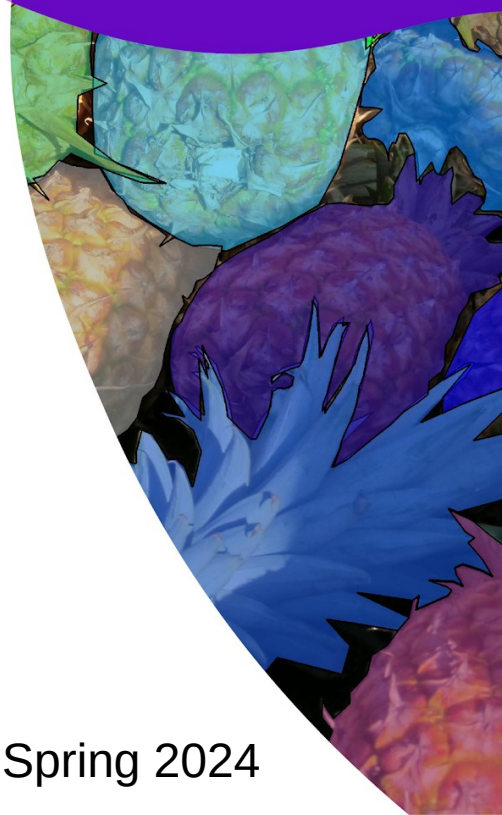
NEW YORK UNIVERSITY

Architectures, Recurrent Nets, Convolutional Nets

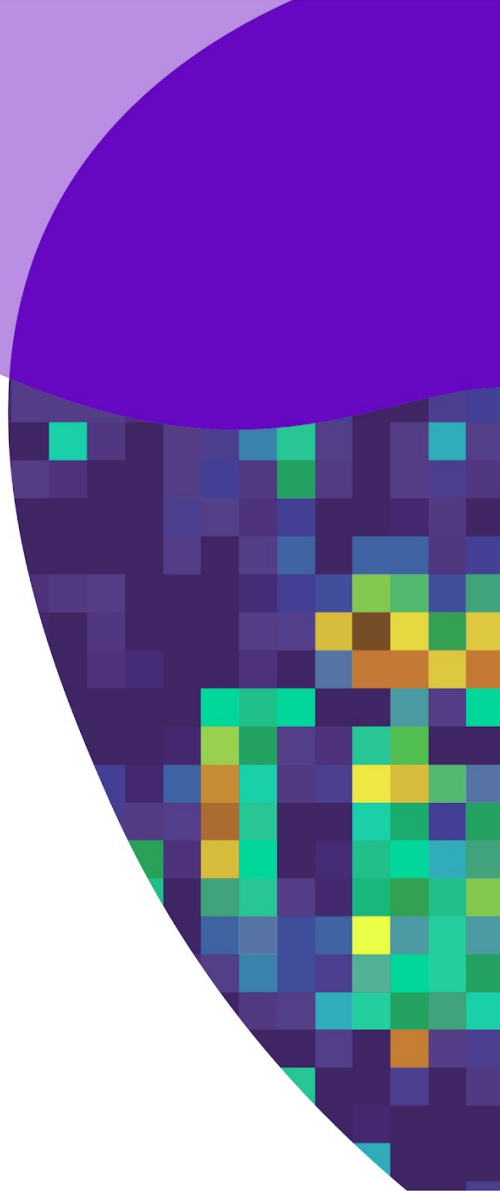
Alfredo Canziani, Mengye Ren, Yann LeCun
NYU - Courant Institute & Center for Data Science



Deep Learning, NYU Spring 2024

Architectures

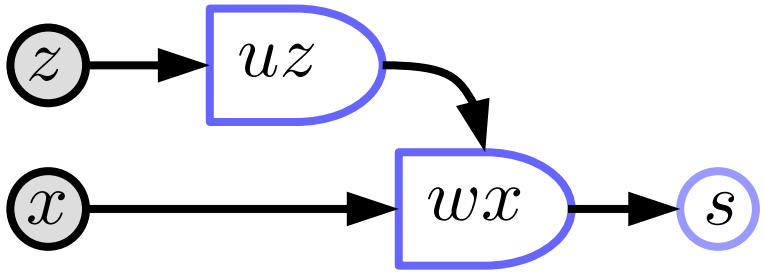
Arranging modules



Multiplicative Modules

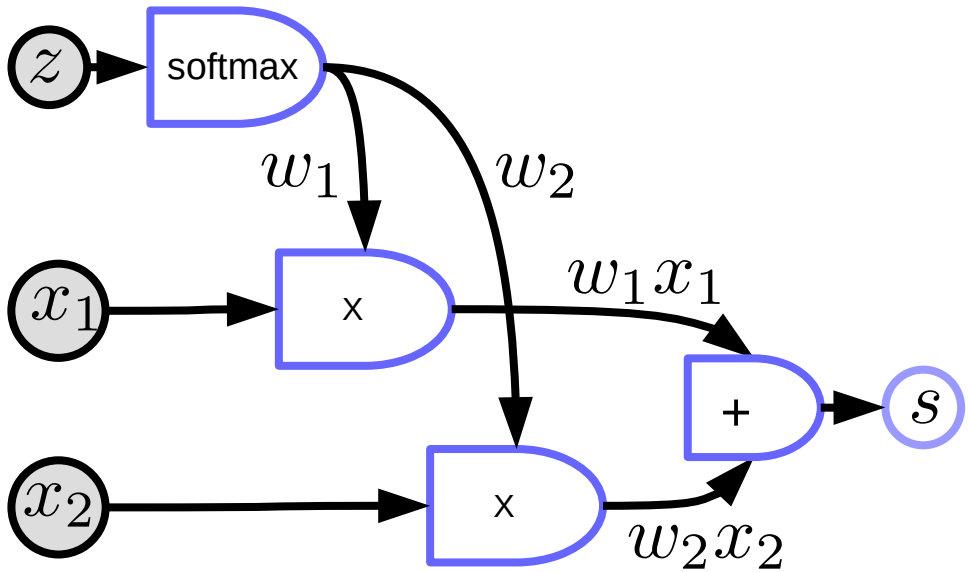
- **Quadratic layer, product units, Sigma-Pi units**

$$s_i = \sum_j w_{ij} x_j \text{ with : } w_{ij} = \sum_k u_{ijk} z_k \text{ ; equiv : } s_i = \sum_{jk} u_{ijk} z_k x_j$$



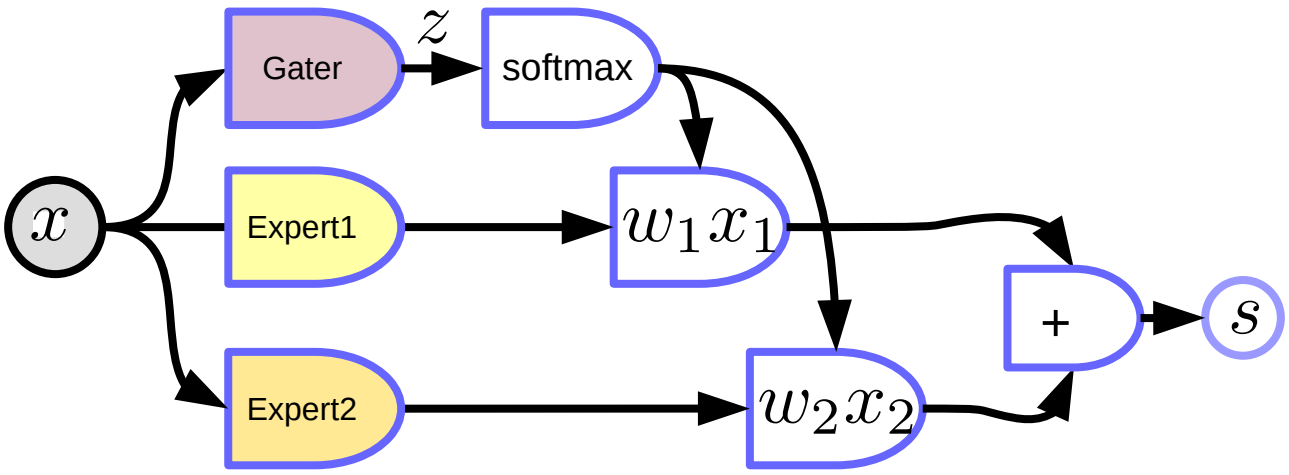
- **Attention module**

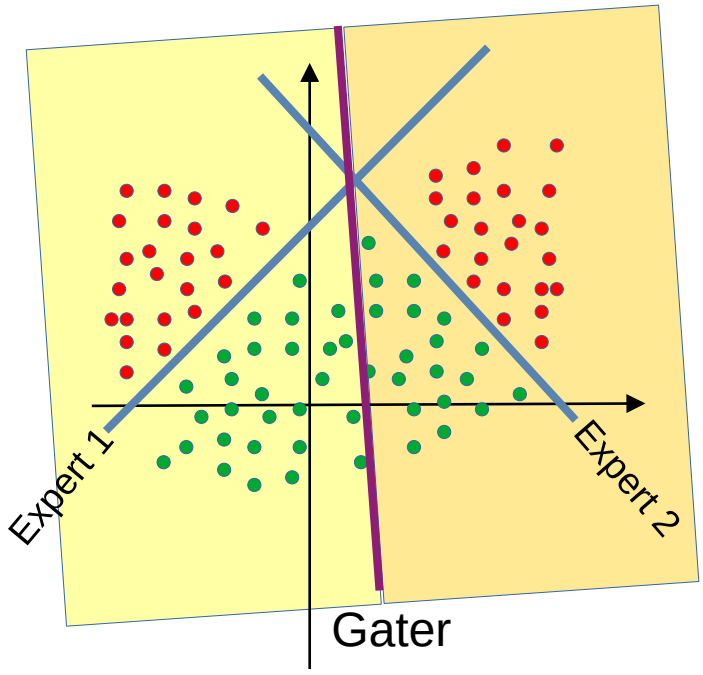
$$s_i = \sum_j w_j x_{ij} \qquad w_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



Mixture of Experts

- **Attention module "Switches" expert networks**



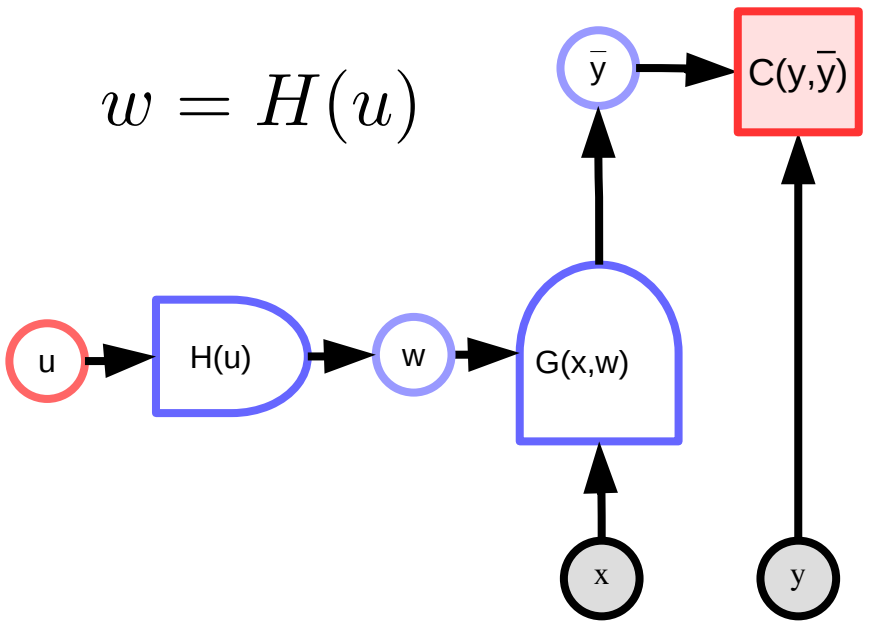


$$s_i = \sum_j w_j x_{ij} \qquad w_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Parameter transformations

- **When the parameter vector is the output of a function**

$$w = H(u)$$



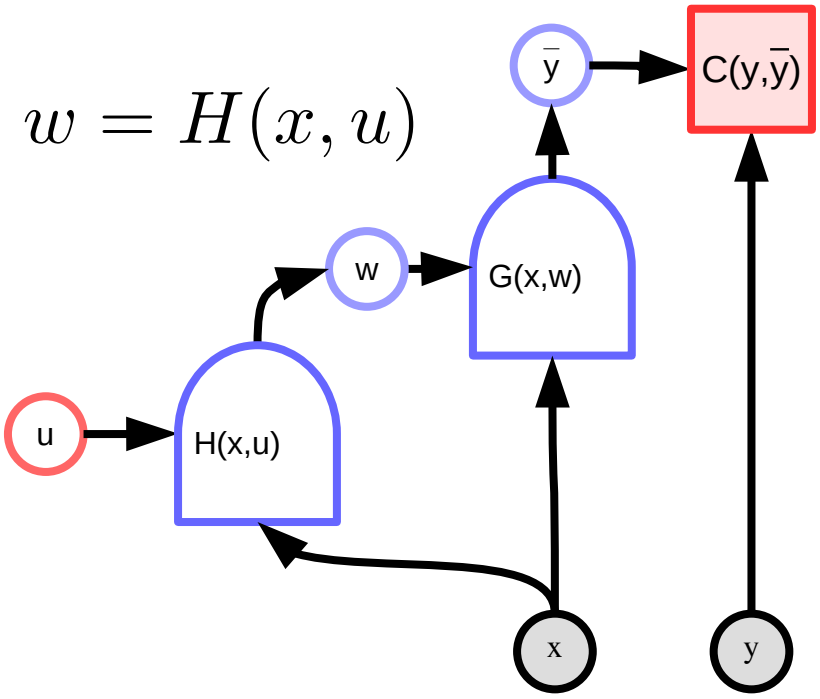
$$u \leftarrow u - \eta \frac{\partial H}{\partial u}^T \frac{\partial C}{\partial w}^T$$

$$w \leftarrow w - \eta \frac{\partial H}{\partial u} \frac{\partial H}{\partial u}^T \frac{\partial C}{\partial w}^T$$

$$[N_w \times N_u] \; [N_u \times N_w] \; [N_w \times 1]$$

“Hypernetwork”

$$w = H(x, u)$$



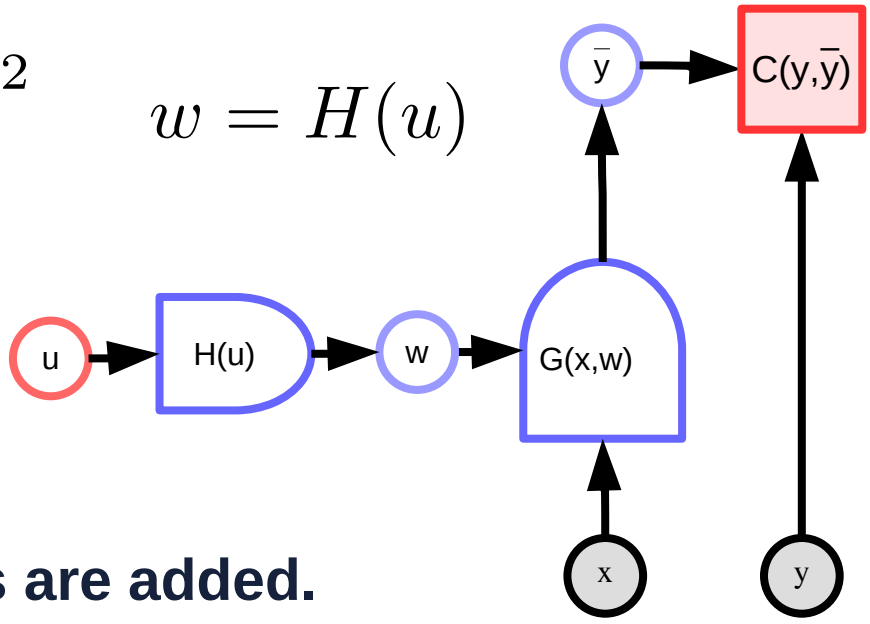
- **When the parameter vector is the output of another network H(x,u)**
- **The weights of network G(x,w) are dynamically configured by network H(x,u)**
- **The concept is very powerful**
  - The idea is very old

Simple parameter transform: weight sharing

- **Function H(u) replicates one component of u into multiple components of w**
  - $w_1, = w_2 = u_1 \quad w_3 = w_4 = u_2$

$$w = H(u)$$

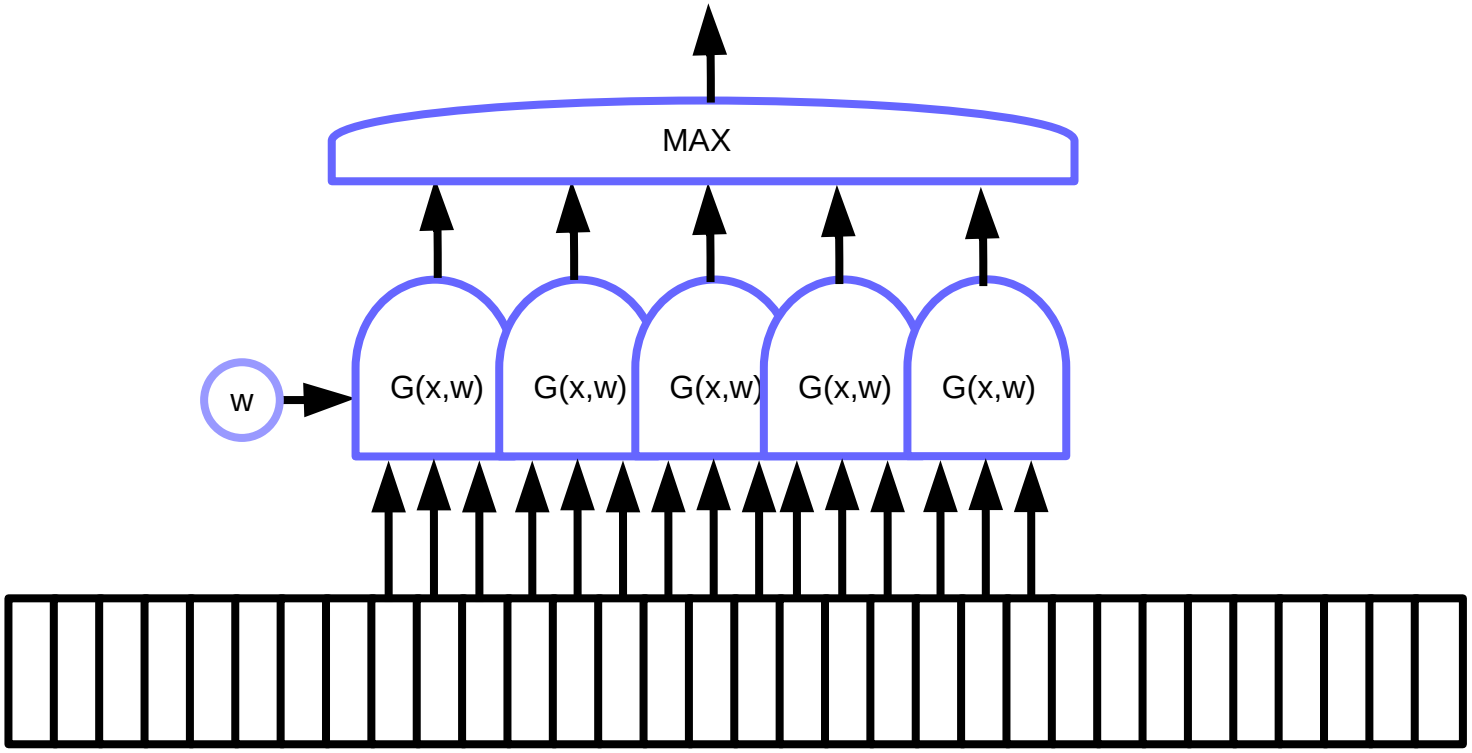
- **H is like a "Y" branch.**
  - Gradients are summed in the backprop



- **The gradients w.r.t. shared parameters are added.**

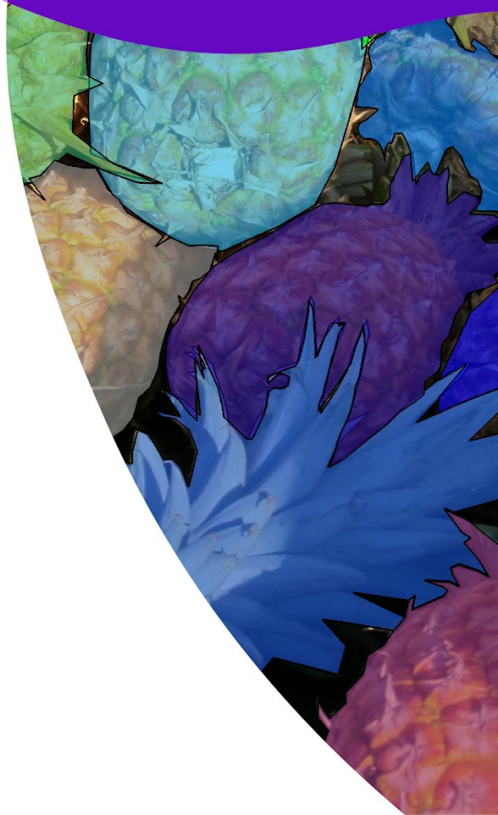
Shared Weights for Shift-Invariant Motif Detection

- **Detecting motifs anywhere on an input**



NEW YORK UNIVERSITY

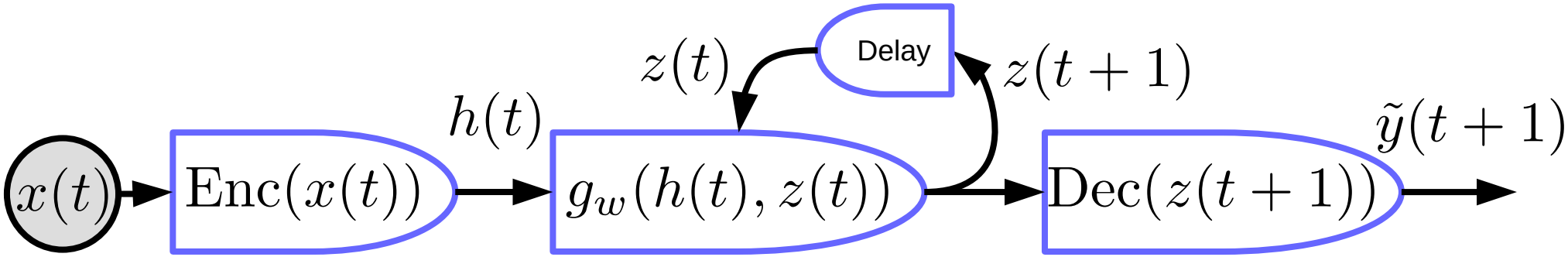
Recurrent Nets



Yann LeCun
NYU - Courant Institute & Center for Data Science
Facebook AI Research

Recurrent Networks

- **Networks with loops. For backprop, unroll the loop.**



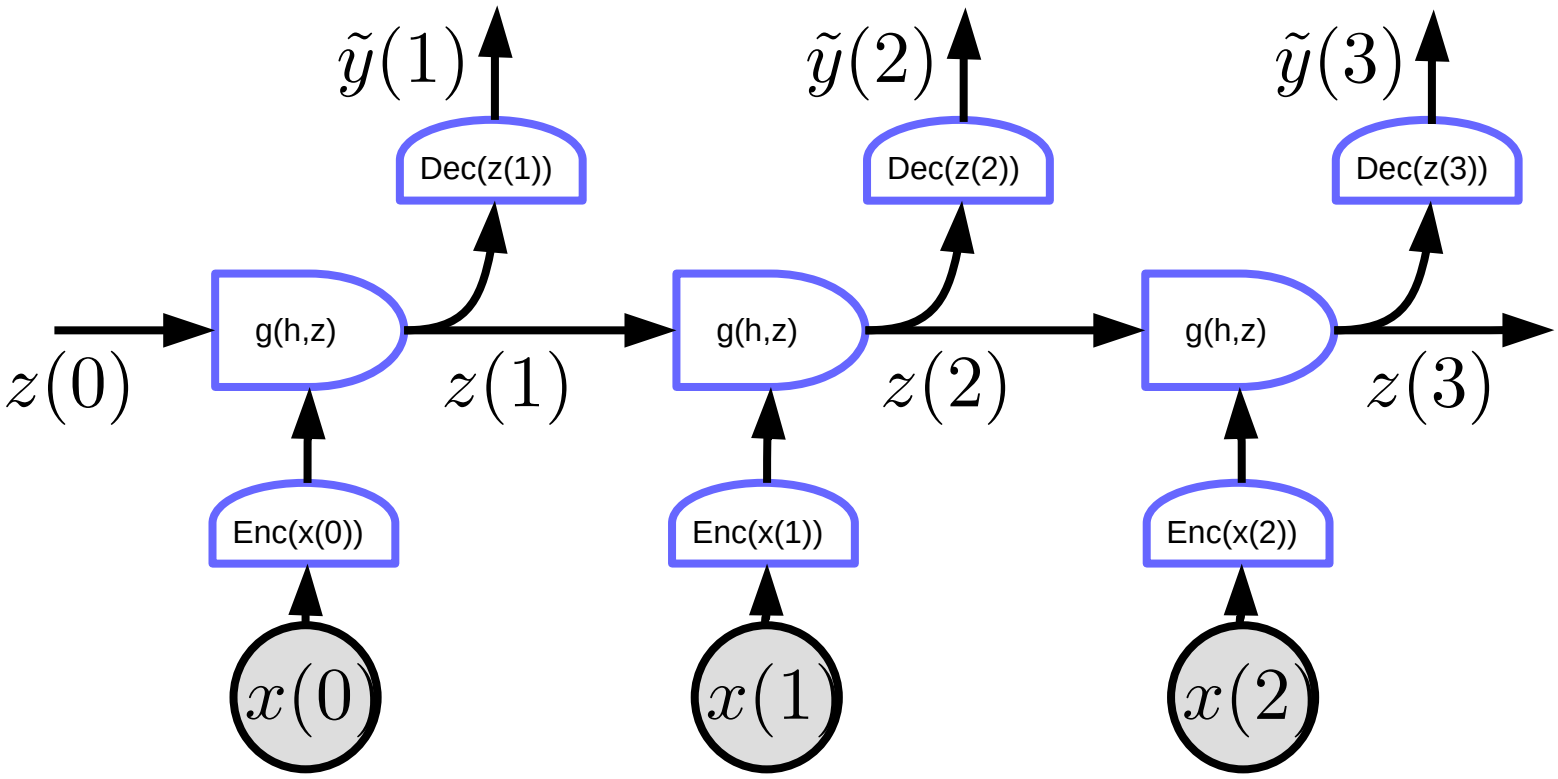
$$h(t) = \mathrm{Enc}(x(t))$$

$$z(t+1) = g_w(h(t), z(t))$$

$$\tilde{y}(t+1) = \mathrm{Dec}(z(t+1))$$

Recurrent Networks

- **Networks with loops. For backprop, unroll the loop.**



RNN tricks

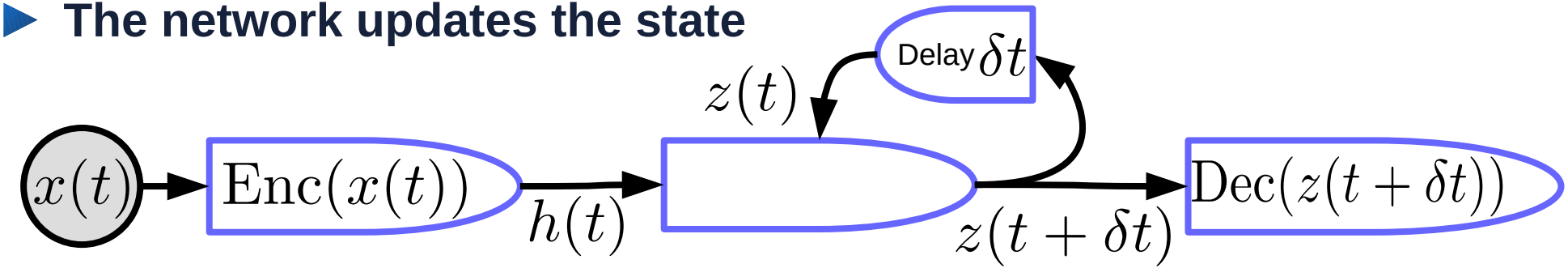
- [Pascanu, Mikolov, Bengio, ICML 2013; Bengio, Boulanger & Pascanu, ICASSP 2013]

- Clipping gradients (avoid exploding gradients)
- Leaky integration (propagate long-term dependencies)
- Momentum (cheap 2nd order)
- Initialization (start in right ballpark avoids exploding/vanishing)
- Sparse Gradients (symmetry breaking)
- Gradient propagation regularizer (avoid vanishing gradient)
- LSTM self-loops (avoid vanishing gradient)

Recurrent Networks for differential equations

- **Differential equation**
- **Time discretization**
- **The network updates the state**

$$\frac{dz(t)}{dt} = g_w(h(t), z(t))$$

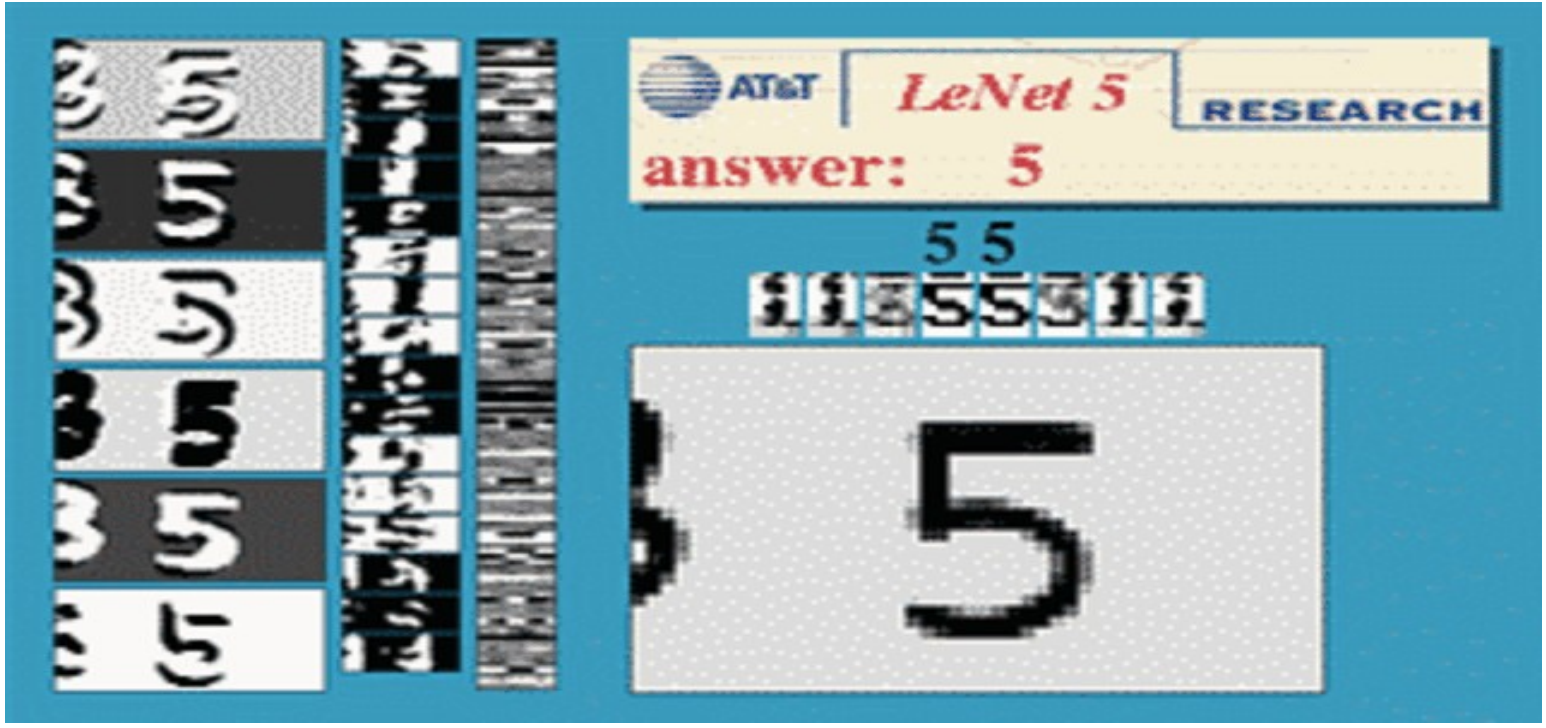


$$h(t) = \mathrm{Enc}(x(t))$$

$$z(t+\delta t) = z(t) + \delta t \; g_w(h(t), z(t))$$

$$y(t+\delta t) = \mathrm{Dec}(z(t+\delta t))$$

Sliding Window ConvNet + Weighted Finite-State Machine

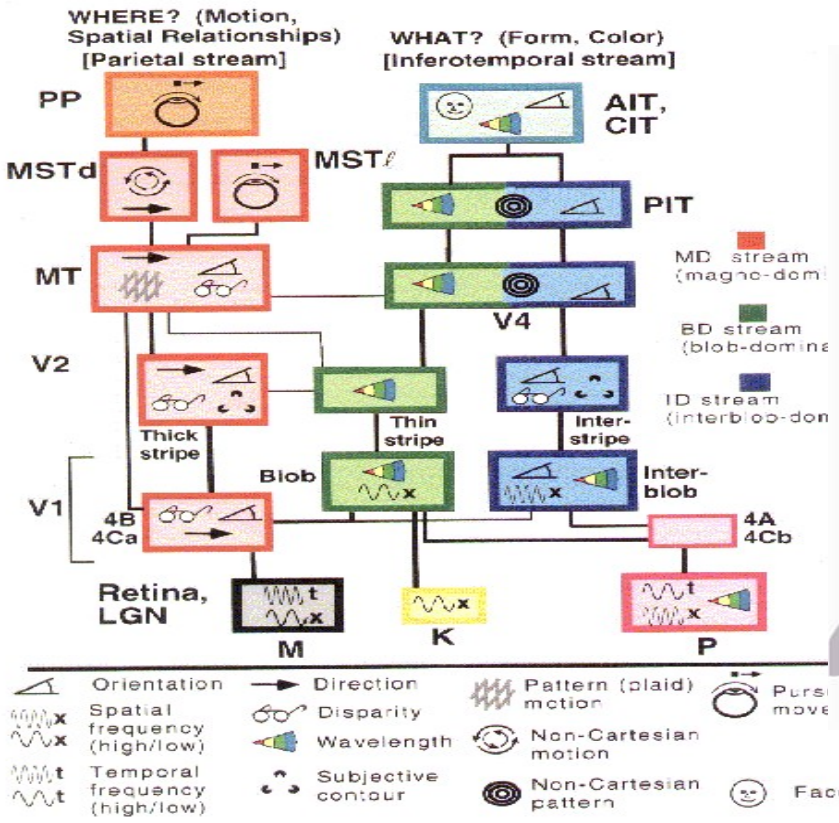


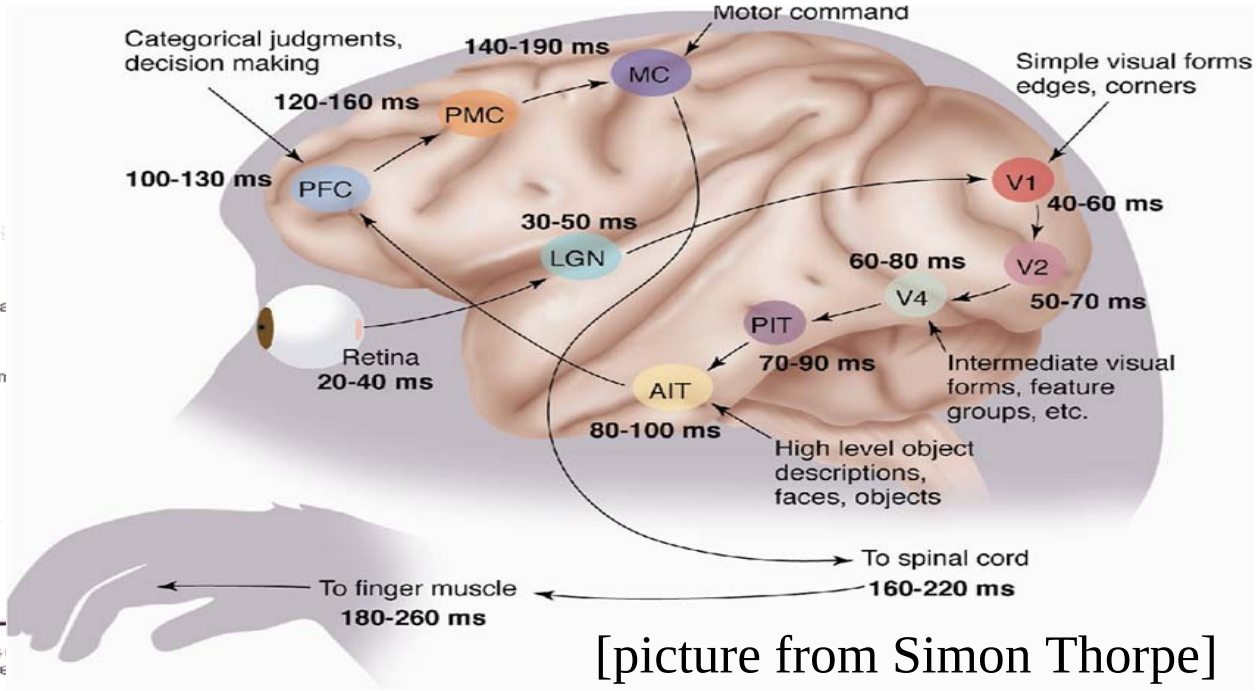
LeNet character recognition demo 1992

- **Running on an AT&T DSP32C (floating-point DSP, 20 MFLOPS)**

How does the brain interprets images?

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





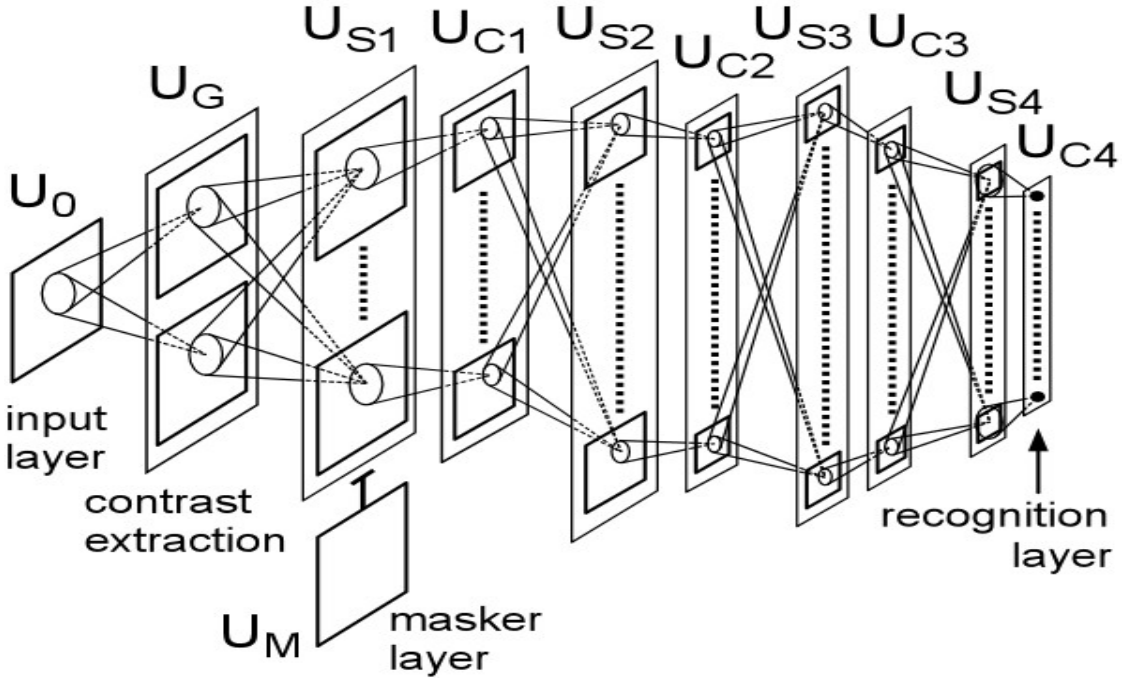
[picture from Simon Thorpe]

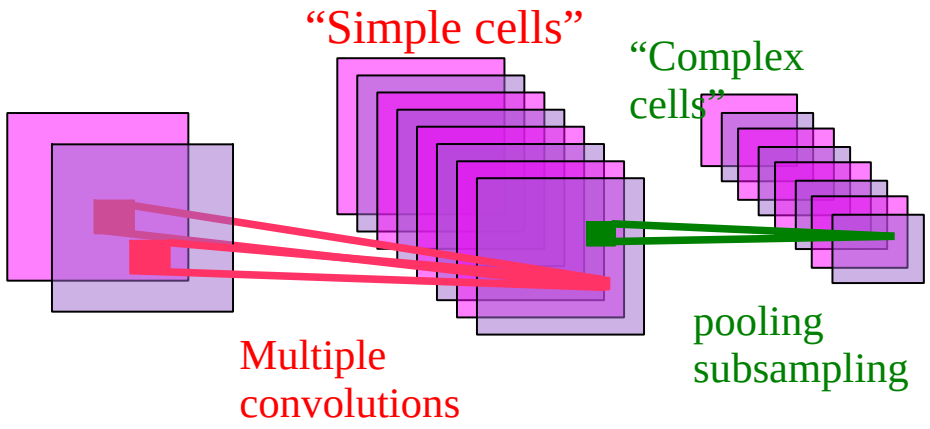
[Gallant & Van Essen]

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

d.





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

- **End of Lecture 3**