REPRODUCIBILITY IN COMPUTATIONAL RESEARCH

Logan Ward
Asst. Computational Scientist
Argonne National Laboratory

15 January 2022

Computational Research *Should* Be Always Reproducible

Why? Every little bit is broken down into precise instructions, executed by something else

Why not?

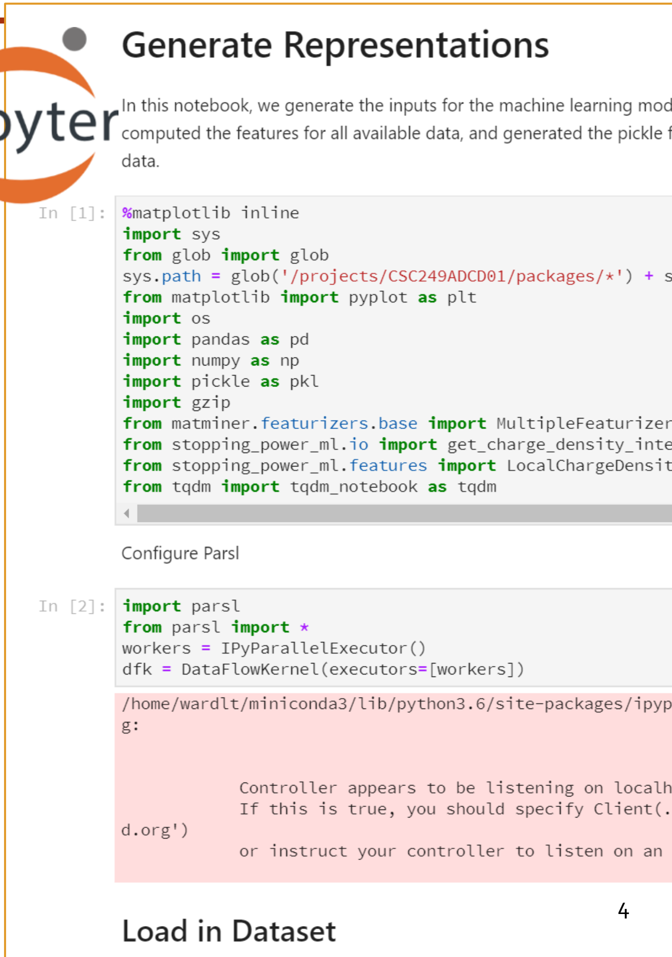
- Some algorithms are random: different results each time
- Computers do not record every bit of data they process

What is your role? Record what (for computer) you did and why (for humans)

ITEM 1: RECORD YOUR WORK IN NOTEBOOKS!

What is Jupyter? Why care?

- Jupyter is environment designed for reproducible computational science
- Stores code with outputs and documentation in a single notebook
  - Modeled after Mathematica
  - https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/
- Why do I care? Jupyter lets me…
  - organize my code
  - easily run on a remote system
  - keep track of results and rationale
  - communicate my research better



Generate Representations

In this notebook, we generate the inputs for the machine learning mod
computed the features for all available data, and generated the pickle f
data.

```
In [1]: %matplotlib inline
        import sys
        from glob import glob
        sys.path = glob('/projects/CSC249ADCD01/packages/*') + s
        from matplotlib import pyplot as plt
        import os
        import pandas as pd
        import numpy as np
        import pickle as pkl
        import gzip
        from matminer.featurizers.base import MultipleFeaturizer
        from stopping_power_ml.io import get_charge_density_inte
        from stopping_power_ml.features import LocalChargeDensit
        from tqdm import tqdm_notebook as tqdm
```

Configure Parsl

```
In [2]: import parsl
        from parsl import *
        workers = IPyParallelExecutor()
        dfk = DataFlowKernel(executors=[workers])
```

```
/home/wardlt/miniconda3/lib/python3.6/site-packages/ipyp
g:

            Controller appears to be listening on localh
            If this is true, you should specify Client(.
d.org')
            or instruct your controller to listen on an
```

Load in Dataset

How I write a notebook

One notebook per "experiment" or "idea"

1. Title and short abstract

   "What am I doing here and why"

2. Load in libraries

   Put them up front, so it crashes early

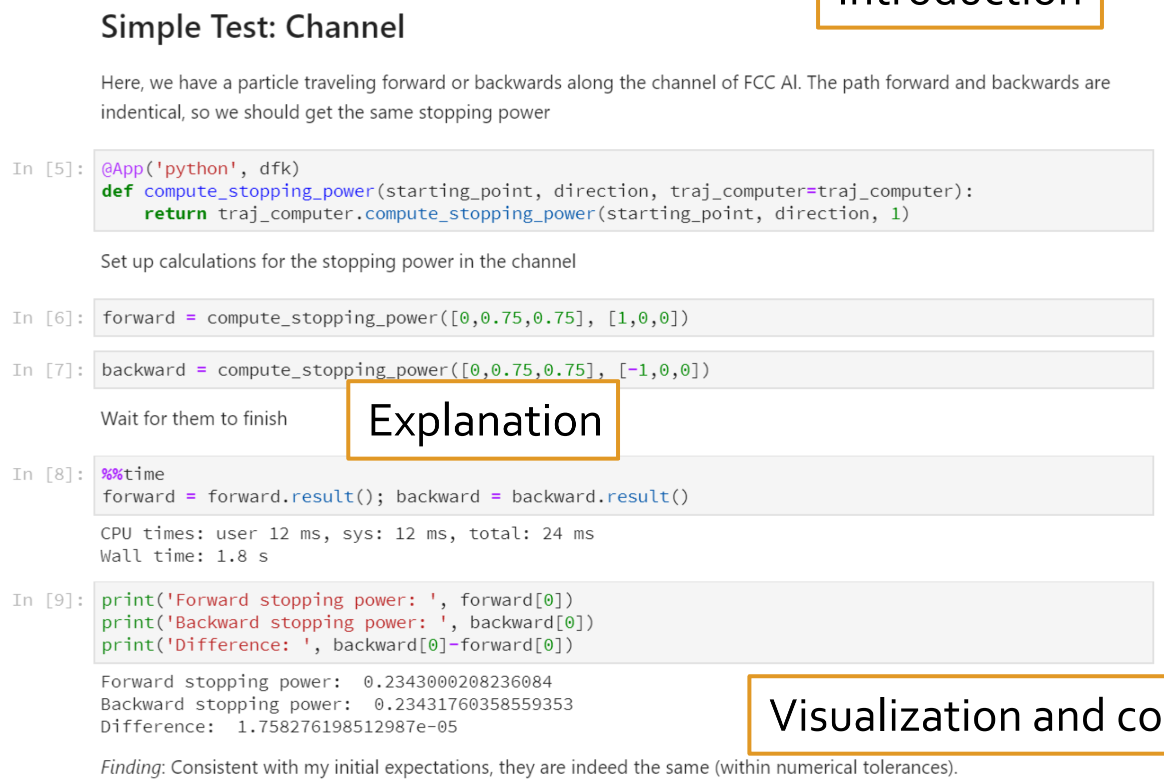
3. Load in data from disk

   Ex: training data, results from other notebook

4. Each step in their own block

   - Introduction
   - Code <u>and explanation</u>
   - Figure/visualization
   - Explanation of finding

Example Step



Introduction

Simple Test: Channel

Here, we have a particle traveling forward or backwards along the channel of FCC Al. The path forward and backwards are indentical, so we should get the same stopping power

```
In [5]: @App('python', dfk)
        def compute_stopping_power(starting_point, direction, traj_computer=traj_computer):
            return traj_computer.compute_stopping_power(starting_point, direction, 1)
```

Set up calculations for the stopping power in the channel

```
In [6]: forward = compute_stopping_power([0,0.75,0.75], [1,0,0])
```

```
In [7]: backward = compute_stopping_power([0,0.75,0.75], [-1,0,0])
```

Explanation

Wait for them to finish

```
In [8]: %%time
        forward = forward.result(); backward = backward.result()

CPU times: user 12 ms, sys: 12 ms, total: 24 ms
Wall time: 1.8 s
```

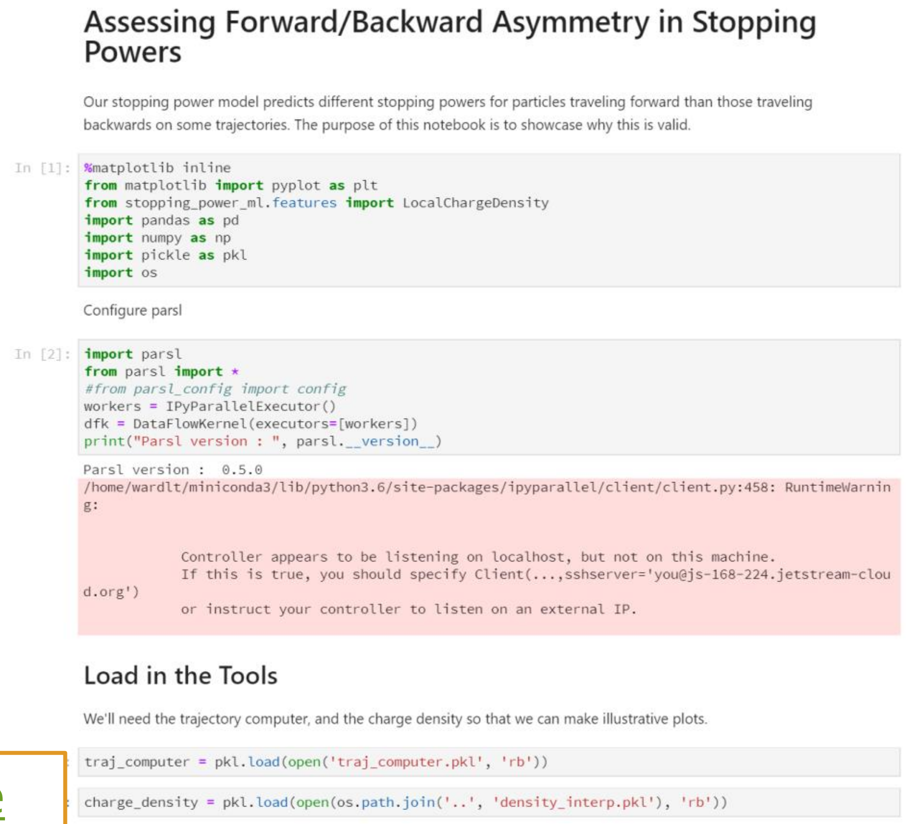
```
In [9]: print('Forward stopping power: ', forward[0])
        print('Backward stopping power: ', backward[0])
        print('Difference: ', backward[0]-forward[0])

Forward stopping power:  0.2343000208236084
Backward stopping power:  0.23431760358559353
Difference:  1.758276198512987e-05
```

Visualization and conclusion

Finding: Consistent with my initial expectations, they are indeed the same (within numerical tolerances).

Someone should be able to understand this without knowing Python!

The Full Narrative



Assessing Forward/Backward Asymmetry in Stopping Powers

Our stopping power model predicts different stopping powers for particles traveling forward than those traveling backwards on some trajectories. The purpose of this notebook is to showcase why this is valid.

In [1]:
```
%matplotlib inline
from matplotlib import pyplot as plt
from stopping_power_ml.features import LocalChargeDensity
import pandas as pd
import numpy as np
import pickle as pkl
import os
```

Configure parsl

In [2]:
```
import parsl
from parsl import *
#from parsl_config import config
workers = IPyParallelExecutor()
dfk = DataFlowKernel(executors=[workers])
print("Parsl version : ", parsl.__version__)
```

```
Parsl version :  0.5.0
/home/wardlt/miniconda3/lib/python3.6/site-packages/ipyparallel/client/client.py:458: RuntimeWarnin
g:


            Controller appears to be listening on localhost, but not on this machine.
            If this is true, you should specify Client(...,sshserver='you@js-168-224.jetstream-clou
d.org')
            or instruct your controller to listen on an external IP.
```

Load in the Tools

We'll need the trajectory computer, and the charge density so that we can make illustrative plots.

```
traj_computer = pkl.load(open('traj_computer.pkl', 'rb'))
```

```
charge_density = pkl.load(open(os.path.join('..', 'density_interp.pkl'), 'rb'))
```

Link to GitHub Page

Pitfalls with Notebooks

1. Not including documentation

   Advice: Write what you're going to do first

2. Notebook not capturing entire process

   Problem: Jupyter lets you execute cells out of order

   Advice: Periodically "Restart Kernel and Run All Cells"

3. Duplicate code between notebooks

   Advice: Make a separate module for common code

4. Library conflicts

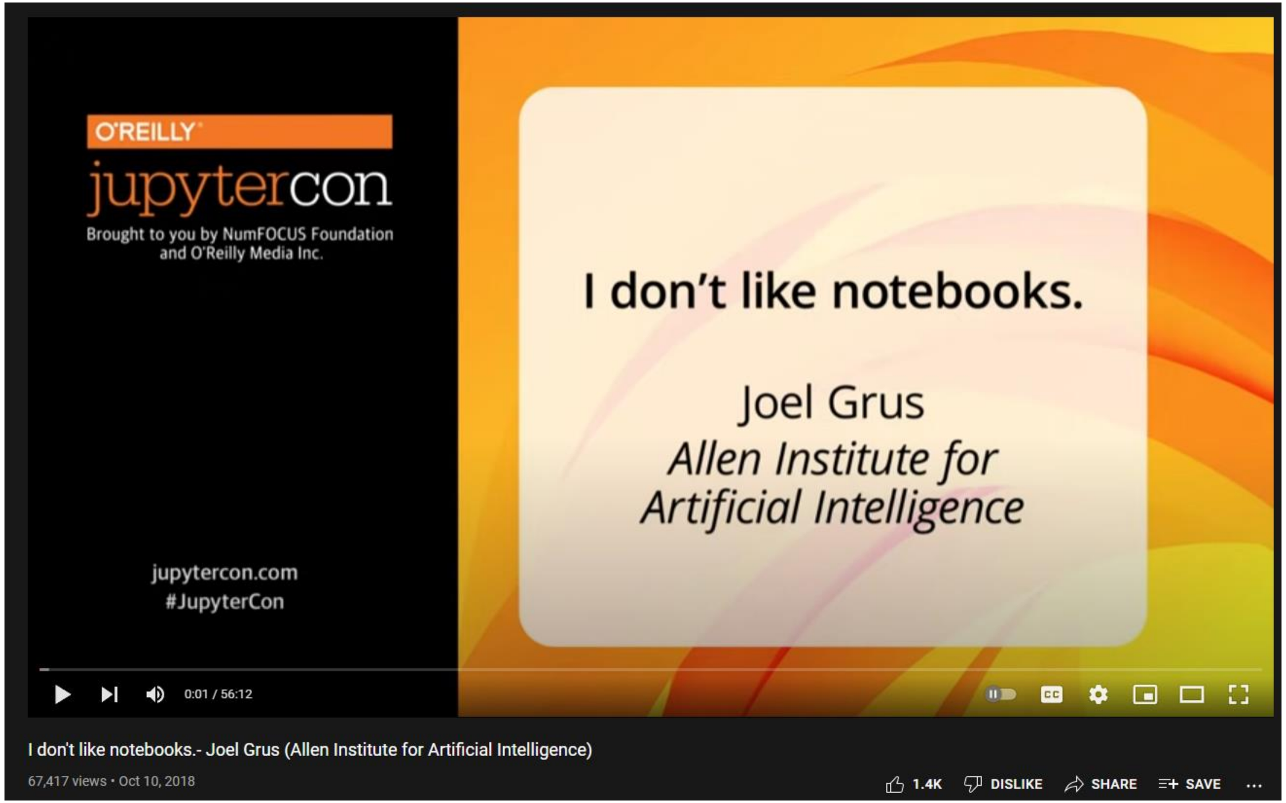
   Advice: Run mature projects in container, separate machines

   Advice: Make an "environment.yml" file

5. The "one cell notebooks"

   Advice: <10 lines of code per cell

More Pitfalls with Notebooks



I don't like notebooks.

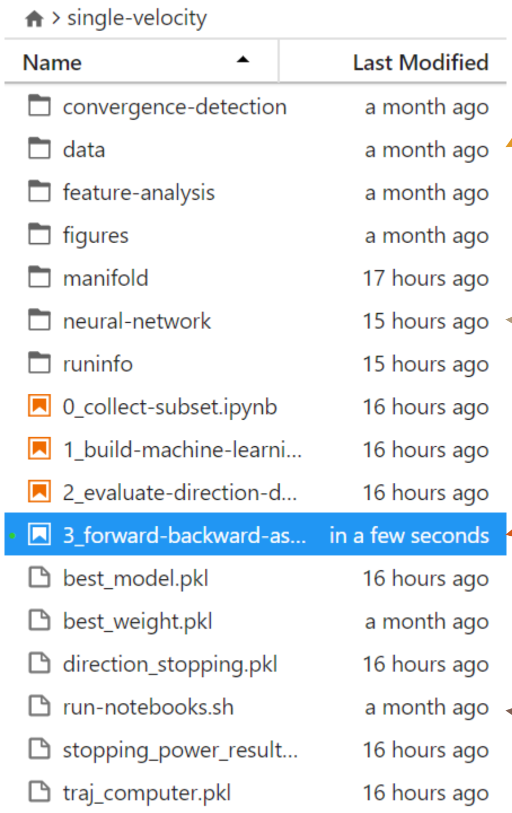
Joel Grus

Allen Institute for Artificial Intelligence

I don't like notebooks.- Joel Grus (Allen Institute for Artificial Intelligence)

67,417 views • Oct 10, 2018

Organizing Multiple Notebooks



⌂ > single-velocity

Name	Last Modified
convergence-detection	a month ago
data	a month ago
feature-analysis	a month ago
figures	a month ago
manifold	17 hours ago
neural-network	15 hours ago
runinfo	15 hours ago
0_collect-subset.ipynb	16 hours ago
1_build-machine-learni...	16 hours ago
2_evaluate-direction-d...	16 hours ago
3_forward-backward-as...	in a few seconds
best_model.pkl	16 hours ago
best_weight.pkl	a month ago
direction_stopping.pkl	16 hours ago
run-notebooks.sh	a month ago
stopping_power_result...	16 hours ago
traj_computer.pkl	16 hours ago

Re-usable datasets in special folder

Figures in special folder (with meaningful names!) (make pub-ready figures!)

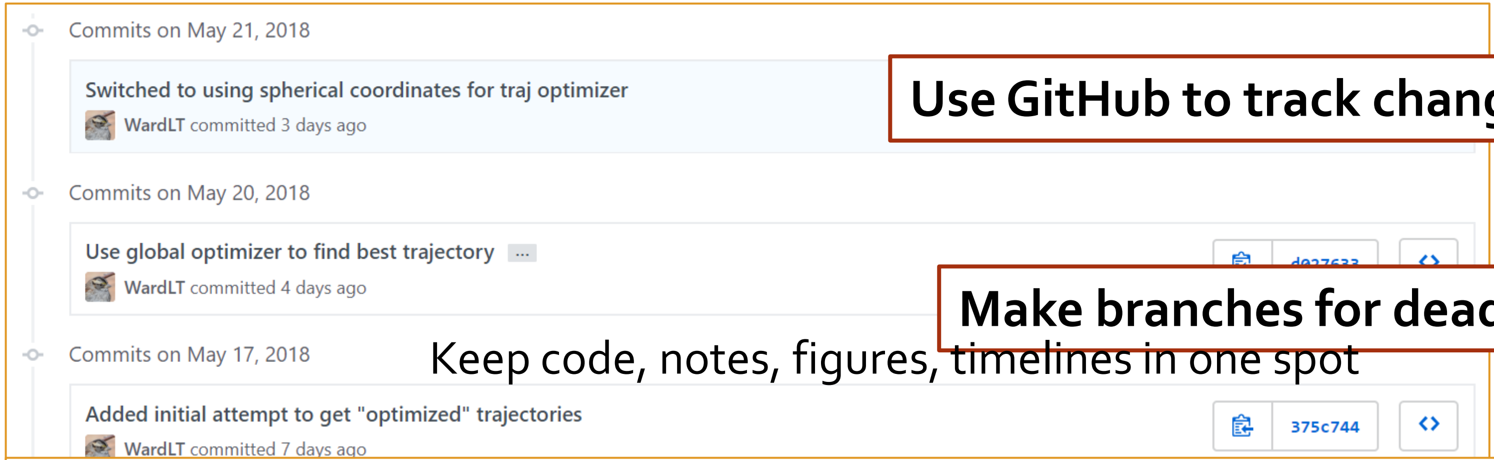
Dependent tasks in subfolders

Notebooks numbered by execution order

Results shared between steps using pickled files

A bash script to re-run all notebooks

Use Github as a Lab Notebooks



Commits on May 21, 2018

Switched to using spherical coordinates for traj optimizer

WardLT committed 3 days ago

Commits on May 20, 2018

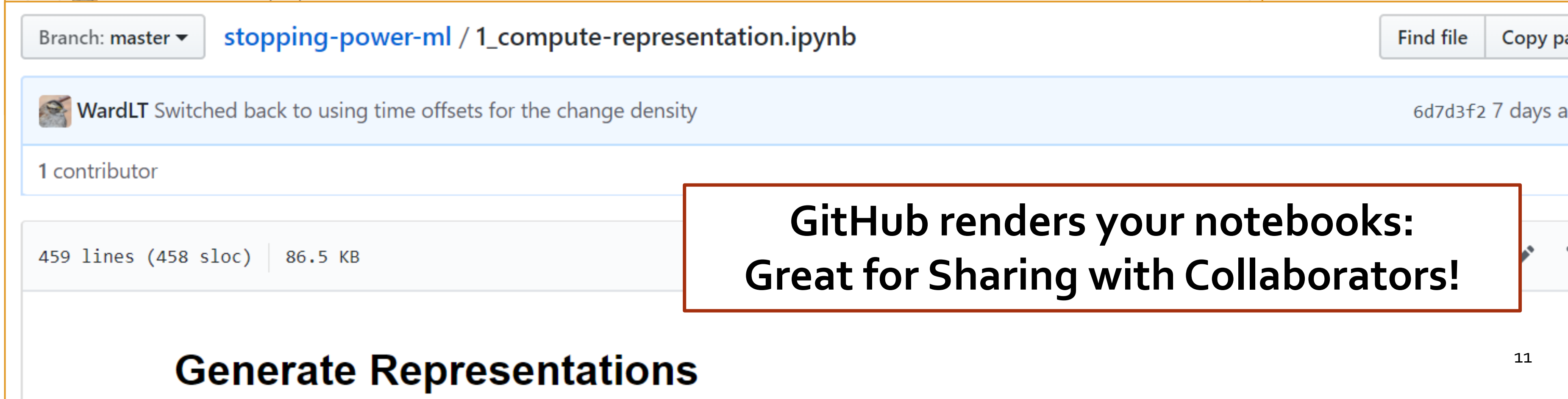
Use global optimizer to find best trajectory …

WardLT committed 4 days ago

Commits on May 17, 2018

Added initial attempt to get "optimized" trajectories

WardLT committed 7 days ago



stopping-power-ml / 1_compute-representation.ipynb

WardLT Switched back to using time offsets for the change density 6d7d3f2 7 days a

1 contributor

459 lines (458 sloc) | 86.5 KB

Generate Representations

Use GitHub to track changes

Make branches for dead-ends

Keep code, notes, figures, timelines in one spot

GitHub renders your notebooks: Great for Sharing with Collaborators!

Takeaway for Using Jupyter

1. **Write your notebooks like papers**

   Explain what you are doing, why, and what you found

   Learn Markdown to include links, equations, pictures

2. **Break up complex projects into multiple steps**

   Each notebook should tell a single story

3. **Notebooks should be easy to re-run**

   Use "Restart and Run," bash scripts

4. **Track your changes with Git**

   Explain what changes you made, and why

ITEM 2: PUBLICATION IS NOT JUST FOR PAPERS!

Is my 2016 paper reproducible?

npj Computational Materials

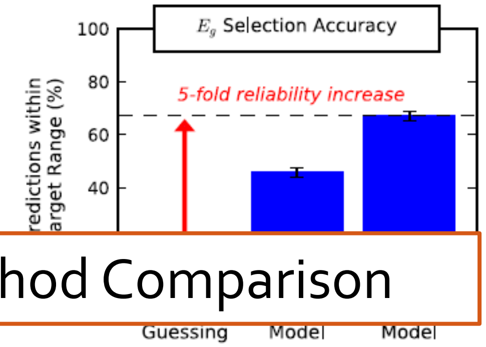
www.nature.com/npjcompumats

ARTICLE OPEN

A general-purpose machine learning framework for predicting properties of inorganic materials

Logan Ward[1], Ankit Agrawal[2], Alok Choudhary[2] and Christopher Wolverton[1]

hat use machin
onstrated succe
impact. To en



Method Comparison

Figure 1. Performance of three different strategies to locate compounds with a band gap energy within a desired range: randomly selecting nonmetal-containing compounds, and two strategies using the machine-learning-based method presented in this work. The first machine learning strategy used a single model trained on the computed band gap energies of 22,667 compounds

Table 2. Compositions and predicted band gap energies of materials predicted using machine learning to be candidates for solar cell applications

Composition	E_g *(eV)*
$ScHg_4Cl_7$	1.26
	1.16
	1.28
	1.11
	1.19

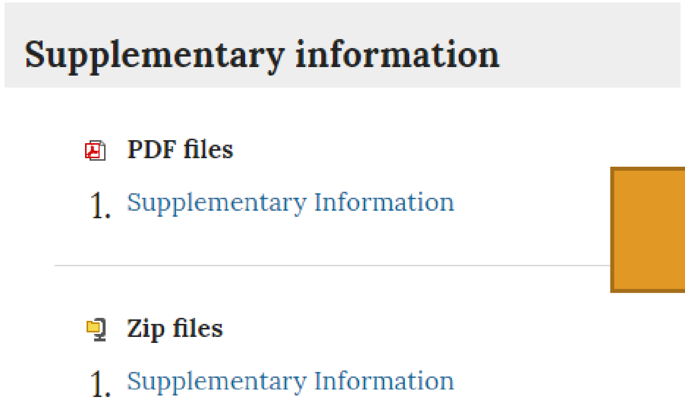
Predicted Materials

Abbreviations: DFT, density functional theory; OQMD, open quantum materials database.
Compositions represent the nominal compositions of novel ternary compounds predicted by using methods developed in ref. 15. Band gap energies were predicted using a machine learning model trained on DFT band gap energies from the OQMD[2] using methods described in this work.

I released the code, but not *ALL of it*

(Spoiler: I regret it!)

From Original Source Code


Supplementary information

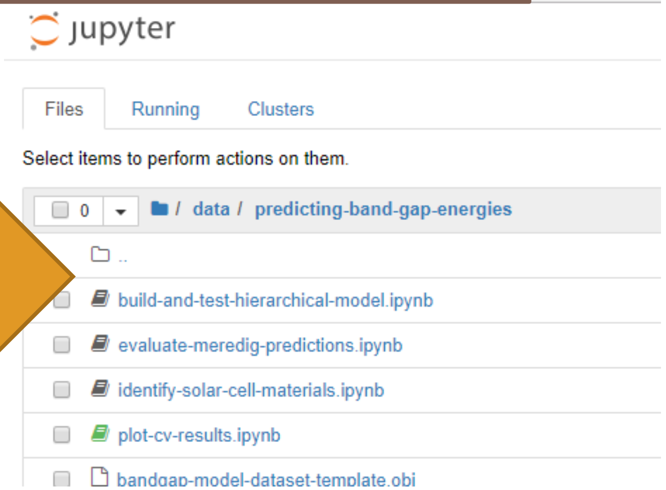
PDF files

1. Supplementary Information

Zip files

1. Supplementary Information

… to New Notebooks


jupyter

Files Running Clusters

Select items to perform actions on them.

0 / data / predicting-band-gap-energies

..

build-and-test-hierarchical-model.ipynb

evaluate-meredig-predictions.ipynb

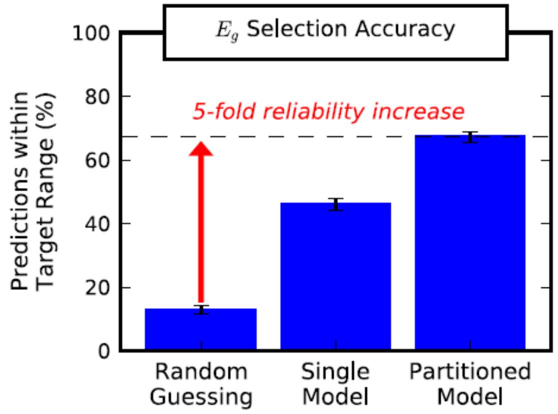
identify-solar-cell-materials.ipynb

plot-cv-results.ipynb

bandgap-model-dataset-template.obj

… and there are differences!

Reported Results (2016)



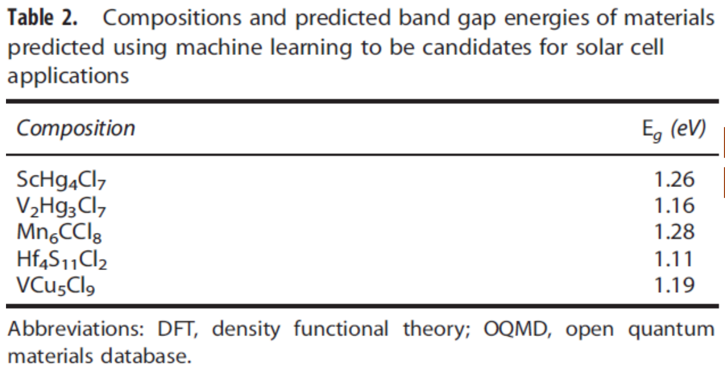
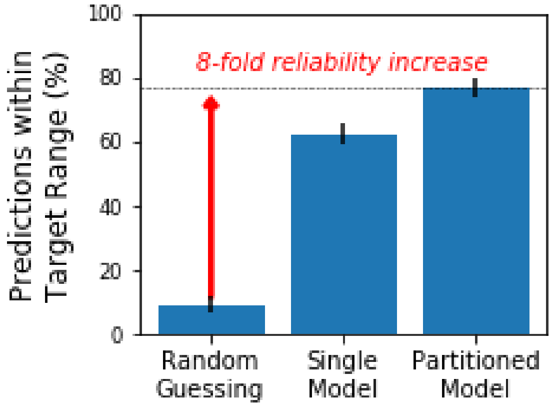


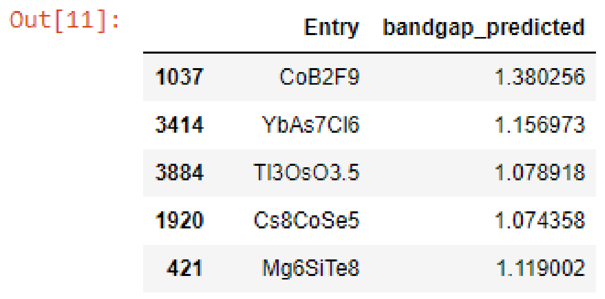
Table 2. Compositions and predicted band gap energies of materials predicted using machine learning to be candidates for solar cell applications

Composition	E_g *(eV)*
$ScHg_4Cl_7$	1.26
$V_2Hg_3Cl_7$	1.16
Mn_6CCl_8	1.28
$Hf_4S_{11}Cl_2$	1.11
VCu_5Cl_9	1.19

Abbreviations: DFT, density functional theory; OQMD, open quantum materials database.

Replication in 2017





Out[11]:

	Entry	bandgap_predicted
1037	CoB2F9	1.380256
3414	YbAs7Cl6	1.156973
3884	Tl3OsO3.5	1.078918
1920	Cs8CoSe5	1.074358
421	Mg6SiTe8	1.119002

Better reproducibility via publication

What do I need to publish?

- *Datasets*: In a well-described format
- *Scripts*: Not just the core methods
- *Outputs:* Exact version from the paper
- *Models:* In a user-friendly way

CITRINE INFORMATICS

GitHub

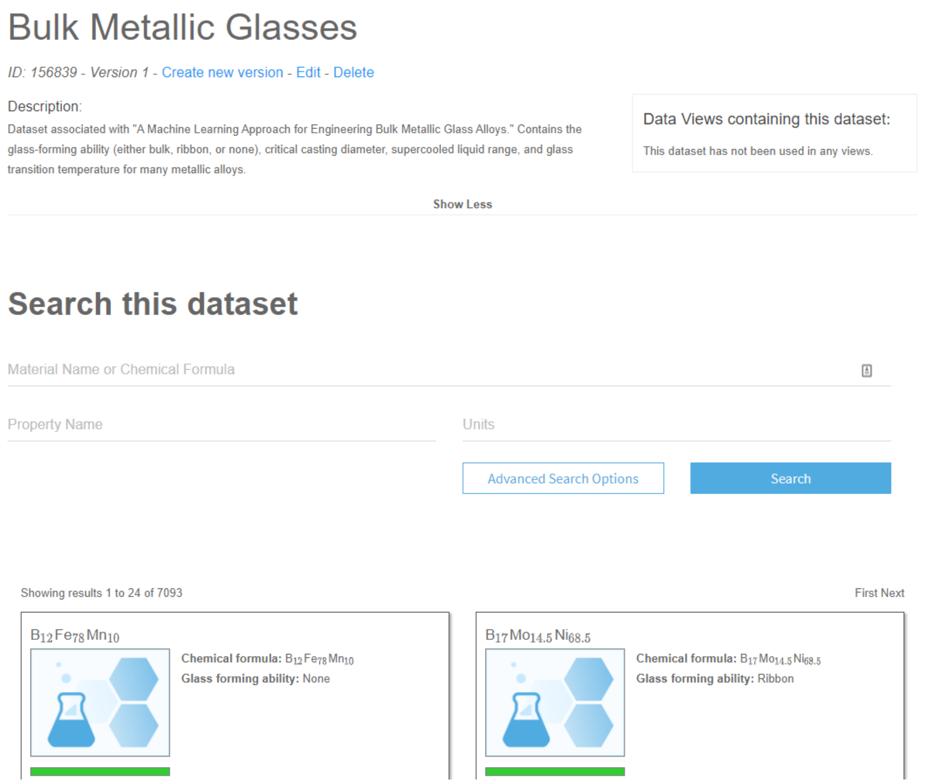
WholeTale

MATERIALS DATA FACILITY

DLHub

Datasets: Citrination

- Or another database that serves *structured data*



Bulk Metallic Glasses

ID: 156839 - Version 1 - Create new version - Edit - Delete

Description:
Dataset associated with "A Machine Learning Approach for Engineering Bulk Metallic Glass Alloys." Contains the glass-forming ability (either bulk, ribbon, or none), critical casting diameter, supercooled liquid range, and glass transition temperature for many metallic alloys.

Data Views containing this dataset:
This dataset has not been used in any views.

Showing results 1 to 24 of 7093

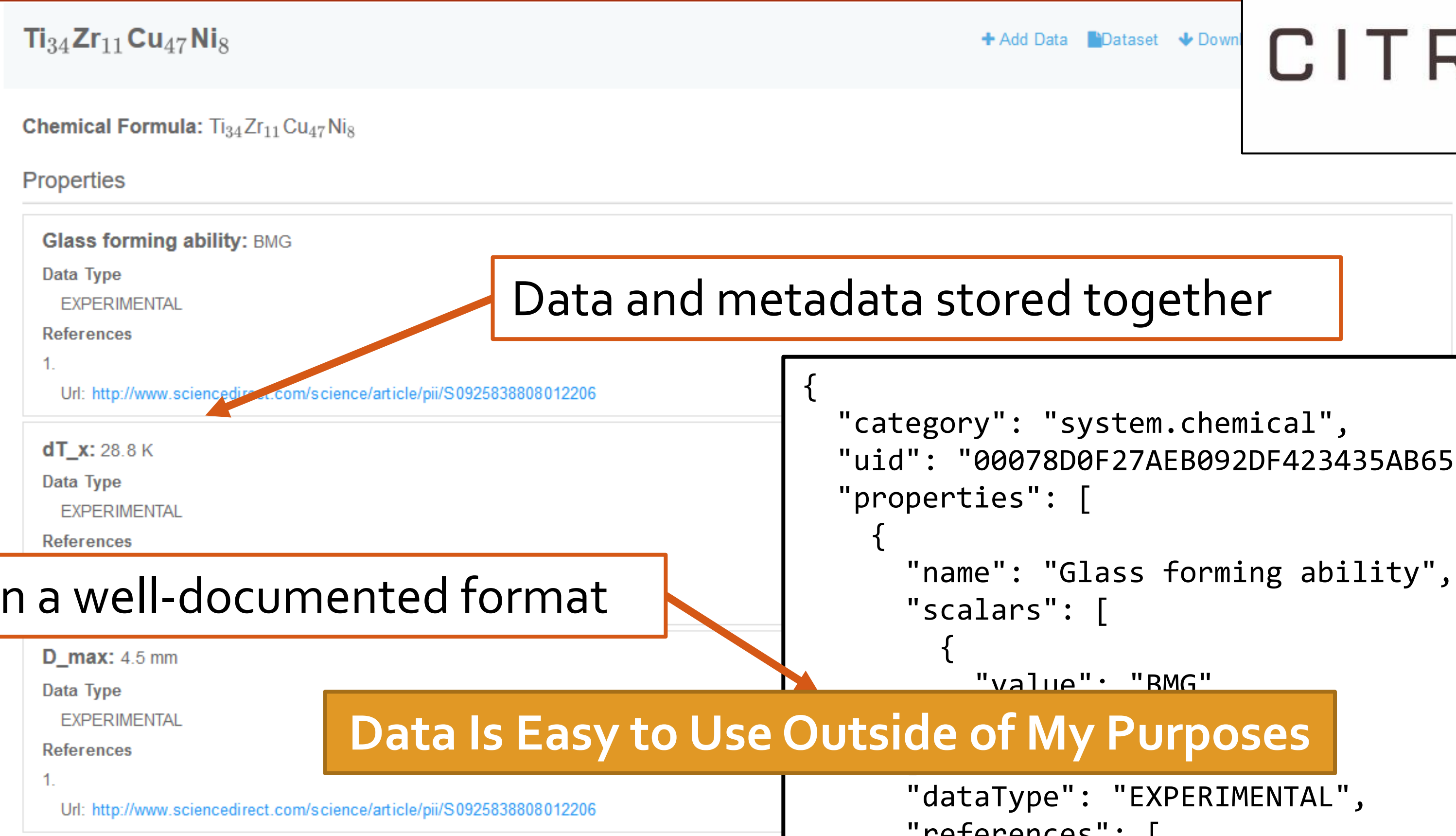
$B_{12}Fe_{78}Mn_{10}$

Chemical formula: $B_{12}Fe_{78}Mn_{10}$
Glass forming ability: None

$B_{17}Mo_{14.5}Ni_{68.5}$

Chemical formula: $B_{17}Mo_{14.5}Ni_{68.5}$
Glass forming ability: Ribbon

Datasets: Citrination



$Ti_{34}Zr_{11}Cu_{47}Ni_8$

Chemical Formula: $Ti_{34}Zr_{11}Cu_{47}Ni_8$

Properties

Glass forming ability: BMG

Data Type

EXPERIMENTAL

References

1\.

Url: http://www.sciencedirect.com/science/article/pii/S0925838808012206

dT_x: 28.8 K

Data Type

EXPERIMENTAL

References

D_max: 4.5 mm

Data Type

EXPERIMENTAL

References

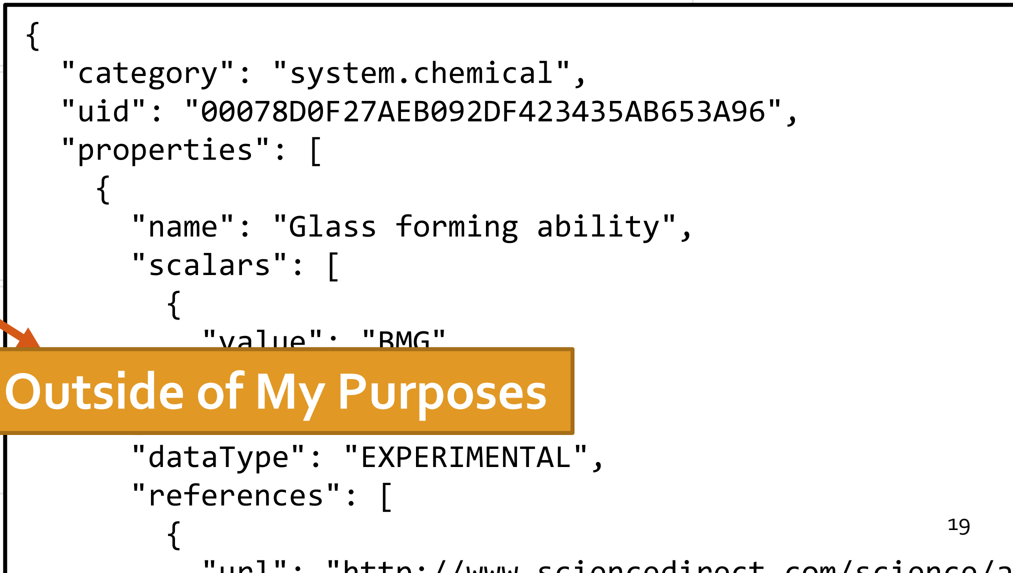
1\.

Url: http://www.sciencedirect.com/science/article/pii/S0925838808012206

CITRINE INFORMATICS

Data and metadata stored together

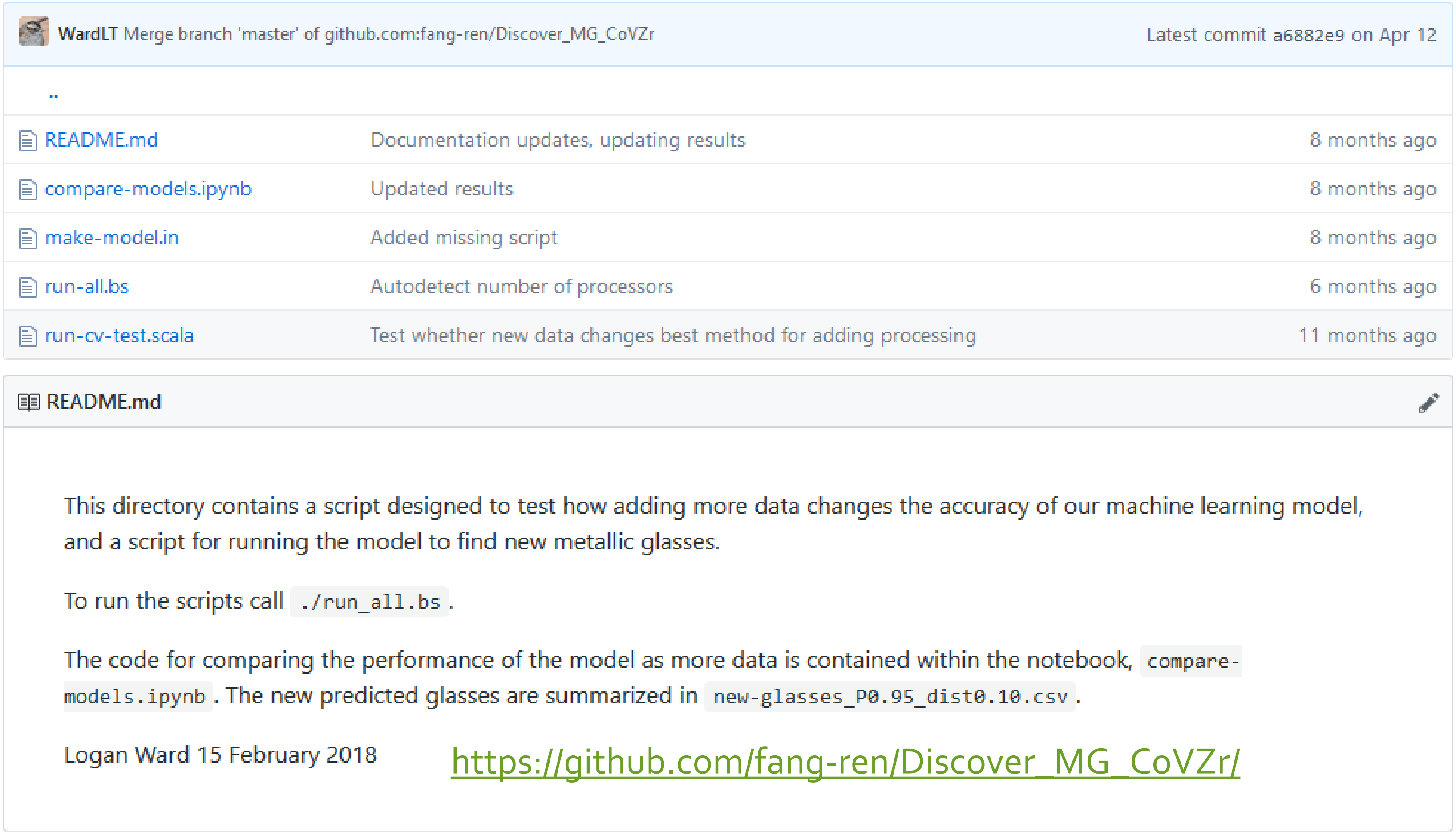
In a well-documented format



```
{
  "category": "system.chemical",
  "uid": "00078D0F27AEB092DF423435AB653A96",
  "properties": [
    {
      "name": "Glass forming ability",
      "scalars": [
        {
          "value": "BMG"

      "dataType": "EXPERIMENTAL",
      "references": [
        {
          "url": "http://www.sciencedirect.com/science/a
```

Data Is Easy to Use Outside of My Purposes

Scripts GitHub and WholeTale



WardLT Merge branch 'master' of github.com:fang-ren/Discover_MG_CoVZr Latest commit a6882e9 on Apr 12

..

README.md	Documentation updates, updating results	8 months ago
compare-models.ipynb	Updated results	8 months ago
make-model.in	Added missing script	8 months ago
run-all.bs	Autodetect number of processors	6 months ago
run-cv-test.scala	Test whether new data changes best method for adding processing	11 months ago

README.md

This directory contains a script designed to test how adding more data changes the accuracy of our machine learning model, and a script for running the model to find new metallic glasses.

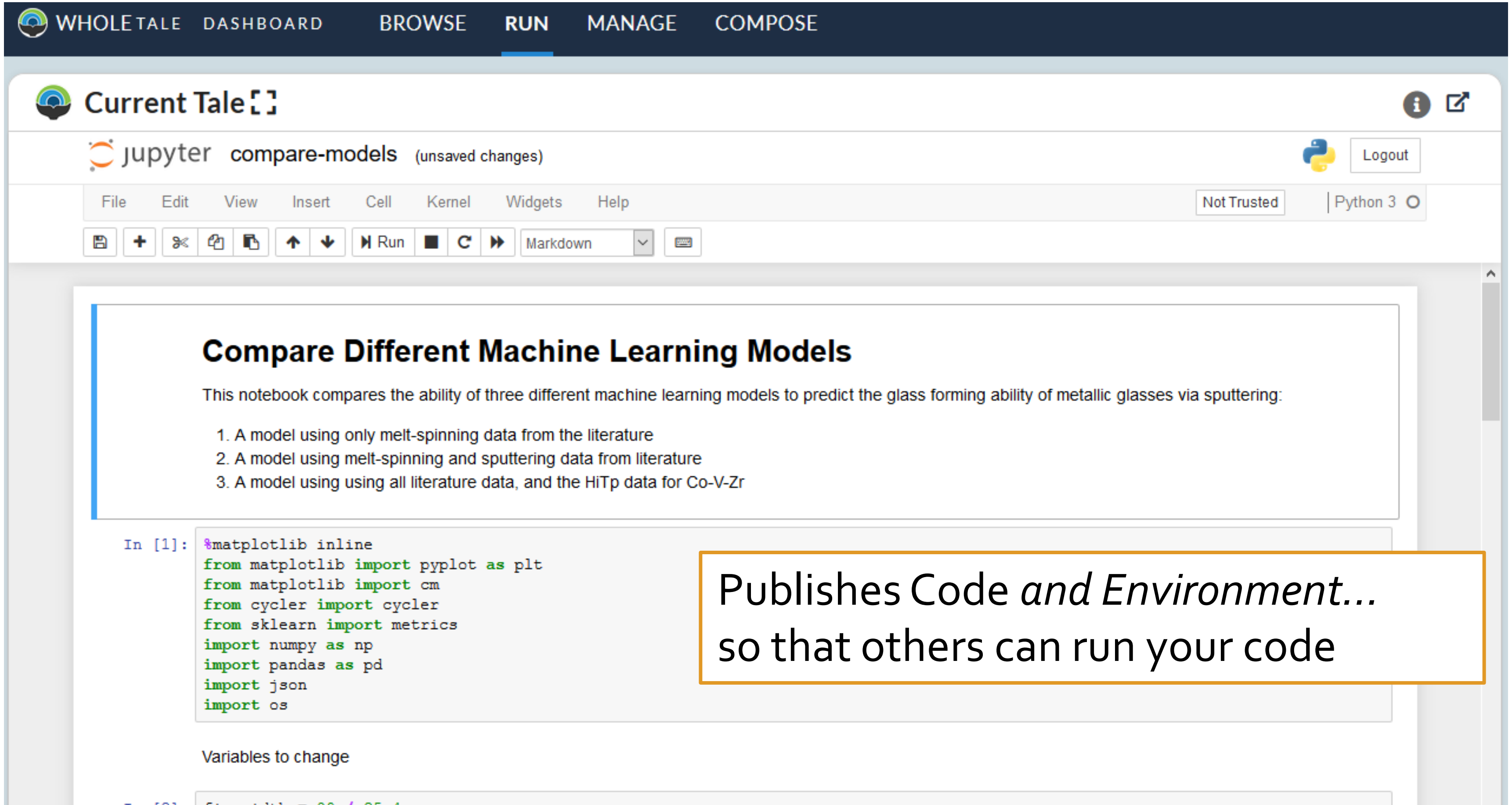
To run the scripts call `./run_all.bs`.

The code for comparing the performance of the model as more data is contained within the notebook, `compare-models.ipynb`. The new predicted glasses are summarized in `new-glasses_P0.95_dist0.10.csv`.

Logan Ward 15 February 2018

https://github.com/fang-ren/Discover_MG_CoVZr/

Scripts: WholeTale



Compare Different Machine Learning Models

This notebook compares the ability of three different machine learning models to predict the glass forming ability of metallic glasses via sputtering:

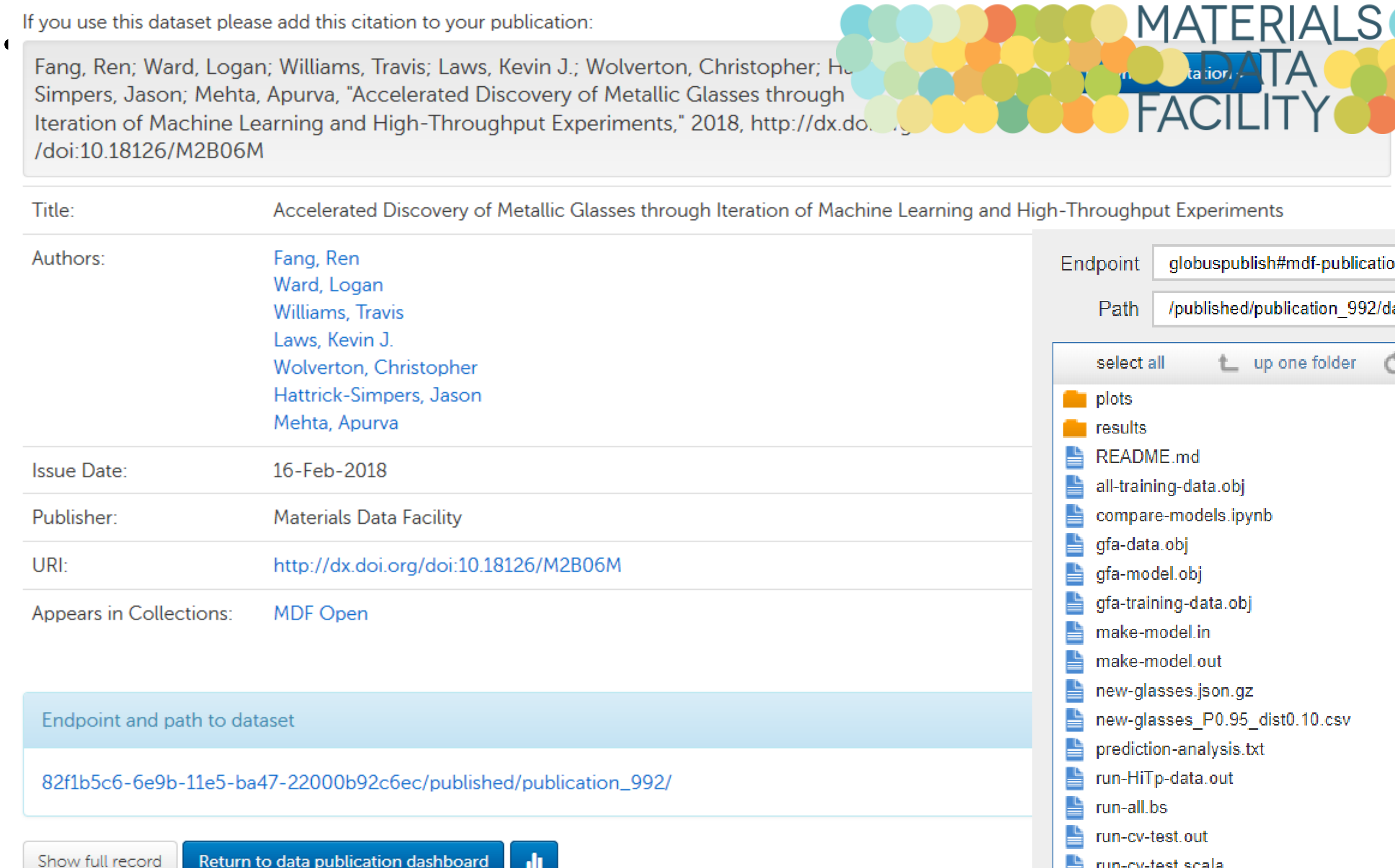
1. A model using only melt-spinning data from the literature
2. A model using melt-spinning and sputtering data from literature
3. A model using using all literature data, and the HiTp data for Co-V-Zr

```
In [1]: %matplotlib inline
        from matplotlib import pyplot as plt
        from matplotlib import cm
        from cycler import cycler
        from sklearn import metrics
        import numpy as np
        import pandas as pd
        import json
        import os
```

Variables to change

Publishes Code *and Environment*…
so that others can run your code

Data: Materials Data Facility



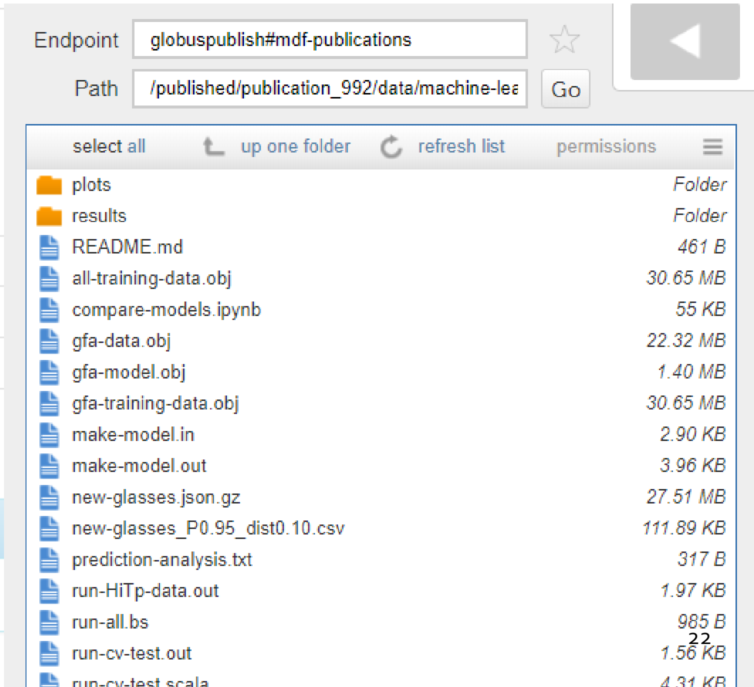
If you use this dataset please add this citation to your publication:

Fang, Ren; Ward, Logan; Williams, Travis; Laws, Kevin J.; Wolverton, Christopher; H
Simpers, Jason; Mehta, Apurva, "Accelerated Discovery of Metallic Glasses through
Iteration of Machine Learning and High-Throughput Experiments," 2018, http://dx.do
/doi:10.18126/M2B06M

Title:	Accelerated Discovery of Metallic Glasses through Iteration of Machine Learning and High-Throughput Experiments
Authors:	Fang, Ren Ward, Logan Williams, Travis Laws, Kevin J. Wolverton, Christopher Hattrick-Simpers, Jason Mehta, Apurva
Issue Date:	16-Feb-2018
Publisher:	Materials Data Facility
URI:	http://dx.doi.org/doi:10.18126/M2B06M
Appears in Collections:	MDF Open

Endpoint and path to dataset

82f1b5c6-6e9b-11e5-ba47-22000b92c6ec/published/publication_992/

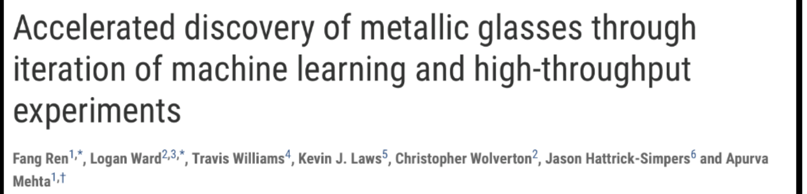


Endpoint: globuspublish#mdf-publications

Path: /published/publication_992/data/machine-lea

plots	Folder
results	Folder
README.md	461 B
all-training-data.obj	30.65 MB
compare-models.ipynb	55 KB
gfa-data.obj	22.32 MB
gfa-model.obj	1.40 MB
gfa-training-data.obj	30.65 MB
make-model.in	2.90 KB
make-model.out	3.96 KB
new-glasses.json.gz	27.51 MB
new-glasses_P0.95_dist0.10.csv	111.89 KB
prediction-analysis.txt	317 B
run-HiTp-data.out	1.97 KB
run-all.bs	985 B
run-cv-test.out	1.56 KB
run-cv-test.scala	4.31 KB

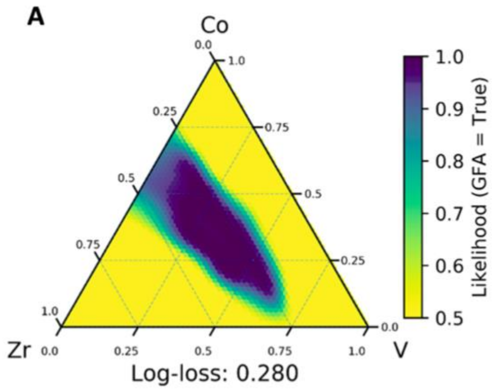
Models: DLHub



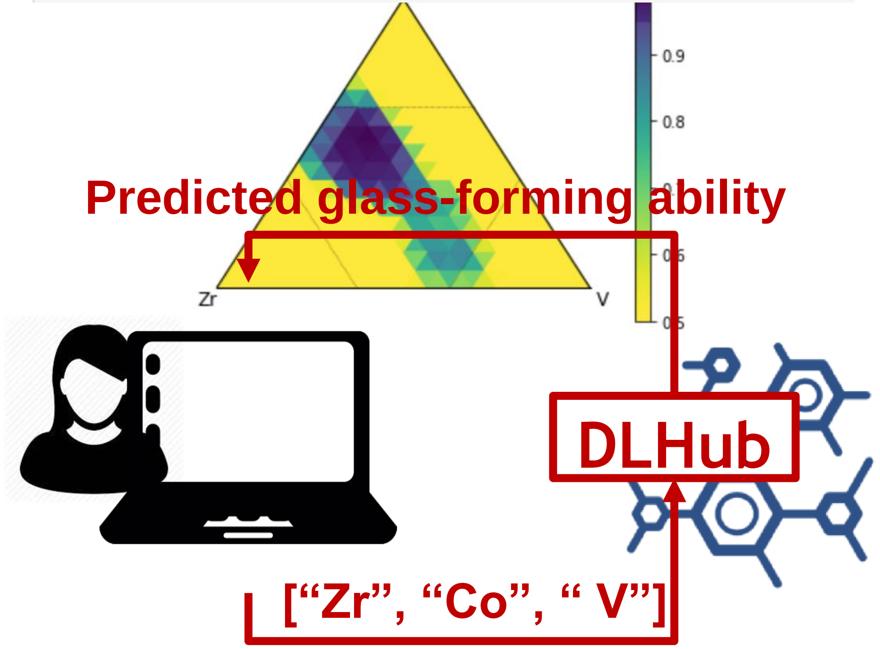
Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments

Fang Ren[1,*], Logan Ward[2,3,*], Travis Williams[4], Kevin J. Laws[5], Christopher Wolverton[2], Jason Hattrick-Simpers[6] and Apurva Mehta[1,†]

10.1126/sciadv.aaq1566

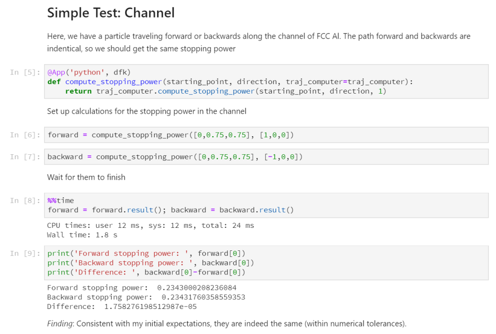


```
servable_name = "metallic_glass"
servable_id = dl.get_id_by_name(servable_name)
elems = ["V","Co","Zr"]

res = dl.run(servable_id, {"data":elems})
```



Overview of Today

1. **Write notebooks like papers**
2. **Publish everything with your papers**



Simple Test: Channel

Here, we have a particle traveling forward or backwards along the channel of FCC Al. The path forward and backwards are indentical, so we should get the same stopping power

```
In [5]: @App('python', dfk)
        def compute_stopping_power(starting_point, direction, traj_computer=traj_computer):
            return traj_computer.compute_stopping_power(starting_point, direction, 1)
```

Set up calculations for the stopping power in the channel

```
In [6]: forward = compute_stopping_power([0,0.75,0.75], [1,0,0])
```

```
In [7]: backward = compute_stopping_power([0,0.75,0.75], [-1,0,0])
```

Wait for them to finish

```
In [8]: %%time
        forward = forward.result(); backward = backward.result()

CPU times: user 12 ms, sys: 12 ms, total: 24 ms
Wall time: 1.8 s
```

```
In [9]: print('Forward stopping power: ', forward[0])
        print('Backward stopping power: ', backward[0])
        print('Difference: ', backward[0]-forward[0])

Forward stopping power:  0.2343000208236084
Backward stopping power:  0.23431760358559353
Difference:  1.758276198512987e-05
```

Finding: Consistent with my initial expectations, they are indeed the same (within numerical tolerances).

Advice:

- Work in Jupyter until you're comfortable
- Read/watch "I don't like Jupyter notebooks"
- Learn Markdown