機械学習のための
データ前処理の基本と実践法
10

医用画像データの扱い

徳島大学
デザイン型AI教育研究センター
鳥井 浩平

今回の内容

1. DICOM画像
2. Pydicom
3. DICOM画像の読み書きと表示

参考書について

- 今回から『Pythonによる医用画像処理入門』に従って解説する
- 今回は参考書に付属するデータセットを使って解説する
- 規約によりデータセットの再配布はしない
- 実装はいつも通り紹介する
- 次回の肺炎予測はKaggleのデータセットを用いる

www.amazon.co.jp/dp/4274225461

前回の復習

畳み込みニューラルネットワークの実装

理想的なデータとは

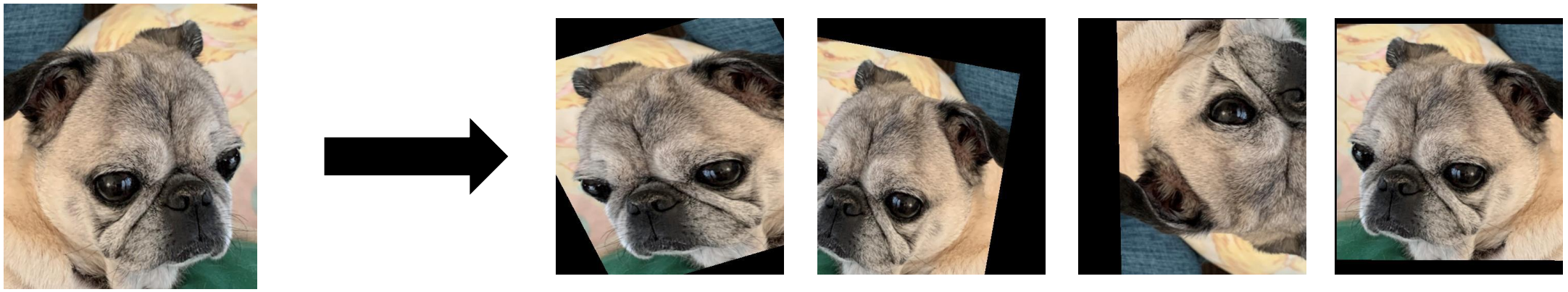
- 機械学習はとにかくデータの質と量が大事
- 大原則　Garbage In, Garbage Out　「ゴミを入れるとゴミが出る」
- ロバストなモデルには多様なデータの学習が不可欠
  ロバスト：未知のデータに対しても頑健で精度が高いこと
- できるだけ学習データのパターンを増やしたい…

【データの増やし方】

- 素直に新しいデータを収集する
- 手元のデータを加工して疑似的にデータを増やす（**データ拡張**）

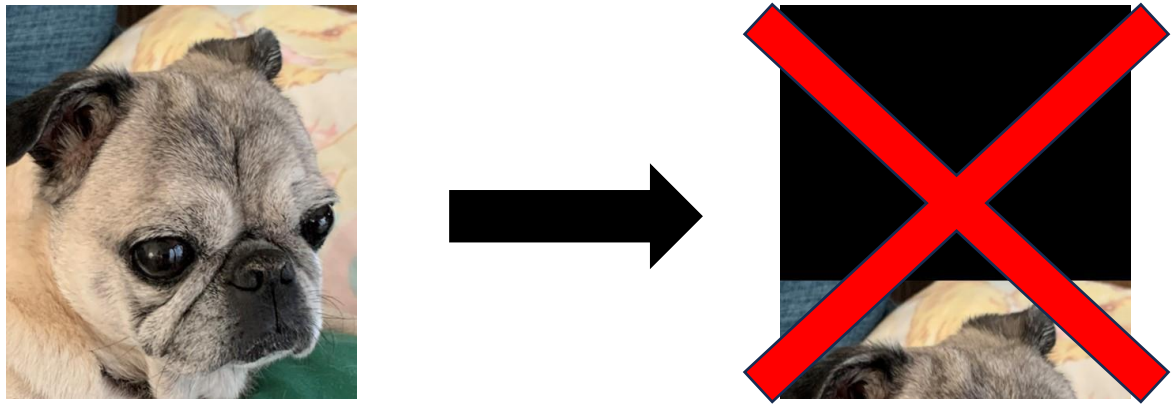
データ拡張

- Data Augmentation
- データを加工して疑似的にデータを増やすこと
- 画像認識分野では特によく用いられる
- 学習データに対してランダムに適用することが多い



データ拡張の注意点

- 不必要なデータ拡張は避ける
- 目的に応じて適度なデータ拡張を心がける
  ＝何事もやりすぎは良くない



大事な特徴である
顔が見えない！

ファインチューニング

- 学習済みモデルを目的に合わせて再学習すること
- ファインチューニングによって効率的にモデルを構築できる

具体的なイメージ

- 未学習CNNのパラメータ初期値はデタラメな値
- 初期値がすでに学習された値であれば効率良く学習できそう...
- （他の仕事で得た経験を今回の仕事に活かせそう...）

ファインチューニングと転移学習の違い

ファインチューニング

出力層（分類器）以外のパラメータ（特徴抽出器）も含めて再学習する

転移学習

出力層（分類器）のみを再学習する

ファインチューニング実装に向けて

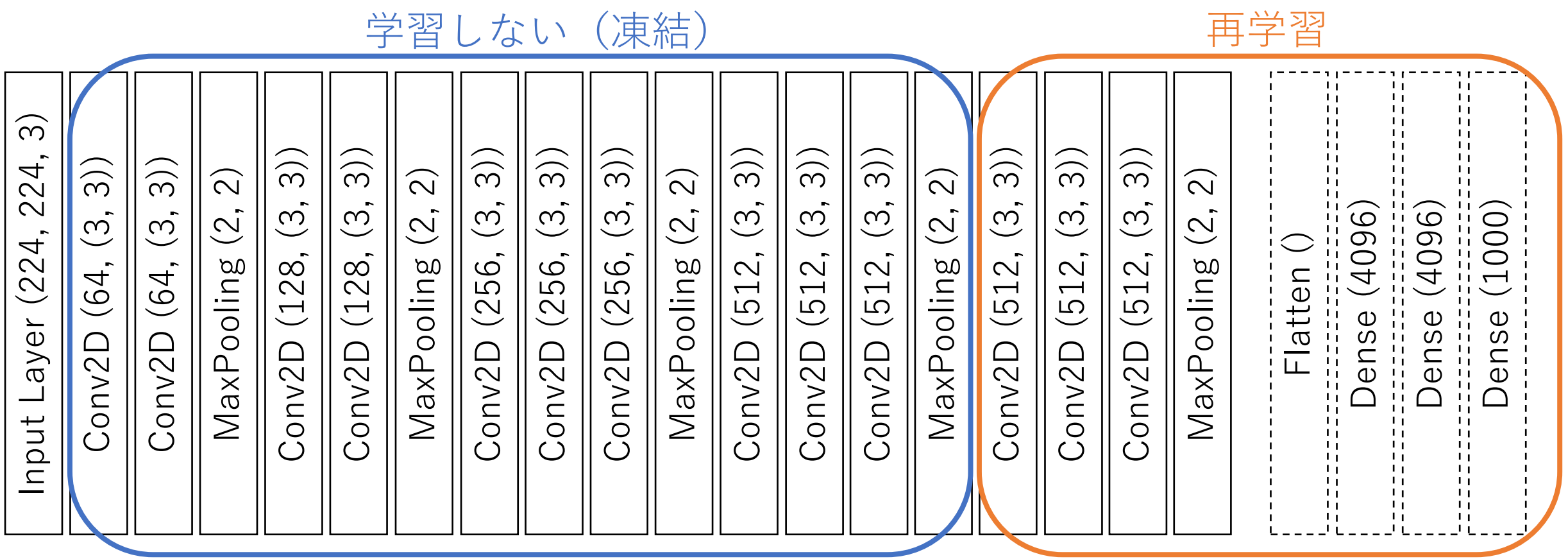
- 今回は馬とシマウマを分類するモデルの作成を目標とする
- データはCaltech 256 Image Datasetを用いる
- 学習済みモデルは**imagenet**を事前学習済みの**VGG16**を用いる
  imagenet: 計1400万枚以上の大規模画像データセット

Caltech 256 Image Dataset (https://www.kaggle.com/datasets/jessicali9530/caltech256)

- カテゴリ数257種、計30,607枚の画像データセット

VGG16の構造とファインチューニング

今回は、局所的特徴はそのままに、大局的特徴と出力のみを再学習する



1. DICOM画像

医用画像データの基本フォーマット

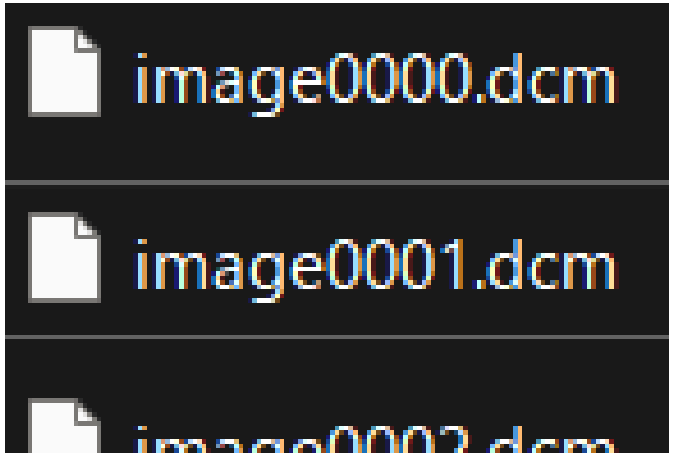
DICOM

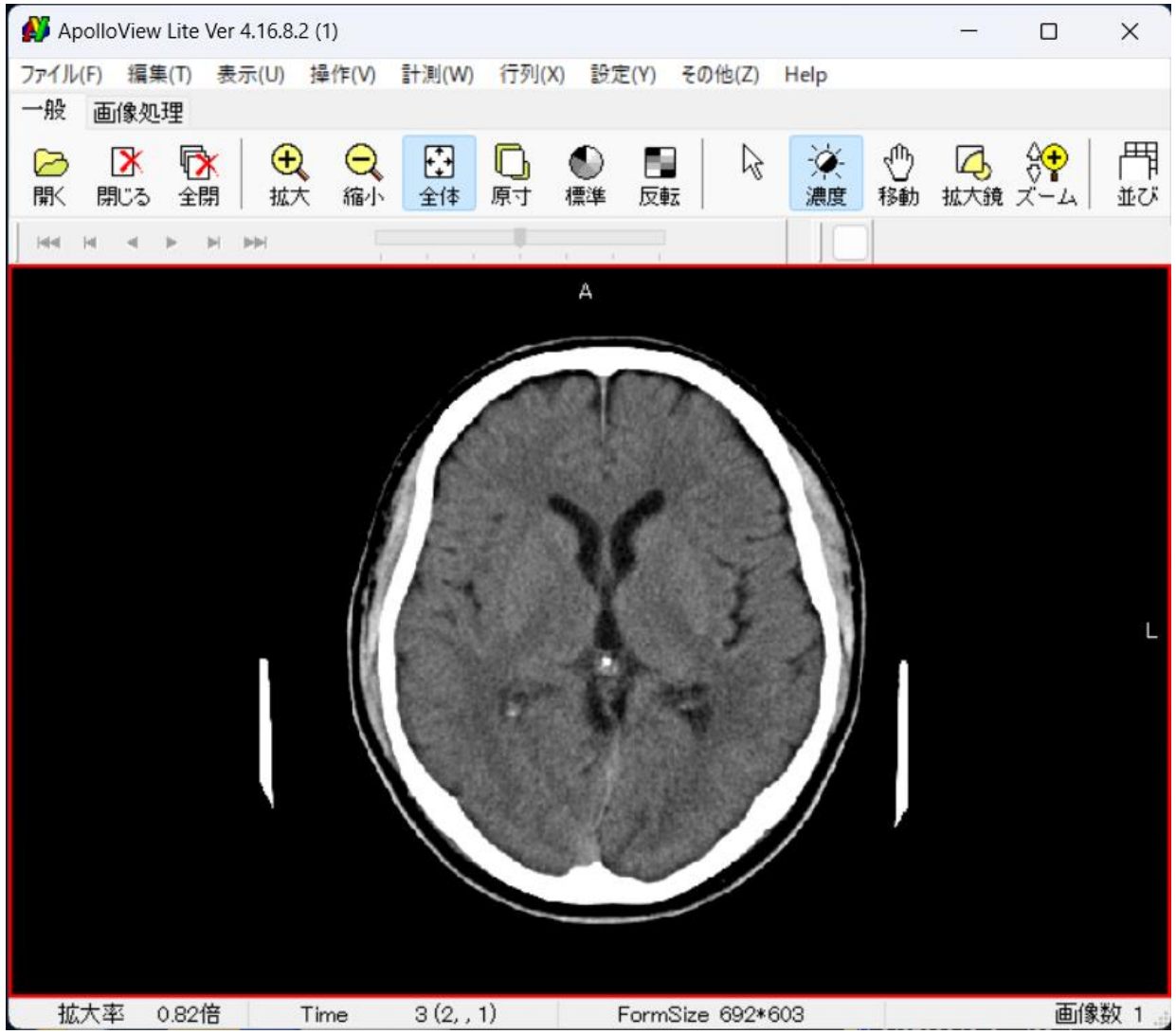
- Digital Imaging and Communications in Medicine → DICOM
- 医用画像（X線、超音波など）のフォーマットと通信に関する規格
  規格がバラバラだとデータ処理が本当に大変！
- 1993年に北米放射線学会と工業会により制定
- JIRA（一般社団法人日本画像医療システム工業会）のHPで公開中

https://www.jira-net.or.jp/dicom/

DICOM画像

- 基本的にDICOM画像の拡張子は**.dcm**（拡張子が無いときもある）
  一般的な画像のフォーマットはPNG（.png）やJPEG（.jpeg）など
- DOCM画像は一般的な画像ビューアでは読み込みや表示ができない
- 専用の画像ビューアが必要　→　**DICOMビューア（DICOM Viewer）**
- Pythonで読み込むためにも専用のライブラリが必要



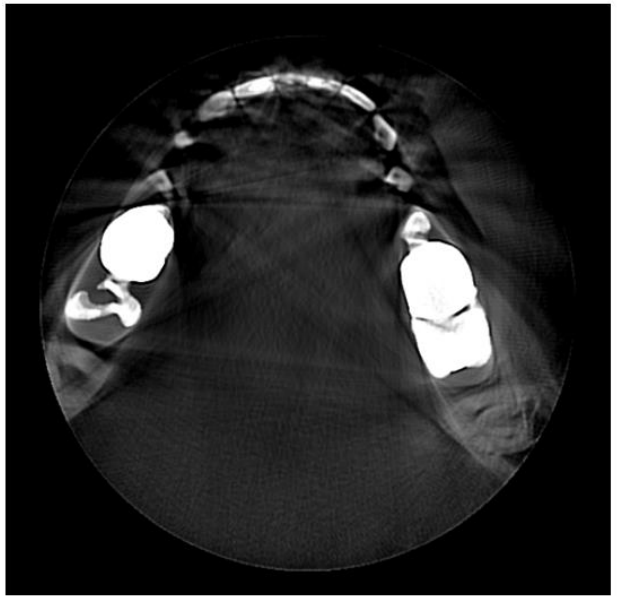


無料DICOMビューア「Apollo View Lite」

https://www.vector.co.jp/soft/dl/winnt/business/se333639.html

歯科パノラマX線

https://cdn.peraichi.com/userData/5fae282f-7dbc-45f7-b028-06410a0000ae/img/60b7209a478a4/original.png



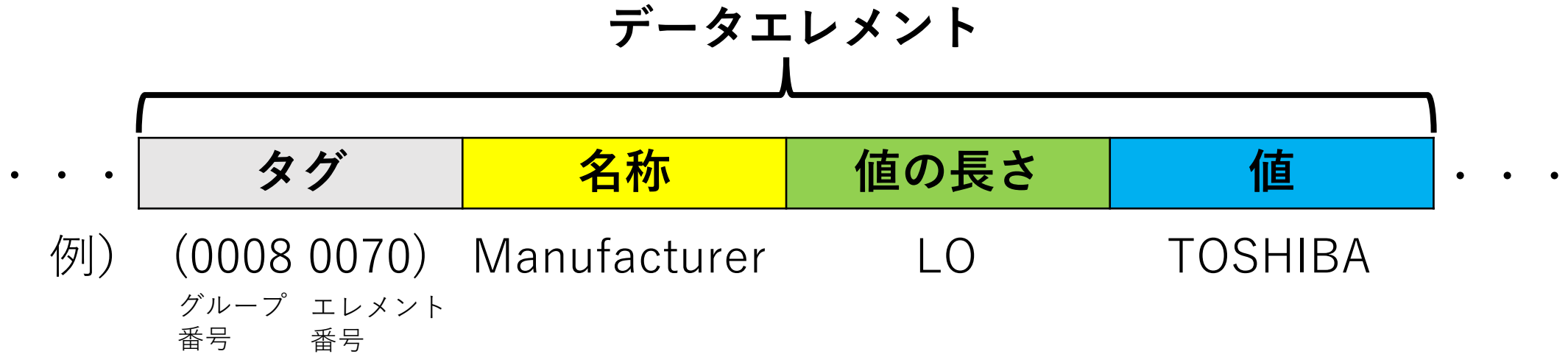
CT体軸断面

https://cdn.peraichi.com/userData/5fae282f-7dbc-45f7-b028-06410a0000ae/img/60bf05c21d84c/original.png

（補足）3次元データのDICOM画像

- CTなど3次元データのDICOM画像はどのようになっている？
- 体軸断のDICOM画像が集まって3次元データを構成している

DICOM画像の構造（１）



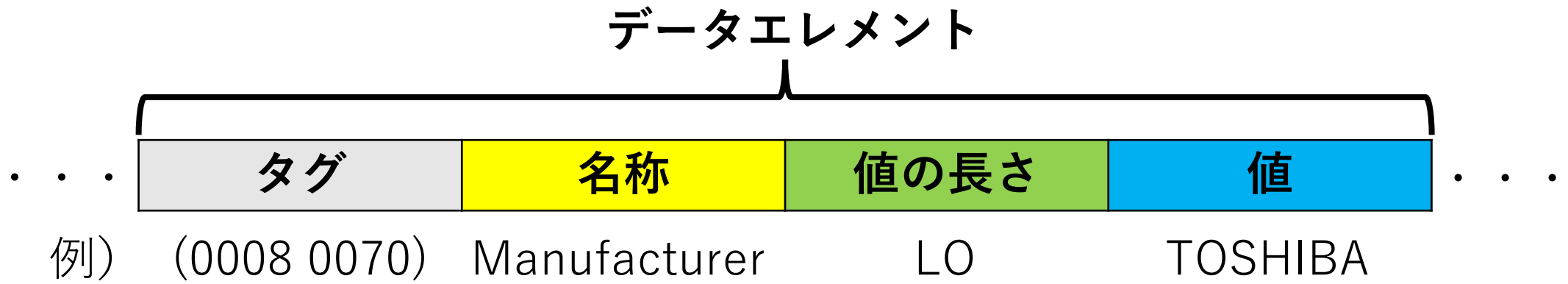
タグ　　データエレメントに格納されている情報を特定するための番号

名称　　データエレメントの名称

値の長さ　値の長さを表すコード

値　　データエレメントに格納されている情報の本体

DICOM画像の構造（２）



- 上記のデータエレメントがずらっと並んでいるのがDICOM画像
- 画素情報以外にも患者や撮影機の情報など、さまざまな情報を含む
  DICOM画像は個人情報として慎重に取り扱う必要がある
- 研究目的などでDICOM画像を使う場合は患者の情報を削除して扱う
  削除＝ダミー情報に置き換える

2. Pydicom

PythonでDICOM画像を扱う

Pydicom

https://github.com/pydicom/pydicom　　https://pypi.org/project/pydicom/

- DICOM画像データを扱うためのPythonライブラリ
- DICOM画像の読み取り、書き込み、編集などの機能を提供している
- オープンソースで現在もメンテナンスと開発が継続中
- pip（Python Package Index）で簡単にインストールできる
  Ubuntu (Linux)における”apt”、Macにおける”brew”と似たようなもの

Google ColabでPydicomをインストール

```
!pip install pydicom
```

- Google Colabは仮想環境のためインストール方法が少々特殊
- 上記のコードを実行することでPydicomをインストールできる

（補足）GDCM

https://pypi.org/project/python-gdcm/

- Grassroot DICOM → GDCM
- オープンソースのクロスプラットフォームライブラリ
- DICOM画像のさまざまな変換や処理に関する機能を提供している
- Pydicomを使うときはGDCMも必要（鳥井の経験則）
  DICOM画像の圧縮方式によってはPydicomのエラーが出る

インストール方法

```
!pip install python-gdcm
```

（補足）ローカル環境におけるインストール

```
pip install pydicom
pip install python-gdcm
```

- 通常は上記のコマンドをターミナルなどで実行する
- コード内に記述する必要はない
- ビックリマークは必要ない
- WindowsやMacの場合はAnacondaなど仮想環境のインストールが必要

3. DICOM画像の読み書きと表示

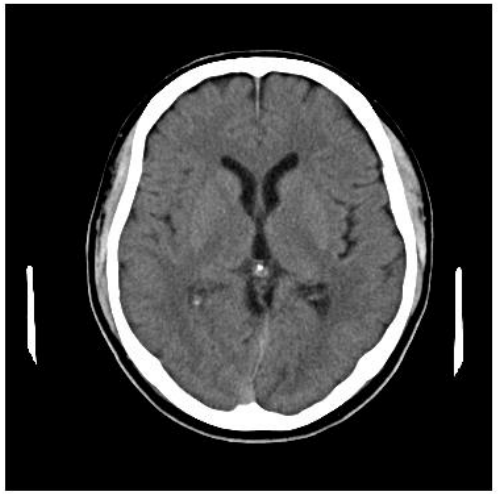
Pydicomを使いこなす

準備

```
import pydicom


from google.colab import drive
drive.mount("/content/drive")
```

- 例によってライブラリをインポートする
- 例によってGoogle Drive上のデータを利用するための2行を書いておく

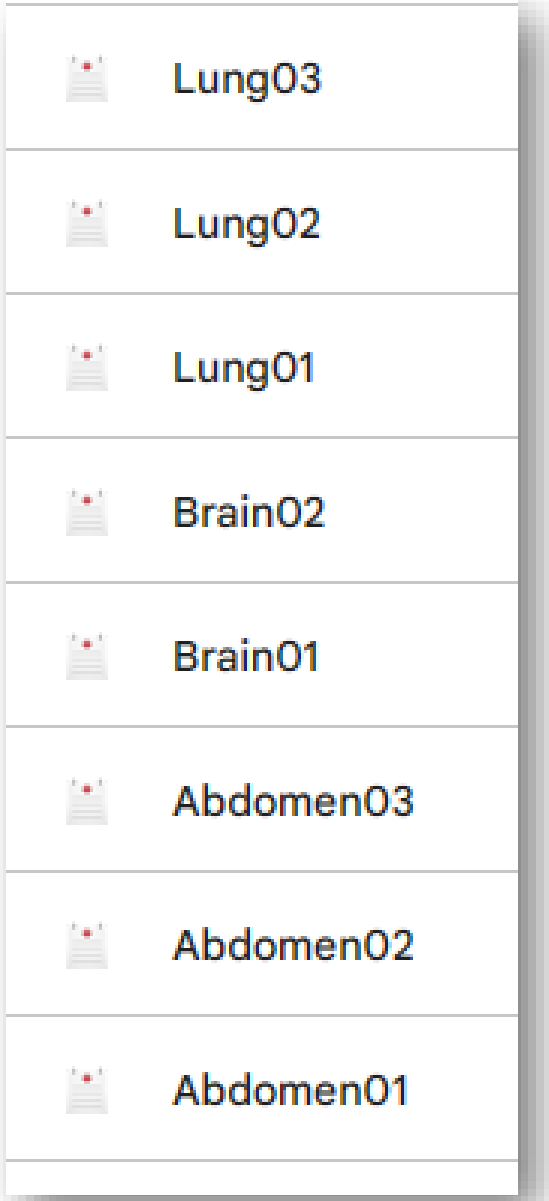
データセット



Brain01

Lung01

Abdomen01



DICOM画像の読み込み

```
dcm_path = "drive/MyDrive/dataset/ct_dcmdir/Brain01"
data = pydicom.dcmread(dcm_path)
```

```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 204
(0002, 0001) File Meta Information Version       OB: b'¥x00¥x01'
(0002, 0002) Media Storage SOP Class UID         UI: CT Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.392.200036.9116.2.6.1.16.1613469034.1289543971.629951
(0002, 0010) Transfer Syntax UID                 UI: Explicit VR Little Endian
(0002, 0012) Implementation Class UID            UI: 1.2.392.200036.9116.2.6.1.100
(0002, 0013) Implementation Version Name         SH: 'TM_CT_CMW_V3.00'
-------------------------------------------------
(0008, 0008) Image Type                          CS: ['ORIGINAL', 'PRIMARY', 'AXIAL']
(0008, 0016) SOP Class UID                       UI: CT Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.2.392.200036.9116.2.6.1.16.1613469034.1289543971.629951
(0008, 0020) Study Date                          DA: '20101112'
(0008, 0021) Series Date                         DA: '20101112'
(0008, 0022) Acquisition Date                    DA: '20101112'
(0008, 0023) Content Date                        DA: '20101112'
(0008, 0030) Study Time                          TM: '153820.000'
(0008, 0031) Series Time                         TM: '153848.921'
```

- pydicom.dcmread(ファイルパス)で読み込み

データエレメントの取得と表示

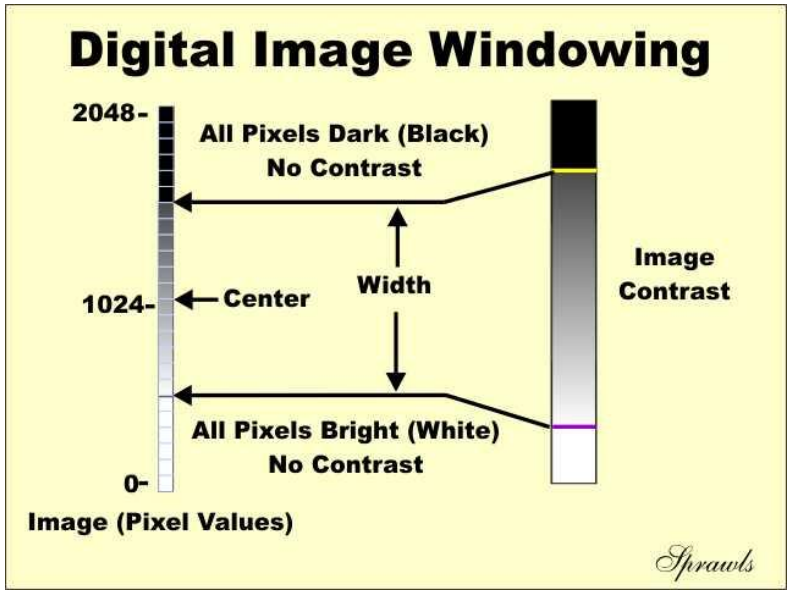
```
pn = data.PatientName
wc = data.WindowCenter
ww = data.WindowWidth
img = data.pixel_array
```

```
Patient Name: Joho^Taro
Window Center: 40
Window Width: 80
Pixel Array: [[-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 ...
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]]
```

- データエレメントの名称がクラスの属性として格納されている
- 名称を指定することで値を取り出すことができる（スペースは省略）
- pixel_arrayは2次元配列の形になっている
- このpixel_arrayは基本的にそのままでは適切に表示できない

階調処理（Windowing）

- 画像の明るさとコントラストを調節する処理
- 画素値を画面表示できる値域にマッピングする
- 通常は[0, 255]にマッピングする（8bit）
- 画面表示する画素値の範囲→**ウィンドウ**



http://www.sprawls.org/resources/DIGPROCESS/window.jpg

ウィンドウ中心（Window Center）

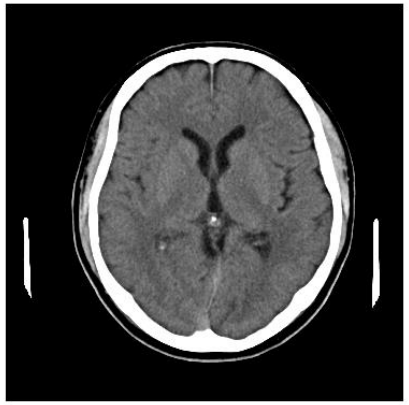
ウィンドウの中心となる画素値

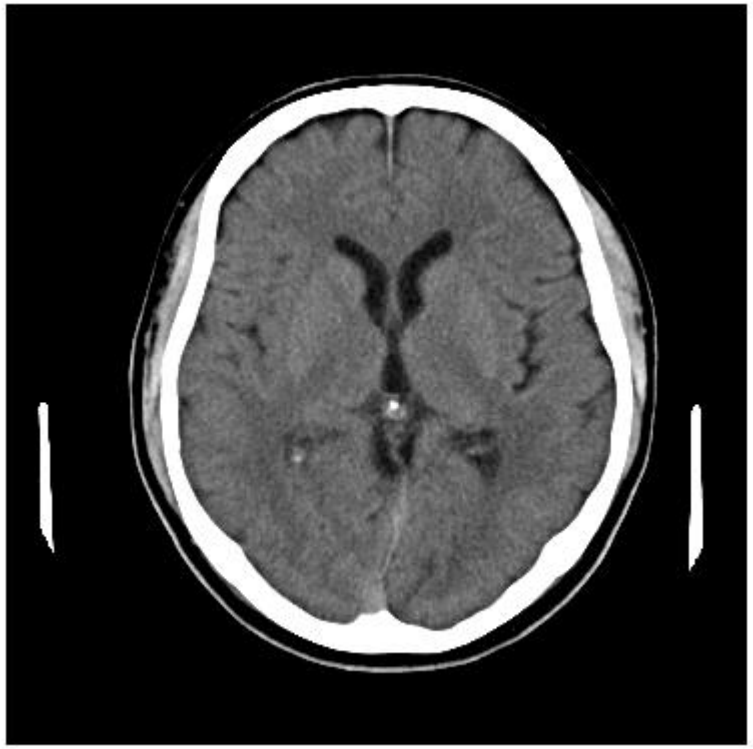
ウィンドウ幅（Window Width）

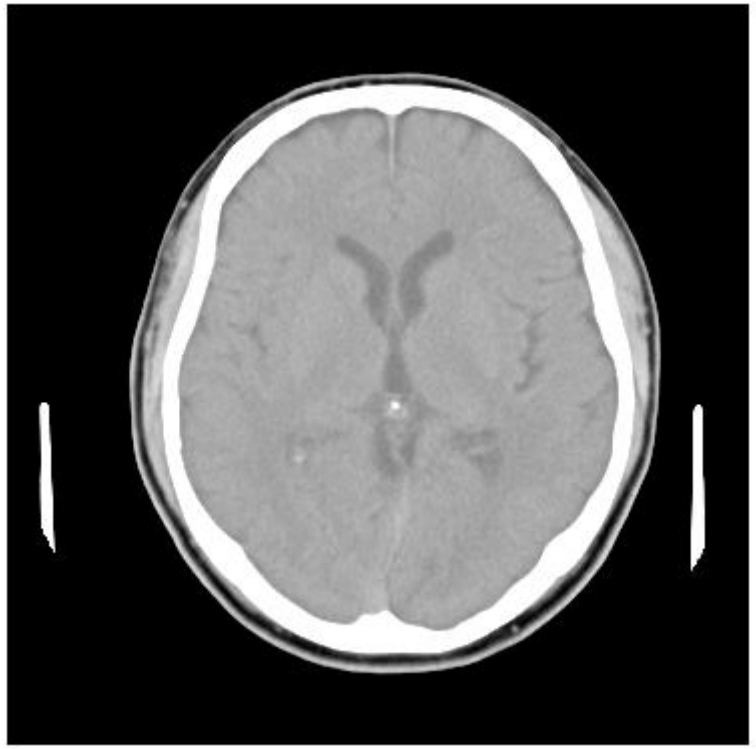
ウィンドウの幅、ウィンドウの最大値－ウィンドウの最小値

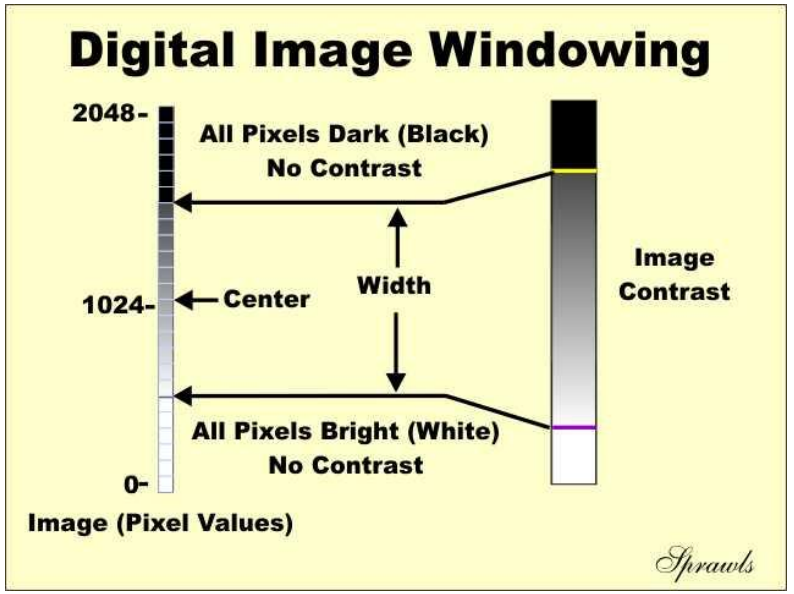
DICOM画像の表示

```
max = wc + ww / 2   # ウィンドウの最大画素値
min = wc - ww / 2   # ウィンドウの最小画素値
plt.imshow(img, cmap="gray", vmax=max, vmin=min)
plt.axis("off")
plt.show()
```







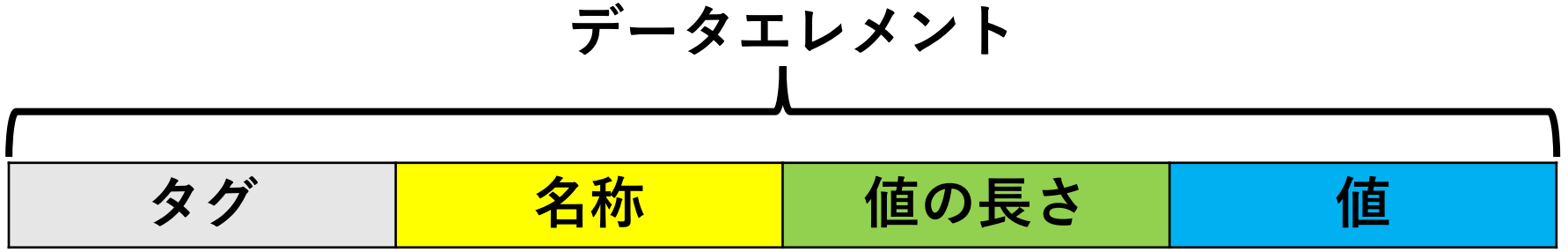


window center: 40
window width: 80
max: 80
min: 0

window center: 0
window width: 200
max: 100
min: -100

http://www.sprawls.org/resources/DIGPROCESS/window.jpg

タグの構成



例）（0008 0070） Manufacturer LO TOSHIBA

- タグは（**グループ番号**、**エレメント番号**）で構成される
  （2byte, 2byte)の計4byteを16進数で表現する
- DICOM画像の各情報はグループに分けられている

<u>グループ番号の例</u>

0008：基本情報　　0010：患者情報

0018：検査情報　　0020：検査情報　　0028：画像情報

基本情報（0008）

```
SOP Instance UID  : (0008, 0018) 1.2.392.200036.9116.2.6.1.16.1613469034.1289543971.629951
Study Date        : (0008, 0020) 20101112
Study Time        : (0008, 0030) 153820.000
Modality          : (0008, 0060) CT
```

- DICOM画像の基本的な情報が含まれているグループ
- 検査日時や画像診断装置などの情報がある

患者情報（0010）

```
Patient Name  : (0010, 0010) Joho^Taro
Patient ID    : (0010, 0020) 12345678
```

- 患者の情報が含まれているグループ
- Patient Name（患者名）やPatient ID（患者ID）などの情報がある
- 患者名の ^ はスペースを表す（苗字と名前を分離する）

※当然だが、今回はダミー情報が入っている

検査情報（0018）

```
Body Part Examined  : (0018, 0015) HEAD
Slice Thickness     : (0018, 0050) 8.0
Convolution Kernel  : (0018, 1210) FC21
```

- 撮影時の検査に関する情報が含まれているグループ
- 検査部位やスライス厚などの情報がある

検査情報（0020）

```
Study Instance UID      : (0020, 000d) 1.2.392.200036.9116.2.6.1.16.1613469034.1289543900.390492
Series Instance UID     : (0020, 000e) 1.2.392.200036.9116.2.6.1.16.1613469034.1289543929.202106
Series Number           : (0020, 0011) 1
Image Position (Patient): (0020, 0032) [-120.000, -119.7761, -462.6741]
Acquisition Number      : (0020, 0012) 17
Instance Number         : (0020, 0013) 17
Slice Location          : (0020, 1041) +430.00
```

- 撮影時の検査に関する情報が含まれているグループ
- 3D（CTなど）空間上における位置などの情報がある
- Image Position（画像位置）は[x, y, z]の順で格納されている
  単位はmm
- zを参照することで3D空間における断面画像の位置を知ることができる

画像情報（0028）

```
Photometric Interpretation : (0028, 0004) MONOCHROME2
Rows                       : (0028, 0010) 512
Columns                    : (0028, 0011) 512
Pixel Spacing              : (0028, 0030) [0.468, 0.468]
Bits Allocated             : (0028, 0100) 16
Bits Stored                : (0028, 0101) 16
High Bit                   : (0028, 0102) 15
Pixel Representation       : (0028, 0103) 1
Window Center              : (0028, 1050) 40
Window Width               : (0028, 1051) 80
Rescale Intercept          : (0028, 1052) 0
Rescale Slope              : (0028, 1053) 1
```

- 画像に関する情報が含まれているグループ
- 画像処理に必要な情報がある
  研究開発上では最も重要！
- Photometric Interpretationの値によって白黒の解釈が異なる
  MONOCHROME2：背景が黒
  MONOCHROME1：背景が白
- Pixel Spacing（画素間隔）は1画素あたりの幅と高さを表す
  単位はmm
- Rescale InterceptとRescale Slopeは後ほど解説する

画素値（7fe0, 0010）

```
Pixel Data: [[-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 ...
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]
 [-2048 -2048 -2048 ... -2048 -2048 -2048]]
(512, 512)
```

- Pydicomではpixel_arrayで簡単に取得が可能
- np.array形式となっている
- shapeを見ると画像サイズがわかる
  今回は512×512

データの匿名化（１）

```
Patient Name  : (0010, 0010) Joho^Taro
Patient ID    : (0010, 0020) 12345678
```

- DICOM画像には検査を受けた患者の情報も保存されている
- これらをダミー情報に置き換えることで匿名化を行う
- Pydicomでもダミー情報への置き換えが可能である
- ディレクトリ内のDICOM画像を一括して匿名化するコードを紹介する

データの匿名化（２）

```
dummy_info = [['00100010', 'no name'], ['00100020', 'no ptid']]


files = os.listdir(orgdir)  # orgdir内のすべてのファイル名を取得
print(files)
```

```
['Lung02', 'Abdomen01', 'Abdomen03', 'Lung03', 'Abdomen02', 'Brain01', 'Lung01', 'Brain02']
```

- ダミー情報として患者名を"no name"、患者IDを"no ptid"とする
- ダミー情報はリストとして定義しておく

データの匿名化（３）

```
for i, name in enumerate(files):
    path = orgdir + name
    data = pydicom.dcmread(path)                # DICOM画像の読み込み
    for j, info in enumerate(dummy_info):
        tag = info[0]
        value = info[1]
        data[tag].value = value                 # 情報の置き換え
    save_name = f"{dstdir}anony_{name}"         # 保存ファイル名
    pydicom.dcmwrite(save_name, data)           # DICOM画像の書き込み
    print(data["00100010"])                     # 患者名を表示する
```

データの匿名化（４）

data[tag].value = value

- data[タグ番号8桁].value = 値　とすることで値の置き換えができる

pydicom.dcmwrite(save_name, data)

- pydicom.dcmwrite(保存ファイル名, data)でDICOM画像を保存できる

データの匿名化（5）

```
['Lung02', 'Abdomen01', 'Abdomen03', 'Lung03', 'Abdomen02', 'Brain01', 'Lung01', 'Brain02']
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
(0010, 0010) Patient's Name                      PN: 'no name'
```

- 実行結果はこのようになる
- 患者名（0010, 0010）に"no name"が代入されたことがわかる
- 同様に患者ID（0010, 0020）も"no ptid"に置き換わっている

画素値について

※歯科では少し扱いが違うかもしれない…

- DICOM画像の画素値は純粋なCT値ではない場合がある
  何らかの理由ですべて正の値に変換されているなど
- 本来のCT値はマイナスの値をもつ
  カルシウムが1000、水が0、空気が-1000と定義されている
- Rescale InterceptとRescale Slopeを用いて本来のCT値に変換できる
- 画像処理を行うときはこの変換を最初に必ず行っておく

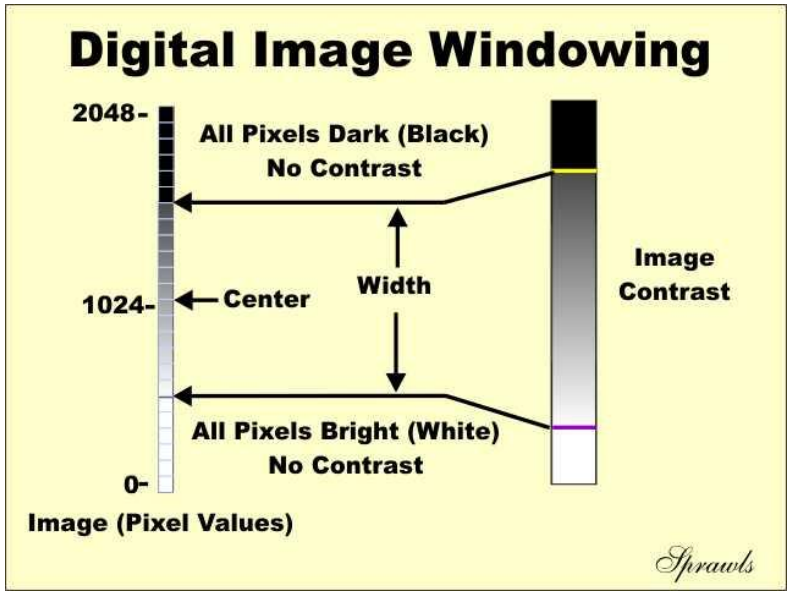
$$CTvalue = RescaleSlope \times PixelData + RescaleIntercept$$

$PixelData$ =DICOM画像に保存されてる画素値

階調処理の定義（8bit）

$$PixelValue = 255 \times \frac{CTvalue - min}{max - min},$$

$$0 \leq PixelValue \leq 255$$



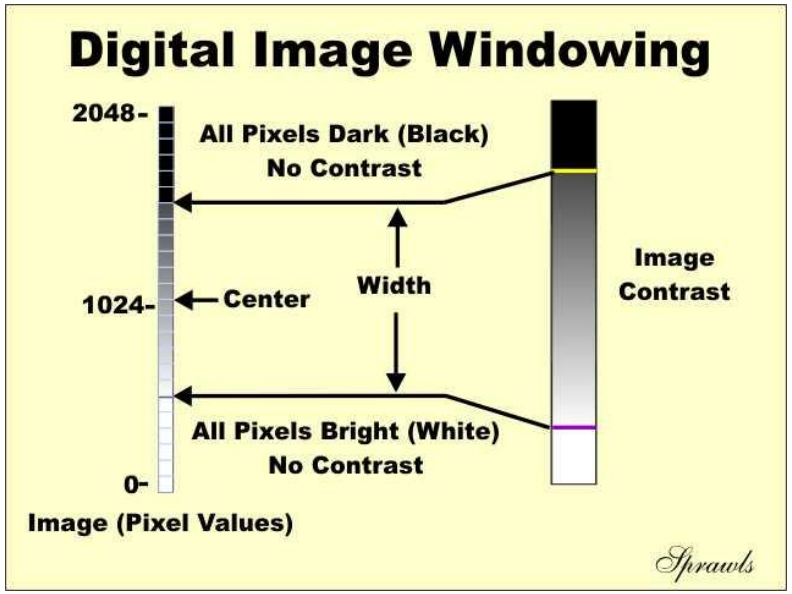
http://www.sprawls.org/resources/DIGPROCESS/window.jpg

- 階調処理の実装は上記の式に従う
- 計算上、[0, 255]に収まらない値が出てくる（ウィンドウ範囲外の値）
- 0未満を0、255を超える値を255に変換する処理が必要
  忘れると大変なことになります
- np.clipで簡単に実装できる

階調処理の定義（16bit）

$$PixelValue = 65535 \times \frac{CTvalue - min}{max - min},$$

$$0 \leq PixelValue \leq 65535$$



http://www.sprawls.org/resources/DIGPROCESS/window.jpg

- 階調処理の実装は上記の式に従う
- 16bitで処理したい場合は255を65535に変更する

$$65535 = 2^{16} - 1$$

DICOM画像の計算処理における注意点

- 型に気を配っておく
- 画像の表示がおかしい…となったときは型を疑ってみる
- 計算処理では明示的に値を浮動小数点に変換した方がよい（経験則）
  Numpyの機能により基本的に必要ないが、バグの温床になる
- 画像の表示や保存の際は適切な型にキャストしておく
  基本的に8bit画像はnp.uint8、16bit画像はnp.uint16にキャストしておく

【例】OpenCVで16bitのPNGとして保存したい

OpenCVの画像保存はデフォルトでは勝手に8bit unsigned intになる

→16bitで保存したい場合はnp.uint16にキャストする必要がある

DICOM画像の画素値変換処理（１）

```
dcm_path = "drive/MyDrive/dataset/ct_dcmdir/Brain01"
data = pydicom.dcmread(dcm_path)


wc = data.WindowCenter
ww = data.WindowWidth
ri = data.RescaleIntercept
rs = data.RescaleSlope
img = data.pixel_array
```

DICOM画像の画素値変換処理（２）

```
ct = img * rs + ri                                  # CT値への変換
max = wc + ww / 2                                   # ウィンドウ上限値
min = wc - ww / 2                                   # ウィンドウ下限値
res = 255 * (ct - min) / (max - min)                # 階調処理
res = np.clip(res, 0, 255)                          # [0, 255]に変換
cv2.imwrite("drive/MyDrive/res_8bit.png", res)      # PNGで保存
```

$$CTvalue = RescaleSlope \times PixelData + RescaleIntercept$$

$$PixelValue = 255 \times \frac{CTvalue - min}{max - min},$$

$$0 \leq PixelValue \leq 255$$

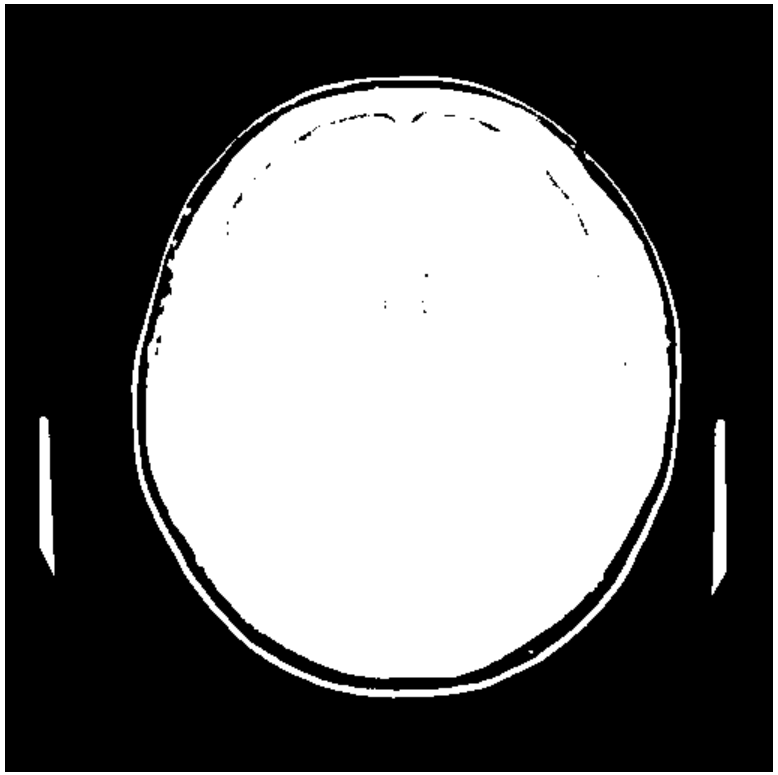
DICOM画像の画素値変換処理（３）

```
plt.imshow(res, cmap="gray")
plt.axis("off")
plt.title("8bit")
plt.show()
```

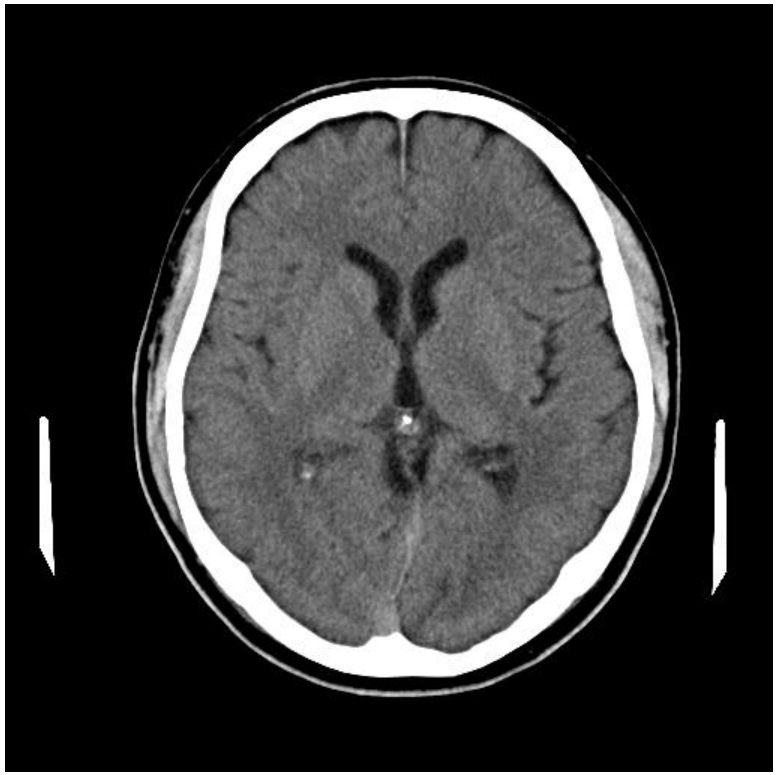
DICOM画像の画素値変換処理（４）

```
res = 65535 * (ct - min) / (max - min)
res = np.clip(res, 0, 65535)

# キャストなし
cv2.imwrite("drive/MyDrive/res_16bit.png", res)

# キャストあり
res = res.astype(np.uint16)       # np.uint16にキャスト
cv2.imwrite("drive/MyDrive/res_16bit_casted.png", res)
```

DICOM画像の画素値変換処理（５）

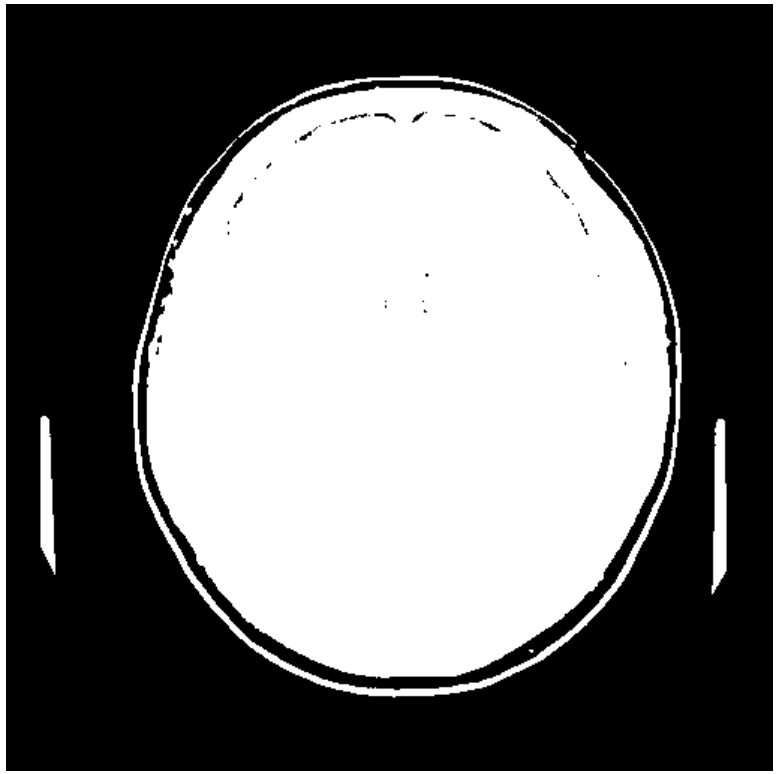


キャストなし

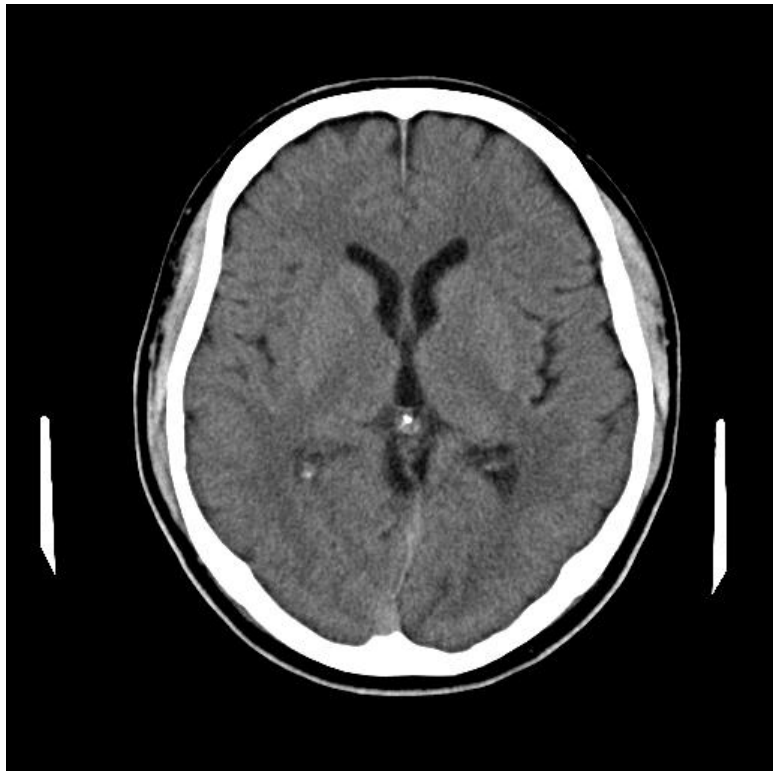


キャストあり

DICOM画像の画素値変換処理（６）



DICOM画像の画素値変換処理（7）



その他の処理

- 今回紹介した処理以外にもPydicomでさまざまなことができる
- 以下に処理の例を挙げる（コードは割愛）
  ぜひチャレンジしてみてください！

【例】
- Image Positionのz座標の値を用いてCTスライスをソート
- CTスライスのデータをすべて読み込んで3次元データを構築
- 独自に作成したデータエレメントの追加　　etc...

演習

- Pydicomを用いてさまざまな医用画像処理を行う

github.com/wt501/sample_programs/blob/main/dicom_processing.ipynb

まとめ

- DICOMについておさらいした
- PydicomをDICOM画像の処理を行った

【次回】

- 胸部X線画像からの肺炎のAI診断を行う