SONY

Vision and Sensing Application SDK モデル量子化 機能仕様書

Version 0.2.0

2023 - 1 - 30

目次

1. 更新履歴 1
2. 用語・略語 2
3. 参照資料 3
4. 想定ユースケース 4
5. 機能概要、アルゴリズム 5
6. 操作性仕様、画面仕様 8
7. 目標性能 15
8. 制限事項 16
9. その他特記事項 17
10. 未決定事項 18

1. 更新履歴

Date	What/Why
2022/11/16	初版作成
2023/01/30	量子化のキャリブレーションに使用する画像、評価時に使用する画像の格納方法について修正。 評価時に使用するlabel情報ファイルの格納方法について追加。 evaluate_ground_truth_file設定を削除しevaluate_label_file設定を追加。 dataset_image_dir, evaluate_image_dir設定のデフォルトのパスを修正。 サポートする画像フォーマットを記載。 フォルダ構成変更。 シンボリックリンク対応の追加。 シークレット情報の削除に伴い、設定ファイルのInitial項目を削除、設定ファイルを作成する旨を追記。 PDFビルド環境更新。

2. 用語・略語

Terms/Abbreviations	Meaning
MCT	モデルを量子化するためのオープンソースソフトウェア
Keras	AIモデルのフォーマットの一種
TFLite	TensorFlow Liteのこと AIモデルのフォーマットの一種
イテレーション	(1回あたりの)学習

3. 参照資料

- Reference/Related documents（関連資料）
  - Model Compression Toolkit (MCT)
    - https://github.com/sony/model_optimization

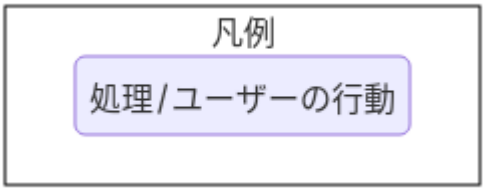
4. 想定ユースケース

- モデルの量子化を行いたい
  量子化を行うことでモデルのサイズを抑え、ターゲットエッジAIデバイスにデプロイできるようにしたい
- 量子化前と後のモデルを使用して推論実行し精度を確認したい

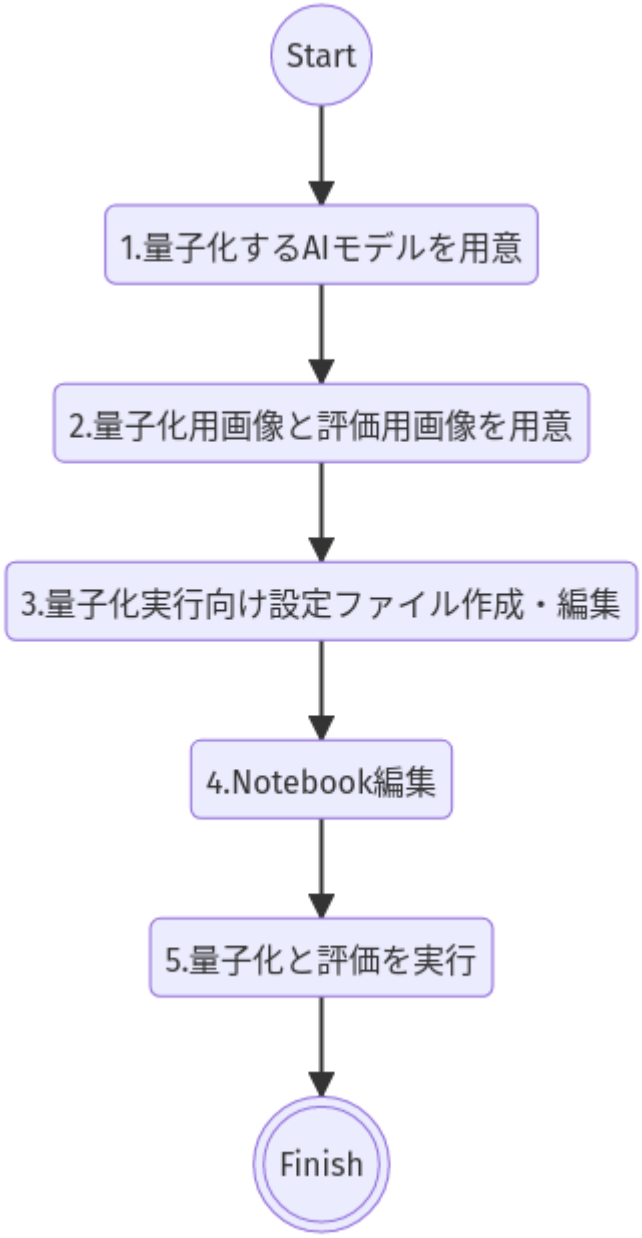
5. 機能概要、アルゴリズム

Functional Overview

- SDKにて下記のフローでImage ClassificationのAIモデル(Keras)を量子化しAIモデル(TFLite)に変換できる
- 量子化前と後のAIモデルで推論実行し、推論実行結果の統計値(Top1 accuracy)を取得できる
- SDKにてサポートするAIモデルは、MCTの supported-features に準拠する
- SDKにてサポートする画像フォーマットはJPEGとする


凡例
処理/ユーザーの行動

- フロー概要


Start
1.量子化するAIモデルを用意
2.量子化用画像と評価用画像を用意
3.量子化実行向け設定ファイル作成・編集
4.Notebook編集
5.量子化と評価を実行
Finish

- フロー詳細
    1. 量子化するAIモデルを用意
        - 変換対象となるAIモデル(Keras)を用意する
    2. 量子化用画像と評価用画像を用意
        - 量子化のキャリブレーションに使用するため、AIモデルのtrainingに使用した画像を用意する
        - 推論評価時に入力として使用するため、AIモデルのvalidationに使用する画像とそのlabel情報を用意する
    3. 量子化実行向け設定ファイル作成・編集
        - 設定ファイルconfiguration.jsonを作成、編集してNotebook実行時の設定を行う
    4. Notebook編集
        - 使用するAIモデルに応じてNotebook内のcalibration用preprocessing処理部の実装を修正する
    5. 量子化と評価を実行
        - AIモデル(Keras)を量子化しAIモデル(TFLite)に変換し、推論評価するNotebookを実行する

6. 操作性仕様、画面仕様

How to start each function

1. SDK環境を立ち上げ、Topの **`README.md`** をプレビュー表示する
2. SDK環境Topの **`README.md`** に含まれるハイパーリンクから、 **`tutorials`** ディレクトリの **`README.md`** にジャンプする
3. **`tutorials`** ディレクトリの **`README.md`** に含まれるハイパーリンクから、 **`3_prepare_model`** ディレクトリの **`README.md`** にジャンプする
4. **`3_prepare_model`** ディレクトリの **`README.md`** に含まれるハイパーリンクから、 **`develop_on_sdk`** ディレクトリの **`README.md`** にジャンプする
5. **`develop_on_sdk`** ディレクトリの **`README.md`** に含まれるハイパーリンクから、 **`2_quantize_model`** ディレクトリの **`README.md`** にジャンプする
6. **`2_quantize_model`** ディレクトリの **`README.md`** に含まれるハイパーリンクから、 **`image_classification`** ディレクトリの **`README.md`** にジャンプする
7. **`image_classification`** ディレクトリの各ファイルから各機能に遷移する

量子化するAIモデルを用意

1. 変換対象となるAIモデル(Keras)を用意する
    - 変換対象となるAIモデル(Keras)を、SDK実行環境に格納する

量子化用画像と評価用画像を用意

1. 量子化のキャリブレーションに使用するため、AIモデルのtrainingに使用した画像を用意する
    - AIモデルのtrainingに使用した画像(300ファイル程度)が含まれるフォルダを、SDK実行環境に格納する
        - tutorials/_common/datasetフォルダ内に格納する場合は、下記のように格納する

```
tutorials/
  └ _common
    └ dataset
      ├ training/  (1)
      │  ├ 画像の分類名/
      │  │   └ 画像ファイル
      │  ├ 画像の分類名/
      │  │   └ 画像ファイル
      │  ├ ・・・・
```

(1) 学習時に使用したデータセット。フォルダ構成は任意。

2. 推論評価時に入力として使用するため、 ImageNet 1.0形式のフォルダ構成 のアノテーションデータとそのlabel情報ファイルを用意する
    - AIモデルのvalidationに使用する画像が含まれるフォルダを、SDK実行環境に格納する
        - tutorials/_common/datasetフォルダ内に格納する場合は、下記のように格納する

```
tutorials/
  └ _common
    └ dataset
      ├ validation/ (1)
      │  ├ 画像の分類名/
      │  │   └ 画像ファイル
      │  ├ 画像の分類名/
      │  │   └ 画像ファイル
      │  ├ ・・・・
      └ labels.json (2)
```

(1) 評価時に使用するデータセット。フォルダ構成は上記のように ImageNet 1.0形式のフォルダ構成 にする。

(2) label情報ファイル

- label情報ファイルのフォーマットは下記のようにlabel名とそのid値が記載されたjsonファイルとする

```
{"daisy": 0, "dandelion": 1, "roses": 2, "sunflowers": 3, "tulips": 4}
```

ユーザー独自で用意したAIモデルをSDKで量子化する場合に、データセットを上記の形式に変換する方法は、 CVAT画像アノテーション 機能仕様書 の アノテーション情報をフォーマット変換 を参照。

量子化実行向け設定ファイル作成・編集

1. 実行ディレクトリに設定ファイル(**`configuration.json`**)を作成し、編集する

「実行ディレクトリ」について、image classificationを実行する場合は **`quantize_model/image_classification`** ディレクトリとなる。

特別な記載がある場合を除き、原則として省略は不可。

特別な記載がある場合を除き、原則として大文字小文字を区別する。

原則としてシンボリックリンクのフォルダパス、ファイルパスは使用不可。

Configuration	Meaning	Range	Remarks
source_keras_model	変換元となるAIモデル(Keras) パス。KerasのSavedModel形式のフォルダまたはh5形式のファイルを指定する	絶対パスまたはNotebook(*.ipynb)からの相対パス	
dataset_image_dir	量子化の際にキャリブレーションを行うためのデータセット画像を格納したディレクトリ	絶対パスまたはNotebook(*.ipynb)からの相対パス	
batch_size	量子化の際にキャリブレーションを行う画像を小分けにして重みやバイアスなどの特徴を見つけるセット枚数	1以上かつ、**dataset_image_dir**に含まれる画像枚数以下	
input_tensor_size	AIモデルの入力テンソルのサイズ(画像の一辺のピクセル数)	AIモデルの入力テンソルに準拠	
iteration_count	量子化時のイテレーション回数	1以上	
output_dir	変換結果AIモデルの出力先となるディレクトリ	絶対パスまたはNotebook(*.ipynb)からの相対パス	
evaluate_image_dir	推論実行時に入力する画像を含むディレクトリ	絶対パスまたはNotebook(*.ipynb)からの相対パス	
evaluate_image_extension	推論実行時に入力する画像の拡張子	文字列	

Configuration	Meaning	Range	Remarks
`evaluate_label_file`	AIモデルのラベル情報	絶対パスまたはNotebook(*.ipynb)からの相対パス	
`evaluate_result_dir`	推論実行結果の統計情報を保存するディレクトリ	絶対パスまたはNotebook(*.ipynb)からの相対パス	

Notebook編集

1. 実行ディレクトリの量子化実行用Notebook(*.ipynb)を開く
2. Notebookの中のcalibration用preprocessing処理部 (**`FolderImageLoader`** の引数 **`preprocessing=[resize, normalization]`**) を編集する
    - 使用するAIモデルの学習時のpreprocessing処理に相当する処理となるよう、編集する

量子化と評価を実行

1. 実行ディレクトリの量子化実行用Notebook(*.ipynb)を開き、その中のPythonスクリプトを実行する
    - その後下記の動作をする
        - 実行ディレクトリのconfiguration.json存在をチェックする
            - エラー発生時はその内容を表示し、中断する
        - configuration.json `source_keras_model` 、 `dataset_image_dir` の存在をチェックする
            - エラー発生時はその内容を表示し、中断する
        - configuration.json の下記の内容を読み取り、MCTへ必要な設定を行い、AIモデル(Keras)を量子化し変換する
            - configuration.json `source_keras_model`
            - configuration.json `dataset_image_dir`
            - configuration.json `batch_size`
            - configuration.json `input_tensor_size`
            - configuration.json `iteration_count`
        - MCTなどの外製ソフトでエラー発生時は、外製ソフトが出力するエラーを表示し、中断する
        - configuration.json `output_dir` に、MCTで量子化したAIモデル(TFLite)ファイル `model_quantized.tflite` と、TensorFlow標準機能でTFLiteに変換したAIモデル(TFLite)ファイル `model.tflite` を出力する
            - `output_dir` で指定するディレクトリがなければ作成し、そこに出力する
        - 変換中はNotebookに下記のような表示をする(`iteration_count` が10の場合)

          ```
            0%|          | 0/10 [00:00<?, ?it/s]
          ...
           30%|███       | 3/10 [00:15<00:35,  5.10s/it]
          ...
          100%|██████████| 10/10 [00:50<00:00,  5.07s/it]
          ```

        - configuration.json `output_dir`、 `evaluate_image_dir`、 `evaluate_label_file` の存在をチェックする
            - エラー発生時はその内容を表示し、中断する

- configuration.json の下記の内容を読み取り、tflite interpreterへ必要な設定を行う
  - configuration.json **`output_dir`**
  - configuration.json **`evaluate_image_dir`**
  - configuration.json **`evaluate_image_extension`**
  - configuration.json **`evaluate_labe_file`**
  - configuration.json **`evaluate_result_dir`**
- 元のAIモデル(Keras)、TensorFlow標準機能でTFLiteに変換したAIモデル(TFLite)、MCTで量子化したAIモデル(TFLite)の3種のAIモデルで推論実行し、統計情報を表示する
- 統計情報を、**`evaluate_result_dir`** 配下に **`results.json`** ファイルとして保存する
- TensorFlowなどの外製ソフトでエラー発生時は、外製ソフトが出力するエラーを表示し、中断する
- AIモデル(TFLite)の推論実行中は下記のような表示をする(画像数が10の場合)

```
  0%|          | 0/10 [00:00<?, ?it/s]
...
 40%|████      | 4/10 [00:03<00:05,  1.08it/s]
...
100%|██████████| 10/10 [00:09<00:00,  1.08it/s]
```

- AIモデル(Keras)の推論実行中はTensorFlowライブラリによるログを表示する
- 処理中でもNotebook Cell機能のStop Cell Executionで中断できる

7. 目標性能

- SDKの環境構築完了後、追加のインストール手順なしに、AIモデル(Keras)を量子化しAIモデル(TFLite)に変換できること
- UIの応答時間が1.2秒以内であること
- 処理に5秒以上かかる場合は、処理中の表現を逐次更新表示できること

8. 制限事項

- なし

9. その他特記事項

- MCT(model-compression-toolkit)、TensorFlowのバージョン確認方法について
  - SDK環境のルートフォルダにある requirements.txt を参照する

10. 未決定事項

- なし