MAiX MAniaX

aNo 研　著

2020-01-21 版　　発行

はじめに

▲図 1　aNo 研ロボット作品

最近、電子工作・Maker の間で、Maix というマイコンボードが話題になっています。MAiX ボードは、RISC-V の CPU とディープラーニングのアクセラレータを搭載している Kendryte K210 という SoC を搭載しています。Kendryte K210 の登場で、パソコンやクラウド環境でないと動かすことができなかったディープラーニングを、小さいマイコンボードの中で動かすことを可能にしました。

本書は、MaixPy の基礎からニューラルネットワークの学習モデルの作り方まで、MAiX を使い倒すためのテクニックをたくさん盛り込んでいます。筆者が感じている「Maix」の面白さを使えることができれば、望外の喜びです。

あなたも本書と MAiX で、AIOT の世界を体験してみませんか！？

ソースコードのダウンロード先

本書に記載しているソースコードは、Github で公開しています。

https://github.com/anoken/maix_maniax

目次

第 1 章

MAiX の紹介

1.1 SiPeed MAiX とは

SiPEED は、中国 深センのスタートアップ企業です。SiPEED は「サイピード」、MAix は「マックス」と呼びます。SiPEED は、Kendryte K210 を搭載した SiPEED MAix シリーズをリリースしています。現在、Kendryte K210 を内蔵した「MAix M1」を搭載したプロトタイプ開発ボード「MAix M1 Dock」「Maixduino」「Maix BIT」、ビジネス向けの顔認証デバイス「MaixFace」などをリリースしてます。

SiPEED は、画像処理のような負荷の高い処理をエッジ (ローカル端末）で行い、ネットワーク上のコンピュータへは必要なデータだけを送信する、「AOIT」という方針を打ち出しています。

「MAix M1」に ESP8255 を搭載して Bluetooth や Wi-Fi 通信機能を追加した「MAix M1w」、「MAix M1w」を搭載した「Maix GO」「MAix M1 Dock」もリリースされているのですが、技適が取得されておらず、日本では使うことができません。「Maixduino」というボードは、外付けで ESP32 を取り付ける構成で技適を取得しており、MAix の中で唯一日本で無線通信ができるボードです。本書は、「Maixduino」ボードを中心にして説明をしていきます。

SiPEED は、MAix 以外にも、FPGA ボードの Lichee Tang シリーズなど魅力的な承認ををリリースしています。

SiPEED 公式サイト：https://www.sipeed.com/

1.2 Kendryte K210 とは？

Kendryte K210 は、Canaan Creative が製造・販売している RISC-V の Soc(System-on-a-chip) です。2018/09 に、Canaan Creative は、RISC-V モジュールの Kendryte K210 をリリースしました。Canaan Creative が会社の名前

で、Kendryte はブランド名です。Canaan Creative は、北京に拠点を置いており、マイニング (仮想通貨採掘）のハードウェア AISC を開発していることで有名な企業です。

Kendryte は「ケンドライト」、Canaan Creative は「カナンクリエイティブ」と呼びます。

第 2 章

Maixduino の開発環境

2.1 K210 の開発環境

Maix ボードは、Sipeed が提供する MicroPython 環境の「MaixPy」、Arduino 環境の「Maxduino」、Kendryte が提供する cmake のライブラリ「kendryte-standalone-sdk」などの開発環境が提供されています。

Sipeed は、MaixPy の機能拡張に力を入れており、積極的に機能拡張や不具合の解消が図られています。MaixPy の内部では、kendryte-standalone-sdk を呼び出している構成です。

本書では、MaixPy での開発を説明していきます。

2.2 MaixPy のファームウェアを書き込む

MaixPy でプログラミングをしていくためには、あらかじめ MaixPy のファームウェアを書き込む必要があります。

書き込むツールには、kflash GUI が提供されています。kflash GUI は、書き込むための設定を変更でき、カスタマイズしたファームウェアや学習ファイルを Maix のフラッシュメモリへ書き込むことができます。

2.2.1 ファームウェアのダウンロード

MaixPy のファームウェアは、Sipeed のダウンロードページから、ダウンロードすることができます。バージョン毎に作られたディレクトリの下の、「maixpy_XXXX.bin」と名前のついたファイルをダウンロードします。

本書執筆時点 (2020/2) では、時点では maixpy_v0.5.0_12_g284ce83 が最新です。

MaixPy ファームウェアのダウンロードページ : `http://dl.sipeed.com/MAIX/`

MaixPy/release/master/

2.2.2 kflash GUI でファームウェアを書き込む

ダウンロードした kflash GUI の Zip ファイルを解凍して、「kflash_gui.exe」を実行します。Maixduino と PC とを USB で接続し、ボードを設定をした上で「Downloads」を押すと書き込むことができます。

本書執筆時点 (2020/2) では、時点では v1.5.3 が最新です。Windows の場合は、「kflash_gui_V1.5.3_windows.7z」をダウンロードします。

https://github.com/sipeed/kflash_gui/releases

2.2.3 MaixPy IDE のダウンロード

MaixPyIDE は、Sipeed のダウンロードページからダウンロードします。

Sipeed のダウンロードページの v0.2.4 の下には、ファームウェア更新を周知するために、注意書きがかかれた readme ファイルだけが入っています。ダウンロードのファイルは、「____」→「v0.2.4」の下に置いてあります。少しわかりにくいですね。

本書執筆時点 (2020/2) では、時点では v0.2.4 が最新です。

Sipeed MaixPy IDE ダウンロードページ：

http://dl.sipeed.com/MAIX/MaixPy/ide/

▲図 2.1　MaixPy IDE のダウンロード

2.2.4 MaixPy IDE の使い方

MaixPy IDE を起動し、Maix ボードと接続してみましょう。最初にボードの設定を、Maixduino に変更します。「ツール」→「Select Board」→「Maixduino」を選択します。

▲図 2.2　MaixPy IDE の設定

MaixPy IDE で Python プログラムを実行する

▲図 2.3　MaixPy IDE での接続

MaixPy IDE の左下の接続ボタンを押し、Maixduino との接続を確立した後に、スクリプトを実行を押すと、MaxiPy IDE で表示している Python のプログラムを

実行します。

▲図 2.4　boot.py へ書き込む

MaixPy IDE と繋がっていないときにも、Python プログラムを実行するには、Maixduino のフラッシュメモリに書き込まれている「boot.py」の書き換えをします。「ツール」→「Save open scripout to borad(boot.py)」を選択すると、デフォルトで起動する boot.py を MaixPy IDE で書いている Python のプログラムに更新します。

MaixPy IDE でターミナルを起動する

MaixPy IDE でプログラムを実行するには、ターミナルを使うという方法も用意されています。筆者はこちらの方法をよく使います。ターミナルでは、Python のプログラムを実行するだけでなく、Python のコマンドでファイルを書き換えたり、ヘルプを呼んだりすることができます。

ターミナルを起動するには、Maixduino が繋がっている COM ポートを MaixPy IDE のターミナルに登録します。

「ツール」→「ターミナルを開く」→「新ターミナル」→「シリアルポートに接続する」→「COMXX(Maixduino が繋がっているポートを選択)」→「115200」

▲図 2.5　MaixPy IDE のターミナル設定

ターミナルを開きます。「ツール」→「ターミナルを開く」→「シルアルポート」を選びます。

▲図 2.6　MaixPy IDE のターミナルを開く

新しく Window が立ち上がり、コンソールに MAIXPY と表示され、ターミナルが立ち上がります。ターミナル上部の、実行ボタンを押すことで、MaixPy IDE に書かれているプログラムを実行します。

▲図 2.7　MaixPy IDE でターミナル起動

シリアルターミナル上部の、停止ボタンを押すと、MaixPy のコマンドを手打ちで入力するモードに切り替わります。

▲図 2.8　MaixPy IDE のコマンドを入力する

第 3 章

MaixPy でプログラミング

Maixpy には LCD とカメラを扱うためのクラスや、MicroPython で画像処理を扱うためのライブラリ OpenMV が組み込まれており、簡単に画像処理や LCD の表示を取り扱うことができます。

3.1 カメラの画像を LCD に表示する

▲図 3.1 LCD に表示

M5StickV のカメラの画像を LCD に表示してみました。M5StickV は、カメラと LCD の取り付けの関係で LCD の表示を 180 度回さないと上下が反転してしまいます。

```
import lcd

lcd.init(type=1, freq=15000000, color=lcd.BLACK)
        type: LCD のタイプ (0 : なし、1 : 液晶シールド)
```

```
        freq： LCD の周波数（実際には SPI の通信速度です）
        color：LCD 初期化の色。0xFFFF などの 16 ビット RGB565 カラー、
                または（236、36、36）などの RGB888 カラー、デフォルトは lcd.BLACK

lcd.rotation(dir)
        dir：0 から 3 まで数値で、時計回りの回転を指定します
```

```
import sensor

sensor.set_pixformat(format)
        k210 は、rgb565 および yuv422 形式をサポートしています。
        MaixPy 開発ボードで推奨される設定は RGB565 形式です。

sensor.set_framesize(framesize)
        カメラの出力フレームサイズを設定すします。
        k210 は最大で VGA フォーマットをサポートし、VGA よりも大きい場合、
        画像を取得できません。 MaixPy 開発ボードの推奨設定は QVGA 形式です。

import image
img = sensor.snapshot()
        画像オブジェクトを返します

import lcd
lcd.display(image, roi=Auto)
        LCD に画像オブジェクト（GRAYSCALE または RGB565）を表示します。
        roi は（x、y、w、h）で指定します。指定ない場合は、画像のサイズです

        roi の幅が LCD より小さい場合、空いている領域を黒で塗りつぶします
        roi の幅が LCD より大きい場合、LCD の大きさに切り取り、中心寄せします
        roi の高さが LCD より低い場合は、空いている領域を黒で塗りつぶします
        roi の高さが LCD より大きい場合、LCD の大きさに切り取り、中心寄せします
```

▼リスト 3.1　009_camera_capture.py

```
import sensor,image,lcd
lcd.init()
lcd.rotation(2)
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)
while True:
    img=sensor.snapshot()
    lcd.display(img)
```

3.2 カメラの画像を SD カードに保存する

M5StickV で撮影した画像を、SD カードに保存します。A ボタンを押すと SD カードに保存、B ボタンを押すと SD カードから写真を読み出して表示するプログラムを作ってみました。

```
import image
image.save(path, quality=95)
        path:画像の保存先のファイル名を入力します。
        画像フォーマットは、bmp/pgm/ppm/jpg/jpeg をサポートしています。
        quality:Jpeg で保存する場合、画質を指定します。画質が低いほど画像サイズが小
さくなります。
```

▼リスト 3.2　010_camera_save_sdcard.py

```
import sensor, image,lcd,os
from fpioa_manager import fm
fm.register(board_info.BUTTON_A, fm.fpioa.GPIO1)
but_a=GPIO(GPIO.GPIO1, GPIO.IN, GPIO.PULL_UP)
fm.register(board_info.BUTTON_B, fm.fpioa.GPIO2)
but_b = GPIO(GPIO.GPIO2, GPIO.IN, GPIO.PULL_UP)
is_button_a = 0
is_button_b = 0

lcd.init()
lcd.rotation(2)
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)

path = "save_"
ext=".jpg"
cnt=0
img_read = image.Image()
print(os.listdir())

while True:
    if is_button_b == 1:
        lcd.display(img_read)

    else :
        img=sensor.snapshot()
        lcd.display(img)

    if but_a.value() == 0 and is_button_a == 0:
        print("save image")
        cnt+=1
        fname=path+str(cnt)+ext
```

```
            print(fname)
            img.save(fname, quality=95)
            is_button_a=1

        if but_a.value() == 1 and is_button_a == 1:
            is_button_a=0

        if but_b.value() == 0 and is_button_b == 0:
            fname=path+str(cnt)+ext
            print(fname)
            img_read = image.Image(fname)
            is_button_b=1

        if but_b.value() == 1 and is_button_b == 1:
            is_button_b=0
```

3.3 動画を録画する

3.4 動画を再生する

3.5 スピーカから音をだす

3.6 マイクで録音する

3.7 FFT で周波数を解析する

3.8 マルチスレッドで並列実行する

3.9 ESP32 へ UART で通信する

3.10 ESP32 と SPI で通信する

参考情報

MaixPy Documentation
https://maixpy.sipeed.com/en/
MaixPy_scripts
https://github.com/sipeed/MaixPy_scripts
OpenMV documentation
http://docs.openmv.io/

第 4 章

KPU でニューラルネットワークを作る

4.1 学習データ kmodel を作成する

▲図 4.1　畳み込みニューラルネットワーク

Kendryte K210 の最大の特徴は、KPU(Knowldge Processing Unit) という、NN アクセラレータを持っていることで、畳み込みニューラルネットワーク (CNN) などのアルゴリズムを高速に実行することができます。KPU で CNN を実行するためには、事前に学習データ kmodel を用意する必要があります。学習データの kmodel の作り方について説明していきます。

4.2 Windows Subsystem for Linux のインストール

KPU の学習データを作る場合、Linux 環境がお勧めです。KPU の学習データのツールは、Windows や Mac の環境のものも提供されているのですが、バージョンが古かったり、不具合があった時に情報を調べてもユーザが少なく、なかなか情報が出てこないという問題があります。LinuxPC を使われている方はよいのですが、WindowsPC を使われている方は、Windows で Linux のバイナリを実行できる、Windows Subsystem for Linux をインストールしてみましょう。

Windows の、「コントロールパネル」→「プログラムと機能」から、「Windows の機能の有効化または無効化」を選択します。「Windows の機能の有効化または無効化」のメニューの中から、「Windows Subsystem for Linux」を探し、チェックを入れます。インストールが始まるので、終わるまで待ちます。インストールが終わったら再起動します。

▲図 4.2　Windows Subsystem for Linux のインストール

▲図 4.3　Microsoft Store で Ubuntu のインストール

「Microsoft Store」から、「Ubuntu」を検索して入手ボタンをクリックします。

スタートに追加される Ubuntu を起動して、インストールの完了を待ちます。完了すると、内部で使用するユーザー名とパスワードの設定を求められるので、プロンプトに従って入力します。ここまでで、 Windows Subsystem for Linux の準備ができました。

▲図 4.4　Windows Subsystem for Linux のインストール

4.3 TensorFlow / Keras をインストールする

TensorFlow は、Google が開発している、ディープラーニングや機械学習を開発するためのソフトウェアライブラリです。Keras は、TensorFlow 上で動くニューラルネットワークライブラリの 1 つで、比較的短いソースコードでニューラルネットワークを実装することができます。TensorFlow は、組み込み機器やモバイル端末動作させるための TensorFlow Lite という環境を用意しています。Kendryte は、TensorFlow Lite 向けに作成した学習データは、Kendryte K210 の学習データに変換するためのツールを用意しています。

仮想環境でも動くように、本書では GPU を使わない設定にしています。筆者は、ネイティブの Ubuntu18.04 と、Windows Subsystem for Linux の Ubuntu18.04 で動作確認をしています。

まず最初に、Miniconda をインストールして、Python の環境を作ります。Miniconda のインストーラは、Miniconda のウェブサイトからダウンロードして取得します。

Miniconda : https://docs.conda.io/en/latest/miniconda.html

```
# wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
# sh https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

Miniconda 上で、Python、TensorFlow、Keras などをインストールします。

```
conda create -n ml python=3.6 tensorflow=1.14 keras pillow \
              numpy pydot graphviz
```

conda をアクティブ化します。conda のアクティブ化する作業は、シェル立ち上げるごとに必要です。conda は、隔離した Python 環境を作成してくれるので、システムの Python 環境を誤って変更することがなく、安心してプログラミングができます。

```
conda activate ml
```

4.4 TensorFlow / Keras の GPU 版をインストールする

NVIDIA Driver のインストール

nvidia のドライバを入れてない (nvidia-smi コマンドが無い) 場合は，ドライバのインストールを行う．

```
$ sudo ubuntu-drivers devices
$ sudo ubuntu-drivers autoinstall
```

インストールが終了したら再び nvidia-smi コマンドを実行し，ドライバのバージョンや GPU の詳細が表示されていることを確認する

keras/tensorflow のインストール

```
conda install tensorflow-gpu==1.14 keras-gpu python==3.6
conda install pillow numpy pydot graphviz
```

```
import tensorflow as tf
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
tf.keras.backend.set_session(tf.Session(config=config))
```

https://qiita.com/hirotow/items/6233266b5fe970203ec5

4.4.1 学習データ変換ツール nncase をインストールする

▲図 4.5 nncase

nncase github: https://github.com/kendryte/nncase/
nncase は、Keras や TensorFlow で作成した学習データを、KPU の学習データ kmodel へ変換します。Kendryte がリリースしており、GitHub からダウンロードできます。nncase は、CNN の MobileNetV1/V2 や YOLOV1/YOLOV3 をサポートしています。筆者執筆時点 (2019/9) では、nccase は v0.1.0-rc5 がリリースされています。

```
# mkdir ./ncc
# cd ./ncc
# wget https://github.com/kendryte/nncase/releases/download/v0.1.0-rc5/
ncc-linux-x86_64.tar.xz
# tar -Jxf ncc-linux-x86_64.tar.xz
```

例えば、TensorFlow の学習モデル「my.tflite」を nncase で KPU の学習データに変換するには、次のコマンドを入力します。dataset は画像が入っているフォルダを指定し、画像からダイナミックレンジ補正を行います。M5StickV で撮影した画像を入れておきましょう。

```
#  ./ncc/ncc -i tflite -o k210model --dataset images my.tflite my.kmodel
```

nncase は、コマンドの引数でオプションを設定します。

-i,--input-format	必須. 入力ファイルのフォーマットを指定します
-o,--output-format	必須. 出力ファイルのフォーマットを指定します
--dataset	必須. データセット（画像データ）のパスを指定します
--dataset-format	(Default: image) データセットのフォーマットです
--help	ヘルプを表示します
--version	Version を表示します
input (pos. 0)	オプション指定ない引数の 0 個目で、入力ファイルの path を指定します
output (pos. 1)	オプション指定ない引数の 1 個目で、出力ファイルの path を指定します

nncase を Python から呼び出す場合は、subprocess を使ってコマンドを呼び出します。

```
import subprocess
subprocess.run(['./ncc/ncc','my.tflite','my.kmodel','-i',
'tflite','-o','k210model','--dataset','images'])
```

参考情報

Train, Convert, Run MobileNet on Sipeed MaixPy and MaixDuino

http://blog.sipeed.com/p/680.html

Maixpy GO Mobilenet Transfer learning for Image Classfication

https://iotdiary.blogspot.com/2019/07/maixpy-go-mobilenet-transfer-learning.html

Image Recognition With Sipeed MaiX and Arduino IDE/Micropython

https://www.instructables.com/id/Transfer-Learning-With-Sipeed-MaiX-and-Arduino-IDE/

4.5 KPU でニューラルネットワークを使ってみる

Kendryte K210 は、ニューラルネットワークプロセッサ KPU を搭載しており、低消費電力で畳み込みニューラルネットワーク（Convolutional Neural Network、CNN）を計算できます。たとえば、顔やオブジェクトを検出および分類したり、検出されたオブジェクトのサイズ、座標、タイプを取得することができます。

Kendryte K210 は以下のような特徴があります。

・いくつかの制限がありますが、固定小数点モデルをサポートしています。

・ネットワーク層の数に直接的な制限はなく、入力および出力チャネルの数、入力データの幅と高さなど、畳み込みニューラルネットワークパラメーターの各レイヤーの個別に設定できます。

・1x1 および 3x3 のサイズの畳み込みカーネルをサポートしています。

・複数の活性化のための機能をサポートします

・リアルタイム作業でサポートされる最大のニューラルネットワークのパラメーターサイズは 5.5MiB?5.9MiB です。

・非リアルタイムで作業する場合の最大ネットワークパラメーターサイズは、フラッシュ容量からファームウェアのサイズを引いた容量のメモリに収まる限り使うことができます。

4.5.1 学習データを M5StickV の SD カードに入れる。

KPU の処理には、学習データが必要です。Sipeed が用意した KPU の学習データを、ダウンロードしてみましょう。

「MaixPy v0.3 demo firmware.zip」というファイルをダウンロードします。

MaixPy v0.3.0 ダウンロード:

http://dl.sipeed.com/MAIX/MaixPy/release/maixpy_v0.3.0/

▲図 4.6　M5StickV

Zip ファイルを解凍すると
・mnist.kmodel：mnist の学習データ (mnist)
・facedetect.kmodel：顔検出の学習データ (YOLOv2)
・20class.kmodel：20 クラス分類の学習データ (YOLOv2)
・mbnet75.kmodel：1000 クラス分類の学習データ (mobilenet)
という学習ファイルが入っています。

この学習ファイルを MicroSD カードに書き込み、M5StickV に MicroSD カードを差してみましょう。

MicroSD カードに格納されているファイルの名前は、MaixPy から一覧を参照することができます。

▼リスト 4.1　013_filelist.py

```
import os
devices = os.listdir("/")

if "flash" in devices:
    os.chdir("/flash")
    print("flash")
    print(os.listdir())
if "sd" in devices:
    os.chdir("/sd")
    print("sd")
    print(os.listdir())
```

```
sd
['mbnet75.kmodel', 'mnist.kmodel', 'facedetect.kmodel', '20class.kmodel']
flash
```

```
[ 'boot.py']
```

4.5.2 MNIST で数字を見分ける

▲図 4.7　KPU で MNIST

MNIST とは、画像のクラス分け問題の入門として有名な、手書き数字のデータセットです。M5StickV に MNIST で学習したデータセットを読み込ませると、手書きの数字を読み取って数字がいくつであるか、認識することができます。

```
import KPU as kpu

task = kpu.load(offset or file_path)
        Kmodel を読み込みます。kpu のネットワークオブジェクトを返します。
        offtset:フラッシュメモリに格納する場合、
        0xd00000 のように先端のアドレスを指定します
        file_path：ファイルシステムに格納する場合、
          "/sd/xxx.kmodel"のようにファイルのパスを指定します。

fmap=kpu.forward(task,img,layers)
        読み込まれたロードされたネットワークモデルを計算し、
        ターゲットレイヤーでの特徴マップを出力します
        kpu_net: kpu のネットワークオブジェクトを入力します
        image_t: カメラの画像データを入力します
        int: specifies the number of layers to calculate to the network
        fmap：特徴マップを返します
```

▼リスト 4.2　014_kpu_camera_mnist.py

```
import sensor,lcd,image
import KPU as kpu
lcd.init()
lcd.rotation(2)
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((224, 224))
task = kpu.load("mnist.kmodel")
sensor.run(1)
while True:
    img = sensor.snapshot()
    lcd.display(img)                #カメラ画像を表示します
    img1=img.to_grayscale(1)        #グレースケールに変換します
    img2=img1.resize(28,28)         #28x28 にリサイズします
    a=img2.invert()                 #反転します
    a=img2.strech_char(1)           #前処理を行います
    lcd.display(img2,oft=(120,32))      #28x28 画像を表示します
    a=img2.pix_to_ai();             #ai データに変換します
    fmap=kpu.forward(task,img2)         #KPU で演算します
    plist=fmap[:]                   #10 個の数字の確率を取り出します
    pmax=max(plist)                 #最も確率の高いものを取り出します
    max_index=plist.index(pmax)         #数字を求めます
    lcd.draw_string(0,0,"%d: %.3f"%(max_index,pmax),lcd.WHITE,lcd.BLACK)
```

4.5.3 YOLOv2 でお顔を見つける

▲図 4.8　KPU で YOLOv2 の顔検出

YOLOv2 は、CNN を使った物体検出のアルゴリズムです。YOLO とは、「You Only Look Once」(あなたを 1 回だけ見る)、の略で、英語圏では「You only live once」(人生一度きり) という有名なことわざから引用しています。YOLOv2 は、画像の中に物体を見つけたら、「バウンディングボックス」で物体の位置と大きさを出

力します。この顔検出のアプリケーションは、M5StickV 出荷時にインストールされているものです。YOLOv2 の顔検出は、とても早く、かつ高精度に顔を検出することができます。

```
kpu.init_yolo2(task, threshold, nms_value, anchor_num, anchor)
        task: kpu のネットワークオブジェクトを入力します
        threshold: 推論の閾値です
        nms_value: bounding 探索の閾値です
        anchor_num: YOLOv2 は anchor と呼ばれるアスペクト比一定の探索用
bounding box を持ちます。anchor の数を指定します。
        anchor:anchor のパラメータを指定します


list = kpu.run_yolo2(task, img)
        YOLOv2 で物体検出を行います。
        list：検出した物体のバウンディングボックスのリストを返します
        kpu_net : kpu のネットワークオブジェクトを入力します
        image_t ：画像データを入力します
```

お顔が検出された時に、M5StickV の赤い LED を点灯するプログラムを用意しました。

▼リスト 4.3　015_kpu_camera_face_detect.py

```
import sensor,image,lcd
import KPU as kpu
from fpioa_manager import *
from Maix import GPIO
from board import board_info
from fpioa_manager import fm
fm.register(board_info.LED_R, fm.fpioa.GPIO4)
led_r = GPIO(GPIO.GPIO4, GPIO.OUT)
led_r.value(1)

lcd.init()
lcd.rotation(2)
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)
task = kpu.load("facedetect.kmodel")

anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437,
        6.92275, 6.718375, 9.01025)
a = kpu.init_yolo2(task, 0.5, 0.3, 5, anchor)
img_lcd=image.Image()
while(True):
    img = sensor.snapshot()
    code = kpu.run_yolo2(task, img)
```

```
    face_detec=False
    if code:
        for i in code:
            a = img.draw_rectangle(i.rect())
            face_detec=True
    if face_detec:
        led_r.value(0)
    else:
        led_r.value(1)

    a = lcd.display(img)
a = kpu.deinit(task)
```

4.5.4 YOLOv2 で 20 種類の物体を見分ける

▲図 4.9 KPU の YOLOv2 でクラス分類

YOLOv2 で、猫・人・テレビなど、20 種類の物体を見分けるように訓練した学習データが用意されています。20 種類にクラス分類する学習データを使い、あなたの身近なものを撮影して、どんな物体がどんなものと認識されるか試してみましょう。顔検出ほとんど同じソースコードで、学習データを差し替えるだけで、今度は、20 種類のクラス分類のアプリケーションを M5StickV で動かすことができます。

▼リスト 4.4 016_kpu_camera_20class.py

```
import sensor,image,lcd,time
import KPU as kpu

lcd.init()
lcd.rotation(2)
```

```
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)
clock = time.clock()
classes = ['aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car',
 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person',
 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor']
task = kpu.load("20class.kmodel")
anchor = (1.08, 1.19, 3.42, 4.41, 6.63, 11.38, 9.42, 5.11, 16.62, 10.52)
a = kpu.init_yolo2(task, 0.5, 0.3, 5, anchor)
while(True):
    clock.tick()
    img = sensor.snapshot()
    code = kpu.run_yolo2(task, img)
    print(clock.fps())
    if code:
        for i in code:
            a=img.draw_rectangle(i.rect())
            a = lcd.display(img)
            for i in code:
                lcd.draw_string(i.x(), i.y(), classes[i.classid()],
                                lcd.RED, lcd.WHITE)
                lcd.draw_string(i.x(), i.y()+12, '%f1.3'%i.value(),
                                lcd.RED, lcd.WHITE)
    else:
        a = lcd.display(img)
a = kpu.deinit(task)
```

4.5.5 MobileNet で 1000 種類の物体を見分ける

▲図 4.10　KPU1000 種類のクラス分類

MobileNet という有名なデータセットでは、深層学習のために、なんと 1000 種類の物体をたくさん撮影した画像が用意されています。Sipeed では、1000 種類の物体

を見分けるように訓練した学習データを用意しています。M5StickV で身近なものを色々撮影して、正しく物体を認識してくれるのか試してみましょう。

```
import KPU as kpu
fmap=kpu.forward(task,img,layer_No)
        畳み込み積分を行います
        kpu_net: kpu のネットワークオブジェクトを入力します
        image_t: カメラからの画像データを入力します
        layer_No: ネットワークの特定のレイヤーを指定することができます
        fmap: 特徴量ベクトルを出力します
```

▼リスト 4.5　007_camera_1000class.py

```
import sensor, image, lcd, time
import KPU as kpu
lcd.init()
lcd.rotation(2)
lcd.clear()

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((224, 224))
sensor.run(1)
lcd.draw_string(100,96,"MobileNet Demo")
lcd.draw_string(100,112,"Loading labels...")
f=open('labels.txt','r')
labels=f.readlines()
f.close()
task = kpu.load("mbnet751.kmodel")
clock = time.clock()
while(True):
    img = sensor.snapshot()
    clock.tick()
    fmap = kpu.forward(task, img)
    fps=clock.fps()
    plist=fmap[:]
    pmax=max(plist)
    max_index=plist.index(pmax)
    a = lcd.display(img)
    lcd.draw_string(10, 96, "%.2f:%s"%(pmax, labels[max_index].strip()))
    print(fps)
a = kpu.deinit(task)
```

4.6　スピーカから音声ファイルを再生する

M5StickV はスピーカを搭載しており、WAV ファイルを再生することができます。M5StickV に内蔵されているスピーカアンプ MAX98357 とは、I2S という通

信インターフェイスで繋がっています。SD カードの WAV ファイルを読み出して、M5StickV の A ボタンを押したタイミングで再生するプログラムを作ってみました。

I2S はスピーカやマイクと接続するために使われ、Kendryte k210 は 3 つの I2S デバイスと繋げることができます。1 つの I2S デバイスは、4 つのチャンネルに対応しています。よくスピーカで 2ch や 5.1ch と呼ばれる商品があるように、1 つのデバイスに複数のチャンネルがぶら下がる構成になっています。I2S を使う前には、ピンを割り付けする必要があります。

```
from Maix import I2S
i2s_dev = I2S(device_num)
        device_num：K210 の I2S デバイス番号を指定します。最大 3 です。

i2s_dev.channel_config(channel, mode, resolution, cycles, align_mode)
        channel：I2S チャンネル番号
        mode：チャネル送信モード、受信モードを選びます
        resolution：チャネルの解像度、つまり受信データビット数を指定します
        cycles：通信データのクロック周期を指定します。
        align_mode：チャネルのアライメントモードを指定します

i2s_dev.set_sample_rate(sample_rate)
        サンプルレートを指定します

i2s_dev.play(audio)
        audio スピーカで再生をします。
```

▼リスト 4.6　018_play_wav.py.py

```
from fpioa_manager import *
from Maix import I2S, GPIO
import audio

def init_wav():                          ##スピーカの初期化
    fm.register(board_info.SPK_SD, fm.fpioa.GPIO0)
    spk_sd=GPIO(GPIO.GPIO0, GPIO.OUT)
    spk_sd.value(1)
    fm.register(board_info.SPK_DIN,fm.fpioa.I2S0_OUT_D1)
    fm.register(board_info.SPK_BCLK,fm.fpioa.I2S0_SCLK)
    fm.register(board_info.SPK_LRCLK,fm.fpioa.I2S0_WS)
    wav_dev = I2S(I2S.DEVICE_0)

def play_wav(fname):                     ##WAV ファイルの再生
    player = audio.Audio(path = fname)
    player.volume(20)
    wav_info = player.play_process(wav_dev)
    wav_dev.channel_config(wav_dev.CHANNEL_1,
        I2S.TRANSMITTER,resolution = I2S.RESOLUTION_16_BIT,
        align_mode = I2S.STANDARD_MODE)
    wav_dev.set_sample_rate(wav_info[1])
    while True:
```

```
        ret = player.play()
        if ret == None:
            break
        elif ret==0:
            break
    player.finish()

fm.register(board_info.BUTTON_A, fm.fpioa.GPIO1)
but_a=GPIO(GPIO.GPIO1, GPIO.IN, GPIO.PULL_UP)
but_a_pressed = 0

while True:
    if but_a.value() == 0 and but_a_pressed == 0:
        play_sound("reset.wav")
        but_a_pressed=1
    if but_a.value() == 1 and but_a_pressed == 1:
        but_a_pressed=0

player.finish()
```

4.7 MobileNet とは？

1000 個の画像を分類する

ハイパーパラメータ 0.75 と 0.5 との比較　　ILSVRC2012_img_train のダウンロード

転移学習で食べ物の分類をする

Food-101 を分類する。Confusion Matrix

4.8 YOLO とは？

4.8.1 raccoon データセット

```
#! /usr/bin/env python
#from __future__ import print_function
from keras.models import Sequential, Model
from keras.layers import Reshape, Activation, Conv2D, Input, MaxPooling2D, BatchNormalizatio
from keras.layers.advanced_activations import LeakyReLU
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard
from keras.optimizers import SGD, Adam, RMSprop
from keras.layers.merge import concatenate
import matplotlib.pyplot as plt
import keras.backend as K
import tensorflow as tf
```

```
import imgaug as ia
from tqdm import tqdm
from imgaug import augmenters as iaa
import numpy as np
import pickle
import os, cv2
from preprocessing import parse_annotation, BatchGenerator
from keras.applications import MobileNet

LABELS =["aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "co


IMAGE_H, IMAGE_W = 224, 224
GRID_H,  GRID_W  = 7 , 7
BOX              = 5
CLASS            = len(LABELS)
CLASS_WEIGHTS    = np.ones(CLASS, dtype='float32')
OBJ_THRESHOLD    = 0.3#0.5
NMS_THRESHOLD    = 0.3#0.45
ANCHORS          = [0.57273, 0.677385, 1.87446, 2.06253, 3.33843, 5.47434, 7.88282, 3.52778,

NO_OBJECT_SCALE  = 1.0
OBJECT_SCALE     = 5.0
COORD_SCALE      = 1.0
CLASS_SCALE      = 1.0

BATCH_SIZE       = 4
WARM_UP_BATCHES  = 0
TRUE_BOX_BUFFER  = 50

wt_path = 'yolo.weights'
train_image_folder = '/mnt/e/LinuxHome/K210/DATASET/VOC_maix_20classes/VOCdevkit/VOC2007/JPE
train_annot_folder = '/mnt/e/LinuxHome/K210/DATASET/VOC_maix_20classes/VOCdevkit/VOC2007/Ann
valid_image_folder = '/mnt/e/LinuxHome/K210/DATASET/VOC_maix_20classes/VOCdevkit/VOC2007/JPE
valid_annot_folder = '/mnt/e/LinuxHome/K210/DATASET/VOC_maix_20classes/VOCdevkit/VOC2007/Ann



input_image = Input(shape=(IMAGE_H, IMAGE_W, 3))
true_boxes  = Input(shape=(1, 1, 1, TRUE_BOX_BUFFER , 4))

mobilenet = MobileNet(input_shape=(IMAGE_H, IMAGE_W, 3),alpha = 0.75,depth_multiplier = 1, d
 weights = "imagenet", classes = 1000, include_top=False,
 #backend=keras.backend, layers=keras.layers,models=keras.models,utils=keras.utils
 )
x = mobilenet(input_image)

# Layer 23
x = Conv2D(BOX * (4 + 1 + CLASS), (1,1), strides=(1,1), padding='same', name='conv_23')(x)
output = Reshape((GRID_H, GRID_W, BOX, 4 + 1 + CLASS))(x)

# small hack to allow true_boxes to be registered when Keras build the model
# for more information: https://github.com/fchollet/keras/issues/2790
#output = Lambda(lambda args: args[0])([output, true_boxes])

model = Model([input_image, true_boxes], output)
```

```
model.summary()



def custom_loss(y_true, y_pred):
    mask_shape = tf.shape(y_true)[:4]

    cell_x = tf.to_float(tf.reshape(tf.tile(tf.range(GRID_W), [GRID_H]), (1, GRID_H, GRID_W,
    cell_y = tf.transpose(cell_x, (0,2,1,3,4))

    cell_grid = tf.tile(tf.concat([cell_x,cell_y], -1), [BATCH_SIZE, 1, 1, 5, 1])

    coord_mask = tf.zeros(mask_shape)
    conf_mask  = tf.zeros(mask_shape)
    class_mask = tf.zeros(mask_shape)

    seen = tf.Variable(0.)
    total_recall = tf.Variable(0.)

    """
    Adjust prediction
    """
    ### adjust x and y
    pred_box_xy = tf.sigmoid(y_pred[..., :2]) + cell_grid

    ### adjust w and h
    pred_box_wh = tf.exp(y_pred[..., 2:4]) * np.reshape(ANCHORS, [1,1,1,BOX,2])

    ### adjust confidence
    pred_box_conf = tf.sigmoid(y_pred[..., 4])

    ### adjust class probabilities
    pred_box_class = y_pred[..., 5:]

    """
    Adjust ground truth
    """
    ### adjust x and y
    true_box_xy = y_true[..., 0:2] # relative position to the containing cell

    ### adjust w and h
    true_box_wh = y_true[..., 2:4] # number of cells accross, horizontally and vertically

    ### adjust confidence
    true_wh_half = true_box_wh / 2.
    true_mins    = true_box_xy - true_wh_half
    true_maxes   = true_box_xy + true_wh_half

    pred_wh_half = pred_box_wh / 2.
    pred_mins    = pred_box_xy - pred_wh_half
    pred_maxes   = pred_box_xy + pred_wh_half

    intersect_mins  = tf.maximum(pred_mins,  true_mins)
    intersect_maxes = tf.minimum(pred_maxes, true_maxes)
    intersect_wh    = tf.maximum(intersect_maxes - intersect_mins, 0.)
    intersect_areas = intersect_wh[..., 0] * intersect_wh[..., 1]
```

```
true_areas = true_box_wh[..., 0] * true_box_wh[..., 1]
pred_areas = pred_box_wh[..., 0] * pred_box_wh[..., 1]

union_areas = pred_areas + true_areas - intersect_areas
iou_scores  = tf.truediv(intersect_areas, union_areas)

true_box_conf = iou_scores * y_true[..., 4]

### adjust class probabilities
true_box_class = tf.argmax(y_true[..., 5:], -1)

"""
Determine the masks
"""
### coordinate mask: simply the position of the ground truth boxes (the predictors)
coord_mask = tf.expand_dims(y_true[..., 4], axis=-1) * COORD_SCALE

### confidence mask: penelize predictors + penalize boxes with low IOU
# penalize the confidence of the boxes, which have IOU with some ground truth box < 0.6
true_xy = true_boxes[..., 0:2]
true_wh = true_boxes[..., 2:4]

true_wh_half = true_wh / 2.
true_mins    = true_xy - true_wh_half
true_maxes   = true_xy + true_wh_half

pred_xy = tf.expand_dims(pred_box_xy, 4)
pred_wh = tf.expand_dims(pred_box_wh, 4)

pred_wh_half = pred_wh / 2.
pred_mins    = pred_xy - pred_wh_half
pred_maxes   = pred_xy + pred_wh_half

intersect_mins  = tf.maximum(pred_mins,  true_mins)
intersect_maxes = tf.minimum(pred_maxes, true_maxes)
intersect_wh    = tf.maximum(intersect_maxes - intersect_mins, 0.)
intersect_areas = intersect_wh[..., 0] * intersect_wh[..., 1]

true_areas = true_wh[..., 0] * true_wh[..., 1]
pred_areas = pred_wh[..., 0] * pred_wh[..., 1]

union_areas = pred_areas + true_areas - intersect_areas
iou_scores  = tf.truediv(intersect_areas, union_areas)

best_ious = tf.reduce_max(iou_scores, axis=4)
conf_mask = conf_mask + tf.to_float(best_ious < 0.6) * (1 - y_true[..., 4]) * NO_OBJECT_

# penalize the confidence of the boxes, which are reponsible for corresponding ground tr
conf_mask = conf_mask + y_true[..., 4] * OBJECT_SCALE

### class mask: simply the position of the ground truth boxes (the predictors)
class_mask = y_true[..., 4] * tf.gather(CLASS_WEIGHTS, true_box_class) * CLASS_SCALE

"""
Warm-up training
"""
```

```
    no_boxes_mask = tf.to_float(coord_mask < COORD_SCALE/2.)
    seen = tf.assign_add(seen, 1.)

    true_box_xy, true_box_wh, coord_mask = tf.cond(tf.less(seen, WARM_UP_BATCHES),
                          lambda: [true_box_xy + (0.5 + cell_grid) * no_boxes_mask,
                                   true_box_wh + tf.ones_like(true_box_wh) * np.reshape(ANCH
                                   tf.ones_like(coord_mask)],
                          lambda: [true_box_xy,
                                   true_box_wh,
                                   coord_mask])

    """
    Finalize the loss
    """
    nb_coord_box = tf.reduce_sum(tf.to_float(coord_mask > 0.0))
    nb_conf_box  = tf.reduce_sum(tf.to_float(conf_mask  > 0.0))
    nb_class_box = tf.reduce_sum(tf.to_float(class_mask > 0.0))

    loss_xy    = tf.reduce_sum(tf.square(true_box_xy-pred_box_xy)     * coord_mask) / (nb_co
    loss_wh    = tf.reduce_sum(tf.square(true_box_wh-pred_box_wh)     * coord_mask) / (nb_co
    loss_conf  = tf.reduce_sum(tf.square(true_box_conf-pred_box_conf) * conf_mask)  / (nb_co
    loss_class = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=true_box_class, logit
    loss_class = tf.reduce_sum(loss_class * class_mask) / (nb_class_box + 1e-6)

    loss = loss_xy + loss_wh + loss_conf + loss_class

    """
    Debugging code
    """
    nb_true_box = tf.reduce_sum(y_true[..., 4])
    nb_pred_box = tf.reduce_sum(tf.to_float(true_box_conf > 0.5) * tf.to_float(pred_box_conf

    current_recall = nb_pred_box/(nb_true_box + 1e-6)
    total_recall = tf.assign_add(total_recall, current_recall)

#    loss = tf.Print(loss, [tf.zeros((1))], message='Dummy Line \t', summarize=1000)
#    loss = tf.Print(loss, [loss_xy], message='Loss XY \t', summarize=1000)
#    loss = tf.Print(loss, [loss_wh], message='Loss WH \t', summarize=1000)
#    loss = tf.Print(loss, [loss_conf], message='Loss Conf \t', summarize=1000)
#    loss = tf.Print(loss, [loss_class], message='Loss Class \t', summarize=1000)
#    loss = tf.Print(loss, [loss], message='Total Loss \t', summarize=1000)
#    loss = tf.Print(loss, [current_recall], message='Current Recall \t', summarize=1000)
#    loss = tf.Print(loss, [total_recall/seen], message='Average Recall \t', summarize=1000)

    return loss



generator_config = {
    'IMAGE_H'         : IMAGE_H,
    'IMAGE_W'         : IMAGE_W,
    'GRID_H'          : GRID_H,
    'GRID_W'          : GRID_W,
    'BOX'             : BOX,
    'LABELS'          : LABELS,
    'CLASS'           : len(LABELS),
    'ANCHORS'         : ANCHORS,
```

```
    'BATCH_SIZE'      : BATCH_SIZE,
    'TRUE_BOX_BUFFER' : 50,
}

train_imgs, seen_train_labels = parse_annotation(train_annot_folder, train_image_folder, lab

def normalize(image):
    image = image / 255.
    return image

### read saved pickle of parsed annotations
#with open ('train_imgs', 'rb') as fp:
#    train_imgs = pickle.load(fp)
train_batch = BatchGenerator(train_imgs, generator_config, norm=normalize)

valid_imgs, seen_valid_labels = parse_annotation(valid_annot_folder, valid_image_folder, lab
### write parsed annotations to pickle for fast retrieval next time
#with open('valid_imgs', 'wb') as fp:
#    pickle.dump(valid_imgs, fp)

### read saved pickle of parsed annotations
#with open ('valid_imgs', 'rb') as fp:
#    valid_imgs = pickle.load(fp)
valid_batch = BatchGenerator(valid_imgs, generator_config, norm=normalize, jitter=False)

optimizer = Adam(lr=0.5e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
#optimizer = SGD(lr=1e-4, decay=0.0005, momentum=0.9)
#optimizer = RMSprop(lr=1e-4, rho=0.9, epsilon=1e-08, decay=0.0)

model.compile(loss=custom_loss, optimizer=optimizer)

history=model.fit_generator(generator        = train_batch,
                    steps_per_epoch  = len(train_batch),
                    epochs           = 30,
                    verbose          = 1,
                    max_queue_size   = 3)

model.save('my_yolo.h5')


np.savetxt("model_top_loss.csv", history.history['loss'])
#np.savetxt("model_top_val_loss.csv", history.history['val_loss'])
#np.savetxt("model_top_acc.csv", history.history['acc'])
#np.savetxt("model_top_val_acc.csv", history.history['val_acc'])
```

https://www.hackster.io/dmitrywat/object-detection-with-sipeed-maix-boards-kendryte-k210-421d55

第 5 章

第 2 章　ESP32 開発環境

5.1　ESP32 の開発環境

Maixduino は BlueTooth・Wifi 通信をするために、ESP32 が実装されていて、UART・SPI で K210 と ESP32 との間の通信を行うことができます。

本章では、MaixDuino での K210 と ESP32 との間の通信方法について説明します。

https://www.espressif.com/en/support/download/at?keys=&field_type_tid%5B%5D=1

https://www.espressif.com/en/support/download/other-tools

ESP32 SPI https://github.com/espressif/esp-idf/blob/v3.2.2/examples/peripherals/spi_s https://docs.espressif.com/projects/esp-idf/en/v3.2.2/api-reference/peripherals/spi_slave.

・UART:K210 → ESP32 ・UART:ESP32 → K210

・SPI:K210 → ESP32 ・SPI:ESP32 → K210

UART https://gist.github.com/anoken/700d19493fffe368fdb683e5979d4290

Maixduino 仕様書 https://docid81hrs3j1.cloudfront.net/medialibrary/2019/07/Sipeed_M

K210-IO25 ESP32-IO5 ESP32-SPI_CS K210-IO26 ESP32-IO23 SPI0_MISO K210-IO27 ESP32-IO18 SPI0_SCLK K210-IO28 ESP32-IO14 SPI0_MOSI K210-IO9 IO25 ESP32_READY IO8 Dedicated pin ESP32_EN

machine.SPI

https://maixpy.sipeed.com/zh/libs/machine/spi.html# https://github.com/sipeed/Maix

SPI（Serial Peripheral Interface）は、マスターとスレーブで構成される同期シリアルプロトコルです。

標準の 4 線モードは、SCK（SCLK）、CS（チップセレクト）、MOSI、MISO の 4 線で構成されています。

K210 では、SPI には次の特性があります。

SPI デバイスは 4 つあり、そのうち SPI0、SPI1、SPI3 はマスターモードでのみ機

能し、SPI2 はスレーブモードでのみ機能します MaixPy では、SPI3 は SPI フラッシュへの接続に使用されています。 次に、オープンインターフェイスと SPI Flash タイムシェアリング多重化を検討します。1/2/4/8 線式全二重モードをサポート MaixPy では、現在標準（Motorola）の 4 線式全二重モード（SCK、MOSI、MISO、CS 4 ピン）のみをサポート最大伝送速度：45M DMA をサポート 4 つの構成可能なハードウェアチップの選択

class machine.SPI(id, mode=SPI.MODE_MASTER, baudrate=500000, polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck, mosi, miso, cs0, cs1, cs2, cs3) 指定されたパラメーターで新しい SPI オブジェクトを作成します

1.1。パラメーター id：SPI ID、値の範囲 [0,3]、現在 0 と 1 のみをサポートし、マスターモードのみ可能、2 はスレーブとしてのみ使用可能、現在実装されていない、3 予約済みモード：SPI モード、MODE_MASTER または MODE_MASTER_2 または MODE_MASTER_4 または MODE_MASTER_8 または MODE_SLAVE、現在 MODE_MASTER のみがサポートされていますボーレート：SPI ボーレート（周波数）極性：極性。値は 0 または 1 で、アイドル時の SPI の極性を示します。0 は低レベルを表し、1 は高レベルを表します phase：フェーズ値は 0 または 1 で、クロックの最初または 2 番目のエッジでデータが収集されることを示します 0 は 1 番目、1 は 2 番目です。ビット：データ幅、デフォルト値は 8、値の範囲 [4,32] firstbit：MSB または LSB の順に送信するかどうかを指定しますデフォルトは SPI.MSB です。sck：ピン値を直接送信できる SCK（クロック）ピン値の範囲は [0,47] です。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。mosi：MOSI（ホスト出力）ピン。ピン値を直接送信できます。値の範囲：[0,47]。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。miso：MISO（ホスト入力）ピン。ピン値を直接送信できます。値の範囲：[0,47]。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。cs0：CS0（チップセレクト）ピン。ピン値を直接渡すことができます。値の範囲：[0,47]。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。cs1：CS1（チップセレクト）ピン。ピン値を直接渡すことができます。値の範囲：[0,47]。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。cs2：CS2（チップセレクト）ピン。ピン値を直接渡すことができます。値の範囲：[0,47]。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。cs3：ピン値を直接渡すことができる CS3（チップセレクト）ピン、値の範囲：[0,47]。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。d0・d7：現在予約されている非標準 4 線モードで使用されるデータピン。設定する代わりに、fm を使用してピンマッピングを均一に管理できます。

コンストラクターのような

SPI.init(id, mode=SPI.MODE_MASTER, baudrate=500000, polarity=0,

phase=0, bits=8, firstbit=SPI.MSB, sck, mosi, miso, cs0)

SPI.read(nbytes, write=0x00, cs=SPI.CS0)

nbytes：読み取る長さ cs：チップセレクトピンを選択します初期化中に cs0・cs3 にピンが設定されている場合、ここで選択する必要があるのは SPI.CS0・SPI.CS3 のみです。デフォルトは SPI.CS0 です。書き込み：全二重なので、読み取り時に MOSI ピンの値を設定しますデフォルト値は 0x00 で、常に低い値です。

変数へのデータの読み取り中にデータを送信、つまり全二重

SPI.write(write_buf, read_buf, cs=SPI.CS0) write_buf : 送信されるデータと長さを定義する bytearray タイプ read_buf : bytearray タイプ。受信データの保存場所を定義します cs：チップセレクトピンを選択します初期化中に cs0・cs3 にピンが設定されている場合、ここで選択する必要があるのは SPI.CS0・SPI.CS3 のみです。デフォルトは SPI.CS0 です。

SPI をログオフし、ハードウェアをリリースし、SPI クロックをシャットダウンします

SPI.deinit()

SPI0 : SPI 0 SPI1 : SPI 1 SPI2 : SPI 2 MODE_MASTER : マスターモードとして MODE_MASTER_2 : マスターモードとして MODE_MASTER_4 : マスターモードとして MODE_MASTER_8 : マスターモードとして MODE_SLAVE : スレーブモードとして MSB : MSB、つまり上位ビットまたは上位バイトが最初に送信されます LSB : LSB、つまり、下位ビットまたは下位バイトが最初に送信されます CS0 : チップセレクト 0 CS1 : チップセレクト 1 CS2 : チップセレクト 2 CS3 : チップセレクト 3

基本的な読み書き

from machine import SPI

spi1 = SPI(SPI.SPI1, mode=SPI.MODE_MASTER, baudrate=10000000, polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=28, mosi=29, miso=30, cs0=27) w = b'1234' r = bytearray(4) spi1.write(w) spi1.write(w, cs=SPI.CS0) spi1.write_readinto(w, r) spi1.read(5, write=0x00) spi1.readinto(r, write=0x00)

第 6 章

顔認識

Sipeed から Maix 用の顔認識モデルが公開されました。なんと、クラウドに繋ぐことなしにコンパクトなマイコンだけで、お顔を認識することができるようになります。

ソースコードはこちらです。https://gist.github.com/anoken/7fc193959d64f139a135e85946

#仕組み

お顔認証の仕組みは以下のような流れになっています。（あくまでソースコードからの予想）1.yolov2 のニューラルネットワークで顔部分を領域特定。2. 顔周辺エリアを切り取って、128x128 にリサイズ。3. 顔周辺から、ニューラルネットワークで目・鼻・口を特定する 4. 正面顔に幾何学変換 5.mobilenetv1 のニューラルネットワークで転移学習により、お顔を認識。

#Maix のシリアル番号の取得 Sipeed のサイトから、key gen をダウンロードします。http://blog.sipeed.com/p/1338.html#more-1338

KFlash で M5StickV に書き込みます。![03.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/5004de33-f8d4-1480-15cf-208287a0e911.jpeg)

MaixPy IDE のコンソールで、起動すると、シリアルコードが表示されます。コピーして保存しておきます。シリアルコードは Maix ボード個体毎に生成されるものです。

![04.png](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/26429e15-26e4-a4c9-2100-0dafc14ee03c.png)

Sipeed の学習モデルの入手

maixhub から、Sipeed の学習モデルを入手します。https://www.maixhub.com/

![05.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/e4fb4a35-b42b-fd61-8628-11215a0b13a3.jpeg)

ダウンロードにはシリアルコードが求められるので、Maix ボードから取得

したものを入力します。![06.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/93f081b6-c4c8-ee59-81b5-d405e981cfb8.jpeg)

以下の 3 つの K210 用のニューラルネットワークのモデルが入っています。![クリップボード 06.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/7837432b-c6b7-51cb-41fb-625d8ef00422.jpeg)

M5StickV への書き込み

M5StickV のファームウェアとニューラルネットワークのモデルを kflash で書き込みして、ソースコードを実行すると、お顔の認識が動かせる状態になり、、、、

ソースコードはこちらです。https://gist.github.com/anoken/7fc193959d64f139a135e85946

![02.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/fb98bdf2-dbd0-25e7-ab94-1ca6a1c4d332.jpeg)

**これだけでは動きません！！ **

#M5StickV のメモリ最適化

お顔認証は、3 つのニューラルネットワークのモデルを読み込む必要があり、Maix ボードのシステムヒープメモリに 3MB 以上の空きサイズが必要です。空きサイズは、kpu.memtest() コマンドで確認することができます。

![EKsm0BDUcAEHP3N.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/72b8867f-bcad-f54d-e5f6-8a37da08f922.jpeg)

M5StickV の MaixPy 最新版 (19/12/11 時点 ver0.4.0.92) では、システムヒープメモリの空きが 2.5MB しかありませんので、モデルを読み込むことができません。Sipeed の Maix ボードであれば、minimum 版のバイナリが提供されているのですが、M5Stickv はカメラや電源 IC が Maix ボードとは異なるので、minimum 版のバイナリでは色々と不都合があります。

![クリップボード 01.jpg](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/e4d77b5f-9cd4-c71a-b605-f73589e90067.jpeg)

そこで、M5StickV のファイムウェアをカスタマイイズし、メモリ最適化をします。

##環境以下の作業は Ubuntu(Linux) が必要です。Ubuntu18.04 (5.0.0-36-generic)

##ファームウェアのコンパイル

Maixpy のダウンロード Maixpy のファームウェアをソースコードからビルドします

“‘ $ git clone https://github.com/sipeed/MaixPy.git $ git submodule update --recursive --init “‘

Python のインストール Miniconda をインストールして、 Python の環境を作ります。Miniconda のインストーラは、 Miniconda のウェブサイトからダウンロードして取得します。

Miniconda : https://docs.conda.io/en/latest/miniconda.html

“‘ $ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh $ sh https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh “‘

conda を起動します。

“‘ $ conda create -n ml python=3.6 $ conda activate ml “‘

pyserial を pip でインストールします。

“‘ $ pip install -r requirements.txt “‘

kendryte-toolchain のインストール

kendryte の toolchain を、/opt/kendryte-toolchain/にインストールします。

“‘ $ wget http://dl.cdn.sipeed.com/kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.xz $ sudo tar -Jxvf kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.xz -C /opt $ ls /opt/kendryte-toolchain/bin “‘ https://github.com/kendryte/k
gnu-toolchain/releases

MaixPy の Firmware のライブラリを減らす

MaixPy の Firmware のライブラリを減らすことでシステムヒープメモリの容量をあけることができます。

“‘ $ cd MaixPy $ cd projects/maixpy_m5stickv/ “‘

“‘ $ python project.py menuconfig “‘

Conponent Configration を選びます。![Screenshot from 2019-12-08 15-04-16.png](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/294a01c3-a7bc-baa5-d698-b49fa8ff5023.png)

MicroPython Configration を選びます。![Screenshot from 2019-12-08 15-04-23.png](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/6c643b76-5d70-56a4-062b-bf869ef6f981.png)

Module Configration を選びます。![Screenshot from 2019-12-08 15-04-33.png](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/91e588ed-9a95-a444-d3d6-279a0102047d.png)

使わないライブラリを OFF にします。![Screenshot from 2019-12-08 15-05-12.png](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/b49d7c61-6324-3acf-18b4-bf0b81943686.png)

今回は、MaixPy IDE Support を残して他を OFF にしました。![Screenshot from 2019-12-08 15-05-54.png](https://qiita-image-store.s3.ap-northeast-1.amazonaws.com/0/73020/ce583596-6dcf-7084-1bb4-e3b914ae62e3.png)

MaixPy をビルドします。maixpy.bin というファイルが生成されれば完了です。

“‘ python project.py build “‘

生成された MaixPy バイナリと学習モデルを Kfash で書き込んでみましょう。今度こそ、動かすことができました。![02.jpg](https://qiita-image-

store.s3.ap-northeast-1.amazonaws.com/0/73020/a857ceb6-f600-e1a5-4e61-7db19800f4cb.jpeg)

#参考サイト

SIPEED BLOG http://blog.sipeed.com/p/1338.html

SIPEED maixhub https://www.maixhub.com/

お顔の画像はぱくたそ様を使いました。https://www.pakutaso.com/

第 7 章

モーション認識する腕時計

I made a watch to your movement and cheer. My name is Cheering watch!! Imaging acceleration data The acceleration of M5StickV is imaged.Acceleration is entered in 8x8 RGB with x = R, y = G, z = B.This is necessary because KPU only supports images.

The data output from the MPU needs to be converted because the minus is a complement. accel = i2c.readfrom_mem(MPU6886_ADDRESS, MPU6886_ACCEL_XOUT_H, 6) accel_x = (accel[0]<<8|accel[1]) accel_y = (accel[2]<<8|accel[3]) accel_z = (accel[4]<<8|accel[5])

if accel_x>32768: accel_x=accel_x-65536 if accel_y>32768: accel_y=accel_y-65536 if accel_z>32768: accel_z=accel_z-65536

The dynamic motion of acceleration appears as moire.After proceeding 8 dots in the X direction, proceed in the Y dot direction.Interference fringes occur due to the number of vertical and horizontal pixels in the image, the sampling rate of the sensor, and the periodicity of hand movement. Save image data to SD card.

https://github.com/anoken/CheeringWatch_M5StickC-V/blob/master/m5stickv_maixp

Install TensorFlow / Keras for Ubuntu This session use Ubuntu18.04 or Windows Subsystem for Linux. Install Miniconda and create a Python environment. The Miniconda installer is downloaded from the Miniconda website. Miniconda : https://docs.conda.io/en/latest/miniconda.html wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh sh https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

Install Python, TensorFlow, Keras, etc. on Miniconda. conda create -n ml python=3.6 tensorflow=1.14 keras pillow \ numpy pydot graphviz Activate conda. conda activate ml Install kendryte nncase nncase converts learning

data created with Keras or TensorFlow into KPU learning data kmodels. nncase github: https://github.com/kendryte/nncase/ mkdir ./ncc cd ./ncc wget https://github.com/kendryte/nncase/releases/download/v0.1.0-rc5/ncc-linux-x86_64.tar.xz tar -Jxf ncc-linux-x86_64.tar.xz Keras learning A model of acceleration is created by deep learning. Acceleration images are classified into M5StickV SD cards for each activity. Learn with Keras from classification and images at CNN. The learning results are saved with Kmodel.

Run the PYTHON program on Ubuntu as follows command. python keras_motion_larning_191030.py https://github.com/anoken/CheeringWatch_M5StickC-V/blob/master/keras/keras_motion_larning_191030.py "my_model.kmodel"file named A is generated. Kflash GUI Write m5stickv firmware and kmodel with tools.

Draw Face and Speak A friendly face is displayed to help you.This face rotates like a gimbal to the tilt. In addition, audio is output according to the motion.

https://github.com/anoken/CheeringWatch_M5StickC-V/blob/master/m5stickv_maixp

Ambient connect m5stickv cannot be connected to the Internet. So connect it to M5StickC and connect to the Internet. Send motion data to an ambient cloud service in Japan.

The m5stickc software creates programming with arduino.Receive from URT and send to AMBIENT.It also gets the current time from the network and displays the time.

https://github.com/anoken/CheeringWatch_M5StickC-V/blob/master/m5stickc/m5stic

参考文献

第 8 章

インターフォンを監視する AI カメラ「見守りアラート」を作ってみた

お外に外出しているときに、おうちに誰かきているんじゃないか不安になった！そんな経験はありませんか⁉

見守りアラートは外出時のインターフォンを監視し、誰かが来たときことを機械学習で検出して LINE に写真と共に通知を行います。これで安心して、お外に外出できるのです。

カメラの前に不審な動きがある場合、カメラは自動的に写真を撮り、通知します。見守りアラートは非常にコンパクトです低コストです。

MiMaMori Alert is two configration. ・M5StickV + M5StickC ・M5UnitV + M5StickC

M5StickV/ユニット V と M5StickC はグローブで接続されています。通信は UART によるものであり、通信は独自のプロトコルを使用して行われる。10 ビット単位でデータを送信します。アルゴリズム

The algorithm is as follows. ① First, feature vectors are calculated using Mobilnet's neural network. Mobilnet has created a V1 weight of 0.5 with Keras and NNcase. NNCASE is a Kendryte tool. task = kpu.load(0x200000) fmap = kpu.forward(task, img) new_data = np.array(fmap[:]) ② The feature vector obtained by the neural network is filtered in the time series direction, and a weighted average is obtained. Vector operation uses Numpy #Feature Vector Update def update(capture,new_data,weight): new_data= new_data*weight+capture*(1.0-weight) return new_data

new_data=update(capture,new_data,cap_weight) ③ The distance between

the average vector and the feature vector at the current time is calculated. This is equivalent to finding motion. The distance is large if moving, and small if static. #Feature Vector Compare def get_dis(new_data,master_data): dist = np.sum((new_data-master_data)*(new_data-master_data)) return dist

dist=get_dis(new_data,master_data) ④ Detects rising edge of waveform and sends data by UART. The rising edge is compared with the previous data in time series

if dist > dist_thresh: if dist_old <= dist_thresh: img_buf = img.copy() img_buf.compress(quality=70) img_size1 = (img_buf.size()& 0xFF0000)>>16 img_size2 = (img_buf.size()& 0x00FF00)>>8 img_size3 = (img_buf.size()& 0x0000FF)>>0 data_packet = bytearray([0xFF,0xF1,0xF2,0xA1,img_siz uart_Port.write(data_packet) uart_Port.write(img_buf) time.sleep(1.0) print("image send,data_packet") ⑤ M5StickC sends images and messages to LINE by notification from UART. M5StickC is developed with ArduinoIDE.

More Application

This application can detect dynamic ones. Installation is also very easy. Only just keep in front of things. Various applications are possible. Here is defence technique for Japanese traditional food "nameko". "MiMaMori Alert" is set behind the nameko. If a nameko is taken, notify your LINE, soon. Demo instruction This source code requires the MixPy option configration Numpy. Custom binaries is stored within Github. And, Neural networks require a MobileNetV1 with a weight of 0.5. This Kmodel file is also stored in Github. Write to M5StickV / UnitV with Kflash GUI Tools.

References PuddingAlert-V https://m5stack.hackster.io/anoken2017/puddingalert-v-34c560 Cheering Watch of M5StickC&V https://m5stack.hackster.io/anoken2017/cheering watch-of-m5stickc-v-34f0cc

著者紹介

aNo 研（@anoken2017）/

aNo 研は 3 人のエンジニアが集って結成されたものづくり集団です。aNo 研の目標は、"あの"心躍る"もの"と出会ったときの感動を、あなたに届けること。それを目指して、私たちは活動をしています。

経歴

・Gugen2017 Vstone 賞　受賞
・池袋アートギャザリング (IAG)2018 入選
・異能 Innovation2018 ジェネレーションアワード部門 ノミネート
・M5 Creativity 2018 The Best Creative Award 受賞

nnn (@nnn112358)

aNo 研の M5Stack 担当です。

MAiX MAniaX

2020 年 3 月 1 日　初版 v1.1.0

著　者　aNo 研