令和 4 年度
プロジェクトデザイン III
プロジェクトレポート
情報工学科

電車の分類をしたよ

提出日　令和 99 年 99 月 99 日

指導教員：鷹合 大輔 准教授

氏名　学籍番号 (クラス–名列番号)
野崎 悠度　1936463 (4EP1–68)
奥野 細道　1936119 (4EP4–75)

概要

本プロジェクトでは，機械学習を用いた電車の車両タイプを分類するシステムの開発を行った．

世の中には似ているようなものでも，実は同じではないものがある．電車の車両タイプの種類は,JR の在来線だけでも１００種類近く存在している．多くの人は電車を見て，電車だと認識することは可能であるが，その電車の車両タイプまでを判断できる人は少ない．電車についての知識がある人は一目見るだけでその電車の車両タイプを判断できるが，大多数の人は似ている電車の車両タイプを判断することが難しい．そのため，だれでも簡単に電車の車両タイプを判断できるシステムの開発を行った．今回は JR 西日本の１７種類の電車を分類，識別するシステムの開発を行った．

モデルの開発は，画像を集めてデータセットを作成し，YOLO とデータセット用いて学習させ評価を行う．学習データは車両タイプ別に YouTube にアップロードされている動画から電車が写っている場面を切り出した画像を利用した．一つの動画から一種類の車両タイプのデータセットを作成すると，似たような画像が大量に保存されてしまうため，複数の動画の任意の場面を連結して一本の動画にするシステムを Python のフレームワークである Django を利用して作成し，様々な場面の電車の画像が保存できるようにした．学習を進めていき性能が向上しなくなるまで学習をさせてモデルを作成した．外見が似ている車両タイプの判別の間違いが多かった．判別対象の車両タイプのなかで外見に特徴がある車両タイプは正確に判別をすることに成功した．

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝ここから田村＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

作成したモデルを使ってブラウザ上で電車の車両タイプを分類する WEB アプリも作成した．

活動履歴 野崎 悠渡

期間	活動内容	活動時間 [h]
4–8 月	調査・実験・実装	200
11–3 月	実験・実装・論文執筆（3 章）	160

活動履歴 鬼 太郎

期間	活動内容	活動時間 [h]
4–8 月	調査・実験・実装	200
11–3 月	実験・実装・論文執筆（3 章）	160

目次

図目次

表目次

ソースリスト目次

第 1 章

序論

電車の車両タイプは JR の在来線だけでも 100 種類近く存在している．多くの人は電車を見て電車だと認識することは可能だが, その電車の車両タイプまでを判断できる人は少ない．電車についての知識がある人は一目見るだけでその電車の車両タイプを判断できるが, 大多数の人は似ている電車の車両タイプを判断することが難しい．現在，車両タイプを調べる方法として Google レンズを使用することや，図鑑と見比べる方法が挙げられる．Google レンズは一枚の画像に 2 つ以上の電車が写っていると車両タイプを正確に分類できず，分類結果は車両タイプが出力されるのではなく，入力画像に写っているモノと似ているものが写っているウェブページ一覧を出力するので，出力されたウェブサイトを開いて知りたい情報を自分で探し出す必要がある．図鑑と見比べる方法では知りたい情報を得るために多くの時間が必要になる．

現在の方法では画像に写っている車両タイプが何なのかが知りたいときに余計な手間をかける必要があるという問題がある．本プロジェクトではこの問題を解決するために画像や動画に写っている車両が何なのかを判別できるシステムを開発する．

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝ここから田村＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

第 2 章

システム概要

2.1　この章で書くこと

- 識別，分類とは
- 現存するサービスでできること
- 概要図
- モデルについて
- サーバ関連について

2.2　分類と識別とは

分類はデータやオブジェクトを異なるクラスやカテゴリに分けるプロセスを指す．画像の分類とは，画像が特定のカテゴリやクラスに属するかどうかを判別する作業である．例えば，画像に写っているのが猫か犬かのクラスに分けることである．

識別とは画像のどこに何が写っているのかを判別するプロセスを指す．1 枚の画像に複数の物体が存在する場合も識別はできる．動画に写っている電車の車両タイプも判別することができる．

2.3　現存するサービス

現在車両タイプを調べるためには，Google レンズを利用することや図鑑と見比べることの 2 つが挙げられる．Google レンズとは画像の分類を行うアプリである．単に分類結果が表示されるのではなく，分類したいオブジェクトが映っているウェブサイト一覧が表示されるものである．表示されたウェブサイトを適当に選び自分が知りたい結果をウェブサイトの中から探し出す必要がある．また，一枚の画像に複数のオブジェクトが存在している場合は正しい結果が得られない．

2.4　YOLO

YOLO とは You Only Look Once の略で，人間のように一目見るだけで物体検出ができることを指している [1]．データセットを作成し学習させることで，任意の物体のみ検出させることが可能である．YOLOv8 は YOLO シリーズの最新バージョンであり，ディープラーニングとコンピュータビジョンの最先端の進歩に基づいており，速度と精度の麺で比類のない性能を提供している．本プロジェクトでは，YOLOv8 を用いて電車の車両タイプを識別，分類する 2 つのモデルを開発する．

2.5　システム概要図

本プロジェクトで開発するシステム概要を図 2.1 に示す．システムは車両判別部と UI に分けられる．

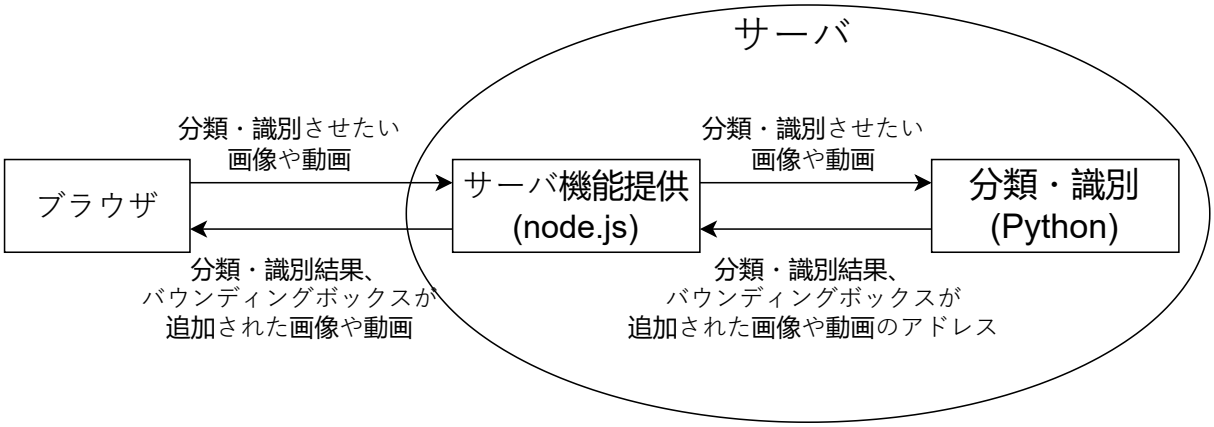
サーバ
分類・識別させたい
画像や動画
ブラウザ
サーバ機能提供
(node.js)
分類・識別させたい
画像や動画
分類・識別
(Python)
分類・識別結果、
バウンディングボックスが
追加された画像や動画
分類・識別結果、
バウンディングボックスが
追加された画像や動画のアドレス

図 2.1: システム概要図

第 3 章

学習データの準備

3.1　データの収集

学習データは，車両タイプごとにその車両タイプが映っている YouTube の動画をダウンロードして，任意の枚数分のランダムなフレームを保存する．その後，保存した画像を識別して電車が映っている画像を保存してレーニングデータとバリデーションデータを収集した．テストデータは様々なウェブサイトから手作業で 17 種類の各車両の画像を 10 枚ずつ収集した．本プロジェクトでは，JR 西日本の在来線の電車を識別または分類する．
205 207 213 221 223 225 227 271 281 283 285 287 321 323 381 521 683
上記の１７種類の画像を集める．

3.1.1　動画の保存方法

動画の保存には yt-dlp という動画や音声をダウンロードするプログラムを使う．１～３種類の動画の任意の秒数をダウンロードしてそれぞれの動画を連結して一つの動画にする web アプリケーションを Python のフレームワークの一つである Django を使って作成した．
作成した web アプリケーションは，図 3.1 図 3.2 図 3.3 のようになっている．YouTube 上の動画の URL と開始時刻 (秒), 終了時刻 (秒), 車両タイプ名を入力して，1～3 種類の動画を保存し連結して一つの動画にしている．

図 3.1: 動画のダウンロード 1

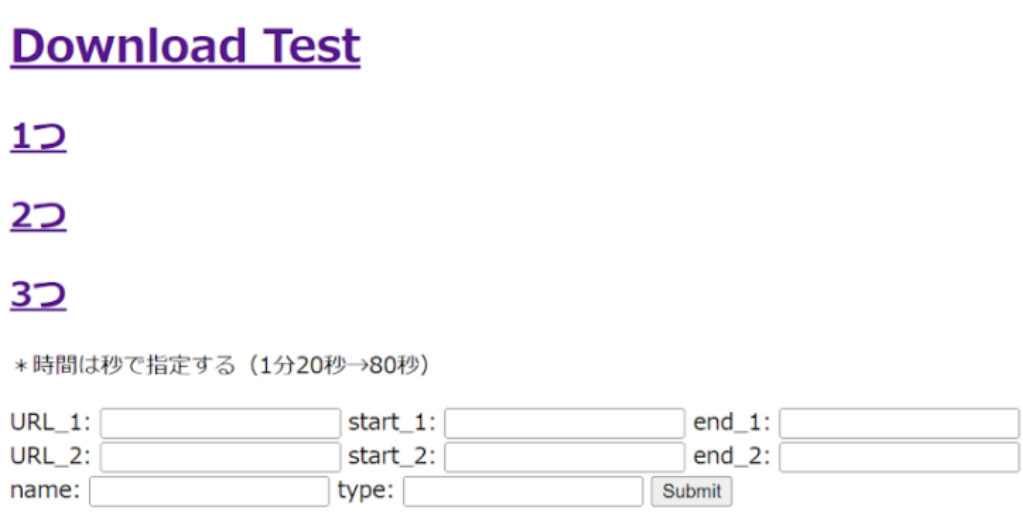

図 3.2: 動画のダウンロード 2

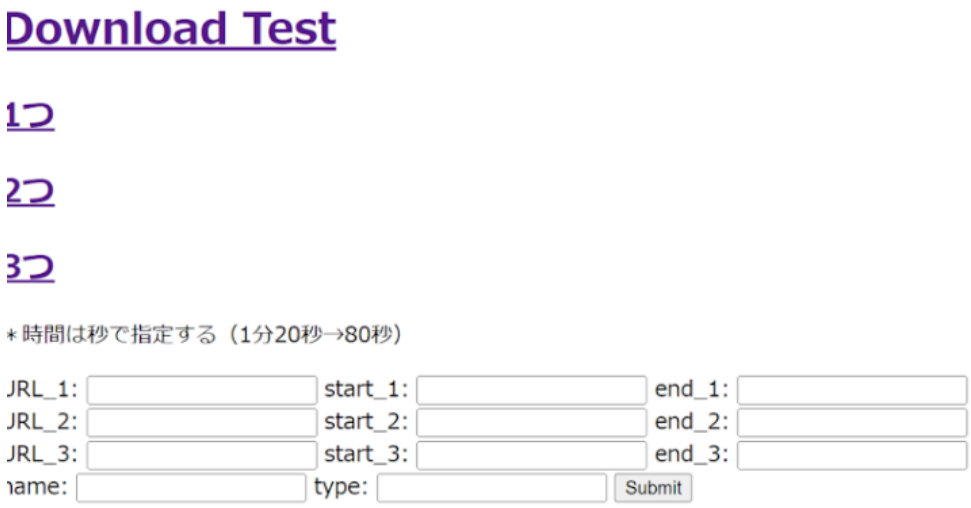

図 3.3: 動画のダウンロード 3

3.1.2　画像の保存方法

保存した動画を利用して指定した枚数分のランダムなフレームを保存する．保存した画像を識別し電車が映っている画像だけを保存す，トレーニングデータとバリデーションデータを収集した．テストデータは様々なウェブサイトから手作業で 17 種類の各車両の画像を 10 枚ずつ収集した．車両タイプごとに保存した画像の枚数を図 3.4 に示す．

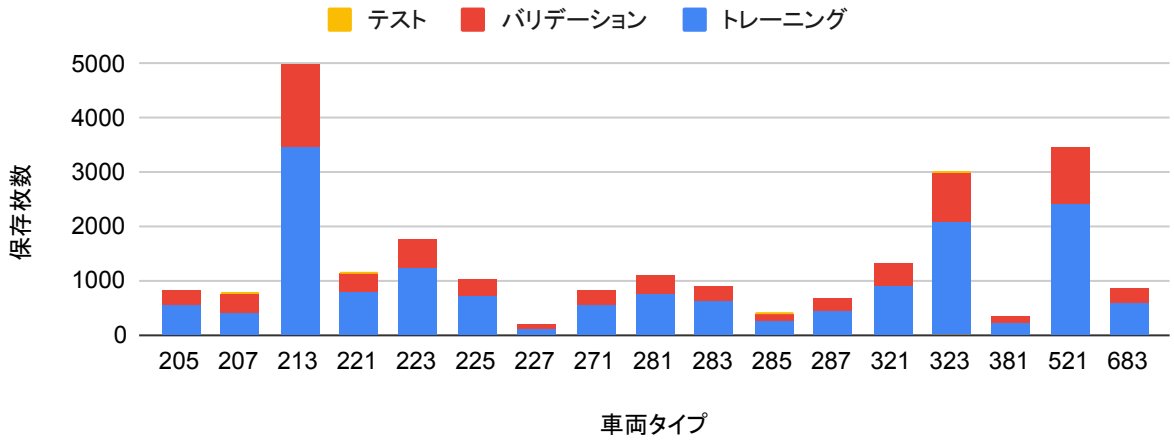

図 3.4

3.2　データセットの作成

識別モデル用データセットと分類モデル用データセットの学習時に使用する画像は同じものを使う．識別モデル用のデータセットには画像のアノテーション情報が必要なので，アノテーションも行った．このテストデータセットと作成したモデルを使って分類または識別を行い，モデルの性能の評価を行う．アノテーションとは、機械学習の分類の一つである教師あり学習において, 分析対象データにラベルを付与するプロセスである. 画像にバウンディングボックスと呼ばれる四角形を描画しクラス番号を指定する．バウンディングボックスを描画することでその画像に写っている物体の座標情報を取得することができる．クラス番号とは，判別したいものリストを作成し，画像に写っている物体に対応した，リストのインデックスのことである．アノテーションをした結果は，識別モデルの学習時に使用する．

一般的にアノテーションは，手作業で行うものである．しかし，数千枚の画像を手作業で行うことは難しいので，自動で行えるようにした．分類モデル用のデータセットでは一枚の画像に電車が一つだけ映っている．その画像を配布されている識別用モデルで識別することでクラス番号と電車が映っている座標情報をテキストに書き込む．その後，本プロジェクトで識別する車両タイプリストに対応するクラス番号を上書きすることで，分類モデル用のデータセットから識別モデル用のデータセットを作成した．

第 4 章

車両タイプ判別モデルについて

4.1　学習の実行

作成したデータセットと YOLOv8 を用いてモデルの学習を行い，分類用と識別用の 2 種類のモデルを作成した．分類モデルでは静止画での判別しかできない．動画から車両タイプを判別するために識別モデルを作成した．

4.2　作成したモデルの使い方

識別モデルと分類モデルで使い方はほぼ同じである．作成したモデルをロードして，モデルに画像または動画を渡すと識別または分類をすることができる．

識別

```
from ultralytics import YOLO
model = YOLO("作成した識別モデルのパス")
results = model.predict("画像または動画のパス")
```

分類

```
from ultralytics import YOLO
model = YOLO("作成した分類モデルのパス")
results = model("画像のパス")
```

4.3　出力されるもの

１７種類の車両が各１０枚ずつ，合計１７０枚のテストデータセットを識別した際にターミナルに出力されるものを図 4.1 に示す．左端の image から，「何枚目か」「画像のパス」「入力画像のサイズ」「予想クラス番号」「予想クラス番号に対応する車両タイプ」「識別にかかった時間」が画像ごとに出力される．識別時に save = True を追加すると入力画像に識別結果が追記された画像が出力される．識別時に save_txt = True を追加すると画像ごとに予想クラスと座標情報がテキストファイルで出力される．モデルを動かした際に端末に出力されるものを図 4.1 に記す．

```
image 1/170 /home/nozaki/yt-dlp/test/detect_test/205_1.jpg: 384x640 1 205, 44.8ms
image 2/170 /home/nozaki/yt-dlp/test/detect_test/205_10.jpg: 448x640 1 205, 43.2ms
image 3/170 /home/nozaki/yt-dlp/test/detect_test/205_2.jpg: 448x640 1 205, 3.3ms
```

図 4.1: 端末に出力されるもの

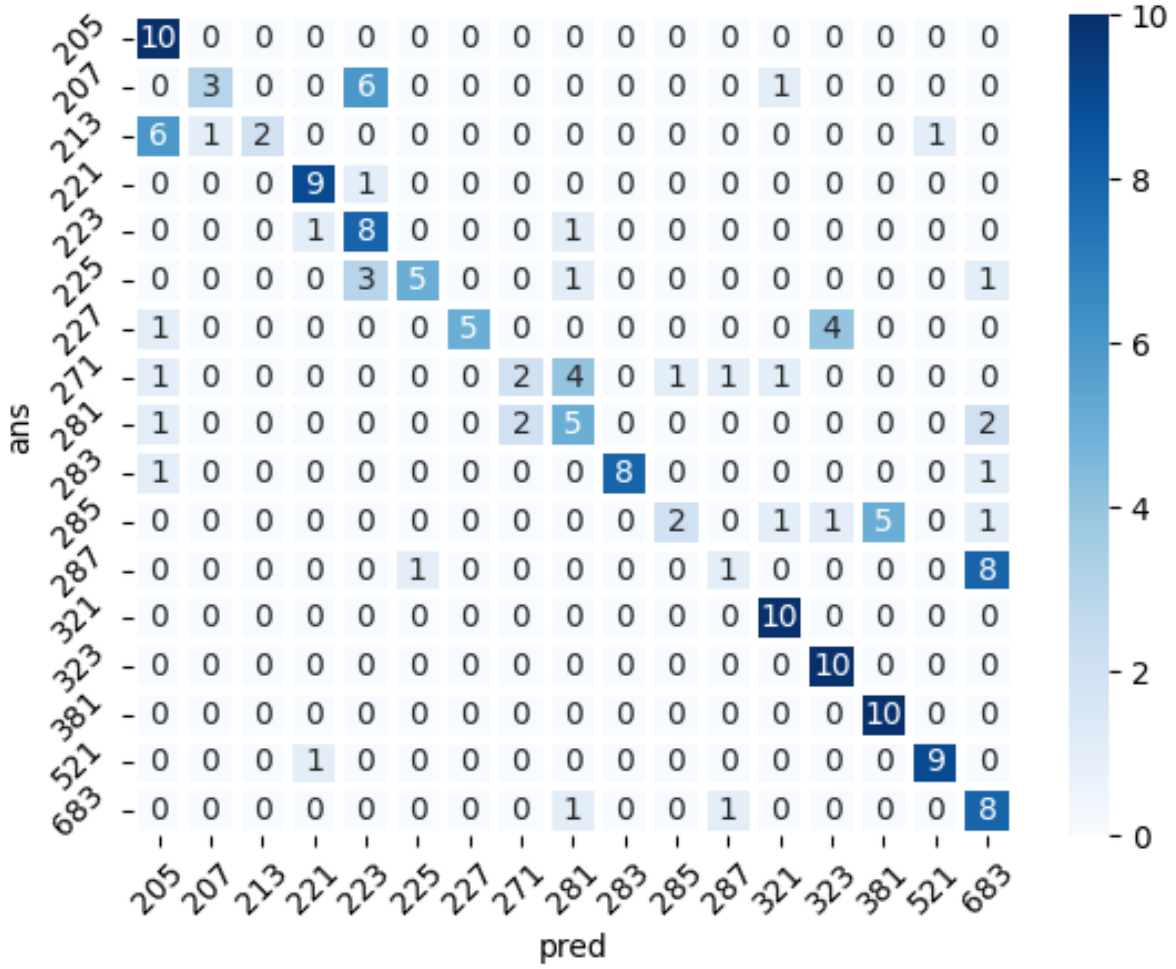

図 4.2: 分類モデルの混合行列

4.4　作成したモデルの評価と考察

4.4.1　分類モデルの評価

作成した分類モデルを用いてテストデータセットの分類を行った結果をを図 4.2 に示す．縦軸が予測した車両タイプ，横軸が正解の車両タイプとするグラフである．

4.4.2　識別モデルの評価

識別時には，どの車両タイプにも当てはまらないと識別されることがある．その場合の車両タイプは 0 として識別モデルの混合行列を作成した．識別結果を表 4.3 に示す．

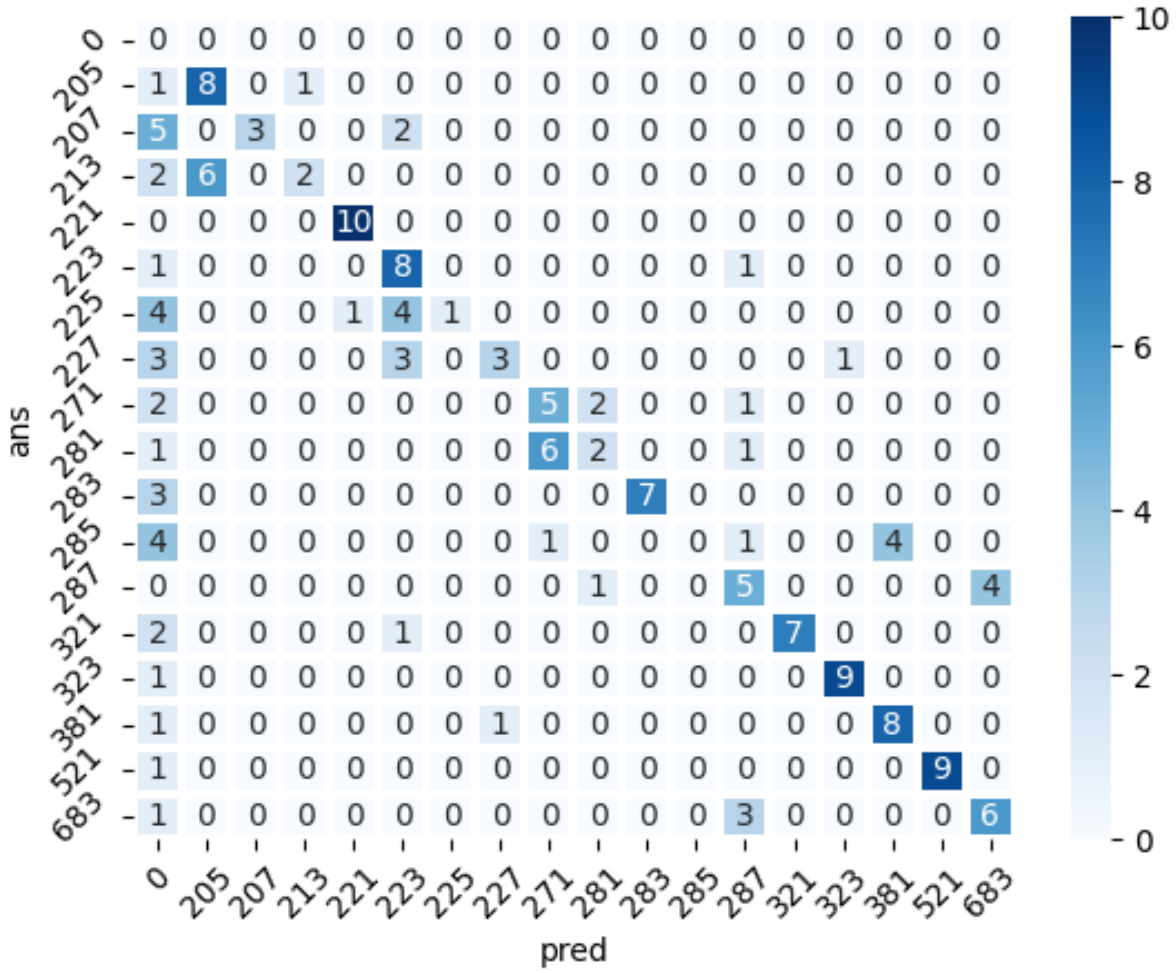

図 4.3: 識別モデルの混合行列

4.4.3　考察

正解率は車両タイプによって異なることがわかる．287 系と 683 系のように外見が似ている電車だと，誤判別していることが多かった．287 系を図 4.4，683 系を図 4.5 に示す．データセットの画像の枚数が少ない車両タイプの判別結果が必ず悪くはならなかった．画像の枚数が判別結果に影響を与えるのではなく，車体の特徴が鮮明に写っている画像の枚数が判別結果に影響を与えると考えられる．判別精度の向上のために，データセットとして質の悪い画像を大量に集めるのではなく，車体の特徴が鮮明に写っている画像を車両タイプごとに集める必要があったと考えられる．

同じ画像を使って，分類用と識別用の 2 種類のデータセットを作成した．分類モデルの混合行列と識別モデルの混合行列が似ているため，車両タイプを知るためのシステムを作る際には，識別モデルのみを作成することで画像と動画の両方に対応した車両判別システムを作成でき，電車の車両タイプを知ることができるようになると考える．

図 4.4: 287 系

図 4.5: 683 系

第 5 章

結論

本プロジェクトでは，機械学習を用いた電車の車両タイプを判別するシステムの開発を行った．また，電車が写っている画像や動画をサーバ上で，分類，識別を行い判別結果を出力する Web アプリを作成した．

YouTube の動画から電車の画像を保存して，トレーニング用とバリデーション用の画像を収集しデータセットを作成した．そのデータセットと YOLO を用いて，車両タイプの分類モデルと識別モデルの 2 種類のモデルの作成を行った．車両タイプを知るためには分類モデルを作成すればよいと考えていたが，分類モデルでは動画に写る電車の車両タイプの判別ができないため，識別モデルを作成した．

本システムを利用することで電車の知識がない人でも画像または動画に写る一部の車両タイプが何なのかを知ることができるようになった．

==========ここから田村＝＝＝＝＝＝＝＝＝＝

参考文献

[1] ”Ultralytics YOLOv8 ドキュメント”,https://docs.ultralytics.com/ja

付録 A

開発したプログラム

A.1　使い方

リスト A.1 には保存した動画から電車の画像を保存するための関数が定義されている．リスト A.2 ではリスト A.1 の関数を利用して画像を保存する．リスト A.2 を実行する際には引数に保存する画像の枚数と車両タイプを指定してから実行する．

A.2　ソースコード

リスト A.1: project_processing.py

```
#from email.mime import image
import re
#from symbol import typelist
from ultralytics import YOLO
import cv2
from PIL import Image
import random
import os
import yt_dlp
import matplotlib.pyplot as plt
import numpy as np
import glob
import torch
import shutil
from pytube import YouTube

class project_processing:
    def get_random_frame(self, name, video_path, out_path, img_sum):
        cap = cv2.VideoCapture(video_path)
        frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        #print(frame_count)
        for count in range(img_sum):
            new_img_name = f"{name}_{str(count).zfill(4)}.jpg"
            new_img_path = out_path + new_img_name
            #出力フォルダが存在しない場合は作成する
            os.makedirs(out_path, exist_ok=True)
            random_frame_number = random.randint(0, frame_count - 1)
            cap.set(cv2.CAP_PROP_POS_FRAMES, random_frame_number)
            ret, frame = cap.read()
            if ret:
                cv2.imwrite(new_img_path, frame)
                print(f"Saved random frame {random_frame_number} as {new_img_name}")
        print(f"saved at {out_path}")

    def cropping(self, train_type, input_path, output_path):
        #model = YOLO("yolov8n.pt")
        #yolov8n.ptで推論しようとすると、obj = result.pandas~の部分でエラーになるよ
        model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained = True)
        image_files = glob.glob(os.path.join(input_path, '*.jpg'))
        # 出力フォルダが存在しない場合は作成する
        os.makedirs(output_path, exist_ok=True)
        #通し番号
        count = 0
        for image_file in image_files:
            # 画像の読み込み
```

```
            image = cv2.imread(image_file)
            # 物体検出の実行
            result = model(image)
            # 物体検出結果をPandasのDataFrame形式で取得する
            obj = result.pandas().xyxy[0]
            # 検出された電車のバウンディングボックスの中身を別のフォルダに保存する
            for j in range(len(obj)):
                # バウンディングボックスの情報を取得
                name = obj.name[j]
                score = obj.confidence[j]
                #画像に複数の電車が映っている場合はその画像の処理はスキップする
                if name.count("train") > 2:
                    continue
                if name == "train" and score >=0.90:
                    #保存先のパスを生成
                    filename = f"{train_type}_{str(count).zfill(3)}"
                    img_filename = filename + ".jpg"
                    save_path = os.path.join(output_path, img_filename)
                    cv2.imwrite(save_path, image)
                    print(f"saved {img_filename}")
                    count += 1
        #保存した画像を消す（物体認識をする前の画像を消す）
        # ディレクトリ内のファイルを取得
        file_list = os.listdir(input_path)
        for file_name in file_list:
            file_path = os.path.join(input_path, file_name)
            # ファイルが存在し、拡張子が.jpgの場合にのみ削除
            if os.path.isfile(file_path) and file_name.endswith(".jpg"):
                os.remove(file_path)
                print(f"{file_name} を削除しました")
        #物体認識をする前の画像フォルダを削除する
        #os.rmdir(input_path)
        #print(f"{input_path} を削除しました")
```

リスト A.2: save.py

```
#/home/nozaki/yt-dlp/pythonにある
#python save.py --img_sum 000 --type 000

import project_processing as pp
import argparse

# コマンドライン引数のパーサーを作成
parser = argparse.ArgumentParser(description="Process images and videos.")
parser.add_argument("--img_sum", type=int, help="Number of images to process")
parser.add_argument("--type", type=str, help="Type of processing")

def main():
    # コマンドライン引数をパース
    args = parser.parse_args()
    # 引数を取得
    img_sum = args.img_sum
    type = args.type
    pp_instance_den = pp.project_processing()

    #これから以下のフォルダに画像を保存する
    base_path = "/home/nozaki/yt-dlp/"
    video_path = base_path + f"concat/{type}_con.webm"
    #ランダム画像の保存先
    out_path = base_path + f"img/{type}/before/"
    pp_instance_den.get_random_frame(type, video_path, out_path, img_sum)
    #クロップ用の変数
    input_path = out_path
    output_path = base_path + f"img/{type}/"
    pp_instance_den.cropping(type, input_path, output_path)
    print(f"{output_path} に保存")

if __name__ == "__main__":
    main()
```

付録 B

いいいいい

　あああああああああああああああああああああいいいいいいいいいいいいいいいいいいいいいいいいいいいいうううううううううううううううううう