Costa Rica Big data School: Introduction to Deep Learning

Weijia Xu
Research Scientist, Group Manager
Data Mining & Statistics
Texas Advanced Computing Center
The University of Texas at Austin

Dec 2018

Why deep learning?

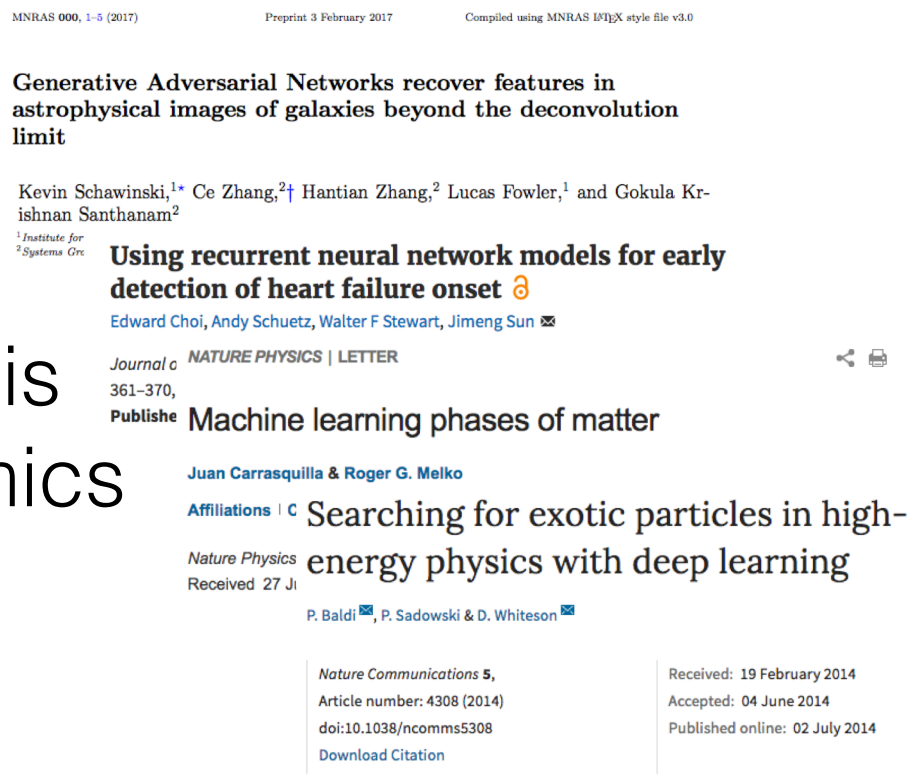
- Provide new solutions to many existing problems:
  - Image classification*
    - →Face recognition
    - →Object recognition
  - Speech recognition (Circa
    - → Google voice input, e
  - Natural language processing
  - Many domain science fields…

Dataset	Best result of others [%]	MCDNN [%]	Relative improv. [%]
MNIST	0.39	0.23	41
NIST SD 19	see Table 4	see Table 4	30-80
HWDB1.0 on.	7.61	5.61	26
HWDB1.0 off.	10.01	6.5	35
CIFAR10	18.50	11.21	39
traffic signs	1.69	0.54	72
NORB	5.00	2.70	46

* http://people.idsia.ch/~juergen/cvpr2012.pdf

Applications of Deep Learning in Science

- Many scientists are exploring and adopting deep learning as the data science methodology to tackle their domain research challeng
  - Astronomy
  - Drug discovery
  - Disease diagnosis
  - Molecular dynamics
  - Neurology
  - Particle physics
  - Social science



MNRAS 000, 1–5 (2017) Preprint 3 February 2017 Compiled using MNRAS LATEX style file v3.0

Generative Adversarial Networks recover features in astrophysical images of galaxies beyond the deconvolution limit

Kevin Schawinski,[1]* Ce Zhang,[2]† Hantian Zhang,[2] Lucas Fowler,[1] and Gokula Krishnan Santhanam[2]

[1] Institute for
[2] Systems Gro

Using recurrent neural network models for early detection of heart failure onset

Edward Choi, Andy Schuetz, Walter F Stewart, Jimeng Sun

Journal o
361–370,
Publishe

NATURE PHYSICS | LETTER

Machine learning phases of matter

Juan Carrasquilla & Roger G. Melko

Affiliations | C

Nature Physics
Received 27 Ju

Searching for exotic particles in high-energy physics with deep learning

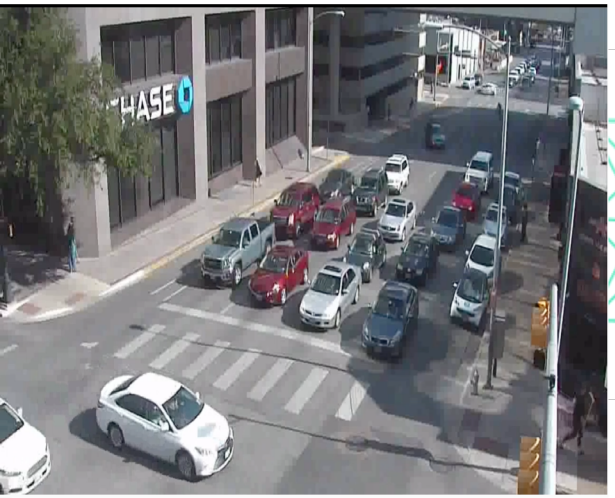
P. Baldi, P. Sadowski & D. Whiteson

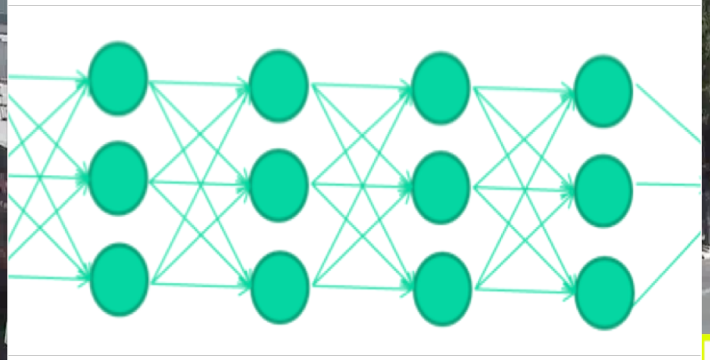
Nature Communications 5,
Article number: 4308 (2014)
doi:10.1038/ncomms5308
Download Citation

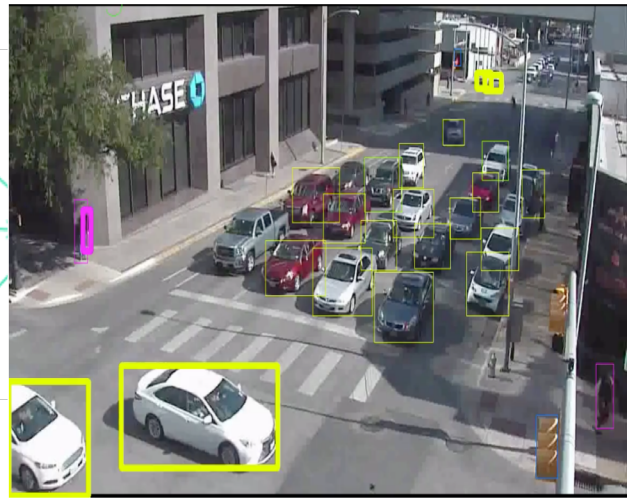
Received: 19 February 2014
Accepted: 04 June 2014
Published online: 02 July 2014

What is the deep learning?

- A short answer:
  - A neural network with multiple layers between input (raw data) and output (prediction).



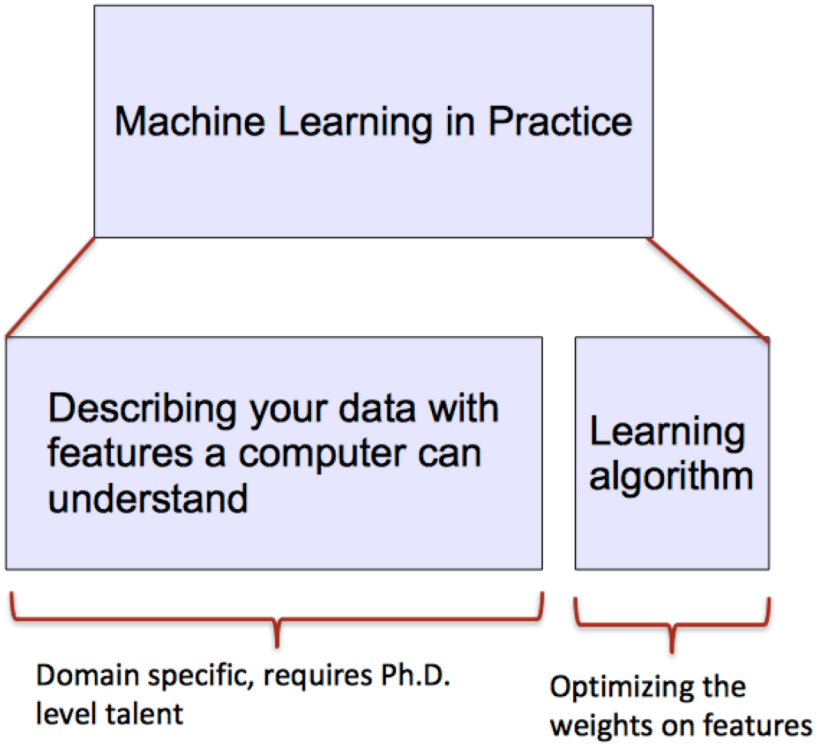




Machine Learning vs. Deep Learning

Most machine learning methods work well because of human-designed representations and input features

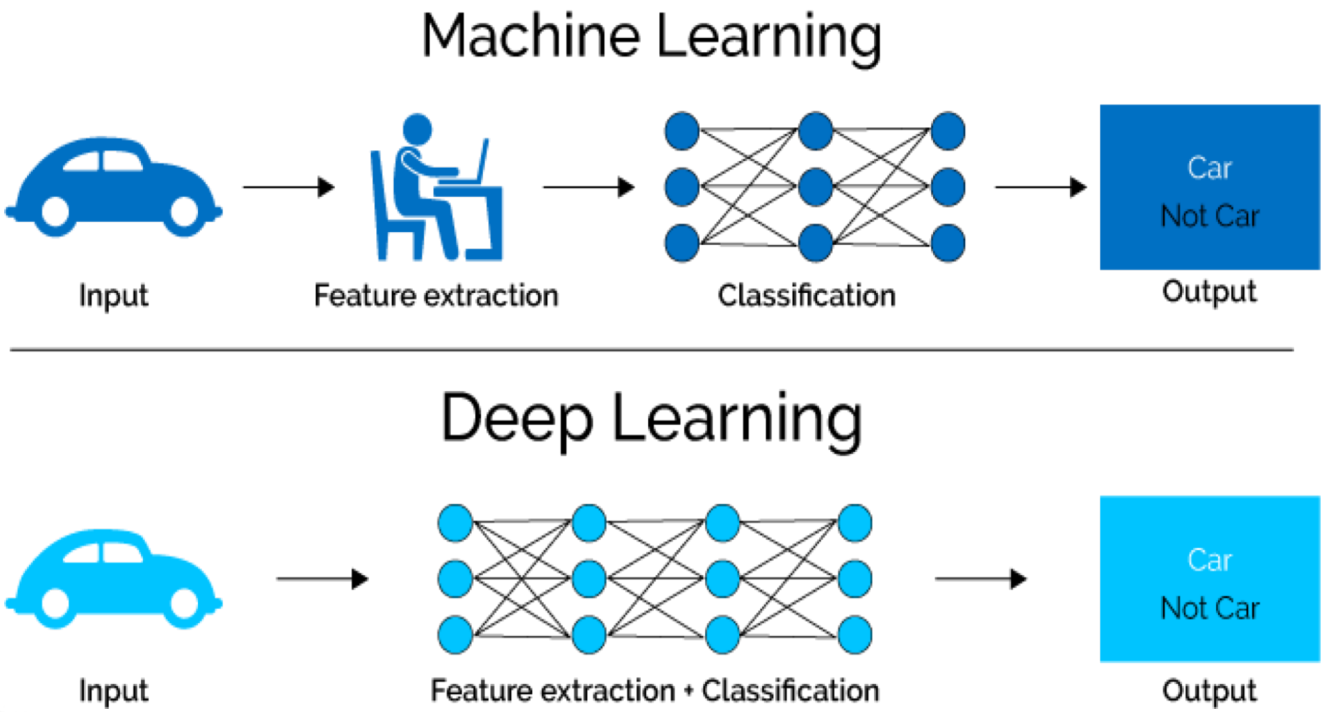
ML becomes just optimizing weights to best make a final prediction



Feature	NER
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

ML vs. DL

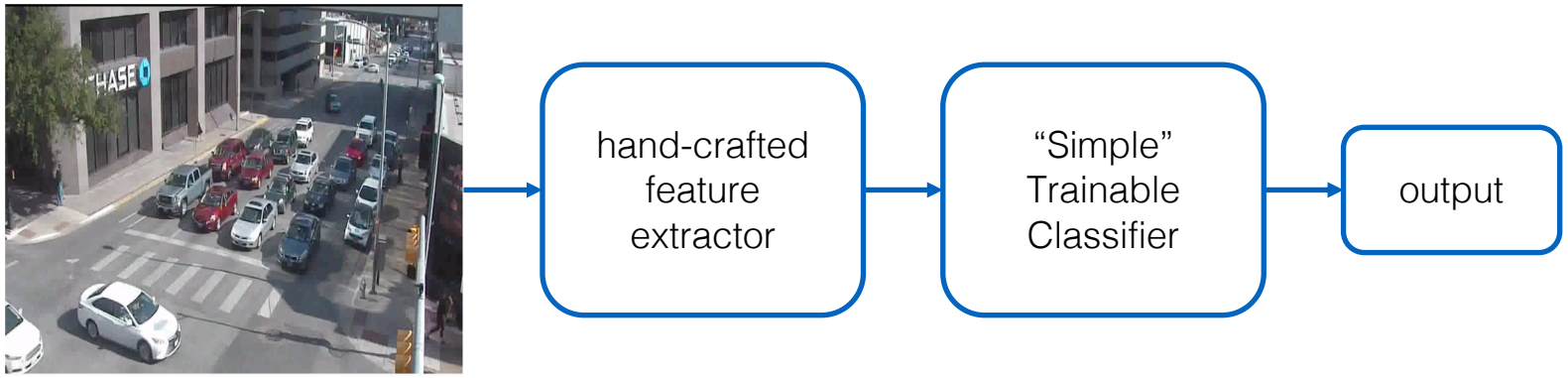
Deep learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers
If you provide the system tons of information, it begins to understand it and respond in useful ways.



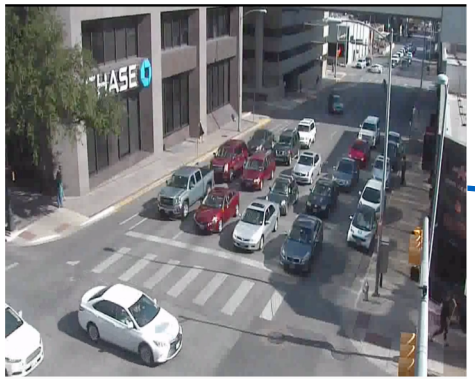
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

What is the deep learning?

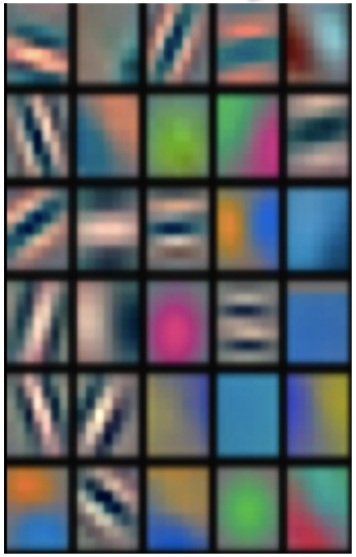
- Traditional image classification

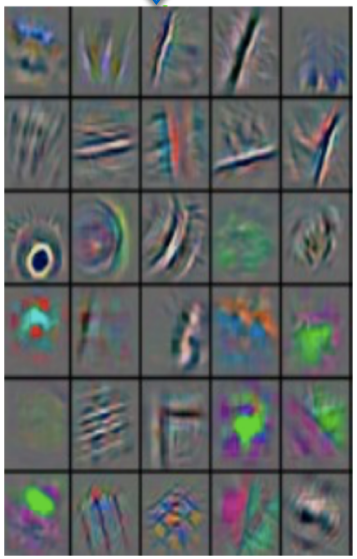


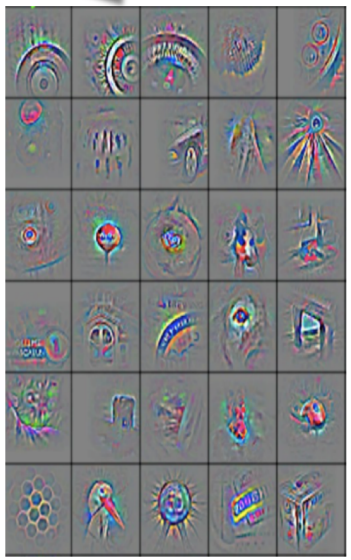
- Deep learning uses a "deep structure" to learn rich hierarchical features.
  - a.k.a representation learning
  - Features are automatically extracted through multiple stages



Low-level features → Mid-level features → High-level features → Trainable classifier → output





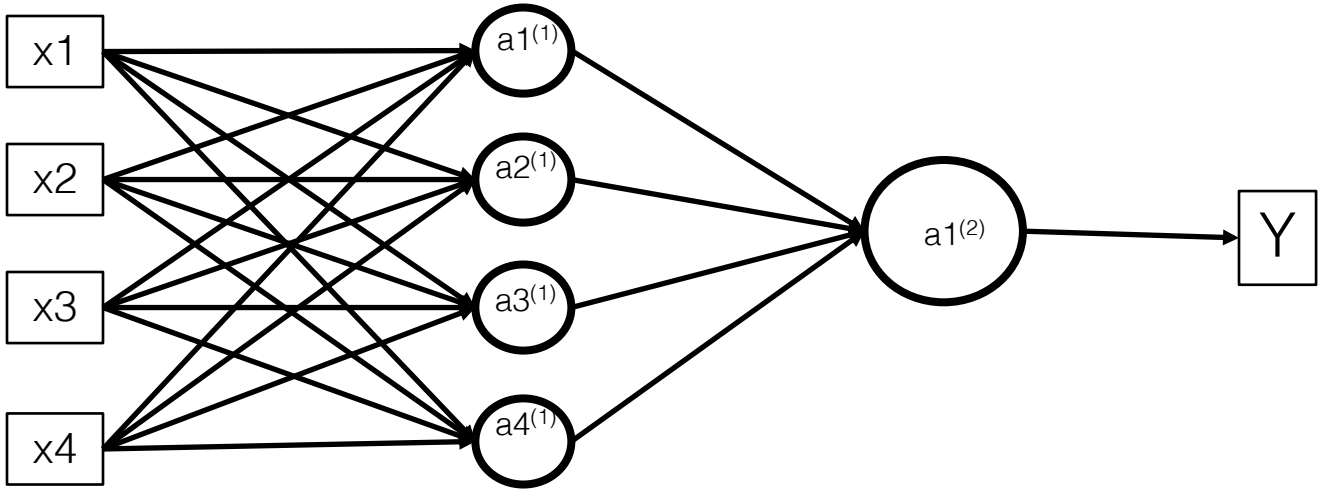


* feature visualization of convolutional net trained on ImageNet (Zeiler and Fergus, 2013)

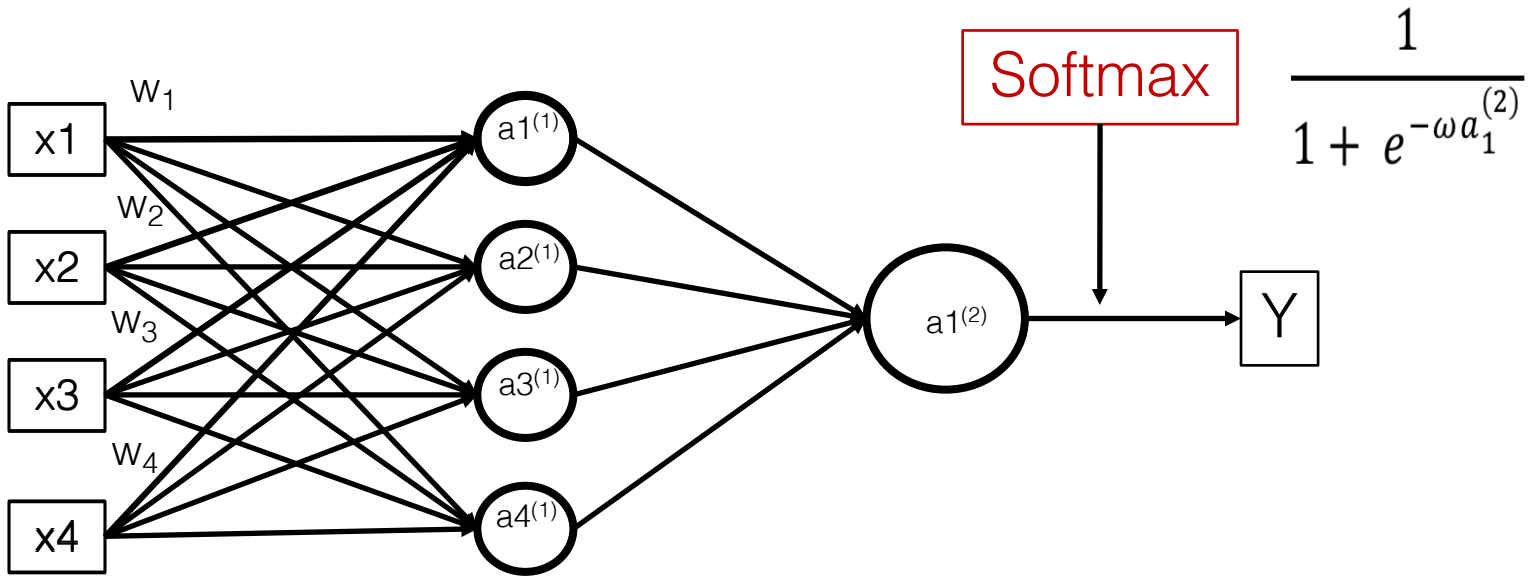
Neuron Network Basics

A Simple Neural Network

- An Artificial Neural Network is an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.



A Simple Neural Network



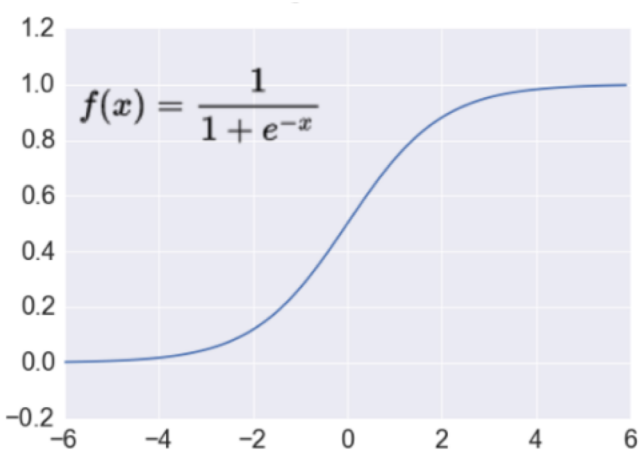
$$a_1^{(1)} = f(\ w_1{*}x_1 + w_2{*}x_2 + w_3{*}x_3 + w_4{*}x_4)$$

$f()$ is **activation function**: Relu or sigmoid

$$Relu: \max(0, x)$$

$$a_1^{(1)} = \max(0,\ w_1{*}x_1 + w_2{*}x_2 + w_3{*}x_3 + w_4{*}x_4)$$

Activation: Sigmoid



http://adilmoujahid.com/images/activation.png

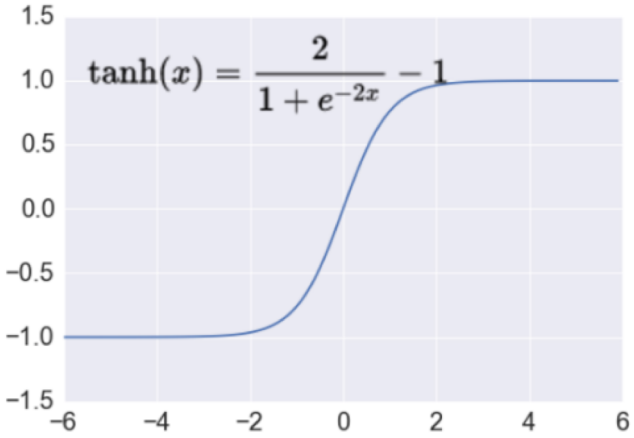
Takes a real-valued number and “squashes” it into range between 0 and 1.

$R^n \rightarrow [0, 1]$

+ Nice interpretation as the firing rate of a neuron
  - 0 = not firing at all
  - 1 = fully firing

- Sigmoid neurons saturate (just 0 and 1) and kill gradients, thus NN will barely learn

Activation: Tanh



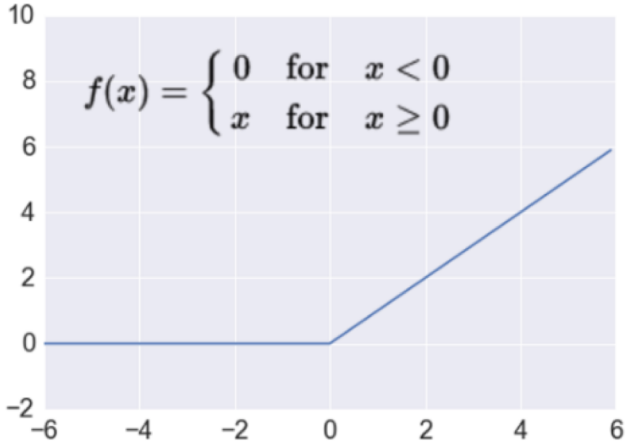
http://adilmoujahid.com/images/activation.png

Takes a real-valued number and "squashes" it into range between -1 and 1.

$R^n \rightarrow [-1, 1]$

- Like sigmoid, tanh neurons saturate
- Unlike sigmoid, output is zero-centered
- Tanh is a scaled sigmoid:

Activation: ReLU



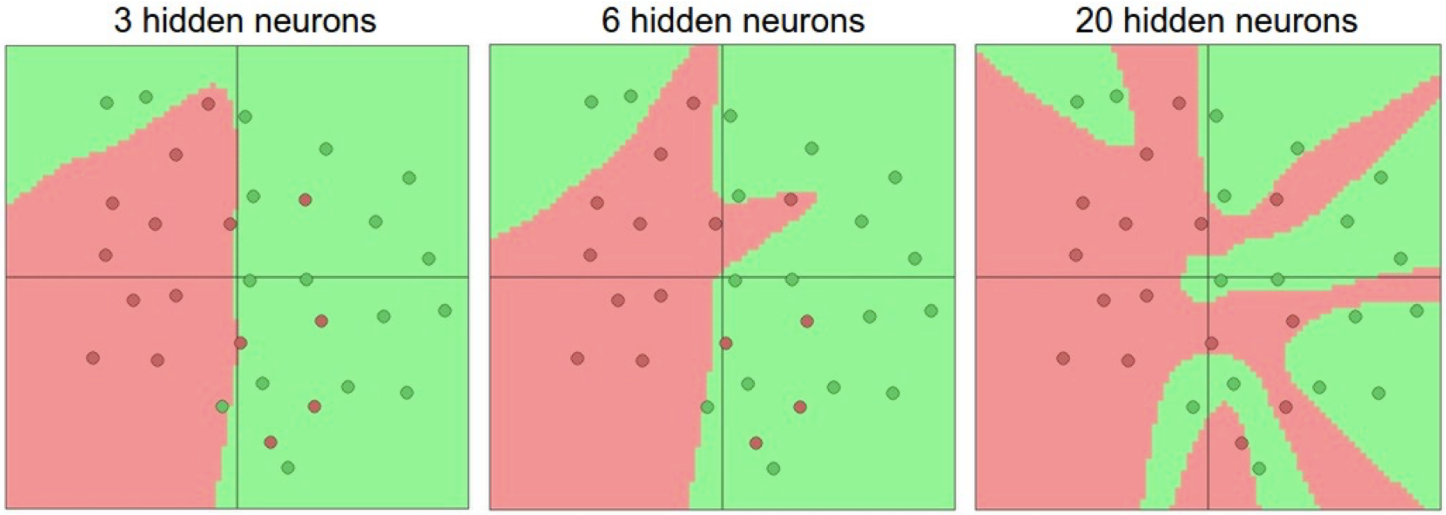
Takes a real-valued number and thresholds it at zero

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

- Trains much faster
  - accelerates the convergence of SGD
  - due to linear, non-saturating form
- Less expensive operations
  - compared to sigmoid/tanh (exponentials etc.)
  - implemented by simply thresholding a matrix at zero
- More expressive
- Prevents the gradient vanishing problem

Why Activation functions

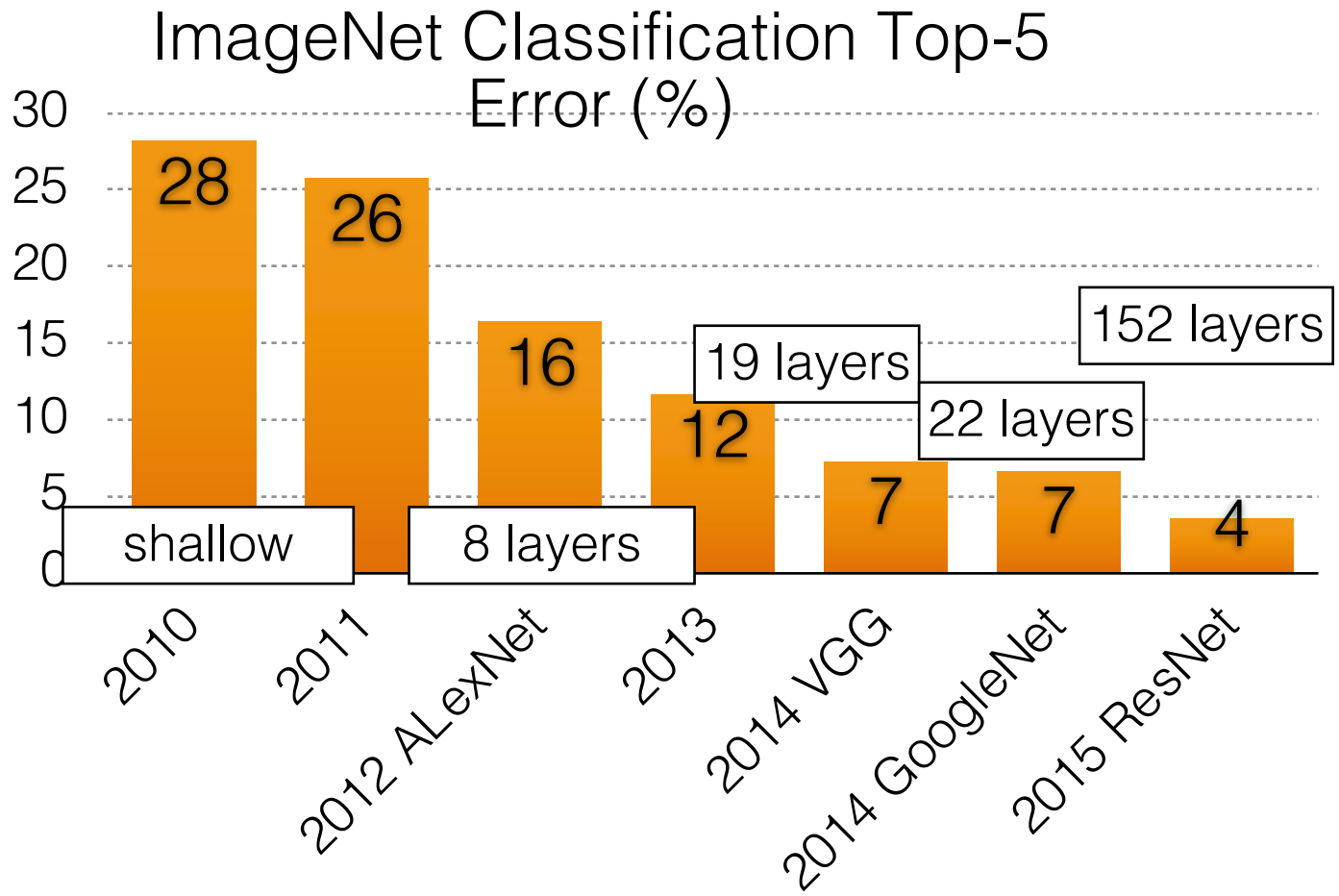
Non-linearities needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function



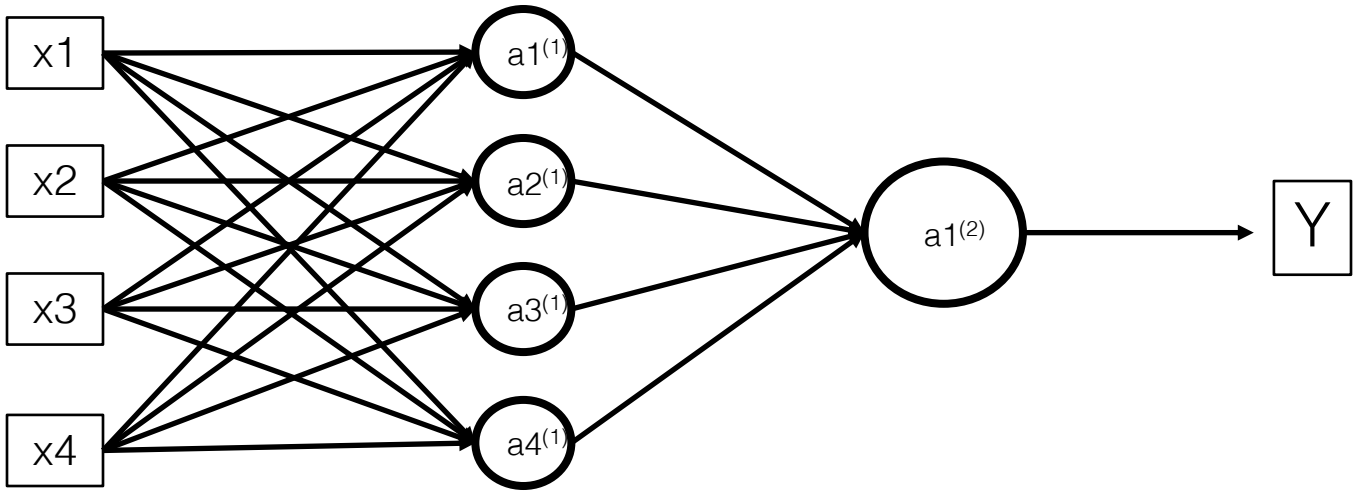
More layers and neurons can approximate more complex functions

http://cs231n.github.io/assets/nn1/layer_sizes.jpeg

- Image Recognition



Number of Parameters



Input | Hidden Layers | Output

Weights: $4 * 4 + 4 * 1 + 1$

biases: $4 + 1$

Training of Neuron Networks

- General Process
  - Initialize random weight
  - Forward labeled data through network to get prediction
  - Computer errors between prediction and label
  - Back-propagate the errors from output to hidden layers
  - Update network parameters
  - Repeat until error rate meets threshold.

- Generate error signal to measure difference between predictions and target values and guide parameter update.

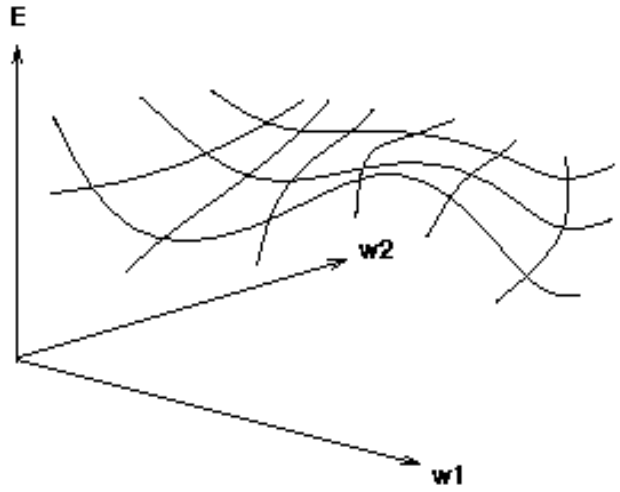
- An iterative process

Back-Propagation

- Gradient descent
  - Consider error and N weights are in a hyperspace with N+1 dimensions.
  - Try to find low positions in error surface.

- Update weight:
  - $W^{1,0} \leftarrow W^{1,0} - \lambda * \partial E / \partial W^{1,0}$

- Backwards from
  - output to hidden layer n
  - to hidden layer n-1
  - …
  - to input layer



Hyper Parameters in Practice

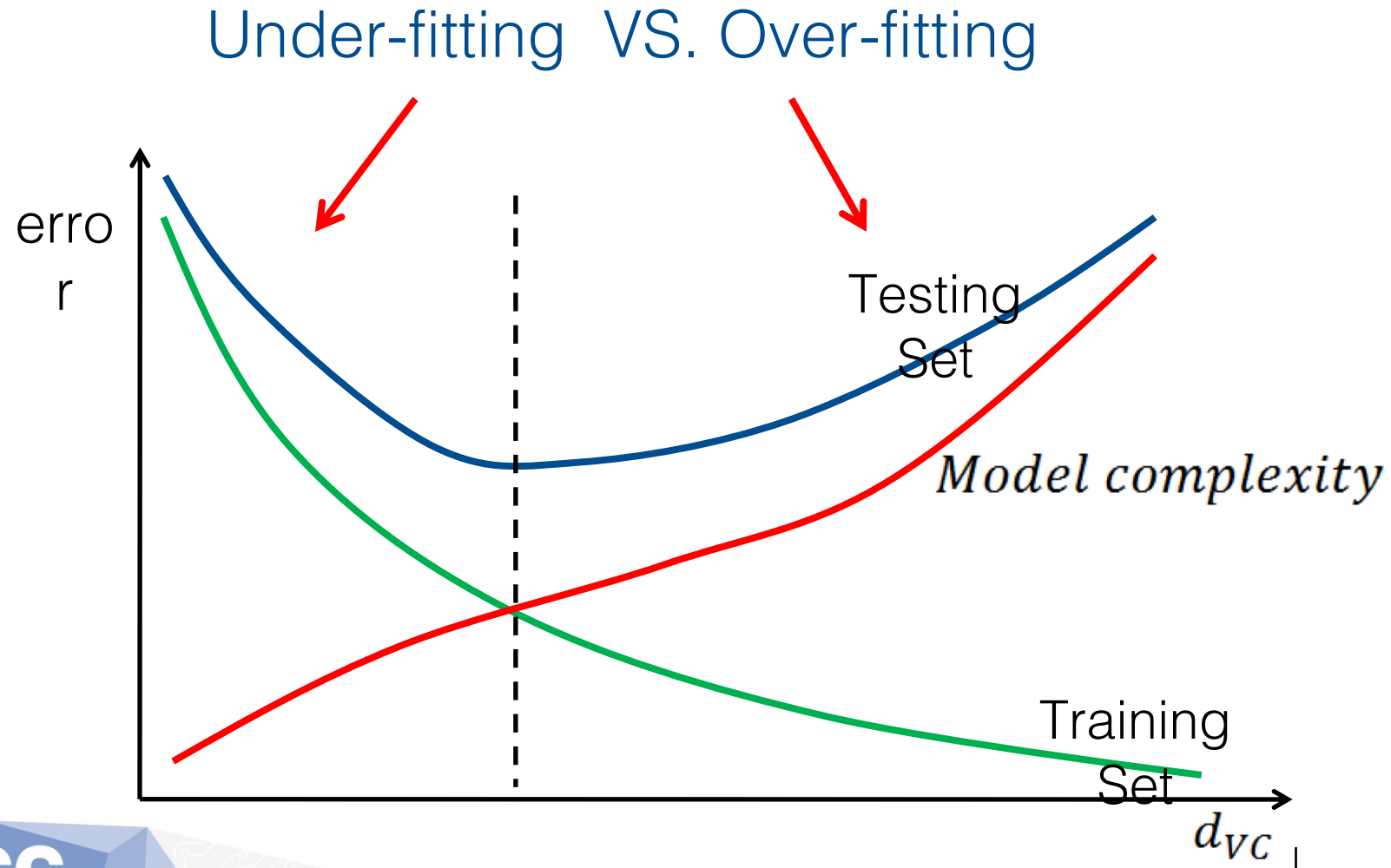
- Learning rate
  - How fast the gradient is updated along the gradient direction.
  - Small value, slow conservative learning

- Mini-batch size
  - Input samples are learned in “batch”
  - Parameters are updated based on average loss

- Epoch
  - The time by which each sample is visited.
  - E.g. 1.2 M images, 512 mini-batch size, an epoch requires ~2400 iteration

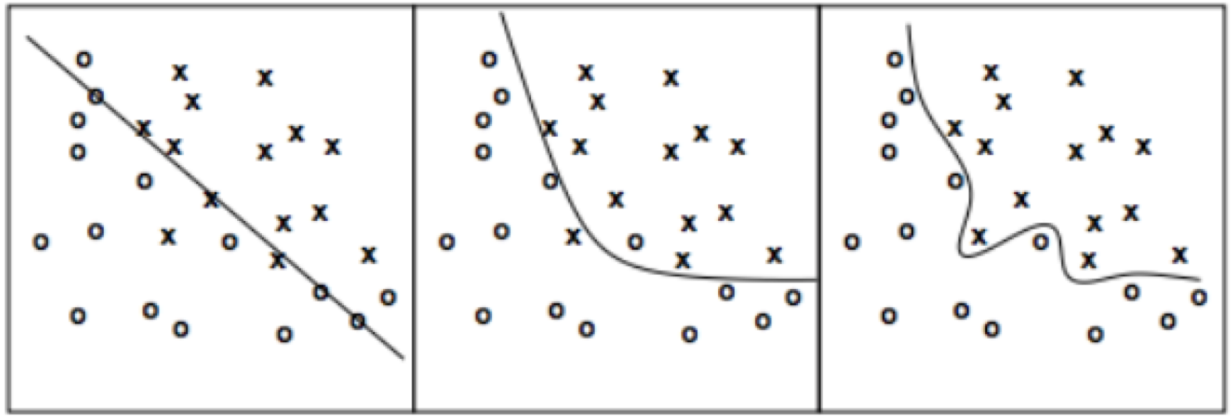
- Early stopping
  - Stop if no improving after certain epochs

Learning Performance



Regularization

- Goal is to find model with low training and validation errors.



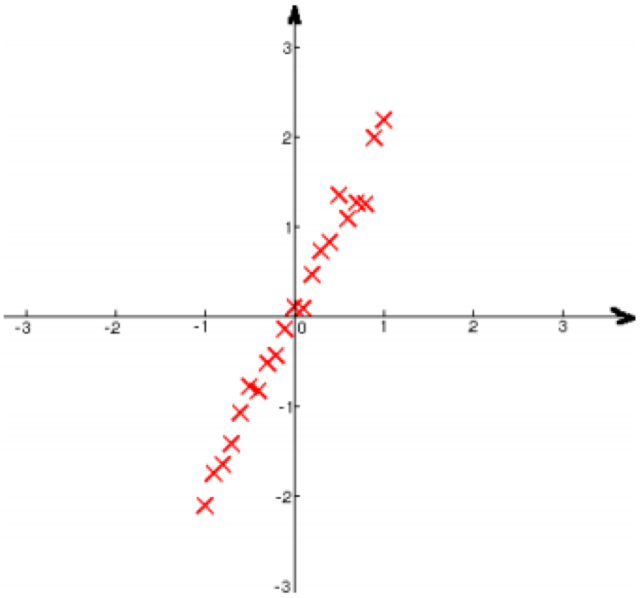
- Dropout rate
  - Randomly drop units during training with fixed probability.

- Weight decay
  - Penalizes big weights
  - How dominant regularization is during gradient computation.

Auto-encoder

A Simple Problem

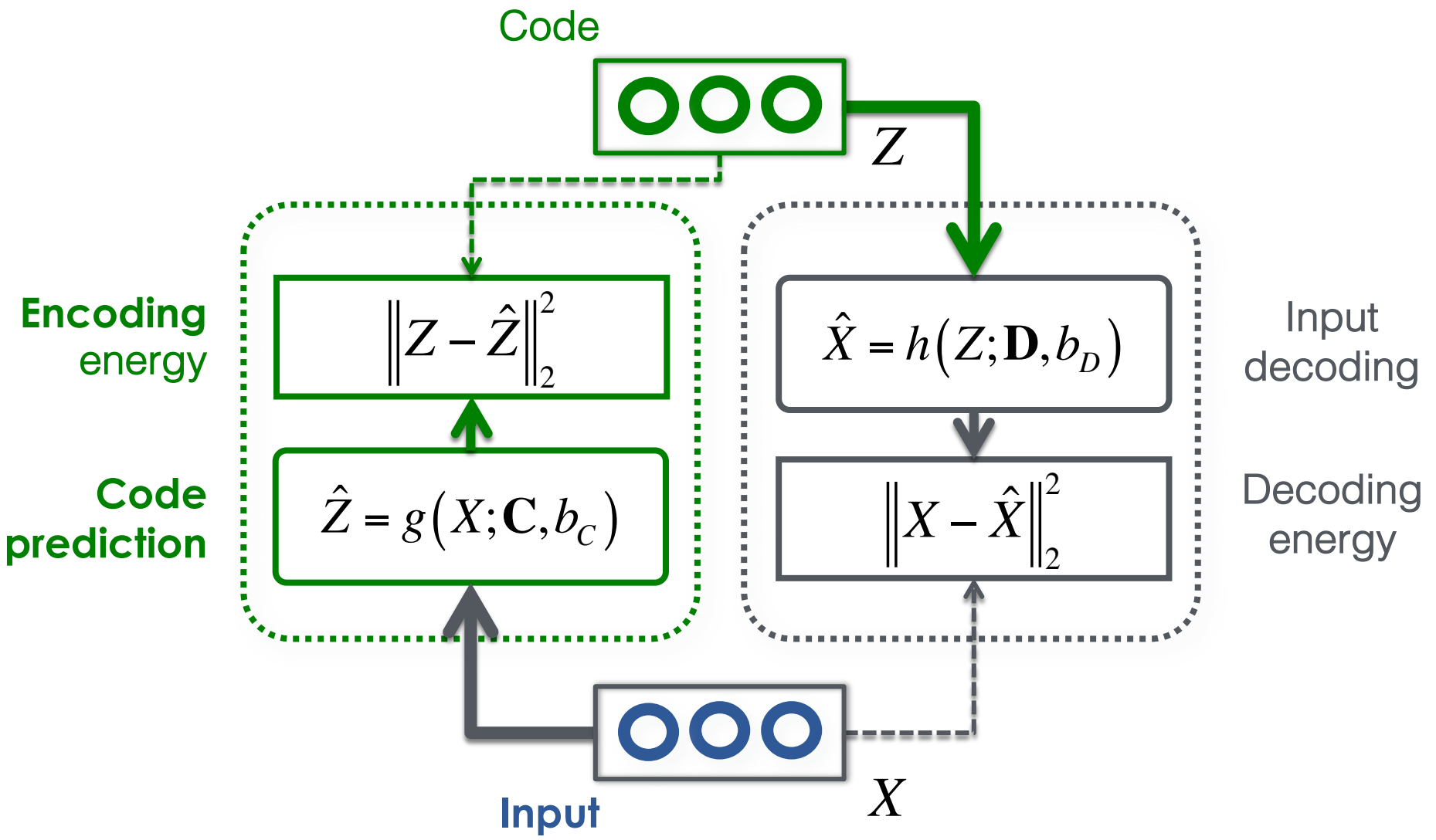
- Considering a set of two dimensional data points.



- What if we need save the storage?
  - Y=2X ?

- We could just retain values of X and compute Y.

Autoencoder



Autoencoder loss function

For one sample t

$$L\left(X(t),Z(t);\mathbf{W}\right)=\alpha\left\|Z(t)-g\left(X(t);\mathbf{C},b_C\right)\right\|_2^2+\left\|X(t)-h\left(Z(t);\mathbf{D},b_D\right)\right\|_2^2$$

Encoding energy

Decoding energy

coefficient of the encoder error

For all T samples

$$L\left(\mathbf{X},\mathbf{Z};\mathbf{W}\right)=\sum_{t=1}^{T}\alpha\left\|Z(t)-g\left(X(t);\mathbf{C},b_C\right)\right\|_2^2+\sum_{t=1}^{T}\left\|X(t)-h\left(Z(t);\mathbf{D},b_D\right)\right\|_2^2$$

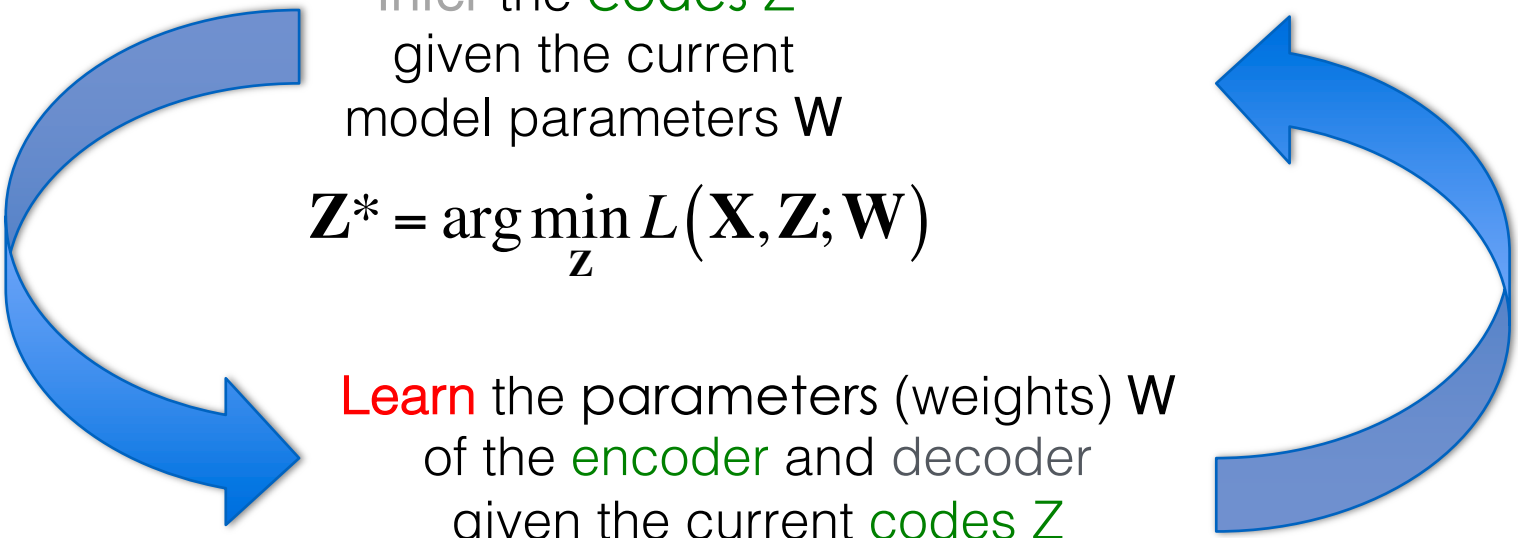
Encoding energy

Decoding energy

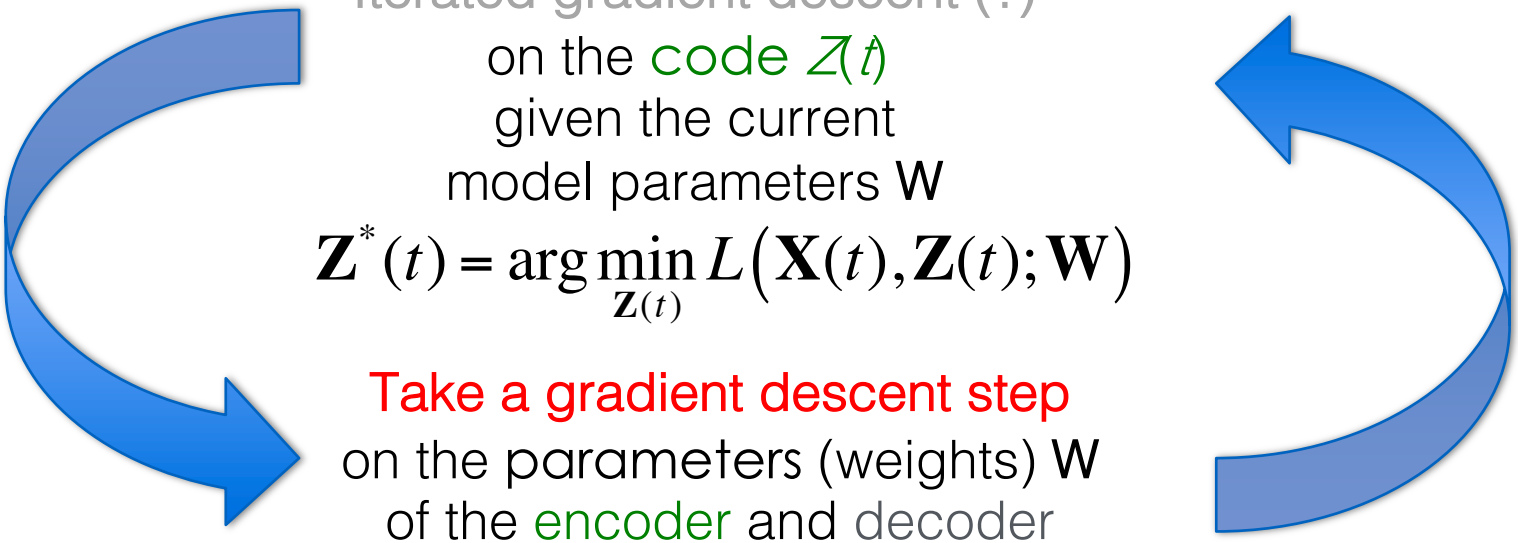
How do we get the codes Z?

We note W={C, b_C, D, b_D}

Learning and inference in auto-encoders



Learning and inference: stochastic gradient descent



Iterated gradient descent (?)
on the code $Z(t)$
given the current
model parameters W

$$\mathbf{Z}^*(t) = \arg\min_{\mathbf{Z}(t)} L\left(\mathbf{X}(t), \mathbf{Z}(t); \mathbf{W}\right)$$

Take a gradient descent step
on the parameters (weights) W
of the encoder and decoder
given the current codes Z

$$L\left(\mathbf{X}(t), \mathbf{Z}(t); \mathbf{W}^*\right) < L\left(\mathbf{X}(t), \mathbf{Z}(t); \mathbf{W}\right)$$

Autoencoder example

- Example notebook
  - /work/00791/xwj/DMS/ML/AE.ipynb

- Using Mnist data set.
  - 28x28 image of digit
  - 60K training and 10K test images
  - Labels are available but not used.

- Other data could be loaded and adapted here.

Single Dense Layer Example

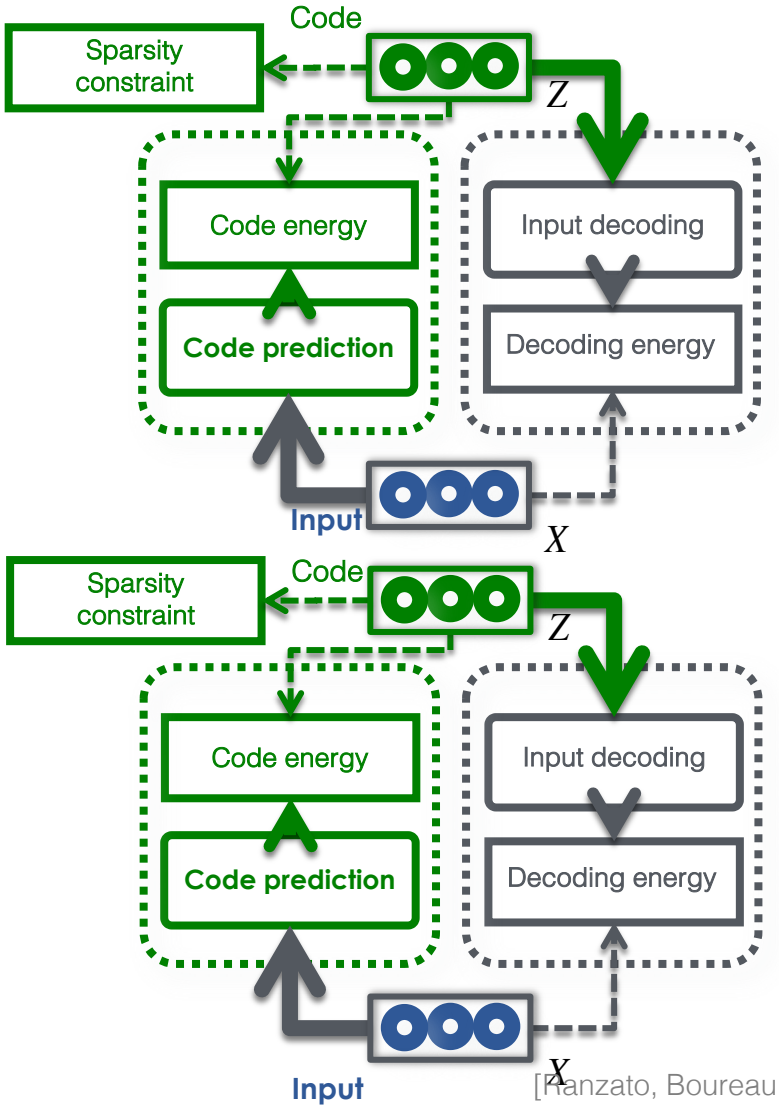
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 784)               0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                25120
_________________________________________________________________
dense_6 (Dense)              (None, 784)               25872
=================================================================
Total params: 50,992
Trainable params: 50,992
Non-trainable params: 0
```

- This example maps input image to a 32d vector.
- ~ 24.5 times compression from 784d vector.
- Sparsity constraint can be introduced through activity regularizer

- Try play with different coding length, and training parameters
  - How those affect the results

- How this method differs from other dimensionality reduction method like PCA?

Deep Autoencoders



- Instead of just a single layer, we can stacking several layers together to form deep structure.

[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

Deep Autoencoders

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	(None, 784)	0
dense_13 (Dense)	(None, 128)	100480
dense_14 (Dense)	(None, 64)	8256
dense_15 (Dense)	(None, 32)	2080
dense_16 (Dense)	(None, 64)	2112
dense_17 (Dense)	(None, 128)	8320
dense_18 (Dense)	(None, 784)	101136

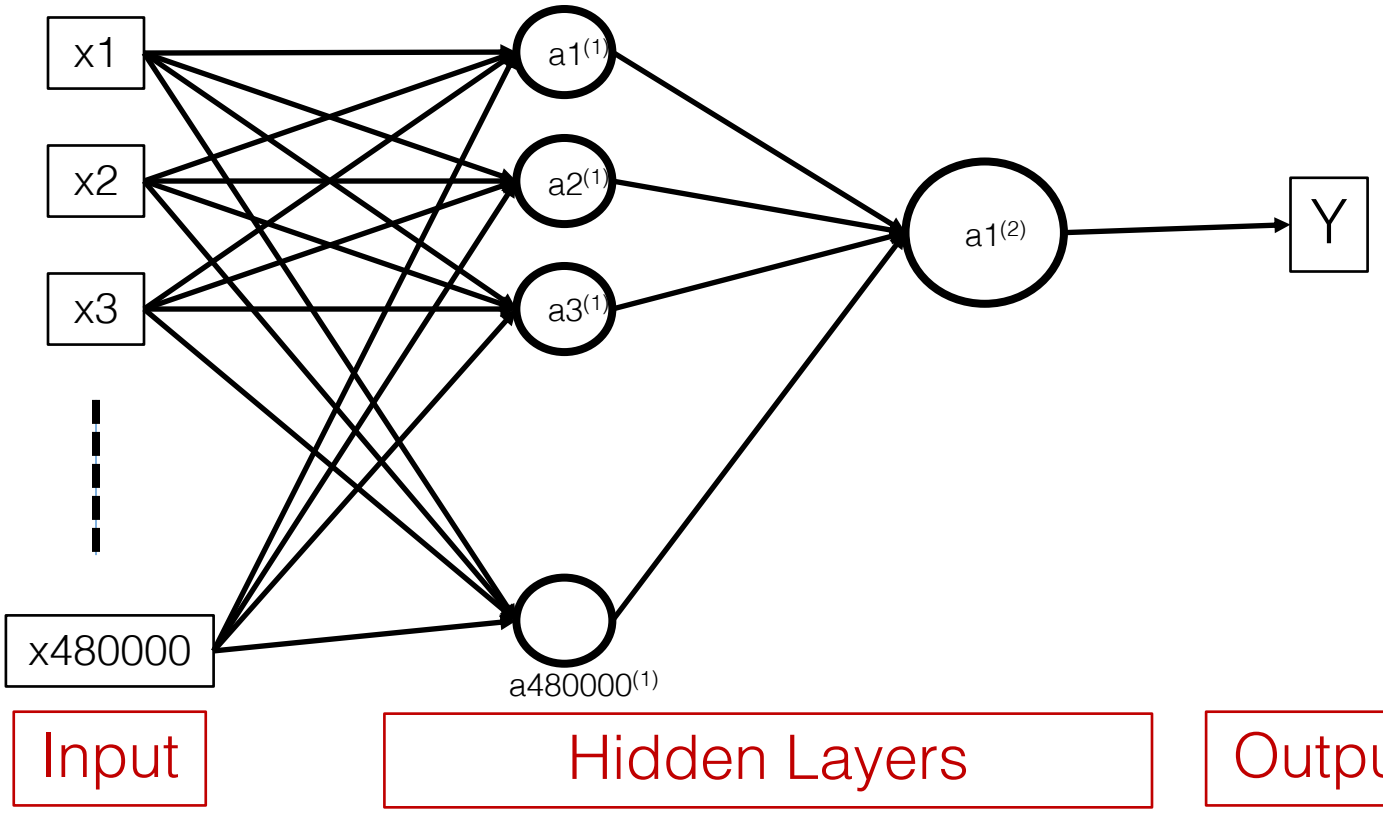
Total params: 222,384
Trainable params: 222,384
Non-trainable params: 0

- What's pros and cons?
- When this might be useful?

Convolutional Neural Network

If the input is an Image?

400 X 400 X 3



Number of Parameters

480000*480000 + 480000 +1 = **approximately 230 Billion !!!**

480000*1000 + 1000 +1 = approximately **480 million !!!**

Convolutional Layers

- Filter  $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

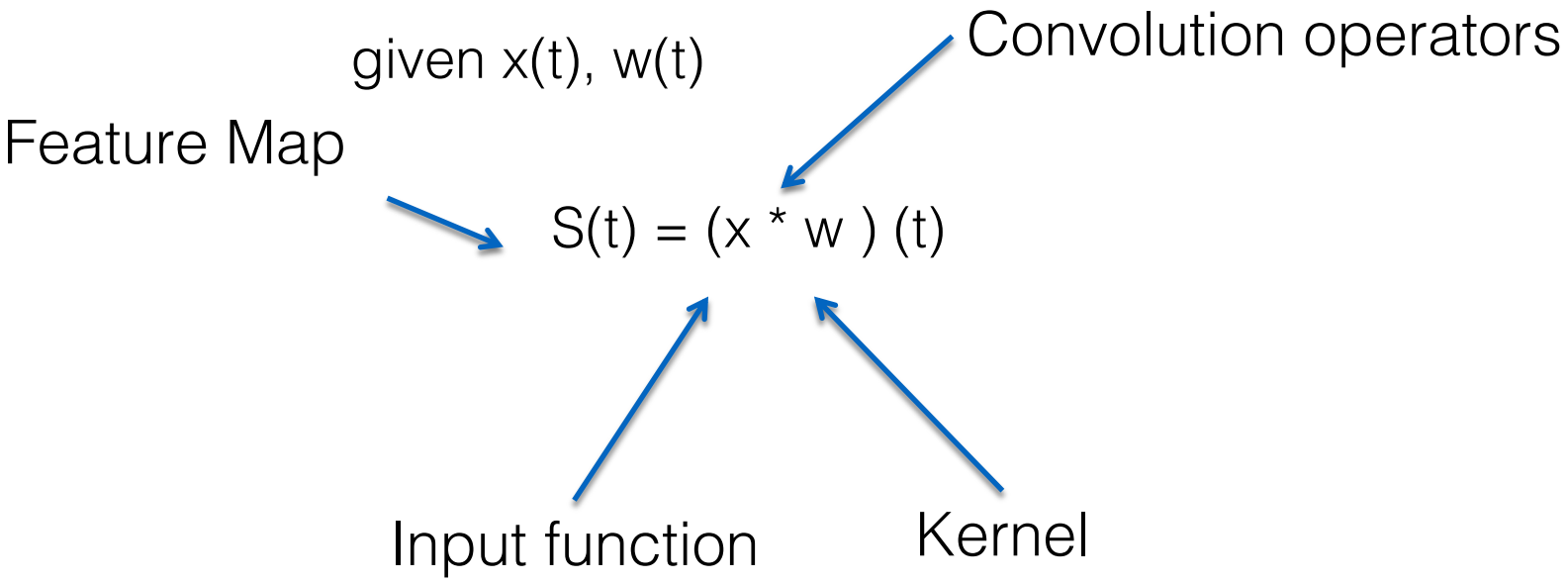
Input Image

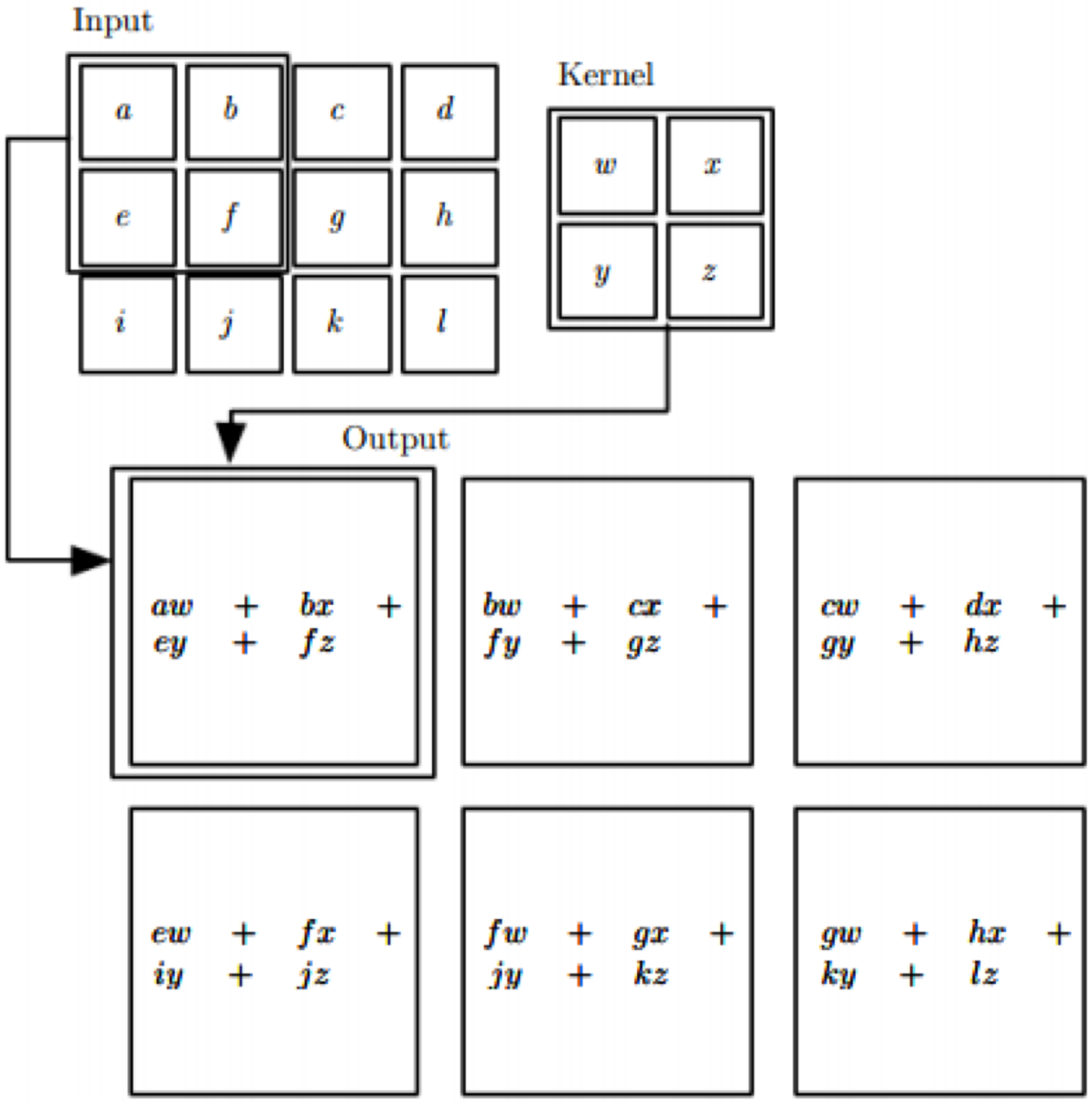
Convoluted Image

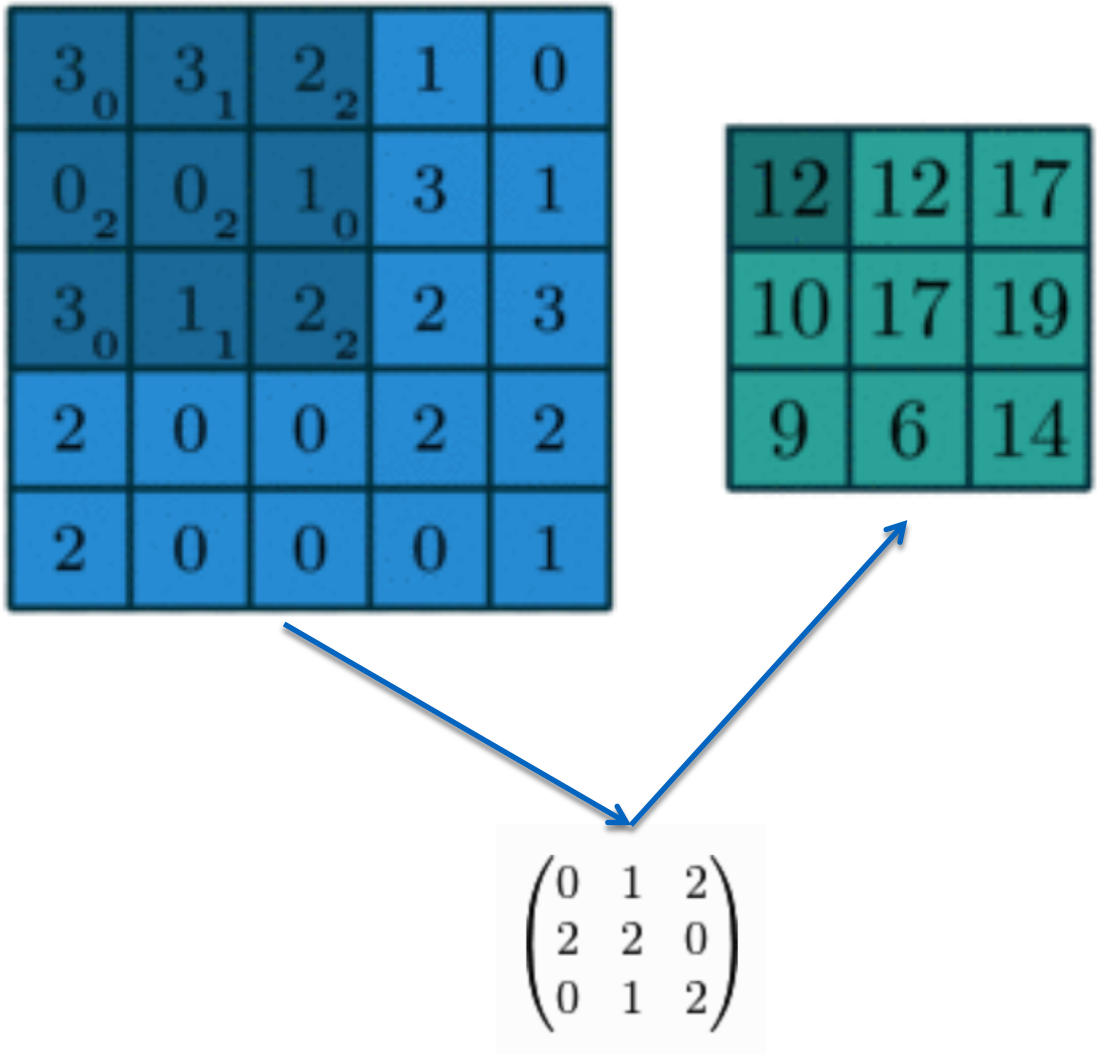
- Inspired by the neurophysiological experiments conducted by Hubel and Wiesel 1962.

Convolution Operation

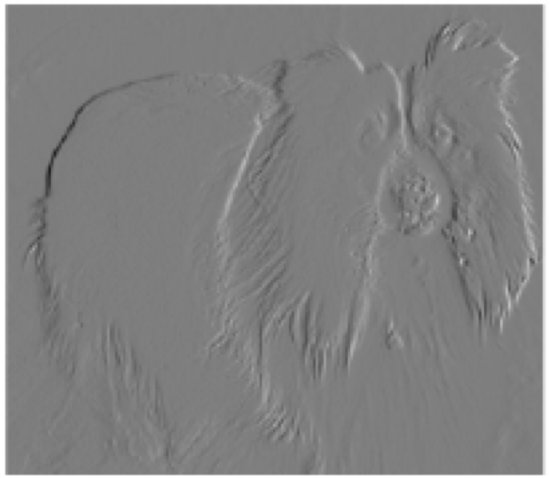
- A function transformation operation




Input
a b c d
e f g h
i j k l
Kernel
w x
y z
Output
aw + bx +
ey + fz
bw + cx +
fy + gz
cw + dx +
gy + hz
ew + fx +
iy + jz
fw + gx +
jy + kz
gw + hx +
ky + lz


3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
12 12 17
10 17 19
9 6 14
$\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$

Original image: 320 x 280
Kernel: { -1, 1}

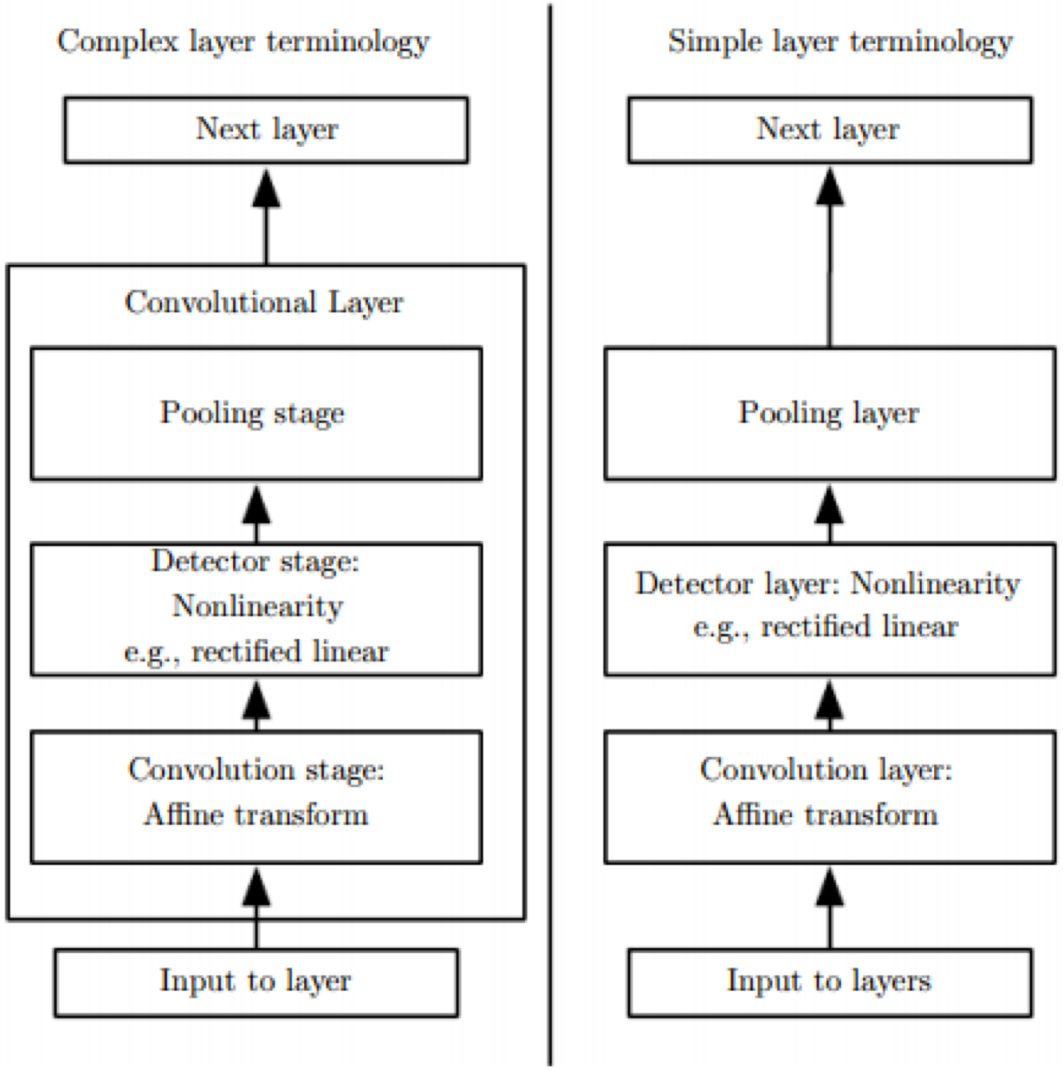


Output image: 319 x 280

Efficiency:

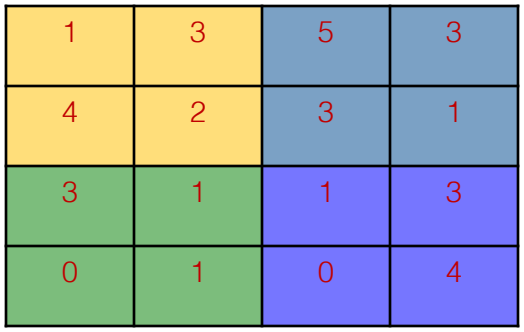
Convolution:
319 x 280 x 3 ops
V.s. Naïve fully connected matrix approach:
320 x 280 x 319 x 280

Convolutional Neural Network



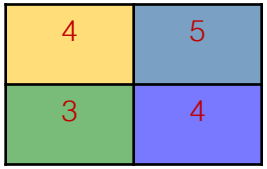
Pooling

- Max pooling: reports the maximum output within a rectangular neighborhood.
- Average pooling: reports the average output of a rectangular neighborhood.



Input Matrix

MaxPool with 2X2 filter with stride of 2

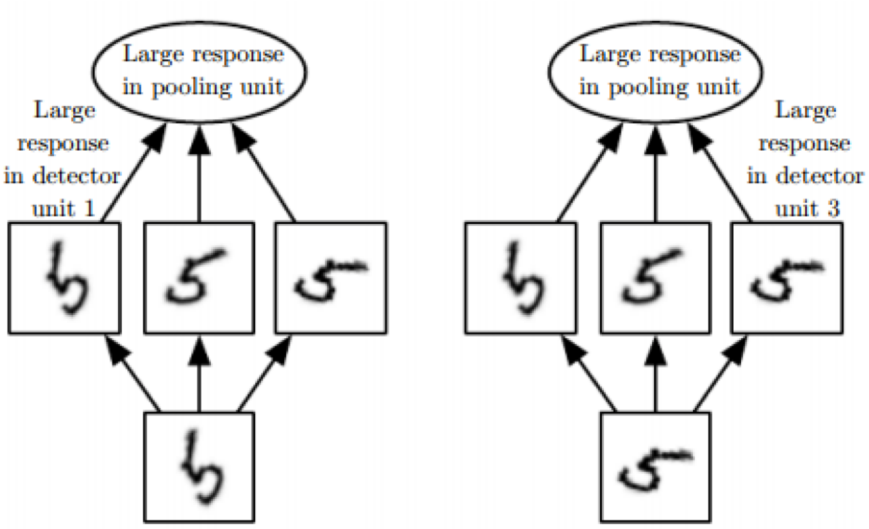


Output Matrix

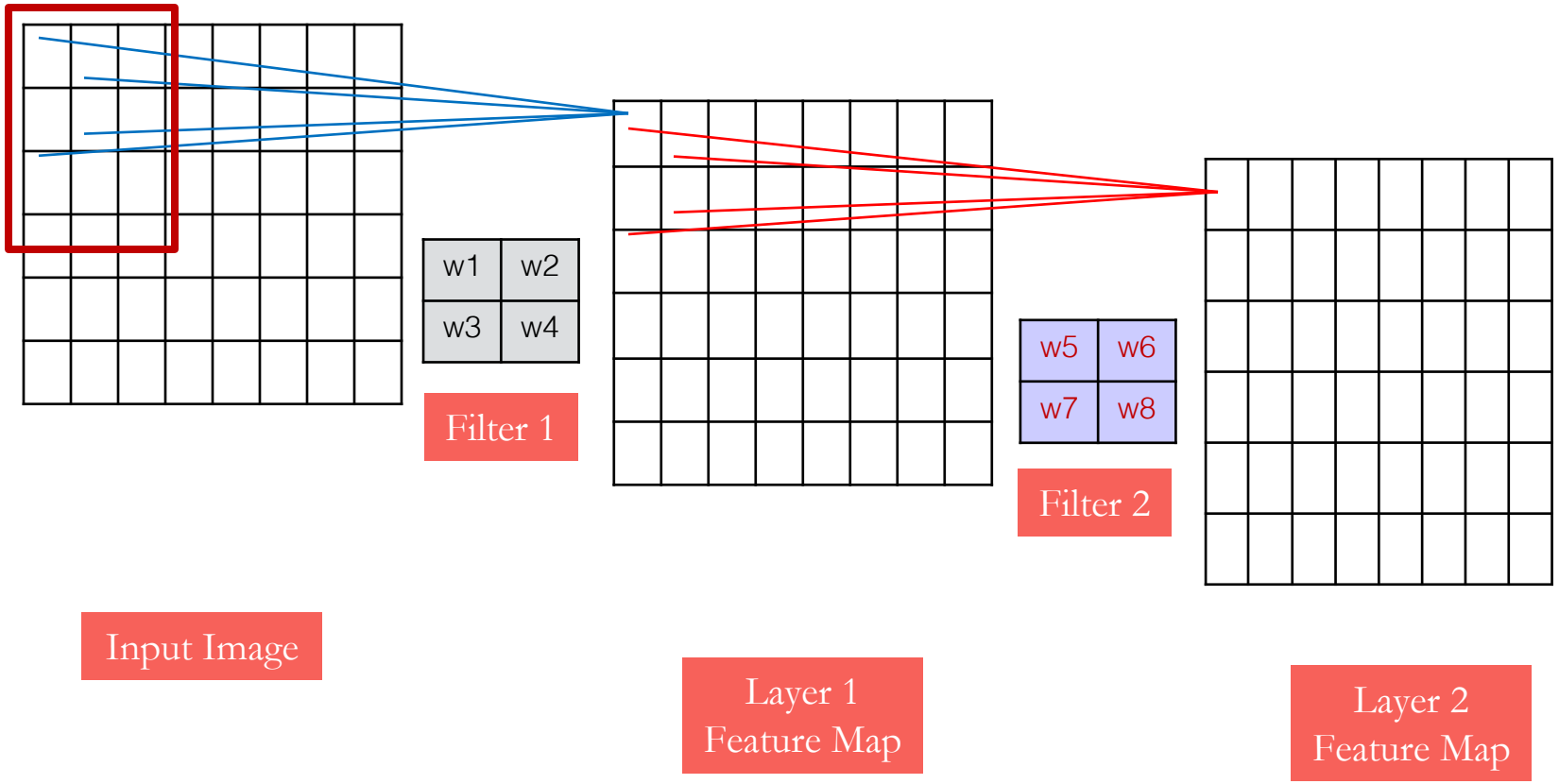
Why Pooling

- Reduce complexity

- Invariance to local translation



Lower Level to More Complex Features



- In Convolutional neural networks, hidden units are only connected to local receptive field.

Example – LeNet-5 – MNIST Classification

- How many parameters to be trained in total?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

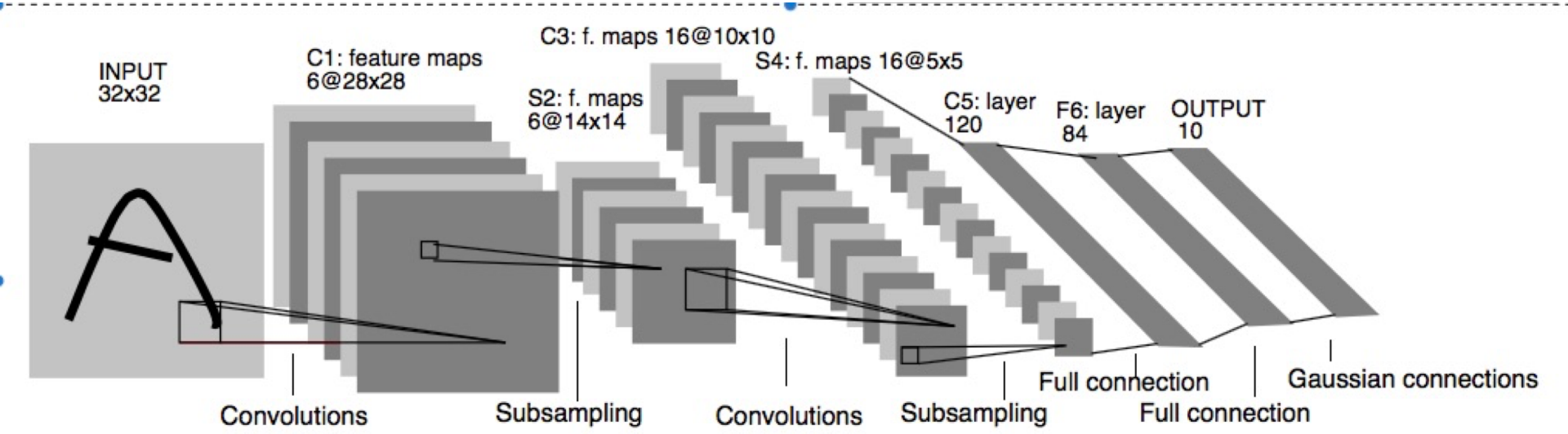


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

5x5 2x1 5x5 2x1 Fully Connected

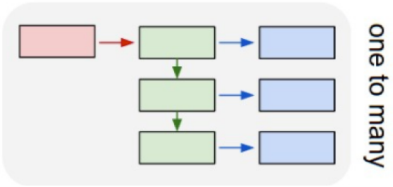
Recurrent Neural Networks

Why Recurrent Neuron Network ?

- The limitations of the Convolutional Neural Networks
  - Take fixed length vectors as input and produce fixed length vectors as output.
  - Allow fixed amount of computational steps.

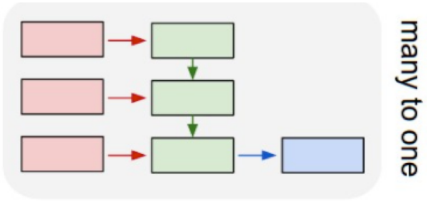
- We need to model the data with temporal or sequential structures and varying length of inputs and outputs

Modeling Sequences



A person riding a motorbike on dirt road

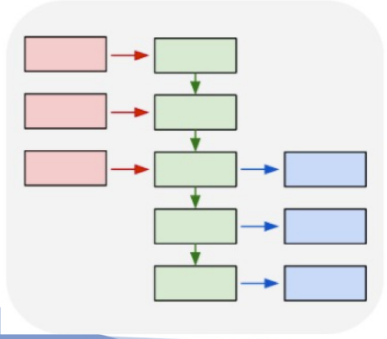
Image Captioning



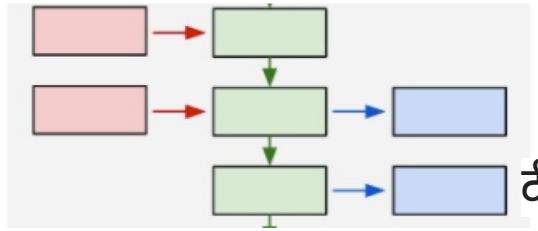
Awesome tutorial.

Positive

Sentiment Analysis



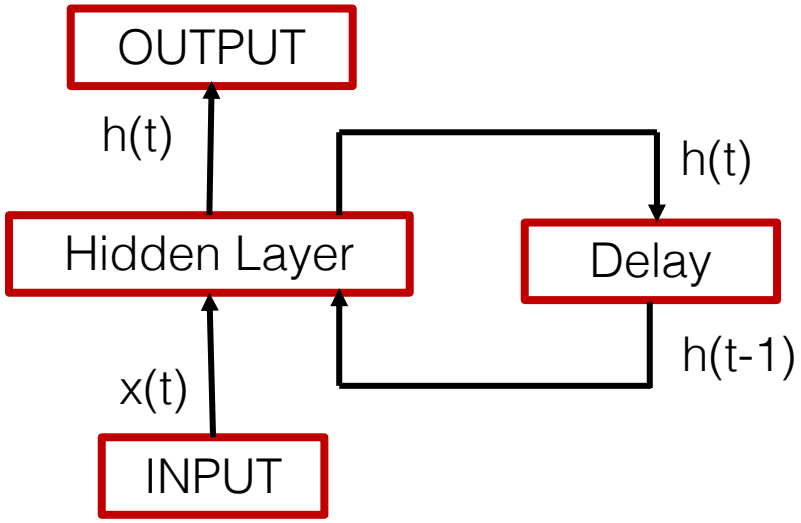
Happy Diwali



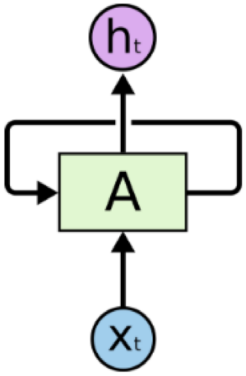
Machine Translation

RNN

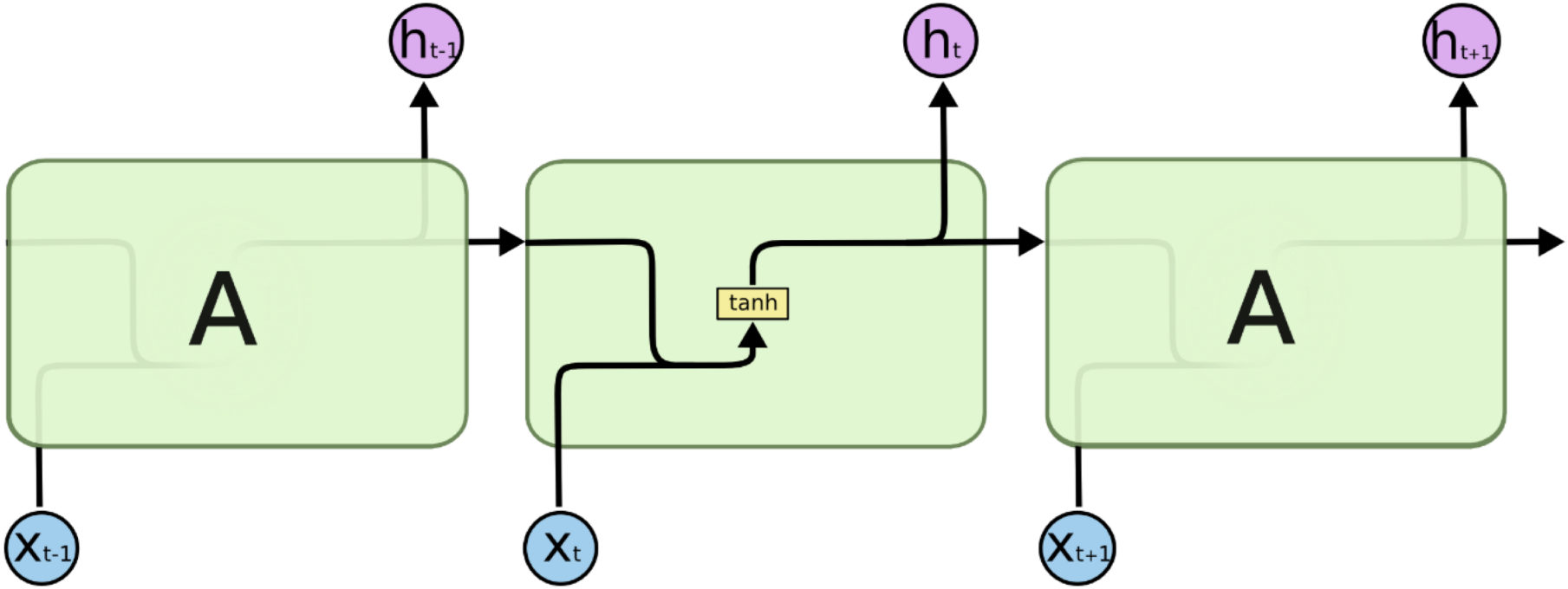
- Recurrent neural networks are connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one element at a time.
- Allows a memory of the previous inputs to persist in the model's internal state and influence the outcome.

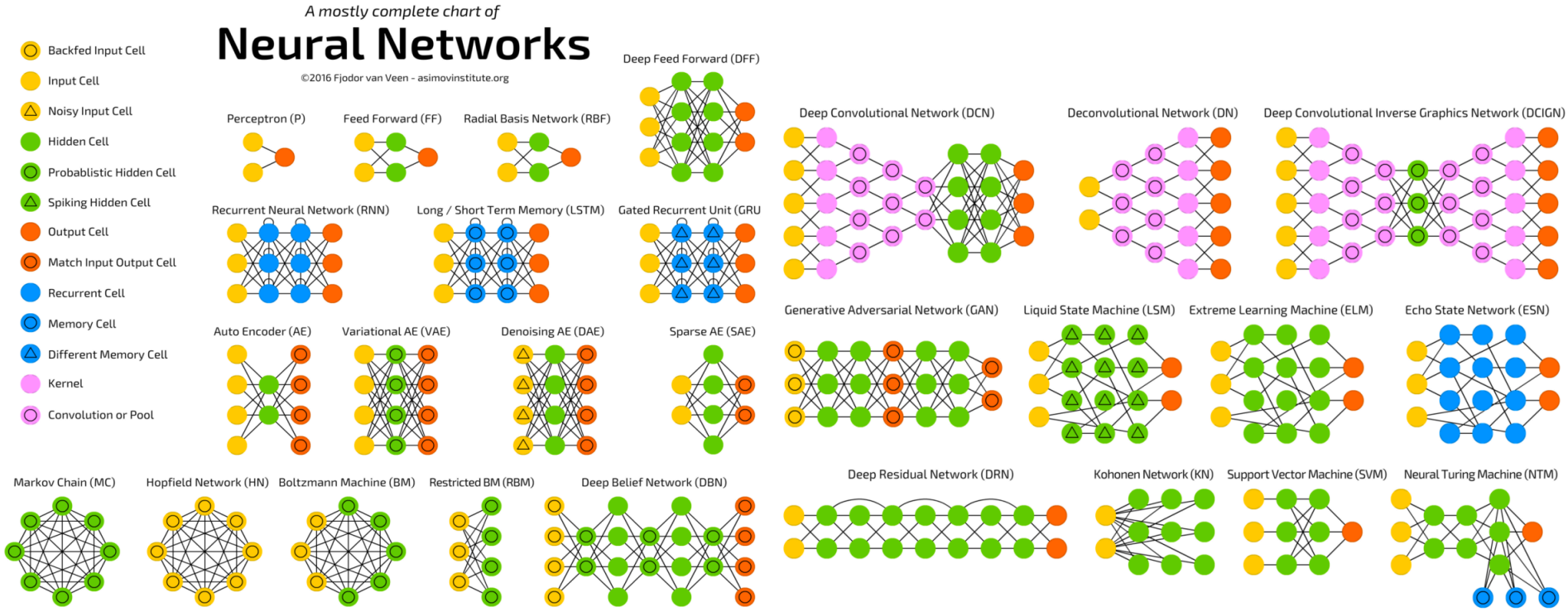


RNN Rolled Over Time



Simple RNN





https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463