COMPUTER VISION: DEEP LEARNING LAB COURSE
DAY 1 – BASICS

SEBASTIAN HOUBEN
Schedule

Today

- Computer Vision and Deep Learning
- Image Classification
- Representation of images in Python
- Feature extraction
- Evaluating an image classifier
- Convolution
- German Traffic Sign Recognition Benchmark
Computer Vision

- Programs that process images as input
- Gain understanding of images or video
- Mimic performance of human visual system

- Typical tasks
  - Object detection
  - Object segmentation
  - Image registration
  - Pose estimation
  - Face recognition
  - Egomotion
  - Optical Flow
Computer Vision

- Programs that process images as input
- Gain understanding of images or video
- Mimic performance of human visual system

Typical tasks
- Object detection
- Object segmentation
- Image registration
- Pose estimation
- Face recognition
- Egomotion
- Optical Flow
Computer Vision

- Programs that process images as input
- Gain understanding of images or video
- Mimic performance of human visual system

Typical tasks

- Object detection
- Object segmentation
- Image registration
- Pose estimation
- Face recognition
- Egomotion
- Optical Flow
Computer Vision

- Programs that process images as input
- Gain understanding of images or video
- Mimic performance of human visual system

- Typical tasks
  - Object detection
  - Object segmentation
  - Image registration
  - Pose estimation
  - Face recognition
  - Egomotion
  - Optical Flow
Computer Vision

- Programs that process images as input
- Gain understanding of images or video
- Mimic performance of human visual system

- Typical tasks
  - Object detection
  - Object segmentation
  - Image registration
  - Pose estimation
  - Face recognition
  - Egomotion
  - Optical Flow
Deep Learning

Popular computer vision technique

- 2012 ImageNet Challenge significantly improved by a new method called AlexNet
  - Building on technique from 1999 (LeCun)
  - That builds on technique from 1980 (Fukushima)
- Let the computer figure out itsself how to solve a problem
- Very successful in nearly all areas of computer vision
  - Defining state-of-the-art
- Prerequisites / reasons for hype
  - Lots of data for a problem
  - Fast parallel architectures (GPUs)
  - New powerful libraries
Image Classification

- Given an image tell me what it depicts
- One of a fixed number of exclusive choices
  - Image depicts one uniquely identifiable object
  - Image may only depict a certain set of objects
- Distinguishable object choices are called classes
- Correct class of an image is called label
- A classification problem with only two classes is called binary
Image Classification Challenges

- Object may be depicted with different acquisition techniques
- Different view angles (geometry)
- Intraclass variation
- Illumination
- Deformation
- Occlusion
- Background clutter
Representation of images in Python
Representation of images in Python
Representation of images in Python

- Each picture element (pixel) is composed of three values
  - R for the red component
  - G for the green component
  - B for the blue component
Representation of images in Python

- Each picture element (pixel) is composed of three values:
  - R for the red component
  - G for the green component
  - B for the blue component
Representation of images in Python

- Each picture element (pixel) is composed of three values
  - R for the red component
  - G for the green component
  - B for the blue component
- Images are often represented in matrix structures
  - Unclear where pixel (0,0) or (1,1) is
  - Unclear which direction is given first
- Watch your data type (OpenCV is picky)
  - uint8 [0,255] (OpenCVs favorite)
  - short [-32768, 32767]
  - float32 (for visualizing)
INTRODUCTION TO DEEP LEARNING FOR COMPUTER VISION

Image Classification

- **Linear classifier** (choice today) finds hyperplane to separate sets of points
- Transform images to point representation
  - i.e. **Feature Extraction**
  - Low-dimensional
  - Compact representation

Feature Extraction

\[
\begin{bmatrix}
2 \\
5 \\
1 \\
8
\end{bmatrix}
\]

Linear Classifier

\{cat, dog\}
Image Classification

**Linear classifier** (choice today) finds hyperplane to separate sets of points

Transform images to point representation

- i.e. **Feature Extraction**
- Low-dimensional
- Compact representation

**Evaluation**

- Error rate:
  Percentage of wrongly classified images
- Confusion matrix:
  Error for each pair of classes
**Convolution**

- Basic image processing operation: Transforms image to image
- Task: Computer similarity of each pixel with given template (kernel)

\[
\begin{align*}
  &\frac{1}{10} \cdot 4 + \frac{1}{10} \cdot 3 + \frac{1}{10} \cdot 2 + \frac{1}{10} \cdot 6 + \frac{2}{10} \cdot 2 + \frac{1}{10} \cdot 4 + \frac{1}{10} \cdot 3 + \\
  &\frac{1}{10} \cdot 5 + \frac{1}{10} \cdot 3
\end{align*}
\]
Convolution

- Basic image processing operation
- Task: Compute similarity of each pixel with given template (kernel)
- Pay attention to range of kernel and image!
Convolution

- Basic image processing operation
- Task: Compute similarity of each pixel with given template (kernel)
- Pay attention to range of kernel and image!
Convolution

- Basic image processing operation
- Task: Computer similarity of each pixel with given template (kernel)
- Pay attention to range of kernel and image!
- Pad borders with zeros

Stride
Convolution
German Traffic Sign Recognition Benchmark

- 38,000 images from (German) traffic signs
  - Vienna Convention
- 43 classes
- Over 1,000 different traffic signs instances
- Variance
  - Illumination
  - Motion Blur
  - Clutter
  - Dirt / Graffiti / Stickers
  - Occlusion
  - Angle
German Traffic Sign Recognition Benchmark

- Filename structure
  - 0000CC/00XXX_00YYY.ppm
  - CC = class index
  - XXX = instance of class index
  - YYY = image of instance index
- e.g. 00003/00004_00024.ppm
  - Class 3: speed limit 60
  - Instance 4
  - Image 24
- Border of at least 5 pixel
- Border of around 10% of traffic sign size
German Traffic Sign Recognition Benchmark

- Best human: 1.16% error rate
- Best machine classifier (2011): 0.54% error rate
Hands-On Python: Numpy

- [https://docs.scipy.org/doc/numpy-dev/user/quickstart.html](https://docs.scipy.org/doc/numpy-dev/user/quickstart.html)
- “Matlab for Python“
- Matrix / Tensor manipulation (numeric)
- Fundamental library for nearly all of scientific computing in Python
- Tensorflow (Day 3 and 4) corresponds in large parts to Numpy
import numpy as np

A = np.array([[0, 1, 2], [2, 3, 4]])  # 2x3 matrix

A.shape  # (2, 3)
A.size   # 6 (numel in Matlab)
A.dtype.name  # ‘int64’

B = [[0, 1, 2], [2, 3]]  # ok
A_ = np.array([[0, 1, 2], [2, 3]])  # error
Numpy: Initialization

A = np.array( [[ 0, 1, 2], [2, 3, 4]] ) # 2x3 matrix

B = np.zeros( (2, 3) )
# 2x3 matrix

C = np.ones( (2, 3, 4), dtype = np.int16 )
# 2x3x4 tensor, created with data type
C_ = np.zeros_like(C)

D = np.empty( (2, 3) )
# 2x3 matrix, uninitialized, np.random.rand, np.random.randn

E = np.arange( 0, 2, 0.3 )
# array([ 0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])

F = np.linspace( 0, 9 )
# array([ 0. , 0.25, 0.5, 0.75, 1. , 1.25, 1.5 , 1.75, 2. ])
Numpy: Operations

# operations usually work element-wise (even *)

```python
a = np.array([20,30,40,50])
b = np.arange(4)  # array([0, 1, 2, 3])

a – b  # array([20, 29, 38, 47])
b**2  # array([0, 1, 4, 9])
10*np.sin(a)  # [9.12945251, -9.88031624, 7.4511316, -2.62374854]
a<35  # array([True, True, False, False], dtype=bool)

np.logical_and(a < 35, b > 0)  # [False, True, False, False]  # brackets are necessary!
(a < 35) & (b > 0)

a.dot(b.transpose())  # matrix product (matmul)

a.astype(np.uint8)  # np.float32, np.int32, np.int16
```
import numpy as np

A = np.arange(0, 20)

A = A.reshape([4, 5])  # 4 rows, 5 columns
A.ravel()  # back to np.arange(0, 20)

A.min()  # smallest element
A.min(axis=1)  # shape = (4,), iterate along the columns (rowwise minimum element)
# max, sum, mean, var, cumsum

B = np.arange(0, 24).reshape([2, 3, 4])

C = B.sum(axis=1)  # shape = (2, 1, 4)
C = C.squeeze()  # shape = (2, 4)

C = np.expand_dims(C, axis=1)  # shape = (2, 1, 4)
import numpy as np

A = np.arange(0, 5)  # [0, 1, 2, 3, 4]
A[0] = 5  # [5, 1, 2, 3, 4]

A[-2] = 4  # [5, 1, 4, 4, 4], A.shape[0] - 2 = 5 - 2 = 3

A[1:4:2] = 0  # [5, 0, 4, 0, 4], start with 1, stepwidth 2, stay below 4
A[1:-1:2] = 0  # [5, 0, 4, 0, 4], start with 1, stepwidth 2, stay below A.shape[0] - 1 = 5 - 1
A[1::2] = 0  # [5, 0, 4, 0, 4], start with 1, stepwidth 2, stay below A.shape[0] = 5
Numpy: Indexing

```python
import numpy as np

A = np.arange(0, 6).reshape([2, 3])  # [[0, 1, 2], [3, 4, 5]]

A[0, 1] = -1  # [[0, -1, 2], [3, 4, 5]]
A[1, :] = -1   # [[0, -1, 2], [-1, -1, -1]]
A[1, 1:] = 6   # [[0, -1, 2], [-1, 6, 6]]

A[ np.array([[True, False, True][True, True, False]], dtype=bool) ] = 1  # [[1, -1, 1], [1, 1, 6]]

A[ A > 1 ] = -1  # [[1, -1, 1], [1, 1, -1]]
```
Numpy: Concatenating

```python
import numpy as np

a = np.array([[1, 2], [3, 4]])  # [[1, 2], [3, 4]]
b = np.array([[5, 6]])  # [5, 6]
	np.concatenate( (a, b), axis = 0 )  # [[1, 2], [3, 4], [5, 6]], same as np.hstack( (a,b) )
	np.concatenate( (a, b.transpose()), axis = 1 )  # [[1, 2, 5], [3, 4, 6]], same as np.vstack( (a,b.transpose()) )
```
Hands-On Python: OpenCV

- see the handout for some important functions
- OpenCV can work with numpy-arrays
- But: Be careful about data types
  - Use uint8 if working in OpenCV
  - Rather receive wrong results than errors
QUESTIONS?
EXERCISES.