

# Deep Neural Networks: A Quick and Rough Overview

Dr. Yanjun Qi

@ UVA Human Machine Intelligence  
Seminar

# Some Recent Trends

- 1. Autoencoder / layer-wise training
- 2. CNN / Residual / Dynamic parameter
- 3. RNN / Attention / Seq2Seq, ...
- 4. Neural Architecture with explicit memory
- 5. NTM 4program induction / sequential decisions
- 6. Learning to optimize / Learning DNN architectures
- 7. Learning to learn / meta-learning/ few-shots
- 8. DNN on graphs / trees
- 9. Deep Generative models, e.g., autoregressive
- 10. Generative Adversarial Networks (GAN)
- 11. Deep reinforcement learning
- 12. Validate / Evade / Test / Understand / Verify DNNs

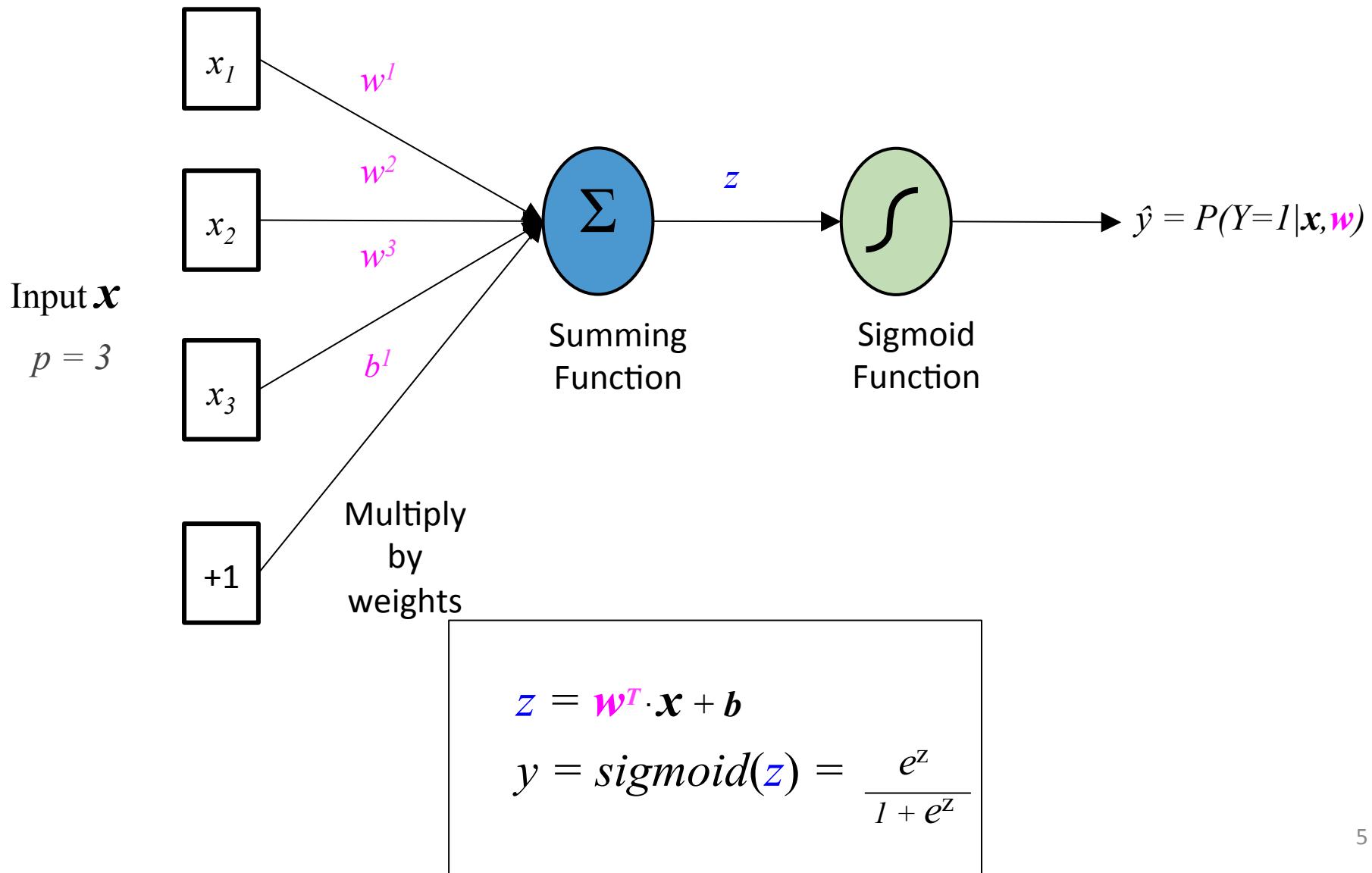
# Today

- Deep Learning
  - Basics
  - History
  - Why is this breakthrough ?
  - Recent trends
- 

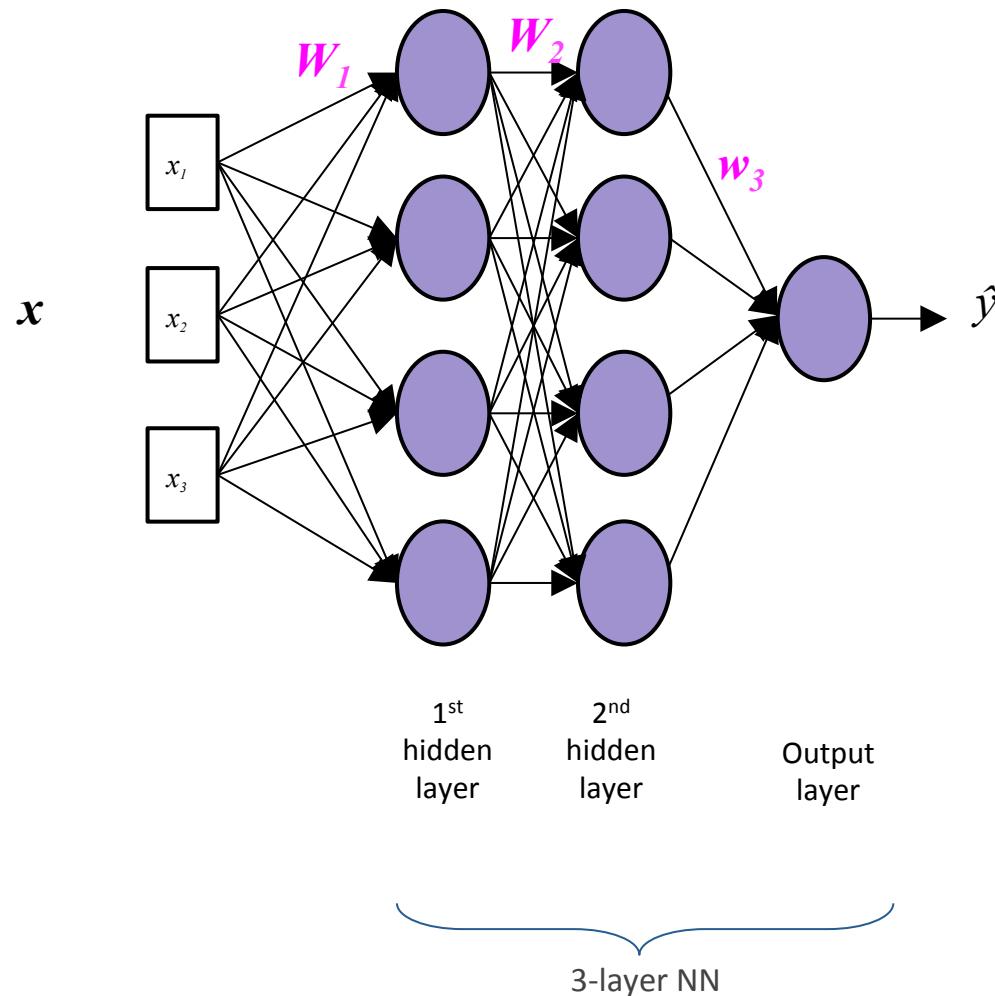
# Basics

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - training NN with backprob algorithm

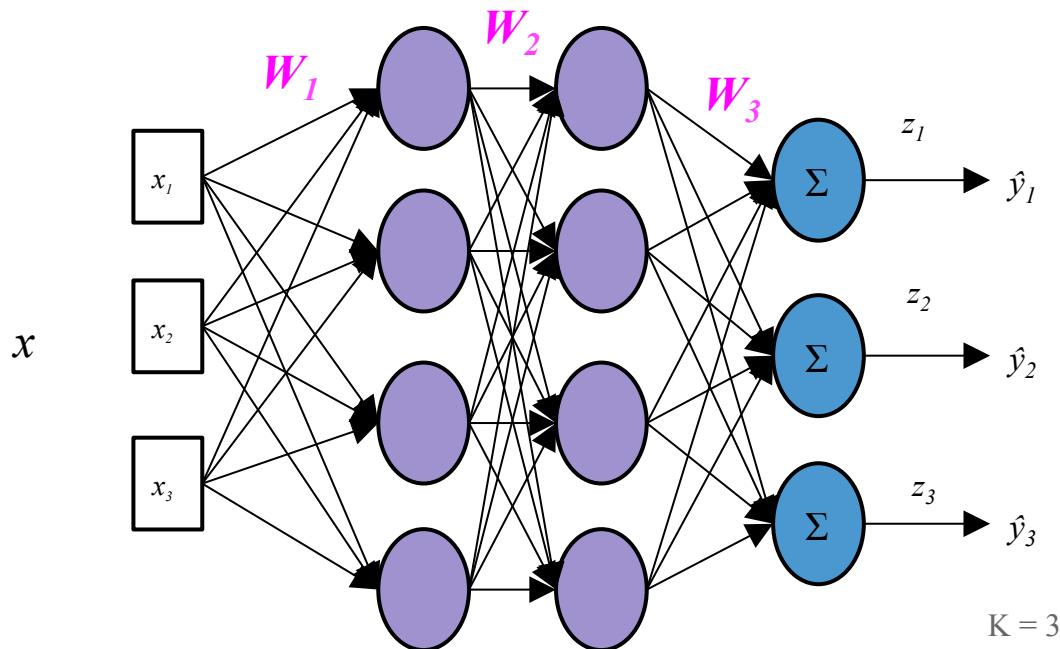
# One “Neuron”: Expanded Logistic Regression



# Multi-Layer Neural Network (MLP)



# Multi-Class Classification Loss



$K = 3$

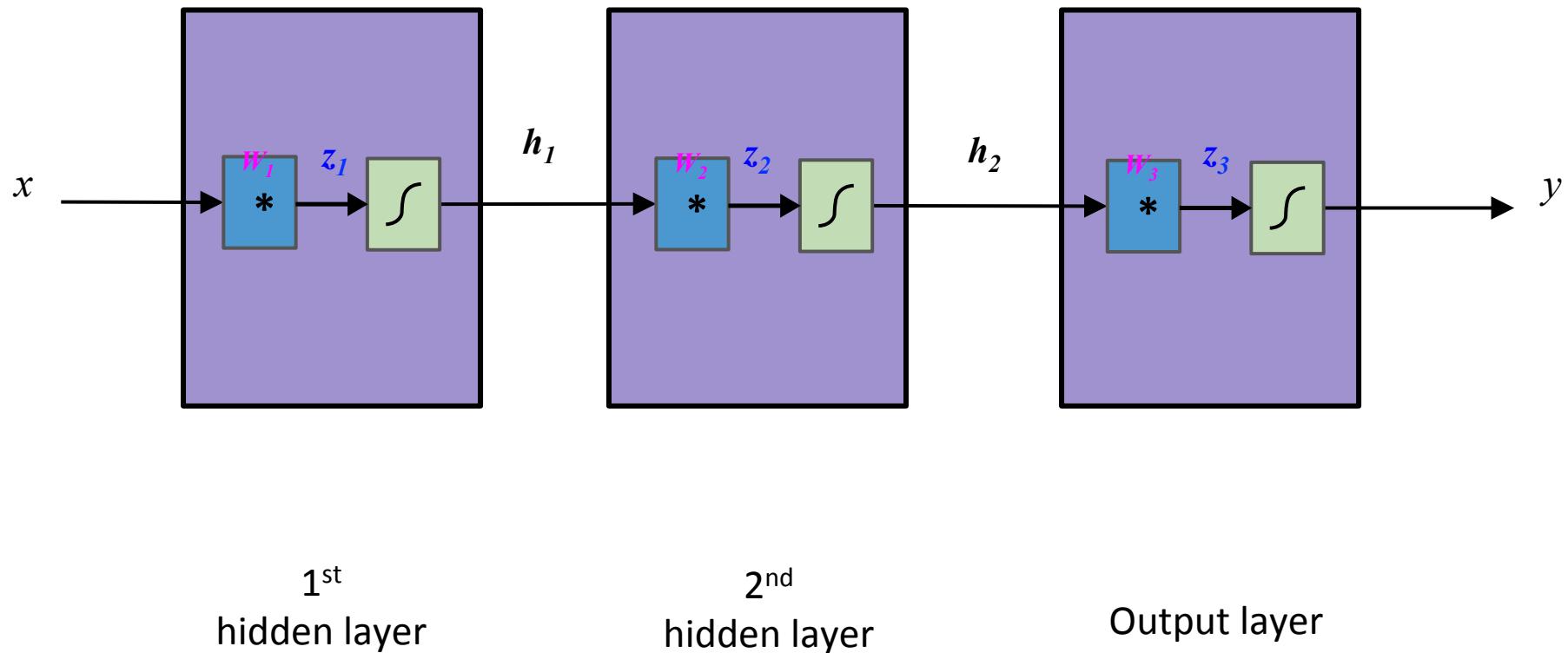
$$\hat{y}_i = \frac{e^{z_i}}{\sum_j e^{z_j}} = P(\hat{y}_i = 1 | \mathbf{x})$$

**“Softmax” function.**  
Normalizing function which converts each class output to a probability.

$$E = \text{loss} = - \sum_{j=1 \dots K} y_j \ln \hat{y}_j$$

“0” for all except true class

# “Block View” Of MLP

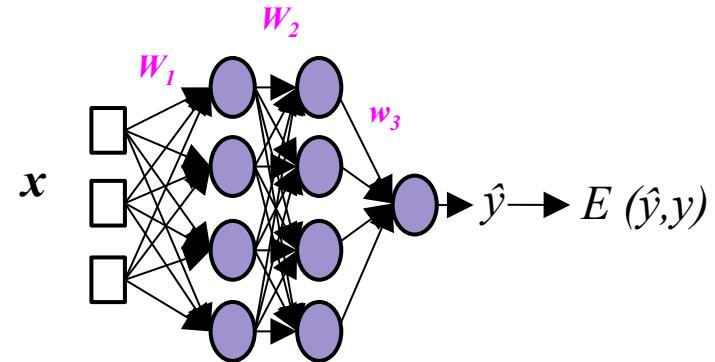


# Training Neural Networks

How do we learn the optimal weights  $\mathbf{W}_L$  for our task??

- **Stochastic Gradient descent:**

$$W_L(x_{t+1}) = W_L(x_t) - \eta \frac{\partial E}{\partial W_L(x_t)}$$



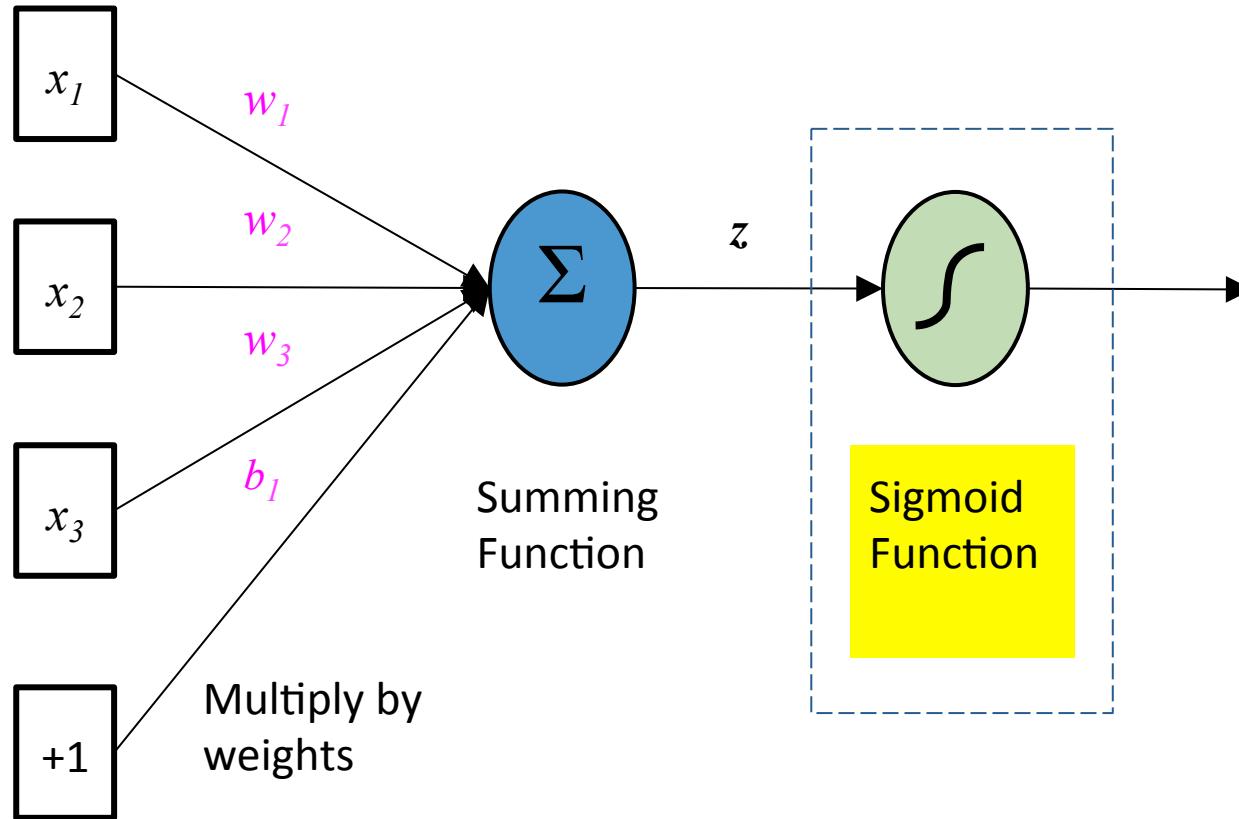
But how do we get gradients of lower layers?

- **Backpropagation!**

- Repeated application of chain rule of calculus
- Locally minimize the objective
- Requires all “blocks” of the network to be differentiable

# Nonlinearity Functions

(i.e. transfer or activation functions)



# Nonlinearity Functions

(aka transfer or activation functions)

Name	Plot	Equation	Derivative ( w.r.t $x$ )
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectifier (ReLU) <sup>[9]</sup>		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$



usually works best in practice

# Today

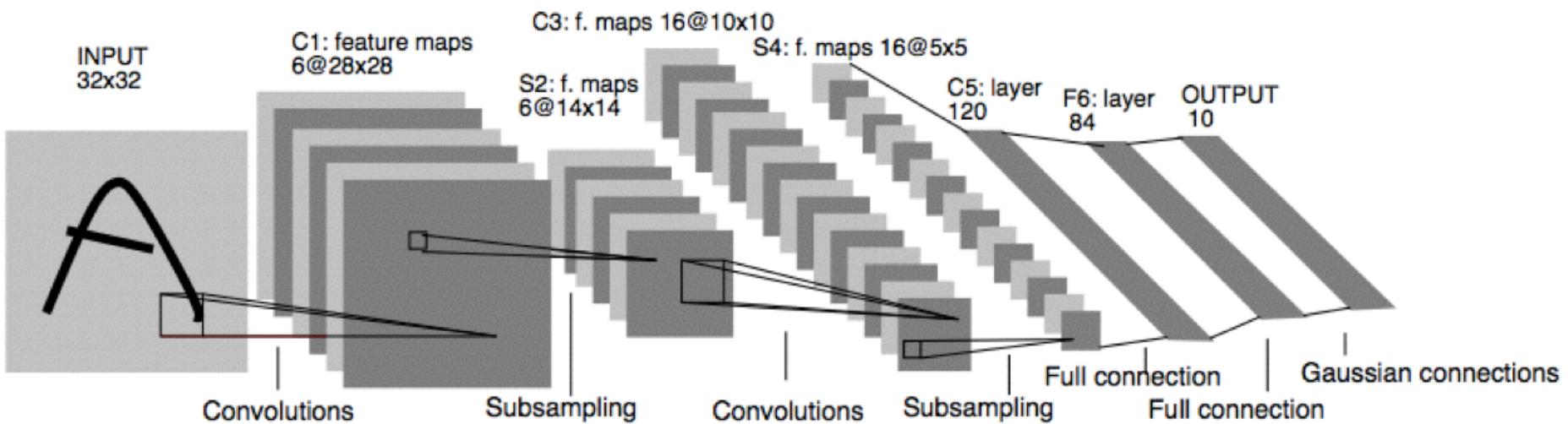
- Deep Learning
- Basics
-  ➤ History
- Why is this breakthrough ?
- Recent trends

# Many classification models invented since late 80's

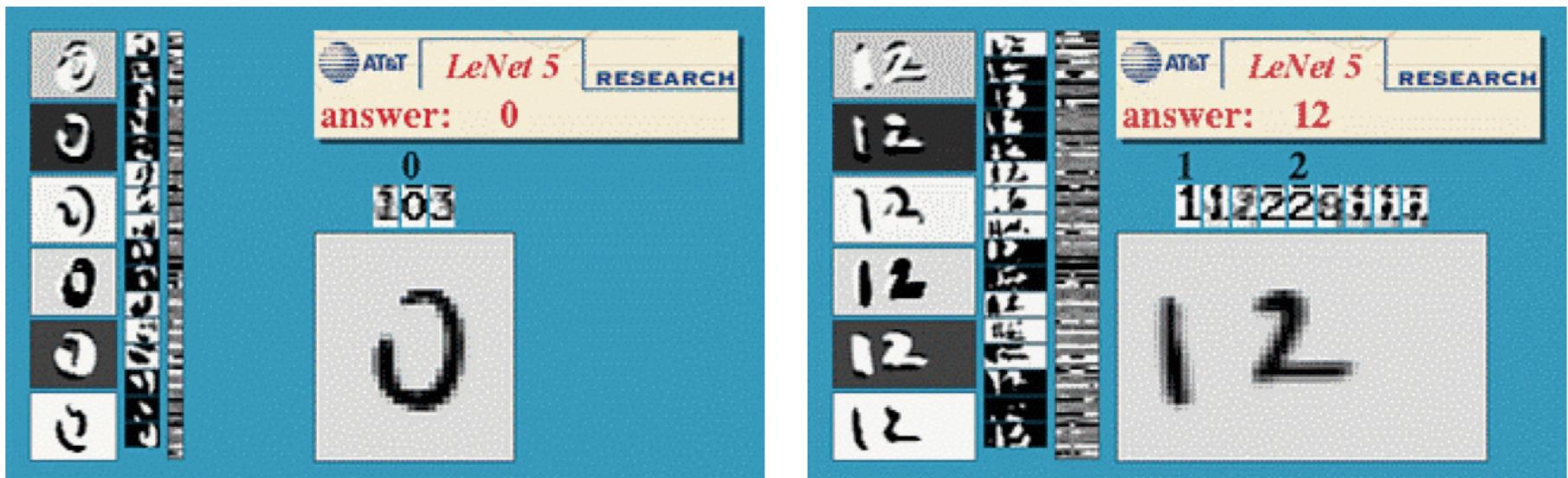
- Neural networks
- Boosting
- Support Vector Machine
- Maximum Entropy
- Random Forest
- .....

# Deep Learning in the 90's

- Prof. Yann LeCun invented Convolutional Neural Networks
- First NN successfully trained with many layers



# “LeNet” Early success at OCR



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition,  
Proceedings of the IEEE 86(11): 2278–2324, 1998.

# Between ~2000 to ~2011 Machine Learning Field Interest

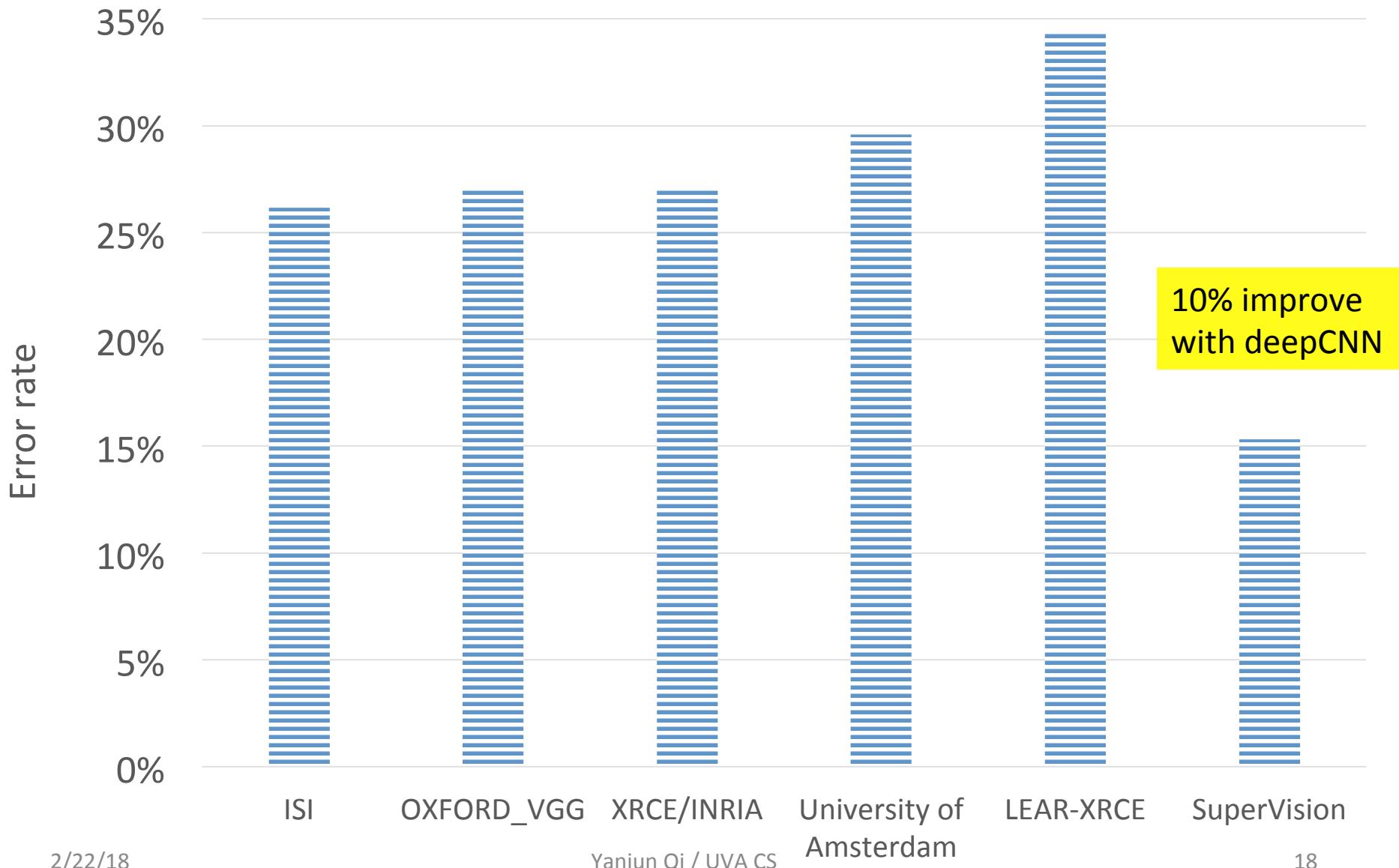
- Learning with Structures ! + Convex Formulation!
  - Kernel learning
  - Transfer Learning
  - Semi-supervised
  - Manifold Learning
  - Sparse Learning
  - Structured input-output learning ...
  - Graphical model

# “Winter of Neural Networks”

## Since 90’s !    to ~2010

- Non-convex
- Need a lot of tricks to play with
  - How many layers ?
  - How many hidden units per layer ?
  - What topology among layers ? .....
- Hard to perform theoretical analysis

# Large-Scale Visual Recognition Challenge 2012



# 10 BREAKTHROUGH TECHNOLOGIES 2013

[Introduction](#)[The 10 Technologies](#)[Past Years](#)

## Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.



## Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.



## Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?



## Additive Manufacturing

Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.



## Baxter: The Blue-Collar Robot

Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.



## Memory Implants

A maverick neuroscientist believes he has deciphered the code by which the brain forms long-term memories. Next: testing a prosthetic implant for people suffering from long-term memory loss.



## Smart Watches

The designers of the Pebble watch realized that a mobile phone is more useful if you don't have to take it out of your pocket.



## Ultra-Efficient Solar Power

Doubling the efficiency of a solar cell would completely change the economics of renewable energy. Nanotechnology just might make it possible.



## Big Data from Cheap Phones

Collecting and analyzing information from simple cell phones can provide surprising insights into how people move about and behave – and even help us understand the spread of diseases.



## Supergrids

A new high-power circuit breaker could finally make highly efficient DC power grids practical.

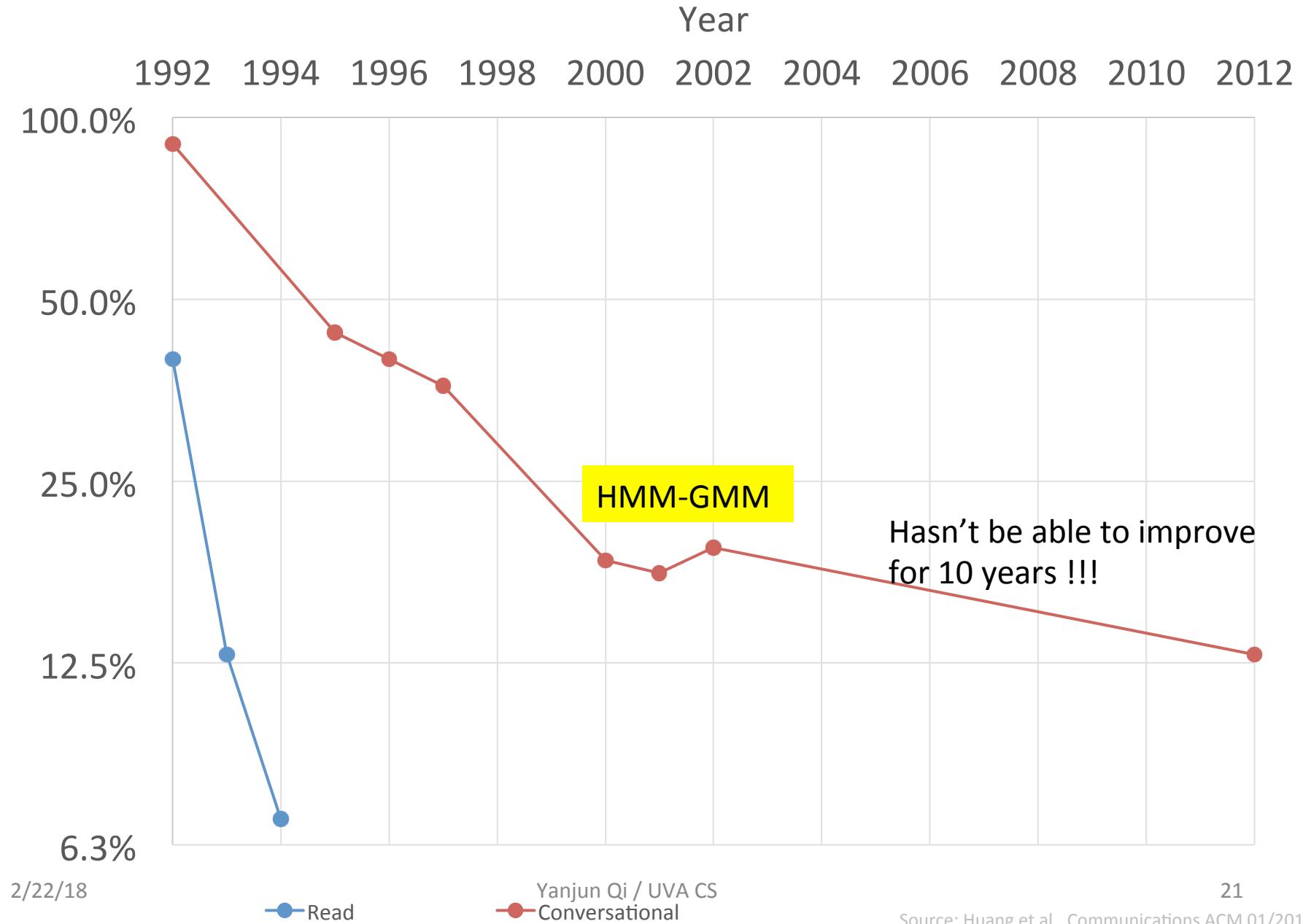


# Today

- Deep Learning
  - Basics
  - History
- Why is this breakthrough ?
- Recent trends



# Speech Recognition



# **WHY BREAKTHROUGH ?**

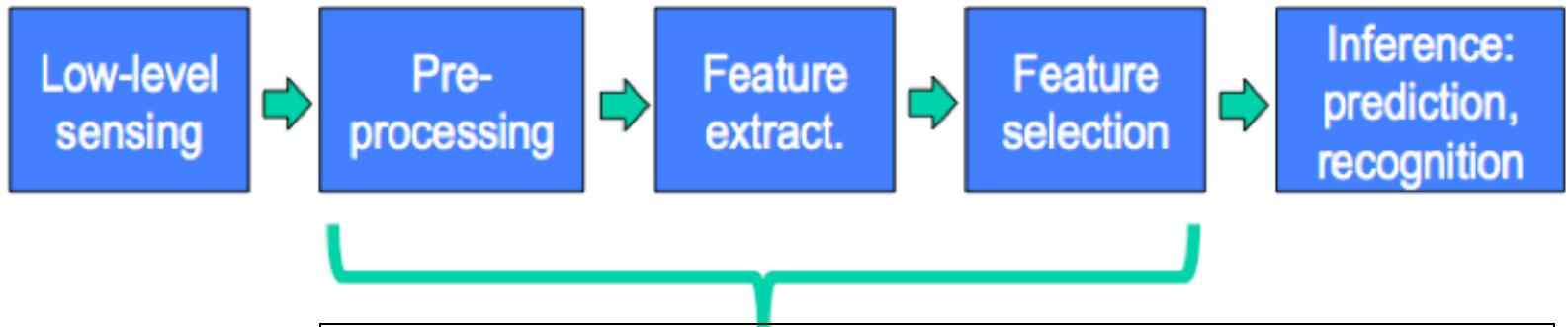
# How can we build more intelligent computer / machine ?

- Able to
  - Perceive the world
  - Understand the world
  - Interact with the world
- This needs
  - Basic speech capabilities
  - Basic vision capabilities
  - Language understanding
  - User behavior / emotion understanding
  - Being Able to think / Reason

# Plenty of Data

- **Text:** trillions of words of English + other languages
- **Visual:** billions of images and videos
- **Audio:** thousands of hours of speech per day
- **User activity:** queries, user page clicks, map requests, etc,
- **Knowledge graph:** billions of labeled relational triplets
- .....

# Deep Learning Way: Learning features / Representation from data



## Feature Engineering

- ✓ Most critical for accuracy
- ✓ Account for most of the computation for testing
- ✓ Most time-consuming in development cycle
- ✓ Often hand-craft and task dependent in practice



## Feature Learning

- ✓ Easily adaptable to new similar tasks
- ✓ Layerwise representation
- ✓ Layer-by-layer unsupervised training
- ✓ Layer-by-layer supervised training

# To perceive the world: Application I: Objective Recognition / Image Labeling



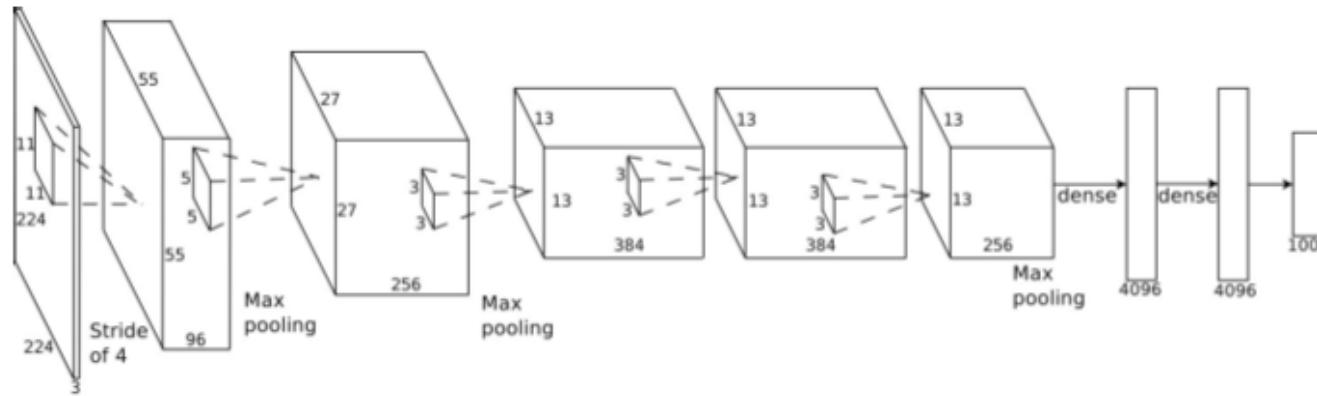
72%, 2010

74%, 2011

85%, 2012

“Very large-scale” ImageNet competition  
(training on 1.2 million images [X]  
vs. 1000 different word labels [Y])

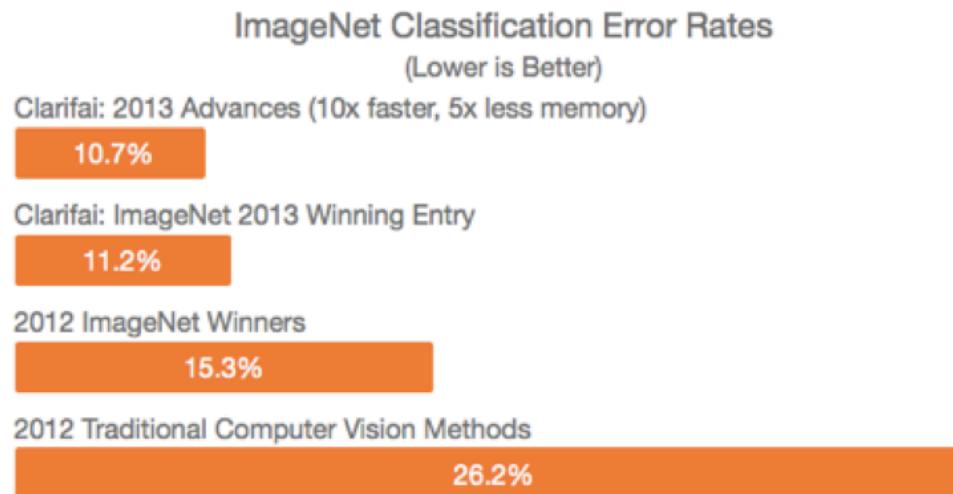
# To perceive the world: Application I: Objective Recognition / Image Labeling



Deep Convolution Neural Network (**CNN**) and variants have won (as best systems) on “very large-scale” ImageNet competition 2012-2017

# ImageNet Challenge 2013

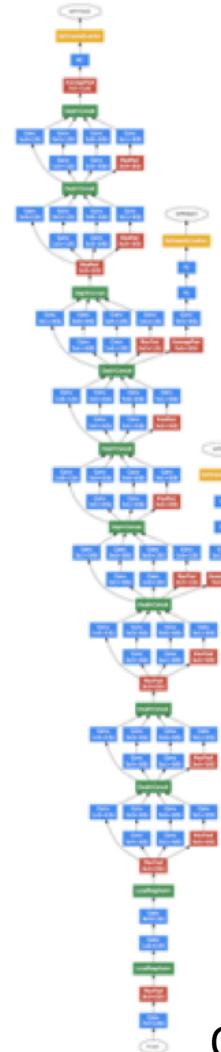
- Clarifai ConvNet model wins at 11% error rate



- Many other participants used ConvNets
- OverFeat by Pierre Sermanet from NYU: shipped binary program to execute pre-trained models

# ImageNet Challenge 2014

- In the mean time Pierre Sermanet had joined other people from Google Brain
- Monster model: GoogLeNet now at **6.7% error rate**

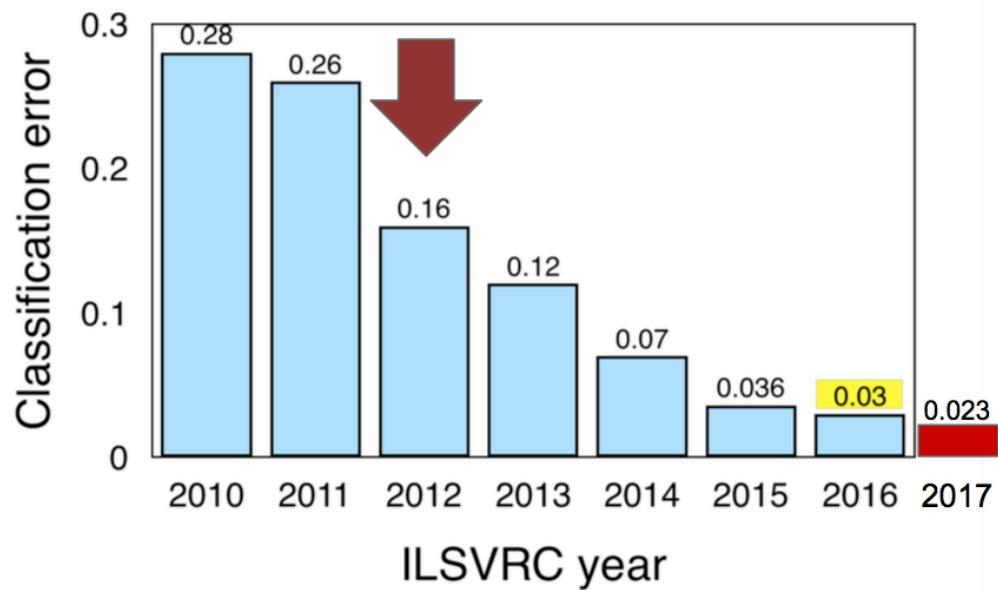


# ImageNet Challenge

Arch

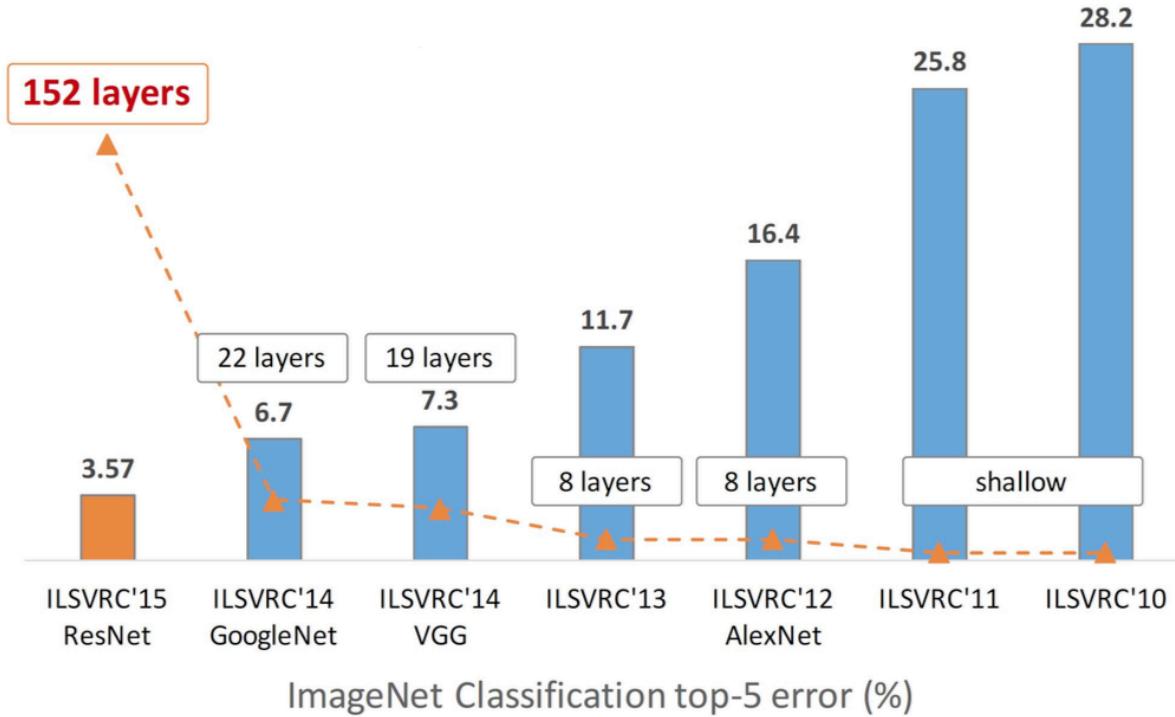


- 2010-11: hand-crafted computer vision pipelines
- 2012-2016: ConvNets
  - 2012: AlexNet
    - major deep learning success
  - 2013: ZFNet
    - improvements over AlexNet
  - 2014
    - VGGNet: deeper, simpler
    - InceptionNet: deeper, faster
  - 2015
    - ResNet: even deeper
  - 2016
    - ensembled networks
  - 2017
    - Squeeze and Excitation Network



# Revolution of Depth

Arch

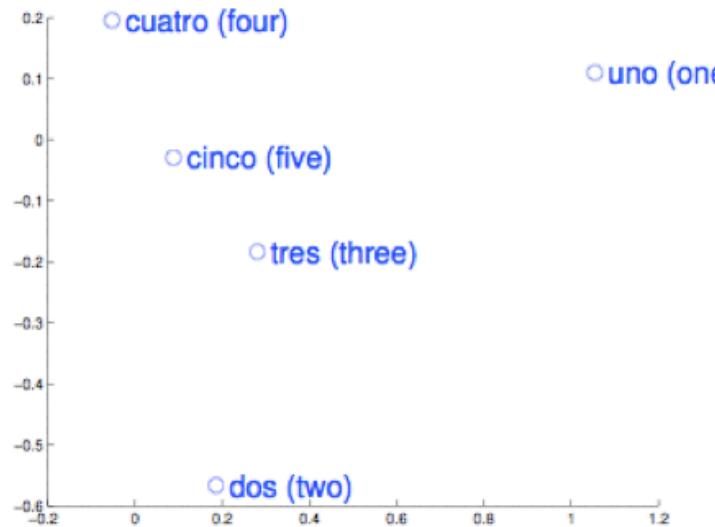


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

# To understand the world. Application II: Semantic Understanding

e.g. Word embedding / Neural Language Models

- Learn to embed each word into a vector of real values
  - Semantically **similar** words have **closer** embedding representations
- Progress in 2013/14
  - Can uncover semantic /syntactic word relationships
    - [king] - [male] + [female]  $\sim=$  [queen]
    - [Berlin] - [Germany] + [France]  $\sim=$  [Paris]
    - [eating] - [eat] + [fly]  $\sim=$  [flying]



Yanjun QI / UVA CS

source: Exploiting Similarities among Languages for MT

# To interact with the world: Application III: Deep Learning to Play Games

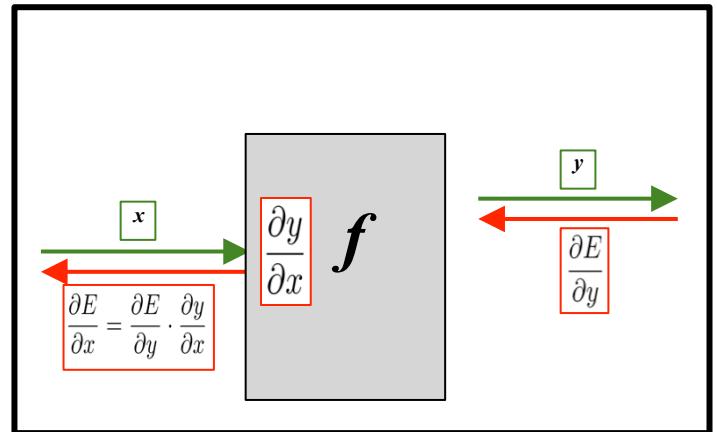
- DeepMind:
  - Learning to Play & win dozens of Atari games
  - new Deep Reinforcement Learning algorithms
  - AlphaGo / AlphaZero

# Able to Reason: Application IV: Deep Learning to Execute and Program

## Neural Turing Machines

- Google DeepMind, October 2014 (very new)
- Neural Network coupled to external memory (tape)
- Analogue to a Turing Machine but differentiable
- Can be used to learn to simple programs from example input / output pairs

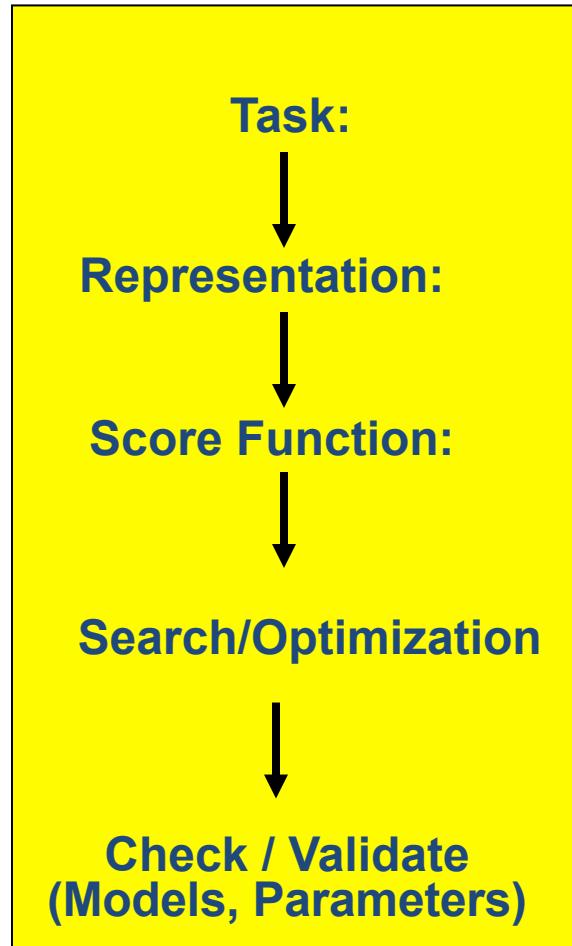
# Building Deep Neural Nets



# Today

- Deep Learning
  - Basics
  - History
  - Why is this breakthrough ?
- ➤ Recent trends

# Machine (Deep) Learning in a Nutshell



# DESIGN Deep NN: Five Aspects

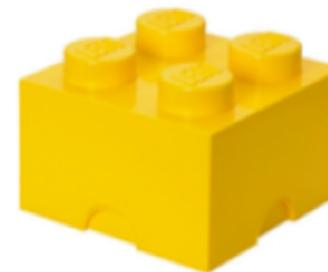
- Tasks:
  - Generative / Discriminative / Reinforce / Reasoning ...
- Formulate Input / Output:
  - Data representation
- Architecture Design:
  - Network Topology, Network Parameters
- Training / Searching / Learning
  - With new losses / with new optimization tricks
  - New formulation of learning
  - Scaling up with GPU, Scaling up with distributed optimization , e.g. Asynchronous SGD
- Validation / Trust / Test / Understand ...

# Some Trends after ~2010

More Details and Materials from  
My 2017 Fall Advanced Deep Learning  
<https://qdata.github.io/deep2Read/>



**Inputs and Outputs**



**Losses**



**Architectures:**

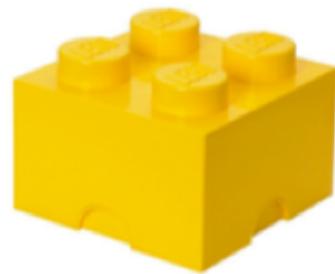


**Validation**

# Some Recent Trends

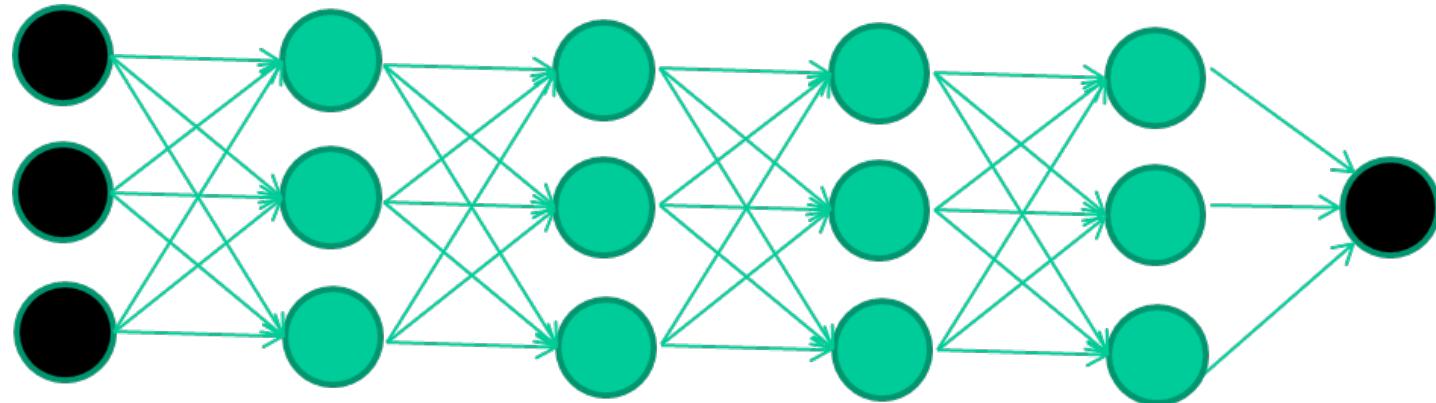
- 1. Autoencoder / layer-wise training
- 2. CNN / Residual / Dynamic parameter
- 3. RNN / Attention / Seq2Seq, ...
- 4. Neural Architecture with explicit memory
- 5. NTM 4program induction / sequential decisions
- 6. Learning to optimize / Learning DNN architectures
- 7. Learning to learn / meta-learning/ few-shots
- 8. DNN on graphs / trees
- 9. Deep Generative models, e.g., autoregressive
- 10. Generative Adversarial Networks (GAN)
- 11. Deep reinforcement learning
- 12. Validate / Evade / Test / Understand / Verify DNNs

# Recent Trend (1): Layer-wise pretraining / Auto-Encoder

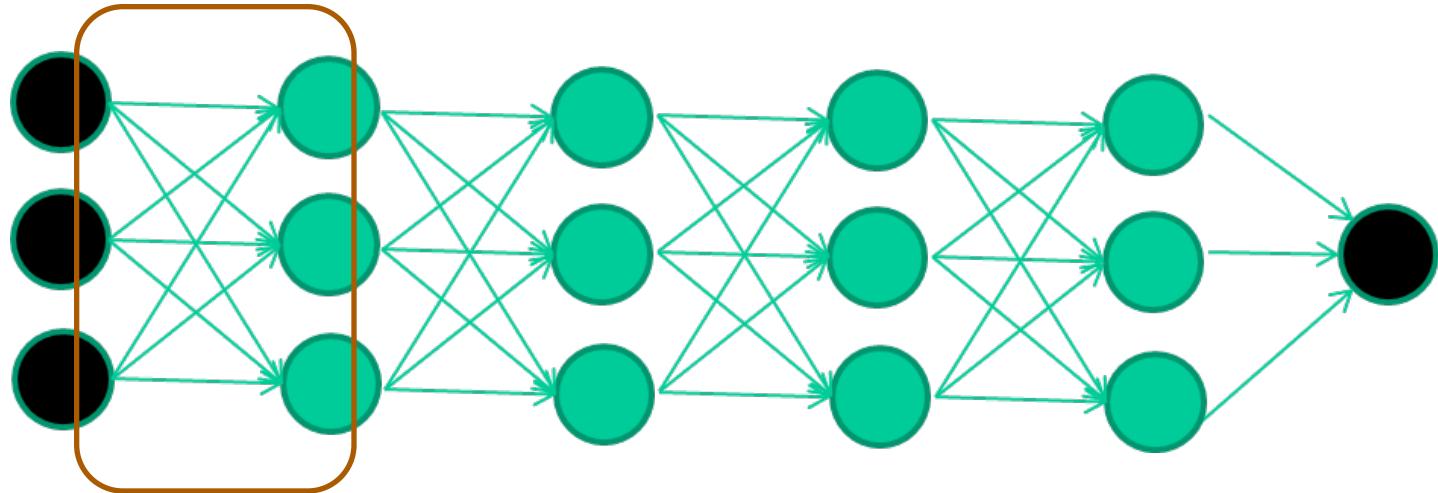


**Losses**

# The new way to train multi-layer NNs...

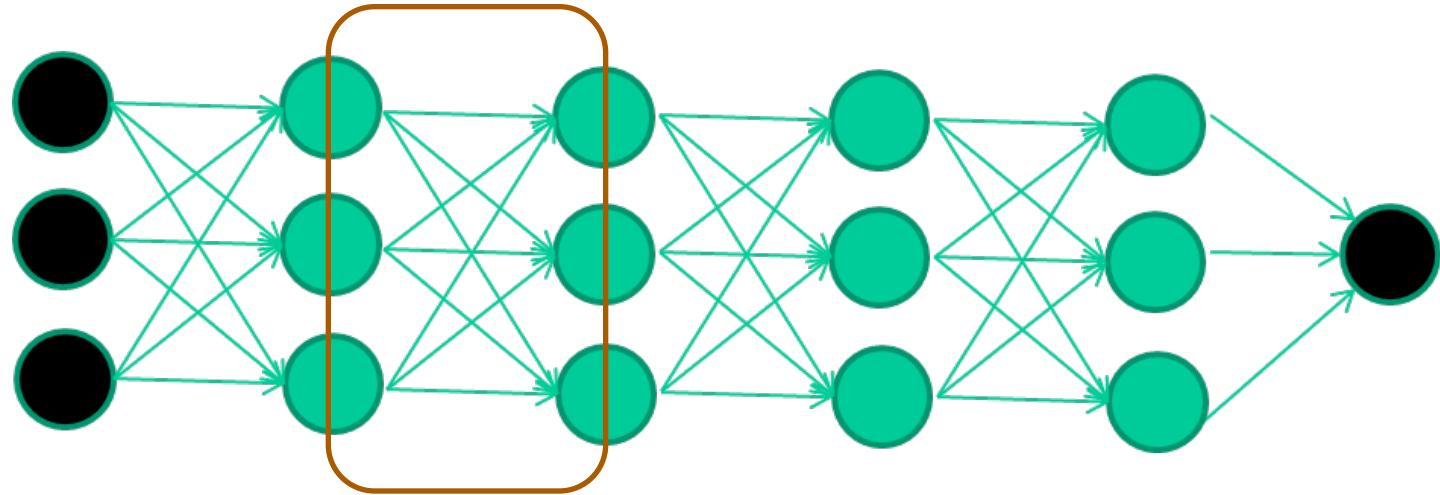


# The new way to train multi-layer NNs...



Train **this** layer first

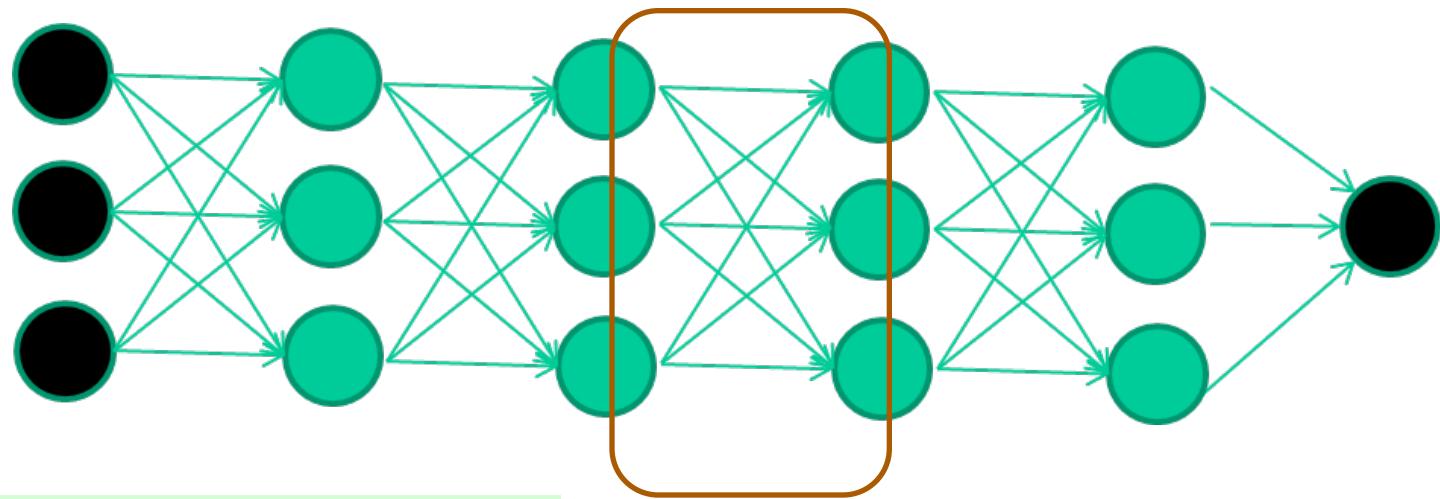
# The new way to train multi-layer NNs...



Train **this** layer first

then **this** layer

# The new way to train multi-layer NNs...

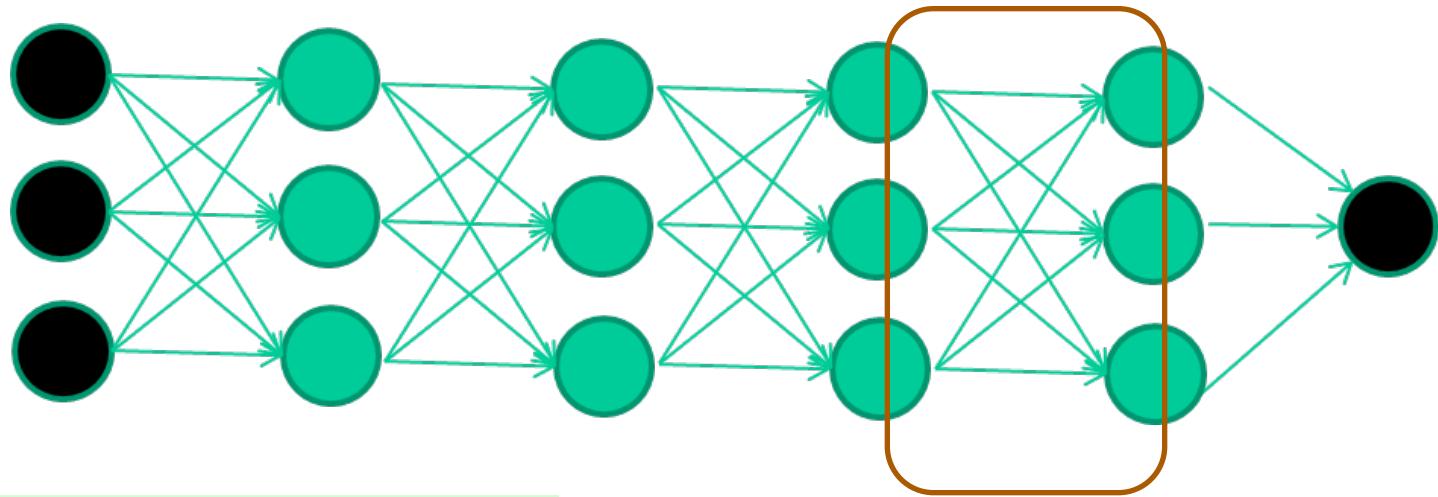


Train **this** layer first

then **this** layer

then **this** layer

# The new way to train multi-layer NNs...



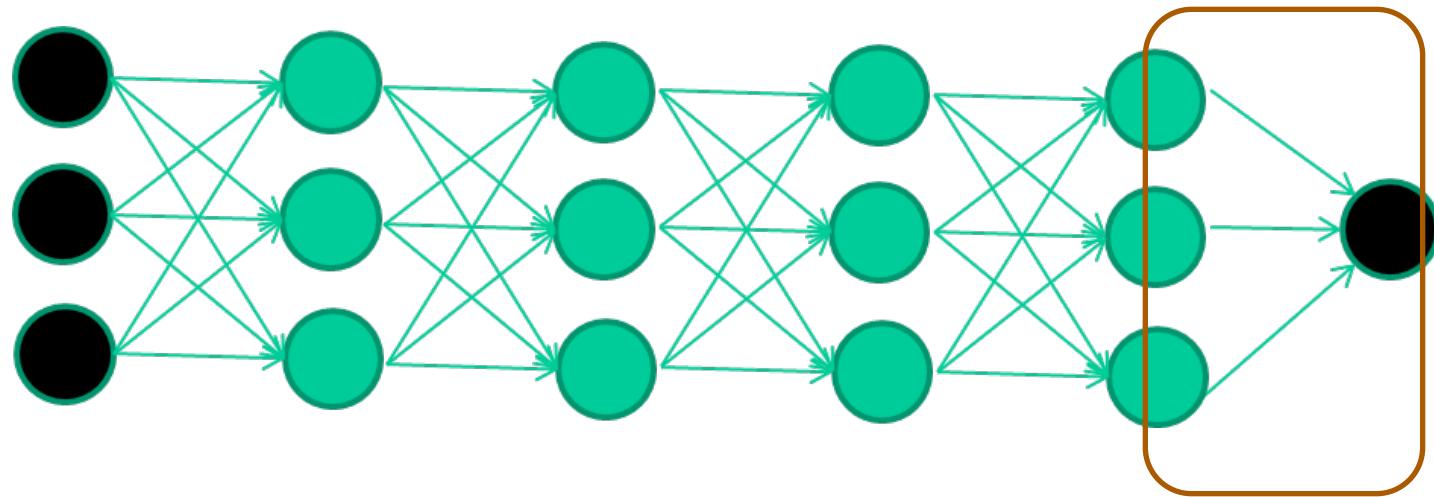
Train **this** layer first

then **this** layer

then **this** layer

then **this** layer

# The new way to train multi-layer NNs...



Train **this** layer first

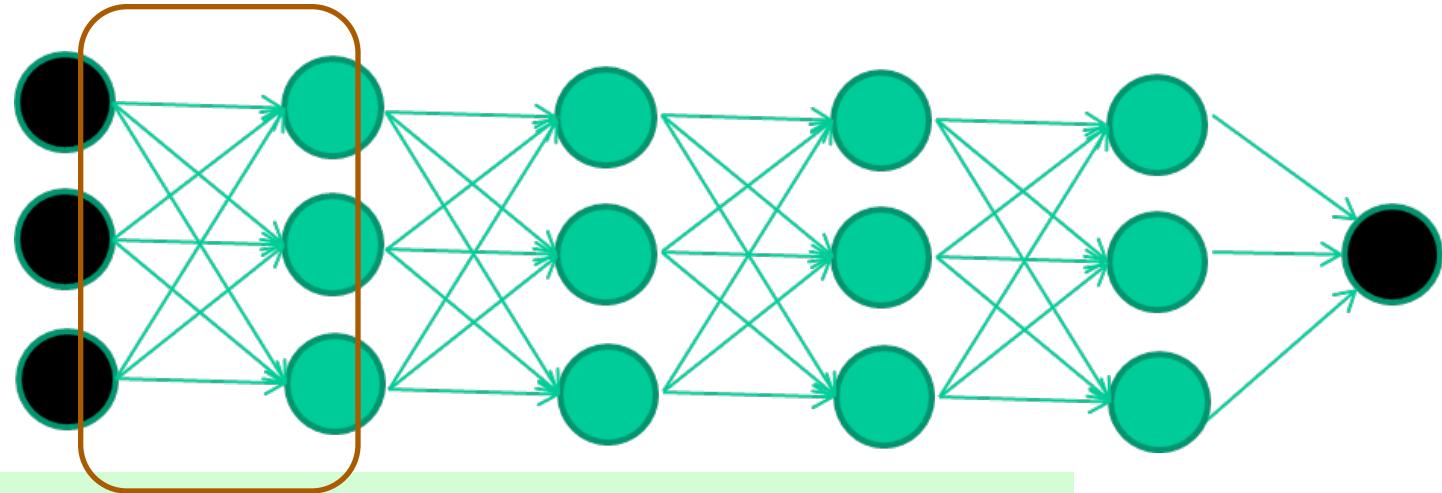
then **this** layer

then **this** layer

then **this** layer

finally **this** layer

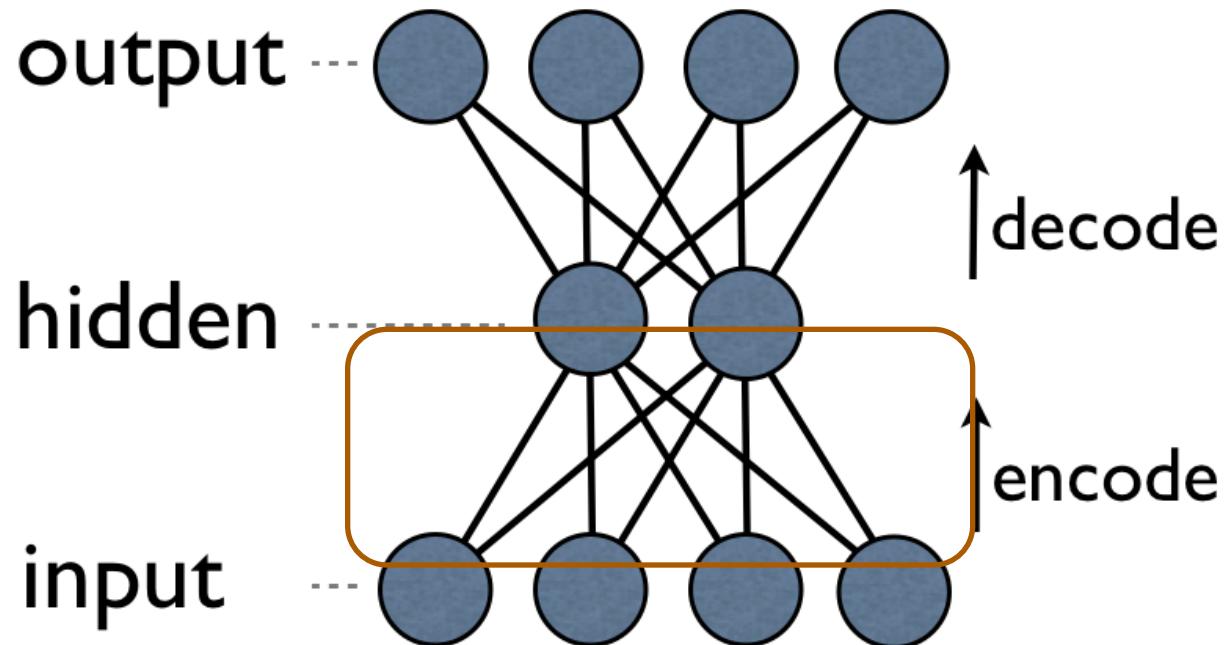
# The new way to train multi-layer NNs...



*EACH of the (non-output) layers is trained  
to be an **auto-encoder***

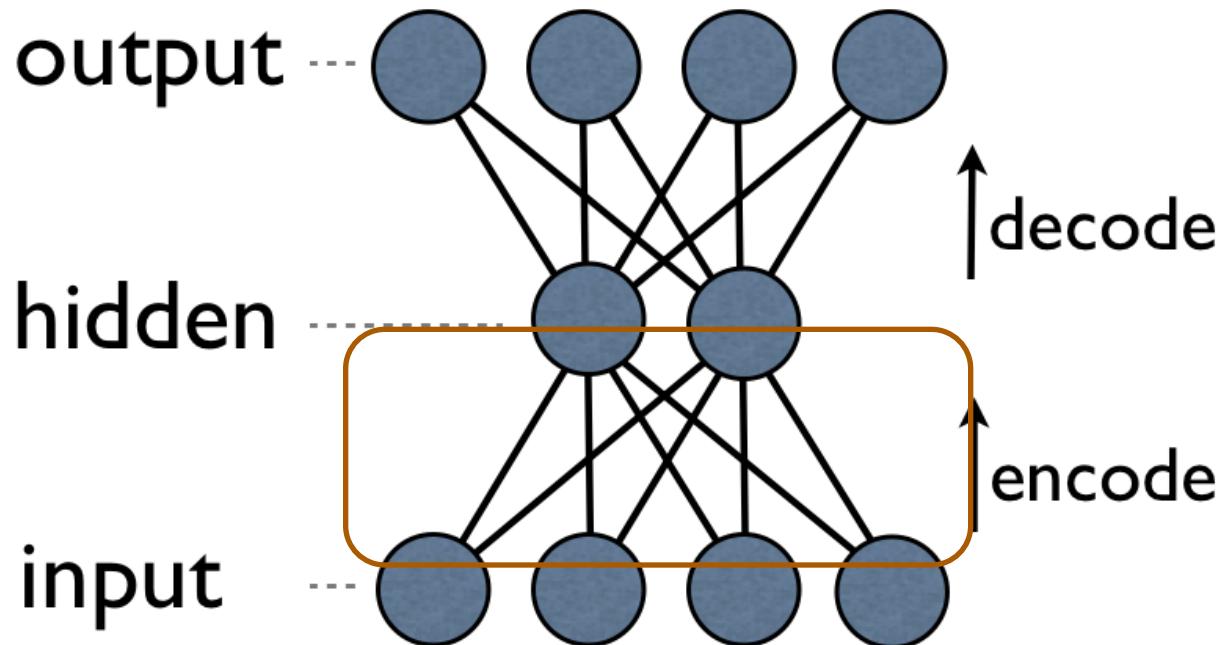
*Basically, it is forced to learn good  
features that describe what comes from  
the previous layer*

an auto-encoder-decoder is trained to reproduce the input



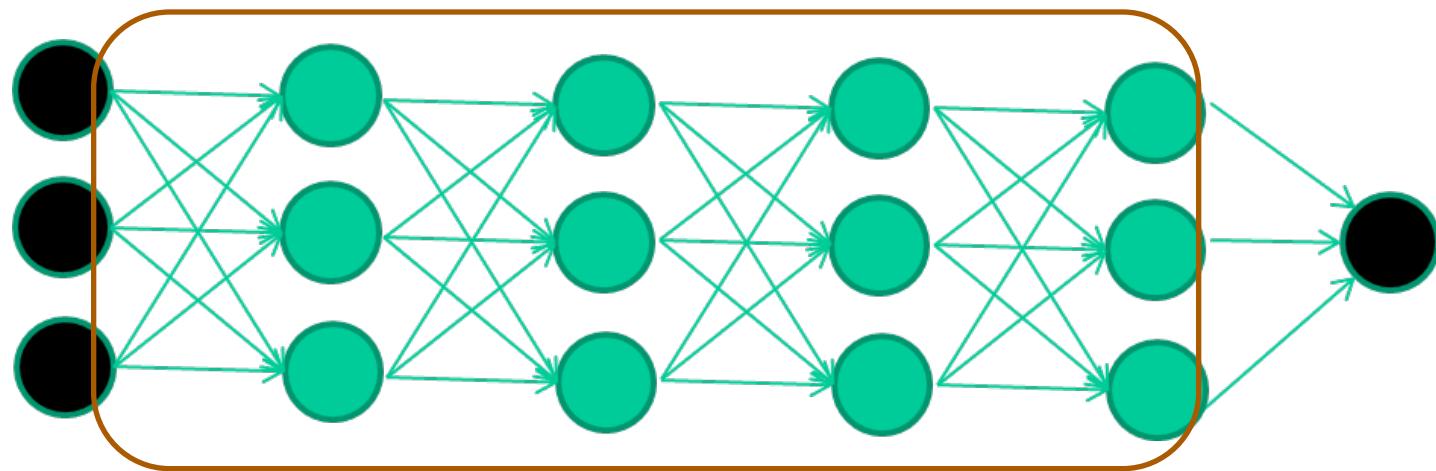
Reconstruction Loss

force the ‘hidden layer’ units to become good / reliable feature detectors



**Reconstruction Loss**

intermediate layers can each be trained to be auto encoders (or similar)



# Recent Trend (2): Convolutional Neural Networks (aka CNNs and ConvNets)

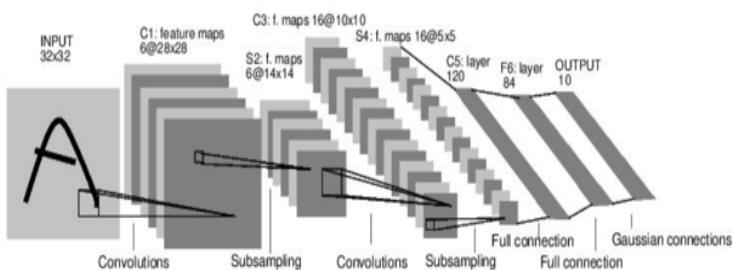


**Architectures:**  
● Convolutions

# History of ConvNets

1998

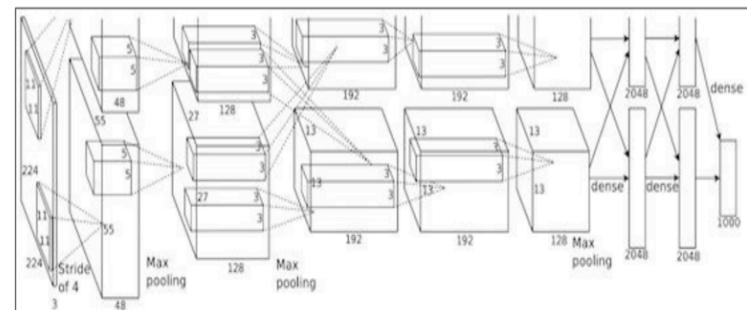
*Gradient-based learning applied to document recognition* [LeCun, Bottou, Bengio, Haffner]



LeNet-5

2012

*ImageNet Classification with Deep Convolutional Neural Networks* [Krizhevsky, Sutskever, Hinton, 2012]



“AlexNet”

# Locality and Translation Invariance



The bird occupies a local area and looks the same in different parts of an image.  
**We should construct neural nets which exploit these properties!**

# Locality and Translation Invariance

- **Locality:** objects tend to have a local spatial support
- **Translation invariance:** object appearance is independent of location
- Can define these properties since an image lies on a grid/lattice
  - ConvNet machinery applicable to other data with such properties, e.g. audio/text

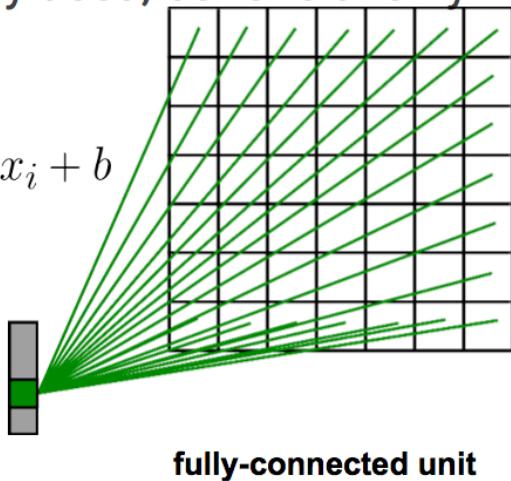
# Incorporating Assumptions: locality

Arch



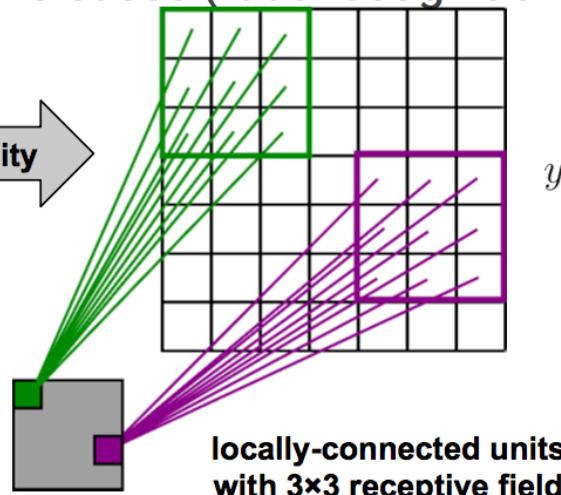
- Make **fully-connected layer** **locally-connected**
- Each unit/neuron is connected to a local rectangular area – receptive field
- Different units connected to different locations
  - output (“feature map”) lies on a grid itself
- Rarely used, beneficial only in specific cases (face recognition)

$$y = \sum_{i \in \text{image}} w_i x_i + b$$



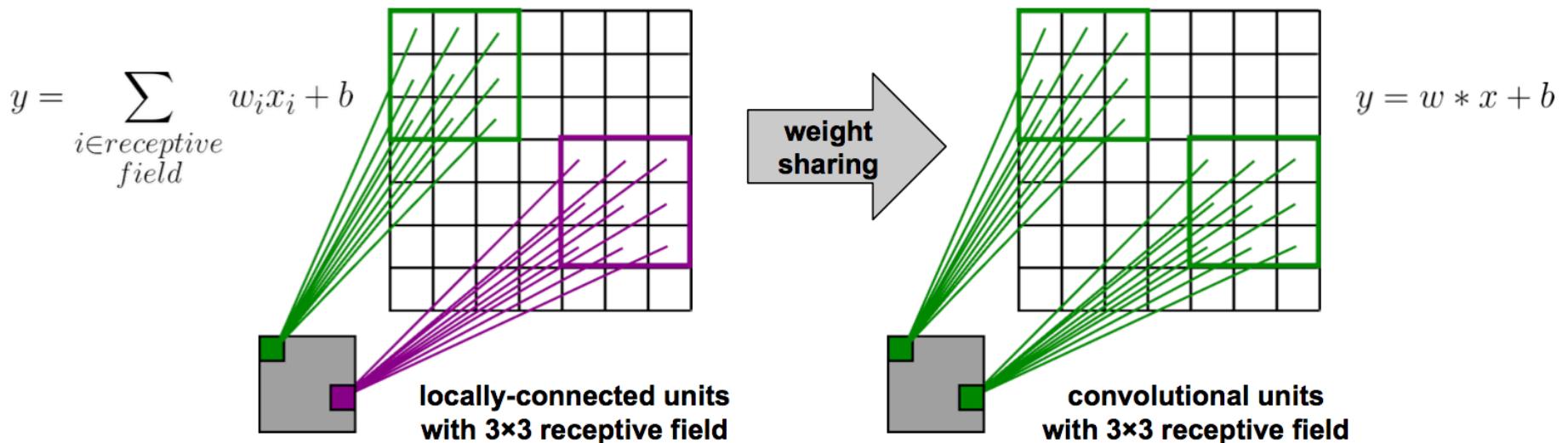
locality

$$y = \sum_{i \in \text{receptive field}} w_i x_i + b$$



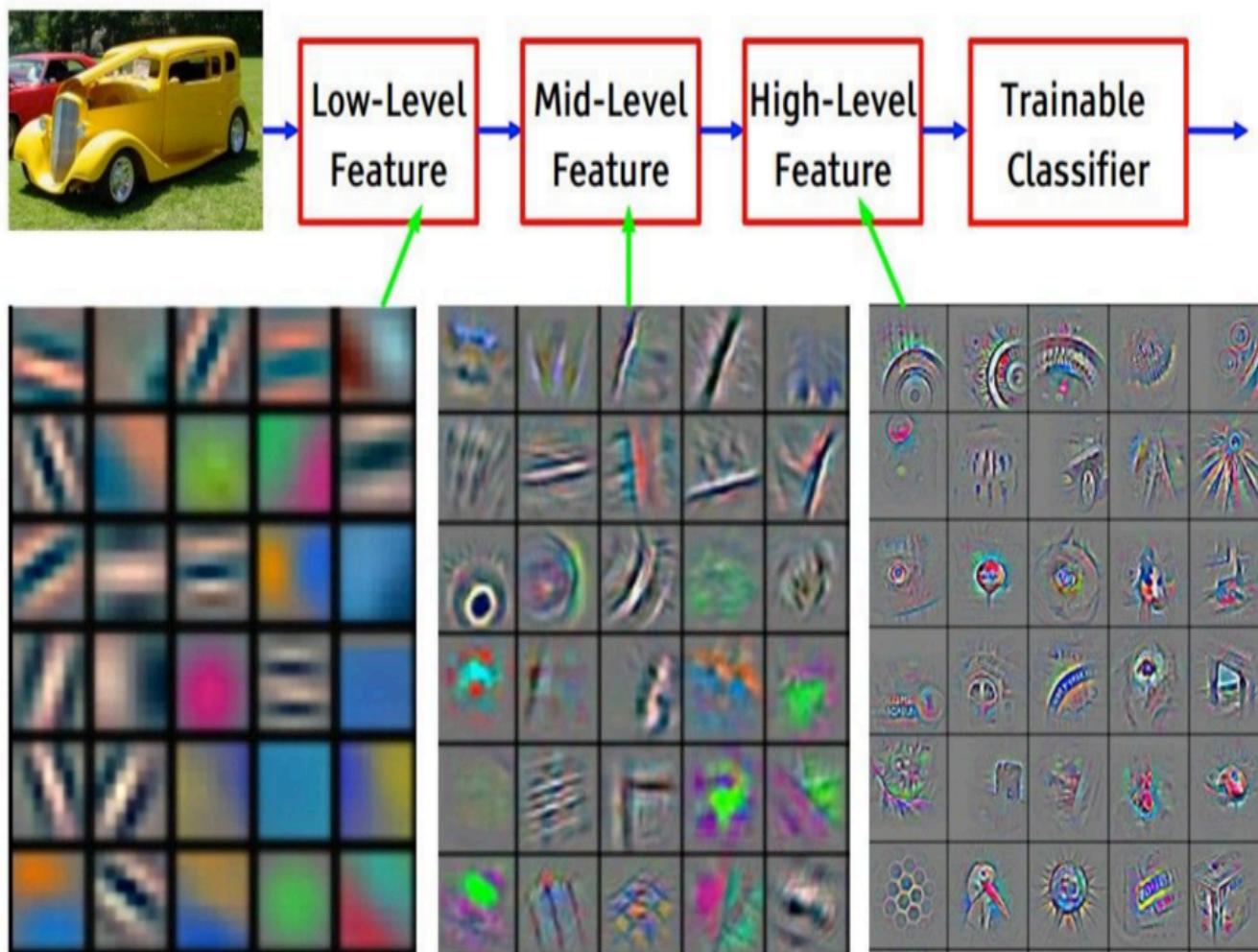
# Incorporating Assumptions: translation invariance

- **Weight sharing**
  - units connected to different locations have the same weights
  - equivalently, each unit is applied to all locations
- *Convolutional layer – locally-connected layer with weight sharing (translation invariance)*
- The weights are invariant, the output is equivariant



# Convolutional Neural Networks

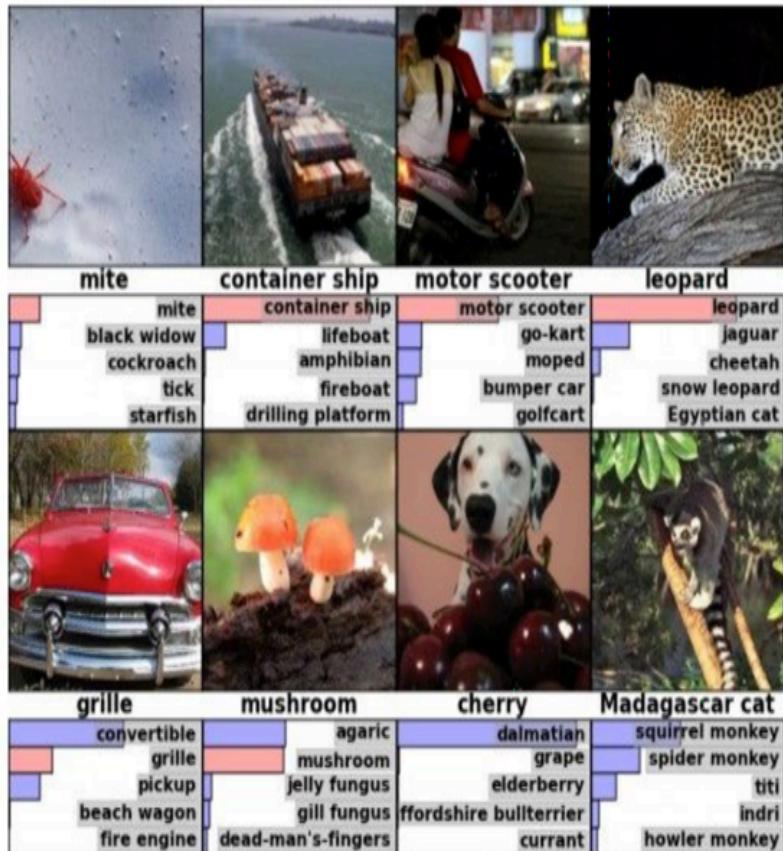
[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Fast-forward to today: ConvNets are everywhere

Classification



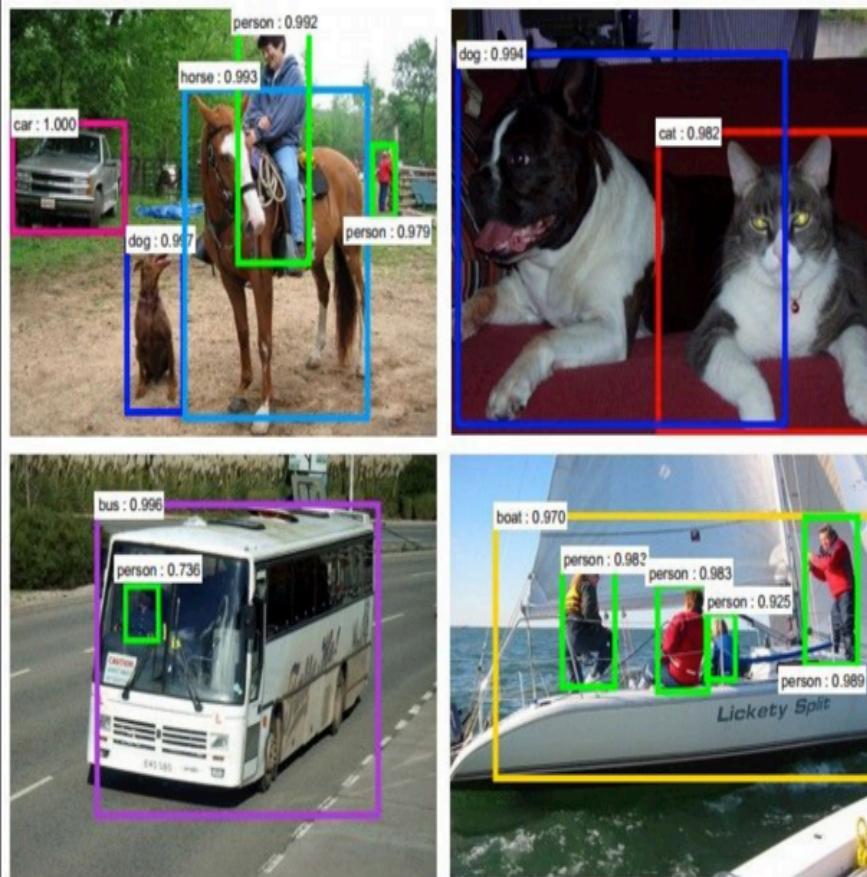
Retrieval



[Krizhevsky 2012]

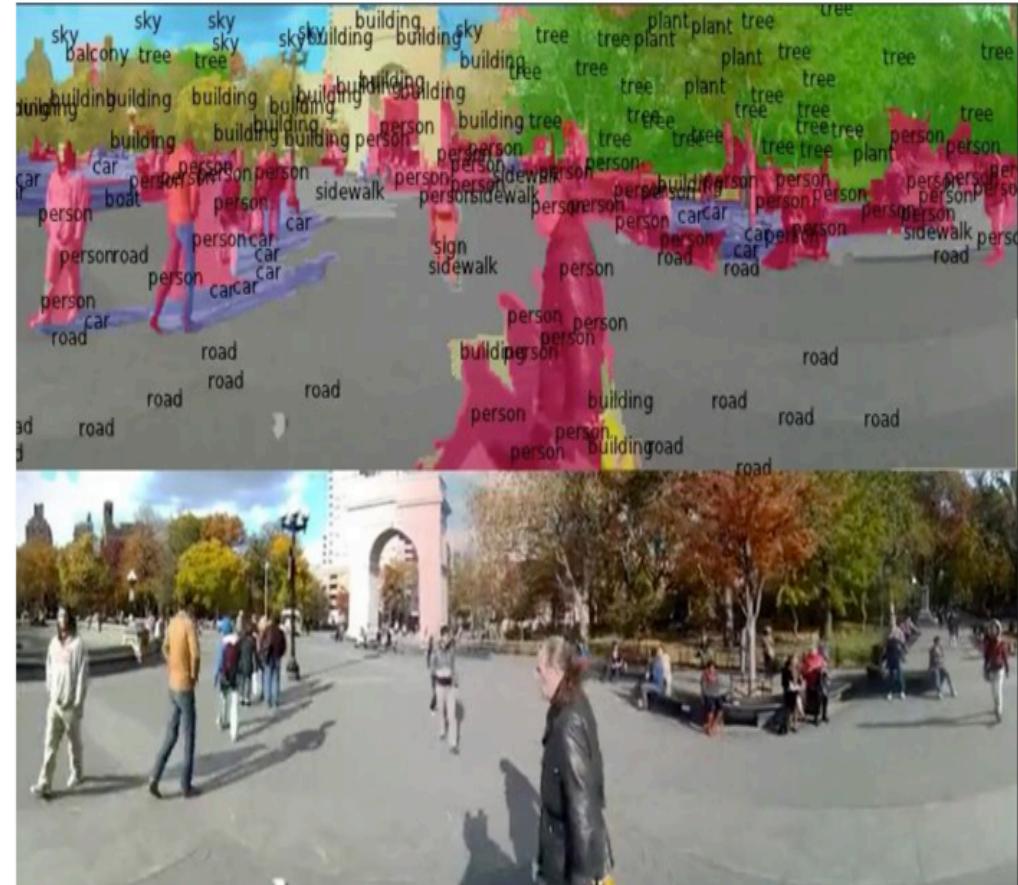
# Fast-forward to today: ConvNets are everywhere

Detection



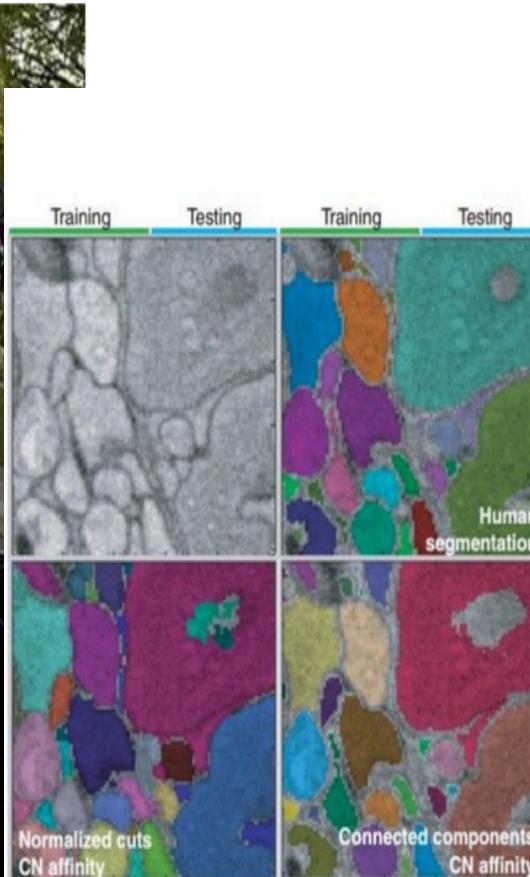
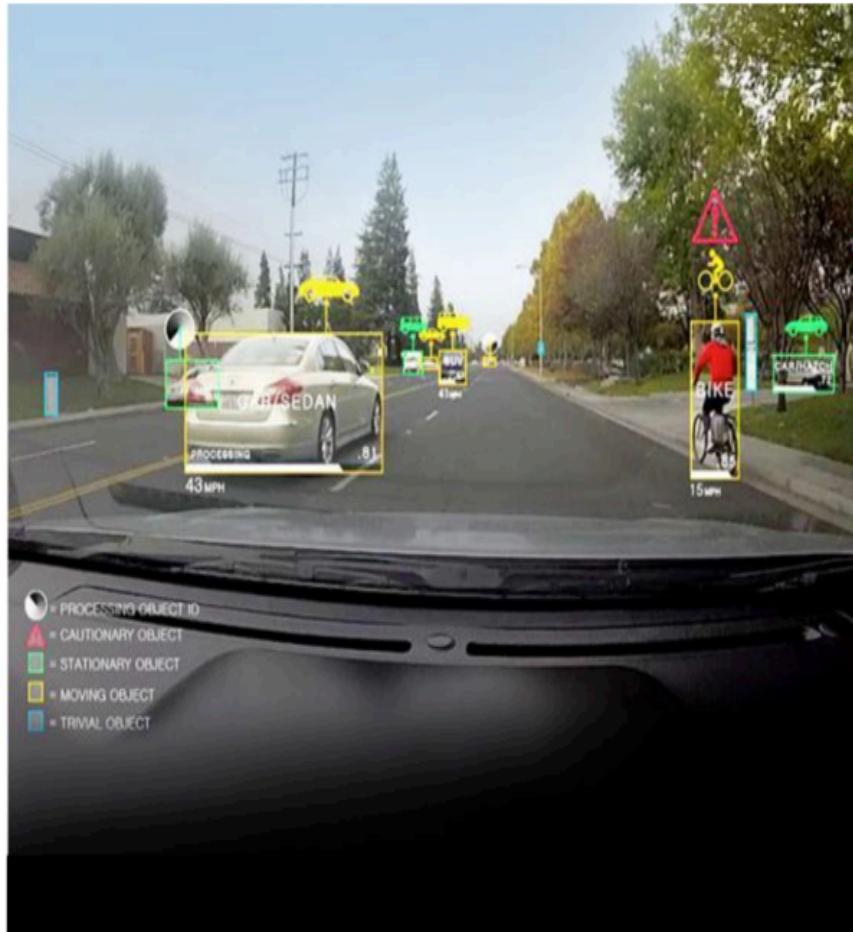
[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation

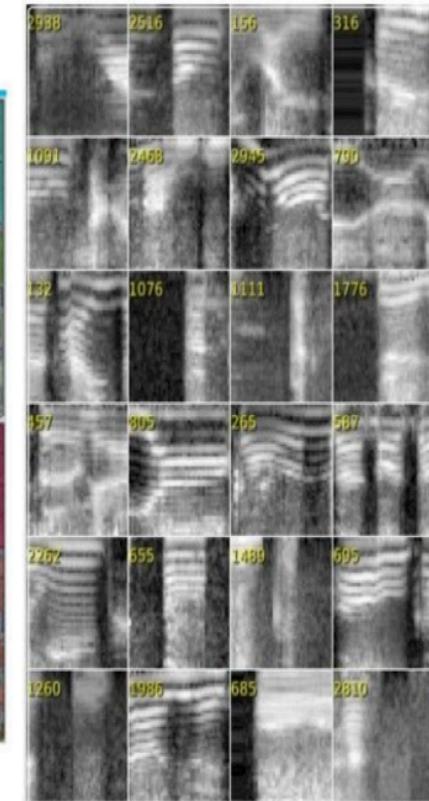


[Farabet et al., 2012]

# Fast-forward to today: ConvNets are everywhere



[Turaga et al., 2010]



self-driving cars

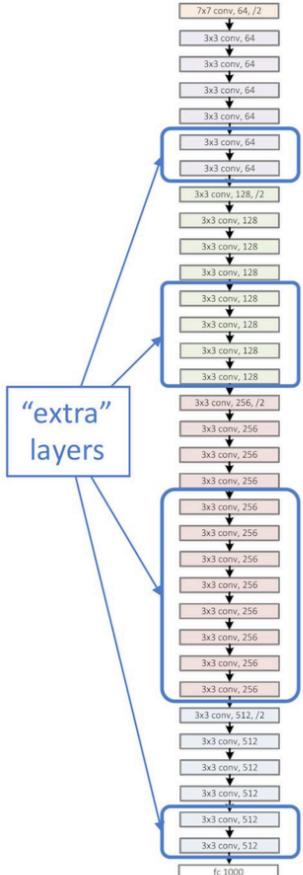
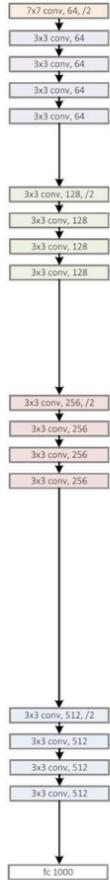
## Residual Trick:

# Residual/Skip Connections

Arch



a shallower  
model  
(18 layers)



a deeper  
counterpart  
(34 layers)

- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as **identity**
  - at least the same training error
- **Optimization difficulties**: solvers cannot find the solution when going deeper...

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

## Adaptive / Dynamic Trick:

- Diet Networks: Thin Parameters for Fat Genomics, ICLR 2017
- Dynamic Filter Networks, NIPS 2016
- Hyper Networks, ICLR 2017
- Optimal Architectures in a Solvable Model of Deep Networks, NIPS16
- AdaNet: Adaptive Structural Learning of Artificial Neural Networks, ICML17
- SplitNet: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization, ICML17
- Image Question Answering using Convolutional Neural Network with Dynamic Parameter , CVPR 2016
- Many others..

# Recent Trend (3): Recurrent Neural Networks

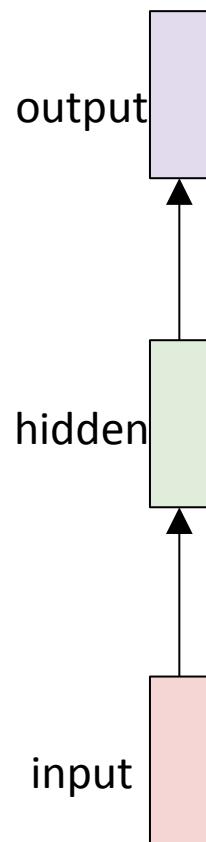


## **Architectures:**

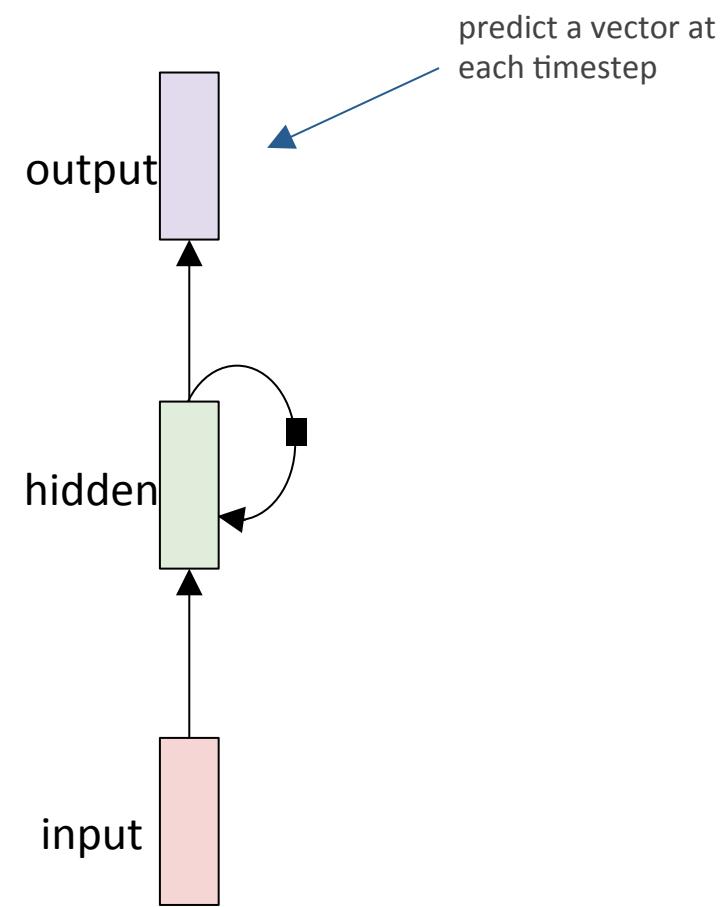
- Recurrent, over space  
and/or time.
  - + attention
- Attention-only!

# Recurrent Neural Networks (RNNs)

Traditional “Feed Forward”  
Neural Network

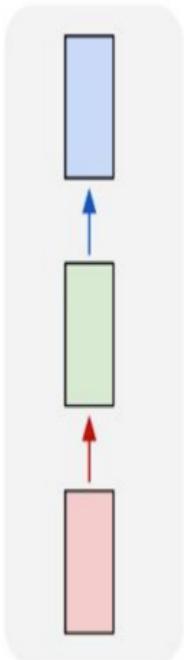


Recurrent Neural Network



# Standard “Feed-Forward” Neural Network

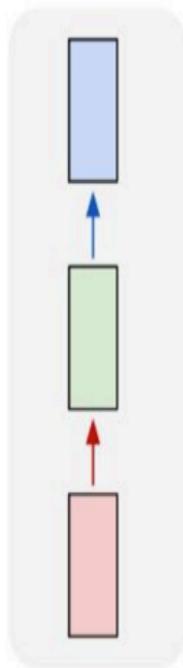
one to one



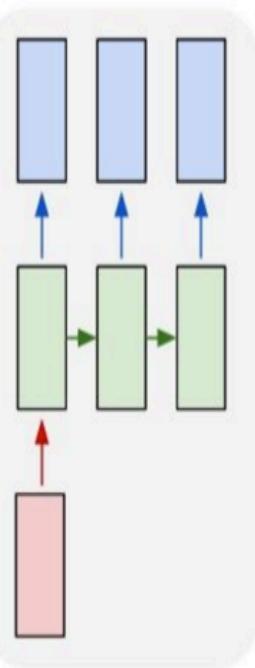
# Recurrent Neural Networks (RNNs)

RNNs can handle

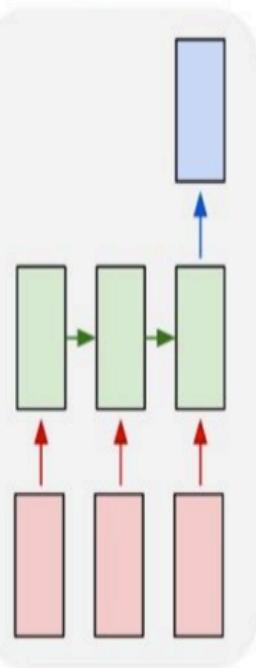
one to one



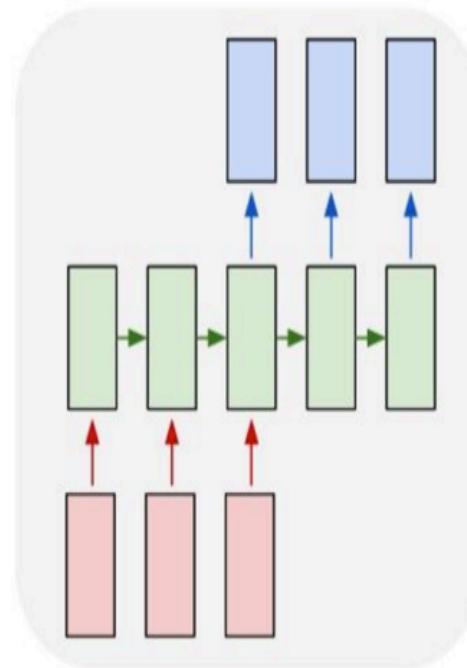
one to many



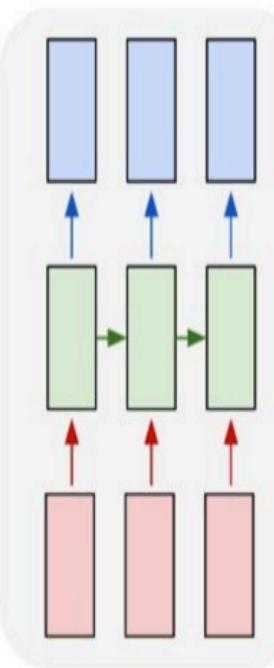
many to one



many to many



many to many

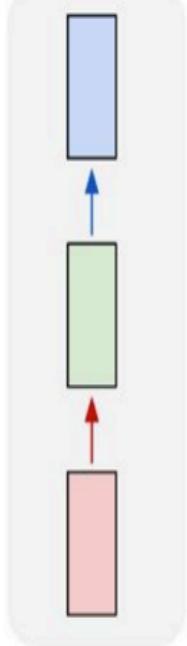


e.g. **Sentiment Classification**  
sequence of words -> sentiment

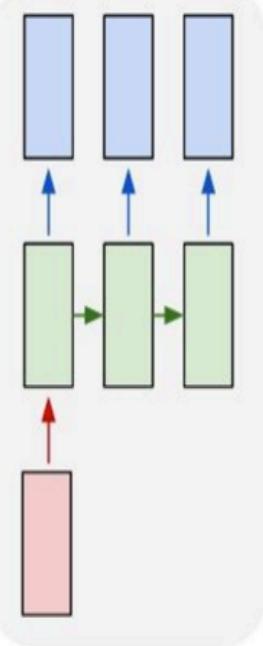
# Recurrent Neural Networks (RNNs)

RNNs can handle

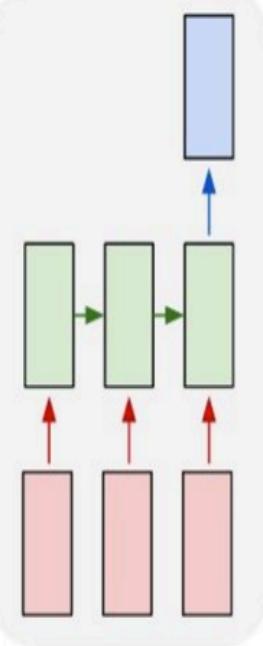
one to one



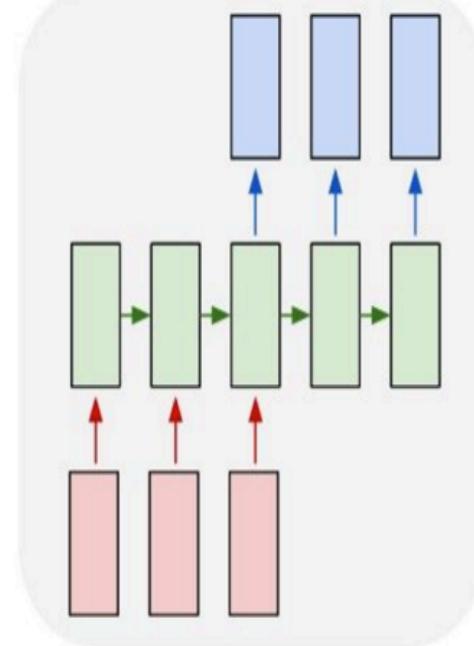
one to many



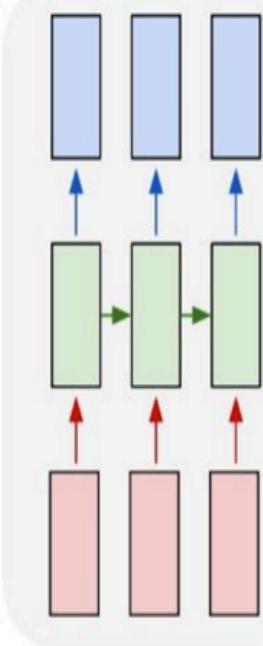
many to one



many to many

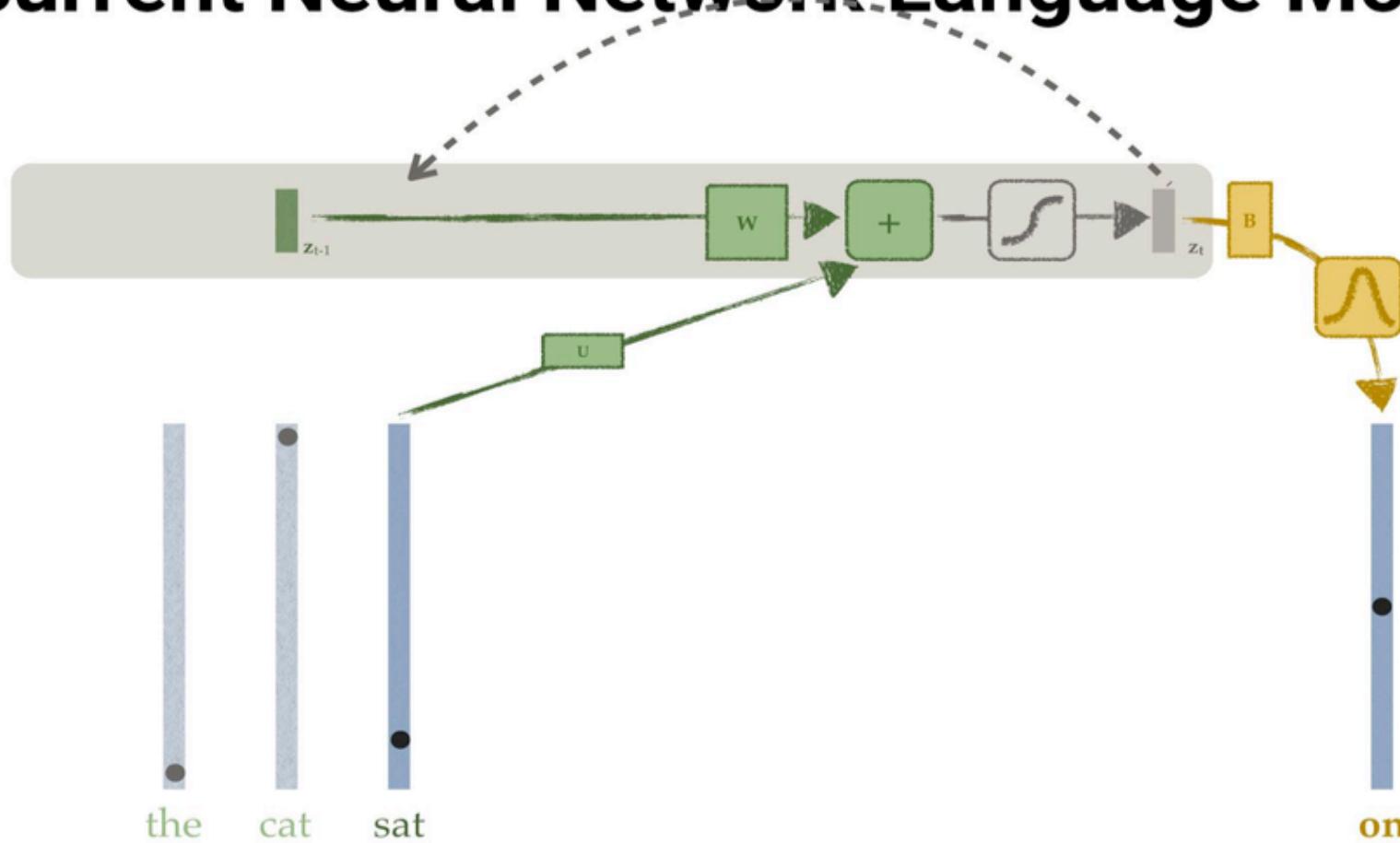


many to many

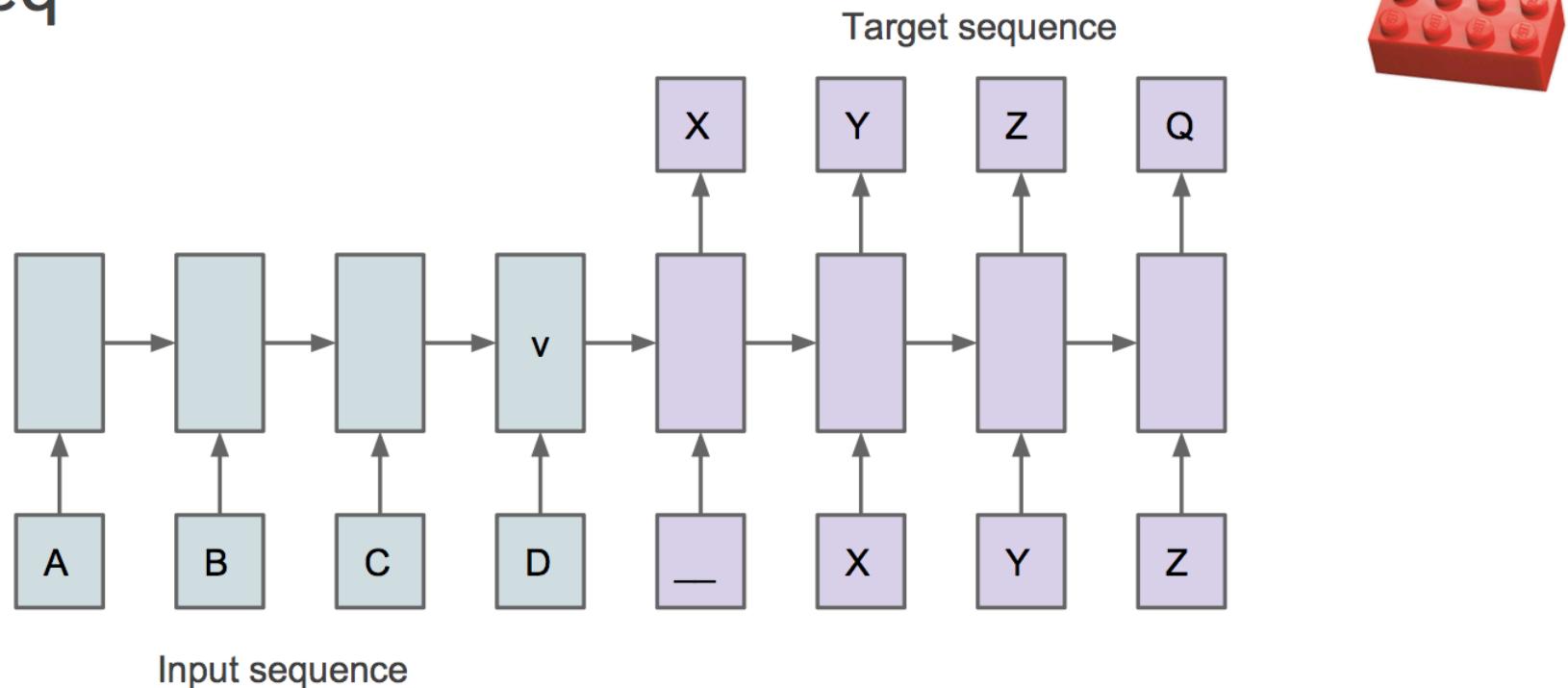


e.g. Machine Translation  
seq of words -> seq of words

# Recurrent Neural Network Language Models



# Seq2Seq

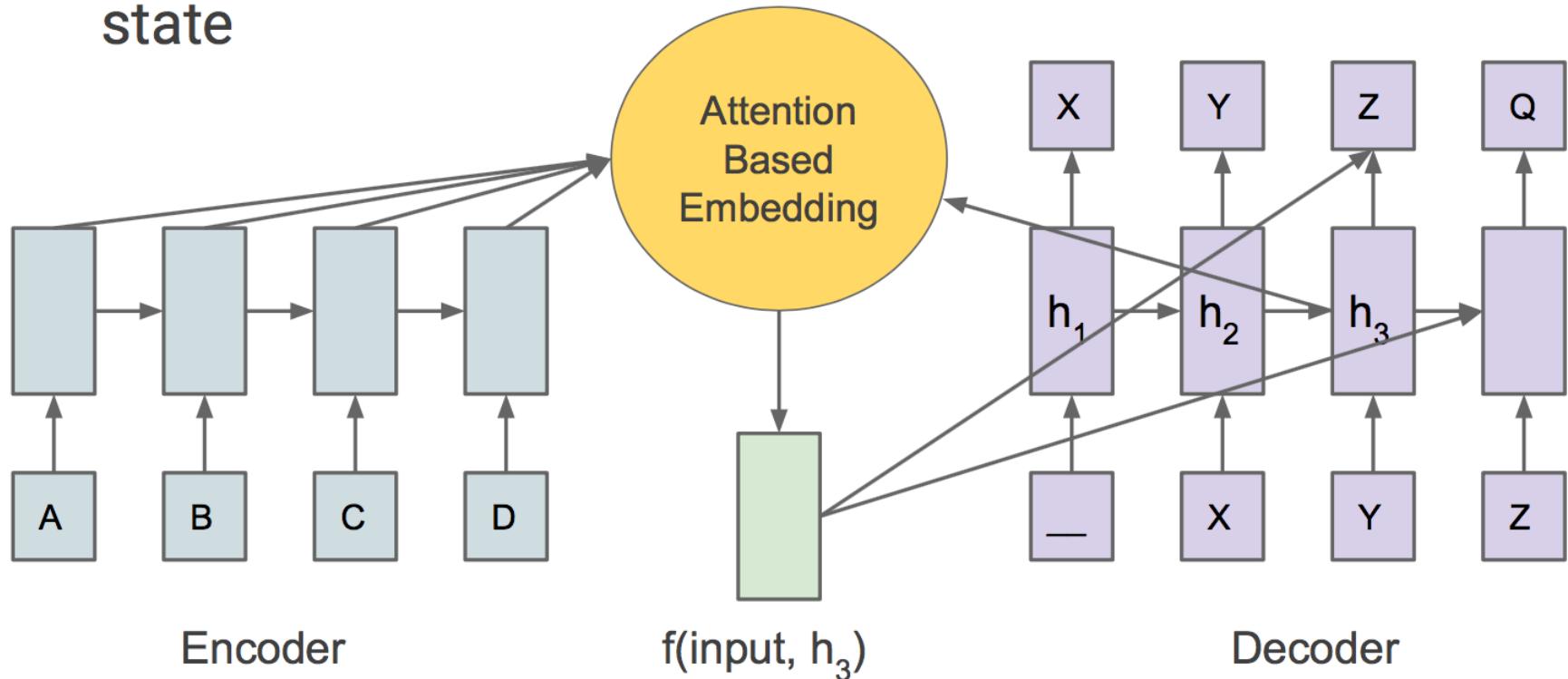


$$P(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

## Attention Trick:

# Seq2Seq with Attention

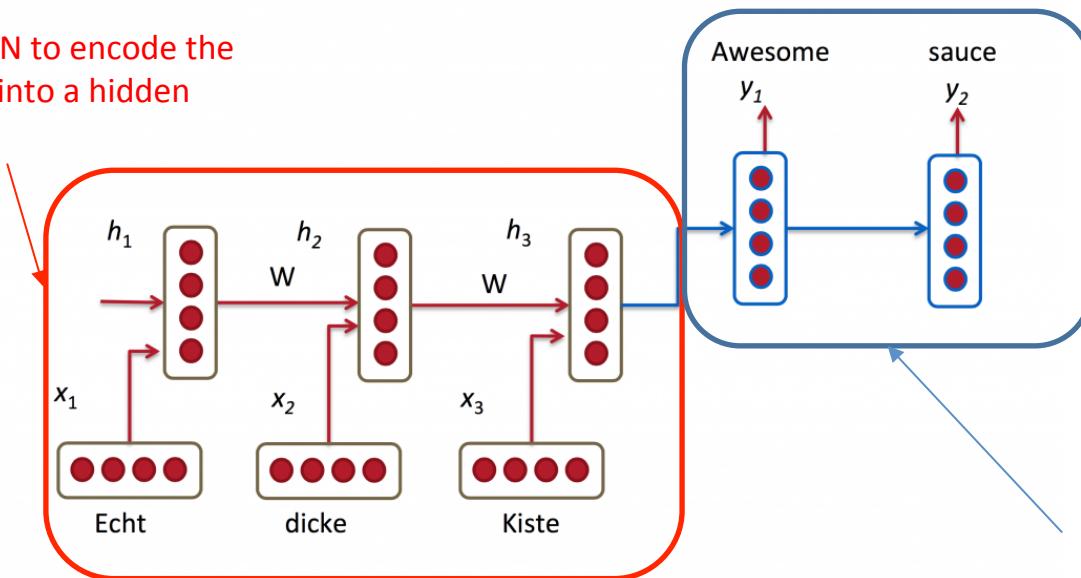
- Embedding used to predict output, and compute next hidden state



# Seq2Seq for Machine Translation

In machine translation, the input is a sequence of words in source language, and the output is a sequence of words in target language.

Encoder: An RNN to encode the input sentence into a hidden state (feature)



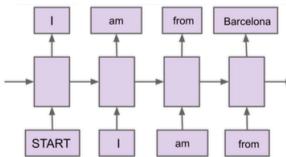
Encoder-decoder architecture for machine translation

Decoder: An RNN with the hidden state of the sentence in source language as the input and output the translated sentence

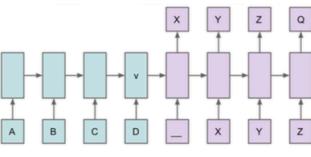
# Attention and Memory Toolbox

Arch

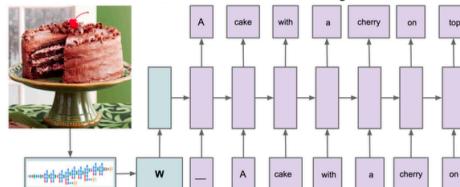
## Sequence Prediction



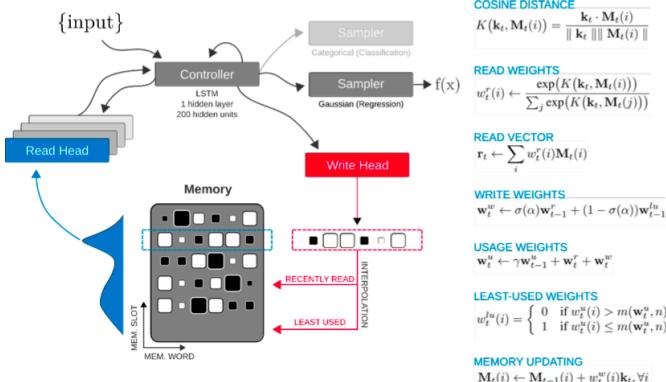
## Seq2Seq



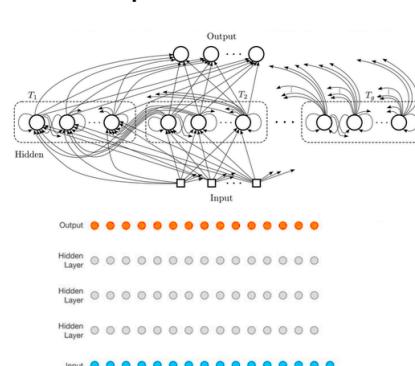
## Multimodality



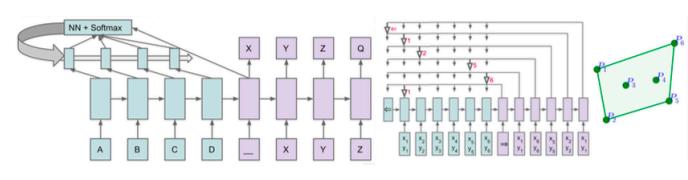
## Read/Write memories



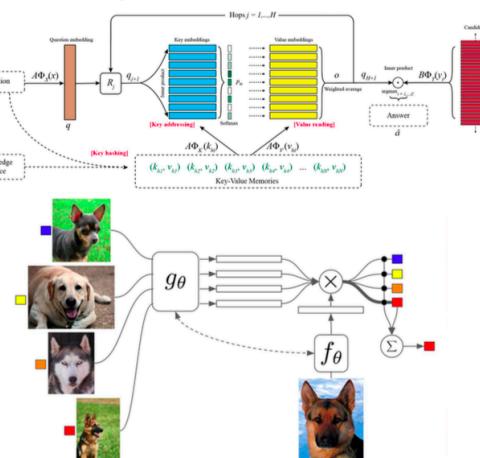
## Temporal Hierarchies



## Attention/Pointers



## Key,Value memories



## Recurrent Architectures

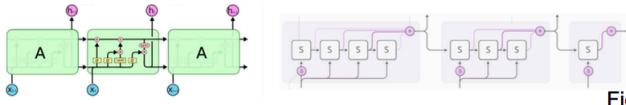


Figure credits: Jeff Dean, Chris Olah, Santoro et al 2016, Koutnik et al 2014, van den Oord et al 2016, Miller et al 2016, Vinyals et al 2016

# Recent Trend (4): Neural Architectures with Memory



## **Architectures:**

memory and multi-hop  
reasoning to perform AI  
tasks better

# Neural Architectures with Memory

- Antoine Bordes, Y-Lan Boureau, Jason Weston, **Learning End-to-End Goal-Oriented Dialog, ICLR 2017**
- Karol Kurach, Marcin Andrychowicz & Ilya Sutskever **Neural Random-Access Machines, ICLR, 2016**
- Emilio Parisotto & Ruslan Salakhutdinov **Neural Map: Structured Memory for Deep Reinforcement Learning, ArXiv, 2017**
- Oriol Vinyals, Meire Fortunato, Navdeep Jaitly **Pointer Networks, ArXiv, 2017**
- Jack W Rae et al., **Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes, ArXiv 2016**
- Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, Honglak Lee, **Control of Memory, Active Perception, and Action in Minecraft, ICML 2016**
- Wojciech Zaremba, Ilya Sutskever, **Reinforcement Learning Neural Turing Machines, ArXiv 2016**

# e.g. for Story Comprehension

Joe went to the kitchen. Fred went to the kitchen. Joe  
picked up the milk. Joe travelled to his office. Joe left the  
milk. Joe went to the bathroom.

Questions from  
Joe's angry mother:

Q1 : Where is Joe?

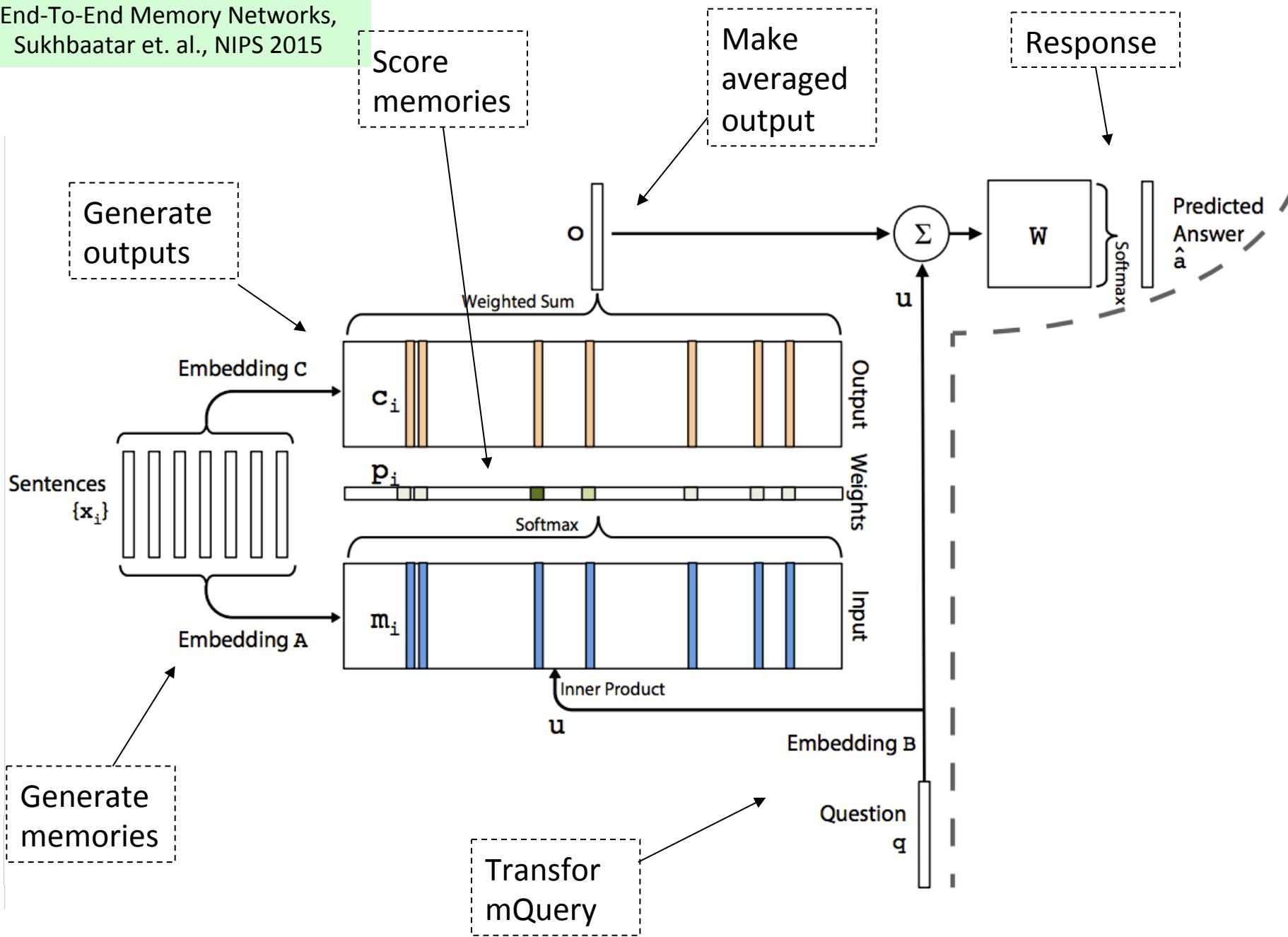
Q2 : Where is the milk now?

Q3 : Where was Joe before the office?

# Need external explicit memory for long-range reasoning

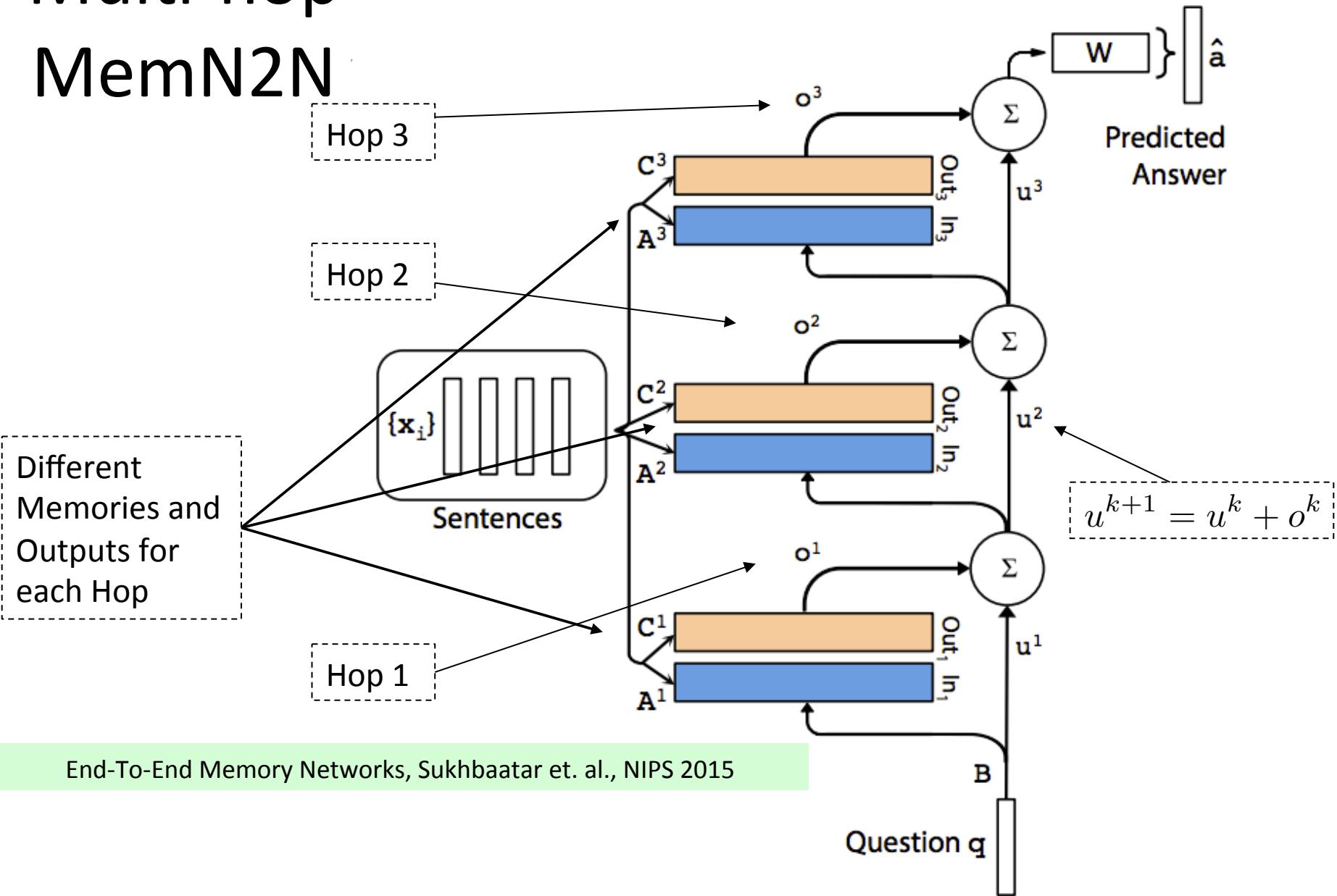
Deeper AI tasks require explicit memory and  
multi-hop reasoning over it

- RNNs have short memory
- Cannot increase memory without increasing number of parameters
- Need for compartmentalized memory
- Read/Write should be asynchronous

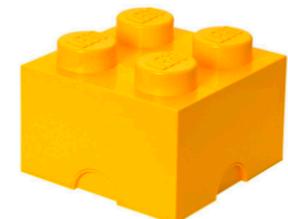
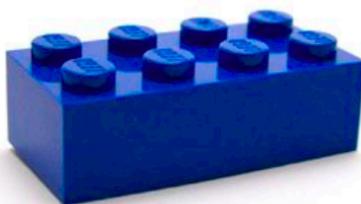


# Multi-hop

## MemN2N



# Recent Trend (5): Tasks in the form of Sequence of Decisions / Program Induction



## Inputs and Outputs:

- Discrete symbols, (e.g. the program itself)
- Program execution traces
- Program I/O pairs

These can also be mixed with perceptual data.

## Architectures:

- (Mostly) recurrent
- Sometimes including ConvNets as a visual front-end.

## Losses:

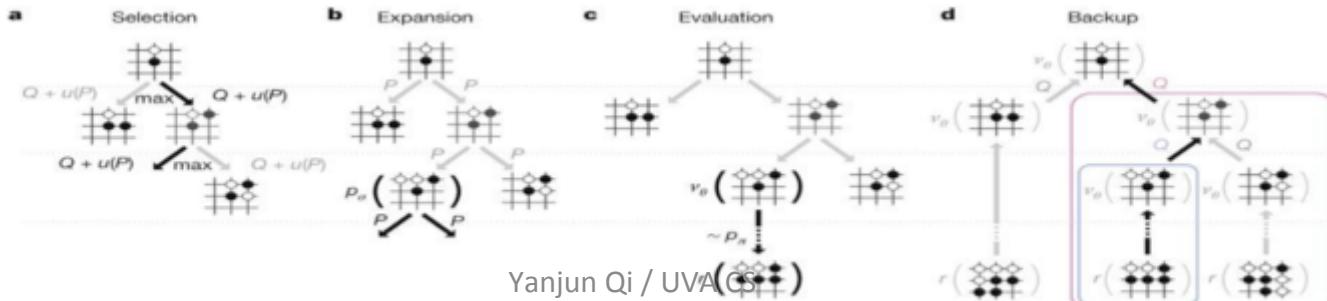
- Differentiable, predicting discrete program outputs or code itself: softmax cross entropy.
- Not differentiable: RL

# Task with Sequential Form

- Words, letters, strings, ..
- Speech, Audio, ...
- Image, Video, Sensor touches, ...
- Computer Programs , ...

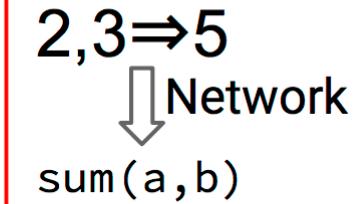
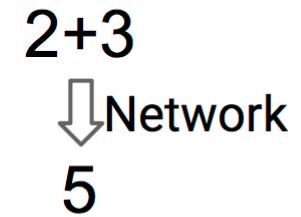
```
while (*d++ = *s++);
```

- Sequence decision making, e.g., games, RL

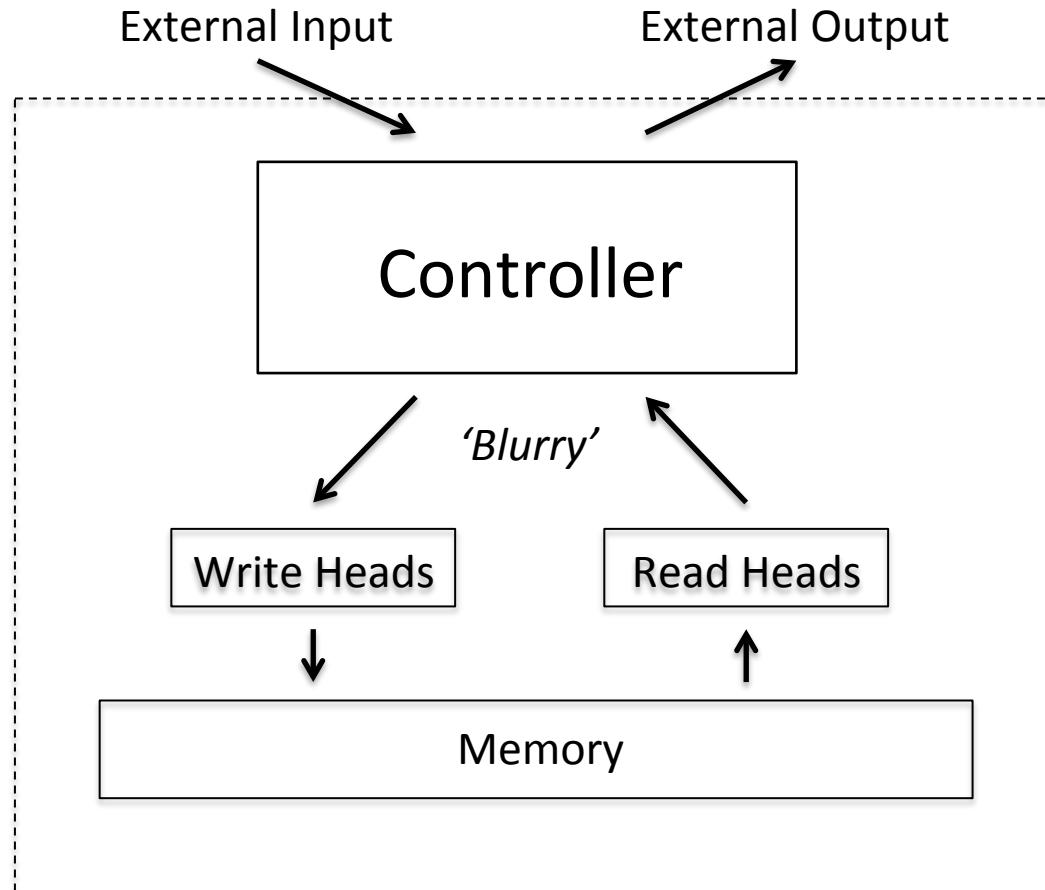


# Neural Program Induction - Research Landscape

- Neural network is the program:
  - [Learning to Execute](#), [Neural Turing Machine](#), [Neural GPU](#), [Neural RAM](#), [Neural Programmer-Interpreter](#), [Neural Task Programmer](#), [Differentiable Forth Interpreter](#)
- Neural network generates source code :
  - [DeepCoder](#), [RobustFill](#), [Neural Inductive Logic Programming](#)
- Probabilistic programming with neural networks:
  - [TerpreT](#), [Edward](#), [Picture](#)

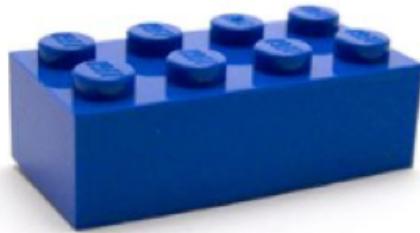


# Neural Turing Machines



Neural Turing Machines, Graves et. al., arXiv:1410.5401

# Recent Trend (6): Learning to Optimize / Learning to Search DNN architecture



**Inputs and Outputs**



**Losses**

# Neural Architecture Search with Reinforcement Learning, ICLR17

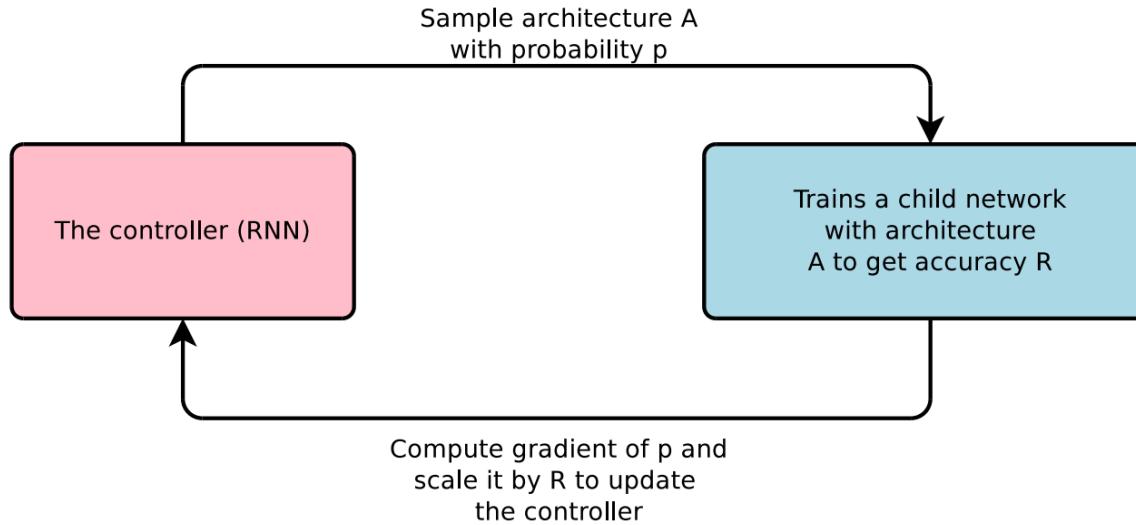
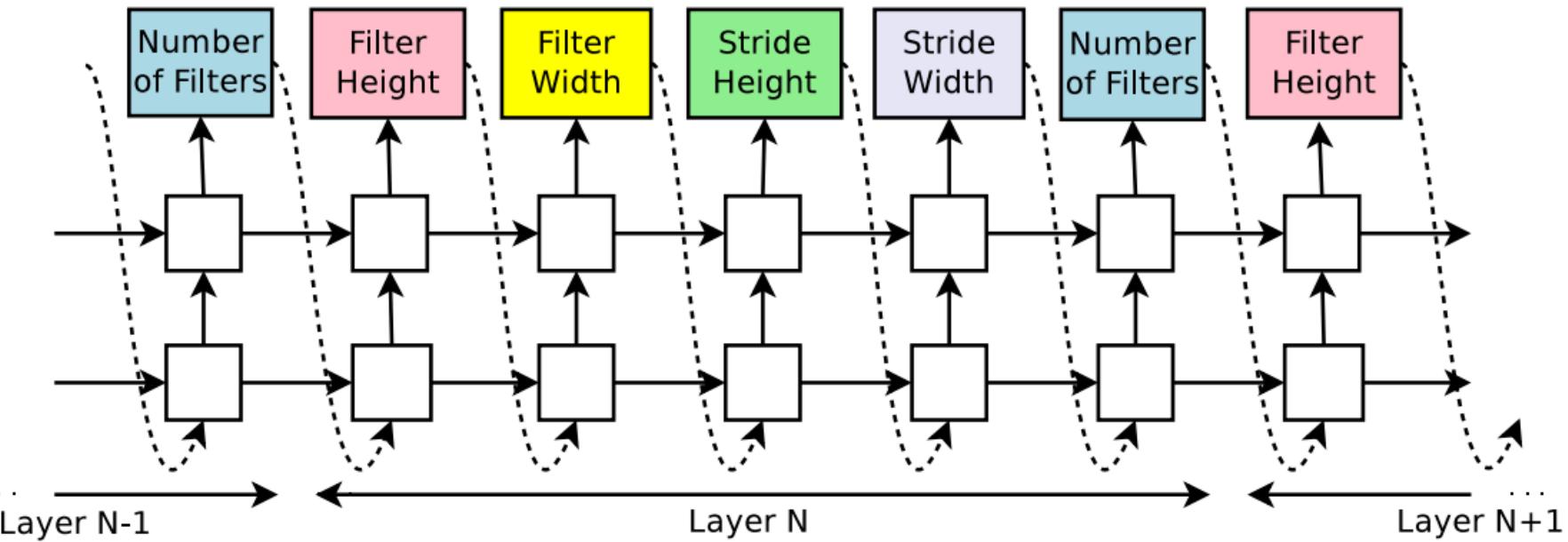


Figure 1: An overview of Neural Architecture Search.



# Neural Optimizer Search with Reinforcement Learning, ICML17

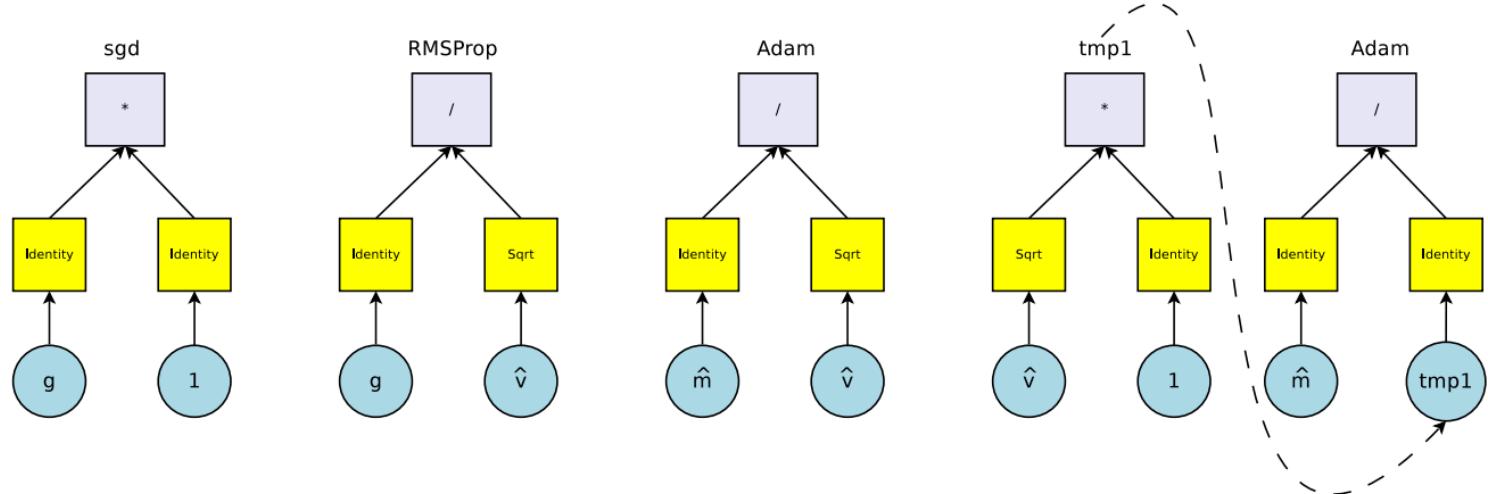


Figure 2. Computation graph of some commonly used optimizers: SGD, RMSProp, Adam. Here, we show the computation of Adam in 1 step and 2 steps. Blue boxes correspond to input primitives or temporary outputs, yellow boxes are unary functions and gray boxes represent binary functions.  $g$  is the gradient,  $\hat{m}$  is the bias-corrected running estimate of the gradient, and  $\hat{v}$  is the bias-corrected running estimate of the squared gradient.

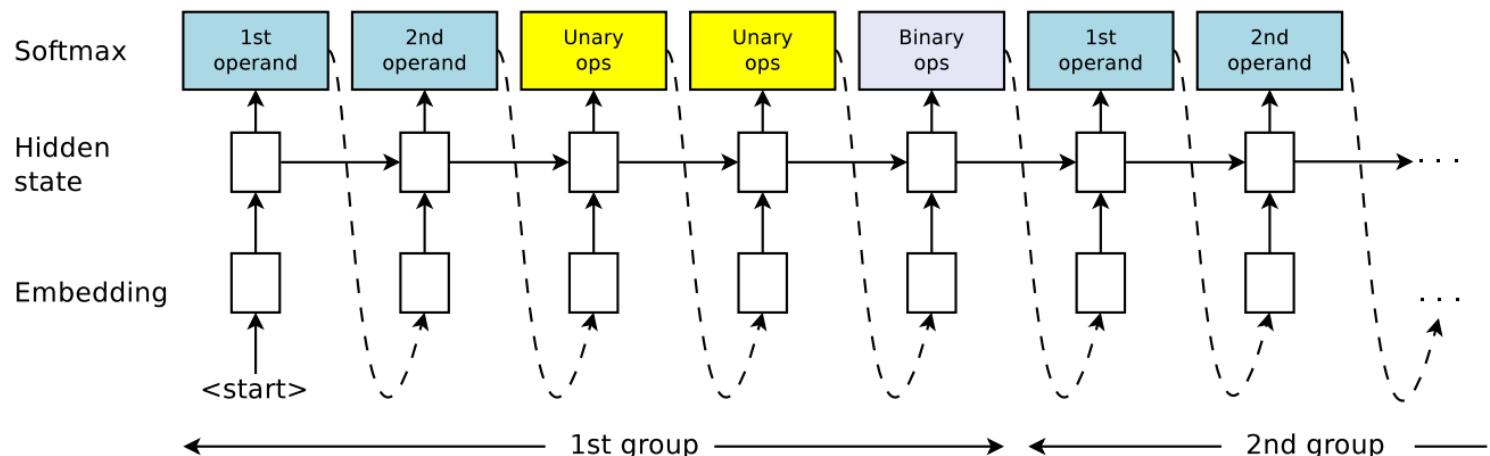


Figure 3. Overview of the controller RNN. The controller iteratively selects subsequences of length 5. It first selects the 1st and 2nd operands  $op_1$  and  $op_2$ , then 2 unary functions  $u_1$  and  $u_2$  to apply to the operands and finally a binary function  $b$  that combines the outputs of the unary functions. The resulting  $b(u_1(op_1), u_2(op_2))$  then becomes an operand that can be selected in the subsequent group of predictions or becomes the update rule. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

# Recent Trend (7): Learning to Learn

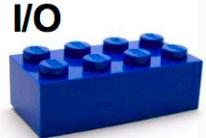


**Inputs and Outputs**

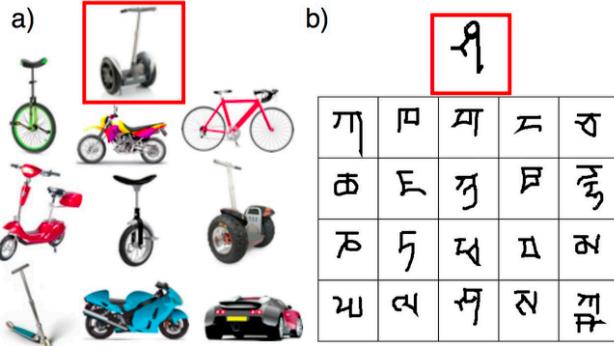


**Losses**

# Learning to Learn



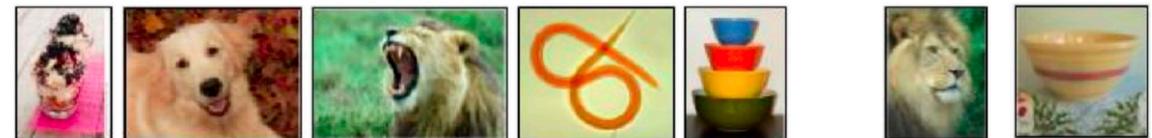
- What is Meta Learning / Learning to Learn?
  - Go beyond train/test from same distribution.
  - Task between train/test changes, so model has to “learn to learn”
- Datasets



Lake et al,  
2013, 2015

## Image recognition

Given 1 example of 5 classes:

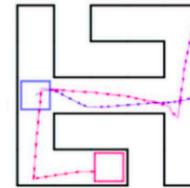


Mini-Imagenet dataset (Vinyals et al. '16)

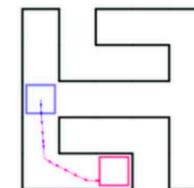
Classify new examples

## Reinforcement learning

Given a small amount of experience



Solve a new task



How? learn to learn many other tasks

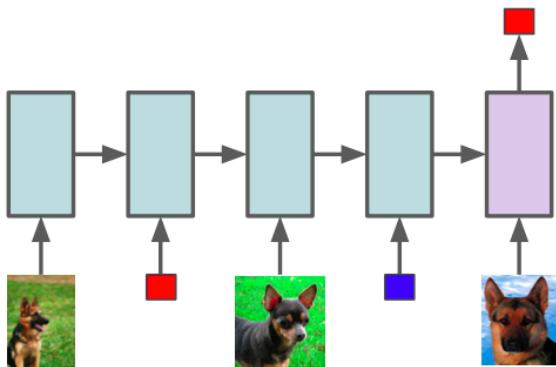
Chelsea Finn, UC Berkeley

fig. from Duan et al. '17

# Learning to Learn

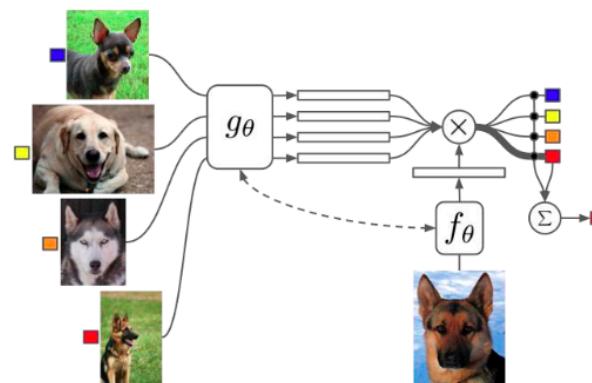


## Model Based



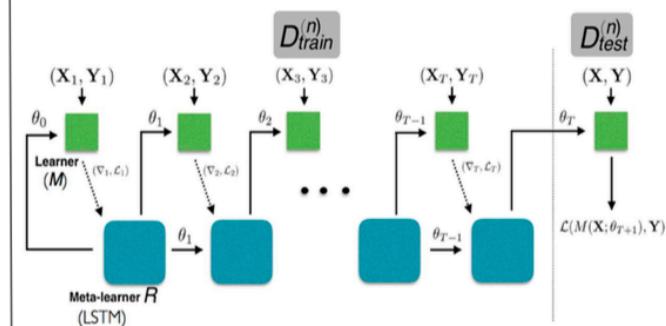
- Santoro et al. '16
- Duan et al. '17
- Wang et al. '17
- Munkhdalai & Yu '17
- Mishra et al. '17

## Metric Based



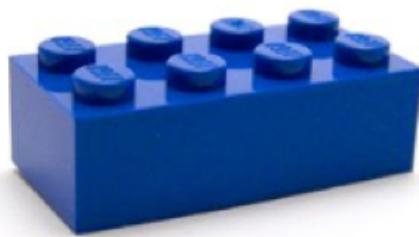
- Koch '15
- Vinyals et al. '16
- Snell et al. '17
- Shyam et al. '17

## Optimization Based



- Schmidhuber '87, '92
- Bengio et al. '90, '92
- Hochreiter et al. '01
- Li & Malik '16
- Andrychowicz et al. '16
- Ravi & Larochelle '17
- Finn et al '17

# Recent Trend (8): Variants of Input, e.g., Graphs, Trees



**Inputs and Outputs**

# Geometric Deep Learning on Graphs and Manifolds,

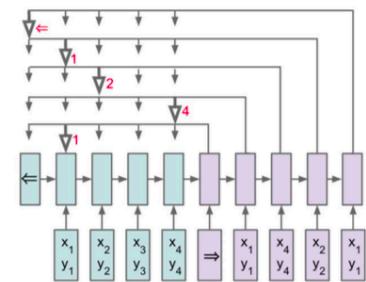
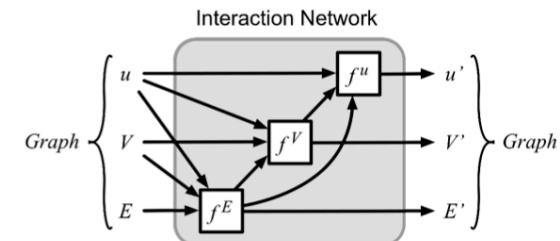
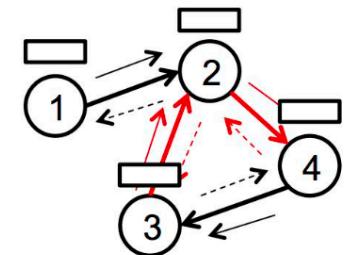
## NIPS 2017 Tutorial

Graph Nets (GNs) are a class of models that:

- Use graphs as inputs and/or outputs and/or latent representation
- Manipulate graph-structured representations
- Reflect relational structure
- Share model components across entities and relations

Examples include:

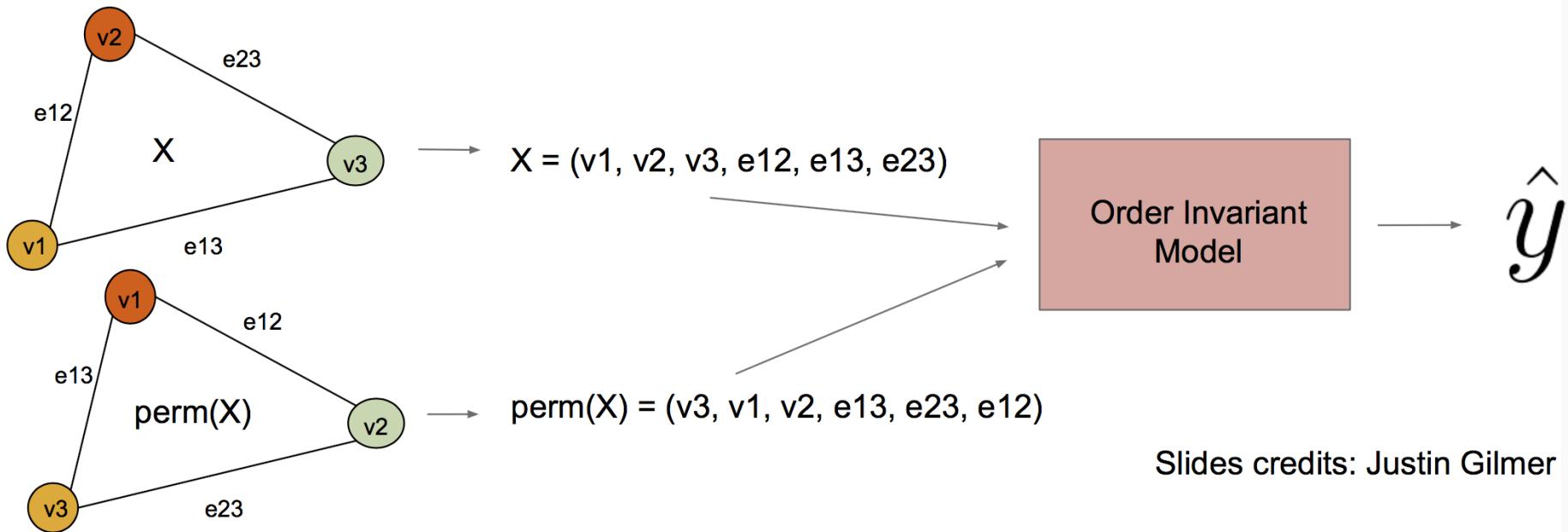
- Graph Neural Networks (*Scarselli et al 07; 08*)
- Recursive Neural Networks (*Goller et al 96*)
- Pointer Networks (*Vinyals et al 2015*)
- Graph Convolutional Networks (*Bruna et al 2013; Duvenaud et al 15; Henaff et al 15; Kipf & Welling 16; Defferrard et al 17*)
- Gated Graph Neural Networks (*Li et al 15*)
- Interaction Networks (*Battaglia et al 2016; Raposo et al 2017;*)
- Message Passing Networks (*Gilmer et al. 2017*)



# Inductive Bias for Graphs

Arch

- If we have a graph on N nodes, there are  $N!$  possible orderings of the nodes.
- Ideally want a model invariant to the order of nodes.



Slides credits: Justin Gilmer

# Recent Trend (9): Deep Generative Models: Autoregressive Kind

# Why Generative Models?

- Excellent test of ability to use high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
  - Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

(Goodfellow 2016)

# Generative models - Research Landscape

- Latent variable models ([VAE](#), [DRAW](#))
- Implicit ([GAN](#), [GMMN](#), [Progressive GAN](#))
- Transform ([NICE](#), [IAF](#), [Real NVP](#))
- **Autoregressive** ([NADE](#), [MADE](#), [RIDE](#), [PixelCNN](#), [WaveNet](#))

UAI 2017 [Tutorial](#) on Deep Generative Models.

NIPS 2016 [Tutorial](#) on Generative Adversarial Networks

# Autoregressive Models

$$P(x; \theta) = \prod_{n=1}^N P(x_n | x_{<n}; \theta)$$

- Each factor can be parametrized by  $\theta$ , which can be shared.
- The variables can be arbitrarily ordered and grouped, as long as the ordering and grouping is consistent.

I/O



# Modeling Text

The cat sat on the mat

(word-level)

Shorter sequences and  
dependencies,  
semantically meaningful  
units, many UNK

The\_cat\_sat\_on\_the\_mat

(character-level)

Long sequences and  
dependencies,  
semantically not  
meaningful units, no UNK

The\_cat\_s a t \_o n \_t h e \_m a t

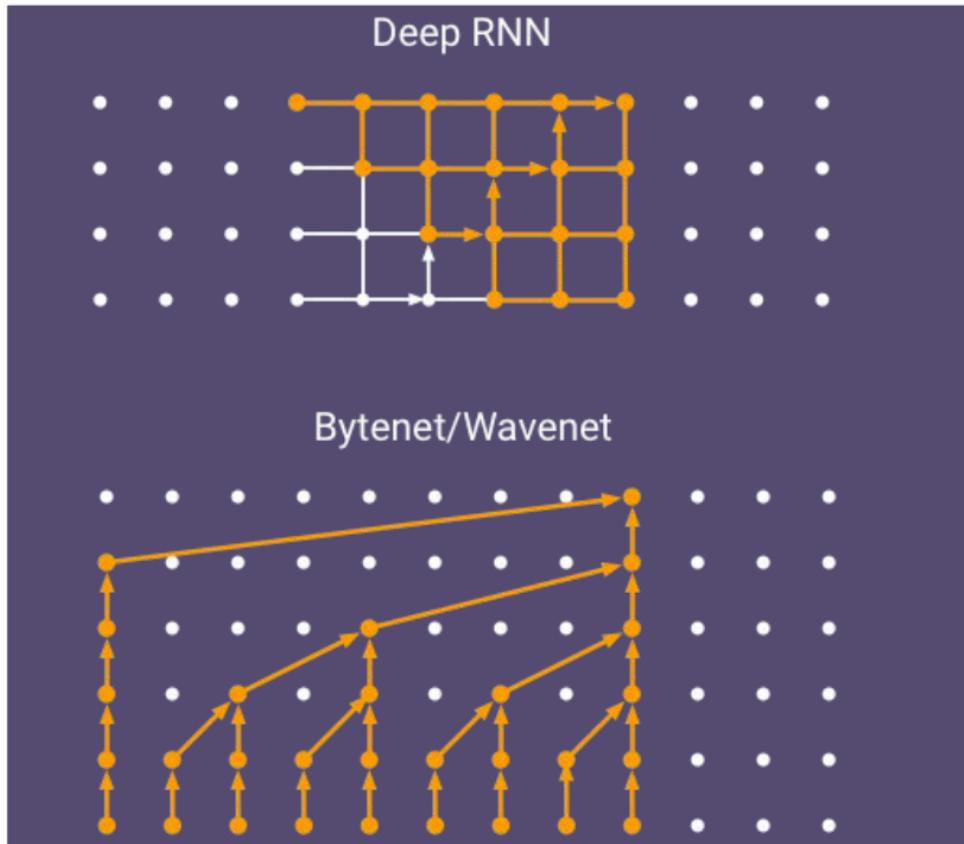
(mixed)

(byte level)

(bit level)



# Recurrent versus Causal Convolutional Nets



- The architecture is parallelizable along the time dimension (during training or scoring)
- Easy access to many states from the past

# Recent Trend (10): Generative Adversarial Networks (GAN)



**Architectures:**

- Convolutions



**Losses**



MIT  
Technology  
Review

### Dueling Neural Networks

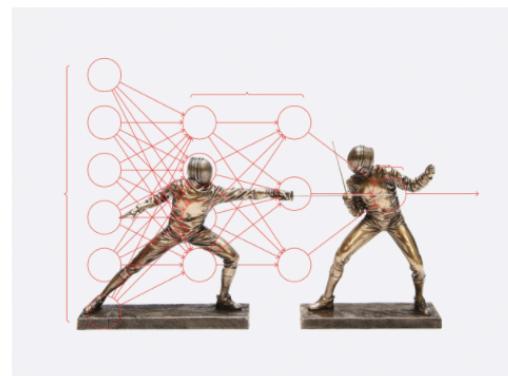
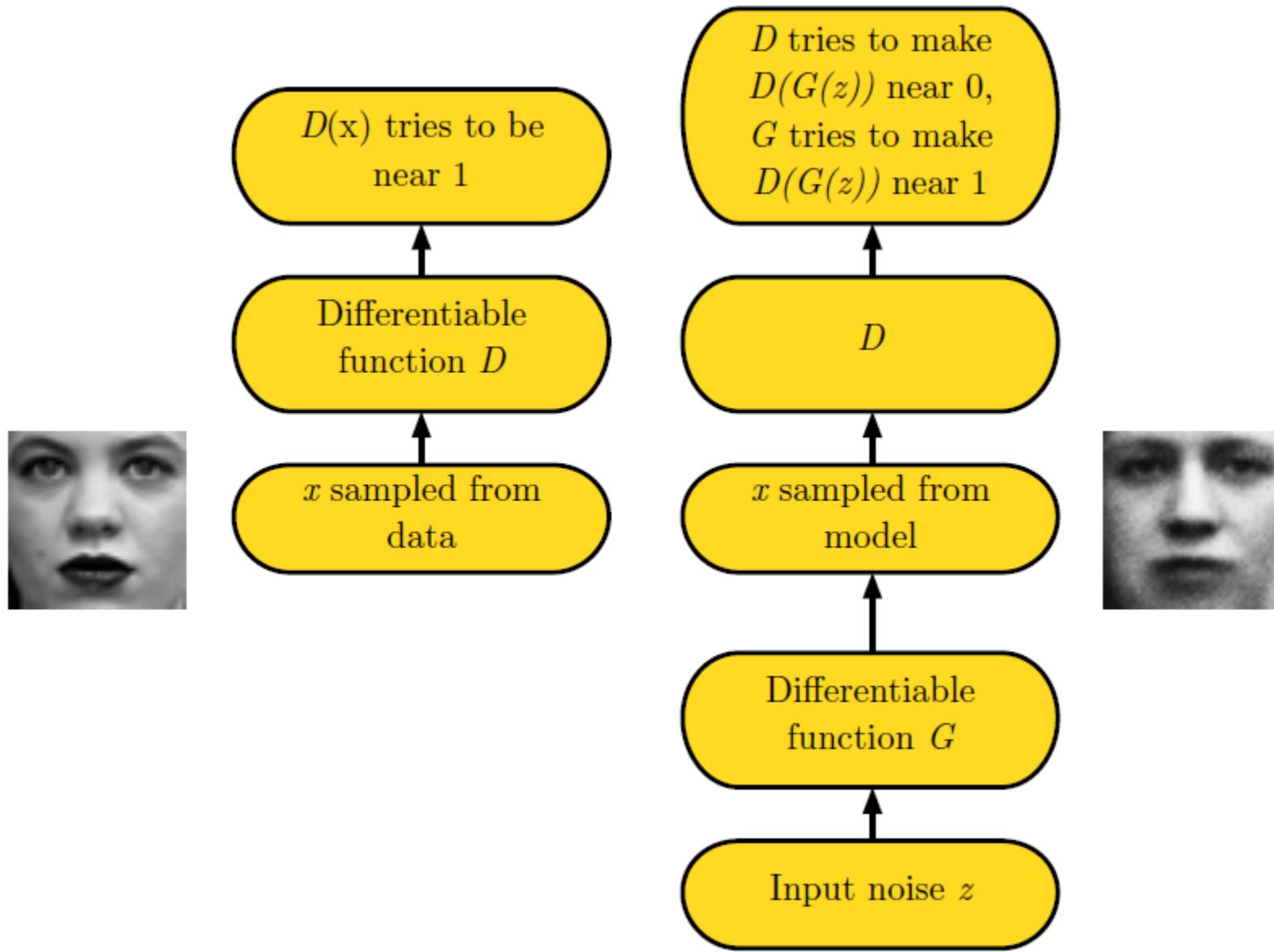


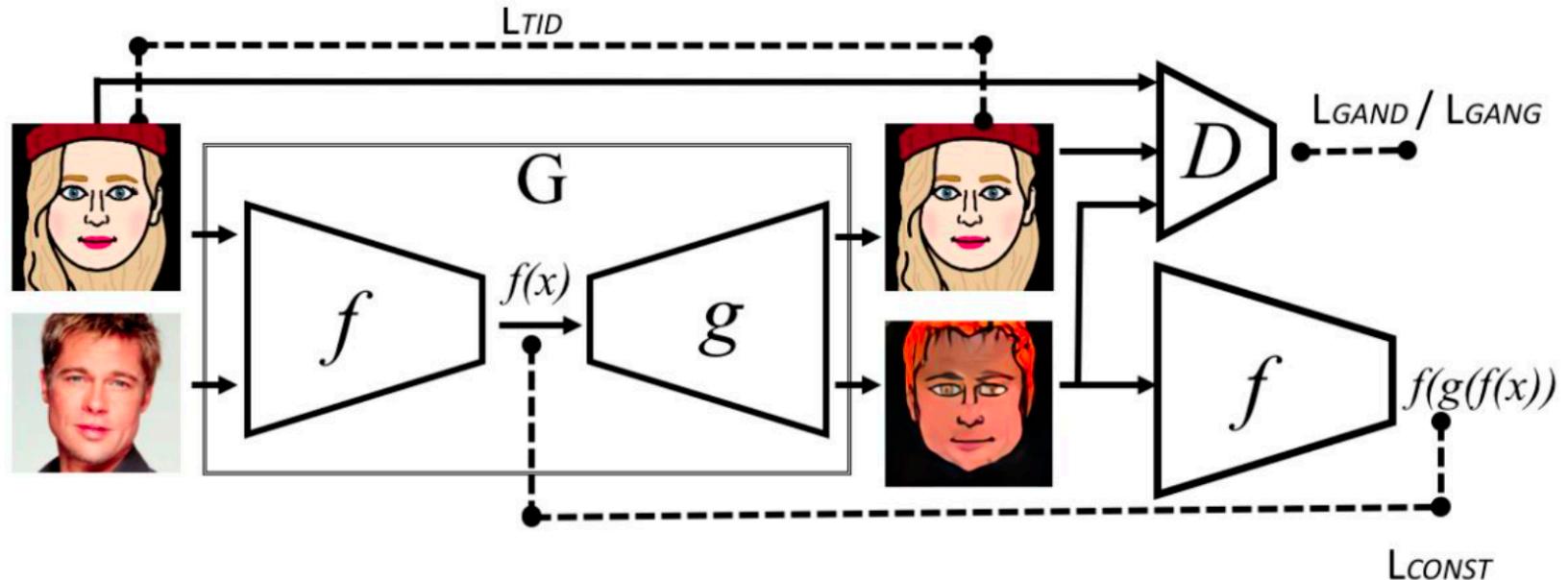
ILLUSTRATION BY DEREK BRAHNEY | DIAGRAM COURTESY OF MICHAEL NIELSEN, "NEURAL NETWORKS AND DEEP LEARNING", DETERMINATION PRESS, 2015

# Adversarial Nets Framework



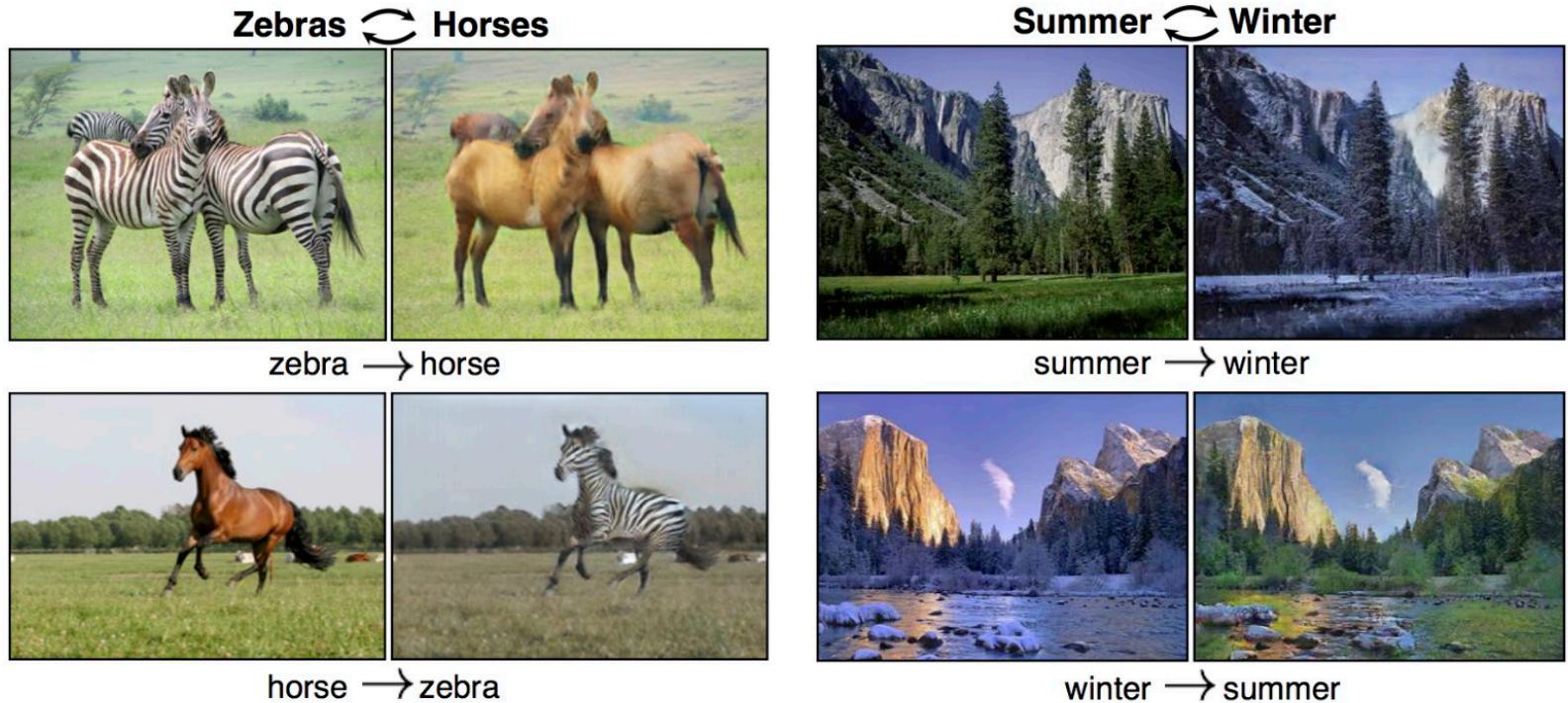


# Unsupervised cross-domain image generation



1. Taigmen et al. "Unsupervised Cross-domain image generation". In ICLR 2017.

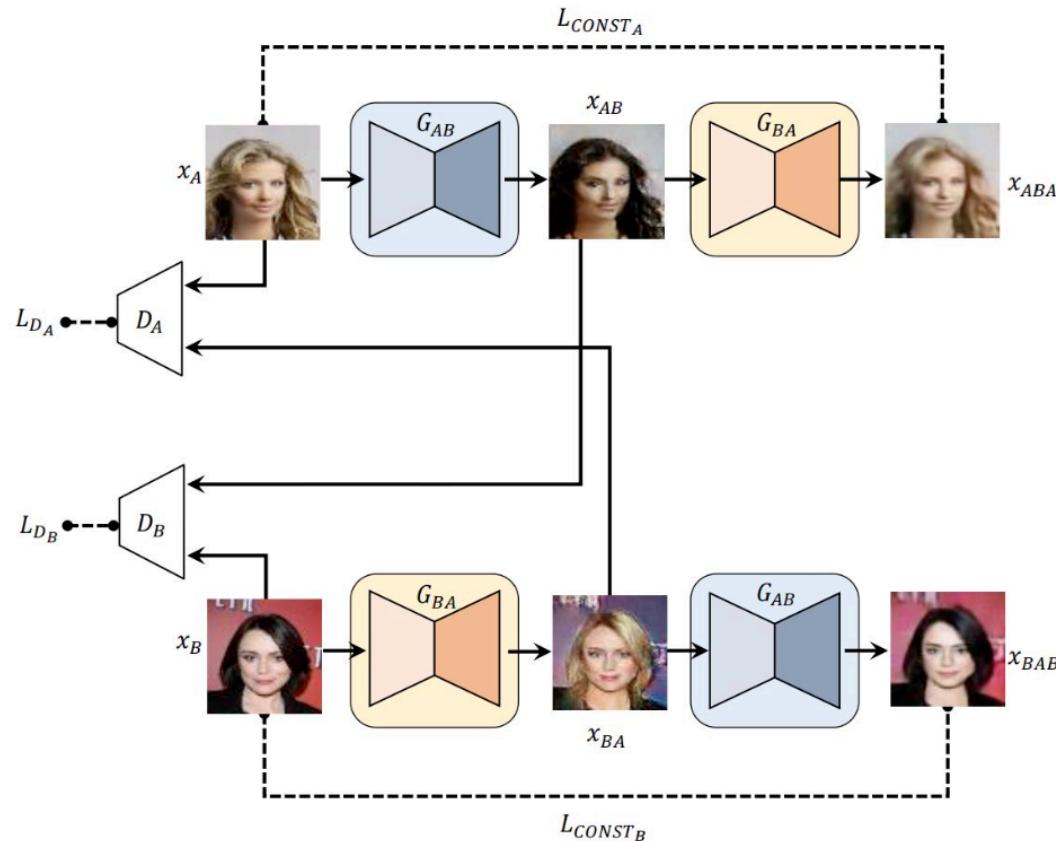
# CycleGAN



1. Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In ICCV, 2017.

# Cross-domain with DiscoGAN

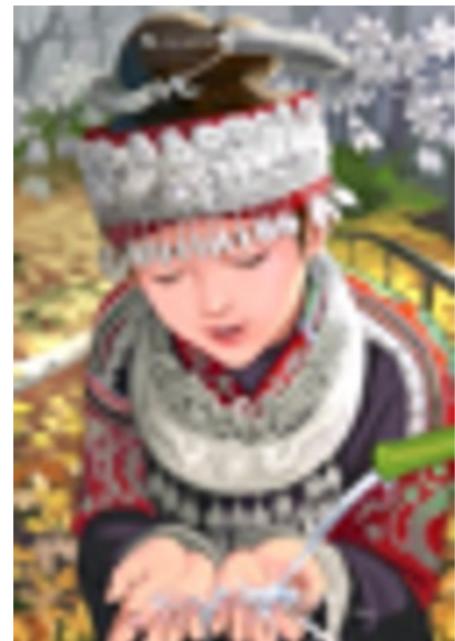
## DiscoGAN



1. Kim et al. "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks". In ICML, 2017.

# Image Super-Resolution

bicubic  
(21.59dB/0.6423)



SRResNet  
(23.53dB/0.7832)



SRGAN  
(21.15dB/0.6868)

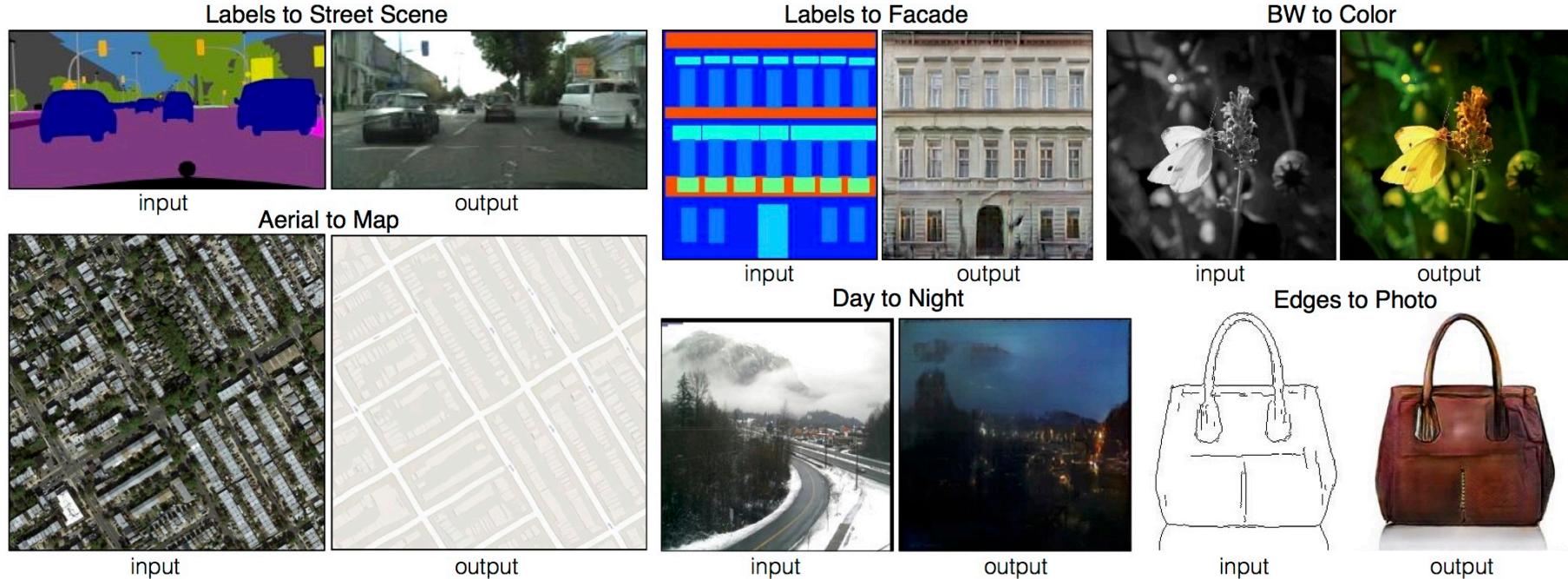


original



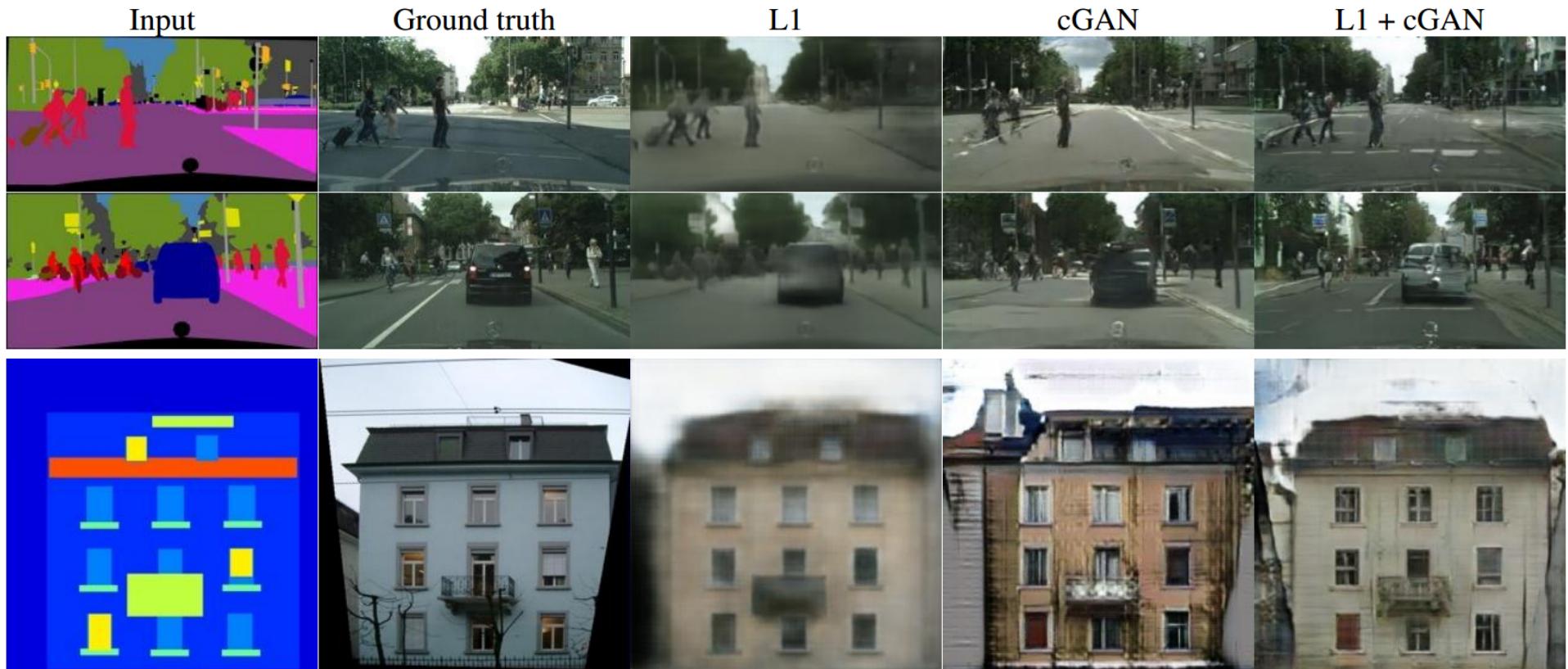
- Conditional on low-resolution input image

# Image-to-Image Translation



- Conditioned on an image of different modality
- No need to specify the loss function

# Label2Image



# Edges2Image



# Text2Image

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen



# Plug & Play Generative Networks



redshank

ant

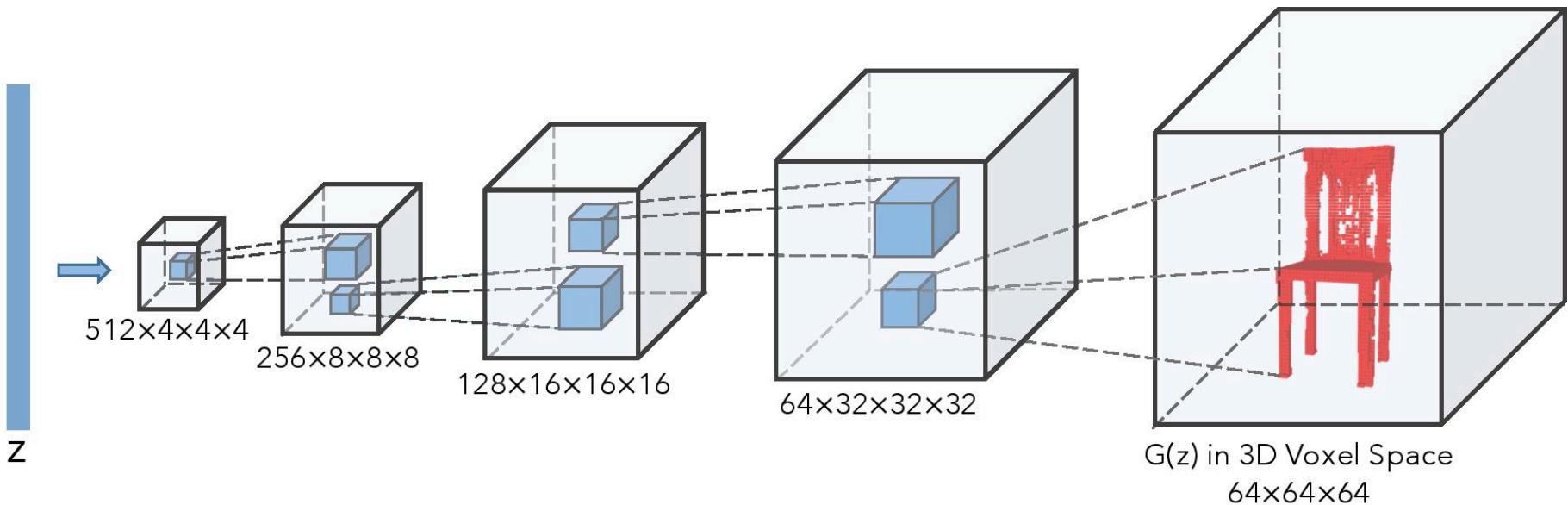
monastery



## volcano

[[Nguyen et al. 2016](#)]

# Shape modeling using 3D Generative Adversarial Network



# Recent Trend (11): Deep Reinforcement Learning

## 10 Breakthrough Technologies 2017

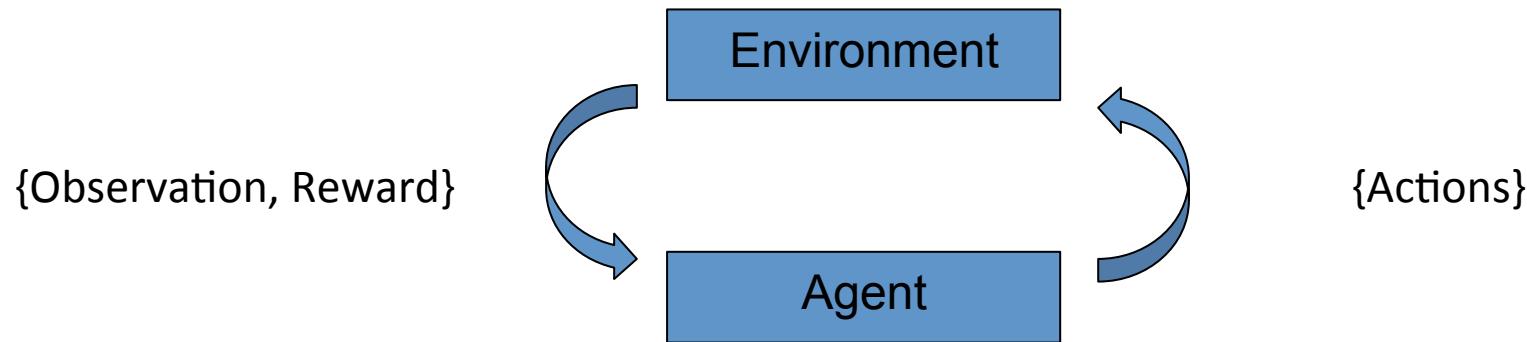
T

These technologies all have staying power. They will affect the economy and our politics, improve medicine, or influence our culture. Some are unfolding now; others will take a decade or more to develop. But you should know about all of them right now.

MIT  
Technology  
Review

# Reinforcement Learning (RL)

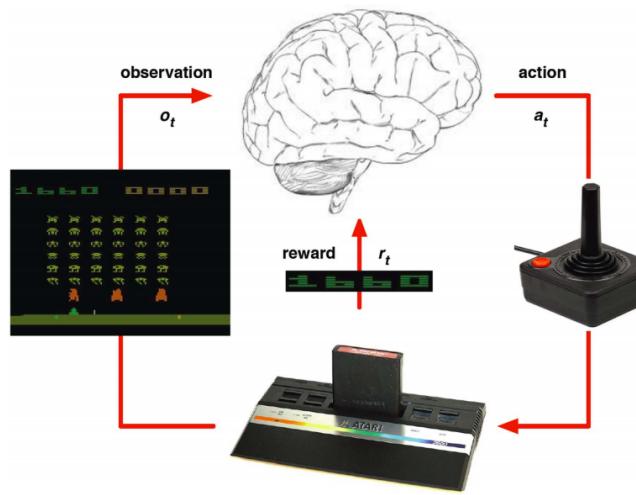
- What's Reinforcement Learning?



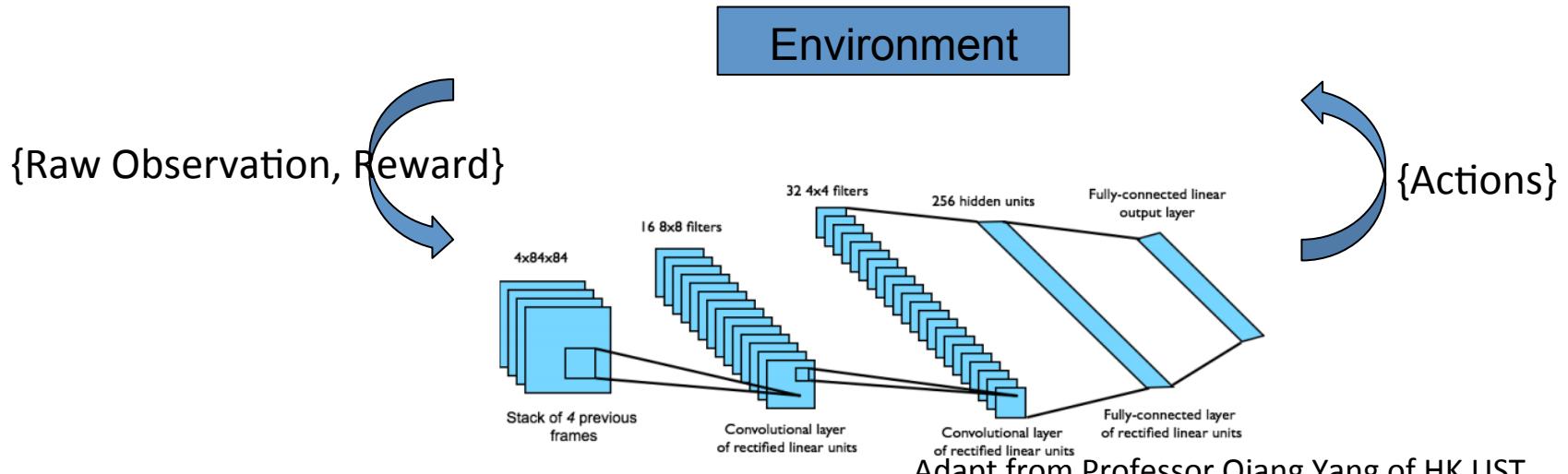
- Agent interacts with an environment and learns by maximizing a scalar reward signal
- No labels or any other supervision signal.
- Previously suffering from hand-craft states or representation.

# Deep Reinforcement Learning

- Human

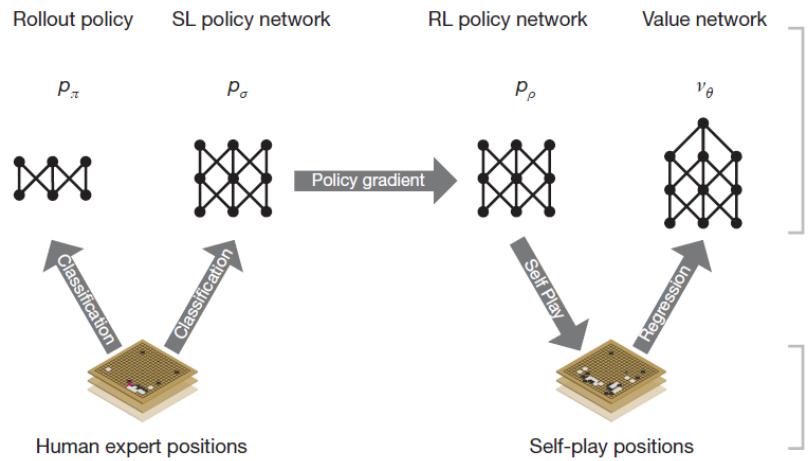


- So what's **DEEP RL**?



# AlphaGO: Learning Pipeline

- Combine Supervised Learning (SL) and RL to learn the search direction in Monte Carlo Tree Search



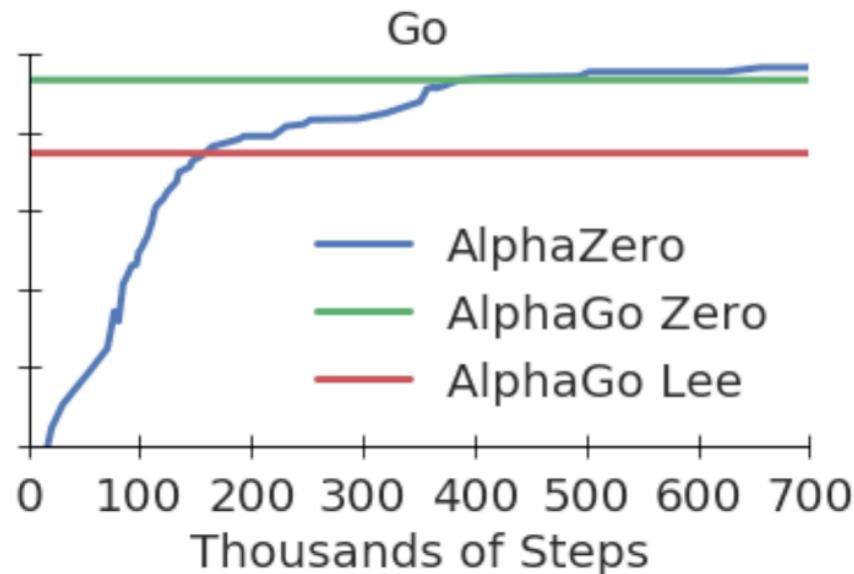
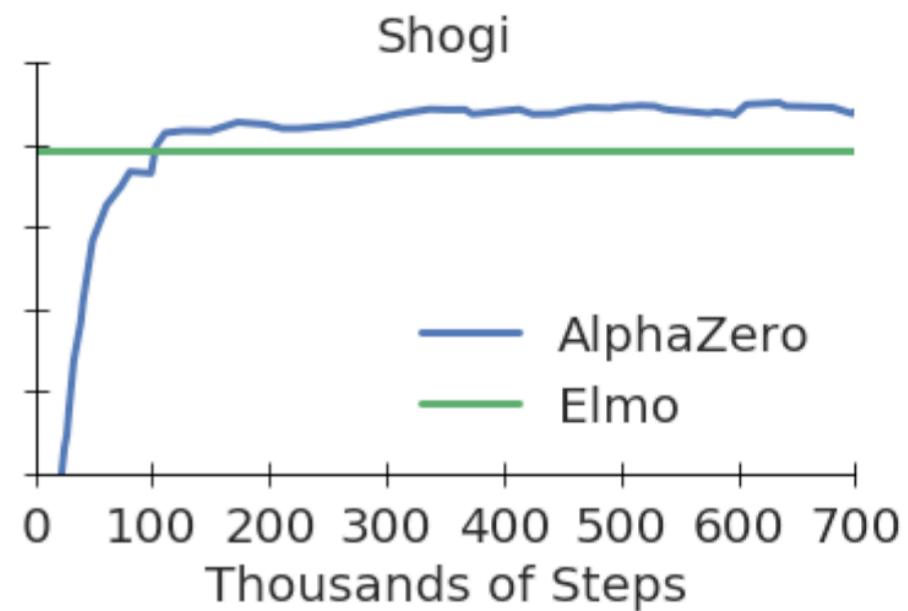
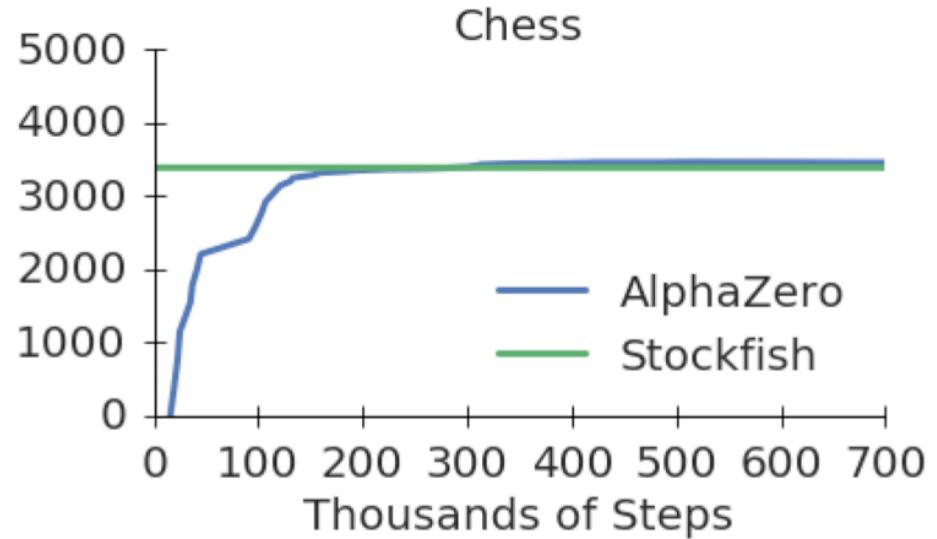
Silver, David, et al. 2016.

- **SL policy Network**
  - Prior search probability or potential
- **Rollout:**
  - combine with MCTS for quick simulation on leaf node
- **Value Network:**
  - Build the Global feeling on the leaf node situation

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.

## AlphaGo {Fan, Lee, Master} × AlphaGo Zero:

- supervised learning from human expert positions × from scratch by self-play reinforcement learning (“tabula rasa”)
- additional (auxiliary) input features × only the black and white stones from the board as input features
- separate policy and value networks × single neural network
- tree search using also Monte Carlo rollouts × simpler tree search using only the single neural network to both evaluate positions and sample moves
- (AlphaGo Lee) distributed machines + 48 tensor processing units (TPUs) × single machines + 4 TPUs
- (AlphaGo Lee) several months of training time × 72 h of training time (outperforming AlphaGo Lee after 36 h)



Recent Trend (12):  
Robustness / Trustworthiness /  
Understand / Verify / Test / Evade /  
Protect DNN



Validation

# Evade DNN, e.g. Adversarial Examples



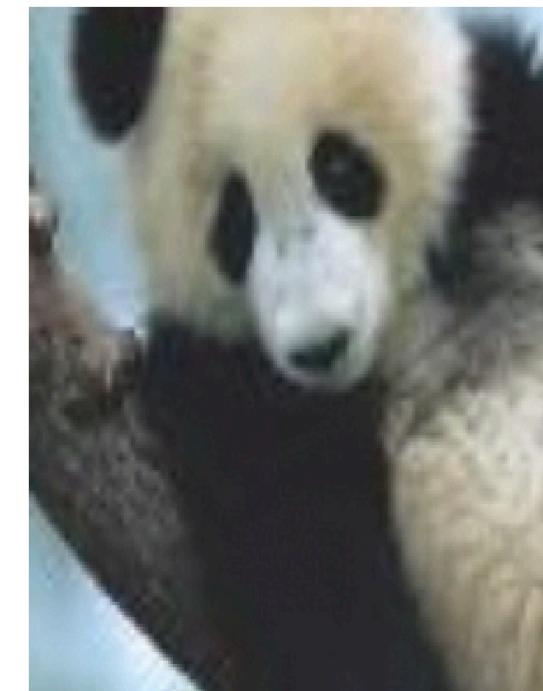
“panda”

+

$0.007 \times [\text{noise}]$

=

“gibbon”



Example from: Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. ICLR 2015.

# Today Recap

- Deep Learning
  - Basics
  - History
  - Why is this a breakthrough ?
  - Recent trends
  - (Many more exciting trends not covered here!)