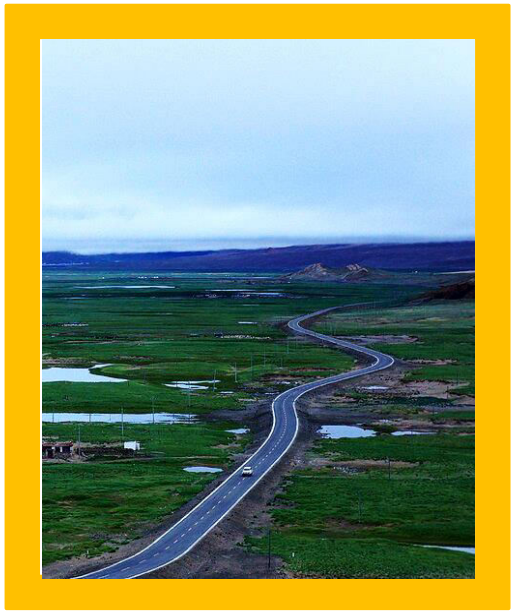


Administrivia: Road Ahead

Lectures, Mini Hackathons, Hackathon

Weeks Ahead

- No Extension of the course
- Resequence between Roundup and Lecture

Week 12:
Review of Deep Learning

Apr. 14

Week 13:
Decision Trees and Ensemble Methods

Apr. 21

Week 14 (Apr 28):
Deep Learning Arch., Appl.

Apr. 28

Week 15:
Deep Learning Arch., Appl.

May 5

Week 16:
Action Workshop, Roundup

May 12

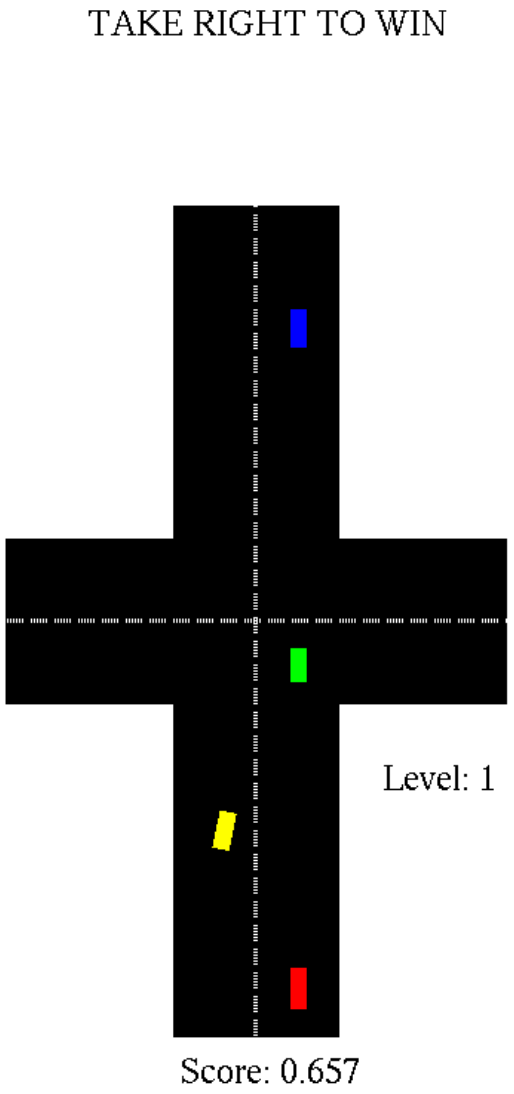
Week 17:
Hackathon II, Conclusion

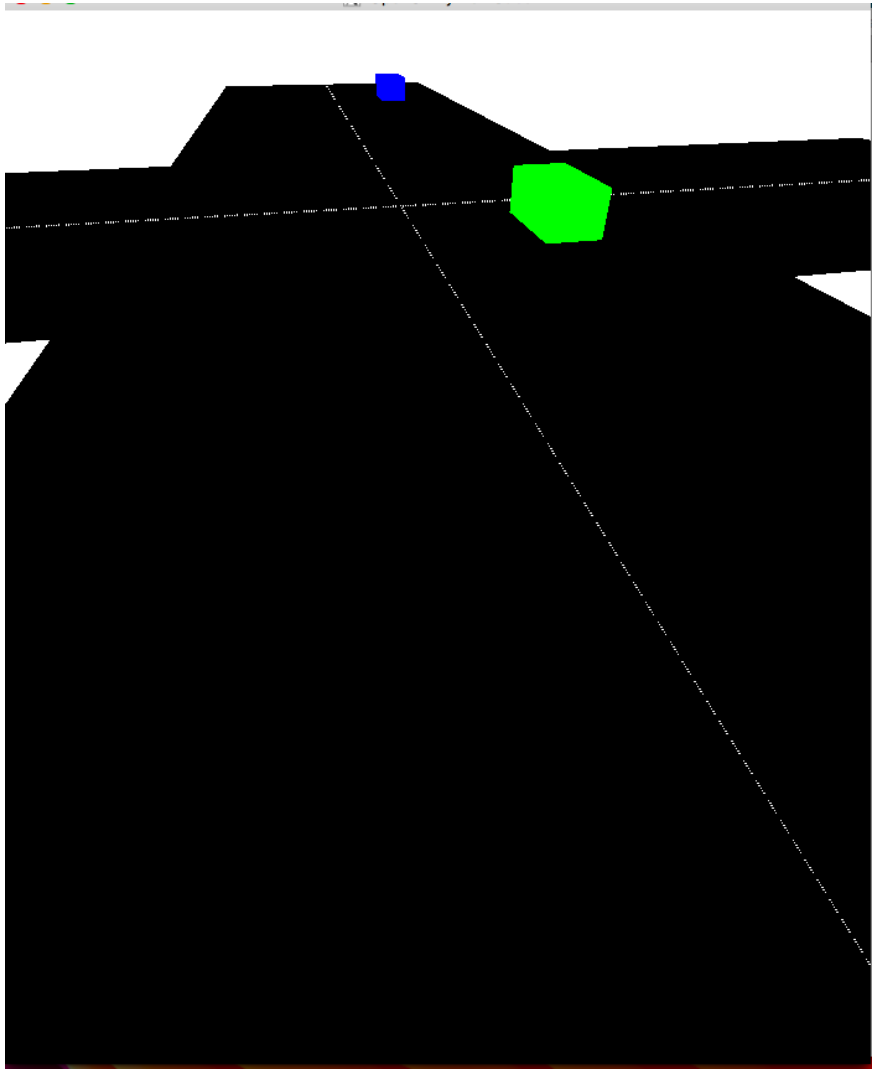
May 19

This Week

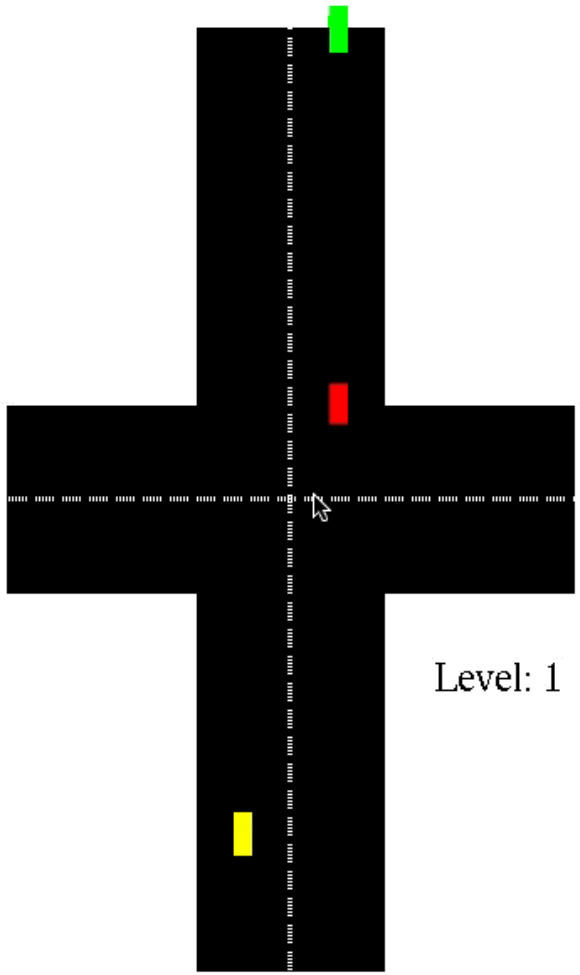
- Lecture
  - Review of Deep Neural Networks
- Lab
  - Catch-up with Experiments 9/10/11
  - Mini Hackathon: Drive a car

The Task





The Task



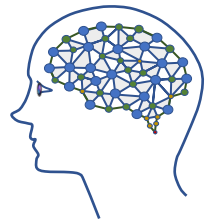
Lab Plan

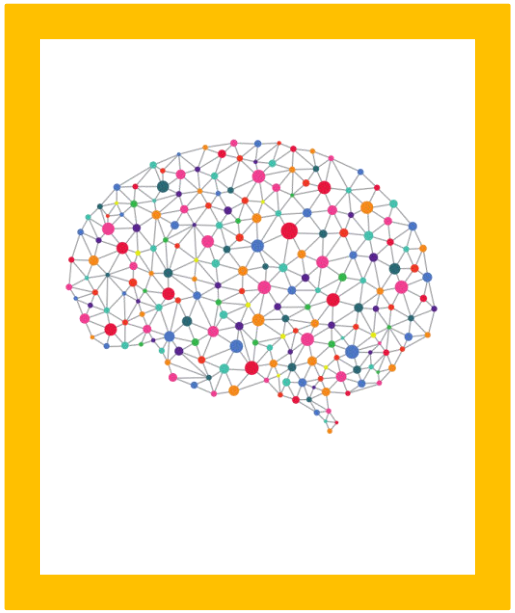
- First 30 minutes: Generate Training data and upload
- Work on experiments for 9/10/11
- Download combined training data
- Use it to train an MLP
  - How do you deal with noise in data?

Mini Hackathons

- Mechanism:
  - Add a group exercise to the regular lab sessions
  - We will start in the labs, you can complete it later in the week
- Goal:
  - Enable you to do larger tasks
  - Enable peer learning
  - Form the groups early to facilitate final Hackathon
  - Build some background for the final Hackathon

Questions?





Deep Neural Networks: A Review

CNNs, RNNs and Auto Encoders

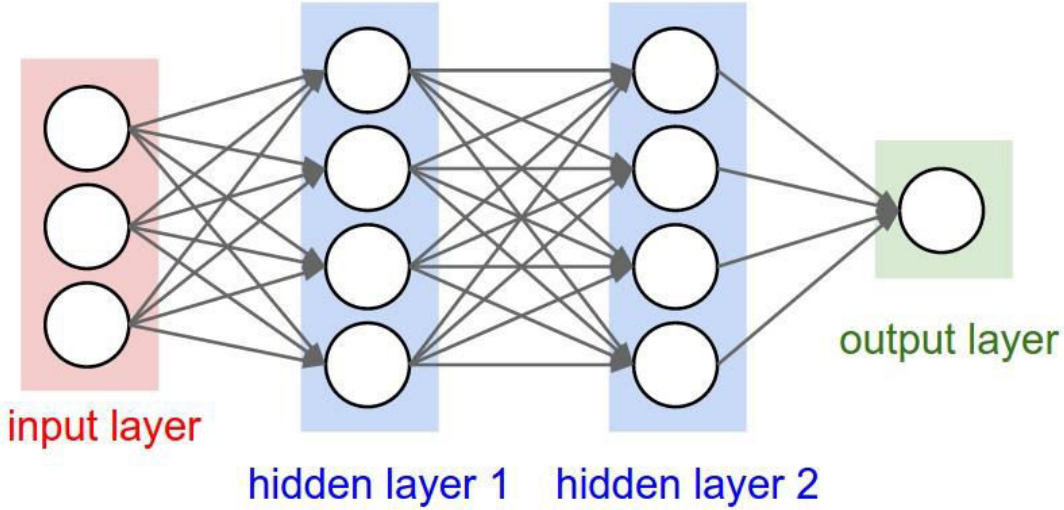
Convolutional Neural Networks

Inference and Training

Convolutional Neural Networks

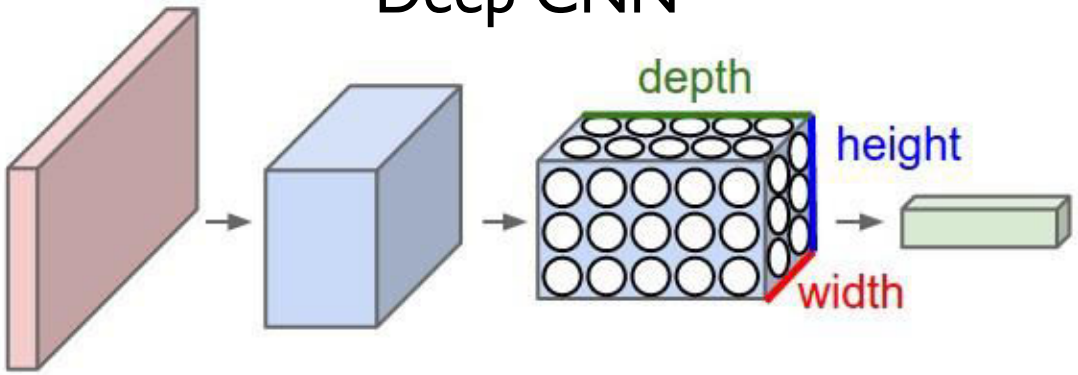
When Input is a vector
(fixed length).

Linear Layer
MLP (Multilayered)



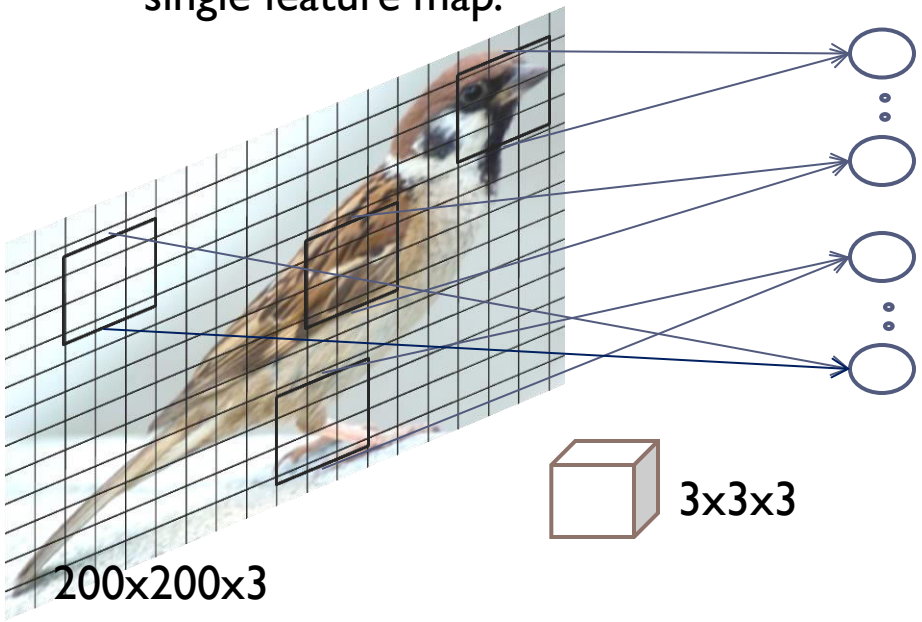
When Input is a 1D/2D/3D sequence.
(variable length)

Convolutional Layer
Deep CNN

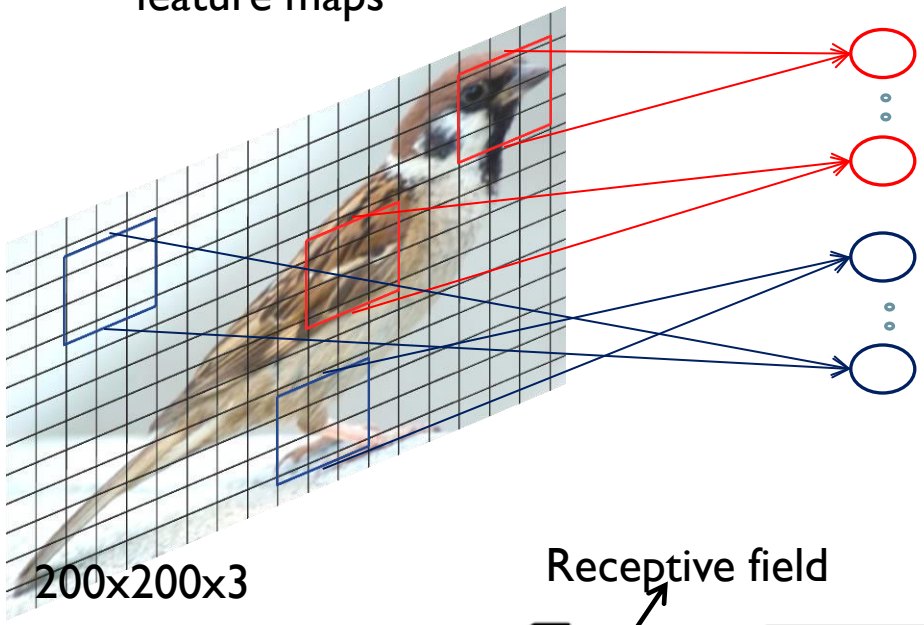


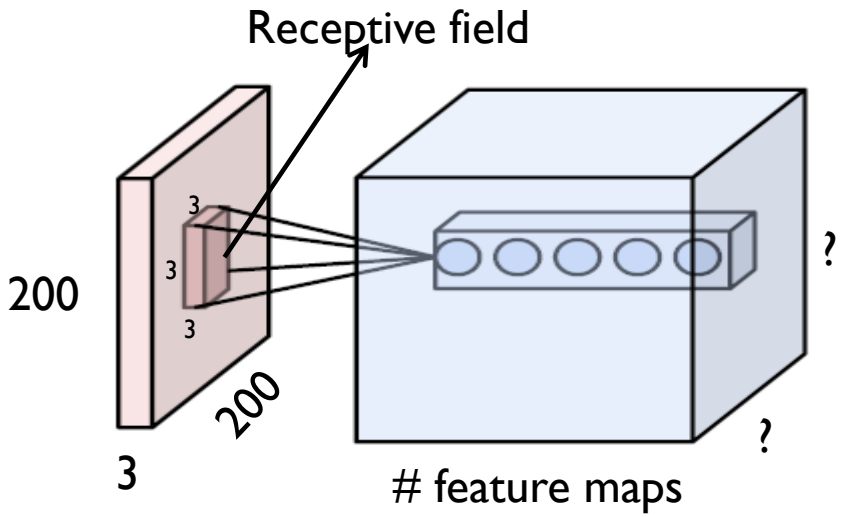
Convolutional Neural Network

Convolutional layer with single feature map.



Convolutional layer with multiple feature maps

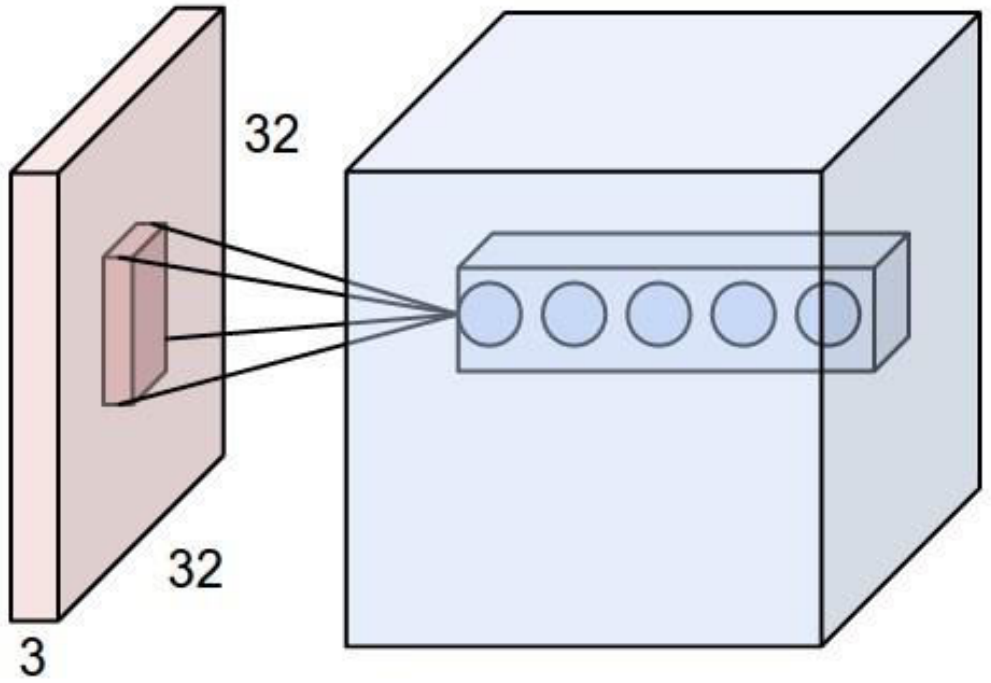




- #Hidden Units: 120,000
- #Params: 27 x #Feature Maps
- Sharing parameters
- Exploiting the stationarity property and preserves locality of pixel dependencies

Input, Output Channels : Multiple Filters

A Closer Look at Convolution:

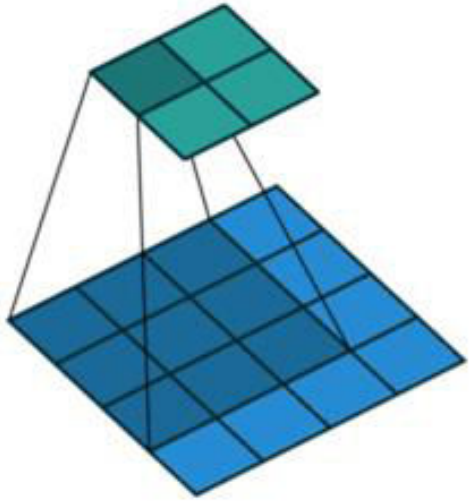


Convolution Layer

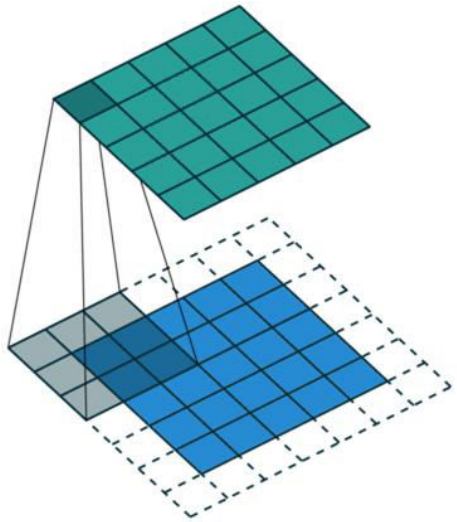
Window size

Stride

Padding



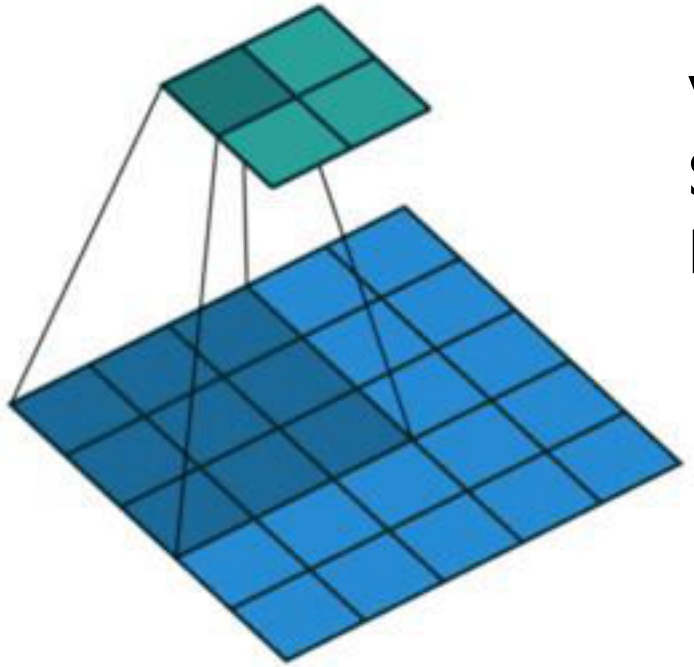
Window size: 3x3
Stride: 1
Padding: 0



Window size: 3x3
Stride: 1
Padding: 1

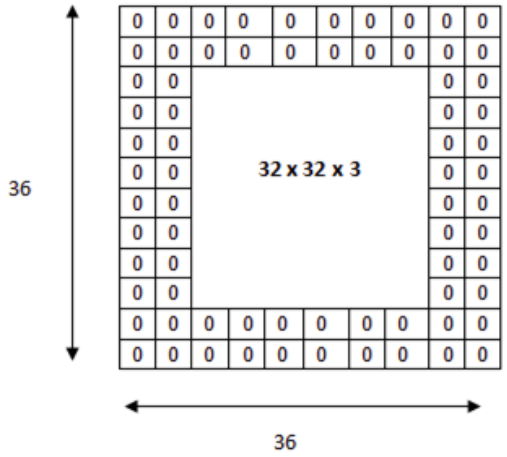
Convolution Layer

Strides reduces dimension

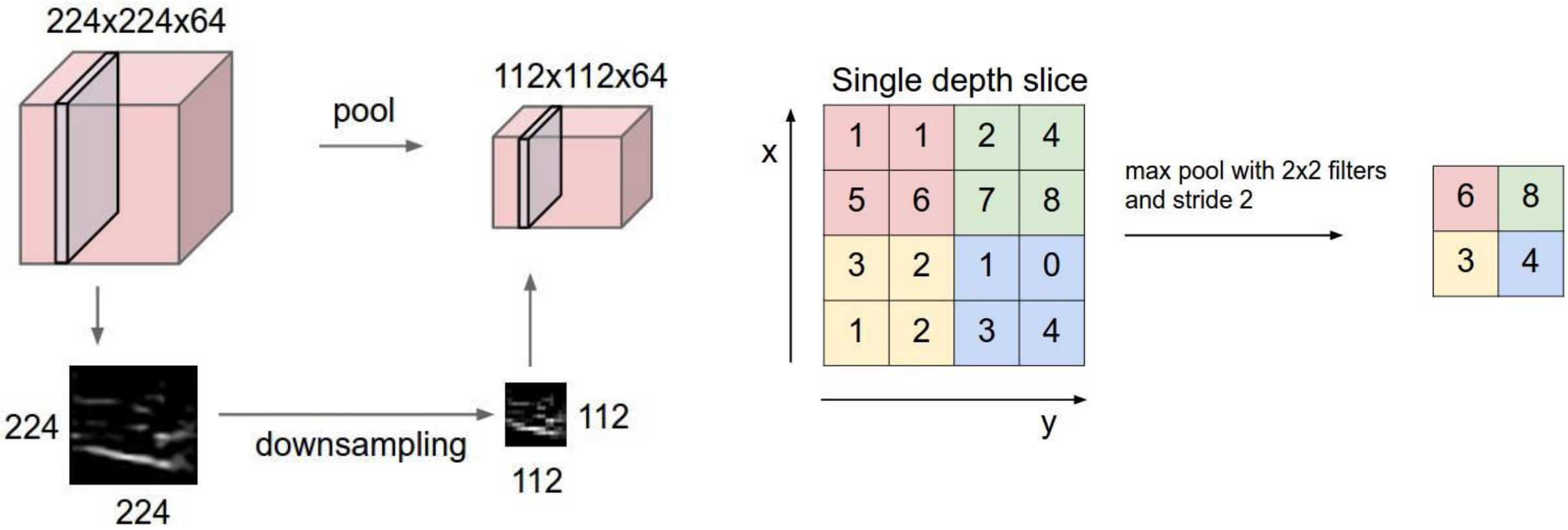


Window size: 3x3
Stride: 2
Padding: 0

$$O = \frac{(W - K + 2P)}{S} + 1$$

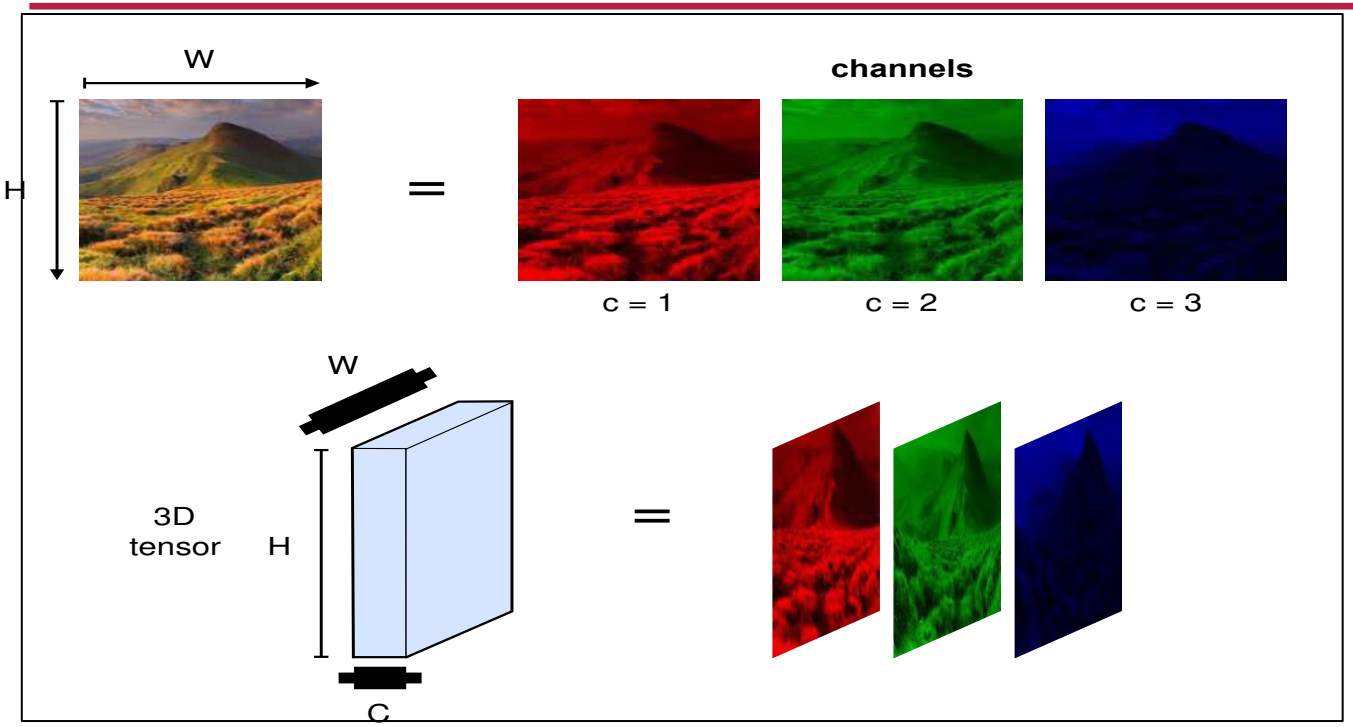


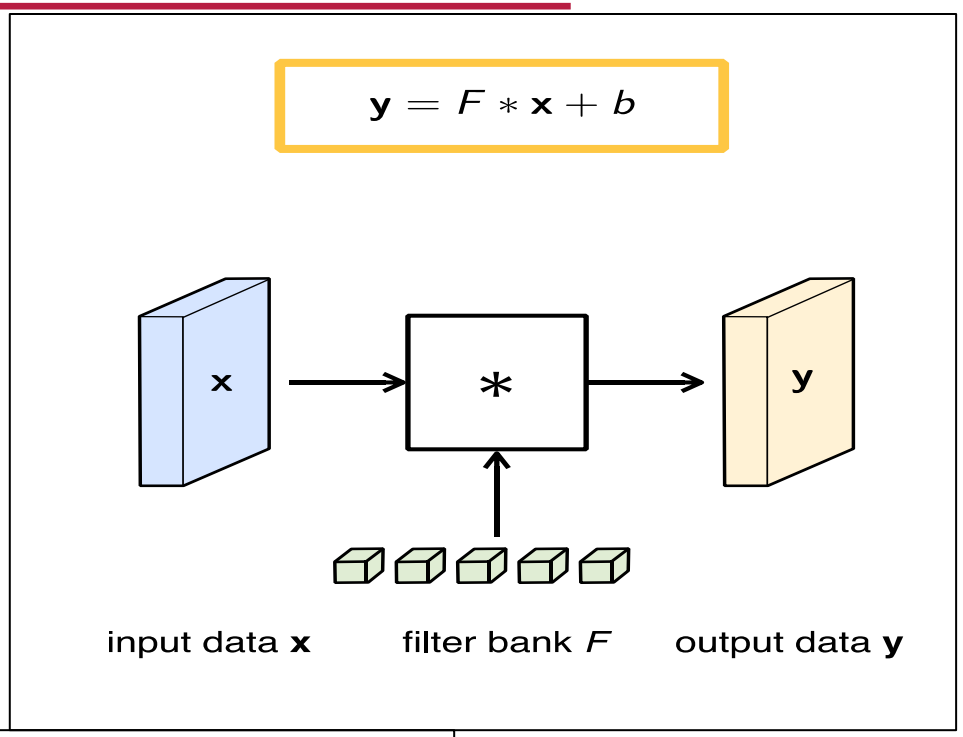
Pooling

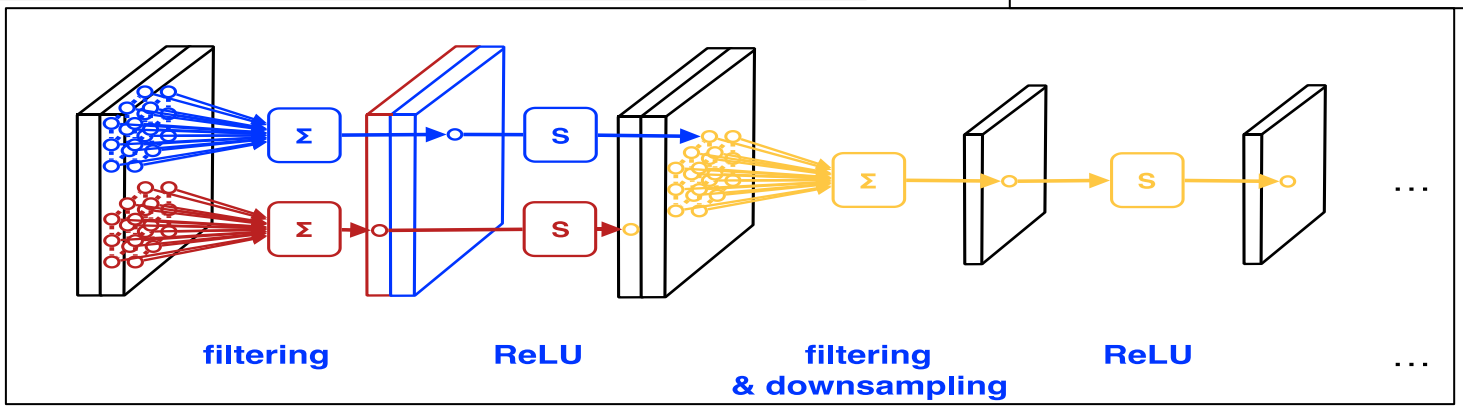


No Learnable Parameters

Summary of CNNs







Learn the full pipeline

VISION

pixels → edge → texton → motif → part → object

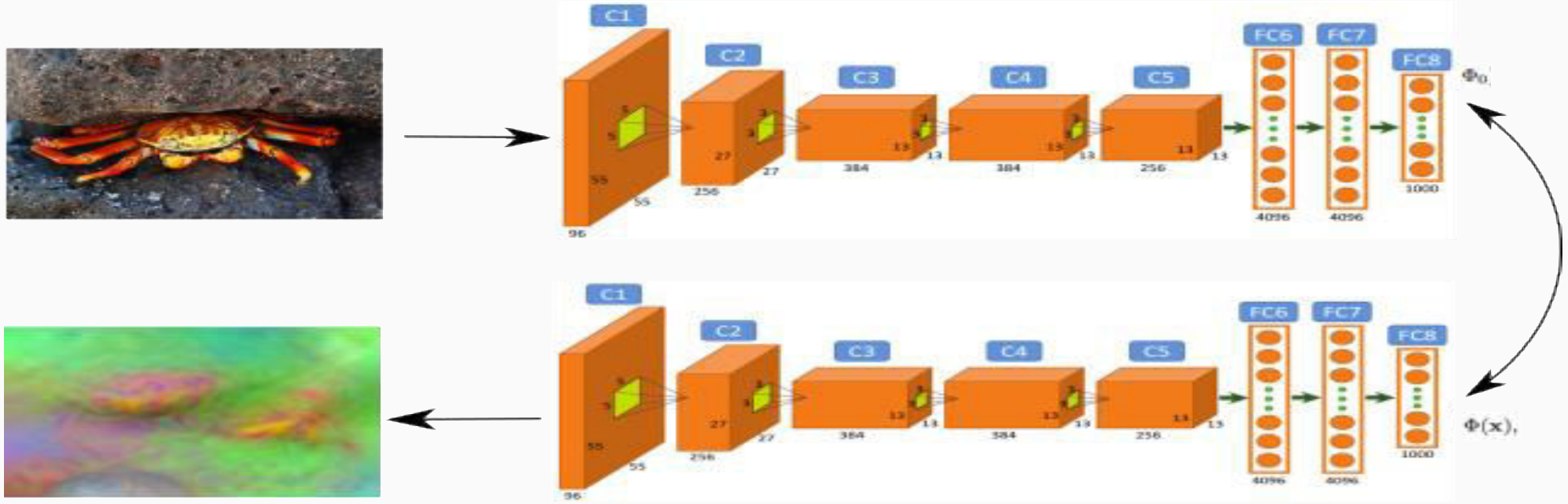
SPEECH

sample → spectral band → formant → motif → phone → word

NLP

character → word → NP/VP/.. → clause → sentence → story

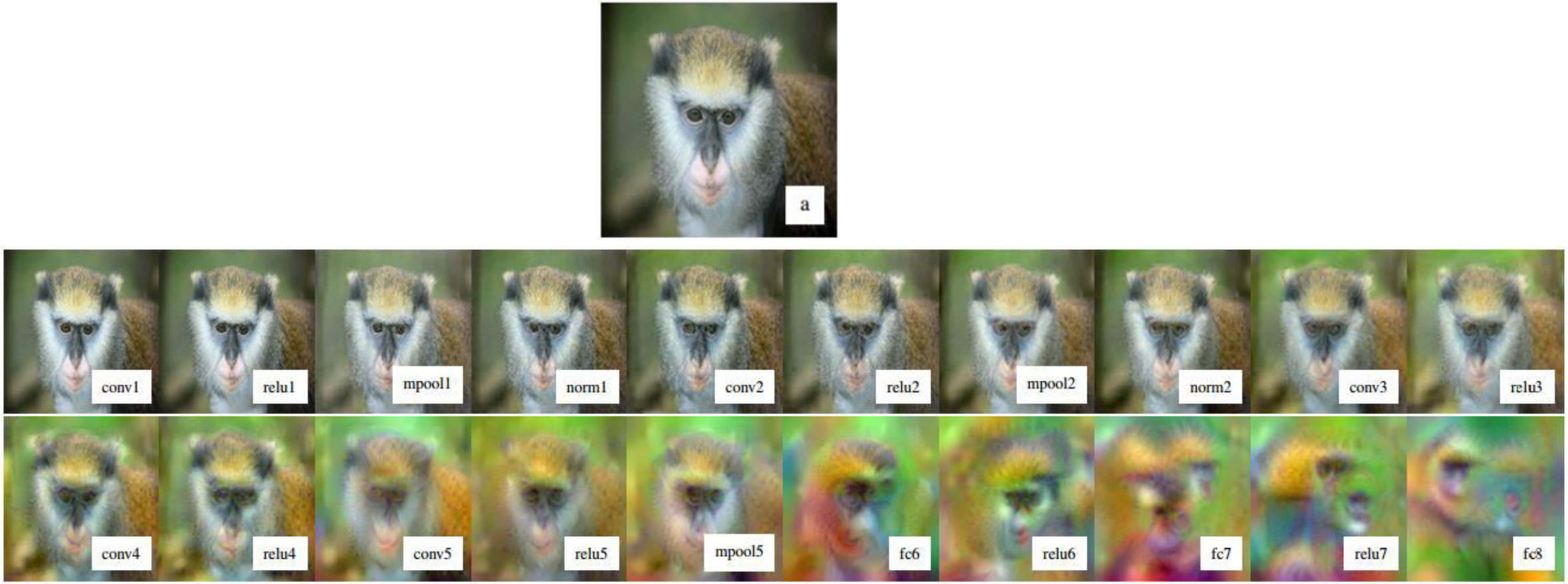
Inverting Specific Representation



$$x^* = \underset{x \in R^{H \times W \times C}}{\operatorname{argmin}} L(\varphi(x), \varphi_0) + \lambda R(x) \qquad (2)$$

Aravindh Mahendran and Andrea Vedaldi, Understanding Deep Image Representations by Inverting Them, CVPR'15

Inverting at Different Stages



Reconstructions from intermediate layers

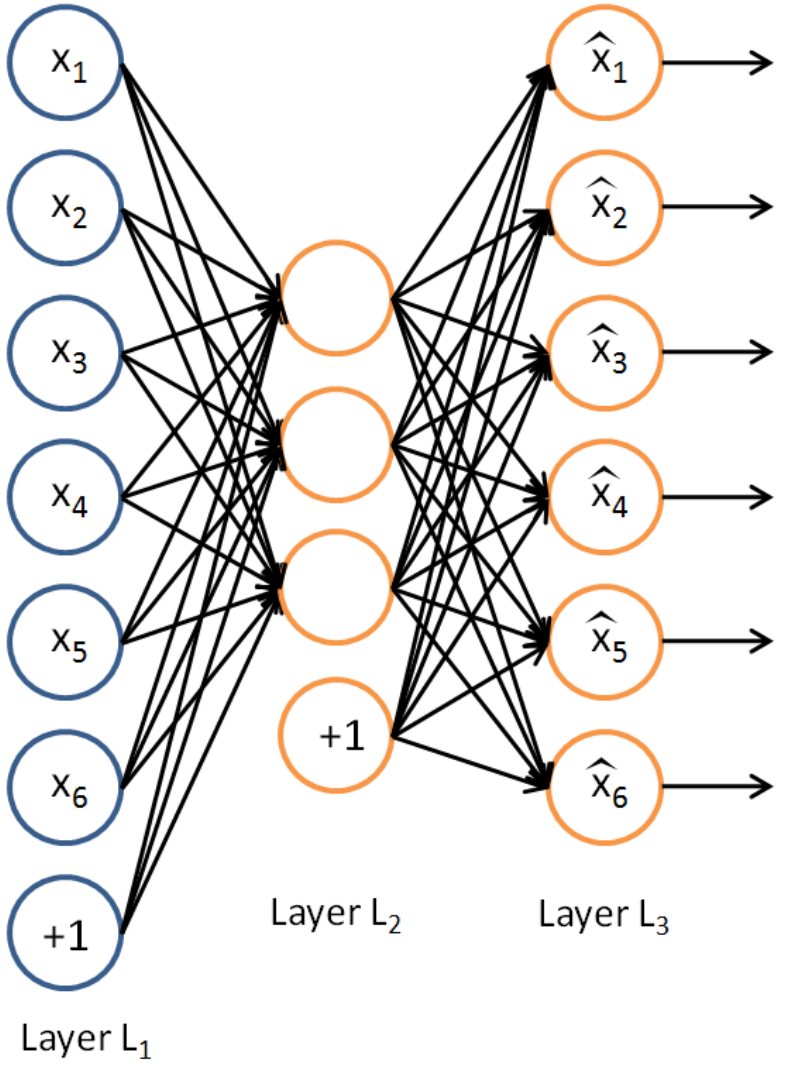
Questions?

Auto Encoder

What if we do not have labels?

Autoencoder

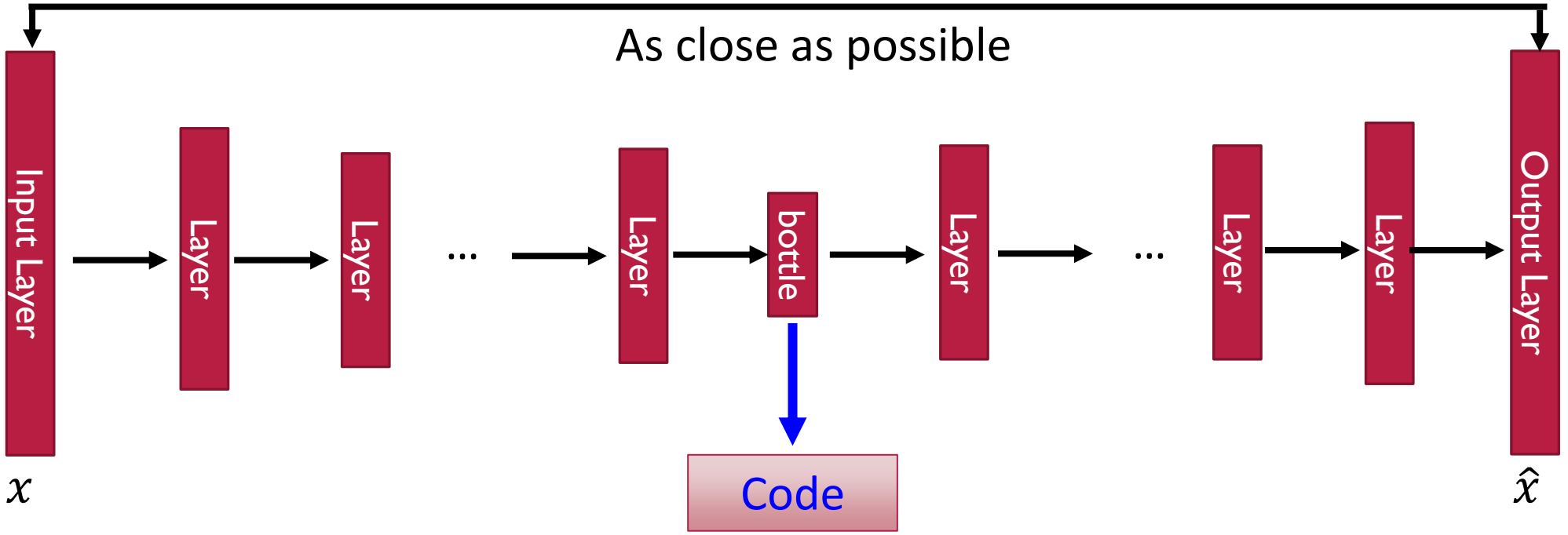
- Similar to MLP
- Input is same as output
- Network learns to reconstruct.
- "Bottleneck" layer learns a compact representation.



Deep Auto-encoder

- Of course, the auto-encoder can be deep

Symmetric is not necessary.



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

View Points

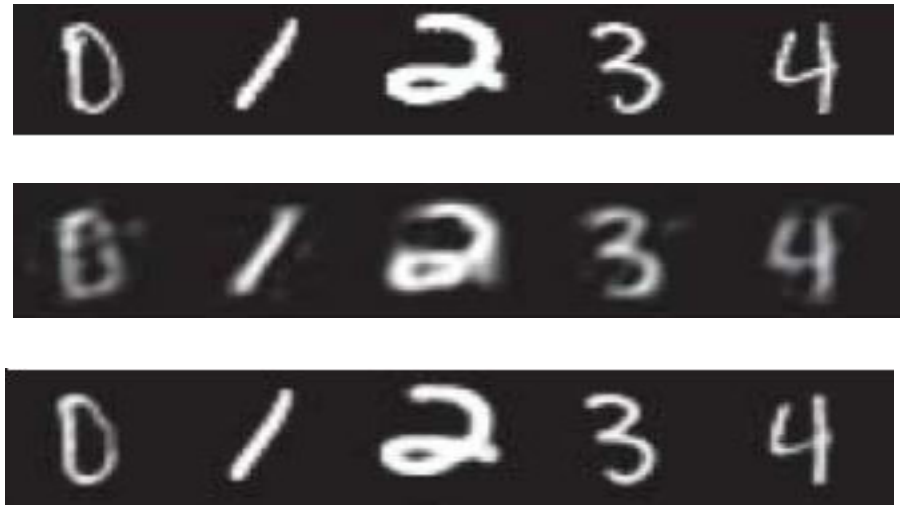
- Nonlinear PCA (Dimensionality Reduction)
- Unsupervised Learning
- Data compression

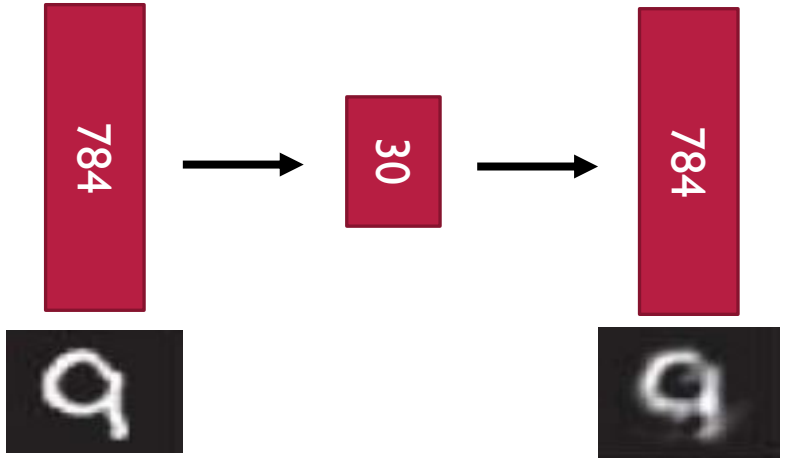
Deep Auto-encoder

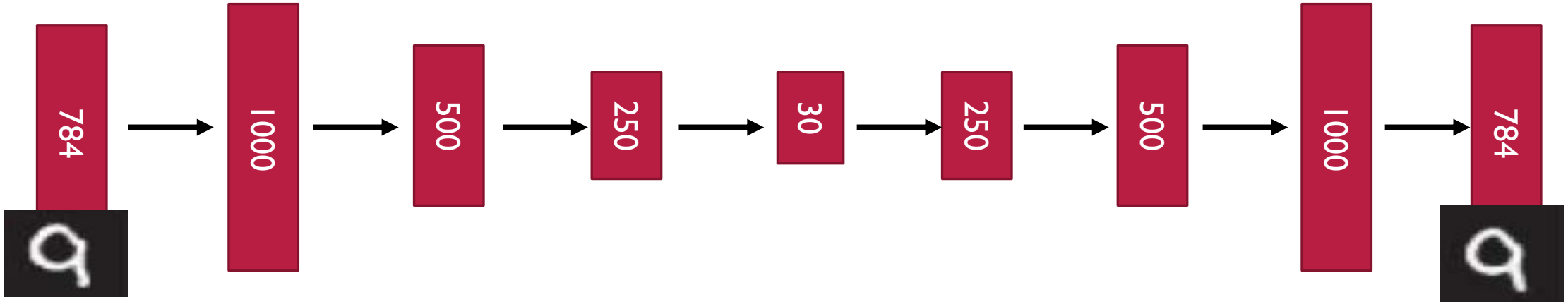
Original Image

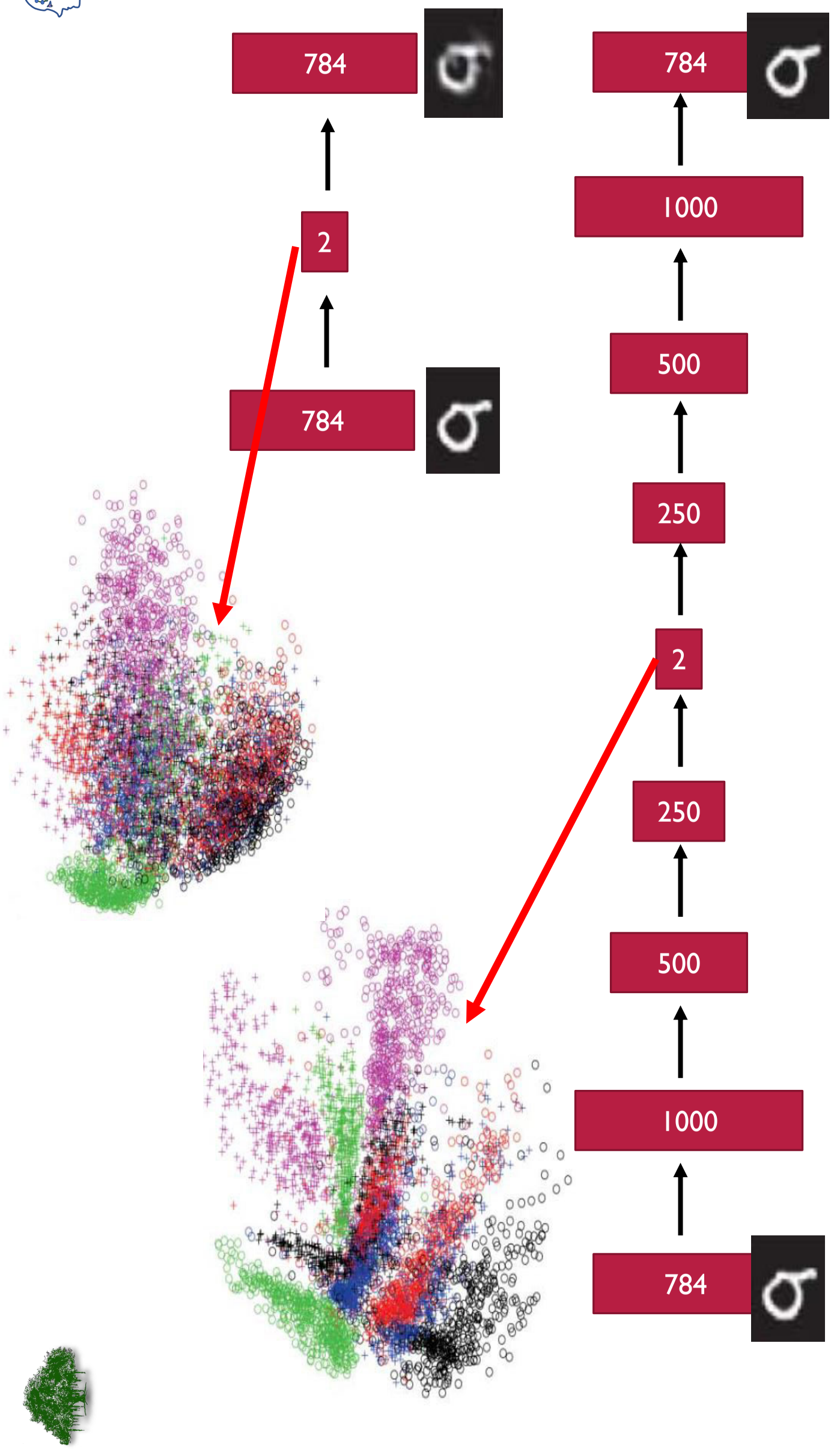
PCA

Deep Auto-encoder



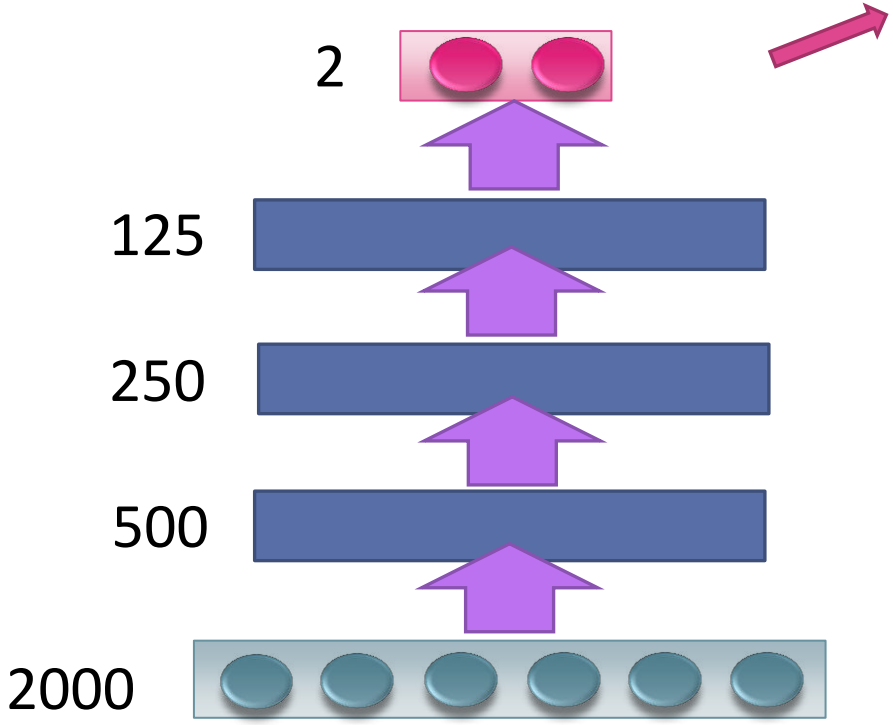




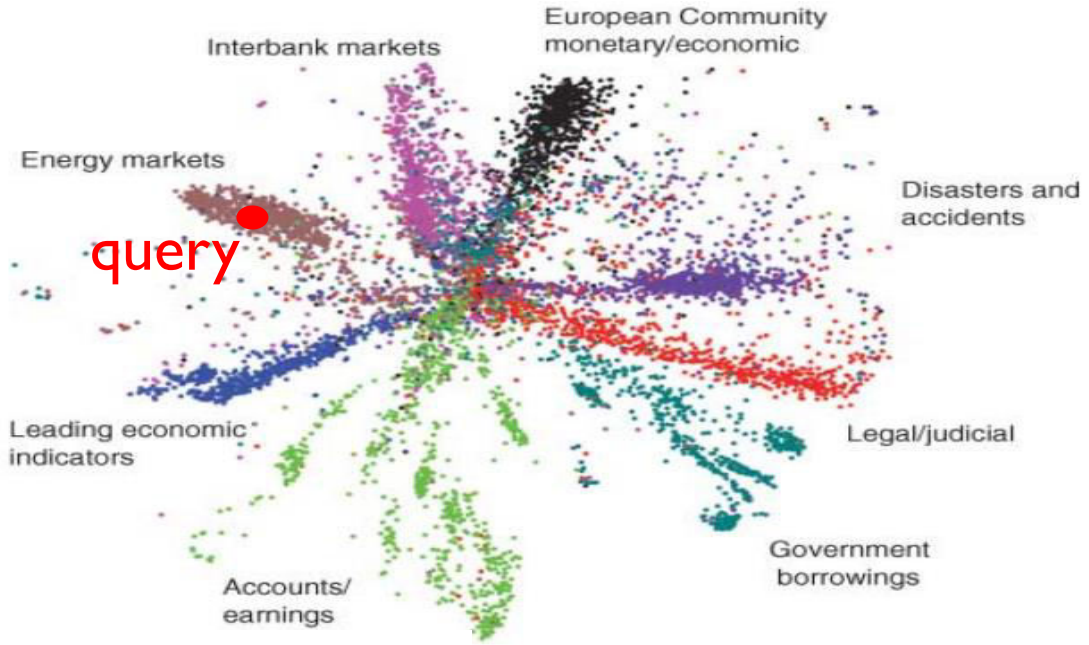

784
2
784
784
1000
500
250
2
250
500
1000
784

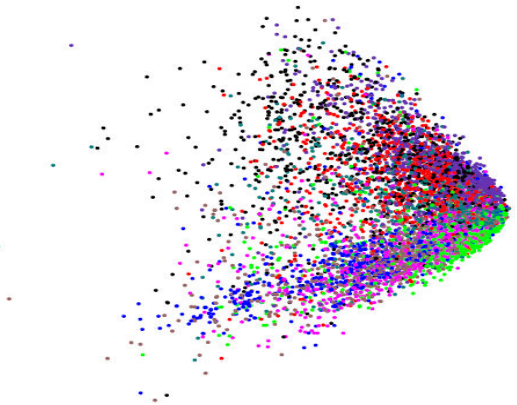
Auto-encoder – Text Retrieval

The documents talking about the same thing will have close code.



Bag-of-word
(document or query)





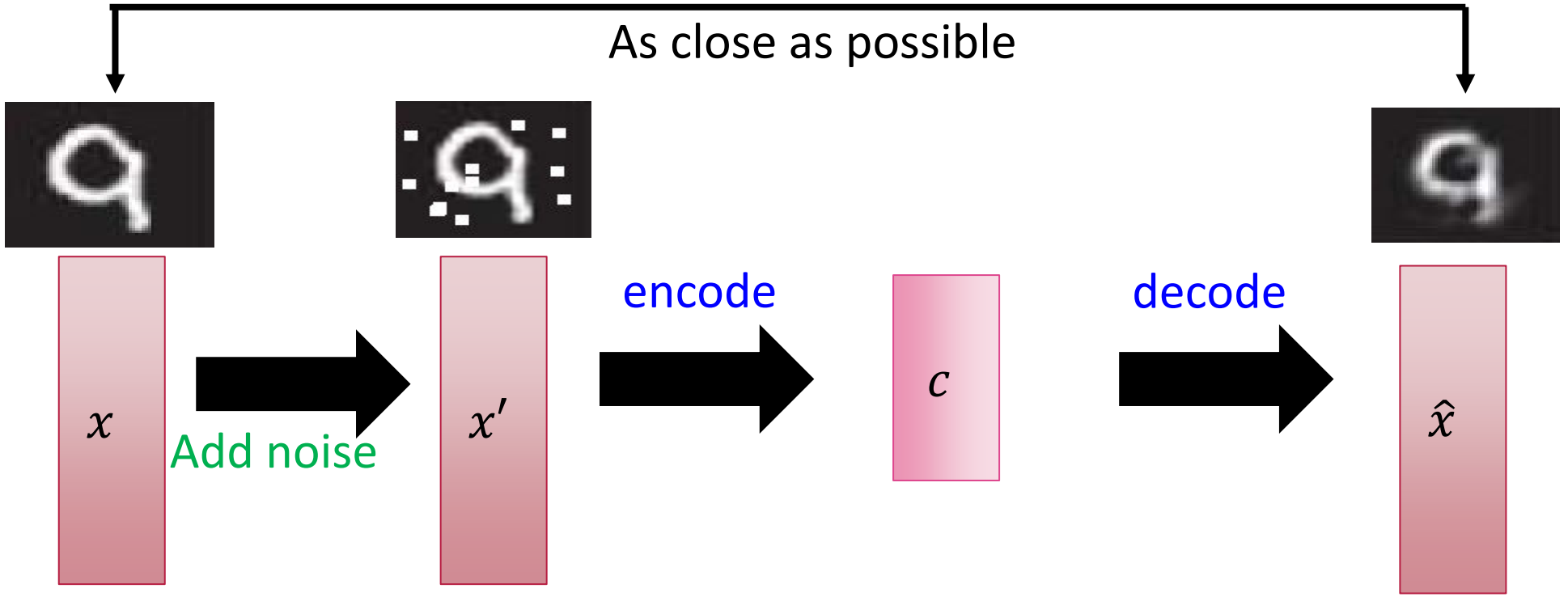
LSA: project documents to 2 latent topics

Variants

- Denoising Auto encoder
- Sparse Auto encoder

Auto-encoder

- De-noising auto-encoder



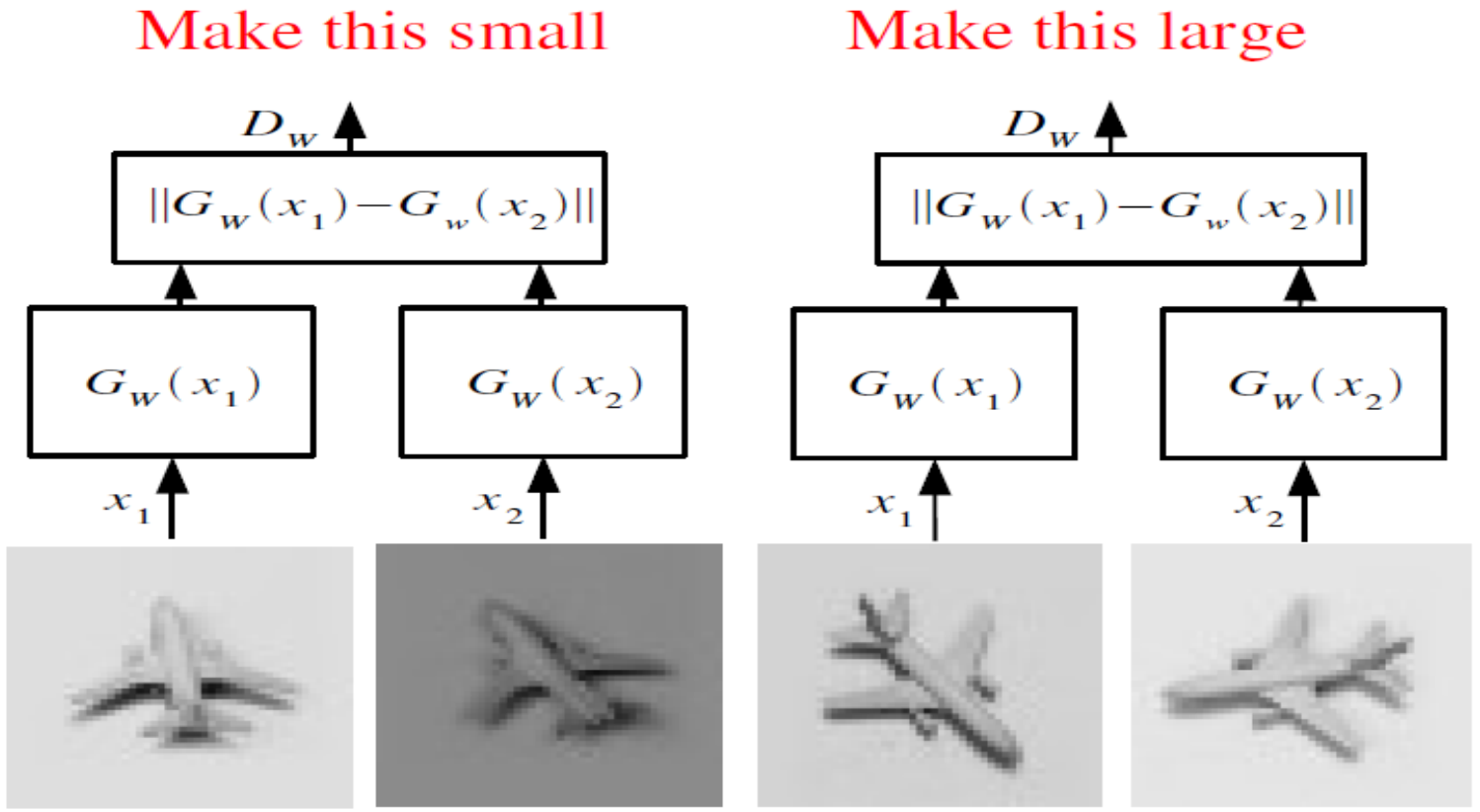
Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML*, 2008.

Questions?

Siamese Networks

Learning from Pairs (Triplets)

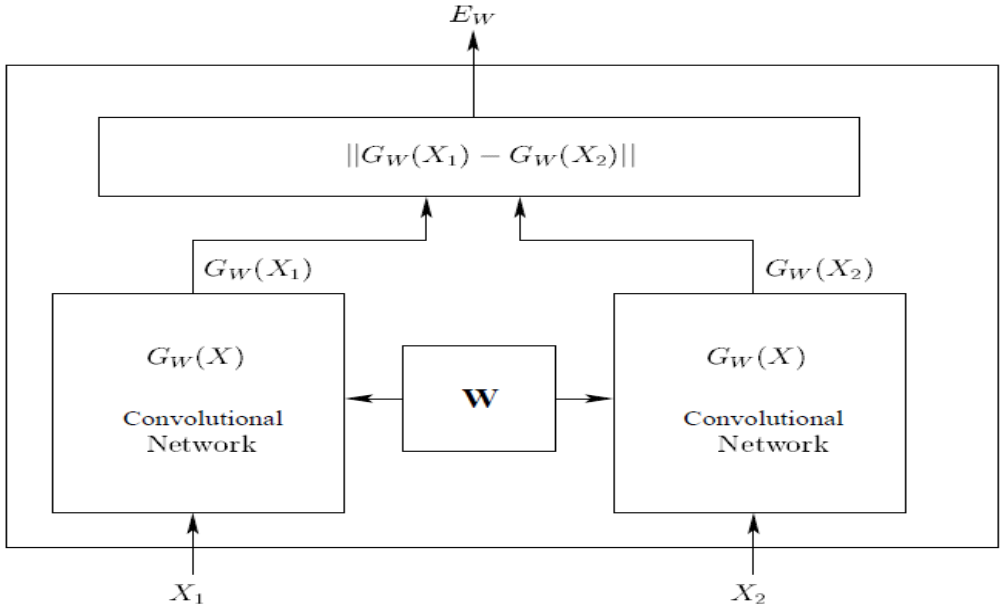
Siamese: Loss



Siamese Architecture

- Given a family of functions $G_W(X)$ parameterized by W, find W such that the similarity metric $D_W(X1, X2)$ is small for similar pairs and large for dissimilar pairs:-

$$D_W(X_1, X_2) = ||G_W(X_1) - G_W(X_2)||$$



Loss function

$$\mathcal{L}(W) = \sum_{i=1}^{P} L(W, (Y, \vec{X}_1, \vec{X}_2)^i)$$

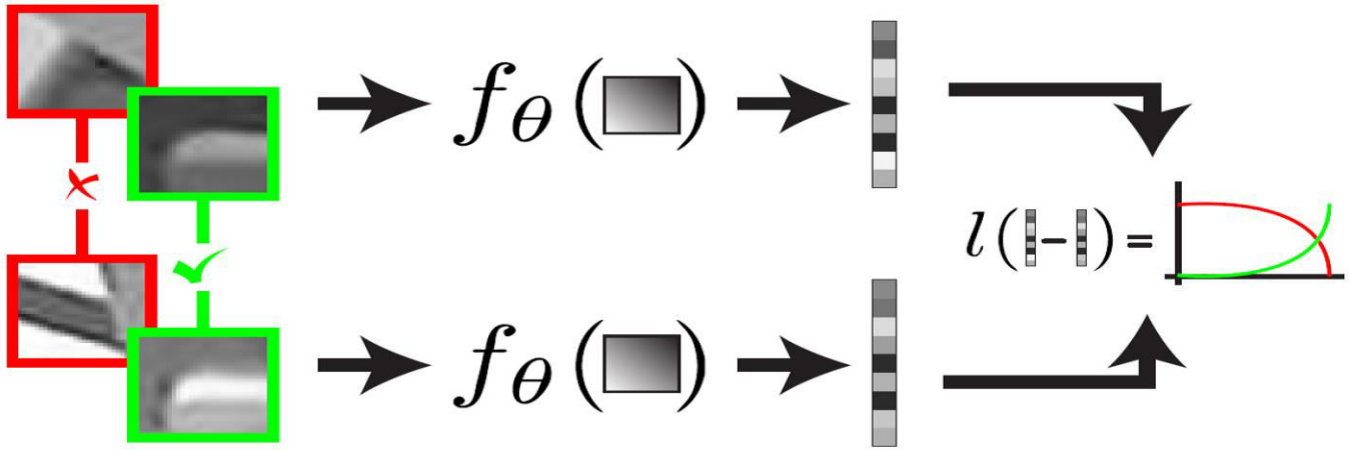
Loss function for similar pairs

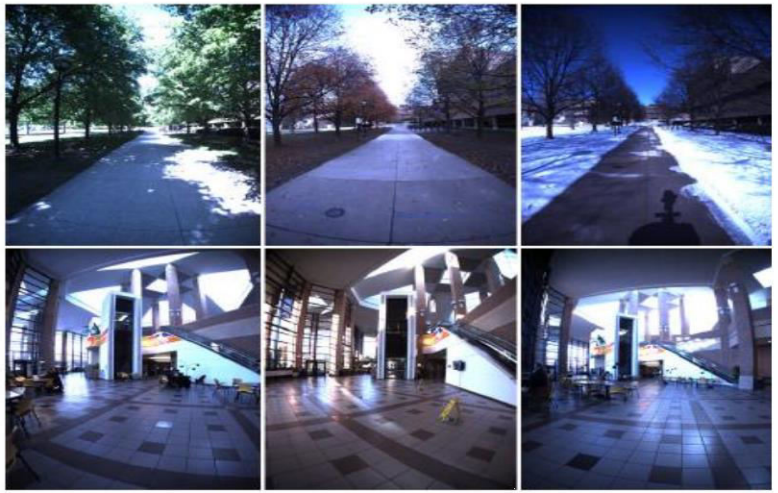
$$L(W, (Y, \vec{X}_1, \vec{X}_2)^i) = (1 - Y)L_S\left(D_W^i\right) + YL_D\left(D_W^i\right)$$

Loss function for dissimilar pairs

Raia Hadsell, Sumit Chopra, Yann LeCun, Dimensionality Reduction by Learning an Invariant Mapping. CVPR 2006

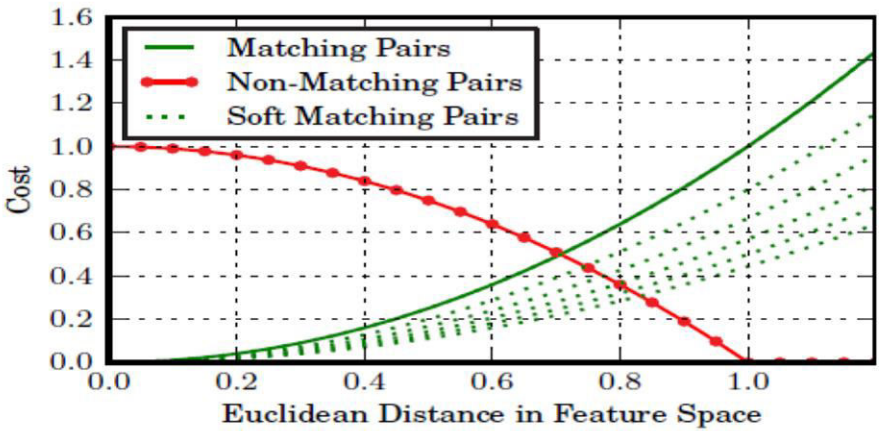
Learning to Match





Using the contrastive cost function

$$l_{\boldsymbol{\theta}}\left(\mathbf{y}_i, \mathbf{y}_j\right) = \begin{cases} s_{ij} d_{ij}^2, & \text{if matching} \\ \max\left(1.0 - d_{ij}^2,\ 0\right), & \text{if non-matching} \end{cases}$$



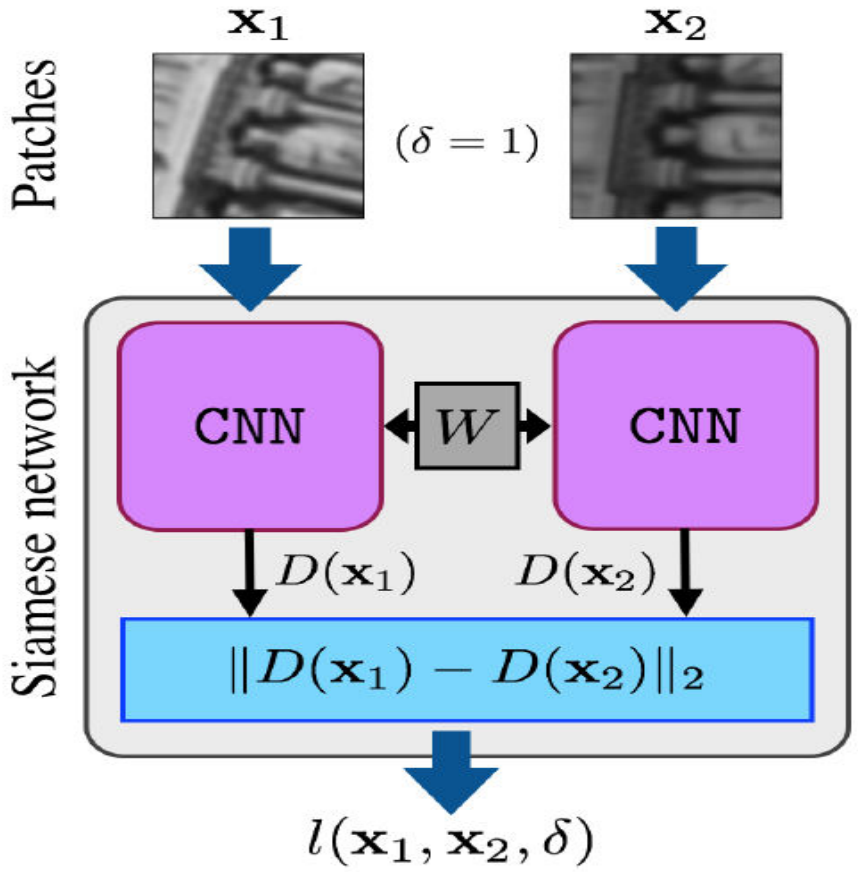
Learning to Match

$$d_D(\mathbf{x}_1, \mathbf{x}_2) = \|D(\mathbf{x}_1) - D(\mathbf{x}_2)\|_2$$

$$l(\mathbf{x}_1, \mathbf{x}_2, \delta) = \delta \cdot l_P(d_D(\mathbf{x}_1, \mathbf{x}_2)) + (1 - \delta) \cdot l_N(d_D(\mathbf{x}_1, \mathbf{x}_2))$$

$$l_P(d_D(\mathbf{x}_1, \mathbf{x}_2)) = d_D(\mathbf{x}_1, \mathbf{x}_2)$$

$$l_N(d_D(\mathbf{x}_1, \mathbf{x}_2)) = \max(0, m - d_D(\mathbf{x}_1, \mathbf{x}_2))$$



Stochastic sampling and agrressive mining

ICLR 2014

Triplet Loss

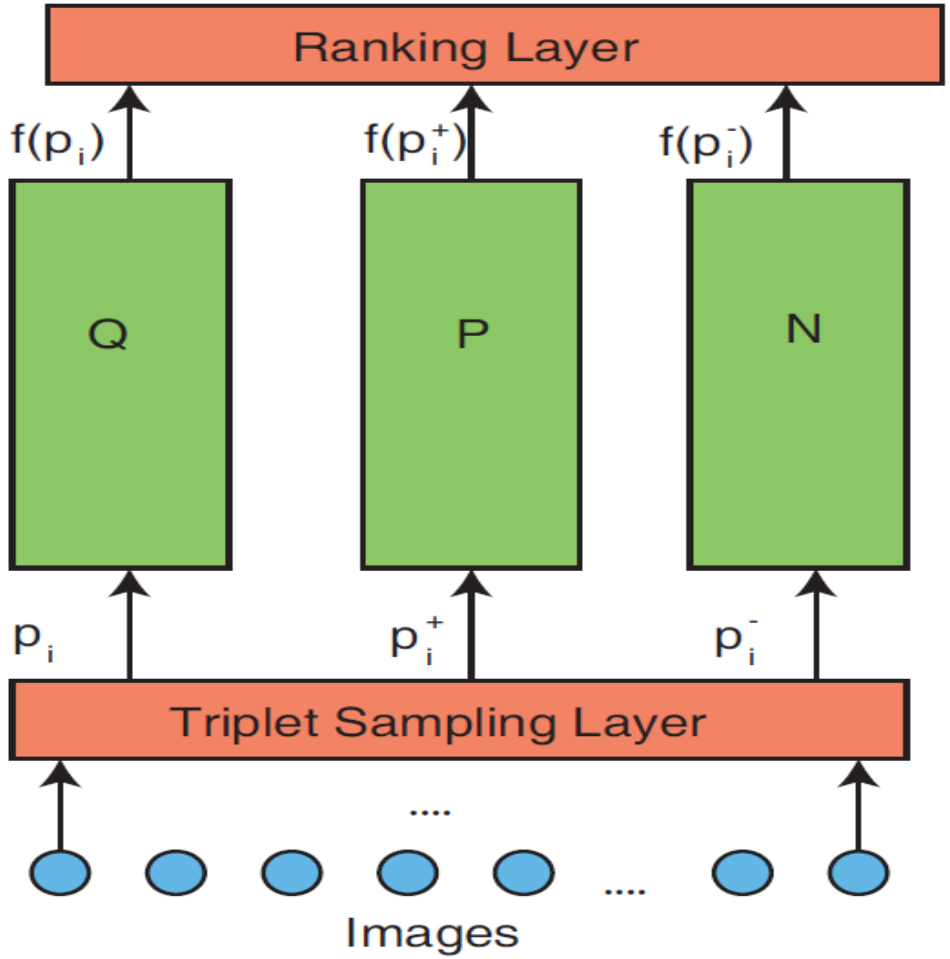


Image Retrieval (Ranking)

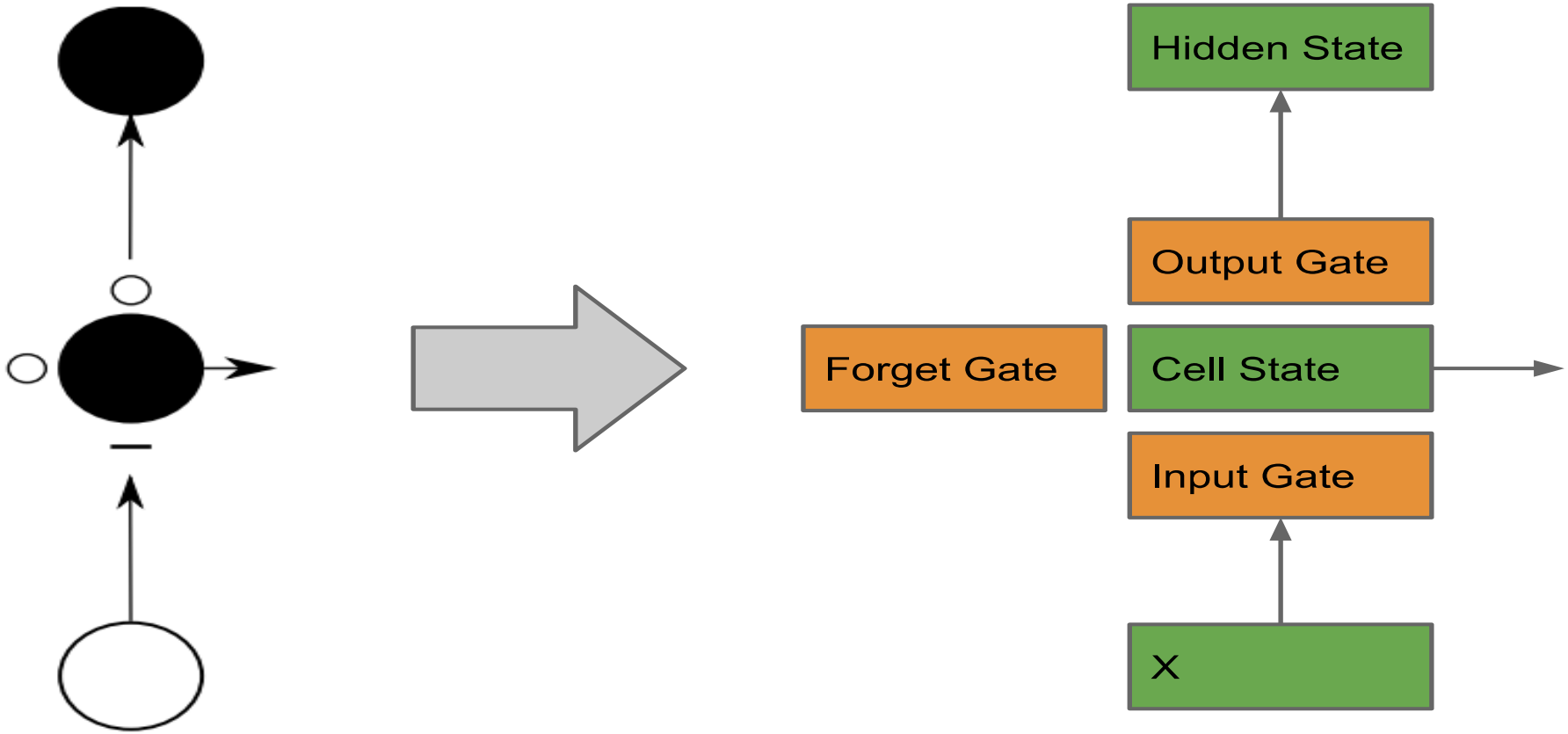
Query					
Positive					
Negative					

Questions?

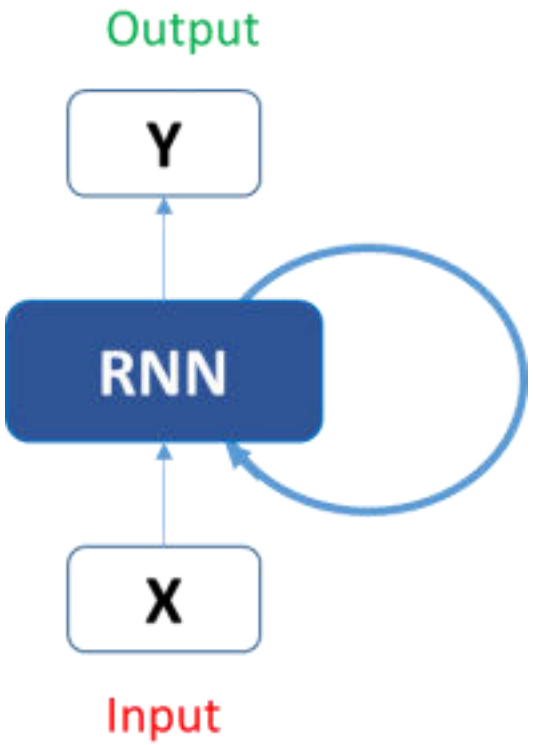
Recurrent Neural Networks

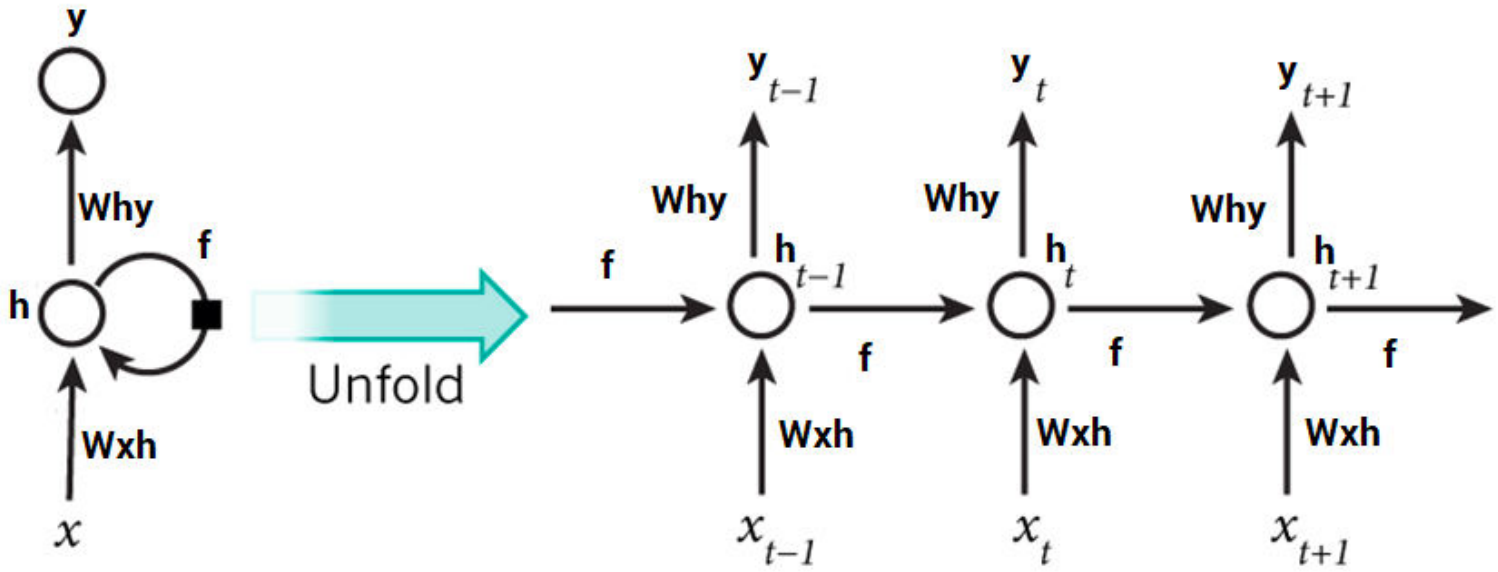
Architecture and Training

Cell with Memory



RNNs: Overview and Summary: Q?

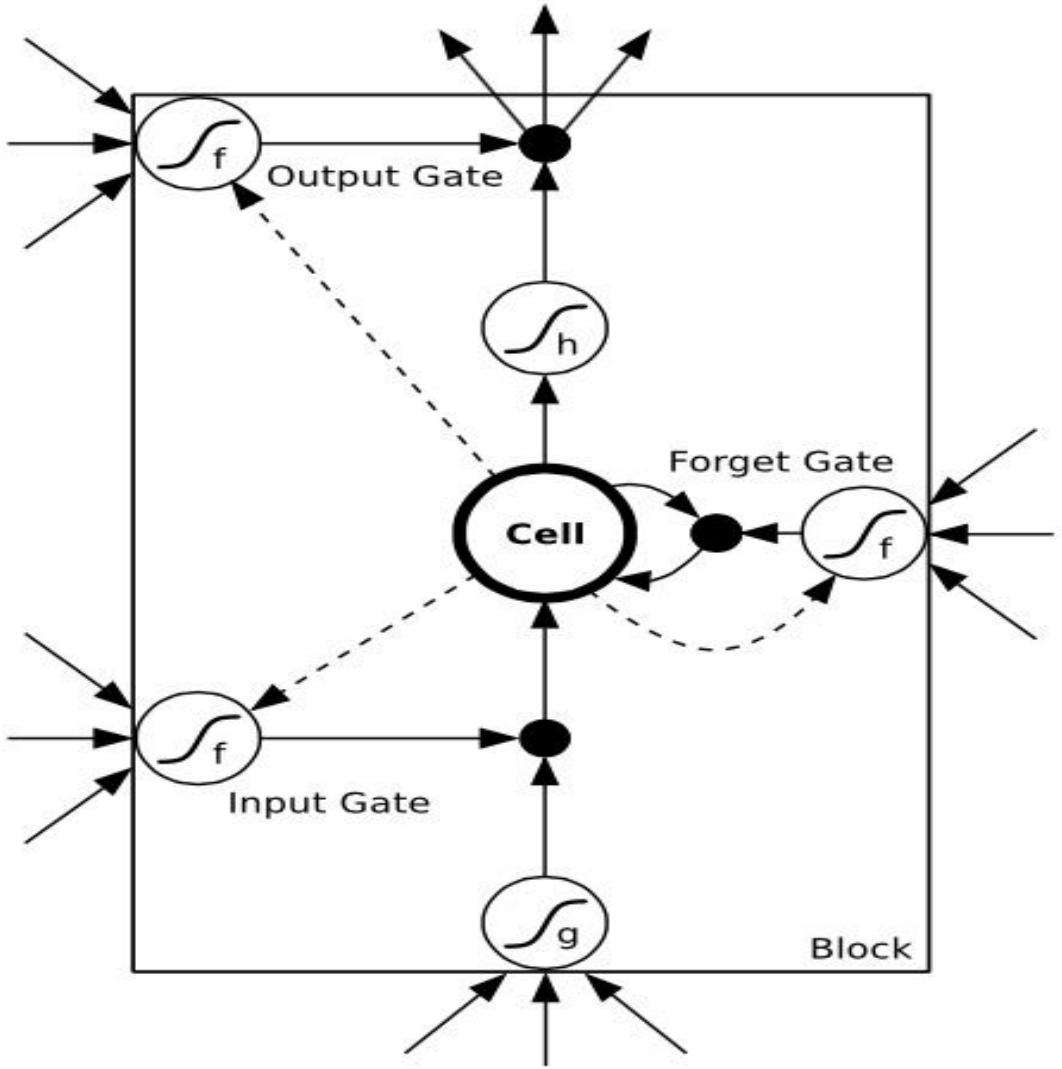




$$h_t = \tanh (W_{hh} h_{t-1} + W_{xh} x_t)$$

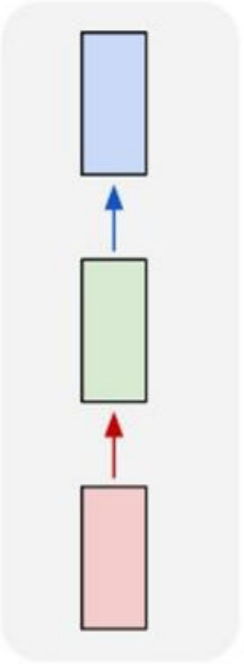
$$y_t = W_{hy} h_t$$

LSTM: Gates



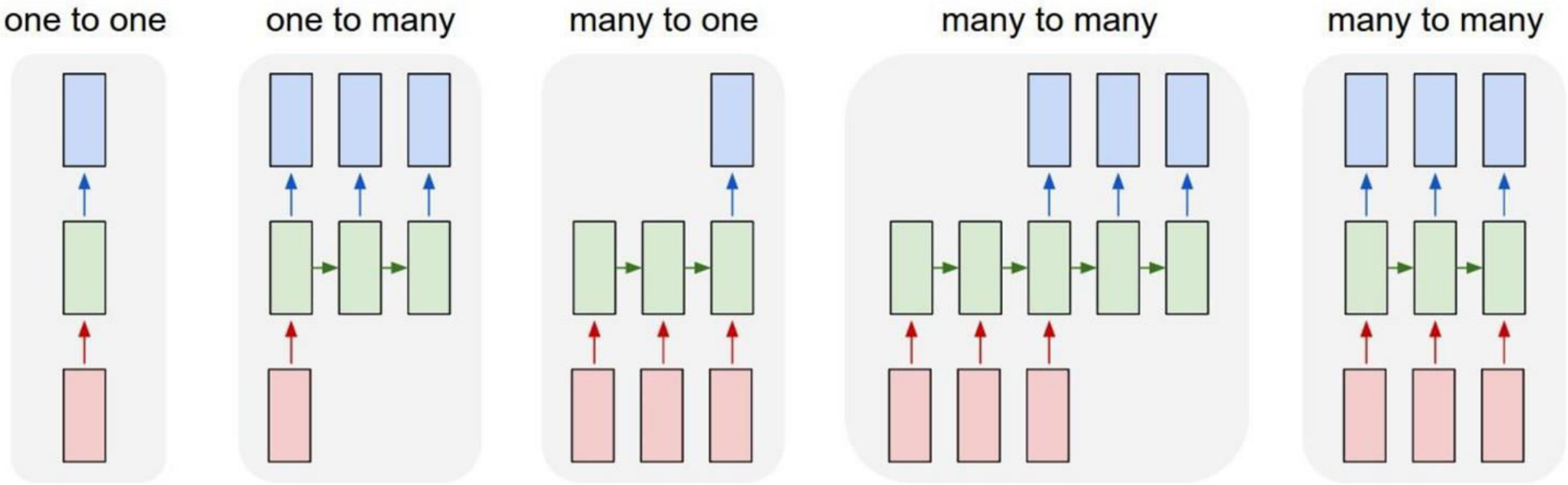
Recurrent Neural Networks (RNNs)

one to one



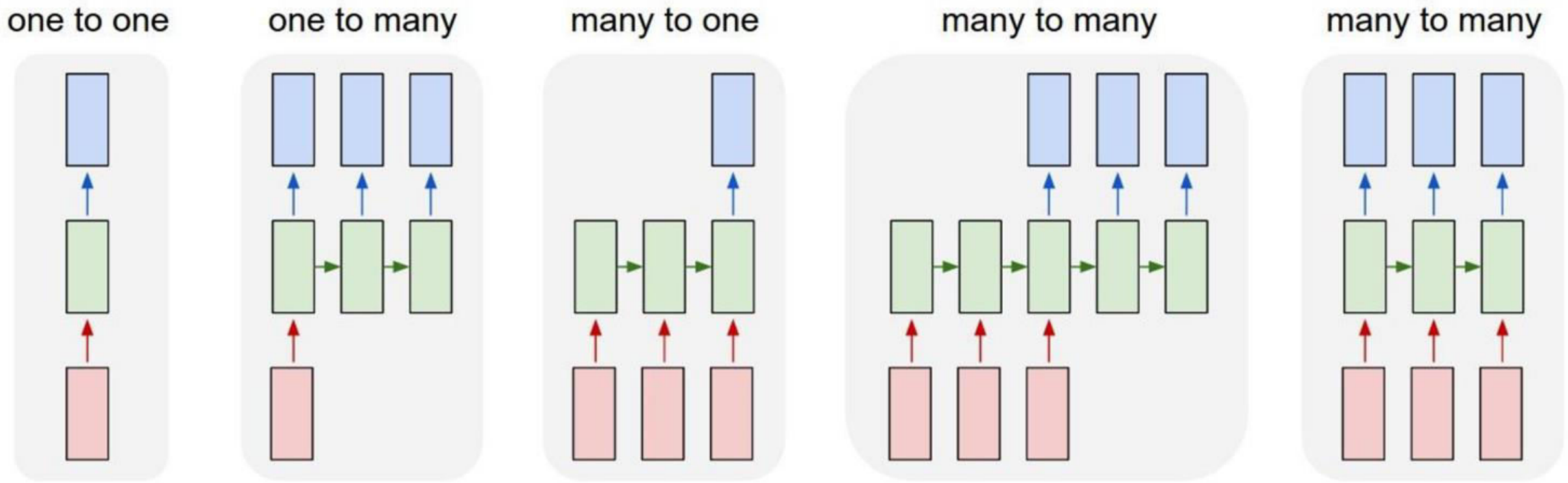
Vanilla Neural Networks

RNNs : Process Sequence



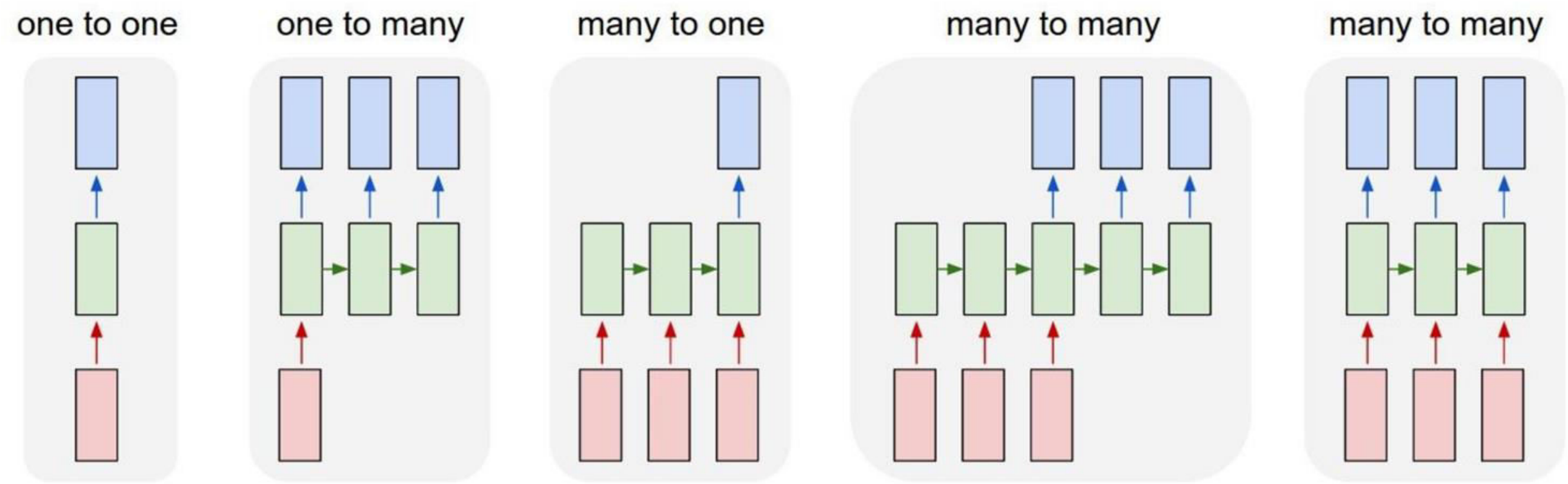
e.g. **Image Captioning**
image -> sequence of words

RNNs : Process Sequence



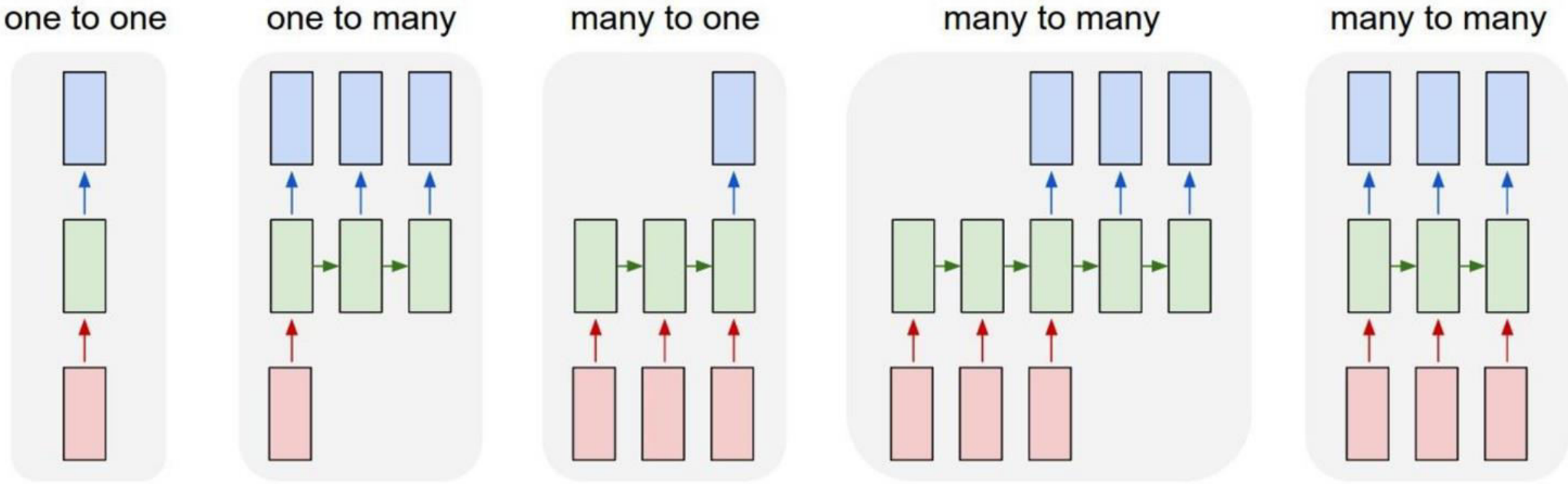
e.g. **Sentiment Classification**
sequence of words -> sentiment

RNNs : Process Sequence



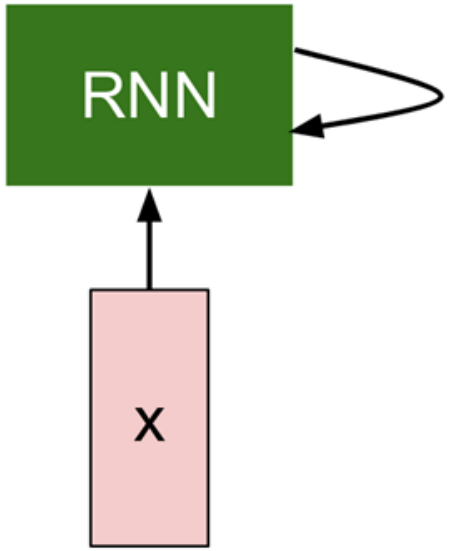
e.g. **Machine Translation**
seq of words -> seq of words

RNNs : Process Sequence

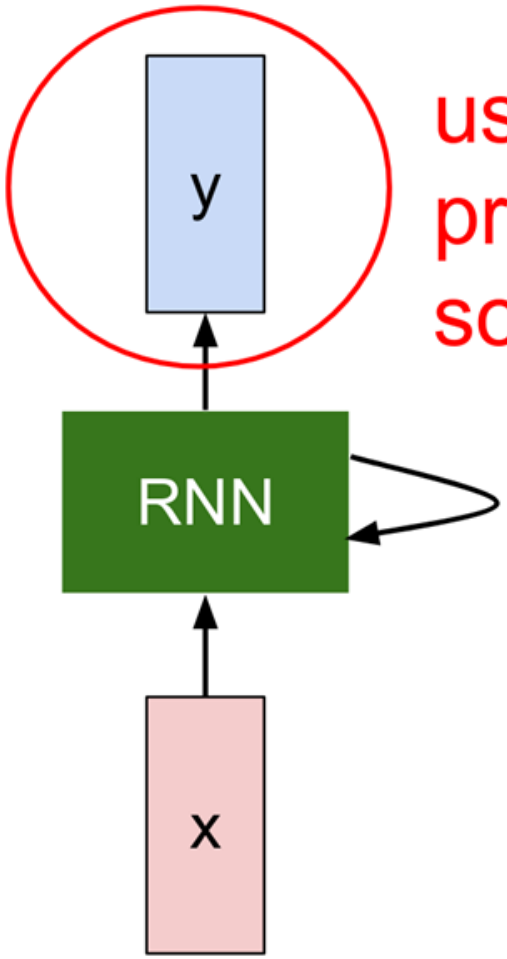


e.g. **Video classification on frame level**

RNNs



RNNs



usually want to predict a vector at some time steps

RNNs

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

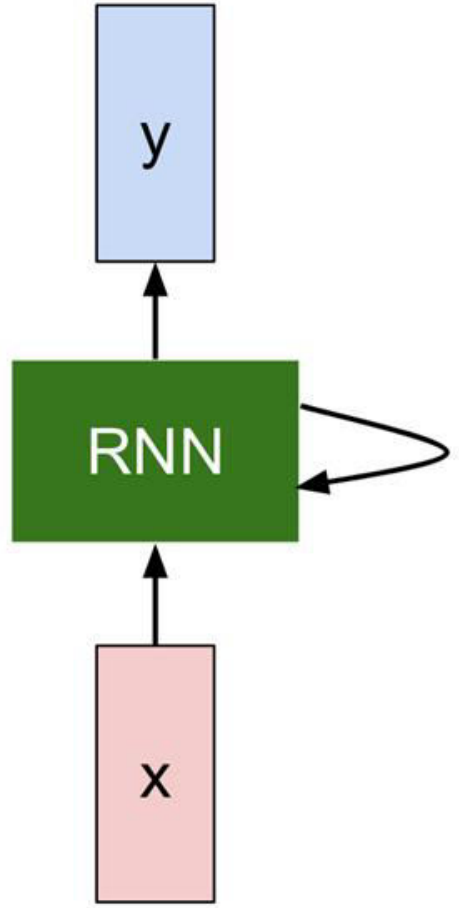
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

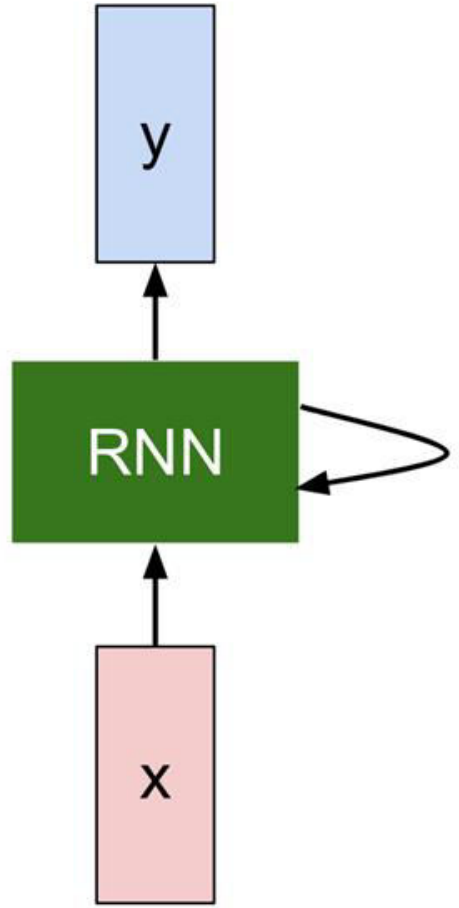
input vector at some time step



RNNs

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

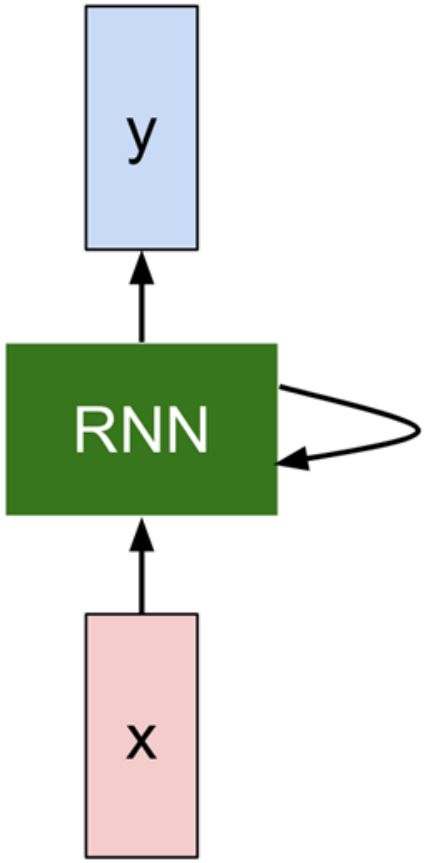
$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

RNNs

The state consists of a single *"hidden"* vector **h**:

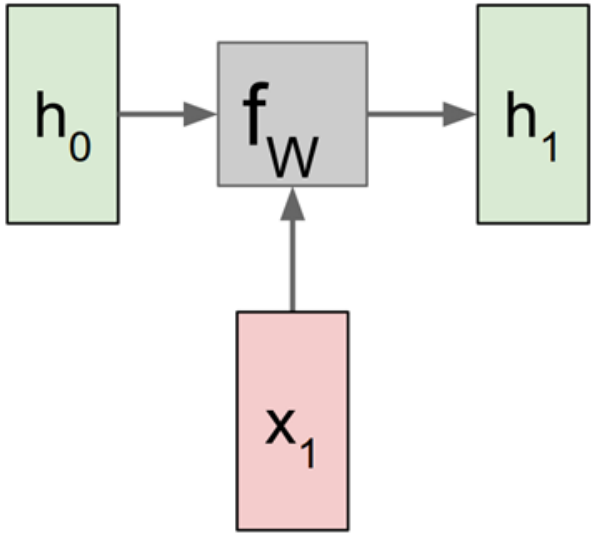


$$h_t = f_W(h_{t-1}, x_t)$$

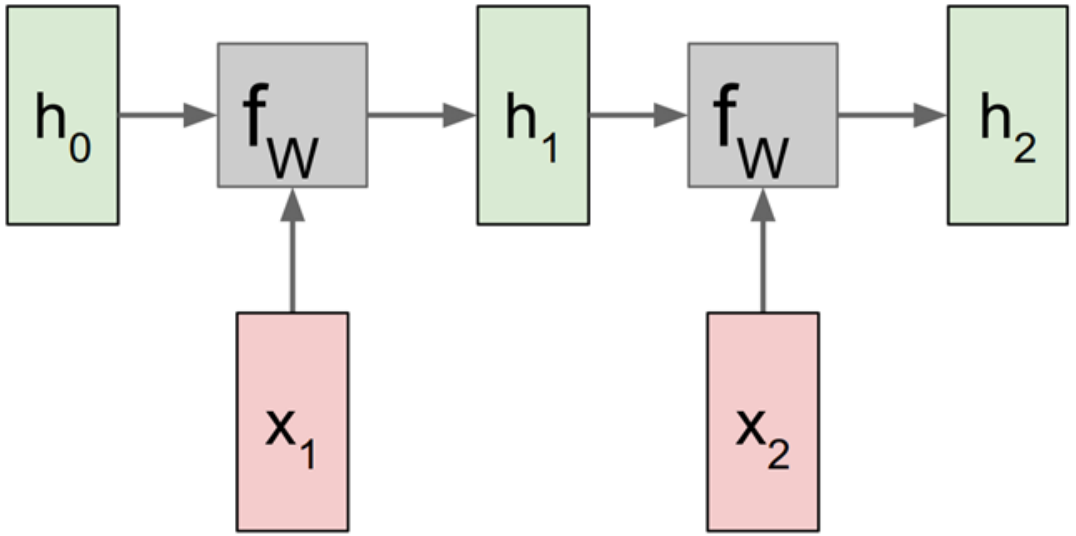
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

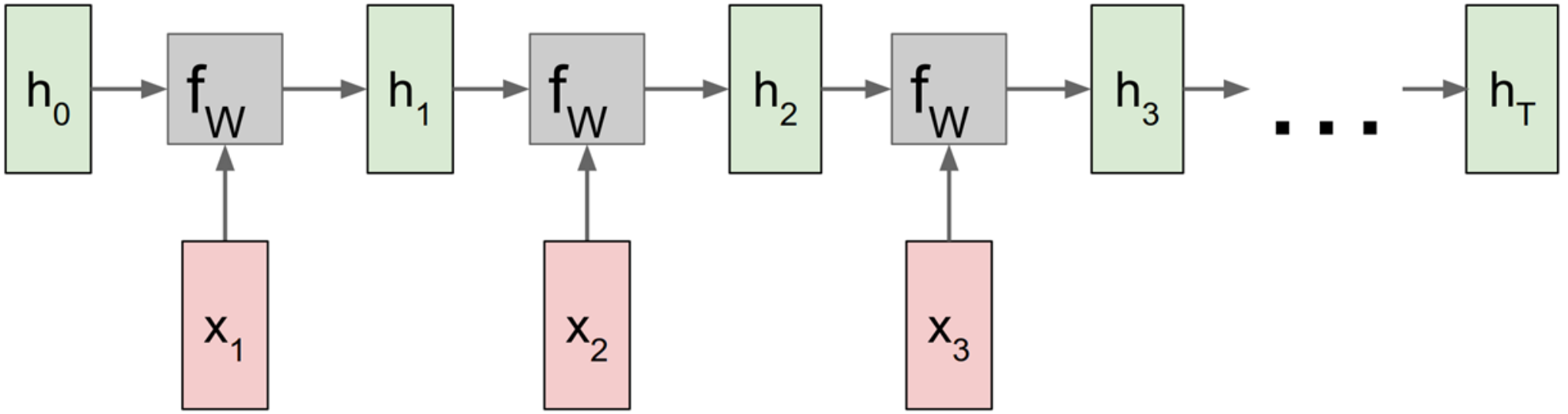
RNN : Computation Graph



RNN : Computation Graph

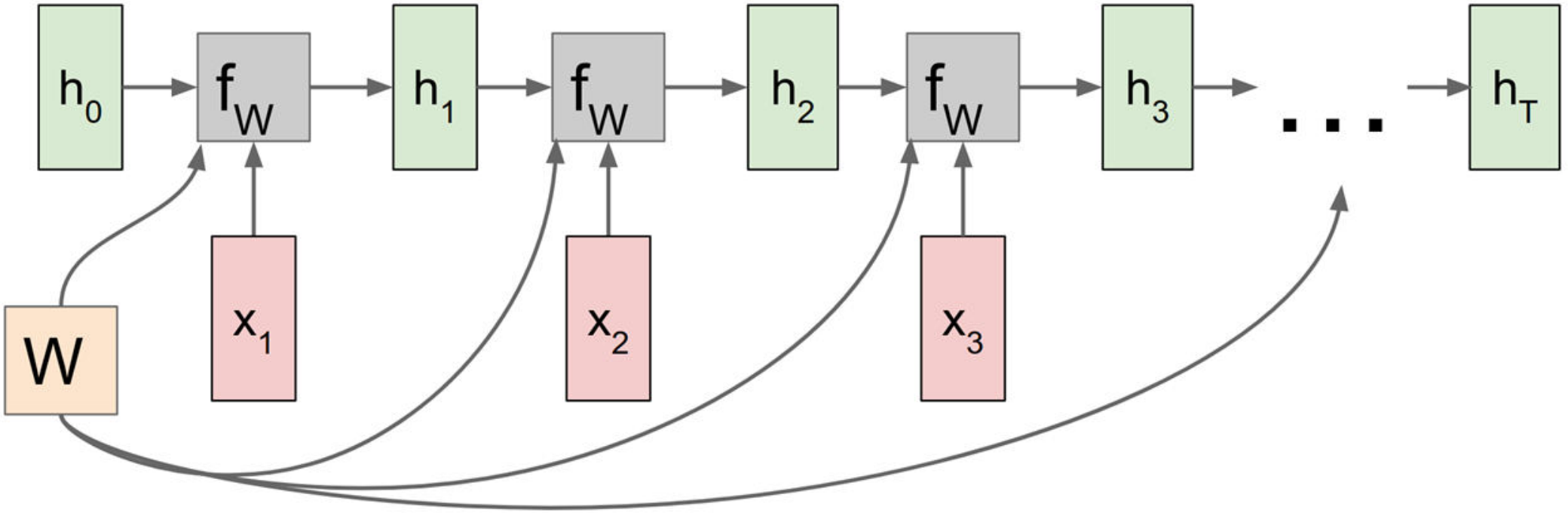


RNN : Computation Graph

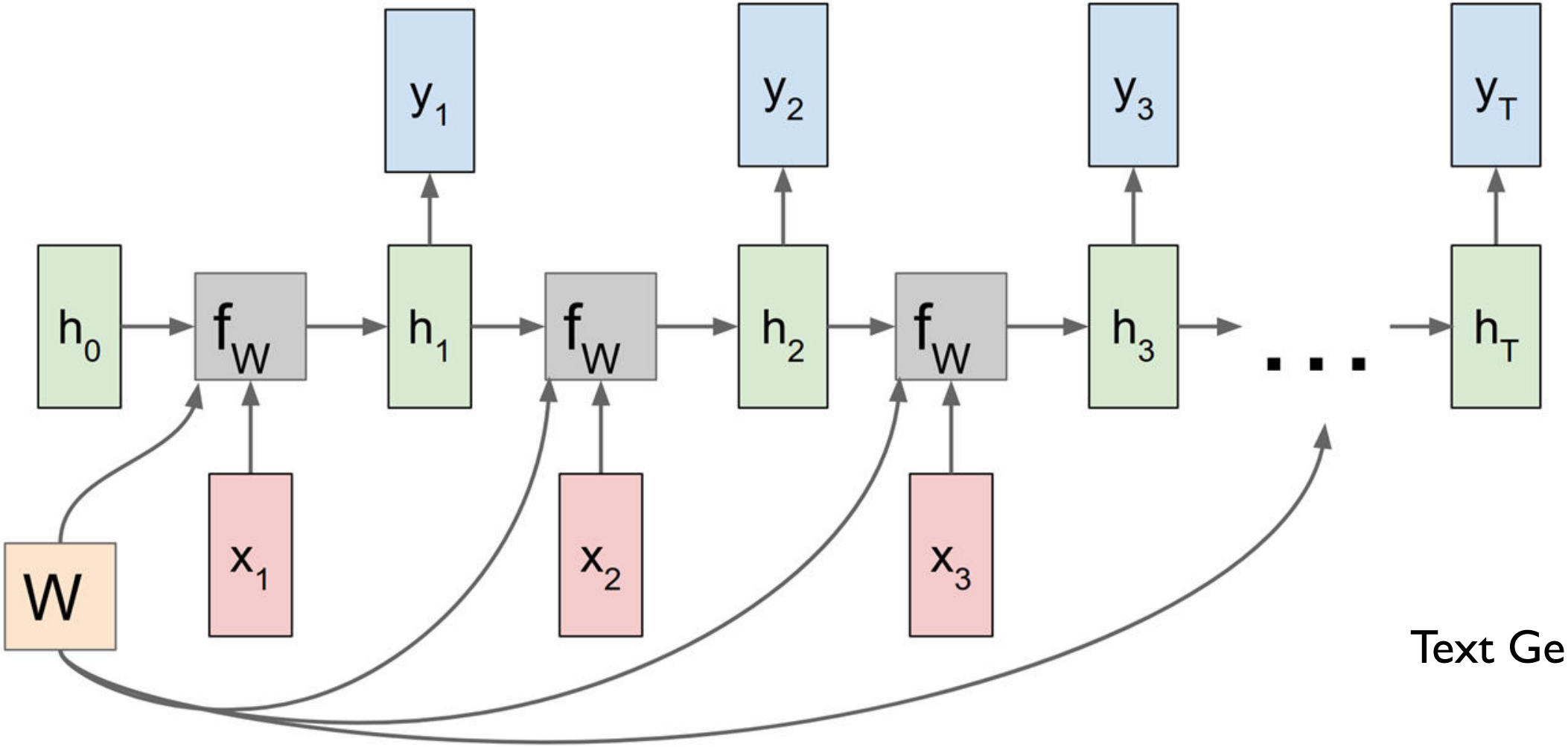


RNN : Computation Graph

Re-use the same weight matrix at every time-step

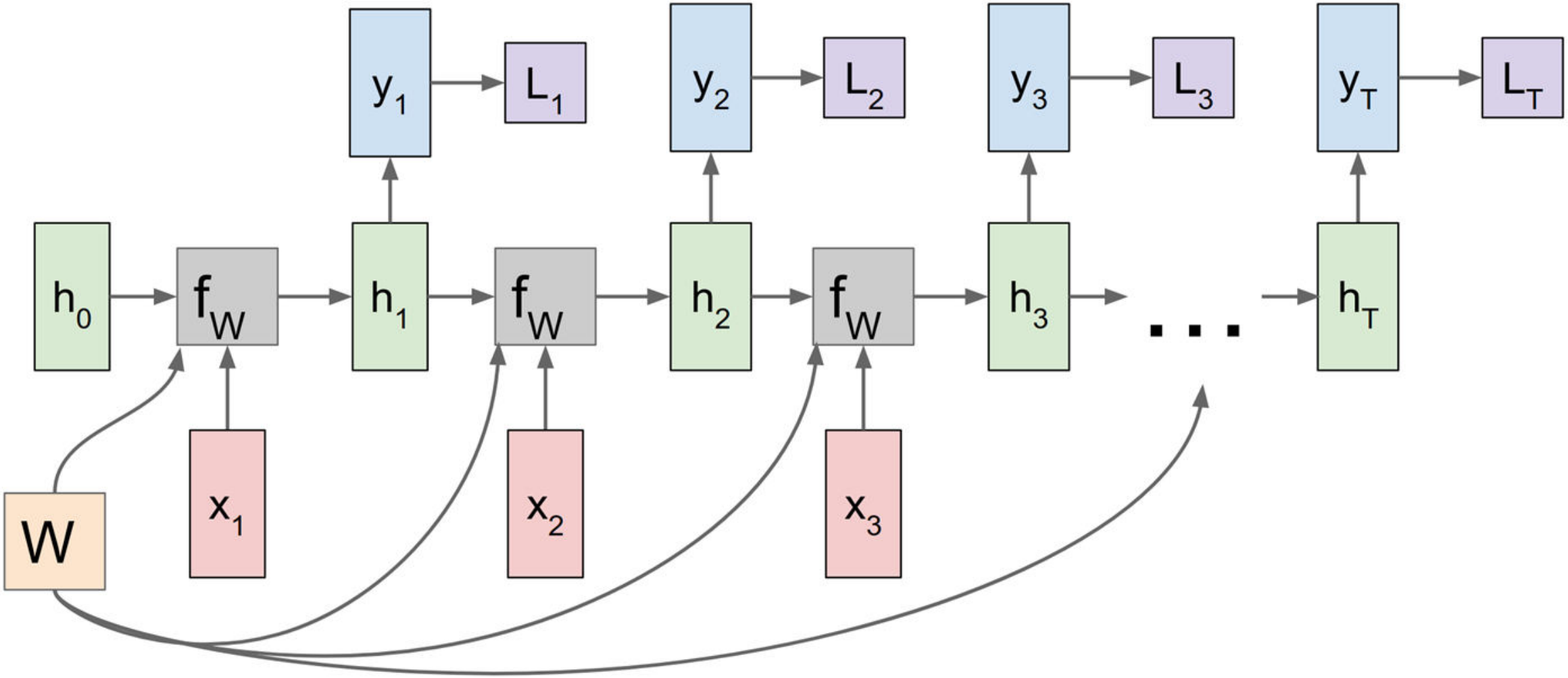


RNN: Many to Many

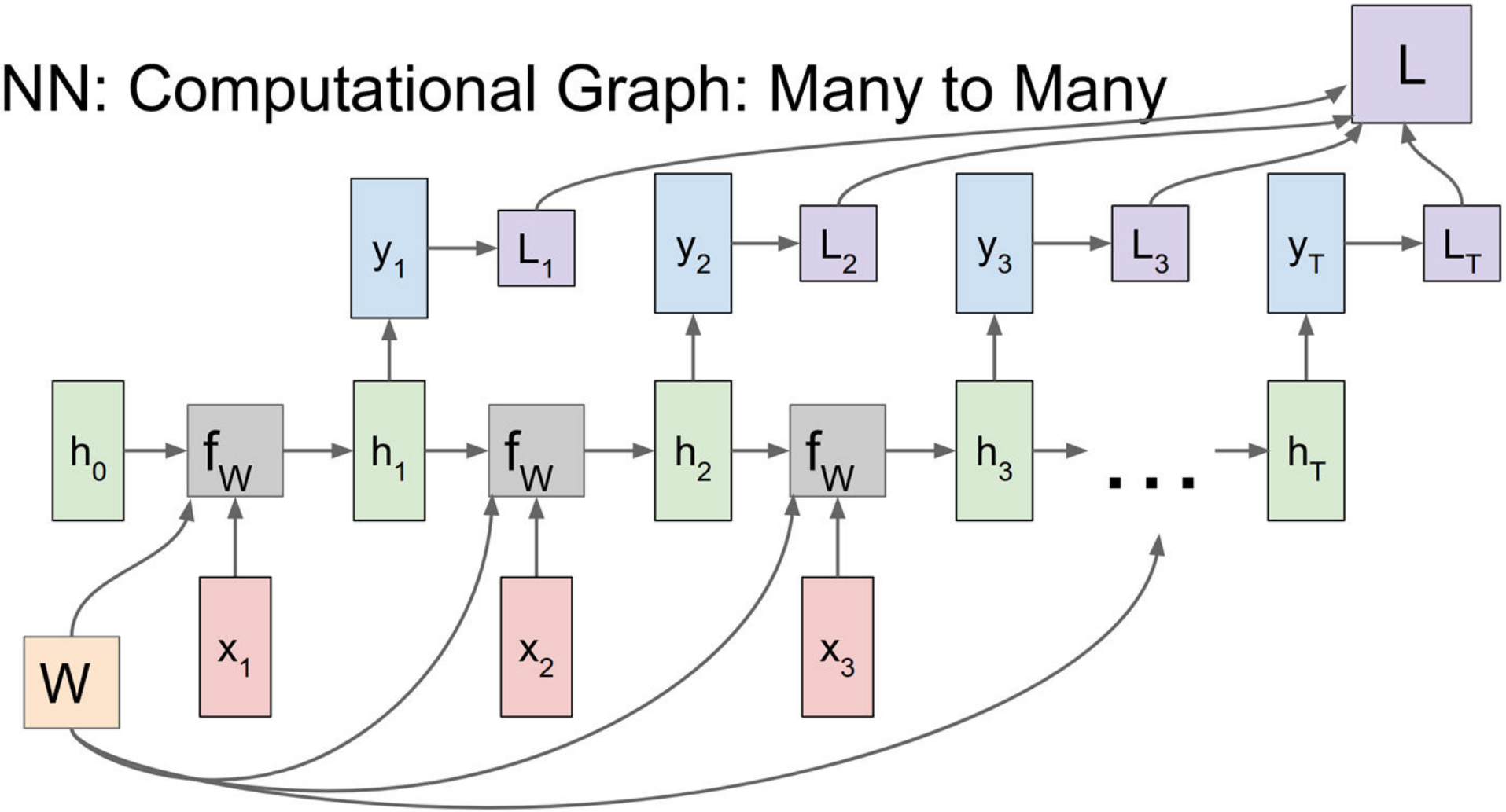


Text Generation

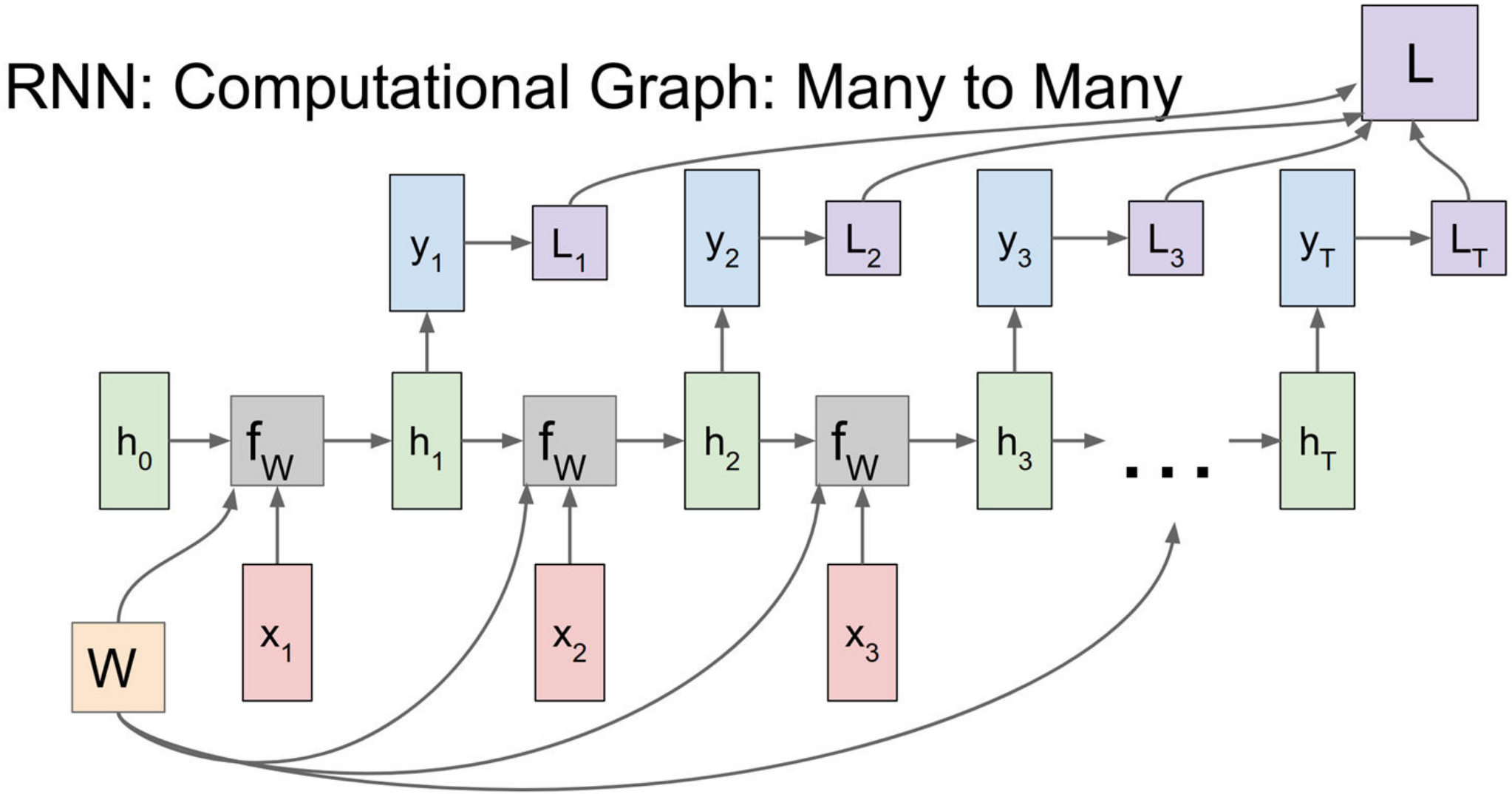
Loss Function for Many to Many



Loss Function for Many to Many

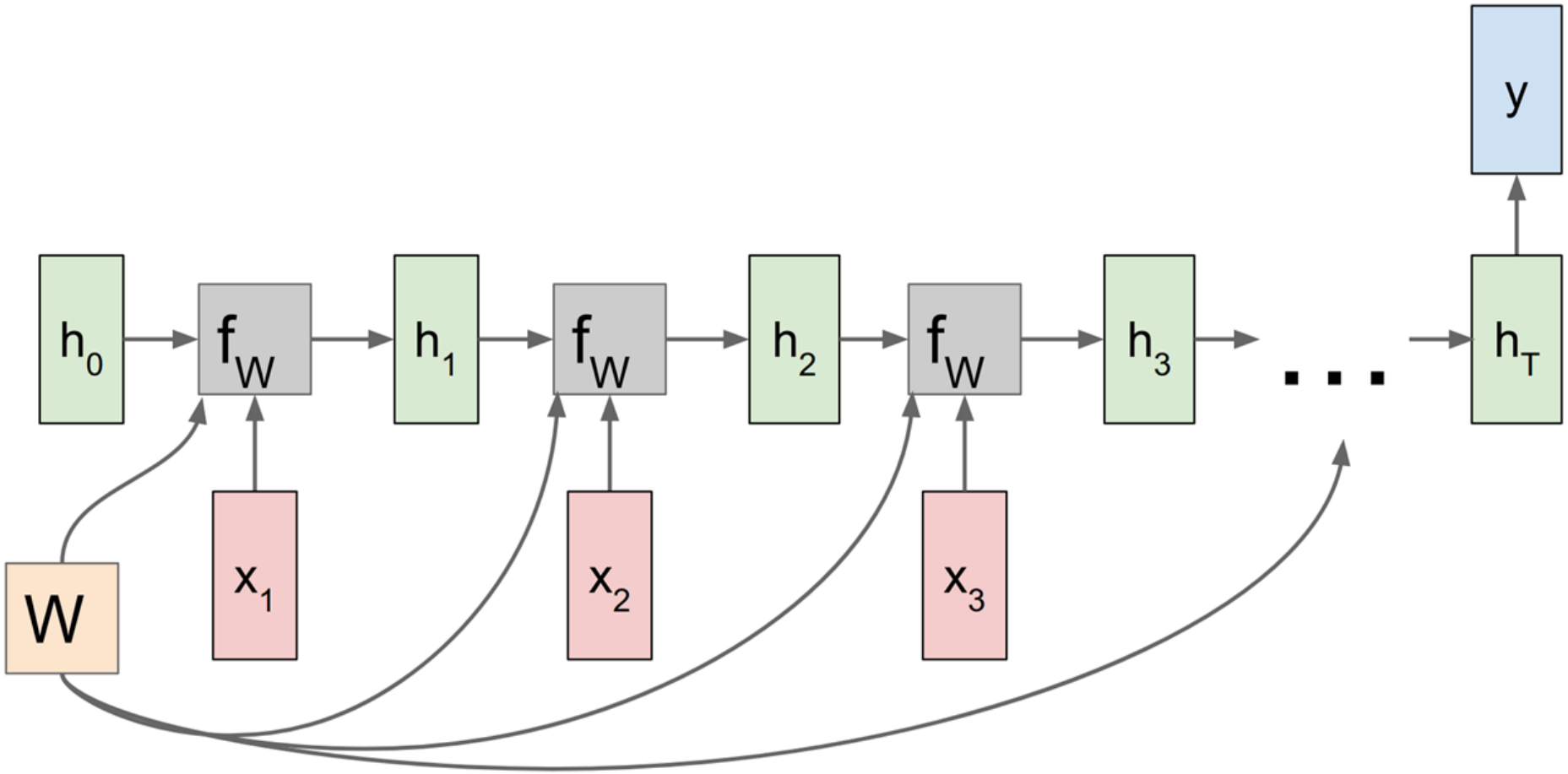


Loss Function for Many to Many



Many to One (Sentiment Analysis)

RNN: Computational Graph: Many to One



One to Many (Image Captioning)

RNN: Computational Graph: One to Many