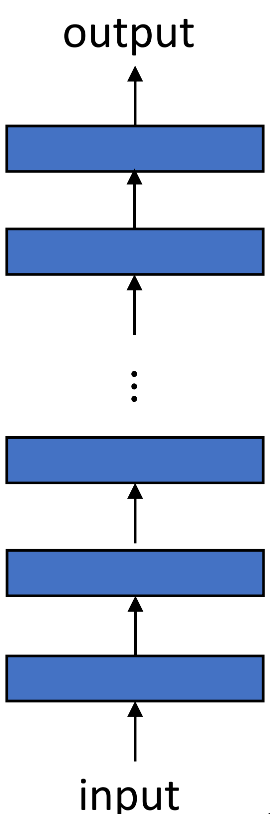
Intro to ML

December 8th, 2021

CHAPTER 12:

Deep Learning

Deep Neural Networks



- Many hidden layers
  - Problematic in training: chain rule multiplication of derivatives (vanish of gradients or explosion)
- End-to-end training
- Learn increasingly abstract representations with minimal human contribution
  - Layers of abstraction in intuitive
  - Vision, speech, language, and so on

Representation learning is really the KEY behind Deep learning

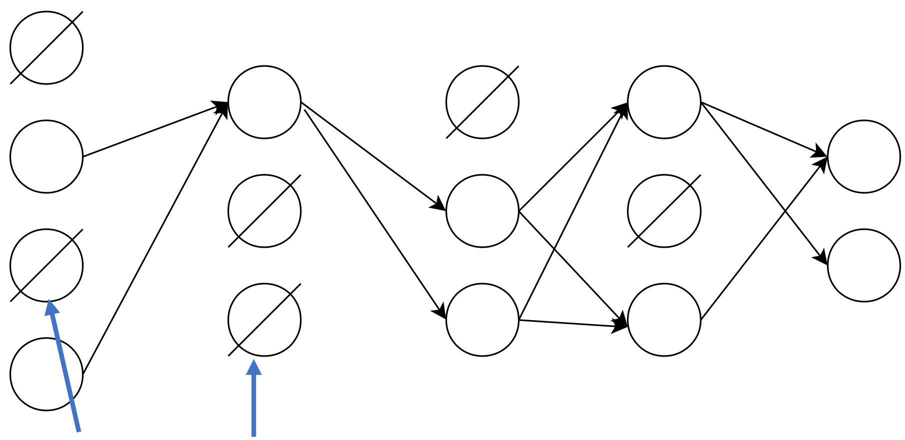
Regularization: Weight Decay

- If a weight is 0, we have a simpler model.
- Initially all weights are close to 0 and they are moved away as learning proceeds.
- Early stopping: Stop before too many weights are updated.
- Weight decay: Add a term that penalizes non-zero weights:

$$E' = E + \frac{\lambda}{2} \sum_i w_i^2 \qquad\qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda' w_i$$

Regularization: Dropout

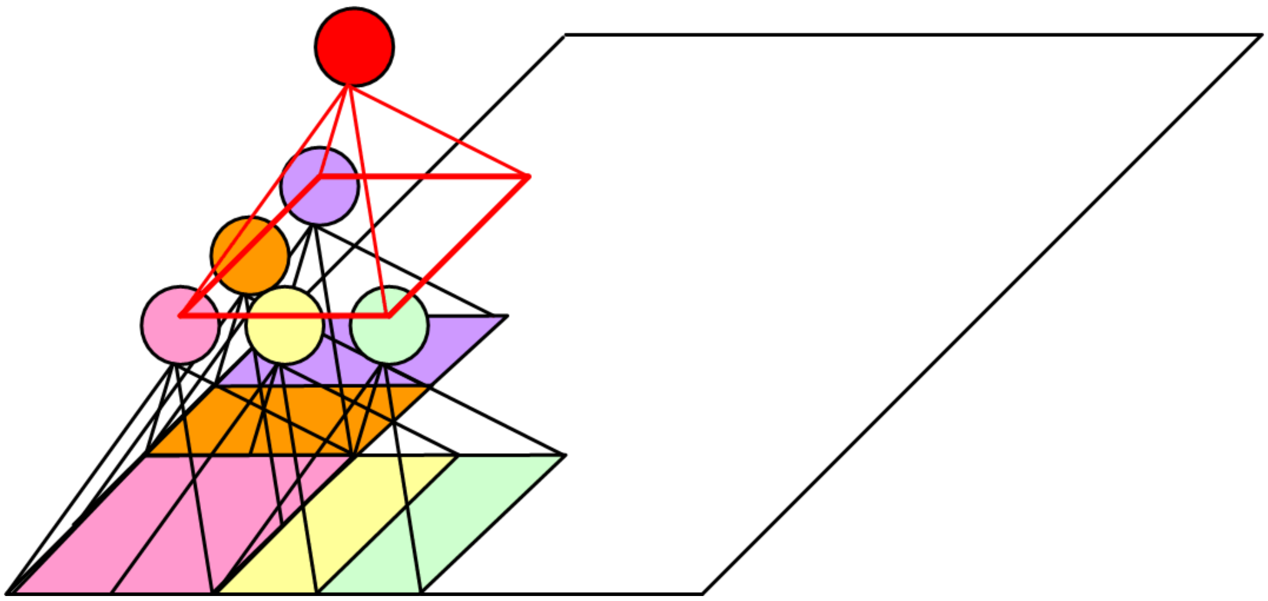
- What if we drop them certain nodes completely?
- Reduce the number of parameters in a ‘random’ fashion to ensure less overfitting and more robust results



Input or hidden unit dropped out (output set to 0) with probability *p* in each minibatch

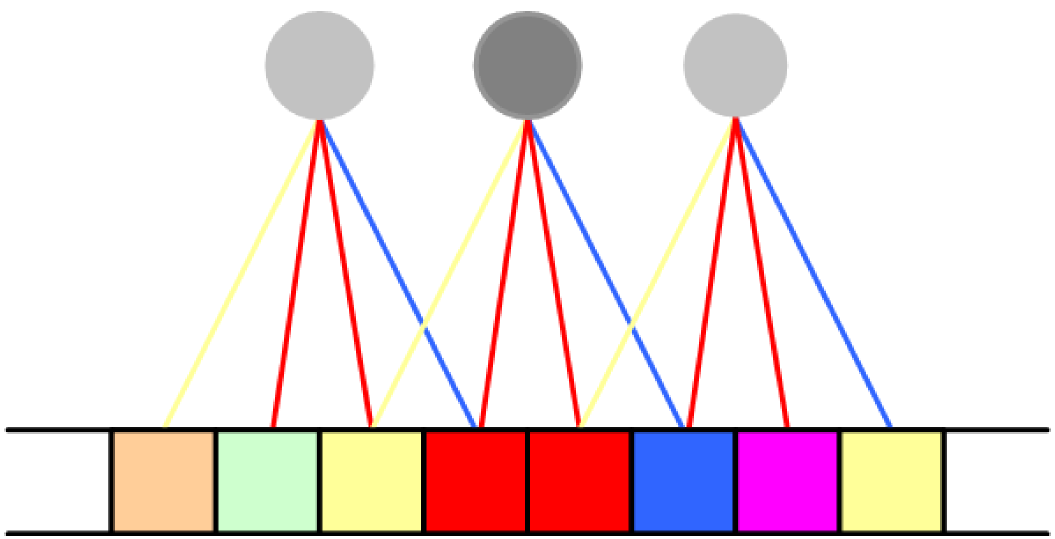
Regularization: Convolutions

- Each unit is connected to a small set of units in the preceding layer. In images, this corresponds to a local patch (LeCun et al 1989).



Regularization: Weight Sharing

The same weights are used in different locations



1d example with 1x4 convolutions and weight sharing

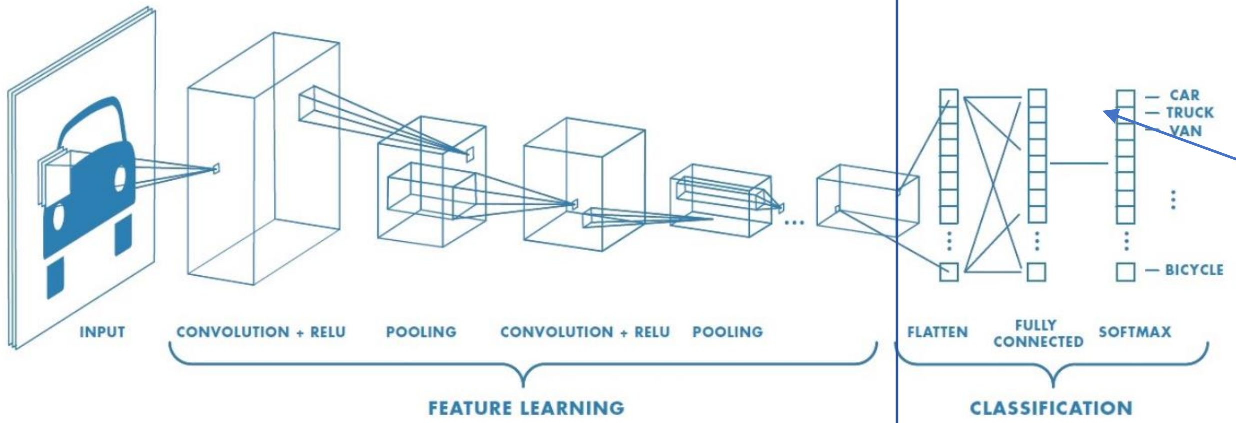
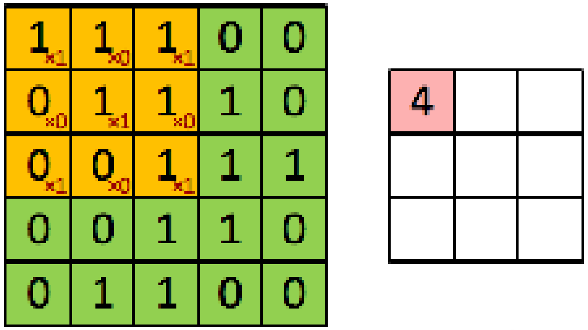


Figure 2 : Neural network with many convolutional layers

Just multilayer perceptron

Or think of this as autoencoder (encoder-decoder)

$$y[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] \cdot x[i-m, j-n]$$



Image

Convolved Feature

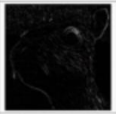
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

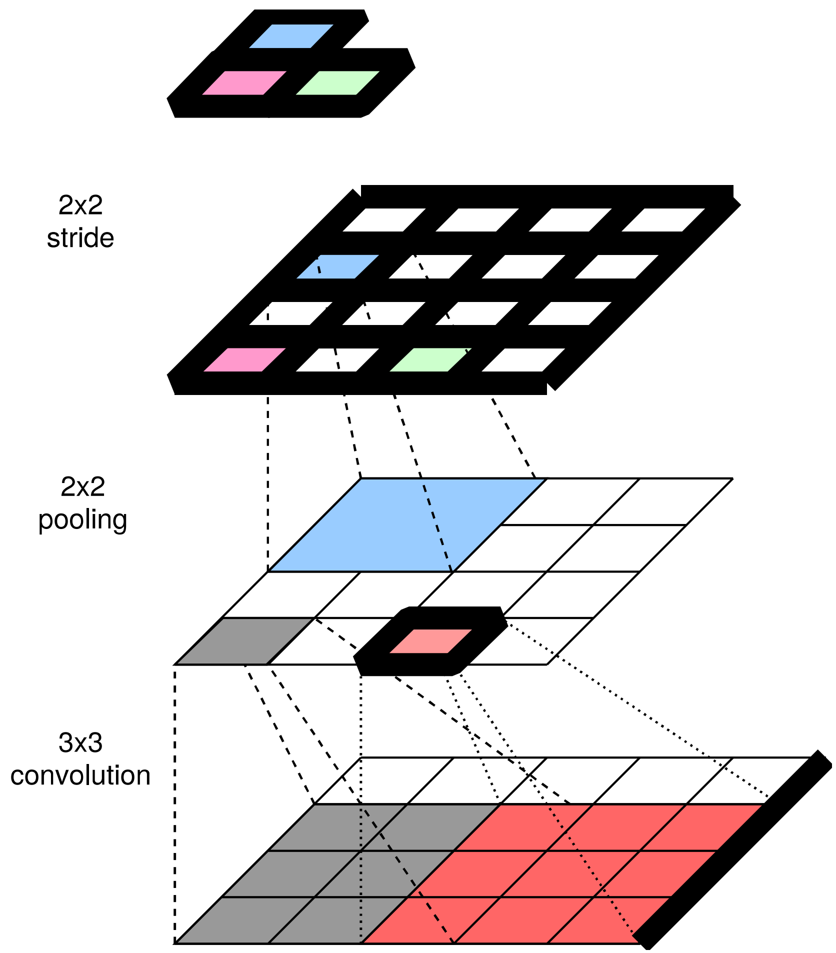
Figure 7 : Some common filters

https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Multiple Convolutional Layers

Move the filter (skip a pixel)

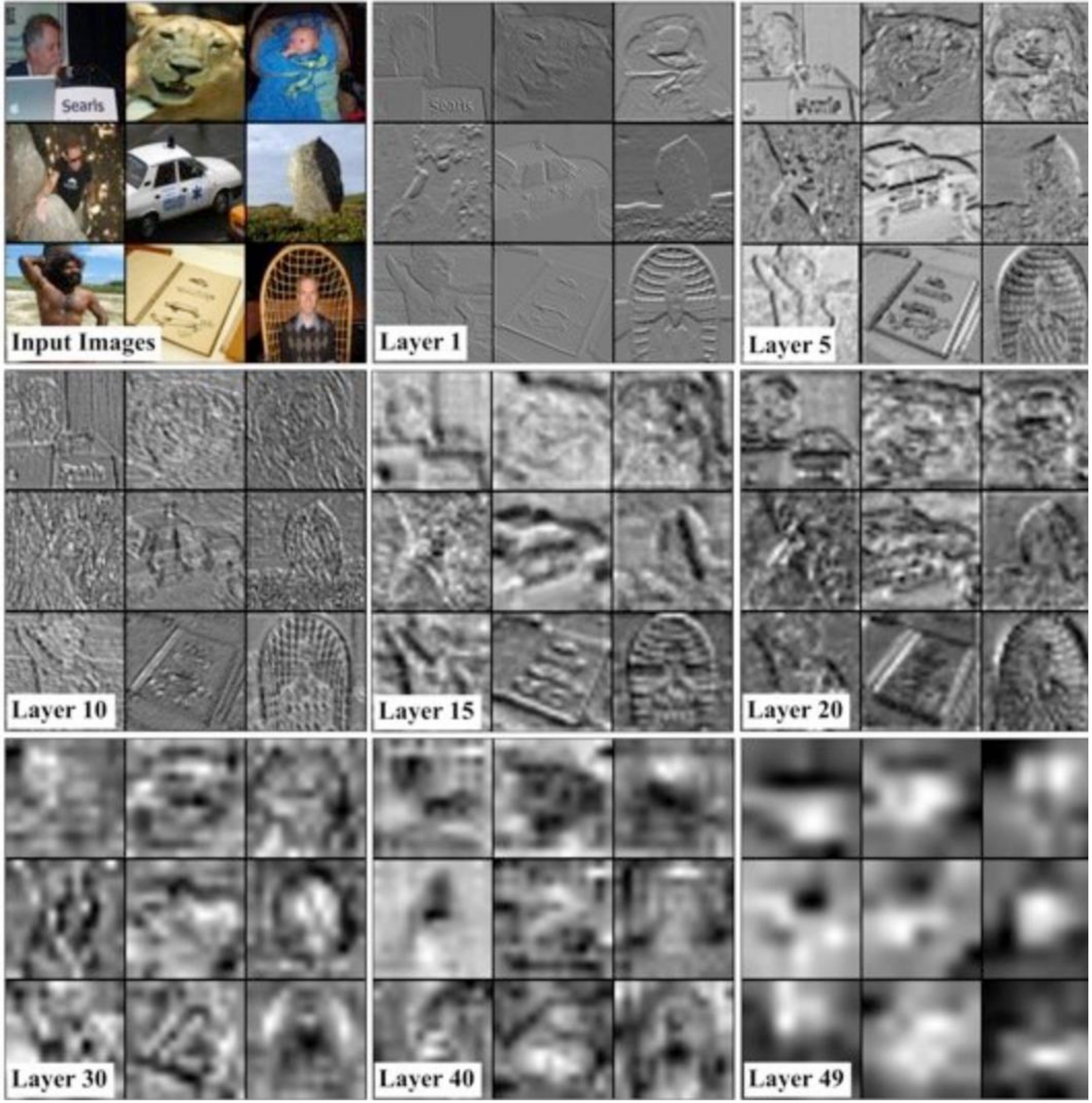
Max, average, so on
Summarization



Multiple layers

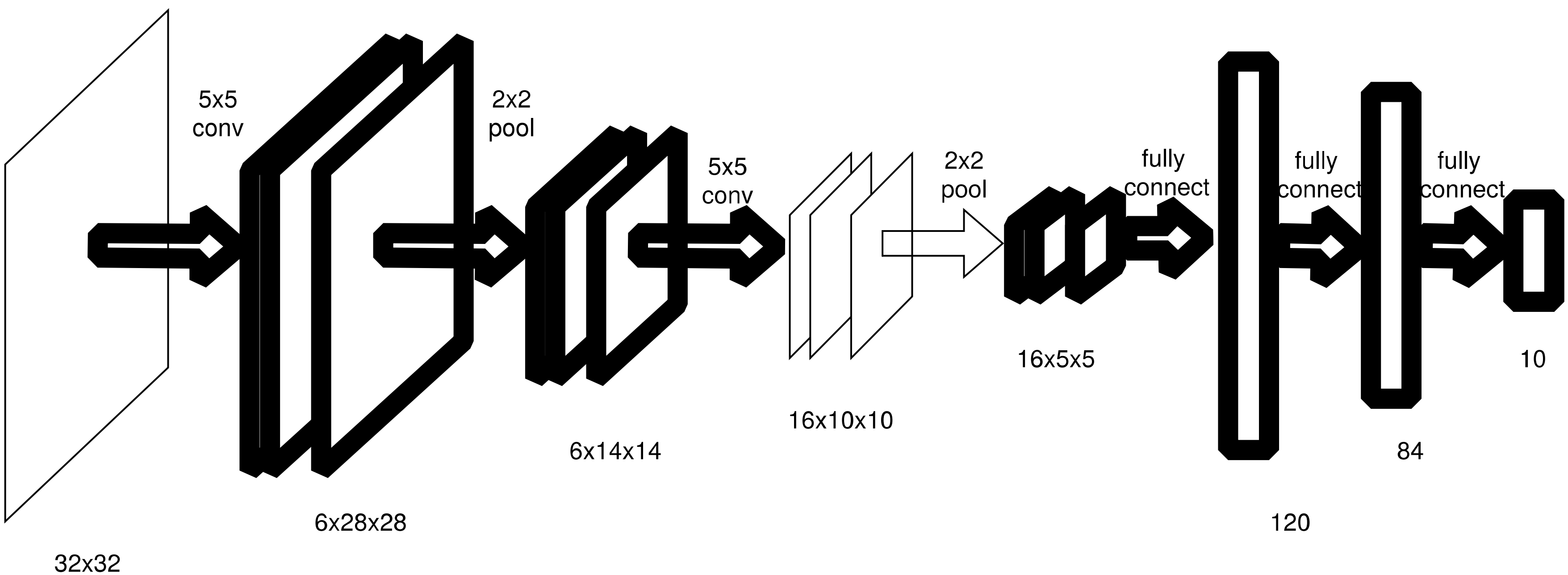
Hierarchical abstraction

Fine-grained to global structure



[DeepFeat: A Bottom Up and Top Down Saliency Model Based on Deep Features of Convolutional Neural Nets]

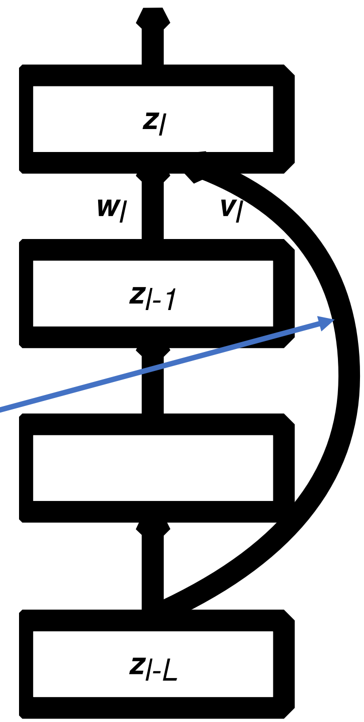
LeNet-5 (LeCun et al 1998)

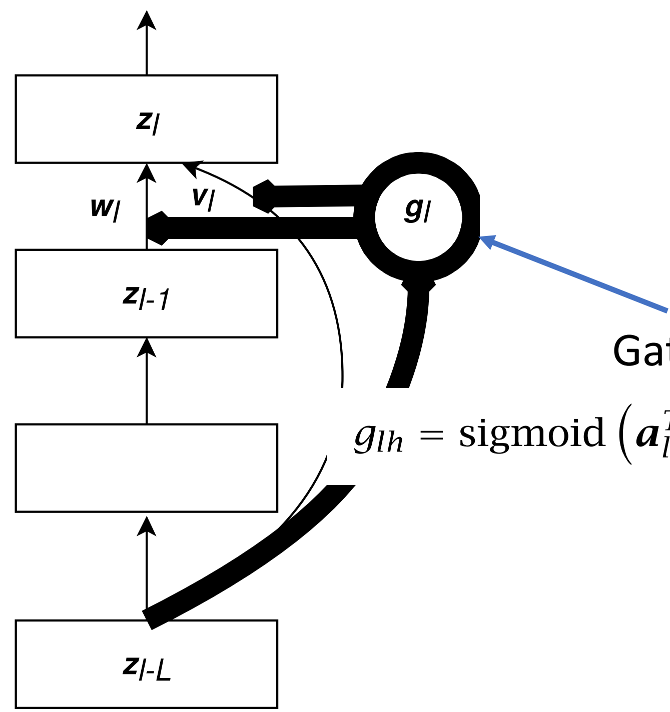


Has approximately 60K weights and trained on 60K examples (MNIST data set)

More ways to handle vanishing gradient
Skip Connections

Skip connection





Gating unit

$$g_{lh} = \mathrm{sigmoid}\left(\boldsymbol{a}_{lh}^{T}\boldsymbol{z}_{l-L} + a_{lh0}\right)$$

$$z_{lh} = f\left(\sum_i w_{l,h,i} z_{l-1,i} + \sum_j v_{l,h,j} z_{l-L,j}\right)$$

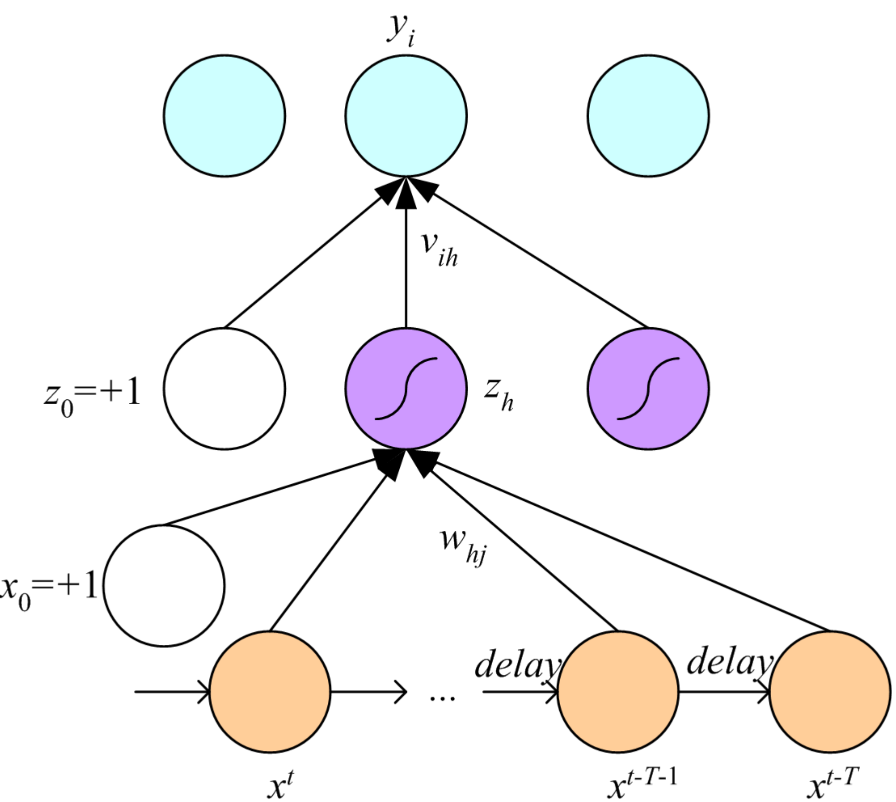
$$z_{lh} = f\left(g_{lh}\sum_i w_{lhi} z_{l-1,i} + (1 - g_{lh})\sum_j v_{lhj} z_{l-L,j}\right)$$

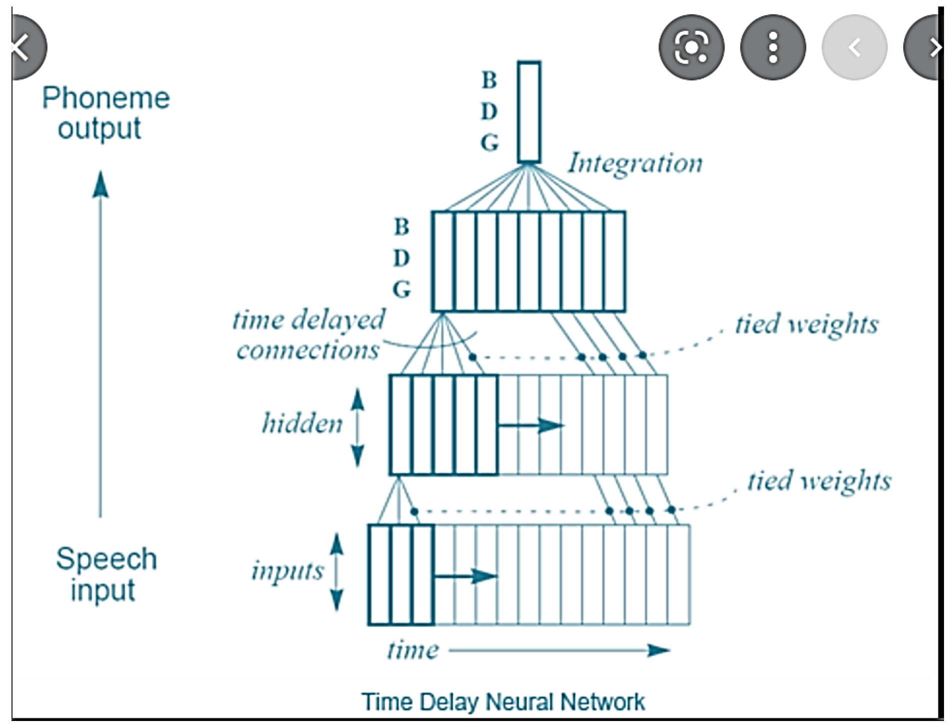
Learning Time (sequence modeling)

- Applications:
  - Sequence recognition: Speech recognition
  - Sequence reproduction: Time-series prediction

- Network architectures
  - Time-delay networks (Waibel et al., 1989)
  - Recurrent networks (Rumelhart et al., 1986)

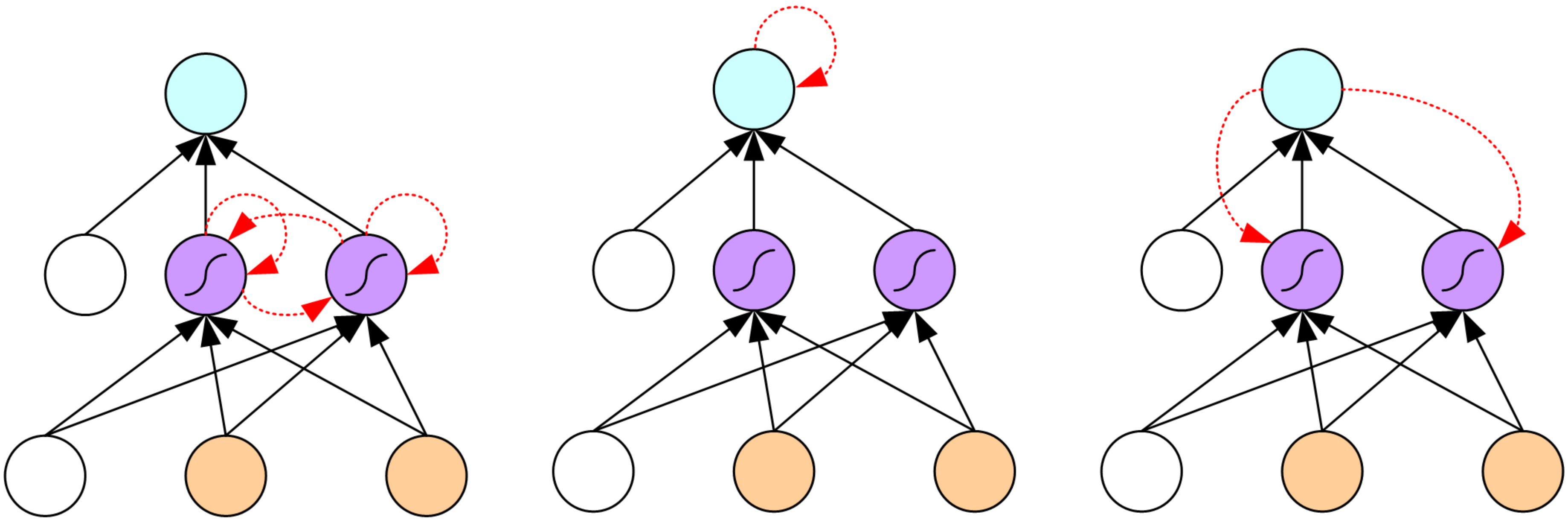
Time-Delay Neural Networks (TDNN)





It is also just like a 1-D convolution neural network

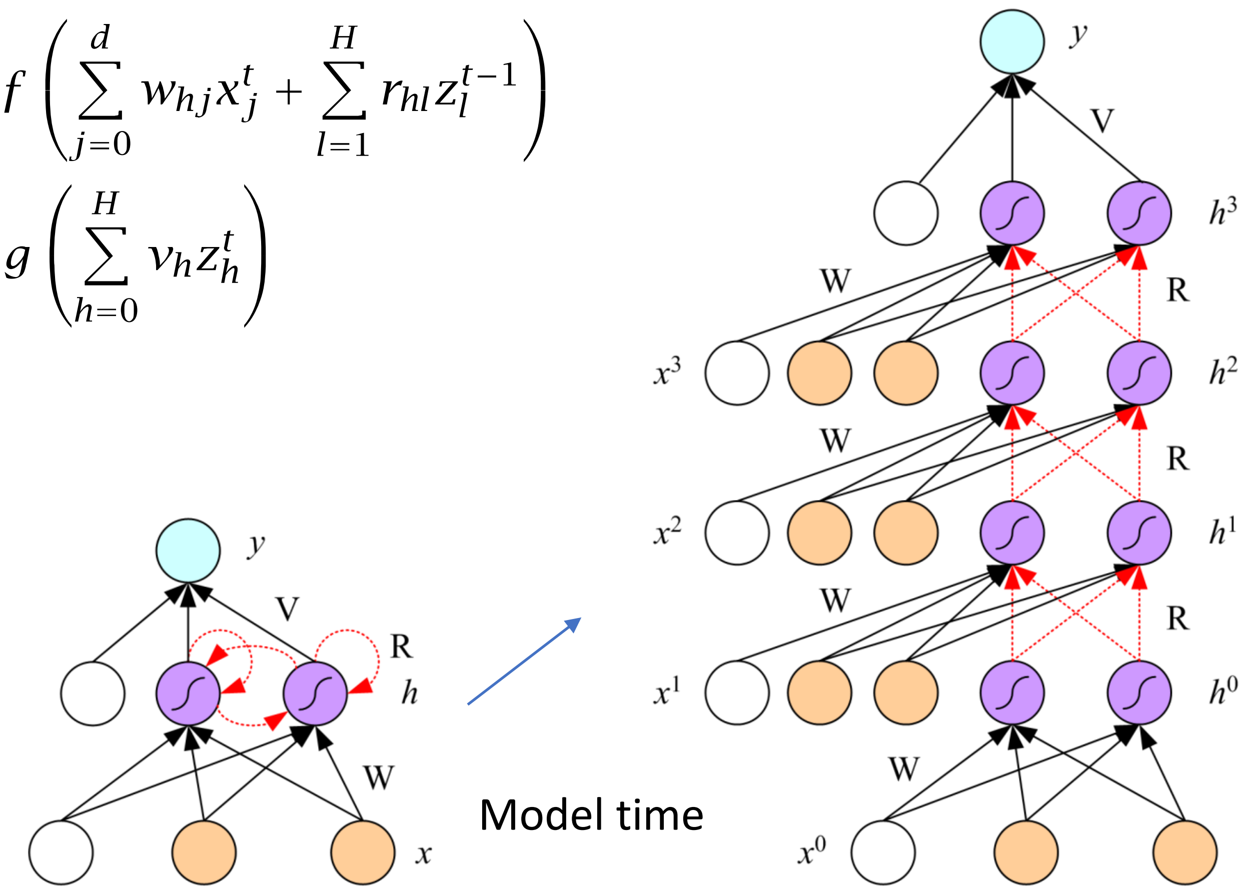
Recurrent Networks(RNN)



(a) (b) (c)

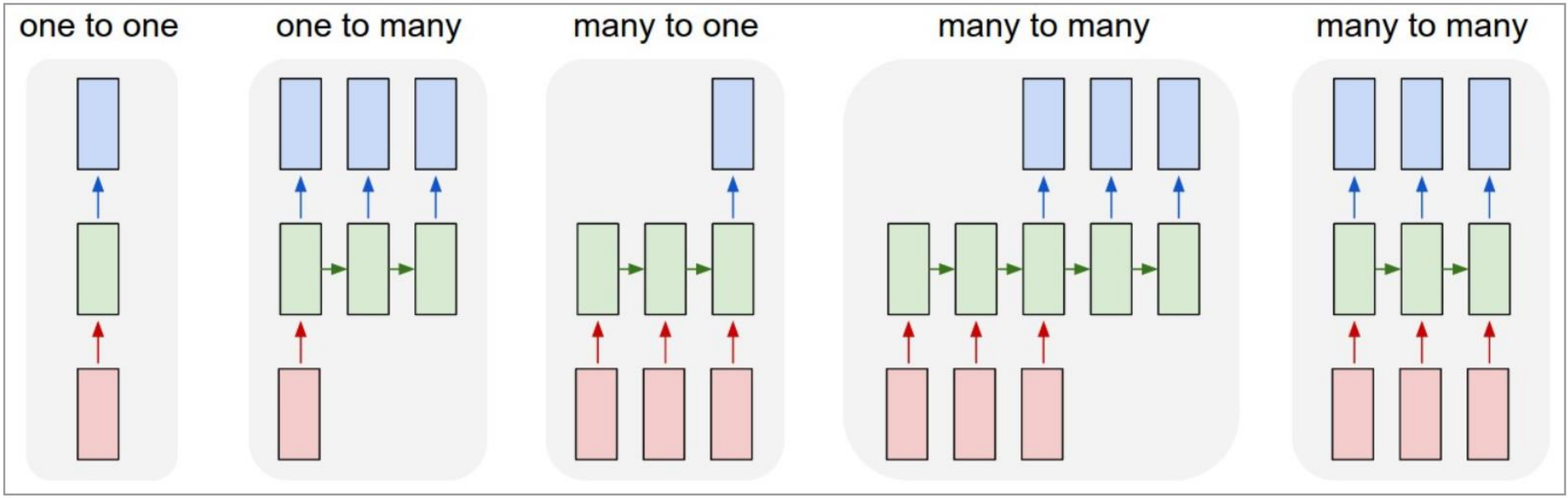
Unfolding in Time -> sequence model

$$
\begin{aligned}
z_h^t &= f\left(\sum_{j=0}^{d} w_{hj} x_j^t + \sum_{l=1}^{H} r_{hl} z_l^{t-1}\right) \\
y^t &= g\left(\sum_{h=0}^{H} v_h z_h^t\right)
\end{aligned}
$$



(a) (b)

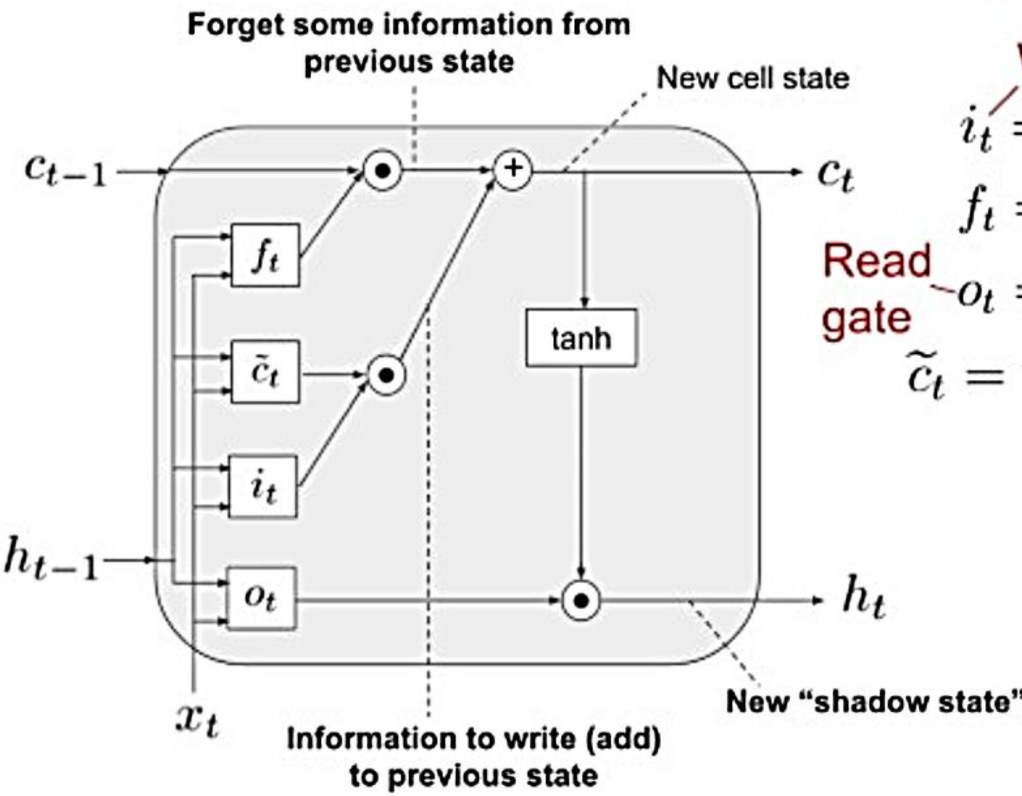
Many variants of RNN for different applications



Long short-term memory (LSTM)

Activation function

LSTM Cell



Conceptually, we lose information

LSTM equations

We use shadow state to calculate gates

Write gate

$$i_t = \boxed{\sigma}(W_i h_{t-1} + U_i x_t + b_i) \quad \text{Input gate}$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \quad \text{Forget gate}$$

Read gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \quad \text{Output gate}$$

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b) \quad \text{Memory cell candidate}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t \quad \text{Memory cell}$$

$$h_t = o_t \circ tanh(c_t) \quad \text{Shadow state}$$

$$y_t = h_t \quad \text{Cell Output}$$

Read occurs after writing

142

deepsystems.ai