機械学習のための
データ前処理の基本と実践法
11

胸部X線画像からの肺炎のAI診断

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. データの準備
2. モデルの実装と学習
3. 学習結果の表示とモデルの評価

前回の復習

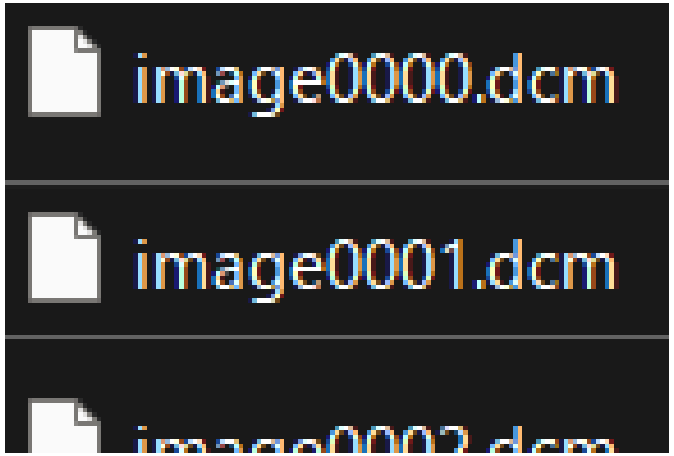
医用画像データの扱い

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

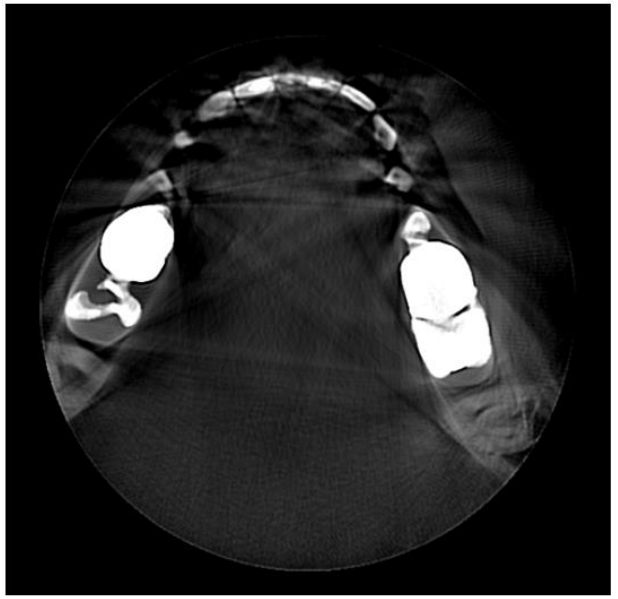
DICOM画像

- 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



歯科パノラマX線

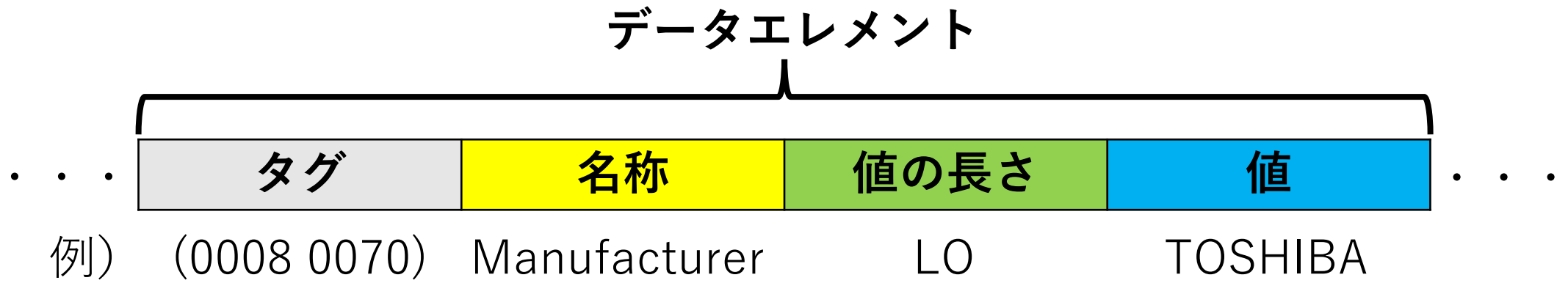
https://cdn.peraichi.com/userData/5fae282f-7dbc-45f7-b028-06410a0000ae/img/60b7209a478a4/original.png



CT体軸断面

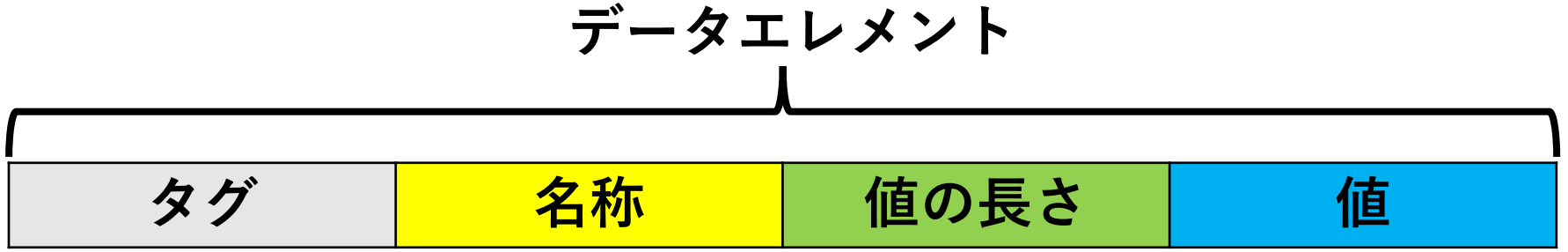
https://cdn.peraichi.com/userData/5fae282f-7dbc-45f7-b028-06410a0000ae/img/60bf05c21d84c/original.png

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

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

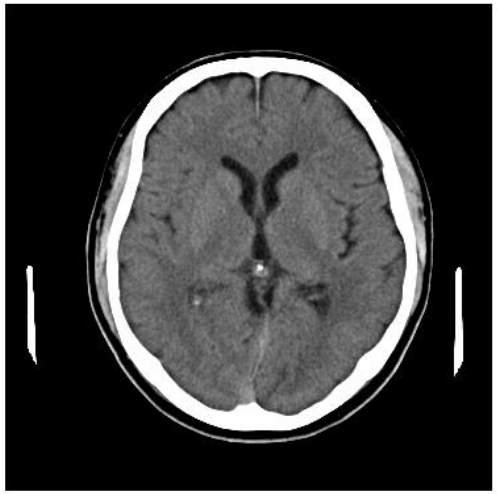
0018：検査情報　　0020：検査情報　　0028：画像情報

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

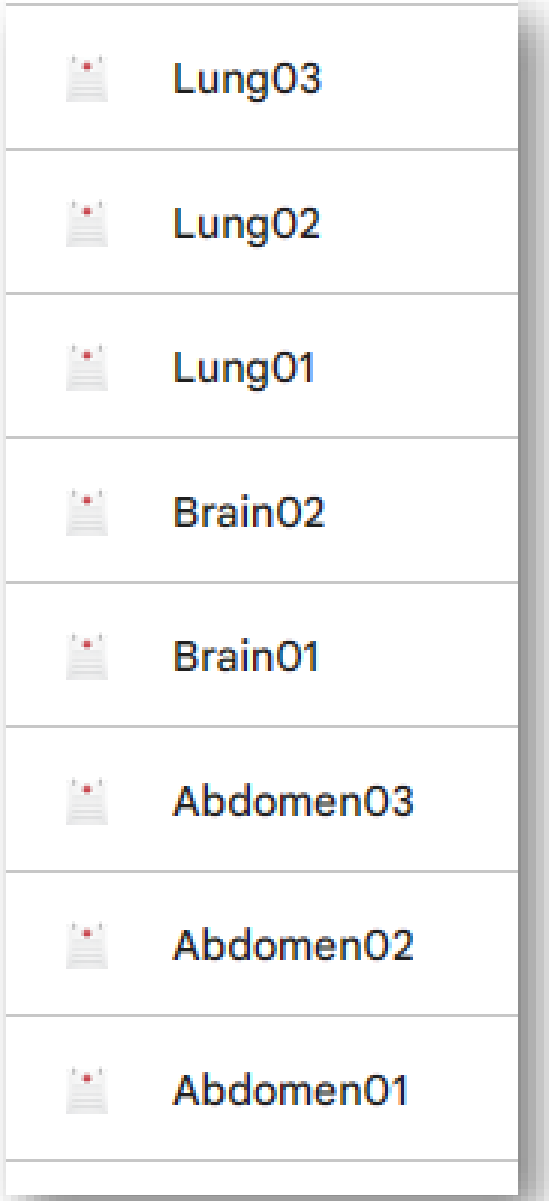
データセット



Brain01

Lung01

Abdomen01



データエレメントの取得と表示

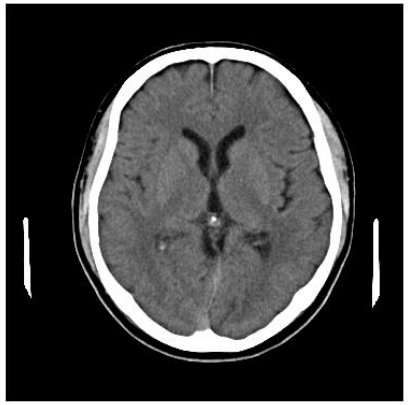
```
pn = data.PatientName
wc = data.WindowCenter
ww = data.WindowWidth
img = data.pixel_array
```

```
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

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()
```



1. データの準備

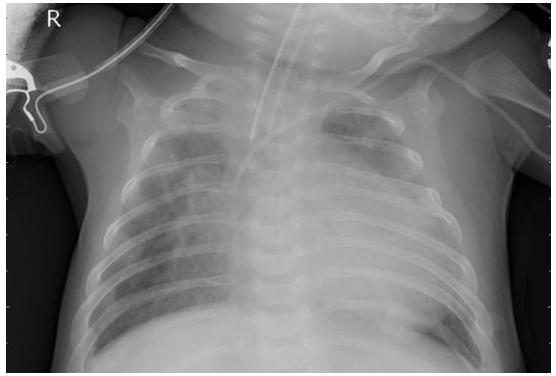
肺炎のAI診断に向けて

胸部X線画像

- 胸部X線画像から肺炎の有無を診断するAIモデルを構築する
- データは**Kaggle**の**Chest X-Ray Images**を用いる

https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

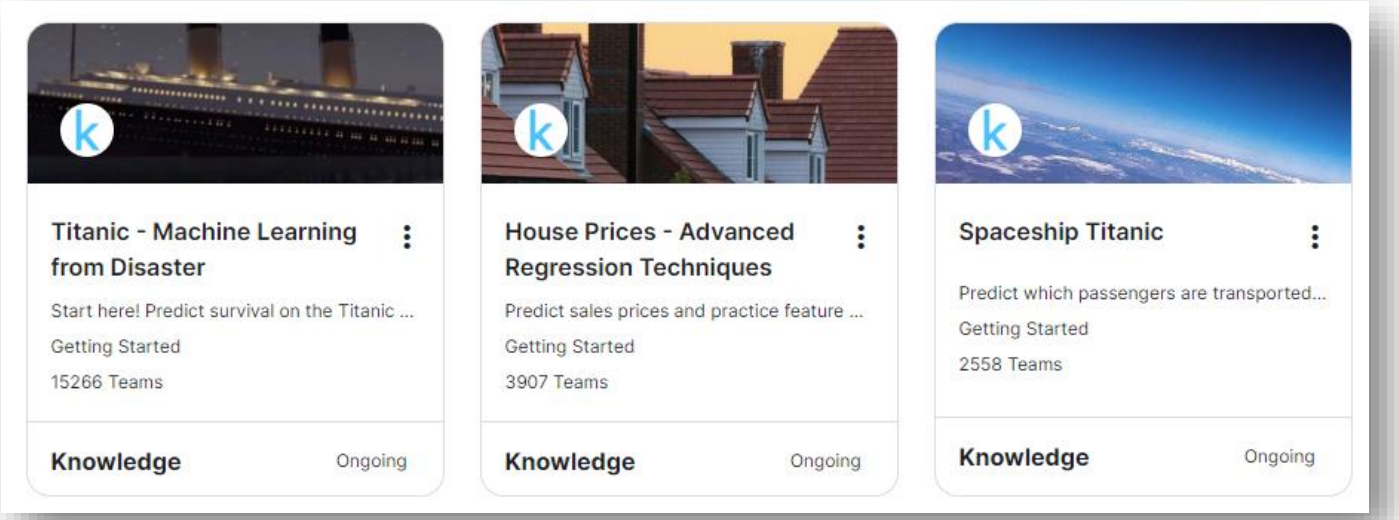

健常（NORMAL）




肺炎（Pneumonia）

Kaggle https://www.kaggle.com/

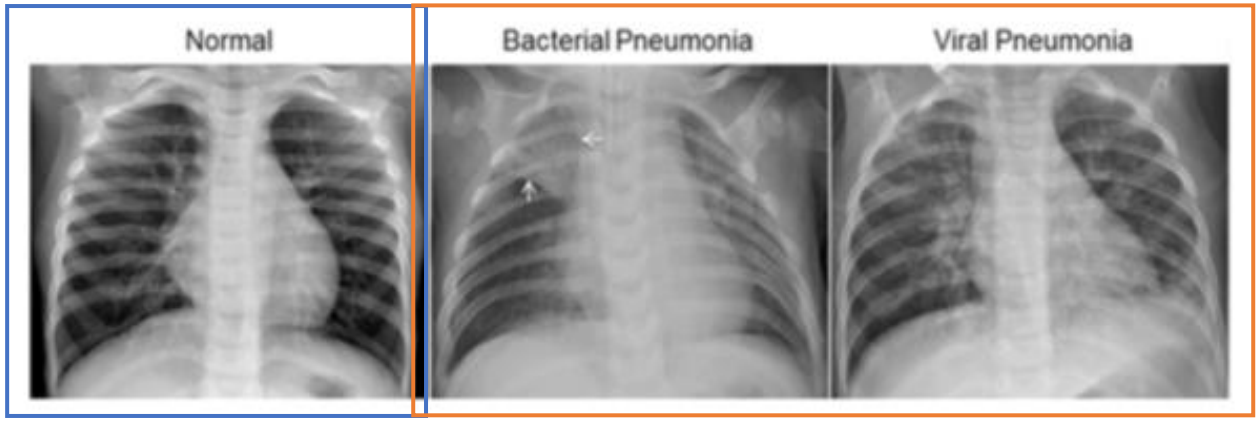
- AIのコンペティションを行うためのプラットフォーム
  ※AIのコンペティション＝AIモデルの精度を競い合う大会
- 多種多様なデータセットが提供されている
- AIの実践的な技術を身に着けたいときに有用なサービスのひとつ



Chest X-Ray Images (Pneumonia)

- 胸部X線画像（JPEG）に以下のラベルが付けられたデータセット

健康（Normal）
細菌性肺炎（Bacterial Pneumonia）
ウイルス性肺炎（Viral Pneumonia）

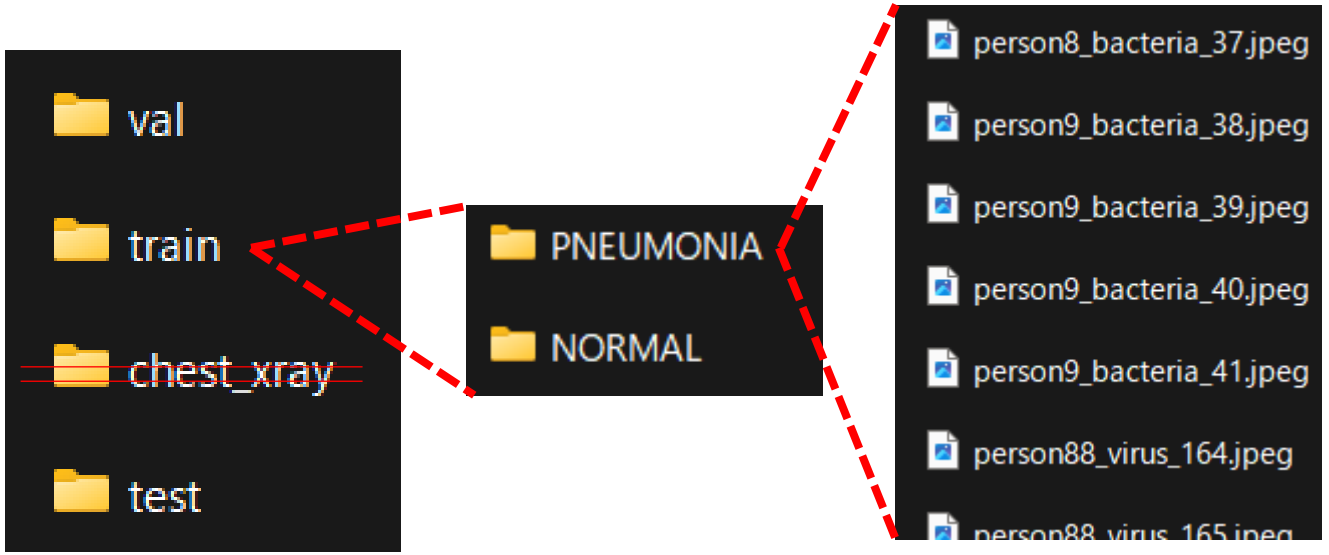


https://i.imgur.com/jZqpV51.png

- 健康はNORMAL、肺炎はPNEUMONIAとしてフォルダ分けされている

データセットの構成

train　学習用データセット
val　　検証用データセット
test　評価用データセット



	train	val	test
NORMAL	1,341	8	234
PNEUMONIA	3,875	8	390

データ数の内訳

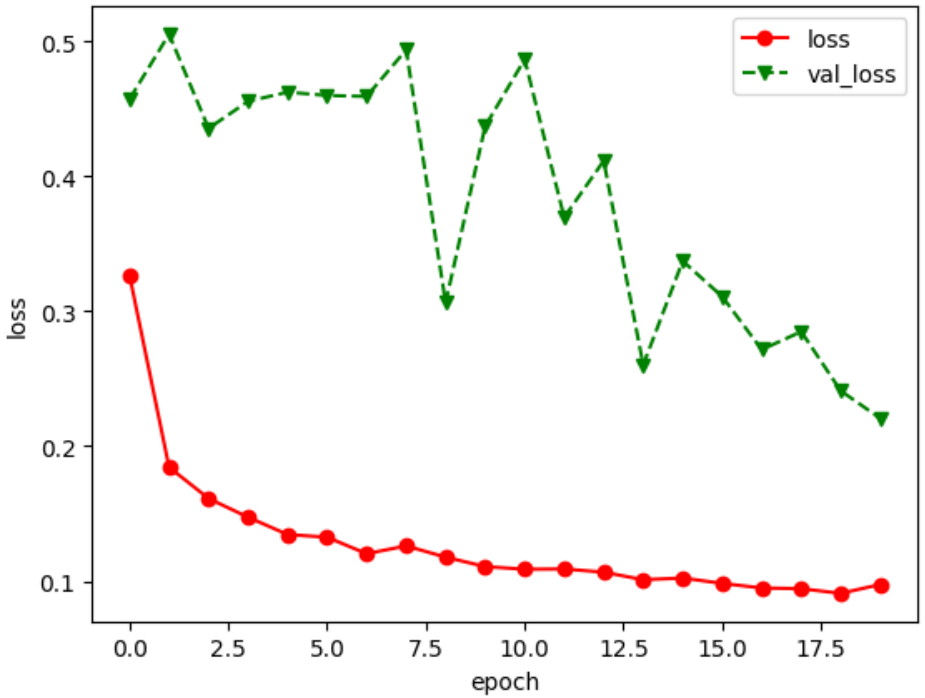
学習・検証・評価

- 一般的にすべてのデータセットを学習に用いることは滅多にない
- データセットは以下の3種類に分けられる

学習用　　AIモデルの学習に用いる
検証用　　AIモデルの学習に用いない、学習中のモデル検証等に用いる
評価用　　AIモデルの学習に用いない、AIモデルの評価に用いる
（**テスト用**）

※検証用＝評価用とすることもある

検証用データセットの役割



1. 学習中のAIモデルの精度確認
- 学習の成否がわかりやすくなる
- 過学習を検出できる

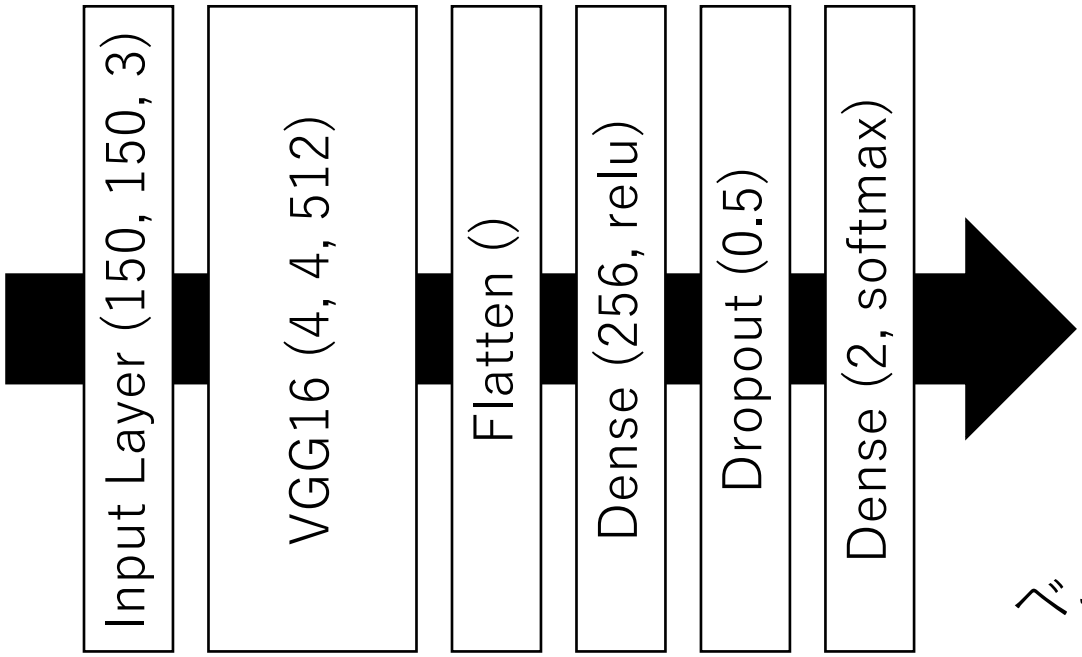
2. ハイパーパラメータのチューニング
- 学習中にハイパーパラメータを調節できる
  例）学習率、最適化アルゴリズム特有のハイパーパラメータなど

2. モデルの実装と学習

構築を最適化しよう

モデルの構築と方針

- 第9回で紹介した手法と実装コードをベースに実験を行う
  VGG16のファインチューニング
- その後、誤差グラフなどを見て各設定やAIモデルの構築を見直す
  ここがAIエンジニアの腕の見せ所になる



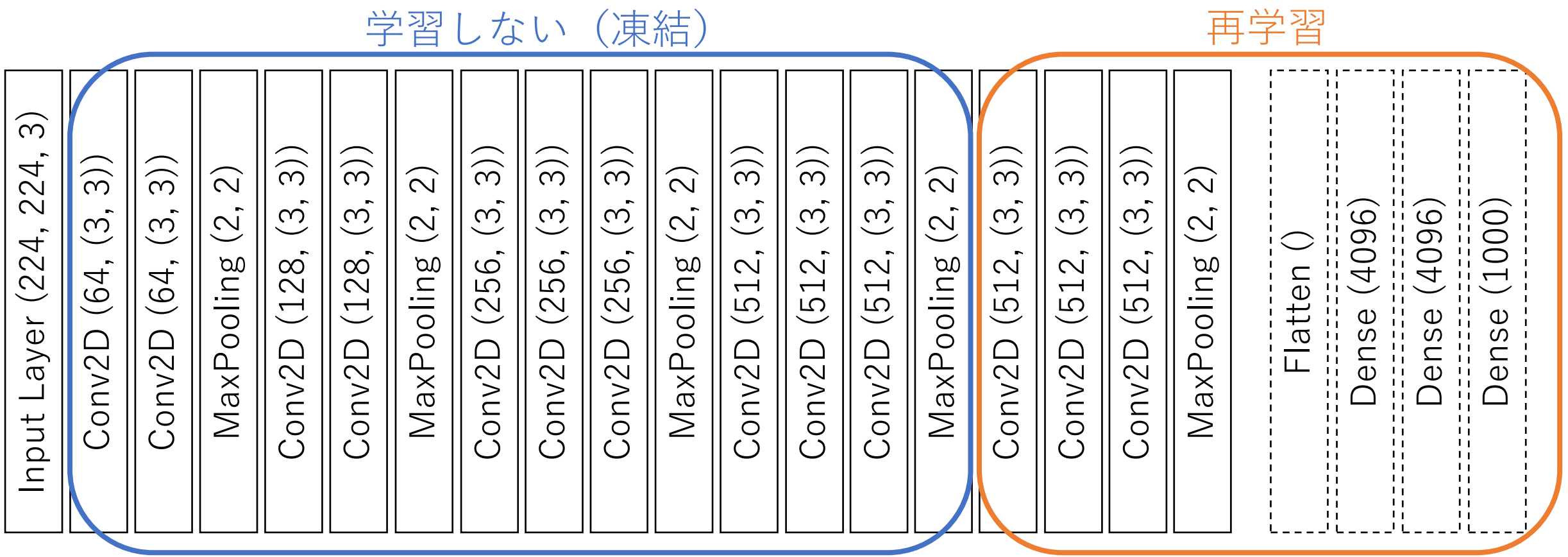
ベースのモデル構造

実験設定一覧

	実験1	実験2	実験3
入力サイズ	150×150	150×150	150×150
層の凍結	15層まで	15層まで	なし（全層学習する）
データ拡張	水平反転 シアー 拡大縮小	水平反転 回転 移動 拡大縮小 明るさ変化	水平反転 回転 移動 拡大縮小 明るさ変化
学習回数	20	40	40
バッチサイズ	32	32	32
最適化手法	SGD(0.001, 0.1)	SGD(0.001, 0.1)	SGD(0.001, 0.1)

※データ拡張の細かいパラメータは省略しています

実験1,2のファインチューニング



実験3のファインチューニング

層の凍結

```
# VGG16モデルの上位15層のパラメータを凍結
for layer in vgg_model.layers[:15]:
    layer.trainable = False
```

- VGG16の15層までは再学習しないようにするコード
- すべての層を再学習する場合は、この2行をコメントアウトすればよい
  デフォルトではlayer.trainable=Trueとなっているため

データ拡張

```
# 学習データのデータ拡張を設定
train_datagen = ImageDataGenerator(
        rescale = 1.0 / 255,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = True)
```

- 必要なデータ拡張があればここで追加する
- 不必要なデータ拡張はコメントアウトすることで無効化できる

（補足）グレースケール画像の扱い

- グレースケール画像は1チャネルの画像
- 学習済みモデルが3チャネル画像を学習していた場合どうする？

⬇

- モデルに入力するときに3チャネルに変換する
  変換といっても1チャネルを3チャネルに複製しているだけ
- OpenCVのcvtColorを使えば簡単に実装できる

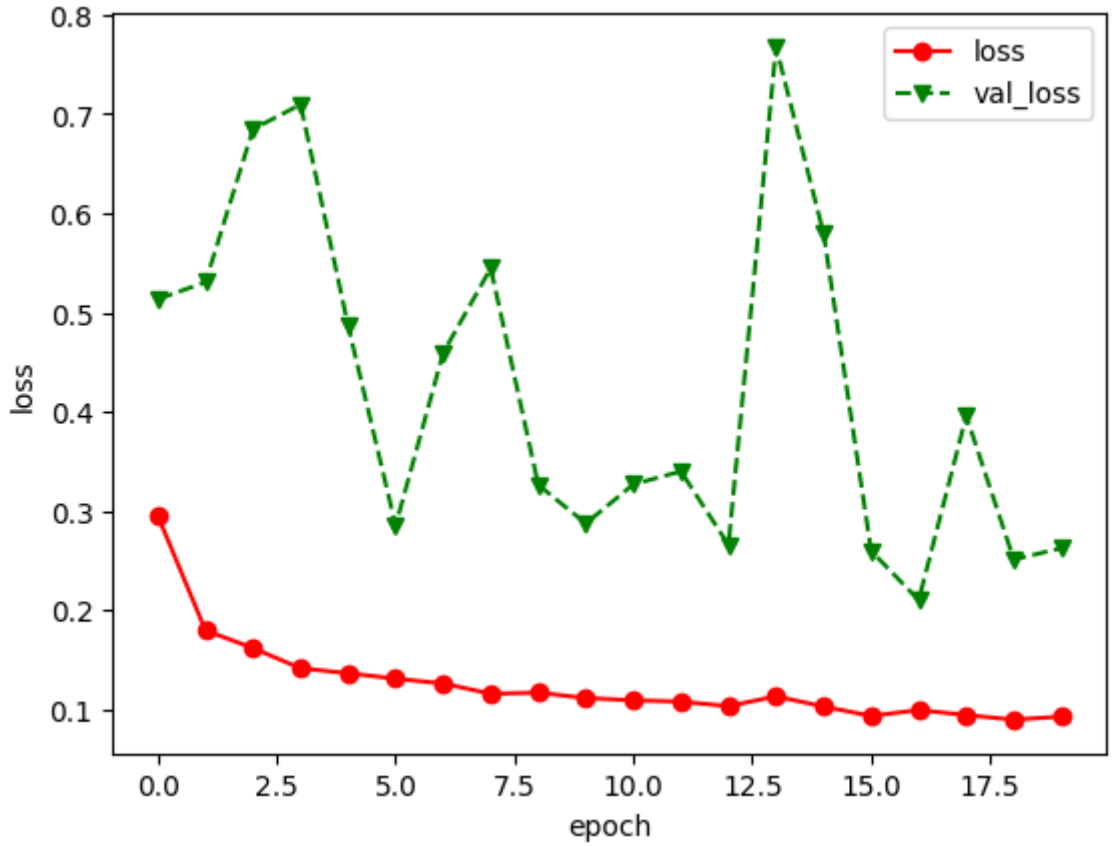
（pixel value）→（pixel value, pixel value, pixel value）

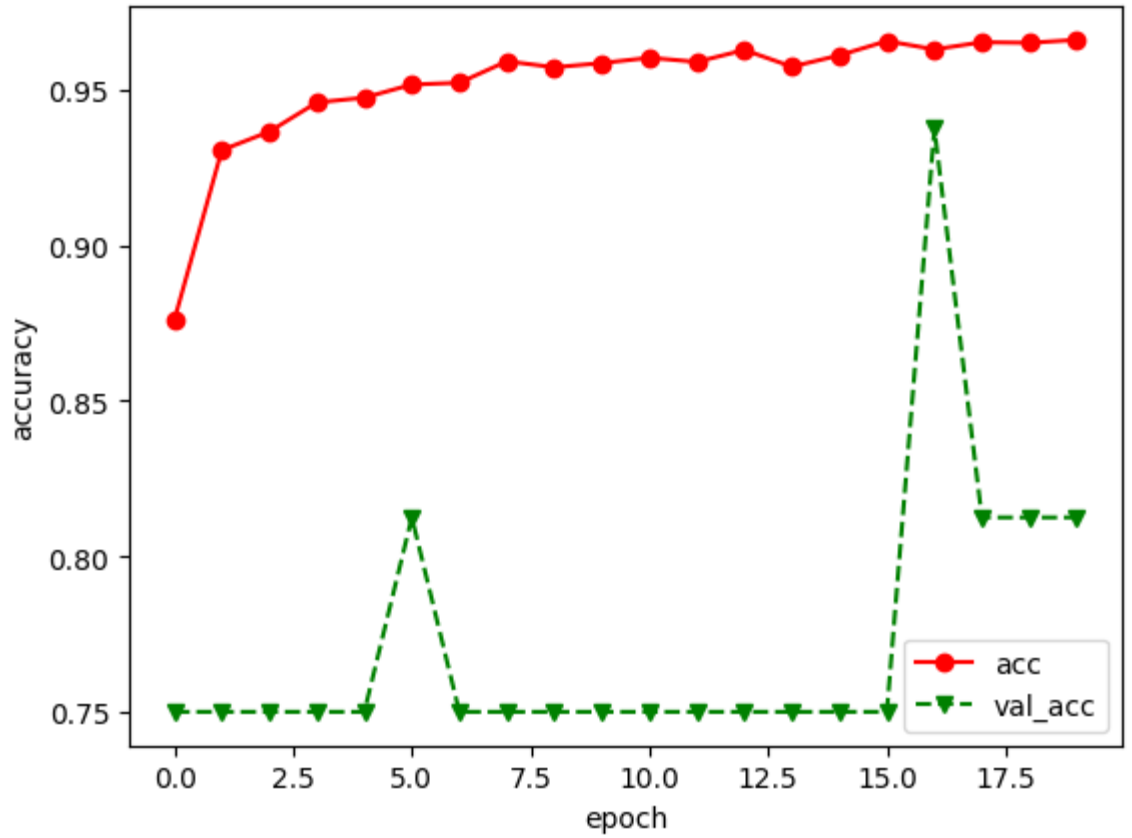
GrayScale　　R　　G　　B

3. 学習結果の表示とモデルの評価

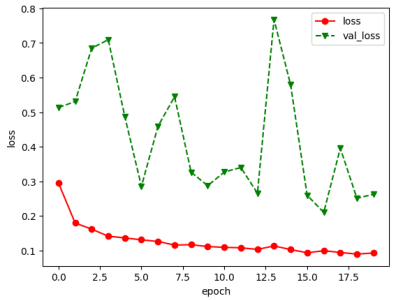
医用画像AIの適切な評価方法を学ぶ

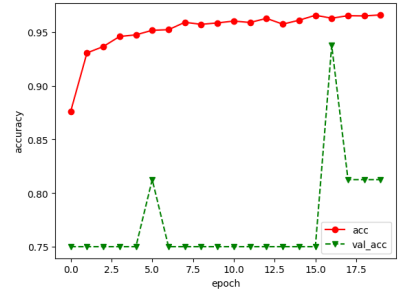
実験1の学習曲線


loss
val_loss
loss
epoch
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


acc
val_acc
accuracy
epoch
0.95
0.90
0.85
0.80
0.75
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

実験1の考察





- 検証データの誤差がばらつき、正解率がほとんど上がっていない
- 学習誤差と検証誤差の差が大きい
  →典型的な**過学習**状態
- 学習回数が少なく、検証誤差が下がりきる前に学習が終了している

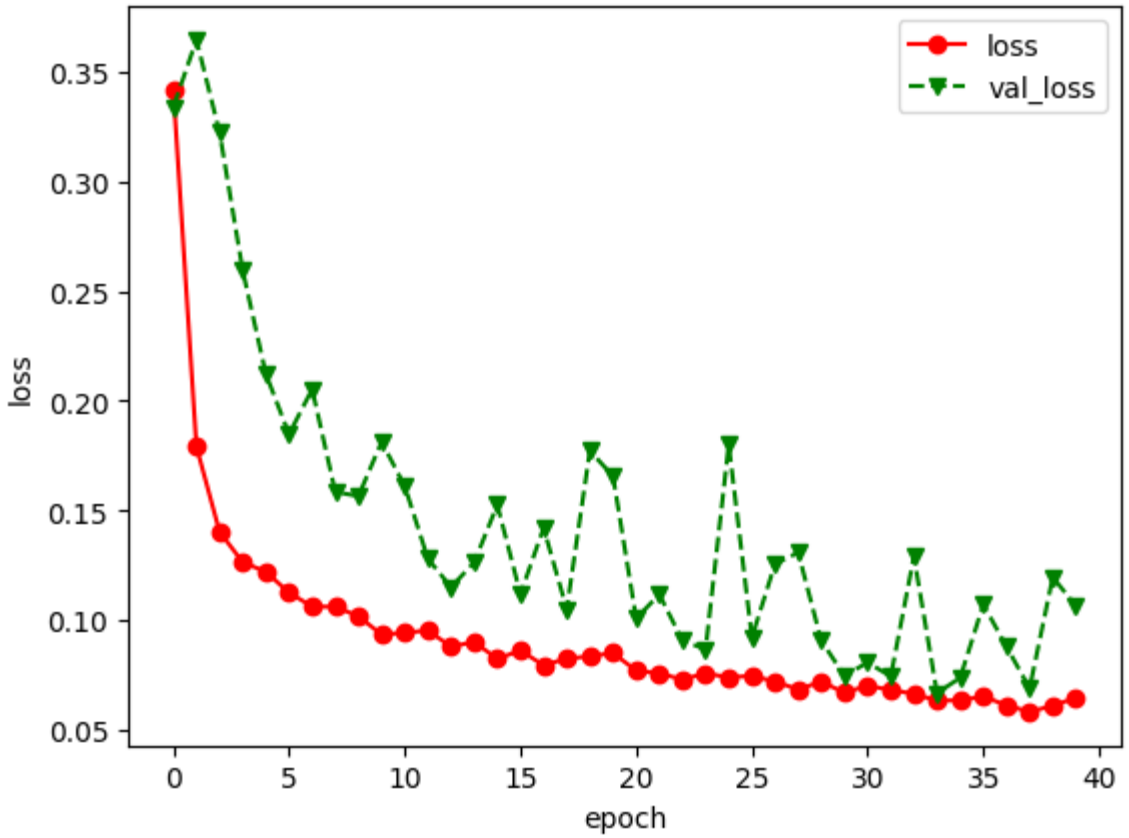
【対策】

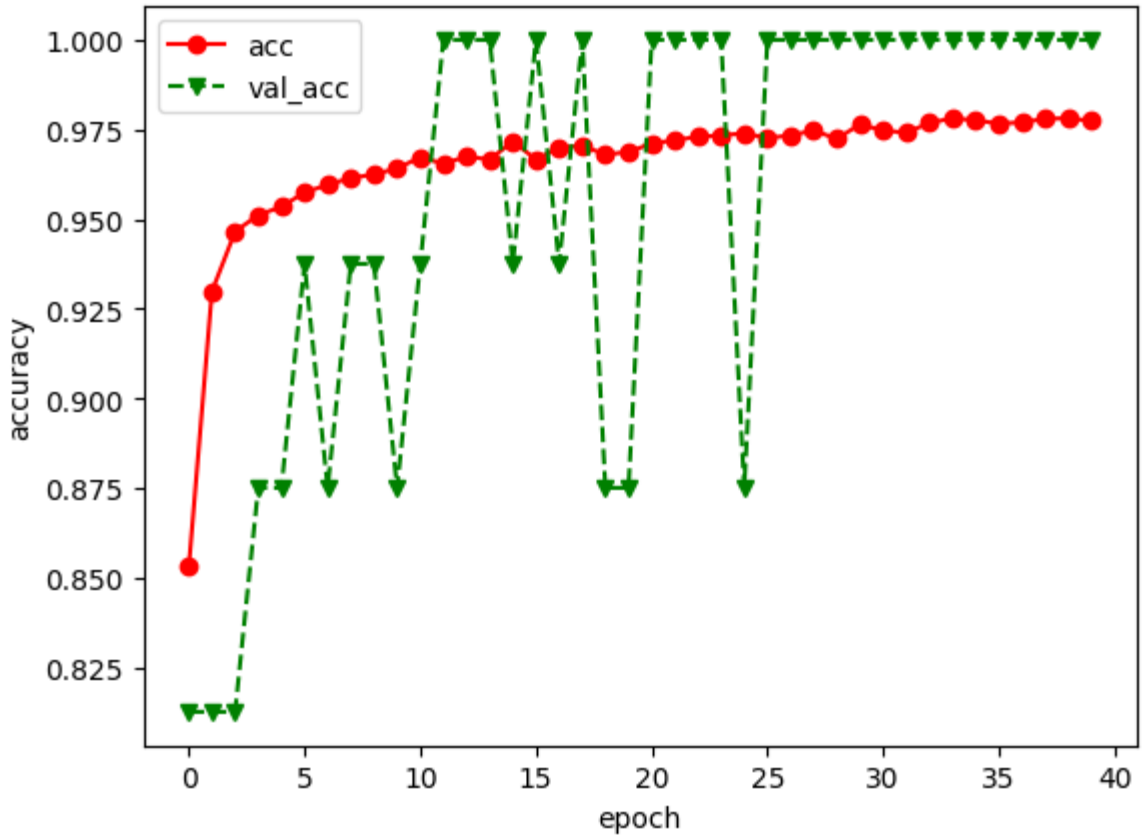
- 可能な限り適切なデータ拡張を追加し、過学習を抑制する
- 学習回数を増やす

実験2, 3のデータ拡張設定

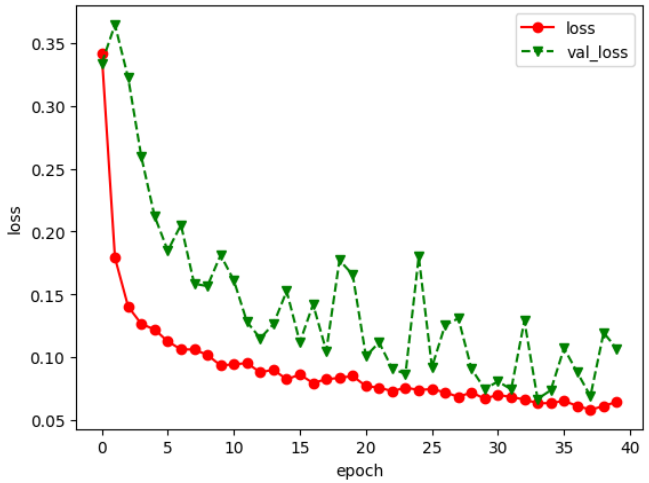
```
# 学習データのデータ拡張を設定
train_datagen = ImageDataGenerator(
        rescale = 1.0 / 255,
        # shear_range = 0.2,
        zoom_range = 0.1,
        rotation_range=10,
        width_shift_range=0.1,
        height_shift_range=0.1,
        brightness_range=[0.9, 1.1],
        fill_mode="constant",
        horizontal_flip = True)
```

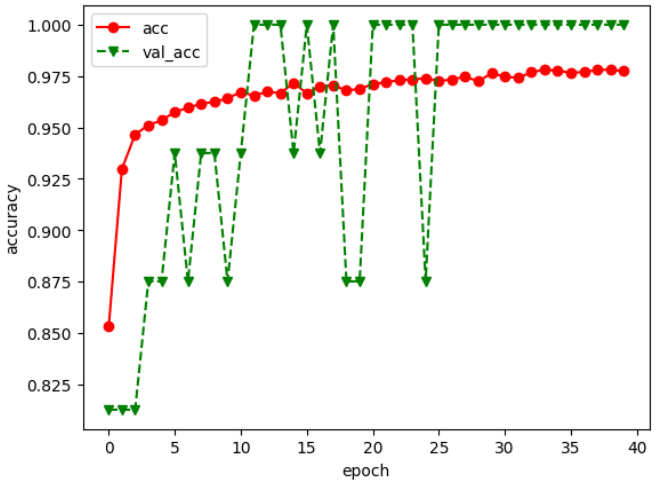
実験2の学習曲線


loss
val_loss
loss
epoch
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
5
10
15
20
25
30
35
40


acc
val_acc
accuracy
epoch
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0
5
10
15
20
25
30
35
40

実験2の考察

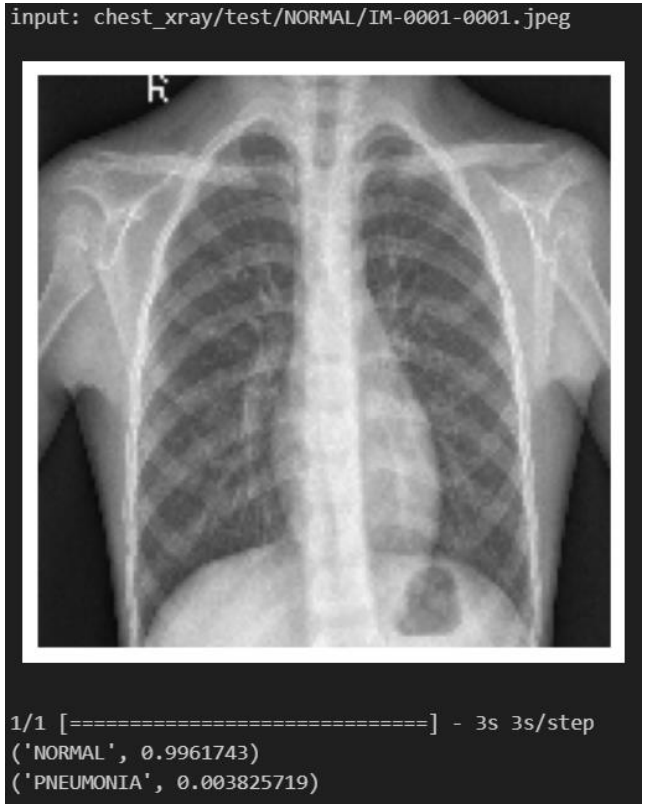




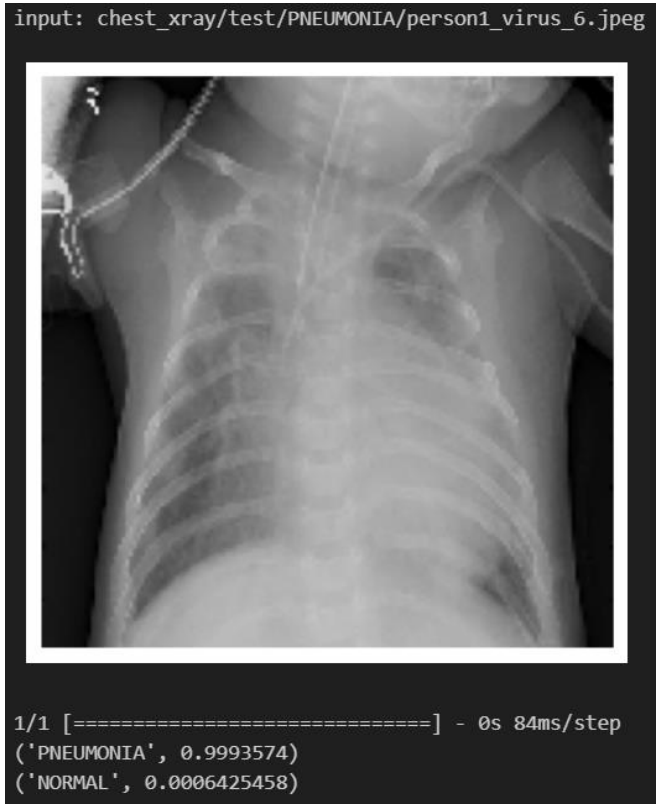
- データ拡張により学習が大きく改善された
- 検証データの正解率が100%（8/8）になった
- 全層ファインチューニングを行えばもっと改善される？
  より医用画像特化のモデルにしたい…という思惑に基づく

※実験2のモデルを使用

テストデータを用いた予測テスト

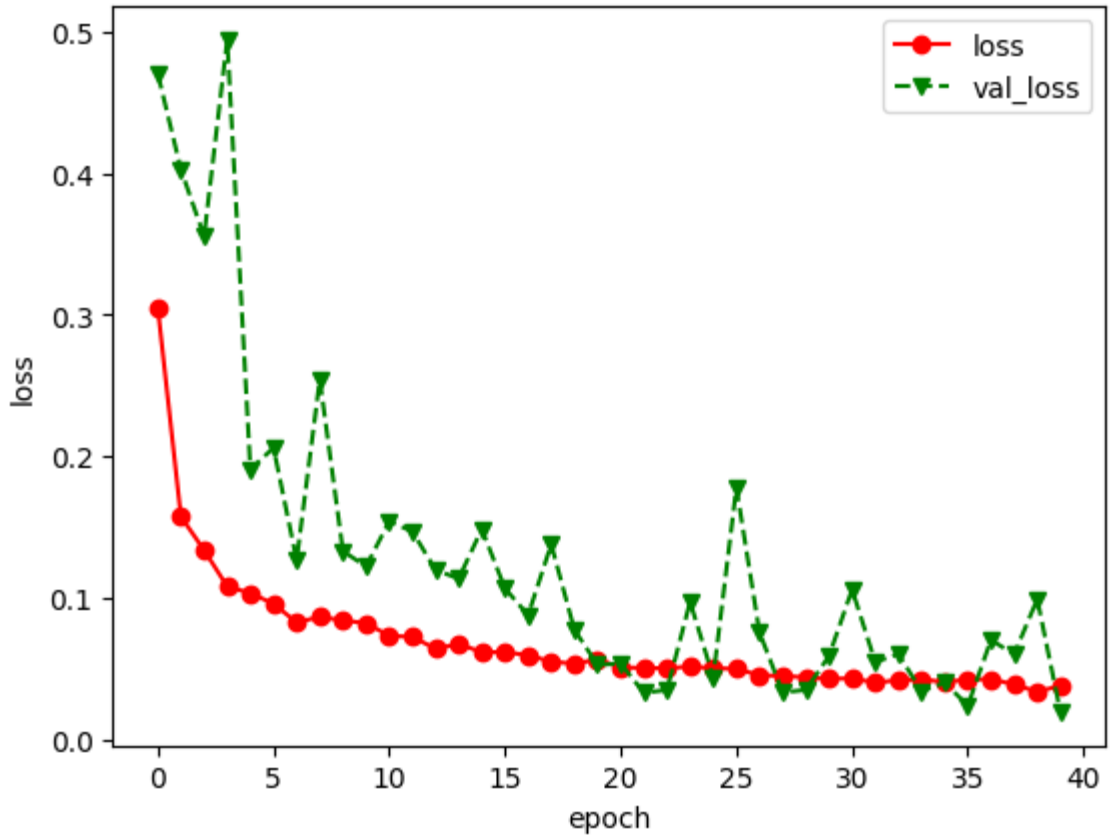


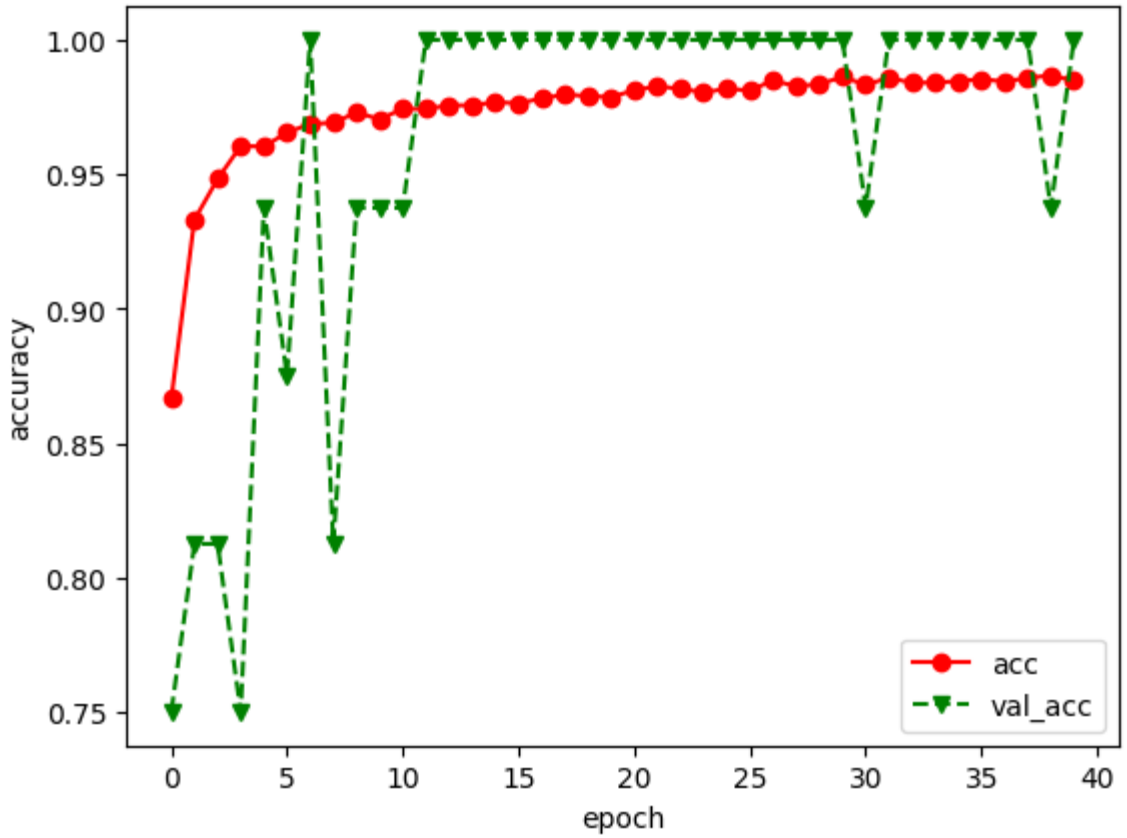
NORMAL: 99%
PNEUMONIA: 0%



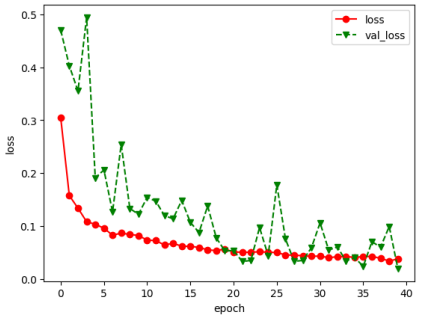
NORMAL: 0%
PNEUMONIA: 99%

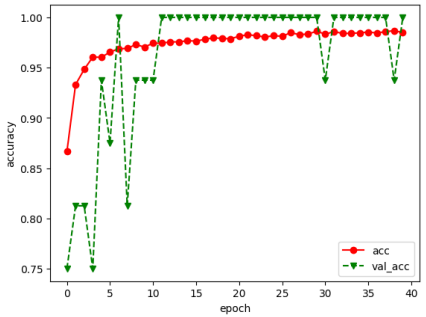
実験3の学習曲線


loss
val_loss
loss
epoch
0.0
0.1
0.2
0.3
0.4
0.5
0
5
10
15
20
25
30
35
40


acc
val_acc
accuracy
epoch
0.75
0.80
0.85
0.90
0.95
1.00
0
5
10
15
20
25
30
35
40

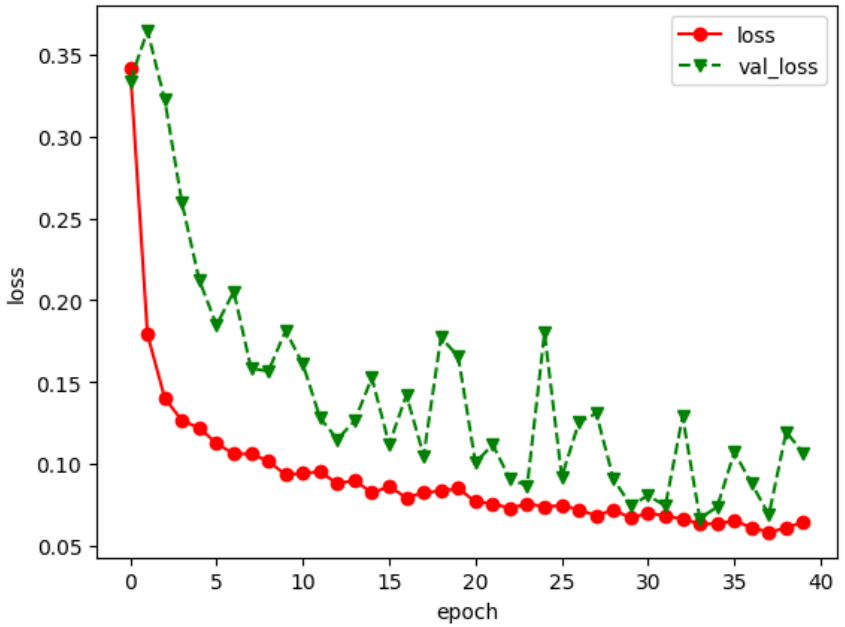
実験3の考察

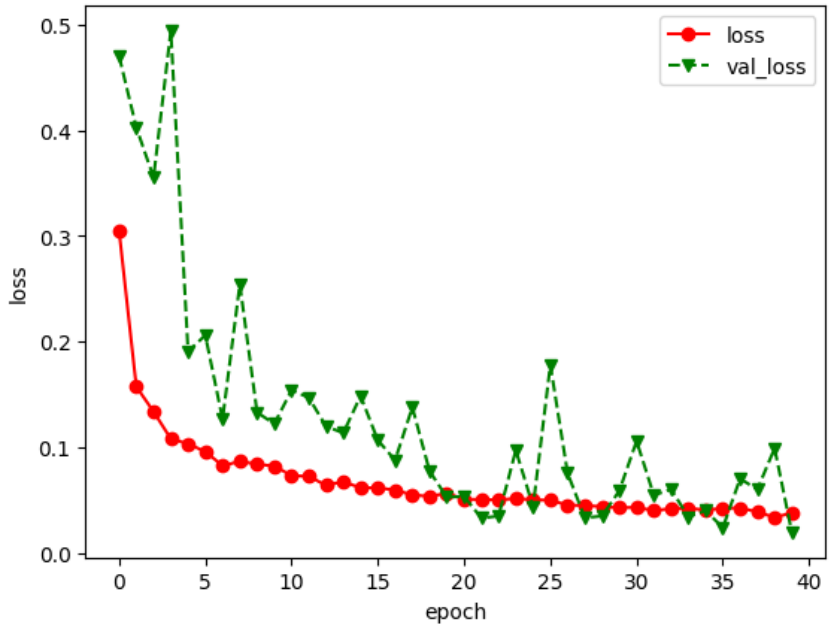




- 実験2に比べて検証誤差がより小さくなった
- 検証データの正解率は変化なし

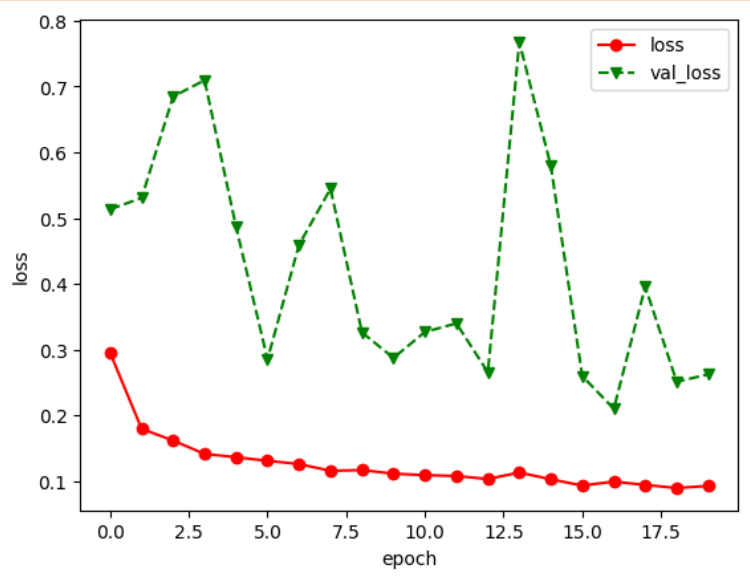
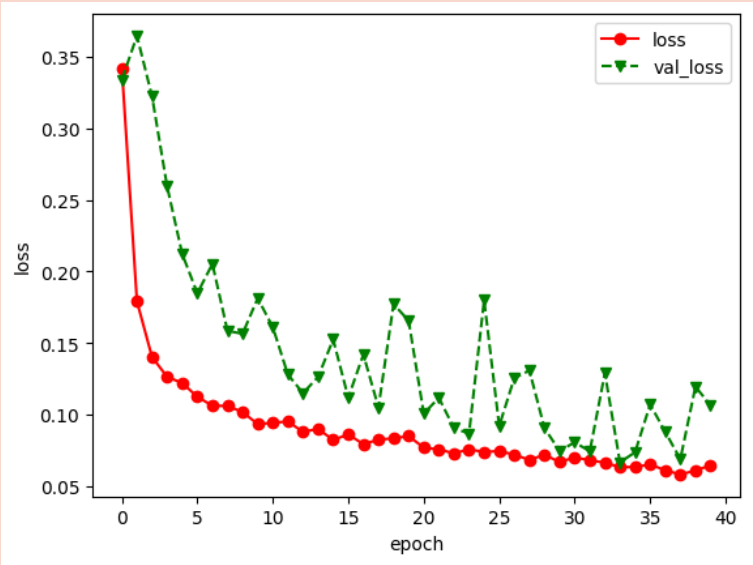
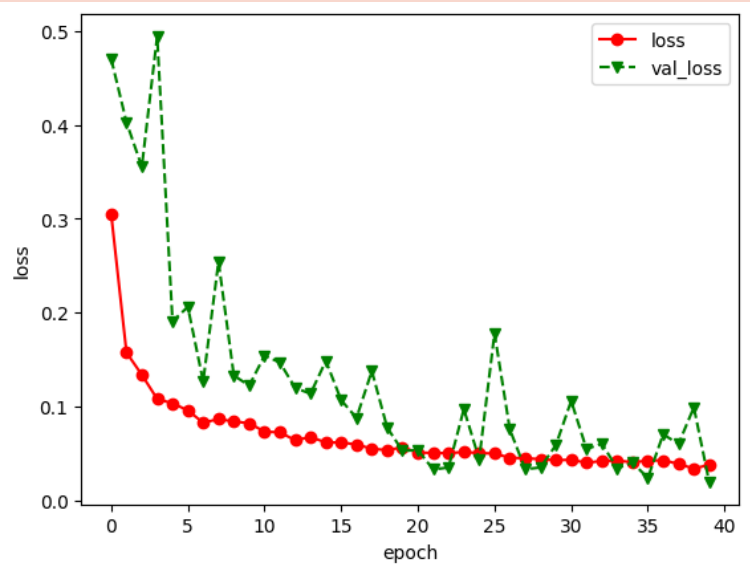
実験2



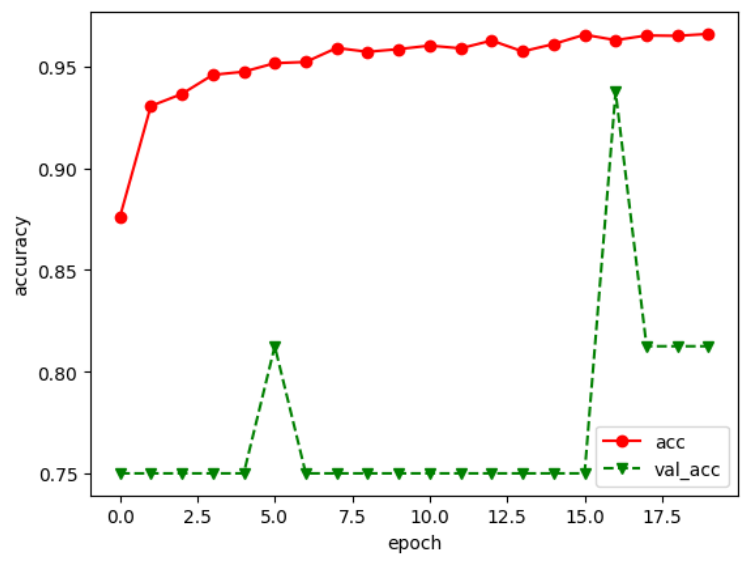
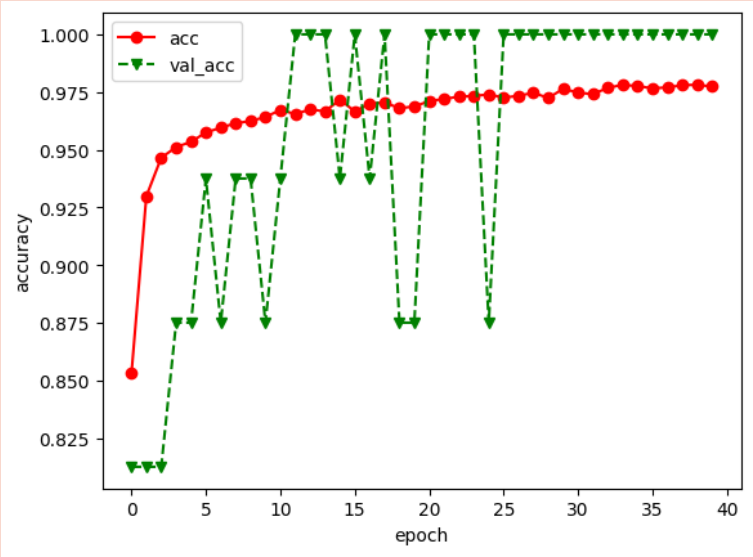
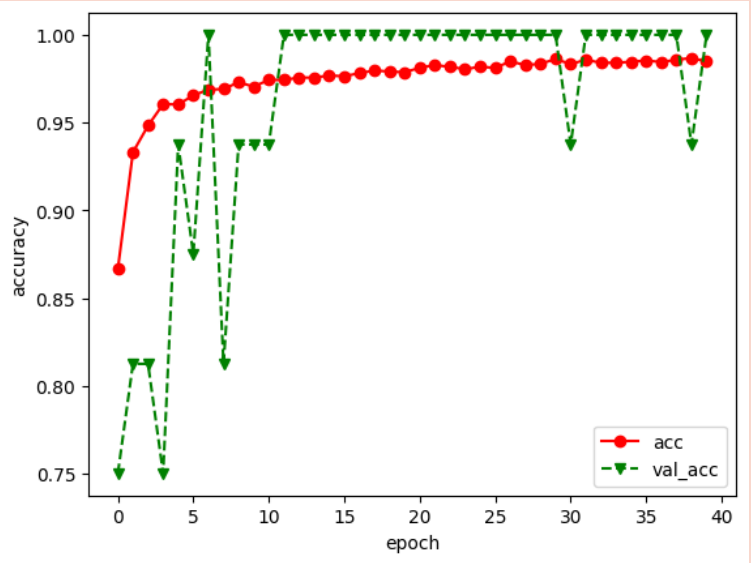


実験3

各実験の学習曲線の比較（loss）

実験1	実験2	実験3
		

各実験の学習曲線の比較（accuracy）

実験1	実験2	実験3
		

モデルの評価

- 学習曲線だけではどのモデルが最も良いか判断しづらい
  特に実験2と実験3のモデル
- Accuracyだけでは本当に良いモデルか判断できない
  理由は後述

- それぞれのモデルに対してテストデータを用いて評価を行う
- さまざまな評価指標を用いて多角的に評価する

モデルの評価指標（1）

Accuracy（正解率、精度？）
全データのうち、正しく予測されたデータの割合
Precision（適合率）
陽性と予測されたデータのうち、実際に陽性であるものの割合
Recall（Sensitivity, 再現率、感度、真陽性率）
正解が陽性であるデータのうち、陽性と予測できたものの割合
Specificity（特異度）
正解が陰性であるデータのうち、陰性と予測できたものの割合
F1 Score（F値）
Precision, Recallの調和平均

モデルの評価指標（2）

Receiver Operating Characteristic Curve（ROC曲線）

縦軸に真陽性率（Recall）、横軸に偽陽性率（1 – Specificity）のグラフ

Area Under Curve（AUC）

ROC曲線の面積、1に近づくほど良い

Precision Recall Curve（PR曲線）

縦軸にPrecision、横軸にRecallのグラフ

Average Precision Score（AP）

PR曲線の面積、1に近づくほど良い

Confusion Matrix（混同行列）

後述

TP, TN, FP, FN

	正解は真	正解は偽
予測は真	TP	FP
予測は偽	FN	TN

TP　True Positive　真陽性　予測と正解が真であるデータの合計
TN　True Negative　真陰性　予測と正解が偽であるデータの合計
FP　False Positive　偽陽性　予測が真、正解が偽であるデータの合計
FN　False Negative　偽陰性　予測が偽、正解が真であるデータの合計

肺炎であるか否かの場合は…

正解

予測	**肺炎である**	**肺炎ではない**
肺炎である	TP	FP
肺炎ではない	FN	TN

各評価指標の計算方法

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

計算例（1）

※モデルの出力は（健常の確率、肺炎の確率）

【例題】以下に対する各クラスのPrecision, Recall, Specificityを求めよ
ただし、予測のしきい値は0.5とする

	予測（健常）	正解（健常）	予測（肺炎）	正解（肺炎）
データ1	0.8	1	0.2	0
データ2	0.1	0	0.9	1
データ3	0.7	0	0.3	1
データ4	0.6	0	0.4	1
データ5	1.0	1	0.0	0
データ6	0.9	1	0.1	0
データ7	0.1	0	0.9	1

計算例（2）

1. 予測確率をしきい値処理により二値化する

	予測（健常）	正解（健常）	予測（肺炎）	正解（肺炎）
データ1	0.8→1	1	0.2→0	0
データ2	0.1→0	0	0.9→1	1
データ3	0.7→1	0	0.3→0	1
データ4	0.6→1	0	0.4→0	1
データ5	1.0→1	1	0.0→0	0
データ6	0.9→1	1	0.1→0	0
データ7	0.1→0	0	0.9→1	1

計算例（3）

2. 各クラスのTP, TN, FP, FNを計算する

	予測（健常）		正解（健常）	予測（肺炎）		正解（肺炎）
データ1	1	**TP**	1	0	**TN**	0
データ2	0	**TN**	0	1	**TP**	1
データ3	1	**FP**	0	0	**FN**	1
データ4	1	**FP**	0	0	**FN**	1
データ5	1	**TP**	1	0	**TN**	0
データ6	1	**TP**	1	0	**TN**	0
データ7	0	**TN**	0	1	**TP**	1

TP=3, TN=2, FP=2, FN=0　　TP=2, TN=3, FP=0, FN=2

計算例（4）

3. 各クラスのPrecision, Recall, Specificityを計算する

【健常の予測】

$$Precision = \frac{3}{3+2} = 0.6 \qquad Recall = \frac{3}{3+0} = 1.0 \qquad Specificity = \frac{2}{2+2} = 0.5$$

【肺炎の予測】

$$Precision = \frac{2}{2+0} = 1.0 \qquad Recall = \frac{2}{2+2} = 0.5 \qquad Specificity = \frac{3}{3+0} = 1.0$$

Accuracyの落とし穴

【例題】以下の結果に対するAccuracyを求めよ

予測＝[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

正解＝[0, 1, 0, 0, 0, 0, 0, 0, 0, 1]

TP＝0, TN＝8, FP＝0, FN＝2より $\mathbf{Accuracy} = \frac{0+8}{0+8+0+2} = \frac{8}{10} = 0.8 = 80\%$

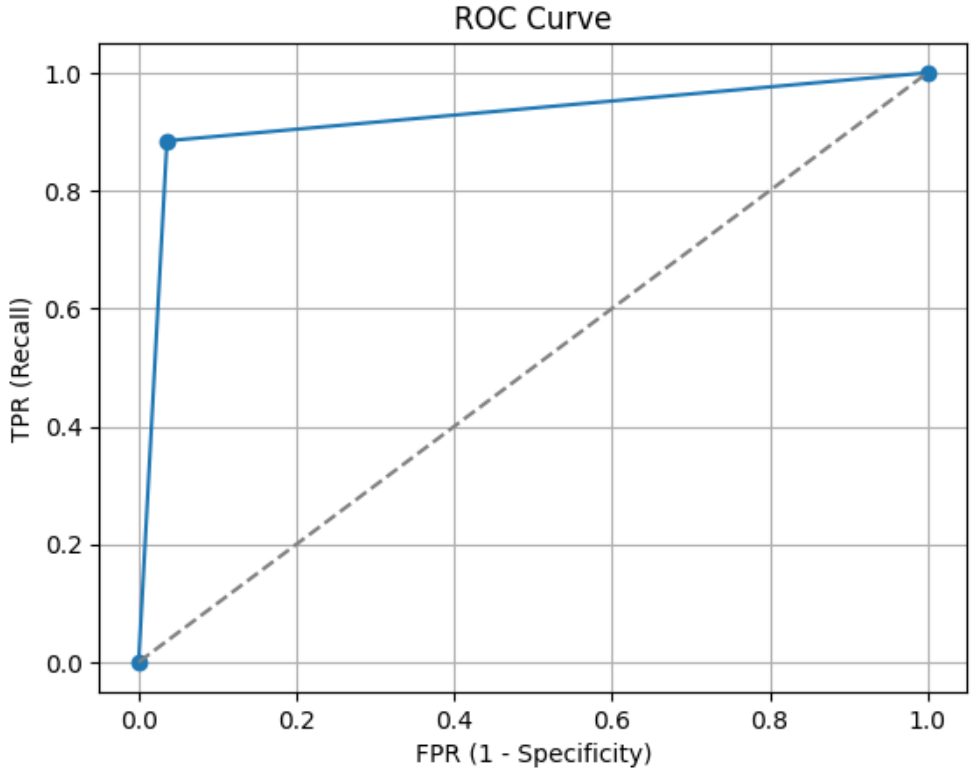
- すべて陰性（0）と予測した場合でも、Accuracyが80%と高い値となる
- 必ず0を出力するデタラメなモデルでも精度が高く見えてしまう
- TNが多ければ多いほど、Accuracyは自然と100%に近づいてしまう

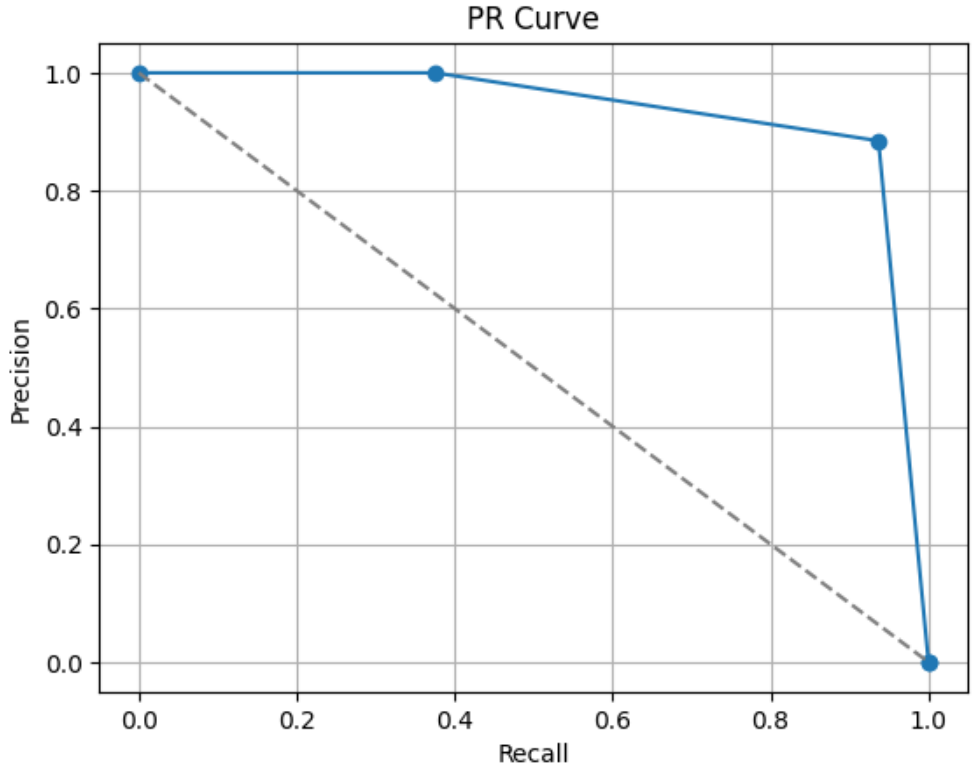
Specificityの重要性

- Specificity（特異度）は陰性をちゃんと陰性と判断できたか
- 医療分野では陰性を陽性と判断すると非常にまずい
  例）陰性のがんを陽性と誤って判断し、不必要な手術を行ってしまう
- 診断を主目的とするAIの場合はSpecificityも計算したほうが良い
- 診断が主目的でない場合は基本的にPrecision, Recallなどで十分

ROC CurveとPR Curve

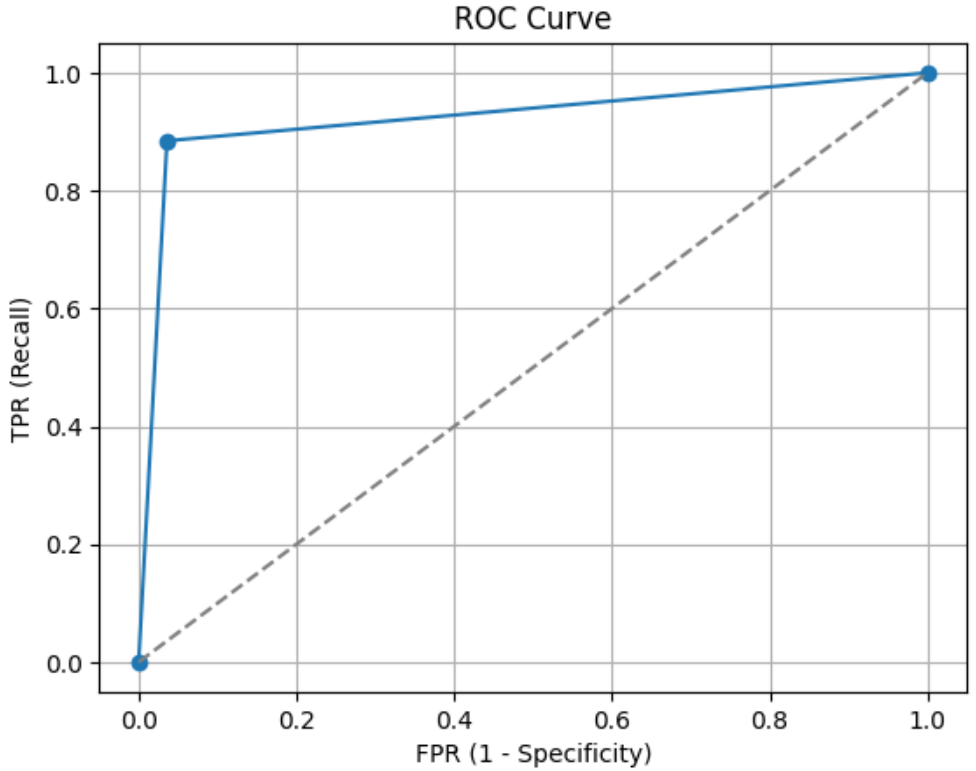
- 予測のしきい値によって評価指標の計算結果は異なる
- しきい値を変化させて評価値の変化を見たい→ROC曲線とPR曲線

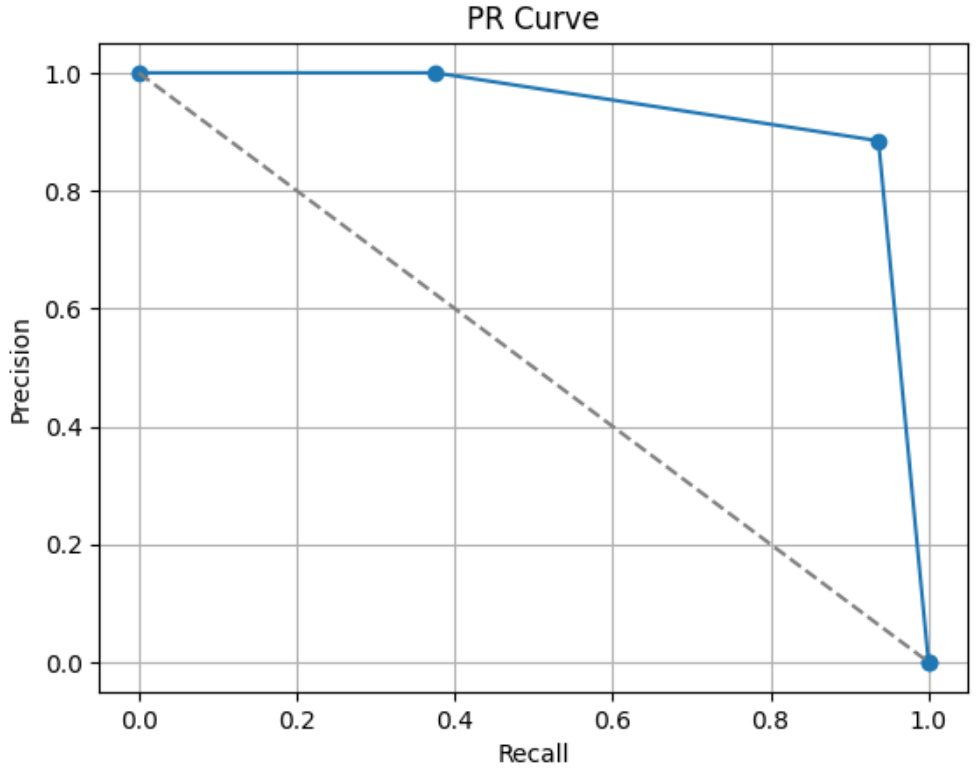




ROC CurveとPR Curveの見方

- 傾き1の直線に比べてどれくらい曲線が膨らんでいるかを見る
- 面積（曲線の積分）が大きいほど良いモデルといえる



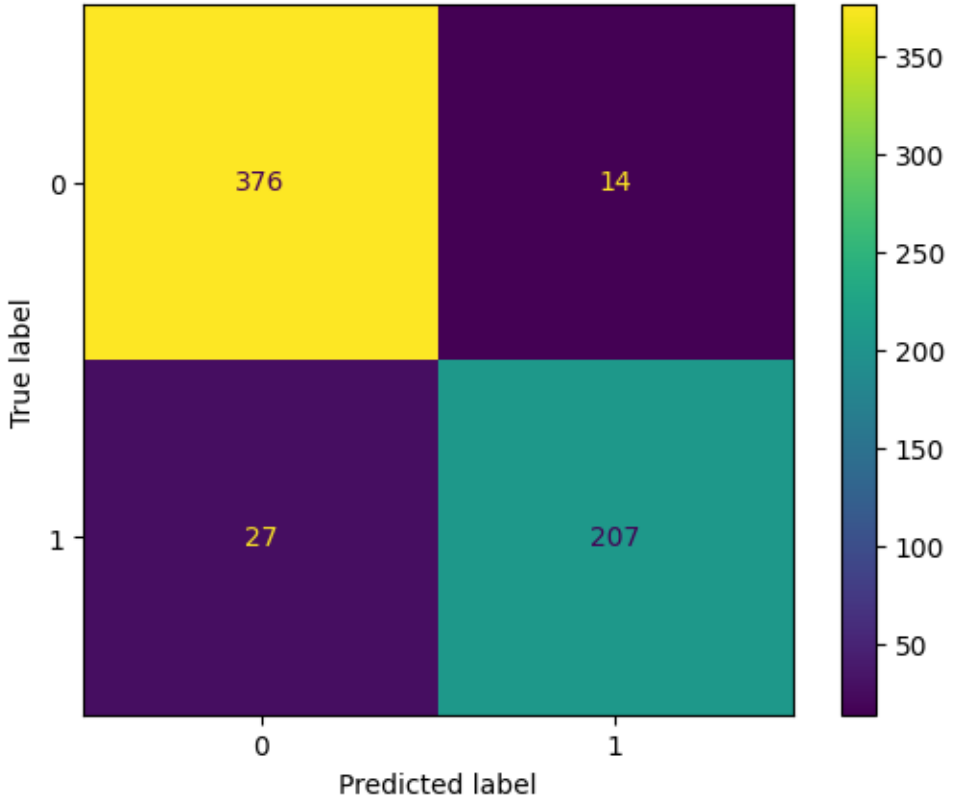


Confusion Matrix（1）

Confusion Matrixは二値分類と多クラス分類で解釈が異なる

【二値分類】
あるクラスに対する
TP, TN, FP, FNの計算と同義

正解は偽	TN	FP
正解は真	FN	TP
	予測は偽	予測は真



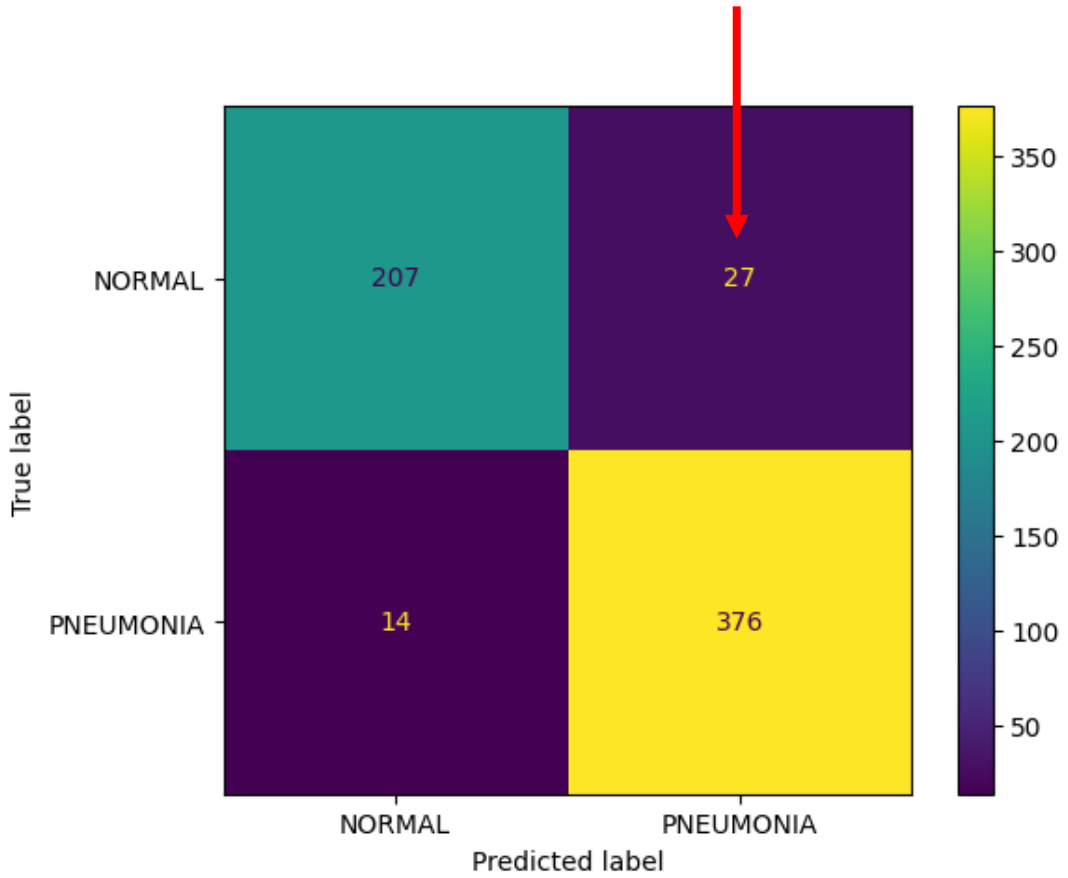
Confusion Matrix（2）

見方の例）予測がPNEUMONIA、正解がNORMALの合計

【多クラス分類】

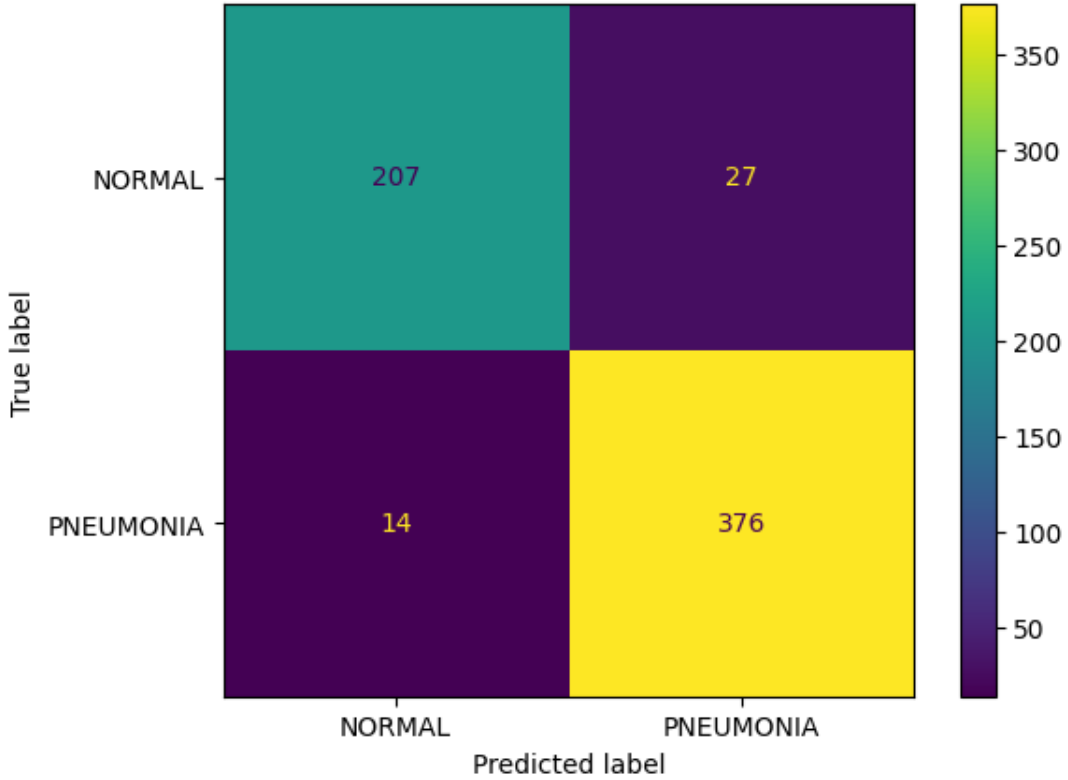
- 予測クラスと正解クラスの関係を表す
- 何を何と間違えたか把握できる

正解		予測			
	クラス1	TP	FP(FN)	FP(FN)	
	クラス2	FP(FN)	TP	FP(FN)	
	クラス3	FP(FN)	FP(FN)	TP	
⋮		クラス1	クラス2	クラス3	…



Confusion Matrixの見方

- 対角成分は予測と正解が合致した数（＝TPの数）
- 対角成分の数値が大きいほど良いモデルといえる



多クラス分類におけるモデル全体の評価（1）

- 評価指標は基本的に各クラスに対して計算する
- 多クラス分類モデルの全体的な評価はどのように計算すればよいのか

⬇

Macro平均

- 各クラスで評価値を計算してから、その平均をとる
- 各クラスにおけるモデルの性能を平等に評価できる

Micro平均

- 各クラスのTP, TN, FP, FNを計算したあと合算し、評価値を計算する
- データセット全体に対するモデルの性能を評価できる

多クラス分類におけるモデル全体の評価（2）

- 医用画像のデータセットは各クラスのデータ数が偏りやすい
- 医療分野では基本的に陽性が陰性よりも少なくなりやすい
  例）虫歯のデータ数：健康な歯のデータ数＝1：9
- データ数に偏りがあるデータセットのことを**不均衡データ**という
- Micro平均を取ると不均衡データの偏りを考慮することができない

⬇

- 特に理由がなければ、医用画像AIの評価では**Macro平均**を用いる

評価指標の注意点

- タスクや目的によって評価指標の計算方法が微妙に異なることがある
  例）物体検出におけるAverage Precisionの計算
- Micro平均は混同行列を用いた計算方法が一般的である
  Scikit Learnでは混同行列を用いた計算を行っている

Scikit Learnを用いた評価指標の実装

```
from sklearn import metrics
```

- sklearnのmetricsモジュールを使えば簡単に実装可能
- Specificityは実装されていないため、今回は多少の手書きが必要

下準備（1）

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

from sklearn import metrics
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical



test_data_dir = "chest_xray/test"
model_name = "pneumonia_3.h5"
```

下準備（2）

```
# 画像サイズの設定
img_height, img_width = 150, 150

# 分類カテゴリ名の設定、学習時と同じ順番にする
classes = ['NORMAL','PNEUMONIA']
nb_classes = len(classes)
```

評価用データセットの準備

```
# 評価用（テスト用）データセットの準備
test_datagen = ImageDataGenerator(rescale = 1.0 / 255)

# ジェネレータを生成
test_generator = test_datagen.flow_from_directory(
        test_data_dir,
        target_size = (img_height, img_width),
        classes = classes,
        batch_size = 1,
        class_mode = 'categorical',
        shuffle=False)

# 正解ラベル（健康=0、肺炎=1）のOne-hotベクトルを取得
y_true = to_categorical(test_generator.labels, nb_classes)
```

予測

```
# ファインチューニングしたCNNモデルのロード
model = tf.keras.models.load_model(model_name)

# テストデータに対する予測
y_pred = model.predict(test_generator, verbose=1)
print(y_pred)
```

予測結果の処理

```
# クラスごとに評価
THRESH = 0.5
for i in range(nb_classes):
    # 特定クラスの予測結果のみを取得
    p = y_pred[:, i]
    p_prob = p[:]        # ROC曲線、PR曲線用に配列をコピー
    gt = y_true[:, i]

    # 予測結果のしきい値処理
    p[p >= THRESH] = 1
    p[p < THRESH] = 0
```

各評価指標の実装

```
# 二値混同行列
cf = metrics.confusion_matrix(gt, p)

# TP, TN, FP, FNの取得
tn, fp, fn, tp = cf.ravel()

# 各評価指標の計算
acc = (tp + tn) / (tp + tn + fp + fn)
precision = tp / (tp + fp)
recall = tp / (tp + fn)
specificity = tn / (tn + fp)
f1 = (2 * precision * recall) / (precision + recall)
sum_f1 += f1
```

AUCとAP

```
# AUCとAP
auc = metrics.roc_auc_score(gt, p_prob)
ap = metrics.average_precision_score(gt, p_prob)
print(f"AUC: {auc:.3f}, AP: {ap:.3f}")
sum_auc += auc
sum_ap += ap
```

ROC曲線

```
# ROC曲線
fpr, tpr, thresholds = metrics.roc_curve(gt, p_prob)
plt.plot(fpr, tpr, marker="o")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")
plt.title("ROC Curve")
plt.xlabel("FPR (1 - Specificity)")
plt.ylabel("TPR (Recall)")
plt.grid()
plt.show()
plt.clf()
```

PR曲線

```
# PR曲線
recalls, precisions, thresholds = metrics.precision_recall_curve(gt, p_prob)
recalls = np.insert(recalls, 0, 0.0)
recalls = np.append(recalls, 1.0)
precisions = np.insert(precisions, 0, 1.0)
precisions = np.append(precisions, 0.0)
plt.plot(recalls, precisions, marker="o")
plt.plot([0, 1], [precisions[0], precisions[-1]], linestyle="--", color="gray")
plt.title("PR Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.grid()
plt.show()
plt.clf()
```

各クラスに対するConfusion Matrix

```
# 混同行列の表示
cf_disp = metrics.ConfusionMatrixDisplay(confusion_matrix=cf)
cf_disp.plot()
plt.show()
plt.clf()
```

Confusion Matrix（多クラス）

```
# 多クラス混同行列
yp = np.argmax(y_pred, axis=1)
yt = np.argmax(y_true, axis=1)
cf_all = metrics.confusion_matrix(yt, yp)
cf_disp = metrics.ConfusionMatrixDisplay(confusion_matrix=cf_all, display_labels=classes)
cf_disp.plot()
plt.show()
```

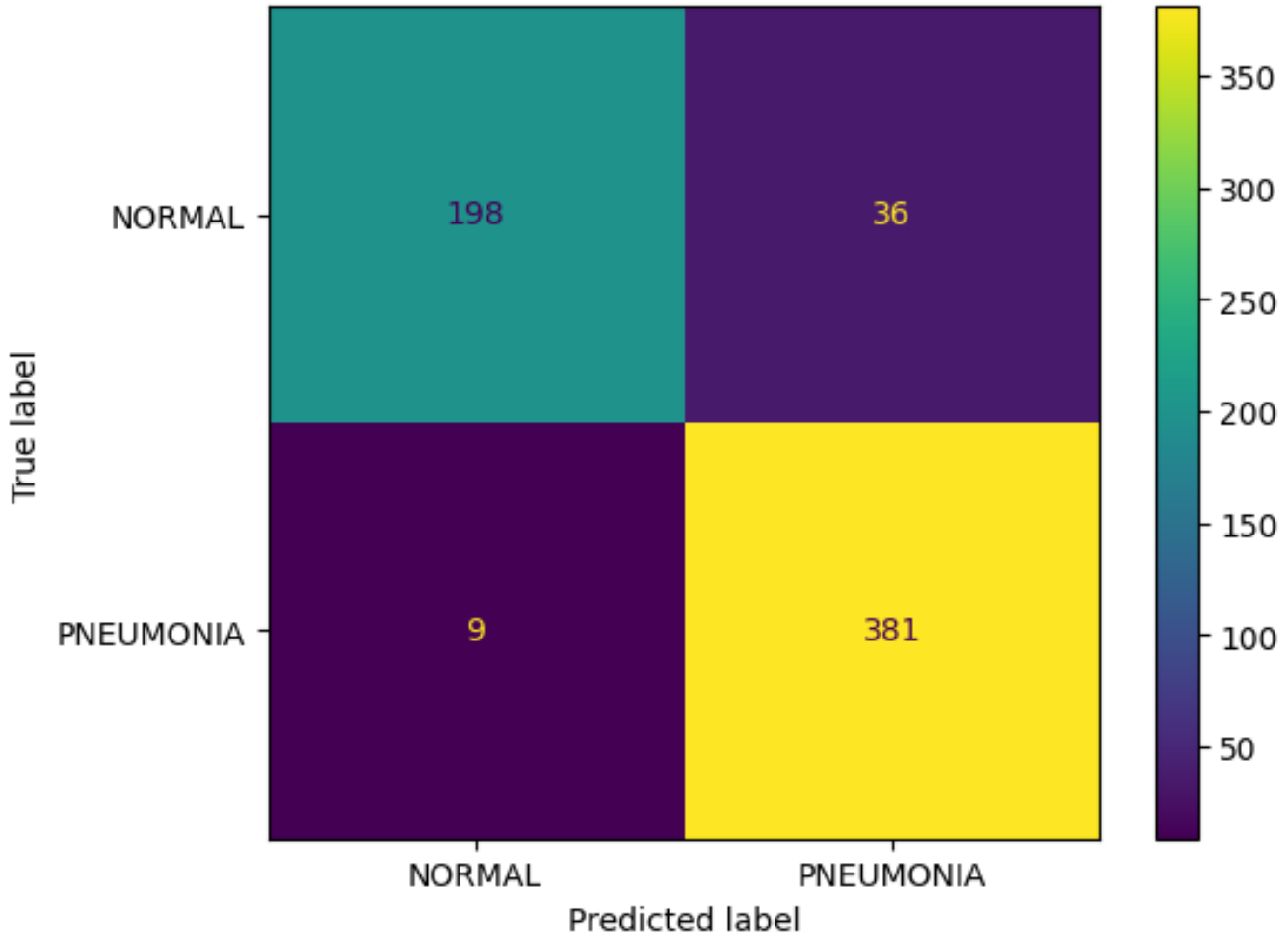
実験1の評価

```
[NORMAL]
Accuracy 0.928, Precision 0.957, Recall 0.846, Specificity 0.977, F1 0.898
AUC: 0.912, AP: 0.867
```

```
[PNEUMONIA]
Accuracy 0.928, Precision 0.914, Recall 0.977, Specificity 0.846, F1 0.944
AUC: 0.912, AP: 0.907
```

	NORMAL	PNEUMONIA	Macro Mean
Accuracy	0.928	0.928	
Precision	0.957	0.914	
Recall	0.846	0.977	
Specificity	0.977	**0.846**	
F1 Score	0.898	0.944	**0.921**
Area Under Curve	0.912	0.912	**0.912**
Average Precision	0.867	0.907	**0.887**

実験1の混同行列


198
36
9
381
NORMAL
PNEUMONIA
True label
NORMAL
PNEUMONIA
Predicted label
350
300
250
200
150
100
50

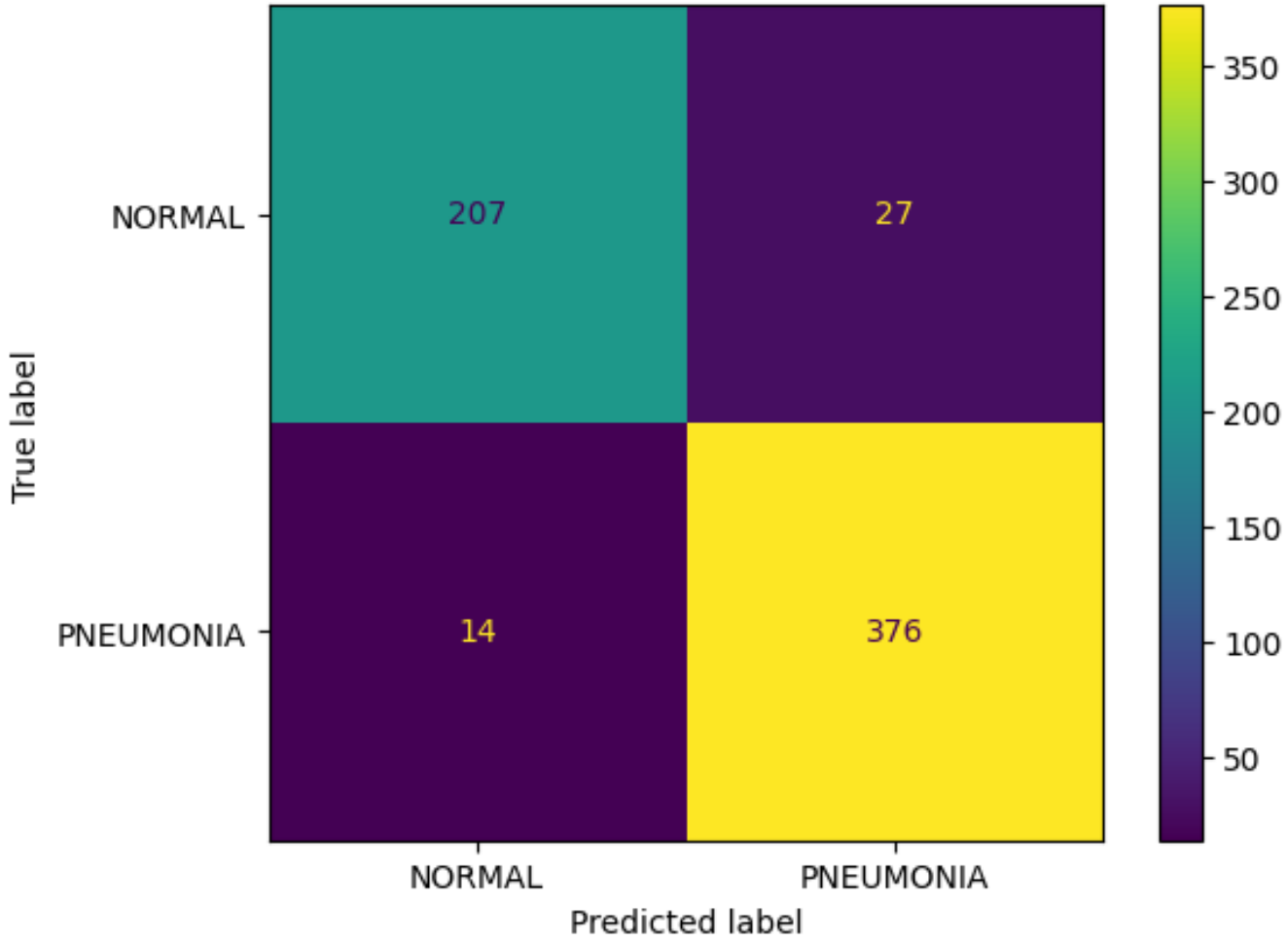
実験2の評価

```
[NORMAL]
Accuracy 0.934, Precision 0.937, Recall 0.885, Specificity 0.964, F1 0.910
AUC: 0.924, AP: 0.872
```

```
[PNEUMONIA]
Accuracy 0.934, Precision 0.933, Recall 0.964, Specificity 0.885, F1 0.948
AUC: 0.924, AP: 0.922
```

	NORMAL	PNEUMONIA	Macro Mean
Accuracy	0.934	0.934	
Precision	0.937	0.933	
Recall	0.885	0.964	
Specificity	0.964	**0.885**	
F1 Score	0.910	0.948	**0.929**
Area Under Curve	0.924	0.924	**0.924**
Average Precision	0.872	0.922	**0.897**

実験2の混同行列


207
27
14
376
NORMAL
PNEUMONIA
True label
NORMAL
PNEUMONIA
Predicted label
350
300
250
200
150
100
50

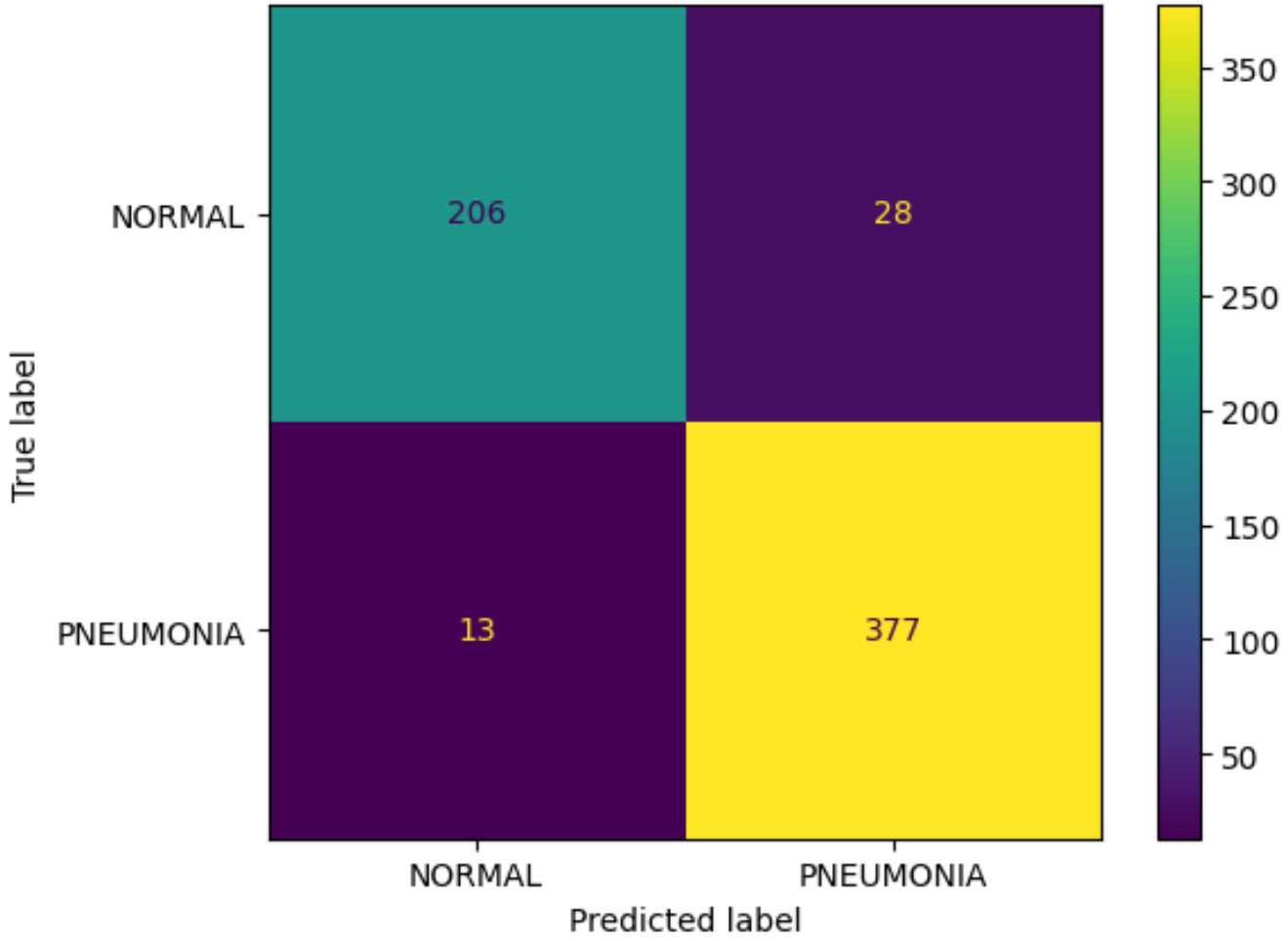
実験3の評価

```
[NORMAL]
Accuracy 0.934, Precision 0.941, Recall 0.880, Specificity 0.967, F1 0.909
AUC: 0.924, AP: 0.873
```

```
[PNEUMONIA]
Accuracy 0.934, Precision 0.931, Recall 0.967, Specificity 0.880, F1 0.948
AUC: 0.924, AP: 0.921
```

	NORMAL	PNEUMONIA	Macro Mean
Accuracy	0.934	0.934	
Precision	0.941	0.931	
Recall	0.880	0.967	
Specificity	0.967	**0.880**	
F1 Score	0.909	0.948	**0.929**
Area Under Curve	0.924	0.924	**0.924**
Average Precision	0.873	0.921	**0.897**

実験3の混同行列


206
28
13
377
NORMAL
PNEUMONIA
True label
NORMAL
PNEUMONIA
Predicted label
350
300
250
200
150
100
50

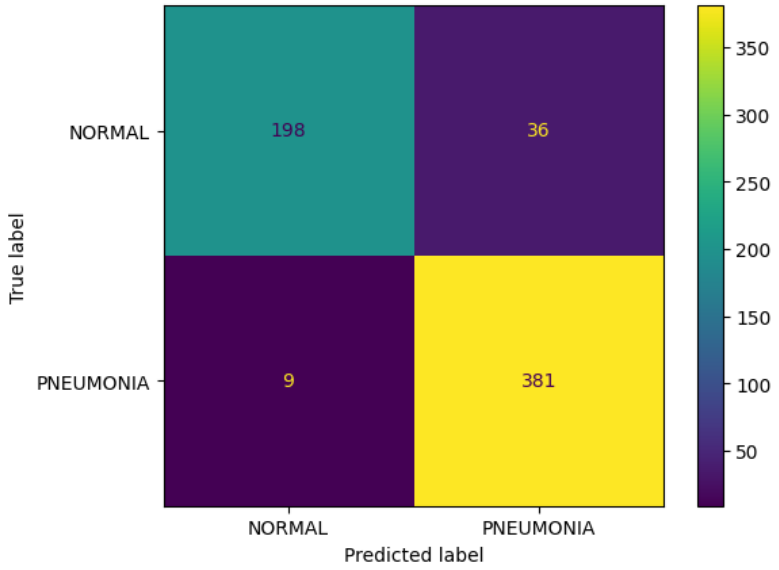
評価の比較（1）

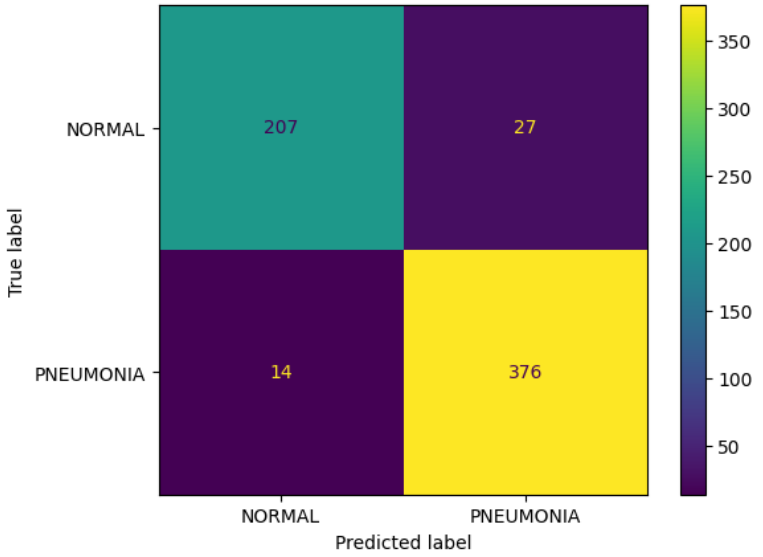
	実験1	実験2	実験3
Mean F1 Score	0.921	**0.929**	**0.929**
Mean AUC	0.912	**0.924**	**0.924**
Mean AP	0.887	**0.897**	**0.897**

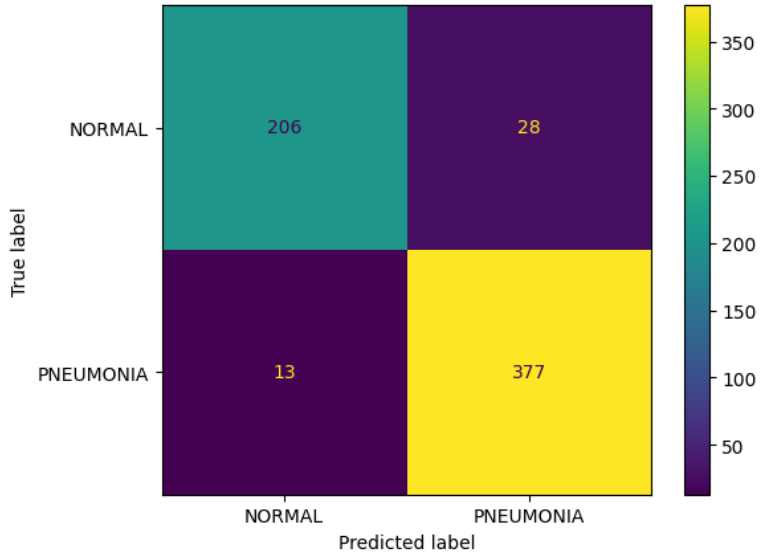
- 実験2および3のモデルは同等の性能
- 実験2および3のモデルは実験1のモデルに比べて約1%の性能向上

評価の比較（2）

※左から順に実験1, 2, 3



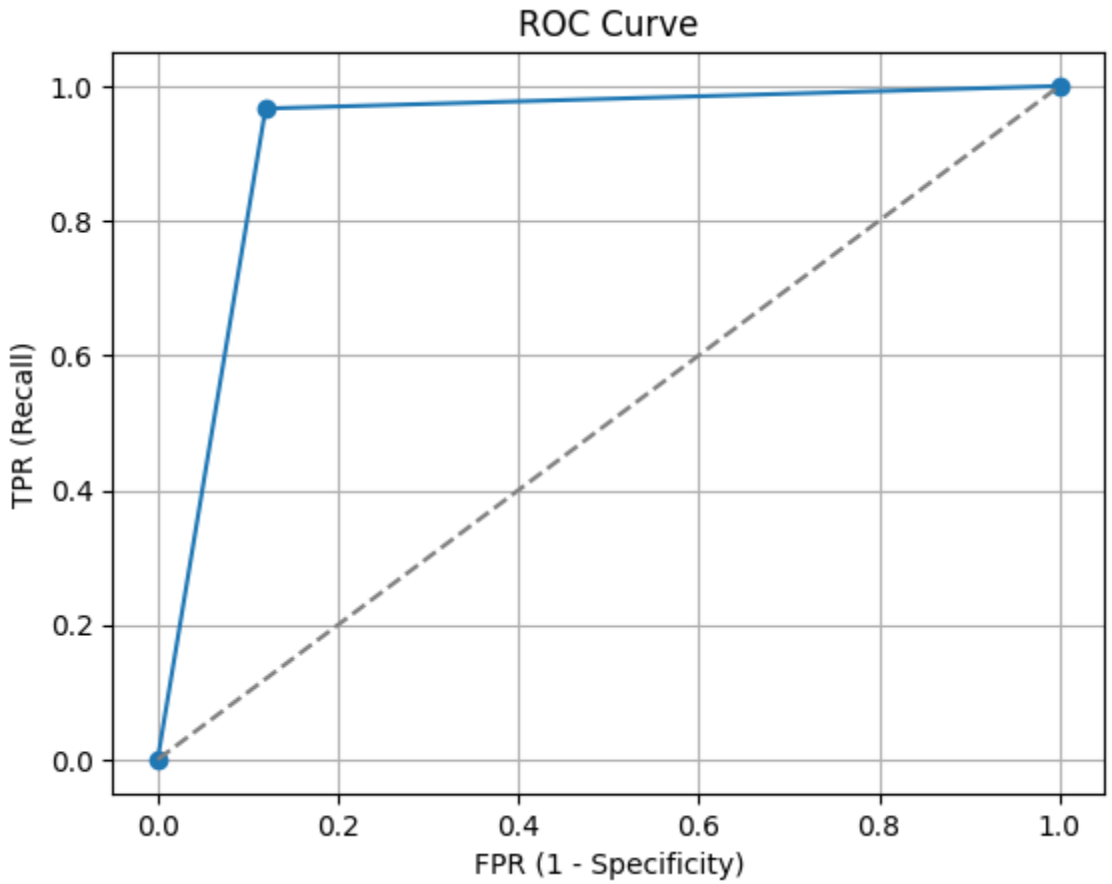


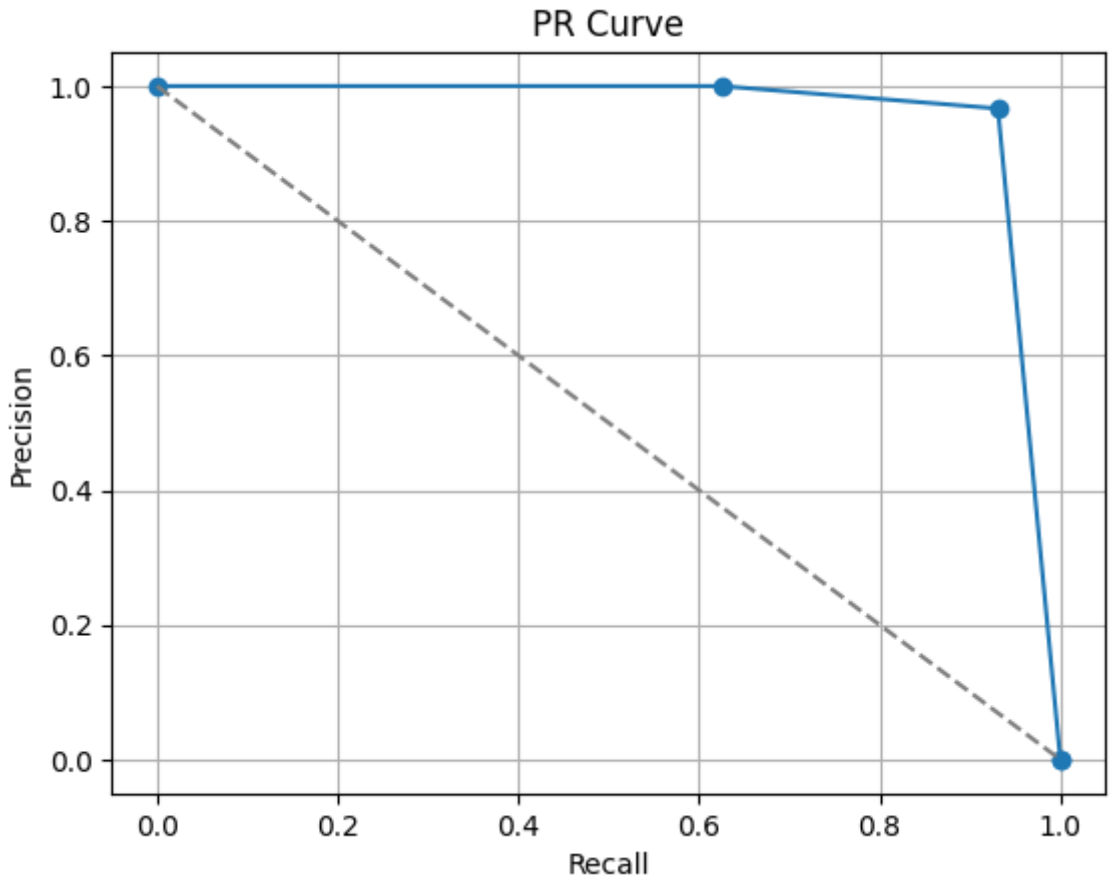


実験2および3は実験1に比べて

- NORMALをPNEUMONIAと誤分類した数が減少↓
- PNEUMONIAをNORMALと誤分類した数が増加↑

実験3における肺炎予測のROC曲線とPR曲線





考察

- 実験2および3のモデルは約93%のF値を達成した
  →この値が実用化に十分な値かどうかは別途考える必要がある
- 実験2および3におけるデータ拡張の有効性が証明された
- 実験3の全層ファインチューニングにおいて、検証誤差が下がったが、実験2に対する大きな性能差はなかった
- 実験3における肺炎予測の特異度は88%であった
  →100人のうち12人は肺炎じゃないのに肺炎と予測してしまう…大丈夫…？
- 予測を誤った例についてはデータを確認したほうが良いかも
  →もしかしたら人間が間違っていて、AIの予測が正しいかも…

などなど…

演習

- 肺炎を予測する多クラス分類AIモデルを構築する
- 実験条件を変えて実験する
- さまざまな評価指標を用いて評価し、性能を比較する

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

まとめ

- 胸部X線画像から肺炎を予測するAIモデルを構築した
- 条件を変えて複数の実験を行った
- さまざまな評価指標を用いてモデルの性能を評価した

【次回】
- 総括
- 医用画像AIの課題と展望
- （余裕があれば）AIによる肺のセグメンテーションを実験する