カメラデータを活用したエッジAIの実行

eForce

アジェンダ

1. 会社紹介
2. µC3でのAI実行
3. エッジ側でAIを実装する理由
4. 今回作成するエッジAIアプリケーションの概要
5. 開発環境
6. 学習済みモデルの作成概要
7. AIの組み込み
8. まとめ
9. 製品紹介

会社紹介

- **本社　　：東京**
- **開発拠点　　：東京 / インド**
- **事業内容**
  - Embedded事業
    - RTOSやミドルウェアの開発・販売
  - IoT事業
    - IoTプラットフォーム「iot-mos」の開発・販売
    - コンサルタント
- **沿革**

2006　:設立
国内のOSベンダとして初めて、ArmのCortex®-M/Aに対応し
「μC3/Compact」、「μC3/Standard」をリリース
2008　:TCP/IPスタック「μNet3」を開発
2013　:マルチコアデバイスに対応した「μC3/Standard+M」をリリース
2015　:産業用イーサネットへの取り組みをスタート
2016　:無線LANモジュール向けのSWパッケージを開発
2017　:RTOSとLinuxを共存させるソリューションをリリース
2018　:IoTプラットフォーム「iot-mos」を発表
2019　:Arm Cortex®-M33(TrustZone)に正式対応

µC3でのAI実行

- µC3で簡単にAI実行が可能に

µC3/Compact

+

STM32 CubeMX X-CUBE-AI

学習済みモデルの作成概要

- **AI実行までの流れ**

マイコン環境でAIを動かすまでにファイル変換が必要となる

Google colaboratry

独自にAIの学習モデルを作成

.tfliteファイル生成

STM32 CubeMX
+
X-CUBE-AI

STM32向けに最適化された
C言語ファイルへの変換

※STM32Cube.AI runtime/
TFLite Micro runtime
の2種類でのC言語ファイル変換が可能

.c/.hファイル生成

IAR Embedded Workbench for ARM

µC3を使用したAIの実行

エッジ側でAIを実装する理由

- クラウドAIのメリット・デメリット

メリット

処理能力が高い

保守性が高い

デメリット

判定までの遅延

通信コストが高い

セキュリティリスク

エッジ側でAIを実装する理由

- エッジAIのメリット・デメリット

メリット

リアルタイムな判断が可能

通信コストの削減が可能

セキュリティの強化

デメリット

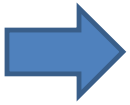
処理能力が端末依存

保守性が悪い

エッジ側でAIを実装する理由

- **実装箇所の使い分けが必要**

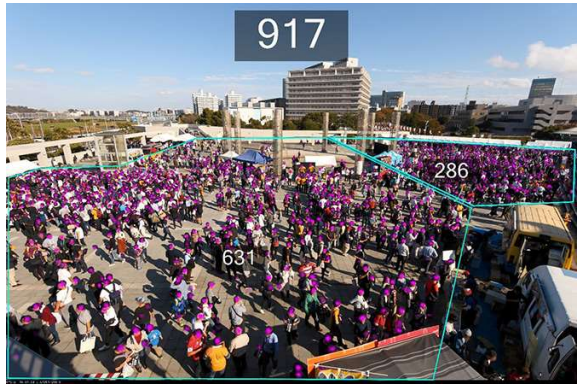
- 端末数が多くて通信コストや、クラウド費が高い
- 個人情報の取り扱い上、外部データを置きたくない
- 処理遅延が気になる
- 大規模データを扱わない



エッジ側でAIを実装！！

エッジ側でAIを実装する理由

- エッジAI活用の実用例



人数のカウント

異常検知

病気の診断

エッジ側でAIを実装する理由

- エッジAI活用の実用例

転倒検知

顔認証

不審物検知

エッジ側でAIを実装する理由

- **エッジでAIを動かす際の気になる制約**

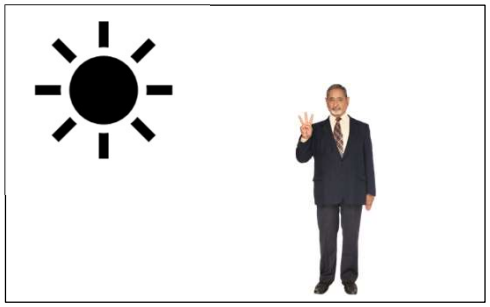
1. 学習もエッジで行うのか？学習はクラウドで推論はエッジで行うのか？
2. エッジAIを実行するためにどれくらいのメモリが必要になるか？
3. エッジAIの処理速度はどれくらいになるか？

実際にエッジAIの画像識別アプリケーションを作成して確認する

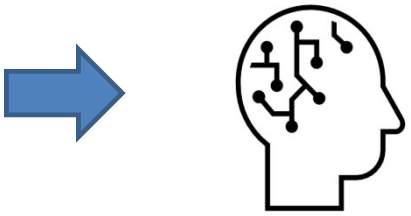
今回作成するエッジAIアプリケーションの概要

- 今回やりたいこと

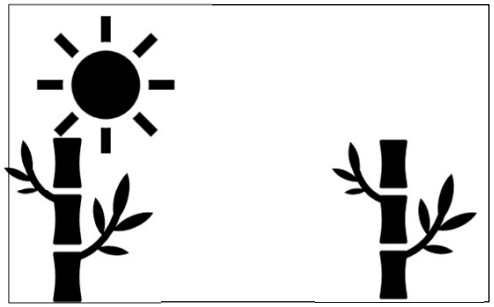
– 人が写っているかどうかをAIで識別する



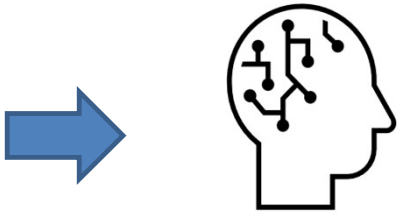
学習済みモデル



人が写っている



学習済みモデル



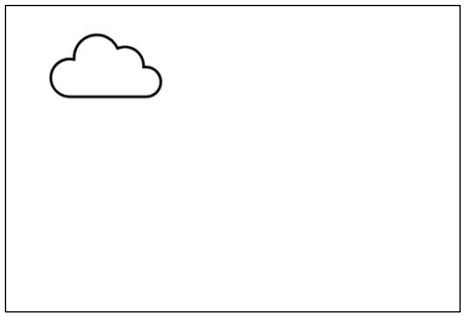
人が写っていない

今回作成するエッジAIアプリケーションの概要

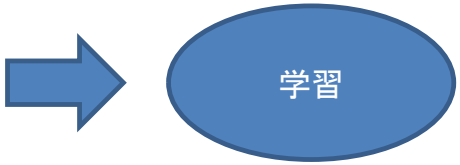
- **必要な処理**

1.学習済みモデルを作成する

学習用の画像



「人が写っている画像」と
「人が写っていない画像」の特徴



学習済みモデルを作成

頭と手と足があれば人がいそう
それ以外は人がいない

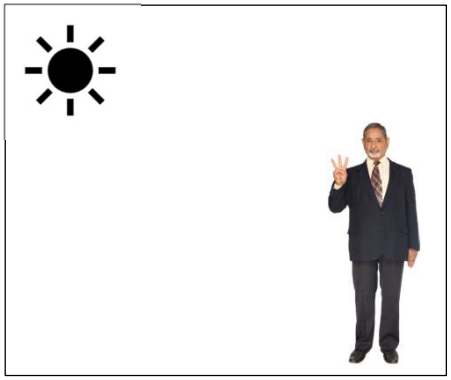
今回作成するエッジAIアプリケーションの概要

- **必要な処理**

2. 学習済みモデルから推論する

頭と手と足がある
たぶん人いるわ

カメラから取得した画像



学習済みモデルによる推論

出力

人がいる

人がいない

今回作成するエッジAIアプリケーションの概要

- 学習と推論の違い

	学習	推論
目的	入力と出力を対応付けるパターンを見つける	入力に対して出力を予測する
入力	パターンを見つけられるくらいの大量の画像	予測に使う画像のみ(1枚でもOK)
計算コスト	入力から出力を計算するパラメータを求めるため非常に多い	学習時のパラメータを固定して使うため学習よりは少ない

– メモリ制約の大きい組み込みデバイスでは大量の画像を載せるのは難しい
– 学習ではクラウド+GPUで数日間かかることもある

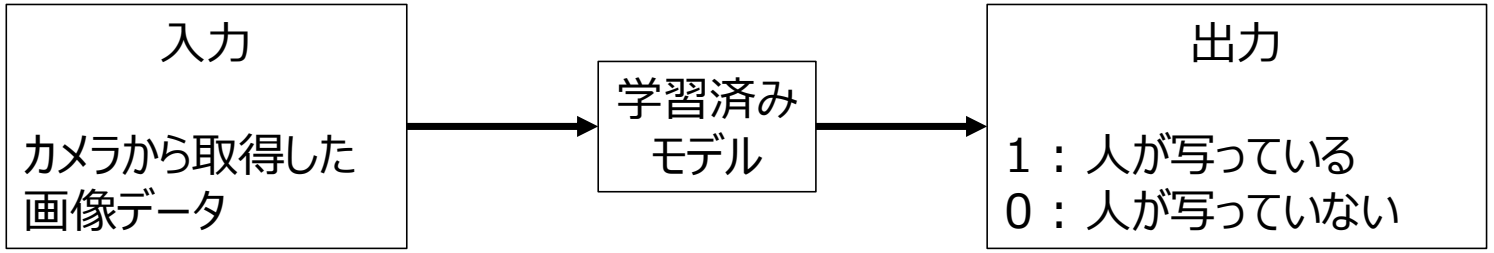
学習にはクラウド+GPU環境

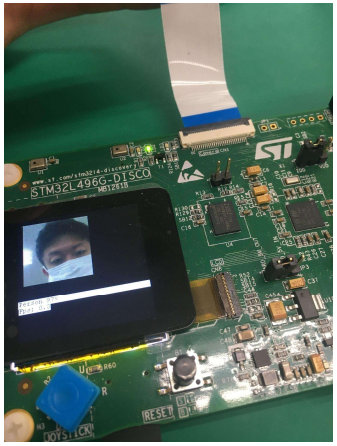
推論にはエッジ(組み込み)デバイス

図: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 より引用

今回作成するエッジAIアプリケーションの概要

- 今回作るもの





「人がいる」
とディスプレイに表示

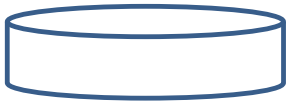
「人がいない」
とディスプレイに表示

今回作成するエッジAIアプリケーションの概要

- **学習から推論までの流れ**

1.学習済みモデルの作成

学習用の画像



公開されている学習用
画像データセット

クラウドで学習

GoogleのクラウドGPU

学習済みモデルの作成

今回作成するエッジAIアプリケーションの概要

- **学習から推論までの流れ**

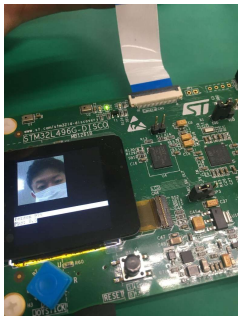
2. 学習済みモデルから推論する

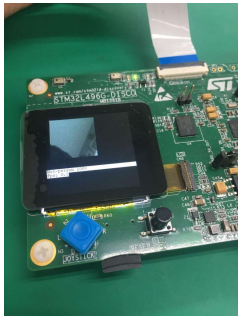
カメラから画像を取得

B-CAMS-OMV

学習済みモデルで推論

LCDに結果を表示





「人がいる」
「人がいない」
を推測する

開発環境

- **実行環境**

使用した評価ボード

- STM32L496G-Discovery

RAM	320KB
CPU	Cortex-M4
Clock	80MHz
周辺機器	240x240Pixel LCD フラットケーブル接続可のカメラ用コネクタ

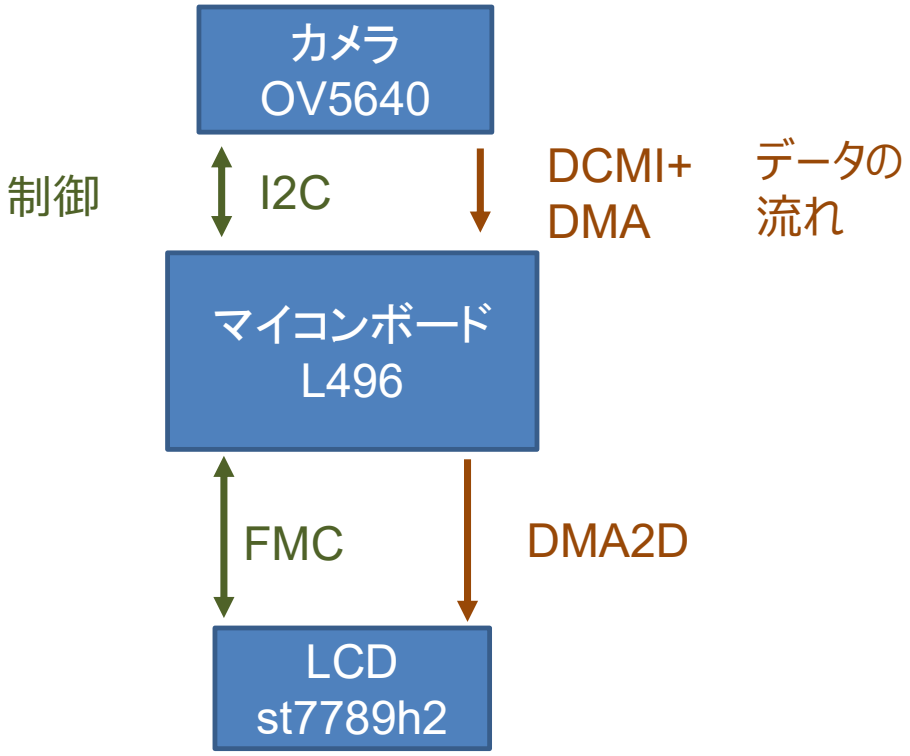
周辺環境

カメラモジュール	B-CAMS-OMV(OV5640)
OS	µC3/Compact for STM32L4
開発PC	Windows 10

開発環境

- 実行環境

画像データと制御の流れ



DCMI: STマイクロ社のカメラ用インタフェース

FMC: STマイクロ社のメモリコントローラ

DMA2D: STマイクロ社のグラフィックスに特化したDMA

開発環境

- **開発環境**

AIモデル作成環境　：　Google Colaboratory

Googleが提供している環境構築が不要で、Python記述で機械学習が行えるサービス

※紹介：Teachable　Machine
https://teachablemachine.withgoogle.com/

マイコン設定環境　　：　SMT32 CubeMX

今回使用するSTM32L496G-Discoveryボードのマイコン設定環境
学習済みAIモデルをC言語ファイル変換を行うX-CUBE-AIも使用できる

デバック環境　　：　IAR　Embedded Workbench for ARM

IARシステムズ社が提供する統合開発環境で、ビルドやデバックに使用する

学習済みモデルの作成概要

- **STM32CubeMX　X-CUBE-AI**

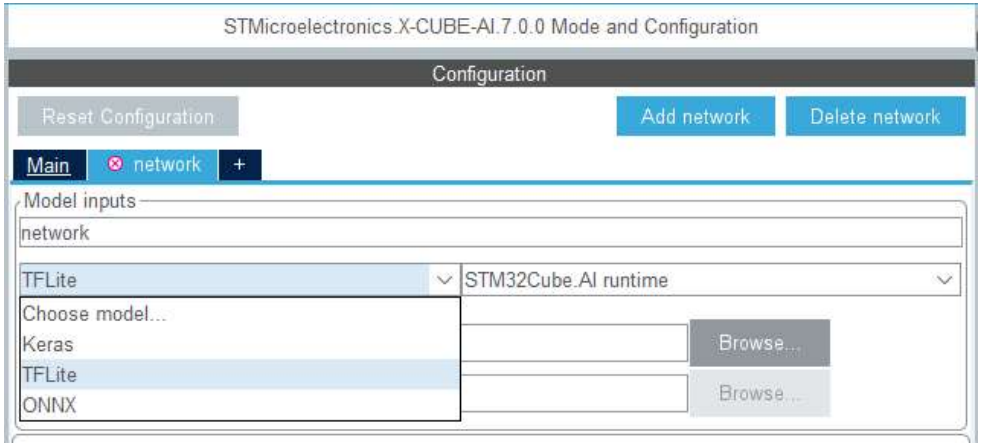
STM32CubeMXの拡張機能としてX-CUBE－AIが用意されている。
X-CUBE-AIを用いて、マイコンで使用できるC言語ファイルへの変換を行う。

使用できるAIフレームワーク

Keras(ケラス)

TFLite(テンソルフローライト)

ONNX(オニキス)



学習済みモデルの作成概要

- **学習済みモデルの最適化**

マイコン環境でAIを動かすまでにファイル変換が必要となる

Google colaboratry

AIモデルの学習と量子化による
tflite形式ファイルへの変換

↓ .tfliteファイル生成

STM32 CubeMX
+
X-CUBE-AI

STM32向けに最適化された
C言語ファイルへの変換

※STM32Cube.AI runtime/
TFLite Micro runtime
の2種類でのC言語ファイル変換が可能

↓ .c/.hファイル生成

IAR Embedded Workbench for ARM

マイコン環境で扱える形式での実行

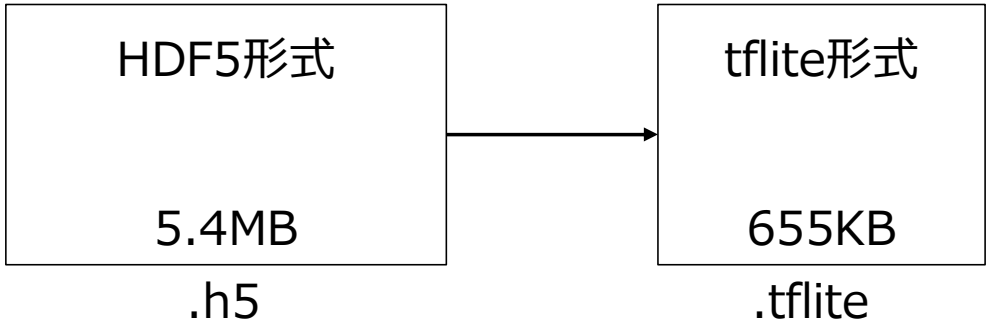
学習済みモデルの作成概要

- 量子化(Quanization)による軽量化

  - 学習済みモデルファイルは5.4MBで340KBのRAMには載らない
  - どうにかしてメモリサイズを小さくしないといけない

量子化(Quantization)と呼ばれるモデルファイルの中身を
32bitのfloat型から8bitのuint型に変換する方法を使う

軽量なtflite形式へのQuantization

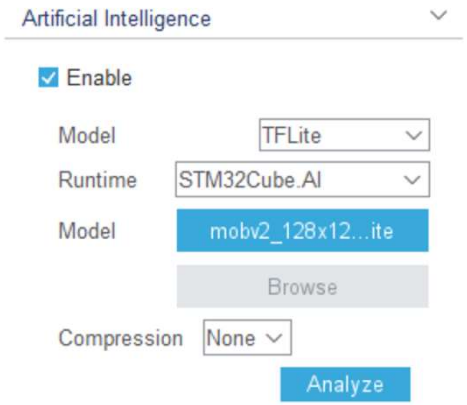


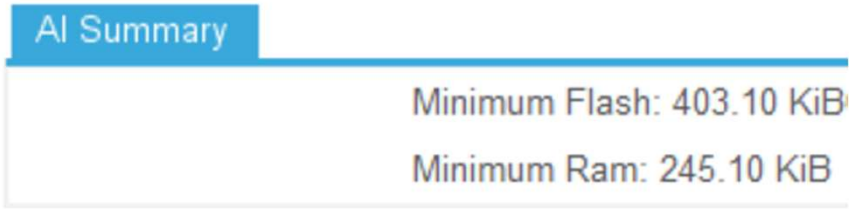
- 予測精度が多少低くなるケースもあるが、
  だいたいはそれほど影響がない

学習済みモデルの作成概要

- **X-CUBE-AIによる軽量化**

  - STマイクロ社のボードでAIによる推論を行う場合にはX-CUBE-AIによって更に軽量化できる
  - CubeMXでtflite形式の学習済みモデルからどれくらいのRAMが必要な計算してくれる



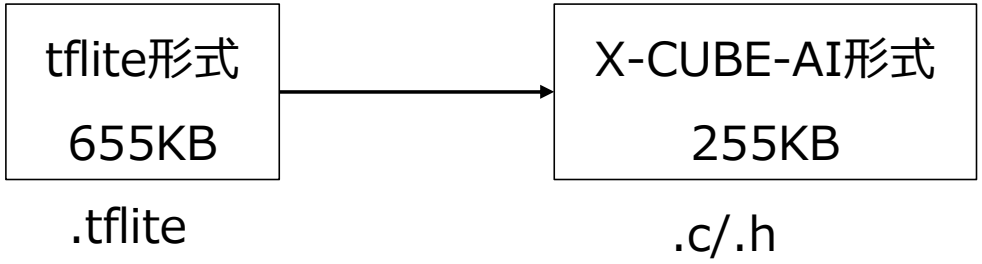


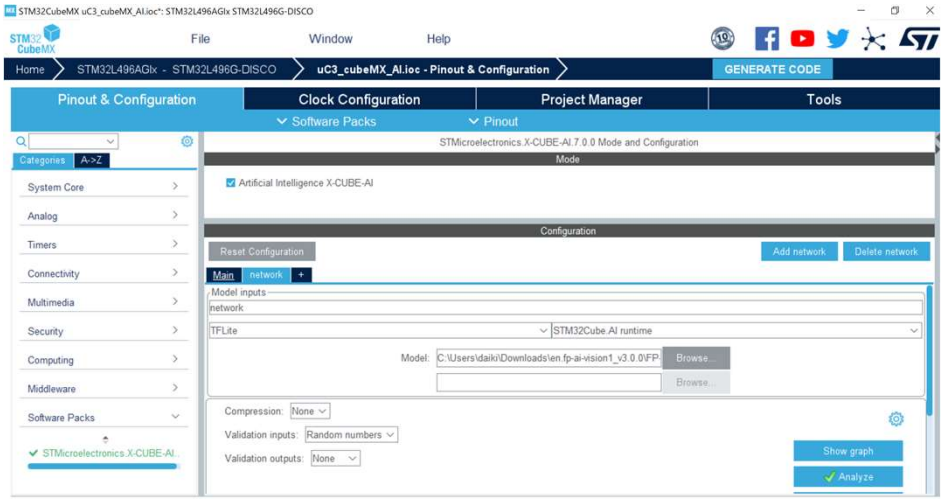
学習済みモデルの作成概要

- **X-CUBE-AIによる軽量化**

X-CUBE-AIでAIのパラメータをSTM32向けに最適化する

- X-CUBE-AIの設定画面からtfliteのパラメータファイルを指定して GENERATE CODEを選択してnetwork.cを生成する
- network.cをプロジェクトに組み込む





AIの組み込み

- モデルに画像データを学習させる

学習環境

- GPUクラウド Google Colaboratory
- NVIDIA V100
- メモリ 25.46GB
- ストレージ 147.15GB

学習モデル

- MobileNetV2
- Google Colaboratoryで2日間の学習
- 5.4MBのパラメータを持つ学習済みモデルファイルが生成

```
run_id = "mobv2_128x128x3_adam02"
m0 = tf.keras.applications.mobilenet_v2.MobileNetV2(input_shape=IMG_SHAPE,
                                                    alpha=0.35,
                                                    weights='imagenet',
                                                    include_top=False)
x = GlobalAveragePooling2D()(m0.output)
x = Dropout(0.5)(x)
p = Dense(2, activation='softmax')(x)
m = tf.keras.Model(inputs=m0.input, outputs=p)
m.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00002),
          loss=CategoricalCrossentropy(),
          metrics=['accuracy'])
m.summary()

acc_val_checkpointer = ModelCheckpoint(
    "/content/drive/MyDrive/TechnicalArticle/AI/model/"+run_id + ".best_val.{epoch:03d}.h5",
                                       verbose=1,
                                       save_best_only=True,
                                       monitor='val_accuracy')
acc_train_checkpointer = ModelCheckpoint(
    "/content/drive/MyDrive/TechnicalArticle/AI/model/"+run_id + ".best_train.{epoch:03d}.h5",
                                        verbose=1,
                                        save_best_only=True,
                                        monitor='accuracy')
csv_logger = CSVLogger("/content/drive/MyDrive/TechnicalArticle/AI/log/"+run_id + '.txt')

m.fit_generator(generator=train_generator,
                epochs=100,
                initial_epoch=0,
                validation_data=test_generator,
                validation_steps=len(test_generator) // 3,
                callbacks=[acc_val_checkpointer, csv_logger, acc_train_checkpointer],
                verbose=1,
                workers=4)
```

学習済みモデルファイルをエッジデバイスに載せて推論を行う

AIの組み込み

- **学習済みモデルの作成**

入力画像データセット

- 公開されているデータセットを使う
- 独自に収集して使う

→ COCO 2014データセット
- 人の画像が多く含まれるCOCO2014 データセットを使用する
- 物体認識用の画像が12万枚入っている
- 人や木、車など80クラスの正解ラベルが付与されている

学習に使用するモデル

- 目的と制約に合わせて選択する

→ MobileNetV2
- 物体認識タスク向けに開発された学習モデル
- 他のモデルよりも計算量・メモリ消費が少ない

出力となる正解ラベル

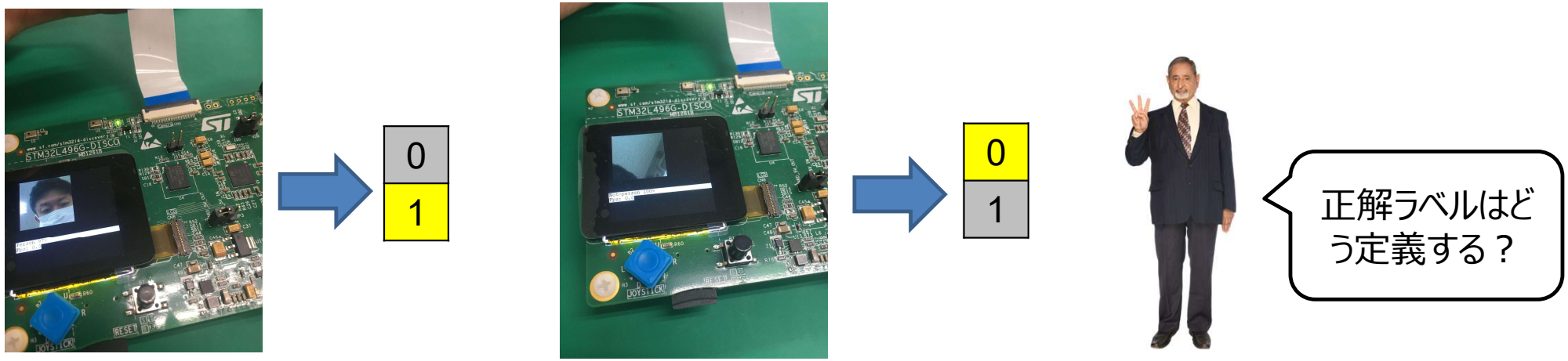
- 識別したいものを定義する

→
- 入力された画像が1(人がいる)なのか、0(人がいない)なのかを出力したい

AIの組み込み

- **正解ラベルの作成**

  - 識別したいものに合わせて正解ラベルは変更する必要がある
  - 今回は人がいる・いないを識別したいため、1は人がいる、0は人がいないとして正解ラベルを再作成したい



AIの組み込み

- **正解ラベルの作成**

  - 画像の領域のうち、20%以上の領域に人が写っていれば、「人がいる」画像ということにする
  - それ以外を「人がいない」画像とする

幅と高さから人が占める割合を求める
area_thresholdを20とする
人が占める割合をarea_thresholdを超えた場合にラベルをpersonに変更

```
persons = annotations.loc[annotations[0] == 0]
if persons.shape[0] != 0:
    big_persons = ((persons[3] * persons[4] * 100.0) > area_threshold).sum()
    if big_persons > 0:
        label = 'person'
    else:
        continue
```

AIの組み込み

- **X-CUBE-AIのライブラリを使った推論の実行**

後はX-CUBE-AIのドキュメントに記載のあるライブラリの関数に入力と出力用バッファを指定するだけ

1. acqire_and_process_dataでは
   in_dataに画像データを取得する
2. aiRunでin_dataに対して推論を行う
3. post_processではout_dataに出力され
   る計算結果を元に結果を画面に出力する

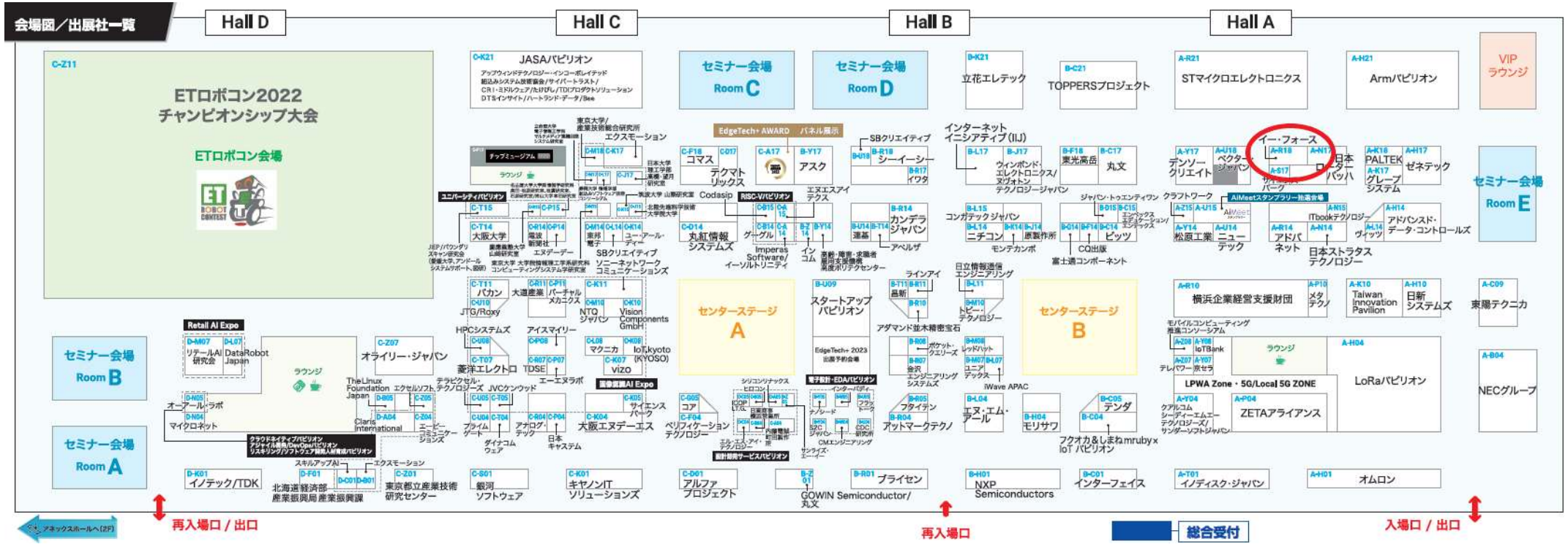
```
void main_loop()
{
  /* The STM32 CRC IP clock should be enabled to use the
network runtime library */
  __HAL_RCC_CRC_CLK_ENABLE();

  aiInit();

  while (1) {
    /* 1 - Acquire, pre-process and fill the input buffers */
    acquire_and_process_data(in_data);

    /* 2 - Call inference engine */
    aiRun(in_data, out_data);

    /* 3 - Post-process the predictions */
    post_process(out_data);
  }
}
```

まとめ

- **STマイクロ社のL496ボードを使ったエッジAIのまとめ**

1. 学習済みモデルでは量子化による軽量化で655KBまで小さくなる(.tfliteファイル)
2. STマイクロ社のX-CUBE-AIを使うことで255KBまでサイズを小さくすることができる(C言語ファイル)
3. Cortex-M4 80MHzで0.8FPSの処理速度
4. メモリと処理速度の制約があり、AIを使用する対象の選択が必要となる
5. 実行速度はCortex-M7が載っているボードなどで実行することで解決する可能性がある

今回作成した評価ボードは「イー・フォース」のブースに展示していますので
興味が御座いましたら見に来てください。



製品紹介

µC3のラインアップ

	µC3 Compact	µC3 Standard	µC3 Standard+M	µC3 +Linux
仕様	**µITRON4.0** 自動車制御プロファイル	**µITRON4.0** スタンダードプロファイル		
追加仕様	-	-	マルチコア対応	OpenAMP対応
特長	極小フットプリント	高い割込み応答性能	AMP型のマルチコア対応	Linuxとの共存が可能
対応CPUコア	arm ・Cortex®-M0/M0+/M3/M4/M7/M33 Renesas ・RX intel ・Nios Ⅱ	arm ・Cortex-M4/M7/M33, A5/A7/A8/A9/A15/A53/A73, R4/R5 Renesas ・Renesas RX 700	arm ・Arm Cortex-A7/A9/A15/A53	arm ・Cortex-A53/A53 ・Cortex-A53/R5 ・Cortex-A53/M4 ・Cortex-A9/A9 ・Cortex-A7/A7 ・Cortex-A7/M4
主な 対応デバイス （詳しくは製品ガイドを参照下さい）	STMicro STM32 NXP Kinetis, LPCxxx Renesas RA, RX200/600/700 Intel Nios Ⅱ Cypress PSoC Sililabs EFR32 Xilinx Microblaze（planning）	Renesas RZ/A, RZ/T, RZ/N STMicro STM32 NXP i.MX RT, i.MX6/7/8 TI AM335x, 57x, 65x Intel SoC Xilinx Zynq-7000, MPSoC	Renesas RZ/G1E, G1N, G1M TI AM57x NXP i.MX6/7, LS1043A Xilixn Zynq-7000, MPSoC Intel SoC	Renesas RZ/G1E NXP i.MX6/7/8M Mini STMicro STM32MP157 TI AM65x Xilinx Zynq-7000, MPSoC Intel SoC

µC3/Compact（ワンチップマイコン向けRTOS）

µC3/Compact はµITRON4.0 自動車制御プロファイルを採用

オーバーヘッドやメモリ使用量の削減を目的とした機能セット

～　他社製品にはないの３つの特徴　～

1．コンフィグレータによる初期設定・コードの自動生成

・**GUIツール**により直接のコーディングが不要
・コーディングミスを防いで**ロケットスタート**

開発期間短縮
品質向上

2．最小2.4Kバイトの極小フットプリント

・ROM/RAMを極限までダウンサイジング
・FreeRTOSと比べて、RAM使用量は約3割減

BOMコスト減

3．業界トップクラスのサポートＣＰＵ

・Arm Cortex®-M、Renesas RX、
Intel Nios Ⅱ等をベースにした国内外**50種類以上**の
マイコンシリーズに対応

実績 No.1

・スケジュール通りに開発を行いたい
（EETimes 2019 Embedded Markets Studyにおける、開発時のお悩みNo.1）
・ライフサイクルコストを抑えたい

市場ニーズ

eForce が提供する

・コンフィグレータ（無償）
・OSプラグイン（無償）
・1営業日以内のサポート（保守）

を活用する事によって
開発期間の短縮や
ライフサイクルコストの低減が可能

お客様の声

・OSが原因のバグはなかった
・遅延なく製品を出荷できた
・コーディング時間を最大20%削減
・トラブルフリーです

ユーザ事例集はコチラ
https://www.eforce.co.jp/case/

〇こんな心配をされている方には
「無償評価版」や「ハンズオンセミナー」がおススメ

☑RTOSの基礎を学びたい
☑µC3の開発イメージを掴みたい
☑費用を掛けずに評価したい

無償評価版や
ハンズオンセミナーを
ご活用下さい

μC3/ConfiguratorによってOS + TCP/IPを簡単設定！　主に、μC3/Compact向け

①CPU型番選択

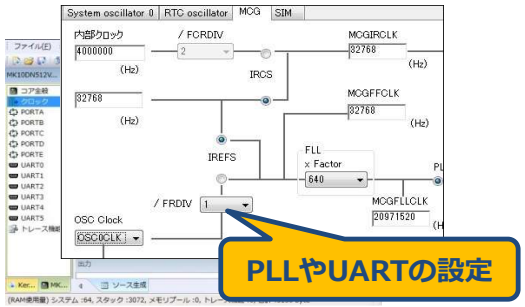
実際に使用する
デバイスの型番を選択

②カーネルの設定

RAMの使用量が
一目瞭然👀

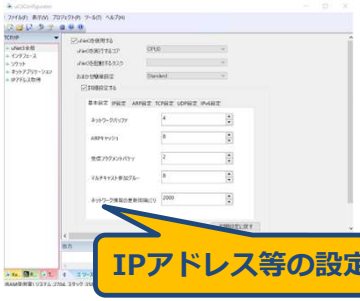
(RAM使用量) システム :174, スタック :4352, メモリプール :0, トレース機能 :0, 合計 :4526 byte

③内蔵ペリフェラルの設定



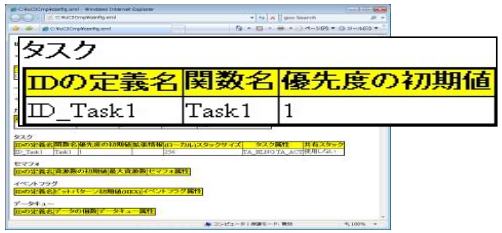
PLLやUARTの設定

④TCP/IPの設定



IPアドレス等の設定が可能

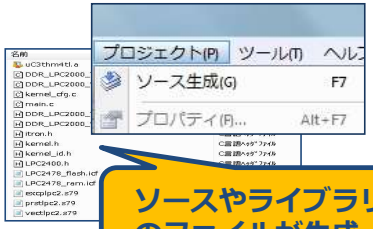
⑤ブラウザで設定内容をチェック



タスク

IDの定義名	関数名	優先度の初期値
ID_Task1	Task1	1

⑥ファイルの生成



ソースやライブラリ
のファイルが生成

⑦コードの自動生成

```
/*********************************
        Task1
*********************************/
void Task1(VP_INT exinf)
{
}
```

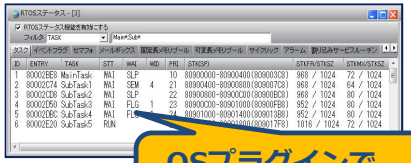
タスクが起動するまでの
コードが自動生成

⑧Appの記述

```
/*********************************
        Task1
*********************************/
void Task1(VP_INT exinf)
{
    for (;;) {
        REG_FIO1.LONG.SET = 0x00002000;
        REG_FIO1.LONG.CLR = 0x00040000;
        dly_tsk(1000);
        REG_FIO1.LONG.CLR = 0x00002000;
        REG_FIO1.LONG.SET = 0x00040000;
        dly_tsk(1000);
    }
}
```

⑨Build & Download

⑩Debug



OSプラグインで
リソース情報を可視化

μC3/Configurator → 無償バンドル
OSプラグイン → 無償ダウンロード
https://www.eforce.co.jp/download#dl02

製品紹介　iot-mos


試作から量産までを実現する
IoTプラットフォームサービス
iot-mos
iot-mos
iot-mos
iot-mos

製品紹介 iot-mos

iot-mos GATEWAY (Wi-Fi / LTE)

製品紹介 iot-mos

T01-EnOcean

1キーリモコンスイッチ

T02-EnOcean

2キーリモコンスイッチ

T04-EnOcean

4キーリモコンスイッチ

株式会社シブタニ製

スイッチストライクエアー

オプテックス株式会社製

ワイヤレスCO2センサー

オプテックス株式会社製

ワイヤレス開閉センサー

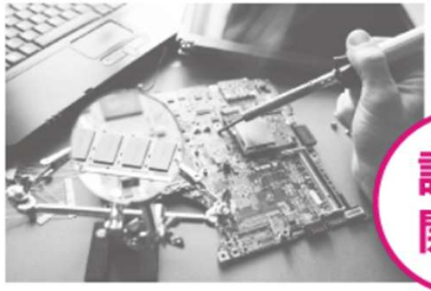
製品紹介　iot-mos

iot-mos CLOUD

サーバー利用料／6ヶ月：¥23,760（税込）

製品紹介　iot-mos

--- セミオーダーでの対応も可能です ---



試作
開発

実証
実験

量産

- 安価な評価用ボードを使った
  モックアップH/Wを開発（数台）
- 試作/実験用のソフトウェア開発
- 期間おおよそ2~3ヶ月

**まずはコストを抑えて
スモール&ファーストスタート**

- プロトタイプH/Wの開発（数台~数十台）
- プロトタイプH/Wでの実証実験
- 実運用レベルのソフトウェア開発

実験を行なって最適化

- 実製品の製造（数十台~数千台）

コストを考えた量産化

開発は3段階に分けて行います

ご視聴ありがとうございました

- 時間内に回答できなかったご質問に関しましては、
  後日、アンケートと共に回答を送付させて頂きます。

- QRコードを読み込んでアンケートに回答をお願い致します。

- アンケートに回答頂けた方には、
  本日のセミナー資料のダウンロードURLを送付させて頂きます。

本日のご参加、誠に有り難う御座いました。

参考文献

1. カメラから画像を取得してディスプレイ表示する方法を学ぶ
   https://www.eforce.co.jp/camera_1/
2. カメラから画像を取得してSDカードに保存する方法を学ぶ
   https://www.eforce.co.jp/camera_2/
3. RTOSを使ってAI分析
   https://www.eforce.co.jp/camera_3/
4. Counting People in Crowds with AI
   https://global.canon/en/technology/count2019.html
5. 鉄道に生きる
   https://www.westjr.co.jp/company/info/issue/bsignal/18_vol_181/rail/
6. COCO dataset
   https://cocodataset.org/#home
7. Common Objects in Context Dataset Mirror
   https://pjreddie.com/projects/coco-mirror/
8. ニューラルネットワークの重みとバイアス
   http://renom.jp/ja/notebooks/tutorial/basic_algorithm/neural_network_general/notebook.html
9. 確率的k勾配降下法(SGD)の設定
   http://renom.jp/ja/notebooks/tutorial/basic_algorithm/sgd-settings/notebook.html
10. A Comprehensive Guide to Convolutional Neural Networks- the ELI5 way
    https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
11. U-Net: Biomedical Image セグメンテーションのためのConvolutionalネットワーク
    http://renom.jp/ja/notebooks/tutorial/image_processing/u-net/notebook.html
12. Image Classification With Mobilenet
    https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470
13. MobileNetV2: Inverted Residuals and Linear Bottlenecks
    https://arxiv.org/abs/1801.04381
14. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications
    https://arxiv.org/abs/1704.04861
15. Discovery kit with STM32L496AG MCU
    https://www.st.com/en/evaluation-tools/32l496gdiscovery.html
16. Using the Chrom-ART Accelerator™ to refresh an LCD-TFT display on STM32L496xx/L4A6xx/L4Rxxx/L4Sxxx microcontrollers
    https://www.st.com/resource/en/application_note/dm00338361-using-the-chromart-accelerator-to-refresh-an-lcdtft-display-on-stm32l496xxl4a6xxl4rxxxl4sxxx-microcontrollers-stmicroelectronics.pdf
17. TFT LCD interfacing with the high-density STM32F10xxx FSMC
    https://www.st.com/resource/en/application_note/cd00201397-tft-lcd-interfacing-with-the-highdensity-stm32f10xxx-fsmc-stmicroelectronics.pdf
18. FatFs – Generic FAT Filesystem Module
    http://elm-chan.org/fsw/ff/00index_e.html
19. libjpeg
    http://libjpeg.sourceforge.net/
20. Teachable Machine
    https://teachablemachine.withgoogle.com/