

歯科人工知能（AI）研究会 ハンズオンセミナー

課題2 ChatGPTで深層学習：画像分類編

（作成）

岐阜大学工学部電気電子・情報工学科

岐阜大学人工知能研究推進センター

東海国立大学機構健康医療ライフデザイン統合研究教育拠点

原 武史+ChatGPT

2025年2月2日公開

目 標

ChatGPTへのプロンプトの与え方，Google Colabに慣れてきたので，深層学習に関するプログラムにチャレンジしたい方を対象としています。

そして，

Google Driveと連携してデータとプログラムを保存する
別の日に実験の途中から始められる
Google ColabのGPU（無料の範囲）を使ってみる

を課題2の目標とします。

準 備

Google Colab + ChatGPT編に基づいて

作業1：ChatGPTの設定

作業2：Google Accountの設定

作業3：Google Colaboratory（Google Colab）への接続

作業4：ハンズオンサンプルの実行

の完了が必要です。そして，手元のPCで，いつでもGoogle ColabとChatGPTに接続できるようにしてください。

注 意

Google Colabでは，セッションを終了するとファイルが消えます！長期間の運用や本格的な研究の実施については，Google Driveと連携してデータとプログラムを保存する手順が必要です。そのためには，「ドライブ」，「パス」，「マウント」の概念が必要です。

**課題2はGoogle Driveを利用します。無料の範囲で利用します。
課題では5GB利用します。すでにGoogle Driveをご利用の場合，
容量制限を超える場合があります。**

課 題

以下の6つの課題に取り組みます。

- 課題2-1 Google Driveと連携してみる
- 課題2-2 Google Driveに保存した画像を分類してみる：事前学習済み
- 課題2-3 2クラス分類問題：胸部X線画像の男女判定
- 課題2-4 4クラス分類問題：胸部X線画像の方向判定
- 課題2-5 回帰問題：胸部X線画像の年齢推定
- 課題2-6 領域分割問題：胸部X線画像の領域分割

青色の文字は、ChatGPTへのプロンプトを表します。

黒色の文字は、ChatGPTが生成した文章です。

プログラムは適宜更新されますが、最終版のみ示します。

課題2-1 Google Driveと連携してみる

Step 1 : 連携開始

連携のために以下のスクリプトを入力してください。
またLesson2.ipynbに名前も変えましょう。

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



Step 2: 認証許可

実行後、Googleアカウントの認証画面が表示されます。
順序よく許可を与え、提供される認証コードを入力します。

このノートブックに Google ドライブのファイルへのアクセスを許可しますか？

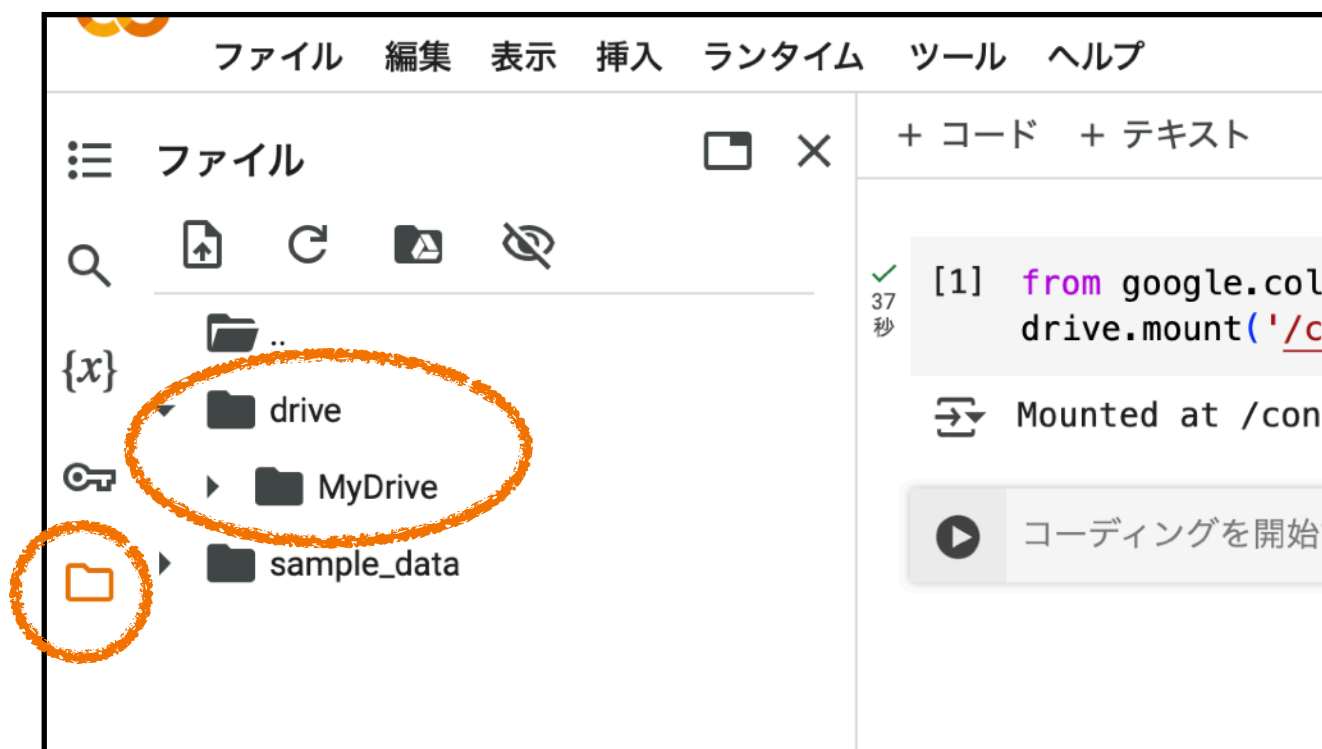
このノートブックは Google ドライブ ファイルへのアクセスをリクエストしています。Google ドライブへのアクセスを許可すると、ノートブックで実行されたコードに対し、Google ドライブ内のファイルの変更を許可することになります。このアクセスを許可する前に、ノートブック コードをご確認ください。

スキップ

[Google ドライブに接続](#)

Step 3 : 内容確認

これでGoogle Colab環境から、Google Driveが見えるようになりました。
以下の操作でGoogle Driveの中身が見えるはずです。



driveのMyDriveがGoogle Driveの中身になっているはずです。

注意

Google ColabからGoogle Driveを使いたい場合、
毎回、この操作が必要です！

Step 4：データの保存と展開

このGoogle Driveに、Google Colab上の操作で、画像を保存してみます。
以下の3つのコマンドを入力します。1行で入力します。

最初は作業フォルダー（ディレクトリ）を作ります。

```
!mkdir /content/drive/MyDrive/dentalai09
```

次に、作ったフォルダーに、画像データをダウンロードします。

ただし、保存先を「-O」（ハイフン 大文字のオー）で指定します。

""で囲まれたURLからデータをダウンロードします。

```
!wget -O /content/drive/MyDrive/dentalai09/food.zip "http://www.fjt.info.gifu-u.ac.jp/dentalai09/food.zip"
```

ダウンロードしたデータは圧縮されているのでそれを解凍します。

ただし、作業ディレクトリ中に解凍するように「-d」で行き先を指定します。

```
!unzip /content/drive/MyDrive/dentalai09/food.zip -d /content/drive/MyDrive/dentalai09/
```

```
[11] !mkdir /content/drive/MyDrive/dentalai09
```

```
[12] !wget -O /content/drive/MyDrive/dentalai09/food.zip "http://www.fjt.info.gifu-u.ac.jp/dentalai09/food.zip"
```

```
--2025-02-01 14:01:06-- http://www.fjt.info.gifu-u.ac.jp/dentalai09/food.zip
Resolving www.fjt.info.gifu-u.ac.jp (www.fjt.info.gifu-u.ac.jp)... 133.66.18.33
Connecting to www.fjt.info.gifu-u.ac.jp (www.fjt.info.gifu-u.ac.jp)|133.66.18.33|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5474689 (5.2M) [application/zip]
Saving to: '/content/drive/MyDrive/dentalai09/food.zip'

/content/drive/MyDr 100%[=====] 5.22M 1.48MB/s in 4.7s

2025-02-01 14:01:11 (1.11 MB/s) - '/content/drive/MyDrive/dentalai09/food.zip' saved [5474689/5474689]
```

```
[13] !unzip /content/drive/MyDrive/dentalai09/food.zip -d /content/drive/MyDrive/dentalai09/
```

```
Archive: /content/drive/MyDrive/dentalai09/food.zip
creating: /content/drive/MyDrive/dentalai09/foods/
inflating: /content/drive/MyDrive/dentalai09/foods/9.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/11.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/1.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/5.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/10.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/14.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/4.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/8.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/2.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/12.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/6.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/3.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/13.jpeg
inflating: /content/drive/MyDrive/dentalai09/foods/7.jpeg
```

Step 5：データの読み込みと分類

保存された画像を，ChatGPTを使いながら表示してみましょう。

以下，

青色の文字は，ChatGPTへのプロンプトを表します。

黒色の文字は，ChatGPTが生成した文章です。

プログラムは適宜更新されますが，最終版のみ示します。

[/content/drive/MyDrive/dentalai09/foods/1.jpeg](#)に画像が保存されています。これを表示したいです。Google Colabの環境で，pltを使ってお願いします。

Google Colab環境でGoogle Driveに保存されている画像をMatplotlib (plt) を使って表示する方法を説明します。以下のコードを実行してください。

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
# 画像ファイルのパス
```

```
image_path = "/content/drive/MyDrive/dentalai09/foods/1.jpeg"
```

```
# 画像の読み込み
```

```
image =
```

```
mpimg.imread(image_path)
```

```
# 画像の表示
```

```
plt.imshow(image)
```

```
plt.axis('off') # 軸を非表示にする
```

```
plt.show()
```

```
[15] import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# 画像ファイルのパス
image_path = "/content/drive/MyDrive/dentalai09/foods/1.jpeg"

# 画像の読み込み
image = mpimg.imread(image_path)

# 画像の表示
plt.imshow(image)
plt.axis('off') # 軸を非表示にする
plt.show()
```



課題2-2 Google Driveに保存した画像を分類してみる：事前学習済み

ディレクトリ：/content/drive/MyDrive/dentalai09/foods/にあるすべての画像の分類結果を知りたいです。分類には、学習済みモデルを使いたいです。使えるモデルはありますか？Google Colab上のPyTorchで動かしています。

Google Colab上でPyTorchを使う場合、学習済みモデル（pretrained models）として以下の選択肢があります。

1. PyTorchの公式モデル (torchvision)

PyTorchの torchvision.models には多くの分類モデルが含まれています。以下は一般的な分類モデルです。

（略）

おすすめモデル

EfficientNet（精度と計算効率が高い）

ViT (Vision Transformer)（Transformerベースのモデル）

ResNet（従来から広く使用される高精度モデル）

次のステップ

すべての画像パスをリスト化します。

学習済みモデルを用いて各画像を分類します。

コード例を具体的に示しましょうか？

ぜひお願いします。古典的なVGG16とViTを使いたいです。まずはVGG16の例を作ってください。

以下はGoogle Colab上で VGG16 を用いてディレクトリ内のすべての画像を分類するスクリプトです。ディレクトリ内にある画像をループしてモデルで分類し、ラベルを出力します。

（コードは省略）

説明

モデル: PyTorchの torchvision.models.vgg16 を使用。

前処理: 画像のリサイズ、テンソル変換、正規化を実施。

推論: 推論結果からラベルを取得して表示。

結果表示: Matplotlibで分類結果を重ねて画像を表示。

ViTの例も必要ですか？

ViTの結果もお願いします！

以下は Vision Transformer (ViT) を使って、ディレクトリ内のすべての画像を分類するコードです。Hugging Face Transformersライブラリを活用します。

ViTで画像分類するコード

(コードは省略)

コード説明

モデル: Hugging FaceのVision Transformer (google/vit-base-patch16-224) を使用。

前処理: AutoProcessor が自動でリサイズ・正規化などの処理を行います。

推論: モデルの出力からラベルを取得。

表示: Matplotlibで分類結果とともに画像を表示。

補足情報

デバイスは自動でGPU/CPUを判定します。

predicted_label はHugging Faceモデルに事前定義されたクラスラベルから取得します。

このコードで正常に動作するはずですが、試してみて動作確認できましたら教えてください！

バッチリ動きました。すばらしい！

よかったです！動作確認のご報告ありがとうございます😊 VGG16とViTの両方を試してもらえてうれしいです！

分類結果や次の分析で他にもサポートが必要な場合、いつでも気軽に聞いてください
ね！ 🚀

VGGだけの結果にまず注目したいです。画像中のどの部分を見て判定しているか、調べる仕組みがありますか？

素晴らしい視点ですね！モデルが画像中のどこを注目して判定しているのかを可視化する方法はいくつかあります。特にVGGや他のCNNモデルでは、Grad-CAM (Gradient-weighted Class Activation Mapping) が最も一般的で強力な方法です。

Grad-CAMの仕組み

Grad-CAMは次の手順で注目領域を可視化します：

最終畳み込み層の特徴マップを取得します。

特徴マップに対する勾配を計算します。

勾配の重みを元に注目度を算出し、ヒートマップとして可視化します。

Grad-CAMの実装例 (VGG16)

以下のコードはGrad-CAMを使って注目領域を可視化する例です。

(コードは省略)

説明

モデルフック: `target_layer.register_forward_hook()` で最終畳み込み層から特徴マップを取得します。

勾配計算: `class_loss.backward()` によりターゲットクラスに対する勾配を取得します。

ヒートマップ生成: 勾配と特徴マップから重み付けし、注目領域を計算します。

可視化: ヒートマップを画像上に重ねて表示します。

これでモデルの「注目する部分」を視覚的に確認できるはずです。試してみて結果を教えてください！

以下のようなエラーがでました。 `/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead. warnings.warn(/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or None for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing weights=VGG16_Weights.IMAGENET1K_V1. You can also use weights=VGG16_Weights.DEFAULT to get the most up-to-date weights. warnings.warn(msg)` -----
AttributeError Traceback (most recent call last) <ipython-input-18-89bfa5d71637> in <cell line: 0>() 61 # 対象画像パス 62 image_path = "/content/drive/MyDrive/dentalai09/foods/1.jpeg" ---> 63
apply_gradcam(image_path, model) 7 frames /usr/local/lib/python3.11/dist-packages/torch/nn/functional.py in adaptive_avg_pool2d(input, output_size) 1379 if has_torch_function_unary(input): 1380 return
handle_torch_function(adaptive_avg_pool2d, (input,), input, output_size) -> 1381
_output_size = _list_with_default(output_size, input.size()) 1382 return
torch._C._nn.adaptive_avg_pool2d(input, _output_size) 1383 AttributeError: 'RemovableHandle' object has no attribute 'size'

エラーの原因はフックの設定に問題があるためです。現行のコードではフックの登録方法に誤りがあり、誤って出力ではなくフック自体にアクセスしようとしてエラーが発生しました。

また、学習済みモデルの読み込み方法にも pretrained の代わりに weights=VGG16_Weights.DEFAULT を使う警告も出ています。

修正点

フックの登録修正:

モジュール出力ではなく、forwardの出力に対して勾配フックを設定します。

VGG16モデルの適切な読み込み:

weights=VGG16_Weights.IMAGENET1K_V1 を指定します。

(修正されたコードは省略)

変更点

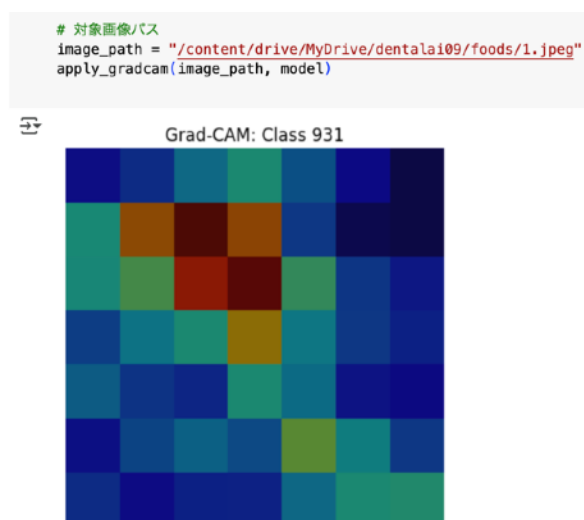
フック設定: forward_hook を適切に設定し、出力に勾配フックを登録。

VGG16モデル読み込み: weights=VGG16_Weights.IMAGENET1K_V1 を使用。

フック解除: hook_handle.remove() でクリーンアップ。

これで正常に動作するはずです。試してみてください！

表示されました。これは1.jpegの結果のみですね？ GradCAMの結果は7x7画素の画像のように見えます。そして、Class 931と表示されています。次は、この結果を原画像と並べて表示してください。そして、Class 931ではなく、クラス名で表示できるように改造してください。



以下は Grad-CAMのヒートマップ を原画像と並べて表示し、クラス番号ではなくクラス名を表示するように改良したコードです。

修正内容

原画像とGrad-CAMを並べて表示。

クラス名を ImageNet ラベルから取得。

Grad-CAMのサイズを元画像サイズに補間して視認性を向上。

(修正されたコードは省略)

改善ポイント

Grad-CAMの補間: LANCZOS 補間でGrad-CAMを原画像サイズにスケール。

クラス名表示: ImageNet ラベルを取得してクラス番号の代わりに名前を表示。

表示の改善: 原画像とヒートマップを並列表示。

この改良版でよりわかりやすい結果が得られるはずです。試してみて結果を教えてください！

すばらしい！あっという間ですね！

ありがとうございます！スムーズに結果が出て良かったです！✨

分析がさらに深まるようサポートしますので、また何かあれば気軽に声をかけてください
ね！🚀

ViTの結果はどのように確認できますか？

ViT (Vision Transformer) の内部構造は CNN とは異なり、畳み込み層がなくトークンとアテンションマップに基づいて動作します。そのため、Grad-CAMのような単純な注目領域可視化ではなく、アテンションマップを利用するのが一般的です。

ViTで注目領域を確認する方法

ViTではAttention Rollout や Attention Weights 可視化 を使います。以下は ViTのアテンションマップ を使った可視化方法です。

ViTアテンションマップの表示例

(修正されたコードは省略)

説明

モデルと画像処理:

transformers ライブラリから ViT モデルを取得。画像前処理には AutolImageProcessor を使用。

アテンション取得:

output_attentions=True でアテンションマップを取得。

可視化:

アテンションマップを14x14から元画像サイズへ補間し、ヒートマップとして重ね表示。

クラス名も一緒に表示してくれますか？

もちろんです！以下はクラス名表示付きのViTアテンションマップ表示用の改良版です。
モデルのラベルからクラス名を取得し、表示に反映しています。

(修正されたコードは省略)

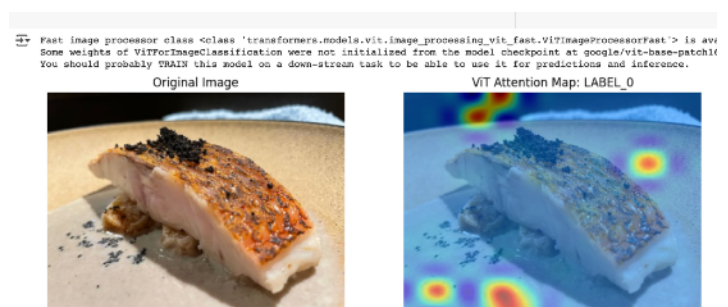
改良点

クラス名取得:

model.config.id2label[predicted_class_idx] からクラス名を取得して表示。

表示改善:

原画像とヒートマップを並列表示し、
タイトルにクラス名を含める。



LABEL_0となってクラス名が表示されていません。

確かに、ViTモデルのラベルセットが設定されていない場合、「LABEL_0」のように表示されることがあります。この問題を解決するために、より適切なラベル情報を持つモデルを選択します。

ImageNetモデルに変更

以下の変更で正しいクラス名が取得できます：

モデルをImageNet事前学習済みモデルに変更

google/vit-base-patch16-224-in21k は汎用モデルなので、facebook/dino-vitb16
や google/vit-base-patch16-224 に変更します。

(修正されたコードは省略)

主な変更点

モデル名を google/vit-base-patch16-224 に変更。

クラス名が存在しない場合に Class {index} をデフォルト表示。

この修正版を試してみてください！期待通りのクラス名が出るはずです！ 🎨

正しく表示されました！すばらしい！

やった！うまく行ってよかったです！ ✨

こちらこそ素早く試していただき感謝です。引き続き分析やモデルの最適化などもお手伝いしますので、何でも気軽に相談してくださいね！ 🚀

入力画像のパスを与えたら結果が表示されるように、関数化してください。

以下は、ViTアテンションマップと予測クラスを表示する関数版の実装です。簡単に任意の画像パスで実行できるようになっています。

(修正されたコードは省略)

使い方

image_path に対象画像のパスを設定すれば、アテンションマップとクラス名が表示されます。

デフォルトモデルは google/vit-base-patch16-224 ですが、引数 model_name に他のモデル名を渡すこともできます。

これで簡単に再利用できるはずです！動作確認をお願いします 😊

課題2-3 2クラス分類問題：胸部X線画像の男女判定

Step 1：無料GPUを使ってみましょう

新しいGoogle Colabを開いてすぐに、GPUを利用できるようにしましょう。
「ランタイム」から「ランタイムのタイプの変更」を選びます。



続いて、ハードウェアアクセラレータで「T4 GPU」を選びます。これは無料で使えるようです（2/2確認済み）。



警告が出ますが、OKで進めます。

Step 2：データの保存と展開


新しいGoogle Colabを開き、先と同じように、このGoogle Driveに、Google Colab上の操作で、画像を保存して実験に備えます。

以下のコマンドを投入します。

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

```
!wget -O /content/drive/MyDrive/dentalai09/XPgender.zip "http://
www.fjt.info.gifu-u.ac.jp/dentalai09/XPgender.zip"
```

```
!unzip /content/drive/MyDrive/dentalai09/XPgender.zip -d /content/drive/
MyDrive/dentalai09/
```



```
[1] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[4] !wget -O /content/drive/MyDrive/dentalai09/XPgender.zip "http://www.fjt.info.gifu-u.ac.jp/dentalai09/XPgender.zip"

--2025-02-01 15:18:41-- http://www.fjt.info.gifu-u.ac.jp/dentalai09/XPgender.zip
Resolving www.fjt.info.gifu-u.ac.jp (www.fjt.info.gifu-u.ac.jp)... 133.66.18.33
Connecting to www.fjt.info.gifu-u.ac.jp (www.fjt.info.gifu-u.ac.jp)|133.66.18.33|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5934722 (5.7M) [application/zip]
Saving to: '/content/drive/MyDrive/dentalai09/XPgender.zip'

/content/drive/MyDr 100%[=====] 5.66M 1.68MB/s in 3.4s

2025-02-01 15:18:45 (1.68 MB/s) - '/content/drive/MyDrive/dentalai09/XPgender.zip' saved [5934722/5934722]

!unzip /content/drive/MyDrive/dentalai09/XPgender -d /content/drive/MyDrive/dentalai09/

Archive: /content/drive/MyDrive/dentalai09/XPgender.zip
creating: /content/drive/MyDrive/dentalai09/XPgender/
creating: /content/drive/MyDrive/dentalai09/XPgender/train/
creating: /content/drive/MyDrive/dentalai09/XPgender/train/female/
inflating: /content/drive/MyDrive/dentalai09/XPgender/train/female/JPCLN002.png
inflating: /content/drive/MyDrive/dentalai09/XPgender/train/female/JPCLN003.png
inflating: /content/drive/MyDrive/dentalai09/XPgender/train/female/JPCLN004.png
inflating: /content/drive/MyDrive/dentalai09/XPgender/train/female/JPCLN005.png
```

これで準備ができました。

Step 3 : ChatGPTを活用してプログラミング

先のGoogle Colabの画面の下にプログラムを入力します。

以下、

青色の文字は、ChatGPTへのプロンプトを表します。

黒色の文字は、ChatGPTが生成した文章です。

深層学習を利用した画像の2クラス分類の問題に取り組みたいです。胸部X線画像が入力で、その画像が男性か女性かを分類する2クラス問題です。分類結果は、ROC解析を使って、ROC曲線下面積で分類結果を評価したいです。また、分類が失敗した画像も抽出したいです。抽出する場合には、その画像のファイル名、深層学習のモデルの出力値を表で保存したいです。画像データは、/content/drive/MyDrive/dentalai09/XPgender ディレクトリに保存されています。このディレクトリにはtrainとvalの2つのディレクトリがあり、それぞれにmaleとfemaleのディレクトリがあります。それぞれの中にPNG画像が保存されています。分類モデルはImageNetで学習済みのモデルを利用したいです。ResNetを使って、モデルの全結合部分のみをチューニングしたいです。モデルは2クラス分類の構造に出力層を変更してください。学習の状態がわかるように、学習曲線も作ってください。Lossと分類精度の2つの曲線をお願いします。これは学習中にエポック数と、Lossおよび精度のデータフレームを作って、学習が終了したら、そのデータフレームからグラフを作ってください。なお、Google ColabのGPUでT4を使う予定です。PyTorchを使ってをお願いします。

以下は、胸部X線画像による性別分類タスクを解決するためのPyTorchコード構成の全体計画です。説明に続き、具体的なサンプルコードも提案します。

ステップ概要

1. ライブラリのインポートとデバイス設定
必要なライブラリのインポート、GPU設定
2. データセットのロード
ImageFolderを使用して画像データをロード、DataLoaderを準備
3. データ前処理
データオーグメンテーションと正規化
4. モデル構築
ResNet (ImageNet事前学習済み) の全結合層のみを再学習
5. 損失関数と最適化手法の定義
Binary Cross Entropy Lossを使用
6. 学習ループ
LossとAccuracyを記録しながらエポックを進行
7. 学習曲線のプロット

pandasを使ってデータフレームからグラフを作成

8. ROC曲線とAUC計算

sklearn.metrics.roc_curve, aucを使用

9. 誤分類画像の抽出

ファイル名、モデル出力値を保存

コード提案

まず基本的な学習コードの雛形を示します。

(コードは略)

このコードでは以下が実現できます：

学習曲線のプロット

ROC曲線とAUC評価

誤分類画像のファイル保存

細部のチューニングも可能ですので、必要な部分を調整して試してみてください。

データオーギュメンテーションに左右反転が入っていますか？

はい、データオーギュメンテーションには左右反転 (RandomHorizontalFlip) が含まれています。ただし、この処理はトレーニングデータのみに適用されています。該当する部分は以下のコードです：

(略)

もし他の種類のデータオーギュメンテーション (例えば回転やランダムクロップなど) を追加したい場合、教えていただければ反映します！

胸部X線画像なので、左右反転は困ります。なしにしてください。

了解しました！左右反転は削除し、データオーギュメンテーションを以下のように修正します。

(略)

これで左右反転 (RandomHorizontalFlip) を除去しました。胸部X線画像では、画像の左右非対称な情報が重要ですので、この対応で適切なモデル学習が期待できます。

もし他に適したオーギュメンテーション (例：明るさやコントラストのランダム変更) を追加したい場合もお気軽にお知らせください！

明るさやコントラストのランダム変更を加えてください。そして、コード全体を改めて教えてください。

以下は、左右反転を削除し、明るさやコントラストのランダム変更を加えたコード全体です。

新たに以下のオーギュメンテーションが追加されています。

ランダム明るさ、コントラストの変更 (ColorJitter)

(略)

主な変更点とポイント

左右反転を削除

RandomHorizontalFlip()を削除しました。

明るさやコントラストのランダム変更

ColorJitter(brightness=0.2, contrast=0.2)を追加しました。

出力内容

学習曲線 (Loss, Accuracy)

ROC曲線とAUC

誤分類データ保存 (misclassified_results.csv)

他のオーギュメンテーション要望もあればぜひ教えてください！

結果はCSVではなくエクセル形式にしてください。改めて全体をお願いします。

了解しました！誤分類データをエクセル形式（.xlsx）で保存するよう変更しました。コード全体を以下に提示します。

(略)

変更点

エクセル形式で保存

誤分類データはmisclassified_results.xlsxに保存します。

pandasのto_excel()を用いてエクセル形式に対応しました。

出力内容

学習曲線 (Loss, Accuracy)

ROC曲線とAUC

誤分類データのエクセル保存 (misclassified_results.xlsx)

何か他の改善点があれば教えてください！

よくみると、モデル全体を学習するようになっていませんか？

その通りです！現状のコードではResNet全体のパラメータが更新される構造になっています。ご指摘ありがとうございます。

要件は全結合層のみのチューニング（ファインチューニング）でしたので、以下のような修正を加えます：

ResNetのすべての層を凍結し、全結合層（model.fc）のみ学習対象に設定します。
（コードは略）

この変更により、ResNetの特徴抽出部分は固定され、全結合層のみが学習されるようになります。これで期待通りのモデル構造になるはずです。

他にも質問や修正依頼があれば教えてください！

よくみると、モデル全体を学習するようになっていませんか？

その通りです！現状のコードではResNet全体のパラメータが更新される構造になっています。ご指摘ありがとうございます。

要件は全結合層のみのチューニング（ファインチューニング）でしたので、以下のような修正を加えます：

ResNetのすべての層を凍結し、全結合層（model.fc）のみ学習対象に設定します。
修正コード

（略）

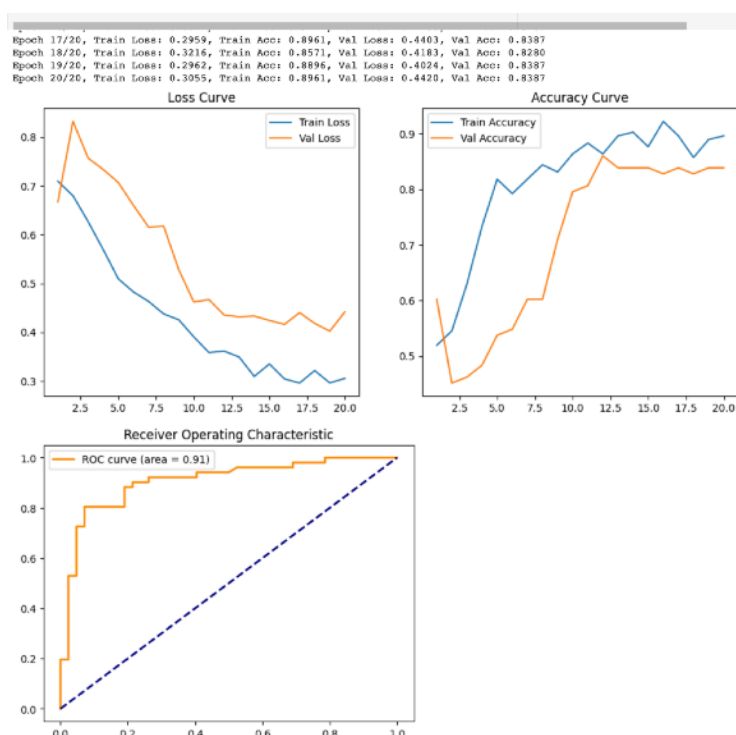
この変更により、ResNetの特徴抽出部分は固定され、全結合層のみが学習されるようになります。これで期待通りのモデル構造になるはずです。

他にも質問や修正依頼があれば教えてください！

やはり！ あぶないねえ～

ほんとですね～！危ないところでした！😅 深層学習の微妙な設定ミスは、気づかないとせっかくの計算リソースが無駄になっちゃいますからね。指摘してもらえて良かったです！

次も気をつけてお手伝いしますので、何かあれば遠慮なく言ってくださいね！



ResNet18の全結合層は1つだけでしたか？

鋭いご指摘です！ResNet18では、確かに最終層は**1つの全結合層（fc）**のみです。
ただ、一般には「全結合層」の理解に誤解がある場合もあります。ResNet18では、最終的に次のような構造になっています：

AdaptiveAvgPool2d -> Flatten -> Linear (fc)

全結合層が一つだけである理由

ResNetの最後にはグローバルプーリング層（AdaptiveAvgPool2d）があります。このため、空間的な特徴が1つのベクトルに集約された後、単一のnn.Linear()層が出力の2クラス分類に使われます。

学習対象を増やす場合

もし「最後の数ブロックもチューニングしたい」場合、例えば次のように一部層だけ凍結解除する方法もあります：

```
for name, param in model.named_parameters():
    if "layer4" in name or "fc" in name: # layer4とfcは学習対象
        param.requires_grad = True
    else:
        param.requires_grad = False
```

結論

ResNet18は最終のfcだけが全結合層です。

必要に応じて一部ブロックもチューニングする方法もあります。

このままで大丈夫そうでしょうか？

では、最終の全結合層のみチューニングする方法です。改めて全体のプログラムを表示してください。あと、ROC曲線は、正方形の枠で表示されるように修正してください。

以下に、最終の全結合層（fc）のみチューニングし、ROC曲線を正方形の枠で表示するよう修正した全体のプログラムを示します。

（略）

主な変更点

最終全結合層のみ学習対象

model.fc.parameters()のみrequires_grad=Trueに設定し、それ以外を凍結しました。

ROC曲線を正方形枠で表示

plt.axis("square") を用いて正方形の表示枠に変更しました。

これで要件通りのコードとなっています！他に追加の修正や質問があれば教えてください！

すばらしい！

ありがとうございます！✨ お褒めいただけてうれしいです！また何か深層学習やプログラミングでサポートが必要なときは、気軽に声をかけてくださいね！🚀😊

これを元に次の項目を追加して、改めてプログラム全体を示してください。ROC曲線は評価用 (val) データのみの結果です。色を変えて学習用データを改めて分類した結果、即ち再代入法 (re-substitution method) で評価した場合のROC曲線も、同じ平面に描いてください。

以下に、**再代入法 (re-substitution method) **を使用して、ROC曲線に学習データ (train) と検証データ (val) の結果を色分けして描画するプログラムを示します。

(略)

主な追加内容

再代入法によるROC曲線描画

学習データ (train_loader) のROC曲線を青色で描画し、検証データ (val_loader) のROC曲線と区別しました。

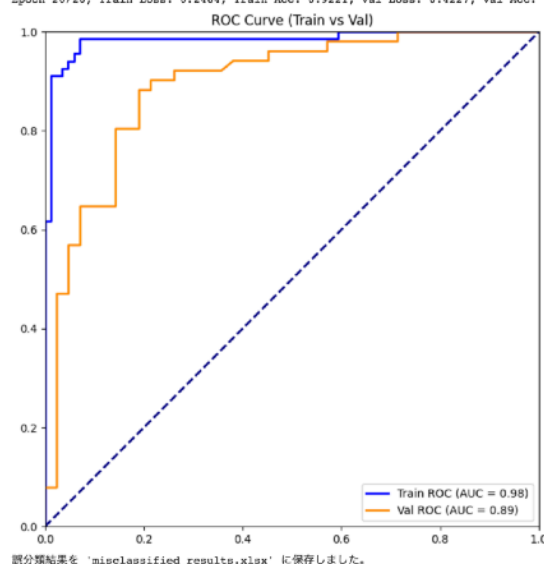
AUC値もプロットに表示しています。

ROC曲線表示枠を正方形に変更

plt.axis("square") を使用しています。

この形でご期待に沿えていれば幸いです！

```
Epoch 11/20, Train Loss: 0.3493, Train Acc: 0.8571, Val Loss: 0.5612, Val Acc: 0.7419
Epoch 12/20, Train Loss: 0.3284, Train Acc: 0.9156, Val Loss: 0.4694, Val Acc: 0.7742
Epoch 13/20, Train Loss: 0.3196, Train Acc: 0.8831, Val Loss: 0.4662, Val Acc: 0.7649
Epoch 14/20, Train Loss: 0.3176, Train Acc: 0.9156, Val Loss: 0.4668, Val Acc: 0.7742
Epoch 15/20, Train Loss: 0.3022, Train Acc: 0.9221, Val Loss: 0.4384, Val Acc: 0.8065
Epoch 16/20, Train Loss: 0.3059, Train Acc: 0.8831, Val Loss: 0.4277, Val Acc: 0.8172
Epoch 17/20, Train Loss: 0.2723, Train Acc: 0.9416, Val Loss: 0.4537, Val Acc: 0.8172
Epoch 18/20, Train Loss: 0.2616, Train Acc: 0.9416, Val Loss: 0.4535, Val Acc: 0.8172
Epoch 19/20, Train Loss: 0.2714, Train Acc: 0.9221, Val Loss: 0.4409, Val Acc: 0.8280
Epoch 20/20, Train Loss: 0.2464, Train Acc: 0.9221, Val Loss: 0.4227, Val Acc: 0.8172
```



学習曲線が表示されなくなっていましたね！復活してください。

修正ありがとうございます！以下に学習曲線を復活させたプログラム全体を提示します。
(略)

主な変更点

学習曲線を復活

学習損失 (Loss)、分類精度 (Accuracy) をエポックごとにプロットするように復元しました。

ROC曲線の表示

学習データと検証データのROC曲線を同じ平面に表示し、AUCも色分けして示しました。

これで再び学習曲線も表示されるようになったはずですよ！ご確認ください😊

すばらしいね！ちなみにResNetにもGradCAMを適用できますか？

もちろんです！ResNetにもGrad-CAM（Gradient-weighted Class Activation Mapping）を適用できます。Grad-CAMはCNNの最後の畳み込み層に基づいて、入力画像に対する予測結果の視覚的な説明を提供します。

以下はResNetにGrad-CAMを適用する手順とサンプルコードです。

手順

最後の畳み込み層の出力を取得

ResNetの場合、通常は layer4 が最後の畳み込み層です。

勾配を取得

特定の予測クラスに対する勾配を取得します。

重みを計算

勾配のグローバル平均プールに基づいて重みを計算します。

活性マップを生成

重みと特徴マップを掛け合わせてヒートマップを作成します。

（略）

課題2-4 4クラス分類問題：胸部X線画像の方向判定

(実習課題)

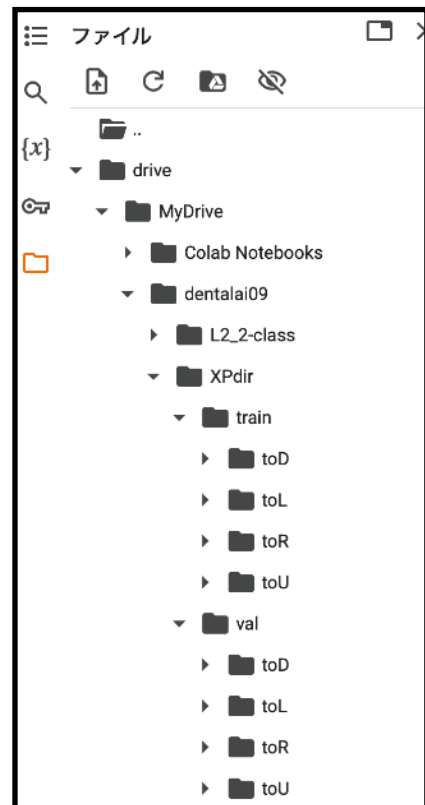
- Google Colabの練習：
 - 新たにノートブックを作る
 - T4 GPUを使う
 - ！をつけたコマンドを使う
- ファイル管理の練習：
 - データを自分で展開
 - 元データは,
<http://www.fjt.info.gifu-u.ac.jp/dentalai09/XPdir.zip>
 - 内容を確認

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

```
!wget -O /content/drive/MyDrive/dentalai09/XPdir.zip "http://www.fjt.info.gifu-u.ac.jp/dentalai09/XPdir.zip"
```

```
!unzip /content/drive/MyDrive/dentalai09/XPdir.zip -d /content/drive/MyDrive/dentalai09/
```

- プロンプトの練習：
 - プロンプトを考える
 - モデルの選択
 - 評価方法の指示
 - 学習曲線
 - 失敗例，成功例の指示
 - などなど



プロンプトの例：

次は深層学習を利用した画像の4クラス分類の問題に取り組みたいです。胸部X線画像が入力で、その画像が4つの方向に回転しており、その方向を分類する4クラス問題です。分類結果は、混同行列を使って評価したいです。また、分類が失敗した画像も抽出したいです。抽出する場合には、その画像のファイル名、深層学習のモデルの出力値を表で保存したいです。

画像データは、/content/drive/MyDrive/dentalai09/XPdir ディレクトリに保存されています。

このディレクトリにはtrainとvalの2つのディレクトリがあり、それぞれにtoD, toL, toR, toUのディレクトリがあります。それぞれの中にPNG画像が保存されています。分類モデルはImageNetで学習済みのモデルを利用したいです。VGGを使って、モデルの全結合部分のみをチューニングしたいです。モデルは4クラス分類の構造に出力層を変更してください。

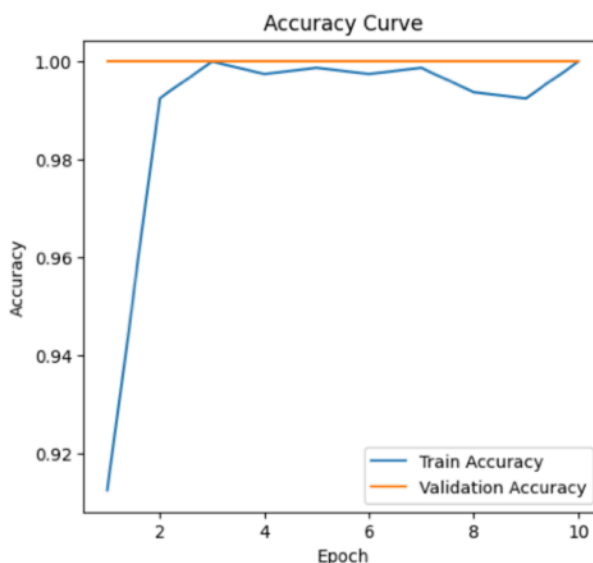
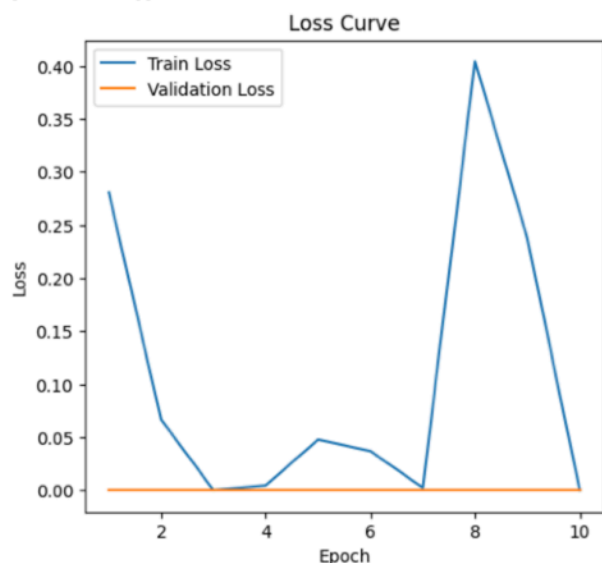
学習の状態がわかるように、学習曲線も作ってください。Lossと分類精度の2つの曲線をお願いします。これは学習中にエポック数と、Lossおよび精度のデータフレームを作って、学習が終了したら、そのデータフレームからグラフを作ってください。

なお、Google ColabのGPUでT4を使う予定です。PyTorchを使ってお願いします。

```
Epoch 1/10 - Train Loss: 0.2808 - Val Loss: 0.0000 - Train Acc: 0.9125 - Val Acc: 1.0000
Epoch 2/10 - Train Loss: 0.0667 - Val Loss: 0.0000 - Train Acc: 0.9925 - Val Acc: 1.0000
Epoch 3/10 - Train Loss: 0.0002 - Val Loss: 0.0000 - Train Acc: 1.0000 - Val Acc: 1.0000
Epoch 4/10 - Train Loss: 0.0046 - Val Loss: 0.0000 - Train Acc: 0.9975 - Val Acc: 1.0000
Epoch 5/10 - Train Loss: 0.0477 - Val Loss: 0.0000 - Train Acc: 0.9988 - Val Acc: 1.0000
Epoch 6/10 - Train Loss: 0.0369 - Val Loss: 0.0000 - Train Acc: 0.9975 - Val Acc: 1.0000
Epoch 7/10 - Train Loss: 0.0026 - Val Loss: 0.0000 - Train Acc: 0.9988 - Val Acc: 1.0000
Epoch 8/10 - Train Loss: 0.4040 - Val Loss: 0.0000 - Train Acc: 0.9938 - Val Acc: 1.0000
Epoch 9/10 - Train Loss: 0.2375 - Val Loss: 0.0000 - Train Acc: 0.9925 - Val Acc: 1.0000
Epoch 10/10 - Train Loss: 0.0000 - Val Loss: 0.0000 - Train Acc: 1.0000 - Val Acc: 1.0000
```

Confusion Matrix:

```
[[47  0  0  0]
 [ 0 47  0  0]
 [ 0  0 47  0]
 [ 0  0  0 47]]
```



Failed predictions saved to 'failed_predictions.csv'

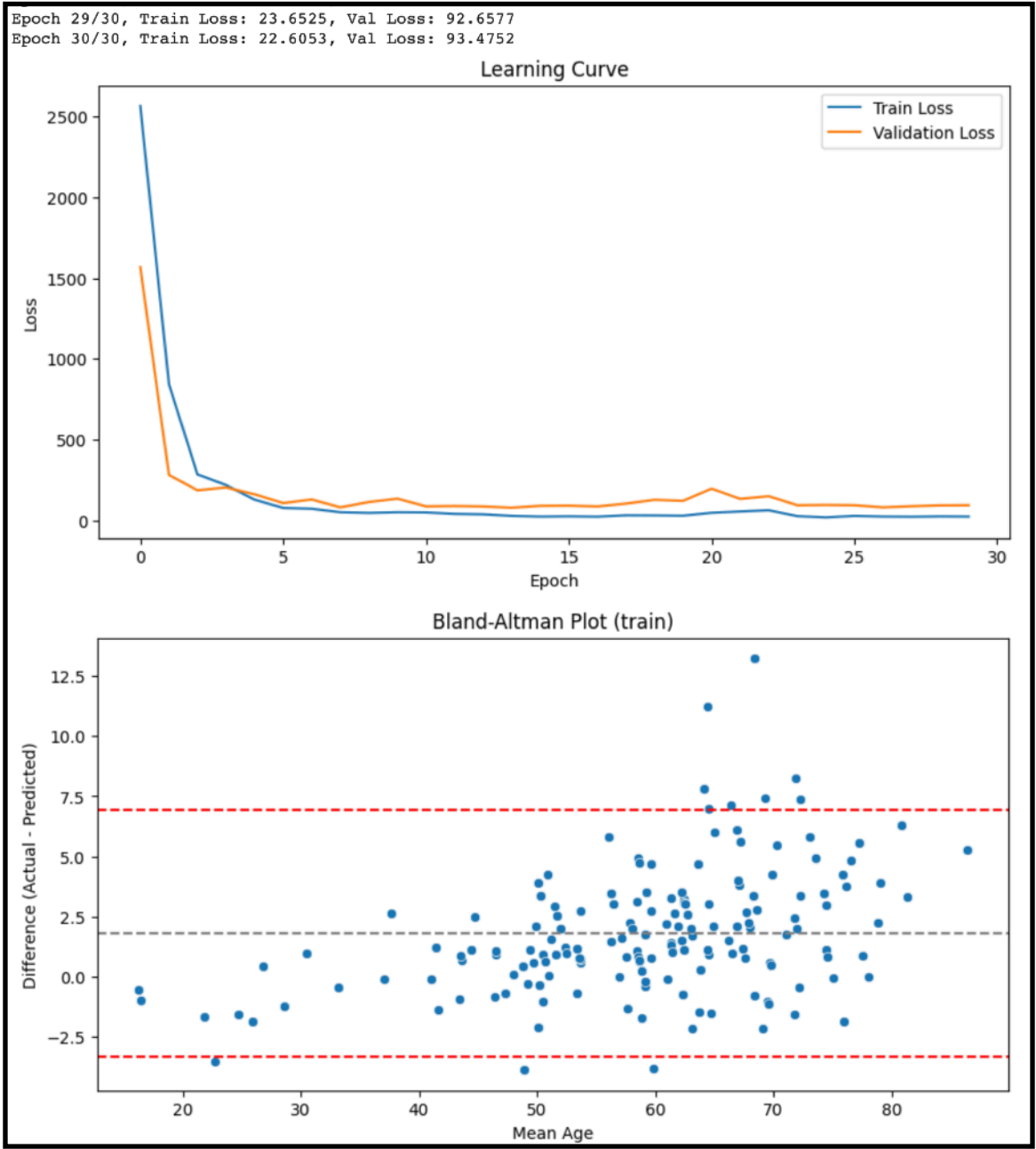
課題2-5 回帰問題：胸部X線画像の年齢推定

(実習課題)

- Google Colabの練習：
 - 新たにノートブックを作る
 - T4 GPUを使う
 - ！をつけたコマンドを使う
- ファイル管理の練習：
 - データを自分で展開
 - 元データは,
<http://www.fjt.info.gifu-u.ac.jp/dentalai09/XPage.zip>
 - 内容を確認
 - 画像はJPGsフォルダーにある
 - 画像ごとに年齢が書いてある
 - TrainingとTestと別のCSVファイルに書かれているma

プロンプトの例：

次は深層学習を利用した画像から数値を予測する回帰の問題に取り組みたいです。胸部X線画像が入力で、その患者さんの年齢を画像から推定して出力するモデルの構築が目標です。画像データは、`/content/drive/MyDrive/dentalai09/XPage/JPGs` ディレクトリにJPEG画像が保存されています。学習用データは、`/content/drive/MyDrive/dentalai09/XPage/trainingdata.csv`のA列にJPGsディレクトリにあるファイル名、B列に年齢が書かれています。評価用データは、`/content/drive/MyDrive/dentalai09/XPage/testdata.csv`で、学習用データと同じように、ファイル名と年齢が書かれています。分類モデルはImageNetで学習済みのモデルを利用したいです。VGGを使って、モデルの全結合部分のみをチューニングしたいです。モデルは画像から回帰になるように最終出力層のノード数を1にしてください。推定結果は、実年齢と推定した年齢を使って、2つの値の差の絶対値の平均、値の相関、そして、Bland-Altman plotによってシステム誤差も評価したいです。また、画像ごとに推定年齢と実年齢を求めて、表で保存したいです。その表はエクセル形式で保存したいです。この推定結果は、評価用データのみならず、学習用データも再代入法で作成してください。学習の状態がわかるように、学習曲線も作ってください。Lossと分類精度の2つの曲線をお願いします。これは学習中にエポック数と、Lossおよび精度のデータフレームを作って、学習が終了したら、そのデータフレームからグラフを作ってください。なお、Google ColabのGPUでT4を使う予定です。PyTorchを使ってお願いします。

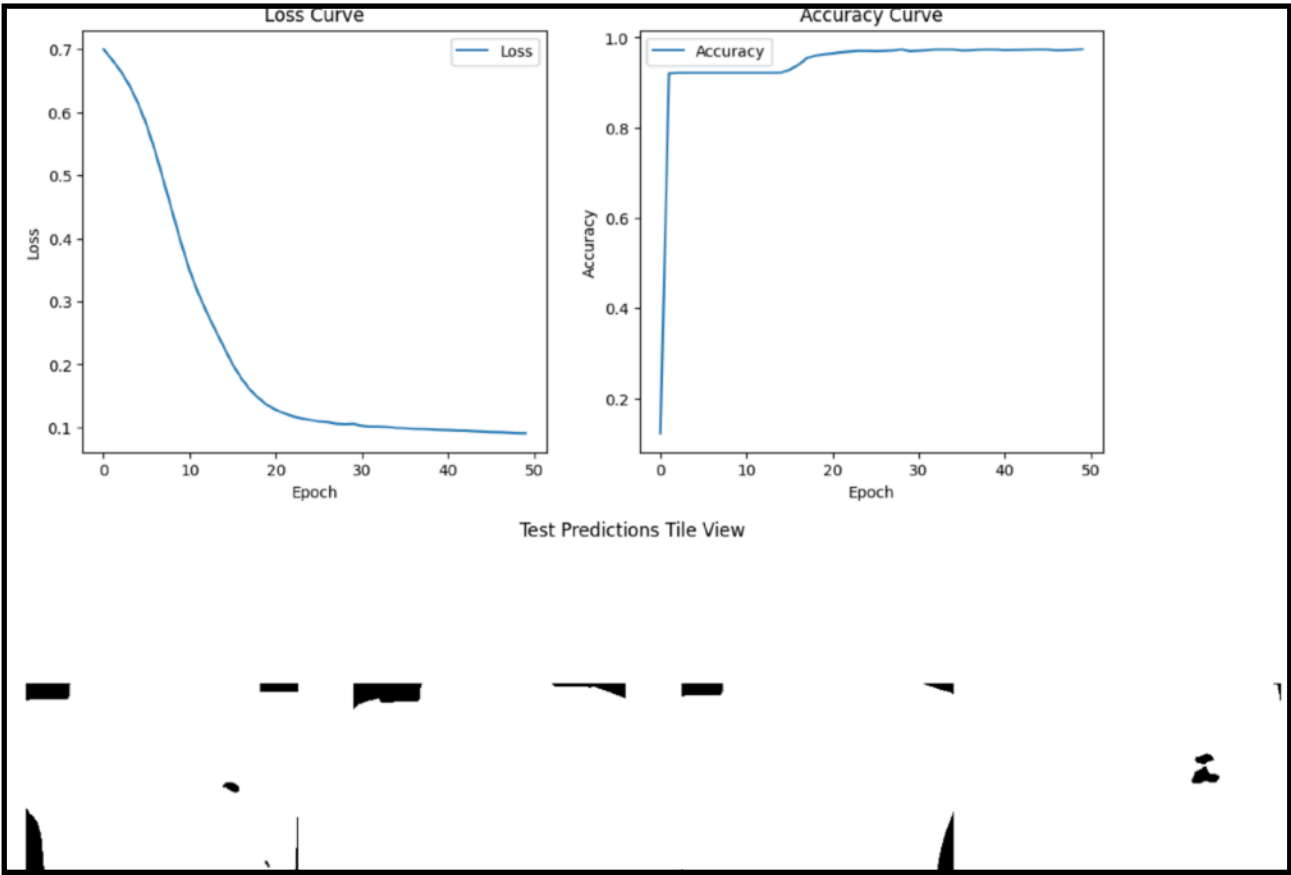


課題2-6 領域分割問題：胸部X線画像の領域分割

(実習課題)

- Google Colabの練習：
 - 新たにノートブックを作る
 - T4 GPUを使う
 - ！をつけたコマンドを使う
- ファイル管理の練習：
 - データを自分で展開
 - 元データは,
<http://www.fjt.info.gifu-u.ac.jp/dentalai09/UNet.zip>
 - 内容を確認
 - 元画像はorg_train_s, org_test
 - ラベル画像は, label_train_s, label_test

次は深層学習の代表的な課題であるセマンティックセグメンテーションを実行したいです。U-Net++を使って、胸部X線画像のセグメンテーションを行います。画像データと手動で作ったアノテーションデータが保存されています。画像データはBMP型式、アノテーションデータ（ラベルデータ）はPNGです。学習用データは、画像データは、
content/drive/MyDrive/dentalai09/Unet/org_train_s アノテーションデータは、
content/drive/MyDrive/dentalai09/Unet/label_train_s 評価用データは、画像データは、
/content/drive/MyDrive/dentalai09/Unet/org_test アノテーションデータは、
content/drive/MyDrive/dentalai09/Unet/label_test に保存されています。学習したU-Net++で評価用のデータを処理した結果は、/content/drive/MyDrive/dentalai09/Unet/に新しいディレクトリを作って保存してください。ただし、この推定結果は、評価用データのみならず、学習用データも再代入法で作成してください。学習用のデータはUnetの下にTrainResults,評価用のデータはUnetの下にTestResults,として保存してください。学習の状態がわかるように、学習曲線も作ってください。Lossと分類精度の2つの曲線をお願いします。これは学習中にエポック数と、Lossおよび精度のデータフレームを作って、学習が終了したら、そのデータフレームからグラフを作ってください。なお、Google ColabのGPUでT4を使う予定です。PyTorchを使ってお願いします。



これで深層学習に関する課題は終了です.