HOW TO TRAIN SPAUN'S VISUAL SYSTEM

Nengo Vision Tutorial

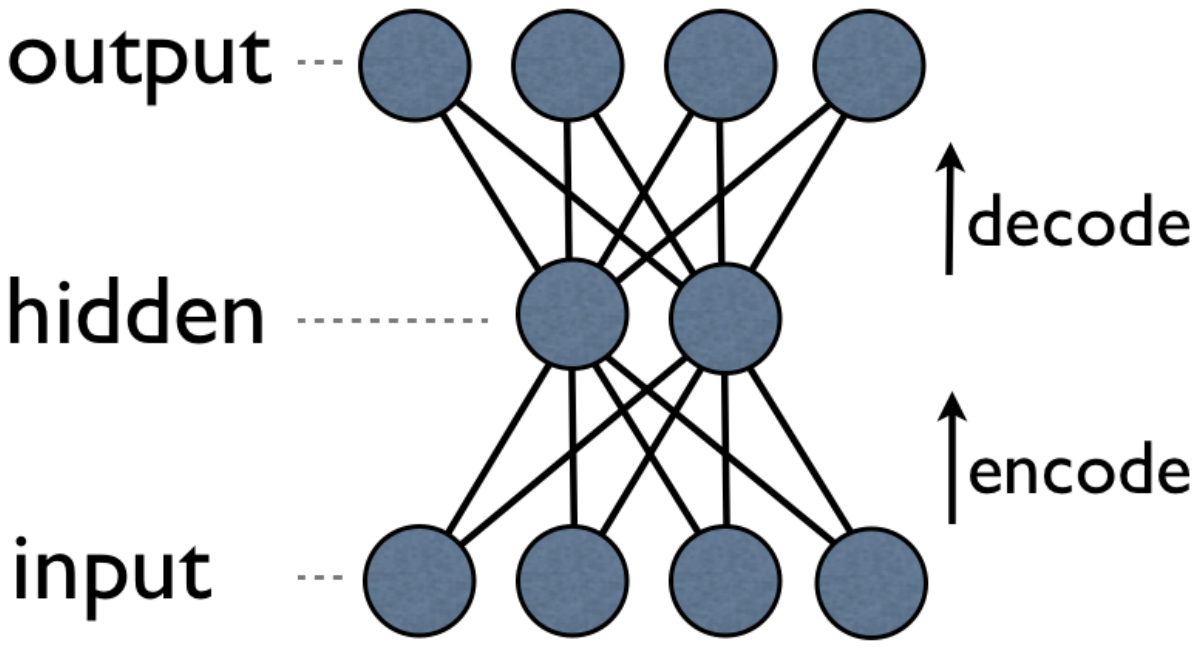
Eric Hunsberger

Monday, June 15, 2015

Autoencoders

Autoencoder

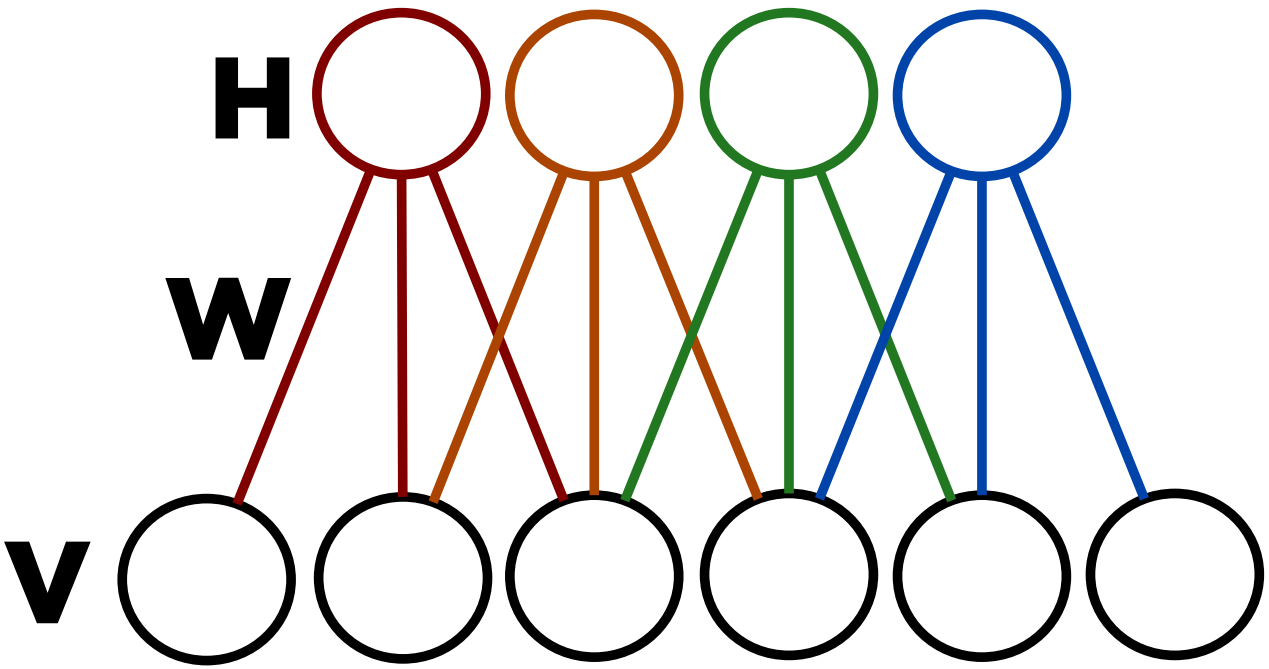
An artifical neural network with one hidden layer that learns to reconstruct its input in its output layer. Often used for dimensionality reduction.



RBMs

Restricted Bolzmann Machine (RBM)

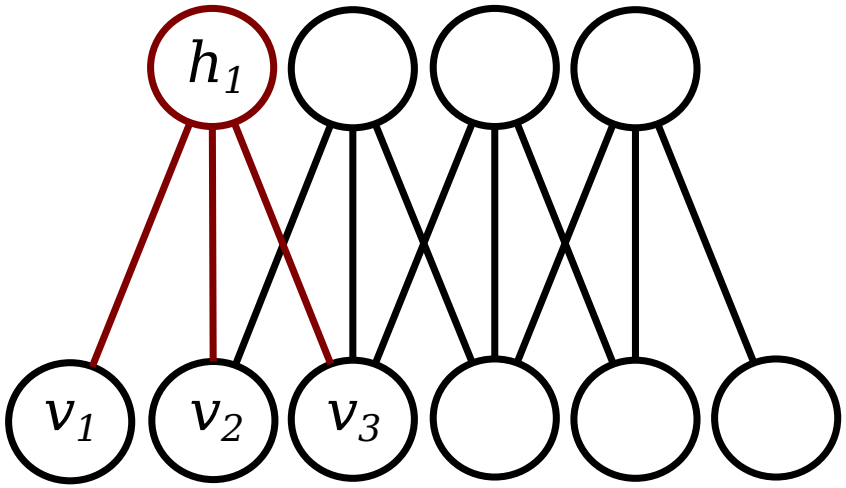
Two-layer generative neural network that learns a statistical model of the training data



RBMs

Representational: can calculate hidden nodes (encoding) from visual nodes (image)

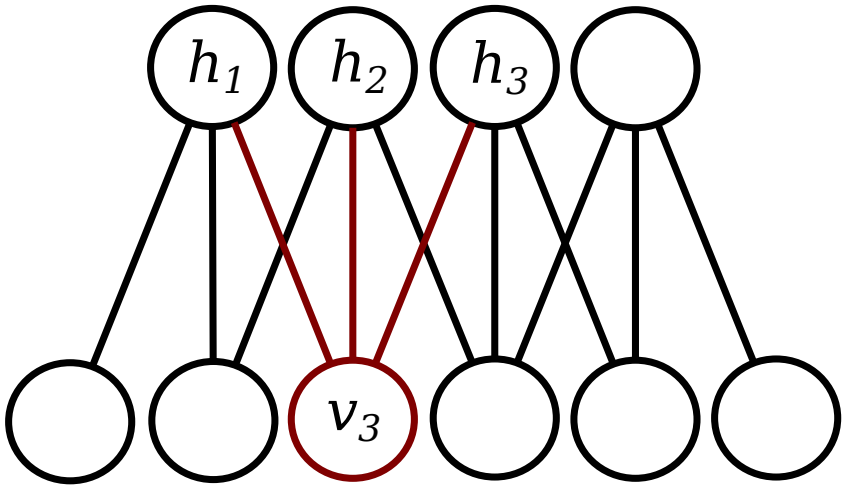
$$h_j = f\left(\sum_i w_{ij} v_i + c_j\right)$$



RBMs

Generative: can calculate visual nodes (image) from hidden nodes (encoding)

$$v_i = f\left(\sum_j w_{ij} h_j + b_i\right)$$



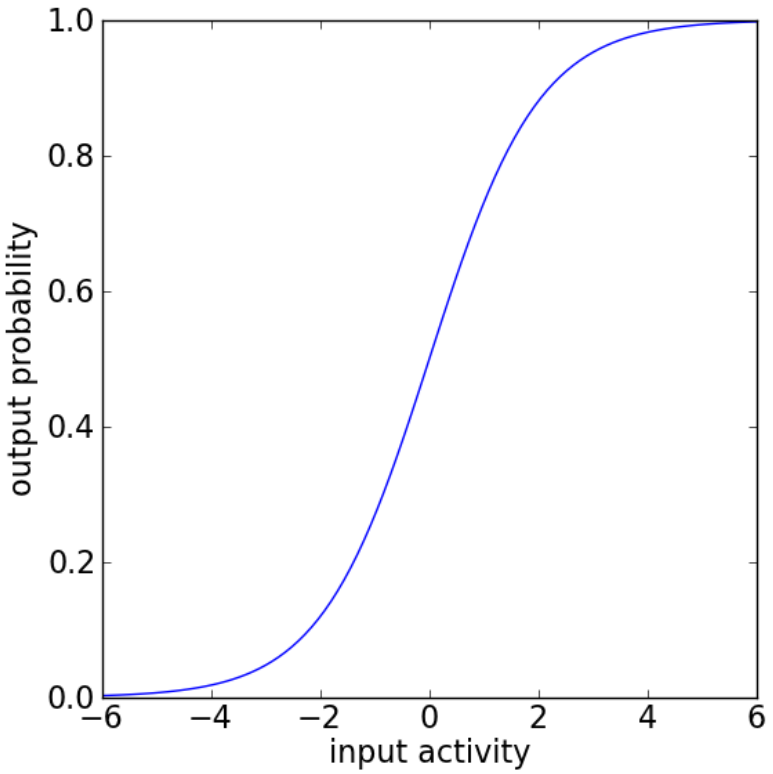
RBMs

Nonlinear: The logistic sigmoid function is often used for the neural nonlinearity

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

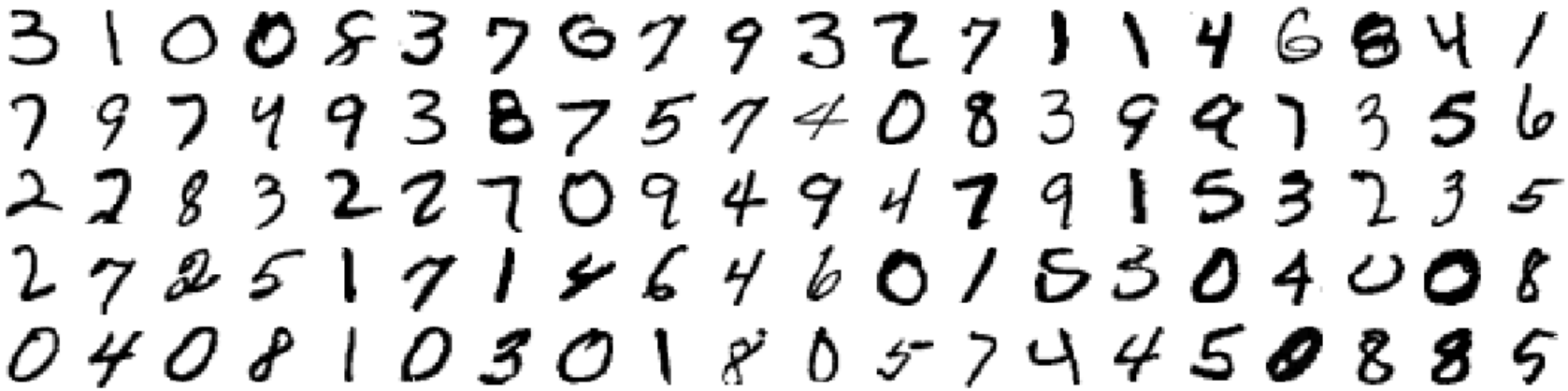
$$h_j = \sigma\left(\sum_i w_{ij} v_i + c_j\right)$$

$$v_i = \sigma\left(\sum_j w_{ij} h_j + b_i\right)$$



RBM Training

- Unsupervised training on a set of images
- Form a generative statistical model of the images
- We will use the MNIST dataset: 60000 handwritten digits



RBM Training

Training an RBM corresponds to minimizing an energy function

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T W \mathbf{h} - \mathbf{b}^T v - \mathbf{c}^T h$$

The energy function gives the probability of the visual and hidden nodes being active

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

where the *partition function* is given by

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

RBM Training

The log probability is given by

$$\log p(\mathbf{v}, \mathbf{h}) = -E(\mathbf{v}, \mathbf{h}) - \log Z$$

Taking the derivative w.r.t. W_{ij} we get

$$\begin{aligned}\frac{\partial \log p(\mathbf{v}, \mathbf{h})}{\partial W_{ij}} &= v_i h_j - \frac{1}{Z} \sum_{\mathbf{v},\mathbf{h}} v_i h_j e^{-E(\mathbf{v},\mathbf{h})} \\ &= v_i h_j - \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v}, \mathbf{h}) v_i h_j\end{aligned}$$

Contrastive Divergence

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

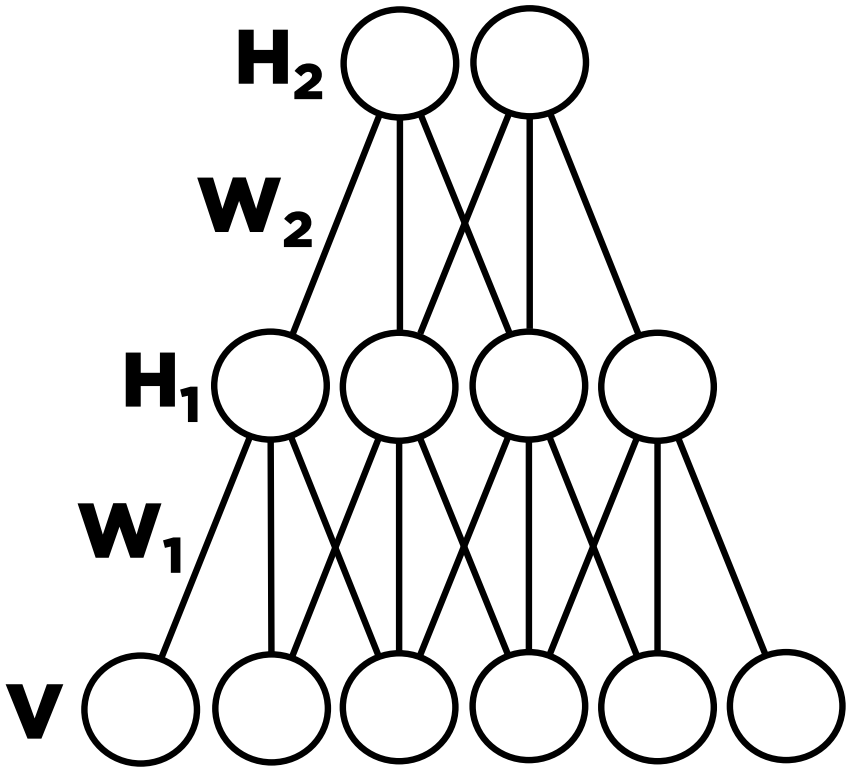
1. Compute and sample hidden layer
2. Re-compute and sample visual layer
3. Re-compute and sample hidden layer
4. Repeat steps 2 and 3 (Gibbs sampling)
5. Update weights

DBNs

Deep Belief Network (DBN)

A stack of RBMs, such that the output of one RBM forms the input to the next

- DBNs learn "features on features"
- High-level features encode (partly) for category
- Some categories require supervised training



Deep autoencoders and classifiers

- Just stacking RBMs means that the layers might not work together as well as they could
- Training the whole network as an autoencoder or classifier improves overall performance
- This is done using backpropagation (easy with Theano)
  - Compute network forward, propagate errors backward
- Why do pretraining at all?

Backpropagation

$$y_j = f(\hat{y}_j) \quad \hat{y}_j = \sum_i x_i w_{ij} + c_j$$

$$z_k = g(\hat{z}_k) \quad \hat{z}_k = \sum_j y_j v_{jk} + b_k$$

$$r = \frac{1}{2} \sum_k (z_k - t_k)^2$$

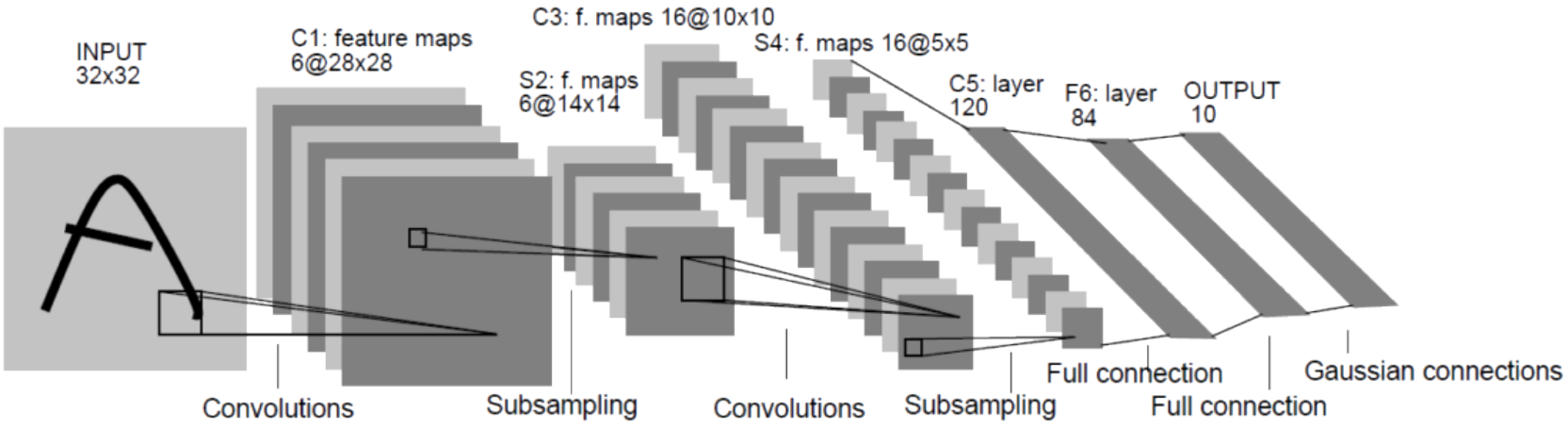
$$\frac{dr}{dv_{jk}} = \frac{\partial r}{\partial z_k} \frac{\partial z_k}{\partial v_{jk}} = (z_k - t_k) g'(\hat{z}_k) y_j$$

$$\frac{dr}{dv_{jk}} = \sum_k \frac{\partial r}{\partial z_k} \frac{\partial z_k}{\partial y_j} \frac{\partial y_j}{\partial w_{ij}}$$

$$= \sum_k (z_k - t_k) g'(\hat{z}_k) v_{jk} f'(\hat{y}_j) x_i$$

Convolutional networks

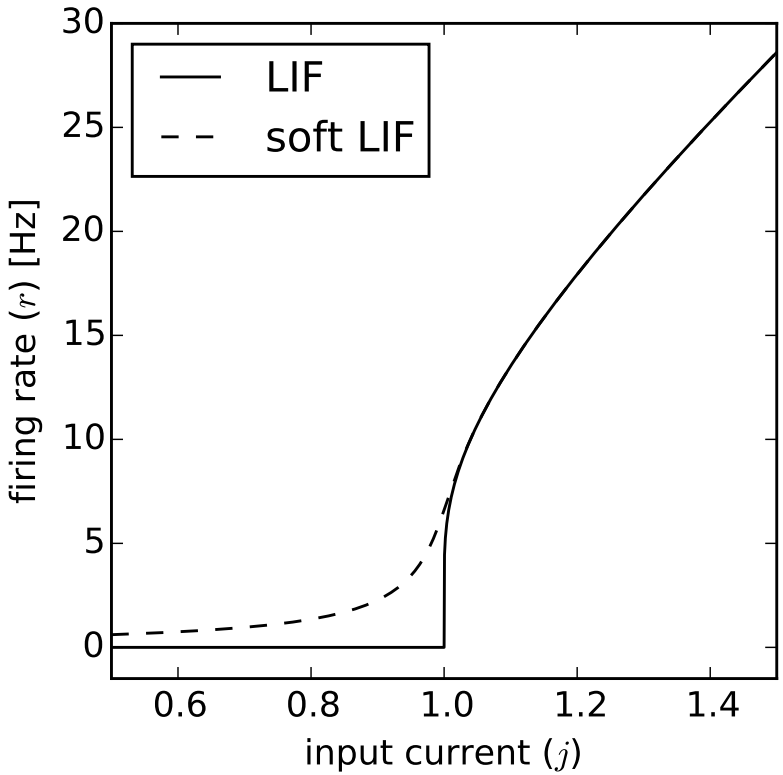
- Use the same filters at each location
- Fewer parameters means learning is faster
- Successful on much harder and larger datasets (CIFAR-10, ImageNet)

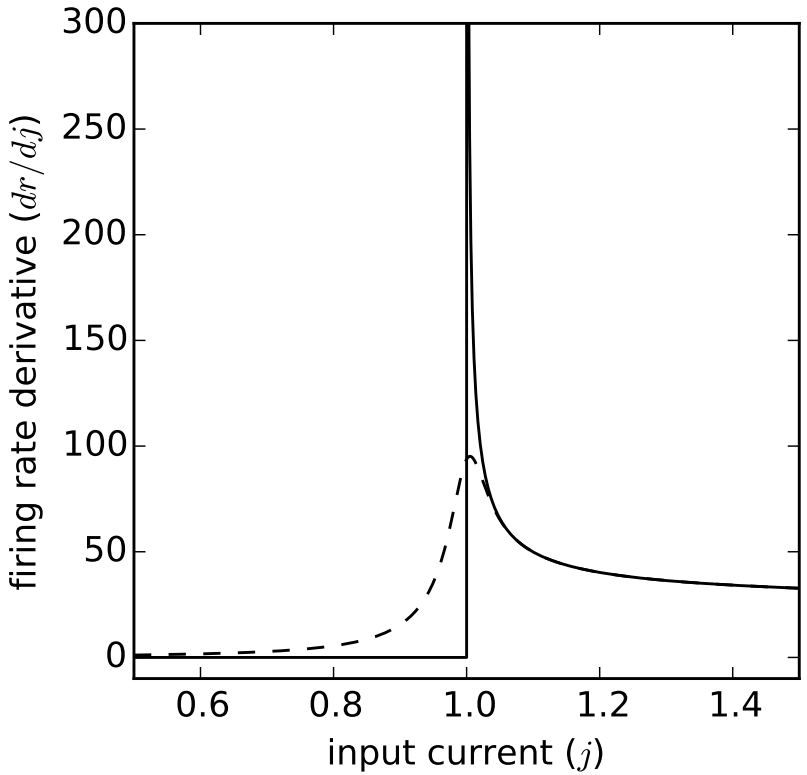


LeNet5, a type of Convolutional Neural Network

Training with LIF functions

- Substitute more biologically-plausible functions for sigmoid
- LIF response function has infinite derivative
- Soft LIF neuron has bounded derivative, and can be made arbitrarily close to LIF





Training with noise

- Filtered spikes result in variability in the network
- Noise on the neuron output during training simulates this variability
- Reduces error when network is transferred to spiking neurons