Hugit

Daniel van Strien

Nov 29, 2022

CONTENTS

1	Supported formats	3
2	Features	5
3	Installation	7
4	Usage	9
5	Contributing	11
6	License	13
7	Issues	15
8	Credits	17

Warning: this code is very much a work in progress and is primarily being intended for a particular workflow. It may not work well (or at all) for your workflow.

`hugit` is a command line tool for loading ImageFolder style datasets into a `datasets` `Dataset` and pushing to the hub.

The primary goal of `hugit` is to help quickly get a local dataset into a format that can be used for training computer vision models. `hugit` was developed to support the workflow for `flyswot` where we wanted a quicker iteration between creating new training data, training a model, and using the new model inside `flyswot`.

docs/assets/hugit-workflow.png

CHAPTER

ONE

SUPPORTED FORMATS

At the moment **hugit** supports ImageFolder style datasets i.e:

```
data/
    dog/
        dog1.jpg
    cat/
        cat.1.jpg
```

FEATURES

- A command line interface for quickly loading a dataset stored on disk into a `datasets.Dataset`
- Push your local dataset to the hub
- Get statistics about your dataset. These statistics focus on 'high level' statistic that would be useful to include in Datasheets and Model Cards. Currently these statistics include:
  - label frequencies, organised by split
  - train, test, valid split sizes

CHAPTER

THREE

INSTALLATION

You can install *Hugit* via pip from PyPI, inside a virtual environment install `hugit` using

```
$ pip install hugit
```

Alternatively, you can use pipx to install `hugit`

```
$ pipx install hugit
```

CHAPTER

FOUR

USAGE

You can see help for hugit using hugit --help

```
 Usage: hugit [OPTIONS] COMMAND [ARGS]...

 Hugit Command Line

─ Options␣
↪────────────────────────────────────────────────────────────────────────────────────
│ --help      Show this message and exit.                                          ␣
↪                                                                           │
─────────────────────────────────────────────────────────────────────────────────────
─ Commands␣
↪────────────────────────────────────────────────────────────────────────────────────
│ convert_images                            Convert images in directory to␣
↪`save_format`                                                                    ␣
↪│
│ push_image_dataset                        Load an ImageFolder style dataset. ␣
↪                                                                           │
─────────────────────────────────────────────────────────────────────────────────────
```

To load an ImageFolder style dataset onto the Hub you can use the push_image_dataset command.

```
 Usage: hugit push_image_dataset [OPTIONS] DIRECTORY

 Load an ImageFolder style dataset.

─ Options␣
↪────────────────────────────────────────────────────────────────────────────────────
│ *  --repo-id                                 TEXT     Repo id for the␣
↪Hugging Face Hub [required]                                                      ␣
↪   │
│    --private/--no-private                            Whether to keep␣
↪dataset private on the Hub [default: private]                                    ␣
↪   │
│    --do-resize/--no-do-resize                        Whether to resize␣
↪images before upload [default: no-do-resize]                                     ␣
↪│
```

(continues on next page)

(continued from previous page)

```
│    --size                                        INTEGER  Size to resize image.␣
↪This will be used on the shortest side of the image i.e. the aspect ratio will be  │
│                                                           maintained               ␣
↪                                                                                   │
│                                                           [default: 224]           ␣
↪                                                                                   │
│    --preserve-file-path/--no-preserve-file-path           preserve original file␣
↪path [default: preserve-file-path]                                                 │
│    --ignore-verifications/--no-ignore-verifications       Whether to perform␣
↪verifications on the file before loading into dataset [default: ignore-verifications] │
│    --huggingface-hub-token                       TEXT     Hugging Face Hub␣
↪authentication token  [default: ***]                                                ␣
↪ │
│    --help                                                 Show this message and␣
↪exit.                                                                               │
```

Under the hood `hugit` uses `typed-settings`, which means that configuration can either be done through the command line or through a `TOML` file. See [usage] for more detailed discussion of how to use `hugit`.

CHAPTER FIVE

CONTRIBUTING

It is likely that *Hugit* may only work for our particular workflow. With that said if you have suggestions please open an issue.

CHAPTER

SIX

LICENSE

Distributed under the terms of the MIT license, *Hugit* is free and open source software.

CHAPTER SEVEN

ISSUES

If you encounter any problems, please file an issue along with a detailed description.

CHAPTER EIGHT

CREDITS

This project was generated from @cjolowicz's Hypermodern Python Cookiecutter template.

8.1 Tutorial

8.1.1 Pushing a local dataset to the hub

We can upload an ImageFolder style dataset to the hub directly using `hugit`. An ImageFolder dataset is where the labels are encoded in the part of the folder structure. This often looks something like:

```
data/
    Dog/
        Image1.jpg
    Cat/
        Image1.jpg
```

Where `dog` and `cat` refer to the label of the images contained witin that folder. This type of folder structure is often used for sharing machine learning datasets. It is also one of the possible output formats we might have from an annotation tool. To upload our local data from our machine (or server) to the Hugging Face hub.

Let's have a look at the help for the `push_image_dataset` command.

```
Usage: hugit push_image_dataset [OPTIONS] DIRECTORY

  Load an ImageFolder style dataset.

Options:
  --train-directory TEXT       Name of train directory
  --valid-directory TEXT       name of valid directory
  --test-directory TEXT        name of test directory
  --repo-id TEXT               Repo id for the Hugging Face Hub  [required]
  --private / --no-private     Whether to keep dataset private on the Hub
                               [default: private]
  --do-resize / --no-do-resize  Whether to resize images before upload
                               [default: do-resize]
  --size INTEGER               Size to resize image. This will be used on the
                               shortest side of the image i.e. the aspect rato
                               will be maintained  [default: 224]
  --help                       Show this message and exit.
```

As you can see we have to pass `hugit` some required arguments and some options.

```
hugit load_image_dataset cifar10 --repo-id davanstrin/cifar10
```

8.1.2 Configuration

When we upload an image to the Hugging Face Hub using `hugit` we have a few settings we can configure. These settings include the hugginface hub ID for where the model will be stored e.g. `davanstrien/CIFAR10` and whether to resize your images before uploading. There are two types of setting:

- optional: these you can specify or not
- required: these you must tell hugit about

There are two main ways in which we can specify these settings:

- through the command line interface of `hugit`
- through a `TOML` configuration file.

Passing settings through the Command-Line

```
--do-resize
```

Storing settings in a configuration file

You can also specify your setting in a `TOML` configuration file. `TOML`

As an example configuration

```
[tool.huggit]
hub_id = "davanstrien/CIFAR10"
do_resize = true
size = 224
```

Which format to use?

The command line overwrites the toml configs settings which don't change much can be stored in config

Example

8.2 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the MIT license and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker

- *Code of Conduct*

8.2.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

8.2.2 How to request a feature

Request features on the Issue Tracker.

8.2.3 How to set up your development environment

You need Python 3.7+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run hugit-cli
```

8.2.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the *tests* directory, and are written using the pytest testing framework.

8.2.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

8.3 Contributor Covenant Code of Conduct

8.3.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

8.3.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment

- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

8.3.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

8.3.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

8.3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at davanstrien@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

8.3.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

8.3.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.1, available at https://www.contributor-covenant.org/version/2/1/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

8.4 License