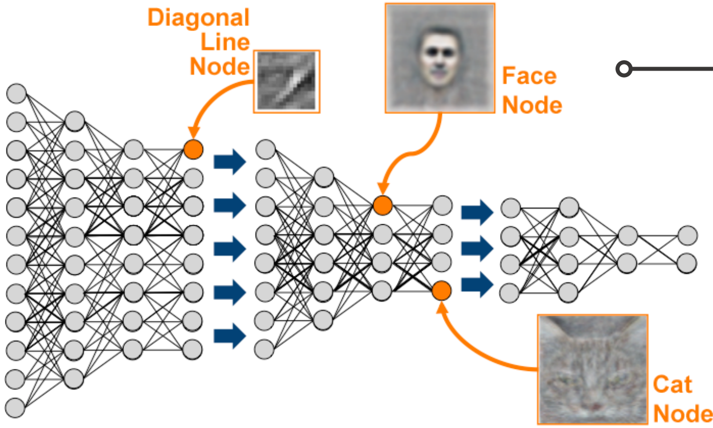
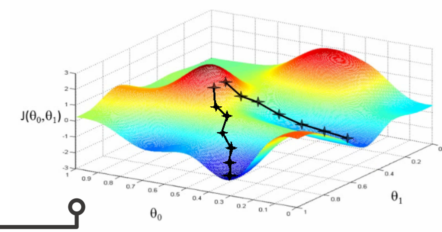
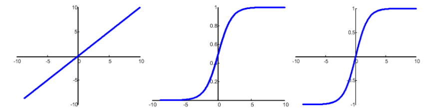


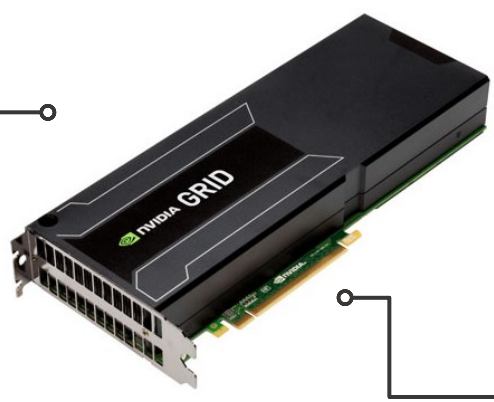


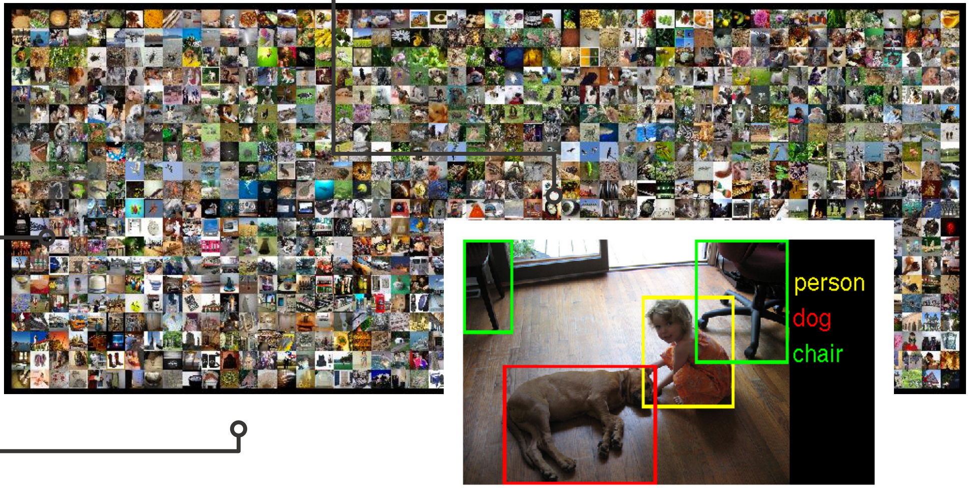


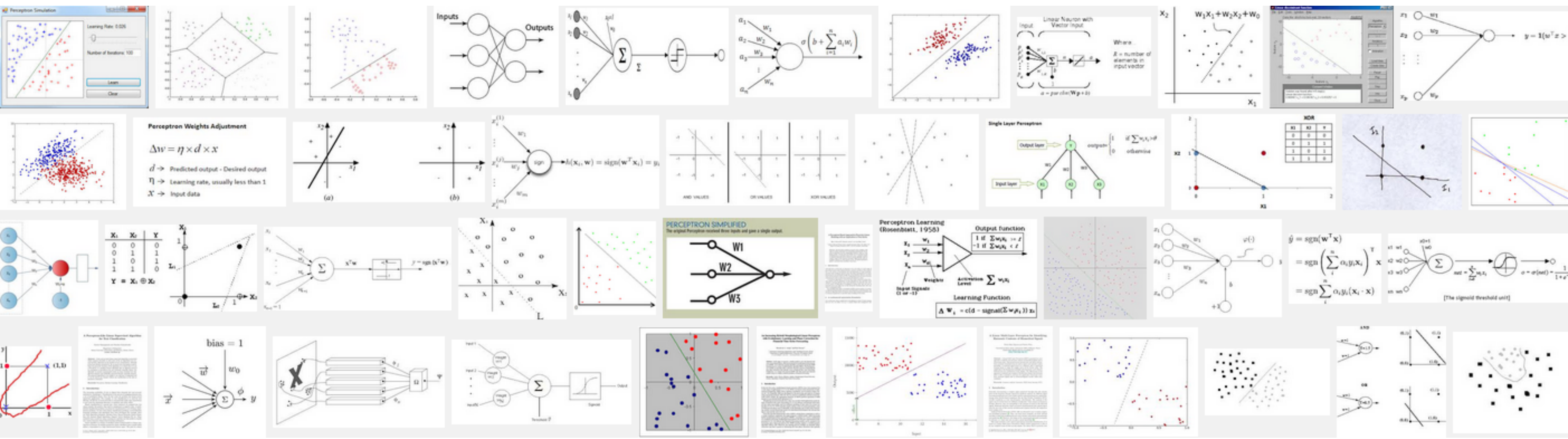
ECI 2014
ESCUELA DE CIENCIAS INFORMÁTICAS

Métodos Actuales de Machine Learning

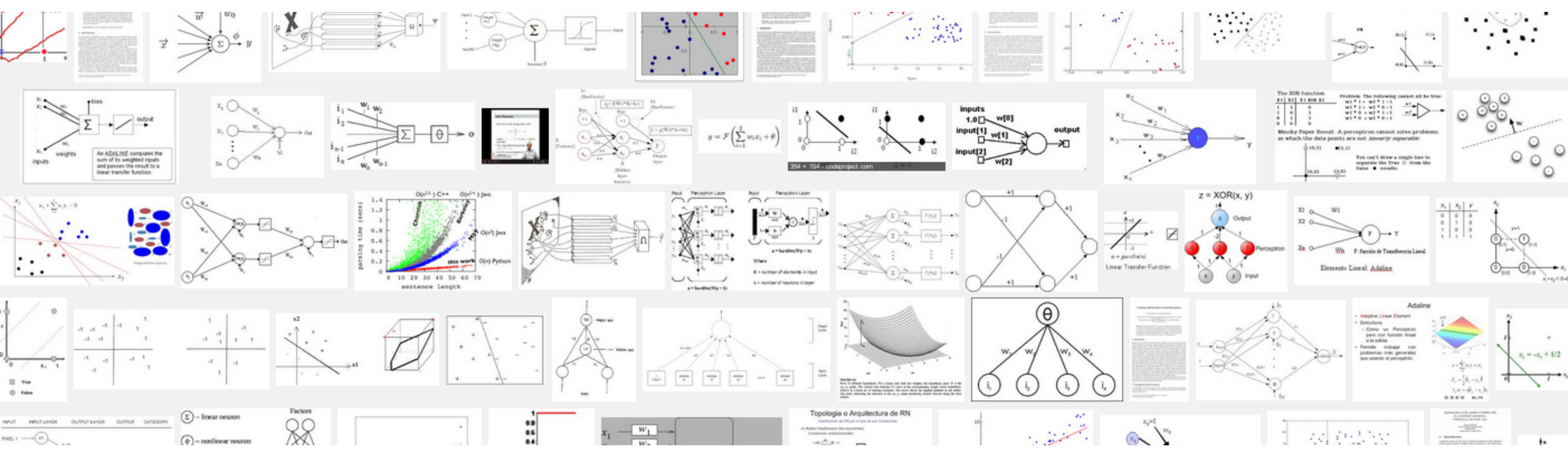
Neural Networks



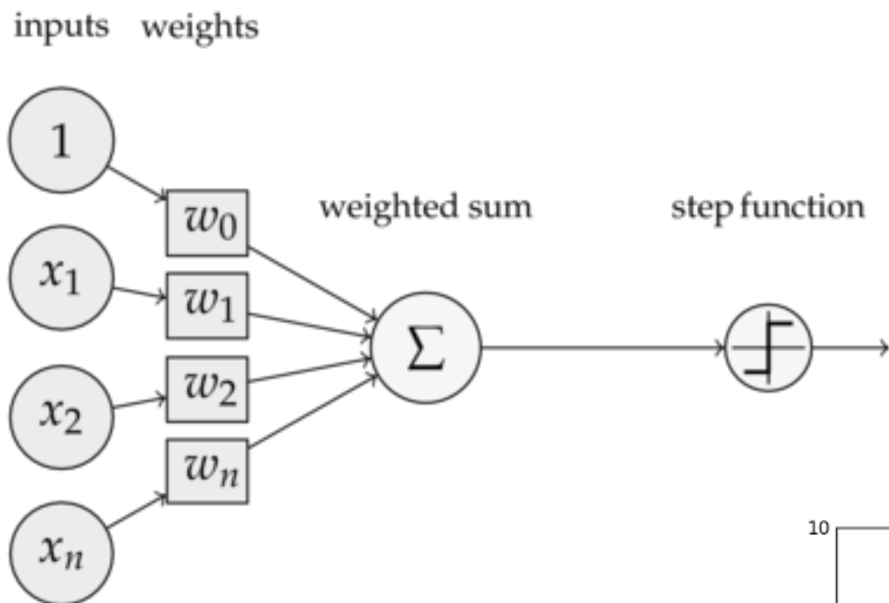




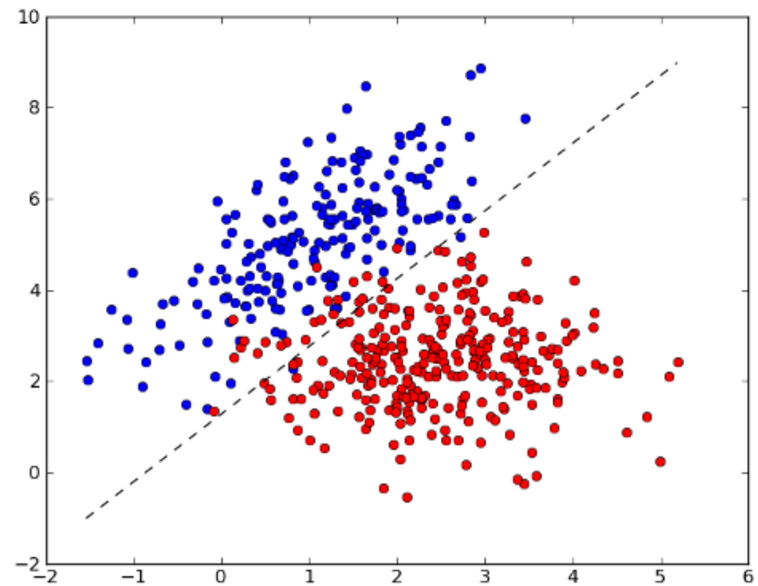
Linear Perceptron



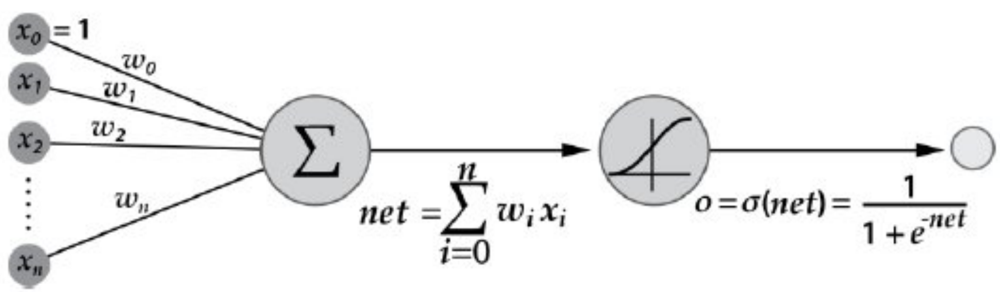
Linear Perceptron



1957



$$g_0(x) = \sigma(x^T W^{(0)} + b^{(0)})$$



Softmax Regression

$$\log \overbrace{p(\underset{\substack{\text{output}\\\text{(multiple categories)}}}{y}|\underset{\text{input}}{x})}^{\substack{\text{class probability}\\\text{vector}}} = \overbrace{\underbrace{x^T W + b}_{\text{affine transform.}} + \underbrace{c(x)}_{\text{normalization}}}^{\substack{\text{parametric}\\\text{model}}}$$

$$p(y|x) = \frac{\exp(x^T W + b)}{\sum_i \exp(x^T W + b)_i} = \mathrm{softmax}(x^T W + b)$$

Softmax Regression Training

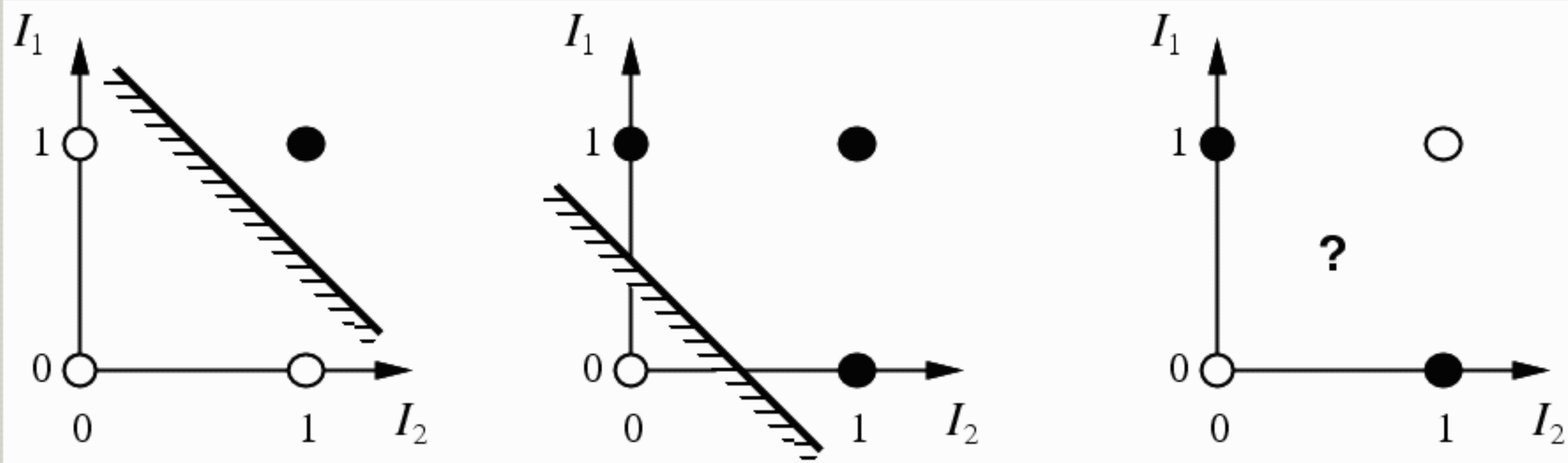
$$J(\mathcal{D}, W, b) = \prod_{x,y \in \mathcal{D}} p(y|x)$$ Maximum Likelihood Estimation (MLE)

training dataset

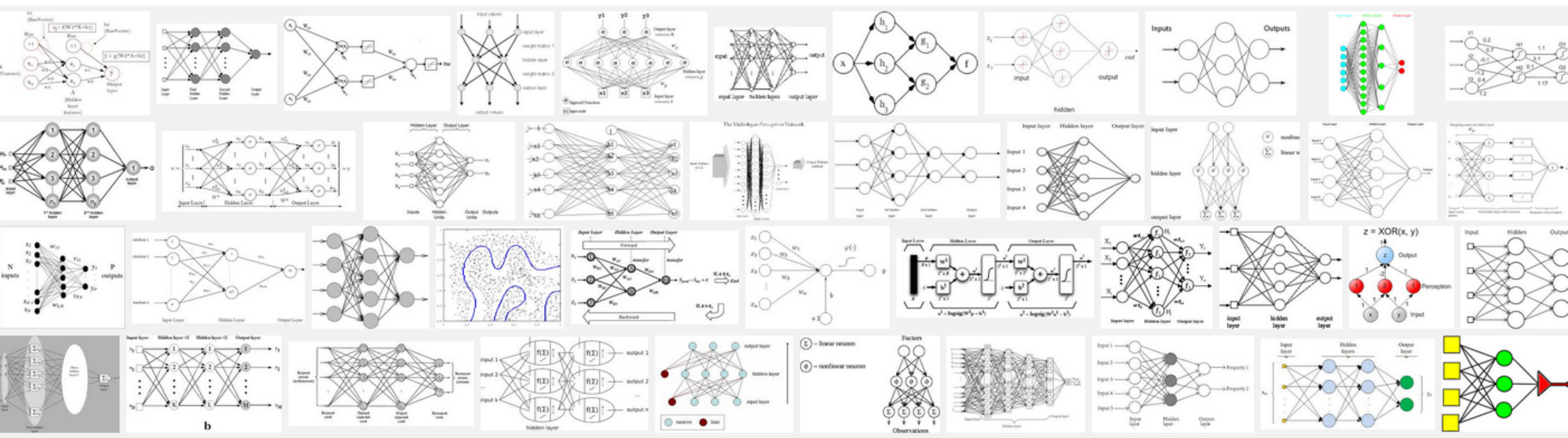
$$J(\mathcal{D}, W, b) = \sum_{x,y \in \mathcal{D}} \log p(y|x)$$ Log-Likelihood

$$NLL(\theta, \mathcal{D}) = -\sum_{i=0}^{|\mathcal{D}|} \log P(Y = y^{(i)}|x^{(i)}, \theta)$$ Negative Log-Likelihood

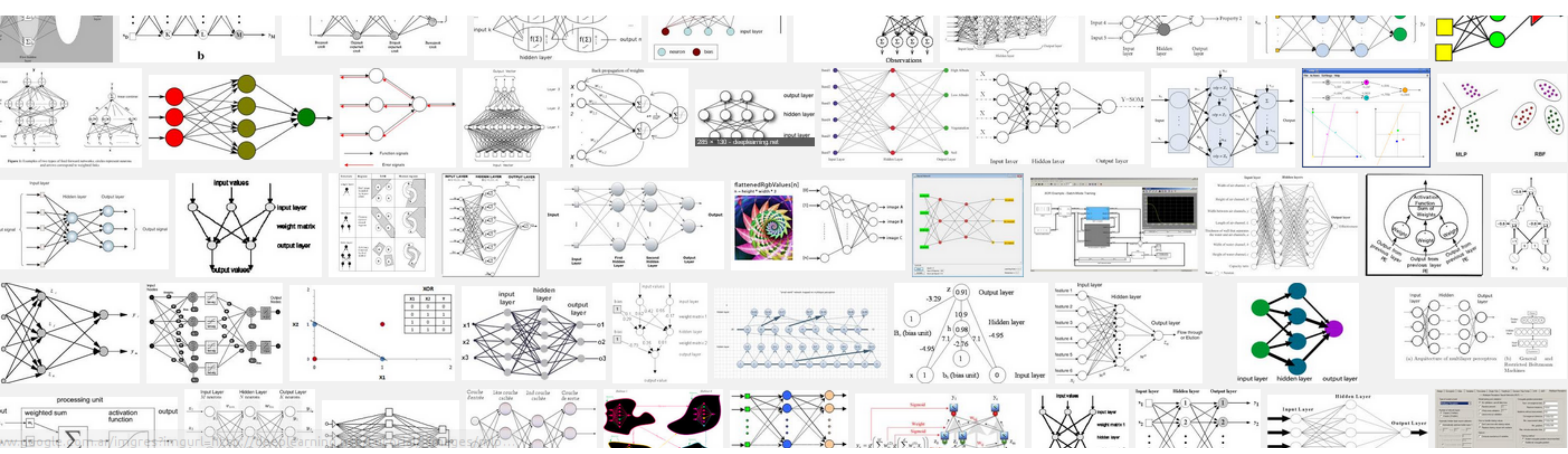
Linear Perceptron



(a) I_1 **and** I_2 (b) I_1 **or** I_2 (c) I_1 **xor** I_2



Multilayer Perceptrons (MLP)



Multilayer Perceptrons (MLP)

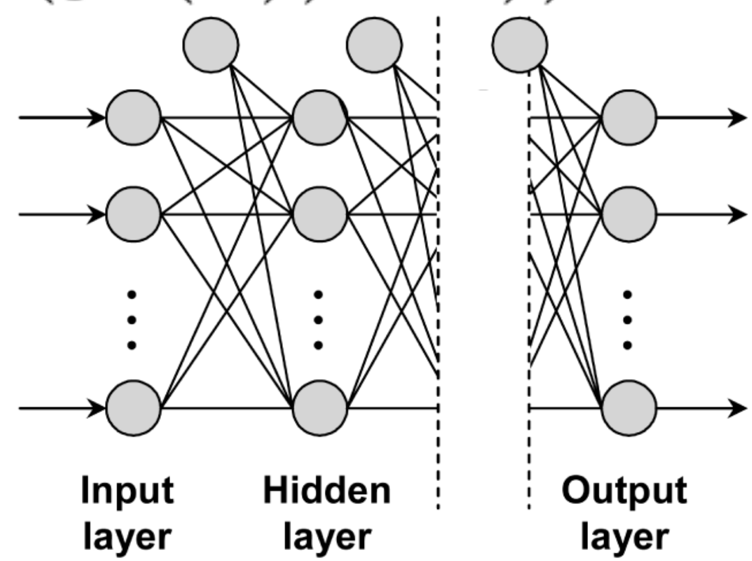
$$J(\mathcal{D}, W, b) = \sum_{x,y \in \mathcal{D}} \log p(y|x)$$ Log-Likelihood

$$\log p(y|x) = \cancel{x^T W + b + c(x)}$$ Single layer softmax

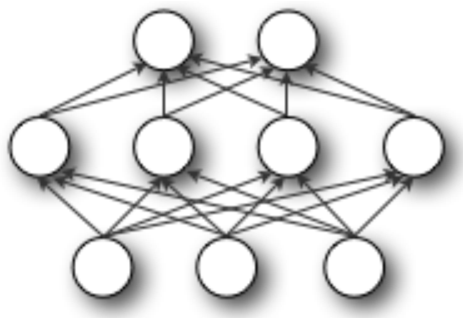
MLP output

$$\underbrace{f(x)}_{\text{MLP output}} = g_L(g_{L-1}(\ldots g_2(g_1(x))\ldots))$$

$$g_\ell(x) = \sigma(x^T W^{(\ell)} + b^{(\ell)})$$

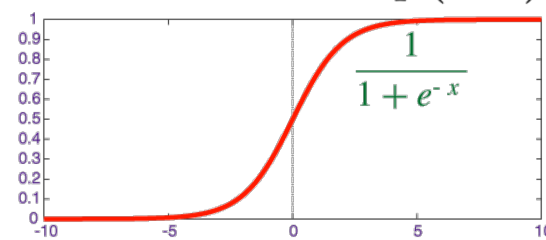


Multilayer Perceptrons (MLP)



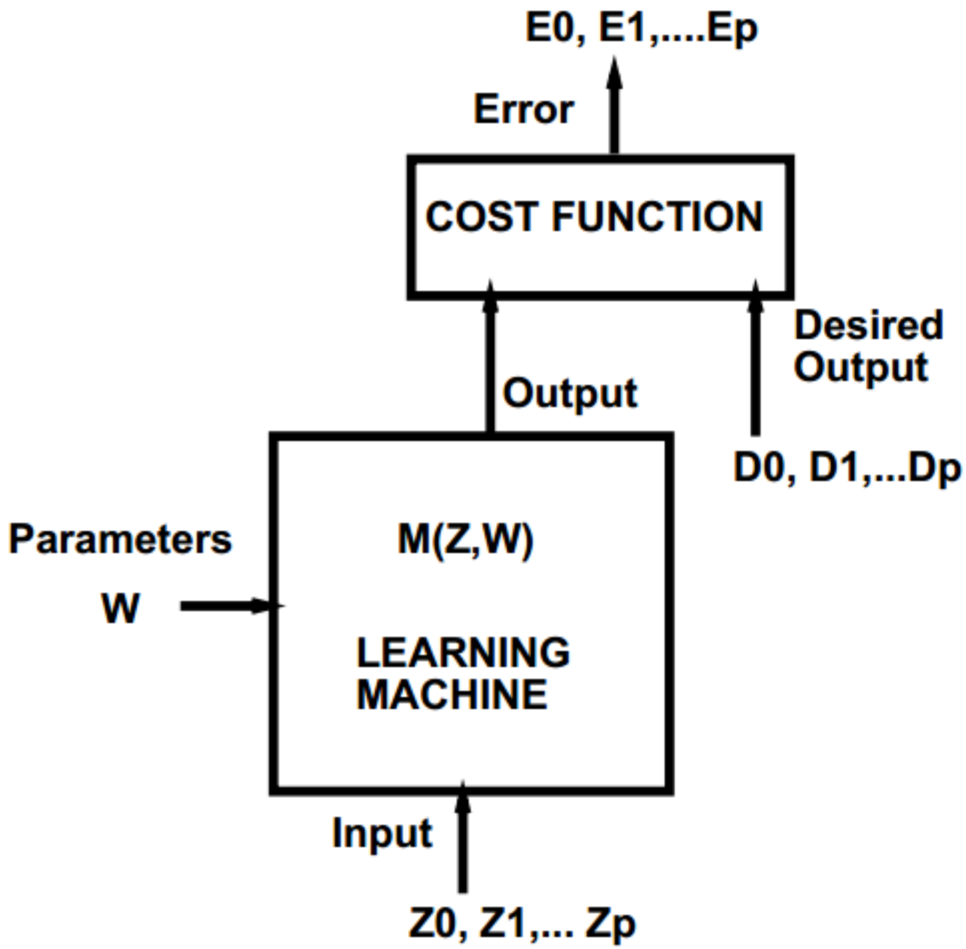
$$g_2(g_1) = \text{softmax}(g_1^T W^{(2)} + b^{(2)})$$

$$g_1(x) = \sigma(x^T W^{(1)} + b^{(1)}) \qquad \sigma(z) = \frac{1}{1+exp(-z)}$$



$$f(x) = \text{softmax}\left(\sigma(x^T W^{(1)} + b^{(1)})^T W^{(2)} + b^{(2)}\right)$$

Gradient Based Learning Machine



$$E^p = \frac{1}{2}(D^p - M(Z^p, W))^2$$

$$E_{train} = \frac{1}{P}\sum_{p=1} E^p$$

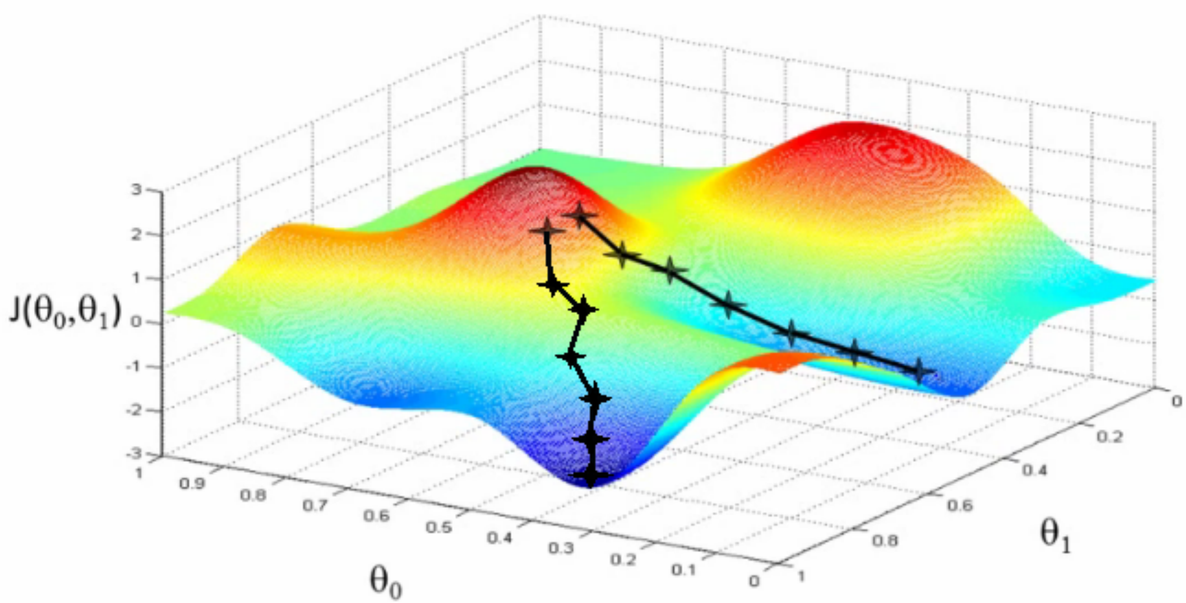
$$W_k = W_{k-1} - \epsilon \frac{\partial E(W)}{\partial W}$$

LeCun, Yann A et al. "Efficient backprop." *Neural networks: Tricks of the trade* (2012): 9-48.

Gradient Descent

Algorithm 1 GRADIENT DESCENT

while True **do**
  loss = f(params)
  d_loss_wrt_params = ... ▷ compute gradient
  params -= learning_rate * d_loss_wrt_params
 if stopping condition is met **then return** params
 end if
end while



Stochastic Gradient Descent

Algorithm 1 GRADIENT DESCENT

while True **do**
  loss = f(params)
  d_loss_wrt_params = ... ▷ compute gradient
  params -= learning_rate * d_loss_wrt_params
  if stopping condition is met **then return** params
  end if
end while

Algorithm 2 STOCHASTIC GRADIENT DESCENT

for $(x_i, y_i) \in \mathcal{D}_{train}$ **do**
  ▷ imagine an infinite generator that may repeat
  ▷ examples (if there is only a finite training set)
  loss = f(params, x_i, y_i)
  d_loss_wrt_params = ... ▷ compute gradient
  params $-$ = learning_rate * d_loss_wrt_params
  if stopping condition is met **then return** params
  end if
end for

Mini-batch Gradient Descent

Algorithm 2 STOCHASTIC GRADIENT DESCENT

```
for (x_i, y_i) ∈ D_train do
                                    ▷ imagine an infinite generator that may repeat
                                    ▷ examples (if there is only a finite training set)
    loss = f(params, x_i, y_i)
    d_loss_wrt_params = ...                                  ▷ compute gradient
    params − = learning_rate * d_loss_wrt_params
    if stopping condition is met then return params
    end if
end for
```

Algorithm 3 MINIBATCH SGD

```
for (x_batch,y_batch) ∈ train_batches do
                                               ▷ imagine an infinite generator
                                               ▷ that may repeat examples
    loss = f(params, x_batch, y_batch)
    d_loss_wrt_params = ...                                  ▷ compute gradient
    params − = learning_rate * d_loss_wrt_params
    if stopping condition is met then return params
    end if
end for
```

Hyperparameters: Learning Rate

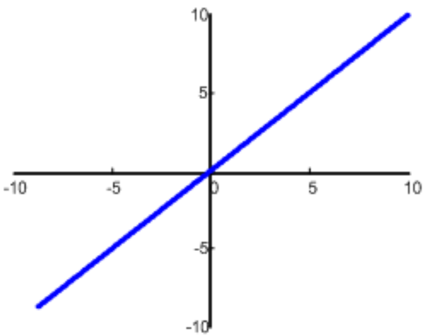
$$W_i \leftarrow W_i - \underbrace{\eta}_{\text{learning rate}} \frac{\partial \mathrm{E}(W_i)}{\partial W_i}$$

- constant learning rate (simplest solution)
- logarithmic grid search (10^{-1}, 10^{-2}, ...)
- decreasing learning rate over time:

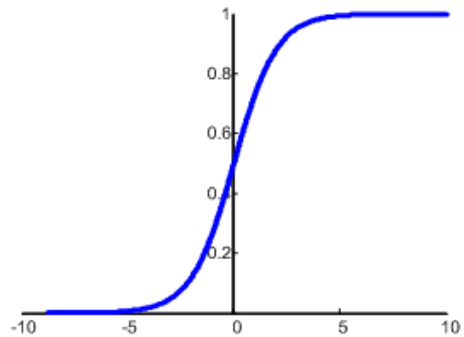
$$\eta_t = \frac{\eta_0}{1 + at}$$

For adaptive learning rate see: LeCun, Yann A et al. "Efficient backprop." *Neural networks: Tricks of the trade* (2012): 9-48.

Hyperparameters: Nonlinearity

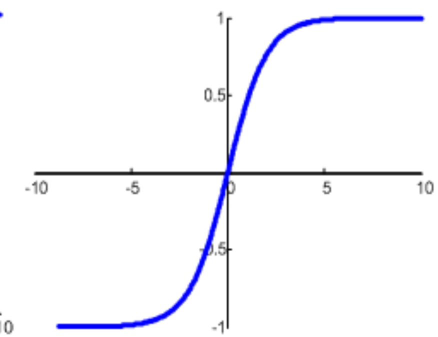


linear



logistic

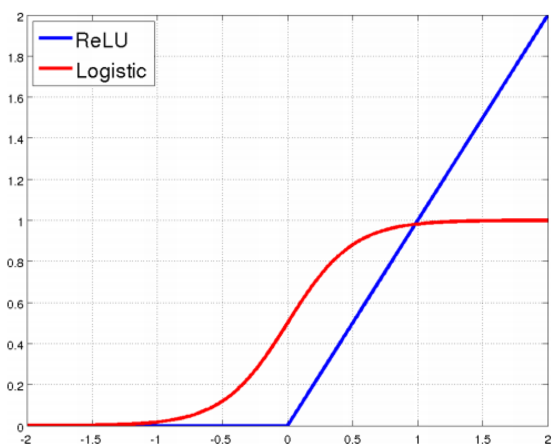
$$f(u) = 1/(1 + \exp(-u))$$



tanh

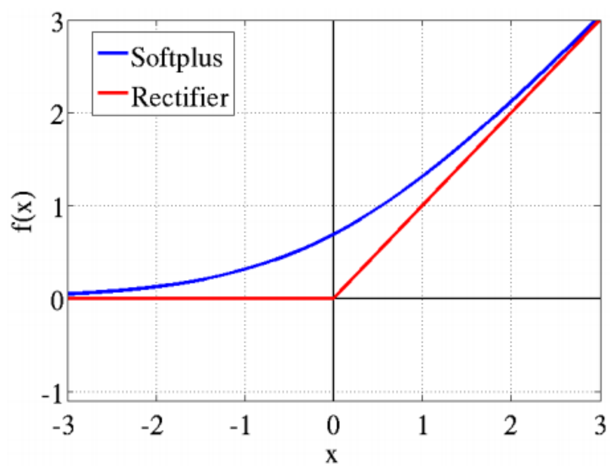
ReLU

$$f(u) = \max(0, u)$$

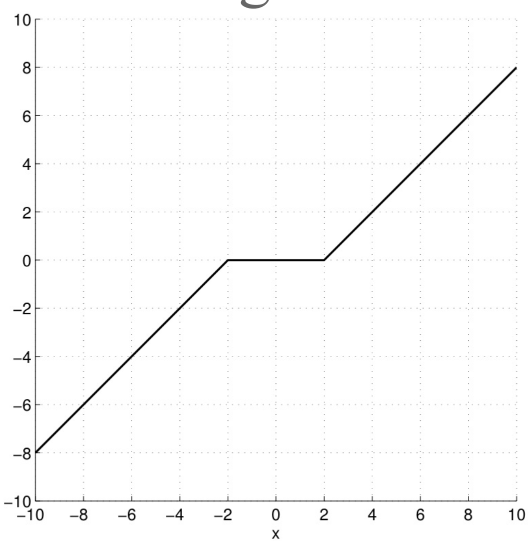


Softplus

$$softplus(x) = log(1 + e^x)$$



Shrinkage



Hyperparameters: Weight Initialization

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

for tanh activations

$$W \sim U\left[-\frac{4\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{4\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

for logistic activations

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *International Conference on Artificial Intelligence and Statistics* 2010: 249-256.

Hyperparameters: momentum

$$\Delta\theta_i(t) = v_i(t) = \alpha v_i(t-1) - \epsilon \frac{dE}{d\theta_i}(t)$$

momentum

learning rate

Hinton, Geoffrey E. "A practical guide to training restricted boltzmann machines." *Neural Networks: Tricks of the Trade* (2012): 599-619.

Regularization

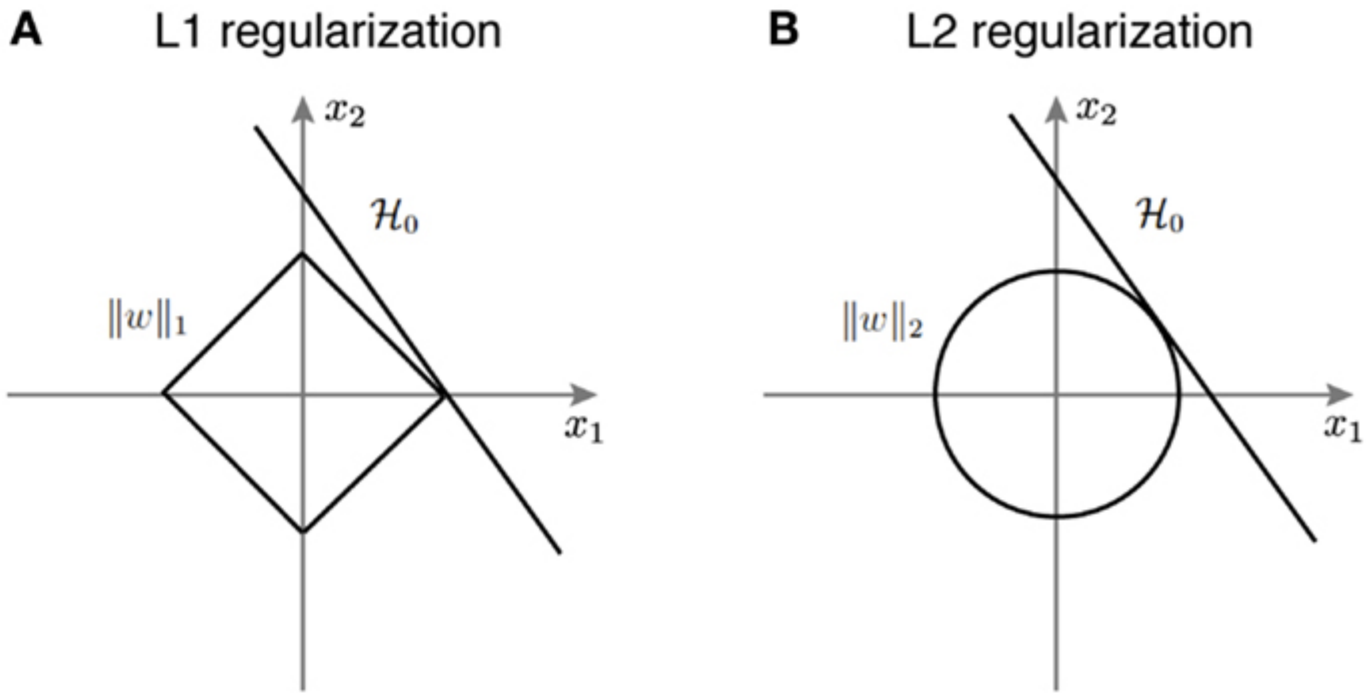
$$E(\theta, \mathcal{D}) = \underbrace{NLL(\theta, \mathcal{D})}_{\text{cost function}} + \underbrace{\lambda \|\theta\|_p^p}_{\text{regularization term}}$$

p-norm of θ $$\|\theta\|_p = \left(\sum_{j=0} |\theta_j|^p \right)^{\frac{1}{p}}$$

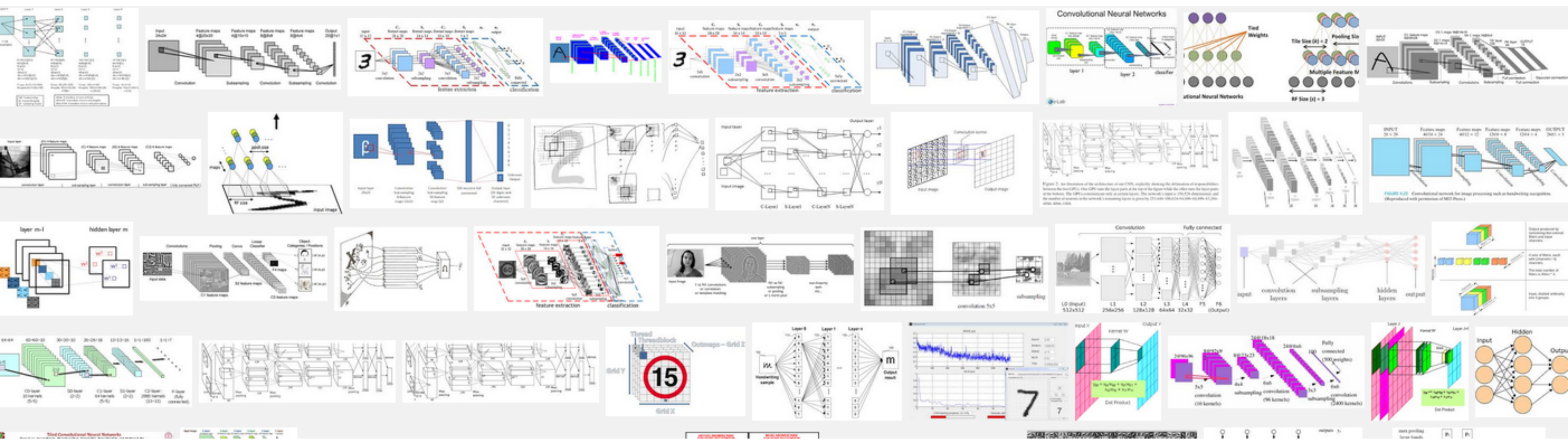
L1 vs L2 regularization

$$\|w\|_1 = \sum_j |w_j|$$

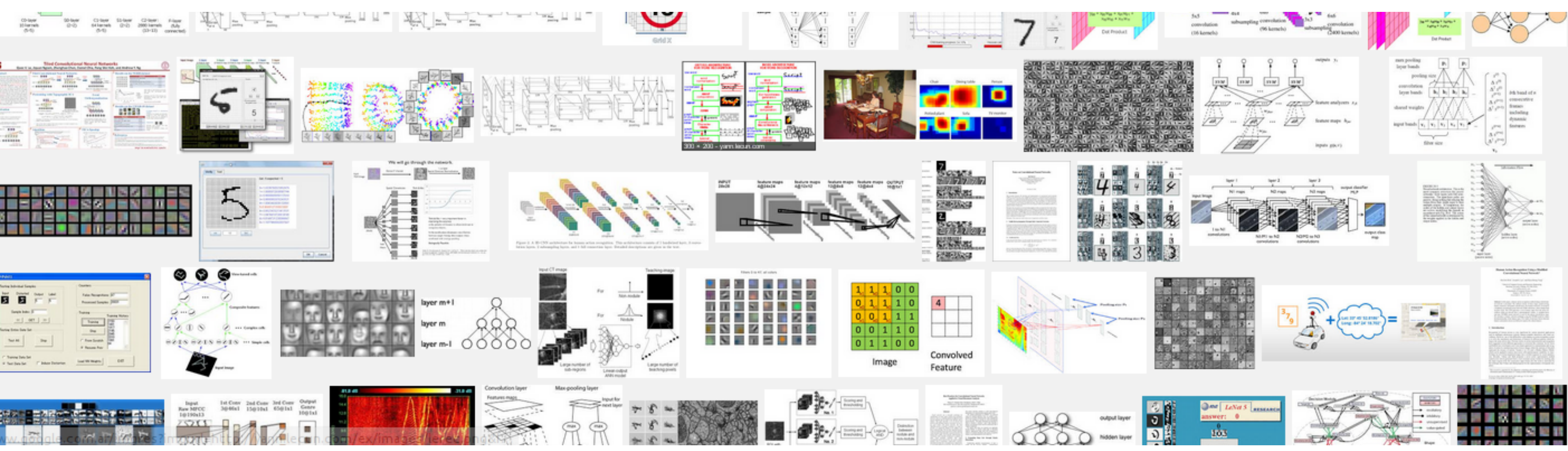
$$\|w\|_2 = \sqrt{\sum_j |w_j|^2}$$



Shi, Jianing V et al. "Perceptual decision making “Through the Eyes” of a large-scale neural model of V1." *Frontiers in psychology* 4 (2013).

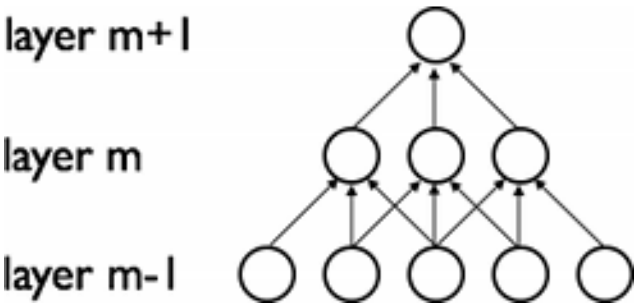


Convolutional Neural Networks (CNN)



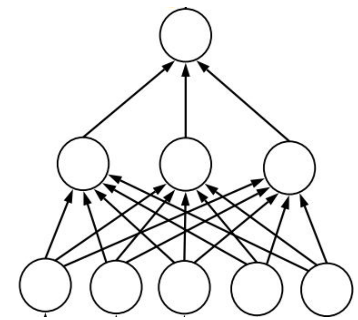
Convolutional Neural Networks

Sparse connectivity

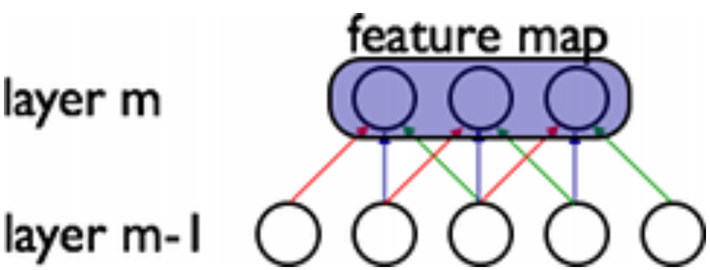


versus

Dense connectivity



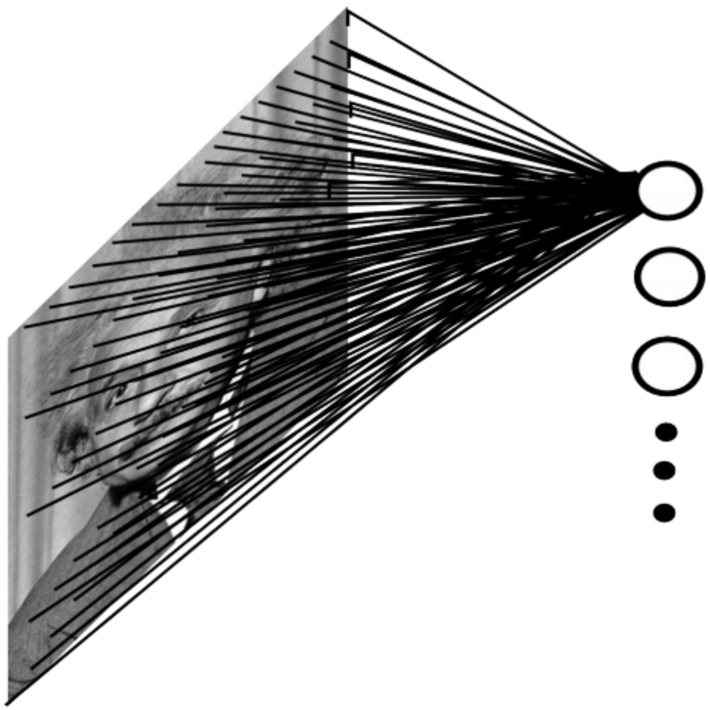
Shared weights



Convolutional Neural Networks

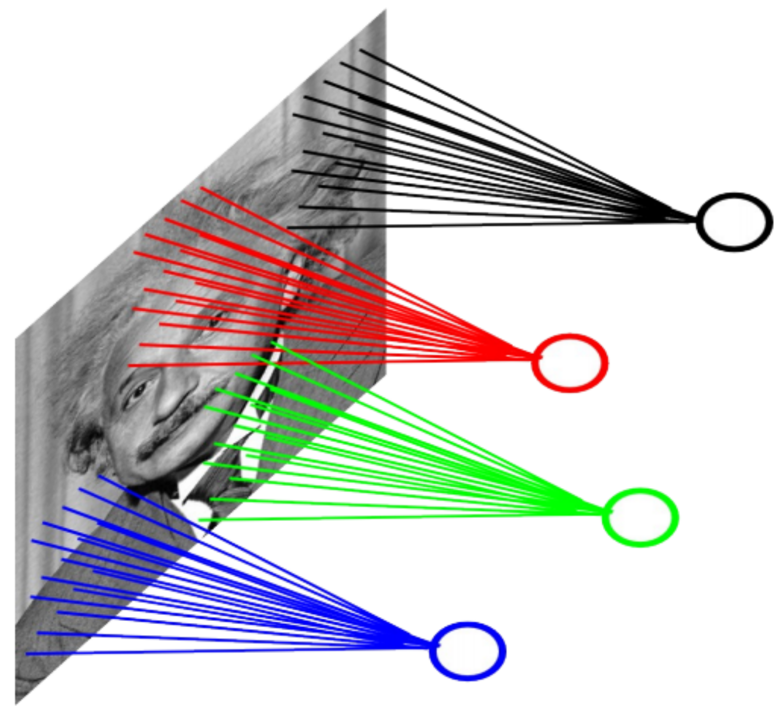
dot product + bias

$$h_k = \tanh(W_k^T x + b_k)$$

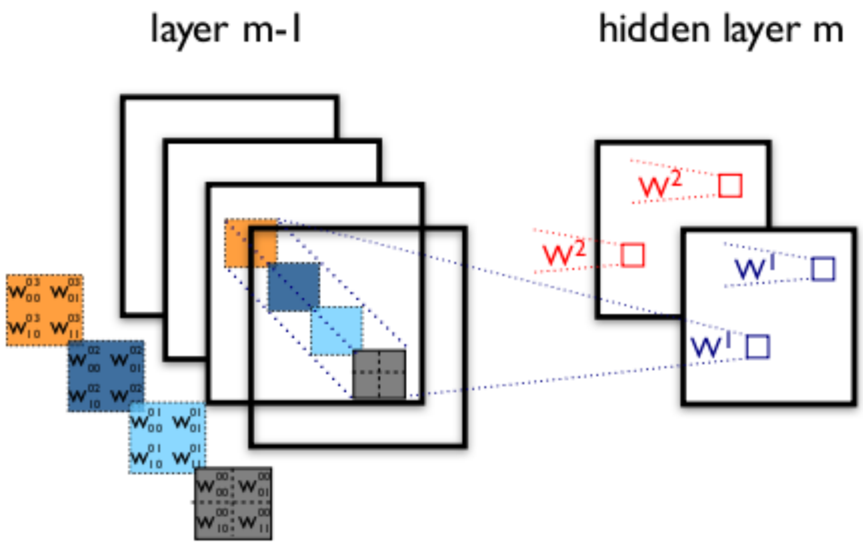


convolution + bias

$$h_{ij}^k = \tanh((W^k * x)_{ij} + b_k)$$



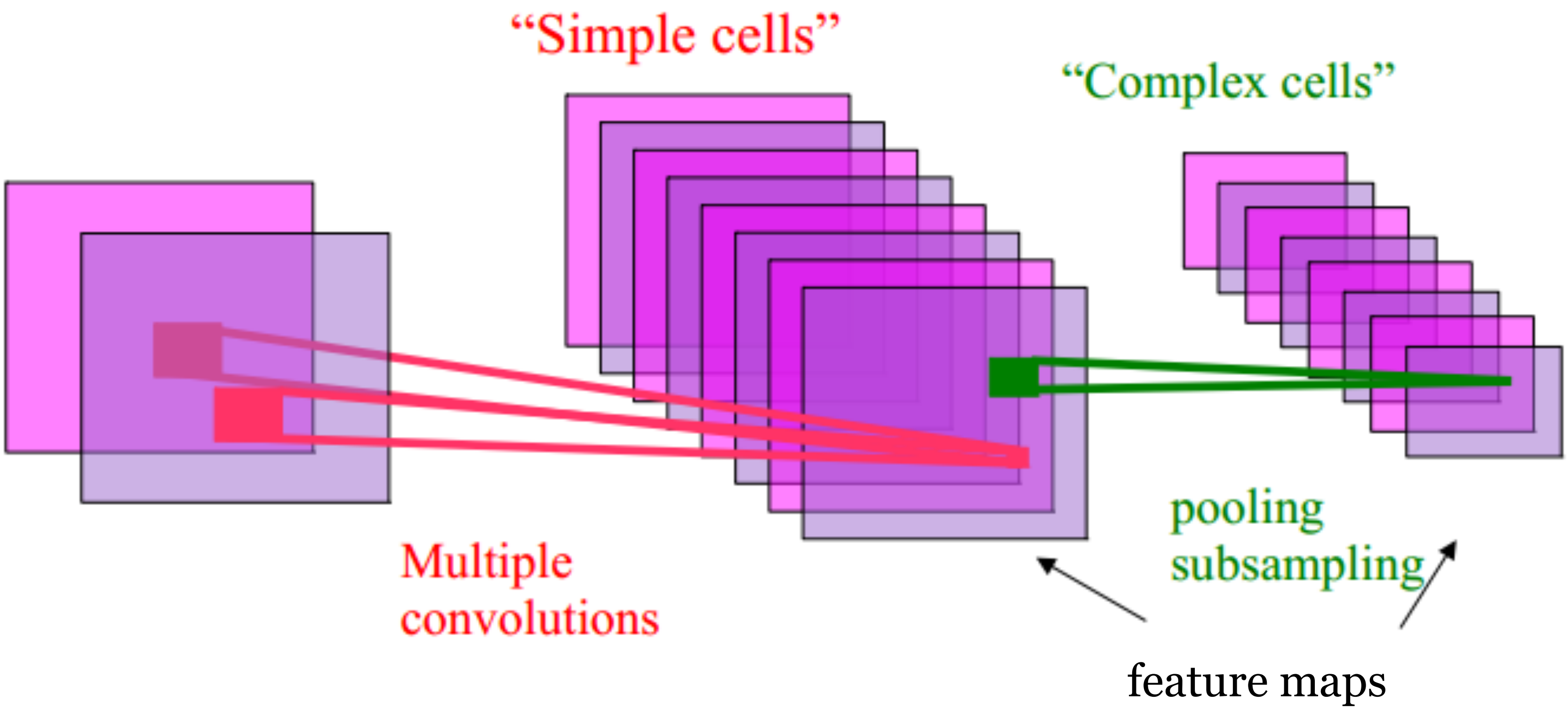
Convolutional Neural Networks



- Detects multiple motifs at each location
- The collection of units looking at the same patch is akin to a feature vector for that patch.
- The result is a 3D array, where each slice is a feature map.

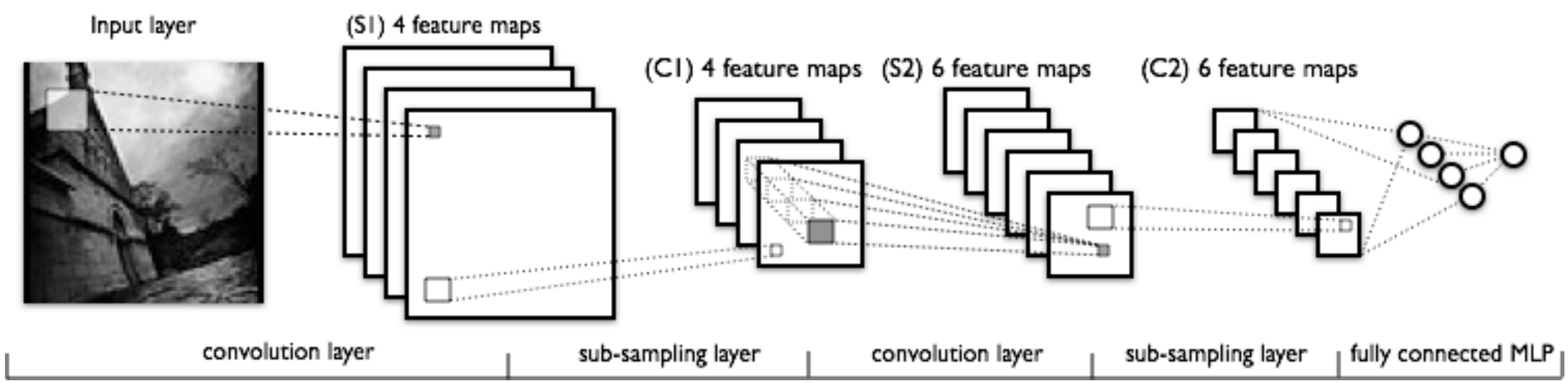
Convolutional Neural Networks

Pooling subsampling

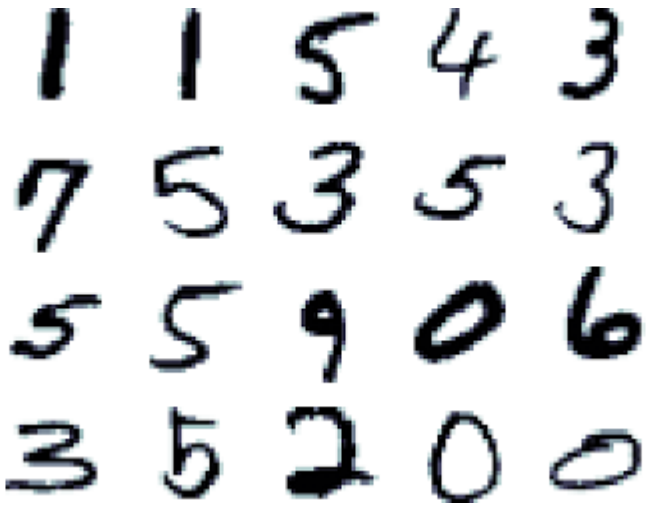


Convolutional Neural Networks

- **Are deployed in many practical applications**
  - Image recognition, speech recognition, Google's and Baidu's photo taggers
- **Have won several competitions**
  - ImageNet, Kaggle Facial Expression, Kaggle Multimodal Learning, German Traffic Signs, Connectomics, Handwriting....
- **Are applicable to array data where nearby values are correlated**
  - Images, sound, time-frequency representations, video, volumetric images, RGB-Depth images,.....
- **One of the few deep models that can be trained purely supervised**

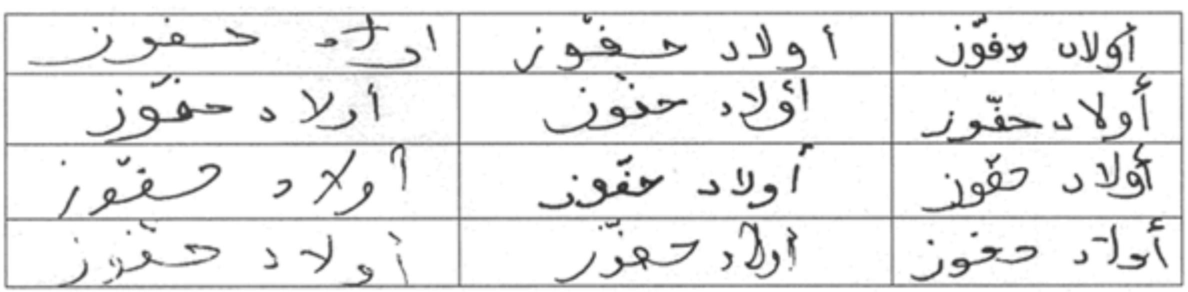


Tasks for Which Deep CNN are the Best



Handwriting recognition MNIST

Arabic Handwriting Recognition



Margner, Volker, and Haikal El Abed. "Arabic handwriting recognition competition." *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on* 23 Sep. 2007: 1274-1278.

Tasks for Which Deep CNN are the Best

StreetView House Numbers [2011]

94.3 % accuracy

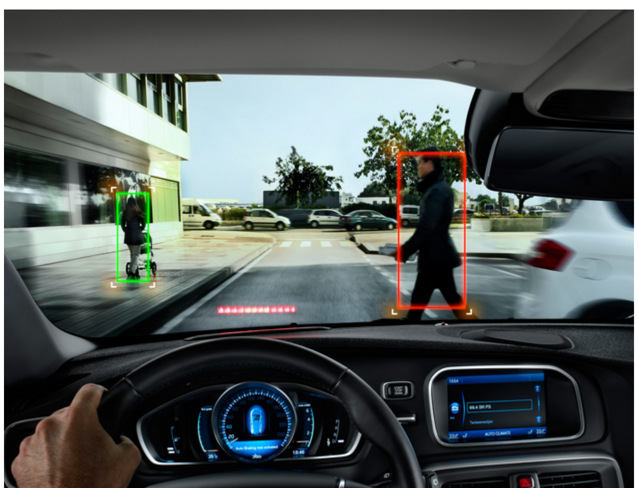
Netzer, Yuval et al. "Reading digits in natural images with unsupervised feature learning." *NIPS workshop on deep learning and unsupervised feature learning* 2011: 4.

Tasks for Which Deep CNN are the Best

Traffic Sign Contest, Silicon Valley, 2011 (IDSIA)

0.56% ERROR

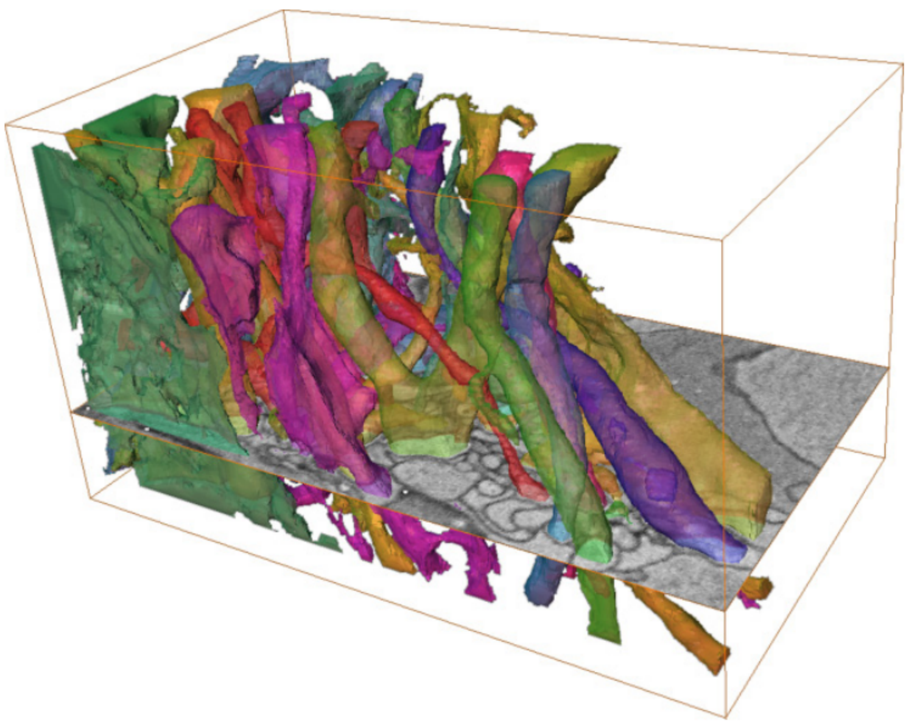
- first place
- twice better than humans
- three times better than the closest artificial competitor
- six times better than the best non-neural method



Pedestrian Detection [2013]: INRIA datasets and others (NYU)

Tasks for Which Deep CNN are the Best

Volumetric brain image segmentation [2009] Connectomics (IDSIA, MIT)



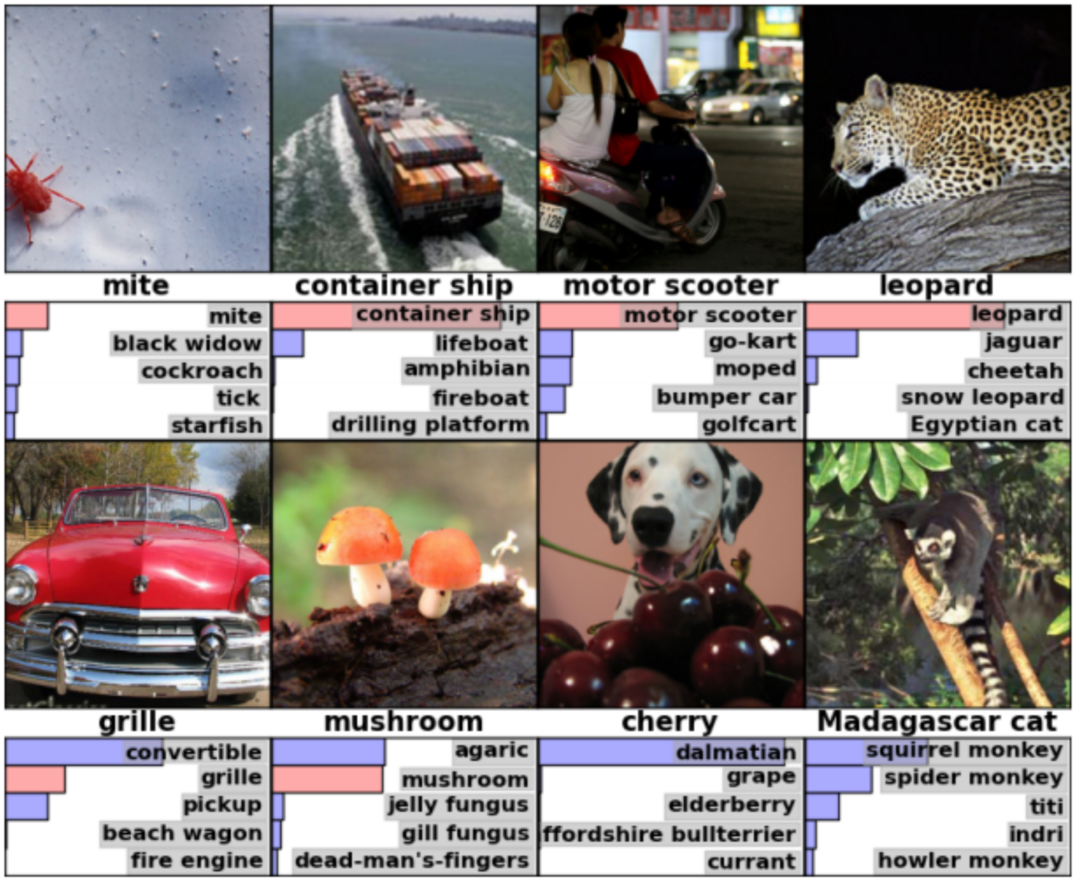
Turaga, Srinivas C et al. "Convolutional networks can learn to generate affinity graphs for image segmentation." *Neural Computation* 22.2 (2010): 511-538.

Human Action Recognition [2011] Hollywood II dataset (Stanford)

Le, Quoc V et al. "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis." *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* 20 Jun. 2011: 3361-3368.

Tasks for Which Deep CNN are the Best

Object Recognition [2012] ImageNet competition

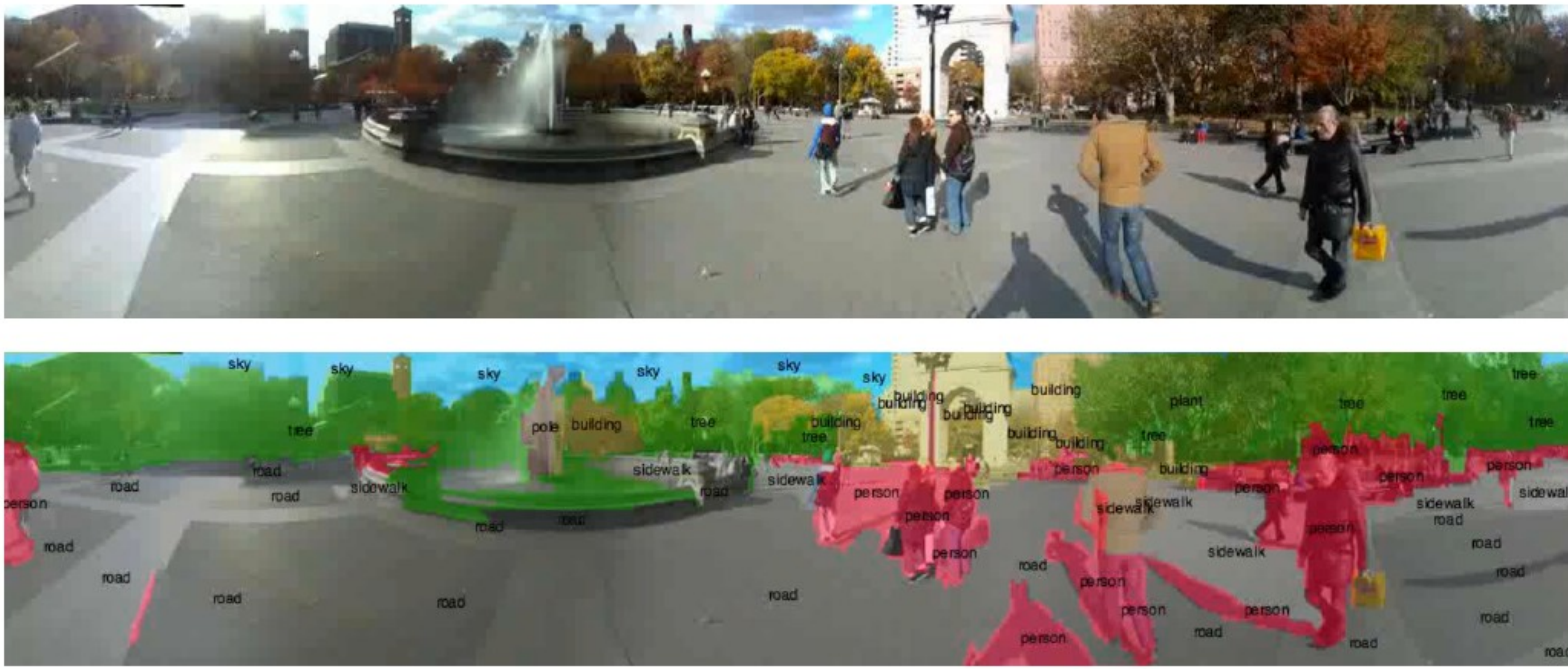


Error rate: 15% (whenever correct class isn't in top 5)
Previous state of the art: 25% error

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 2012: 1097-1105.

Tasks for Which Deep CNN are the Best

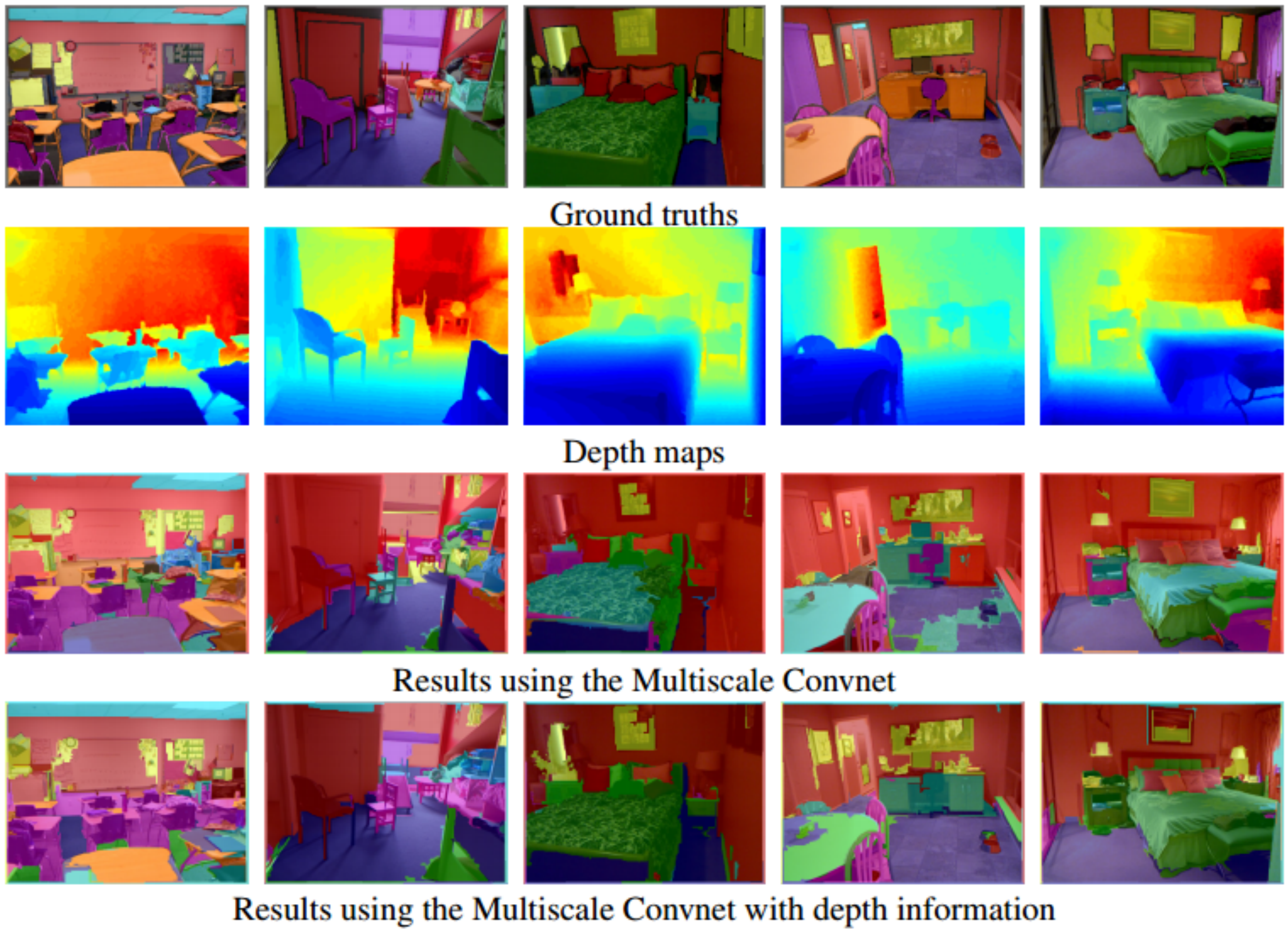
Scene Parsing [2012]



Farabet, Clément et al. "Scene parsing with multiscale feature learning, purity trees, and optimal covers." *arXiv preprint arXiv:1202.2160* (2012).

Tasks for Which Deep CNN are the Best

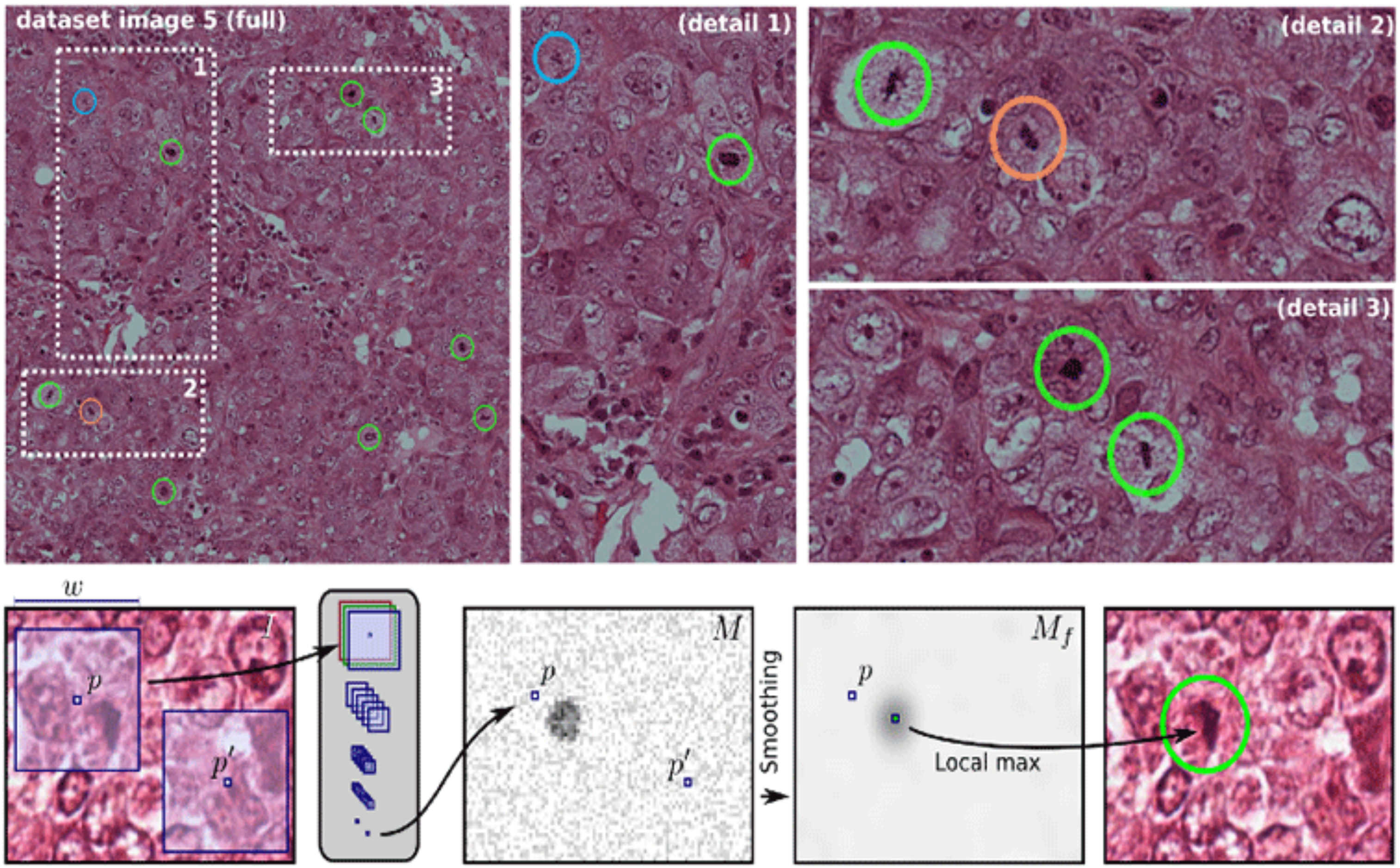
Scene Parsing from depth images [2013]



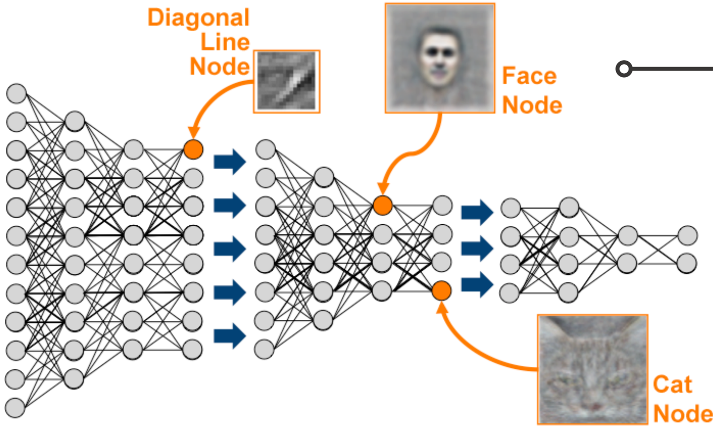
Couprie, Camille et al. "Indoor semantic segmentation using depth information." *arXiv preprint arXiv:1301.3572* (2013).

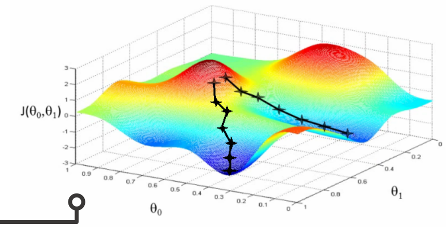
Tasks for Which Deep CNN are the Best

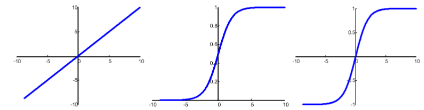
Breast cancer cell mitosis detection [2011] MITOS (IDSIA)



Cireşan, Dan C et al. "Mitosis detection in breast cancer histology images with deep neural networks." *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013* (2013): 411-418.





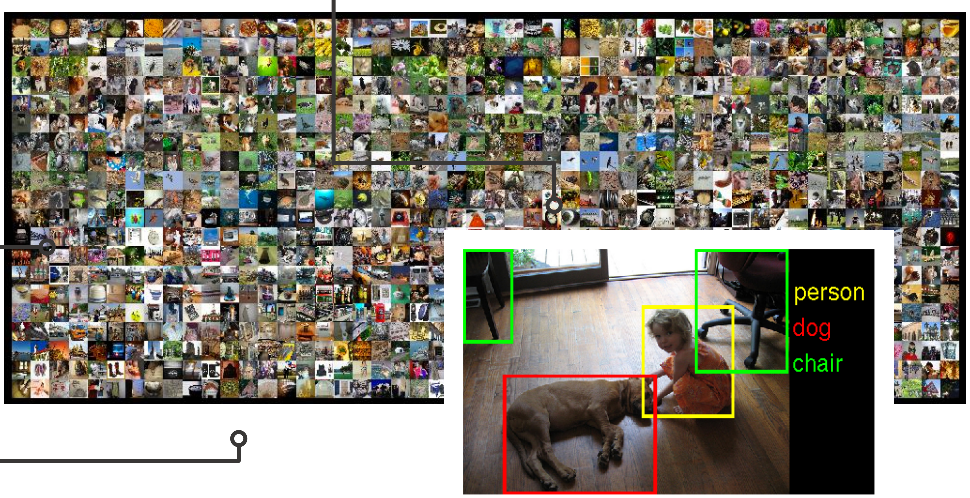


ECI 2014 ESCUELA DE CIENCIAS INFORMÁTICAS

Métodos Actuales de Machine Learning

Neural Networks

http://www.cifasis-conicet.gov.ar/granitto/ECI2014/

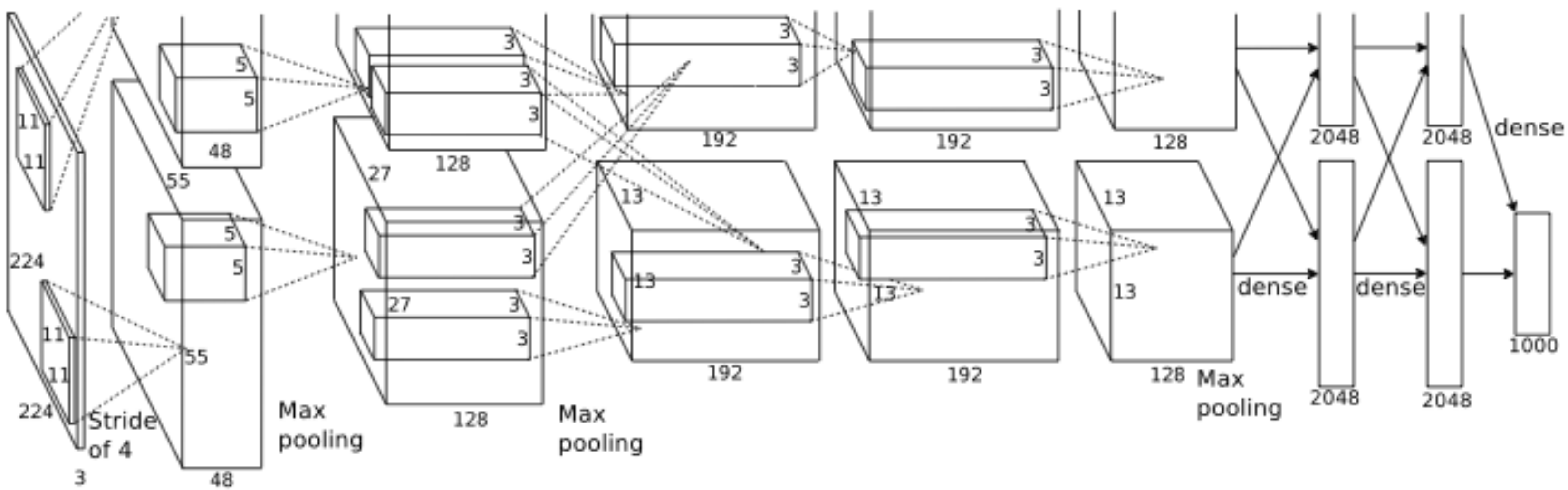


ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



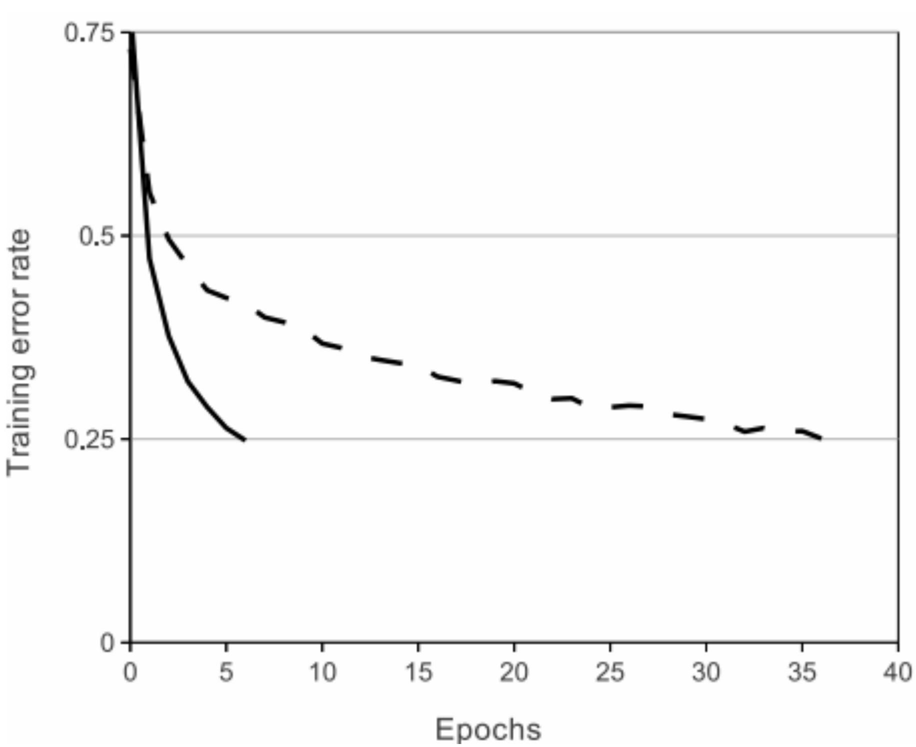
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 2012: 1097-1105.

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

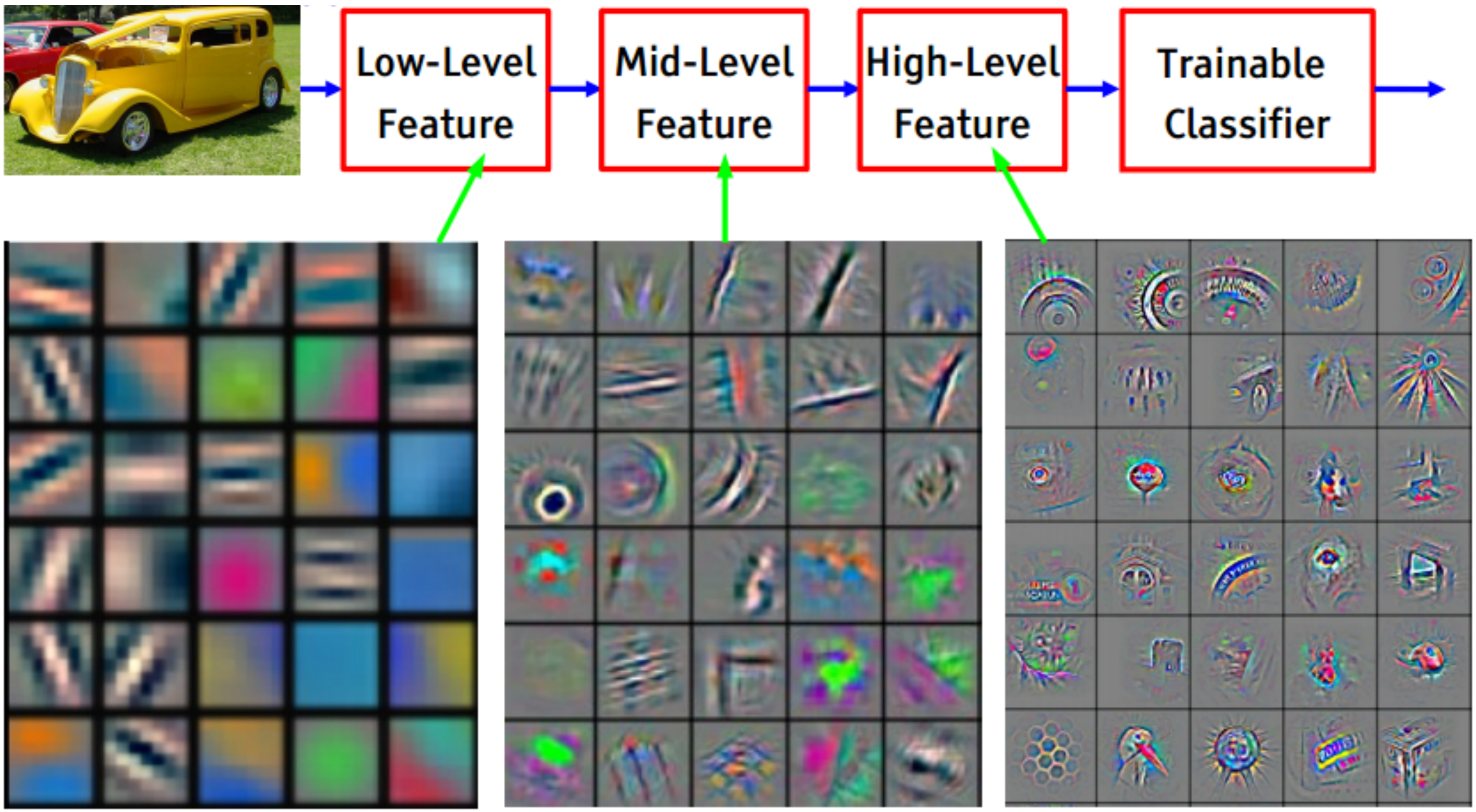
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



A four-layer convolutional neural network with **ReLUs (solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with **tanh neurons (dashed line)**.

Networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.



Zeiler, Matthew D, and Rob Fergus. "Visualizing and understanding convolutional neural networks." *arXiv preprint arXiv:1311.2901* (2013).

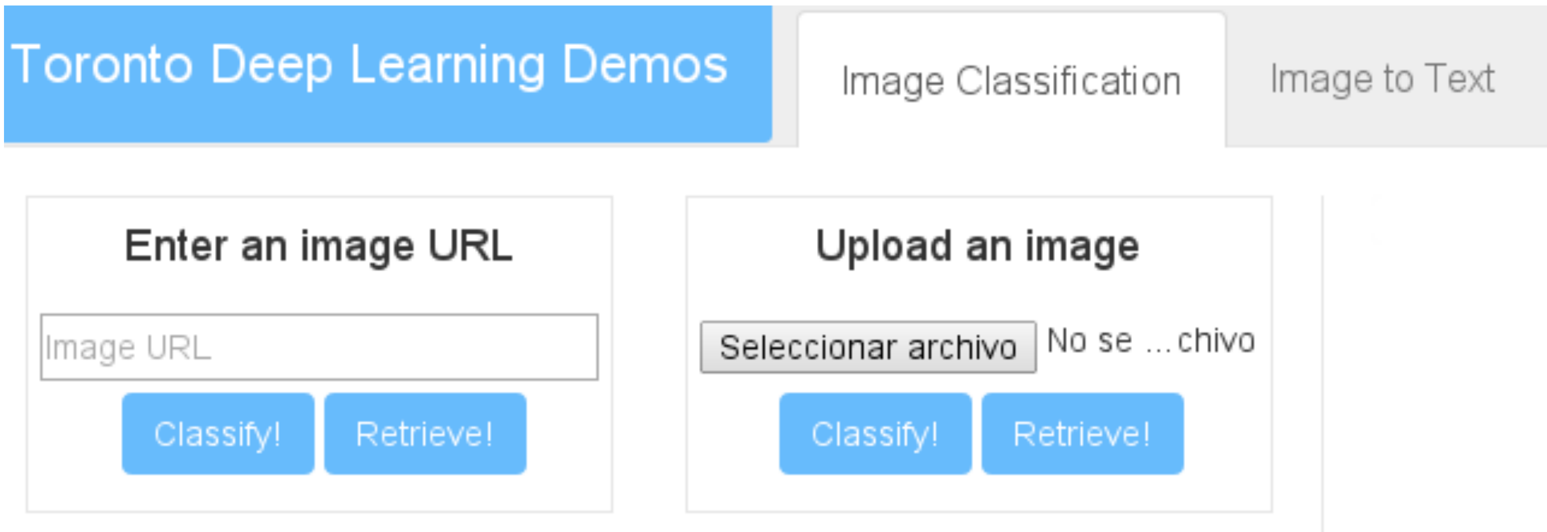
OverFeat: Object Recognizer, Feature Extractor

URL: http://cilvr.nyu.edu/doku.php?id=software:overfeat:start

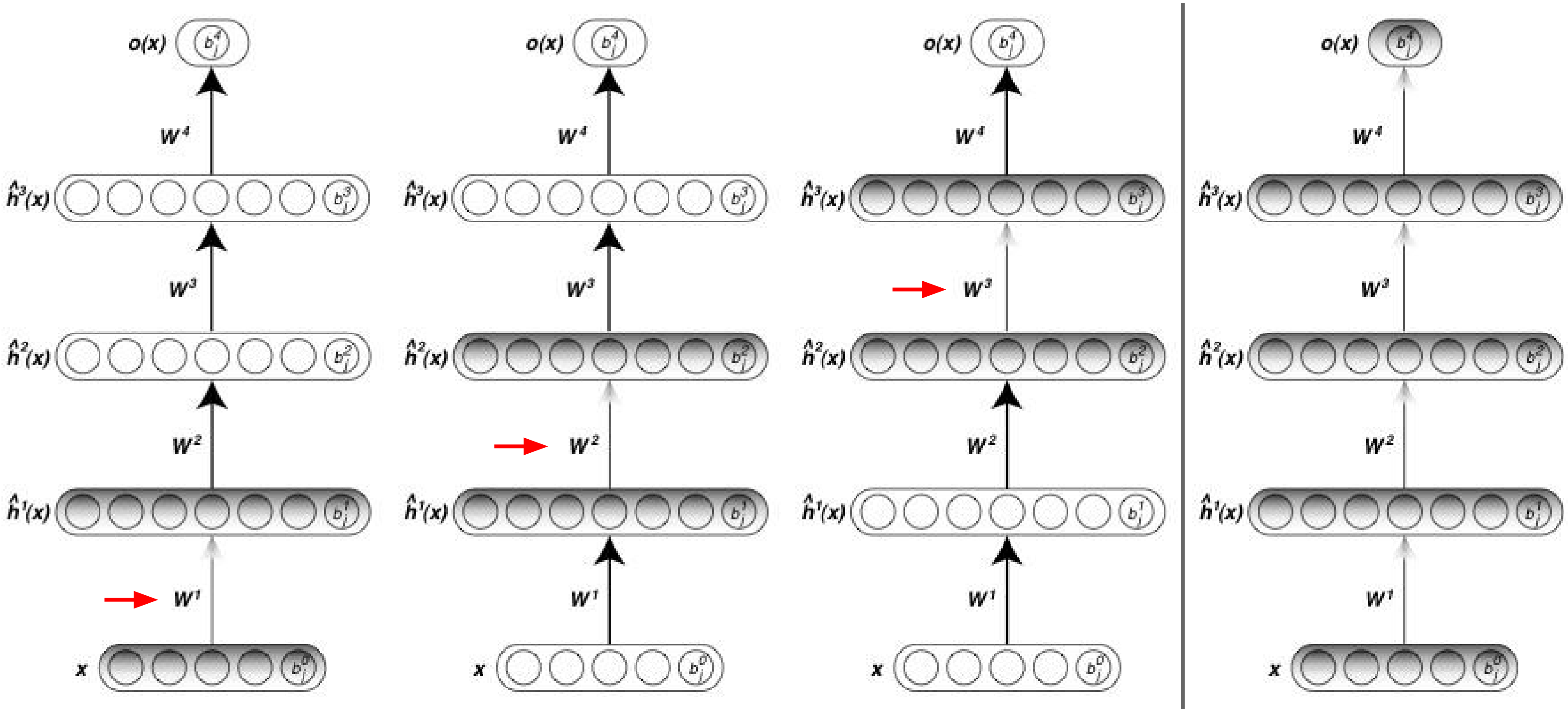
- OverFeat was trained on the ImageNet dataset and participated in the **ImageNet 2013 competition**.
- This package allows researchers to use OverFeat to **recognize images and extract features**.
- A library with C++ source code is provided for running the OverFeat convolutional network, together with wrappers in various scripting languages (Python, Lua, Matlab coming soon).
- OverFeat was trained with the Torch7 package (http://www.torch.ch). This package provides tools to run the network in a standalone fashion. The training code is not part of this package.

Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun: "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks", International Conference on Learning Representations (ICLR 2014), April 2014. (OpenReview.net), (arXiv:1312.6229), (BibTeX).

http://deeplearning.cs.toronto.edu/



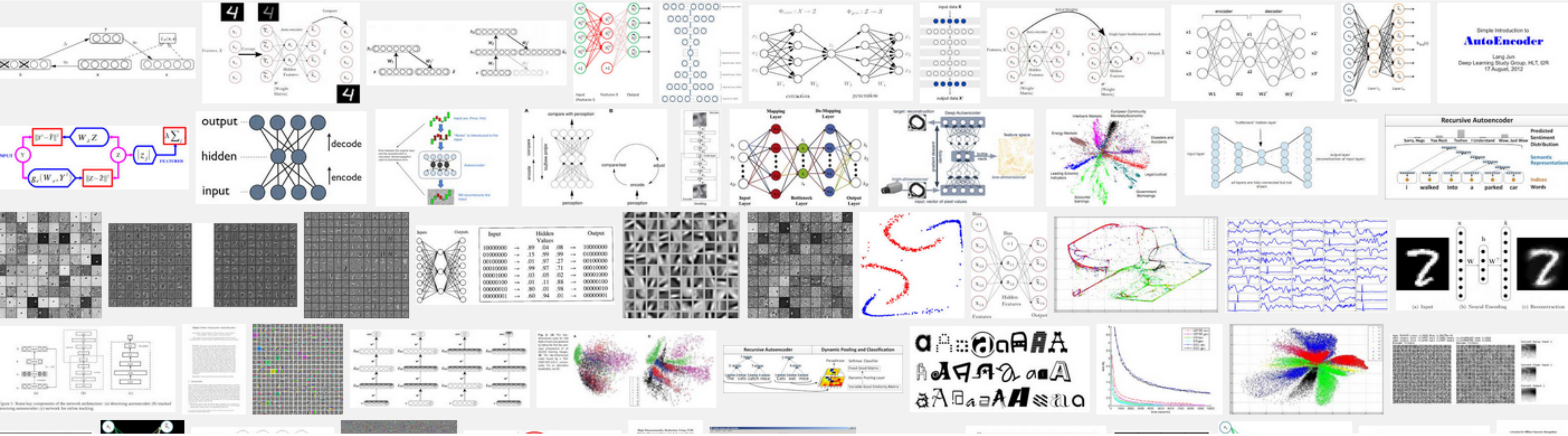
Greedy layer-wise training of deep networks



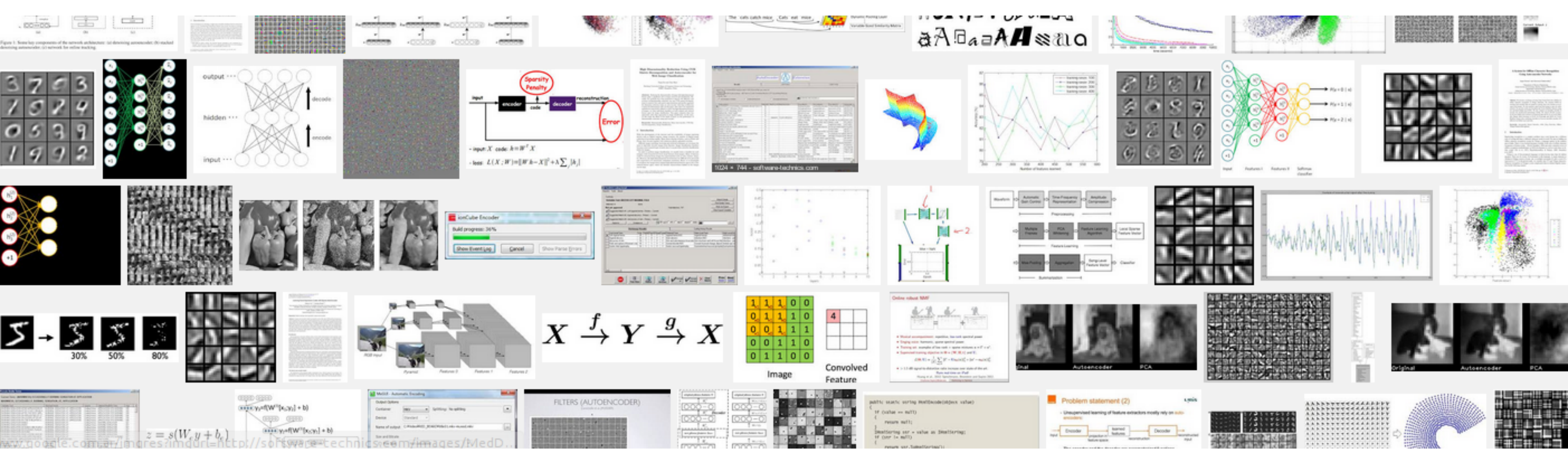
Hinton, Geoffrey E, and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507.

Hinton, Geoffrey, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.

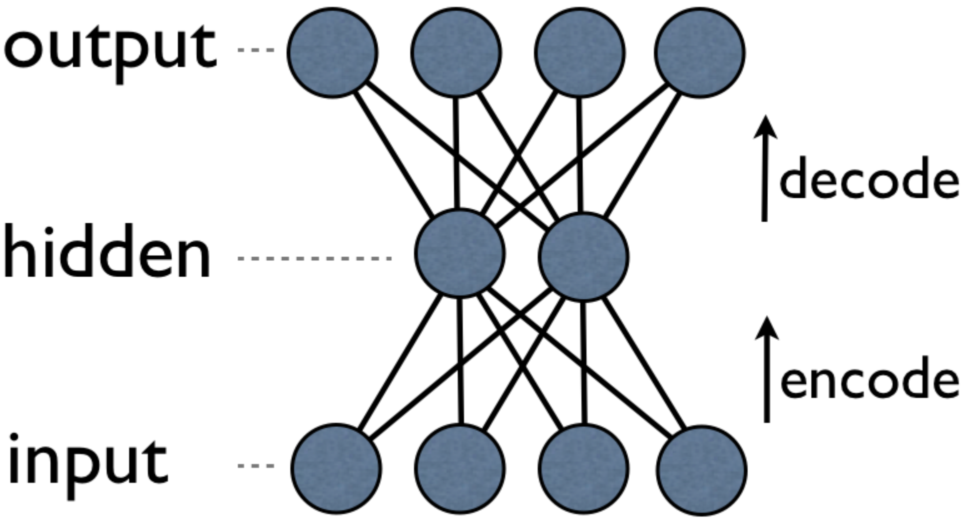
Bengio, Yoshua et al. "Greedy layer-wise training of deep networks." *Advances in neural information processing systems* 19 (2007): 153.



Auto-Encoders



Auto-Encoder



$$z = s(W'y + b')$$

$$y = s(Wx + b)$$

tied weights $W' = W^T$
(optional)

Reconstruction Error

squared error

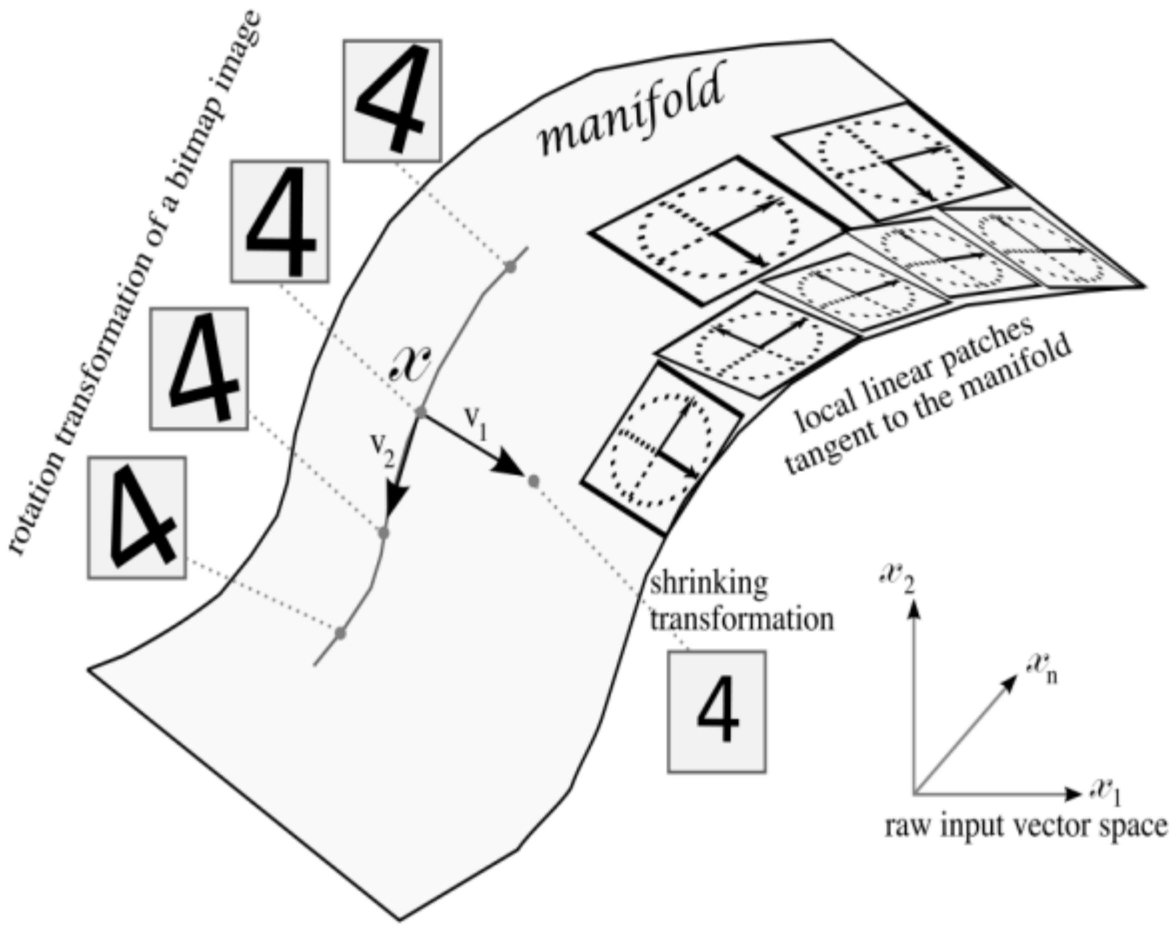
$$L(x, z) = \|x - z\|^2$$

cross-entropy

$$L_H(x, z) = -\sum_{k=1}^{d} [x_k \log z_k + (1 - x_k) \log(1 - z_k)]$$

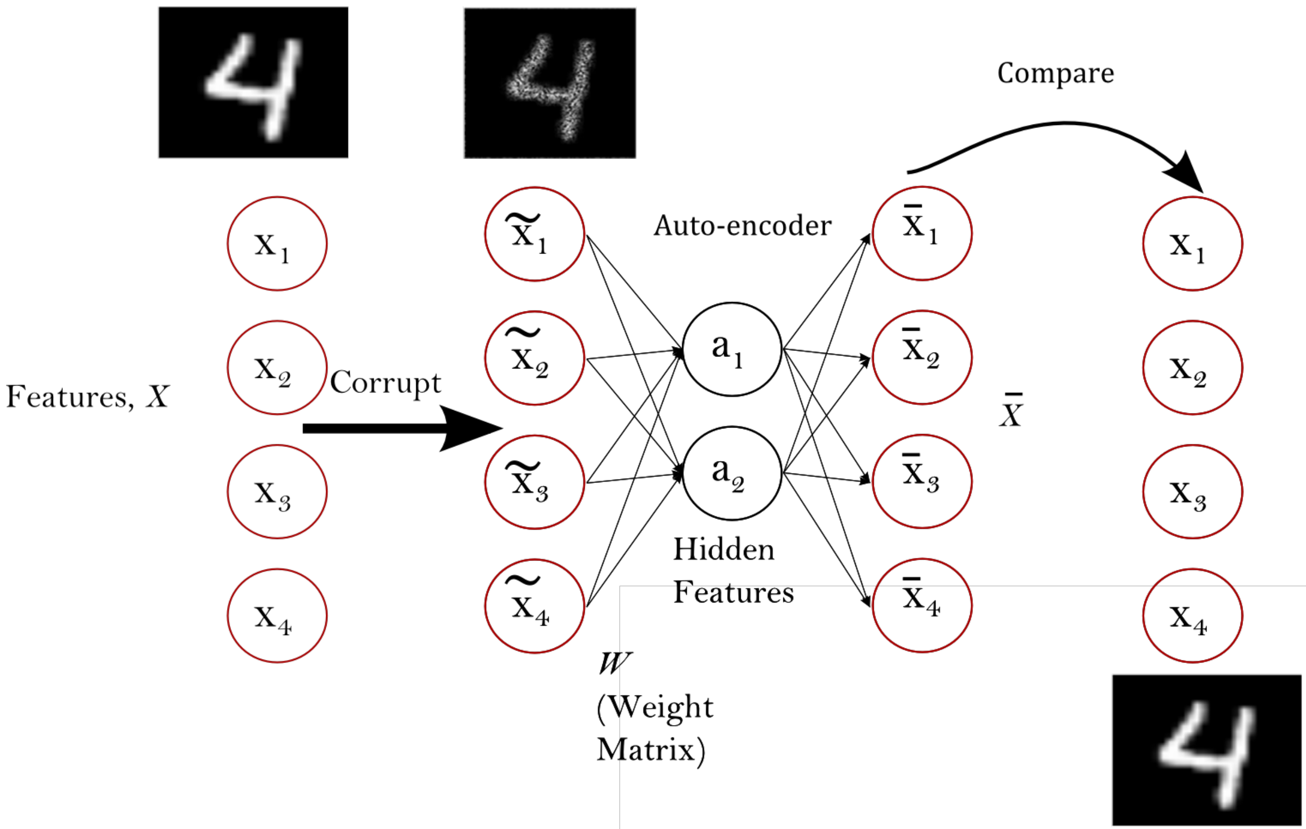
http://www.cifasis-conicet.gov.ar/granitto/ECI2014/

Manifold Hypothesis

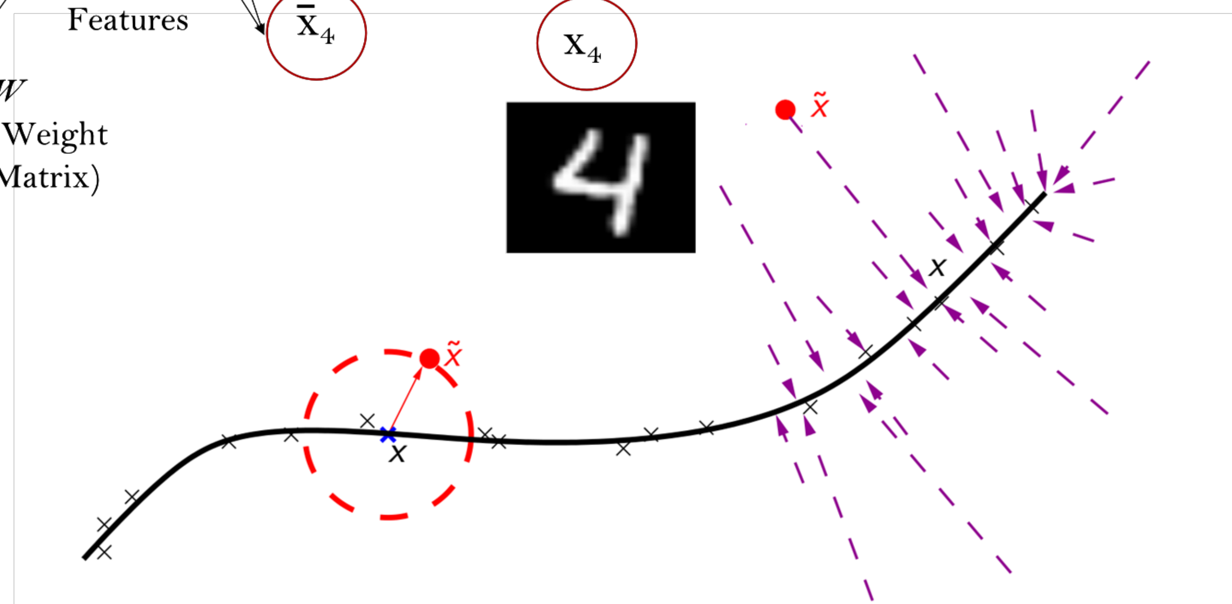


Additional prior:
data density concentrates near low-dimensional manifolds

Denoising Auto-Encoders



We train the auto-encoder to reconstruct the input from a corrupted version of it.



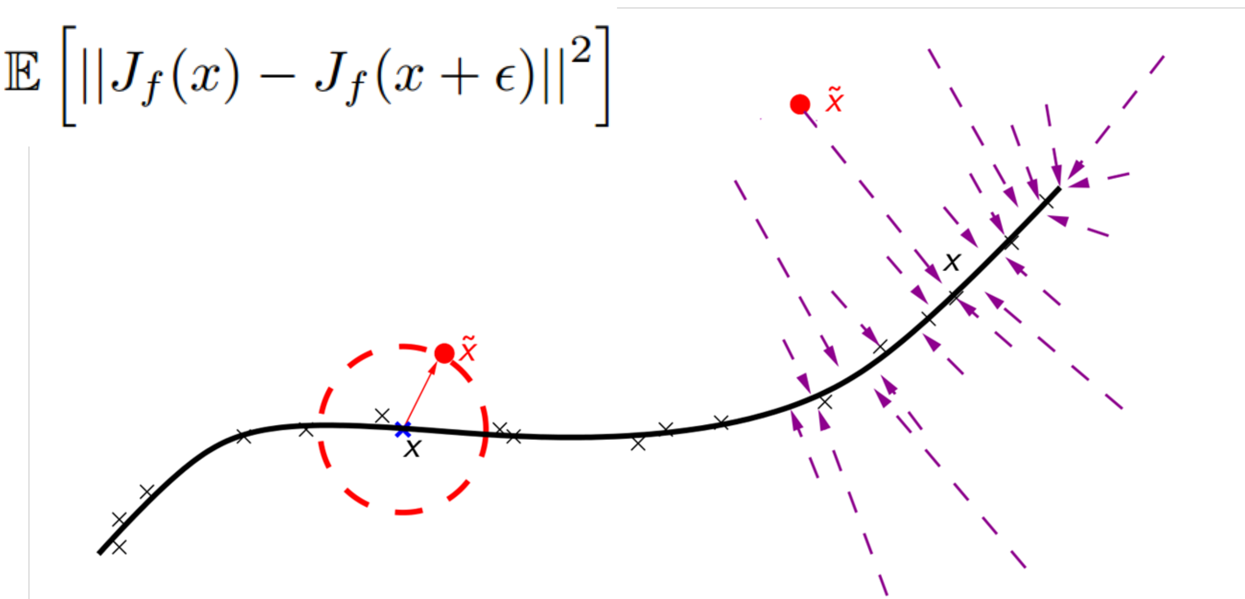
Contractive Auto-Encoders

First-order contractive Auto-Encoder

$$\mathcal{J}_{\mathrm{CAE}}(\theta) = \sum_{x \in D_n} L(x, g(f(x))) + \lambda \|J_f(x)\|^2$$

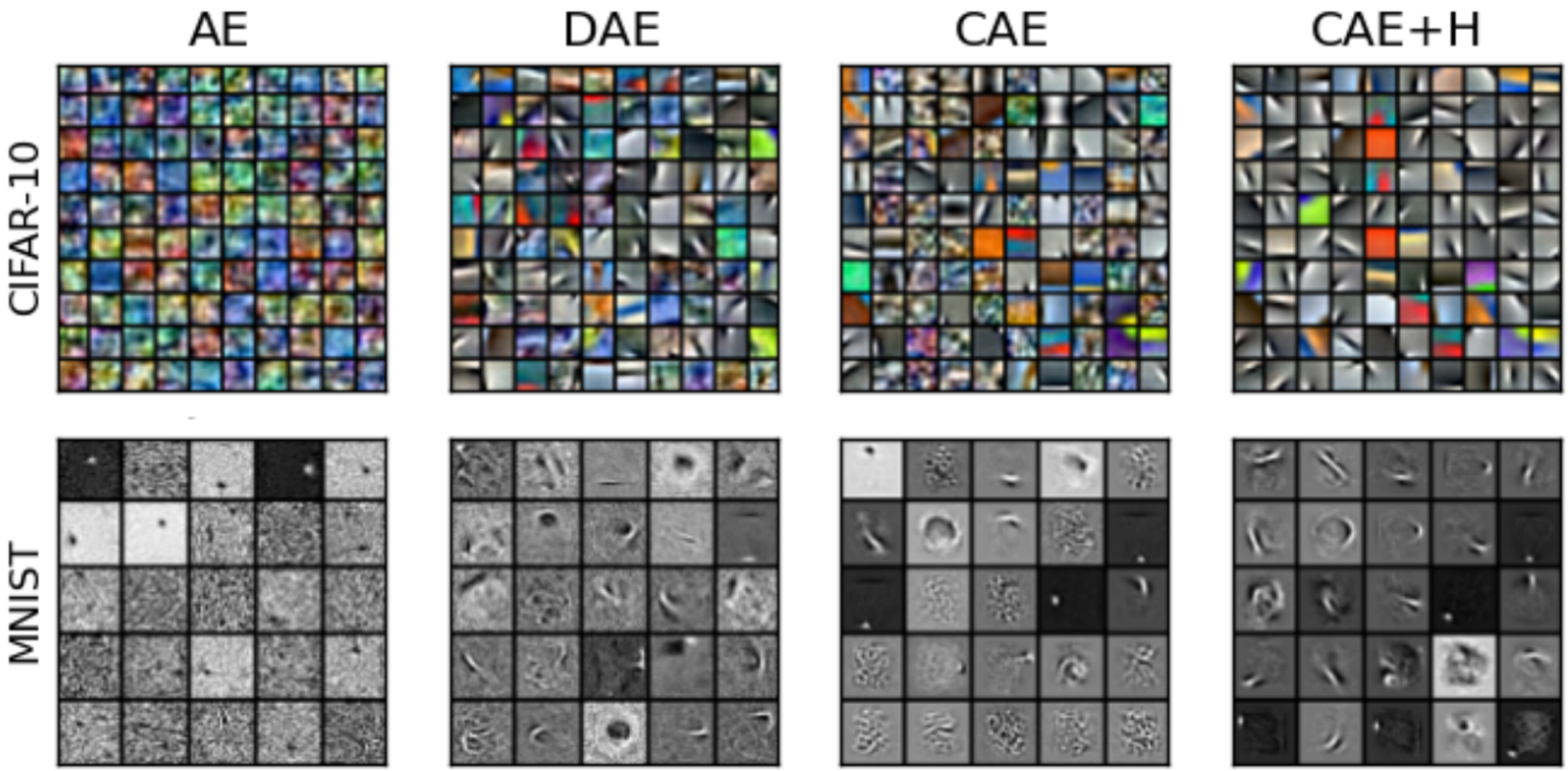
Higher order regularization:

$$\|H_f(x)\|^2 = \lim_{\sigma \to 0} \frac{1}{\sigma^2} \mathbb{E}\left[\|J_f(x) - J_f(x+\epsilon)\|^2 \right]$$

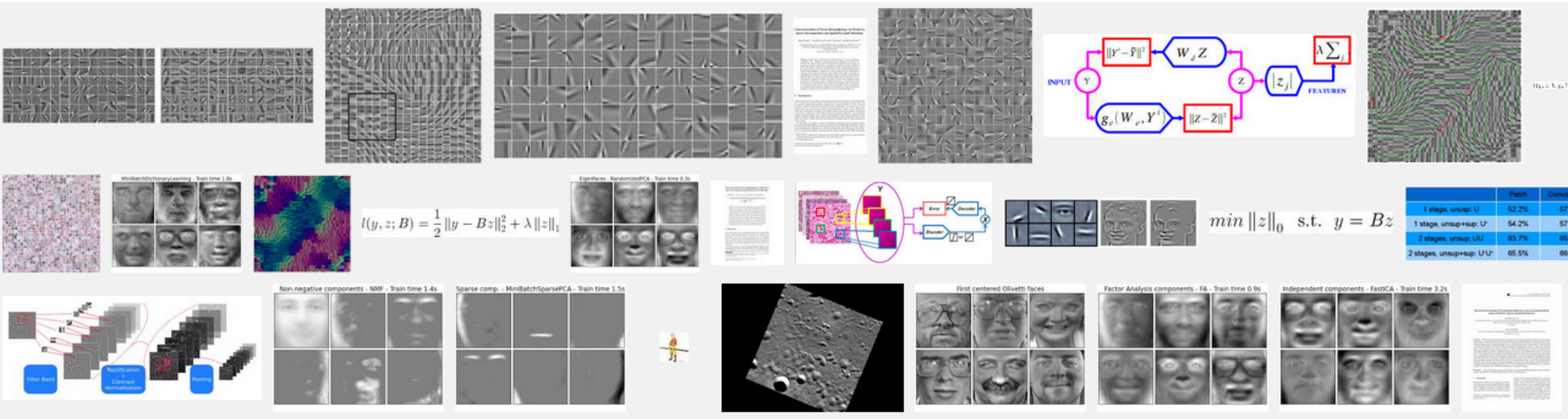


Rifai, Salah, et al. "Higher order contractive auto-encoder." *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2011. 645-660.

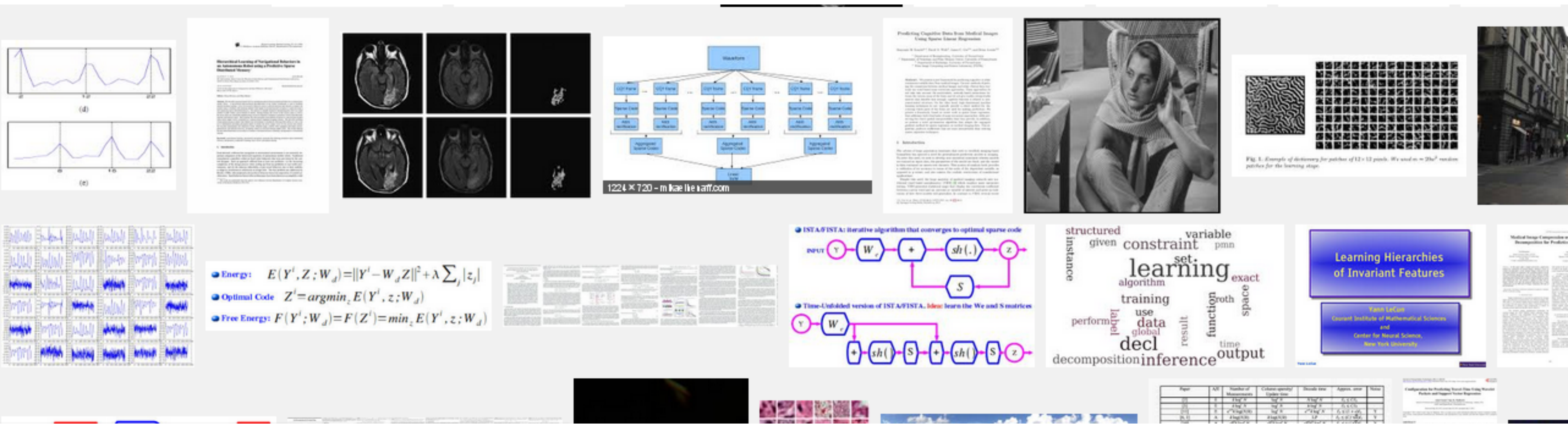
Contractive Auto-Encoders



Rifai, Salah, et al. "Higher order contractive auto-encoder." *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2011. 645-660.

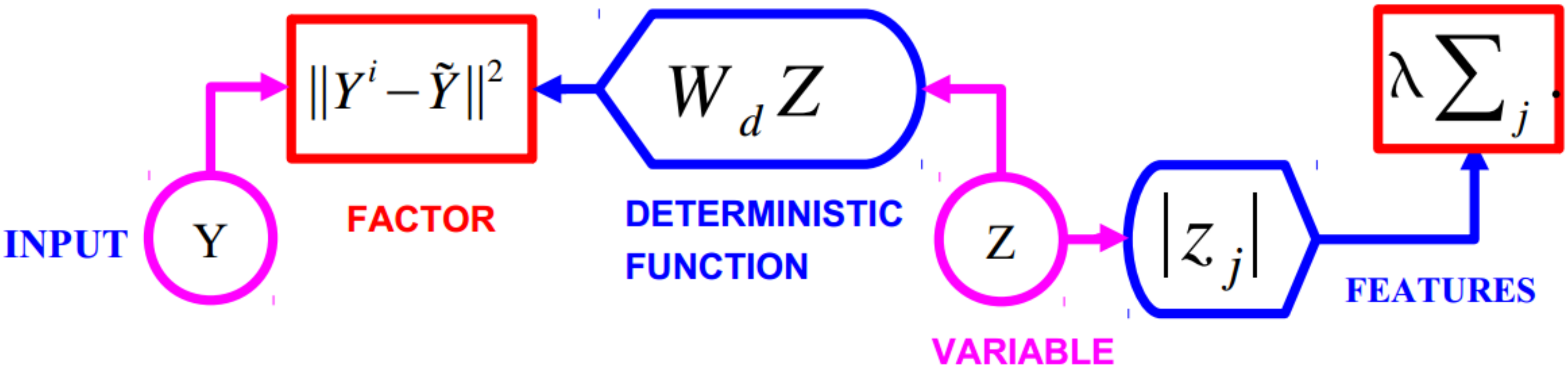


Predictive Sparse Decomposition (PSD)



Sparse Coding

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



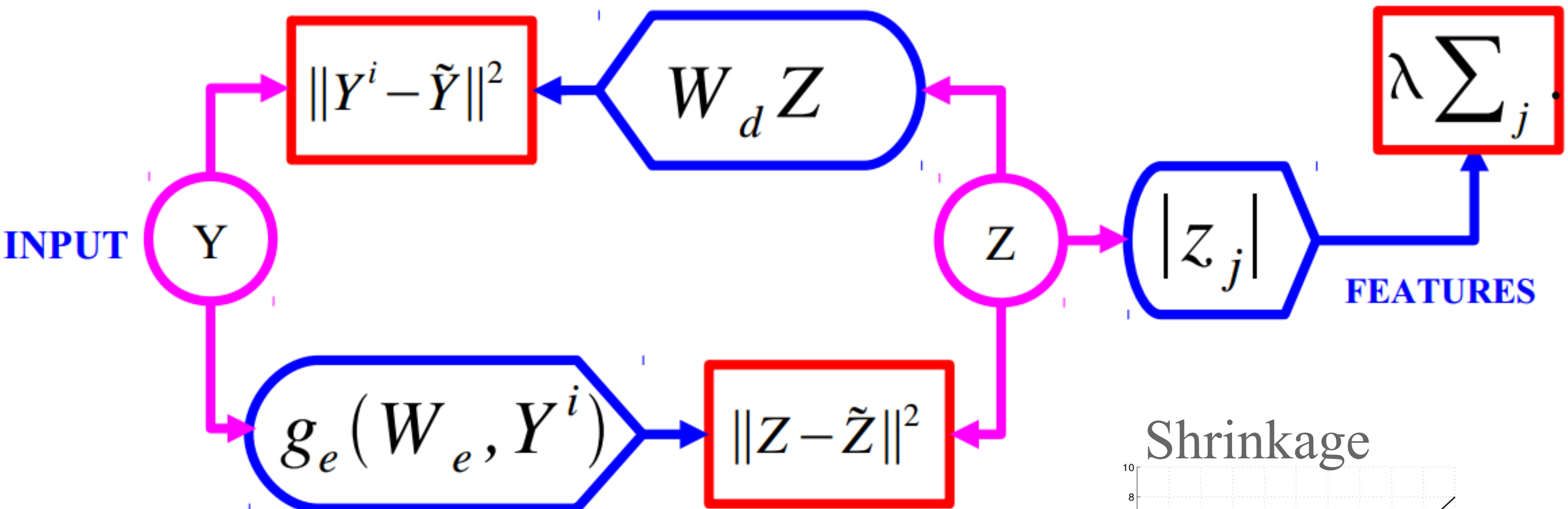
Inference is slow:

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

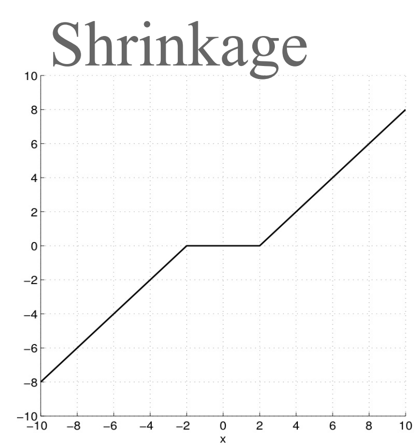
Olshausen, Bruno A, and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?." *Vision research* 37.23 (1997): 3311-3325.

Predictive Sparse Decomposition (PSD)

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



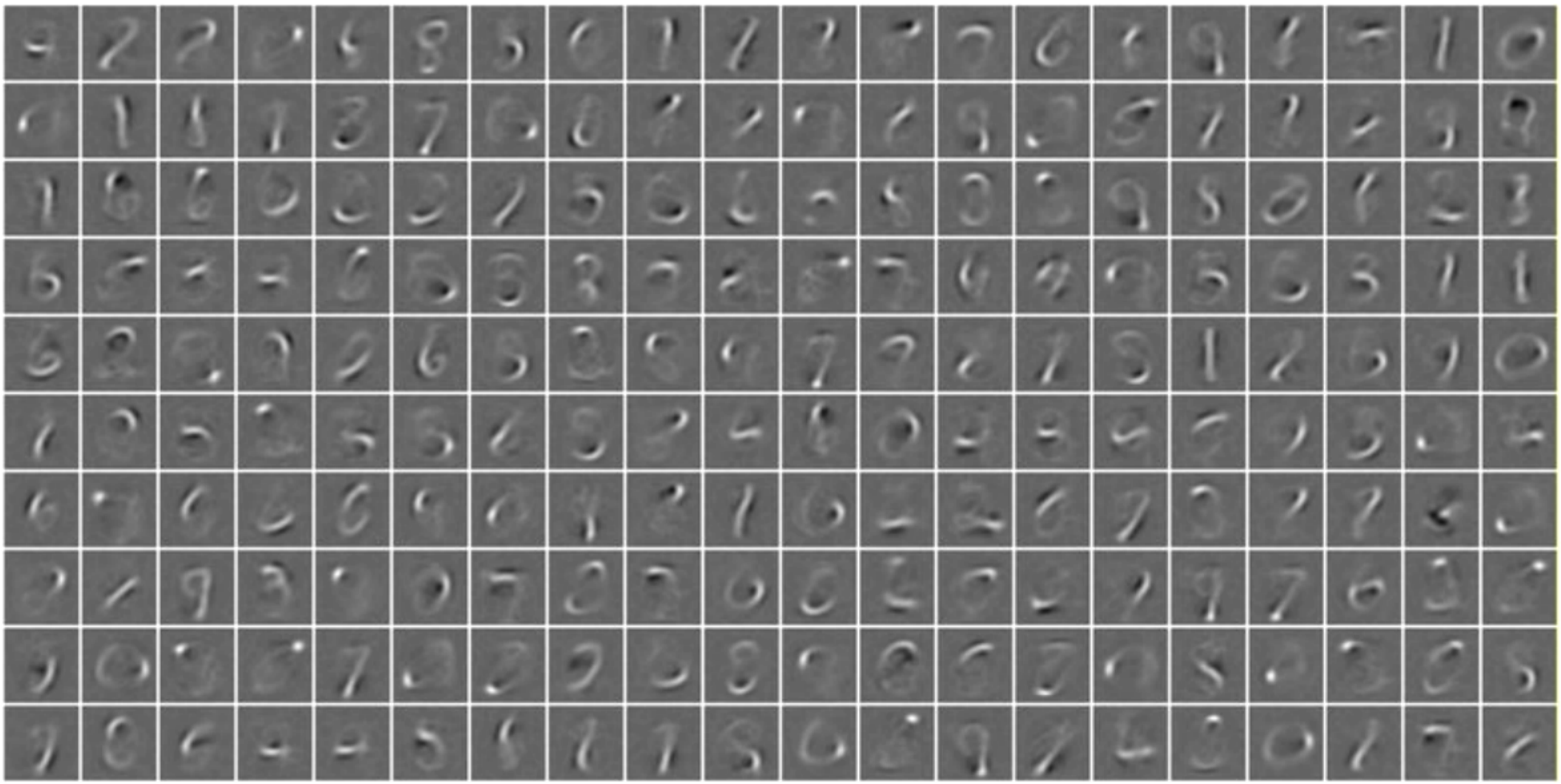
$$g_e(W_e, Y^i) = shrinkage(W_e Y^i)$$



Kavukcuoglu, Koray, Marc'Aurelio Ranzato, and Yann LeCun. "Fast inference in sparse coding algorithms with applications to object recognition." *arXiv preprint arXiv:1010.3467* (2008).

PSD: Basis Functions on MNIST

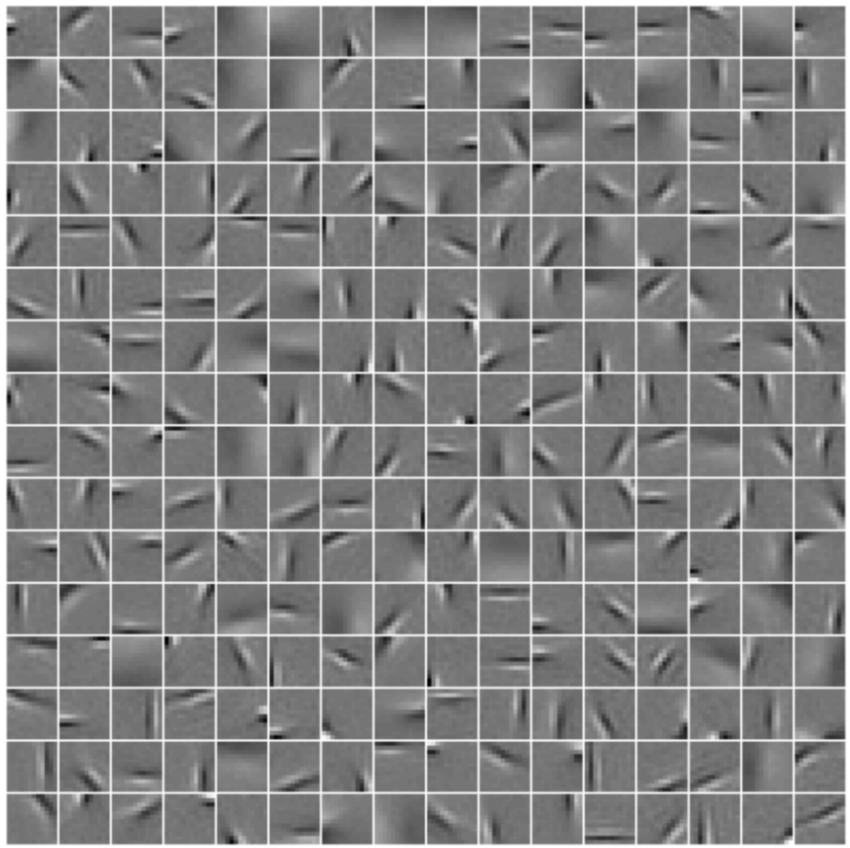
Basis functions (and encoder matrix) are digit parts



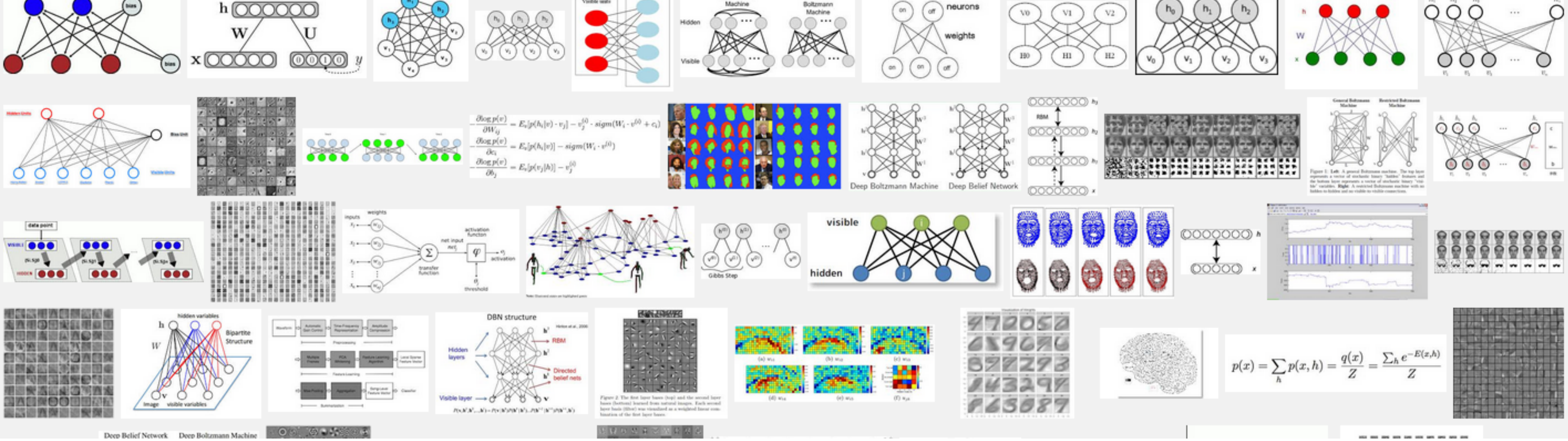
Kavukcuoglu, Koray, Marc'Aurelio Ranzato, and Yann LeCun. "Fast inference in sparse coding algorithms with applications to object recognition." *arXiv preprint arXiv:1010.3467* (2010).

PSD: Basis Functions on Natural Images Patches

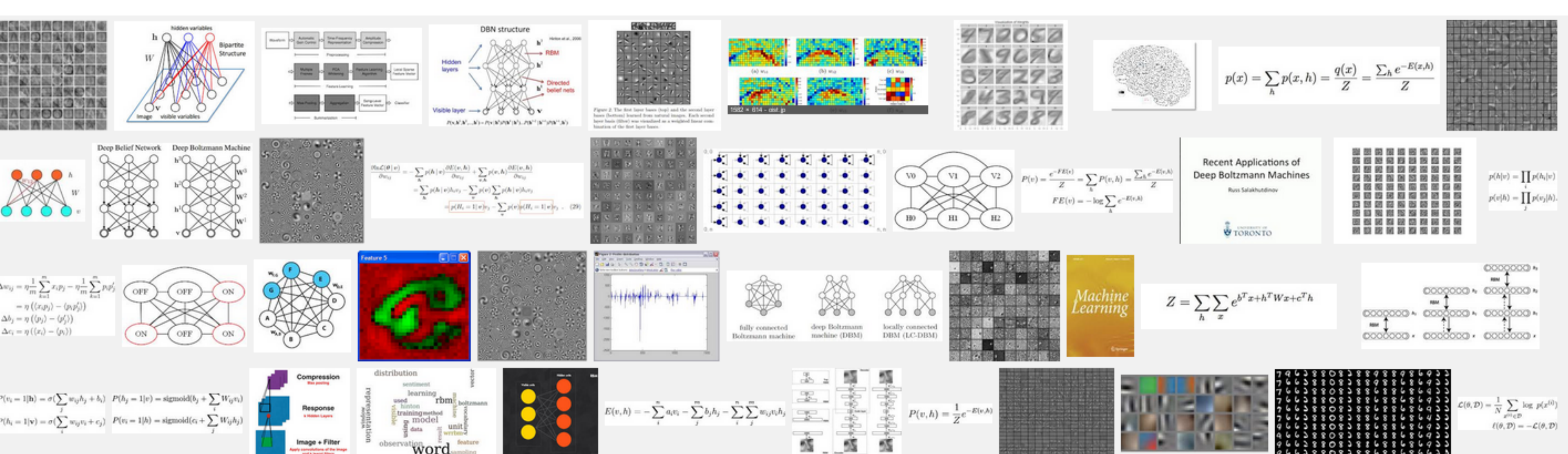
Basis functions are V1-like receptive fields



Kavukcuoglu, Koray, Marc'Aurelio Ranzato, and Yann LeCun. "Fast inference in sparse coding algorithms with applications to object recognition." *arXiv preprint arXiv:1010.3467* (2010).



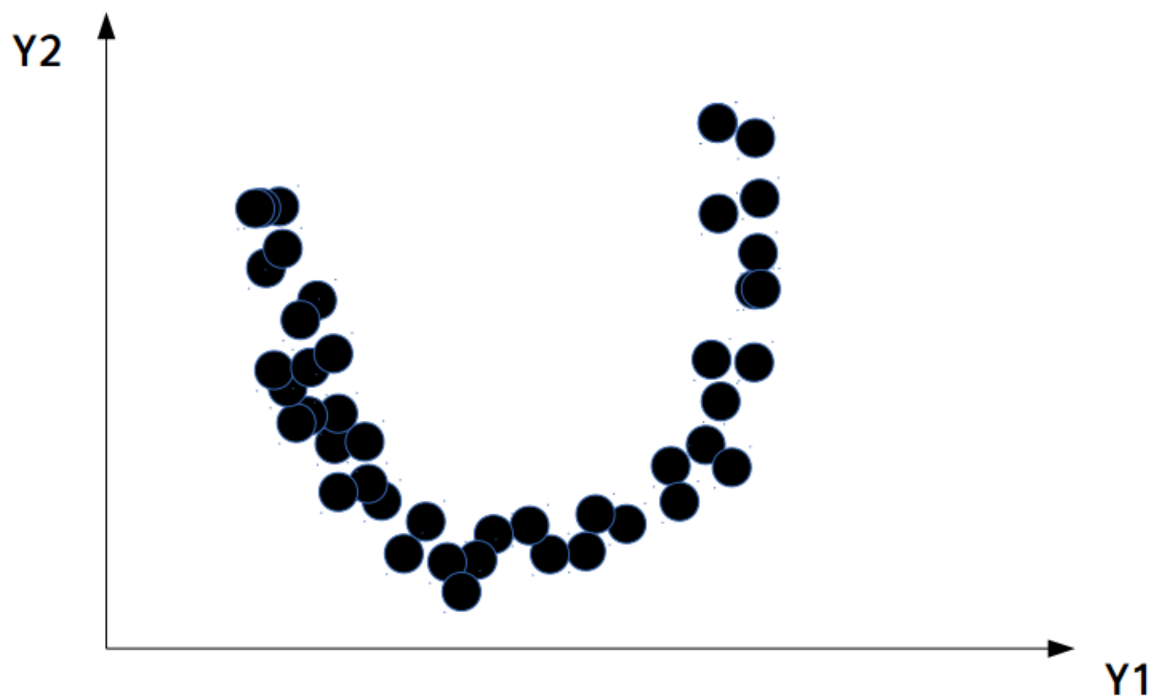
Energy-Based Unsupervised Learning



Energy-Based Unsupervised Learning

Learning an energy function that takes:

- low values on the data manifold
- higher values everywhere else

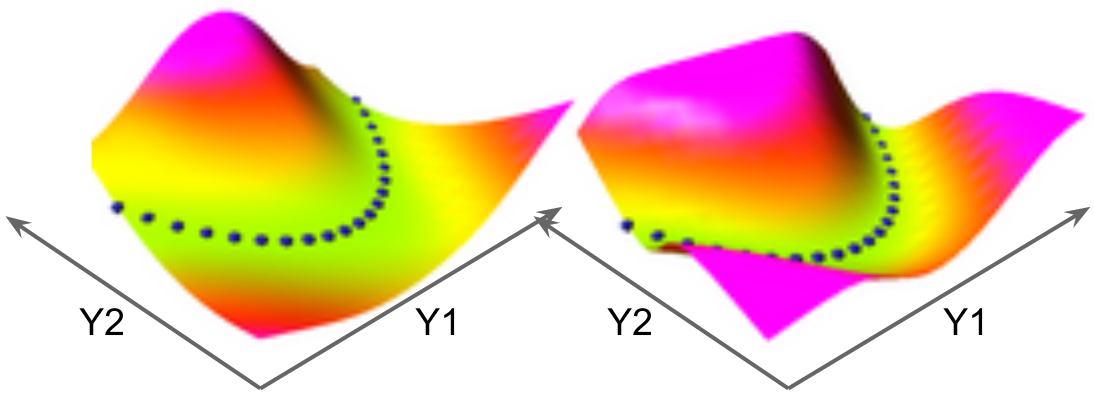


Source: Y LeCun & MA Ranzato. ICML 2013

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 $Y2 = (Y1)^2$



[Source: Y LeCun & MA Ranzato. ICML 2013]

Capturing Dependencies Between Variables with an Energy Function

The energy can be interpreted as an unnormalized negative log density

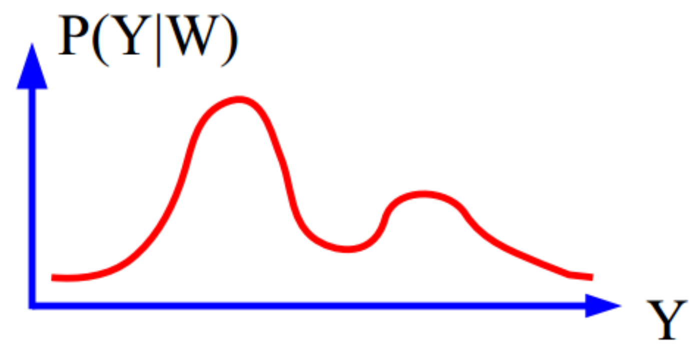
Gibbs distribution: Probability proportional to exp(-energy)

- Beta parameter is akin to an inverse temperature

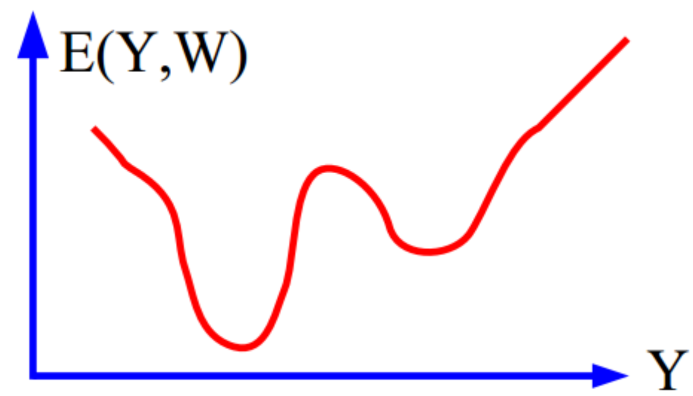
Don't compute probabilities unless you absolutely have to

- Because the denominator is often intractable

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



$$E(Y,W) \propto -\log P(Y|W)$$

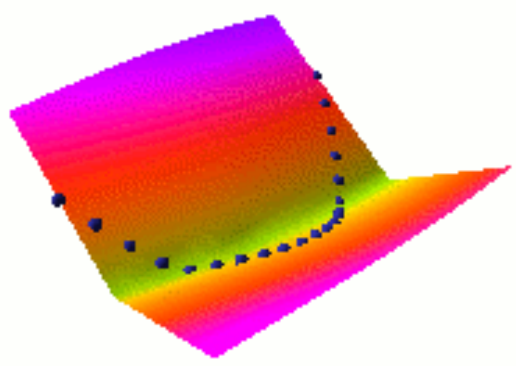


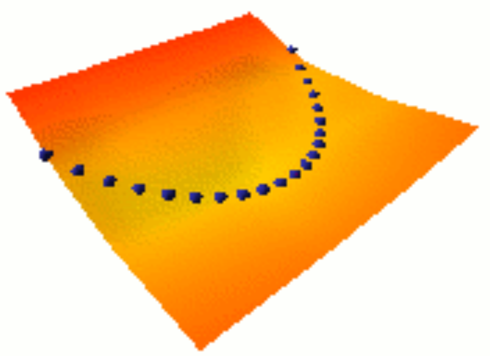
Source: Y LeCun & MA Ranzato. ICML 2013

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?**





[Source: Y LeCun & MA Ranzato. ICML 2013]

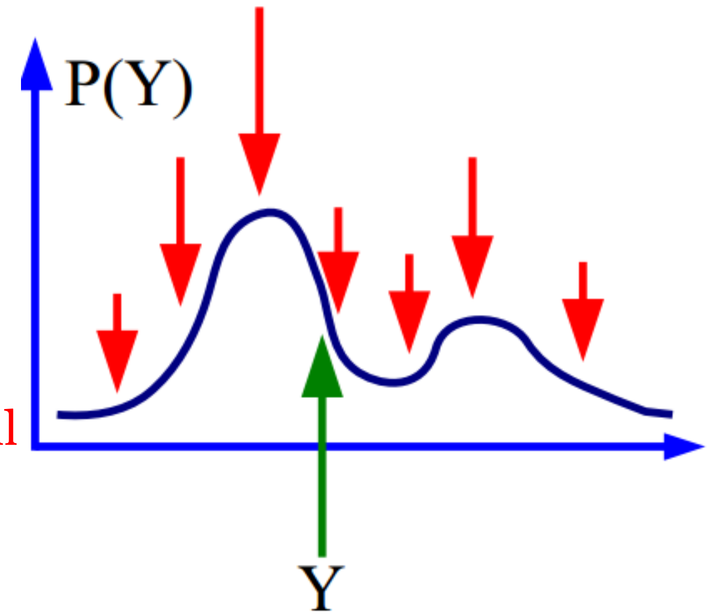
Max Likelihood

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

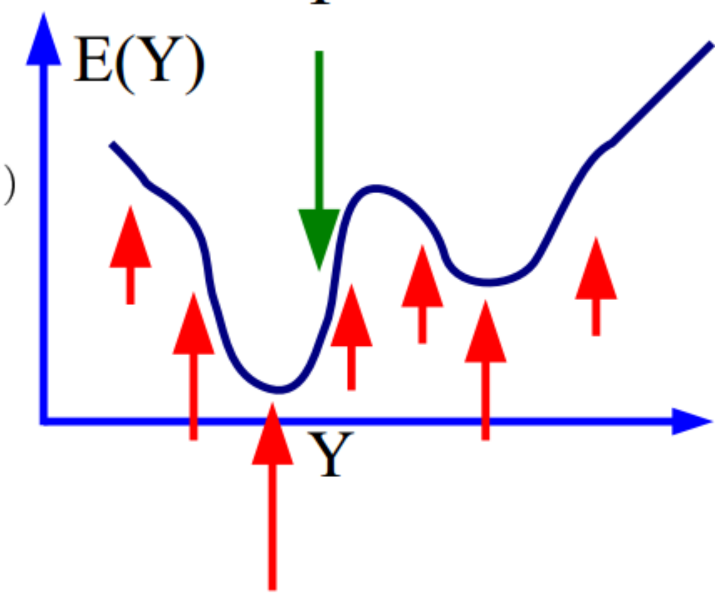


Minimizing -log P(Y,W) on
training samples

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

make this small

make this big

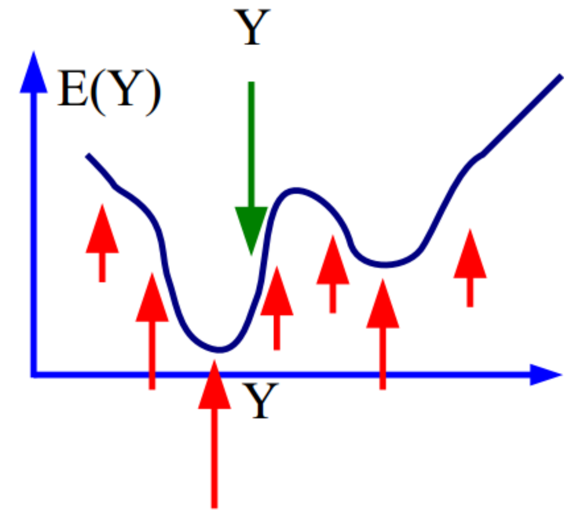


Source: Y LeCun & MA Ranzato. ICML 2013

Max Likelihood

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

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

[Source: Y LeCun & MA Ranzato. ICML 2013]

Contrastive Divergence (CD)

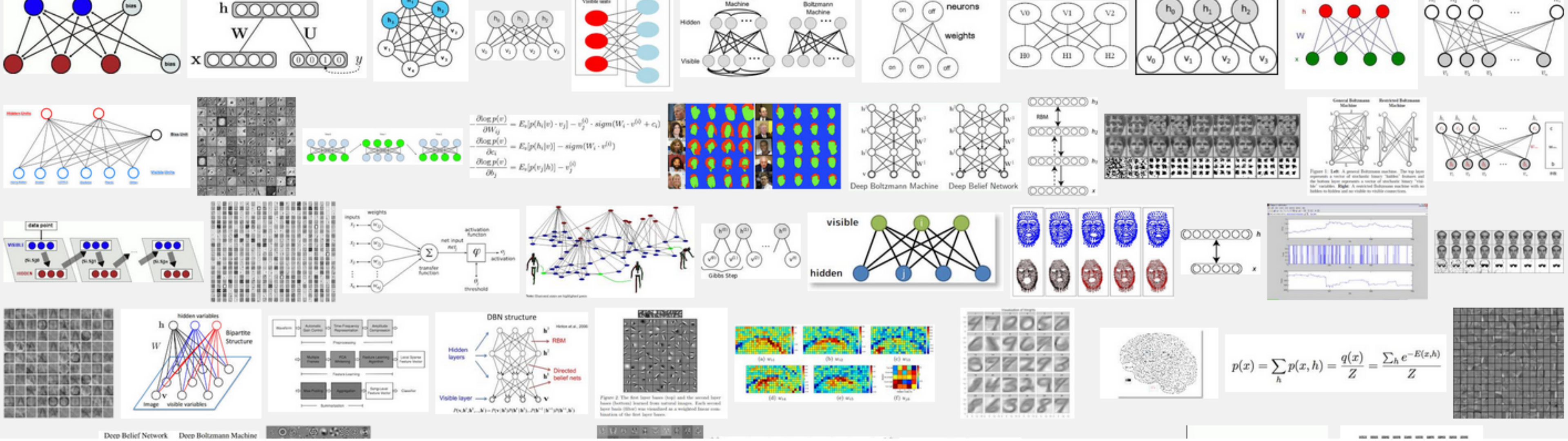
Basic Idea:

- Pick a training sample, lower the energy at that point
- From the sample, move down in the energy surface with noise
- Stop after a while
- Push up on the energy of the point where we stopped
- This creates grooves in the energy surface around data manifolds

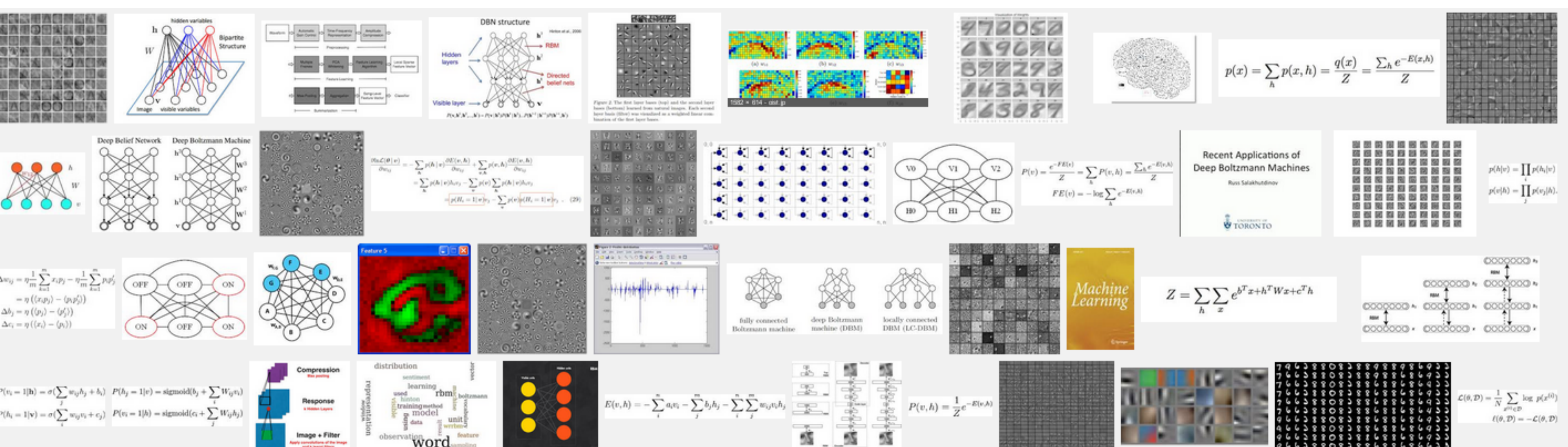
Persistent CD: use a bunch of “particles” and remember their positions

- Make them roll down the energy surface with noise
- Push up on the energy wherever they are
- Faster than CD

Source: Y LeCun & MA Ranzato. ICML 2013



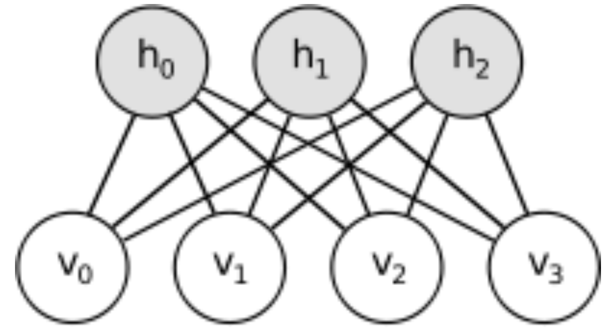
Restricted Boltzmann Machines (RBM)



Restricted Boltzmann Machines (RBM)

Energy function

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{visible}} a_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$



$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\, e^{-E(\mathbf{v},\mathbf{h})} \qquad Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

learning rate

Energy Based Models (EBM)

$$p(x) = \frac{e^{-E(x)}}{Z}$$

$e^{-E(x)}$ → scalar energy

Z → partition function

$$Z = \sum_x e^{-E(x)}$$

Log-likelihood

$$L(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x^{(i)} \in \mathcal{D}} \log p(x^{(i)})$$

Negative Log-likelihood (NLL)

$$\ell(\theta, \mathcal{D}) = -L(\theta, \mathcal{D})$$

EBMs with Hidden Units

$$P(x) = \sum_h P(x, h) = \sum_h \frac{e^{-E(x,h)}}{Z}$$

observed variables

hidden variables

Free energy:

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$$

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \quad \Rightarrow \quad Z = \sum_x e^{-\mathcal{F}(x)}$$

EBMs with Hidden Units

Gradient descent

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial \mathcal{F}(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}$$

expectation over all possible configurations of the input x under the distribution P formed by the model.

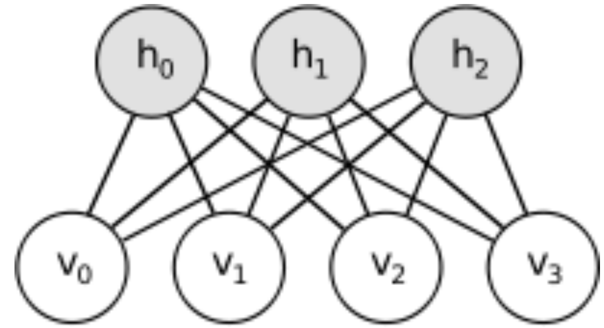
$$E_P \left[\frac{\partial \mathcal{F}(x)}{\partial \theta} \right]$$

$$-\frac{\partial \log p(x)}{\partial \theta} \approx \frac{\partial \mathcal{F}(x)}{\partial \theta} - \frac{1}{|\mathcal{N}|} \sum_{\tilde{x} \in \mathcal{N}} \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}$$

Restricted Boltzmann Machines (RBM)

Energy function

$$E(x, h) = -b^T x - c^T h - h^T W x$$



Free energy

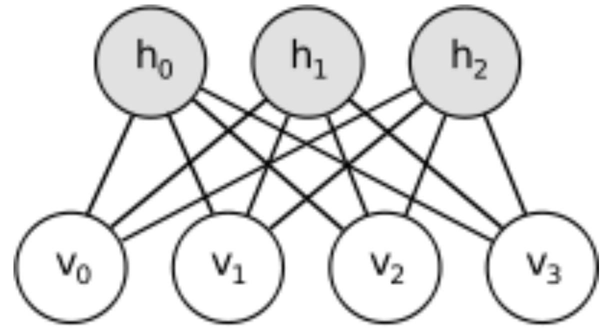
$$\mathcal{F}(x) = -b^T x - \sum_i \log \sum_{h_i} e^{h_i (c_i + W_i x)}$$

Conditional independence:

$$p(h|x) = \prod_i p(h_i|x) \qquad p(x|h) = \prod_i p(x_i|h)$$

RBMs with binary units

$$\begin{aligned} P(h_i = 1|x) &= \text{sigmoid}(c_i + W_i x) \\ P(x_j = 1|h) &= \text{sigmoid}(b_j + W_j^T x) \end{aligned}$$



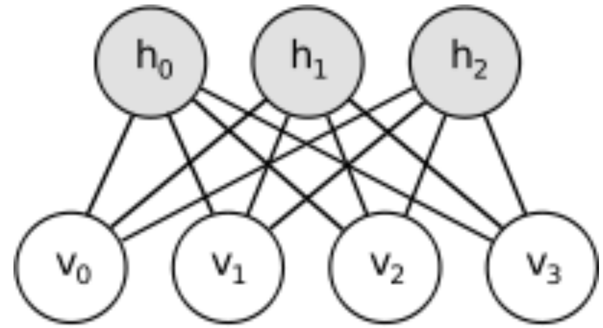
$$\mathcal{F}(x) = -b^T x - \sum_i \log \sum_{h_i \in \{0,1\}} e^{h_i(c_i + W_i x)}$$

$$\Downarrow$$

$$\mathcal{F}(x) = -b^T x - \sum_i \log(1 + e^{c_i + W_i x})$$

Update Equations

$$\begin{cases} \mathcal{F}(x) = -b^T x - \sum_i \log(1 + e^{c_i + W_i x}) \\ -\dfrac{\partial \log p(x)}{\partial \theta} \approx \dfrac{\partial \mathcal{F}(x)}{\partial \theta} - \dfrac{1}{|\mathcal{N}|} \sum_{\tilde{x} \in \mathcal{N}} \dfrac{\partial \mathcal{F}(\tilde{x})}{\partial \theta} \end{cases}$$



$$\begin{cases} -\dfrac{\partial \log p(x)}{\partial W_{ij}} = E_x[p(h_i|x) \cdot x_j] - x_j \cdot \text{sigmoid}(W_i \cdot x + c_i) \\ -\dfrac{\partial \log p(x)}{\partial c_i} = E_x[p(h_i|x)] - \text{sigmoid}(W_i \cdot x) \\ -\dfrac{\partial \log p(x)}{\partial b_j} = E_x[p(x_j|h)] - x_j \end{cases}$$

Update Equations

$$
\begin{cases}
-\dfrac{\partial \log p(x)}{\partial W_{ij}} & = & E_x[p(h_i|x) \cdot x_j] - x_j \cdot \mathrm{sigmoid}(W_i \cdot x + c_i) \\
-\dfrac{\partial \log p(x)}{\partial c_i} & = & E_x[p(h_i|x)] - \mathrm{sigmoid}(W_i \cdot x) \\
-\dfrac{\partial \log p(x)}{\partial b_j} & = & E_x[p(x_j|h)] - x_j
\end{cases}
$$

Log-likelihood

$$L(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x^{(i)} \in \mathcal{D}} \log p(x^{(i)})$$

Negative Log-likelihood (NLL)

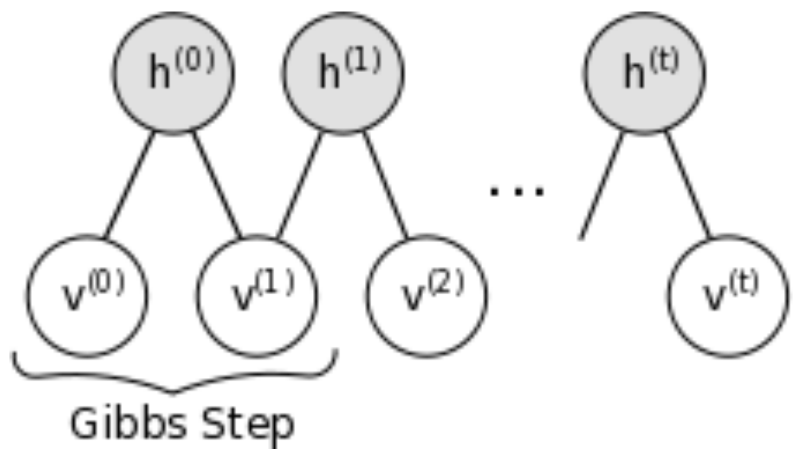
$$\ell(\theta, \mathcal{D}) = -L(\theta, \mathcal{D})$$

$$
\begin{cases}
\Delta W_{ij} & = & \eta \left(E^{(data)}[h_i x_j] - E^{(model)}[h_i x_j] \right) \\
\Delta c_i & = & \eta \left(E^{(data)}[h_i] - E^{(model)}[h_i] \right) \\
\Delta b_j & = & \eta \left(E^{(data)}[x_j] - E^{(model)}[x_j] \right)
\end{cases}
$$

Sampling in an RBM

Markov chain step

$$\begin{aligned} h^{(n+1)} &\sim \text{sigmoid}(W^T x^{(n)} + c) \\ x^{(n+1)} &\sim \text{sigmoid}(W h^{(n+1)} + b) \end{aligned}$$

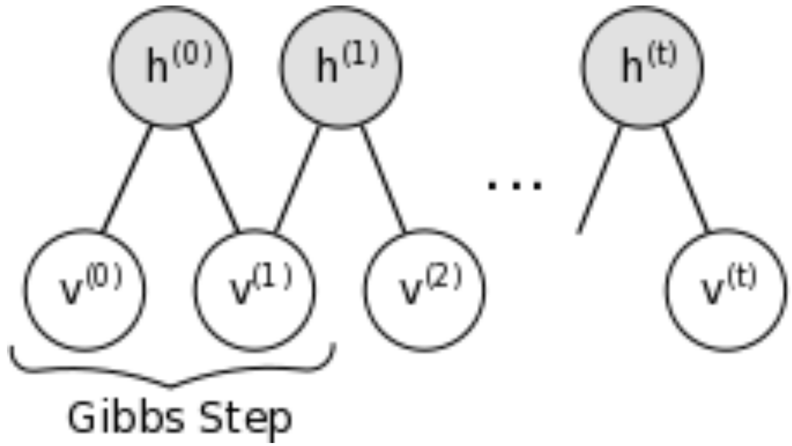


Contrastive Divergence (CD-k)

- Initialize the Markov chain with a training example

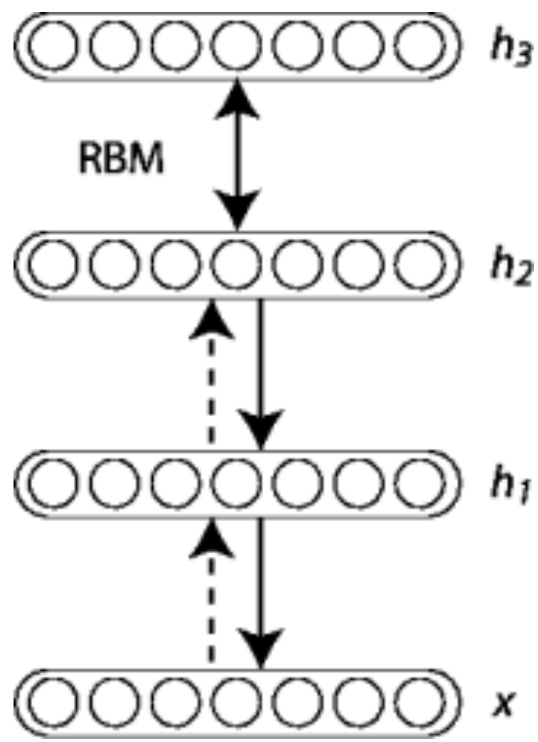
- CD does not wait for the chain to converge.

Samples are obtained after only k-steps of Gibbs sampling



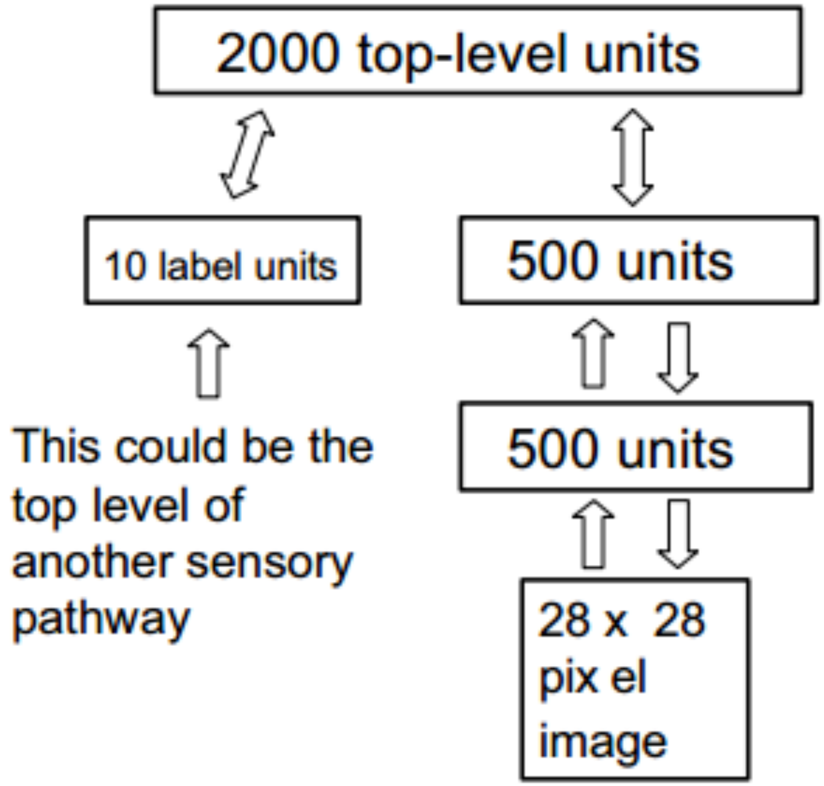
Deep Belief Networks (DBN)

$$P(x, h^1, \ldots, h^\ell) = \left(\prod_{k=0}^{\ell-2} P(h^k | h^{k+1}) \right) P(h^{\ell-1}, h^\ell)$$



Deep Belief Networks (DBN)

Hinton, G. E., Osindero, S. and Teh, Y. (2006) A fast learning algorithm for deep belief nets. Neural Computation 18, pp 1527-1554. [ps.gz] [pdf]



Deep Belief Networks (DBN)

Multiresolution Deep Belief Networks **(AISTATS 2012)**
Yichuan Tang and *Abdel-rahman Mohamed*
Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS), 2012, La Palma, Canary Islands
[pdf] [poster] [bibtex]

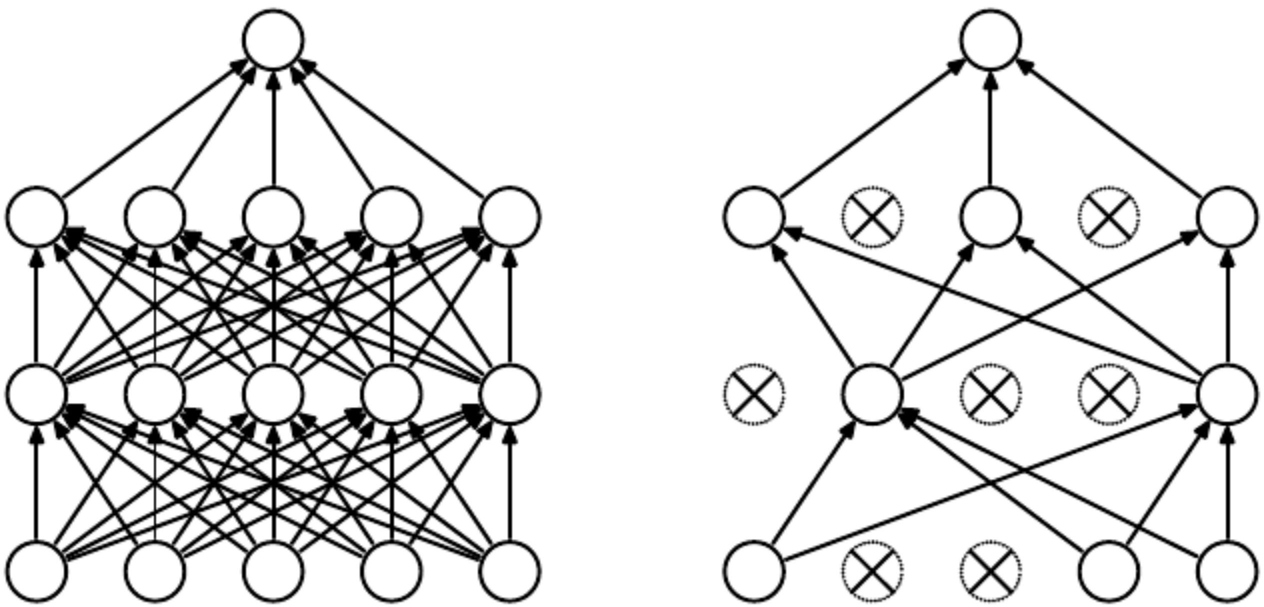
Samples drawn from MrDBN. Starting at the top row, from left to right, each sample is generated after 100 steps of block Gibbs sampling. Note the level of details and the ability of the MCMC chain to mix rapidly.

Dropout:

A Simple Way to Prevent Neural Networks from Overfitting

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

Dropout Neural Net Model

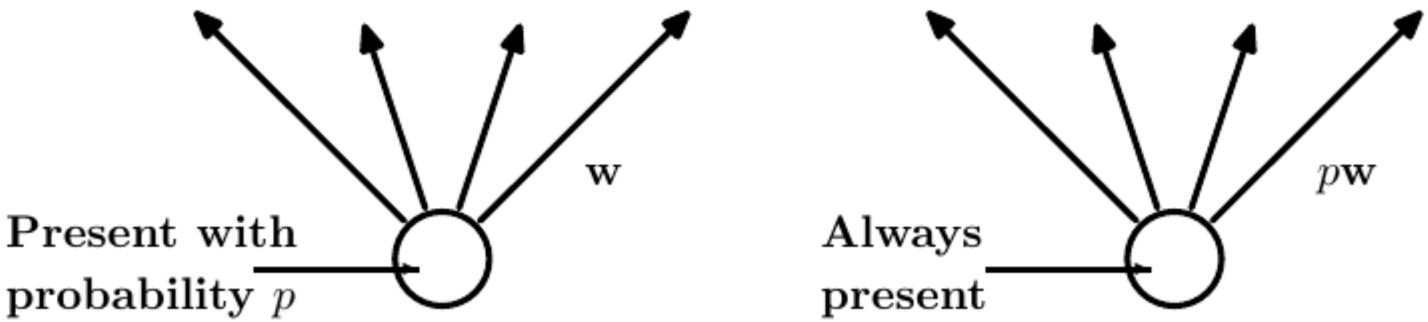


(a) Standard Neural Net (b) After applying dropout.

Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

Dropout units



(a) At training time (b) At test time

Left: A unit at training time that is present with probability p and is connected to units in the next layer with weights $\mathbf{w}$. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

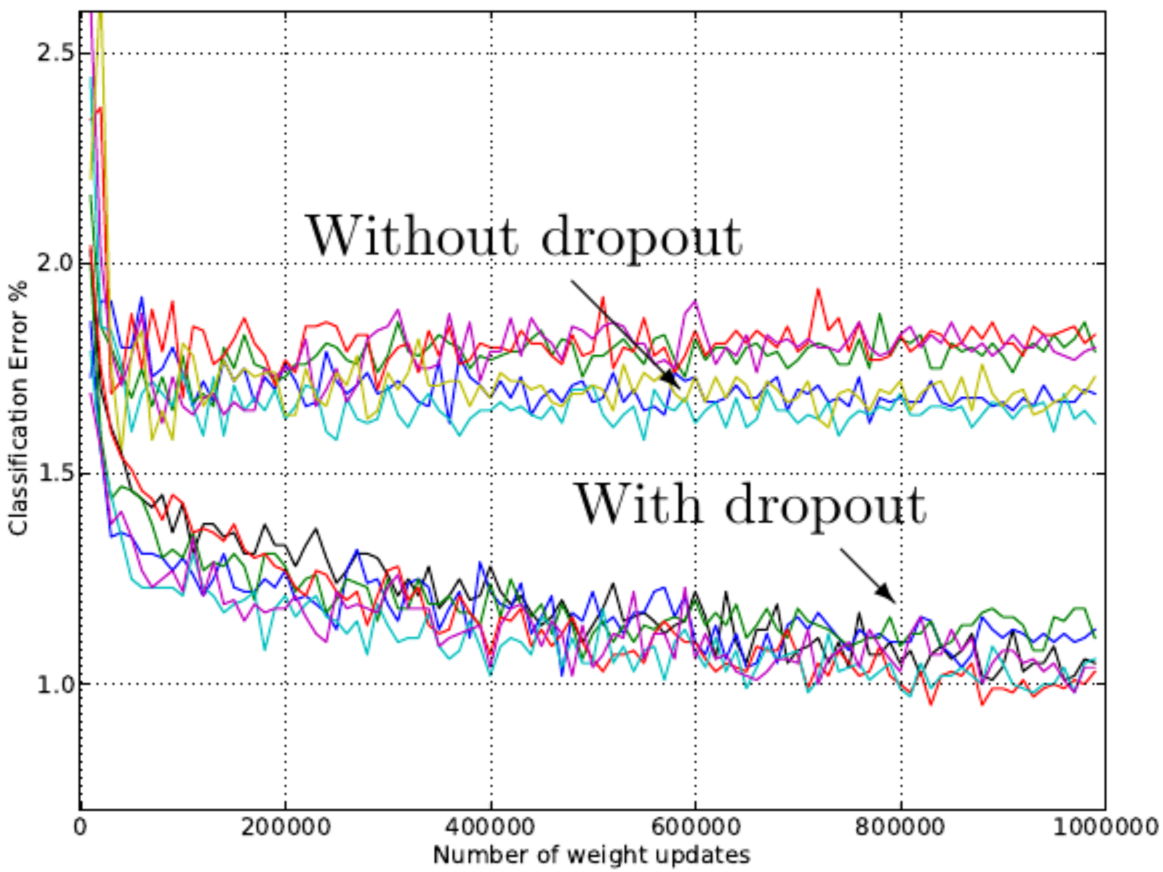
Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

MNIST results

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5×240) units	0.94
DBN + finetuning (Hinton and Salakhutdinov, 2006)	Logistic	500-500-2000	1.18
DBM + finetuning (Salakhutdinov and Hinton, 2009)	Logistic	500-500-2000	0.96
DBN + dropout finetuning	Logistic	500-500-2000	0.92
DBM + dropout finetuning	Logistic	500-500-2000	**0.79**

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

Dropout: Robustness



Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

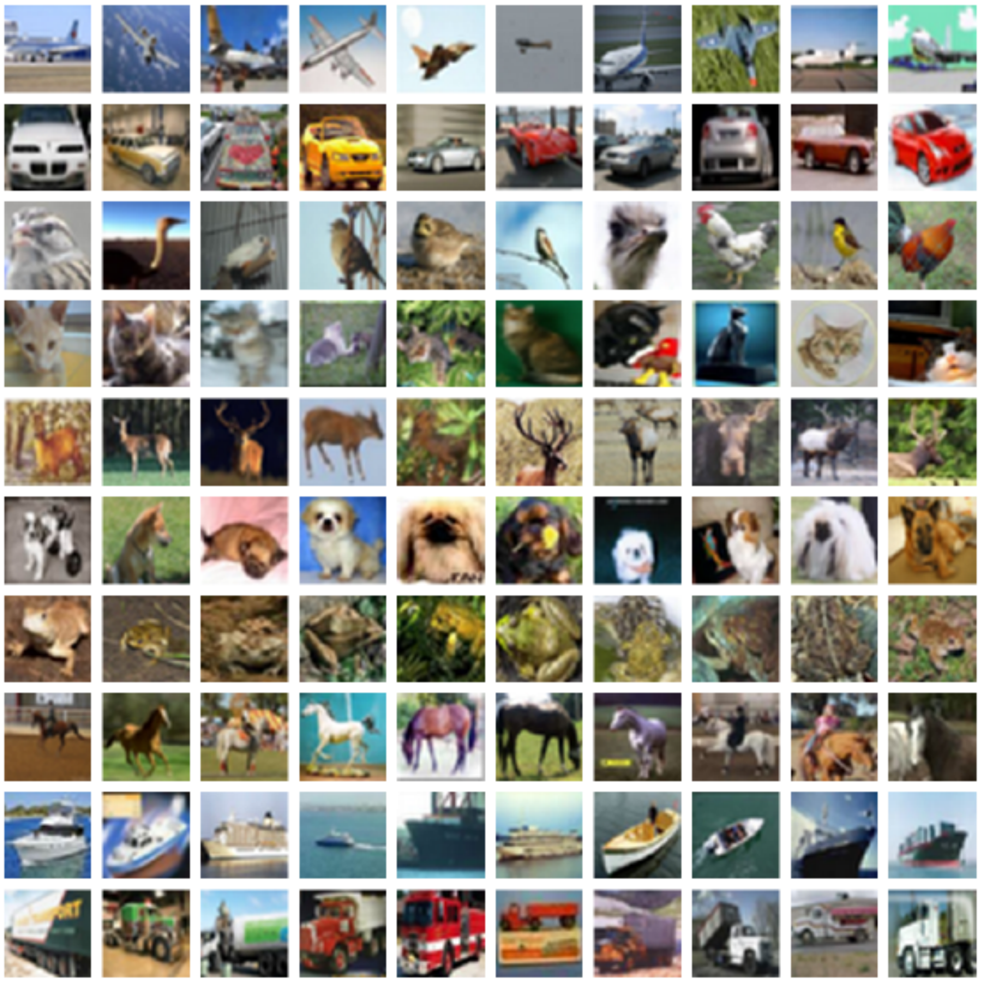
Dropout: Street View House Numbers

Method	Error %
Binary Features (WDCH) (Netzer et al., 2011)	36.7
HOG (Netzer et al., 2011)	15.0
Stacked Sparse Autoencoders (Netzer et al., 2011)	10.3
KMeans (Netzer et al., 2011)	9.4
Multi-stage Conv Net with average pooling (Sermanet et al., 2012)	9.06
Multi-stage Conv Net + L2 pooling (Sermanet et al., 2012)	5.36
Multi-stage Conv Net + L4 pooling + padding (Sermanet et al., 2012)	4.90
Conv Net + max-pooling	3.95
Conv Net + max pooling + dropout in fully connected layers	3.02
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	2.80
Conv Net + max pooling + dropout in all layers	2.55
Conv Net + maxout (Goodfellow et al., 2013)	**2.47**
Human Performance	2.0

Table 3: Results on the Street View House Numbers data set.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

CIFAR-10 and CIFAR-100 datasets



The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

Dropout: CIFAR datasets

Method	CIFAR-10	CIFAR-100
Conv Net + max pooling (hand tuned)	15.60	43.48
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	15.13	42.51
Conv Net + max pooling (Snoek et al., 2012)	14.98	-
Conv Net + max pooling + dropout fully connected layers	14.32	41.26
Conv Net + max pooling + dropout in all layers	12.61	**37.20**
Conv Net + maxout (Goodfellow et al., 2013)	**11.68**	38.57

Error rates on CIFAR-10 and CIFAR-100.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (2014): 1929-1958.

Dropout: ImageNet

Model	Top-1	Top-5
Sparse Coding (Lin et al., 2010)	47.1	28.2
SIFT + Fisher Vectors (Sanchez and Perronnin, 2011)	45.7	25.7
Conv Net + dropout (Krizhevsky et al., 2012)	37.5	17.0

Results on the ILSVRC-2010 test set

Dropout: TIMIT

A standard speech benchmark for clean speech recognition.

Method	Phone Error Rate%
NN (6 layers) (Mohamed et al., 2010)	23.4
Dropout NN (6 layers)	21.8
DBN-pretrained NN (4 layers)	22.7
DBN-pretrained NN (6 layers) (Mohamed et al., 2010)	22.4
DBN-pretrained NN (8 layers) (Mohamed et al., 2010)	20.7
mcRBM-DBN-pretrained NN (5 layers) (Dahl et al., 2010)	20.5
DBN-pretrained NN (4 layers) + dropout	**19.7**
DBN-pretrained NN (8 layers) + dropout	**19.7**

Useful Links

Theano: Python-based learning library

Main page: http://deeplearning.net/software/theano/

Theano Tutorial: http://deeplearning.net/software/theano/tutorial/

Deep Learning Tutorial: http://deeplearning.net/tutorial/

Torch7: learning library that supports neural net training

Main page: http://www.torch.ch

Tutorial: http://code.cogbits.com/wiki/doku.php

OverFeat: Object Recognizer, Feature Extractor

URL: http://cilvr.nyu.edu/doku.php?id=software:overfeat:start

EBLearn: C++ Library with convnet support by P. Sermanet

Main page: http://eblearn.sourceforge.net/

Links

Yann LeCun: Google+ profile

URL: https://plus.google.com/+YannLeCunPhD

(2013) Deep Learning Tutorial, ICML 2013

PDF: http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf

Online video (Part 1): http://techtalks.tv/talks/deep-learning/58122/

Online video (Part 2): http://techtalks.tv/talks/energy-based-unsupervised-learning/58128/

Geoffrey E. Hinton: Home page (University of Toronto)

URL: http://www.cs.toronto.edu/~hinton/

(2012) Brains, Sex and Machine Learning (Google Tech Talks)

Online video: http://youtu.be/DleXA5ADG78

Links

Yoshua Bengio: Home page (Université de Montréal)

URL: http://www.iro.umontreal.ca/~bengioy/

(2013) UCL Masterclass lectures

URL: http://www.csml.ucl.ac.uk/events/series/15

1. Deep Learning of Representations (video) (pdf)

2. Non-local Manifold Learning by Regularized Auto-encoders (video) (pdf)

3. Generative Stochastic Networks: How to Get Rid of Approximate Inference over Latent Variables (video) (pdf)

Links

Juergen Schmidhuber: Home page (IDSIA)

URL: http://www.idsia.ch/~juergen/

Google+ profile: https://plus.google.com/100849856540000067209/

(2014) 1st Züri Machine Learning Meetup

URL: http://www.meetup.com/Zurich-Machine-Learning/events/166282162/
Deep Learning (video) (pdf)