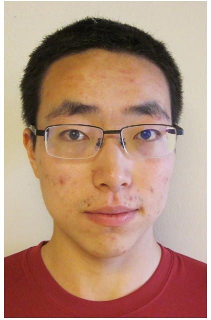
XYZ studies

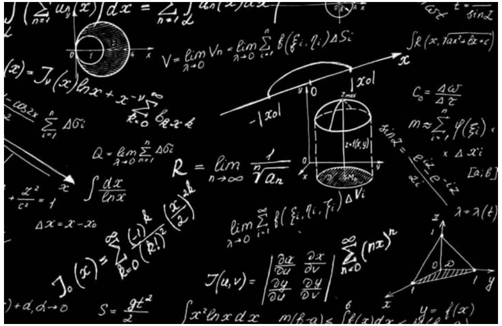
a paradigm for research in data science

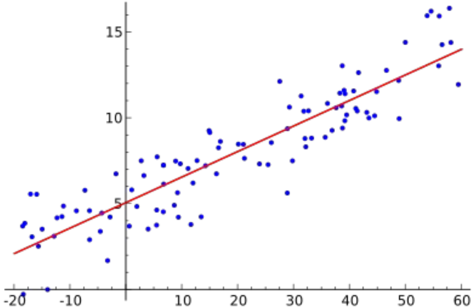
Vardan Papyan

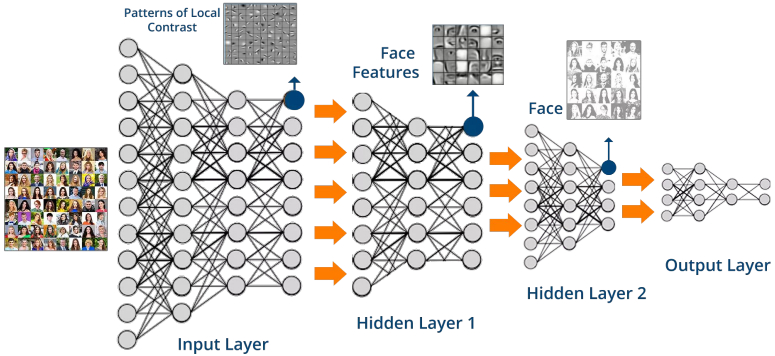
Nobody knows what data science is

Statistics:





Machine learning:



We are proposing to show you what data science is…

XYZ studies

Y — datasets considered canonical for certain task

X — all relevant methods

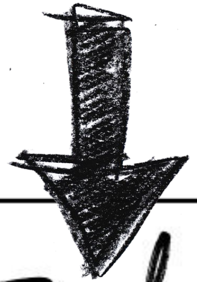
Z — control parameters

W — observables of interest

Algorithm 1: Description of XYZ experiment

Input : methods X, datasets Y, control parameters Z
Output: observables W

```
foreach method x ∈ X do
    foreach dataset y ∈ Y do
        foreach control parameter z ∈ Z do
            /* run experiment and collect observables */
            W(x, y, z) = Experiment(x, y, z)
        end
    end
end
```



Finding

Navigating in the XYZ space

- Python
- Dedicated software: Tableau (more on this later)
- Online website: D3

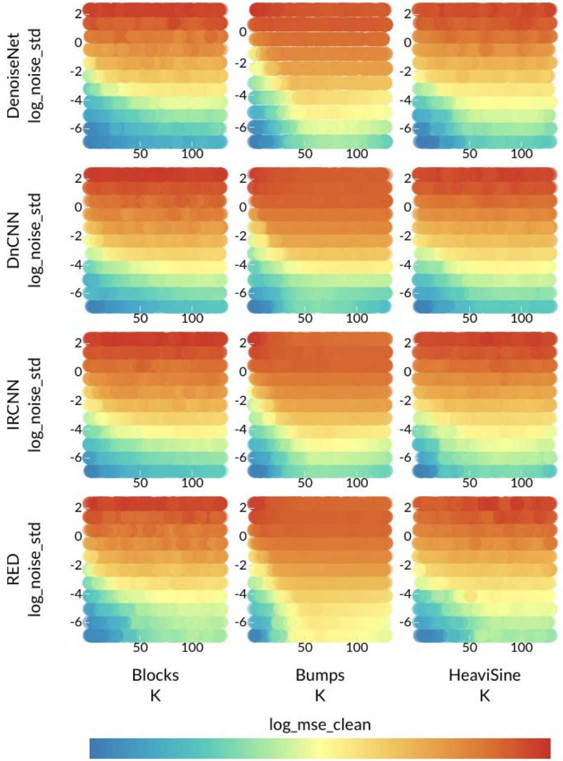
Change plot size: - +

Change circle size: - +

Choose control parameters Z or observables W:

V1:	V2:	V3:
K	K (selected)	K
log_K	log_K	log_K
path_sparsity	path_sparsity	path_sparsity
log_path_sparsity	log_path_sparsity	log_path_sparsity
noise_std	noise_std	noise_std
log_noise_std (selected)	log_noise_std	log_noise_std
mse_noisy	mse_noisy	mse_noisy
log_mse_noisy	log_mse_noisy	log_mse_noisy
mse_clean	mse_clean	mse_clean
log_mse_clean	log_mse_clean	log_mse_clean (selected)
mse_clean_div_noise	mse_clean_div_noise	mse_clean_div_noise
DF_mc_tr	DF_mc_tr	DF_mc_tr
log_DF_mc_tr	log_DF_mc_tr	log_DF_mc_tr
clean_sub_noisy	clean_sub_noisy	clean_sub_noisy
log_clean_sub_noisy	log_clean_sub_noisy	log_clean_sub_noisy
bias	bias	bias
log_bias	log_bias	log_bias
SURE	SURE	SURE
log_SURE	log_SURE	log_SURE
SURE_div_noise	SURE_div_noise	SURE_div_noise

For each method X and dataset Y, V1 is plotted against V2 and colored with V3.



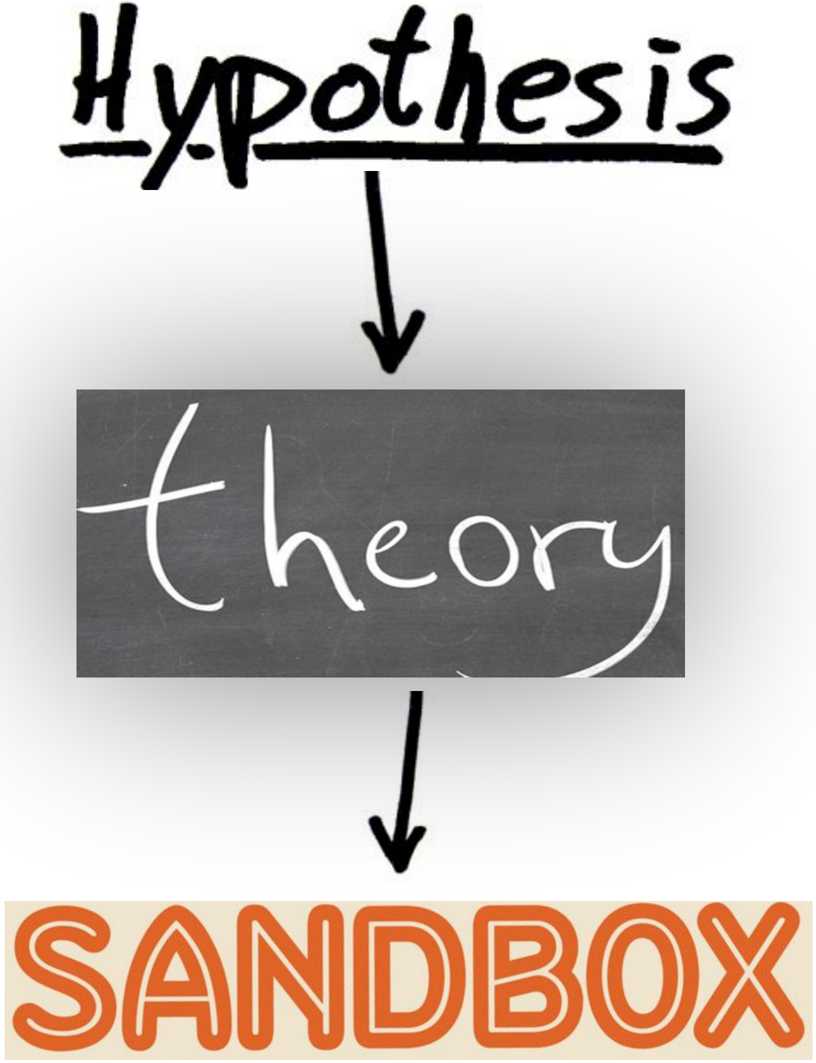
to pdf

download reproducible code

download models

download xyz array

add data


Hypothesis
theory
SANDBOX

If you want to be a data scientist…

Follow the paradigm

Work that way

Create tools that work that way

Evaluate other people’s work that way

This is what data science is about

Data science v.s STATS and ML

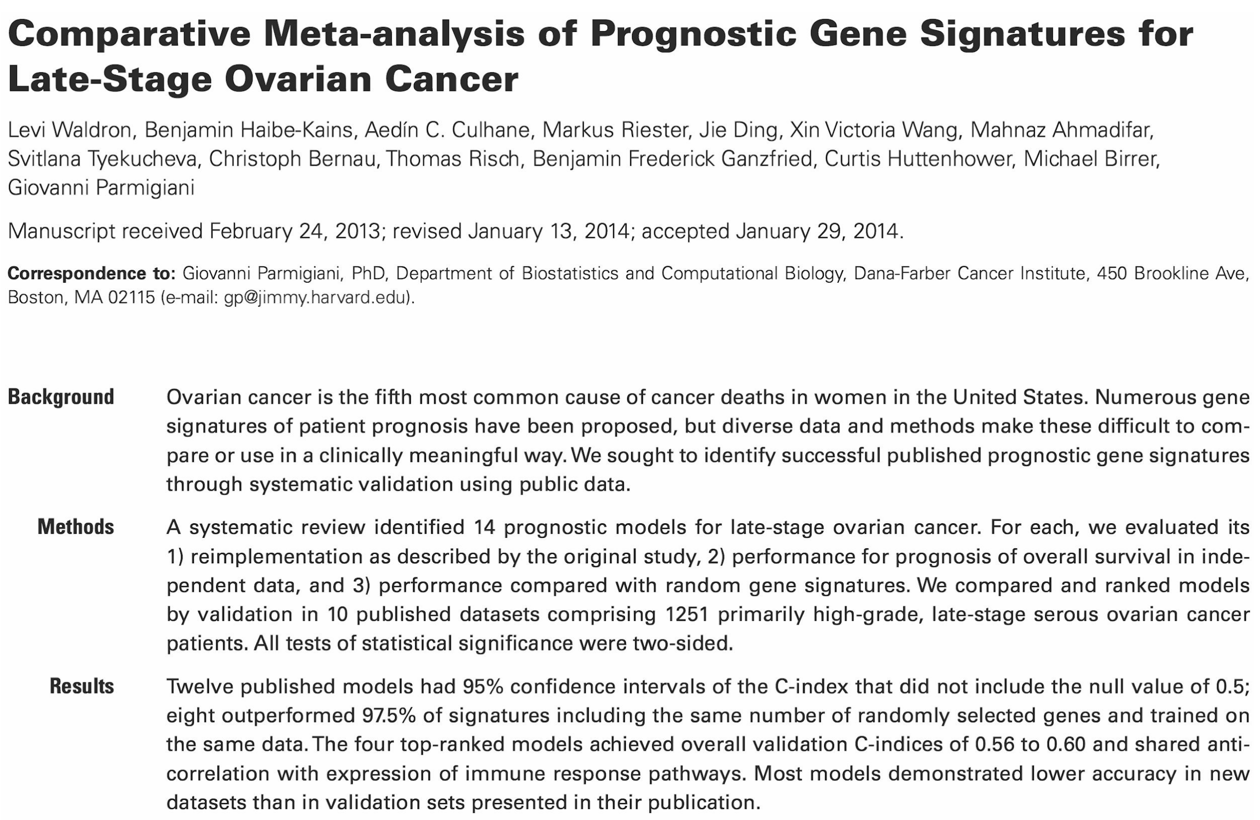
Statistics:
- Too much math
- Over simplified generative models
- Evading the truth

Staying true to phenomenons seen in practice

Machine learning:
- Uninformative predictive models
- Too quick to jump to conclusions
- Too much reliance on poetry
- Sees the truth through the pinhole of a single method-dataset

Comprehensive experimentation

People are groping for this


Comparative Meta-analysis of Prognostic Gene Signatures for Late-Stage Ovarian Cancer

Levi Waldron, Benjamin Haibe-Kains, Aedín C. Culhane, Markus Riester, Jie Ding, Xin Victoria Wang, Mahnaz Ahmadifar, Svitlana Tyekucheva, Christoph Bernau, Thomas Risch, Benjamin Frederick Ganzfried, Curtis Huttenhower, Michael Birrer, Giovanni Parmigiani

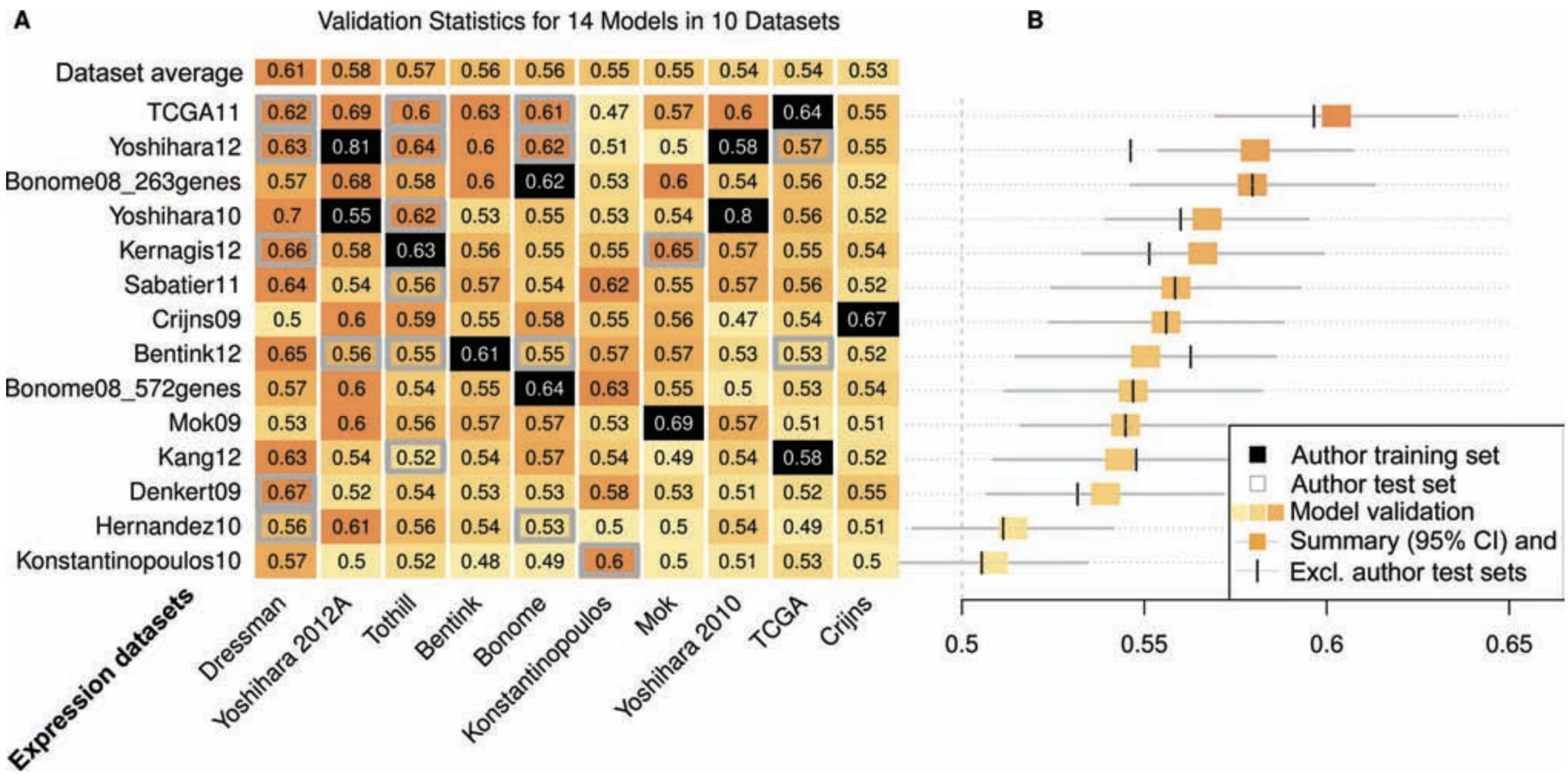
Manuscript received February 24, 2013; revised January 13, 2014; accepted January 29, 2014.

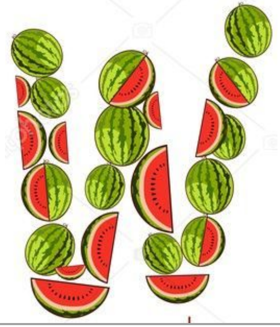
Correspondence to: Giovanni Parmigiani, PhD, Department of Biostatistics and Computational Biology, Dana-Farber Cancer Institute, 450 Brookline Ave, Boston, MA 02115 (e-mail: gp@jimmy.harvard.edu).

Background Ovarian cancer is the fifth most common cause of cancer deaths in women in the United States. Numerous gene signatures of patient prognosis have been proposed, but diverse data and methods make these difficult to compare or use in a clinically meaningful way. We sought to identify successful published prognostic gene signatures through systematic validation using public data.

Methods A systematic review identified 14 prognostic models for late-stage ovarian cancer. For each, we evaluated its 1) reimplementation as described by the original study, 2) performance for prognosis of overall survival in independent data, and 3) performance compared with random gene signatures. We compared and ranked models by validation in 10 published datasets comprising 1251 primarily high-grade, late-stage serous ovarian cancer patients. All tests of statistical significance were two-sided.

Results Twelve published models had 95% confidence intervals of the C-index that did not include the null value of 0.5; eight outperformed 97.5% of signatures including the same number of randomly selected genes and trained on the same data. The four top-ranked models achieved overall validation C-indices of 0.56 to 0.60 and shared anti-correlation with expression of immune response pathways. Most models demonstrated lower accuracy in new datasets than in validation sets presented in their publication.


A
Validation Statistics for 14 Models in 10 Datasets
B
Dataset average 0.61 0.58 0.57 0.56 0.56 0.55 0.55 0.54 0.54 0.53
TCGA11 0.62 0.69 0.6 0.63 0.61 0.47 0.57 0.6 0.64 0.55
Yoshihara12 0.63 0.81 0.64 0.6 0.62 0.51 0.5 0.58 0.57 0.55
Bonome08_263genes 0.57 0.68 0.58 0.6 0.62 0.53 0.6 0.54 0.56 0.52
Yoshihara10 0.7 0.55 0.62 0.53 0.55 0.53 0.54 0.8 0.56 0.52
Kernagis12 0.66 0.58 0.63 0.56 0.55 0.55 0.65 0.57 0.55 0.54
Sabatier11 0.64 0.54 0.56 0.57 0.54 0.62 0.55 0.57 0.56 0.52
Crijns09 0.5 0.6 0.59 0.55 0.58 0.55 0.56 0.47 0.54 0.67
Bentink12 0.65 0.56 0.55 0.61 0.55 0.57 0.57 0.53 0.53 0.52
Bonome08_572genes 0.57 0.6 0.54 0.55 0.64 0.63 0.55 0.5 0.53 0.54
Mok09 0.53 0.6 0.56 0.57 0.57 0.53 0.69 0.57 0.51 0.51
Kang12 0.63 0.54 0.52 0.54 0.57 0.54 0.49 0.54 0.58 0.52
Denkert09 0.67 0.52 0.54 0.53 0.53 0.58 0.53 0.51 0.52 0.55
Hernandez10 0.56 0.61 0.56 0.54 0.53 0.5 0.5 0.54 0.49 0.51
Konstantinopoulos10 0.57 0.5 0.52 0.48 0.49 0.6 0.5 0.51 0.53 0.5
Expression datasets
Dressman
Yoshihara 2012A
Tothill
Bentink
Bonome
Konstantinopoulos
Mok
Yoshihara 2010
TCGA
Crijns
Author training set
Author test set
Model validation
Summary (95% CI) and
Excl. author test sets
0.5
0.55
0.6
0.65

					
Rethinking Generalization by Zhang et. al	CIFAR10, ImageNet	MLP, AlexNet, Inception	% randomized labels	number of epochs until perfect fit, test error at epoch of perfect fit	Could be done on more datasets and methods

Understanding deep learning requires rethinking generalization

https://arxiv.org › cs

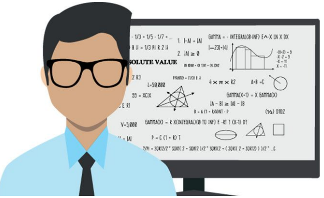
by C Zhang - 2016 - Cited by 303 - Related articles

Perfect score on the ICLR reviews

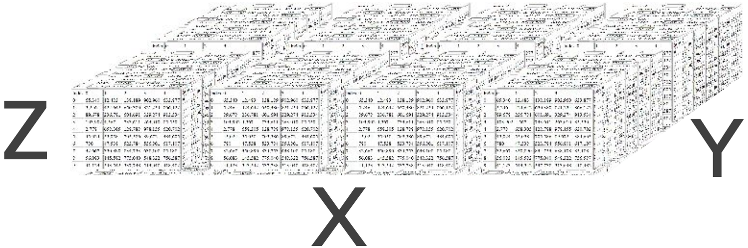
ICLR 2017 best paper award

OCT 13, 2017 @ 01:23 PM 7,420

What You Need To Know About One Of The Most Talked-About Papers On Deep Learning To Date



CFG



Public Cloud Providers

Microsoft Azure

IBM Cloud

Google Cloud Platform

amazon web services

hadoop

CodaLab

APACHE Spark

ElastiCluster

Pywren

Caffe

Chainer

DL4J Deeplearning4j

KERAS

Microsoft CNTK

MatConvNet

MINERVA

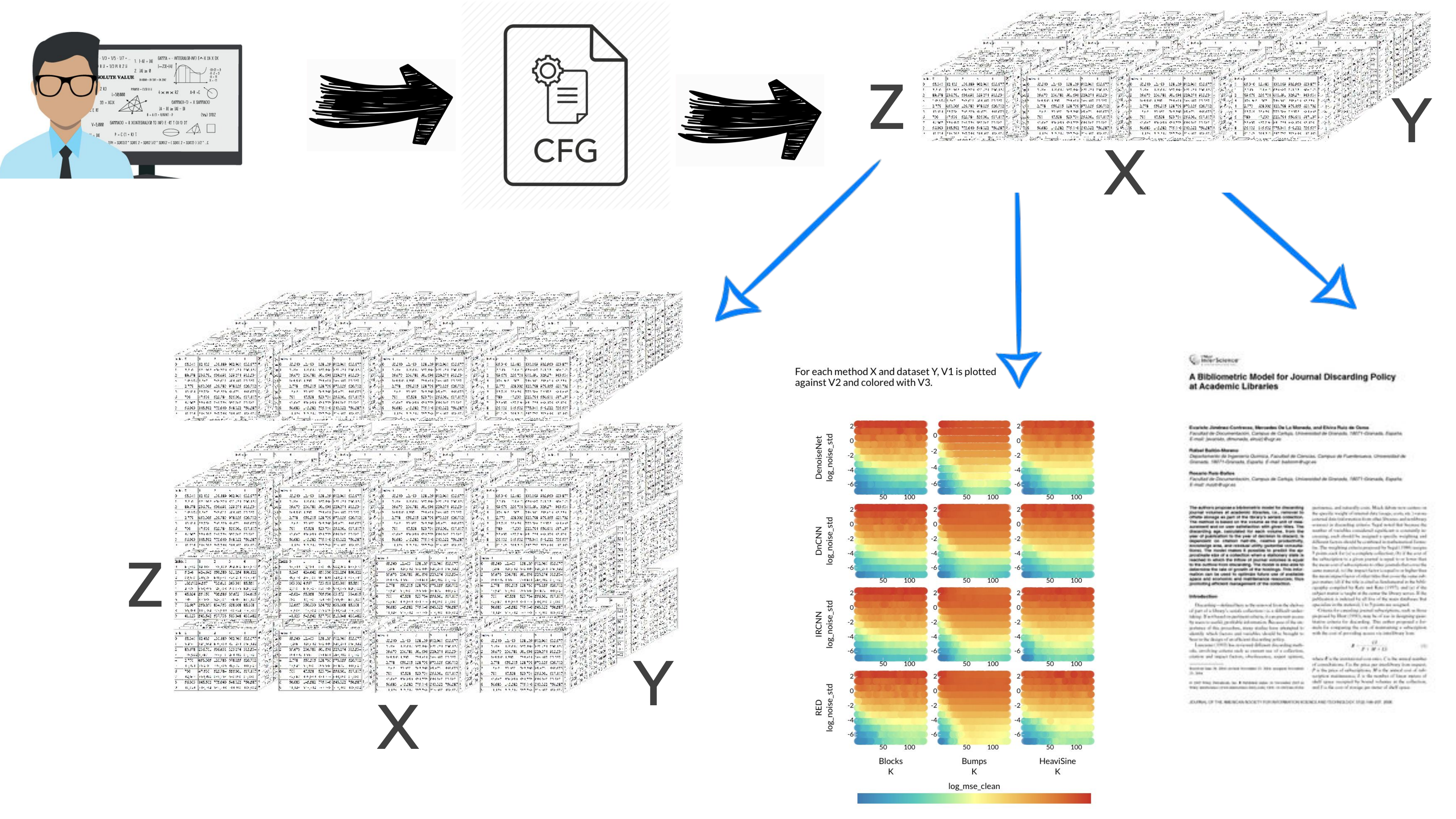
mxnet

Purine

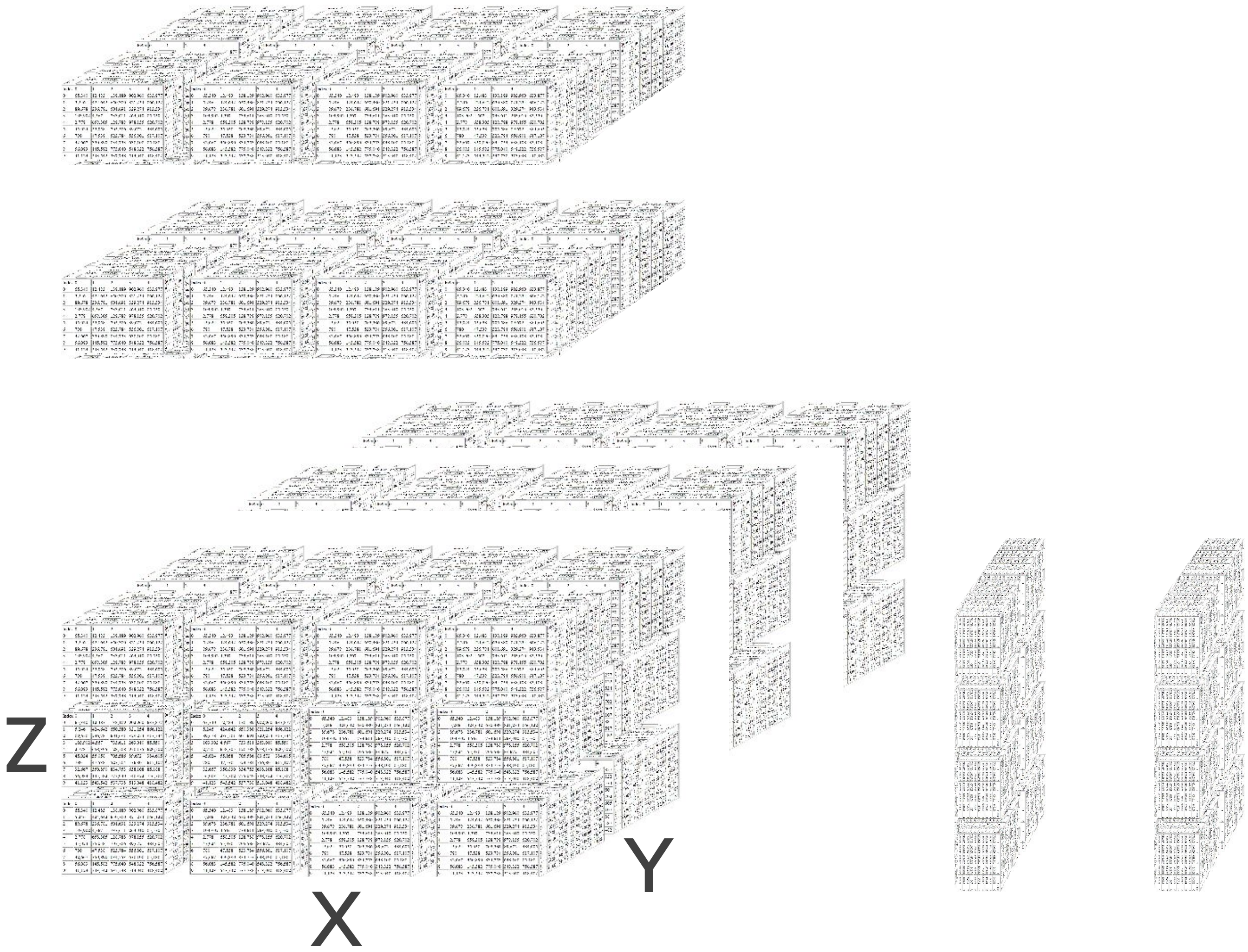
TensorFlow

theano

torch



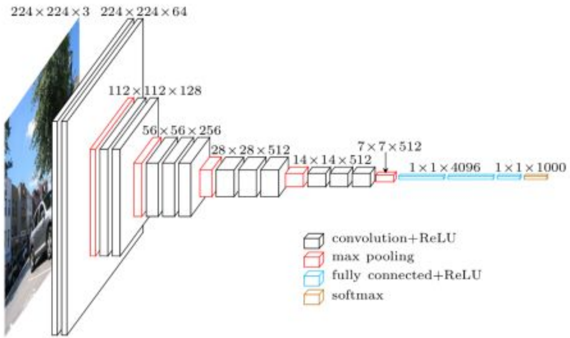
CFG
Z
X
Y
Z
X
Y
For each method X and dataset Y, V1 is plotted against V2 and colored with V3.
DenoiseNet log_noise_std
DnCNN log_noise_std
IRCNN log_noise_std
RED log_noise_std
Blocks K
Bumps K
HeaviSine K
log_mse_clean
A Bibliometric Model for Journal Discarding Policy at Academic Libraries

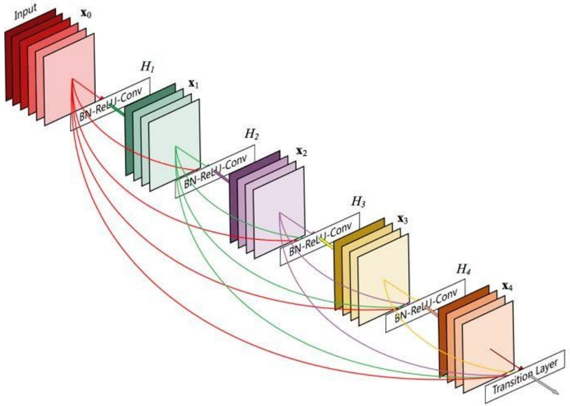

Z
X
Y

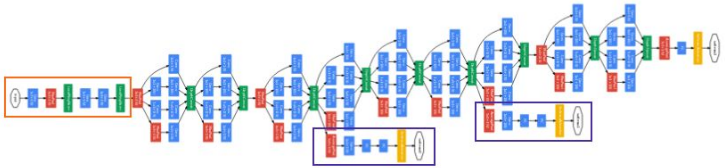
Personal

XYZ

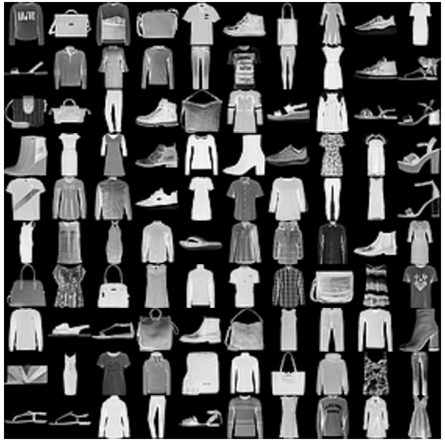
```
net_list          = [
                      'CNN',
                      'AlexNet',
                      'VGG11_bn',
                      'VGG13_bn',
                      'VGG16_bn',
                      'VGG19_bn',
                      'ResNet18',
                      'ResNet34',
                      'ResNet50',
                      'ResNet101',
                      'ResNet152',
                      'SqueezeNet_1_0',
                      'SqueezeNet_1_1',
                      'DenseNet121',
                      'DenseNet161',
                      'DenseNet169',
                      'DenseNet201',
                      'Inception3'
                    ]
```







```
dataset_list    = [
                    'MNIST',
                    'FashionMNIST',
                    'EMNIST_byclass',
                    'EMNIST_bymerge',
                    'EMNIST_balanced',
                    'EMNIST_letters',
                    'EMNIST_digits',
                    'CIFAR10',
                    'CIFAR100',
                    'STL10',
                    'SVHN',
                  ]
```



```
lr_list          = [
                    0.5,
                    0.25,
                    0.1,
                    0.075,
                    0.05,
                    0.025,
                    0.01,
                    0.0075,
                    0.0050,
                    0.0025,
                    0.001,
                    0.00075,
                    0.0005,
                    0.00025,
                    0.0001,
                    ]
```

XYZ experiment

```
for model_name in [...]:
    for dataset_name in [...]:
        for learning_rate in [...]:

            network = create_model(model_name)
            dataset = create_dataset(dataset_name)

            for epoch in range(num_epochs):
                for image, target in dataset:

                    # forward pass
                    output = network(image)

                    # backward pass
                    loss(output, target).backward()

                    # update model
                    optimizer.step(learning_rate)

                    # compute accuracy
                    acc = compute_accuracy()

                    # save to csv
                    save_results(acc)
```

save **EVERYTHING** about the experiment in the CSV

XYZ experiment in practice

```
loader_opts  = {'train_dataset'     : str(row['train_dataset']),
                'test_dataset'      : row['test_dataset'],
                'phase'             : None,
                'loader_type'       : str(row['loader_type']),
                'pytorch_dataset'   : bool(row['pytorch_dataset']),
                'dataset_path'      : '../../data',
                'dataset_path'      : '/scratch/users/papyan/datasets',
                'dataset_kwargs'    : {},
                'im_size'           : int(row['im_size']),
                'padded_im_size'    : int(row['padded_im_size']),
                'num_classes'       : int(row['num_classes']),
                'input_ch'          : int(row['input_ch']),
                'threads'           : 0,
                'limited_dataset'   : bool(row['limited_dataset']),
                'examples_per_class': int(row['examples_per_class']),
                'epc_seed'          : epc_seed_idx,
                'train_seed'        : train_seed_idx,
                'size_list'         : str(row['size_list']),
                'pretrained'        : bool(row['pretrained']),
                'multilabel'        : bool(row['multilabel']),
                'corrupt_prob'      : 0,
                'test_trans_only'   : True,
                'concat_loader'     : False,
                'loader_constructor': Constructor,
                'drop_last'         : False,
                }
```

```
train_opts   = {'crit'              : str(row['crit']),
                'net'               : str(row['net']),
                'optim'             : str(row['optim']),
                'epochs'            : int(row['epochs']),
                'lr'                : float(row['lr']),
                'milestones_perc'   : str(row['milestones_perc']),
                'gamma'             : float(row['gamma']),
                'train_batch_size'  : 128,
                'test_batch_size'   : 128,
                'cuda'              : torch.cuda.is_available(),
                'seed'              : int(row['seed']),
                'epsi'              : float(row['seed']),
                }
```

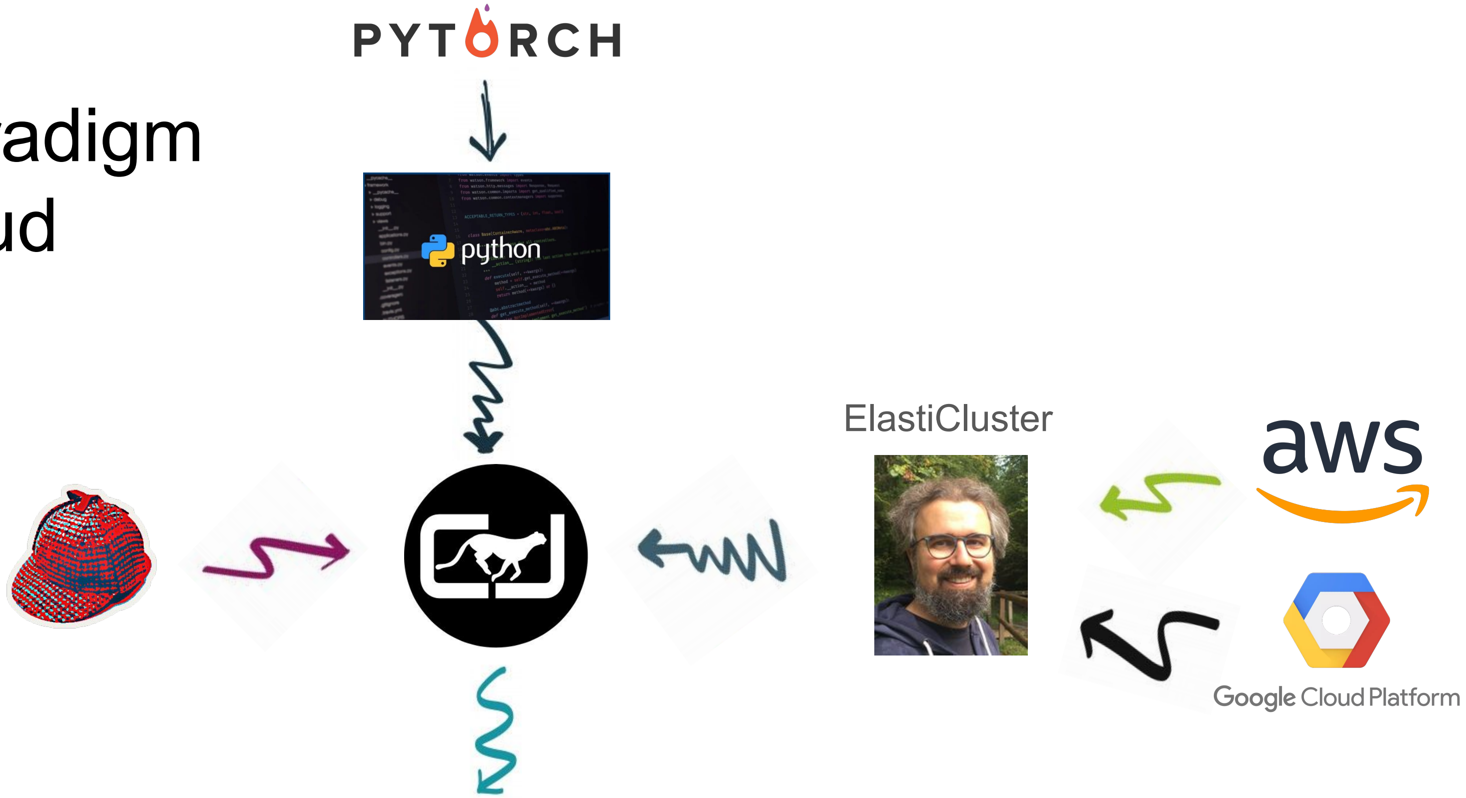
```
results_opts = {'training_results_path': training_results_path,
                'train_dump_file'   : str(row['train_dump_file']),
                'save_init_epoch'   : bool(row['save_init_epoch']),
                'garbage_collect'   : bool(row['garbage_collect']),
                'save_middle'       : bool(row['save_middle']),
                }

cpu_opts     = {'one_batch'         : bool(row['one_batch'])}

anals_opts   = {'k'                 : float('inf'),
                'project_last'      : False,
                'anals_results_path': analysis_results_path,
                'do_visual'         : False,
                'embedded_max_examples': 512,
                'stats_max_examples': float('inf'),
                'save_Sigma_wc'     : True,
                'vgg_remove_last_dropout' : True,
                'reset_classifier'  : True,
                'analyze_last_only' : True,
                'l_analysis'        : l,
                'layers_func'       : 'get_imp_layers',
                'hook_type'         : 'output',
                'activations_per_example': 10,
                'distribution'      : 'norm',
                'coeff_max_examples': 1000,
                'single_coeff_model': True,
                'record_activation' : False,
                'compute_norm_mean' : False,
                'compute_Sigma_b_w' : False,
                'compute_w_norm_mean': True,
                'compute_t_norm_mean': True,
                'power'             : 0.75,
                'seed'              : False,
                }

spectral_opts = {'hessian_type'     : hessian_type_list[hessian_type_i
                 'init_poly_deg'    : 64,
                 'poly_deg'         : 256, # paper suggests M=100
                 'mat_vec_iters'    : float('inf'),
                 'poly_points'      : 2**9,
                 'spectrum_margin'  : 0.05,
                 'log_hessian'      : False,
                 'start_eig_range'  : -float('inf'),
                 'stop_eig_range'   : float('inf'),
                 'power_method_iters': 256,
                 'repeat_idx'       : repeat_idx,
                 }
```

Stack paradigm & the cloud




```
pid 8dee32690f1fadf3ad36770d66874d6bb29a8bef
remote_account: papyan@login.sherlock.stanford.edu
1          28560970              COMPLETED
2          28560972              COMPLETED
3          28560973              COMPLETED
```

⋮

Me coding plots on python:

```python
import pandas as pd
import matplotlib.pyplot as plt

df = get_data_frame(path_to_csv)

colors = cm.rainbow(np.linspace(0, 1, num_learning_rates))

for dataset in [...]:
    for net in [...]:
        for learning_rate in [...]:

            df = df[(df['dataset'] == dataset)
                  & (df['net'] == net)
                  & (df['learning_rate'] == learning_rate)]

            plt.plot(df.epoch, df.accuracy, color=colors[learning_rate])
            plt.title('dataset: {}, net: {}, learning_rate: {}',
                      dataset,
                      net,
                      learning rate)
```

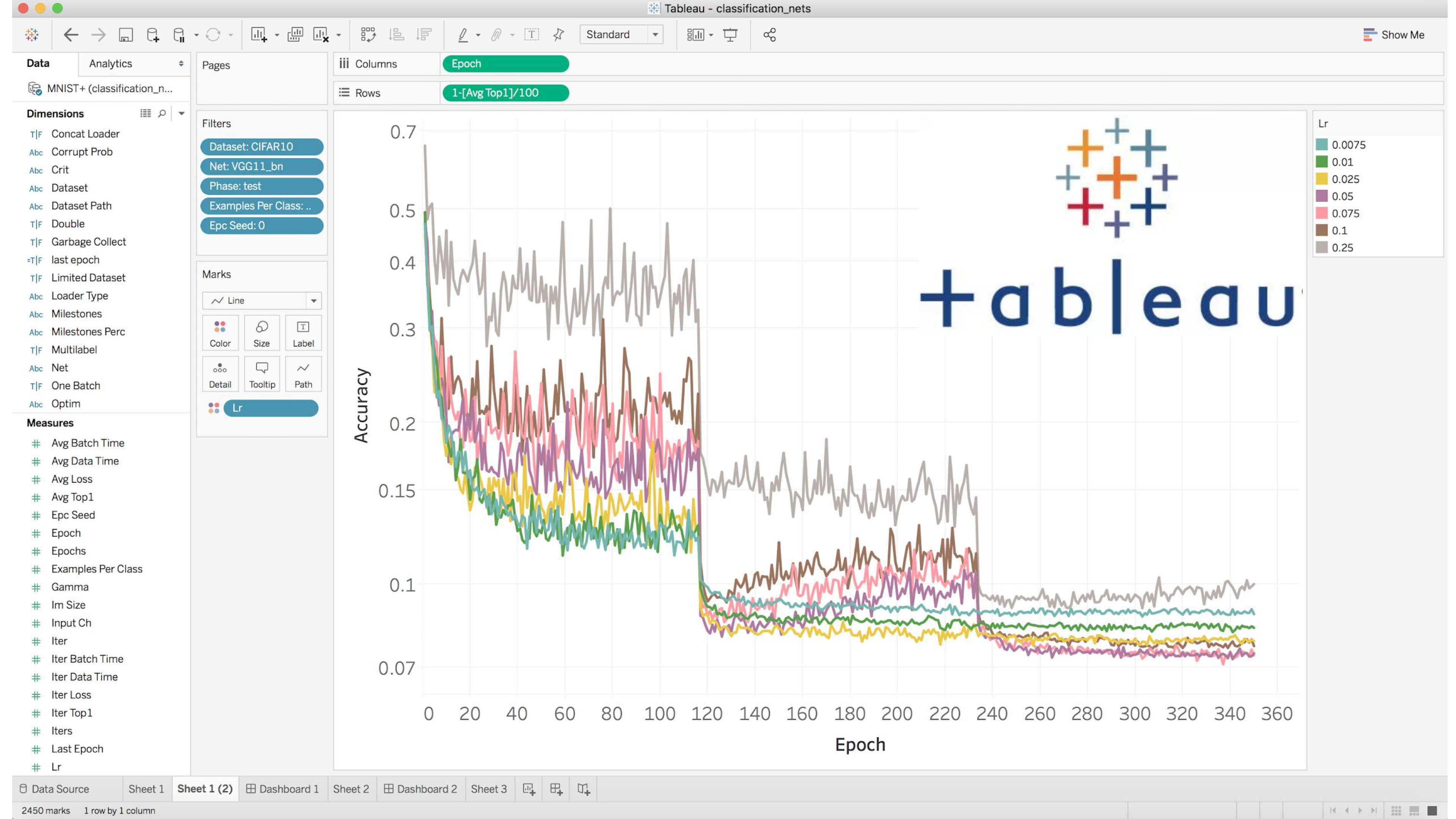
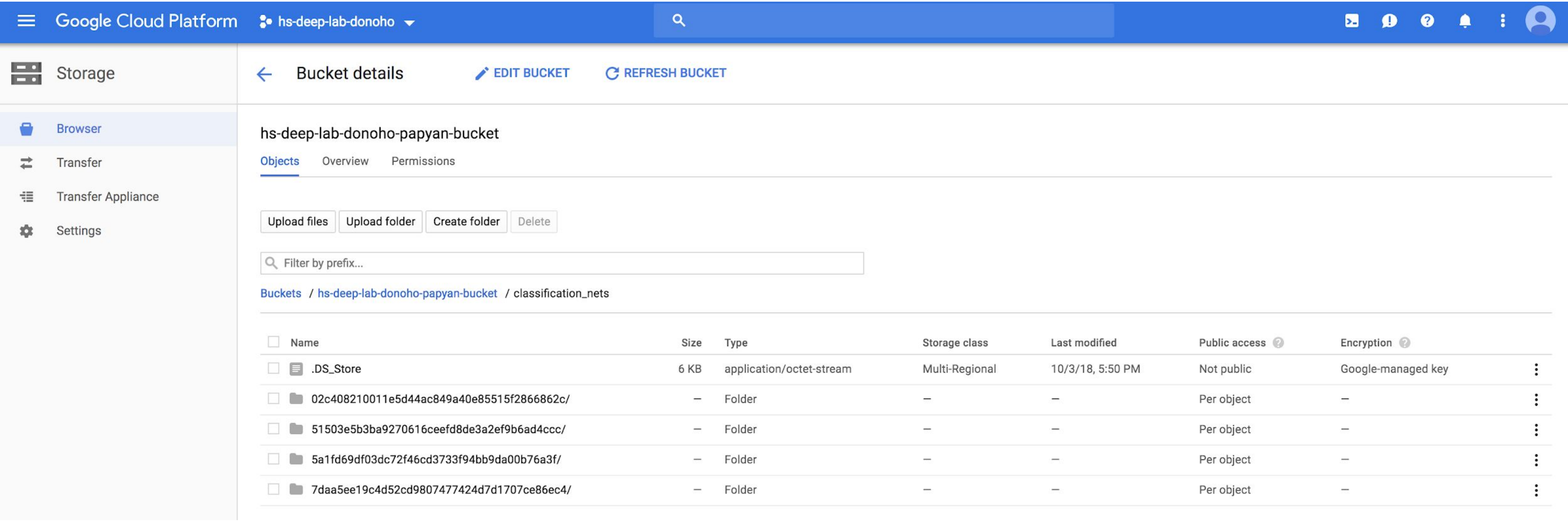


Tableau - classification_nets
Standard
Show Me
Data
Analytics
MNIST+ (classification_n...
Dimensions
Concat Loader
Corrupt Prob
Crit
Dataset
Dataset Path
Double
Garbage Collect
last epoch
Limited Dataset
Loader Type
Milestones
Milestones Perc
Multilabel
Net
One Batch
Optim
Measures
Avg Batch Time
Avg Data Time
Avg Loss
Avg Top1
Epc Seed
Epoch
Epochs
Examples Per Class
Gamma
Im Size
Input Ch
Iter
Iter Batch Time
Iter Data Time
Iter Loss
Iter Top1
Iters
Last Epoch
Lr
Pages
Filters
Dataset: CIFAR10
Net: VGG11_bn
Phase: test
Examples Per Class: ..
Epc Seed: 0
Marks
Line
Color
Size
Label
Detail
Tooltip
Path
Lr
Columns
Epoch
Rows
1-[Avg Top1]/100
Accuracy
0.7
0.5
0.4
0.3
0.2
0.15
0.1
0.07
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
Epoch
Lr
0.0075
0.01
0.025
0.05
0.075
0.1
0.25
Data Source
Sheet 1
Sheet 1 (2)
Dashboard 1
Sheet 2
Dashboard 2
Sheet 3
2450 marks
1 row by 1 column

Tableau is...

- **Powerful:** can compute mathematical expressions
- **Efficient:** can handle tens of GB easily
- **R:** you write R scripts (can do regression!)
- **Fast**: few clicks to create plot
- **Easy**: drag and drop
- **Cloud**: data sits on cloud
- **Time**: spent on more useful things

CJ › GCP ➤ Tableau ▶ Laptop



Final thoughts

Data science is XYZ studies

This is the field done properly

This is what there is to do

There is no other way

Join before it’s too late

Science In The Cloud

Monajemi/Donoho/Murri

Cloud is inevitable

Modern Data Science:

- Computationally demanding
- Varied in scale and scope

Campus-resident resources:

- Limited resources (e.g., GPUs, TPUs)
- Fixed policies

Cloud in inevitable

- Scalable and Fast
- Flexible (individual resources)
- Reliable

Google Cloud Platform	Amazon Web Services[7]	Microsoft Azure[8]
Google Compute Engine	Amazon EC2	Azure Virtual Machines
Google App Engine	AWS Elastic Beanstalk	Azure Cloud Services
Google Container Engine	Amazon EC2 Container Service	Azure Container Service
Google Cloud Bigtable	Amazon DynamoDB	Azure Cosmos DB
Google BigQuery	Amazon Redshift	Microsoft Azure SQL Database
Google Cloud Functions	Amazon Lambda	Azure Functions
Google Cloud Datastore	Amazon DynamoDB	Cosmos DB
Google Storage	Amazon S3	Azure Blob Storage

Existing Cloud Computing Models for DS

- Computing by provisioning a server
  - ElastiCluster-ClusterJob Model (Stanford-UZH)
  - CodaLab Worksheets Model (Stanford-Microsoft)
- Serverless Computing
  - PyWren (UC-Berkeley)
- Third Party Products
  - Databricks
  - Domino DataLabs
  - Civis Analytics
  - SageMaker (AWS)
  - Cloud ML (Google)

ElastiCluster-ClusterJob Model

\+ ElastiCluster +

ElastiCluster-ClusterJob Model

```
$ elasticluster start gce

$ cj parrun experiment.py gce
```

ElastiCluster

- Open-source software (GPL License) started at UZH
- Python API to setup and resize clusters
- Uses Ansible (Infrastructure as Code)
- Cloud-agnostic (AWS, GCE, AZURE)
- Offers many Operating Systems and Job Schedulers
- Offers many add-ons (JupyterHub, Cuda, etc.)

ElastiCluster Config

[setup/slurm]
provider=ansible
frontend_groups=slurm_master
compute_groups=slurm_worker,cuda

[login/gmail]
image_user=hatefmonajemi

[cloud/google]
provider=google
gce_client_id=***
gce_client_secret=***
gce_project_id=***

[cluster/gce]
setup=slurm
cloud=google
login =gmail
frontend_nodes=1
compute_nodes=4
flavor=n1-standard-4

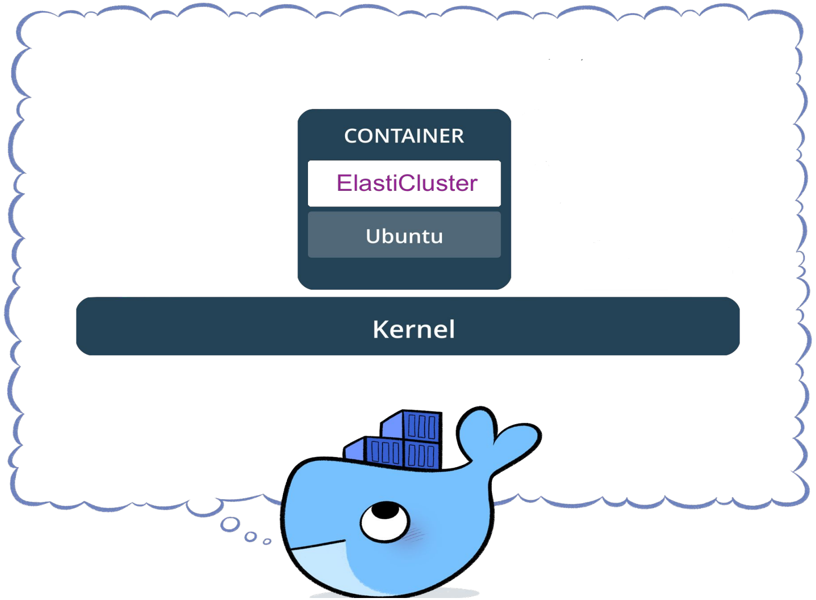
You choose!

ElastiCluster 0-Install Script

- Uses Docker containers to facilitate installation

```
elasticluster.sh:

exec docker run murri/elasticluster “$@”
```



[Link to Dockerfile]

Using ElastiCluster

- Use the script directly

```
$ ./elasticluster.sh start gce
```

- Define an alias for convenience (inside `~/.bash_profile`)

```
alias elasticluster='/Users/hatef/./elasticluster.sh'
```

```
$ elasticluster start gce
```

Give info of your cluster to CJ

- Extract the front-node IP address

```
$ elasticluster list-nodes gce
```

- Provide CJ with necessary config

```
$ cj config gce --update
```

Run and manage experiments!

```
$ cj parrun experiments.py gce
```

```
$ cj state

$ cj runlog $/1

$ cj error $/1

$ cj sanity (exists|lines) PID

$ cj reduce results.txt PID
```