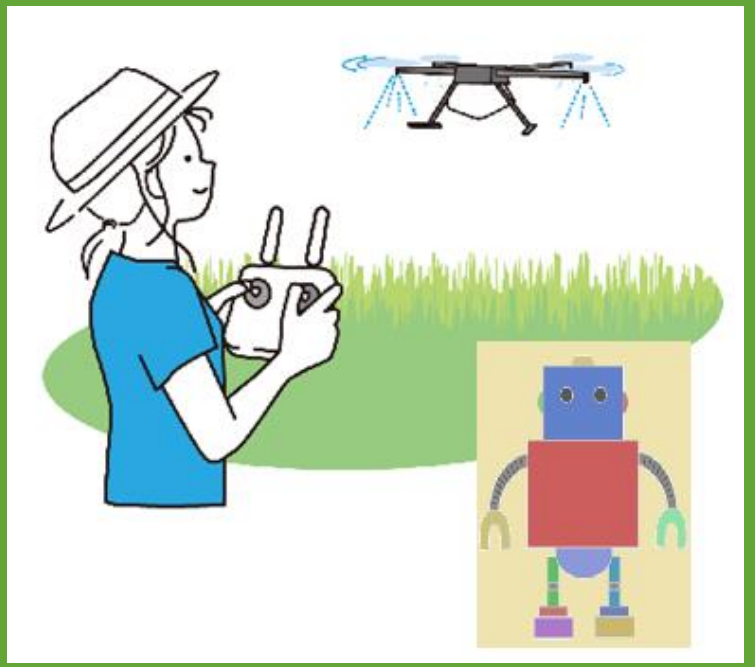
プロダクト開発演習

藤原　昇

2022年6月26日



1. テーマ選定

- 目的: 農作物生産作業の効率化、省人化
- 背景: 日本の食料自給率低下、農業従事者の高齢化の一方で、人工知能、高速通信（5G）、ドローン、ロボティクスなどICT技術拡大を背景として、スマート農業への期待が高まっている。
- 採用手段: 農作物の生育状況を画像から判定する。今回の演習では、トマト画像から、トマト果実、1個ずつの成熟度をクラス分類する。このため、セグメンテーションの新分野となる「オブジェクト・インスタンス・セグメンテーション」の適用を行う。静止画、動画どちらでも判断できることを要件のひとつとする。
- 期待効果: 生育状況の把握、収穫時期・地点の推定、収穫量予測、収穫作業の自動化のシステム／プロセスに効果的な手段の提供

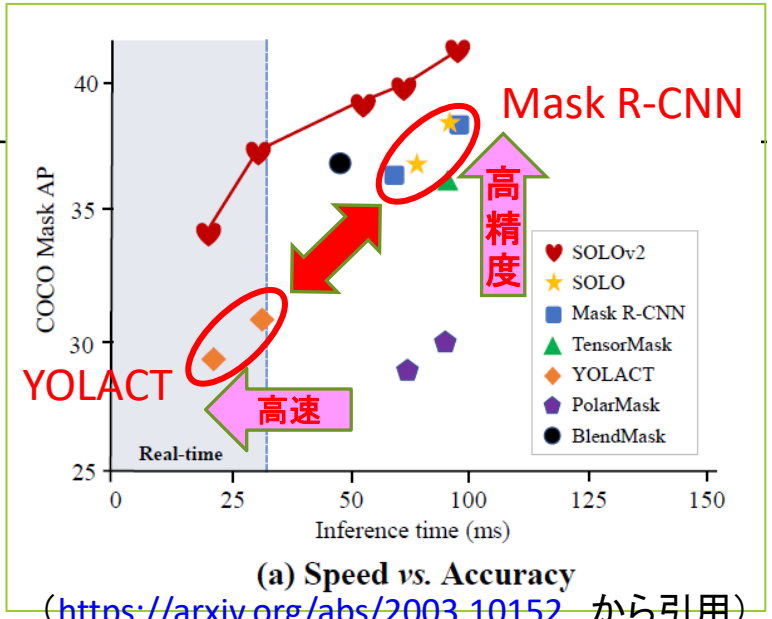
【テーマ】トマト画像（静止画、動画）に対するインスタンス・セグメンテーションを実施し、画像中、果実の位置を個別にマスクし、その成熟度（成熟、中間、未成熟）クラスを判定する

2. 参考情報の収集（1）

a. 公開されている学習用ラベル済みトマト画像と学習済みモデル
   株式会社LABORO.AIがトマト画像・物体検出データセット『Laboro Tomato』を公開（2020-07-14）。
   https://laboro.ai/activity/column/engineer/laboro-tomato/
   https://github.com/laboroai/LaboroTomato
b. インスタンス・セグメンテーションをサポートするPyTorch対応ツールキット
   PyTorch向けの物体検出ライブラリーとしては、Detectron2（Meta社）、MMDetection（OpenMMLab）などがある。なおLaboro TomatoではMMDetectionを使い、学習済みパラメタ（checkpoint)とそのMask R-CNNモデルのconfig情報、アノテーション済みdataset（MS COCO形式）を提供。
   https://mmdetection.readthedocs.io/en/latest/
   https://github.com/open-mmlab/mmdetection
c. インスタンス・セグメンテーションのアルゴリズム動向
   物体検出モデル（分類：cls、回帰：BBox）としてはFaster R-CNN, YOLO, SSDなど、またインスタンス・セグメンテーション（cls, BBox, & mask）としては、Mask R-CNN(2017), YOLACT(2019), SOLO(2020)などがある。
d. 無料で利用可能なフリーライセンス画像ソース
   ロイヤルティフリーで利用可能な画像の提供サイトがあり、一定の条件下で学習用画像としても利用できるものもある。今回、検証用画像の一部に「写真AC」（ https://www.photo-ac.com/ ）の画像を利用する。（個人での商用利用可、加工可）
   Windowsで利用できるアノテーション用ツールとしては、labelme（power shellから起動）、coco-annotator（docker コンテナ）などがある。（今回は使用せず）

2. 参考情報の収集(2)

- Mask R-CNN：https://arxiv.org/abs/1703.06870 (He et al., 2018)
  特徴：B-Box高精度、多機能（姿勢検出に拡張可能）、～5 fps
- YOLACT：https://arxiv.org/abs/1904.02689 (Bolya et al., 2019)
  特徴：判定（inference）の高速性、B-Box回帰精度はやや低
  29.8 mAP on MS COCO at 33.5 fps evaluated on a single Titan X
- SOLO：https://arxiv.org/abs/1912.04488 (Wang et al., 2020)
- SOLOv2：https://arxiv.org/abs/2003.10152 (Wang et al., 2020)
  特徴：高精度かつ高速



(a) Speed *vs.* Accuracy

（https://arxiv.org/abs/2003.10152 から引用）

Faster R-CNNに対し
この分岐(mask_head)を追加

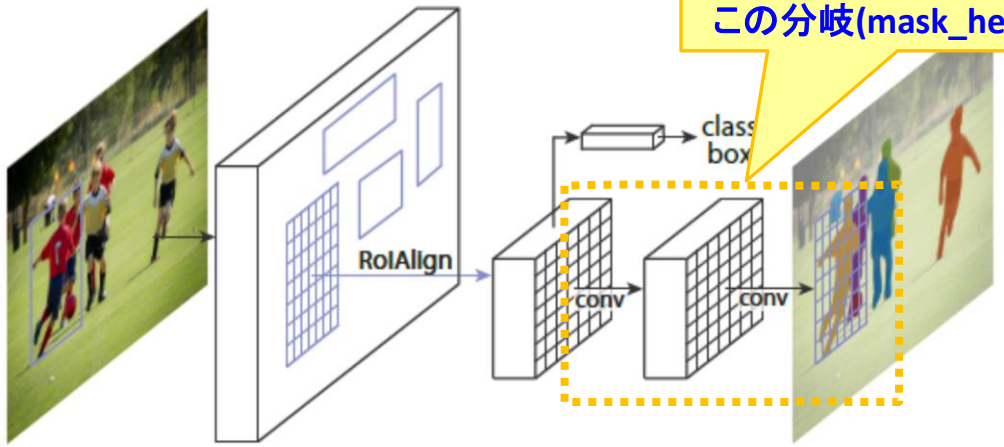


Figure 1. The Mask R-CNN framework for instance segmentation.

（https://arxiv.org/abs/1703.06870 から引用）

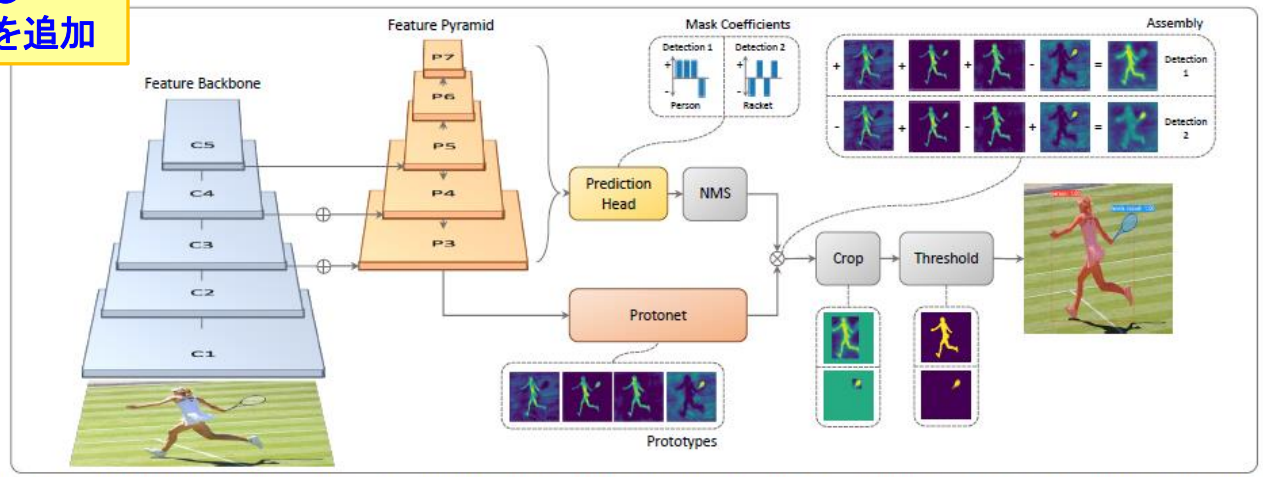


Figure 2: YOLACT Architecture Blue/yellow indicates low/high values in the prototypes, gray nodes indicate functions that are not trained, and $k = 4$ in this example. We base this architecture off of RetinaNet [27] using ResNet-101 + FPN

（https://arxiv.org/abs/1904.02689 から引用）

3. 実施方針の決定

a. Laboro Tomato Datasetを利用し、MMDtectionフレームワーク上でMask R-CNNモデルの検証を行う
b. YOLACTモデルをファインチューニングさせ、Mask R-CNNモデルとの比較検証を行う
c. 学習(train)、評価(test)用データはLaboro Tomatoを使い、MMDetectionフレームワークで実行する
d. 上記とは別に、検証用データとしてスマートフォン撮影動画・静止画、ロイヤルティフリー画像を使用する
e. 本課題中では、検証用データにはアノテーションを行わず、出力された推定結果を目視確認して、定性的な評価と考察のみ行う
f. 以下のステップで実施する
   ① MMDetectionフレームワークの動作確認、取扱い習得
      公開tutorialによるMMDetectionの実行環境構築、動作確認
      https://github.com/open-mmlab/mmdetection/blob/master/demo/MMDet_InstanceSeg_Tutorial.ipynb
   ② Laboro Tomato dataset, pretrained modelの検証
      実行環境を再現、test dataによる正当性の確認（validation）、新たに準備したデータによる検証（verification）
   ③ Laboro Tomato datasetを使い、YOLACT modelに切り替えての検証
      実行環境構築（COCO2017学習済みconfig）、train dataによるファインチューニング、test dataによる評価・計測（evaluation）、新たに準備したデータによる検証（verification）

4. データセット準備（1）

Laboro Tomatoデータの内訳

```
name: tomato_mixed
images: 643 train, 161 test
cls_num: 6
cls_names: b_fully_ripened, b_half_ripened, b_green,
           l_fully_ripened, l_half_ripened, l_green
```

＃学習用：643、評価用：161のjpegファイル

＃トマト：成熟、中間、未成熟、ミニトマト：成熟、中間、未成熟の6クラス

```
total_bboxes: train[7781], test[1,996]
bboxes_per_class:
    *Train: b_fully_ripened[348], b_half_ripened[520], b_green[1467],
            l_fully_ripened[982], l_half_ripened[797], l_green[3667]
    *Test:  b_fully_ripened[72], b_half_ripened[116], b_green[387],
            l_fully_ripened[269], l_half_ripened[223], l_green[929]
```

datasetディレクトリ構造

```
data
├── laboro_tomato
    ├── annotations    ### COCO annotation
    │     ├── train.json, test.json
    ├── train    ### train image datasets, 643 jpegファイル
    ├── test     ### test image datasets, 161 jpegファイル
                 ### image_resolutions: 3024x4032, 3120x4160の2種混在
```

COCO annotation 形式jsonファイル構造：

```
{
    "images": [image],
    "annotations": [annotation],
    "categories": [category]
}


image = {
    "id": int,
    "width": int,
    "height": int,
    "file_name": str,
}

annotation = {
    "id": int,
    "image_id": int,
    "category_id": int,
    "segmentation": RLE or [polygon],
    "area": float,
    "bbox": [x,y,width,height],
    "iscrowd": 0 or 1,
}

categories = [{
    "id": int,
    "name": str,
    "supercategory": str,
}]
```

4. データセット準備（2）

data/eval_tomato　フォルダ内　評価用静止画ファイル（アノテーションなし）

File名	サイズ	記事
eval_001.jpg	640×480	ミニトマト、水滴付き
eval_002.jpg	640×480	トマト、水滴付き
eval_003.jpg	640×480	ミニトマト
eval_004.jpg	640×427	トマト
eval_005.jpg	427×640	トマト
eval_006.jpg	1920×1280	ミニトマト、箱入り、多品種
eval_007.jpg	640×640	ミニトマト
eval_008.jpg	640×360	ミニトマト、動画ファイルから カットした為ピンボケ気味
eval_009.jpg	640×427	リンゴ、木成り
eval_010.jpg	427×640	リンゴ、木成り

data/video_tomato フォルダ内
評価用動画ファイル（アノテーションなし）

File名	サイズ	記事
tomato1.mp4	960×540	00:00:08, 30.00 flm/s
tomato2.mp4	640×480	00:00:03, 29.97 flm/s
tomato3.mp4	640×480	00:00:58, 29.97 flm/s

スマートフォン撮影したオリジナルファイルは、サイズ：
640×360、フレームレート：24.00flm/sで記録されているので、
PC上の動画編集ソフトにて編集を実施、
また音声データを削除。

5. フレームワーク事前調査

今回使用するMMDetectionフレームワークのチュートリアルを再現する。
これによりフレームワークの使い方を身に付けるとともに、正常にインスタンス・セグメンテーションが実行できる環境を構築する。
再現実行するチュートリアルは、google Colab環境でGPUを使用することを前提とし、Mask R-CNNを使ったものを選定する。
https://github.com/open-mmlab/mmdetection/blob/master/demo/MMDet_Tutorial.ipynb
この実施結果を”**05prelim_survey.ipynb**”にて示す。

【実施結果】
MMDetection v2.24.0 ～v2.25.0(現時点の最新リリース版)では、上記チュートリアルはエラーとなる。下記の通りこのエラーを解消した結果、pretraind modelを使ってのセグメンテーション動作の確認、および新たなdatasetを使い、風船の有無のみのクラスをセグメントするために、ファインチューニング学習させる方法の動作確認を行った。

症状: tools/train.pyを実行すると、”### AttributeError: ‘ConfigDict’ object has no attribute ‘device’”エラーが発生する。
なお、参照しているチュートリアルの実行環境はMMDetection v2.21.0であることが示されている。
このエラーについては、v2.24.0リリース後、たびたびGitHub上issueとして報告され、対策提案がなされている。その対策を実施しようとしたが、自分の環境では解決しなかった。
対策:MMDetection v2.23.0にバージョンダウンすることで解決し、チュートリアルの結果を確認できた。
なおその際は、mmcvについても最新版(1.5.x)ではなく、1.3.17にバージョンダウンさせる必要がある。

６．LaboloTomato事前学習モデルの検証

実施結果を”06LaboroTomato_verif.ipynb”にて示す。

６．１　MMDetectionのロード

６．２　LaboroTomato model実行環境の再現

６．３　LaboroTomato/test データによる正当性の確認（validation）

画像161枚、検証時間：0.1～0.3task/sなので、画像1枚当たり3~10秒程度を要している。
Average PrecisionはBoundary Boxについて64.7%、セグメンテーションについて66%。
LaboroTomato GitHub上に記載された訓練時データでは「bbox AP:64.3, mask AP:65.7」なので、
ほぼ再現されていると判断できる。

'bbox_mAP'	0.647	'segm_mAP’	0.66
'bbox_mAP_50’	0.822	'segm_mAP_50'	0.818
'bbox_mAP_75'	0.735	'segm_mAP_75'	0.736
'bbox_mAP_s'	0.0	'segm_mAP_s',	0.0
'bbox_mAP_m'	0.146	'segm_mAP_m'	0.131
'bbox_mAP_l'	0.681	'segm_mAP_l'	0.697

６．LaboloTomato事前学習モデルの検証

６．４　新たに準備したデータによる検証（verification）

File名	結果	記事
eval_001.jpg	×	水滴がつくと正しく判定できない。水滴をトマトと誤検出。
eval_002.jpg	△	奥の2個は葉っぱ、水滴に紛れて認識できていない
eval_003.jpg	△	光線、枝の影などの影響か、2個重なっていると誤検出
eval_004.jpg	△	茎やヘタ部分で仕切られると、複数有りと誤検出
eval_005.jpg	△	同上
eval_006.jpg	○	箱に数多く有るが良好に検出、黄色トマトは未成熟と認識
eval_007.jpg	○	
eval_008.jpg	×	ピンぼけ写真だとかなり未検出、またミニを通常サイズと誤認識している
eval_009.jpg	×	リンゴは学習していないのでトマトと誤認識される
eval_010.jpg	×	同上

６. LaboloTomato事前学習モデルの検証

６．４　新たに準備したデータによる検証（verification）

動画ファイル　tomato3.mp4は58秒、1752フレームであるが、判定速度は**5.3 task/s**であった。
1 taskを1 flameと読み替えれば、論文中に記載の5fps並みの実行速度は出ているといえる。
結果動画を見る限りでは、葉陰に隠れているものは検出できない場合がたびたびあるが、全体的には良好な判定結果のように見える。

なお、今回の実行環境は、Google Colab Pro、GPU：有り、メモリ：ハイメモリで演算させた。
‘GPU 0’: ‘Tesla P100-PCIE-16GB’（GPUメモリ：25.46GB）

７．別モデルでの学習・評価・検証

公開されているLaboro Tomato Datasetを使い、別の特徴を持つYOLACTモデルにて、インスタンス・セグメンテーションの検証を行う。
Mask R-CNNモデルと比較し、YOLACTモデルは判定(infelence)速度が速く、条件次第では30fpsを上回るとされている。もしもその速度がエッジデバイスでも出るのであれば、WebCamからの動画入力に対して、ローカルにリアルタイムでセグメンテーション、クラス判定が行えると期待できるため、今回比較対象として取り上げる。

実施結果を”**07YOLACT_verif.ipynb**”にて示す。

７．１　MMDetectionのロード

７．２　LaboroTomato dataset、YOLACTモデル実行環境の構築
config設定用ファイルyolact_r50_1x8_coco.pyを、LaboroTomato dataset、GoogleColab環境に合わせて yolact_r50_1x8_coco_tomato.pyに書き換える。
この時、特にdata = dict()内のsample_per_gpu, workers_per_gpuの各ハイパーパラメタの調整が難しく、次節７．３の学習時のエラー発生具合を見ながら、最終的には以下の値に調整した。
(4, 2), (8, 1), (1, 1)などの組み合わせでは早々に実行環境にてRun Timeエラーが発生するため、最終的に(4, 1)の値に調整している。

```
data = dict(
    samples_per_gpu = 8,
    workers_per_gpu = 4,
    ....
```

```
data = dict(
    samples_per_gpu = 4,
    workers_per_gpu = 1,
    ....
```

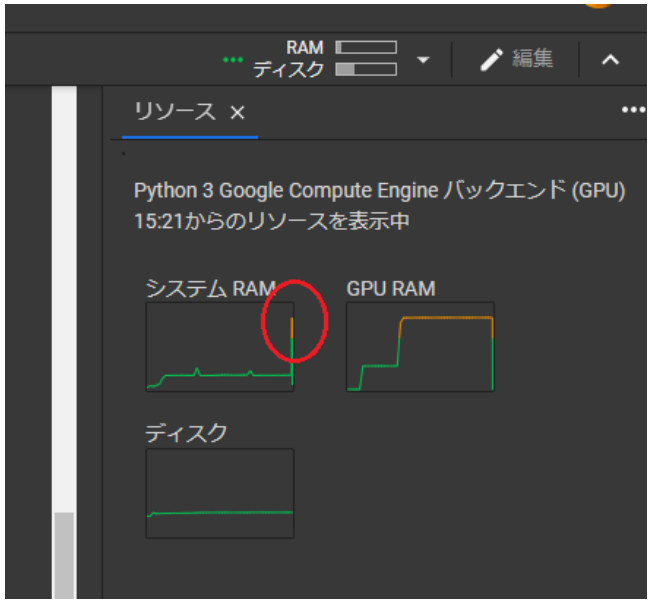
7．別モデルでの学習・評価・検証

7．3　LaboroTomato/train データによる学習（training）

学習前にCOCOデータで事前学習済みのcheckpoint：yolact_r50_1x8_coco_20200908-f38d58df.pthをロードさせ、これをLaboroTomato Trainデータによってファインチューニングさせる。学習は、single GPUに対し4sampleを処理させるので、画像643枚の学習について、1epoch辺りでは161回のバッチを通す設定となっている。

今回、12epoch実行させる設定としていたが、Epoch 2を過ぎた後、Epoch3の途中でRun Timeエラーが発生する事態となった。（再現性あり）。1epoch辺り20～30分程度要しており、エラー発生時システムRAMの異常が見られるので、実行環境のリソース不足と推定し、これ以上の実行を断念した。

“RuntimeError: DataLoader worker (pid 1764) is killed by signal: Killed. “



なお、Epoch2までのAP、ARについては、以下に示す値となっている。
YOLACTのCOCO val ではmAP:29.0と公開されているので、それなりには使えるのではと推定し、更に評価・検証を継続した。

Epoch 1終了時　bbox AP: 0.146, segm AP: 0.159　# 平均Precision
　　bbox AR: 0.501, segm AR: 0.548　# 平均Recall
Epoch 2 終了時　bbox AP: 0.269, segm AP: 0.285
　　bbox AR: 0.546, segm AR: 0.548

７．別モデルでの学習・評価・検証

７．４　LaboroTomato/testデータによる評価（validation）
学習中Epoch2まで完了の後、Epoch3途中で google ColabのRun Timeエラー発生！！
エラー発生前のEpch 2までの学習結果をtestデータを使い評価する。このためconfigデータを書き換える

この結果は、当然ながら、学習時のEpoch2終了時点の評価結果と一致した値となった。

bbox AP: 0.269, segm AP: 0.285
bbox AR: 0.546, segm AR: 0.548

これにて、Laboro Tomatoデータセットでの学習済みモデルが準備できたとみなし、
次節にて、新データによる検証を行うこととした。

7．別モデルでの学習・評価・検証

7．5　新たに準備したデータによる検証（verification）

detectorの構築を行い、検証用の画像、動画ファイルを判別させ、出力結果を目視評価する。

File名	YOLACT	Mask RCNN時	記事
eval_001.jpg	△	×	水滴がついていてもトマトを検出するが、サイズ（通常/ミニ）の区別が不正確
eval_002.jpg	○	△	奥のトマト（2個中の1個）まで検出できている
eval_003.jpg	×	△	Bboxの位置がずれているか、または目立つ個体が検出できていない
eval_004.jpg	×	△	熟成したトマト2個が検出できておらず、未成熟の2個も確率値が低い
eval_005.jpg	△	△	マスクはできているが、確率値が低い
eval_006.jpg	△	○	成熟したトマトの検出ができていなかったり、クラスを誤る
eval_007.jpg	○	○	
eval_008.jpg	△	×	MaskR-CNNと比較して多少良い
eval_009.jpg	×	×	リンゴは学習していないのでトマトと誤認識される
eval_010.jpg	×	×	同上

7．別モデルでの学習・評価・検証

7．5　新たに準備したデータによる検証（verification）（続き）
動画ファイル　tomato3.mp4は58秒、1752フレームであるが、判定速度は**6.8 task/s**であった。
Mask R-CNNモデルの場合5.3 task/sだったので、約28%改善しているが、論文中で示されている30fpsには程遠く、十分なパフォーマンスが出ていない。

実行環境の問題が大きいのかもしれない。
今回の実行環境は、Google Colab Pro、GPU：有り、メモリ：ハイメモリで演算させた。
‘GPU 0’: ‘Tesla P100-PCIE-16GB’（GPUメモリ：25.46GB）

８．まとめと考察

８．１　まとめ

a. Laboro Tomatoのデータセット、学習済みモデルにより、Mask R-CNNモデルによるインスタンス・セグメンテーションが実現されていることが確認できた。
b. Mask R-CNNモデルでは重なったオブジェクト、大小のオブジェクトが含まれるケースでも比較的良好に検出できている。（主観的意見）
c. 一方で、判定処理時間は5fps程度と遅く、リアルタイムでのWebCam動画判定は難しいという特徴も確認できた。
d. 別のモデルとなるYOLACTについて、学習中に実行環境のリソース不足と思われる原因で異常終了してしまい、十分な確認はできなかった。
e. 異常終了前の学習不完全なDetectorを使っても、ある程度のインスタンス・セグメンテーションが実現できていることは確認できた。一方、その特徴である判定速度については、確認が取れなかった。

８．２　考察

８．まとめと考察

８．２　考察

a. 今回、新規画像による検証結果をresultファイルとしても保存してあり、BBox位置データ（中心位置:　x, yとサイズ：h, w）、クラスラベルなどが格納されているはずであるが時間が足りず、そのデータ活用を行っていない。検出した果実の位置、成熟度別の個数などのデータに活用できると考えるが、それは今後の課題となるだろう。
b. 本演習の当初の狙いとしていた、農作物の収穫自動化を想定した場合、現状では収穫ロボットによる画像認識、収穫処理に使うには、まだ性能不足と考えられる。特にマニピュレータで収穫させるためには、2次元画像上の物体位置ではなく、RGB-Dカメラなどを使った3次元位置の検出が必要であり、この点でも更に改善と試行が必要と考える。
c. トマトなどの果実、野菜の収穫の場合、茎や葉の重なり、さらに光線やそれらの影の影響で、正しく対象物の成熟状態を判定させることが難しい。判定速度、判定精度ともに高いSOLOv2など、新しいモデルの評価も必要になるだろう。

付録：関係ファイル一覧

ファイル名	内容	備考
05prelim_survey.ipynb	MMDetection事前調査	jupyterNotebook
06LaboloTomat_verif.ipynb	LaboroTomato事前学習モデル確認	jupyterNotebook
07YOLACT_verif.ipynb	YOLACTモデルでの学習・評価・検証	jupyterNotebook
Laboro_tomato_etc	LaboroTomatoモデル用config関連ファイル（６件）	ディレクトリ
Yolact_tomato_etc	YOLACTモデル用config関連ファイル（３件）	ディレクトリ
eval_tomato	検証用トマト画像（１０件）	ディレクトリ
video_tomato	検証用トマト動画（３件）	ディレクトリ
eval_result	評価結果ファイル（静止画評価結果bin, tomato3.mp4）	ディレクトリ
work_dirs	YOLACTモデル学習結果関連ファイル	ディレクトリ