Deep Learning for AI

Yoshua Bengio

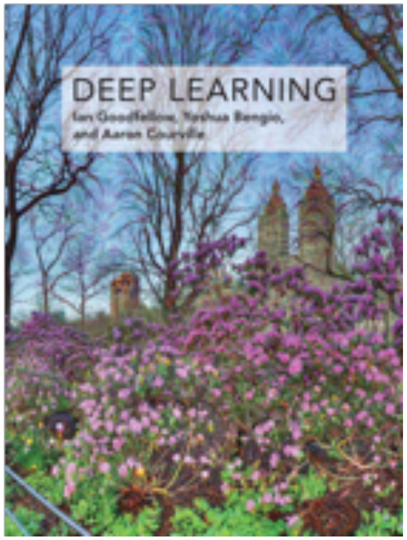
June 25-26th, 2019

MINOA Summer School, Ischia, Italy

MILA

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

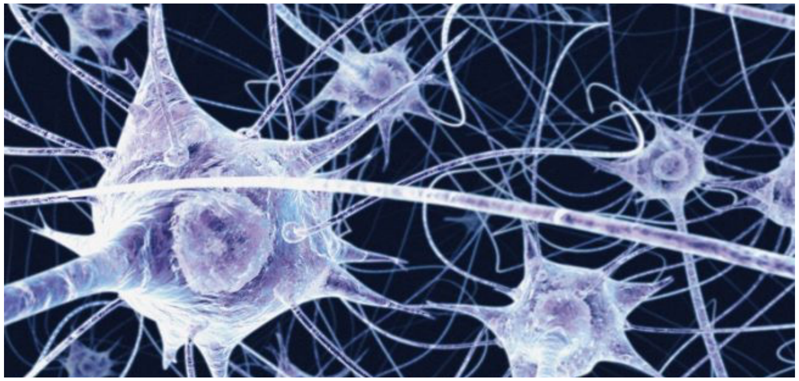
PLUG: **Deep Learning**, MIT Press book is out, chapters will remain online



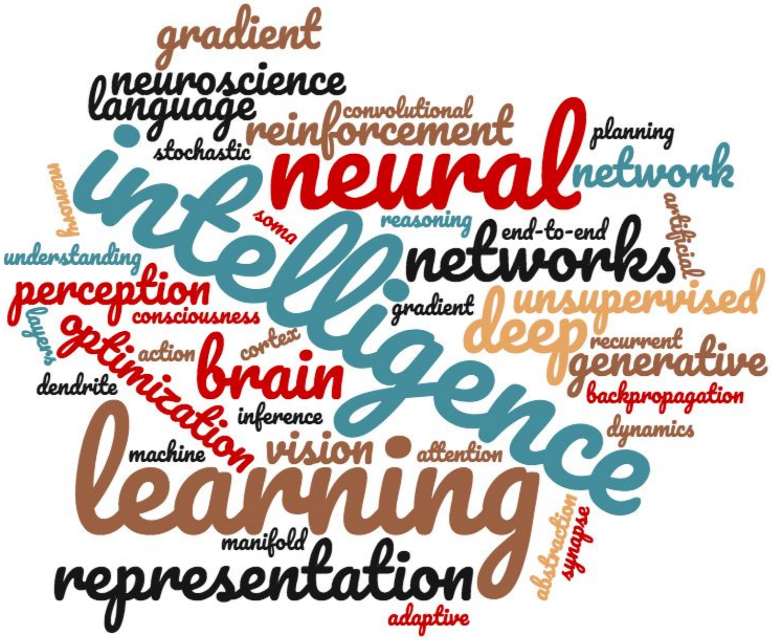
Université de Montréal

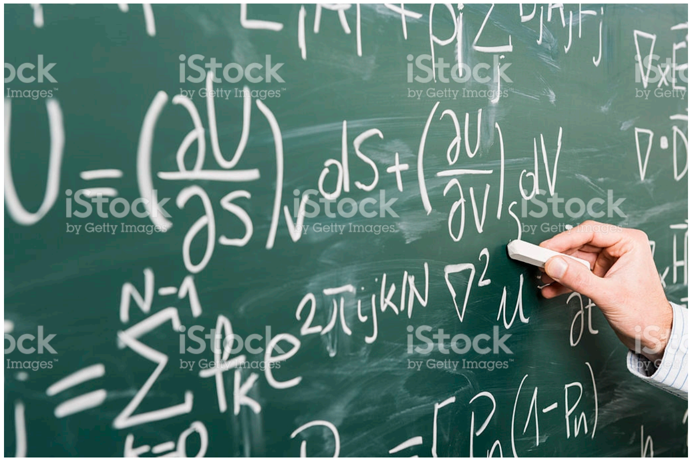
IVADO

Neural Networks & AI: Underlying Assumption



- There are principles giving rise to intelligence (machine, human or animal) via learning, simple enough that they can be described compactly, similarly to the laws of physics, i.e., our intelligence is not just the result of a huge bag of tricks and pieces of knowledge, but of general mechanisms to acquire knowledge.



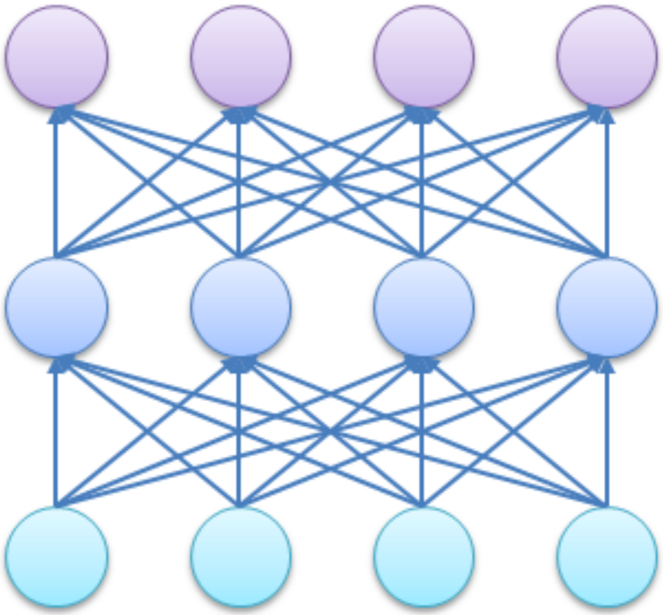


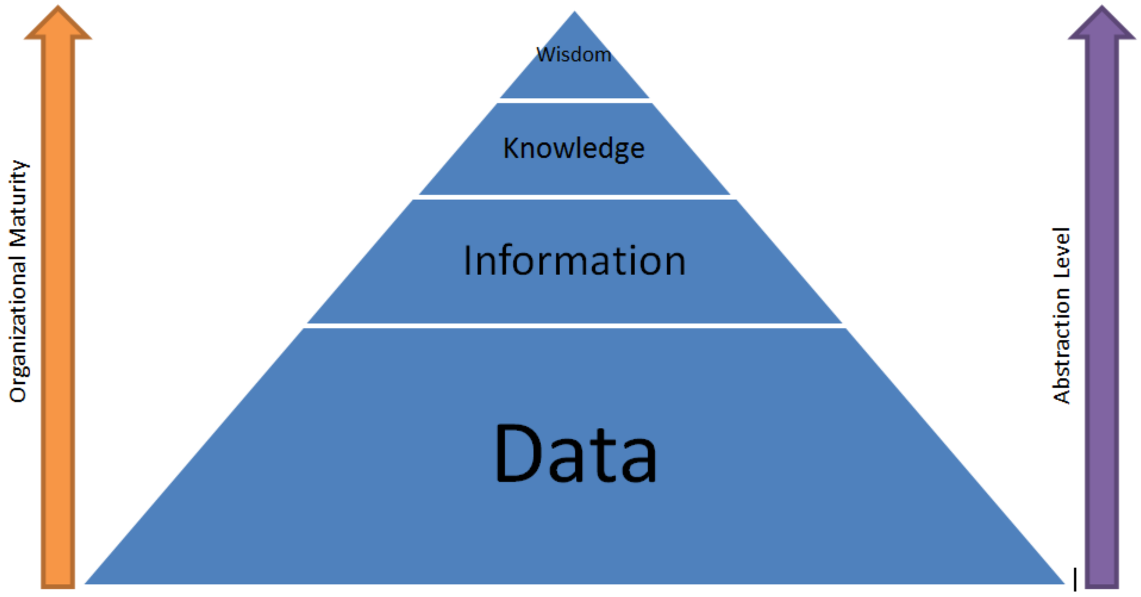
Learning Multiple Levels of Abstraction

(Bengio & LeCun 2007)

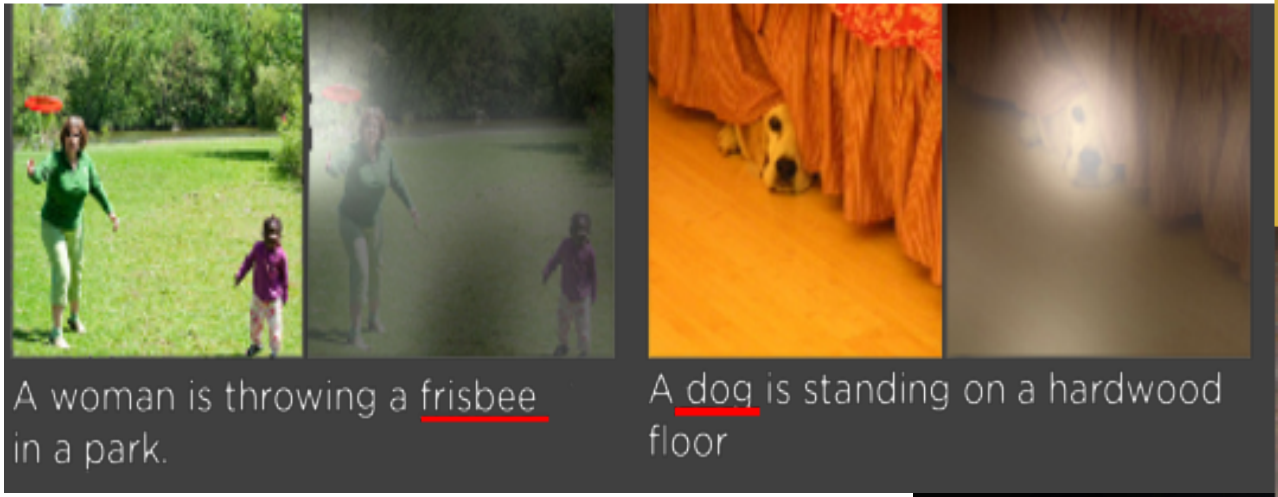
- The big payoff of deep learning is to facilitate learning higher levels of abstraction

- Higher-level abstractions can **disentangle the factors of variation**, which allows much easier generalization and transfer



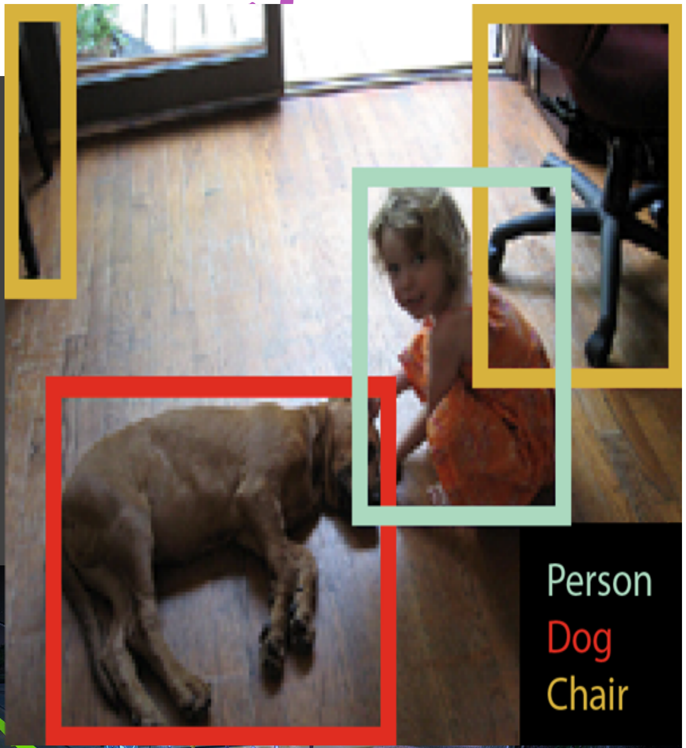


Deep Learning AI Breakthroughs



A woman is throwing a frisbee in a park.

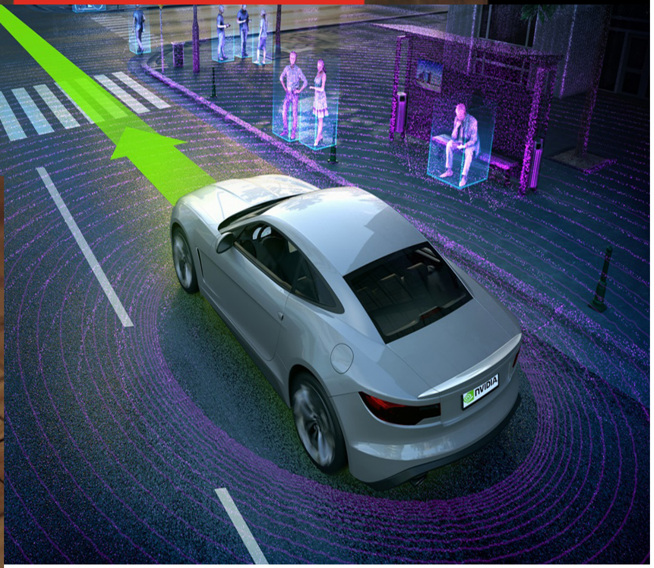
A dog is standing on a hardwood floor

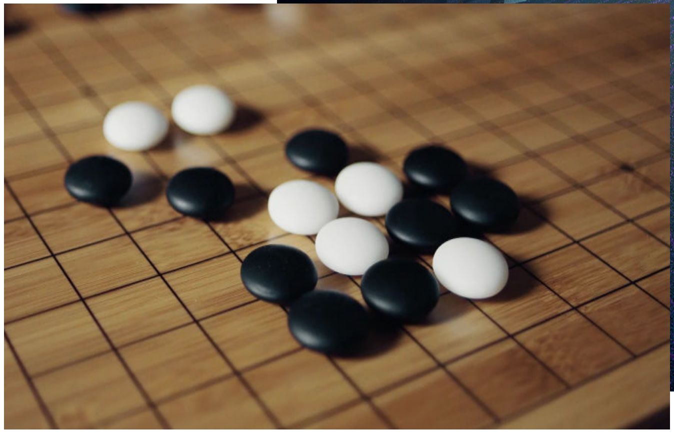


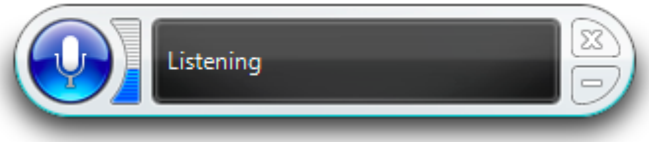
Computers have made huge strides in

perception,

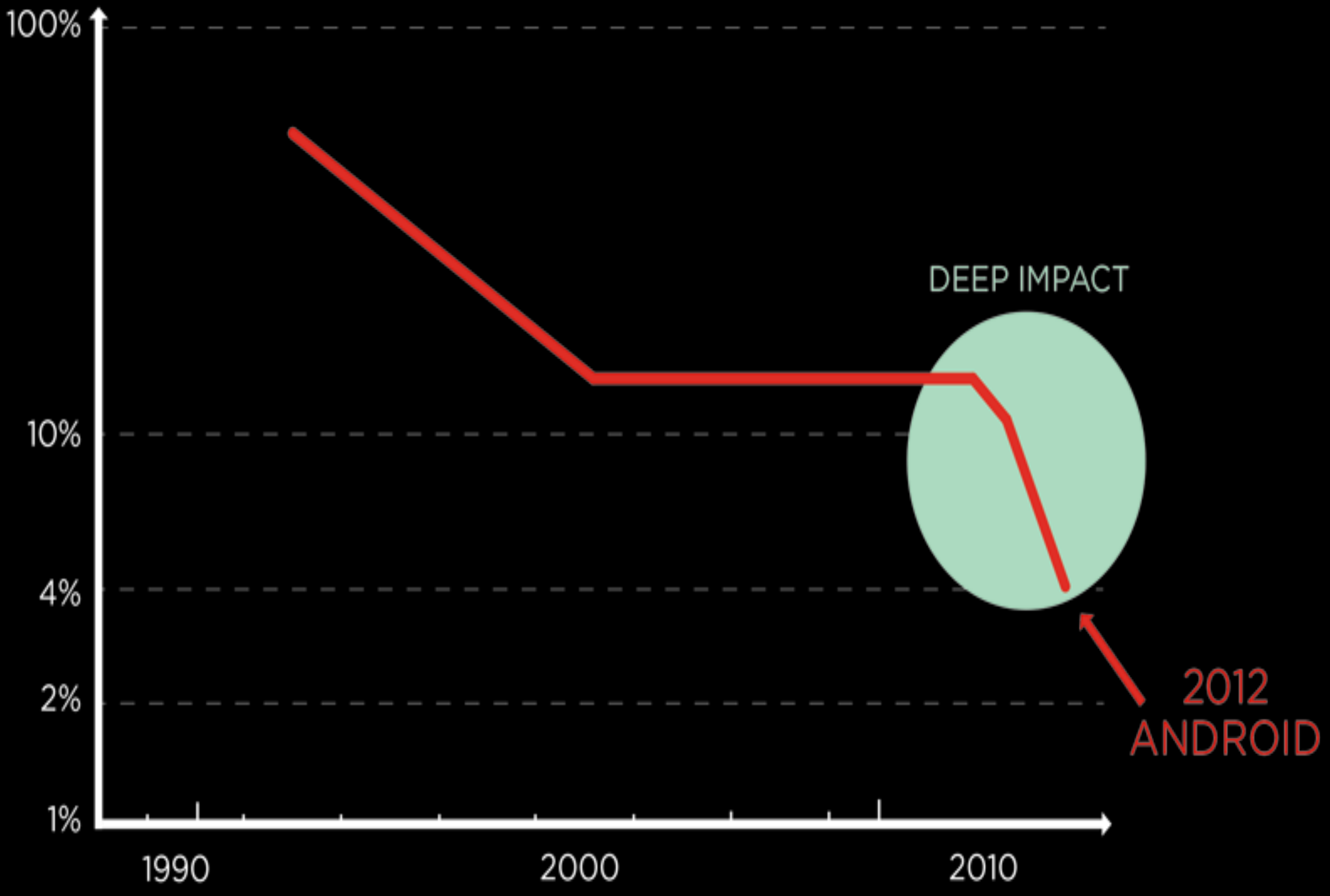
manipulating language, games, reasoning, ...







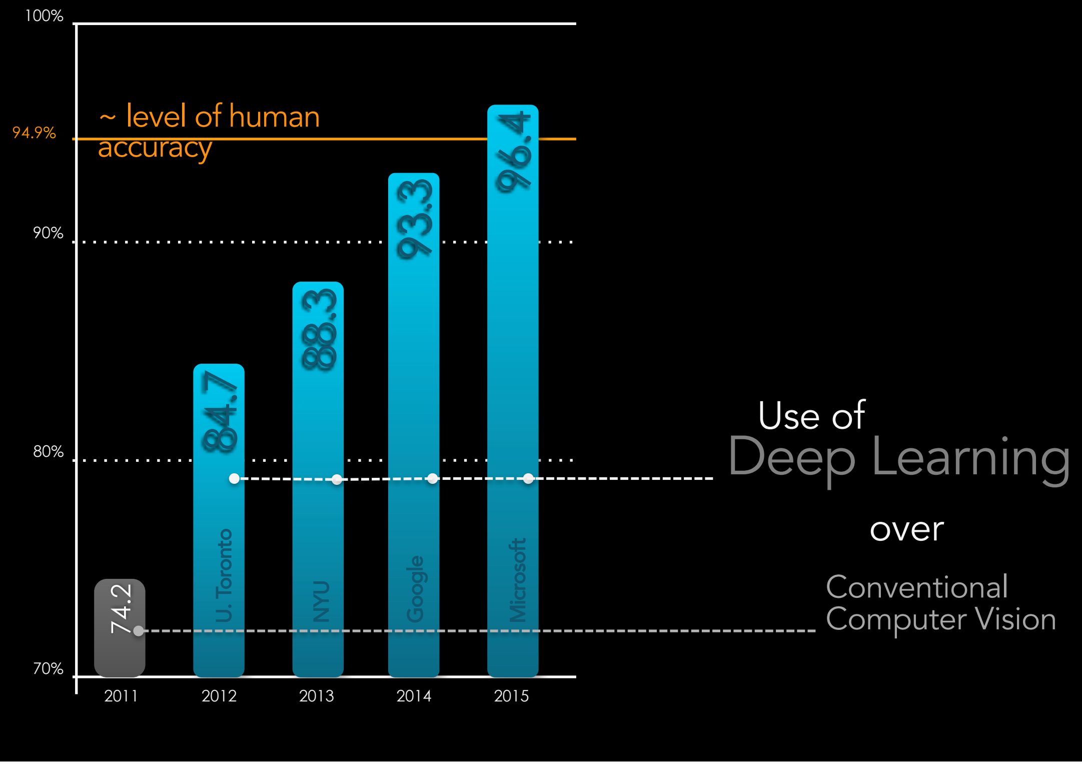
2010-2012: breakthrough in speech recognition



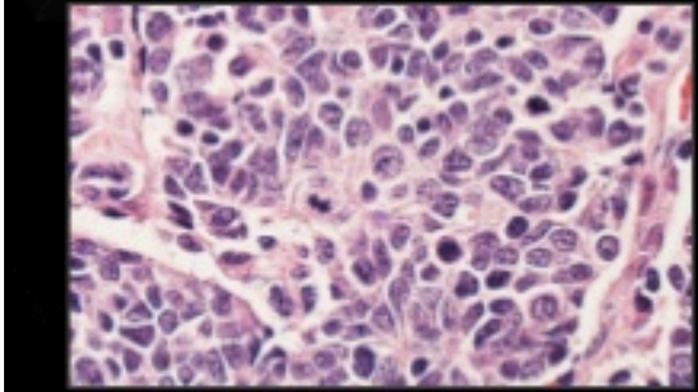
Source: Microsoft

2012-2015: breakthrough in computer vision

- Graphics Processing Units (GPUs) + 10x more data
- 1,000 object categories,
- Facebook: millions of faces
- ***2015: human-level performance***

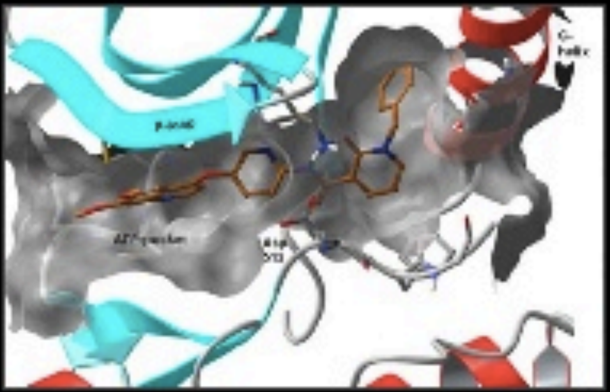


DEEP LEARNING REVOLUTIONIZING MEDICAL RESEARCH



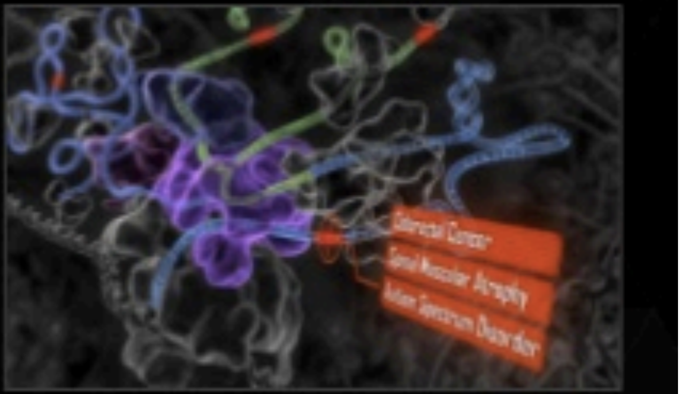
Detecting Mitosis in Breast Cancer Cells

— IDSIA



Predicting the Toxicity of New Drugs

— Johannes Kepler University



Understanding Gene Mutation to Prevent Disease

— University of Toronto

Medical Image Classification

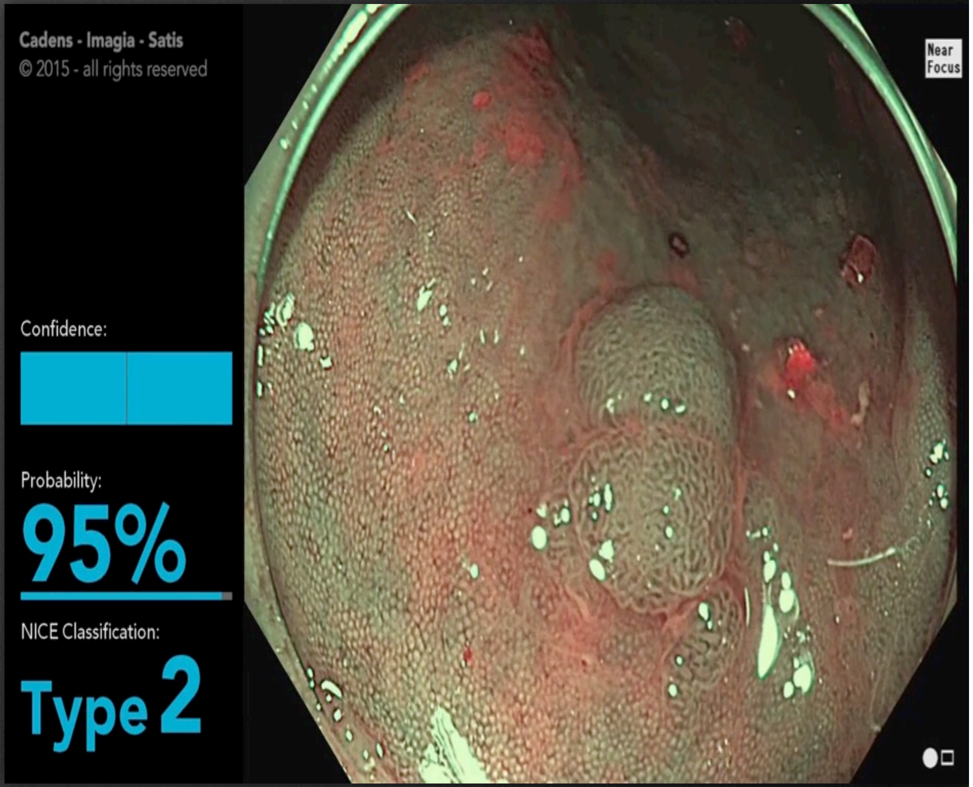
Clinical Validation: Optical Colonoscopy

imagia

World's first real-time colon polyp malignancy determination from unmodified endoscope raw video with deep learning

	Accuracy
Imagia	**> 90%, real time**
*GI Experts (Key Opinion Leaders)**	~ 90%
*GI Doctors Trained by KOLs**	~ 75%

*(D. Rex, 2015)



Cadens - Imagia - Satis
© 2015 - all rights reserved

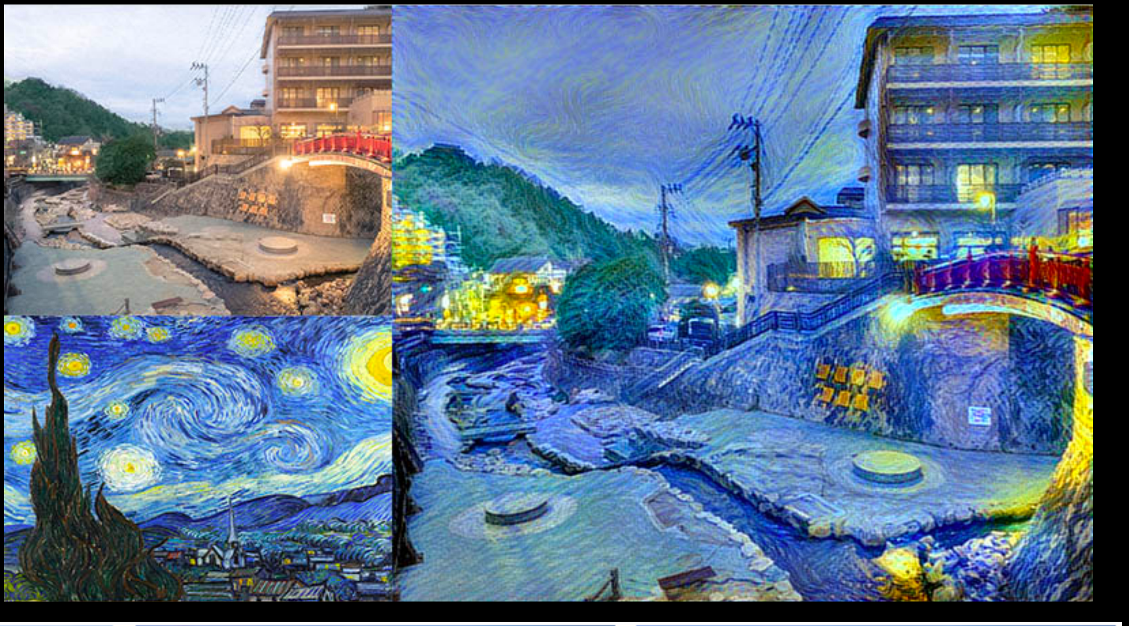
Confidence:

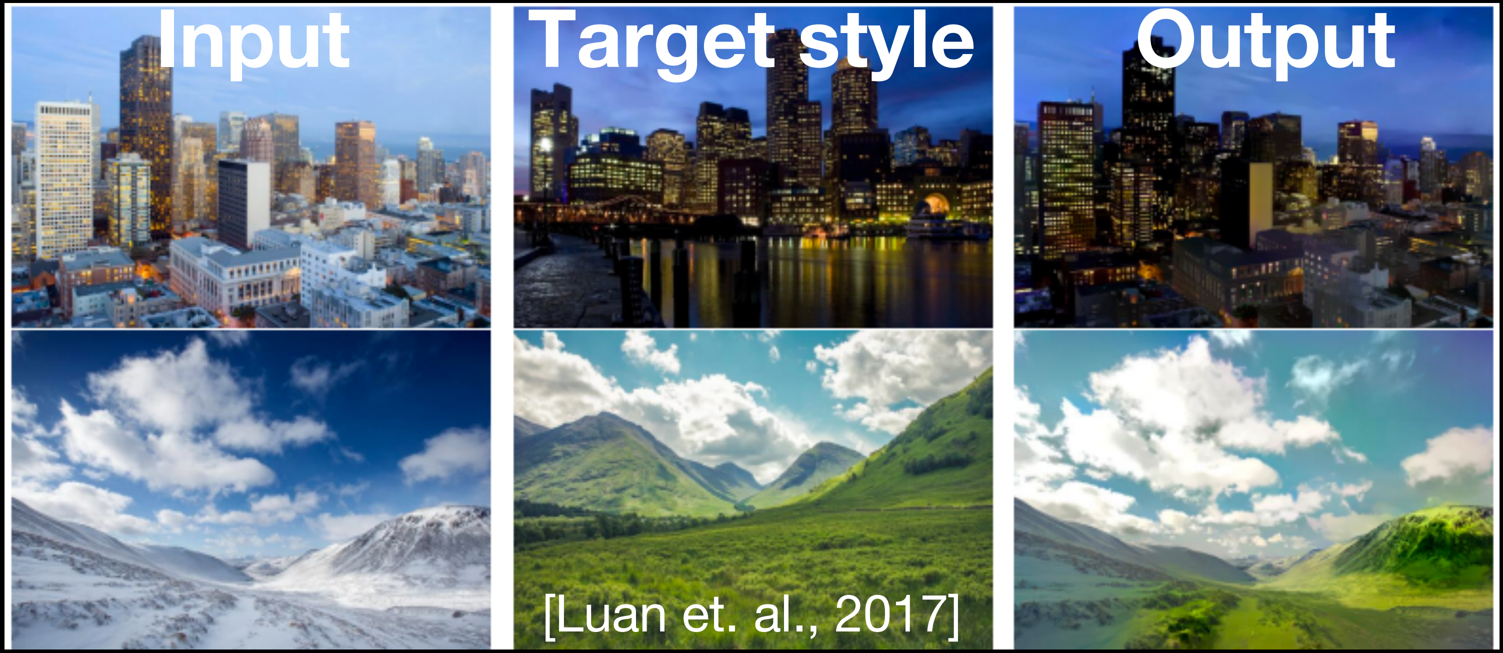
Probability:
95%

NICE Classification:
Type 2

Well received by expert clinicians and industry at many conferences, including Digestive Disease Week 2016

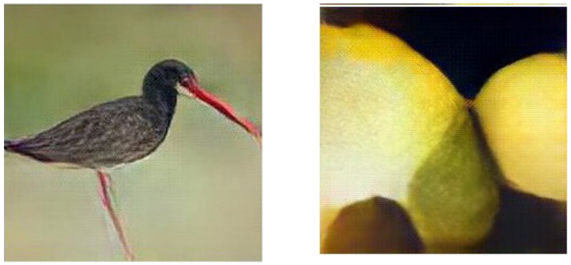
Separately Controlling Style & Content





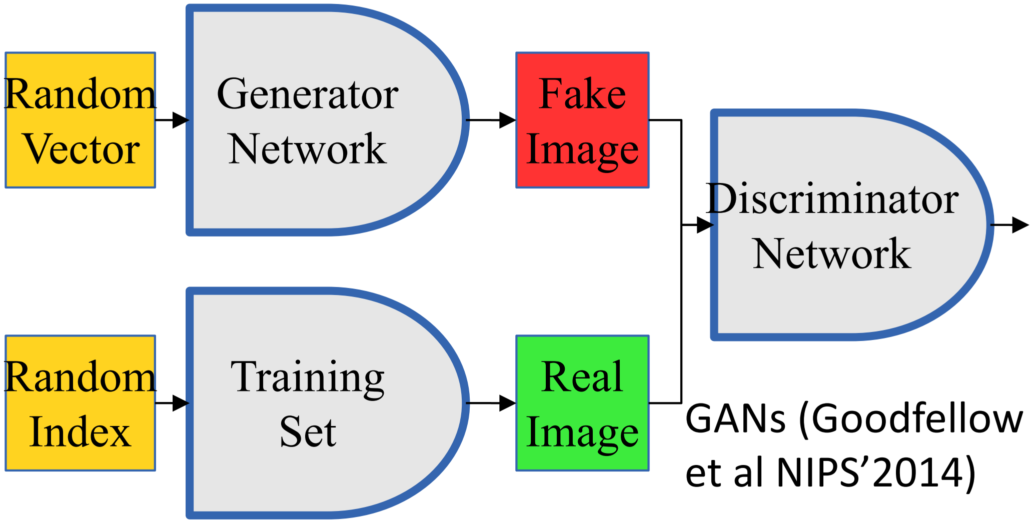
[Luan et. al., 2017]

Computers become Creative with Deep Generative Models

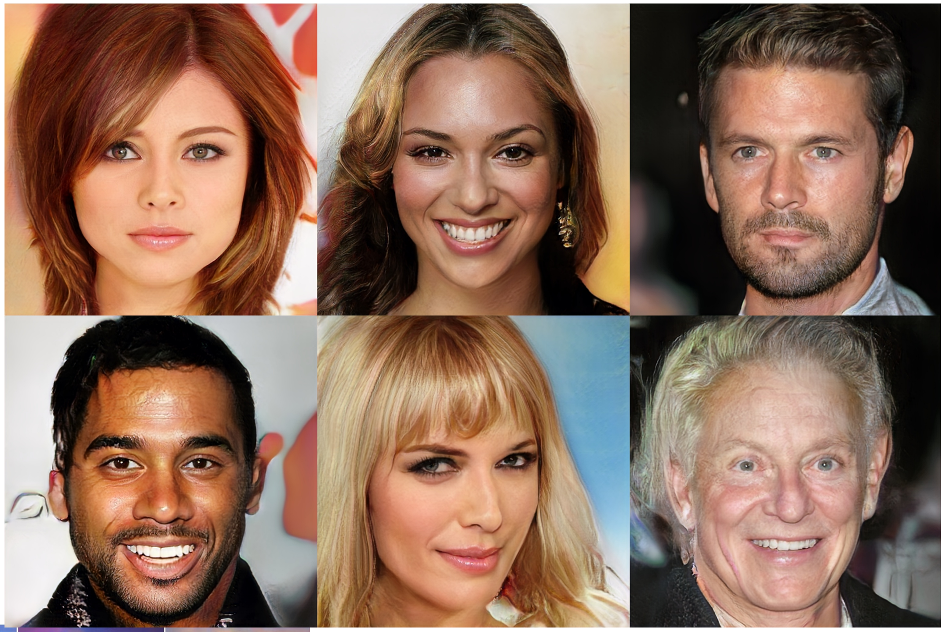


Predict a multi-modal future

- Progress in unsupervised generative neural nets allows them to synthesize a diversity images, sounds and text imitating unlabeled images, sounds or text



GANs (Goodfellow et al NIPS'2014)



(Karras et al 2017)

(Nguyen et al 2016)

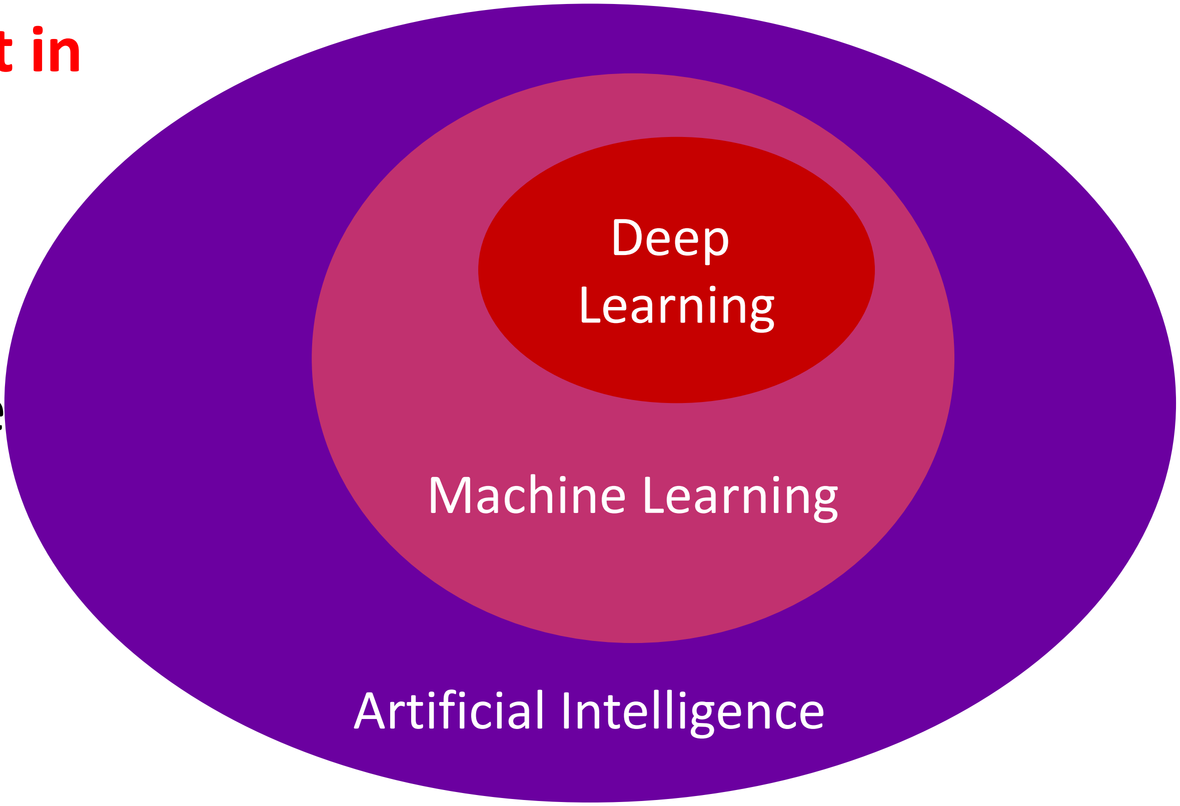
Intelligence Needs Knowledge

- Learning:

  powerful way to transfer knowledge to intelligent agents

- Failure of classical symbolic AI: a lot of knowledge is **intuitive, difficult to put in rules & facts, not consciously accessible**

- Solution: get knowledge from data & experience



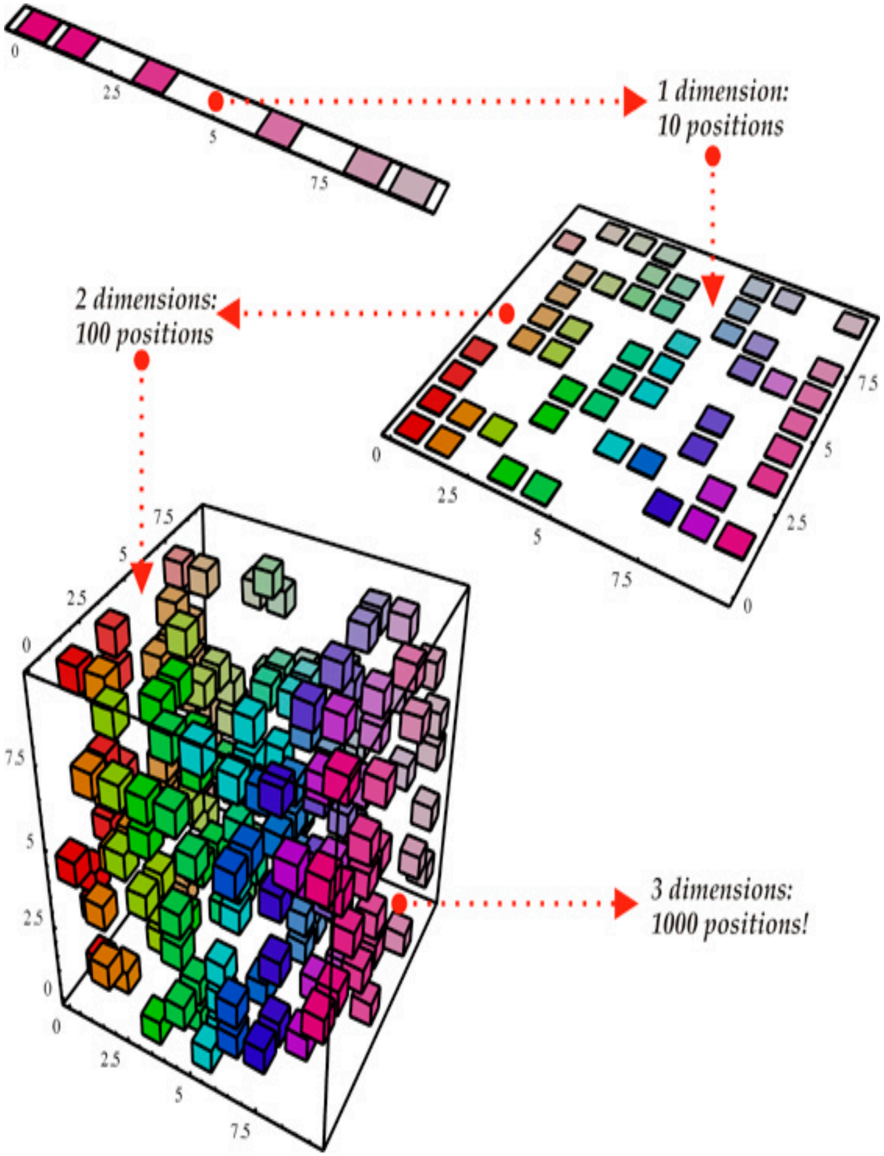
Machine Learning, AI & No Free Lunch

- Five key ingredients for ML towards AI
  1. Lots & lots of data
  2. Very flexible models
  3. Enough computing power
  4. Computationally efficient inference
  5. **Powerful priors that can defeat the curse of dimensionality**

ML 101. What We Are Fighting Against: The Curse of Dimensionality

To generalize
locally, need
representative
examples for all
relevant
variations!

Classical solution:
hope for a smooth
enough target
function, or make
it smooth by
handcrafting good
features / kernel



Bypassing the curse of dimensionality

We need to build compositionality into our ML models

Just as human languages exploit compositionality to give representations and meanings to complex ideas

Exploiting compositionality can give an **exponential** gain in representational power

Distributed representations / embeddings: feature learning

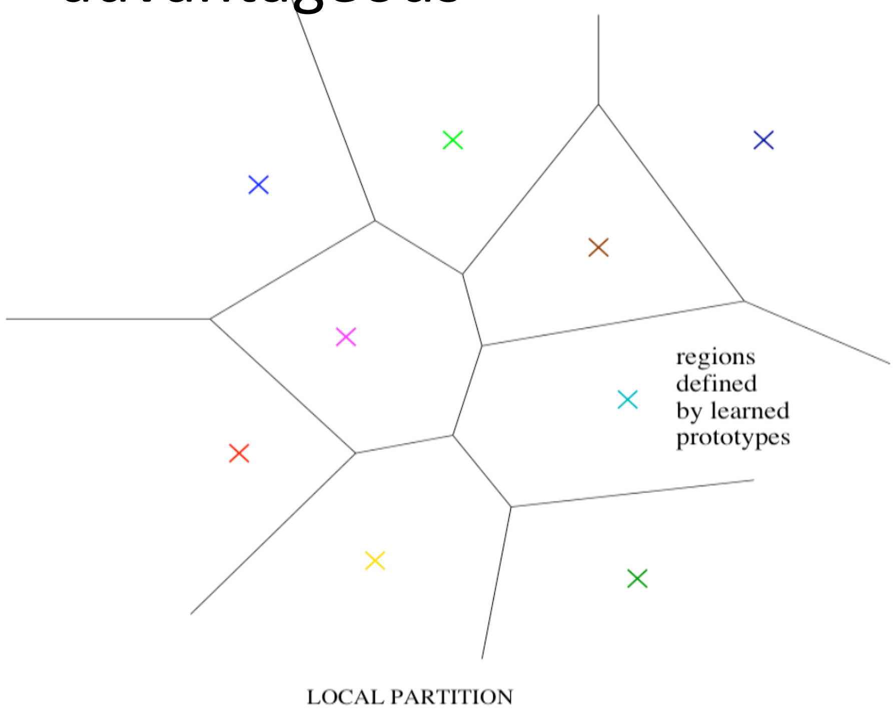
Deep architecture: multiple levels of feature learning

Prior assumption: compositionality is useful to describe the world around us efficiently

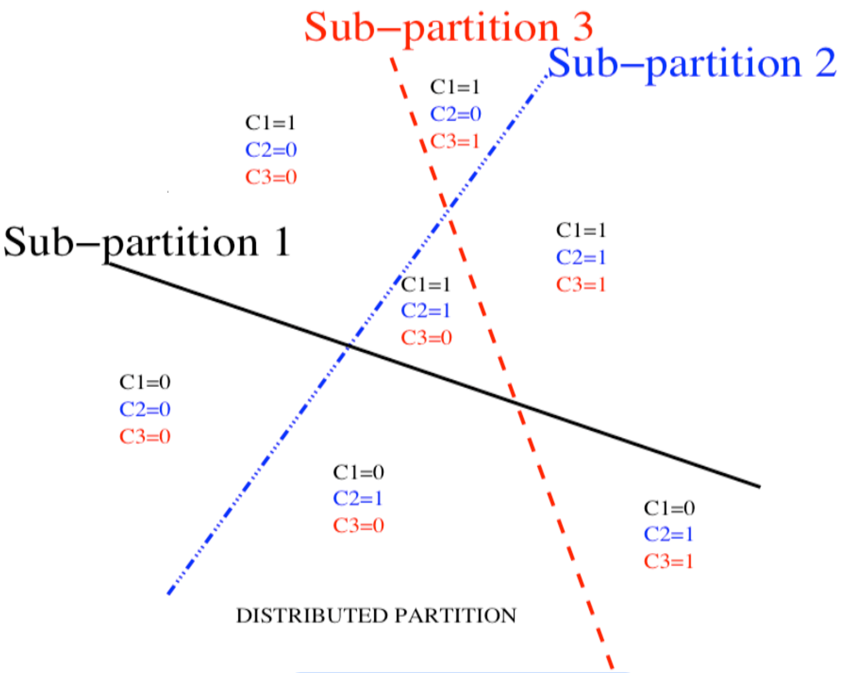
Learning Representations

Distributed Representations: The Power of Compositionality – Part 1

- Distributed (possibly sparse) representations, learned from data, can capture the **meaning** of the data and state
- Parallel composition of features: can be exponentially advantageous



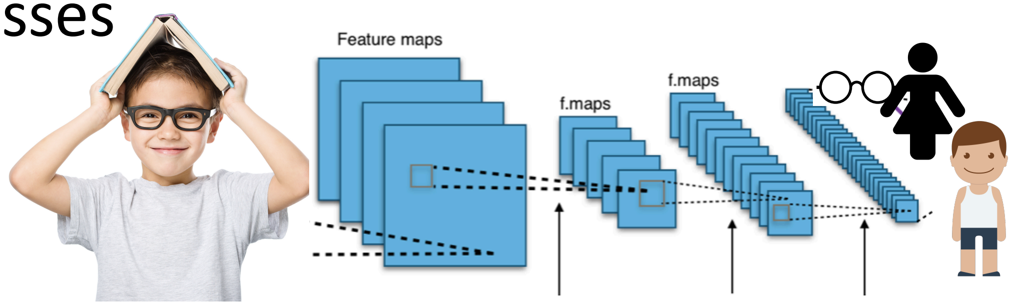
Not Distributed



Distributed

Each feature can be discovered without the need for seeing the exponentially large number of configurations of the other features

- Consider a network whose hidden units discover the following features:
  - Person wears glasses
  - Person is female
  - Person is a child
  - Etc.

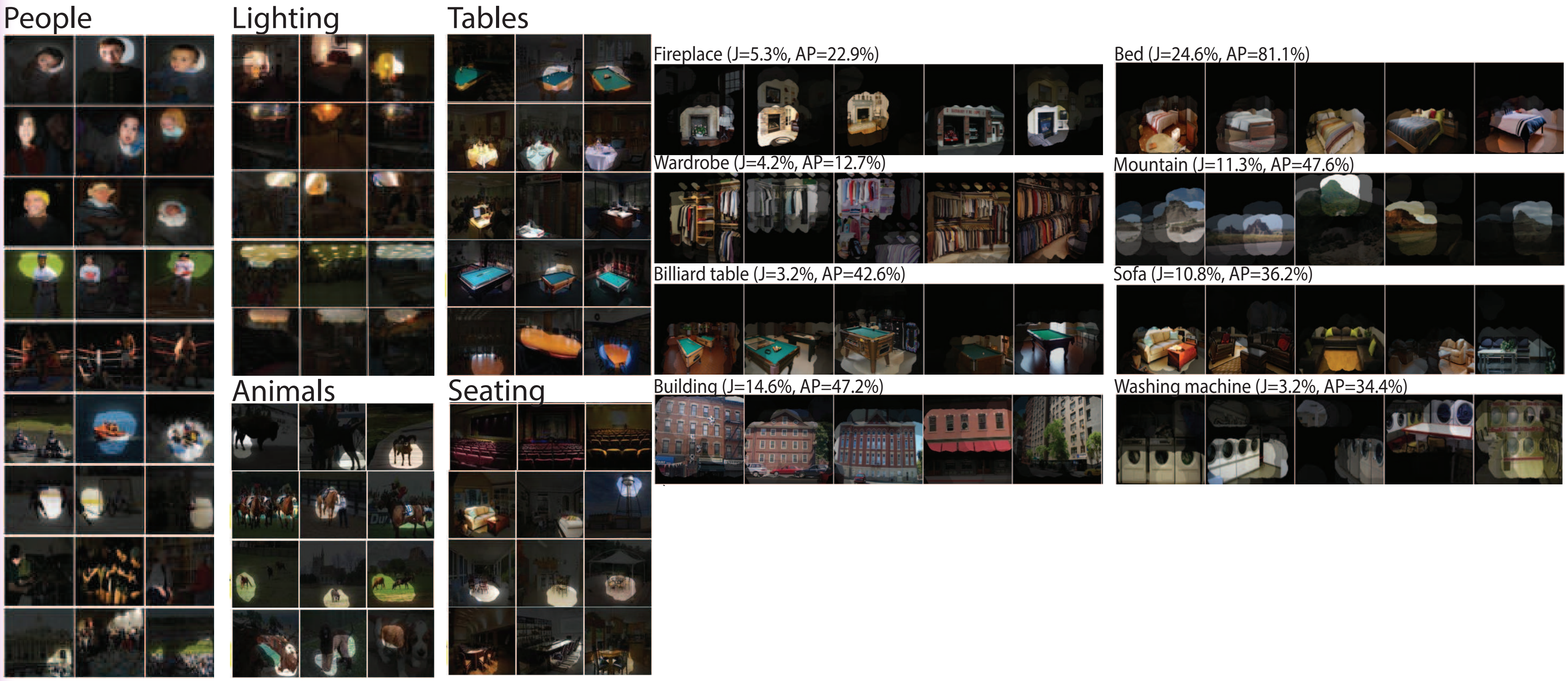


If each of *n* feature requires *O(k)* parameters, need *O(nk)* examples

Non-parametric methods would require $O(n^d)$ examples

Hidden Units Can Discover Semantically Meaningful Concepts

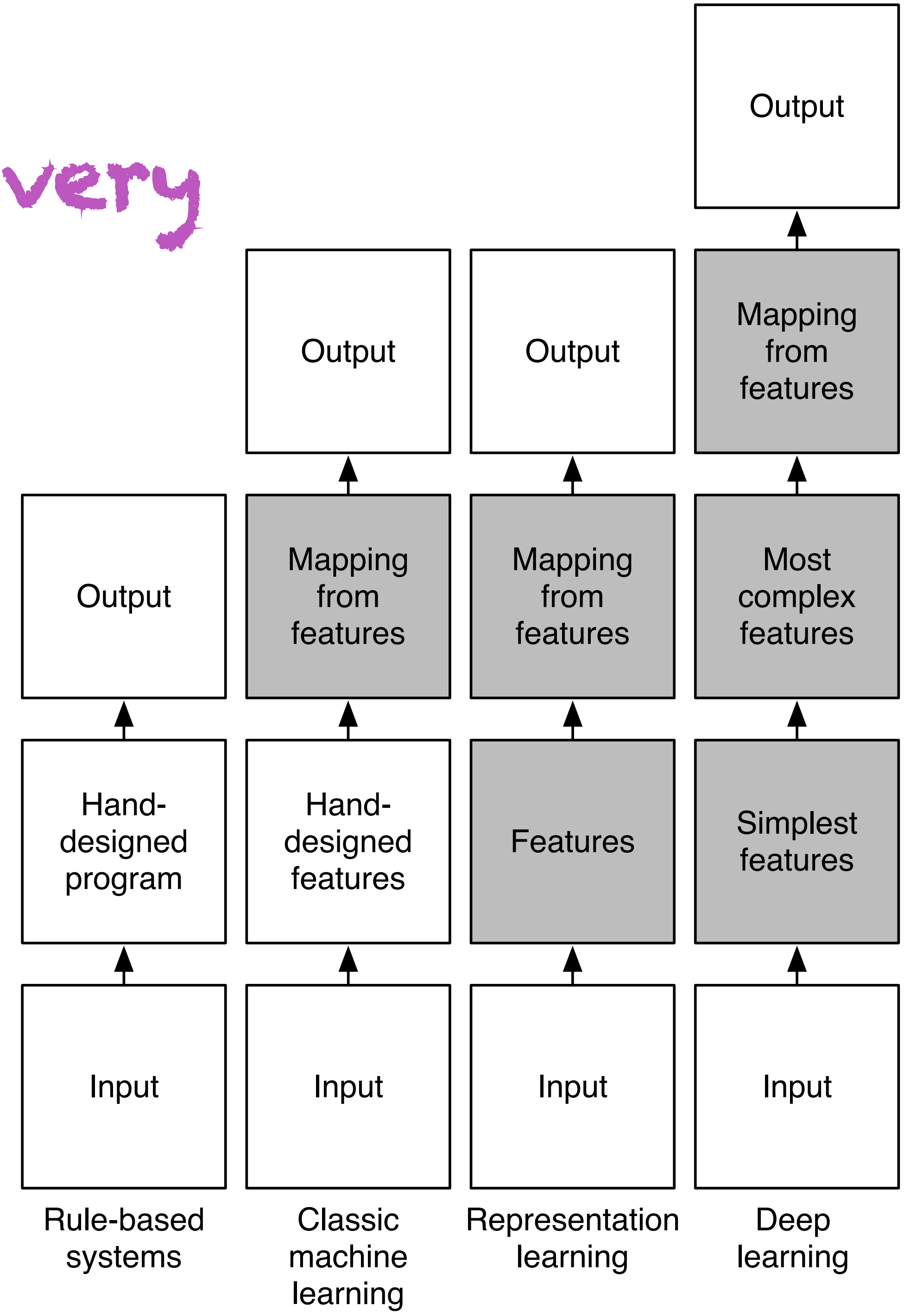
- *Zhou et al & Torralba, arXiv1412.6856 ,* ICLR 2015
- *Network trained to recognize places, not objects*


People
Lighting
Tables
Animals
Seating
Fireplace (J=5.3%, AP=22.9%)
Wardrobe (J=4.2%, AP=12.7%)
Billiard table (J=3.2%, AP=42.6%)
Building (J=14.6%, AP=47.2%)
Bed (J=24.6%, AP=81.1%)
Mountain (J=11.3%, AP=47.6%)
Sofa (J=10.8%, AP=36.2%)
Washing machine (J=3.2%, AP=34.4%)

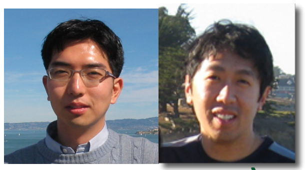
Deep Learning: Learning an Internal Representation

- Unlike other ML methods with either
  - no intermediate representation (linear)
  - or fixed (generally very high-dimensional) intermediate representations (SVMs, kernel machines)
- What is a good representation? Makes other tasks easier.

Automating Feature Discovery



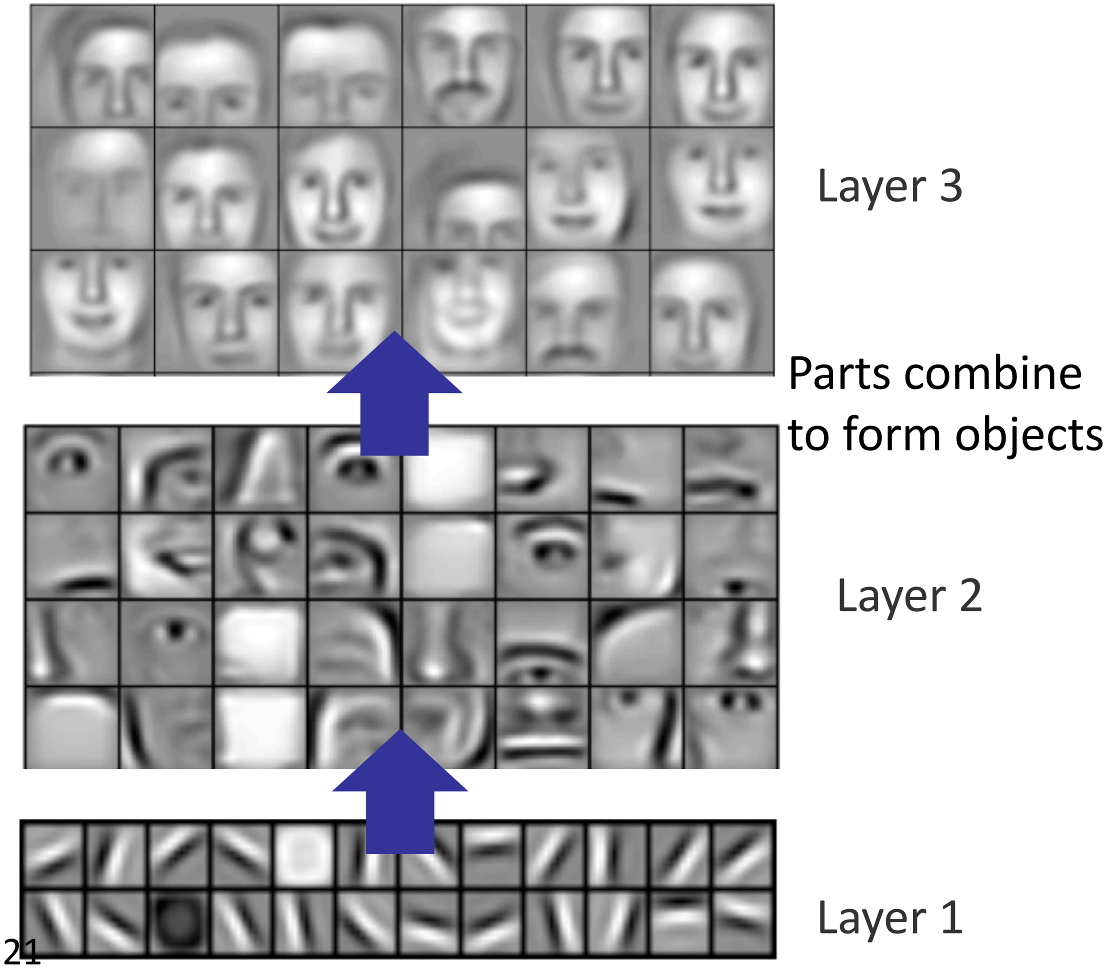
Learning multiple levels of representation

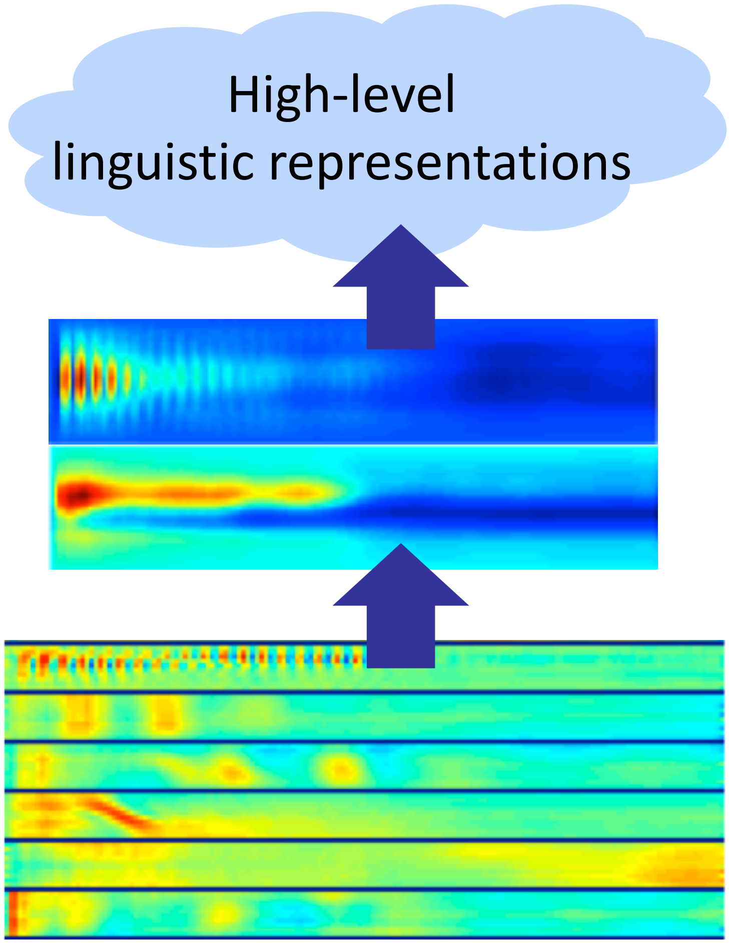


(Lee, Largman, Pham & Ng, NIPS 2009)

(Lee, Grosse, Ranganath & Ng, ICML 2009)

Successive model layers learn deeper intermediate representations

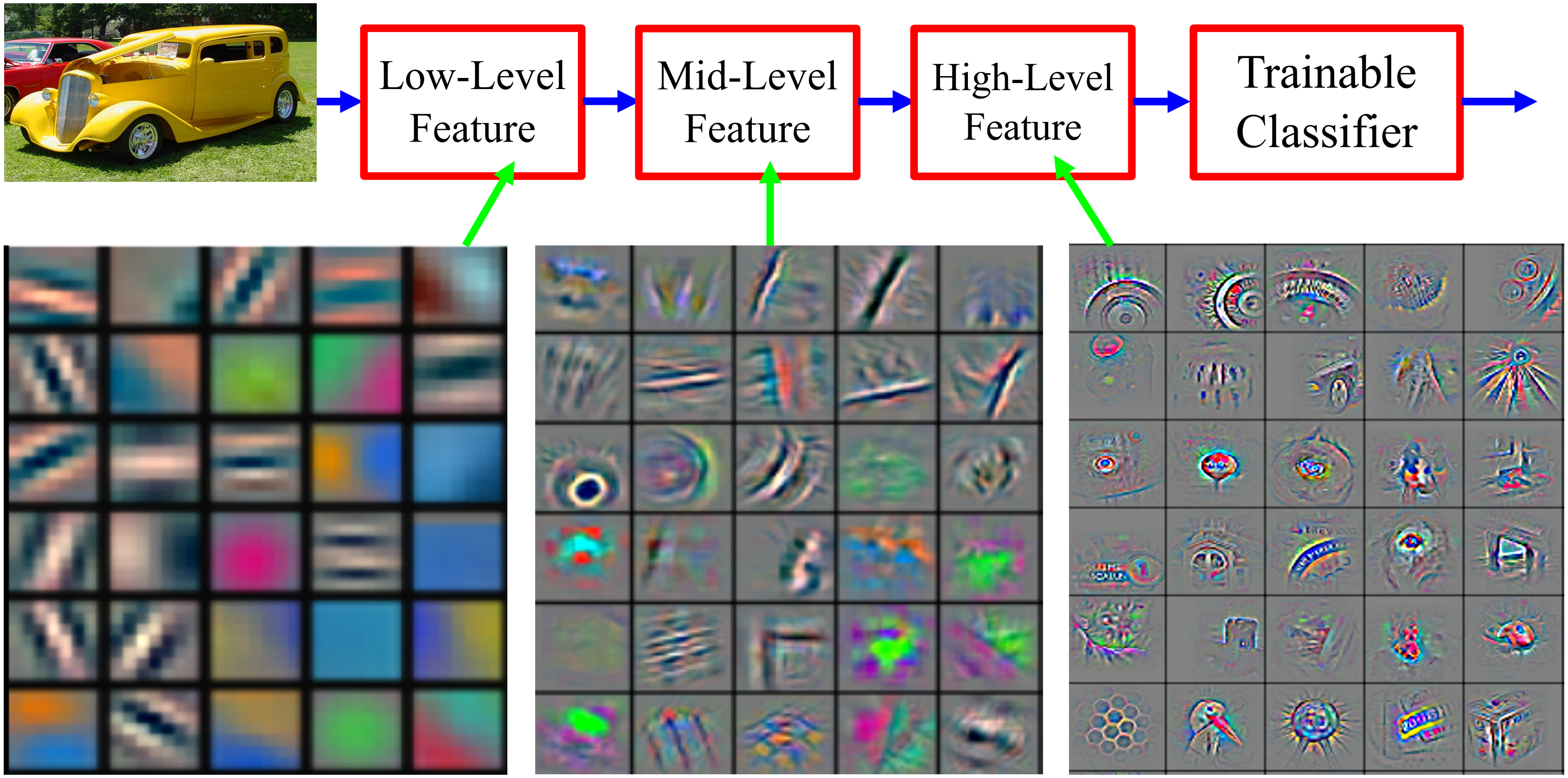




Prior: underlying factors & concepts compactly expressed w/ multiple levels of abstraction

Why Multiple Layers? The World is Compositional

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition: Pixel → edge → texton → motif → part → object
- Text: Character → word → word group → clause → sentence → story
- Speech: Sample → spectral band → sound → … → phone → phoneme → word



Low-Level Feature

Mid-Level Feature

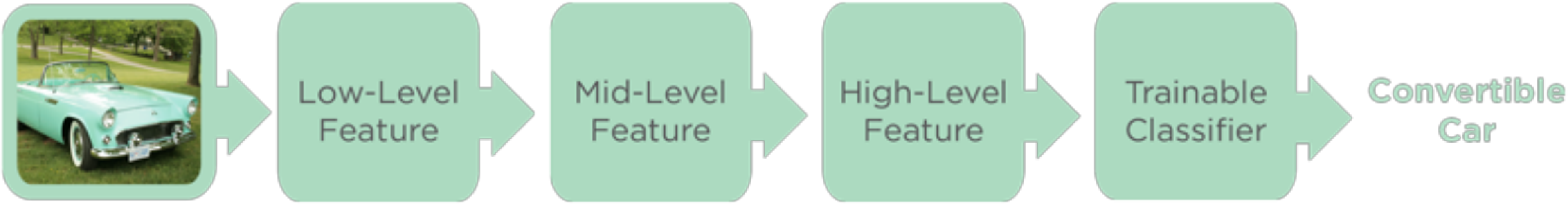
High-Level Feature

Trainable Classifier

Deep Representations: The Power of Compositionality – Part 2

- Learned function seen as a composition of simpler operations, e.g. inspired by neural computation
- Hierarchy of features, concepts, leading to more abstract factors enabling better generalization
- Again, theory shows this can be exponentially advantageous

Why multiple layers? The world is compositional



Exponential advantage of depth

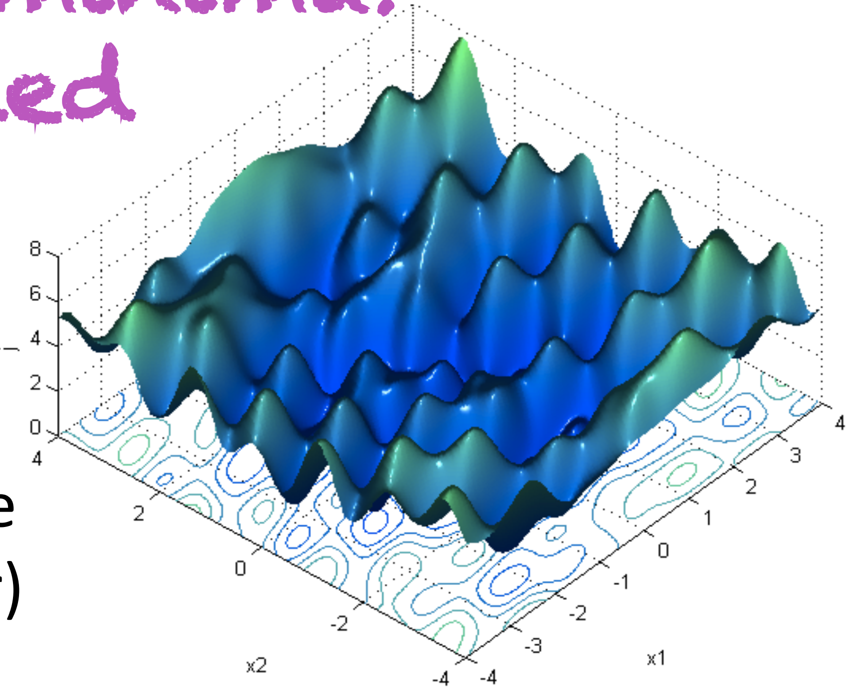
- Expressiveness of deep networks with piecewise linear activation functions: exponential advantage for depth
- *(Montufar et al & Bengio, NIPS 2014)*
- Number of pieces distinguished for a network with depth *L* and n_i units per layer is at least

$$\left(\prod_{i=1}^{L-1} \left\lfloor \frac{n_i}{n_0} \right\rfloor^{n_0}\right) \sum_{j=0}^{n_0} \binom{n_L}{j}$$

or, if hidden layers have width *n* and input has size n_0

$$\Omega\left(\left({}^{n}/_{n_0}\right)^{(L-1)n_0} n^{n_0} \right)$$

Not so terrible local minima: convexity is not needed

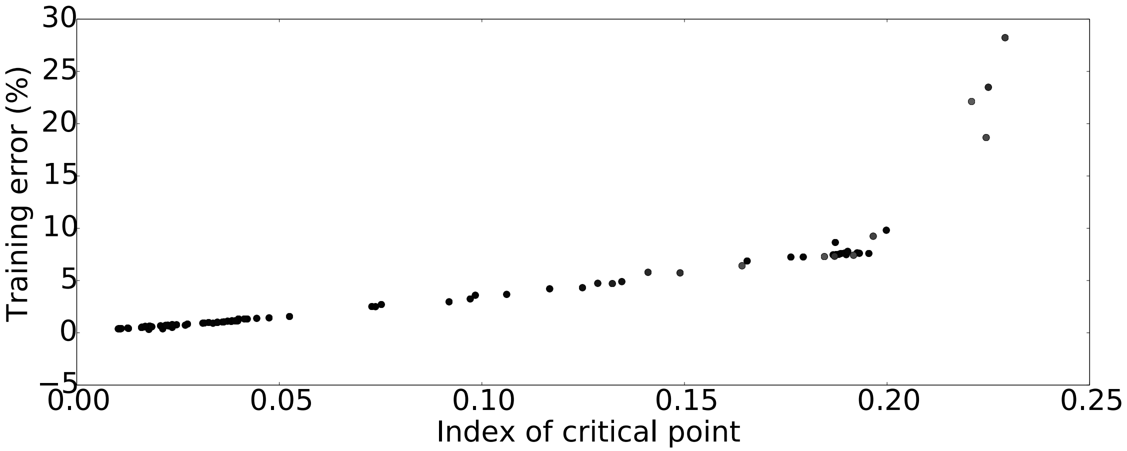


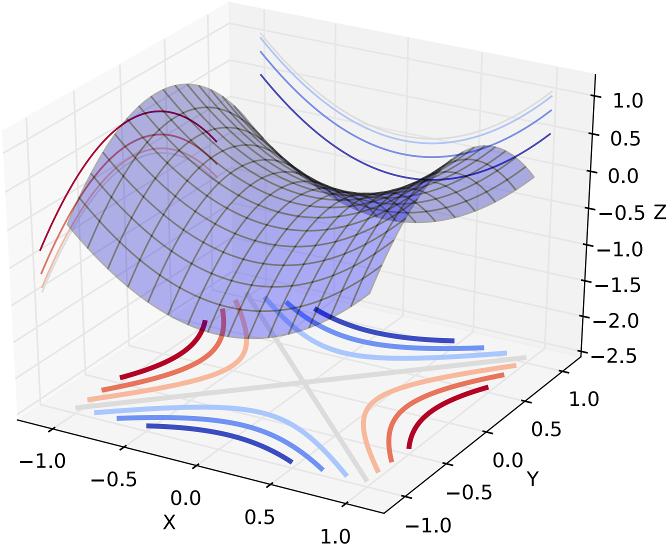
Myth busted:

- Local minima dominate in low-D, but saddle points dominate in high-D
- Most local minima are relatively close to the bottom (global minimum error)

(Dauphin et al NIPS'2014,

Choromanska et al AISTATS'2015)





Deep Nets and Backprop

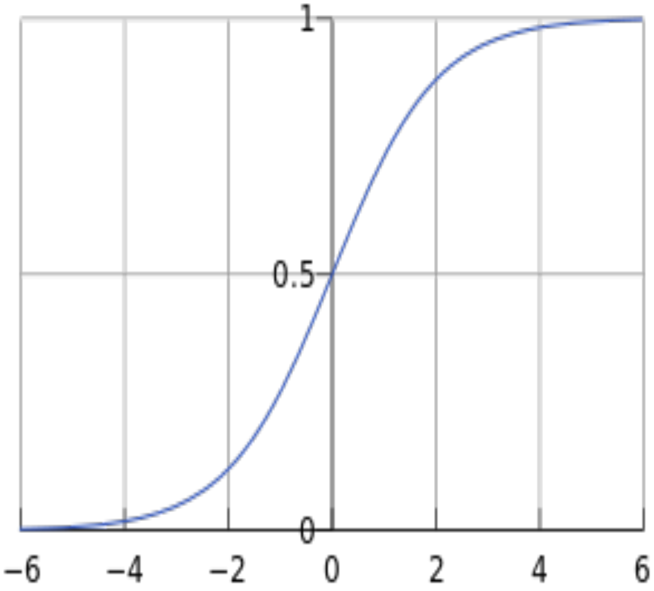
Recap: Machine Learning 101

- Family of functions f_θ
- Tunable parameters θ
- Examples *(x,y)* sampled from unknown data generating distribution *P(x,y)*
- Loss fn *L* compares target y and output $f_\theta(x)$, returns a number
- Regularizer *R* (typically depends on θ but possibly also on *x* & *y*)
- Training criterion for supervised learning:

$$C(\theta) = \text{average}_{(x,y)\sim\text{dataset}} L(f_\theta(x), y) + R(\theta, x, y)$$

- Approximate minimization algorithm to search for good θ

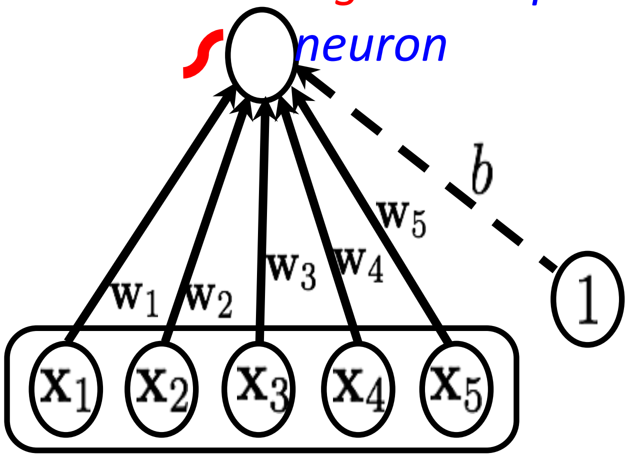
Logistic Regression



- Predict the probability of a **category** *y*, given input *x*
  - *P(Y=y | X=x)*
- Simple extension of linear regression (binary case):
  - *P(Y=1 | X=x)* = sigmoid*(b + w. x)*
- Train by tuning *(b,w)* to maximize average log-likelihood

  Average(***log P(Y=y|X=x)*** **)**

  over training pairs *(x,y)*, by gradient-based optimization
- This is a very **shallow neural network** (no hidden layer)



P(Y=1|x) *logistic output neuron*

input x

Hidden units

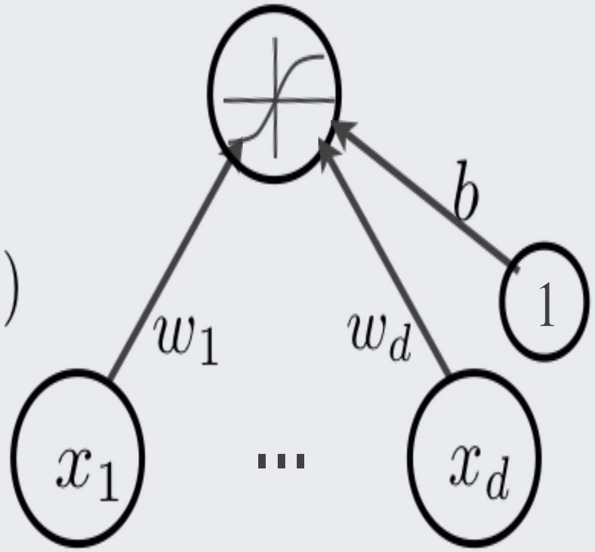
(from Hugo Larochelle)

- Neuron pre-activation (or input activation):

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron (output) activation

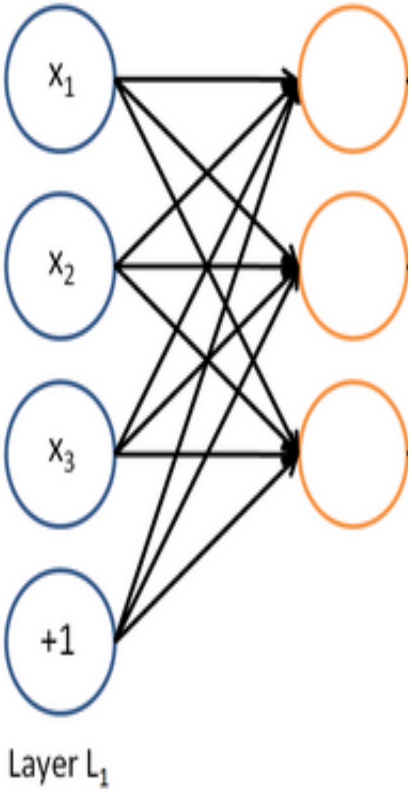
$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$



- $\mathbf{w}$ are the connection weights
- b is the neuron bias
- $g(\cdot)$ is called the activation function

A neural network = running several logistic regressions at the same time

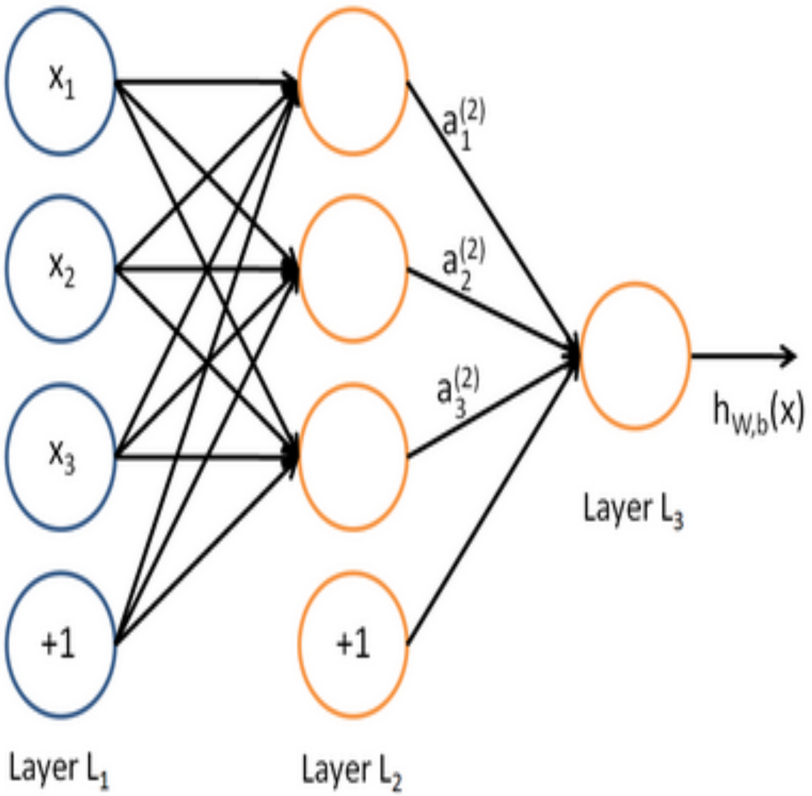
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs



But we don’t have to decide ahead of time what variables these logistic regressions are trying to predict!

A neural network = running several logistic regressions at the same time

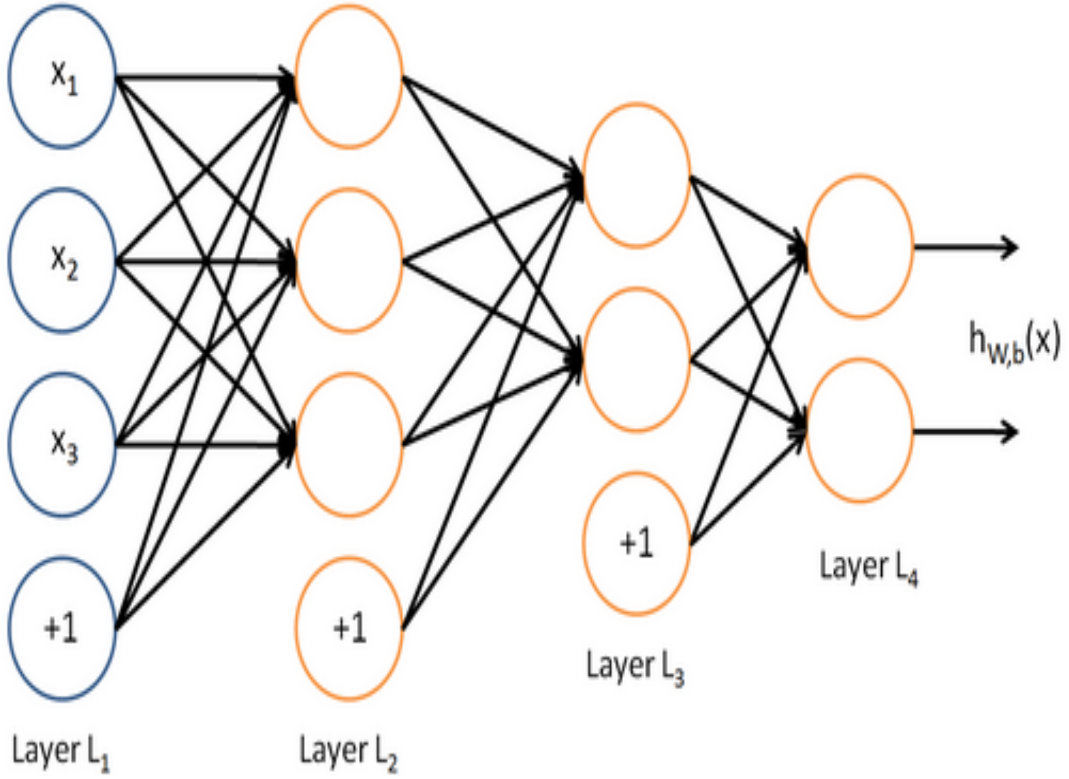
… which we can feed into another logistic regression function



and it is the training criterion that will decide what those intermediate binary target variables should be, so as to make a good job of predicting the targets for the next layer, etc.

A neural network = running several logistic regressions at the same time

- Before we know it, we have a multilayer neural network….

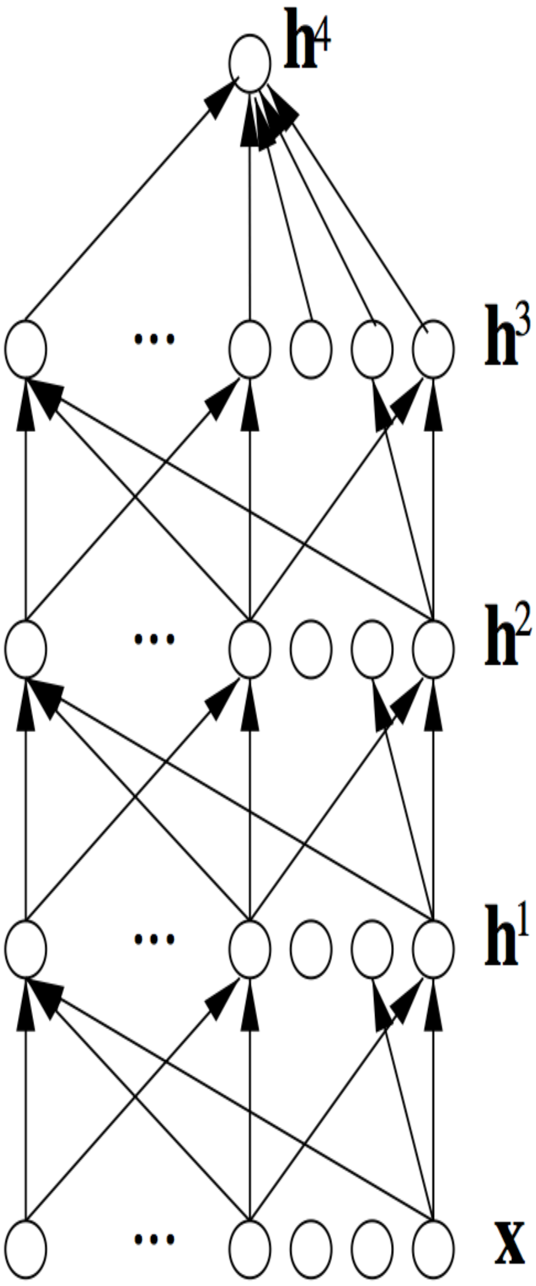


Multilayer network as universal approximator

A series of non-linear transformations of the same type but different parameters

A single but large enough hidden layer yields a **universal approximator**

More layers allow representing more complex functions with less parameters



Universal approximator property does not guarantee

1. easy optimization (low training error is found)
2. good generalization

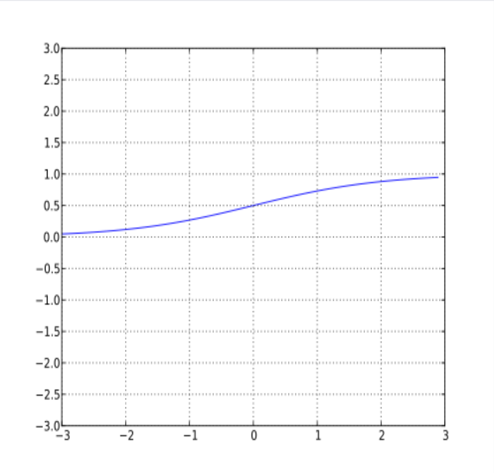
Non-Linearity = activation function

- Stacking linear layers: like one (factorized) linear layer
- Universal approximator : stack linear+nonlinear transformations
- Many types of non-linearities are possible: activation function
    - E.g. linear, sigmoid, tanh, rectifier (ReLU), softmax

- *Breakthrough in 2011: it is much easier to train a deep multilayer network with rectifiers (ReLU) than with sigmoid or tanh, making it possible to train deep nets in a purely supervised way for the first first time (Glorot & Bengio AISTATS 2011)*

Topics: sigmoid activation function

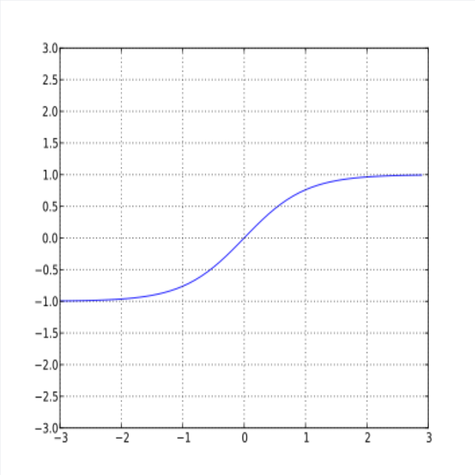
- Squashes the neuron's pre-activation between 0 and 1
- Always positive
- Bounded
- Strictly increasing



$$g(a) = \mathrm{sigm}(a) = \frac{1}{1+\exp(-a)}$$

Topics: hyperbolic tangent ("tanh") activation function

- Squashes the neuron's pre-activation between -1 and 1
- Can be positive or negative
- Bounded
- Strictly increasing



$$g(a) = \tanh(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)} = \frac{\exp(2a)-1}{\exp(2a)+1}$$

Topics: softmax activation function

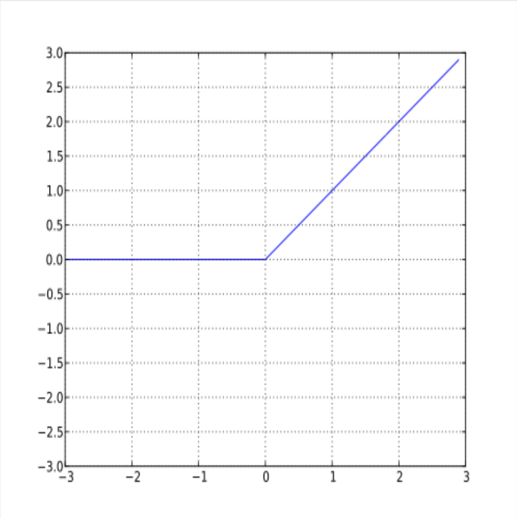
- For multi-class classification:
  - we need multiple outputs (1 output per class)
  - we would like to estimate the conditional probability $p(y=c|\mathbf{x})$
- We use the softmax activation function at the output:

$$\mathbf{o}(\mathbf{a}) = \mathrm{softmax}(\mathbf{a}) = \left[\frac{\exp(a_1)}{\sum_c \exp(a_c)} \dots \frac{\exp(a_C)}{\sum_c \exp(a_c)}\right]^\top$$

  - strictly positive
  - sums to one
- Predicted class is the one with highest estimated probability

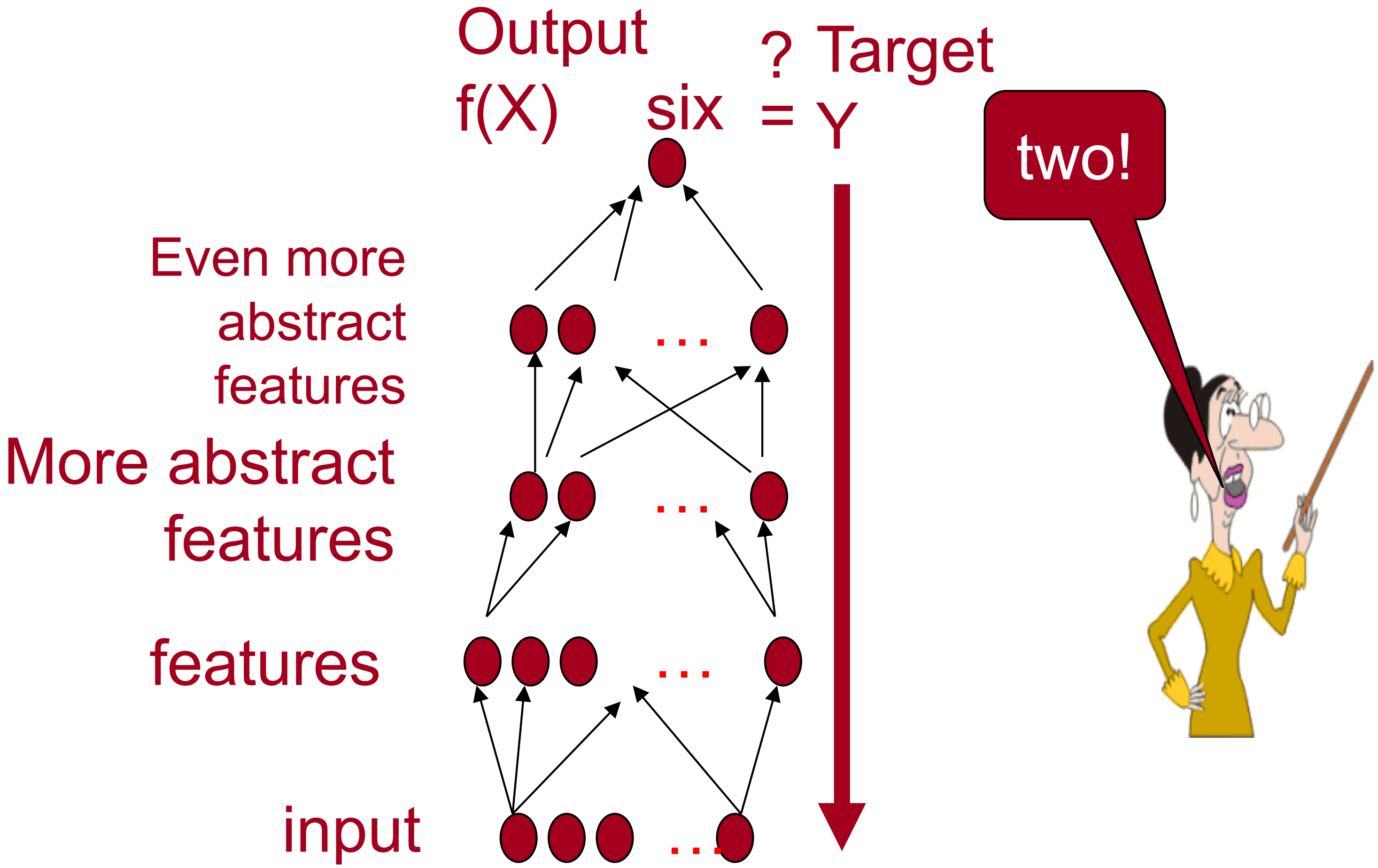
Topics: rectified linear activation function

- Bounded below by 0 (always non-negative)
- Not upper bounded
- Strictly increasing
- Tends to give neurons with sparse activities



$$g(a) = \mathrm{reclin}(a) = \max(0,a)$$

Supervised training of an MLP by backpropagation



Output f(X)

six

? = Target Y

Even more abstract features

More abstract features

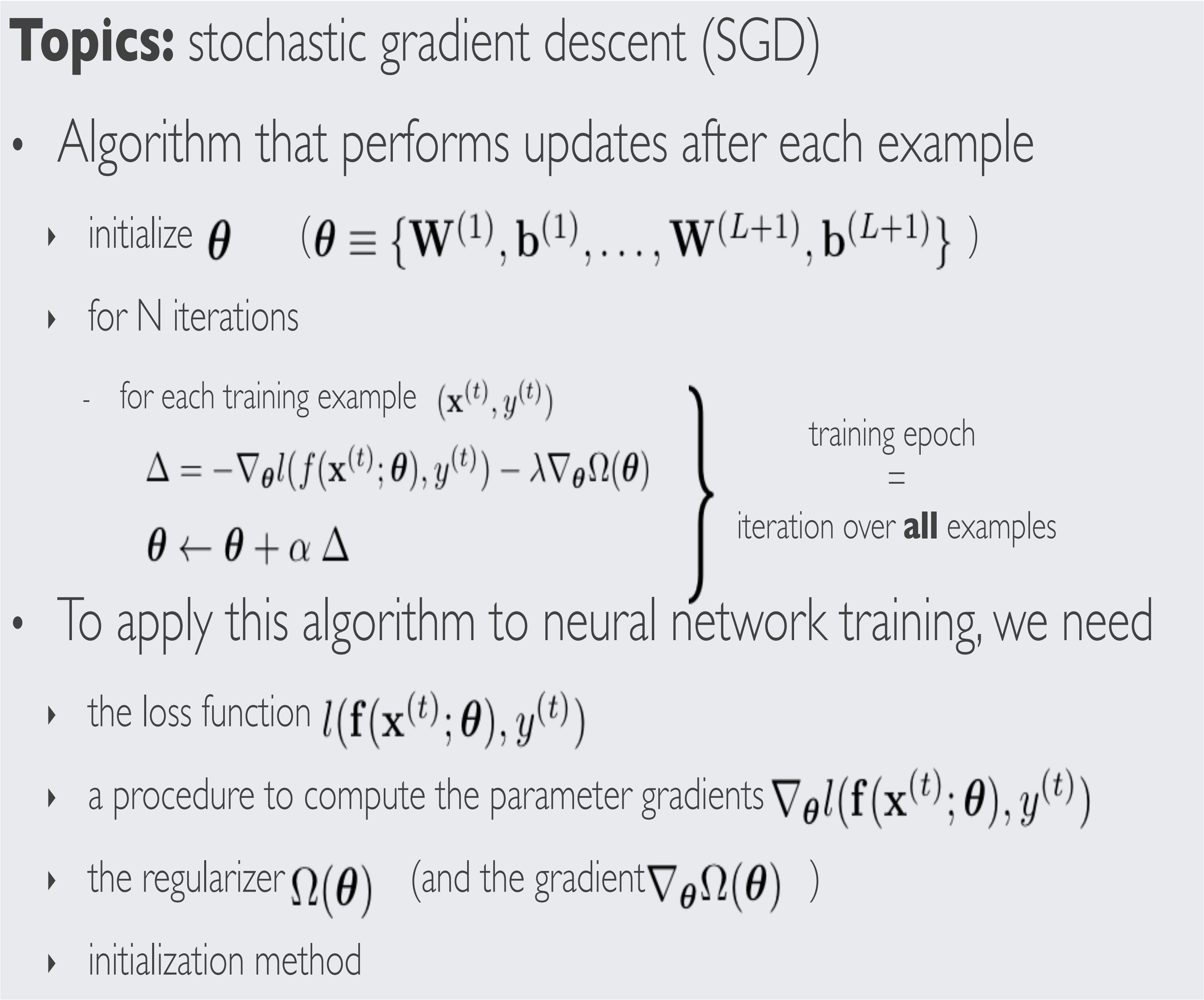
features

input

Requires(*X*,*Y*)=(input,target) pairs as training data

Iterative training by SGD

(from Hugo Larochelle)



Topics: stochastic gradient descent (SGD)

- Algorithm that performs updates after each example
  - initialize $\boldsymbol{\theta}$ $\quad (\boldsymbol{\theta} \equiv \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{W}^{(L+1)}, \mathbf{b}^{(L+1)}\})$
  - for N iterations
    - for each training example $(\mathbf{x}^{(t)}, y^{(t)})$

      $\Delta = -\nabla_{\boldsymbol{\theta}} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$

      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\, \Delta$

training epoch = iteration over **all** examples

- To apply this algorithm to neural network training, we need
  - the loss function $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - a procedure to compute the parameter gradients $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - the regularizer $\Omega(\boldsymbol{\theta})$ (and the gradient $\nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$)
  - initialization method

Motivation for backpropagation: gradient-based optimization

- Knowing how a small change of parameters influences loss *L* tells us how to change the parameters θ

- The gradient $\frac{\partial L}{\partial \theta}$ measures the ratio of error change due to a small parameter change.

- Indicates the best local descent direction!

Why backprop is powerful

- With *n* parameters need O(*n*) computations to obtain *L*
- Also need only O(*n*) computations to obtain gradient by backprop

- Dumb alternative, by finite differences:

$$\frac{\partial L(\theta_i, \theta_{-i})}{\partial \theta_i} \approx \frac{L(\theta_i + \epsilon, \theta_{-i}) - L(\theta_i, \theta_{-i})}{\epsilon}$$

- But that would cost O(n^2) instead of O(*n*) by backprop!

Confusion on the word BACKPROP

- **Backprop**: the backward accumulation procedure to compute gradients efficiently wrt a scalar (the loss)

- NOT THE SAME THING AS **gradient descent**, nor the MLP architecture.

- Backprop is **not just used for supervised learning**: also for unsupervised learning and RL, with different losses

Back-Prop & Chain Rule

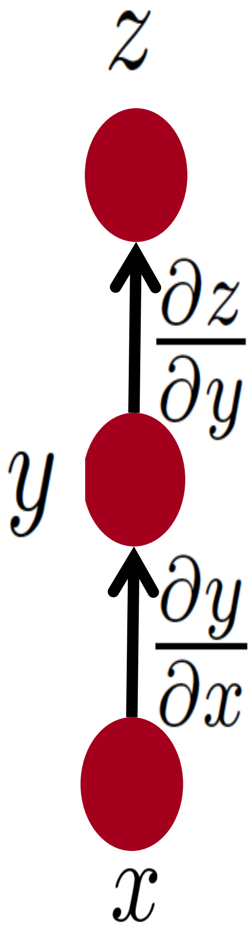
- Compute gradient of example-wise loss wrt parameters, by considering **intermediate values** such as the outputs of neurons

- **Simply applying the derivative chain rule wisely**

$$z = f(y) \quad y = g(x) \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

Chain Rule

Also works if all these quantities are tensors, using the appropriate tensor products



z
$\frac{\partial z}{\partial y}$
y
$\frac{\partial y}{\partial x}$
x

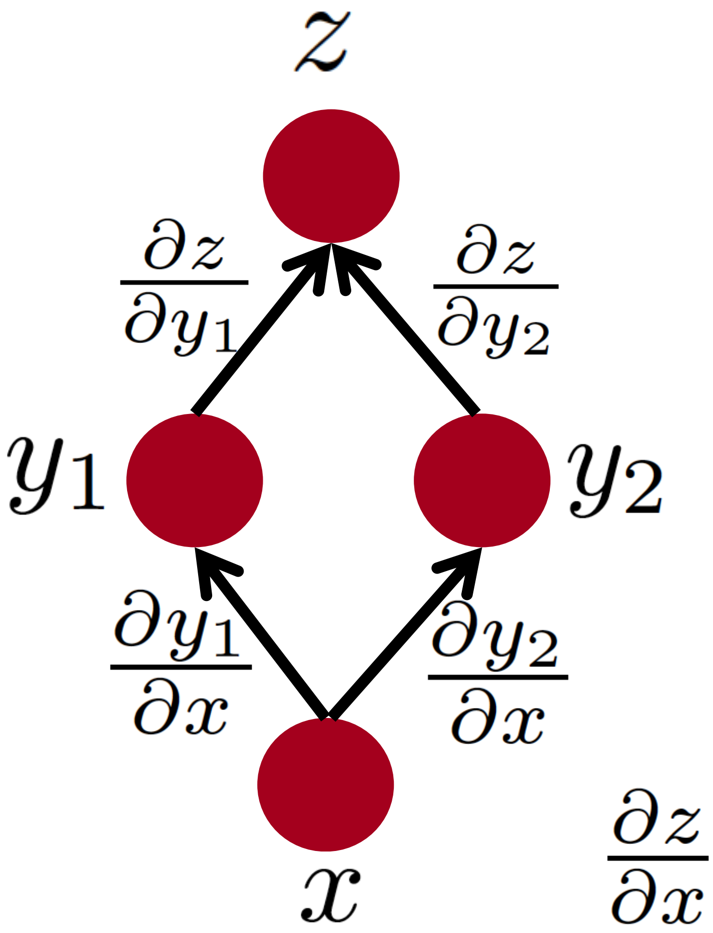
$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$

$$\Delta y = \frac{\partial y}{\partial x} \Delta x$$

$$\Delta z = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \Delta x$$

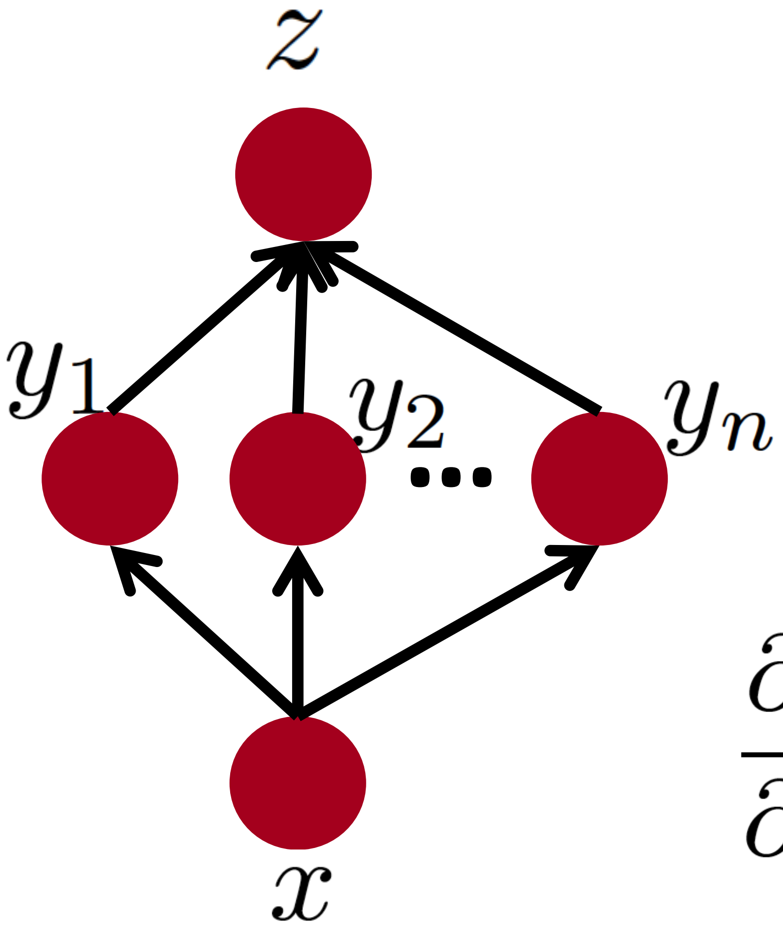
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

Multiple Paths Chain Rule


z
$\frac{\partial z}{\partial y_1}$
$\frac{\partial z}{\partial y_2}$
y_1
y_2
$\frac{\partial y_1}{\partial x}$
$\frac{\partial y_2}{\partial x}$
x

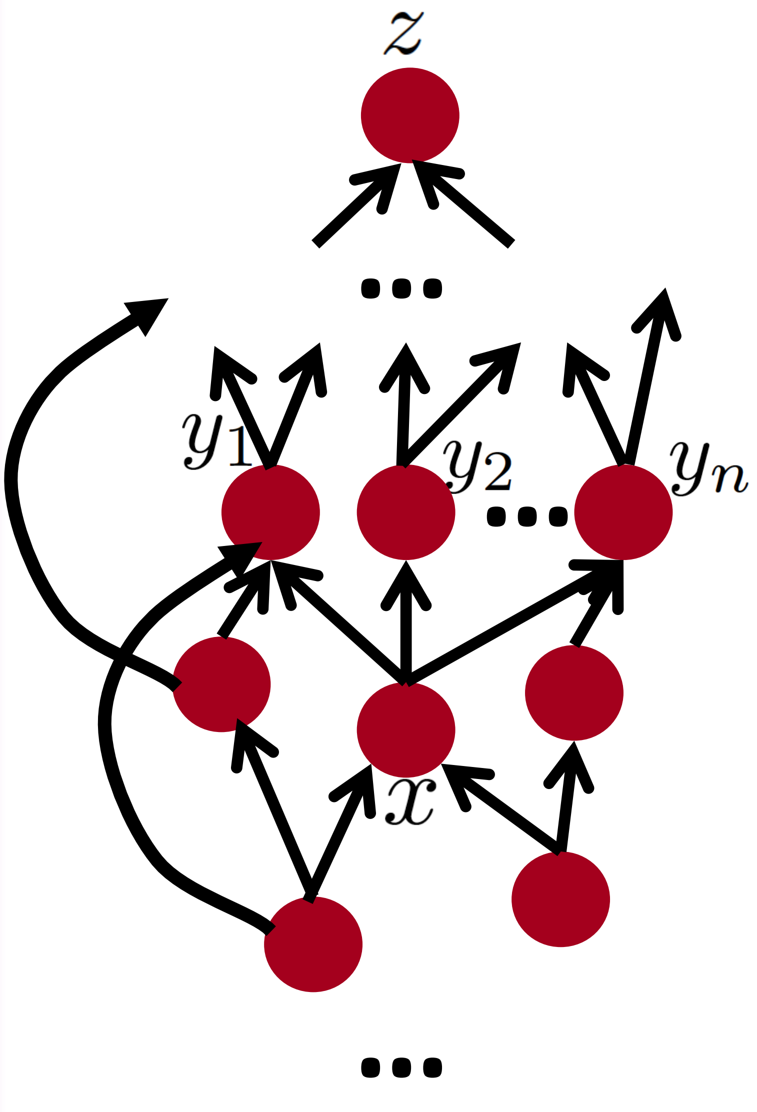
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

Multiple Paths Chain Rule - General



$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

Chain Rule in Flow Graph



Flow graph: any directed acyclic graph

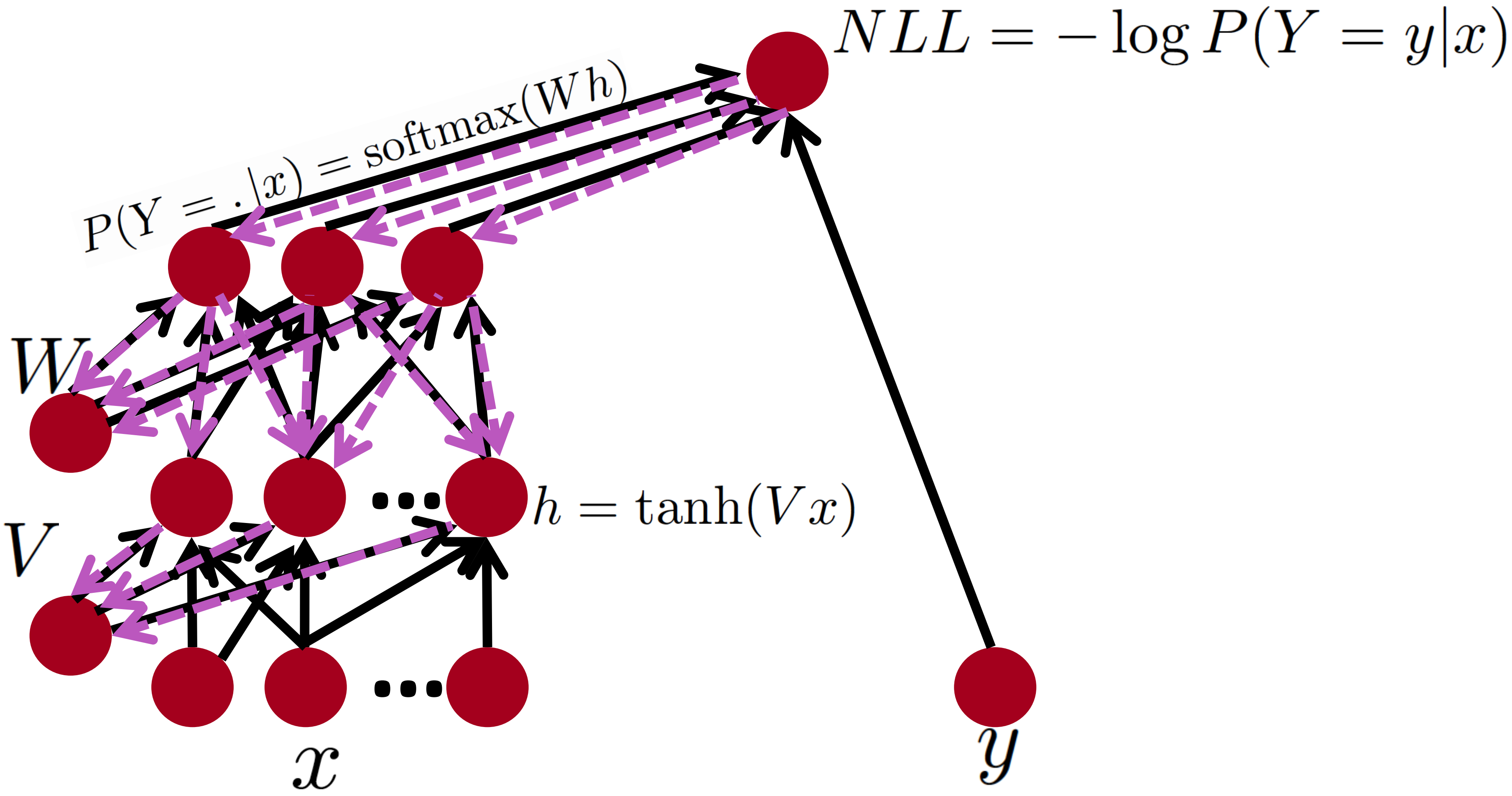
node = computation result

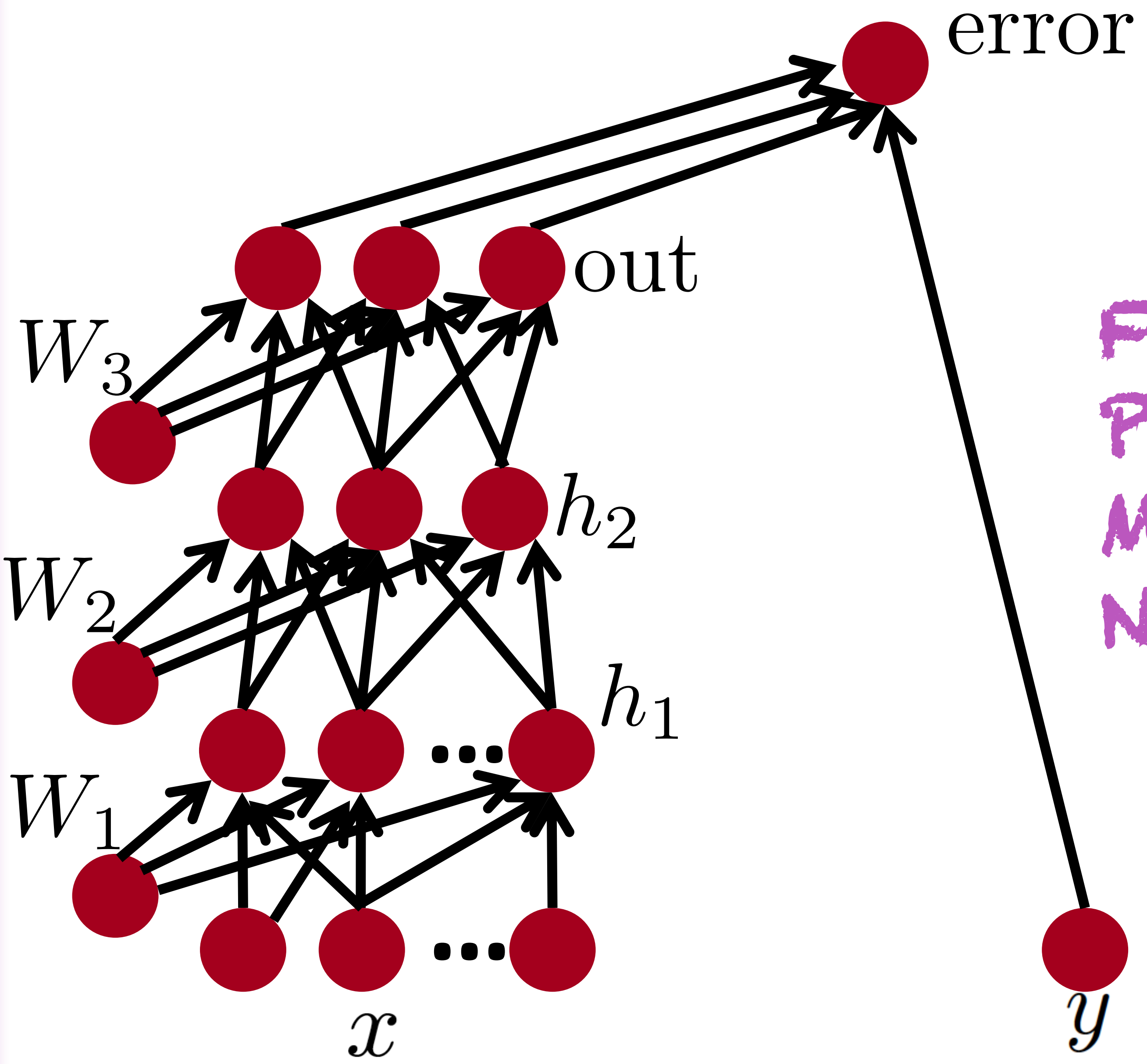
arc = computation dependency

$\{y_1,\ y_2,\ \ldots\ y_n\}$ = successors of x

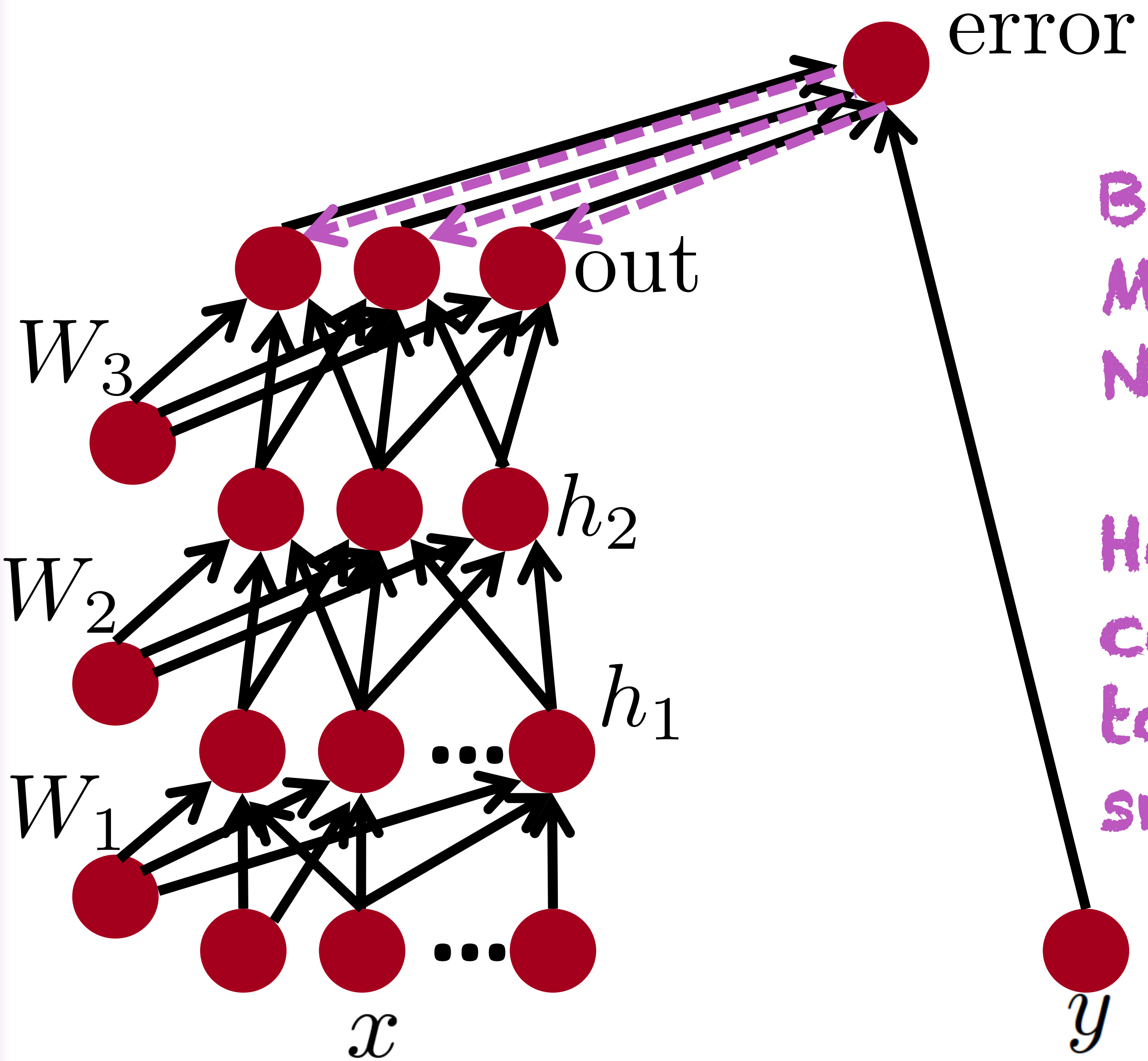
$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x}$$

Back-Prop in Multi-Layer Net



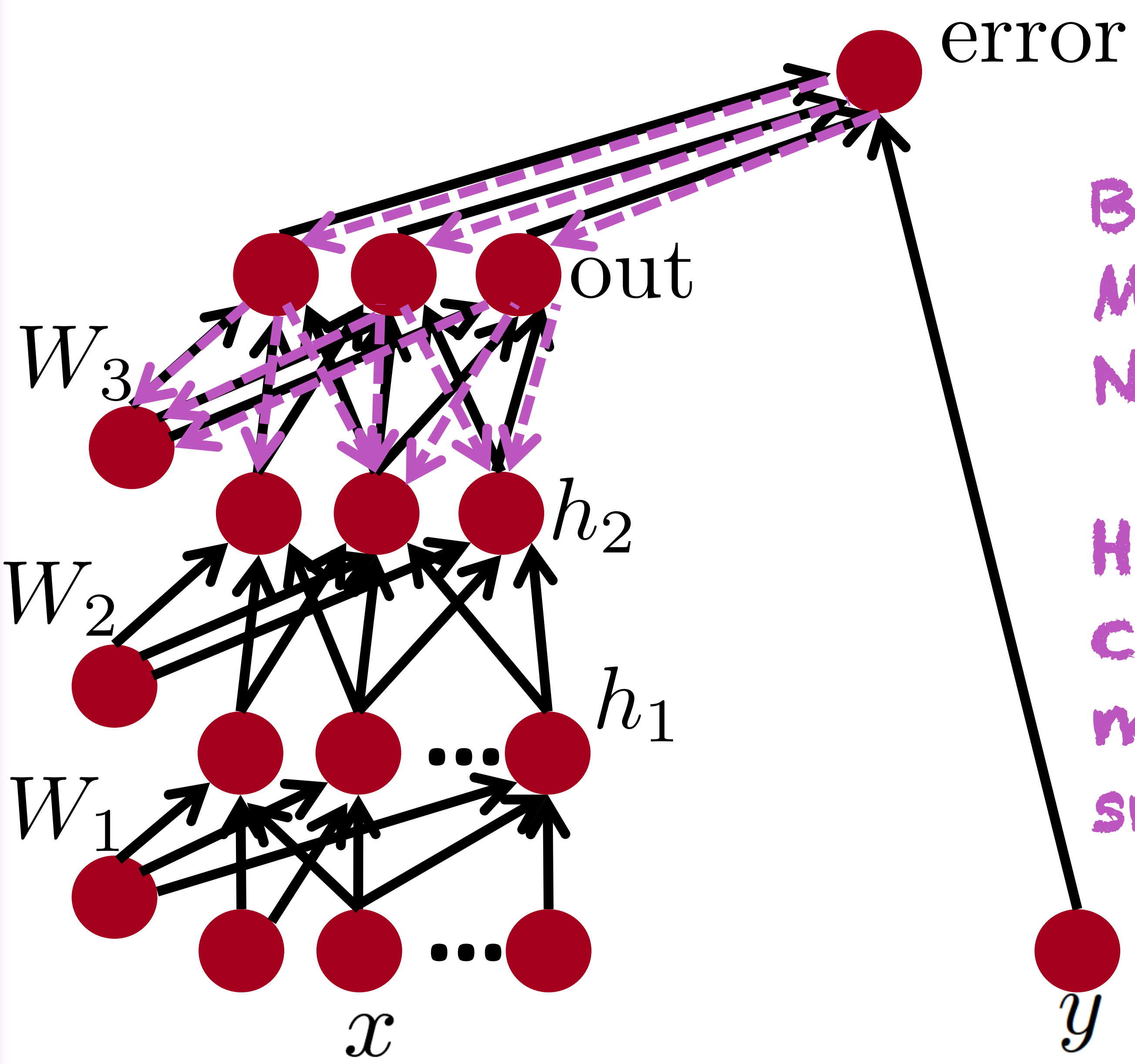


Forward-Prop in Multi-Layer Net



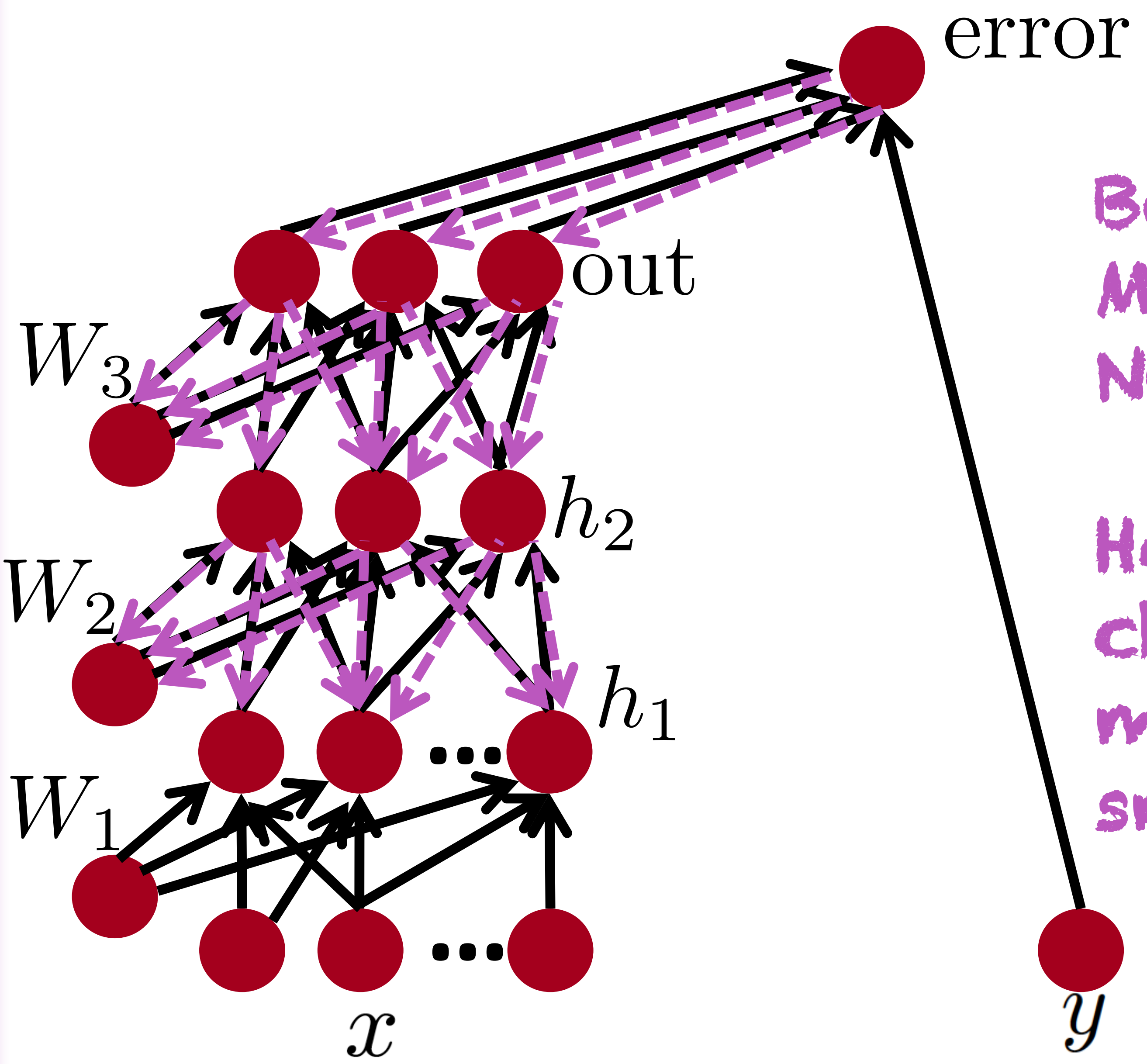
Backprop in Multi-Layer Net:

How outputs could change to make error smaller



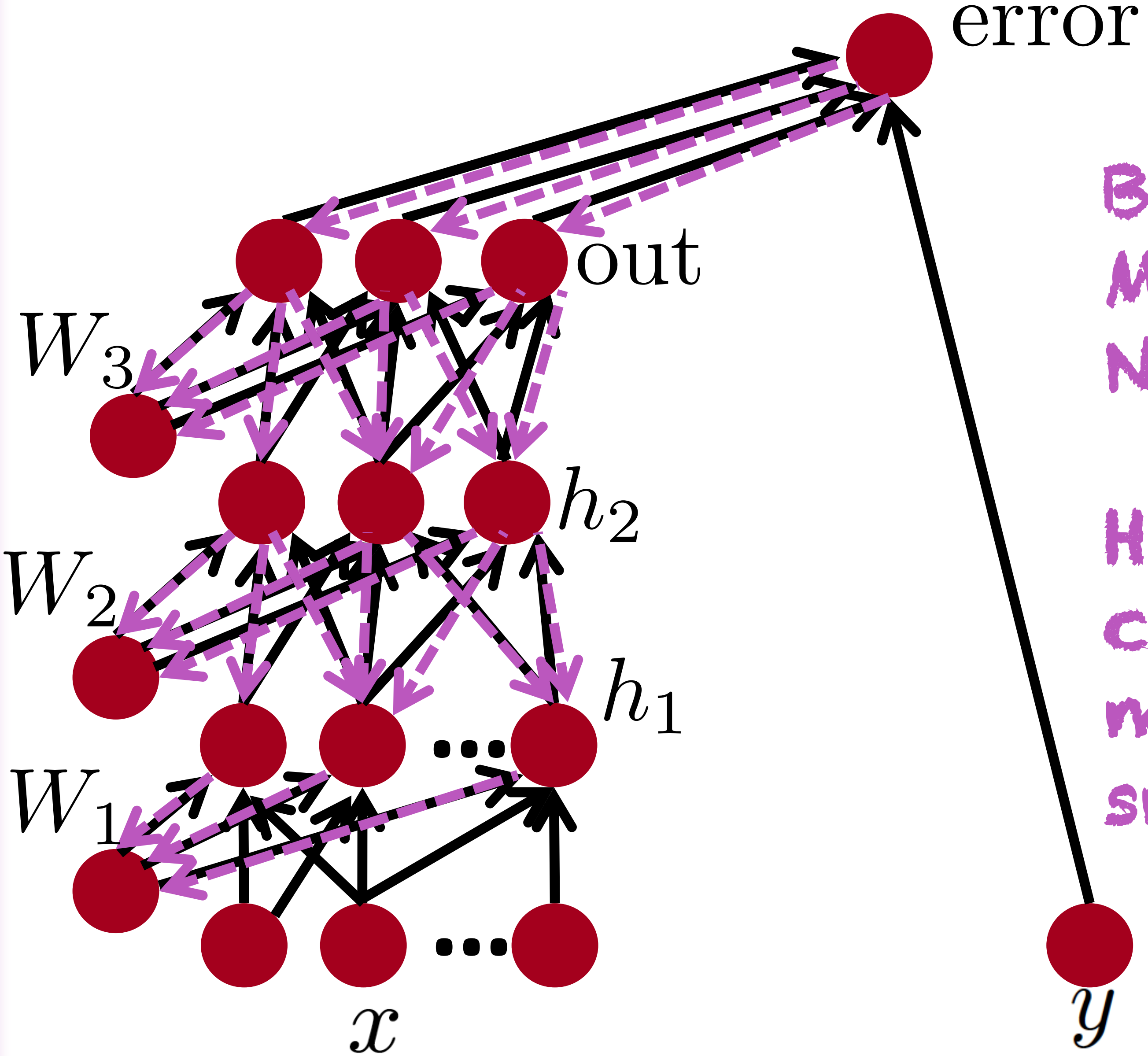
Backprop in Multi-Layer Net:

How h_2 could change to make error smaller



Backprop in Multi-Layer Net:

How h_1 could change to make error smaller

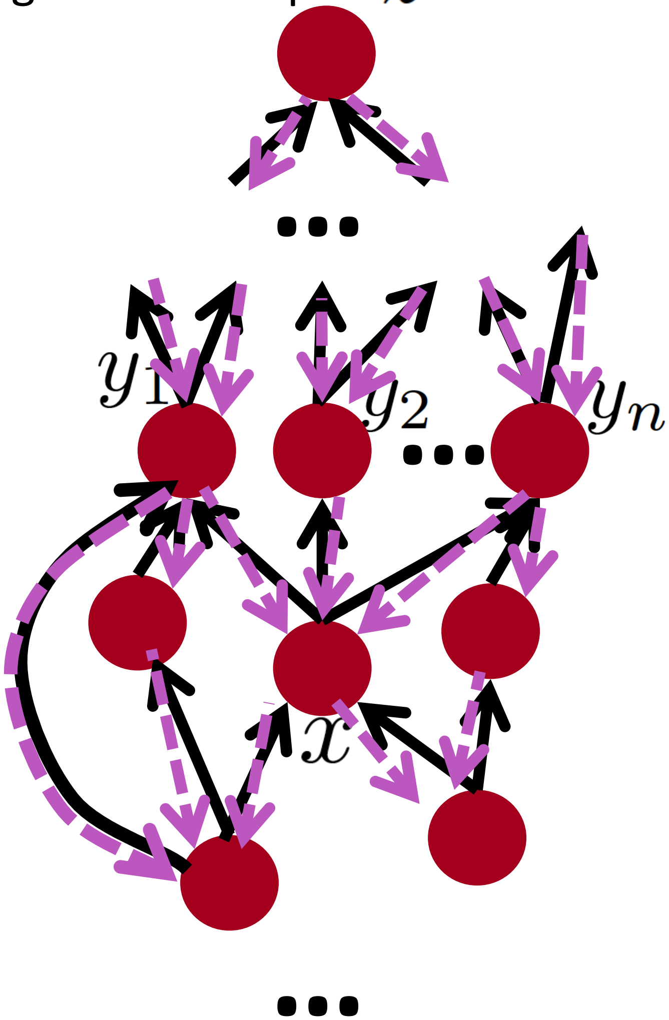


Backprop in Multi-Layer Net:

How W_1 could change to make error smaller

Back-Prop in General Flow Graph

Single scalar output z



1. Fprop: visit nodes in topo-sort order
   - Compute value of node given predecessors
2. Bprop:
   - initialize output gradient = 1
   - visit nodes in reverse order:

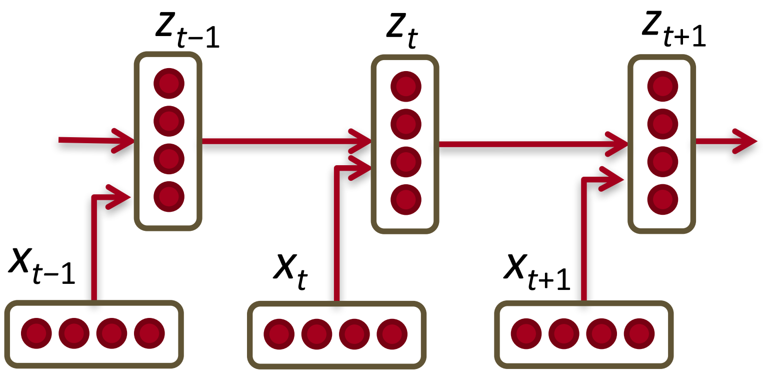
     Compute gradient wrt each node using gradient wrt successors

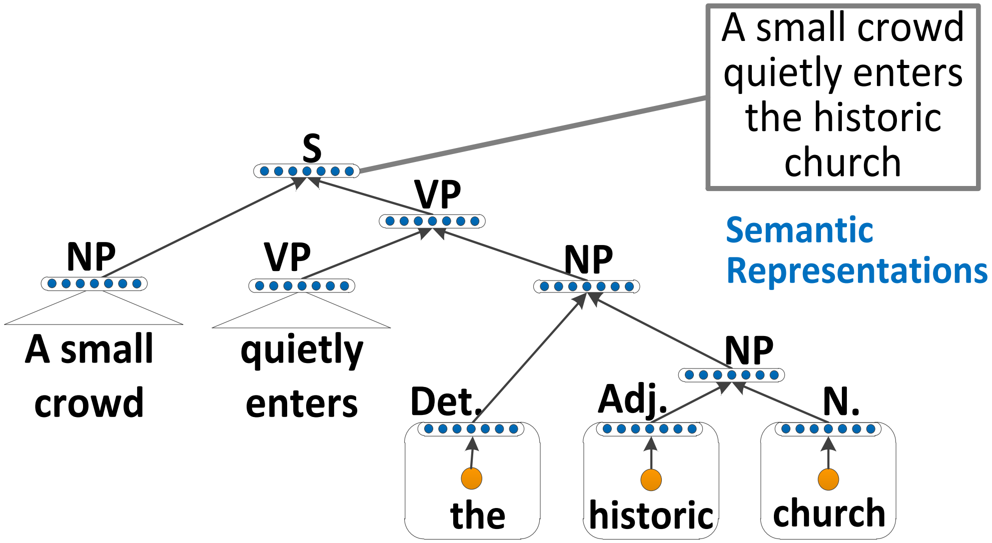
$\{y_1,\ y_2,\ \ldots\ y_n\}$ = successors of x

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

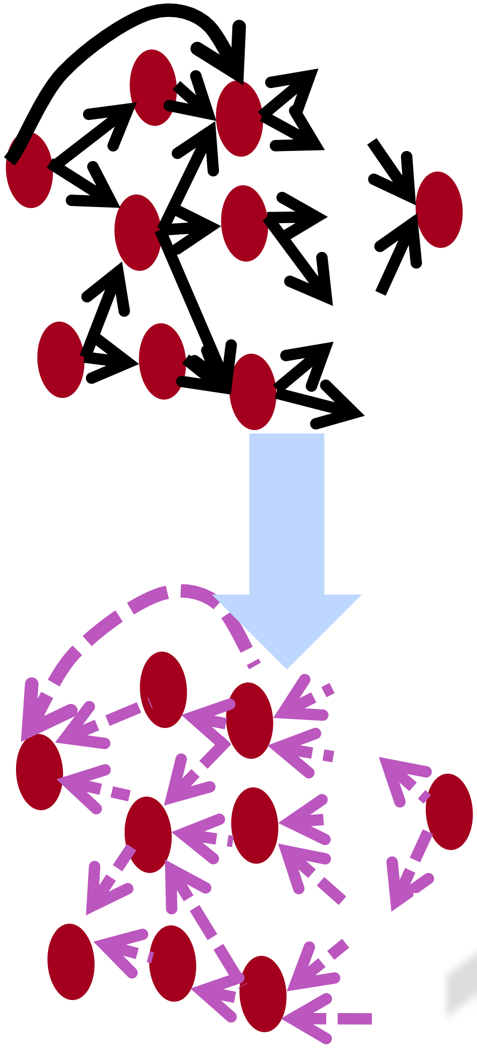
Back-Prop in Recurrent & Recursive Nets

- Replicate a parameterized function over different time steps or nodes of a DAG
- Output state at one time-step / node is used as input for another time-step / node





Automatic Differentiation



- The gradient computation can be automatically inferred from the symbolic expression of the fprop.
- Each node type needs to know how to compute its output and how to compute the gradient wrt its inputs given the gradient wrt its output

Easy and fast prototyping

theano TensorFlow PYTORCH

Batch Normalization (Ioffe & Szegedy 2015)

- Helps training by reparametrization which improves condition number, helps generalization by acting as a regularizer
- Other normalization methods proposed since then

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

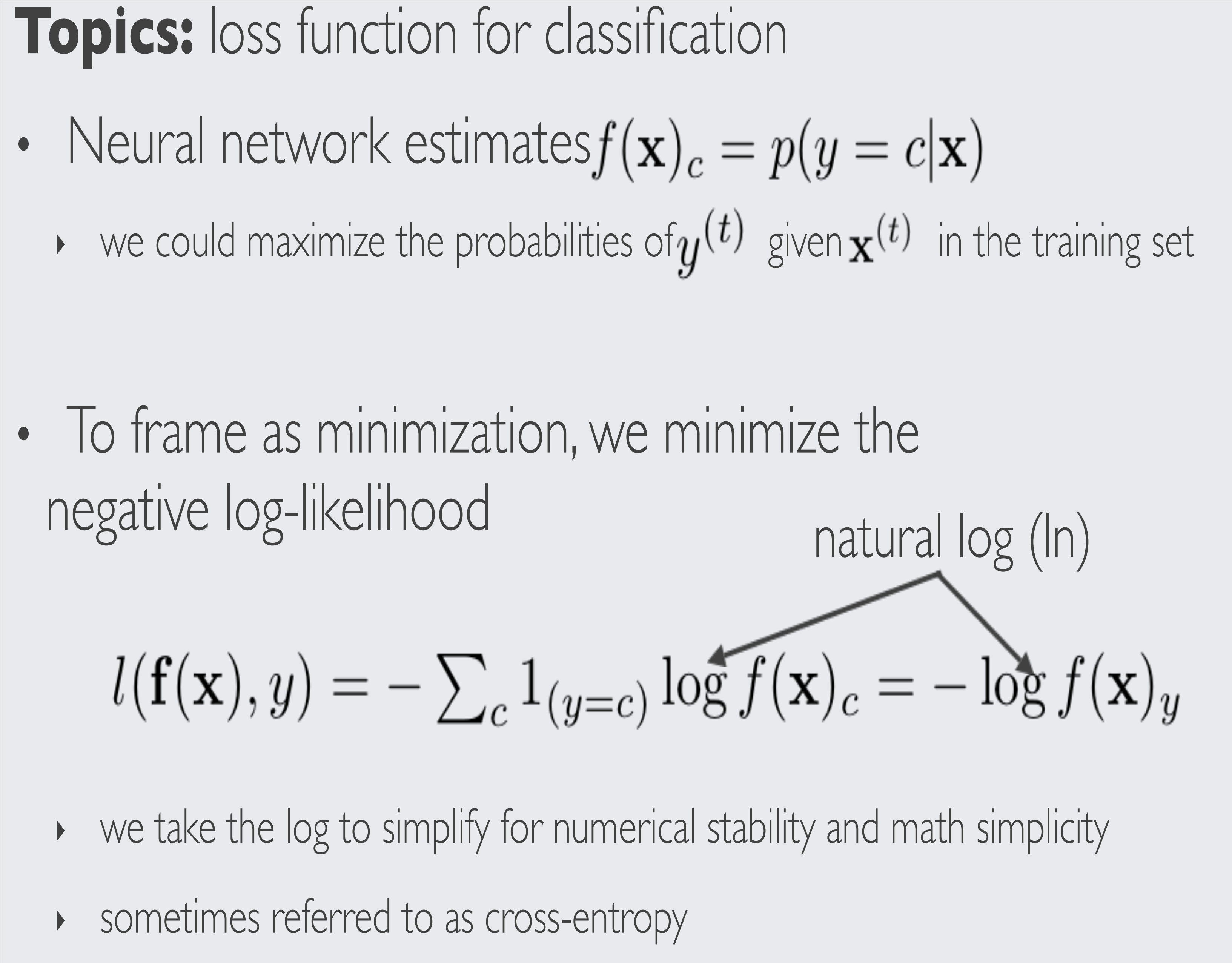
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Log-likelihood as loss function

(from Hugo Larochelle)



Topics: loss function for classification

- Neural network estimates $f(\mathbf{x})_c = p(y = c|\mathbf{x})$
  - we could maximize the probabilities of $y^{(t)}$ given $\mathbf{x}^{(t)}$ in the training set
- To frame as minimization, we minimize the negative log-likelihood

natural log (ln)

$$l(\mathbf{f}(\mathbf{x}), y) = -\sum_c 1_{(y=c)} \log f(\mathbf{x})_c = -\log f(\mathbf{x})_y$$

  - we take the log to simplify for numerical stability and math simplicity
  - sometimes referred to as cross-entropy

Log-Likelihood for Neural Nets

- Estimating a conditional probability $P(Y|X)$
- Parametrize it by $P(Y|X) = P(Y|\omega = f_\theta(X))$
- Loss = $-\log P(Y|X)$
- E.g. Gaussian *Y*, $\omega = (\mu, \sigma)$

  typically only μ is the network output, depends on *X*

  Equivalent to MSE criterion:

  Loss = $-\log P(Y|X) = \log \sigma + ||f_\theta(X) - Y||^2/\sigma^2$
- E.g. Multinoulli *Y* for classification,

  $$\omega_i = P(Y = i|x) = f_{\theta,i}(X) = \text{softmax}_i(a(X))$$

  Loss = $-\log \omega_Y = -\log f_{\theta,Y}(X)$

Multiple Output Variables

- If they are conditionally independent (given X), the individual prediction losses add up:

$$-\log P(Y|X) = -\log P(Y_1, \ldots Y_k|X) = -\log \prod_i P(Y_i|X) = -\sum_i \log P(Y_i|X)$$

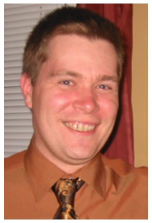
- Likelihood if some *Y*$_i$'s are missing: just ignore those losses

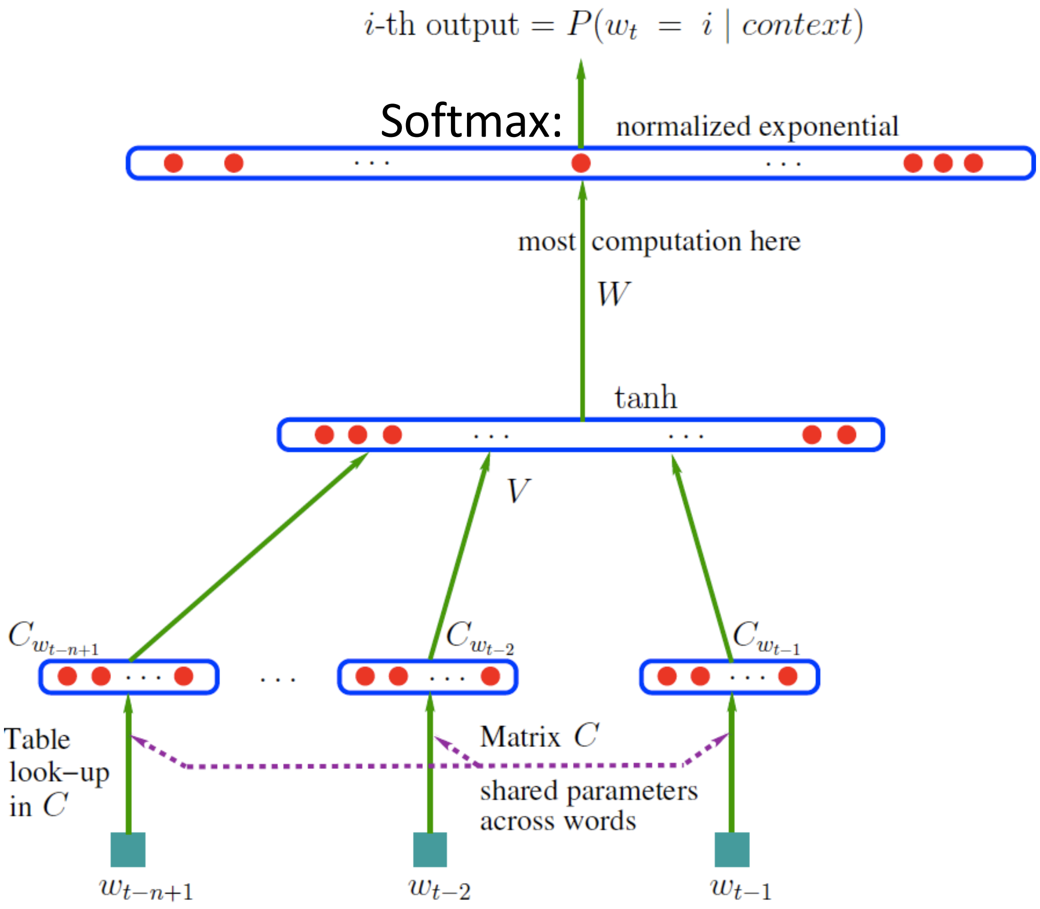
- If not conditionally independent, need to capture the conditional joint distribution $P(Y_1, \ldots Y_k|X)$
  - Example: output = image, sentence, tree, etc.
  - Similar to unsupervised learning problem of capturing joint
  - Exact likelihood may similarly be intractable, depending on model

Combining Representations

Neural Language Models

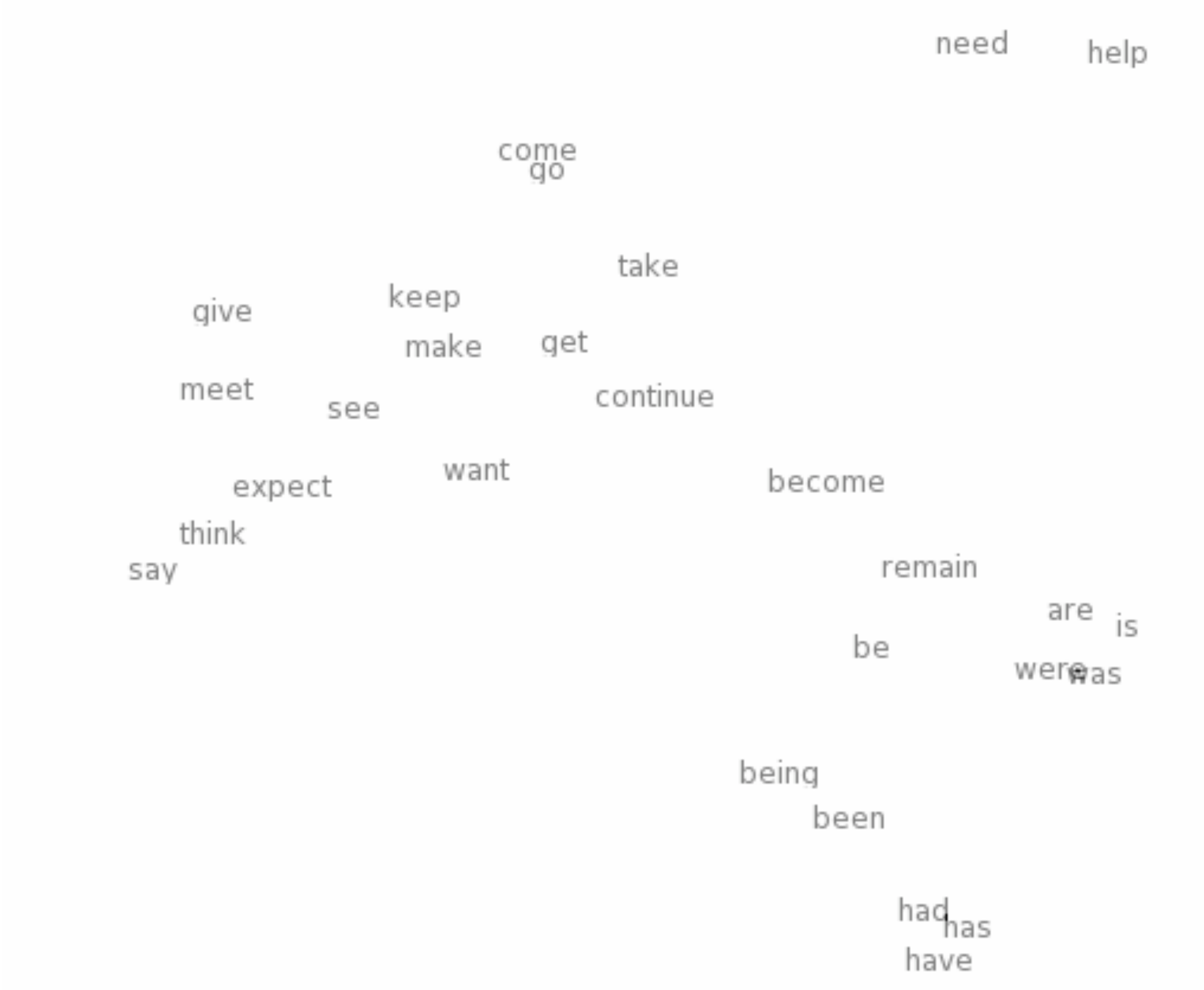
- *Bengio et al NIPS'2000 and JMLR 2003 "A Neural Probabilistic Language Model"*
  - Each word represented by a distributed continuous-valued code vector = embedding
  - Generalizes to sequences of words that are semantically similar to training sequences





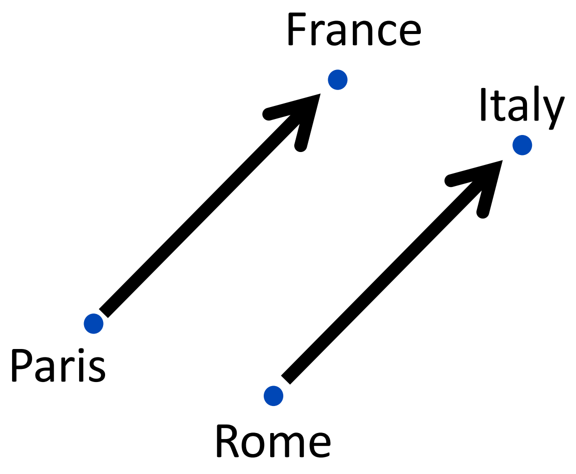
$$P(w_1, w_2, w_3, \ldots, w_T) = \prod_t P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

Neural word embeddings - visualization



Analogical Representations for Free (Mikolov et al, ICLR 2013)

- Semantic relations appear as linear relationships in the space of learned representations
- King – Queen ≈ Man – Woman
- Paris – France + Italy ≈ Rome



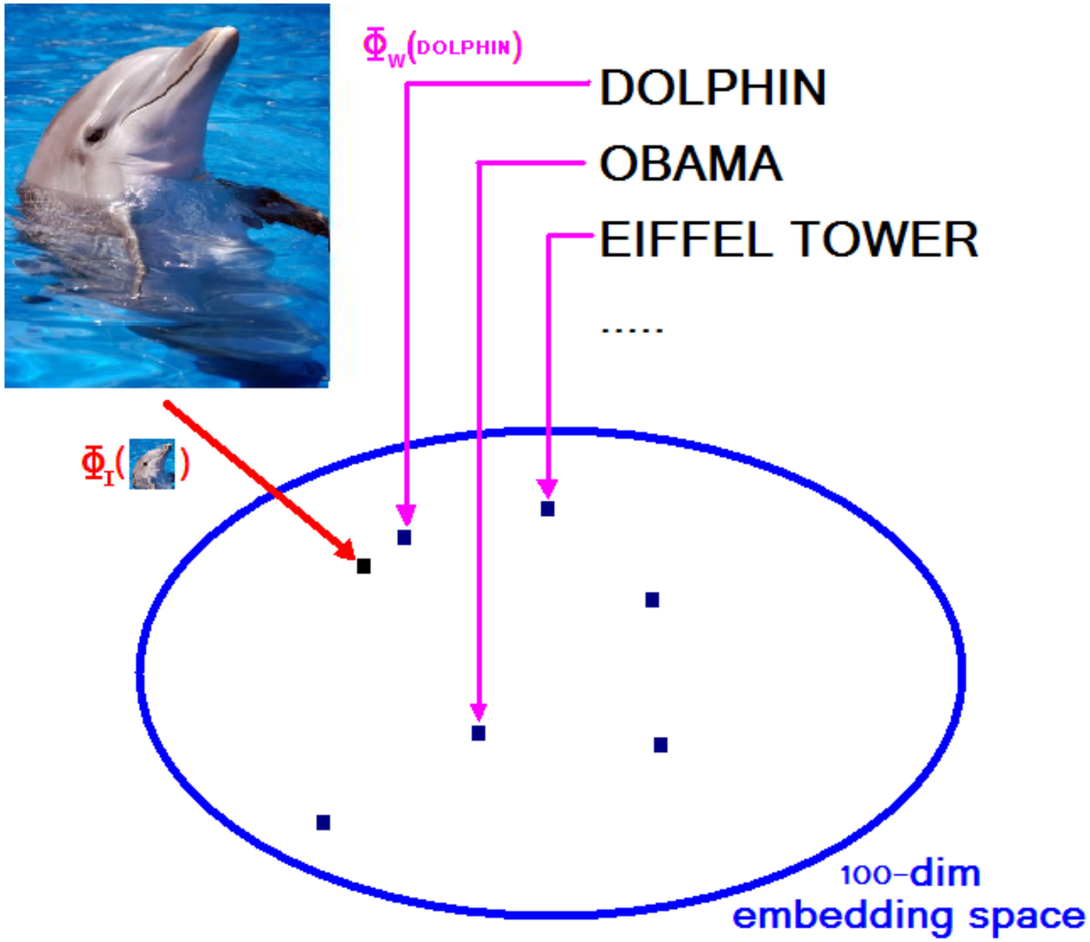
Google Image Search:

Different object types represented in the same space

Google:

S. Bengio, J. Weston & N. Usunier

(IJCAI 2011, NIPS'2010, JMLR 2010, MLJ 2010)



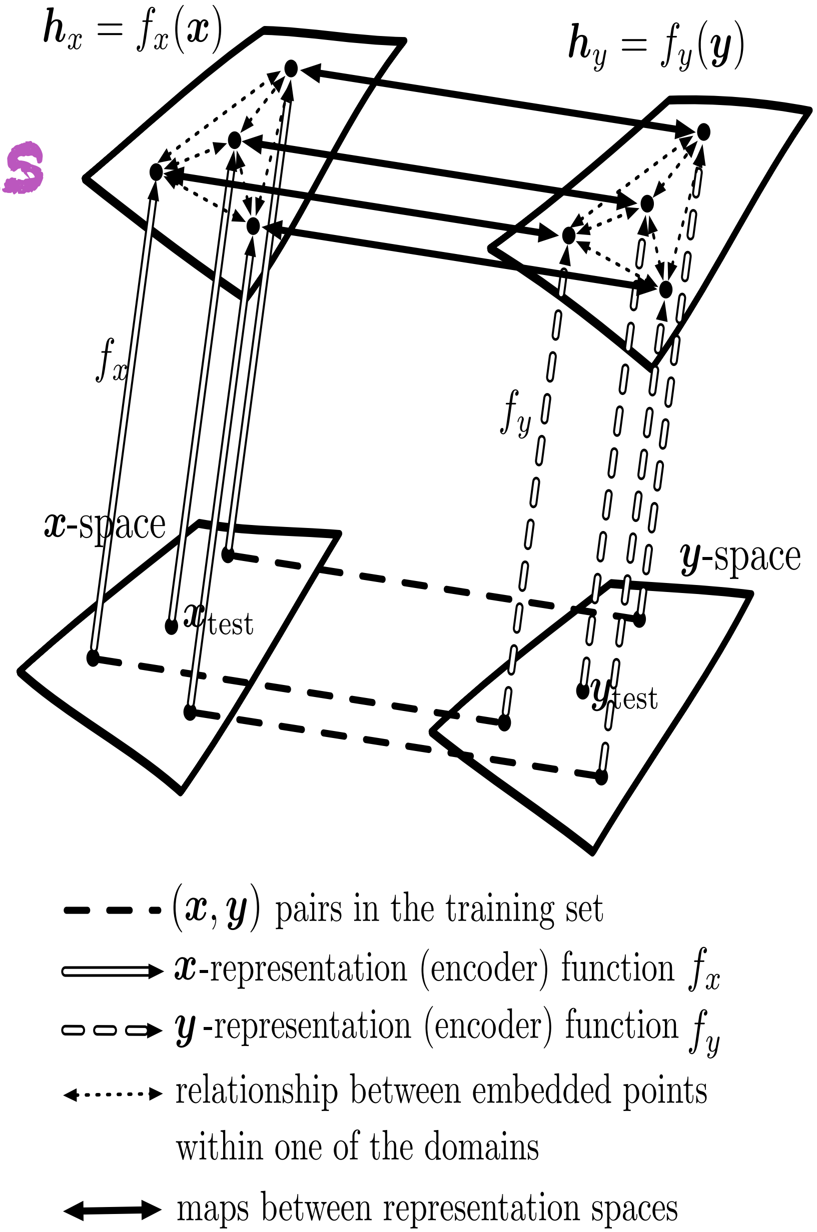
Learn $\Phi_I(\cdot)$ and $\Phi_W(\cdot)$ to optimize precision@k.

Maps Between Representations

x and *y* represent different modalities, e.g., image, text, sound…

Can provide 0-shot generalization to new categories (values of *y*)

(Larochelle et al AAAI 2008)



$\boldsymbol{h}_x = f_x(\boldsymbol{x})$
$\boldsymbol{h}_y = f_y(\boldsymbol{y})$
f_x
f_y
$\boldsymbol{x}$-space
$\boldsymbol{y}$-space
$\boldsymbol{x}_{\text{test}}$
$\boldsymbol{y}_{\text{test}}$
$(\boldsymbol{x}, \boldsymbol{y})$ pairs in the training set
$\boldsymbol{x}$-representation (encoder) function f_x
$\boldsymbol{y}$-representation (encoder) function f_y
relationship between embedded points within one of the domains
maps between representation spaces

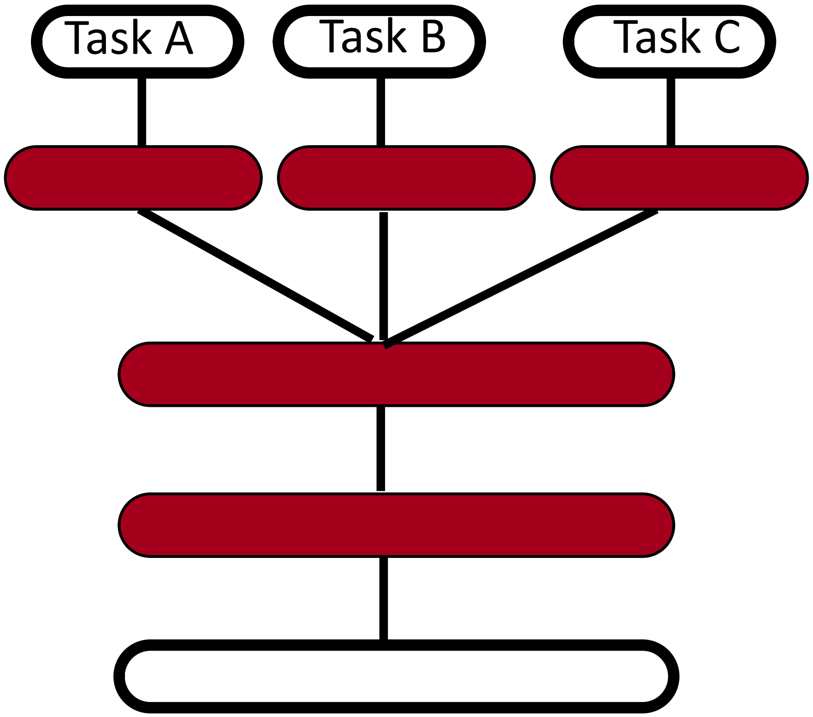
Multi-Task Learning

- Generalizing better to new tasks (tens of thousands!) is crucial to approach AI

- Deep architectures learn good intermediate representations that can be shared across tasks

  (Collobert & Weston ICML 2008, Bengio et al AISTATS 2011)

- Good representations that disentangle underlying factors of variation make sense for many tasks because **each task concerns a subset of the factors**



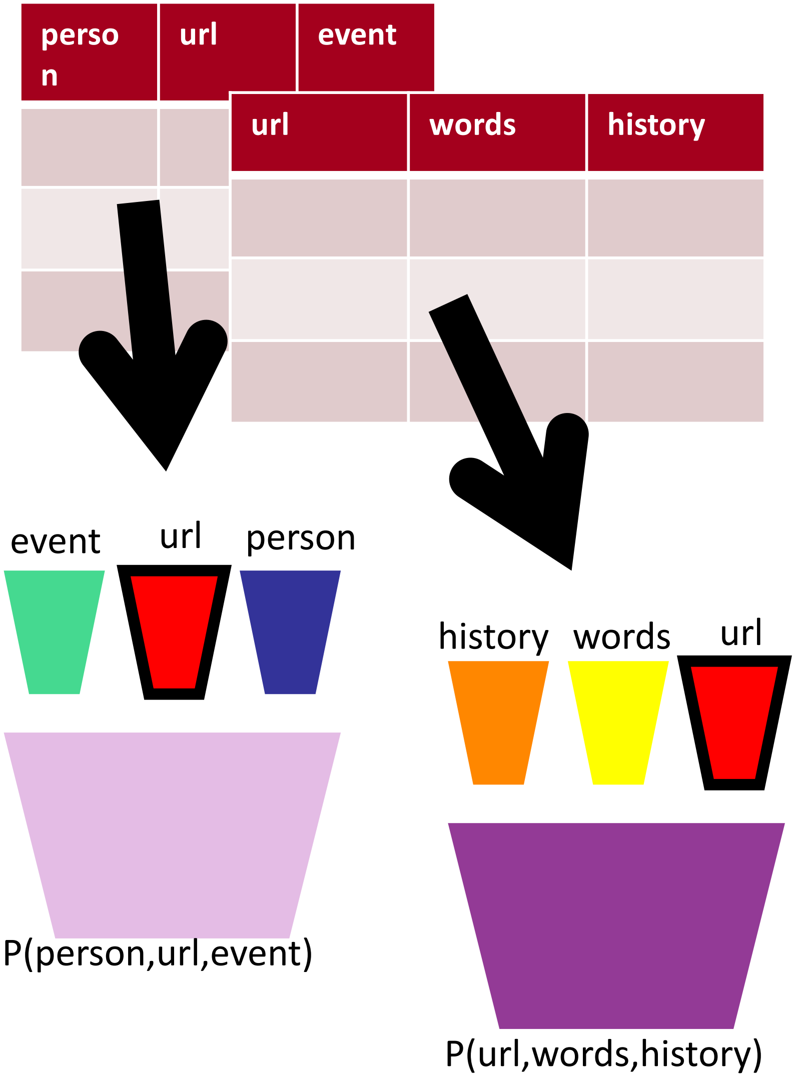
E.g. dictionary, with intermediate concepts re-used across many definitions

Prior: shared underlying explanatory factors between tasks

Combining Multiple Sources of Evidence with Shared Representations

- Traditional ML: data = matrix
- Relational learning: multiple sources, different tuples of variables
- Share representations of same types across data sources
- Shared learned representations help propagate information among data sources: e.g., WordNet, XWN, Wikipedia, **FreeBase**, ImageNet…(Bordes et al AISTATS 2012, ML J. 2013)

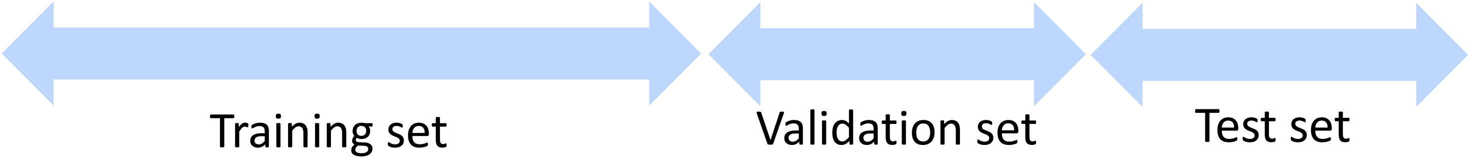
- **FACTS = DATA**
- **Deduction = Generalization**



Hyper-Parameters & Meta-Learning

Hyper-parameters & validation set

- Parameters: optimized by gradient-based optimization on the training set
- Hyper-parameters: design decisions and settings of the optimization procedure
  - Optimized based on performance on a validation set disjoint from training set.
- Choosing hyper-parameters based on training set would lead to high-capacity choices with overfitting (hence need a validation set)
- A disjoint test set is used to obtain final unbiased estimation of generalization performance.
- Training, validation and test sets are subsets of randomized (shuffled) data, to mimic iid assumption



Hyper-parameters of MLPs

- **Global learning rate**
- Number of training epochs (passes over training set)
- Number of neurons per layer
- Depth (number of layers)
- Choice of activation function(s)
- Regularization coefficients (L1, L2, etc.)
- Noise injection & dropout
- Loss function and output non-linearity
- Minibatch size (with parallel computation within minibatch)
- Weight normalization method (e.g. batch normalization)
- Input and targets normalization
- Data deformations
- Etc.

Nested optimisation of parameters and hyper-parameters

- For each considered configuration of hyper-parameters
  - Train parameters with this configuration (optimize train loss)
  - Measure resulting model’s validation error
  - Keep this configuration if it’s the best seen up to now
- An old form of **meta-learning**: two nested optimizations

- Optionally: Retrain with training+validation set

- Measure resulting model’s test error

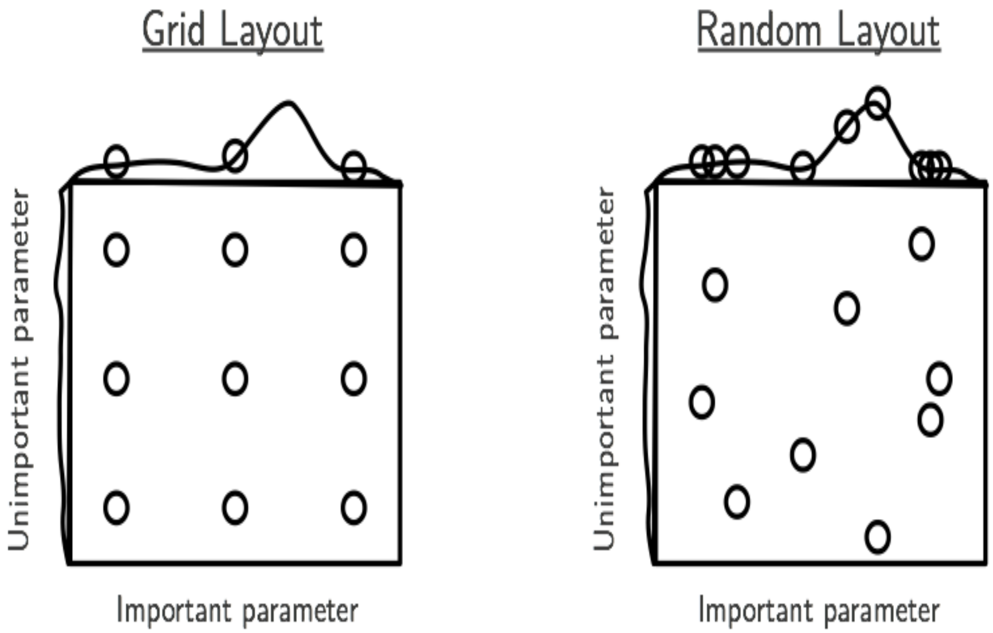
Hyper-Optimization

- Manual search
  - Don’t use test error!
  - Slow and sequential, but trained humans still generally do it.
  - Not systematic, harder to reproduce
- Grid search: inefficient with more than 2 hyper-parameters
- Random search *(Bergstra & Bengio, 2012, JMLR)*
  - *Simple, robust & parallelizable*
- Bayesian optimisation (sequential, automated), reinforcement learning

Random Sampling of Hyperparameters

(Bergstra & Bengio 2012)

- Random search: simple & efficient
  - Independently sample each HP, e.g. l.rate~exp(U[log(.1),log(.0001)])
  - Each training trial is iid
  - If a HP is irrelevant grid search is wasteful
  - More convenient: ok to early-stop, continue



L1 regularisation to remove weights and inputs

Add a term that pushes weights or groups of weights to 0

prediction error + $\lambda \sum_{ij} |W_{ij}|$

pushes individual weights to 0, whereas

prediction error + $\lambda \sum_{i} \sqrt{\sum_{j} W_{ij}^2}$

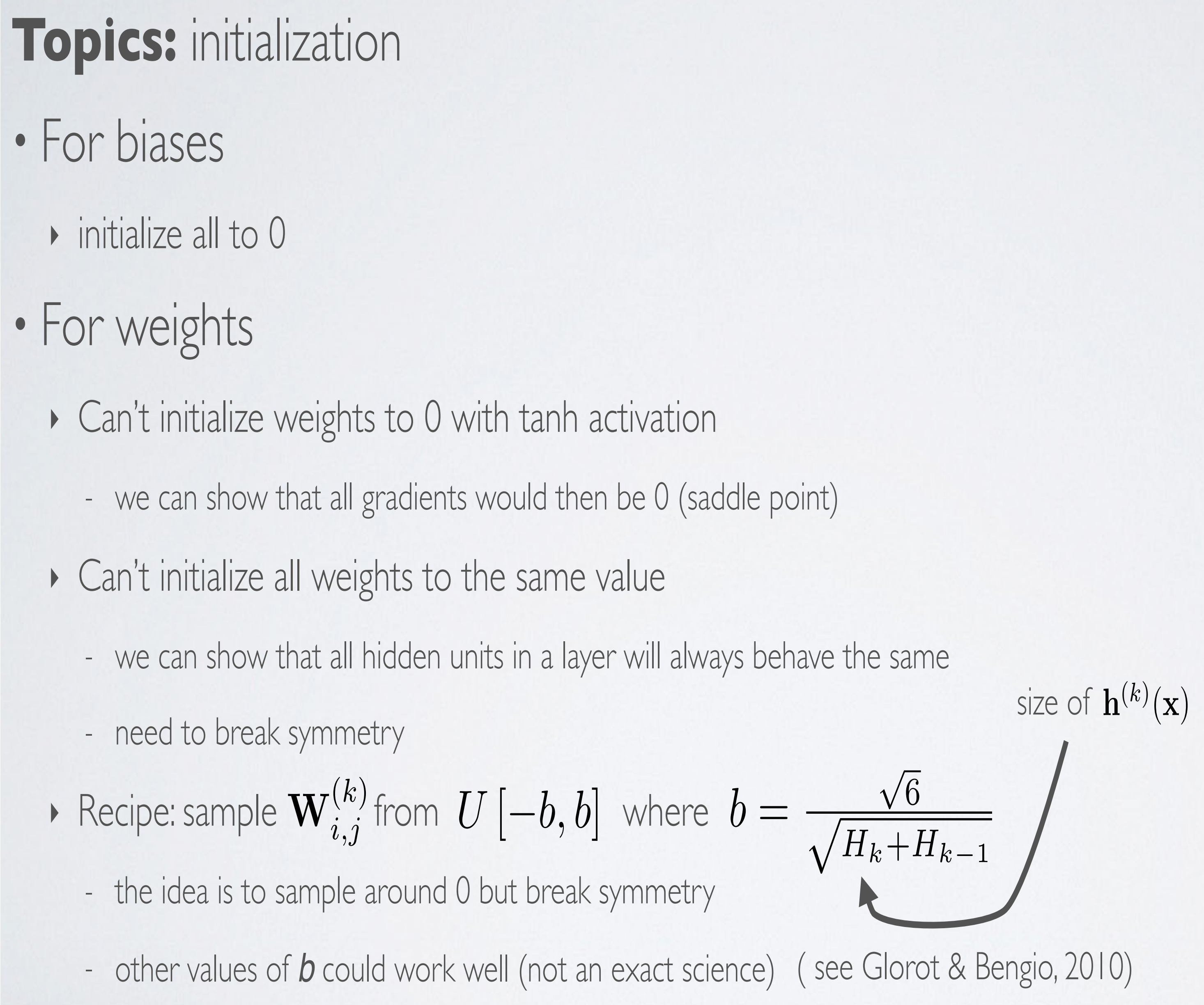
is trying to make all the weights in the group $W_{i,\cdot}$ go to 0

Weight Initialisation

(from

Hugo Larochelle)

Attempts to be invariant to the size of the layers



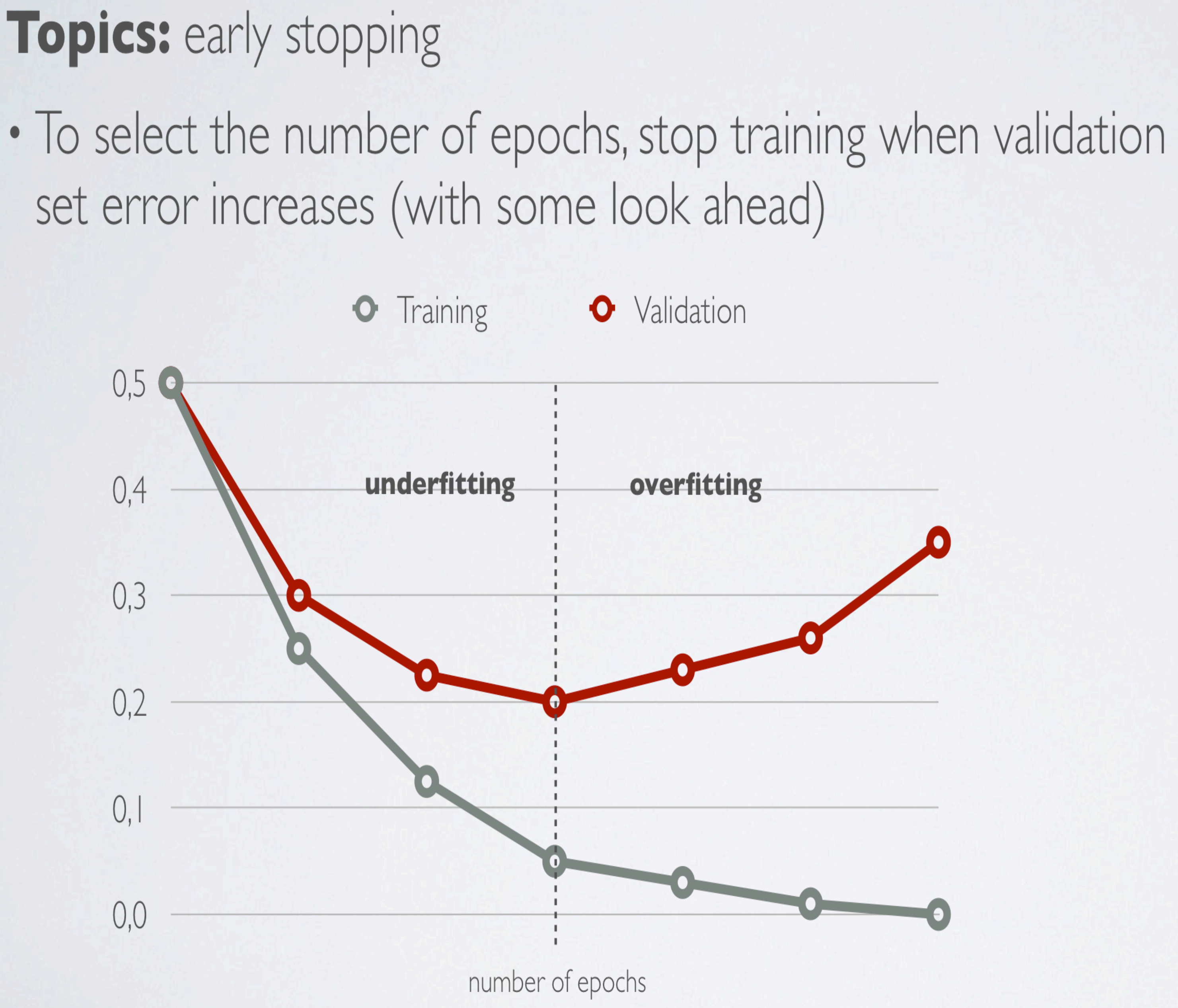
Topics: initialization

- For biases
  - initialize all to 0
- For weights
  - Can't initialize weights to 0 with tanh activation
    - we can show that all gradients would then be 0 (saddle point)
  - Can't initialize all weights to the same value
    - we can show that all hidden units in a layer will always behave the same
    - need to break symmetry
  - Recipe: sample $\mathbf{W}^{(k)}_{i,j}$ from $U\left[-b, b\right]$ where $b = \frac{\sqrt{6}}{\sqrt{H_k + H_{k-1}}}$
    - the idea is to sample around 0 but break symmetry
    - other values of *b* could work well (not an exact science) (see Glorot & Bengio, 2010)

size of $\mathbf{h}^{(k)}(\mathbf{x})$

Early Stopping : free lunch (+ jobs for the price of 1)

(from Hugo Larochelle)



Topics: early stopping

- To select the number of epochs, stop training when validation set error increases (with some look ahead)

Regularizing by injecting noise: dropout

(from

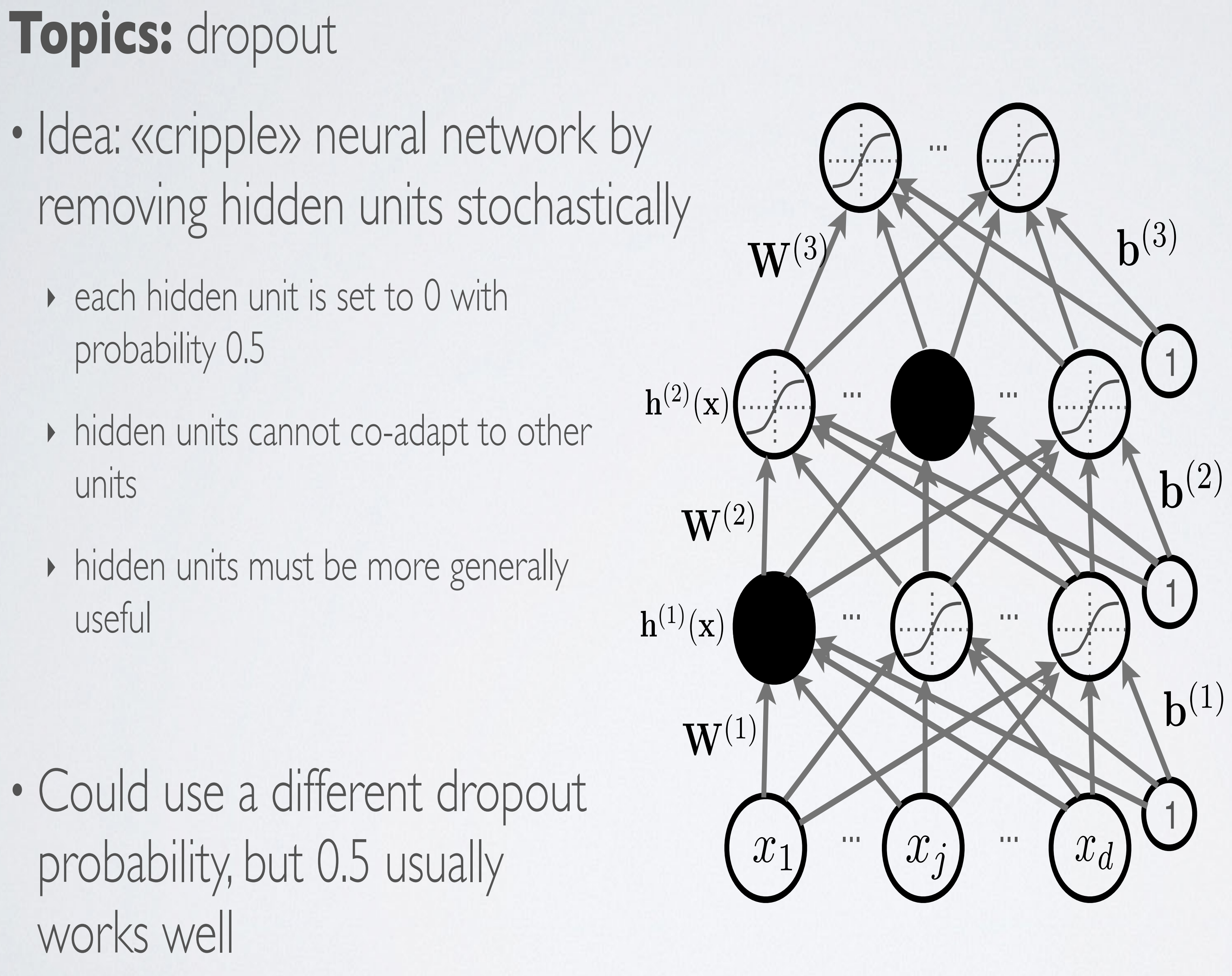
Hugo

Larochelle)

No noise

at test

time.

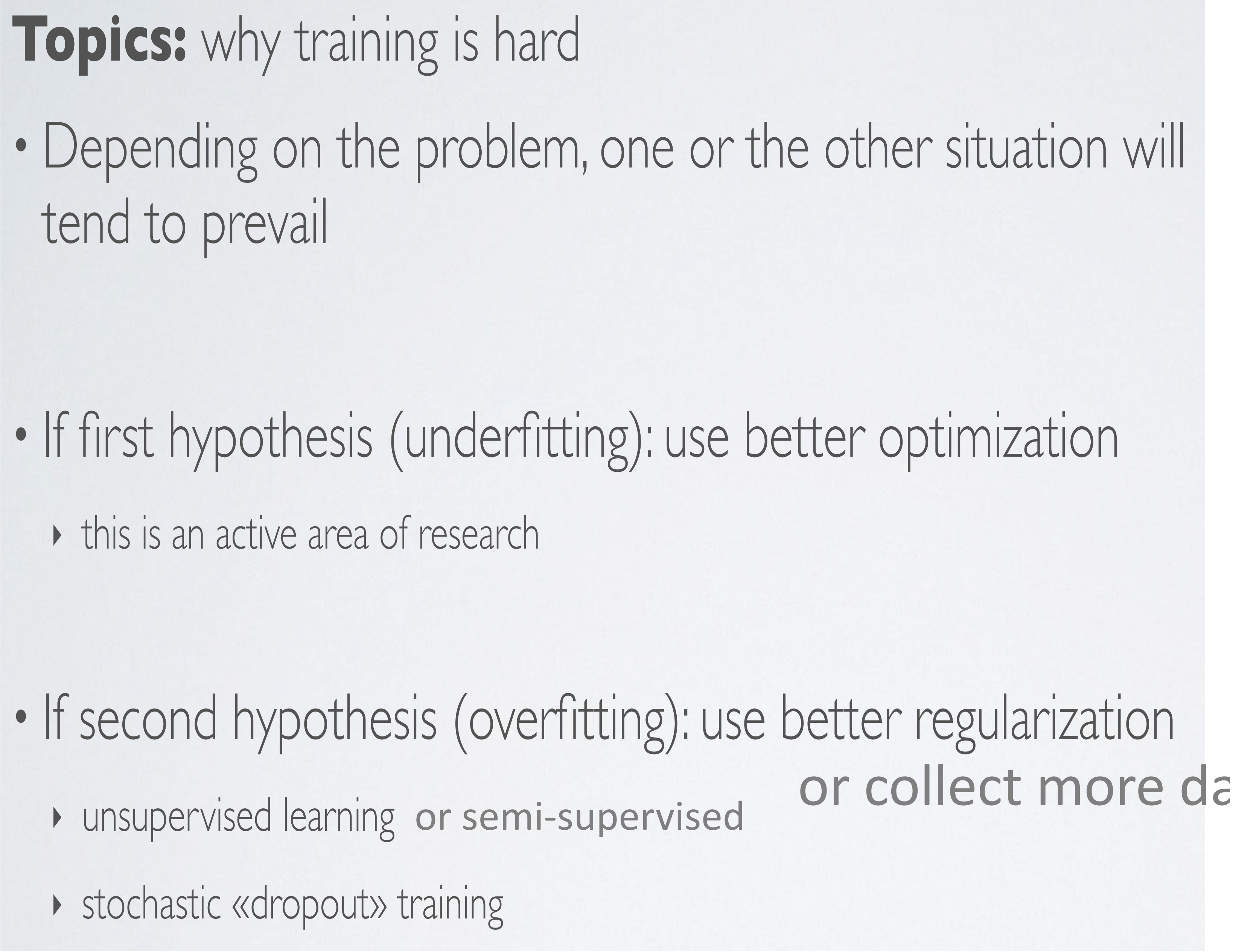


Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well

Diagnostic: overfitting vs underfitting?

(from Hugo Larochelle)

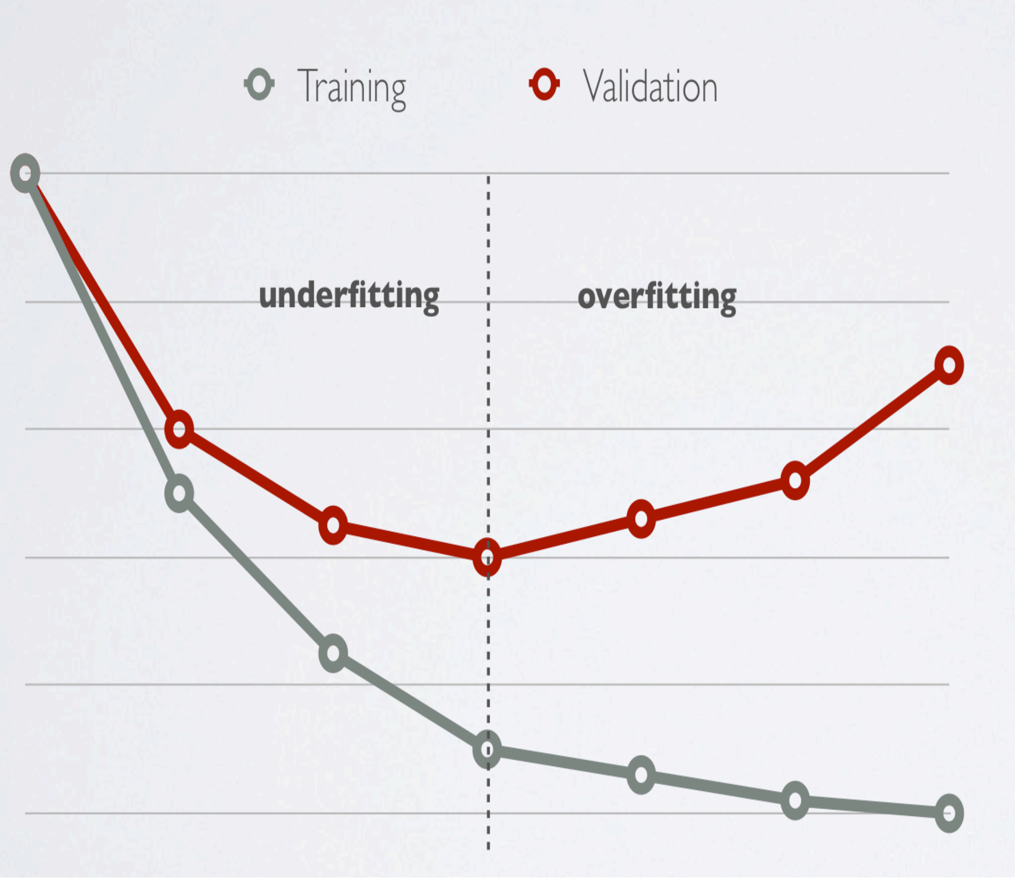


Topics: why training is hard

- Depending on the problem, one or the other situation will tend to prevail
- If first hypothesis (underfitting): use better optimization
  - this is an active area of research
- If second hypothesis (overfitting): use better regularization or collect more da
  - unsupervised learning or semi-supervised
  - stochastic «dropout» training

How to know if you are overfitting or underfitting?

Overfitting: if you increase capacity (number of parameters, training time, better optimizer, smaller regularization coefficient, etc.), test or validation error increase



Meta-Learning / Learning to Learn

- Generalize the idea of hyper-parameter optimization
  - Inner loop optimization (normal training), a fn of meta-params

$$\theta_t(\omega) = \mathrm{approxmin}_\theta C(\theta, \omega, \mathcal{D}^t_{train})$$

  - Outer loop optimization (meta-training), optimize meta-params

$$\omega = \mathrm{approxmin}_\omega \sum_t L(\theta_t(\omega), \omega, \mathcal{D}^t_{test})$$

- Meta-parameters can be the learning rule itself (Bengio & Bengio 1991; Schmidhuber 1992), learn 2 optimize
- Meta-learn an objective or reward function, or a shared encoder
- Meta-learning can be used to learn to generalize or transfer
- Can backprop through θ_t , use RL, evolution, or other tricks

Injecting Noise in a Nonsmooth Net

- Injecting noise corresponds to convolving the objective function with the noise kernel:

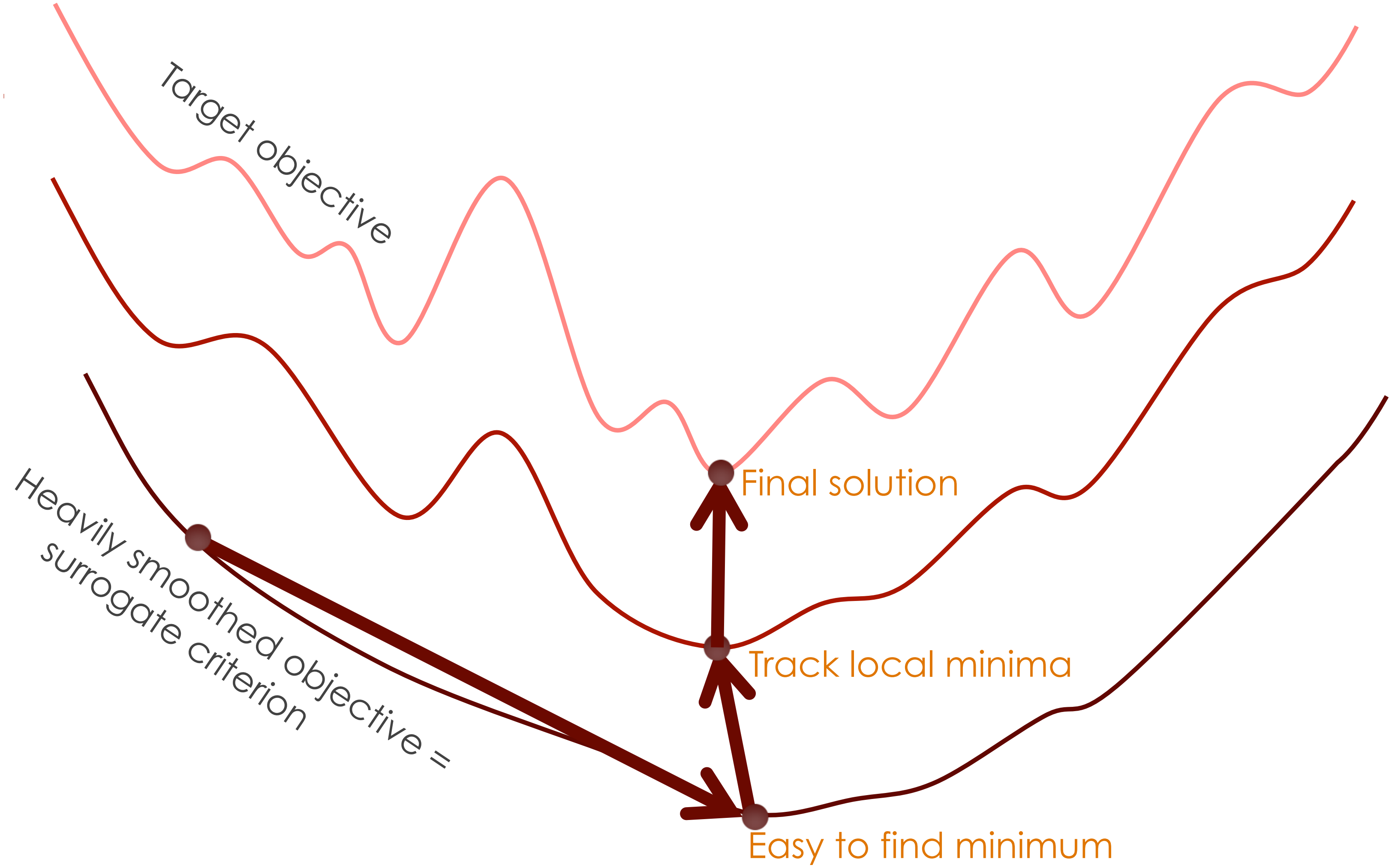
$$C(\theta) * \mathcal{N}(\epsilon) = \int_\epsilon C(\theta - \epsilon)\mathcal{N}(\epsilon) d\epsilon$$

$$\approx \text{mean}_{\epsilon \sim \mathcal{N}(\epsilon)} C(\theta - \epsilon)$$

- Same thing for the gradient, so we get a stochastic gradient on a smooth of the original objective function, which should be easier to optimize.
- Gradually reducing the noise level = simulated annealing

Continuation Methods and Simulated Annealing

- Gradually consider less easy versions of the objective of interest, tracking the local minima found along the way



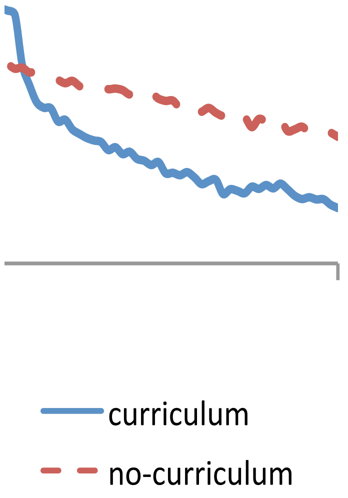
Order & Selection of Examples Matters

(Bengio, Louradour, Collobert & Weston, ICML'2009)

- Curriculum learning

  (Bengio et al 2009, Krueger & Dayan 2009)

  is a form of continuation method

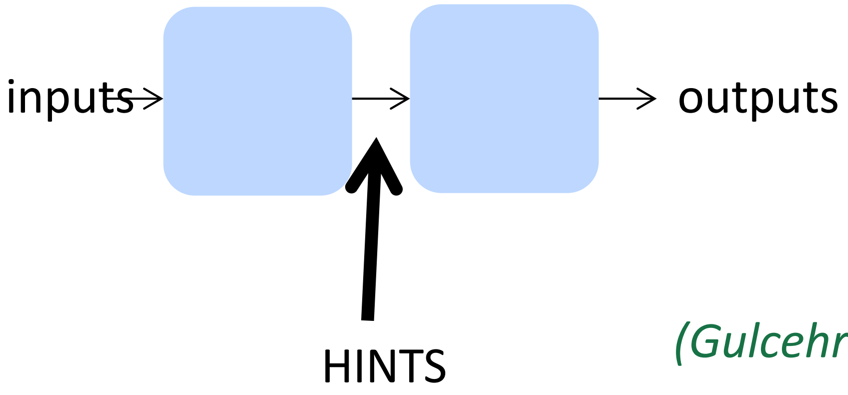


- **Start with easier examples**

- Faster convergence to a better solutions in deep architectures

Guided Training, Intermediate Concepts

- Breaking the problem in two sub-problems and pre-training each module separately, then fine-tuning, nails it
- *Need prior knowledge to decompose the task*
- Guided pre-training allows to find much better solutions, escape effective local minima



(Gulcehre & Bengio ICLR’2013)

Debugging

- Instrument the code to make experiments reproducible
- Use tools to verify gradients (finite differences)
- **Train on a small dataset**: verify can reach 0 training error
- Track error curves during training (training error, validation error); training error should roughly go down
- Track distribution statistics of weights and gradients during training

Validate and Analyze

- Vary capacity and observe error curves to identify if the system is rather overfitting or rather underfitting
- Compare with simpler reference models (logistic regression, SVMs, random forests)
- Track several relevant metrics
- Look at the training and validation examples which give the worse error (input, output and target)
- Measure effect of changing the number of training examples
- Make sure you have enough test examples to be able to conclude with statistical significance

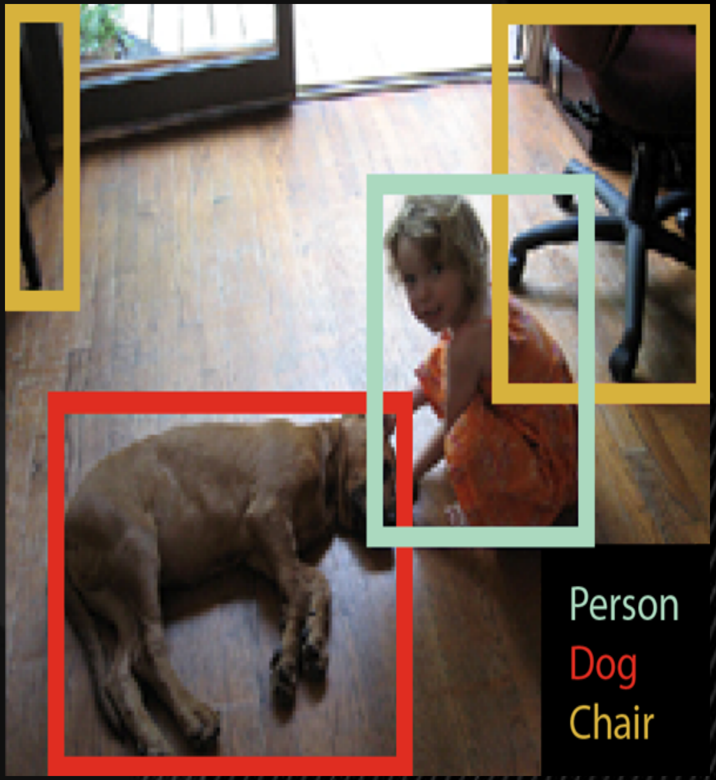
Convolutional Nets

Anything New with Deep Learning since the Neural Nets of the 90s?

- Rectified linear units instead of sigmoids, enable training much deeper networks by backprop (Glorot & Bengio AISTATS 2011)
- Some forms of noise (like dropout) are powerful regularizers yielding superior generalization abilities
- Success of deep **convnets** trained on large labeled image datasets, success of skip connections (ResNets)
- Success of recurrent nets with more memory, with gating units
- Success of word embedding, neural machine translation, deep NLP
- Attention mechanisms liberate neural nets from fixed-size inputs, self-attention allows to work on sets, graphs
- Autoencoders, adversarial training, generating images & sounds
- Transfer learning, meta-learning, deep reinforcement learning

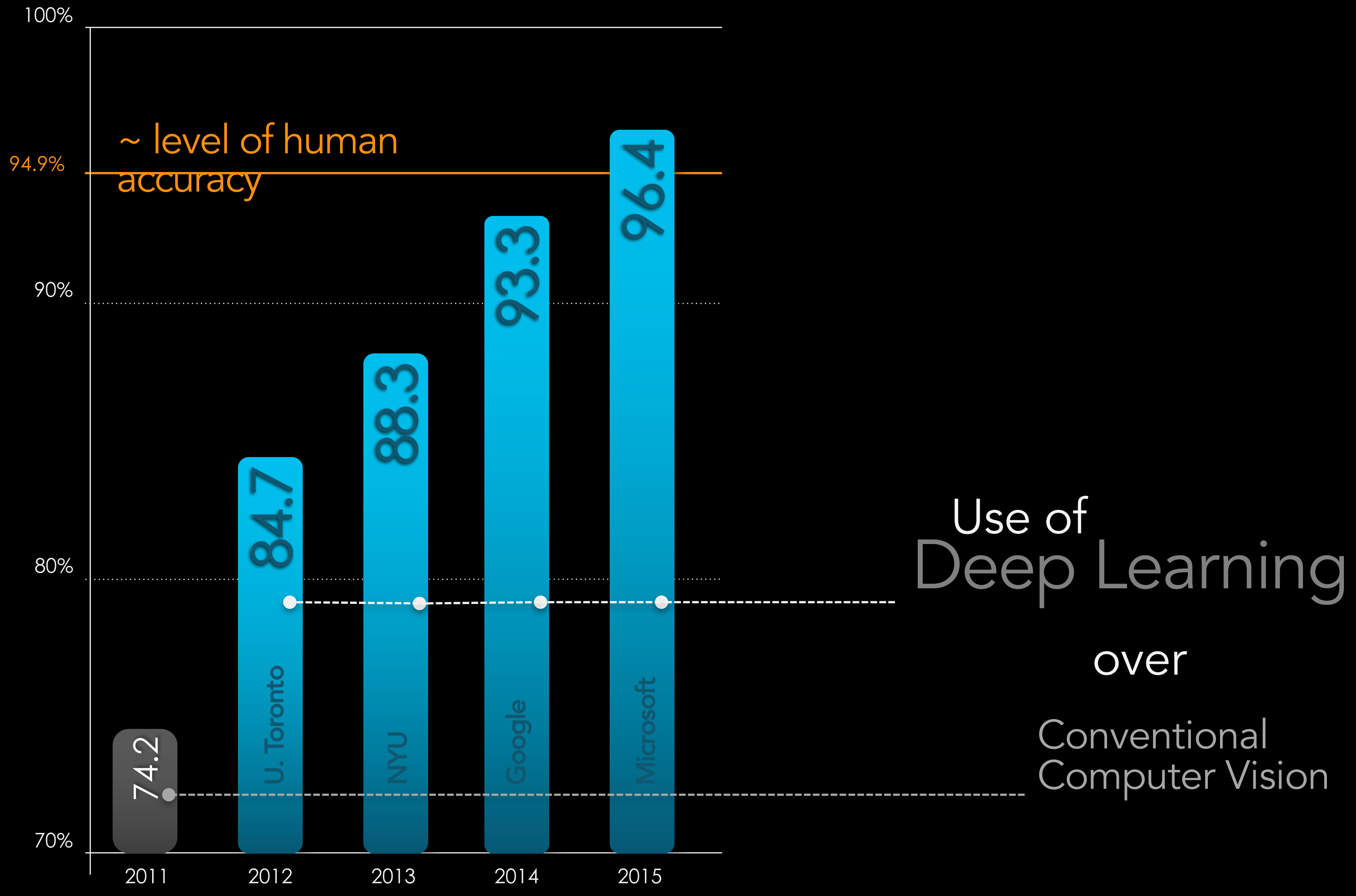
2012-2015: breakthrough in computer vision

- Graphics Processing Units (GPUs) + 10x more data
- 1,000 object categories
- Facebook: millions of faces



Approaching Human Accuracy

Top-5 Classification task, ImageNet



Use of Deep Learning over Conventional Computer Vision

Convolutional Networks

- Scale up neural networks to process very large images / video sequences
    - Sparse connections
    - Parameter sharing
- Automatically generalize across spatial translations of inputs
- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, …)

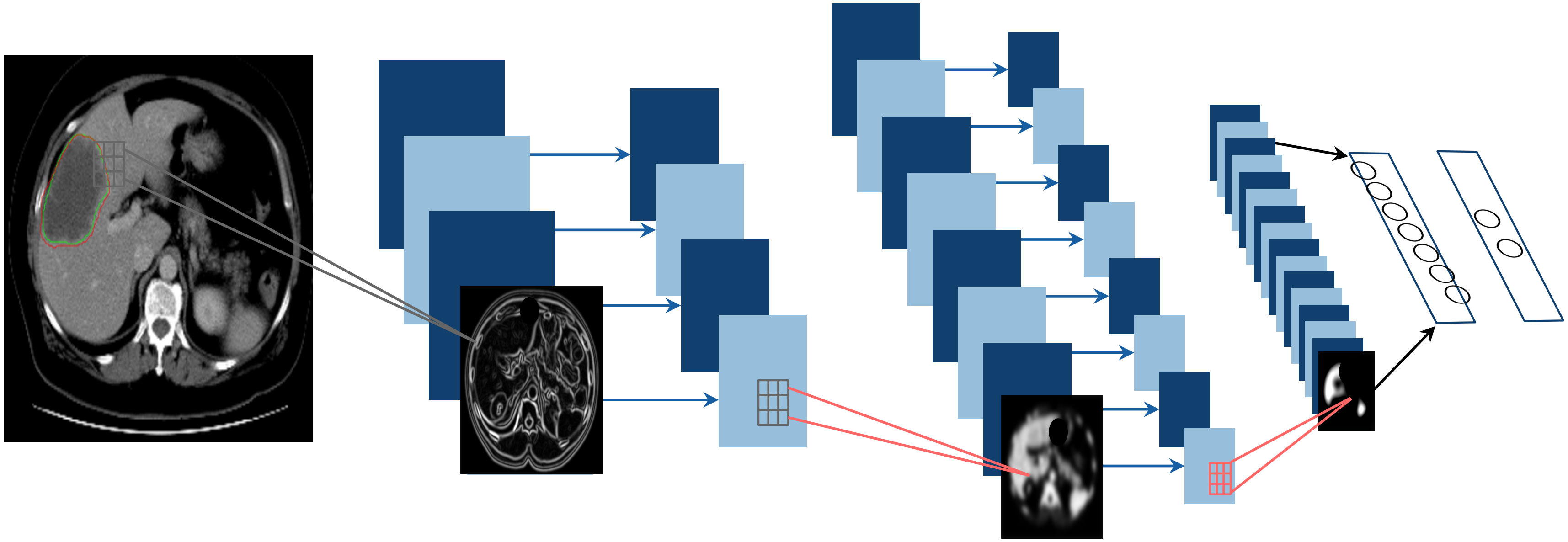
Convnets: Key Idea

- Replace matrix multiplication in ordinary neural nets with convolution
- Everything else stays the same
  - Maximum likelihood
  - Back-propagation
  - etc.

Convolutional Neural Networks

- A special kind of deep learning tailored for images
- Exploits the invariance to translations
- Exploits multi-scale hierarchy

Convolutional neural network for imaging data



2D Convolution

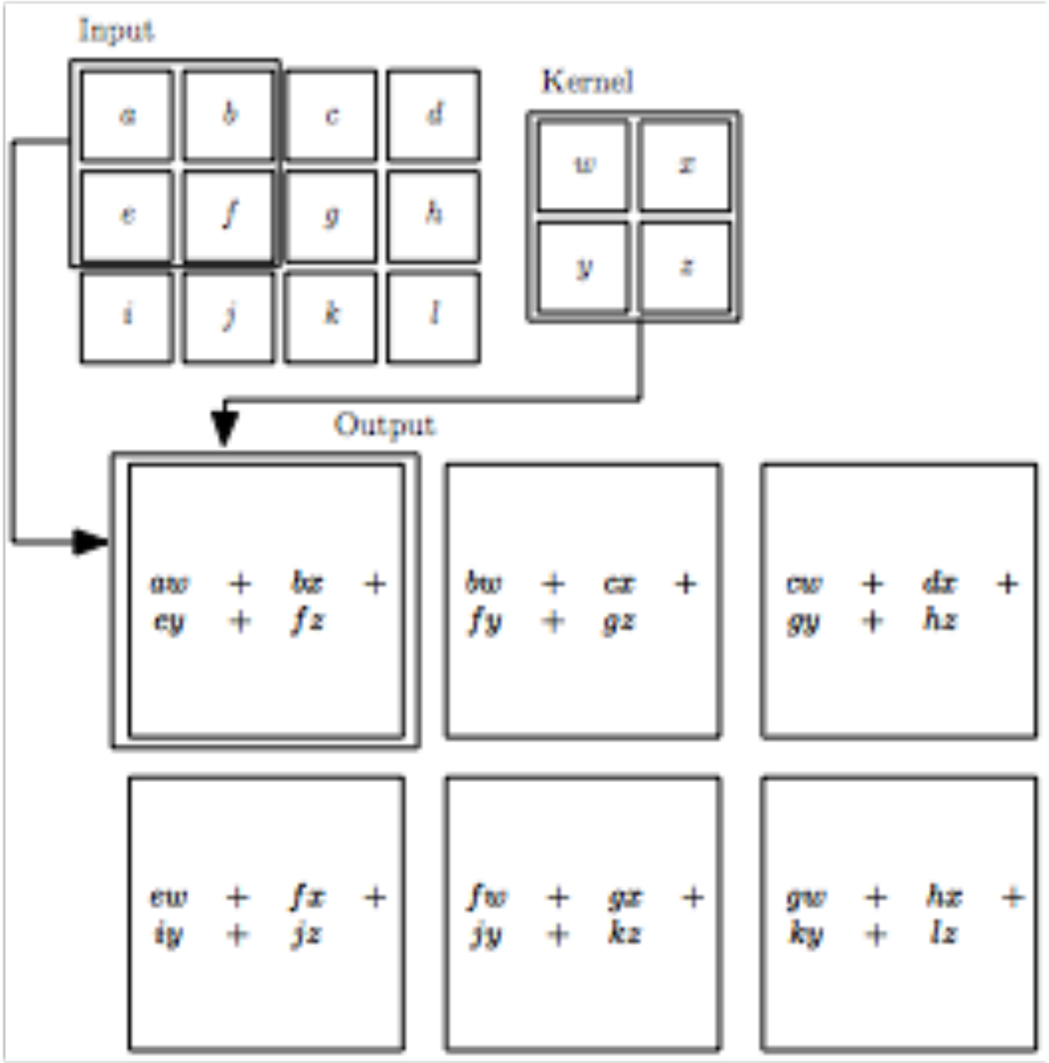


Figure 9.1, Deep Learning book, Goodfellow et al 2016

Sparse Connectivity

Sparse connections due to small convolution kernel

Dense connections

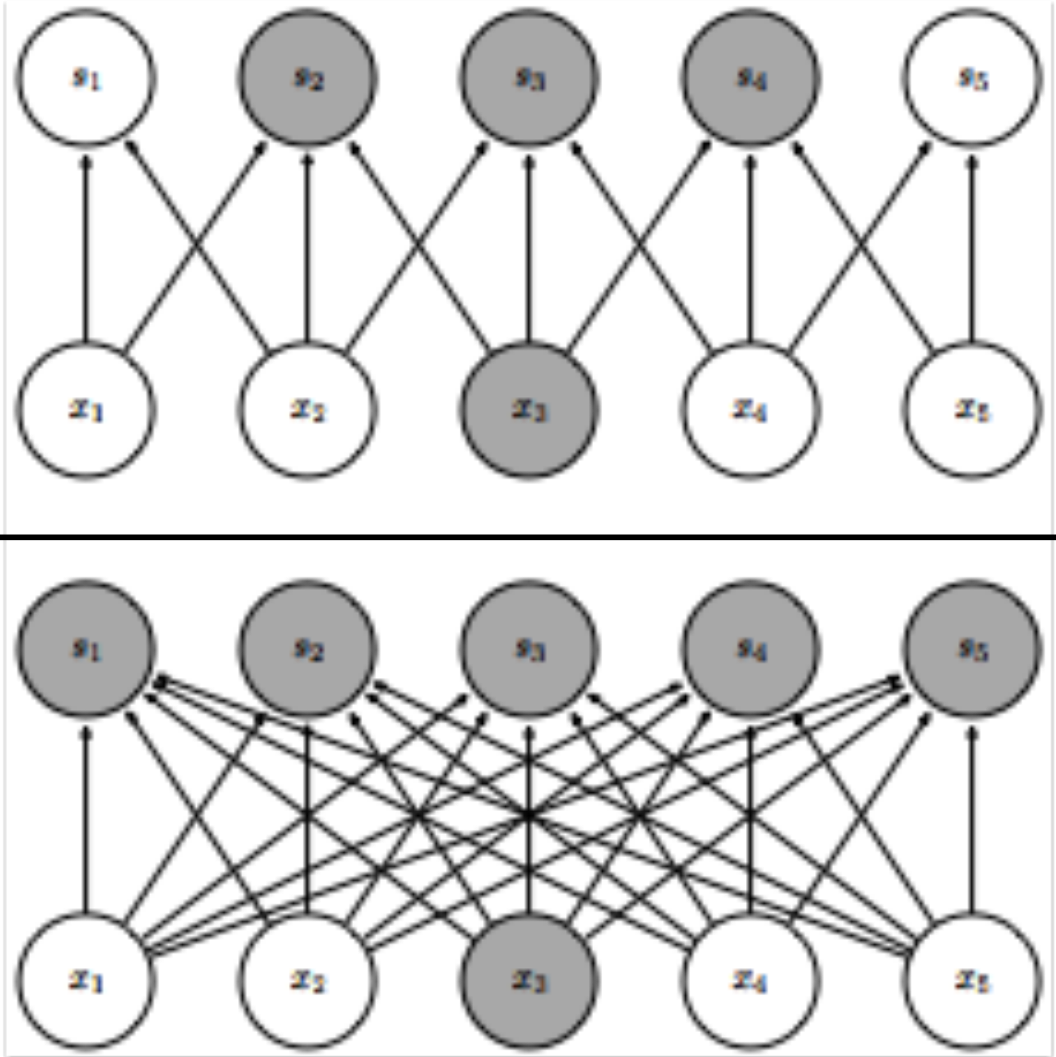


Figure 9.2

Sparse Connectivity

Sparse connections due to small convolution kernel

Dense connections

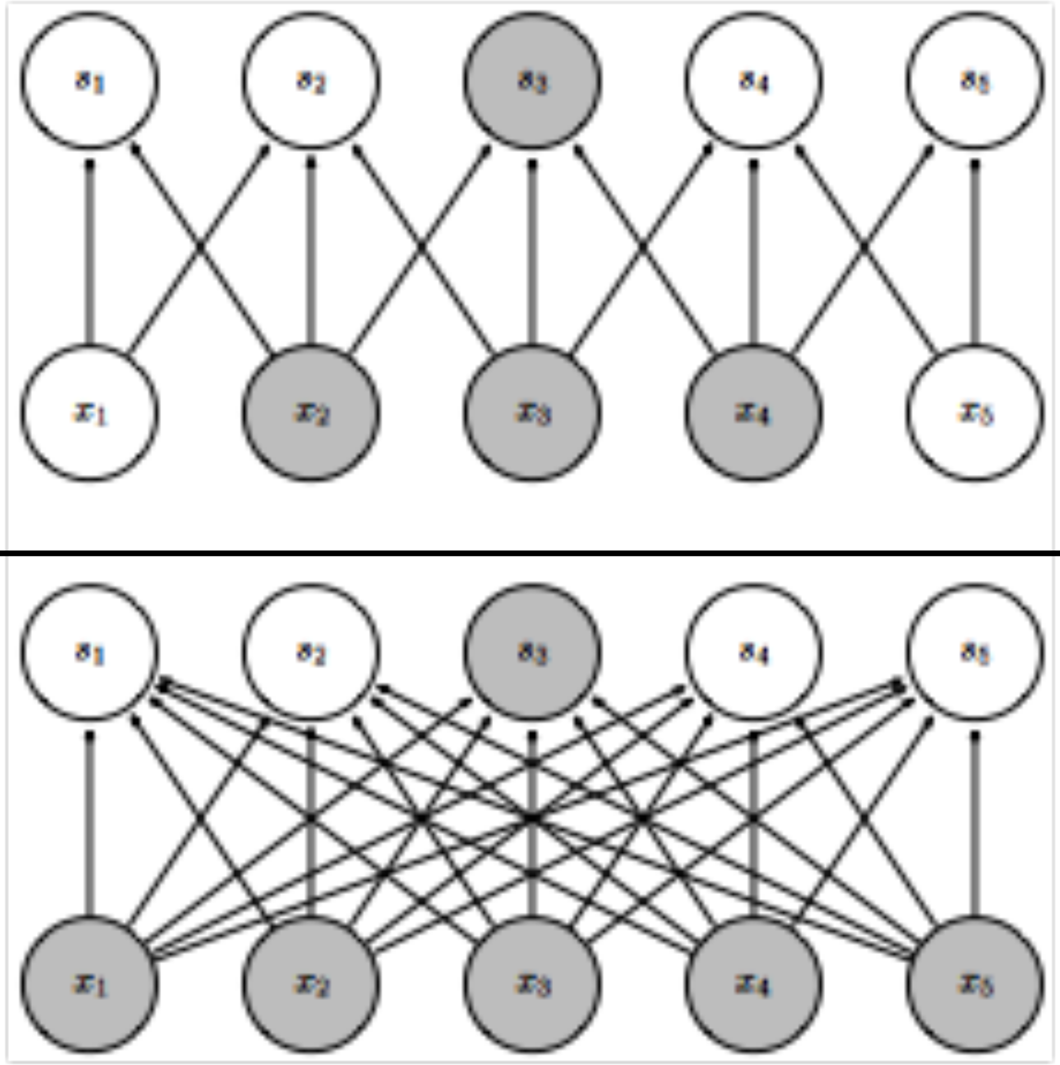


Figure 9.3

Growing Receptive Fields

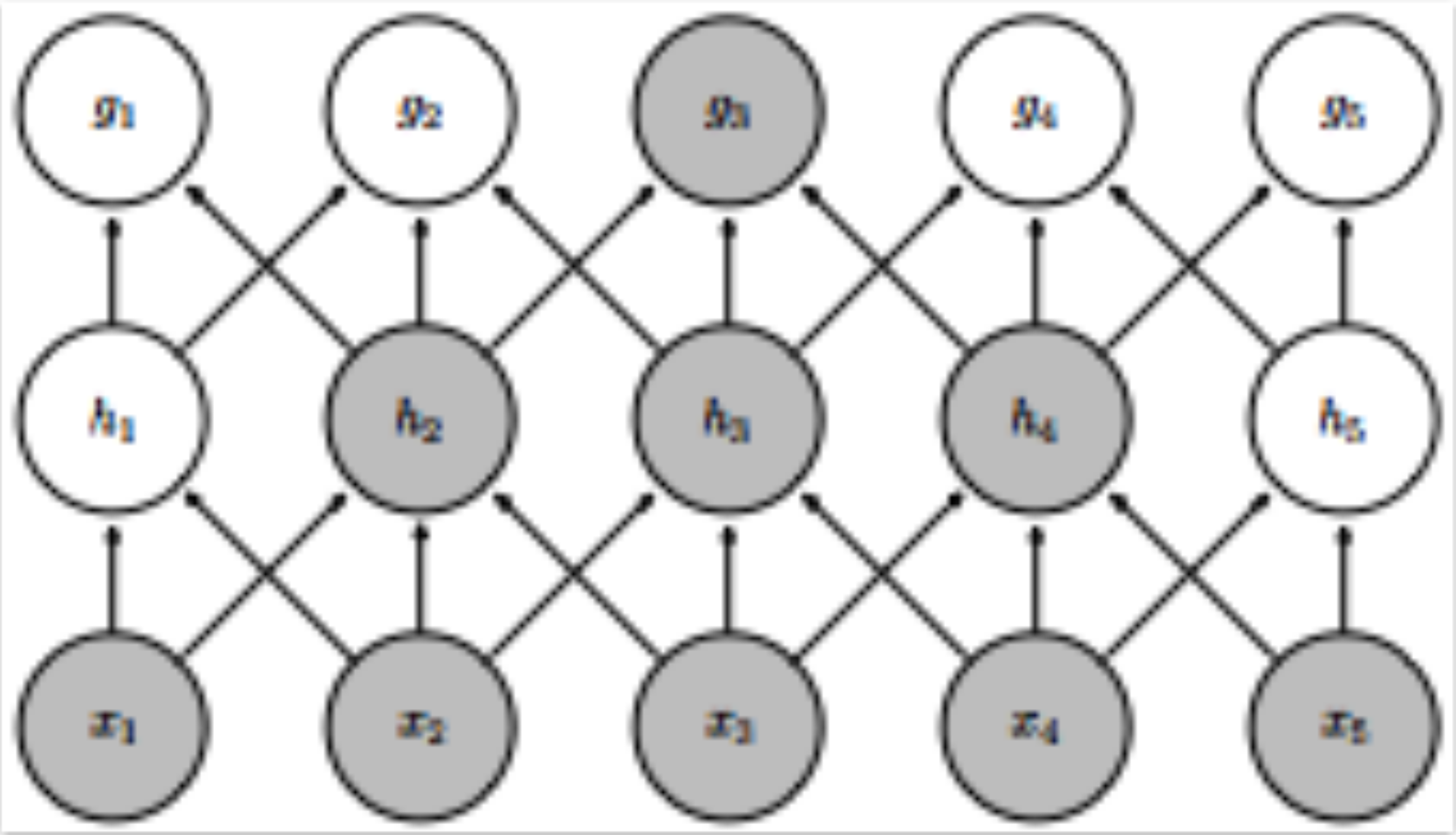
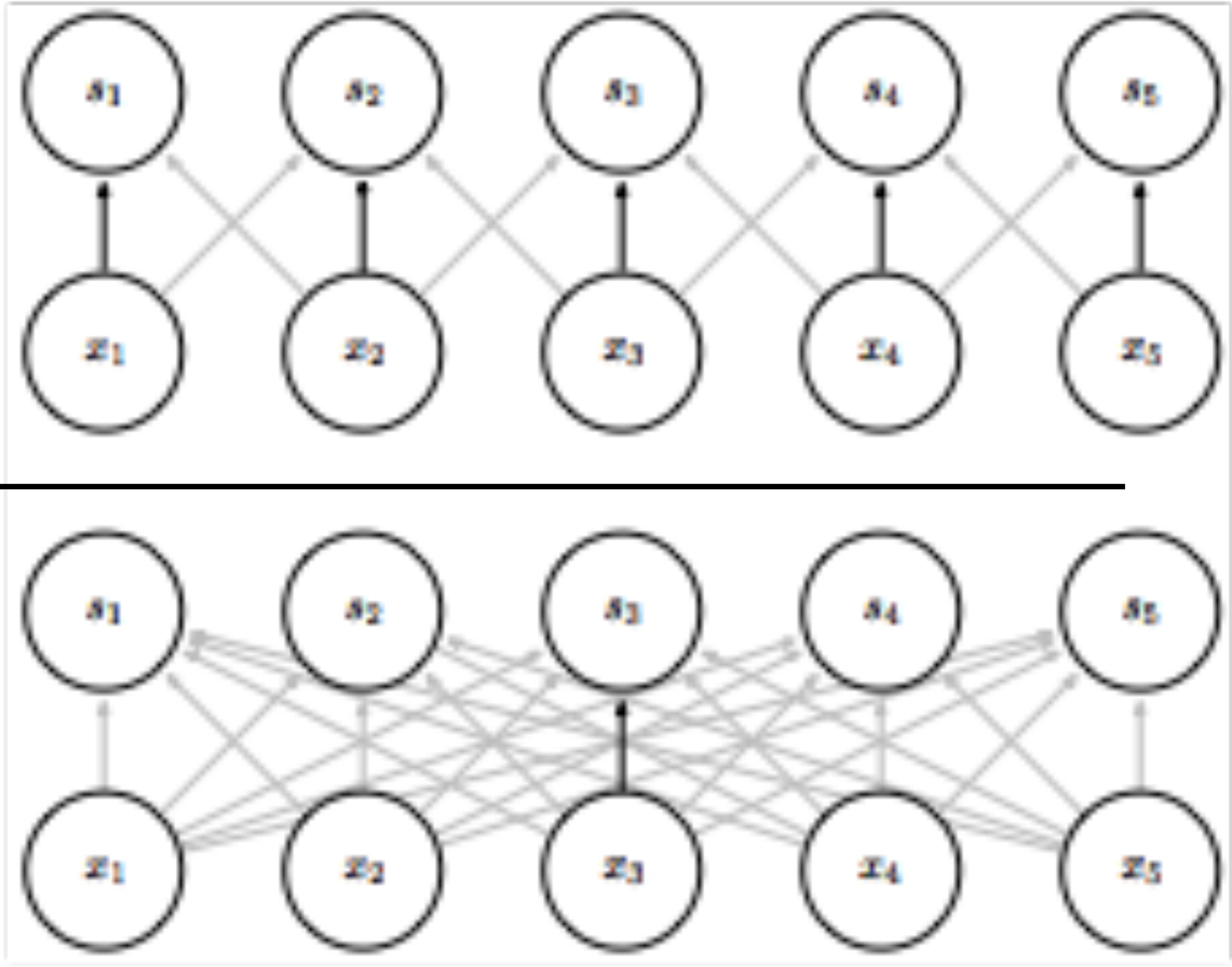


Figure 9.4

Parameter Sharing



Convolution shares the same parameters across all spatial locations

Traditional matrix multiplication does not share any parameters

Figure 9.5

Cross-Channel Pooling and Invariance to Learned Transformations

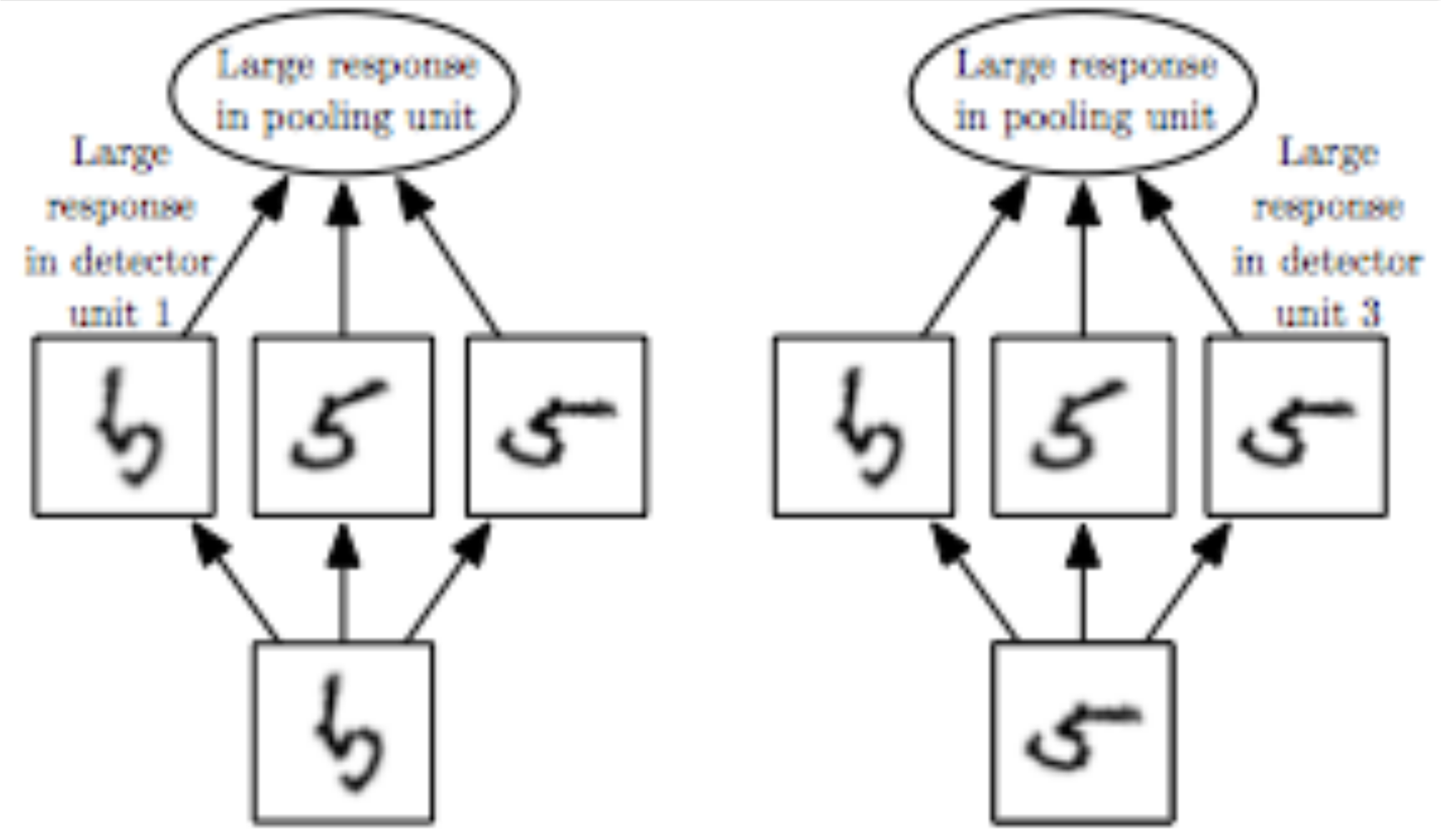

Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

Figure 9.9

Pooling with Downsampling

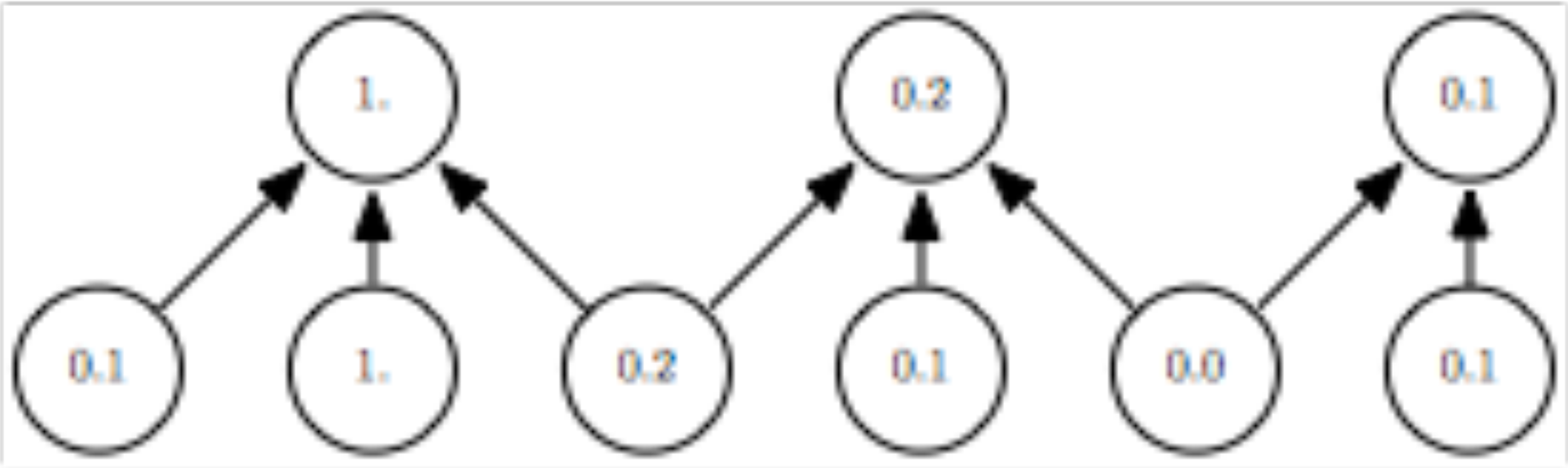


Figure 9.10

Convolution with Stride

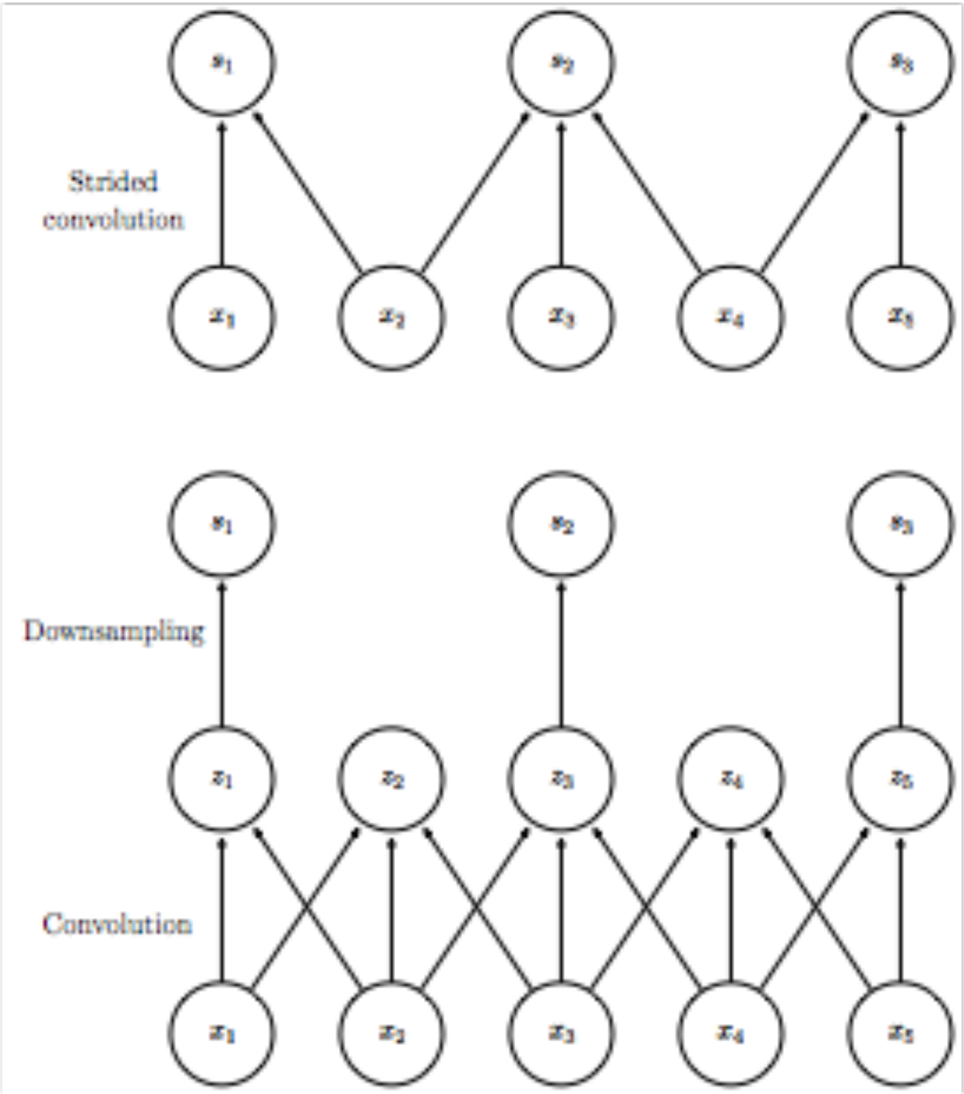


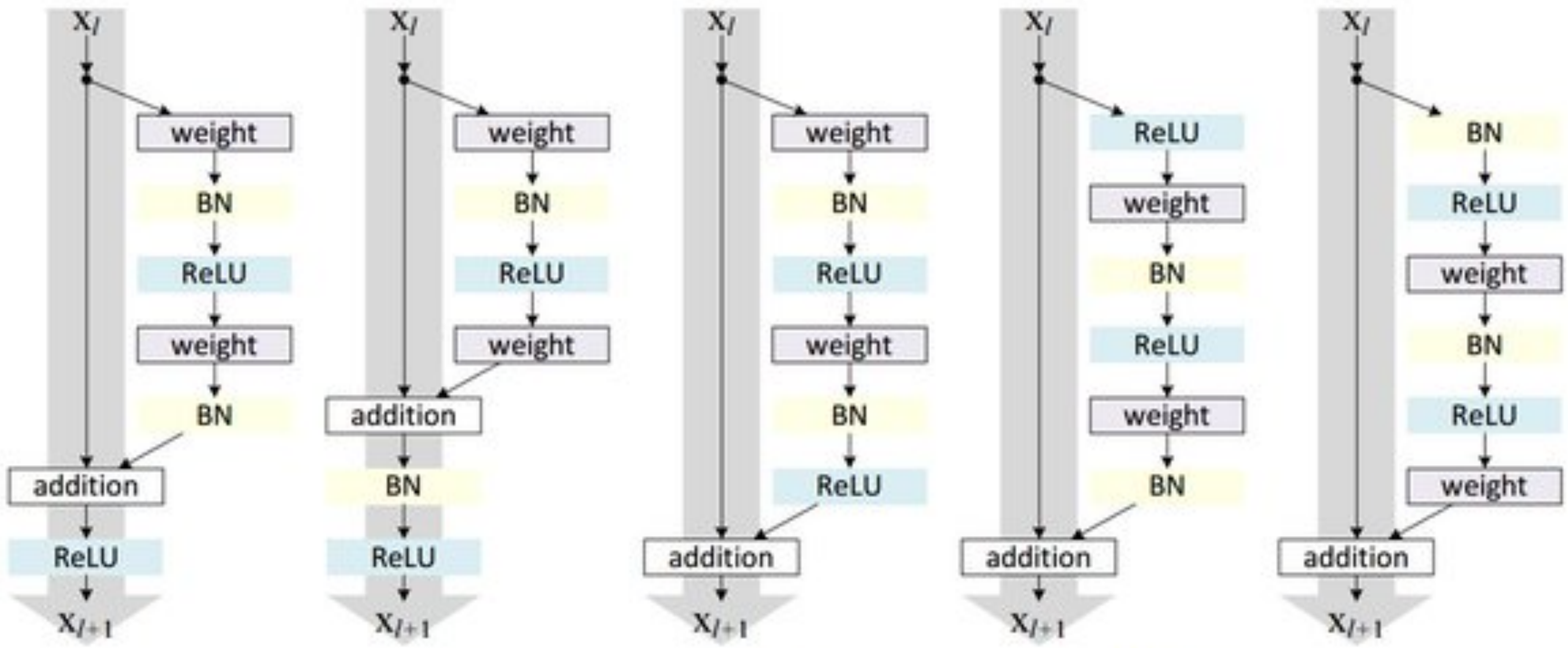
Figure 9.12

Major ConvNet Architectures

- Spatial Transducer Net: input size scales with output size, all layers are convolutional
- All Convolutional Net: no pooling layers, just use strided convolution to shrink representation size
- Inception: complicated architecture designed to achieve high accuracy with low computational cost
- ResNet: blocks of layers with same spatial size, with each layer’s output added to the same buffer that is repeatedly updated. Very many updates = very deep net, but without vanishing gradient.

ResNets: Skip Connections

- Identity paths make it possible for gradients to flow through deeper networks (He et al 2015), SOTA on object recognition



(a) original

(b) BN after addition

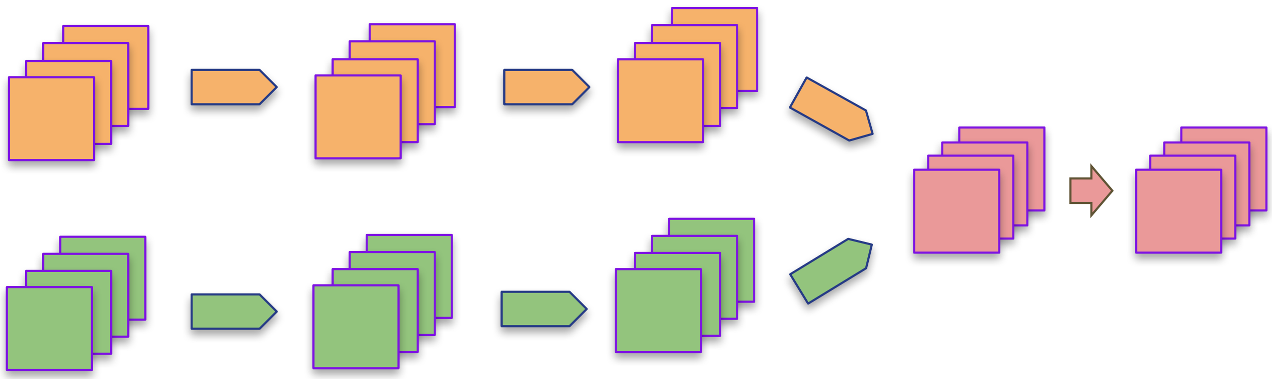
(c) ReLU before addition

(d) ReLU-only pre-activation

(e) **full pre-activation**

Deep Data Fusion

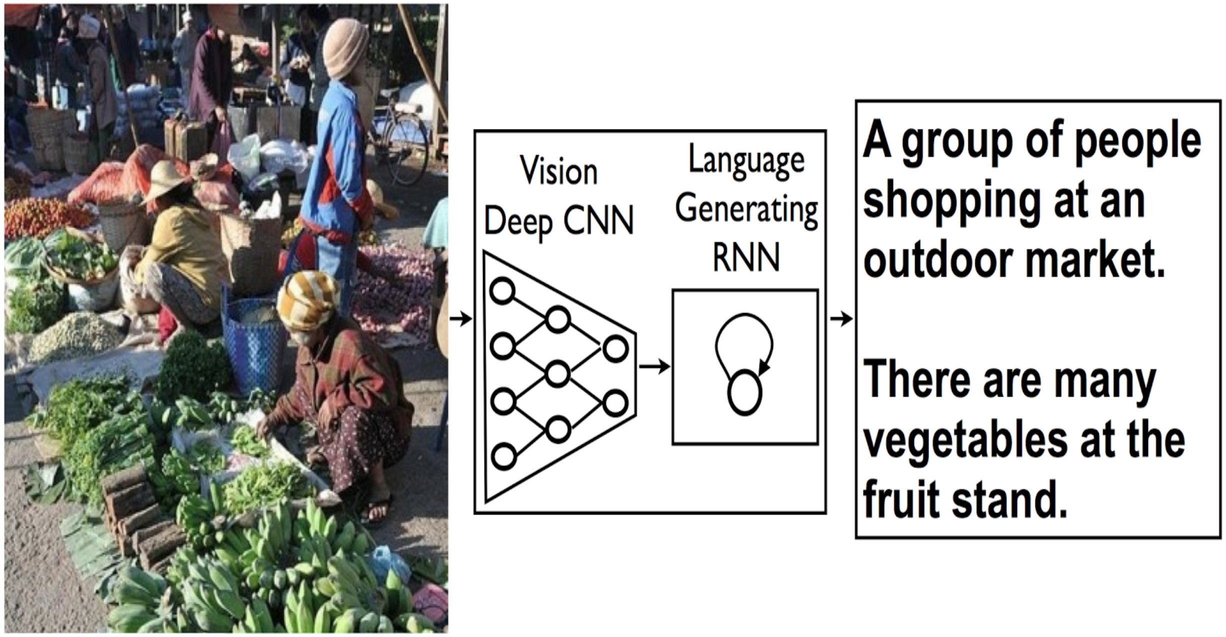
- Deep nets are very good at combining multiple sources of data, multiple sensors or modalities
- Can have separate pre-processing stages for each modality, then CONCATENATE the representations before continuing processing



Need to map to the same spatial scale, or ‘copy’ a non-spatial modality at all positions.

Generating Text from Images

- (Kiros *et al.*, 2014; Mao *et al.*, 2014; Donahue *et al.*, 2014; **Vinyals *et al.*, 2014**; Fang *et al.*, 2014; Chen and Zitnick, 2014; Karpathy and Li, 2014; Venugopalan *et al.*, 2014).
- Convolutional net → generative RNN



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

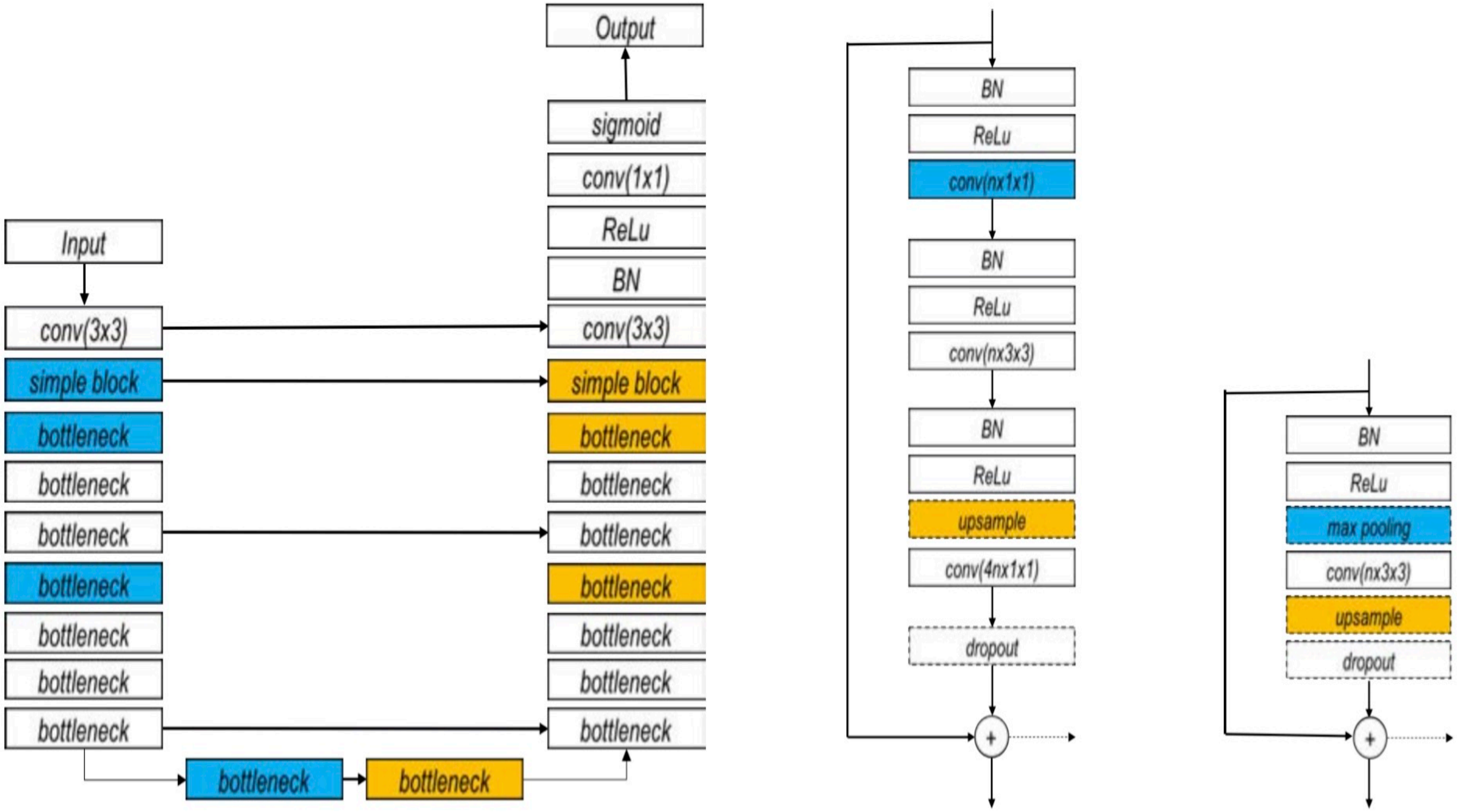
A close up of a child holding a stuffed animal

(GT: A young girl asleep on the sofa cuddling a stuffed bear.)

Two pizzas sitting on top of a stove top oven.

(GT: Three different types of pizza on top of a stove.)

U-Net Architecture for CNNs with Pixel-Wise Outputs



(a) FC-ResNet

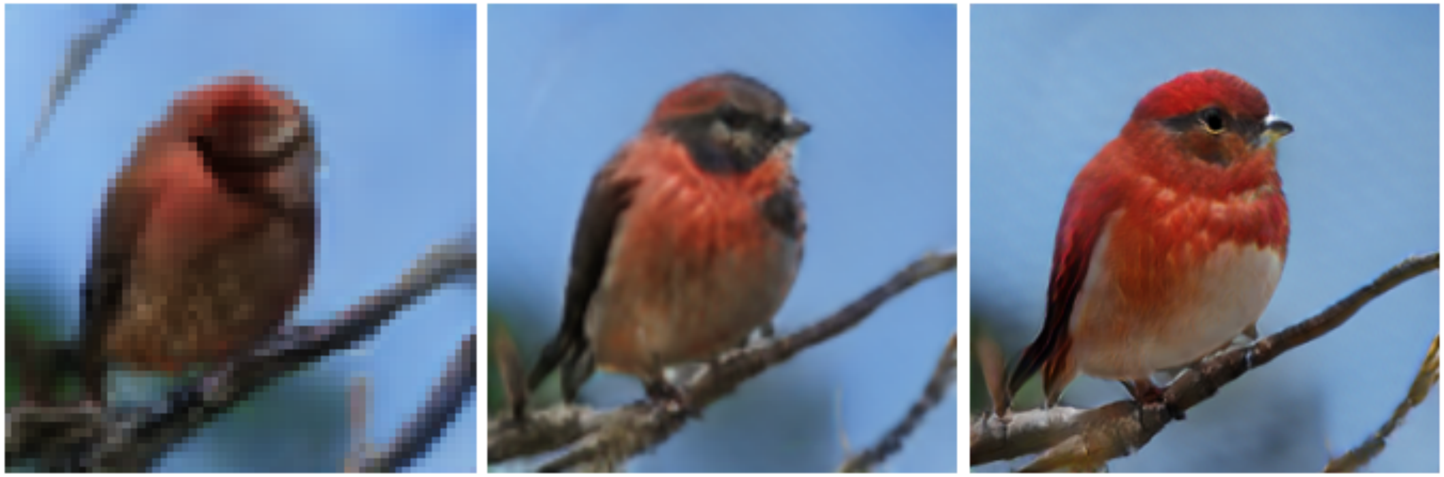
(b) Bottleneck block

(c) Simple block

Generating Images from Text

- With U-Net like architectures and multi-stage refinement
- With GAN types of objectives
- With attention mechanism

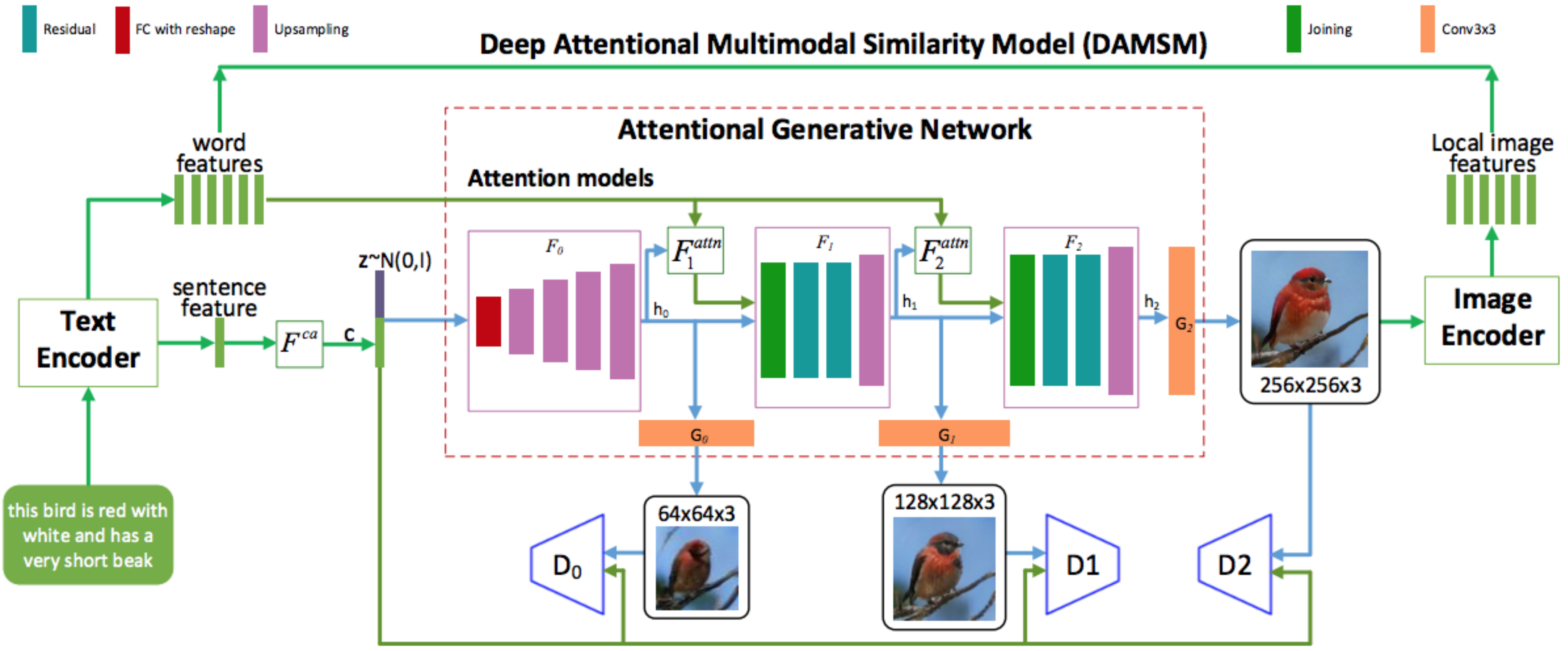
this bird is red with white and has a very short beak



Xu et al 2018, AttnGAN

AttnGAN Architecture

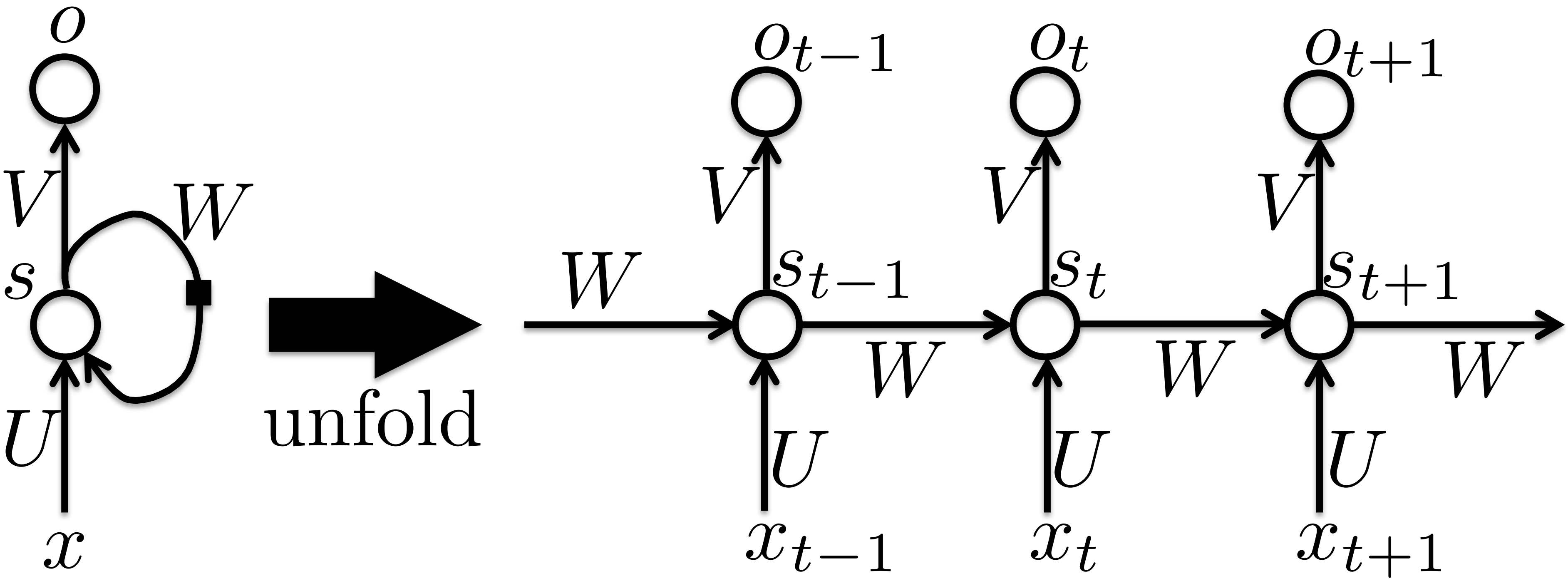
Many bells and whistles in modern deep learning…



Recurrent Neural Networks

Recurrent Neural Networks

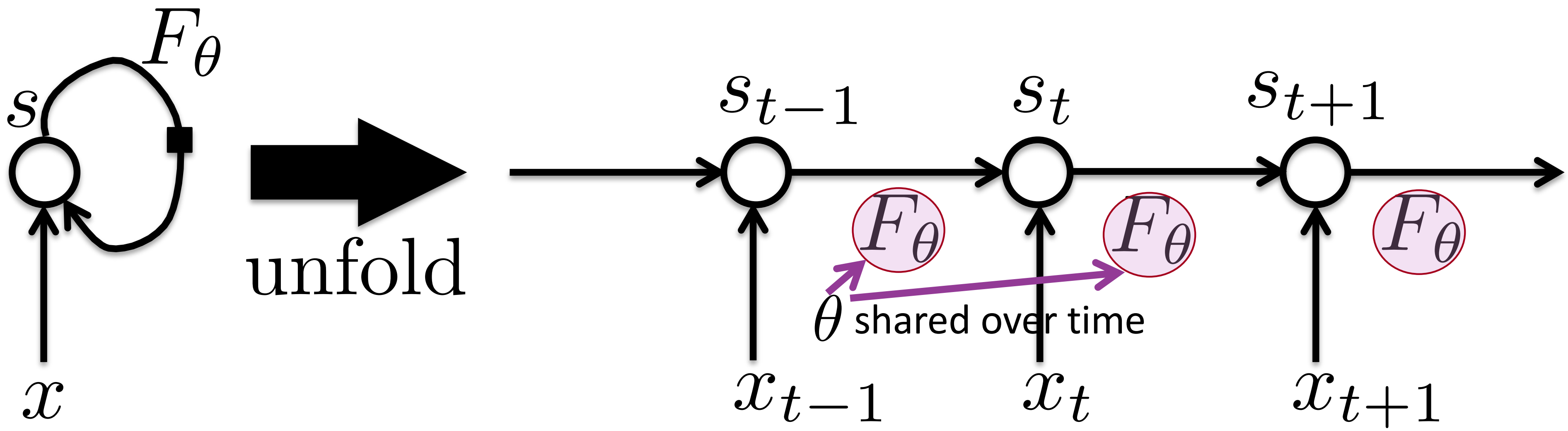
- Can produce an output at each time step: unfolding the graph tells us how to back-prop through time.



Recurrent Neural Networks

- Selectively summarize an input sequence in a fixed-size state vector via a recursive update

$$s_t = F_\theta(s_{t-1}, x_t)$$



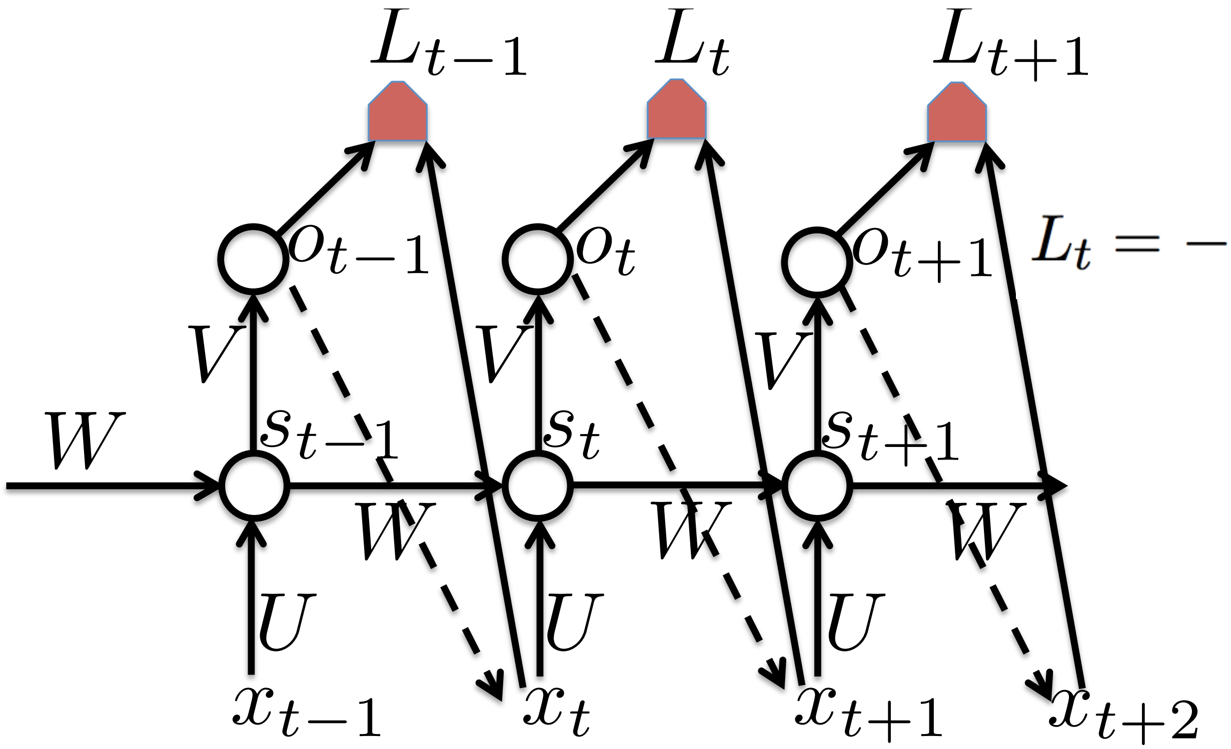
$$s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$$

➔ Generalizes naturally to new lengths not seen during training

Generative RNNs

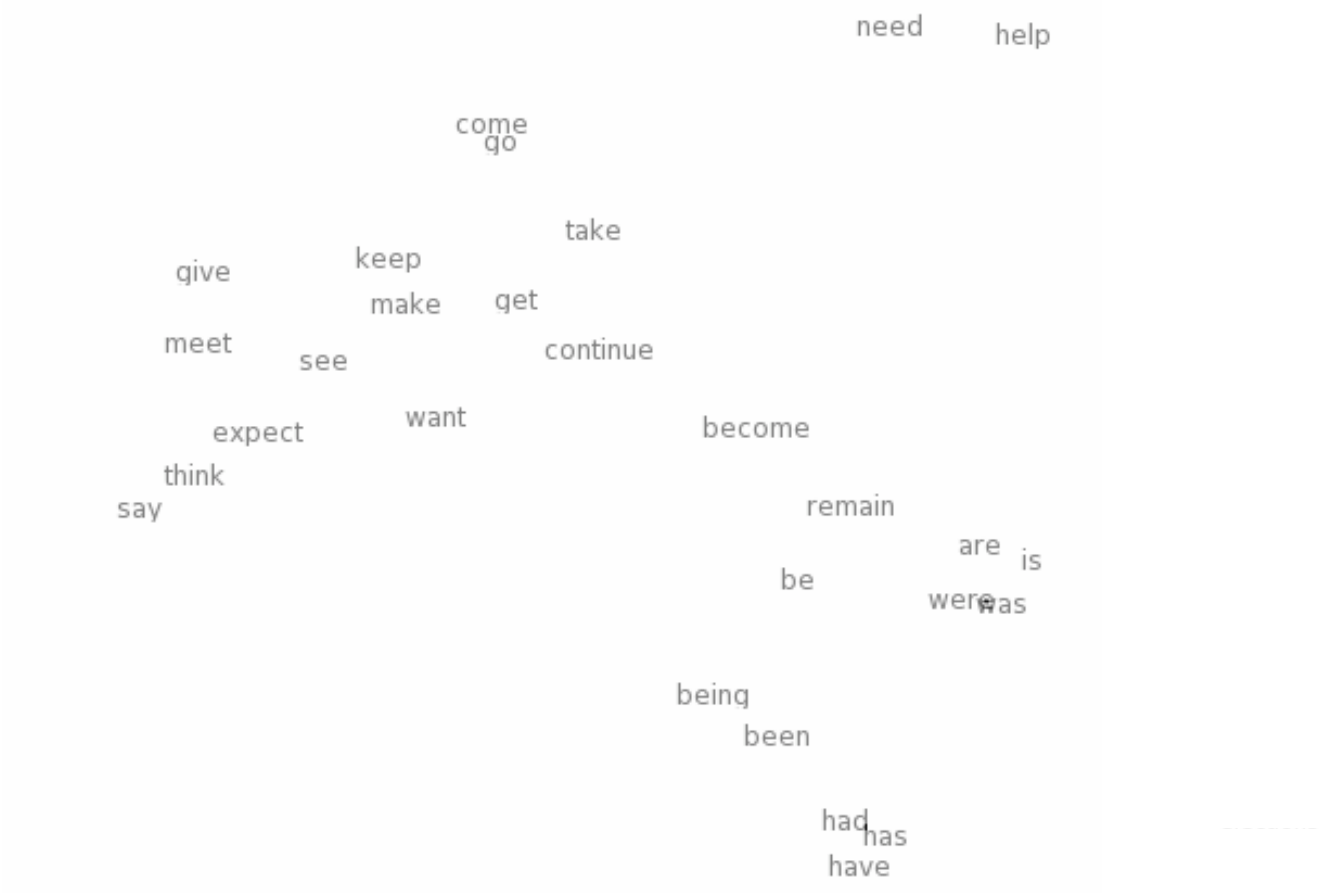
- An RNN can represent a fully-connected **directed generative model**: every variable predicted from all previous ones.

$$P(\mathbf{x}) = P(x_1, \ldots x_T) = \prod_{t=1}^{T} P(x_t | x_{t-1}, x_{t-2}, \ldots x_1)$$



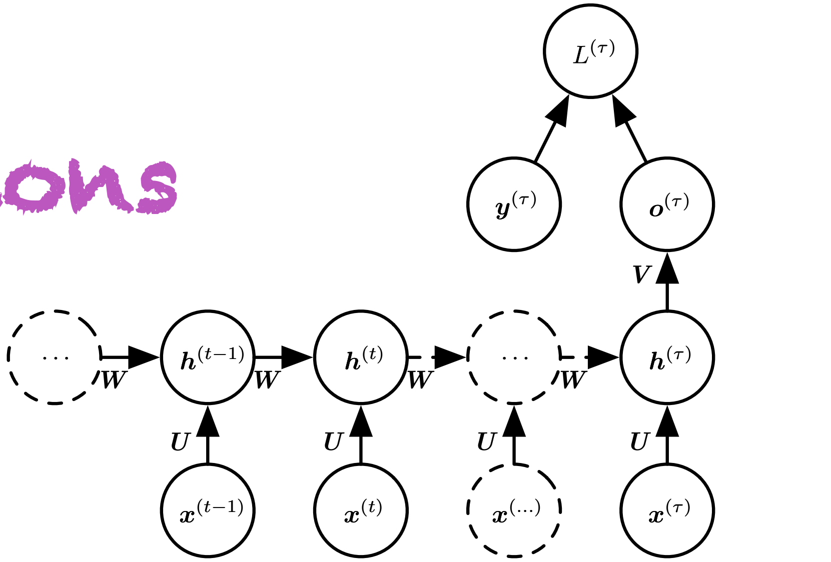
$$L_t = -\log P(x_t | x_{t-1}, x_{t-2}, \ldots x_1)$$

Neural word embeddings - visualization

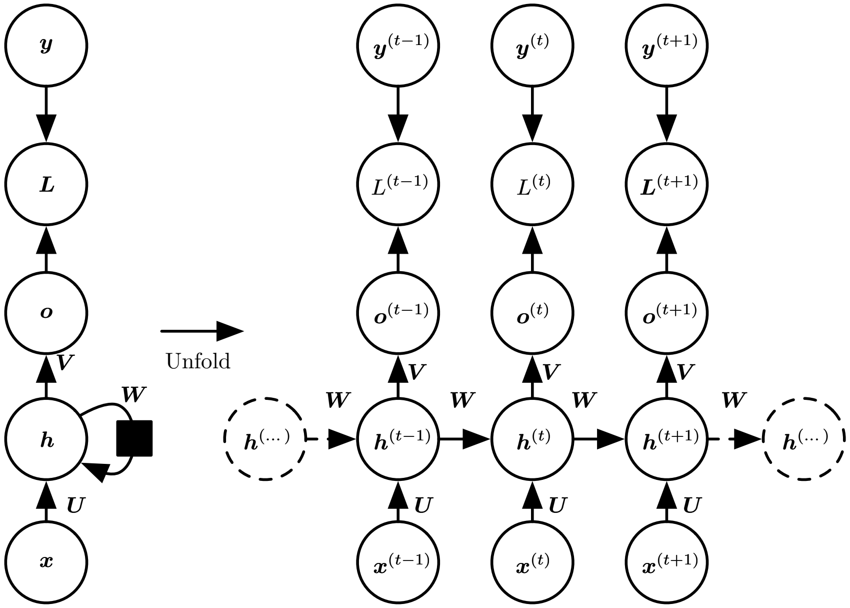


Conditional Distributions

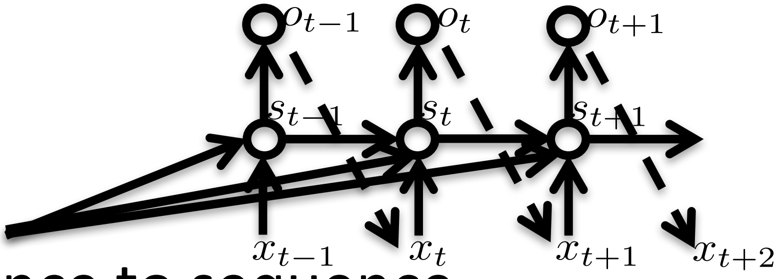
- Sequence to vector



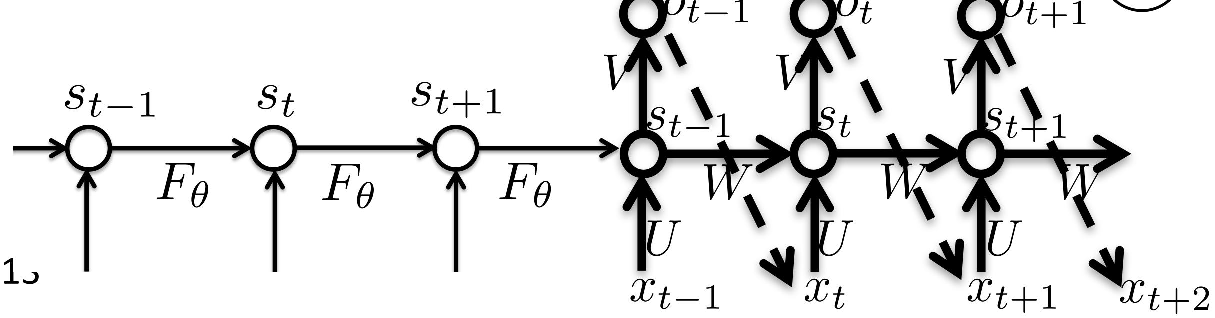
- Sequence to sequence of the same length, aligned



- Vector to sequence

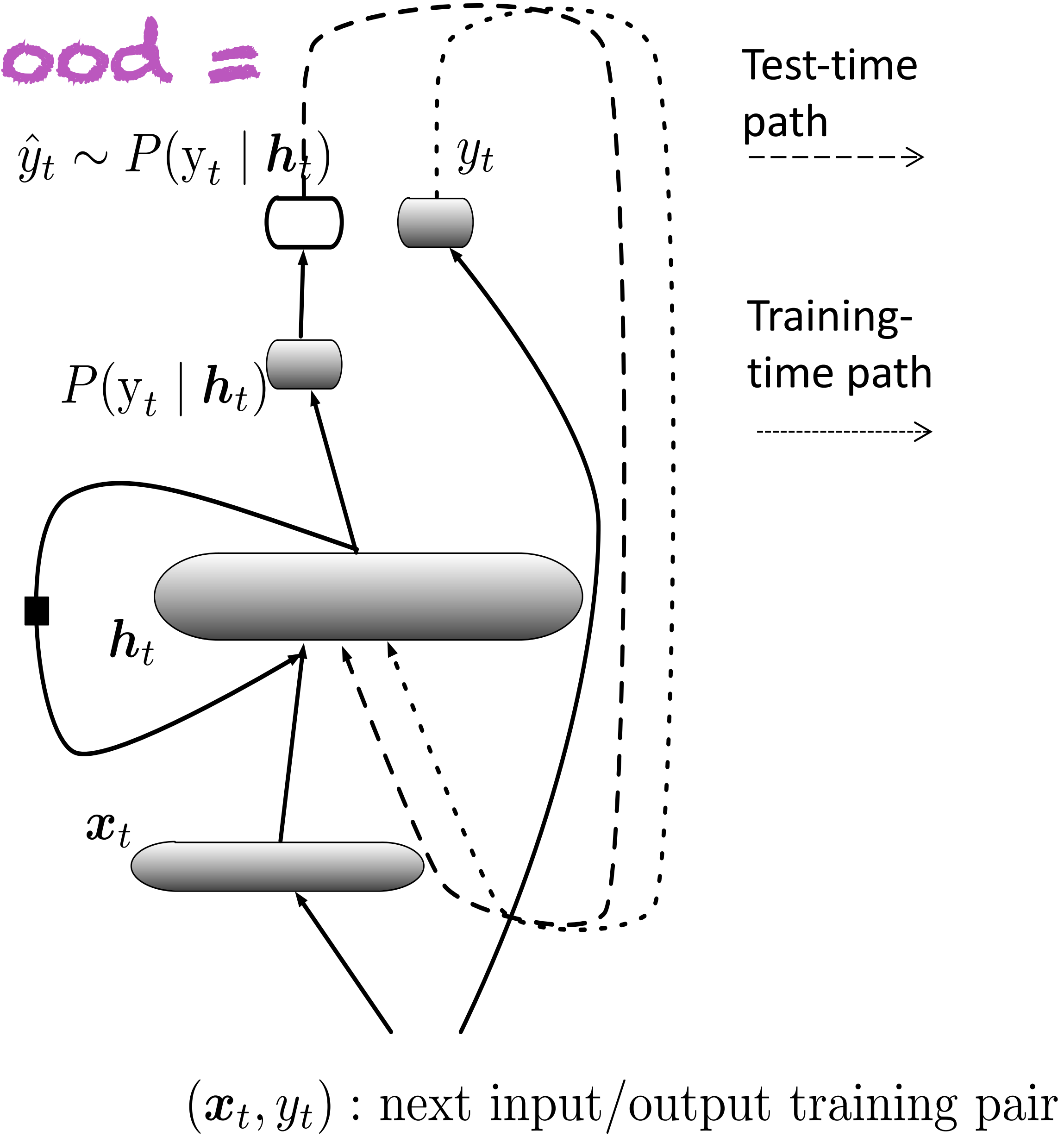


- Sequence to sequence



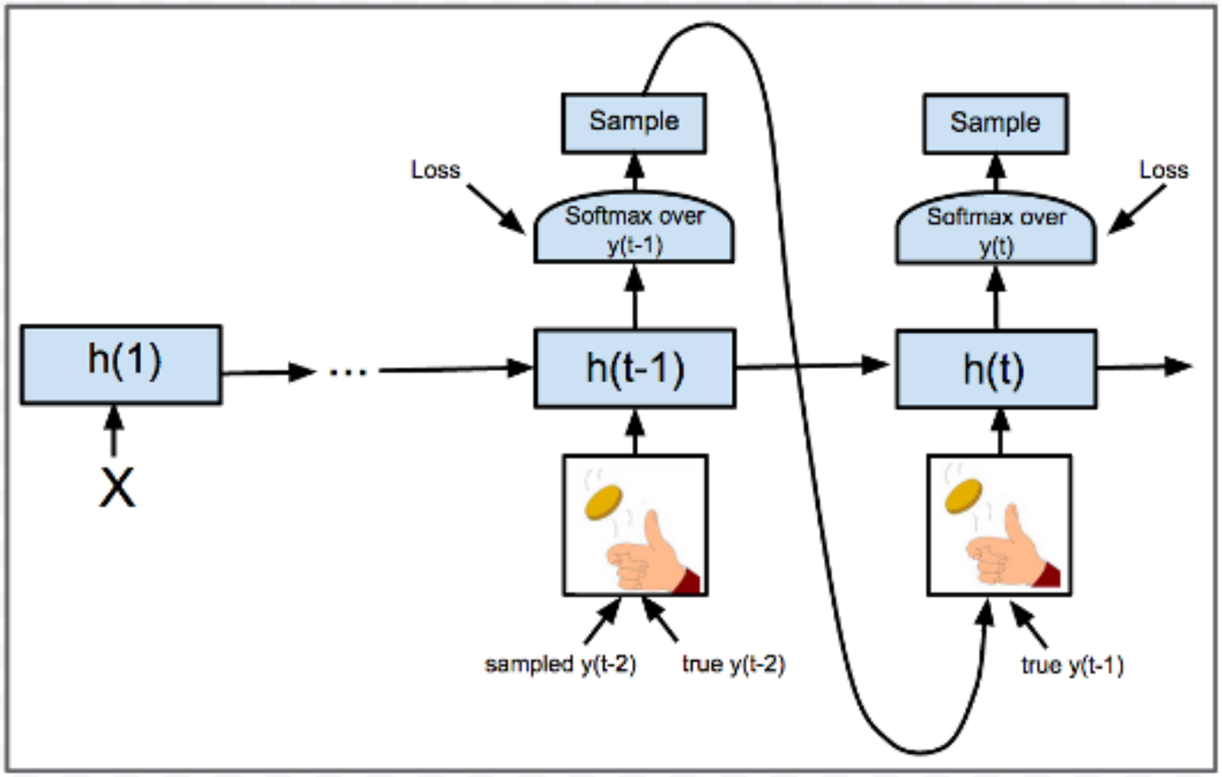
Maximum Likelihood = Teacher Forcing

- During training, past *y* in input is from training data
- At generation time, past *y* in input is generated
- Mismatch can cause ”compounding error”


$\hat{y}_t \sim P(\mathrm{y}_t \mid \boldsymbol{h}_t)$
y_t
Test-time path
Training-time path
$P(\mathrm{y}_t \mid \boldsymbol{h}_t)$
$\boldsymbol{h}_t$
$\boldsymbol{x}_t$
$(\boldsymbol{x}_t, y_t)$: next input/output training pair

Ideas to reduce the train/generate mismatch in teacher forcing

- Scheduled sampling *(S. Bengio et al, NIPS 2015)*



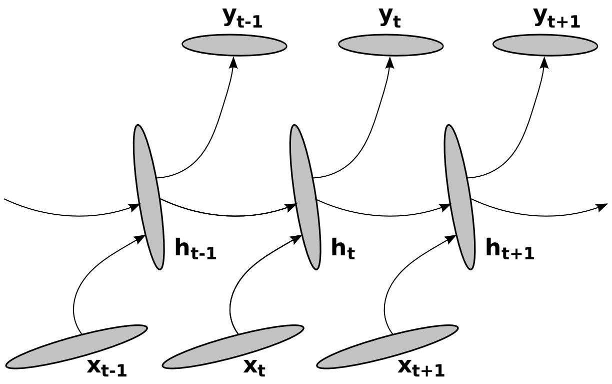
Related to
SEARN (Daumé et al 2009)
DAGGER (Ross et al 2010)

Gradually increase the probability of using the model’s samples vs the ground truth as input.

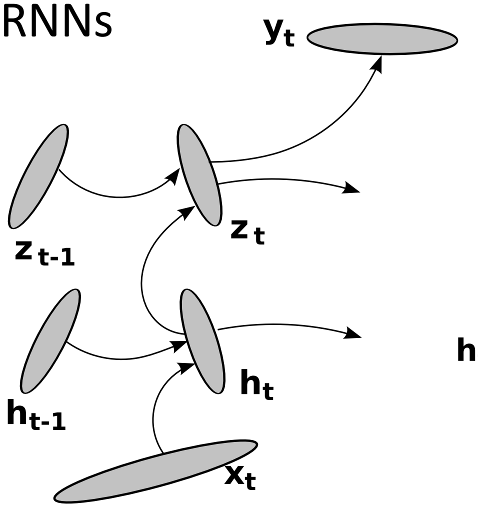
- Backprop through open-loop sampling recurrence & minimize long-term cost (but which one? GAN would be most natural → Professor Forcing)

Increasing the Expressive Power of RNNs with more Depth

- ICLR 2014, *How to construct deep recurrent neural networks*

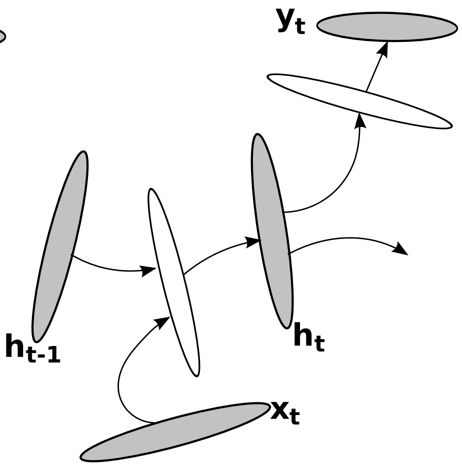


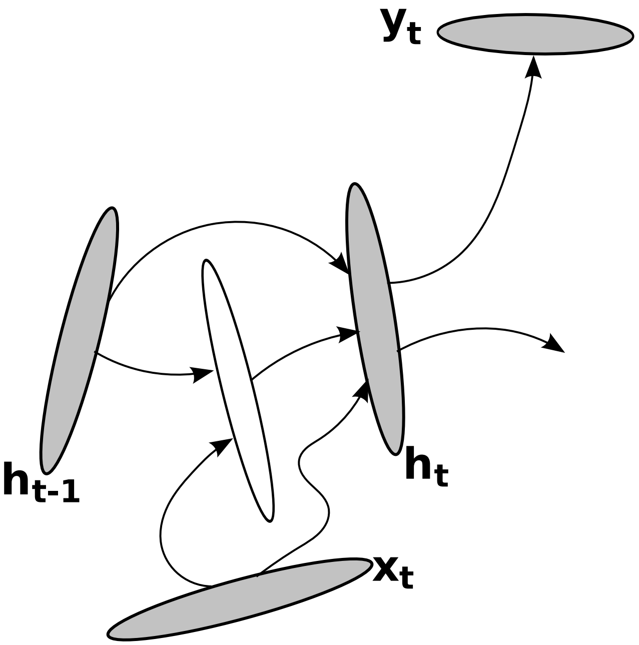
Ordinary RNNs



+ stacking

+ deep hid-to-out
+ deep hid-to-hid
+deep in-to-hid

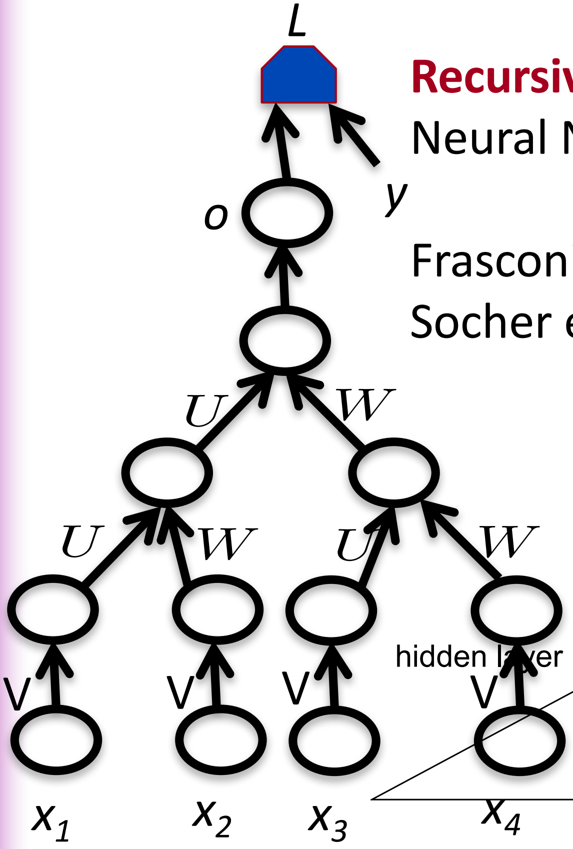




+ skip connections for
creating shorter paths

Bidirectional RNNs, Recursive Nets, Multidimensional RNNs, etc.

- The unfolded architecture needs not be a straight chain

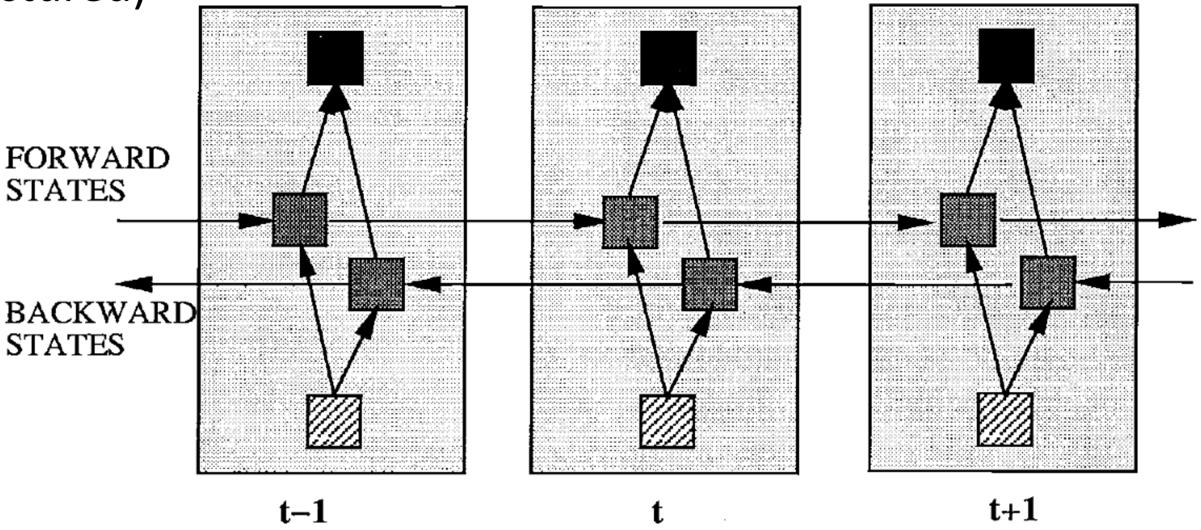


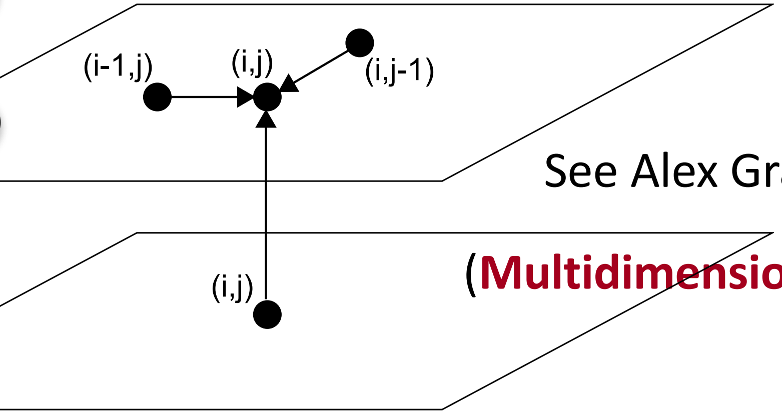
Recursive (tree-structured) Neural Nets:

Frasconi et al 97

Socher et al 2011

Bidirectional RNNs (Schuster and Paliwal, 1997)





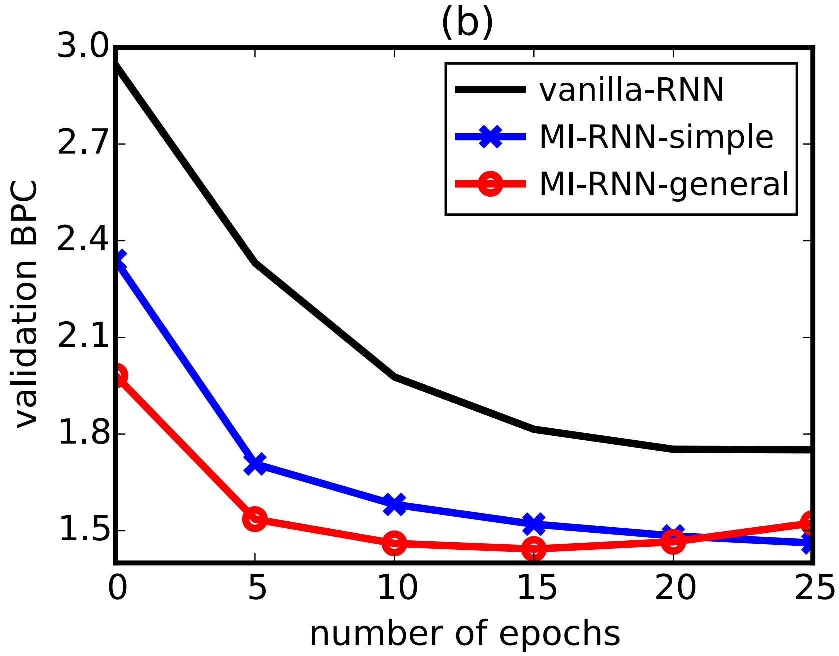
See Alex Graves's work, e.g., 2012

(**Multidimensional** RNNs, Graves et al 2007)

Multiplicative Interactions

(Wu et al, 2016, arXiv:1606.06630)

- Multiplicative Integration RNNs:



  - Replace

$$\phi(\mathbf{W}\boldsymbol{x} + \mathbf{U}\boldsymbol{z} + \mathbf{b})$$

  - By

$$\phi(\mathbf{W}\boldsymbol{x} \odot \mathbf{U}\boldsymbol{z} + \mathbf{b})$$

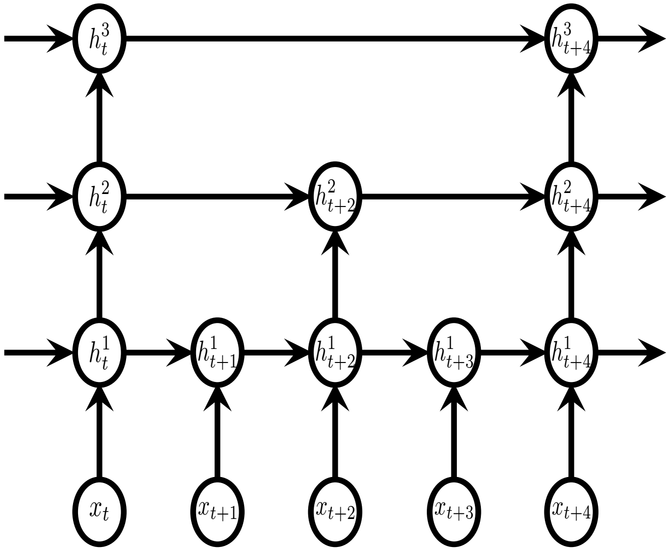
  - Or more general:

$$\phi(\boldsymbol{\alpha} \odot \mathbf{W}\boldsymbol{x} \odot \mathbf{U}\boldsymbol{z} + \boldsymbol{\beta}_1 \odot \mathbf{U}\boldsymbol{z} + \boldsymbol{\beta}_2 \odot \mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

Multiscale or Hierarchical RNNs

(Bengio & Elhihi, NIPS 1995)

- Motivation :
  - Gradients can propagate over longer spans through slow time-scale paths
- Approach :
  - Introduce a network architecture that update the states of its hidden layers with different speeds in order to capture multiscale representation of sequences.

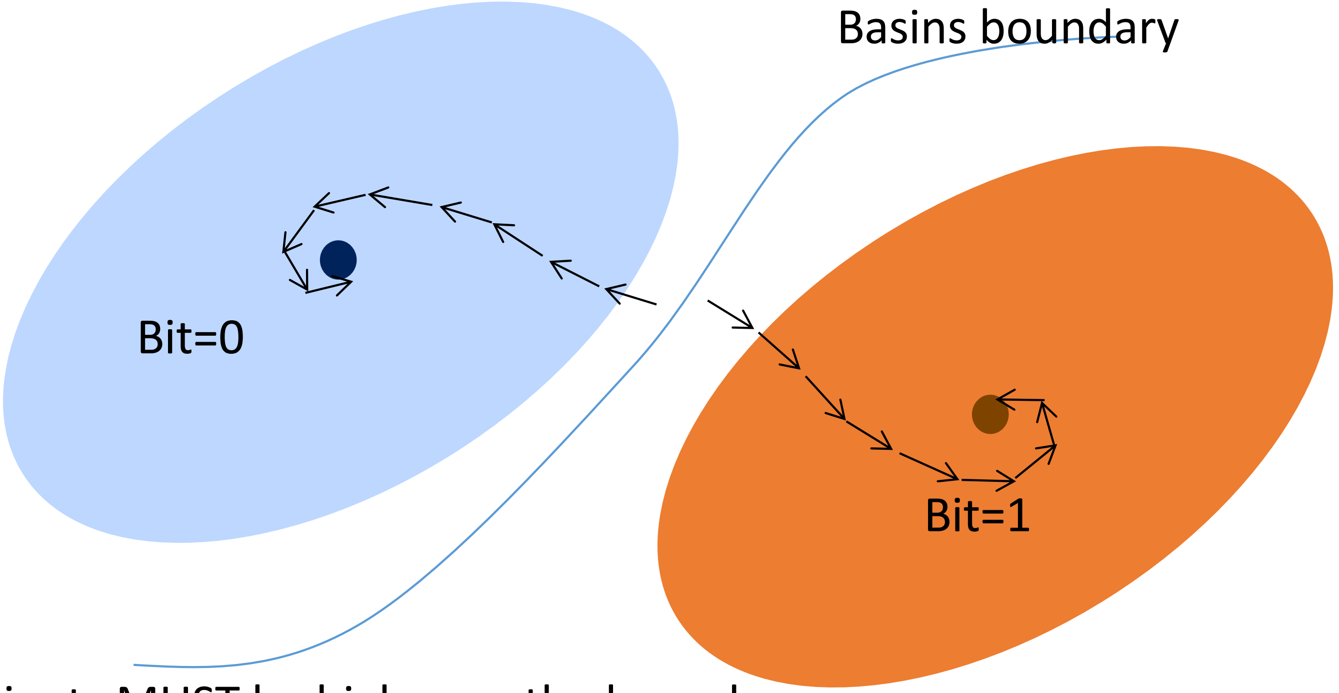


Learning Long-Term Dependencies with Gradient Descent is Difficult

Y. Bengio, P. Simard & P. Frasconi, IEEE Trans. Neural Nets, **1994**

How to store 1 bit? Dynamics with multiple basins of attraction in some dimensions

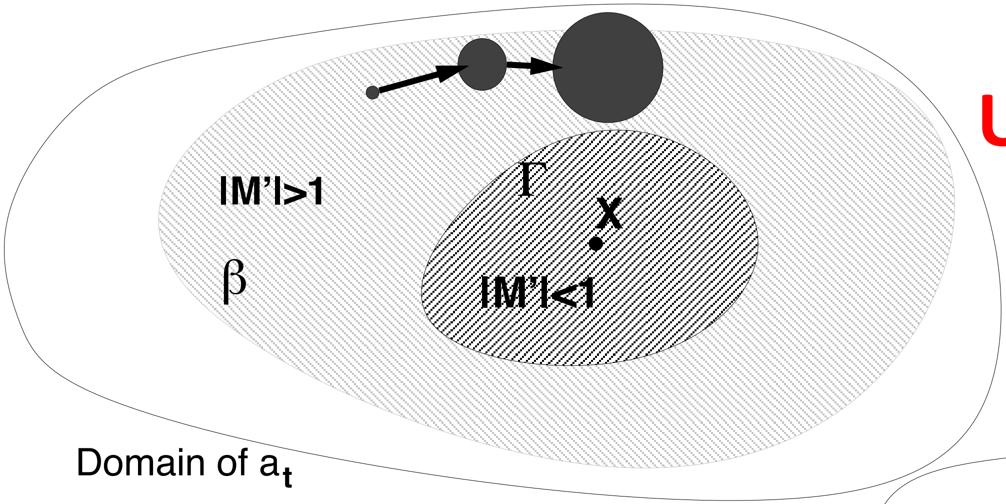
- Some subspace of the state can store 1 or more bits of information if the dynamical system has multiple basins of attraction in some dimensions



Note: gradients MUST be high near the boundary

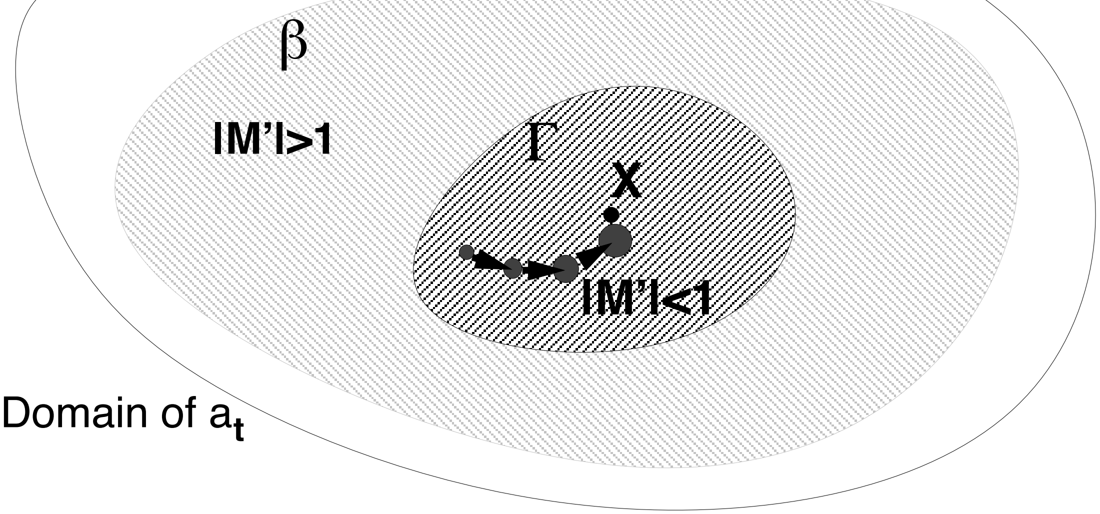
Robustly storing 1 bit in the presence of bounded noise

- With spectral radius > 1, noise can kick state out of attractor



UNSTABLE

- Not so with radius<1



CONTRACTIVE
→ STABLE

Storing Reliably ➔ Vanishing gradients

- Reliably storing bits of information requires spectral radius<1
- The product of T matrices whose spectral radius is < 1 is a matrix whose spectral radius converges to 0 at exponential rate in T

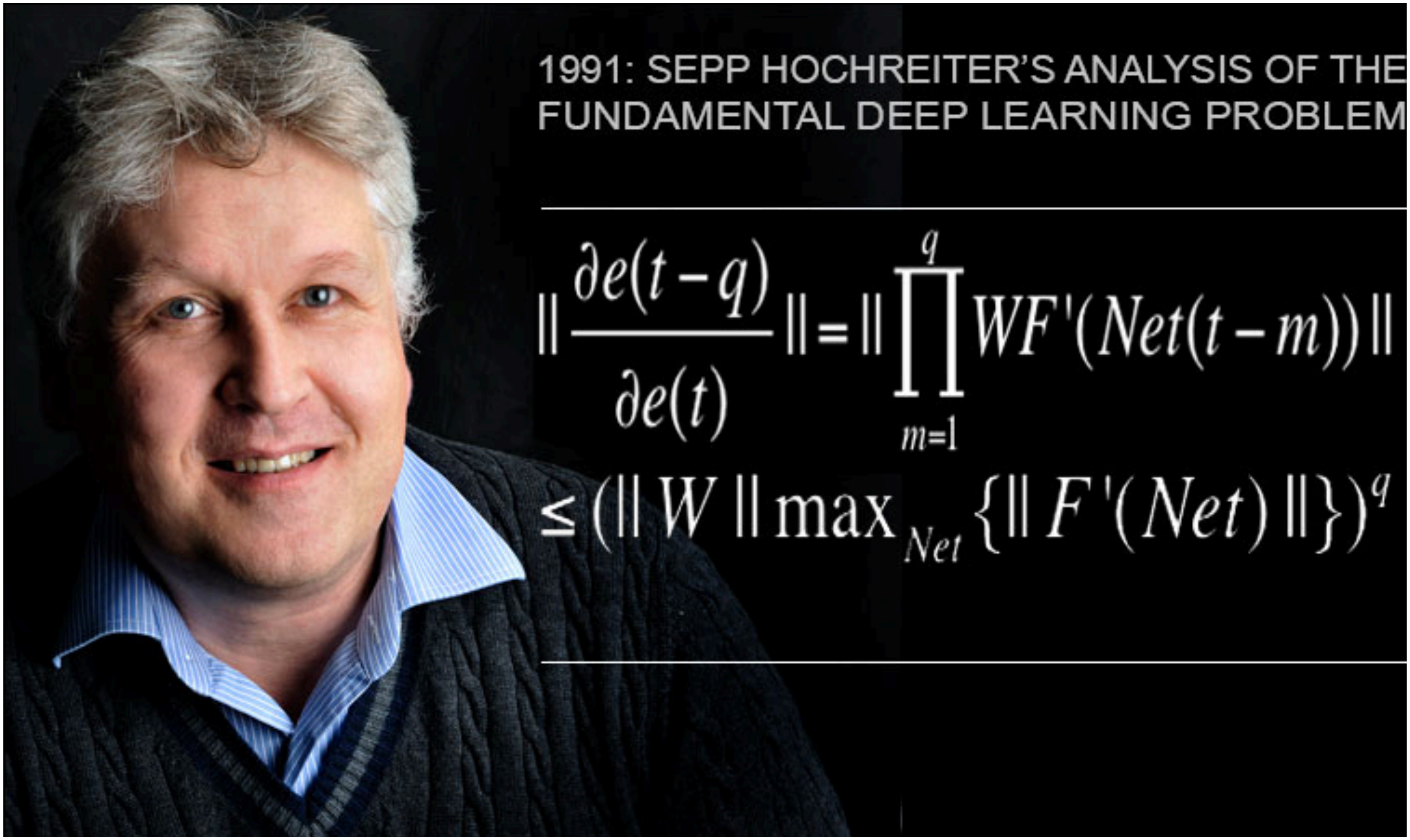
$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \cdots \frac{\partial s_{t+1}}{\partial s_t}$$

- If spectral radius of Jacobian is < 1 ➔ propagated gradients vanish

Vanishing or Exploding Gradients

- Hochreiter’s 1991 MSc thesis (in German) had independently discovered that backpropagated gradients in RNNs tend to either vanish or explode as sequence length increases



1991: SEPP HOCHREITER’S ANALYSIS OF THE FUNDAMENTAL DEEP LEARNING PROBLEM

$$\| \frac{\partial e(t-q)}{\partial e(t)} \| = \| \prod_{m=1}^{q} W F'(Net(t-m)) \|$$

$$\le (\| W \| \max_{Net} \{\| F'(Net) \|\})^{q}$$

Why it hurts gradient-based learning

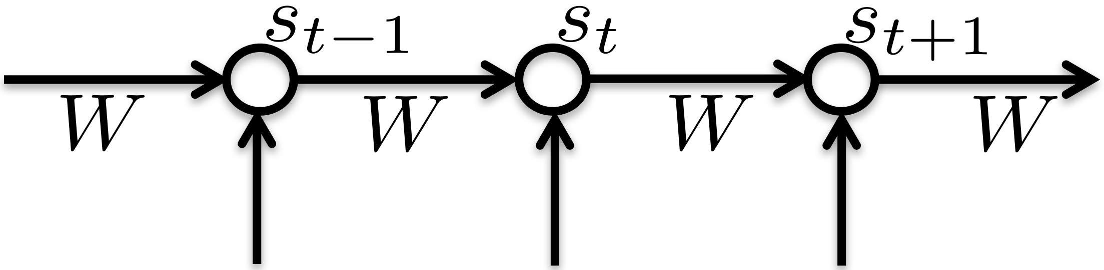
- Long-term dependencies get a weight that is exponentially smaller (in T) compared to short-term dependencies

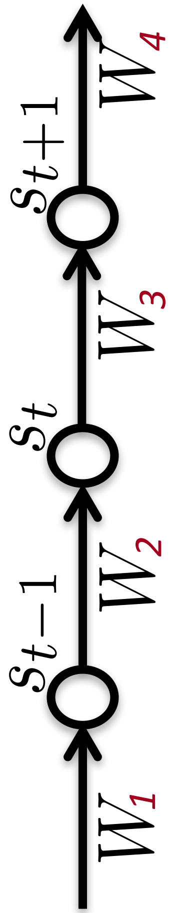
$$\frac{\partial C_t}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_t} \frac{\partial a_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W}$$

Becomes exponentially smaller for longer time differences, when spectral radius < 1

Vanishing Gradients in Deep Nets are Different from the Case in RNNs

- If it was just a case of vanishing gradients in deep nets, we could just rescale the per-layer learning rate, but that does not really fix the training difficulties.





- Can’t do that with RNNs because the weights are shared, & total true gradient = sum over different “depths”

$$\frac{\partial C_t}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_t} \frac{\partial a_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W}$$

To store information robustly the dynamics must be contractive

- The RNN gradient is a product of Jacobian matrices, each associated with a step in the forward computation. To store information robustly in a finite-dimensional state, the dynamics must be contractive [Bengio et al 1994].

$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \cdots \frac{\partial s_{t+1}}{\partial s_t}$$

Storing bits robustly requires e-values<1

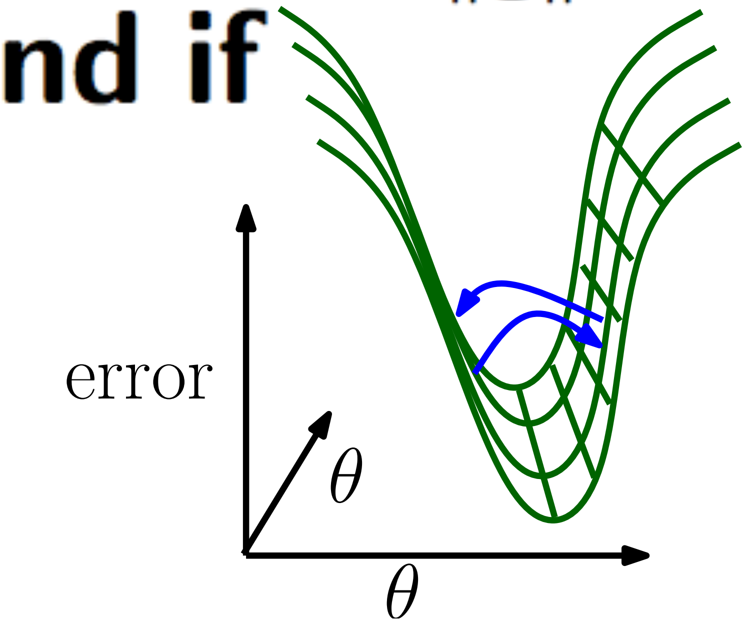
- Problems:
  - e-values of Jacobians > 1 → *gradients explode*
  - **or e-values < 1 → *gradients shrink & vanish***
  - or random → variance grows exponentially

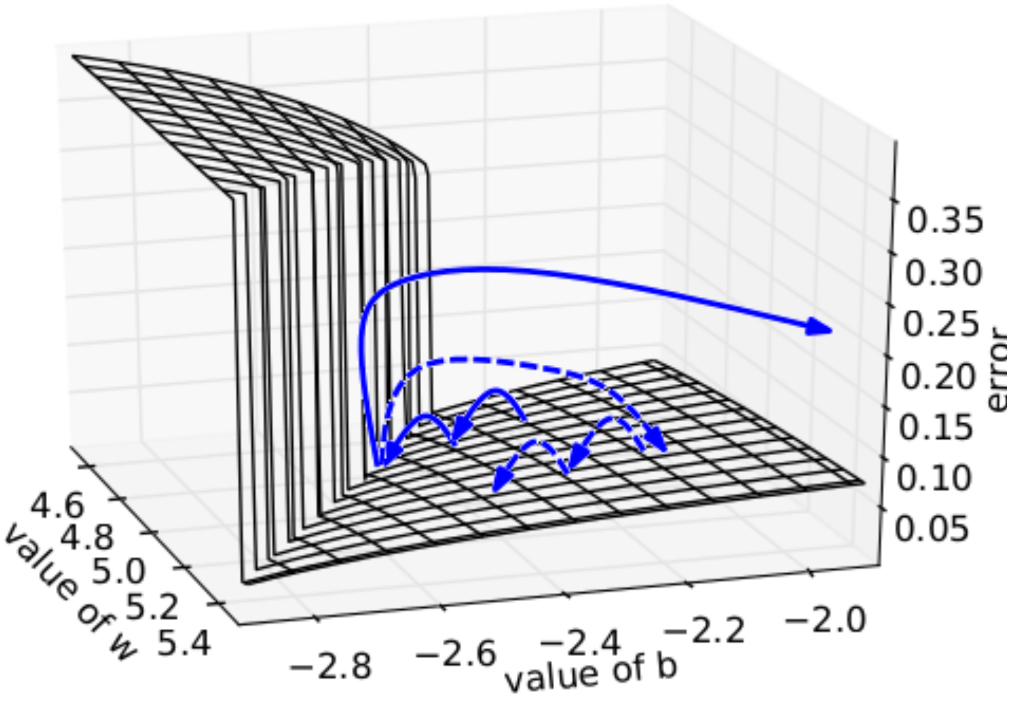
→ **Gradient clipping**

Dealing with Gradient Explosion by Gradient Norm Clipping

(Mikolov thesis 2012;
Pascanu, Mikolov, Bengio, ICML 2013)

```
$\hat{\mathbf{g}} \leftarrow \frac{\partial error}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if
```





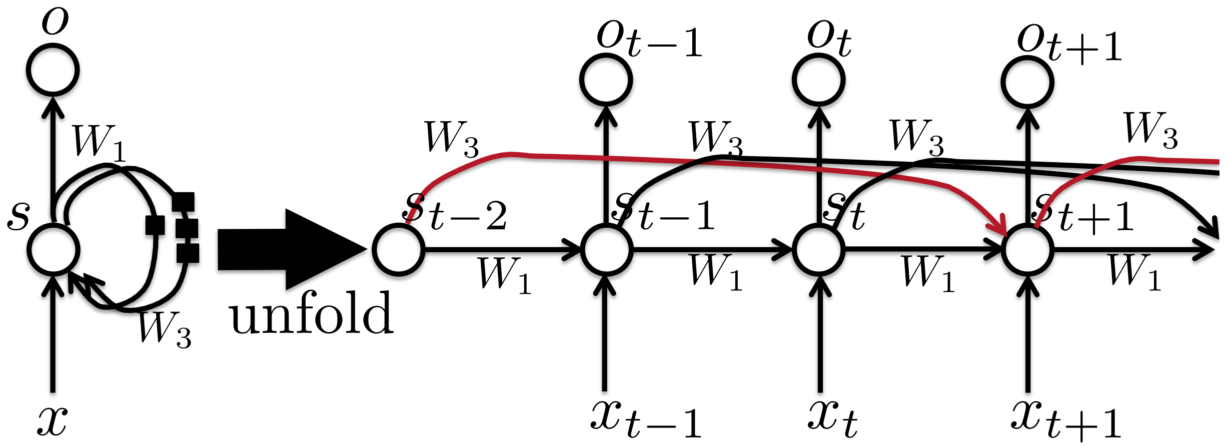
RNN Tricks

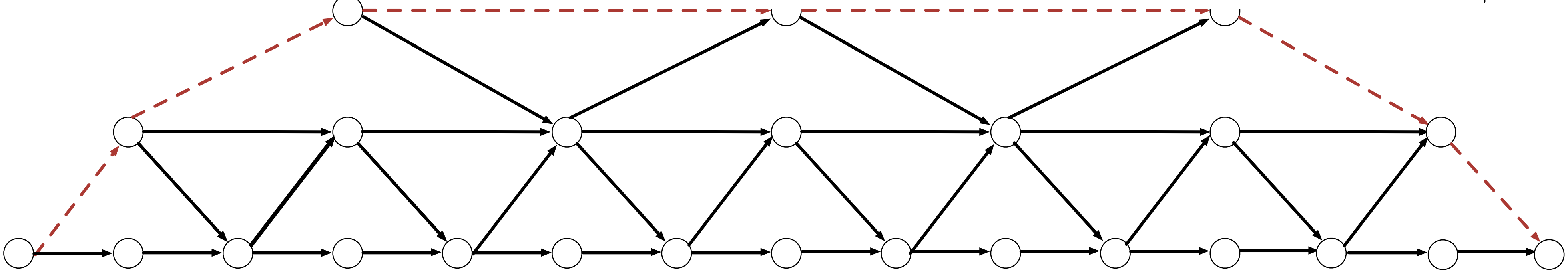
(Pascanu, Mikolov, Bengio, ICML 2013; Bengio, Boulanger & Pascanu, ICASSP 2013)

- Clipping gradients (avoid exploding gradients)
- Leaky integration (propagate long-term dependencies)
- Momentum (cheap 2nd order)
- Initialization (start in right ballpark avoids exploding/vanishing)
- Sparse Gradients (symmetry breaking)
- Gradient propagation regularizer (avoid vanishing gradient)
- Gated self-loops (LSTM & GRU, reduces vanishing gradient)

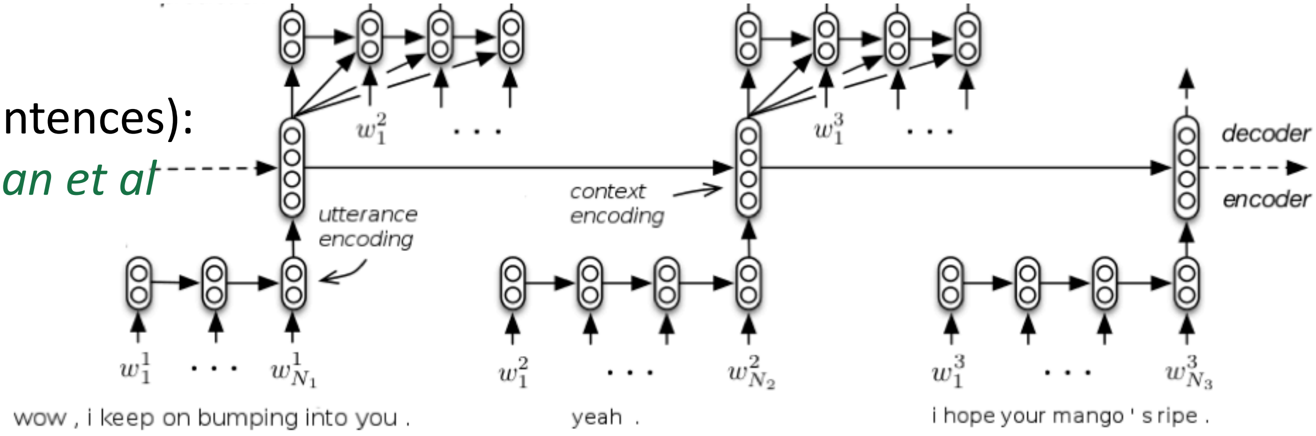
Delays & Hierarchies to Reach Farther

- Delays and multiple time scales, *Elhihi & Bengio NIPS 1995, Koutnik et al ICML 2014*
- *How to do this right?*
- *How to automatically and adaptively do it?*





Hierarchical RNNs (words / sentences): *Sordoni et al CIKM 2015, Serban et al AAAI 2016*

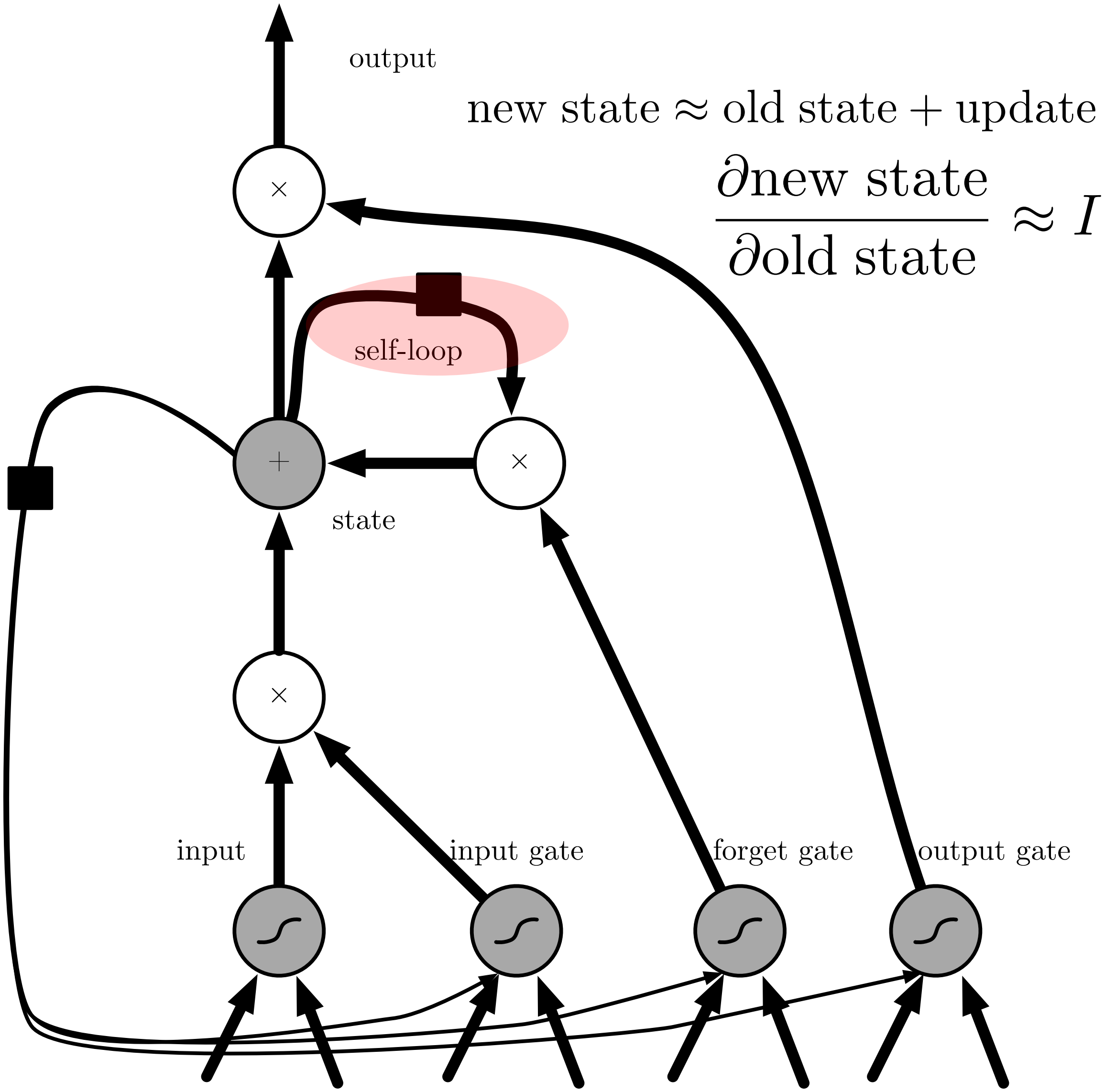


Fighting the vanishing gradient: LSTM & GRU

(Hochreiter 1991); first version of the LSTM, called Neural Long-Term Storage with self-loop

- Create a path where gradients can flow for longer with a self-loop
- Corresponds to an eigenvalue of Jacobian slightly less than 1
- LSTM is now **heavily used** *(Hochreiter & Schmidhuber 1997)*
- GRU light-weight version *(Cho et al 2014)*

LSTM: (Hochreiter & Schmidhuber 1997)



$$\text{new state} \approx \text{old state} + \text{update}$$

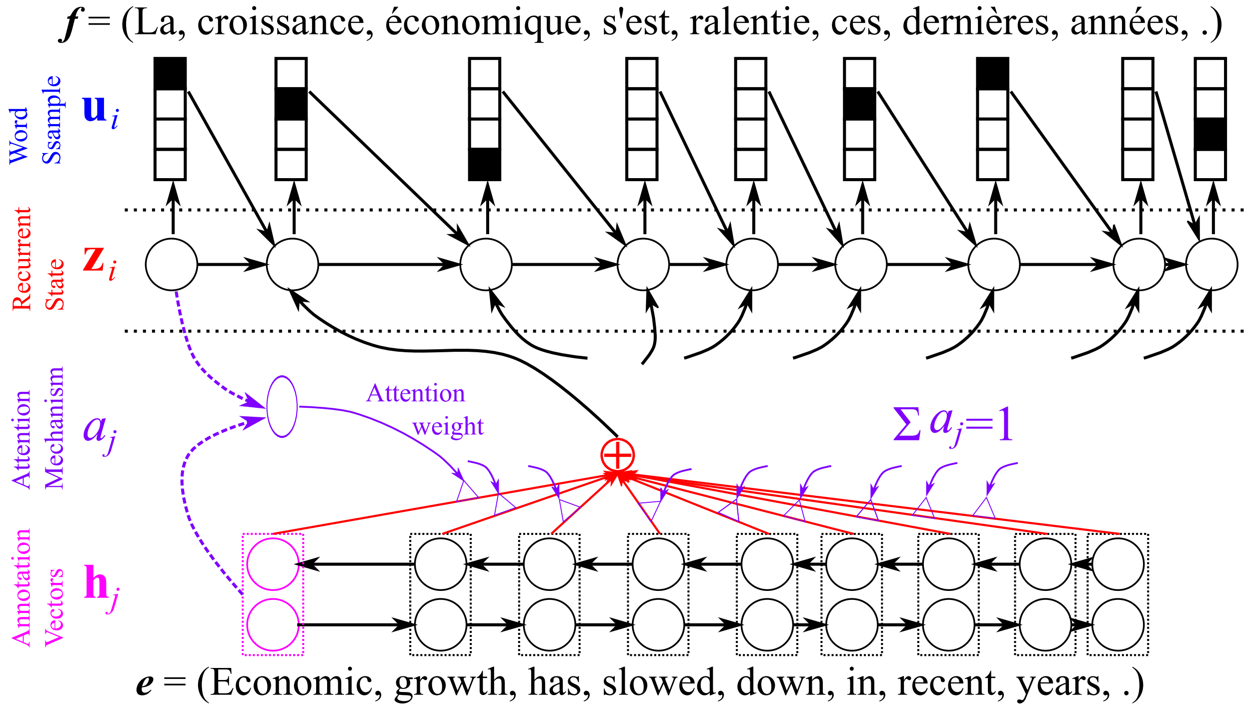
$$\frac{\partial \text{new state}}{\partial \text{old state}} \approx I$$

Attention Mechanisms

Gating for Attention-Based Neural Machine Translation

Related to earlier Graves 2013 for generating handwriting

- (Bahdanau, Cho & Bengio, arXiv sept. 2014)
- (Jean, Cho, Memisevic & Bengio, arXiv dec. 2014)

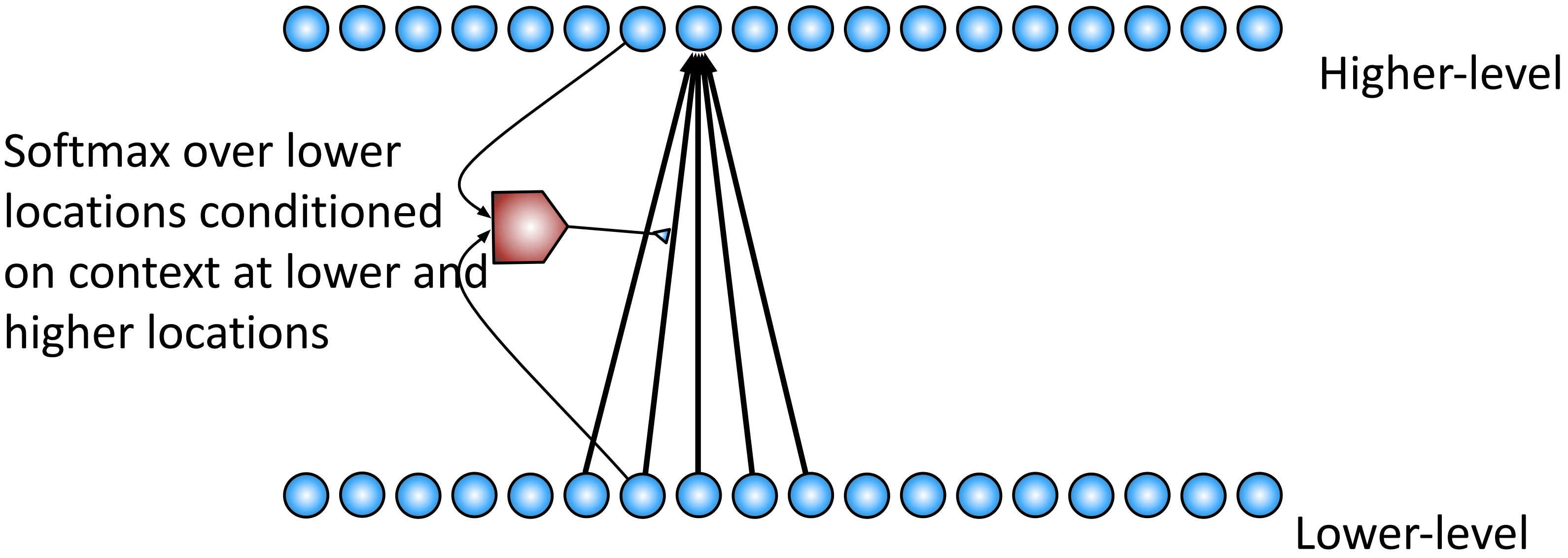


f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Word Ssample
$\mathbf{u}_i$
Recurrent State
$\mathbf{z}_i$
Attention Mechanism
a_j
Attention weight
$\Sigma a_j = 1$
Annotation Vectors
$\mathbf{h}_j$
e = (Economic, growth, has, slowed, down, in, recent, years, .)

What's New with Deep Learning?

- Incorporating the idea of **attention, using GATING units,** has unlocked a breakthrough in machine translation:

Neural Machine Translation (ICLR'2015)

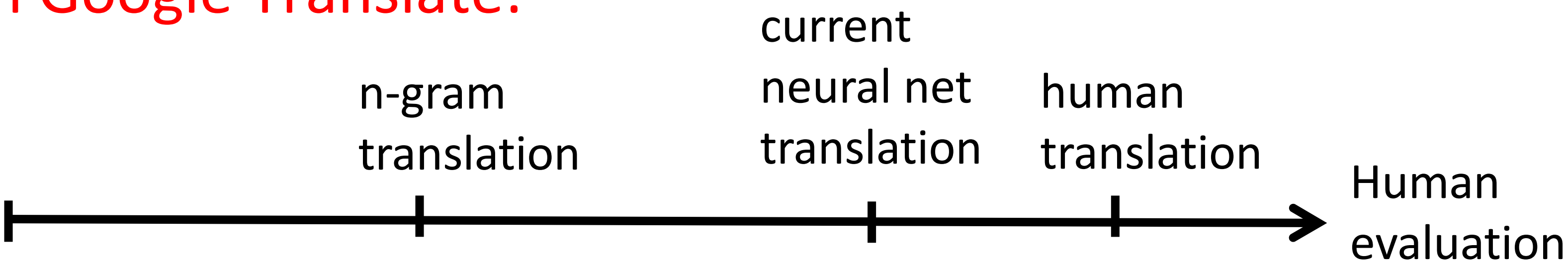


Higher-level

Softmax over lower locations conditioned on context at lower and higher locations

Lower-level

- Now in Google Translate:



n-gram translation

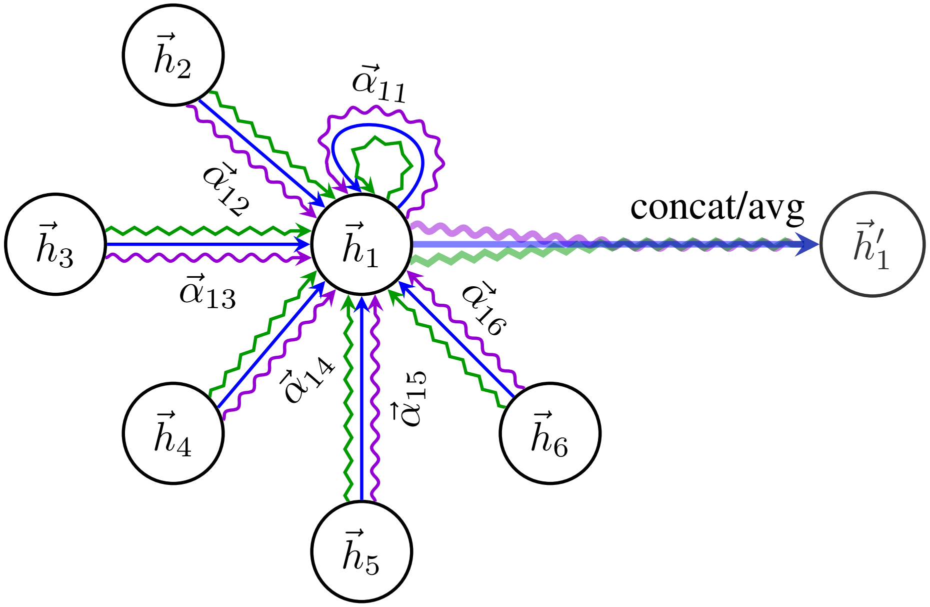
current neural net translation

human translation

Human evaluation

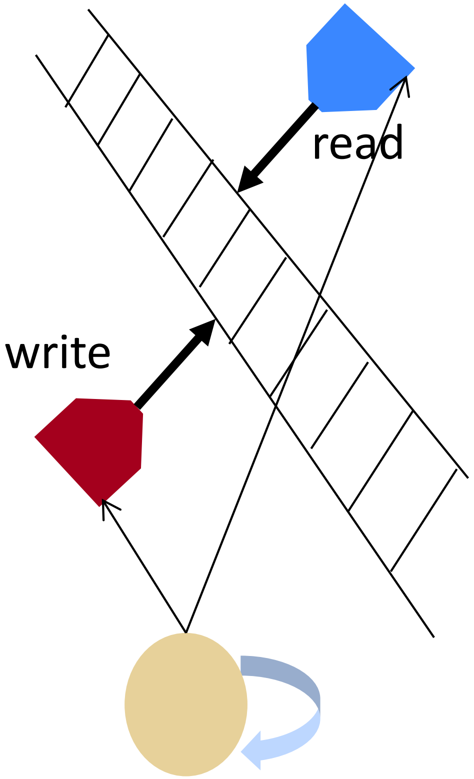
Graph Attention Networks
Velickovic et al, ICLR 2018

- Handle variable-size neighborhood of each node using the same neural net by using an attention mechanism to aggregate information from the neighbors
- Use multiple attention heads to collect different kinds of information



What's New with Deep Learning?

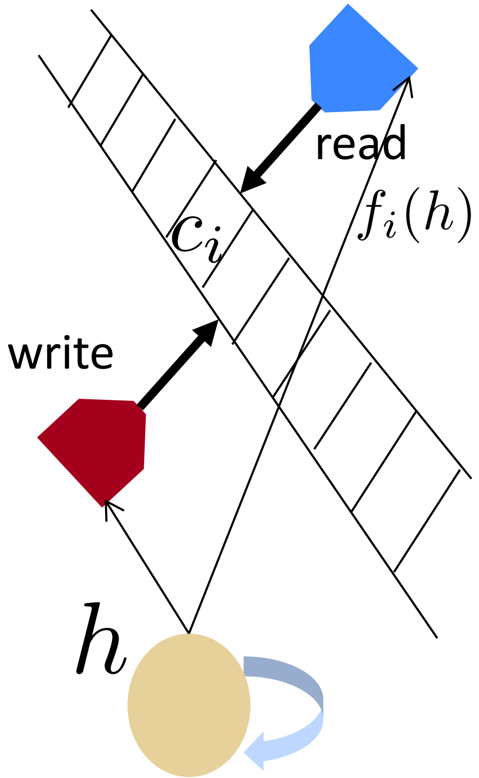
- Attention has also opened the door to neural nets which can write to and read from a memory
  - 2 systems:
    - Cortex-like (state controller and representations)
      - System 1, intuition, fast heuristic answer
    - Hippocampus-like (memory) + prefrontal cortex
      - System 2, slow, logical, sequential



- Memory-augmented networks gave rise to
  - Systems which reason
    - Sequentially combining several selected pieces of information (from the memory) in order to obtain a conclusion
  - Systems which answer questions
    - Accessing relevant facts and combining them

Attention Mechanisms for Memory Access

- Neural Turing Machines *(Graves et al 2014)*
- and Memory Networks *(Weston et al 2014)*
- Use a content-based attention mechanism *(Bahdanau et al 2014)* to control the read and write access into a memory
- The attention mechanism outputs a softmax over memory locations



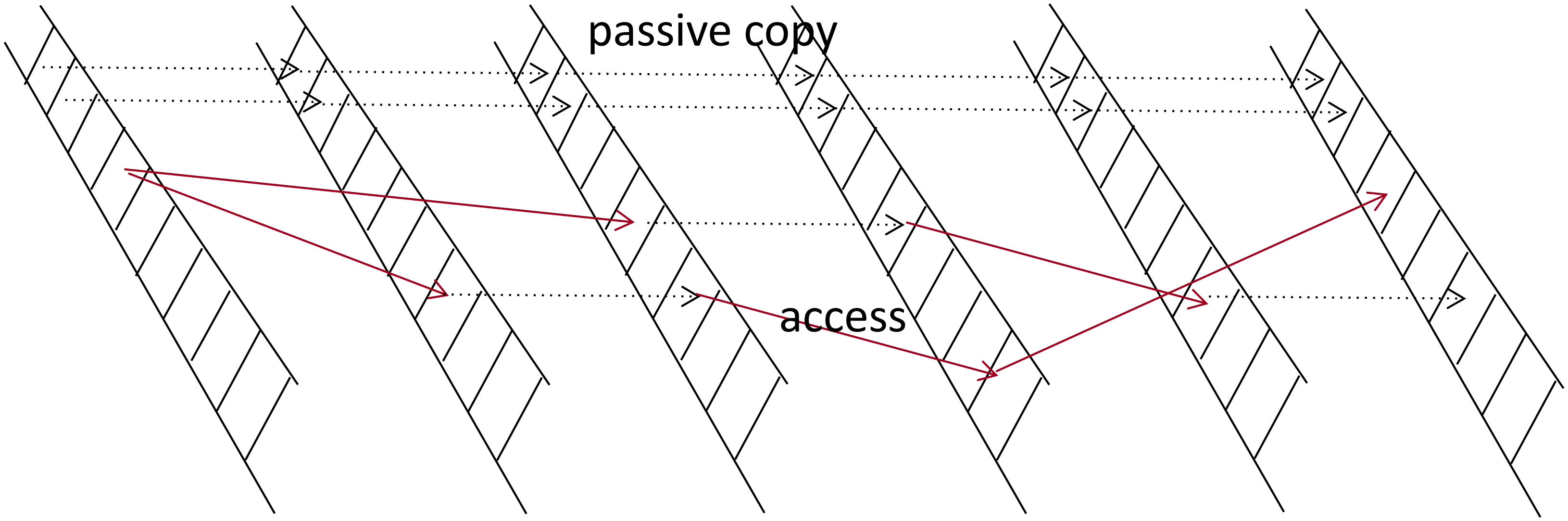
$$\alpha = \frac{e^{f_i(h)}}{\sum_i e^{f_i(h)}}$$

$$r = \sum_i \alpha_i c_i$$

Read = weighted average of attended contents

Large Memory Networks: Sparse Access Memory for Long-Term Dependencies

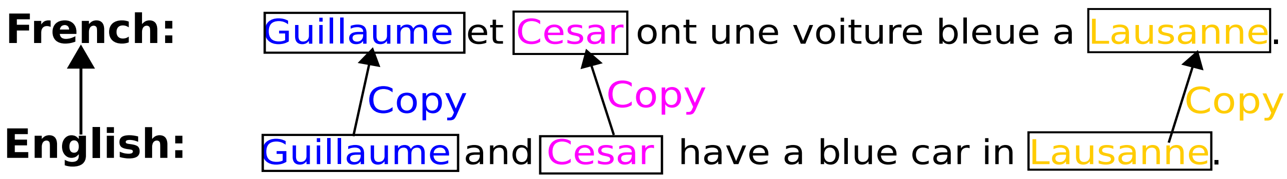
- Memory = part of the state
- Memory-based networks are special RNNs
- A mental state stored in an external memory can stay for arbitrarily long durations, until it is overwritten (partially or not)
- Forgetting = vanishing gradient.
- Memory = **higher-dimensional state**, avoiding or reducing the need for forgetting/vanishing



Pointing the Unknown Words

Gulcehre, Ahn, Nallapati, Zhou & Bengio ACL 2016
Based on 'Pointer Networks', Vinyals et al 2015

The next word generated can either come from vocabulary or is copied from the input sequence.



French: Guillaume et Cesar ont une voiture bleue a Lausanne.

English: Guillaume and Cesar have a blue car in Lausanne.

Table 5: Europarl Dataset (EN-FR)

Machine Translation

	BLEU-4
NMT	20.19
NMT + PS	**23.76**

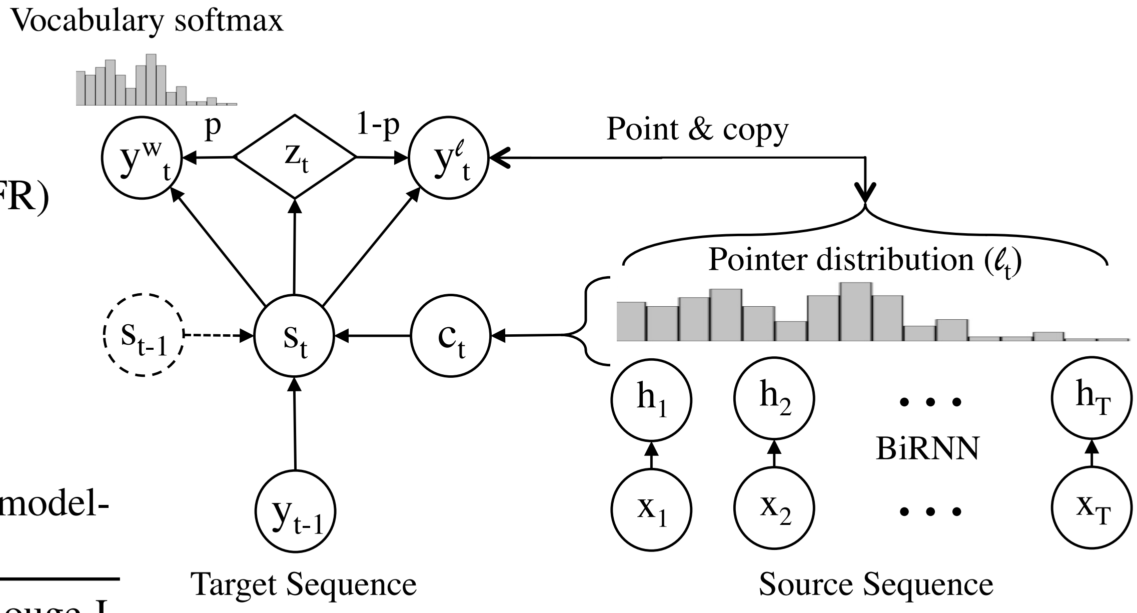


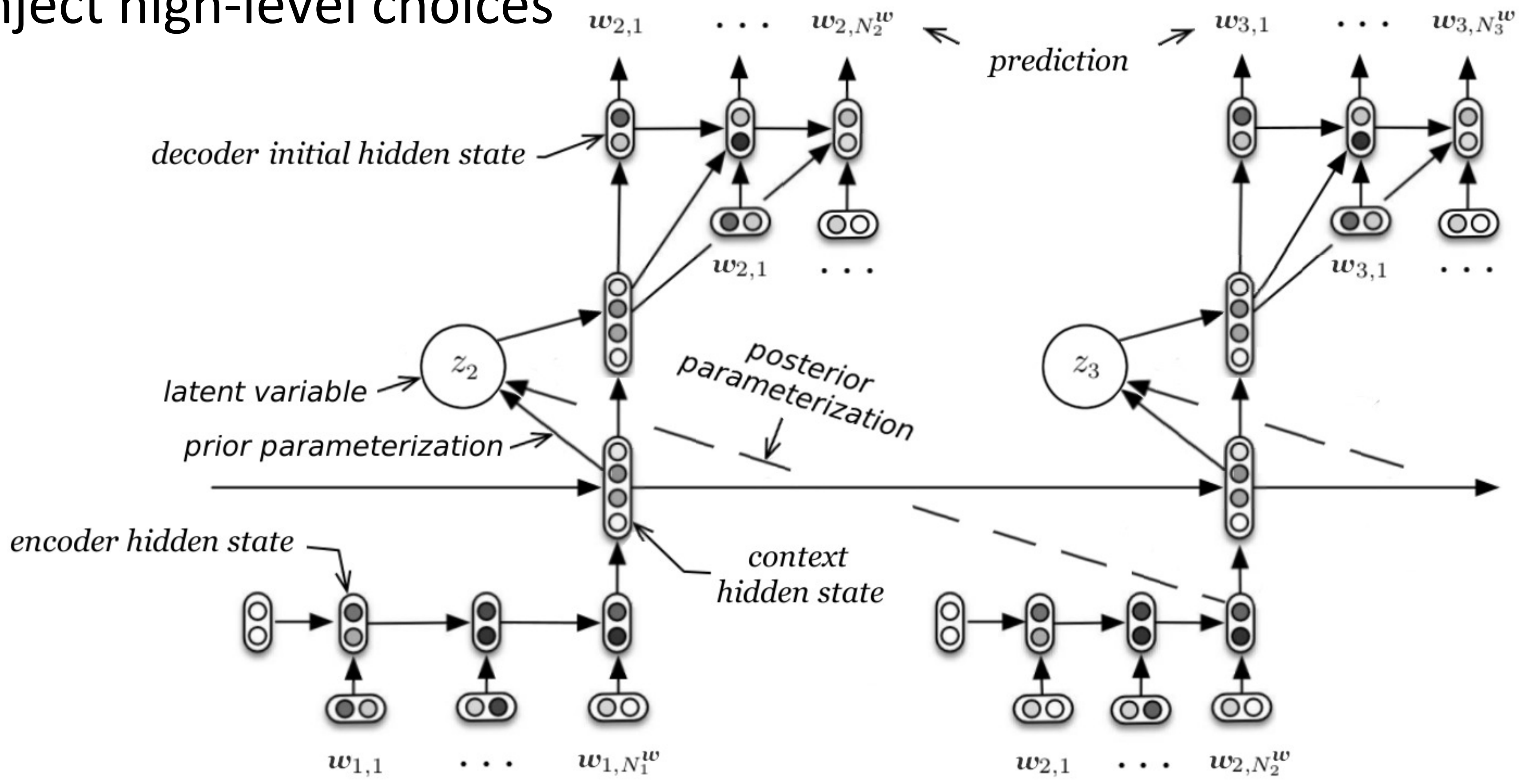
Table 3: Results on Gigaword Corpus for modeling UNK's with pointers in terms of recall.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	36.45	17.41	33.90
NMT + lvt + PS	**37.29**	**17.75**	**34.70**

Text summarization

Variational Hierarchical RNNs for Dialogue Generation (Serban et al 2016)

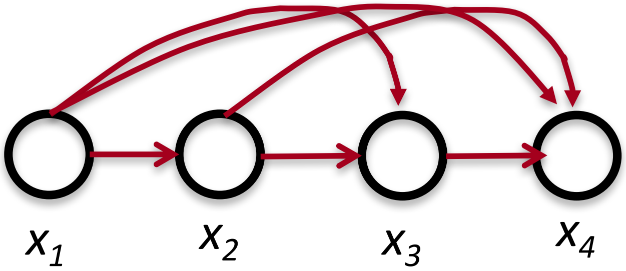
- Lower level = words of an utterance (turn of speech)
- Upper level = state of the dialogue
- Inject high-level choices



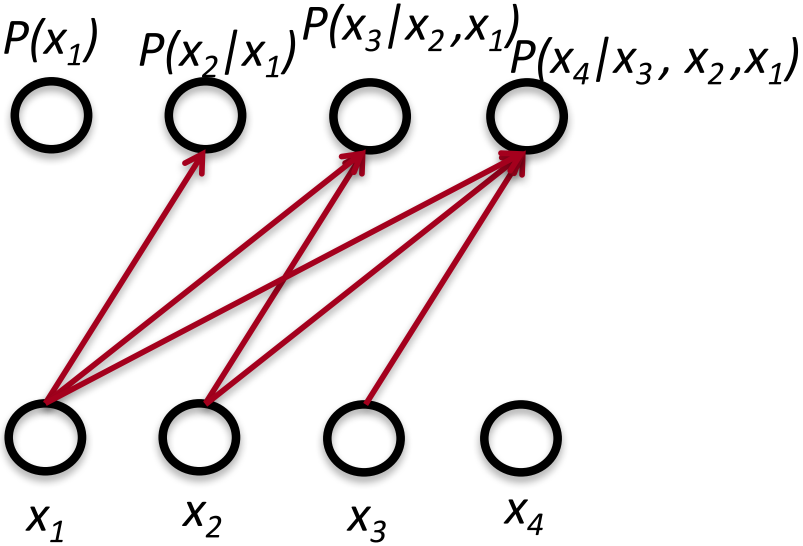
Auto-Encoders and Generative Neural Networks

Neural Auto-Regressive Models

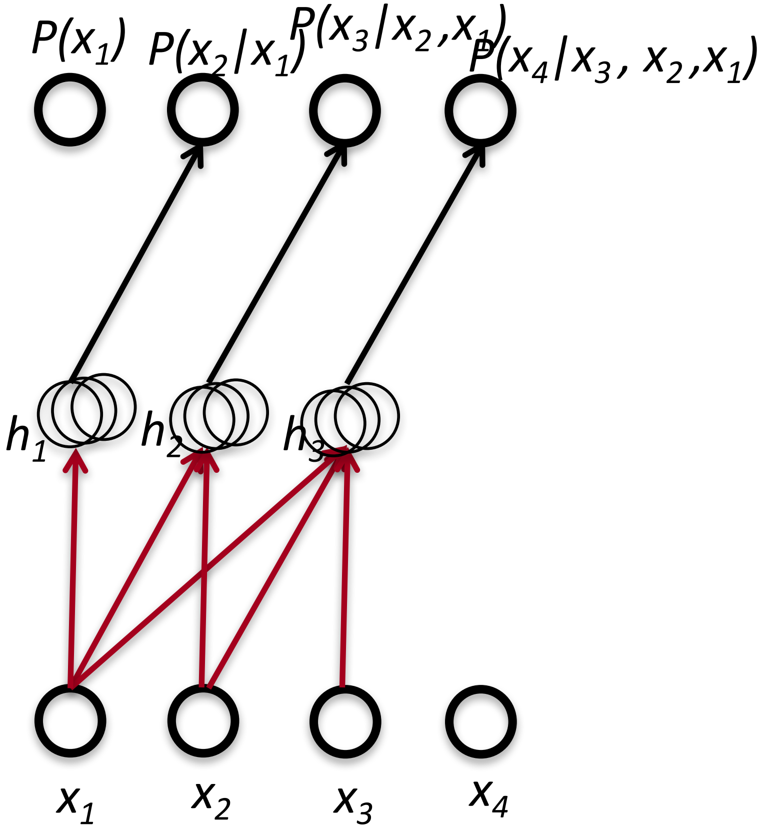
$$P(\mathbf{x}) = P(x_1, \dots x_T) = \prod_{t=1}^{T} P(x_t | x_{t-1}, x_{t-2}, \dots x_1)$$



- Decomposes the joint of a fully observed directed model in terms of conditionals
- Logistic auto-regressive: *(Frey 1997)*



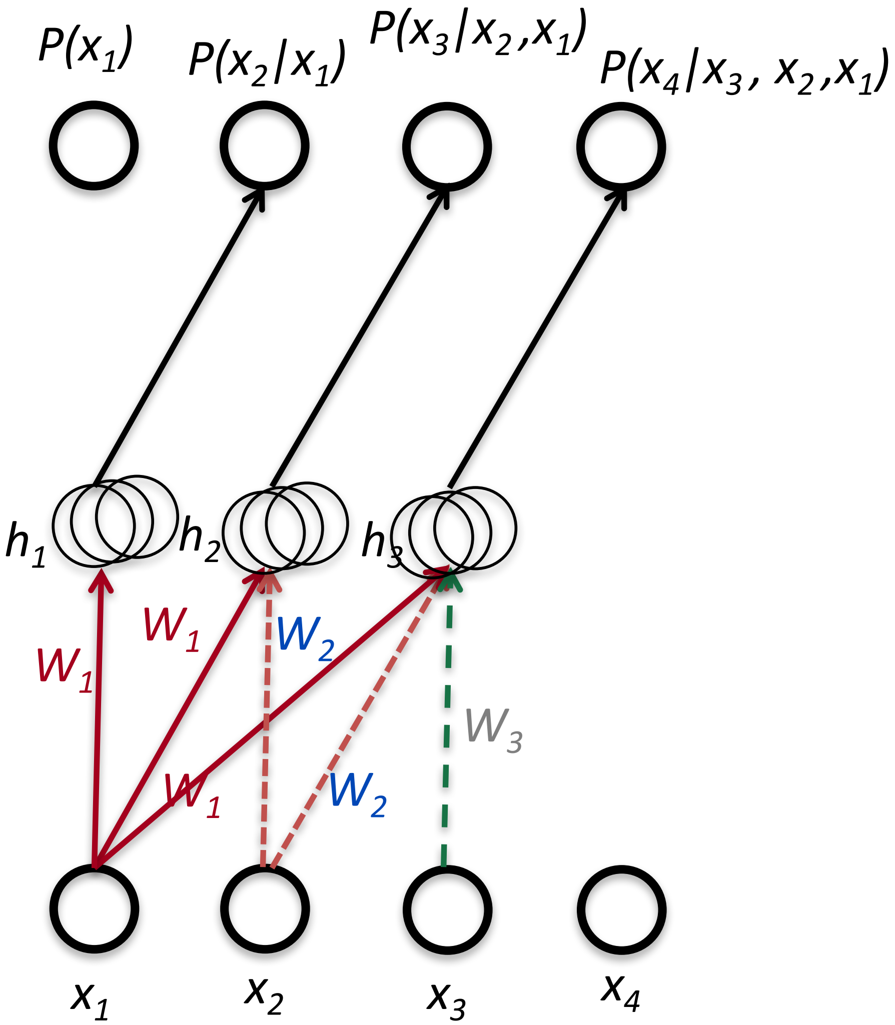
- First neural version: *(Bengio&Bengio NIPS'99)*



NADE: Neural AutoRegressive Density Estimator

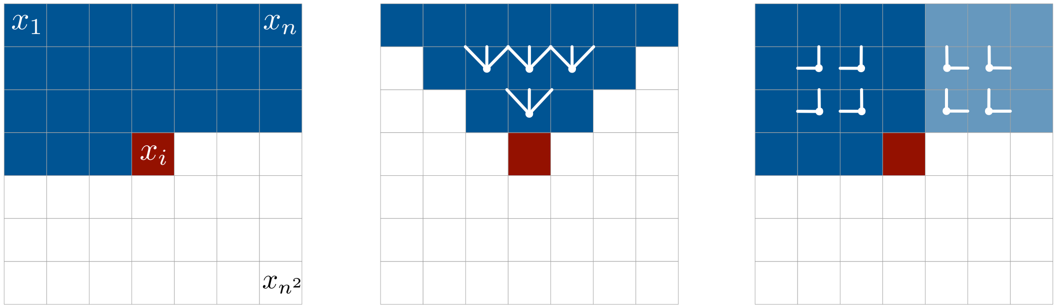
(Larochelle & Murray AISTATS 2011)

- Introduces smart sharing between some weights so that the different hidden groups use the same weights to the same input but look at more and more of the inputs.

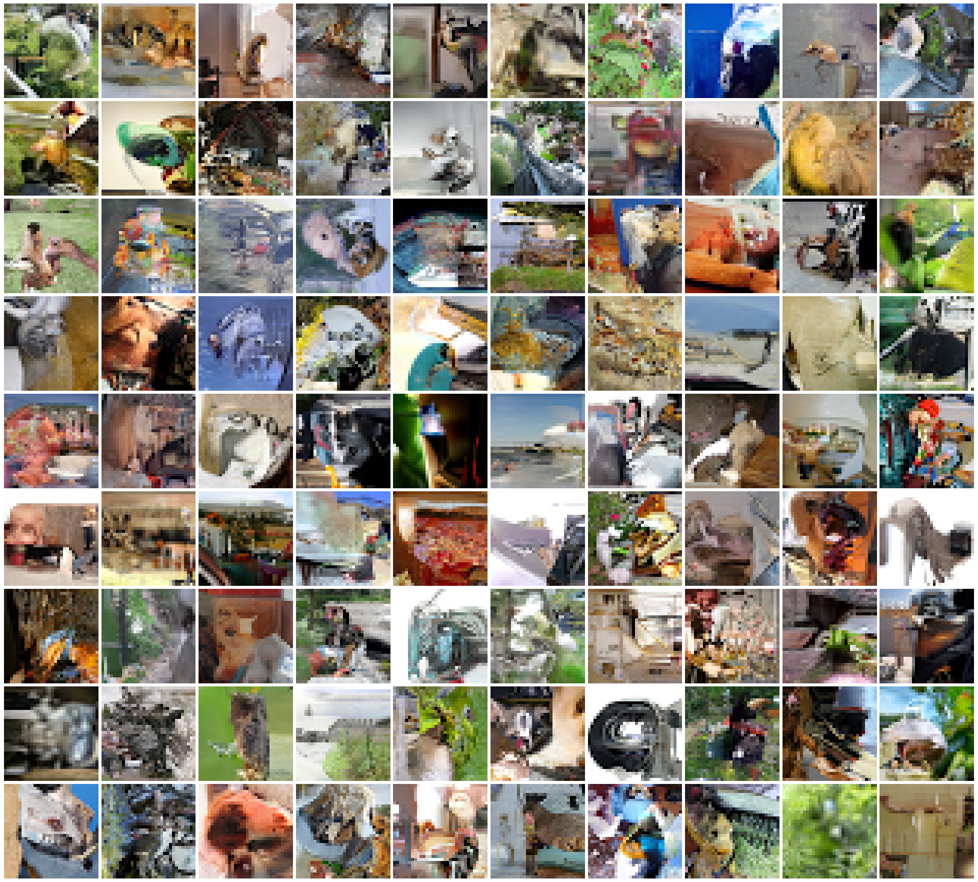


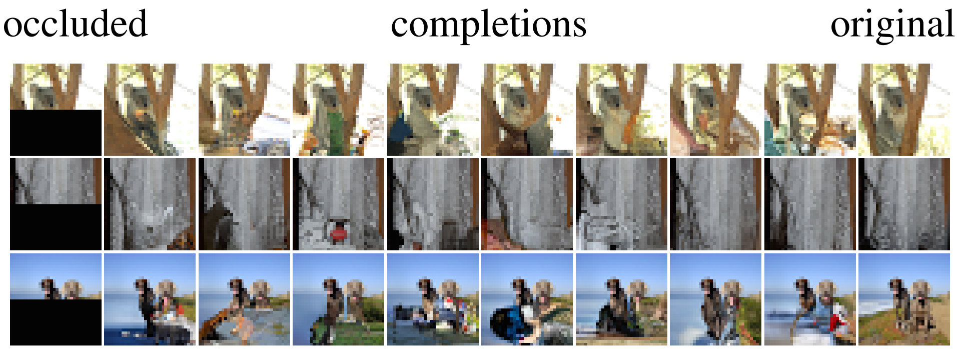
Pixel RNNs

(van den Oord et al ICML 2016, best paper)



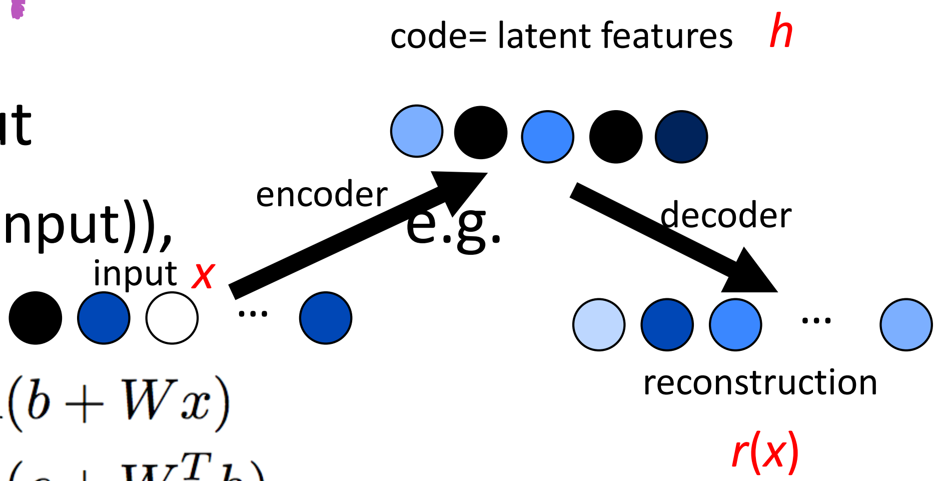
- Similar to NADE and RNNs but for 2-D images
- Surprisingly sharp and realistic generation
- Gets texture right but not necessarily global structure





Unsupervised Learning of Representations: Simple Auto-Encoders

- MLP whose target output = input
- Reconstruction=decoder(encoder(input)), e.g.



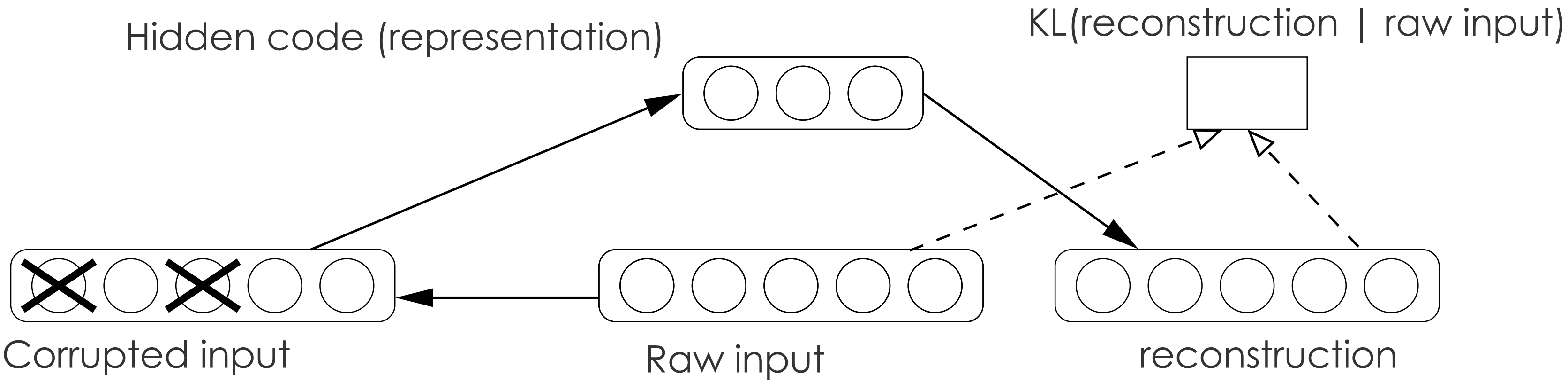
$$
\begin{aligned}
h &= \tanh(b + Wx) \\
\text{reconstruction} &= \tanh(c + W^T h) \\
\text{Loss } L(x, \text{reconstruction}) &= ||\text{reconstruction} - x||^2
\end{aligned}
$$

- Code = new coordinate system
- Encoder and decoder can have more layers
- Reconstruction can be probability distribution

Denoising Auto-Encoder

(Vincent et al 2008)

- Corrupt the input during training only
- Train to reconstruct the uncorrupted input



- Encoder & decoder: any parametrization
- As good or better than RBMs for unsupervised pre-training

Denoising Auto-Encoder

- Learns a vector field pointing towards hig probability direction (Alain & Bengio 2013)

$r(x)-x \propto dlogp(x)/dx$

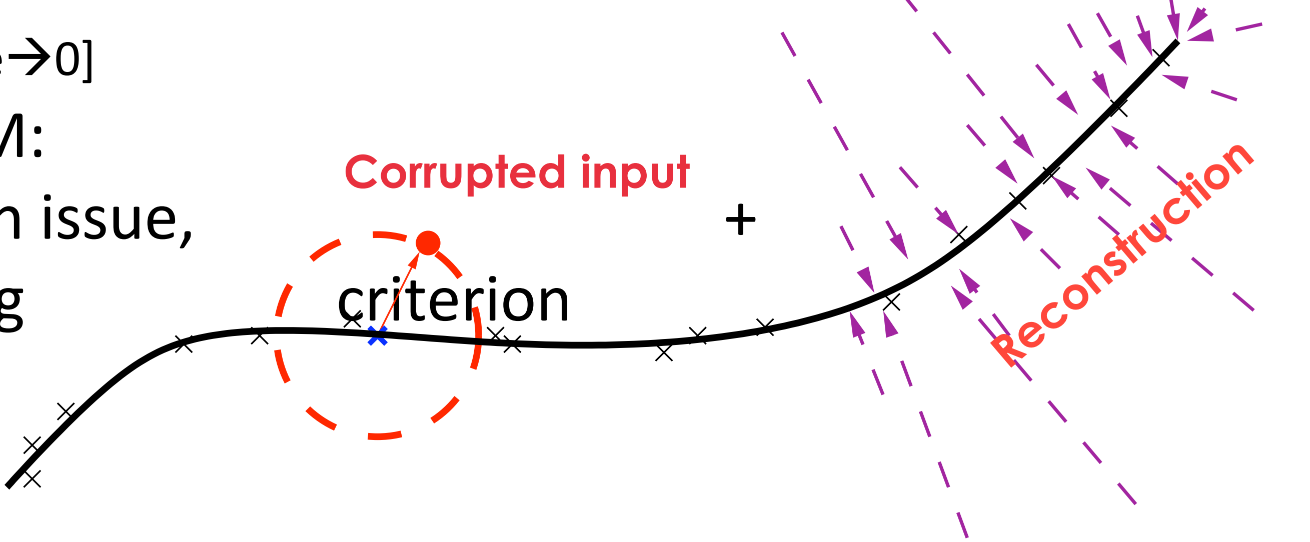
- Some DAEs correspond to a kind of Gaussian RBM with *regularized* Score Matching (Vincent 2011)

[equivalent when noise→0]

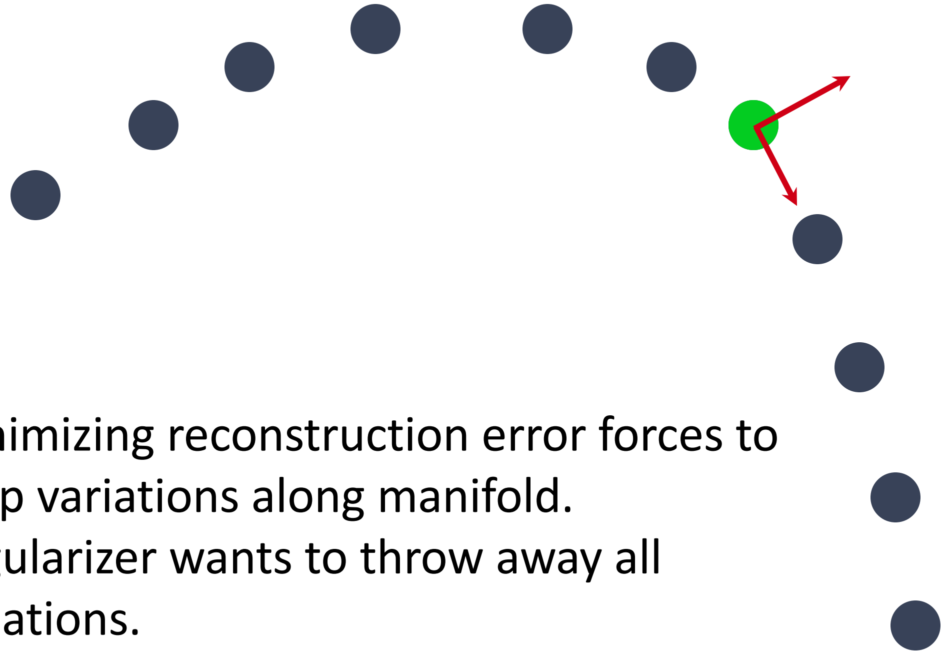
- Compared to RBM:

No partition function issue, can measure training criterion

prior: examples concentrate near a lower dimensional "manifold"

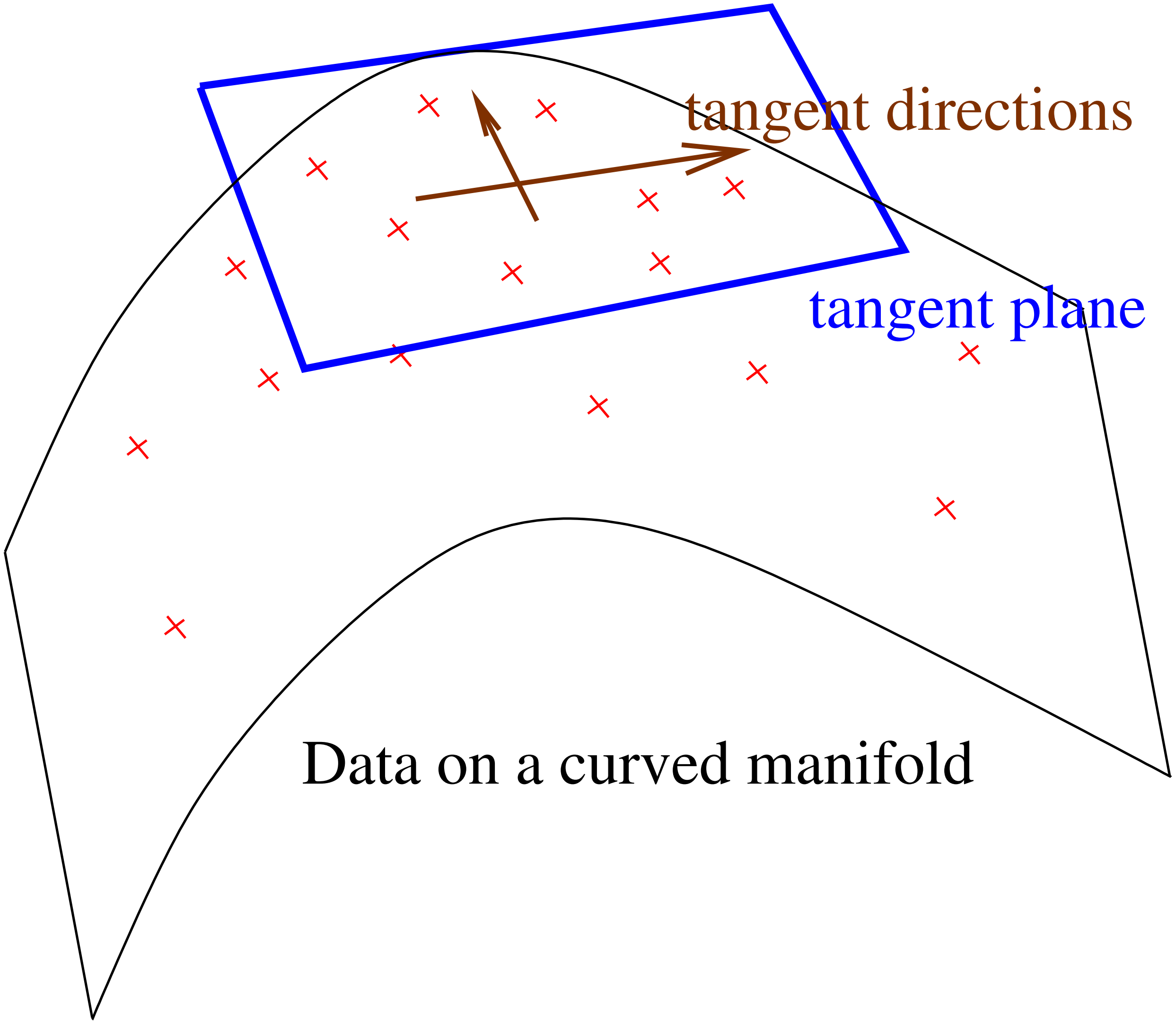


Auto-Encoders Learn Salient Variations, like a non-linear PCA



- Minimizing reconstruction error forces to keep variations along manifold.
- Regularizer wants to throw away all variations.
- With both: keep ONLY sensitivity to variations ON the manifold.

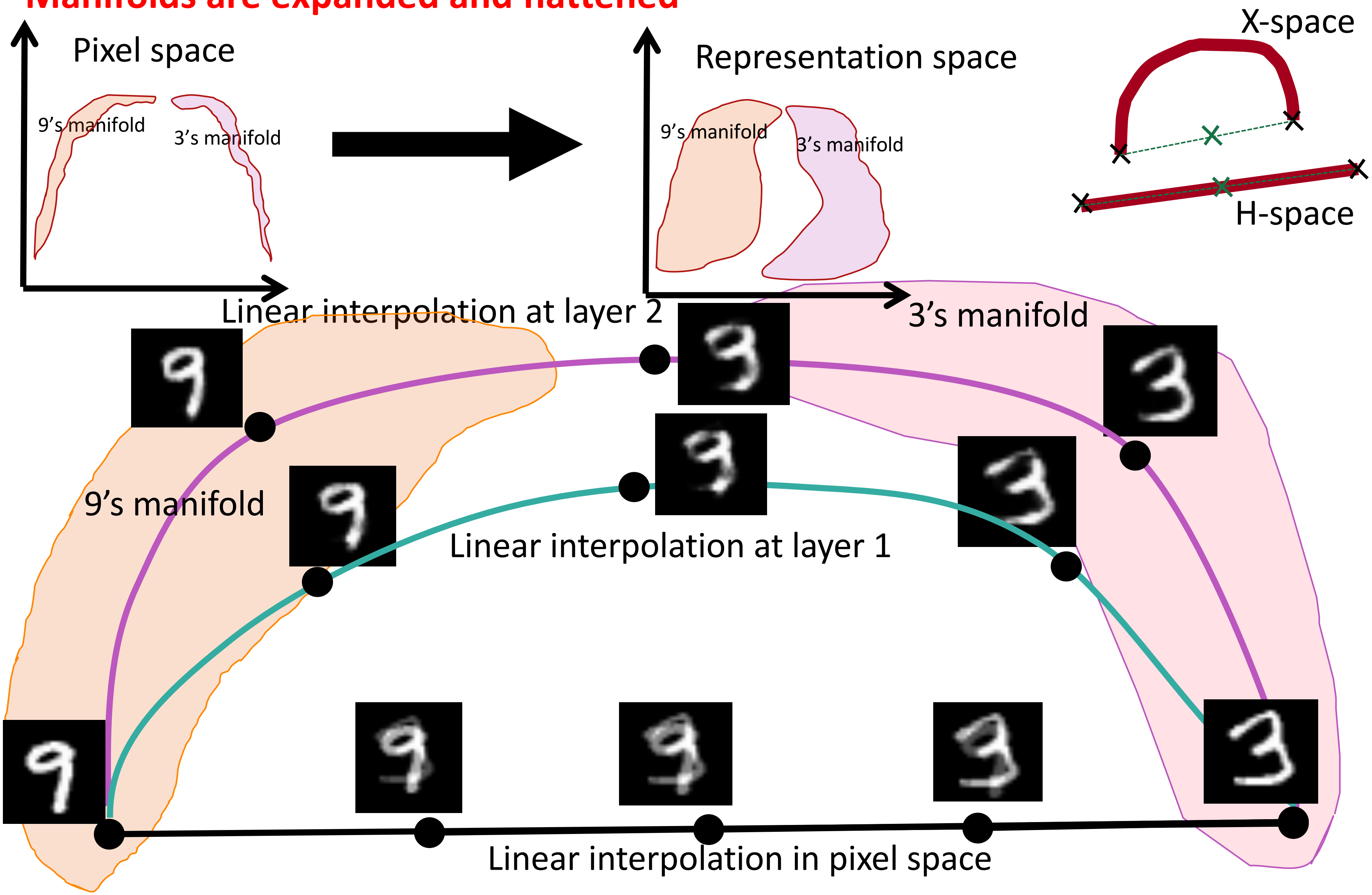
Manifold Learning = Representation Learning



Space-Filling in Representation-Space

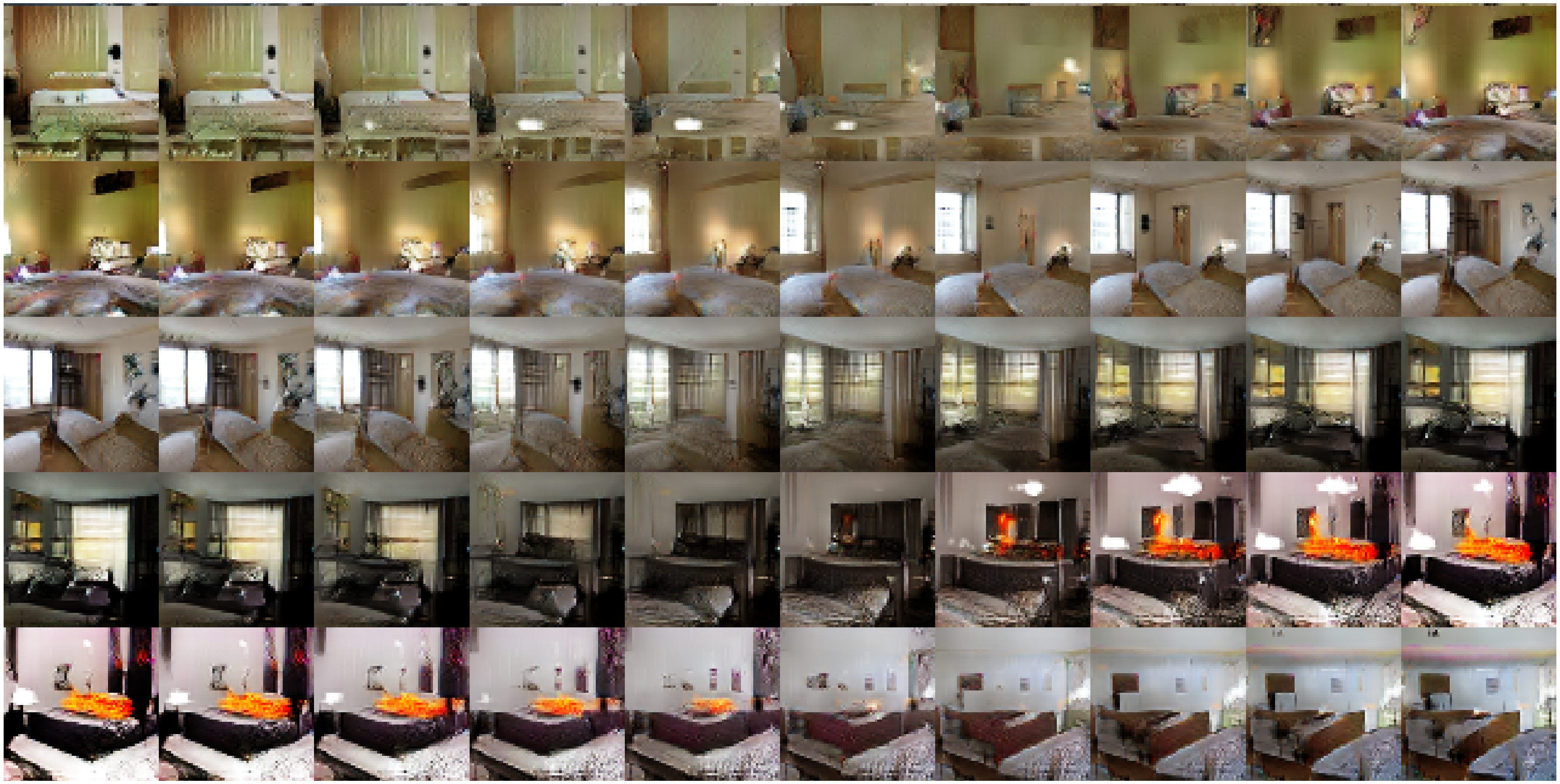
(Bengio et al ICML 2013)

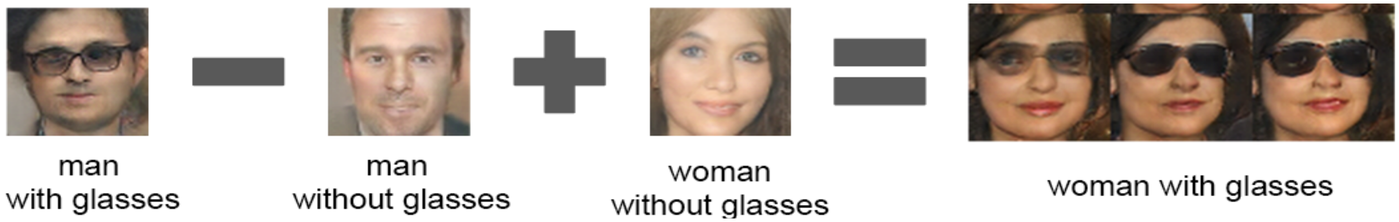
- **Deeper representations ➔ abstractions ➔ disentangling**
- **Manifolds are expanded and flattened**



Interpolating in Latent Space

If the model is good (unfolds the manifold), interpolating between latent values yields plausible images.





Radford et al 2016

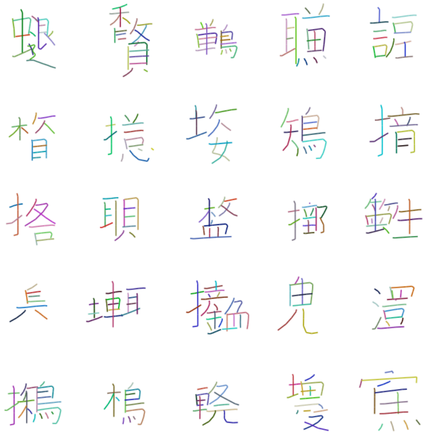
Deep Unsupervised Generative Models

Texture

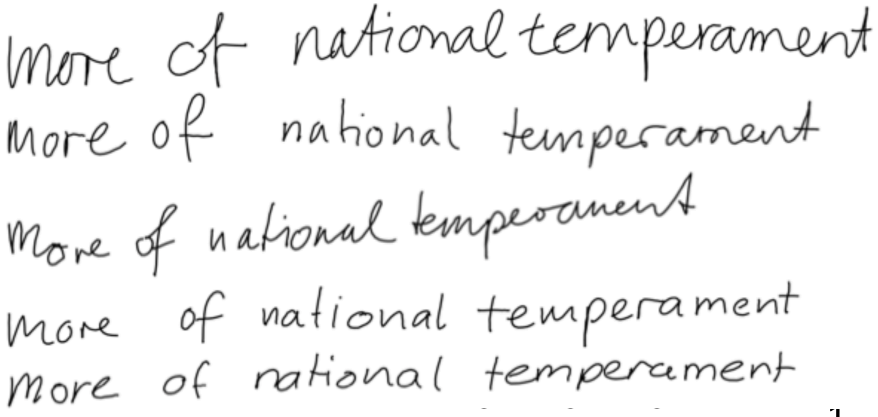
Shakespeare

Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

Chinese characters



Hand-writing


more of national temperament
more of national temperament
More of national temperament
more of national temperament
more of national temperament

Bedrooms

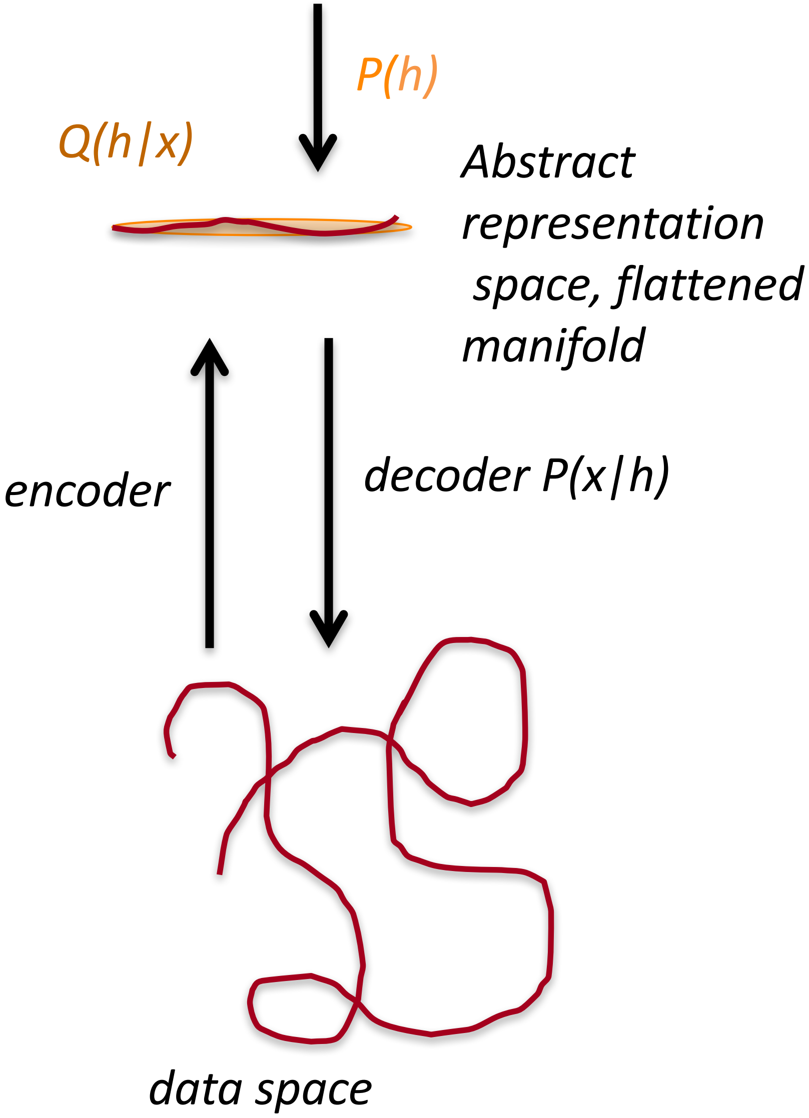
Latent Variables and Abstract Representations

- Encoder/decoder view: maps between low & high-levels

- Encoder does inference: interpret the data at the abstract level

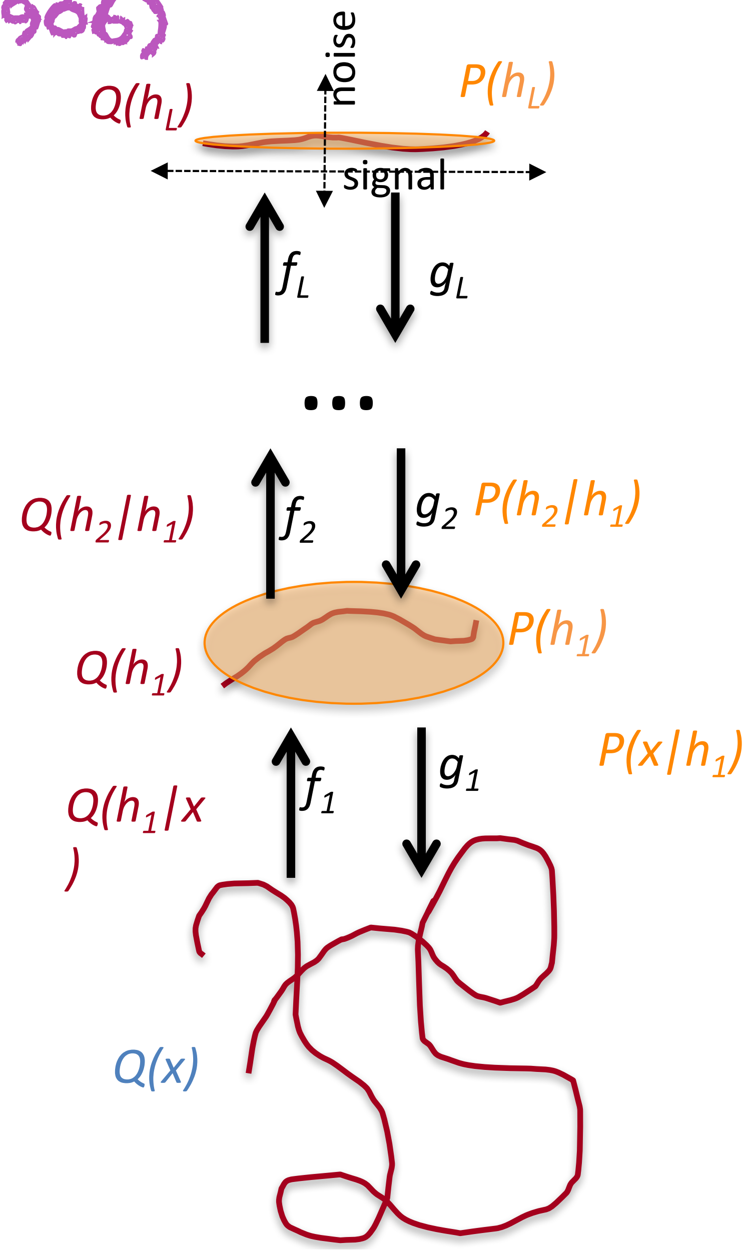
- Decoder can generate new configurations

- Encoder flattens and disentangles the data manifold



Extracting Structure By Gradual Disentangling and Manifold Unfolding (Bengio 2014, arXiv 1407.7906)

Each level transforms the data into a representation in which it is easier to model, unfolding it more, contracting the noise dimensions and mapping the signal dimensions to a factorized (uniform-like) distribution.

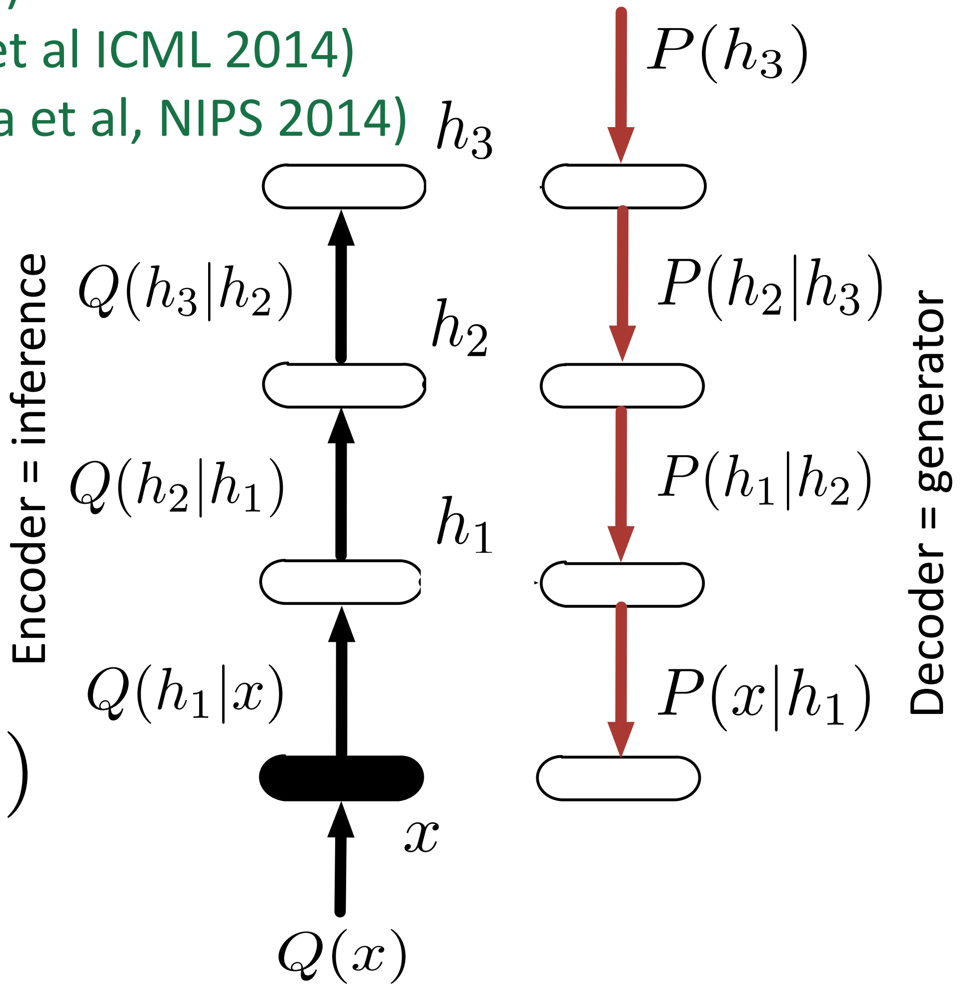


$$\min KL(Q(x,h)||P(x,h))$$

for each intermediate level h

Helmholtz Machines *(Hinton et al 1995)* and Variational Auto-Encoders (VAEs)

(Kingma & Welling 2013, ICLR 2014)
(Gregor et al ICML 2014; Rezende et al ICML 2014)
(Mnih & Gregor ICML 2014; Kingma et al, NIPS 2014)



- Parametric approximate inference
- Successors of Helmholtz machine *(Hinton et al '95)*
- Maximize variational lower bound on log-likelihood:

$$\min KL(Q(x,h)||P(x,h))$$

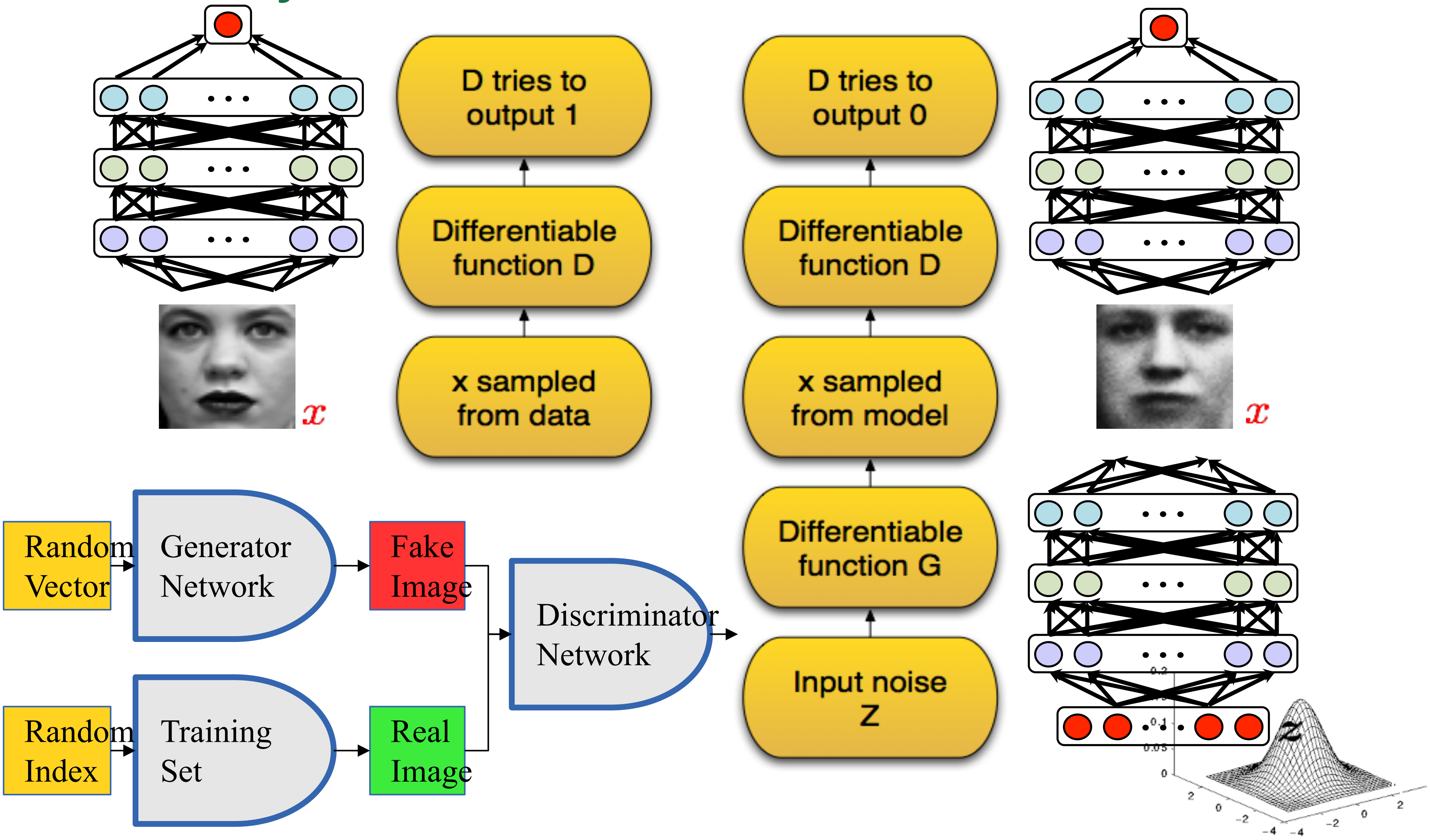
where $Q(x)$ = data distr.

or equivalently

$$\sum_{x,h} Q(x)Q(h|x)\log\frac{P(x,h)}{Q(h|x)} = \sum_{x,h} Q(x)Q(h|x)\log P(x|h) + KL(Q(h|x)||P(h))$$

GAN: Generative Adversarial Networks
A radical alternative to max. likelihood

Goodfellow et al NIPS 2014



D tries to output 1

Differentiable function D

x sampled from data

D tries to output 0

Differentiable function D

x sampled from model

Differentiable function G

Input noise Z

Random Vector

Generator Network

Fake Image

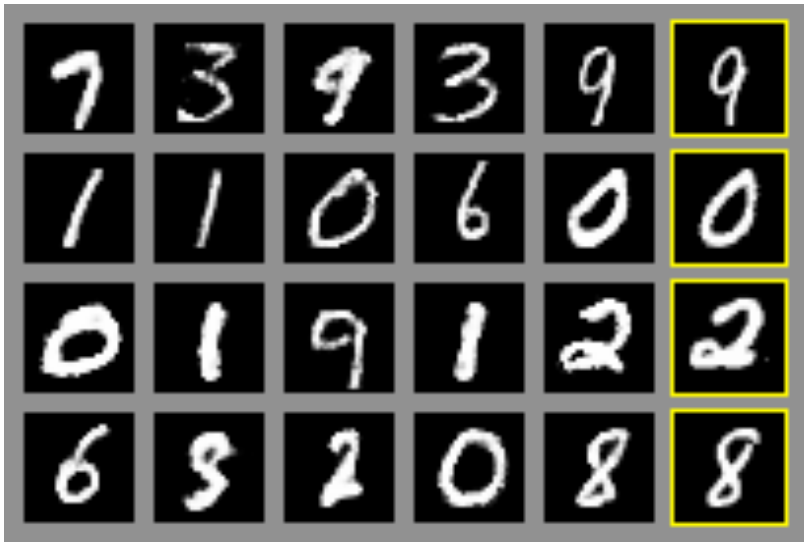
Random Index

Training Set

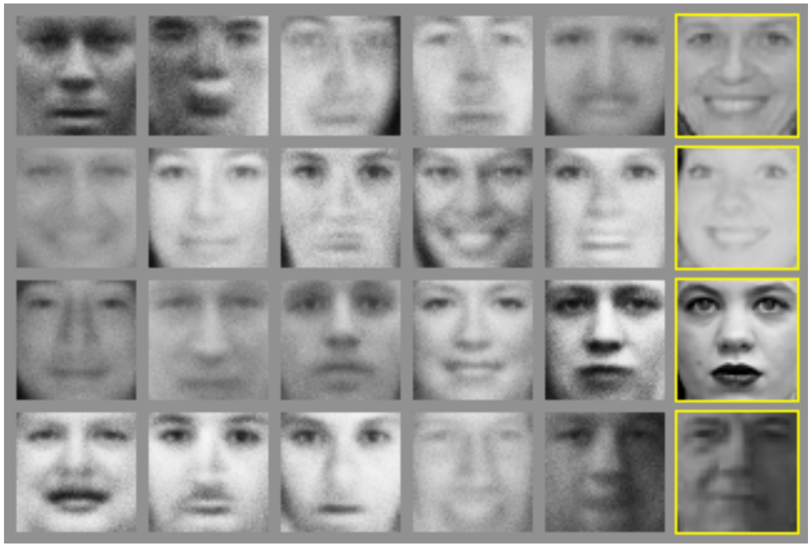
Real Image

Discriminator Network

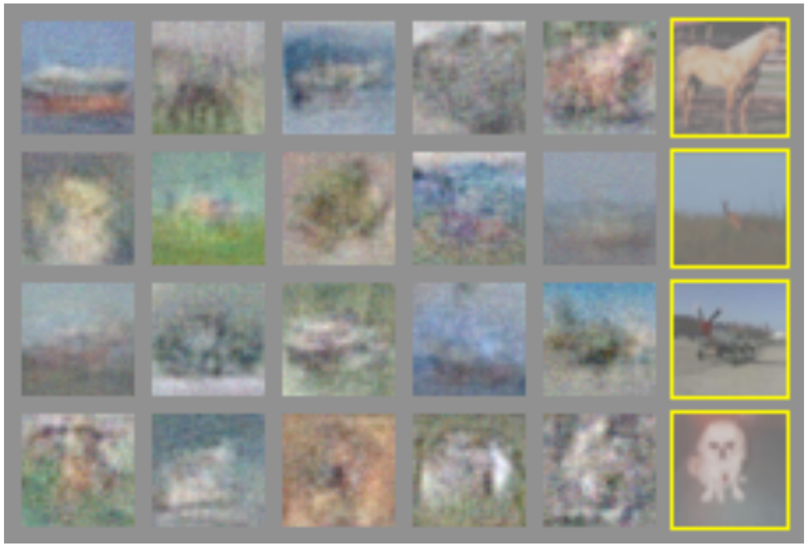
Early Days of GAN Samples



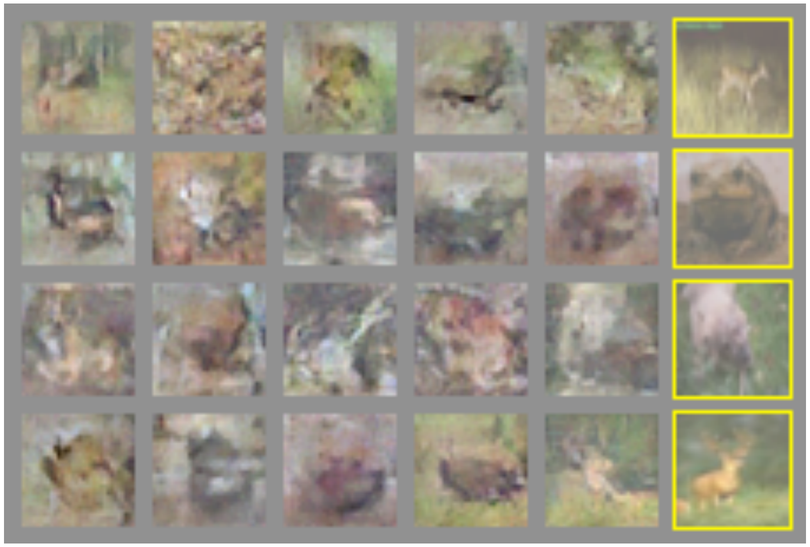
MNIST



TFD



CIFAR-10 (fully connected)

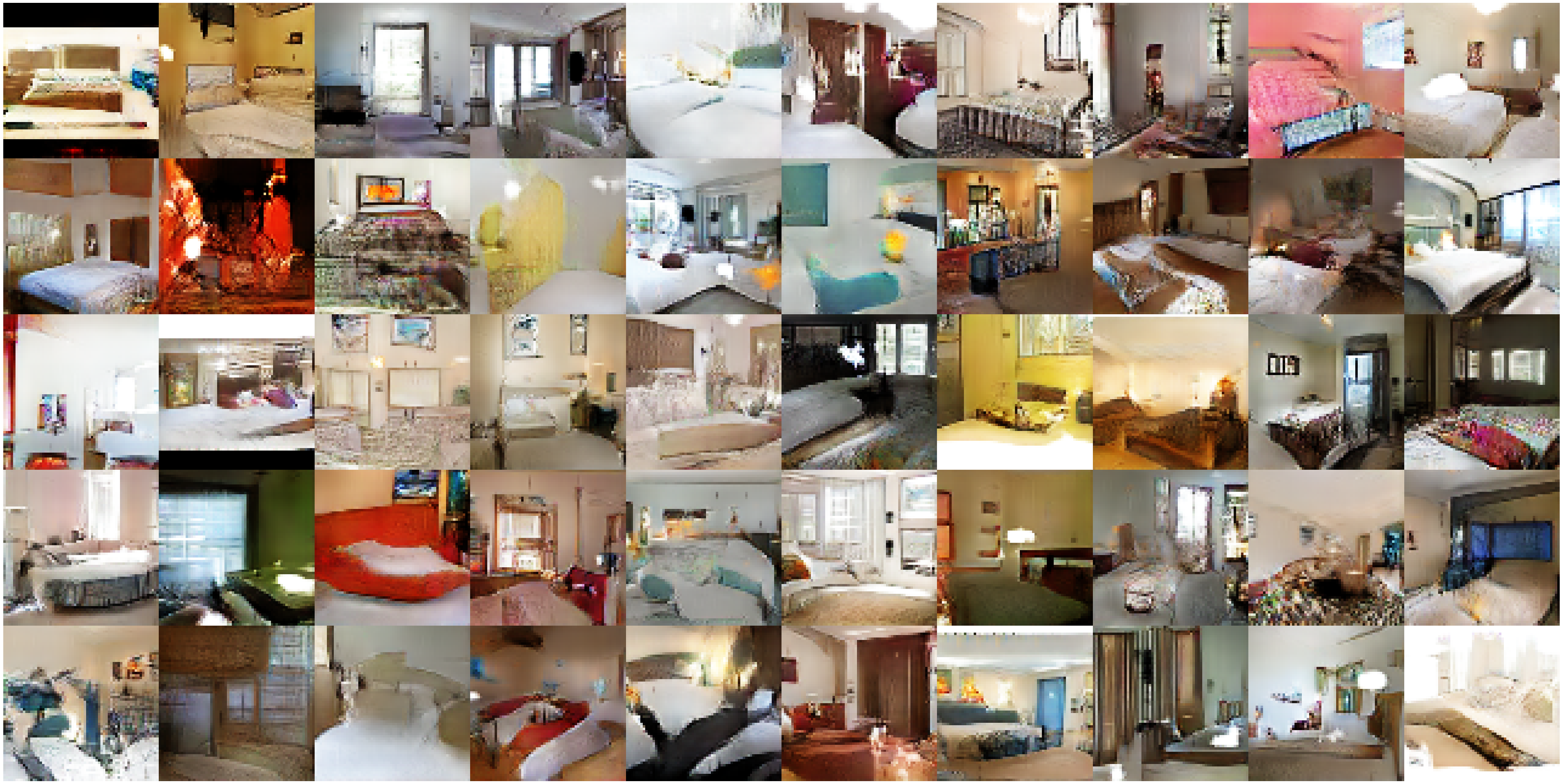


CIFAR-10 (convolutional)

Convolutional GANs

(Radford et al, arXiv 1511.06343)

Strided convolutions, batch normalization, only convolutional layers, ReLU and leaky ReLU



Generative Adversarial Networks

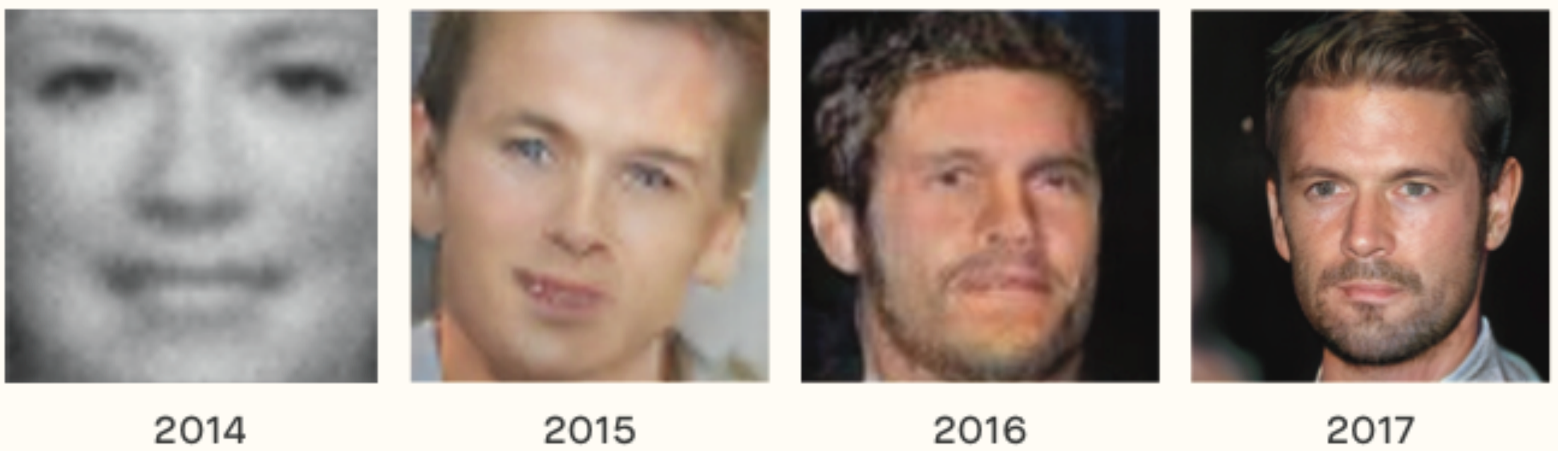
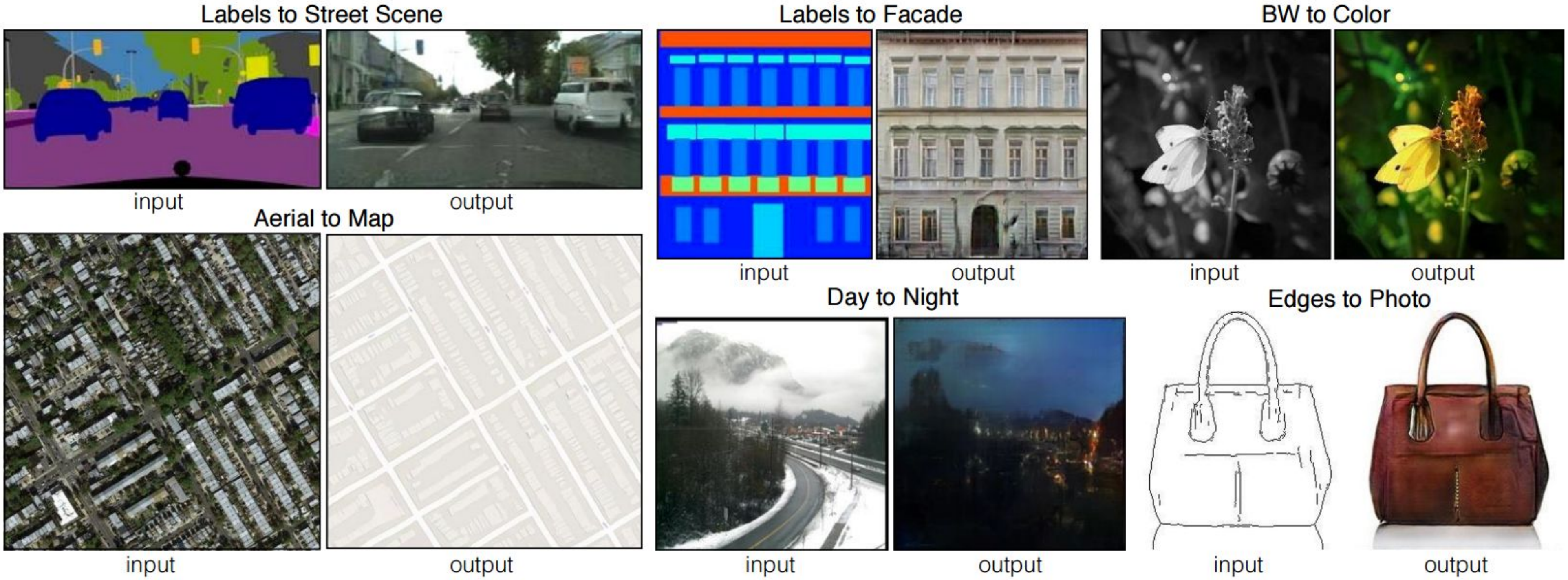


Image 2 Image



Isola et al. 2016

Text 2 Image, B&W 2 Color

This bird is red and brown in color, with a stubby beak

The bird is short and stubby with yellow on its body

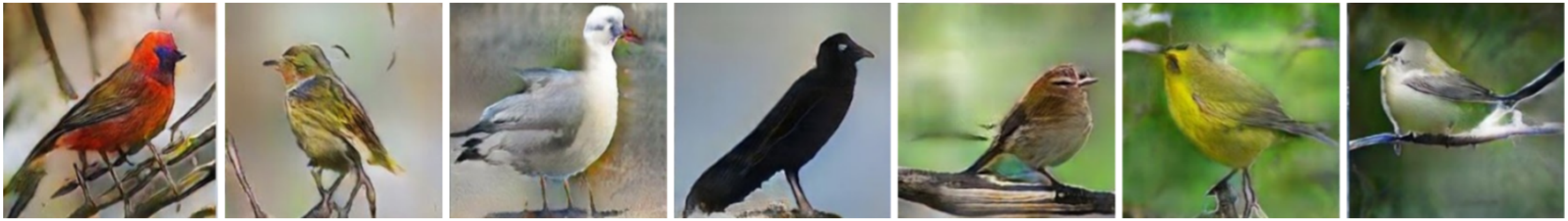
A bird with a medium orange bill white body gray wings and webbed feet

This small black bird has a short, slightly curved bill and long legs

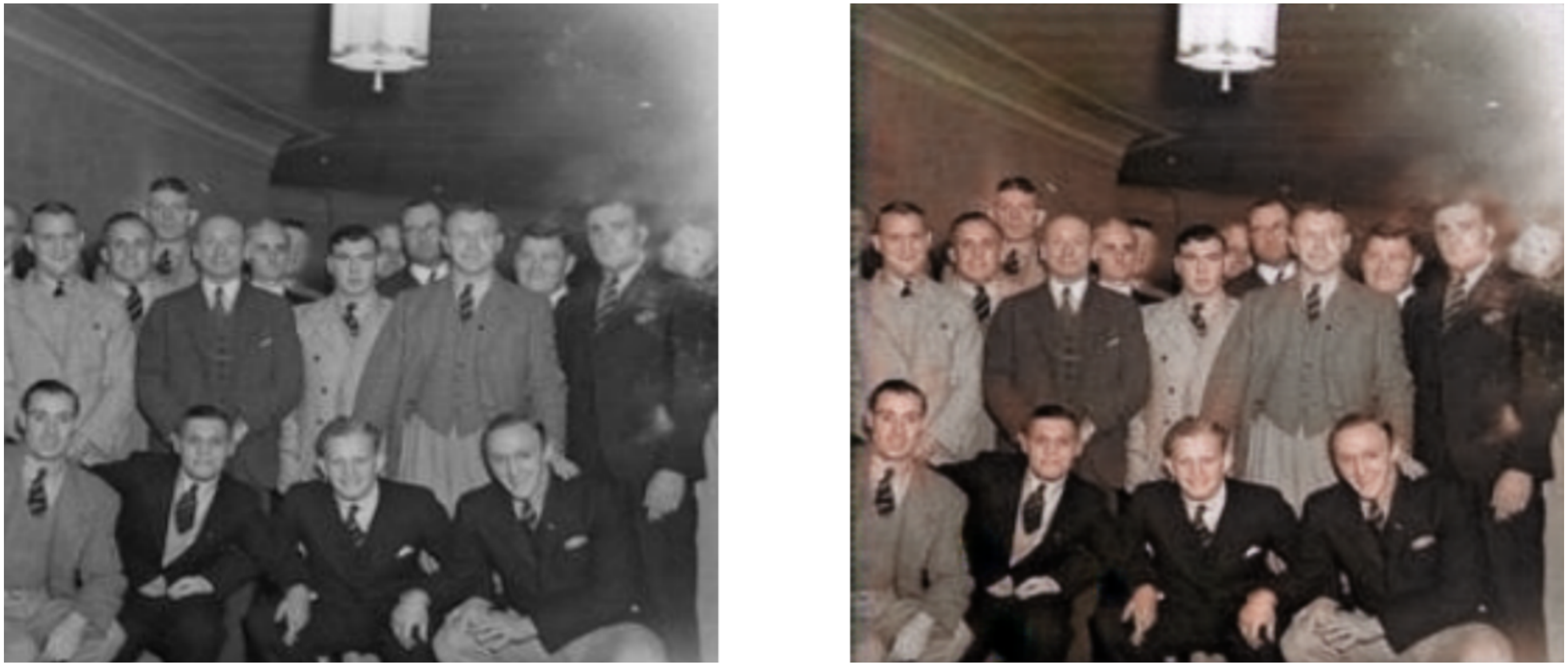
A small bird with varying shades of brown with white under the eyes

A small yellow bird with a black crown and a short black pointed beak

This small bird has a white breast, light grey head, and black wings and tail



Zhang et al. 2017

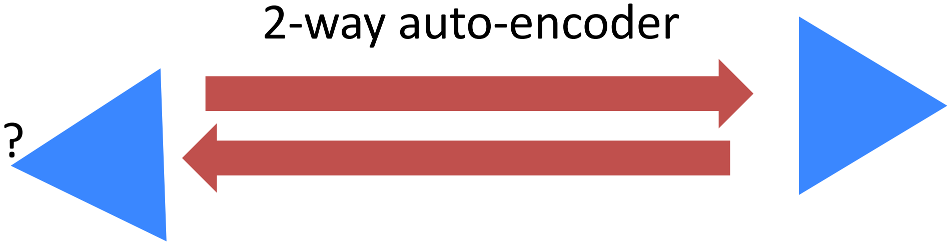


Lucy Li

Horse 2 Zebra: matching 2 domains by analogy of their distribution structure

Input video

Output video



2-way auto-encoder

Looks like a horse?

Looks like a zebra?

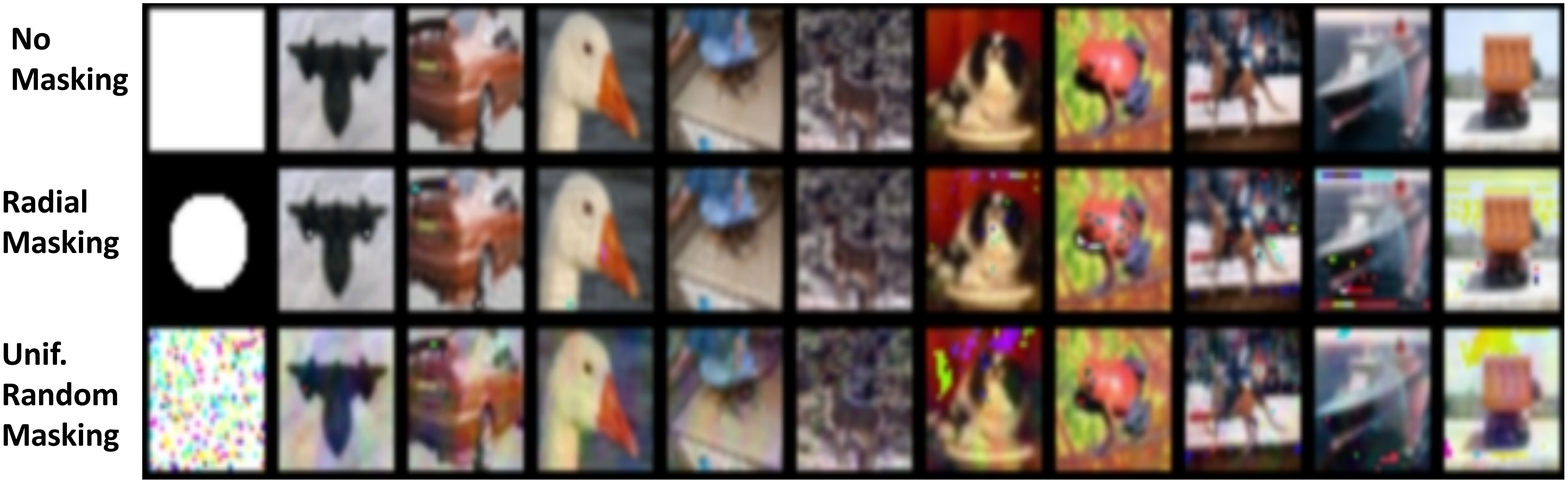
CycleGANs: Zhu et al. 2017

Measuring the Tendency of CNNs to Learn Surface Statistical Regularities

Jason Jo and Yoshua Bengio 2017, arXiv:1711.11561

- **Hypothesis**: *Deep CNNs have a tendency to learn superficial statistical regularities in the dataset rather than high level abstract concepts*.

- From the perspective of learning high level abstractions, Fourier image statistics can be *superficial* regularities, not changing object category



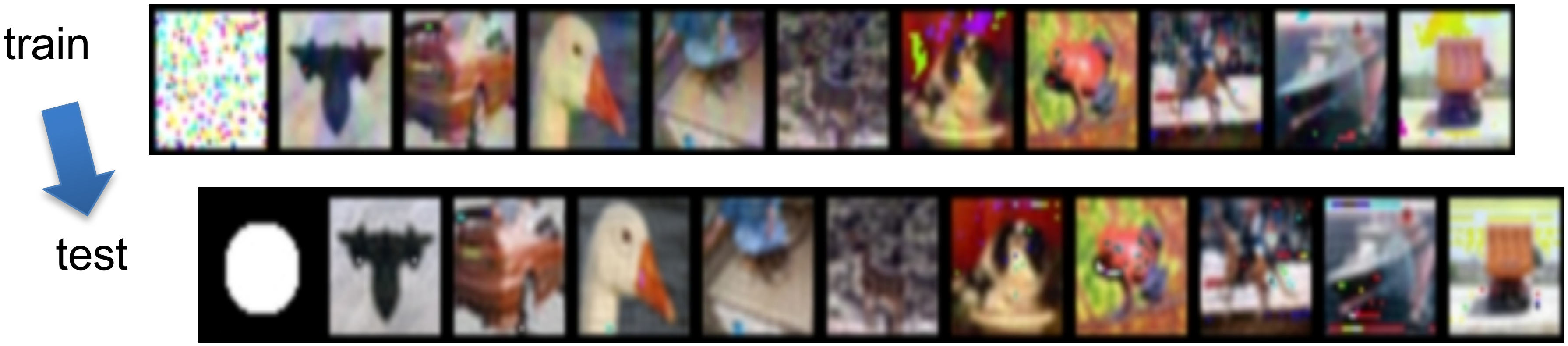
Fourier Maske d CIFAR- 10 Images

Measuring the Tendency of CNNs to Learn Surface Statistical Regularities

Jason Jo and Yoshua Bengio 2017, arXiv:1711.11561

- Different Fourier filters, same high level abstractions (objects) but different surface statistical regularities (Fourier image statistics).

- Experiment: Train on one training set and evaluate the test sets.
- A generalization gap: max difference in test accuracies



train

test

- Large generalization gap: CNN exploits too much of low level regularities, as opposed to learning the abstract high level concepts.

Rare & Dangerous States

- Example: autonomous vehicles in near-accident situations
- Current supervised learning may not handle well these cases because they are too rare (not enough data)

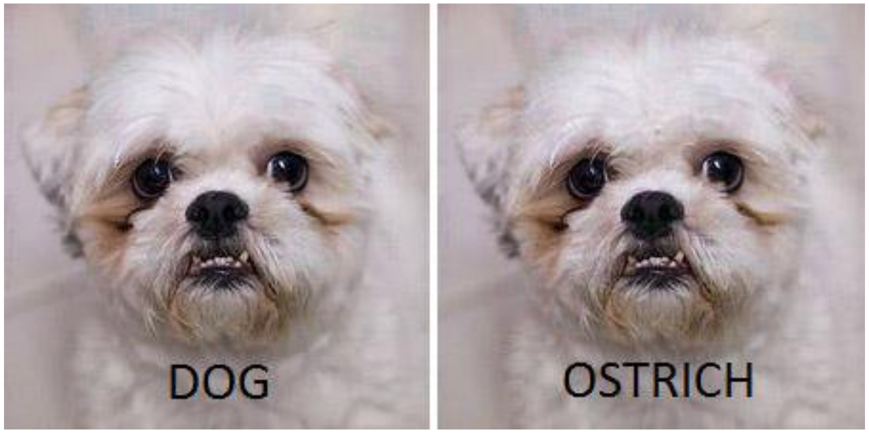
- It would be even worse with current RL (statistical inefficiency)
- Long-term objective: develop better predictive models of the world able to generalize in completely unseen scenarios
- Example of similar human ability: figuring out intuitive physics, no need to die a thousand deaths

What’s Missing with Deep Learning?

Deep Understanding

Still Far from Human-Level AI

- Industrial successes mostly based on **supervised** learning

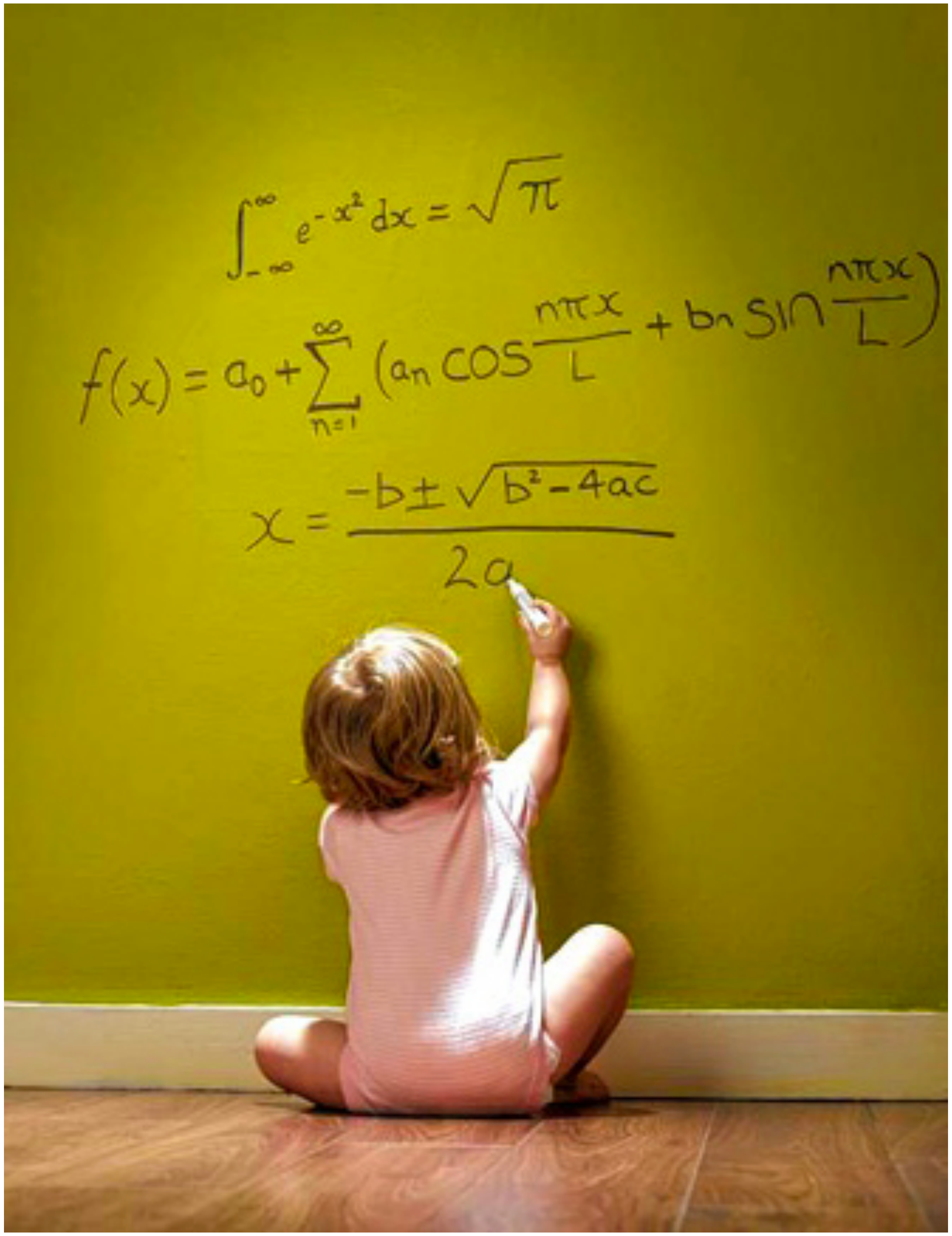


- Learning superficial clues, not generalizing well outside of training contexts, easy to fool trained networks:
  - Current models cheat by picking on surface regularities
- Still unable to discover higher-level abstractions

Humans outperform machines at unsupervised learning

- Humans are very good at unsupervised learning, e.g. a 2 year old knows intuitive physics

- Babies construct an approximate but sufficiently reliable model of physics, how do they manage that? Note that they interact with the world, not just observe it.


$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$
$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$
$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Learning « How the world ticks »

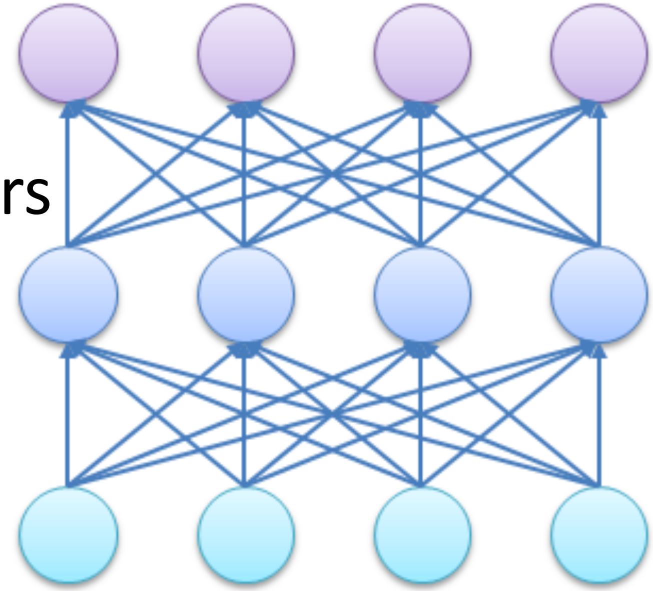
- So long as our machine learning models « cheat » by relying only on superficial statistical regularities, they remain vulnerable to out-of-distribution examples

- Humans generalize better than other animals thanks to a more accurate internal model of the **underlying causal relationships**

- To predict future situations (e.g., the effect of planned actions) far from anything seen before while involving known concepts, an essential component of reasoning, intelligence and science

How to Discover Good Disentangled Representations

- How to discover abstractions?
- What is a good representation? ***(Bengio et al 2013)***
- Need clues (= priors) to help **disentangle** the underlying factors, such as
  - Spatial & temporal scales
  - Marginal independence
  - Simple dependencies between factors
    - *Consciousness prior*
  - Causal / mechanism independence
    - *Controllable factors*



Agent-Based Learning (aka RL)

Acting to Guide Representation Learning & Disentangling

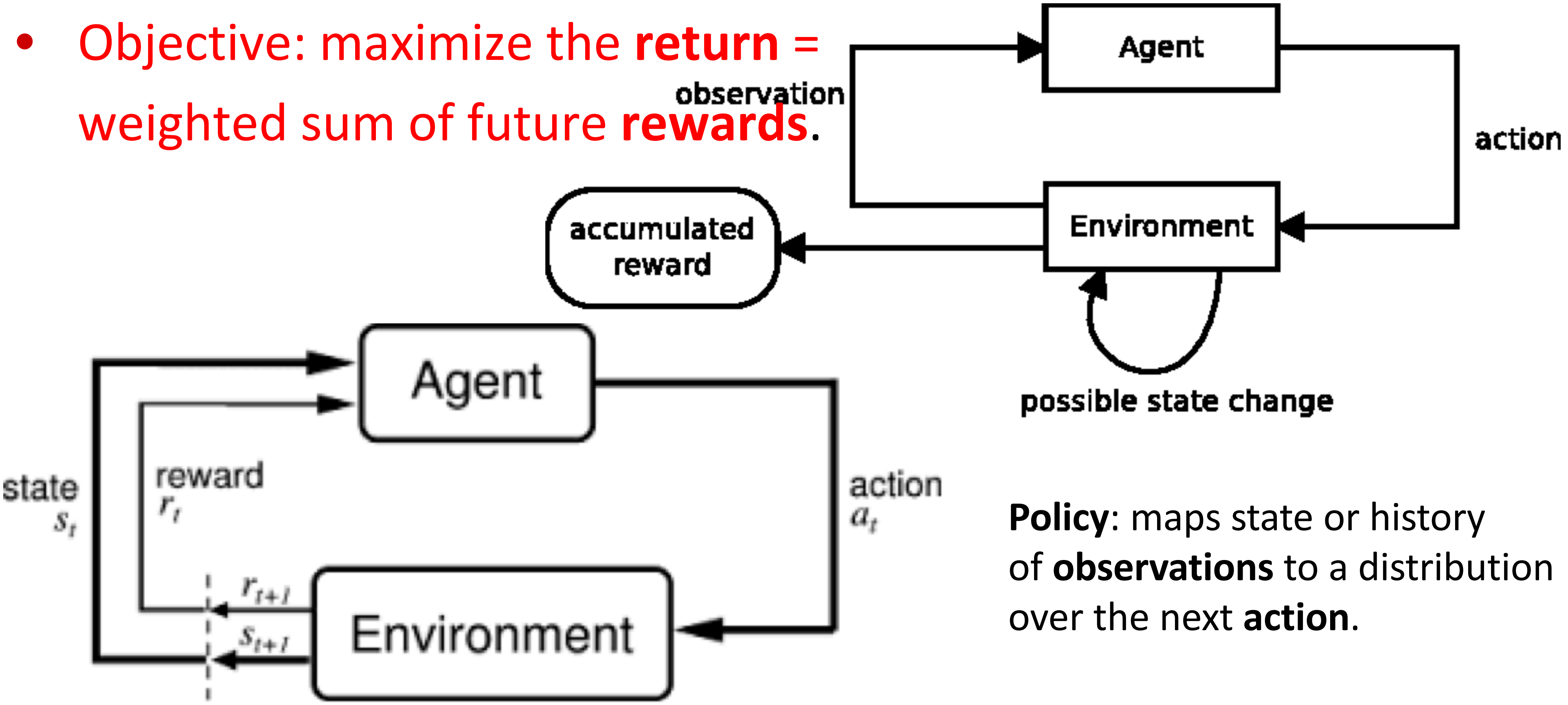
(E. Bengio et al, 2017; V. Thomas et al, 2017)

- **Some factors (e.g. objects) correspond to 'independently controllable' aspects of the world**

- *Can only be discovered by acting in the world*
  - *Control linked to notion of objects & agents*
  - *Causal but agent-specific & subjective: affordances*

Reinforcement Learning

- In general the full **state** of the environment is not observed, leading to the **partially observable** setting. When it is fully observed we have a Markov decision process.
- Objective: maximize the **return** = weighted sum of future **rewards**.



Policy: maps state or history of **observations** to a distribution over the next **action**.

Model-free vs Model-based RL

- Model-free: directly learn a policy or a **value function** (which associates a state or a state-action pair Q with an estimated return), trying to maximize returns.
  - Policy-gradient methods: estimate the stochastic gradient of the expected return wrt the policy itself, to update it.
- Model-based:
  - Unsupervisedly learn to model the environment (state transition, rewards)
  - Use planning (approximate search/optimization) to choose actions
- Dyna: combine both → internal simulations from estimated model trains a policy

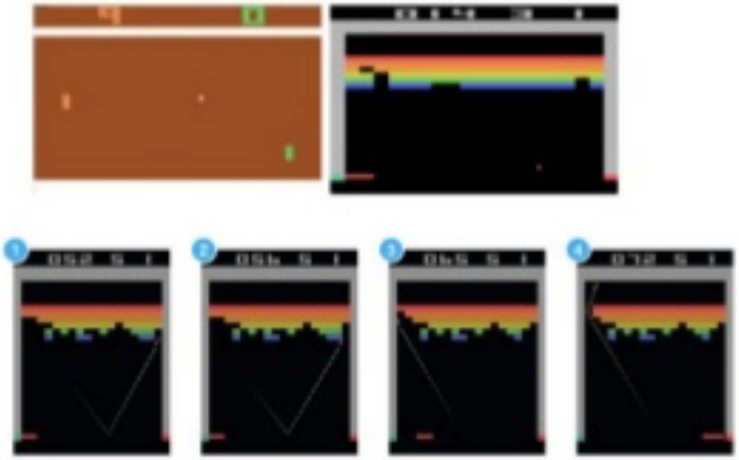
Deep Reinforcement Learning

- Map state or observation sequence to a learned representation to better generalize to new states
- Use neural nets to learn policy, value function, Q-function, estimated reward function, estimated transition operator, etc.
- Share representation across different networks
- Use offline training or replay buffer (memory of past state-action-nextstate-reward tuples) to avoid catastrophic forgetting
- Task rewards are like sparse supervision, use intrinsic rewards (e.g. curiosity, discovery) as dense unsupervised objectives

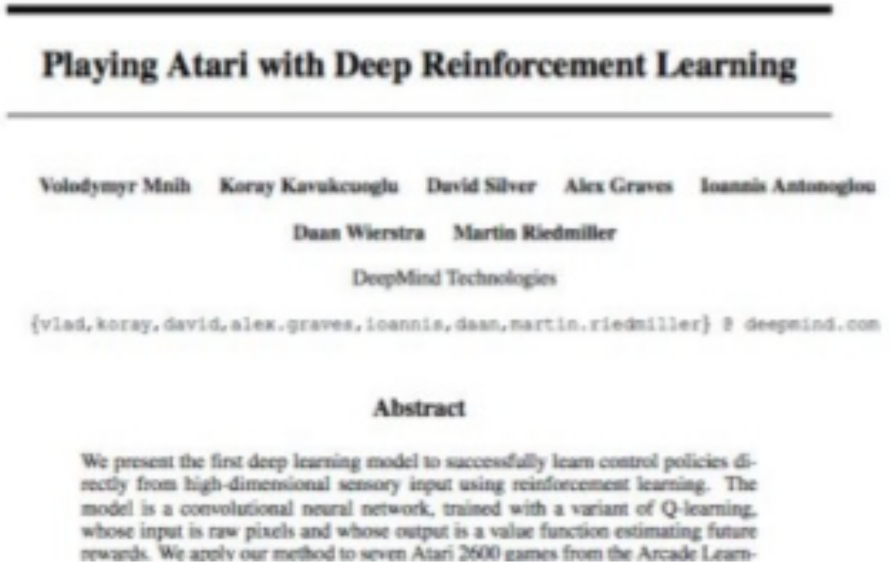
Playing all 50 Atari games @ DeepMind

Simulator from U. Alberta's Sutton's group. First DRL breakthrough.

2013: Deep RL



http://arxiv.org/abs/1312.5602



Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller
DeepMind Technologies
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learn-

March 2016: World Go Champion Beaten by Machine

Coming Deep Learning Revolution in Robotics (& Mobile Robotics)

Groups of Pieter Abbeel & Sergey Levine @ Berkeley

The Deep Learning way of training autonomous agents

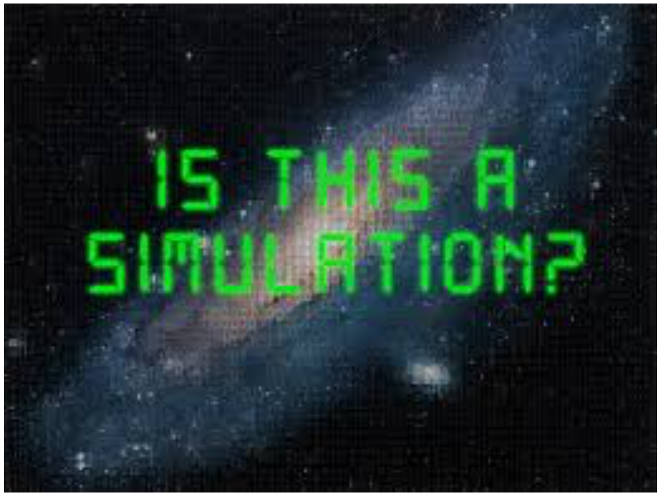
- Distributed representations everywhere
- Shared representations across all forms of predictions (value, policy, rewards, transitions)
- Learn to represent goals (intentions), subgoals, policies (skills), manipulate distributions over them and share representations
- Model the future and plan in latent (representation) space
- Partially observed setting + recurrence to estimate state internally
- Use an associative memory to handle short and long-term memory and associate events across long time spans
- Use attention to focus on a few aspects of the world at each step of a high-level plan

What’s Missing

- More autonomous learning, better **unsupervised learning**
- Discovering the **underlying causal factors**
- Model-based RL which extends to completely new situations by **unrolling powerful predictive models which can help reason about rarely observed dangerous states**
- **Deep learning to expand from perception & system 1 cognition to reasoning & system 2 cognition**

Current Model-Free RL is too Statistically Inefficient: Combine Model-Based and Model-Free RL

- Simulate possible futures (given current state and actions) in order to train the policy (which can act quickly, without having to perform expensive planning)
- Need a good generative model of how agents cause changes in the world (effects)
- Better to generate future abstract states rather than future perceptions



Causality

Deep Learning Objective: discover causal representation

- What are the right representations? Causal variables explaining the data
- How to discover them?
- How to discover their causal relationship, the causal graph?

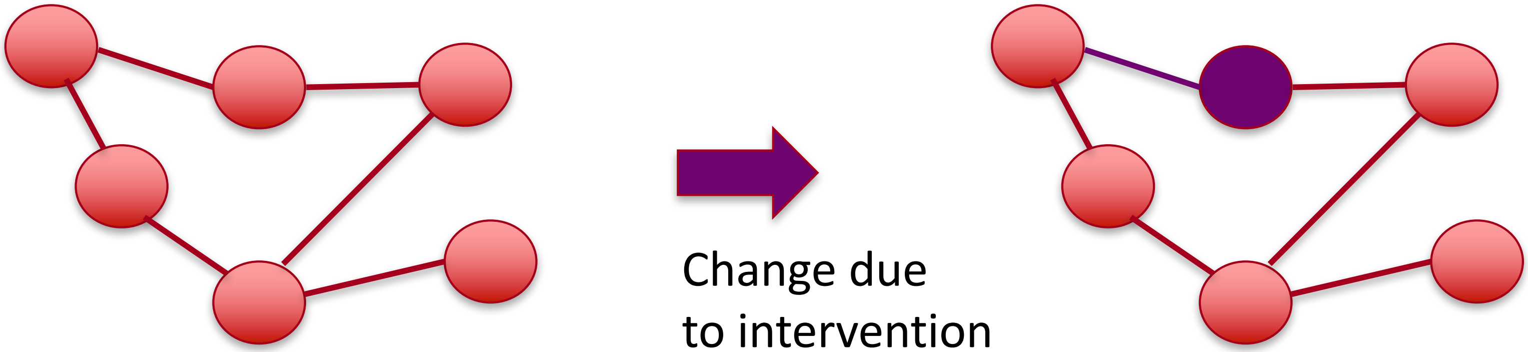
Disentangling: Factoring out aspects of the acquired knowledge

- How to disentangle the unobserved explanatory variables?
- How to separate the dependencies between these variables into separate easily re-usable pieces?
- How to modularize procedural knowledge into easily re-usable pieces? (options etc)
- How to modularize knowledge for easier re-use & adaptation, good transfer?

Mila

Separating Knowledge in Small Pieces

- Pieces which can be re-used combinatorially
- Pieces which are stable vs nonstationary, subject to interventions



Change due to intervention

Missing from Current ML: Understanding & Generalization Beyond the Training Distribution

- Learning theory only deals with generalization within the same distribution
- Models learn but do not generalize well (or have high sample complexity when adapting) to modified distributions, non-stationarities, etc.
- Poor reuse, poor modularization of knowledge

Beyond iid: Hypotheses about how the environment changes
Independent Mechanisms and the Small Change Hypothesis

- Independent mechanisms:
  - changing one mechanism does not change the others *(Peters, Janzig & Scholkopf 2017)*
- Small change:
  - Non-stationarities, changes in distribution, involve few mechanisms (e.g. the result of a single-variable intervention)

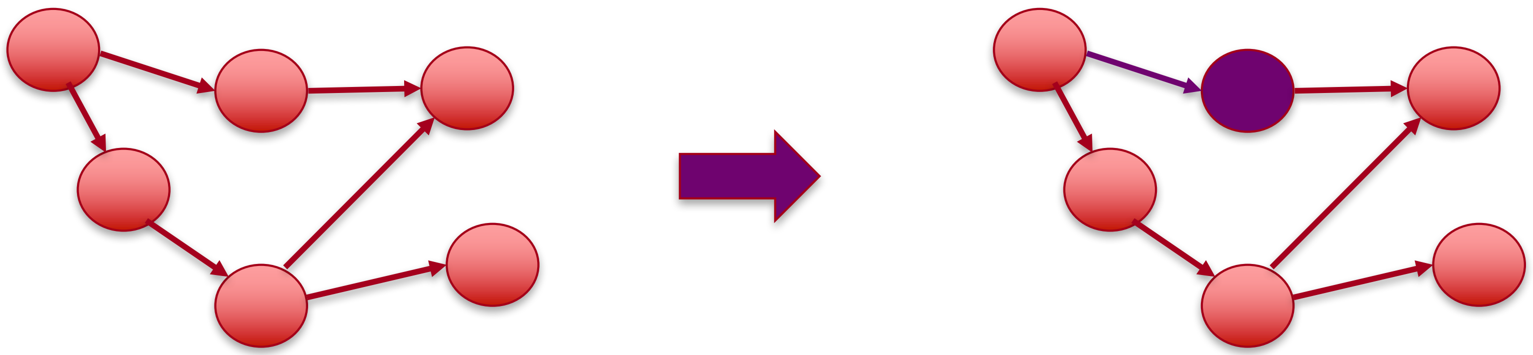
What if we had the right modular structure?

CLAIM: Under the hypothesis of independent mechanisms and small changes across different distributions:

- **smaller sample complexity to recover from a distribution change**
    - E.g. for transfer learning, agent learning, domain adaptation, etc.

Small Change in the Right Space

Distribution change: only one or a few mechanisms change



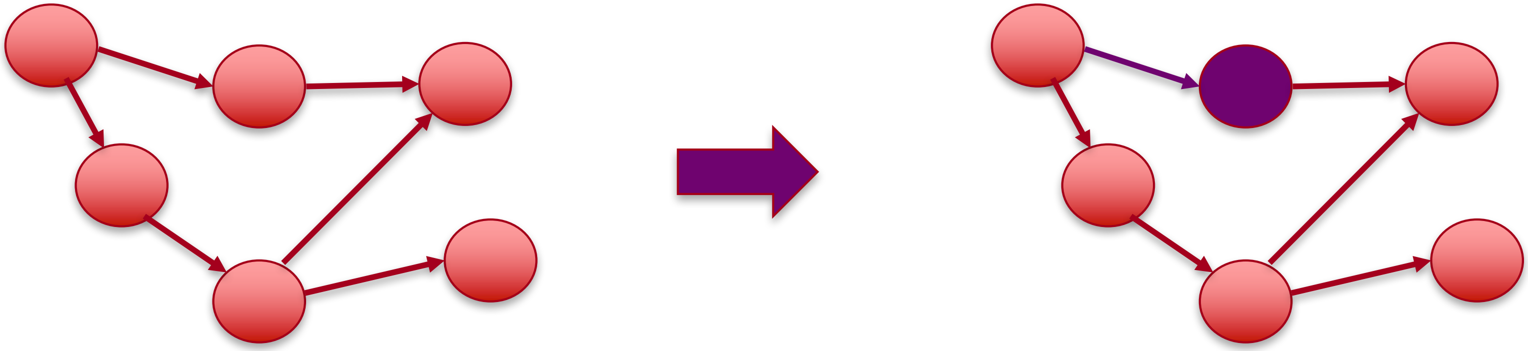
Before: eyes open

After: eyes closed,
totally different in pixel space,
small change in object space

Under the right parametrization, few parameters need to change after an intervention

Small Change → Small Sample Complexity

Few parameters need to change → small L2 change → few examples needed to recover from the change



Under the right parametrization → fast adaptation to interventions

Mila

Current Causal Team @ Mila

ArXiv paper: *A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms*

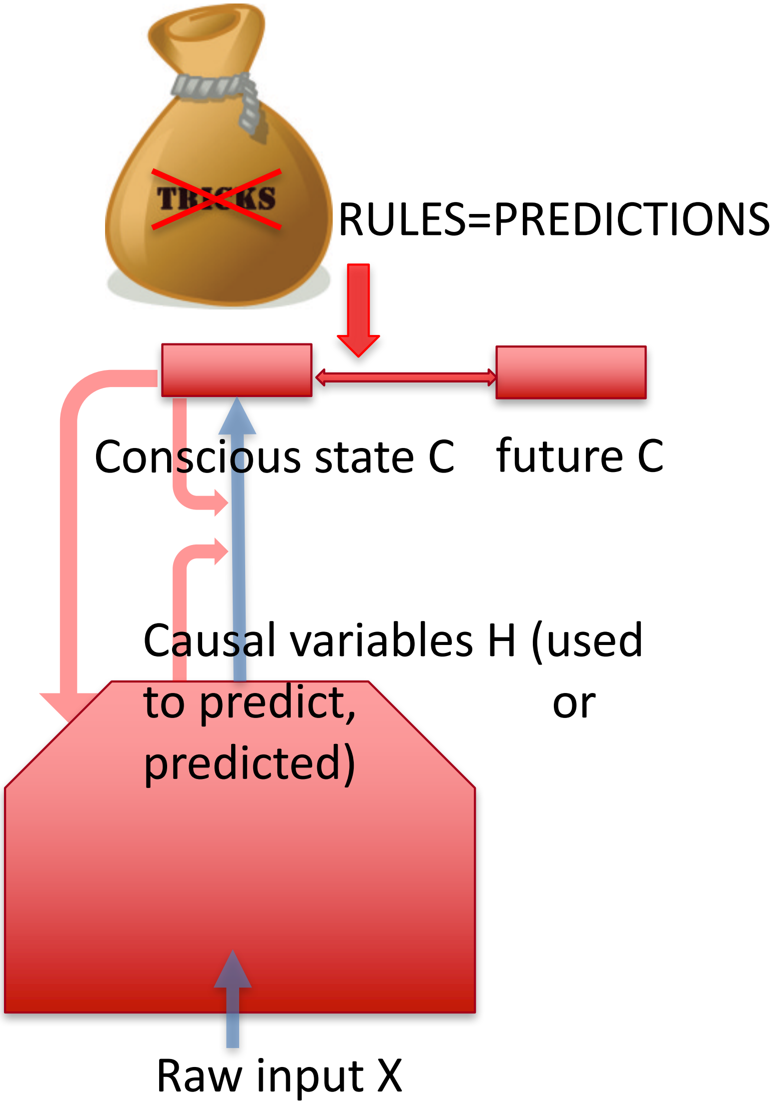
- Yoshua Beng
- Tristan Del
- Nasim Rahaman
- Rosemary Ke
- Olexa Bilaniuk
- Anirudh Goyal
- Sébastien Lachapelle
- Chris Pal
- Rémi Le Priol
- Simon Lacoste-Julien

Bigger Picture

  - Encoder maps sensory data to space where a few **sparse predictive rules** relate causal variables together, following the ***consciousness prior (Bengio 2017)***
  - ***Best graphical model assumption:***

    sparse factor graph

- Reasoning: sequentially focussing on a few entities (objects) and relations (rules) linked via causal links



RULES=PREDICTIONS

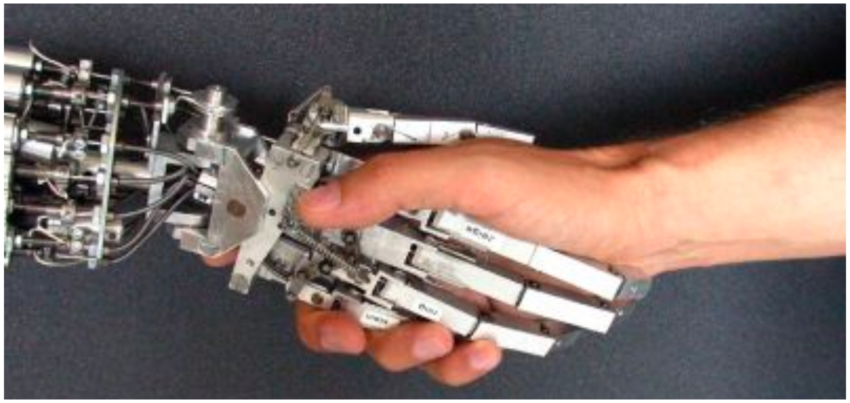
Conscious state C

future C

Causal variables H (used to predict, or predicted)

Raw input X

The Future of Deep AI



- Scientific progress is slow and continuous, but social and economic impact can be disruptive
- Many fundamental research questions are in front of us, with much uncertainty about when we will crack them, but we will
- Importance of continued investment in basic & exploratory AI research, for both practical (recruitment) short-term and long-term reasons
- Let us continue to keep the field open and fluid, be mindful of social impacts, and make sure AI will bloom for the benefit of all