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・識別とは

文の量が少ない？ 加えたほうが良いのか
分類とは，画像に写っているものが何かを判断することである．
(例) 犬，猫，電車，人
識別とは，画像に写っているものが何か，どこに写っているのかを判断することである．
(例) 家族の集合写真で，自分がどこにいるのかを特定する．

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．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．
田村と相談，画像は後で変える

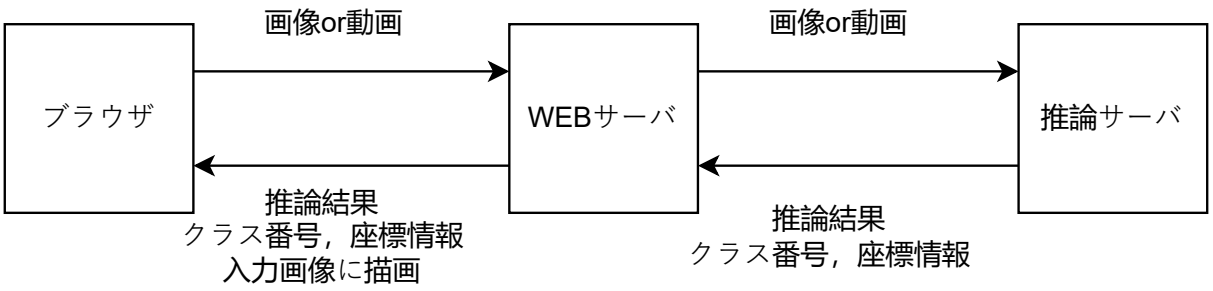
画像or動画
画像or動画
ブラウザ
WEBサーバ
推論サーバ
推論結果
クラス番号，座標情報
入力画像に描画
推論結果
クラス番号，座標情報

図 2.1: システム概要図

第 3 章

学習データの準備

3.1　この章で書くこと

- django の件
- 電車が映っている場面だけ画像で保存
- データセットを作成（識別・分類）
- アノテーションについて

3.2　画像の収集

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，任意の枚数だけランダムで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場面のみ画像で保存する．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．

3.2.1　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．

3.2.2　画像の保存方法

画像の保存は二回行う．一回目は，保存した動画でランダムなフレームを保存する．二回目は，保存した画像を配布されているモデル (yolov8n.pt) で識別をして，電車が一つだけ映っているものだけを保存する．
↓は消しても良いのか
→電車の画像を保存する際には，車両タイプ（数字３桁）というディレクトリに
車両タイプ（数字３桁）_通し番号（数字４桁）.jpg という名前で保存する．
車両タイプごとに保存した画像の枚数を図 3.4 に示す．

3.3　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．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.

この章は他に書くことがあるのか

図 3.1: 動画のダウンロード 1

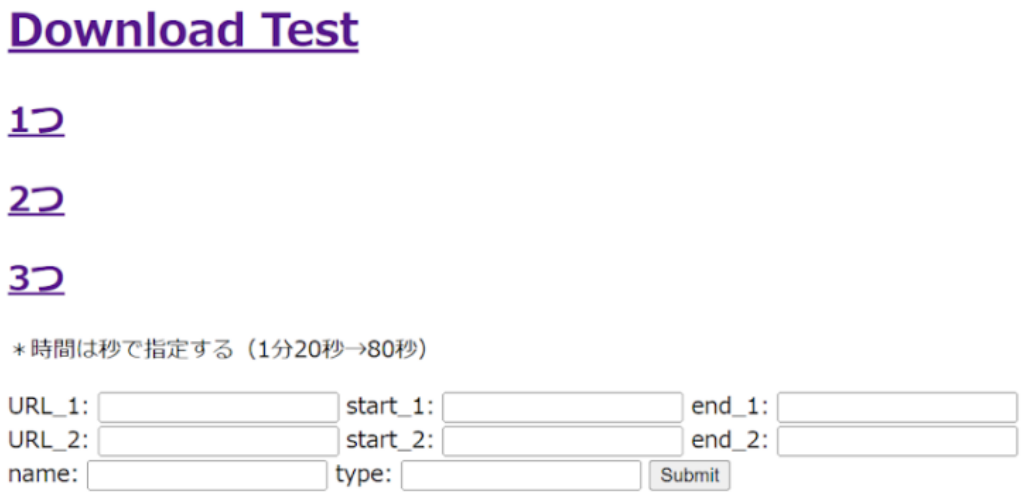

図 3.2: 動画のダウンロード 2

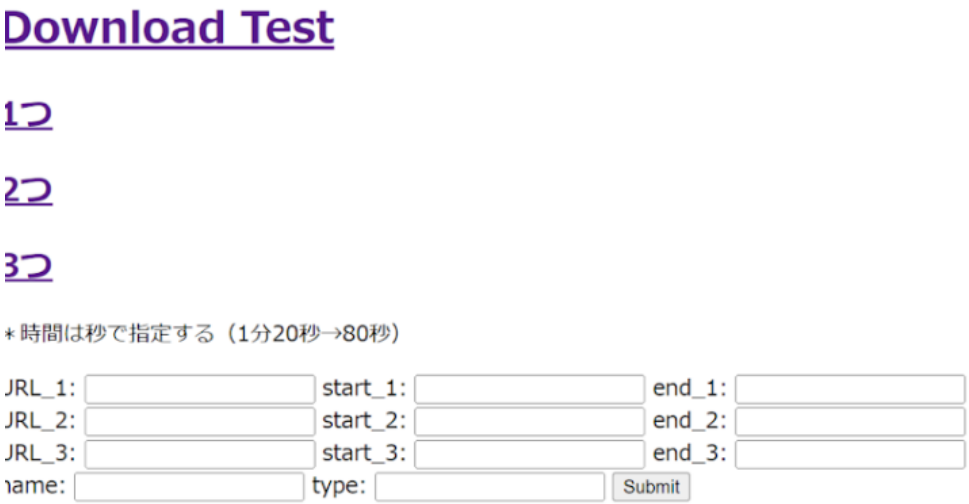

図 3.3: 動画のダウンロード 3

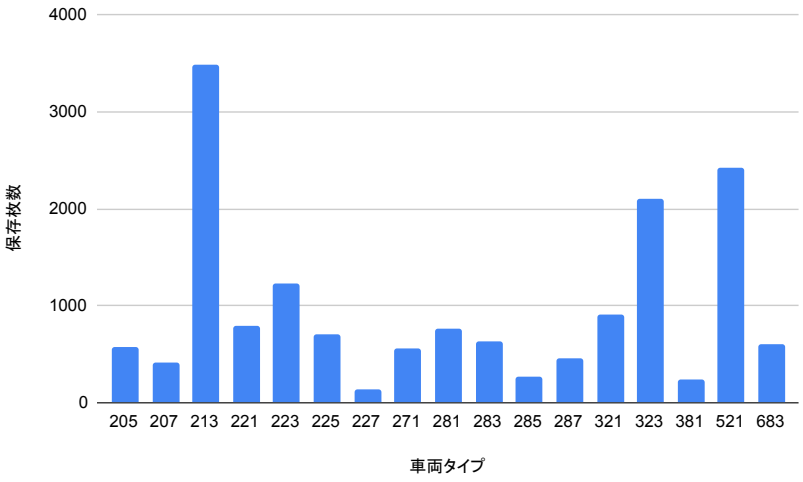

図 3.4: 車両タイプ別画像の枚数

第 4 章

車両タイプ判別モデルについて

4.1　この章で書くこと

- 学習の実行
- 性能評価
- 作成したモデルの使い方
- 出力されるものについて

4.2　学習の実行

学習を進めると性能が徐々に向上していくが学習を繰り返しても性能が向上しなくなると学習が強制終了する．エポック数という学習用データを何回繰り返して学習させるのかを表す数を１０万回程度に設定して学習を進めた．強制終了した際のモデルが使用したデータセットで作れる最高の性能のモデルとなる．

4.3　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")
```

4.4　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 sabe = 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

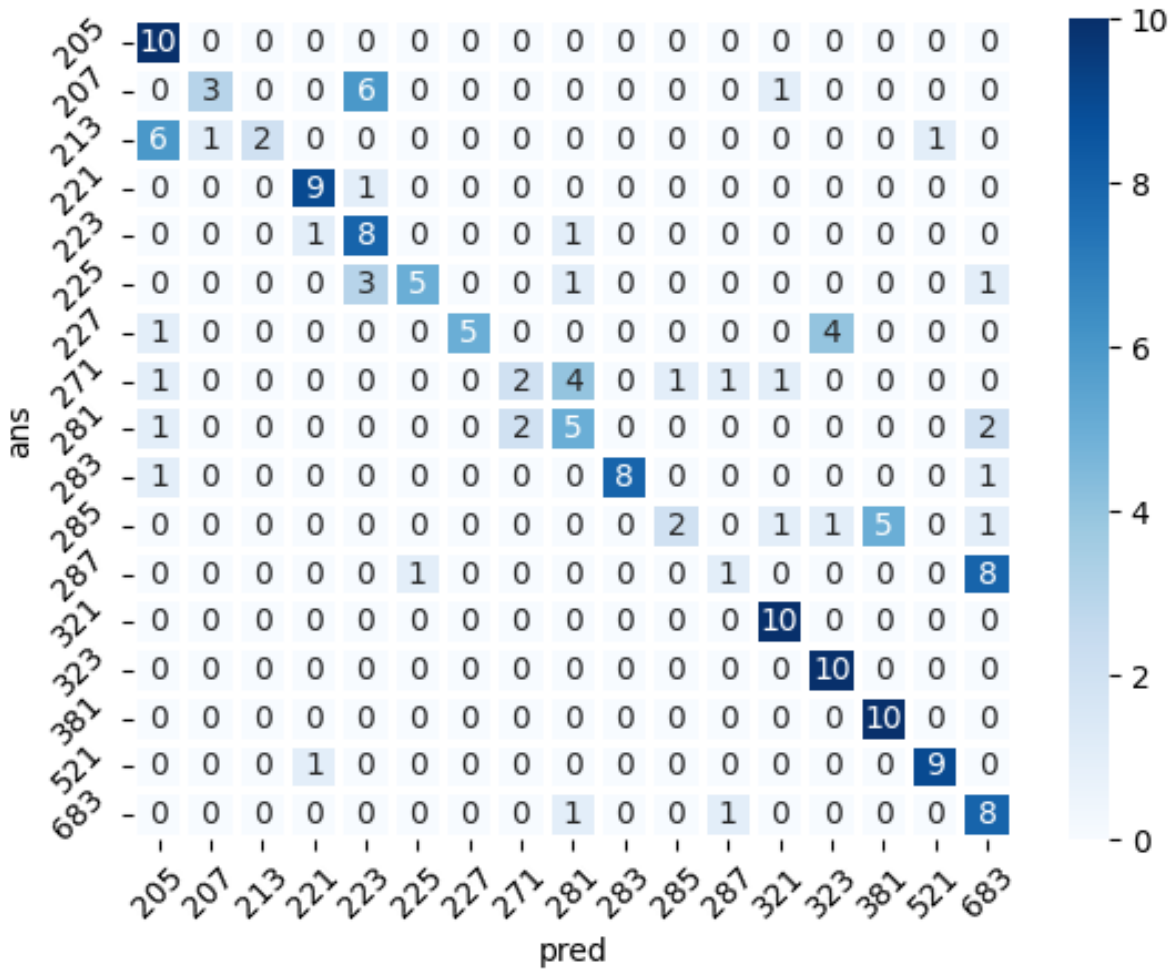

図 4.2: 分類モデルの評価

4.5　性能の評価

１７種類の各車両の画像を１０枚ずつ集めて，テストデータセット作成した．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．

4.5.1　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．

ここに分類結果からわかることを書けば良い？

図 4.2 から，207 系と 223 系，223 系と 225 系，227 系と 323 系，285 系と 381 系，287 系と 683 系の誤分類が多いことがわかる．223 系，225 系は帯の色が，白・青・黄土色の 3 色で構成されている．227 系は赤色，323 系はオレンジ色, 285 系と 381 系は旧国鉄カラー（クリーム色，薄い黄色）, 外見が似ている車両タイプは

誤分類が多い．正しく分類できているのは，外見が他の車両と似ていないものばかりだった．

4.5.2　識別モデルの評価

識別失敗数とは間違ったクラスだと判断した数であり，識別不能数とはどのクラスにも当てはまらないと判断した数とする．識別結果を表 4.3 に示す．

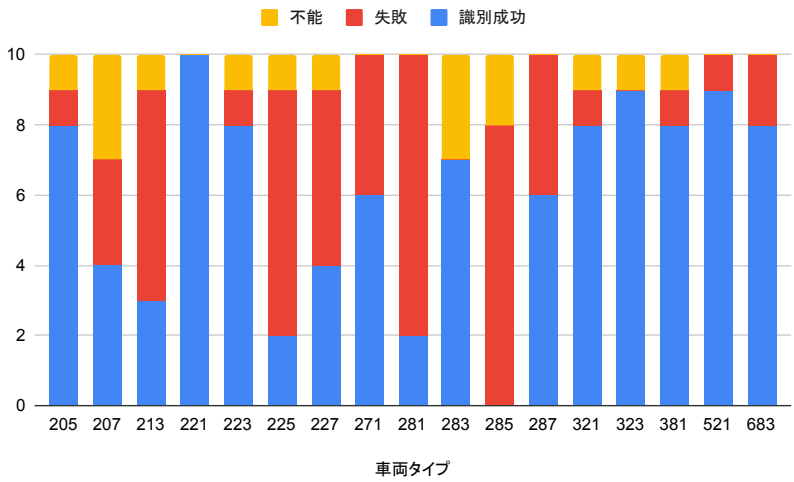

図 4.3: 識別結果

テストデータセットの各画像を識別して７枚以上識別に成功した車両タイプは，２０５系，２２１系，２２３系，２８３系，３２１系，３２３系，３８１系，５２１系，６８３系の９種類だった．５枚以上識別に失敗している車両タイプは２１３系，２２５系，２２７系，２８１系，２８５系の５種類だった．

4.5.3　考察

正解率は電車によって異なることがわかる. 特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. 各車両タイプの画像の枚数に差があったことが判別結果に影響を与えていると考えられる．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．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．

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，「たむらたむらたむら」の開発を行った．

YouTube の動画から電車の画像を保存するという方法を用いてデータセットの作成を試みた．データセットに含まれる画像の枚数が車両タイプごとに大きな差があったが，画像の枚数が少ない車両タイプの判別結果が悪く，枚数の多い車両タイプの判別結果が良いというわけではなかったため，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大量に集める必要があったと考えられる．判別する車両タイプをある路線を走る 5 種類程度に絞ることでデータ収集とデータの質を精査することをより効率的に行うことができると考えられる．学習用データの収集を簡略化しようと工夫を重ねたがデータセットの質を確保できなかったため，特徴的な車両の判別は成功して似ている車両の判別にミスが多く見られていると考える．

==========ここから田村＝＝＝＝＝＝＝＝＝＝

参考文献

[1] V.D. Vaughen and T.S. Wilkinson, ”System considerations for multispectral image compression designs,” IEEE Signal Process. Mag., vol.12, no.1, pp. 19-31, Jan. 1995.

[2] A. Said and W. Pearlman, ”An image multiresolution representation for lossless and lossy compression,” IEEE Trans. Image Process., vol.5, no.9, pp.1303-1310, Sept. 1996.

[3] 小野文孝, ” 静止画符号化の新国際標準方式（JPEG2000）の概要,” 映像情報メディア学会誌, vol.54, no.2, pp.164-171, Feb. 2000.

[4] D. Tretter and C.A. Bouman, ”Optimum transform for multispectral and multilayer image coding,” IEEE Trans. Image Process., vol.4, no.3, pp.296-308, March 1995.

[5] F. Amato, C. Galdi and G. Poggi, ”Embeded zerotree wavelet coding on multispectral images,” IEEE Proc. ICIP 97, vol.1, pp.612-615, 1997.

[6] B.R. Epstein, R. Hingorani, J.M. Shapiro and M. Czigler, ”Multispectral KLT-wavelet data compression for Landsat thematic mapper images,” Proc. Data Compression Conf., IEEE Computer Society Press, pp.200-205, 1992.

[7] J.M. Shapiro, S.A. Martucci, and M. Czigler, ”Comparison of multispectral Landsat imagery using the embeded zerotree wavelet(EZW) algorithm,” DLPO at Image Compress. Appli. & Innovation Workshop, pp.105-113, March 1994.

[8] J.A. Saghri, A.G. Tescher, and J.T. Reagan, “Practical transform coding of multspectral imagery,” IEEE Signal Process. Mag., vol.12, no.1, pp.32-43, Jan. 1995.

[9] J. Lee, “Optimized quadtree for Karhunen-Loeve transform in multispectral image coding,” IEEE Trans. Image Process., vol.8, no.4, pp.453-461, April 1999.

[10] B. Brower, B. Gandhi, D. Couwenhouven and C. Smith, “ADPCM for advanced LANDSAT downlink applications,” in Proc. 27th Asilomar Conf. Signals, Systems and Computers, Nov. 1993.

[11] N.D. Memom, K. Sayood and S.S. Magliveras. ,”Lossless compression of multispectral image data,” IEEE Trans. Geosci. and Remote Sensing., vol.32, no.2 , pp.282-289, March 1994.

[12] S. Gupta and A. Gersho, “Feature predictive vector quantization of multispectral images,” IEEE Trans. Geosci. and Remote Sensing., vol.30, no.3, pp.491-501, May 1992.

[13] K. Irie and R. Kishimoto, “A study on perfect reconstructive subband coding,” IEEE Trans. Circuits Syst. Video Technol., vol.1, no.1 , pp.42-48, March 1991.

[14] 小松 邦紀, 瀬崎 薫, 安田 靖彦, “濃淡画像の可逆的なサブバンド符号化法”, 信学論（D-II）, vol.J78-D-II, no.3, pp.429-436, March 1995.

[15] A.R. Calderbank, I. Daubechies, W. Sweldens and B. Yeo, ”Lossless image compression using integer to integer wavelet transforms,” IEEE Proc. ICIP 97, vol.1, pp.596-599, 1997.

[16] F.A.M.L. Bruekers and A.W.M.V.D. Enden, “New networks for perfect inversion and perfect reconstruction,” IEEE J. Sel. Areas Commun., vol.10, no.1, pp.130-137, Jan. 1992.

[17] 小松 邦紀, 瀬崎 薫, “可逆的離散コサイン変換とその画像情報圧縮への応用,” 信学技報, vol.IE97-83, pp.1-6, Nov. 1997.

[18] K. Komatsu and K. Sezaki, ”Design for lossless block transforms and filter banks for image coding,” IEICE Trans. Fundamentals, vol.E82-A, no.8, pp.1656-1664, Aug. 2000.

[19] 福間 慎治, 岩橋 雅宏, 神林 紀嘉, “可逆的色変換を用いた色彩画像の可逆符号化,” 信学論（D-II）,

vol.J81-D-II, no.11, pp.2680-2684, Nov. 1998.

[20] T. Nakachi, T. Fujii, and J. Suzuki,"A unified coding algorithm of lossless and near-lossless color image compression," IEICE Trans. Fundamentals, vol.E83-A, no.2, pp. 301-310, Feb. 2000.

[21] 仲地 孝之, 藤井 竜也, "可逆 KL 変換を用いた病理顕微鏡画像符号化法の研究," 2000 信学総大, D16-13, March 2000.

[22] 鷹合 大輔, 武部 幹, "可逆 WT・KLT を用いるマルチスペクトル画像の情報圧縮," 信学論（A）, vol.J84-A, no.3, pp.1-11, March 2001.

[23] 尾上 守夫, "画像処理ハンドブック," 昭晃堂, pp.554-561, 1987.

[24] 鷹合 大輔, 武部 幹, "マルチスペクトル画像のバンド間・バンド内相関除去による可逆情報圧縮," 第 15 回 DSP シンポジウム講演論文集, pp.475-48, Nov. 2000.

[25] 鷹合 大輔, 武部 幹, "ウェーブレット変換を用いたマルチスペクトル画像の情報圧縮符号化法の研究," 平成 10 年度金沢工大卒業論文, 1998.

[26] 酒井 幸市, "デジタル画像処理入門," コロナ社,1997.

[27] 榊原 進, "ウェーブレットビギナーズガイド," 電機大出版局, 1995.

[28] 武部 幹, "情報圧縮、通信と回路理論," 信学技報, IT97-40, July 1997.

[29] Gilbert Strang and Troung Nguyen, " Wavelet and filter banks,' 'Wellesly Cambridge Press, 1996.

[30] 貴家 仁志, "よくわかるディジタル画像処理," CQ 出版社, 1996.

[31] " 金沢の暮らし", http://www.kanazawa-it.ac.jp

[32] 山田 太郎, " 金沢の一人暮らし", トンチンカン出版, 2016.

付録 A

開発したプログラム

A.1　セットアップ方法

にプログラムの使い方，セットアップ方法などを書きましょう．ここにプログラここにプログラムの使い方，セットアップ方法などを書きましょう．ムの使い方，セットアップ方法などを書きましょう．ここにプログラムの使い方，セットアップ方法などを書きましょう．

```
$ sudo apt-get install python3-pip      # PIP コマンドの導入
$ echo "Hello WOrld"
Hello WOrld
```

にプログラムの使い方，セットアップ方法などを書きましょう．ここにプログラここにプログラムの使い方，セットアップ方法などを書きましょう．ムの使い方，セットアップ方法などを書きましょう．ここにプログラムの使い方，セットアップ方法などを書きましょう．

A.2　使い方

ここにプログラムの使い方，セットアップ方法などを書きましょう．ここにプログラムの使い方，セットアップ方法などを書きましょう．ここにプログラここにプログラムの使い方，セットアップ方法などを書きましょう．ムの使い方，セットアップ方法などを書きましょう．ここにプログラムの使い方，セットアップ方法などを書きましょう．ここにプログラムの使い方，セットアップ方法などを書きましょう．

A.3　ソースコード

リスト A.1: 画像の保存 1

```
1  #from email.mime import image
2  import re
3  #from symbol import typelist
4  from ultralytics import YOLO
5  import cv2
6  from PIL import Image
7  import random
8  import os
9  import yt_dlp
10 import matplotlib.pyplot as plt
11 import numpy as np
12 import glob
13 import torch
14 import shutil
15 from pytube import YouTube
16
17 class project_processing:
18     def get_random_frame(self, name, video_path, out_path, img_sum):
19         cap = cv2.VideoCapture(video_path)
20         frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
21         #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: 画像の保存 2

```
#/home/nozaki/yt-dlp/pythonにある
#python save.py 00img_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
```

```
17      type = args.type
18      pp_instance_den = pp.project_processing()
19
20      #これから以下のフォルダに画像を保存する
21      base_path = "/home/nozaki/yt-dlp/"
22      video_path = base_path + f"concat/{type}_con.webm"
23      #ランダム画像の保存先
24      out_path = base_path + f"img/{type}/before/"
25      pp_instance_den.get_random_frame(type, video_path, out_path, img_sum)
26      #クロップ用の変数
27      input_path = out_path
28      output_path = base_path + f"img/{type}/"
29      pp_instance_den.cropping(type, input_path, output_path)
30      print(f"{output_path} に保存")
31
32  if __name__ == "__main__":
33      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