Week 2, Lesson 2:
Images, image density, pixel wise loss function

Alex Treacher

UTSouthwestern

Departments: Lyda Hill Department of Bioinformatics, Radiology, and Advanced Imaging Research Center

Goals for this lesson:

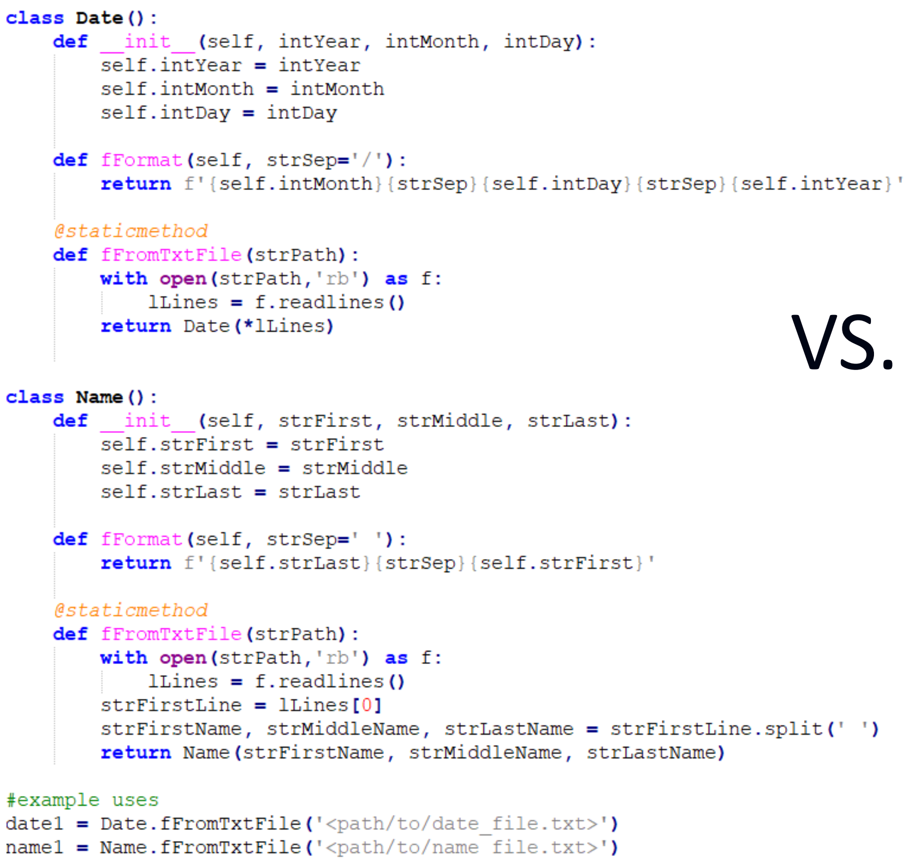
1. **Practice using OOP**
   - Develop/modify a custom class
     - Adding new class methods
     - Learn about staticmethods
   - Work with the custom class
2. **Learn basic image manipulation and analysis methods**
   1. How images are stored in computers
   2. How to load images in python
   3. How to plot images
   4. Image distributions
   5. Difference images
   6. Pixel wise loss function
      1. Mean square error (MSE)

Quick! More OOP!! Staticmethod

- A static method are similar to class methods in that they are bound to a specific class.
- However, they are independent of the state of the object, meaning they do not require class creation
- They are largely used for organization, and code clarity/readability/context.
- A very common use is to create an object of a class in a specific manor, e.g. loading from a text file.
- Practically speaking:
  - Functions that are related to a class, but don’t require an object, can be organized with that class.
  - Similar methods across classes can have consistent naming

Staticmethod example

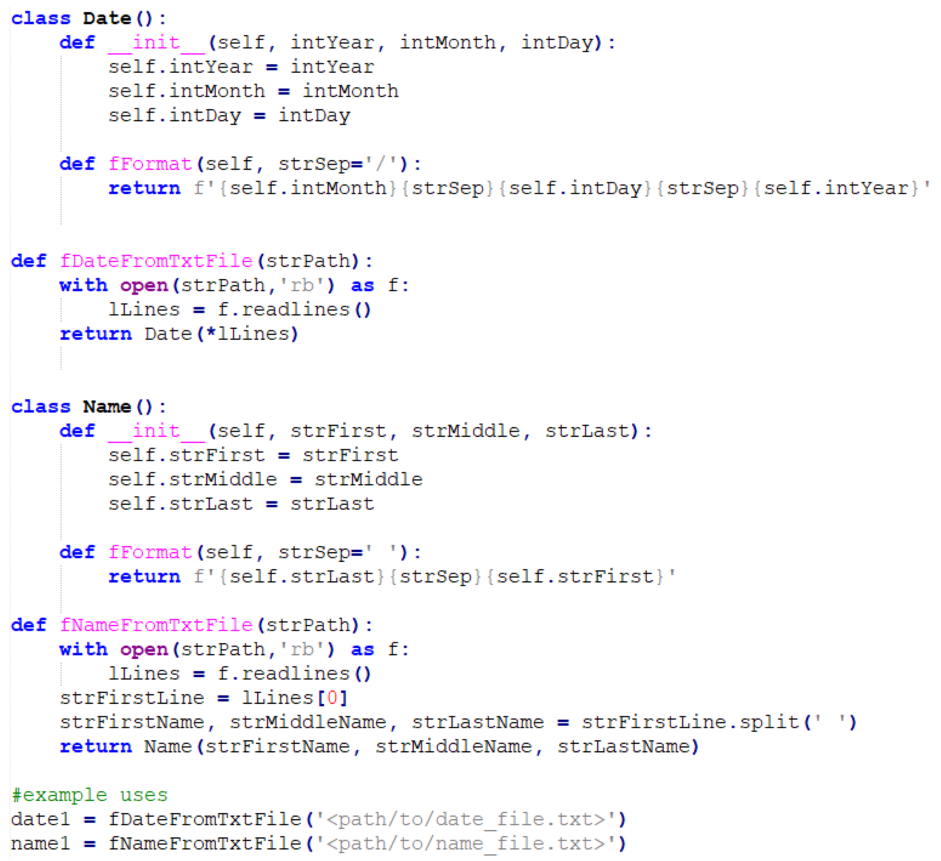
Static method



```python
class Date():
    def __init__(self, intYear, intMonth, intDay):
        self.intYear = intYear
        self.intMonth = intMonth
        self.intDay = intDay

    def fFormat(self, strSep='/'):
        return f'{self.intMonth}{strSep}{self.intDay}{strSep}{self.intYear}'

    @staticmethod
    def fFromTxtFile(strPath):
        with open(strPath,'rb') as f:
            lLines = f.readlines()
        return Date(*lLines)


class Name():
    def __init__(self, strFirst, strMiddle, strLast):
        self.strFirst = strFirst
        self.strMiddle = strMiddle
        self.strLast = strLast

    def fFormat(self, strSep=' '):
        return f'{self.strLast}{strSep}{self.strFirst}'

    @staticmethod
    def fFromTxtFile(strPath):
        with open(strPath,'rb') as f:
            lLines = f.readlines()
        strFirstLine = lLines[0]
        strFirstName, strMiddleName, strLastName = strFirstLine.split(' ')
        return Name(strFirstName, strMiddleName, strLastName)

#example uses
date1 = Date.fFromTxtFile('<path/to/date_file.txt>')
name1 = Name.fFromTxtFile('<path/to/name_file.txt>')
```

VS.

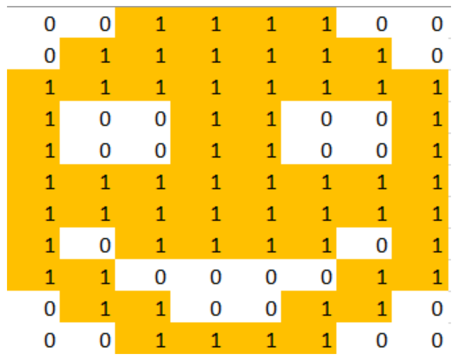
Function



```python
class Date():
    def __init__(self, intYear, intMonth, intDay):
        self.intYear = intYear
        self.intMonth = intMonth
        self.intDay = intDay

    def fFormat(self, strSep='/'):
        return f'{self.intMonth}{strSep}{self.intDay}{strSep}{self.intYear}'


def fDateFromTxtFile(strPath):
    with open(strPath,'rb') as f:
        lLines = f.readlines()
    return Date(*lLines)


class Name():
    def __init__(self, strFirst, strMiddle, strLast):
        self.strFirst = strFirst
        self.strMiddle = strMiddle
        self.strLast = strLast

    def fFormat(self, strSep=' '):
        return f'{self.strLast}{strSep}{self.strFirst}'

def fNameFromTxtFile(strPath):
    with open(strPath,'rb') as f:
        lLines = f.readlines()
    strFirstLine = lLines[0]
    strFirstName, strMiddleName, strLastName = strFirstLine.split(' ')
    return Name(strFirstName, strMiddleName, strLastName)

#example uses
date1 = fDateFromTxtFile('<path/to/date_file.txt>')
name1 = fNameFromTxtFile('<path/to/name_file.txt>')
```

Images

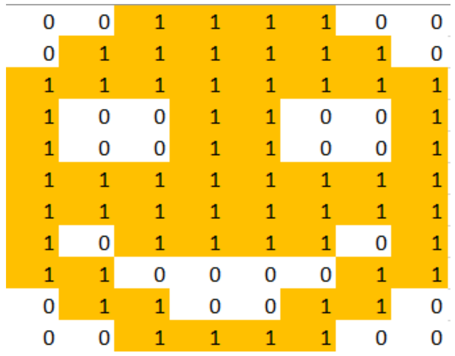
- **2D Images are represented as an array of values**
  - Dimension of the image is the height by width
  - E.g. 1080x1920 (aka 1080p)
    - 1080 pixels tall, 1920 pixels wide
  - Color images include 3 channels
    - One for each RGB
    - 1080x1920x3



0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	0
0	0	1	1	1	1	0	0

Images

- **2D Images are represented as an array of values**
  - Dimension of the image is the height by width
  - E.g. 1080x1920 (aka 1080p)
    - 1080 pixels tall, 1920 pixels wide
  - Color images include 3 channels
    - One for each RGB
    - 1080x1920x3



0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	0
0	0	1	1	1	1	0	0

- **Other dimensions are easy to add:**
  - 3 Color channel
    - Color images (example in next slide)
    - E.g. 1080x1920x3
  - Temporal dimension
    - Live cell imaging (2D or 3D + time=3 or 4D)
    - fMRI (3D + time = 4D)
    - Video (2D + time = 3D)

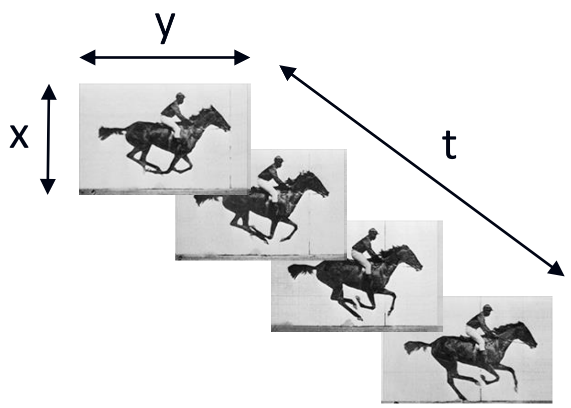
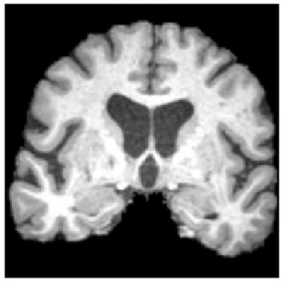


Image resolution

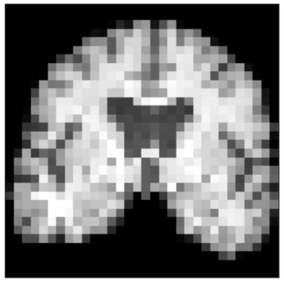
- **The resolution of real word imaging describes the size of the pixel/voxel (3D pixel).**
- **This size can describe a spatial and temporal description.**
- **The higher the resolution, the smaller the dimension**
- **For example**
  - Often the nature of imaging is that you can choose high res spatial OR temporal, but not both.
    - E.g. you can increase the temporal resolution, but cause a reduction in the spatial resolution.
    - 2D image of size 100mm by 100mm, and you wanted to image for 10 mins

	Spatial Res	**Temporal Res**	**Total Pixels**
↑ spatial res, ↓temporal res	0.25mm	2 sec	800,000
↓ spatial res, ↑ temporal res	0.5mm	.5 sec	800,000

Hi Res MRI
.8mm x .8mm

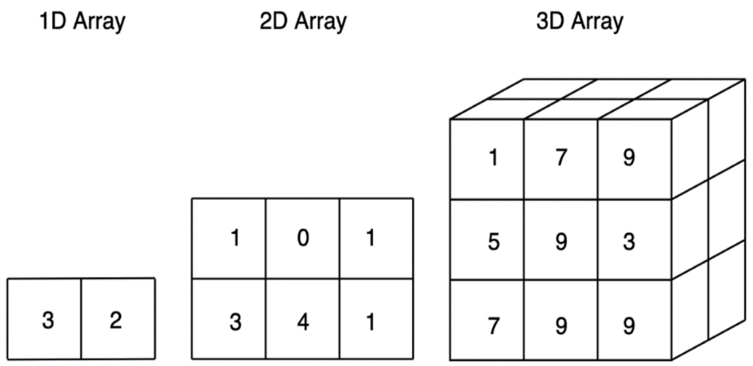


Low Res MRI
4.8mm x 4.8mm

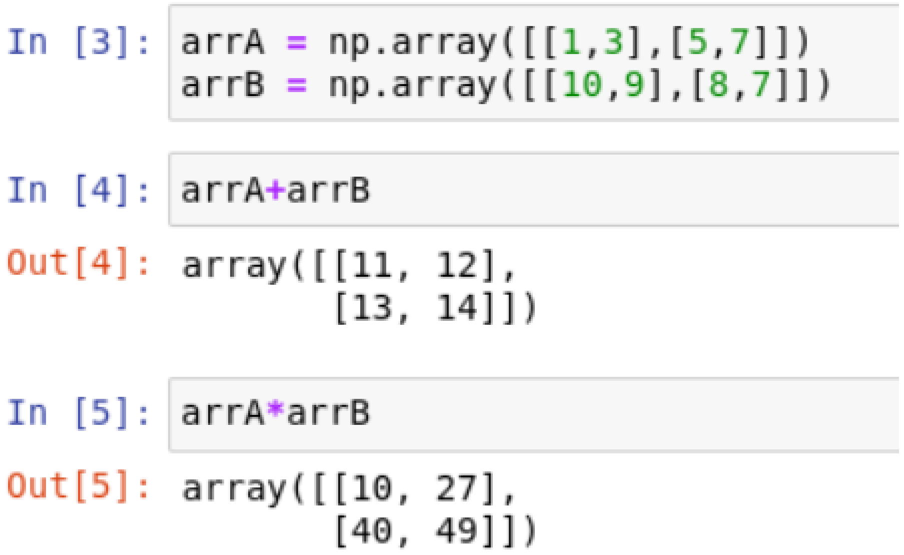


Numpy (and Pandas)

- **Numpy is a python module that adds support for multi-dimensional arrays, and their manipulation**
  - https://numpy.org/doc/



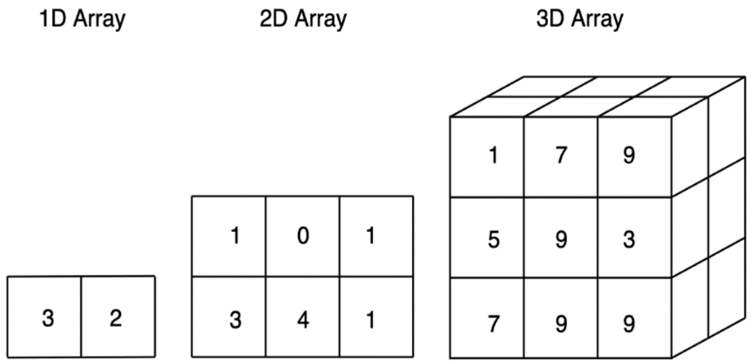
https://towardsdatascience.com/numpy-array-cookbook-generating-and-manipulating-arrays-in-python-2195c3988b09



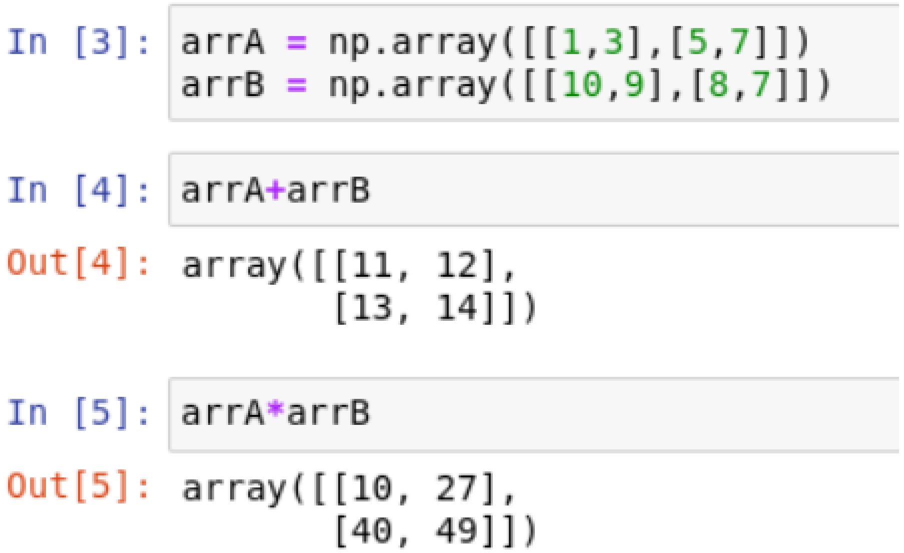
```
In [3]:  arrA = np.array([[1,3],[5,7]])
         arrB = np.array([[10,9],[8,7]])

In [4]:  arrA+arrB

Out[4]:  array([[11, 12],
                [13, 14]])

In [5]:  arrA*arrB

Out[5]:  array([[10, 27],
                [40, 49]])
```

Numpy (and Pandas)

- **Numpy is a python module that adds support for multi-dimensional arrays, and their manipulation**
  - https://numpy.org/doc/



https://towardsdatascience.com/numpy-array-cookbook-generating-and-manipulating-arrays-in-python-2195c3988b09



```
In [3]: arrA = np.array([[1,3],[5,7]])
        arrB = np.array([[10,9],[8,7]])

In [4]: arrA+arrB

Out[4]: array([[11, 12],
               [13, 14]])

In [5]: arrA*arrB

Out[5]: array([[10, 27],
               [40, 49]])
```

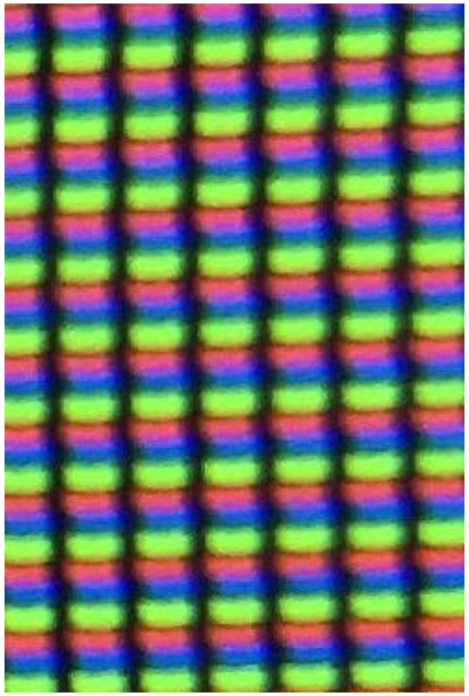
- **Pandas is essentially excel in python (but better).**
  - Two of the main pandas classes:
    - pandas.Series is a dictionary like class (similar to a single column in an excel sheet)
    - pandas.DataFrame is a like a 2D dictionary, akin to a excel sheet
  - Pandas allows for easy loading and exporting to .csv, .xlsx, etc
  - Pandas is (currently) limited to 2D

Numpy and images

- Numpy is an obvious choice to handle images in python
- Represent images as an array of values
- As many dimensions as you need!

Excel Image example

- For display on a screen, the 3 color channels are often stacked to create a ‘single’ color pixel.
- E.g. a 1080x1920 monitor has 2073600 pixels, each with an R,G and B channel



- Lets take a look at the demonstration to cement how images are saved on disk.

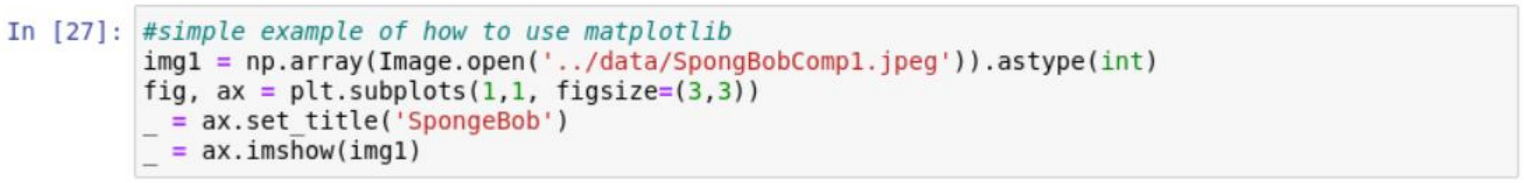
Types of images and file formats

- **The number that make up an image are of a specific type**
  - Commonly:
    - 0-255 as an integer (8 bit color)
    - 0-1 as a float (single or double precision)

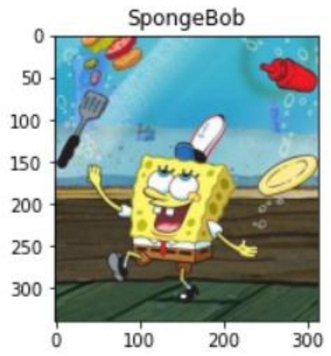
- **Images come in many file formats:**
  - Lossy (compression is used, and some information is lost)
    - JPEG/JPG
  - Lossless (compression may be used, but no information is lost)
    - TIFF
    - GIF
    - PNG
  - Many more!
    - https://en.wikipedia.org/wiki/Image_file_format#:~:text=Image%20file%20compression,-There%20are%20two&text=Lossless%20compression%20should%20be%20used,is%20not%20a%20perfect%20copy.

Matplotlib

- Matplotlib is a commonly used python package used for plotting.
- Matplotlib has a commonly used class API that defines plotting axes when then are modified using the axis methods to display the desired plot.
- The Axis class has a useful method called “.imshow” that can be used to visualize an array.



```
In [27]: #simple example of how to use matplotlib
         img1 = np.array(Image.open('../data/SpongBobComp1.jpeg')).astype(int)
         fig, ax = plt.subplots(1,1, figsize=(3,3))
         _ = ax.set_title('SpongeBob')
         _ = ax.imshow(img1)
```



- plt.subplots() returns N axis objects that is the same shape as the values given, and truncated for 1s.
  - E.g
    - fig, axs = plt.subplots(1,1). axs is just a single axis object
    - fig, axs = plt.subplots(2,1). axs is a numpy array of axis of shape (2,)
    - fig, axs = plt.subplots(3,3). axs is a numpy array of axis of shape (3,3)

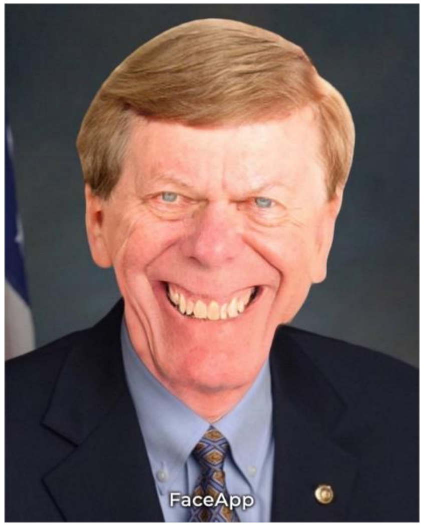
Image distributions

- **When you have a collection of images, creating summary images can be a useful tool to better understand your data.**
  - What your data looks like, differences between groups of similar images

Image distributions

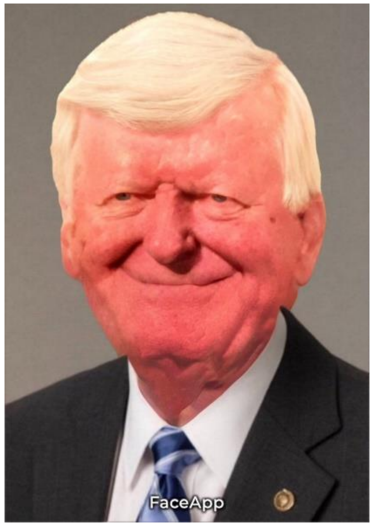
- **When you have a collection of images, creating summary images can be a useful tool to better understand your data.**
  - What your data looks like, differences between groups of similar images

Average democratic senator



https://twitter.com/_sn_n

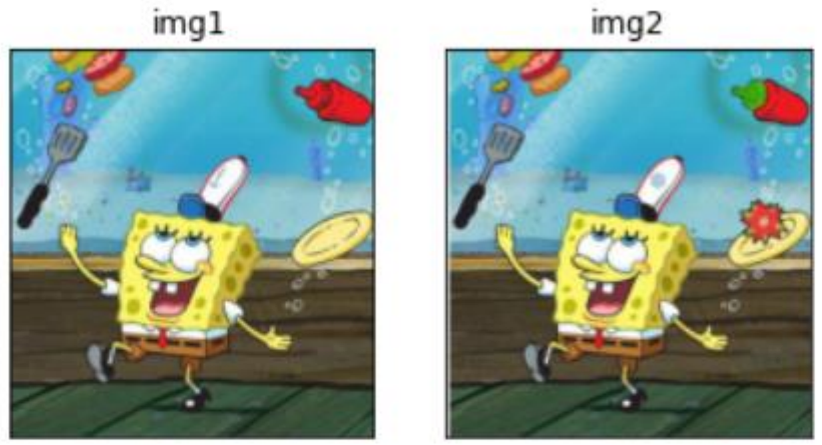
Average republican senator



- For example, 'mean' images of the US senators split by party affiliation.

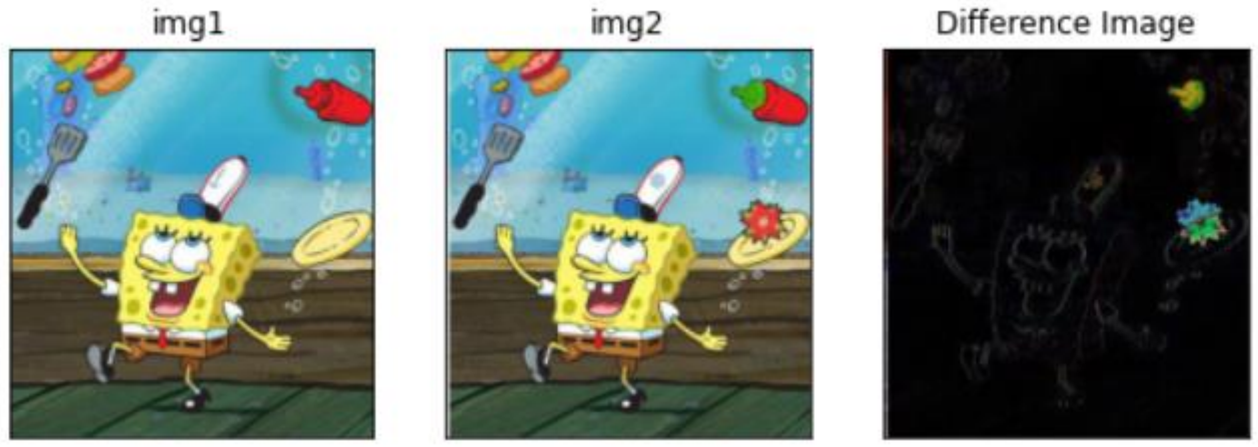
Difference images

- **Difference images can be used to compare 2 images.**
  - What, and how many differences are in this image?!



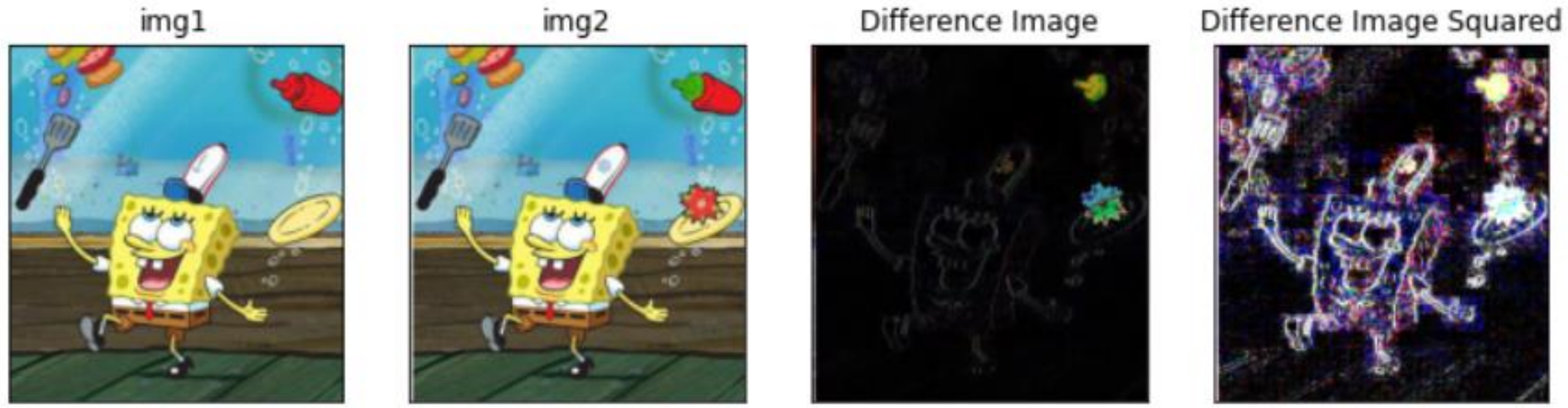
Difference images

- **Difference images can be used to compare 2 images.**
  - What, and how many differences are in this image?!
- **Element wise subtraction is a simple method to create a difference image.**
  - $abs(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})$



Difference images

- **Difference images can be used to compare 2 images.**
  - What, and how many differences are in this image?!
- **Element wise subtraction is a simple method to create a difference image.**
  - $abs(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})$
- **To highlight smaller differences, the square may be used.**
  - $(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^2$



Ideas on why the squared image shows SpongeBob?

Hint: the image is saved as a .jpg

Mean squared error (Pixelwise loss)

- **MSE is frequently used pixelwise loss function to compare 2 images.**

$$L = \frac{1}{n}\sum_{i=0}^{n}(y_i - \widehat{y}_i)^2$$

y_i=

1	0.4
0.3	0.8

$\widehat{y}_i$=

0.5	0.8
0.3	0.5

Mean squared error (Pixelwise loss)

- **MSE is frequently used pixelwise loss function to compare 2 images.**

$$L = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

y_i=

1	0.4
0.3	0.8

$\hat{y}_i$=

0.5	0.8
0.3	0.5

$y_i - \hat{y}_i =$

0.5	-0.4
0.0	0.3

Mean squared error (Pixelwise loss)

- **MSE is frequently used pixelwise loss function to compare 2 images.**

$$L = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

$y_i =$

1	0.4
0.3	0.8

$\hat{y}_i =$

0.5	0.8
0.3	0.5

$y_i - \hat{y}_i =$

0.5	-0.4
0.0	0.3

$(y_i - \hat{y}_i)^{\boxed{2}} =$

0.25	0.16
0.0	0.09

The square achieves 2 things.

1. Ensures the loss is commutative
   - i.e. MSE(img1,img2) = MSE(img2,img1)
2. Penalizes the values more the larger the difference
   - Often useful when used as a loss function

Mean squared error (Pixelwise loss)

- **MSE is frequently used pixelwise loss function to compare 2 images.**

$$\boldsymbol{L = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}$$

$\boldsymbol{y_i} =$

1	0.4
0.3	0.8

$\boldsymbol{\hat{y}_i} =$

0.5	0.8
0.3	0.5

$\boldsymbol{y_i - \hat{y}_i} =$

0.5	-0.4
0.0	0.3

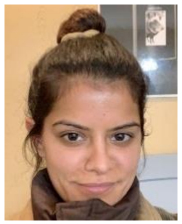
$\boldsymbol{(y_i - \hat{y}_i)^{\textcircled{2}}} =$

0.25	0.16
0.0	0.09

The square achieves 2 things.
1. Ensures the loss is commutative
   - i.e. MSE(img1,img2) = MSE(img2,img1)
2. Penalizes the values more the larger the difference
   - Often useful when used as a loss function

$$\boldsymbol{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2} = \frac{.25+.16+0+.09}{4} = \mathbf{0.125}$$

Acknowledgements

Dr. Albert Montillo	Dr. Son Nguyen	Cooper Mellema	Kevin Nguyen	Mahak Virlley	Aixa Hernandez	Atef Ali	**Krishna Chitta**
							

- **Everyone in the BioHPC team**
- **All of you for listening today**