CAP 5516
Medical Image Computing
(Spring 2022)

Dr. Chen Chen
Center for Research in Computer Vision (CRCV)
University of Central Florida
Office: HEC 221
Address: 4328 Scorpius St., Orlando, FL 32816-2365
Email: chen.chen@crcv.ucf.edu
Web: https://www.crcv.ucf.edu/chenchen/
Lecture 5: Introduction to Deep Learning (1)
ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt

MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time

DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data
Why Deep Learning?

Hand engineered features are time consuming, brittle and not scalable in practice.

Can we learn the **underlying features** directly from data?

<table>
<thead>
<tr>
<th>Low Level Features</th>
<th>Mid Level Features</th>
<th>High Level Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines &amp; Edges</td>
<td>Eyes &amp; Nose &amp; Ears</td>
<td>Facial Structure</td>
</tr>
</tbody>
</table>
Why Now?

Neural Networks date back decades, so why the resurgence?

1. **Big Data**
   - Larger Datasets
   - Easier Collection & Storage

2. **Hardware**
   - Graphics Processing Units (GPUs)
   - Massively Parallelizable

3. **Software**
   - Improved Techniques
   - New Models
   - Toolboxes

4. **Open resource (GitHub)**
Machine Learning ≈ Looking for a Function

- Speech Recognition
  \[ f\left( \text{audio signal} \right) = \text{“How are you”} \]

- Image Recognition
  \[ f\left( \text{cat image} \right) = \text{“Cat”} \]

- Playing Go
  \[ f\left( \text{board with stones} \right) = \text{“5-5” (next move)} \]

- Dialogue System
  \[ f\left( \text{“Hi” (what the user said)} \right) = \text{“Hello” (system response)} \]

Credit: The following slides are adapted from Hung-yi Lee, Andrew, Stanford course
Framework

A set of function $f_1, f_2, \ldots$

Model

Image Recognition:

$$f(\text{cat}) = \text{“cat”}$$

$$f(\text{dog}) = \text{“dog”}$$

$$f(\text{money}) = \text{“money”}$$

$$f(\text{snake}) = \text{“snake”}$$
Framework

A set of function $f_1, f_2 \cdots$

Model

Goodness of function $f$

Training Data

Supervised Learning

Image Recognition:

$$f(\text{cat}) = \text{“cat”}$$

Better!

$$f_1(\text{cat}) = \text{“cat”}$$

$$f_2(\text{dog}) = \text{“dog”}$$

$$f_2(\text{money}) = \text{“money”}$$

$$f_2(\text{snake}) = \text{“snake”}$$

function input:

function output: “monkey” “cat” “dog”
Framework

Image Recognition:

\[ f(\text{image}) = \text{“cat”} \]

Step 1: A set of functions \( f_1, f_2, \ldots \)

Step 2: Goodness of function \( f \)

Step 3: Pick the “Best” Function \( f^* \)

Training Data:

- “monkey”
- “cat”
- “dog”

Using \( f^* \): \( “cat” \)
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function
Neural Network

Neuron

\[ z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b \]
Neural Network

Neuron

Sigmoid Function

\[ \sigma(z) = \frac{1}{1 + e^{-z}} \]

Activation function

weights

bias

\[ \sigma(z) \]

\[ 0.98 \]
Neural Network

Different connections lead to different network structures

The neurons have different values of weights and biases.

Weights and biases are network parameters $\theta$
Fully Connected Feedforward Network

Sigmoid Function

\[ \sigma(z) = \frac{1}{1 + e^{-z}} \]
Fully Connected Feedforward Network

Diagram of a fully connected feedforward network with nodes and edges labeled with values.
Given parameters $\theta$, define a function

$$f \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

This is a function. Input vector, output vector

Given network structure, define a function set
Fully Connected Feedforward Network

Deep means many hidden layers
Deep = Many hidden layers


Deep = Many hidden layers


Special structure

152 layers
# Activation Function

## Common Activation Functions

### Sigmoid Function

\[ g(z) = \frac{1}{1 + e^{-z}} \]

\[ g'(z) = g(z)(1 - g(z)) \]

### Hyperbolic Tangent

\[ g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \]

\[ g'(z) = 1 - g(z)^2 \]

### Rectified Linear Unit (ReLU)

\[ g(z) = \max(0, z) \]

\[ g'(z) = \begin{cases} 
1, & z > 0 \\
0, & \text{otherwise} 
\end{cases} \]

**NOTE:** All activation functions are non-linear

---


More details about the activation functions: [https://www.v7labs.com/blog/neural-networks-activation-functions](https://www.v7labs.com/blog/neural-networks-activation-functions)
Activation Function

- Why non-linear activation function?
Activation Function

- Why non-linear activation function?

The purpose of activation functions is to introduce non-linearities into the network.

What if we wanted to build a Neural Network to distinguish green vs red points?
Activation Function

• Why non-linear activation function?

The purpose of activation functions is to introduce non-linearities into the network.

Linear Activation functions produce linear decisions no matter the network size.
Activation Function

- Why non-linear activation function?

*The purpose of activation functions is to introduce non-linearities into the network.*

Linear Activation functions produce linear decisions no matter the network size.

Non-linearities allow us to approximate arbitrarily complex functions.
Activation Function

• Why non-linear activation function?

- **All layers of the neural network collapse into one**—with linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function). So a linear activation function turns the neural network into just one layer.

- A neural network with a linear activation function is simply a linear regression model. It has limited power and ability to handle complexity varying parameters of input data.
• Why non-linear activation function?

A linear activation function turns the neural network into just one layer.
Common Activation Functions

Sigmoid Function

\[ g(z) = \frac{1}{1 + e^{-z}} \]
\[ g'(z) = g(z)(1 - g(z)) \]

Hyperbolic Tangent

\[ g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \]
\[ g'(z) = 1 - g(z)^2 \]

Rectified Linear Unit (ReLU)

\[ g(z) = \max(0, z) \]
\[ g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases} \]

NOTE: All activation functions are non-linear.
Activation Function

- Activation function ReLU (rectified linear unit)
  - $ReLU(z) = \max\{z, 0\}$

The Rectified Linear Activation Function

Gradient 0

Gradient 1
Activation Function

- Generalizations of ReLU $g\text{ReLU}(z) = \max\{z, 0\} + \alpha \min\{z, 0\}$
  - Leaky-ReLU$(z) = \max\{z, 0\} + 0.01 \min\{z, 0\}$
  - Parametric-ReLU$(z)$: $\alpha$ learnable
Output Layer

- Softmax layer as the output layer

**Ordinary Layer**

\[ y_1 = \sigma(z_1) \]
\[ y_2 = \sigma(z_2) \]
\[ y_3 = \sigma(z_3) \]

In general, the output of the network can be any value. May not be easy to interpret.
• Softmax layer as the output layer

**Softmax Layer**

\[
\begin{align*}
y_1 &= e^{z_1} / \sum_{j=1}^{3} e^{z_j} \\
y_2 &= e^{z_2} / \sum_{j=1}^{3} e^{z_j} \\
y_3 &= e^{z_3} / \sum_{j=1}^{3} e^{z_j}
\end{align*}
\]

**Probability:**
- \(1 > y_i > 0\)
- \(\sum_i y_i = 1\)
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function
Example Application

- Handwriting Digit Recognition

What is needed is a function …

Input: 256-dim vector

Neural Network

Output: 10-dim vector
Example Application

Input

Output

Each dimension represents the confidence of a digit.

16 x 16 = 256

Machine

“2”

The image is “2”

0.1 is 1

0.7 is 2

0.2 is 0

Example Application

Input

Output

Each dimension represents the confidence of a digit.
Training Data

- Preparing training data: images and their labels

The learning target is defined on the training data.
Loss

A good function should make the loss of all examples as small as possible.

Loss can be **square error** or **cross entropy** between the network output and target.
Total Loss

For all training data …

\[ L = \sum_{r=1}^{R} l_r \]

As small as possible

Find a function in function set that minimizes total loss \( L \)

Find the network parameters \( \theta^* \) that minimize total loss \( L \)
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Testing or Inference
Convolutional Neural Network (CNN)

Widely used in image processing and computer vision
Why CNN for Image

- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.
Connecting to small region with less parameters

“beak” detector
Why CNN for Image

- The same patterns appear in different regions.

“upper-left beak” detector

Do almost the same thing

They can use the same set of parameters.

“middle beak” detector
Why CNN for Image

- Subsampling the pixels will not change the object

![Bird](image)

We can subsample the pixels to make the image smaller.

Less parameters for the network to process the image.
CNN for Image

Property 1
- Some patterns are much smaller than the whole image.

Property 2
- The same patterns appear in different regions.

Property 3
- Subsampling the pixels will not change the object.
The whole CNN

cat dog ……

Fully Connected Feedforward network

Convolution -> Max Pooling

Convolution -> Max Pooling

Flatten

Can repeat many times
The whole CNN

Property 1
- Some patterns are much smaller than the whole image

Property 2
- The same patterns appear in different regions.

Property 3
- Subsampling the pixels will not change the object

Can repeat many times

Convolution → Max Pooling → Convolution → Max Pooling → Flatten
CNN – Convolution

Those are the network parameters to be learned.

```
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
```

6 x 6 image

Filter/kernel 1

Matrix

```
1 -1 -1
-1 1 -1
-1 -1 1
```

Filter 2

Matrix

```
-1 1 -1
-1 1 -1
-1 1 -1
```

Each filter detects a small pattern (3 x 3).
CNN – Convolution

Filter/Kernel 1

<table>
<thead>
<tr>
<th>1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

6 x 6 image

stride=1
CNN – Convolution

If stride=2

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

6 x 6 image

Filter 1

\[
\begin{array}{ccc}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1 \\
\end{array}
\]

We set stride=1 below

3

-3
CNN – Convolution

Filter 1

stride=1

6 x 6 image

Property 2
CNN – Convolution

Do the same process for every filter

6 x 6 image

4 x 4 image (reduced size)
Demo

Image

Convolved Feature

Filter/kernel

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
CNN – Zero Padding

You will get another 6 x 6 image in this way

Zero padding
CNN – Color image

Color image

Filter 1

Filter 2

Filter 1

Filter 2
Input Volume (+pad 1) (7x7x3)  Filter W0 (3x3x3)  Filter W1 (3x3x3)  Output Volume (3x3x2)

```
x[;:,0]
0 0 0 0 0 0 0
0 2 0 0 2 1 0
0 2 1 0 2 2 0
0 2 1 0 2 2 0
0 2 2 1 2 2 0
0 0 1 1 0 1 0
0 0 0 0 0 0 0

w0[;:,0]
-1 -1 1
1 0 1
-1 0 -1
w0[;:,1]
0 0 1
-1 -1 0
-1 1 -1

w1[;:,0]
0 1 0
1 1 1
0 1 1
-2 2 -1

w1[;:,1]
-1 -1 -1
-1 1 -1
-1 1 -1
4 1 -2

h0[;:,0]
1

h1[;:,0]
0

```

http://cs231n.github.io/convolutional-networks/
Convolutional Network

- Convolution network is a sequence of these layers
Convolution - Intuition

Source: https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/
Visualizing CNN

Convolution v.s. Fully Connected

Image

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Convolution

<table>
<thead>
<tr>
<th>1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Fully-connected

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ x_1, x_2, \ldots, x_{36} \]
Filter 1

6 x 6 image

Only connect to 9 input, not fully connected

Less parameters!
6 x 6 image

Less parameters!

Even less parameters!
Thank you!

Question?
References and Slide Credits

• Many slides are adapted from the existing teaching or tutorial slides by Hung-yi Lee, Andrew Ng, Alexander Amini, Lex Fridman, and Stanford course - CS231n: Convolutional Neural Networks for Visual Recognition

• Special thanks to Dr. Hung-yi Lee for making his machine learning course slides and materials available

  – Youtube videos:
    https://www.youtube.com/watch?v=5tvmMX8r_OM&list=PLtBw6njQ RU-rwp5__7C0olVt26ZgjG9NI&index=1

• Lex Fridman, MIT Deep Learning and Artificial Intelligence Lectures: https://deeplearning.mit.edu/
  https://www.youtube.com/watch?v=O5xeyoRL95U