

2022年度情報処理（15週目）

YOLOv4参考資料

薛 軼同

立命館大学ロボティクス学科M1

実験概要

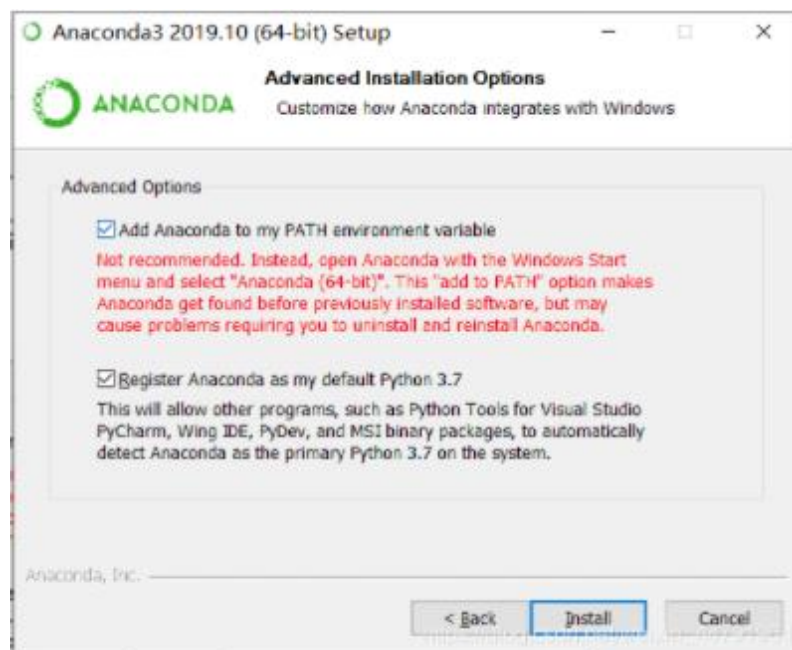
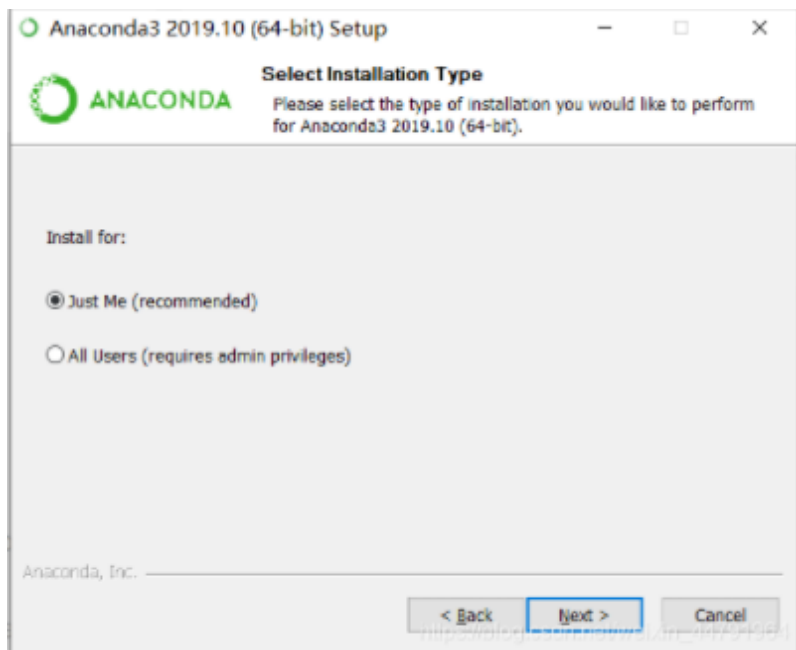
今回はYOLOv4というアルゴリズムを使い、認識を行う。

必要な作業は以下6つある

- 1 : Anacondaのインストール
- 2 : cuDNNとCUDAのインストール
- 3 : pytorch環境の配置
- 4 : 必要なライブラリのインストール
- 5 : データ写真の用意と画像のアノテーション処理
- 6 : トレーニングの実行

Anacondaのインストール

YOLOv4のアルゴリズムは今回pythonで実現したする。
プログラムを動かす為には、Anacondaを使用する。



CudnnとCUDAのインストール

- ダウンロード先：

[CUDAのダウンロード](#)

[cuDNNのダウンロード](#)

CUDA 11.0 cuDNN8.0.5.39は実際に使って成功したので推奨する。

CUDAインストールの注意事項

cuda_11.0.2_451.48_win10.exe

Exeファイルをクリックし、
セットアップを開始する



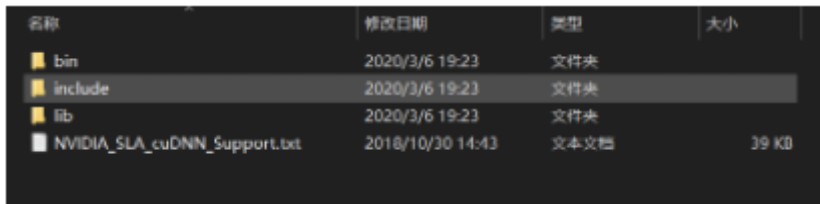
カスタマイズ(高級)を選択



全部チェック

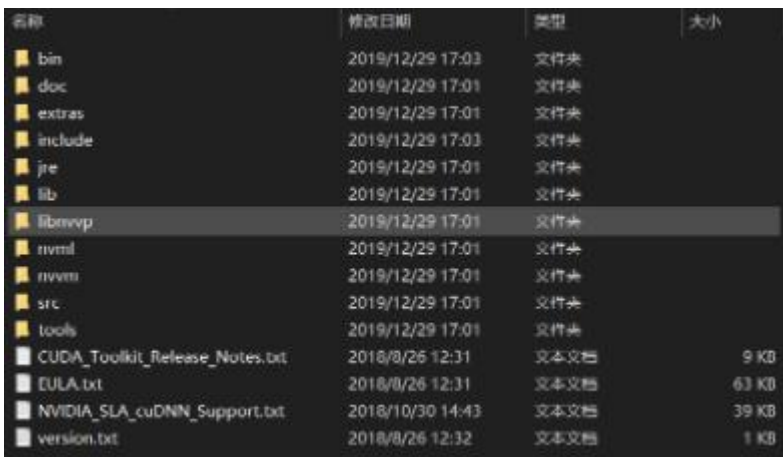
cuDNNの配置

1 : まず圧縮ファイルcudnnを展開する



名称	修改日期	类型	大小
bin	2020/3/6 19:23	文件夹	
include	2020/3/6 19:23	文件夹	
lib	2020/3/6 19:23	文件夹	
NVIDIA_SLA_cuDNN_Support.txt	2018/10/30 14:43	文本文档	39 KB

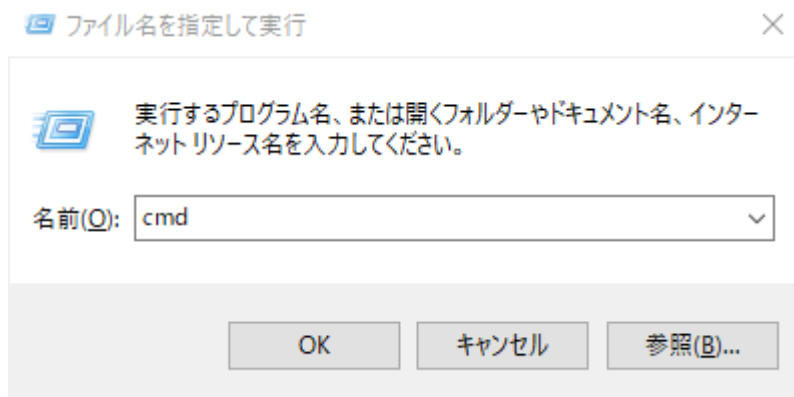
2 : 展開した内容(上の図)はCUDAのファイル(C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0)に貼り付け



名称	修改日期	类型	大小
bin	2019/12/29 17:03	文件夹	
doc	2019/12/29 17:01	文件夹	
extras	2019/12/29 17:01	文件夹	
include	2019/12/29 17:03	文件夹	
jre	2019/12/29 17:01	文件夹	
lib	2019/12/29 17:01	文件夹	
libmvp	2019/12/29 17:01	文件夹	
nvml	2019/12/29 17:01	文件夹	
nvvm	2019/12/29 17:01	文件夹	
src	2019/12/29 17:01	文件夹	
tools	2019/12/29 17:01	文件夹	
CUDA_Toolkit_Release_Notes.txt	2018/8/26 12:31	文本文档	9 KB
EULA.txt	2018/8/26 12:31	文本文档	63 KB
NVIDIA_SLA_cuDNN_Support.txt	2018/10/30 14:43	文本文档	39 KB
version.txt	2018/8/26 12:32	文本文档	1 KB

Pytorch環境の配置その1

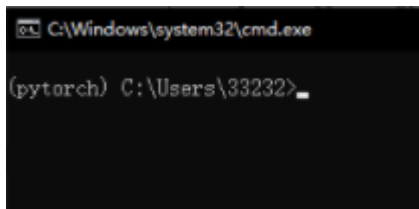
- Win+Rでcmdを入力



以下2つの指令を入力する

- `conda create -n pytorch python=3.6`
- `activate pytorch`

Pytorch環境の配置その2



```
C:\Windows\system32\cmd.exe
(pytorch) C:\Users\33232>
```

こういう画面が出ると思う

GPUある場合、以下の指令を入力する：

```
pip install tensorflow-gpu==2.4.0
```

CPUのみの場合、以下の指令を入力する：

```
pip install tensorflow-cpu==2.2.0
```


他のライブラリのインストール

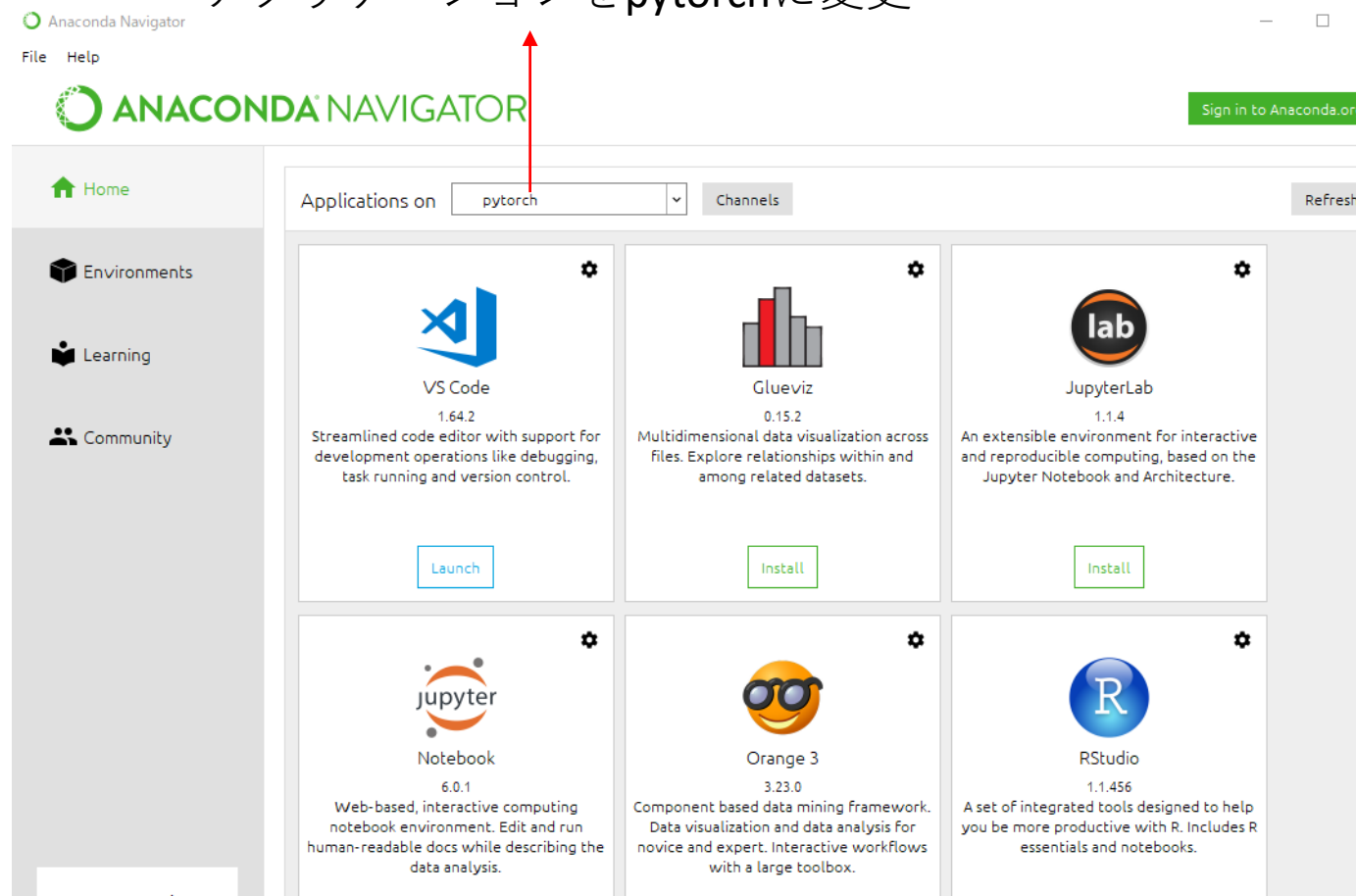
- `scipy==1.4.1`
- `numpy==1.19.2`
- `matplotlib==3.2.1`
- `opencv_python==4.2.0.34`
- `tensorflow_gpu==2.4.0 / tensorflow_cpu==2.2.0`
- `tqdm==4.46.1`
- `Pillow==8.2.0`
- `h5py==2.10.0`

例 : `pip install scipy==1.4.1`

VSCodeのインストール

プログラムの実行は様々な方式がありますが、vscodeを推奨する

アプリケーションをpytorchに変更



データ集の作成

アノテーションデータ

- 名前
- 1_Color.xml
- 2_Color.xml
- 3_Color.xml
- 4_Color.xml
- 5_Color.xml
- 6_Color.xml
- 7_Color.xml

```
<annotation>
  <folder>JPEGImages</folder>
  <filename>1_Color.png</filename>
  <path>C:\yolov4-pytorch-master\VOCdevkit\VOC2007 JPEGImages\1_Color.png</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>1280</width>
    <height>720</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Shrimo</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>486</xmin>
      <ymin>396</ymin>
      <xmax>654</xmax>
      <ymax>559</ymax>
    </bndbox>
  </object>
</annotation>
```

← → ↑ ↓ PC > ローカルディスク (C:) > yolov4-pytorch-master > VOCdevkit > VOC2007

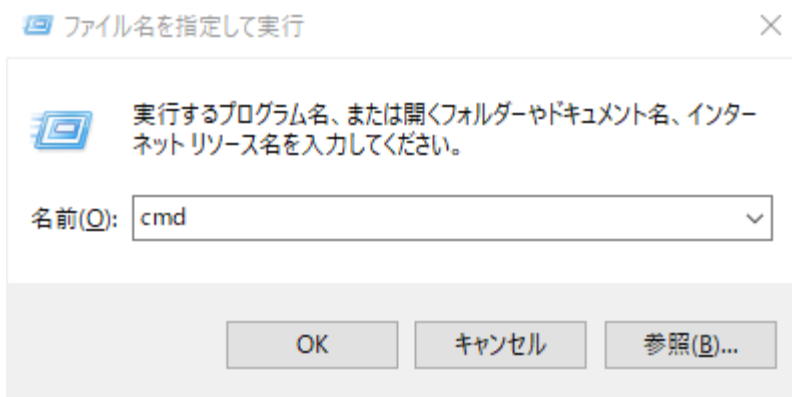
名前	更新日時	種類	サイズ
Annotations	2021/12/06 23:17	ファイル フォルダ	
ImageSets	2021/12/05 20:06	ファイル フォルダ	
JPEGImages	2021/12/07 1:17	ファイル フォルダ	

写真データ



アノテーション方法その1

まずlabellmgをダウンロードする
pip install labellmg



```
C:\> C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kawarabo>pip install labelling
```

インストール完了後、labellmgで実行

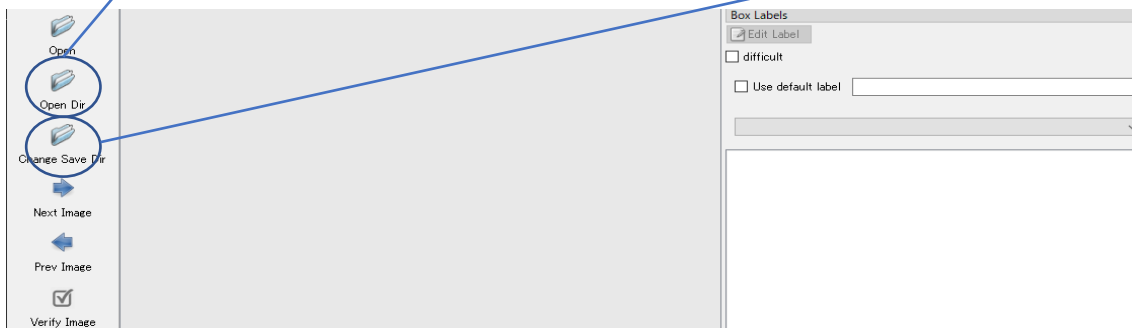
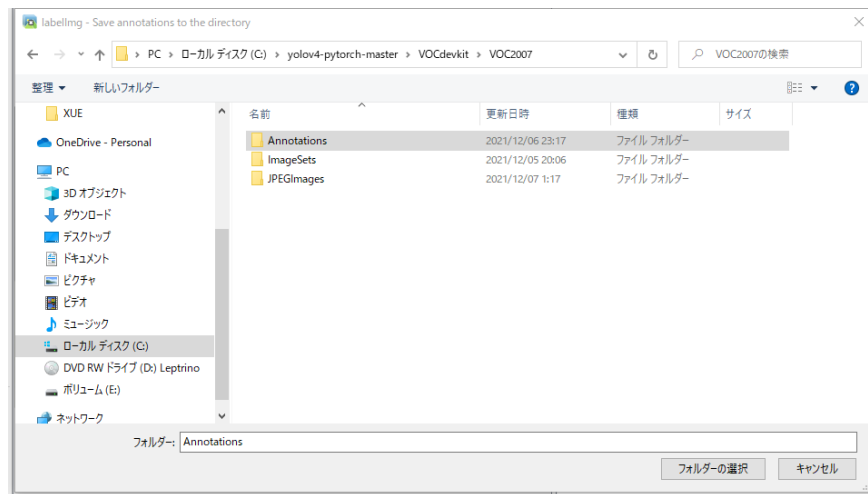
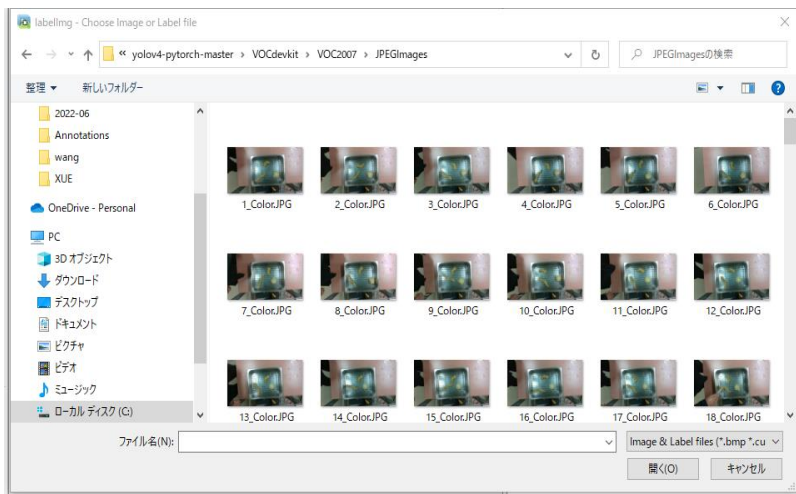
```
C:\> C:\WINDOWS\system32\cmd.exe - labelling
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kawarabo>labelling
```

アノテーション方法その2

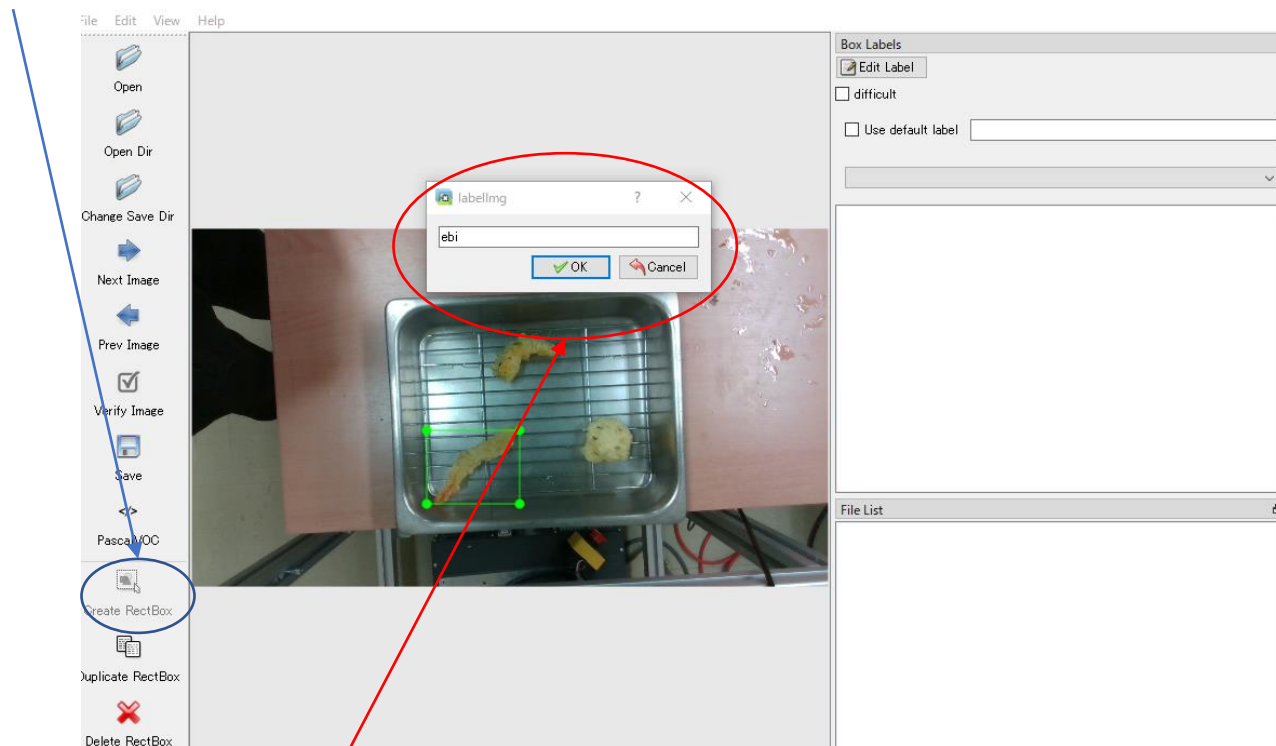
Open Dirで写真データの保存先を指定する

Change Save Dirでアノテーションデータを保存したいファイルを指定する



アノテーション方法その3

Create RectBoxをクリックして、矩形領域を作る
この領域で、目標をマーキングする



そしてターゲットの名前を付ける

プログラムの設定(train.py)

GPUない場合はFalseに調整

```
37
38  if __name__ == "__main__":
39      #-----#
40      # GPU使ってない場合は、Falseに変更
41      #-----#
42      Cuda = True
43      #-----#
44      # 要注意 classes_pathは自分のラベルファイルに対応する。
45      #-----#
46      classes_path = 'model_data/cls_classes.txt'
47      #-----#
```

cls_classes.txtファイルを作り、ラベルの名称を記入する
そしてclasses_passのパスはcls_classes.txtのパスに変更

```
*cls_classes.txt - メモ帳
ファイル(F) 編集(E) 書式(O)
ebi
rennkonn
kabocya
```

cls_classes.txtの例

プログラムの設定 (voc_annotation.py)

```
20 #-----#  
21 classes_path = 'model_data/cls_classes.txt'  
22 #-----#
```

↑
train.pyと同じく、自分で新しいデータセットを作る場合、
cls_classes.txtのパスに変更する

プログラムの実行順番(学習)

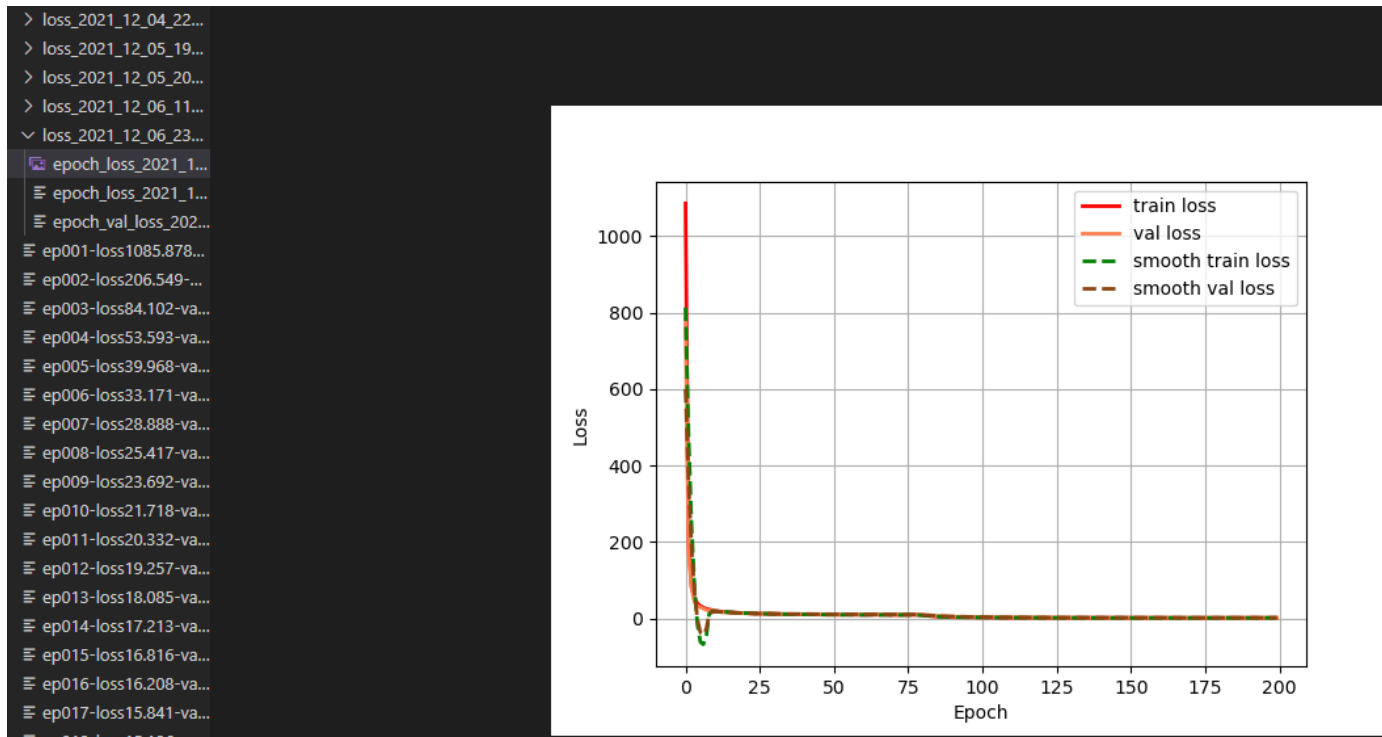
1 : **voc_annotation.py**先ず実行する

目的：学習用データ写真とテスト用データ写真に分ける

2 : **train.py**を実行して、学習を行う

学習結果の確認

学習と共に、モデル内部のパラメータは自動的に更新する、loss関数を使い、モデルを評価する。一般的に、0に近い方がいい。しかしながら、0に近すぎると「過学習」になり、モデルは使えない可能性はある



学習結果の検証

1つの学習結果を使い、model_pathに代入

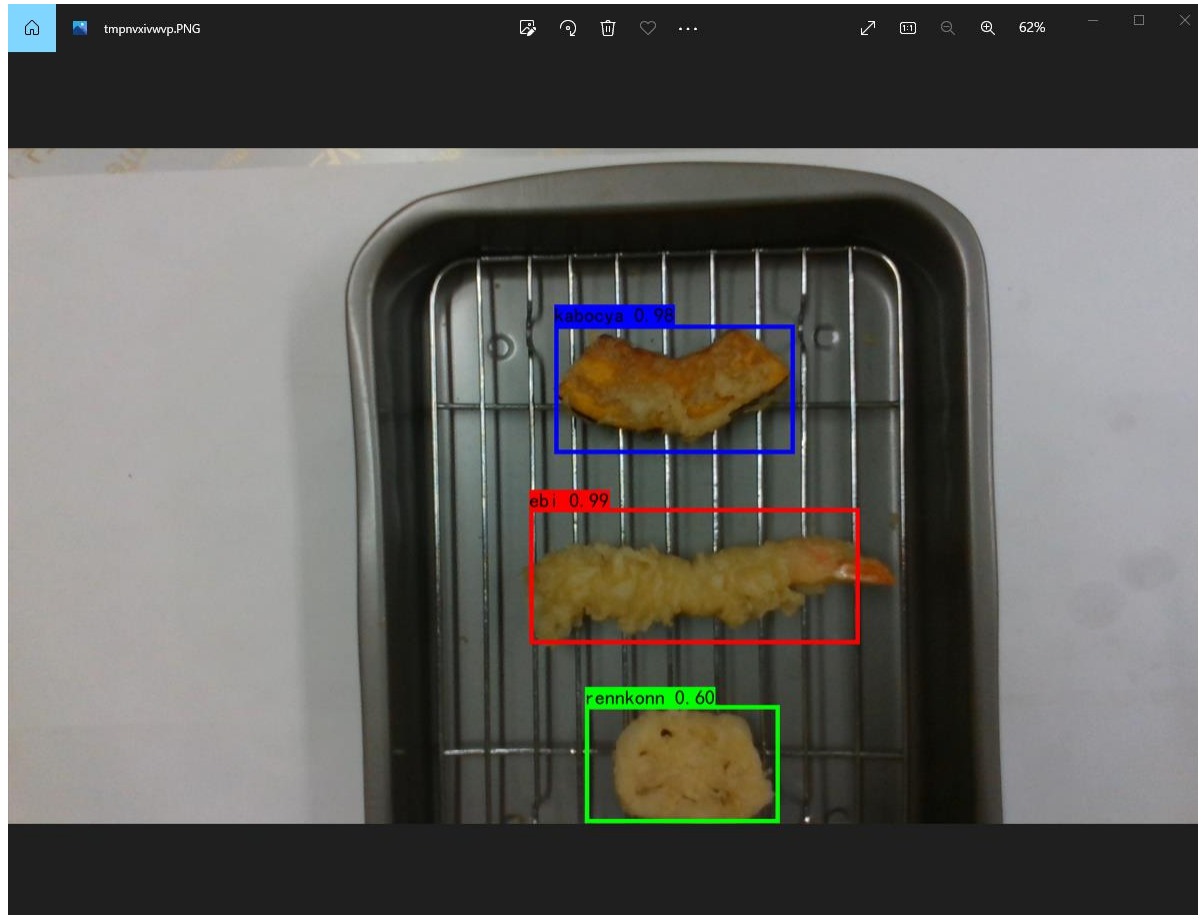
```
≡ ep193-loss1.897-val_loss2.054.pth
≡ ep194-loss1.524-val_loss2.052.pth
≡ ep195-loss1.807-val_loss1.946.pth
≡ ep196-loss1.743-val_loss1.919.pth
≡ ep197-loss1.657-val_loss2.003.pth
≡ ep198-loss1.686-val_loss2.023.pth
≡ ep199-loss1.857-val_loss1.946.pth
≡ ep200-loss1.829-val_loss1.938.pth
```

```
31 #-----
32 "model_path"      : 'logs/ep197-loss1.657-val_loss2.003.pth',
33 "classes_path"    : 'model_data/cls_classes.txt',
34 #-----
35 # checkpoints_path代表检测对应的.pth文件 一般不修改
```

yolo.py

そして最後は（predict.py）を実行し、
任意写真のバスを入力したら、検出を行える

出力結果



ターゲットを正しく検出しましたか？
できなかつたら、データ写真を増やしてみよう！