Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

IDSIA / sacred

Sacred is a tool to help you configure, organize, log and reproduce experiments developed at IDSIA.

#python #machine-learning #infrastructure #reproducible-research #reproducibility #reproducible-science #mongodb

1,279 commits | **6** branches | **0** packages | **26** releases | **72** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

boeddeker Made git at import time optional (#724) ··· Latest commit `c1c19a5` on 17 Mar

.github	Make stale time longer (#484)	12 months ago
docs	Fix typo in docs (#715)	4 months ago
examples	Removed create from FileStorageObserver. (#605)	10 months ago

sacred	Made git at import time optional (#724)	3 months ago
tests	Add option to pass a collection prefix to MongoObserver (#704)	6 months ago
.gitignore	[API Change] Changed the way host information is collected. (#569)	10 months ago
.pre-commit-config.yaml	Use jsonpickle handlders for numpy and pandas (#695)	7 months ago
CONTRIBUTING.rst	More info to contribute. (#654)	9 months ago
HISTORY.rst	Minor Patch release	6 months ago
LICENSE.txt	added MIT license as a LICENSE.txt file	6 years ago
MANIFEST.in	Some cleanup in the setup.py (#666)	8 months ago
Makefile	removed unnecessary 'setup.py test' support because it was breaking a...	4 years ago
README.rst	Link to stable version of docs in readme (#620)	9 months ago
azure-pipelines.yml	Added a check for pre-commit in the CI and added flake8 in the pre-co...	9 months ago
dev-requirements.txt	Update jsonpickle version in dev-requirements (#702)	6 months ago
doc-requirements.txt	Add documentation for docstring format (#619)	9 months ago
pyproject.toml	Use black to format code (#579)	10 months ago
requirements.txt	Use jsonpickle handlders for numpy and pandas (#695)	7 months ago
setup.cfg	Added a check for pre-commit in the CI and added flake8 in the pre-co...	9 months ago
setup.py	Use jsonpickle handlders for numpy and pandas (#695)	7 months ago
tox.ini	Update black to stable version (#688)	7 months ago

README.rst

Sacred

> *Every experiment is sacred*
> *Every experiment is great*
> *If an experiment is wasted*
> *God gets quite irate*

Sacred is a tool to help you configure, organize, log and reproduce experiments. It is designed to do all the tedious overhead work that you need to do around your actual experiment in order to:

- keep track of all the parameters of your experiment
- easily run your experiment for different settings
- save configurations for individual runs in a database
- reproduce your results

Sacred achieves this through the following main mechanisms:

- **ConfigScopes** A very convenient way of the local variables in a function to define the parameters your experiment uses.
- **Config Injection**: You can access all parameters of your configuration from every function. They are automatically injected by name.
- **Command-line interface**: You get a powerful command-line interface for each experiment that you can use to change parameters and run different variants.
- **Observers**: Sacred provides Observers that log all kinds of information about your experiment, its dependencies, the configuration you used, the machine it is run on, and of course the result. These can be saved to a MongoDB, for easy

access later.

- **Automatic seeding** helps controlling the randomness in your experiments, such that the results remain reproducible.

Example

Script to train an SVM on the iris dataset

```
from numpy.random import permutation
from sklearn import svm, datasets




C = 1.0
gamma = 0.7



iris = datasets.load_iris()
perm = permutation(iris.target.size)
iris.data = iris.data[perm]
iris.target = iris.target[perm]
clf = svm.SVC(C, 'rbf', gamma=gamma)
clf.fit(iris.data[:90],
        iris.target[:90])
print(clf.score(iris.data[90:],
                iris.target[90:]))
```

The same script as a Sacred experiment

```
from numpy.random import permutation
from sklearn import svm, datasets
from sacred import Experiment
ex = Experiment('iris_rbf_svm')

@ex.config
def cfg():
  C = 1.0
  gamma = 0.7

@ex.automain
def run(C, gamma):
  iris = datasets.load_iris()
  per = permutation(iris.target.size)
  iris.data = iris.data[per]
  iris.target = iris.target[per]
  clf = svm.SVC(C, 'rbf', gamma=gamma)
  clf.fit(iris.data[:90],
          iris.target[:90])
  return clf.score(iris.data[90:],
                   iris.target[90:])
```

Documentation

The documentation is hosted at ReadTheDocs.

Installing

You can directly install it from the Python Package Index with pip:

```
pip install sacred
```

Or if you want to do it manually you can checkout the current version from git and install it yourself:

```
git clone https://github.com/IDSIA/sacred.git
cd sacred
python setup.py install
```

You might want to also install the `numpy` and the `pymongo` packages. They are optional dependencies but they offer some cool features:

```
pip install numpy, pymongo
```

Tests

The tests for sacred use the pytest package. You can execute them by running `pytest` in the sacred directory like this:

```
pytest
```

There is also a config file for tox so you can automatically run the tests for various python versions like this:

```
tox
```

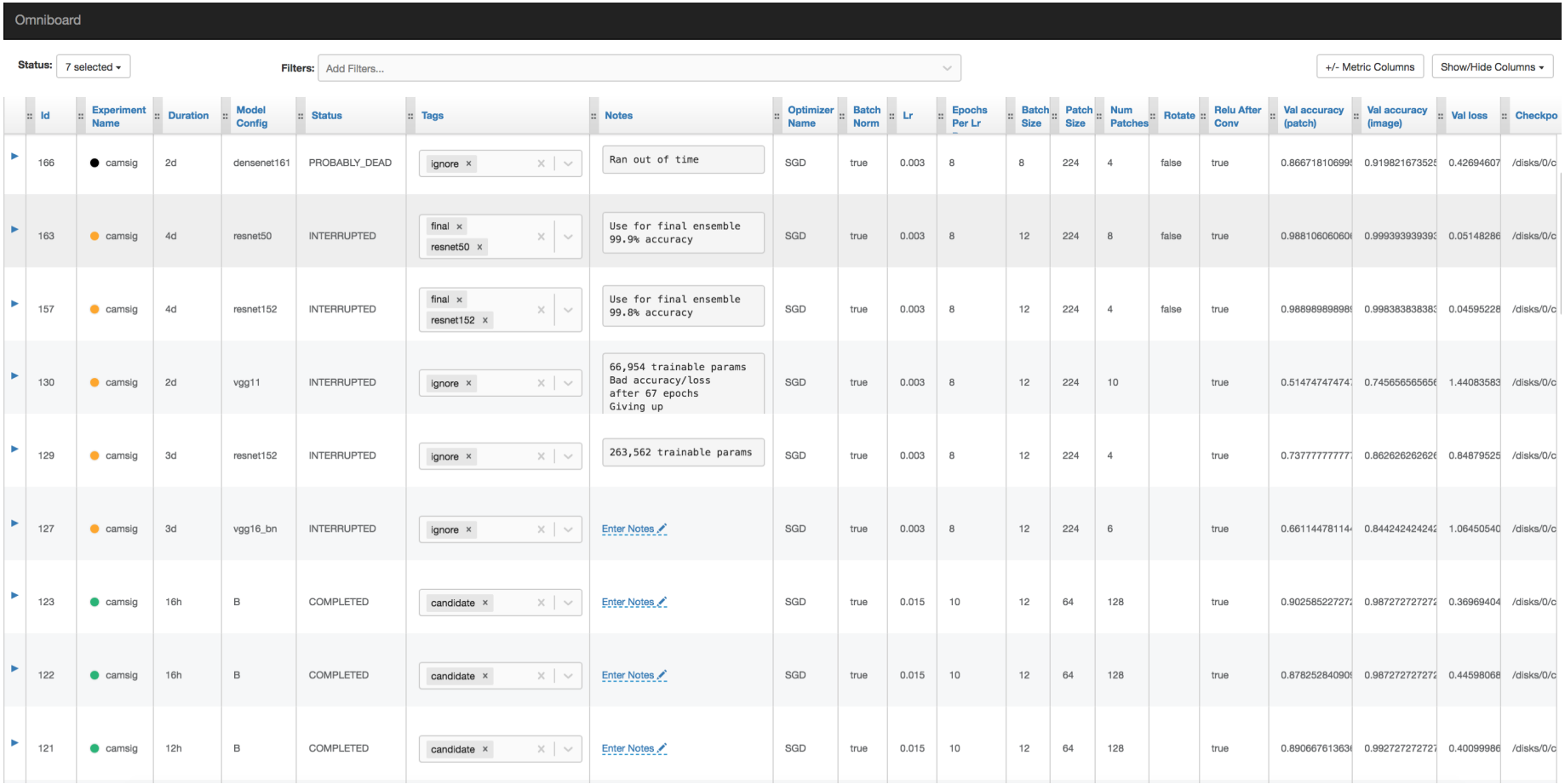
Contributing

If you find a bug, have a feature request or want to discuss something general you are welcome to open an issue. If you have a specific question related to the usage of sacred, please ask a question on StackOverflow under the python-sacred tag. We value documentation a lot. If you find something that should be included in the documentation please document it or let us know whats missing. If you are using Sacred in one of your projects and want to share your code with others, put your repo in the Projects using Sacred <docs/projects_using_sacred.rst>_ list. Pull requests are highly welcome!

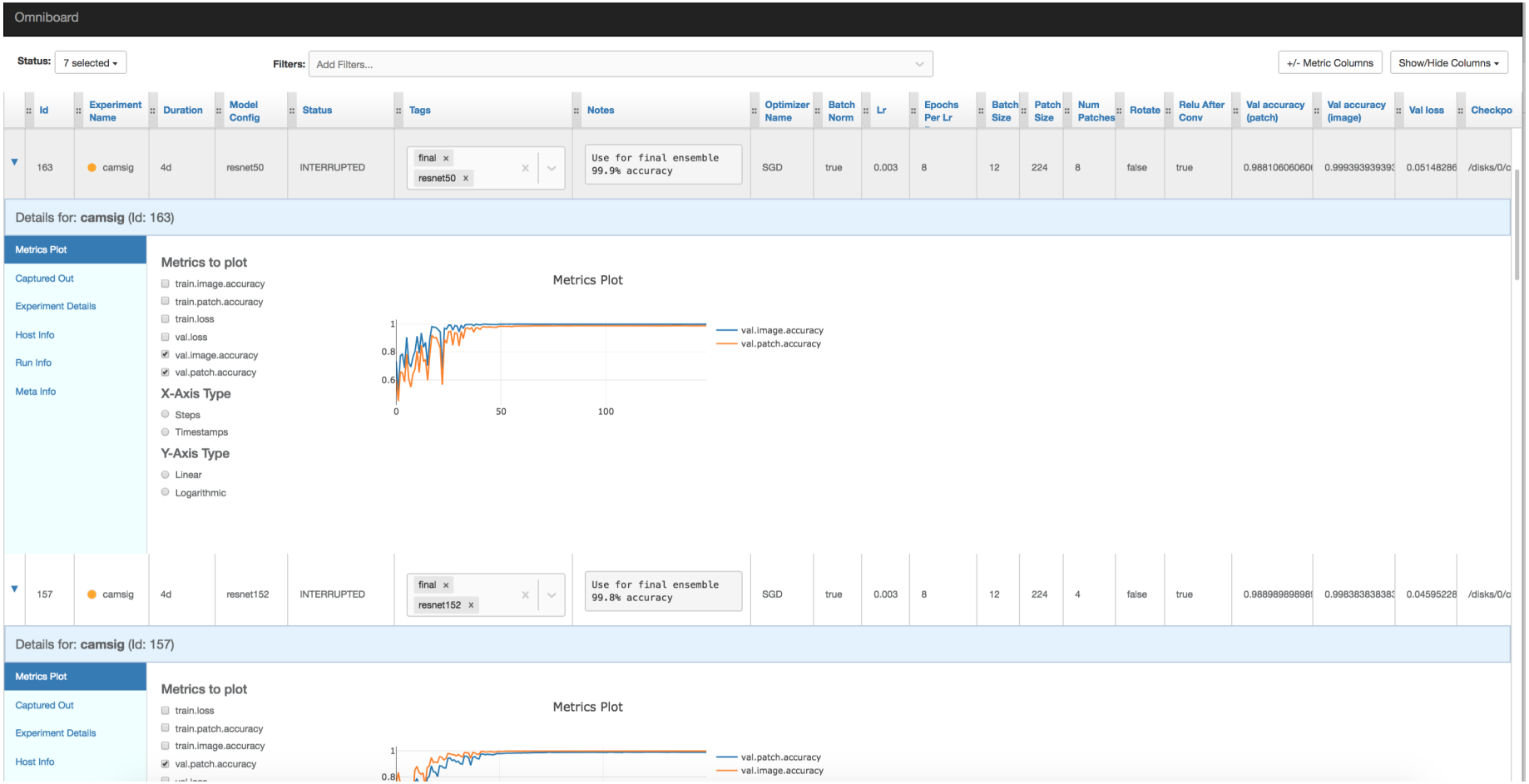
Frontends

At this point there are three frontends to the database entries created by sacred (that I'm aware of). They are developed externally as separate projects.

Omniboard



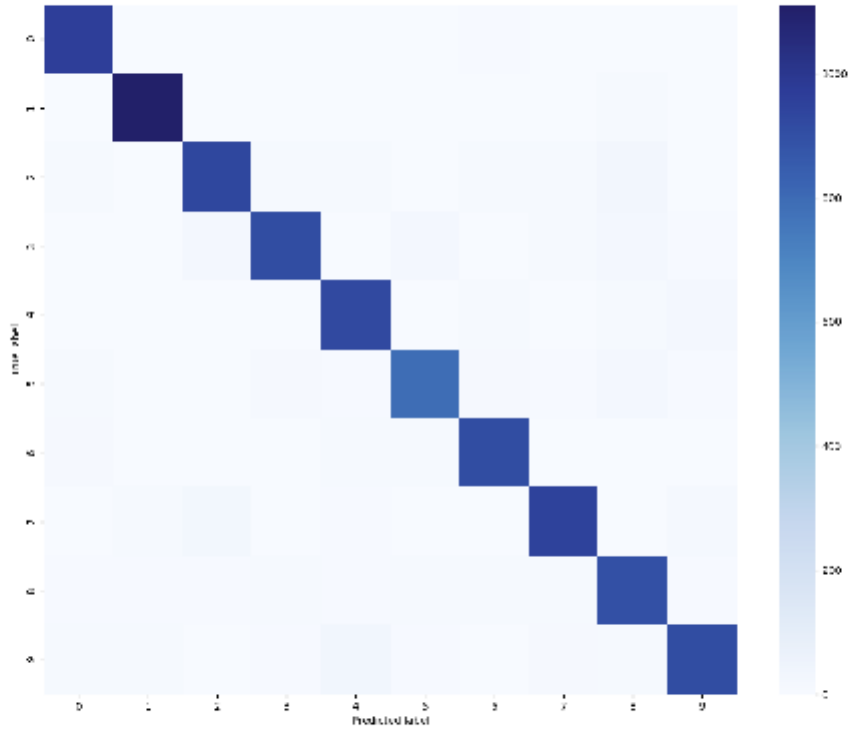
	Id	Experiment Name	Duration	Model Config	Status	Tags	Notes	Optimizer Name	Batch Norm	Lr	Epochs Per Lr	Batch Size	Patch Size	Num Patches	Rotate	Relu After Conv	Val accuracy (patch)	Val accuracy (image)	Val loss	Checkpo
▶	166	camsig	2d	densenet161	PROBABLY_DEAD	ignore	Ran out of time	SGD	true	0.003	8	8	224	4	false	true	0.866718106995	0.919821673525	0.42694607	/disks/0/c
▶	163	camsig	4d	resnet50	INTERRUPTED	final, resnet50	Use for final ensemble 99.9% accuracy	SGD	true	0.003	8	12	224	8	false	true	0.988106060606	0.999393939393	0.05148286	/disks/0/c
▶	157	camsig	4d	resnet152	INTERRUPTED	final, resnet152	Use for final ensemble 99.8% accuracy	SGD	true	0.003	8	12	224	4	false	true	0.988989898989	0.998383838383	0.04595228	/disks/0/c
▶	130	camsig	2d	vgg11	INTERRUPTED	ignore	66,954 trainable params Bad accuracy/loss after 67 epochs Giving up	SGD	true	0.003	8	12	224	10		true	0.514747474747	0.745656565656	1.44083583	/disks/0/c
▶	129	camsig	3d	resnet152	INTERRUPTED	ignore	263,562 trainable params	SGD	true	0.003	8	12	224	4		true	0.737777777777	0.862626262626	0.84879525	/disks/0/c
▶	127	camsig	3d	vgg16_bn	INTERRUPTED	ignore	Enter Notes	SGD	true	0.003	8	12	224	6		true	0.661144781144	0.844242424242	1.06450540	/disks/0/c
▶	123	camsig	16h	B	COMPLETED	candidate	Enter Notes	SGD	true	0.015	10	12	64	128		true	0.902585227272	0.987272727272	0.36969404	/disks/0/c
▶	122	camsig	16h	B	COMPLETED	candidate	Enter Notes	SGD	true	0.015	10	12	64	128		true	0.878252840909	0.987272727272	0.44598068	/disks/0/c
▶	121	camsig	12h	B	COMPLETED	candidate	Enter Notes	SGD	true	0.015	10	12	64	128		true	0.890667613636	0.992727272727	0.40099986	/disks/0/c



Omniboard is a web dashboard that helps in visualizing the experiments and metrics / logs collected by sacred. Omniboard is written with React, Node.js, Express and Bootstrap.

Incense

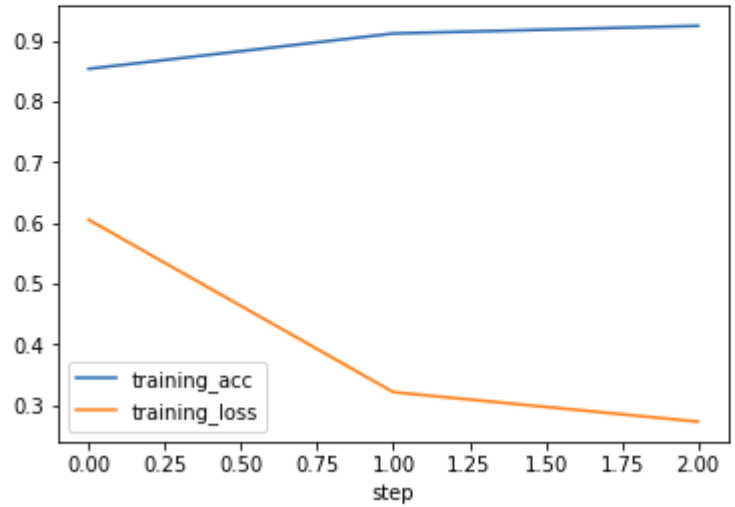
[16]:
```
exp.artifacts['confusion_matrix'].show(figsize=(10, 10));
```



[17]:
```
exp.artifacts['confusion_matrix'].save()
```

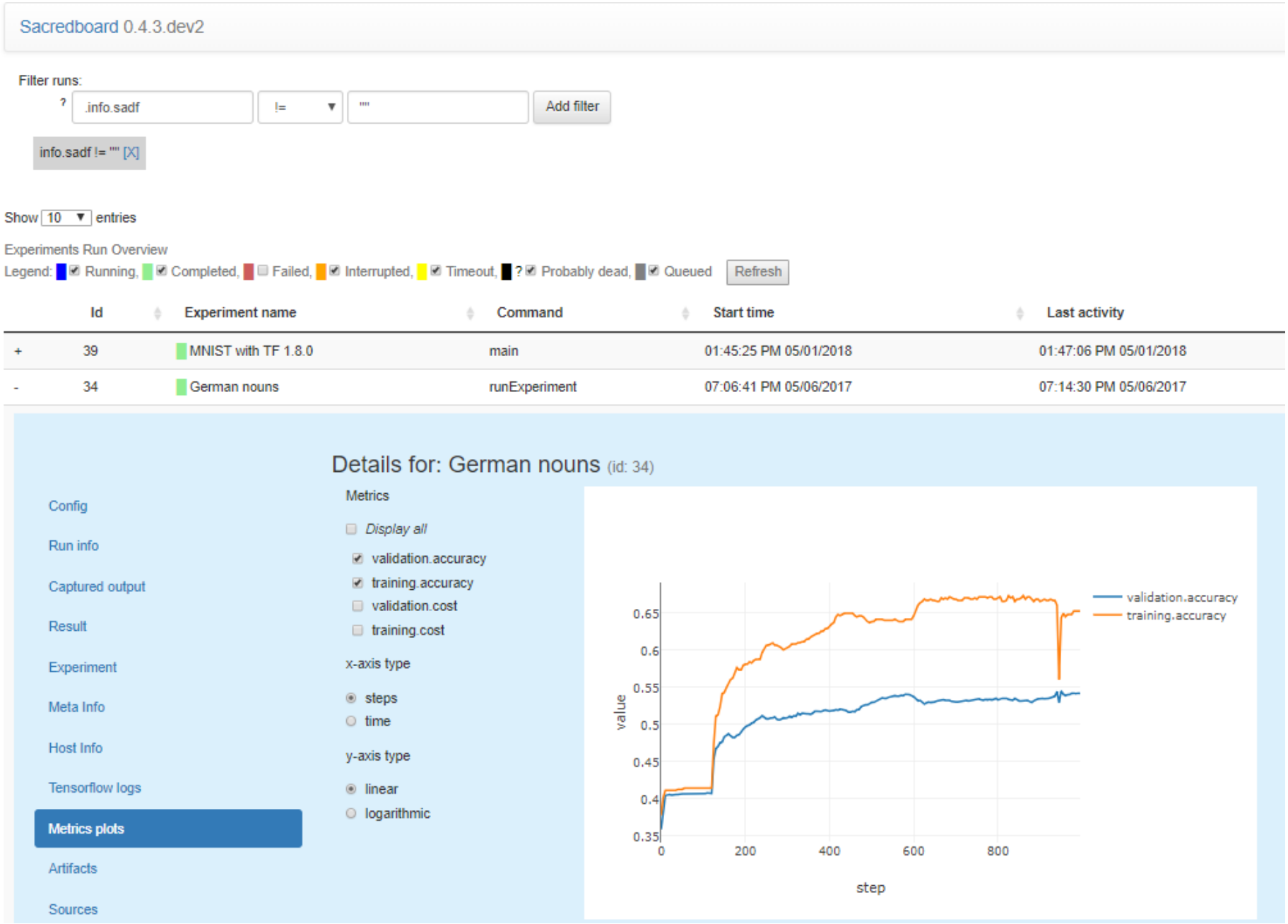
```
[23]: exp.metrics['training_acc'].plot()
      exp.metrics['training_loss'].plot()
      plt.legend()
```

```
[23]: <matplotlib.legend.Legend at 0x7f5b34d9feb8>
```



Incense is a Python library to retrieve runs stored in a MongoDB and interactively display metrics and artifacts in Jupyter notebooks.

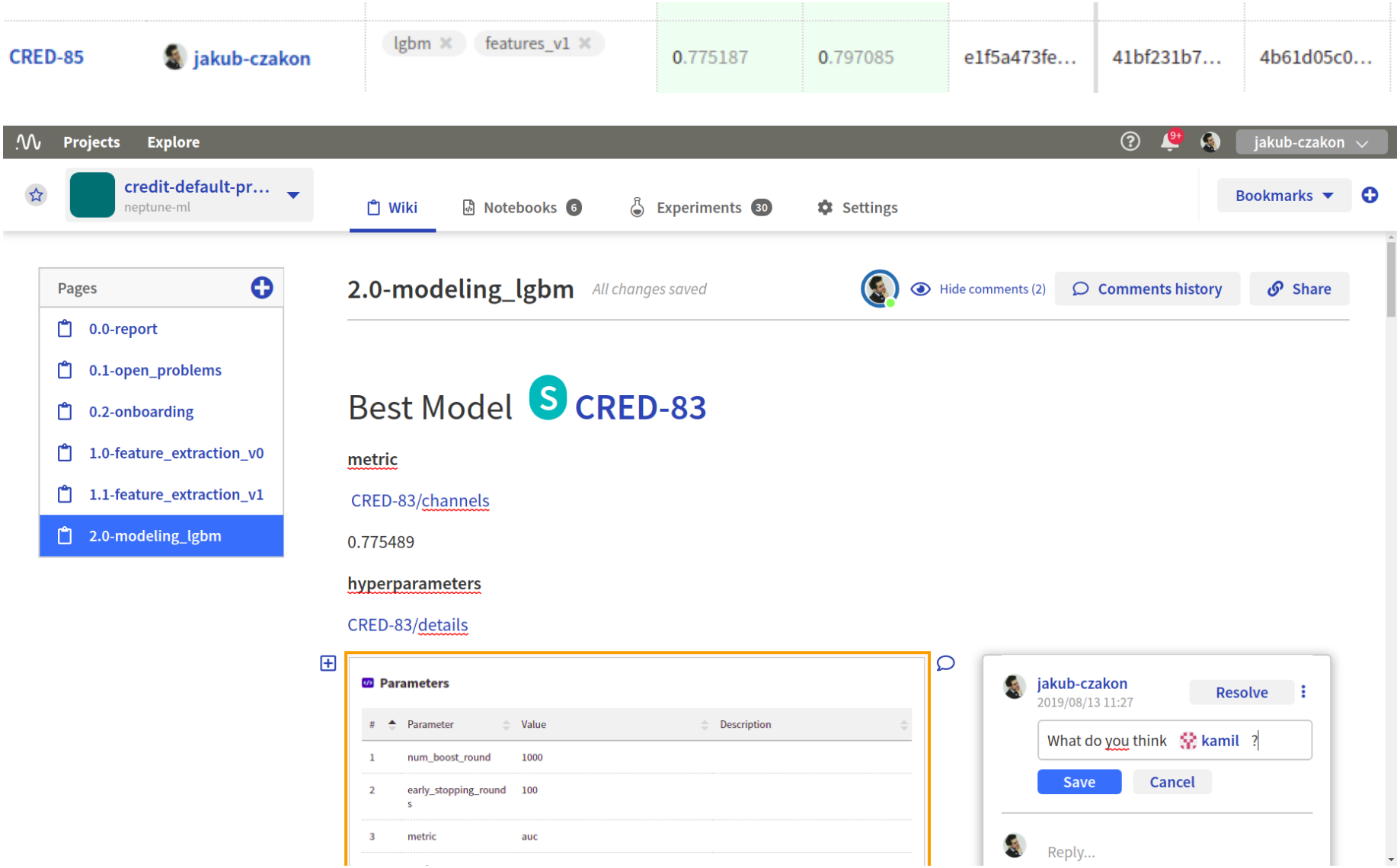
Sacredboard



Sacredboard is a web-based dashboard interface to the sacred runs stored in a MongoDB.

Neptune

Short ID	Owner	Tags	valid_auc	train_auc	features_...	train_spli...	valid_spli...
CRED-93	kamil	lgbm features_v1	0.773536	0.807047	e1f5a473fe...	41bf231b7...	4b61d05c0...
CRED-92	kamil	lgbm features_v1	0.765647	0.790213	e1f5a473fe...	41bf231b7...	4b61d05c0...
CRED-91	kamil	lgbm features_v1	0.767254	0.78022	e1f5a473fe...	41bf231b7...	4b61d05c0...
CRED-89	kamil	lgbm features_v1	0.772704	0.788251	e1f5a473fe...	41bf231b7...	4b61d05c0...

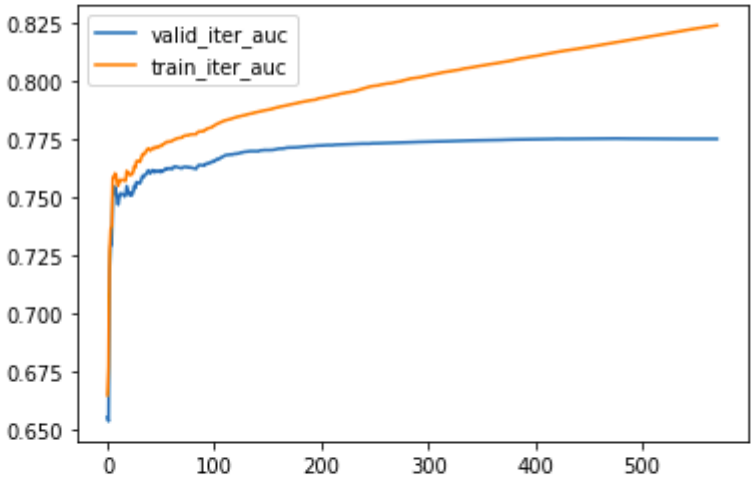


Neptune is a web service that lets you visualize, organize and compare your experiment runs. Once things are logged to Neptune you can share it with others, add comments and even access objects via experiment API:

[3]:
```
exp = project.get_experiments(id=['CRED-80'])[0]
exp
```

[3]:
```
Experiment(CRED-80)
```

[4]:
```
fig, ax = plt.subplots()
df = exp.get_numeric_channels_values('train_iter_auc','valid_iter_auc')
ax.plot(df.x, df.valid_iter_auc, label='valid_iter_auc')
ax.plot(df.x, df.train_iter_auc, label='train_iter_auc')
plt.legend()
plt.show()
```



[5]:
```
exp.get_properties()
```

[5]:
```
{'features_path': 'data/processed/features_joined_v1.csv',
 'features_version': 'e1f5a473fedb57d12a792c02cc52993d',
 'train_split_version': '41bf231b7dc76abb7a2fe15e5cbd9d20',
 'valid_split_version': '4b61d05c05aa16f71015614e3e3b1d1a'}
```

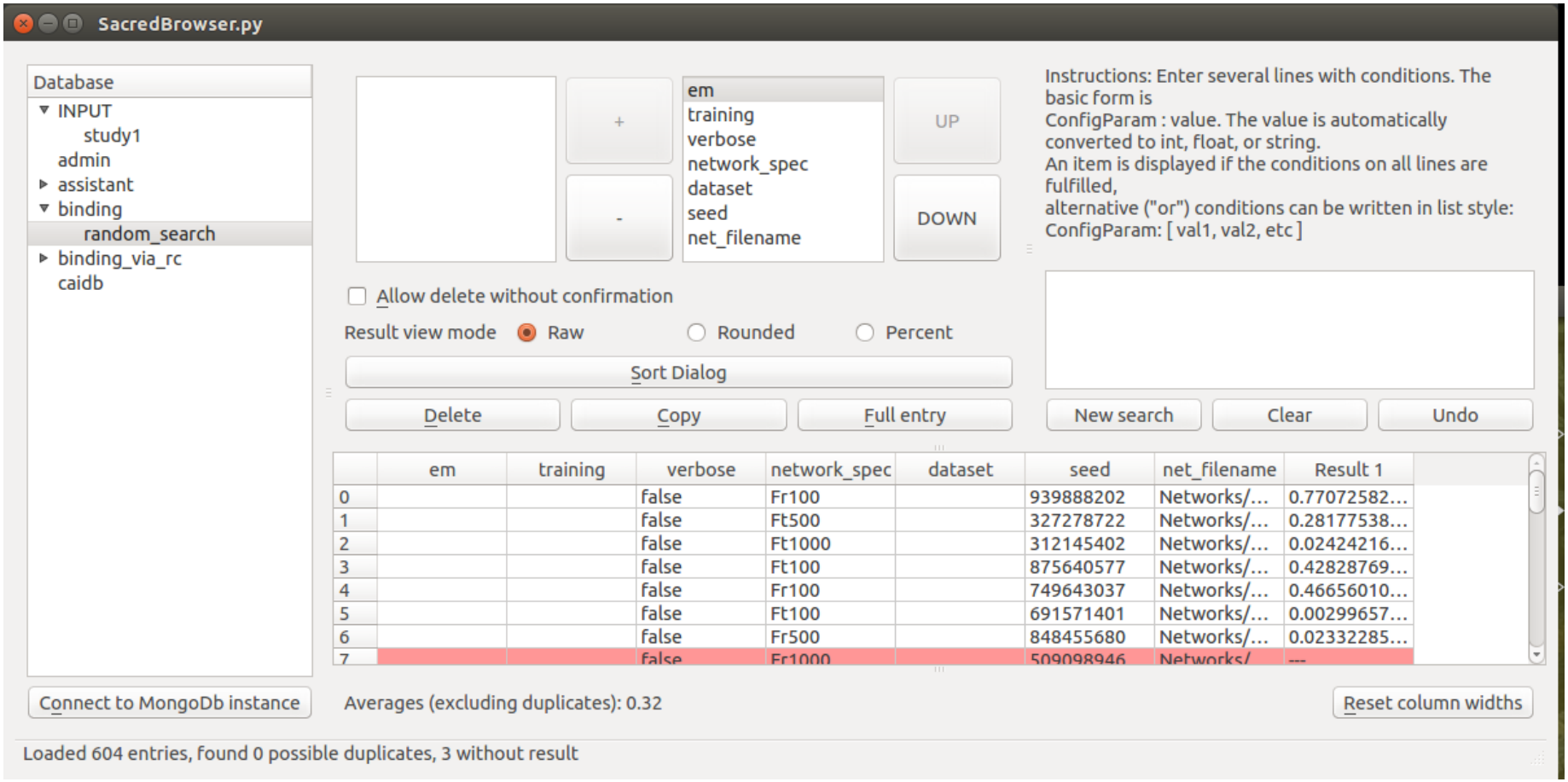
In order to log your runs to Neptune, all you need to do is add an observer:

```
from neptunecontrib.monitoring.sacred import NeptuneObserver
ex.observers.append(NeptuneObserver(api_token='YOUR_API_TOKEN',
```

```
project_name='USER_NAME/PROJECT_NAME'))
```

For more info, check the neptune-contrib library.

SacredBrowser



SacredBrowser is a PyQt4 application to browse the MongoDB entries created by sacred experiments. Features include custom queries, sorting of the results, access to the stored source-code, and many more. No installation is required and it can connect to a local database or over the network.

Prophet

Prophet is an early prototype of a webinterface to the MongoDB entries created by sacred experiments, that is discontinued. It requires you to run RestHeart to access the database.

Related Projects

Sumatra

> Sumatra is a tool for managing and tracking projects based on numerical simulation and/or analysis, with the aim of supporting reproducible research. It can be thought of as an automated electronic lab notebook for computational projects.

Sumatra takes a different approach by providing commandline tools to initialize a project and then run arbitrary code (not just python). It tracks information about all runs in a SQL database and even provides a nice browser tool. It integrates less tightly with the code to be run, which makes it easily applicable to non-python experiments. But that also means it requires more setup for each experiment and configuration needs to be done using files. Use this project if you need to run non-python experiments, or are ok with the additional setup/configuration overhead.

Future Gadget Laboratory

> FGLab is a machine learning dashboard, designed to make prototyping experiments easier. Experiment details and results are sent to a database, which allows analytics to be performed after their completion. The server is FGLab, and the clients are FGMachines.

Similar to Sumatra, FGLab is an external tool that can keep track of runs from any program. Projects are configured via a JSON schema and the program needs to accept these configurations via command-line options. FGLab also takes the role of a basic scheduler by distributing runs over several machines.

CDE

By tracing system calls during program execution CDE creates a snapshot of **all** used files and libraries to guarantee the ability to reproduce any unix program execution. It *only* solves reproducibility, but it does so thoroughly.

License

This project is released under the terms of the MIT license.

Citing Sacred

K. Greff, A. Klein, M. Chovanec, F. Hutter, and J. Schmidhuber, 'The Sacred Infrastructure for Computational Research', in Proceedings of the 15th Python in Science Conference (SciPy 2017), Austin, Texas, 2017, pp. 49–56.