IAS
2019-10-16

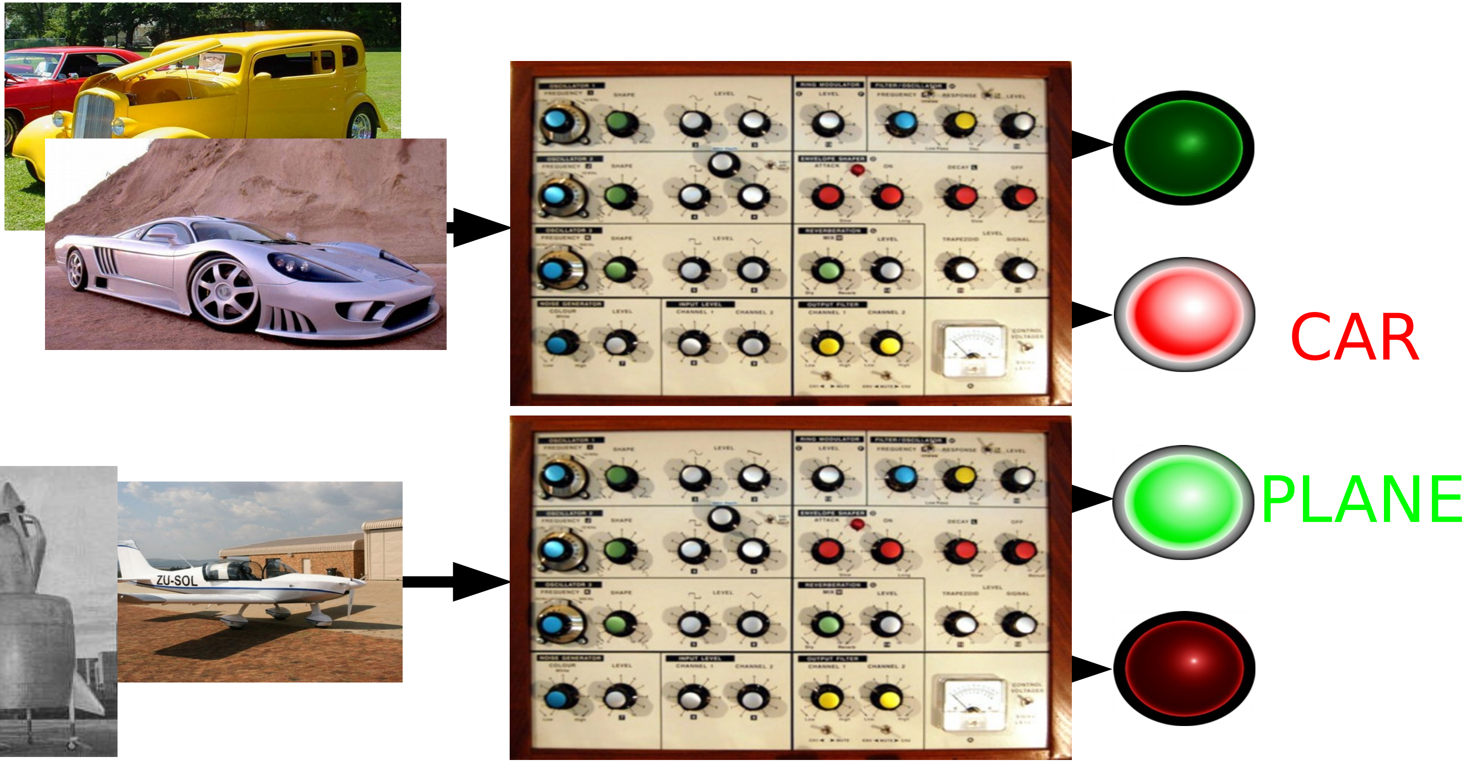
Energy-Based Approaches To Representation Learning

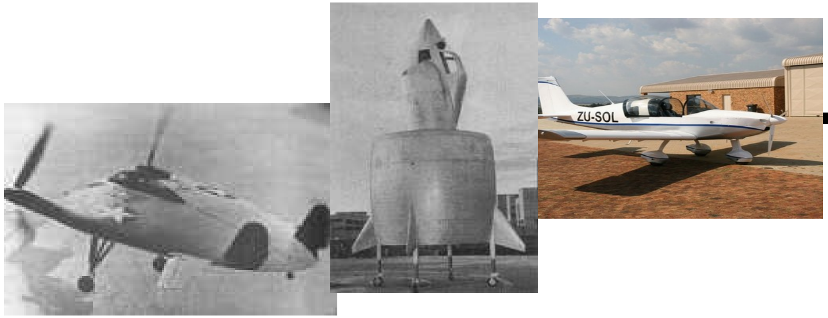
Yann LeCun
New York University
Facebook AI Research
http://yann.lecun.com

facebook
Artificial Intelligence Research

Supervised Learning works but requires many labeled samples

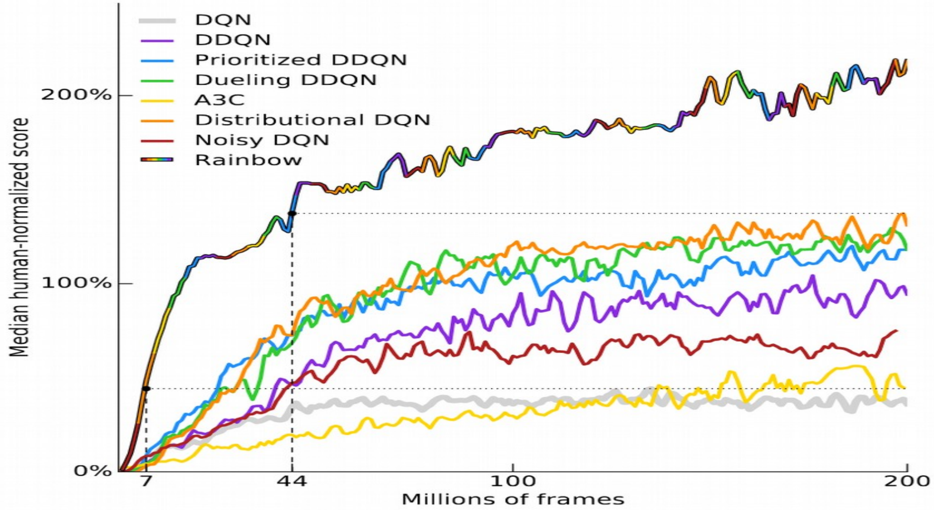
- **Training a machine by showing examples instead of programming it**
- **When the output is wrong, tweak the parameters of the machine**
- **Works well for:**
  - Speech→words
  - Image→categories
  - Portrait→ name
  - Photo→caption
  - Text→topic
  - ….

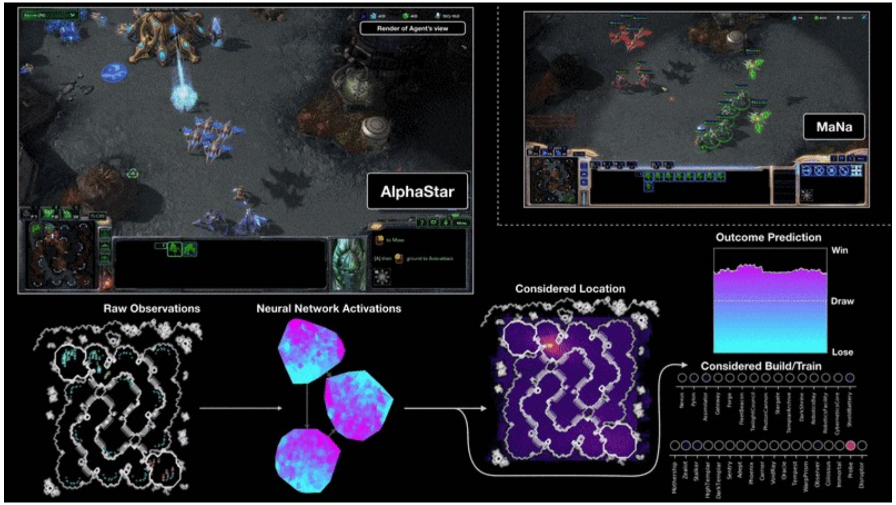




Reinforcement Learning: works great for games and simulations.

- **57 Atari games: takes 83 hours equivalent real-time (18 million frames) to reach a performance that humans reach in 15 minutes of play.**
  - [Hessel ArXiv:1710.02298]
- **Elf OpenGo v2: 20 million self-play games. (2000 GPU for 14 days)**
  - [Tian arXiv:1902.04522]
- **StarCraft: AlphaStar 200 years of equivalent real-time play**
  - [Vinyals blog post 2019]
- **OpenAI single-handed Rubik's cube**
  - 10,000 years of simulation





But RL Requires too many trials in the real world

- **Pure RL requires too many trials to learn anything**
  - it's OK in a game
  - it's not OK in the real world
- **RL works in simple virtual world that you can run faster than real-time on many machines in parallel.**

- **Anything you do in the real world can kill you**
- **You can't run the real world faster than real time**

How do Humans and Animal Learn?

So quickly

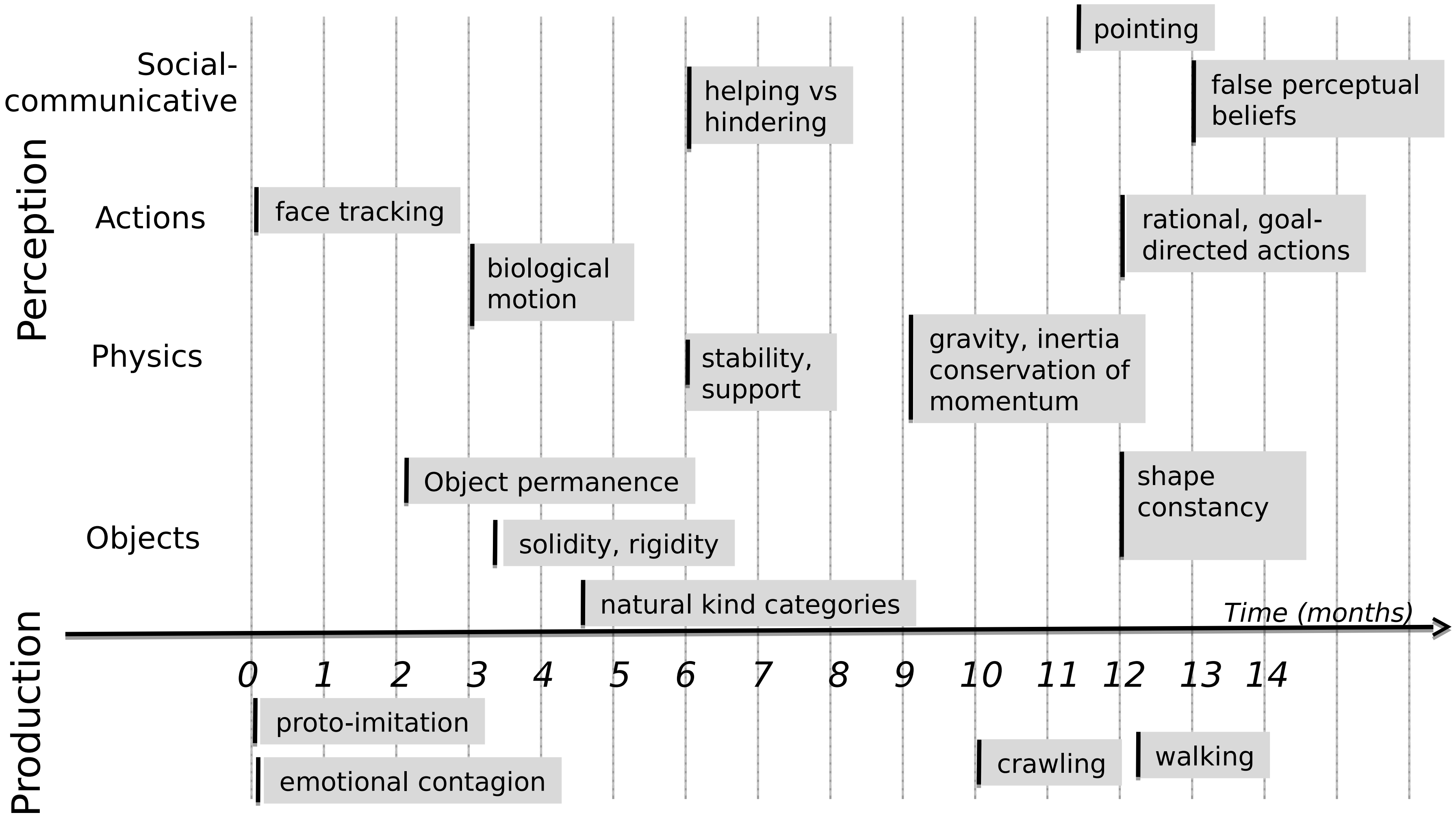
Babies learn how the world works by observation

- **Largely by observation, with remarkably little interaction.**

Photos courtesy of Emmanuel Dupoux

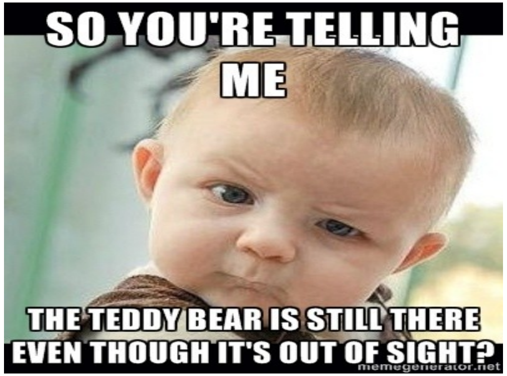
Y. LeCun

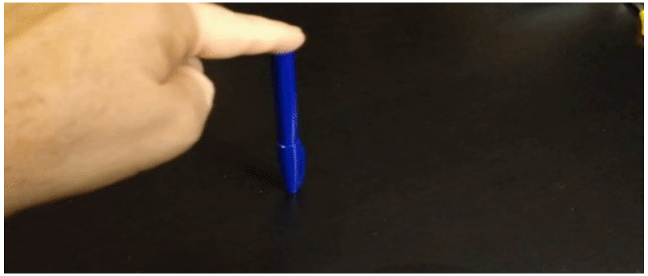
Early Conceptual Acquisition in Infants [from Emmanuel Dupoux]

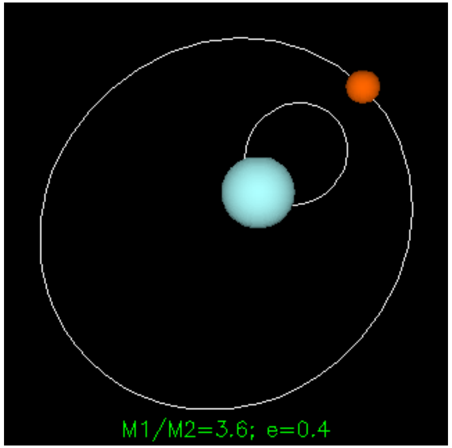


Prediction is the essence of Intelligence

- **We learn models of the world by predicting**







The Next AI Revolution

With thanks to Alyosha Efros and Gil Scott Heron

Get the T-shirt!

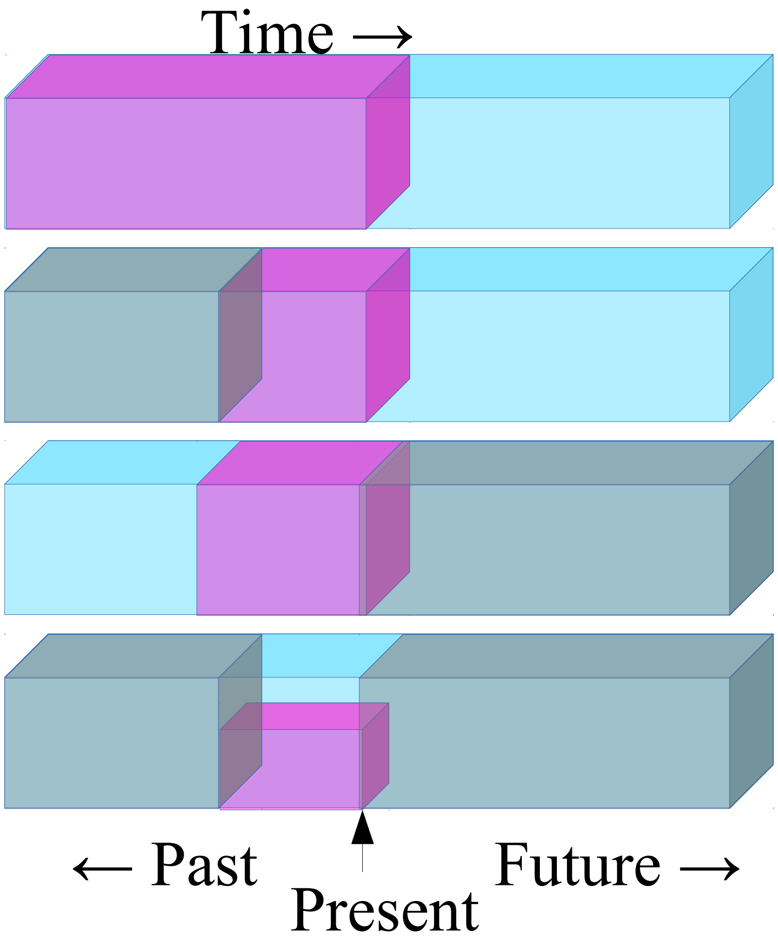
The Salvation?
Self-Supervised Learning

Training very large networks to Understand the world through prediction

facebook
Artificial Intelligence Research

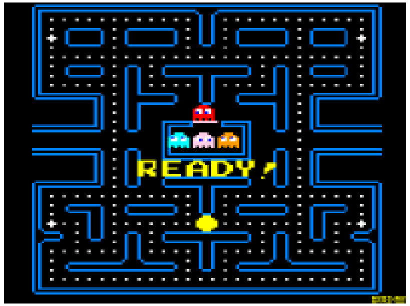
Self-Supervised Learning: Prediction & Reconstruction

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.



Three Types of Learning

- **Reinforcement Learning**
  - The machine predicts a scalar reward given once in a while.
  - **weak feedback**



- **Supervised Learning**
  - The machine predicts a category or a few numbers for each input
  - **medium feedback**

- **Self-supervised Learning**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **A lot of feedback**

How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Self-Supervised Learning: filling in the bl_nks

- **Natural Language Processing: works great!**

OUTPUT: This is a **piece** of text extracted **from** a large set of **news** articles

Decoder

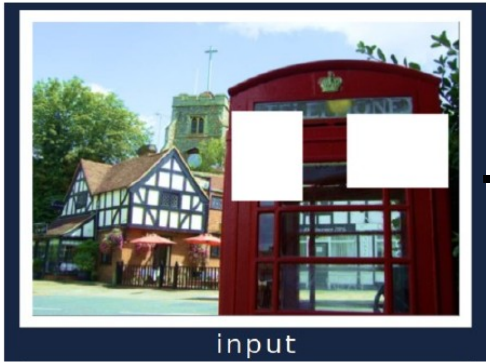
CODE

Encoder

INPUT: This is a [......] of text extracted [.....] a large set of [......] articles

- **Image Recognition / Understanding: works so-so**

[Pathak et al 2014]



Encoder → CODE → Decoder →

Self-Supervised Learning works well for text

- **Word2vec**
  - [Mikolov 2013]
- **FastText**
  - [Joulin 2016] (FAIR)
- **BERT**
  - Bidirectional Encoder Representations from Transformers
  - [Devlin 2018]
- **Cloze-Driven Auto-Encoder**
  - [Baevski 2019] (FAIR)
- **RoBERTa** [Ott 2019] (FAIR)

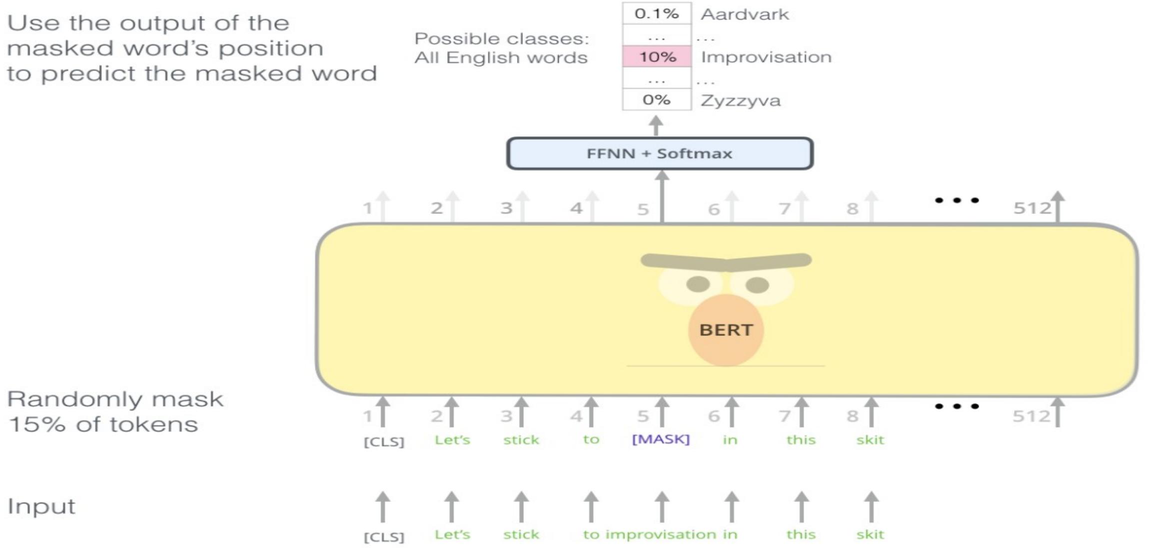
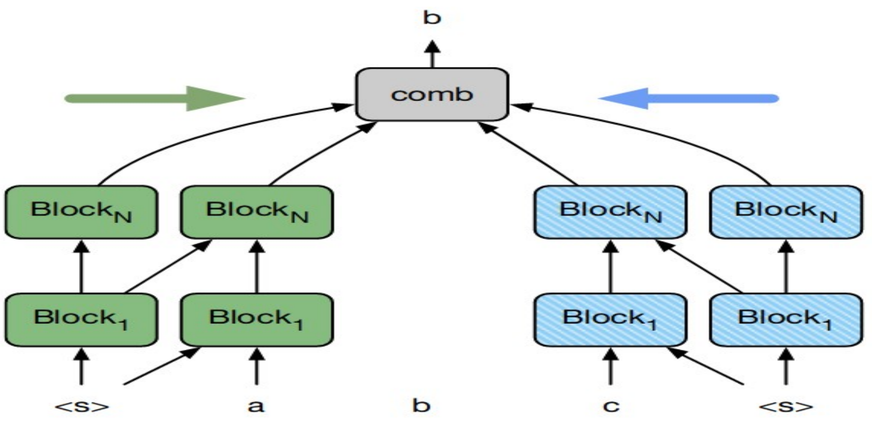


Figure credit: Jay Alammar http://jalammar.github.io/illustrated-bert/



Y. LeCun

Self-Supervised Learning in vision: Filling in the Blanks

input

Barnes et al. | 2009

Darabi et al. | 2012

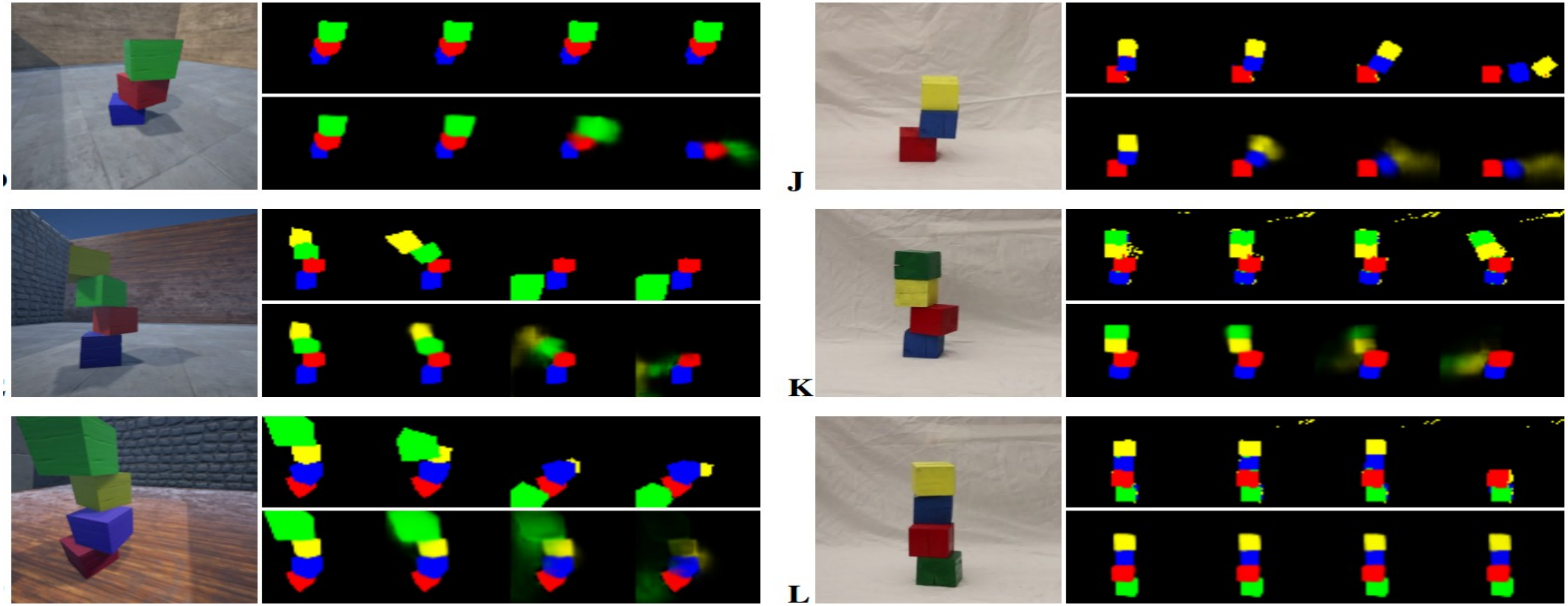
Huang et al. | 2014

Pathak et al. | 2016

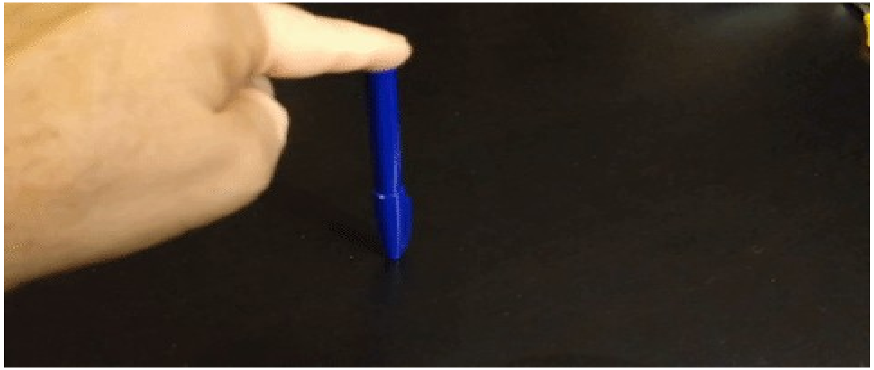
Iizuka et al. | 2017

PhysNet: tell me what happens next

- **[Lerer, Gross, Fergus ICML 2016, arxiv:1603.01312]**
  - ConvNet produces object masks that predict the trajectories of falling blocks. Blurry predictions when uncertain



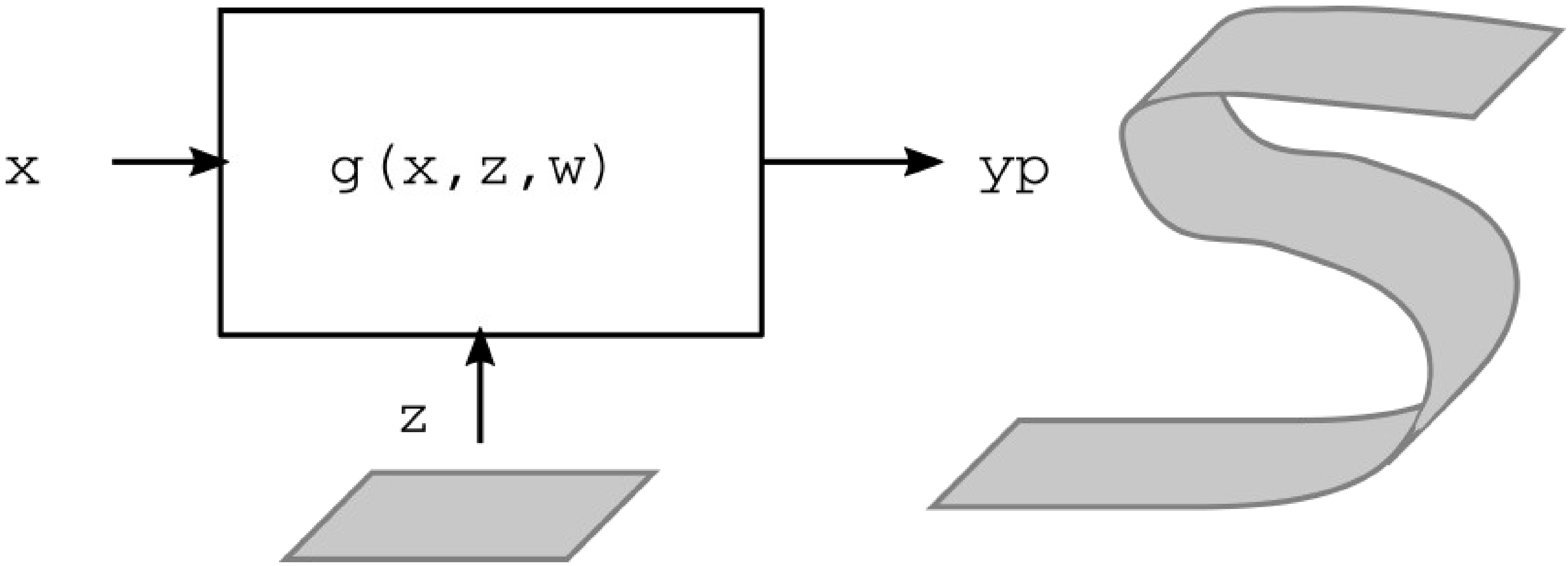
The world is not entirely predictable / stochastic



- **Video prediction:**
  - Multiple futures are possible.
  - Training a system to make a single prediction results in “blurry” results
  - the average of all the possible futures

For multiple predictions: latent variable model

- **As the latent variable z varies, the prediction yp varies over the region of high data density**



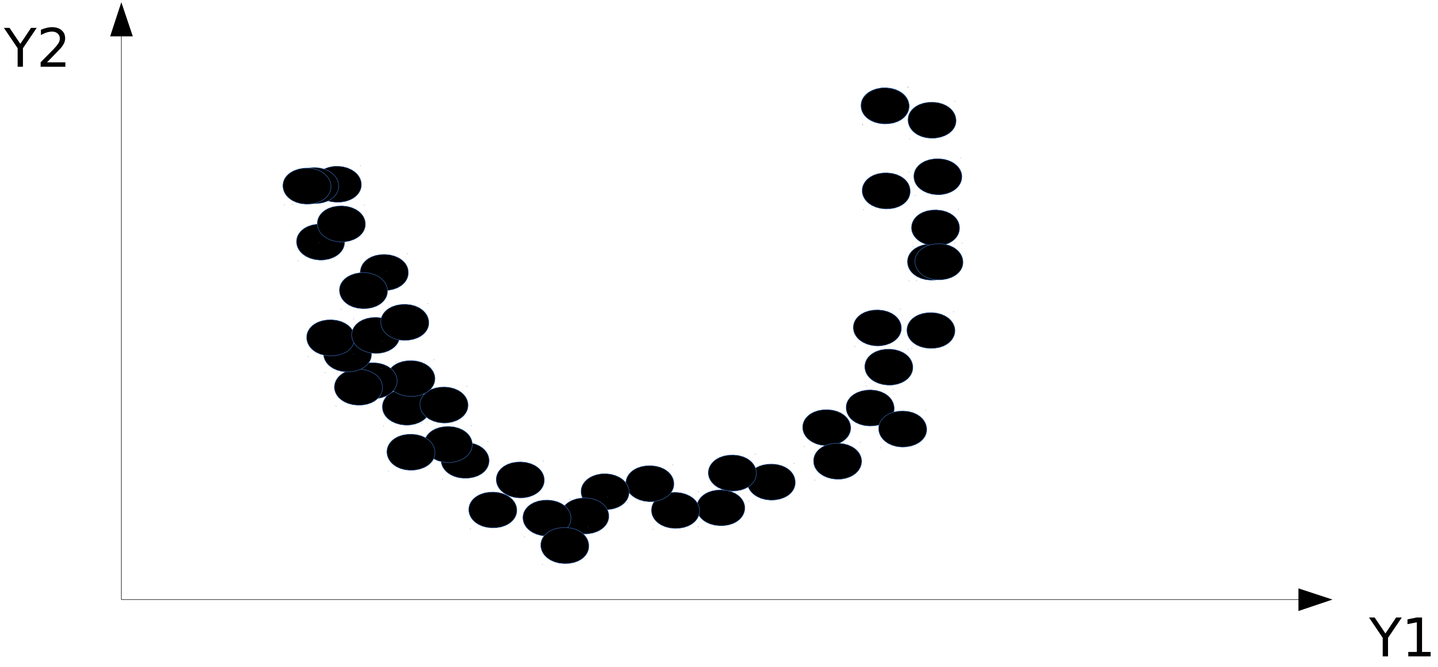
Energy-Based Learning

Energy Shaping
Regularized Auto-Encoders

facebook
Artificial Intelligence Research

Energy-Based Unsupervised Learning

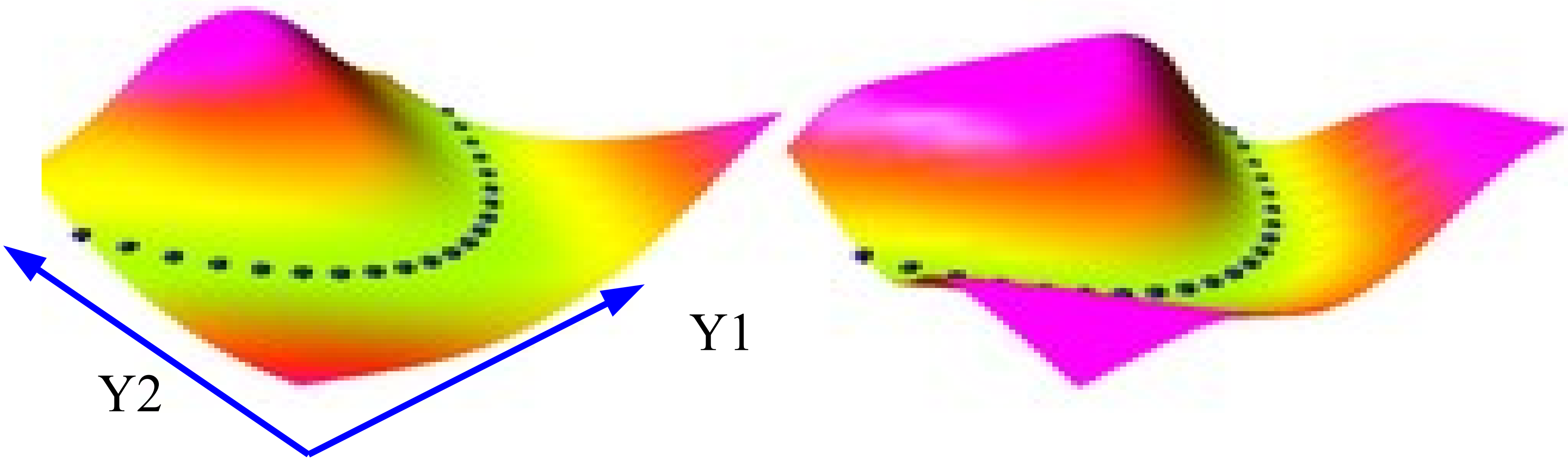
- **Learning an energy function (or contrast function) F(y), that takes**
  - Low values on the data manifold
  - Higher values everywhere else



Capturing Dependencies Between Variables with an Energy Function

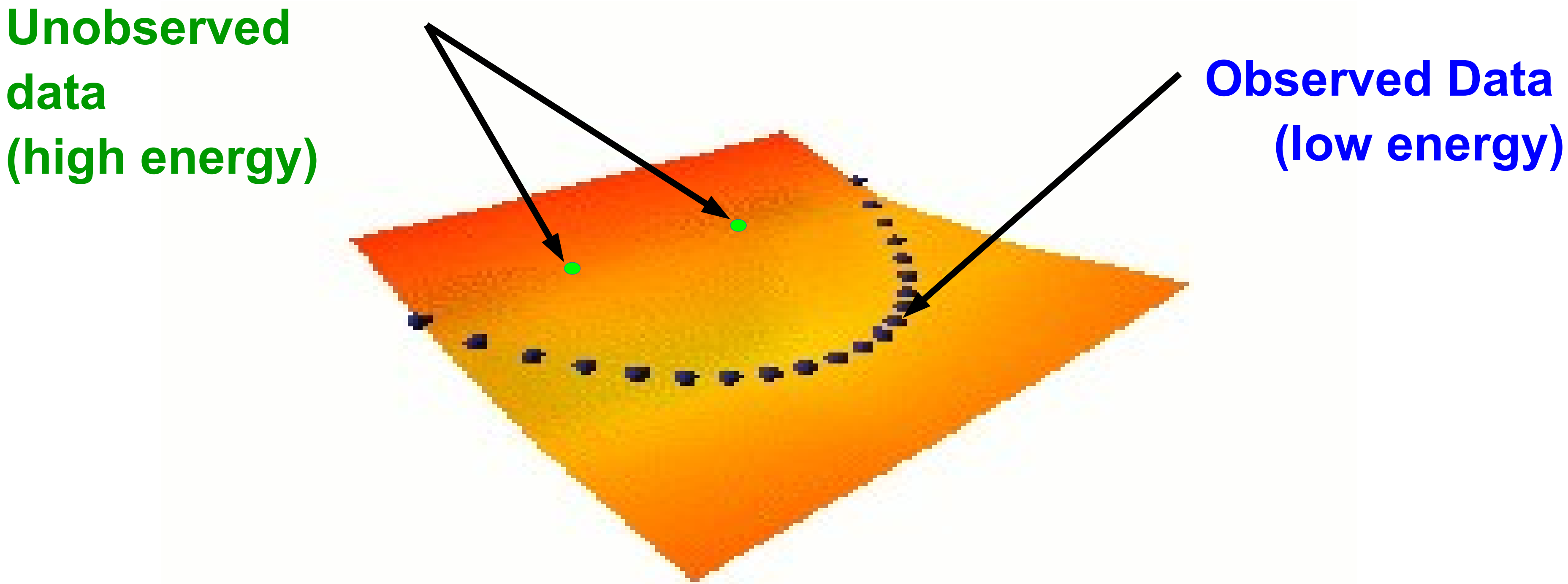
- **The energy surface is a "contrast function" that takes low values on the data manifold, and higher values everywhere else**
  - Special case: energy = negative log density
  - Example: the samples live in the manifold

$$Y_2 = (Y_1)^2$$



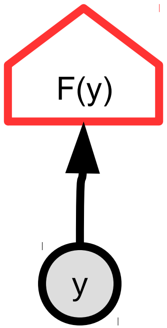
Energy-Based Self-Supervised Learning

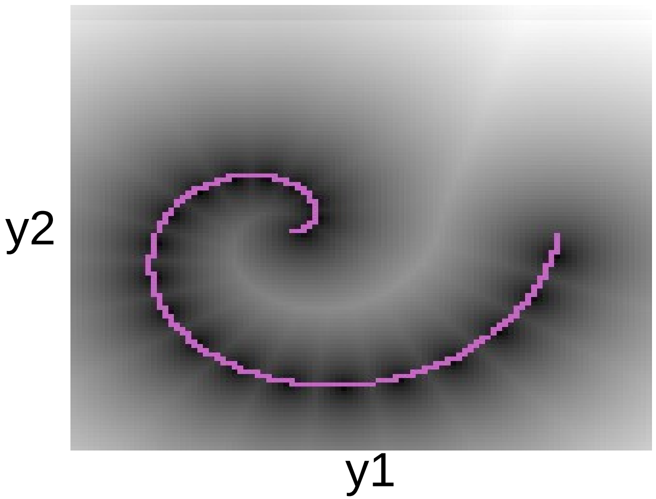
- **Energy Function:** **Takes low value on data manifold, higher values everywhere else**
- **Push down on the energy of desired outputs. Push up on everything else.**
- **But how do we choose where to push up?**



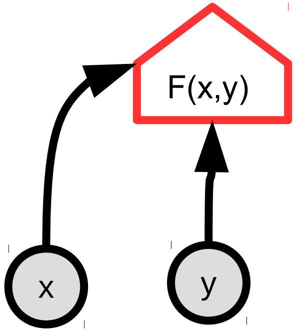
Conditional and Unconditional Energy-Based Model

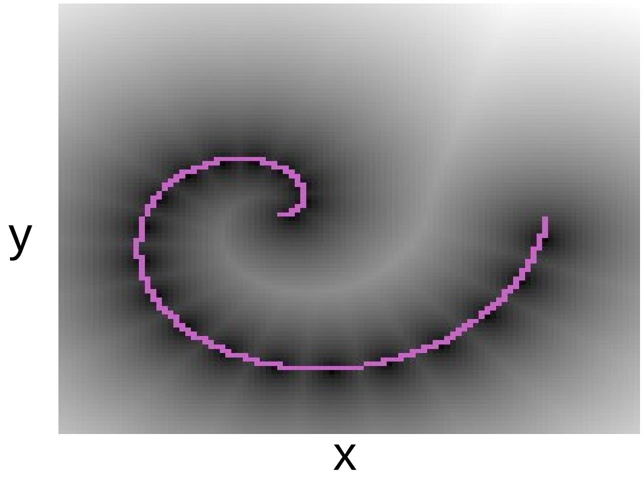
- **Unconditional**
  F(y)





- **Conditional**
  F(x,y)
  - Inference:
    $\bar{y} = \text{argmin}_y F(x,y)$





Learning the Energy Function

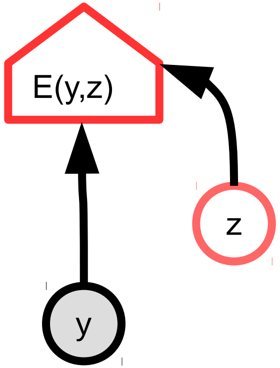
- **parameterized energy function $F_w(x,y)$**
  - Make the energy low on the samples
  - Make the energy higher everywhere else
  - Making the energy low on the samples is easy
  - But how do we make it higher everywhere else?

Conditional and Unconditional Latent Variable EBM

- **Unconditional**

$$F(y)=min_z E(y,z)$$

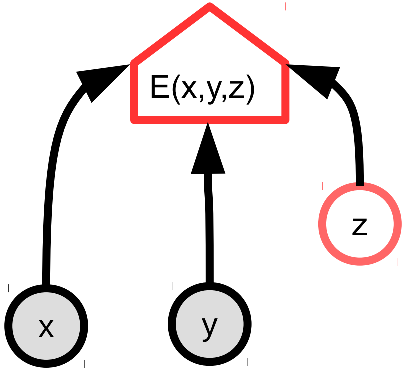
$$F(y)=-\frac{1}{\beta}\log[\int_z \exp(-\beta E(y,z))]$$



- **Conditional**

$$F(x,y)=min_z E(x,y,z)$$

$$F(x,y)=-\frac{1}{\beta}\log[\int_z \exp(-\beta E(x,y,z))]$$



Energy-Based Latent Variable Model for Prediction

- **As the latent variable z varies, the prediction varies over the region of high data density**

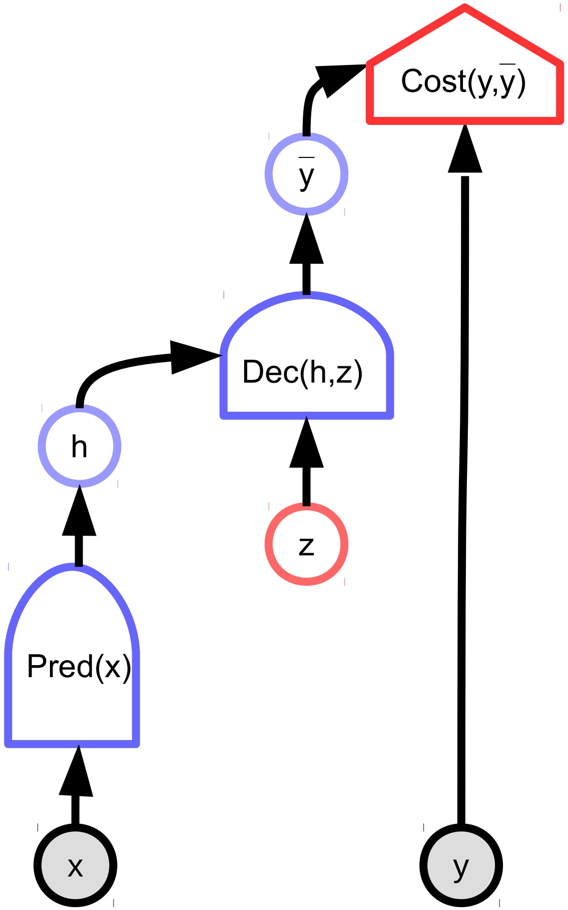
$$E(x,y,z)=Cost(y,Dec(Pred(x),z))$$

- **Inference, free energy:**

$$F(x,y)=min_z E(x,y,z)$$

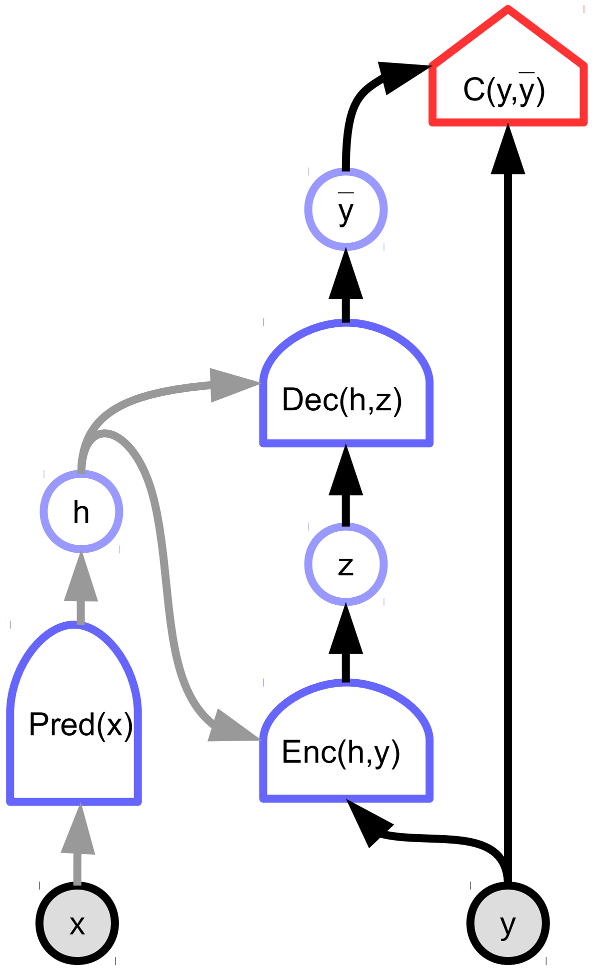
$$F(x,y)=-\frac{1}{\beta}\log\left[\int_z \exp(-\beta E(x,y,z))\right]$$

- **How to make sure that there is no z for y outside the region of high data density?**



(Conditional) Auto-Encoder

- **Auto-encoders learns to reconstruct.**
- **Reconstruction error = Energy function**
- **If it reconstructs everything it's useless**
  - Identity function == flat energy surface
- **How to make sure that the reconstruction error is high outside the region of high data density?**

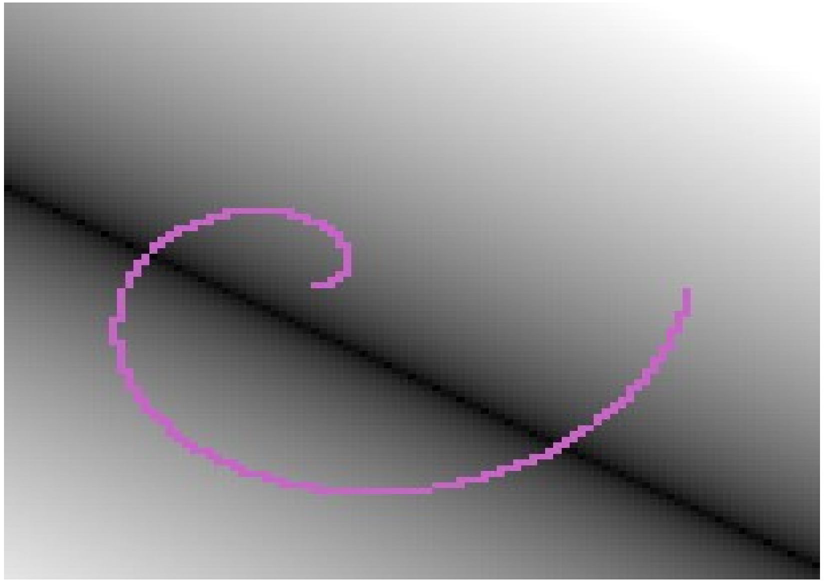


Simple examples: PCA and K-means

- **Limit the capacity of z so that the volume of low energy stuff is bounded**
  - PCA, K-means, GMM, square ICA...

PCA: z is low dimensional

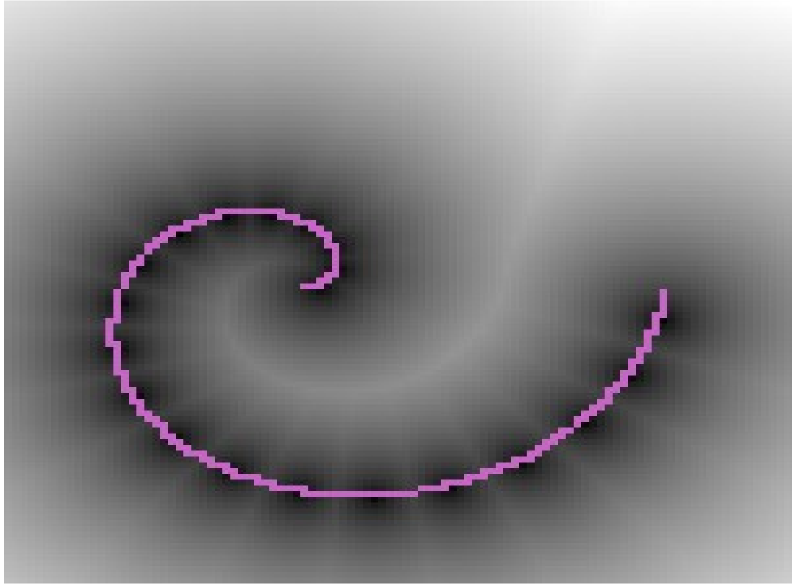
$$F(Y) = ||W^T WY - Y||^2$$



K-Means,

Z constrained to 1-of-K code

$$F(Y) = min_z \sum_i ||Y - W_i Z_i||^2$$

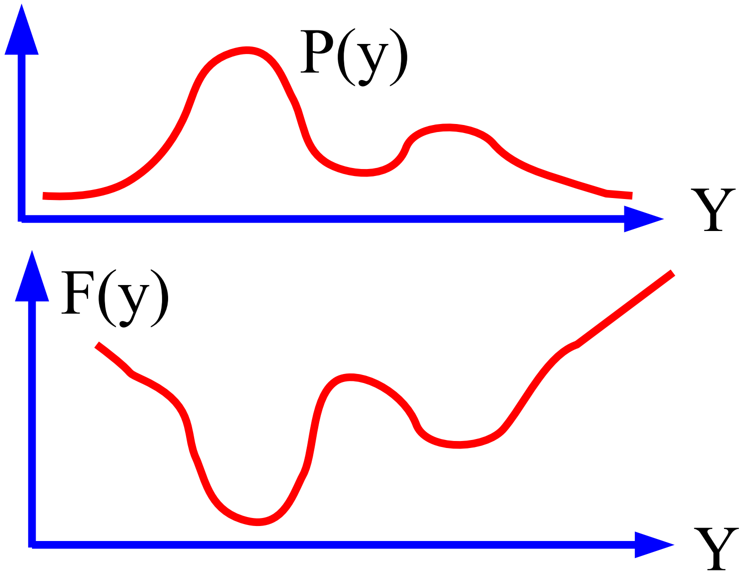


Familiar Example: Maximum Likelihood Learning

- **The energy can be interpreted as an unnormalized negative log density**
- **Gibbs distribution: Probability proportional to exp(-energy)**
  - Beta parameter is akin to an inverse temperature
- **Don't compute probabilities unless you absolutely have to**
  - Because the denominator is often intractable

$$P(y) = -\frac{\exp[-\beta F(y)]}{\int_{y'} \exp[-\beta F(y')]}$$

$$P(y|x) = -\frac{\exp[-\beta F(x,y)]}{\int_{y'} \exp[-\beta F(x,y')]}$$



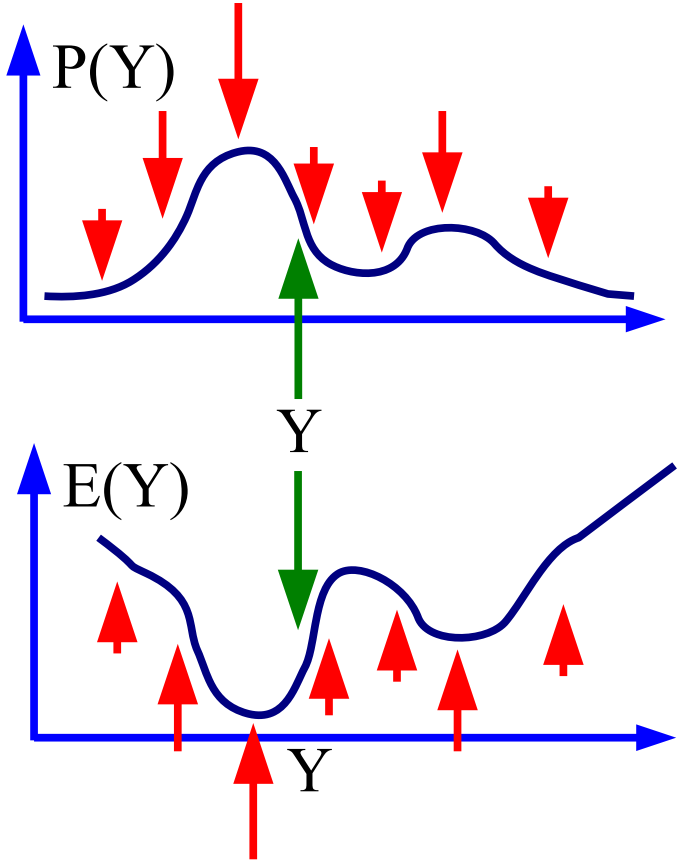
Max likelihood (requires a tractable partition function)

Maximizing P(Y|W) on training samples

make this big

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_y e^{-\beta E(y,W)}}$$

make this small



Minimizing -log P(Y,W) on training samples

$$L(Y,W) = E(Y,W) + \frac{1}{\beta} \log \int_y e^{-\beta E(y,W)}$$

make this small

make this big

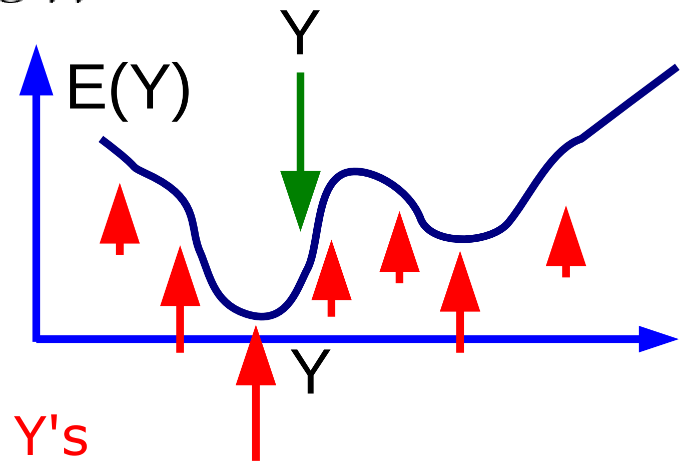
push down of the energy of data points, push up everywhere else

Gradient of the negative log-likelihood loss for one sample Y:

$$\frac{\partial L(Y,W)}{\partial W} = \frac{\partial E(Y,W)}{\partial W} - \int_y P(y|W)\frac{\partial E(y,W)}{\partial W}$$

Gradient descent:

$$W \leftarrow W - \eta \frac{\partial L(Y,W)}{\partial W}$$



Pushes down on the energy of the samples

Pulls up on the energy of low-energy Y's

$$W \leftarrow W - \eta \frac{\partial E(Y,W)}{\partial W} + \eta \int_y P(y|W)\frac{\partial E(y,W)}{\partial W}$$

Seven Strategies to Shape the Energy Function

MOST IMPORTANT SLIDE

- **1. build the machine so that the volume of low energy stuff is constant**
  - PCA, K-means, GMM, square ICA
- **2. push down of the energy of data points, push up everywhere else**
  - Max likelihood (needs tractable partition function or variational approximation)
- **3. push down of the energy of data points, push up on chosen locations**
  - Contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Min Probability Flow, adversarial generator/GANs
- **4. minimize the gradient and maximize the curvature around data points** score matching
- **5. if F(Y) = ||Y - G(Y)||^2, make G(Y) as "constant" as possible.**
  - Contracting auto-encoder, saturating auto-encoder
- **6. train a dynamical system so that the dynamics goes to the data manifold**
  - denoising auto-encoder, masked auto-encoder (e.g. BERT)
- **7. use a regularizer that limits the volume of space that has low energy**
  - Sparse coding, sparse auto-encoder, LISTA & PSD, Variational auto-encoders

Regularized Latent Variable EBM

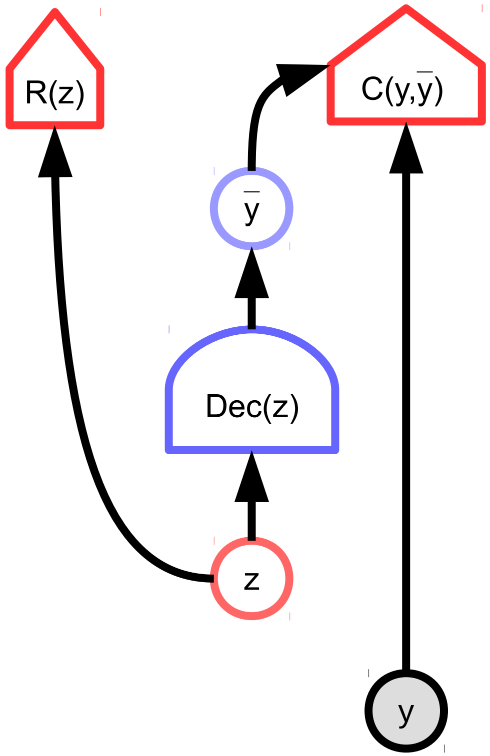
Sparse Modeling, Regularized Auto-Encoders

facebook
Artificial Intelligence Research

Regularized Latent-Variable EBM

- **Restrict the information capacity of z through regularization**

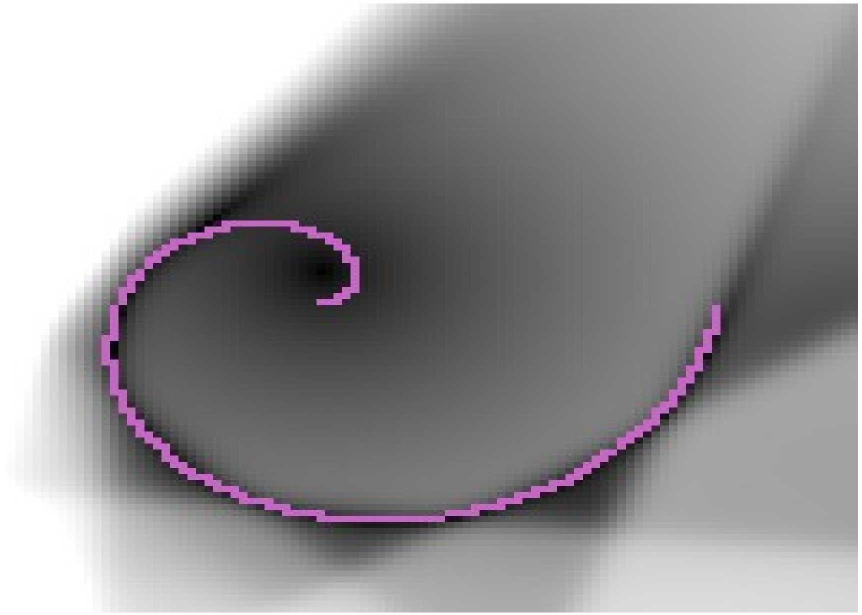
$$E(y,z)=C(y,Dec(z))+\lambda R(z)$$

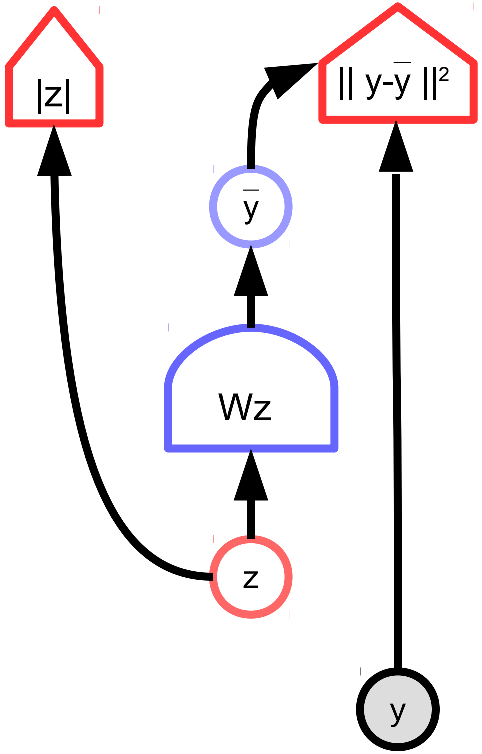


Familiar Example: Sparse Coding / Sparse Modeling

- **Regularized latent variable through L1 norm**
  - Induces sparsity

$$E(y,z) = ||y - Wz||^2 + \lambda |z|$$





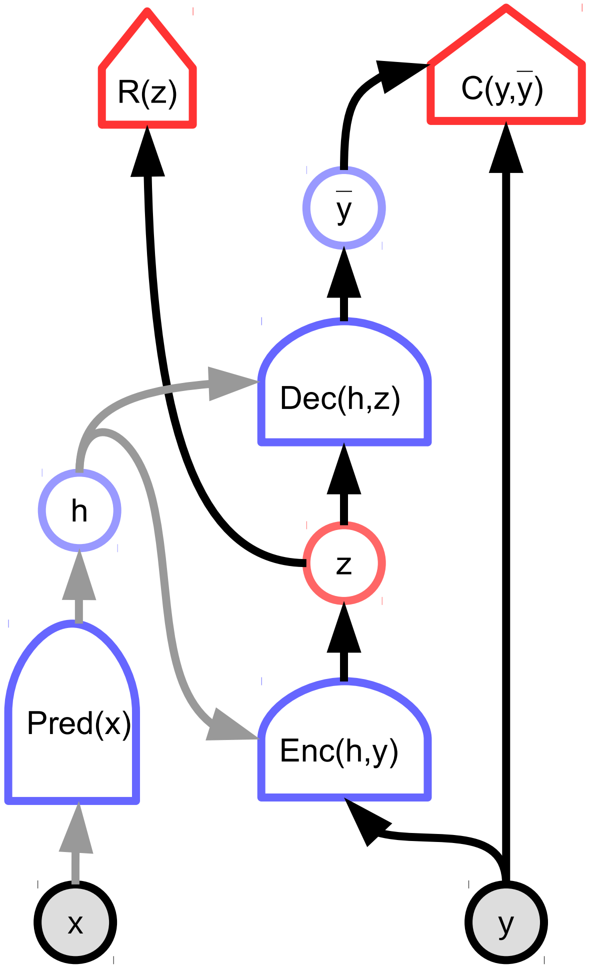
Sparse Auto-Encoders

Computing the sparse code efficiently

facebook
Artificial Intelligence Research

Sparse Auto-Encoder

- **Encoder learns to compute the latent variable efficiently**
- **No explicit inference step**

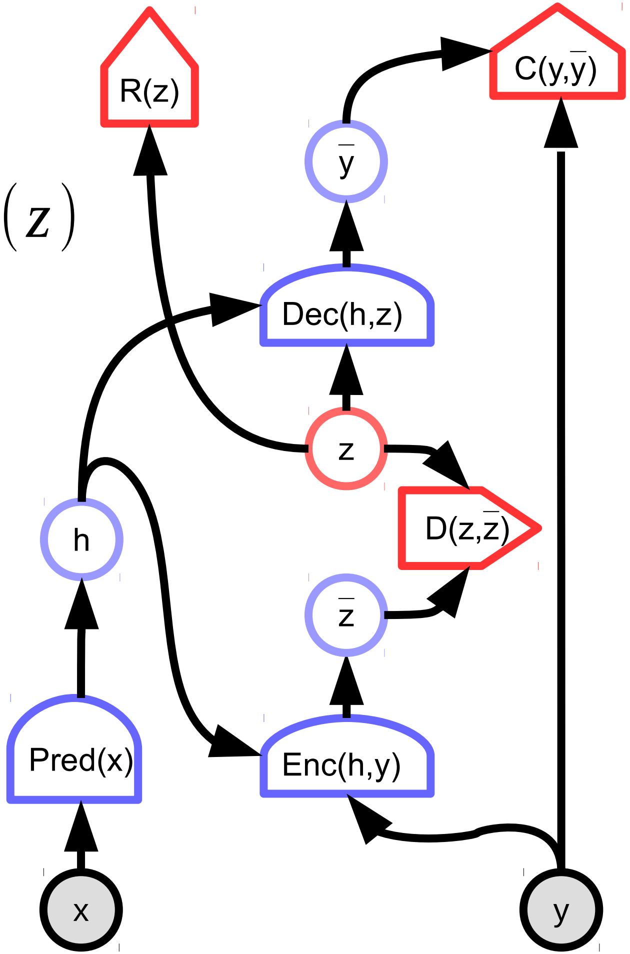


Sparse (conditional) Auto-Encoder

- **Encoder learns to compute the latent variable efficiently**

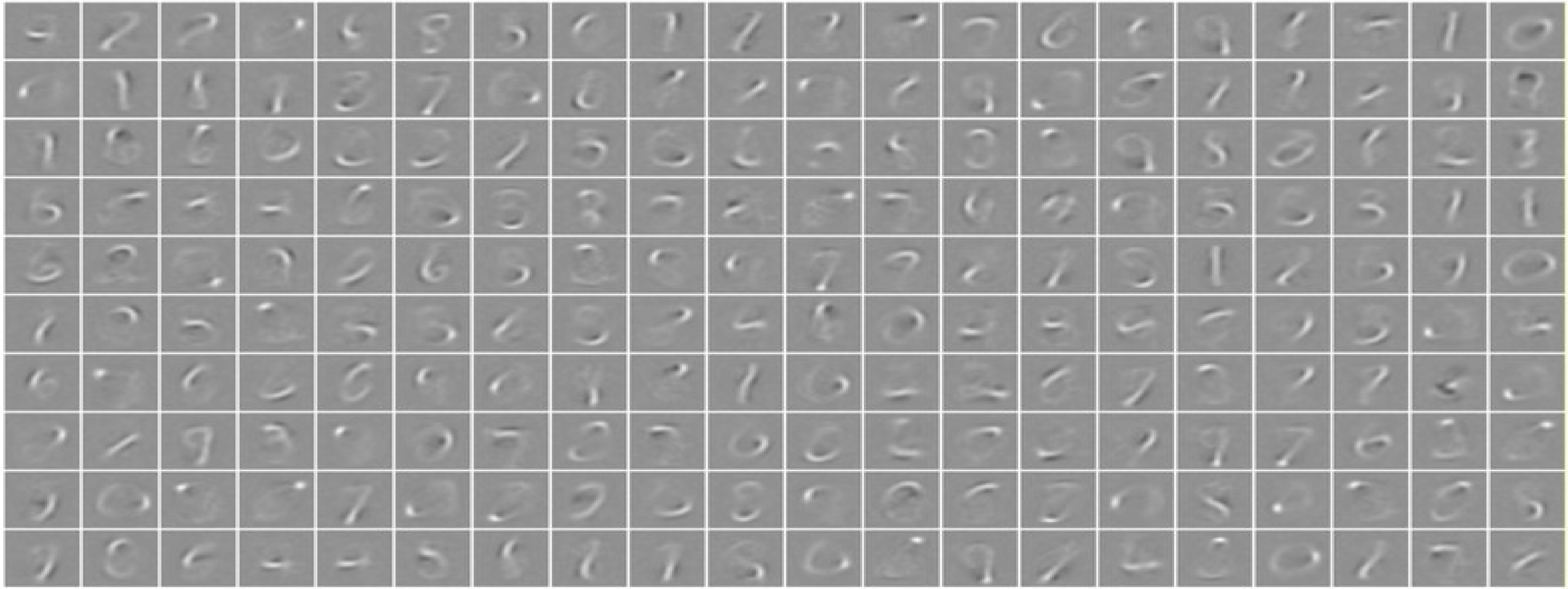
$$E(x,y,z)=C(y,Dec(h,z))+D(z,Enc(h,y))+R(z)$$

- **Simple inference step**



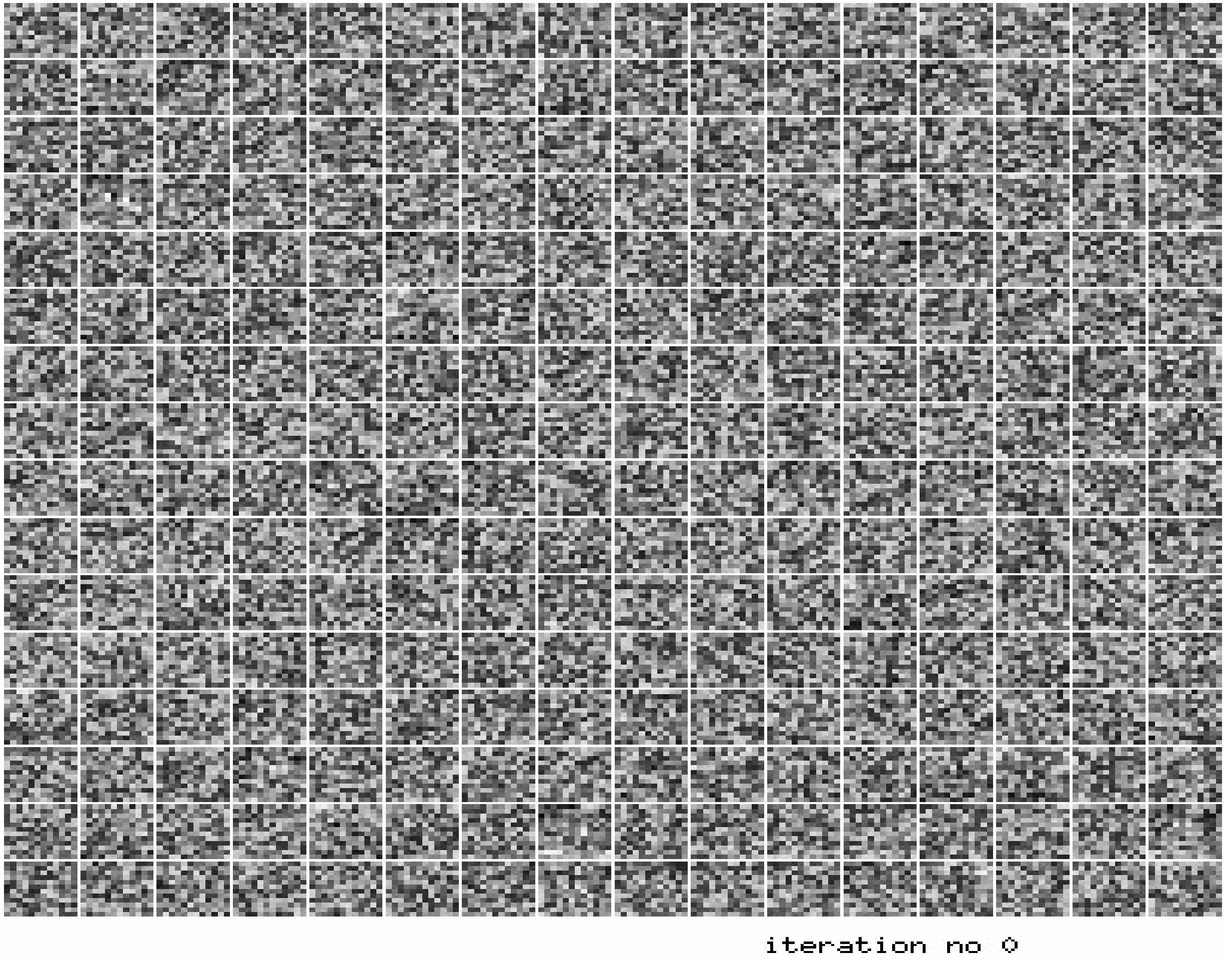
Sparse Modeling on handwritten digits (MNIST)

- **Basis functions (columns of decoder matrix) are digit parts**
- **All digits are a linear combination of a small number of these**

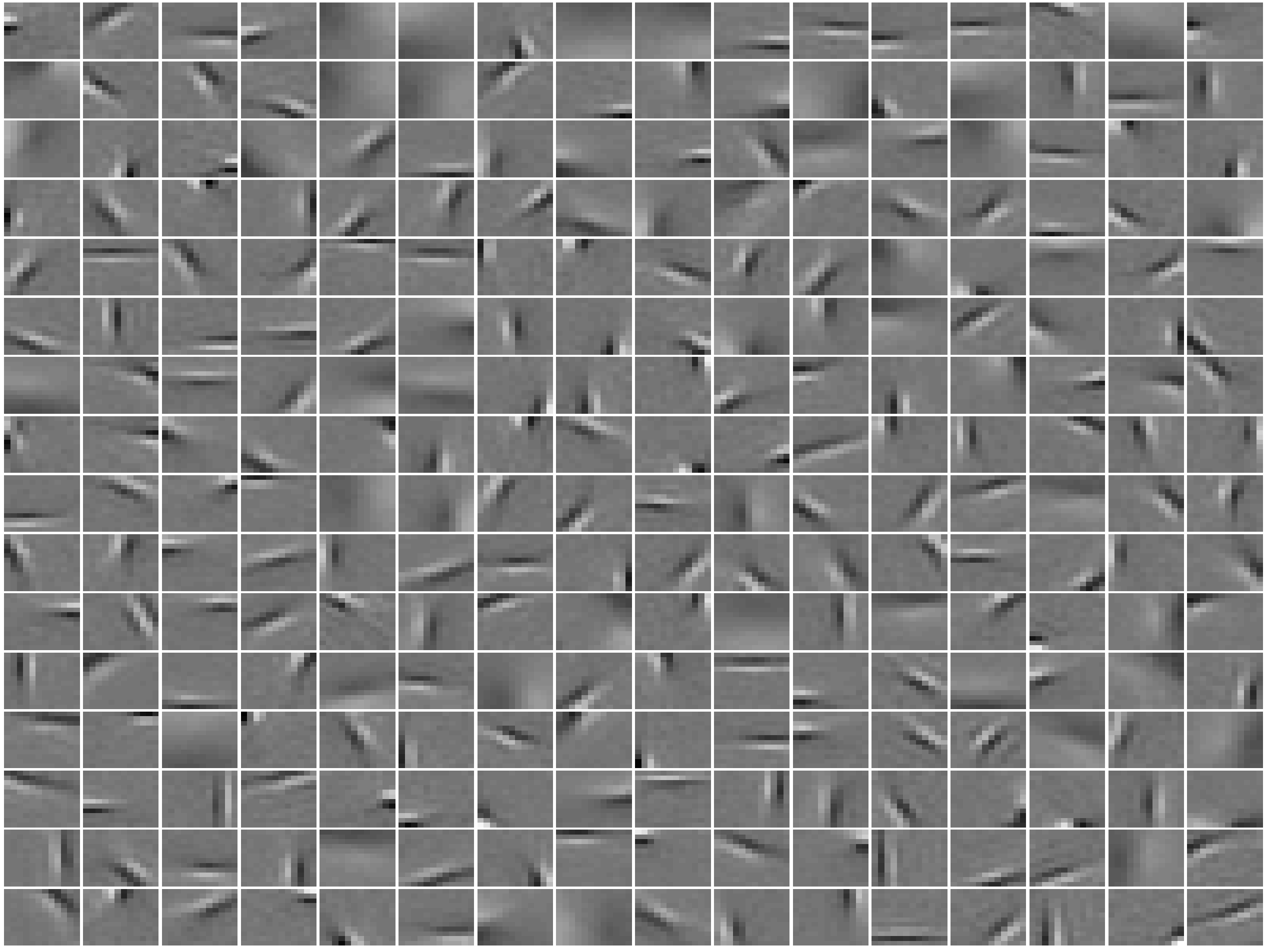


Predictive Sparse Decomposition (PSD): Training

- **Training on natural images patches.**
  - 12X12
  - 256 basis functions

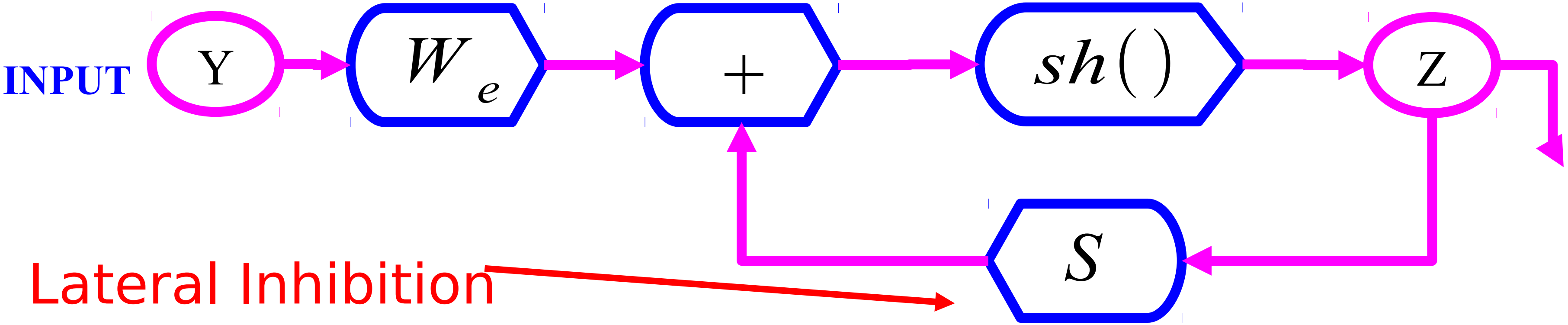


Learned Features on natural patches: V1-like receptive fields



Giving the “right” structure to the encoder

- **ISTA/FISTA: iterative algorithm that converges to optimal sparse code**



$$Z(t+1) = \mathrm{Shrinkage}_{\lambda/L}\left[Z(t) - \frac{1}{L}W_d^T(W_d Z(t) - Y)\right]$$

- **ISTA/FastISTA reparameterized:**

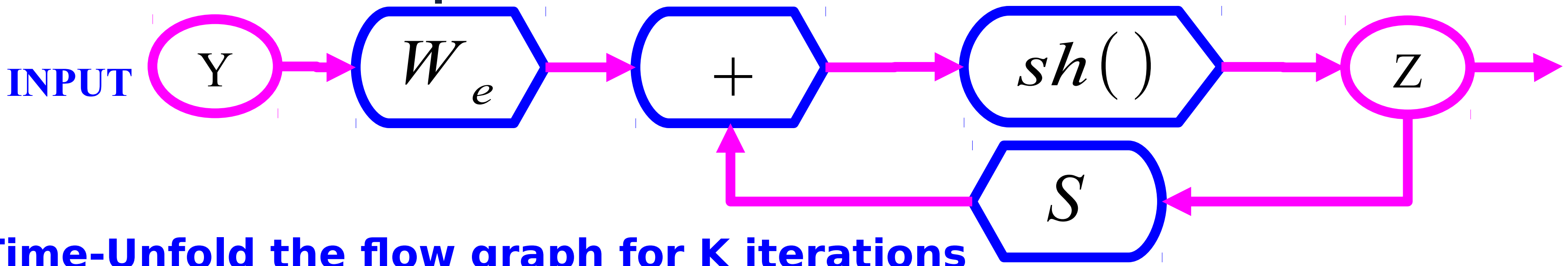
$$Z(t+1) = \mathrm{Shrinkage}_{\lambda/L}\left[W_e^T Y + S Z(t)\right]; \quad W_e = \frac{1}{L}W_d; \quad S = I - \frac{1}{L}W_d^T W_d$$

- **LISTA (Learned ISTA): learn the We and S matrices to get fast solutions**

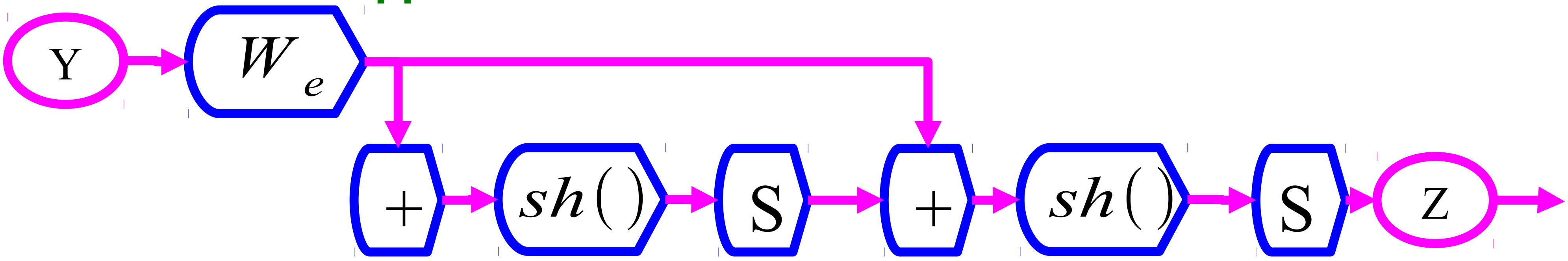
[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

LISTA: Train We and S matrices to give a good approximation quickly

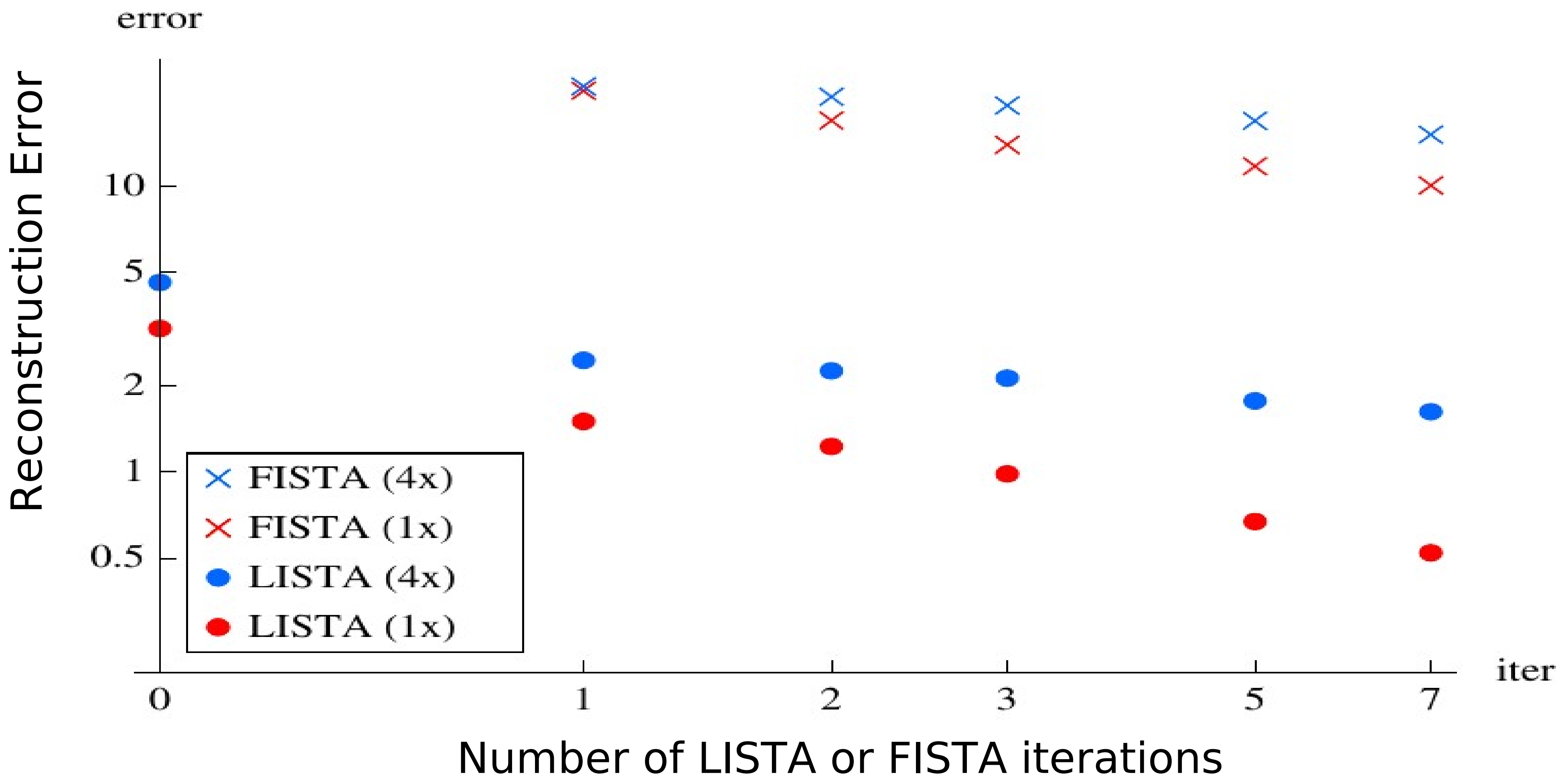
- **Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters**



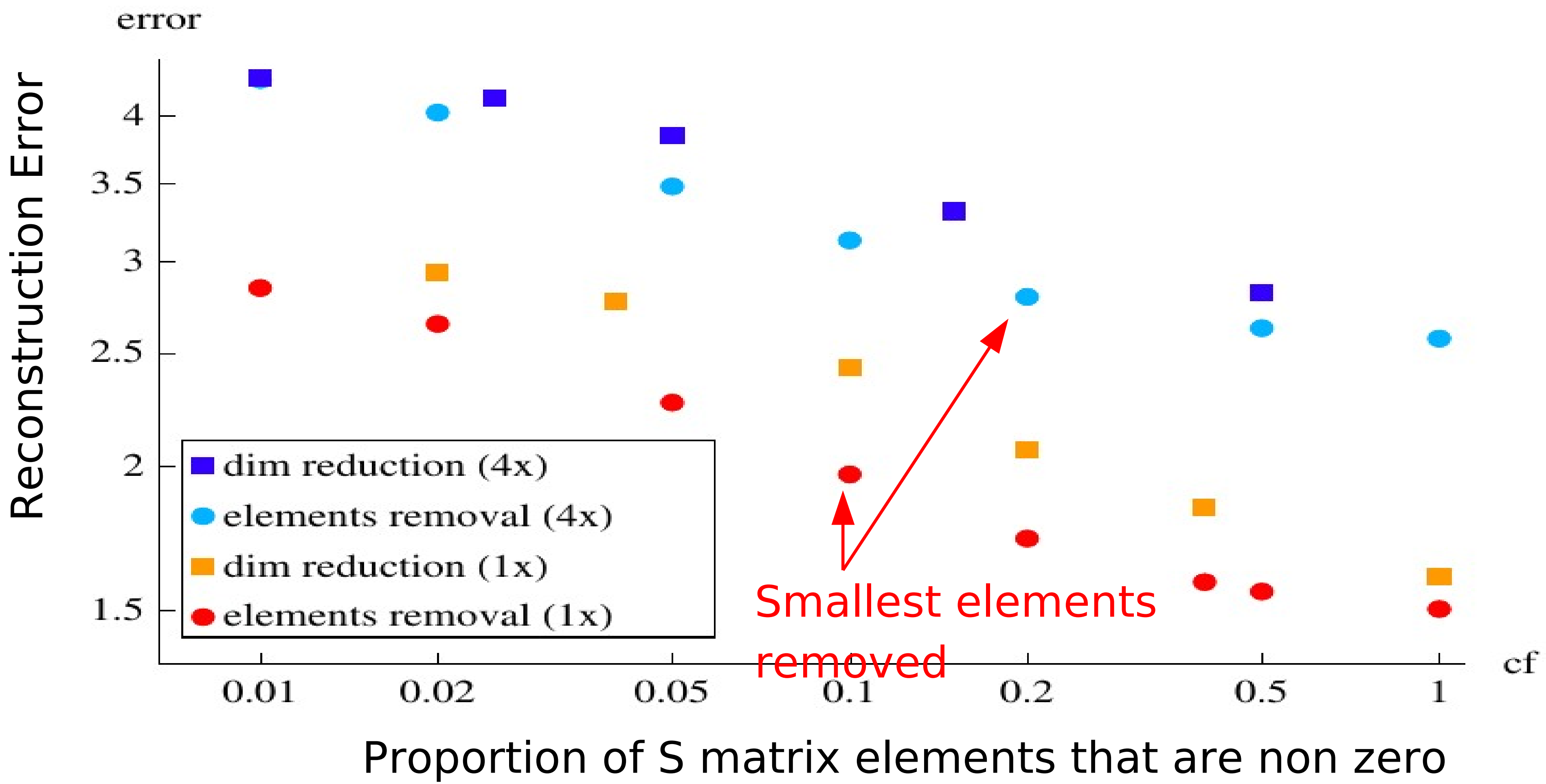
- **Time-Unfold the flow graph for K iterations**
- **Learn the We and S matrices with "backprop-through-time"**
- **Get the best approximate solution within K iterations**



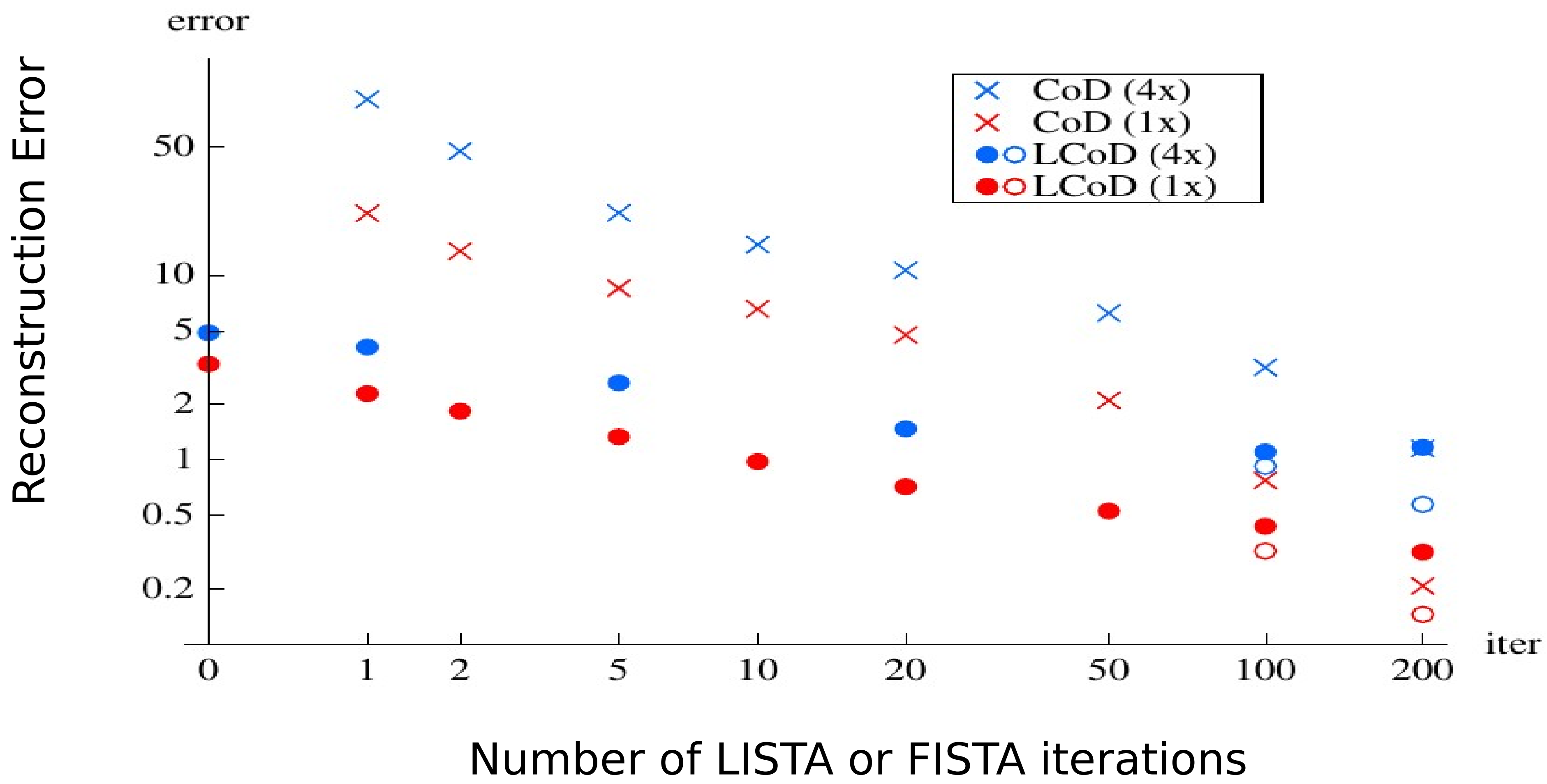
Learning ISTA (LISTA) vs ISTA/FISTA



LISTA with partial mutual inhibition matrix

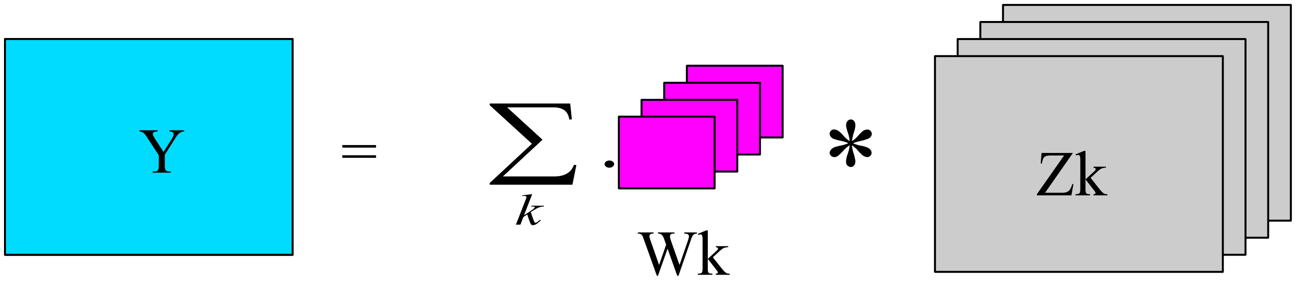


Learning Coordinate Descent (LcoD): faster than LISTA



Convolutional Sparse Coding

- **Replace the dot products with dictionary element by convolutions.**
  - Input Y is a full image
  - Each code component Zk is a feature map (an image)
  - Each dictionary element is a convolution kernel
- **Regular sparse coding** $E(Y,Z) = ||Y - \sum_k W_k Z_k||^2 + \alpha \sum_k |Z_k|$
- **Convolutional S.C.** $E(Y,Z) = ||Y - \sum_k W_k * Z_k||^2 + \alpha \sum_k |Z_k|$

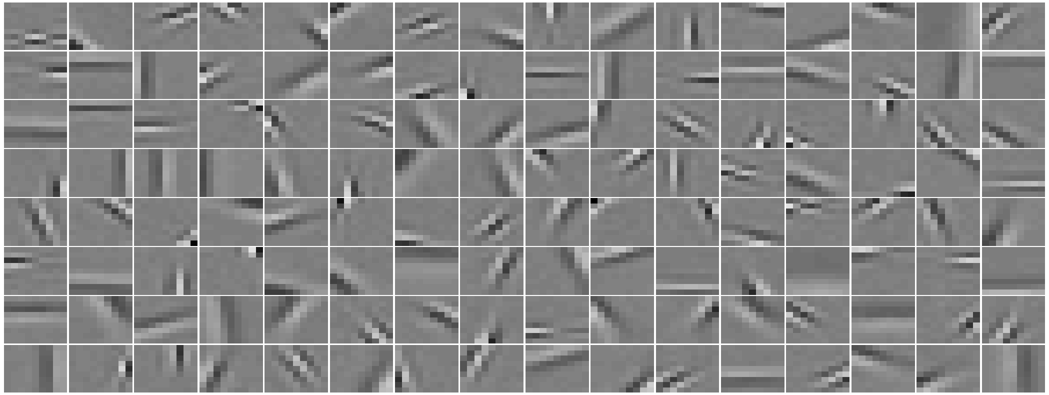


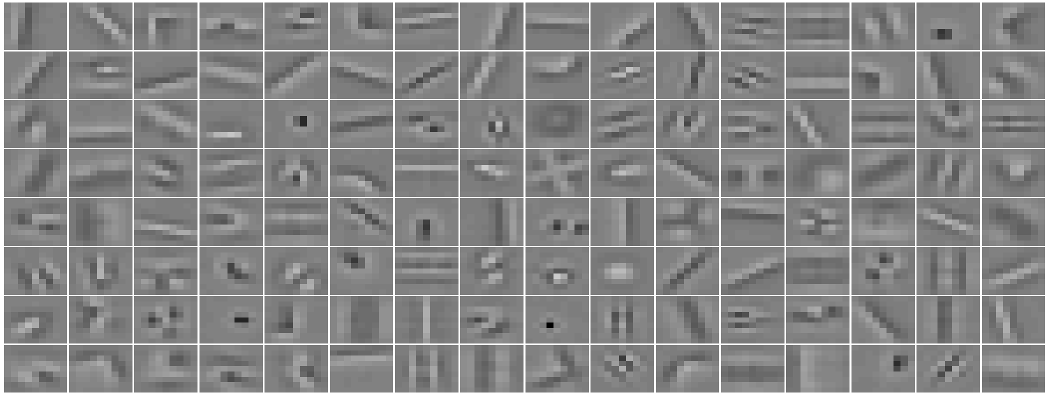
Also used in "deconvolutional networks" [Zeiler, Taylor, Fergus CVPR 2010]

Convolutional PSD: Encoder with a soft sh() Function

- **Convolutional Formulation**
  - Extend sparse coding from ***PATCH*** to ***IMAGE***

$$\mathcal{L}(x, z, \mathcal{D}) = \frac{1}{2}||x - \sum_{k=1}^{K} \mathcal{D}_k * z_k||_2^2 + \sum_{k=1}^{K} ||z_k - f(W^k * x)||_2^2 + |z|_1$$



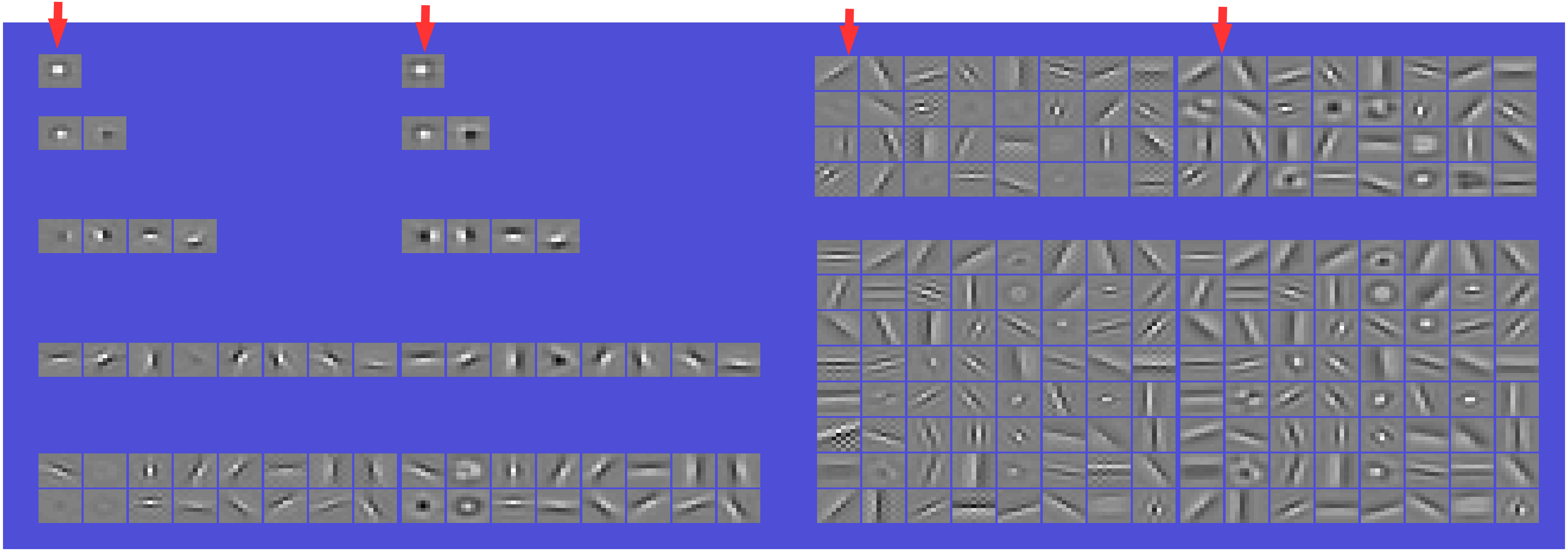


- ***PATCH*** based learning
- ***CONVOLUTIONAL*** learning

Convolutional Sparse Auto-Encoder on Natural Images

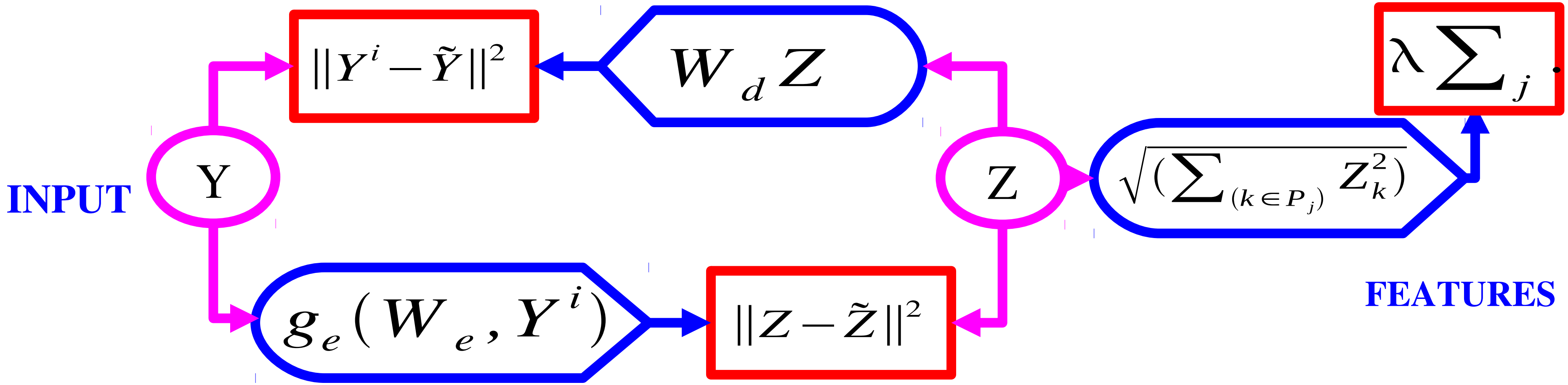
- **Filters and Basis Functions obtained**
  - with 1, 2, 4, 8, 16, 32, and 64 filters.

Encoder Filters   Decoder Filters   Encoder Filters   Decoder Filters



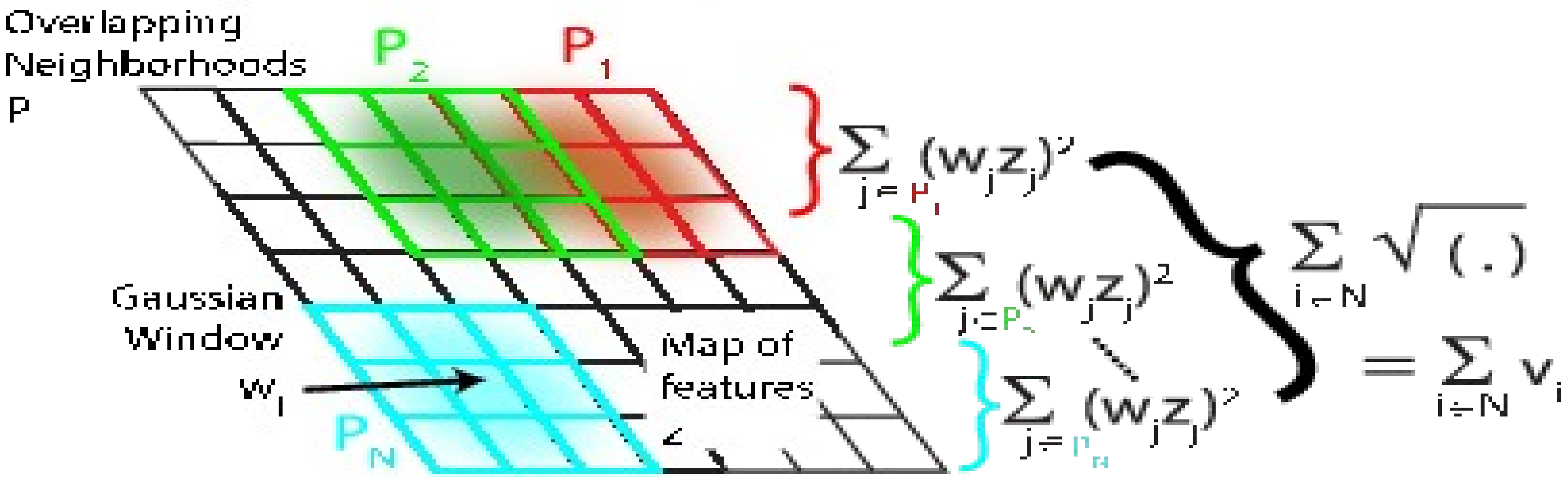
Learning Invariant Features [Kavukcuoglu et al. CVPR 2009]

- **Unsupervised PSD ignores the spatial pooling step.**
- **Could we devise a similar method that learns the pooling layer as well?**
- **Idea [Hyvarinen & Hoyer 2001]: group sparsity on pools of features**
  - Minimum number of pools must be non-zero
  - Number of features that are on within a pool doesn't matter
  - Polls tend to regroup similar features



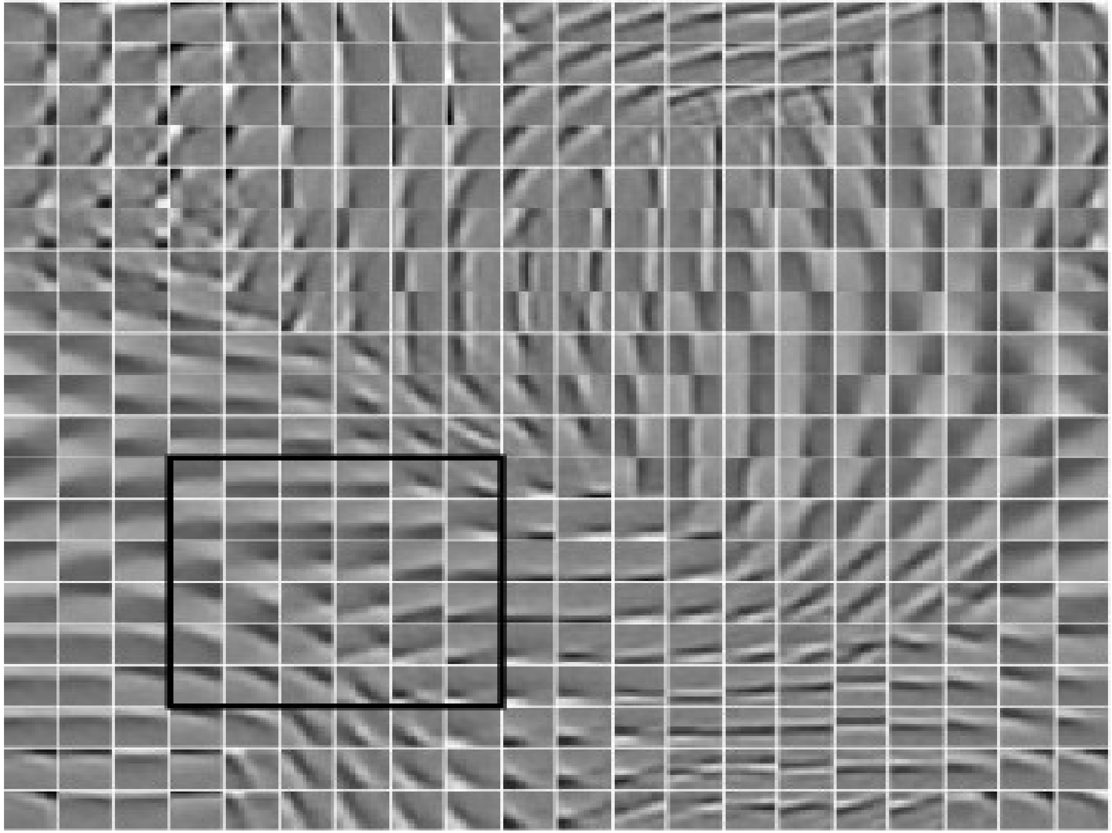
Why just pool over space? Why not over orientation?

- **Using an idea from Hyvarinen: topographic square pooling (subspace ICA)**
  - 1. Apply filters on a patch (with suitable non-linearity)
  - 2. Arrange filter outputs on a 2D plane
  - 3. square filter outputs
  - 4. minimize sqrt of sum of blocks of squared filter outputs



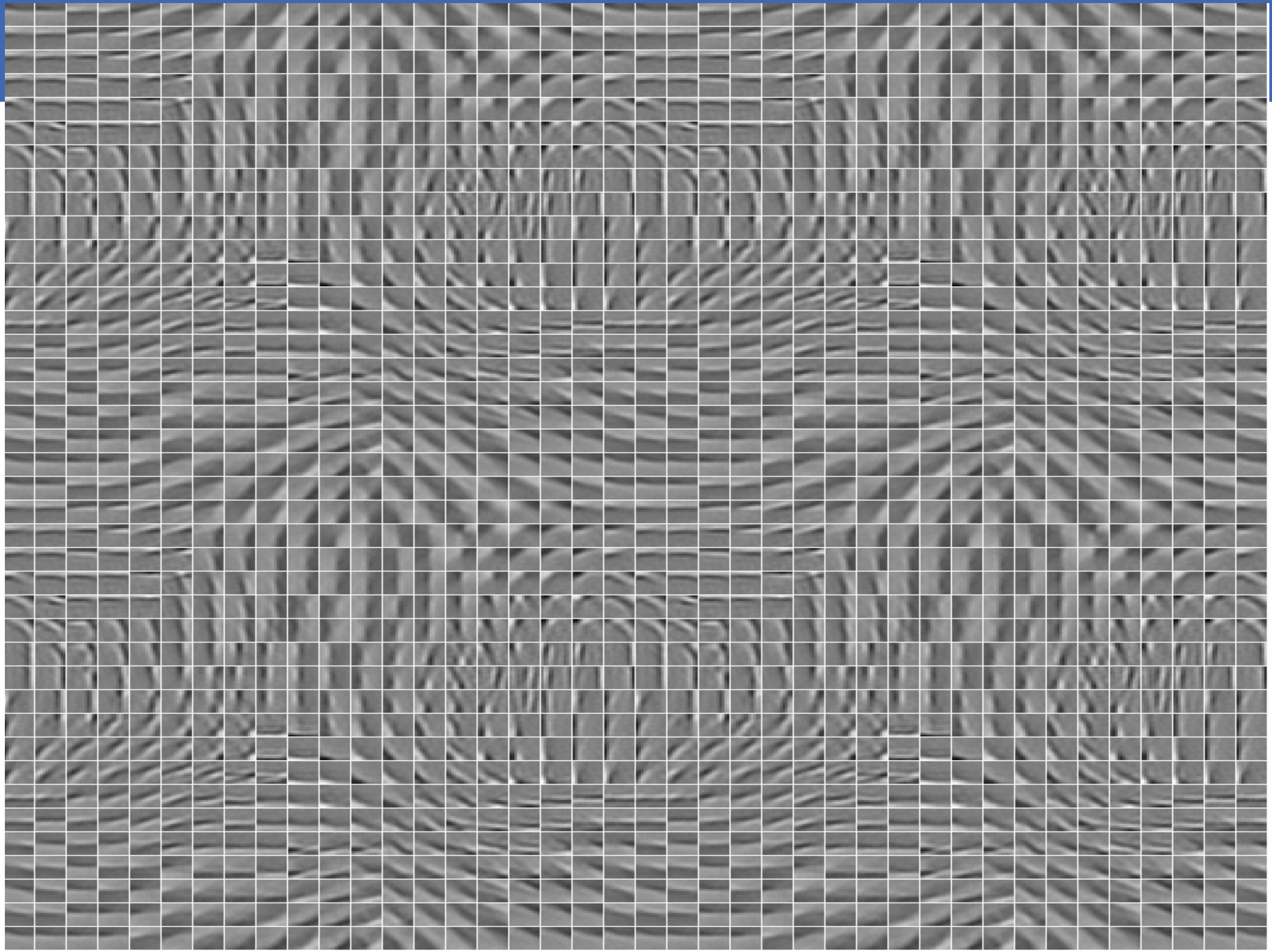
Why just pool over space? Why not over orientation?

- **The filters arrange themselves spontaneously so that similar filters enter the same pool.**
- **The pooling units can be seen as complex cells**
- **They are invariant to local transformations of the input**
  - For some it's translations, for others rotations, or other transformations.



Pinwheels!

- **Does that look pinwheely to you?**

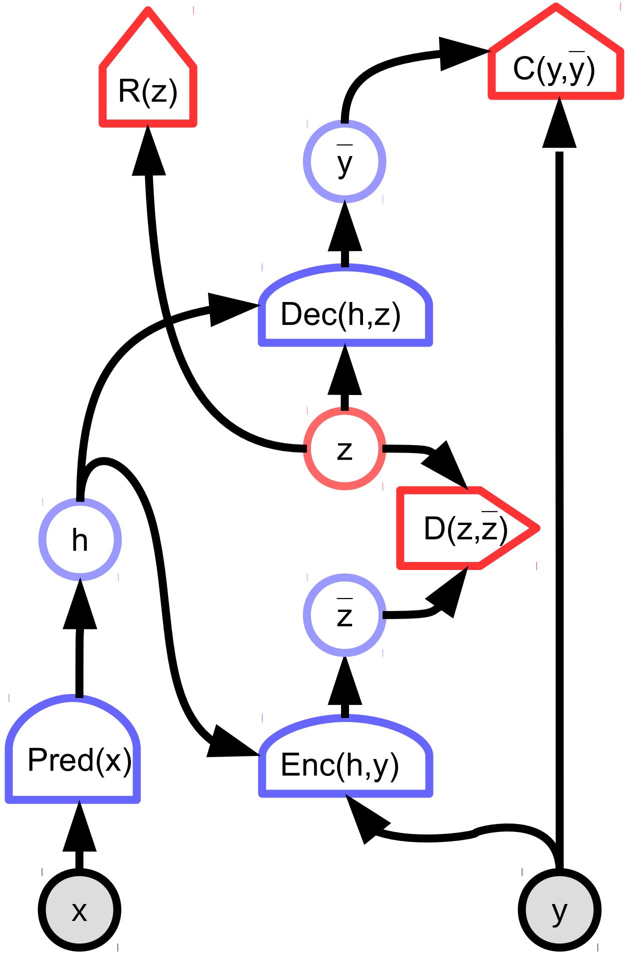


Variational Auto-Encoder

Limiting the information content
of the code by adding noise to it

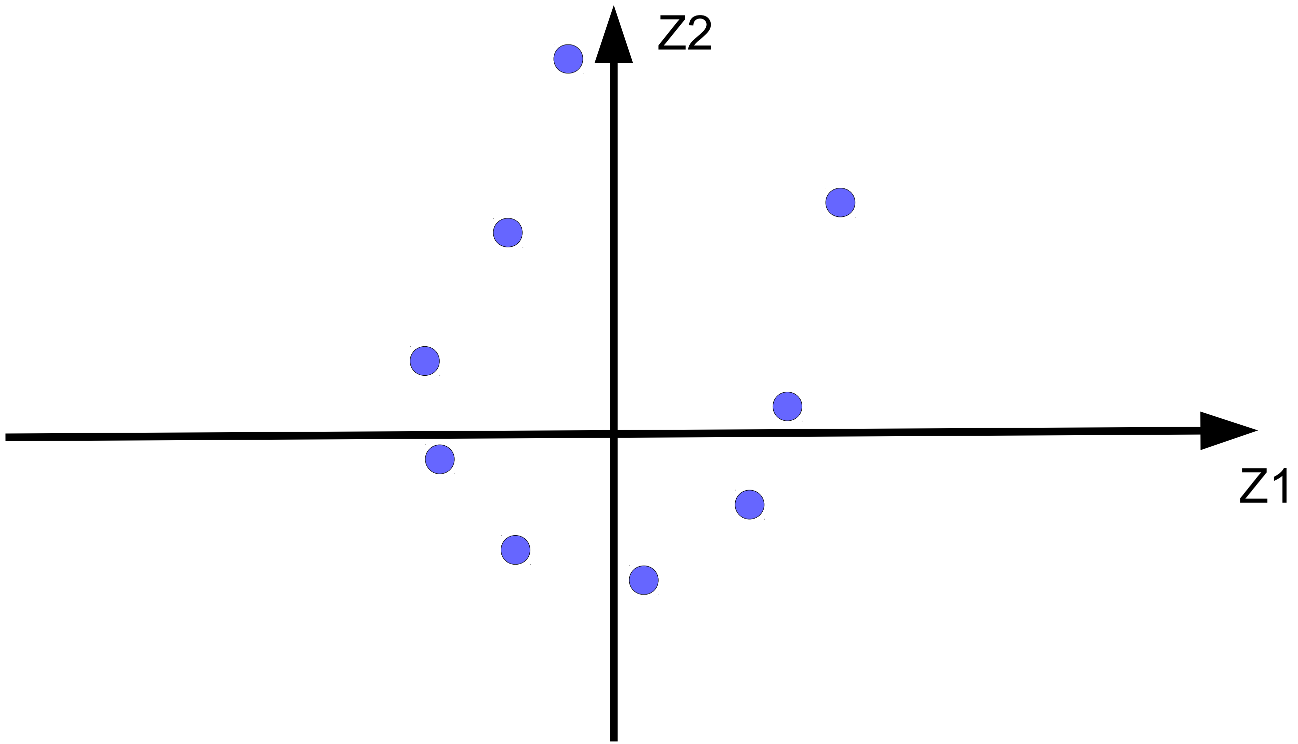
Variational (conditional) AE

- **Use the D() energy term as a prior to sample z from**



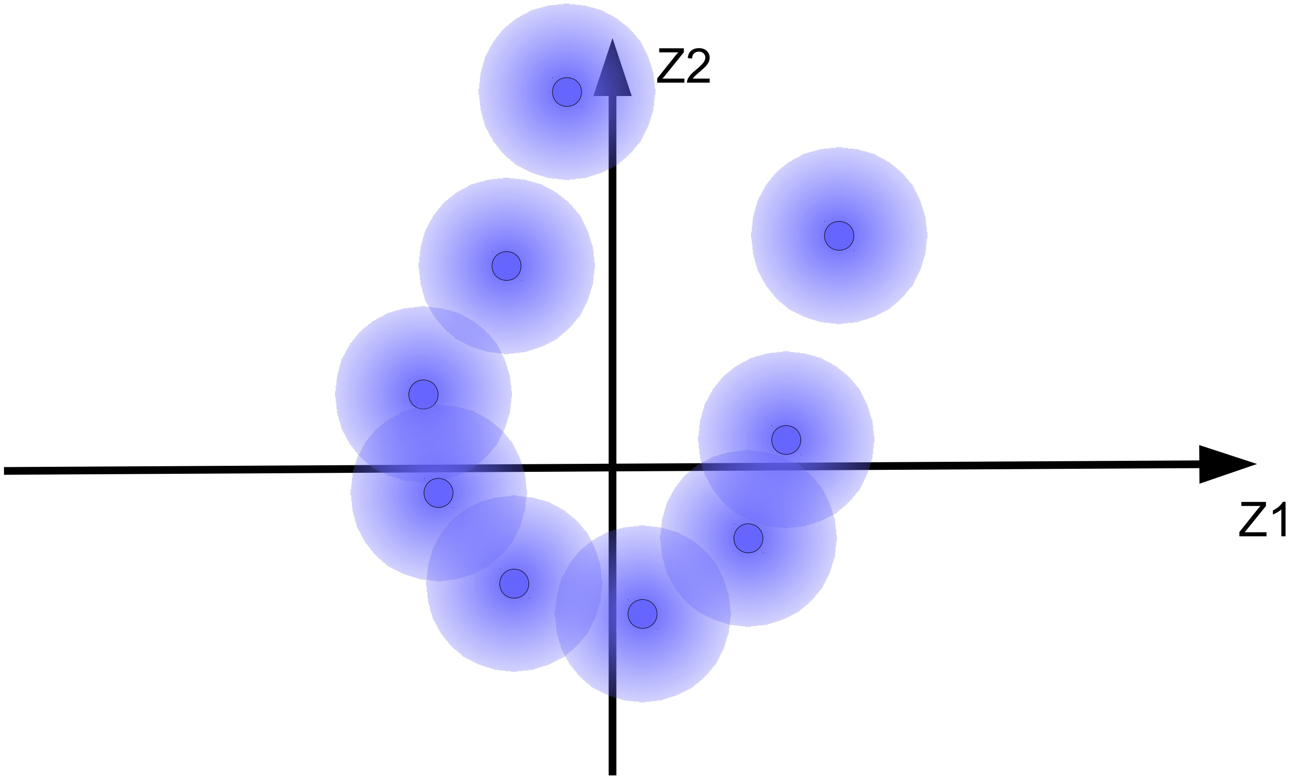
Variational Auto-Encoder

- **Code vectors for training samples**



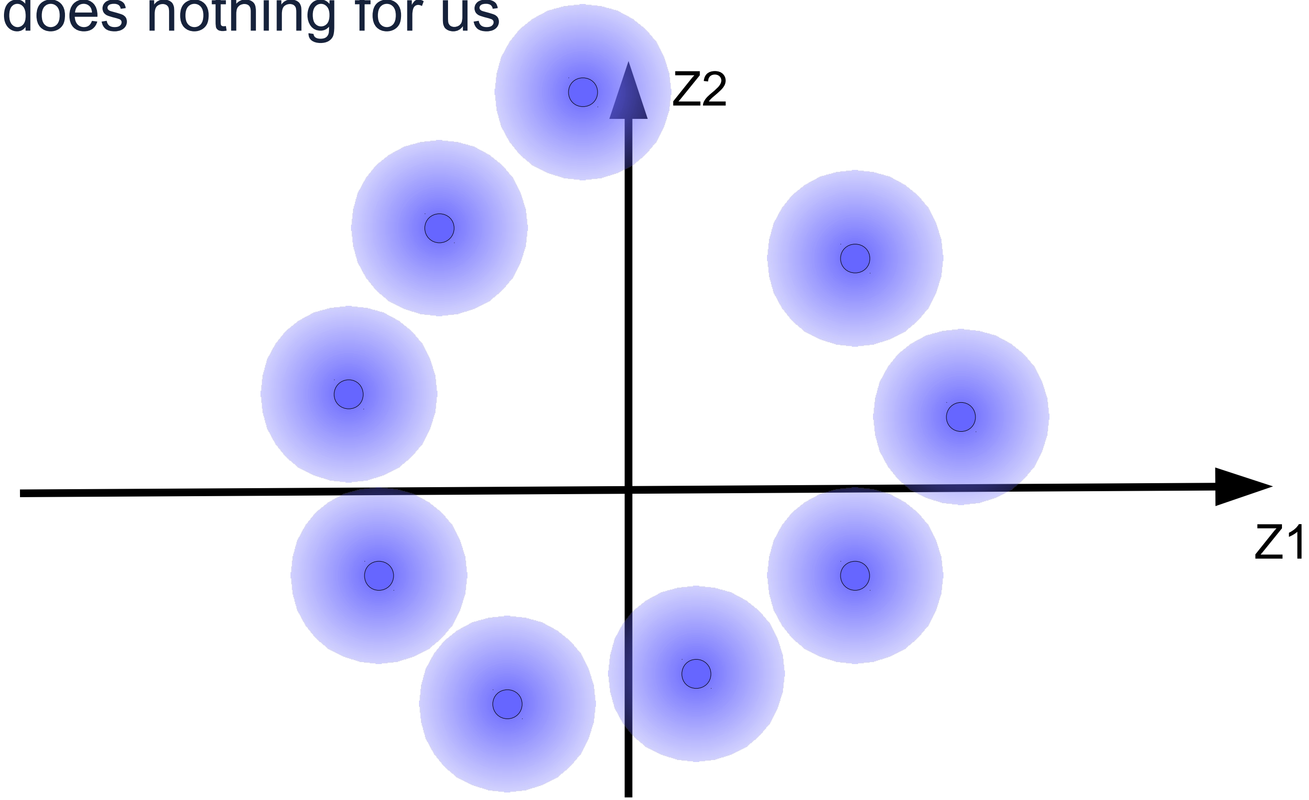
Variational Auto-Encoder

- **Code vectors for training sample with Gaussian noise**
  - Some fuzzy balls overlap, causing bad reconstructions



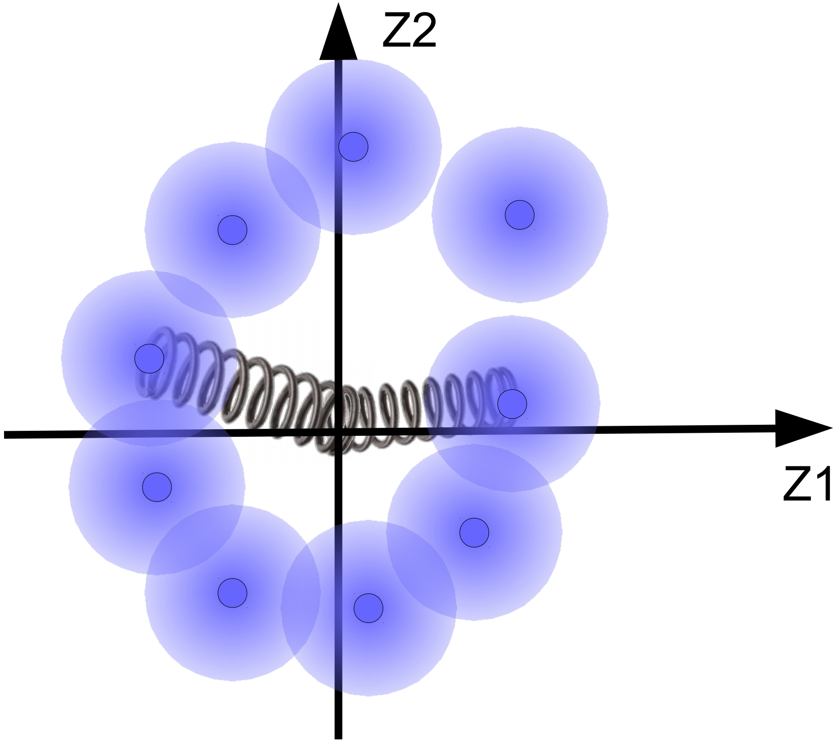
Variational Auto-Encoder

- **The code vectors want to move away from each other to minimize reconstruction error**
  - But that does nothing for us



Variational Auto-Encoder

- **Attach the balls to the center with a sping, so they don’t fly away**
  - Minimize the square distances of the balls to the origin
- **Center the balls around the origin**
  - Make the center of mass zero
- **Make the sizes of the balls close to 1 in each dimension**
  - Through a so-called KL term

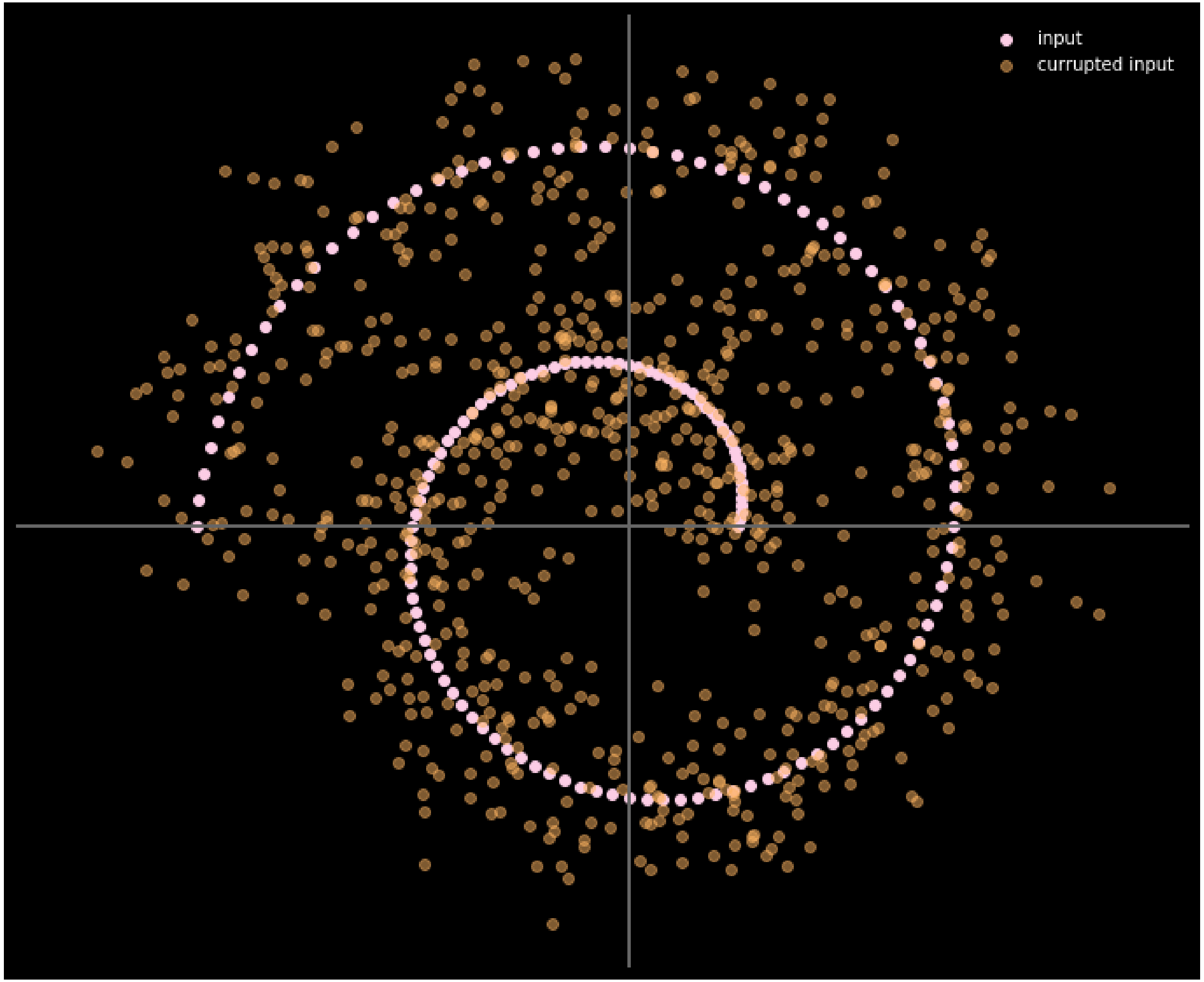


Denoising Auto-Encoders

Masked AE, BERT...

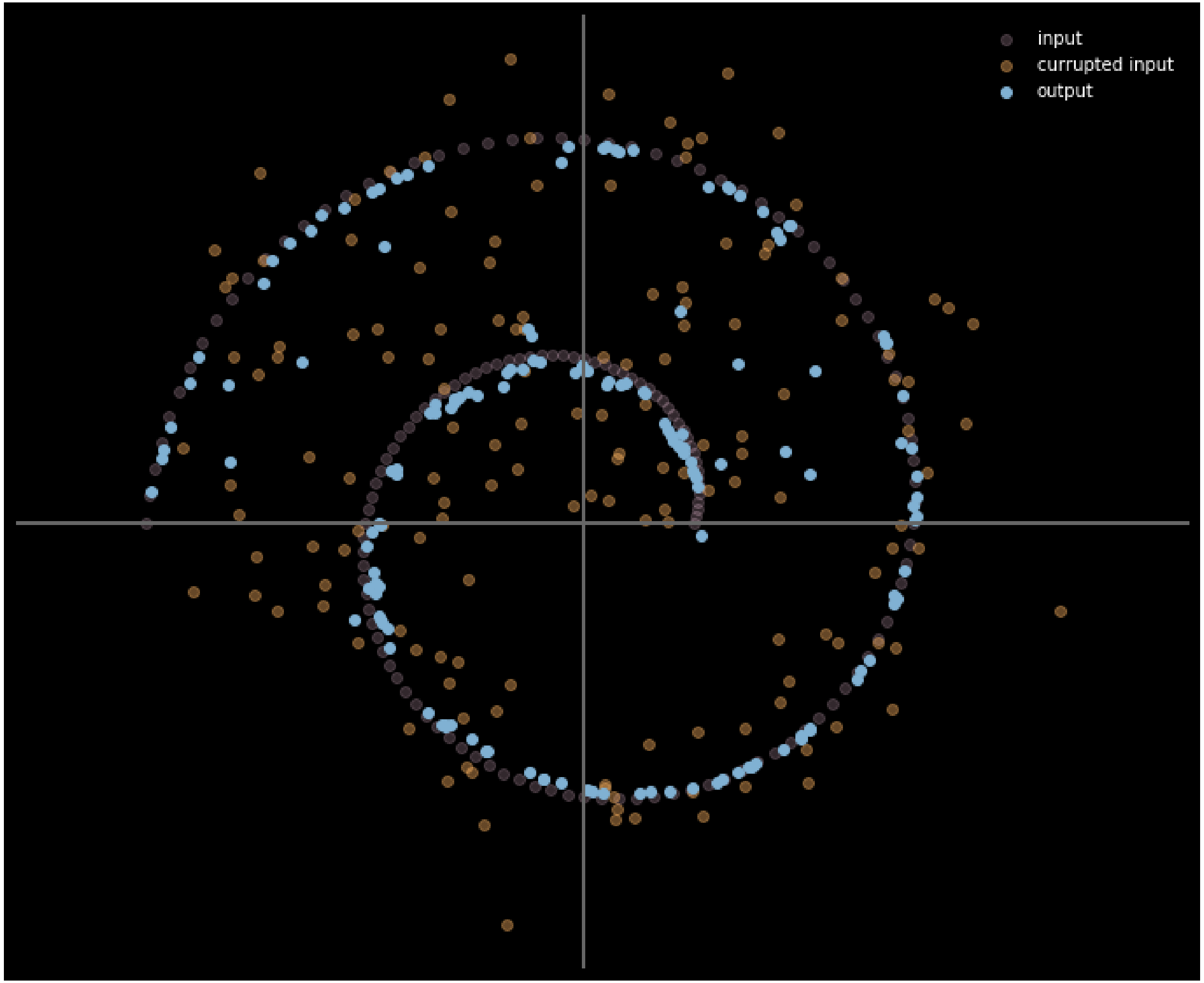
Denoising Auto-Encoder

- **Pink points:**
  - training samples
- **Orange points:**
  - corrupted training samples
  - Additive Gaussian noise
- **Figures credit: Alfredo Canziani**



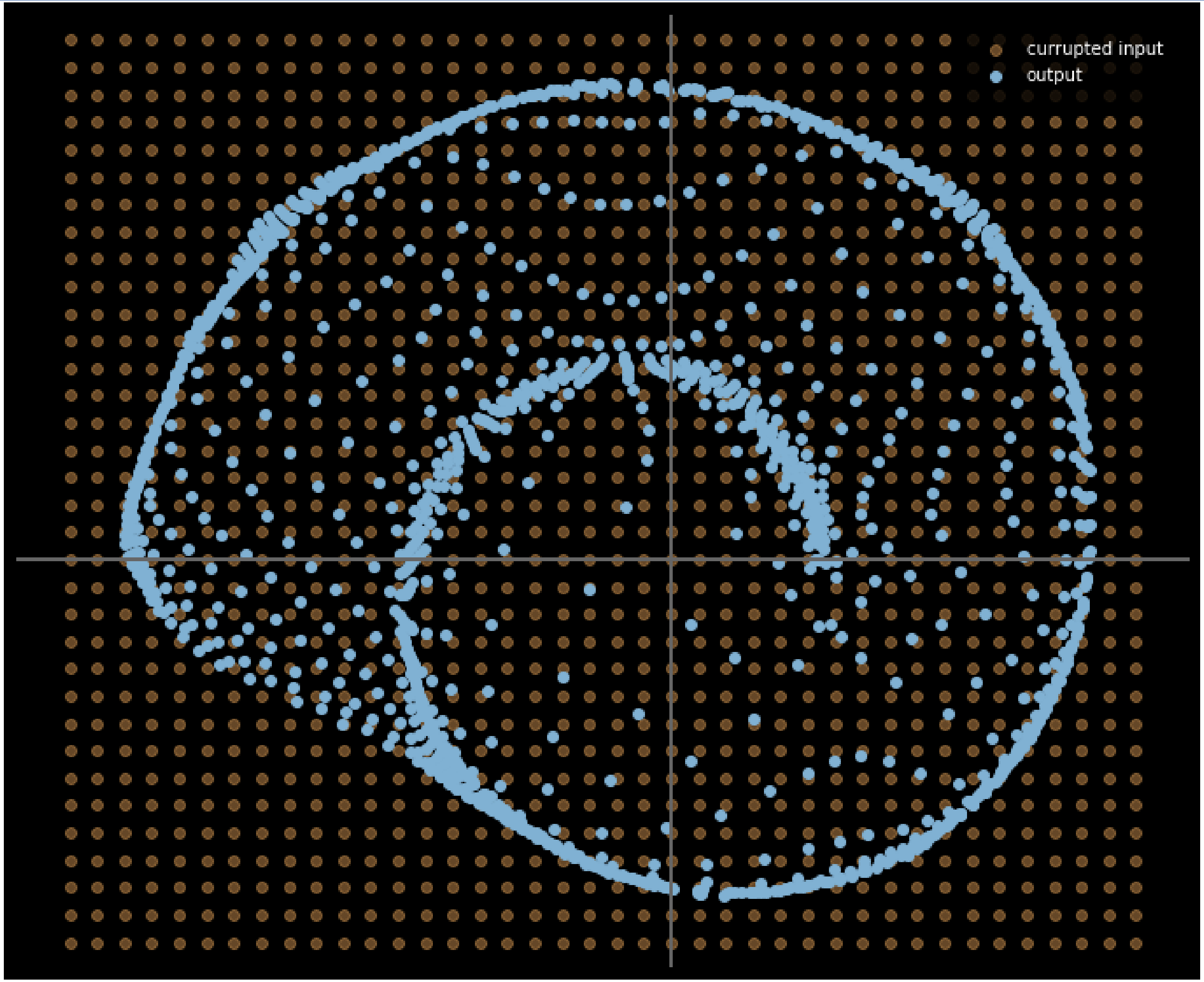
Data

- **Pink points:**
  - training samples
- **Orange points:**
  - corrupted training samples
  - Additive Gaussian noise
- **Blue points:**
  - Network outputs
  - Denoised samples



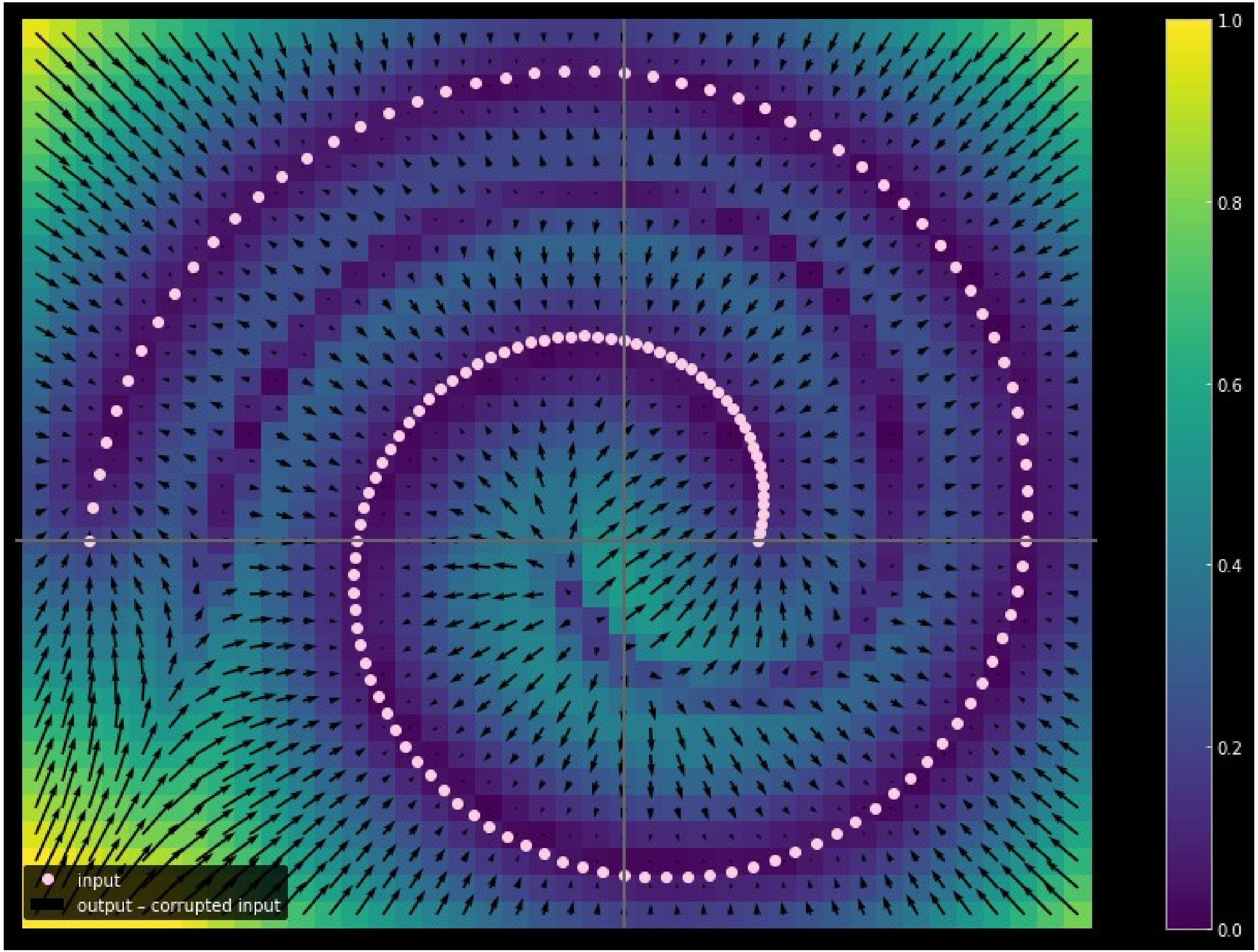
Data

- **Orange points:**
  - Test samples on a grid
- **Blue points:**
  - Network outputs



Data

- **Pink points:**
  - Training samples
- **Color:**
  - Recunstruction energy
- **Vector field:**
  - Displacement from network input to network output (scaled down)

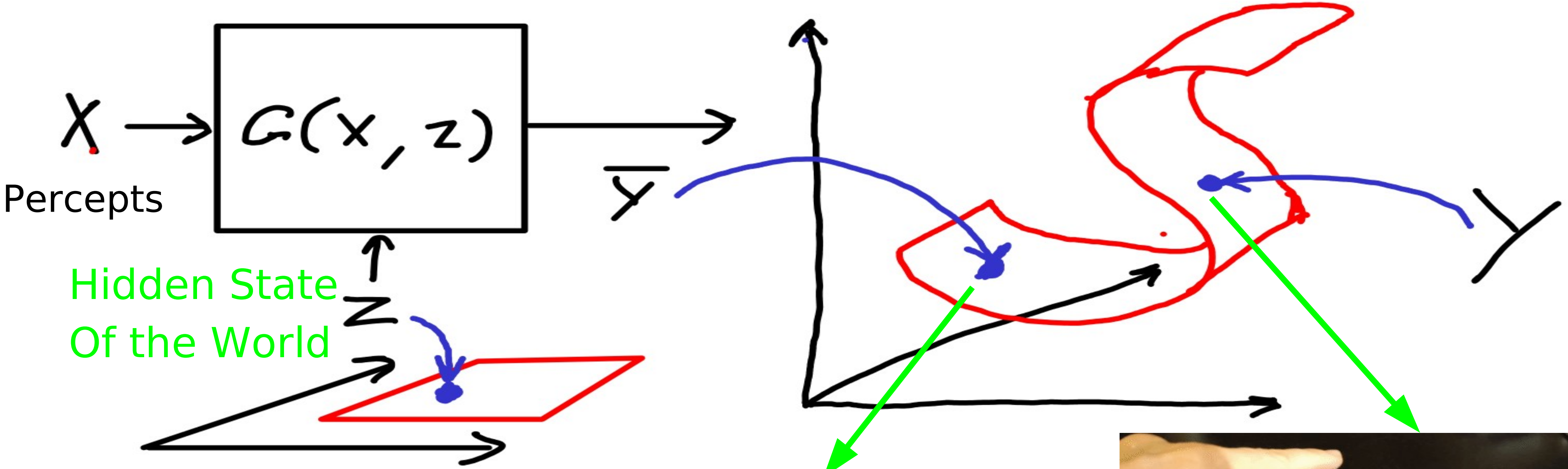


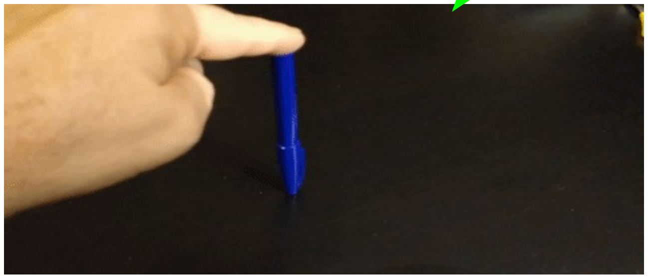
Adversarial Training & Video Prediction

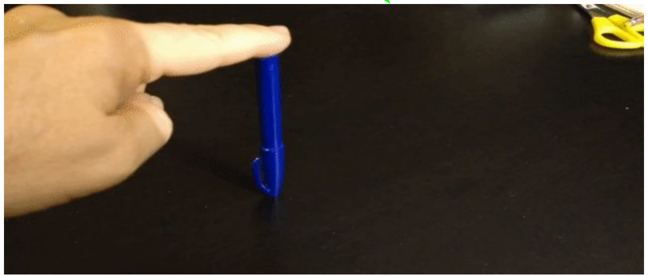
facebook
Artificial Intelligence Research

The Hard Part: Prediction Under Uncertainty

- **Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).**

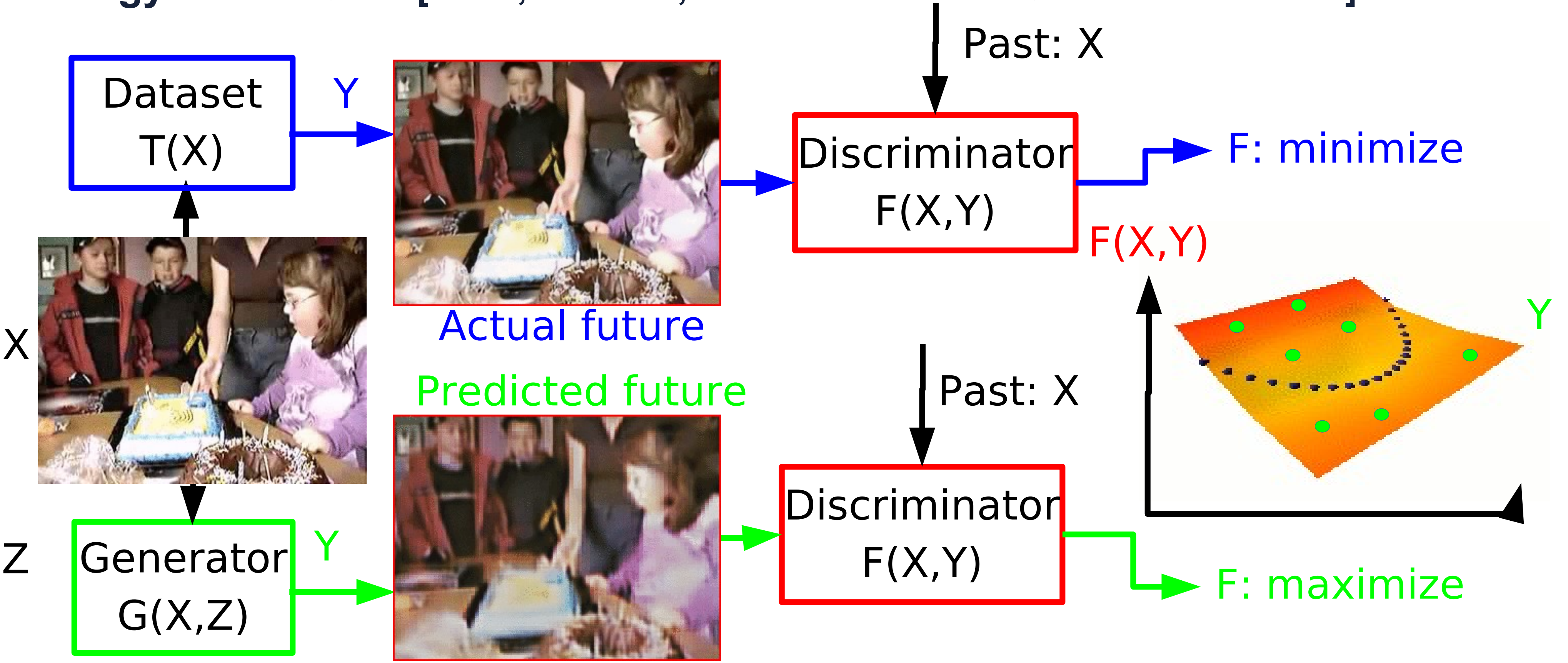






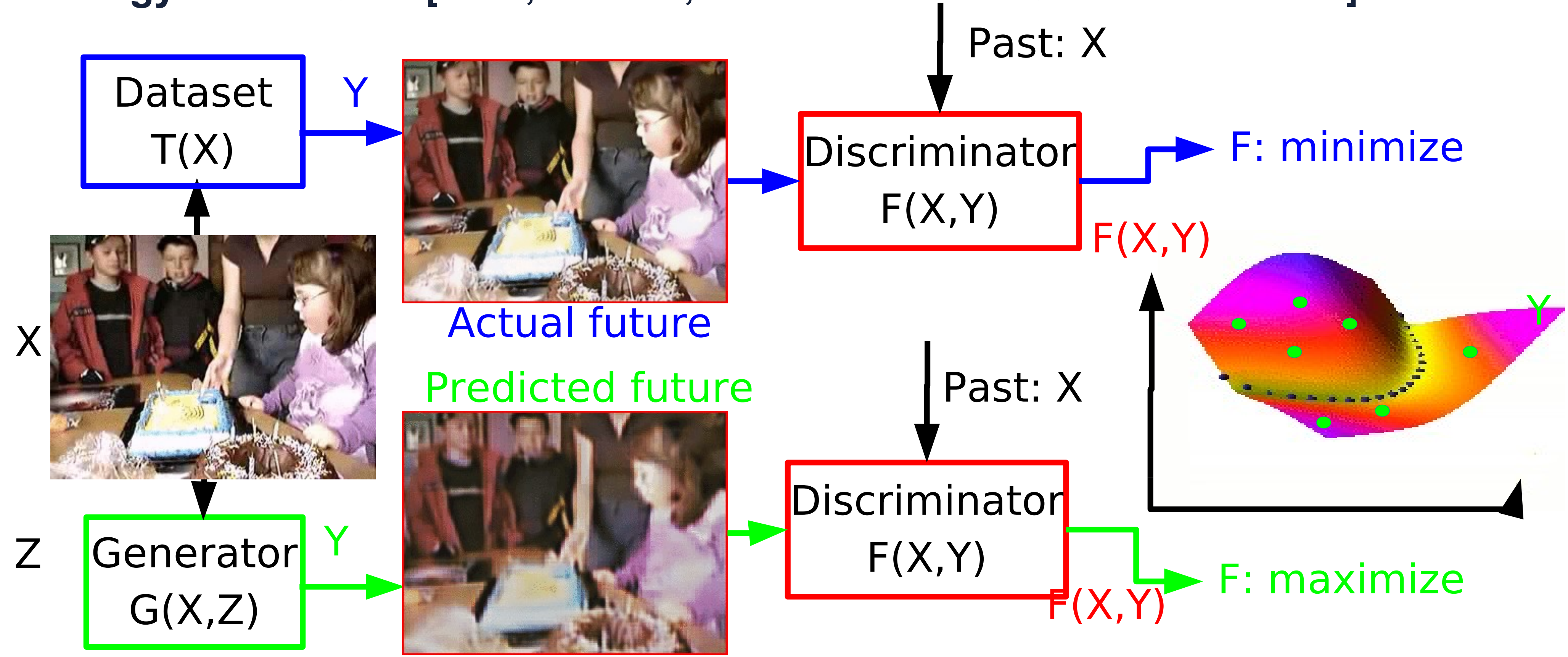
Adversarial Training: the key to prediction under uncertainty?

- **Generative Adversarial Networks (GAN)** [Goodfellow et al. NIPS 2014],
- **Energy-Based GAN** [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]



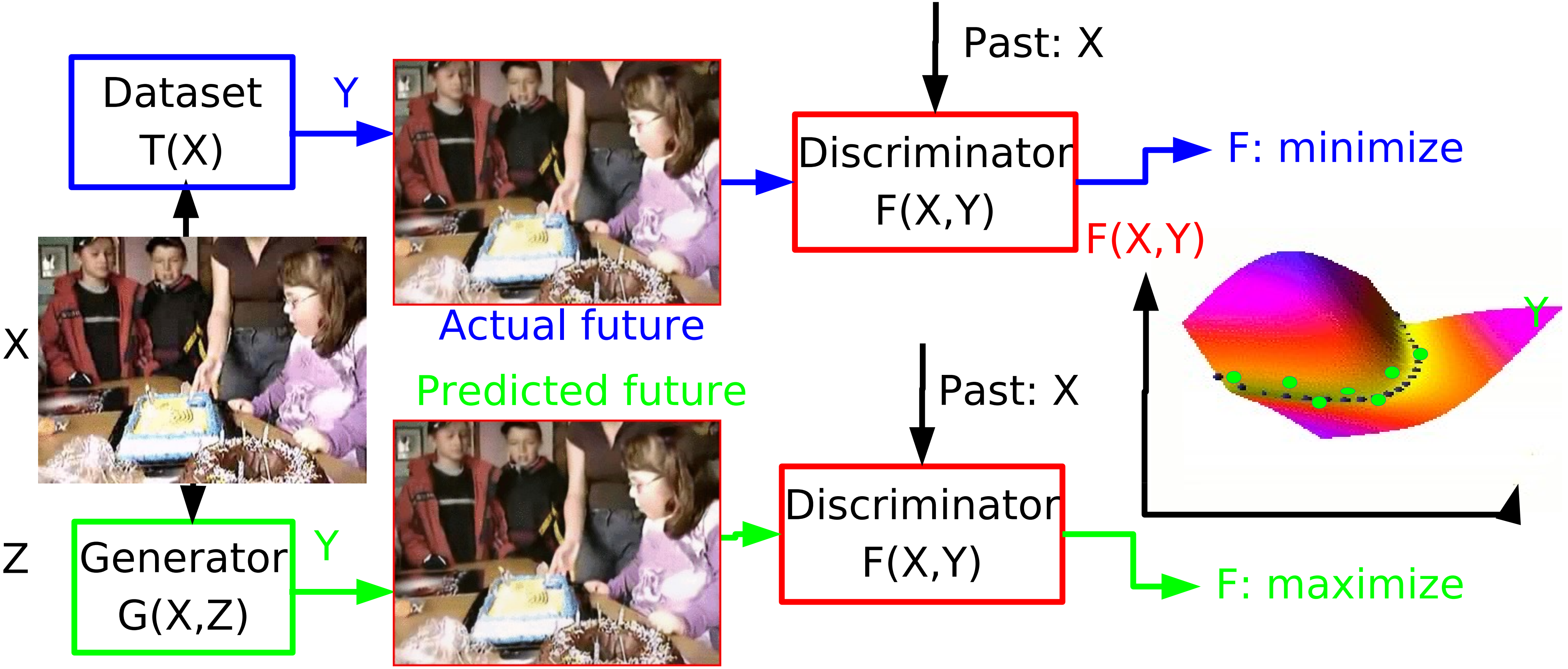
Adversarial Training: the key to prediction under uncertainty?

- **Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],**
- **Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]**



Adversarial Training: the key to prediction under uncertainty?

- **Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],**
- **Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]**



Faces "invented" by a GAN (Generative Adversarial Network)

- **Random vector → Generator Network → output image** **[Goodfellow NIPS 2014]**
  [Karras et al. ICLR 2018] (from NVIDIA)

Generative Adversarial Networks for Creation

- [Sbai 2017]

Self-supervised Adversarial Learning for Video Prediction

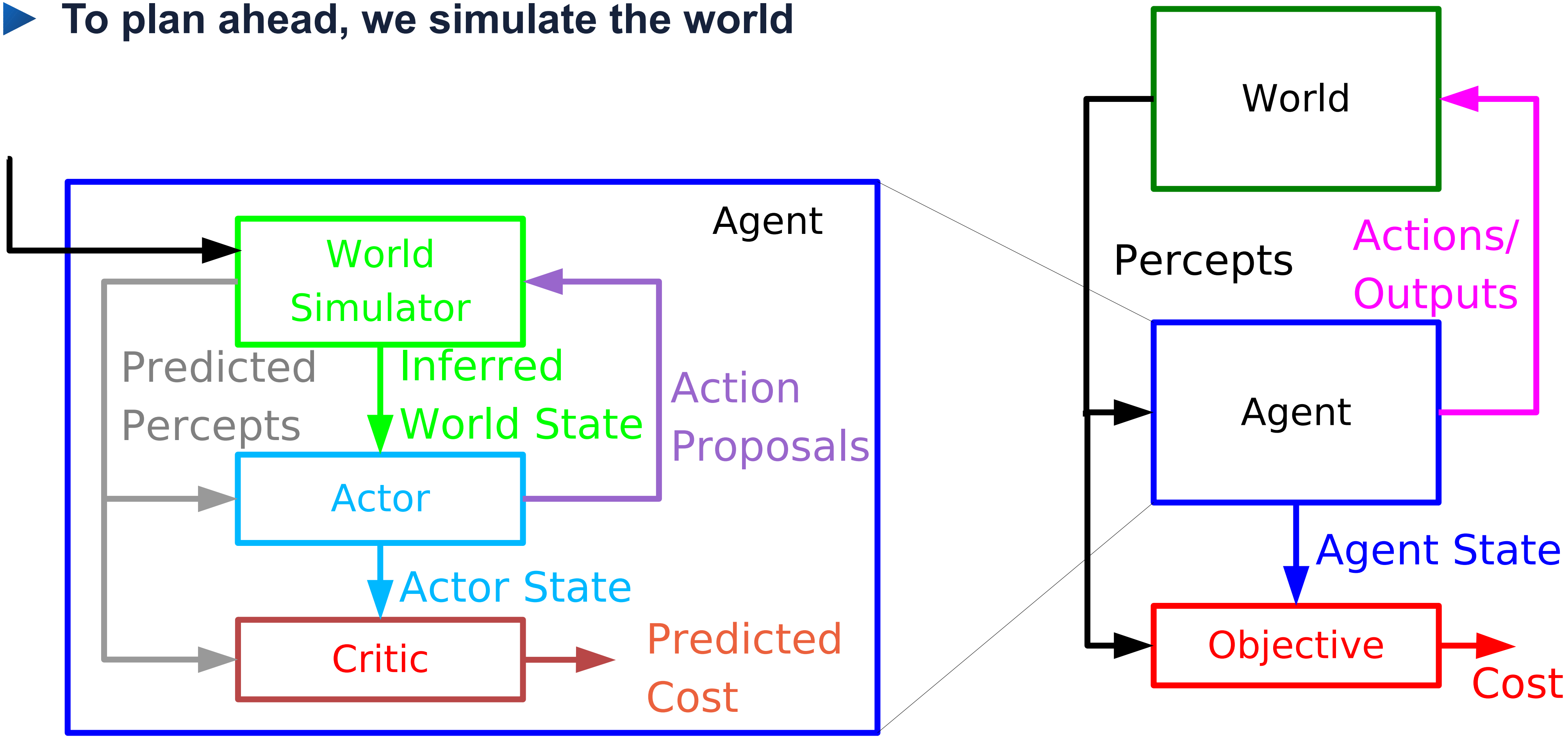
- **Our brains are "prediction machines"**
- **Can we train machines to predict the future?**
- **Some success with "adversarial training"**
  - [Mathieu, Couprie, LeCun arXiv:1511:05440]
- **But we are far from a complete solution.**

Learning Models of the World

Learning motor skills with no interaction with the real world
[Henaff, Canziani, LeCun ICLR 2019]
[Henaff, Zhao, LeCun ArXiv:1711.04994]
[Henaff, Whitney, LeCun Arxiv:1705.07177]

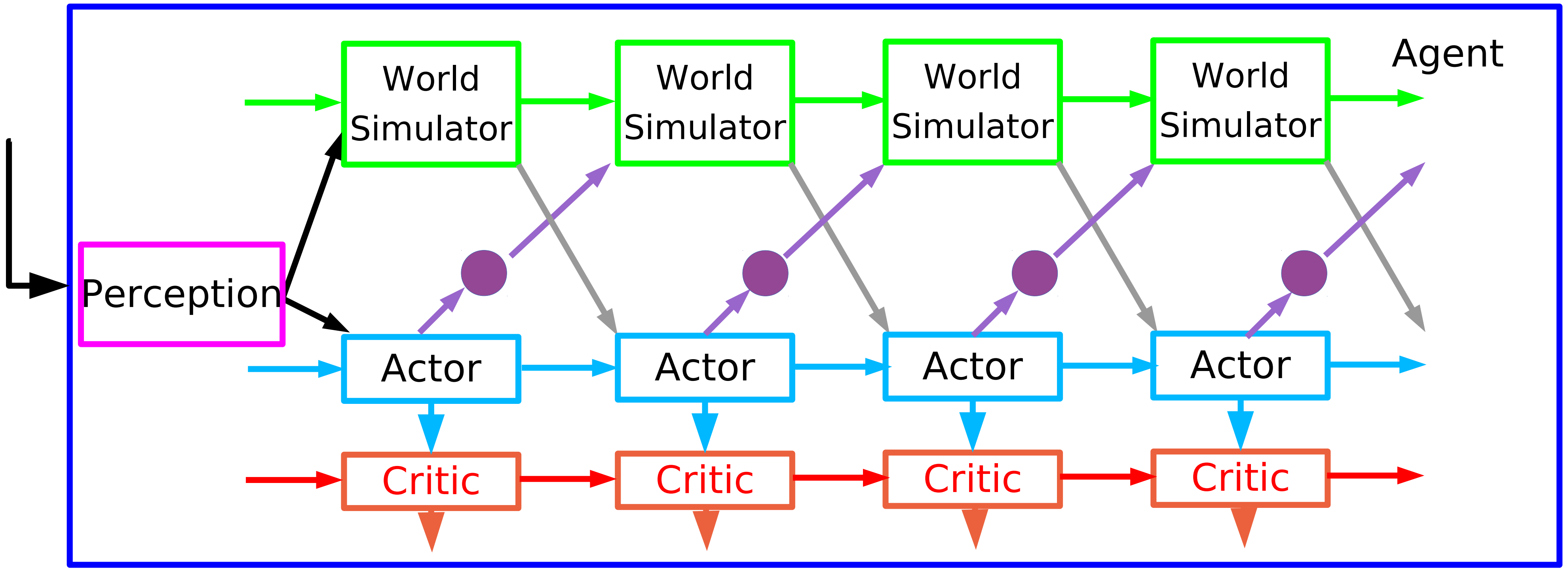
Planning Requires Prediction

- To plan ahead, we simulate the world



Training the Actor with Optimized Action Sequences

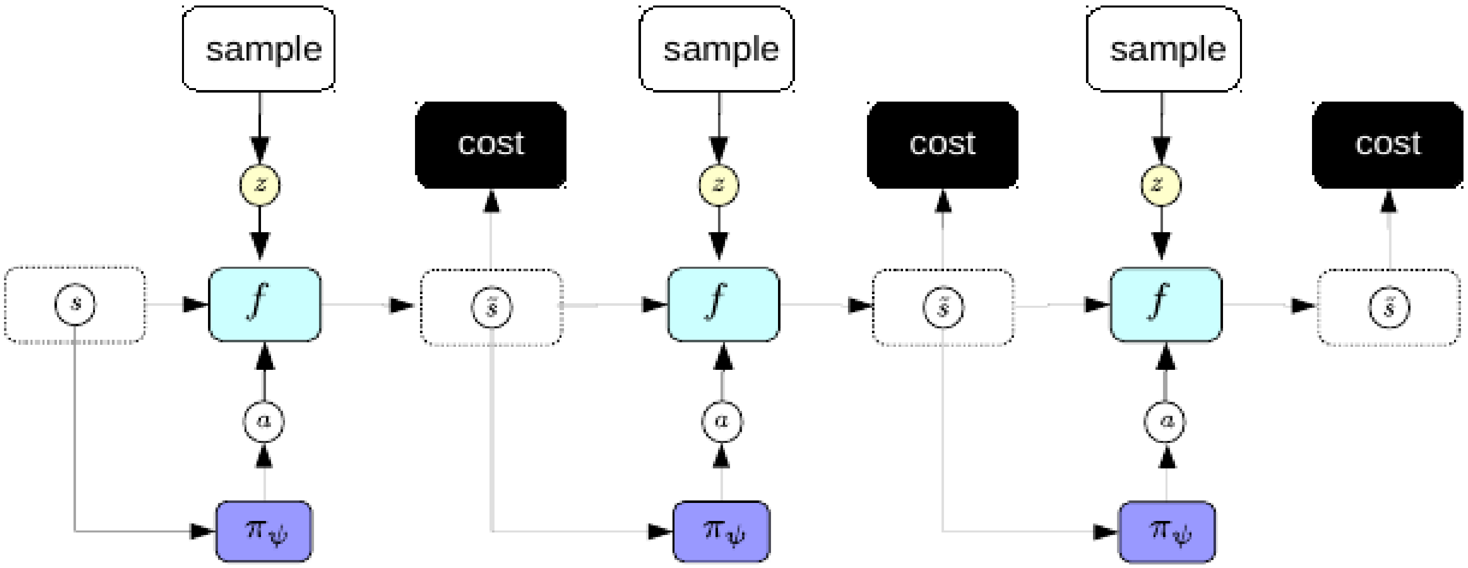
- **1. Find action sequence through optimization**
- **2. Use sequence as target to train the actor**
  - Over time we get a compact policy that requires no run-time optimization



Planning/learning using a self-supervised predictive world model

- **Feed initial state**
- **Run the forward model**
- **Backpropagate gradient of cost**
- **Act**
  - (model-predictive control)

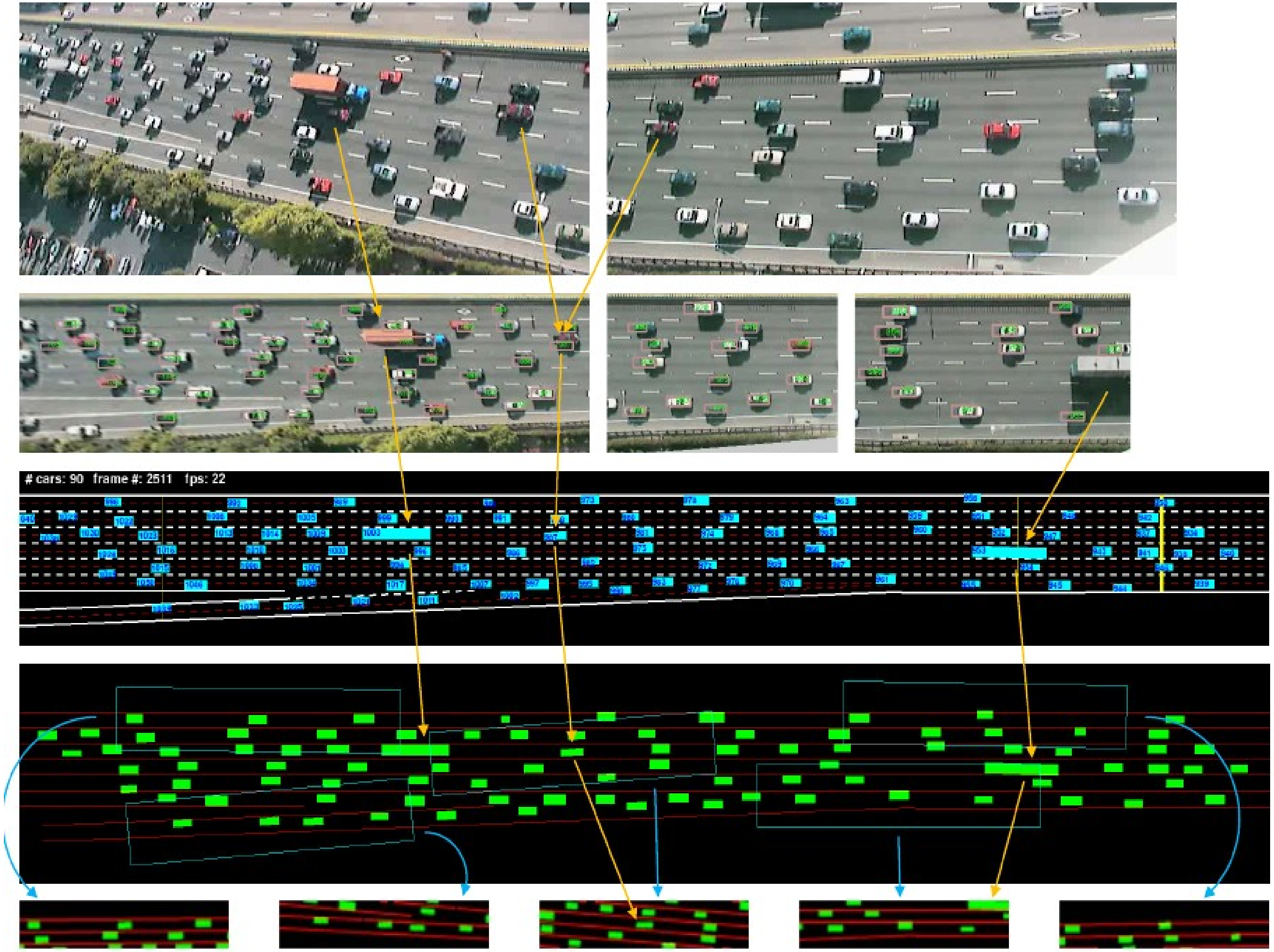
  or
- **Use the gradient t train a policy network.**
- **Iterate**



Stochastic policy network (optimized)

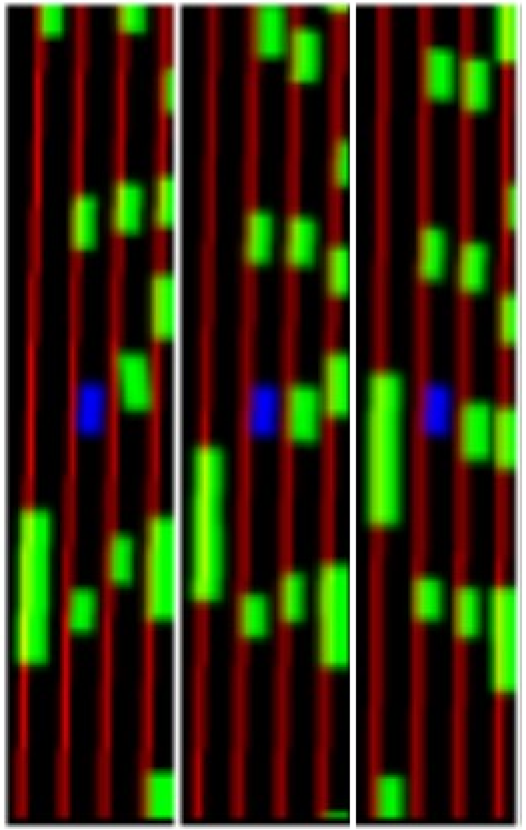
Using Forward Models to Plan (and to learn to drive)

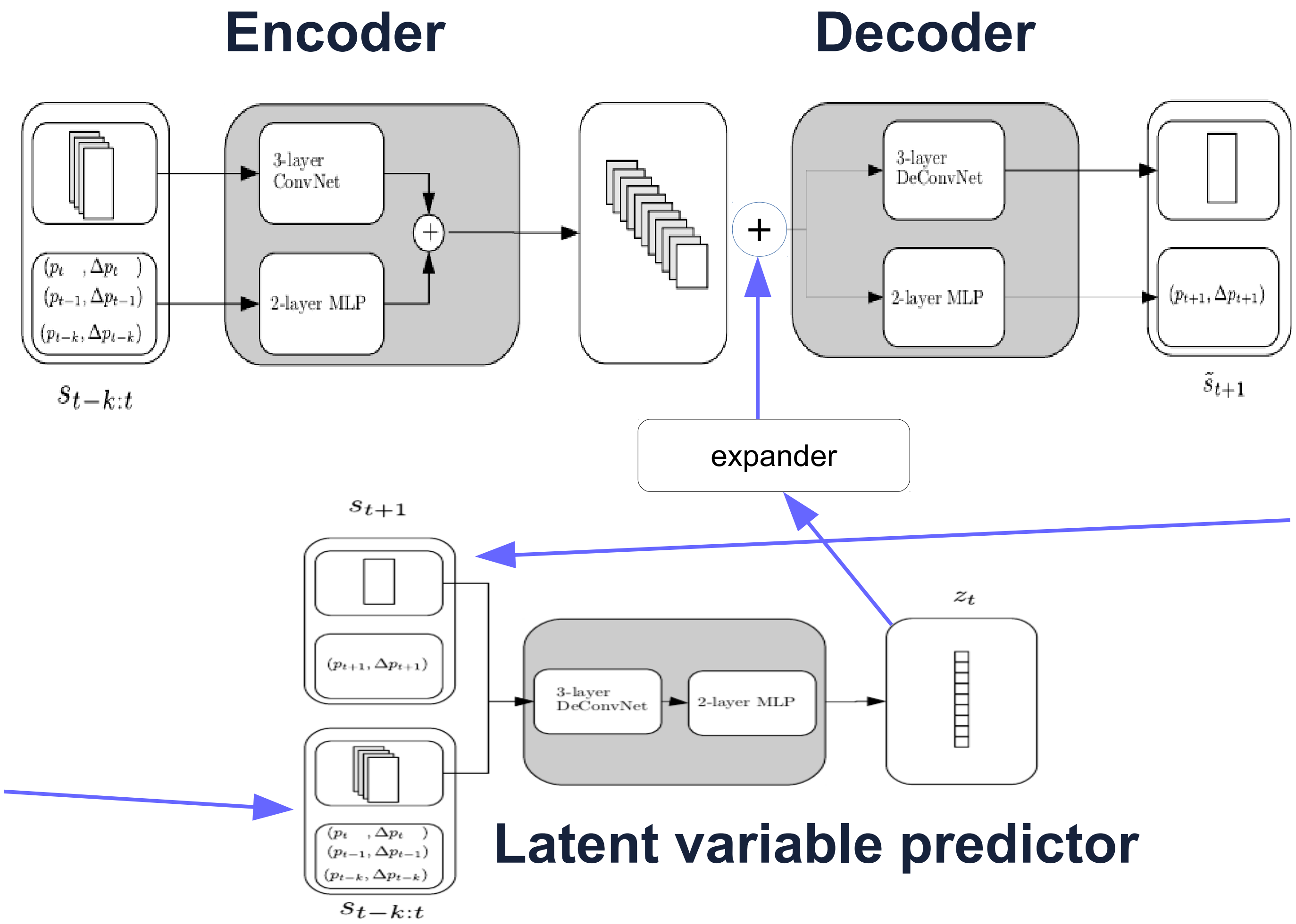
- **Overhead camera on highway.**
  - Vehicles are tracked
- **A “state” is a pixel representation of a rectangular window centered around each car.**
- **Forward model is trained to predict how every car moves relative to the central car.**
  - steering and acceleration are computed

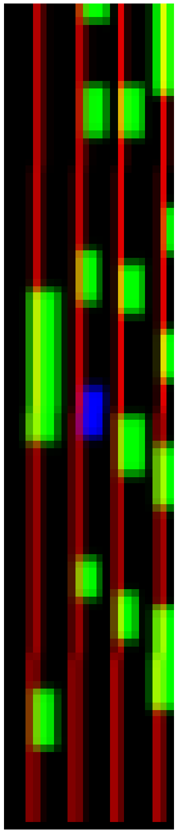


Forward Model Architecture

- **Architecture:**

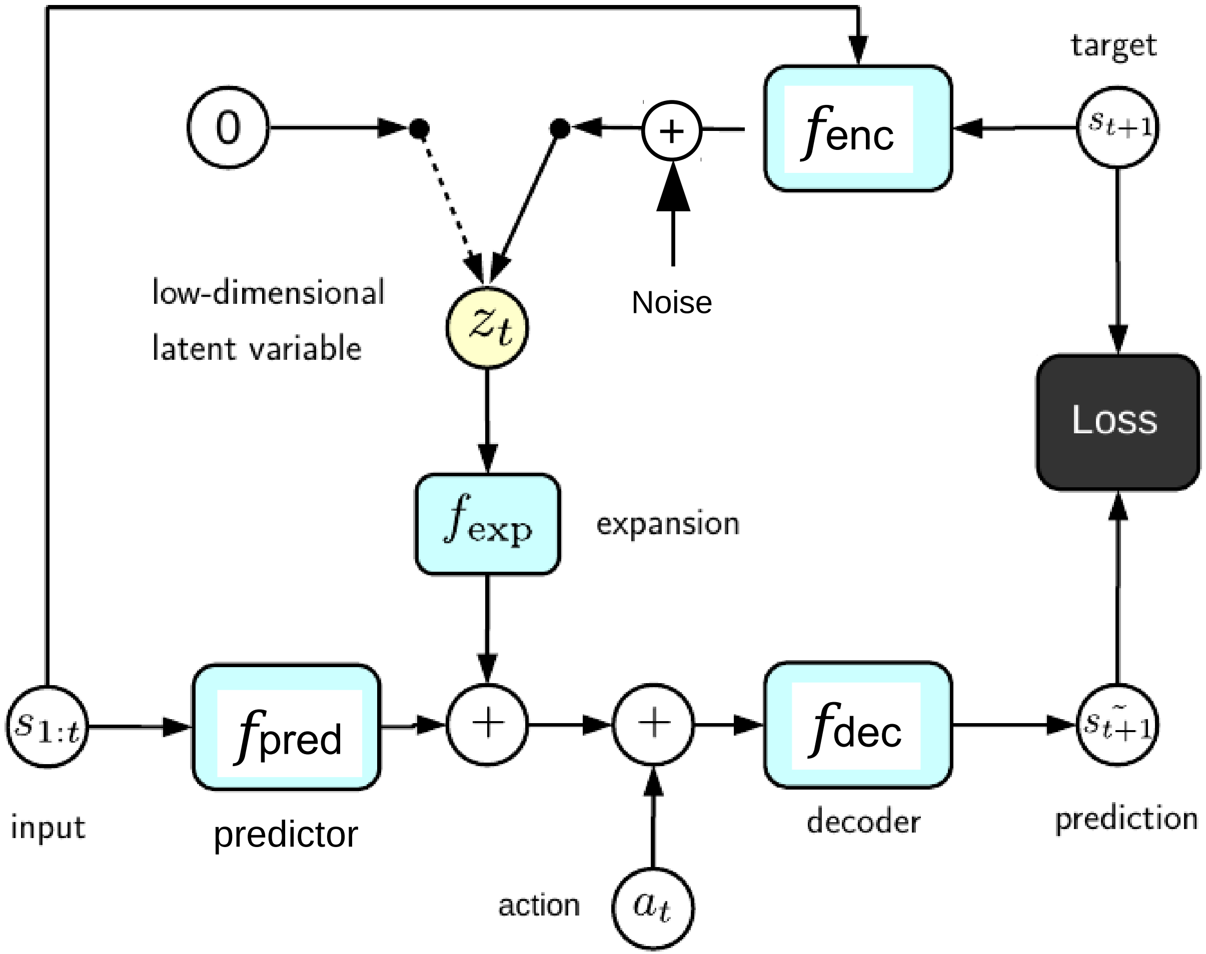




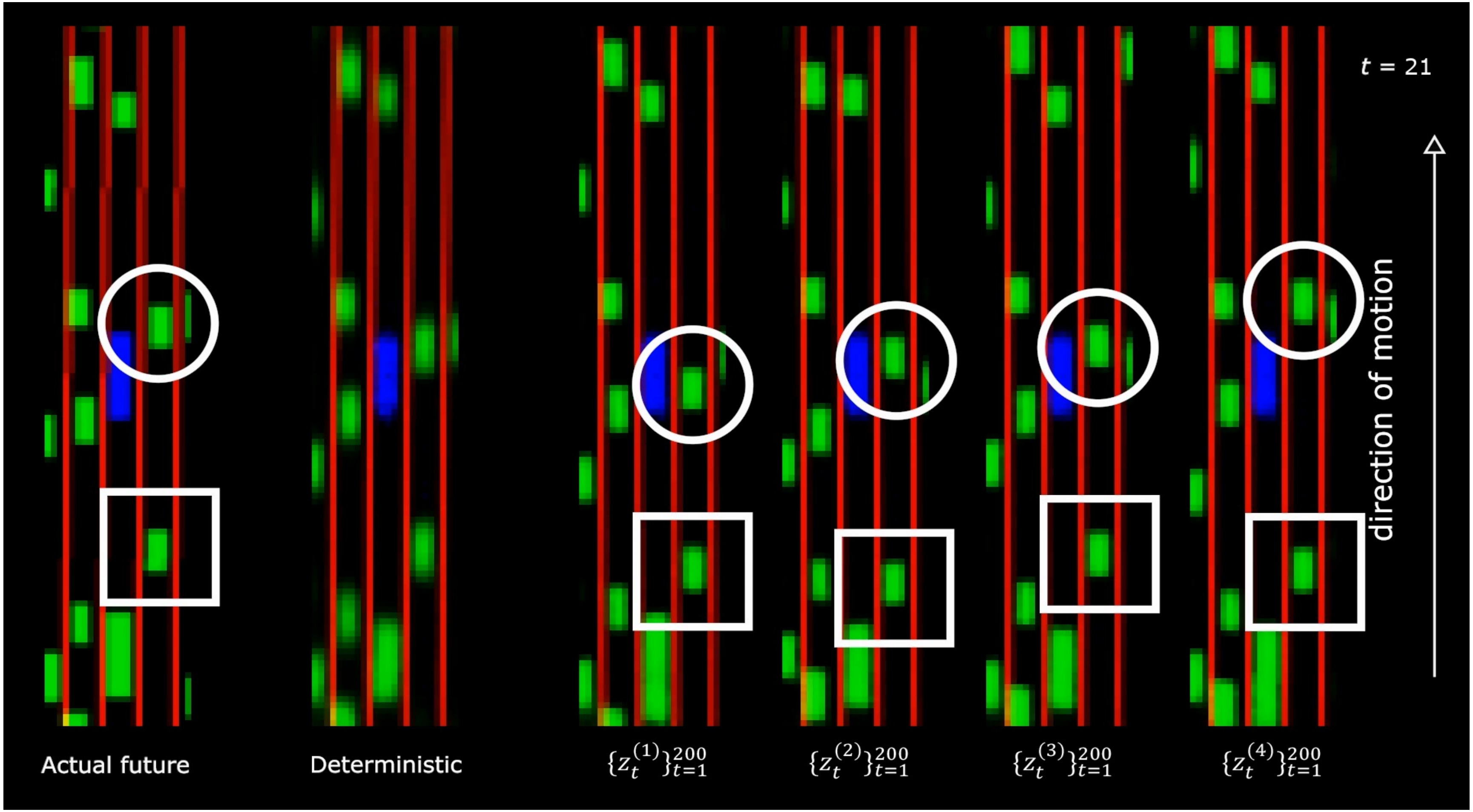


Stochastic Forward Modeling: regularized latent variable model

- **Latent variable is predicted from the target.**
- **The latent variable is set to zero half the time during training (drop out) and corrupted with noise**
- **The model predicts as much as it can without the latent var.**
- **The latent var corrects the residual error.**

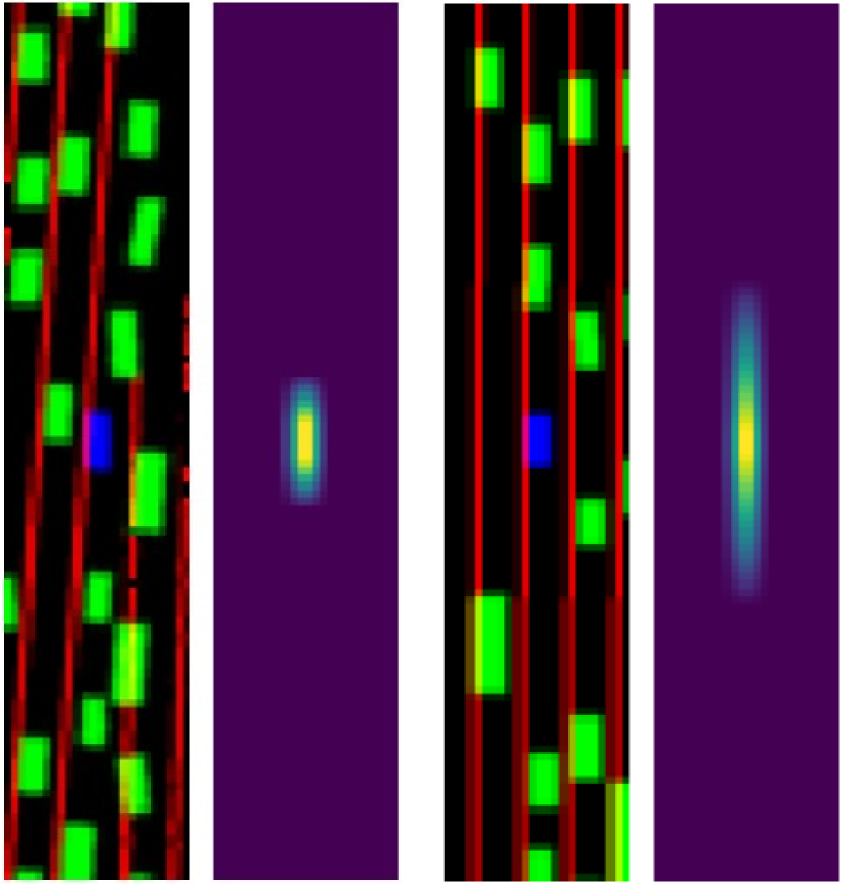


Actual, Deterministic, VAE+Dropout Predictor/encoder



Cost optimized for Planning & Policy Learning

- **Differentiable cost function**
  - Increases as car deviates from lane
  - Increases as car gets too close to other cars nearby in a speed-dependent way
- **Uncertainty cost:**
  - Increases when the costs from multiple predictions (obtained through sampling of drop-out) have high variance.
  - Prevents the system from exploring unknown/unpredictable configurations that may have low cost.

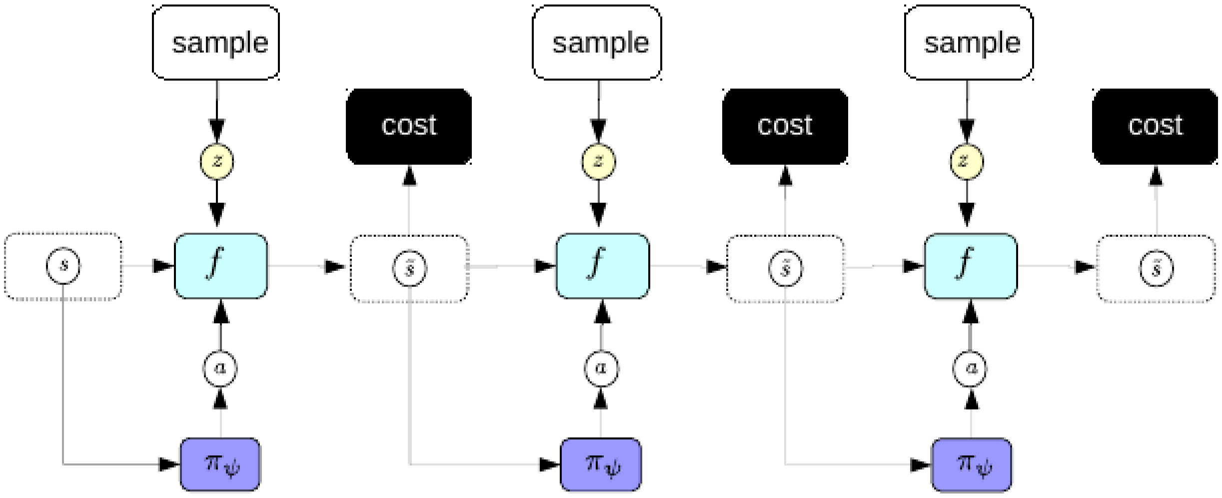


(a) 19.8 km/h (b) 50.3 km/h

Learning to Drive by Simulating it in your Head

- Feed initial state
- Sample latent variable sequences of length 20
- Run the forward model with these sequences
- Backpropagate gradient of cost to train a policy network.
- Iterate

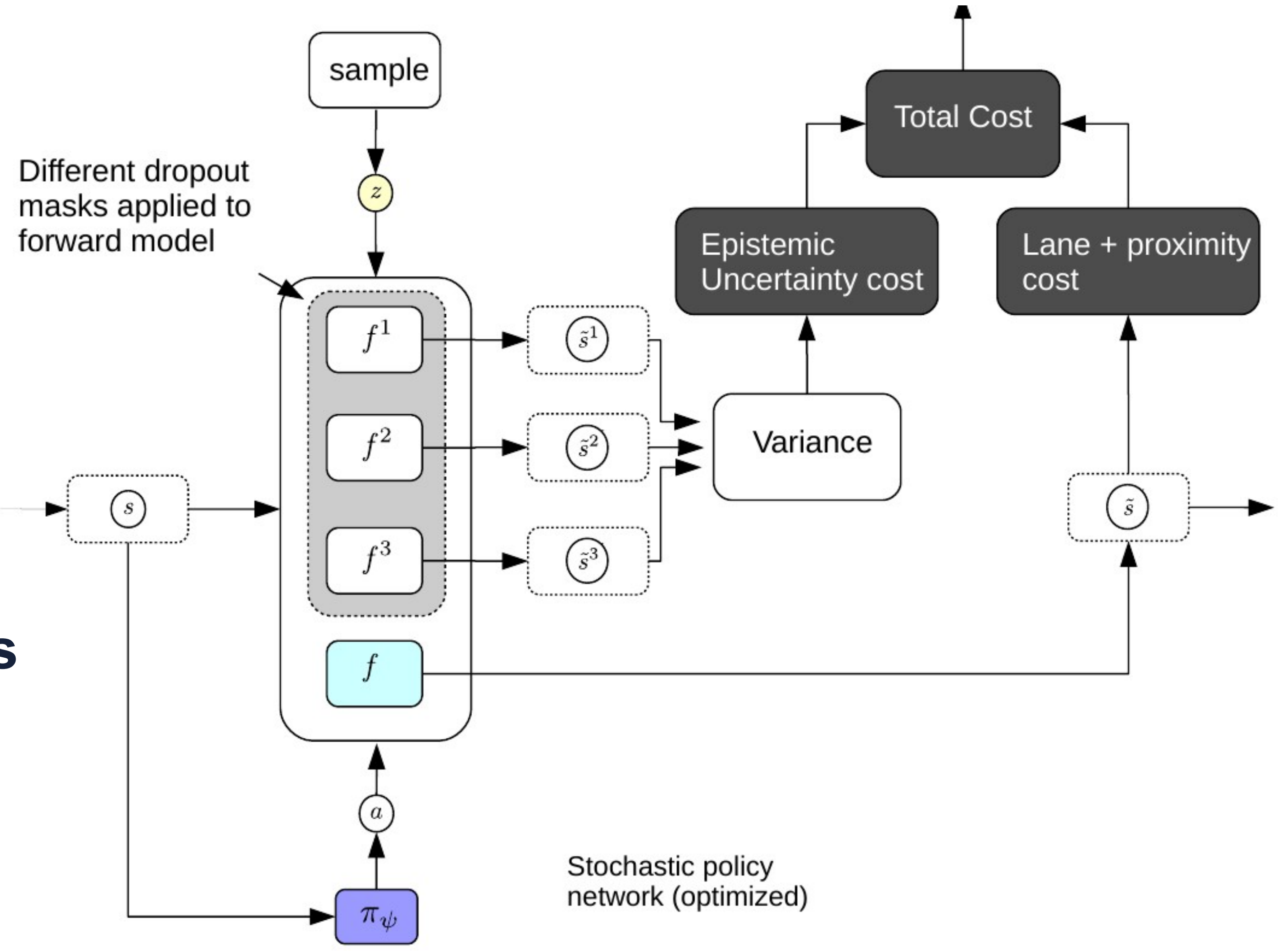
- No need for planning at run time.



Stochastic policy network (optimized)

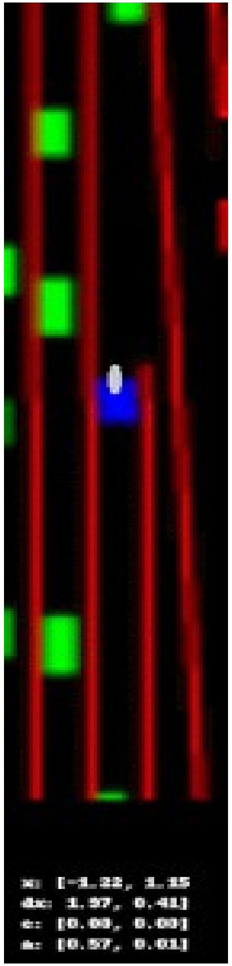
Adding an Uncertainty Cost (doesn't work without it)

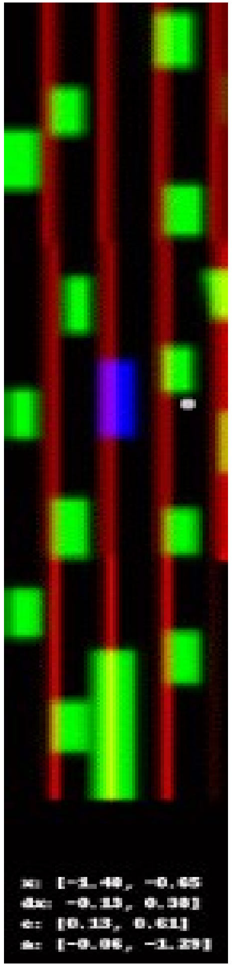
- **Estimates epistemic uncertainty**
- **Samples multiple drop-puts in forward model**
- **Computes variance of predictions (differentiably)**
- **Train the policy network to minimize the lane&proximity cost plus the uncertainty cost.**
- **Avoids unpredictable outcomes**

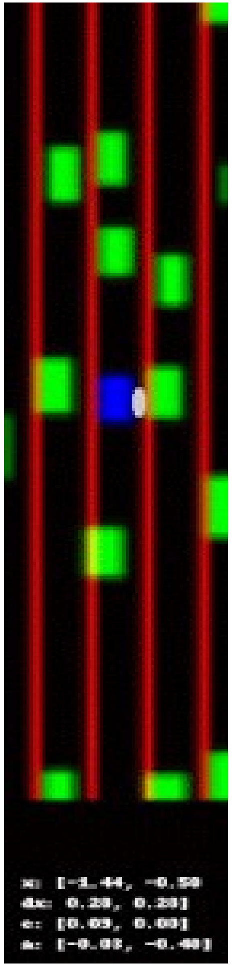


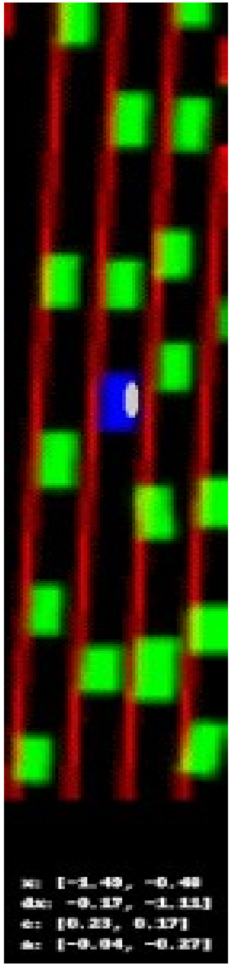
Y. LeCun

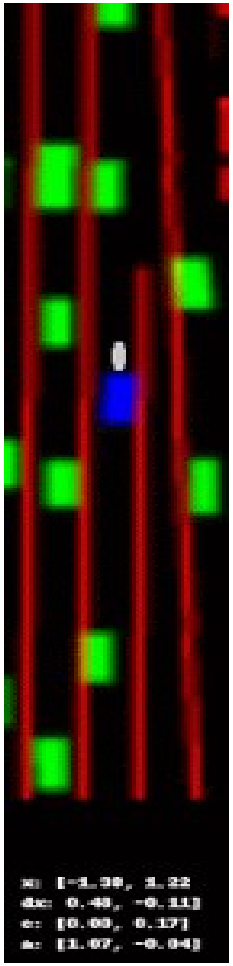
Driving an Invisible Car in “Real” Traffic





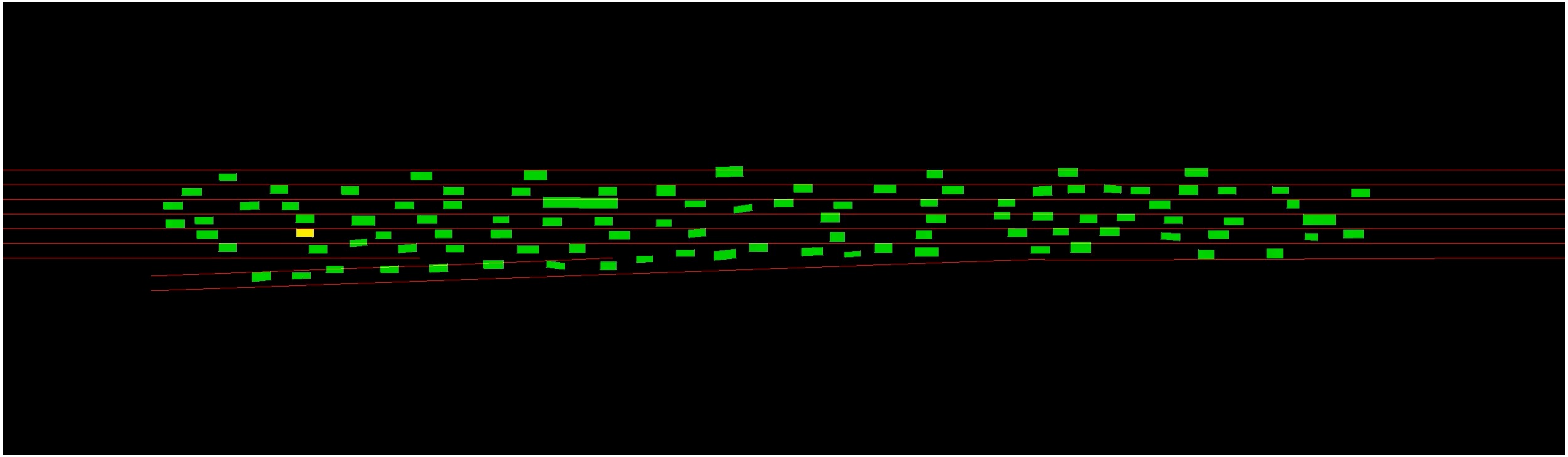






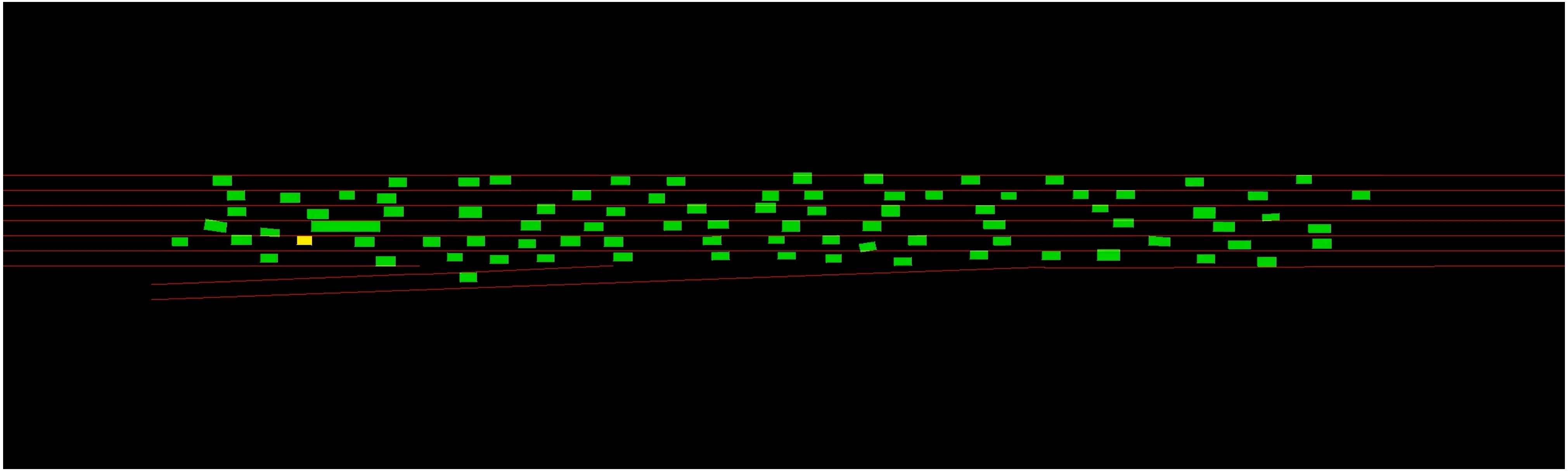
Driving!

- Yellow: real car
- Blue: bot-driven car



Driving!

- **Yellow: real car**
- **Blue: bot-driven car**



Thank you

facebook
Artificial Intelligence Research