Jovian

Aakash N S, Siddhant Ujjain

Sep 10, 2019

CONTENTS

Jovian is a platform that helps data scientists and ML engineers:

- Track & reproduce data science projects
- Collaborate easily with friends/colleagues, and
- Automate repetitive tasks in their day-to-day workflow.

CHAPTER

ONE

GETTING STARTED

Learn more about installing Jovian python library and some of the core features of Jovian.

Run this command in your terminal:

```
pip install jovian -q --upgrade
```

1.1 Installation

You can do this from the terminal:

```
pip install jovian -q --upgrade
```

Or directly within a Jupyter notebook.

```
!pip install jovian -q --upgrade
```

1.2 Uploading your work to Jovian

It's really easy to get started with Jovian!

1.2.1 Step 1: Install the `jovian` python library

You can do this from the terminal, or directly within a Jupyter notebook.

```
!pip install jovian -q
```

1.2.2 Step 2: Import the library

```
import jovian
```

1.2.3 Step 3: Run jovian.commit

After writing some code, running some experiments, training some models and plotting some charts, you can save and commit your Jupyter notebook.

```
jovian.commit()
```

Here's what `jovian.commit` does:

- It saves and uploads the Jupyter notebook to your Jovian account.
- It captures and uploads the python virtual environment containing the list of libraries required to run your notebook.
- It returns a link that you can use to view and share your notebook with friends or colleagues.

NOTE: When you run `jovian.commit` for the first time, you'll be asked to provide an API, which you can find on your Jovian account.

1.3 Reproducing uploaded notebooks

Once a notebook is uploaded to Jovian, anyone (including you) can download the notebook and it's Python dependencies by running `jovian clone <notebook_id>` command on the Linux/Mac terminal or Windows Command Prompt. Try clicking the 'Clone' button at the top of this page to copy the command (including notebook ID) to clipboard.

```
pip install jovian --upgrade
jovian clone 903a04b17036436b843d70443ef5d7ad
```

Once cloned, you can enter the directly and setup the virtual environment using `jovian install`.

```
cd jovian-demo
jovian install
```

Jovian uses conda internally, so make sure you have it installed before running the above commands. Once the libraries are installed, you can activate the environment and start Jupyter in the usual way:

```
conda activate jovian-demo
jupyter notebook
```

In this way, Jovian seamlessly ensures the end-to-end reproducibility of your Jupyter notebooks.

1.3.1 Updating existing notebooks

Updating existing notebooks is really easy too! Just run `jovian.commit` once again, and Jovian will automatically identify and update the current notebook on your Jovian account.

```
# Updating the notebook
jovian.commit()
```

Jovian keeps track of existing notebooks using a `.jovianrc` file next to your notebook. If you don't want to update the current notebook, but create a new notebook instead, simply delete the `.jovianrc` file. Note that if you rename your notebook, Jovian will upload a new notebook when you commit, instead of updating the old one.

If you run into issues with updating a notebook, or want to replace a notebook in your account using a new/renamed notebook, you can provide the `notebook_id` argument to `jovian.commit`.

```
jovian.commit(notebook_id="903a04b17036436b843d70443ef5d7ad")
```

1.3.2 Getting new changes on cloned notebooks

Once a notebook has been updated, the new changes can be retrieved at any cloned location using the `jovian pull` command.

```
cd jovian-demo # Enter cloned directory
jovian pull    # Pull the latest changes
```

1.4 Development and Testing

To run the tests, run the following command in the project directory

```
python -m unittest discover

["-v" for verbose]
```

1.5 Coming Soon

- Callbacks for Tensorflow, Keras, PyTorch and FastAI to record hyperparameters and metrics automatically
- Full support for Windows, Python 2.7+, non-Anaconda environments and `.py` script files
- Real time monitoring and email/Slack notifications for long running training jobs
- Check out and reproduce tracked experiments on any machine with a single command

For feedback, suggestions and feature requests, drop us a line at hello@jvn.io or create a ticket in the issues tab .

1.6 Commit

jovian.**commit**(*secret=False*, *nb_filename=None*, *files=[]*, *capture_env=True*, *env_type='conda'*, *notebook_id=None*, *create_new=None*, *artifacts=[]*)

Commits a Jupyter Notebook with its environment to Jovian.

Saves the checkpoint of the notebook, capture the required dependencies from the python environment and uploads the notebook, env file, additional files like scripts, csv etc. to https: // www.jvn.io . Capturing the python environment ensures that the notebook can be reproduced and executed easily using the ** *{links to reprodue notebooks}.***

Parameters

- **secret** (*bool, optional*) – Create a secret notebook on Jovian, which is only accessible via the link, and is not visible on the owner's public profile. By default, committed notebooks are public and visible on the owner's profile.
- **nb_filename** (*string, optional*) – The filename of the jupyter notebook(including the .ipynb extension). This is detected automatically in most cases, but in certain environments like Jupyter Lab, the detection may fail and the filename needs to be provided using this argument.
- **files** (*array, optional*) – Any additional scripts(.py files), CSVs that are required to run the notebook. These will be available in the files tab on Jovian .

- **capture_env** (*bool, optional*) – If *True*, the Python environment(python version, libraries etc.) are captured and uploaded along with the notebook.
- **env_type** (*string, optional*) – The type of environment to be captured. Allowed options are 'conda' and 'pip'.
- **notebook_id** (*string, optional*) – If you wish to update an existing notebook owned by you, you can use this argument to provide the base64 ID(present in the URL) of an notebook hosted on Jovian . In most cases, this argument is not required, and the library can automatically infer whether you are looking to update an existing notebook or create a new one.
- **create_new** (*bool, optional*) – If set to True, doesn't update the existing notebook on Jovian(if one is detected). Instead, it creates a new notebook when commit is called.
- **artifacts** (*array, optional*) – Any outputs files or artifacts generated from the modeling processing. This can include model weights/checkpoints, generated CSVs, images etc.

Attention: Pass notebook's name to nb_filename in certain environments like Jupyter Lab, password protected notebooks as sometimes it may fail to detect automatically in these environments.

1.7 Log Dataset, Hyperparams & Metrics

jovian.**log_dataset**(*data*, *verbose=True*)

Record dataset details for the current experiment

Parameters

- **data** (*dict*) – A python dict or a array of dicts to be recorded as Dataset.
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.

Example

```
import jovian

data = {
    'path': '/datasets/mnist',
    'description': '28x28 images of handwritten digits (in grayscale)'
}
jovian.log_dataset(data)
```

jovian.**log_hyperparams**(*data*, *verbose=True*)

Record hyperparameters for the current experiment

Parameters

- **data** (*dict*) – A python dict or a array of dicts to be recorded as hyperparmeters.
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.

Example

```
import jovian

hyperparams = {
    'arch_name': 'cnn_1',
    'lr': .001
}
jovian.log_hyperparams(hyperparams)
```

jovian.**log_metrics**(*data*, *verbose=True*)

Record metrics for the current experiment

Parameters

- **data** (*dict*) – A python dict or a array of dicts to be recorded as metrics.
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.

Example

```
import jovian

metrics = {
    'epoch': 1,
    'train_loss': .5,
    'val_loss': .3,
    'acc': .94
}
jovian.log_metrics(metrics)
```

1.8 Command Line Commands

1.8.1 Initialize

Requests for a API Key for a new user, can find the key at Jovian. By clicking on `API key` button, key will be copied to the clipboard.

```
$ jovian init
```

1.8.2 Clone a Notebook

Clone a notebook form Jovian, by clicking on the `Clone` button of a notebook repo the whole clone command will be copied to the clipboard.

```
$ jovian clone {notebook_id}
```

1.8.3 Pull the latest Notebook

Pull the latest version of the notebook, use the command in a cloned repository or from a repository where you have committed to jovian.

```
$ jovian pull
```

Caution: Make sure the changes are committed if needed, pull overwrites the current notebook.

1.8.4 Install the required dependencies

Install all the dependencies required to the the cloned notebook, use the command in a cloned repository.

```
$ jovian install
```

Important: The above command prompts ' Please provide a name for the conda environment [{env_name}]: '

Press enter to install the dependencies to *env_name* (base env if the content of the square brackets is empty) else provide the env name in the prompt.

1.8.5 Version

Displays the current installed version of jovian library.

```
$ jovian version
```

1.8.6 Enable or Disable Jupyter Notebook Extension

By default, the jovian jupyter extension is enabled.

```
$ jovian enable-ext
```

```
$ jovian disable-ext
```

Note: The changes are observed when the webpage of the notebook is refreshed.

1.9 Fastai Callback

class jovian.callbacks.fastai.**JovianFastaiCallback**(*learn: fastai.basic_train.Learner*, *arch_name: str*)

Fastai callback to automatically log hyperparameters and metrics.

Parameters

- **learn** (*Learner*) – A learner object reference of your current model.
- **arch_name** (*string*) – A name for the model you're training.

Example

```
from jovian.callbacks.fastai_callback import FastaiCallback

jvn_cb = FastaiCallback(learn, 'res18')
learn.fit_one_cycle(5, callbacks = jvn_cb)
```

Tutorial

Visit this for a detailed example on using the keras callback, also visit the *Records* tab to see all the logs of that notebook logged by the callback.

1.10 Keras Callback

class jovian.callbacks.keras.**JovianKerasCallback**(*reset_tracking=True*, *arch_name=''*, *every_epoch=False*, *notify=False*)

Keras Callback to log hyperparameters and metrics during model training.

Parameters

- **reset_tracking** (*string, optional*) – Will clear previously tracked hyperparameters & metrics, and start a fresh recording. Defaults to True.
- **arch_name** (*string, optional*) – A name for the model you're training.
- **every_epoch** (*bool, optional*) – Whether to record losses & metrics for every epoch or just the final loss & metric. Defaults to False.
- **notify** (*bool, optional*) – Whether to send notification on slack when the training ends. Defaults to False.

Example

```
from jovian.callbacks.keras import JovianKerasCallback

# To record logs of every epoch and to notify on slack
jvn_cb = JovianKerasCallback(arch_name='resnet18', every_epoch=True,␣
↪notify=True)
model.fit(x_train, y_train, ...., callbacks=[jvn_cb])
```

Tutorial

Visit this for a detailed example on using the fastai callback, also visit the *Records* tab to see all the logs of that notebook logged by the callback.

1.11 Use Extension to Commit

1.12 Enable or Disable

INDEX