

RECURRENT NEURAL NETWORKS



NOTATION

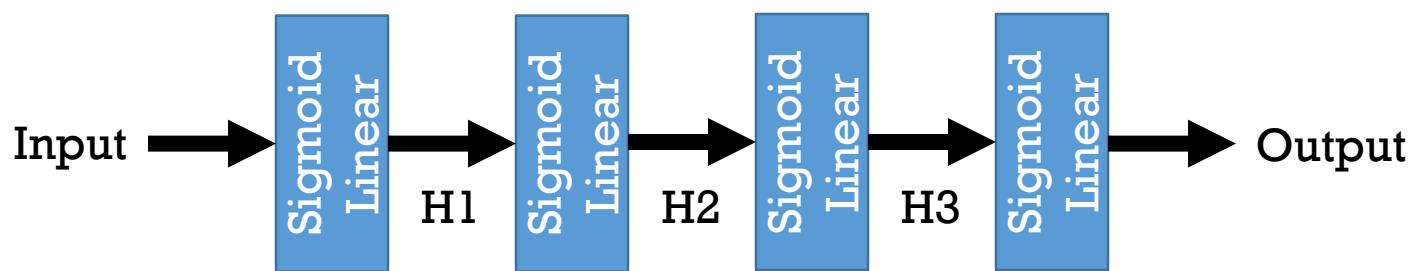


Function (Compute)



Data (Transport)

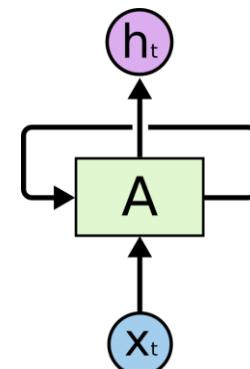
Example. “Four-Layer”(Weighted) Neural Network.



RECURRENT NEURAL NETWORKS

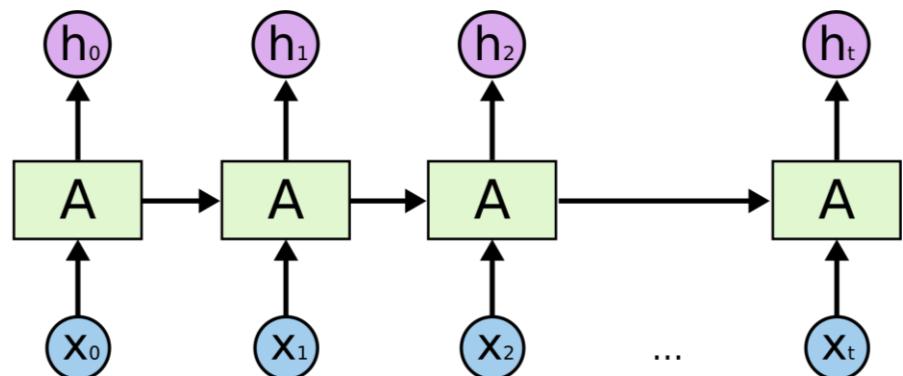
Definition.

Neural networks where the connections form a directed cycle. As opposed to feedforward neural networks.



Redefinition.

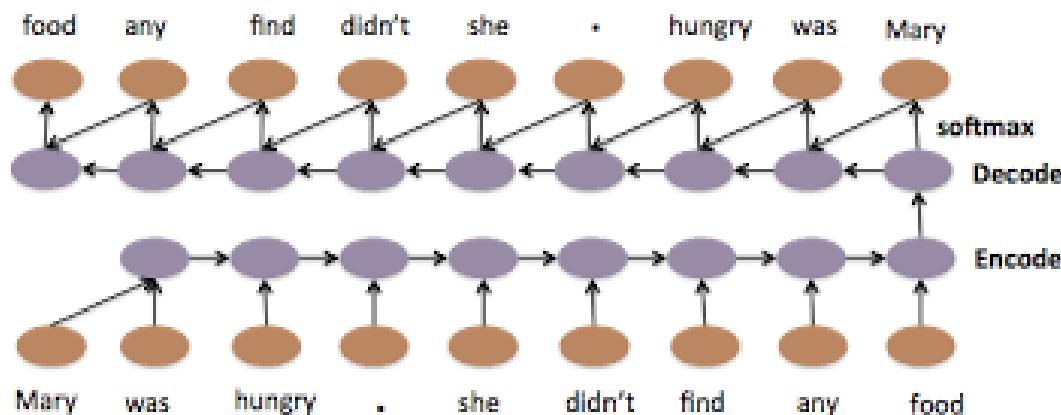
May be unrolled as a feedforward network with shared weights propagating through time.



RECURRENT NEURAL NETWORKS

Temporal State.

- RNN dynamics encode some time-varying state.
- Perfect for modeling sequences, e.g. speech recognition, natural language processing, financial modeling.



PROBLEM WITH VERY DEEP NETWORKS

Vanishing or Exploding Gradient Problem.

- Difficult to train very deep networks because of issues with backpropagation algorithm
- Error signals $\delta^{(l)}$ either shrink exponentially, or grow exponentially with the number of layers

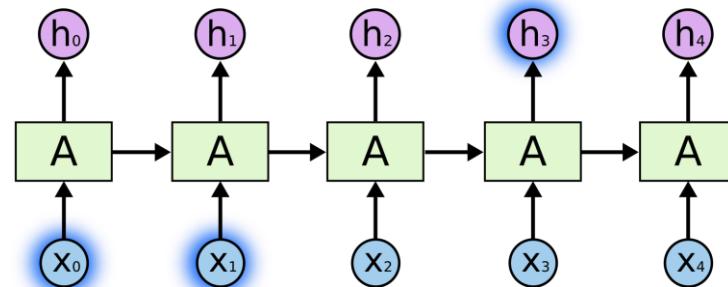
Recurrent Neural Networks are Very Deep!

- We want RNNs to learn long-term dependencies
- But for a long time, nobody knew how to train them

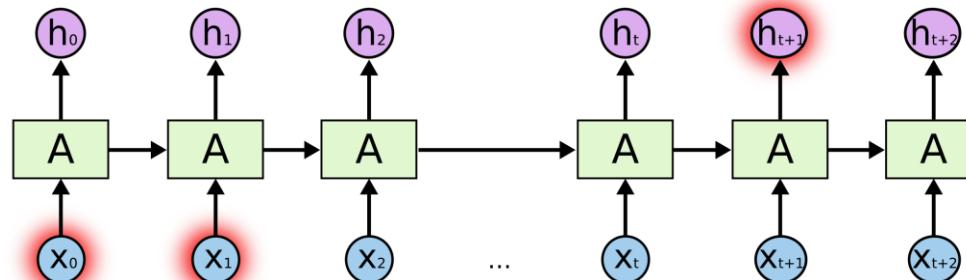


LONG-TERM DEPENDENCIES

“the clouds are in the sky”

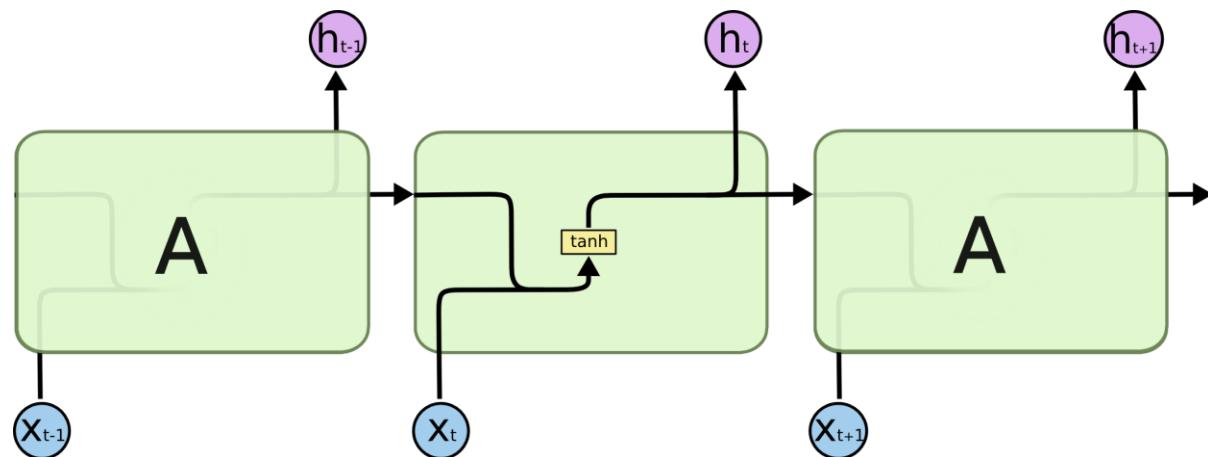


I grew up in France... I speak fluent *French*.



STANDARD RNN

Single Layer RNN.



Neural Network Layer

Pointwise Operation

Vector Transfer

Concatenate

Copy



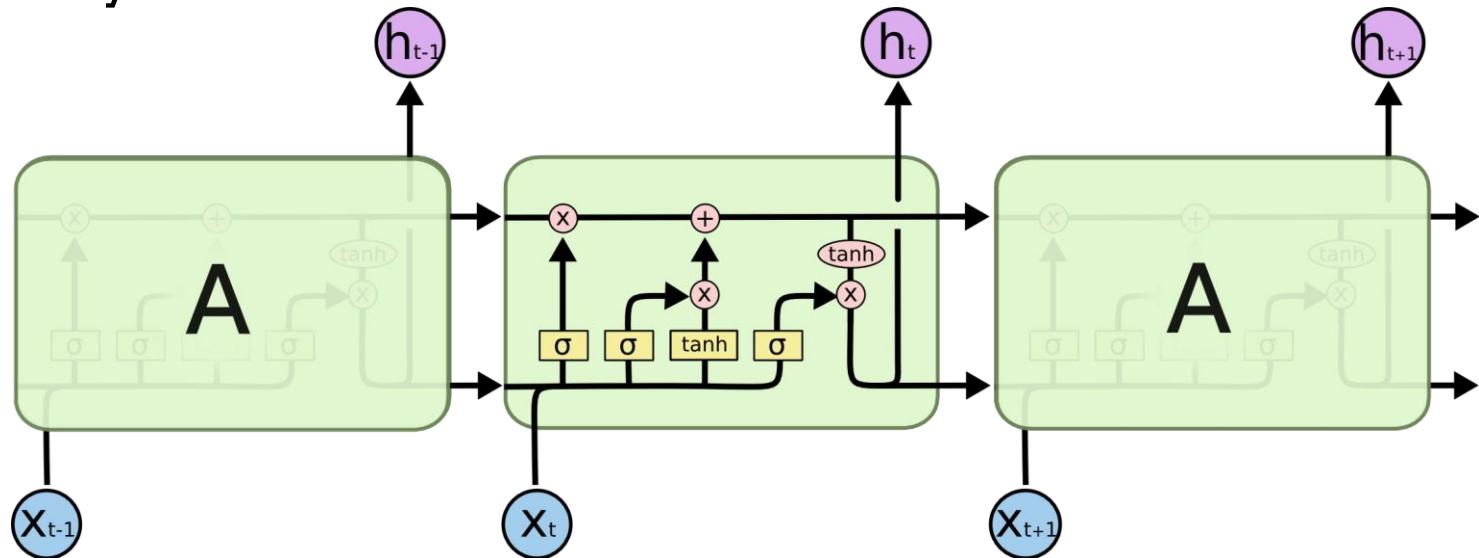


LONG SHORT TERM MEMORY (LSTM)



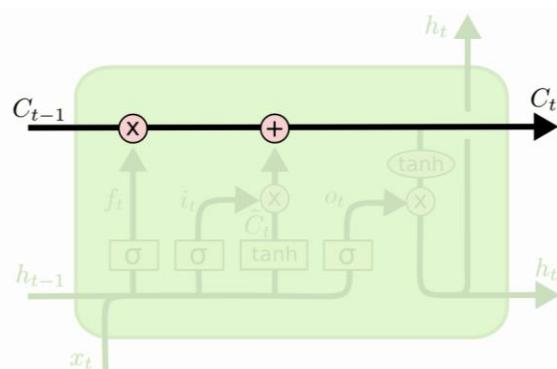
LONG SHORT TERM MEMORY

Four-Layer RNN.

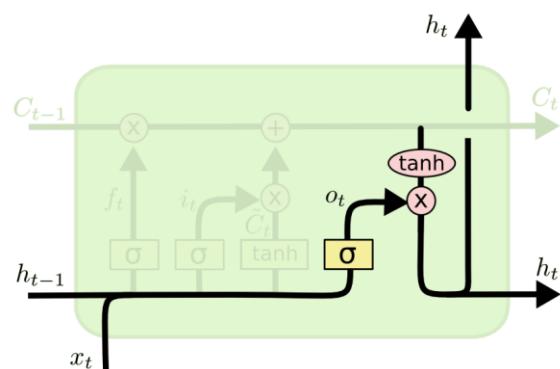


STATES AND GATES

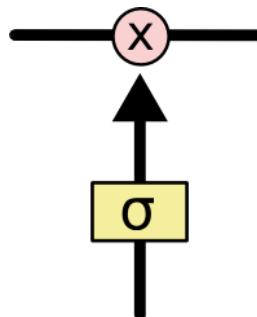
Cell State



Output State



Gates

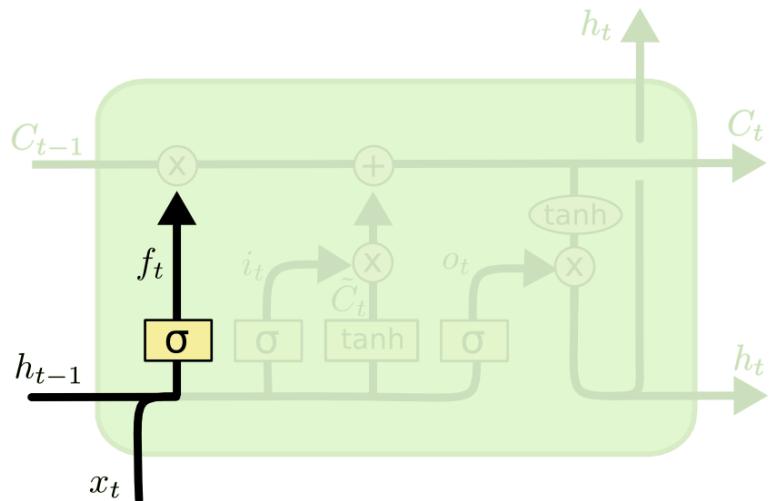


Output of sigmoid
is between 0 and 1
0 – forget past state
1 – keep past state



LAYER 1 – FORGET PAST STATE

Example. Forget gender of subject when subject changes.



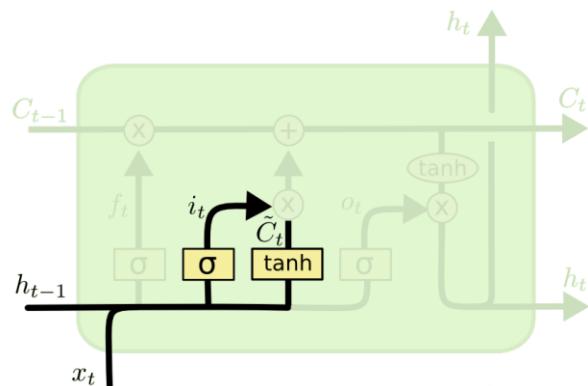
How much to forget?

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$



LAYER 2,3 – ADD NEW INFORMATION

Example. Add gender of new subject.

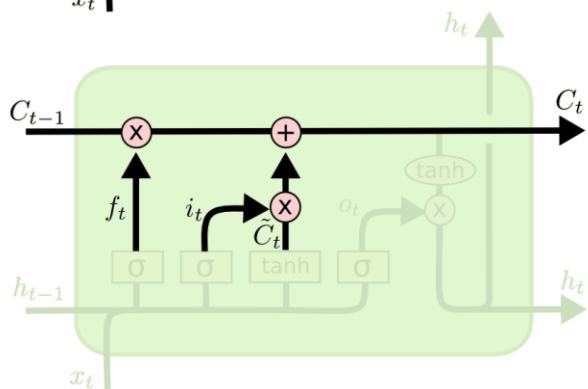


How much to add?

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

What info to add?

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$



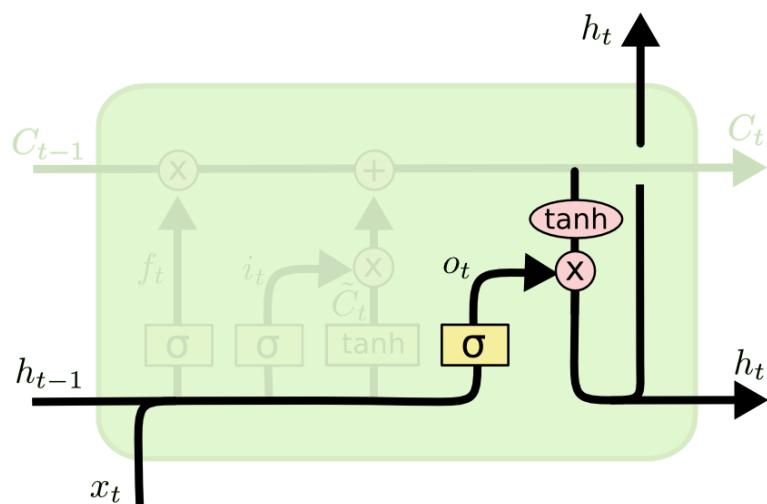
New cell state.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



LAYER 4 – SELECTIVE OUTPUT

Example. Singular or plural form of verb? Output and forward.



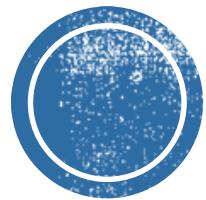
How much to output?

$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

What to output?

$$h_t = o_t * \tanh (C_t)$$





ASSOCIATIVE MEMORIES



ENERGY-BASED LEARNING

- Discrete model – states x , parameters w
- Probabilities defined in terms of *energy* $f(x|w)$

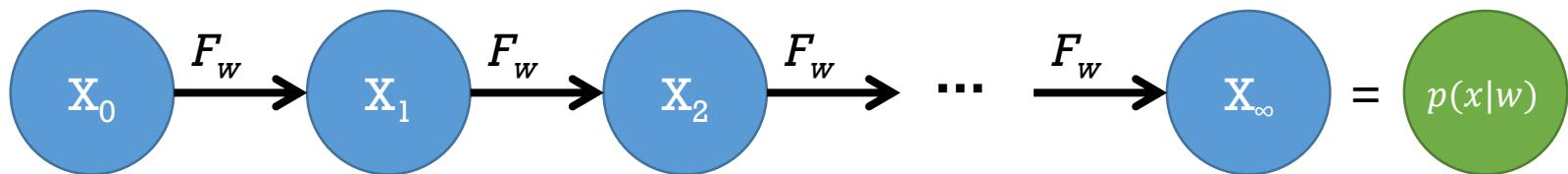
$$p(x|\omega) = \frac{e^{-f(x|\omega)}}{Z(\omega)} \quad \text{where } Z(\omega) = \sum_x e^{-f(x|\omega)}$$

- **Energy wells** represent likely states of the model



ENERGY-BASED LEARNING

- Partition function $Z(w)$ often difficult to compute, but it is easy to sample from model distributions using MCMC techniques such as Gibbs sampling
- Let $X_0 = (x_1, x_2, \dots, x_N)$ denote the data
- Given parameter w , let F_w be a corresponding MCMC update that acts on the state space



- Loosely, the X_i converges to the model distribution $p(x|w)$



ISING MODEL (1924)

Magnetic spins on a lattice.

- Energy of a configuration σ

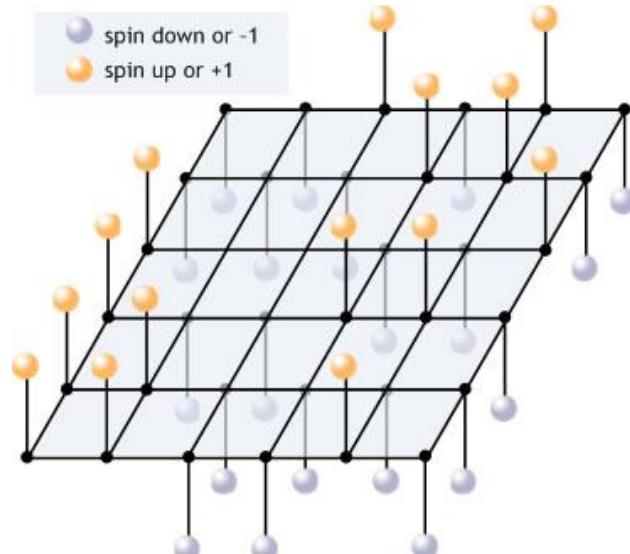
$$H(\sigma) = - \sum_{\langle i j \rangle} J_{ij} \sigma_i \sigma_j - \mu \sum_j h_j \sigma_j$$

- Probability of observing σ

$$P_\beta(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z_\beta}$$

- Normalization constant

$$Z_\beta = \sum_{\sigma} e^{-\beta H(\sigma)}$$



HOPFIELD NETWORKS (1982)

Using Ising Model to mimic neural networks.

- Energy

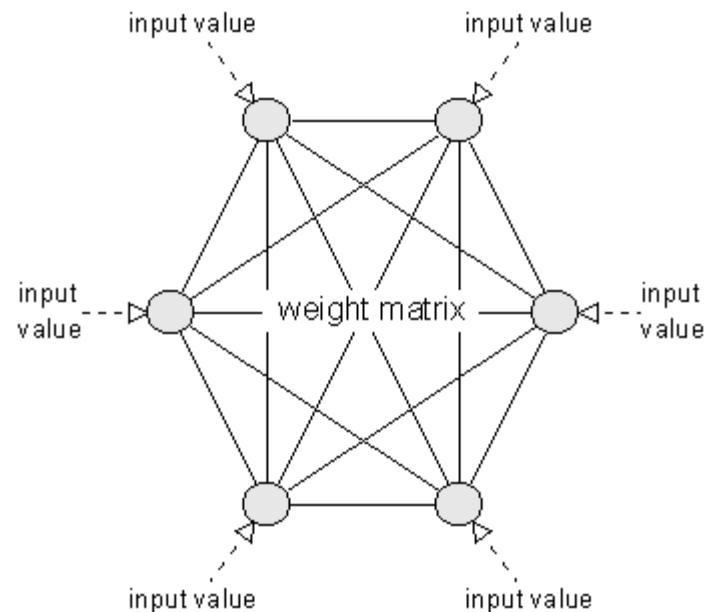
$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

- Update Rule

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

- Learning Rule (Hebbian)

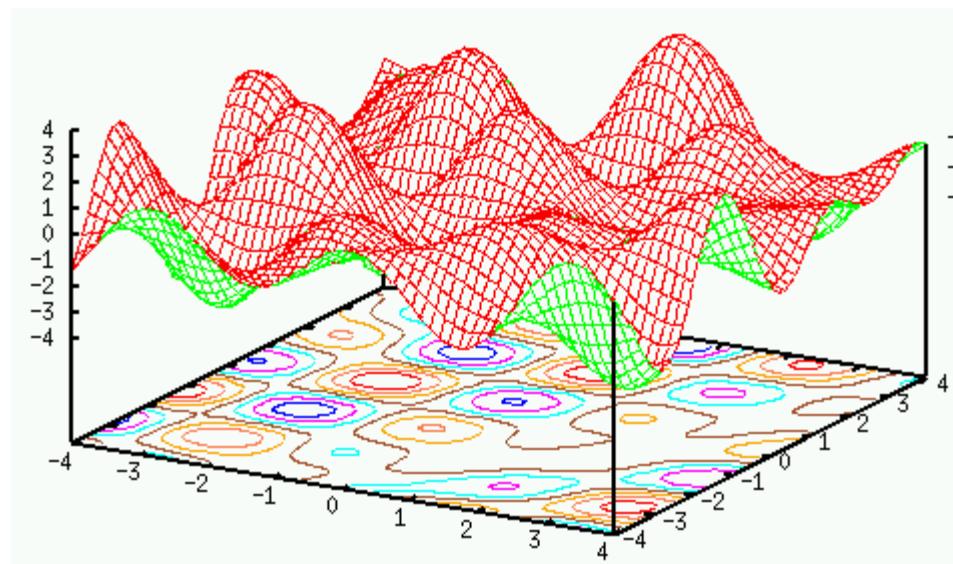
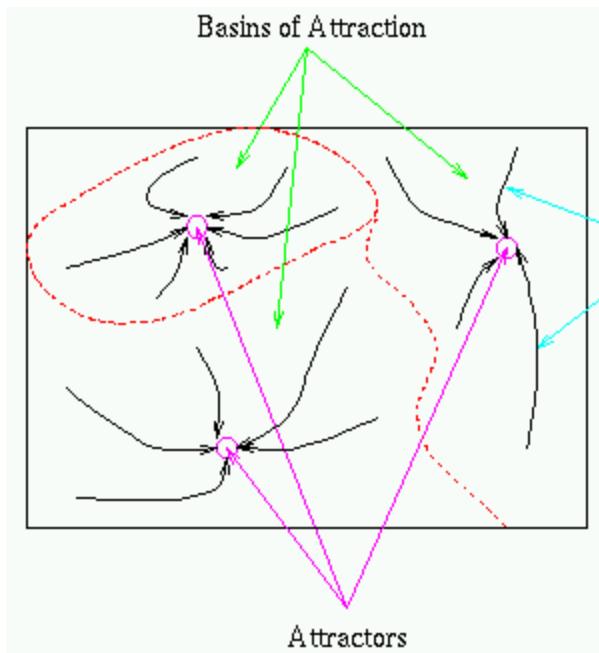
$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^n \epsilon_i^\mu \epsilon_j^\mu$$



CONVERGENCE

Theorem. The update rule always converges.

Proof. The energy decreases with each update.



BOLTZMANN MACHINES (1985)

Stochastic version of Hopfield networks.

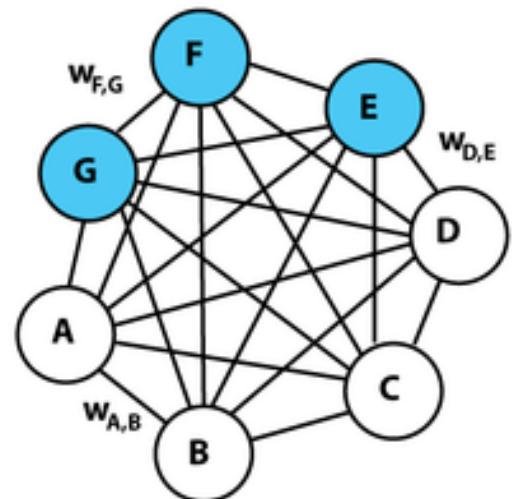
- Energy

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

- Gibbs sampling

$$\mathbb{P}(x_i = 1 | x_{-i}) = \text{sigmoid}\left(\sum_{j \neq i} w_{ij} x_j + b_i\right)$$

- Learning (typically not easy or too slow)



RESTRICTED BOLTZMANN MACHINES

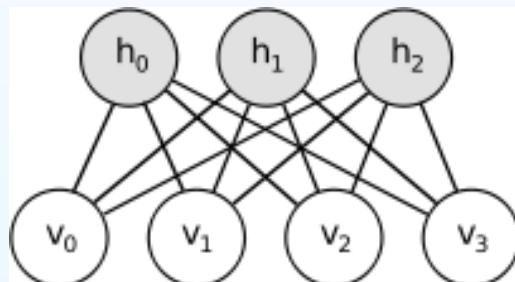
- Binary-valued neurons (Geoffrey Hinton)

Undirected discrete graphical model with binary states.

Graph G is bipartite with two layers: observed and hidden nodes.

Parameters: weights ω_{ij} for each edge (h_i, v_j) ,

bias b_i for each h_i , bias c_j for each v_j .



$$p(x|W, b) \propto e^{-f(x|W, b)}$$

$$f(x|W, b) = \sum_{i < j} W_{ij}x_i x_j + \sum_i b_i x_i$$

$$\mathbb{P}(h_i = 1|v) = \text{sig}(b_i + \sum_j \omega_{ij}v_j), \quad \text{sig}(x) = \frac{1}{1 + e^{-x}}$$



CONTRASTIVE DIVERGENCE

- MLE minimizes

$$\ell(\omega) = \log Z(\omega) + \frac{1}{N} \sum_i f(x_i|\omega)$$

- Gradient

$$\begin{aligned}\frac{\partial \ell}{\partial \omega} &= \mathbb{E}_{X_0} \left(\frac{\partial f(x|\omega)}{\partial \omega} \right) + \frac{\partial \log Z(\omega)}{\partial \omega} \\ &= \mathbb{E}_{X_0} \left(\frac{\partial f(x|\omega)}{\partial \omega} \right) - \mathbb{E}_{X_\infty} \left(\frac{\partial f(x|\omega)}{\partial \omega} \right)\end{aligned}$$

- CD relaxes it to

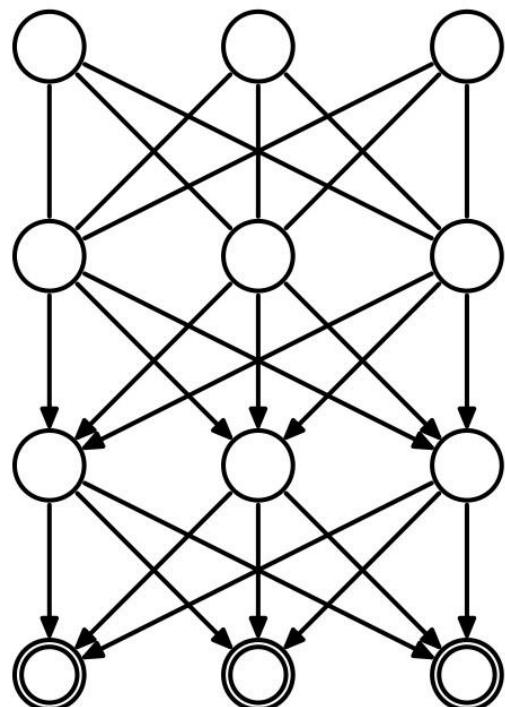
$$\frac{\partial \ell}{\partial \omega} = \mathbb{E}_{X_0} \left(\frac{\partial f(x|\omega)}{\partial \omega} \right) - \mathbb{E}_{X_1} \left(\frac{\partial f(x|\omega)}{\partial \omega} \right)$$

- This relaxation seems to work well in experiments.

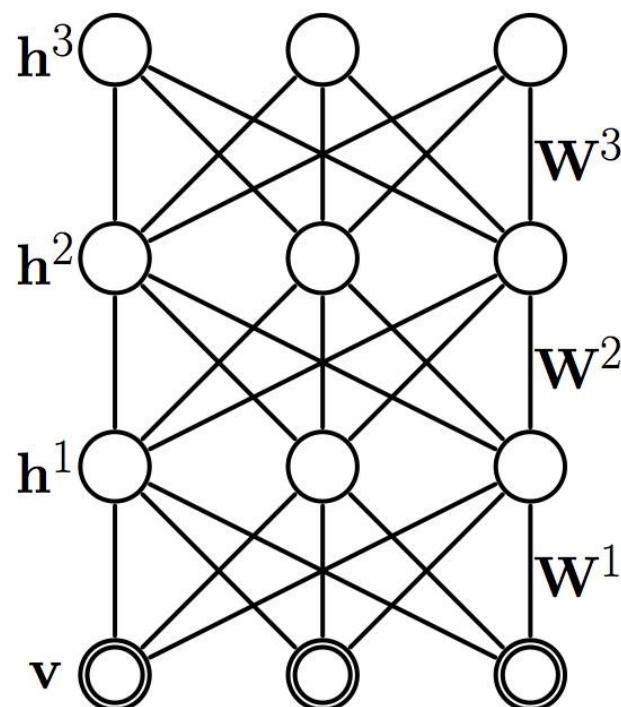


DEEP BELIEF NETS

Deep Belief Network



Deep Boltzmann Machine





HISTORY



RIDICULOUSLY SIMPLIFIED HISTORY

1943 Artificial Neuron (McCullough, Pitts)

1957 Perceptrons (Rosenblatt)

1969 Problem with XOR (Minsky, Papert)

FIRST AI WINTER

1986 Backpropagation (Rumelhart, Hinton, Williams)

1989 Convolutional Neural Nets (LeCun)
Autoencoders, Belief Nets, Recurrent NN, Reinforcement

1995 Problems with Backprop.
Rise of SVMs and Random Forests.

SECOND AI WINTER



DEEP LEARNING CONSPIRACY



Yann LeCun,
Geoffrey Hinton,
Yoshua Bengio,
Andrew Ng

- 2006 Greedy Initialization of Layers
- 2009 Graphics Processing Units
- 2012 Dropout (ImageNet)



WHAT WAS WRONG WITH BACKPROPAGATION IN 1986?



1. Our labeled datasets were thousands of times too small.
2. Our computers were millions of times too slow.
3. We initialized the weights in a stupid way.
4. We used the wrong type of non-linearity.

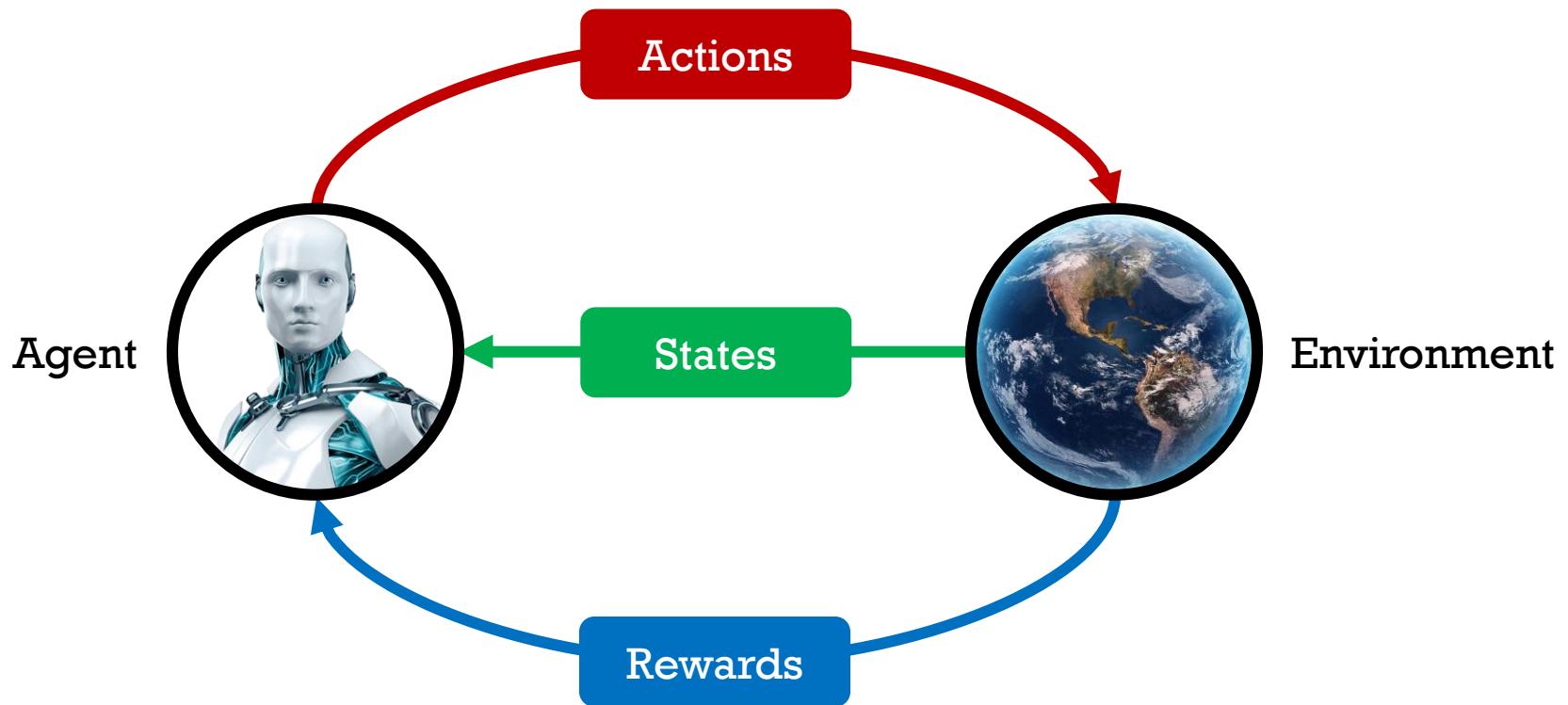




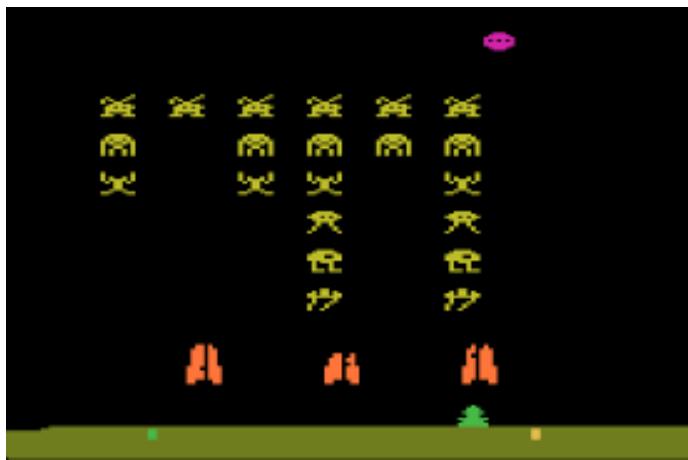
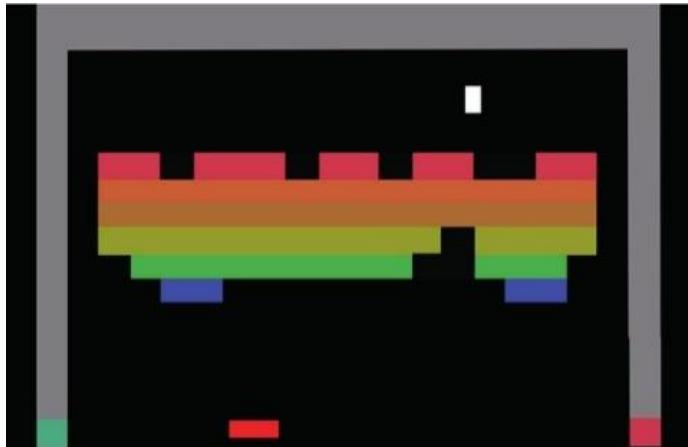
FUTURE



REINFORCEMENT LEARNING



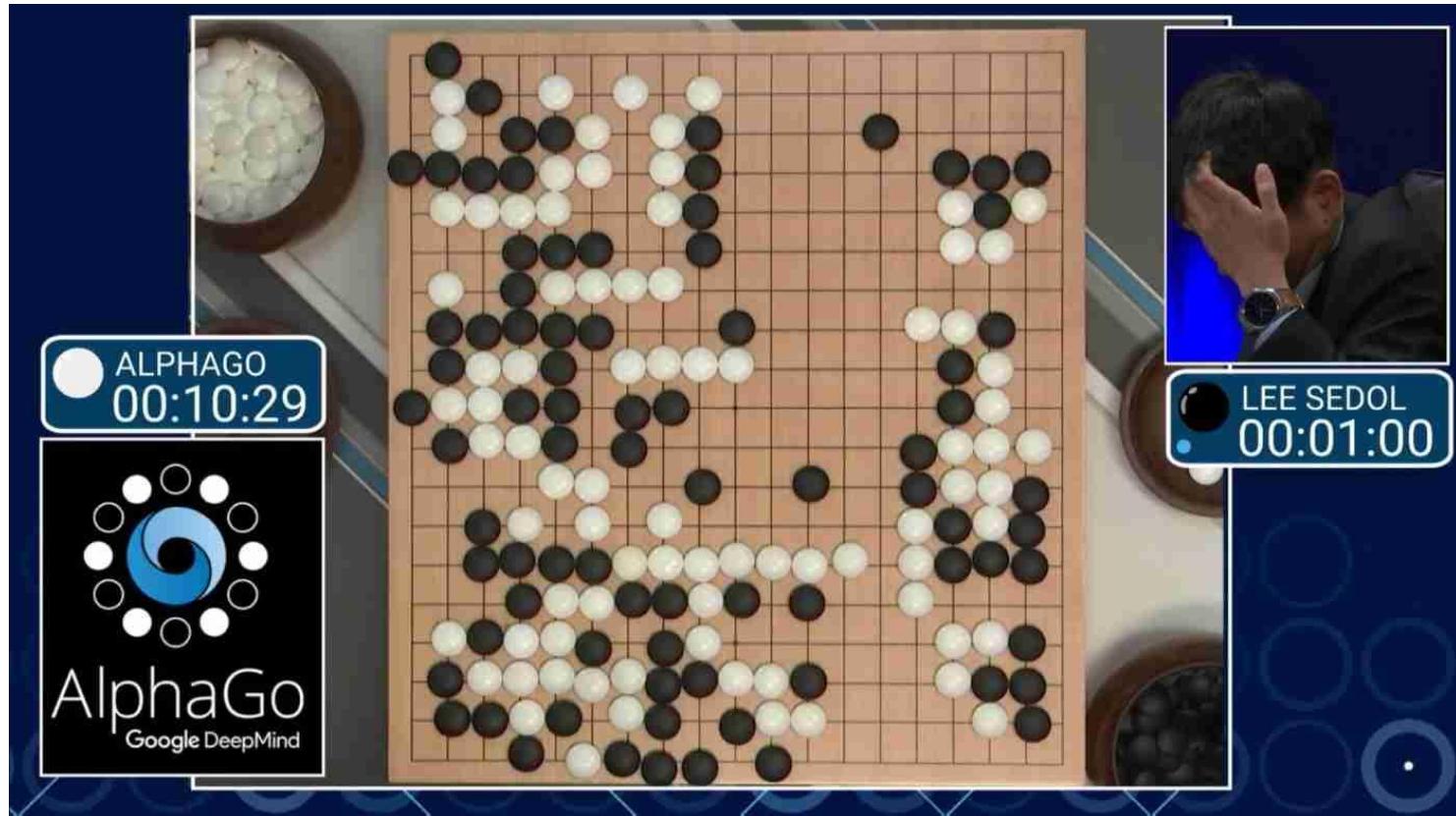
ATARI GAMES



Google DEEPMIND



ALPHAGO

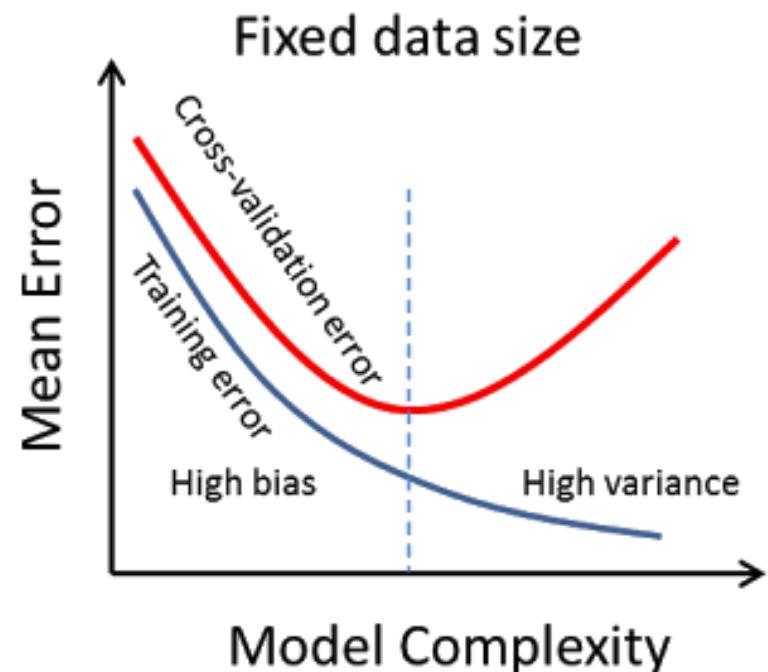


DRIVE.AI

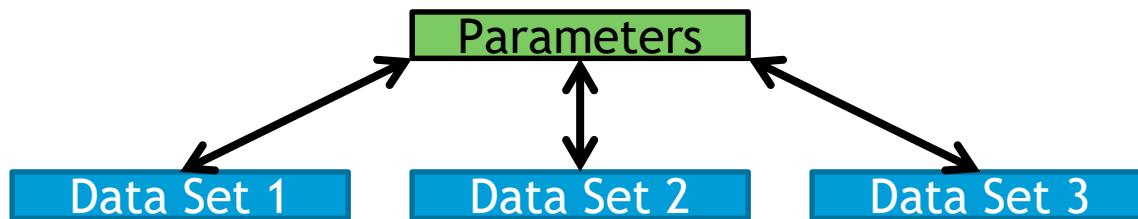


HYPERPARAMETER CHALLENGE

- Number of layers
- Number of neurons in each layer
- Parameter initialization
- Weight decay parameter
- Sparsity parameter
- Learning rate
- ...



HARDWARE ACCELERATION



Costly to train massive neural networks



Yann LeCun, NYU

“What we’re missing is the ability to parallelize the training of the network, mostly because the **communication is the killer**.”

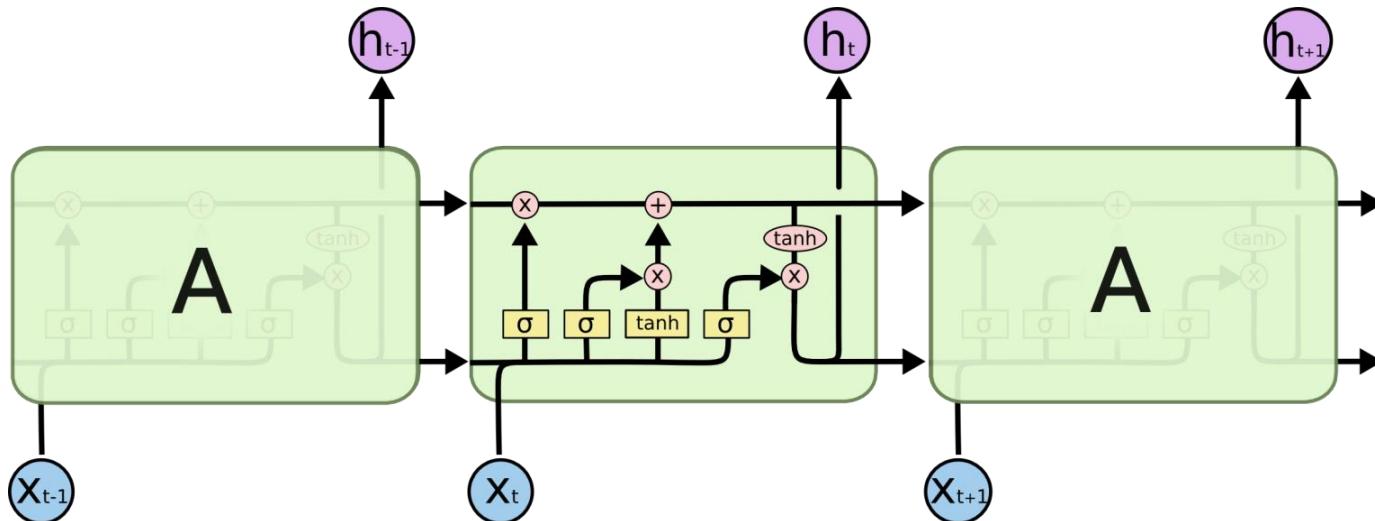
“A lot of people are trying to solve this problem at Baidu, Facebook, Google, Microsoft and elsewhere [...] it is **both a hardware and software issue**.”

“Eventually a lot of the deep learning task will be done on the device, which will keep pushing the need for **on-board neural network accelerators** that operate at very low power and can be trained and used quickly.”

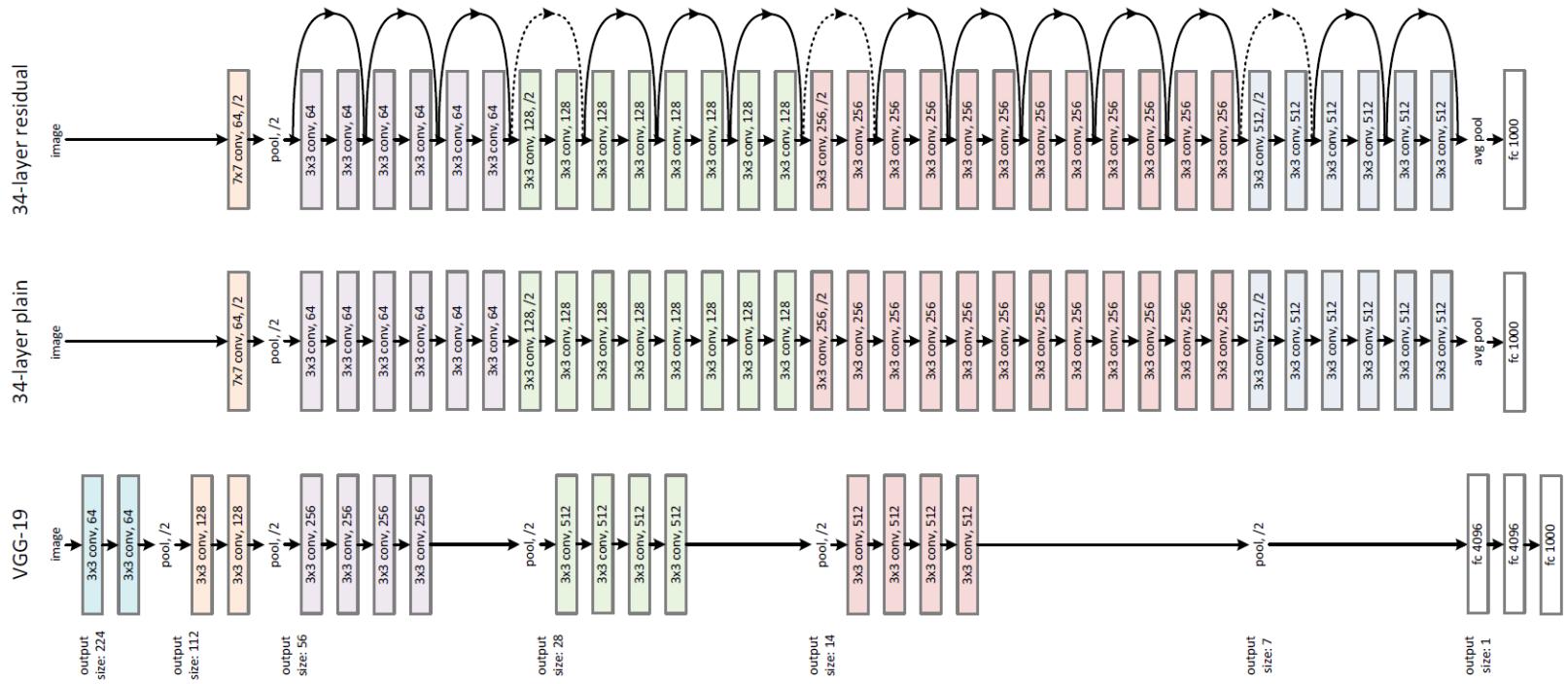


RECURRENT NEURAL NETWORKS

Long Short Term Memory (1997 Schmidhuber, Hochreiter)



VERY DEEP NETWORKS



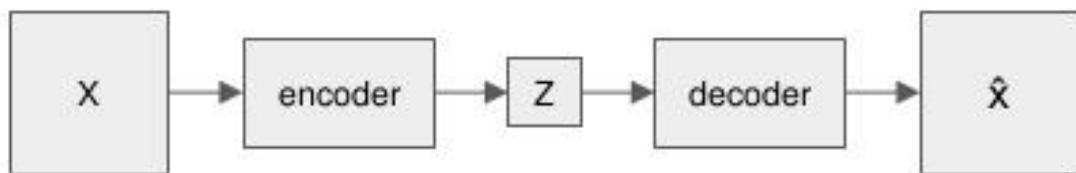
2014 **VGG Nets.**

2015 **Deep Residual Learning.**

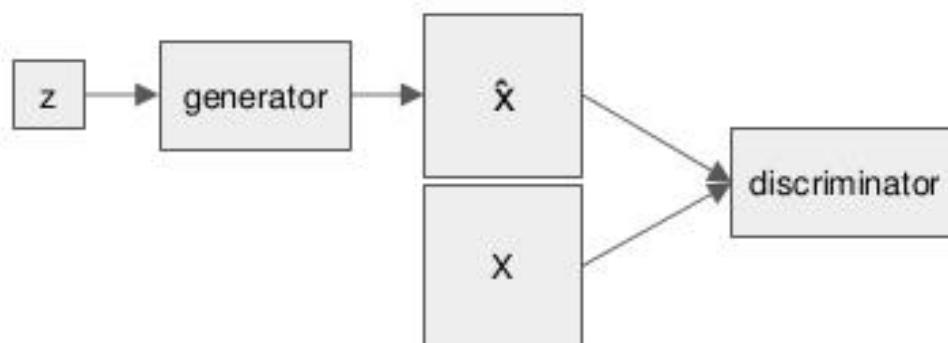


UNSUPERVISED LEARNING

Two promising approaches

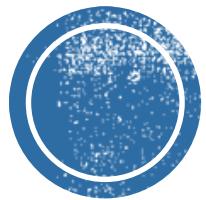


Variational
Autoencoders (VAE)
[Kingma and Welling
\[1312.6114\]](#)



Generative Adversarial
Networks (GAN)
[Goodfellow et al. \[1406.2661\]](#)

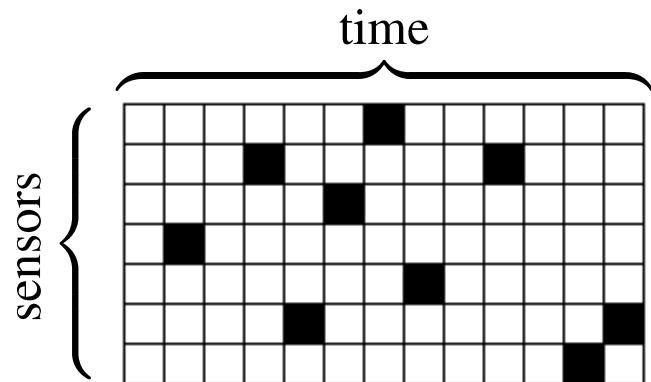




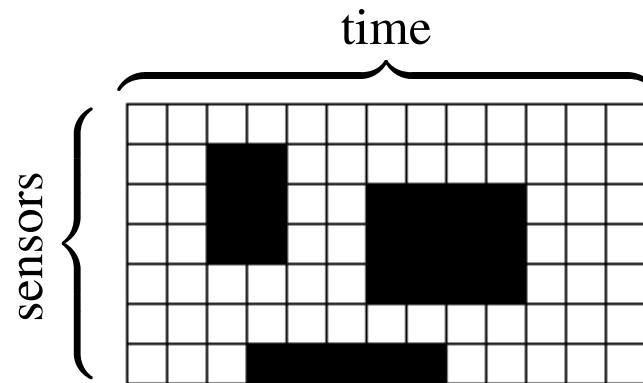
APPLICATIONS



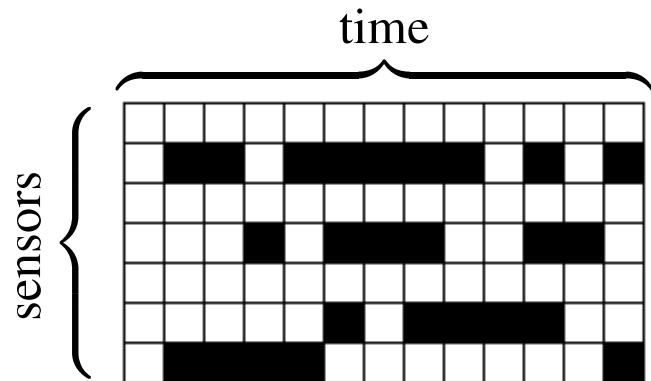
STRUCTURED MISSING DATA



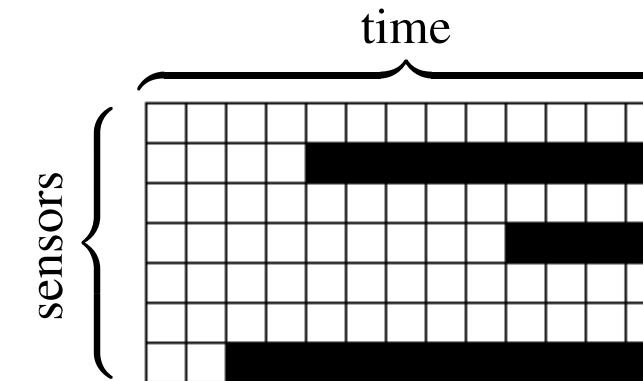
(a) Random Loss



(b) Block Random Loss



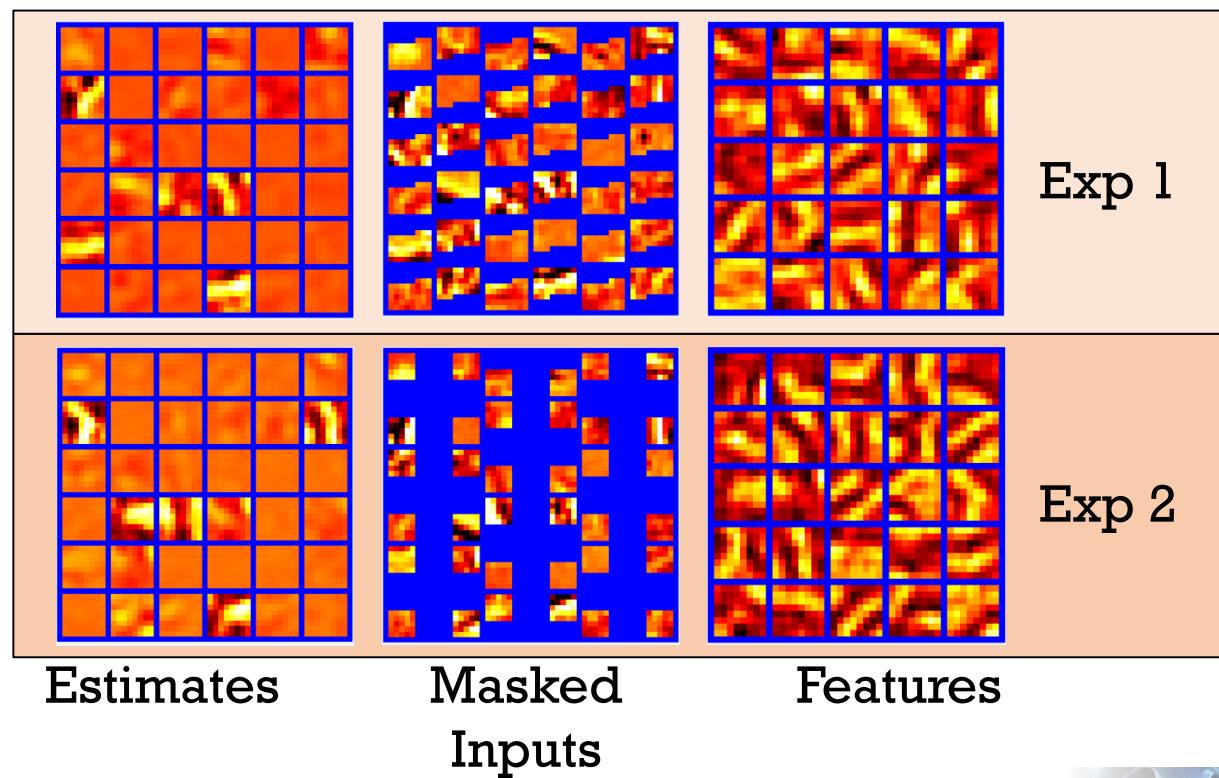
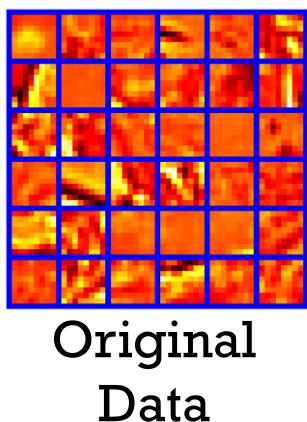
(c) Frequent Loss in Row



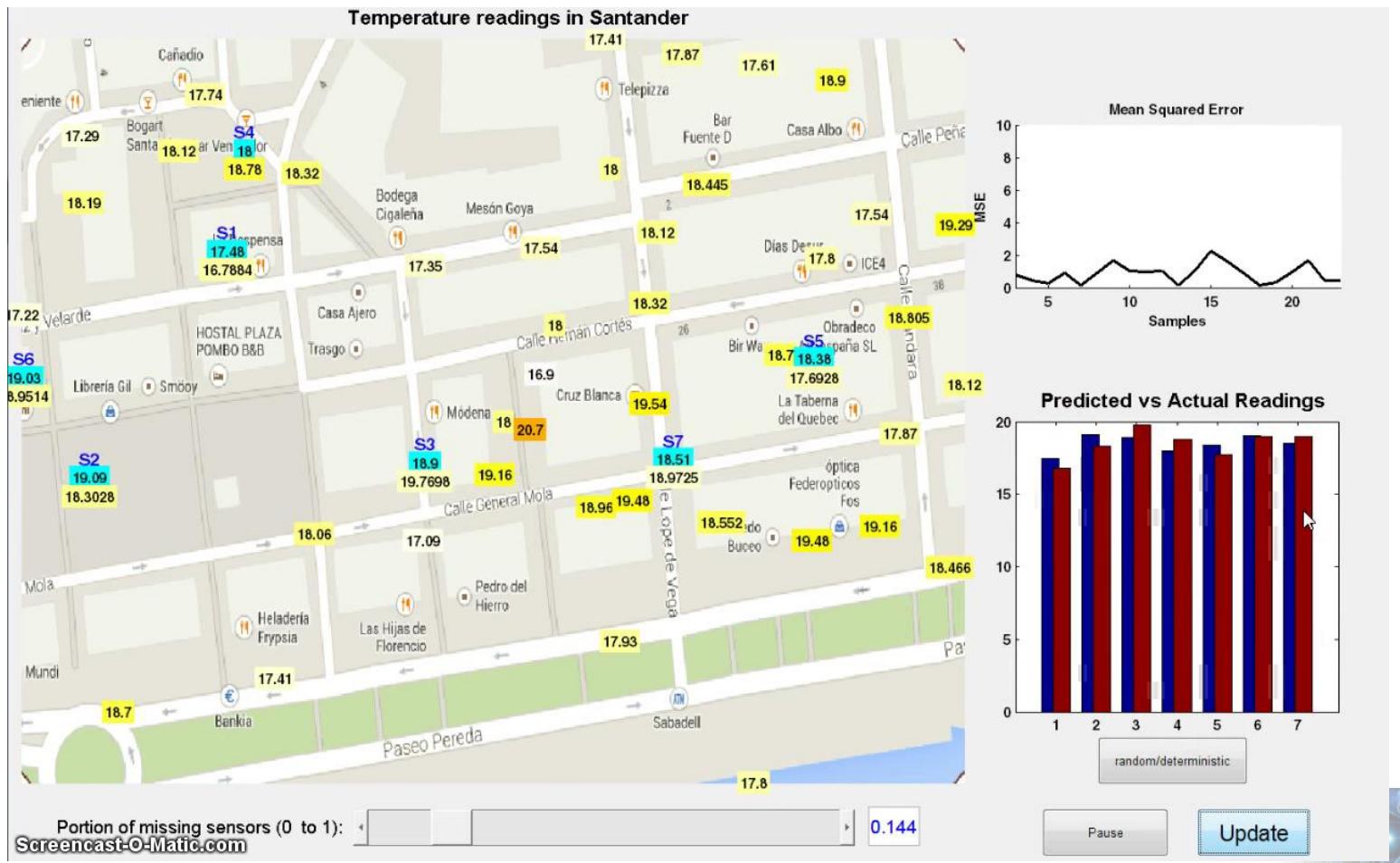
(d) Successive Loss



MISSING DATA IMPUTATION



MISSING DATA IMPUTATION



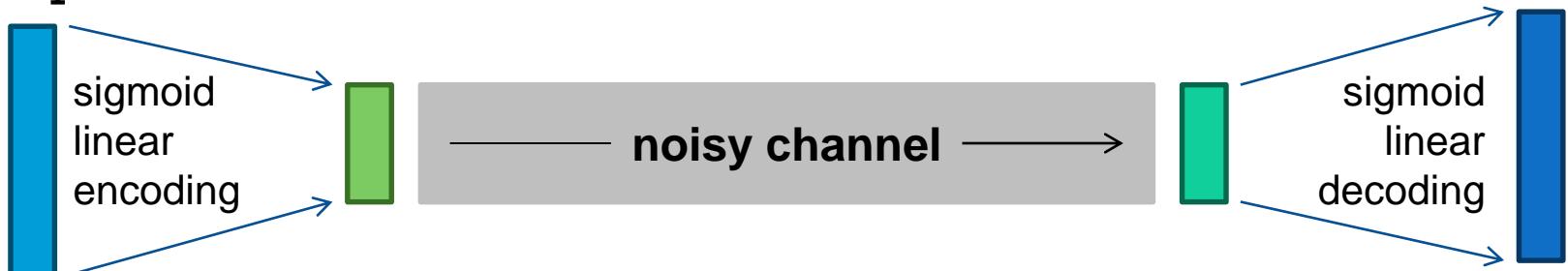
DATA COMPRESSION

- Compressed Sensing



Training: sparse coding (learning a sparse linear dictionary)

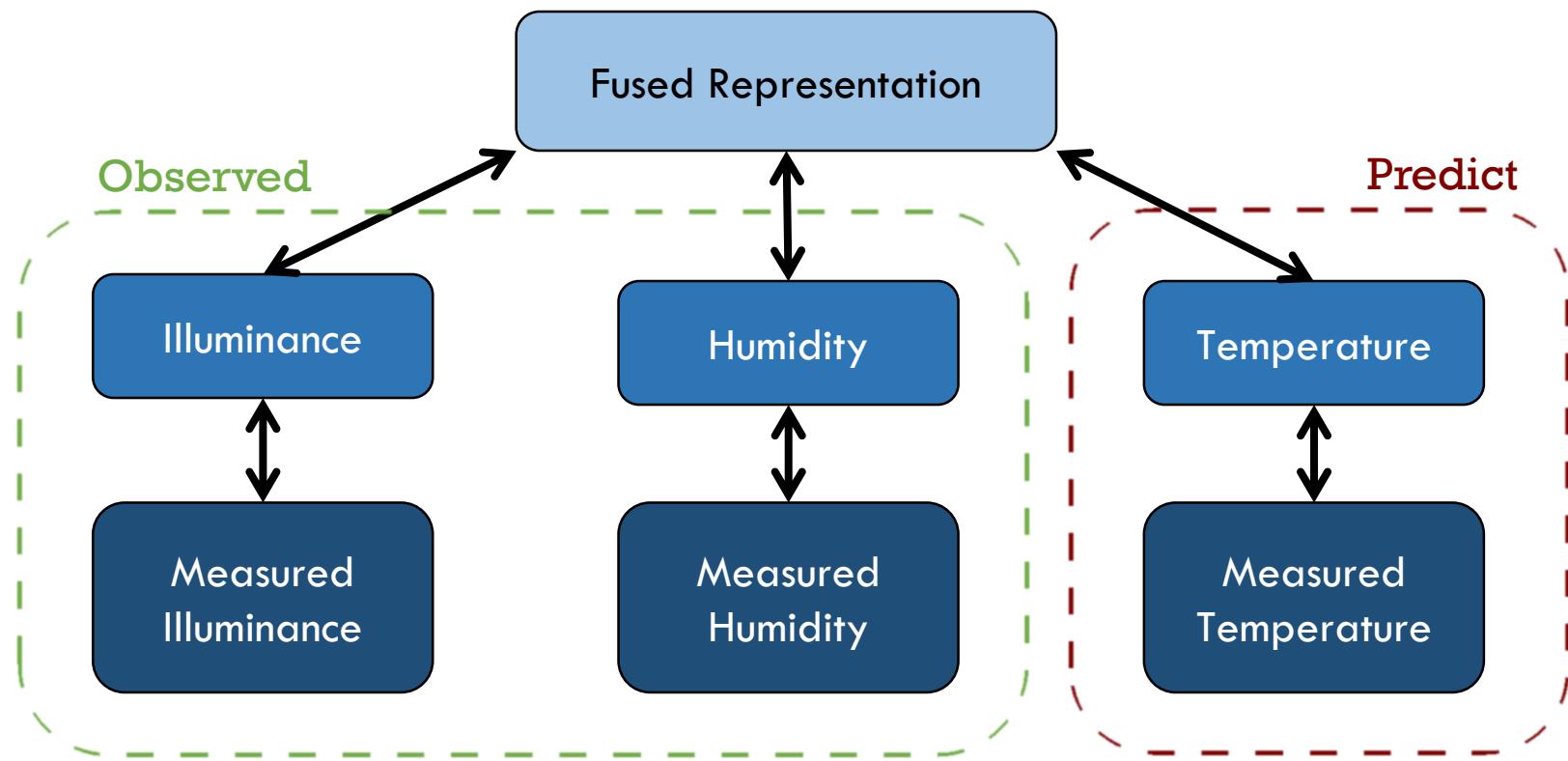
- Sparse Autoencoder



Training: sparse autoencoder (learning a coding-decoding function)



MULTIMODAL DATA FUSION



MULTIMODAL DATA FUSION

