Customization Guide

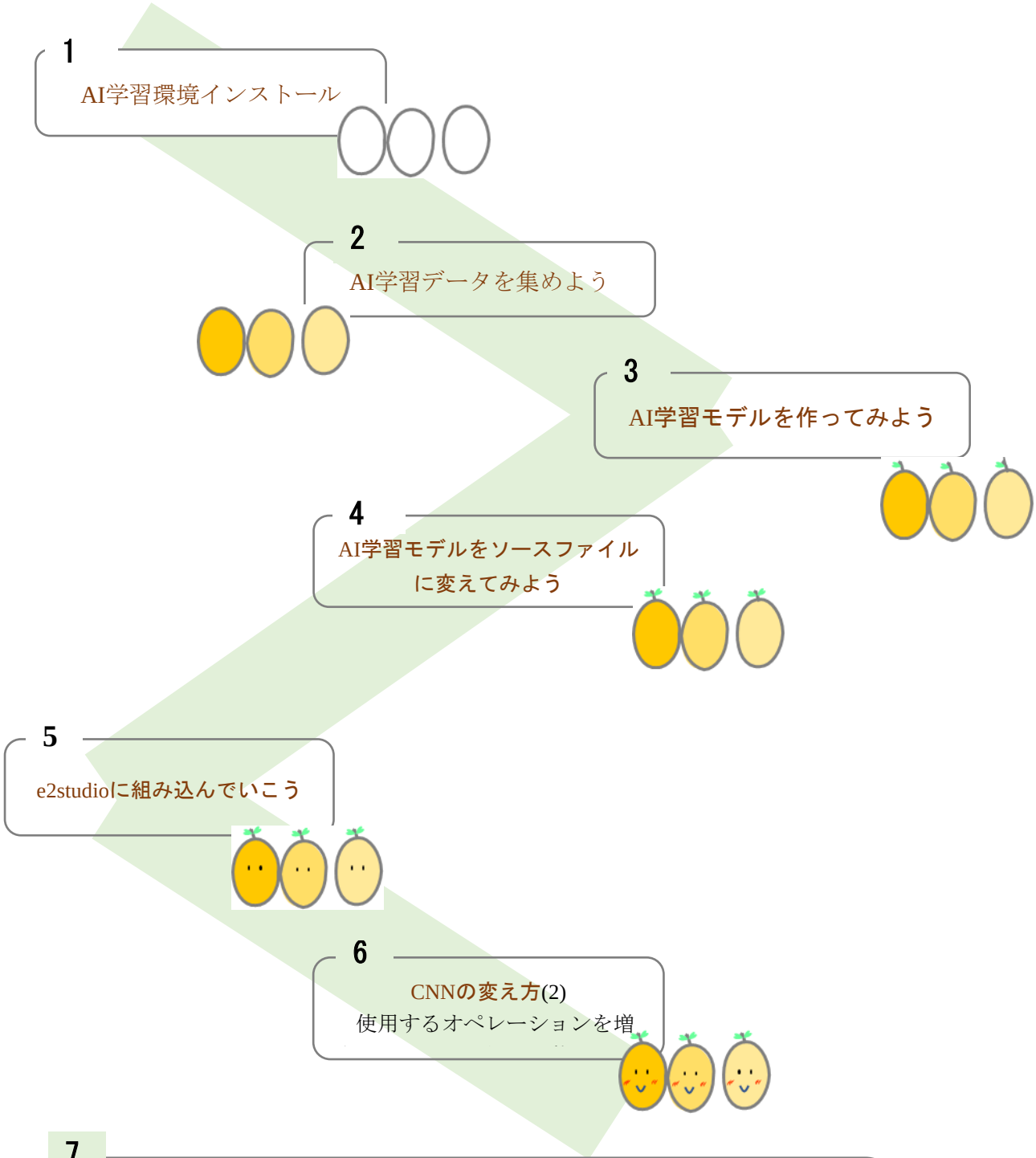
GR-MANGO で AI カスタマイズガイド

ルネサス エレクトロニクス EPMP-IMB-20-0265-02 2021.2
www.renesas.com

目次

1. はじめに

本書では、GR-MANGO（RZ/A2M 搭載）上で**お客様のお好きなフードメニューに変更して、AI でメニューを判別していく**手順をご説明していきます。



2. 使用環境のインストール

2.1 AI 学習環境インストール

AI 学習環境をインストールしていきます。

※e-AI starting package をご実施された方は「2.1.2(1)tensorflow インストール」のみ実施し、「2.1.2(7)version 確認」で Tensorflow のみバージョンが変わっていることを確認してください。

2.1.1 Python インストール

以下の WEB サイトより Python3.5.3 をダウンロードします。
https://www.python.org/downloads/release/python-353/
画面を下までスクロールして、「Files」の中の「Windows x86-64 executable installer」をクリックします。

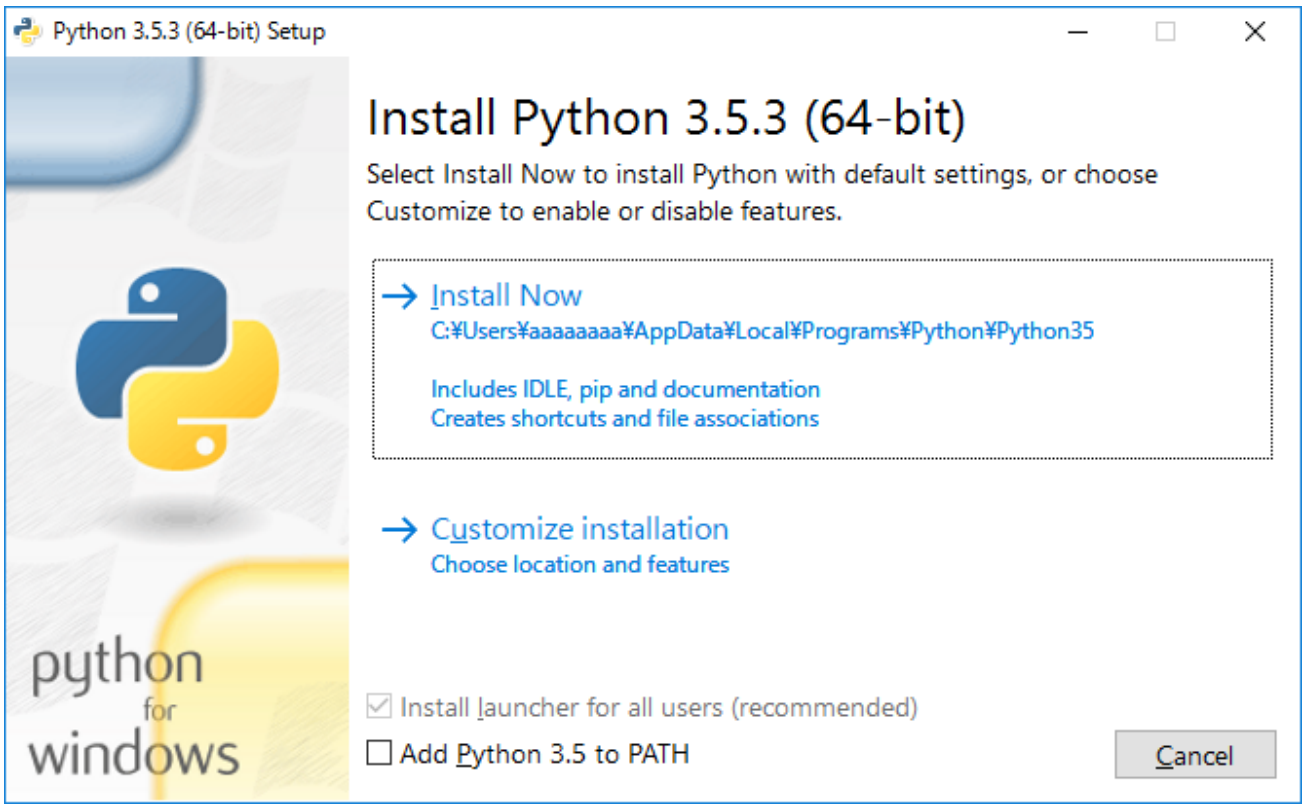
Files

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64

保存を押します。

python-3.5.3-amd64.exe を実行します。

インストーラをダブルクリックしてインストールを開始します。
“Install Now”をダブルクリックします。以降、インストーラの指示に従ってインストールを進めます。
備考 1：コマンド実行時にバージョンは指定するため、”Add Python 3.5 to PATH”の設定はチェックを付けないで実行します。
備考 2：Python のライセンスについては、インストールフォルダにある”Lisence.txt”をご一読ください。
[Python のインストールフォルダ]
C:¥Users¥<windows-user-name>¥AppData¥Local¥Programs¥Python¥Python35¥Scripts
※”<windows-user-name>”には Windows にログオン時のユーザ名



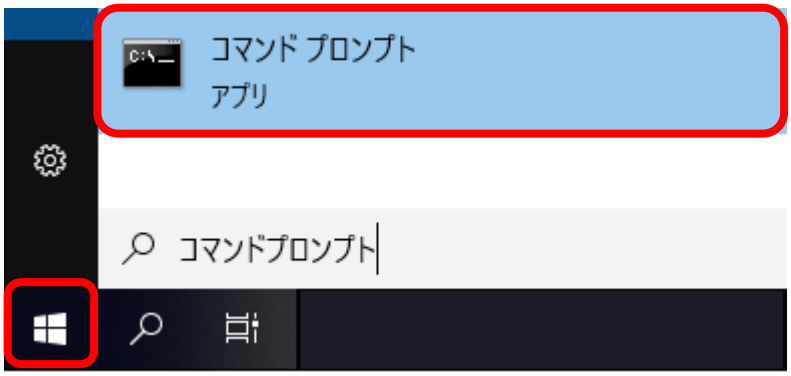
インストール完了後、以下の手順で正常にインストールされていることを確認します。
・Windows のコマンドプロンプトを開きます。
以下のコマンドを実行し、バージョンを確認します。下記はコピー＆ペーストでお使いいただけます。

```
py -3.5 -V
```

・コマンドの実行結果が”Python 3.5.3”となった場合、正常にインストールされています。

2.1.2 Python package のインストール

① Windows メニューから「コマンドプロンプト」を起動してください。



② 2.1.1 Python インストールでインストールした位置にフォルダを移動します。
下記コマンドを実行してください。

```
cd C:¥Users¥renesas¥AppData¥Local¥Programs¥Python¥Python35¥Scripts
```

※"renesas"という windows ユーザ名前の例です。

(1) tensorflow インストール

下記コマンドを実行してください。

```
pip3 install --upgrade tensorflow==2.0.1
```

(2) ProgressBar インストール

下記コマンドを実行してください。

```
pip3 install progressbar33==2.4
```

(3) Prettytable インストール

下記コマンドを実行してください。

```
pip3 install prettytable==0.7.2
```

(4) imageio インストール

下記コマンドを実行してください。

```
pip3 install imageio==2.6.1
```

(5) keras インストール

下記コマンドを実行してください。

```
pip3 install keras==2.2.4
```

(6) matplotlib インストール

下記コマンドを実行してください。

```
pip3 install matplotlib==3.0.3
```

(7) version 確認

下記コマンドを実行してください。

```
pip3 list
```

インストールしたバージョンが正しいか確認してください。

tensorflow	2.0.1
ProgressBar	2.4
Prettytable	0.7.2
imageio	2.6.1
keras	2.2.4
matplotlib	3.0.3

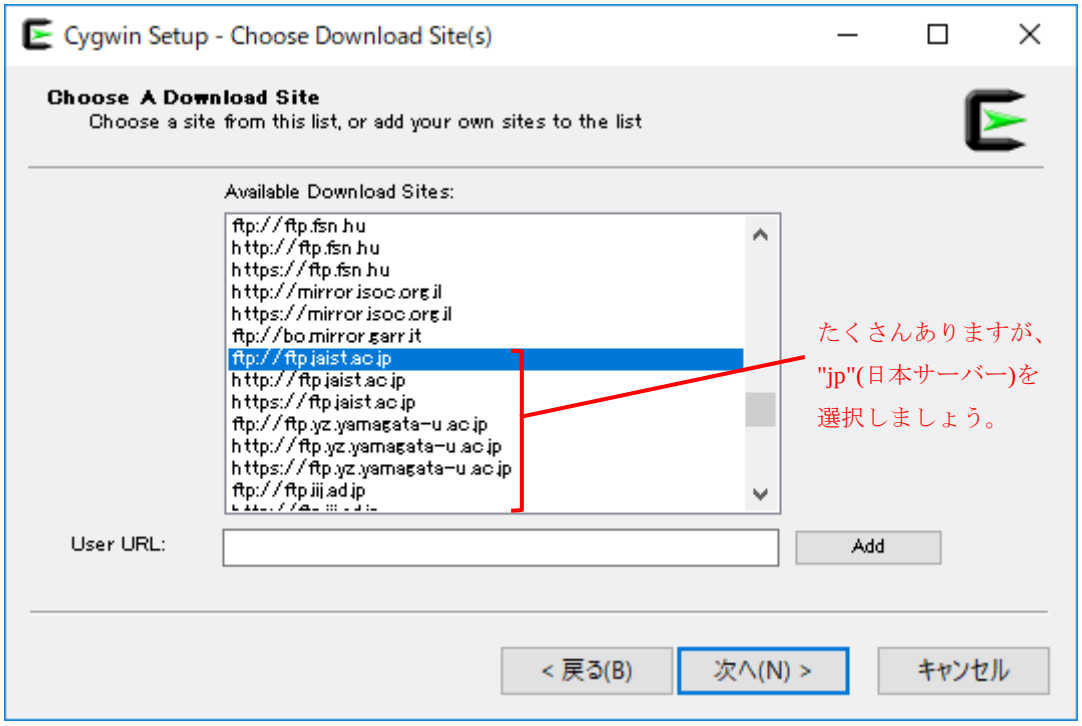
2.2 Cygwin のインストール方法

※Cygwin は Linux の xxd コマンドを使うためにインストールします。Ubuntu 等 Linux のコマンドを実行できる環境をお持ちの方はインストールの必要はありません。

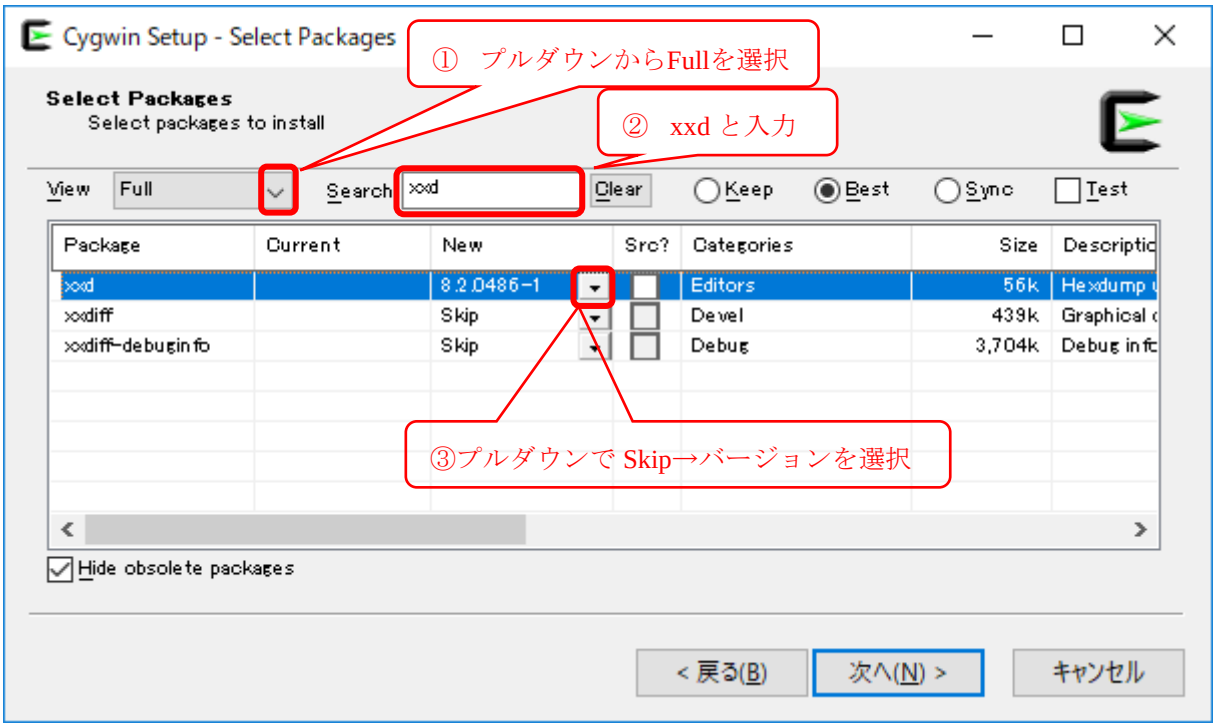
① Cygwinをhttps://www.cygwin.com/のページ内の**setup-x86_64.exe**をクリックし、ダウンロードします。

② setup-x86_64.exe をダブルクリックします。変更せず次へ次へと進めていきます。

③ ファイルをダウンロードするサイトを以下の図を参考に選択します。



④　パッケージの選択を以下の図を参考に行います。



⑤　次へ次へと変更せず進んで、完了です。

3.　AI 学習データを集めよう

AI の学習で使用するデータを収集していきます。方法を二つ書きます。

3.1　　写真を撮る

AI の学習で使用する写真を撮っていきます。以下のポイントに気をつけると AI で認識する精度が高まります。

POINT!

ＡＩの認識率をあげるためには

1　画質の悪くないカメラで撮影を行う
2　余計なものが入りこまないように撮影する
3　実際に AI を行う場所で撮影を行う
4　できるだけたくさん写真を撮る

FAQ

写真は何枚撮ったらよいの？

F：いったい何枚くらい写真を撮っているの？
A：１種類に対して 1,000 枚以上、本ガイドでは 1,000 枚/種類、合計 15,000 の写真を使用しています。
F：ええーっ。それは大変じゃないですか？
A：とても大変です。ですが赤ちゃんや子供が初めてものを認識することを想像してください。「これはマンゴーだ！」と分かるためには、何度も何度もマンゴーを見る必要があります。しかしそれが難しいな、時間がないなと思われる方にはもう少し簡単な方法をご紹介できます。
F：ほっ。簡単な方法もあるのですね。
A：はい。少し注意点もあるのでご紹介していきますね。では次の章にいきましょうか。
F：はい。よろしくお願いします。

次は、**3.3 集めた AI 学習データを保存しよう**　へ進んでください。

3.2　データセットを使用する

WEB 上にデータセットが存在します。そちらをご自由にご利用いただけます。ただし、**画質がよくないものや商業用途で使用できない**ものもあります。注意して利用してください。

URL

フードメニューの画像※

https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/
http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz

その他

https://www.tensorflow.org/datasets/catalog/overview?hl=ja

※商業用途で利用するには、それぞれの画像の所有者に確認をとる必要があります。

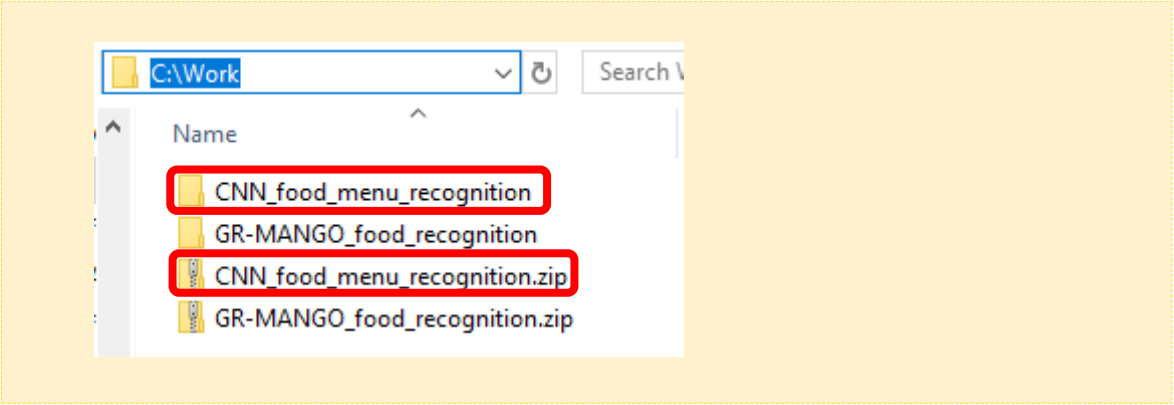
3.3　集めた AI 学習データを保存しよう

それでは、前章で集めた学習データを保存していきます。

１．本パッケージのフードメニュー認識の AI 学習用環境一式の zip を解凍

CNN_food_menu_recognization.zip を解凍します。

例は、C ドライブに Work というフォルダを作っています。



２．画像データを入れるフォルダの名前変更し写真を格納

指定されたフォルダに収集したカテゴリごとのフォルダ名を変更します。

青枠の `train` と緑枠の `valid` の両方で同じように変更してください。

作成したフォルダに写真を格納します。

COLUMN
なんで同じフォルダを 2 個ずつ作るの？train と valid って？

AI の学習モデルを作るときは、最初に AI 学習のトレーニングを行い、同時に学習がきちんと進んでいるかを確かめるために、検証も行います。トレーニングで使う画像データ(train)と検証で使う画像データ(valid)は分離することが一般的です。今回はビギナー編ではトレーニング用に 75%、検証用に 25%（トレーニングを 12,000 枚、検証を 3,000 枚）という比に分けています。用意できた枚数で割合を決めて写真をいれてください。

4. AI 学習モデルを作ってみよう

4.1 Python ファイルを変更してみよう

food_menu_recognition.py のコードを変更していきます。

4.1.1 カテゴリを変えてみよう

Before

```
Label_list = ['caesar_salad','club_sandwich','donuts',
  'edamame','french_fries','fried_rice','hamburger',
  'hot_dog','oysters',
  'pancakes','pizza','ramen',
  'spaghetti_carbonara','steak','sushi']
```

After

```
Label_list = ['beer','club_sandwich','donuts',
  'edamame','french_toast','fried_rice','hamburger',
  'hot_dog','oysters',
  'pancakes','pizza','ramen',
  'spaghetti_meatsauce','steak','sushi']
```

4.1.2 学習データの枚数の変え方

Before

```
train_data_num = 12000
valid_data_num =  3000
test  data  num   =  1000
```

After

```
train_data_num =  1200
valid_data_num =   300
test_data_num   =   100
```

4.1.3 CNN の変え方(1)

Conv2D の層を 11→13 に変えたいときの例を書きます。

Before

```
model = Sequential()

model.add(Conv2D(8,    (3, 3), padding = 'same', activation
                = 'relu', input_shape = (data_height, data_width, data_channel)))
model.add(Conv2D(8,    (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(AveragePooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))
#model.add(BatchNormalization())

model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu'))
```

After

```
model = Sequential()

model.add(Conv2D(8,    (3, 3), padding = 'same', activation
                = 'relu', input_shape = (data_height, data_width, data_channel)))
model.add(Conv2D(8,    (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(8,    (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(AveragePooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))
#model.add(BatchNormalization())

model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu'))
model.add(AveragePooling2D(pool_size = (2, 2)))
```

他の部分のコードも変更可能です。"e-AI starting package の 2.AI 学習ガイド"や"Keras の公式サイト"の内容を参考に行ってみてください。

URL Keras Document https://keras.io/ja/

4.1.4 CNN の変え方(2)

よく使用される MaxPooling2D を例に書きます。

Before

```
model = Sequential()

model.add(Conv2D(8,    (3, 3), padding = 'same', activation
                = 'relu', input_shape = (data_height, data_width, data_channel)))
model.add(Conv2D(8,    (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(AveragePooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))
#model.add(BatchNormalization())

model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu'))
```

After

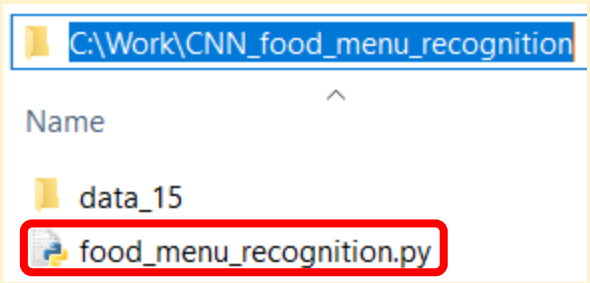
```
model = Sequential()

model.add(Conv2D(8,    (3, 3), padding = 'same', activation
                = 'relu', input_shape = (data_height, data_width, data_channel)))
model.add(Conv2D(8,    (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
#model.add(BatchNormalization())

model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(16, (3, 3), padding = 'same', activation = 'relu'))
model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

他の部分のコードも変更可能です。"e-AI starting package の 2.AI 学習ガイド"や"Keras の公式サイト"の内容を参考に行ってみてください。

URL Keras Document https://keras.io/ja/

4.2　AI 学習モデルを作ってみよう

下記の food_menu_recognition.py ファイルを実行していきます。



① Windows メニューから「コマンドプロンプト」を起動してください。

② ディレクトリを移動します。以下のコマンドを実行してください。（例のワークディレクトリは C:¥Work です）

```
cd C:¥Work¥CNN_food_menu_recognition
```

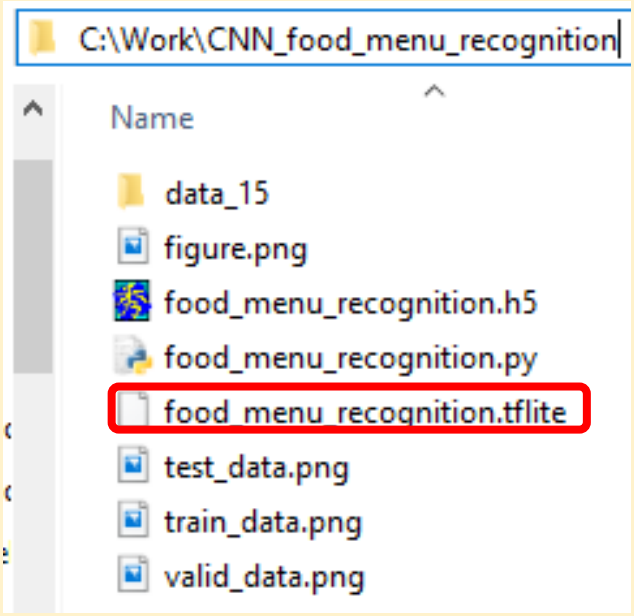
③ Python ファイルを実行します。

```
py -3.5 food_menu_recognition.py
```

以下のようなログが出てきます。時間は今回ビギナー編のものは WindowsPC で 24 時間かかりました。

```
C:¥Users¥          >cd C:¥CNN_food_menu_recognition

C:¥CNN_food_menu_recognition>py -3.5 food_menu_recognition.py
15
Found 12000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
2020-12-07 16:24:55.473630: I tensorflow/core/platform/cpu_feature
ctions that this TensorFlow binary was not compiled to use: AVX2
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 8)       224
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 8)       584
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 16)      1168
_________________________________________________________________
average_pooling2d (AveragePo (None, 64, 64, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 16)        2320
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 16)        2320
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
average_pooling2d_1 (Average (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
average_pooling2d_2 (Average (None, 16, 16, 32)        0
```

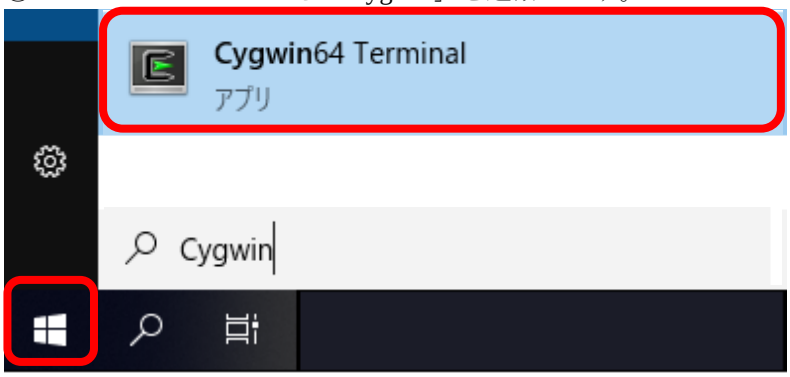
④ AI モデル(Tensorflow lite micro format)完成
AI の学習の実行が成功すると food_menu_recognition.tflite が生成されます。
figure.png には AI 学習の推移を表すグラフが、test_data.png、train_data.png、valid_data.png にはテスト、トレーニング、評価で使用されたデータの画像の一部が入っています。



5. AI 学習モデルをソースファイルに変えてみよう

5 章で作成した"**food_menu_recognition.tflite**"をソースコードに変えていきます。tflite 形式のファイルをソースファイルに変換するために xxd コマンドという Linux コマンドを Windows 上で実行します。

① Windows メニューから「Cygwin」を起動します。



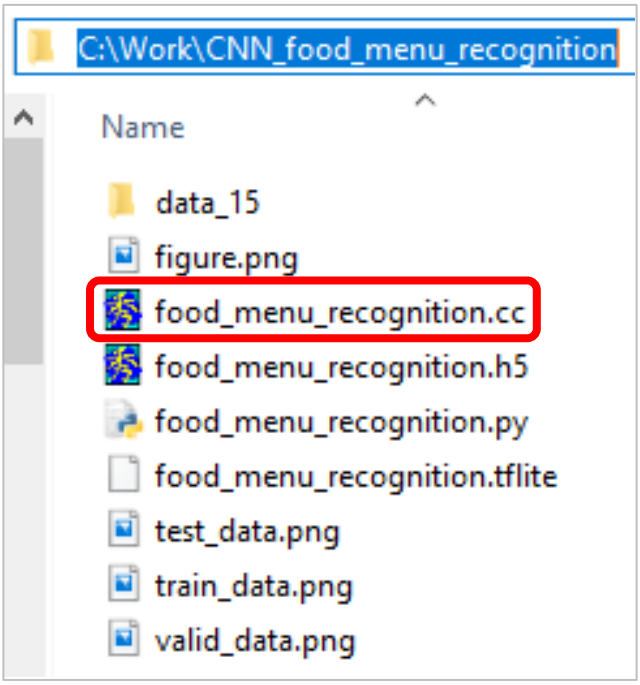
② food_menu_recognition.tflite が置いてあるフォルダまで移動します。下記コマンドを実行してください。

```
cd c:/Work¥CNN_food_menu_recognition
```

③ ".tflm"形式を".cc"ファイルに変換します。下記コマンドを実行してください。

```
xxd -I food_menu_recognition.tflite > food_menu_recognition.cc
```

④ food_menu_recognition.cc ファイルができあがります。



中を開いてみると、food_menu_recognition_tflite という重みデータの配列とその配列の長さを表す food_menu_recognition_tflite_len というデータが入っています。変更を行わないため、const をつけます。

Before ①

```
unsigned char food_menu_recognition_tflite[] = {
  0x20, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00,
  0x10, 0x00, 0x14, 0x00, 0x00, 0x00, 0x18, 0x00, 0x12, 0x00, 0x00, 0x00,
  0x03, 0x00, 0x00, 0x00, 0xc8, 0x4d, 0x09, 0x00, 0xf8, 0x14, 0x09, 0x00,
  0xe0, 0x14, 0x09, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00,
  0x04, 0x00, 0x08, 0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00,
  0x32, 0x00, 0x00, 0x00, 0x13, 0x00, 0x00, 0x00, 0x6d, 0x69, 0x6e, 0x5f,
  0x72, 0x75, 0x6e, 0x74, 0x69, 0x6d, 0x65, 0x5f, 0x76, 0x65, 0x72, 0x73,
  0x69, 0x6f, 0x6e, 0x00, 0x33, 0x00, 0x00, 0x00, 0x98, 0x14, 0x09, 0x00,
  0x88, 0x02, 0x09, 0x00, 0x80, 0x02, 0x09, 0x00, 0xf0, 0xfd, 0x08, 0x00,
                                  :
  0xf2, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x0a, 0x00, 0x0e, 0x00, 0x07, 0x00, 0x00, 0x00, 0x08, 0x00,
  0x0a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x0a, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00,
  0x0a, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00
};
unsigned int food_menu_recognition_tflite_len = 609944;
[EOF]
```

②

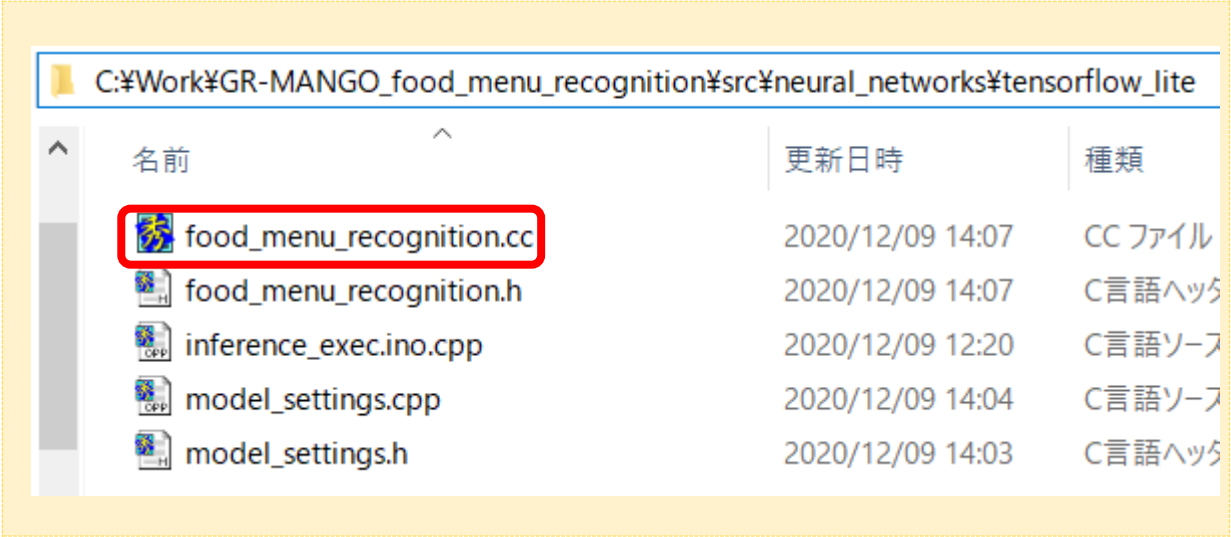
After

```
const unsigned char food_menu_recognition[] = {
  0x20, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00,
  0x10, 0x00, 0x14, 0x00, 0x00, 0x00, 0x18, 0x00, 0x12, 0x00, 0x00, 0x00,
  0x03, 0x00, 0x00, 0x00, 0xc8, 0x4d, 0x09, 0x00, 0xf8, 0x14, 0x09, 0x00,
  0xe0, 0x14, 0x09, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00,
  0x04, 0x00, 0x08, 0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00,
  0x32, 0x00, 0x00, 0x00, 0x13, 0x00, 0x00, 0x00, 0x6d, 0x69, 0x6e, 0x5f,
  0x72, 0x75, 0x6e, 0x74, 0x69, 0x6d, 0x65, 0x5f, 0x76, 0x65, 0x72, 0x73,
  0x69, 0x6f, 0x6e, 0x00, 0x33, 0x00, 0x00, 0x00, 0x98, 0x14, 0x09, 0x00,
  0x88, 0x02, 0x09, 0x00, 0x80, 0x02, 0x09, 0x00, 0xf0, 0xfd, 0x08, 0x00,
  0xe0, 0xd9, 0x08, 0x00, 0xd8, 0xd9, 0x08, 0x00, 0xd0, 0xd9, 0x08, 0x00,

  0xf2, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x0a, 0x00, 0x0e, 0x00, 0x07, 0x00, 0x00, 0x00, 0x08, 0x00,
  0x0a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x0a, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00,
  0x0a, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00
};
const unsigned int food_menu_recognition_len = 609944;
[EOF]
```

6. e²studio に組み込んでいこう

6.1 AI 学習モデルを変えてみよう

5 章で作成したファイルを以下のファイルに上書きします。



6.2 カテゴリを変えてみよう

下記のファイルを変更していきます。

6.2.1 inference_exec.h

Before

```
typedef struct
{
    uint32_t us_total;
    float    score;
    char     *category;
    float    caesar_salad_score;
    float    club_sandwich_score;
    float    donuts_score;
    float    edamame_score;
    float    french_fries_score;
    float    fried_rice_score;
    float    hamburger_score;
    float    hot_dog_score;
    float    oysters_score;
    float    pancakes_score;
    float    pizza_score;
    float    ramen_score;
    float    spaghetti_carbonara_score;
    float    steak_score;
    float    sushi_score;
    char     *price;
}
inference_result_t;
```

After

```
typedef struct
{
    uint32_t us_total;
    float    score;
    char     *category;
    float    beer_score;
    float    club_sandwich_score;
    float    donuts_score;
    float    edamame_score;
    float    french_fries_score;
    float    french_toast_score;
    float    hamburger_score;
    float    hot_dog_score;
    float    oysters_score;
    float    pancakes_score;
    float    pizza_score;
    float    ramen_score;
    float    spaghetti_meatsauce_score;
    float    steak_score;
    float    sushi_score;
    char     *price;
}
inference_result_t;
```

6.2.2 inference_exec.ino.cpp

Before

```
/* Process the inference results. */
inference_result.caesar_salad_score          = output->data.f[k_caesar_salad];
inference_result.club_sandwich_score         = output->data.f[k_club_sandwich];
inference_result.donuts_score                = output->data.f[k_donuts];
inference_result.edamame_score               = output->data.f[k_edamame];
inference_result.french_fries_score          = output->data.f[k_french_fries];
inference_result.fried_rice_score            = output->data.f[k_fried_rice];
inference_result.hamburger_score             = output->data.f[k_hamburger];
inference_result.hot_dog_score               = output->data.f[k_hot_dog];
inference_result.oysters_score               = output->data.f[k_oysters];
inference_result.pancakes_score              = output->data.f[k_pancakes];
inference_result.pizza_score                 = output->data.f[k_pizza];
inference_result.ramen_score                 = output->data.f[k_ramen];
inference_result.spaghetti_carbonara_score   = output->data.f[k_spaghetti_carbonara];
inference_result.steak_score                 = output->data.f[k_steak];
inference_result.sushi_score                 = output->data.f[k_sushi];
```

After

```
/* Process the inference results. */
inference_result.beer_score                  = output->data.f[k_beer];
inference_result.club_sandwich_score         = output->data.f[k_club_sandwich];
inference_result.donuts_score                = output->data.f[k_donuts];
inference_result.edamame_score               = output->data.f[k_edamame];
inference_result.french_fries_score          = output->data.f[k_french_fries];
inference_result.french_toast_score          = output->data.f[k_french_toast];
inference_result.hamburger_score             = output->data.f[k_hamburger];
inference_result.hot_dog_score               = output->data.f[k_hot_dog];
inference_result.oysters_score               = output->data.f[k_oysters];
inference_result.pancakes_score              = output->data.f[k_pancakes];
inference_result.pizza_score                 = output->data.f[k_pizza];
inference_result.ramen_score                 = output->data.f[k_ramen];
inference_result.spaghetti_meatsauce_score   = output->data.f[k_spaghetti_meatsauce];
inference_result.steak_score                 = output->data.f[k_steak];
inference_result.sushi_score                 = output->data.f[k_sushi];
```

6.2.3 model_settings.cpp

Before

```
/* Process the inference results. */
inference_result.caesar_salad_score          = output->data.f[k_caesar_salad];
inference_result.club_sandwich_score         = output->data.f[k_club_sandwich];
inference_result.donuts_score                = output->data.f[k_donuts];
inference_result.edamame_score               = output->data.f[k_edamame];
inference_result.french_fries_score          = output->data.f[k_french_fries];
inference_result.fried_rice_score            = output->data.f[k_fried_rice];
inference_result.hamburger_score             = output->data.f[k_hamburger];
inference_result.hot_dog_score               = output->data.f[k_hot_dog];
inference_result.oysters_score               = output->data.f[k_oysters];
inference_result.pancakes_score              = output->data.f[k_pancakes];
inference_result.pizza_score                 = output->data.f[k_pizza];
inference_result.ramen_score                  = output->data.f[k_ramen];
inference_result.spaghetti_carbonara_score   = output->data.f[k_spaghetti_carbonara];
inference_result.steak_score                 = output->data.f[k_steak];
inference_result.sushi_score                 = output->data.f[k_sushi];
```

After

```
/* Process the inference results. */
inference_result.beer_score                  = output->data.f[k_beer];
inference_result.club_sandwich_score         = output->data.f[k_club_sandwich];
inference_result.donuts_score                = output->data.f[k_donuts];
inference_result.edamame_score               = output->data.f[k_edamame];
inference_result.french_fries_score          = output->data.f[k_french_fries];
inference_result.french_toast_score           = output->data.f[k_french_toast];
inference_result.hamburger_score             = output->data.f[k_hamburger];
inference_result.hot_dog_score               = output->data.f[k_hot_dog];
inference_result.oysters_score               = output->data.f[k_oysters];
inference_result.pancakes_score              = output->data.f[k_pancakes];
inference_result.pizza_score                 = output->data.f[k_pizza];
inference_result.ramen_score                  = output->data.f[k_ramen];
inference_result.spaghetti_meatsauce_score   = output->data.f[k_spaghetti_meatsauce];
inference_result.steak_score                 = output->data.f[k_steak];
inference_result.sushi_score                 = output->data.f[k_sushi];
```

6.2.4 model_settings.h

Before

```
constexpr int kCategoryCount = 15;

// Index=15

constexpr int k_caesar_salad          = 0;
constexpr int k_club_sandwich         = 1;
constexpr int k_donuts                = 2;
constexpr int k_edamame               = 3;
constexpr int k_french_fries          = 4;
constexpr int k_fried_rice            = 5;
constexpr int k_hamburger             = 6;
constexpr int k_hot_dog               = 7;
constexpr int k_oysters               = 8;
constexpr int k_pancakes              = 9;
constexpr int k_pizza                 = 10;
constexpr int k_ramen                 = 11;
constexpr int k_spaghetti_carbonara   = 12;
constexpr int k_steak                 = 13;
constexpr int k_sushi                 = 14;
```

After

```
constexpr int kCategoryCount = 15;

// Index=15

constexpr int k_beer                  = 0;
constexpr int k_club_sandwich         = 1;
constexpr int k_donuts                = 2;
constexpr int k_edamame               = 3;
constexpr int k_french_fries          = 4;
constexpr int k_french_toast          = 5;
constexpr int k_hamburger             = 6;
constexpr int k_hot_dog               = 7;
constexpr int k_oysters               = 8;
constexpr int k_pancakes              = 9;
constexpr int k_pizza                 = 10;
constexpr int k_ramen                 = 11;
constexpr int k_spaghetti_meatsauce   = 12;
constexpr int k_steak                 = 13;
constexpr int k_sushi                 = 14;
```

6.3 CNN の変え方(1)

Conv2D の層数を変えたいときの例を記載します。

6.3.1 inference_exec.ino.cpp

Conv2D の層を 11→13 に変えたいときの例を書きます。**ただし、8.2 章に記載していない条件の conv2D を使用するときは、bit を 0 としてください。**

Before

```
void inference_exec_init(void) {

    static tflite::MicroErrorReporter micro_error_reporter;
    error_reporter = &micro_error_reporter;
    model = tflite::GetModel(food_menu_recognition_tflite);
    drp_en_conv_layer = 0x07ff;       /* DRP Layer enable bit */
    drp_conv_total_num = 0;
```

After

```
void inference_exec_init(void) {

    static tflite::MicroErrorReporter micro_error_reporter;
    error_reporter = &micro_error_reporter;
    model = tflite::GetModel(food_menu_recognition_tflite);
    drp_en_conv_layer = 0x1fff;       /* DRP Layer enable bit */
    drp_conv_total_num = 0;
```

１１層の場合
0x07fff → 0b0000 0111 1111 1111
Bitが11個 "1"

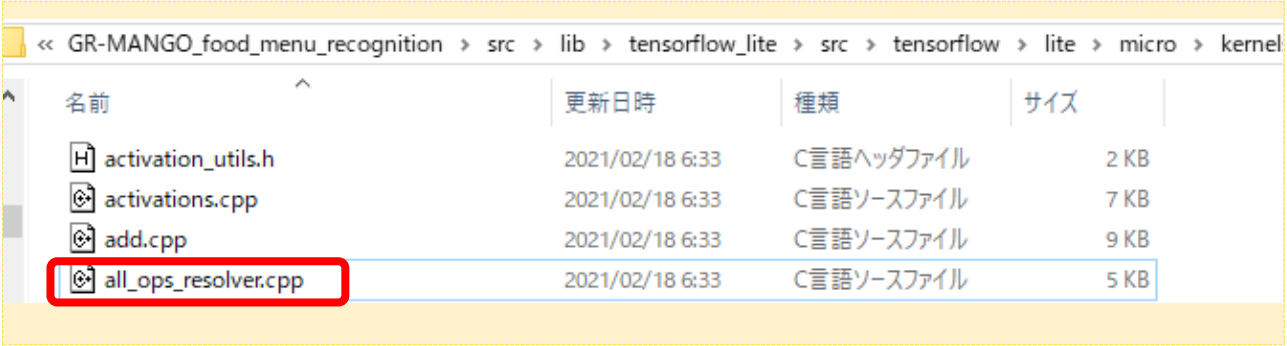
１３層の場合
0x1ffff → 0b0001 1111 1111 1111
Bitが13個 "1"

ただし、8.2章に記載していない条件のconv2Dを使用するときは、
bitを0としてください。
例えば、13層目が条件に合わないconv2Dを使用する場合は0x0fffとしてください。

6.4 CNN の変え方(2)

使用するオペレーションを増やしたいときの例を記載します。

6.4.1 all_ops_resolver.cpp



使用されることが多い MaxPooling を例に書きます。

Before

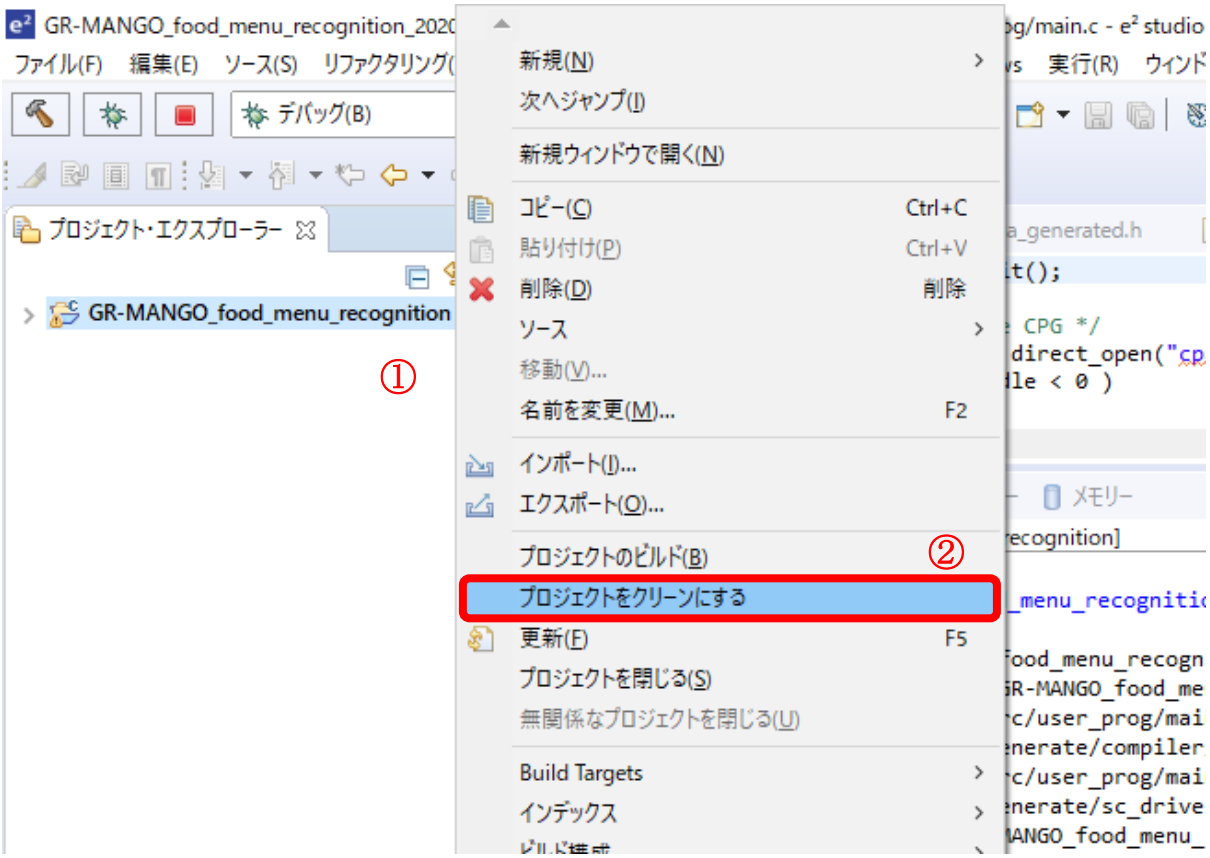
```
AllOpsResolver::AllOpsResolver() {
  AddBuiltin(BuiltinOperator_FULLY_CONNECTED, Register_FULLY_CONNECTED(), 1, 4);
// AddBuiltin(BuiltinOperator_MAX_POOL_2D, Register_MAX_POOL_2D(), 1, 2);
  AddBuiltin(BuiltinOperator_SOFTMAX, Register_SOFTMAX(), 1, 2);
//  AddBuiltin(BuiltinOperator_LOGISTIC, Register_LOGISTIC(), 1, 2);
//  AddBuiltin(BuiltinOperator_SVDF, Register_SVDF(), 1, 3);
  AddBuiltin(BuiltinOperator_CONV_2D, Register_CONV_2D(), 1, 3);
//  AddBuiltin(BuiltinOperator_CONCATENATION, Register_CONCATENATION(), 1, 3);
//  AddBuiltin(BuiltinOperator_DEPTHWISE_CONV_2D, Register_DEPTHWISE_CONV_2D(), 1, 3);
    AddBuiltin(BuiltinOperator_AVERAGE_POOL_2D, Register_AVERAGE_POOL_2D(), 1, 2);
```

After

```
AllOpsResolver::AllOpsResolver() {
  AddBuiltin(BuiltinOperator_FULLY_CONNECTED, Register_FULLY_CONNECTED(), 1, 4);
  AddBuiltin(BuiltinOperator_MAX_POOL_2D, Register_MAX_POOL_2D(), 1, 2);
  AddBuiltin(BuiltinOperator_SOFTMAX, Register_SOFTMAX(), 1, 2);
//  AddBuiltin(BuiltinOperator_LOGISTIC, Register_LOGISTIC(), 1, 2);
//  AddBuiltin(BuiltinOperator_SVDF, Register_SVDF(), 1, 3);
  AddBuiltin(BuiltinOperator_CONV_2D, Register_CONV_2D(), 1, 3);
//  AddBuiltin(BuiltinOperator_CONCATENATION, Register_CONCATENATION(), 1, 3);
//  AddBuiltin(BuiltinOperator_DEPTHWISE_CONV_2D, Register_DEPTHWISE_CONV_2D(), 1, 3);
    AddBuiltin(BuiltinOperator_AVERAGE_POOL_2D, Register_AVERAGE_POOL_2D(), 1, 2);
```

7. オリジナルのフードメニューを AI で自動認識

1. GR-MANGO で AI_ビギナーガイド.pdf を参考に e²studio を立ち上げます。
2. 赤枠上で①右クリックし、②プロジェクトをクリーンにする　をクリック。



3. 赤枠上で③右クリックし、②プロジェクトのビルド　をクリック。

4. GR-MANGO で AI_ビギナーガイド.pdf を参考にロードし、実行します。

5. 以下のようなデモが表示されます。

8. Appendix

8.1 カラー画像の AI 処理のメモリ使用量が少ないのはなぜ？

TFLM を使っています

TFLM とは、TensorFlow Lite for Microcontrollers の略称です。Google が提供しており**メモリが限られる**マイクロコントローラなどのデバイス上で機械学習モデルを実行するように設計された、TensorFlow Lite の移植版の**オープンソース**です。TFLM には量子化対応できる機能があり、約 1/4 のメモリ削減効果を期待することができます。いくつか種類があり今回はわずかなモデルの精度の低下を伴いますが、モデルの大きさを削減できる訓練後の量子化 post training integer quantized を使用しています。
本パッケージに含まれるソースコードはアップデートされる可能性があります。下記の URL 情報からご確認ください。

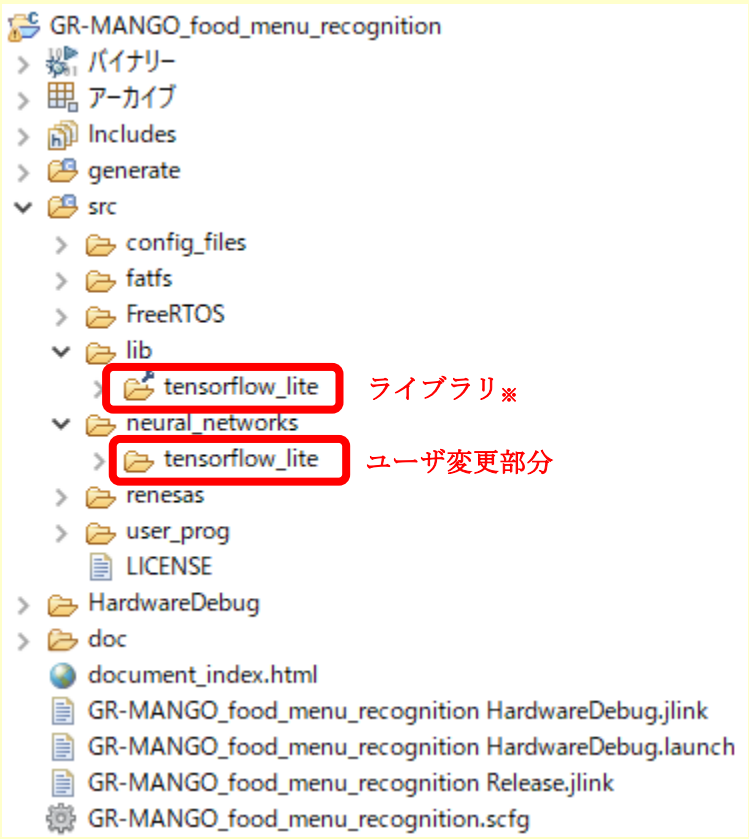
用語

TensorFlow ： 機械学習向けに Google が開発した OSS platform
TensorFlow Lite： デバイス上での推論を可能にする OSS Deep Learning framework

URL

TFLM： **https://www.tensorflow.org/lite/microcontrollers**
TensorFlow： **https://www.tensorflow.org/?hl=ja**

Souce Code



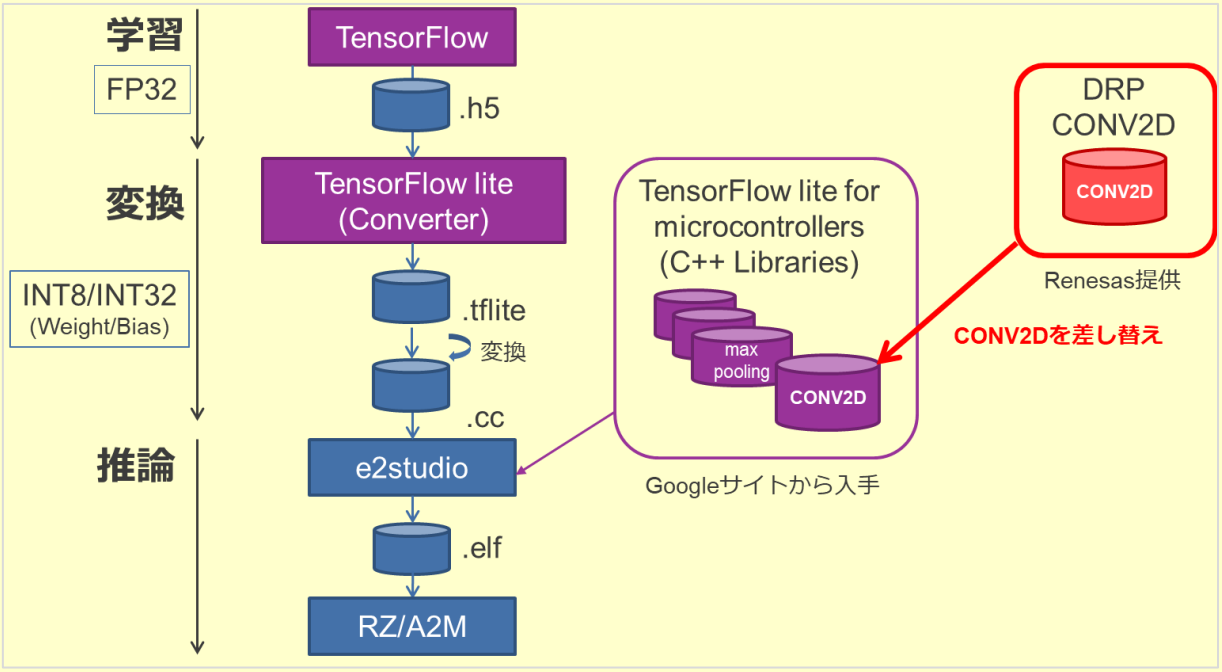
※ 一部ルネサスエレクトロニクスで変更を加えています。コメントでその旨追記しています。

8.2　カラー画像の AI 処理が 400msec 弱で行えるのはなぜ？

CNN 処理に DRP ライブラリを使用して高速化しています

CNN のコンボリューション層を DRP ライブラリに差し替えることで CPU の約 5 倍高速で AI 処理を行うことを実現しています。

Flow



DRP ライブラリ CONV2D の対応パラメータ

特徴マップ X サイズ、Y サイズ	4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256
ich(入力チャネル数)	3, 4, 8, 12, 16, 20, 24, 28, 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224
och(出力チャネル数)	4, 8, 12, 16, 20, 24, 28, 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224
特徴マップ X サイズ、Y サイズ、入出力チャンネルの制約	X サイズ x Y サイズ x och ≦ 1M(1,048,576) ich ≦ och

(例)パラメータ組み合わせ

X x Y x och = 256 x 256 x 16 ≦ 1M, ich = 8 ≦ 16 OK
X x Y x och = 16 x 16 x 224 ≦ 1M, ich = 112 ≦ 224 OK
X x Y x och = 160 x 120 x 8 ≦ 1M, ich = 3 ≦ 8 OK

DRP ライブラリ CONV2D のフィルタ仕様

・サイズ　3x3 固定、ストライド　X 方向 1 固定、Y 方向 1 固定
・パディング　0 固定、・特徴マップの入力/出力サイズ　SAME 固定

8.3　商用利用される際の注意点

Apache License2.0 のライブラリを使用しています

8.1 でご説明しました TFLM は Apache License2.0 です。商用利用される前に必要な内容をご確認ください。

URL

Apache License 2.0：　https://www.apache.org/licenses/LICENSE-2.0

※本パッケージでは、一部ルネサスエレクトロニクスで変更を加えた部分があり、ソースコードのコメントにその旨が明記されています。

8.4　このパッケージを作ろうと思ったのはなぜ？

　このパッケージを作ろうと思ったころ、世界的に大きな病気が流行っていました。セルフレジなどの自動化が進めば多くの人が不要不急の外出を避けることができるのではと思いました。買い物をしていた時に、バーコードの貼られていないリンゴを見たときに、バーコードリーダーで AI 認識も同時にできたら、バーコードを貼る手間も省け、レジも混まず、色々な方のお役にたてるのではと思いました。
（もしかしたら、もう実現し普及している地域や国もあるかもしれません。）
　AI を始めるのはとてもハードルが高いように思われています。そのハードルを下げるために、簡単に変更ができるようにこのガイドを作りました。
　多くの人に役立つリファレンスを作っていただけることを心から願っています。

2020.12.24

9. 改版履歴

リビジョン	日付	内容
1.1	2021.02.18	4.1.4　6.4　オペレーションの変更・追加方法を記載
1.0	2020.12.24	New