Convolutional Networks

Lecture 03

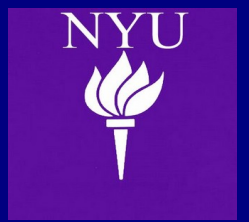
Yann Le Cun

Facebook AI Research,

Center for Data Science, NYU

Courant Institute of Mathematical Sciences, NYU

http://yann.lecun.com

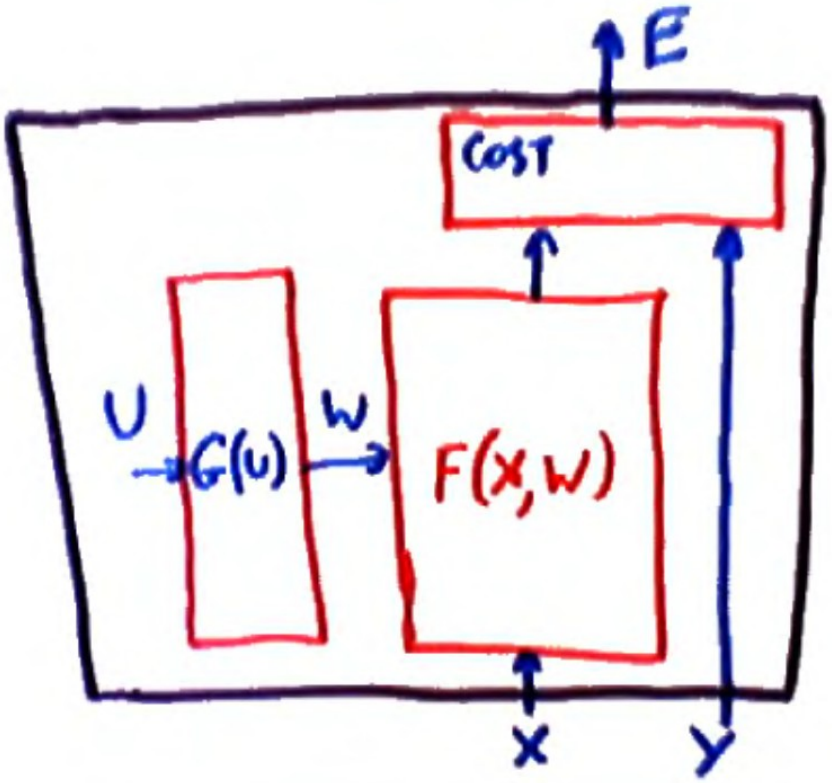



Parameter Transforms
In
Deep Networks

Parameter Space Transform

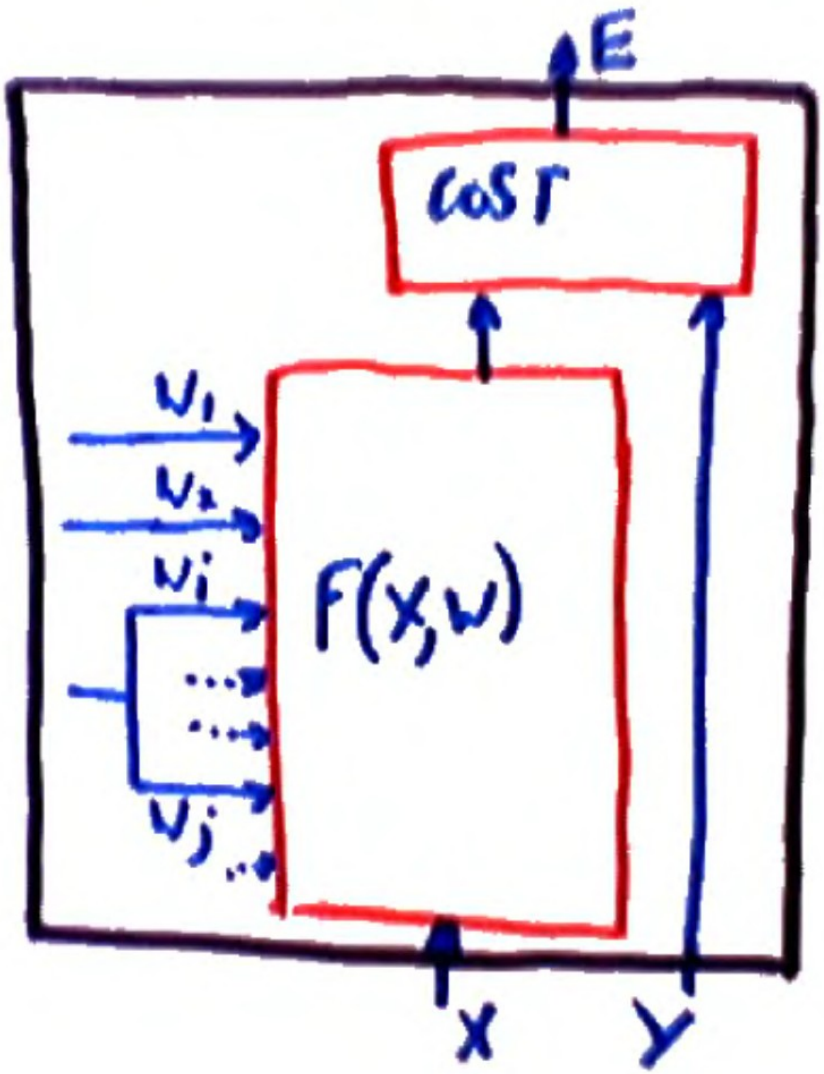
Reparameterizing the function by transforming the space

$$E(Y, X, W) \rightarrow E(Y, X, G(U))$$



- gradient descent in U space:
  $U \leftarrow U - \eta \frac{\partial G}{\partial U}' \frac{\partial E(Y,X,W)}{\partial W}'$
- equivalent to the following algorithm in W space: $W \leftarrow W - \eta \frac{\partial G}{\partial U} \frac{\partial G}{\partial U}' \frac{\partial E(Y,X,W)}{\partial W}'$
- dimensions: $[N_w \times N_u][N_u \times N_w][N_w]$

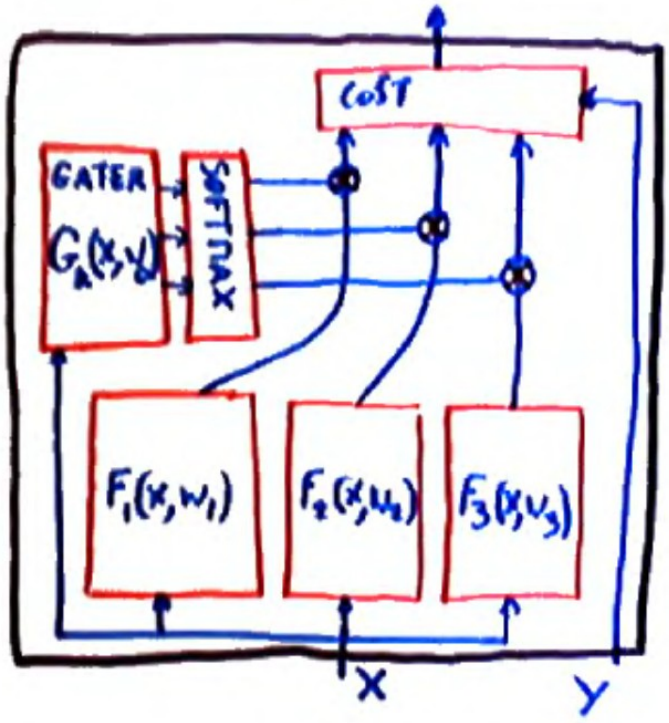
Parameter Space Transform: Weight Sharing



- A single parameter is replicated multiple times in a machine
- $E(Y, X, w_1, \ldots, w_i, \ldots, w_j, \ldots) \rightarrow E(Y, X, w_1, \ldots, u_k, \ldots, u_k, \ldots)$
- gradient: $\frac{\partial E()}{\partial u_k} = \frac{\partial E()}{\partial w_i} + \frac{\partial E()}{\partial w_j}$
- w_i and w_j are tied, or equivalently, u_k is shared between two locations.

Mixture of Experts

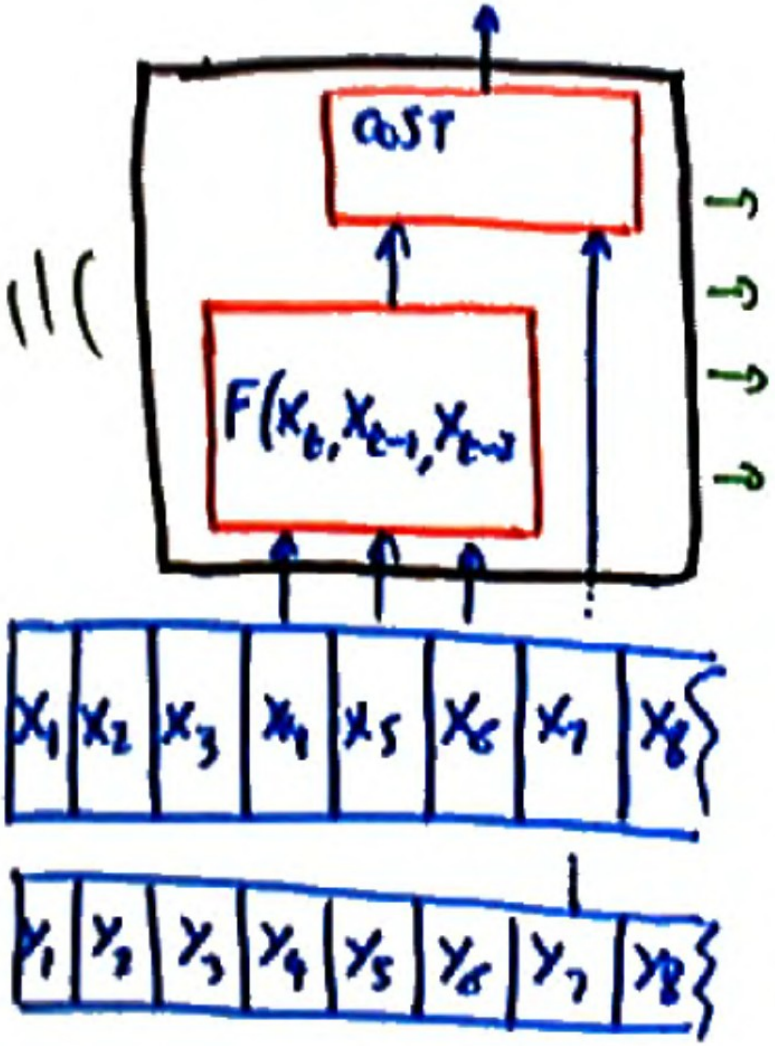
Sometimes, the function to be learned is consistent in restricted domains of the input space, but globally inconsistent. Example: piecewise linearly separable function.



- Solution: a machine composed of several "experts" that are specialized on subdomains of the input space.
- The output is a weighted combination of the outputs of each expert. The weights are produced by a "gater" network that identifies which subdomain the input vector is in.
- $F(X,W) = \sum_k u_k F^k(X,W^k)$ with
  $u_k = \frac{\exp(-\beta G_k(X,W^0))}{\sum_k \exp(-\beta G_k(X,W^0))}$
- the expert weights u_k are obtained by softmax-ing the outputs of the gater.
- example: the two experts are linear regressors, the gater is a logistic regressor.

Time-Delayed Inputs

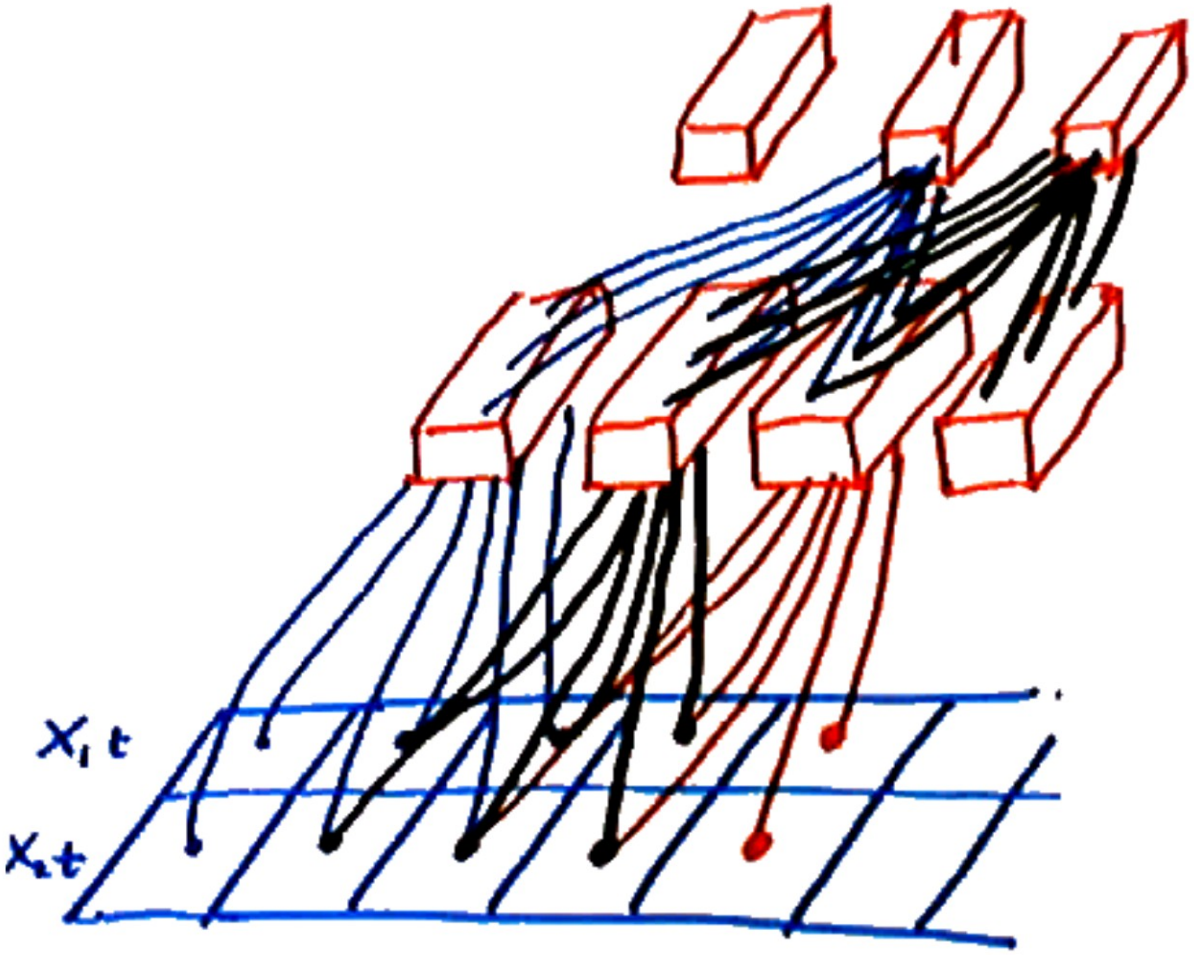
The input is a sequence of vectors X_t.



- simple idea: the machine takes a time window as input
- $R = F(X_t, X_{t-1}, X_{t-2}, W)$
- Examples of use:
  - predict the next sample in a time series (e.g. stock market, water consumption)
  - predict the next character or word in a text
  - classify an intron/exon transition in a DNA sequence

Example: 1D (Temporal) convolutional net

- 1D (Temporal) ConvNet, aka Timed-Delay Neural Nets
- Groups of units are replicated at each time step.
- Replicas have identical (shared) weights.




Convolutional Networks
(ConvNet or CNN)

Deep Learning = The Entire Machine is Trainable

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor

Feature Extractor → Trainable Classifier

Mainstream Modern Pattern Recognition: Unsupervised mid-level

Feature Extractor → Mid-Level Features → Trainable Classifier

Deep Learning: Representations are hierarchical and trained

Low-Level Features → Mid-Level Features → High-Level Features → Trainable Classifier

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 2

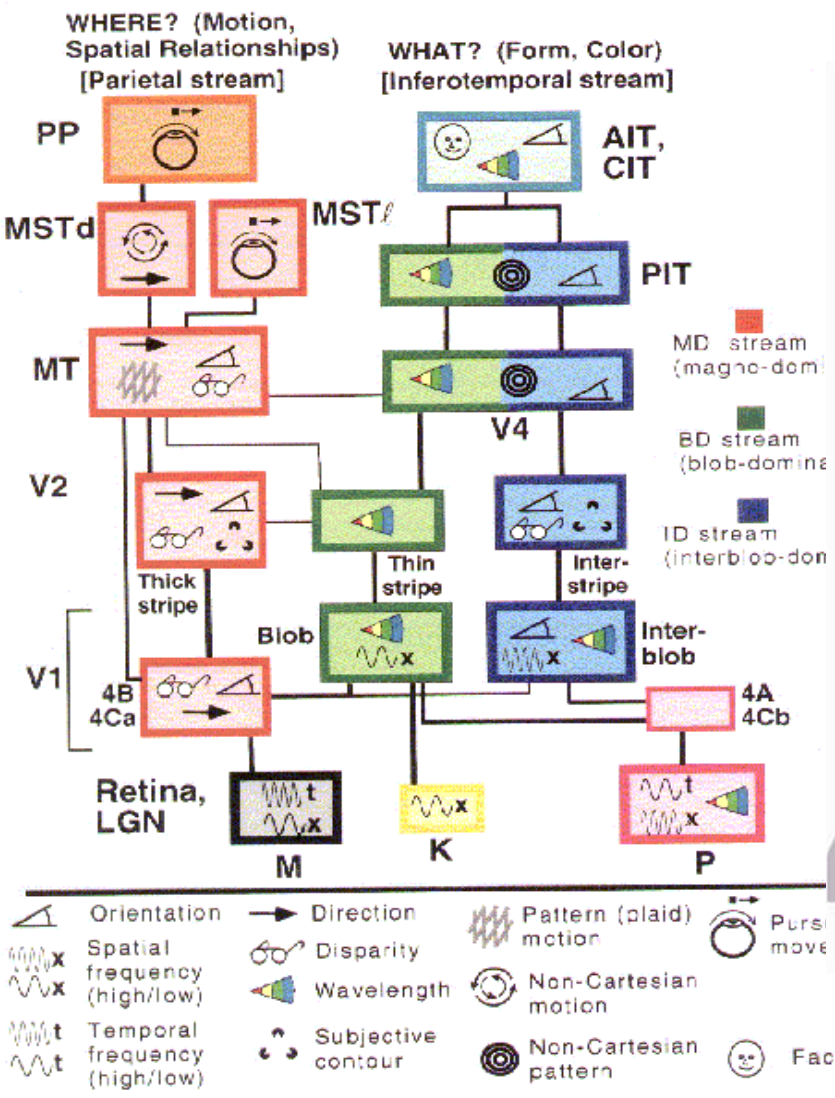
Deep Learning = Learning Hierarchical Representations

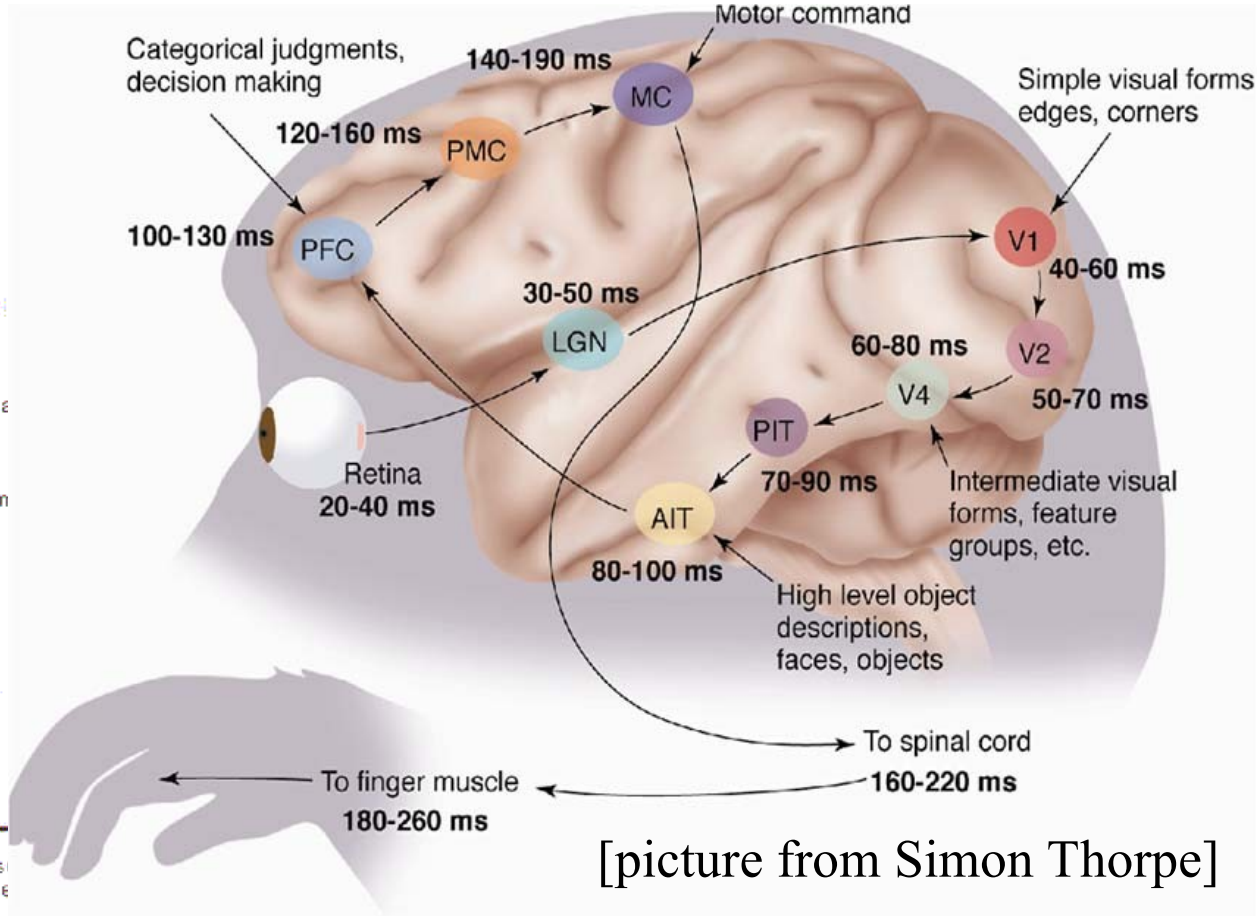
- It's deep if it has more than one stage of non-linear feature transformation

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2

How does the brain interprets images?

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



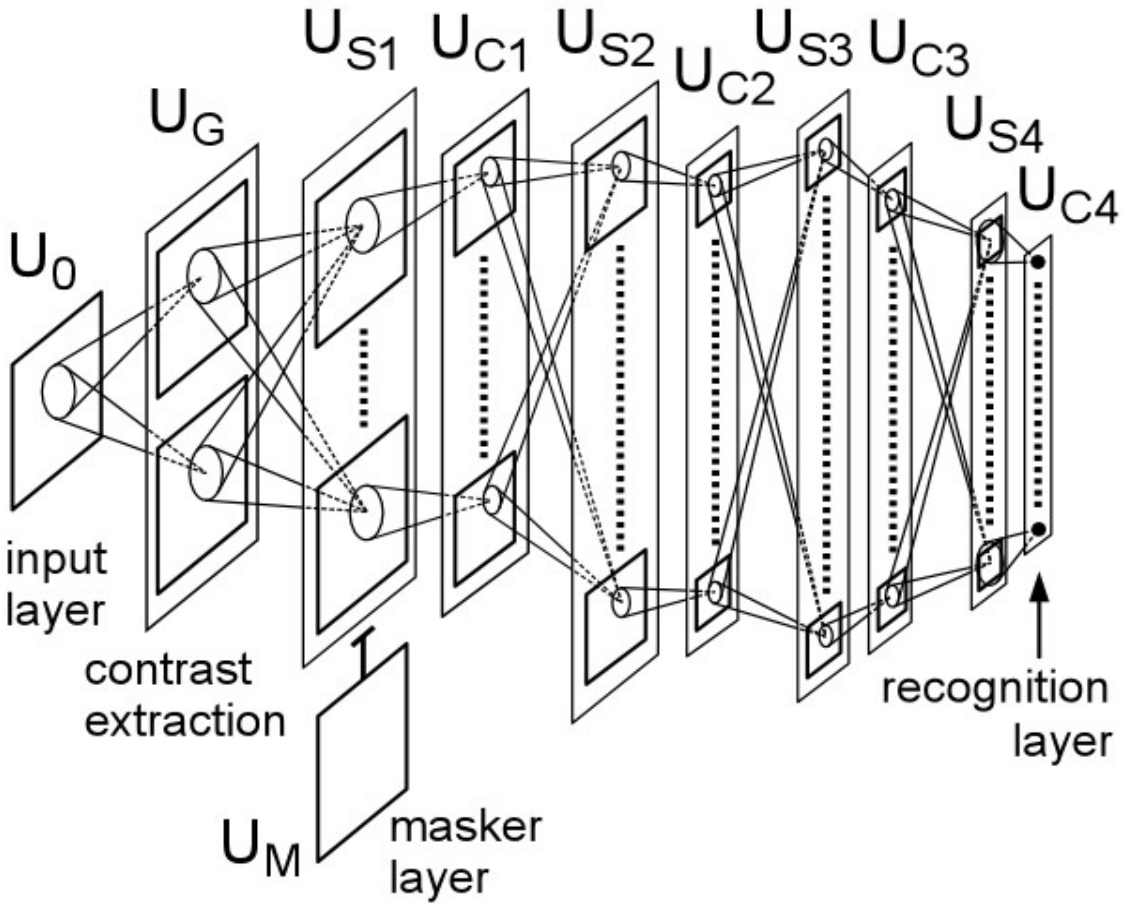


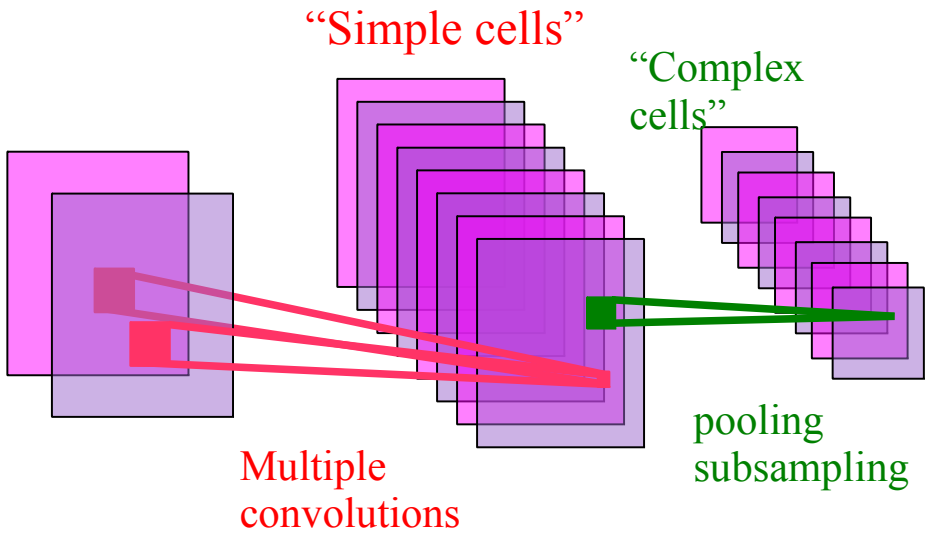
[picture from Simon Thorpe]

[Gallant & Van Essen]

Hubel & Wiesel's Model of the Architecture of the Visual Cortex

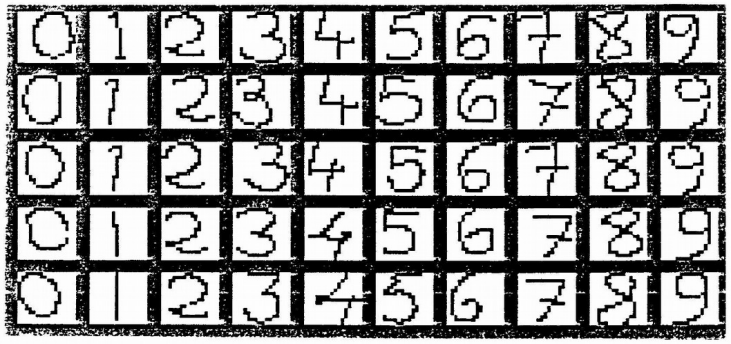
- **[Hubel & Wiesel 1962]:**
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



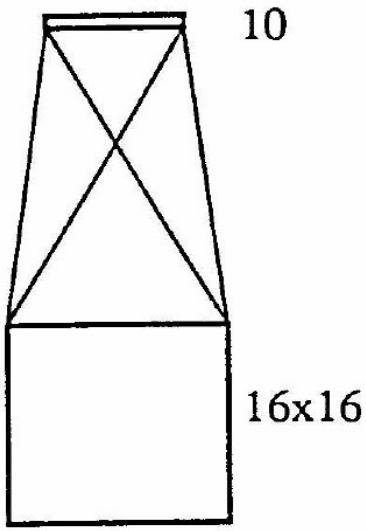


[Fukushima 1982][LeCun 1989, 1998],[Riesenhuber 1999].....

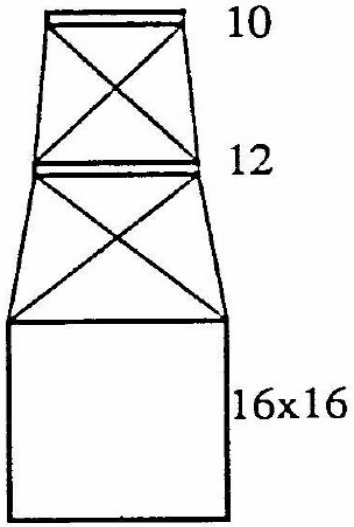
First ConvNets (U Toronto)[LeCun 88, 89]



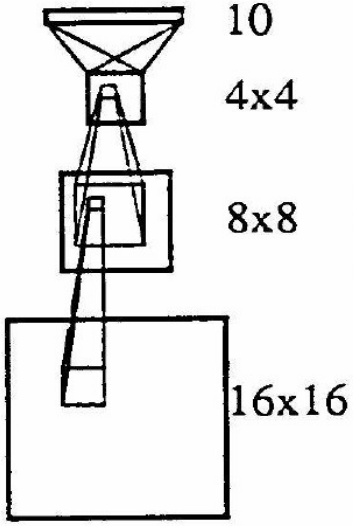
Trained with Backprop. 320 examples.



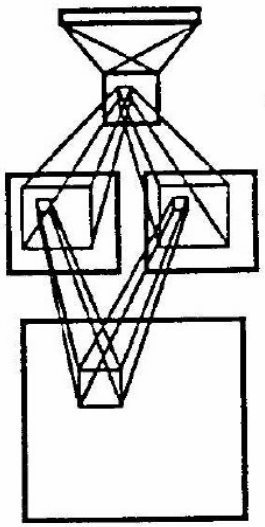
Single layer



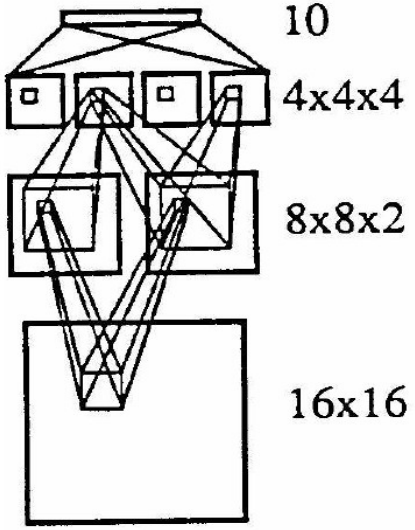
Two layers FC



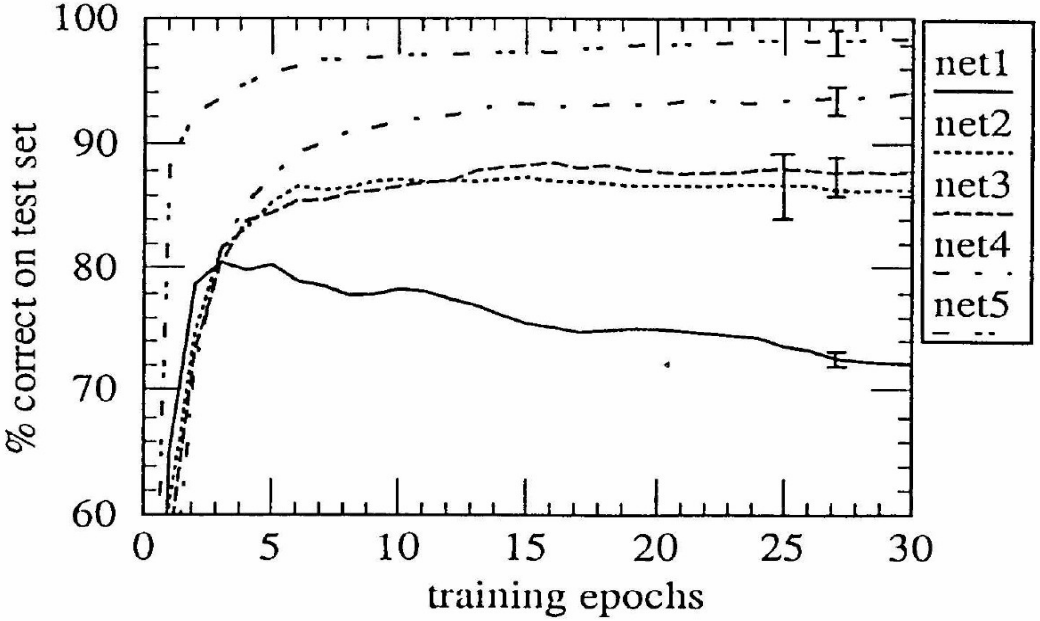
locally connected



Shared weights



Shared weights



- Convolutions with stride (subsampling)
- No separate pooling layers

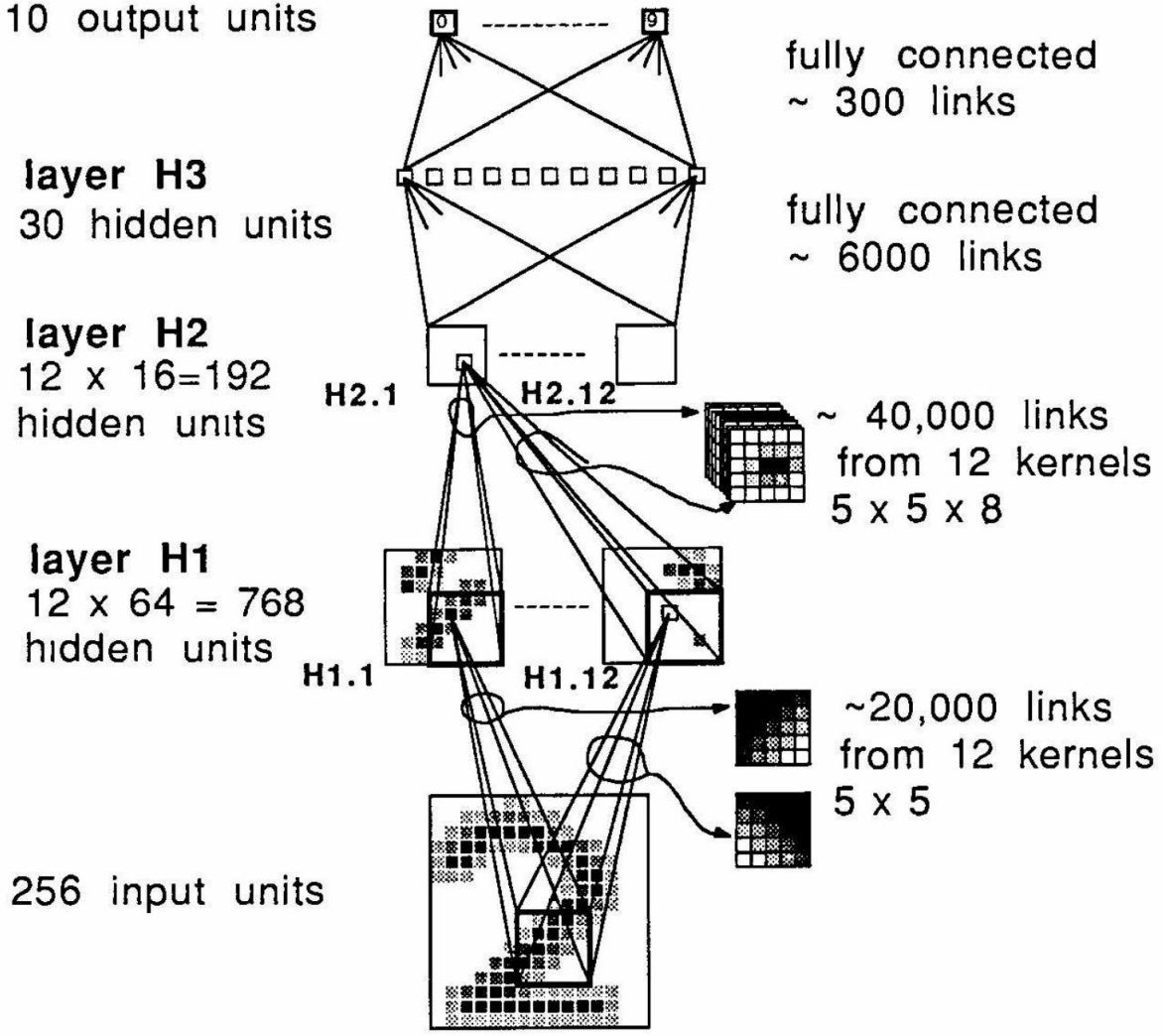
network architecture	links	weights	performance
single layer network	2570	2570	80 %
two layer network	3240	3240	87 %
locally connected	1226	1226	88.5 %
constrained network	2266	1132	94 %
constrained network 2	5194	1060	98.4 %

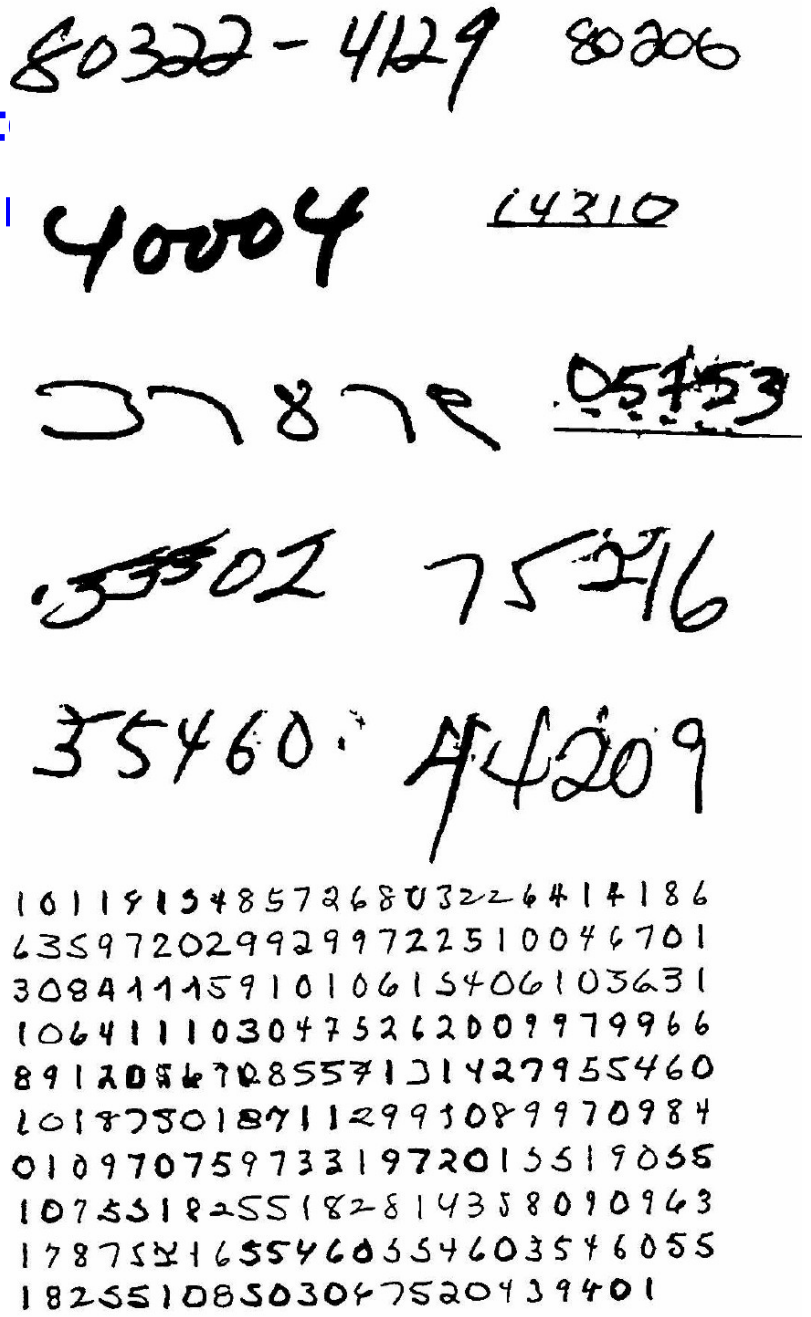
First “Real” ConvNets at Bell Labs [LeCun et al 89]

Trained with Backprop.

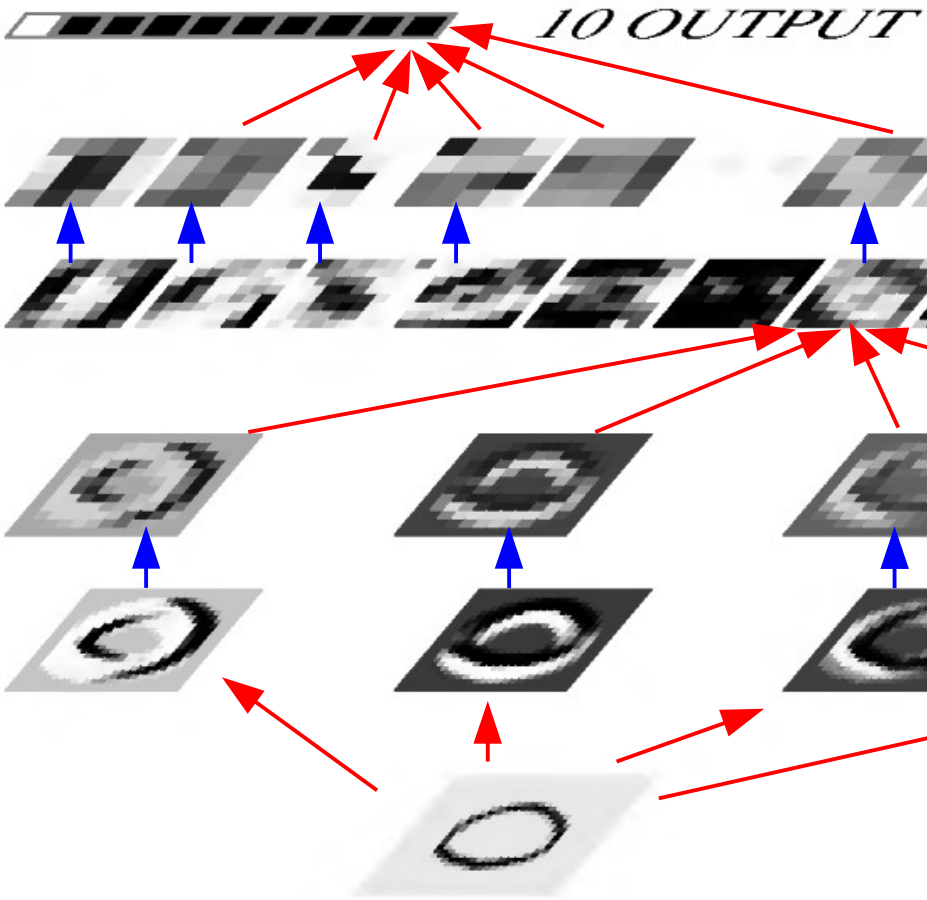
USPS Zipcode digits: 7300 training, 2000 t

Convolution with stride. No separate pooli





Convolutional Network Architecture



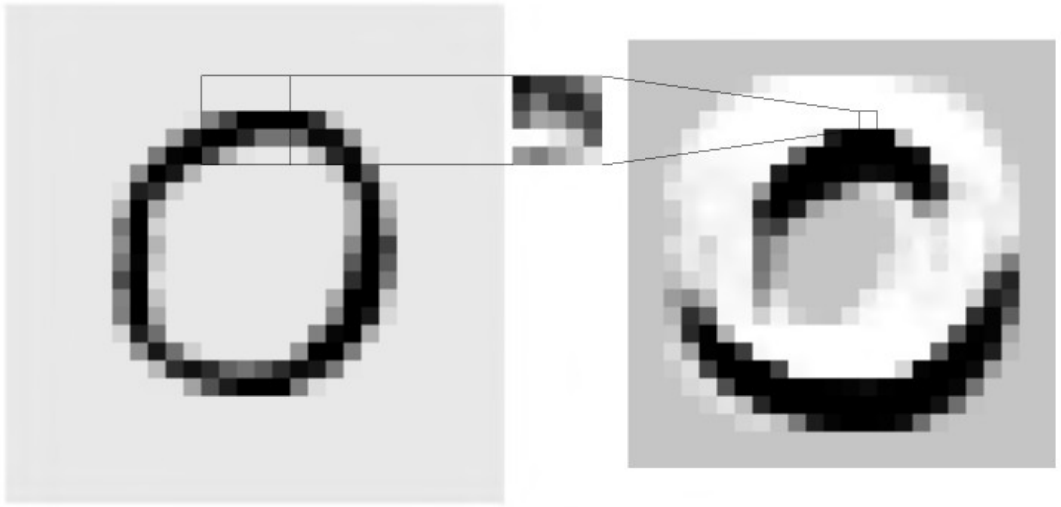
Filter Bank +non-linearity

Pooling

Filter Bank +non-linearity

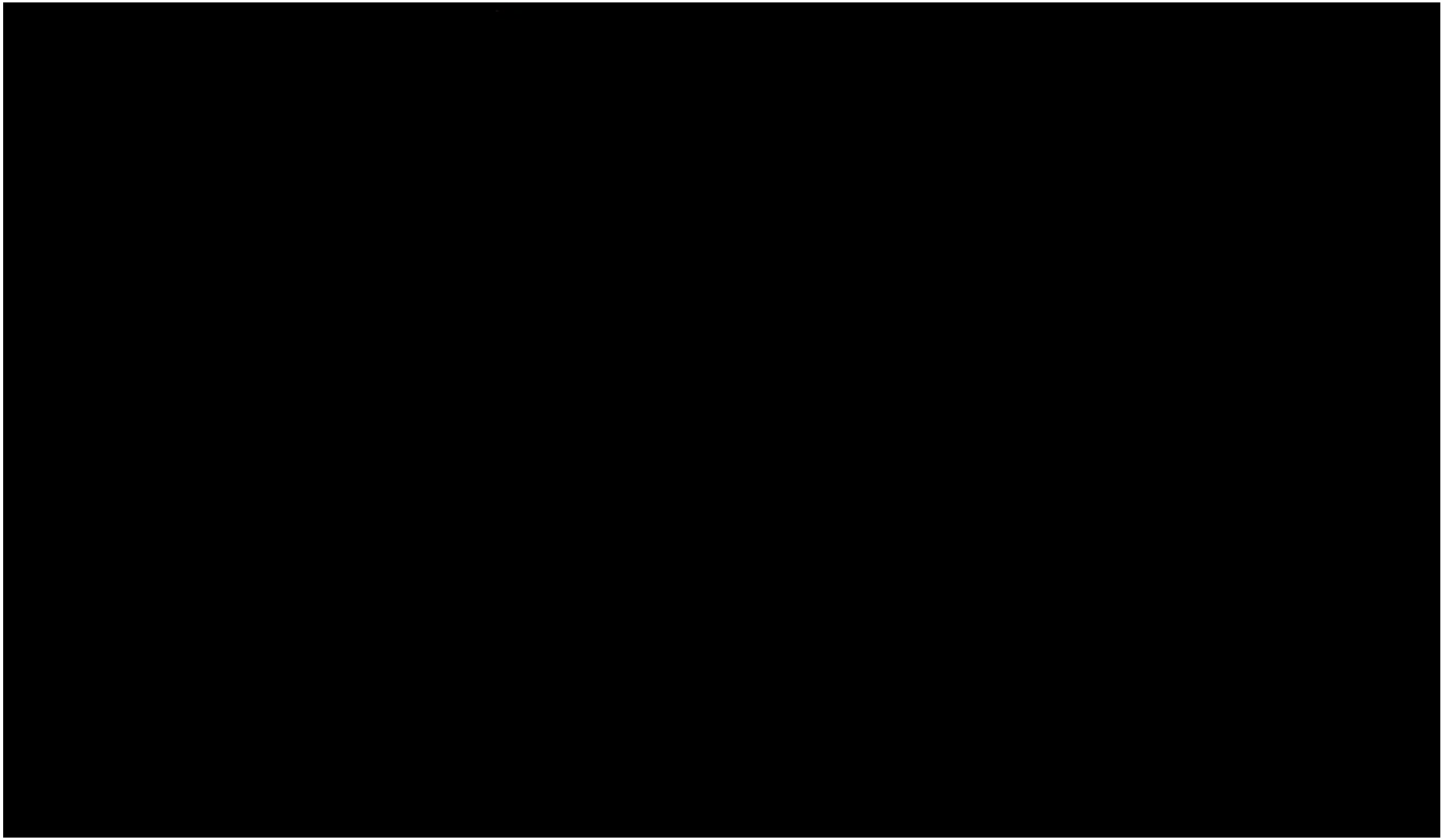
Pooling

Filter Bank +non-linearity



[LeCun et al. NIPS 1989]

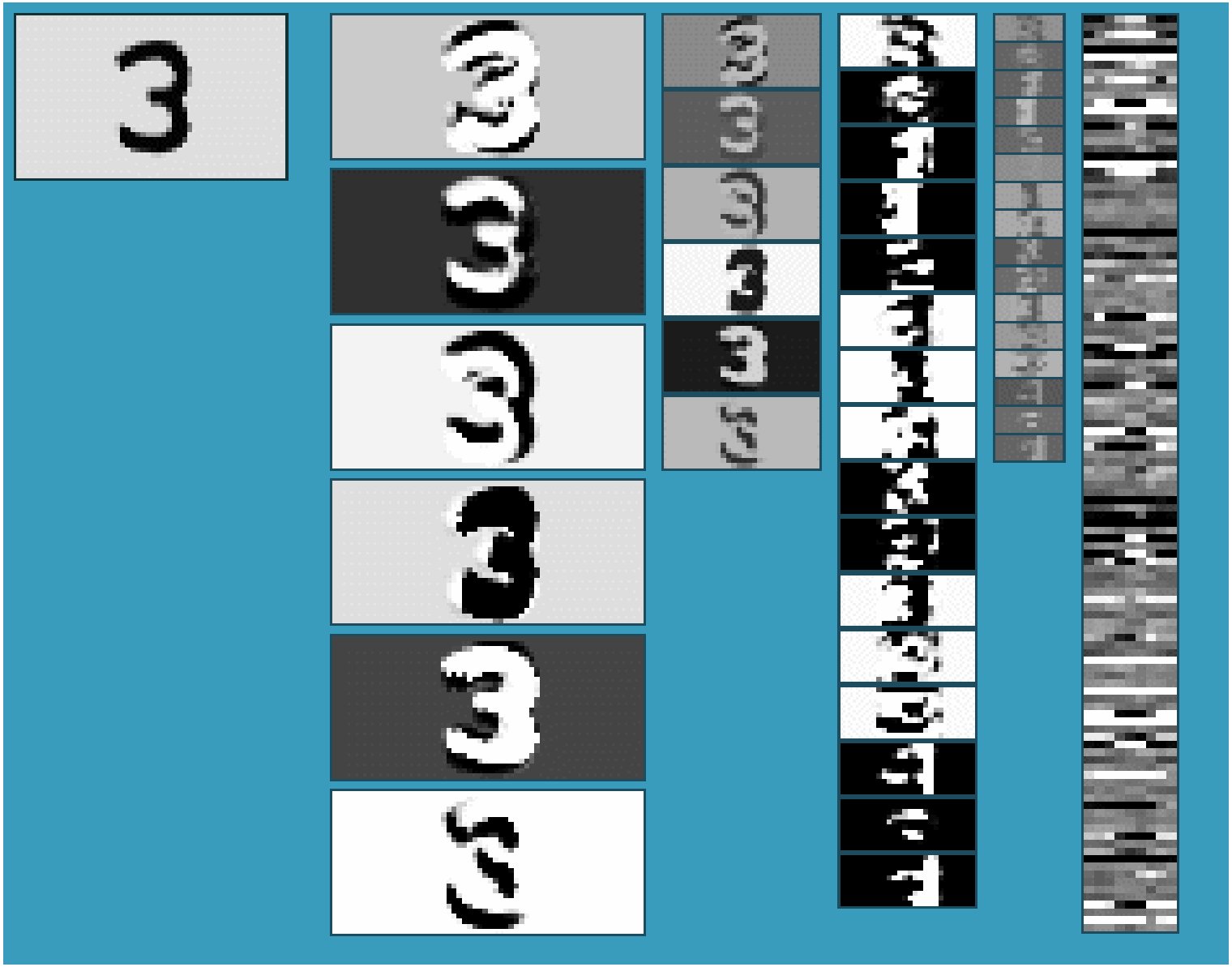
Multiple Convolutions



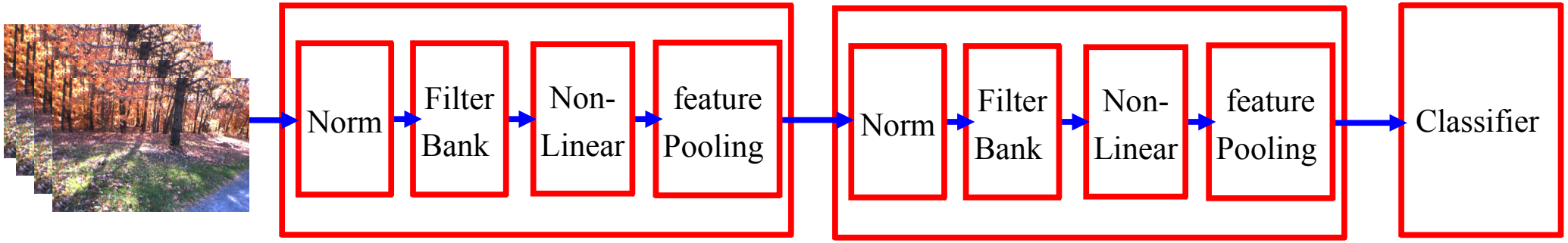
Animation: Andrej Karpathy http://cs231n.github.io/convolutional-networks/

Convolutional Network (vintage 1990)

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

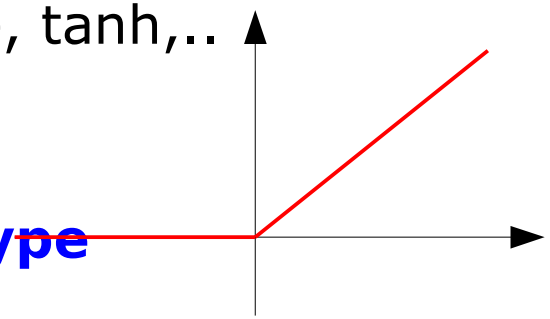


Overall Architecture: multiple stages of Normalization → Filter Bank → Non-Linearity → Pooling



- **Normalization: variation on whitening (optional)**
  - Subtractive: average removal, high pass filtering
  - Divisive: local contrast normalization, variance normalization
- **Filter Bank: dimension expansion, projection on overcomplete basis**
- **Non-Linearity: sparsification, saturation, lateral inhibition....**
  - Rectification (ReLU), Component-wise shrinkage, tanh,..

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



- **Pooling: aggregation over space or feature type**
  - Max, Lp norm, log prob.

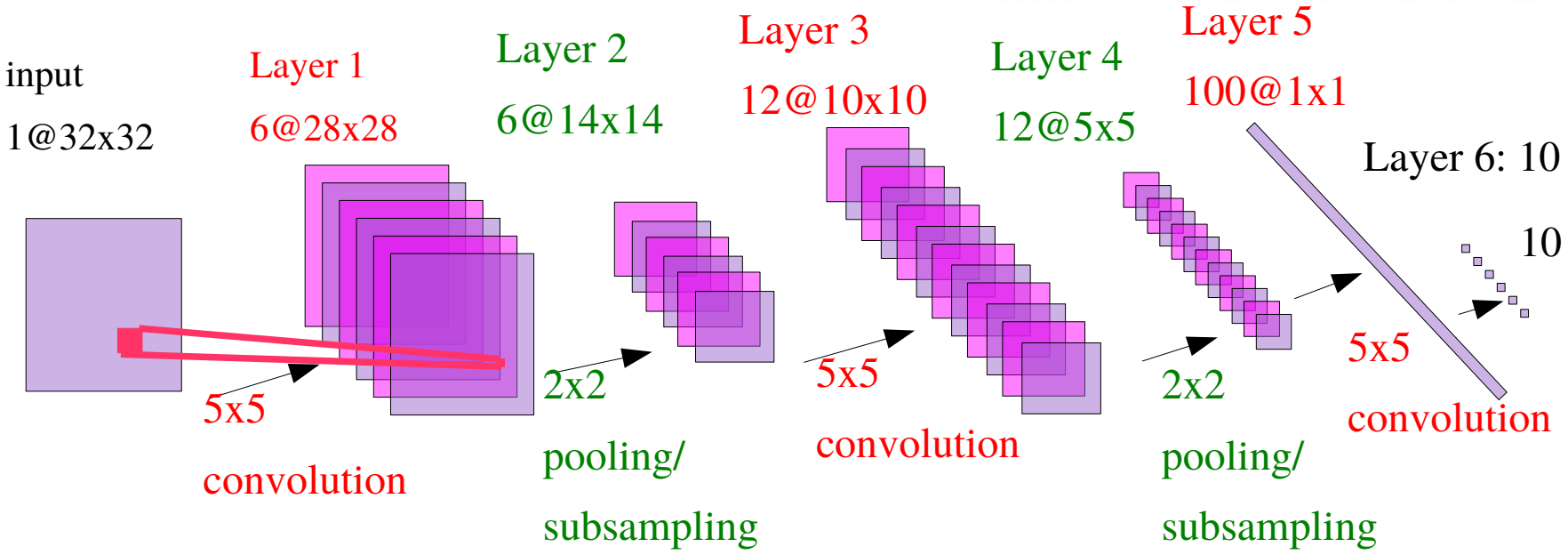
$$MAX: Max_i(X_i); \quad L_p: \sqrt[p]{X_i^p}; \quad PROB: \frac{1}{b}\log\left(\sum_i e^{bX_i}\right)$$

LeNet5

- **Simple ConvNet**
- **for MNIST**
- **[LeCun 1998]**

- **PyTorch code ------->**
  - (slightly different net)

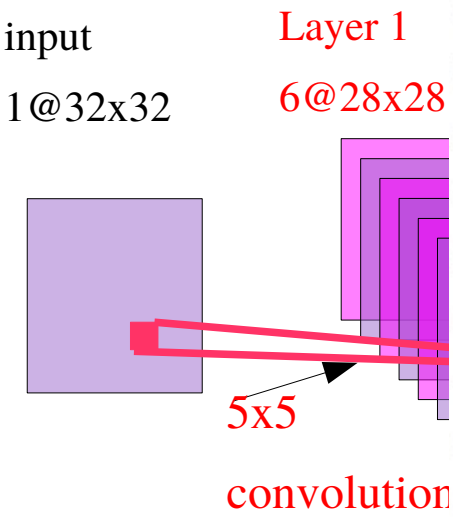
```
10  class Net(nn.Module):
11      def __init__(self):
12          super(Net, self).__init__()
13          self.conv1 = nn.Conv2d(1, 20, 5, 1)
14          self.conv2 = nn.Conv2d(20, 50, 5, 1)
15          self.fc1 = nn.Linear(4*4*50, 500)
16          self.fc2 = nn.Linear(500, 10)
17
18      def forward(self, x):
19          x = F.relu(self.conv1(x))
20          x = F.max_pool2d(x, 2, 2)
21          x = F.relu(self.conv2(x))
22          x = F.max_pool2d(x, 2, 2)
23          x = x.view(-1, 4*4*50)
24          x = F.relu(self.fc1(x))
25          x = self.fc2(x)
26          return F.log_softmax(x, dim=1)
```



LeNet5

- Simple ConvNet
- for MNIST
- [LeCun 1998]

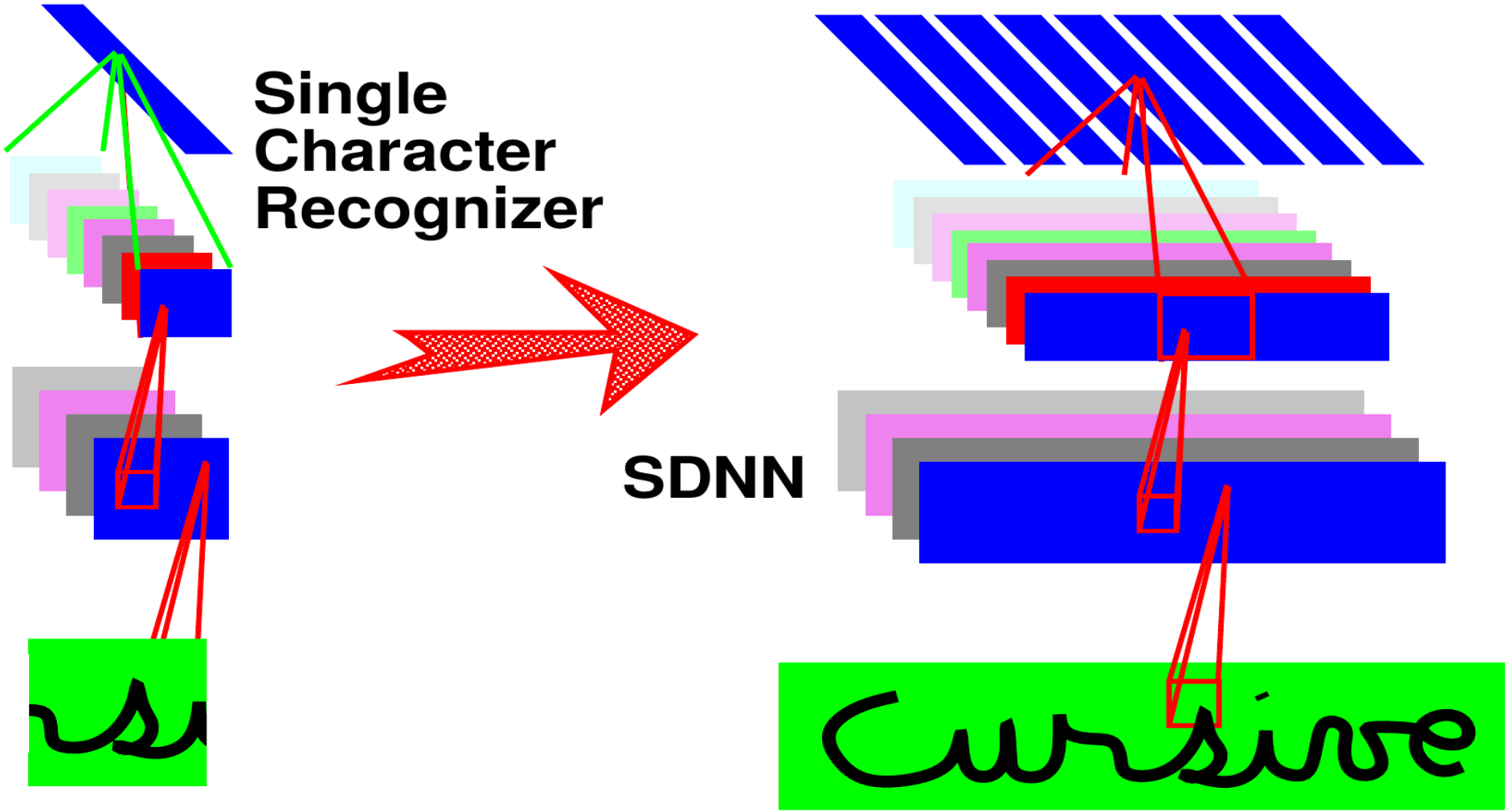
- PyTorch code --→
  github.com/activatedgeek/LeNet-5
- nn.sequential() with ordered dictionary argument.




```
19      def __init__(self):
20          super(LeNet5, self).__init__()
21
22          self.convnet = nn.Sequential(OrderedDict([
23              ('c1', nn.Conv2d(1, 6, kernel_size=(5, 5))),
24              ('relu1', nn.ReLU()),
25              ('s2', nn.MaxPool2d(kernel_size=(2, 2), stride=2)),
26              ('c3', nn.Conv2d(6, 16, kernel_size=(5, 5))),
27              ('relu3', nn.ReLU()),
28              ('s4', nn.MaxPool2d(kernel_size=(2, 2), stride=2)),
29              ('c5', nn.Conv2d(16, 120, kernel_size=(5, 5))),
30              ('relu5', nn.ReLU())
31          ]))
32
33          self.fc = nn.Sequential(OrderedDict([
34              ('f6', nn.Linear(120, 84)),
35              ('relu6', nn.ReLU()),
36              ('f7', nn.Linear(84, 10)),
37              ('sig7', nn.LogSoftmax(dim=-1))
38          ]))
39
40      def forward(self, img):
41          output = self.convnet(img)
42          output = output.view(img.size(0), -1)
43          output = self.fc(output)
44          return output
```

Multiple Character Recognition [Matan et al 1992]

- Every layer is a convolution



Sliding Window ConvNet + Weighted Finite-State Machine

Sliding Window ConvNet + Weighted FSM

What are ConvNets Good For

- **Signals that comes to you in the form of (multidimensional) arrays.**
- **Signals that have strong local correlations**
- **Signals where features can appear anywhere**
- **Signals in which objects are invariant to translations and distortions.**

- **1D ConvNets: sequential signals, text**
  - Text Classification
  - Musical Genre Recognition
  - Acoustic Modeling for Speech Recognition
  - Time-Series Prediction
- **2D ConvNets: images, time-frequency representations (speech and audio)**
  - Object detection, localization, recognition
- **3D ConvNets: video, volumetric images, tomography images**
  - Video recognition / understanding
  - Biomedical image analysis
  - Hyperspectral image analysis