SONY

Vision and Sensing Application SDK Version Control Functional Specifications

Version 0.2.0

2023 - 1 - 30

TOC

1. Change history

Date	What/Why
2022/12/12	Initial draft
2023/01/30	Directory structure change. Added version control targets. Following the deletion of the secret information, description about configuration.json was removed. Updated the PDF build environment.

2. Terms/Abbreviations

Terms/Abbreviations	Meaning
GitHub	Software version control platform

3. Reference materials

- Reference/Related documents
  - GitHub Document
    - https://docs.github.com/ja

4. Expected use case

- You want to version Jupyter Notebook for downloading datasets
- You want to version Jupyter Notebook for AI model quantization
- You want to version post-processing source code

5. Functional overview

Functional overview

- The SDK is provided to users as a GitHub repository
- Users fork (or clone) the SDK repository from GitHub for development
- Users use GitHub features for version control

6. User interface specifications

Prerequisite

- Have a GitHub account

How to start each function

- Access the SDK's repository on GitHub to fork or clone the SDK's repository to user's environment
- Proceed with development according to team or project operating rules

7. FYI: Examples of versioning operations

- The following is an example of how to operate version control
- Follow team or project operating rules, if any
- About the SDK directory structure:
  - This SDK has a separate directory for each feature type as follows
  - Features are independent of each other and can be independently versioned

```
/tutorials
    /_common
    /1_initialize
    /2_prepare_dataset
    /3_prepare_model
    /4_prepare_application
    /5_evaluate
/.devcontainer
/README.md
```

Table 1. Files edited by the user (version control targets)

Feature type	Control target	Description	Possible fix	Purpose of version control (example)
prepare dataset	Notebooks, configuration files	Jupyter Notebooks to download image data for learning	Edit various settings according to your purpose. (Image category, number of images acquired, etc.)	Record the parameters and settings specified at the time of image download, so that you can refer to them when you want to reproduce your learning with the same image data.

Table 1. Files edited by the user (version control targets)

Feature type	Control target	Description	Possible fix	Purpose of version control (example)
prepare model	Notebooks for learning, quantization, configuration files	Jupyter Notebooks for learning and quantizing your own AI models	Edit various settings according to your purpose.	Record parameters and other information used to learn and quantize the model, so that you can refer to them when you want to reproduce the learning with the same parameters.
prepare application	A set of post-processing sources, Makefile	Source code describing AI model post-processing, and build files to compile to Wasm format	Develop post-processing logic.	Improve development efficiency by keeping a history of updates. It also makes it easier for teams to develop.

- Branch
  - By creating a branch, multiple feature development can proceed simultaneously
  - It can also be independently versioned
    by editing the prepare dataset, prepare model and prepare application in separate branches
    - Branch operation example:

      ```
      main
          |-- feature/prepare_application/object_detection_xxx (1)
          |-- feature/prepare_application/image_classification_xxx (2)
          |-- feature/prepare_model/xxx (3)
          |-- feature/prepare_dataset/xxx (4)
          |-- bugfix/XXX (5)
      ```

      (1) Feature development branch for object detection model
      (2) Feature development branch for image classification model
      (3) AI model creation management branch
      (4) Dataset download management branch
      (5) Bug fix branch

- Commit history
  - Commit changes to a file when you want to save them, so you can refer to them later as a revision history
    - Operating example:
      - Save notebook runtime information
        1. Commit to the Git branch with notebooks edited
           - You can reference the settings from your commit history
             if you want to rerun your notebook with the same parameter settings as in the past
        2. In addition, by tying information from the data used for input to the commit, you can reproduce a notebook execution under identical conditions (input data, parameters)
           - It's up to the user how to tie input data to a commit, but an example is to write it as a commit message
             - Sample commit message (for quantize model):

```
Quantization test
    description: xxxxxx
    input model: <url_to_model_resistry>
    dataset: <url_to_dataset_resistry>
    ....

# Please enter the commit message for your changes. Lines
starting
# with '#' will be ignored, and an empty message aborts the
commit.
#
# Committer: XXXXXX
#
# On branch feature/quantize/xxx
# Changes to be committed:
#       modified:   xxxx.ipynb
```

- Tag
  - Tagging commits makes it easier to access the version you need
    - Tagging with Git commands:

```
git tag -a [tag name] -m 'tag comment' [commit id]
```

8. Target performances/Impact on performances

- None

9. Assumption/Restriction

- None

10. Remarks

- None

11. Unconfirmed items

- None