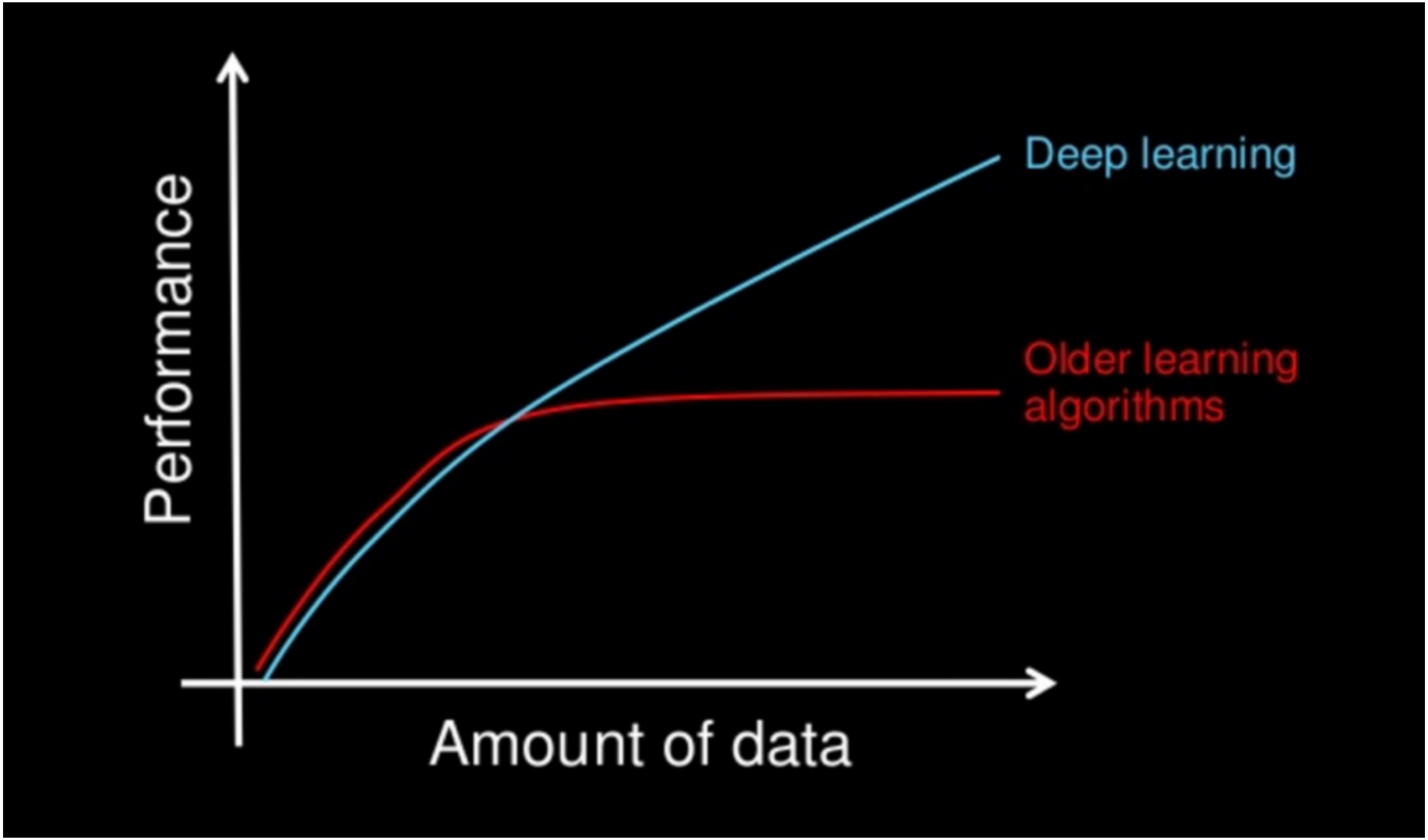
ResNet, 152 layers
(ILSVRC 2015)

Open Issues

Scale: how to stop this???



Deep Learning & Data



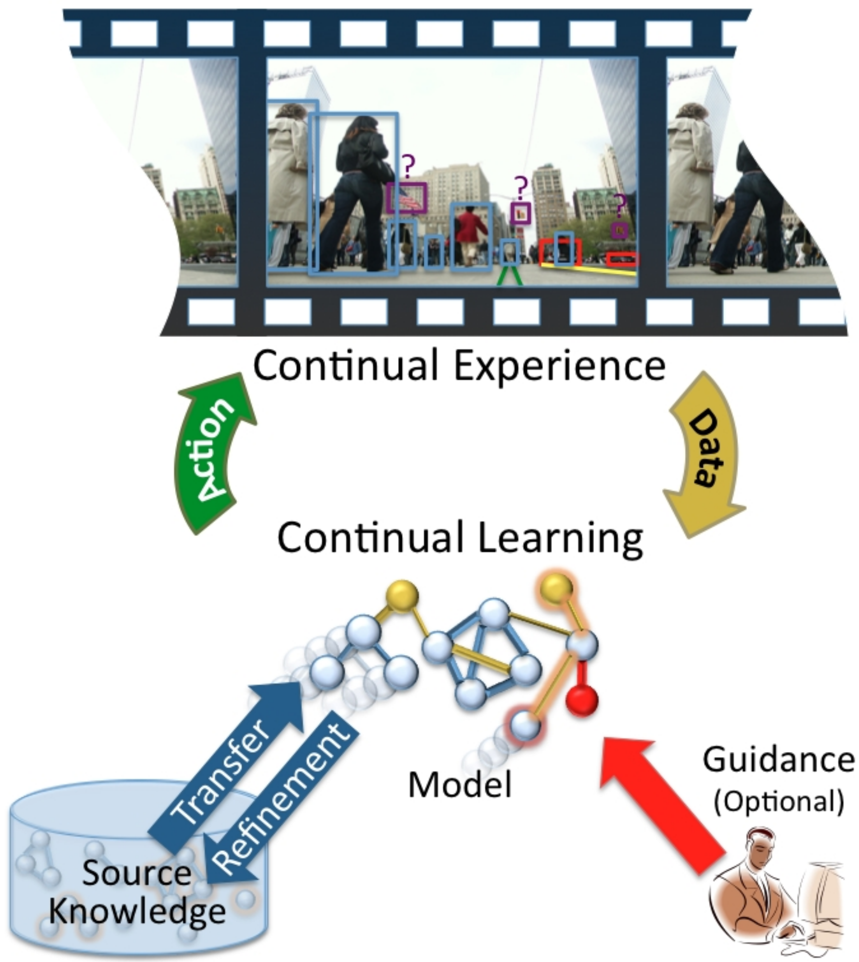
https://www.linkedin.com/pulse/how-artificial-intelligence-revolutionizing-finance-del-toro-barba

Open Issues

Unsupervised Learning

Open Issues

Life-long Learning



[Eaton]

Summary

- Impressive results
- Works well in structured/Markovian spaces - CNNs, etc.
- Much recent excitement, still much to be discovered
- More work needed to understand how and why deep learning works so well – How deep should we go?
- Potential for significant improvements