COMP3420 — Artificial Intelligence for Text and Vision

Week 01 Lecture 2: Image Processing in Python

Diego Mollá

COMP3420 2023H1

Abstract

In this lecture we will do a quick revision of Python and its use for image processing.

Update February 16, 2023

Contents

Reading

- LinkedIn Learning `https://www.linkedin.com/learning/computer-vision-deep-dive-in-python`, Section 2 "The Basics of Image Processing".

Additional Reading

- `https://docs.python.org/3/tutorial/index.html`

1 A Review of Python

1.1 Practicalities

Why Python

Scripting Language

- Rapid prototyping.
- Platform neutral.

Python

- Even easier prototyping.

  - jupyter notebooks.
- Clean, object oriented.
- Good text manipulation.
- Wide range of libraries.
  - Specific libraries for text and image processing.
  - pandas, sklearn, tensorflow for data mining.
  - NumPy and SciPy for scientific computing.
  - matplotlib and pyplot for plotting.

Installing Python

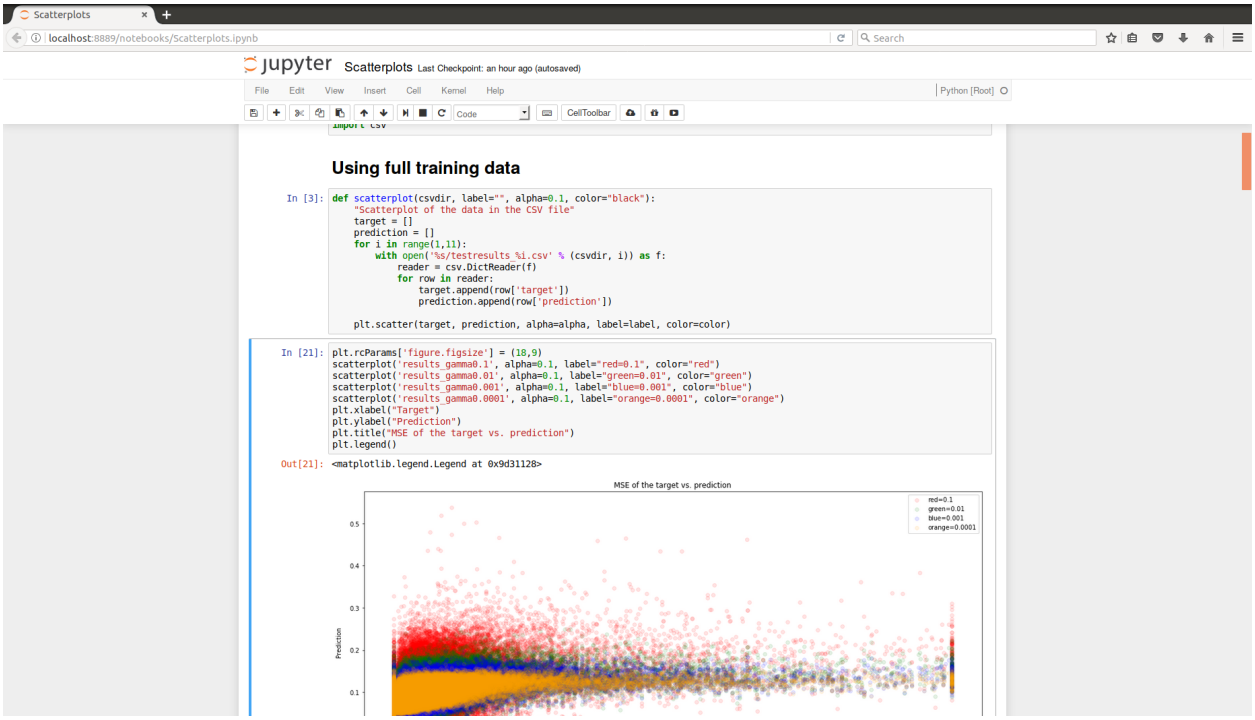
- Official Python at `http://www.python.org`.
- We will use the Anaconda Python environment from `https://www.anaconda.com/distribution/`.
- Current version is 3.x *do not use 2.x.*
- Windows/Mac/Linux versions.
- Download includes many libraries.
- Anaconda includes Jupyter notebooks and Spyder, a useful IDE, plus numerous libraries.

Popular IDEs for Python

1. Eclipse + Pydev `https://www.pydev.org/`
2. Pycharm `https://www.jetbrains.com/pycharm/`
3. Visual Studio Code `https://code.visualstudio.com`
4. IDLE `https://docs.python.org/3/library/idle.html`
5. Spyder `https://github.com/spyder-ide/spyder`

My Recommendation
Visual Studio Code

Jupyter Notebooks



1.2 Basic Python

Beginning Python

This and other Python code available as Jupyter notebooks in github: `https://github.com/COMP3420-2023S1/public_material_2023S1`.

```
def hello (who):                                    # 1
    """Greet somebody"""                            # 2
    print("Hello " + who + "!")                     # 3

hello("Diego")                                      # 4
hello('World')                                      # 5
people = ['Greg', "Abid", 'Diego']                  # 6
for person in people:                               # 7
    hello(person)                                   # 8
```

Comments, line per line:

1. Defines a new function/procedure called hello which takes a single argument. Note that python variables are not typed, who could be a string, integer, array ... The line ends with a colon (:) which means we're beginning an indented code block ...

2. Firstly note that there are no brackets delimiting the body of the procedure, Python instead uses indentation to delimit code blocks. So, getting the indentation right is crucial!

   This line (2) is a documentation string for the procedure which gets associated with it in the python environment (IDLE uses it for balloon help). The three double quotes delimit a multi-line string (could use ' or " in this context).

3. This is the body of the procedure, **print** is a built in command in python. Note that the Python 2.x versions do not use round brackets, this is a major difference with Python 3.x. We also see here the

+ operator used on strings (I'm assuming who is a string) to perform concatenation — thus we have operator overloading based on object type just like other OO languages.

4. Here I'm calling the new procedure with a literal string argument delimited by ".
5. And here delimited by ' — both of these delimiters are equivalent, use one if you want to include the other in the string, eg "Steve's".
6. This defines a variable people to have a value which is a list of strings, lists are 1-D arrays and the elements can be any python object (including lists).
7. A **for** loop over the elements of the list. Again the line ends with a colon indicating a code block to follow.
8. Call the procedure with the variable which will be bound to successive elements of the list.

Core Data Types

- Strings.
- Numbers (integers, float, complex).
- Lists.
- Tuples (immutable sequences).
- Dictionaries (associative arrays).

Lists

```
>>> a = ['one', 'two', 3, 'four']
>>> a[0]
'one'
>>> a[-1]
'four'
>>> a[0:3]
['one', 'two', 3]}
>>> len(a)
4
>>> a[1]=2
>>> a
['one', 2, 3, 'four']
>>> a.append('five')
>>> a
['one', 2, 3, 'four', 'five']
>>> top = a.pop()
>>> a
['one', 2, 3, 'four']
>>> top
'five'
```

List Comprehensions

```
>>> a = ['one', 'two', 'three', 'four']
>>> len(a[0])
3
>>> b = [w for w in a if len(w) > 3]
>>> b
['three', 'four']
>>> c = [[1,'one'],[2,'two'],[3,'three']]
>>> d = [w for [n,w] in c]
>>> d
['one', 'two', 'three']
```

For more details on list comprehensions: `https://docs.python.org/3/tutorial/datastructures.html`

Tuples

- Tuples are a sequence data type like lists but are immutable:
  - Once created, elements cannot be added or modified.
- Create tuples as literals using parentheses:
  a = ('one', 'two', 'three')
- Or from another sequence type:
  a = ['one', 'two', 'three']
  b = **tuple**(a)
- Use tuples as fixed length sequences: memory advantages.

Dictionaries

- Associative array datatype (hash).
- Store values under some hash key.
- Key can be any immutable type: string, number, tuple.

```
>>> names = dict()
>>> names['madonna'] = 'Madonna'
>>> names['john'] = ['Dr.', 'John', 'Marshall']
>>> list(names.keys())
['madonna', 'john']
>>> ages = {'steve':41, 'john':22}
>>> 'john' in ages
True
>>> 41 in ages
False
>>> for k in ages:
...     print(k, ages[k])
steve 41
john 22
```

Organising Source Code: Modules

- In Python, a module is a single source file which defines one or more procedures or classes.
- Load a module with the **import** directive.

```
import mymodule
```

- This loads the file mymodule.py and evaluates its contents.
- By default, all procedures are put into the mymodule namespace, accessed with a dotted notation:
  - mymodule.test() — calls the test() procedure defined in mymodule.py

Modules

- Can import names into global namespace.

```
from mymodule import test, doodle
from mymodule import *
```

- The Python distribution comes with many useful modules.

```
from math import *
x = 20 * log(y)
import webbrowser
webbrowser.open('http://www.python.org')
```

Defining Modules

- A module is a source file containing Python code.
  - Usually class/function definitions.
- First non-comment item can be a docstring for the module.

```
# my python module
"""This is a python module to
do something interesting"""

def foo(x):
    'foo the x'
    print('the foo is ' + str(x))
```

Documentation in Python

- Many Python objects have associated documentation strings.
- Good practice is to use these to document your modules, classes and procedures.
- Docstring can be retrieved as the __doc__ attribute of a module/class/procedure name:

```
def hello (who):
    """Greet somebody"""
    print("Hello " + who + "!")
```

```
>>> hello.__doc__
'Greet somebody'
```

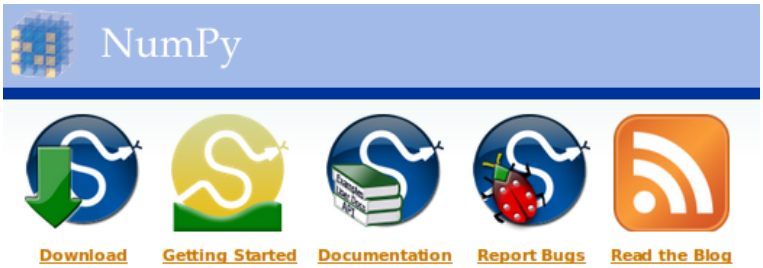
- The function **help**() uses the docstring to generate interactive help.

1.3 Vectors and Matrices in Python

Vectors and Matrices in Python

numpy

- Python's numpy is a collection of libraries that include manipulation of vectors and matrices.
- `http://www.numpy.org/`
- It's pre-loaded in the Anaconda distribution.



Manipulating Vectors

```
>>> import numpy as np
>>> a = np.array([1,2,3,4])
>>> a[0]
1
>>> a[1:3]          # slicing
array([2, 3])
>>> a+1             # add a constant to a vector
array([2, 3, 4, 5])
>>> b=np.array([2,3,4,5])
>>> a+b             # add two vectors
array([3, 5, 7, 9])
>>> a*b             # pairwise multiplication
array([ 2,  6, 12, 20])
```

```
>>> np.dot(a,b) # dot product between vectors, a . b
40
```

Manipulating Matrices

```
>>> x = np.array([[1,2,3],[4,5,6]])
>>> x
array([[1, 2, 3],
       [4, 5, 6]])
>>> y = np.array([[1,1,1],[2,2,2]])
>>> x+y              # add two matrices
array([[2, 3, 4],
       [6, 7, 8]])
>>> x*y              # pairwise multiplication
array([[ 1,  2,  3],
       [ 8, 10, 12]])
>>> x.T              # transpose
array([[1, 4],
       [2, 5],
       [3, 6]])
>>> np.dot(x.T,y)  # dot product
array([[ 9,  9,  9],
       [12, 12, 12],
       [15, 15, 15]])
>>>
```

2 Image Processing in Python

Image Processing in Python

- Images are represented as Python arrays.
- The first 2 dimensions of the array represent the pixel.
- The third dimension represents the pixel colour, which can be a vector.
- Each element of the pixel vector represents a channel. There are several options:
  - 3 channels for Red, Green, Blue.
  - 4 channels for Red, Green, Blue, Alpha (transparency).
  - There are other possibilities which we will not cover here.

Practical Demonstration

See Jupyter notebook "W01L2Python.ipynb".

Take-home Messages

- Get to learn Python.
  - If you know how to program in another language, read this tutorial: `https://docs.python.org/3/tutorial/index.html`
- Practice with Python's numpy and matplotlib to read and manipulate images.

What's Next

Week 2

- Machine Learning for Image Classification.

Reading

- Practical Machine Learning for Computer Vision, Chapters 1, 2.