Mila

Deep Learning for AI

Yoshua Bengio

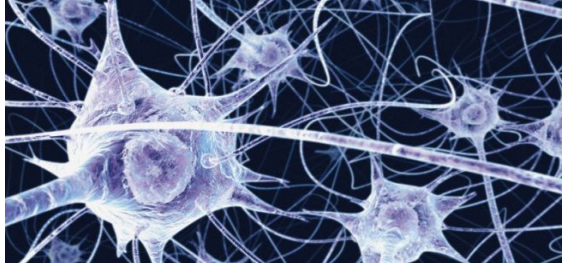
Université de Montréal

MACHINE LEARNING SUMMER SCHOOL
6 & 7 OF JULY 2020

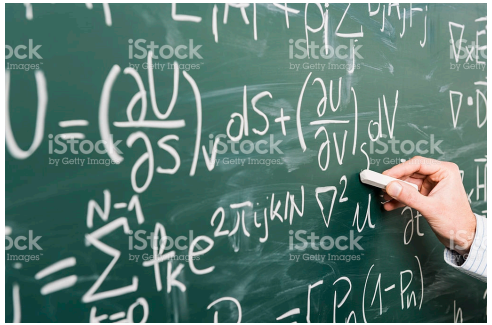
CIFAR
CANADIAN INSTITUTE FOR ADVANCED RESEARCH

ICRA
INSTITUT CANADIEN DE RECHERCHES AVANCÉES

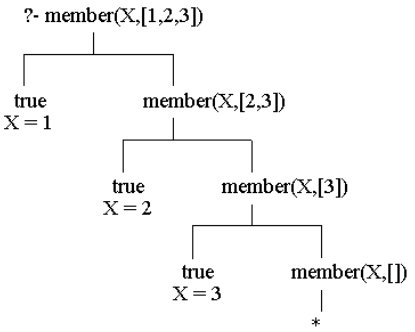
Neural Networks & AI: Underlying Assumption



- 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.

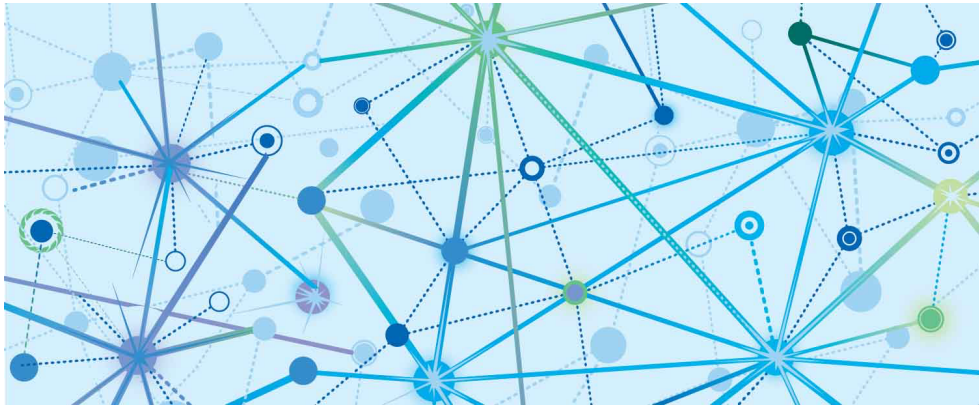


The Machine Learning approach to AI



- **Classical AI, rule-based, symbolic**
  - knowledge is provided by humans
    - but intuitive knowledge (e.g. much of common sense) not communicable
  - machines only do inference
  - no strong learning, adaptation
  - insufficient handling of uncertainty
  - not grounded in low-level perception and action
- **Machine learning tries to fix these problems**
  - succeeded to a great extent
  - higher-level (conscious) cognition not achieved yet

The Neural Net Approach to AI

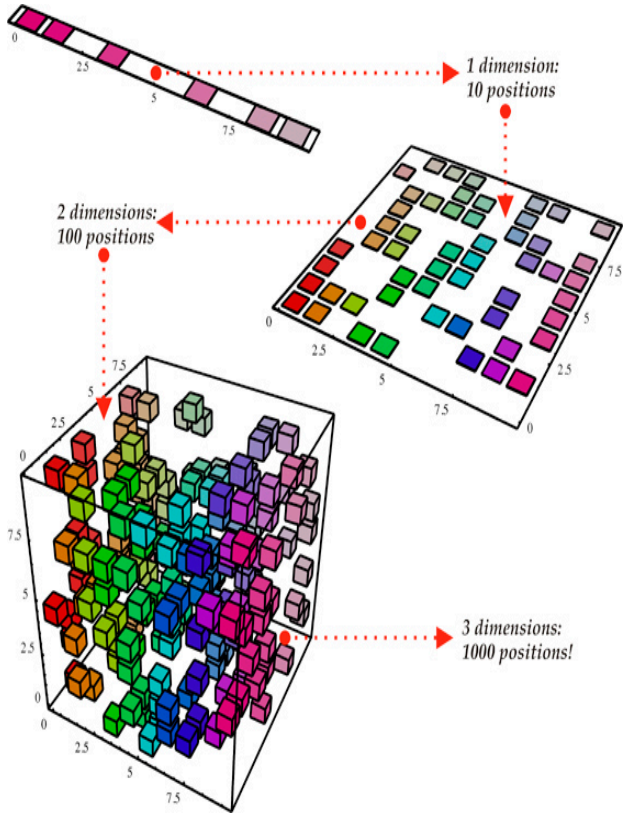


- **Brain-inspired**
- Synergy of a large number of simple adaptive computational units
- Focus on **distributed representations**
  - **E.g. word representations** (Bengio et al NIPS'2000)
- View intelligence as arising of combining
  - an objective or reward function
  - an approximate optimizer (learning rule)
  - an initial architecture / parametrization
- End-to-end learning (all the pieces of the puzzle adapt to help each other)

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



Learning Representations

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

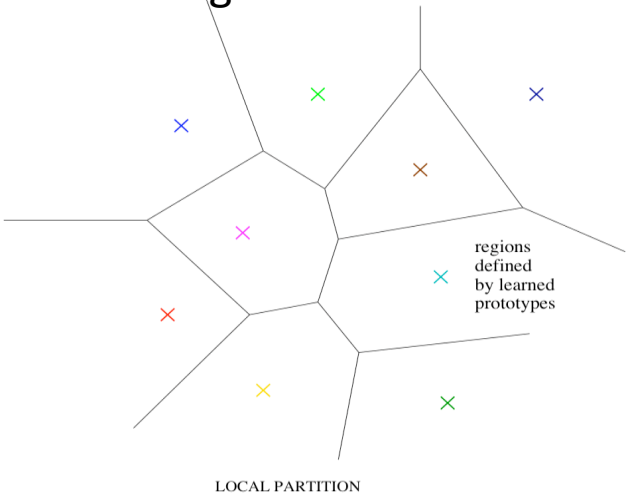
Exploiting compositionality can give an **exponential** gain in representational power

- Distributed representations / embeddings: feature learning
- Deep architecture: multiple levels of feature learning

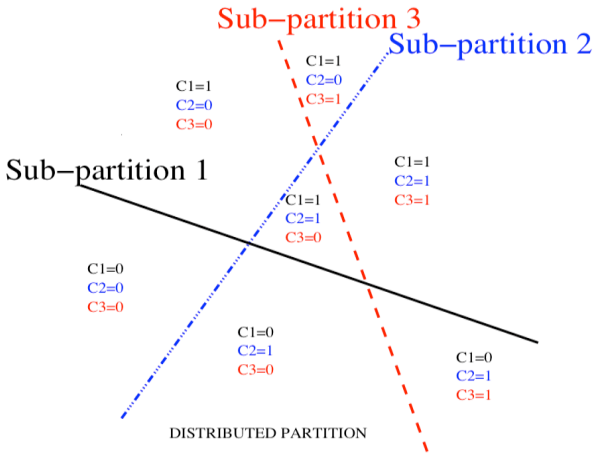
Prior assumption: compositionality is useful to describe the world around us efficiently

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



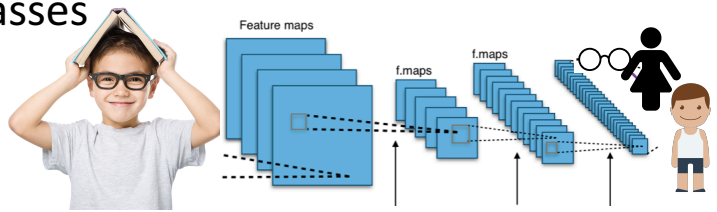
Not Distributed



Distributed

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

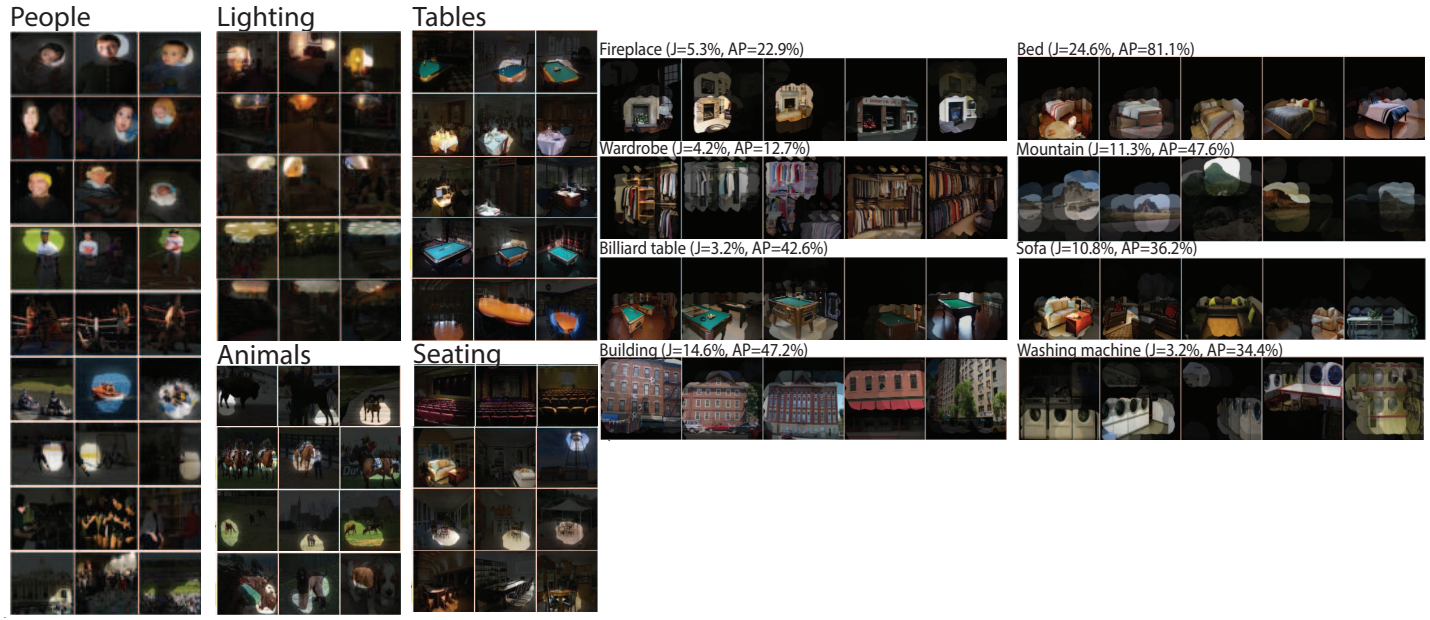
- Consider a network whose hidden units discover the following features:
  - Person wears glasses
  - Person is female
  - Person is a child
  - Etc.



- If each of *n* feature requires *O(k)* parameters, need *O(nk)* examples
- Parallel composition of features: can be exponentially advantageous
- Non-distributed non-parametric methods would require $O(n^d)$ examples

Hidden Units Can Discover Semantically Meaningful Concepts

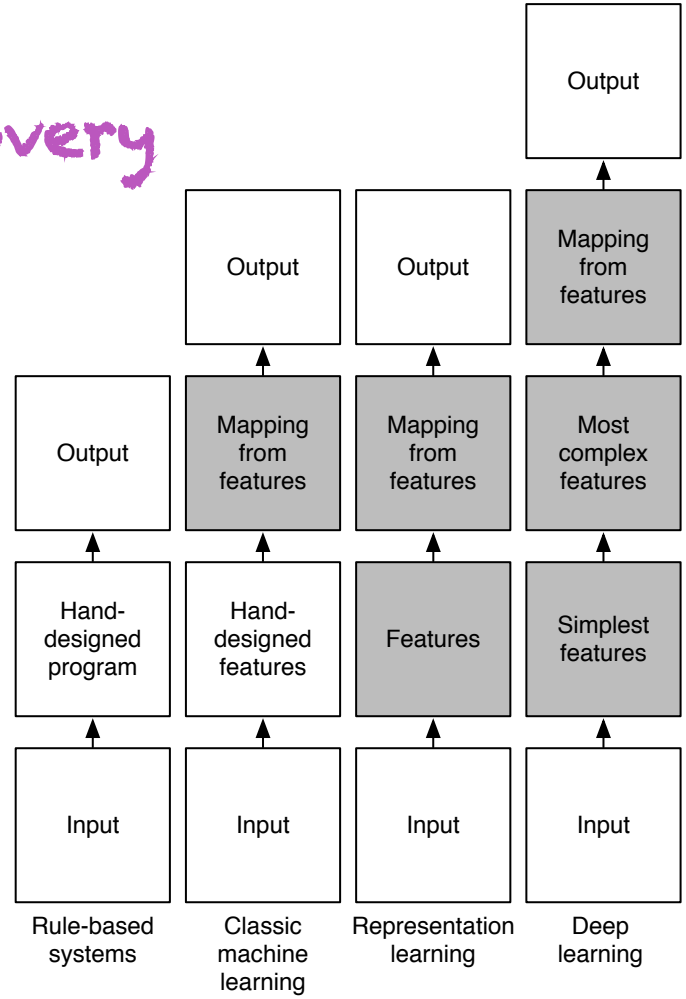
- *Zhou et al & Torralba, arXiv1412.6856 ,* ICLR 2015
- *Network trained to recognize places, not objects*



Deep Learning: Learning an Internal Representation

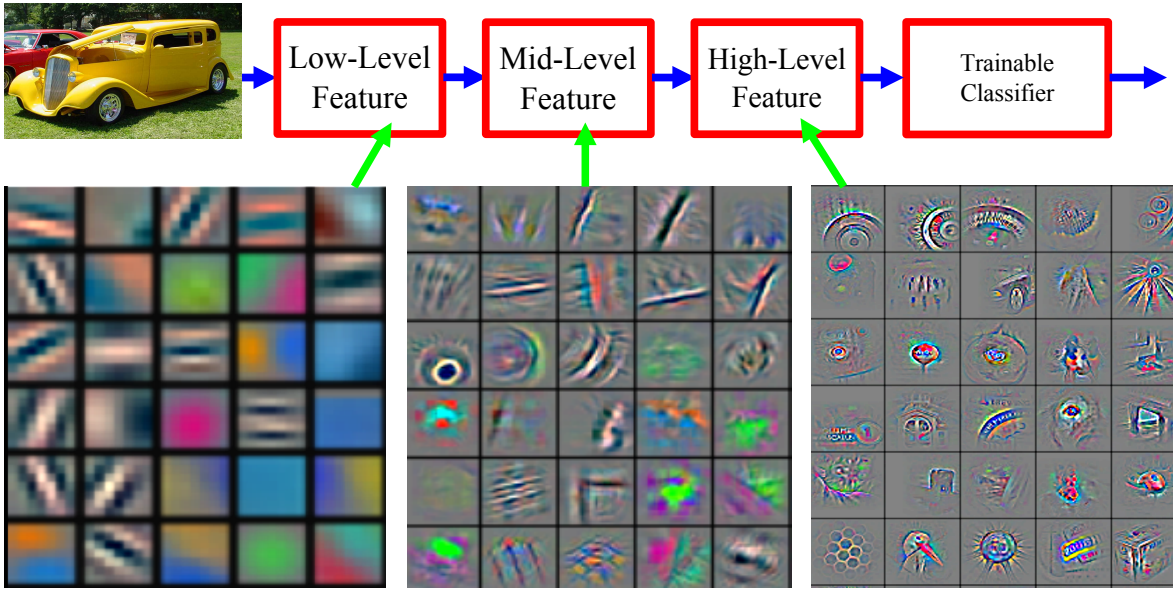
- Unlike other ML methods with either
  - no intermediate representation (linear)
  - or fixed (generally very high-dimensional) intermediate representations (SVMs, kernel machines)
- What is a good representation? Makes other tasks easier.

Automating Feature Discovery



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

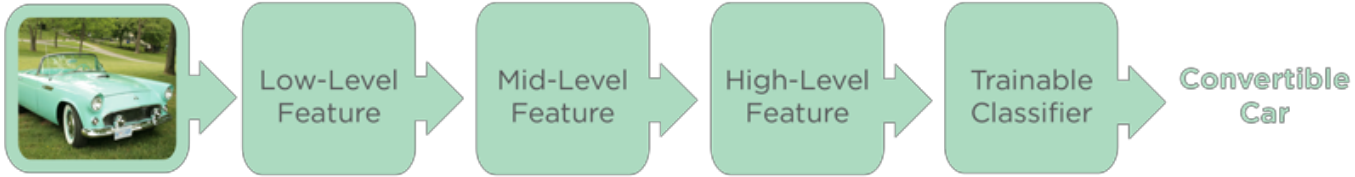


Credits: Yann LeCun

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



Credits: Yann LeCun

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

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

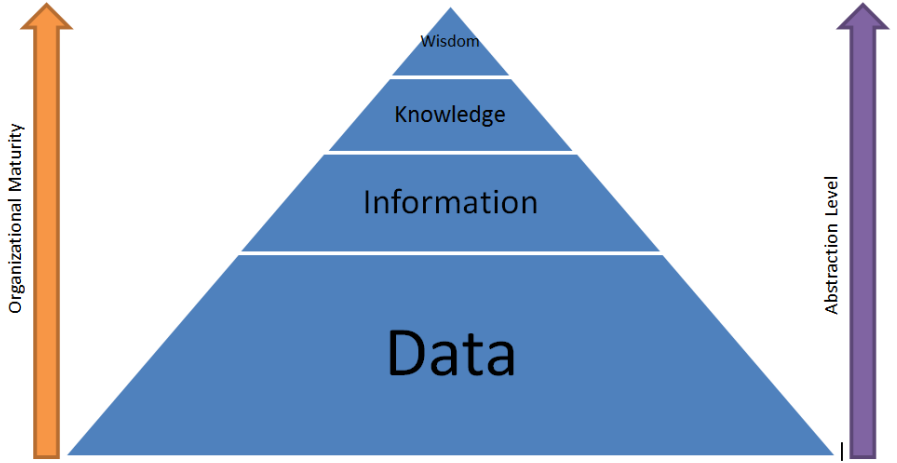
or, if hidden layers have width n and input has size n_0

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

Learning Multiple Levels of Abstraction

(Bengio & LeCun 2007)

- The big payoff of deep learning is to allow learning higher levels of abstraction
- Higher-level abstractions **disentangle the factors of variation**, which allows much easier generalization and transfer



Deep Nets and Backprop

Hidden units

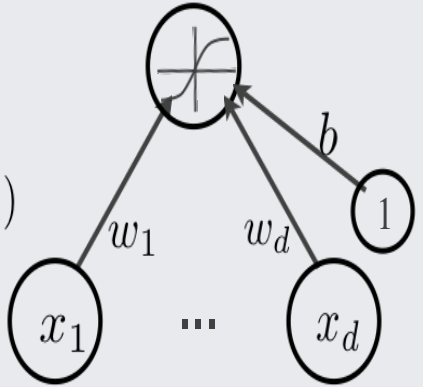
(from Hugo Larochelle)

- 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

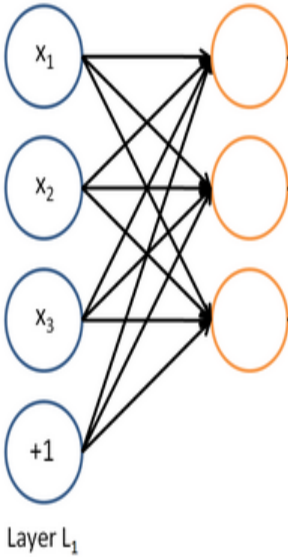
$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$



- $\mathbf{w}$ 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

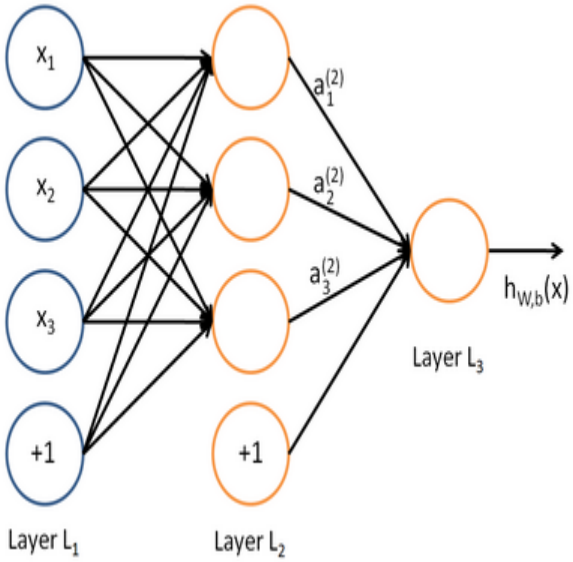
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs



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

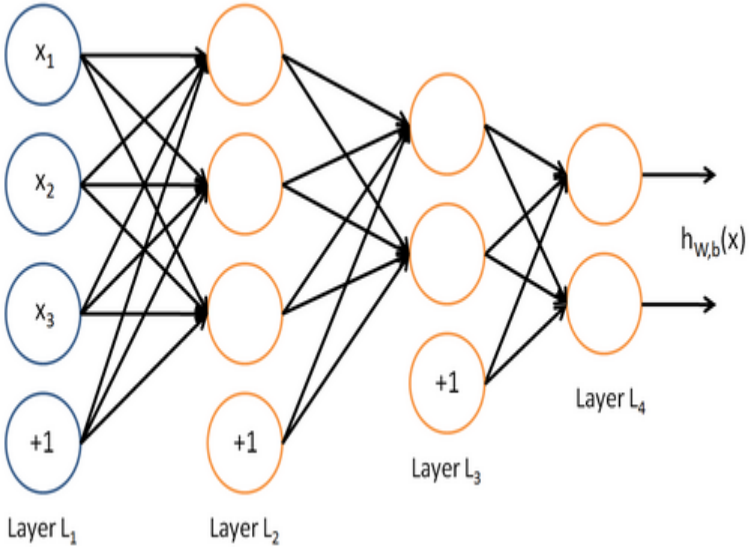
… which we can feed into another logistic regression function



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….

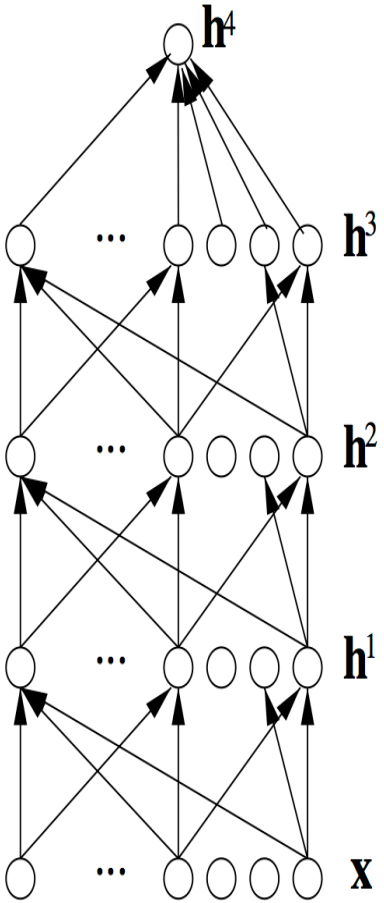


Multilayer network as universal approximator

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

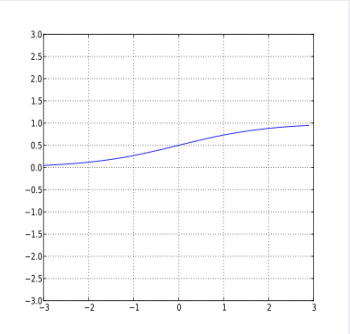
Non-Linearity = activation function

- 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

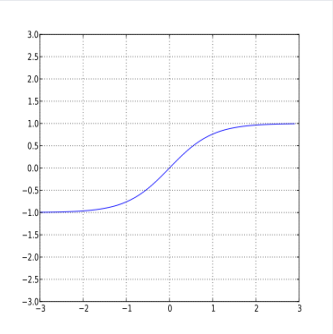
- Squashes the neuron's pre-activation between 0 and 1
- Always positive
- Bounded
- Strictly increasing



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

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

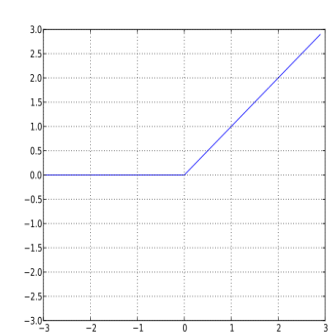
- For multi-class classification:
  - we need multiple outputs (1 output per class)
  - we would like to estimate the conditional probability $p(y = c|\mathbf{x})$
- We use the softmax activation function at the output:

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

  - 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



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

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

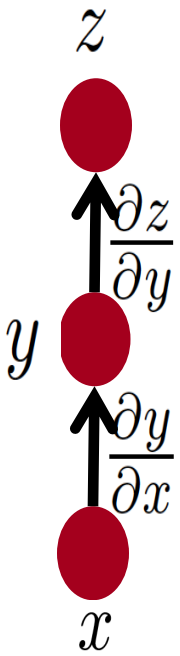
- Compute gradient of example-wise loss wrt parameters, by considering **intermediate values** such as the outputs of neurons

- **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



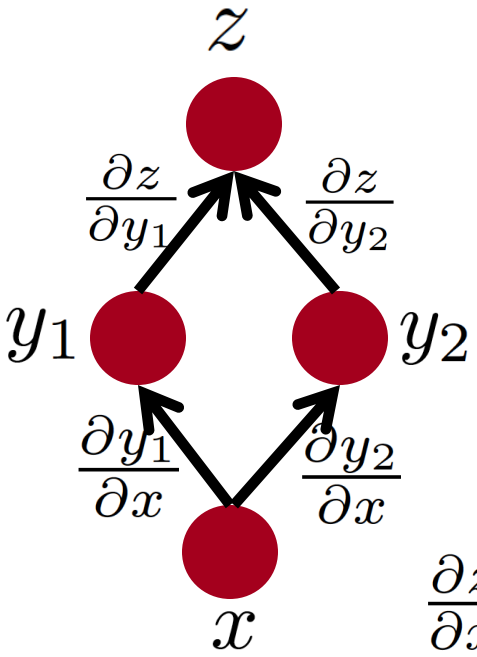
$$\Delta z = \frac{\partial z}{\partial y}\Delta y$$

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

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

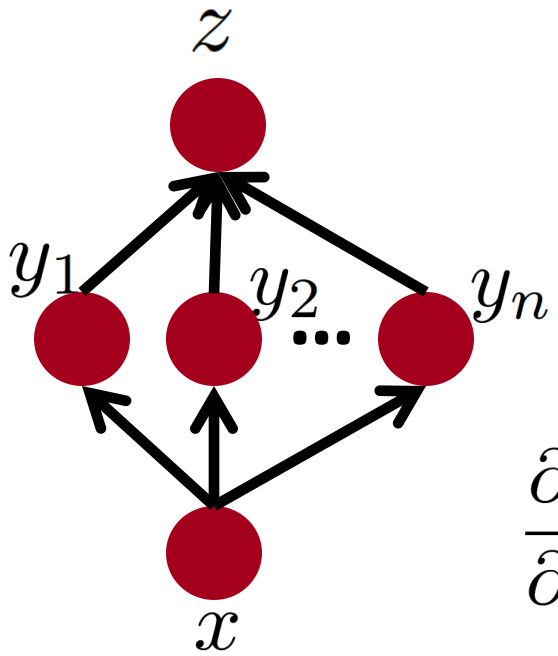
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

Multiple Paths Chain Rule



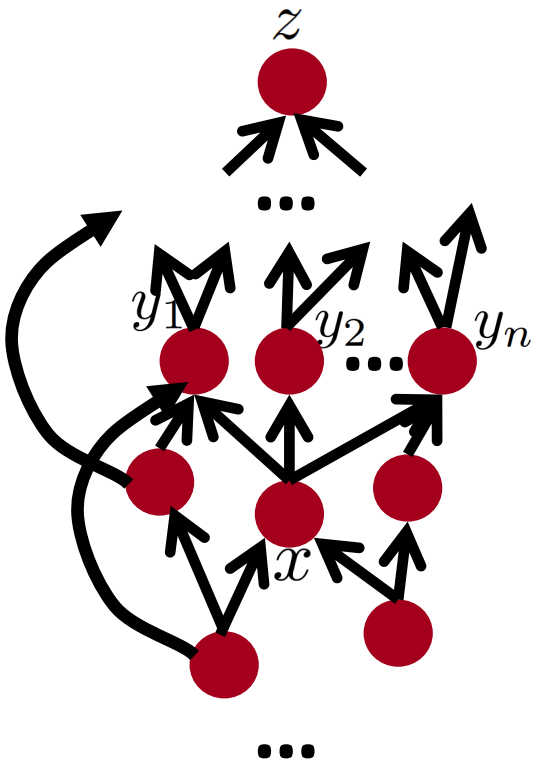
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

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



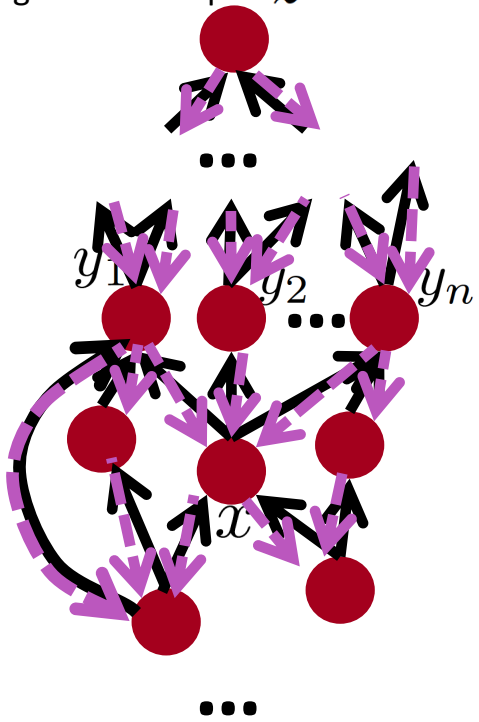
Flow graph: any directed acyclic graph
node = computation result
arc = computation dependency

$\{y_1, y_2, \ldots y_n\}$ = 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 General Flow Graph

Single scalar output z



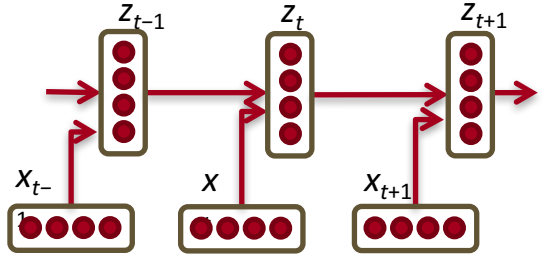
1. Fprop: visit nodes in topo-sort order
   - Compute value of node given predecessors
2. Bprop:
   - initialize output gradient = 1
   - visit nodes in reverse order:
     Compute gradient wrt each node using gradient wrt successors

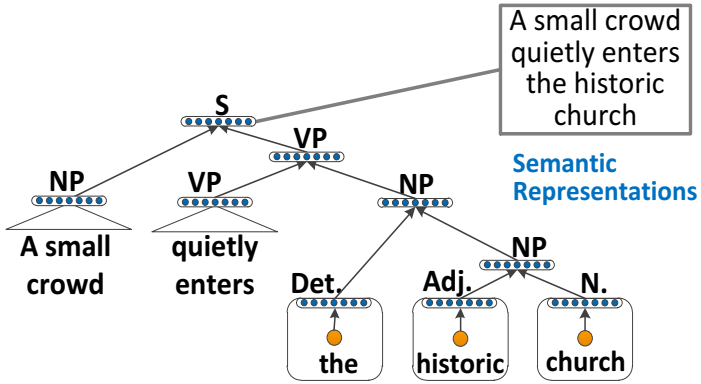
$\{y_1, y_2, \ldots y_n\}$ = successors of x

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

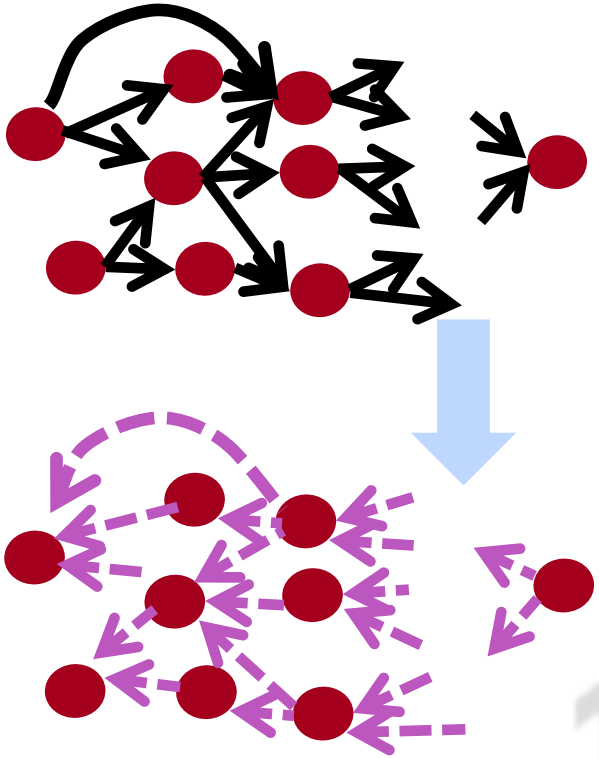
Back-Prop in Recurrent & Recursive Nets

- 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

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

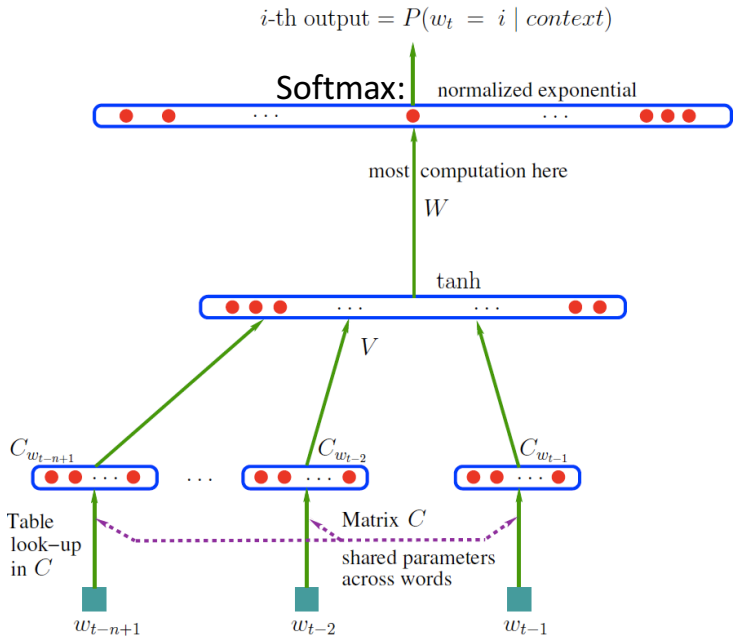
Combining Representations

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

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)

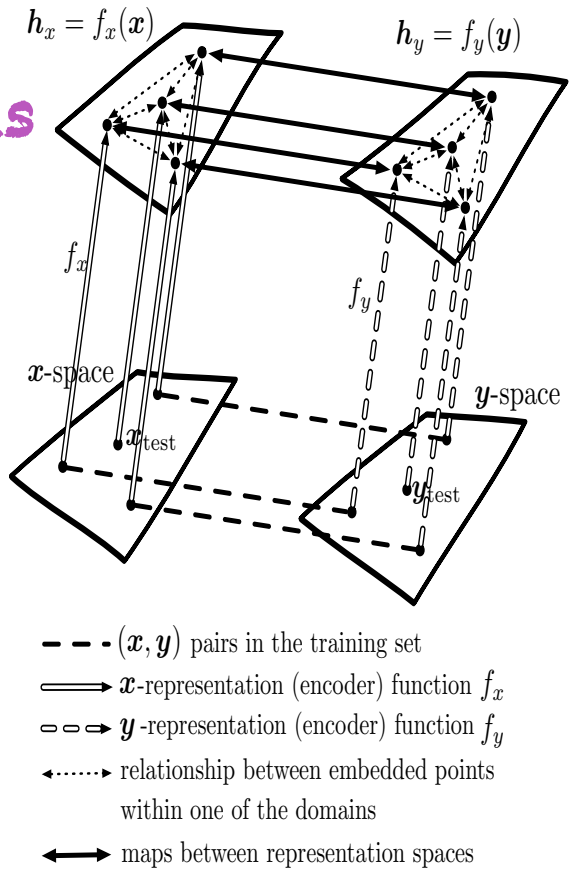
Learn $\Phi_I(\cdot)$ and $\Phi_W(\cdot)$ to optimize precision@k.

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*)

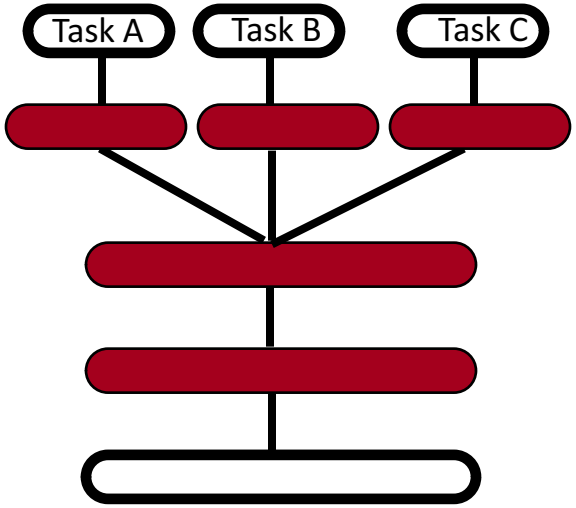
(Larochelle et al AAAI 2008)



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**

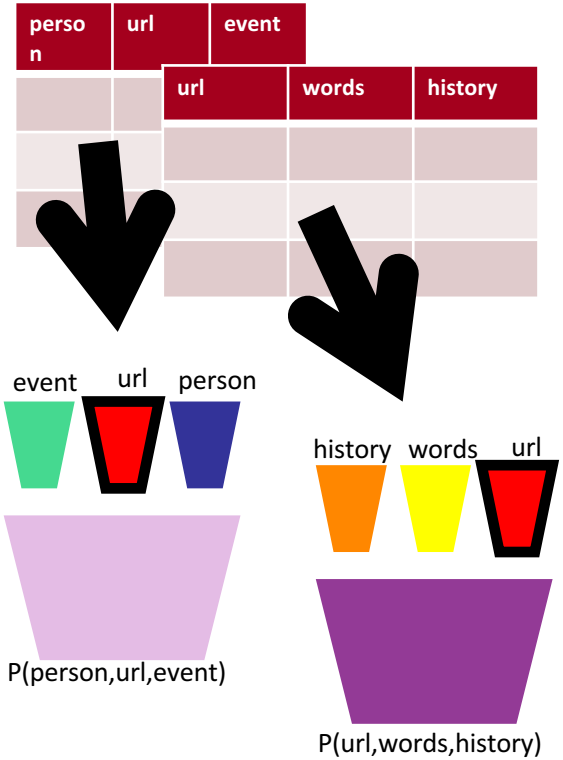


E.g. dictionary, with intermediate concepts re-used across many definitions

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)

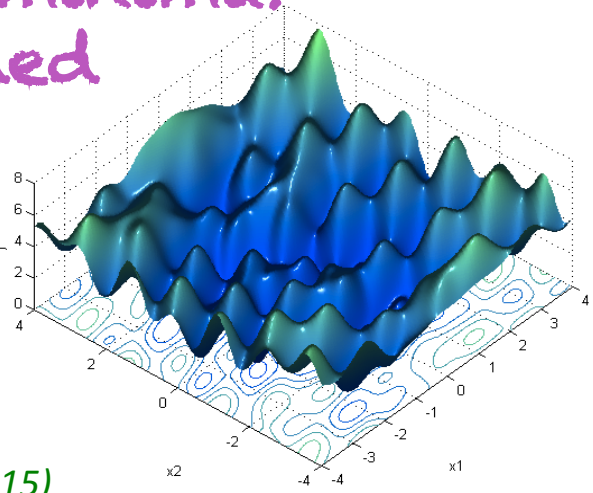


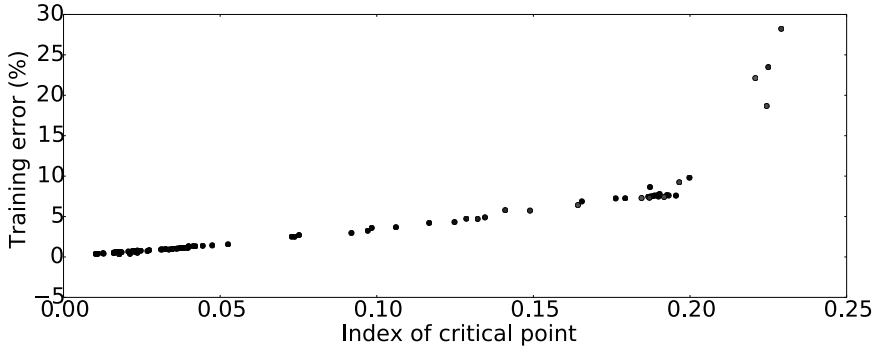
Not so terrible local minima: convexity is not needed

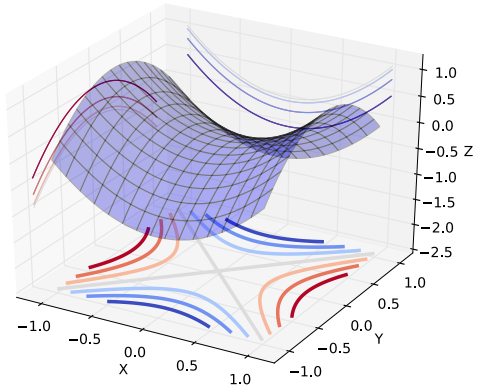
Myth busted:

- 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)







Injecting Noise in a Nonsmooth Net

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

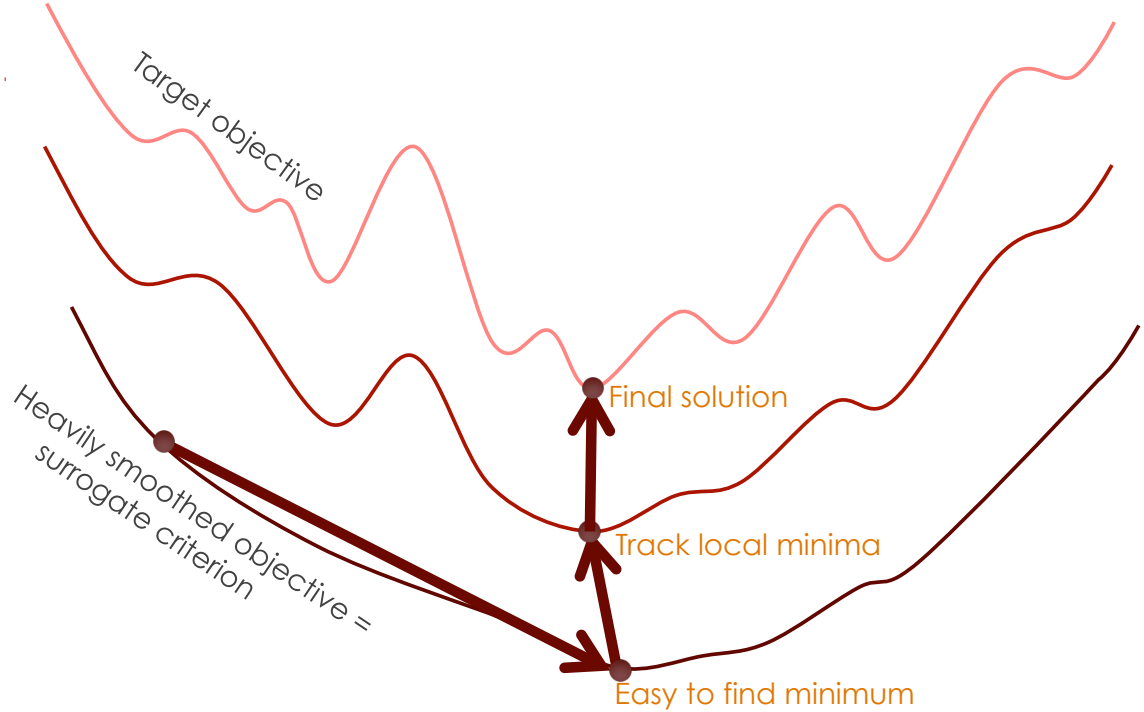
$$C(\theta) * \mathcal{N}(\epsilon) = \int_{\epsilon} C(\theta - \epsilon)\mathcal{N}(\epsilon)d\epsilon$$

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

- 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



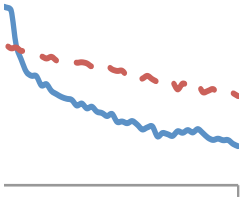
Order & Selection of Examples Matters

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

- Curriculum learning

  (Bengio et al 2009, Krueger & Dayan 2009)

  is a form of continuation method

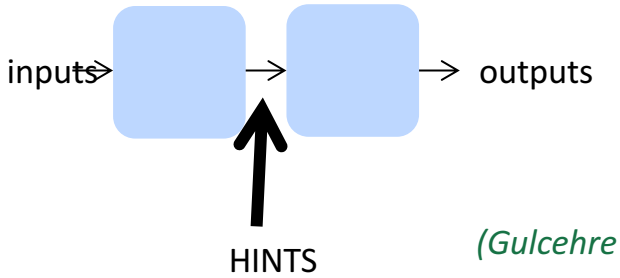


- **Start with easier examples**

- 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

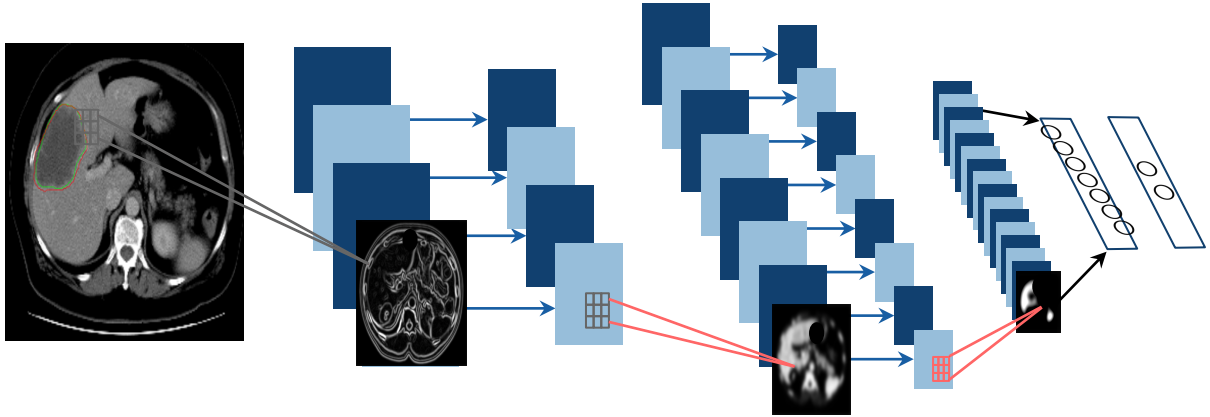
Convnets: Key Idea

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

Convolutional Neural Networks

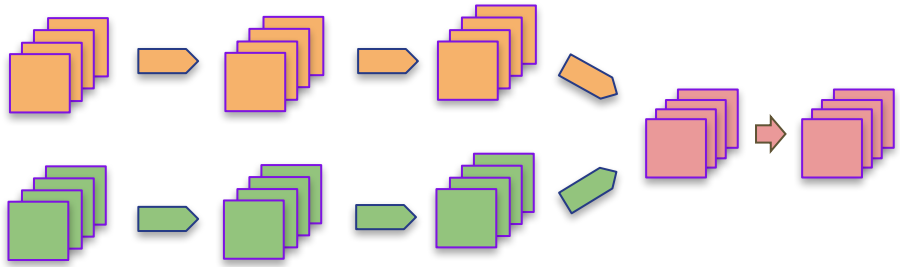
- A special kind of deep learning tailored for images
- Exploits the invariance to translations
- Exploits multi-scale hierarchy

Convolutional neural network for imaging data



Deep Data Fusion

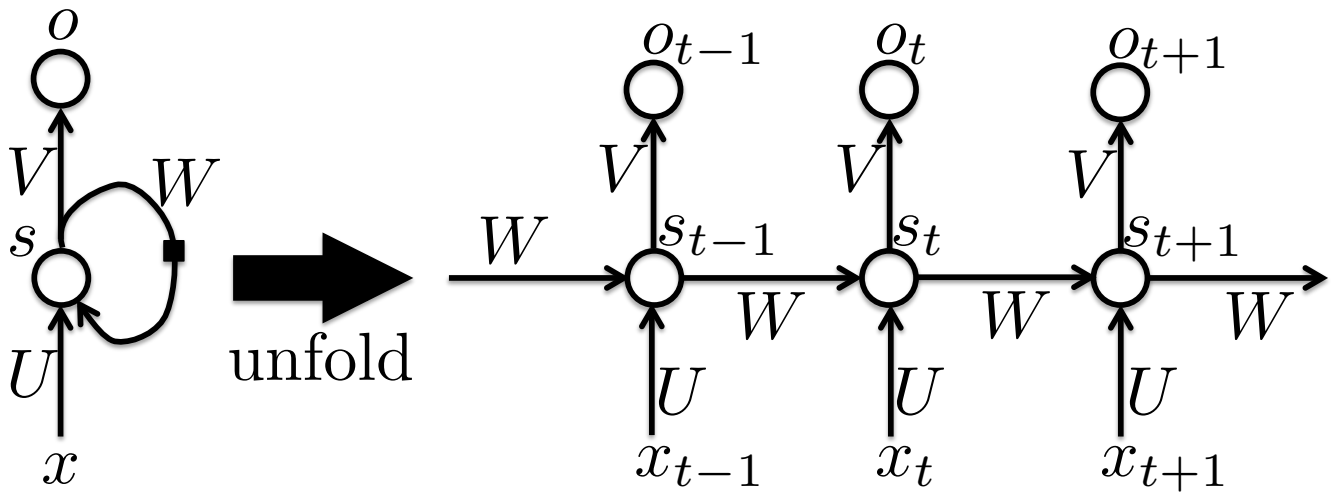
- Deep nets are very good at combining multiple sources of data, multiple sensors or modalities
- Can have separate pre-processing stages for each modality, then CONCATENATE the representations before continuing processing



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

Recurrent Neural Networks

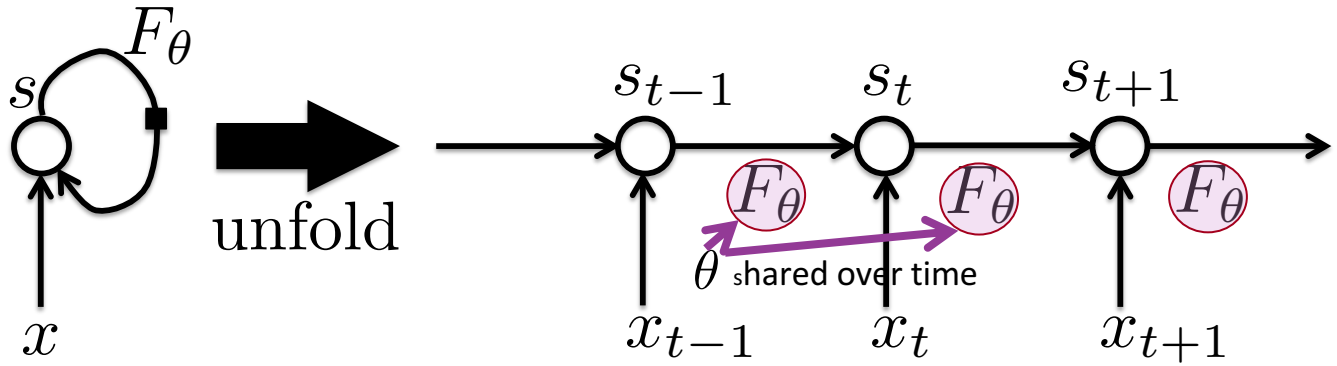
- Can produce an output at each time step: unfolding the graph tells us how to back-prop through time.



Recurrent Neural Networks

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

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



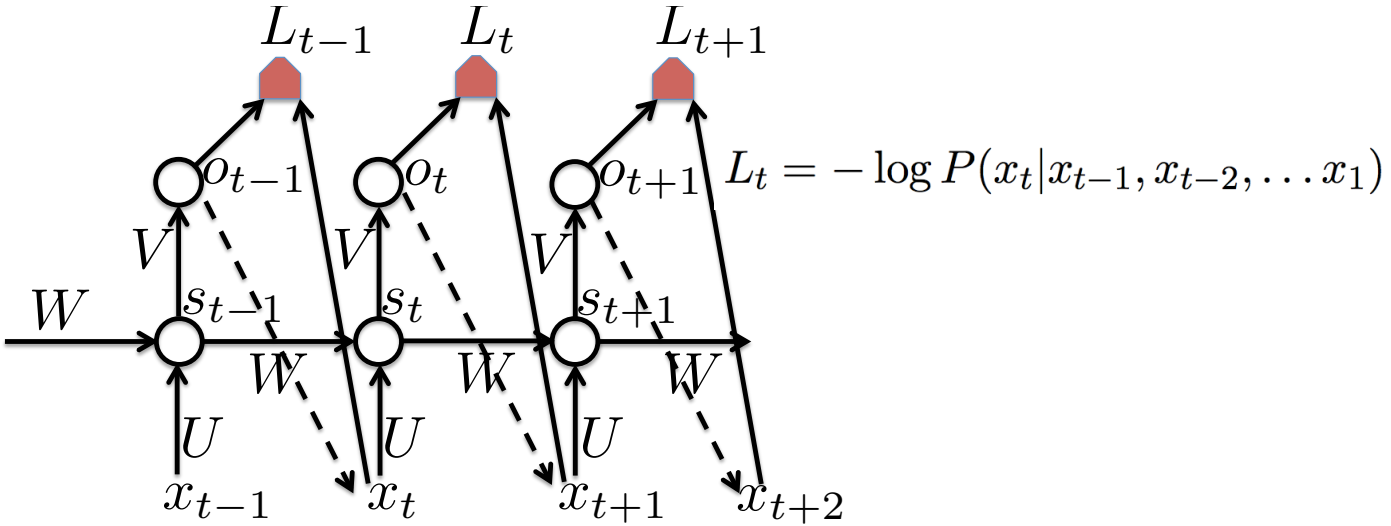
$$s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$$

➔ **Generalizes naturally to new lengths not seen during training**

Generative RNNs

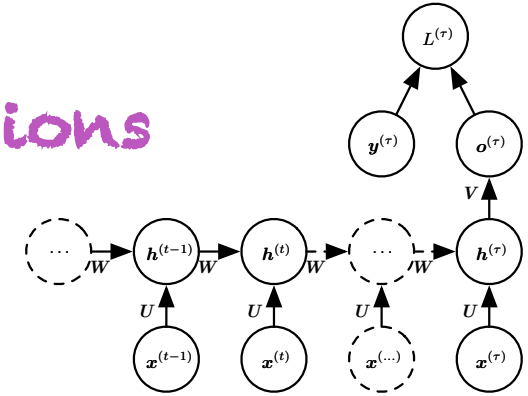
- An RNN can represent a fully-connected **directed generative model**: every variable predicted from all previous ones.

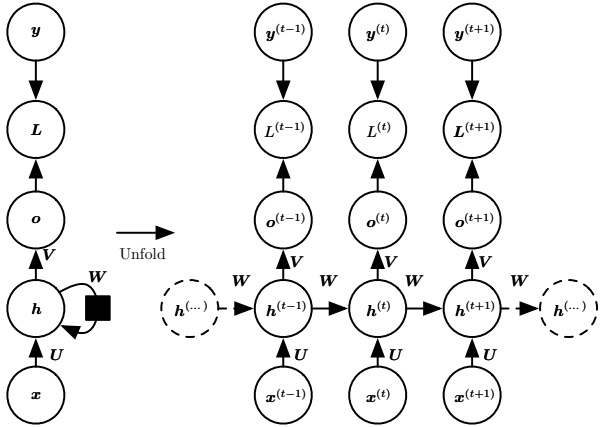
$$P(\mathbf{x}) = P(x_1, \ldots x_T) = \prod_{t=1}^{T} P(x_t | x_{t-1}, x_{t-2}, \ldots x_1)$$

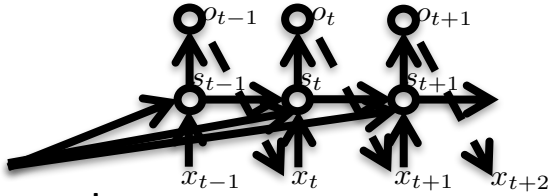


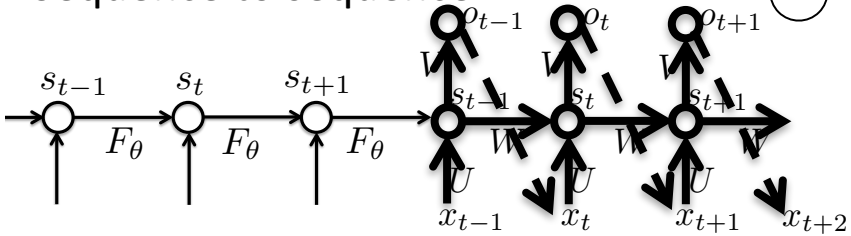
Conditional Distributions

- Sequence to vector
- Sequence to sequence of the same length, aligned
- Vector to sequence
- Sequence to sequence



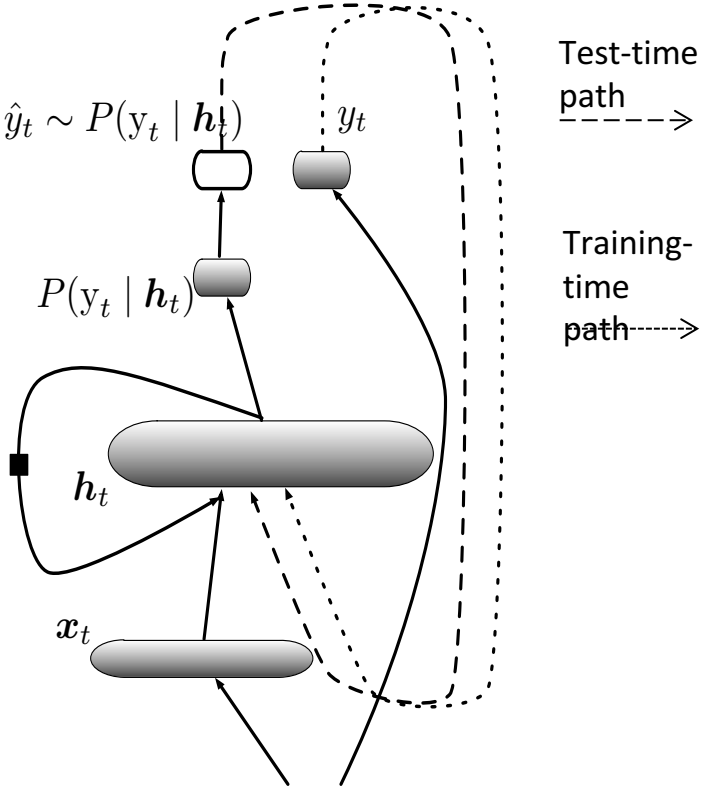






Maximum Likelihood = Teacher Forcing

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

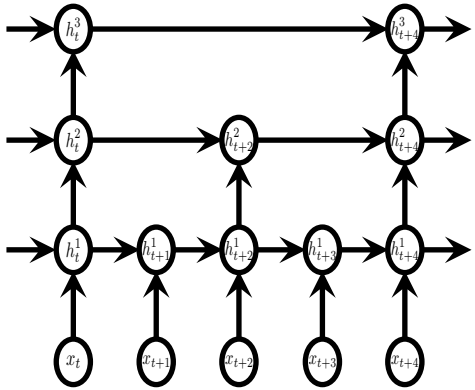


$(\boldsymbol{x}_t, y_t)$: next input/output training pair

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.



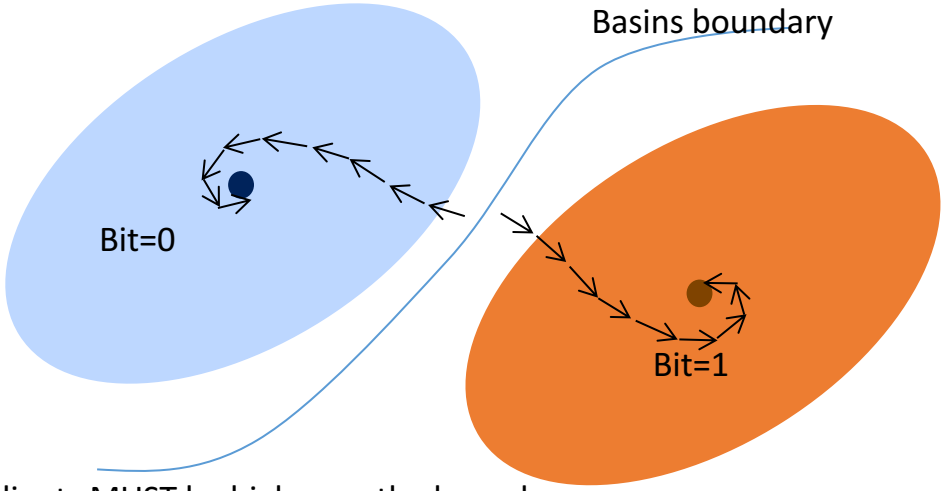
Learning Long-Term Dependencies with Gradient Descent is Difficult



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

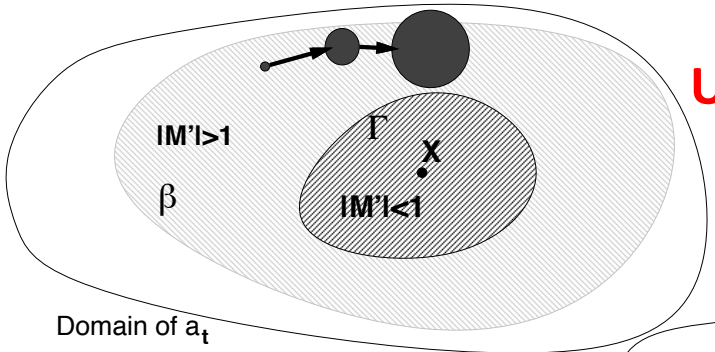
- Some subspace of the state can store 1 or more bits of information if the dynamical system has multiple basins of attraction in some dimensions



Note: gradients MUST be high near the boundary

Robustly storing 1 bit in the presence of bounded noise

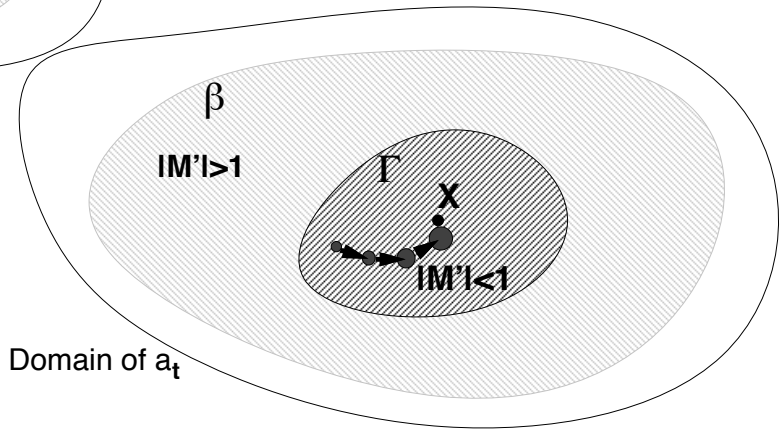
- With spectral radius > 1, noise can kick state out of attractor



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

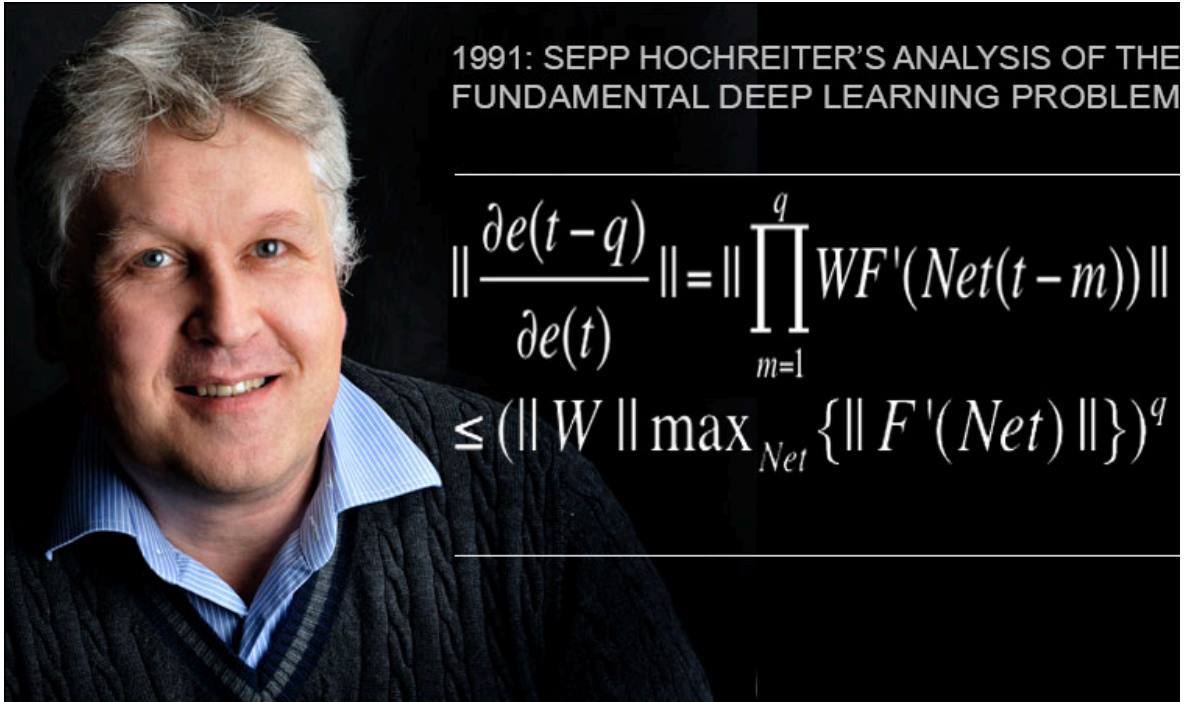
$$L = L(s_T(s_{T-1}(\dots s_{t+1}(s_t, \dots))))$$

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

- 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



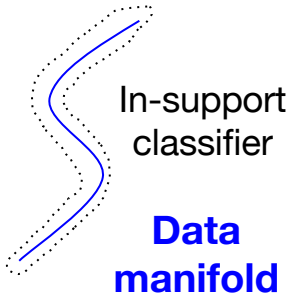
Why it hurts gradient-based learning

- 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

Classifiers for modeling distributions



- We were inspired by the work of Gutmann & Hyvarinen using probabilistic classifiers to estimate energy functions

  Gutmann & Hyvarinen 2012, Noise-Contrastive Estimation

- In high dimension, more relevant then density is whether you are in-support vs out-of-support

- A classifier of in-support vs out-of-support pays a *constant* price (rather than huge) for not putting support at a training example

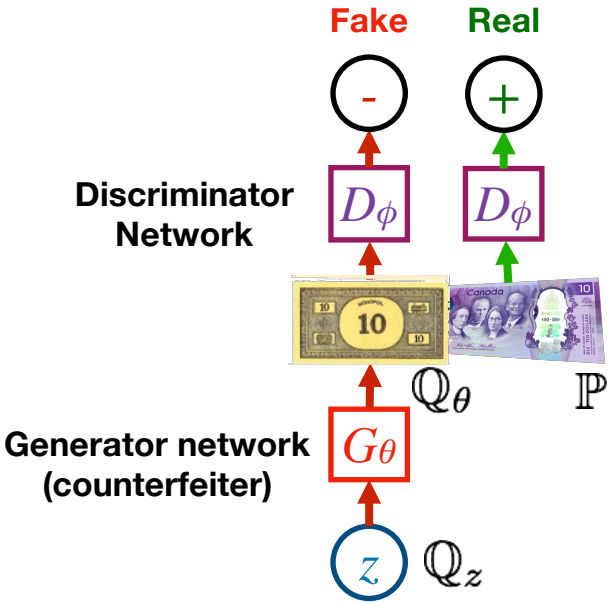
Generative adversarial networks (GANs): a two player game with neural networks

Givens: $\mathbb{Q}_z$ $\mathbb{P}$

Samples from a **target distribution**
(Simple) prior

Player 1: Generator
A neural network with parameters, θ, whose samples **fool the discriminator**

Player 2: Discriminator
Distinguish (classify) real and fake correctly



[Goodfellow et. Al & Bengio, 2014]

Convolutional GANs

(Radford et al, arXiv 1511.06343)

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

Generative Adversarial Networks

Goodfellow et al & Bengio NIPS 2014

2014 2015 2016 2017 2018

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

Xu et al 2018, AttnGAN

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

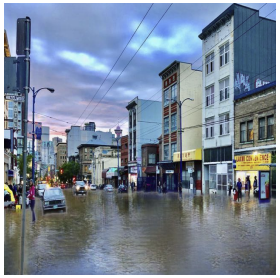
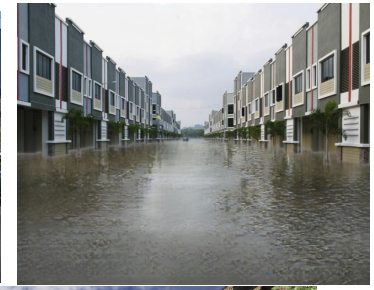
Zhang et al. 2017

Lucy Li

The Climate Change Visualization Project

- We are working on a project that aims to generate **images** that depict **vivid** and **personalized** outcomes of climate change using GAN-like models.




With Sasha Luccioni, Victor Schmidt, Vahe Vardanyan

Mila

From Attention to System 2 Deep Learning

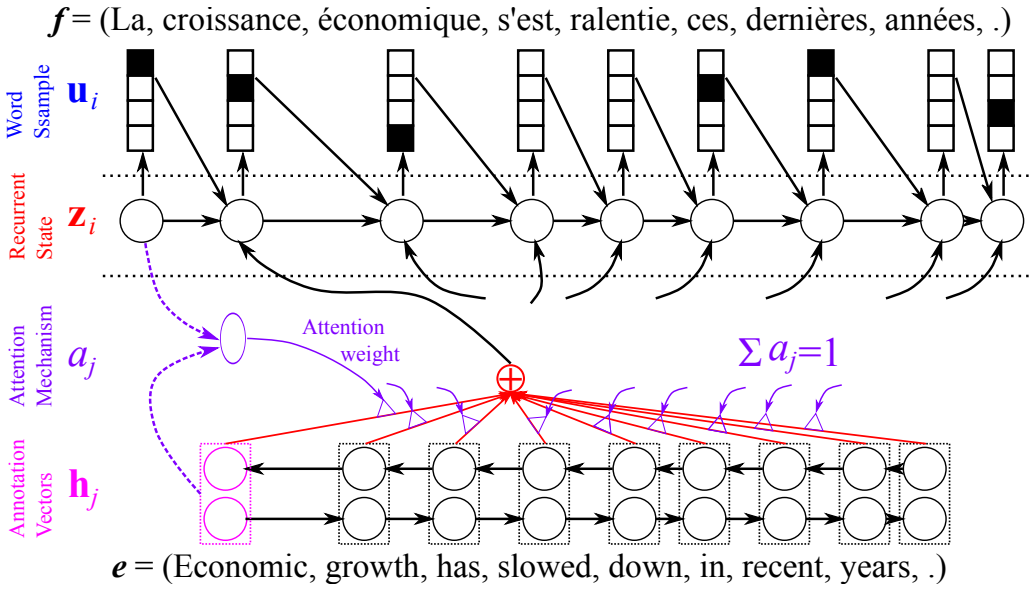
Gating for Attention-Based Neural Machine Translation

Related to earlier Graves 2013 for generating handwriting

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

Soft-matching of key & value:

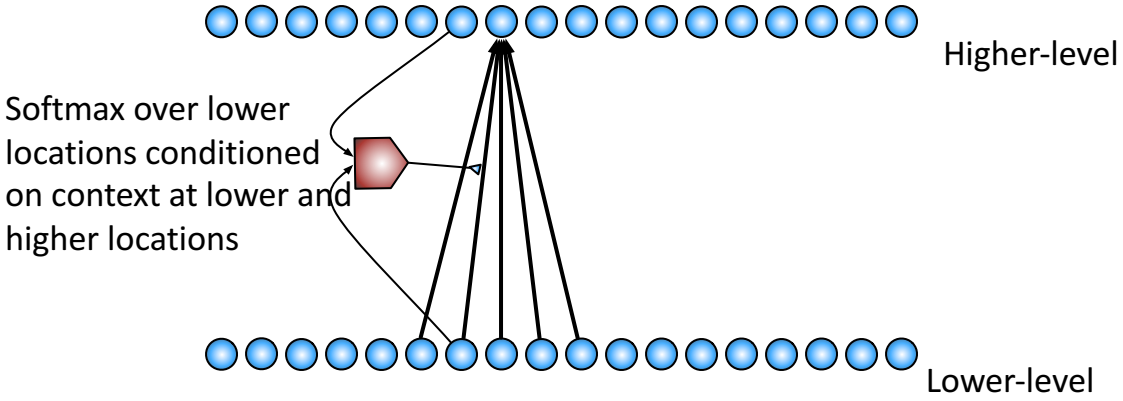
input seen as a set of
(key, value) objects



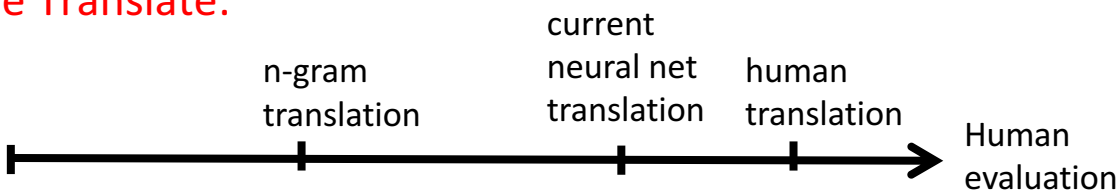
What's New with Deep Learning?

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

Neural Machine Translation (ICLR'2015)

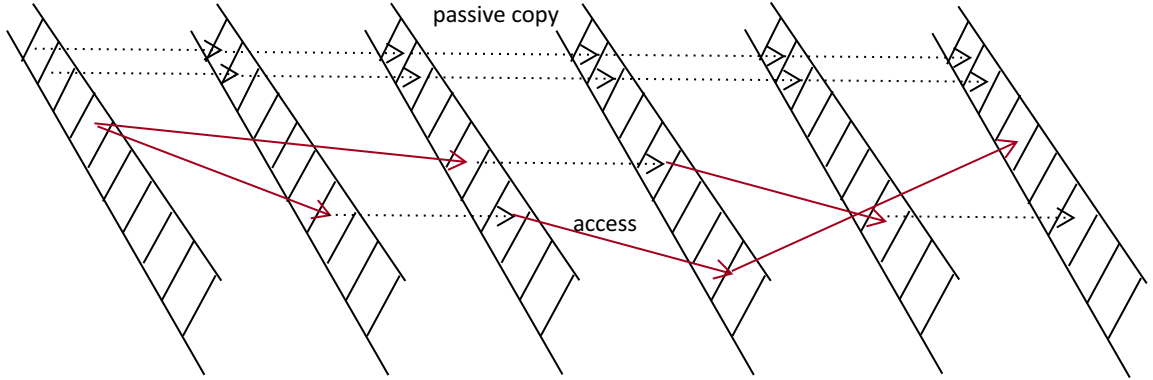


- Google Translate:



Memory-Extended 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



Multi-Head Attention

We can run multiple attention mechanisms in parallel to focus on different aspects of the data

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHeadAttention(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

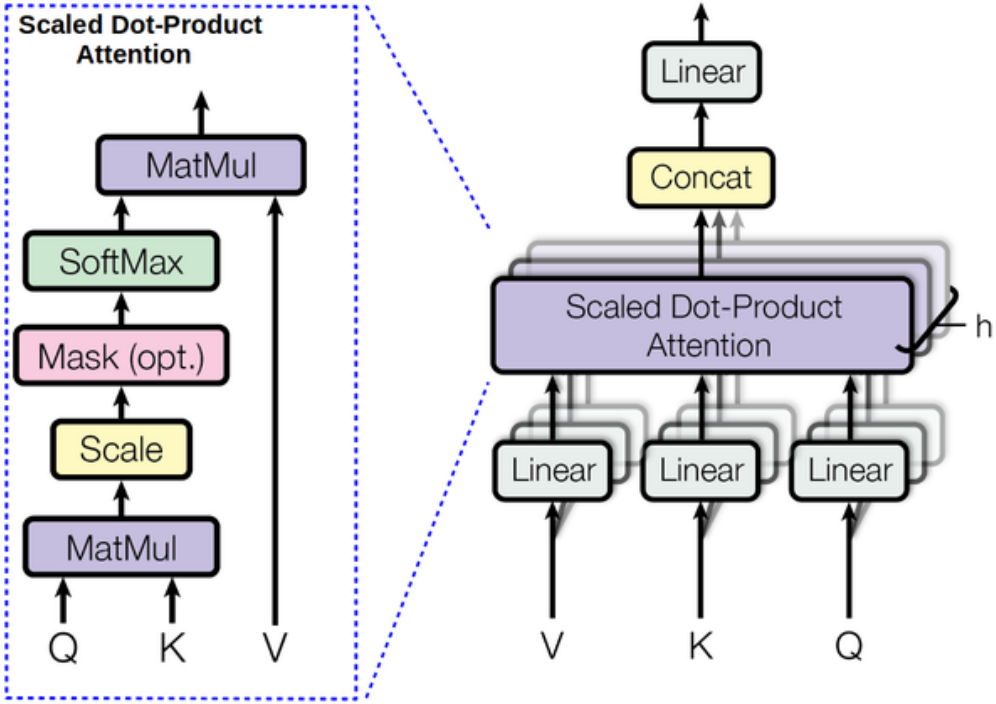
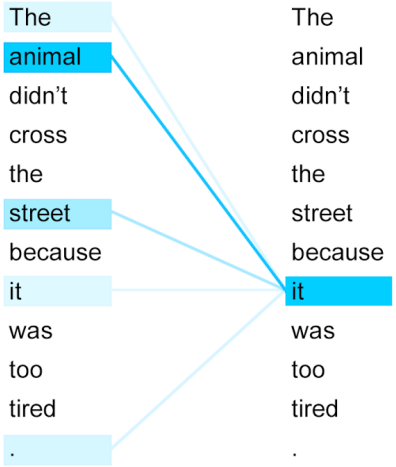


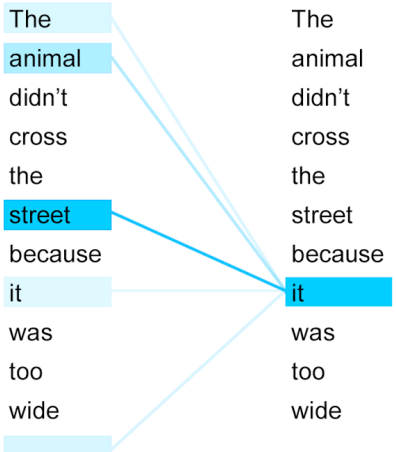
Fig: Michal Chromiak's blog

Self-Attention & Transformers

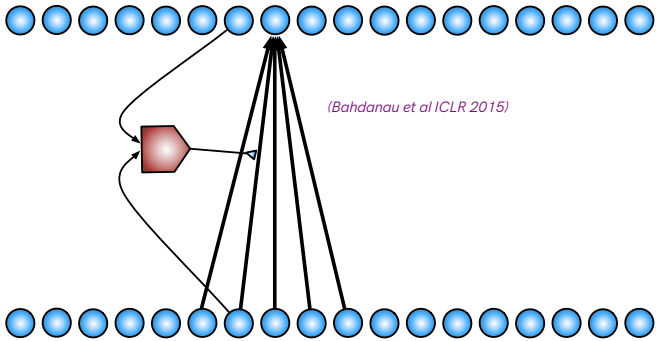
Lin et al ICLR 2017;
Vaswani et al NIPS'2017

- **Use attention on previous computation**
- **Parallelize** encoder
- Encode location of each item, no need for RNN
- Transform each location based on attention from all others
- See also Sparse Attentive Backtracking, Ke et al NIPS'2018

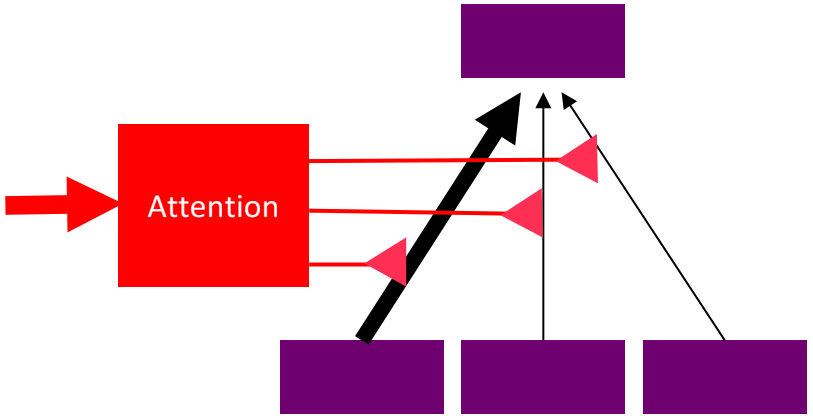




CORE INGREDIENT FOR CONSCIOUS PROCESSING: **ATTENTION**



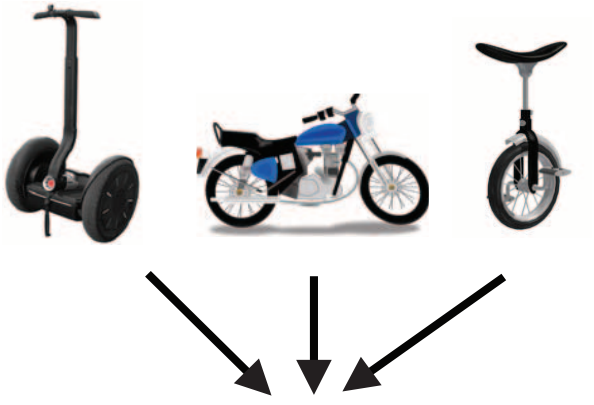
- **Focus** on a one or a few elements at a time
- **Content-based soft attention** is convenient, can backprop to *learn where to attend*
- Attention is an **internal action**, needs a **learned attention policy** *(Egger et al 2019)*



- Operating on unordered SETS of (key, value) pairs
- SOTA in NLP

SYSTEMATIC GENERALIZATION

- Studied in linguistics

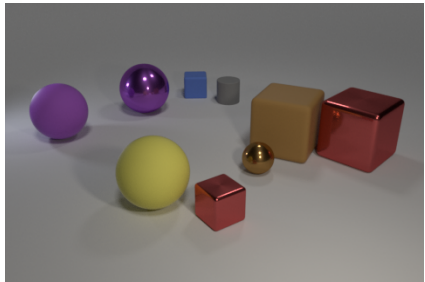


- **Dynamically recombine existing concepts**
- Even when new combinations have 0 probability under training distribution
  - E.g. Science fiction scenarios
  - E.g. Driving in an unknown city

(Lake et al 2015)

- Not very successful with current DL, which can "overfit" the training **distribution**

(Lake & Baroni 2017)
(Bahdanau et al & Courville ICLR 2019)
CLOSURE: (Bahdanau et al & Courville arXiv:1912.05783) on CLEVR



Mila

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

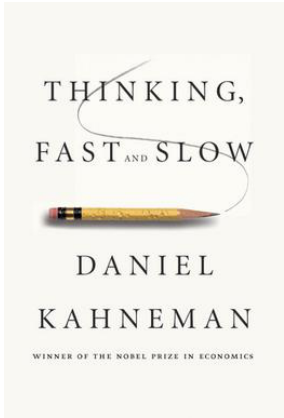
SYSTEM 1 VS. SYSTEM 2 COGNITION

2 systems (and categories of cognitive tasks):

Manipulates high-level / semantic concepts, which can be recombined combinatorially

System 1

- Intuitive, fast, **UNCONSCIOUS**, 1-step parallel, non-linguistic, habitual
- Implicit knowledge
- Current DL



System 2

- Slow, logical, **sequential**, **CONSCIOUS**, linguistic, algorithmic, planning, reasoning
- Explicit knowledge
- DL 2.0

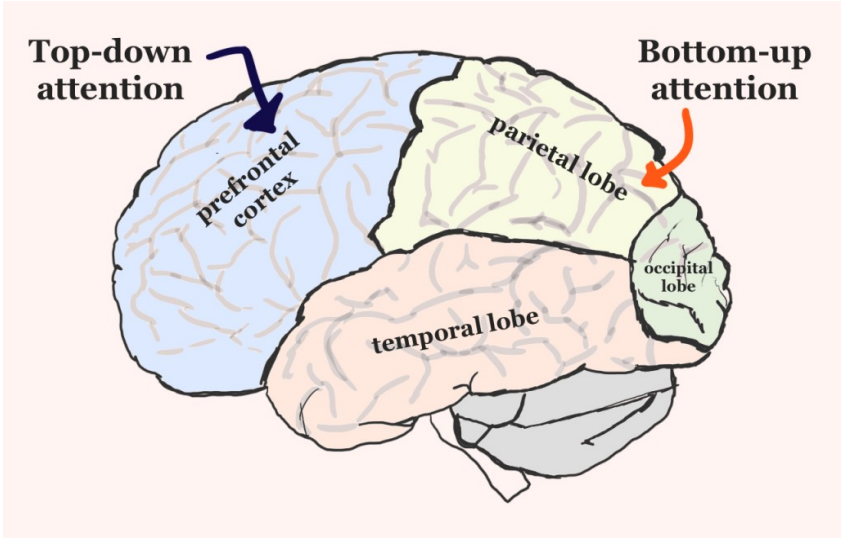
FROM ATTENTION TO **CONSCIOUSNESS**

C-word not taboo anymore in cognitive neuroscience

Global Workspace Theory

(Baars 1988++, Dehaene 2003++)

- Bottleneck of conscious processing
  - *WHY A BOTTLENECK?*
- Selected item is broadcast, stored in short-term memory, conditions perception and action
- System 2-like sequential processing, conscious reasoning & planning & imagination
- Can only run 1 simulation at a time, unlike a movie, only few abstract concepts involved at each step



IMPLICIT VS VERBALIZABLE KNOWLEDGE: UNDERLYING ASSUMPTIONS BEHIND VERBALIZABLE KNOWLEDGE

- Most knowledge in our brain is implicit and **not verbalizable** (hence the explainability challenge, even for humans)
- Some of our knowledge however is verbalizable and we can reason and plan explicitly with it
- The concepts manipulated in this way are those we can name with language
- The joint distribution between these concepts and the way that distribution can change over time satisfies special assumptions, exploited in system 2 tasks and conscious processing
- We want to clarify these assumptions as priors to be able to embed them in ML architectures and training frameworks, for better in- and out-of distribution generalization

SOME SYSTEM 2 INDUCTIVE PRIORS
all inspired by human cognition

- Sparse factor graph in space of high-level semantic variables
- Semantic variables are causal: agents, intentions, controllable objects
- Distributional changes due to localized causal interventions (in semantic space)
- Simple mapping between high-level semantic variables / thoughts and words / sentences
- Shared ’rules’ across instance tuples (as arguments), requiring variables & indirection
- Meaning (e.g. grounded by an encoder) is stable & robust wrt changes in distribution
- Credit assignment is only over short causal chains

CONSCIOUSNESS **PRIOR**
➔ SPARSE FACTOR GRAPH

Bengio 2017, arXiv:1709.08568

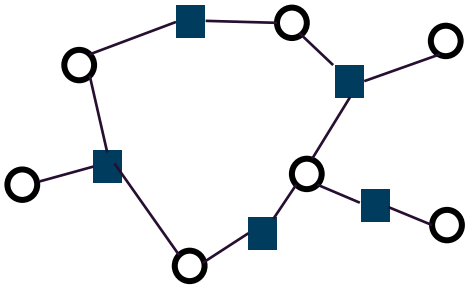
- Property of **high-level variables which we manipulate with language**:

  we can predict

  some given very few others

  - E.g. "if I drop the ball, it will fall on the ground"
- **Disentangled factors** $\neq$ marginally independent,

  e.g. ball & hand
- **Prior**: sparse factor graph joint distribution between high-level variables
- Inference involves few variables at a time, selected by **attention mechanism** and memory retrieval

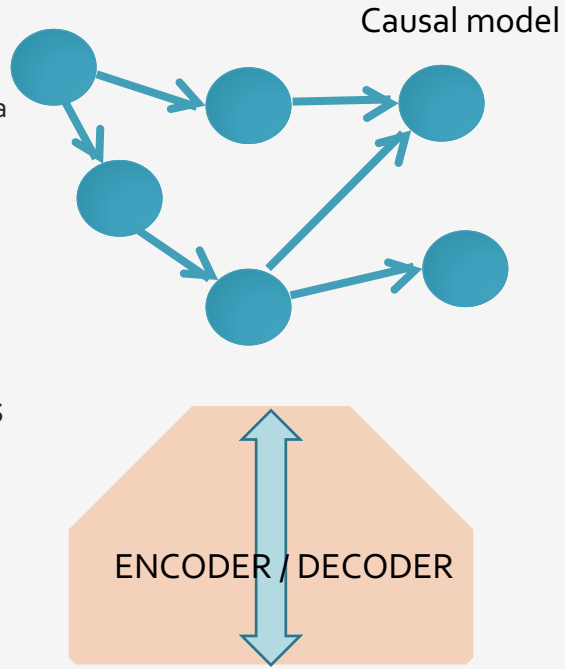


Mila

Deep Learning Objective: discover causal representation

- What are the right representations? Causal variables explaining the data
- How to discover them (as a function of observed data)?
- How to discover their causal relationship, the causal graph?
- How are actions corresponding to causal interventions?
- How is raw sensory data mapped to high-level causal variables and how do high-level causal variables turn into low-level actions and partial observations?

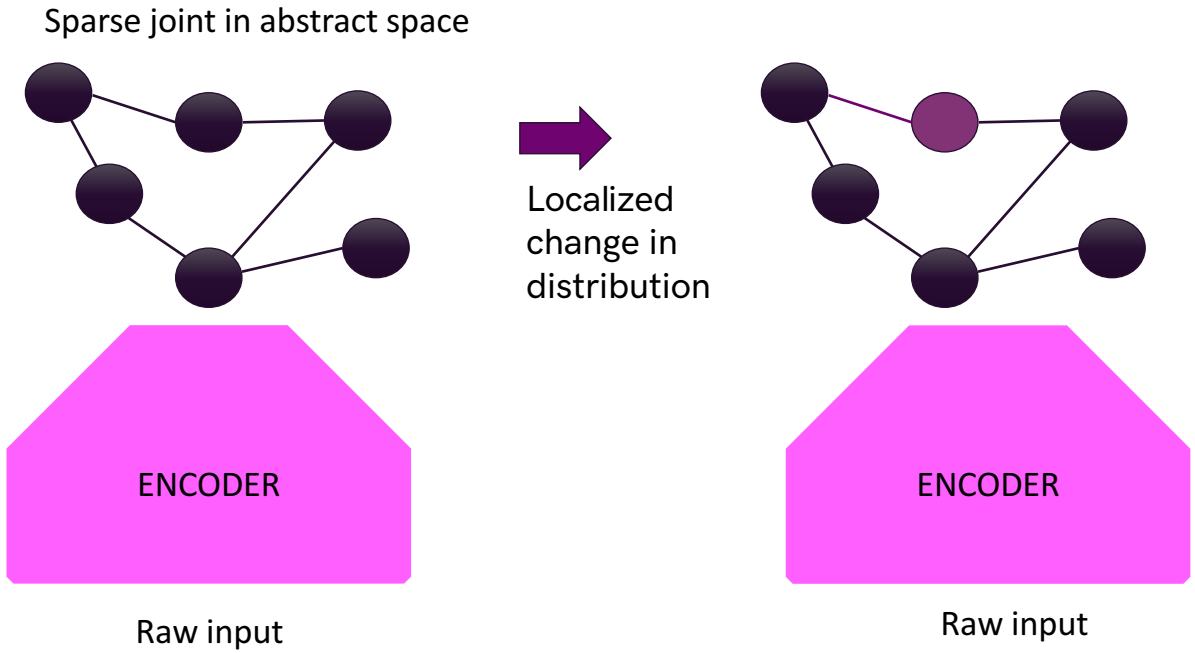
Need additional biases: causality is about changes in distribution



Mila

SPARSE CHANGE IN ABSTRACT LATENT SPACE

Change may be drastic in pixel space but tiny in semantic space of causal variables

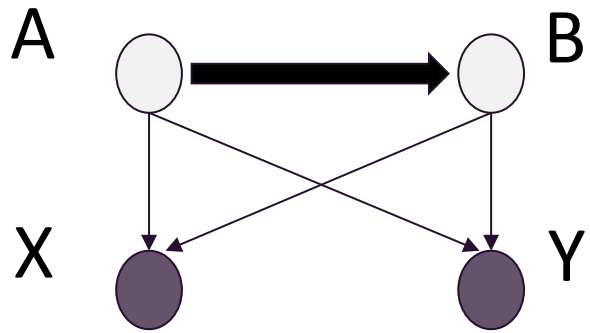


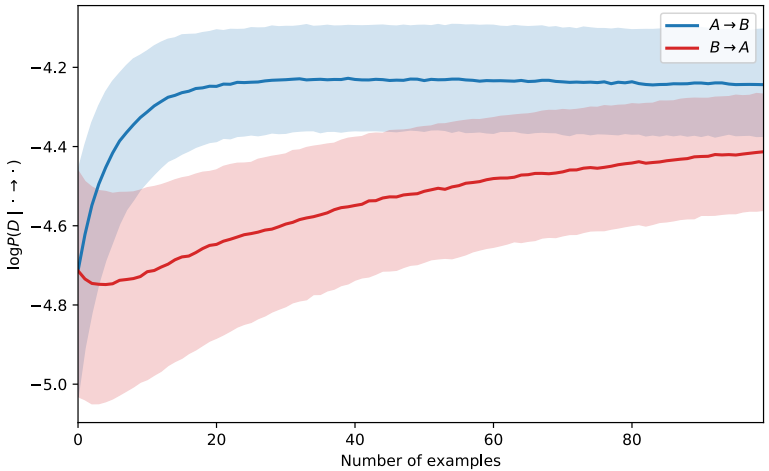
EXAMPLE: DISCOVERING CAUSE AND EFFECT
= HOW TO FACTORIZE A JOINT DISTRIBUTION?

A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms

- Learning whether A causes B or vice-versa
- Learning to disentangle (A,B) from observed (X,Y)
- Exploit changes in distribution and speed of adaptation to guess causal direction

Bengio et al 2019 arXiv:1901.10912





- *Ongoing work: theory proving when the correct model converges faster by online SGD*

Mila

EXAMPLE: DISCOVERING CAUSE AND EFFECT = HOW TO FACTORIZE A JOINT DISTRIBUTION?

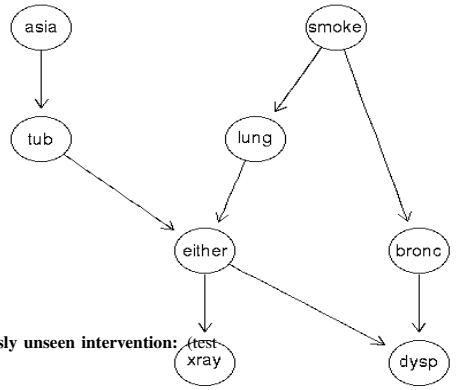
Learning Neural Causal Models from Unknown Interventions *Ke et al 2019 arXiv:1910.01075*

Dependency Structure Discovery from Interventions *Ke et al 2020, submitted*

- Learning small causal graphs, avoid exponential explosion of # of graphs by parametrizing factorized distribution over graphs
- With enough observations of changes in distribution: perfect recovery of the causal graph without knowing the intervention; converges faster on sparser graphs
- Inference over the intervention: faster causal

Table 1: **Baseline comparisons:** Structural Hamming Distance (SHD) (lower is better) for learned and ground-truth edges on various graphs from both synthetic and real datasets, compared to [33], [48], [14], [11] and [10]. The proposed method (Structural Discovery from Interventions (SDI)) is run on random seeds $1-5$ and we pick the worst performing model out of the random seeds in the table. OOM: out of memory. Our proposed method correctly recovers the true causal graph, with the exception of Sachs and full13, and it significantly outperforms all other baseline methods.

Method	Asia	Sachs	collider	chain	jungle	collider	full
M	8	11	8	13	13	13	13
Zheng et al. [10]	14	22	18	39	22	24	71
Yu et al. [11]	10	19	7	14	16	12	77
Heinze-Deml et al. [48]	8	17	7	12	12	7	28
Peters et al. [33]	5	17	2	2	8	2	16
Eaton and Murphy [49]	0	OOM	7	OOM	OOM	OOM	OOM
Proposed Method (SDI)	0	6	0	0	0	0	7



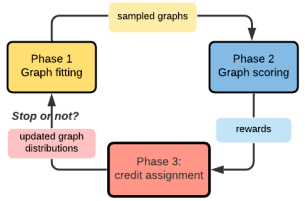


Figure 2: Workflow for our proposed method SDI. Phase 1 samples graphs under the model's current belief about the edge structure and fits parameters to observational data. Phase 2 scores a small set of graphs against interventional data and assigns rewards according to graphs' ability to predict interventions. Phase 3 uses the rewards from Phase 2 to update the beliefs about the edge structure. If the believed edge probabilities have all saturated near 0 or 1, the method has converged.

Table 2: **Evaluating the consequences of a previously unseen intervention:** (test log-likelihood under intervention)

	fork3	chain3	confounder3	collider3
Baseline	-0.5036	-0.4562	-0.3628	-0.5082
SDI	-0.4502	-0.3801	-0.2819	-0.4677

THOUGHTS, CONSCIOUSNESS, LANGUAGE

- Consciousness: from humans reporting
- High-level representations ⟺ language
- High-level concepts: meaning anchored in low-level perception and action → **tie system 1 & 2**
- Grounded high-level concepts

→ better natural language understanding

- **Grounded language learning**
  e.g. BabyAI: *(Chevalier-Boisvert and al ICLR 2019)*

RIMS: MODULARIZE COMPUTATION AND OPERATE ON SETS OF NAMED AND TYPED OBJECTS

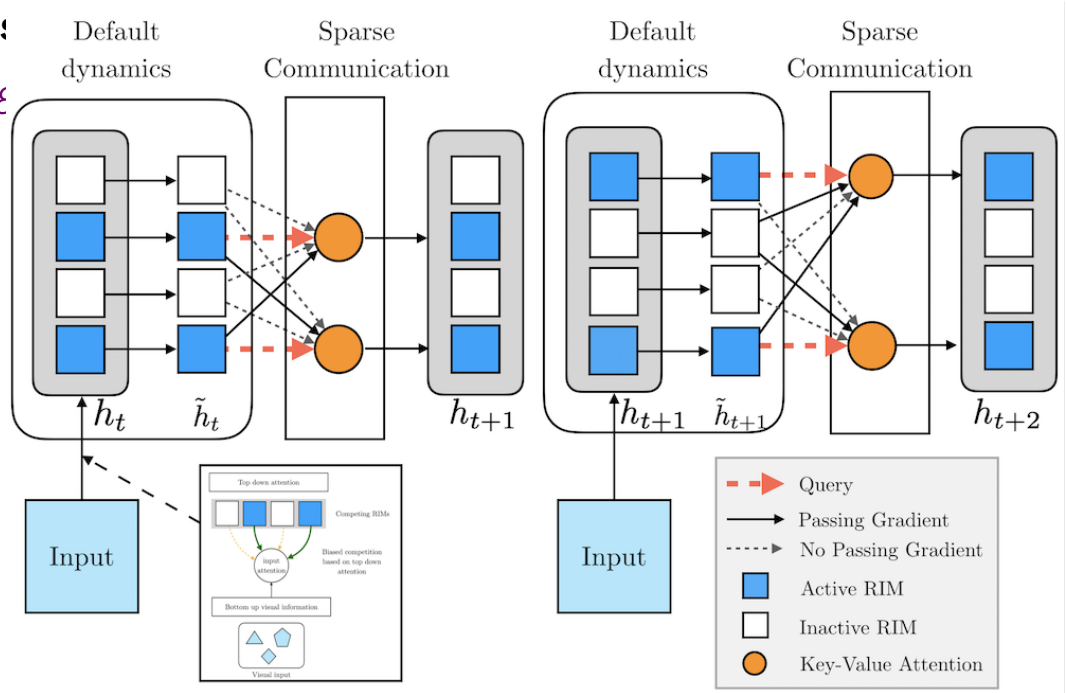
Recurrent Independent Mechanisms

Goyal et al 2019, arXiv:1909.10

Multiple recurrent sparsely interacting modules, each with their own dynamics, with object (key/value pairs) input/outputs selected by multi-head attention

Results: better ood generalization

Ongoing work: hierarchy, top-down broadcasting, spatial layout of modules



Builds on rich recent litterature on object-centric representations (mostly for images)

Modules + Global Workspace

Adding a shared global workspace similar to the GWT greatly improves RIMs

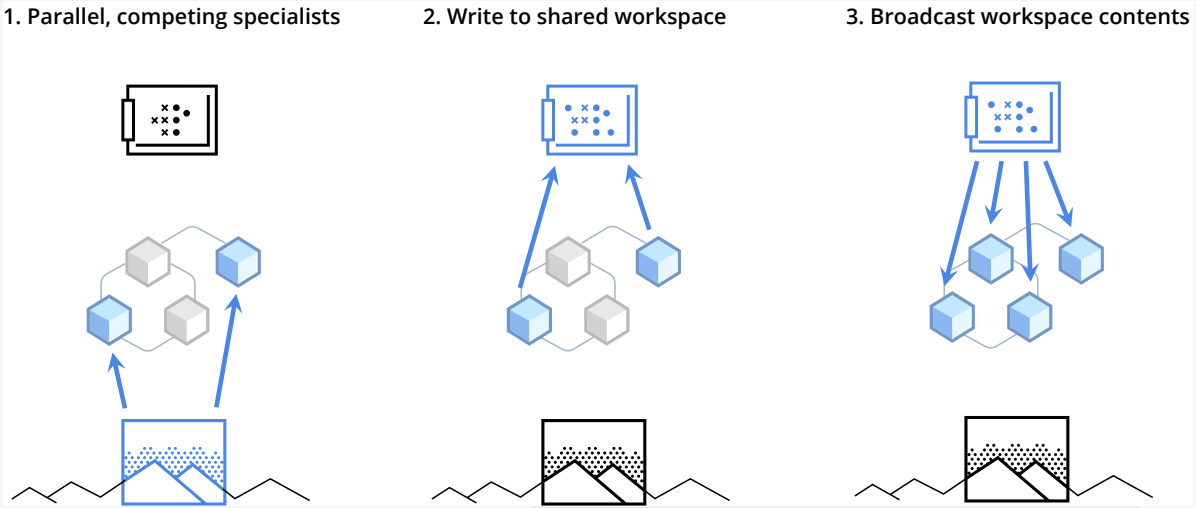
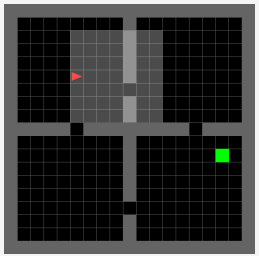
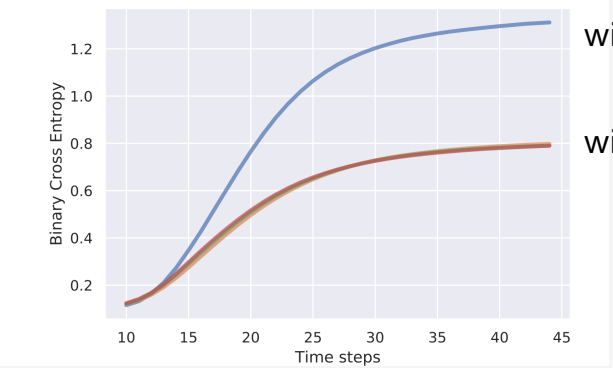


Table 2: **FourRoom Navigation Task:** Success Rate of the proposed method vs. the baselines on the FourRoom navigation environment illustrated on the right, with the agent in red, its field of visibility greyed out, and the object to get in green.

RIMs	RMC	LSTM	Ours
0.72 ± 0.02	0.67 ± 0.05	0.62 ± 0.02	0.96 ± 0.02





without GW

with GW

Tracking bouncing balls (Goyal et al 2020, submitted)

Mila

SCHEMAS AND SLOTS

Separate values (slots) from rules (schemas)

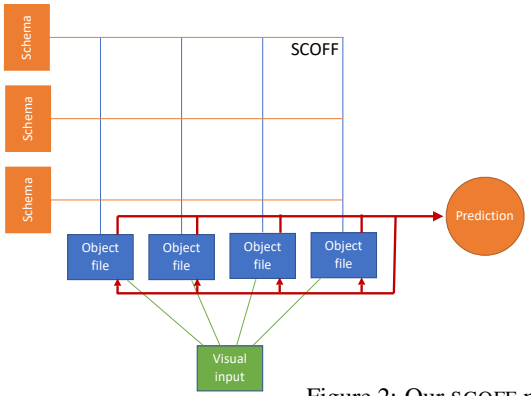


Figure 2: Our SCOFF model. Schemata are sets of parameters that specify the dynamics of objects. Object files are active modules that maintain the time-varying state of an object, seek information from the input, and select schemata for updating.

Object Files	Schema 1 Pacman	Schema 2 Normal Ghost	Schema 3 Scared Ghost
Top Frame			
A	✓		
B		✓	
C		✓	
D		✓	
E		✓	
Bottom Frame			
A	✓		
B			✓
C			✓
D			✓
E			✓

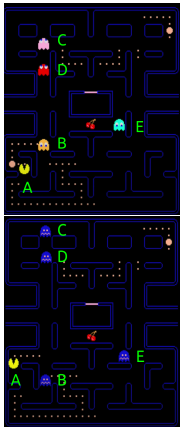


Figure 1: As a motivating example, we show two successive frames of the game PacMan and show how procedural and declarative knowledge must be dynamically factorized. The "B" ghost has a persistent object file (with its location and velocity), yet its procedure mostly depends on whether it is in its *scared* or *normal* routine.

Object Files and Schemata: factorizing declarative and procedural knowledge in dynamical systems

Goyal, Lamb, Gampa, Blundell, Mozer, Beaudoin, Levine & Bengio, submitted, 2020

SCHEMAS AND SLOTS: RESULTS

Separate values (slots) from rules (schemas)

Number of Values	*LSTM*	*RIMS*	SCOFF
2	0.8731	0.0007	0.0005
3	1.3017	0.0009	0.0007
4	1.6789	0.0014	0.0013
5	2.0334	0.0045	0.0030
8	4.8872	0.0555	0.0191
9	7.3730	0.1958	0.0379
10	11.3595	0.8904	0.0539

Table 1: **Adding Task:** Mean test set error on 200 length sequences with number of numbers to add varying among $\{2, 3, 4, 5, 8, 9, 10\}$. The models are trained to add a mixture of two and four numbers from sequences of length 50.

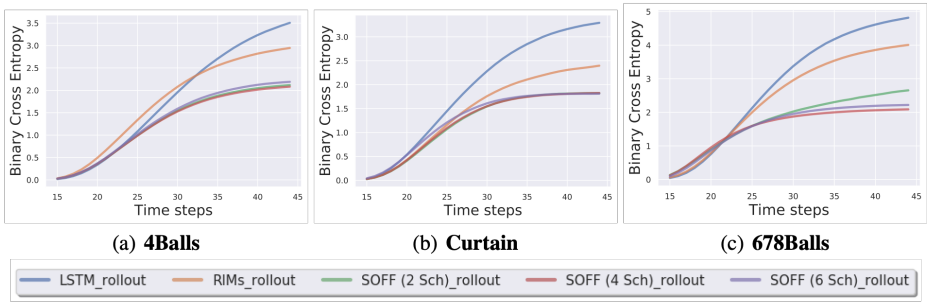
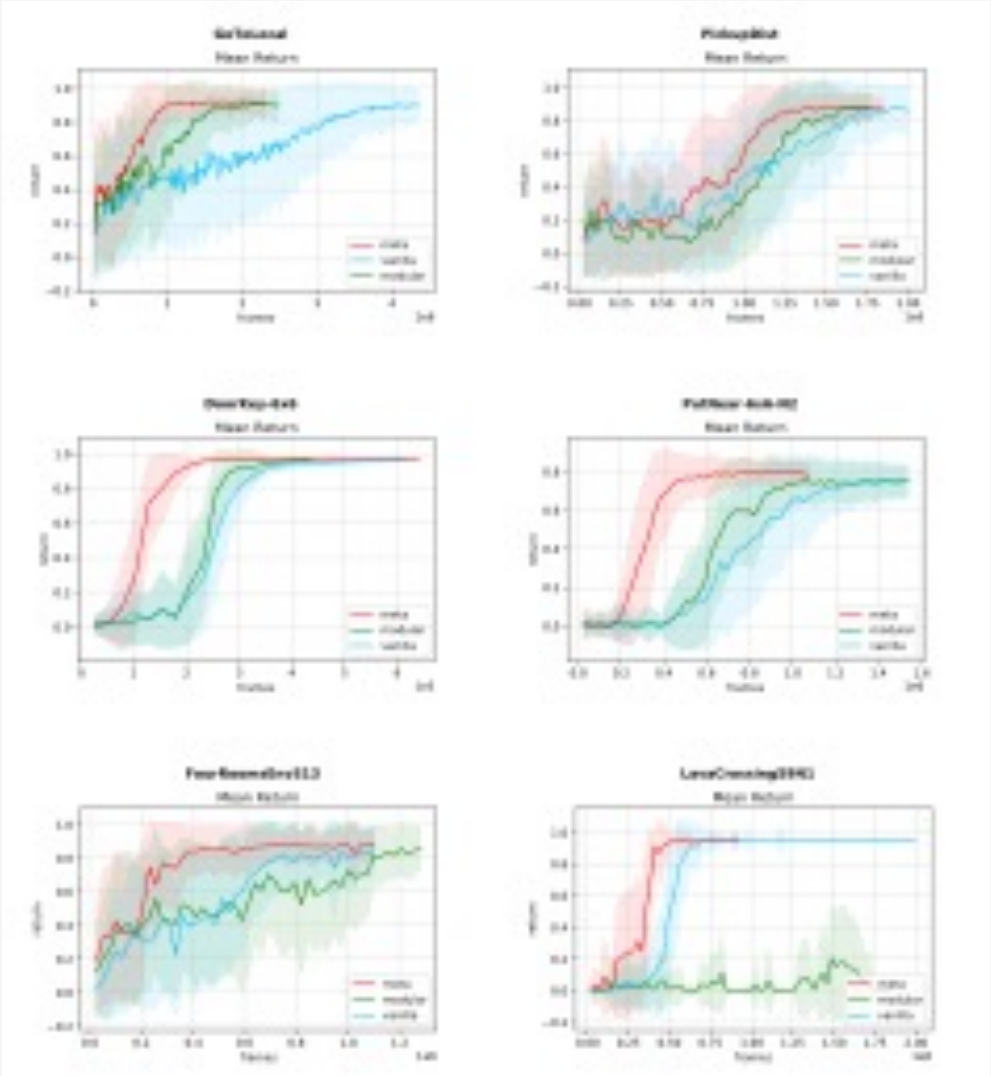


Figure 5: **Bouncing ball motion:** Prediction error comparison of SCOFF, LSTM, and RIMs. Given 10 frames of ground truth, the model predicts the rollout over the next 35 steps. SCOFF performs better than LSTM and RIMs in accurately predicting the dynamics. The advantage of SCOFF is amplified as the number of balls increases—(a) versus (c).

Experiments on Baby AI RL tasks show that slots specialize on objects (like a key) and schematas specialize on procedures (like opening a door) or object detection (like being triggered when the key is in view).

Object Files and Schemata: factorizing declarative and procedural knowledge in dynamical systems
Lamb, Goyal, Blundell, Mozer, Beaudoin, Levine & Bengio, submitted, 2020

Meta-Attention Networks



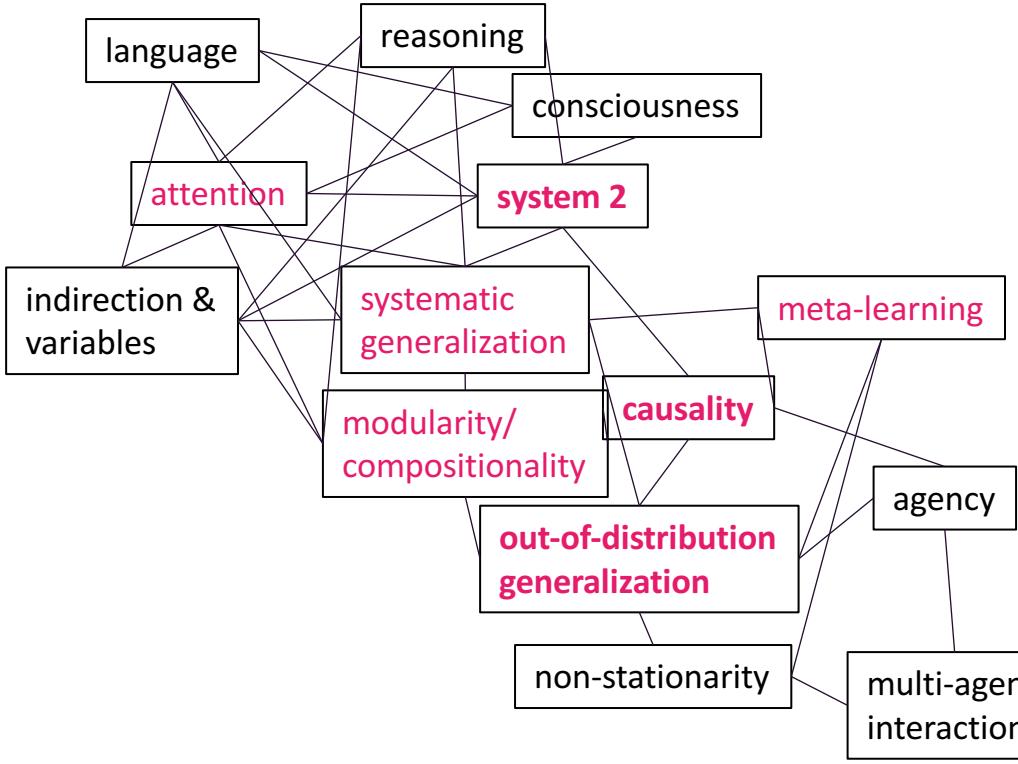
RIMs + meta-learning

Fast learning: modules

Slow learning: attention mechanism

Experiments on Baby AI tasks (Kanika Madan et al 2020, submitted)

Mila



CONSCIOUSNESS PRIORS

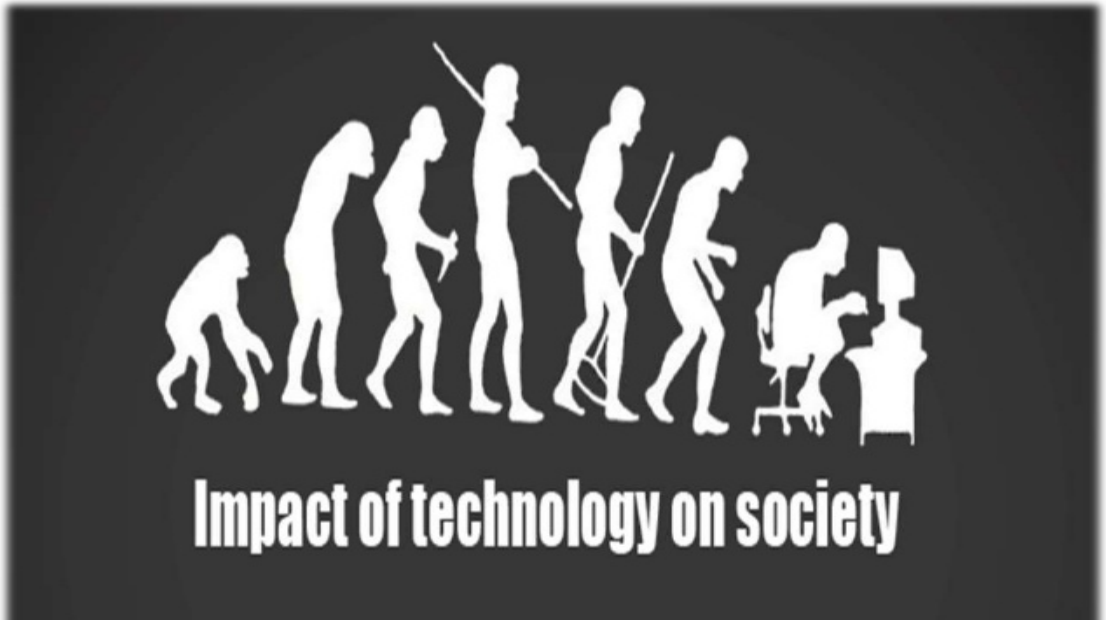
- Sparse factor graph in space of high-level semantic variables
- Semantic variables are causal: agents, intentions, controllable objects
- Simple mapping between high-level semantic variables / thoughts and words / sentences
- Shared 'rules' across instance tuples (as arguments), requiring variables & indirection
- Distributional changes due to localized causal interventions (in semantic space)
- Meaning (e.g. grounded by an encoder) is stable & robust wrt changes in distribution
- Credit assignment is only over short causal chains

We have a responsibility

ML going out of labs, into society

- ML is not just a research question anymore
- ML-based products are being designed and deployed

➔ new responsibility for AI scientists and engineers



Dangers and Concerns with AI

- Big Brother and killer robots
- Misery for jobless people, at least in transition
- Manipulation from advertising and social media
- Reinforcement of social biases and discrimination
- Increased inequality and power concentration in few people, companies & countries

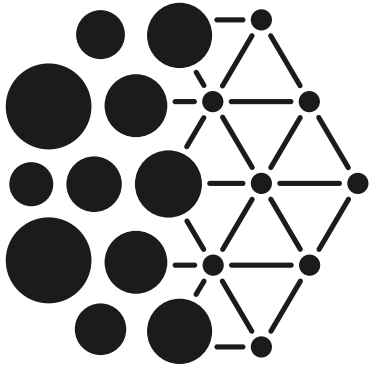
Mila

Everyone Must Benefit or Else

Think about what happens when any human (e.g. crazy person) can throw off major nuclear weapons on the face of the Earth. This is the threat of unbridled technological development.

To reap the amazing benefits of technology, we also need to increase the pace of progress of collective wisdom, manage international AI governance, make sure every human is well-fed, well-educated and healthy

Mila



Mila

Université de Montréal

McGill

Québec

CIFAR