

FaBo JetFormer Docs

JetFormer ドキュメント

目次

1. JetRacer Docs	3
1.1 JetRacer Docsについて	4
1.2 PDF版	4
1.3 サポート Discord	4
1.4 修正依頼等	4
1.5 購入先(FaBo Store)	4
1.6 本ドキュメントでのセミナーの実施	5
1.7 関連記事	5
1.8 FaBo AI Hardware教材	5
1.9 FaBo教材	5
2. FaBo JetRacer Kit	6
2.1 FaBo JetRacer Kitについて	6
2.2 チュートリアルを進め方	6
3. JetRacer	7
3.1 基本編	7
3.2 項目別	25
3.3 設定	96
3.4 各種Tips	103

1. JetRacer Docs



FaBo JetRacer Kit JR1-S-P



FaBo JetRacer Kit JR1-S-C-D (GR86レッド完成イメージ)

🖥 1.1 JetRacer Docsについて

JetRacer Docsは、FaBo JetRacerの学習用ドキュメントです。

🖥 1.2 PDF版

本ドキュメント全体をまとめたPDFをダウンロードできます。

- [jetformer-docs.pdf](#)

🖥 1.3 サポート Discord

JetRacerに関するお問い合わせ、ご質問、トラブルなど

<https://discord.gg/StJ84Hb>

🖥 1.4 修正依頼等

GithubのRepoに[Issues](#)を投稿お願いいたします。

🖥 1.5 購入先(FaBo Store)

- [FaBo Store](#)

1.6 本ドキュメントでのセミナーの実施

- 2022年11月30日 [大分県立情報科学技術高校で授業](#)
- 2019年12月5日 JetRacerハンズオン(主催 NTTコムエンジニアリング)(35台)
- 2019年12月17日 JetRacerハンズオン(主催 NTTコムエンジニアリング)(35台)
- 2019年12月24日 JetRacerハンズオン(主催 NTTコムエンジニアリング)(35台)

1.7 関連記事

- [誰もが童心に帰って楽しくAIを習得できるJetRacerハンズオン](#)
- [NTTコムエンジが社員のAI学習教材にJetson Nano搭載「JetRacer」を採用](#) [深層学習とAI推論を体験](#) [マクニカらが提供](#)

1.8 FaBo AI Hardware教材

- [JetRacer Docs](#)
- [JetBot Docs](#)
- [DonkeyCar Docs](#)
- [EdgeAI Docs](#)
- [JetBook Docs](#)
- [NSynth Docs](#)

1.9 FaBo教材

- [Arduino Docs](#)
- [RaspberryPi Docs](#)
- [IchigoJam Docs](#)
- [Circuit Docs](#)
- [Denbun Docs](#)
- [NRF Docs](#)
- [Android Things Docs](#)
- [DeviceWebAPI Docs](#)

2. FaBo JetRacer Kit

2.1 FaBo JetRacer Kitについて

このプロジェクトは、[NVIDIA-AI-IOT/jetracer](#)のForkプロジェクトです。教育現場への導入を目的に、株式会社FaBoがキット化し、改良したバージョンとなります。

最新版のノートブックは、[FaBo JetRacer Kitレポジトリ](#)に、公開してあります。

2.2 チュートリアルを進め方

動作までの流れは以下の通りです。

- [2.組立](#)
- [3.設定](#)
- [4.学習と走行](#)

3. JetRacer

3.1 基本編

3.1.1 基本編

本章では、JetRacerの最新版を自動走行せる手順を簡易的、手短に解説します。

流れ

1～5の流れで進めていきます。

項目	Notebook	処理
1	01_find_pwm.ipynb	TT02(RCカー)のPWM値の調整
2	11_record_camera.ipynb	走行のカメラ画像を録画
3	13_annotation.ipynb	RCカーの走行させたい向きをアノテーション, 学習
4	16_convert.ipynb	TensorRTに変換し、量子化、軽量化
5	17_run.ipynb	RCカーで自動走行

【手順1】 01_find_pwm.ipywn

【手順2】 11_record_camera.ipynb

11_record_camera.ipynbで、走行時のカメラ画像を録画します。Notebookを下記画像のところまで実行します。dataset名は、任意の名前のdataset名にし、[Record] ボタンを押します。カメラの処理が問題なく動いている場合は、3秒ごとに、カメラのfpsの周期がメッセージで表示されます。

The screenshot shows the JupyterLab interface. On the left, the file explorer displays a list of notebooks in the 'jetracer / notebooks /' directory. '11_record_camera.ipynb' is selected. The main area shows the code editor for this notebook. The code defines a data collection widget with a title, task widget, dataset widget, annotation box, process widget, and buttons for start, stop, and release. The rendered output shows the widget with a dropdown menu for 'task' set to 'camera', a text input for 'dataset' set to 'aizu_data_101', and an 'Auto Annotation' checkbox. A 'Record' button is highlighted with a red box.

```
[14]: separator = ipywidgets.HTML('<hr style="border-color:gray;margin:10px 0"/>')
title1 = ipywidgets.HTML('<b> [1.カメラの録画] </b><br>データセット用の走行データを保存できます。走行データはcameraフォルダにデータが保存されます。')
title2 = ipywidgets.HTML('<b> [2.カメラの開放処理] </b><br>走行データの収集が完了したらカメラの開放処理を実行します。')

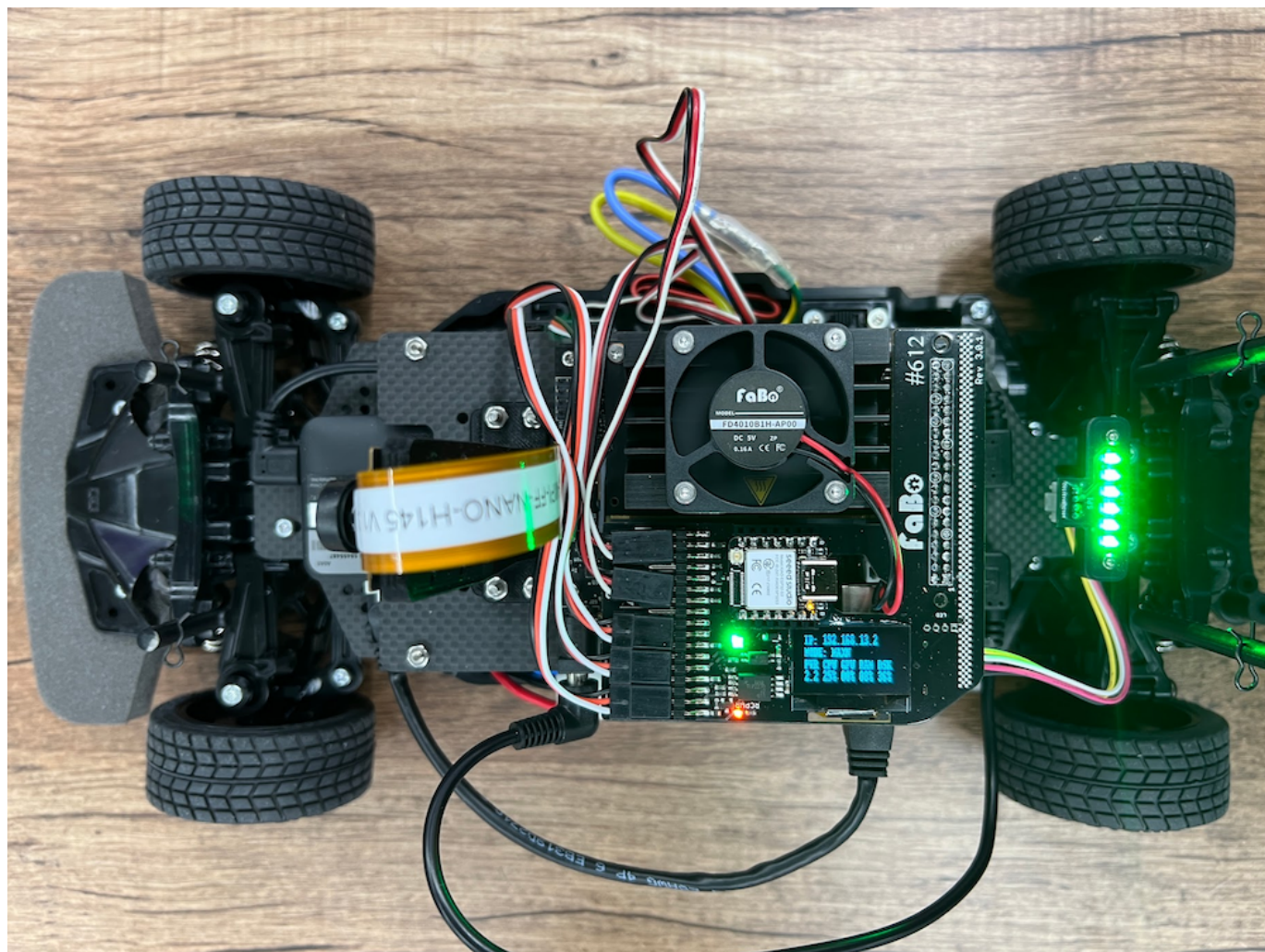
data_collection_widget = ipywidgets.VBox([
    separator,
    title1,
    task_widget,
    dataset_widget,
    annotation_box,
    ipywidgets.HBox([start_button, stop_button]),
    process_widget,
    separator,
    title2,
    release_button,
    process_widget
])

display(data_collection_widget)
```

【1.カメラの録画】
 データセット用の走行データを保存できます。走行データはcameraフォルダにデータが保存されます。Auto Annotationにチェックをいれると操作の値も保存できます。JetRacerでは、Auto Annotationでつけたデータセットの学習はうまくいきません。参考値を取得する用途でのみ使用してください。

task: camera
 dataset: aizu_data_101
☐ Auto Annotation
 Record Stop
 ログ

RCカーの状態は、緑色の点灯の状態で、RCカーをプロポで手動走行させます。



モードの切り替えは、プロポの後ろのボタンでおこなうことができます。配線等が問題なければ、ボタンを押す度に、緑色⇄ピンク色の色に切り替わります。



コースを5～10週程度、手動走行したら、[Stop]ボタンを押して録画を終了します。[Stop]ボタンを押すと、そこまでの録画枚数も表示されます。

Launcher 11_record_camera.ipynb Python 3

```
separator,
title1,
task_widget,
dataset_widget,
annotation_box,
ipywidgets.HBox([start_button, stop_button]),
process_widget,
separator,
title2,
release_button,
process_widget
])

display(data_collection_widget)
```

【1.カメラの録画】
データセット用の走行データを保存できます。走行データはcameraフォルダにデータが保存されます。Auto Annotationにチェックをいれると操作の値も保存できます。JetRacerでは、Auto Annotationでつけたデータセットの学習はうまくいきません。参考値を取得する用途でのみ使用してください。

task: camera
dataset: aizu_data_102
☐ Auto Annotation

Record Stop

ログ 29: 1191枚の画像を生成し、録画を終了しました。
28: FPS: 31.3 (3秒ごとに表示)
27: FPS: 30.3 (3秒ごとに表示)
26: FPS: 30.3 (3秒ごとに表示)
25: FPS: 30.3 (3秒ごとに表示)

【2.カメラの開放処理】
走行データの収集が完了したらカメラの開放処理を実行します。

[カメラ開放]ボタンを押して、カメラを停止します。

Launcher 11_record_camera.ipynb Python 3

データセット用の走行データを保存できます。走行データはcameraフォルダにデータが保存されます。Auto Annotationにチェックをいれると操作の値も保存できます。JetRacerでは、Auto Annotationでつけたデータセットの学習はうまくいきません。参考値を取得する用途でのみ使用してください。

task: camera
dataset: aizu_data_301
☐ Auto Annotation

Record Stop

ログ 15: 1203枚の画像を生成し、録画を終了しました。
14: FPS: 30.3 (3秒ごとに表示)
13: FPS: 30.3 (3秒ごとに表示)
12: FPS: 30.3 (3秒ごとに表示)
11: FPS: 30.3 (3秒ごとに表示)

【2.カメラの開放処理】
走行データの収集が完了したらカメラの開放処理を実行します。

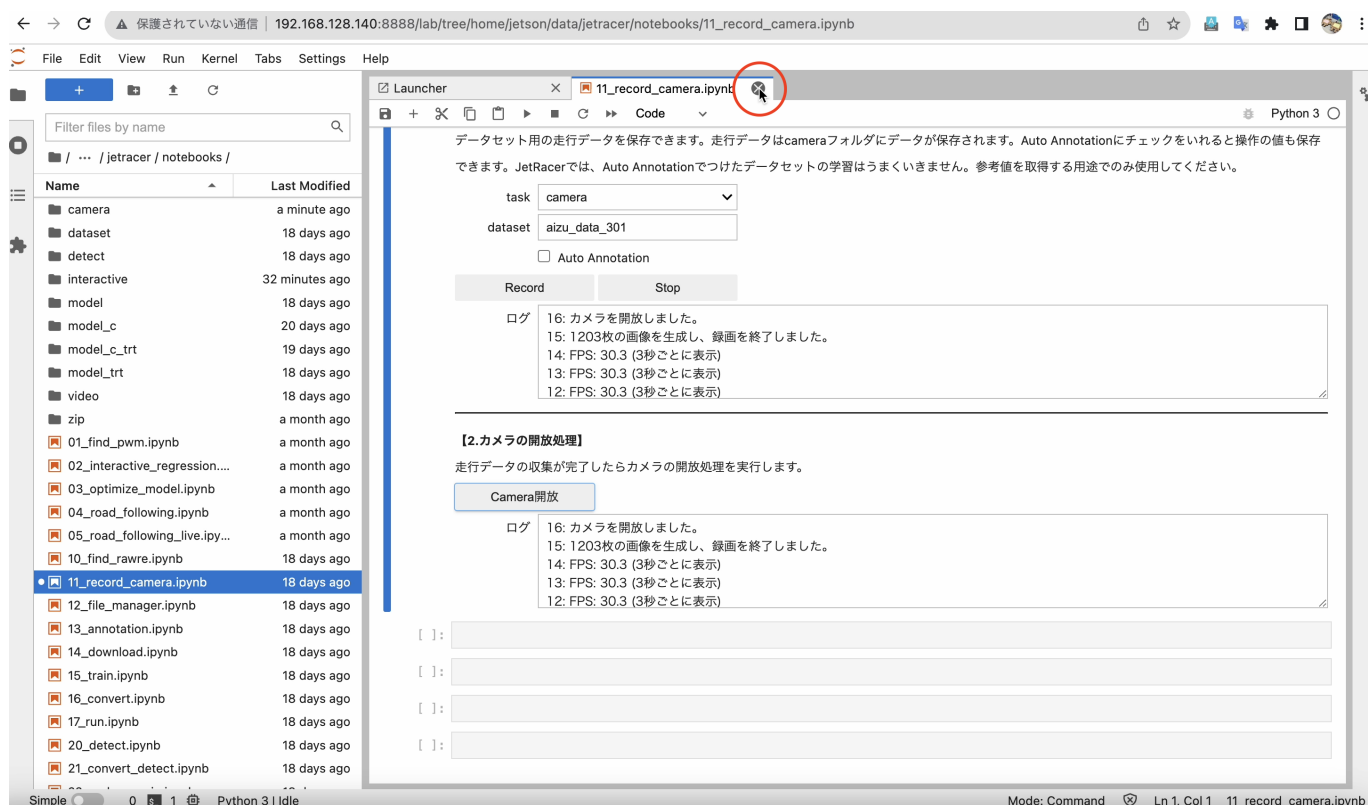
Camera開放

ログ 15: 1203枚の画像を生成し、録画を終了しました。
14: FPS: 30.3 (3秒ごとに表示)
13: FPS: 30.3 (3秒ごとに表示)
12: FPS: 30.3 (3秒ごとに表示)
11: FPS: 30.3 (3秒ごとに表示)

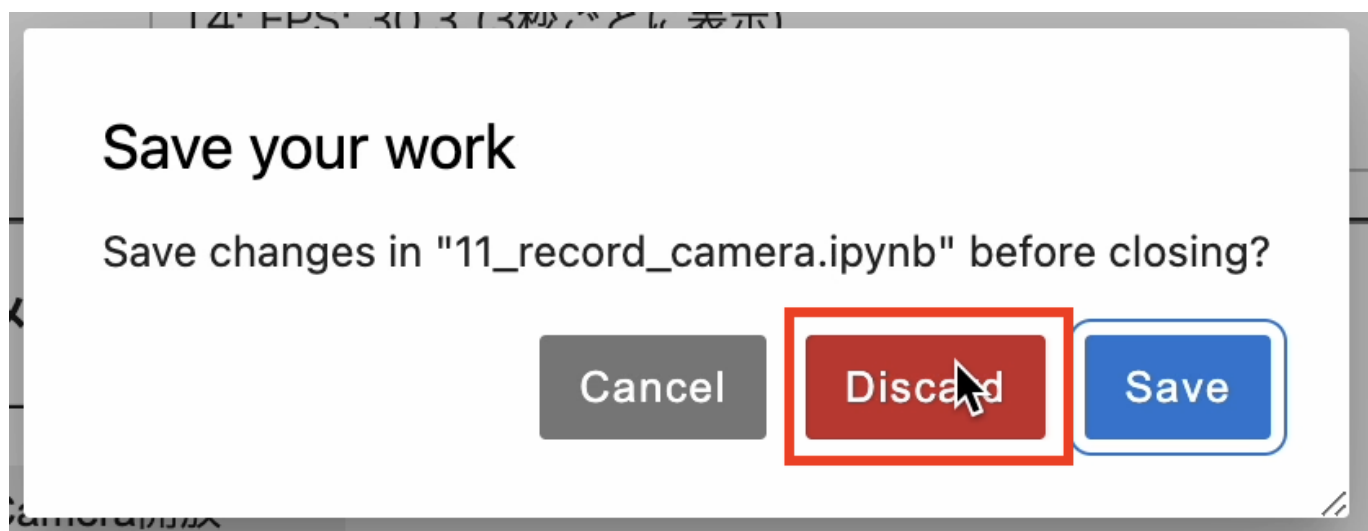


Jetsonでは、メモリ容量の制限のため、必ず処理が終了したNotebookはシャットダウンしてください。

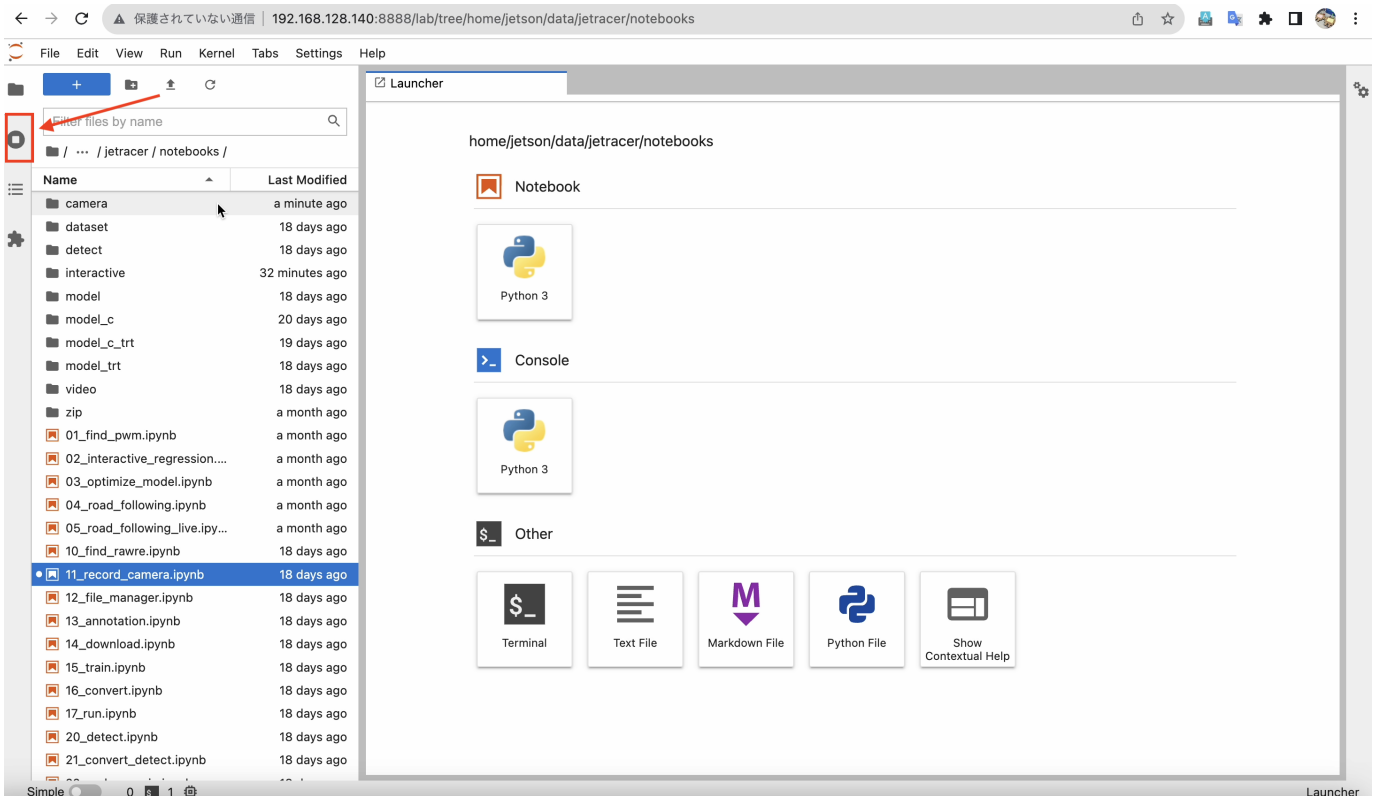
xボタンでNotebookを閉じます。



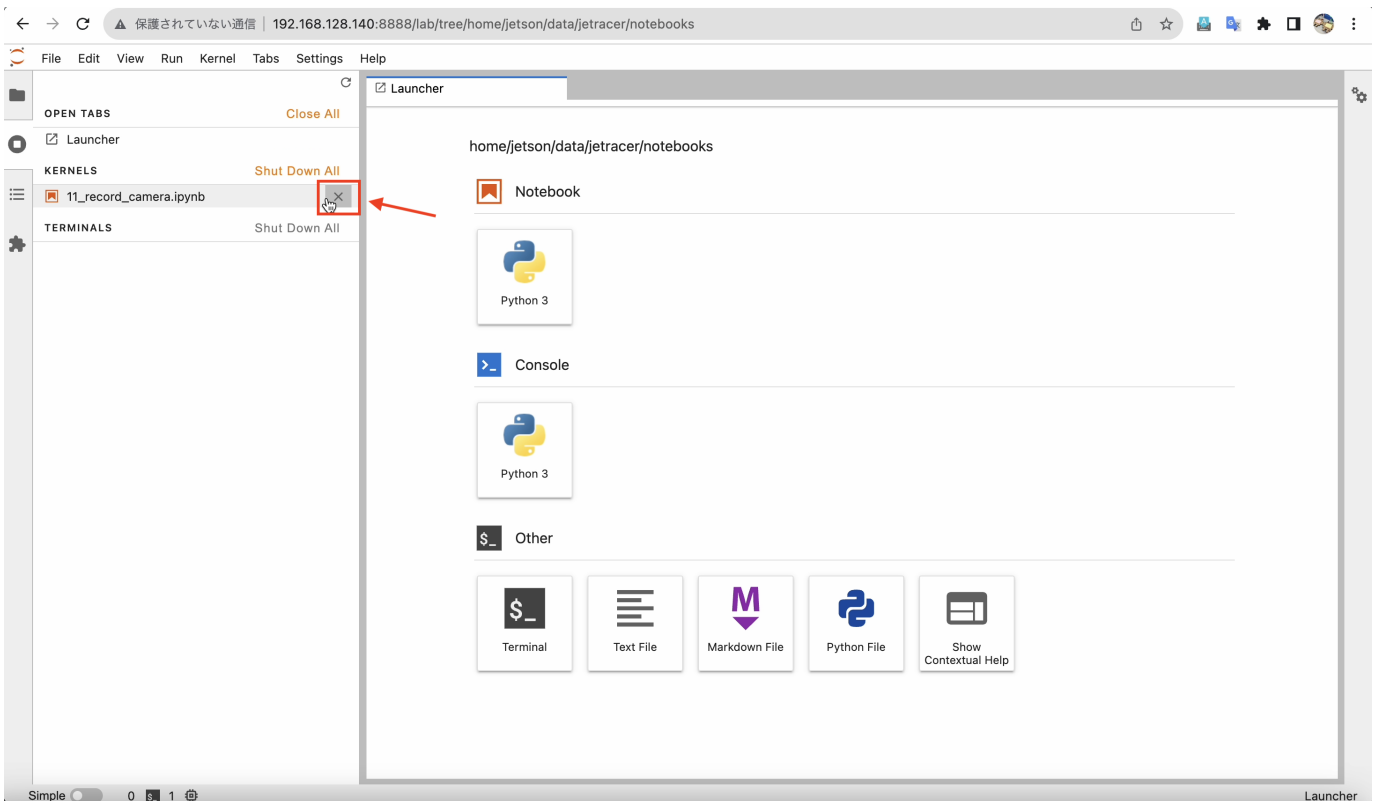
閉じる際に、保存するかを聞いてきますので、[Discard]を選択します。



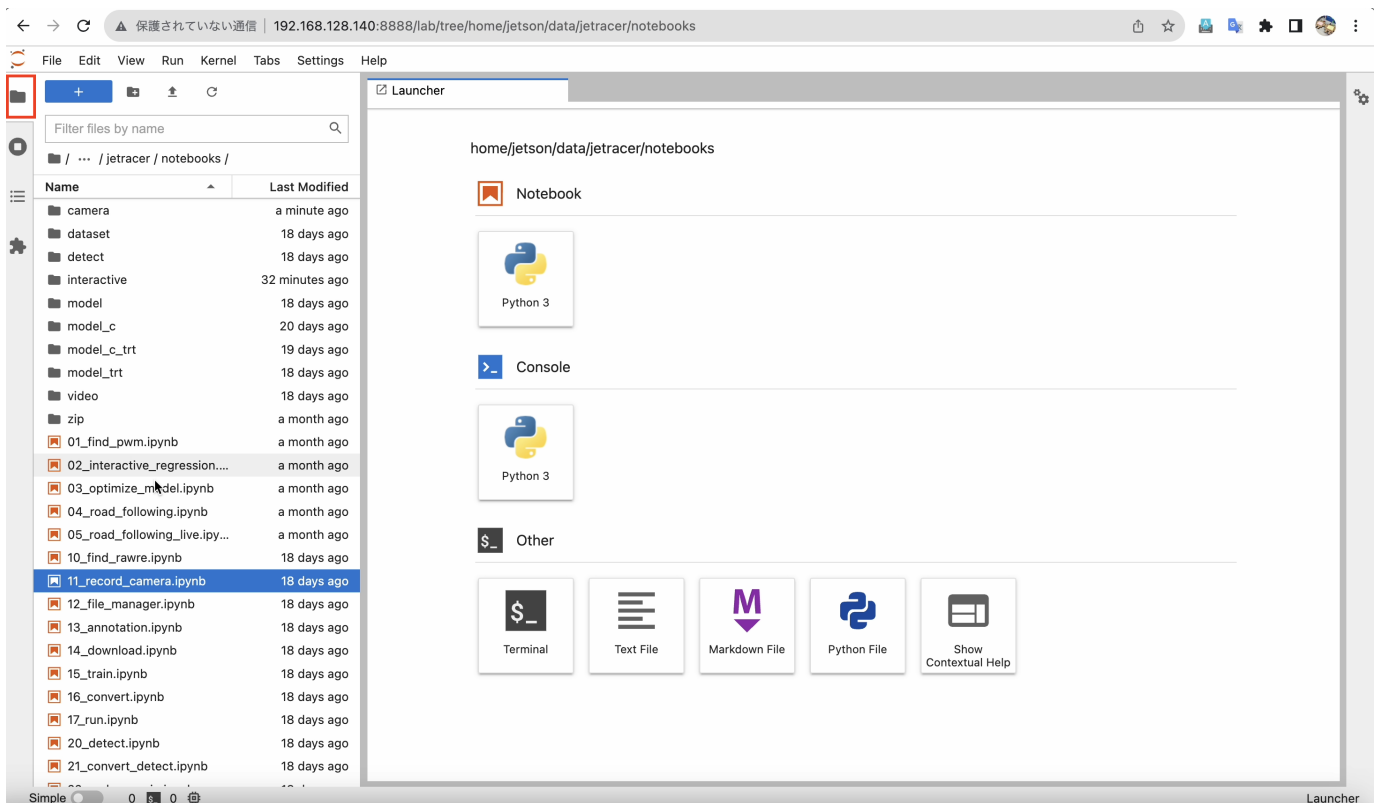
[プロセス]タブを選択します。



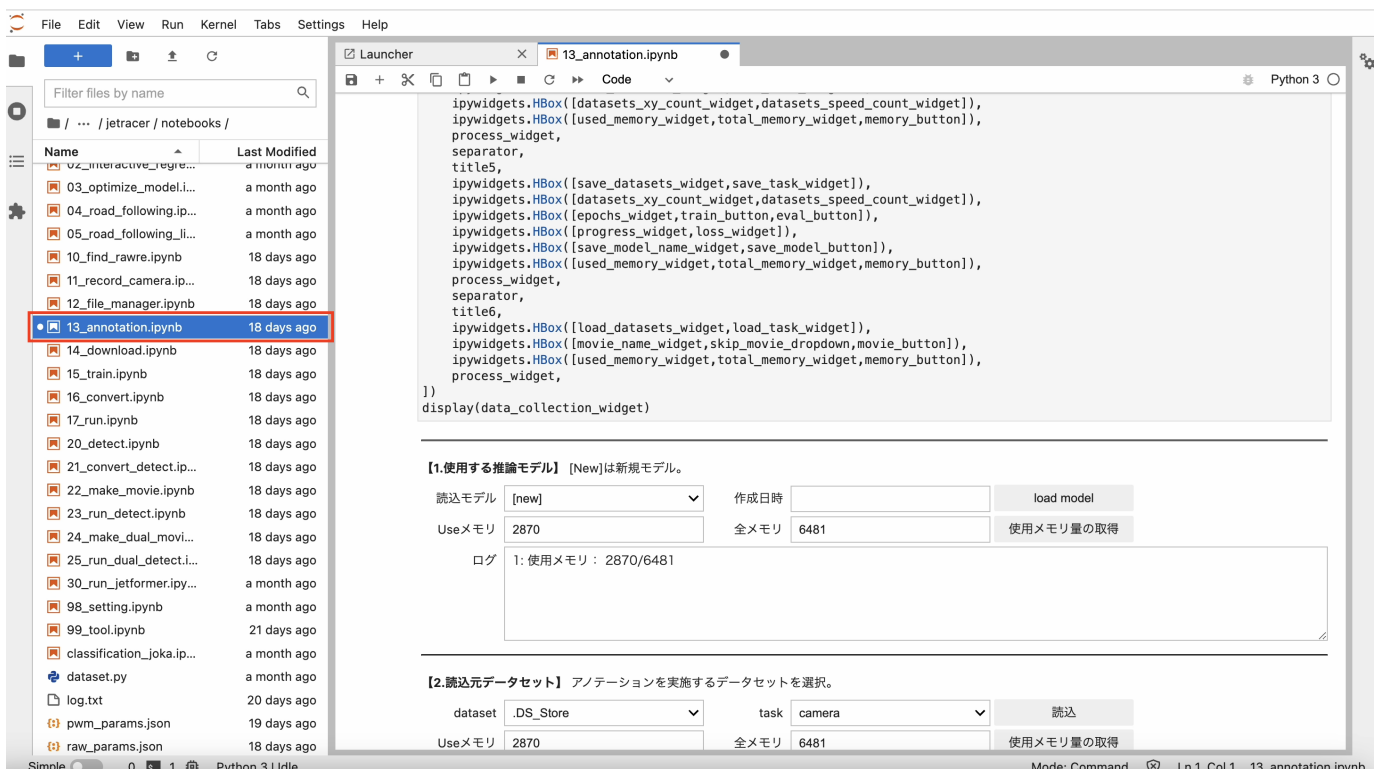
11_record_cameraを [shutdown] もしくは [x] ボタンをクリックし、Notebookのプロセスをshutdownします。この処理をおこなう事で、Notebookが専有していたメモリが開放されます。



[ファイル]タブを再び選択するとファイル一覧が表示されます。



【手順3】 13_annotation.ipynb



1.使用する推論モデル

新規にモデル作成する場合は、読み込モデルは[new]を選択してください。

【1.使用する推論モデル】 [New]は新規モデル。

読込モデル	[new] ▼	作成日時		load model
Useメモリ	2965	全メモリ	6481	使用メモリ量の取得

ログ 1: 使用メモリ : 2965/6481

2.読込元データセット

カメラ画像の録画で作成したデータセットを選択します。 taskはcamera, datasetはカメラ画像の録画時につけたデータセット名を選択します。

【2.読込元データセット】 アノテーションを実施するデータセットを選択。

dataset	aizu_data_102 ▼	task	camera ▼	読込
Useメモリ	2966	全メモリ	6481	使用メモリ量の取得

ログ 3: 使用メモリ : 2966/6481
2: ./model/model_aizu.pthのモデルを読込みました(torchvision 0.13以降)。
1: 使用メモリ : 2965/6481

[読込み]ボタンを押すとデータセットの読込みが開始されます。

【2.読込元データセット】 アノテーションを実施するデータセットを選択。

dataset	aizu_data_102 ▼	task	camera ▼	読込
Useメモリ	2966	全メモリ	6481	使用メモリ量の取得

ログ 5: 初回の読込みには時間がかかります。(30秒~1分) ←
4: データセット: /home/jetson/data/jetracer/notebooks/camera/aizu_data_301を読込みます。
3: 使用メモリ : 2966/6481
2: ./model/model_aizu.pthのモデルを読込みました(torchvision 0.13以降)。
1: 使用メモリ : 2965/6481

30秒から1分程度待ちます。下記のようなメッセージが表示されたら、読込み完了です。

【2.読込元データセット】 アノテーションを実施するデータセットを選択。

dataset	aizu_data_102 ▼	task	camera ▼	読込
Useメモリ	2966	全メモリ	6481	使用メモリ量の取得

ログ 6: 0枚目の/home/jetson/data/jetracer/notebooks/camera/aizu_data_102/xy/0_0_00000.jpgを読込みました。
5: 初回の読込みには時間がかかります。(30秒~1分)
4: データセット: /home/jetson/data/jetracer/notebooks/camera/aizu_data_301を読込みます。
3: 使用メモリ : 2966/6481
2: ./model/model_aizu.pthのモデルを読込みました(torchvision 0.13以降)。

3.保存先データセット

保存先データセットを新規に作成します。 dataset新規作成から任意の名前のデータセット名(例:aizu_dataset_102)を入力し、[Create dataset]ボタンを押します。

【3.保存先データセット】 データセットの保存先を選択。

dataset task

XYデータ数 速度データ数

datasetの新規作成 Name

Useメモリ 全メモリ

ログ
 7: 使用メモリ : 4313/6481
 6: 0枚目の/home/jetson/data/jetracer/notebooks/camera/aizu_data_301/xy/0_0_00000.jpgを読み込みました。
 5: 初回の読み込みには時間がかかります。(30秒~1分)
 4: データセット: /home/jetson/data/jetracer/notebooks/camera/aizu_data_301を読み込みます。
 3: 使用メモリ : 2966/6481

作成した名前のデータセットが設定されます。

【3.保存先データセット】 データセットの保存先を選択。

dataset task

XYデータ数 速度データ数

datasetの新規作成 Name

Useメモリ 全メモリ

ログ
 8: Datasetを作成しました : aizu_dataset_102
 7: 使用メモリ : 4313/6481
 6: 0枚目の/home/jetson/data/jetracer/notebooks/camera/aizu_data_301/xy/0_0_00000.jpgを読み込みました。
 5: 初回の読み込みには時間がかかります。(30秒~1分)
 4: データセット: /home/jetson/data/jetracer/notebooks/camera/aizu_data_301を読み込みます。

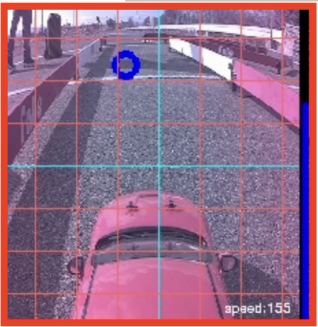
4. アノテーションの実施

アノテーションでは、車体を向かわせたい向きを、赤い太枠のエリアをクリックし、登録していきます。左折させたいとは、左方向をクリックして、データ登録していきます。青○はAIの推論結果で、緑○がユーザがクリックしたアノテーション結果となります。

【4.アノテーションの実施】 緑○がアノテーション, 青○がAIでの推論。車両の走らせたい場所で、画面をクリックすると保存先データセットのxyにデータが登録されます。Speedは[速度追加]で追加します。

no

skip(枚)

 speed

8個単位での処理

編集 編集 編集 編集 編集 編集 編集 編集

現在表示している画像の前後8枚の推論結果を表示したい場合は、<[-8],[8]> ボタンをクリックします。

【4.アノテーションの実施】 緑○がアノテーション, 青○がAIでの推論。車両の走らせたい場所で、画面をクリックすると保存先データセットのxyにデータが登録されます。Speedは[速度追加]で追加します。

no

skip(枚) ▼



speed

0

速度追加

▶

■

<

>

8個単位での処理

<[-8]

[8]>

編集

編集

編集

編集

編集

編集

編集

編集

dataset

aizu_dataset_101

task

dataset

XYデータ数

141

速度データ数

30

Useメモリ

4324

全メモリ

6481

使用メモリ量の取得

登録したデータ数は、XYデータ数, 速度データ数で確認できます。シンプルなコースの場合は、100枚程度のアノテーションを実施します。

【4.アノテーションの実施】 緑○がアノテーション, 青○がAIでの推論。車両の走らせたい場所で、画面をクリックすると保存先データセットのxyにデータが登録されます。Speedは[速度追加]で追加します。

no

skip(枚) ▼



speed

0

速度追加

▶

■

<

>

8個単位での処理

<[-8]

[8]>

編集

編集

編集

編集

編集

編集

編集

編集

dataset

aizu_dataset_101

task

dataset

XYデータ数

141

速度データ数

0

Useメモリ

4324

全メモリ

6481

使用メモリ量の取得

5.学習

登録したアノテーションデータを30 epoch学習させましょう。epochsの値を30に増やし、[train]ボタンをクリックし、学習を開始します。

【5.学習】 EPOCH指定で学習できます。

dataset	aizu_dataset_101	task	dataset
XYデータ数	39	速度データ数	0
epochs	30	train evaluate	
progress	0		
保存モデル名	model.pth	save model	
Useメモリ	4241	全メモリ	6481
	使用メモリ量の取得		
ログ	<pre> 147: [dataset/aizu_dataset_101]のxyカテゴリにデータを追加しました。 146: STOP 145: 450枚目の/home/jetson/data/jetracer/notebooks/camera/aizu_data_101/xy/0_0_00450.jpgを読込みました。 144: [dataset/aizu_dataset_101]のxyカテゴリにデータを追加しました。 143: STOP </pre>		

学習が開始されると、Progressのゲージが動き始めます。

【5.学習】 EPOCH指定で学習できます。

dataset	aizu_dataset_101	task	dataset
XYデータ数	39	速度データ数	0
epochs	30	train evaluate	
progress	0.16302440315485		
保存モデル名	model.pth	save model	
Useメモリ	4241	全メモリ	6481
	使用メモリ量の取得		
ログ	<pre> 152: ----- 151: XYデータ数: 39 Speedデータ数: 0 150: データセット: dataset/aizu_dataset_101 149: 学習を開始します。 148: ----- </pre>		

30 epochの学習が完了したら、4.アノテーションの実施に戻り、推論結果を確認します。

noを100ぐらいに戻し[▶]ボタンを押して、青○で表示される推論結果を確認します。認識がうまくいっていない箇所があったら、[■]ボタンを押し、画面をクリックし、アノテーションを追加して、データを補正します。

【4.アノテーションの実施】 緑○がアノテーション, 青○がAIでの推論。車両の走らせたい場所で、画面をクリックすると保存先データセットのxyにデータが登録されます。Speedは[速度追加]で追加します。

no

更新

削除

skip(枚)

▶

■

<

>

8個単位での処理

<[-8]

[8]>

速度追加

編集

編集

編集

編集

編集

編集

編集

編集

dataset

aizu_dataset_101

task

dataset

XYデータ数
速度データ数

Useメモリ
全メモリ

使用メモリ量の取得

4.アノテーションの実施⇒5.学習を相互に行き来しながら、繰り返し、アノテーションデータを作成していきます。満足のいく推論結果になったら、[save model]でモデルデータを model.pth の名前で保存します。

【5.学習】 EPOCH指定で学習できます。

dataset

aizu_dataset_101

task

dataset

XYデータ数
速度データ数

epochs

train

evaluate

progress
loss

保存モデル名

model.pth

save model

Useメモリ
全メモリ

使用メモリ量の取得

ログ

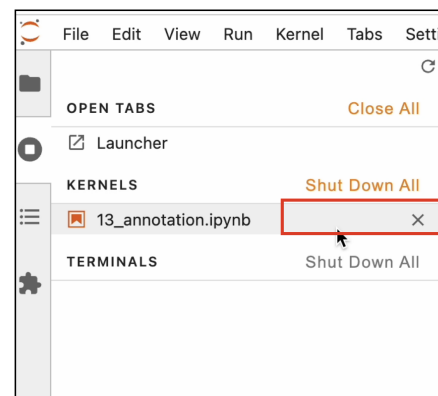
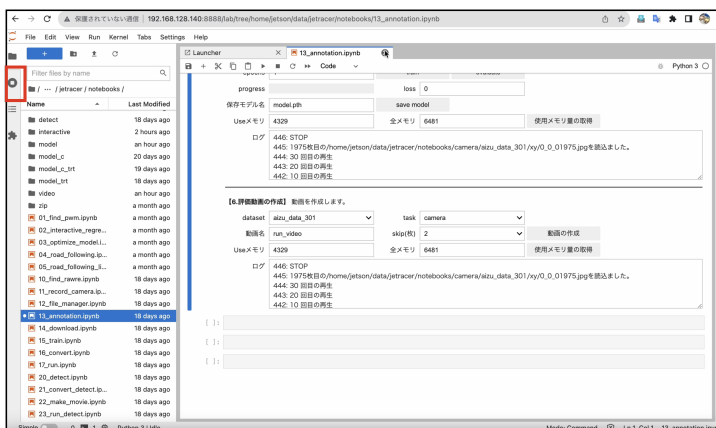
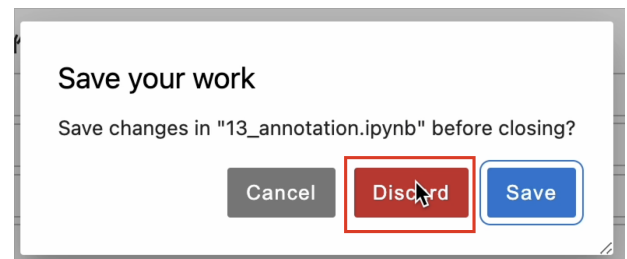
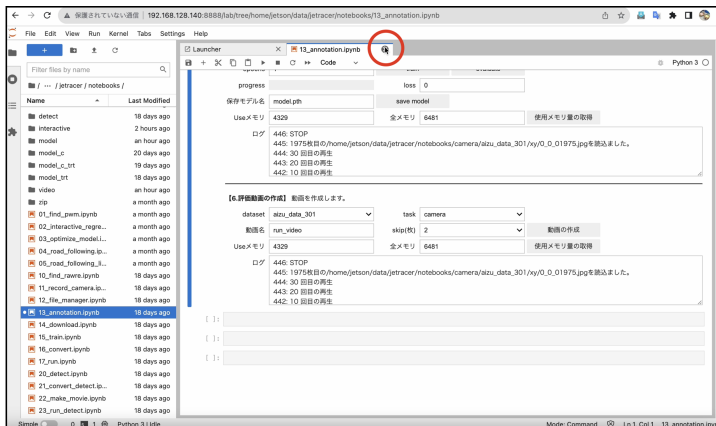
```

311: 30 Epoch目: 3.44秒
310: 新しいベストモデルが保存されました。Epoch loss: 0.0011
309: 29 Epoch目: 3.45秒
308: 28 Epoch目: 3.42秒
307: 27 Epoch目: 3.44秒

```

Info

Jetsonでは、メモリ容量の制限のため、必ず処理が終了したNotebookはシャットダウンしてください。



【手順3】 16_convert.ipynb

16_convert.ipynbでは、保存したモデルをTensorRTのフォーマットに変換し、軽量化します。前項で保存した model.pth を result.pth という名称で、TensorRTフォーマットで変換します。

【1.TensorRTへの変換】 軽量化、高速化を目的にモデルをTensorRT形式に変換します。

変換前

./model/model.pth

▼

変換後

result.pth

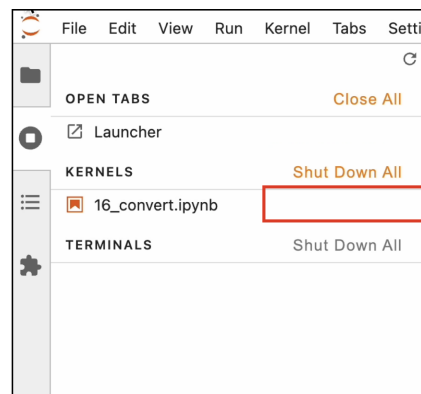
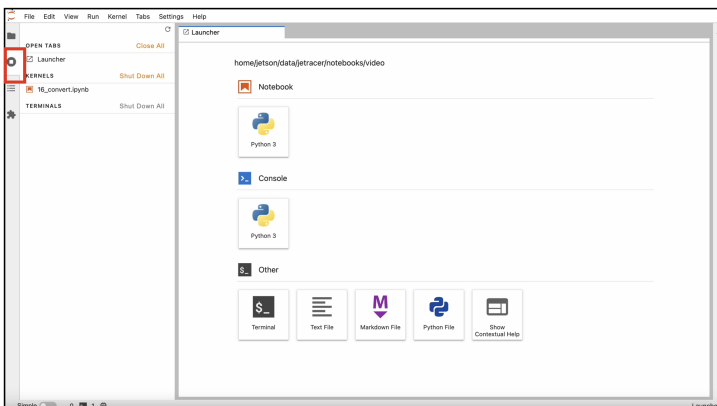
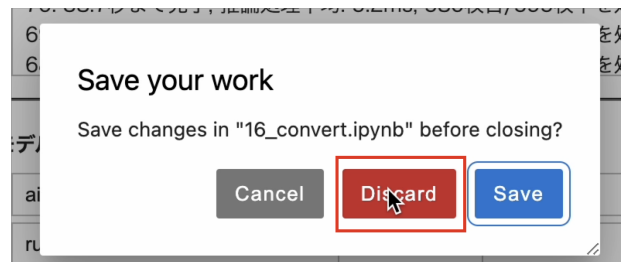
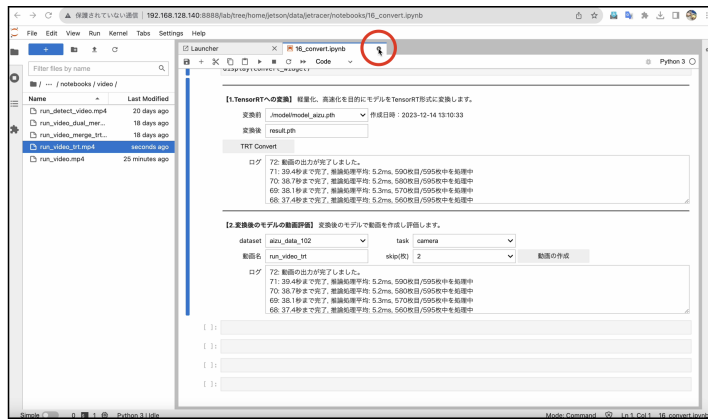
作成日時：2023-11-24 11:49:04

TRT Convert

ログ



Jetsonでは、メモリ容量の制限のため、必ず処理が終了したNotebookはシャットダウンしてください。



【手順4】 17_run.ipynb

17_run.ipynbで、自動走行をおこないます。

【1.使用する推論モデル】
TensorRTに変換済みのモデルをLoadします。

モデル

./model_trt/result.pth

▼

2023-11-26 11:17:06

TRTモデルを読み込み

ログ

Steeringゲインを設定します。値を大きくすると、より大きく曲がるようになります。

【2.Steeringゲイン】
Steeringのゲイン調整します。周りが悪い時は値を1.0以上にします。

Steering gain

1.70

Speedは、固定値を選びます。値を大きくすると、走行速度が速くなります。

【3.速度】
速度は固定値か、推論から反映かが選べます。

Speed

固定値

▼

Speed raw

80

[走行開始]ボタンを押して、走行を開始します。

【5.走行の開始】 走行開始を押す前にタイヤが空転していのを確認してください。プロポの裏のボタンを変換しAIモードにすると動き始めます。

Useメモリ 全メモリ 使用メモリ量の取得

走行開始 停止

ログ

```
2: ./model_trt/result.pthの読み込に成功しました。
1: ./model_trt/result.pthの読み込を実行します(初回は時間がかかります)。
```

AIが動くと、下記のようなメッセージが表示されます。

【5.走行の開始】 走行開始を押す前にタイヤが空転していのを確認してください。プロポの裏のボタンを変換しAIモードにすると動き始めます。

Useメモリ 全メモリ 使用メモリ量の取得

走行開始 停止

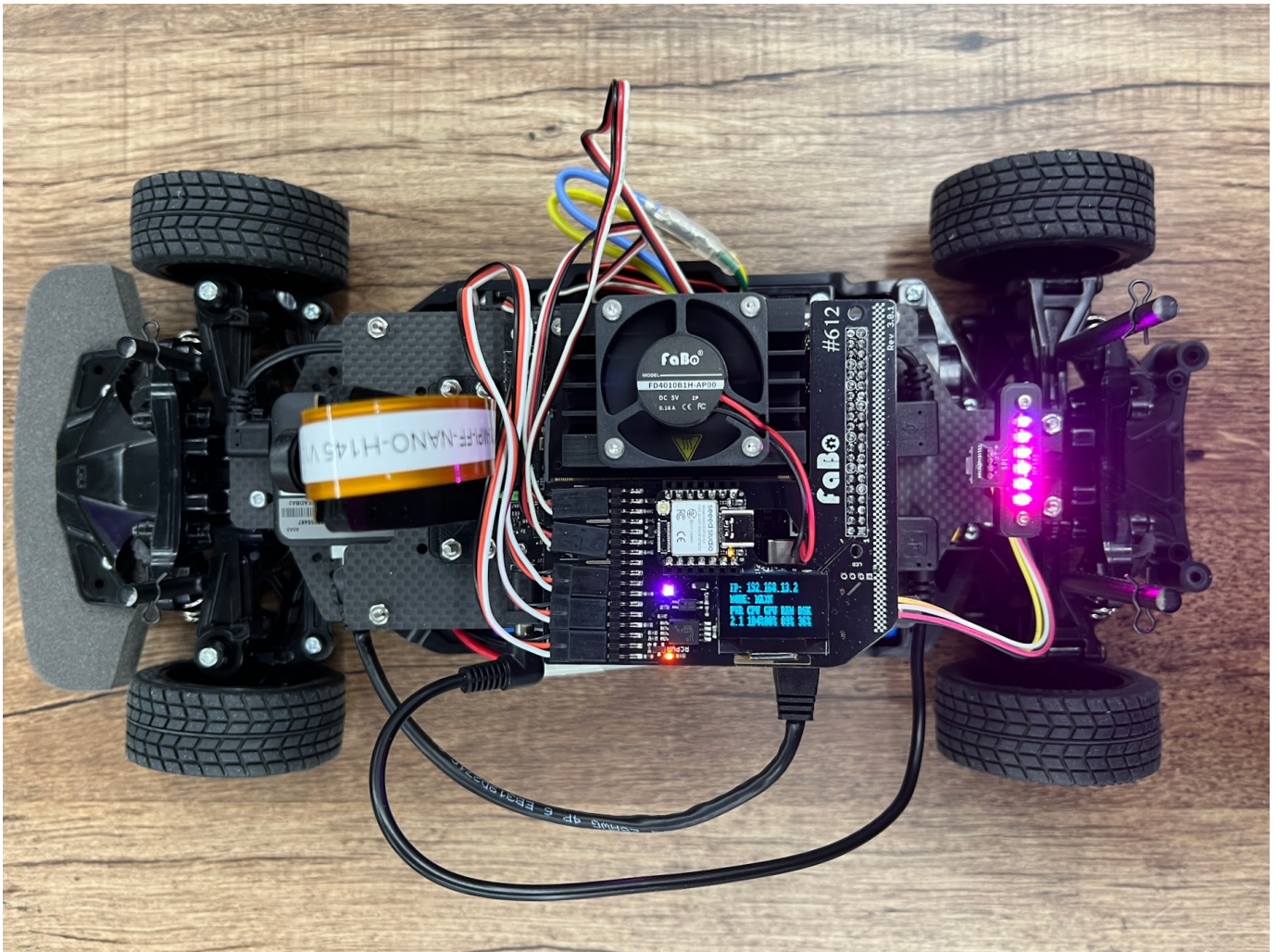
ログ

```
5: 推論処理: 8.7ms, Speed: 224.0 (推論), Steering Gain: 1.7, FPS: 64.0 (3秒ごとに表示)
4: AIが起動しました。
3: camera/run_aizu_301/xy/にデータを保存します。
2: ./model_trt/result.pthの読み込に成功しました。
1: ./model_trt/result.pthの読み込を実行します(初回は時間がかかります)。
```

プロポの裏のボタンを押して、手動走行モードから自動走行モードに変更します。Speedで設定した速度で走り始めます。



AIモードでの走行は、LEDがピンク色に点灯します。



3.2 項目別

3.2.1 Notebooks

使用するNotebooks

NOTEBOOKの場所

`/home/jetson/jetracer/notebooks` にあります。

FaBoの付属のSDカードご使用ならばソフトウェアがインストールと設定済みで、すぐにJetRacerを使うことができます。

AI86 Version2

最新版入手先：[AI86 Version2のSDカードイメージダウンロード](#) [AI86 Version2.0.3](#)

Google Colaboratoryによるクラウドでの学習を追加しました。Jetson Nanoでは難しかった大きなデータセットの学習にも対応させております。

オートアノテーション機能追加

画像カット編集機能追加

環境情報判断追加 2つのカメラを利用してターゲットとなる環境であると画像で判断したらモデルを入れ替えることができます。（周回数をカウントなどに利用）※Jetson Orin Nano対象



フロー

順番に従って実行するとスムーズに行うことができます。

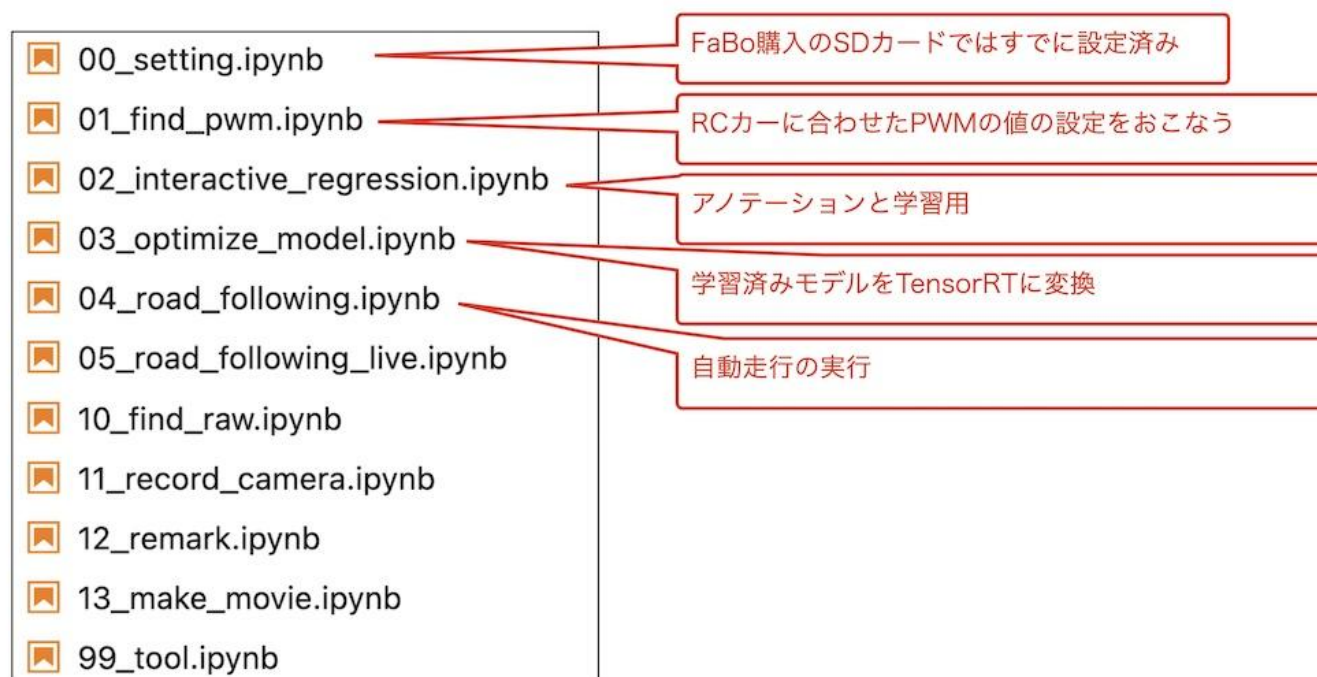
-  01_find_pwm.ipynb
-  02_interactive_regression.ipynb
-  03_optimize_model.ipynb
-  04_road_following.ipynb
-  05_road_following_live.ipynb
-  10_find_raw.ipynb
-  11_record_camera.ipynb
-  12_file_manager.ipynb
-  13_annotation.ipynb
-  14_download.ipynb
-  15_train.ipynb
-  16_convert.ipynb
-  17_run.ipynb
-  20_detect.ipynb
-  21_convert_detect.ipynb
-  22_make_movie.ipynb
-  23_run_detect.ipynb
-  24_make_dual_movie.ipynb
-  25_run_dual_detect.ipynb
-  30_run_jetformer.ipynb
-  98_setting.ipynb
-  99_tool.ipynb

コード番号	Jetsonボード
01,10~17	Jetson Nano Developer Kit B-01,A-02,Jetson Orin Nano
20~25	Jetson Orin Nano

順番	notebook	内容
1	01_find_pwm.ipynb	RCカー用にPWMの値を設定
-	02_interactive_regression.ipynb	アノテーションと学習 AI86 Version1
-	03_optimize_model.ipynb	軽量化モデル(TensorRT)への変換 AI86 Version1
-	04_road_following.ipynb	自動走行 AI86 Version1
-	05_road_following_live.ipynb	自動走行（ライブ） AI86 Version1
2	10_find_raw.ipynb	PWMの値の吸い上げ 受信機信号計測
3	11_record_camera.ipynb	走行の録画(データセット用のデータの作成)
4	12_file_manager.ipynb	ファイルマネージャー 画像カットなど
5	13_annotation.ipynb	アノテーション 教師データ入力
6	14_download.ipynb	ダウンロード GoogleColaboratory使用のための複数のファイルをZip化、圧縮および解凍
7	15_train.ipynb	学習 GoogleColaboratoryへのリンクおよびJetsonでの学習し動画で検証
8	16_convert.ipynb	TensorRTへの変換と動画で検証
9	17_run.ipynb	自動走行、走行録画
10	20_detect.ipynb	環境情報判断 コース場所を検知
11	21_convert_detect.ipynb	環境情報判断 TensorRTへの変換
12	22_make_movie.ipynb	環境情報判断 評価用動画の作成
13	23_run_detect.ipynb	自動走行と環境情報判断 走行検証
14	24_make_dual_movie.ipynb	自動走行と環境情報判断 動画検証
15	25_run_dual_detect.ipynb	自動走行と環境情報判断 条件により走行モデル切替
16	30_run_jetformer.ipynb	-----
-	98_setting.ipynb	JetRacerに必要なライブラリのインストール（実行済み）
-	99_tool.ipynb	Wi-Fi設定、I2Cデバイスアドレス確認、カメラ認識の確認、電力モード確認、コードのリカバリー

AI86 Version1





フロー

順番	notebook	内容
1	00_setting.ipynb	パッケージ等の設定, FaBoストア購入車体ではすでに設定済み
2	01_find_pwm.ipynb	RCカー用にPWMの値を設定
3	02_interactive_regression.ipynb	アノテーションと学習
4	03_optimize_model.ipynb	軽量化モデル(TensorRT)への変換
5	04_road_following.ipynb	自動走行

3.2.2 PWMの設定

JetRacerの車体や電子基板には個体差があります。ここではその調整と設定を行います。

解説動画



設定後、pwm_params.jsonファイルができます。

3.2.3 走行録画



RCカーを実際走行させて録画します。録画したデータを一枚一枚アノテーションして教師データを作成し学習させます。※オプションでプロポ信号を計測し自動でデータ収集する自動でアノテーションをする機能があります。



オートアノテーション機能は、コントロール基板 #612 JetRacer Rev 2.0.9以降の基板のみ対応



オートアノテーションは、使用目的によりお使いください。教材としては初めは録画のみとしデータを収集しないことがお勧めです。

11_record_camera.ipynbを開きます。今回は、30fpsで録画します。

The screenshot shows the JupyterLab interface. On the left, the file explorer displays a list of files and folders. The file '11_record_camera.ipynb' is highlighted with a red arrow. On the right, the code editor shows the following code:

```

[1]: import Jetson.GPIO as GPIO

BOARD_NAME=GPIO.gpio_pin_data.get_data()[0]
if BOARD_NAME == "JETSON_NX":
    print("Jetson Xavier NXを認識")
    I2C_BUSNUM = 8
elif BOARD_NAME == "JETSON_XAVIER":
    print("Jetson AGX Xavierを認識")
    I2C_BUSNUM = 8
elif BOARD_NAME == "JETSON_NANO":
    print("Jetson NANOを認識")
    I2C_BUSNUM = 1

Jetson NANOを認識

[2]: from jetcam.csi_camera import CSICamera
    from jetcam.utils import bgr8_to_jpeg

    camera = CSICamera(width=224, height=224, capture_fps=30)

[3]: camera.running = False

[4]: import ipywidgets
    from ipywidgets import Button, Layout, Textarea, HBox, VBox, Label
    import subprocess
  
```



AI86 version1のお使いの方は、第5セル 5 1行目のtime.sleep(0.100)を削除するか、先頭に#をつけて、コメントアウトしてください。

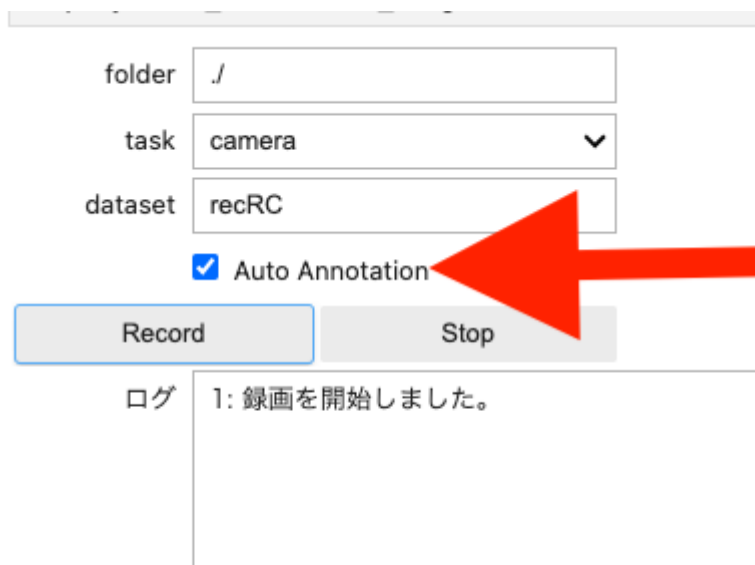
Warning

オートアノテーションを実行する場合は、10_find_raw.ipynbで事前にプロポ信号の計測が必要になります。

切り替え信号をRCモードにしてプロポを使って任意の時間、走行します。オートアノテーションを実行する場合は、Auto Anotationにチェックを付けます。チェックを付けない場合は、アノテーションのデータは付加されず後でアノテーションを手作業でします。(AI86 version 1は、12_remark.ipynb,AI86 version2は、13_annotation.ipynb)

folderは./で/home/jetson/jetracer/notebooks/

taskはデフォルトではcameraになります。データセットは任意の名前をつけます。Recordボタンをクリックすると録画が開始されます。



folder ./

task camera ▼

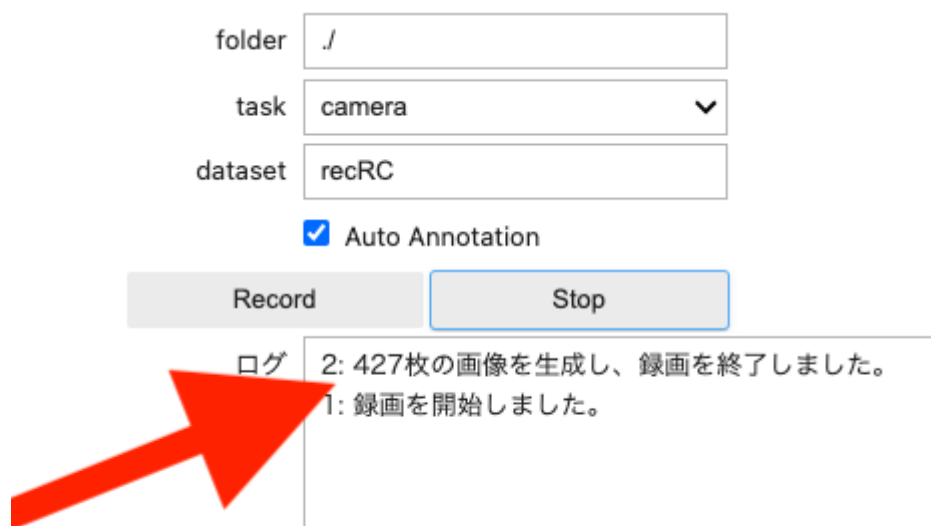
dataset recRC

☒ Auto Annotation

Record Stop

ログ 1: 録画を開始しました。

ストップボタンを押すと、録画が停止されます。撮影された枚数が表示されます。



folder ./

task camera ▼

dataset recRC

☒ Auto Annotation

Record Stop

ログ 2: 427枚の画像を生成し、録画を終了しました。
1: 録画を開始しました。

データがcameraフォルダへ蓄積されます。

camera	3 hours ago
interactiv	21 hours ago
model	3 hours ago
model_trt	4 hours ago
zip	21 hours ago
01_find_pwm.ipynb	21 hours ago
02_interactive_regression.ipynb	4 hours ago
03_optimize_model.ipynb	4 hours ago
04_road_following.ipynb	21 hours ago
05_road_following_live.ipynb	21 hours ago
10_find_raw.ipynb	3 hours ago
11_record_camera.ipynb	3 hours ago
12_download_data.ipynb	21 hours ago

保存したファイルはステアリングのフォルダ(xy)とスロットルのフォルダ(speed)にそれぞれ保存されます。

ファイル名の一部がアノテーションデータになります。

Name		Last Modified
104_112_00146.jpg		3 hours ago
107_112_00058.jpg		3 hours ago
107_112_00235.jpg		3 hours ago
108_112_00056.jpg		3 hours ago
108_112_00057.jpg		3 hours ago
108_112_00079.jpg		3 hours ago
108_112_00145.jpg		3 hours ago
108_112_00182.jpg		3 hours ago
108_112_00233.jpg		3 hours ago
108_112_00275.jpg		3 hours ago
108_112_00276.jpg		3 hours ago
108_112_00312.jpg		3 hours ago
108_112_00367.jpg		3 hours ago

X座標データ、Y座標データ、連番.jpg ファイル

撮影が終了し、タブを閉じる場合はカメラを開放して、カーネルも終了してください。

カメラの終了処理(必須)

```
[9]: import time
      camera.running = False
      time.sleep(1)
      camera.cap.release()
```

```
[ ]:
```



AI86 Version2の場合は、Camera開放ボタンを押します。

【2.カメラの開放処理】

走行データの収集が完了したらカメラの開放処理を実行します。

Camera開放

ログ 1: カメラを開放しました。

Warning

カメラの終了処理を実行しないで終了すると、カメラが再起動できずJetsonNanoごと再起動が必要となります。

Warning

画像ファイルが壊れている場合があります。findコマンドでファイルが壊れていないかチェックが必要な場合がございます。

空のファイルを見つける

```
1 sudo find ./ -empty
```

空のファイルの削除する。

```
1 sudo find ./ -empty -delete
```



レコードライン（理想的な走行ライン）と他に少しコースアウトした画像も録画しましょう。

3.2.4 画像カット



録画した走行画像にはアノテーション作業に録画開始時など不要な画像やノイズが含まれますのでアノテーション作業の前に画像をカットします。

開始

12_file_manager.ipynbを実行しセルを順番に上から実行して、shift + Enterキーまたは、▶ボタンを押していきます。

05_road_following_live.ipynb	21 hours ago
10_find_raw.ipynb	21 hours ago
11_record_camera.ipynb	4 days ago
12_file_manager.ipynb	a minute ago
13_annotation.ipynb	17 minutes ago
14_download.ipynb	a day ago
15_train.ipynb	a day ago
16_convert_ipynb	a day ago

不要データ削除

録画したデータセットを読み込みボタンを押して呼び出します。すると0番からの10枚の画像が列挙されて表示されます。左（戻る）右（進む）のボタンを押して不要な画像の範囲番号を入力して削除ボタンを押して画像を削除します。

■ 不要データの削除

【削除開始位置】，【削除終了位置】を指定して【削除ボタン】で不要データを削除します。

読込dataset: b 読込task: camera 読込

< >

画像枚数: 881

開始位置: 0 更新 終了位置: 10

削除開始位置: 0 削除終了位置: 10 削除

切り出し

ある範囲のデータだけを取り出してフォルダ名をつけることができます。

■ データセットの切出

【切出開始位置】，【切出終了位置】を指定して、【指定区間を切出し】ボタンで新しいデータセットに切出します。

切出dataset	samples	切出task	camera	指定区間を切出し
切出開始位置	0	切出終了位置	10	

ログ

```
3: 0から10までを切出しました。
2: xyの1枚目/618枚中 切出中
```

データセット削除

不要になったデータセットのフォルダごと削除できます。

■ データセットの削除

選択したデータセットを削除します。

読込dataset	run001	読込task	camera	全削除
-----------	-------------------------------------	--------	-------------------------------------	------------------------------------

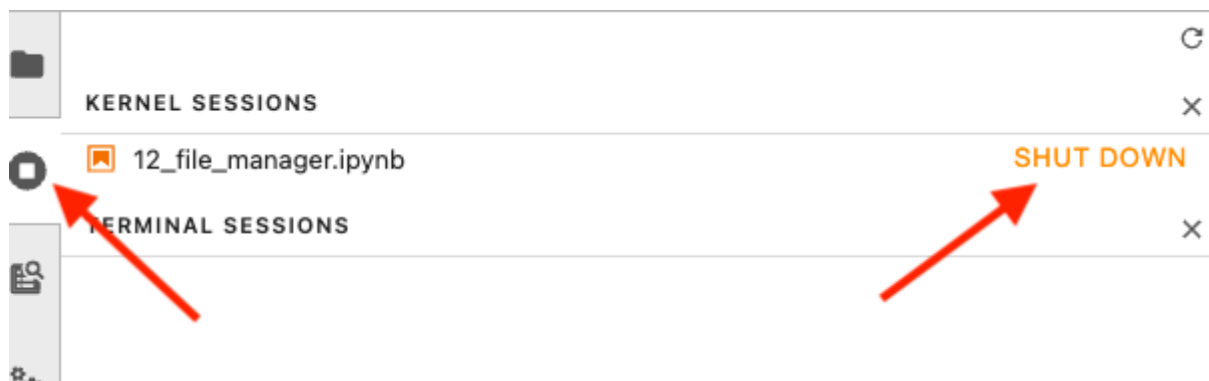
ログ

終了

タブを閉じます。



シャットダウンします。



3.2.5 アノテーションと学習方法

解説動画



Tips

走行ラインを修正する画像を追加するとうまく走ります。

Tips

推論している状態で、走行ラインを確認し上手くできない場所を追加学習行い狙ったラインに走行するように調整すると、さらによく走ります。



AI86 Version2

11_record_camera.ipynbで走行録画して、その走行画像を12_file_manager.ipynbでアノテーションしやすいように不要画像カットします。

次に13_annotaion.ipynbを実行します。

Danger

1コマずつ実際のステアリングやスロットルの動きを実機のRCカーで再現しますRCのAIモードにすると動き出すので作業台に置いて必ず実行してください。使用しないときはRCカーの電源をOFFにするか、RCモードでご使用ください。

	10_find_raw.ipynb	21 hours ago
	11_record_camera.ipynb	4 days ago
	12_file_manager.ipynb	4 days ago
	13_annotaion.ipynb	a day ago
	14_download.ipynb	a day ago
	15_train.ipynb	a day ago
	16_convert.ipynb	a day ago
	17_run.ipynb	43 minutes ago
	20_detect.ipynb	a day ago

セルを順番に上から実行して、shift + Enterキーまたは、 ボタンを押していきます。しばらくすると下のような画面が現れます。

作成した走行モデルがない場合は、[new]でload modelボタンを押します。

【1.使用する推論モデル】 [New]は新規モデル。

読みモデル	[new]	作成日時		load model
Useメモリ	2757	全メモリ	3964	使用メモリ量の取得
ログ	1: 使用メモリ : 2757/3964			

すでに作成した走行モデルがある場合は読み込みモデルを選択してload modelボタンを押して読み込みます。



既存のモデルを使用しないで特定のデータセット内の画像を学習したいときはnewを選択します。

【1.使用する推論モデル】 [New]は新規モデル。

読みモデル	/model/model202402061601.pt	作成日時	2024-02-06 16:01:12	load model
Useメモリ	2757	全メモリ	3964	使用メモリ量の取得
ログ	1: 使用メモリ : 2757/3964			

読み込みのデータセットを選択して読み込みボタンを押します。走行録画はcameraフォルダまたはrunフォルダにあります。使用メモリはJetsonのRAMメモリです。画像データが多いと不安定になったり止まったりすることがあります。使用メモリの取得ボタンを押して、メモリをチェックしましょう。

【2.読み元データセット】 アノテーションを実施するデータセットを選択。

dataset	run001	task	camera	読み
Useメモリ	2757	全メモリ	3964	使用メモリ量の取得
ログ	4: 0枚目の/home/jetson/jetracer/notebooks/camera/run001/xy/0_00001.jpgを読み込みました。 3: 初回の読み込みには時間がかかります。(30秒~1分) 2: データセット: /home/jetson/jetracer/notebooks/camera/run001を読み込みます。 1: 使用メモリ : 2757/3964			

新規に保存するデータセットフォルダを作る場合は、Nameにデータセット名をつけてcreate datasetボタンを押して新規フォルダを選択します。(既存にあるデータセットを選択する場合は、taskでフォルダを選んでdatasetを選択します。)

【3.保存先データセット】 データセットの保存先を選択。

dataset	02091709	task	dataset
XYデータ数	0	速度データ数	0
datasetの新規作成		Name	02091709
		<button>Create dataset</button>	
Useメモリ	3425	全メモリ	3964
		<button>使用メモリ量の取得</button>	

画像1枚1枚に対して教師データを追加していきます。初めに更新ボタンを押して画像を読み出します。画像の○はステアリング値、画像の縦棒はスロットル値を表します。

【4.アノテーションの実施】 緑○がアノテーション、青○がAIでの推論。車両の走らせたい場所で、画面をクリックすると保存先データセットのxyに追加します。

画像番号 更新 削除

skip(枚) 画像再生 画像停止

画像をクリックしてステアリング値の教師データを追加する。

速度 画像Skipのコマ数戻る < 画像Skipのコマ数進む >

スロット値 8個単位での処理 <[-8] [8]>

スロット値教師データ追加 速度追加

推論結果を8枚ずつ列挙する

縦横224ドット

データセットはdatasetフォルダに追加される

dataset 02061452 task dataset

XYデータ数 322 速度データ数 187

Useメモリ 3382 全メモリ 3964 使用メモリ量の取得

ログ

```

68: 76枚目の/home/jetson/jetracer/notebooks/camera/b/xy/72_112_00156.jpgを読込みました。
67: 74枚目の/home/jetson/jetracer/notebooks/camera/b/xy/168_112_00154.jpgを読込みました。
66: 76枚目の/home/jetson/jetracer/notebooks/camera/b/xy/72_112_00156.jpgを読込みました。
65: 78枚目の/home/jetson/jetracer/notebooks/camera/b/xy/0_112_00158.jpgを読込みました。
64: 80枚目の/home/jetson/jetracer/notebooks/camera/b/xv/0_112_00160.jpgを読込みました。

```

Info

画像をクリックするとステアリングの値が入力されdatasetフォルダにあるxyフォルダに蓄積されます。画像のY軸の値は学習には使用しておりません。ご注意ください。なお、スロットル値のデータセットは縦のスライダーで速度追加ボタンを押すと、datasetフォルダspeedフォルダに蓄積されます。



8個単位で表示するボタンを使用すると推論が失敗している画像が見つけやすくなります。画像上の編集ボタンを押すとその画像にジャンプできます。(モデルを仕上げる時に有効です。)

エポック数を入力します。保存モデル名を入力してtrainボタンを押します。しばらくするとlossが表示されます。ephocsがゼロになると学習終了です。終了しましたら、モデル名をつけてsave_modelをボタンを押します。できたモデルはmodelフォルダにあります。

【5.学習】 EPOCH指定で学習できます。

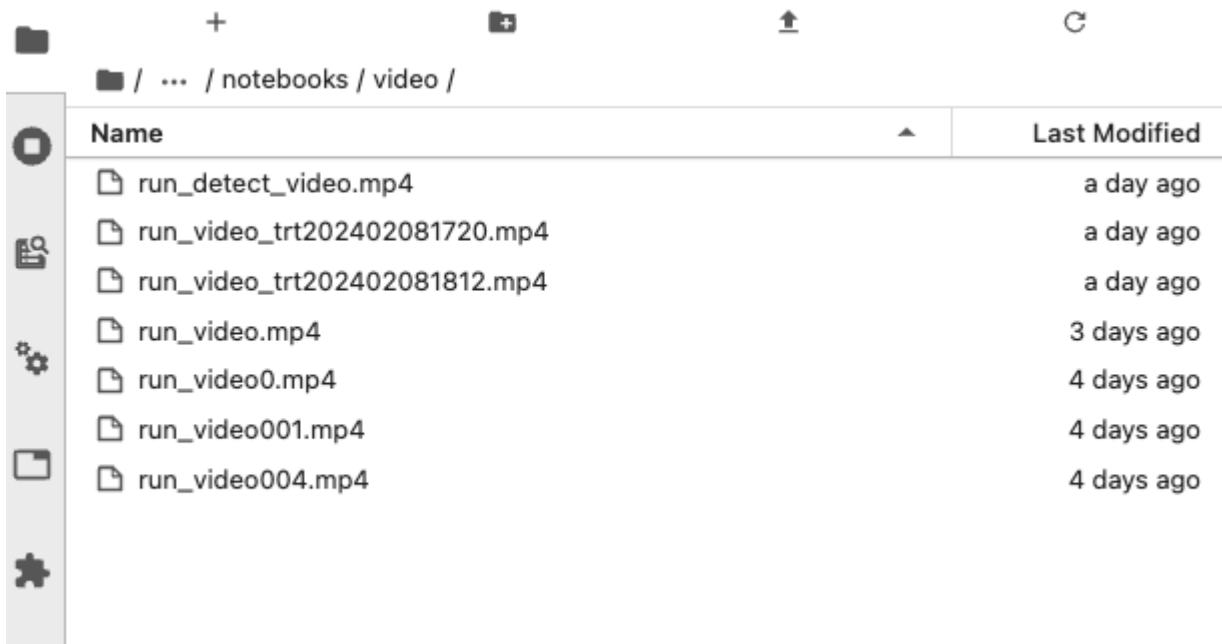
dataset	02061452	task	dataset
XYデータ数	322	速度データ数	187
epochs	30	train	evaluate
progress		loss	0
保存モデル名	model.pth	save model	
Useメモリ	3382	全メモリ	3964
			使用メモリ量の取得
ログ	68: 76枚目の/home/jetson/jetracer/notebooks/camera/b/xy/72_112_00156.jpgを読込みました。 67: 74枚目の/home/jetson/jetracer/notebooks/camera/b/xy/168_112_00154.jpgを読込みました。 66: 76枚目の/home/jetson/jetracer/notebooks/camera/b/xy/72_112_00156.jpgを読込みました。 65: 78枚目の/home/jetson/jetracer/notebooks/camera/b/xy/0_112_00158.jpgを読込みました。 64: 80枚目の/home/jetson/jetracer/notebooks/camera/b/xv/0_112_00160.jpgを読込みました。		

動画を作成します。検証に使う画像は学習に使われていない画像を使います。動画作成ボタンを押すとログにより推論時間も表示されます。

【6.評価動画の作成】 動画を作成します。

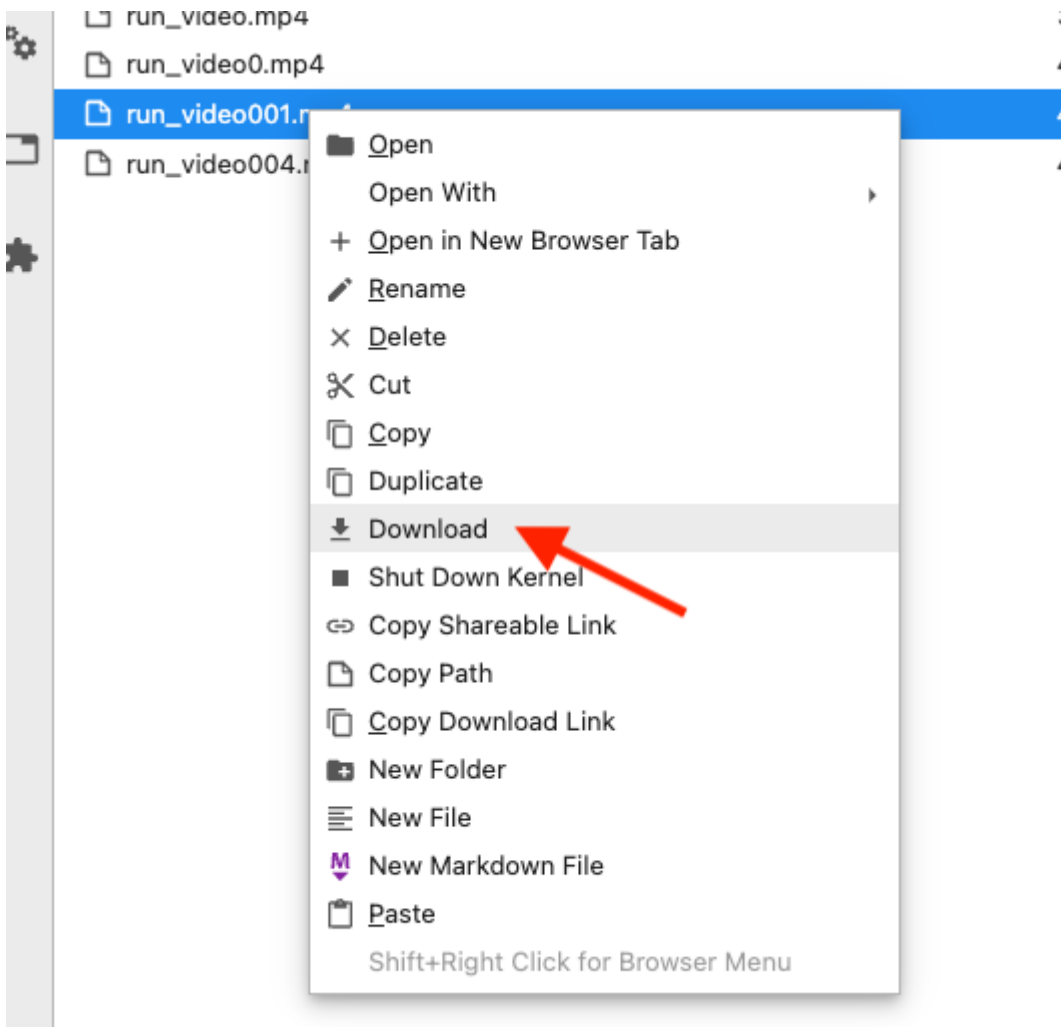
dataset	run001	task	camera
動画名	run_video	skip(枚)	2
			動画の作成
Useメモリ	3382	全メモリ	3964
			使用メモリ量の取得
ログ	68: 76枚目の/home/jetson/jetracer/notebooks/camera/b/xy/72_112_00156.jpgを読込みました。 67: 74枚目の/home/jetson/jetracer/notebooks/camera/b/xy/168_112_00154.jpgを読込みました。 66: 76枚目の/home/jetson/jetracer/notebooks/camera/b/xy/72_112_00156.jpgを読込みました。 65: 78枚目の/home/jetson/jetracer/notebooks/camera/b/xy/0_112_00158.jpgを読込みました。 64: 80枚目の/home/jetson/jetracer/notebooks/camera/b/xv/0_112_00160.jpgを読込みました。		

生成された動画ファイルはvideoフォルダに保存されます。

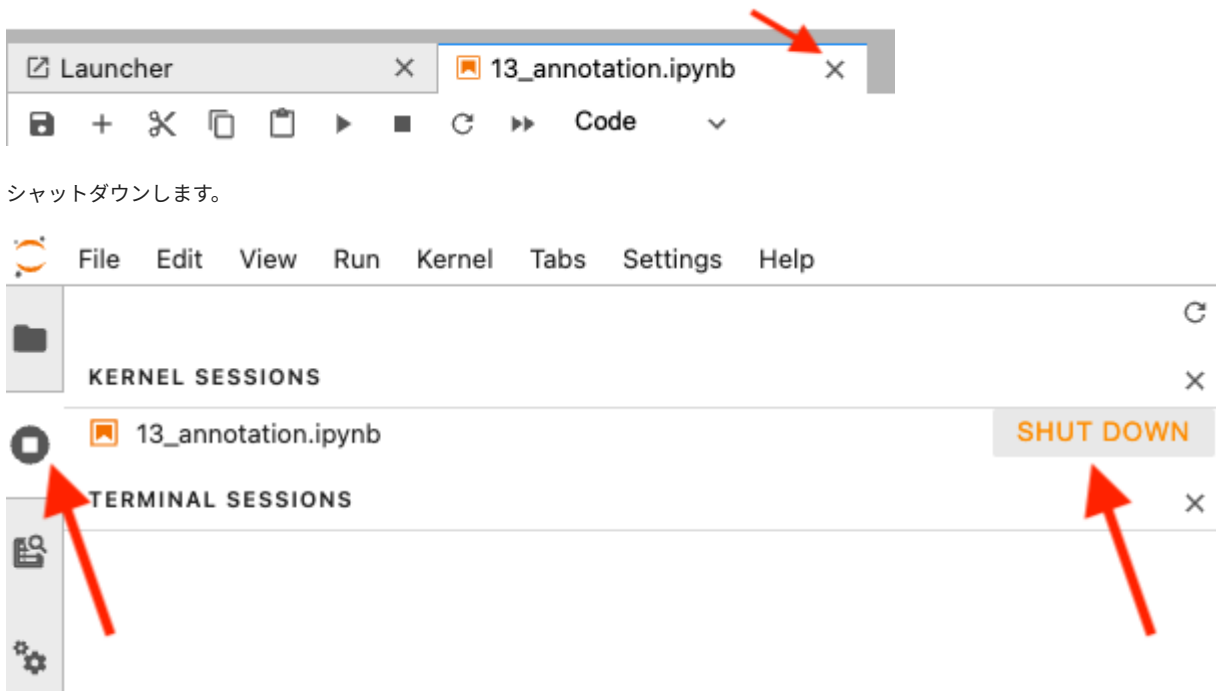


Name	Last Modified
run_detect_video.mp4	a day ago
run_video_trt202402081720.mp4	a day ago
run_video_trt202402081812.mp4	a day ago
run_video.mp4	3 days ago
run_video0.mp4	4 days ago
run_video001.mp4	4 days ago
run_video004.mp4	4 days ago

動画再生するには、ファイルをマウス右クリックでダウンロードしPCにて再生します。動画結果が悪いと判断した場合は、再びアノテーションをしてデータを追加し、学習をしてモデルを作ります。



終了するには、タブを閉じます。



モデルは生成されましたが自動走行する前にモデル変換が必要となります。

生成されたモデルは、RC走行では重く遅延が生じます。そこでTensorRTで最適化が必要になります。

次は、16_convert.ipynbにてモデルの最適化します。

学習が止まってしまう場合は？

空のファイルを見つける

```
1 sudo find ./ -empty
```

空のファイルの削除する。

```
1 sudo find ./ -empty -delete
```

Info

時刻が取得できないなど環境によっては、画像ファイルのタイムスタンプにより前後する場合があります。

3.2.6 TensorRTへの変換

モデルはできましたが、このままでは推論時間が時間がかかりRCカーの速度についていけません。ここでは軽量化して推論時間を短縮化させます。

解説動画

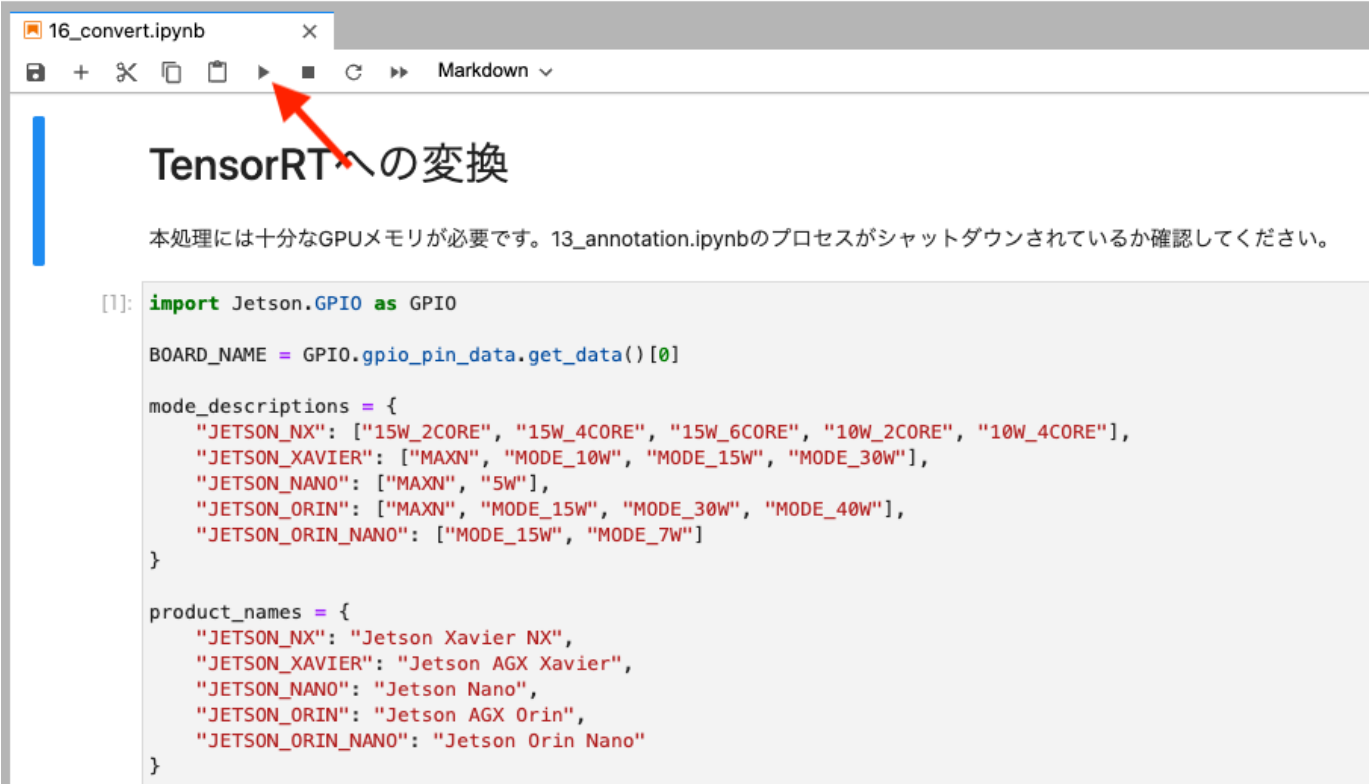


AI86 Version2

15.train.ipynbまたは13_annotation.ipynbでモデルを完成しましたら、推論の実行速度向上のため、16.convert.ipynbでモデルをTensor RTで変換いたします。

	05_road_following_live.ipynb	5 days ago
	10_find_raw.ipynb	5 days ago
	11_record_camera.ipynb	8 days ago
	12_file_manager.ipynb	4 days ago
	13_annotation.ipynb	4 days ago
	14_download.ipynb	5 days ago
	15_train.ipynb	5 days ago
	16_convert.ipynb	5 days ago
	17_run.ipynb	4 days ago
	20_detect.ipynb	5 days ago
	21_convert_detect.ipynb	5 days ago

セルを順番に上から実行して、shift + Enterキーまたは、 ボタンを押していきます。



しばらくすると、ウィジェットが現れて変換したいモデルを選んで変換後の名前をつけて、TRT Convertボタンを押して変換します。約1分弱ほど変換時間がかかります。

【1.TensorRTへの変換】 軽量化、高速化を目的にモデルをTensorRT形式に変換します。

変換前

./model/model001.pth

▼

作成日時：2024-02-05 20:03:47

変換後

result001.pth

TRT Convert

ログ

3: TRTへの変換処理中...(時間がかかります)

2: 変換前: ./model/model001.pth

1: モデルの変換を開始します。

変換が完了後、動画で評価をします。datasetのドロップダウンリストから検証で使いたい画像を選びます（今回は、13_annotation.ipynbでアノテーション済みの画像を選択します。）

【2.変換後のモデルの動画評価】 変換後のモデルで動画を作成し評価します。

dataset	run001	task	camera	
動画名	run_video_trt001	skip(枚)	2	動画の作成
ログ	3: TRTへの変換処理中...(時間がかかります) 2: 変換前: ./model/model001.pth 1: モデルの変換を開始します。			

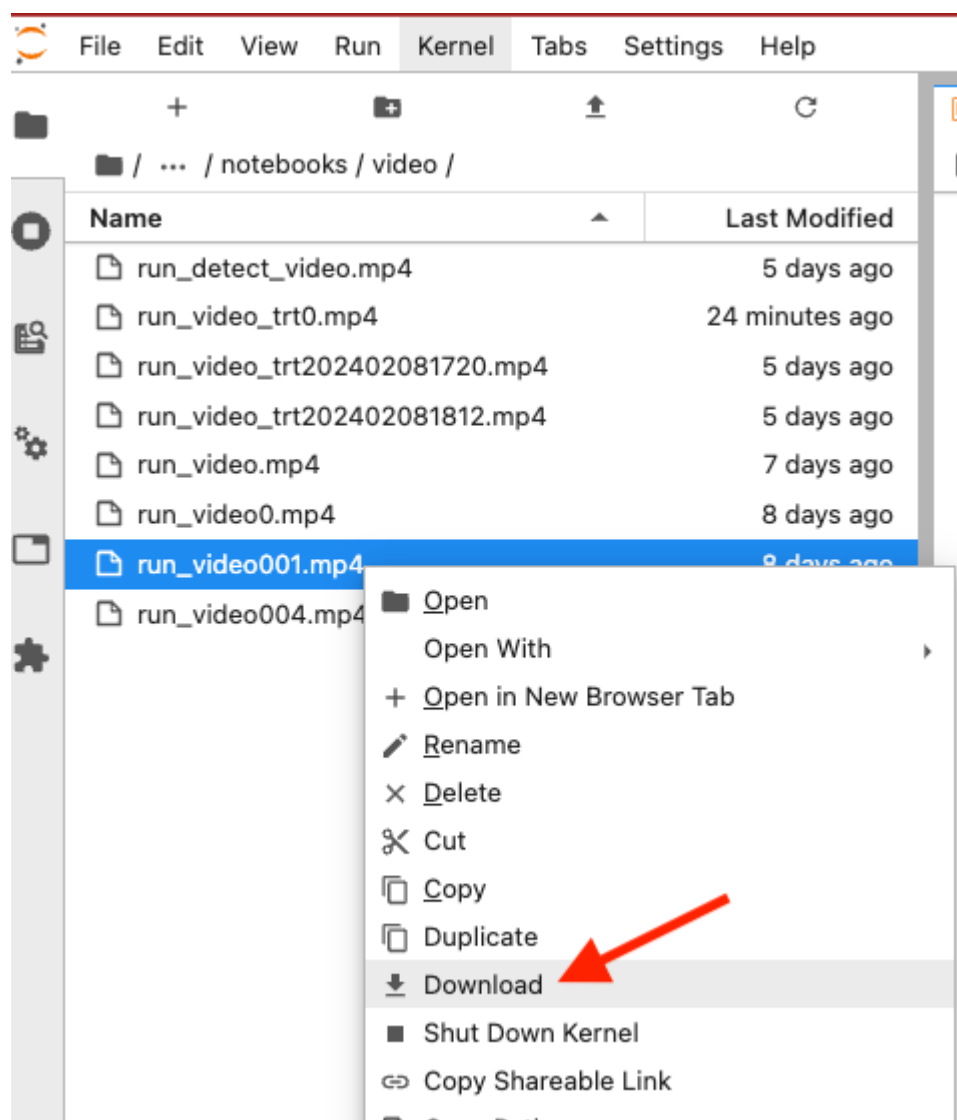
推論します。ログより推論時間が少なくなったことが確認できます。

【2.変換後のモデルの動画評価】 変換後のモデルで動画を作成し評価します。

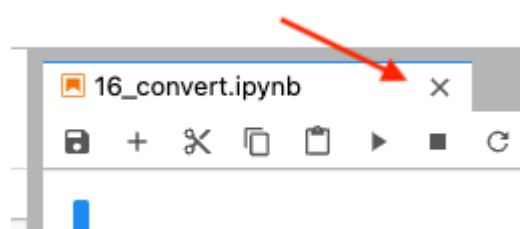
dataset	run001	task	camera	
動画名	run_video_trt001	skip(枚)	2	動画の作成
ログ	43: 動画の出力が完了しました。 42: 20.1秒まで完了, 推論処理平均: 16.0ms, 300枚目/304枚中を処理中 41: 19.4秒まで完了, 推論処理平均: 17.4ms, 290枚目/304枚中を処理中 40: 18.7秒まで完了, 推論処理平均: 15.8ms, 280枚目/304枚中を処理中 39: 18.1秒まで完了, 推論処理平均: 15.4ms, 270枚目/304枚中を処理中			

[]:

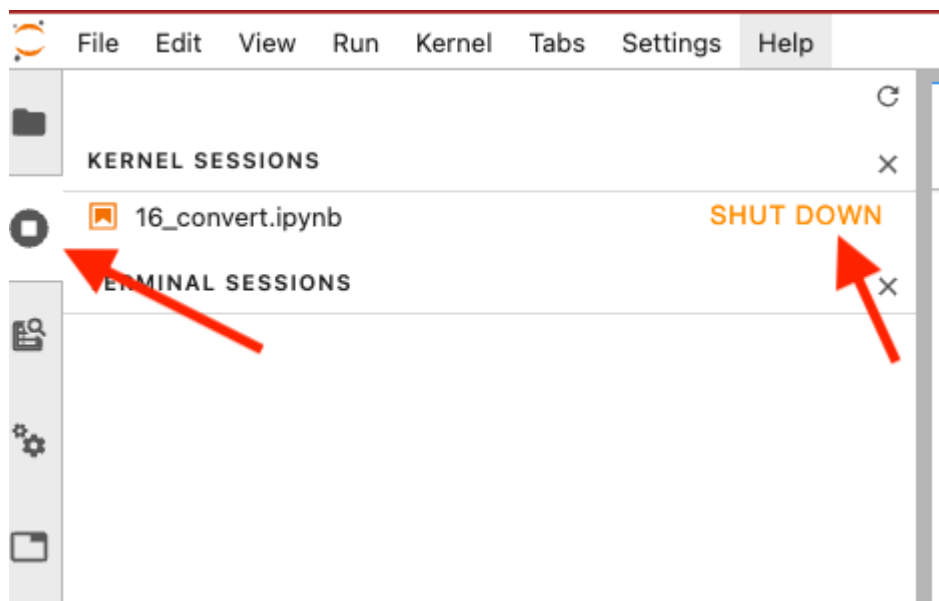
右クリック選択で、videoフォルダからダウンロードしパソコンで動画ファイルを再生し確認します。



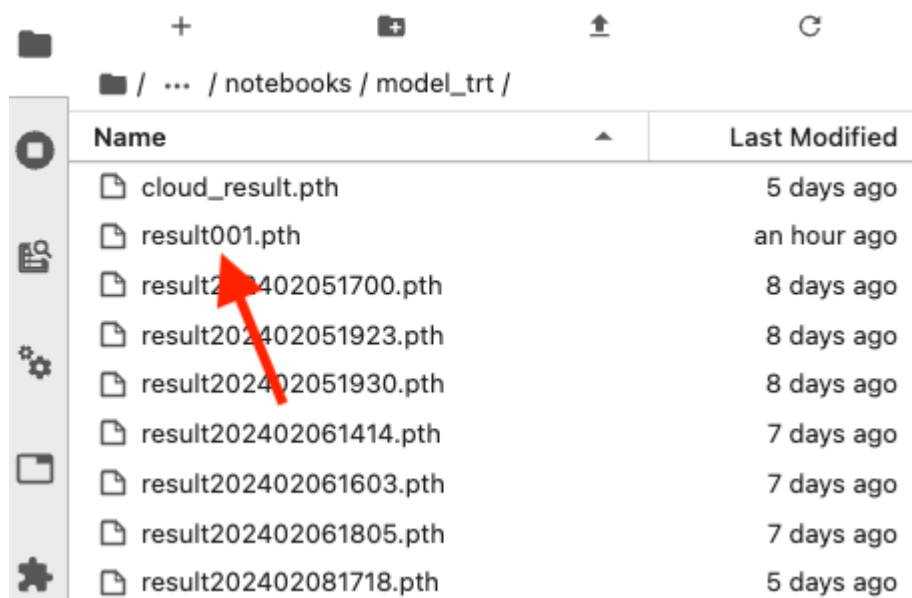
タブを閉じます。



シャットダウンします。



変換したモデルはmodel_trtフォルダに保存されます。



いよいよ次は、自動走行です。

3.2.7 自動走行

解説動画



04_road_following.ipynbの場合は、走行録画データはrunフォルダに蓄積されます。



AI86 Version2の自動走行では、17_run.ipynbを使います。ステアリングだけでなくスロットルに対しても推論を実行することができます。

セルを順番に上から実行して、shift + Enterキーまたは、 ボタンを押していきます。

しばらく経つと以下の画面が現れます。走行したいモデルを選んでTRTモデル読み込みを押します。呼び出しには数十秒かかります。

【1.使用する推論モデル】 TensorRTに変換済みのモデルをLoadします。

モデル	/model_trt/result202402081718	2024-02-08 17:19:20	<button>TRTモデルを読み込み</button>
ログ	2: ./model_trt/result202402081718.pthの読み込に成功しました。 1: ./model_trt/result202402081718.pthの読み込みを実行します(初回は時間がかかります)。		

ステアリングゲインは、実際の走行し、調整します。（初めは1.0がおすすめです。）

【2.Steeringゲイン】 Steeringのゲイン調整します。周りが悪い時は値を1.0以上にします。

Steering gain 1.00

スピードゲイン（スロットルのゲイン）は固定値か推論値で選べます。スロットルは一定の場合は、固定値を選びます。こちらも実際走行し、よく走るとな値を調整します。

【3.速度】 速度は固定値か、推論から反映かが選べます。

Speed

Speed raw 80

スロットルのアノテーションして学習させた場合は、推論値を選びます。

【3.速度】 速度は固定値か、推論から反映かが選べます。

Speed 推論値 ▼

Speed gain

 1.00

走行を録画したい場合は、録画にチェックを入れ、ファイル名を記入します。

【4.録画】 走行中の映像を録画したい場合はチェックマークを選択し、保存データセット名を指定してください。

☒ 録画 映像の保存名 run001

RCモード（グリーン点灯）になっていることを確認して、走行開始ボタンを押します。しばらく経つと、AIが起動し下記のようなログが現れます。プロポにあるボタンを押してAIモード（ピンク、レッド点灯）にします。RCカーの制御信号がプロポからJetsonに切り替わり自動走行します。走行が終了したい場合は、プロポのボタンを押して、RCモード（グリーン点灯）にして車体を停止から停止ボタンを押します。



周囲の安全を確保して走行します。危険を感じた時や車体衝突が起きそうなときはAIモードからRCモードに即座に切り替えて衝突を停止または回避しましょう。

【5.走行の開始】 走行開始を押す前にタイヤが空転していのを確認してください。プロポの裏のボタンを変換しAIモードにすると動き始めます。

Useメモリ 2817 全メモリ 3964 使用メモリ量の取得

走行開始 停止

ログ

8: 推論処理: 19.5ms, Speed: 63.0 (推論), Steering Gain: 1.0, FPS: 30.3 (3秒ごとに表示)

7: 推論処理: 19.4ms, Speed: 63.0 (推論), Steering Gain: 1.0, FPS: 30.3 (3秒ごとに表示)

6: 推論処理: 19.9ms, Speed: 63.0 (推論), Steering Gain: 1.0, FPS: 33.3 (3秒ごとに表示)

5: 推論処理: 401.2ms, Speed: 66.0 (推論), Steering Gain: 1.0, FPS: 2.3 (3秒ごとに表示)

4: AIが起動しました。

カメラの開放ボタンを押します。

【6.カメラのリリース】 走行を終了する場合は、カメラのリリースを実行し、Notebookをshutdownしてください。

Camera開放

ログ

12: カメラを開放しました。

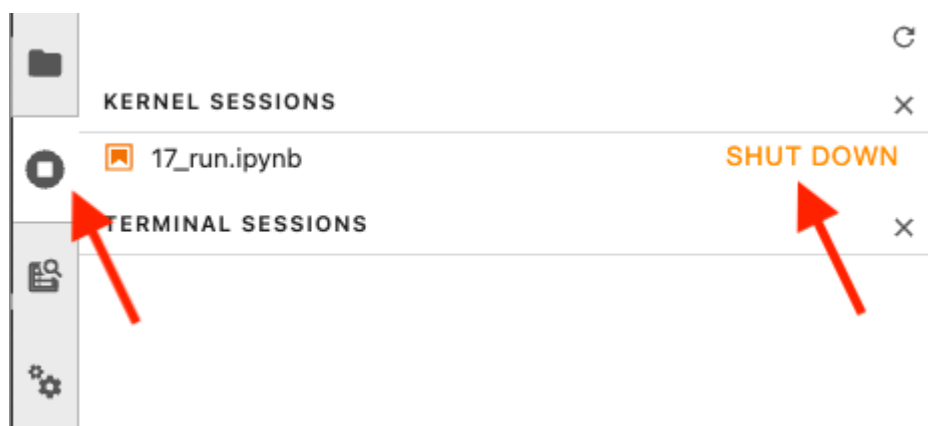
11: 推論処理: 19.5ms, Speed: 64.0 (推論), Steering Gain: 1.0, FPS: 30.3 (3秒ごとに表示)

10: 推論処理: 19.9ms, Speed: 63.0 (推論), Steering Gain: 1.0, FPS: 30.3 (3秒ごとに表示)

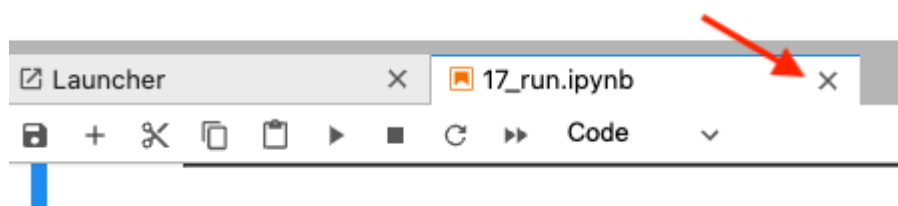
9: 推論処理: 19.7ms, Speed: 64.0 (推論), Steering Gain: 1.0, FPS: 30.3 (3秒ごとに表示)

8: 推論処理: 19.5ms, Speed: 63.0 (推論), Steering Gain: 1.0, FPS: 30.3 (3秒ごとに表示)

シャットダウンします。



タブも閉じます。



Info

17_run.ipynbでの走行録画データはcameraフォルダに蓄積されます。こちらのデータを元に教師データを追加して再び学習させます。

Info

うまく走れるようにStreeing Gain、Speed rawまたはSpeed Gainを調整してみてください。

Info

走行を改善したいときは、走行録画の画像データより悪い箇所を探し出して再びアノテーションしてモデルを生成してよく走るまで繰り返します。

3.2.8 Google Colaboratoryを使ったクラウド学習



FaBo JetRacerCloud Version 2.0.7

Jetson Nanoだけでもエッジでの学習はできますが、インターネット環境があればクラウドでも学習はできます。

Google Colaboratory(Google Colab)を使用することでGPUなどのハードウェアなしで使うことができます。Jupyter NoteBookは初めから用意されており、リッチな環境により学習時間を短縮、画像枚数もより多く扱うことが容易になります。

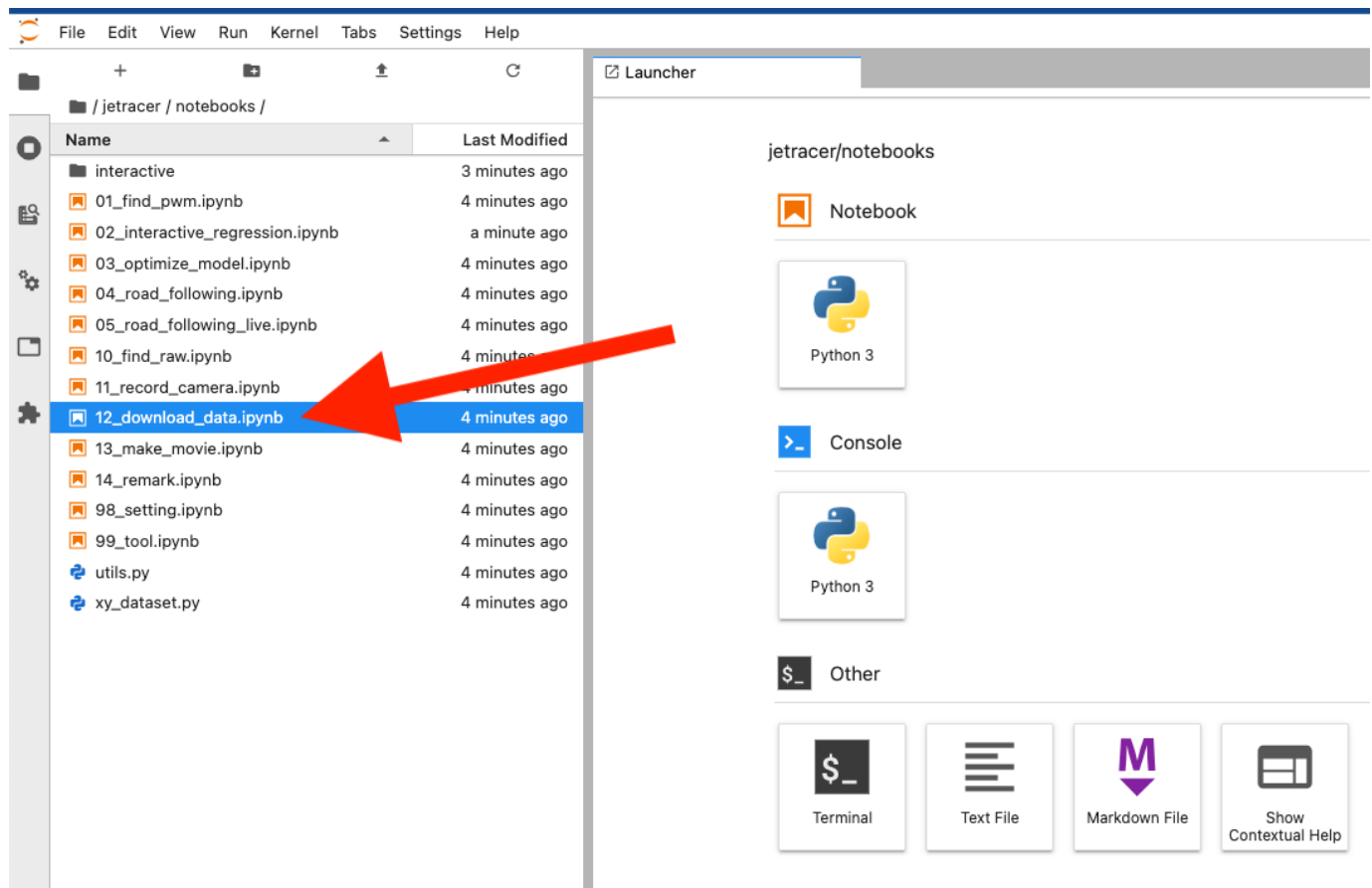
データの準備

13_anotation.ipynbで作成した教師データのsample001と11_record_camera.ipnbで作成した走行画像のsample001をあらかじめ用意しておきます。

データセットをzipファイルにまとめる

11_record_camera.ipnbで録画した画像データを12_file_manager.ipynbで編集し13_anotation.ipynbで教師データを作り、収集したデータセットをクラウドへアップします。JetsonNanoの14_download_data.ipynbを開いて

データセット（画像データ）を一つのzipファイルにしてGoogle Colaboratoryへのファイルアップロードできるようにします。



タスク(ディレクトリ)をドロップダウンリストから選んで、圧縮したいデータセットフォルダを選択します。

学習用のデータsample001(datasetフォルダのアノテーションで作成されたデータ)を1つのファイルにまとめます。ZIPに圧縮ボタンを押します。

【Download ZIP】 データセットを選択し、ZIPに圧縮します。zip/downloadフォルダにファイルは生成されます。

データセット	sample_001	タスク	dataset
XYデータ数	1166	速度データ数	1166
ZIPに圧縮			

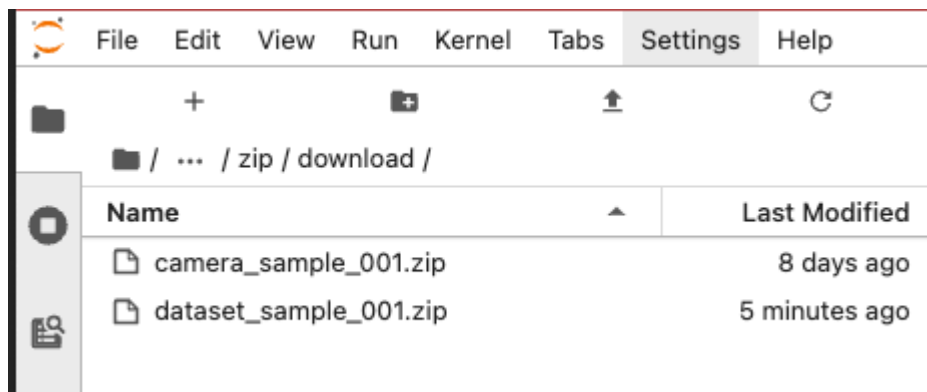
続きまして、テスト用データsample001（cameraフォルダにあるデータ）を1つのファイルにまとめます。

【Download ZIP】 データセットを選択し、ZIPに圧縮します。zip/downloadフォルダにファイルは生成されます。

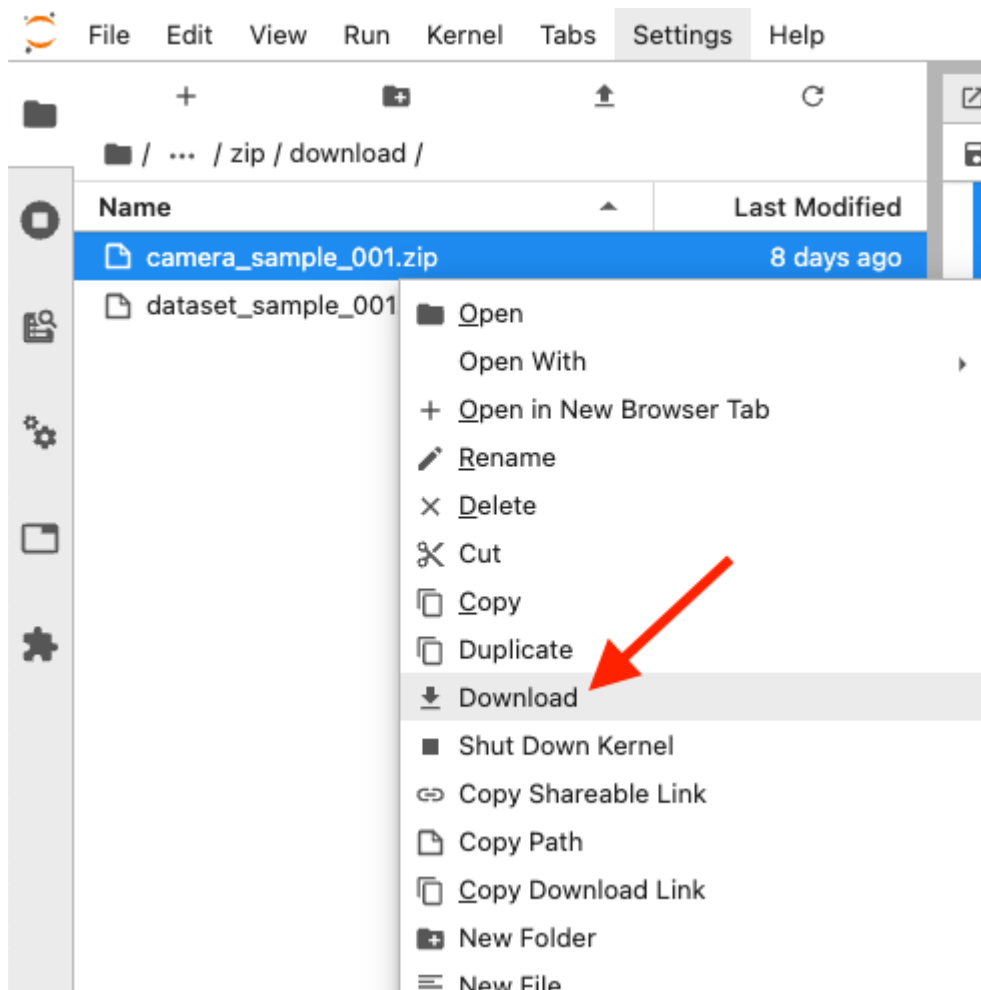
データセット	sample_001	タスク	camera
XYデータ数	1166	速度データ数	1166
ZIPに圧縮			

画像ファイル枚数によって終了時間が異なります。処理完了が出るまで待ちましょう。

zip化されたファイル名は、アノテーション済みのdataset、走行録画はcameraがデータセット名に接頭辞がつきます。



dataset_sample_001.zipとcamera_sample_001.zipができます。こちら2つのファイルをお使いのPCにダウンロードします。



Googleのマイドライブヘデータセットをアップロードする

作成したzipファイルは、zipフォルダにあります。ダウンロードしたいファイルを右クリックDownloadを選択します。

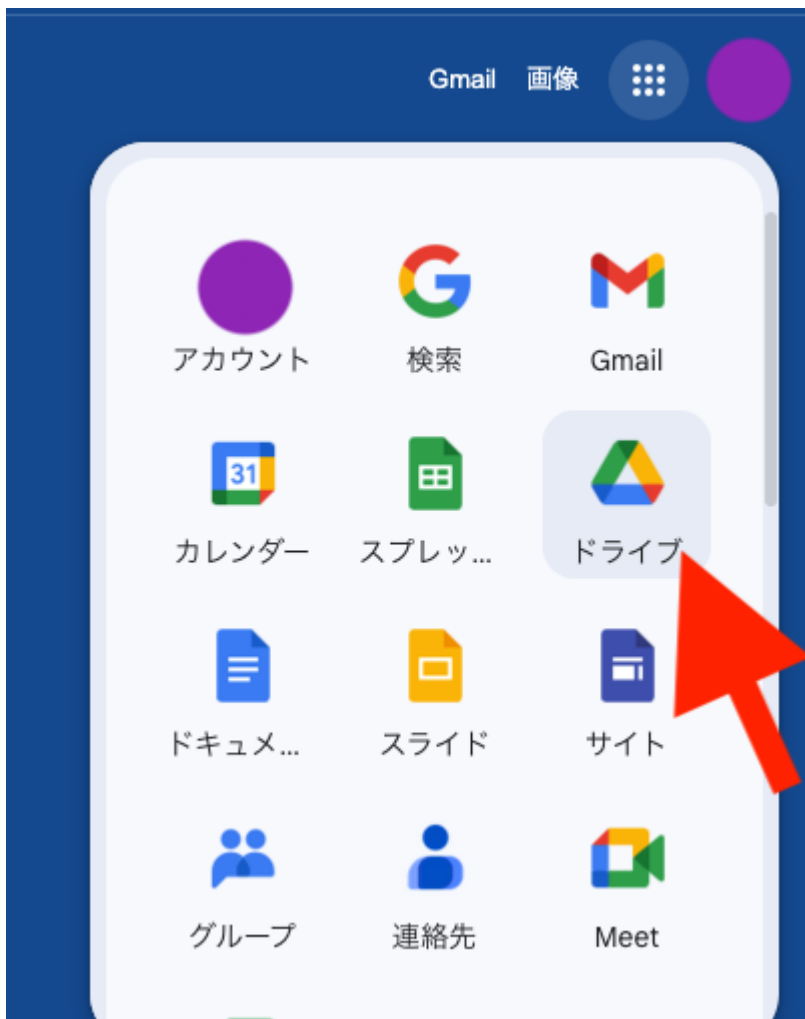
一旦、お使いのパソコンにダウンロードします。

右クリックでダウンロードします。

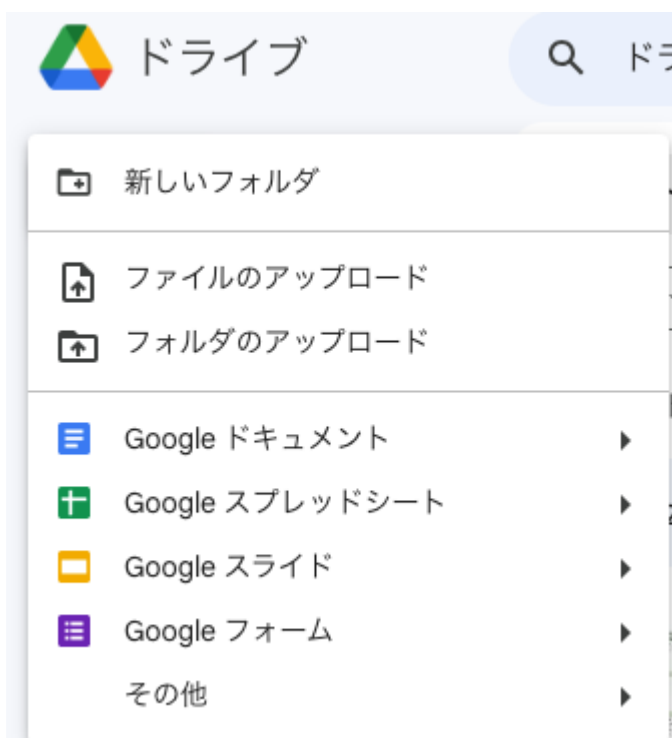
Google Colaboratoryを使用する前に、事前にGoogleアカウントが必要です。Googleにログインします。

事前に、Googleドライブ内にworkspace_jetracerという名前の専用のフォルダを作りましょう。

zipファイルをGoogleドライブにアップロードします。



新規ボタンをクリックして、作業フォルダ（workspace_jetracer）を作り、そのフォルダ内にzipファイルをアップロードします。



14_download.ipynbでzipにしたdataset_sample_001.zipとcamera_sample_001.zipをPCからGoogleドライブにアップロードします。



(例) 保存先のディレクトリ workspace_jetracer/

Google Colaboratory を使うための準備

Google Colaboratoryにアクセスします。

<https://colab.research.google.com/?hl=ja>

Google Colaboratoryにようこそ


Colaboratory へようこそ

[ファイル](#)
[編集](#)
[表示](#)
[挿入](#)
[ランタイム](#)
[ツール](#)
[ヘルプ](#)

☰

目次

🔍

はじめに

{x}

データサイエンス

📁

機械学習

📁

その他のリソース

📄

使用例

+

セクション

+ コード

+ テキスト

📁 ドライブにコピー

Colab へようこそ

すでに Colab をよくご存じの場合は、この動画を
ご覧ください。

Colab とは

Google Colaboratoryによる学習

15_train.ipynbにあるリンク先にアクセスします。

Train

データセットを結合して、学習します。

Jetson Orin Nanoでは、2000枚のデータセットでも1時間程度で学習はおわります。Jetson Nanoでは、2000枚のデータセットの学習には10時間程度は、下記URLから起動できるColabを試してください。

<https://colab.research.google.com/drive/1GbDrNiosTKSJNOJiCiVgv6V8X-0GDBfW?usp=sharing>

```
[ ]: import Jetson.GPIO as GPIO

BOARD_NAME = GPIO.gpio_pin_data.get_data()[0]

mode_descriptions = {
    "JETSON_NX": ["15W_2CORE", "15W_4CORE", "15W_6CORE", "10W_2CORE", "10W_4CORE"],
    "JETSON_XAVIER": ["MAXN", "MODE_10W", "MODE_15W", "MODE_30W"],
    "JETSON_NANO": ["MAXN", "5W"],
    "JETSON_ORIN": ["MAXN", "MODE_15W", "MODE_30W", "MODE_40W"],
    "JETSON_ORIN_NANO": ["MODE_15W", "MODE_7W"]
}

product_names = {
    "JETSON_NX": "Jetson Xavier NX",
    "JETSON_XAVIER": "Jetson AGX Xavier",
    "JETSON_NANO": "Jetson Nano".
```

<https://colab.research.google.com/drive/1GbDrNiosTKSJNOJiCiVgv6V8X-0GDBfW?usp=sharing>

下の矢印にあるセルに移動してShift+Enterを実行します。2つのファイルが/content/にコピーされます。



JetRacerCloud.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ 変更を保存できませんでした

ファイル

sample_data

utils.py

xy_dataset.py

▼ FaBo JetRacerCloud ver 2.0.7

- Version 2.0.7(2023年11月29日公開)
- 対応車体: FaBo JetRacer AI86 version 2.x

使用実績

- 自動運転ミニカーバトル2023年決勝走行モデルの作成でも使用

▼ 必要なパッケージのインストール

xy_dataset.pyとutils.pyをダウンロードしてきます。

```
!wget https://raw.githubusercontent.com/NVIDIA-AI-IOT/jetracer/master/notebooks/utils.py
!wget https://raw.githubusercontent.com/NVIDIA-AI-IOT/jetracer/master/notebooks/xy_dataset.py
```

--2024-02-13 09:38:38-- https://raw.githubusercontent.com/NVIDIA-AI-IOT/jetracer/master/notebooks/utils.py
 Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.
 Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
 HTTP request sent, awaiting response... 200 OK
 Length: 483 [text/plain]
 Saving to: 'utils.py'
 utils.py 100%[=====>] 483 --KB/s in 0s

下のコードを実行するとダイアログが表示されます。GoogleDriveへのアクセスを許可します。

▼ データセットの用意

Jetson上で、**14_download.ipynb** を起動し、データセットをzipで圧縮します。圧縮したzipは、JupyterLabのzipフォルダにA.zipのようなファイルで生成されください。

Google Driveをcolabにマウントします。

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

今回は学習用のデータセットが1つの場合は、矢印のところに、DATA_ZIP = ["dataset_sample_001.zip"]を追記します。

- **DATA_FOLDER** Google Driveのフォルダ名にあわせます
- **DATA_NAME** zipファイル名を記載します

```

import os
import shutil
import zipfile

DATA_FOLDER = "/content/drive/MyDrive/workspace_jettracer/" # ユーザ環境に依存
DATA_ZIP = ["dataset_sample_001.zip"]
DATA_ZIPS_STRIGHT = ["dataset_straight_101.zip",
                     "dataset_straight_102.zip",
                     "dataset_straight_103.zip",
                     "dataset_straight_104.zip",
                     "dataset_straight_105.zip",
                     "dataset_straight_106.zip",
                     "dataset_straight_107.zip",
                     "dataset_straight_108.zip",
                     "dataset_straight_109.zip",
                     "dataset_straight_110.zip",
                     "dataset_straight_111.zip"]

```

また、同じセル内で下の矢印のように

```

1 DATA_ZIPS = DATA_ZIP
2 #DATA_ZIPS = DATA_ZIPS_STRIGHT

```

と変更する。

```

                     "dataset_cut_110.zip",
                     "dataset_cut_111.zip",
                     "dataset_cut_112.zip",
                     "dataset_cut_113.zip"
                 ]

DATA_ZIPS_TIMEATTACK = ["dataset_timeattack_101.zip",
                       "dataset_timeattack_102.zip",
                       "dataset_timeattack_103.zip",
                       "dataset_timeattack_104.zip",
                       "dataset_timeattack_105.zip"]

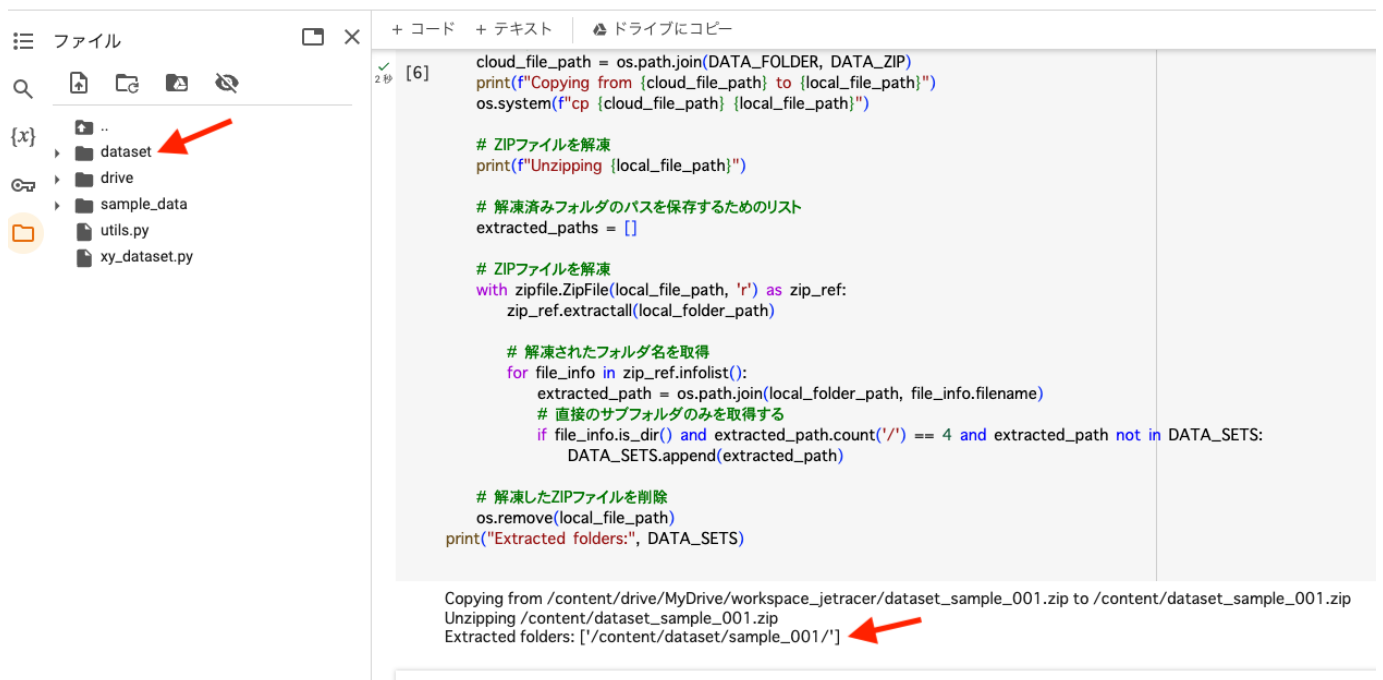
DATA_ZIPS = DATA_ZIP
#DATA_ZIPS = DATA_ZIPS_STRIGHT
#DATA_ZIPS = DATA_ZIPS_SHORTCUT
#DATA_ZIPS = DATA_ZIPS_TIMEATTACK

DATA_SETS = []
local_folder_path = "/content/"

for DATA_ZIP in DATA_ZIPS:

```

実行すると/content/ディレクトリにdatasetフォルダに解凍されるはずですが。/content/dataset/sample001/



The screenshot shows the Google Colaboratory interface. On the left, the file explorer shows a directory structure with 'dataset' highlighted. The main area contains a code cell [6] that performs the following actions:

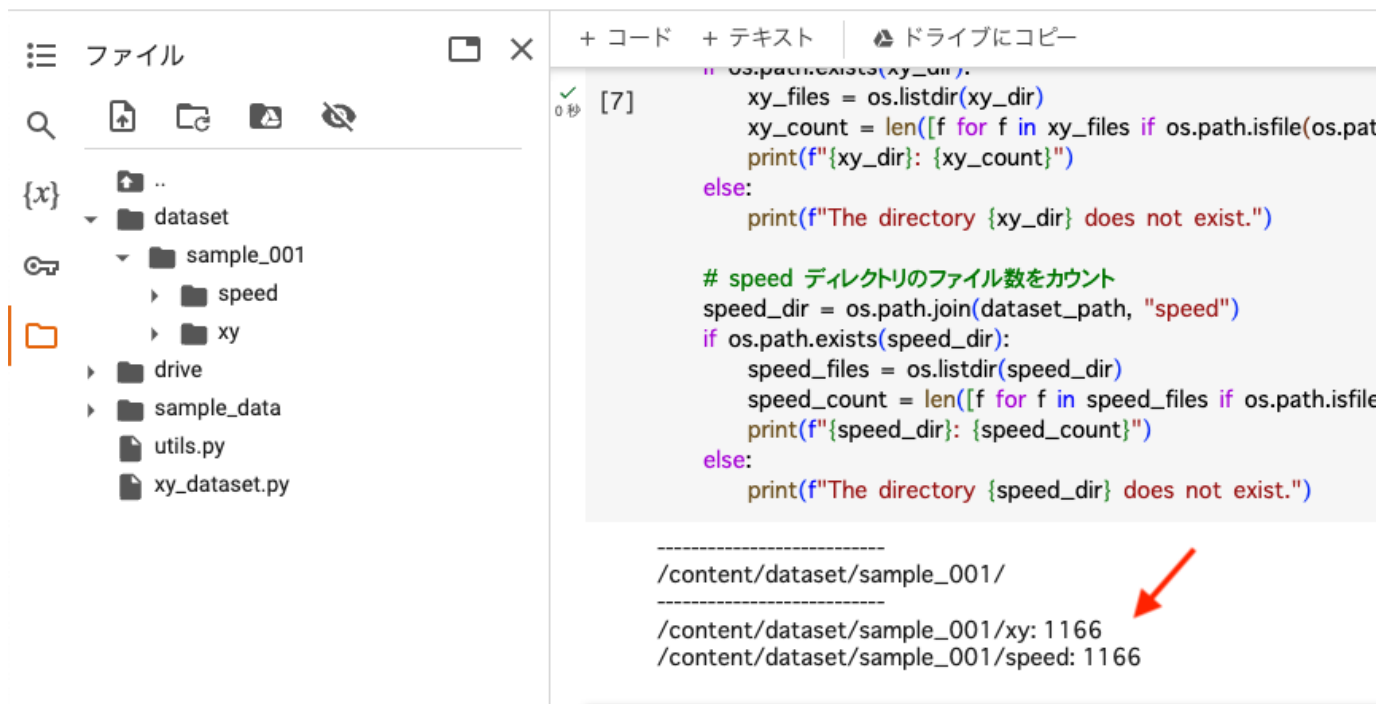
- Defines `cloud_file_path` and `local_file_path`.
- Prints a message: "Copying from {cloud_file_path} to {local_file_path}" and executes `os.system(f"cp {cloud_file_path} {local_file_path}")`.
- Prints "Unzipping {local_file_path}" and executes `zipfile.ZipFile(local_file_path, 'r').extractall(local_folder_path)`.
- Iterates over `zip_ref.infolist()` to extract files, skipping subdirectories.
- Prints "Extracted folders:" followed by the contents of `DATA_SETS`.

The output at the bottom shows the execution results:

```
Copying from /content/drive/MyDrive/workspace_jetracer/dataset_sample_001.zip to /content/dataset_sample_001.zip
Unzipping /content/dataset_sample_001.zip
Extracted folders: ['/content/dataset/sample_001/']
```

ファイルの結合、ファイルの枚数を確認します。

データ存在確認



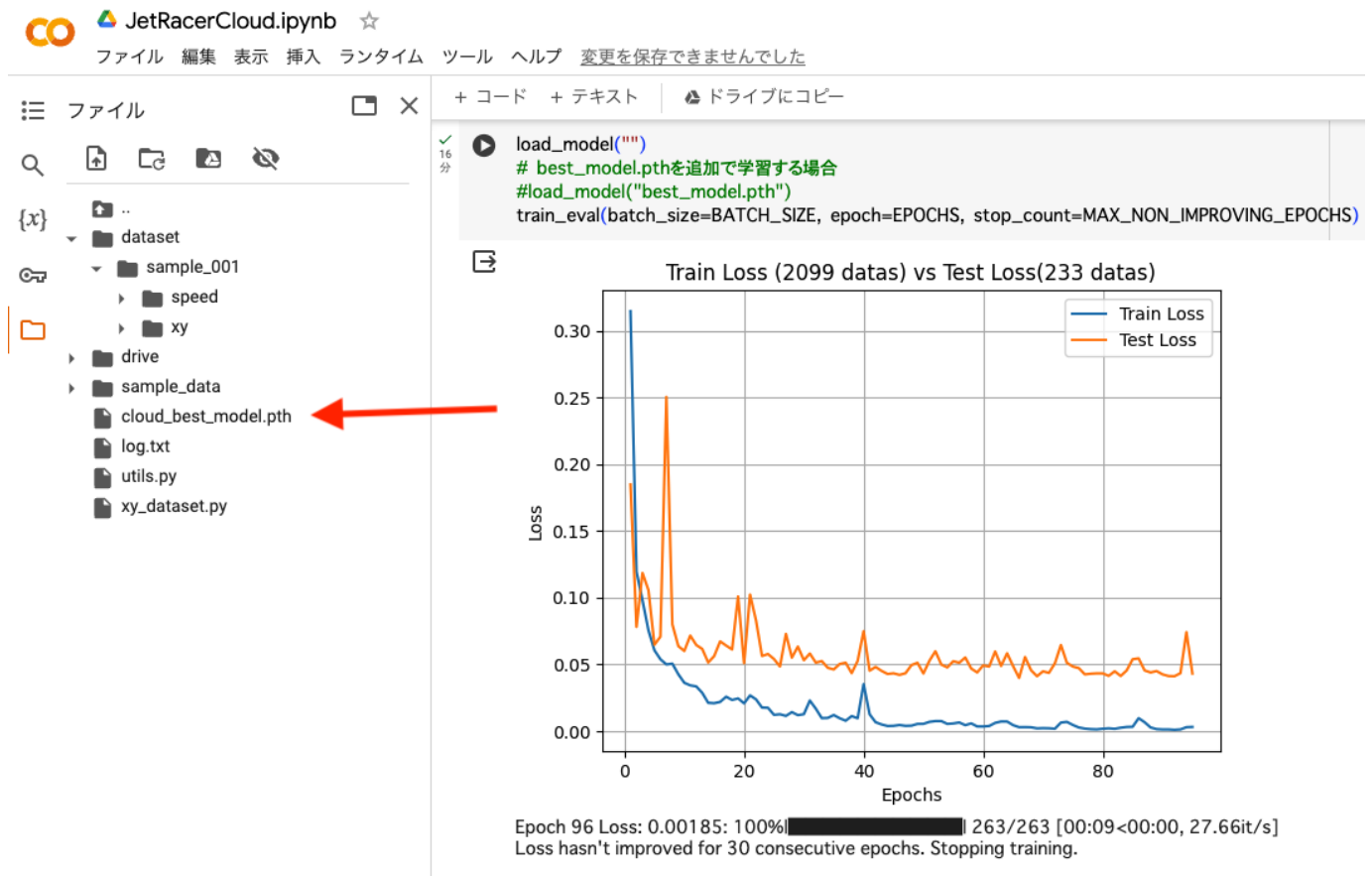
The screenshot shows the Google Colaboratory interface. On the left, the file explorer shows the directory structure with 'dataset/sample_001' expanded, showing 'speed' and 'xy' subdirectories. The main area contains a code cell [7] that counts files in these directories:

- Checks if `xy_dir` exists and counts files in `xy_files`.
- Checks if `speed_dir` exists and counts files in `speed_files`.

The output at the bottom shows the results:

```
/content/dataset/sample_001/
/content/dataset/sample_001/xy: 1166
/content/dataset/sample_001/speed: 1166
```

データの結合（データが1つしかないのでは結合はしません。）



学習が完了し! `cp /content/cloud_best_model.pth {DATA_FOLDER}`を実行するとGoogleドライブにモデルがコピーされます。

ベストモデルをGoogle Driveのフォルダにコピーする

```
[14] !cp /content/cloud_best_model.pth {DATA_FOLDER}
```

モデルが追加されました。

マイドライブ > workspace_jetracer ▾

workspace_jetracer

種類 ▾ ユーザー ▾ 最終更新 ▾

名前 ▾	オーナー
dataset_sample_001.zip	自分
cloud_best_model.pth	自分
camera_sample_001.zip	自分

以下を実行されますと作成したモデルがお使いのPCにダウンロードされます。こちらのファイルをJetsonのnotebookアップロードボタンでmodelフォルダにアップロードします。

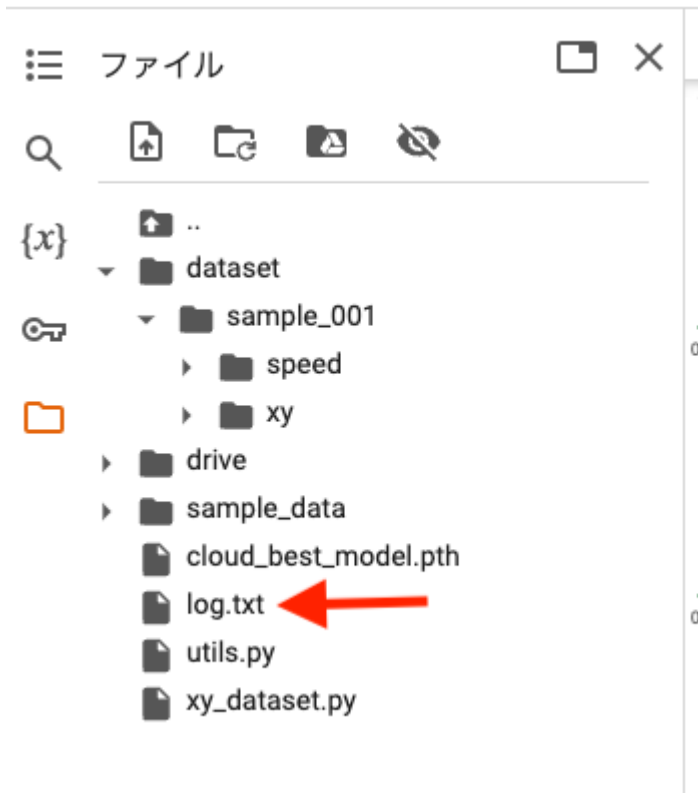
▽ Bestモデルのダウンロード

```

✓
0 秒 [23] if 'google.colab' in str(get_ipython()):
        from google.colab import files
        files.download('cloud_best_model.pth')

```

log.txt(実行結果)も保存されます。



lossが低くなるように任意のエポック数に変えて、学習させます。（無料でのご使用の場合は、時間制限があります。ご注意ください。）

検証動画を作成

Googleドライブworkspaceにある画像データ(camaera_sample_001.zip)を使用します。

以下を変更します。

```
1 CAMERA_DATA_NAME = "camera_sample_001.zip"
```

✓ 走行データのパスを指定して映像を作成

学習にはつかっていない走行データをつかって映像の作成を実施してください。

```

CAMERA_DATA_NAME = "camera_sample_001.zip"
CAMERA_DATA_FOLDER = "/content/drive/MyDrive/workspace_jetracer/"

# "_"で分割してフォルダ名を生成
folder_parts = CAMERA_DATA_NAME.split("_")
top_folder = folder_parts[0] # トップレベルのフォルダ名
sub_folder = "_".join(folder_parts[1:]).replace(".zip", "") # サブフォルダ名
camera_folder_name = os.path.join(top_folder, sub_folder) # 完全なフォルダパス

local_folder_path = "/content/"
local_file_path = os.path.join(local_folder_path, CAMERA_DATA_NAME)
extracted_folder_path = os.path.join(local_folder_path, camera_folder_name)

```


JetRacerCloud.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ 変更を保存できませんでした

ファイル

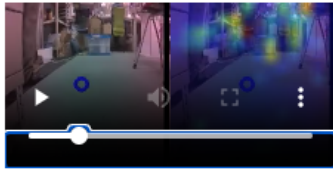
- ..
- camera
- dataset
 - sample_001
 - speed
 - xy
- drive
- sample_data
 - cloud_best_model.pth
 - log.txt
 - utils.py
 - video_0213_1914.mp4
 - xy_dataset.py

+ コード + テキスト | ドライブにコピー

[24] 13 秒

```
from IPython.display import HTML
from base64 import b64encode

mp4 = open(compressed_path, 'rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML("""
<video width=224 controls>
  <source src="%s" type="video/mp4">
</video>
""") % data_url)
```



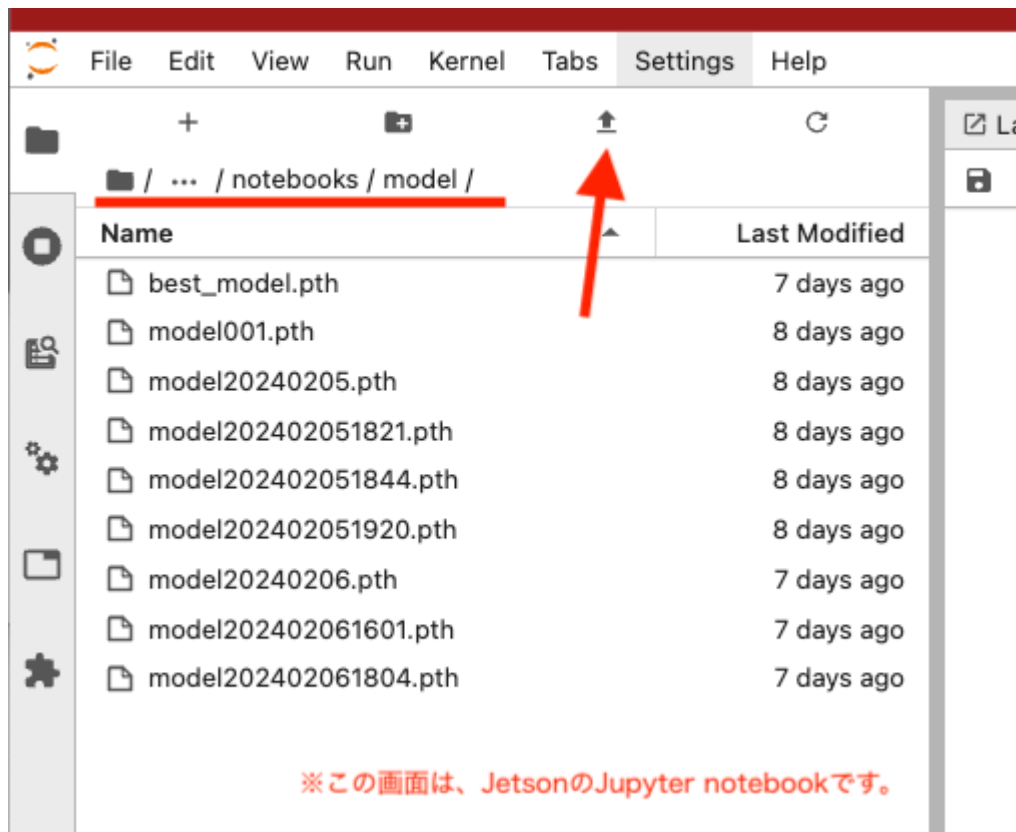
出来たモデルの評価を撮影済みの画像を使って評価します。結果が芳しくなかった場合は、データセットを増やしたり、エポック数を増やして再び学習し調整します。



Google Colaboratoryでの無料でお使いの場合は、GPUの使用時間制限がございます。データセット枚数とエポック数を調整しましょう。

保存した学習済みモデルのファイルをJetson Nanoのmodelディレクトリ(/home/jetson/jetracer/notebooks/model)へアップロードします。

Jetson NanoのJupyter labのアップロードボタンをクリックして、モデルpthファイルを/home/jetson/notebooks/model/へアップロードし16_convert.ipynb(TensorRT)にてモデルの最適化し、17_run.ipynbにて自動走行します。



どんどん画像を結合し全体で学習して精度を上げていきましょう。



コーナリング中に乱れるような推論を極力無くしていきましょう。

3.2.9 受信機信号計測



概要

実際のRCカー受信機の信号をもとに自動でアノテーション作業（オートアノテーション）を実行するようにします。
(11_record_camera.ipynb)

RCカーは、プロポ（送信機）から受信機で信号を受信し、ステアリング信号（ST）とスロットル信号（TH）でRCカー制御していますが、その受信機から出ている2つのPWM信号を計測します。信号の周期は約60Hz、波長は、およそ1000μ秒～2000μ秒になります。

Danger

作業台に置いて必ず実行してください。

Info

コントロール基板 #6112 JetRacer Rev 2.0.9以降の基板のみ対応

Info

Connectボタンを押すと下記のエラーが出る時がございます。

Traceback (most recent call last): File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner

もし、その場合は本コードにある数字をお願いいたします。以下のコードの8の部分を変更して下さるようお願いいたします。

```
data1 = i2c.read_i2c_block_data(addr, 0x01, 8)を
```

```
data1 = i2c.read_i2c_block_data(addr, 0x01, 12)に変更して保存ボタンを押します。
```

計測方法

制御信号を計測しているのは、JetsonNanoではなく、コントロール基板のマイコンが受信機からの制御信号を計測しています。I2Cのデバイスアドレスは0x08になります。

I2Cの認識確認

- 0x08: PWMの値の吸い上げ
- 0x3c: OLED Display
- 0x40: PCA9685(PWMの出力)
- 0x70: PCA9685(PWMの出力)

```
[4]: !i2cdetect -y -r $I2C_BUSNUM
```

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:									08							
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	3c	--	--	--	--
40:	40	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	70	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

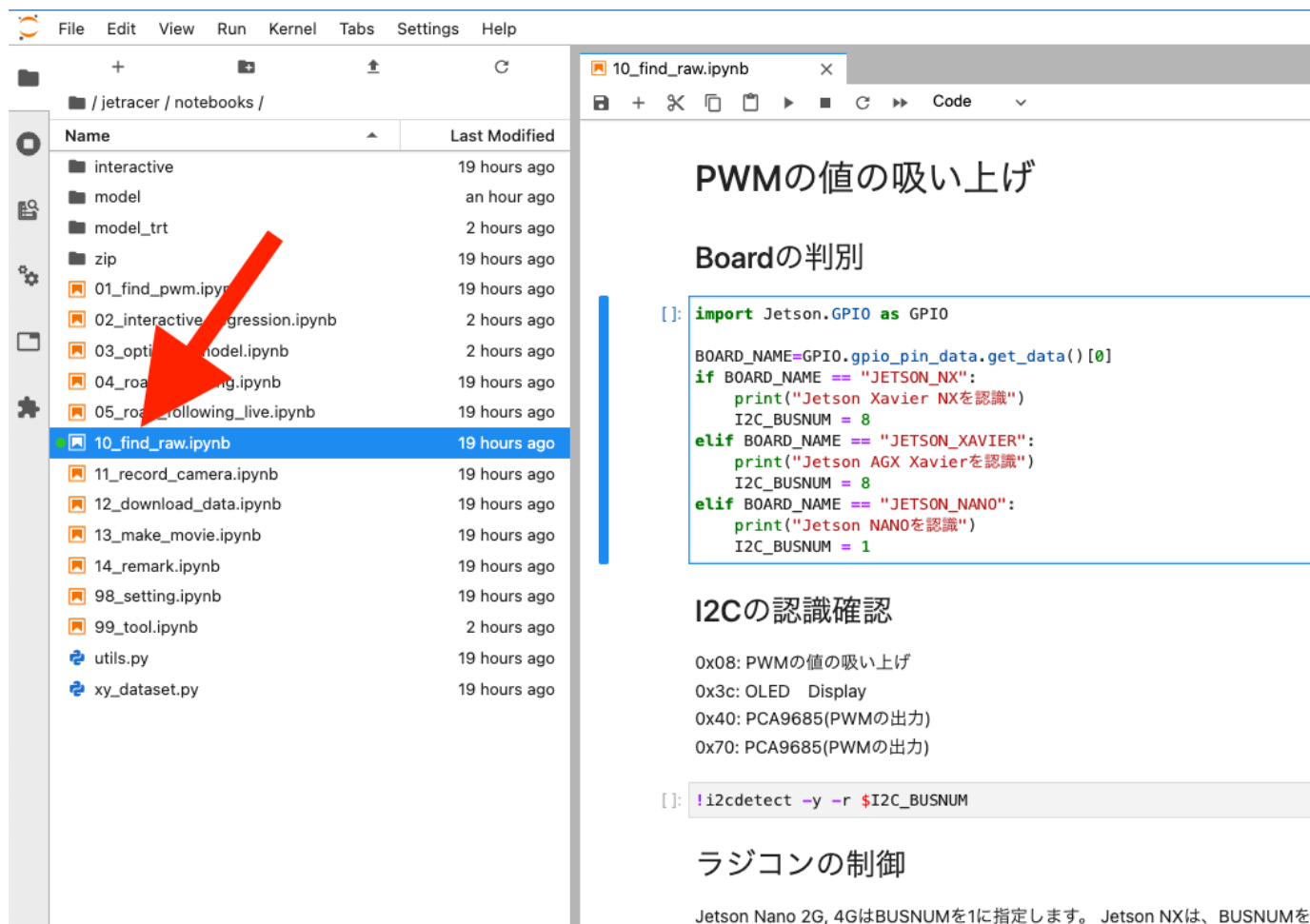
事前にRCモードでトリム調整した走行が良い状態を維持し、設定後は、ステアリングトリム、スロットルトリム、リバーススイッチはなるべく動かさないようにしてください。



必ずJetRacerを作業台に設置し、プロポの後ろのボタンによりRCカーモード（赤色）に切り替えます。



10_find_raw.ipynbを開きます。



File Edit View Run Kernel Tabs Settings Help

/ jetracer / notebooks /

Name	Last Modified
interactive	19 hours ago
model	an hour ago
model_trt	2 hours ago
zip	19 hours ago
01_find_pwm.ipynb	19 hours ago
02_interactive_regression.ipynb	2 hours ago
03_optimal_model.ipynb	2 hours ago
04_robot_navigation.ipynb	19 hours ago
05_robot_following_live.ipynb	19 hours ago
10_find_raw.ipynb	19 hours ago
11_record_camera.ipynb	19 hours ago
12_download_data.ipynb	19 hours ago
13_make_movie.ipynb	19 hours ago
14_remark.ipynb	19 hours ago
98_setting.ipynb	19 hours ago
99_tool.ipynb	2 hours ago
utils.py	19 hours ago
xy_dataset.py	19 hours ago

PWMの値の吸い上げ

Boardの判別

```
[ ]: import Jetson.GPIO as GPIO

BOARD_NAME=GPIO.gpio_pin_data.get_data()[0]
if BOARD_NAME == "JETSON_NX":
    print("Jetson Xavier NXを認識")
    I2C_BUSNUM = 8
elif BOARD_NAME == "JETSON_XAVIER":
    print("Jetson AGX Xavierを認識")
    I2C_BUSNUM = 8
elif BOARD_NAME == "JETSON_NANO":
    print("Jetson NANOを認識")
    I2C_BUSNUM = 1
```

I2Cの認識確認

0x08: PWMの値の吸い上げ
0x3c: OLED Display
0x40: PCA9685(PWMの出力)
0x70: PCA9685(PWMの出力)

```
[ ]: !i2cdetect -y -r $I2C_BUSNUM
```

ラジコンの制御

Jetson Nano 2G, 4GはBUSNUMを1に指定します。 Jetson NXは、BUSNUMを

ステアリング信号を計測

RCカー電源をONにして、プロポの電源もONにします。



ピーピーと音が聞こえる場合は、ESC（アンプ）が制御信号を受信できてない時に音が鳴ります。


```
[9]: type = TYPE_SPEED
train_eval_widget = ipywidgets.VBox([
    ipywidgets.HBox([raw_front_widget]),
    ipywidgets.HBox([raw_stop_widget]),
    ipywidgets.HBox([raw_back_widget]),
    ipywidgets.HBox([start_button, set_button, stop_button])
])

display(train_eval_widget)
```



前進	1028
停止	1516
後進	1960

Connect
Set
Stop

```
[10]: print("raw 前進: %s" % raw_front)
print("raw 中央: %s" % raw_stop)
print("raw 後進: %s" % raw_back)

raw 前進: 1028
raw 中央: 1516
raw 後進: 1960
```

Connectボタンをクリックします。

プロポのステアリングを左にいっぱいに切り、計測します。



Setボタンをクリックします。

プロポのステアリングを中立にし、計測します。

中立の状態にします。



Setボタンをクリックします。

プロポのステアリングをを右にいっぱいに切り、計測します。



Setボタンをクリックします。

スロットル信号を計測

```
[7]: type = TYPE_STEERING
train_eval_widget = ipywidgets.VBox([
    ipywidgets.HBox([raw_left_widget]),
    ipywidgets.HBox([raw_center_widget]),
    ipywidgets.HBox([raw_right_widget]),
    ipywidgets.HBox([start_button, set_button, stop_button])
])

display(train_eval_widget)
```



左	1140
中央	1572
右	2024

Connect

Set

Stop

```
[8]: print("raw 左: %s" % raw_left)
print("raw 中央: %s" % raw_center)
print("raw 右: %s" % raw_right)
```

raw 左: 1140

raw 中央: 1572

raw 右: 2024

プロポのレバーを引いて前進状態にして計測します。



Setボタンをクリックします。

プロポのレバーを離します。中立状態にして計測します。

中立の状態にします。



Setボタンをクリックします。

プロポのレバーを前に引いて後進状態にして計測します。



Setボタンをクリックします。

計測されたデータはraw_params.jsonファイルへ記憶されます。実際、計測された数値が正しいか確認してください。

```
[11]: import json

json_str = {
    "raw_steering": {
        "left": raw_left,
        "center": raw_center,
        "right": raw_right,
    },
    "raw_speed": {
        "front": raw_front,
        "stop": raw_stop,
        "back": raw_back,
    }
}

with open('raw_params.json', 'w') as f:
    json.dump(json_str, f, ensure_ascii=False)

[12]: print(json_str)
{'raw_steering': {'left': 1140, 'center': 1572, 'right': 2024}, 'raw_speed': {'front': 1028, 'stop': 1516, 'back': 1960}}
```

/home/jetson/jetracer/notebooksにraw_params.jsonファイルが生成されます。



 05_road_following_live.ipynb	19 hours ago
 10_find_raw.ipynb	3 minutes ago
 11_record_camera.ipynb	19 hours ago
 12_download_data.ipynb	19 hours ago
 13_make_movie.ipynb	19 hours ago
 14_remark.ipynb	19 hours ago
 98_setting.ipynb	19 hours ago
 99_tool.ipynb	2 hours ago
 raw_params.json	2 minutes ago
 utils.py	19 hours ago
 xy_dataset.py	19 hours ago

3.2.10 動画の作成



録画した一枚一枚の撮影した写真を動画にします。出来上がったモデルを動画で検証できるので便利です。

13_make_movie.ipynbを開きます。

The screenshot shows the JupyterLab interface. On the left, the file explorer displays a list of files and folders in the directory `/jetracer / notebooks /`. A red arrow points to the file `13_make_movie.ipynb`, which is highlighted. On the right, the code editor shows the title `動画の作成` and several code cells. The code includes imports for `ipywidgets`, `os`, and `cv2`, and defines a `task_widget` and `process_widget` using `ipywidgets`. It also includes a `write_log` function and imports `TRTModule` from `torch2trt`.

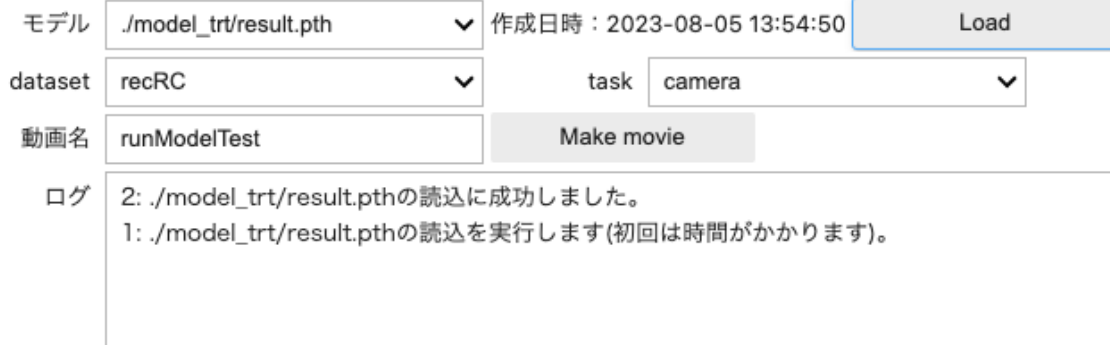
TensorRTのモデルを選び画像をLoadボタンをクリックして読み込みます。

```

change_task(TASK[0])

movie_collection_widget = ipywidgets.VBox([
    ipywidgets.HBox([model_widget,model_time_widget,load_button]),
    ipywidgets.HBox([datasets_widget,task_widget]),
    ipywidgets.HBox([video_name_widget,make_button]),
    ipywidgets.HBox([progress_widget])
])
display(movie_collection_widget)

```



モデル ▼ 作成日時：2023-08-05 13:54:50

dataset ▼ task ▼

動画名

ログ
2: ./model_trt/result.pthの読込に成功しました。
1: ./model_trt/result.pthの読込を実行します(初回は時間がかかります)。

[]:

[]:

[]:

動画にしたい画像データを選択します。taskはcameraかrunを選びます。cameraは、11_record_camera.ipynbで録画したファイル群、runは04_road_following.ipynbで録画したファイル群になります。

```
display(movie_collection_widget)
```



モデル ▼ 作成日時：2023-08-05 13:54:50

dataset ▼ task ▼

動画名

ログ

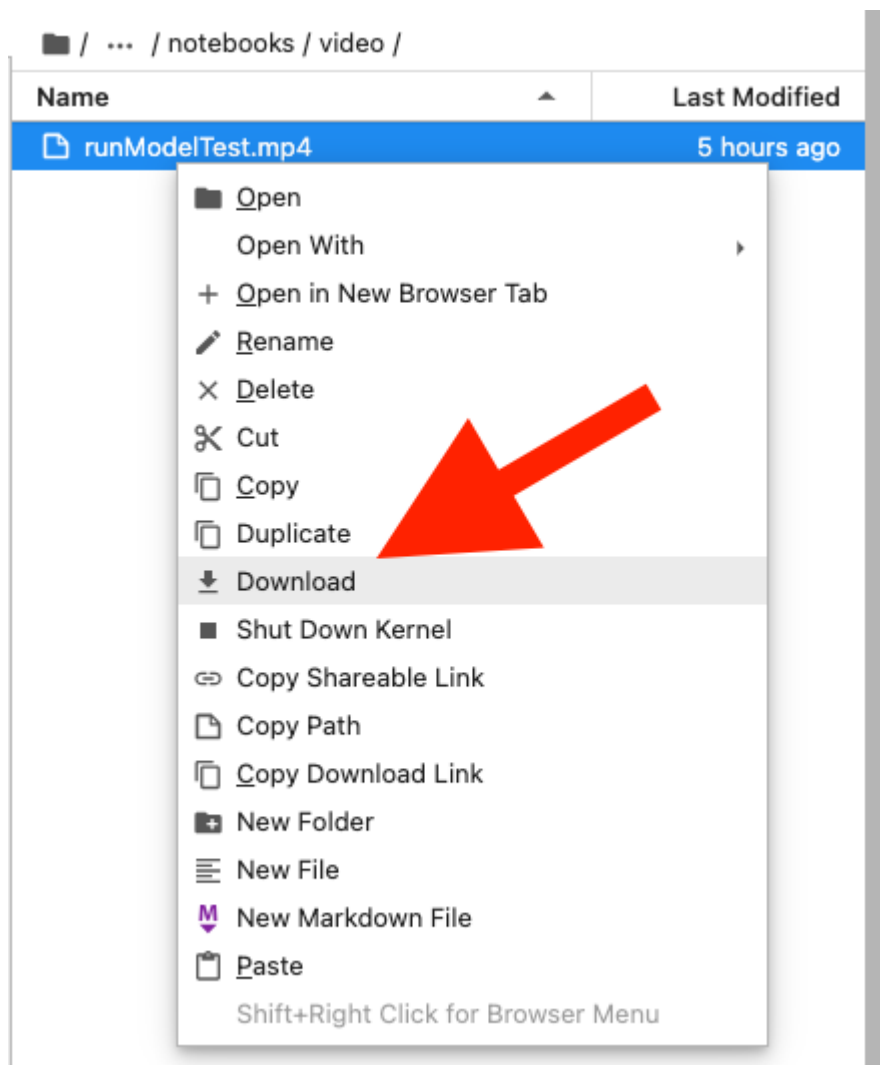
動画名に名前を付けます。

モデル	/model_trt/result.pth	作成日時: 2023-08-05 13:54:50	Load
dataset	recRC	task	camera
動画名	runModelTest	Make movie	

Make Movieボタンをクリックします。Videoディレクトリが生成されmp4ファイルができあがります。

camera	4 hours ago
interactive	a day ago
model	4 hours ago
model_trt	5 hours ago
video	5 hours ago
zip	a day ago
01_find_pwm.ipynb	a day ago
02_interactive_regression.ipynb	5 hours ago
03_optimize_model.ipynb	5 hours ago
04_road_following.ipynb	a day ago
05_road_following_live.ipynb	a day ago
10_find_raw.ipynb	4 hours ago

ファイルを右クリックしてお使いのパソコンにダウンロードし、動画を確認します。



青は推論結果、緑はアノテーションになります。

3.2.11 アノテーション (remark)



road_following_liveおよび、road_followingの自動走行でうまくいかない場合は、再度、interactive_regressionでデータセットの追加と転移学習を実施する方法（再学習）がございいますが、remarkは、作成済みのモデルに推論させ録画済みのデータやアノテーションした画像データを一枚一枚検証し、アノテーションを変更、追加、スキップできます。アノテーションを追加または修正した画像で一から学習させます。画像データを結合したりできるので便利です。

まず、12_remark.ipynbを開いて

The screenshot shows the JupyterLab interface. On the left, the file explorer displays a list of files and notebooks in the '/jetracer / notebooks /' directory. The file '12_remark.ipynb' is highlighted. The main editor area shows the content of '12_remark.ipynb', which is titled 'アノテーション' (Annotation). The notebook content includes a section 'ボードの識別' (Board Identification) with two code cells. Cell [1] contains code to identify the Jetson board (NANO, AGX Xavier, or NX) using GPIO pins. Cell [2] contains code to import various libraries for image processing and data handling, including matplotlib, jupyter_clickable_image_widget, and torchvision.



Danger

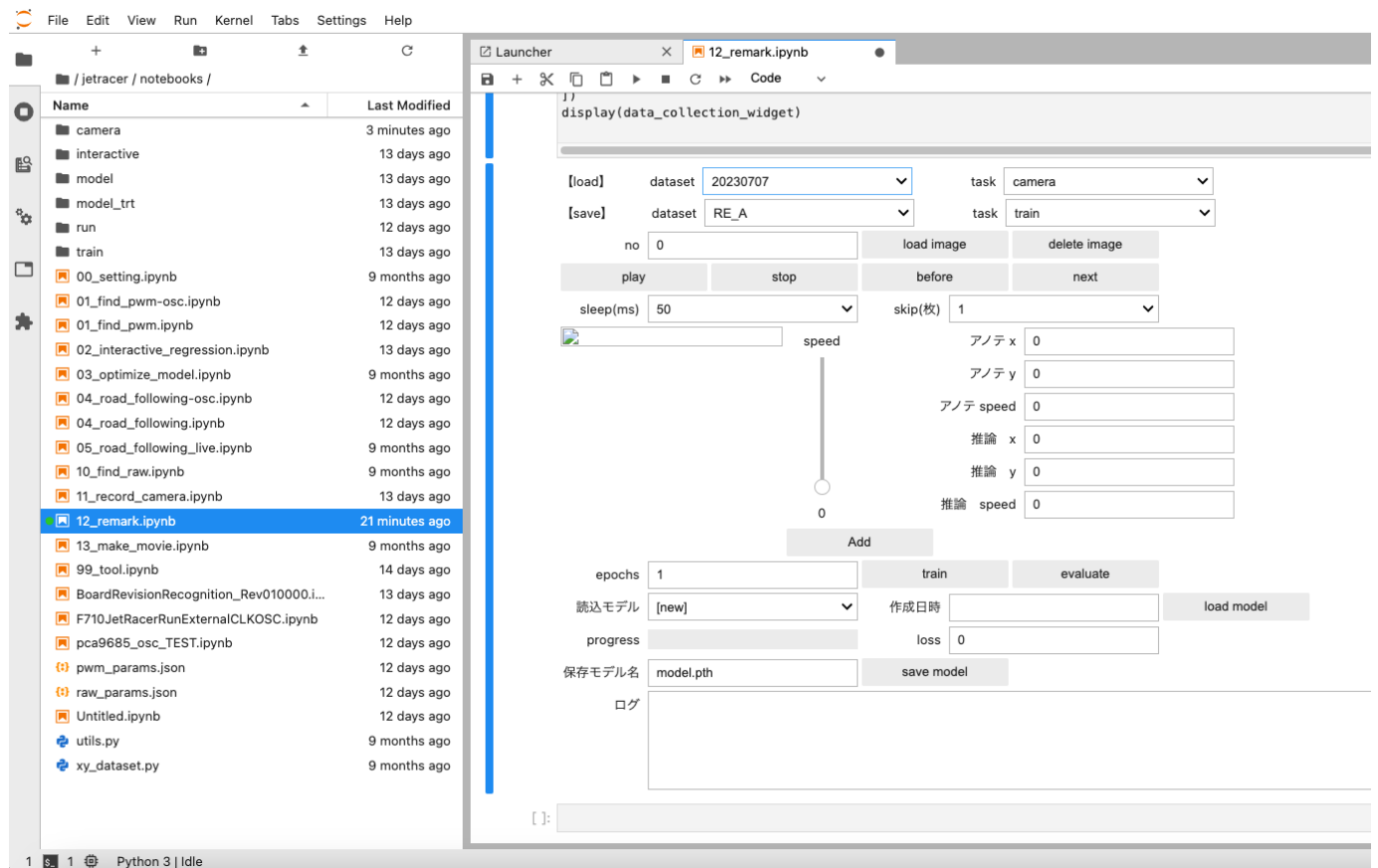
1 コマツ実際のステアリングやスロットルの動きを実機のRCカーで再現しますRCのAIモードにすると動き出すので作業台に置いて必ず実行してください。使用しないときはRCカーの電源をOFFにするか、RCモードでご使用ください。



Warning

実行には、01_find_pwm.ipynbで生成したpwm_params.jsonファイルが必要になります。

セルを全て実行すると、下の画面が現れます。



データセットの入出力

road_followingやrecord_cameraで録画した画像を呼び出します。

record_cameraで録画しcameraフォルダを指定した場合は、loadの項目にあるtaskをcameraを選択、road_followingで録画した場合はrunを選択します。datasetは編集したいフォルダ名を選択します。

The screenshot shows the JupyterLab interface with the '12_remark.ipynb' notebook open. The left sidebar lists various files and notebooks, including 'camera', 'interactive', 'model', 'model_trt', 'run', 'train', and several IPYNB files. The main area displays the notebook content, which includes a code cell with the following code:

```
display(data_collection_widget)
```

Below the code cell, a widget interface is shown. It features a dropdown menu for 'dataset' (set to '20230707') and a dropdown menu for 'task' (set to 'camera'). There are buttons for 'load image', 'delete image', 'play', 'stop', 'before', and 'next'. A 'sleep(ms)' slider is set to 50, and a 'skip(枚)' dropdown is set to 1. Below these are input fields for 'speed' (set to 0) and 'epochs' (set to 1). There are also input fields for 'read model' (set to '[new]'), 'progress' (set to 0), 'save model name' (set to 'model.pth'), and 'log'. The bottom status bar indicates 'Python 3 | Idle'.

load imageをボタンを押して画像をロードします。ロードが開始されるとBusyになり、読み込みが完了するとIdleになります。（枚数によっては時間がかかります）時系列で画像が表示されます。

The screenshot shows the JupyterLab interface. On the left, the file explorer displays a list of files and folders under the path `/jetracer / notebooks /`. The file `12_remark.ipynb` is highlighted. On the right, the notebook `12_remark.ipynb` is open, showing a `display(data_collection_widget)` widget. The widget includes controls for loading and saving data, a video player, and a table of annotations. A red arrow points to the `12_remark.ipynb` file in the file explorer, and another red arrow points to the `before` button in the notebook interface.

Name	Last Modified
camera	5 minutes ago
interactive	13 days ago
model	13 days ago
model_trt	13 days ago
run	12 days ago
train	13 days ago
00_setting.ipynb	9 months ago
01_find_pwm-osc.ipynb	12 days ago
01_find_pwm.ipynb	12 days ago
02_interactive_regression.ipynb	13 days ago
03_optimize_model.ipynb	9 months ago
04_road_following-osc.ipynb	12 days ago
04_road_following.ipynb	12 days ago
05_road_following_live.ipynb	9 months ago
10_find_raw.ipynb	9 months ago
11_record_camera.ipynb	13 days ago
12_remark.ipynb	seconds ago
13_make_movie.ipynb	9 months ago
99_tool.ipynb	14 days ago
BoardRevisionRecognition_Rev010000.i...	13 days ago
F710JetRacerRunExternalCLKOSC.ipynb	12 days ago
pca9685_osc_TEST.ipynb	12 days ago
pwm_params.json	12 days ago
raw_params.json	12 days ago
Untitled.ipynb	12 days ago
utils.py	9 months ago
xy_dataset.py	9 months ago

The notebook interface shows the following controls and data:

- Dataset Selection:** [load] dataset: 20230707, task: camera; [save] dataset: RE_A, task: camera.
- Buttons:** load image, delete image, play, stop, before, next.
- Parameters:** no: 0, sleep(ms): 50, skip(枚): 1.
- Video Player:** A video frame showing a road scene with a yellow car and a green circle.
- Annotations Table:**

skip(枚)	アノテ x	アノテ y	アノテ speed	推論 x	推論 y	推論 speed
127	112	0	160	110	87	
- Buttons:** Add, train, evaluate, load model.
- Progress:** epochs: 1, 読込モデル: [new], 作成日時: , loss: 0.
- Save Model:** 保存モデル名: model.pth, save model.
- Log:**

```

5: 0枚目の./camera/20230707/xy/127_112_00241.jpgを読み込みました。
4: 1枚目の./camera/20230707/xy/83_112_00180.jpgを読み込みました。
3: 2枚目の./camera/20230707/xy/149_112_01289.jpgを読み込みました。
2: 1枚目の./camera/20230707/xy/83_112_00180.jpgを読み込みました。
1: 0枚目の./camera/20230707/xy/127_112_00241.jpgを読み込みました。

```

すでに作成済みモデルがある場合(modelフォルダ内にあるファイル)は、読み込みモデルのドロップダウンリストから選択し、load modelをボタンを押します。画像を推論するためevaluateボタンを押します。(通常、数秒間で実行)

しばらく経って、Playボタンを押すと画像が次々と表示されます。表示が速い場合は、sleep(ms)のドロップダウンリストで一枚表示の時間を選べます。

画像をスキップしたい場合は、1～5枚の範囲でスキップできます。(スキップしない場合は1枚を選択します。)

stopボタンでPlayを中断、beforeで前の画像へ、nextで次の画像に移動できます。noは画像の番号になります。

【load】 dataset 20230707 task camera
 【save】 dataset RE_A task camera

no 34 load image delete image
 play stop before next

sleep(ms) 50 skip(枚) 1

speed 128

アノテ x 125
 アノテ y 112
 アノテ speed 0
 推論 x 156
 推論 y 109
 推論 speed 91

Add



再アノテーションした画像を保存する場所を選択します。saveにあるデータセット(RE_A,RE_B,RE_C,RE_Dのいずれか),taskは通常trainまたはcameraを選択します。

【load】 dataset 20230707 task camera
 【save】 dataset RE_A task camera

no 34 load image delete image
 play stop before next

sleep(ms) 50 skip(枚) 1

speed 128

アノテ x 125
 アノテ y 112
 アノテ speed 0
 推論 x 156
 推論 y 109
 推論 speed 91

Add



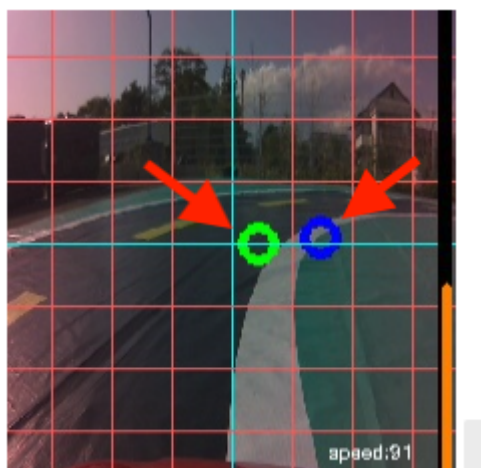
推論結果と実行結果を検証して、アノテーション修正や画像をスキップしていきます。画像をクリックするとxyフォルダにデータが保存されます。

ステアリング値のアノテーション

緑色のマーカーがステアリング走行(xy)のデータ（既存のアノテーション）、青色のマーカーがステアリングの推論データとなります。下の画面内をクリックして再びアノテーションします。

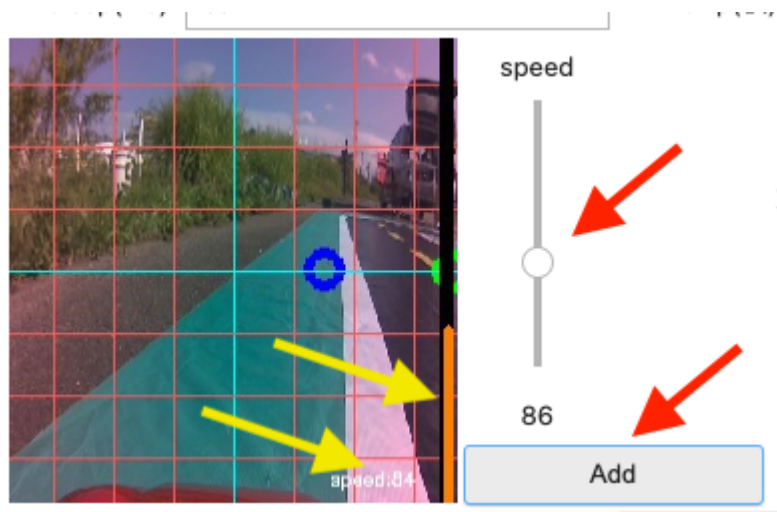
RE_Aの直下（例）のxyフォルダに画像データが保存されます。

※画面サイズは縦横2 2 4ドットであり、中央値は1 1 2となります。原点は左上。



保存されましたら、nextボタンを押して次の画像をアノテーション作業をします。

スロットル値のアノテーション



また、スロットル量のアノテーションを追加したい場合は、speedのスライダーを変化させAddボタンを押します。RE_Aの直下（例）のspeedフォルダに画像データが保存されます。

※黄色い矢印はスロットル量の推論値です。（オレンジの縦のバー0～2 2 4）

nextボタンを押して次のアノテーションを行い、

全てのアノテーション作業が終了したのなら、saveにあるこれまでの作業で行われた蓄積したデータセットを最初から学習させます。

学習

任意のepochs（学習数）※のテキストボックスに入力してtrainボタンを押します。

[load]	dataset	20230707	task	camera
[save]	dataset	RE_A	task	train
no	34	load image	delete image	
play	stop	before	next	
sleep(ms)	50	skip(枚)	1	



speed

128

125

112

0

156

109

91

epochs	30	train	evaluate
読みモデル	./model/model202306291059.pt	作成日時	2023-06-29 10:59:44
progress		loss	0.04384587239474058
保存モデル名	model.pth	save model	

ログ

74: 34枚目の./camera/20230707/xy/125_112_00507.jpgを読込みました。

73: 33枚目の./camera/20230707/xy/215_112_01684.jpgを読込みました。

72: [camera/RE_A]のspeedカテゴリにデータを追加しました。

71: 34枚目の./camera/20230707/xy/125_112_00507.jpgを読込みました。

70: [camera/RE_A]のspeedカテゴリにデータを追加しました。

しばらくすると学習が開始され、progressとlossが表示されます。学習が止まってしまう場合は、画像ファイルが空の可能性がございます。

壊れている画像ファイルを取り除いてください。

空のファイルを見つける。

```
1 sudo find ./ -empty
```

空のファイルの削除する。

```
1 sudo find ./ -empty -delete
```

[詳しくはFaBoブログをご覧ください](#)

※lossが十分に低くかつ学習時間なるべく短くなるようにepochsを決めてください。

学習が終了したら、任意の保存モデル名に変更しsave_modelボタンを押します。

保存されたモデルはmodelフォルダに保存されます。

epochs	0	train	evaluate	
読込モデル	./model/model202306291059.pt ▼	作成日時	2023-06-29 10:59:44	load model
progress		loss	0.004278683758996151	
保存モデル名	model.pth	save model		

ログ

3.2.12 環境情報判断



3.2.13 本ページ、チュートリアル工事中

ここでは、1つのカメラからカメラの画像による場所を推論します。

Jetson Orin Nano環境で行いましょう。

20_detect.ipynbを開きます。

もとのコードを以下のようにカテゴリ名を変更します。

例：Sector1から9そしてetに変更いたします。

```
WIDTH = 80
HEIGHT = 80

LOAD_DATASETS = []
LOAD_TASK = ['camera', 'dataset']

#CATEGORIES = ["mall", "diner", "bank", "school", "hotel", "supermarket", "hospital", "campus", "etc"]
CATEGORIES = ["Sector1", "Sector2", "Sector3", "Sector4", "Sector5", "Sector6", "Sector7", "Sector8", "etc"]
```

Jetson Orin Nanoをインターネットに繋ぎ、ダウンロードします。

▼ dataset.pyの取得

<https://github.com/sangyy/jetson-dlinano/tree/master/classification>

のページよりdataset.pyをダウンロードし、JetRacerのJupyterlabにアップロードします。下記コマンドでも取得可能です。ダウンロードが完了したら、JupyterLabのNotebookをShutdownし、もう一度開き直します。

```
[5]: import os
file_path = 'dataset.py'
if not os.path.exists(file_path):
    !wget https://raw.githubusercontent.com/sangyy/jetson-dlinano/master/classification/dataset.py

--2024-09-11 08:34:47-- https://raw.githubusercontent.com/sangyy/jetson-dlinano/master/classification/dataset.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2067 (2.0K) [text/plain]
Saving to: 'dataset.py'

dataset.py      100%[=====]  2.02K  --.-KB/s   in 0s

2024-09-11 08:34:47 (5.22 MB/s) - 'dataset.py' saved [2067/2067]
```

モデルはNewを選択して読み込みます。既存のモデルがある場合は、モデルを読み込みます。その場合は、スライダーによって各画像に対しての推論結果が表示されます。

■ 1.モデル読み込み

分類のモデルを読み込みます。

読み込みモデル

[new]

 作成日時

load model

ログ

3: [new]が選択されたのでresnet18の最新の重みから始めます。
2: /nvdli-nano/jetracer/notebooks/detectが存在していません。
1: change_save_dataset

学習に使用するアノテーション後のデータセットのフォルダ名を設定します。作成したフォルダはdetectフォルダの直下に作成されます。もうすでにデータセットがある場合はdatasetのドロップダウンリストから選択します。

■ 3.データセット保存

指定したデータセットに保存します。

dataset

direction202409071932

 保存task

detect

datasetの新規作成

Name

direction202409071932

Create dataset

ログ

8: change_save_dataset
7: Datasetを作成しました：direction202409071932
6: name: direction202409071932
5: direction202409071932を生成します。
4: camera/direction202409071932 cam0を呼び込み(30秒〜1分以上かかります)

画像が表示され、特定の場所をそれぞれアノテーションしていきます。▶ボタンで再生、>コマ送り、<コマ戻しになります。アノテーションは、追加ボタンを押します。画像を追加した数が表示されます。アノテーションした画像はdetectフォルダ直下の各フォルダへ保存されます。

■ 4.アノテーション

アノテーションを実施。

▶

■

 < >

sleep(ms)

50

 skip(枚)

2

no

220

更新

削除

Sector1

0.10

0

追加

Sector2

0.10

0

追加

Sector3

0.10

0

追加

Sector4

0.10

0

追加

Sector5

0.20

0

追加

Sector6

0.10

0

追加

Sector7

0.10

0

追加

Sector8

0.10

0

追加

etc

0.10

0

追加

ログ

13: 70枚目を読みました。
12: 60枚目を読みました。
11: 50枚目を読みました。
10: 40枚目を読みました。
9: 30枚目を読みました。

任意のエポック数で学習します。学習が終了しましたら、モデル名をつけて保存します。

■ 5.学習

転移学習を実施。

epochs	1
progress	
loss	0
accuracy	0
train evaluate	
保存モデル名	model.pth 保存
ログ	13: 70枚目を読みました。 12: 60枚目を読みました。 11: 50枚目を読みました。 10: 40枚目を読みました。 9: 30枚目を読みました。

できたモデルがうまく推論しているかどうかを確認します。ビデオファイルが作成され、右クリックで動画をダウンロードしてPC上で確認します。動画の左上に推論した各確率が表示されます。

■ 6.評価動画

評価動画を作成し、目視確認します。

動画名	run_detect_video	skip(枚)	2 ▼	動画の作成
ログ	13: 70枚目を読みました。 12: 60枚目を読みました。 11: 50枚目を読みました。 10: 40枚目を読みました。 9: 30枚目を読みました。			

学習結果が良ければ、TensorRTへ変換して推論時間を短縮します。20_detect.ipynbを閉じてシャットダウンし21_convert_detect.ipynbへ進みましょう。

21_convert_detect.ipynbを実行して、20_detect.ipynbで生成されたモデルをTensor RTへ変換します。

次に、22_make_movie.ipynbでは環境モデルの動作を撮影画像で確かめます。

さらに、23_run_detect.ipynbは実際走行してみて、環境モデルにより周回カウントがカウントされているのかを実際の走行によって確かめます。

確かめるために、24_make_dual_movie.ipynbでは走行モデルと環境モデルの動作を撮影画像で確かめます。

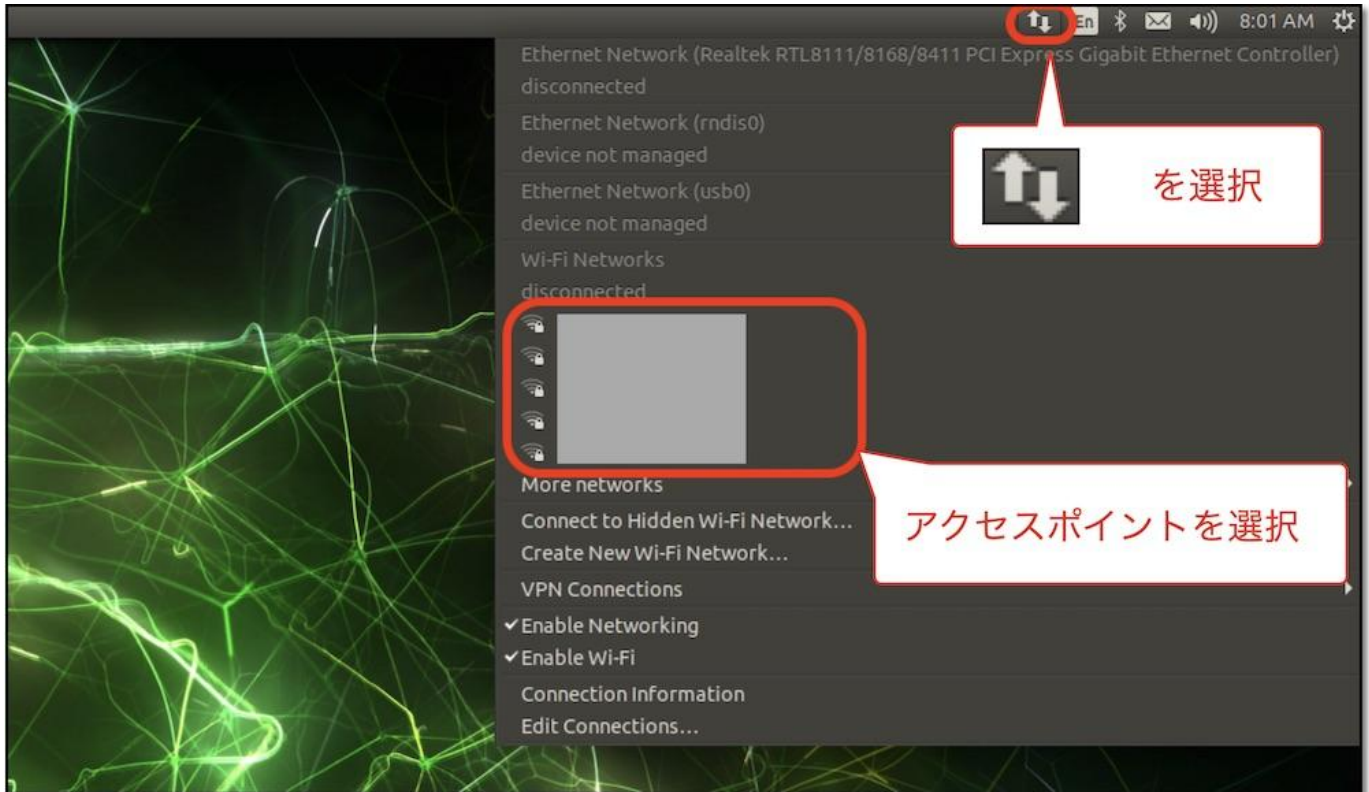
25_run_dual_detect.ipynbは、Aモデル、Bモデルの走行モデルを奇数、偶数の周回数によってモデルを使い分けて推論走行させます。

3.3 設定

3.3.1 Wi-Fiの設定

Wi-Fiの設定(GUI)

インターネットに接続可能なWi-Fiアクセスポイントに接続します。



Wi-Fiの設定(コマンド)

コマンド(`nmcli`)でWi-Fiを設定することもできます。ターミナルから操作します。

アクセスポイントのスキャン

周辺のWi-Fiアクセスポイントをスキャンして、一覧を表示します。

```
1 sudo nmcli device wifi rescan
2 nmcli device wifi list
```

SSID、電波強度(SIGNAL)、セキュリティ(SEcurity)の一覧が表示されるので、接続したいアクセスポイントのSSIDを確認します。

SSIDとPASSWORDを指定して接続

SSIDとPasswordを指定して接続します。`MySSID` と `MyPassword` は、接続するアクセスポイントのSSIDとPasswordに置き換えてください。

```
1 sudo nmcli device wifi connect 'MySSID' password 'MyPassword' ifname wlan0
```

接続の確認

接続状態を確認します。`wlan0` が `connected` になっていれば接続完了です。

```
1 nmcli device status
```

3.3.2 Windowsの設定

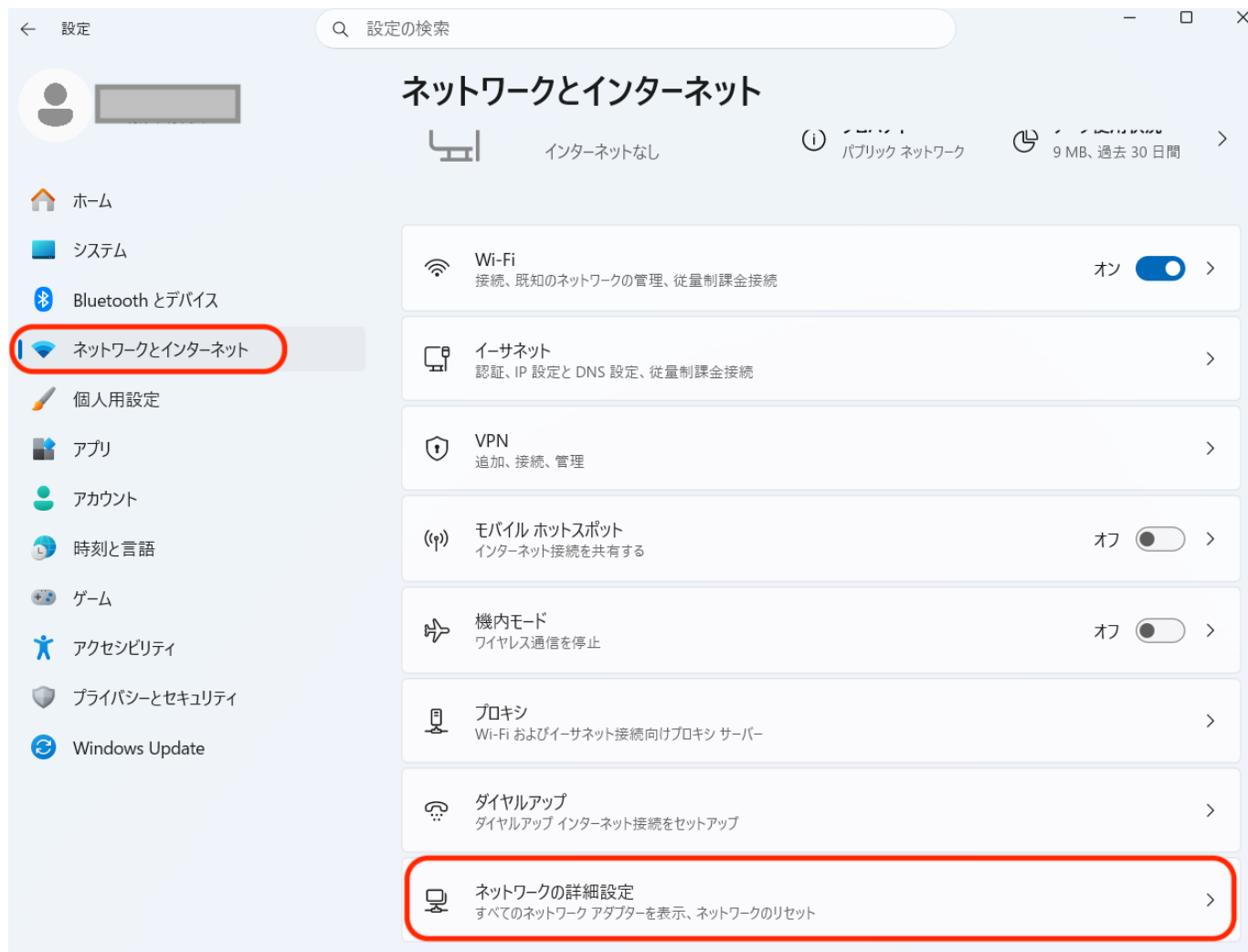
問題の整理

WindowsOSでは、JetsonとWindowsを接続しても、IPアドレスが自動で割り振られないケースが多いです。そのため、手動で割り振ります。

設定方法

JetsonとWindowsをUSB Typeで接続してください。

[Windows]-[設定]-[ネットワーク&インターネット]を選びます。



イーサネット接続の中のUsbNcm Host Deviceを探します。

← 設定

Q 設定の検索

ネットワークとインターネット > ネットワークの詳細設定

ネットワーク アダプター

Wi-Fi 未接続 Intel(R) Wireless-AC 9560 160MHz	無効にする	▼
Bluetooth ネットワーク接続 Bluetooth Device (Personal Area Network)	無効にする	▼
イーサネット 未接続 Realtek PCIe GbE Family Controller	無効にする	▼
イーサネット 3 識別されていないネットワーク <u>UsbNcm Host Device</u>	無効にする	▼
イーサネット 2 識別されていないネットワーク Remote NDIS Compatible Device	無効にする	▼

その他の設定

共有の詳細設定 ネットワークの検出と共有の設定を変更する	>
データ使用状況	>
ハードウェアと接続のプロパティ	>

ホーム

システム

Bluetooth とデバイス

ネットワークとインターネット

個人用設定

アプリ

アカウント

時刻と言語

ゲーム

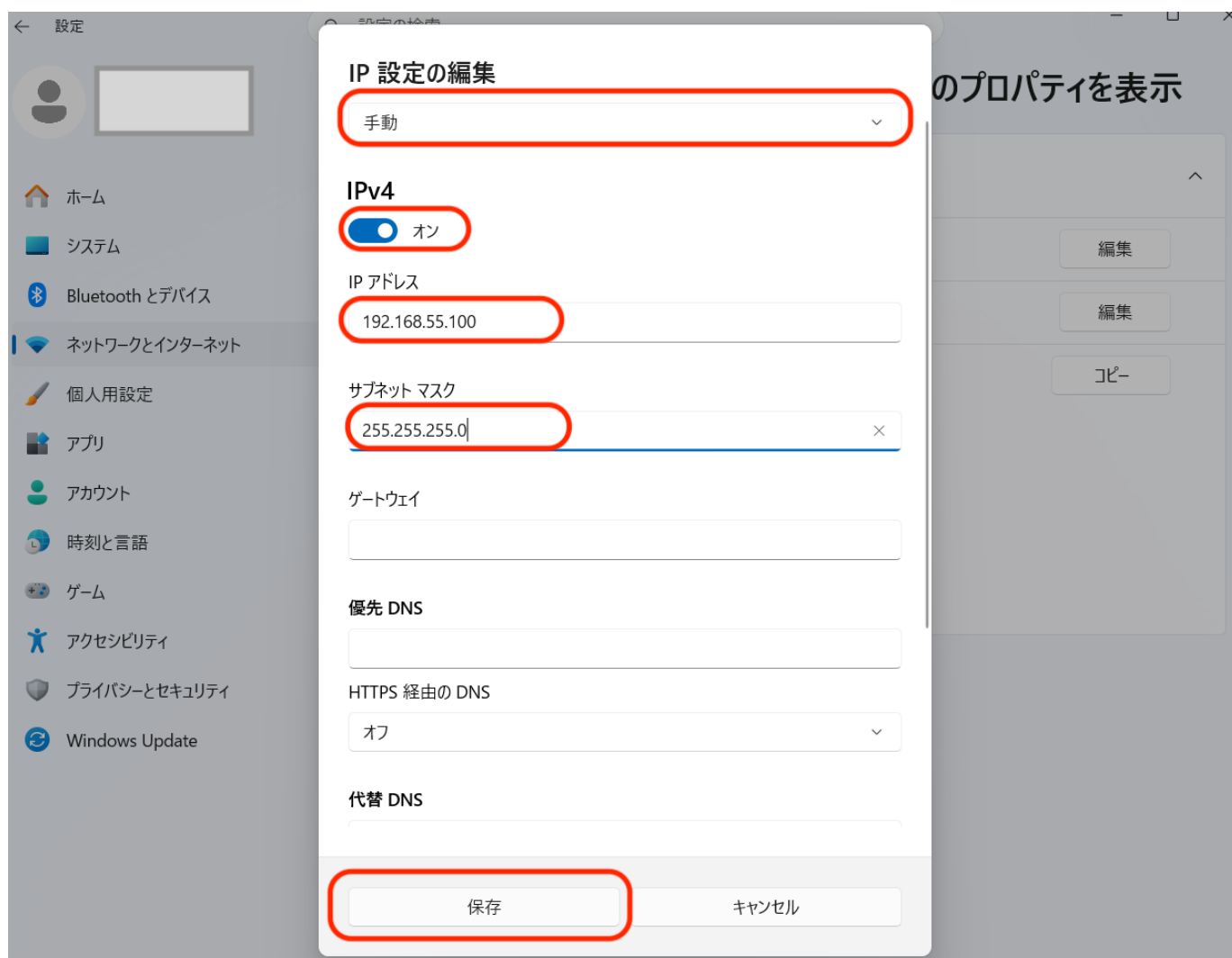
アクセシビリティ

プライバシーとセキュリティ

Windows Update



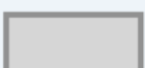
手動で、IP '192.168.55.100', ネットマスク '255.255.255.0'を固定IPアドレスとして割り振ります。Jetsonには、'192.168.55.1'が割り振られているので、Windowsを'192.168.55.100'に固定すると通信が安定します。



← 設定

Q 設定の検索

— □ ×



ホーム

システム

Bluetooth とデバイス

ネットワークとインターネット

個人用設定

アプリ

アカウント

時刻と言語

ゲーム

アクセシビリティ

プライバシーとセキュリティ

Windows Update

… > ネットワークの詳細設定 > 追加のプロパティを表示

イーサネット 3 のプロパティ

IP 割り当て: 手動

IPv4 アドレス: 192.168.55.100

IPv4 マスク: 255.255.255.0

編集

DNS サーバーの割り当て: 自動 (DHCP)

編集

集計リンク速度 (送受信): 425/425 (Mbps)

リンク ローカル IPv6 アドレス: fe80::bf1b:41db:2bd4:8f1d%40

IPv6 DNS サーバー: fec0:0:0:ffff::1%1 (非暗号化)
fec0:0:0:ffff::2%1 (非暗号化)
fec0:0:0:ffff::3%1 (非暗号化)

IPv4 アドレス: 192.168.55.100

製造元: Microsoft

説明: UsbNcm Host Device

ドライバのバージョン: 10.0.26100.8737

物理アドレス (MAC): 12:1A:8D:B2:11:FA

コピー

ヘルプを表示

フィードバックの送信

- 102/103 -

3.4 各種Tips

3.4.1 Tips

WiFiに接続

```
1 sudo nmcli device wifi connect 'MySSID' password 'MyPassword' ifname wlan0
```

I2Cのアドレス確認

Jetson Nano

```
1 sudo i2cdetect -r -y 1
```

Jetson NX

```
1 sudo i2cdetect -r -y 8
```

カメラの認識確認

```
1 dmesg | grep imx219
```