

人工知能を 作ろう、使おう、育てよう！

瀬々 潤

sesejun@humanome.jp

株式会社ヒューマノーム研究所・代表取締役社長



全ての資料は

<https://github.com/HumanomeLab/CT2019>

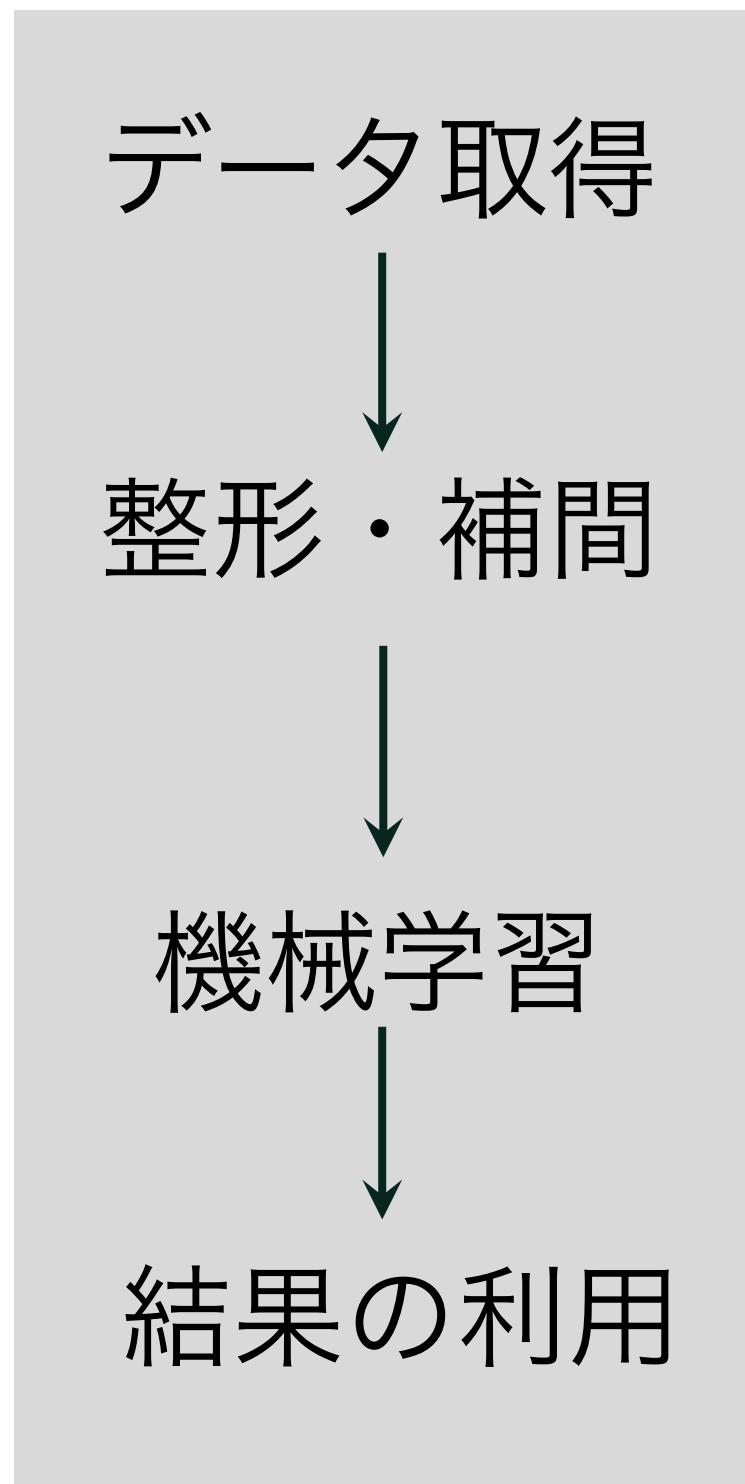
に掲載しています。

演習、復習等に、お役立てください。





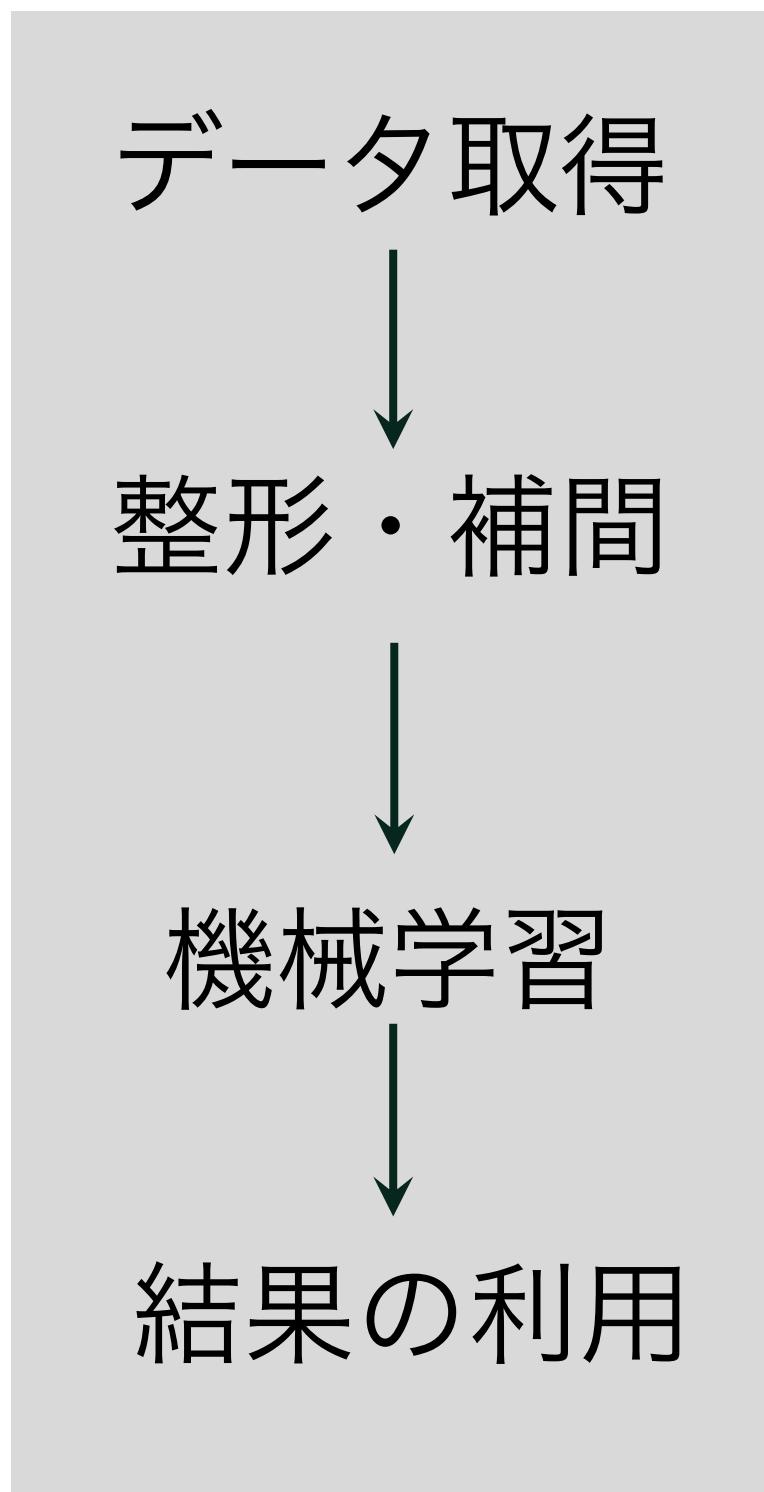
AI（機械学習）開発の一般的な手順



- 年齢・性別が含まれた画像情報の収集
- ピンぼけの画像や、人が写っていないものなど、意図しないデータの削除
- 機械学習による、顔→年齢・性別モデルの学習
- 飲料の推薦



AI（機械学習）開発の一般的な手順



データを収集・
整理する人材

AIシステムを
開発する人材

AIシステムを
運用・チューニング
する人材

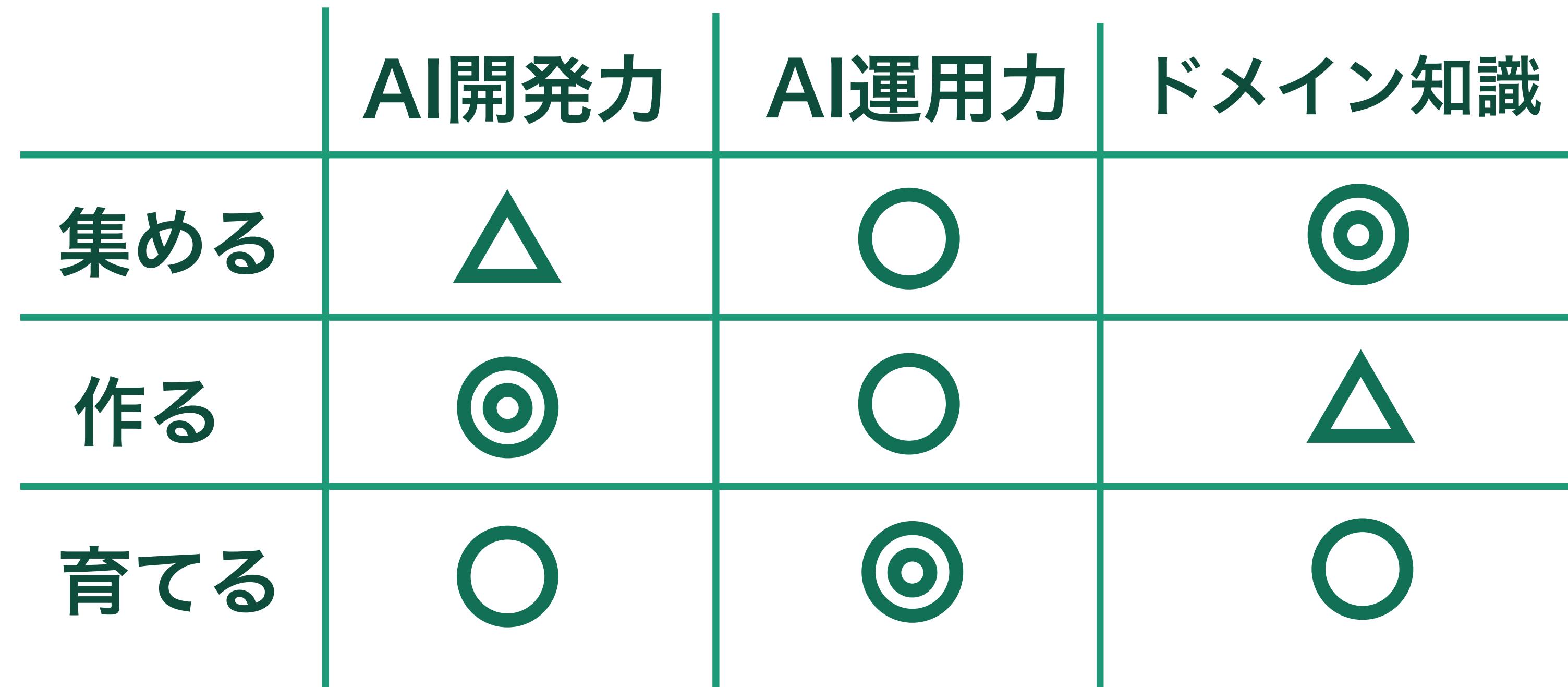
集める

作る

育てる



それぞれの人材に、異なるスキルが必要。
横断的なスキルを持つほど、開発はスムーズに



画像の機械学習には3種類の目標がある 用途に応じて選択を

クラス分類



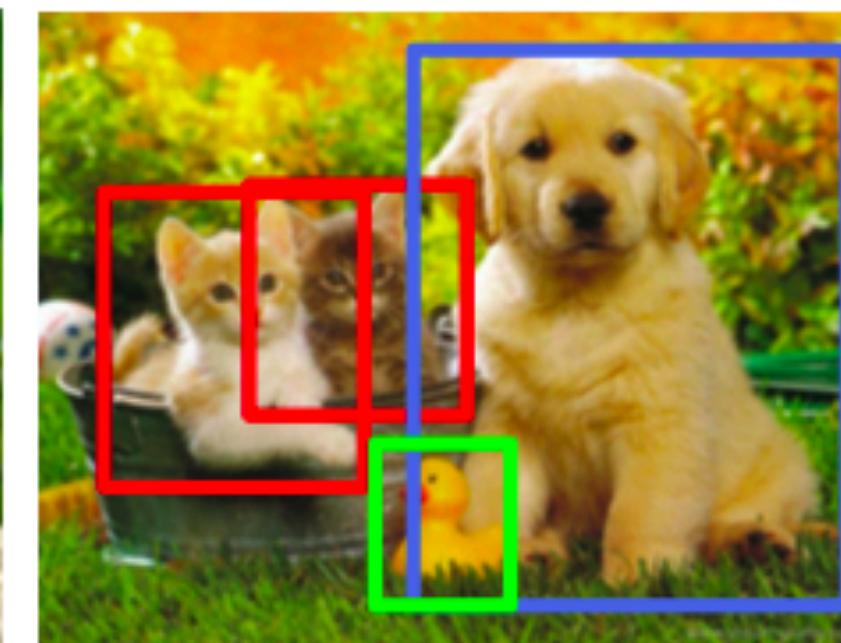
CAT

位置同定
(Localization)



CAT

セグメンテーション
(Segmentation)



CAT, DOG, DUCK

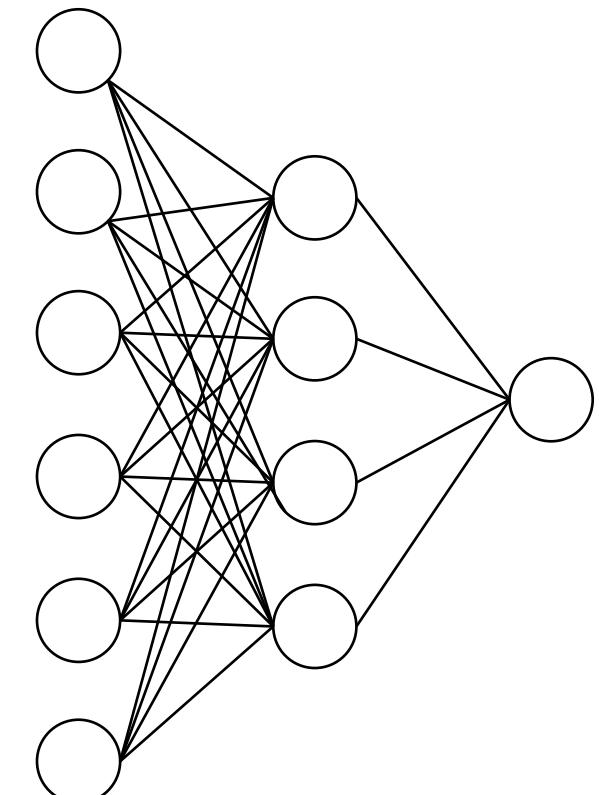
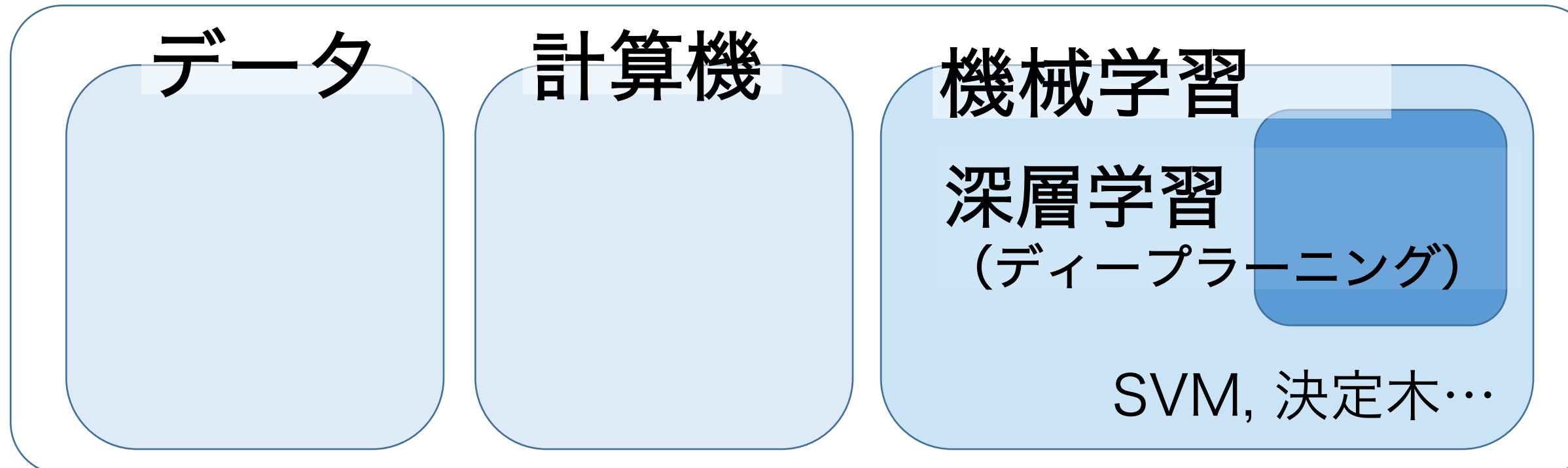


CAT, DOG, DUCK

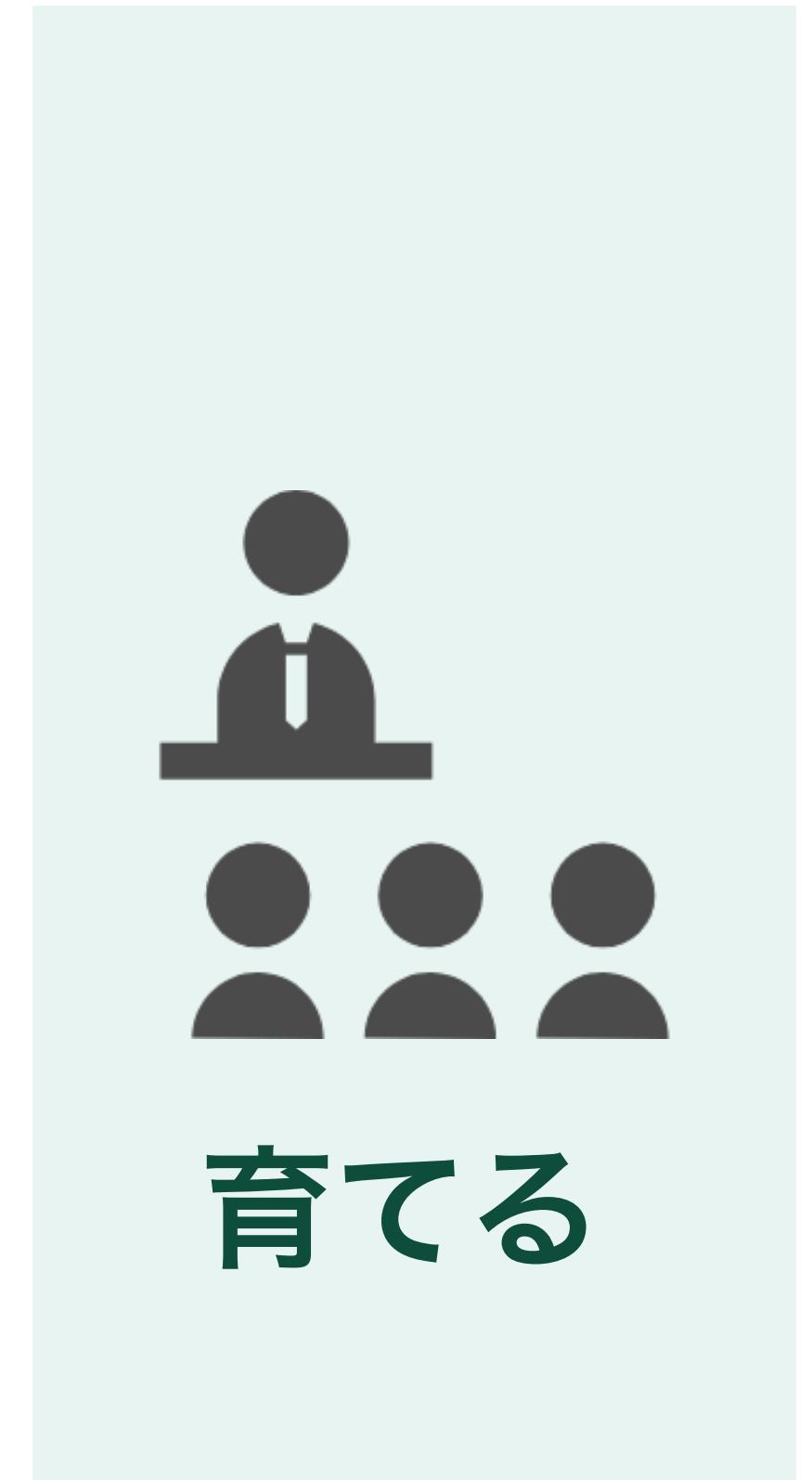
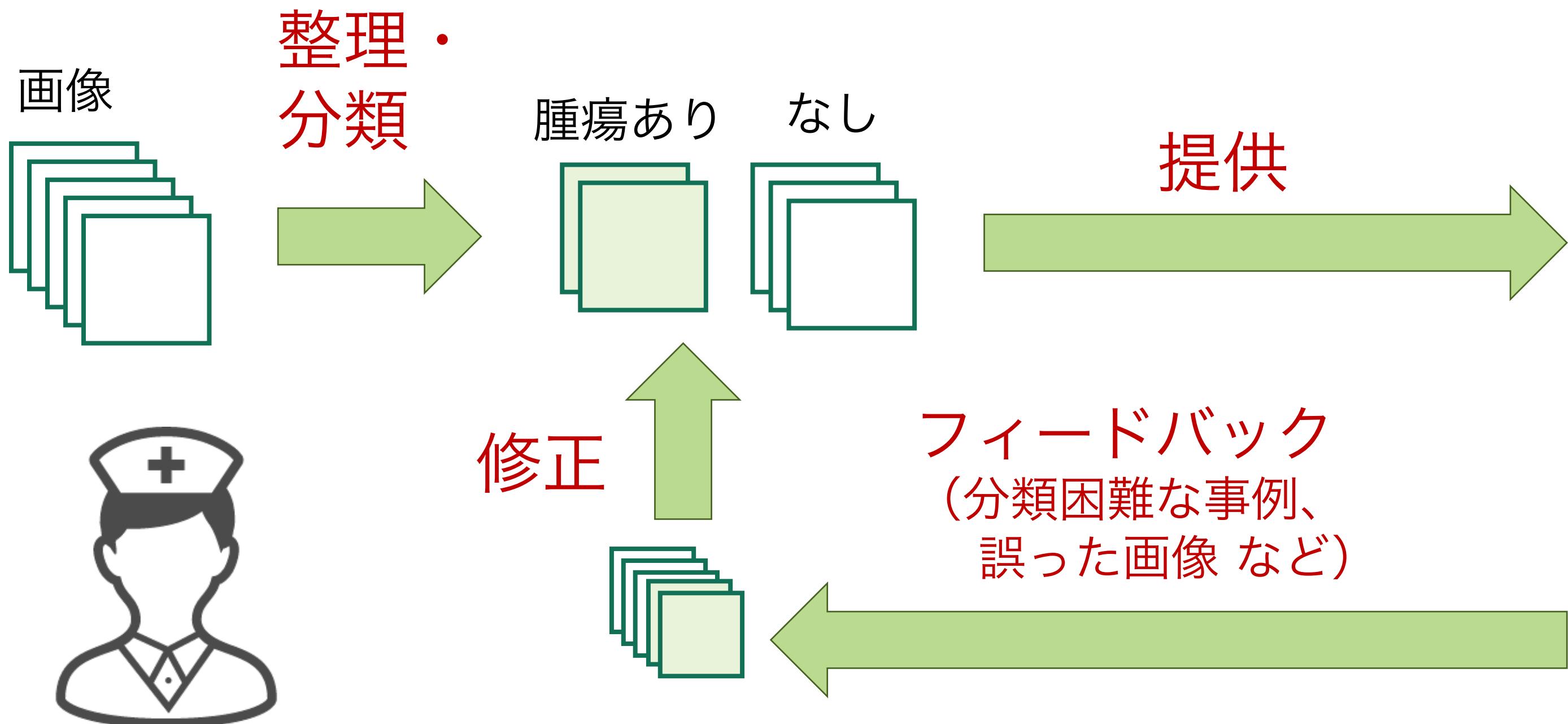
機械学習では、プログラムの作成に加え、パラメータのチューニングが必要

- ・ネットワークの構成とハイパープラメータ（学習率等）を決定
 - ・計算機が、データを基に、パラメータを計算する。
- ・ネットワークやハイパープラメータを調整しつつ、精度を上げる

人工知能

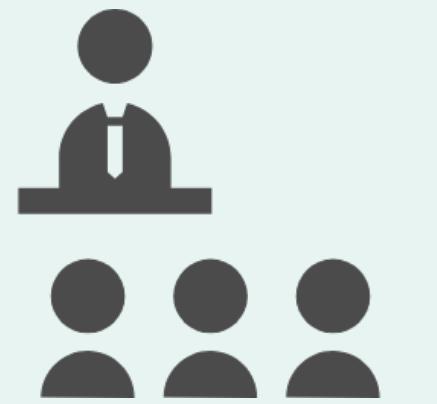


集める：データの整理＆教師データの作成

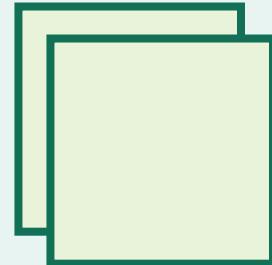


作る：人工知能エンジンの作成

育てる



腫瘍あり



なし



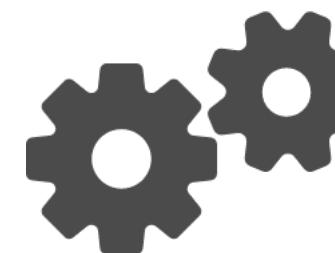
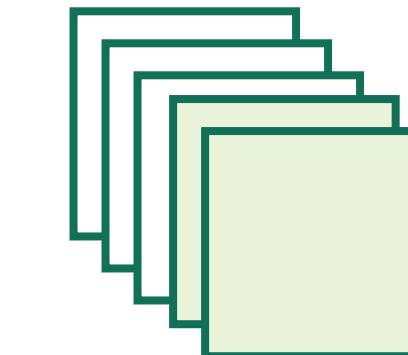
データの提供



プログラムの作成



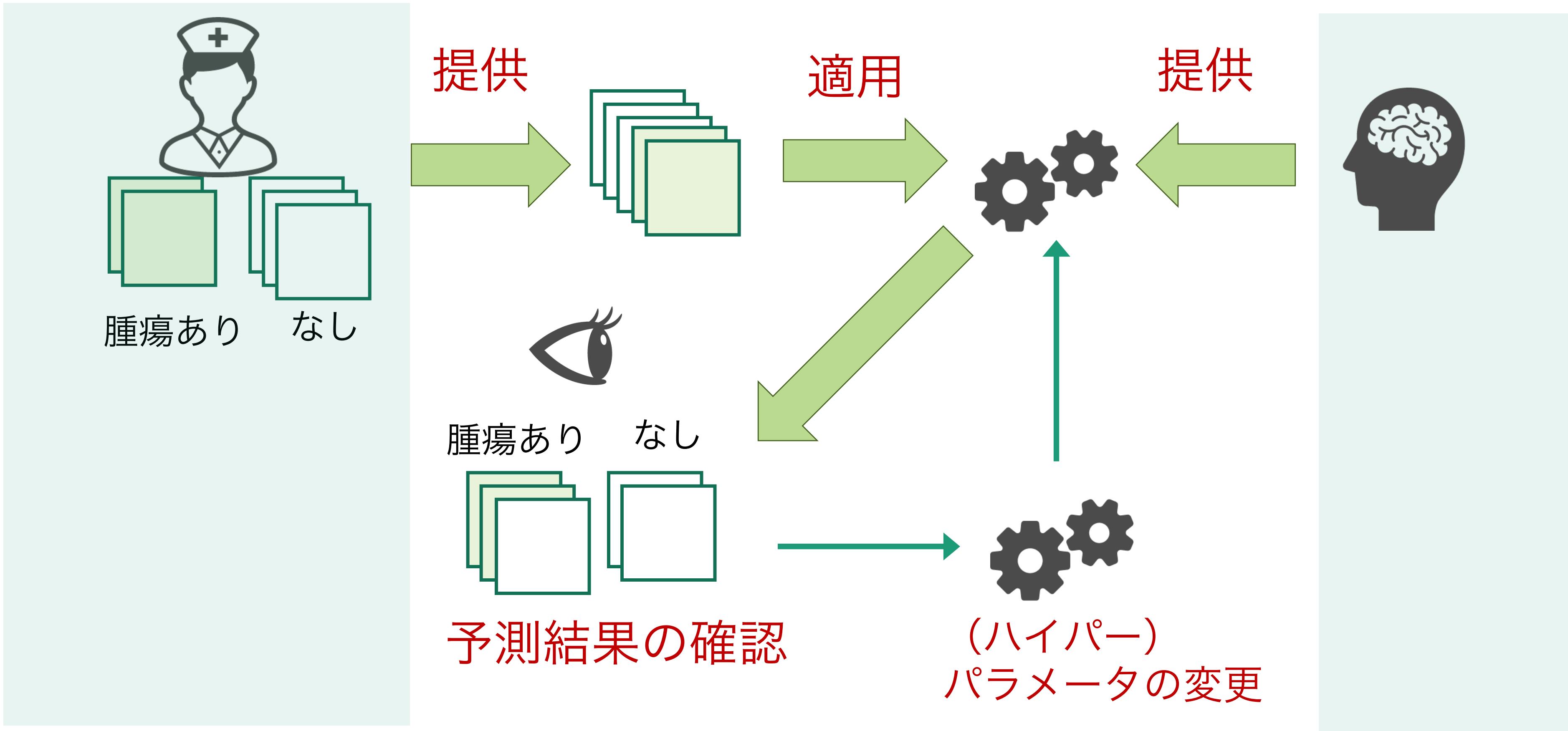
実データ
でのテスト



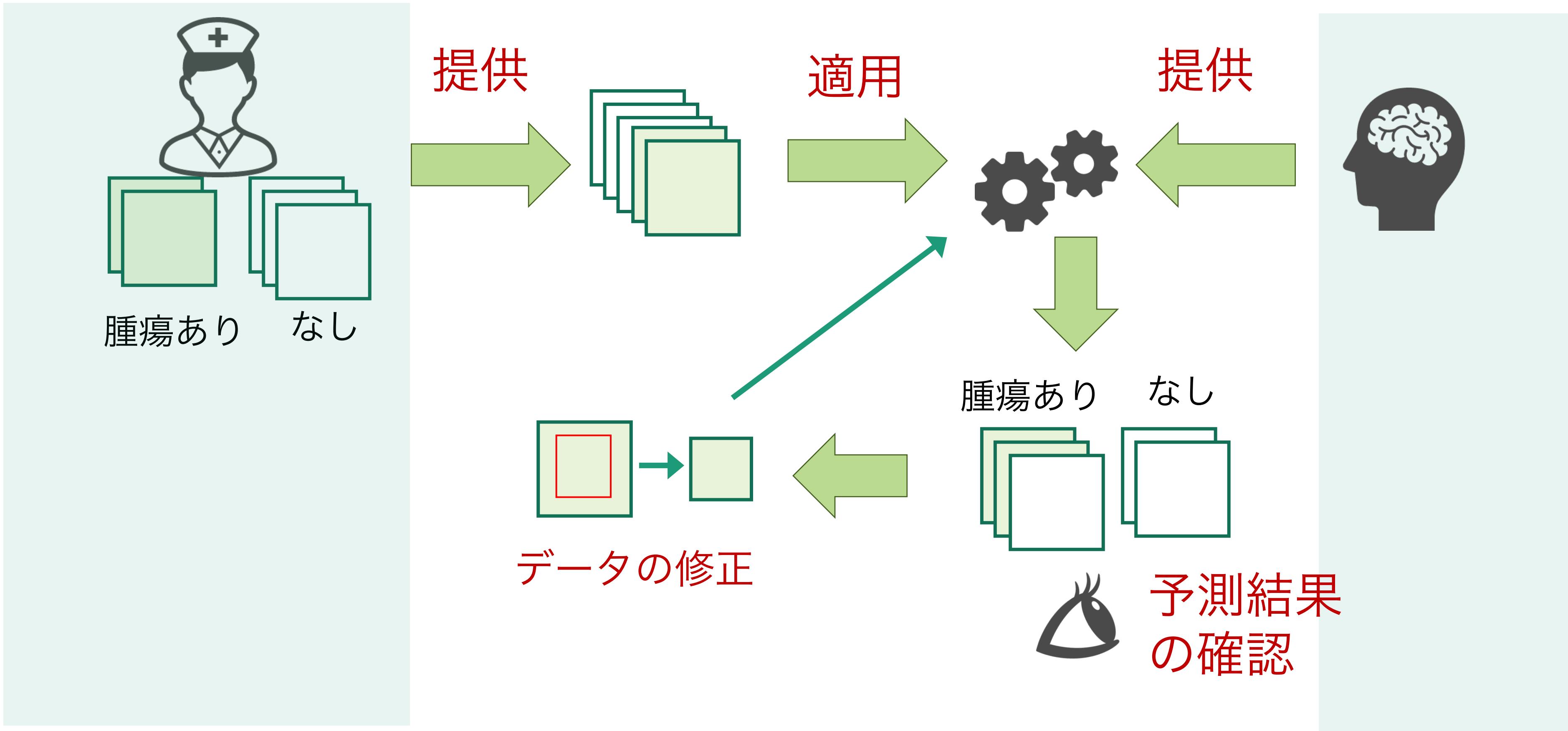
プログラム・
モデル提供



育てる（モデル）：人工知能を成熟させる



育てる（データ）：データを成熟させる



工学の鉄則：スマールスタート

- ・ いつもモノツクリは複雑
- ・ まずは、簡単な問題かつ少数のデータから始める
 - ・ 難題からスタート
 - ・ 失敗したときに、何が原因かわからない
 - ・ 大規模なデータからスタート
 - ・ いつまでもスタートできない
 - ・ 徐々に、データ量を増やしたり、難しい問題へと発展させる



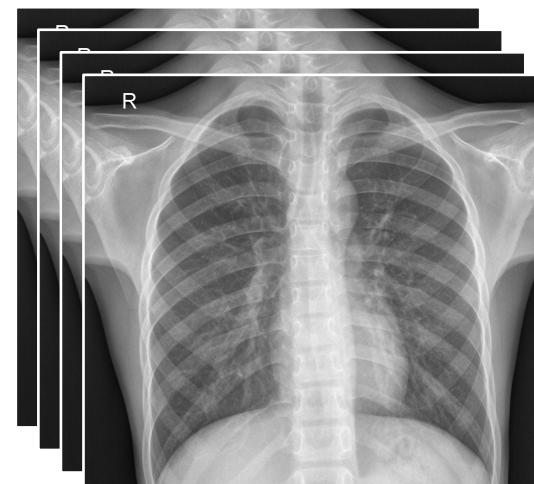


使ってみよう

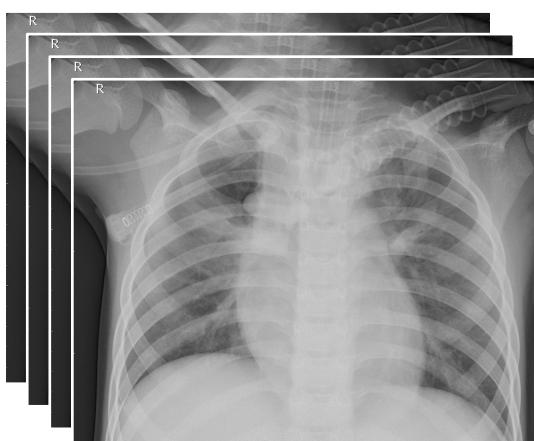


演習1（使ってみる）：肺炎患者の予測

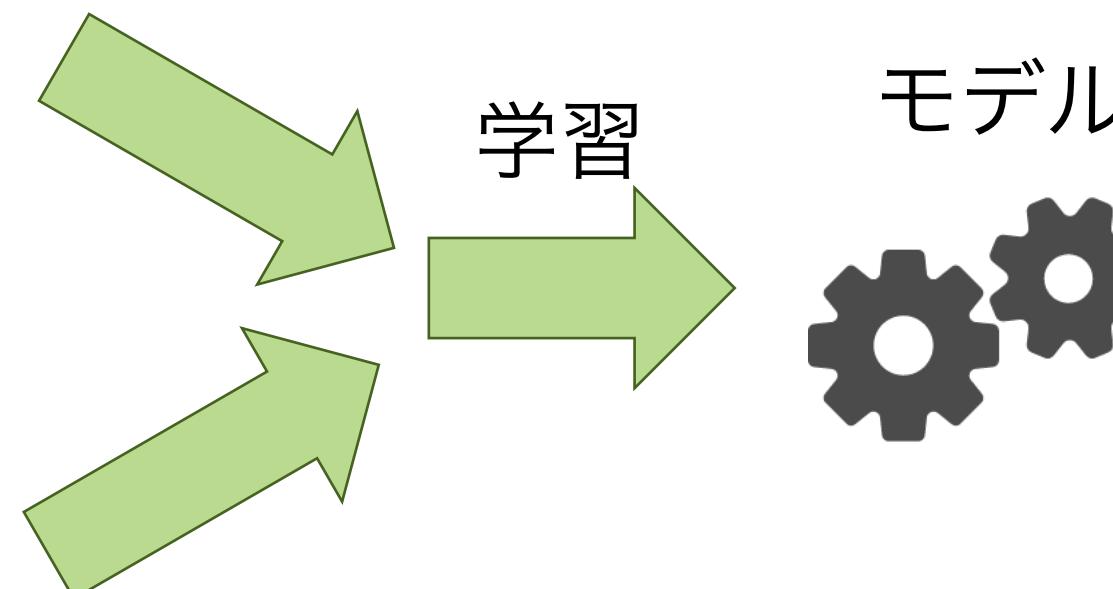
- ・X線で申し訳ない・・・



通常 (NORMAL)



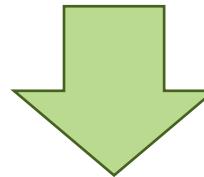
肺炎患者 (PNEUMONIA)



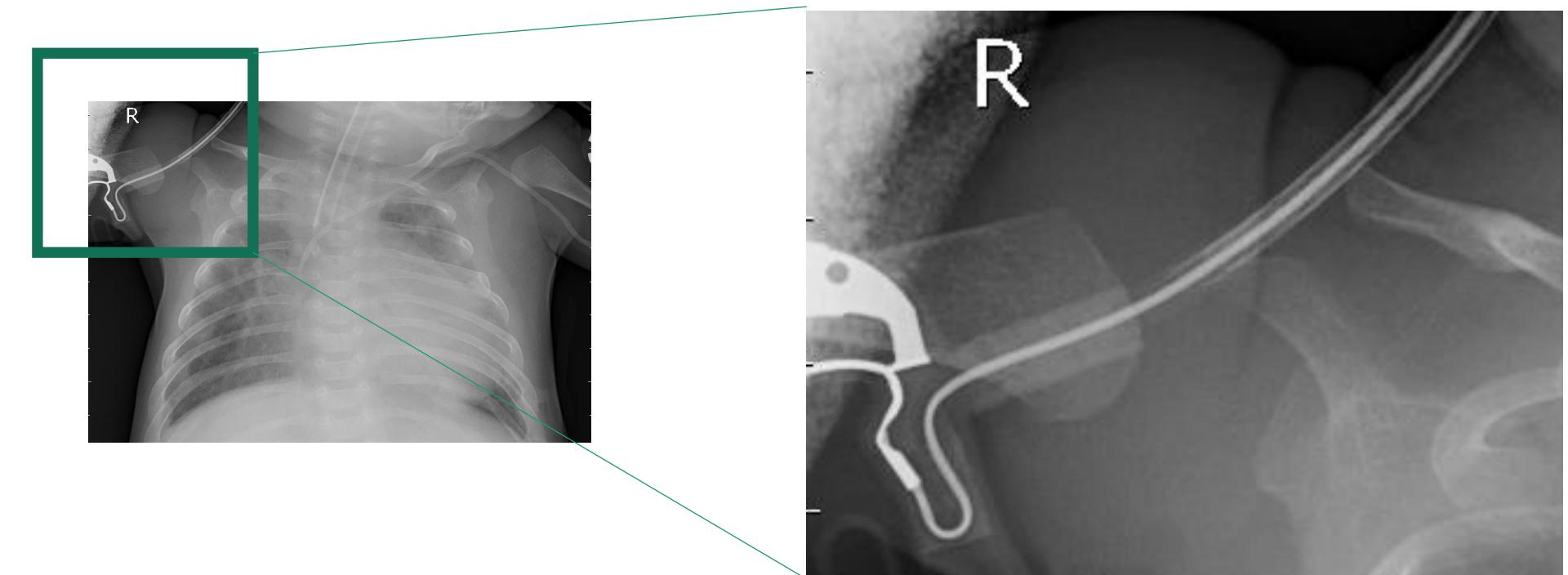
演習2 (データを集め、育てる) :

肺の手動切り出し

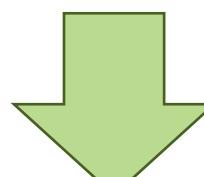
- 学習に不要な情報
 - 画像のメモ、器具、etc.



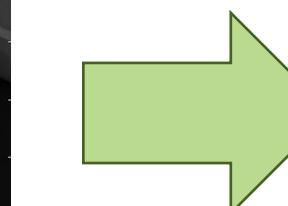
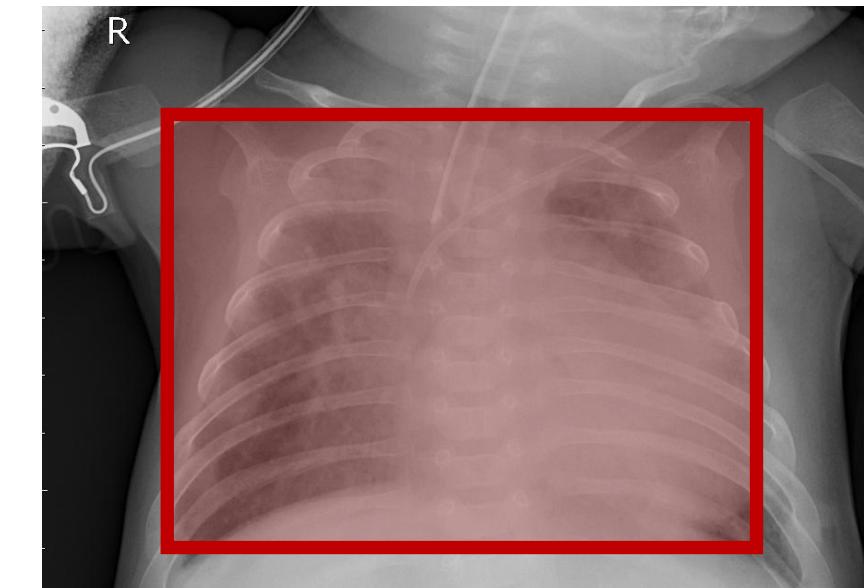
無ければもっと学習しやすいかも
(適切に学習できるかも)



肺の部分だけ、切り出してみる



学習



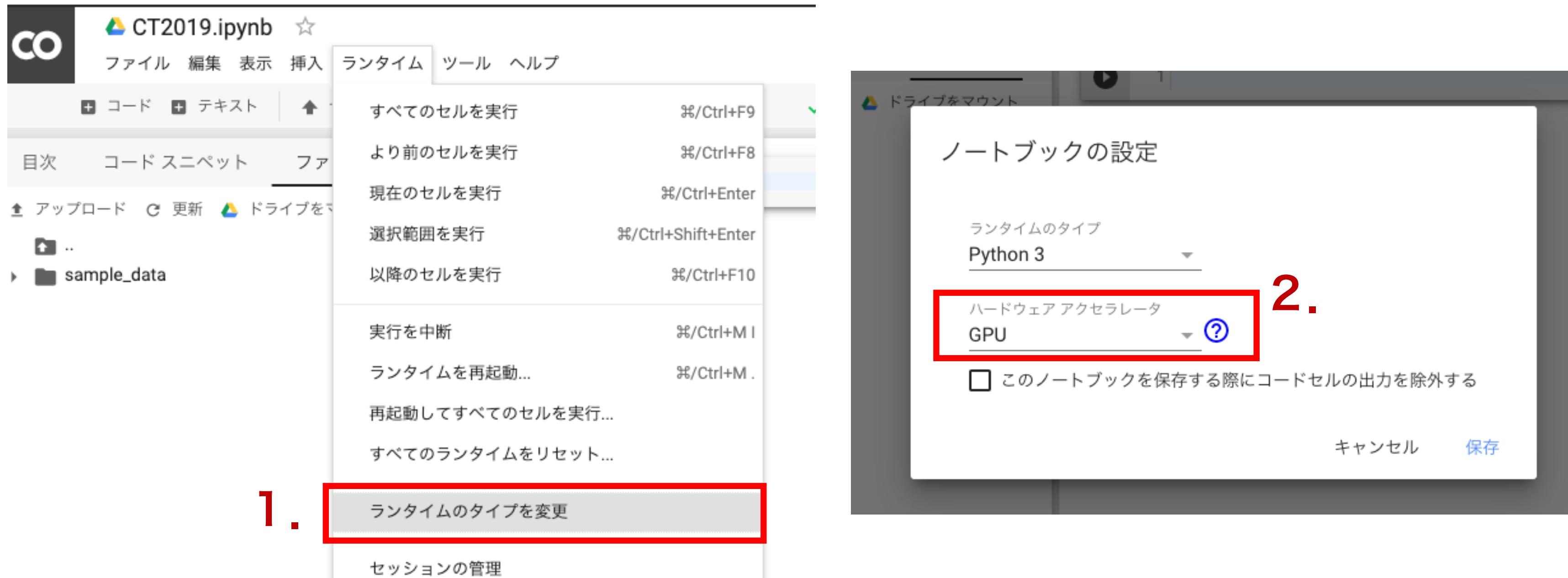
演習で使うファイルを開く

1. Webブラウザから <https://colab.research.google.com> にアクセスする。
2. 「ファイル」 → 「ノートブックを開く」をクリックし、下図のようなポップアップが表示される。下記のa-cの手順で ipynb を選択する。
 - a. 「GITHUB」のタブをクリックする
 - b. 「HumanomeLab」を検索。レポジトリとブランチが選択できるようになる。
 - c. レポジトリ：HumanomeLab/CT2019、ブランチ：master を選択する
 - d. CT_screening_2019.ipynbをクリックする



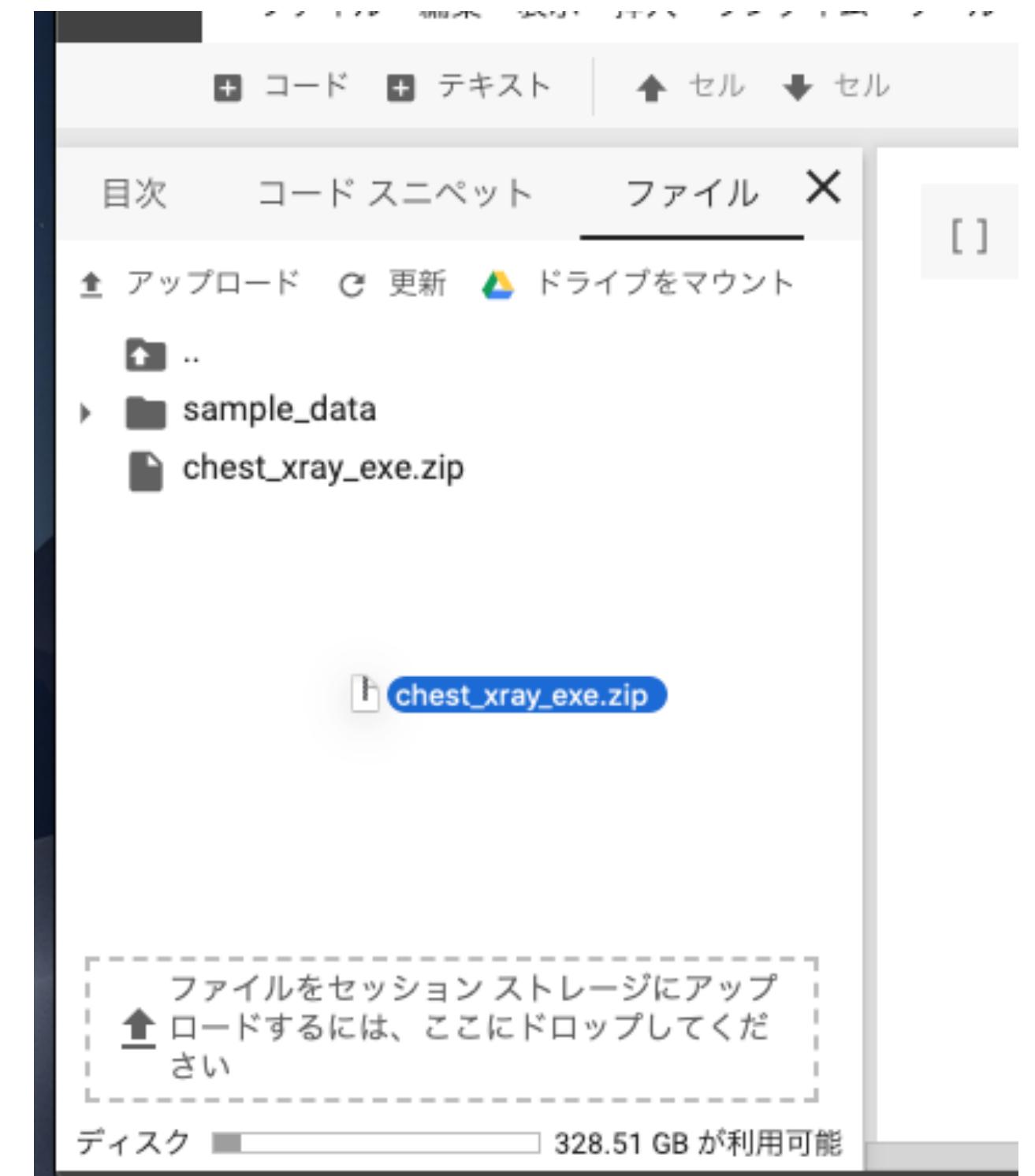
ColaboratoryのGPUを設定

1. 「ランタイム」から、「ランタイムのタイプを変更」を選択
2. 別窓が開くので、「ハードウェアアクセラレータ」がGPUであることを確認



Colaboratoryにデータのアップロード

1. ダウンロードした "CT2019_screening.zip" を、左の「ファイル」タブの中にドラッグアンドドロップ
 - ・アップロードに5-10分程度要する

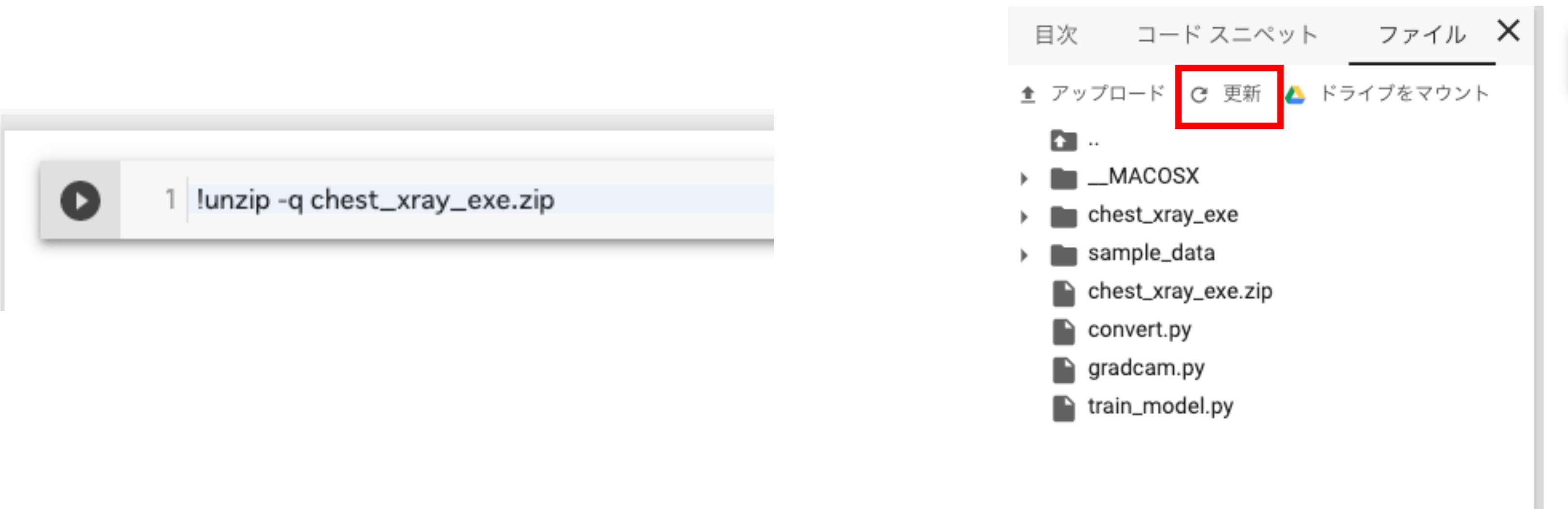


アップロードしたデータの展開

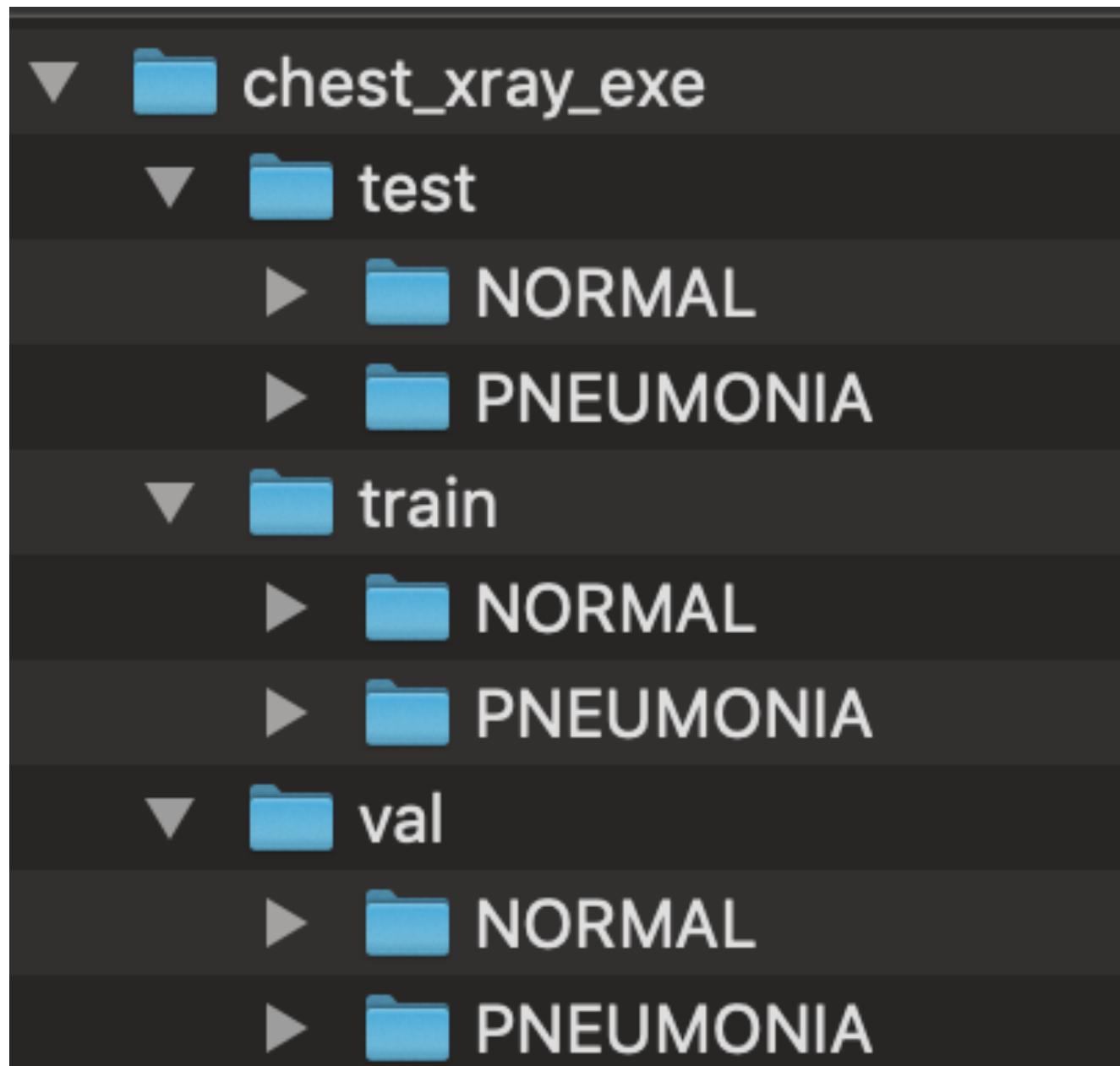
1.右のまどに、

1.!unzip -q CT2019_screening.zip” を入力し、再生ボタン

2.しばらくして、“更新”ボタンを押すと、展開したあとのファイルが見える



データの中身



- 訓練データ、バリデーションデータ、テストデータが、それぞれ別のディレクトリ
- クラス (NORMAL, PNEUMONIA) がそれぞれ別のディレクトリ

学習を実行してみる

- ・下のとおりに入力されているので、各セルで再生ボタンを押す。
- ・初回は、事前学習モデルのダウンロードがはじまり、学習がスタートする

```
[2] 1 import train
```

```
[3] 1 datadir = "chest_xray_exe"
2 label_set = ["NORMAL", "PNEUMONIA"]
3 params = {
4     "epochs":5
5 }
6 final_model, train_loss, val_loss = train.train_and_test(datadir, label_set,params)
```

学習の様子

- Lossが徐々に下がり、Acc (Accuracy)が徐々に上昇していることが見える。
- Epochが5なので、5回繰り返しが行われている
- 指定していないハイパーパラメータは、プログラムがデフォルトで定めた値になっている。

```
[2] 1 import train  
  
[3] 1 datadir = "chest_xray_exe"  
2 label_set = ["NORMAL", "PNEUMONIA"]  
3 params = {  
4     "epochs":5  
5 }  
6 final_model, train_loss, val_loss = train.train_and_test(datadir, label_set,params)  
  
↳ Settings:  
    Device: cuda  
    Batch size: 64  
    Epochs: 5  
    Learning rate: 0.0001  
    Momentum(SGD): 0.9  
    Step size for LR: 5  
    Gamma for LR: 0.1  
    Pretrained model: Use  
    Train Mode: Fine Tuning  
  
train: 249  
val: 16  
test: 30  
Epoch:0/4  train Loss: 0.7238 Acc: 0.5181 Time: 6.9290  val Loss: 0.7677 Acc: 0.3750 T  
Epoch:1/4  train Loss: 0.6755 Acc: 0.5823 Time: 6.8635  val Loss: 0.7137 Acc: 0.5000 T  
Epoch:2/4  train Loss: 0.6183 Acc: 0.6627 Time: 6.7844  val Loss: 0.7093 Acc: 0.3750 T  
Epoch:3/4  train Loss: 0.5673 Acc: 0.7470 Time: 6.8527  val Loss: 0.7229 Acc: 0.5625 T  
Epoch:4/4  train Loss: 0.5174 Acc: 0.8273 Time: 6.7932  val Loss: 0.7194 Acc: 0.5625 T  
  
Training complete in 0m 36s  
Best val loss: 0.7093  
On Test: Loss: 0.7553 Acc: 0.4000
```

学習済みモデルの保存。 学習の様子を可視化。

- 学習したモデルは保存することができる。以下のようなことが可能になる。
 - 複数の学習結果の比較
 - このモデルを利用して、テストデータでの精度を確認する
 - このモデルをもとに、更に学習を進める

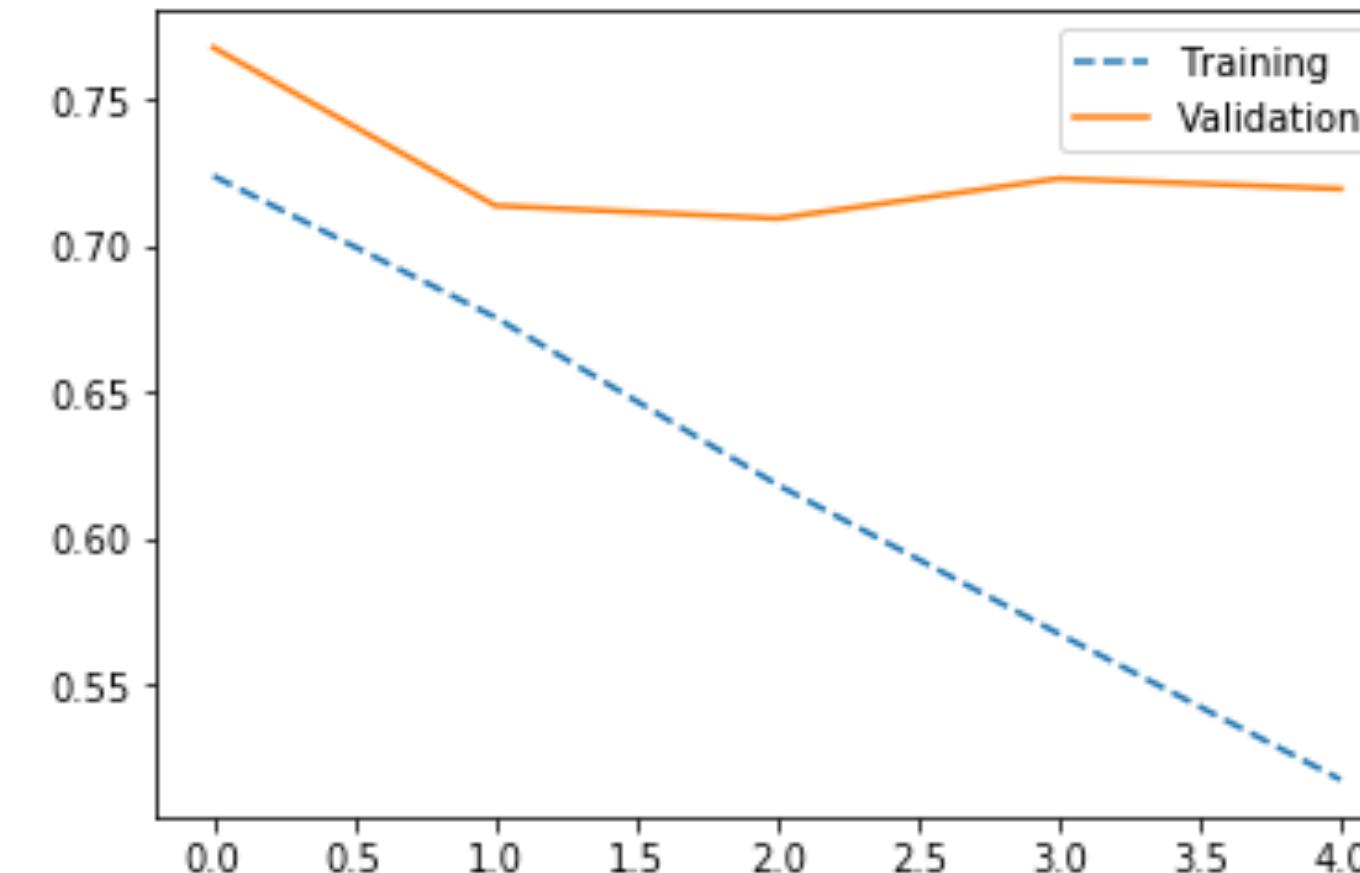
[4]

```
1 # バリデーションデータで一番ロスの少なかったモデルを保存  
2 import torch  
3 model_file_name = "best_model.torch"  
4 torch.save(final_model.state_dict(), model_file_name)
```

[5]

```
1 # 学習の様子を描画  
2 import matplotlib.pyplot as plt  
3 p1 = plt.plot(list(range(len(train_loss))), train_loss, linestyle="dashed")  
4 p2 = plt.plot(list(range(len(val_loss))), val_loss, linestyle="solid")  
5 plt.legend((p1[0], p2[0]), ("Training", "Validation"), loc=1)
```

⇨ <matplotlib.legend.Legend at 0x7fa5d4a297f0>

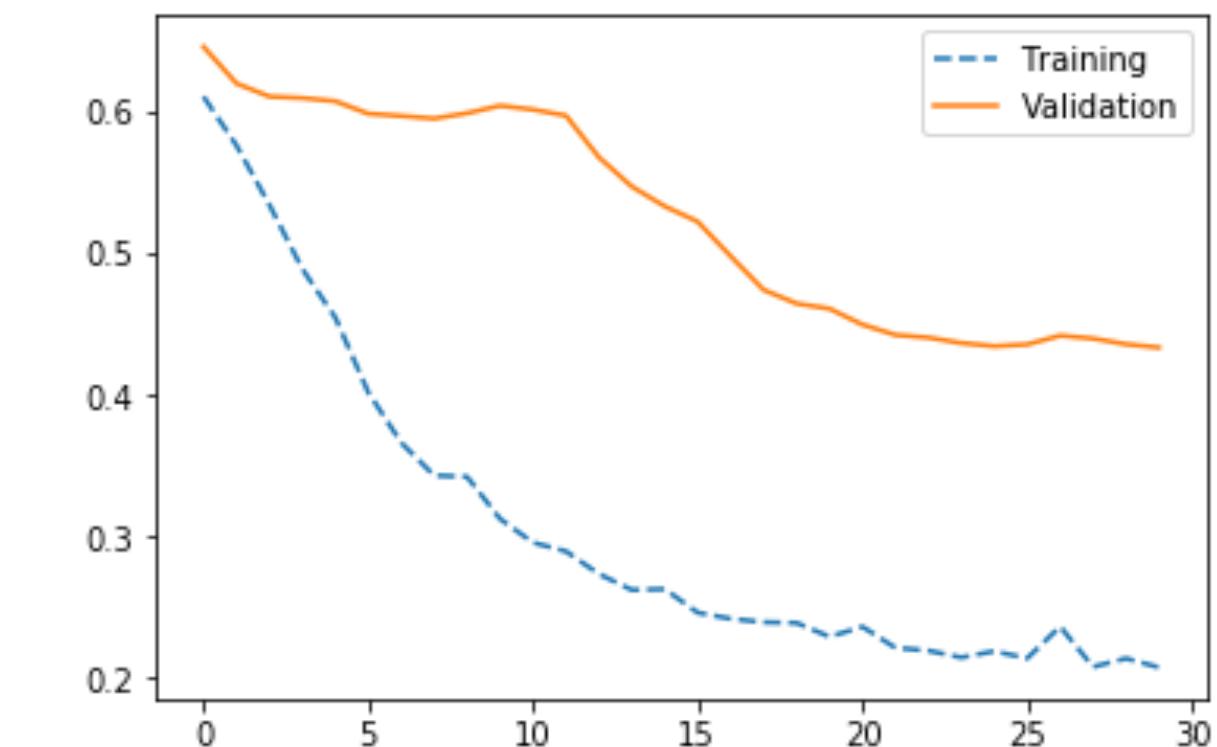


ハイパー parameter を変えて、学習を進めてみる

```
1 datadir = "chest_xray_exe"  
2 label_set = ["NORMAL", "PNEUMONIA"]  
3 params = {  
4     "epochs":30,  
5     "batch_size":64,  
6     "lr":0.0001,  
7     "momentum":0.9,  
8     "pretrained":True,  
9     "train_mode":"FT",  
10    "step_size":8,  
11    "gamma":0.5  
12}  
13 final_model, train_loss, val_loss = train.train_and_test(datadir, label_set, params)  
14  
15 # バリデーションデータで一番ロスの少なかったモデルを保存  
16 import torch  
17 model_file_name = "best_model.torch"  
18 torch.save(final_model.state_dict(), model_file_name)  
19  
20 # 学習の様子を描画  
21 import matplotlib.pyplot as plt  
22 p1 = plt.plot(list(range(len(train_loss))), train_loss, linestyle="dashed")  
23 p2 = plt.plot(list(range(len(val_loss))), val_loss, linestyle="solid")  
24 plt.legend((p1[0], p2[0]), ("Training", "Validation"), loc=1)
```

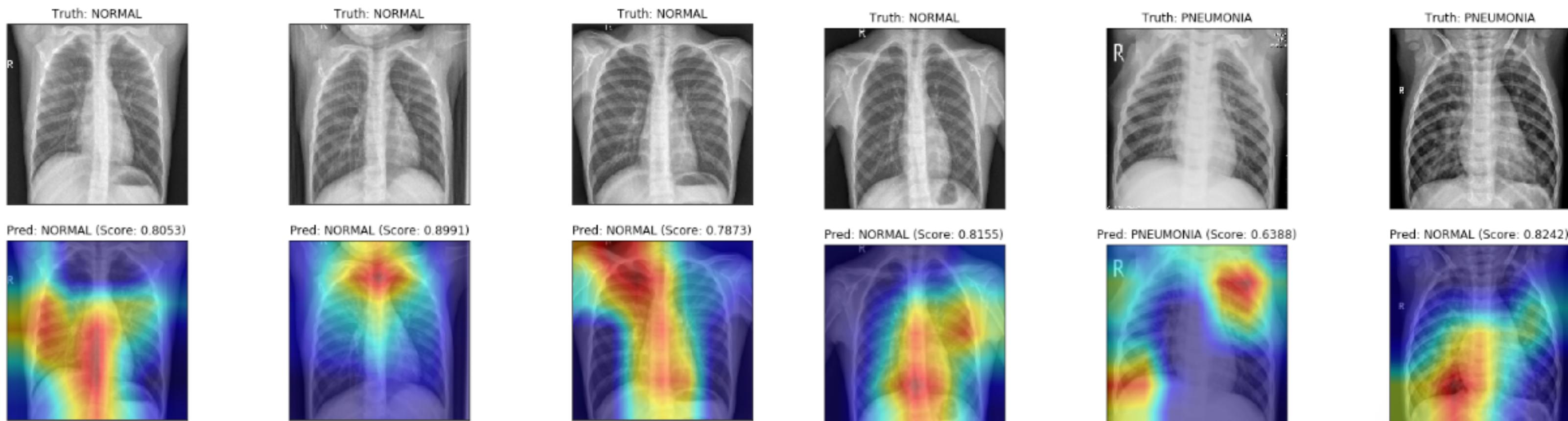
```
Epoch:27/29 train Loss: 0.2082 Acc: 0.9839 Time: 6.8097  
Epoch:28/29 train Loss: 0.2141 Acc: 0.9880 Time: 6.8494  
Epoch:29/29 train Loss: 0.2078 Acc: 0.9839 Time: 6.8540
```

```
Training complete in 3m 37s  
Best val loss: 0.4337  
On Test: Loss: 0.5516 Acc: 0.6333  
<matplotlib.legend.Legend at 0x7fa5d1df8518>
```



計算機は何を見て判別しているのか (GradCAMの例)

```
[35] 1 import gradcam  
2 datadir = "chest_xray_exe"  
3 gradcam.gradcam("best_model.torch", datadir)
```





育ててみよう



画像の一部に注釈をつける

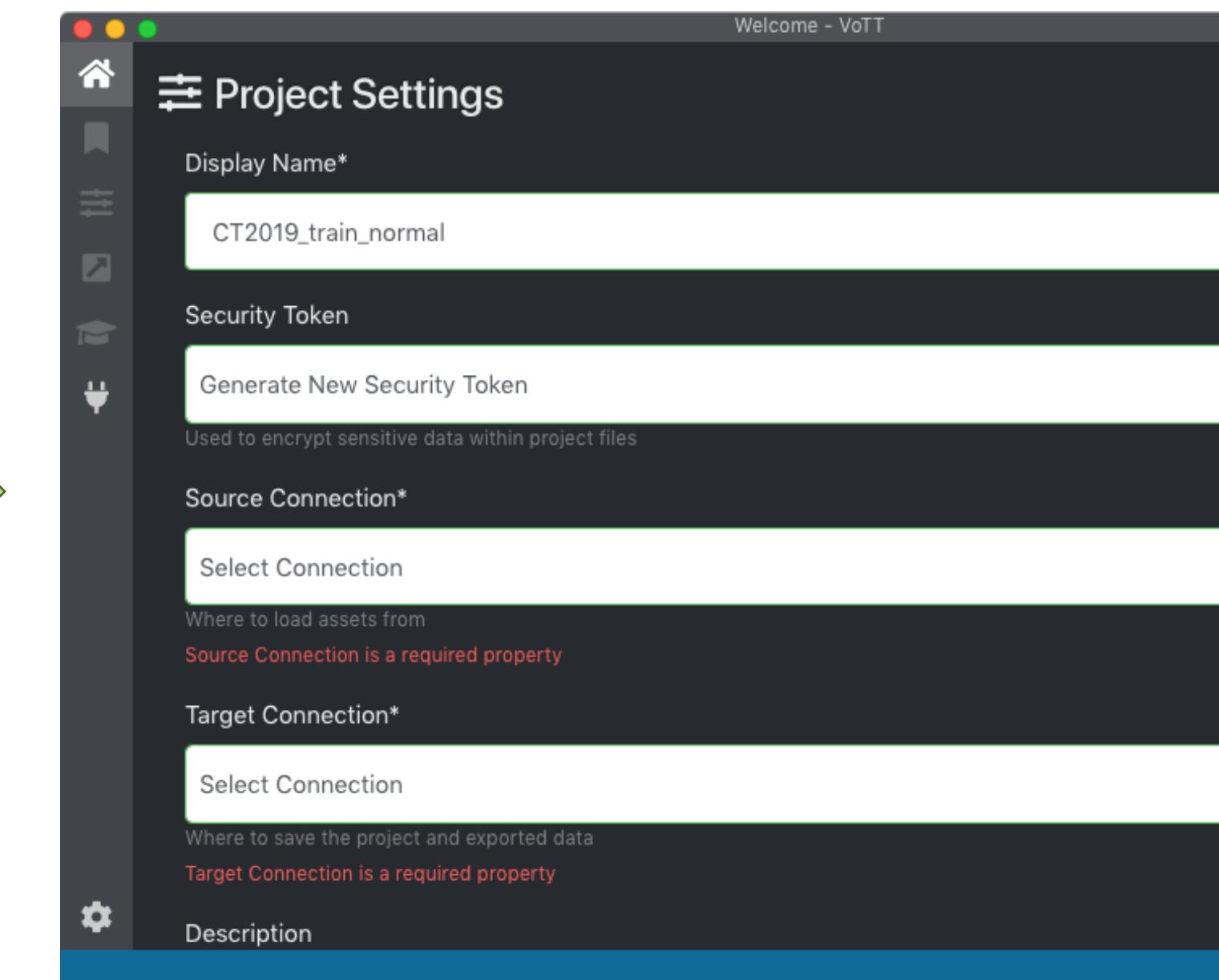
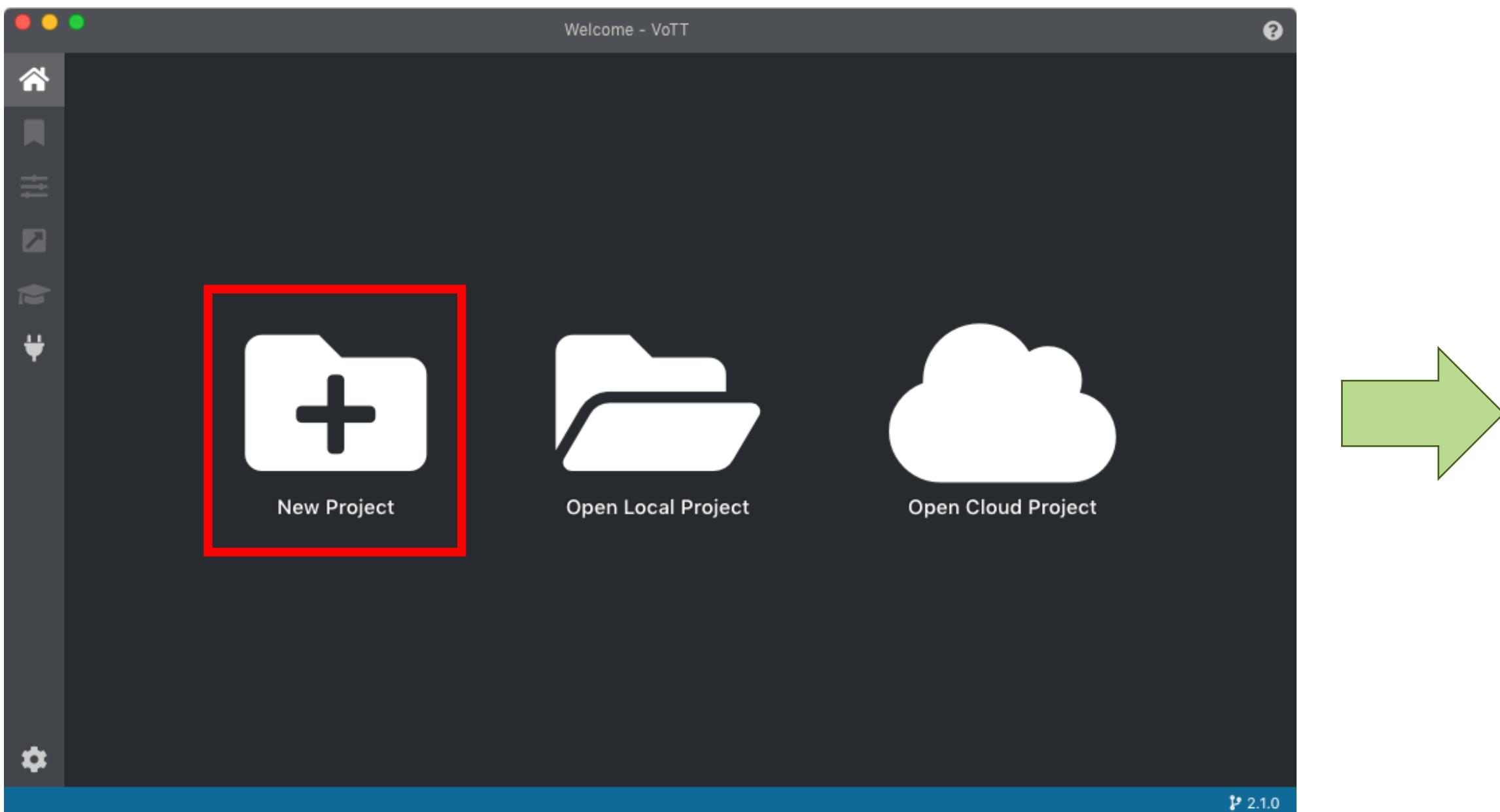
- ・部分的に取り出したり、マーク付けるだけなら、ペイントやPreviewでも可能だが、計算機に入力しやすくしたり、ラベル（物体名）をつけたりするために、アノテーションソフトウェアが開発されている
- ・今回はVoTT (<https://github.com/microsoft/VoTT>) を利用する
- ・インストール方法
 - ・<https://github.com/Microsoft/VoTT/releases>にOSごとのバイナリがあるので、環境にあったものをダウンロードしてインストール
 - ・欠点：DICOMは直接使えないでの、png, jpeg等に変換する必要あり

▼ Assets 5	
	vott-2.1.0-darwin.dmg
	vott-2.1.0-linux.snap
	vott-2.1.0-win32.exe
	...



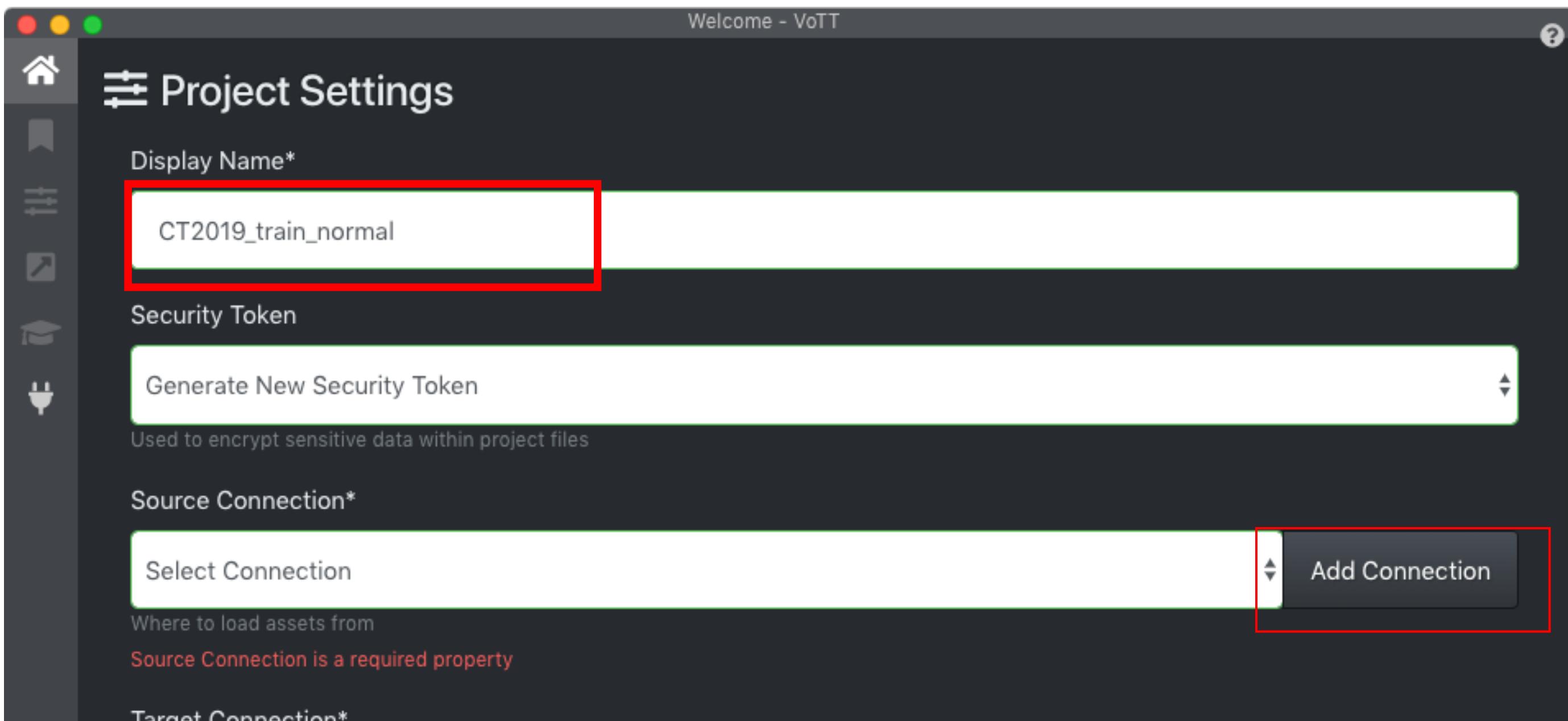
VoTT起動画面

- 起動後（左）New Projectを選択すると右の画面になる



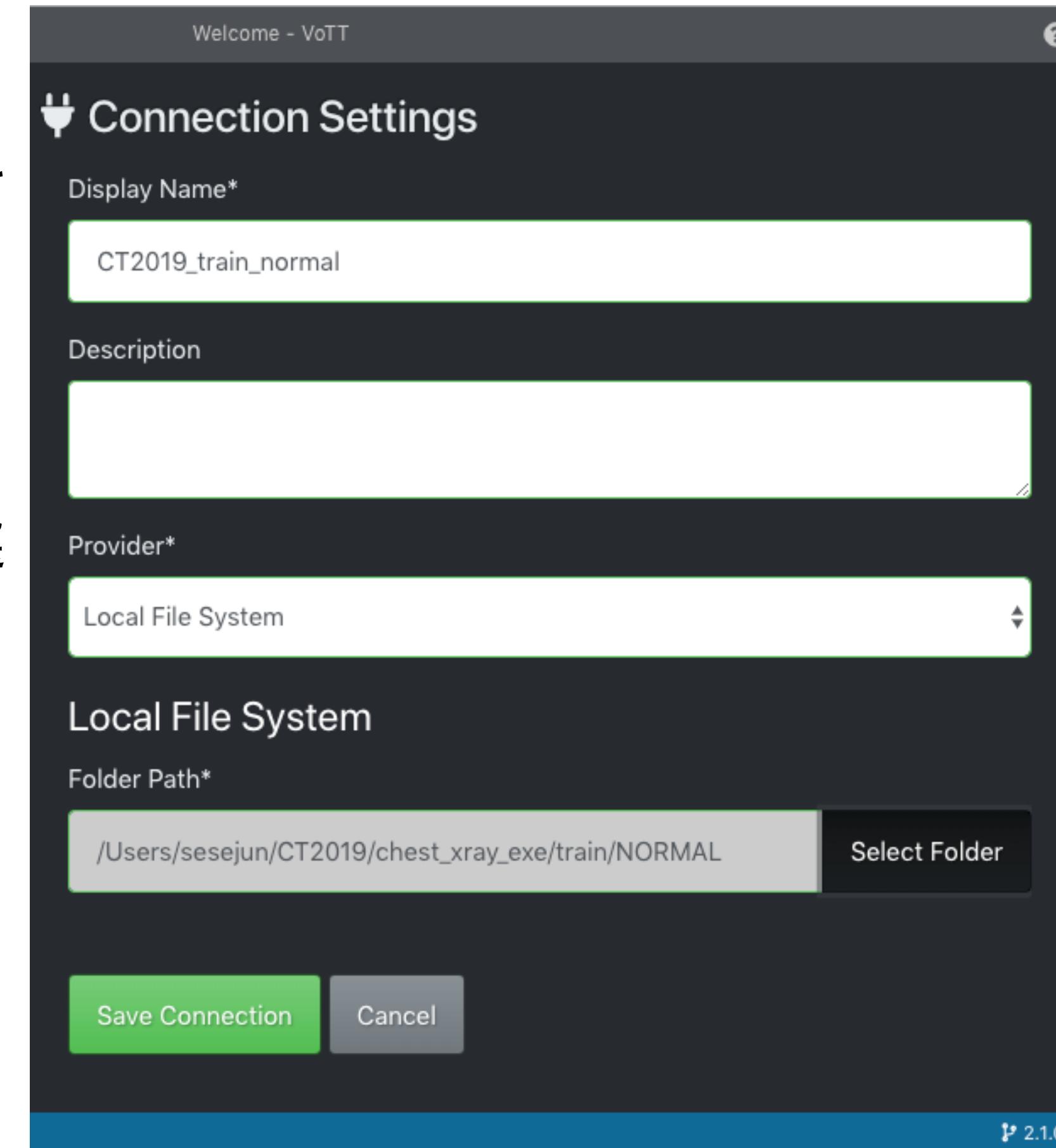
プロジェクトの作成（開始）

- Display Name を入力。プロジェクト名になる。
 - ここでは CT2019_train_normal とした
- 次に、右のAdd Connection をクリック。読み出す画像の場所を指定する
 - 次のページ



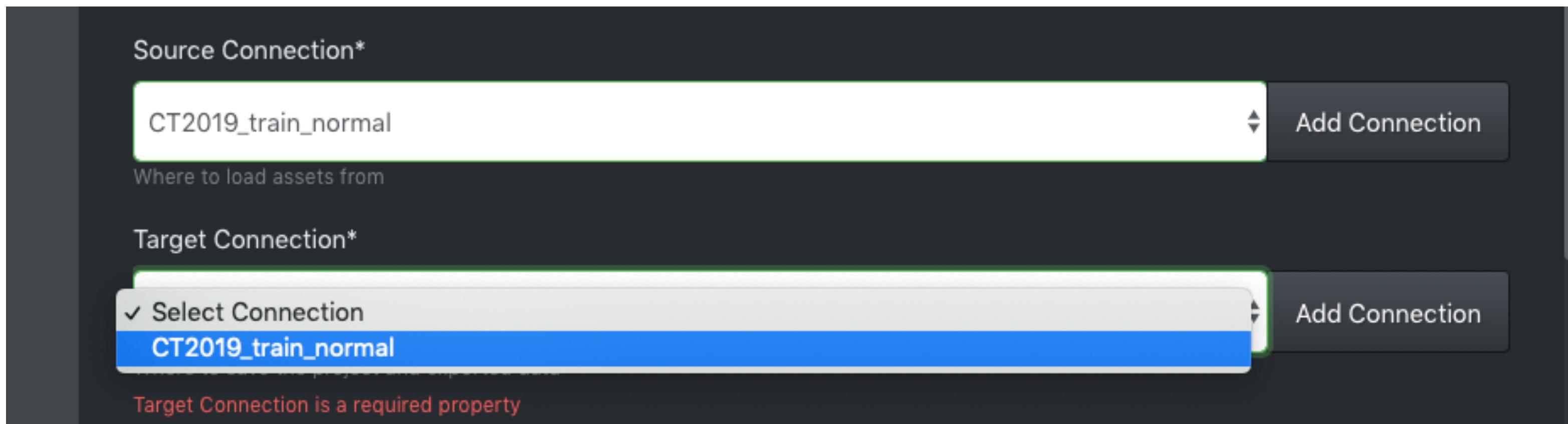
データの入ったディレクトリを指定（1）

- Add Connection を押すと、右の画面になる。
- Display Name に 任意の名前を指定
 - ここではプロジェクト名と同じに CT2019_train_normal とした
- “Provider” で “Local File System” を選択。
- ファイルが選択できるようになるので、Select Folder から、”chest_xray_exe/train/NORMAL” を選択。
- 緑のSave Connectionを押す。



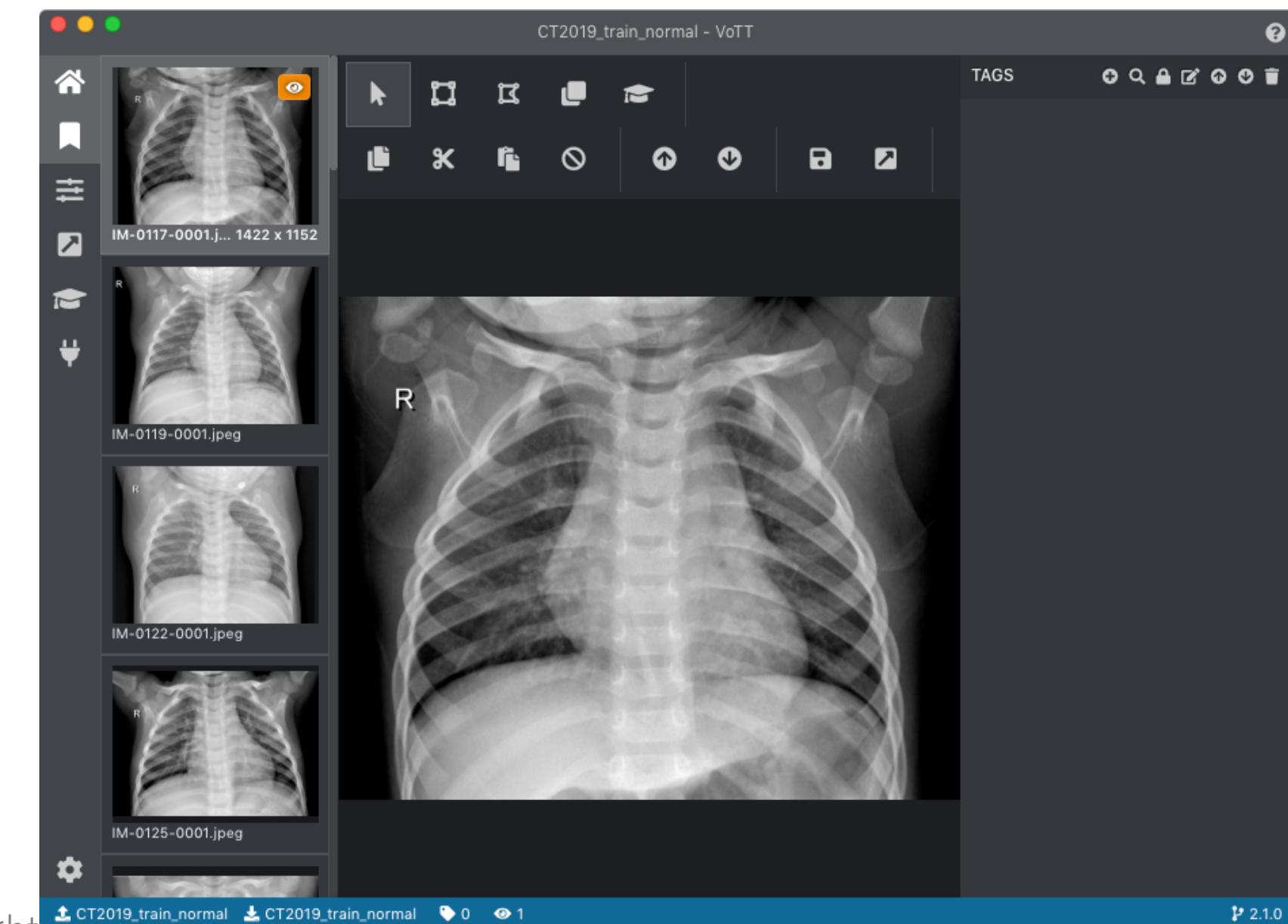
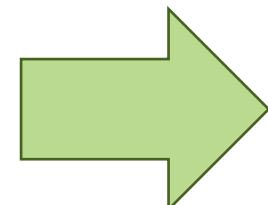
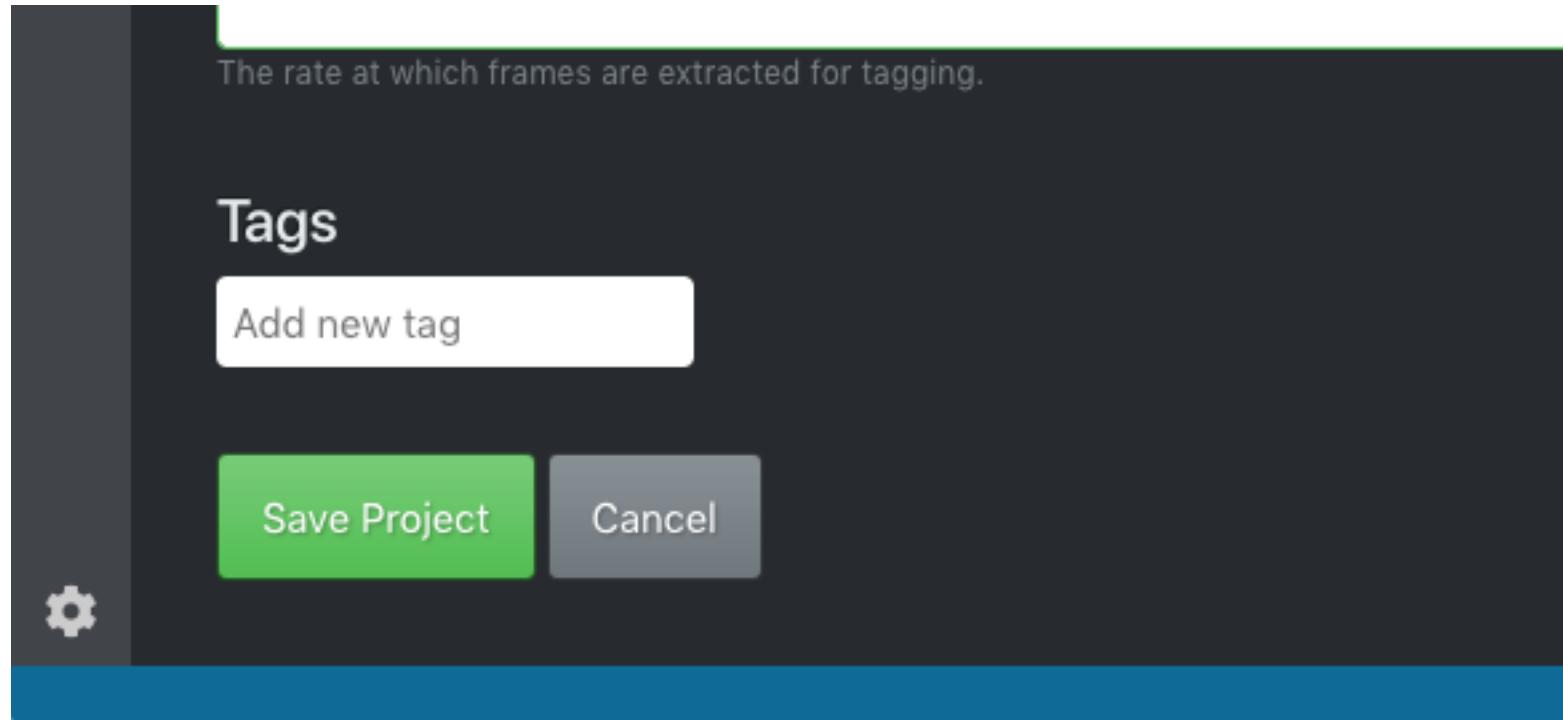
データの入ったディレクトリを指定（2）

- ・元の”Project Settings” に戻る
- ・”Source Connection” (データの読み込み場所) と “Target Connection” (結果の出力場所)共に、前ページで作成した “CT2019_train_normal” を指定



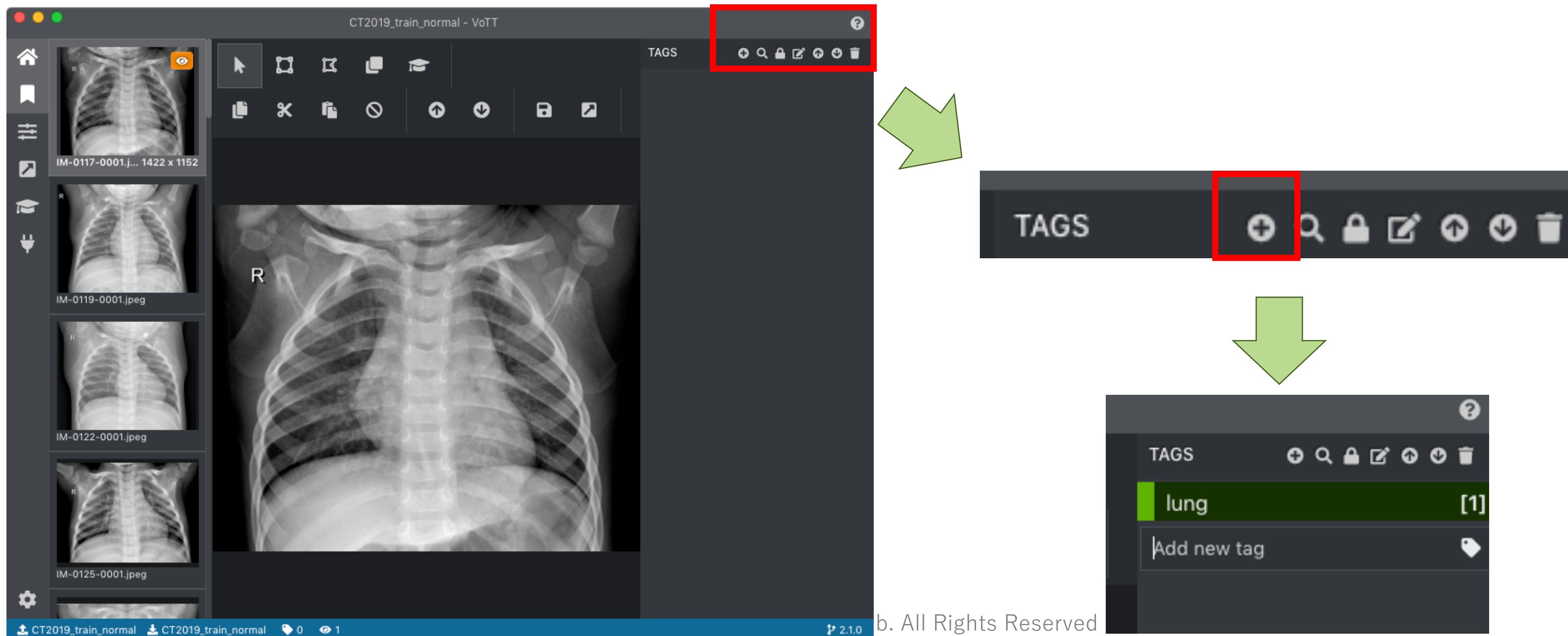
プロジェクトの作成（終了）

- ・画面を最後までスクロールすると、緑の Save Project が現れるので、クリック。
- ・画像が読み込まれる。



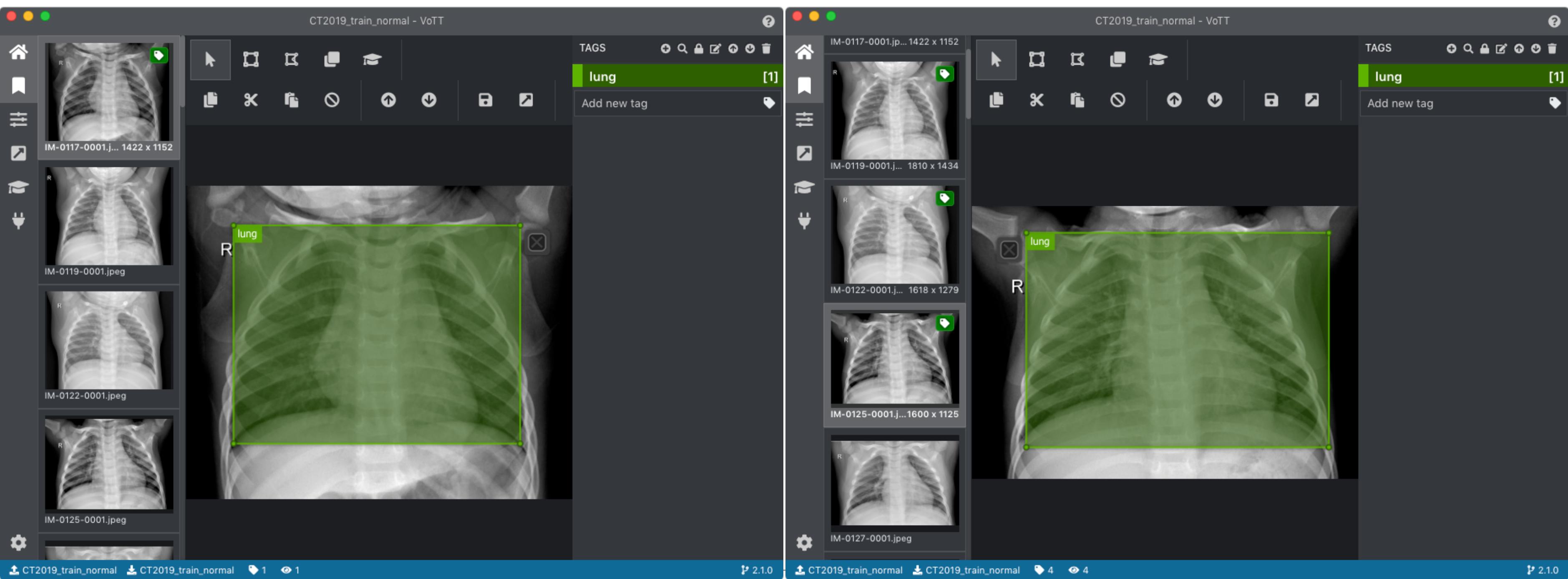
ラベルの作成

- 右上のツールバーの「+」マークを押してラベル名を作成
- 今回は肺の領域を囲みたいので“lung” ラベルを作成



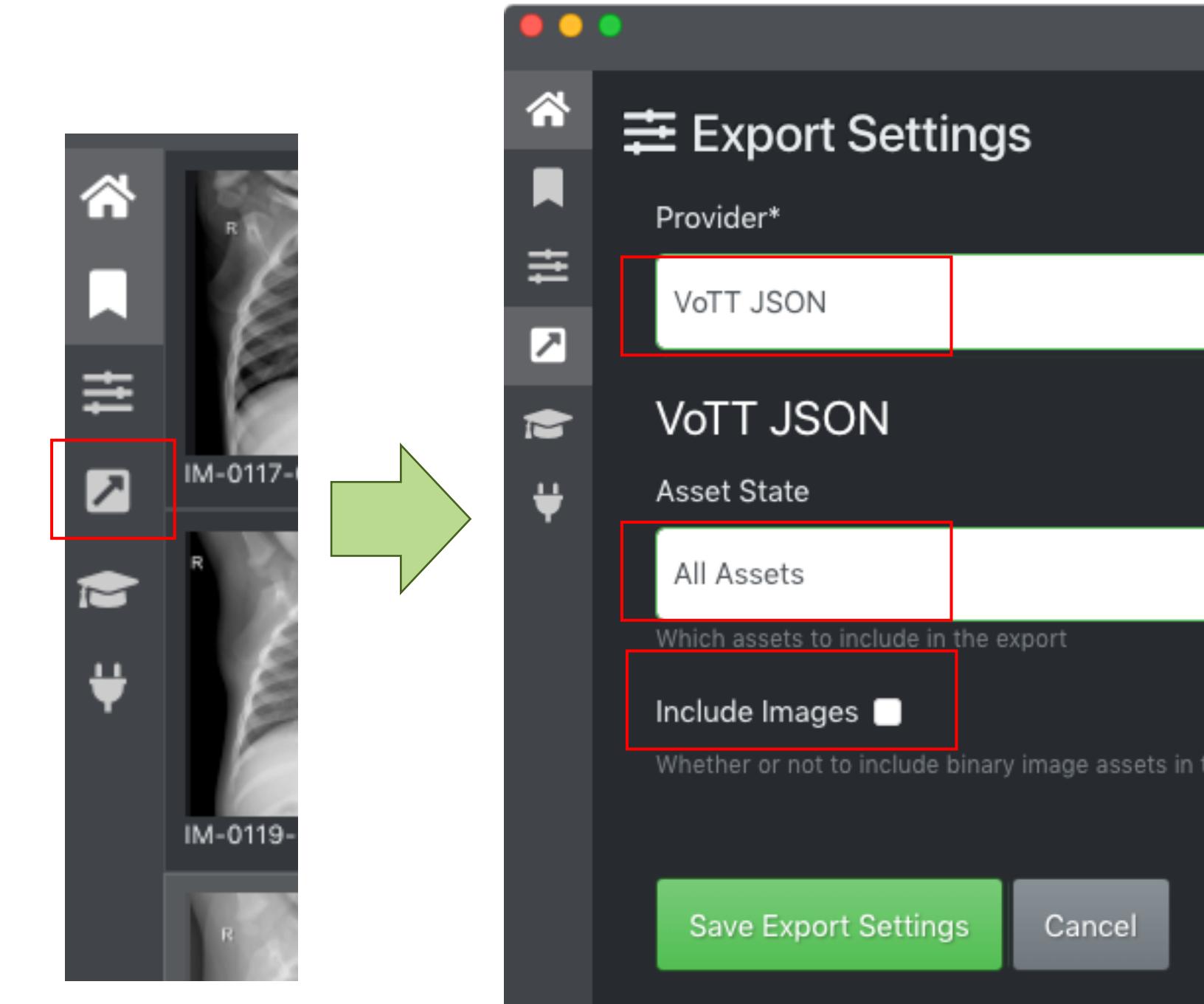
領域の印をつける

- マウスで領域選択をして、右上のラベル名を押すと、領域にラベルが付く。

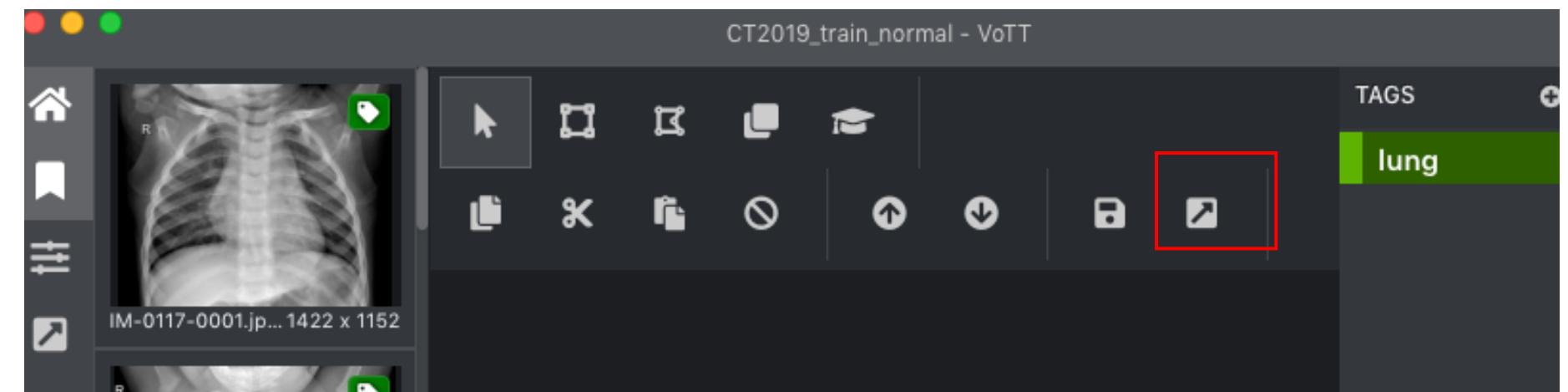


出力の準備

- ・通常の保存は、のマーク。
- ・今回は、領域情報を計算機で加工しやすいように、出力する
- ・左のツールバーから「矢印」をクリック
- ・Export setting が開くので、以下の3つを実施
 - ・Provider を VoTT JSON
 - ・Asset State を All Assets
 - ・Including Images のチェックボックスを外す
- ・緑のSave Export Settingsをクリック



実際の出力



- ・上のツールバーの「右上矢印」をクリックすると、Target Connectionで指定したディレクトに出力される
 - ・前ページの準備は1度だけ行えばよい。

肺炎画像に対してアノテーションを実施

- ・新規プロジェクト"CT2019_train_pneumonia"を作成。
- ・新規コネクション"CT2019_train_pneumonia"を、
chest_xray_exe/train/PNEUMONIA を指定して作成
- ・以下、同様の手順

アノテーション部分だけの切り出しの実施

- Collaboratory で実施。
- chest_xray_exe のディレクトリを圧縮して、アップロード
- 展開を行う（下）
- その後、右の様に実行すると chest_xray_exe 内にある画像を、アノテーションがあるものは、従って切り出し、chest_xray_crop に収める。

```
1 | !unzip -q chest_xray_exe.zip
```

```
[40] 1 import convert
[41] 1 labels = ["NORMAL", "PNEUMONIA"]
      2 convert.run("chest_xray_exe", "chest_xray_crop", labels)
      3
```

```
IM-0125-0001.jpeg {'height': 884.3652950310559, 'width': 1252
IM-0125-0001.jpeg {'height': 884.3652950310559, 'width': 1252}
```

切り出したデータをもとに学習

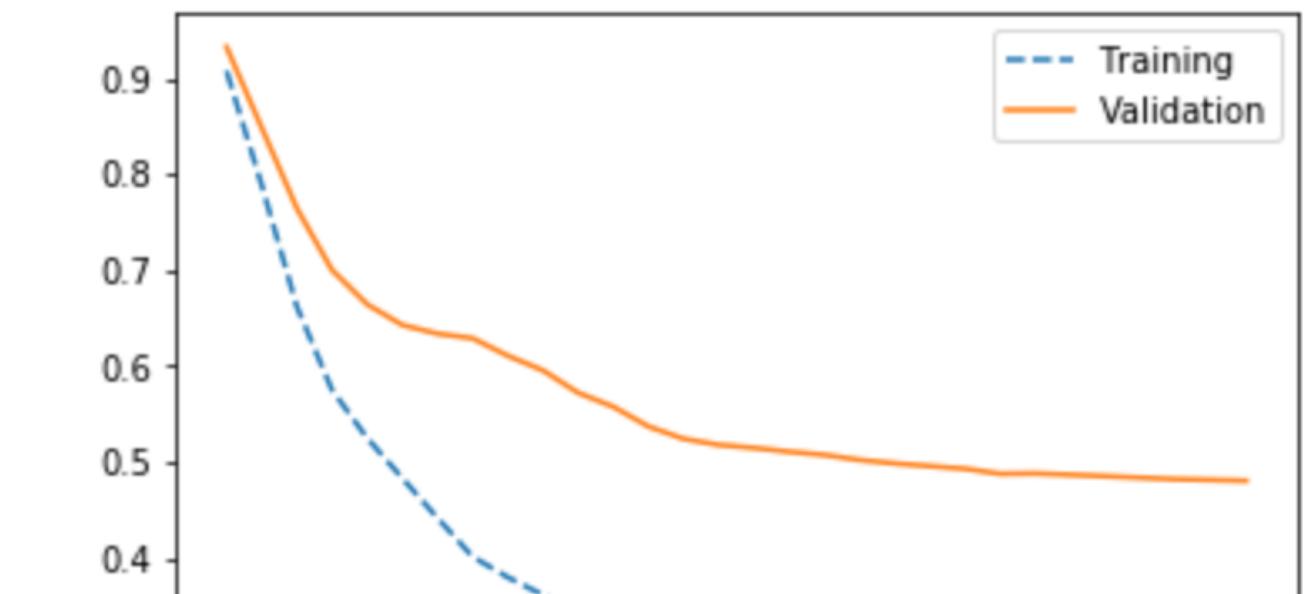
- 手順ははじめの学習と同じ。
- 利用するデータのディレクトリが異なる。
- 入力データでも、学習の様子が異なる

[49]

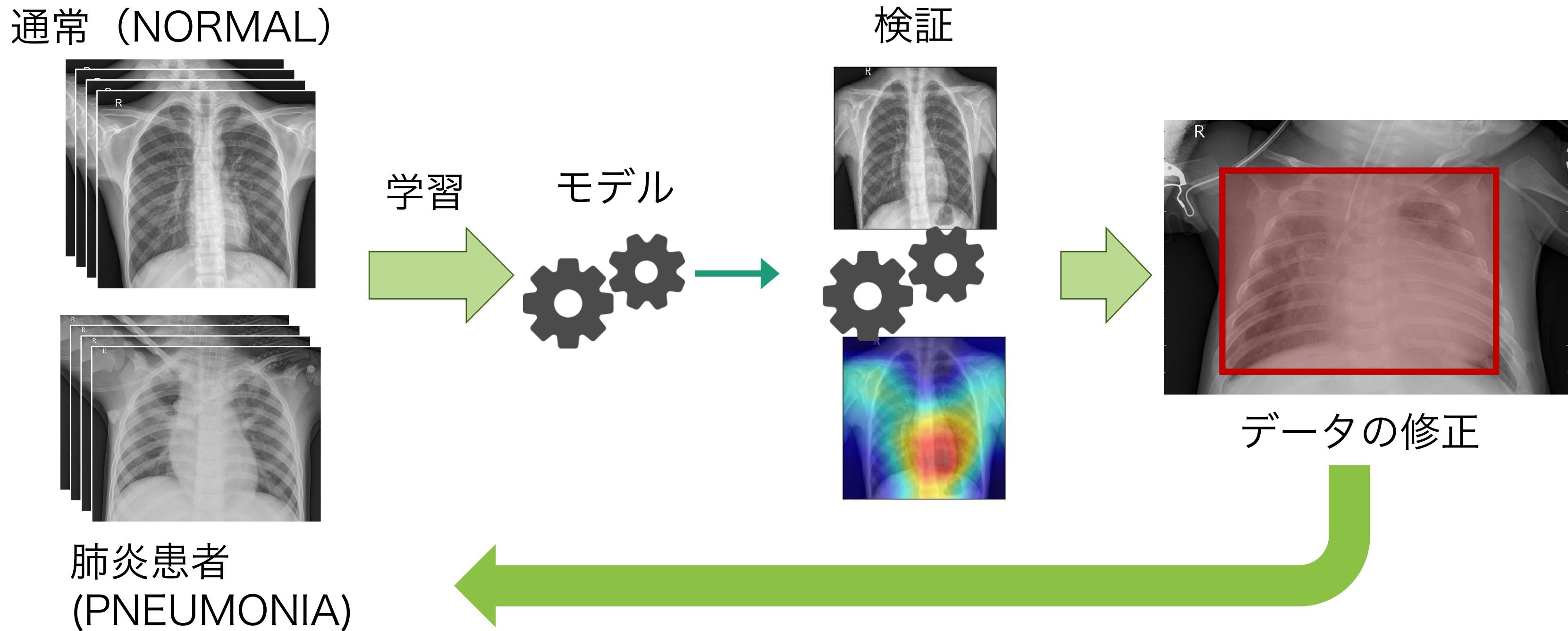
```
1 datadir = "chest_xray_crop"
2 label_set = ["NORMAL", "PNEUMONIA"]
3 params = {
4     "epochs":30,
5     "batch_size":64,
6     "lr":0.0001,
7     "momentum":0.9,
8     "pretrained":True,
9     "train_mode":"FT",
10    "step_size":8,
11    "gamma":0.5
12}
13 final_model, train_loss, val_loss = train.train_and_test(datadir, label_set, params)
14
15 # バリデーションデータで一番ロスの少なかったモデルを保存
16 import torch
17 model_file_name = "best_model_cropped.torch"
18 torch.save(final_model.state_dict(), model_file_name)
19
20 # 学習の様子を描画
21 import matplotlib.pyplot as plt
22 p1 = plt.plot(list(range(len(train_loss))), train_loss, linestyle="dashed")
23 p2 = plt.plot(list(range(len(val_loss))), val_loss, linestyle="solid")
```

```
. Epoch:27/29 train Loss: 0.2418 Acc: 0.9679 Time: 6.7538
Epoch:28/29 train Loss: 0.2482 Acc: 0.9639 Time: 6.8595
Epoch:29/29 train Loss: 0.2434 Acc: 0.9639 Time: 6.8418
```

```
Training complete in 3m 37s
Best val loss: 0.4804
On Test: Loss: 0.4889 Acc: 0.7333
<matplotlib.legend.Legend at 0x7fa5b8a34940>
```



データやモデルを「育てる」プロセス





まとめ



はじめてみよう！

- ・人工知能を始めるツールは揃っている
- ・簡単な問題かつ少数のデータから始めよう
 - ・徐々に、データ量を増やしたり、難しい問題へと発展させる
- ・人工知能はプログラムを組むだけでなく、データやモデルを育てるこども重要
- ・身の回りの問題からはじめてみると「モノの新しい見方」が出てきます

