SONY

AITRIOS

Vision and Sensing Application SDK Model Training Functional Specifications

Version 0.2.0

2023 - 1 - 30

TOC

1. Change history

Date	What/Why
2023/01/30	Initial draft

2. Terms/Abbreviations

Terms/Abbreviations	Meaning
MCT	Open source software for quantizing neural network models
Keras	A Keras model is a type of neural network format
TFLite	TensorFlow Lite A *.tflite* model is a type of neural network format
Iteration	One occasion of neural network model training

3. Reference materials

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

4. Expected use case

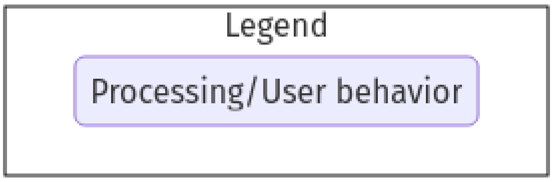
- You want to run transfer learning

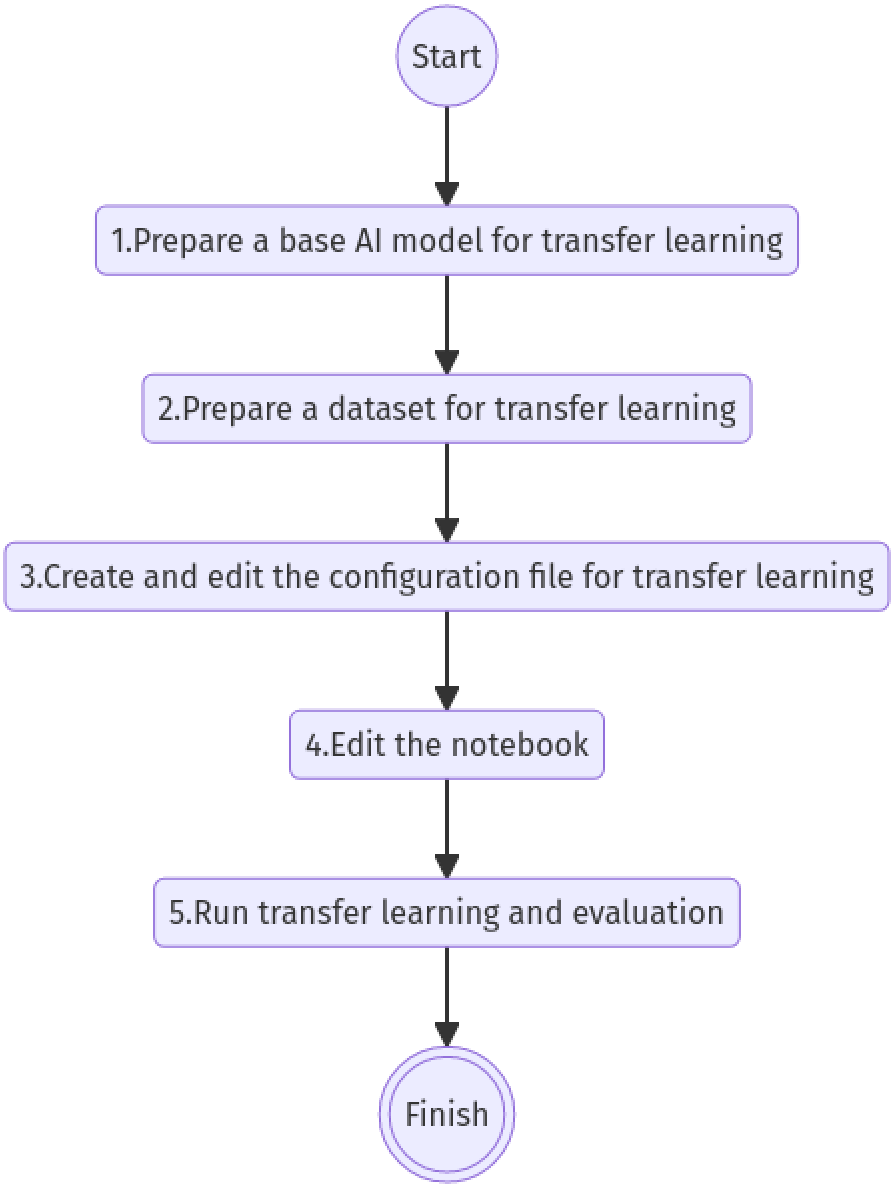
  You want to run inferences and then check the accuracy of the learning process

5. Functional overview/Algorithm

Functional overview

- The SDK enables transfer learning of AI models (Keras) of Image Classification in the following flow
- The SDK runs inferences with AI models learned by transfer to get statistics (Top1 accuracy) for the inference results
- The AI models supported by the SDK conform to MCT supported features
- The image format supported by the SDK is JPEG
- Flow overview




Start
1.Prepare a base AI model for transfer learning
2.Prepare a dataset for transfer learning
3.Create and edit the configuration file for transfer learning
4.Edit the notebook
5.Run transfer learning and evaluation
Finish

- Flow details

    1. Prepare a base AI model for transfer learning

        - Prepare a base AI model in Keras for transfer learning

    2. Prepare a dataset for transfer learning

        - Prepare the dataset images and label information for transfer learning

    3. Create and edit the configuration file for transfer learning

        - Create and edit the configuration file *configuration.json* to configure notebook runtime settings

    4. Edit the notebook

        - If the base AI model contains a top (output) layer, modify the implementation of *remove_top_layer_if_needed()* in the notebook

    5. Run transfer learning and evaluation

        - Run the notebook to run transfer learning and evaluate the inferences

6. User interface specifications

How to start each function

1. Launch the SDK environment and preview the `README.md` in the top directory
2. Jump to the `README.md` in the `tutorials` directory from the hyperlink in the SDK environment top directory
3. Jump to the `README.md` in the `3_prepare_model` directory from the hyperlink in the `README.md` in the `tutorials` directory
4. Jump to the `README.md` in the `develop_on_sdk` directory from the hyperlink in the `README.md` in the `3_prepare_model` directory
5. Jump to the `README.md` in the `1_train_model` directory from the hyperlink in the `README.md` in the `develop_on_sdk` directory
6. Jump to the `README.md` in the `image_classification` directory from the hyperlink in the `README.md` in the `1_train_model` directory
7. Jump to each feature from each file in the `image_classification` directory

Prepare a base AI model for transfer learning

1. Prepare a base AI model in Keras for transfer learning
    - Store the prepared model in the SDK execution environment.

Prepare a dataset for transfer learning

1. Prepare dataset images and label information for transfer learning.

    - Create and store the annotation data in two directories according to the directory structure for ImageNet 1.0 format. Set up one directory for transfer learning and one for evaluation. Store them in the SDK execution environment.

        - For example, if you want to use the *tutorials/_common/dataset* directory, store it as follows:

```
tutorials/
  └ _common
    └ dataset
        ├ training/  (1)
        |  ├ Image class name/
        |  |   └ Image file
        |  ├ Image class name/
        |  |   └ Image file
        |  ├ . . . .
        ├ validation/ (2)
        |  ├ Image class name/
        |  |   └ Image file
        |  ├ Image class name/
        |  |   └ Image file
        |  ├ . . . .
        └ labels.json  (3)
```

(1) Dataset used during transfer learning

(2) Dataset used during evaluation (after transfer learning)

(3) Label information file

        - The format of label information files is a json file with the label name and its id value as follows:

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

> See the "CVAT Image Annotation Functional Specifications" for how to export CVAT-annotated dataset and store it in the SDK runtime environment.

Create and edit the configuration file for transfer learning

1. Create and edit the configuration file, `configuration.json`, in the execution directory.

> ℹ All parameters are required, unless otherwise indicated.

> ℹ All values are case sensitive, unless otherwise indicated.

> ℹ Do not use symbolic links to files and directories.

Configuration	Meaning	Range	Remarks
`source_keras_model`	The base AI model (Keras) path. Specify a directory in Keras SavedModel format or a file in h5 format.	Absolute path or relative to notebook (*.ipynb)	If not specified, uses the Keras standard MobileNetV2 AI model
`dataset_training_dir`	Directory containing dataset images for transfer learning input. Specify a directory structure for ImageNet 1.0 format.	Absolute path or relative to notebook (*.ipynb)	
`dataset_validation_dir`	Directory containing dataset images for evaluation after transfer learning. Specify a directory structure for ImageNet 1.0 format.	Absolute path or relative to notebook (*.ipynb)	
`batch_size`	Batch size of input and evaluation dataset for transfer learning	1 or more (2^n is recommended)	

Configuration	Meaning	Range	Remarks
`input_tensor_size`	Size of the AI model input tensor (number of pixels on one side of image)	Comply with AI model input tensor	
`epochs`	Number of epochs during transfer learning	1 or more	
`output_dir`	Directory to store transfer learned AI models	Absolute path or relative to notebook (*.ipynb)	
`evaluate_result_dir`	Directory to store statistics of inference results	Absolute path or relative to notebook (*.ipynb)	

Edit the notebook

1. Open the notebook for running transfer learning, **.ipynb*, in the execution directory.
2. If the base AI model contains a top (output) layer, modify the implementation of *remove_top_layer_if_needed()* in the notebook

Run transfer learning and evaluation

1. Open the notebook for running transfer learning, **.ipynb*, in the execution directory, and run the python scripts in it.
    - The script does the following:
        - Checks that *configuration.json* exists in the execution directory.
            - If an error occurs, the error description is displayed and running is interrupted.
        - Checks that *configuration.json* includes values for `source_keras_model` and `dataset_training_dir`.
            - If an error occurs, the error description is displayed and running is interrupted.
        - Reads the following values from *configuration.json*, makes the necessary settings in TensorFlow, and then runs transfer learning:
            - *configuration.json* `source_keras_model`
            - *configuration.json* `dataset_training_dir`
            - *configuration.json* `input_tensor_size`
            - *configuration.json* `epochs`
        - If an error occurs in external software, for example, TensorFlow, the error output by the external software is displayed and running is interrupted.
        - Outputs the AI model in Keras SavedModel format to the directory specified in *configuration.json* for `output_dir`.
            - If the directory specified by `output_dir` does not already exist, it is created at the same time.
        - While training, information is displayed as follows (when `epochs` is set to `10`), for example:

          ```
          Epoch 1/10
          3/3 [==============================] - 4s 1s/step - loss: 1.6911 - acc:
          0.3000 - val_loss: 1.8147 - val_acc: 0.1500
          ...
          Epoch 3/10
          3/3 [==============================] - 2s 769ms/step - loss: 1.0132 -
          acc: 0.6750 - val_loss: 1.5243 - val_acc: 0.4000
          ...
          Epoch 10/10
          3/3 [==============================] - 2s 673ms/step - loss: 0.2634 -
          acc: 0.9625 - val_loss: 1.1520 - val_acc: 0.6000
          ```

- Checks that *configuration.json* includes a value for `dataset_validation_dir`.
    - If an error occurs, the error description is displayed and running is interrupted.
- Reads the following values from *configuration.json*, makes the necessary settings in TensorFlow:
    - *configuration.json* `dataset_validation_dir`
    - *configuration.json* `output_dir`
    - *configuration.json* `evaluate_result_dir`
- Runs inferences and displays statistics on AI models learned by transfer.
- Saves statistics as the file `results.json` in the directory specified in `evaluate_result_dir`.
- If an error occurs in external software, for example, TensorFlow, the error output by the external software is displayed and running is interrupted.
- While the AI model is being inferred, logs from TensorFlow library are displayed.
- While processing, you can interrupt with the Stop Cell Execution of notebook cell function.

7. Target performances/Impact on performances

- When the SDK environment is built, transfer learning can be run without any additional installation steps
- UI response time of 1.2 seconds or less
- If processing takes more than 5 seconds, then the display during processing can be updated sequentially

8. Assumption/Restriction

- Depending on the size of the dataset, even if Codespaces has a Machine Type of 4-core, an error will occur due to insufficient memory during transfer learning. In this case, select a Machine Type of 8-core or higher

9. Remarks

- To check the versions of Model Compression Toolkit (MCT) and TensorFlow
    - See *requirements.txt* in the SDK environment root directory.

10. Unconfirmed items

- None