Jovian

Aakash N S, Siddhant Ujjain

Jul 07, 2020

GETTING STARTED

is a platform for sharing and collaboraring on Jupyter notebooks and data science projects. *jovian-py* is an open-source Python package for uploading your data science code, Jupyter notebooks, ML models, hyperparameters, metrics etc. to your Jovian.ml account.

CHAPTER

ONE

INSTALLATION

The `jovian` python library can be installed using the `pip` package manager. To install `jovian` via terminal or command line, run:

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

You can also install the `jovian` library directly within a Jupyter Notebook, by running the following command in a code cell:

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

Caution: If you get a `Permission denied` error, try installing with sudo permission (on Linux/Mac).

```
$ sudo pip install jovian --upgrade
```

Another alternative is to try installing with the `--user` flag, but you'll need to ensure that the target directory is added to your system `PATH`.

```
$ pip install jovian --upgrade --user
```

Once the installation is complete, you can start uploading Jupyter notebooks to Jovian.

Configuration (for Jovian Pro users only)

If you are a Jovian Pro user, run the following commands on the terminal (or command line) to connect the `jovian` library with your company's internal Jovian Pro site:

```
jovian configure
```

You can also do this directly within a Jupyter notebook, by executing the following inside a code cell:

```
import jovian
jovian.configure()
```

The above command prompts for the following information:

1. **Organization ID**: The Organization ID provided by your company for authentication. E.g. if you are accessing Jovian Pro at https://mycompany.jovian.ml , your organization ID is `mycompany`.
2. **API key**: You'll get the API key when you're logged in to your organization's Jovian Pro site. By clicking on the *API key* button, the key will be copied to clipboard.

Note: You need to run `jovian configure` or `jovian.configure()` only once after installation. Your credentials are cached in the `~/.jovian` directory on your computer. You can run `jovian reset` to clear this

configuration.

You can learn more about Jovian Pro here, or start uploading Jupyter notebooks to Jovian in the next section.

CHAPTER

TWO

UPLOADING JUPYTER NOTEBOOKS TO JOVIAN

Jovian allows you to upload and share Jupyter Notebook instantly with a single command, directly within Jupyter. Make sure you've completed the installation before reading further.

2.1 Uploading Notebooks

Step 1: Import `jovian` by running the following command within a Jupyter notebook.

```
import jovian
```

Step 2: After writing some code, running some experiments, training some models and plotting some charts, you can save and commit your Jupyter notebook.

```
jovian.commit()
```

When you run `jovian.commit` for the first time you'll be asked to provide an API key, which you can get from your Jovian.ml (or Jovian Pro) account.

Here's what `jovian.commit` does:

- It saves and uploads the Jupyter notebook to your Jovian.ml (or Jovian Pro) 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.

For more features of `jovian.commit` and API reference visit *Commit*.

> **Attention:** In certain environments like JupyterLab and password protected notebooks, `jovian` may not be able to detect the notebook filename automatically. In such cases, pass the notebook's name as the `nb_filename` argument to `jovian.commit`.

2.2 Benefits of Jovian

Easy sharing and collaboration: Just copy the link to share an uploaded notebook with your friends or colleages. Your notebooks are also visible on your profile page, unless you mark them *Secret*. You can also add collaborators and let others contribute to your project (learn more).

Cell-level comments and discussions: Jovian's powerful commenting interface allows your team to discuss specific parts of a notebook with cell-level comment threads. Just hover over a cell and click the *Comment* button. You'll receive an email when someone comments on your notebook, or replies to your comment.

End-to-end reproducibility: Jovian automatically captures Python libraries used in your notebook, so anyone (including you) can reproduce your work on any computer with a single command: `jovian clone`. You can also use the 'Run' dropdown on the Jovian notebook page to run your notebooks on free cloud GPU platforms like Google Colab, Kaggle Kernels and BinderHub.

This is just a small selection of features that Jovian offers. Continue reading by clicking the `Next ->` button to learn more, or use the sidebar to jump to a specific section.

CHAPTER

THREE

REPRODUCING UPLOADED NOTEBOOKS

An uploaded notebook on Jovian.ml can be reproduced in any other machine. Following are the steps involved to reproduce a notebook.

3.1 Clone

1. Visit the link of the uploaded notebook.
2. Click on the `Clone` button, to copy the notebook's clone command to the clipboard.
3. Paste the command in the terminal, in the directory where you want to clone the notebook project and then run the command.

The copied command will be of the the following format

```
jovian clone <username/project-title>
```

3.1.1 Install

Jovian.ml captures the original python environment of the notebook, which make it easier to reproduce the notebook by installing all the required dependencies. The following commands uses Anaconda to install all the required packages, make sure that conda is installed.

Once the notebook is cloned, it would have created a folder with the name of the notebook project.

Move into that directory.

```
cd jovian-demo
```

Then run

```
jovian install
```

The above command prompts for a virtual environment name where it will install all the required packages. By default it will have the original environment name in the square brackets, just click `enter` key to retain the name else specify the environment name.

In this way, Jovian.ml seamlessly ensures the end-to-end reproducibility of your Jupyter notebooks across different operating systems.

Note: You have to own the notebook or have to be a collaborator to commit changes to the same notebook project. If not you can commit the cloned notebook with any changes to your Jovian profile as a new notebook project.

3.1.2 Pull

If there are any new versions uploaded after you have cloned the notebook by any of the collaborator. You can use `pull` to get all those changes.

Move to the cloned directory and run

```
jovian pull
```

Attention: Beware any uncommitted changes will be lost during the process of `jovian pull`. When you pull the notebook it will be a duplicate of the latest version of the notebook on Jovian.

3.2 Fork

A *fork* is a copy of a notebook. Forking a notebook allows you to freely experiment with changes without affecting the original notebook.

When you clone a notebook from Jovian, it creates a *local* copy of the notebook in your machine. You can fork a notebook instead, to create a copy in your Jovian profile.

Forking a notebook This procedure assumes that the notebook version you're trying to Fork is public, or shared with you if it is secret/private. See Collaboration section for more details.

Forking a notebook will save a copy of the Notebook in your Jovian profile.

1. Visit the notebook version page that you want to Fork.
2. Click on the `Fork` button.
3. This will create a copy of the notebook in your profile, and you will be redirected to the forked version of the Notebook.

CHAPTER

FOUR

NOTEBOOK VERSIONING AND DIFFS

4.1 Version control

If you're used to creating many duplicate versions of notebooks with slight modifications and long file names. Look no further, Jovian.ml will be your version control for notebooks.

`jovian.commit` records all the versions under same notebook project. So, each change can be a version by author and collaborators which can be easily toggled in the website

Note: You have to own the notebook or have to be a collaborator to commit changes to the same project notebook. If not you can commit any changes made to your profile as a new notebook.

4.2 View Differences

All the versions are comparable, you can view additions, deletions made among any 2 versions of the notebook and also hide/show common part of the code.

How to view the differences?

1. *Commit* different versions and visit Jovian.ml .
2. Click on `Version` drop down on the right top corner.
3. Click on `Compare Versions`
4. Select any 2 versions with the use of check boxes and click on `View Diff` button.

There are more things to be compared, but first let's add more content to the notebook to understand all the parameters that can be compared. Click on `Next` to follow through.

CHAPTER

FIVE

ATTACHING FILES AND MODEL OUTPUTS

As seen in the previous section by committing, source code and environment files are captured & uploaded. More files can be attached to the notebook such as files with helper code, output files/model checkpoints that the notebook is generating.

5.1 How to attach files?

```
jovian.commit(files=[], outputs=[])
```

5.2 What to include in the `files` argument?

The type of files which is required to run the notebook.

- Helper code (.py)
- Some input CSVs

5.3 What to include in the artifacts argument?

Any type of outputs that the notebook is generating.

- Saved model or weights (.h5, .pkl, .pth)
- Outputs, Submission CSVs
- Images outputs

5.4 Where to search for the files after committing?

All the attached files are listed under `Files` Tab.

Files can be:

1. Renamed
2. Downloaded
3. Deleted
4. View Raw

5. Uploaded

CHAPTER

SIX

TRACKING DATASETS, HYPERPARAMETERS AND METRICS

Spreadsheets is one of the ways to track information & results of multiple ML experiments. However, using spreadsheets can be tiresome and non-intuitive without the context of the code.

Jovian.ml makes its easy for anyone to track information about datasets, hyperparameters and metrics which are associated with each version of the your experiment in notebooks. Its also displays these information version-by-version of your notebook under single UI.

These information of a notebook are all added to `Records` Tab where you can toggle and view each version's log.

6.1 Dataset

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

6.2 Hyperparameters

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

6.3 Metrics

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

The input to any of these can be a python dict . You can add custom parameters that are related to your experiment and have it record values manually, or automate it to record the values of a variable in a loop. Visit *this* page for these logging API reference.

We have callbacks for *keras* and *fastai* to automatically record hyperparams and metrics check it out.

6.4 Reset

If you're not satisfied with some experiment and want to discard the current recorded logs before a commit. Use

```
jovian.reset()
```

Click `Next` to look at how to compare all of these information of all the versions.

CHAPTER

SEVEN

COMPARING AND ANALYZING EXPERIMENTS

Once you have more than one versions of a notebook, you will be able to use `Compare Versions` present in the `Version` dropdown on the top right corner.

Here you can observe all types of information about all of your versions.

- Title
- Time of Creation
- Author
- All the parameters logged under dataset.
- All the parameters logged under hyperparameters.
- All the parameters logged under metrics.
- Notes (for author and collaborators add extra notes)

7.1 Sort

You can sort any column or a sub-column (For ex: accuracy or any other metric, date of creation etc.) by clicking on the column header.

7.2 Show, Hide and Reorder columns

You can create a custom view to analyse & compare your choice of parameters. Click on `Configure` button and then tick on the checkboxes to create a customized view. Click and drag the elements to reorder them based on your preference.

7.3 Add notes

You can add notes to summarize the experiment for reference or for collaborators to refer to.

7.4 View Diff between specific versions

Select any of the 2 versions by ticking the checkbox next to each version-row of the compare table which can be seen when you hover over any row. Click on `View Diff` button to view the additions and deletion made.

7.5 Archive/Delete versions

Select version/versions by ticking the checkbox of the row/rows. This enables both `Archive` and `Delete` ready for the respective actions.

7.6 Filter

By default all the archived versions are hidden, you can display them by enabling `Show Archived` in `Filter` dropdown.

CHAPTER

EIGHT

COLLABORATING ON JOVIAN PROJECTS

Jovian.ml allows you to add collaborators to work with you on a ML Project.

8.1 How to add collaborators?

Click on `Share` button of the notebook and add them by their username or email id registered with Jovian.ml (you can add a non jovian user email id as well to send an invite).

This will allow the contributors to be able to commit changes to the same notebook project. The experiments by all the collaborators will also show up in the compare table tab.

8.2 Comment on individual code cells

Users can comment on any code cells individually and maintain that thread to have specific discussion about a part of the source code with context.

8.3 Maintain public, secret and private notebooks

You can find the option to `Public`, `Secret` and `Private` in the settings for each notebook.

- **Public** : These notebooks are visible on your public profile and accessible to all.
- **Secret** : These notebooks are hidden from your public profile but anyone with the link can access the notebook.
- **Private** : These notebooks are also hidden from your public profile and are only accessible to the owner and collaborators.

Note that Private and Secret notebooks will still be visible when you're viewing your own profile. To hide a notebook from your own profile you can archive it.

CHAPTER

NINE

EMBED JUPYTER NOTEBOOKS WITH JOVIAN.ML

If you're a blogger who takes screen snips from your Jupyter Notebooks to attach to your blog or want to showcase your wonderful notebooks on a website and wondering how to embed one. We have heard your worries and are here with this feature, any notebook on Jovian.ml is embeddable.

We got you covered if you want to embed any of the following.

- Markdown cell
- Code cell with output
- Code cell without output
- Just the output cell
- or Whole Notebook

9.1 Live Demo

Before seeing how to to embed, have a preview of the embed right here.Below we have embedded a Jupyter notebook in our docs page, using iframe. You can interact with the notebook like copy the source code of a cell, copy a image output, scroll each cells etc.

9.2 Open up Embed modal

Commit a notebook or visit a uploaded notebook on Jovian.ml

9.2.1 Click on Embed button

This is ideal if you need embed the whole notebook

OR

9.2.2 Click on Embed icon on any cell

This is ideal when you need to embed a specific cell

9.3 Customize options and preview

Go ahead if u need more customization like `only output cell` and preview the embed below.

9.4 Copy and Embed

9.4.1 Copy Embed Link for sites which support oEmbed

Medium and Reddit support oEmbed so if you're writing a blog and want to add snippets with or without output charts or helping someone on reddit thread you can use this.

Just copy and paste the embed link, here is a example. Click on the image to visit the medium blog where snippets are added.

9.4.2 Copy iframe code and add it to a website

If you're building a profile website to showcase your projects or academic institution/community/meetup website making a resources page this is right for you.

Just copy and paste the iframe code in the website's code, here is a example. Click on the image to visit the a example Github Page where the whole notebook is embedded.

CHAPTER

TEN

SLACK NOTIFICATIONS

Get notifications from your training experiment and stay updated with all the milestones of your code. No more watching the progress bar of your fit function to keep track of your model training. Use the same integration to get notification about other activities on Jovian.

10.1 Connect to a Slack Workspace

Visit Jovian.ml and click on the `Connect Slack`. You'll be redirected to Slack Webpage.

Choose a workspace from the top right corner and a channel to integrate our Slack app. By clicking on `Allow`, integration will be completed and will get a acknowledgement on the selected channel, this is where you'll be getting all your notifications.

Important: We suggest you to create your own Slack Workspace so that you won't spam with notifications on a public workspace. Or you can choose your DM instead if you don't want to create a new workspace.

10.2 Send Notifications from your script

This will be helpful to get updates on while training a model. You can send any `python dict` or `string`, it can be when some milestones are reached or information about the metrics(accuracy, loss . . .).

We have this integrated to our callbacks to get automated notifications about the metrics, check out *Callbacks Section*.

For API documentation check out *Jovian Slack Notify*

10.3 Integration Preferences

You can customize on what notifications you get to your Slack. To update the preferences visit Jovian.ml Settings or you go to your `Profile Dropdown` on the top right corner and click on `Settings`.

CHAPTER

ELEVEN

ADDING TOPICS TO YOUR NOTEBOOKS

Jovian allows you to add topics to your notebooks. Adding topics to your notebooks helps other people find and contribute to your projects easily. You can add topics related to your project's intended purpose, subject area, frameworks/languages used, or other important qualities that you find useful.

11.1 About Topics

With topics, you can explore notebooks in a particular subject area, find projects to contribute to, discover new solutions to a specific problem, or simply explore the frameworks that you love in action. Topics appear on the main page of a notebook. You can click a topic name to see related list of other notebooks tagged with that topic.

To browse the most used topics, go to https://jovian.ml/explore.

11.2 How to add Topics in your notebooks?

1. Click on **Add Topics** button in the Notebook page.

2. Type the topic you want to add to your notebook, then type a space. Only lowercase letters, digits and hyphens are allowed in a topic name.

3. After you're done adding topics, click **Save**.

Important: Collaborators in your project will also be able to Add/Modify topics.

11.3 Explore Topics

Public, secret and private notebooks can have topics, although you will only see private/secret notebooks that you have access to in topic search results.

To explore more notebooks tagged with a topic, simply click on a topic badge that appears in the Notebook page.

You can search for notebooks that are associated with a particular topic. You can also search for a list of topics on Jovian. Change the URL on topics page with a comma separated list of topics.

Examples:

- https://jovian.ml/topics/pytorch
- https://jovian.ml/topics/pytorch,tutorial

Topics Page

CHAPTER

TWELVE

GIT INTEGRATION

jovian.commit automatically performs git commit if the current notebook/script is in a git repository, as `git_commit` is `True` by default and works only inside a git repository.

Use `git_message` parameter to give a different commit message to git, else it will take jovian's commit message by default.

```
jovian.commit(message="jovian version commit message",
              git_message="git commit message")
```

Jovian also generates a link to the `git commit` associated to each `jovian commit` versions and is accessible with a button on the notebook linking to github/gitlab.

Important: Jovian does not perform `git push`, so if the asscociated link is not available then you'll have push your repo.

CHAPTER

THIRTEEN

JOVIAN PRO

Schedule a meeting for a Demo and discussion with Aakash NS (CEO, Jovian.ml).

Please contact us at `hello@jovian.ml`

CHAPTER

FOURTEEN

COMMIT

jovian.**commit**(*message=None*, *files=[]*, *outputs=[]*, *environment='auto'*, *privacy='auto'*, *filename=None*, *project=None*, *new_project=None*, *git_commit=False*, *git_message='auto'*, ***kwargs*)

Uploads the current file (Jupyter notebook or python script) to

Saves the checkpoint of the notebook, captures the required dependencies from the python environment and uploads the notebook, env file, additional files like scripts, csv etc. to . Capturing the python environment ensures that the notebook can be reproduced.

Parameters

- **message** (*string, optional*) – A short message to be used as the title for this version.
- **files** (*array, optional*) – Any additional scripts(.py files), CSVs etc. that are required to run the notebook. These will be available in the files tab of the project page on Jovian.ml
- **outputs** (*array, optional*) – Any outputs files or artifacts generated from the modeling processing. This can include model weights/checkpoints, generated CSVs, output images etc.
- **environment** (*string, optional*) – The type of Python environment to be captured. Allowed options are 'conda' , 'pip', 'auto' (for automatic detection) and None (to skip environment capture).
- **privacy** (*bool, optional*) – Privacy level of the project (if a new one is being created).
  - 'auto' - use account level settings. Defaults to 'public'
  - 'public' - visible on profile and publicly accessible/searchable
  - 'secret' - not on profile only accessible via the direct link
  - 'private' - only for the accessible to owner and collaborators

  This argument has no effect on existing project. Change the privacy settings of a existing notebook on the webapp.
- **filename** (*string, optional*) – The filename of the current Jupyter notebook or Python script. This is detected automatically in most cases, but in certain environments like Jupyter Lab or password protected notebooks, the detection may fail and the filename needs to be provided using this argument.
- **project** (*string, optional*) – Name of the project to which the current notebook/file should be committed. Format: 'username/title' e.g. 'aakashns/jovian-example' or 'jovian-example' (username of current user inferred automatically). If the project does

not exist, a new one is created. If it exists, the current notebook is added as a new version to the existing project, if you are a owner/collaborator. If left empty, project name is picked up from the *.jovianrc* file in the current directory, or a new project is created using the filename as the project name.

- **new_project** (*bool, optional*) – Whether to create a new project or update the existing one. Allowed option are False (use the existing project, if a .jovianrc file exists, if available), True (create a new project)
- **git_commit** (*bool, optional*) – If True, also performs a Git commit and records the commit hash. This is applicable only when the notebook is inside a Git repository.
- **git_message** (*string, optional*) – Commit message for git. If not provided, it uses the *message* argument

Attention: Pass notebook's name to *filename* argument, in certain environments like Jupyter Lab and password protected notebooks sometimes it may fail to detect notebook automatically.

CHAPTER

FIFTEEN

LOG DATASET, HYPERPARAMS & METRICS

jovian.**log_dataset**(*data_dict=None*, *verbose=True*, ***data_args*)

Record dataset details for the current experiment

Parameters

- **data_dict** (*dict, optional*) – A python dict to be recorded as dataset.
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.
- ****data_args** (*optional*) – Instead of passing a dictionary, you can also pass each individual key-value pair as a argument (see example below)

Example

```
import jovian

path = 'data/mnist'
description = '28x28 images of handwritten digits (in grayscale)'

jovian.log_dataset(path=path, description=description)
# or
jovian.log_dataset({ 'path': path, 'description': description })
```

jovian.**log_hyperparams**(*data_dict=None*, *verbose=True*, ***data_args*)

Record hyperparameters for the current experiment

Parameters

- **data_dict** (*dict, optional*) – A python dict to be recorded as hyperparmeters.
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.
- ****data_args** (*optional*) – Instead of passing a dictionary, you can also pass each individual key-value pair as a argument (see example below)

Example

```
import jovian
jovian.log_hyperparams(arch='cnn', lr=0.001)
# or
jovian.log_hyperparams({ 'arch': 'cnn', 'lr': 0.001 })
```

jovian.**log_metrics**(*data_dict=None*, *verbose=True*, ***data_args*)

Record metrics for the current experiment

Parameters

- **data_dict** (*dict, optional*) – A python dict to be recorded as metrics.
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.
- ****data_args** (*any, optional*) – Instead of passing a dictionary, you can also pass each individual key-value pair as a argument (see example below)

Example

```
import jovian
jovian.log_metrics(epochs=1, train_loss=0.5,
                   val_loss=0.3, val_accuracy=0.9)
# or
jovian.log_metrics({ 'epochs': 1, 'train_loss': 0.5 })
```

jovian.**reset**(**record_types*)

Reset the tracked hyperparameters, metrics or dataset (for a fresh experiment)

Parameters ***record_types** (*strings, optional*) – By default, resets all type of records. To reset specific type of records, pass arguments *metrics*, *hyperparams*, *dataset*

Example

```
import jovian
jovian.reset('hyperparams', 'metrics')
```

CHAPTER

SIXTEEN

SEND NOTIFICATIONS TO SLACK

jovian.**notify**(*data*, *verbose=True*, *safe=False*)

Sends the data to the Slack workspace connected with your Jovian account.

Parameters

- **data** (*dict|string*) – A dict or string to be pushed to Slack
- **verbose** (*bool, optional*) – By default it prints the acknowledgement, you can remove this by setting the argument to False.
- **safe** (*bool, optional*) – To avoid raising ApiError exception. Defaults to False.

Example

```
import jovian

data = "Hello from the Integration!"
jovian.notify(data)
```

Important: This feature requires for your Jovian account to be connected to a Slack workspace, visit Jovian Integrations to integrate them and to control the type of notifications.

CHAPTER

SEVENTEEN

CLI COMMANDS

CLI - Command line interface, is a text-based interface to interact with your system. On Mac/Linux based OS you'll have terminal application where these commands can be executed, on windows it'll be named as command prompt or conda prompt can be used if Anaconda has been installed.

These commands requires *installation* of jovian library, if you're working with pip/conda virtual environments make sure you activate the environment where the jovian library is installed.

These commands ensure that you can directly interact with Jovian from CLI.

CHAPTER

EIGHTEEN

FASTAI CALLBACK

class jovian.callbacks.fastai.**JovianFastaiCallback**(*learn: fastai.basic_train.Learner*, *arch_name=None*, *reset_tracking=True*)

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 import JovianFastaiCallback

jvn_cb = JovianFastaiCallback(learn, 'res18')
learn.fit_one_cycle(5, 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.

CHAPTER

NINETEEN

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.

CHAPTER

TWENTY

OEMBED ENDPOINT FOR EMBEDDING JOVIAN NOTEBOOKS

oEmbed is an open standard to easily embed content from oEmbed providers into your site. You can use the oEmbed standard for embedding Jovian notebooks into your website, have a look at oEmbed.com.

20.1 API Endpoint URL

You can use our API endpoint to request the embed code for public Notebooks, all responses are in json format. Replace {notebook-url} by your Jovian Notebook URL, or Jovian Viewer URL, or any raw `.ipynb` file URL:

- [GET]https://jovian.ml/api/oembed.json/?url={notebook-url}&maxwidth={max-width}

Parameters:

- **url** *(String, required)*: The URL of the notebook or Jovian Viewer URL.
- **cellId** *(Integer, optional)*: Index of the cell of the notebook for cell-level embeds. If no `cellId` id present, whole notebook gets embedded.
- **maxwidth** *(Integer, optional)*: The maximum width of the embedded resource (optional). Note that the max-height parameter is not supported. This is because the embed code is responsive and its height varies depending on its width.

Example URLs:

- Jovian Notebook URLs
  - https://jovian.ml/aakashns/01-pytorch-basics
  - https://jovian.ml/aakashns/movielens-fastai/v/14
- Jovian Viewer URLs
  - http://jovian.ml/viewer?url={notebook-url}
- Raw `.ipynb` file URL

Response:

```
{
  "title": "Jovian Viewer",
  "provider_name": "Jovian",
  "provider_url": "https://jovian.ml",
  "author_name": "Jovian ML",
  "version": "1.0",
  "type": "rich",
  "author_url": "https://github.com/JovianML",
  "height": 800,
```

(continues on next page)

(continued from previous page)

```
  "width": 800,
  "html": "<iframe src='https://jovian.ml/embed?url=https%3A//jvn.storage.googleapis.
↪com/gists/aakashns/5bc23520933b4cc187cfe18e5dd7e2ed/raw/
↪901a9d2508bd441dbf06954c5f46bf58/movielens-fastai.ipynb' title='Jovian Viewer'␣
↪height=800 width=800 frameborder=0 allowfullscreen></iframe>"
}
```

CHAPTER

TWENTYONE

JOVIAN: LIBRARIES AND INTEGRATIONS

Jovian integrates seamlessly with your favorite tools and libraries. Automate your workflow and boost your productivity.

21.1 Jupyter Notebook Extension

Now you can commit your Jupyter Notebook to Jovian with just **One Click**. Make sure you've completed the *Installation* before reading further.

21.1.1 Using Jovian Jupyter Extension

Once you have successfully installed jovian, a new button `Commit` will appear on the tool bar. When using `Commit` button for first time you'll be asked to provide an API key.

You can get the API key at Jovian. Once you log in, just click on `API key` button, and the key will be copied to the clipboard.

Valid API key

If the key is valid you will be notified with the following alert.

Error with API key

If the entered API key is invalid you will get following error.

Successful Commit

Once the API key has been validated, you can start committing to Jovian by clicking `Commit` button. Once the Notebook has been committed successfully you will get the confirmation message with the link where the Jupyter Notebook has been uploaded to, you can use the copy button to get the link to the share the notebook.

21.1.2 Commit with more options

This makes use of *jovian.commit's* parameters to enable the user to commit with preferences like private notebook, new notebook project, to add outputs and files .....

Step 1: click the dropdown menu

Step 2: choose `commit with options`

Note: By default the parameters are derived from *jovian.commit*, changes to any parameter persists after commit.

Step 3: Click on `Commit` to commit the notebook with following options.

21.1.3 Enable or Disable the extension

By default, the Jovian Jupyter Notebook Extension is enabled to the environment where jovian is installed.

You can also disable the extension by running the following command.

```
$ jovian disable-extension
```

To Enable the Notebook Extension, when you have manually disabled it.

```
$ jovian enable-extension
```

21.2 Jupyter Lab Extension

Now you can commit your Jupyter Notebook to Jovian with just **One Click**. Make sure you've completed the *Installation* before reading further.

21.2.1 Using Jovian Jupyter Lab Extension

Once you have successfully installed jovian, a new button `Commit` will appear on the tool bar. When using `Commit` button for first time you'll be asked to provide an API key.

You can get the API key at Jovian. Once you log in, just click on `API key` button, and the key will be copied to the clipboard.

Valid API key

If the key is valid you will be notified with the following alert.

Error with API key

If the entered API key is invalid you will get following error.

Successful Commit

Once the API key has been validated, you can start committing to Jovian by clicking `Commit` button. Once the Notebook has been committed successfully you will get the confirmation message with the link where the Jupyter Notebook has been uploaded to, you can click the link to your Notebook in Jovian.

21.2.2 Commit with more options

This makes use of *jovian.commit's* parameters to enable the user to commit with preferences like private notebook, new notebook project, to add outputs and files .....

Step 1: click the dropdown menu

Step 2: choose `commit with options`

Note: By default the parameters are derived from *jovian.commit*, changes to any parameter persists after commit.

Step 3: Click on `Commit` to commit the notebook with following options.

21.2.3 Enable or Disable the extension from CLI

You can also disable the extension by running the following command.

```
$ jupyter labextension disable jovian-jupyterlab
```

To Enable the Notebook Extension, when you have manually disabled it.

```
$ jupyter labextension enable jovian-jupyterlab
```

21.3 Github Integration

jovian.commit automatically performs git commit if the current notebook/script is in a git repository, as `git_commit` is `True` by default and works only inside a git repository.

Use `git_message` parameter to give a different commit message to git, else it will take jovian's commit message by default.

```
jovian.commit(message="jovian version commit message",
              git_message="git commit message")
```

Jovian also generates a link to the `git commit` associated to each `jovian commit` versions and is accessible with a button on the notebook linking to github/gitlab.

Important: Jovian does not perform `git push`, so if the asscociated link is not available then you'll have push your repo.

21.4 Keras Integration

Step 1 Import

```
import jovian
from jovian.callbacks.keras import JovianKerasCallback
```

Step 2 Pass the callback to the fit method.

```
# 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])
```

For more details visit *Keras callback API reference*

Step 3 Perform jovian commit

```
jovian.commit(message="keras callback")
```

Step 4 View and compare experiment logs

View all the log of a certain version is the `Records Tab`

Compare the results of many expriments that you have performed. For more usage of compare details visit *Compare*

21.5 Fastai Integration

Step 1 Import

```
import jovian
from jovian.callbacks.fastai import JovianFastaiCallback
```

Step 2 Pass the callback to the fit method.

```
learn = cnn_learner(data, resnet34)
jvn_cb = JovianFastaiCallback(learn, 'res18')
learn.fit_one_cycle(5, callbacks = jvn_cb)
```

For more details visit *Fastai callback API reference*

Step 3 Perform jovian commit

```
jovian.commit(message="fastai callback")
```

Step 4 View and compare experiment logs

View all the log of a certain version is the `Records Tab`

Compare the results of many expriments that you have performed.For more usage of compare details visit *Compare*

21.6 VS Code Integration

jovian.commit works for VS Code Notebooks too.

```
jovian.commit(message="example commit", filename="lesson1-pets")
```

Visit *commit api reference* for more commit options

21.7 PyCharm Integration

jovian.commit works for PyCharm notebooks too.

```
jovian.commit(message="pycharm commit")
```

Visit *commit api reference* for more commit options

Important: Jupyter support is provided in PyCharm's Professional version.

21.8 Tensorflow Integration

Page under Construction

21.9 PyTorch Integration

Page under Construction

21.10 Telegram Integration

Page under Construction

21.11 Scikit Learn Integration

Page under Construction

21.12 Xgboost Integration

Page under Construction

21.13 SciPy Integration

Page under Construction

21.14 OpenCV Integration

Page under Construction

21.15 Apache Spark Integration

Page under Construction

21.16 Anaconda Integration

Page under Construction

INDEX