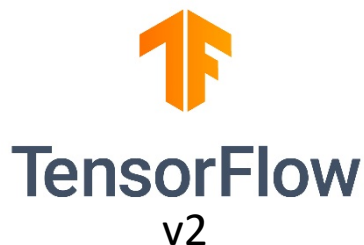


抜粋

はじめての顔・物体認識用AI開発キットTF2

～Tensorflow v2、YoloなどAI開発キット(写真、ビデオ、カメラ対応)～

開発編(グラフィック_ubuntu版)



スペクトラム・テクノロジー株式会社

<https://spectrum-tech.co.jp>

sales@spectrum-tech.co.jp



開発編 目次

	ページ
• ubuntu運用マニュアル	
1. ubuntuについて	4
2. 基本コマンド	4
3. 基本操作	6
4. 日常運用	7
• ハード・ソフトウェア概要	
1. ハード概要	9
2. ソフト概要	
① ソフト一覧	10
② 機能一覧	11
③ プロトコルスタック	12
• Tensorflow	13
• Tensorflow AI概要	
1. 開発キット全体像	14
2. 開発プログラム一覧	15
• Tensorflow開発	
1. Object detection	
① カメラ(物体検出)	17
② 写真(物体検出)	18
③ ペットモデル作成手順	19
④ トランプモデル作成手順	25
⑤ 写真(信号機、ヘルメット検出)	31
⑥ セグメンテーション(写真、カメラ)	33
⑦ セグメンテーション風船(写真、カメラ)	37
⑧ アノテーション(Labellmg、Roboflow)	40

抜粋版のため、本文とページ
は一致しません

開発編 目次

	ページ
• Yolo	44
• Yolo AI概要	
1. 開発キット全体像	45
2. 開発プログラム一覧	46
• Yolo開発	
1. Yolo	
① 物体検出(写真)	47
② 物体検出(ビデオ)	48
③ 物体検出(カメラ)	49
2. Yoloモデル作成手順	50
3. Yolo顔認識例1	54
4. Yolo顔認識例2	57
• Opencv開発	
1. Opencv顔検出	58
2. Opencv顔認識	59
• 顔認識、物体認識 応用例	60
1. 居眠り運転検出事例	61
2. ポーズ検出事例	64

抜粋版のため、本文と
ページは一致しません

Ubuntu運用マニュアル

1. Ubuntuについて

Linuxの中でも一番シェアの高いOSです。2004年にDebian系から派生。

2. Linux基本コマンド

① システム関係

- 起動: 電源を入れると自動で起動します。
- 再起動: `$ reboot`
又は、左上のメニューの「ゲストを再起動」
- 終了: `$ shutdown`
又は、左上のメニューの「ゲストをシャットダウン」
- ログアウト `$ exit`
ルートからログアウトします
- **日本語／英語の入力切替**: 半角／全角のボタン(ESCボタンの下)

Ubuntu運用マニュアル

2. Linux基本コマンド

② ディレクトリ操作、コピー、移動、削除

masa@ubuntu:~\$ **cd** /home/masa/Documents

ディレクトリの切り替え

masa@ubuntu:~/home/masa/Documents\$ **ls** ファイルとディレクトリの表示(表示したら操作したいファイルを右クリックでコピーして操作します)

masa@ubuntu:~\$ **cp** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへコピー

masa@ubuntu:~\$ **mv** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへ移動

masa@ubuntu:~\$ **rm** ファイル名 ファイルの削除

便利な機能
てのコマンド共通(マイナスを2個とhelp)

rm -help

コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべ

③ ユーザ権限、プロセス他

masa@ubuntu:~\$ **su -**

スーパーユーザ(root)に切り替え、パスワードを入力

masa@ubuntu:~\$ **sudo**

ルート権限で各種コマンドを実施します。

masa@ubuntu:~\$ **ps** a

現状の動いているプロセスを表示

masa@ubuntu:~\$ **kill**

特定のプロセスを強制終了

masa@ubuntu:~\$ **apt-get** install pkg

パッケージのインストールなどに使用

masa@ubuntu:~\$ **date**

日付、時間の設定を行います。

masa@ubuntu:~\$ **leafpad** /etc/network/interfaces

インタフェースに記述している内容を変更します。Viよりも使いやすいです。

④ モジュール、usb、メモリ、HDDなどの表示

masa@ubuntu:~\$ **lsmod**

linuxのモジュールリスト表示

masa@ubuntu:~\$ **lsusb**

usbのデバイス表示

masa@ubuntu:~\$ **free -mt**

メモリ使用状態表示

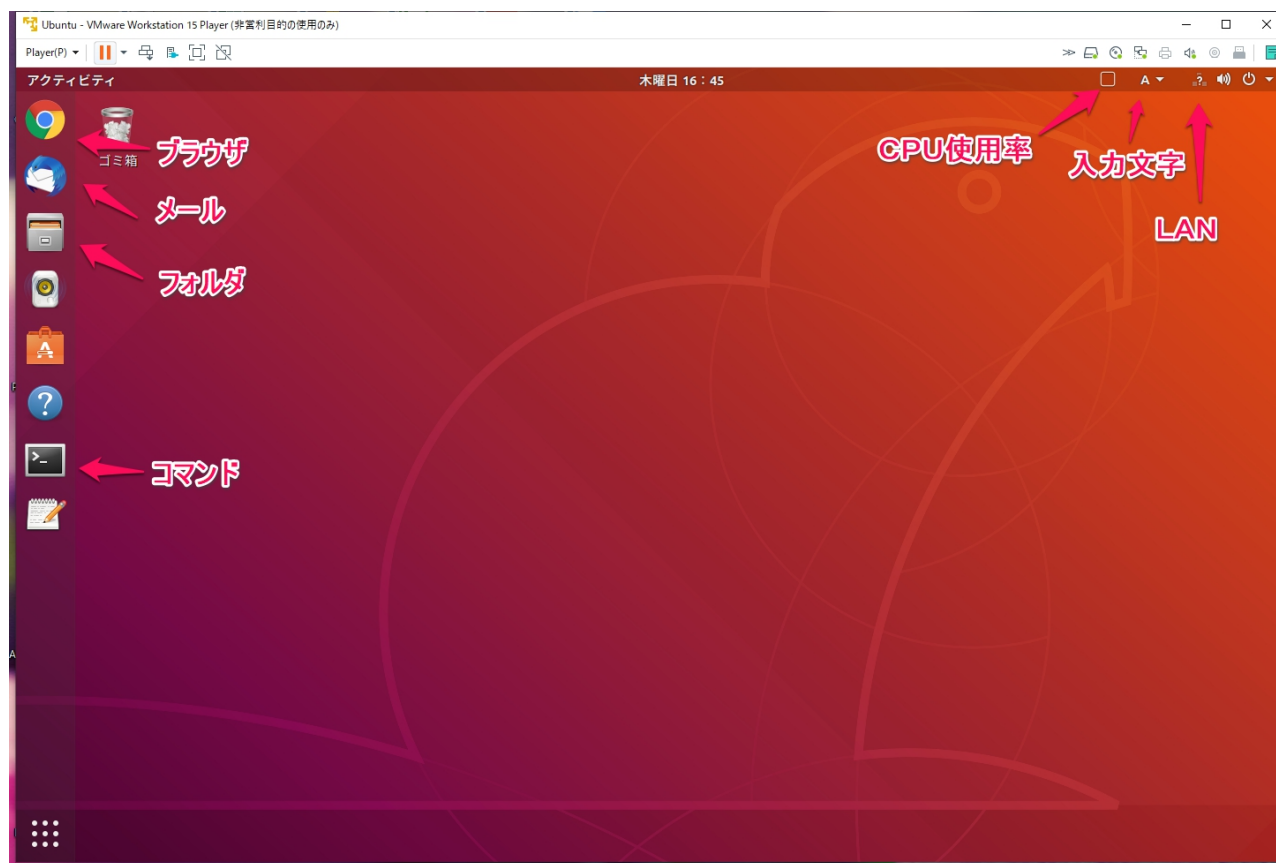
masa@ubuntu:~\$ **df**

HDD(マイクロSD)の使用状態表示

Ubuntu運用マニュアル

3. 基本操作

① 表示画面と内容



主に使用するもの

- ・ブラウザ: Chrome
- ・フォルダ: Documents内に必要なファイルがあります。
- ・コマンド: コマンド画面を立ち上げて、python3のプログラムを動作させます。

Ubuntu運用マニュアル

4. 日常運用

① セキュリティ対策(アンチウイルス更新、スキャン)

- アンチウイルス対策として無料のclamAVをインストールしてます。
- 手動での運用を基本としています。

```
masa@ubuntu: ~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
masa@ubuntu:~$ sudo freshclam  
Fri Jul 12 09:58:58 2019 -> ClamAV update process started  
2019  
Fri Jul 12 09:58:58 2019 -> ^Your ClamAV installation is O  
Fri Jul 12 09:58:58 2019 -> ^Local version: 0.100.3 Recomm  
Fri Jul 12 09:58:58 2019 -> DON'T PANIC! Read https://www.  
pgrading-clamav  
Fri Jul 12 09:58:58 2019 -> main.cvd is up to date (versio  
f-level: 60, builder: sigmgr)  
Fri Jul 12 09:58:58 2019 -> daily.cld is up to date (versi  
19, f-level: 63, builder: raynman)  
Fri Jul 12 09:58:58 2019 -> bytecode.cvd is up to date (version: 328, stgs: 94,  
f-level: 63, builder: neo)  
masa@ubuntu:~$ sudo clamscan --infected --remove --recursive
```

パターンファイル更新

```
$ sudo freshclam
```

手動スキャン時にも更新されます

手動でスキャン

```
$ sudo clamscan --infected --remove --recursive
```

自動化可能ですが、バックグラウンドで重くなる可能性大。コマンド入力後時間がかかります。

Ubuntu運用マニュアル

4. 日常運用

② インストール済パッケージの更新リスト、アップグレード

- Linuxの場合は、頻繁に更新が発生します。アップグレードを定期的 to 実施してください。
- 更新前には、バックアップを取ることをお勧めします。特にアップグレードはまれに動作不良、戻せない状態が発生します。自己責任で実施してください。

```
masa@ubuntu: ~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
^C  
masa@ubuntu:~$ sudo apt-get update  
ヒット:1 http://jp.archive.ubuntu.com/ubuntu bionic InRelease  
取得:2 http://jp.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]  
無視:3 http://dl.google.com/linux/debian1804/x86_64 InRelease  
取得:4 http://jp.archive.ubuntu.com/ubuntu bionic-security InRelease  
無視:5 https://developer.download.nvidia.com/linux-tegra/tu1804/x86_64 InRelease  
無視:6 http://developer.download.nvidia.com/linux-tegra/tu1804/x86_64 InRelease  
InRelease  
ヒット:7 https://developer.download.nvidia.com/linux-tegra/tu1804/x86_64 Release  
無視:8 http://developer.download.nvidia.com/linux-tegra/tu1804/x86_64 Release  
64 Release  
ヒット:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 mesa-va-drivers  
ヒット:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 mesa-vdpau-drivers  
ヒット:11 http://dl.google.com/linux/debian1804/x86_64 InRelease  
取得:13 http://security.ubuntu.com/ubuntu bionic-security/main amd64 firefox amd64  
取得:16 http://security.ubuntu.com/ubuntu bionic-security/main amd64 firefox amd64  
9 kB]  
取得:17 http://security.ubuntu.com/ubuntu bionic-security/main amd64 firefox amd64  
50 kB]  
取得:18 http://security.ubuntu.com/ubuntu bionic-security/main amd64 firefox amd64  
56 kB]  
masa@ubuntu:~$ sudo apt-get upgrade  
パッケージリストを読み込んでいます... 完了  
依存関係ツリーを作成しています  
状態情報を読み取っています... 完了  
アップグレードパッケージを検出しています... 完了  
以下のパッケージは保留されます:  
libgl1-mesa-dri libxatracker2 mesa-va-drivers mesa-vdpau-drivers  
以下のパッケージはアップグレードされます:  
firefox firefox-locale-en firefox-locale-ja gnome-settings-daemon  
gnome-settings-daemon-schemas libsysmetrics1 ubuntu-report  
アップグレード: 7 個、新規インストール: 0 個、削除: 0 個、保留: 4 個。  
54.4 MB のアーカイブを取得する必要があります。  
この操作後に追加で 4,019 kB のディスク容量が消費されます。  
続行しますか? [Y/n] y  
取得:1 http://jp.archive.ubuntu.com/ubuntu bionic-updates/main amd64 gnome-setti  
ngs-daemon-schemas all 3.28.1-0ubuntu1.3 [12.9 kB]  
取得:2 http://jp.archive.ubuntu.com/ubuntu bionic-updates/main amd64 gnome-setti  
ngs-daemon amd64 3.28.1-0ubuntu1.3 [316 kB]  
取得:3 http://jp.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsysmetri  
cs1 amd64 1.3.2 [1,475 kB]  
取得:4 http://security.ubuntu.com/ubuntu bionic-security/main amd64 firefox amd64  
4 68.0+build3-0ubuntu0.18.04.1 [49.8 MB]
```

更新リスト取得

\$ sudo apt-get update

アップグレード実施

\$ sudo apt-get upgrade

1. ハード概要

①開発キットと必要なハードウェア仕様

開発キットと必要なハードウェアの概要です。

品名		仕様	備考
USBメモリ(ubuntu版)		128GB USB3.0 TF2用画像認識プログラム一式	
GPU		GTX1650: HDMI, DVI	オプションで提供可能 入門用,ディスプレイへの接続は、 HDMIになります RTX30、20シリーズも対応可能
SSD		500GB(ubuntu用)、2.5inch	オプションで提供可能 GNU GRUBにより起動時にubuntu, windowsを選択可能
カメラ		HD用 USBカメラ	オプションで提供可能 リアルタイムで画像認識を行う場合
必要なハードウェア			
PC本体	cpu	i9-7900以上(第7世代以降) i7-7700以上(同上) i5-7400以上(同上)	マザーボードによっては、GPUが搭載できない場合があります。AMDも対応可能
	メモリ	16GB以上	
	PCIスロット	PCI express 3.0 x16: GPU搭載用空スロット	
	SSD	SSD空スロット	ない場合は、既存のHDDを外して搭載します

2. ソフト概要

①ソフトウェア一覧

本PCにインストールするソフトウェアの概要です。

区分	ソフト名	バージョン	備考
OS	ubuntu	18.04.3 LTS	
GPU用	cuDNN	8.1.0+cuda11.2	Nvidia用,搭載するGPUに依存
画像	Opencv	3.4.4	
	Object detection	0.1	Tensorflow 2対応
	yolo	V3	
プログラム言語	python3	3.6.9	基本的にこちらで動作します
	Node.js	V10.24	
AI用プログラム	tensorflow	2.5.0	Google、搭載GPU依存
	Tensorflow.js	1.7.4	Google
	Pytorch	1.8	Facebook
その他	Jupyter notebook、 matplotlibなど多数のpipライブラリ		

2. ソフトウェア概要

②機能一覧

本開発キットでサポートする機能一覧です。

機能	方式	Tensorflow v2	yolo	Opencv	応用事例	備考
顔認識	写真	—	○:登録も 可		○ポーズ検 出	
	ビデオ	—	○			
	カメラ(*1)	—	○	○:登録も可	○居眠り運 転検出、 ポーズ検出	
物体認識	写真	○ペット、トラ ンプ、信号機、 ヘルメット、セ グメンテーショ ン例	○	—		
	ビデオ	○ペット、トラ ンプ例	○	—		
	カメラ(*1)	○ペット、トラ ンプ、セグメン テーション例	○	—		

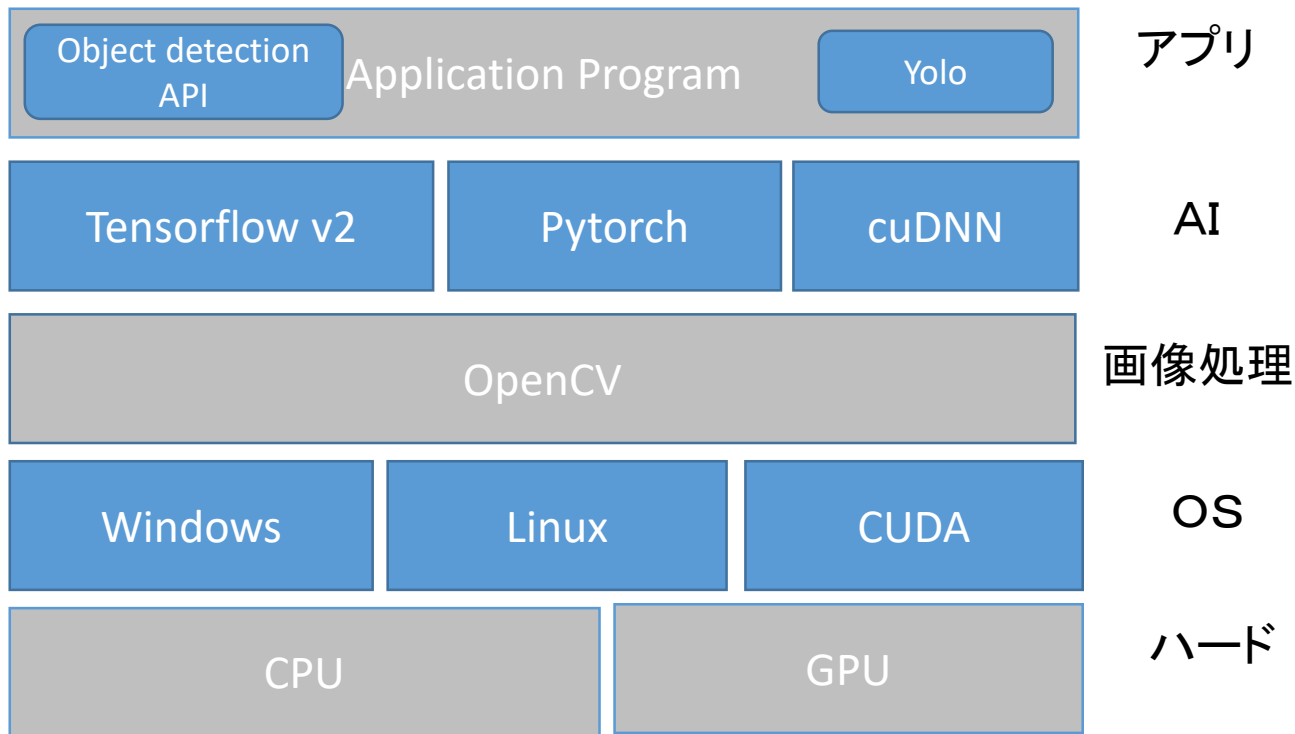
(*1)リアルタイム動画です。

2. ソフト概要

③ プロトコルスタック

各ソフトの位置付けです。

python3: プログラム言語

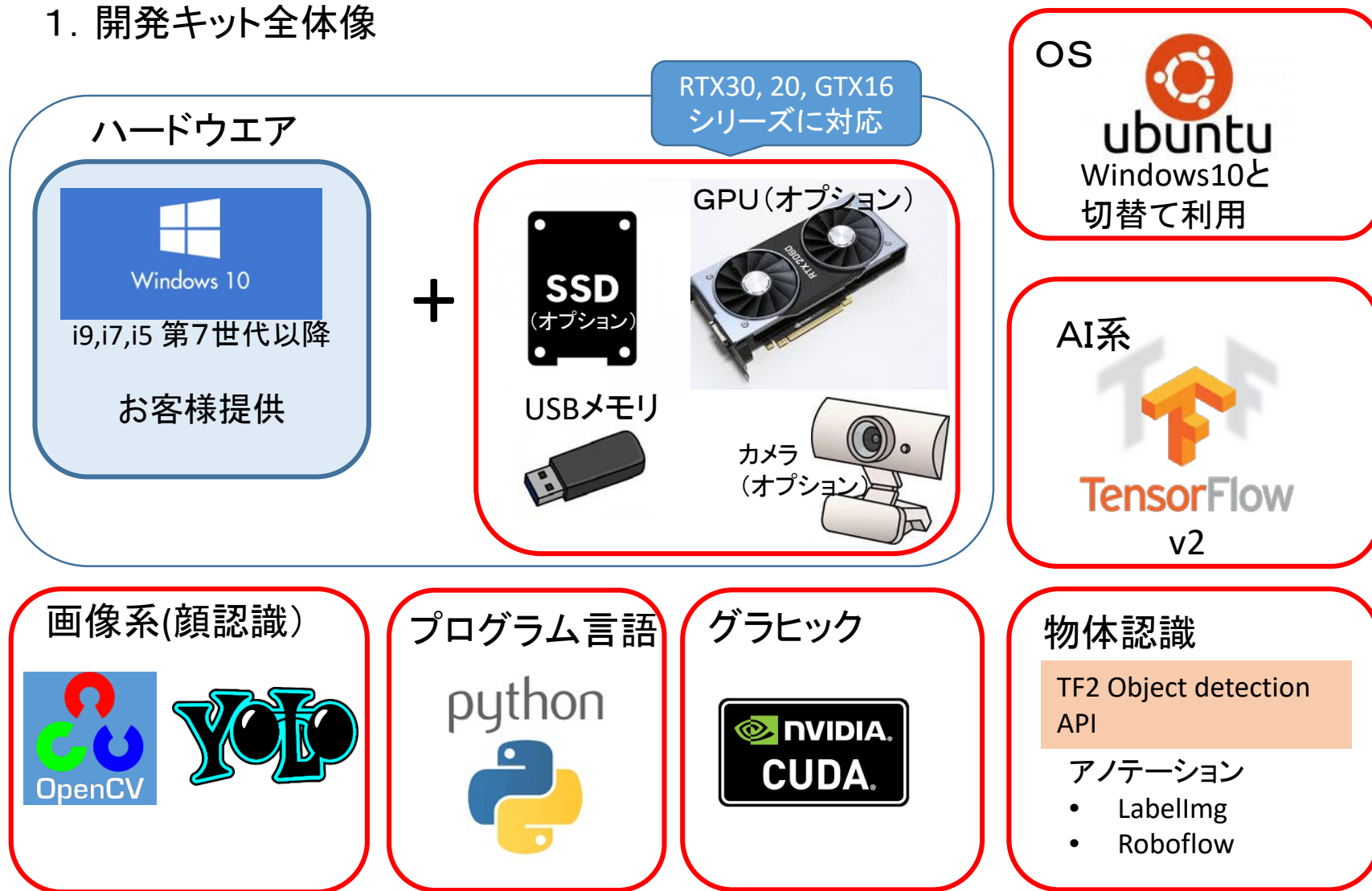


Tensorflow

- Tensorflowはグーグルが開発したAIプログラムです。ローカルで動作し、pythonで記述します。ライブラリーも豊富です。またlinux, windowsなどマルチプラットフォーム対応です。今回はそのうちの、物体認識部分を記述します。
- Tensorflow: <https://www.tensorflow.org/>
- Object detection <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/>
- Github: https://github.com/tensorflow/models/tree/master/research/object_detection

Tensorflow v2 AI概要(Ubuntu版)

1. 開発キット全体像



A:実用可能
B:チャレンジ
C:試験段階



API名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
Object detection api(tf2対 応)	Webcam1(物体検 出)	カメラから人、物を含めた90モデルの物 体を抽出。	A	90%	
	写真(物体検出)	人、物を含めた90モデルの物体を写真 から検出	A	90%	
	写真(ペット検出)	写真から登録済のペット名を検出	A	90%	37分類 、モデル作成手 順も解説
	ビデオ(ペット検 出)	ビデオから登録済のペット名を検出	A	90%	
	カメラ(ペット検出)	カメラから登録済のペット名を検出	A	90%	
	写真(トランプ検 出)	写真から登録済のトランプを検出	A	90%	6分類、モデル 作成手順も解説
	ビデオ(トランプ検 出)	ビデオから登録済のトランプを検出	A	90%	
	カメラ(トランプ検 出)	カメラから登録済のトランプを検出	A	90%	

A:実用可能
B:チャレンジ
C:試験段階

API名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
Object detection api(tf2対応)	写真(信号機検出)	信号機の3モデル(緑、黄、赤)を写真から検出、学習、エクスポート、予測を jupyter notebookで習得	A	90%	
	写真(ヘルメット検出)	ヘルメット、人などのモデルを写真から検出、学習、エクスポート、予測を jupyter notebookで習得	A	90%	
	セグメンテーション(写真)	Mask RCNNを使って、写真の物体をセグメンテーションします。形まで色付けします。	A	90%	81分類(人物、馬、花瓶など)
	セグメンテーション(カメラ)	Mask RCNNを使って、カメラ内の物体をセグメンテーションします。形まで色付けします。	A	90%	81分類(人物、馬、花瓶など)
	セグメンテーション風船(写真)	Mask RCNNを使って、写真の風船をセグメンテーションします。形まで色付けします。	A	90%	1分類、カスタマイズ可能
	セグメンテーション風船(カメラ)	Mask RCNNを使って、カメラ内の風船をセグメンテーションします。形まで色付けします。	A	90%	1分類、カスタマイズ可能
アノテーション	Labellmg(アノテーションツール)	写真にタグをつけ、アノテーションを作成するツール。	A	90%	
	Roboflow	クラウド型のアノテーション及び関連ファイル作成	A	90%	

Tensorflow開発

1. Object detection

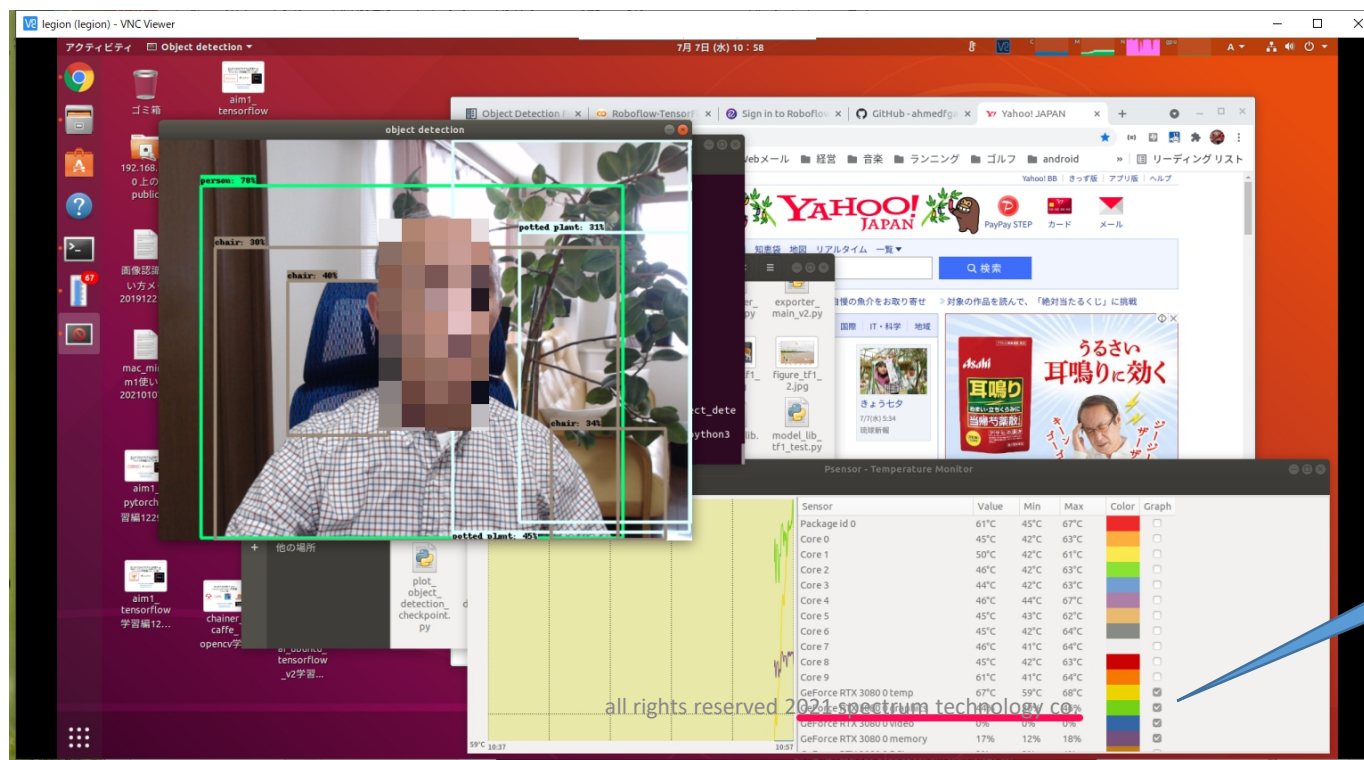
①カメラ(物体検出)

TensorFlow

コマンド

```
$ cd  
/home/masa/Documents/deeplearn/TF2/models/research/object_detection  
$ python3 object_detection_camera.py
```

- カメラを使って物体を検出する試験を行います。90の物体(人、椅子など)がすでに登録されてます。
- python3 object_detection_camera.py
- カメラの立ち上げに数分かかります。特にプログラムは修正不要です。



GPU搭載でかなり早い

Tensorflow開発

1. Object detection

②写真(物体検出)

- 写真を使って物体を検出する試験を行います。以下のモデルを使い、90の物体(人、カイトなど)がすでに登録されています。
- モデル: centernet_hg104_512x512_coco17_tpu-8
- Jupyter notebook
- TensorFlowObjectDetectionAPISample.ipynb

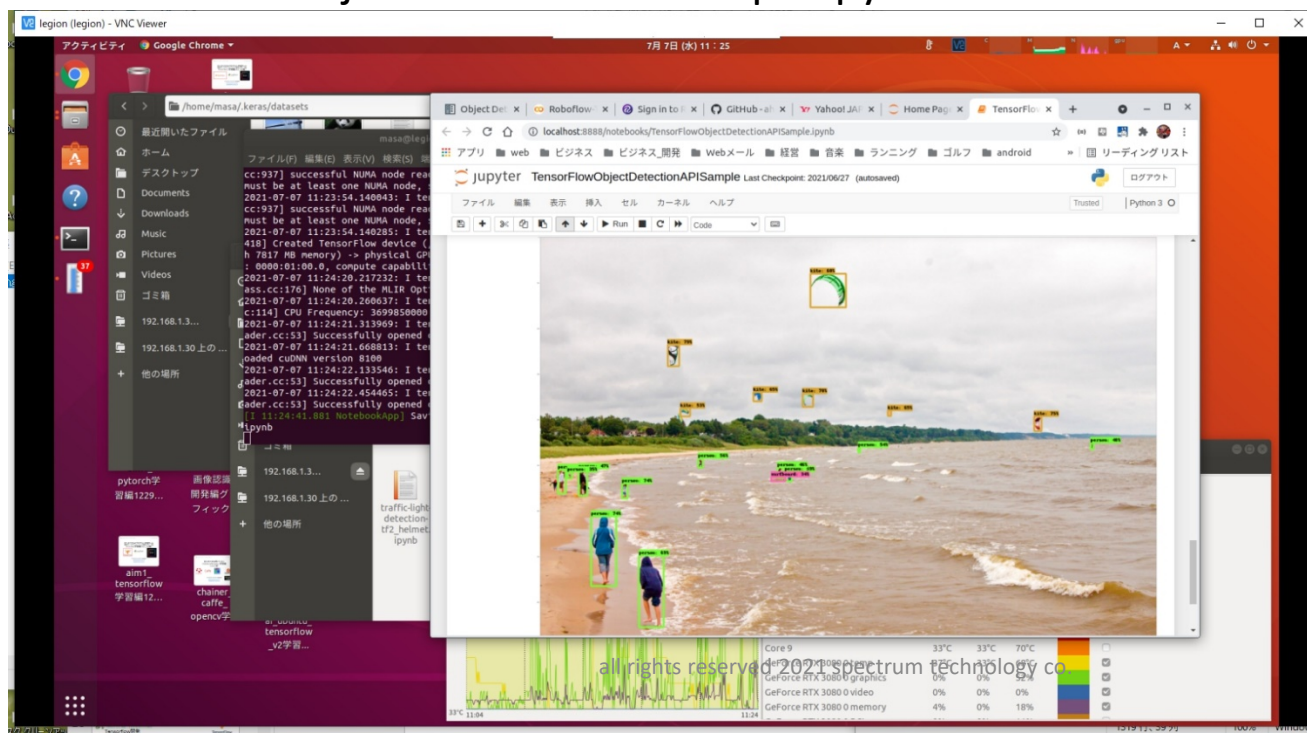
コマンド

TensorFlow

```
$ cd /home/masa/Documents/deeplearn/TF2/  
$ jupyter notebook  
$ TensorFlowObjectDetectionAPISample.ipynb
```

https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/auto_examples/plot_object_detection_saved_model.html#

上記の正式版のTF2、object detectionは、何故か検出結果がでません。不明。



③ペットモデル作成手順

\$ cd

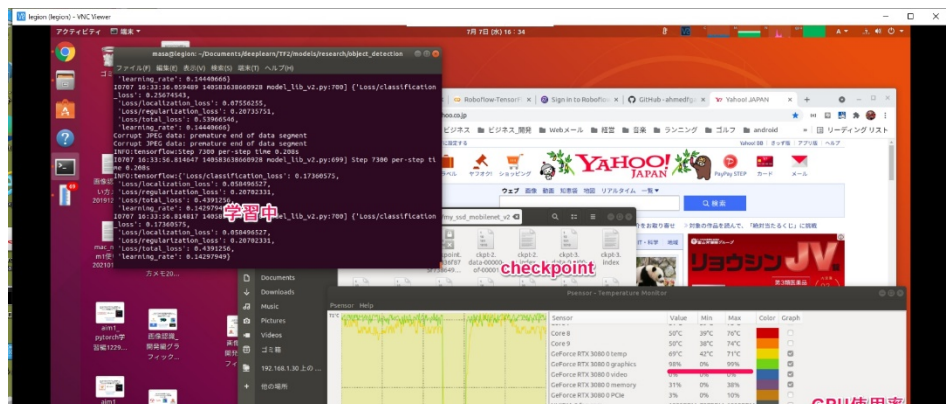
```
/home/masa/Documents/deeplearn/TF2/models/research/  
object_detection
```

- ペットのモデルを作成します。どのモデルも同じ手順です。
- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_pets.md

4. 学習させます(2万回、2時間)

```
python3 model_main_tf2.py --logtostderr --
train_dir=/home/masa/Documents/deeplearn/TF2/pet/training/ --
pipeline_config_path=/home/masa/Documents/deeplearn/TF2/pet/models/tf2/my_ssd_mobilenet
v2/pipeline.config --
model_dir=/home/masa/Documents/deeplearn/TF2/pet/models/tf2/my_ssd_mobilenet_v2
```

- ・再度学習する場合は、モデル内のcheckpoint, ckpt-x.dataxxx, ckpt-x.indexなどを削除してください。
- ・学習途中でスタックする場合は、config内のbatch sizeを16から8 or 1へ変更のこと。



Tensorflow開発

TensorFlow

1. Object detection

③ペットモデル作成手順

コマンド

```
$ cd /home/masa/Documents/deeplearn/TF2/pet
```

- ペットのモデルを作成します。どのモデルも同じ手順です。
- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_pets.md

5. 学習データをエクスポート

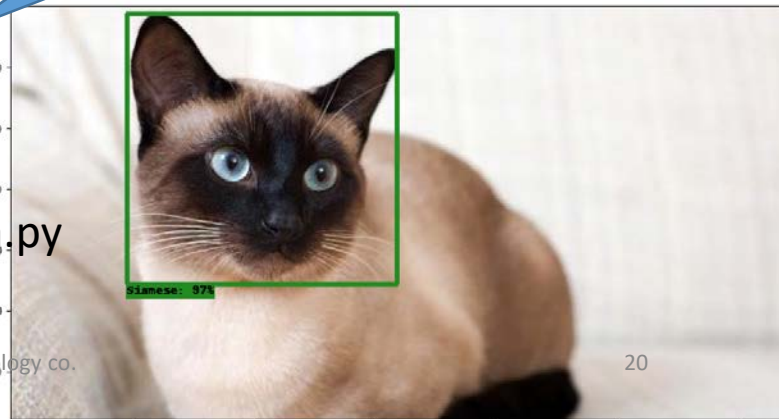
```
python3  
/home/masa/Documents/deeplearn/TF2/models/research/object_detection/exporter_main_v2.py ¥  
  --input_type image_tensor ¥  
  --pipeline_config_path=models/tf2/my_ssd_mobilenet_v2/pipeline.config ¥  
  --trained_checkpoint_dir=models/tf2/my_ssd_mobilenet_v2 ¥  
  --output_directory=exported-models
```

結果は、ポ
プアップしま
せん

6. テスト

写真: `python3 object_detect_pet_image.py`ビデオ: `python3 object_detect_pet_video.py`

(最後にエラーがでます。未解決)

カメラ: `python3 object_detection_pet_camera.py`

Tensorflow開発

1. Object detection

④トランプモデル作成手順

コマンド

```
$ cd /home/masa/Documents/deeplearn/TF2/trump
```

- トランプのモデルを作成します。どのモデルも同じ手順です。Trump配下に全てのデータがあります。

5. 学習データをエクスポート

```
python3 exporter_main_v2.py ¥  
--input_type image_tensor ¥  
--pipeline_config_path=models/tf2/my_centernet_resnet50_v1_fpn/pipeline.config ¥  
--trained_checkpoint_dir=models/tf2/my_centernet_resnet50_v1_fpn ¥  
--output_directory=exported-models
```

Exporter_main_v2.pyの原本は、
TF2/models/research/object_detectionの場所からコピーしたものです。

6. テストします

```
python3 object_detect_trump_image.py  
python3 object_detection_trump_video.py  
(最後にエラーがでます。未解決)  
python3 object_detection_trump_camera.py
```

結果は、ポップアップしません



Tensorflow開発

TensorFlow

1. Object detection

⑤信号機事例(jupyter版)

コマンド

\$ cd /home/masa/Documents/deeplearn/TF2/

\$ jupyter notebook

- 信号機の色を判定する事例です。
- Jupyter notebookで全体の流れを確認できます。
- #skipと表示してある項目は実施しないでください。
- cd /home/masa/Documents/deeplearn/TF2/
- Jupyter notebook
- ブラウザがポップアップします。
- traffic-light-detection-tf2.ipynb を選択
- 使用しているフォルダ /home/masa/Documents/deeplearn/TF2/driving-object-detection
- ディレクトリの位置はお客様の環境に合わせて変更してください。
- 学習(1万回、30分)、エクスポート、予測など一連の確認ができます。

<https://github.com/yuki678/driving-object-detection>

信号機事例のオリジナル。

The collage consists of three screenshots:

- Left Screenshot:** A terminal window showing the execution of commands to set up the environment, including installing TensorFlow and other dependencies.
- Middle Screenshot:** A Jupyter Notebook interface with the 'traffic-light-detection-tf2.ipynb' file selected in the file browser.
- Right Screenshot:** The Jupyter Notebook output, showing a plot of a traffic light image with detected bounding boxes and labels.

Tensorflow開発

1. Object detection

⑥ヘルメット事例(jupyter版)

- ヘルメットを判定する事例です。
- Jupyter notebookで全体の流れを確認できます。
- #skipと表示してある項目は実施しないでください。
- cd /home/masa/Documents/deeplearn/TF2/
- Jupyter notebook
- ブラウザがポップアップします。
- traffic-light-detection-tf2_helmet.ipynb を選択
- 使用しているフォルダ /home/masa/Documents/deeplearn/TF2/helmet
- ディレクトリの位置はお客様の環境に合わせて変更してください。
- 学習(30万回、1-2日)、エクスポート、予測など一連の確認ができます。

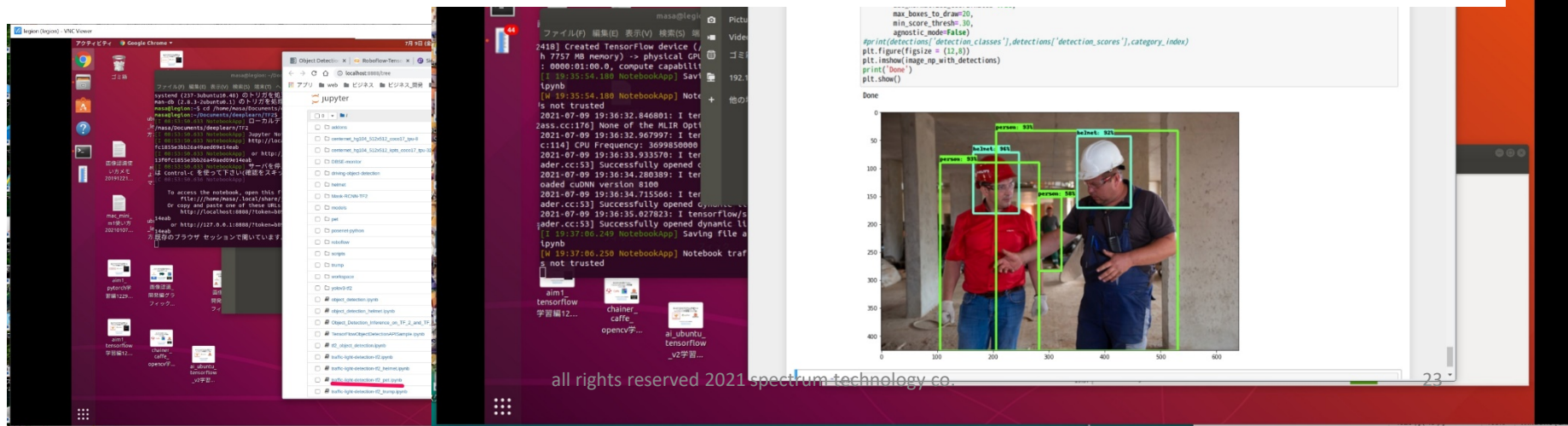
コマンド

```
$ cd /home/masa/Documents/deeplearn/TF2/
$ jupyter notebook
```

TensorFlow

<https://clear.ml/blog/construction-feat-tf2-object-detection-api/>

上記のオリジナルは、最終の予測boxが表示されない。不明。



Tensorflow開発

1. Object detection

⑦セグメンテーション事例(写真)

- Mask RCNNを使ったセグメンテーション事例です。
- `cd /home/masa/Documents/deeplearn/TF2/Mask-RCNN-TF2/`
- `python3 mask-rcnn-prediction.py`
- モデル: 学習済のmask_rcnn_coco.h5使用。81分類(人物、馬、花瓶など)
- ディレクトリの位置, 写真はお客様の環境に合わせて変更してください。
- 検出した物体には、該当の場所をマスクして識別します。

コマンド

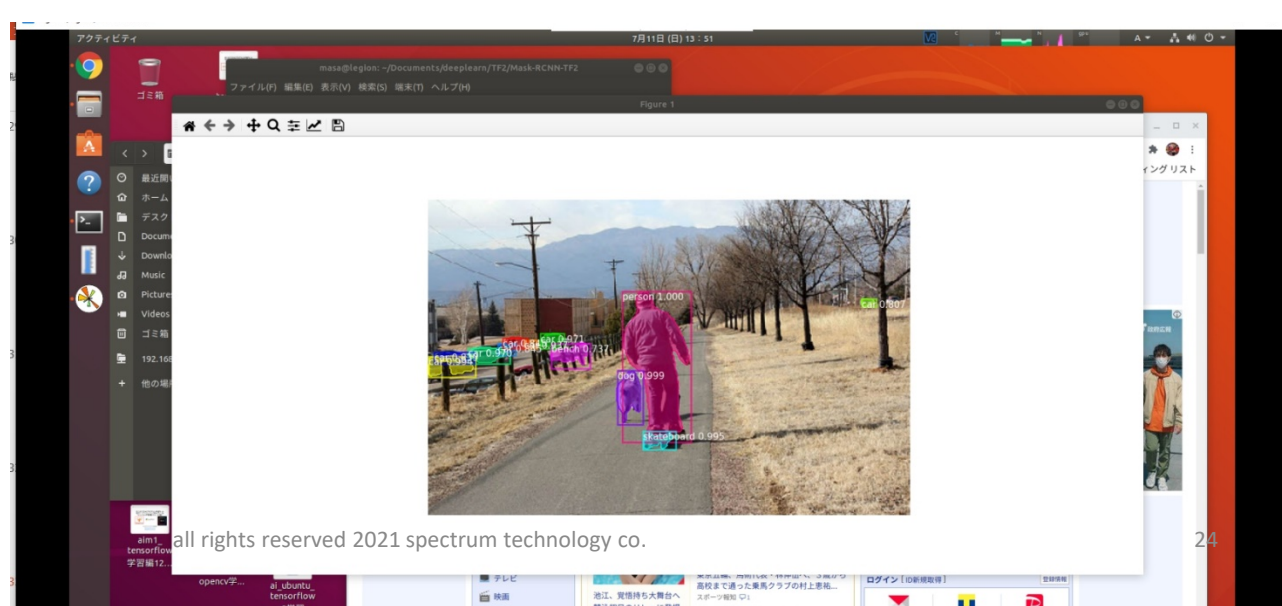
```
$ cd /home/masa/Documents/deeplearn/TF2/Mask-RCNN-TF2/
```

```
$ python3 mask-rcnn-prediction.py
```



<https://github.com/ahmedfgad/Mask-RCNN-TF2>

上記のオリジナル版を修正してフォルダにアップロードしています。上書きしないように注意。



Tensorflow開発

1. Object detection

⑧セグメンテーション事例_風船(jupyter版)

コマンド

```
$ cd /home/masa/Documents/deeplearn/TF2/Mask-RCNN-TF2/
```

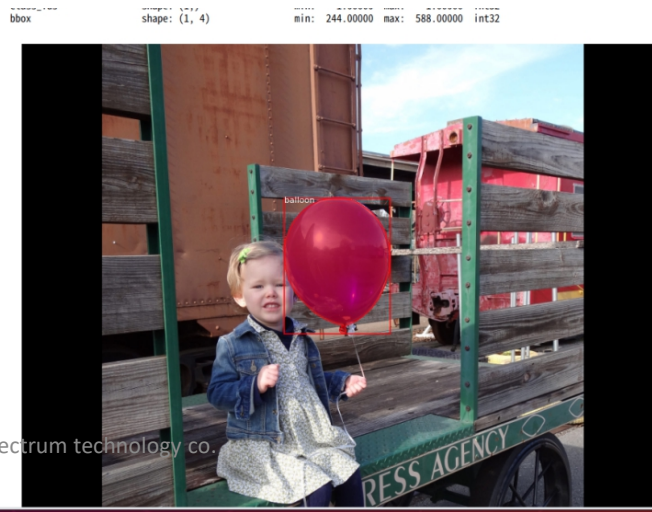
```
$ jupyter notebook
```



<https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46>

上記に資料がございます。

- Mask RCNNを使ったセグメンテーション事例(風船)です。
- Jupyter notebookで全体の流れを確認できます。
- #skipと表示してある項目は実施しないでください。
- cd /home/masa/Documents/deeplearn/TF2/Mask-RCNN-TF2/
- Jupyter notebook
- ブラウザがポップアップします。
- inspect_balloon_data.ipynbを選択
- モデル: 学習済のmask_rcnn_balloon.h5使用。1分類(風船)
- ディレクトリの位置はお客様の環境に合わせて変更してください。
- 検出した物体には、該当の場所をマスクして識別します。



Tensorflow開発

1. Object detection

⑨アノテーションツール (labellmg)

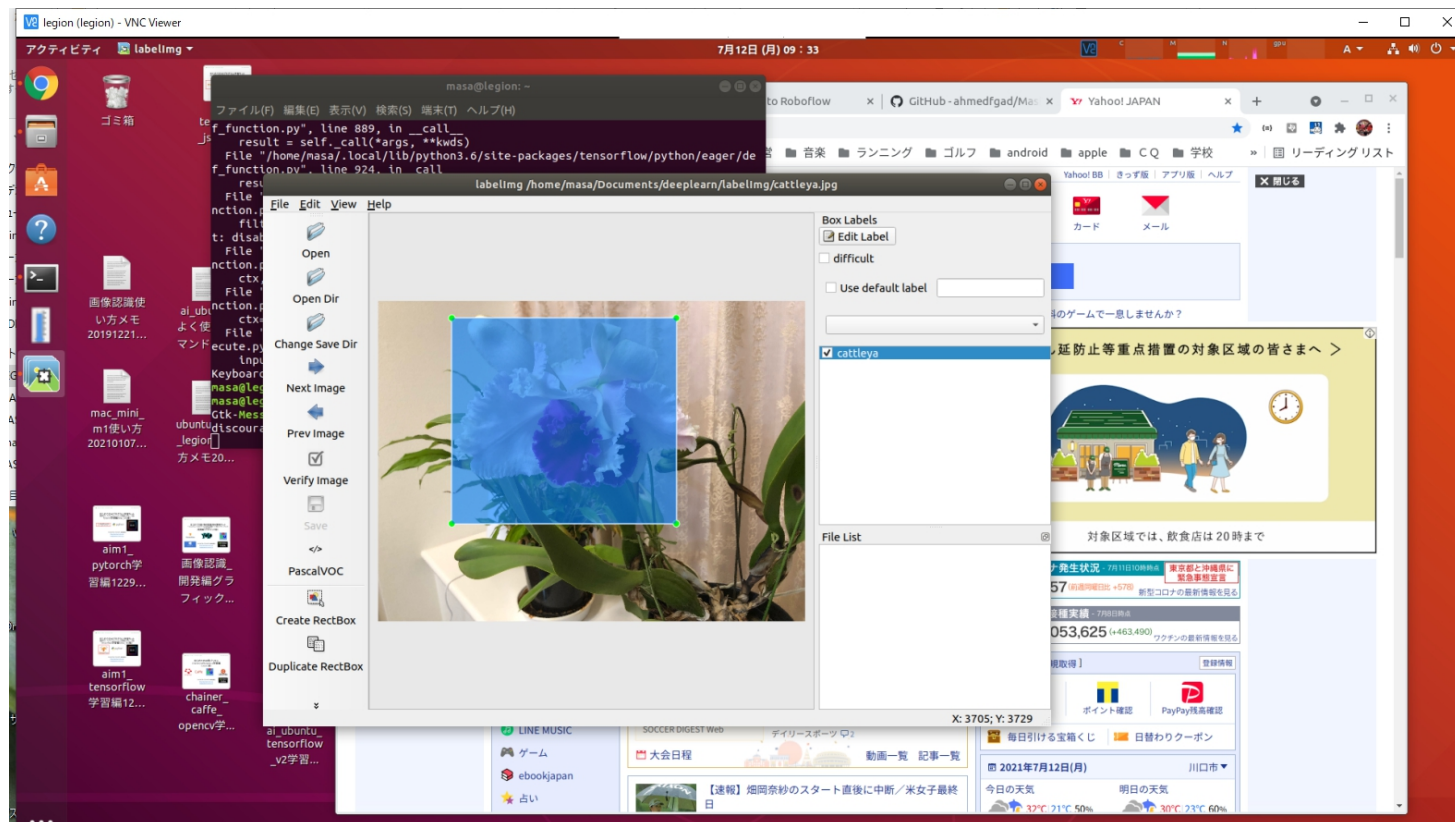
コマンド

```
$ cd /home/masa/Documents/deeplearn/labellmg
```

```
$ labellmg
```

TensorFlow

- labellmg ツールが起動します。
- 使用しているフォルダを開き、作業します。
- <https://github.com/tzutalin/labellmg>



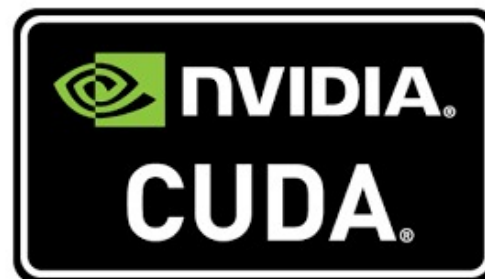
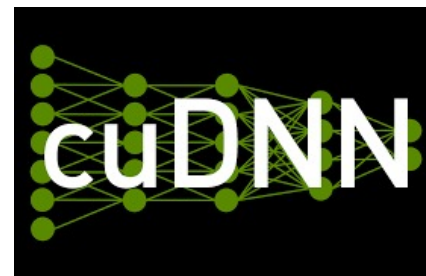
Yolo

- Darknetとして有名なYoloです。高速処理が可能な点が強みです。基本c言語で書いたものをpython3でも使える用になっています。
- Darknet: <https://pjreddie.com/darknet/>

Yolo AI概要

1. 開発キット全体像

開発キット



Yolo AI概要

2.開発キットプログラム一覧

A:実用可能
B:チャレンジ
C:試験段階

API名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
yolov3	物体検出(写真)	写真から登録済の物体を検出	A	100%	80分類 、モデル作成手 順も解説
	物体検出(ビデオ)	ビデオから登録済の物体を検出	A	100%	
	物体検出(カメラ)	カメラから登録済の物体を検出	A	100%	
	顔認識1(写真、カ メラ)	写真から登録済の人名を検出、登録も あり	A	90%	6名の登録者
	顔認識2(カメラ)	カメラから登録済の人名を検出、登録も あり	A	90%	

Yolo開発

1. Yolo v3

②物体検出(ビデオ)

コマンド

```
$ cd /home/masa/Documents/deeplearn/TF2/yolov3-tf2  
$ python3 detect_video.py --video car_chase_01.mp4 --  
output ./output.avi
```

- Darknetとして有名なYoloです。高速処理が可能な点が強みです。基本c言語で書いたものをpython3でも使える用になっています。
- <https://github.com/zzh8829/yolov3-tf2>
- `python3 detect_video.py --video car_chase_01.mp4 --output ./output.avi`
- 認識速度は、GPUの性能に依存します。



Yolo開発

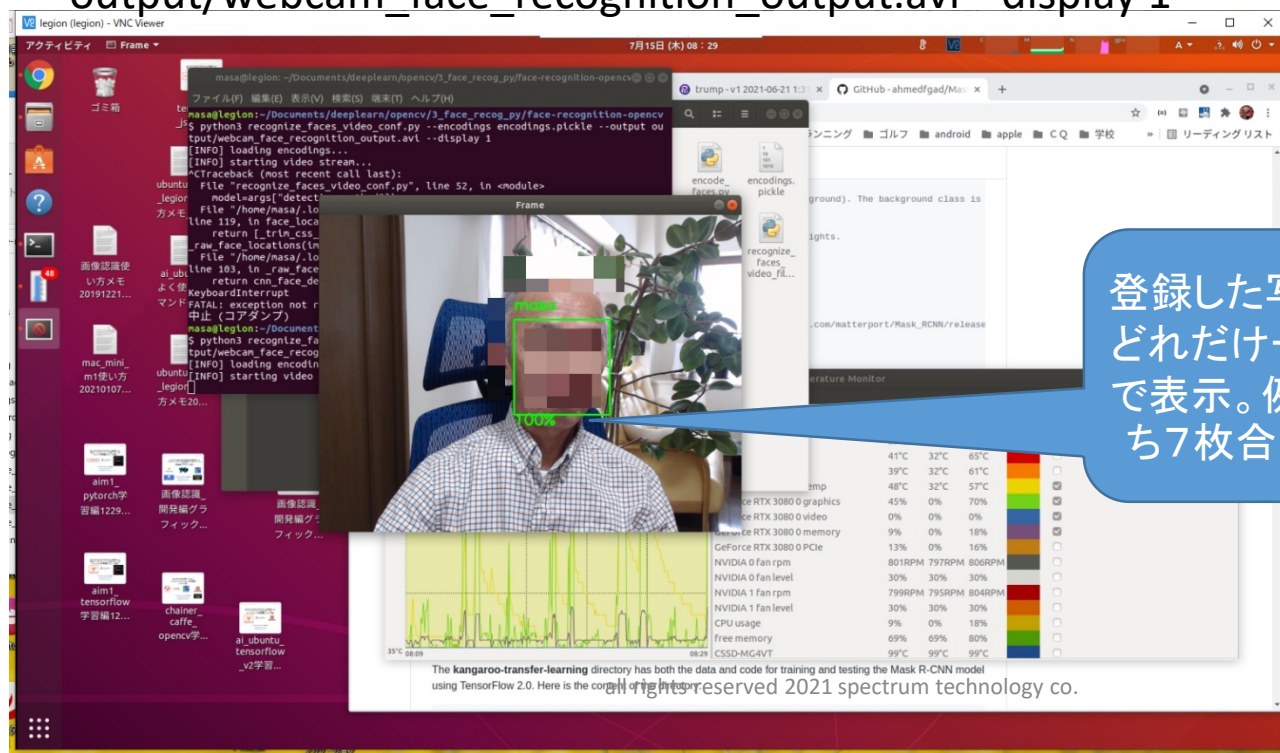
3. Yolo例

①顔認識1

コマンド

```
$ cd /home/masa/Documents/deeplearn/opencv/3_face_recog_py/face-  
recognition-opencv
```

- <https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python3-and-deep-learning/>
- 顔検出(カメラ) 一致率表示
- `python3 recognize_faces_video_conf.py --encodings encodings.pickle --output output/webcam_face_recognition_output.avi --display 1`



Opencv開発

1. Opencv

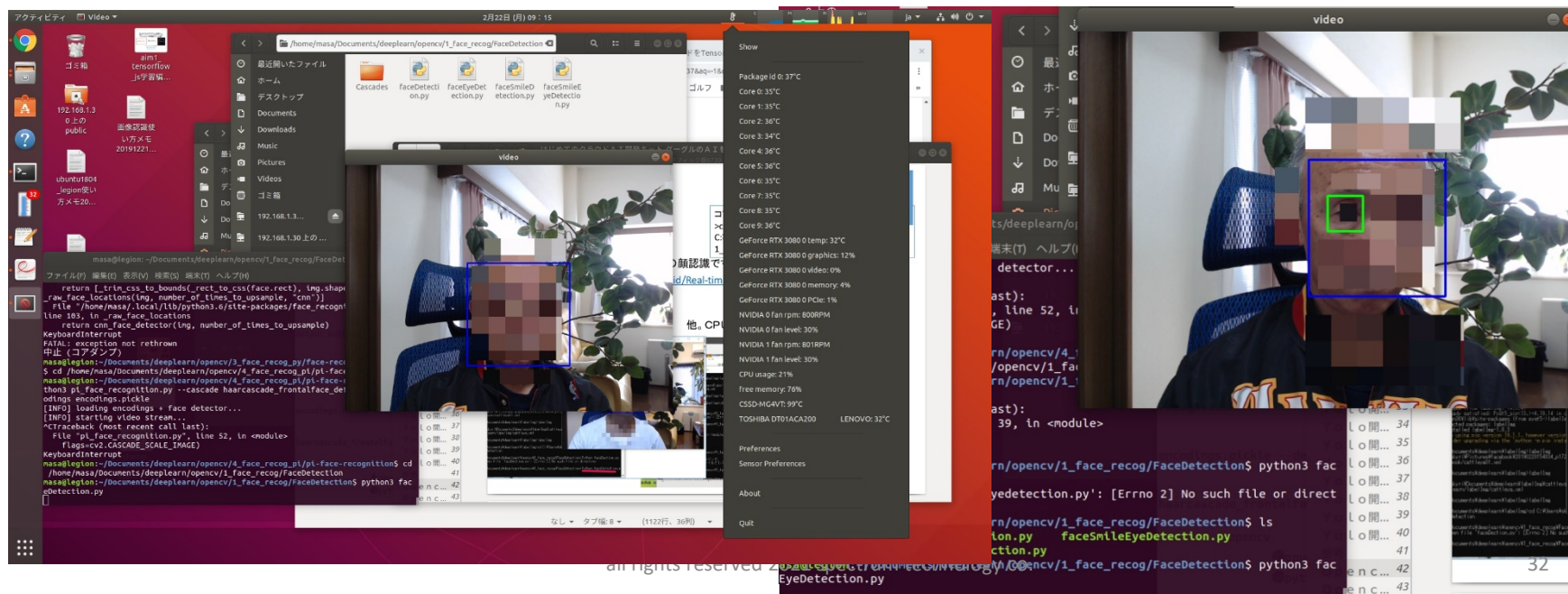
①顔認識(カメラ)

コマンド

\$ cd

/home/masa/Documents/deeplearn/opencv/1_face_recog/Face
Detection

- Opencvを使ったwebカメラでの顔認識です。
- <https://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/>
- python3 faceDetection.py
- python3 faceEyedetection.py



顔認識、物体認識 応用例 開発キットプログラム一覧

A:実用可能
B:チャレンジ
C:試験段階

区分	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
顔認識	居眠り運転検出 (Jupyter,カメラ)	顔から眼を検出し、瞼の動きにより数秒間閉じると警報発出。	B	70%	
	居眠り運転検出 (Pytorch, カメラ)	顔から眼を検出し、瞼の動きにより数秒間閉じると警報発出。	B	70%	
	居眠り運転検出 (Python、カメラ)	眼詳細を検出し、眼の閉じ具合により警報発出。	B	70%	
人体認識	Posenet(写真)	人体の顔、骨格などの位置を検出し、動きを可視化	B	70%	
	Posenet(カメラ)	人体の顔、骨格などの位置を検出し、動きを可視化	B	70%	

顔認識、物体認識 応用例

④ポーズネット事例 (python版、写真)

コマンド

```
$ cd /home/masa/Documents/deeplearn/TF2/posenet-  
python/  
$ python3 image_demo_v2.py --model 101 --  
image_dir ./images --output_dir ./output
```

- 写真の人体の位置を示す事例
- <https://github.com/rwightman/posenet-python>
- `cd /home/masa/Documents/deeplearn/TF2/posenet-python/`
- `python3 image_demo_v2.py --model 101 --image_dir ./images --output_dir ./output`
- モデル: 学習済の101使用。
- 人体の足、腰、肩、顔などの位置を示します。

