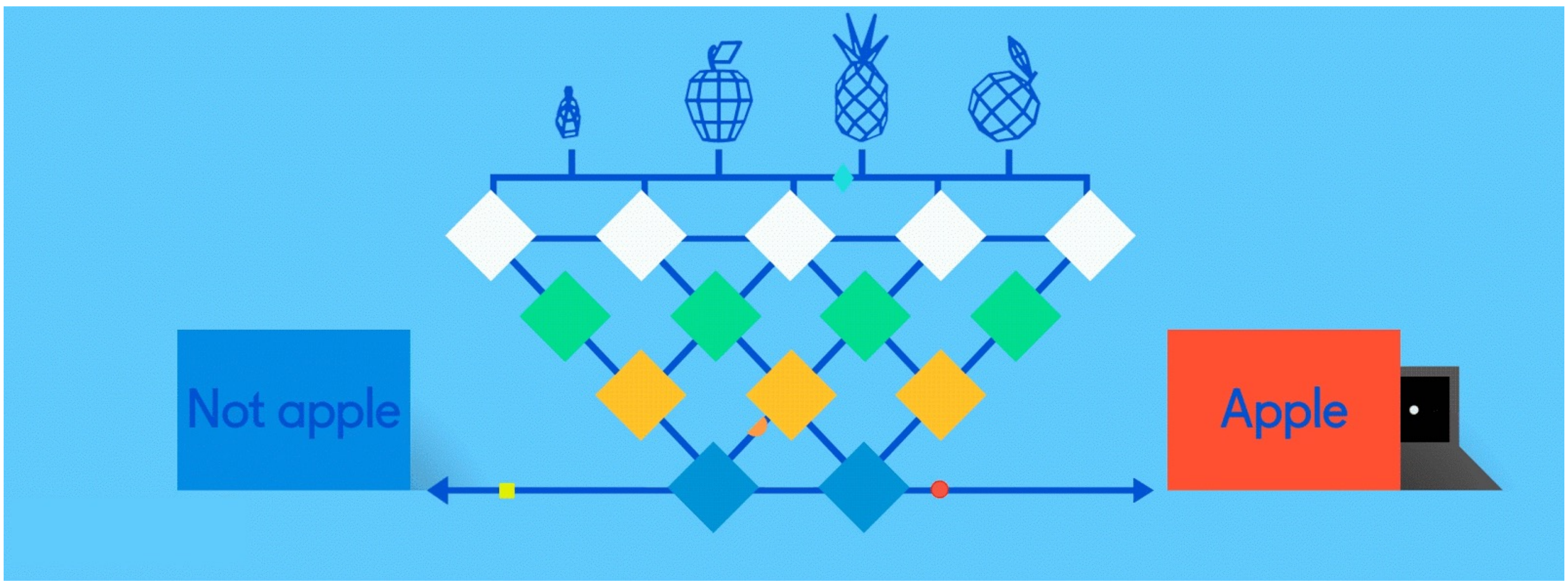
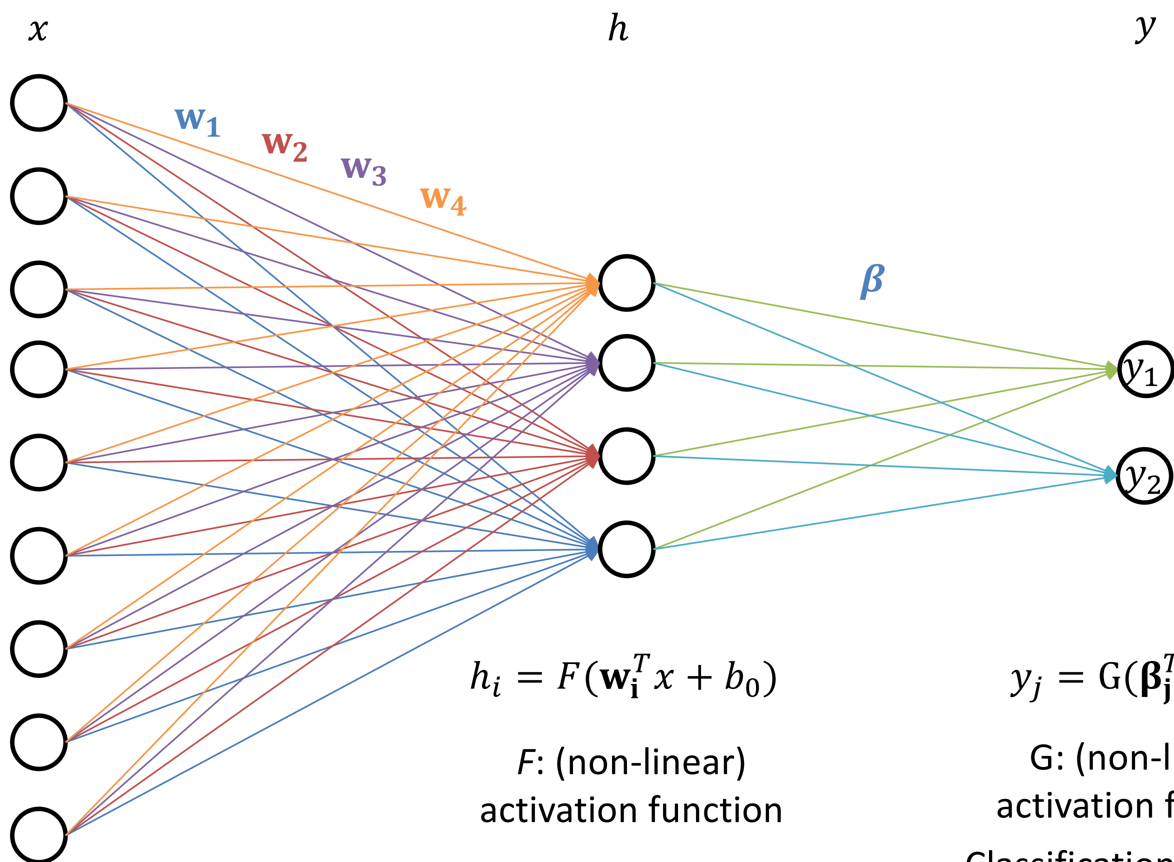
Neural Networks for Machine Learning



Multilayer Perceptron, a.k.a. Feed-Forward Neural Network



$$h_i = F(\mathbf{w_i}^T x + b_0)$$

F: (non-linear) activation function

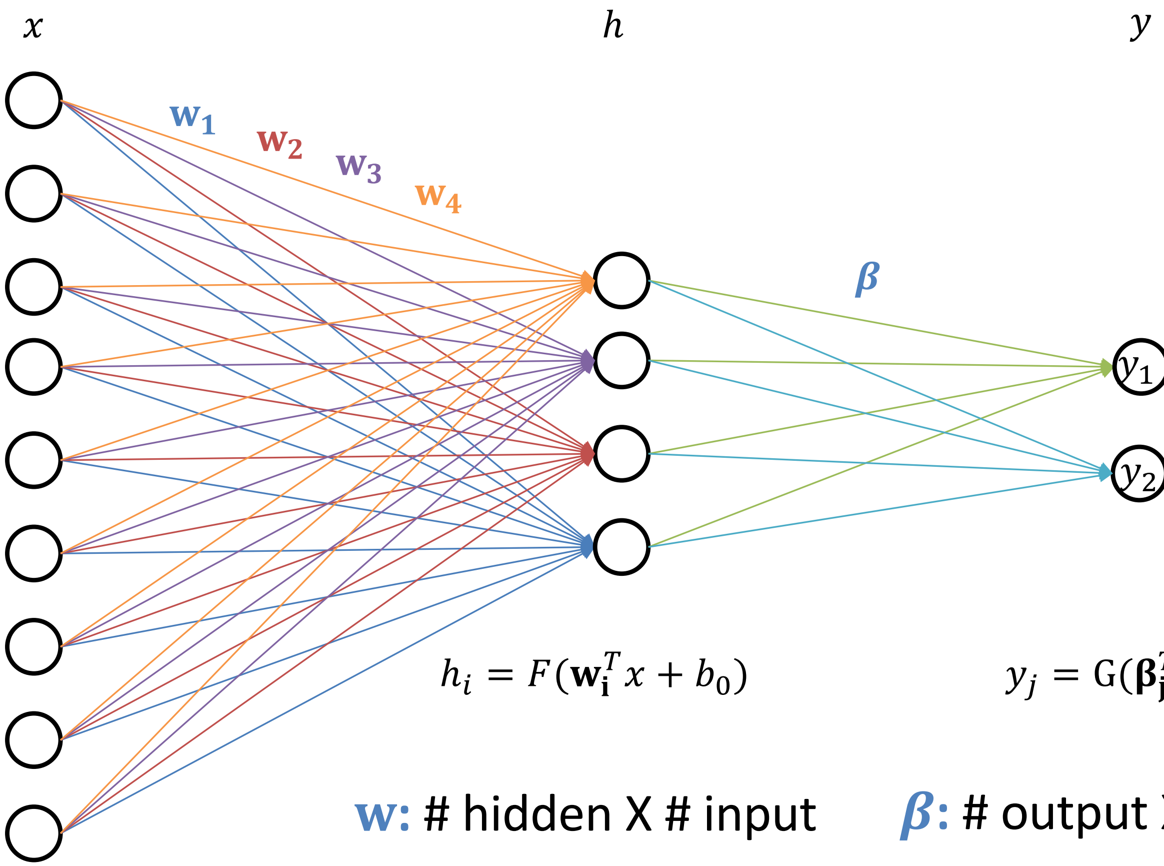
$$y_j = \mathrm{G}(\boldsymbol{\beta}_\mathbf{j}^T h + b_1)$$

G: (non-linear) activation function

Classification: softmax

Regression: identity

Feed-Forward Neural Network



$h_i = F(\mathbf{w_i}^T x + b_0)$

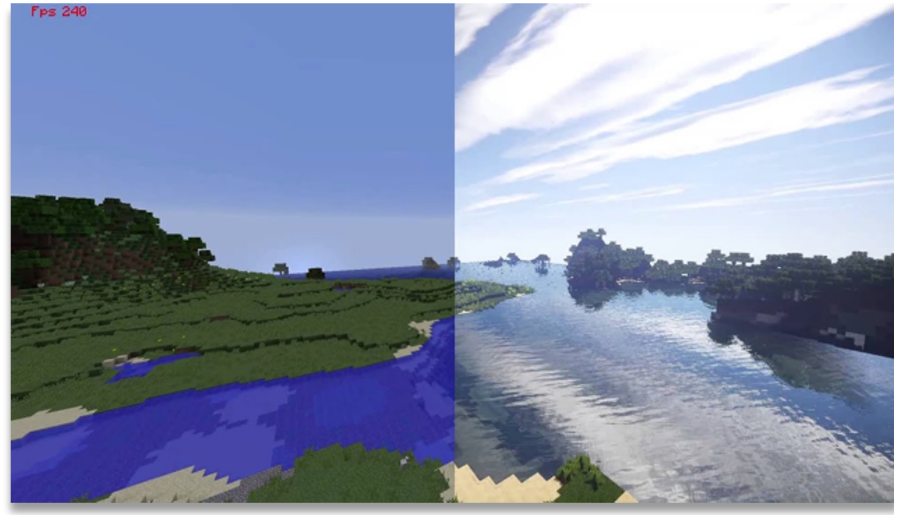
$y_j = \mathrm{G}(\boldsymbol{\beta}_\mathbf{j}^T h + b_1)$

w: # hidden X # input $\boldsymbol{\beta}$**:** # output X # hidden

But problems remained …

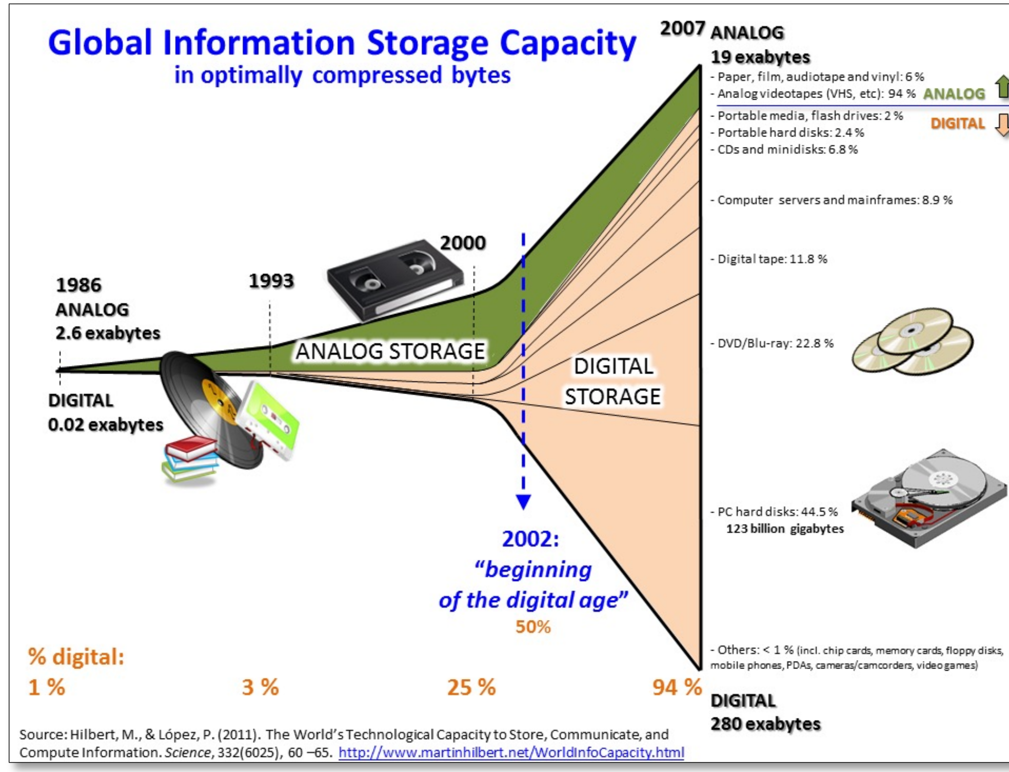
- It’s often the case that solving a problem just reveals a new one that needs solving
- For a large MLPs, backpropagation takes forever to converge!
- Two issues:
  - Not enough compute power to train the model
  - Not enough labeled data to train the neural net
- SVMs may be better, since they converge to global optimum in O(n^2)

GPUs solve compute power problem



- GPUs (Graphical Processing Units) became popular in the 1990s to handle computing needed for better computer graphics
- GPUs are SIMD (single instruction, multiple data) processors
- Cheap, fast, and easy to program
- GPUs can do matrix multiplication and other matrix computations very fast

Need lots of data!



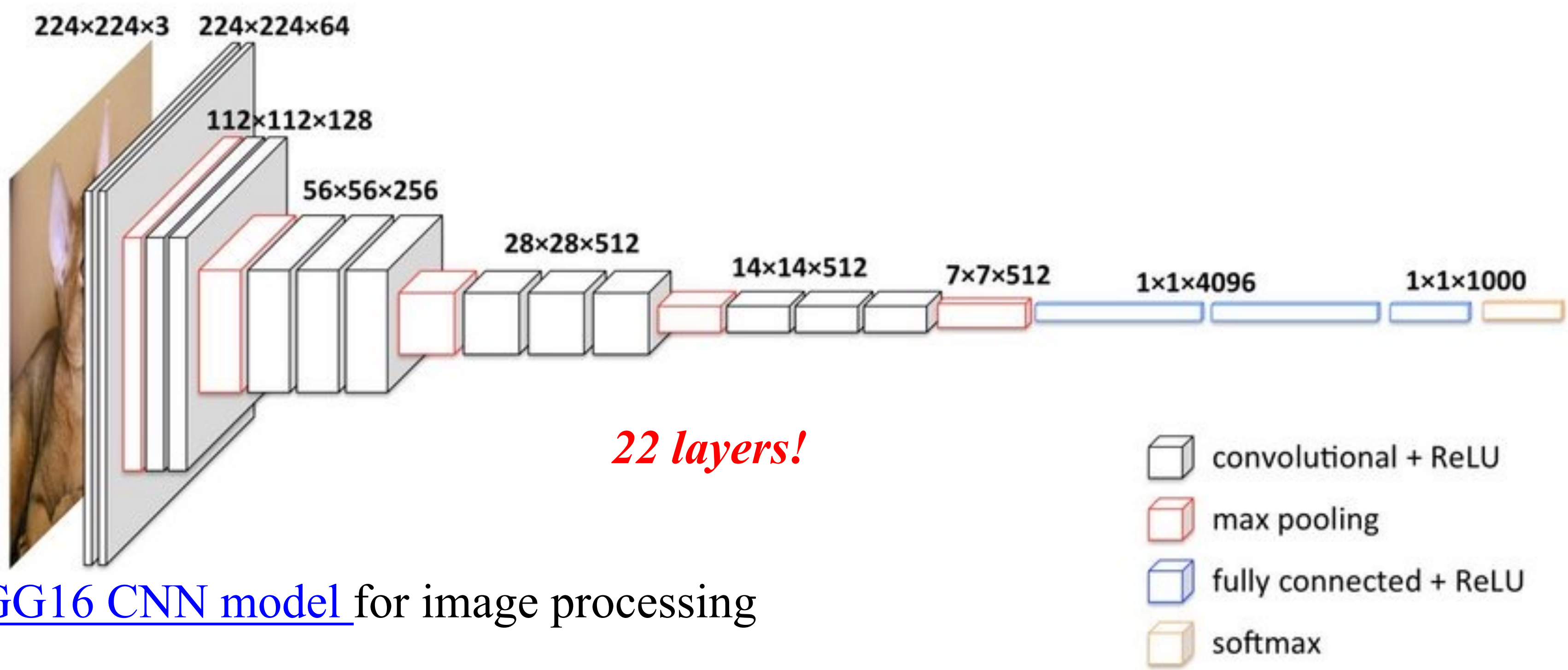
- 2000s introduced [big data]
- Cheaper storage
- Parallel processing (e.g., MapReduce, Hadoop, Spark, grid computing)
- Data sharing via the Web
  - Lots of images, many with captions
  - Lots of text, some with labels
- Crowdsourcing systems (e.g., [Mechanical Turk]) provided a way to get more human annotations

New problems are surfaced

- 2010s was a decade of domain applications
- These came with new problems, e.g.,
  - Images are too highly dimensioned!
  - Variable-length problems cause gradient problems
  - Training data is rarely labeled
  - Neural nets are uninterpretable
  - Training complex models required days or weeks
- This led to many new "deep learning" neural network models

Deep Learning

- Deep learning refers to models going beyond simple feed-forward multi-level perceptron
  - Though it was used in a ML context as early as 1986
- "deep" refers to the models having many layers (e.g., 10-20) that do different things



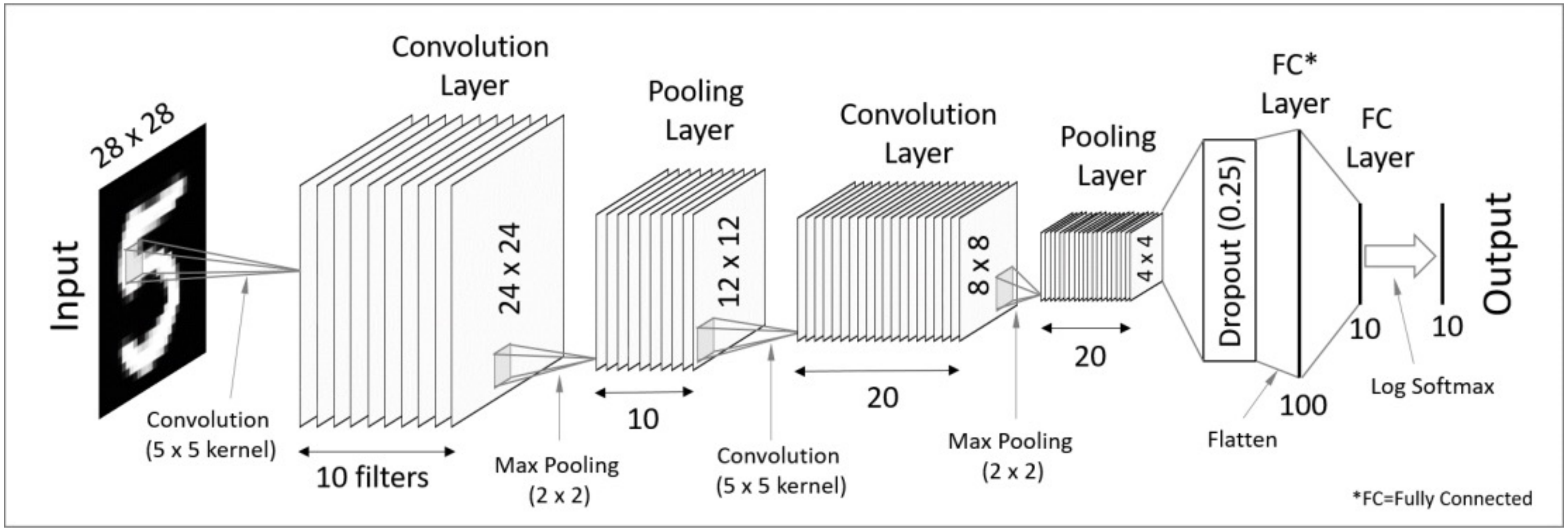
The VGG16 CNN model for image processing

Neural Network Architectures

Current focus on large networks with different "architectures" suited for different tasks

- Feedforward Neural Network
- CNN: **C**onvolutional **N**eural **N**etwork
- RNN: **R**ecurrent **N**eural **N**etwork
- LSTM: **L**ong Short **T**erm **M**emory
- GAN: **G**enerative **A**dversarial **N**etwork
- Transformers: generating output sequence from input sequence

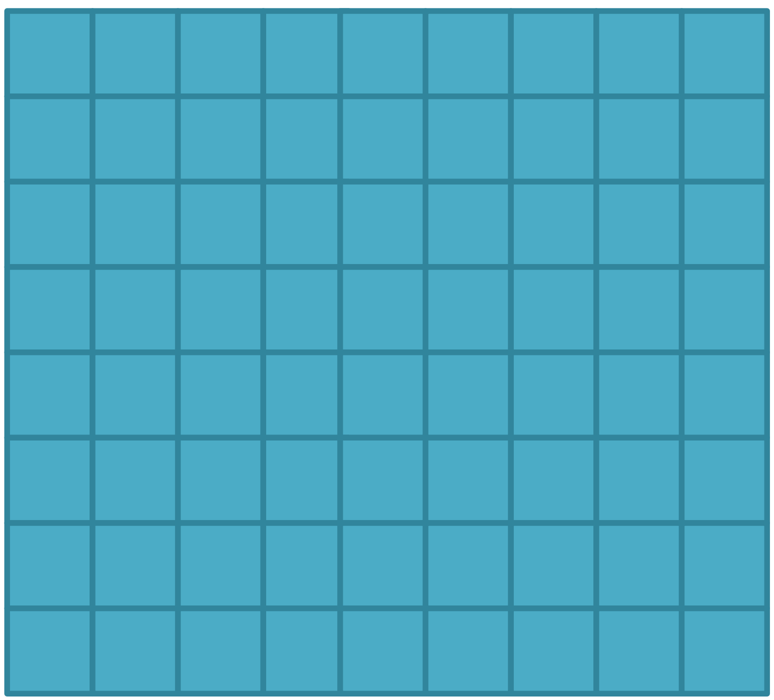
CNN: Convolutional Neural Network



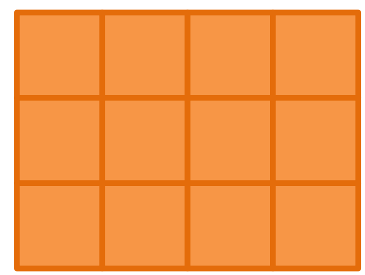
- Good for 2D image processing: classification, object recognition, automobile lane tracking, etc.
- Successive convolution layers learn higher-level features
- Classic demo: learn to recognize hand-written digits from MNIST data with 70K examples

In-Depth

2-D Convolution



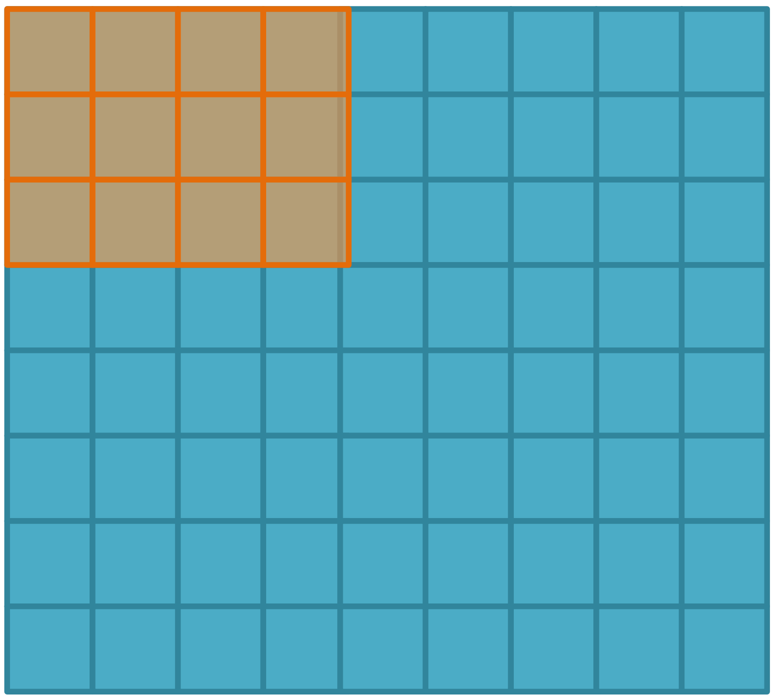
input
("image")



kernel

width: shape of the kernel
(often square)

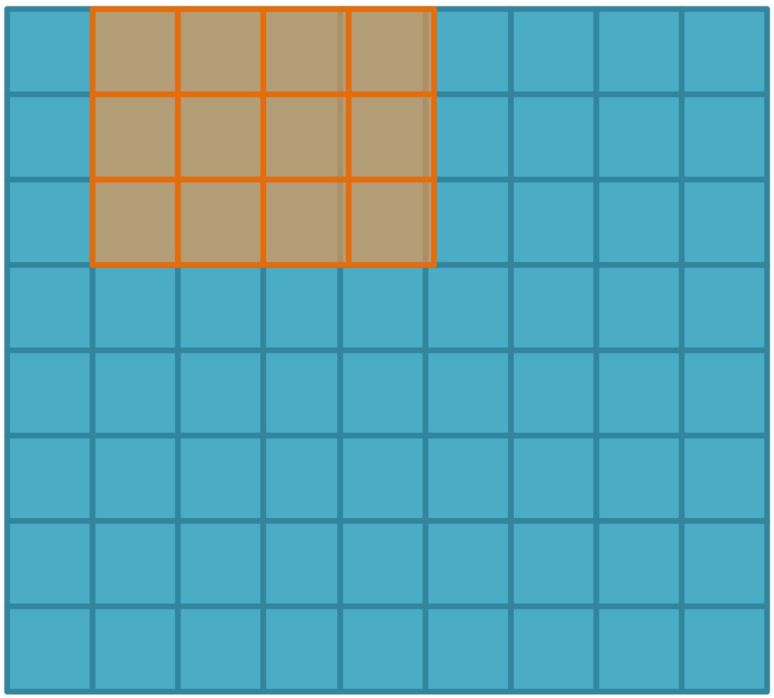
2-D Convolution



stride(s): how many spaces to move the kernel

width: shape of the kernel (often square)

2-D Convolution



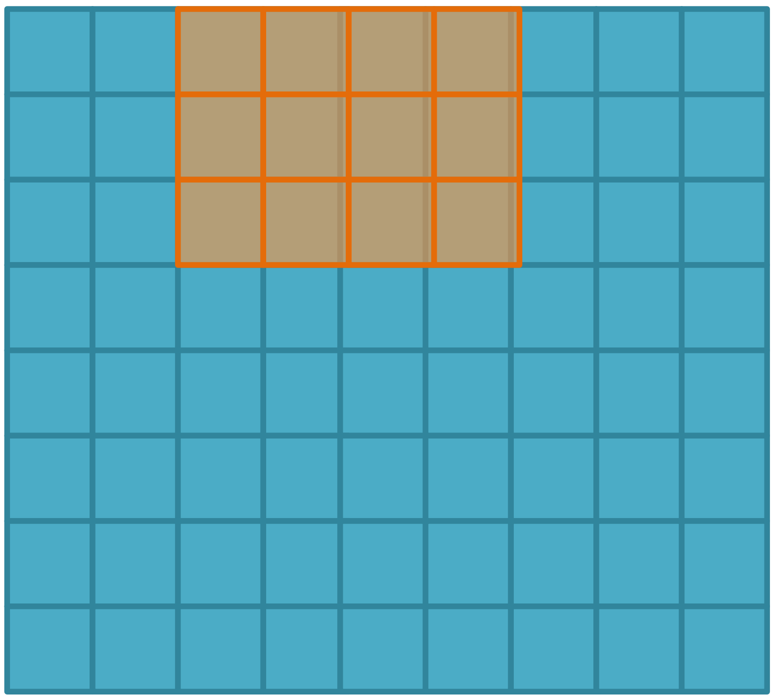
input
(“image”)

stride(s): how many spaces to move the kernel

stride=1

width: shape of the kernel (often square)

2-D Convolution



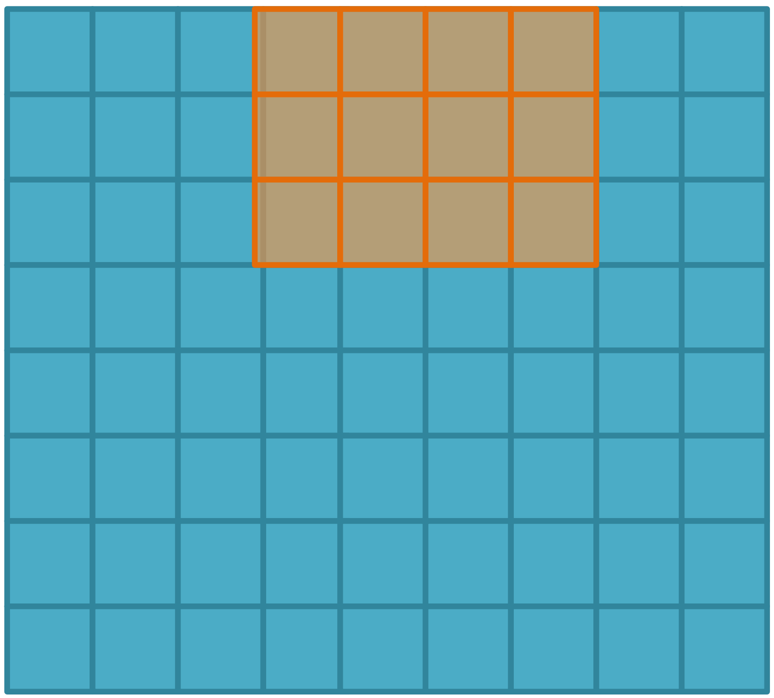
input
("image")

stride(s): how many spaces to move the kernel

stride=1

width: shape of the kernel (often square)

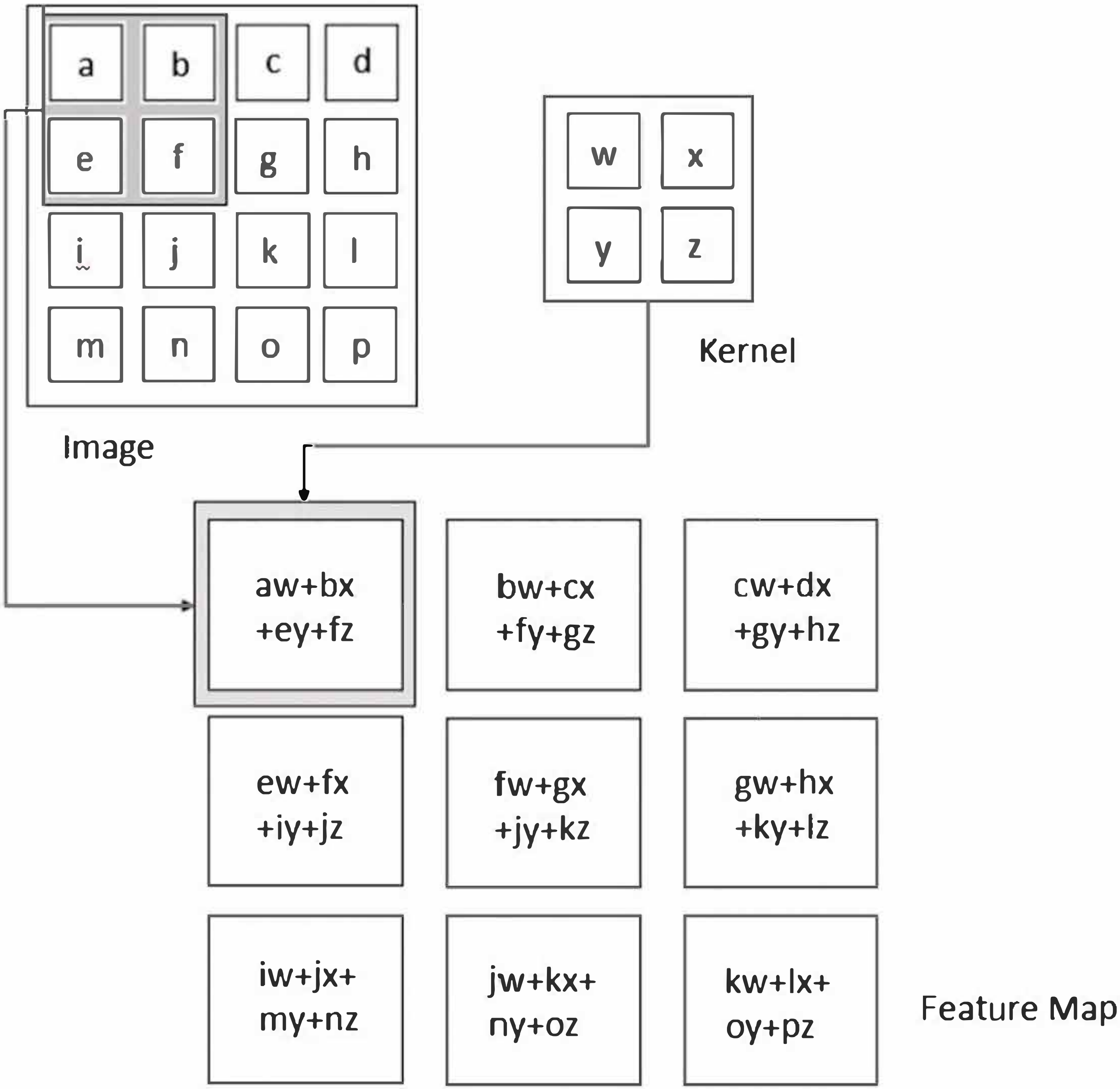
2-D Convolution



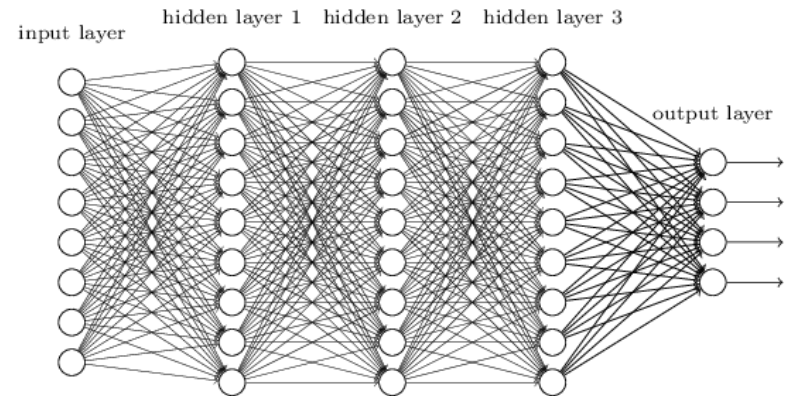
stride(s): how many spaces to move the kernel

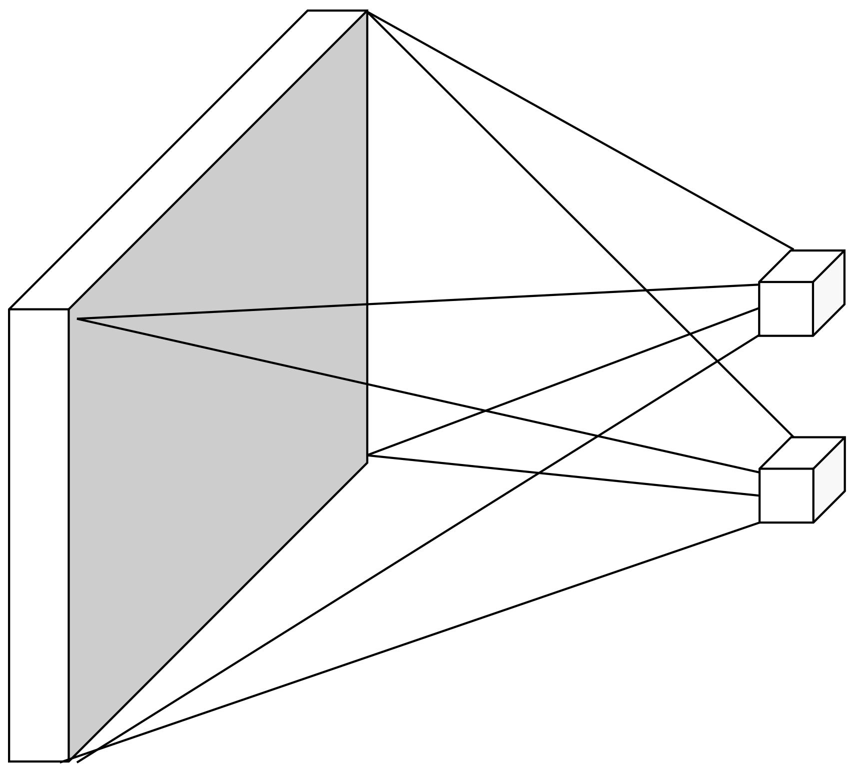
stride=1

width: shape of the kernel (often square)


a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
Image
w
x
y
z
Kernel
aw+bx +ey+fz
bw+cx +fy+gz
cw+dx +gy+hz
ew+fx +iy+jz
fw+gx +jy+kz
gw+hx +ky+lz
iw+jx+ my+nz
jw+kx+ ny+oz
kw+lx+ oy+pz
Feature Map

From fully connected to convolutional networks



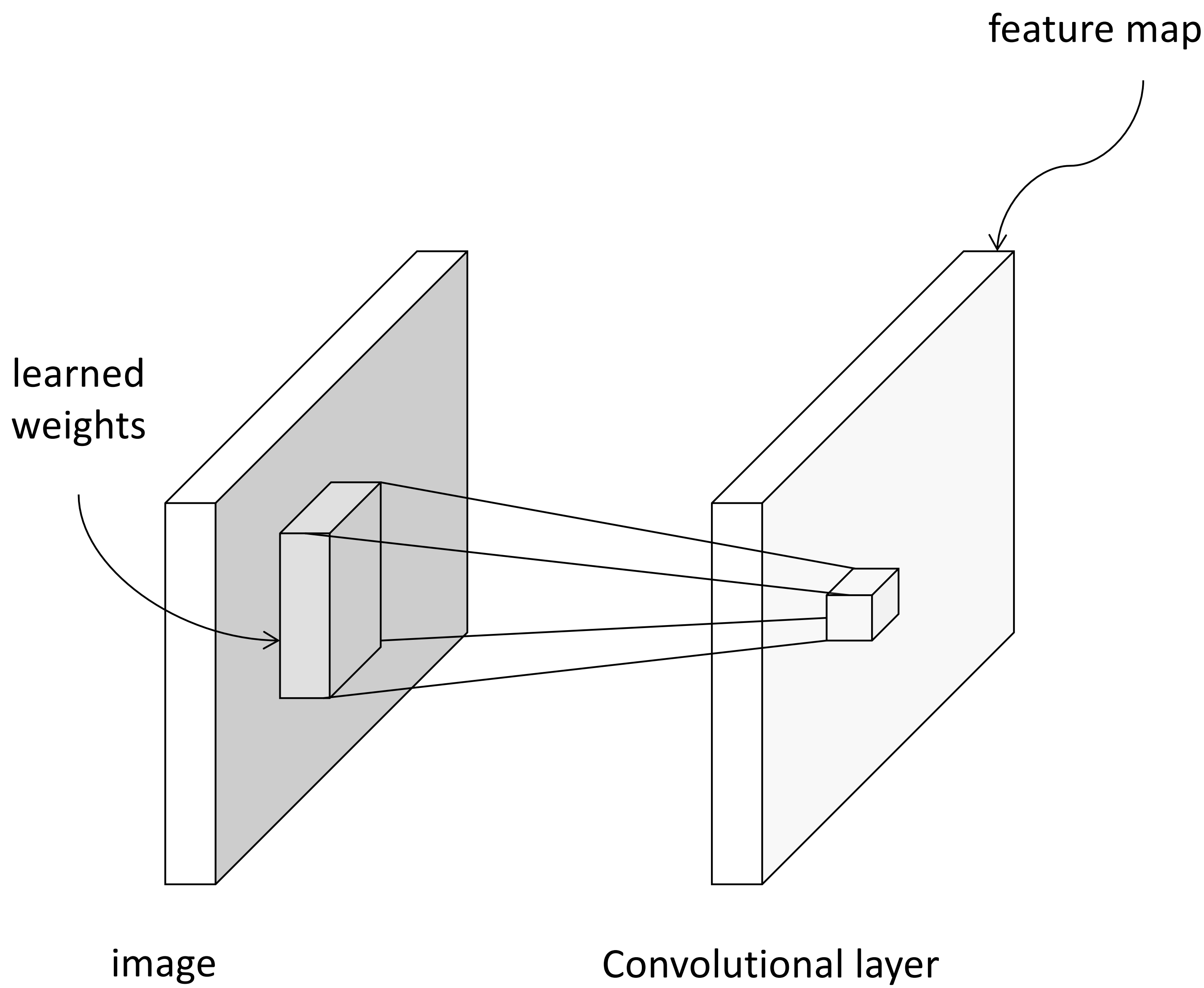


image

Fully connected layer

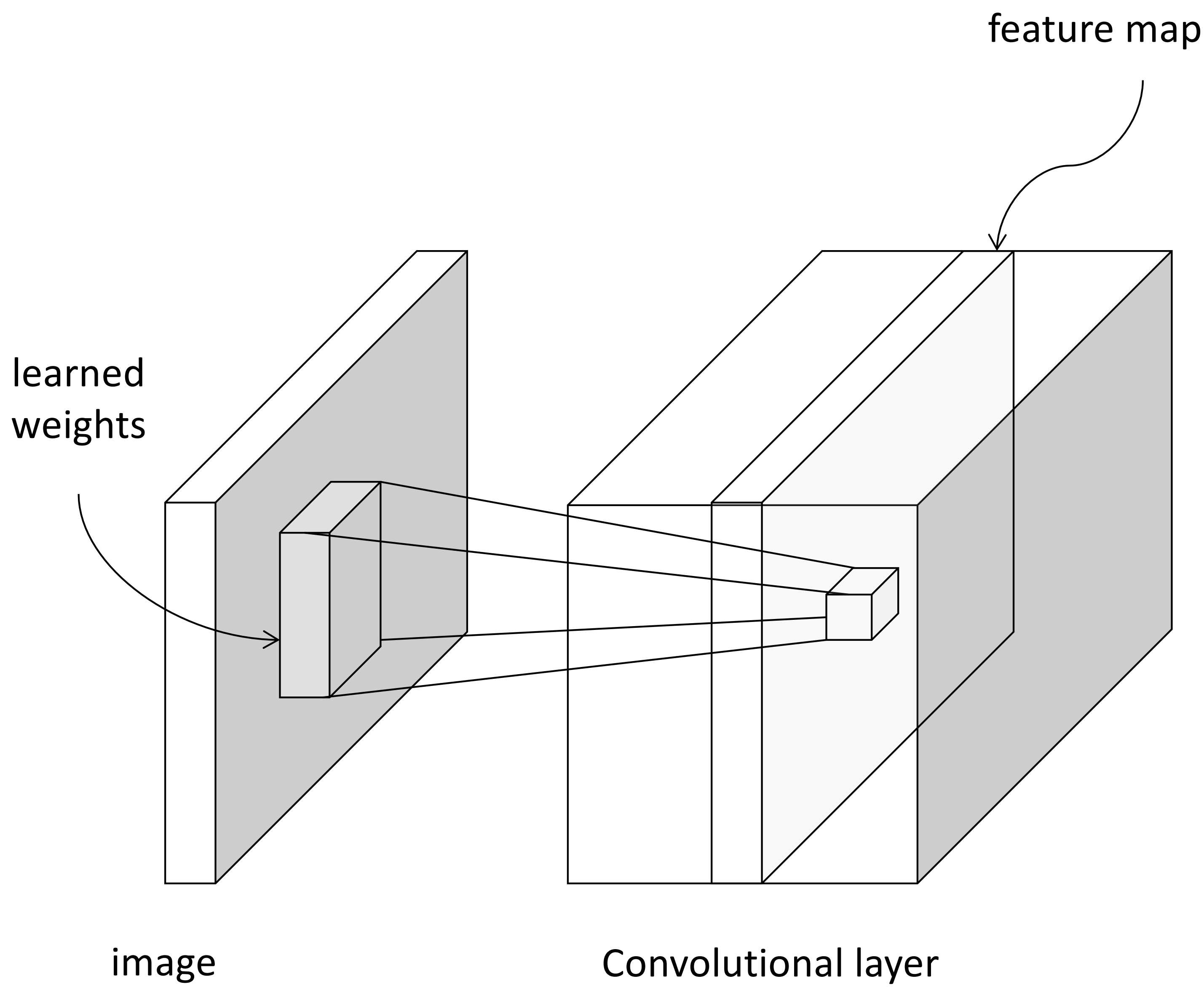
Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks



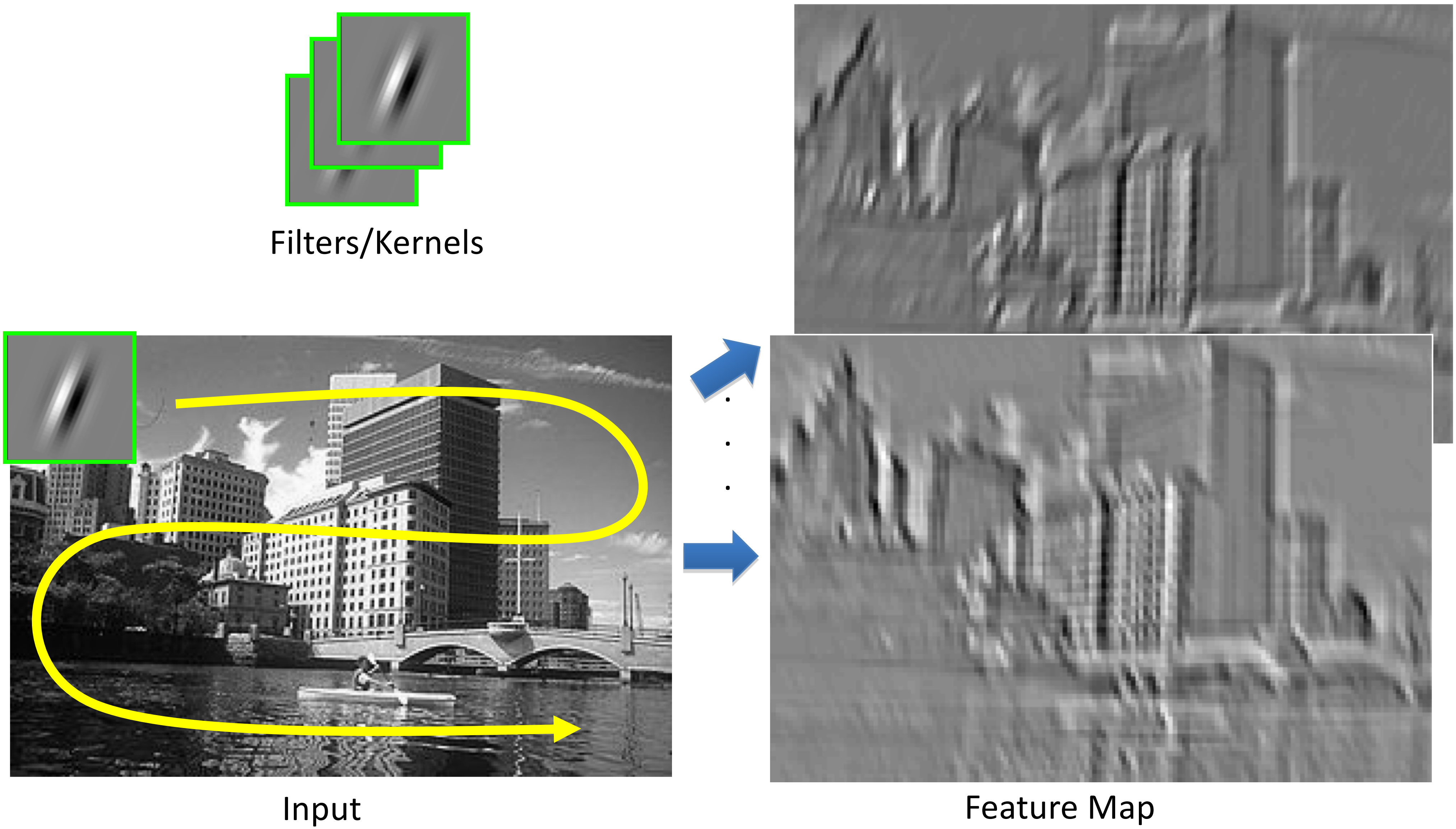
Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks



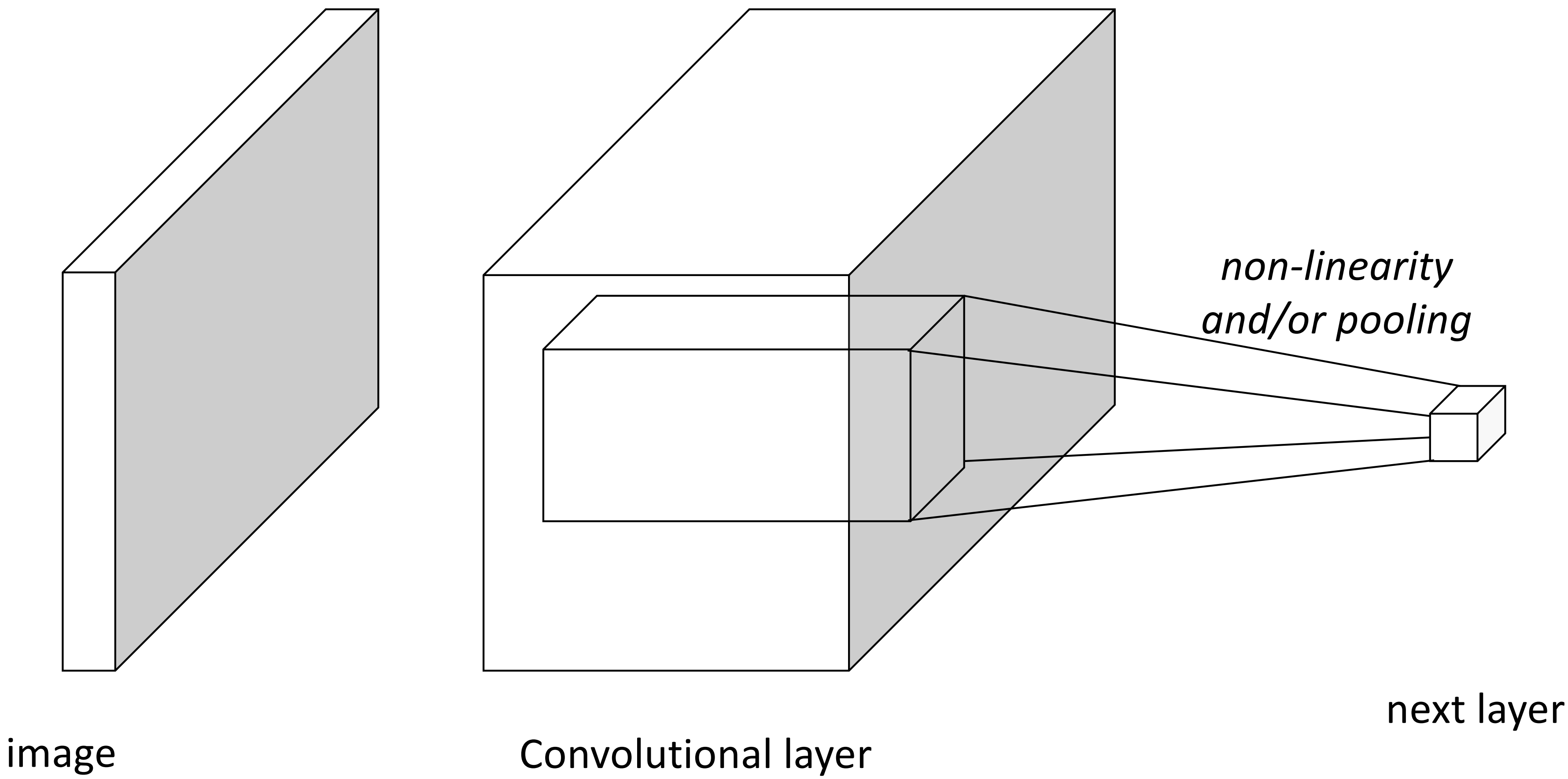
Slide credit: Svetlana Lazebnik

Convolution as feature extraction



Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks

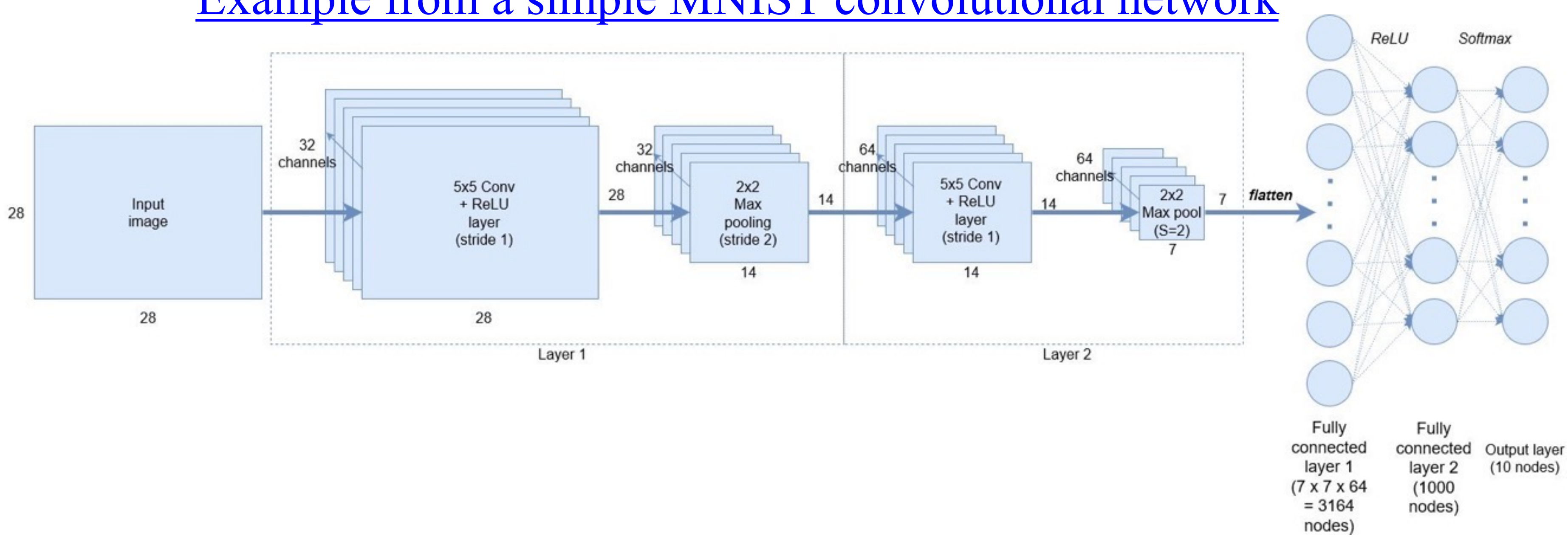


Slide adapted: Svetlana Lazebnik

Keras: API works with TensorFlow

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

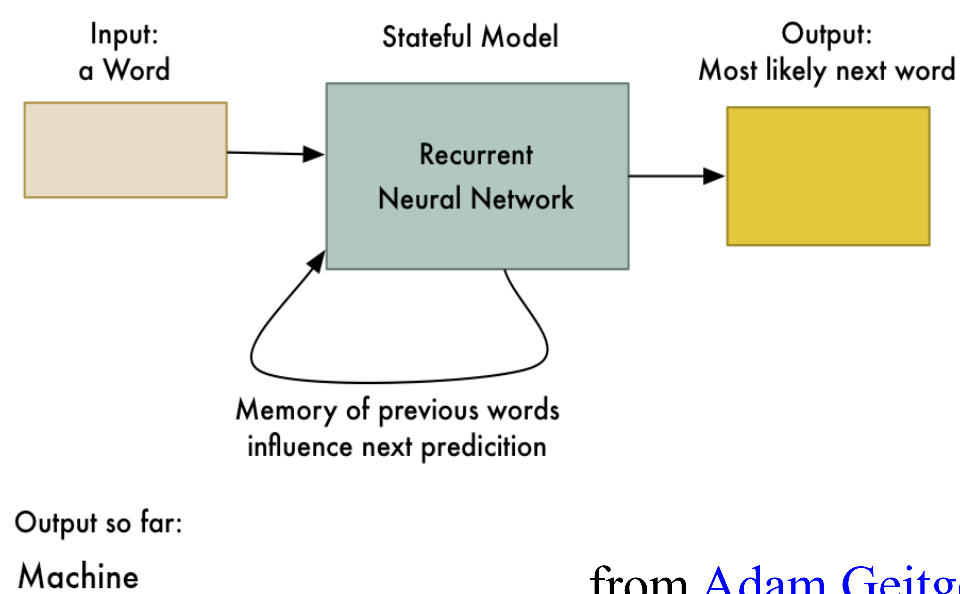
Example from a simple MNIST convolutional network



RNN: Recurrent Neural Networks

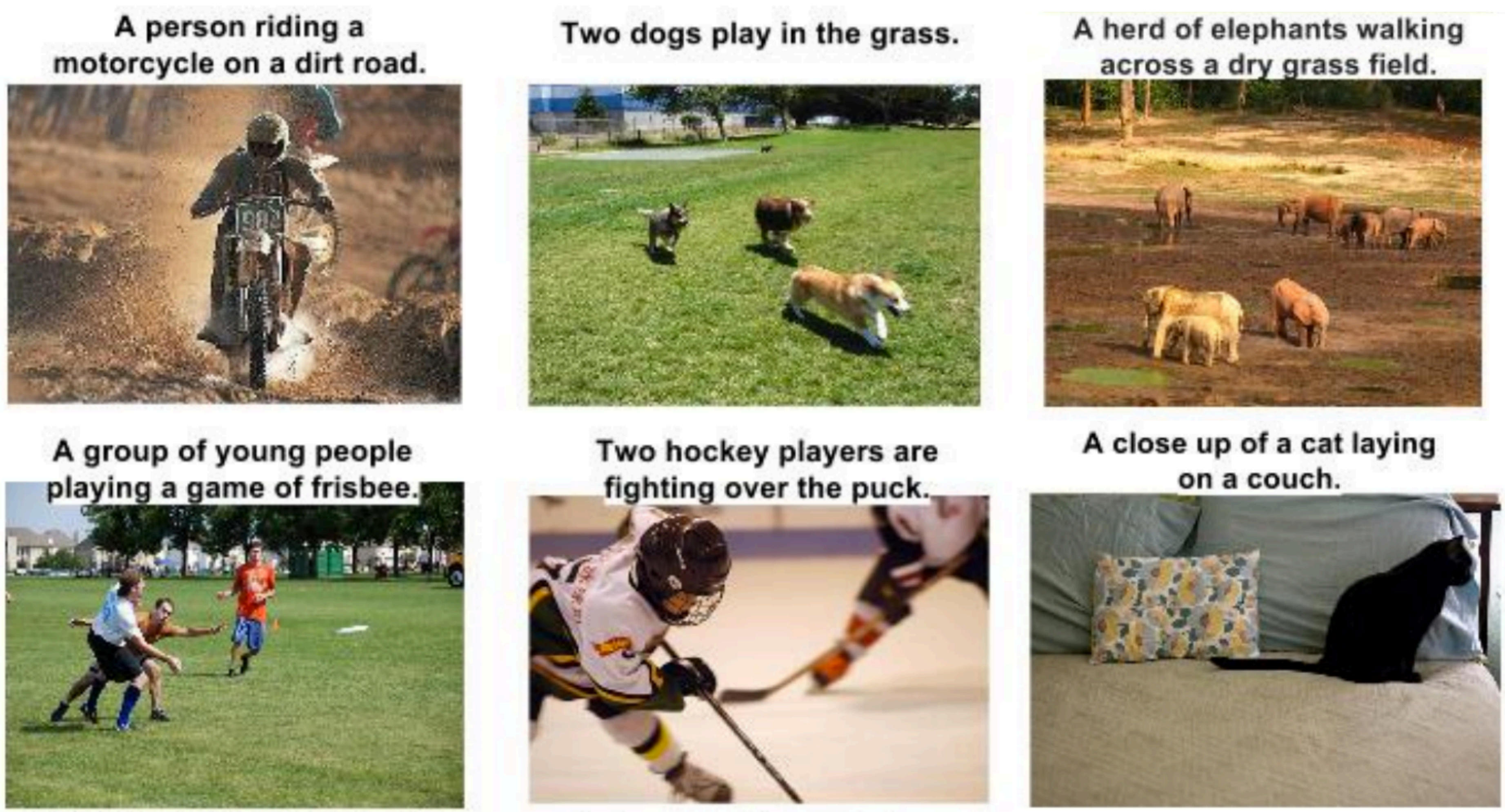
- Good for learning over sequences of data, e.g., a sentence of words
- LSTM: (Long Short Term Memory) a popular architecture that remembers and uses previous N inputs
- BI-LSTM: knows previous and upcoming inputs
- Attention: recent idea that learns long-range dependencies between inputs

Simulation in Board



from Adam Geitgey

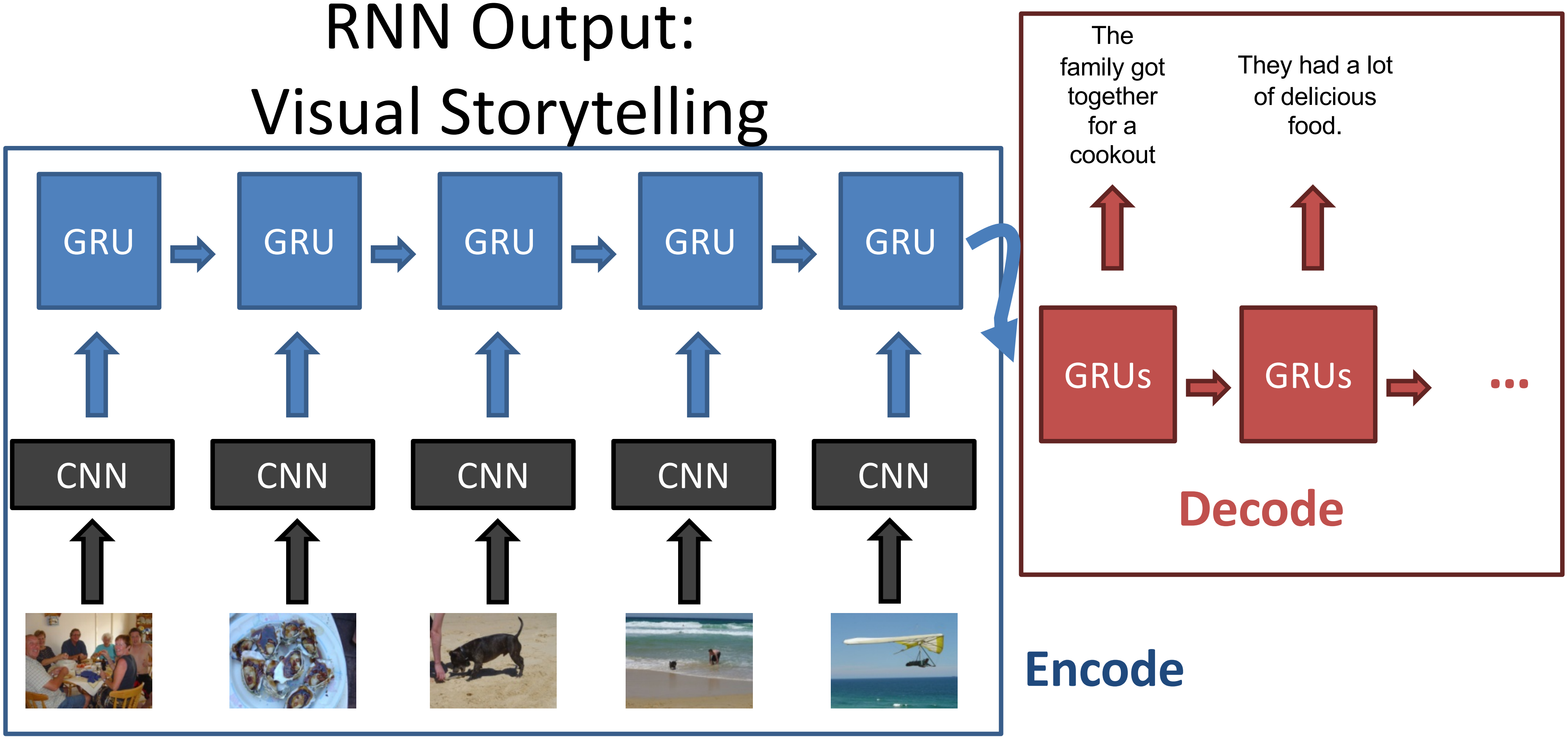
RNN Outputs: Image Captions



[Show and Tell: A Neural Image Caption Generator, CVPR 15]

Slide credit: Arun Mallya

RNN Output: Visual Storytelling



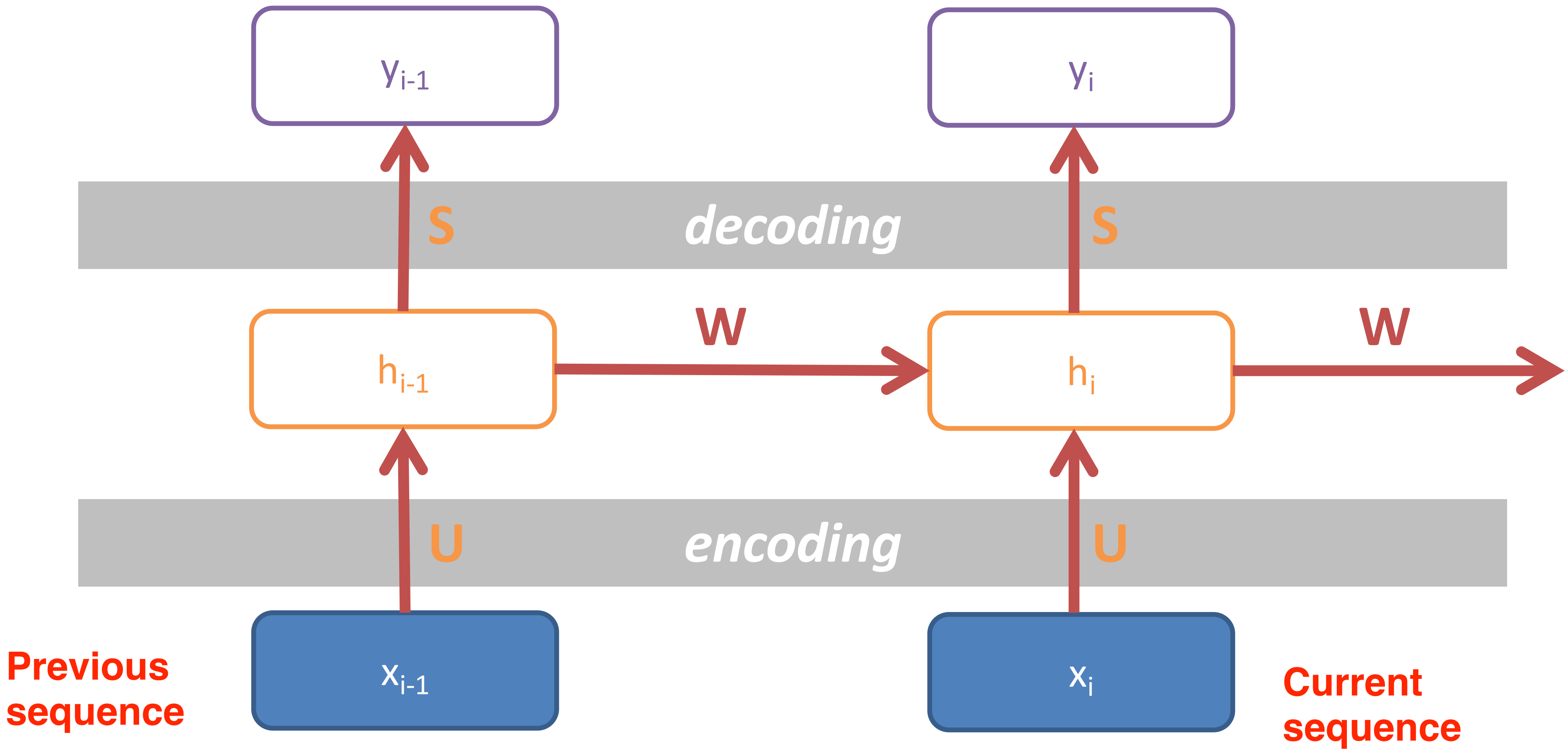
Human Reference

The family has gathered around the dinner table to share a meal together. They all pitched in to help cook the seafood to perfection. Afterwards they took the family dog to the beach to get some exercise. The waves were cool and refreshing! The dog had so much fun in the water. One family member decided to get a better view of the waves!

Huang et al. (2016)

The family got together for a cookout. They had a lot of delicious food. The dog was happy to be there. They had a great time on the beach. They even had a swim in the water.

A *Simple* Recurrent Neural Network Cell



$$h_i = \tanh(W h_{i-1} + U x_i)$$

$$y_i = \mathrm{softmax}(S h_i)$$

Weights are shared over time

unrolling/unfolding: copy the RNN cell across time (inputs)

GAN: Generative Adversarial Network

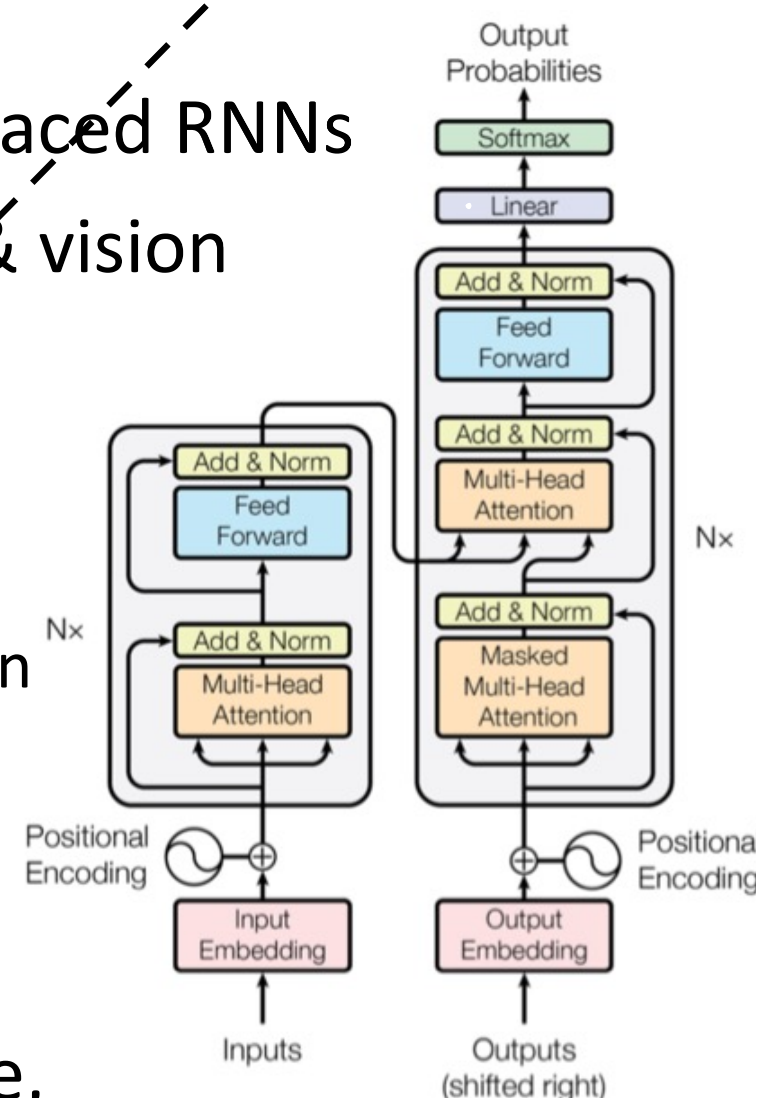
Not Covered

- System of **two neural networks** competing against each other in a zero-sum game framework
- Provides a kind of **unsupervised learning** that improves the network
- Introduced by Ian Goodfellow et al. in 2014
- Can learn to draw samples from a model that is similar to data that we give them

Not Covered

Transformer

- Introduced in 2017 & has largely replaced RNNs
- Used primarily for natural language & vision processing tasks
- NLP applications "transform" an input text into an output text
  - E.g., translation, summarization, question answering
- Uses encoder-decoder architecture with attention
- Popular pre-trainted models available, e.g. BERT and GPT



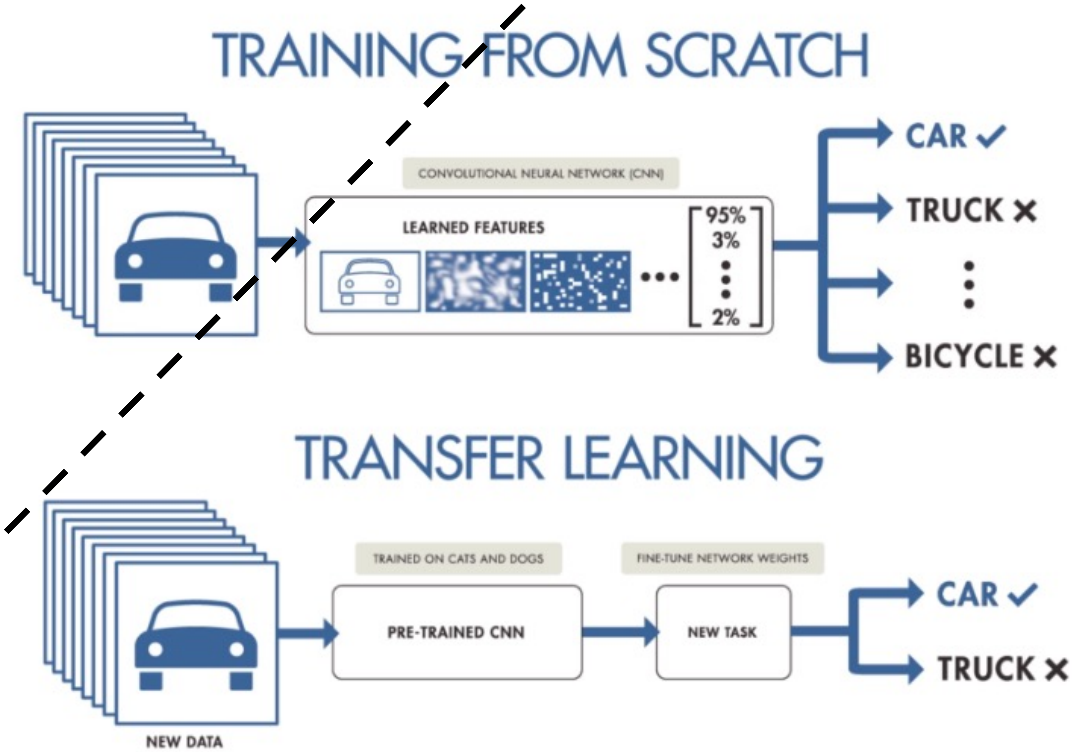
Not Covered

NNs Good at Transfer Learning

- Neural networks effective for [transfer learning]

  Using parts of a model trained on a task as an initial model to train on a different task

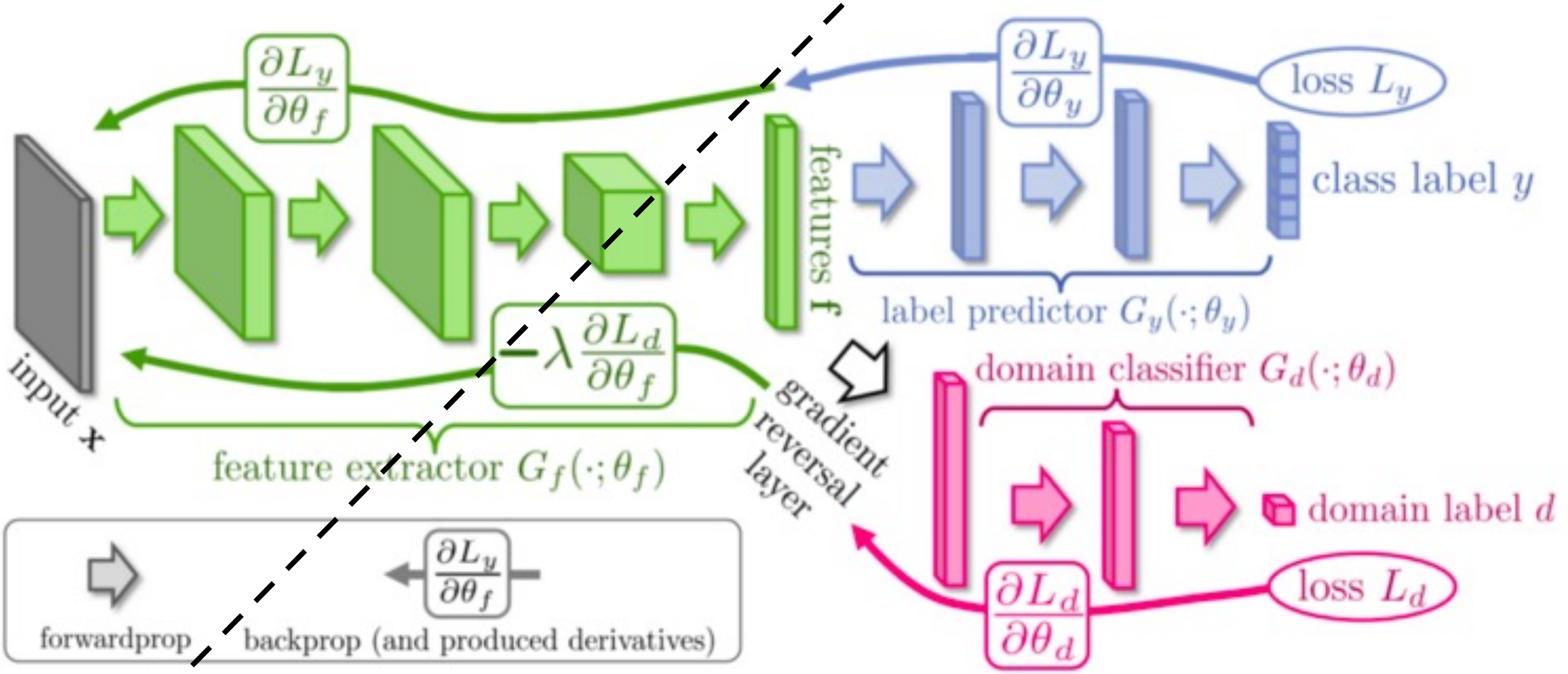
- Particularly effective for image recognition and language understanding tasks



Not Covered

Good at Transfer Learning

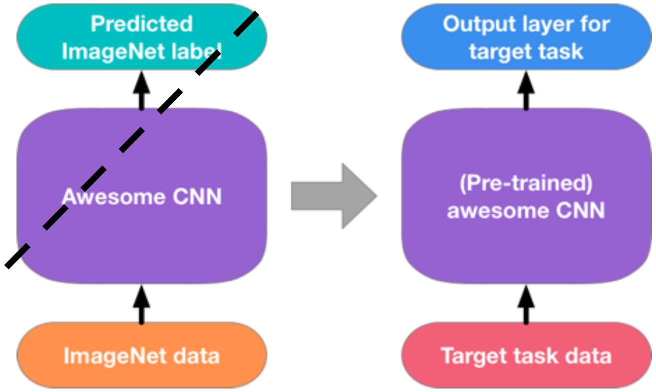
- For images, the initial stages of a model learn high-level visual features (lines, edges) from pixels
- Final stages predict task-specific labels



source:http://ruder.io/transfer-learning/

Not Covered

Fine Tuning a NN Model



- Special kind of transfer learning
  - Start with a pre-trained model
  - Replace last output layer(s) with a new one(s)
  - One option: Fix all but last layer by marking as trainable:false
- Retraining on new task and data very fast
  - Only the weights for the last layer(s) are adjusted
- Example
  - Start: NN to classify animal pix with 100s of categories
  - Finetune on new task: classify pix of 10 common pets

Conclusions

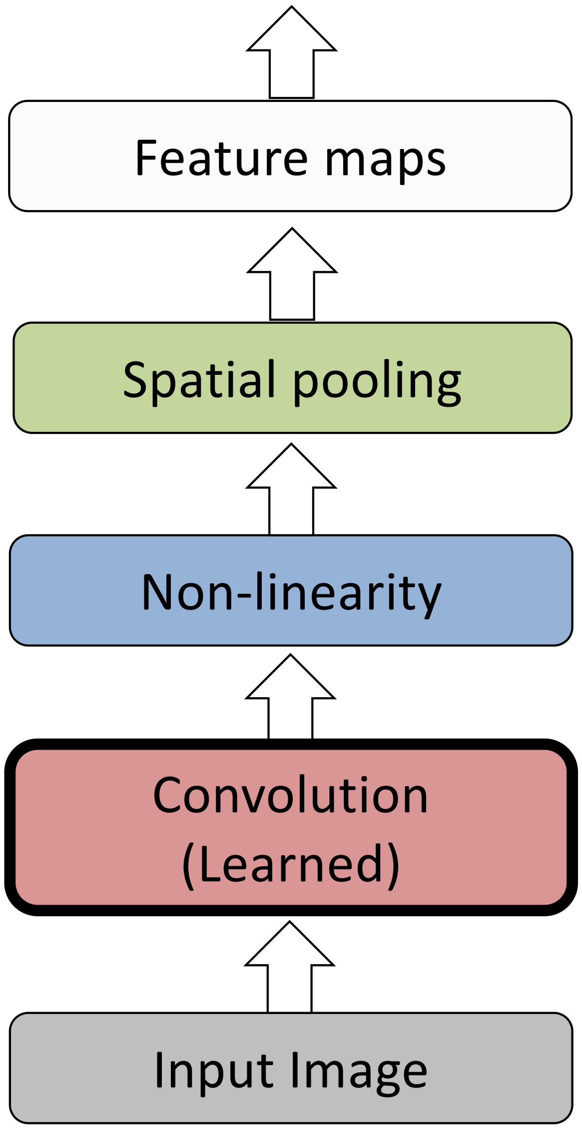
- Quick intro to neural networks & deep learning
- Learn more by
  - Try scikit-learn's neural network models
  - Explore TensorFlow with Keras / PyTorch
  - Work through examples
- and then try your own project idea

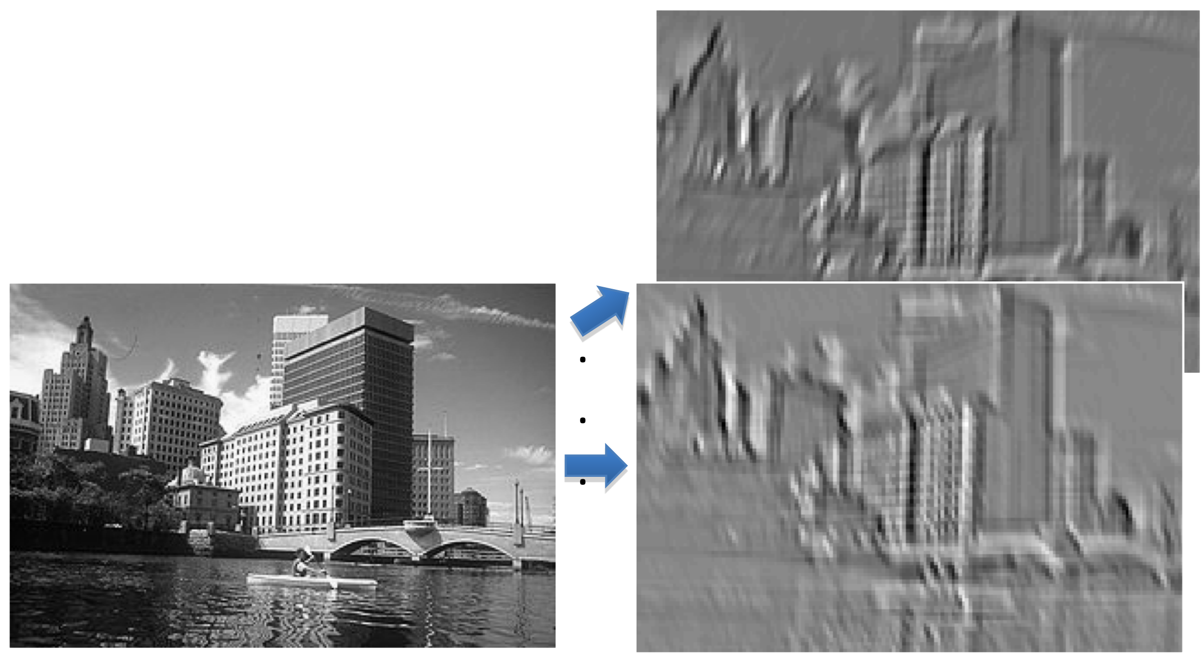
Student Course Evaluations

- Check for email from **StudentCourseEvaluations@umbc.edu**
- **Announcement:** *"The Student Evaluation of Educational Quality (SEEQ) is a standardized course evaluation instrument used to provide measures of an instructor's teaching effectiveness. The results of this questionnaire will be used by promotion and tenure committees as part of the instructor's evaluation. The Direct Instructor Feedback Forms (DIFFs) are designed to provide feedback to instructors and are not intended for use by promotion and tenure committees. The responses to the SEEQ and the DIFFs will be kept anonymous and will not be distributed until UMBC final grades are in"*

Extra Slides

Key operations in a CNN



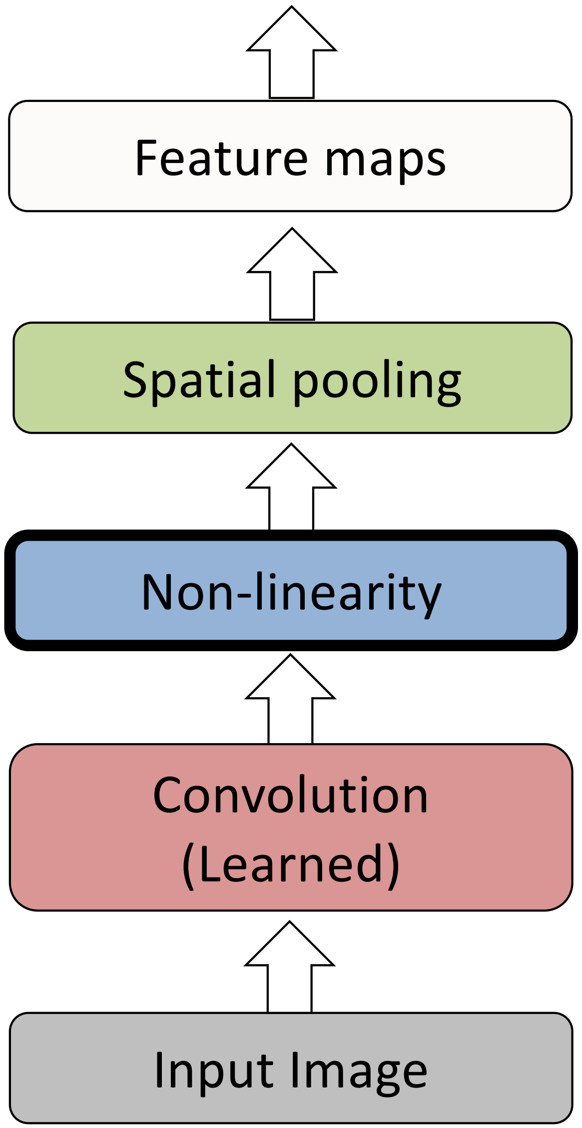


Input

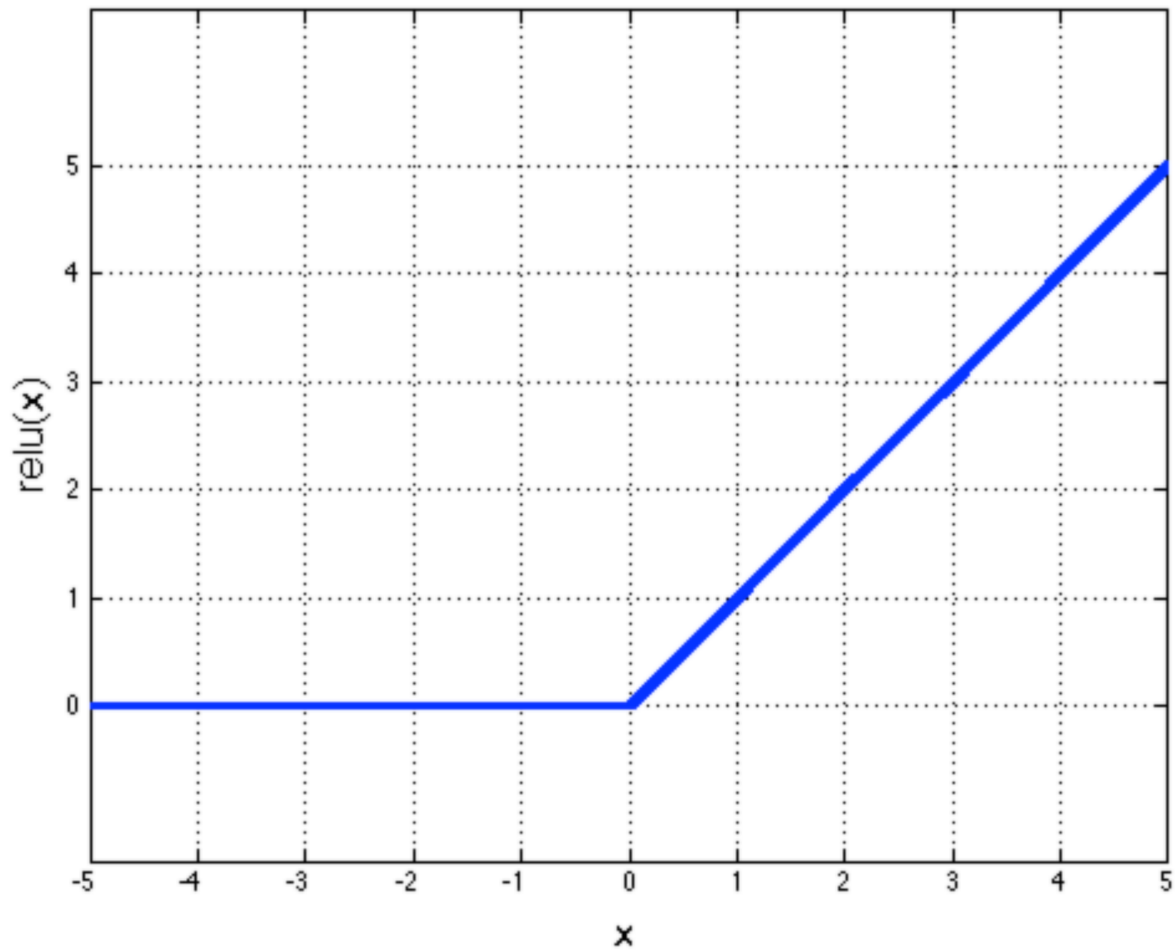
Feature Map

Slide credit: Svetlana Lazebnik, R. Fergus, Y. LeCun

Key operations

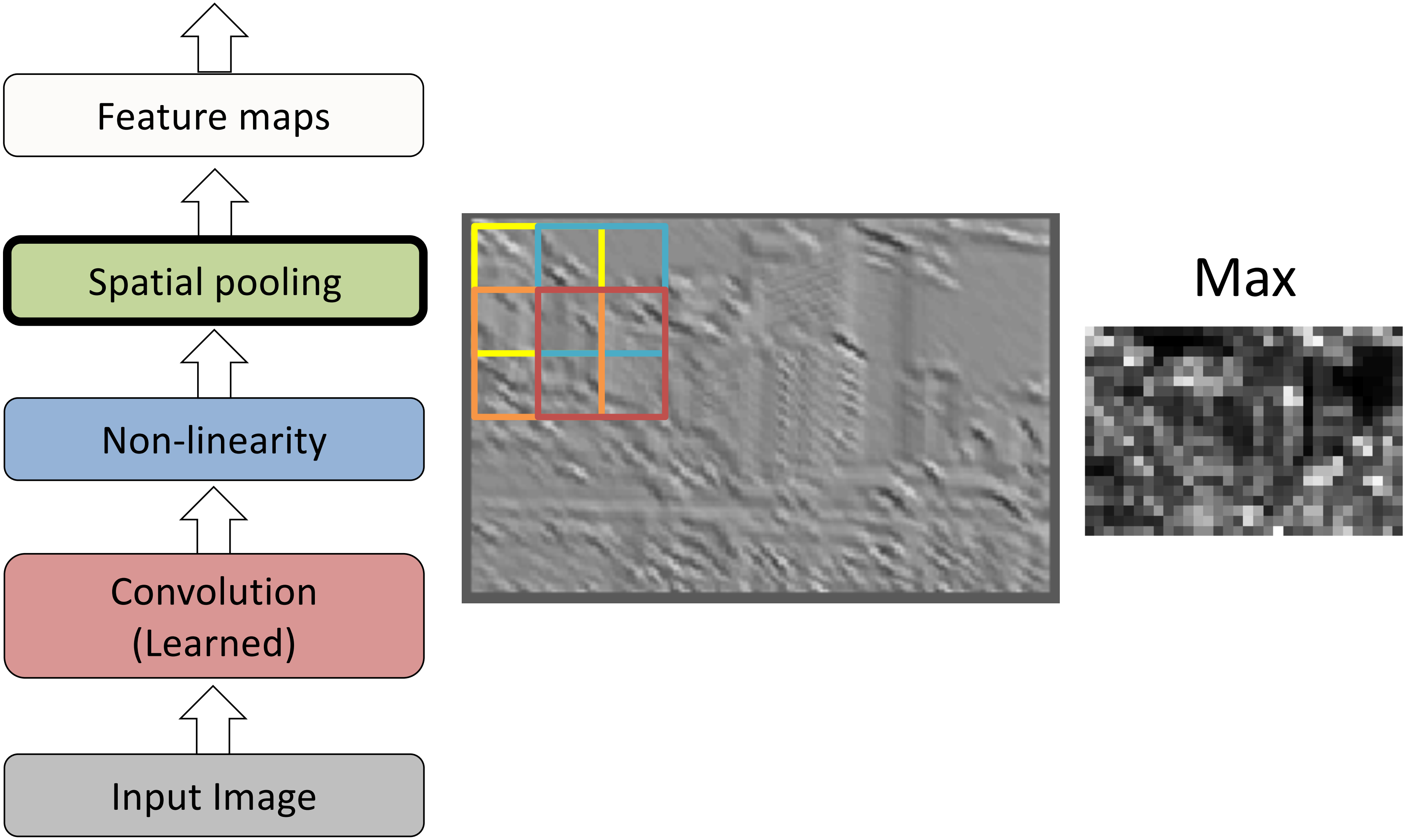


Example: Rectified Linear Unit (ReLU)



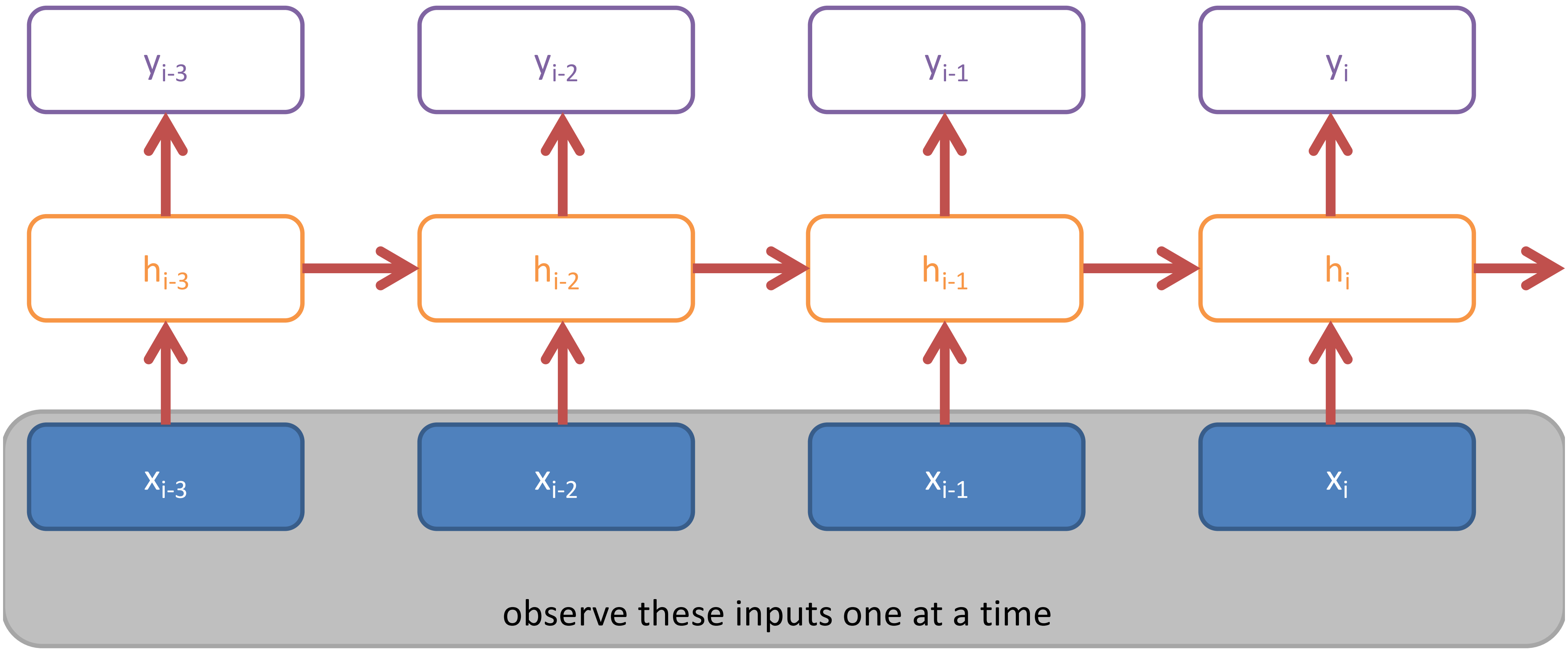
Slide credit: Svetlana Lazebnik, R. Fergus, Y. LeCun

Key operations

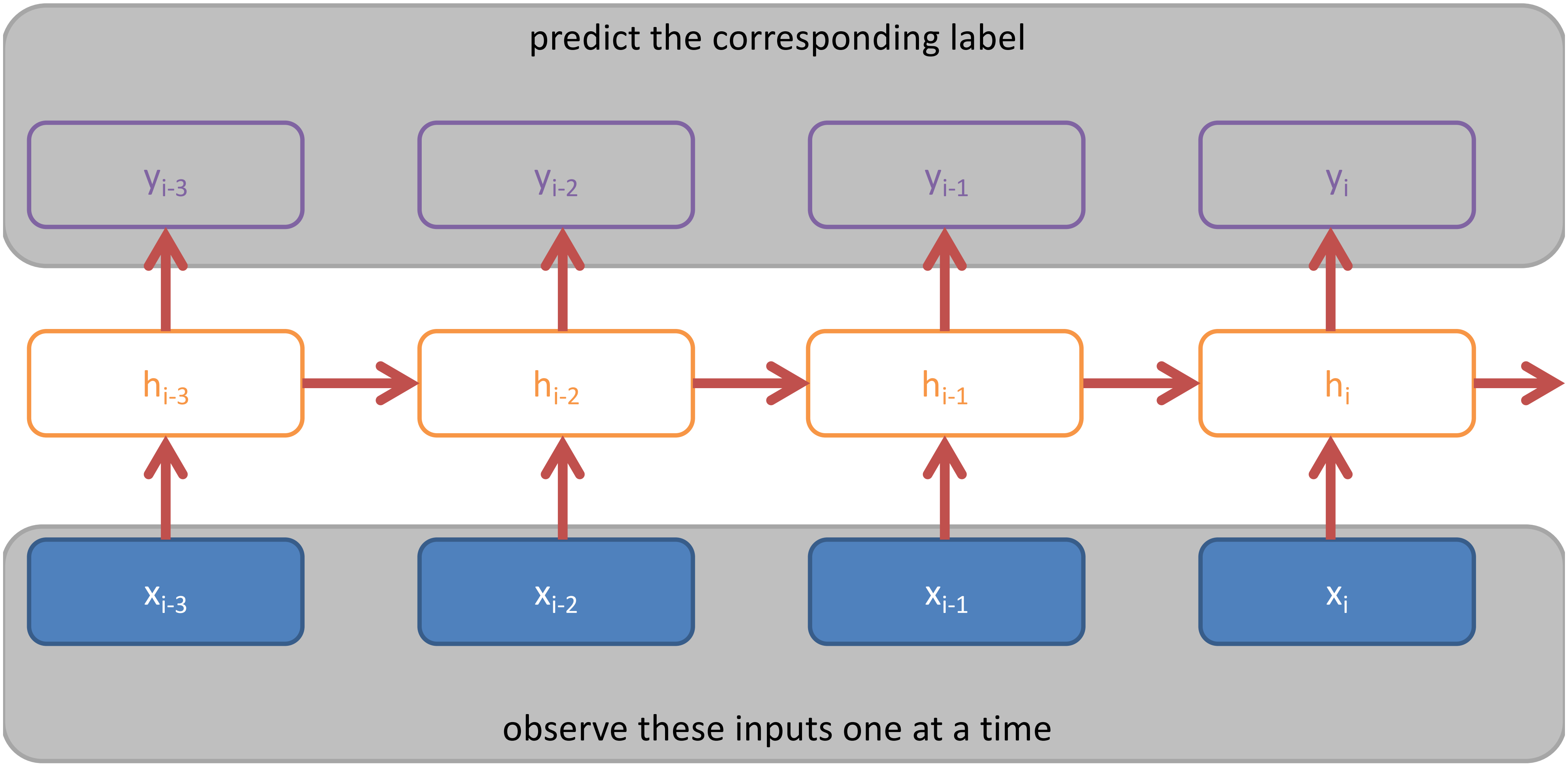


Slide credit: Svetlana Lazebnik, R. Fergus, Y. LeCun

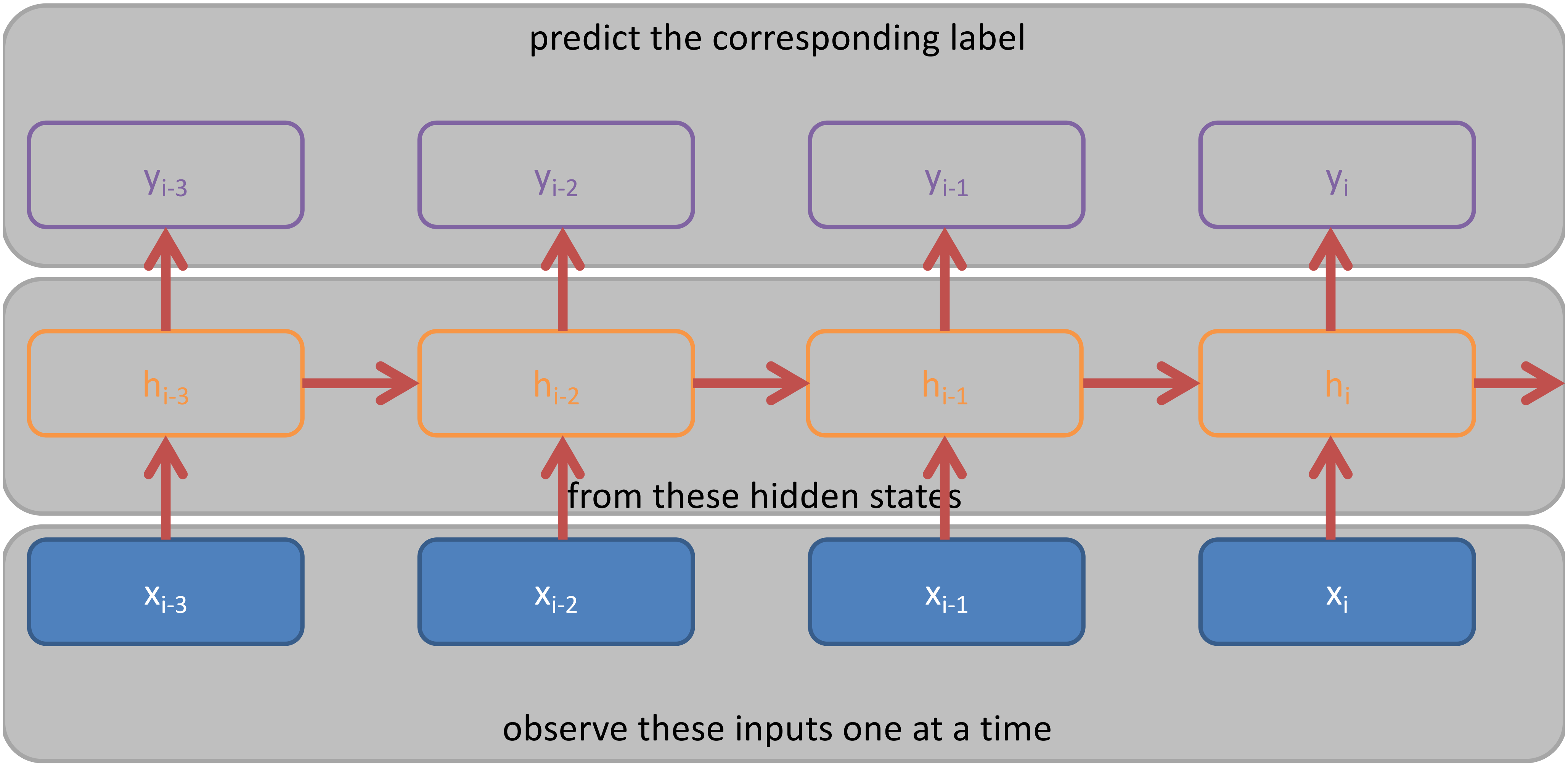
Recurrent Networks



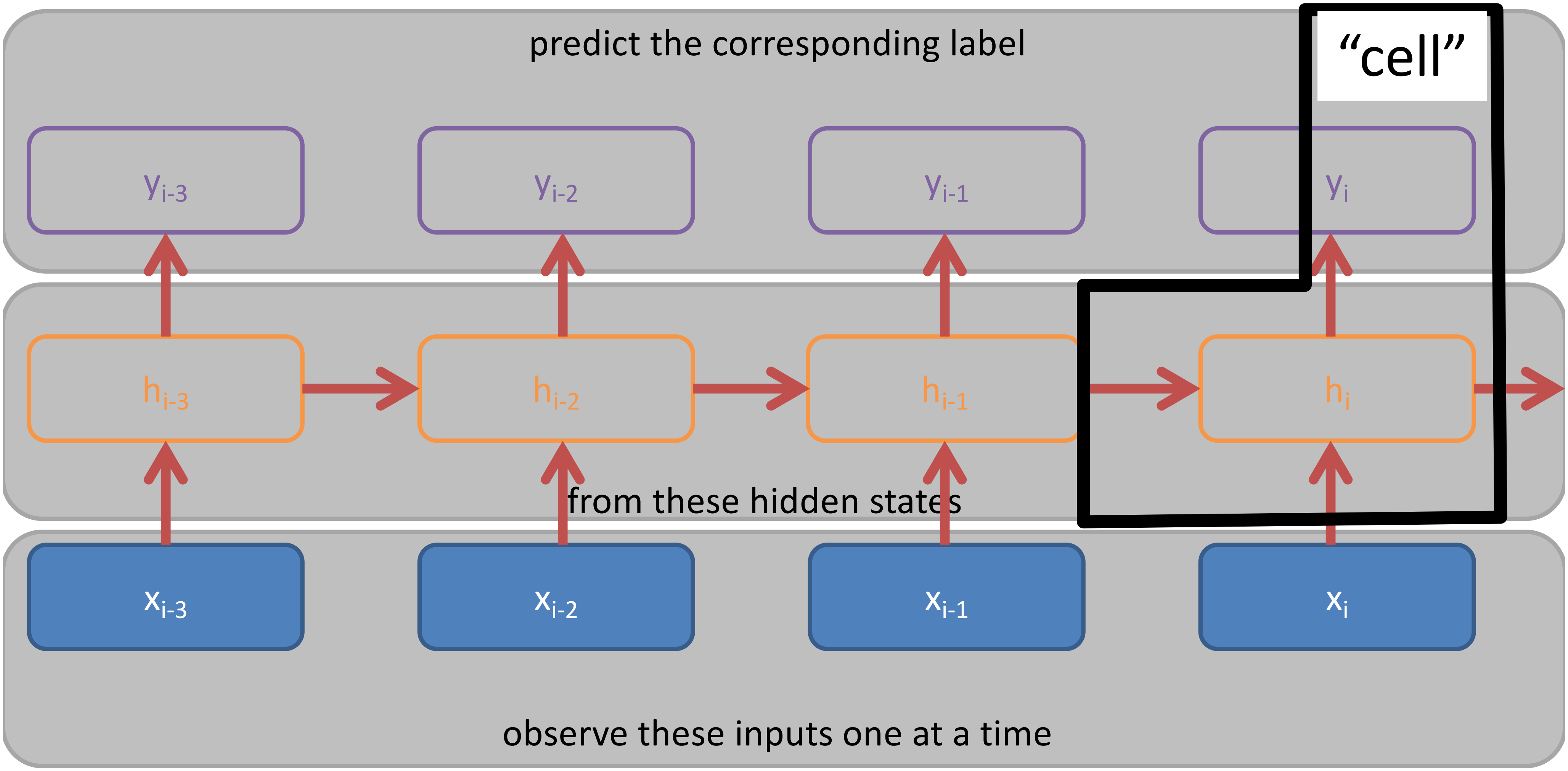
Recurrent Networks



Recurrent Networks



Recurrent Networks



Outline

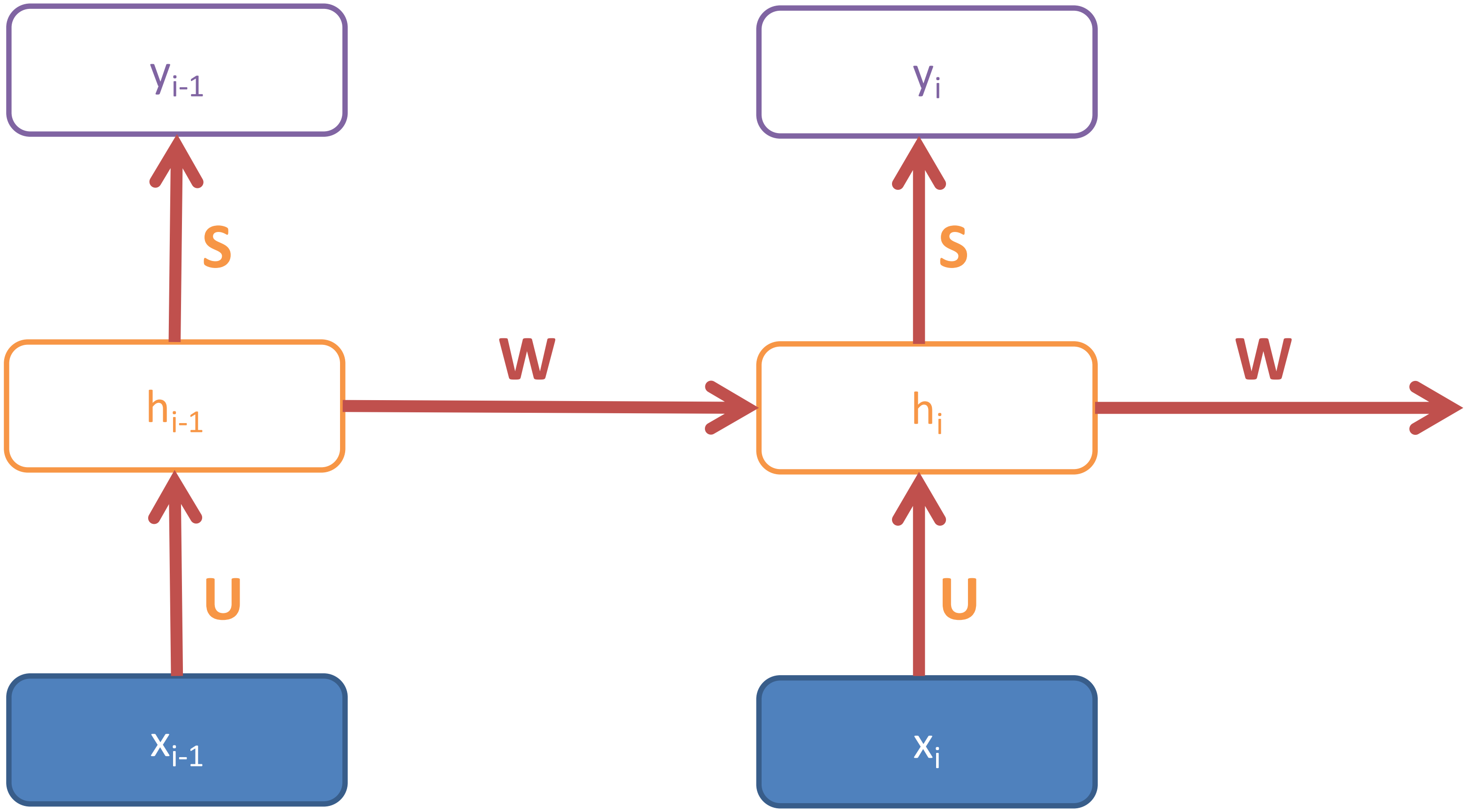
Convolutional Neural Networks

- What *is* a convolution?
- Multidimensional Convolutions
- Typical Convnet Operations
- Deep convnets

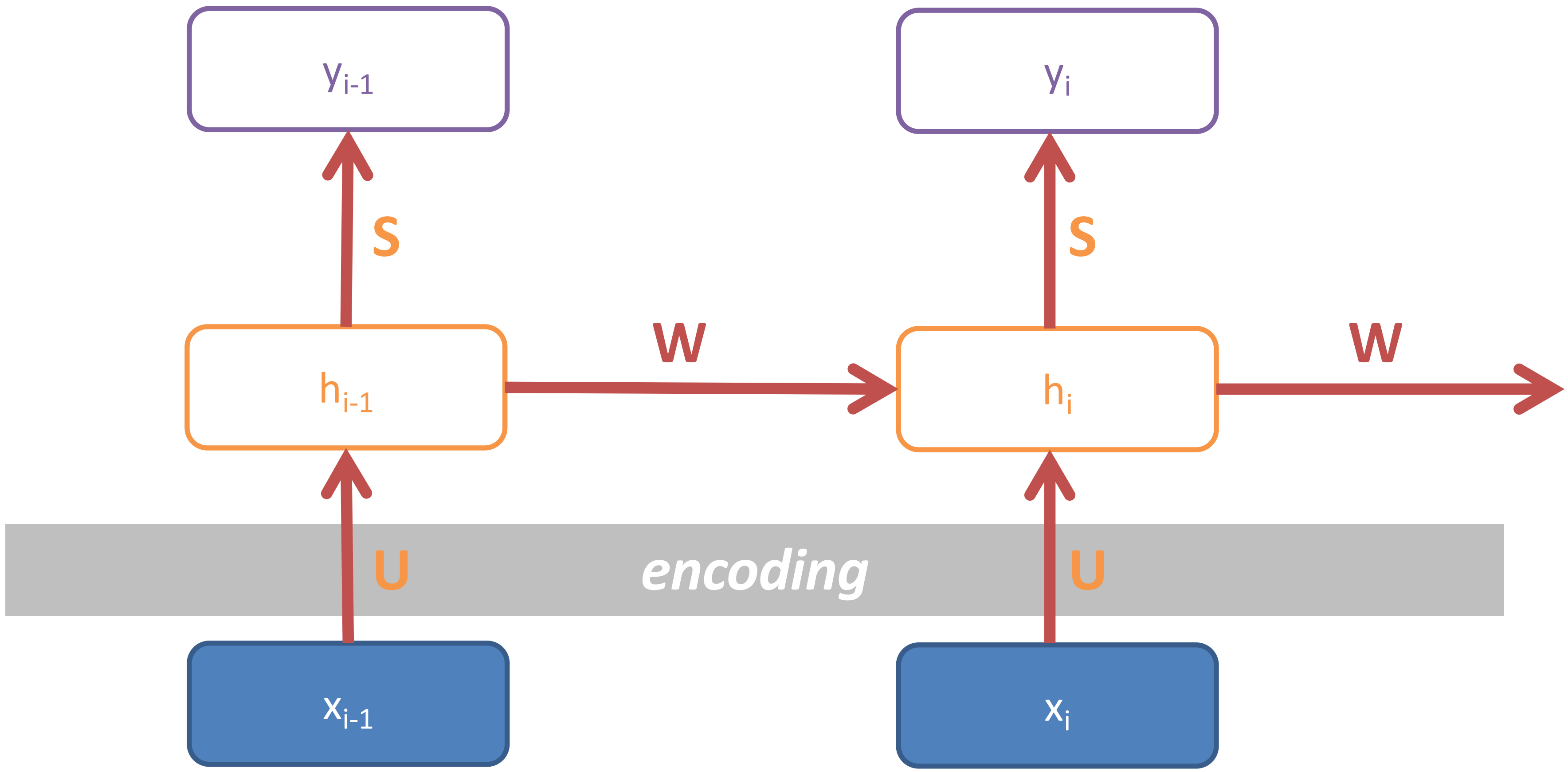
Recurrent Neural Networks

- Types of recurrence
- A basic recurrent cell
- BPTT: Backpropagation through time

A *Simple* Recurrent Neural Network Cell

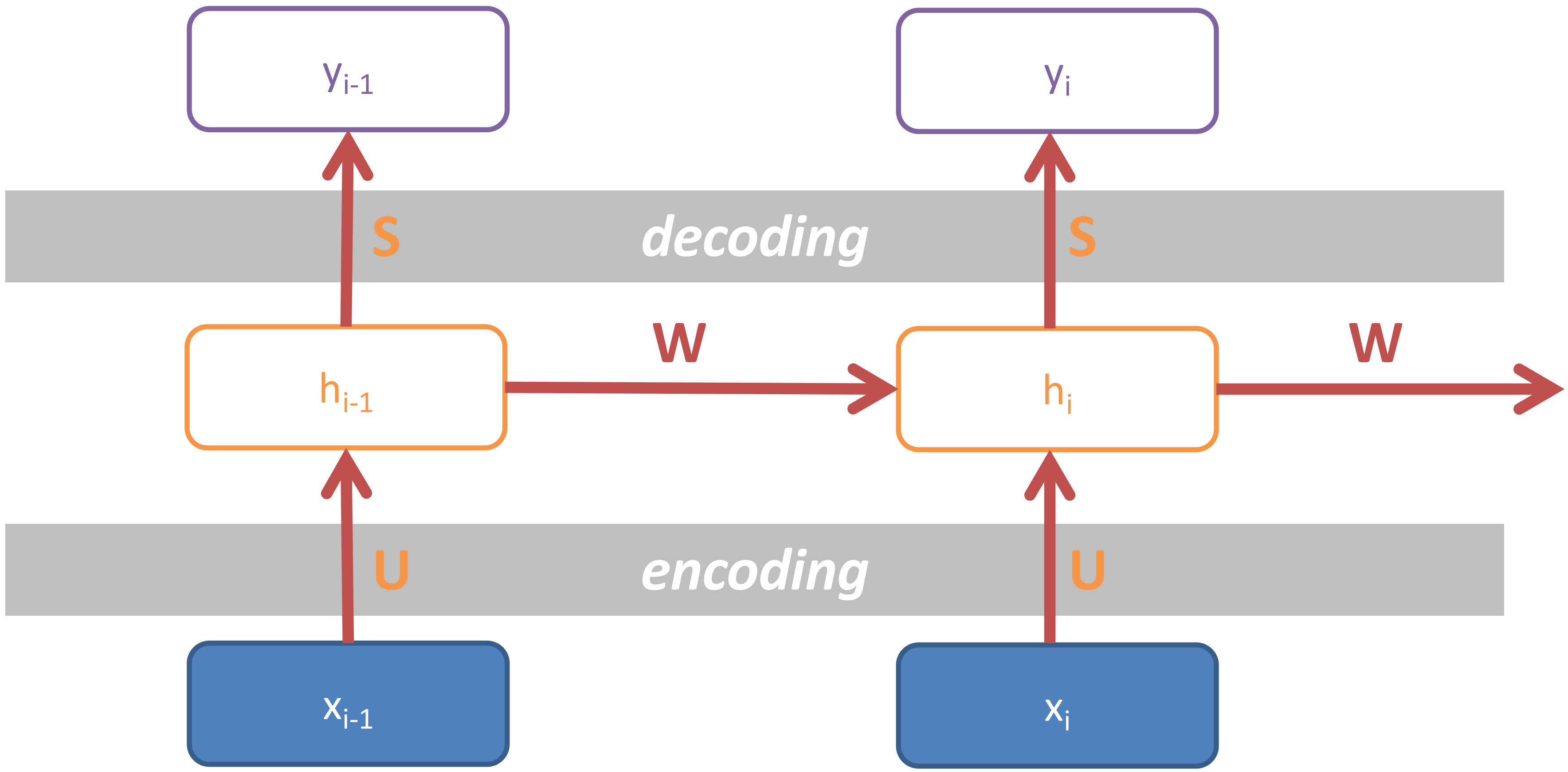


A *Simple* Recurrent Neural Network Cell



$$h_i = \tanh(W h_{i-1} + U x_i)$$

A *Simple* Recurrent Neural Network Cell



$$h_i = \tanh(W h_{i-1} + U x_i) \qquad y_i = \mathrm{softmax}(S h_i)$$