STM32H7 マイクロコントローラ用人工知能（AI）とコンピュータ・ビジョン機能パック

概要

FP-AI-VISION1 は、コンピュータ・ビジョン・タスクに関して効率的に畳み込みニューラルネットワーク（CNN）を実行する STM32H7 シリーズマイクロコントローラの能力をデモンストレーションする機能パック（FP）です。FP-AI-VISION1 には、STM32H7 マイクロコントローラに CNN ベースのコンピュータ・ビジョン・アプリケーションを構築するために必要なすべてのものが含まれています。

また、FP-AI-VISION1 では、アプリケーション関連データのためのメモリ割り当て設定もいくつかデモンストレーションします。いずれの設定でも、アプリケーションに必要なデータ量に関する具体的な要件に対応可能です。したがって、FP-AI-VISION1 には、さまざまなタイプのデータをオンチップ・メモリと外部メモリの両方に効率的に配置する方法を説明する例が提供されています。これらの例から、ユーザは、要件に最適なメモリ割り当てを簡単に理解できます。

このユーザ・マニュアルには、FP-AI-VISION1 機能パックの内容と、STM32H7 マイクロコントローラに CNN ベースのコンピュータ・ビジョン・アプリケーションを構築するために実行するさまざまな手順の詳細が記載されています。

1 一般情報

FP-AI-VISION1 機能パックは、Arm® Cortex®-M7 プロセッサに基づく STM32H7 マイクロコントローラで動作します。

注 Arm は、米国内およびその他の地域にある Arm Limited（またはその子会社）の登録商標です。

arm

1.1 FP-AI-VISION1 機能パック機能の概要

- B-CAMS-OMV カメラ・モジュール・バンドル（推奨）または STM32F4DIS-CAM カメラドーター・ボード（レガシのみ）に接続された STM32H747I-DISCO ボードで動作します。
- CNN に基づく画像分類のアプリケーション例が 3 つ含まれています。
  - カラー（RGB 24bit）フレーム画像で動作する食品分類アプリケーション
  - カラー（RGB 24bit）フレーム画像で動作する人検出アプリケーション
  - グレースケール（8bit）フレーム画像で動作する人検出アプリケーション
- 物体検出 CNN モデルに基づく人数カウント・アプリケーション（シーン内人数カウント）が 1 つ含まれています。
- カメラ・キャプチャ、フレーム画像前処理、推論実行、出力後処理のための完全なアプリケーション・フレームワークが含まれています。
- 100 を超える処理演算がサポートされている機能豊富な画像処理ライブラリが付属されています。
- 浮動小数点と 8bit 量子化の C モデルの両方の実装例が含まれています。
- アプリケーション要件を満たすために、データ・メモリ配置に対するいくつかの設定がサポートされています。
- 組込みアプリケーションをテスト、デバッグ、検証するためのテストと検証のファームウェアが含まれています。
- データセット収集を可能とするキャプチャ・ファームウェアが含まれています。
- 外部 microSD™カード上のファイル処理（FatFS 上）に対するサポートが含まれています。
- 画像およびビデオのデータセットの作成や、ホスト上でのライブ・テストの実行に使用できる USB ウェブカメラ・アプリケーションが付属されています。

1.2 ソフトウェア・アーキテクチャ

FP-AI-VISION1 機能パックの使用方法の最上位アーキテクチャを図 1 に示します。

図 1. FP-AI-VISION1 アーキテクチャ

1.3 用語と定義

この資料の文脈理解を深めるために関連する略称の定義を表 1 に示します。

表 1. 略称のリスト

項目（略称）	定義
API	アプリケーション・プログラミング・インタフェース
BSP	ボード・サポート・パッケージ
CNN	畳み込みニューラルネットワーク
DMA	ダイレクト・メモリ・アクセス

項目（略称）	定義
FAT	ファイル･アロケーション･テーブル
FatFS	軽量な汎用 FAT ファイル･システム
FP	機能パック
FPS	1 秒あたりフレーム数
HAL	ハードウェア抽象化レイヤ
LCD	液晶ディスプレイ
MCU	マイクロコントローラユニット
microSD™	マイクロ･セキュア･デジタル
MIPS	1 秒あたり命令数（百万単位）
NN	ニューラルネットワーク
RAM	ランダムアクセス･メモリ
QVGA	1/4 VGA
SRAM	スタティック･ランダムアクセス･メモリ
USB	ユニバーサル･シリアル･バス
VGA	ビデオ･グラフィック･アレイ解像度

1.4 入手可能なドキュメントおよびリファレンスの概要

FP-AI-VISION1 を使用するための補足リファレンスを表 2 に示します。

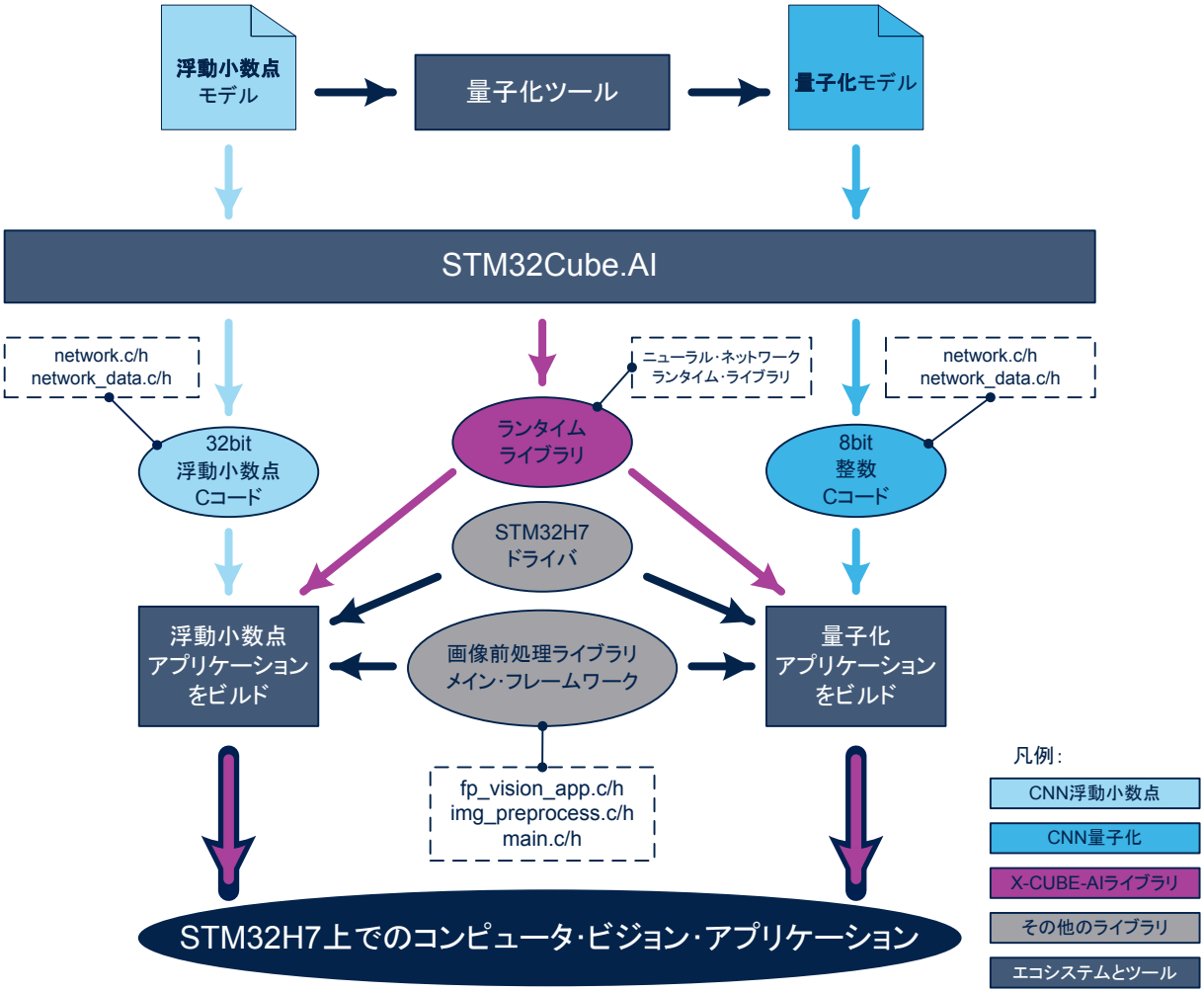
表 2. 参照

ID	説明
[1]	ユーザ･マニュアル： Getting started with X-CUBE-AI Expansion Package for Artificial Intelligence (AI) (UM2526)
[2]	リファレンス･マニュアル： STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs (RM0399)
[3]	MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications: https://arxiv.org/pdf/1704.04861.pdf
[4]	Food-101 データセット： https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/
[5]	STM32 製品プログラミング用 STM32CubeProgrammer ソフトウェア： STM32CubeProg
[6]	Keras - Python 深層学習ライブラリ： https://keras.io/
[7]	STM32Cube 初期化コードジェネレータ： STM32CubeMX

2 STM32H7 上の CNN ベースのコンピュータ·ビジョン·アプリケーションの構築

STM32H7 マイクロコントローラで動作する CNN ベースのコンピュータ·ビジョン·アプリケーションを取得するためのさまざまなステップを図 2 に示します。

図 2. CNN ベースのコンピュータ·ビジョン·アプリケーションのビルドフロー



STM32H7 にコンピュータ·ビジョン·アプリケーションを構築するには、(Keras などのフレームワークで設計と訓練が行われた)浮動小数点 CNN モデルから始めて、(STM32Cube.AI ツール、[1]を使用して)最適化された C コードを生成し、(FP-AI-VISION1 の一部として提供される)コンピュータ·ビジョン·フレームワークに統合します。

C コードの生成には、次の 2 つのオプションのいずれかを選択できます。

- 浮動小数点の CNN モデルから浮動小数点 C コードを直接生成
- あるいは、浮動小数点の CNN モデルを量子化して 8bit モデルを取得した後に、対応する量子化 C コードを生成

ほとんどの CNN モデルにおいては、2 番目のオプションを使用するとメモリ・フットプリント（Flash および RAM）と推論時間を削減できます。最終出力精度への影響は、CNN モデルと量子化プロセス（主としてテスト・データセットと量子化アルゴリズム）に依存します。

FP-AI-VISION1 機能パックの一部として、以下の資料を含む、画像分類アプリケーション例が 3 つ提供されています。

- 食品分類アプリケーション（1つ）:
  - 浮動小数点 Keras モデル（.h5 ファイル）
  - （X-CUBE-AI）量子化器を使用して取得した 8bit 量子化モデル（.h5 ファイル + .json ファイル）
  - 浮動小数点と 8bit 量子化フォーマットの両方で生成された C コード
  - STM32Cube.AI（X-CUBE-AI）によって生成された C コードに基づくコンピュータ・ビジョン・アプリケーション統合例
- 人検出アプリケーション（2 つ）:
  - TFLiteConverter ツール(1)を使用して取得した 8bit 量子化モデル（.tflite ファイル）
  - 8bit 量子化フォーマットで生成された C コード
  - STM32Cube.AI（X-CUBE-AI）によって生成された C コードに基づくコンピュータ・ビジョン・アプリケーション統合例

1. TensorFlow は Google 社の商標です。

2.1 生成コードの統合

浮動小数点モデルまたは量子化モデルから、STM32Cube.AI ツール（X-CUBE-AI）を使用して対応する最適化 C コードを生成する必要があります。

ユーザ自身の .ioc ファイルとともに GUI バージョンの STM32Cube.AI（X-CUBE-AI）を使用する場合は、出力ディレクトリに次のファイルのセットが生成されます。

- Src\network.c と Inc\network.h：CNN トポロジの説明を格納
- Src\network_data.c と Inc\network_data.c：CNN の重みとバイアスを格納

注 ネットワークの場合は、network というデフォルトの名前を保持する必要があります。それ以外の場合は、ファイル ai_interface.c と ai_interface.h に含まれているすべての関数とマクロの名前を変更する必要があります。ai_interface.c ファイルと ai_interface.h ファイルの目的は、NNAPI に抽象化インタフェースを提供することです。

その点から、上記で生成された .c ファイルと .h ファイルは、それぞれ以下のディレクトリにコピーして置換する必要があります。

- \Projects\STM32H747I-DISCO\Applications\<app_name>\CM7\Src
  <app_name>は、次のいずれか
  - FoodReco_MobileNetDerivative\Float_Model
  - FoodReco_MobileNetDerivative\Quantized_Model
  - PersonDetection\Google_Model
  - PersonDetection\MobileNetv2_Model
  - PeopleCounting
- \Projects\STM32H747I-DISCO\Applications\<app_name>\CM7\Inc
  <app_name>は、次のいずれか
  - FoodReco_MobileNetDerivative\Float_Model
  - FoodReco_MobileNetDerivative\Quantized_Model
  - PersonDetection\Google_Model
  - PersonDetection\MobileNetv2_Model
  - PeopleCounting

別の方法は、CLI（コマンドライン・インタフェース）バージョンの STM32Cube.AI（X-CUBE-AI）を使用して、コマンド・ラインで指定した出力ディレクトリに含まれる Src ディレクトリと Inc ディレクトリに、生成されたファイルを直接コピーするやり方です。この方法では、手動でのコピー / 貼り付け操作は必要ありません。

アプリケーション・パラメータはファイル fp_vision_app.c と fp_vision_app.h の中で設定しますので、ユーザのニーズに簡単に合わせることができます。

ファイル fp_vision_app.c:

- output_labels[]文字列テーブル(それぞれの文字列がニューラルネットワークモデルの出力クラスの 1 つに対応しています)が、新規アプリケーションに対するアダプテーションが絶対に必要な唯一の箇所です。
- App_Context_Init()関数が、アプリケーションのさまざまなソフトウェア・コンポーネントを初期化する役割を持っています。以下の設定については、変更が必要な場合があります。
  - カメラの向きの調整
  - オンボード検証モードにおいて microSD™カードから入力画像を読み出すパスの調整
  - トレーニング・フェーズ中に使用される NN 入力データ範囲の調整
    これを行うには、初期化中に nn_input_norm_scale 変数と nn_input_norm_zp 変数の両方の値を更新します。nn_input_norm_scale 変数と nn_input_norm_zp 変数は、ピクセル・フォーマット・アダプテーションステージに影響します(セクションセクション 3.2.5.1 の図 7 参照)。
  - NN 入力データのピクセル・カラー・フォーマットの調整
    これを行うには、Pfc_Dst_Img.format 変数の初期化値を更新します。Pfc_Dst_Img.format 変数は、ピクセル・カラー・フォーマット変換ステージに影響します(PFC、セクションセクション 3.2.5.1 の図 7 参照)。

3 パッケージの内容

3.1 CNN モデル

FP-AI-VISION1 機能パックには、以下の 2 つの CNN ベース画像分類アプリケーションのデモンストレーションが含まれています。

- 18 種類の飲食物を認識する食品分類アプリケーション
- 画像内に人が存在するか否かを識別する人検出アプリケーション

FP-AI-VISION1 機能パックには、オブジェクト検出ニューラルネットワークモデルに依存した人数カウント・アプリケーションのデモンストレーションも含まれています。

3.1.1 食品分類アプリケーション

食品分類 CNN は、MobileNet モデルの派生です（[3]参照）。

MobileNet は、モバイルと組込みのビジョン・アプリケーションに適した効率的なモデル・アーキテクチャ[3]です。このモデル・アーキテクチャは、Google®から提案されました。

MobileNet モデル・アーキテクチャには、遅延と精度の効率的なトレードオフを行うシンプルなグローバル・ハイパーパラメータが 2 つ含まれています。基本的に、モデル・ビルダは、これらのハイパーパラメータを使用して、問題の制約条件に基づいてアプリケーションに合ったサイズのモデルを決定できます。

この FP で使用する食品分類モデルは、STM32H747 ターゲット制約条件を考慮して、精度、計算コスト、メモリ・フットプリントの間で最適なトレードオフとなるようにこれらのハイパーパラメータを調整して構築されています。

食品分類 CNN モデルは、18 種類の飲食物のカスタム・データベースでトレーニングされています。

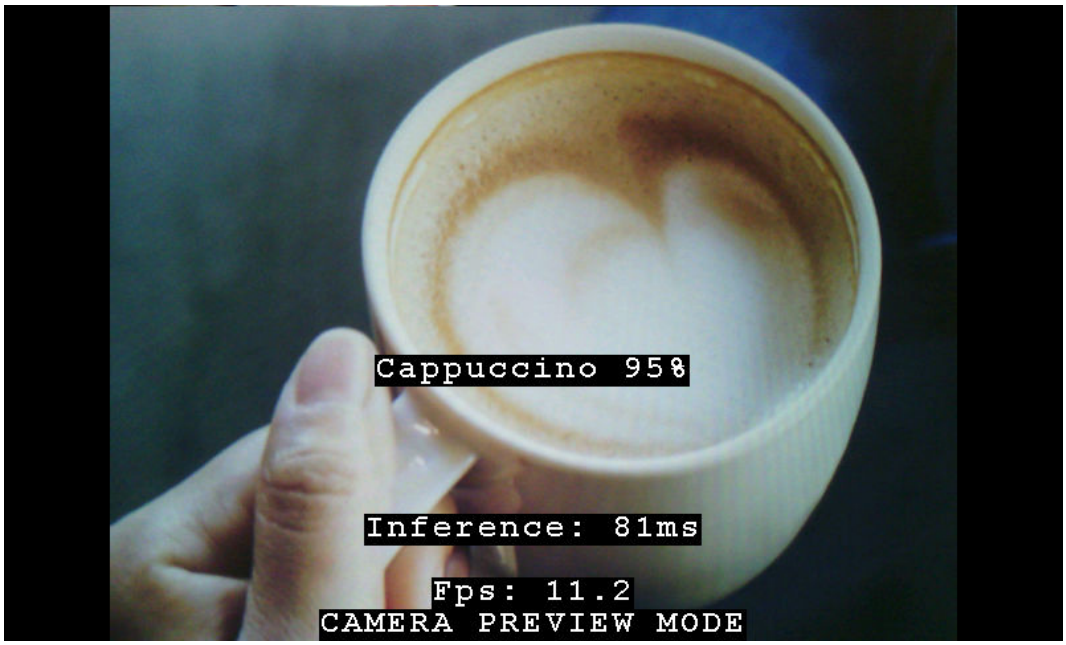
- アップル・パイ
- ビール
- シーザ・サラダ
- カプチーノ
- チーズケーキ
- チキン・ウィング
- チョコレート・ケーキ
- Coke™
- カップケーキ
- ドーナツ
- フライド・ポテト
- ハンバーガー
- ホット・ドッグ
- ラザニア
- ピザ
- リゾット
- スパゲッティ・ボロネーゼ
- ステーキ

食品分類 CNN は、入力としてサイズが 224 x 224 ピクセルのカラー画像を想定しており、各ピクセルは RGB888 の 3 バイトにコード化されます。

FP-AI-VISION1 機能パック には、食品分類アプリケーションに基づくサンプルが 2 つ含まれています。1 つは浮動小数点バージョンの生成コードを実装する例、もう 1 つは量子化バージョンの生成コードを実装する例です。

図 3 に、食品分類アプリケーションを示します。

図 3. 食品分類アプリケーション



3.1.2 人検出アプリケーション

このパッケージには、以下の 2 つの人検出アプリケーションが含まれています。

- 解像度が 96 x 96 ピクセルのグレースケール画像(ピクセルあたり 8bit)を処理する低複雑度 CNN モデル(いわゆる `Google_Model`)ベースのアプリケーション。モデルは、storage.googleapis.com からダウンロードします。
- 解像度が 128 x 128 ピクセルのカラー画像(ピクセルあたり 24bit)を処理する高複雑度 CNN モデル(いわゆる `MobileNetv2_Model`)ベースのアプリケーション。

人検出モデルには、`Person` と `Not Person` の 2 つの出力クラスが含まれています。

FP-AI-VISION1 機能パック では、8 ビット量子化モデルのデモンストレーションを行ないます。

図 4 に、人検出アプリケーションを示します。

図 4. 人検出アプリケーション

3.1.3 人数カウント·アプリケーション

人数カウント·アプリケーションは、キャプチャした画像を処理し、画像内の人の位置に対応したバウンディング·ボックスを出力する、STM32H7 マイクロコントローラに最適化されたオブジェクト検出モデルに依存しています。

人数カウント·アプリケーションは、画像内の人の認識、カウント、追跡を行うことができます。

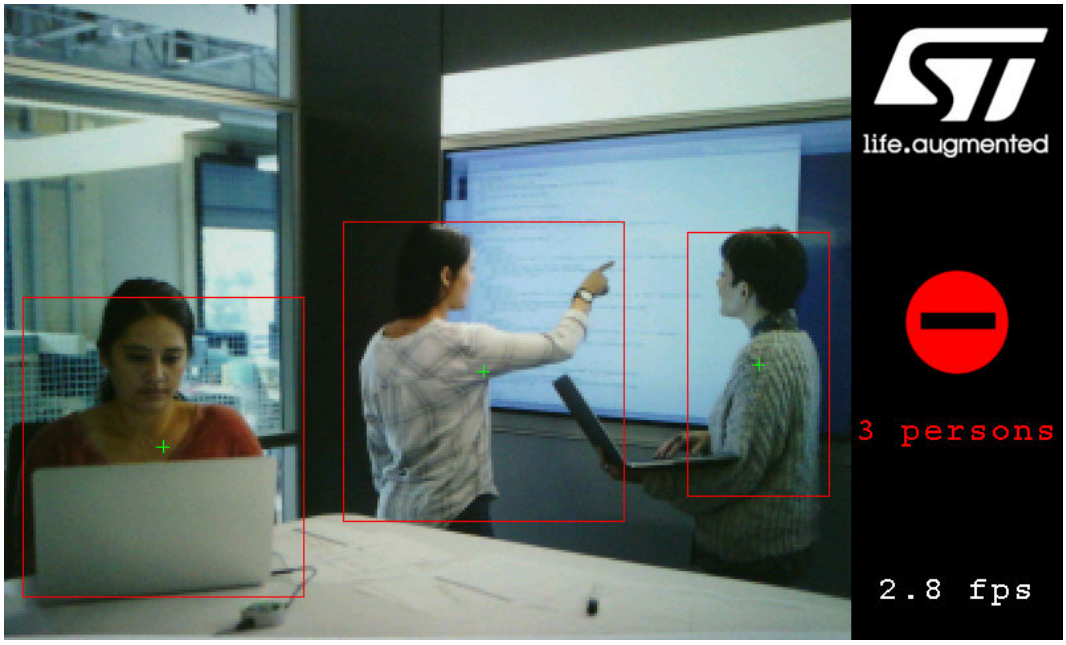
使用されている物体検出モデルは、STMicroelectronics のモデルである Zoo の一部であり、8bit で量子化されて、RGB888 フォーマットの 240 x 240 解像度の入力画像が想定されている YOLO モデルの派生です。

人数カウント·アプリケーションには、完全な内部メモリ割り当て方式が実装されています。

使用されているオブジェクト検出ニューラルネットワークモデルは、バイナリ·ライブラリとして提供されます。詳細については、STMicroelectronics にお問い合わせください。

図 5 に、人数カウント·アプリケーションを示します。

図 5. 人数カウント·アプリケーション

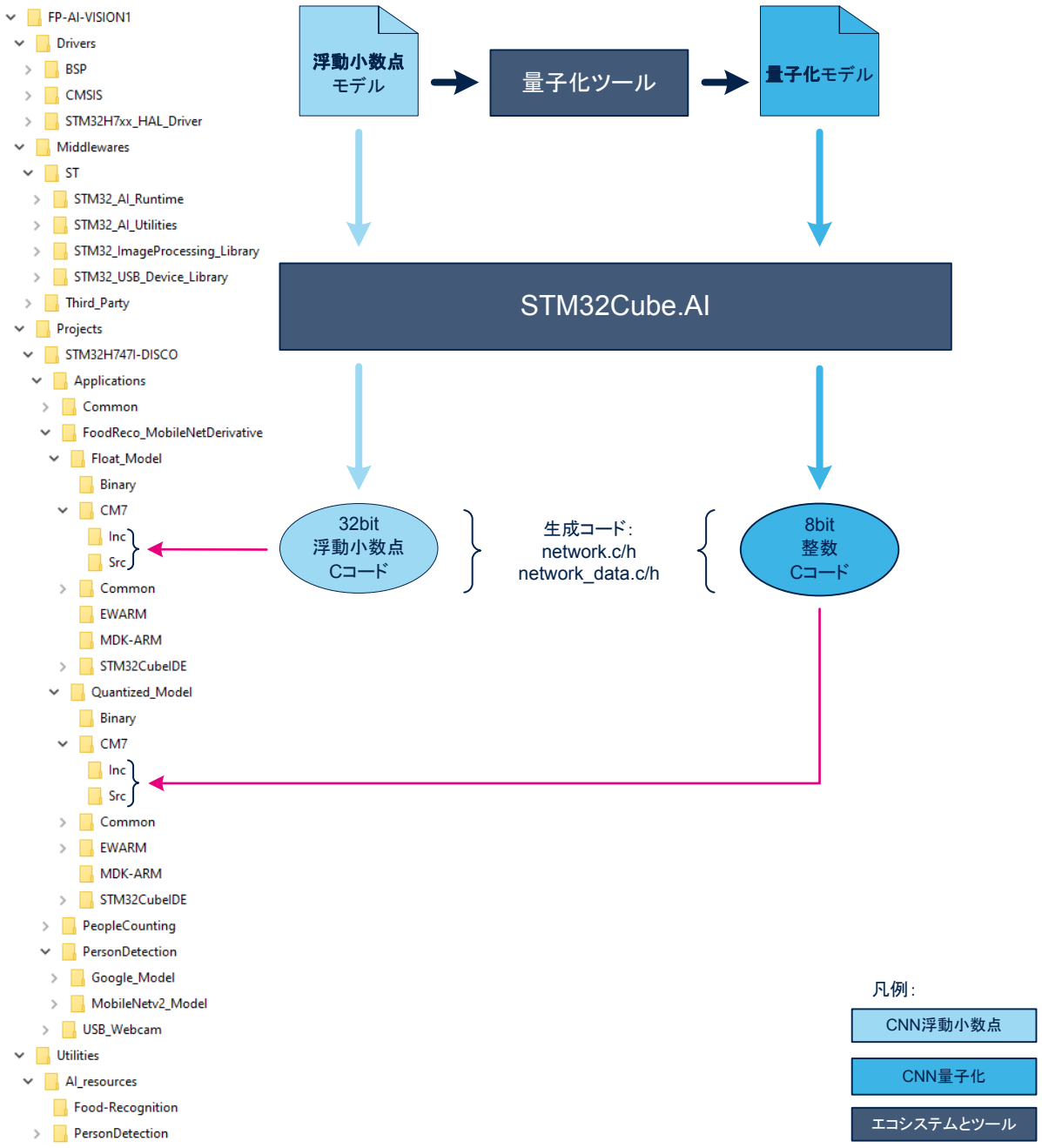


3.2 ソフトウェア

3.2.1 フォルダ構成

FP-AI-VISION1 機能パックのフォルダ構成を図 6 に示します。

図 6. FP-AI-VISION1 フォルダ·ツリー



ドライバ

すべての BSP と STM32H7HAL ソース·コードが含まれています。

ミドルウェア

以下の 5 つのサブフォルダが含まれています。

- `ST/STM32_AI_Runtime`
  `lib` フォルダには、IAR Systems 社の IAR Embedded Workbench®（EWARM）、Keil®社の MDK-ARM、ST マイクロエレクトロニクスの STM32CubeIDE のそれぞれの IDE に対して、STM32Cube.AI（X-CUBE-AI）によって生成されたニューラルネットワークランタイム・ライブラリが格納されています。これらのライブラリは、新規のニューラルネットワークを変換するときに置き換える必要はありません。.
  `Inc` フォルダには、ランタイム・ライブラリに必要なインクルード・ファイルが格納されています。
  これら 2 つのフォルダは、新バージョンの X-CUBE-AI コード・ジェネレータを使用しない限り、新規のニューラルネットワークを変換するときに置き換える必要はありません。
- `ST/STM32_AI_Utilities`
  最適化されたルーチンが格納されています。
- `ST/STM32_ImageProcessing_Library`
  画像処理関数のライブラリが格納されています。これらの関数は、カメラでキャプチャされた入力フレーム画像の前処理に使用されます。この前処理の目的は、推論中のニューラルネットワークへの入力として適切なデータ（サイズ、フォーマットなど）を生成することです。このライブラリの詳細については、セクション 3.2.2 を参照してください。
- `ST/STM32_USB_Device_Library`
  STM32 用 USB デバイス・ライブラリが格納されています。
- `Third_Party/FatFS`
  FAT ファイル・システムに対するサポートを提供する 3rd パーティ製ミドルウェアです。

`Project/STM32H747I-DISCO/Applications`

FP-AI-VISION1FP の中で提供されているアプリケーションのプロジェクトとソース・コードが格納されています。これらのアプリケーションは、Cortex®-M7 プロセッサと Cortex®-M4 プロセッサをベースとするデュアルコア・マイクロコントローラである STM32H747 で動作します（[2]参照）。アプリケーション・コードは、Cortex®-M7 コアでのみ動作します。

`Project/STM32H747I-DISCO/Applications/Common`

このフォルダには、すべてのアプリケーションに共通のソース・コードが格納されています。

- `ai_interface.c` と `ai_interface.h`
  NN API の抽象化を行います。
- `fp_vision_ai.c` と `fp_vision_ai.h`
  NN 入力データ表現の調整、NN 出力データの後処理、NN の初期化、NN の推論の実行に必要なユーティリティを提供します。新規のニューラルネットワークモデルを統合する場合は、アプリケーション・パラメータに合わせてユーザがこれらのファイルを調整する必要があります。
- `fp_vision_camera.c` と `fp_vision_camera.h`
  カメラ・モジュールの設定と管理のための機能を提供します。
- `fp_vision_display.c` と `fp_vision_display.h`
  LCD ディスプレイの設定と管理を行うための機能を提供します。
- `fp_vision_preproc.c` と `fp_vision_preproc.h`
  画像前処理ライブラリに抽象化レイヤを提供します（`Middlewares/ST/STM32_Image` にあります）。
- `fp_vision_test.c` と `fp_vision_test.h`
  アプリケーションのテスト、デバッグ、検証のための機能セットを提供します。
- `fp_vision_utils.c` と `fp_vision_utils.h`
  各種ユーティリティを提供します。

`Project/STM32H747I-DISCO/Applications/FoodReco_MobileNetDerivative`

このフォルダには、食品分類アプリケーションに関連するすべてのソース・コードが格納されています。サブフォルダが 2 つあり、アプリケーション例ごとに 1 つのサブフォルダとなっています。

- 浮動小数点 C モデルの統合デモンストレーション（32bit 浮動小数点 C コード）
- 量子化 C モデルの統合デモンストレーション（8bit 整数 C コード）

それぞれのサブフォルダは、次のような構成となっています。

- 2 進数
  アプリケーションのバイナリが格納されています。
  - STM32H747I-DISCO_u_v_w_x_y_z.bin
    バイナリは、Float_Model/CM7 フォルダと Quantized_Model/CM7 フォルダに格納されているソース・ファイルから生成されています。
    - u はアプリケーション名に相当します。食品分類アプリケーションの場合は、次の値となります。
      - Food
    - v はモデル・タイプに相当します。食品分類アプリケーションの場合は、次の値が可能です。
      - Std（標準）
      - Optimized（最適化済み）

      v が Opt である場合には、特定バージョンの食品分類 CNN モデルから生成されていて、FP-AI-VISION1 機能パックの一部としてリリースされたのではないソースから、そのバイナリが生成されたことを示します。この特定バージョンのモデルは、精度と、メモリ・フットプリントや MIPS のような組み込み制約事項との間のトレードオフを改善するために、さらに最適化されています。この特定バージョンの詳細については、STMicroelectronics にお問い合わせください。
    - w はモデル・タイプのデータ表現に相当します。食品分類アプリケーションの場合は、次の値が可能です。
      - Float（浮動小数点 32bit）
      - Quant8（量子化 8bit）
    - x は揮発性データ・メモリ割り当ての設定に相当します。食品分類アプリケーションの場合は、次の値が可能です。
      - Ext（外部 SDRAM）
      - Split（内部 SRAM と外部 SDRAM で分割）
      - IntMem（メモリ最適化された内部 SRAM）
      - IntFps（FPS 最適化された内部 SRAM）
    - y は不揮発性データのメモリ割り当て設定に相当します。食品分類アプリケーションの場合は、次の値が可能です。
      - IntFlash（内部 Flash メモリ）
      - QspiFlash（外部 Q-SPI Flash メモリ）
      - ExtSdram（外部 SDRAM）
    - z は FP-AI-VISION1 リリースのバージョン番号に相当します。これは Vabc のように表され、ここで a と b と c は、メジャー・バージョン、マイナー・バージョン、パッチ・バージョンの番号をそれぞれ表します。このユーザ・マニュアルに対応した食品分類アプリケーションの場合は、次の値となります。
      - V300
- CM7
  Cortex®-M7 コア上で実行される食品分類アプリケーション例専用のソース・コードが格納されています。ファイルには次の 2 種類があります。
  - STM32Cube.AI ツールで生成されるファイル（X-CUBE-AI）:
    - network.c と network.h:CNN トポロジの説明が格納されています。
    - network_data.c と network_data.h:CNN の重みとバイアスが格納されています。
  - アプリケーションが格納されているファイル:
    - main.c と main.h
    - fp_vision_app.c と fp_vision_app.h
      アプリケーション固有の設定に使用します。
    - stm32h7xx_it.c と stm32h7xx_it.h
      割込みハンドラが実装されています。
- CM4
  食品分類アプリケーションのすべてのコードは Cortex®-M7 コア上で実行されるため、このフォルダは空です。
- 共通
  Cortex®-M7 コアと Cortex®-M4 コアに共通するソース・コードが格納されています。
- EWARM
  アプリケーション例の IAR System IAR Embedded Workbench®ワークスペース・ファイルとプロジェクト・ファイルが格納されています。両方のコアのスタートアップ・ファイルも含まれています。

- MDK-ARM
  アプリケーション例の Keil® MDK-ARM ワークスペース·ファイルとプロジェクト·ファイルが格納されています。両方のコアのスタートアップ·ファイルも含まれています。
- STM32CubeIDE
  アプリケーション例の STM32CubeIDE ワークスペース·ファイルとプロジェクト·ファイルが格納されています。両方のコアのスタートアップ·ファイルも含まれています。

注 EWARM、MDK-ARM、STM32CubeIDE サブフォルダには、それぞれのアプリケーション·プロジェクトが複数の設定を格納していることがあります。それぞれの設定は、以下に対応しています。

- 揮発性メモリ(RAM)における特定のデータ配置
- 不揮発性メモリ(Flash)における重みバイアステーブルの特定の配置

Project/STM32H747I-DISCO/Applications/PeopleCounting

このフォルダには、人数カウント·アプリケーション専用のソース·コードが格納されています。

サブフォルダの構成は、食品分類アプリケーション例に対してすでに説明したサブフォルダの構成と似ています。CM7 フォルダには、lib という名前のもう 1 つのサブフォルダがあり、オブジェクト検出モデルを実装するバイナリ·ライブラリが格納されています。

Binary サブフォルダには、アプリケーションのバイナリが格納されています。バイナリの名前は、STM32H747I-DISCO_u_w_x_y_z.bin という形式になっています。

- u はアプリケーション名に相当します。人数カウント·アプリケーションの場合は、次の値となります。
  - PeopleCounting
- w はモデル·タイプのデータ表現に相当します。人数カウント·アプリケーションの場合は、次の値となります。
  - Quant8(量子化 8bit)
- x は揮発性データのメモリ割り当て設定に相当します。人数カウント·アプリケーションの場合は、次の値となります。
  - IntFps (FPS 最適化された内部 SRAM)
- y は不揮発性データのメモリ割り当て設定に相当します。人数カウント·アプリケーションの場合は、次の値となります。
  - IntFlash (内部 Flash メモリ)
- z は FP-AI-VISION1 リリースのバージョン番号に相当します。これは Vabc のように表され、ここで a と b と c は、メジャー·バージョン、マイナー·バージョン、パッチ·バージョンの番号をそれぞれ表します。このユーザ·マニュアルに対応した人数カウント·アプリケーションの場合は、次の値となります。
  - V300

Project/STM32H747I-DISCO/Applications/PersonDetection

このフォルダには、人検出アプリケーション専用のソース·コードが格納されています。サブフォルダが 2 つあり、アプリケーション例ごとに 1 つのサブフォルダとなっています。

- 低複雑度モデル(いわゆる Google_Model)の統合デモンストレーション
- 中複雑度モデル(いわゆる MobileNetv2_Model)の統合デモンストレーション

サブフォルダの構成は、食品分類アプリケーション例に対してすでに説明したサブフォルダの構成と同じです。

Binary サブフォルダには、アプリケーションのバイナリが格納されています。バイナリの名前は、STM32H747I-DISCO_u_v_w_x_y_z.bin という形式になっています。

- u はアプリケーション名に相当します。人検出アプリケーションの場合は、次の値となります。
  - Person
- v はモデル・タイプに相当します。人検出アプリケーションの場合は、次の値が可能です。
  - Google
  - MobileNetV2
- w はモデル・タイプのデータ表現に相当します。人検出アプリケーションの場合は、次の値となります。
  - Quant8（量子化 8bit）
- x は揮発性データのメモリ割り当て設定に相当します。人検出アプリケーションの場合は、次の値となります。
  - IntFps（FPS 最適化された内部 SRAM）
- y は不揮発性データのメモリ割り当て設定に相当します。人検出アプリケーションの場合は、次の値となります。
  - IntFlash（内部 Flash メモリ）
- z は FP-AI-VISION1 リリースのバージョン番号に相当します。これは Vabc のように表され、ここで a と b と c は、メジャー・バージョン、マイナー・バージョン、パッチ・バージョンの番号をそれぞれ表します。このユーザ・マニュアルに対応した人検出アプリケーションの場合は、次の値となります。
  - V300

Project/STM32H747I-DISCO/Applications/USB_Webcam

このフォルダには、USB ウェブカメラ・アプリケーション専用のソース・コードが格納されています。フォルダの構成は、食品分類アプリケーション例に対してすでに説明したサブフォルダの構成と同じです。

Binary サブフォルダには、アプリケーションのバイナリが格納されています。バイナリの名前は、STM32H747I-DISCO_u_z.bin という形式になっています。

- u はアプリケーション名に相当します。USB ウェブカメラ・アプリケーションの場合は、次の値となります。
  - Webcam
- z は FP-AI-VISION1 リリースのバージョン番号に相当します。これは V.abc のように表され、ここで a と b と c は、メジャー・バージョン、マイナー・バージョン、パッチ・バージョンの番号をそれぞれ表します。このユーザマニュアルに対応した USB ウェブカメラ・アプリケーションの場合は、次の値となります。
  - V300

Utilities/AI_resources/Food-Recognition

このサブフォルダには、次のファイルが含まれます。

- アプリケーション例で使用されている食品分類 CNN のためのオリジナルの学習済みモデル（ファイル FoodReco_MobileNet_Derivative_Float.h5）。このモデルは、以下のいずれかのために使用されます。
  - STM32Cube.AI（X-CUBE-AI）によって浮動小数点 C コードを直接生成する
  - 量子化プロセスによって 8bit 量子化モデルを生成した後に、STM32Cube.AI（X-CUBE-AI）によって整数 C コードを生成する
- 量子化プロセスに必要なファイル（参照セクション 3.2.3 量子化プロセス）:
  - config_file_foodreco_nn.json:量子化演算の設定パラメータが格納されているファイル
  - test_set_generation_foodreco_nn.py:量子化プロセスで使用されるテスト・ベクタの準備に使用される関数が格納されているファイル
- 量子化ツールで生成された量子化モデル（ファイル FoodReco_MobileNet_Derivative_Quantized.json と FoodReco_MobileNet_Derivative_Quantized.h5）
- 再トレーニング・スクリプト（セクション 3.2.4 トレーニング・スクリプト参照）:FoodDetection.py と Jupyter™ ノートブック（FoodDetection.ipynb）
- オンボードで検証を行なるファームウェアが想定するフォーマットに画像データセットを変換するスクリプト（セクション 3.2.12 組み込みの検証、キャプチャ、テストのオンボード検証モード参照）

Utilities/AI_resources/PersonDetection

このサブフォルダには、次のファイルが含まれます。

- MobileNetv2_Model/README.md:TensorFlow™を使用して事前学習済みのネットワークから、新規の人検出画像分類器を再トレーニングする方法が説明されています。

- `MobileNetv2_Model/create_dataset.py`: Python™ `README.md` ファイルに記載されている、過去にダウンロードした COCO データセットから`***Person20***`データセットを作成するスクリプト。
- `MobileNetv2_Model/train.py`: Python™ 事前学習済みの MobileNetV2 ヘッドから画像分類器モデルを作成するスクリプト。
- `MobileNetv2_Model/quantize.py`: Python™ TensorFlow™の TFLiteConverter ツールを使用して Keras モデル上でポストトレーニング量子化を行うスクリプト量子化演算を実行するには、サンプル·イメージが必要となります。

3.2.2 STM32 画像処理ライブラリ

`STM32_ImageProcessing_Library` は、100 を上回る画像処理演算がサポートされている高機能な画像処理ライブラリです。一般的な演算（画像再スケーリング、画像色変換など）のみならず、高度な演算（ブロブトラッキング、顔検出など）も含まれています。

このライブラリは、`.bmp` と `.jpeg` などの複数のファイル·フォーマットに対する読出し操作と書込み操作に対応しています。詳細については、`Middlewares/ST/STM32_ImageProcessing_Library` フォルダにあるマニュアルを参照してください。

3.2.3 量子化プロセス

量子化プロセスは、パラメータ（重みとバイアス）の量子化と、8bit 整数で表現されたパラメータと活性化を有する量子化モデルを得るための NN の活性化から構成されます。

重み、バイアス、活性化が 32bit 浮動小数点ではなく 8bit で行われることから、モデルを量子化するとメモリ・フットプリントが削減されます。また、Cortex®-M7 コアの最適化された DSP ユニットにより、推論の実行時間が短縮されます。

STM32Cube.AI（X-CUBE-AI）ツールでは、次の複数の量子化方式がサポートされています。

- 固定小数点 Qm,n（非推奨機能）
- 整数演算（符号付きと符号なし）

X-CUBE-AI は、8bit 整数フォーマットのさまざまなタイプの量子化モデルをインポートできます。

- テンソル・フォーマット設定ファイルに関連付けられた Keras 浮動小数点モデル。32bit 浮動小数点重みバイアス・テンソルの 8bit 整数フォーマットへの変換は、提供されている設定を通じてインポータによって直接行われます。
- 量子化された TensorFlow™ Lite モデル（ポストトレーニング・プロセスまたはトレーニング対応プロセスにより生成）。この場合の較正は、主として、TensorFlow™ Lite ファイルをエクスポートする TFLiteConverter ユーティリティを通じて TensorFlow™ Lite フレームワークによって実行済みです。

各種の量子化スキームと量子化プロセスの実行方法の詳細については、[1]の STM32Cube.AI ツール（X-CUBE-AI）マニュアルを参照してください。

注
- 量子化演算には、データセットが 2 つ必要となります。独自データセットの準備はユーザが行ないます。
- 最終出力の精度に対する量子化の影響は、CNN モデル（そのトポロジ）だけではなく、量子化プロセスにも依存します。テスト・データセットと量子化アルゴリズムは、最終精度に大きな影響を与えます。

3.2.4 トレーニング・スクリプト

各アプリケーションにトレーニング・スクリプトが提供されています。

3.2.4.1 食品分類アプリケーション

ファイル `Utilities/AI_ressources/Food-Recognition/FoodDetection.ipynb` には、機能パックで使用される MobileNet 派生モデルをトレーニングする方法を示したサンプル・スクリプトが格納されています。機能パックで提供されているモデルのトレーニングに使用されるデータセットは非公開であるため、このトレーニング・スクリプトは Food-101 データセットのサブセットに依存しています（[4]参照）。公開されているデータセットには、101 の食品カテゴリの画像（1 カテゴリあたり 1000 枚）が格納されています。

トレーニング・プロセスを短くするために、このスクリプトでは食品カテゴリあたり 50 枚のみの画像を使用し、モデルのトレーニングを 20 エポックに制限しています。データセット全体でトレーニングを行うには、Prepare the test and train datasets セクションの変数 `max_imgs_per_class` を `np.inf` に書き換える必要があります。

注 データセット全体で完全なトレーニングを行うには、GPU の使用を推奨します。

Jupyter™ ノートブックは、`Utilities/AI_ressources/Food-Recognition/FoodDetection.py` ファイルのプレイン Python™ スクリプトとしても入手可能です。

3.2.4.2 人検出アプリケーション

ファイル `Utilities/AI_ressources/PresenceDetection/MobileNetv2_Model/train.py` には、転移学習を使用して MobileNetV2 モデルを再トレーニングする方法を示したサンプル・スクリプトが格納されています。トレーニング・スクリプトは、`***Person20***` データセットに依存しています。一般公開されている `COCO-2014` データセットから `***Person20***` データセットを構築する方法のマニュアルは、COCO 画像をフィルタする `Utilities/AI_resources/PersonDetection/MobileNetv2_Model/create_dataset.py` Python™ スクリプトと合わせて、`Utilities/AI_resources/PersonDetection/MobileNetv2_Model/README.md` にあります。ポストトレーニング量子化を行うためのサンプル Python™ スクリプトは、`Utilities/AI_resources/PersonDetection/MobileNetv2_Model/quantize.py` にあります。ポストトレーニング量子化は、TensorFlow™ の TFLiteConverter ツールを使用して Keras モデル上で実行されます。量子化演算を実行するには、サンプル・イメージが必要となります。サンプル・イメージはモデル・トレーニング・セットから抽出できます。

3.2.4.3 人数カウント・アプリケーション

再トレーニング・スクリプトの入手については、STMicroelectronics にお問い合わせください。

3.2.5 メモリ要件

STM32Cube.AI（X-CUBE-AI）ツールで生成された C モデルの統合時には、次のメモリ要件を考慮する必要があります。

- 揮発性（RAM）メモリ要件：次を割り当てるためにメモリ空間が必要です。
  - 推論作業バッファ（この資料では `activation` バッファ）。このバッファは、ニューラルネットワークの中に中間レイヤの一時的な結果を格納するために推論中に使用されます。
  - 推論入力バッファ（この資料では `nn_input` バッファ）。ニューラルネットワークの入力データの保持に使用されます。
- 不揮発性（Flash）メモリ要件：ネットワーク·モデルの重みとバイアスが含まれているテーブルを格納するためのメモリ空間が必要です。

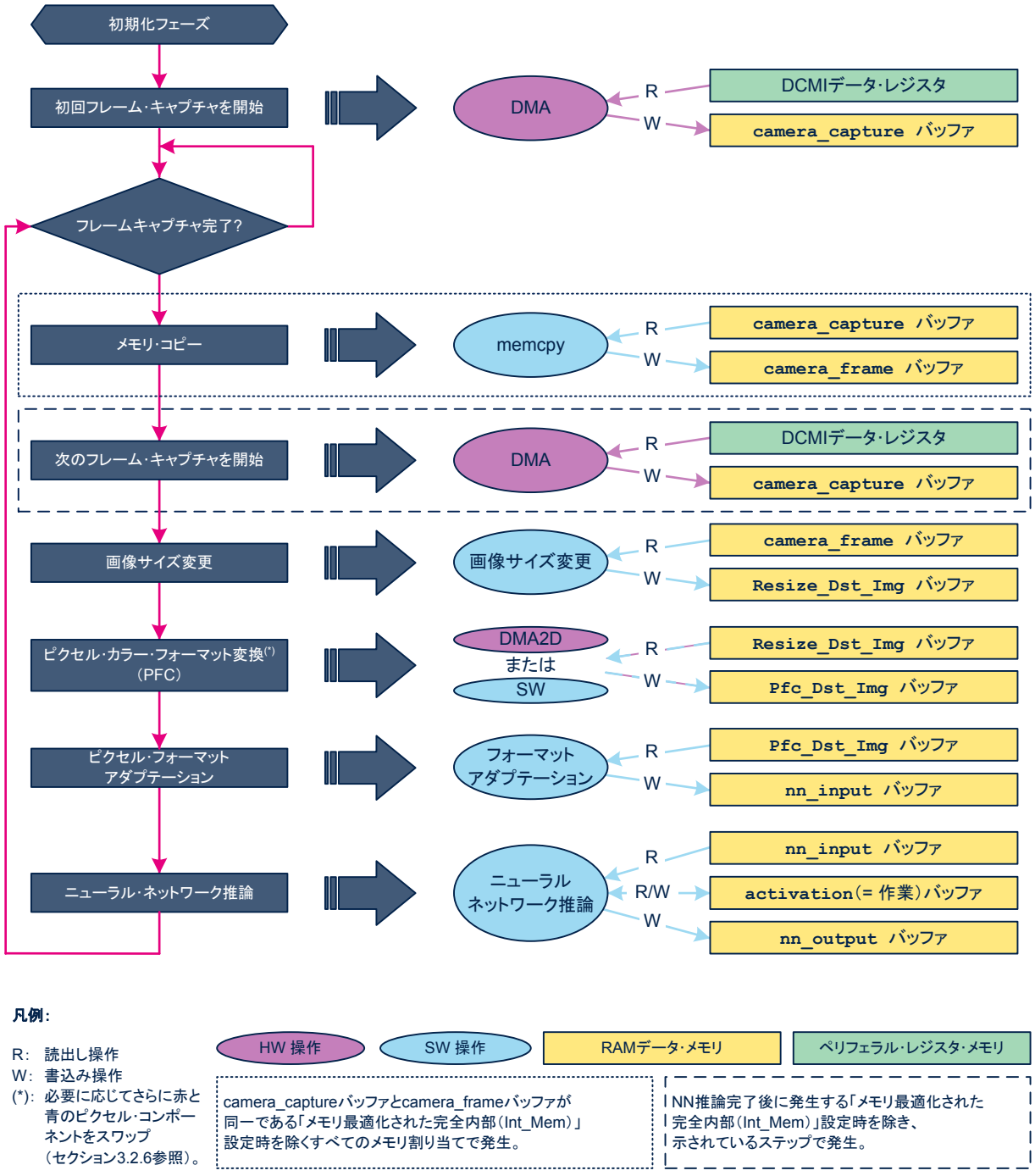
コンピュータ·ビジョン·アプリケーションの C モデルを統合する場合は、上記のメモリ要件に加えていくつかの要件が追加されます。

- 揮発性（RAM）メモリ要件：画像パイプライン（カメラ・キャプチャ、フレーム前処理）の実行時に使用されるさまざまなバッファを割り当てるためのメモリ空間が必要です。

3.2.5.1 アプリケーション実行フローと揮発性(RAM)データ・メモリ要件

コンピュータ・ビジョン・アプリケーションでは、図 7 に示されているいくつかのデータ・バッファが統合に必要となります。種々のアプリケーションの実行フロー中に必要となるさまざまなデータ・バッファを表 3、表 4、表 5 に示します。

図 7. 実行フロー中のデータ・バッファ



凡例:

R: 読出し操作
W: 書込み操作
(*): 必要に応じてさらに赤と青のピクセル・コンポーネントをスワップ(セクション3.2.6参照)。

HW 操作 | SW 操作 | RAMデータ・メモリ | ペリフェラル・レジスタ・メモリ

camera_captureバッファとcamera_frameバッファが同一である「メモリ最適化された完全内部(Int_Mem)」設定時を除くすべてのメモリ割り当てで発生。

NN推論完了後に発生する「メモリ最適化された完全内部(Int_Mem)」設定時を除き、示されているステップで発生。

アプリケーションは、次の演算を順に実行します。

1. （DMA エンジンによって DCMI データ･レジスタから）camera_capture バッファでカメラ･フレームを取得します。取得完了後、camera_capture バッファの内容は、RGB565 キャプチャ･フォーマットから ARGB8888 ディスプレイ･フォーマットへと変換され、DMA2D エンジンによって LCD フレーム･バッファ（表 3、表 4、表 5 では省略されていますが、必ず外部 SDRAM に配置されます）にコピーされます。
2. この時点で、選択したメモリ割り当て設定に応じて、camera_capture バッファの内容が camera_frame バッファにコピーされ、後続フレームのキャプチャが開始されます。
3. 想定されている CNN 入力テンソルの次元に合致するように、camera_frame バッファに格納されている画像を Resize_Dst_Img バッファの中に再スケーリングします。たとえば、食品分類 NN モデルでは、Height x Width = 224 x 224 ピクセルのような入力テンソルが必要となります。
4. Resize_Dst_Img バッファから Pfc_Dst_Img バッファへと、ピクセル･カラー･フォーマット変換および赤青カラー･チャネル･スワッピングを実行します（セクション 3.2.7 参照）。これらの演算は、DMA2D ハードウェア･エンジンまたはソフトウェア･ルーチンを使用して実行できます。たとえば、食品分類の例では、食品分類 CNN モデルが想定する 3 つの入力チャネルを提供するために、RGB565 キャプチャ･フォーマットが RGB888 フォーマットに変換されます。
5. Pfc_Dst_Img バッファの内容に含まれる各ピクセルのフォーマットを nn_input バッファに合わせて調整します。このアダプテーションは、NN モデルのトレーニングと量子化された NN が想定する量子化フォーマットによって定義されている範囲に合わせて各ピクセルの表現を変更することにあります。
例：食品分類浮動小数点モデルが 0 から 1 の間で正規化された入力データでトレーニングされた場合、入力データ（0 から 255 までの値で構成）は、推論時に 255 で割って、範囲 [0,1] に収まるようにする必要があります。
6. NN モデルの推論を実行します。nn_input バッファに加えて activation バッファが NN への入力として提供されます。分類結果は、nn_output バッファに格納されます。
7. nn_output バッファの内容を後処理し、その結果を LCD ディスプレイに表示します。

activation バッファは、NN によって推論中の作業バッファとして使用されます。そのサイズは、使用されている NN モデルに依存します。

注 activation バッファのサイズは、NN の解析時に STM32Cube.AI ツール（X-CUBE-AI）から提供されます（例についてはセクション 3.2.5.1 参照）。

nn_output バッファは、分類出力結果が格納される場所です。たとえば、FP-AI-VISION1 機能パックに提供されている食品分類 NN では、このバッファのサイズは 18 x 4 = 72 バイトとなります。18 は出力クラスの数に相当し、4 は各出力クラスの確率が浮動小数点値（単精度、4 バイトコード化）として提供されることに相当するものです。

量子化 C モデルまたは浮動小数点 C モデルを統合する場合に食品分類アプリケーションに必要となるデータ RAM 容量の詳細を表 3 に示します。

表 3. 食品分類アプリケーション用 SRAM メモリ・バッファ

SRAM データ・バッファ		SRAM データ・バッファ・サイズ（バイト）			
		食品分類 CNN			
名前	ピクセル・フォーマット	量子化 C モデル		浮動小数点 C モデル	
		VGA キャプチャ（640 x 480）	QVGA キャプチャ（320 x 240）	VGA キャプチャ（640 x 480）	QVGA キャプチャ（320 x 240）
camera_capture	16 ビット（RGB565）	600 K（640 x 480 x 2）	150 K（320 x 240 x 2）	600 K（640 x 480 x 2）	150 K（320 x 240 x 2）
camera_frame	16 ビット（RGB565）	600 K（640 x 480 x 2）	150 K（320 x 240 x 2）	600 K（640 x 480 x 2）	150 K（320 x 240 x 2）
Resize_Dst_Img	16 ビット（RGB565）	98 K（224 x 224 x 2）			
Pfc_Dst_Img	24 ビット（RGB888）	147 K（224 x 224 x 3）			
nn_input(1)	24 ビット（RGB888）	147 K（224 x 224 x 3）		588 K（224×224 x 3 x 4）	
activation(1)	-	98 K		395 K	
nn_output	-	72（18 x 4）		72（18 x 4）	

1. STM32Cube.AI ツールで C コードを生成する場合：（CLI で--allocate-inputs オプションを使用するか、GUI の高度な設定で Use activation buffer for input buffer チェックボックスを選択するかして）allocate input in activation オプションが選択された場合、STM32Cube.AI は nn_input バッファを activation バッファにオーバーレイします。これによって、アクティベーション・サイズが大きくなることがあります。ただし、この場合は nn_input バッファはそれ以降必要ありません。最終的に、全体の必要メモリ量が削減されます。与えられた食品分類量子化 C モデルでは、allocate input in activation オプションが選択された場合、生成される activation バッファのサイズは約 148KB となり、98KB より大きいものの、245KB（98 + 147）よりは小さなサイズとなります。与えられた食品分類浮動小数点 C モデルについては、nn_input バッファ（サイズは 588KB に等しい）は内部 SRAM に収まらないことから、allocate input in activation オプションは選択されません。

MobileNetV2 モデルまたは Google モデルを統合する場合に存在検出アプリケーションに必要となるデータ RAM 容量の詳細を表 4 に示します。

表 4. 人検出アプリケーション用 SRAM メモリ・バッファ

SRAM データ・バッファ	SRAM データ・バッファ・サイズ（バイト）	
	人検出 CNN	
名前	MobileNetV2 モデル	Google モデル
	QVGA キャプチャ（320 x 240）	
camera_capture	150 K（320 x 240、16bit RGB565）	150 K（320 x 240、16bit RGB565）
camera_frame	150 K（320 x 240、16bit RGB565）	150 K（320 x 240、16bit RGB565）
Resize_Dst_Img	32 K（128 x 128、16bit RGB565）	18 K（96 x 96、16bit RGB565）
Pfc_Dst_Img	48 K（128 x 128、24bit RGB888）	9 K（96 x 96、8bit グレースケール）
nn_input	48 K（128 x 128、24bit RGB888）	9 K（96 x 96、8bit グレースケール）
activation(1)	197 K	37 K
nn_output	8（2 x 4）	8（2 x 4）

1. STM32Cube.AI ツールを使用した C モデルの生成時に--allocate-inputs オプションが選択された場合。その結果、nn_input バッファと activation バッファは、サイズが 197KB のメモリ領域にオーバーレイされます。

人数カウント・アプリケーションに必要となるデータ RAM 容量の詳細を表 5 に示します。

表 5. 人数カウント・アプリケーション用 SRAM メモリ・バッファ

SRAM データ・バッファ	SRAM データ・バッファ・サイズ（バイト）
名前	人数カウント・アプリケーション （QVGA 320 x 240）
camera_capture	150 K （320 x 240、16bit RGB565）
camera_frame	150 K （320 x 240、16bit RGB565）
Resize_Dst_Img	112.5 K （240 x 240、16bit RGB565）
Pfc_Dst_Img	168.75 K （240 x 240、24bit RGB888）
nn_input	168.75 K （240 x 240、24bit RGB888）
activation(1)	233 K
nn_output	26 K （15 x 15 x 30 x 4）

1. STM32Cube.AI ツールを使用した C モデルの生成時に--allocate-inputs オプションが選択された場合。その結果、nn_input バッファと activation バッファは、サイズが 233KB のメモリ領域にオーバーレイされます。

3.2.5.2 STM32H747 内部 SRAM

STM32H747XIH6 マイクロコントローラ内部 SRAM のメモリ·マップを表 6 に示します。

表 6. STM32H747XIH6SRAM メモリ·マップ

メモリ	アドレス範囲	サイズ(KB)
DTCM RAM	0x2000 0000 - 0x2001 FFFF	128
予約済みです。	0x2002 0000 - 0x23FF FFFF	-
AXI SRAM	0x2400 0000 - 0x2407 FFFF	512
予約済みです。	0x2400 8000 - 0x2FFF FFFF	-
SRAM1	0x3000 0000 - 0x3001 FFFF	128
SRAM2	0x3002 0000 - 0x3003 FFFF	128
SRAM3	0x3004 0000 - 0x3004 7FFF	32
予約済みです。	0x3004 8000 - 0x37FF FFFF	-
SRAM4	0x3800 0000 - 0x3800 FFFF	64
予約済みです。	0x3801 0000 - 0x387F FFFF	-
BCKUP SRAM	0x3880 0000 - 0x3880 0FFF	4
合計	-	1MB

STM32H747XIH6 は約 1MB の内部 SRAM を備えています。ただし、次の点に注意してください。

- この 1MB は連続したメモリ空間ではありません。SRAM の最大ブロックは AXI SRAM で、512KB です。
- `nn_input` バッファは、メモリの連続領域に配置する必要があります。`activation` バッファについても同様です。その結果として、`nn_input` バッファまたは `activation` バッファが 512KB よりも大きい場合は、生成された C モデルは内部 SRAM のみでは実行できないという基本ルールが適用されます。この場合、追加の外部データ RAM が必要となります。

3.2.5.3 揮発性(RAM)データ·メモリにおけるバッファ配置

表 3、表 4、表 5、表 6 におけるサイズ比較の結果として、FP-AI-VISION1 FP に提供されているさまざまなアプリケーション例に対する C モデルの統合に関しては、以下の結論が得られます。

- 食品分類アプリケーションに関しては、選択したカメラ解像度を問わず、浮動小数点 C モデルの実装は内部 SRAM に完全には収まりません。このユース·ケースの実行には、ある容量の外部 RAM が必要となります。
  量子化 C モデルの実装に関しては:
  - VGA カメラ解像度が選択された場合には、実装は内部 SRAM に完全には収まりません。このユース·ケースの実行には、ある容量の外部 RAM が必要となります。
  - QVGA カメラ解像度が選択された場合には、実装は内部 SRAM に完全に収まります。外部 RAM は必要ありません。
- 人検出アプリケーションに関しては、提供されているすべての実装例で内部 SRAM に完全に収まります。外部 RAM は必要ありません。

FP-AI-VISION1 FP では、STM32 内部 SRAM に完全には収まらない実装のために、以下の 2 つのメモリ割り当て設定がサポートされています。

- 完全外部(Ext)メモリ割り当て設定:(表 3 のリストにある)すべてのバッファを外部 SDRAM に配置するものです。
- 外部 / 内部分割(Split)メモリ割り当て設定:`activation` バッファ(STM32Cube.AI(X-CUBE-AI)ツールで allocate input in activation オプションが選択された場合には `nn_input` バッファを包含しています)に加えて、`Resize_Dst_Img` バッファと `Pfc_Dst_Img` バッファ(現在のバージョンの FP では、どちらのバッファも設計により `activation` バッファにオーバーレイされています)を内部 SRAM に配置し、`camera_capture` バッファと `camera_frame` バッファを外部 SDRAM に配置するものです。(allocate input in activation オプションを選択することで)`nn_input` バッファが `activation` バッファによってオーバーレイされることが可能なサイズではない限り、(たとえば食品分類浮動小数点 C モデルのように)`nn_input` バッファも外部 SDRAM に配置されます。

いずれか設定を選択するかは、結局のところ次のトレードオフに帰着します。

- 内部 SRAM を大量に必要とする可能性のある他のユーザ·アプリケーションが使用できるように、可能な限り多くの内部 SRAM を解放する。
- 内部 SRAM のような高速メモリに作業バッファ(activation バッファ)を配置して、推論時間を短縮する。

FP-AI-VISION1 FP では、STM32 内部 SRAM に完全に収まる実装のために、以下の 2 つのメモリ割り当て設定がサポートされています。

- メモリ最適化された完全内部(Int_Mem)メモリ割り当て設定:表 3、表 4、表 5 いずれかのリストにあるすべてのバッファを内部メモリに配置するものです。この配置では、SRAM 占有最適化が可能となります。
- FPS 最適化された完全内部(Int_Fps)メモリ割り当て設定:表 3、表 4、表 5 いずれかのリストにあるすべてのバッファを内部メモリに配置するものです。この配置では、1 秒あたりに処理されるフレーム数の最適化が可能となります。

メモリ最適化された完全内部メモリ割り当て設定と FPS 最適化完全内部メモリ割り当て設定については、セクション 3.2.5.4 内部 SRAM メモリ空間の最適化 に詳しく説明されています。

3.2.5.4 内部 SRAM メモリ空間の最適化

内部 SRAM に完全に収まるユース·ケースに対しては、内部 SRAM メモリ空間を可能な限り最適化することを目的として、2 つのメモリ割り当て方式がサポートされています。

これら 2 つのメモリ割り当て方式は、activation バッファと nn_input バッファに必要な空間を最適化する目的で STM32Cube.AI ツール(X-CUBE-AI)が提供する allocate input in activation 機能に依存しています。基本的に、nn_input バッファは activation バッファの内部に割り当てられて、両方のバッファがオーバーレイされます。(ニューラルネットワーク C コードの生成時にこれらの最適化を有効にする方法については、[1]を参照してください)

第 1 のメモリ割り当て方式:メモリ最適化された完全内部(Int_Mem)

単一の固有な物理メモリ空間がすべてのバッファ(camera_capture、camera_frame、Resize_Dst_Img、Pfc_Dst_Img、nn_input、nn_output、activation)に割り当てられます。言い換えると、これら 7 つのバッファはオーバーレイされて、アプリケーション実行フローの間に 7 つの異なる目的で固有の物理メモリ空間が使用されることになります。

このメモリ空間のサイズは、activation バッファと camera_frame バッファのうち大きな方のサイズと等しくなります。たとえば、QVGA の食品分類量子化モデルでは、このメモリ空間のサイズは camera_frame バッファのサイズと等しい 150KB となります。

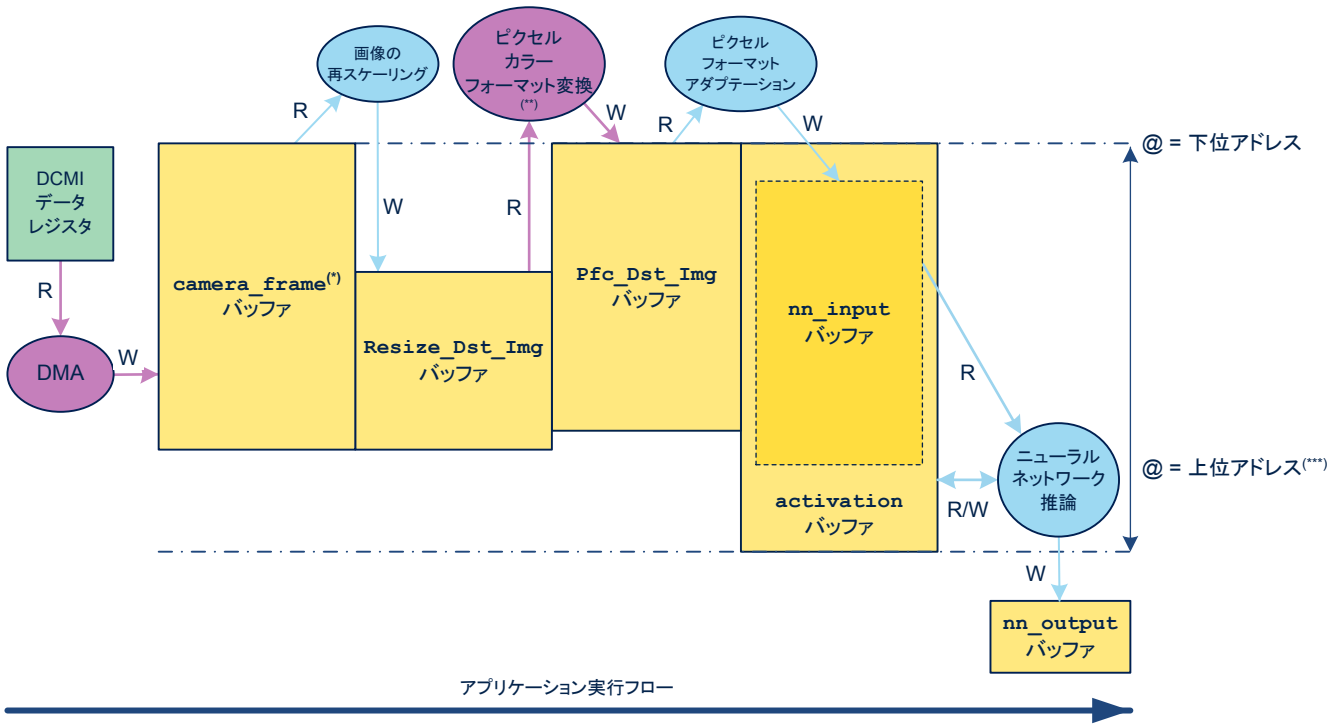
この方法の長所は、必要メモリ空間がベストな状態に最適化されることです。

この方法には主な短所も 2 つあります。

- メモリ空間に格納されたデータは、実行フローにしたがってアプリケーションが進むにつれて上書きされます(したがって、使用できなくなります)。
- NN 推論が完了するまで、メモリ空間は新たなカメラ·キャプチャに使用できません。その結果として、1 秒あたりに処理可能な最大フレーム数は最適化されません。

図 8 に示すように、必要メモリの総量が単一領域に割り当てられます。.

図 8. SRAM 割り当て - メモリ最適化方式



注:

(*): メモリ最適化された完全内部割り当て設定の場合、後続のキャプチャはNN推論が完了後のみ開始されるため、**camera_captureバッファとcamera_frameバッファは**単一の固有なバッファとなります。
(**): ピクセル·カラー·フォーマット変換は、DMA2Dを介したハードウェアまたはソフトウェアで実行可能です。
(***): 上位アドレス = 下位アドレス + Max(**activation**バッファ·サイズ, **camera_frame**バッファ·サイズ)

凡例:

R: 読出し操作
W: 書込み操作

注 食品分類アプリケーション(量子化モデルと QVGA キャプチャ)では、STM32Cube.AI ツール(allocate input in activation)のメモリ最適化機能を有効にすることで、`activation` バッファと `nn_input` バッファの両方の保持に必要なメモリはわずか 148KB となります(この機能を有効にしない場合は 147KB + 98KB = 245KB)。

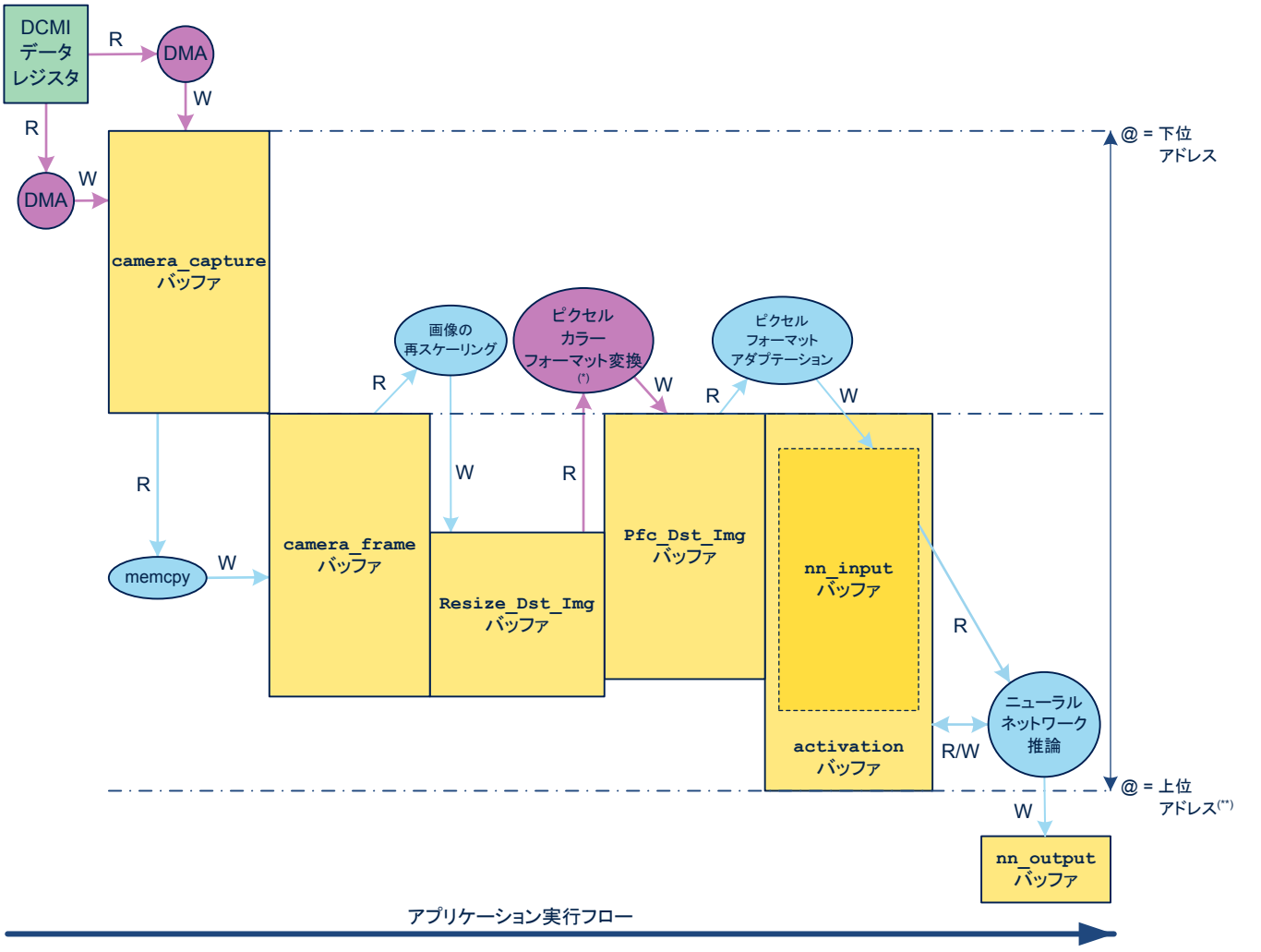
第 2 のメモリ割り当て方式:FPS 最適化された完全内部(Int_Fps)

- 第 1 の物理メモリ空間は、`camera_capture` バッファのみに割り当てられます。この方法の長所は、`camera_capture` バッファの内容が `camera_frame` バッファにコピーされると、推論の完了を待つことなく直ちに次のフレームのキャプチャが可能となることです。その結果として、1 秒あたりに処理されるフレーム数は最適な値となります。
  そのために、メモリ空間の必要量が犠牲になっています。
- 2 番目の固有な物理メモリ空間は、それ以外のすべてのバッファ(`camera_frame`、`Resize_Dst_Img`、`Pfc_Dst_Img`、`nn_input`、`nn_output`、`activation`)に割り当てられます。言い換えると、これら 6 つのバッファはオーバーレイされて、アプリケーション実行フローの間に 6 つの異なる目的で固有の物理メモリ空間が使用されることになります。
  このメモリ空間のサイズは、オーバーレイされる 5 つのバッファの中で最大のサイズに等しくなります。今回のケースでは、`activation` バッファと `camera_frame` バッファのうち大きな方のサイズとなります。たとえば、QVGA の食品分類量子化モデルでは、このメモリ空間のサイズは `camera_frame` バッファのサイズと等しい 150KB となります。

結論:食品分類アプリケーション(量子化モデルと QVGA キャプチャ)では、2 番目の割り当て方式で必要となる全体のメモリサイズは、150KB + 150KB = 300KB となります。2 番目の方式では 1 番目の方式よりも必要メモリ量は多くなりますが、2 番目の方式では 1 秒あたりに処理されるフレーム数を最適化することができます。

図 9 に示すように、必要メモリの総量が単一領域に割り当てられます。.

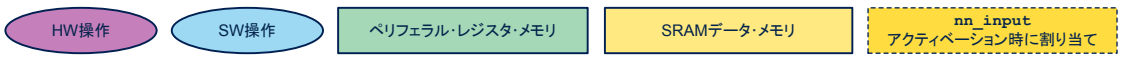
図 9. SRAM 割り当て - FPS 最適化方式



注:
(*): ピクセル·カラー·フォーマット変換は、DMA2Dを介したハードウェアまたはソフトウェアで実行可能です。
(**): 上位アドレス = 下位アドレス + `camera_frame`バッファサイズ + Max(`activation`バッファ·サイズ, `camera_frame`バッファ·サイズ)

凡例:

R: 読出し操作
W:書込み操作



注　食品分類アプリケーション（量子化モデルと QVGA キャプチャ）では、STM32Cube.AI ツール（allocate input in activation）のメモリ最適化機能を有効にすることで、`activation` バッファと `nn_input` バッファの両方の保持に必要なメモリはわずか 148KB となります（この機能を有効にしない場合は 147KB + 98KB = 245KB）。

3.2.5.5 不揮発性（Flash）メモリにおける重みとバイアスの配置

STM32Cube.AI（X-CUBE-AI）コード・ジェネレータは、重みとバイアスを含むテーブル（定数として定義）を生成します。したがって、重みとバイアスはデフォルトで内部 Flash メモリに格納されます。

重みとバイアスのテーブルが内部 Flash メモリ（サイズが 2MB で、読出し専用のデータとコードで共有）には収まらないユース・ケースが存在します。このような場合には、重みバイアステーブルを外部 Flash メモリに格納することができます。

この FP でサポートされている STM32H747I-DISCO ディスカバリ・ボードには、Quad-SPI 経由で STM32H747I MCU にインタフェースされているシリアル外部 Flash が搭載されています。この FP では、`Projects/STM32H747I-DISCO/Applications/FoodReco_MobileNetDerivative/Float_Model` ディレクトリの中に、重みバイアステーブルを外部 Flash メモリに配置する方法を示す例が提供されています（`STM32H747I-DISCO_FoodReco_Float_Ext_Qspi`、`STM32H747I-DISCO_FoodReco_Float_Split_Qspi`、`STM32H747I-DISCO_FoodReco_Float_Split_Sdram`、`STM32H747I-DISCO_FoodReco_Float_Ext_Sdram` の設定）。

次の手順に従って、重みバイアステーブルを Q-SPI 外部 Flash メモリにロードします。

1. フラグ `WEIGHT_QSPI` を 1 に、`WEIGHT_QSPI_PROGED` を 0 にそれぞれ定義して、Quad-SPI メモリ・インタフェース（0x9000 0000）に相当するメモリ範囲にメモリ配置セクションを定義し、重みバイアステーブルをその中に配置します。
   推論を高速化するために、（たとえば、`STM32H747I-DISCO_FoodReco_Float_Split_Sdram` と `STM32H747I-DISCO_FoodReco_Float_Ext_Sdram` の設定でデモンストレーションされているように）フラグ `WEIGHT_EXEC_EXTRAM` を 1 にセットして、プログラムの初期化時に重みバイアステーブルが外部 Q-SPI Flash メモリから内部 SDRAM メモリにコピーされるようにもできます。

2. アプリケーション全体のバイナリを.hex ファイル(.bin ファイルではありません)として生成します。STM32CubeProgrammer(STM32CubeProg)ツール[5]を使用して、.hex ファイルをロードします。最初に、STM32H747I-DISCO ディスカバリ·ボード用の外部 Flash ローダを選択します。図 10 に示されているように、External loaders ビューにアクセスするには、左側の External loaders タブを選択する必要があります。正しい Flash ローダを選択したら、図 11 に示されている Erasing & programming にアクセスするために、左側の Erasing & programming タブを選択する必要があります。
図 10 と図 11 は、STM32CubeProgrammer(STM32CubeProg)ツールのスナップショットであり、アプリケーションのフル·バイナリを Flash メモリにプログラムする手順のシーケンスが示されています。

図 10. Flash プログラミング(1/2)

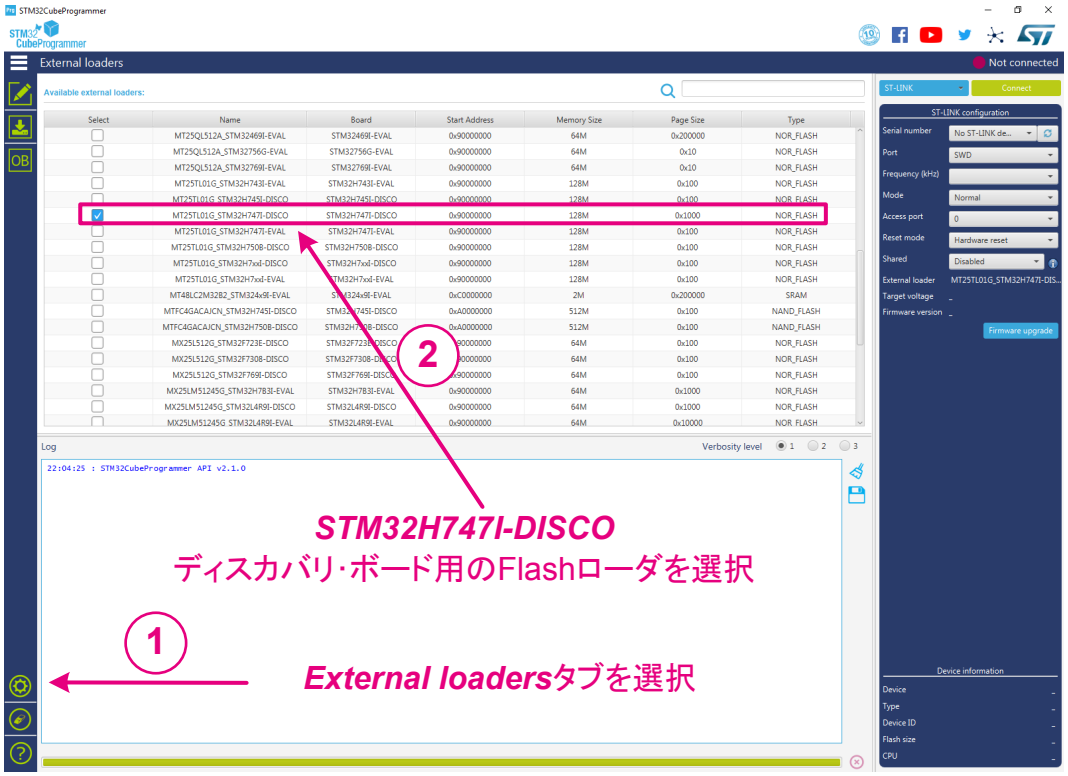
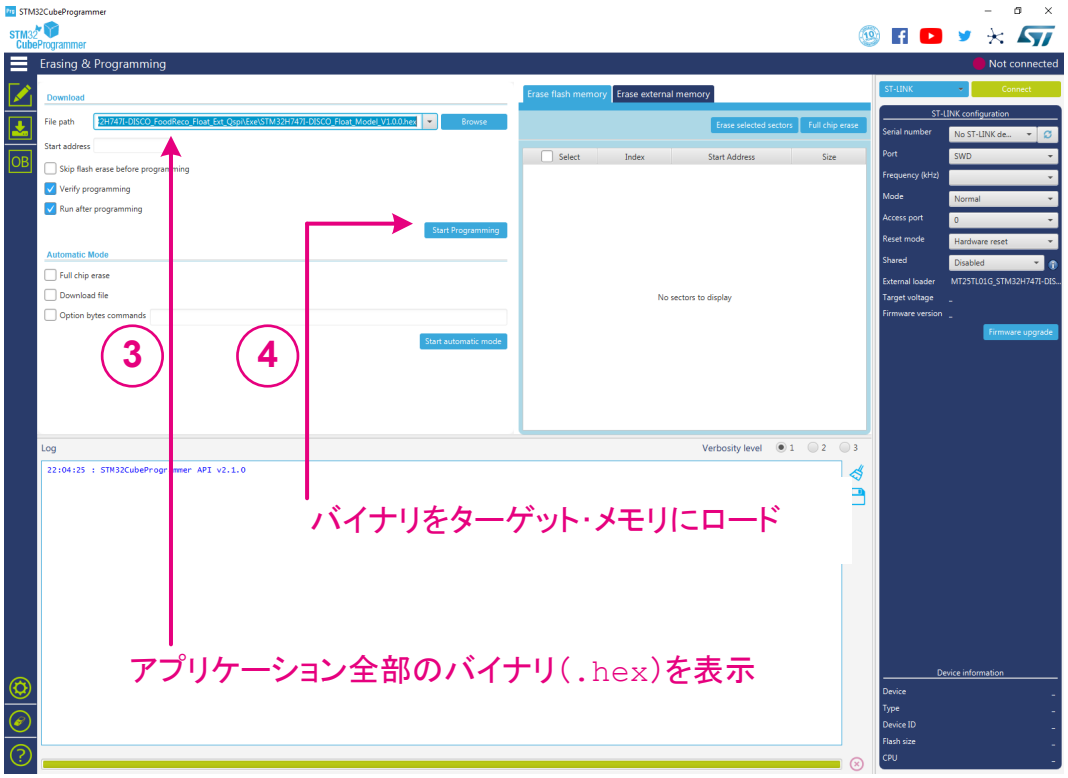


図 11. Flash プログラミング (2/2)



3. 重みとバイアスのテーブルが外部メモリにロードされると、フラグ WEIGHT_QSPI_PROGED を 1 に設定することにより、IDE 経由でデバッグを続けることができます。
フラグ WEIGHT_QSPI を 0 に設定すると、内部 Flash メモリに重みバイアステーブルが配置されたプログラムが生成されます。

コンパイル・フラグの組み合わせを表 7 に示します。

表 7. コンパイル・フラグ

影響		コンパイル・フラグ		
		WEIGHT_QSPI	WEIGHT_QSPI_PROGED	WEIGHT_EXEC_EXTRAM
内部 Flash メモリに重みバイアステーブルが配置されたバイナリを生成します。		0	0	0
		0	0	1
		0	1	0
		0	1	1
外部 Q-SPI Flash メモリに重みバイアステーブルが配置されたバイナリを生成します。(1)	-	1	0	0
	起動時に外部 Q-SPI Flash メモリから外部 SDRAM に重みバイアステーブルがコピーされます。	1	0	1
(ロード済みであるため)重みバイアステーブルを含まないバイナリを生成します。	-	1	1	0
	起動時に外部 Q-SPI Flash メモリから外部 SDRAM に重みバイアステーブルがコピーされます。	1	1	1

1. STM32CubeProgrammer ツール(STM32CubeProg)を使用してロードします。

外部 Q-SPI Flash メモリから重みバイアステーブルにアクセスした場合の性能と、外部 SDRAM と内部 Flash メモリの比較については、セクション 3.2.8 アプリケーションの性能を参照してください。

新規のモデルから新規の C コードを生成する場合の通常のやり方は、(セクション 2.1 生成コードの統合に説明されているように)既存の network.c、network.h、network_data.c、network_data.h ファイルを生成されたファイルへと置き換える方法です。外部メモリへの配置が必要な場合には、既存の network_data.c ファイルは置き換えないでください。その代わりに、既存の network_data.c ファイルに含まれている重みバイアステーブルの内容(名前は s_network_weights[])のみを、生成された network_data.c ファイルに含まれているテーブルへと置き換える必要があります。

3.2.5.6 揮発性データと不揮発性データのメモリ内配置の要約

FP-AI-VISION1 機能パックには、メモリに揮発性データを割り当てる 4 つの異なる方式の実装がデモンストレーションされています。

- FPS 最適化された完全内部メモリ(Int_Fps):
  (表 3、表 4、表 5 のリストにある)すべてのバッファが内部 SRAM に配置されます。全体システムを内部メモリから実行可能であるように、(図 9 に示すように)一部のメモリ・バッファがオーバーレイされます。
  このメモリ・レイアウト方式では、camera_capture バッファがオーバーレイされないために、推論実行中の新規カメラ・キャプチャが可能となり、1 秒あたりのフレーム数(FPS)が最大化されます。
- メモリ最適化された完全内部メモリ(Int_Mem):
  (表 3、表 4、表 5 のリストにある)すべてのバッファが内部 SRAM に配置されます。全体システムを内部メモリから実行可能にするように、(図 8 に示すように)一部のメモリ・バッファがオーバーレイされます。
  このメモリ・レイアウト方式では、camera_capture バッファと camera_frame バッファが単一の固有なバッファであり、内部 SRAM の占有を可能な限り最適化するようにオーバーレイされます。
- 完全外部メモリ(Ext):
  (表 3、表 4、表 5 のリストにある)すべてのバッファが外部 SDRAM に配置されます。
- 内部 / 外部メモリ分割(Split):
  camera_capture バッファと camera_frame バッファが外部 SDRAM に配置されます。activation バッファ(STM32Cube.AI(X-CUBE-AI)ツールで allocate input in activation オプションが選択された場合には nn_input バッファを包含しています)に加えて、Resize_Dst_Img バッファと Pfc_Dst_Img バッファ(現在のバージョンの FP では、どちらのバッファも設計により activation バッファにオーバーレイされています)を内部 SRAM に配置します。この割り当て方式では、(allocate input in activation オプションを選択することで) nn_input バッファが activation バッファによってオーバーレイされることが可能なサイズではない限り、(たとえば食品分類浮動小数点 C モデルのように)nn_input バッファも外部 SDRAM に配置されます。

FP-AI-VISION1 機能パックには、メモリに格納された不揮発性データに(推論時に)アクセスする 3 つの異なる方法の実装もデモンストレーションされています。

- 内部 Flash メモリからのアクセス
- 外部 Q-SPI Flash メモリからのアクセス
- 外部 SDRAM からのアクセス:この場合には、不揮発性データは内部 Flash メモリまたは外部 Q-SPI Flash メモリに格納され、プログラム起動時に外部 SDRAM にコピーされます。

FP-AI-VISION1 で使用可能な IAR Embedded Workbench®プロジェクトのさまざまな設定のリストを表 8 に示します。

表 8. IAR Embedded Workbench®プロジェクトの設定とメモリ構成の一覧

IAR Embedded Workbench® IAR Embedded Workbench®プロジェクト設定名	揮発性データのメモリ割り当て方式	重みとバイアスの格納に使用されるメモリ
STM32H747I-DISCO_FoodReco_Float_Ext	完全外部	内部 Flash メモリ
STM32H747I-DISCO_FoodReco_Float_Ext_Qspi		外部 Q-SPI Flash メモリ
STM32H747I-DISCO_FoodReco_Float_Ext_Sdram		外部 SDRAM
STM32H747I-DISCO_FoodReco_Float_Split	内部 / 外部分割	内部 Flash メモリ
STM32H747I-DISCO_FoodReco_Float_Split_Qspi		外部 Q-SPI Flash メモリ
STM32H747I-DISCO_FoodReco_Float_Split_Sdram		外部 SDRAM
STM32H747I-DISCO_FoodReco_Quantized_Ext	完全外部	内部 Flash メモリ
STM32H747I-DISCO_FoodReco_Quantized_Split	内部 / 外部分割	
STM32H747I-DISCO_FoodReco_Quantized_Int_Mem	メモリ最適化された完全内部	
STM32H747I-DISCO_FoodReco_Quantized_Int_Fps	FPS 最適化された完全内部	
STM32H747I-DISCO_PersonDetect_Google		
STM32H747I-DISCO_PersonDetect_MobileNetv2		
STM32H747I-DISCO_PeopleCounting		

3.2.6 カメラ·ピクセル·クロック

3.2.6.1 USB ウェブカメラ·アプリケーション

FCC 認証への適合のため、B-CAMS-OMV カメラ·モジュール·バンドル（OV5640 カメラ）の PCLK カメラ·ピクセル·クロックは 12MH zに制限されています。PCLK = 12MHz の場合、OV5640 カメラ出力のフレーム·レートは、VGA と QVGA どちらの解像度でも 7.5fps となります。

さらに高い FPS を得るには、`CAMERA_BOOST_PCLK = 1` を使用してピクセル·クロックを高めることができます。これにより、OV5640 カメラのピクセル·クロックは以下のようになります。

- VGA 解像度で 15 fps を得るには 24MHz
- QVGA 解像度で 30 fps を得るには 48MHz

注 STM32F4DIS-CAM（OV9655）カメラを使用している場合は、`CAMERA_BOOST_PCLK` による影響はありません。

USB ウェブカメラ·アプリケーションの場合、USB ディスクリプタで定義されたフレーム·レートを得るためには、OV5640 ピクセル·クロックを 24MHz に上げたり、解像度によっては 48MHz まで上げたりする必要があります。`CAMERA_BOOST_PCLK` のプリプロセッサ定義は、コンパイラ設定で有効にすることも、ファイル `stm32h747i_discovery_camera_ex.c` の中で直接有効することも可能です。

```
/*
OV5640 カメラを最適モードで動作させるには、CAMERA_BOOST_PCLK=1にセットします。
 注:カメラを最適モードで動作させると、ピクセル周波数が12MHzを超えて、FCC認証が保証されなくなります。
*/

#ifndef CAMERA_BOOST_PCLK
#define CAMERA_BOOST_PCLK   1
#endif
```

利用条件に関する説明

CAMERA_BOOST_PCLK コンパイラ·フラグを 1 にセットすることで、STMicroelectronics が提供するキットのデフォルト設定を変更することになります。このような新規の設定に伴い、ユーザは次の利用条件を遵守する必要があります。

FCC 未承認の評価ボードに適用される通知

このキットは、(1)製品開発者による、キットに関連した電子部品、回路、ソフトウェアの評価ならびにそれらを最終製品に組み込むか否かの決定と、(2)ソフトウェア開発者による、最終製品で使用するソフトウェア·アプリケーションの作成とが可能であるように設計されています。このキットは最終製品ではありません。すべての必要な FCC 機器認証を最初に取得しない限り、組み立て状態での転売も販売もできません。動作は、この製品が免許を取得した無線局に対して有害な干渉を引き起こさず、この製品が有害な干渉を許容するという条件で適用されます。組み立て状態のキットが 47 CFR チャプター I のパート 15、パート 18、またはパート 95(FCC 規則)に基づいて動作するように設計されている場合を除き、キットの操作者は、FCC 免許取得者の権限の下で操作するか、このチャプターのパート 5 に基づく実験許可を得る必要があります。

3.2.6.2 その他のアプリケーション

その他のアプリケーション(人検出、食品分類、人数カウント)では、 OV5640 カメラのピクセル·クロックはデフォルトで 48MHz に設定されます。これにより、VGA と QVGA 両方のカメラ·キャプチャ·モードで 30 fps のレートが可能となります。

3.2.7 ピクセル·データの順序

動作モードによって(セクション 3.2.12 組み込みの検証、キャプチャ、テスト参照)、入力データ画像はカメラ·センサからのものであることも、STM32H747I-DISCO ボードに存在する microSD™に格納されている .bmp ファイルからのものであることもあります。

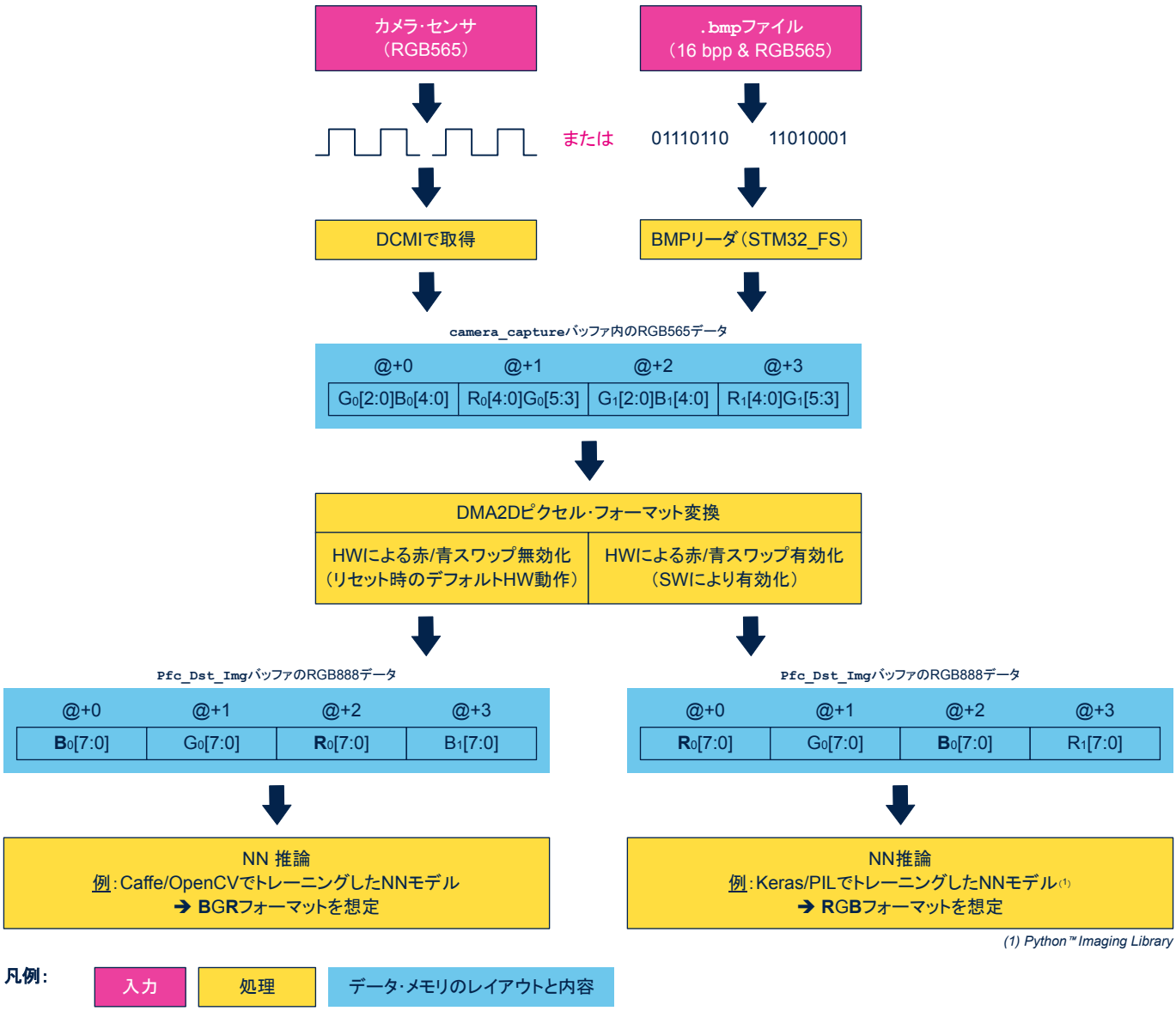
カメラ·センサは、RGB565 フォーマットでデータを出力するように設定されています。カメラからの取得中に、バイト列は、第 1 バイト G[2:0]B[4:0] の後に第 2 バイト R[4:0]G[5:3] が送信されるという順序で、DCMI インタフェースを通じて送信されます。

.bmp ファイルには、RGB565 フォーマットで格納されたピクセルあたり 16 ビットのデータが含まれていることが想定されていますこの場合、.bmp ファイルから入力データ画像を読み出すと、バイト G[2:0]B[4:0] が最初にメモリに格納され、バイト R[4:0]G[5:3] が次のメモリ·バイトに格納されます。

その結果として、入力データ画像がカメラ·センサからのものであろうと、.bmp ファイルからのものであろうと、camera_capture バッファ内のデータ·メモリ·レイアウトは完全に同じとなります。

各ステージの後のピクセル·データ·メモリ·レイアウトを図 12 に示します。

図 12. ピクセルの順序とデータ·メモリ·レイアウト



(1) Python™ Imaging Library

食品分類 NN モデルと MobileNet-V2 人検出 NN モデルは、入力データ·テンソルが RGB フォーマットでメモリに格納されることを想定した方法ですべてトレーニングされます。

したがって、赤と青のコンポーネントをある時点でスワップする必要があります。機能パックファームウェアでは、スワップはピクセル·フォーマット変換（PFC）演算中に DMA2D によって行なわれます。

ハードウェアがリセットされると、PFC を実行するように設定されている場合、DMA2D エンジンは、図 12 に示されているように青コンポーネントを最初にメモリに格納します。.赤コンポーネントを最初にメモリに格納する必要がある場合、赤コンポーネントと青コンポーネントをスワップするように DMA2D を設定する必要があります。そのためには、DMA2D ピクセル·フォーマット変換演算を開始する前に、（`PreprocContext` の一部である）変数 `red_blue_swap` を 1 にセットする必要があります（ファイル `ai_utilities.c` の関数 `Run_Preprocessing()` 参照）。

3 つの異なるピクセル（赤、緑、青）のメモリ内容の例を図 13 に示します。

図 13. ピクセルの順序とデータ・メモリ配置の例

Camera RGB565 data
(camera_capture buffer)

```
addr    @+0      @+1      @+2      @+3    ...
bin  00000000 11111000 00000000 11111000  ...
hex    0x00     0xF8     0x00     0xF8    ...

addr    @+0      @+1      @+2      @+3    ...
bin  11100000 00000111 11100000 00000111  ...
hex    0xE0     0x07     0xE0     0x07    ...

addr    @+0      @+1      @+2      @+3    ...
bin  00011111 00000000 00011111 00000000  ...
hex    0x1F     0x00     0x1F     0x00    ...
```

DMA2D RGB888 output with red_blue_swap=0
(Pfc_Dst_Img buffer)

```
addr    @+0      @+1      @+2      @+3    ...
bin  00000000 00000000 11111111 00000000  ...
hex    0x00     0x00     0xFF     0x00    ...

addr    @+0      @+1      @+2      @+3    ...
bin  00000000 11111111 00000000 00000000  ...
hex    0x00     0xFF     0x00     0x00    ...

addr    @+0      @+1      @+2      @+3    ...
bin  11111111 00000000 00000000 11111111  ...
hex    0xFF     0x00     0x00     0xFF    ...
```

DMA2D RGB888 output with red_blue_swap=1
(Pfc_Dst_Img buffer)

```
addr    @+0      @+1      @+2      @+3    ...
bin  11111111 00000000 00000000 11111111  ...
hex    0xFF     0x00     0x00     0xFF    ...

addr    @+0      @+1      @+2      @+3    ...
bin  00000000 11111111 00000000 00000000  ...
hex    0x00     0xFF     0x00     0x00    ...

addr    @+0      @+1      @+2      @+3    ...
bin  00000000 00000000 11111111 00000000  ...
hex    0x00     0x00     0xFF     0x00    ...
```

3.2.8 アプリケーションの性能

3.2.8.1 前処理のタイミング

サポート対象アプリケーションごとに測定されたカメラ·キャプチャ時間と前処理時間を表 9、表 10、表 11 に示します。

表 9. 食品分類アプリケーション例のフレーム·キャプチャ時間と前処理時間の測定値

測定(1) (ms)	食品分類		
	VGA キャプチャ(640 x 480)		QVGA キャプチャ(320 x 240)
	量子化 C モデル	浮動小数点 C モデル	量子化 C モデル
カメラ·フレーム·キャプチャ時間(2) (3)	~35		~35
画像再スケーリング時間(4)	~5 から~7		~5
ピクセル色変換(5)	< 1 から~2		< 1
ピクセル·フォーマット·アダプテーション時間(6)	~1 から~3	~9 から~11	~1

1. 測定条件:STM32H747、CPU クロック 400MHz、オプション-O3 付きの EWARM v8.50.6 でコンパイルされたコード。
2. 露光時間が変化するため、値は照明条件に依存します。
3. STM32H747I-DISCO ディスカバリ·ボードに接続された B-CAMS-OMV カメラ·モジュール·バンドルの値。.
4. FP-AI-VISION1 では、再スケーリング手法として最近傍アルゴリズムが使用されます。`camera_capture` バッファと `Resize_Dest_Img` バッファのメモリ位置がタイミングに影響します。
5. `Resize_Dest_Img` バッファと `Pfc_Dest_Img` バッファのメモリ位置がタイミングに影響します。
6. 量子化モデルを扱う場合は、事前計算されたルックアップ·テーブルがピクセル·フォーマット·アダプテーションに使用されます。`Pfc_Dest_Img` バッファと `nn_input` バッファのメモリ位置がタイミングに影響します。

表 10. 人検出アプリケーション例のフレーム·キャプチャ時間と前処理時間の測定値

測定(1) (ms)	人検出 QVGA キャプチャ(320 x 240)	
	MobileNetV2 モデル (128 x 128、24bit RGB888)	Google モデル (96 x 96、8bit グレースケール)
カメラ·フレーム·キャプチャ時間(2) (3)	~35	
画像再スケーリング時間(4)	~2	~1
ピクセル色変換(5)	< 1	
ピクセル·フォーマット·アダプテーション時間(6)	< 1	

1. 測定条件:STM32H747、CPU クロック 400MHz、オプション-O3 付きの EWARM v8.50.6 でコンパイルされたコード。
2. 露光時間が変化するため、値は照明条件に依存します。
3. STM32H747I-DISCO ディスカバリ·ボードに接続された B-CAMS-OMV カメラ·モジュール·バンドルの値。.
4. FP-AI-VISION1 では、再スケーリング手法として最近傍アルゴリズムが使用されます。
5. `Resize_Dest_Img` バッファと `Pfc_Dest_Img` バッファのメモリ位置がタイミングに影響します。
6. 量子化モデルを扱う場合は、事前計算されたルックアップ·テーブルがピクセル·フォーマット·アダプテーションに使用されます。

表 11. 人数カウントアプリケーション例のフレーム・キャプチャ時間と前処理時間の測定値

測定(1) (ms)	人数カウント：240 x 240 x 3、24bit RGB888 QVGA キャプチャ（320 x 240）
カメラ・フレーム・キャプチャ時間(2) (3)	~35
画像再スケーリング時間(4)	~5
ピクセル色変換	< 1
ピクセル・フォーマット・アダプテーション時間(5)	< 1

1. 測定条件：STM32H747、CPU クロック 400MHz、オプション−O3 付きの EWARM v8.50.6 でコンパイルされたコード。
2. 露光時間が変化するため、値は照明条件に依存します。
3. STM32H747I-DISCO ディスカバリ・ボードに接続された B-CAMS-OMV カメラ・モジュール・バンドルの値。.
4. FP-AI-VISION1 では、再スケーリング手法として最近傍アルゴリズムが使用されます。
5. 量子化モデルを扱う場合は、事前計算されたルックアップ・テーブルがピクセル・フォーマット・アダプテーションに使用されます。

3.2.8.2 サポート対象の設定

可能性のあるさまざまなモデル（浮動小数点、量子化）、カメラ解像度（VGA、QVGA）、揮発性メモリのデータ位置（完全外部、分割、FPS 最適化された完全内部、メモリ最適化された完全内部）、推論実行中の重みバイアス・テーブルの位置（内部 Flash メモリ、外部 Q-SPI Flash メモリ、外部 SDRAM）を考慮すると、可能性のある設定は数多くあります。

サポート対象アプリケーションごとに、FP-AI-VISION1 機能パックがサポートしている設定を要約したものを表 12、表 13、表 14 に示します。

表 12. 食品分類アプリケーションに対して FP-AI-VISION1 機能パックがサポートしている設定

揮発性メモリ・レイアウト方式	重みバイアス・テーブルの位置	食品分類			
		量子化 C モデル		浮動小数点 C モデル	
		VGA キャプチャ（640 x 480）	QVGA キャプチャ（320 x 240）	VGA キャプチャ（640 x 480）	QVGA キャプチャ（320 x 240）
メモリ最適化された完全内部	内部 Flash メモリ	不可	はい	不可	不可
	外部 Q-SPI Flash メモリ		サポートされません		
	外部 SDRAM		サポートされません		
FPS 最適化された完全内部	内部 Flash メモリ	不可	はい	不可	不可
	外部 Q-SPI Flash メモリ		サポートされません		
	外部 SDRAM		サポートされません		
完全外部	内部 Flash メモリ	はい	サポートされません	はい	サポートされません
	外部 Q-SPI Flash メモリ	サポートされません		はい	
	外部 SDRAM	サポートされません		はい	
外部 / 内部分割	内部 Flash メモリ	はい	サポートされません	はい	サポートされません
	外部 Q-SPI Flash メモリ	サポートされません		はい	
	外部 SDRAM	サポートされません		はい	

表 13. 人検出アプリケーションに対して FP-AI-VISION1 機能パックがサポートしている設定

揮発性メモリ·レイアウト方式	重みバイアス·テーブルの位置	人検出	
		MobileNetV2 モデル	Google モデル
メモリ最適化された完全内部	内部 Flash メモリ	いいえ	いいえ
	外部 Q-SPI Flash メモリ		
	外部 SDRAM		
FPS 最適化された完全内部	内部 Flash メモリ	はい	はい
	外部 Q-SPI Flash メモリ	いいえ	いいえ
	外部 SDRAM		
完全外部	内部 Flash メモリ	いいえ	いいえ
	外部 Q-SPI Flash メモリ		
	外部 SDRAM		
スプリット 外部 / 内部分割	内部 Flash メモリ	いいえ	いいえ
	外部 Q-SPI Flash メモリ		
	外部 SDRAM		

表 14. 人数カウント·アプリケーションに対して FP-AI-VISION1 機能パックがサポートしている設定

揮発性メモリ·レイアウト方式	重みバイアス·テーブルの位置	人数カウント
メモリ最適化された完全内部	内部 Flash メモリ	いいえ
	外部 Q-SPI Flash メモリ	
	外部 SDRAM	
FPS 最適化された完全内部	内部 Flash メモリ	はい
	外部 Q-SPI Flash メモリ	いいえ
	外部 SDRAM	
完全外部	内部 Flash メモリ	いいえ
	外部 Q-SPI Flash メモリ	
	外部 SDRAM	
スプリット 外部 / 内部分割	内部 Flash メモリ	いいえ
	外部 Q-SPI Flash メモリ	
	外部 SDRAM	

3.2.8.3 サポート対象アプリケーションの実行性能

以下の表 15、表 16、表 17、表 18 に記載されている値は、EWARM IDE で構築されたアプリケーション・バイナリを使用して測定した結果です。

次の条件における食品分類アプリケーションの実行性能を表 15 に示します。

- IAR Systems EWARM v8.50.6
- `-O3` オプション
- Cortex®-M7 コア・クロック周波数を 400MHz に設定
- B-CAMS-OMV カメラ・モジュール・バンドル

表 15. 食品分類アプリケーションの実行性能

C モデル	カメラ解像度	揮発性メモリ・レイアウト方式	重みバイアス・テーブルの位置	NN 推論時間(ms)	1 秒あたりの処理フレーム数(FPS)(1)	出力精度(%)	バイナリ・ファイル(2)
浮動小数点	VGA	完全外部	内部 Flash メモリ	242	3.7	約 73	A
			外部 Flash メモリ	283	3.3		B
			外部 SDRAM	267	3.5		セクション 3.2.8.3 サポート対象アプリケーションの実行性能
		スプリット	内部 Flash メモリ	220	4.1		D
			外部 Flash メモリ	261	3.6		E
			外部 SDRAM	243	3.9		F
量子化(3)	VGA	完全外部	内部 Flash メモリ	100	9.2	約 72.5	G
		スプリット		79	11.8		H
	QVGA	メモリ最適化された完全内部		79	8.5		I
		FPS 最適化された完全内部		79	11.8		J

1. FPS は、キャプチャ時間、前処理時間、推論時間を考慮して計算されています。
2. IAR Embedded Workbench®設定 > バイナリファイル。
3. 量子化ツール(STM32Cube.AI ツールによる提供)を使用して UaUa per layer 量子化スキームで量子化された食品分類 CNN モデル

表 15 に記載されている FPS 値は最大値です。FPS 値は、照明条件によって異なることがあります。事実、FPS 値には推論時間に加えて、前処理時間とカメラ・キャプチャ時間も含まれています(表 9 参照)。キャプチャ時間は、露光時間による照明条件に依存します。

表 15 のシナリオに対応付けられている、`Binary` フォルダに格納されているバイナリ・ファイルのリストを以下に示します。

- A: `STM32H747I-DISCO_FoodReco_Float_Ext > STM32H747I-DISCO_Food_Std_Float_Ext_IntFlash_V300.bin`
- B: `STM32H747I-DISCO_FoodReco_Float_Ext_Qspi > STM32H747I-DISCO_Food_Std_Float_Ext_QspiFlash_V300.hex`
- C: `STM32H747I-DISCO_FoodReco_Float_Ext_Sdram > STM32H747I-DISCO_Food_Std_Float_Ext_ExtSdram_V300.hex`
- D: `STM32H747I-DISCO_FoodReco_Float_Split > STM32H747I-DISCO_Food_Std_Float_Split_IntFlash_V300.bin`
- E: `STM32H747I-DISCO_FoodReco_Float_Split_Qspi > STM32H747I-DISCO_Food_Std_Float_Split_QspiFlash_V300.hex`

- F: STM32H747I-DISCO_FoodReco_Float_Split_Sdram > STM32H747I-DISCO_Food_Std_Float_Split_ExtSdram_V300.bin
- G: STM32H747I-DISCO_FoodReco_Quantized_Ext > STM32H747I-DISCO_Food_Std_Quant8_Ext_IntFlash_V300.bin
- H: STM32H747I-DISCO_FoodReco_Quantized_Split > STM32H747I-DISCO_Food_Std_Quant8_Split_IntFlash_V300.bin
- I: STM32H747I-DISCO_FoodReco_Quantized_Int_Mem > STM32H747I-DISCO_Food_Std_Quant8_IntMem_IntFlash_V300.bin
- J: STM32H747I-DISCO_FoodReco_Quantized_Int_Fps > STM32H747I-DISCO_Food_Std_Quant8_IntFps_IntFlash_V300.bin

最適化モデル（FP-AI-VISION1 機能パックの一部として提供されたものではありません。この特定のバージョンに関する詳細は、STMicroelectronics にお問い合わせください）で生成された認識アプリケーションの実行性能を表 16 に示します。測定条件は次のとおりです。

- IAR Systems EWARM v8.50.6
- -O3 オプション
- Cortex®-M7 コア・クロック周波数を 400MHz に設定
- B-CAMS-OMV カメラ・モジュール・バンドル

表 16. 最適化された食品分類アプリケーションの実行性能

C モデル	カメラ解像度	揮発性メモリ・レイアウト方式	重みバイアス・テーブルの位置	NN 推論時間（ms）	1 秒あたりの処理フレーム数（FPS）(1)	出力精度（%）	バイナリ・ファイル
浮動小数点	VGA	完全外部	内部 Flash メモリ	492	1.9	約 78	K
		スプリット		不可			
量子化(2)	VGA	完全外部		220	4.3	約 77.5	L
		スプリット		145	6.6		M
	QVGA	メモリ最適化された完全内部		145	5.4		N
		FPS 最適化された完全内部		145	6.6		O

1. FPS は、キャプチャ時間、前処理時間、推論時間を考慮して計算されています。
2. 量子化ツール（STM32Cube.AI ツールによる提供）を使用して UaUa per layer 量子化スキームで量子化された食品分類 CNN モデル

表 16 のシナリオに対応付けられている、Binary フォルダに格納されているバイナリ・ファイルのリストを以下に示します。

- K: STM32H747I-DISCO_Food_Opt_Float_Ext_IntFlash_V300.bin
- L: STM32H747I-DISCO_Food_Opt_Quant8_Ext_IntFlash_V300.bin
- M: STM32H747I-DISCO_Food_Opt_Quant8_Split_IntFlash_V300.bin
- N: STM32H747I-DISCO_Food_Opt_Quant8_IntMem_IntFlash_V300.bin
- O: STM32H747I-DISCO_Food_Opt_Quant8_IntFps_IntFlash_V300.bin

ストレート・モデル（表 15）は、一部のケースで最適化モデル（表 16）よりも高い実行性能を示しています。ただし、これには精度が犠牲になっており、いずれのケースでも CNN 作業バッファ（activation バッファ）は内部 SRAM に配置されていることから、最適化モデルを使用した方が高精度となります。

次の条件における人検出アプリケーションの実行性能を表 17 に示します。

- IAR Systems EWARM v8.50.6
- `-O3` オプション
- Cortex®-M7 コア・クロック周波数を 400MHz に設定
- B-CAMS-OMV カメラ・モジュール・バンドル

表 17. 人検出アプリケーションの実行性能

C モデル	カメラ解像度	揮発性メモリ・レイアウト方式	重みバイアス・テーブルの位置	NN 推論時間(ms)	1 秒あたりの処理フレーム数(FPS)(1)	出力精度(%)(2)	バイナリ・ファイル(3)
MobileNetV2	QVGA	完全内部	内部 Flash メモリ	145	6.8	92	セクション 3.2.8.3 サポート対象アプリケーションの実行性能
Google				41	23.3	87	セクション 3.2.8.3 サポート対象アプリケーションの実行性能

1. FPS は、キャプチャ時間、前処理時間、推論時間を考慮して計算されています。
2. ***Person20***データセットに対するトップ 1 精度。
3. IAR Embedded Workbench®設定 > バイナリファイル。

表 17 のシナリオに対応付けられている、`Binary` フォルダに格納されているバイナリ・ファイルのリストを以下に示します。

- P: `STM32H747I-DISCO_PersonDetect_MobileNetv2 > STM32H747I-DISCO_Person_MobileNetv2_Quant8_IntFps_IntFlash_V300.bin`
- Q: `STM32H747I-DISCO_PersonDetect_Google > STM32H747I-DISCO_Person_Google_Quant8_IntFps_IntFlash_V300.bin`

最終精度に対する CNN の影響

上の量子化プロセスで説明したように、CNN のタイプは最終精度に影響します。このことは、表 15 と表 16 の「出力精度」の列に示されています。

- 表 15 には、推論時間を短縮するようにトポロジが最適化されたネットワークに関係した数値が記載されています。ただし、これには出力精度が犠牲になっています(約 10%の精度低下)。
- 表 16 には、出力精度を最適化するようにトポロジが最適化されたネットワークに関係した数値が記載されています。その結果として、表 15 に示した精度よりも高い精度が表 16 には示されています。.ただし、これは推論時間と、ネットワークが必要とするメモリ空間(`activation` バッファと重みバイアス・テーブルが大きくなります)が犠牲になっています。

次の条件における人数カウントアプリケーションの実行性能を表 17 に示します。

- IAR Systems EWARM v8.50.6
- `-O3` オプション
- Cortex®-M7 コア・クロック周波数を 400MHz に設定
- B-CAMS-OMV カメラ・モジュール・バンドル

表 18. 人数カウント・アプリケーションの実行性能

C モデル	カメラ解像度	揮発性メモリ・レイアウト方式	重みバイアス・テーブルの位置	NN 推論時間(ms)	1 秒あたりの処理フレーム数(FPS)(1)	出力精度(%)(2)	バイナリ・ファイル(3)
YOLO 派生 240 x 240	QVGA	完全内部	内部 Flash メモリ	371	2.7	55.88	セクション 3.2.8.3 サポート対象アプリケーションの実行性能

1. FPS は、キャプチャ時間、前処理時間、推論時間を考慮して計算されています。
2. 50% IoU を使用して PASCAL VOC テスト・データベースに対して得られた平均精度。
3. IAR Embedded Workbench®設定 > バイナリファイル。

表 18 のシナリオに対応付けられている、`Binary` フォルダに格納されているバイナリ・ファイルのリストを以下に示します。

- R: `STM32H747I-DISCO_PeopleCounting > STM32H747I-DISCO_PeopleCounting_Quant8_IntFps_IntFlash_V300.bin`

3.2.9 メモリ·フットプリント

すべてのアプリケーションのメモリ要件を表 19 に示します。このテーブルの数字は、LCD ディスプレイに関連するコードとデータにあたるものでも、テスト·モードの実装に関連するコードとデータにあたるものでもありません。

表 19. アプリケーションごとのメモリ·フットプリント

アプリケーション C モデル	カメラ解像度	揮発性データ·メモリ設定(1)	Flash（バイト）		RW データ（バイト）	
			コード	RO データ(2)	内部 SRAM(3)	外部 SDRAM
食品分類 浮動小数点、標準	VGA	完全外部	約 60K	約 520K	約 20K	約 1.6 M
		スプリット			約 420K	約 1.2 M
食品分類 浮動小数点、最適化		完全外部		約 590K	約 20K	約 2.0 M
食品分類 量子化標準(4)	VGA	完全外部	約 80K	約 140K	約 20K	約 1.2 M
		スプリット			約 170K	
	QVGA	メモリ最適化された完全内部				0
		FPS 最適化された完全内部			約 320K	
食品分類 量子化、最適化(4)	VGA	完全外部		約 160K	約 20K	約 1.2 M
		スプリット			約 240K	
	QVGA	メモリ最適化された完全内部				0
		FPS 最適化された完全内部			約 390K	
人物検出 MobileNetV2	QVGA	FPS 最適化された完全内部		約 500K	約 390K	0
人物検出 Google				約 230K	約 320K	
人数カウント YOLO 派生 （240 x 240）				約 280K	約 430K	

1. C コード生成用の STM32Cube.AI ツールの中で allocate input in activation オプションが選択されています（食品分類浮動小数点モデルを除く）。
2. 重みバイアス·テーブルを含みます。
3. スタックとヒープ·サイズの要件を含みます。
4. 量子化ツール（STM32Cube.AI ツールによる提供）を使用して UaUa per layer 量子化スキームで量子化された食品分類 CNN モデル

外部 SDRAM には、次の 2 つの要素も含まれています。

- LCD 上の表示を管理するための読出し書込みバッファ：SDRAM アクセスを最適化するために（840 x 480 x 4 = 1.575MB のサイズの）バッファはそれぞれ別々の SDRAM バンクに配置されるため、合計サイズは 16MB となります。
- テスト·モードの実装に使用されるバッファ：合計サイズ 5.4MB

浮動小数点モデルと量子化モデルではコード·サイズが異なります。これは、両方のモデルでランタイム·ライブラリの同一関数セットを使用していないためです。

3.2.10 USB ウェブカメラ·アプリケーション

USB ウェブカメラ·アプリケーションにより、(B-CAMS-OMV カメラ·モジュール·バンドルまたは STM32F4DIS-CAM カメラドーター·ボードに接続されている)STM32H747I-DISCO ボードが USB ビデオ·カメラ·デバイスとして機能するようになります。USB ビデオ·デバイスは、次の目的に使用できます。

- 画像データセットの作成
- ビデオ·データセットの作成
- あるモデルがターゲットの STM32 デバイスに導入された場合と同一の画像取得デバイスを使用した、ホストマシン(x86)上での機械学習評価の実行

ファームウェアは 2 つのフォーマットをサポートしており、それぞれ 2 種類のフレーム解像度があります。

- MJPEG または YUY2 フォーマット
- VGA 640 x 480 または QVGA 320 x 240 解像度

MJPEG フォーマットでは、カメラは RGB565 でフレームをキャプチャするように設定されます。RGB565 フレームは、圧縮のために STM32H7 ハードウェア JPEG エンコーダへと供給されます。

YUY2 フォーマットでは、カメラは YUV 4:2:2 でフレームをキャプチャするように設定されます。STM32H7 による圧縮は行われません。

カメラを使用するために必要となる追加のドライバはありません。このアプリケーションには、ビデオ·デバイス用 USB デバイス·クラス定義(リビジョン 1.1、2005 年 6 月 1 日)が実装されています。ネイティブ USB ビデオ·ドライバ·サポートは、Windows®、Ubuntu®、macOS®で提供されています。

注 macOS® は、米国内およびその他の国と地域で登録された、Apple 社の商標です。
その他の商標は、それぞれの所有者に帰属します。

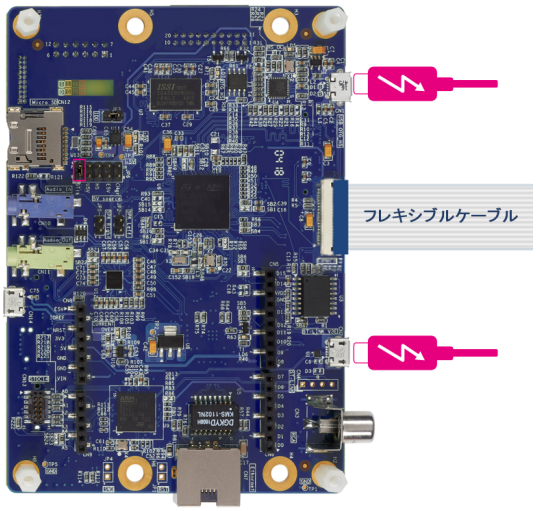
ビデオ·ドライバには、仕様の以下の部分が実装されています。

- デバイス·ディスクリプタ管理
- 設定ディスクリプタ管理
- インタフェース·アソシエーション·ディスクリプタ
  - 標準 VC インタフェース·ディスクリプタ = インタフェース 0
  - 標準 VS インタフェース·ディスクリプタ = インタフェース 1
- ビデオ·ストリーミング·インタフェース x 1
- ビデオ·ストリーミング·エンドポイント x 1
- ビデオ端子入力(カメラ) x 1
- ビデオクラス固有 AC インタフェース
- ビデオクラス固有 AS インタフェース
- ビデオ制御リクエスト
- ビデオ同期タイプ:非同期

3.2.10.1 USB ウェブカメラの使用方法

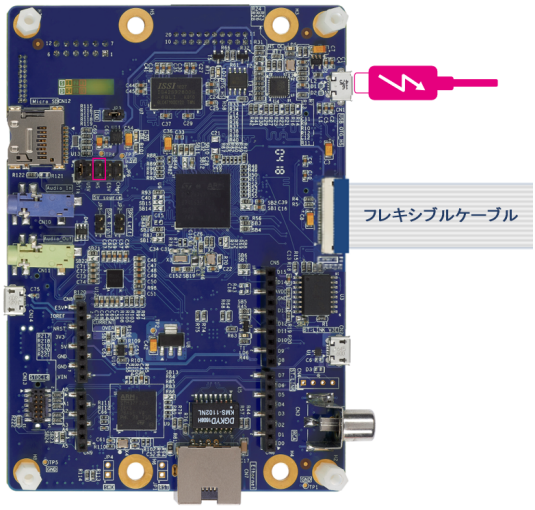
1. STM32H747I-DISCO デバイスにバイナリをロードします。
2. USB_OTG_HS コネクタに USB ケーブルを接続します。標準 ST-LINK USB 経由、または JP6 ジャンパを HS の位置に変更することにより、ボードに電源を供給できます。

図 14. STM32H747I-DISCO 電源（CN2）



JP6 が STLk 位置 – STM32H747I-DISCO は CN2 Micro-B USB レセプタクルから電源供給を受けます。

図 15. STM32H747I-DISCO 電源（CN1）



JP6 が HS 位置 – STM32H747I-DISCO は CN1 Micro-AB USB レセプタクルから電源供給を受けます。

接続されると、デバイスはエニュメレーションします。ボード LCD 画面には何も表示されません。

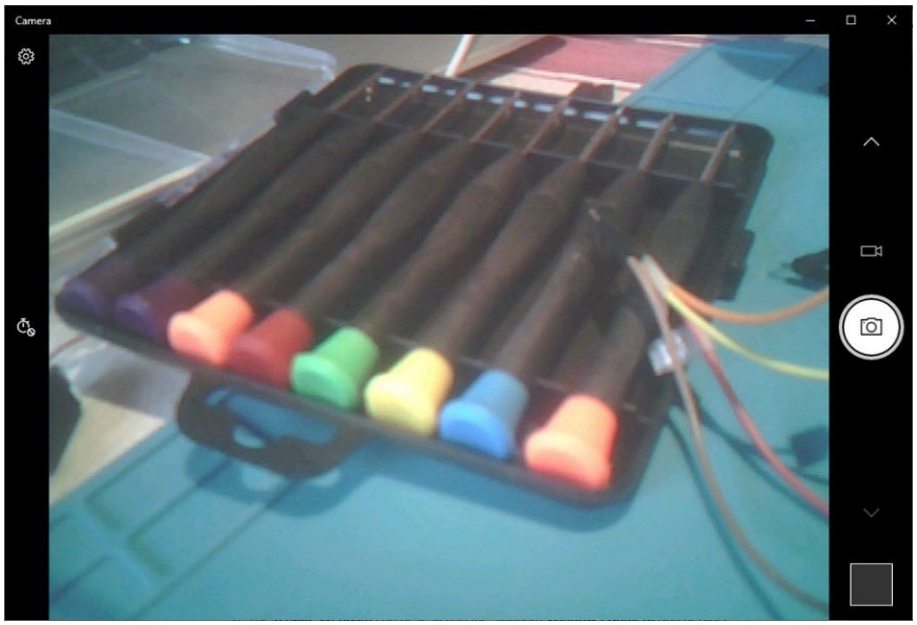
3. STM32 ウェブカメラは、標準 Windows Camera アプリや、FFmpeg のように制御オプションが豊富なその他のアプリケーションでテスト可能です（以下の FFmpeg の使用方法セクション参照）。

注 ホスト·ソフトウェアによっては、デフォルト設定とは異なるフォーマットと解像度をドライバが選択することがあります。

Windows Camera アプリの使用方法

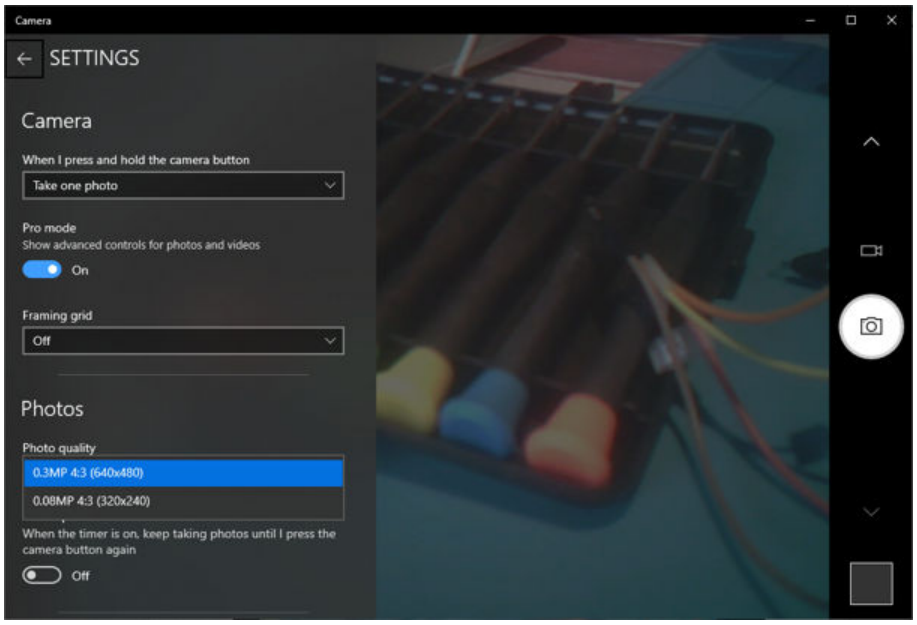
Microsoft®から提供されている Windows Camera アプリを使用して、STM32 ウェブカメラから画像とビデオをキャプチャできます。画像は、MJPEG ストリームを使用してキャプチャされます。写真を撮影するには、カメラ·アイコンをクリックするか、スペース·バーを押します。

図 16. Windows Camera アプリ



カメラ設定を使用すると、カメラ解像度を 640 x 480（VGA）から 320 x 240（QVGA）へと変更できます。

図 17. Windows Camera 設定 - カメラ解像度



場合によっては、解像度を変更すると画像がトリミングされることがあります。この問題を解決するには、Windows Camera アプリを閉じてから再起動して、最後に使用した設定を復元します。

FFmpeg の使用方法

FFmpeg は、ビデオ・ストリームのキャプチャに使用可能なコマンドライン・ユーティリティです。Windows Camera アプリよりも多くのオプションが提供されます。特に、非圧縮の RAW YUY2 ストリームの表示に使用できます。

FFmpeg は、ffmpeg.org からダウンロードしてインストールできます。.

FFmpeg はコマンドライン・ツールです。グラフィカル・ユーザ・インタフェースの代替として、ContaCam (www.contaware.com)ソフトウェアを使用できます。

最初に、STM32 Webcam デバイスで使用可能なデバイスと対応オプションを一覧表示します。

```
ffmpeg -f dshow -list_devices true -i dummy                ## List devices
ffmpeg -f dshow -list_options true -i video="STM32 Webcam" ## list options
```

次に、次のいずれかのコマンドを使用して、STM32 Webcam のライブ・ストリームを表示します。

```
ffplay -f dshow -i video="STM32 Webcam"
ffplay -f dshow -video_size 640x480 -vcodec mjpeg -i video="STM32 Webcam"
ffplay -f dshow -video_size 320x240 -vcodec mjpeg -i video="STM32 Webcam"
ffplay -f dshow -video_size 640x480 -pixel_format yuyv422 -i video="STM32 Webcam"
ffplay -f dshow -video_size 320x240 -pixel_format yuyv422 -i video="STM32 Webcam"
```

追加情報については、FFmpeg wiki (trac.ffmpeg.org/wiki/Capture/Webcam)を参照してください。

その他の使用方法例

その他の使用方法の例については、STM32 MCU wiki (wiki.st.com/stm32mcu)を参照してください。

3.2.10.2 USB ウェブカメラ・アプリケーションの実行性能

DCMI キャプチャは連続モードで動作します。キャプチャされたフレームごとに、USB 割り込みによる非同期オフローディングのために、画像が USB 転送バッファにコピーされます。USB は、パケット・サイズが 1024 バイト(USB ハイスピード)または 1023 バイト(USB フルスピード)のアイソクロナス・モードでバッファをオフロードします。

MJPEG フォーマットが選択されると、JPEG エンコードと USB 転送が DCMI キャプチャにパイプラインされます。JPEG エンコーディングはソフトウェア・エンコーディングとハードウェア・エンコーディングに依存するため、処理時間はコンパイラと最適化設定によって異なります。

表 20. USB ウェブカメラ・アプリケーションのタイミング(CAMERA_BOOST_PCLK = 0 / 1 の場合)

性能	MJPEG VGA	MJPEG QVGA	非圧縮 VGA	非圧縮 QVGA
DCMI キャプチャ(1)	133.3 / 66.7 ms	133.3 / 33.3 ms	133.3 / 83.3 ms	133.3 / 33.3 ms
JPEG エンコード	約 46～47 ms	約 11～12 ms	N/A	N/A
USB 転送	約 6～9 ms	約 2～3 ms	75.3 ms	18.9 ms
フレームレート(1)	7.5 / 15 fps	7.5 / 30 fps	7.5 / 12 fps	7.5 / 30 fps
ビット・レート(1)	約 3.3 / 9.2 Mbps	約 831 / 3.9 kbps	37 / 58.9 Mbps	9.2 / 37 Mbps

1. 最初の値は CAMERA_BOOST_PCLK = 0 の場合、2 番目の値は CAMERA_BOOST_PCLK = 1 の場合。CAMERA_BOOST_PCLK フラグはプロジェクト・コンパイラ・フラグで設定可能です(詳細はセクション 3.2.6 を参照してください)。

注

- MJPEG の場合、USB 転送時間はフレーム・サイズによって異なることがあります。
- 非圧縮 VGA モードは、USB 転送時間による制限を受けます。
- MJPEG モードは、JPEG エンコード時間による制限を受けます。

設定:

- すべてのバッファは外部 SDRAM です。
- B-CAMS-OMV カメラ・モジュール・バンドルに MB1379STMicroelectronics カメラ・モジュール(OV5640)がバンドルされています。
- JPEG 品質 = 90%。JPEG_QUALITY マクロを使用してファームウェアで調整できます。
- 1024 バイト・パケット・サイズの USB ハイスピード。マイクロフレーム(125 µs)ごとにパケットが 1 つ送信されます。マイクロフレームあたり最大 3 つのアイソクロナス転送を指定可能ですが、HAL はマイクロフレームごとに 1 つのトランザクションしかサポートしていません。

3.2.10.3 USB ウェブカメラ·アプリケーションの既知の問題と制限事項

- `CAMERA_BOOST_PCLK = 1` ではない場合の、OV5640 カメラ使用時の不正なフレーム·レート。すべての設定において、OV5640 カメラはデフォルトで 7.5 fps に設定されます。不正なフレーム·レートの報告は、ビデオの録画速度や再生速度の問題の原因となることがあります。teachablemachine.withgoogle.com との互換性維持のため、報告されたフレーム·レートは変更されていないことに注意してください。
- ハイスピード·モードで動作している場合には、カメラからデフォルトで供給される非圧縮 VGA フレーム·レート（15 fps）は、USB 転送速度と整合しません。この場合には、カメラのフレーム·レートが低下します。非圧縮 15 fps フレーム·レートを得るためには、マイクロフレームごとに追加のトランザクションが必要となりますが、この機能は現在 HAL でサポートされていません。
- フルスピード·モードで動作している場合には、性能が低下します。
  - フレーム·レートの低下
  - 一部にティアリング現象が観察されます

3.2.11 視覚化モード

3.2.11.1 食品分類アプリケーション

メイン·アプリケーションには、以下の 2 つの視覚化モードが用意されています。

- 食品のカテゴリがロゴで表現されているロゴ·ビュー
- カメラ·フィードを表示する画像ビュー

デフォルトでは、アプリケーションはロゴ·ビューが有効なった状態で起動します。モードを切り替えるには、実行中に[WakeUp]ボタンを押します。

注 [WakeUp]ボタンを押したときには、`Entering/Exiting CAMERA PREVIEW mode, Please release button` というメッセージが画面に表示されるまで待ってから、ボタンを離す必要があります。いずれの視覚化モードでも、推論時間（ms）、1 秒あたりの処理フレーム数（FPS）、トップ 1 出力クラス確率（%）についてもレポートされます。

3.2.11.2 人検出アプリケーション

人検出アプリケーションでは、カメラ画像表示モードのみ使用可能です。キャプチャされたカメラの内容が、推論時間（ms）、1 秒あたりの処理フレーム数（FPS）、トップ 1 出力クラス確率（%）とともに表示されます。人検出アプリケーションには、以下の 4 つのカメラ方向モードが用意されています。

- ノーマル
- 反転画像（デフォルト）
- 鏡像
- 反転鏡像

青色の[WakeUp]ボタンを押すと、カメラ·センサーは上のリストにあるさまざまなカメラ方向モードを繰り返します。この機能を使用すると、ボードの LCD ディスプレイに対してカメラを様々な方向に配置できます。

3.2.11.3 人数カウント検出アプリケーション

人数カウント·アプリケーションでは、カメラ画像表示モードのみ使用可能です。キャプチャされたカメラの内容が、1 秒あたりの処理フレーム数（FPS）とともに表示されます。シーンの中に人が検出されない場合は、VACANT ロゴが表示されます。シーンの中に 1 人以上の人が検出された場合は、検出された人数が表示されます。

人数カウント·アプリケーションには、以下の 4 つのカメラ方向モードが用意されています。

- ノーマル
- 反転画像（デフォルト）
- 鏡像
- 反転鏡像

青色の[WakeUp]ボタンを押すと、カメラ·センサーは上のリストにあるさまざまなカメラ方向モードを繰り返します。この機能を使用すると、ボードの LCD ディスプレイに対してカメラを様々な方向に配置できます。

3.2.12 組み込みの検証、キャプチャ、テスト

FP-AI-VISION1 機能パックには、特定の動作モードで検証、キャプチャ、テストを実行する可能性が組み込まれています。

- フレーム·キャプチャ·モード：カメラから microSD™カードにフレームをキャプチャします。
- メモリ·ダンプ·モード：デバッグのために処理パイプラインのそれぞれのステップを microSD™カードにダンプします。
- オンボード検証モード：microSD™カードに格納されている画像を用いてニューラルネットワークを評価します。

このセクションでは、これら 3 つのモードそれぞれについてと、それぞれの開始方法について説明します。

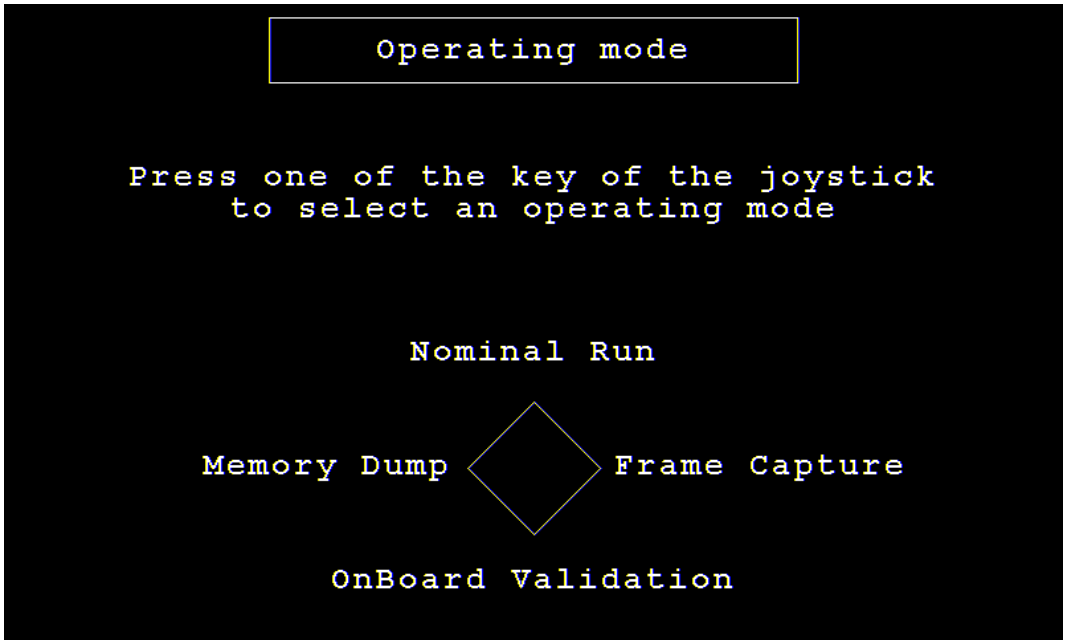
前提条件

これら 3 つのモードでは、STM32H747I-DISCO ディスカバリ·ボードに挿入する microSD™カードが必要です。読出し / 書込み操作が低速になるのを避けるため、少なくともクラス 10 の microSD™カードの使用を推奨します。さらに、カードは FatFS フォーマットされていて、ただ 1 つのパーティションを含んでいる必要があります。

検証モード、キャプチャ·モード、テスト·モードの有効化

（リセット後の）ウェルカム画面の間、[WakeUp]ボタンを押し続けて、メイン·アプリケーションの代わりに、図 18 に示されているテスト·メニューを有効にします。

図 18. 検証 / キャプチャ / テスト·メニュー

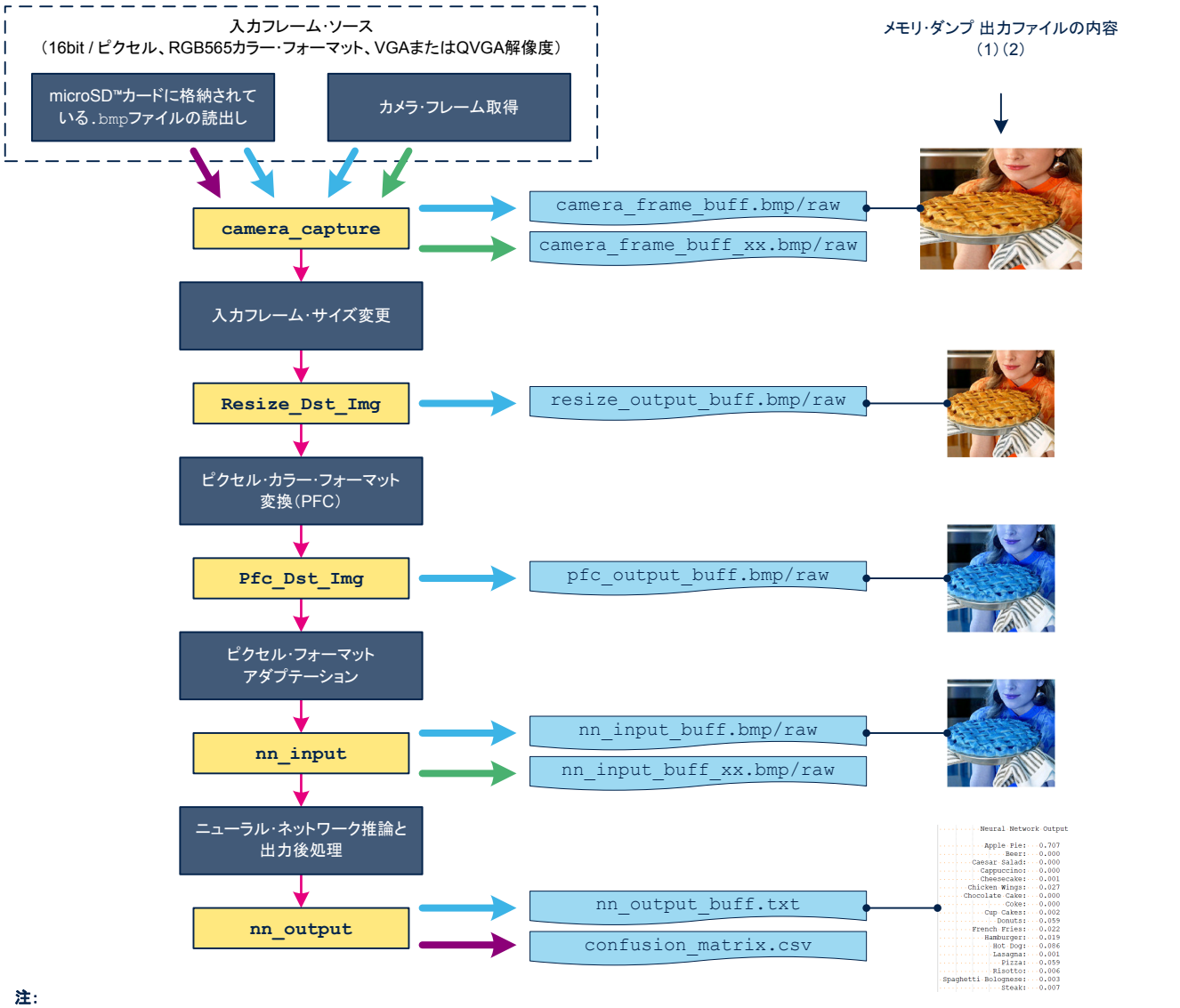


ジョイスティックの対応する矢印キーを押して、希望するアプリケーションを起動します。

- [UP]：デフォルト·アプリケーション（カメラからの画像を使用したニューラルネットワーク推論）が開始されます。
- [RIGHT]：フレーム·キャプチャ·モードが開始されます。
- [LEFT]：メモリ·ダンプ·モードが開始されます。
- [DOWN]：オンボード検証モードが開始されます。

オンボード検証モード、フレーム･キャプチャ･モード、テスト･モードの概要

図 19. 検証モード、キャプチャ･モード、テスト･モードの概要



注:

（1）この例は、食品分類（量子化Cモデル）アプリケーションに基づいています。この特定の例では、各ピクセルの青と赤のコンポーネントをスワップするピクセル･カラー･フォーマット変換（PFC）が（アプリケーションによって）設定されています。PFC演算中にDMA2Dによって生成されるフレームにはBGRフォーマットで構成されるピクセルが含まれている一方で、食品分類NNモデルはRGBフォーマットのピクセルを含む画像でトレーニングされるため（セクション3.2.6参照）、この特定のケースにはこのスワップが必要となります。

（2）画像はFood-101 データ･セットから抽出されています。

凡例:

フレーム·キャプチャ·モード

フレーム·キャプチャ·アプリケーションの主な目的は、データ収集を有効にすることです。アプリケーションのデフォルトでは、フレームは以下の 2 段階でキャプチャされます。

- カメラ取得の直後（RGB565 フォーマット）

および

- すべての前処理ステージの後、ニューラルネットワークの入力に供給される直前

ただし、アプリケーション·コードを変更することにより、実行チェーンの任意の段階でフレームをキャプチャできます。そうするには、実行チェーンの各ステージの前に初期化される `TestRunContext` 構造体の `PerformCapture` フィールドを 1 にセットします。

フレームがキャプチャされるファイルのフォーマットは、次のいずれかのフォーマットに設定できます。

- 16bit ピクセル·フォーマット（RGB565）でデータがエンコードされた `.bmp` ファイル
- `.raw` バイナリ·ファイル·フォーマット

これは、フレーム·キャプチャ·モードの初期化中に、`CaptureContext` 構造体の `capture_file_format` フィールドで設定可能となっています。デフォルト·フォーマットは `.bmp` です。

最終的に、キャプチャされたフレームは microSD™カードに格納されます。

フレーム·キャプチャ·モードの場合は、バッファの内容を出力ファイルに書き込む前に、ピクセルの赤コンポーネントと青コンポーネントをスワップすることも可能です。これは、`TestRunContext` 構造体の `rb_swap` フィールドを 1 にセットすることで行われます。

アプリケーションの起動時に、`Camera_Capture` フォルダが存在しない場合には作成され、セッション名が生成されて画面の左上コーナーに表示されます。`CAM_CAPTURE_SESS_<session name>`という名前のサブフォルダも作成されます。画面にはカメラから取得した画像が表示されます。右上コーナーには `READY` メッセージが表示され、プログラムに microSD™カードに書き込む準備ができたことを示します。

フレーム·キャプチャをトリガするには、[WakeUp]ボタンを押してください。`READY` メッセージが `BUSY` メッセージに切り換わって、フレームを microSD™カードに書込み中であることを示し、その間はそれ以外のキャプチャはすべてブロックされます。書込み操作が終了すると、`READY` メッセージが表示され、キャプチャが再び有効になります。

キャプチャするごとに、そのセッションに対応したサブフォルダに以下の 2 つのファイルが生成されます。

- `camera_frame_buff_xx.bmp` または `camera_frame_buff_xx.raw`: 前処理ステージの前に、カメラ取得直後にキャプチャされたフレームが格納されます。
- `nn_input_buff_xx.bmp` または `nn_input_buff_xx.raw`: 前処理ステージの後、ニューラルネットワークの入力の前に、キャプチャされたフレームが格納されます。

あるセッションについて、`xx` というファイル名の接尾辞は、インクリメント·キャプチャ番号に相当します。

メモリ·ダンプ·モード

メモリ·ダンプ·アプリケーションは、主としてデバッグを目的としています。この動作モードに入ったときに、`Memory_Dump` フォルダが存在しない場合には作成されます。

画面には `camera_frame` バッファに格納されている画像が表示されます。右上コーナーには `READY` メッセージが表示され、プログラムにメモリ·ダンプ·セッションを開始する準備ができたことを示します。

メモリ·ダンプ·セッションをトリガするには、[WakeUp]ボタンを押してください。`READY` メッセージが `BUSY` メッセージに切り換わって、メモリ·ダンプ操作が実行中で microSD™カードへの書込みが行われていることを示し、その間はそれ以外のキャプチャはすべてブロックされます。フレーム·キャプチャ ·モードと同様に、セッション名が生成されて画面の左上コーナーに表示されます。`DUMP_SESS_<session name>`という名前のサブフォルダも作成されます。メモリ·ダンプ操作が完了すると、`READY` メッセージが表示され、メモリ·ダンプが再び有効になります。

メモリ·ダンプ·セッションの間は、アプリケーションの各実行ステージ（セクション 3.2.5.1 アプリケーション実行フローと揮発性（RAM）データ·メモリ要件参照）の後に、出力データを含むメモリ部分のダンプが行われます。メモリの内容は、microSD™カードに格納されているファイルにダンプされます。これらのファイルは、`.bmp`、`.raw`、`.txt` のいずれかのフォーマットとなります。

メモリ·ダンプ·セッションごとに、そのセッションに対応したサブフォルダに以下の 5 つのファイルが生成されます。

- camera_frame_buff.bmp または camera_frame_buff.raw:取得後の camera_frame バッファの内容。
- resize_output_buff.bmp または resize_output_buff.raw:サイズ変更操作後の Resize_Dst_Img バッファの内容。
- pfc_output_buff.bmp または pfc_output_buff.raw:ピクセル·カラー·フォーマット変換演算後の Pfc_Dst_Img バッファの内容。
- nn_input_buff.bmp または nn_input_buff.raw:ピクセル·フォーマット·アダプテーション演算後の nn_input バッファの内容。
- nn_output_buff.txt:ニューラルネットワーク推論後の nn_out バッファの内容

メモリ·ダンプ·モードには、入力イメージのソースに応じて以下の 3 つのサブモードがあります。

- SD カード:入力画像は、microSD™カードの dump_src_image_xx という名前のフォルダに格納されている.bmp ファイルから入力されます。ここで、xx は画像の解像度を示します(可能な値は vga と qvga)。
- カメラ·ライブ:入力画像は、カメラ取得から入力されます(公称動作モードと同様)。
- テスト·カラー·バー:入力画像は、テスト·カラー·バー·モードに設定されているカメラ取得から入力されます

メモリの内容がダンプされるファイルのフォーマットは、次のいずれかのフォーマットに設定できます。

- ピクセルあたり 8bit でデータがエンコードされた.bmp ファイル·フォーマット
- ピクセルあたり 16bit(RGB565)でデータがエンコードされた.bmp ファイル·フォーマット
- ピクセルあたり 24bit(RGB888)でデータがエンコードされた.bmp ファイル·フォーマット
- .raw バイナリ·ファイル·フォーマット
- .txt ファイル·フォーマット

上の図 19 は、食品分類量子化モデルを QVGA 解像度で使用した場合に得られるメモリ·ダンプ出力ファイルの例を示しています。ピクセル·カラー·フォーマット変換ステージ後に得られる出力ファイルでは、赤と青のピクセル·コンポーネントがスワップされています。スワップする理由は、食品分類 NN モデルでは、赤コンポーネントがアドレス·オフセット 0 の位置にある入力ピクセルのメモリ·レイアウトが想定されているためです。ただし、(ピクセル·フォーマット変換演算中の)DMA2D は、青コンポーネントがアドレス·オフセット 0 の位置にあるピクセルのフレームを供給します。正しい入力データが NN モデルに供給されるように、赤と青のピクセル·コンポーネントをスワップする必要があります(セクション 3.2.7 参照)。

オンボード検証モード

オンボード検証アプリケーションは、ニューラルネットワークを評価するための入力として、microSD™カードに格納されている画像を使用します。

重要

ここで説明されているオンボード検証プロセスと、X-CUBE-AI ツールに統合されている検証プロセスとを区別することが重要です。オンボード検証プロセスが想定している入力データは、組み込みニューラルネットワークモデルに供給される前に特定の前処理演算を経ることを意味する RAW データ(.bmp ファイル)です。オンボード検証プロセスで所得された結果を、X-CUBE-AI ツールに統合されている検証プロセスで取得された結果と比較することをユーザがいとわない状況では、オンボード検証プロセスによって使用された入力データは、X-CUBE-AI 環境のモデルの検証を実行するための入力などとして使用することはできません。詳細については、X-CUBE-AI 組み込みマニュアルの Evaluation report and metrics/Specific attention on the provided data のセクションを参照してください。

プログラムが microSD™カード内の画像を見つけるには、ファイル·システムのルートに onboard_valid_dataset_xx という名前のディレクトリが存在する必要があります。ここで、xx は画像の解像度を示します(可能な値は vga と qvga)。このディレクトリには、コードの中で NN_OUTPUT_CLASS_LIST によって定義されているものと同じ名前の画像が格納されている、1 つのディレクトリがクラスごとに存在する必要があります。

機能パックにおいて、この変数には、ファイル fp_vision_app.c で定義されている output_labels というエイリアスが設定されます。次の例は、食品分類アプリケーションのためのものです。

```
#define NN_OUTPUT_CLASS_LIST output_labels
const char *output_labels[AI_NET_OUTPUT_SIZE] = {
"Apple Pie", "Beer", "Caesar Salad", "Cappuccino", "Cheesecake", "Chicken Wings",
"Chocolate Cake", "Coke", "Cup Cakes", "Donuts", "French Fries", "Hamburger", "Hot
Dog", "Lasagna", "Pizza", "Risotto", "Spaghetti Bolognese", "Steak"};
```

すべての画像は、VGA または QVGA 解像度の BMP フォーマット（16bit / ピクセル、RGB565 フォーマット）で格納する必要があります。このクラスは、画像が格納されているディレクトリから直接派生しています。ファイル名は考慮されません。

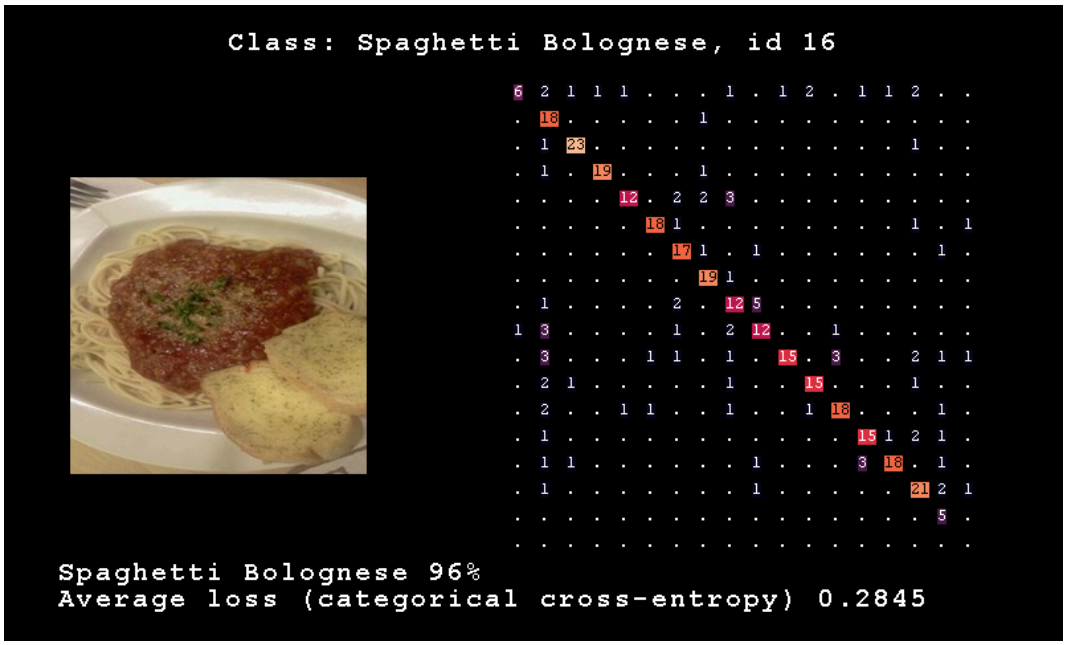
画像のデータセットを BMP フォーマットに変換するヘルパー・スクリプト（`create_dataset.py`）が、`Utilities\AI_resources\Food-Recognition` ディレクトリに提供されています。

食品分類の例では、microSD™ファイル・システムの構造は次のとおりである必要があります。

```
。
|-- onboard_valid_dataset_xx
    |-- Apple Pie
    |   |-- apple_pie_example1.bmp
    |   |-- another_apple_pie.bmp
    |   `-- ...
    |-- Beer
    |   |-- beer.bmp
    |   |-- another_beer.bmp
    |   `-- ...
    |-- Caesar Salad
    |   |-- a_caesar_salad.bmp
    |   `-- ...
    |-- Cappuccino
    |   |-- cappuccino.bmp
    |   `-- ...
    |-- Cheesecake
    |   |-- cheesecake.bmp
    |   `-- ...
    |-- Chicken Wings
    |   |-- chicken_wing.bmp
    |   `-- ...
    |-- Chocolate Cake
    |   |-- chocolate_cake.bmp
    |   `-- ...
    |-- Coke
    |   |-- coke.bmp
    |   `-- ...
    |-- Cup Cakes
    |   |-- cup_cakes.bmp
    |   `-- ...
    |-- Donuts
    |   |-- donuts.bmp
    |   `-- ...
    |-- French Fries
    |   |-- french_fries.bmp
    |   `-- ...
    |-- Hamburger
    |   |-- hamburger.bmp
    |   `-- ...
    |-- Hot Dog
    |   |-- hot_dog.bmp
    |   `-- ...
    |-- Lasagna
    |   |-- lasagna.bmp
    |   `-- ...
    |-- Pizza
    |   |-- pizza.bmp
    |   `-- ...
    |-- Risotto
    |   |-- risotto.bmp
    |   `-- ...
    |-- Spaghetti Bolognese
    |   |-- spaghetti_bolognese.bmp
    |   `-- ...
    `-- Steak
        |-- steak.bmp
        `-- ...
```

起動時に、オンボード検証アプリケーションは、このセクションで説明されている情報の概要を表示します。[WakeUp]ボタンを押して、検証を開始します。すると、図 20 に示す画面が表示されます。

図 20. 食品分類の例に対するオンボード検証情報の概要



ディスプレイの上部には、クラス名が `NN_OUTPUT_CLASS_LIST` のインデックスとともに表示されます。

左側には、ニューラルネットワークによって処理中の現在の画像が、ネットワーク入力サイズに合うようにサイズ変更されて表示されます。画像の下には、処理したすべての画像に対する NN Top-1 出力、信頼水準、平均多クラス交差エントロピー損失が表示されます。

ディスプレイの右側には、混同行列が表示され、ニューラル·ネットワークで画像が処理されている間に常に更新されます。行はグラウンド·トゥルース·ラベルを、列は予測クラスを示します。

すべての画像が処理されるとメッセージが表示され、[WakeUp]ボタンをクリックすると、表示が更新されて図 21 に示した分類レポートが表示されます。.

図 21. 食品分類の例に対するオンボード検証分類レポート

	precision	recall	f1-score	support
Apple Pie	0.857	0.316	0.462	19
Beer	0.500	0.947	0.655	19
Caesar Salad	0.885	0.920	0.902	25
Cappuccino	0.950	0.905	0.927	21
Cheesecake	0.857	0.632	0.727	19
Chicken Wings	0.900	0.857	0.878	21
Chocolate Cake	0.607	0.850	0.708	20
Coke	0.760	0.950	0.844	20
Cup Cakes	0.545	0.600	0.571	20
Donuts	0.571	0.600	0.585	20
French Fries	0.938	0.536	0.682	28
Hamburger	0.789	0.750	0.769	20
Hot Dog	0.818	0.720	0.766	25
Lasagna	0.750	0.750	0.750	20
Pizza	0.900	0.720	0.800	25
Risotto	0.656	0.808	0.724	26
Spaghetti Bolognese	0.731	0.905	0.809	21
Steak	0.857	0.692	0.766	26
accuracy			0.747	395
macro avg	0.771	0.748	0.740	395
weighted avg	0.778	0.747	0.744	395

分類レポートには、各クラスの精度、再現率、f1 スコアのほか、マクロ平均（クラスごとの非加重平均の平均）と加重平均（クラスごとのサポート加重平均の平均）が表示されます。

混同行列、誤分類されたファイルのリスト、分類レポートが、microSD™カードのファイル･システムのルートに以下の名前で保存されます。

- `confusion_matrix.csv`
- `missclassified.txt`
- `classification_report.txt`

3.2.13 出力表示

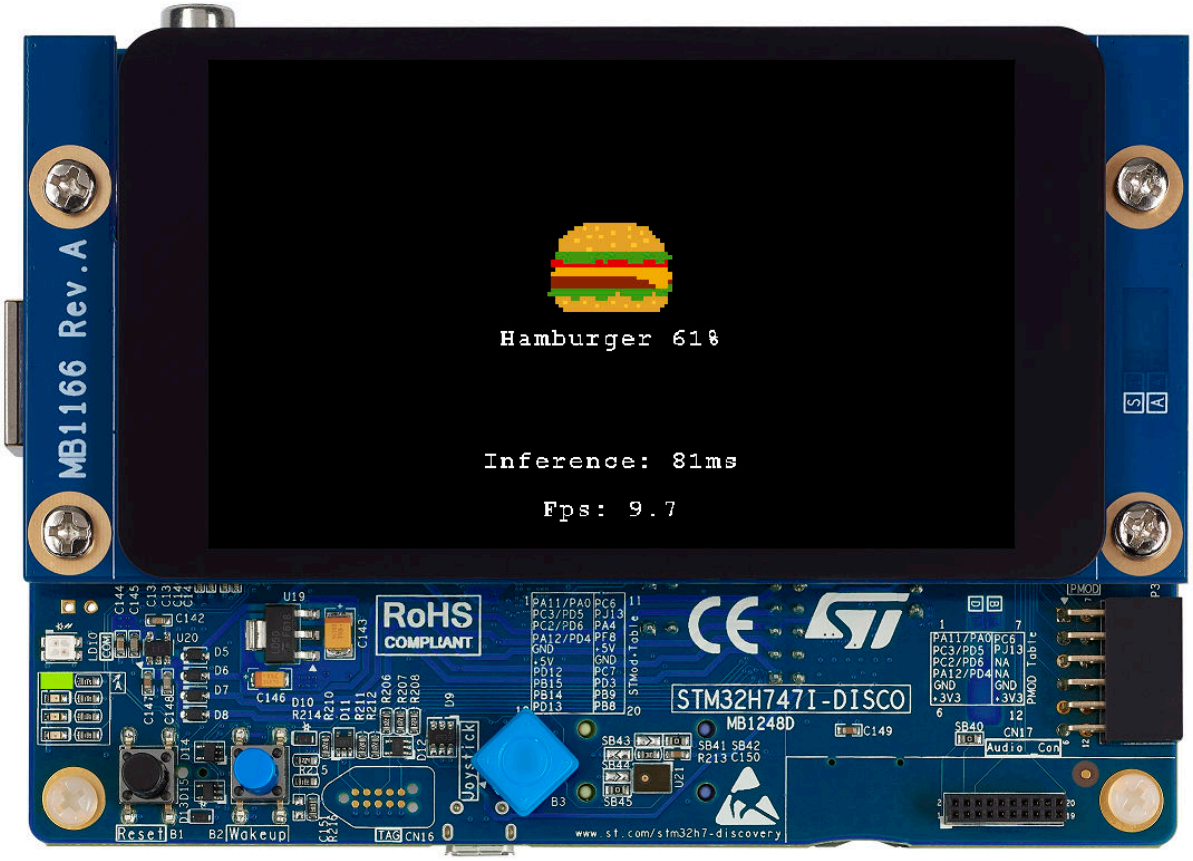
デフォルトでは、可能性が最も高い出力クラス（Top1）に関する情報（クラス名 + %単位の信頼水準）のみが、推論後に LCD に表示されます。

推論後には、CNN 推論時間（ミリ秒単位）と 1 秒あたりの処理フレーム数（FPS）も LCD に表示されます。

推論が終わるたびに、以下の 3 個の LED のいずれかが点灯します。

- 赤色 LED：出力信頼度が 55%未満の場合
- 橙色 LED：出力信頼度が 55%～70%の場合
- 緑色 LED：出力信頼度が 70%超の場合

図 22. 緑色 LED が点灯している食品分類の例



3.3 ハードウェアのセットアップ

FP-AI-VISION1 機能パックがサポートしているハードウェア構成は、B-CAMS-OMV カメラ･モジュール･バンドル（推奨）または STM32F4DIS-CAM カメラ･ドーターボード（レガシー製品のみ）に接続された STM32H747I-DISCO ディスカバリ･ボードです。

STM32H747I-DISCO ディスカバリ･ボードの特徴：

- STM32H747XIH6 マイクロコントローラ（x1）
  - SRAM：
    - システム SRAM 864KB（512KB + 288KB + 64KB）
    - DTCM RAM 128KB
  - Flash メモリ：
    - 2MB（1MB x 2 バンク）
- 8bit カメラ･コネクタ（x1）
- 256Mbit 外部 SDRAM（x1）

B-CAMS-OMV カメラ･モジュール･バンドルの特徴：

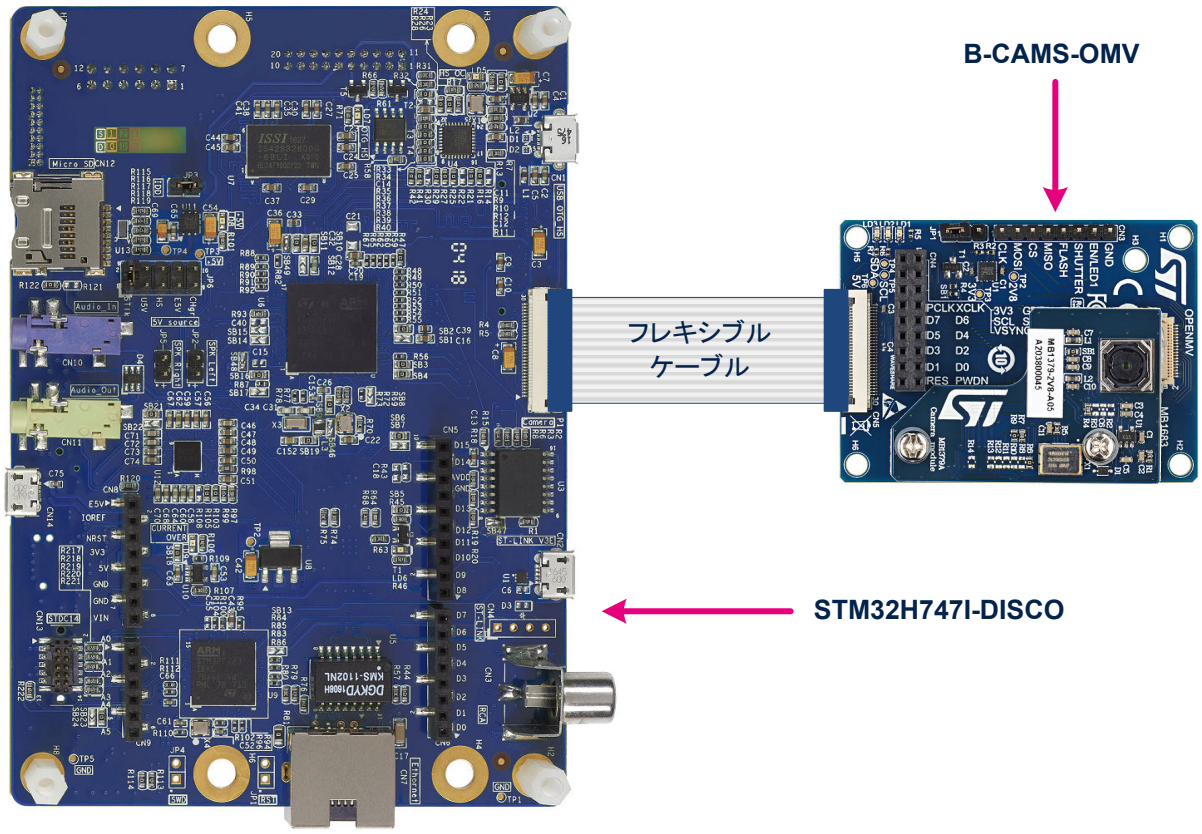
- OmniVision® OV5640 カメラ･センサ（x1）

STM32F4DIS-CAM カメラ･ドーターボードの特徴：

- OmniVision® OV9655 カメラ･モジュール（x1）

図 23 に示されているように、B-CAMS-OMV カメラ·モジュール·バンドル（または STM32F4DIS-CAM カメラ·ドータボード）は、カメラ·モジュール·バンドル（またはカメラ·ドータボード）に付属のフレキシブル·ケーブルを介して STM32H747I-DISCO ディスカバリ·ボードに接続されます。

図 23. STM32H747I-DISCO と B-CAMS-OMV のハードウェアのセットアップ



4 CNN のテスト

4.1 テスト条件

標準的なテスト環境では、カメラの前にタブレット・デバイスを配置します。その場合、次のパラメータが認識精度に影響することに注意してください。

- ネオン・ランプのちらつきなど、周囲の光と照明
- タブレットに表示される入力画像の品質。高解像度が必要。
- カメラとタブレットの距離
- 反射（タブレットが入力画像ソースとして使用される場合など）

ジョイスティックの[LEFT]ボタンと[RIGHT]ボタンを押して、カメラのコントラストを調整すると、認識精度が改善されます。

4.2 カメラと LCD の設定の調整

LCD の明るさは、ジョイスティック（[UP]と[DOWN]）を使用して設定します。

カメラのコントラスト・レベルは、ジョイスティック（[LEFT]と[RIGHT]）を使用して調整します。

LCD の明るさとカメラのコントラスト・レベルは、ジョイスティック（[SEL]）を使用してデフォルト値に設定します。

改版履歴

表 21. 文書改版履歴

日付	版	変更内容
2019 年 7 月 17 日	1	初版発行
2019 年 12 月 24 日	2	以前のセクションテスト·モードをセクション 3.2.8 組み込みの検証とテストに差し替え。 以前のセクション 2 つの取得モードをセクション 3.2.7 2 つの視覚化モード.に差し替え。 セクション 3.2.3 トレーニング·スクリプトを追加。 ドキュメント全体でフォルダ·ツリーを更新。 STM32Cube.AI の allocate input in activation オプションによって提供される利用可能なバッファ最適化に関して、ドキュメント全体を更新。 セクション 3.2.5 実行パフォーマンスと全体のメモリ·フットプリントの測定条件と結果を更新。
2020 年 9 月 10 日	3	ドキュメント全体を更新: • 人検出アプリケーションを追加 • データ·バッファとメモリ使用量を更新。 • 性能測定値とメモリ·フットプリントを更新。 • フォルダの構成と内容を更新。 • シナリオを更新。 • 検証モード、キャプチャ·モード、テスト・モードを更新。
2021 年 4 月 13 日	4	USB ウェブカメラ·アプリケーションと B-CAMS-OMV カメラ·モジュール·バンドルの使用方法を追加。 • FP-AI-VISION1 機能パックの機能概要、ソフトウェア·アーキテクチャ、フォルダ構成、食品分類アプリケーションと存在検出アプリケーションの実行性能、ハードウェアのセットアップを更新。 • カメラ·ピクセル·クロック、ピクセル·データの順序、USB ウェブカメラ·アプリケーションを追加。
2021 年 8 月 30 日	5	人数カウントアプリケーションを追加。 • FP-AI-VISION1 機能パックの機能概要、ソフトウェア·アーキテクチャ、生成コードの統合、フォルダ構成、カメラ·ピクセル·クロック、アプリケーションの性能を更新。 • 人数カウント·アプリケーション、表 5. 人数カウントアプリケーション用 SRAM メモリ·バッファ、その他のアプリケーション、人数カウント検出アプリケーションを追加。 STM32H7 上の CNN ベースのコンピュータ·ビジョン·アプリケーションの構築からデュアルコア MCU に関する注記を削除。
2021 年 12 月 17 日	6	STM32_Image ライブラリを STM32_ImageProcessing_Library に差し替え、STM32_Fs への参照を削除。 • FP-AI-VISION1 機能パック機能の概要、ソフトウェア·アーキテクチャ、およびフォルダ構成 を更新 • STM32 画像処理ライブラリを追加 人数カウント·アプリケーションを追加。

目次

表一覧

図一覧

重要なお知らせ（よくお読み下さい）

STMicroelectronics NV およびその子会社（以下、ST）は、ST 製品及び本書の内容をいつでも予告なく変更、修正、改善、改定及び改良する権利を留保します。購入される方は、発注前に ST 製品に関する最新の関連情報を必ず入手してください。ST 製品は、注文請書発行時点で有効な ST の販売条件に従って販売されます。

ST 製品の選択並びに使用については購入される方が全ての責任を負うものとします。購入される方の製品上の操作や設計に関して ST は一切の責任を負いません。

明示又は黙示を問わず、ST は本書においていかなる知的財産権の実施権も許諾致しません。

本書で説明されている情報とは異なる条件で ST 製品が再販された場合、その製品について ST が与えたいかなる保証も無効となります。

ST および ST ロゴは STMicroelectronics の商標です。ST の登録商標については ST ウェブサイトをご覧ください。www.st.com/trademarks.その他の製品またはサービスの名称は、それぞれの所有者に帰属します。

本書の情報は本書の以前のバージョンで提供された全ての情報に優先し、これに代わるものです。