

# YOLOv8マニュアル

---

Last updated: 2024/05/04 10:59:54

- YOLOv8はUltralytics社が開発した物体検出ソフトウェアです。
- YOLOv8は物体検出，物体追跡，骨格推定などともに，扱うことのできる物体の学習も可能です。

## 公式HP

<https://github.com/ultralytics/ultralytics>

## 前提

- NVIDIAのGPUを実装したWindows11 PC

## ターミナル(PowerShell 7)のインストール

- [ここ](#)からインストーラをダウンロードして，インストールする。
- インストール時に以下にチェックを入れる
  - Add 'Open here' context menu to Explorer
  - Add 'Run with PowerShell 7' context menu for PowerShell files

## Anacondaのインストール

- [ここ](#)からインストーラをダウンロードして，インストールする。
- Anaconda Promptで以下を実行する。
  - conda init powershell

## YOLOv8のインストール

- 適当なフォルダで，ターミナルを用いて以下を実行

```
conda create -n yolov8 python=3.12
conda activate yolov8
pip install ultralytics
```

## 物体検出

- 以下のプログラムをdetect.pyとして保存する。

```
from ultralytics import YOLO

model = YOLO("yolov8x.pt")

results = model("https://ultralytics.com/images/bus.jpg", save=True,
save_txt=True, save_conf=True)
```

```
boxes = results[0].boxes
for box in boxes:
    print(box.data)
print(results[0].names)
```

- detect.pyを実行する。
- runs/detectフォルダの中のpredict?フォルダの中に実行結果（jpgファイル）が保存される。?は整数で、実行するごとに1増加する。ただし、そのフォルダを削除してから実行すると?の値は変わらない。
- predict?/labelsフォルダには認識結果を記録したテキストファイルが生成される。動画の場合はフレームごとにファイルが生成される。各行に、物体ID, 始点x, 始点y, 終点x, 終点y, 信頼度, が記録されている。（座標系は相対座標である。）
- 認識結果はresults[0].boxes[i].dataにも格納されている。物体ごとに、始点x, 始点y, 終点x, 終点y, 信頼度, 物体ID, が記録されている。（座標系は絶対座標である。）
- 物体IDの一覧はresults[0].namesで得られる。
- modelの第1引数を動画（mp4ファイル）にしても、そのまま解析できる。フレームjの結果はresults[j]に格納される。

## 物体の学習

### データセットの準備

- 学習用の画像データ（100枚程度）を収集する。
- labelImgなどのアノテーションツールのインストールし、実行する。Python3.9で動作する。

```
pip install labelImg
labelImg
```

- Open Dirで画像データフォルダを開く。
- Change Saved Dirでアノテーションデータを保存するフォルダを指定する。
- 出力フォーマットがYOLOでない場合、PascalVOCをクリックして出力フォーマットをYOLOに設定する。
- Create RectBoxで画像にアノテーションを加える。
- Saveで保存する。
- 保存した画像データとアノテーションデータを7:3の割合で訓練(train)データと検証(val)データに分け、datasetsフォルダに以下のようなフォルダ構成にして保存する。

```
datasets
├── images
│   ├── train
│   └── val
└── labels
    ├── train
    └── val
```

- dataset.yamlは以下のような内容にする。trainは訓練データへのパス、valは検証データへのパス、ncはクラスの数、namesはクラスの名前。

```
train: images/train
val: images/val

nc: 2
names: ['class1', 'class2']
```

## 学習の実行

- 以下のプログラムlearn.py、datasetsフォルダ、dataset.yamlを同じフォルダに保存する。

```
from ultralytics import YOLO

if __name__ == "__main__":
    model = YOLO("yolov8m.pt")
    results = model.train(data="dataset.yaml", epochs=300)
```

- learn.pyを実行する。エポック数は300に設定する。学習に進展がなければ途中で学習は終了する。
- 学習モデルはruns/detect/train?/weights/best.ptに保存されるので、それを物体認識に用いる。
- 実行時にエラーが出る場合は、C:\Users\ユーザ名\AppData\Roaming\Ultralytics\settings.yamlを削除する。AppDataフォルダは隠しファイルなので、エクスプローラで表示できるようにしておく必要がある。

## 骨格推定

- 以下のプログラムをpose.pyとして保存する。

```
from ultralytics import YOLO

model = YOLO("yolov8x-pose.pt")

results = model("https://ultralytics.com/images/bus.jpg", save=True,
save_txt=True, save_conf=True)
keypoints = results[0].keypoints
print(keypoints.data)
```

- pose.pyを実行する。
- runs\poseフォルダの中のpredict?フォルダの中に実行結果（jpgファイル）が保存される。?は整数で、実行するごとに増加する。
- predict?/labelsフォルダには認識結果を記録したテキストファイルが生成される。各行に、物体ID、始点x、始点y、終点x、終点y、キーポイント1x、キーポイント1y、キーポイント1の信頼度、キーポイント2x、

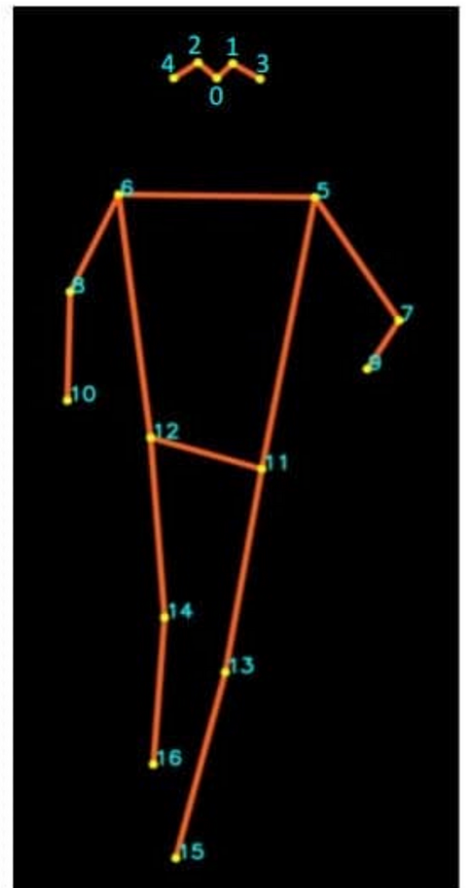
キーポイント2y, キーポイント2の信頼度, ..., が記録されている.

- 認識結果はresults[0].keypoints.dataにも格納されている.
- キーポイントの対応関係は以下の通り.

```
"categories": [  
  {  
    "supercategory": "person",  
    "id": 1,  
    "name": "person",  
    "keypoints": [  
      "nose", "left_eye", "right_eye", "left_ear", "right_ear",  
      "left_shoulder", "right_shoulder",  
      "left_elbow", "right_elbow", "left_wrist", "right_wrist",  
      "left_hip", "right_hip",  
      "left_knee", "right_knee", "left_ankle", "right_ankle"  
    ],  
    "skeleton": [  
      [16, 14], [14, 12], [17, 15], [15, 13], [12, 13], [6, 12], [7, 13], [6, 7],  
      [6, 8], [7, 9], [8, 10], [9, 11], [2, 3], [1, 2], [1, 3], [2, 4], [3, 5], [4, 6], [5, 7]  
    ]  
  },  
  ...  
]
```



Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



- modelの第1引数を動画（mp4ファイル）にしても、そのまま解析できる。フレームjの結果はresults[j]に格納される。

## 物体追跡

```
import cv2
from ultralytics import YOLO

# YOLOv8モデルをロード
model = YOLO("yolov8n.pt")

# ビデオファイルを開く
video_path = "sample.mp4"
cap = cv2.VideoCapture(video_path)

# ビデオフレームをループする
while cap.isOpened():
    # ビデオからフレームを読み込む
    success, frame = cap.read()

    if success:
        # フレームでYOLOv8トラッキングを実行し、フレーム間でトラックを永続化
        results = model.track(frame, persist=True)

        # フレームに結果を可視化
        annotated_frame = results[0].plot()

        # 注釈付きのフレームを表示
        cv2.imshow("YOLOv8トラッキング", annotated_frame)

        # 'q'が押されたらループから抜ける
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # ビデオの終わりに到達したらループから抜ける
        break

# ビデオキャプチャオブジェクトを解放し、表示ウィンドウを閉じる
cap.release()
cv2.destroyAllWindows()
```

## GPUの利用

- GPUが利用可能であるかの確認は以下のコードを実行し、Trueであれば利用可能である。

```
import torch
print(torch.cuda.is_available())
```

- 利用可能でない場合は、以下の手順を行う。

- GPUの型番を確認する。タスクバーを右クリックし、タスクマネージャを起動し、パフォーマンス→GPUで確認する。GPUがNVIDIA製でなければ利用できない。
- [ここ](#)からGPUに対応する最新のNVIDIAドライバをダウンロードし、インストールする。
- [ここ](#)からCUDA12.1をダウンロードし、インストールする。
- PytorchのGPU版をインストールする。

```
pip uninstall torch torchvision
pip install torch torchvision --extra-index-url
https://download.pytorch.org/whl/cu121
```

- GPUが利用可能であるか、再確認する。

## 補足：YouTube動画のダウンロード

- yt\_dlpをインストールする。

```
pip install yt-dlp
```

- 以下のプログラムを実行する。

```
from yt_dlp import YoutubeDL

# ダウンロード条件を設定する。今回は画質・音質ともに最高の動画をダウンロードする
ydl_opts = {"format": "best"}

# 動画のURLを指定
with YoutubeDL(ydl_opts) as ydl:
    ydl.download(["https://youtu.be/_YZZfaMGE0U"])
```

- 動画のURLはYouTube画面を右クリックし、「動画のURLをコピー」で得られる。

## 参考文献

- つくもちブログ, YOLOv8まとめ, <https://tt-tsukumochi.com/archives/6275>
- MS COCO datasetのフォーマットのまとめ, <https://qiita.com/kHz/items/8c06d0cb620f268f4b3e>
- YOLOv8でオリジナルデータの物体検出をする, <https://farml1.com/yolov8/>
- Ali MUstofa. YoloV8 Pose Estimation and Pose Keypoint Classification using Neural Net PyTorch. <https://alimustooftaa.medium.com/yolov8-pose-estimation-and-pose-keypoint-classification-using-neural-net-pytorch-98469b924525>