Deep Learning

Yann Le Cun

Facebook AI Research

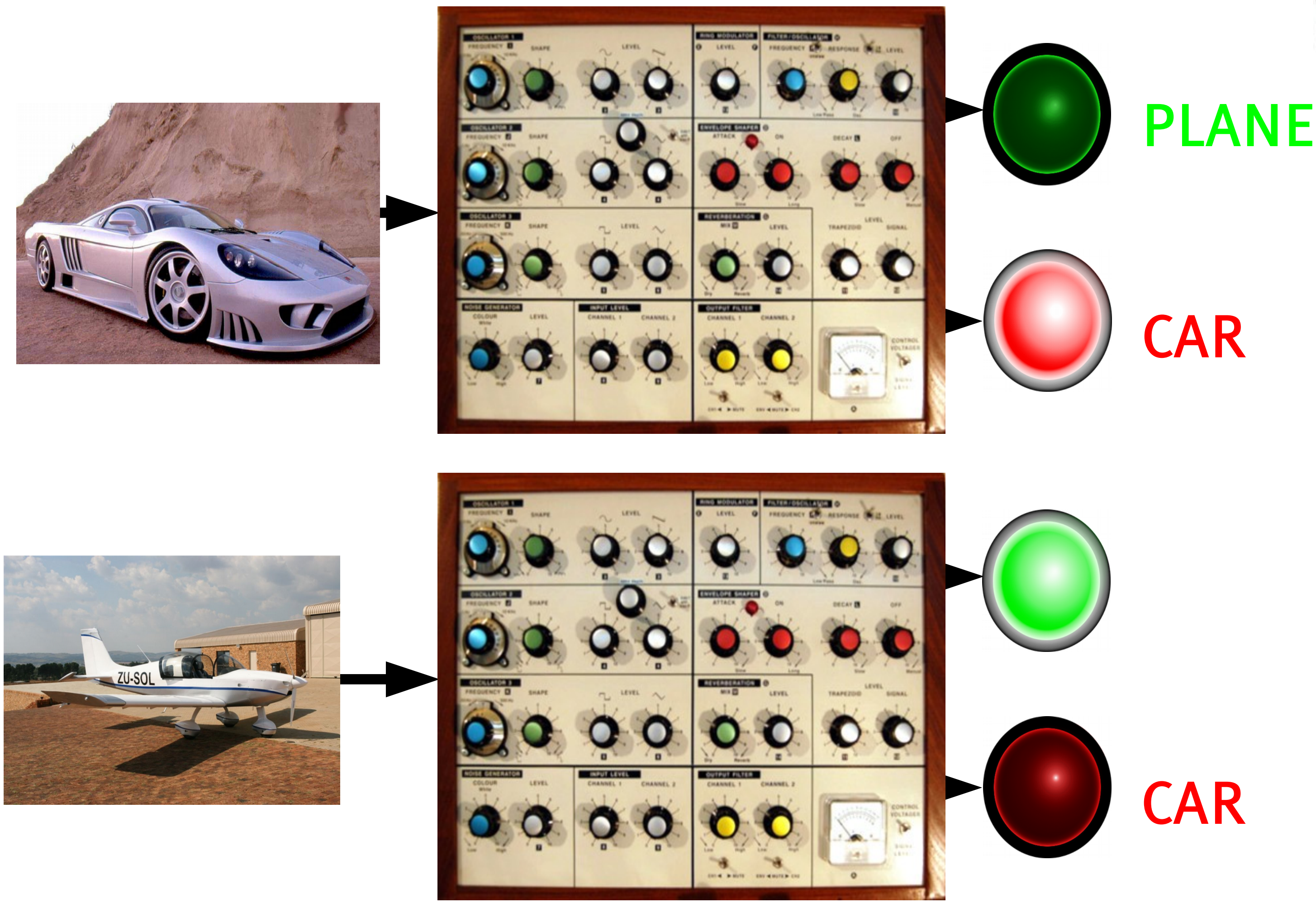
Center for Data Science, NYU

Courant Institute of Mathematical Sciences, NYU

http://yann.lecun.com

Supervised Learning

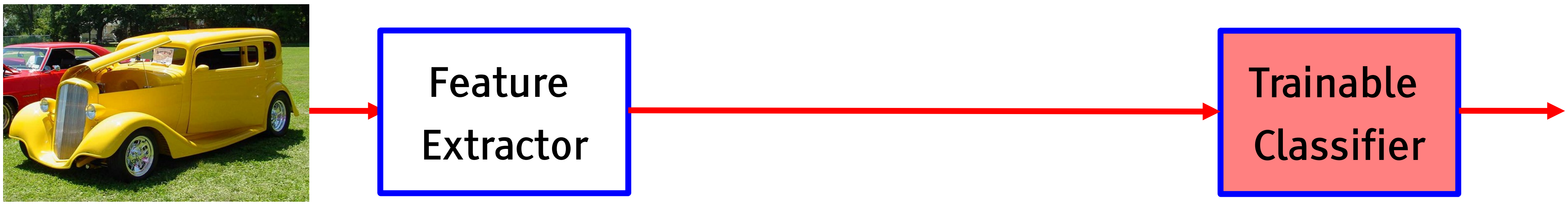
- We can train a machine on lots of examples of tables, chairs, dog, cars, and people
- But will it recognize table, chairs, dogs, cars, and people it has never seen before?



Deep Learning = The Entire Machine is Trainable

Y LeCun

- **Traditional Pattern Recognition:** Fixed/Handcrafted Feature Extractor



- **Mainstream Modern Pattern Recognition:** Unsupervised mid-level features

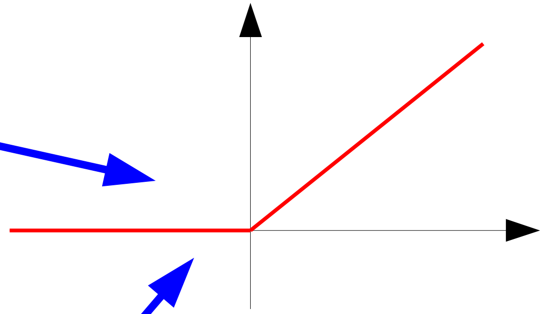
- **Deep Learning:** Representations are hierarchical and trained

Multi-Layer Neural Nets

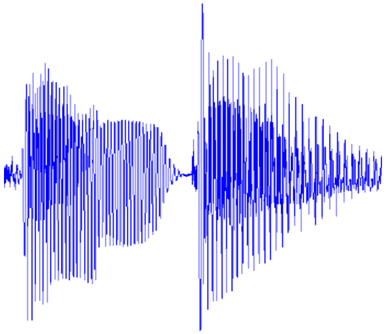
Y LeCun

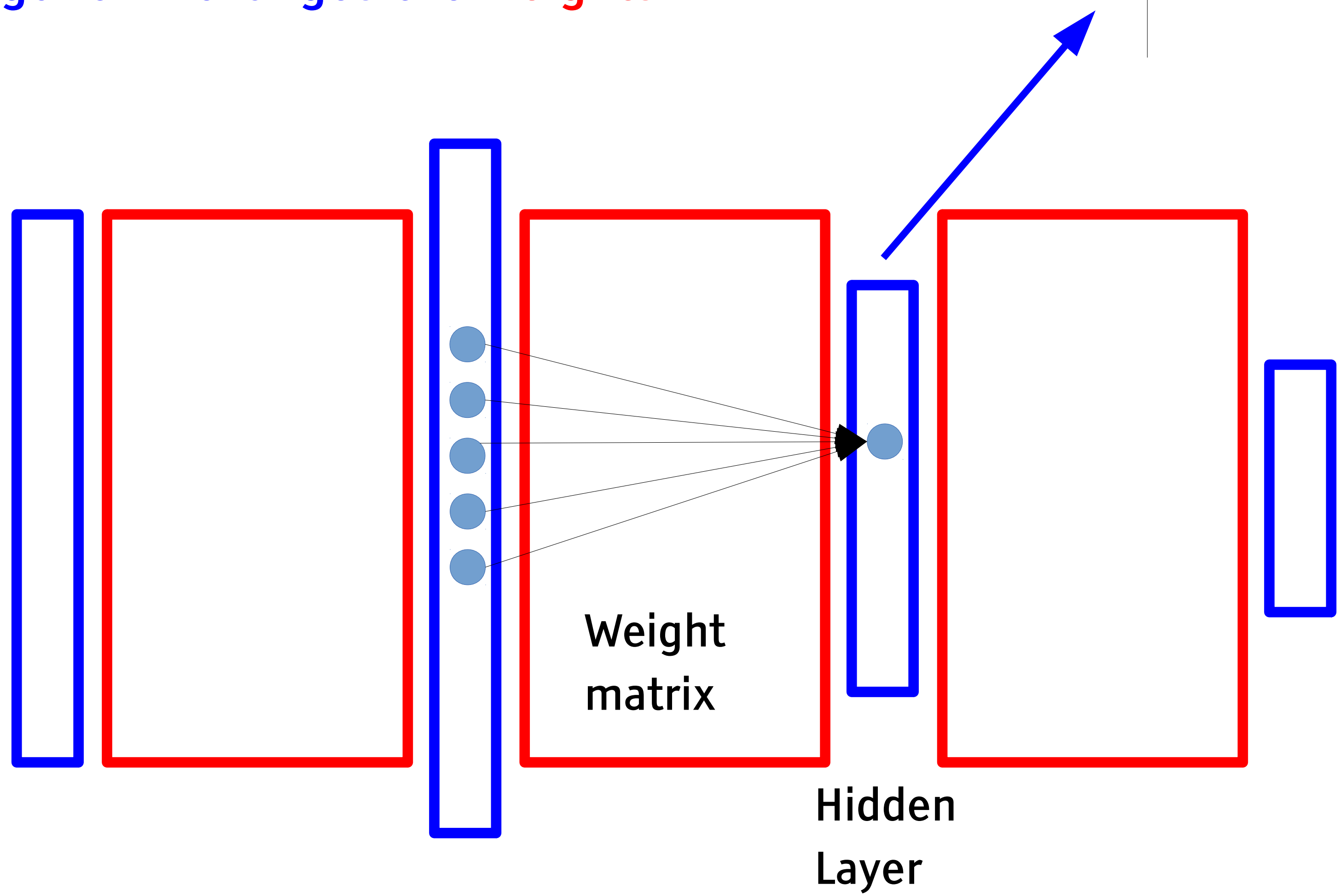
- Multiple Layers of simple units
- Each units computes a weighted sum of its inputs
- Weighted sum is passed through a non-linear function
- The learning algorithm changes the weights

$$ReLU(x) = max(x, 0)$$

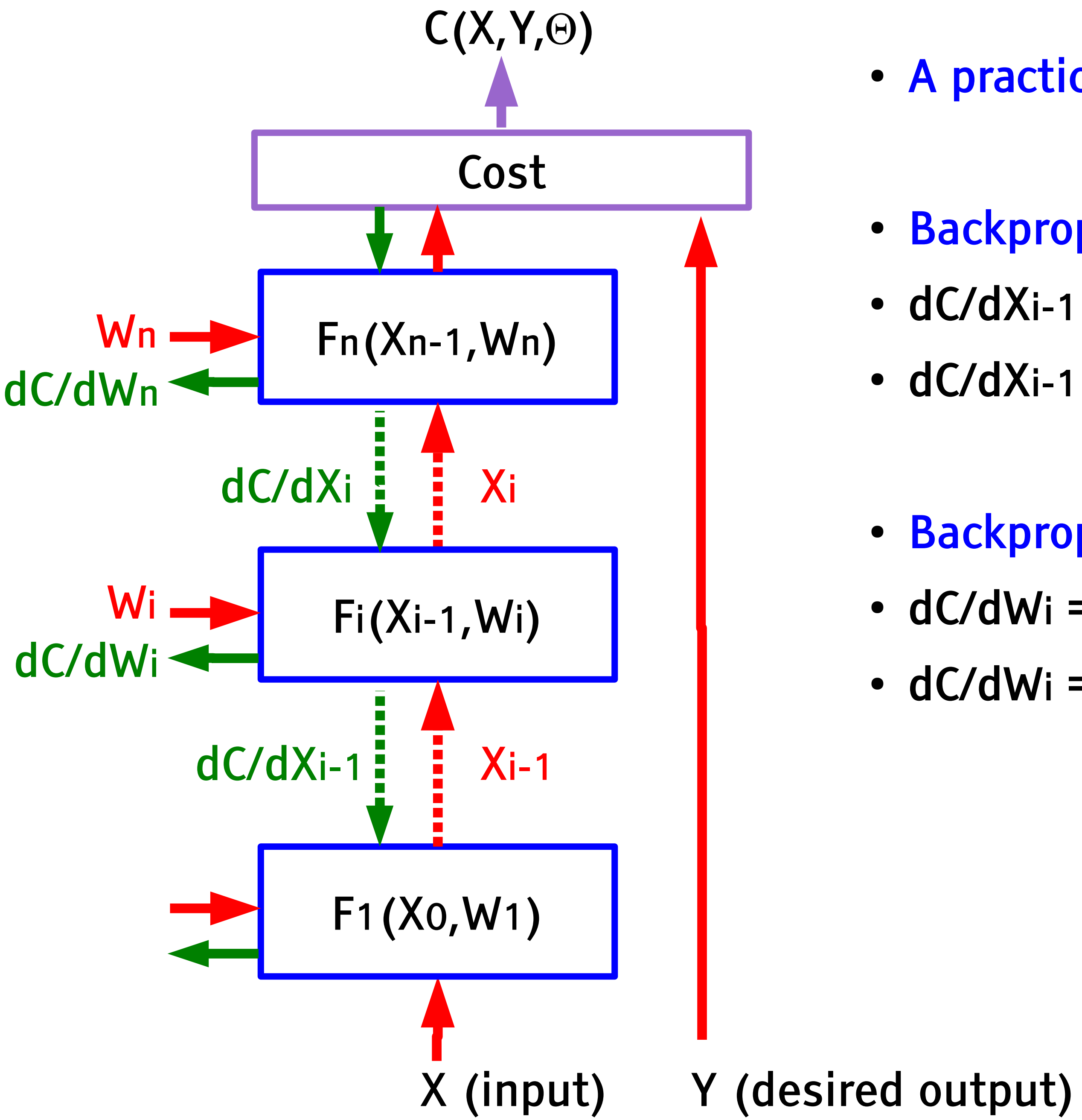


Ceci est une voiture





Computing Gradients by Back-Propagation



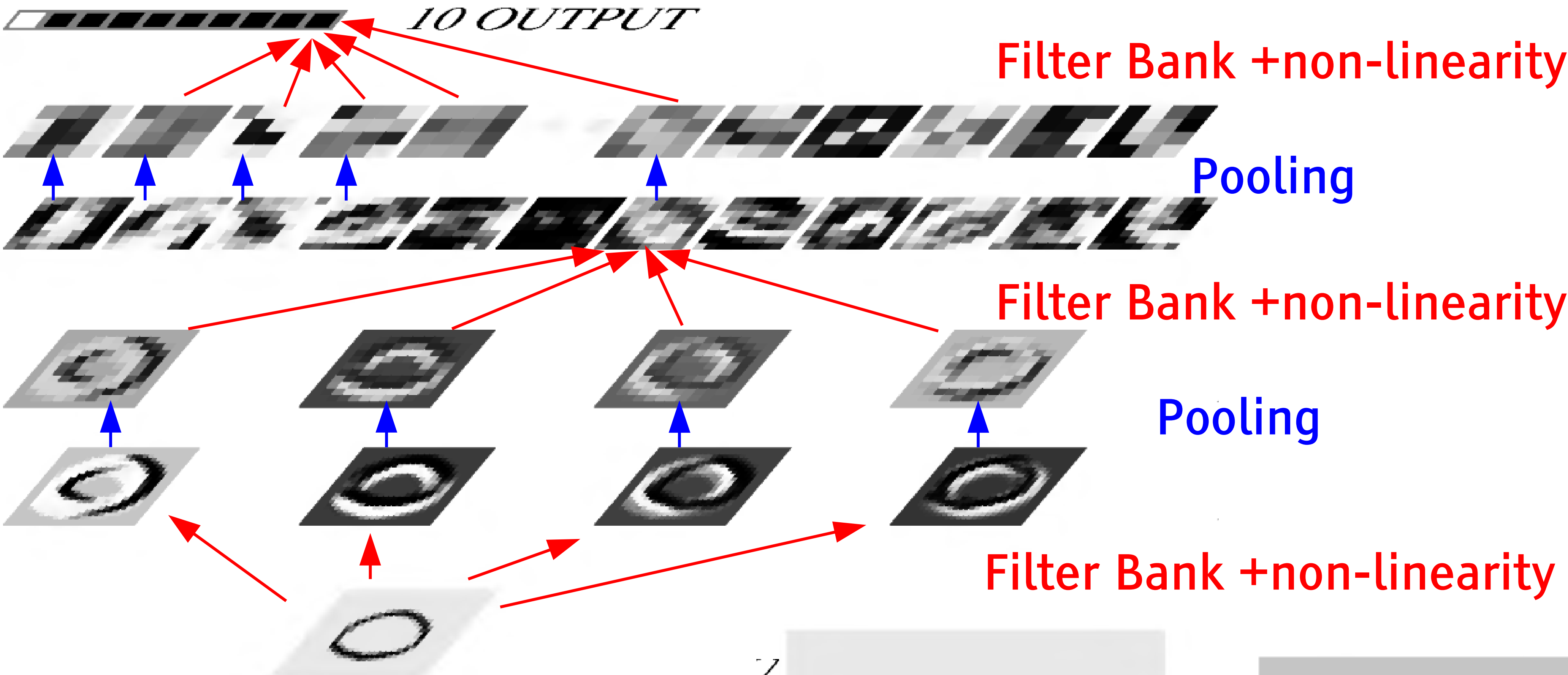
- **A practical Application of Chain Rule**

- **Backprop for the state gradients:**
- $dC/dX_{i-1} = dC/dX_i \,.\, dX_i/dX_{i-1}$
- $dC/dX_{i-1} = dC/dX_i \,.\, dF_i(X_{i-1},W_i)/dX_{i-1}$

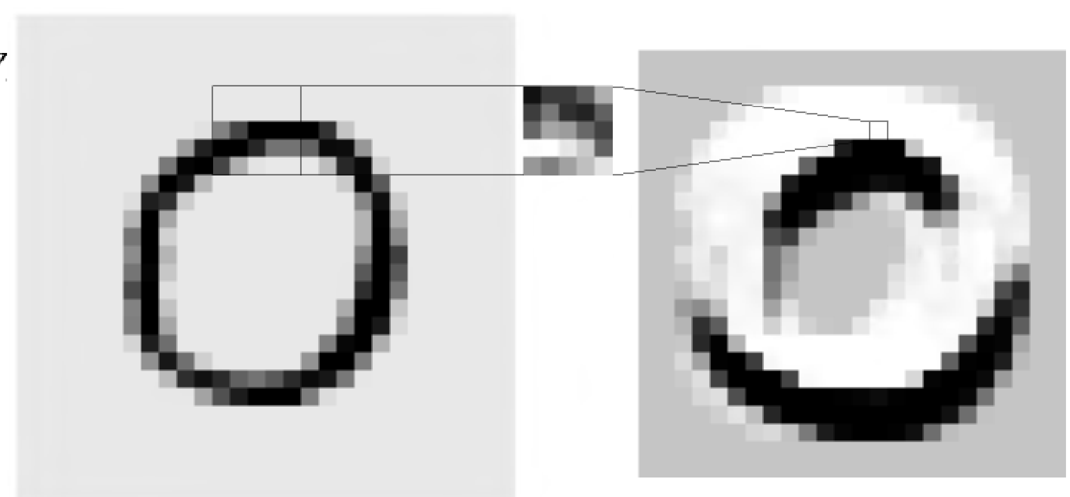
- **Backprop for the weight gradients:**
- $dC/dW_i = dC/dX_i \,.\, dX_i/dW_i$
- $dC/dW_i = dC/dX_i \,.\, dF_i(X_{i-1},W_i)/dW_i$

Convolutional Network Architecture [LeCun et al. NIPS 1989]

Y LeCun



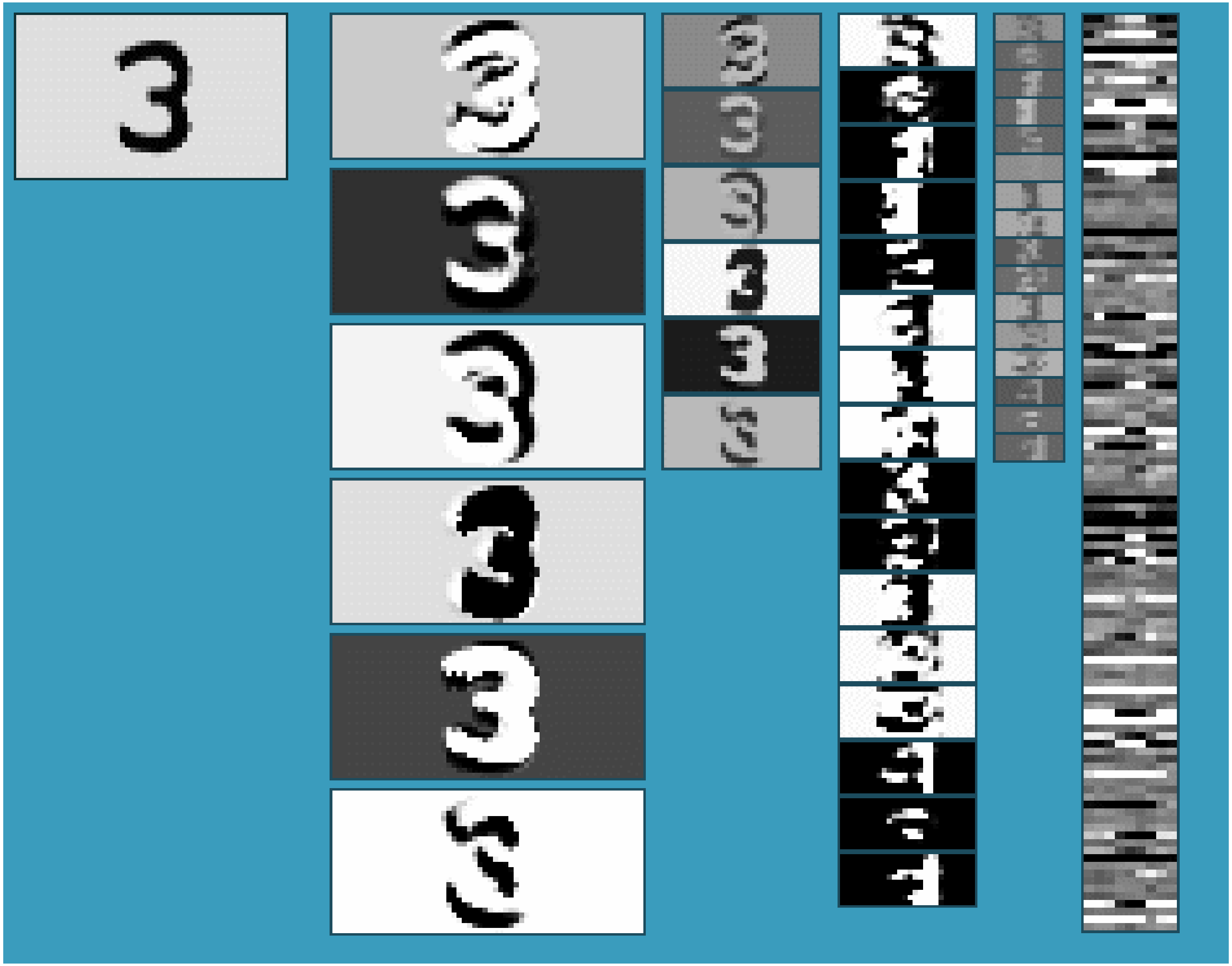
- Inspired by [Hubel & Wiesel 1962] & [Fukushima 1982] (Neocognitron):
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



Convolutional Network (vintage 1990)

Y LeCun

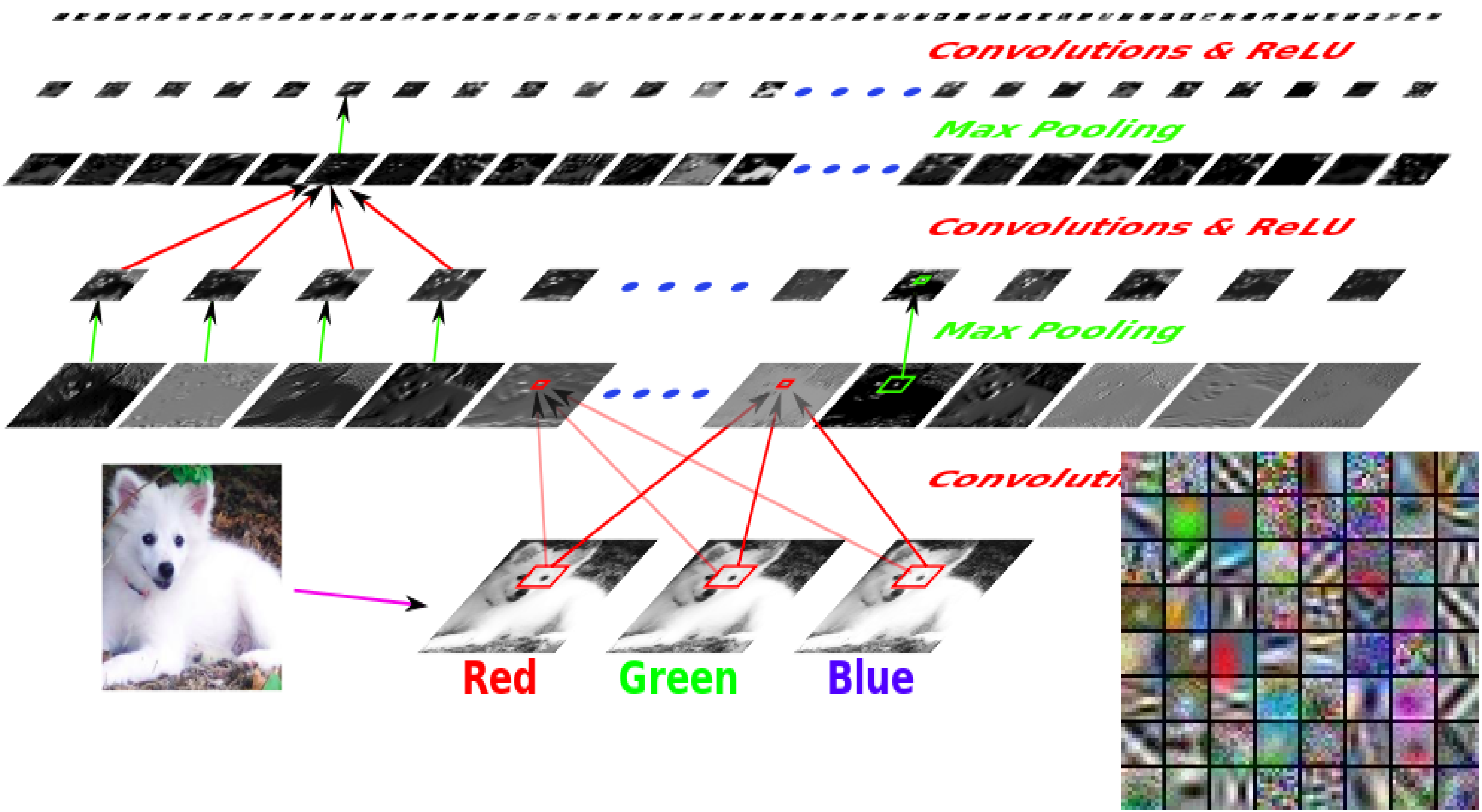
- Filters-tanh → pooling → filters-tanh → pooling → filters-tanh



Deep Convolutional Nets for Object Recognition

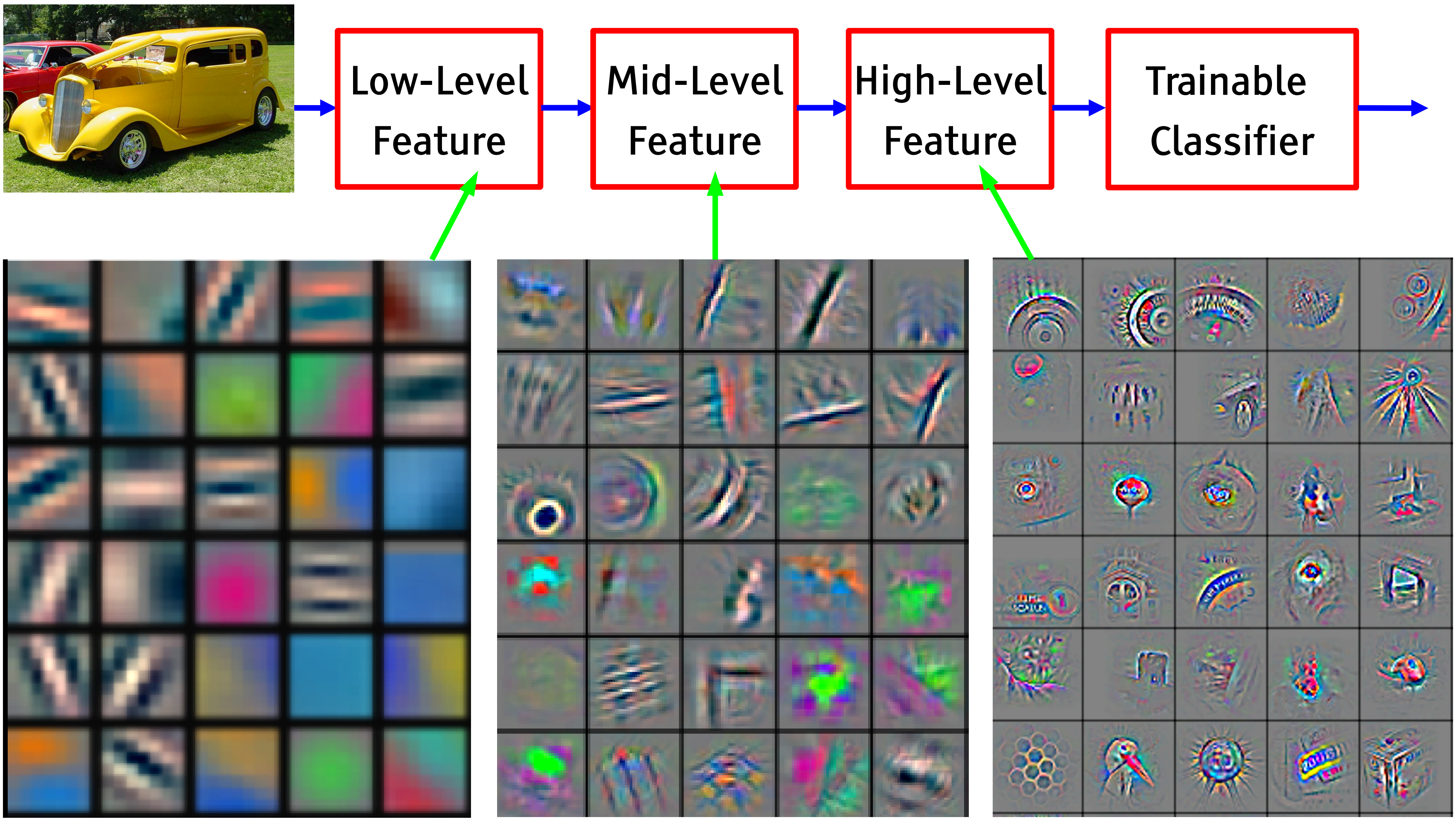
- **1 to 10 billion connections, 10 million to 1 billion parameters, 8 to 20 layers.**

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic Fox (1.0); Eskimo Dog (0.6); White Wolf (0.4); Siberian Husky (0.4)



Deep Learning = Learning Hierarchical Representations

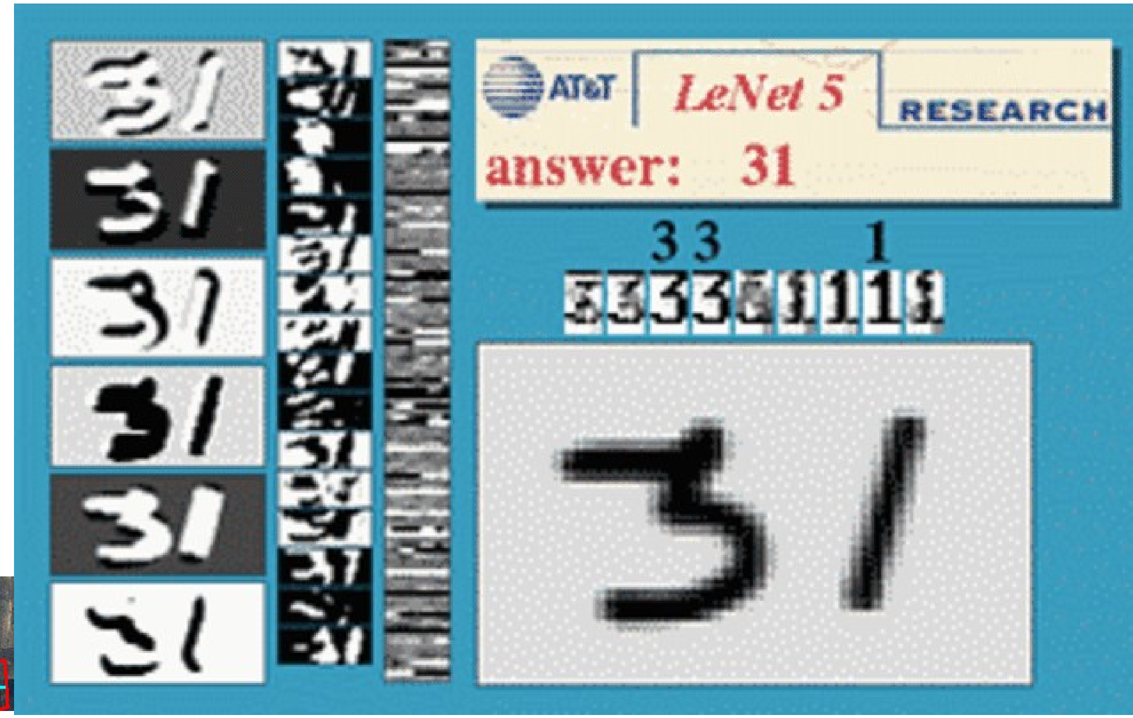
- It's deep if it has more than one stage of non-linear feature transformation

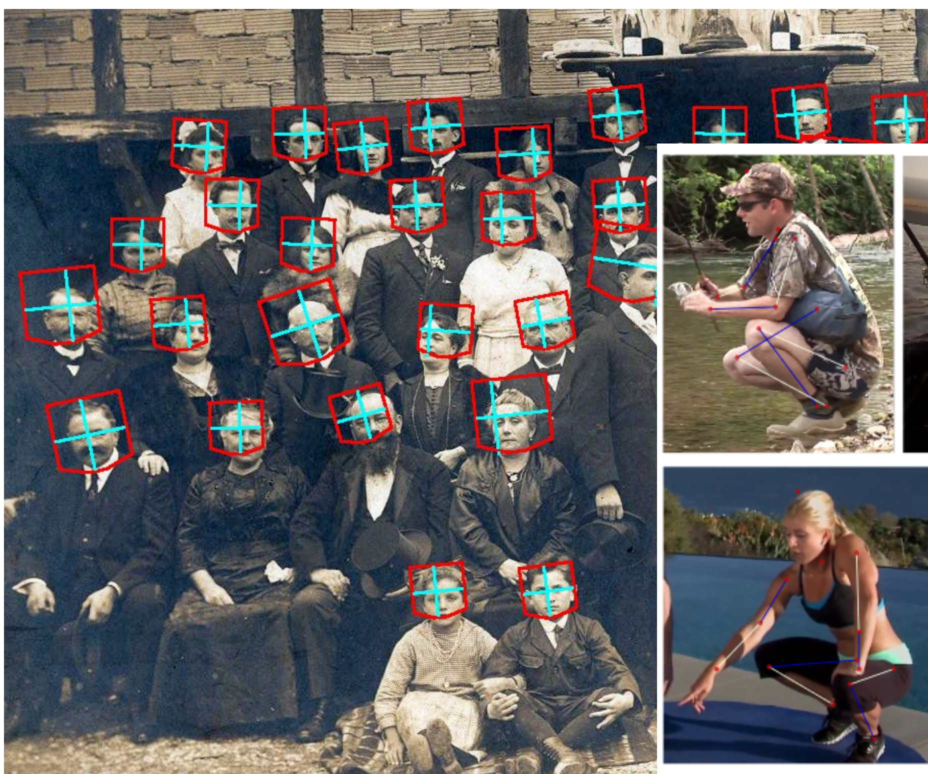


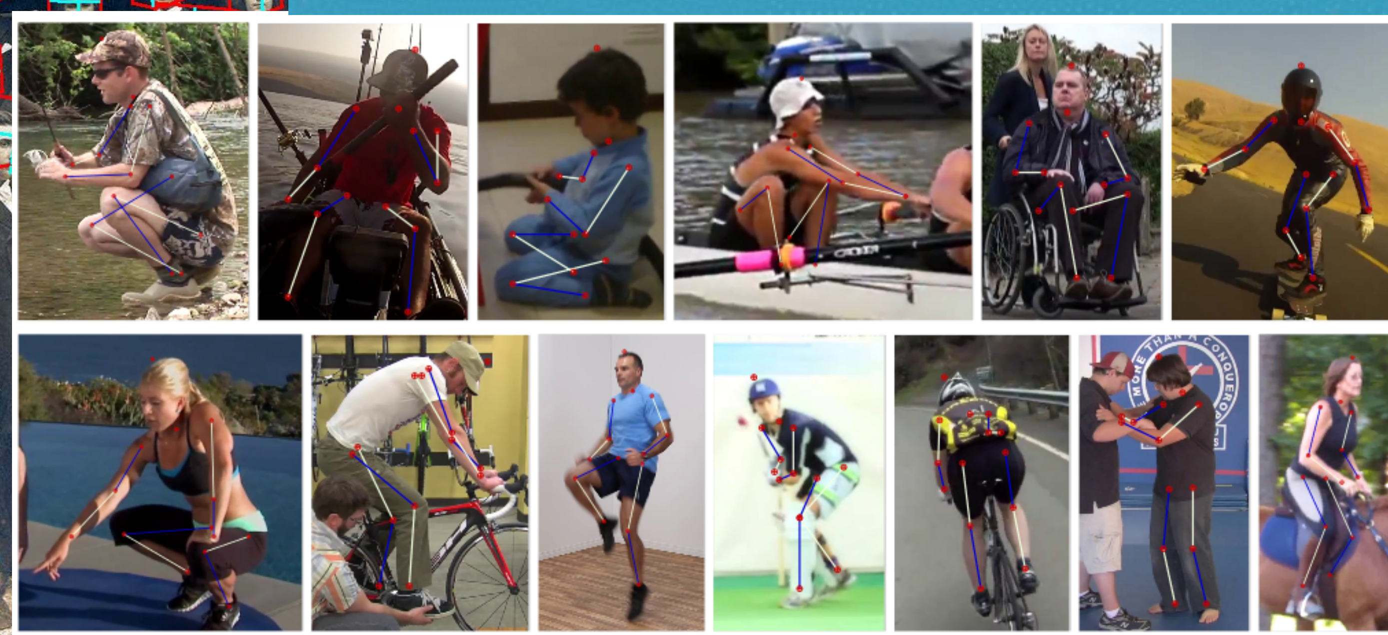
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Supervised Convolutional Nets

- **We can do a lot with Supervised ConvNets**
  - Detect and localize objects
  - Recognize multiple objects
  - Estimate the pose of articulatred objects (human bodies)
  - ….







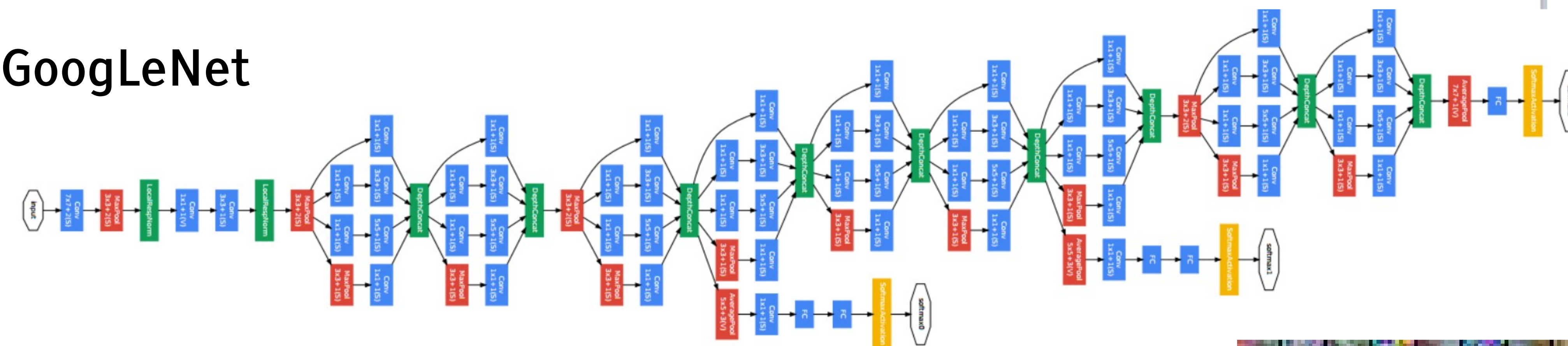
Very Deep ConvNet Architectures

Y LeCun

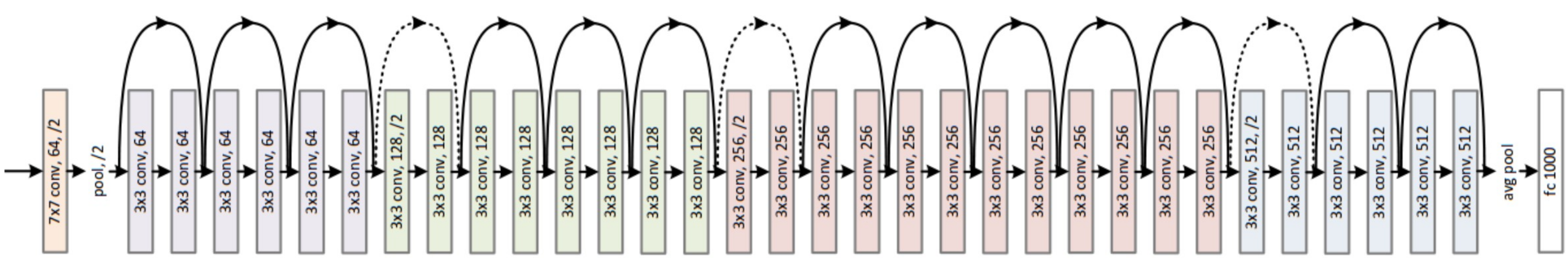
- Small kernels, not much subsampling (fractional subsampling).

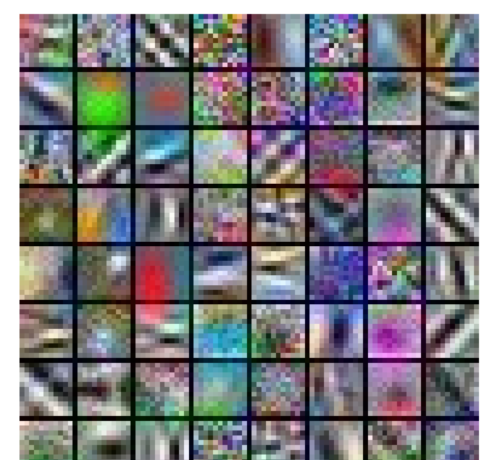
VGG

GoogLeNet



ResNet

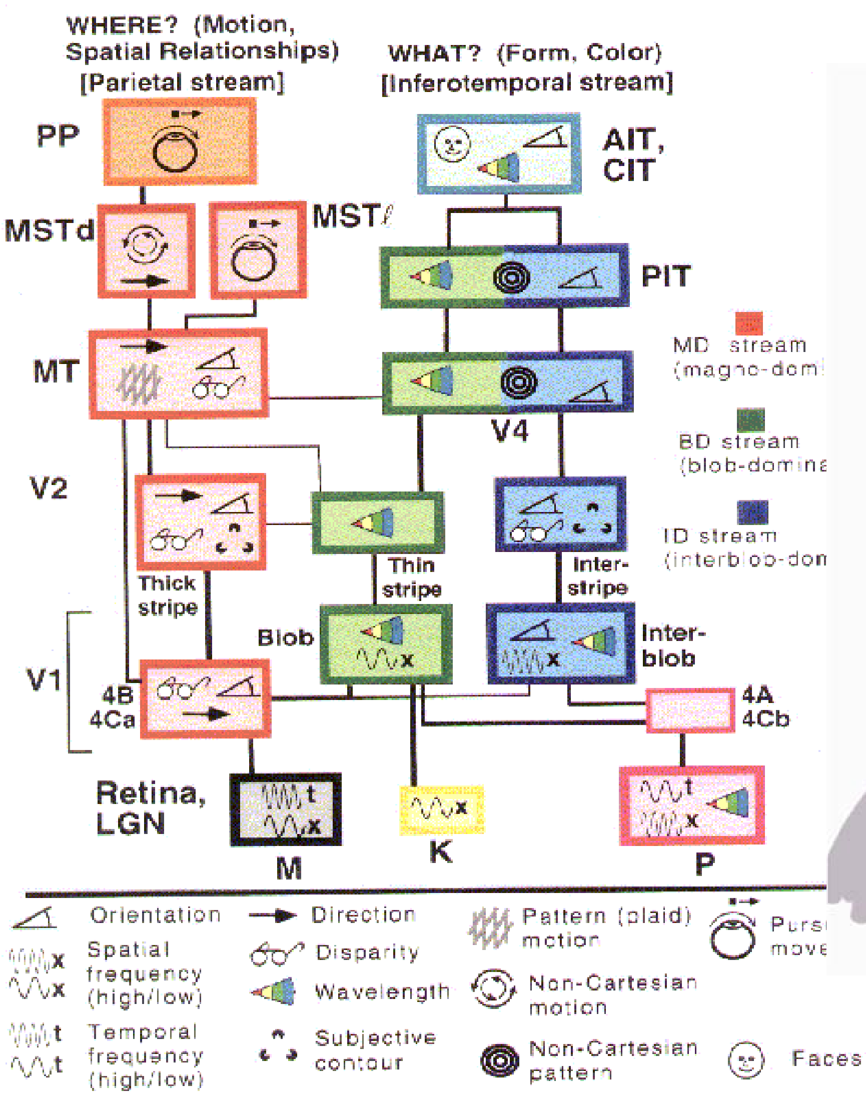




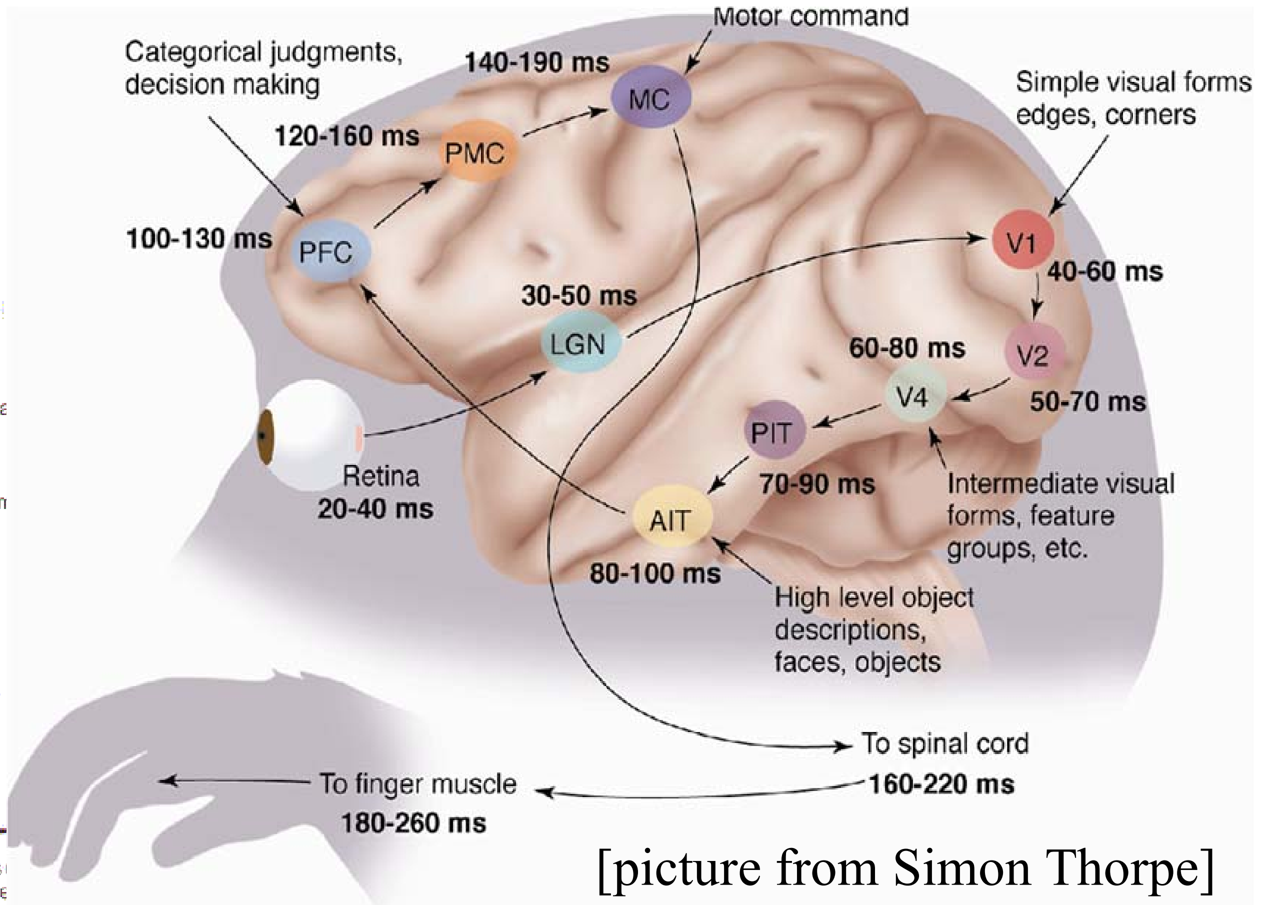
Hierarchical Structure in the Visual Cortex

Y LeCun

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....



[Gallant & Van Essen]

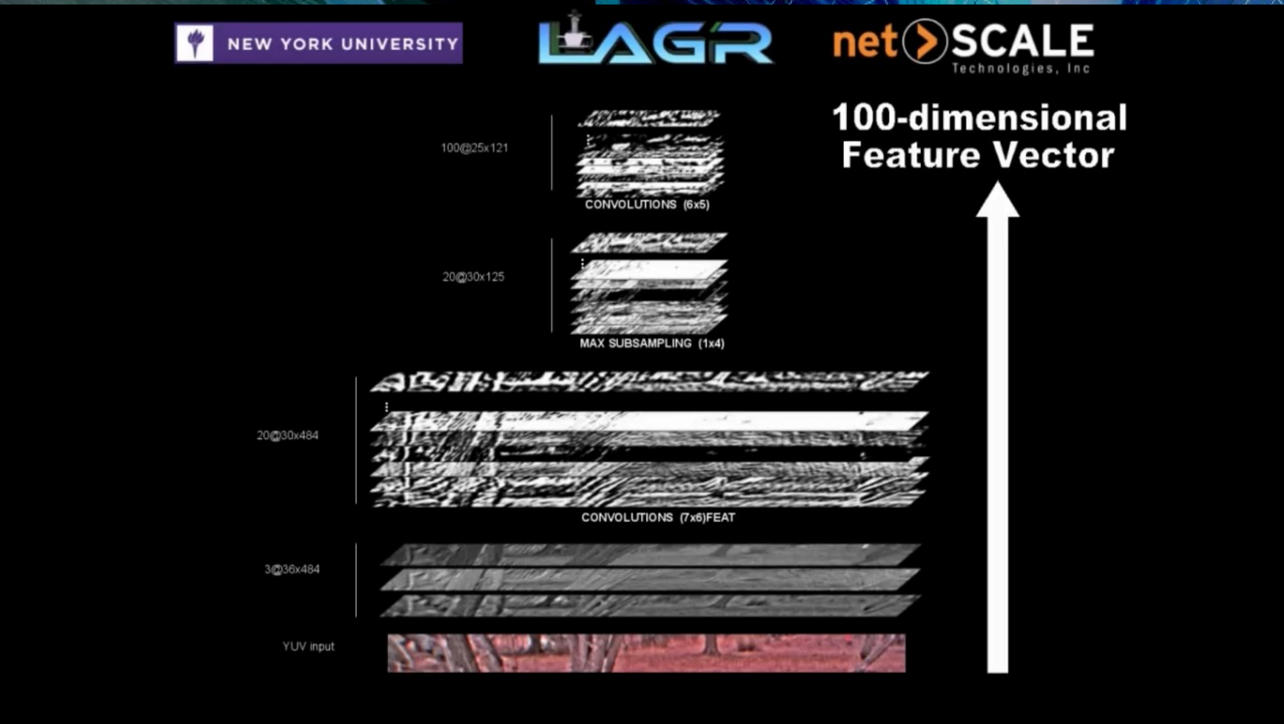


[picture from Simon Thorpe]

ConvNet for Driving

(DARPA LAGR program 2005-2008)

[Hadsell et al., J. of Field Robotics 2009]



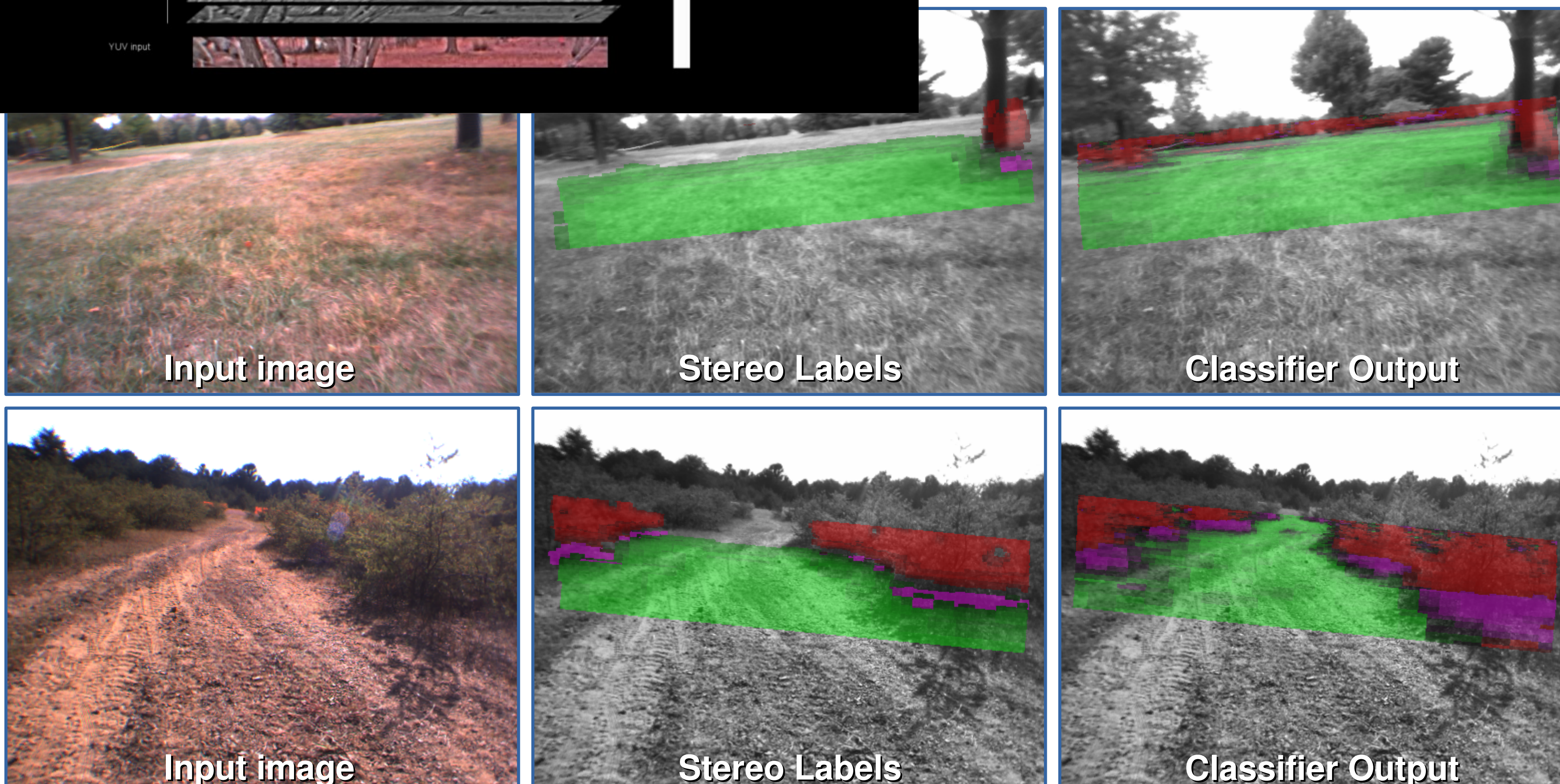


Image captioning, Semantic Segmentation with ConvNets

[Farabet et al. ICML 2011]

[Farabet et al. PAMI 2013]

A man riding skis on a snow covered ski slope.

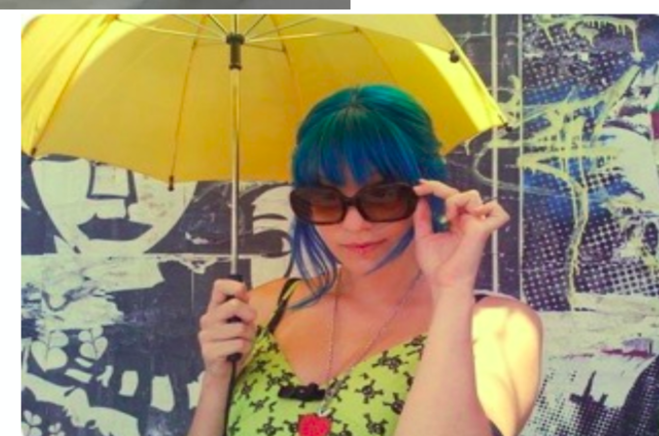
NP: a man, skis, the snow, a person, a woman, a snow covered slope, a slope, a snowboard, a skier, man.
VP: wearing, riding, holding, standing on, skiing down.
PP: on, in, of, with, down.

A man wearing skis on the snow.

A man is doing skateboard tricks on a ramp.

NP: a skateboard, a man, a trick, his skateboard, the air, a skateboarder, a ramp, a skate board, a person, a woman.
VP: doing, riding, is doing, performing, flying through.
PP: on, of, in, at, with.

A man riding a skateboard on a ramp.



The girl with blue hair stands under the umbrella.

NP: a woman, an umbrella, a man, a person, a girl, umbrellas, that, a little girl, a cell phone.
VP: holding, wearing, is holding, holds, carrying.
PP: with, on, of, in, under.

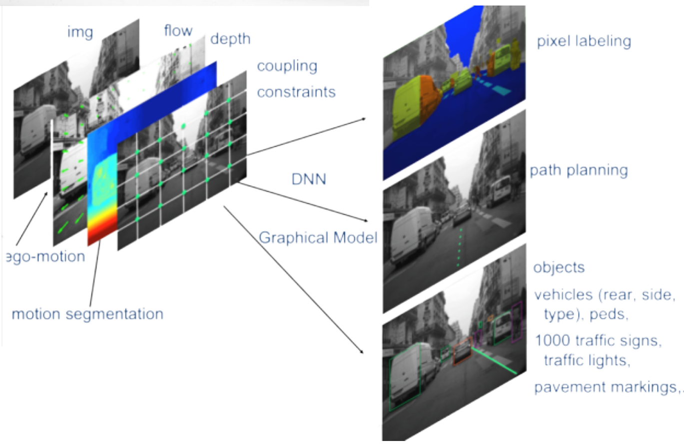
A woman is holding an umbrella.

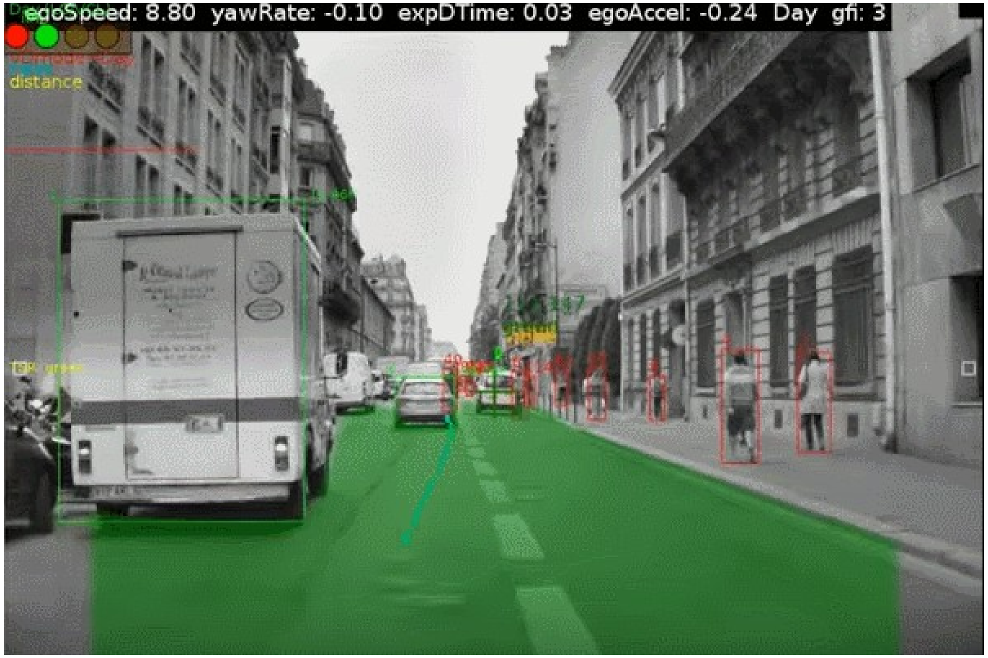
[Lebret, Pinheiro, Collobert 2015] [Kulkarni 11] [Mitchell 12] [Vinyals 14] [Mao 14] [Karpathy 14] [Donahue 14]...

Driving Cars with Convolutional Nets

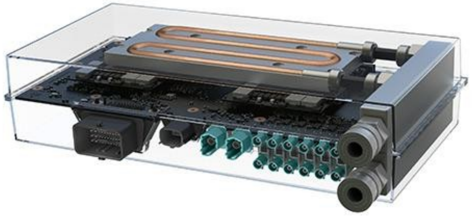
Y LeCun

- MobilEye





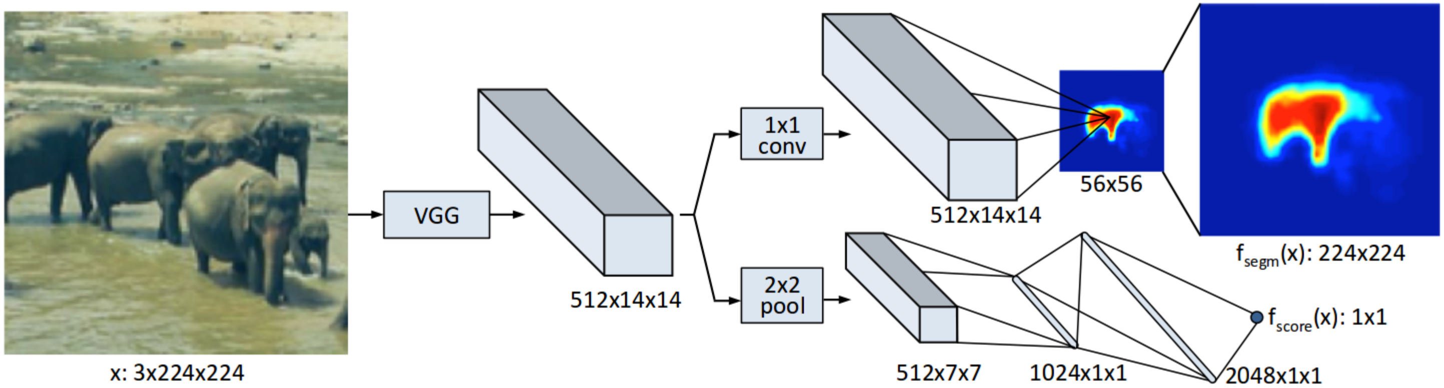
- NVIDIA



DeepMask: ConvNet Locates and Recognizes Objects

Y LeCun

- [Pinheiro, Collobert, Dollar ICCV 2015]
  - ConvNet produces object masks and categories



DeepMask++ Proposals

- https://arxiv.org/abs/1604.02135
- Zagoruyko, Lerer, Lin, Pinheiro, Gross, Chintala, Dollár

Image Recognition

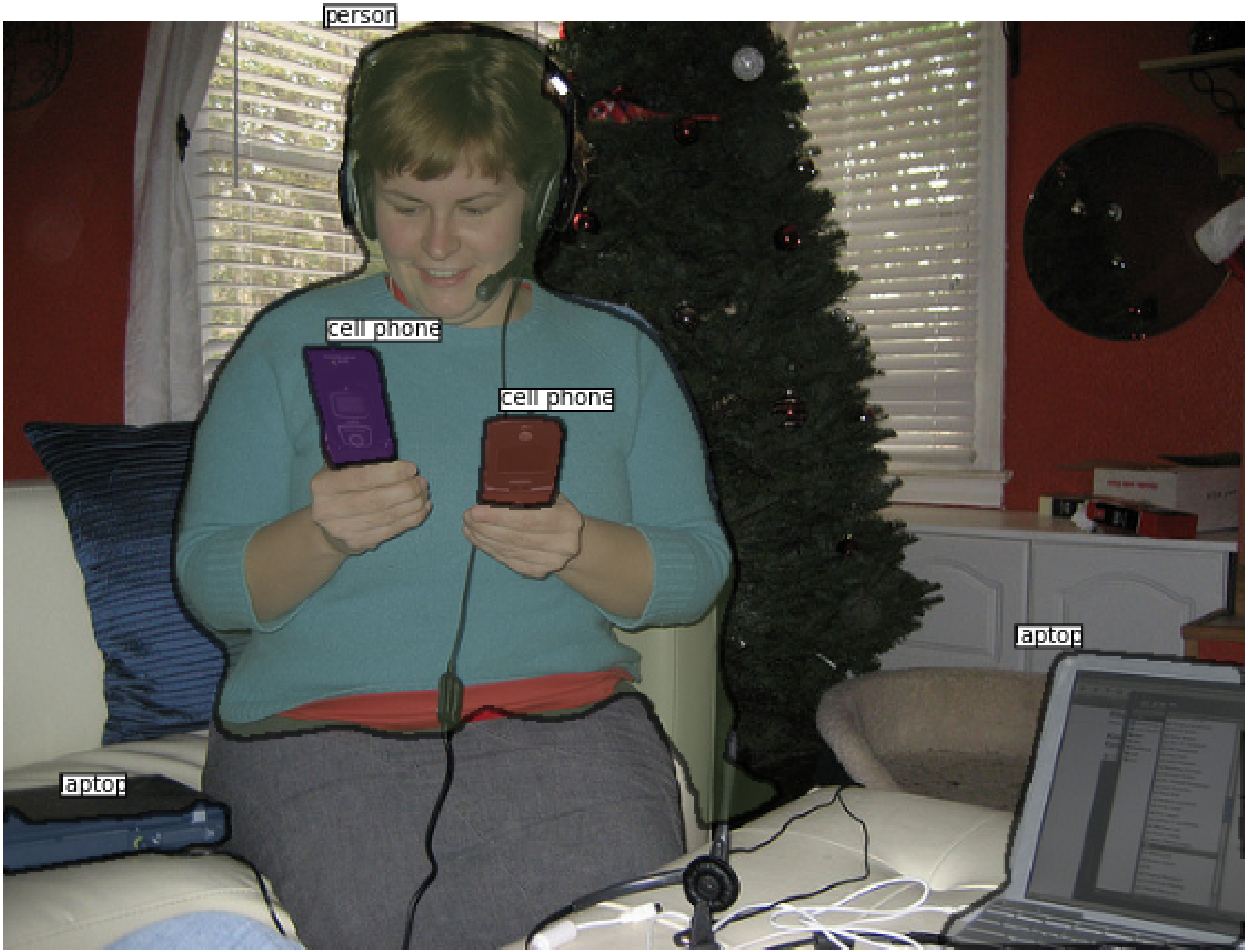


Image Recognition

Y LeCun

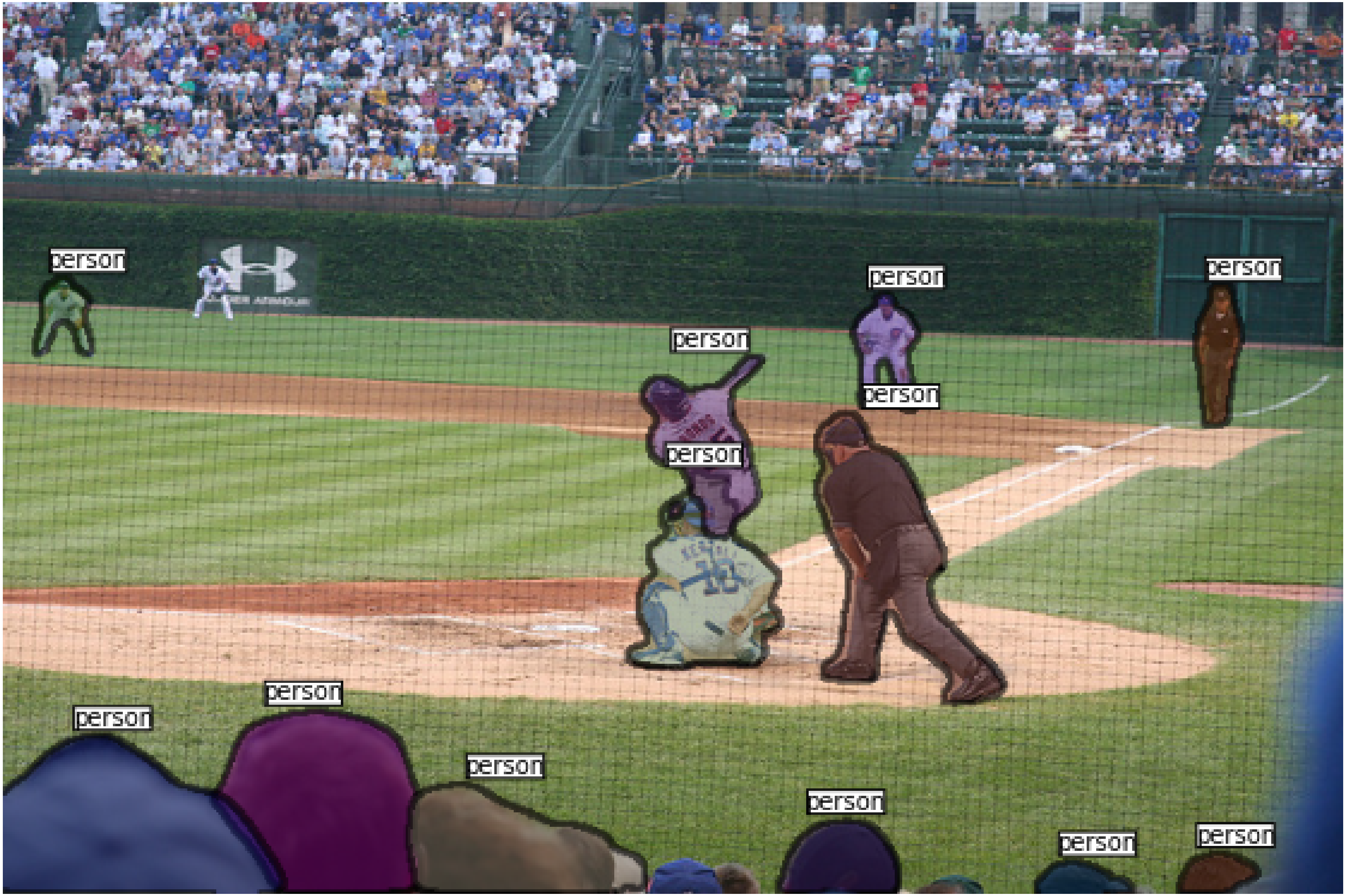


Image Recognition

Image Recognition

Y LeCun

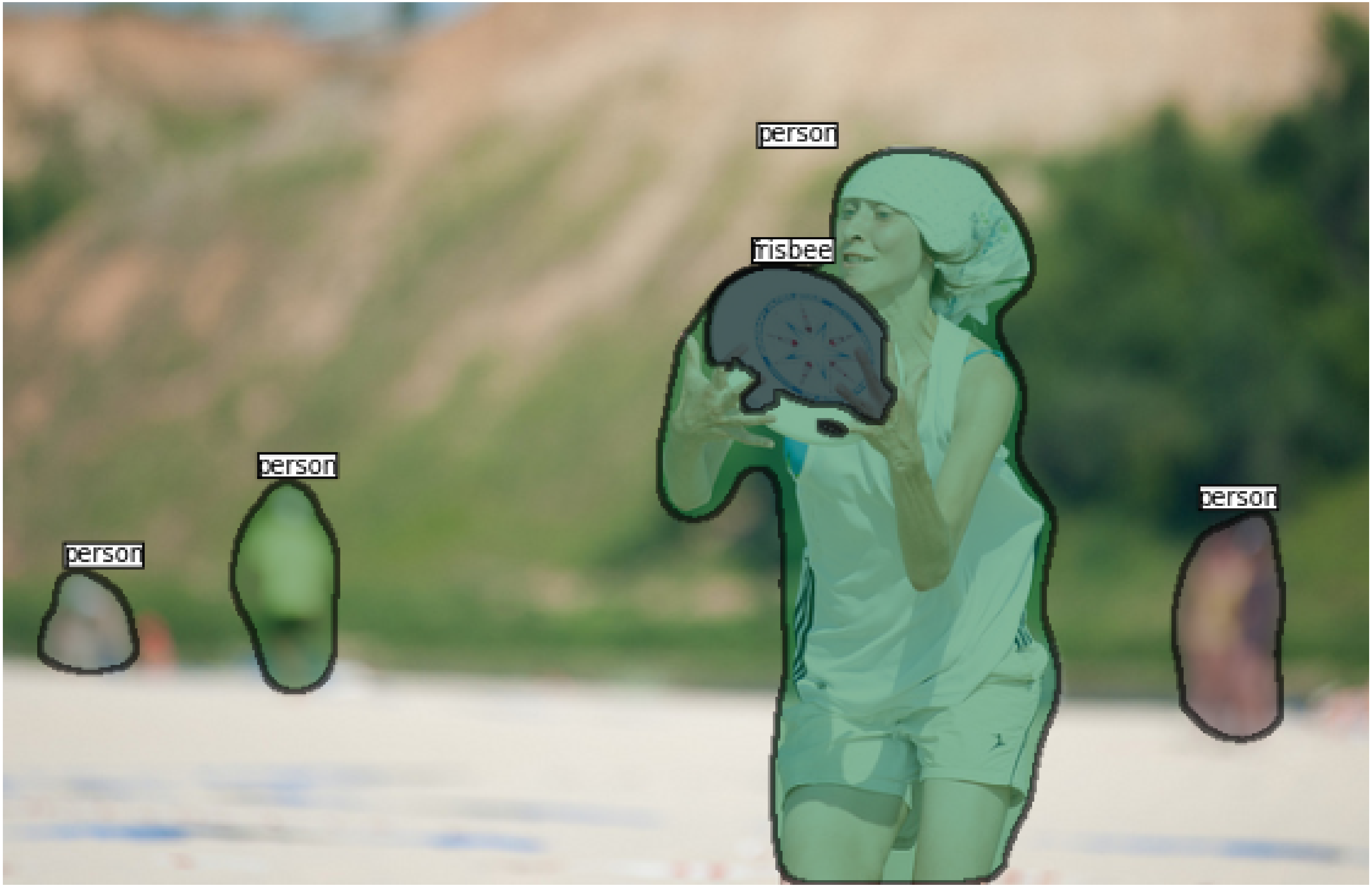


Image Recognition

Y LeCun

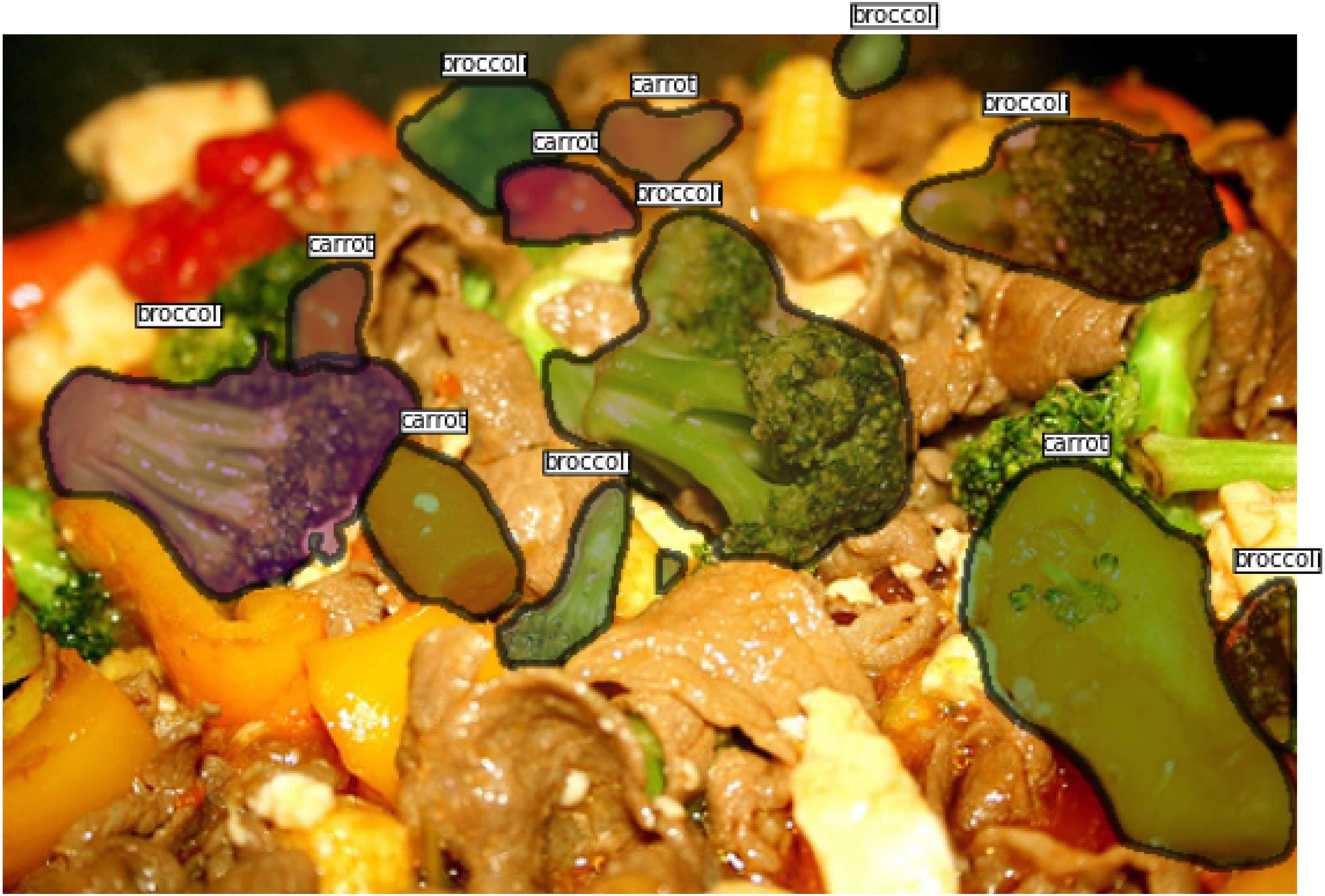


Image Recognition

Results

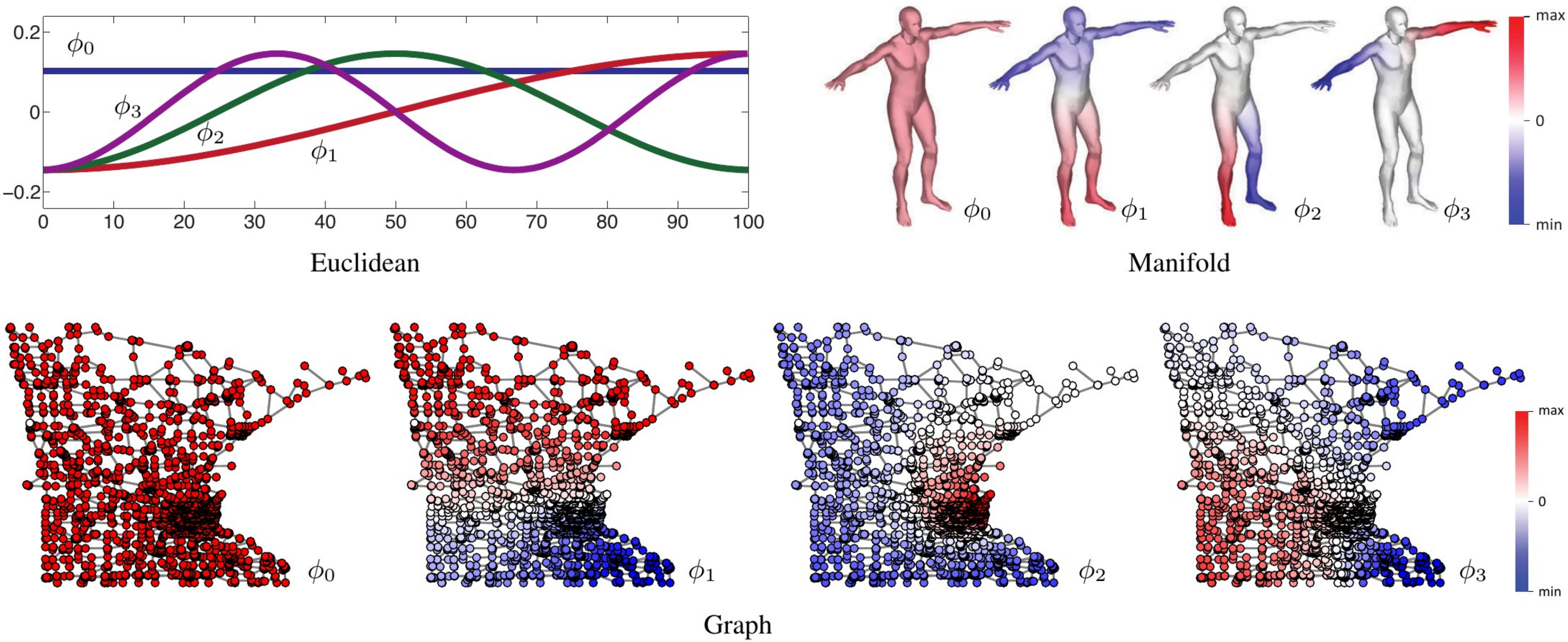

sheep
sheep
sheep
sheep
sheep
sheep
sheep
sheep
sheep
sheep
sheep
sheep

Spectral Networks

Convolutional Nets on Graphs

Spectral Networks: Convolutional Nets on Irregular Graphs

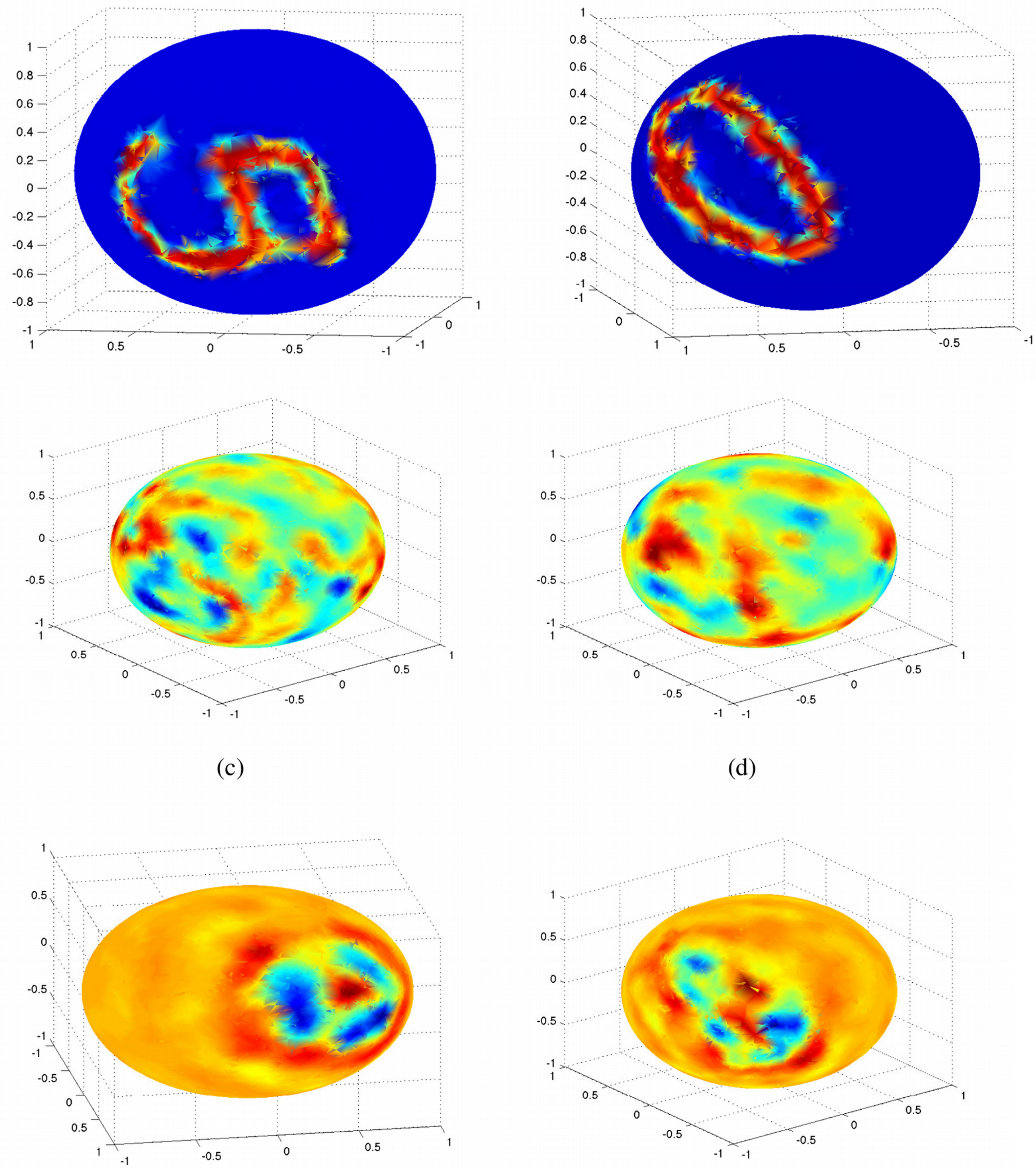
- Convolutions are diagonal operators in Fourier space
- The Fourier space is the eigenspace of the Laplacian
- We can compute graph Laplacians
- Review paper: [Bronstein et al. 2016, ArXiv:1611.08097]



Spectral Networks: Convolutional Nets on Irregular Graphs

Y LeCun

- **Compute graph Laplacian**
- **Compute transformation into its eigenspace**
  - Fourier transform
- **Learn "smooth" pointwise multipliers in Fourier space**
  - Localized kernels
- **Applicable to any function on graphs**
  - Chemistry, text, images with holes, image on non-euclidean surfaces...
  - [Bruna et al. Arxiv:1312.6203]
  - [Henaff et al. Arxiv:1506.05163]



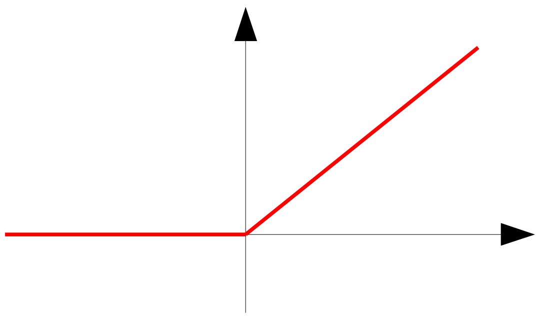
(c) (d)

Why Doesn't Deep Learning Get Trapped In Local Minima?

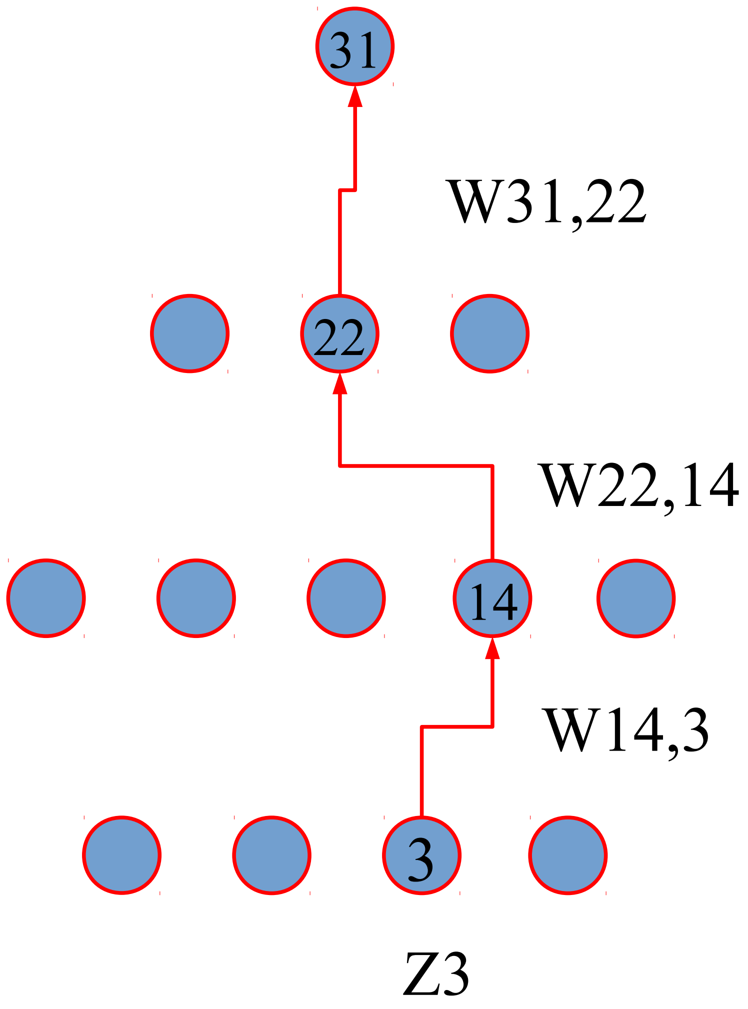
Deep Nets with ReLUs and Max Pooling

- Stack of linear transforms interspersed with Max operators
- Point-wise ReLUs:

$$ReLU(x) = max(x, 0)$$

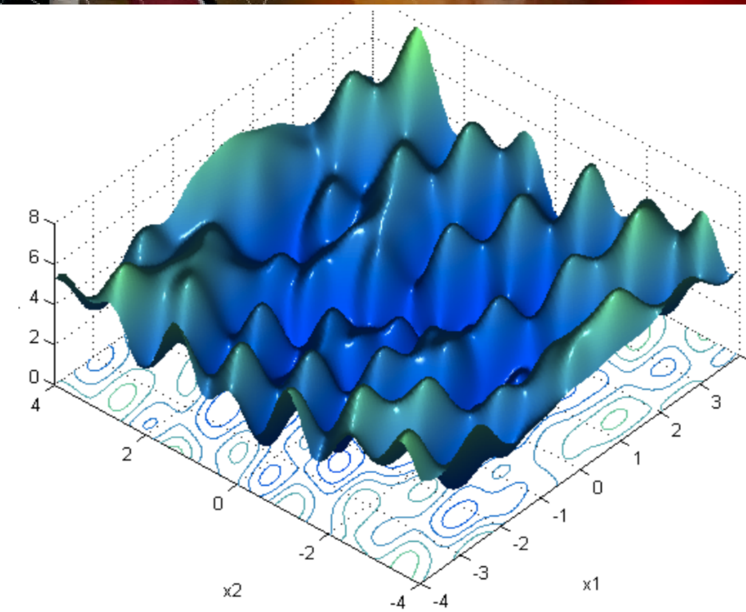


- Max Pooling
  - "switches" from one layer to the next

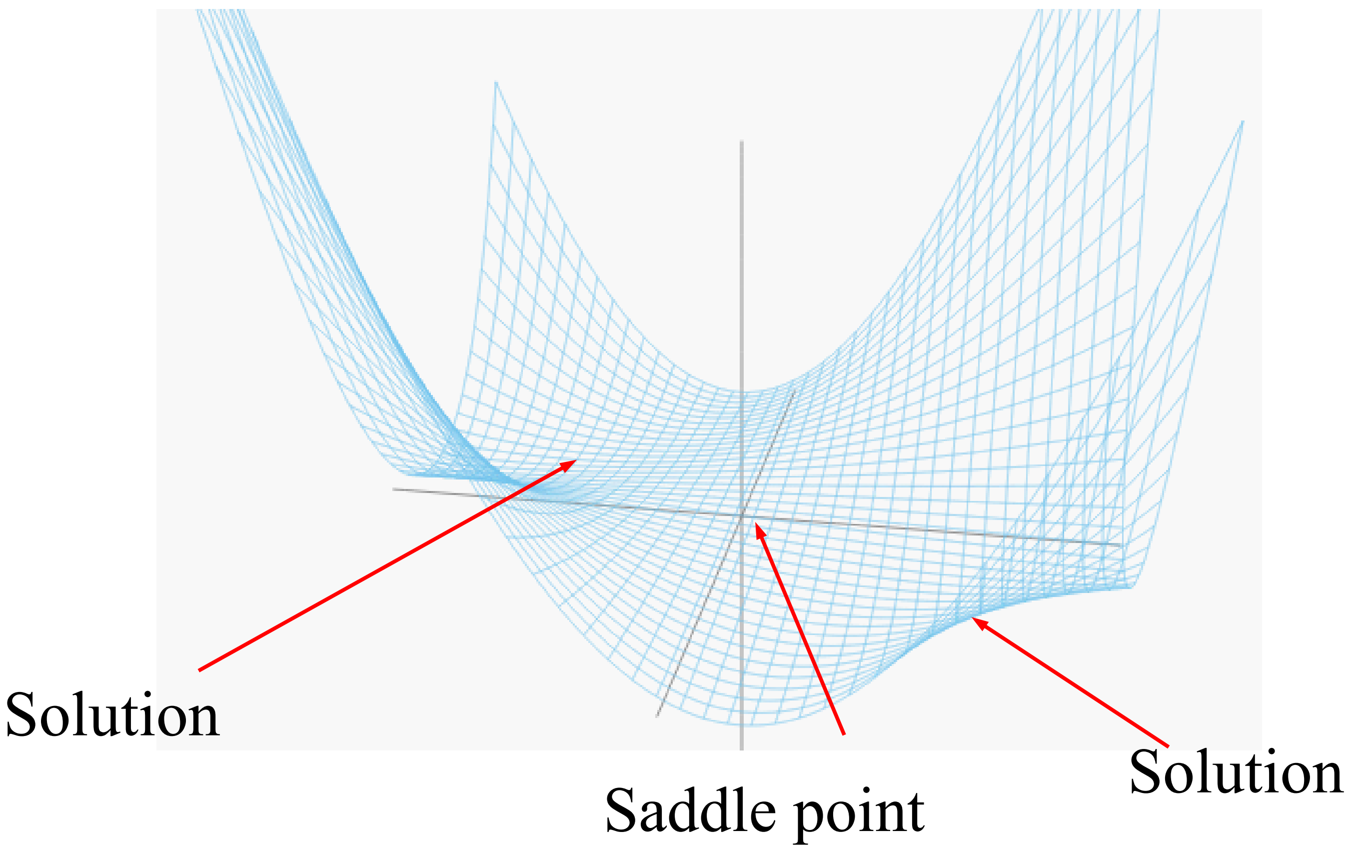


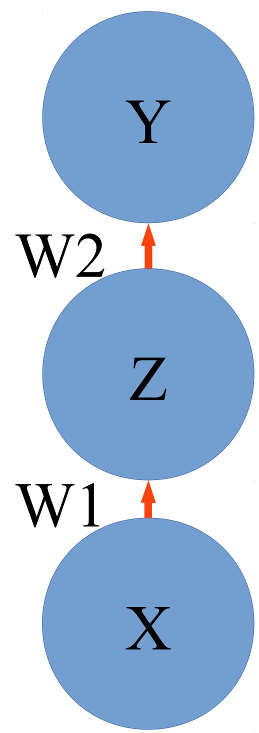
The Objective Function of Multi-layer Nets is Non Convex

Y LeCun



- **1-1-1 network**
  - Y = W1*W2*X
- **Objective: identity function with quadratic loss**
- **One sample: X=1, Y=1 L(W) = (1-W1*W2)^2**



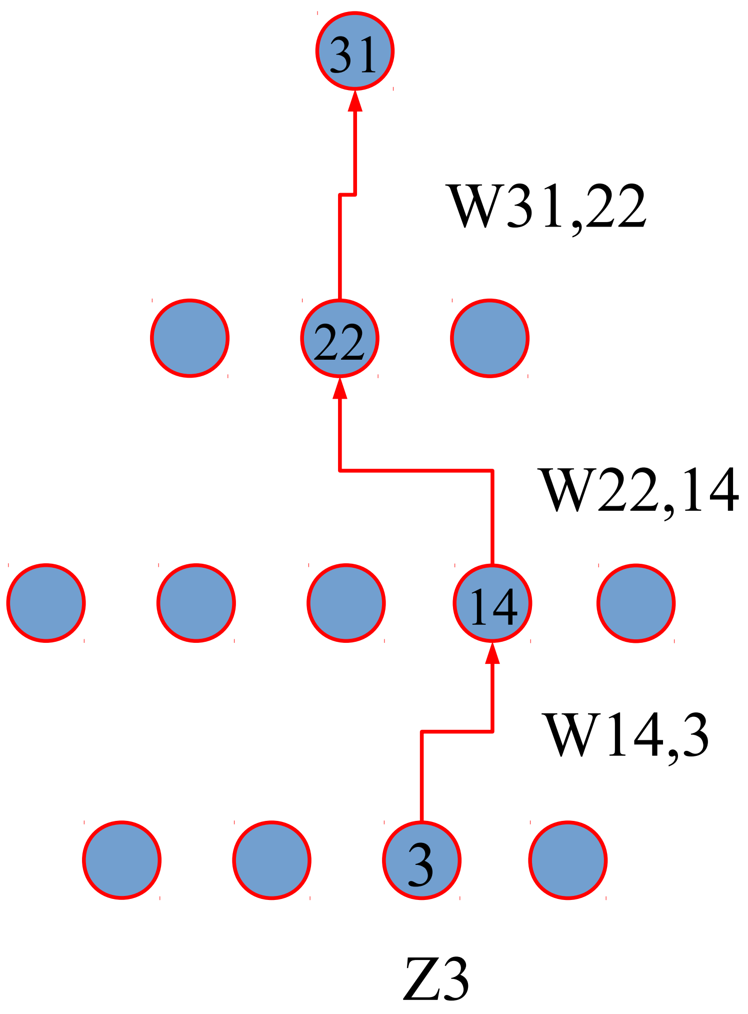


Deep Nets with ReLUs

- **Single output:**

$$\hat{Y} = \sum_{P} \delta_P(W, X) \left(\prod_{(ij) \in P} W_{ij} \right) X_{P_{start}}$$

- **Wij: weight from j to i**
- **P: path in network from input to output**
  - P=(3,(14,3),(22,14),(31,22))
- **di: 1 if ReLU i is linear, 0 if saturated.**
- **Xpstart: input unit for path P.**



$$\hat{Y} = \sum_{P} \delta_P(W, X) \left(\prod_{(ij) \in P} W_{ij} \right) X_{P_{start}}$$

- **Dp(W,X): 1 if path P is "active", 0 if inactive**
- **Input-output function is piece-wise linear**
- **Polynomial in W with random coefficients**

Deep Convolutional Nets (and other deep neural nets)

Y LeCun

- Training sample: (Xi,Yi) k=1 to K
- Objective function (with margin-type loss = ReLU)

$$L(W)=\sum_k ReLU\left(1-Y^k \sum_P \delta_P(W,X^k)\left(\prod_{(ij)\in P} W_{ij}\right) X^k_{P_{start}}\right)$$

$$L(W)=\sum_k \sum_P \left(X^k_{P_{start}} Y^k\right) \delta_P(W,X^k)\left(\prod_{(ij)\in P} W_{ij}\right)$$

$$L(W)=\sum_P \left[\sum_k \left(X^k_{P_{start}} Y^k\right) \delta_P(W,X^k)\right]\left(\prod_{(ij)\in P} W_{ij}\right)$$

$$L(W)=\sum_P C_p(X,Y,W)\left(\prod_{(ij)\in P} W_{ij}\right)$$

- Polynomial in W of degree l (number of adaptive layers)
- Continuous, piece-wise polynomial with “switched” and partially random coefficients
  - Coefficients are switched in an out depending on W

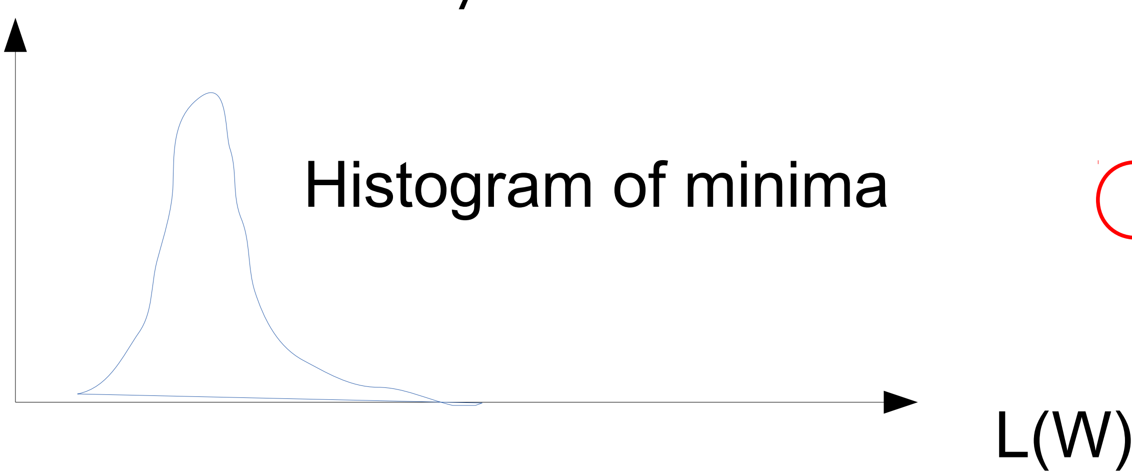
Deep Nets with ReLUs: Objective Function is Piecewise Polynomial

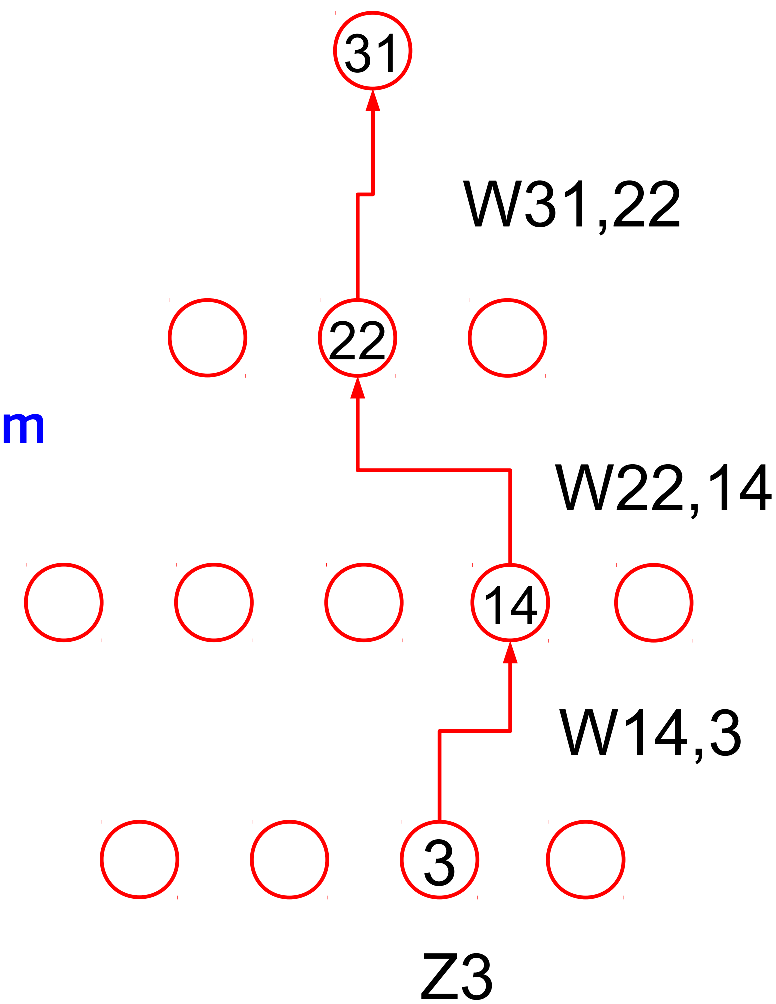
Y LeCun

- If we use a hinge loss, delta now depends on label Yk:

$$L(W)=\sum_{P} C_p(X,Y,W)\Big(\prod_{(ij)\in P} W_{ij}\Big)$$

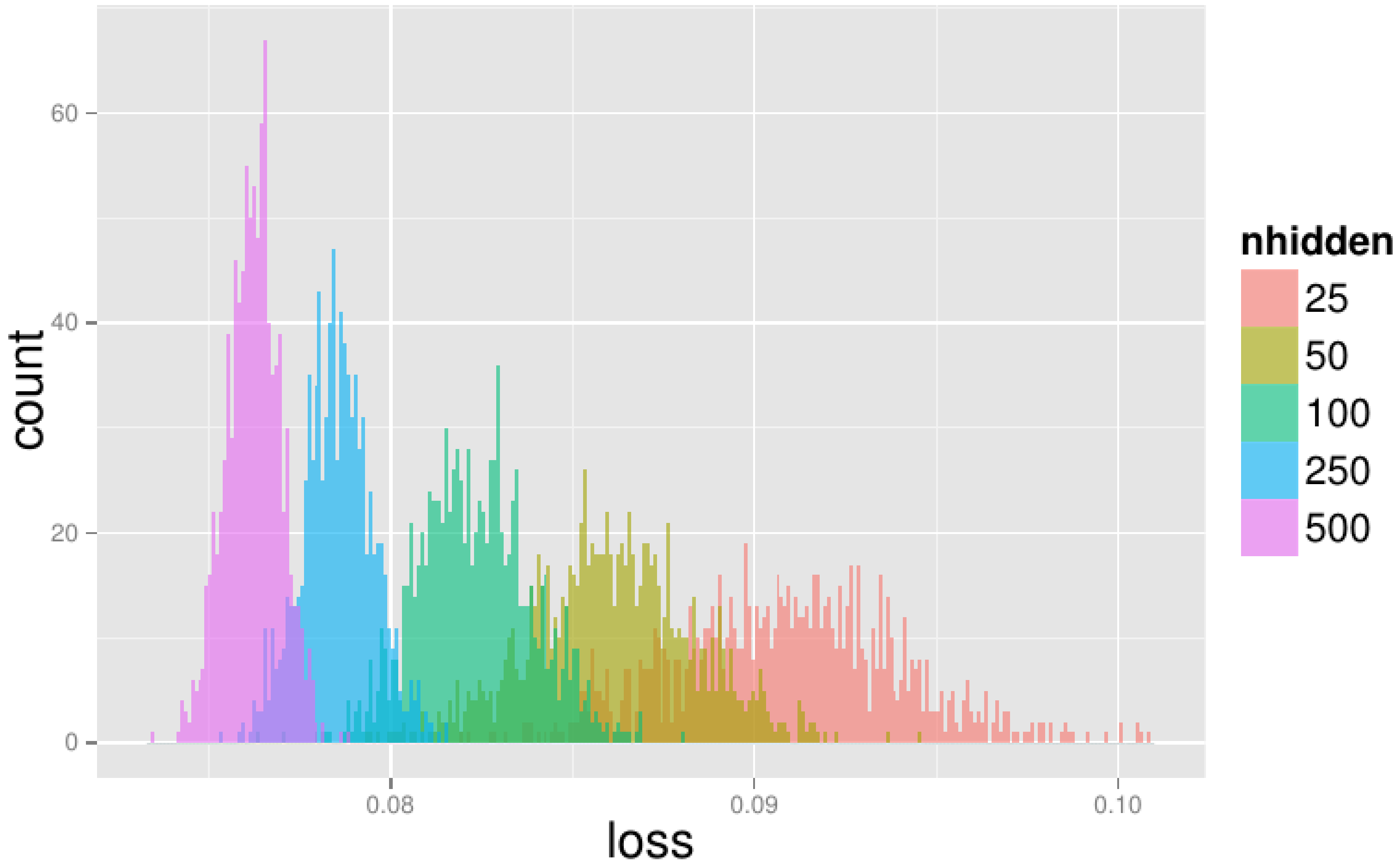
- Piecewise polynomial in W with random coefficients
- A lot is known about the distribution of critical points of polynomials on the sphere with random (Gaussian) coefficients [Ben Arous et al.]
  - High-order spherical spin glasses
  - Random matrix theory





Deep Nets with ReLUs: Objective Function is Piecewise Polynomial

- Train 2-layer nets on scaled-down MNIST (10x10) from multiple initial conditions. Measure loss on test set.

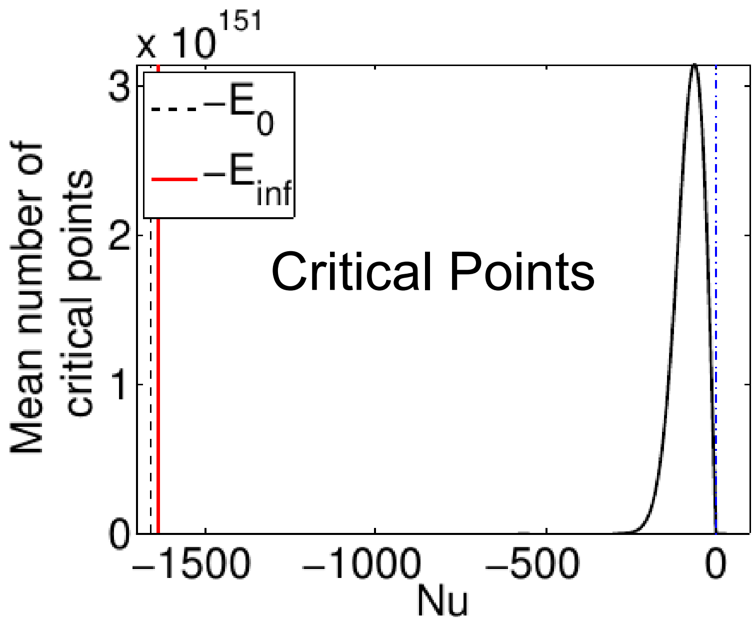


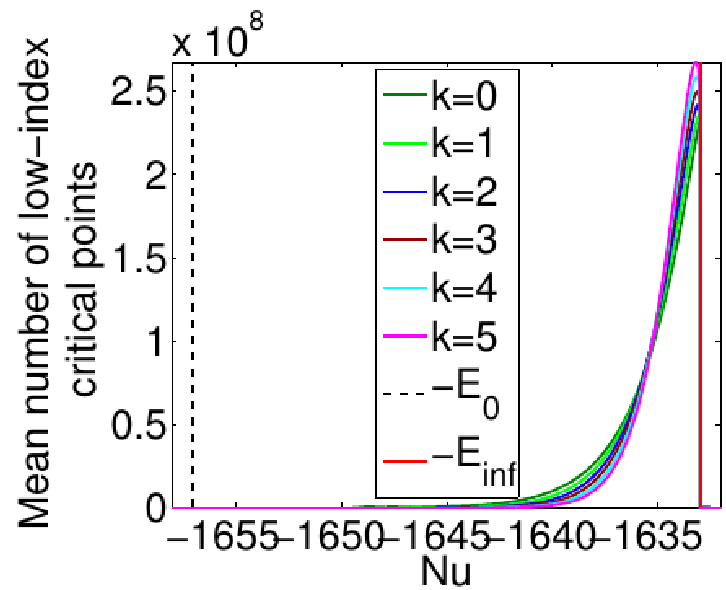
[Choromanska, Henaff, Mathieu, Ben Arous, LeCun 2015]

Spherical Spin Glass theory

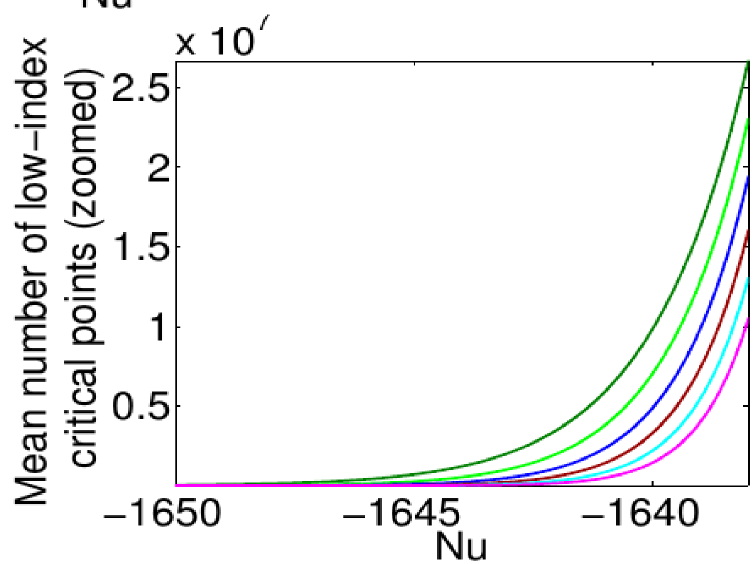
Distribution of critical points (saddle points, minima, maxima)

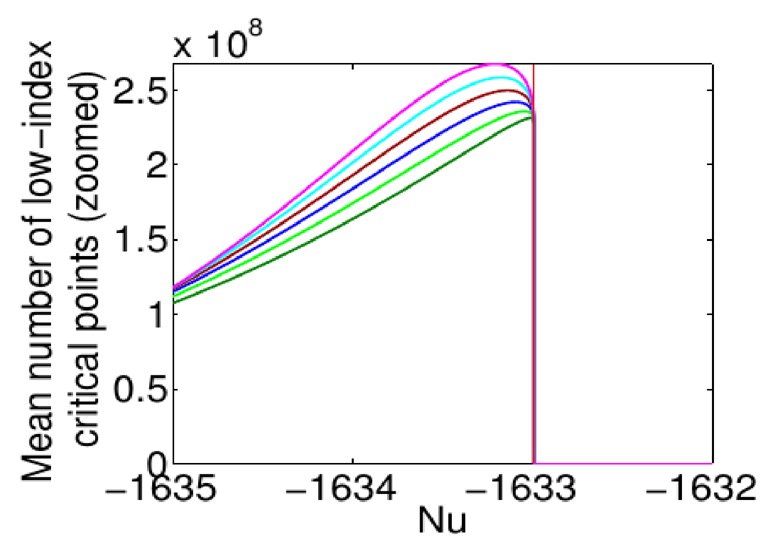
- K=number of negative eigenvalues of Hessian (K=0 → minimum)





Zoomed:





Orthogonal Recurrent Neural Nets

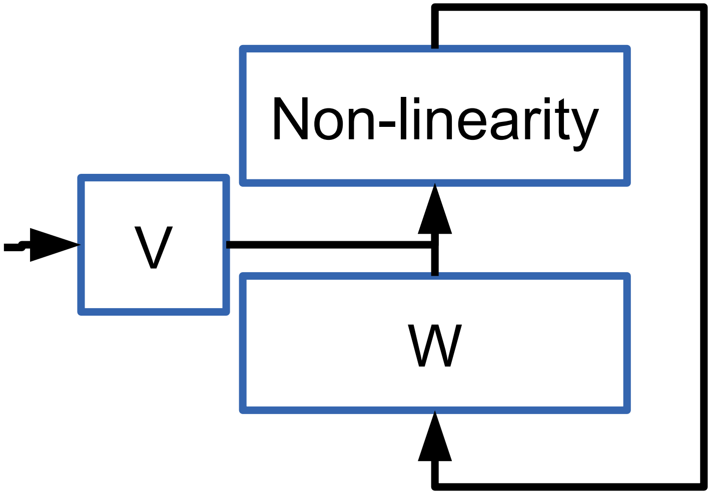
[Henaff, Szlam, LeCun ICML 2016]

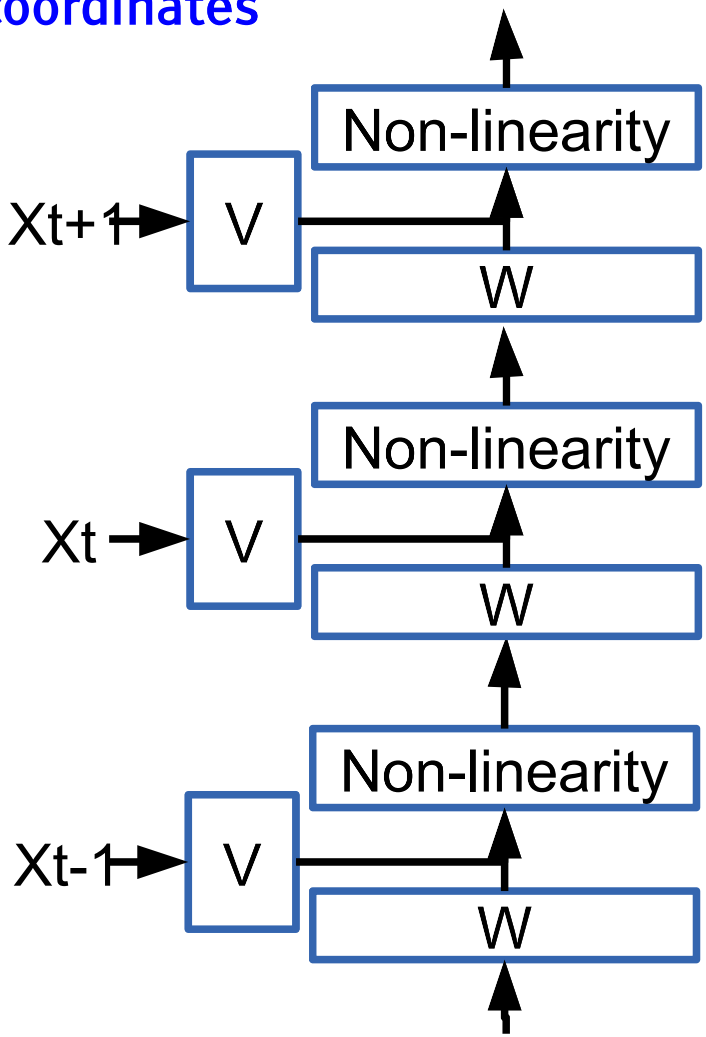
[ArXiv:1602.06662]

Recurrent neural net, and the vanishing gradient problem

Y LeCun

- When the dynamics is not invertible, gradient back-propagation fails
- Idea 1: make the W matrix orthogonal
- Idea 2: perform L2 pooling over pairs of coordinates
- QM!





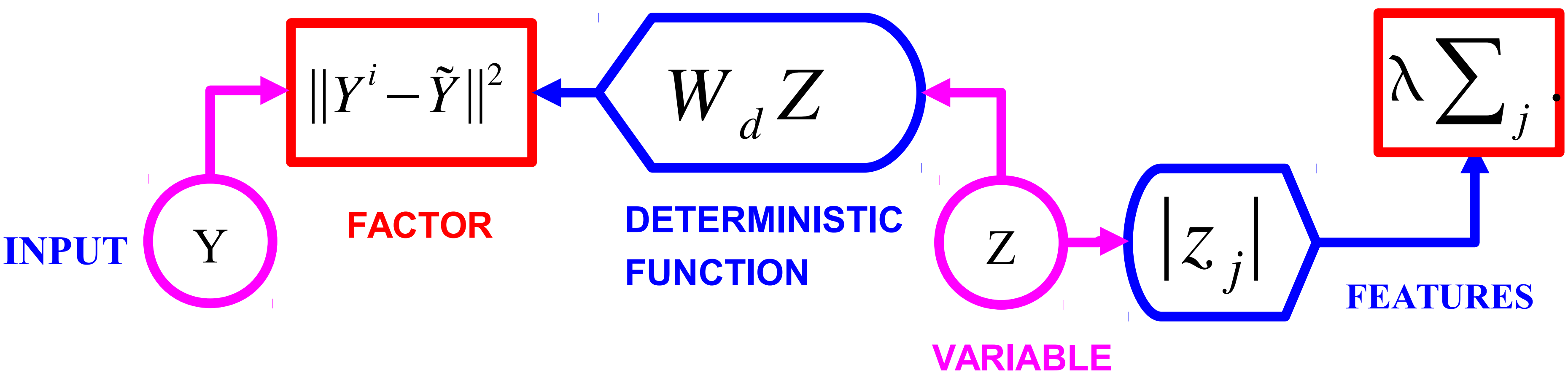
Learning to Perform Approximate Inference LISTA

Sparse Modeling: Sparse Coding + Dictionary Learning

[Olshausen & Field 1997]

- Sparse linear reconstruction
- Energy = reconstruction_error + code_prediction_error + code_sparsity

$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \lambda \sum_j |z_j|$$

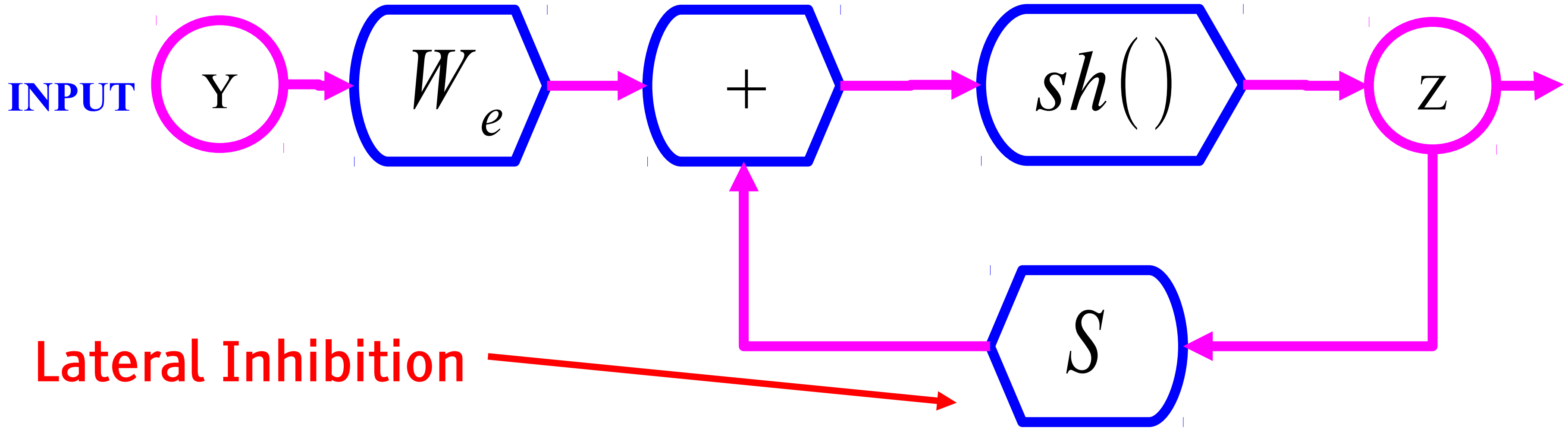


- Inference is expensive: ISTA/FISTA, CGIHT, coordinate descent....

$$Y \rightarrow \hat{Z} = argmin_Z E(Y, Z)$$

Better Idea: Give 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/FISTA reparameterized:

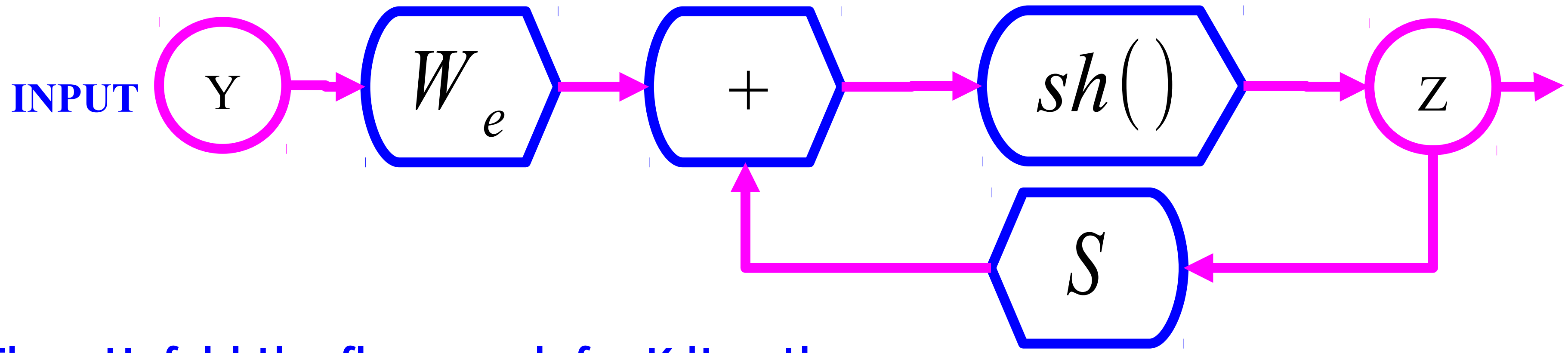
$$Z(t+1) = \mathrm{Shrinkage}_{\lambda/L}\left[W_e^T Y + SZ(t)\right]; \quad W_e = \frac{1}{L}W_d; \quad S = I - \frac{1}{L}W_d^T W_d$$

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

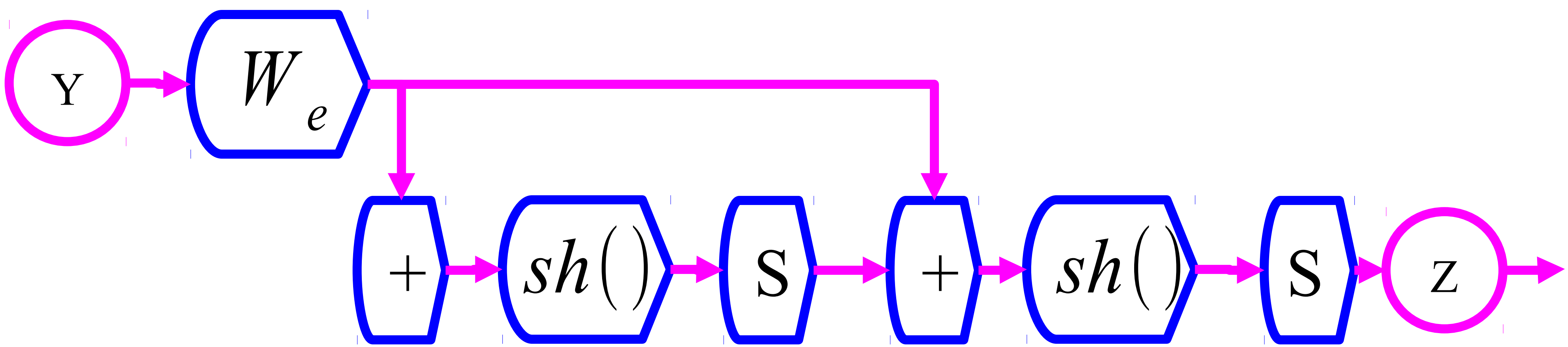
[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

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

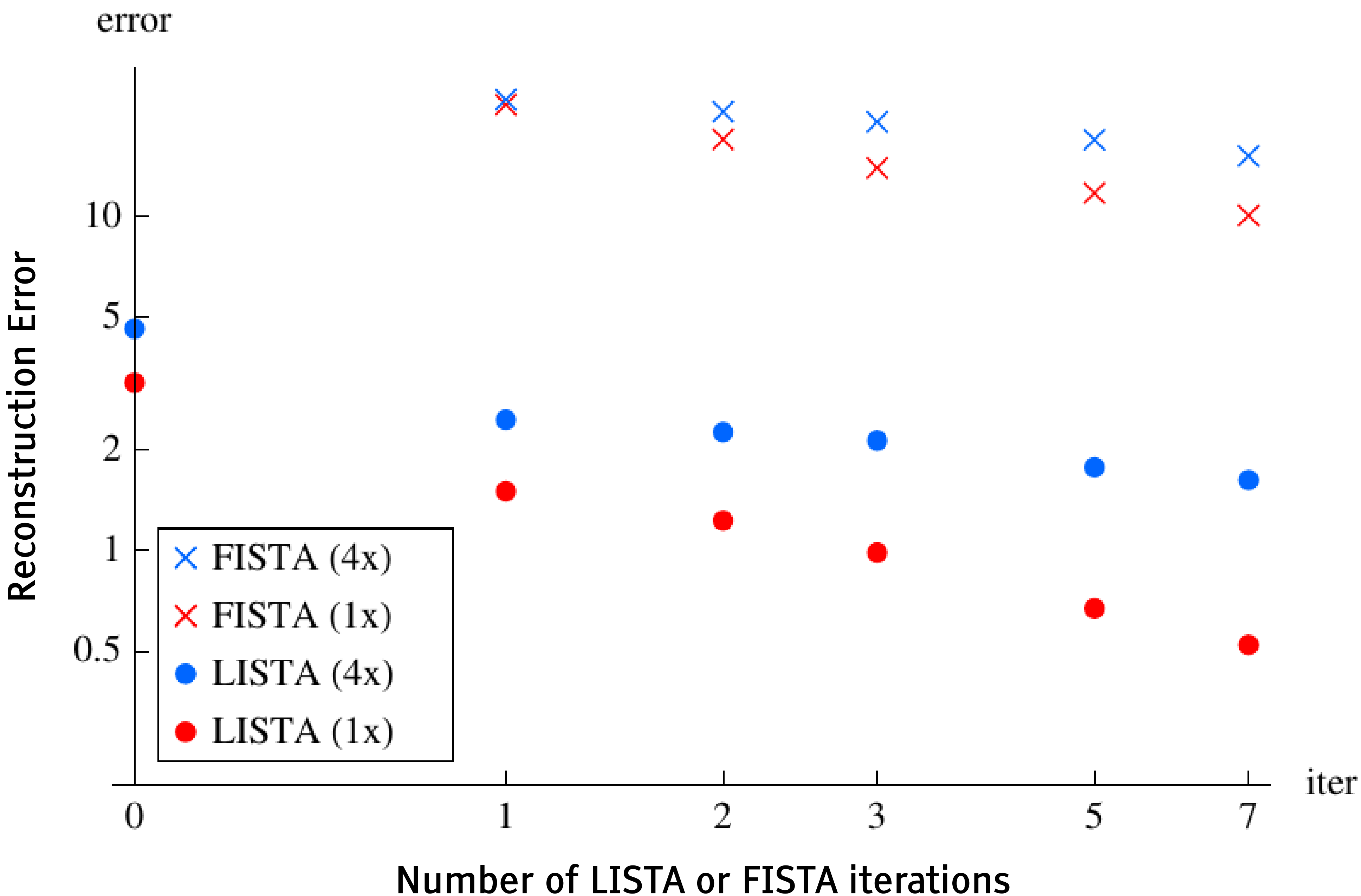
- Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters



- 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

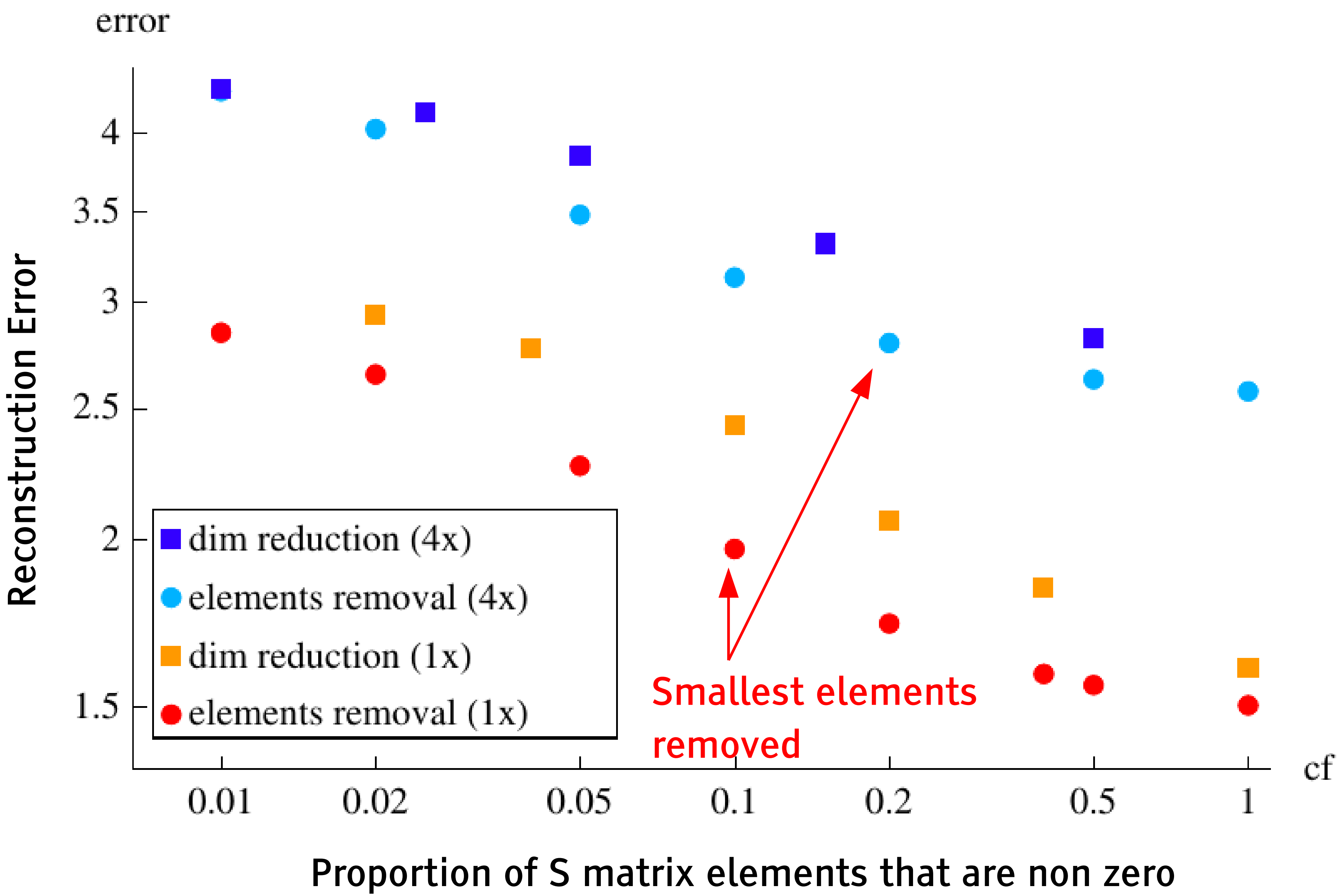


Learning ISTA (LISTA) vs ISTA/FISTA



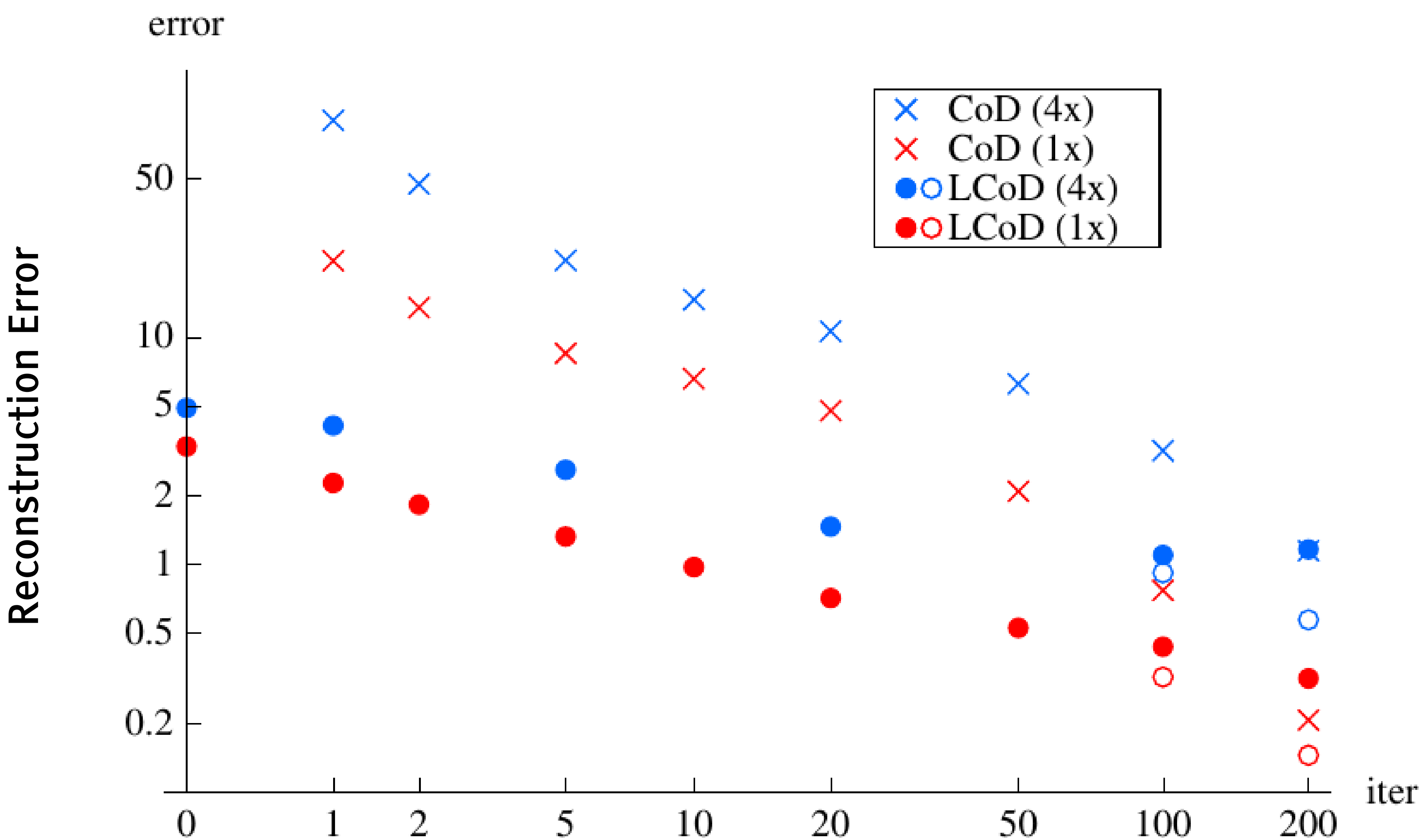
LISTA with partial mutual inhibition matrix

Y LeCun



Learning Coordinate Descent (LcoD): faster than LISTA

Y LeCun



Number of LISTA or FISTA iterations

Obstacles to AI

Obstacles to Progress in AI

- **Machines need to learn/understand how the world works**
  - Physical world, digital world, people,….
  - They need to acquire some level of common sense
- **They need to learn a very large amount of background knowledge**
  - Through observation and action
- **Machines need to perceive the state of the world**
  - So as to make accurate predictions and planning
- **Machines need to update and remember estimates of the state of the world**
  - Paying attention to important events. Remember relevant events
- **Machines neet to reason and plan**
  - Predict which sequence of actions will lead to a desired state of the world

- **Intelligence & Common Sense =**

Perception + Predictive Model + Memory + Reasoning & Planning

What is Common Sense?

- "The trophy doesn't fit in the suitcase because it's too large/small"
  - (winograd schema)

- "Tom picked up his bag and left the room"

- We have common sense because we know how the world works
- How do we get machines to learn that?

Common Sense is the ability to fill in the blanks

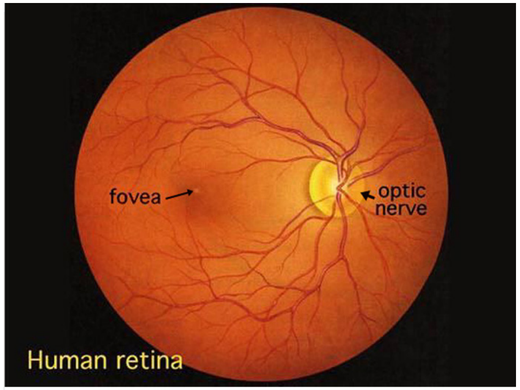


Fig. 1. Human retina as seen through an opthalmoscope.

- Infer the state of the world from partial information
- Infer the future from the past and present
- Infer past events from the present state

- Filling in the visual field at the retinal blind spot
- Filling in occluded images
- Fillling in missing segments in text, missing words in speech.
- Predicting the consequences of our actions
- Predicting the sequence of actions leading to a result

- Predicting any part of the past, present or future percepts from whatever information is available.

- That's what predictive learning is
- But really, that's what many people mean by unsupervised learning

The Necessity of Unsupervised Learning / Predictive Learning

- The number of samples required to train a large learning machine (for any task) depends on the amount of information that we ask it to predict.
  - The more you ask of the machine, the larger it can be.

- “The brain has about 10^14 synapses and we only live for about 10^9 seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get 10^5 dimensions of constraint per second.”
  - Geoffrey Hinton (in his 2014 AMA on Reddit)
  - (but he has been saying that since the late 1970s)

- Predicting human-provided labels is not enough

- Predicting a value function is not enough

How Much Information Does the Machine Need to Predict?

Y LeCun

- **"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**

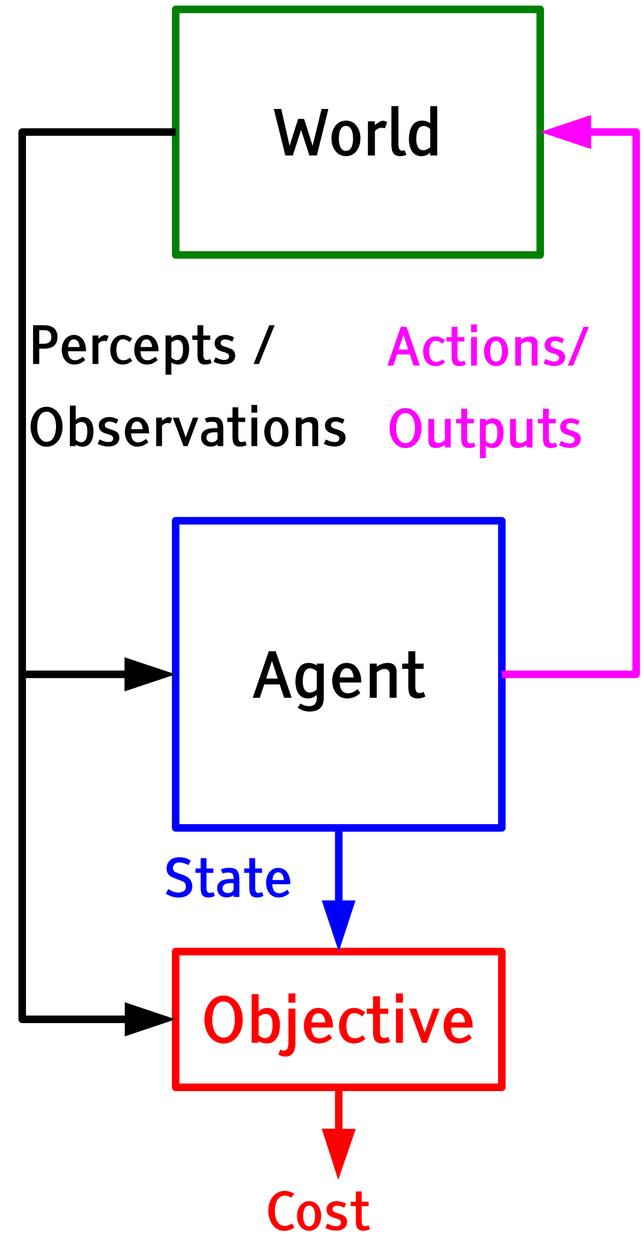
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

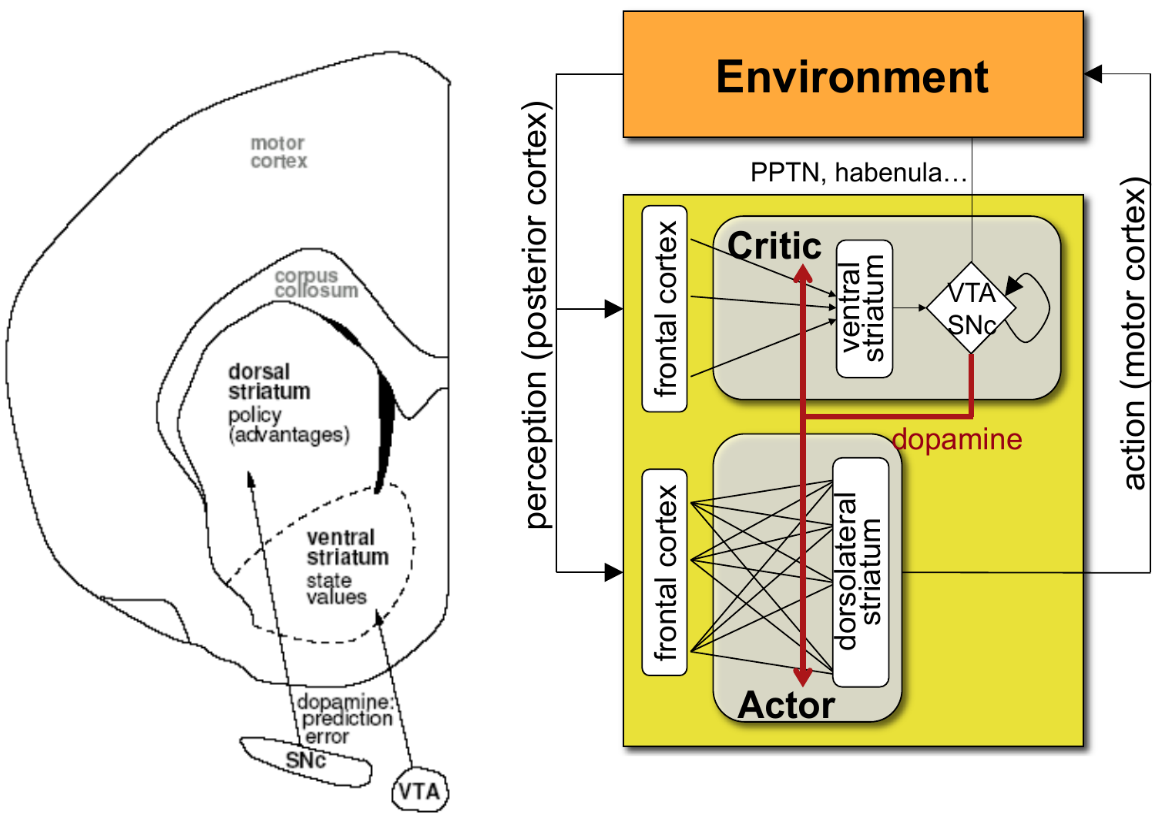
- **Unsupervised learning is the Dark Matter (or Dark Energy) of AI**

The Architecture Of an Intelligent System

AI System: Learning Agent + Immutable Objective

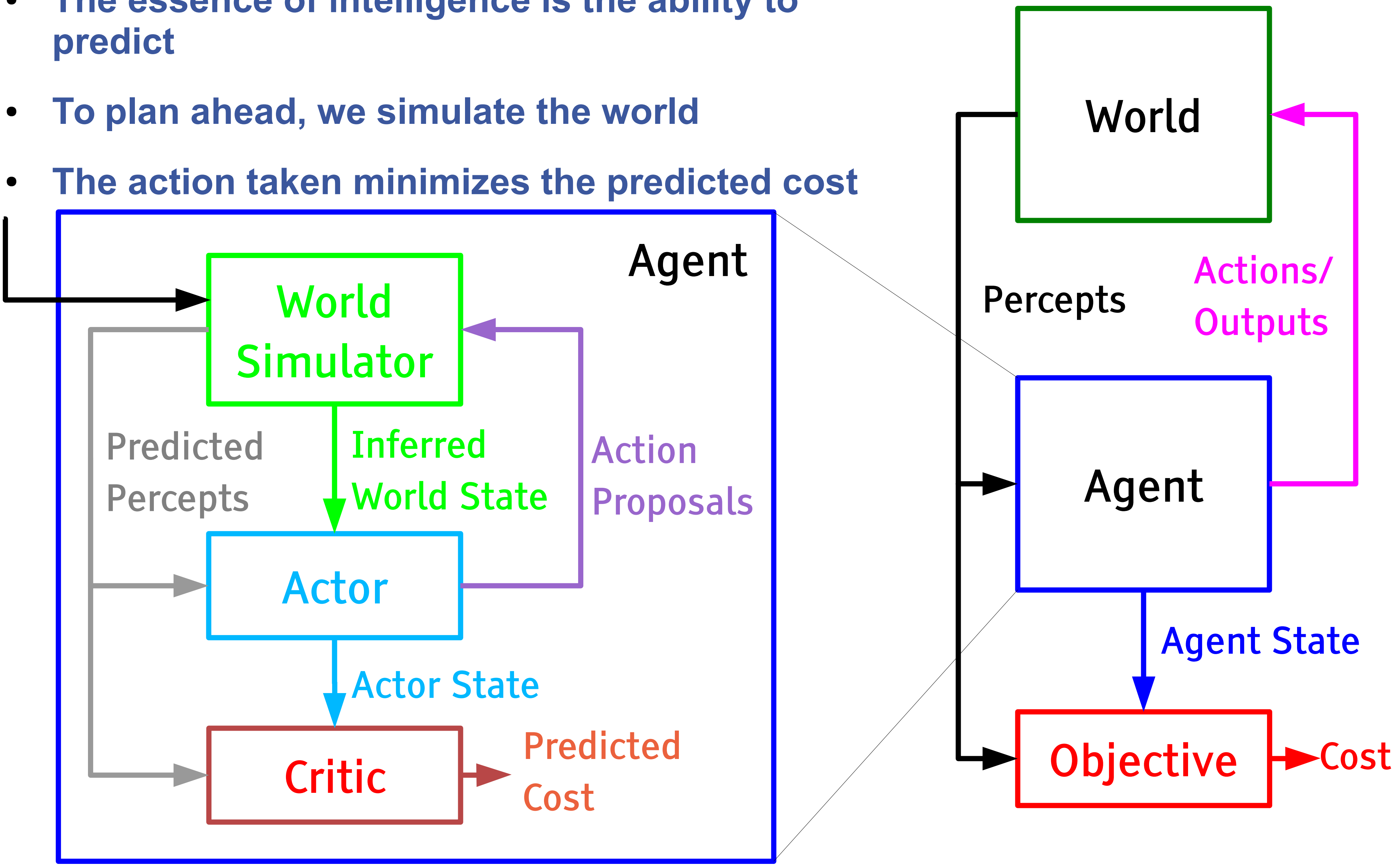
- **The agent gets percepts from the world**
- **The agent acts on the world**
- **The agents tries to minimize the long-term expected cost.**





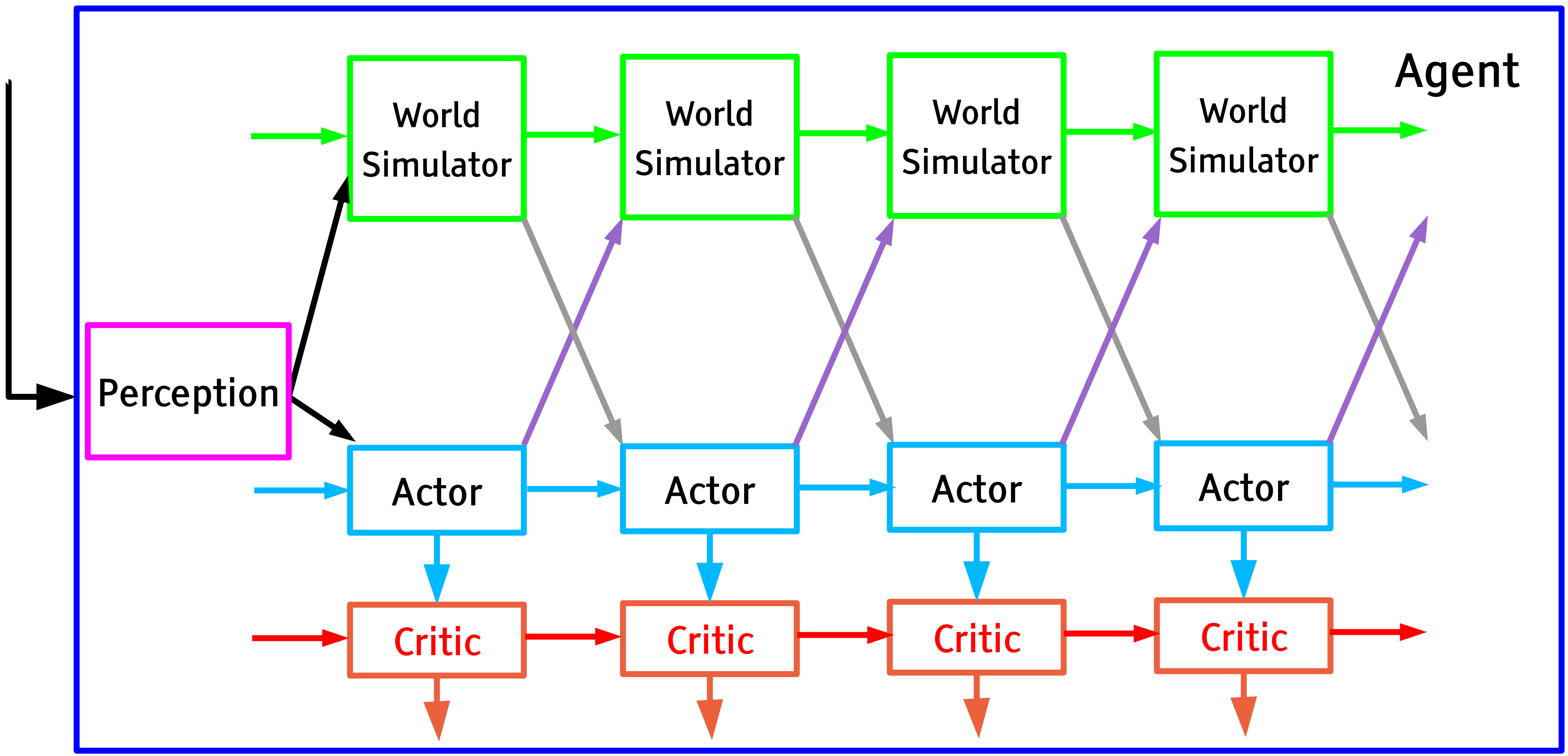
AI System: Predicting + Planning = Reasoning

- **The essence of intelligence is the ability to predict**
- **To plan ahead, we simulate the world**
- **The action taken minimizes the predicted cost**



AI System: Predicting + Planning = Reasoning

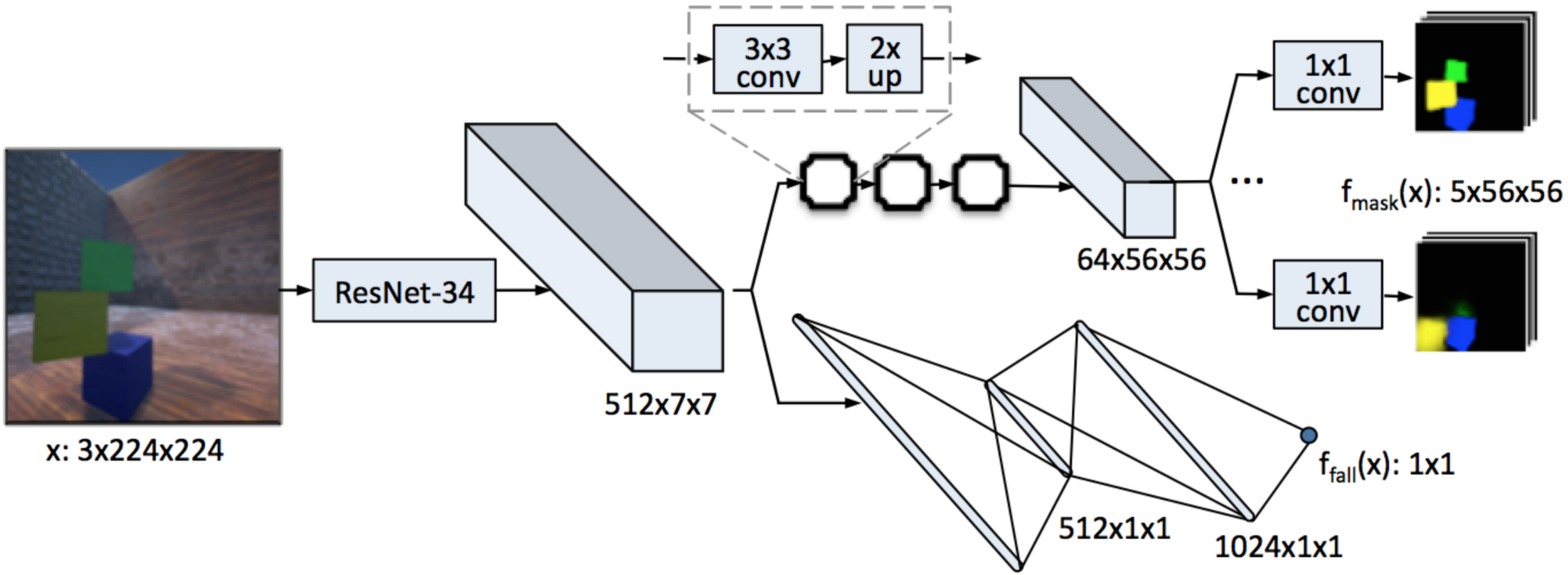
- The essence of intelligence is the ability to predict
- To plan ahead, we must simulate the world
- The action taken minimizes the predicted cost



Learning
Predictive Forward Models
Of the World

Learning Physics (PhysNet)

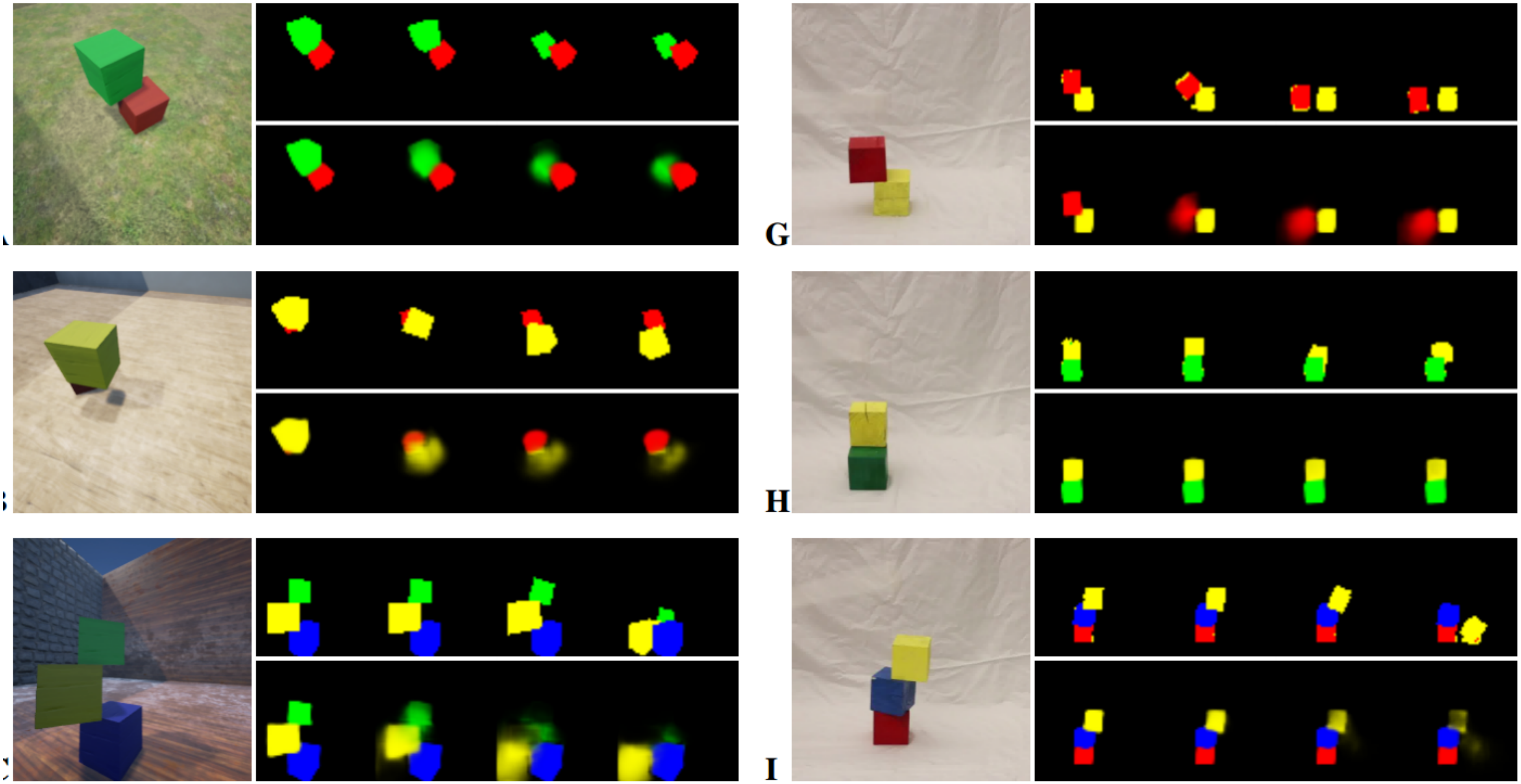
- [Lerer, Gross, Fergus arxiv:1603.01312]
  - ConvNet produces object masks that predict the trajectories of falling blocks
  - Uses the Unreal game engine.



Learning Physics (PhysNet)

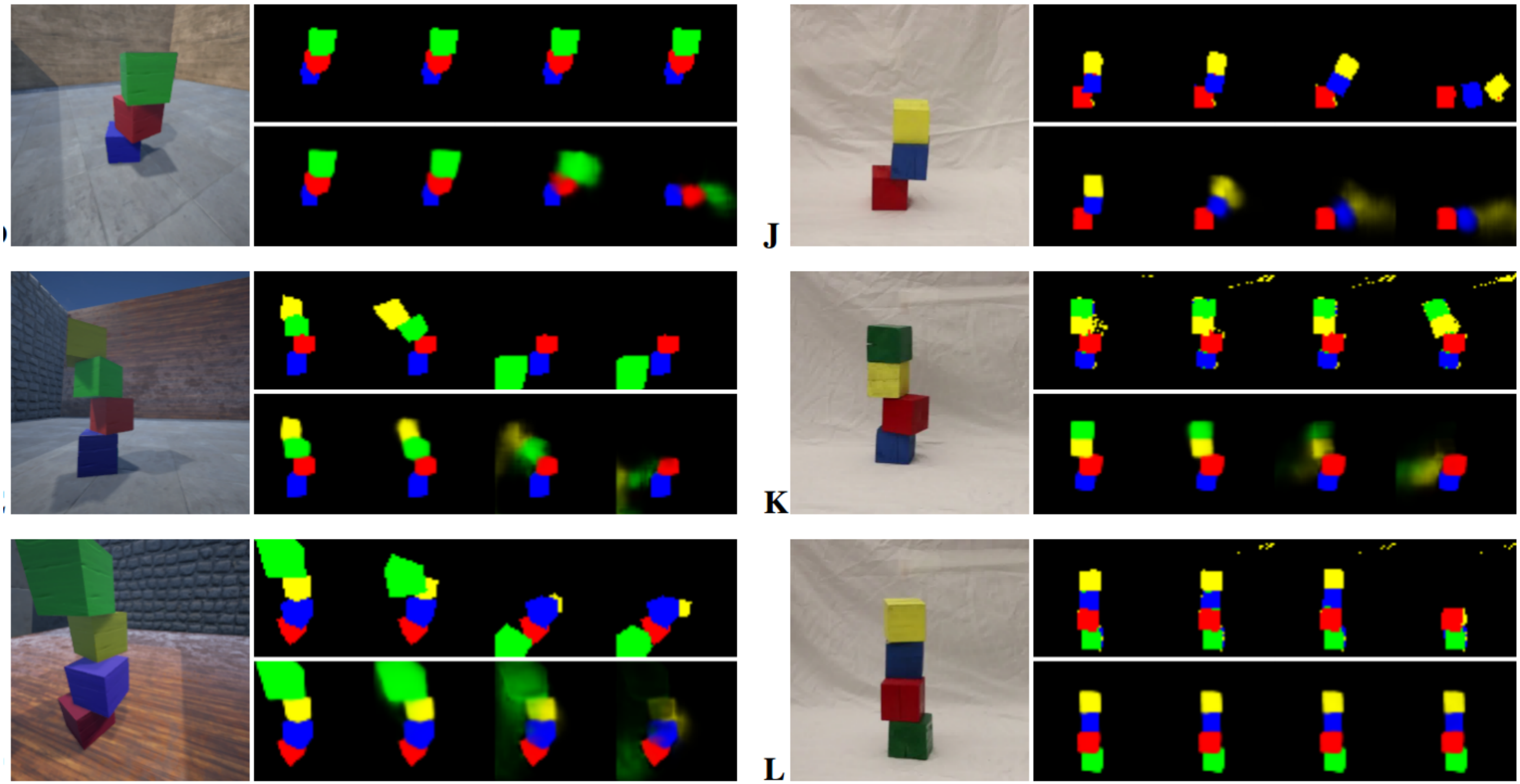
Y LeCun

- [Lerer, Gross, Fergus arxiv:1603.01312]
  - ConvNet produces object masks that predict the trajectories of falling blocks
  - Uses the Unreal game engine.



Learning Physics (PhysNet)

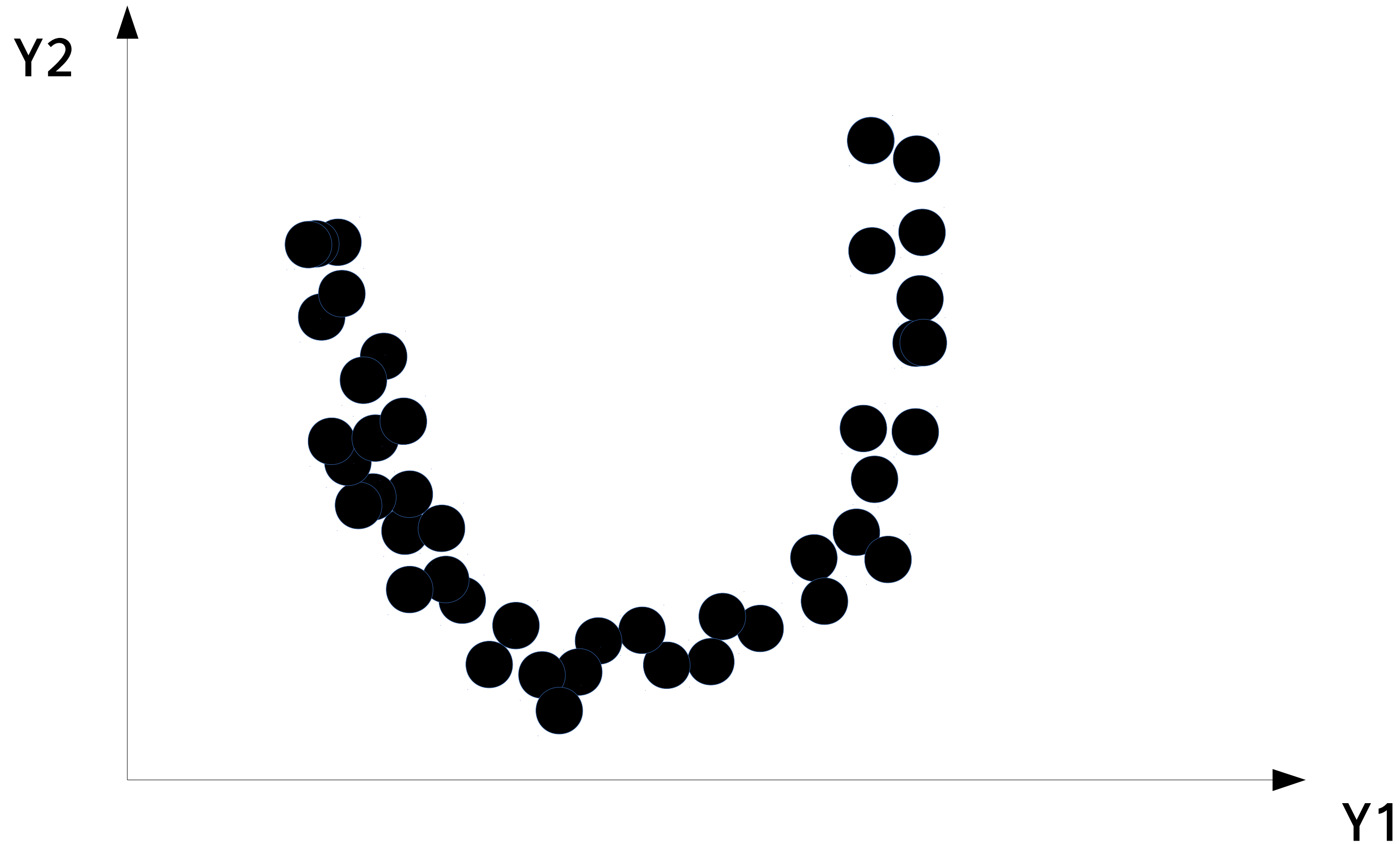
- [Lerer, Gross, Fergus arxiv:1603.01312]
  - ConvNet produces object masks that predict the trajectories of falling blocks
  - Uses the Unreal game engine.



Unsupervised Learning

Energy-Based Unsupervised Learning

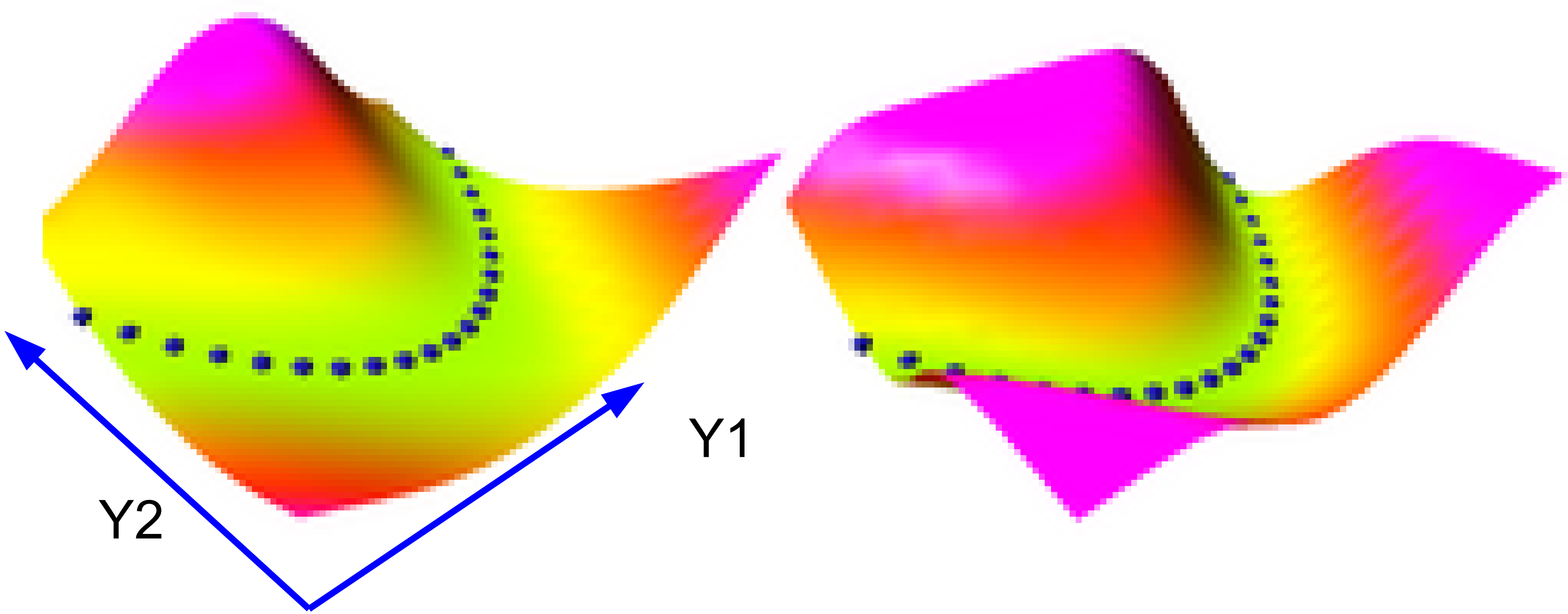
- Learning an energy function (or contrast function) that takes
  - Low values on the data manifold
  - Higher values everywhere else



Capturing Dependencies Between Variables with an Energy Function

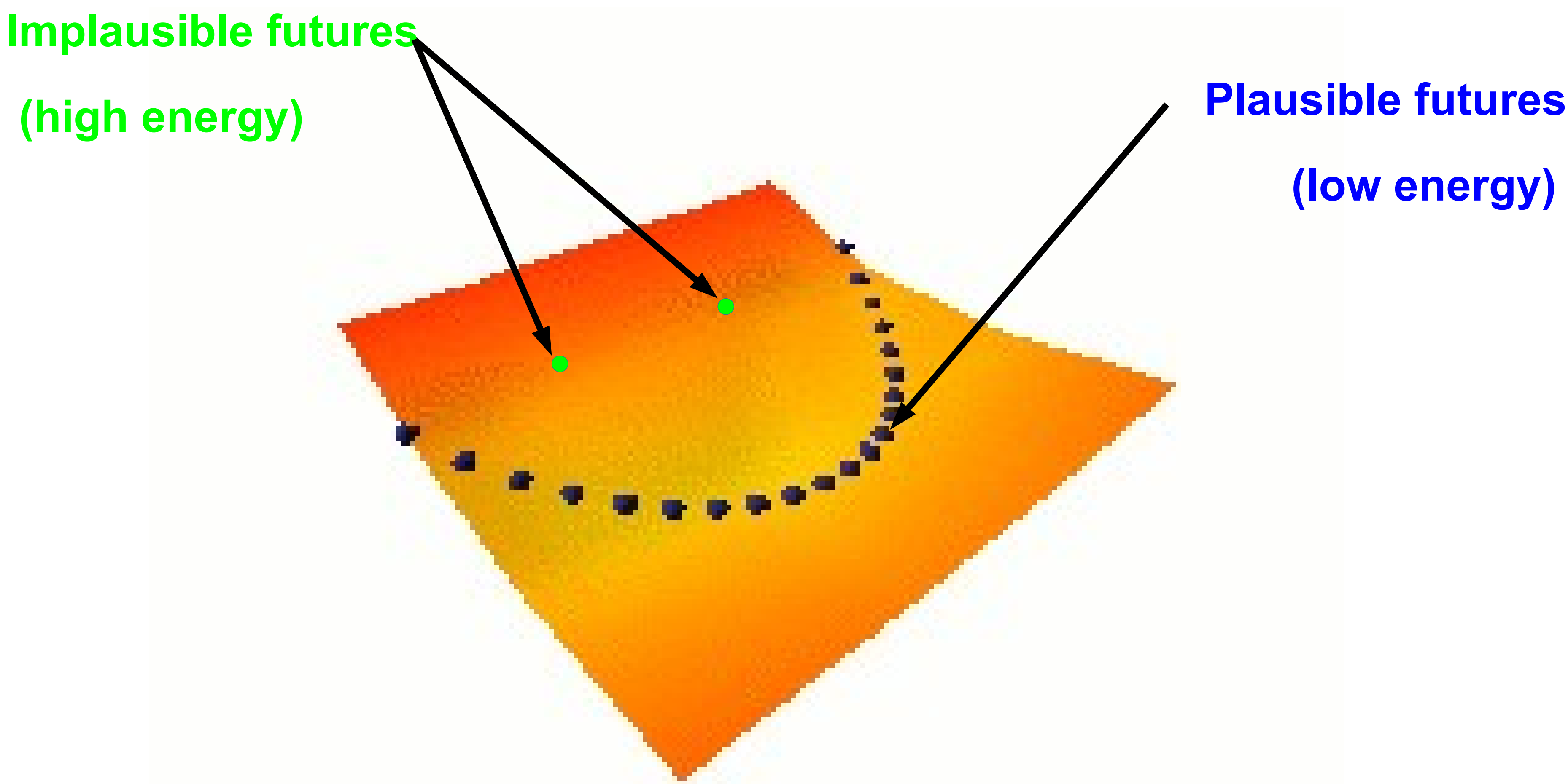
- **The energy surface is a "contrast function" that takes low values on the data manifold, and higher values everywhere else**
  - Special case: energy = negative log density
  - Example: the samples live in the manifold

$$Y_2 = (Y_1)^2$$



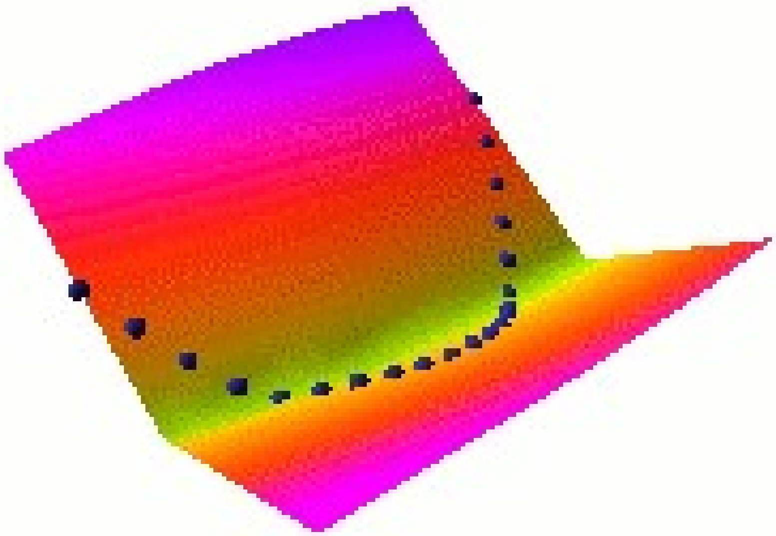
Energy-Based Unsupervised Learning

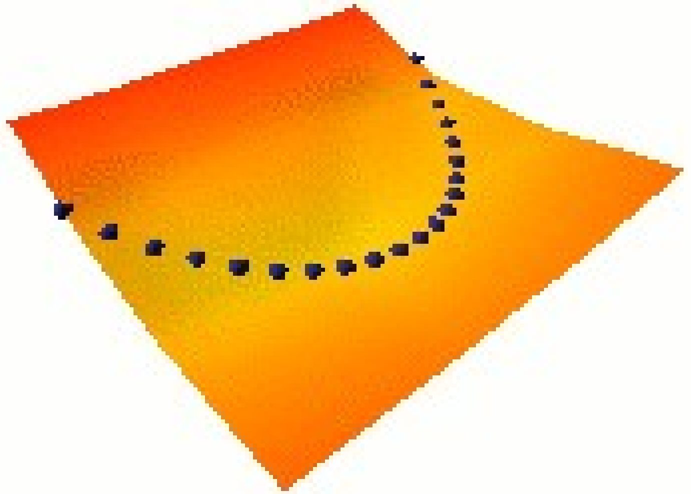
- **Energy Function:** **Takes low value on data manifold, higher values everywhere else**
- **Push down on the energy of desired outputs. Push up on everything else.**
- **But how do we choose where to push up?**



Learning the Energy Function

- **parameterized energy function E(Y,W)**
  - 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?





Seven Strategies to Shape the Energy Function

- **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)
- **3. push down of the energy of data points, push up on chosen locations**
  - contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow
- **4. minimize the gradient and maximize the curvature around data points**
  - score matching
- **5. train a dynamical system so that the dynamics goes to the manifold**
  - denoising auto-encoder
- **6. use a regularizer that limits the volume of space that has low energy**
  - Sparse coding, sparse auto-encoder, PSD
- **7. if E(Y) = ||Y - G(Y)||^2, make G(Y) as "constant" as possible.**
  - Contracting auto-encoder, saturating auto-encoder

#1: constant volume of low energy
Energy surface for PCA and K-means

- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA...

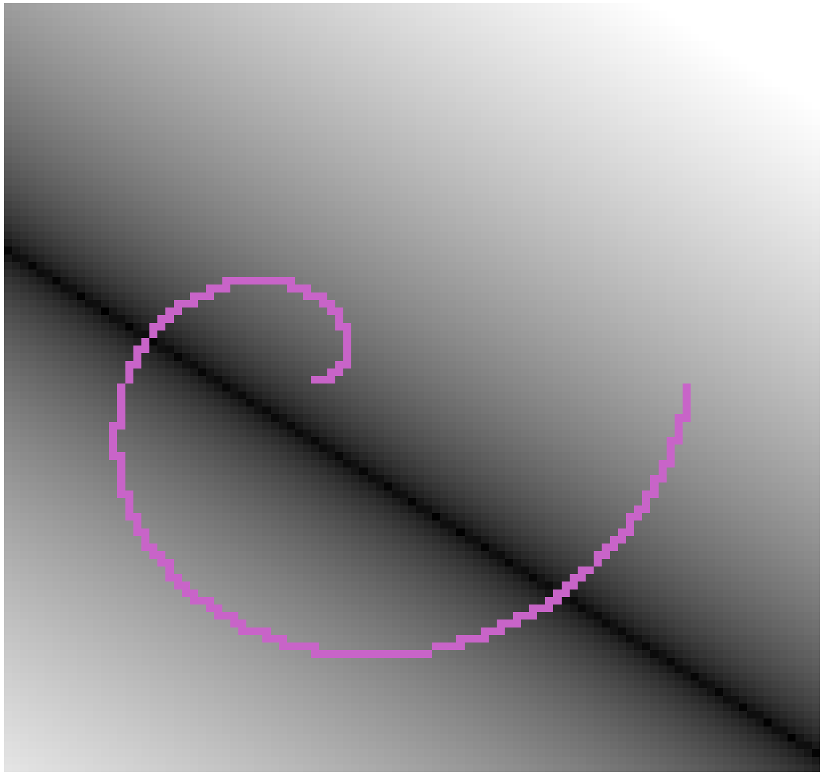
PCA

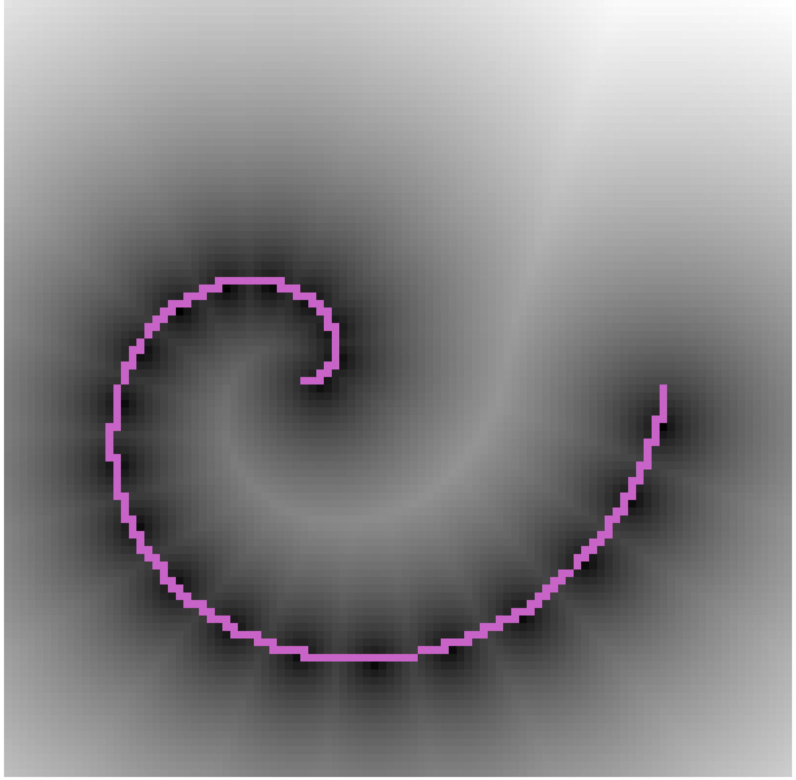
$$E(Y) = \|W^T W Y - Y\|^2$$

K-Means,

Z constrained to 1-of-K code

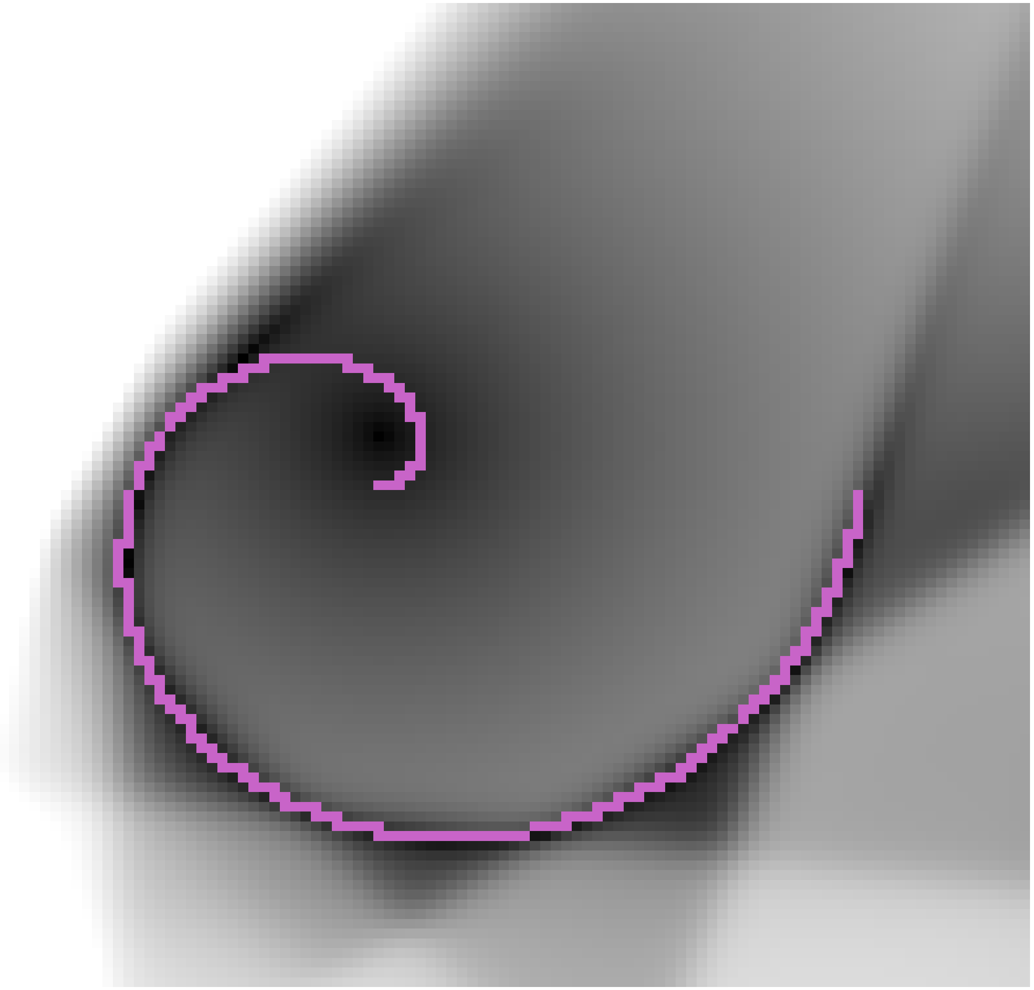
$$E(Y) = min_z \sum_i \|Y - W_i Z_i\|^2$$





#6. use a regularizer that limits the volume of space that has low energy

- Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition



Adversarial Training

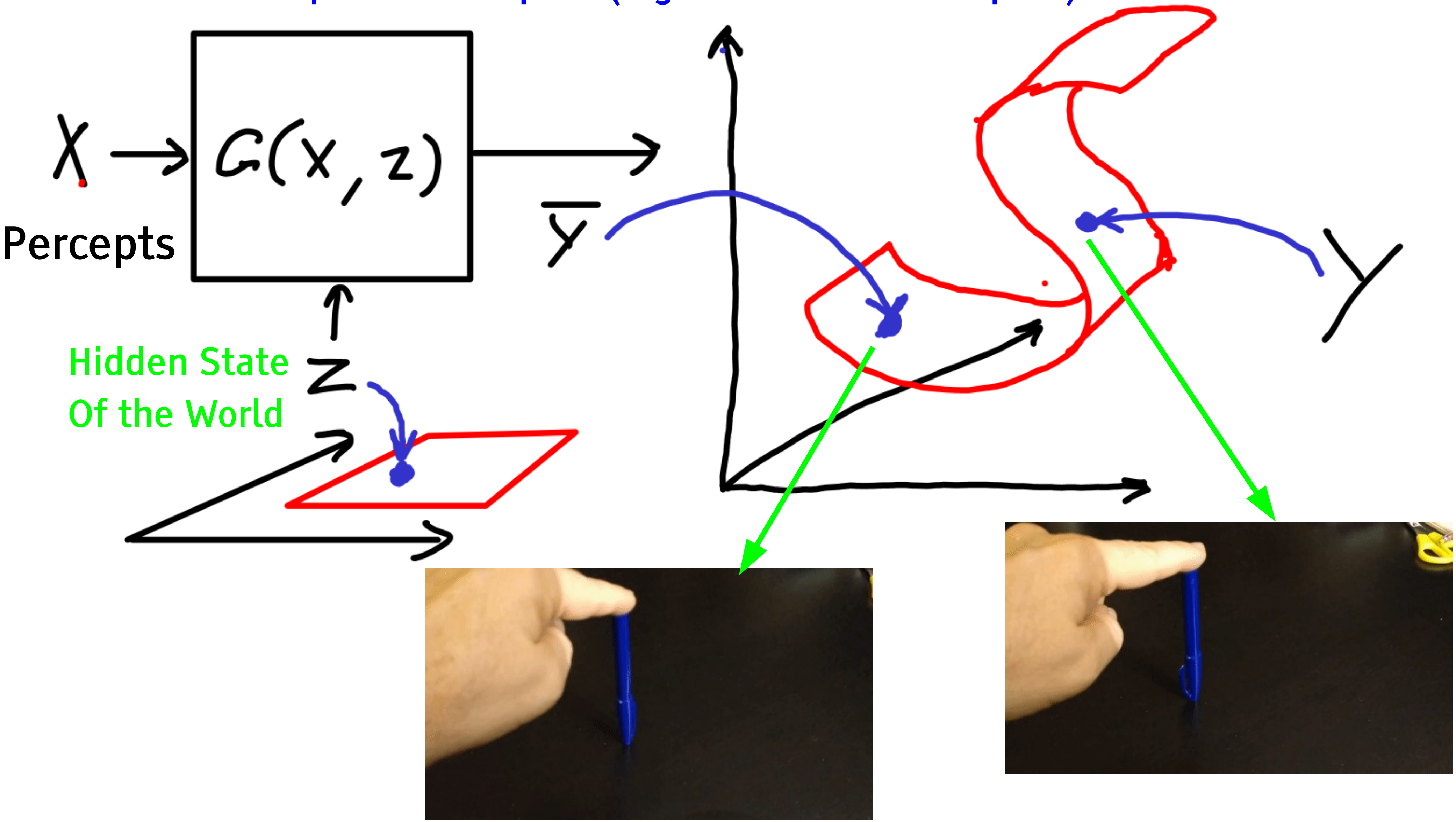
But in the real world, the future is uncertain...

- **Naïve predictive learning**
  - Minimize the prediction error
  - Predict the average of all plausible futures
  - Blurry results

- **Better predictive learning**
  - Learning the loss function
  - Predict one plausible future among many
  - Sharper results

The Hard Part: Prediction Under Uncertainty

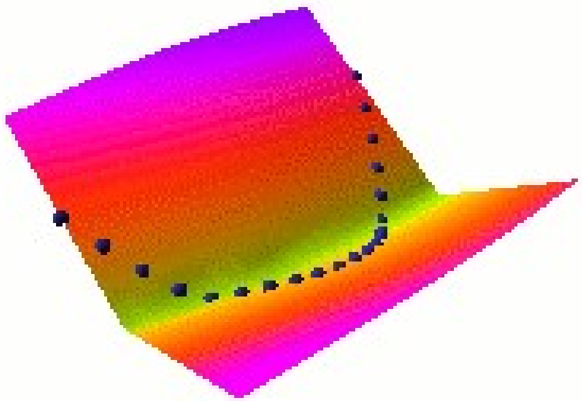
- Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).

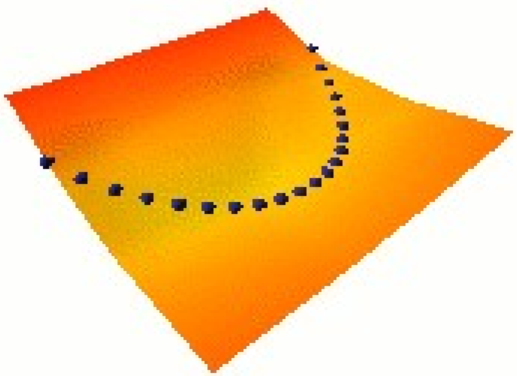


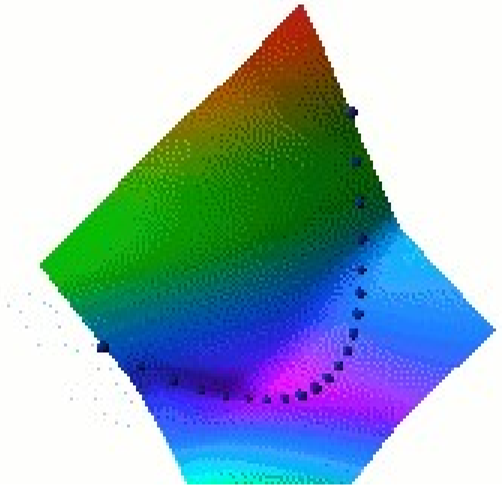
Energy-Based Unsupervised Learning

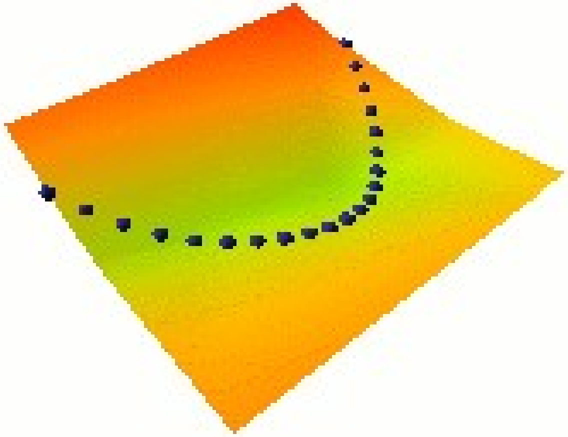
Y LeCun

- **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?



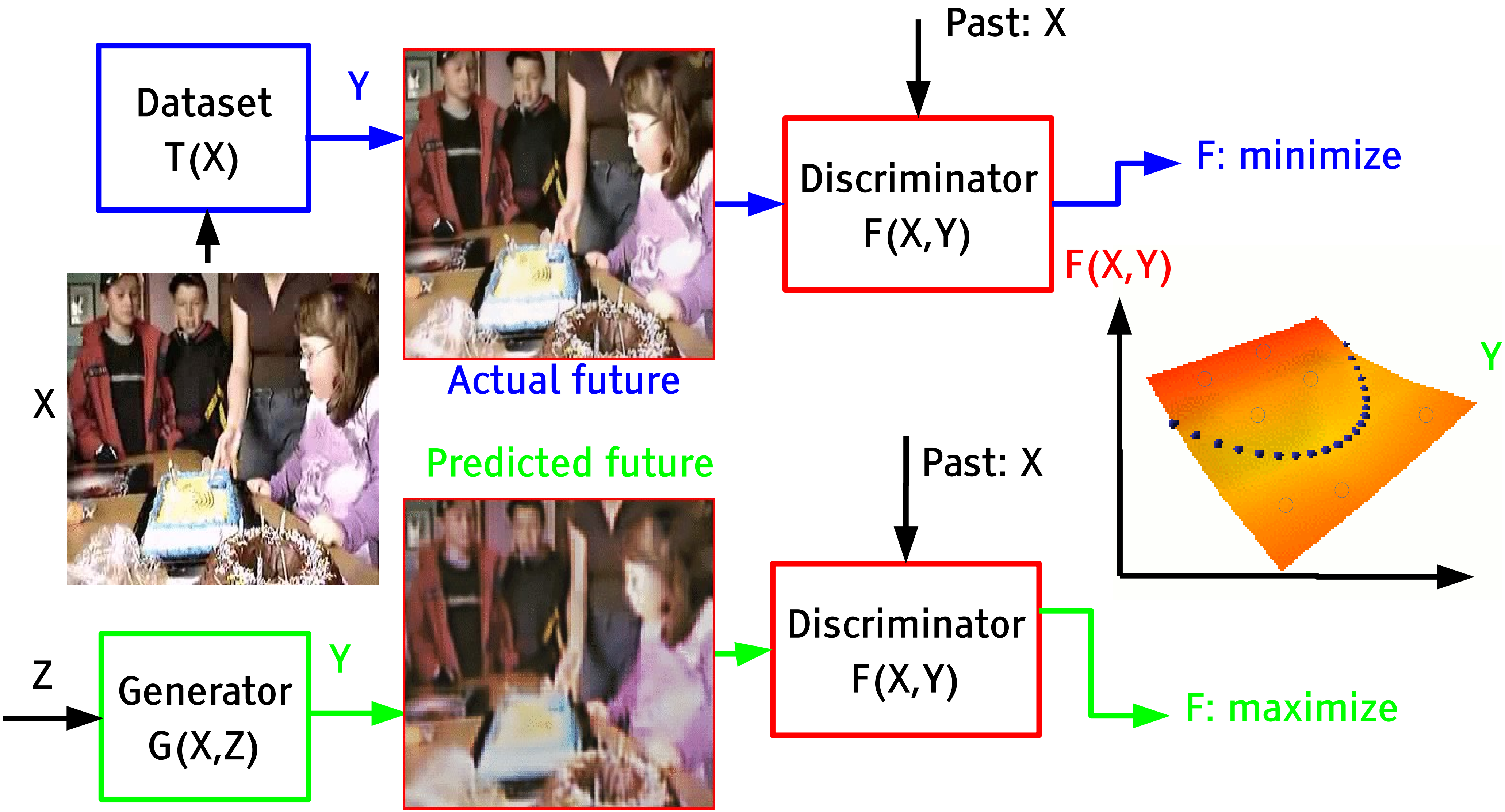






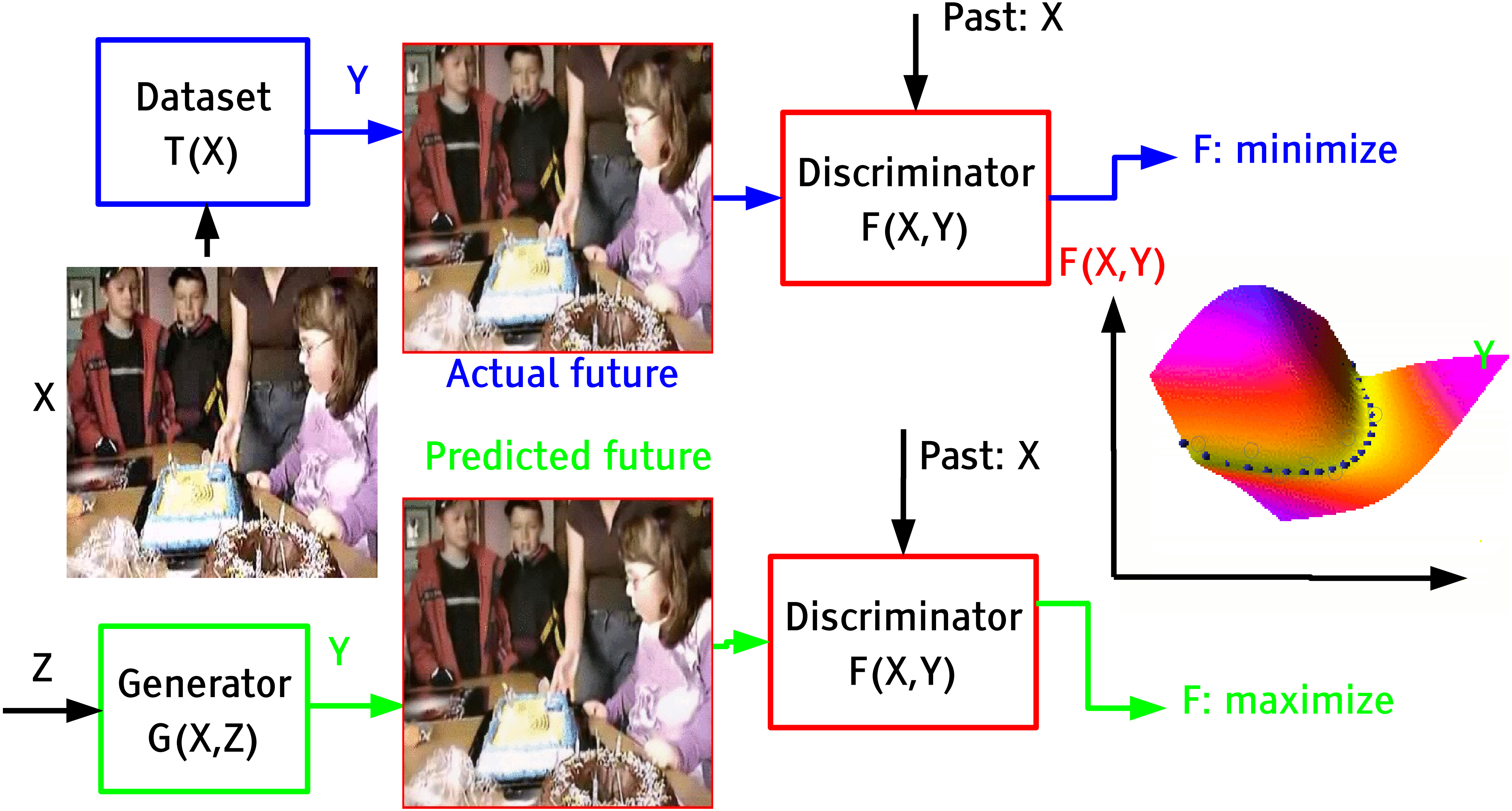
Adversarial Training: the key to predicting under uncertainty

- Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- Energy-Based GAN [Mathieu et al. 2016]



Adversarial Training: the key to predicting under uncertainty

- Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- Energy-Based GAN [Mathieu et al. 2016]



DCGAN: "reverse" ConvNet maps random vectors to images

- DCGAN: adversarial training to generate images.
- [Radford, Metz, Chintala 2015]
  - Input: random numbers; output: bedrooms.

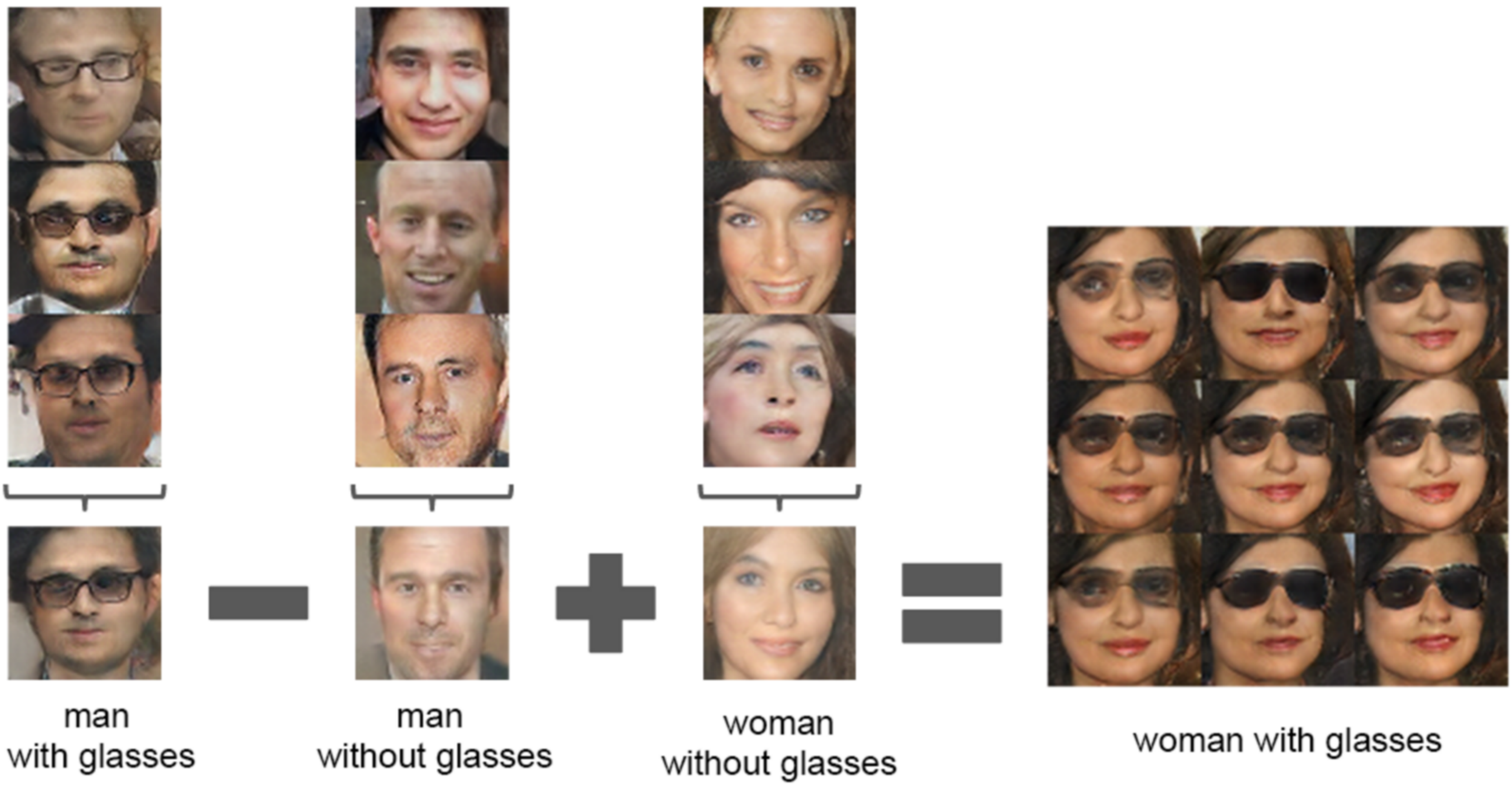
Navigating the Manifold

- DCGAN: adversarial trainin[g] to generate images.
- Trained on Manga characters
- Interpolates between character[s]

Face Algebra (in DCGAN space)

Y LeCun

- **DCGAN: adversarial training to generate images.**
  - [Radford, Metz, Chintala 2015]



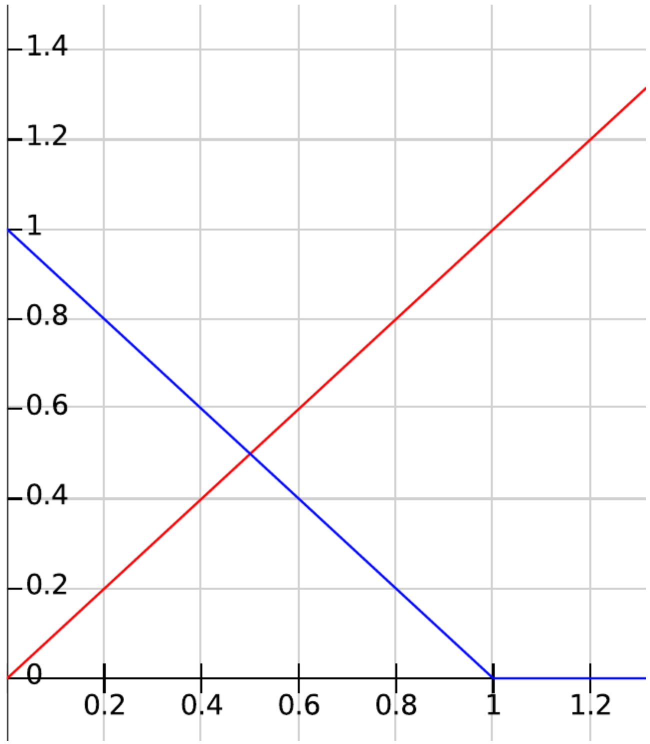
EBGAN Loss function

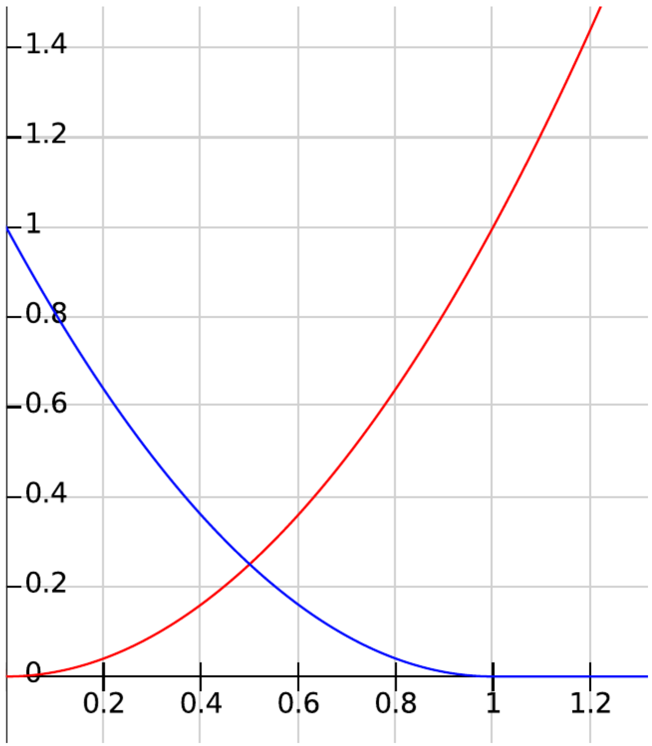
- **Loss functions for Discriminator and Generator. Assume D(x) is positive.**

$$L_D(x,z)=f(D(x))+f([m-D(G(z))]^+)$$

$$L_G(z)=f(D(G(z)))$$

- **f must be strictly increasing & convex, with f(0)=0**
  - Examples: half-wave rectification, square





EBGAN solutions are Nash Equilibria

- **Loss functions for Discriminator and Generator. D(x) is positive.**

$$L_D(x,z) = f(D(x)) + f([m - D(G(z))]^+)$$

$$L_G(z) = f(D(G(z)))$$

- **f must be strictly increasing & convex with f(0)=0**

- **(1) there is a Nash equilibrium, (2) if it is reached, the distributions are equal**

We define $V(G,D) = \int_{x,z} \mathcal{L}_D(x,z) p_{data}(x) p_z(z) \mathrm{d}x \mathrm{d}z$ and $U(G,D) = \int_z \mathcal{L}_G(z) p_z(z) \mathrm{d}z$.

$$V(G^*,D^*) \leq V(G^*,D) \qquad \forall D$$
$$U(G^*,D^*) \leq U(G,D^*) \qquad \forall G$$

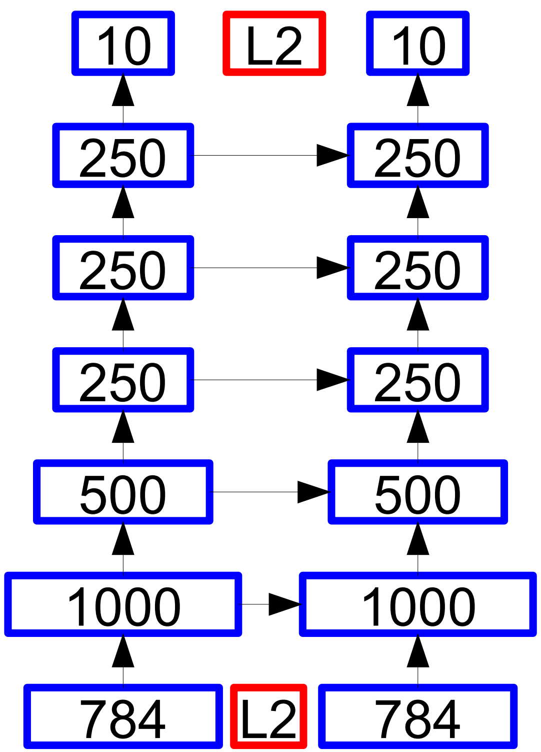
Theorem 1. *If (D^*, G^*) is a Nash equilibrium of the system, then $p_{G^*} = p_{data}$ almost everywhere, and $V(D^*, G^*) = m$.*

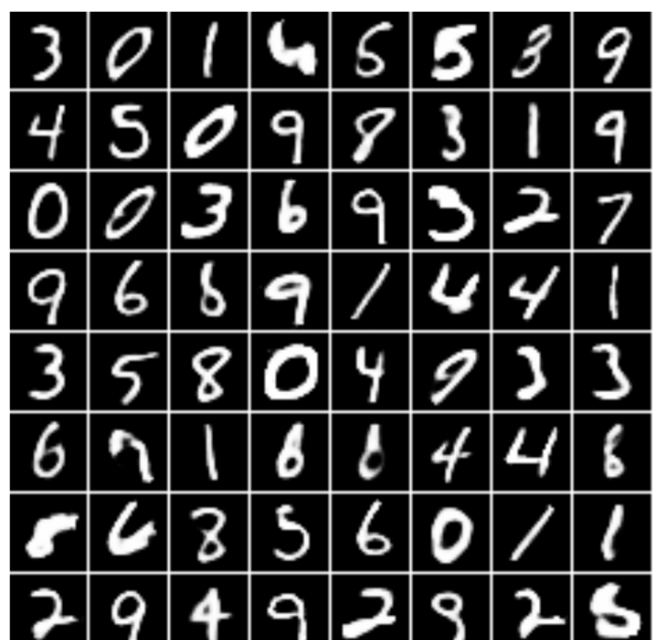
Theorem 2. *Nash equilibrium of this system exists and is characterized by (a) $p_{G^*} = p_{data}$ (almost everywhere) and (b) there exists a constant $\gamma \in [0, m]$ such that $D^*(x) = \gamma$ (almost everywhere).*[1]

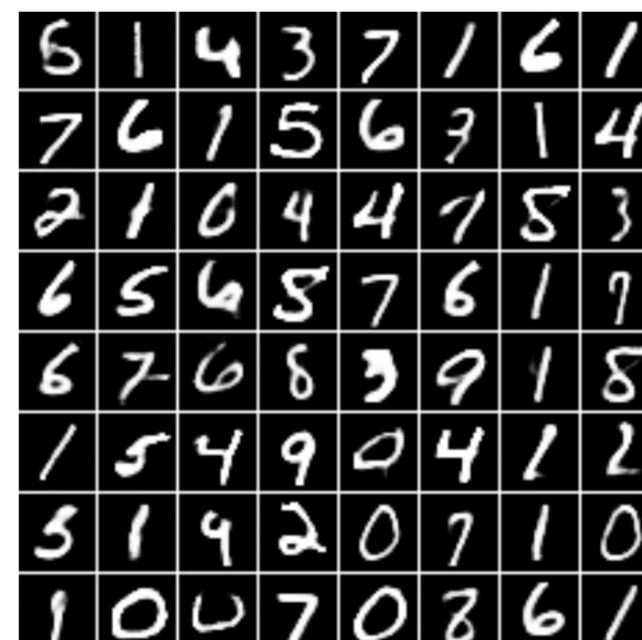
EBGAN in which D is a Ladder Network

- **Ladder Network: auto-encoder with skip connections [Rasmus et al 2015]**
- **Permutation-invariant MNIST (fully connected nets)**

model	**100**	**200**	**1000**
LN bottom-layer-cost, reported in Pezeshki et al. (2015)	1.69±0.18	-	1.05±0.02
LN bottom-layer-cost, reported in Rasmus et al. (2015)	1.09±0.32	-	0.90±0.05
LN bottom-layer-cost, reproduced in this work (see appendix D)	1.36±0.21	1.24±0.09	1.04±0.06
LN bottom-layer-cost within EBGAN framework	**1.04±0.12**	**0.99±0.12**	**0.89±0.04**
Relative percentage improvement	23.5%	20.2%	14.4%

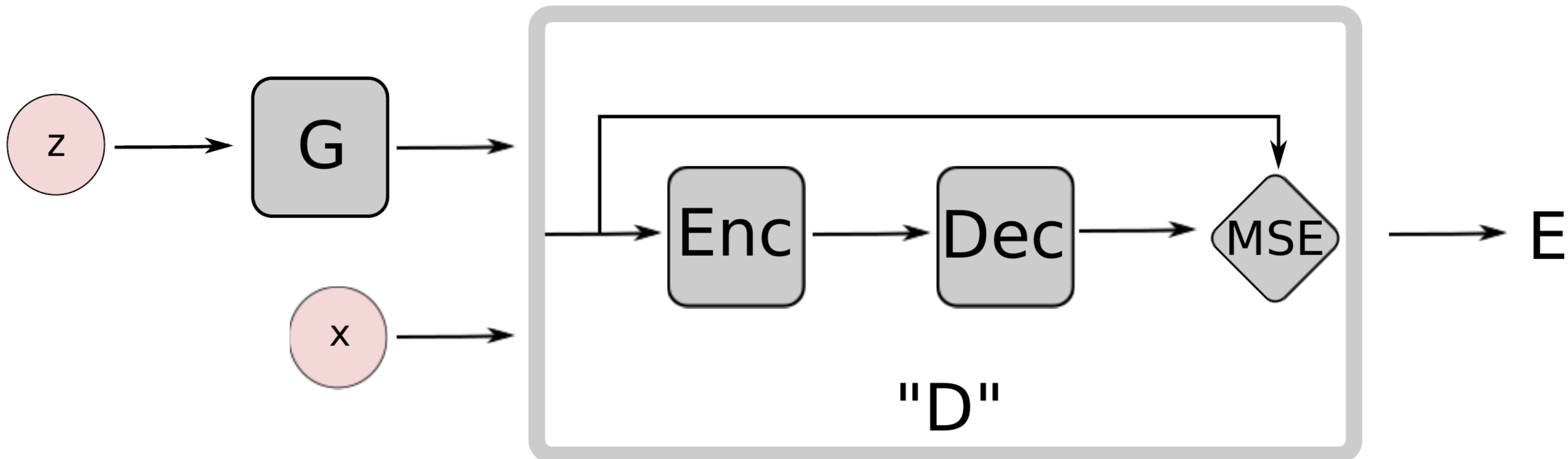






Energy-Based GAN [Zhao, Mathieu, LeCun: arXiv:1609.03.126]

- **Architecture: discriminator is an auto-encoder**



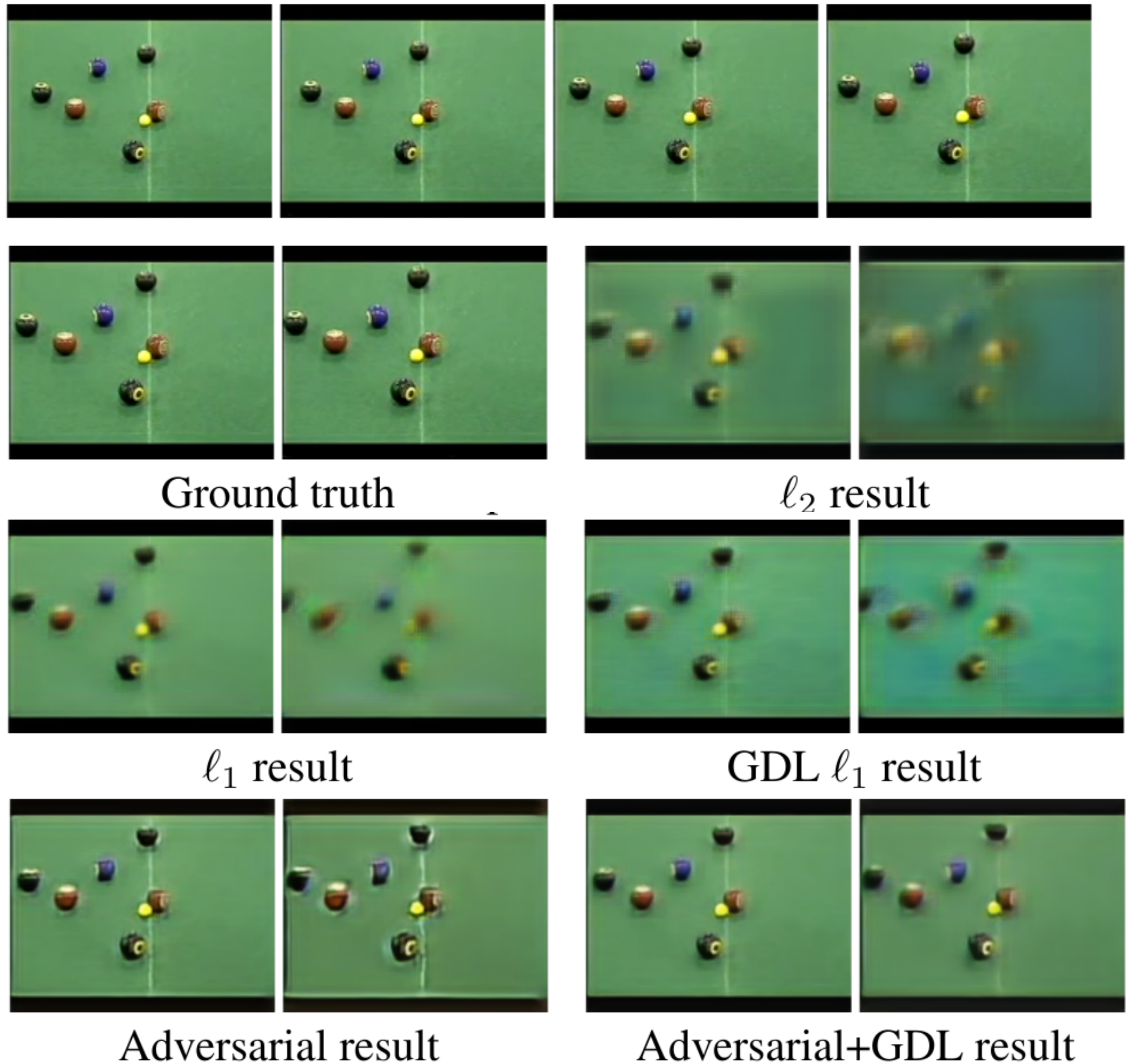
- **Loss functions**

$$
\begin{aligned}
f_D(x,z) &= D(x) + [m - D\big(G(z)\big)]^+ \\
&= \|Dec\big(Enc(x)\big) - x\| + [m - \|Dec\Big(Enc\big(G(z)\big)\Big) - G(z)\|]^+,
\end{aligned}
$$

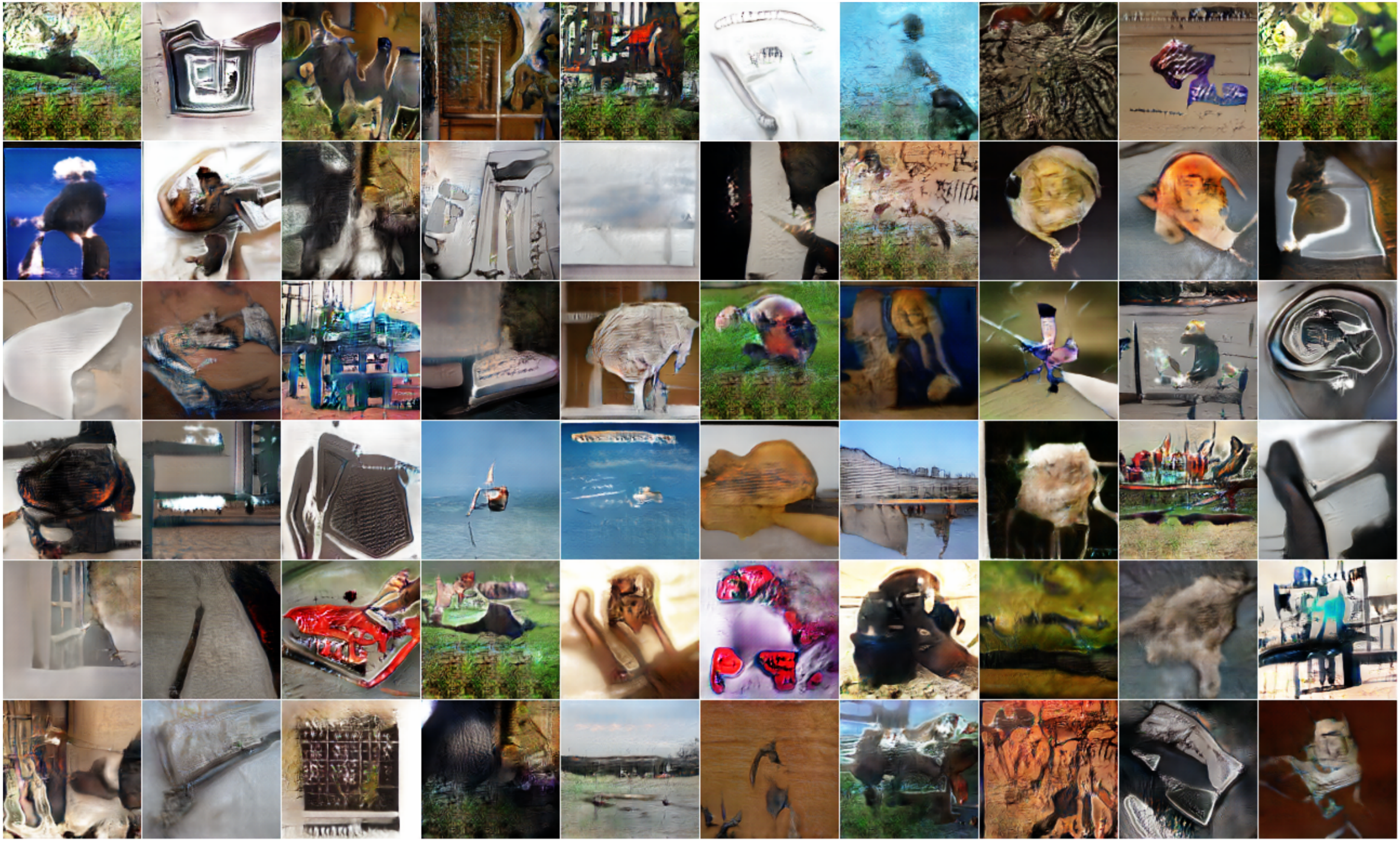
$$
\begin{aligned}
f_G(z) &= \|D\big(G(z)\big)\| \\
&= \|Dec\Big(Enc\big(G(z)\big)\Big) - G(z)\|
\end{aligned}
$$

Multi-Scale ConvNet for Video Prediction

Examples



Energy-Based GAN trained on ImageNet at 128x128 pixels



Energy-Based GAN trained on ImageNet at 256x256 pixels

Y LeCun

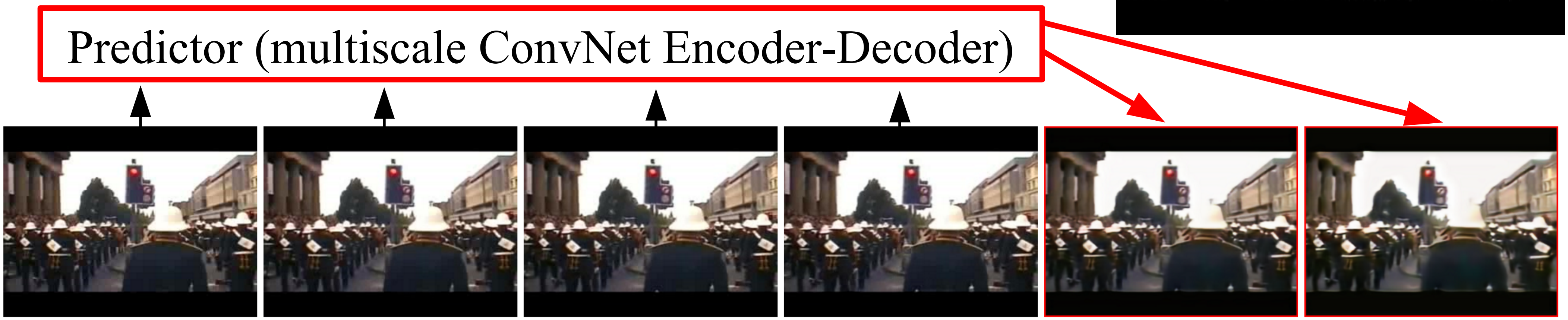
- Trained on dogs

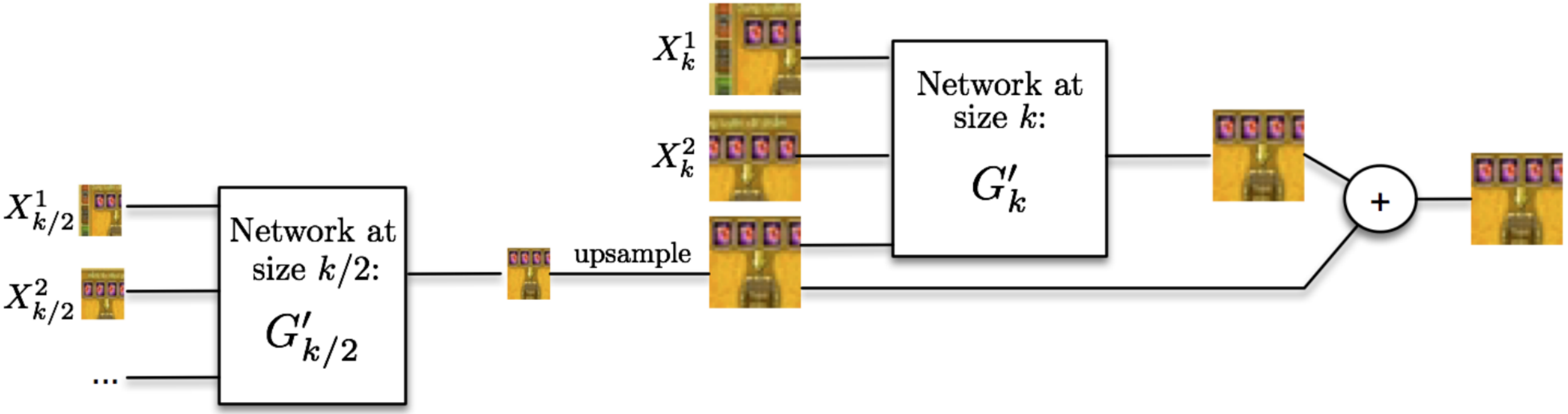
Video Prediction
(with adversarial training)

[Mathieu, Couprie, LeCun ICLR 2016]
arXiv:1511:05440

Multi-Scale ConvNet for Video Prediction

- 4 to 8 frames input → ConvNet → 1 to 8 frames output
- Multi-scale ConvNet, without pooling
- If trained with least square: blurry output



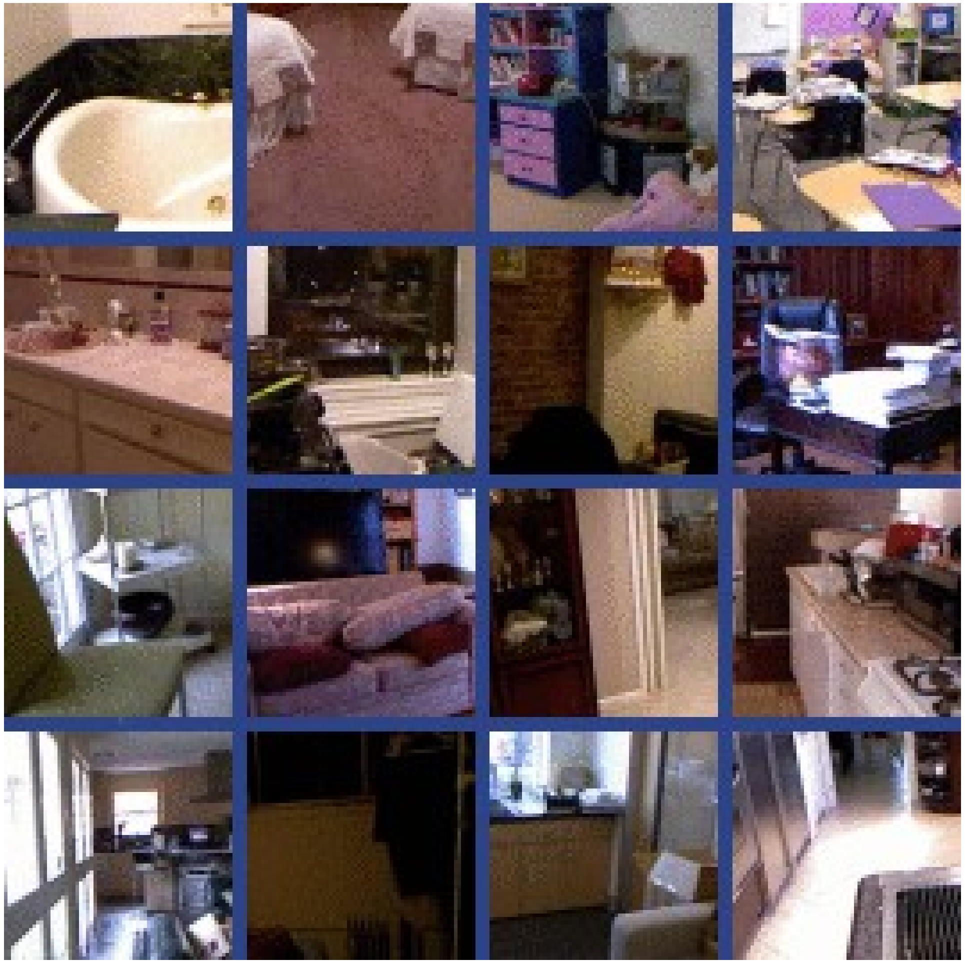


Predictive Unsupervised Learning

- 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.

Video Prediction: predicting 5 frames

Y LeCun



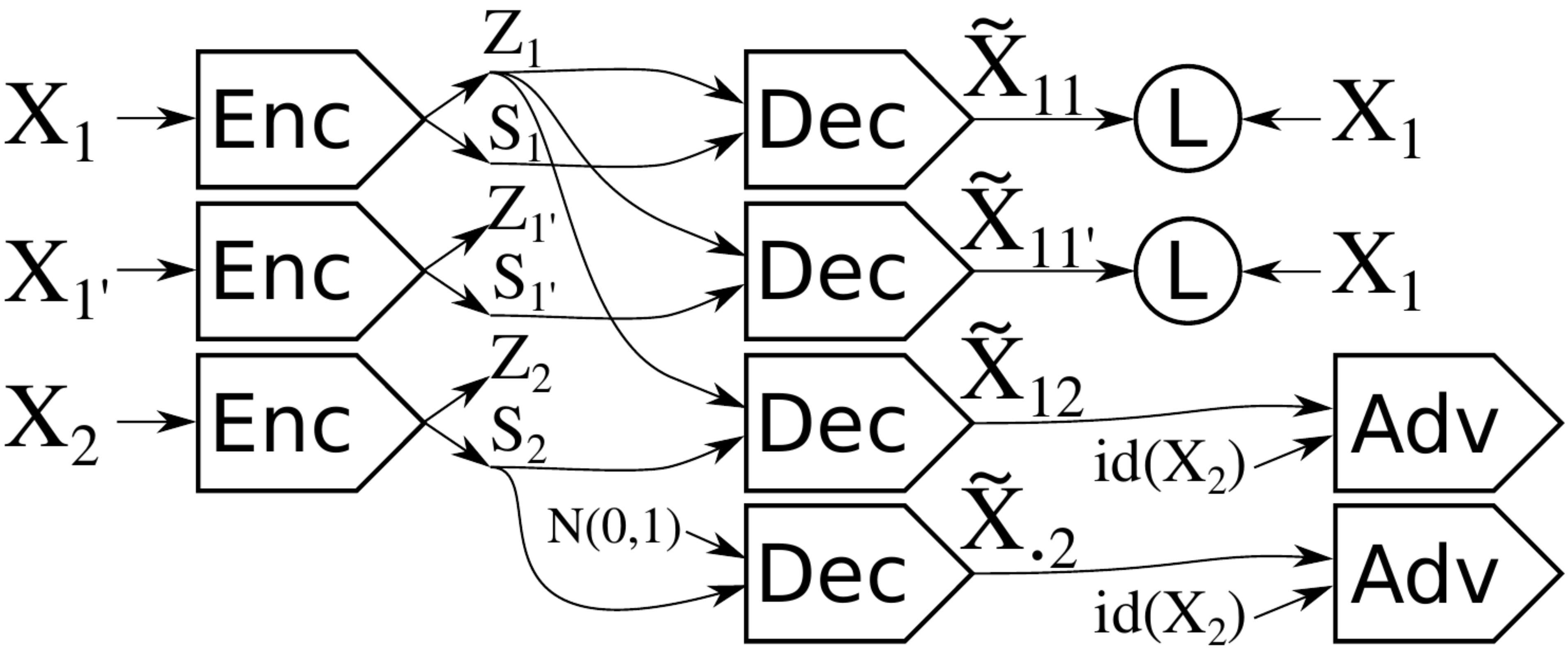
Video Prediction: predicting 5 frames

Video Prediction: predicting 50 frames

Style Transfer
(Mathieu et al. NIPS 2016)

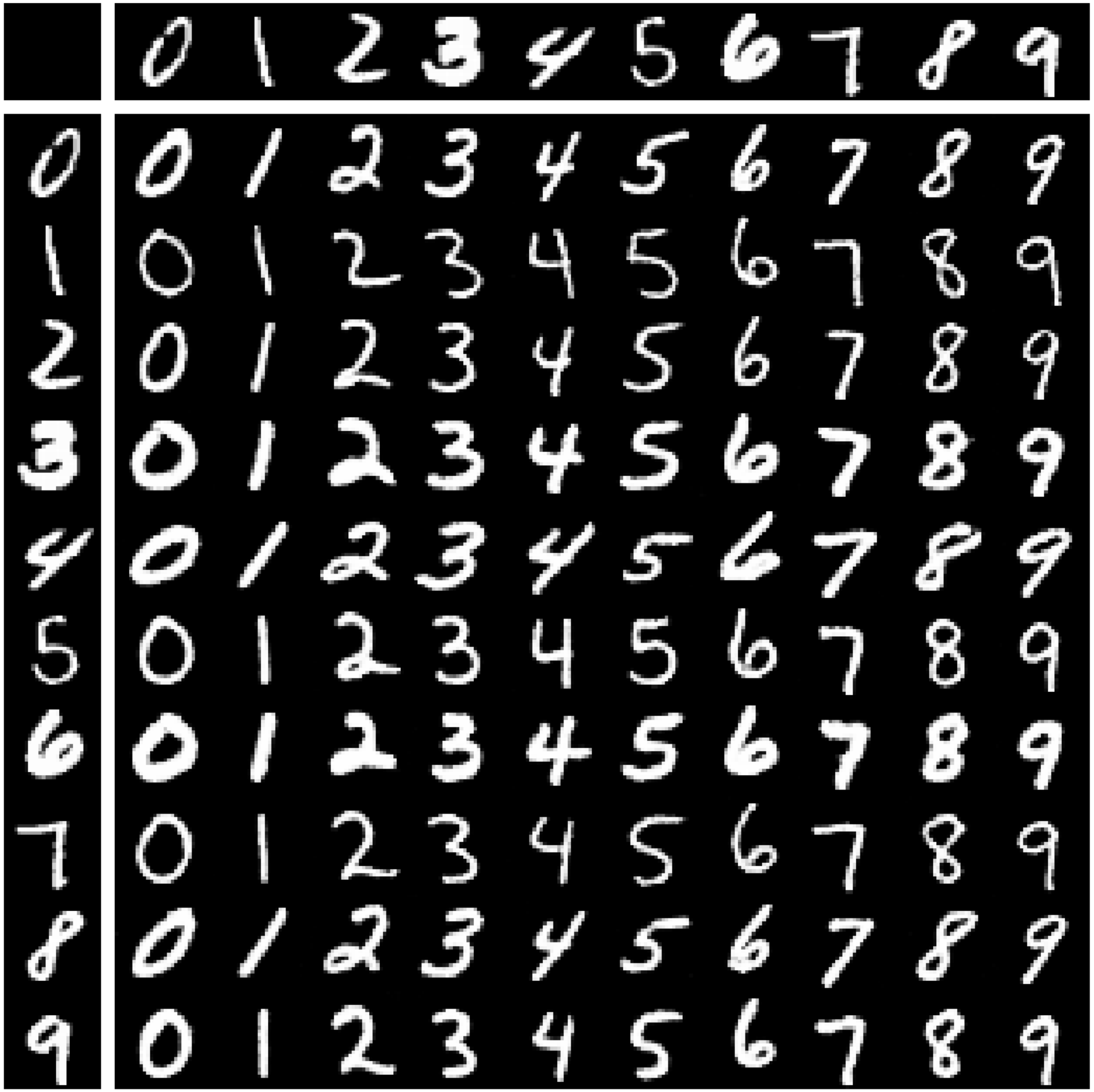
Style transfer architecture

- X1 and X1′ have same “label” (or known features)
- X2 can have any label
- S1 and S1′ are meant to represent the “label” (the known part of the representation)
- Z1, Z1′ and Z2 are the unspecified part (eg the pose)



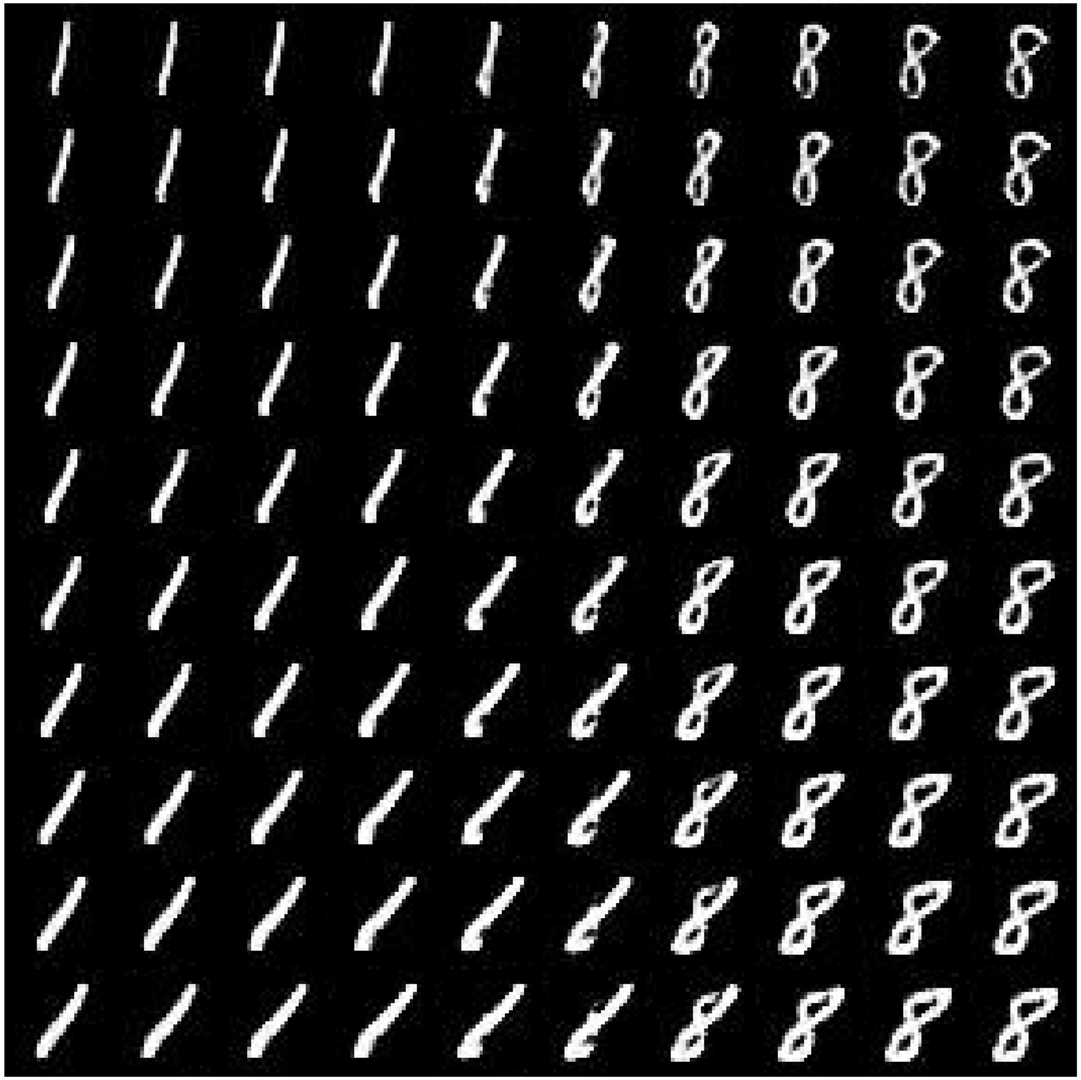
Style transfer results

- Transfer category from top row to style of left column



Style transfer: interpolation

- Interpolate between top left and bottom right characters
- Style changes vertically, identity changes horizontally.



Style transfer results

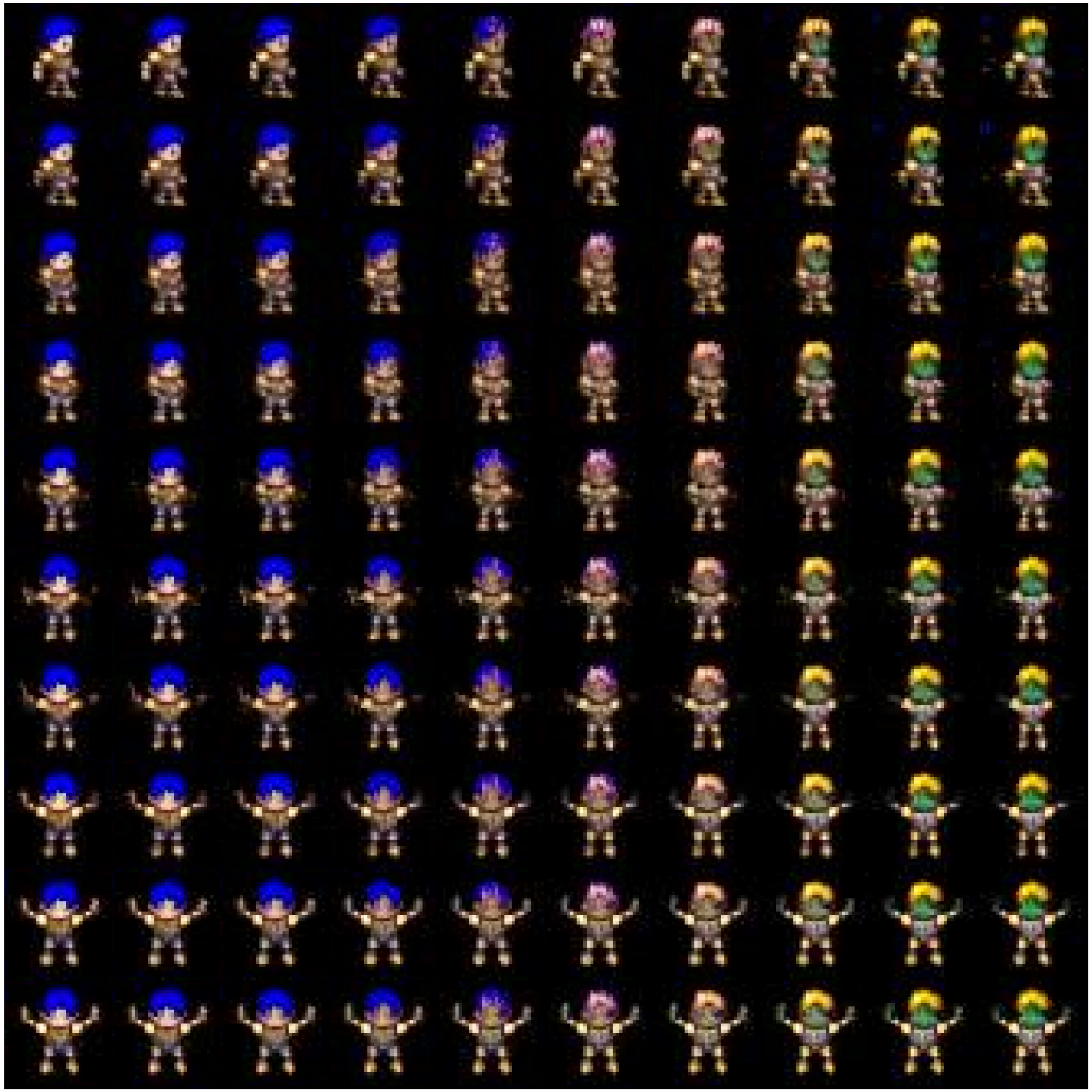
- Transfer category from top row to style of left column

Style transfer results

- Transfer category from top row to style of left column

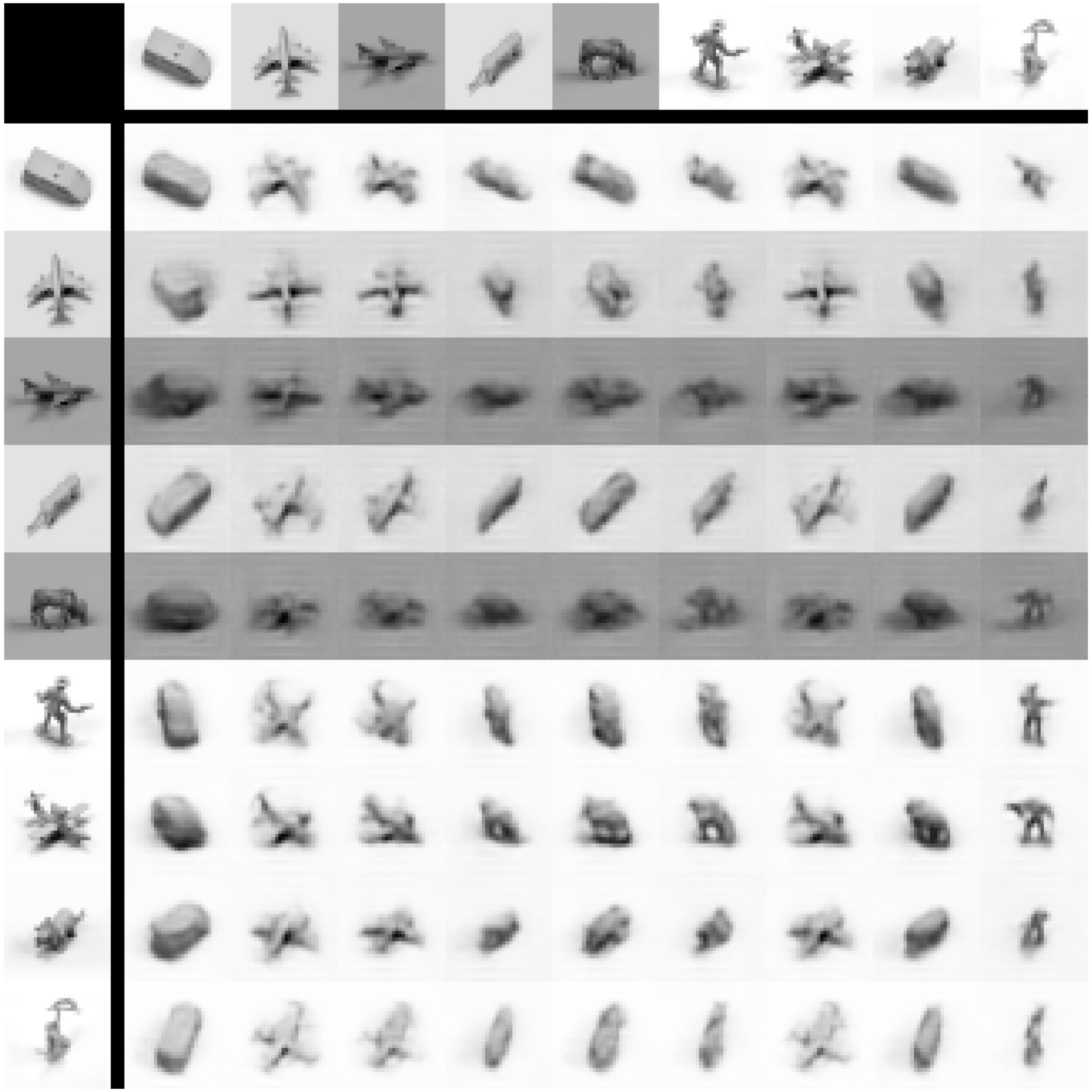
Style transfer: interpolation

- Interpolate between top left and bottom right characters
- Style changes vertically, identity changes horizontally.



Pose transfer results

- Transfer category from top row to orientation of left column



Pose transfer results

- Transfer category from top row to orientation of left column

Let's be inspired by nature, but not too much

- It's nice imitate Nature,
- But we also need to understand
  - How do we know which details are important?
  - Which details are merely the result of evolution, and the constraints of biochemistry?
- For airplanes, we developed aerodynamics and compressible fluid dynamics.
  - We figured that feathers and wing flapping weren't crucial
- QUESTION: What is the equivalent of aerodynamics for understanding intelligence?

L'Avion III de Clément Ader, 1897

(Musée du CNAM, Paris)

His "Eole" took off from the ground in 1890,

13 years before the Wright Brothers, but you

probably never heard of it (unless you are french).

What will future AI systems be like?

- **Human and animal behavior has basic "drives" hardwired by evolution**
  - Fight/flight, hunger, self-preservation, pain avoidance, desire for social interaction, etc…
- **Humans do bad things to each other because of these drives (mostly)**
  - Violence under threat, desire for material resource and social power…
- **But an AI system will not have these drives unless we build them into it.**
- **It's difficult for us humans to imagine an intelligent entity without these drives**
  - But they are specific to humans
  - We have plenty of different forms of intelligence in the animal world