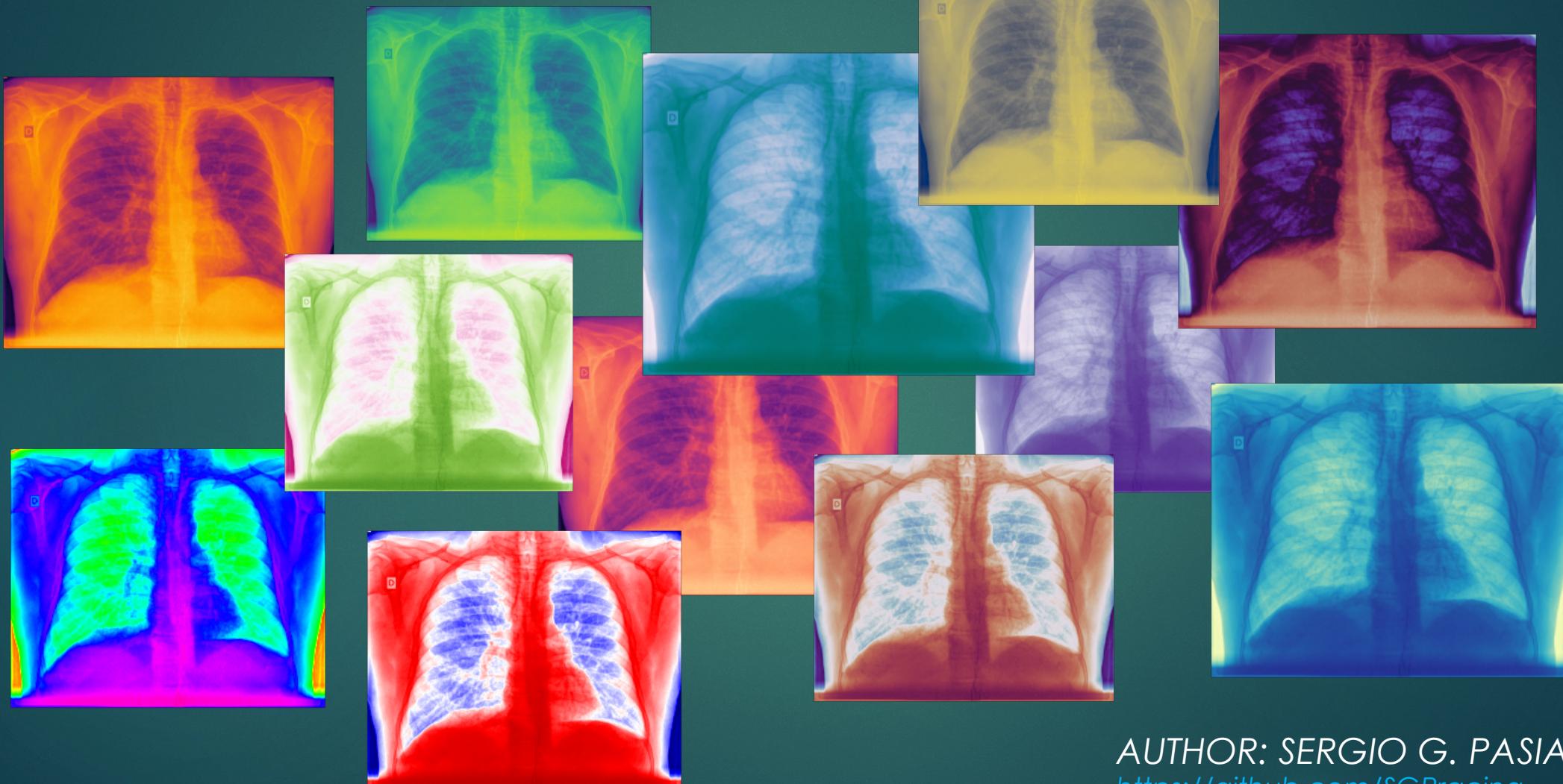


PROJECT X-RAY

IRONHACK: DATA ANALYTICS



AUTHOR: SERGIO G. PASIAN
<https://github.com/SGPracing>



Link: <https://www.kaggle.com/competitions/unifesp-x-ray-body-part-classifier>

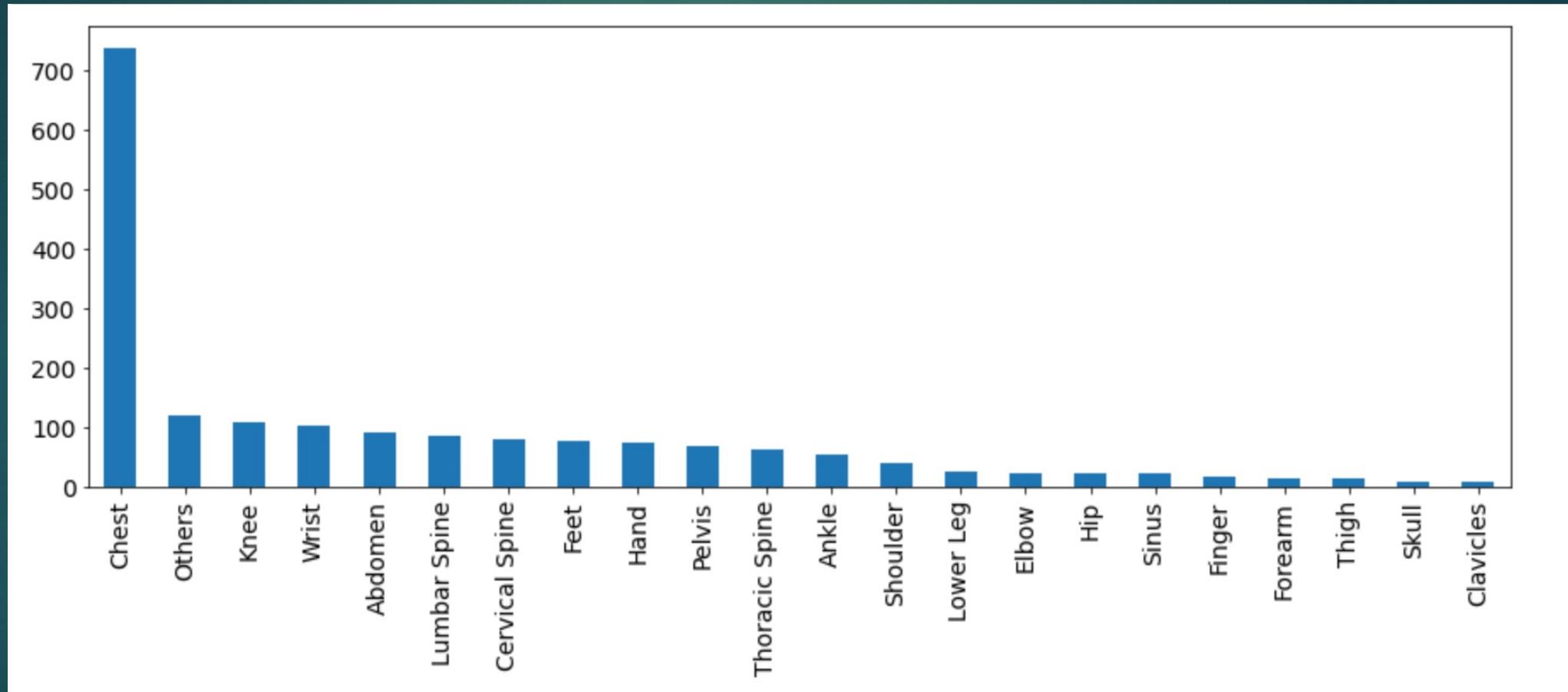
► **Objective:**

- to predict the body part from a given X-ray

► **Data:**

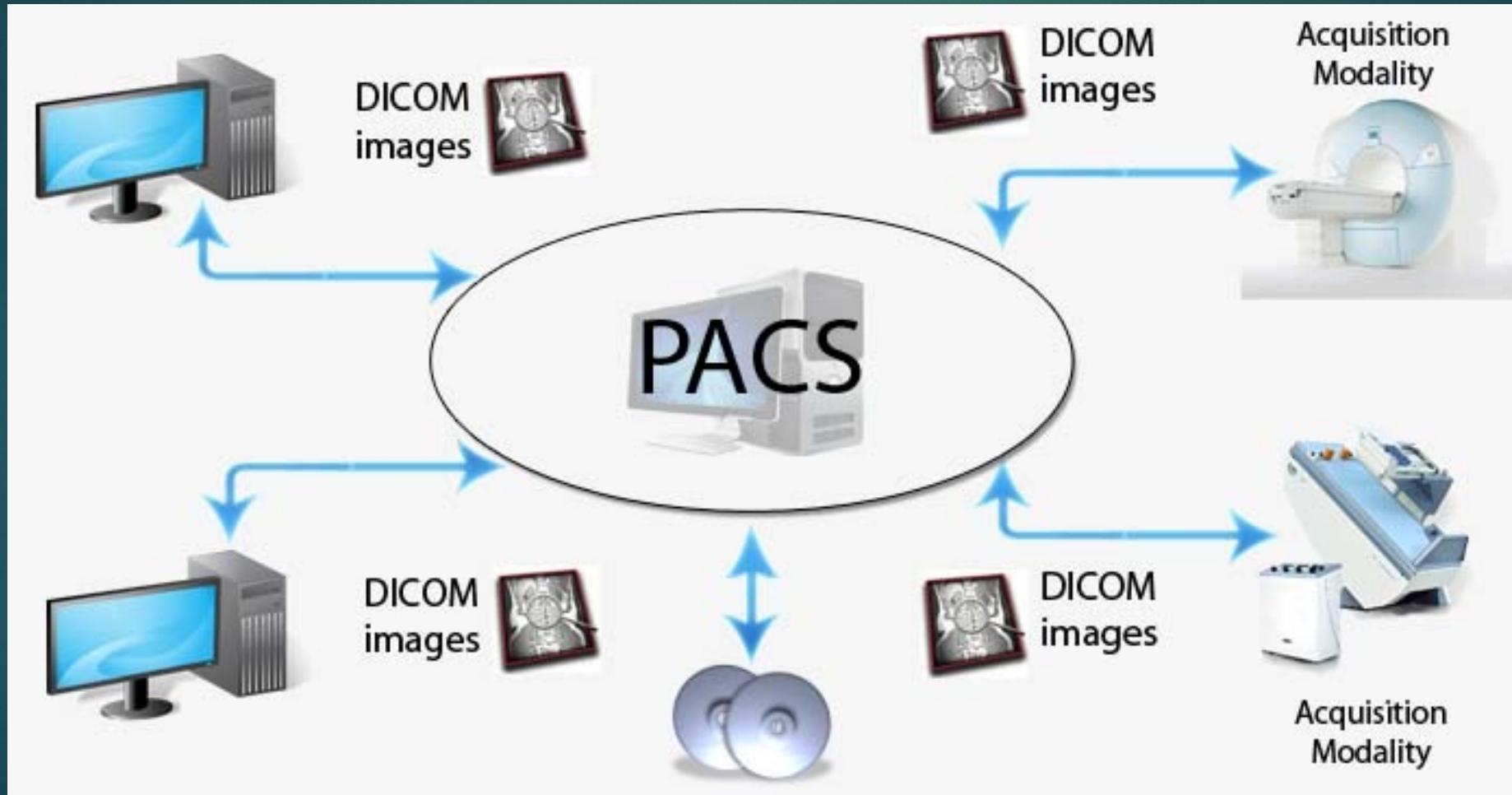
- Sets of images in DICOM format;
- “train.csv” file, with the following information :
 - ‘SOPInstanceUID’: image identification (from metadata)
 - ‘Target’: label (body part): **21 labels**

Labels



Why solving this problem matters?

Picture Archiving and Communication System



Why solving this problem matters?

Proc Description	Study Date Time	Image Count	Modality	Status
		11	CR	
JOELHO: AP. + LAT...	11/05/2021 19:02:59	11	CR	Complete
PERNA	11/05/2021 18:53:00	0	CR	Scheduled
QUADRIL - ARTICU...	11/05/2021 18:53:00	0	CR	Scheduled
RAIO X DE TORN...	11/05/2021 18:53:00	0	CR	Scheduled
COXA	11/05/2021 18:52:00	0	CR	Scheduled
BACIA	11/05/2021 18:52:00	0	CR	Scheduled

- ▶ Several X-rays for the same patient are entered in the system by the receptionist
- ▶ Technician opens the knee study and execute all the examens under the same description
- ▶ All images (exams) from different body parts are stored within the "knee" study
- ▶ It may cause problems for Radiologist to make the report
- ▶ Impairs comparison with older exams
- ▶ The empty (no images) studies remain with an open status (Schedule), which trigger further problems

It should be a very simple task...

Chest



Abdominal



Knee



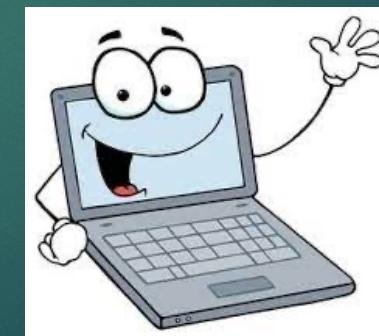
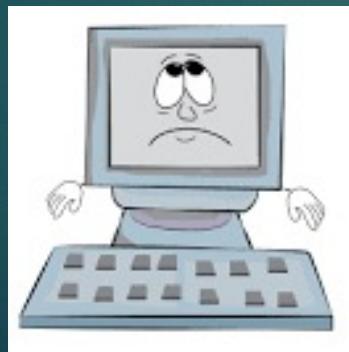
Wrist



... but it is not that simple...



```
array([[140, 144, 143, ..., 51, 50, 50],  
       [140, 138, 138, ..., 41, 40, 39],  
       [135, 138, 141, ..., 36, 35, 38],  
       ...,  
       [ 49,  50,  48, ..., 34, 34, 36],  
       [ 51,  50,  49, ..., 38, 34, 33],  
       [ 53,  51,  50, ..., 35, 35, 36]]),
```



... and several challenges were in the way

- ▶ Heavy files (train image files approx. 22GB):
 - ▶ How to store the transformed images (numpy arrays)
- ▶ Load and read the image inside the interface:
 - ▶ type of image files (DICOM)
 - ▶ decoder (pydicom)
- ▶ Image contrast display
- ▶ Matching image ID, label from dataframe with image itself
- ▶ Image sizing
- ▶ Transformation and classification algorithms for images

Image Loading and Visualization

train	Apr 20, 2022 at 8:32 AM	--	Folder
train	Apr 21, 2022 at 4:57 PM	--	Folder
1	Apr 21, 2022 at 4:57 PM	--	Folder
1.2.826.0.1.368004...5855773961083133	Apr 21, 2022 at 4:57 PM	--	Folder
1.2.826.0.1.36800...237519434199794	Apr 20, 2022 at 8:23 AM	--	Folder
1.2.826.0.1.3680...75343109-c.dcm	Mar 30, 2022 at 3:17 PM	17.4 MB	Gen110
2	Today at 1:22 PM	--	Folder
1.2.826.0.1.368004...3033510994286311	Today at 1:22 PM	--	Folder
1.2.826.0.1.36800...481308217430401	Apr 20, 2022 at 8:24 AM	--	Folder
1.2.826.0.1.3680...76800615-c.dcm	Mar 30, 2022 at 3:30 PM	17 KB	Gen110
1.2.826.0.1.36800...227948776841518	Apr 20, 2022 at 8:24 AM	--	Folder
1.2.826.0.1.3680...95756627-c.dcm	Mar 30, 2022 at 3:30 PM	12 KB	Gen110
3	Apr 20, 2022 at 8:24 AM	--	Folder
4	Apr 20, 2022 at 8:24 AM	--	Folder
10	Apr 20, 2022 at 8:23 AM	--	Folder

```
1 from pathlib import Path  
2
```

executed in 8ms, finished 13:07:41 2022-05-01

```
1 images_train = Path().cwd().glob("**/train/**/*.dcm")  
2 img_train_path_lst = [path for path in images_train]  
3
```

executed in 5.39s, finished 13:07:47 2022-05-01

Image Loading & Visualization

```
1 data = read_xray(img_train_lst[100])
2 plt.figure(figsize=(8, 8))
3 plt.imshow(data, 'gray');
4
```

executed in 5.49s, finished 13:10:59 2022-05-01

```
1 import numpy as np
2 import pydicom
3 from pydicom.pixel_data_handlers.util import apply_voi_lut
4
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7
8
9 def read_xray(path, voi_lut = True, fix_monochrome = True):
10     dicom = pydicom.read_file(path)
11
12     # VOI LUT (if available by DICOM device) is used to transform raw DICOM data to "human-friendly" view
13     if voi_lut:
14         data = apply_voi_lut(dicom.pixel_array, dicom)
15     else:
16         data = dicom.pixel_array
17
18     # depending on this value, X-ray may look inverted - fix that:
19     if fix_monochrome and dicom.PhotometricInterpretation == "MONOCHROME1":
20         data = npamax(data) - data
21
22     data = data - npmin(data)
23     data = data / npmax(data)
24     data = (data * 255).astype(np.uint8)
25
26     return data
27
```

executed in 18ms, finished 13:10:27 2022-05-01

```
1 data
executed in 10ms, finished 13:11:03 2022-05-01
array([[ 41,  41,  41, ..., 137, 144, 141],
       [ 41,  41,  41, ..., 136, 137, 138],
       [ 41,  41,  41, ..., 139, 141, 141],
       ...,
       [ 33,  33,  33, ..., 199, 201, 201],
       [ 33,  33,  33, ..., 201, 205, 205],
       [ 33,  33,  33, ..., 201, 208, 204]], dtype=uint8)
```

```
1 data.shape
executed in 8ms, finished 13:11:04 2022-05-01
(4240, 3480)
```

Vectorizing

Resizing (1000 x 1000)

Matrix to Vector

```
3 from skimage.transform import resize  
1 vector_sample_lst = []  
2 for i in range(len(img_sample_lst)):  
3     raw_img = read_xray(img_sample_lst[i])  
4     image = resize(raw_img, (1000, 1000), anti_aliasing=True)  
5     vector_data = np.ravel(image)  
6     vector_sample_lst.append(vector_data)  
7 vector_sample_array = np.array(vector_sample_lst)  
8 vector_sample_array.shape  
9
```

executed in 21m 18s, finished 17:07:06 2022-04-30

(394, 1000000)

images
(rows)

columns

Transform list into a numpy array

Saving the variables (Pickle file)

```
1 import pickle  
2
```

execution queued 13:12:30 2022-05-01

```
1 with open('vector_train_lst.pkl', 'wb') as f:  
2     pickle.dump(vector_train_lst, f)
```

execution queued 13:12:31 2022-05-01

Building Final Dataframe

The diagram illustrates the process of building a final DataFrame. It starts with raw DICOM data on the left, which is processed to extract specific fields (SOP Instance UID, Target, Path, Anatomy) and then aggregated into a final DataFrame on the right.

Raw DICOM Data:

```
1 image = img_sample_lst[1]
2 ds = pydicom.read_file(image)
3
executed in 36ms, finished 15:14:10 2022-04-30
```

```
1 ds
executed in 25ms, finished 15:14:13 2022-04-30
```

```
Dataset.file_meta
(0002, 0000) File Meta Information Group Length UL: 244
(0002, 0001) File Meta Information Version OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID UI: Computed Radiography Image Storage
(0002, 0003) Media Storage SOP Instance UID UI: 1.2.826.0.1.3680043.8.498.10242799675195671634897807131985
48
(0002, 0010) Transfer Syntax UID UI: JPEG 2000 Image Compression (Lossless Only)
(0002, 0012) Implementation Class UID UI: 1.2.826.0.1.3680043.2.1143.107.104.103.115.3.0.10
(0002, 0013) Implementation Version Name SH: 'GDCM 3.0.10'
(0002, 0016) Source Application Entity Title AE: 'gdcmconv'

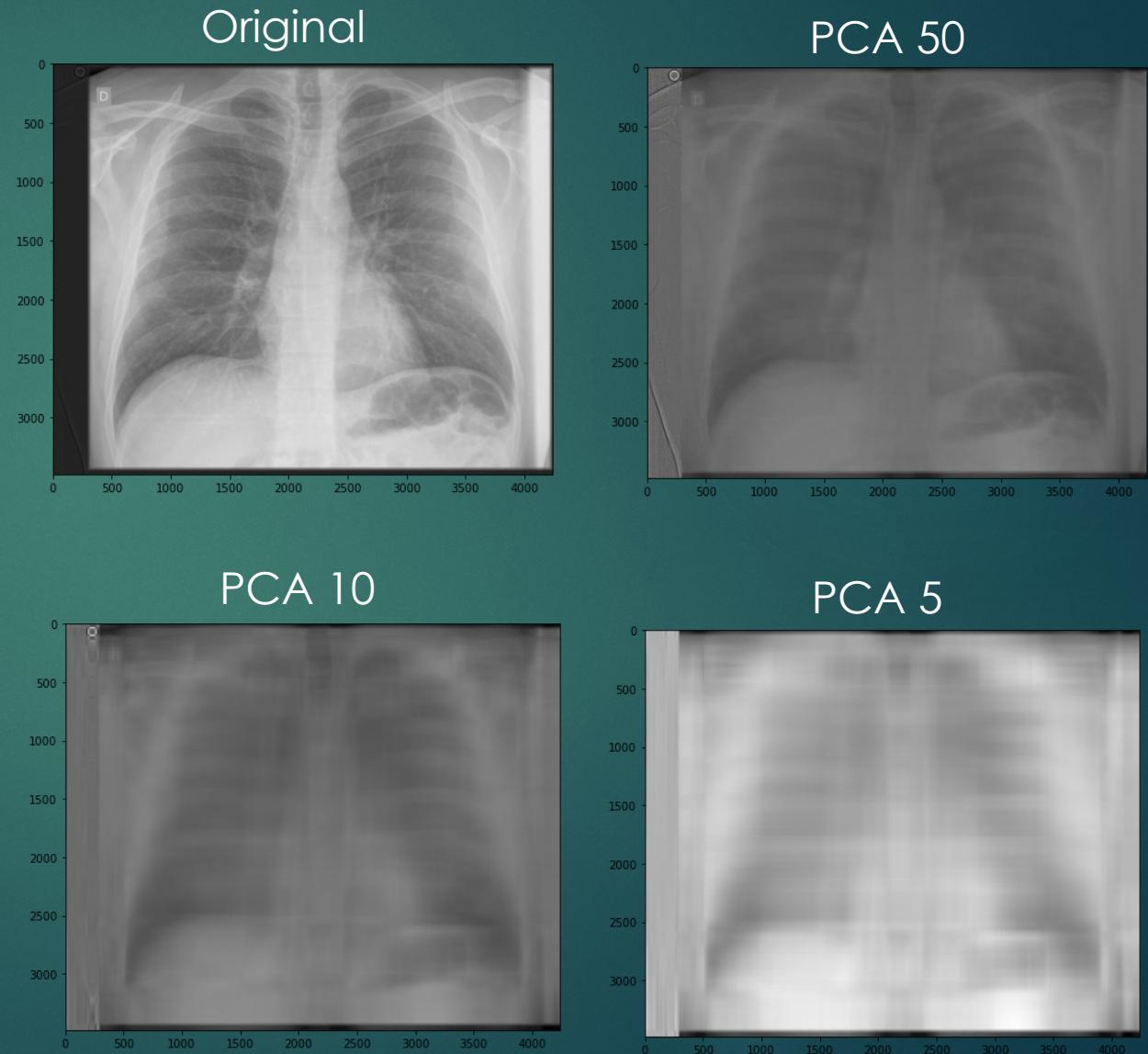
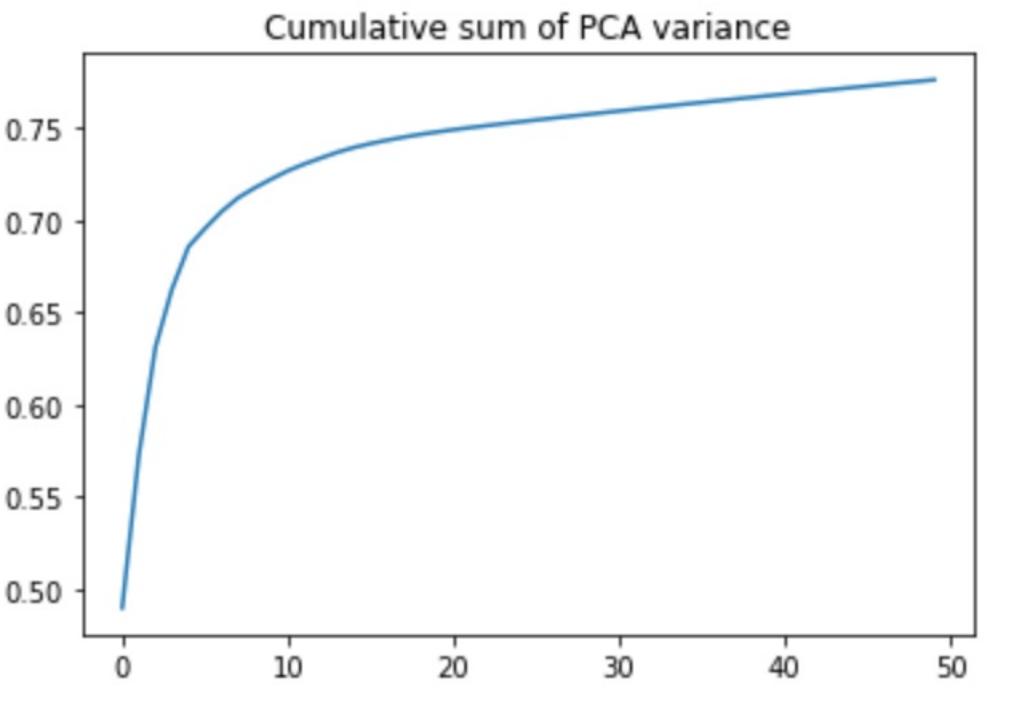
(0008, 0008) Image Type CS: ['ORIGINAL', 'PRIMARY']
(0008, 0012) Instance Creation Date DA: ''
(0008, 0013) Instance Creation Time TM: ''
(0008, 0016) SOP Class UID UI: Computed Radiography Image Storage
(0008, 0018) SOP Instance UID UI: 1.2.826.0.1.3680043.8.498.102427996751956716348978071319850004
48
(0008, 0020) Study Date DA: ''
(0008, 0021) Series Date DA: ''
(0008, 0023) Content Date DA: ''
(0008, 0030) Study Time TM: '111144'
(0008, 0033) Content Time TM: ''
(0008, 0050) Accession Number SH: '44'
(0008, 0060) Modality CS: 'CR'
(0008, 0070) Manufacturer LO: ''
(0008, 0090) Referring Physician's Name PN: ''
(0008, 0100) Code Value SH: ''
(0008, 1030) Study Description LO: 'XRAY'
```

```
1 elem = ds[0x0008, 0x0018]
2 elem.value
3
executed in 14ms, finished 15:14:28 2022-04-30
'1.2.826.0.1.3680043.8.498.10242799675195671634897807131985000448'
```

Final DataFrame:

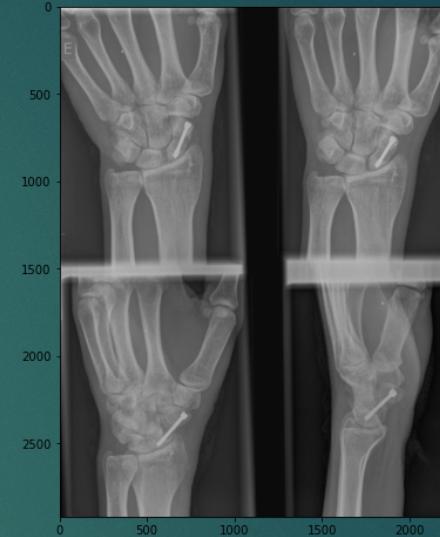
	SOPInstanceUID	Target	Path	Anatomy
0	1.2.826.0.1.3680043.8.498.10025629581362719970...	0	/Users/sgp/Documents/Ironhack/UNIFESP/data/trai...	Abdomen
1	1.2.826.0.1.3680043.8.498.10065930002825553435...	13	/Users/sgp/Documents/Ironhack/UNIFESP/data/trai...	Lumbar Spine
2	1.2.826.0.1.3680043.8.498.10107388868056003719...	0	/Users/sgp/Documents/Ironhack/UNIFESP/data/trai...	Abdomen
3	1.2.826.0.1.3680043.8.498.10166555243811009418...	21	/Users/sgp/Documents/Ironhack/UNIFESP/data/trai...	Wrist
4	1.2.826.0.1.3680043.8.498.10181745787571911943...	0	/Users/sgp/Documents/Ironhack/UNIFESP/data/trai...	Abdomen

Principle Component Analysis (PCA)

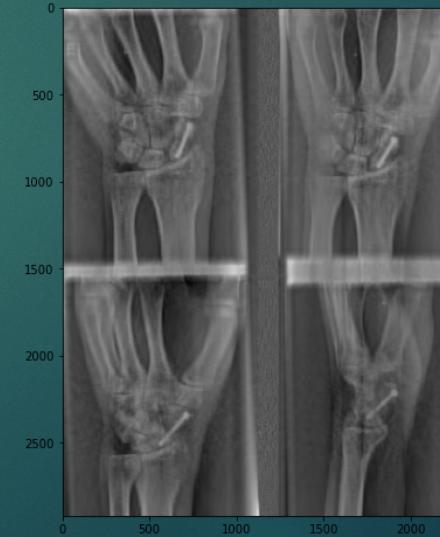
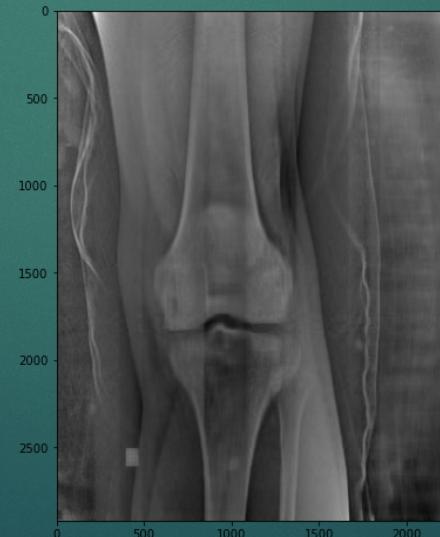
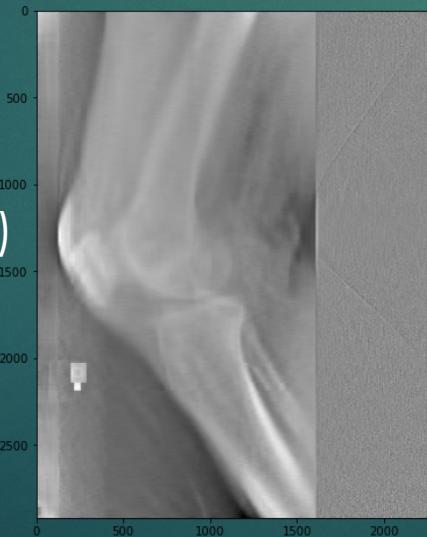


Principle Component Analysis (PCA)

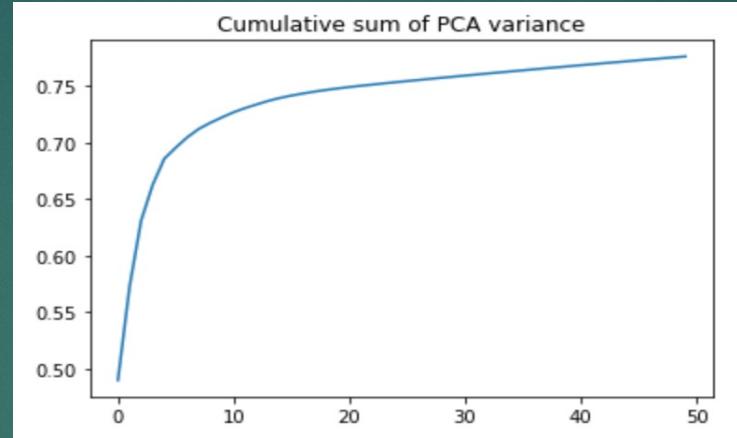
Original images
(2920 x 2320)



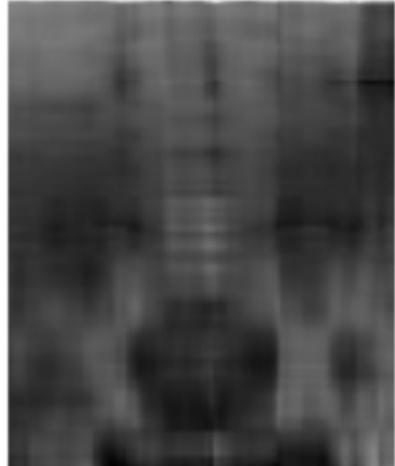
PCA (n_components=50)
(2920 x 50)



Principle Component Analysis (PCA)



PCA n=5



PCA n=30



PCA n=60



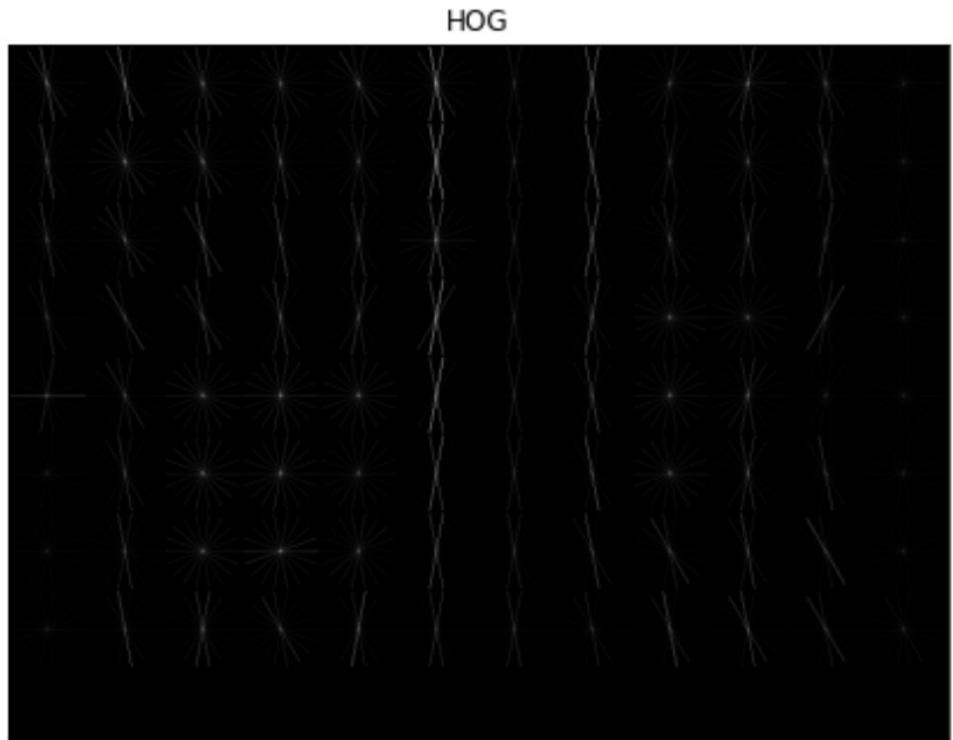
PCA n=100



Original



Histogram of Oriented Gradients (HOG)



Number of pixels: 442929

Number of HOG features: 2772

np.mean(xray_hog_img): 0.0003925043990108372

Singular Value Decomposition (SVD)

```
3 | for r in (5, 20, 100):  
4 |     Xapprox = u[:, :r] @ np.diag(s)[0:r, :r] @ v[:r, :]
```



Models (Classifiers)

► Stochastic Gradient Descent (SGD)

```
1 sgd_clf = SGDClassifier(random_state=42, max_iter=1000, tol=1e-3)
2 sgd_clf.fit(pca_data_train, y_train)
```

executed in 65ms, finished 22:38:30 2022-05-05

```
SGDClassifier(random_state=42)
```

► CatBoost

```
1 cat_fit = cat.CatBoostClassifier(
2     iterations=20000, depth=8, auto_class_weights="Balanced", od_type="Iter", od_wait=500)
3 cat_fit.fit(pca_data_train, y_train,
4             eval_set=(pca_data_test, y_test))
5
```

executed in 47.9s, finished 19:13:01 2022-05-02

Models

► Stochastic Gradient Descent (SGD)

Sample: 2 labels (knee and wrist)

```
1 sgdc_pred = sgd_clf.predict(pca_data_test)
2 print(f"F1-Score PCA{pca_data_test.shape[1]}: {f1_score(y_test, sgdc_pred, average='weighted')}")
3
executed in 12ms, finished 13:26:12 2022-05-02
F1-Score PCA50: 0.8129334965320277
```

Sample: 6 labels (knee, wrist, abdomen, l. spine, c. spine, feet)

```
1 sgdc_pred = sgd_clf.predict(pca_data_test)
2 print(f"F1-Score PCA{pca_data_test.shape[1]}: {f1_score(y_test, sgdc_pred, average='weighted')}")
3
executed in 15ms, finished 23:37:22 2022-05-02
F1-Score PCA50: 0.3555522420184074
```

Models

► CatBoost

Sample: 2 labels (knee and wrist)

```
1 cat_pred = cat_fit.predict(pca_data_test)
2 print(f"F1-Score PCA{pca_data_test.shape[1]}: {f1_score(y_test, cat_pred, average='weighted')}")
3
executed in 11ms, finished 19:17:43 2022-05-02
F1-Score PCA50: 0.8121334681496463
```

Sample: 6 labels (knee, wrist, abdomen, l. spine, c. spine, feet)

```
1 # 6 categories
2 cat_pred = cat_fit.predict(pca_data_test)
3 print(f"F1-Score PCA{pca_data_test.shape[1]}: {f1_score(y_test, cat_pred, average='weighted')}")
executed in 90ms, finished 00:08:25 2022-05-03
F1-Score PCA50: 0.7978020272243637
```

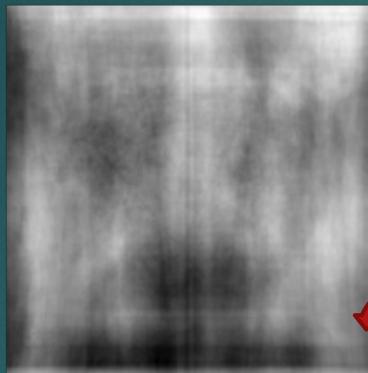
Visualizing the Results

From the CatBoost Model (6 categories)
F1score = 0.79

Original Image



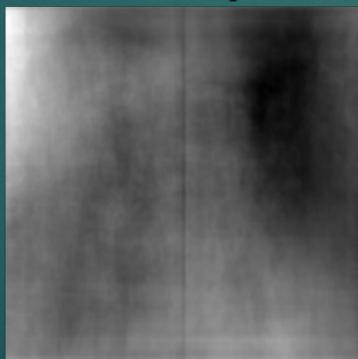
Transformed Image (PCA)



Original Image



Transformed Image (PCA)

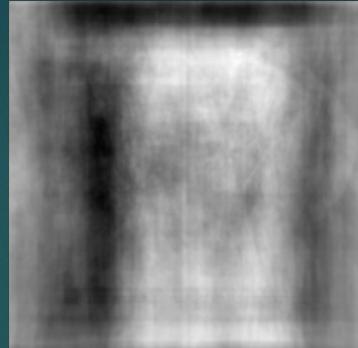


index	Real_Anatomy	Pred_Anatomy
0	270	Lumbar Spine
1	90	Wrist
2	133	Wrist
3	221	Wrist
4	224	Wrist
5	286	Abdomen
6	159	Lumbar Spine
7	47	Lumbar Spine
8	298	Abdomen
9	406	Cervical Spine
10	437	Wrist
11	31	Wrist
12	283	Lumbar Spine
13	267	Wrist
14	326	Wrist
15	334	Lumbar Spine

Original Image



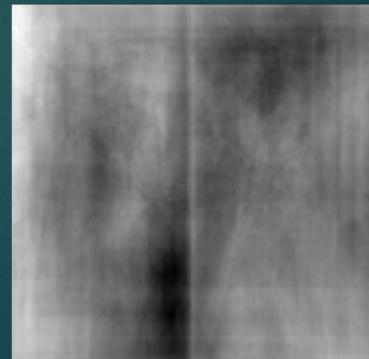
Transformed Image (PCA)



Original Image



Transformed Image (PCA)



Current State of the Project

- ▶ Dataframe:
 - ▶ Relevant information aggregated in one single table;
 - ▶ Image data aligned with dataframe values
- ▶ Images:
 - ▶ DICOM: loading and visualization
 - ▶ Resizing for standardization (1000 x 1000)
 - ▶ Conversion to a vector data (arrays)
 - ▶ Storage of image data (arrays) - Pickle
- ▶ Transformation/Compression algorithms:
 - ▶ PCA
 - ▶ HOG
 - ▶ SVD
- ▶ Classification Models:
 - ▶ CatBoost
 - ▶ SGD

Next Steps

- ▶ Look deeper into imaging standardization:
 - ▶ Resizing
 - ▶ File conversion
 - ▶ Contrast
- ▶ Test different transformation/compression algorithms and optimize parameters
- ▶ Classification models:
 - ▶ Parameters optimization
 - ▶ Test New models
- ▶ Work on a pipeline to automation of the entire process

