

Massive Computational Experiments, Painlessly

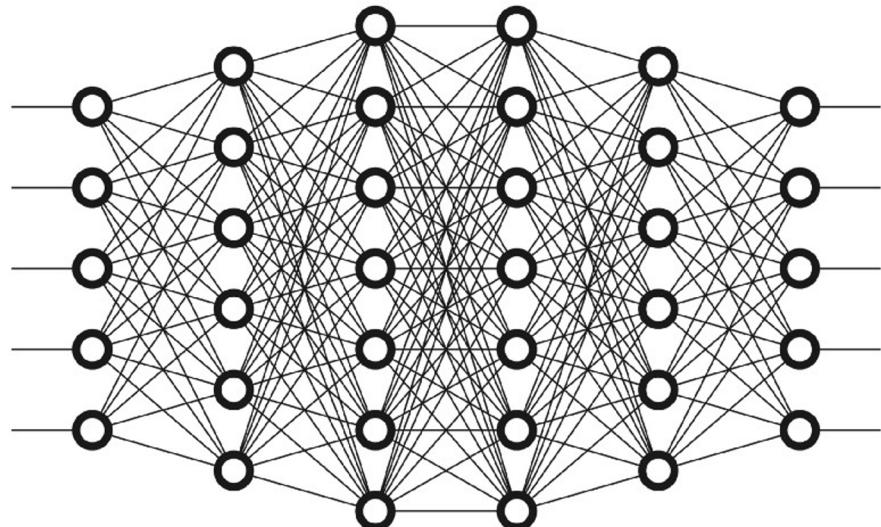
STATS 285
Stanford University



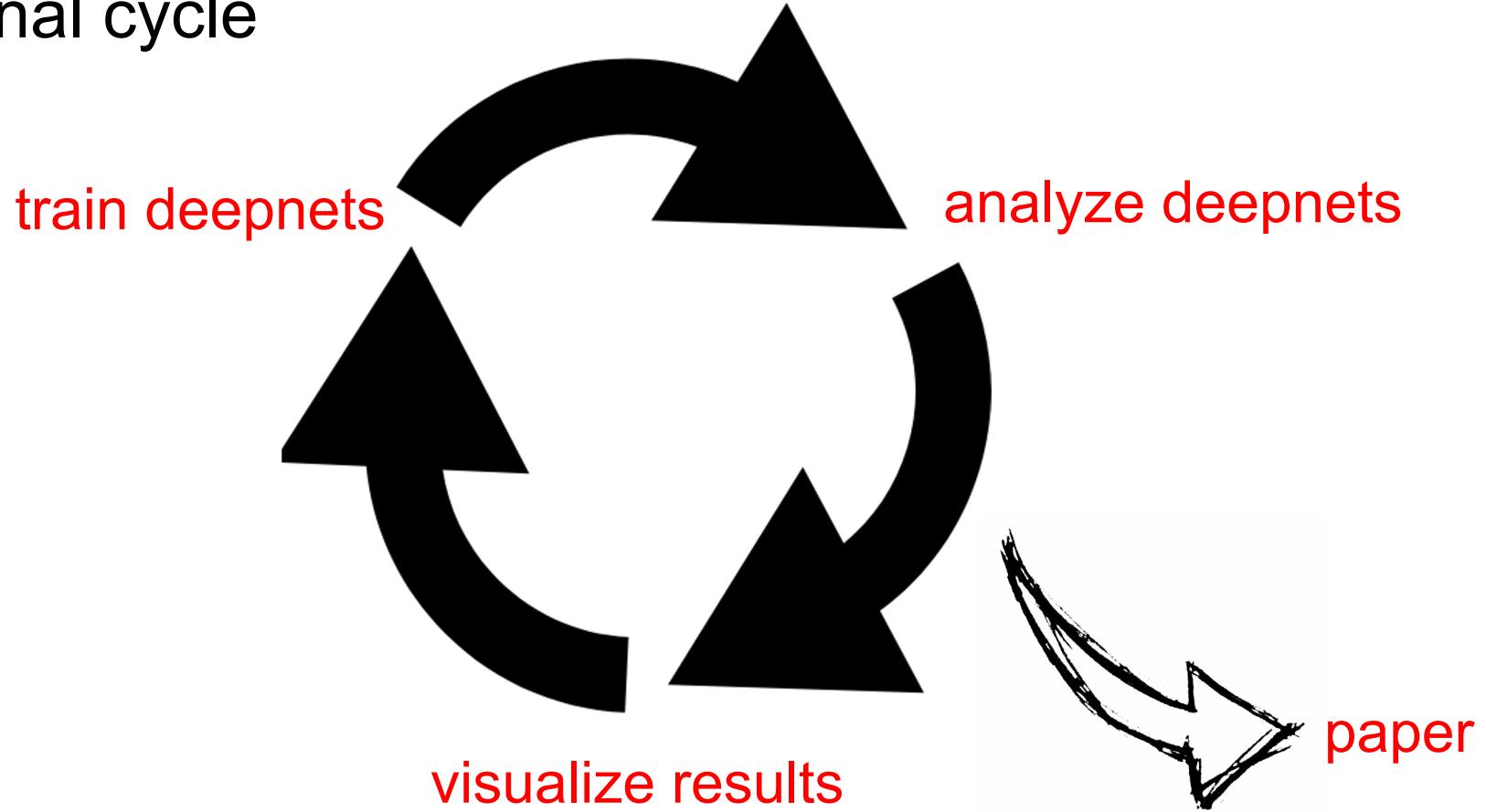
My research

Study of deep net:

- Features
- Weights
- Backpropagated errors
- Gradients
- Fisher information matrix
- Hessian
- ...



Eternal cycle



Training deepnets: experiment specification

- Dataset:
 - MNIST, FashionMNIST, CIFAR10, CIFAR100, ImageNet
- Network:
 - MLP, LeNet, VGG, ResNet
- Control parameters:
 - Dataset: sample size, number of classes
 - Network: width, depth
 - Optimization: algorithm, learning rate, learning rate scheduler, batch size
- Observables:
 - Top1 error, loss

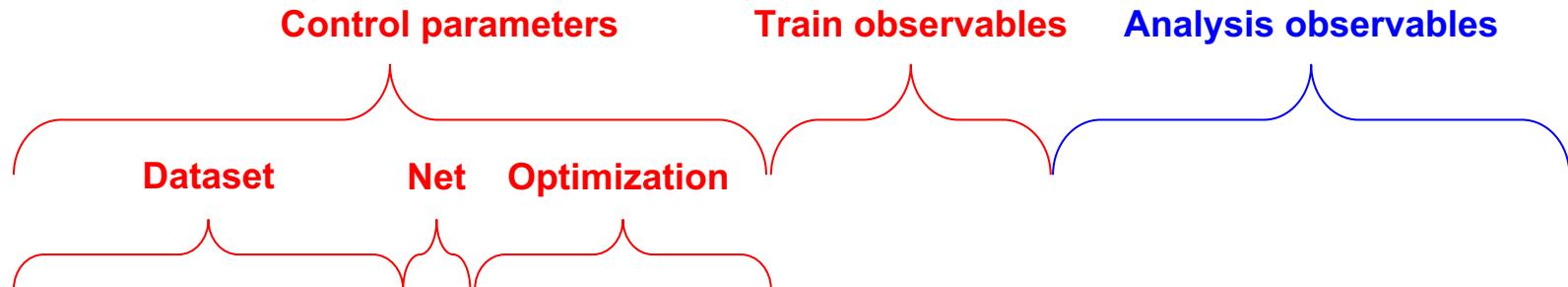
Training deepnets: experiment results

	Control parameters											Train observables		
	Dataset	Network	Number of Classes	Examples per class	Depth	Optimizer	Learning rate	Batch size	Phase	Epoch	Top-1 error			
Dataset			Network			Optimization								
0006d21e9a	MNIST	VGG	10	5000	11	SGD	0.25	128	test	250	99.38322368421052	0.028507341263129524		
000e518c8d	MNIST	VGG	10	5000	11	SGD	0.25	128	train	196	99.81637286324786	0.007986395810850156		
01003dd1ea	MNIST	VGG	10	5000	11	SGD	0.25	128	test	30	98.41694078947368	0.057988858144534264		
01421e353c	MNIST	VGG	10	5000	11	SGD	0.25	128	test	143	99.22902960526316	0.029485975980366532		
0215d9cc00	MNIST	VGG	10	5000	11	SGD	0.25	128	train	129	99.8046875	0.0081629870324117		
0216427c78	MNIST	VGG	10	5000	11	SGD	0.25	128	train	187	99.86812232905983	0.0058947943158957185		
022684ea96	MNIST	VGG	10	5000	11	SGD	0.25	128	test	25	98.4888980263158	0.05374518193696674		
027314b8b3	MNIST	VGG	10	5000	11	SGD	0.25	128	test	234	99.34210526315789	0.027728269660943432		
02a513b57a	MNIST	VGG	10	5000	11	SGD	0.25	128	test	349	99.44490131578948	0.028374243527650833		
031de9266f	MNIST	VGG	10	5000	11	SGD	0.25	128	test	44	98.01603618421052	0.07326431356762585		
03585ce035	MNIST	VGG	10	5000	11	SGD	0.25	128	train	77	98.5844017094017	0.05233209439688641		
362668092	MNIST	VGG	10	5000	11	SGD	0.25	128	train	292	100	0.001758613106277254		
03672e776e	MNIST	VGG	10	5000	11	SGD	0.25	128	train	317	100	0.001763534063521104		

Analyzing deepnets: analysis specification

- Dataset:
 - MNIST, FashionMNIST, CIFAR10, CIFAR100, ImageNet
- Network:
 - MLP, LeNet, VGG, ResNet
- Control parameters:
 - Dataset: sample size, number of classes
 - Network: width, depth
 - Optimization: find control parameters leading to best top-1 error
- Observables:
 - Spectra of deepnets features, backpropagated errors, gradients, Fisher information matrix, Hessian, ...

Analyzing deepnets: analysis results

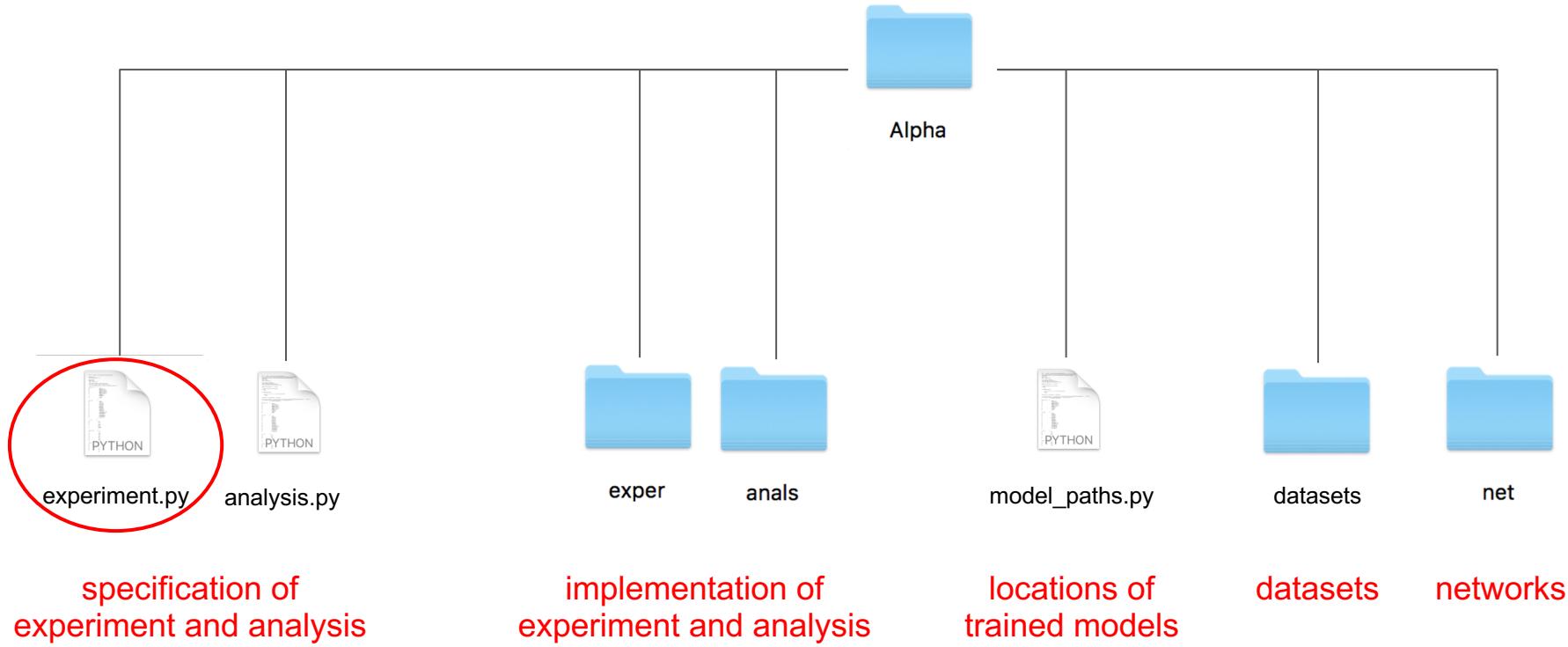


	Dataset	Network	Number of classes	Examples per class	Depth	Optimizer	Learning rate	Batch	Top-1 error	Path to model	Matrix type	Layer	Index of eigenvalue	Eigenvalue
00008c3007	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	features	0	240	0.095884531
0003bb8fa2	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	1562	6.224693205
0009cbc2ee	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	228	4.193413758
000c50127e	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	features	0	148	0.263989627
000d56712f	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	1084	3.365795819
001039f911	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	features	0	274	0.086070373
0015db9977	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	699	1.015973296
001d99a2ed	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	496	1.637548194
00224a8725	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	975	5.148281800
00254c969b	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	1986	-4.59046075
0027ad40da	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	1477	7.881029162
002b584223	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	1794	9.020487135
002fdcbd3a	MNIST	MLP	10	5000	8	SGD	[0.0678604404	128	98.97203947	/scratch/users/p	backprop errors	0	103	8.312590580

In practice slightly more complicated...

Phase	K_Normalization	Repeat_idx	Dataset_kwargs	Double
Dataset_path	Damping	N_vec	Im_size	Loader_constructor
Test_trans_only	Ignore_bias	Mult_num_classes	Padded_im_size	Sampler
Drop_last	save_K	Trace_est_iters	Num_classes	Pin_memory
Sampler	Hessian_layer	Perplexity_list	Input_ch	normalized_Fashion
Corrupt_prob	All_params	Double	Threads	Momentum
Load_epoch	Hessian_type	Rand_model	Limited_dataset	Weight_decay
Train_batch_size	Init_poly_degpoly_deg	Bidiag	Examples_per_class	GAN
Test_batch_size	Poly_points	Cpu_eigvec	Epc_seed	Forward_class
Training_results_path	Spectrum_margin	G_decomp_cpu	Train_seed	Classification
Analys_results_path	Kappa	Train_dataset	Size_list	Forward_func
Layers_func	Log_hessian	Test_dataset	Pretrained	Critnet
Seed	Start_eig_range	Loader_type	Retrain_last	Optim
Absorb_bn	Stop_eig_range	Pytorch_dataset	Multilabel	Optim_kwargs
Filter_bn	Power_method_iters	Dataset_path	Corrupt_prob	Epochs
Milestones_perc	Test_batch_size	Concat_loader	Reset_classifier	Lr
Gamma	Device	Switch_relu_pool	Resnet_type	Net_width
Train_batch_size	Seed	Scattering	Test_trans_only	Num_layers
Training_results_path	Train_dump_file	Save_init_epoch	Garbage_collect	
Save_middle	Epoch_list	One_batch	Epochs	

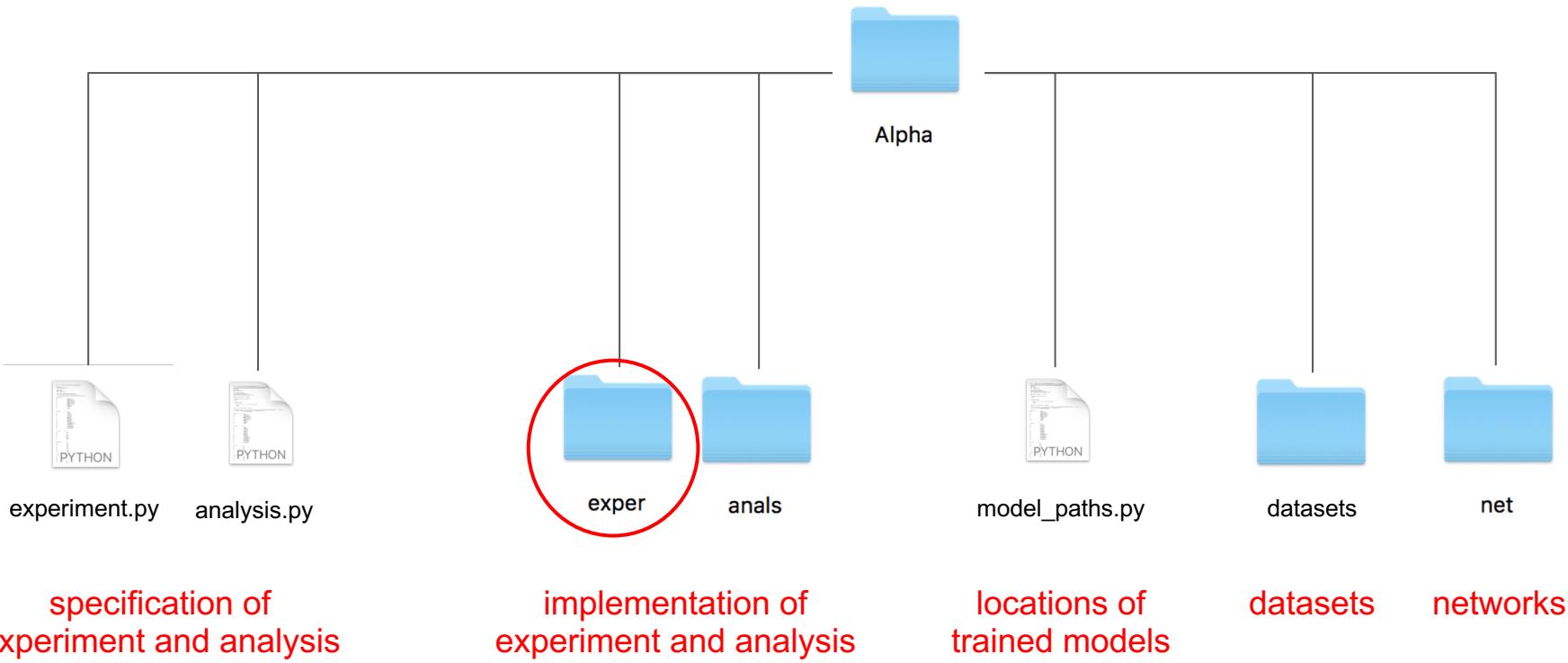
Alpha



experiment.py -- experiment specification

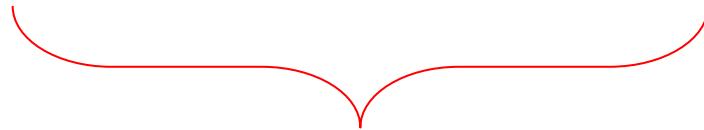
```
1 from exper import Experiment
2
3 dataset_list = ['MNIST', 'FashionMNIST', 'CIFAR10']
4 net_list = ['VGG11_bn', 'ResNet18']
5 lr_list = [0.1, 0.05, 0.001]
6 size_list = [13, 26, 51, 98, 189, 365, 702, 1351, 2599, 5000]
7
8 for dataset_idx in range(3):
9     for net_idx in range(2):
10         for size_idx in range(10):
11             for lr_idx in range(3):
12
13                 dataset_opts = {'dataset' : dataset_list[dataset_idx],
14                                 'examples_per_class': size_list[size_idx],
15                                 }
16
17                 network_opts = {'depth' : 8,
18                                 }
19
20                 optimization_opts = {'net' : net_list[net_idx],
21                                     'optim' : 'SGD',
22                                     'momentum' : 0.9,
23                                     'weight_decay' : 5e-4,
24                                     'epochs' : 350,
25                                     'lr' : lr_list[lr_idx],
26                                     'batch_size' : 2**7,
27                                     }
28
29                 opts = dict(dataset_opts, **optimization_opts)
30                 opts = dict(opts, **network_opts)
31
32                 Experiment(opts).run()
```

Alpha



Experiment class -- experiment implementation

```
class Experiment:  
  
    def __init__(self, opts):  
  
        for key, value in opts.items():  
            setattr(self, key, value)
```



Save all experiment specification in self

Experiment class -- experiment implementation

```
# criterion (loss)
import torch.nn as nn
func = getattr(nn, self.crit)
self.criterion = func()

# network
self.model = Network().construct(self.net, self)

# optimizer
import torch.optim as optim
func = getattr(optim, self.optim)
optimizer = func(self.model.parameters(), lr=lr, **self.optim_kwargs)

# datasets and loaders
constructor = LoaderConstructor().init(self)

self.train_dataset, self.train_loader = constructor.get_loader('train')
self.test_dataset, self.test_loader = constructor.get_loader('test')
```

The diagram illustrates the use of the `self` parameter in the code. Red arrows point from the `self` parameters in the `criterion`, `model`, `optimizer`, and `constructor` initialization calls to a central `self` box. A red box also surrounds the `self` parameter in the `get_loader` call. A red arrow points from this box to another `self` box at the bottom, which is labeled "Use fields from experiment specification".

Experiment class -- experiment implementation

```
# iterate over batches
for input,label in self.loader:
    input = input.to(self.device)
    label = label.to(self.device)

# run model
prediction = self.model

# compute loss
loss = self.crit(prediction, label)

# backpropagate
self.optimizer.zero_grad()
loss.backward()
self.optimizer.step()
```

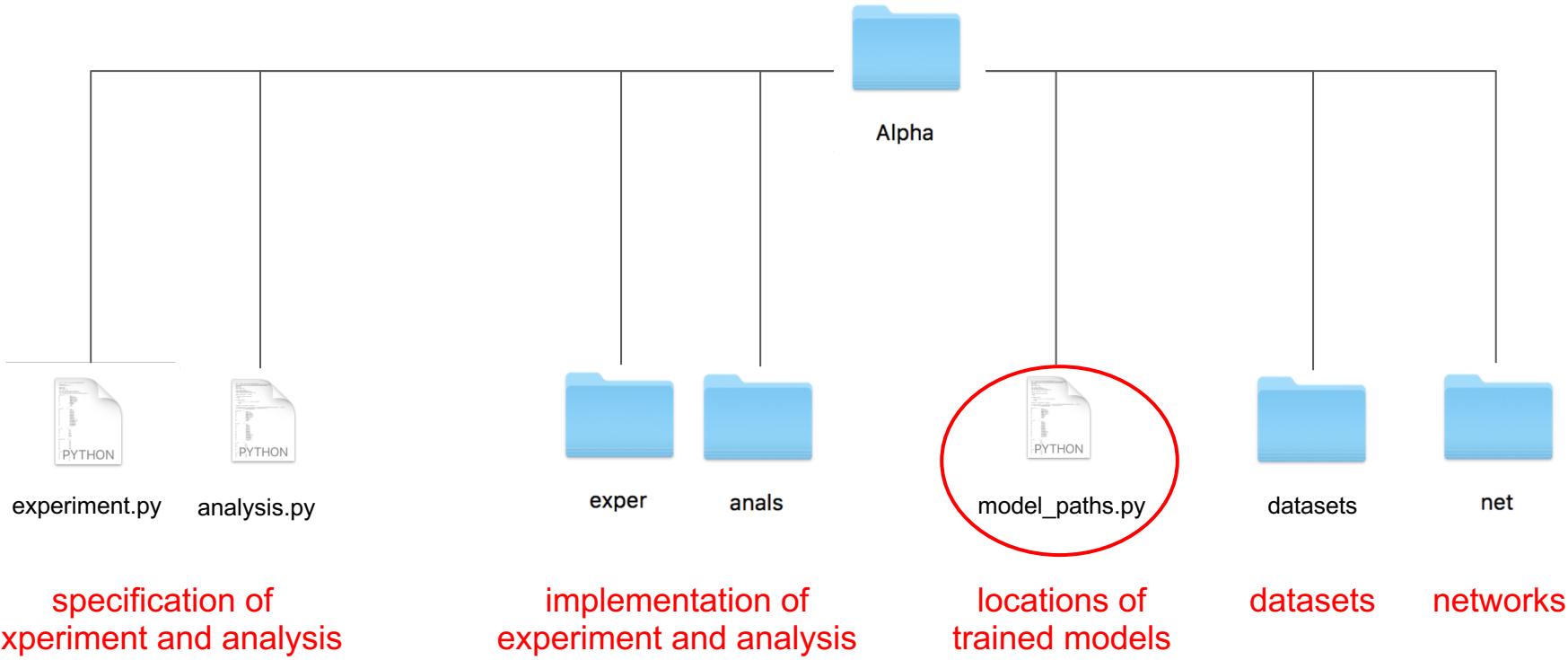
Experiment class -- experiment implementation

```
stats = {'phase' : phase,  
         'dataset' : dataset,  
         'epoch' : epoch,  
         'iter' : iter,  
         'iters' : len(loader),  
         'iter_batch_time' : batch_time.val,  
         'avg_batch_time' : batch_time.avg,  
         'iter_data_time' : data_time.val,  
         'avg_data_time' : data_time.avg}  
  
results.append(dict(self.__getstate__(), **stats))
```

experiment specification observables

Concatenate experiment specification to observables and as row to csv

Alpha



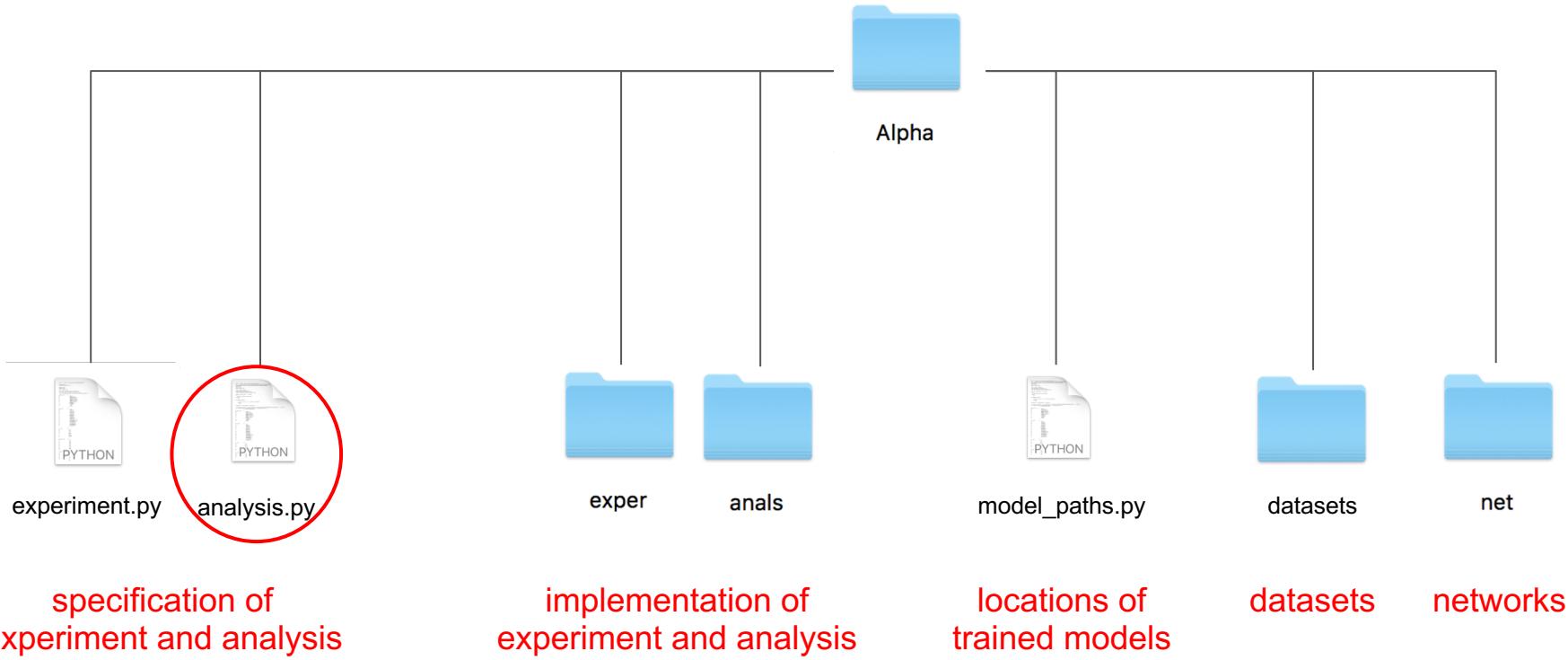
model_paths.py

dictionary of trained model paths

```
1 def get_model_path(dataset, network):
2     return paths[dataset+'.'+network]
3
4 paths = { 'MNIST.VGG11_bn'      : '/scratch/users/papyan/trained_models/3ad186bbfe8d7f6a6dfc1b116463956e72cd8b6b/' ,
5           'FashionMNIST.VGG11_bn' : '/scratch/users/papyan/trained_models/8dd94ceaea2fe0a49f9b2f369ae12ccb35323056/' ,
6           'CIFAR10.VGG11_bn'     : '/scratch/users/papyan/trained_models/43a7a251e5f0377cc0d59cd0bd15b94e87cc5f15/' ,
7           'MNIST.ResNet18'       : '/scratch/users/papyan/trained_models/eafc959c4215e81a053b9823832cee8f42db760f/' ,
8           'FashionMNIST.ResNet18': '/scratch/users/papyan/trained_models/5dc2d5768e4dae39739b13a0ebcafbe57bd92cae/' ,
9           'CIFAR10.ResNet18'     : '/scratch/users/papyan/trained_models/8d8c31045a4252a1b9a7b6aa6d3e041321b063f7/' ,
10          }
```

* Each of this paths corresponds to all the models trained for a certain dataset and a certain network

Alpha



analysis.py -- analysis specification

```
1 from model_paths import get_path
2 from anals.analysis import Analysis
3 from misc import get_csv
4 from misc import find_best_model
5
6 dataset_list = ['MNIST', 'FashionMNIST', 'CIFAR10']
7 net_list = ['VGG11_bn', 'ResNet18']
8 size_list = [13, 26, 51, 98, 189, 365, 702, 1351, 2599, 5000]
9 epoch_list = [10, 100]
10
11 for dataset_idx in range(3):
12     for net_idx in range(2):
13         for size_idx in range(10):
14             for epoch_idx in range(2):
15
16                 path = get_path(dataset_list[dataset_idx], net_list[net_idx])
17
18                 df = get_csv(path)
19
20                 best_df, _ = find_best_model(size_list[size_idx])
21
22                 analysis_opts = {'load_epoch' : epoch_list[epoch_idx],
23                                  }
24
25                 opts = dict(df, **analysis_opts)
26
27                 Analysis(opts).run()
```

Sherlock

- Cluster at Stanford
- Has many computational resources
 - CPUs
 - GPUs
- Useful for storing data
 - Laptop very limited in terms of memory
 - Data can get deleted if not touched for too long
 - Cloud costs money
- Interactive IPython notebook (Sherlock on demand)

ClusterJob: goal

```
1 from model_paths import get_path
2 from analns.analysis import Analysis
3 from misc import get_csv
4 from misc import find_best_model
5
6 dataset_list = ['MNIST', 'FashionMNIST', 'CIFAR10']
7 net_list = ['VGG11_bn', 'ResNet18']
8 size_list = [13, 26, 51, 98, 189, 365, 702, 1351, 2599, 5000]
9 epoch_list = [10, 100]
10
11 for dataset_idx in range(3):
12     for net_idx in range(2):
13         for size_idx in range(10):
14             for epoch_idx in range(2):
15
16                 path = get_path(dataset_list[dataset_idx], net_list[net_idx])
17
18                 df = get_csv(path)
19
20                 best_df, _ = find_best_model(size_list[size_idx])
21
22                 analysis_opts = {'load_epoch' : epoch_list[epoch_idx],
23                                 }
24
25                 opts = dict(df, **analysis_opts)
26
27                 Analysis(opts).run()
```

Easily parallelizable!

The diagram illustrates the parallelization of a nested loop structure. A central red arrow points from the loop body to a set of parameter values. Four diagonal red arrows point from this central arrow to four specific parameter sets, each labeled with its dataset, network, size, and epoch indices. Ellipses indicate additional iterations between the first and second parameter sets, and between the second and third parameter sets.

dataset_idx=0, net_idx=0, size_idx=0, epoch_idx=0

dataset_idx=0, net_idx=0, size_idx=0, epoch_idx=1

...

dataset_idx=2, net_idx=1, size_idx=3, epoch_idx=0

...

dataset_idx=2, net_idx=1, size_idx=9, epoch_idx=1

ClusterJob: jobs submission

```
vpapyan — -bash — 84x20
~ — -bash + ]
```

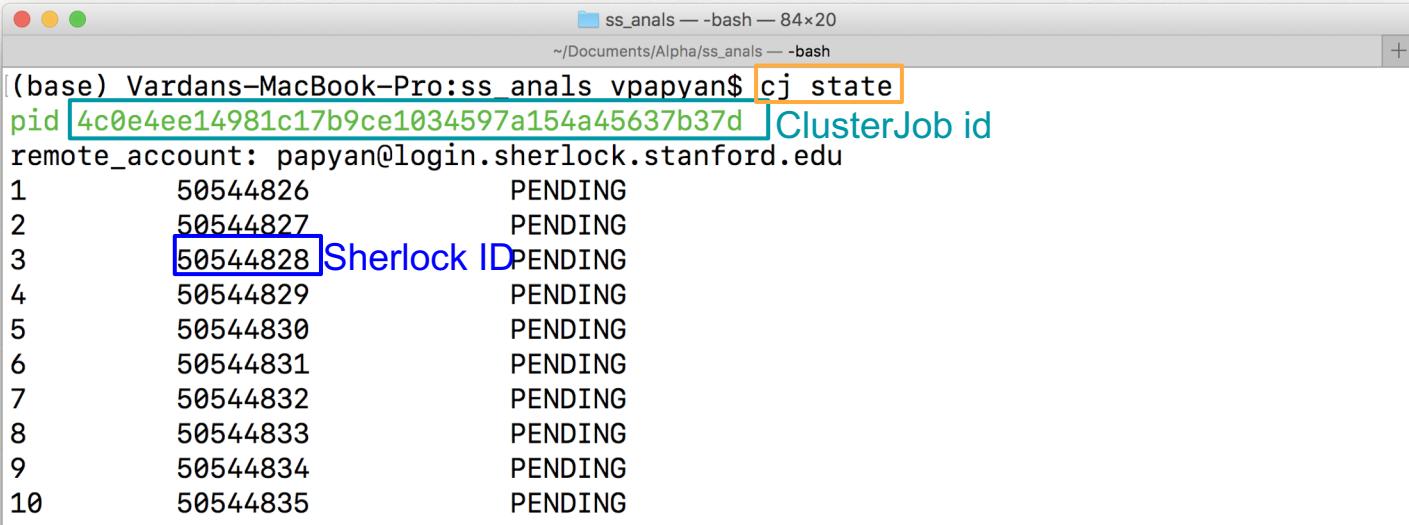
```
(base) Vardans-MacBook-Pro:~ vpapyan$ [ ]
(base) Vardans-MacBook-Pro:~ vpapyan$ [ ]
(base) Vardans-MacBook-Pro:~ vpapyan$ cj parrun analysis.py sherlock2 -dep .. -alloc
'-p donoho,owners --gres gpu:1 --mem=32GB --exclude=sh-115-02,sh-114-11 -m 'deepnet
spectra analysis' [ ]
```

ClusterJob: jobs submission

```
vpapyan@Vardans-MacBook-Pro:~$ cj parrun analysis.py sherlock2 -dep . -alloc
' -p donoho,owners --gres gpu:1 --mem=32GB --exclude=sh-115-02,sh-114-11 -m 'deepnet
spectra analysis' 1 GPU 32GB memory nodes in sherlock that description
partitions in per job per job don't work for me of jobs
sherlock I use
```

dependencies
cluster to run it on except analysis.py
run it on analysis.py
-alloc
nodes in sherlock that don't work for me
description of jobs

ClusterJob: check state of jobs



The screenshot shows a terminal window titled "ss_anals — -bash — 84x20" with the path "~/Documents/Alpha/ss_anals — -bash". The command entered is "cj state". The output includes the ClusterJob id (pid) and the Sherlock ID for each job, along with their current state (all are PENDING). The "cj state" command is highlighted with a yellow box, and the ClusterJob id (pid) and Sherlock ID are highlighted with blue boxes.

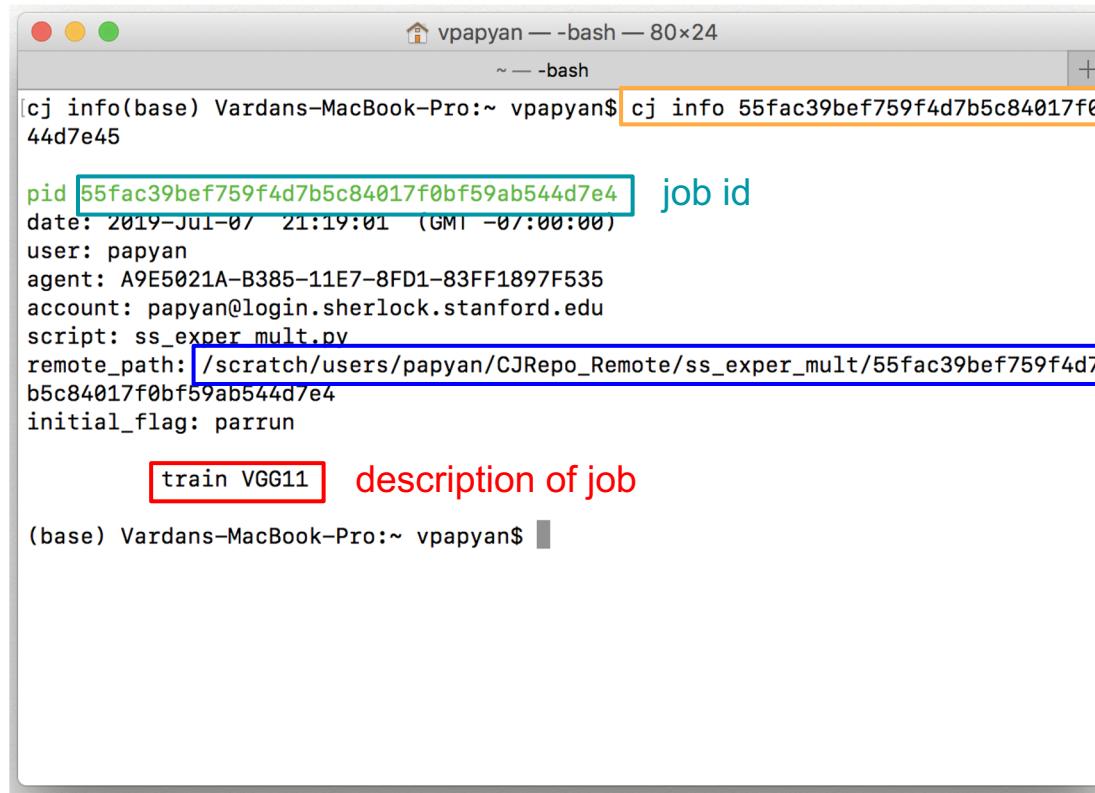
```
(base) Vardans-MacBook-Pro:ss_anals vpapyan$ cj state
pid [4c0e4ee14981c17b9ce1034597a154a45637b37d] ClusterJob id
remote_account: papyan@login.sherlock.stanford.edu
1      50544826      PENDING
2      50544827      PENDING
3      50544828 [Sherlock ID] PENDING
4      50544829      PENDING
5      50544830      PENDING
6      50544831      PENDING
7      50544832      PENDING
8      50544833      PENDING
9      50544834      PENDING
10     50544835      PENDING
```

ClusterJob: check progress of jobs

```
vpapyan -- bash -- 167x20
~ -- bash

(base) Vardans-MacBook-Pro:~ vpapyan$ cj runlog 55fac39bef759f4d7b5c84017f0bf59ab544d7e4/1
Datasets path is /scratch/users/papyan/datasets
Creating Train Loader...
Test transform only
Loader created with 5000.0 examples per class from 10 classes.
Not Pinning Images to GPU Memory (Slower, Low memory)
Creating Test Loader...
Test transform only
Loader created with 1000.0 examples per class from 10 classes.
Not Pinning Images to GPU Memory (Slower, Low memory)
Total parameters in VGG11_bn: 28156554
Checkpoint saved to ./results/dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=0.
pth
/scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39bef759f4d7b5c84017f0bf59ab544d7e4/1
Train Network: ['VGG11_bn'] Dataset: MNIST Epoch: [1/350][1/468] Time: 1.259 (1.259) Data: 0.359 (0.359) top1: 8.5938 (8.5938) loss: 2.3421 (2.3421)
Train Network: ['VGG11_bn'] Dataset: MNIST Epoch: [1/350][2/468] Time: 0.055 (0.657) Data: 0.000 (0.180) top1: 19.5312 (14.0625) loss: 3.5384 (2.9402)
Train Network: ['VGG11_bn'] Dataset: MNIST Epoch: [1/350][3/468] Time: 0.040 (0.451) Data: 0.000 (0.120) top1: 21.8750 (16.6667) loss: 7.6854 (4.5219)
Train Network: ['VGG11_bn'] Dataset: MNIST Epoch: [1/350][4/468] Time: 0.041 (0.349) Data: 0.000 (0.090) top1: 24.2188 (18.5547) loss: 9.7100 (5.8190)
Train Network: ['VGG11_bn'] Dataset: MNIST Epoch: [1/350][5/468] Time: 0.028 (0.285) Data: 0.000 (0.072) top1: 15.6250 (17.9688) loss: 10.4643 (6.7480)
Train Network: ['VGG11_bn'] Dataset: MNIST Epoch: [1/350][6/468] Time: 0.047 (0.245) Data: 0.000 (0.060) top1: 19.5312 (18.2292) loss: 8.3796 (7.0200)
```

ClusterJob: find location of jobs on cluster

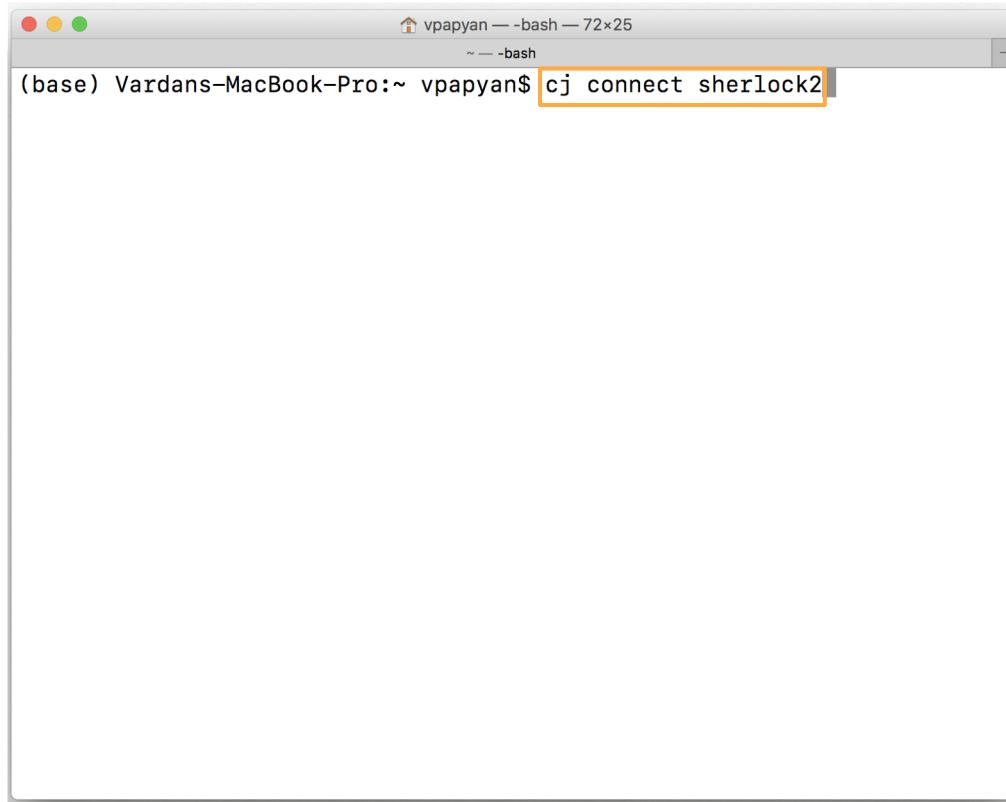


A terminal window titled "vpapyan — -bash — 80x24" showing the output of the "cj info" command. The command "cj info 55fac39bef759f4d7b5c84017f044d7e45" is highlighted in orange. The output includes job details: pid (55fac39bef759f4d7b5c84017f0bf59ab544d7e4), job id (highlighted in green), date (2019-Jul-07 21:19:01 (GMT -07:00:00)), user (papyan), agent (A9E5021A-B385-11E7-8FD1-83FF1897F535), account (papyan@login.sherlock.stanford.edu), script (ss_exper_mult.py), remote_path (/scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39bef759f4d7b5c84017f0bf59ab544d7e4), and initial_flag (parrun). The text "train VGG11" is highlighted in red and followed by the text "description of job". The prompt "(base) Vardans-MacBook-Pro:~ vpapyan\$" is at the bottom.

```
[cj info(base) Vardans-MacBook-Pro:~ vpapyan$ cj info 55fac39bef759f4d7b5c84017f044d7e45
pid [55fac39bef759f4d7b5c84017f0bf59ab544d7e4] job id
date: 2019-Jul-07 21:19:01 (GMT -07:00:00)
user: papyan
agent: A9E5021A-B385-11E7-8FD1-83FF1897F535
account: papyan@login.sherlock.stanford.edu
script: ss_exper_mult.py
remote_path: /scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39bef759f4d7b5c84017f0bf59ab544d7e4
initial_flag: parrun
train VGG11 description of job
(base) Vardans-MacBook-Pro:~ vpapyan$ ]
```

path on cluster to job

ClusterJob: connecting to cluster



A screenshot of a macOS terminal window titled "vpapyan — bash — 72x25". The window shows a command-line interface with the prompt "(base) Vardans-MacBook-Pro:~ vpapyan\$". A command "cj connect sherlock2" is being typed into the terminal, with the text highlighted by a yellow rectangular selection box.

```
(base) Vardans-MacBook-Pro:~ vpapyan$ cj connect sherlock2
```

ClusterJob: changing directory to folder of jobs

```
vpapyan — ssh ↵ CJ.pl connect sherlock2 — 80x24
~ — ssh ↵ CJ.pl connect sherlock2

-----
Sherlock      status   | n/a
                usage    | normal: 85.99% | use/tot: 1,541/ 1,792 cores
                           | global: 91.61% | use/tot: 22,708/24,788 cores

papyan      cur.jobs | 2 RUNNING (2 cores), 169 PENDING (169 cores)
                job wait | 8 days 21 hours and 4 minutes in normal

+-----+
| Disk usage for user papyan (group: donoho) |
+-----+
|   Filesystem |   volume /   limit           |   inodes /   limit   |
+-----+
|     HOME    | 12.6GB / 15.0GB [██████████] 84% |   - /   - ( -%) |
| GROUP_HOME | 989.0GB / 1.0TB  [███████████] 98% |   - /   - ( -%) |
|   SCRATCH  | 15.0TB / 20.0TB [██████████] 75% | 1.4M / 20.0M ( 7%) |
| GROUP_SCRATCH | 24.8TB / 30.0TB [██████████] 82% | 8.0M / 30.0M ( 26%) |
|     OAK    | 346.6GB / 10.0TB [          ] 3% | 1.5M / 1.5M (  %) |
+-----+
[papyan@sh-1n04 login ~]$ cd /scratch/users/papyan/CJRepo_Remote/ss_exper_mult/5
5fac39bef759f4d7b5c84017f0bf59ab544d7e4
```

path on cluster to job

ClusterJob: folder of jobs

```
papyan@sh-ln08 login /scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39bef759f4d7b5c84017f0bf59ab544d7e4_py_conda_req.txt ls
1 145 55fac39bef759f4d7b5c84017f0bf59ab544d7e4_py_conda_req.txt
10 146 56
100 147 57
101 148 58
102 149 59
103 15 6
104 150 60
105 16 61
106 17 62
107 18 63
108 19 64
109 2 65
11 20 66
110 21 67
111 22 68
112 23 69
113 24 7
114 25 70
115 26 71
116 27 72
117 28 73
118 29 74
119 3 75
12 30 76
```

one folder per each job

```
anal_func
backup
cfg.py
CJ_python_interpreter_script.py.bak
data_utils
debugging
demo
dist
exec_files
exper
exper_spec
extra_datasets
extra_loss
extra_optim
forward
Frick
loader
master.sh
matlab
misc
model_paths.py
net
package_mods
params
```

dependencies

ClusterJob: folder of a single job



A screenshot of a terminal window titled "vpapyan — ssh ↵ CJ.pl connect sherlock2 — 80x24". The window shows the output of a command-line session:

```
[papyan@sh-ln04 login /scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39b  
ef759f4d7b5c84017f0bf59ab544d7e4/1]$ ls  
bashMain.sh      logs      reproduce_ss_exper_mult.py  scripts  
CJrandState.pickle  __pycache__  results          ss_exper_mult.py  
[papyan@sh-ln04 login /scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39b  
ef759f4d7b5c84017f0bf59ab544d7e4/1]$
```

The "results" directory is highlighted with a red box. Below the terminal window, a red annotation text states:

results folder created after
experiment was run

ClusterJob: results folder

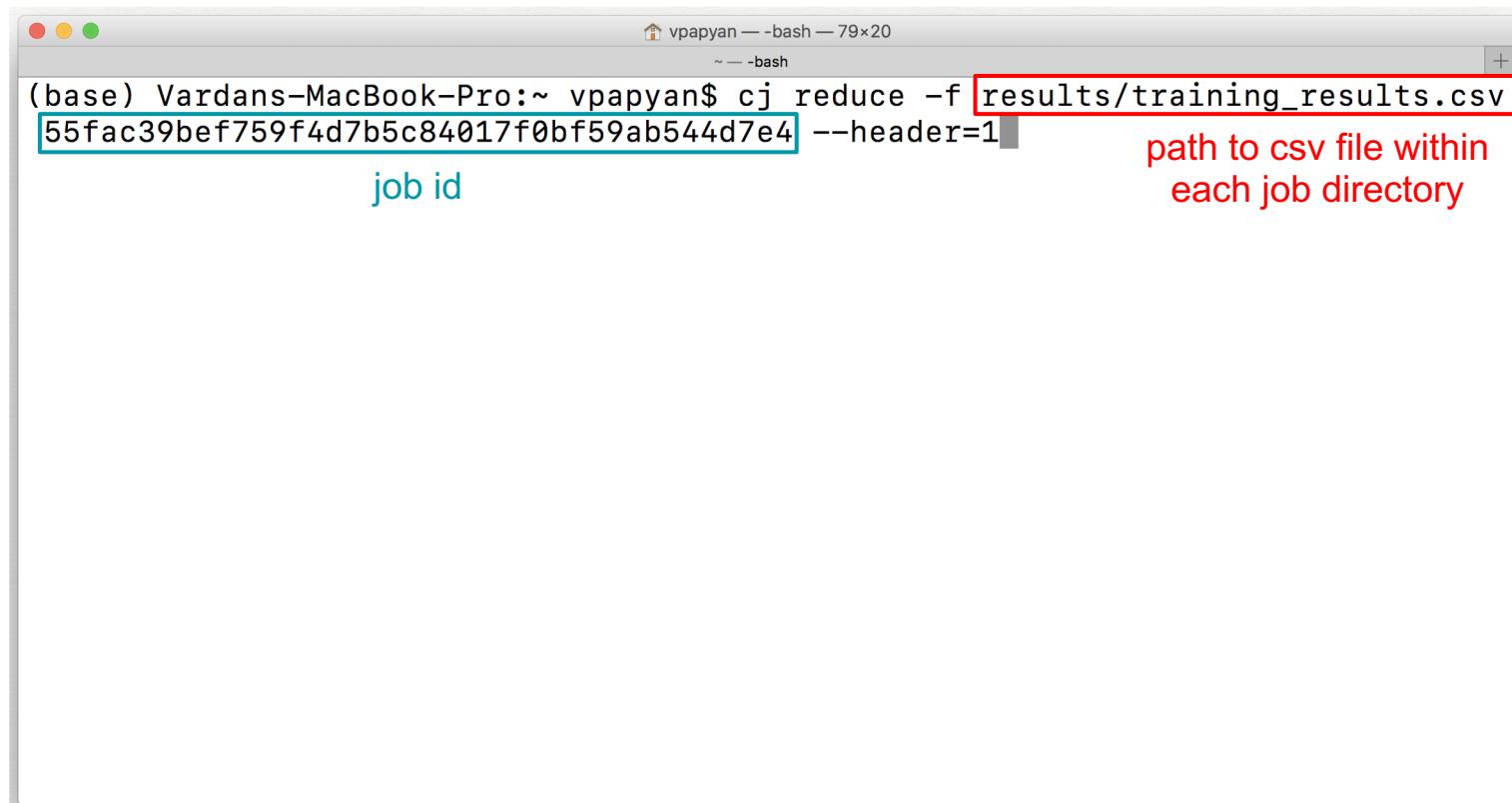
```
vpapyan — ssh • CJ.pl connect sherlock2 — 143x28
~ — ssh • CJ.pl connect sherlock2

[papyan@sh-1n04 login /scratch/users/papyan/CJRepo_Remote/ss_exper_mult/55fac39bef759f4d7b5c84017f0bf59ab544d7e4/1/results]$ ls
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=115.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=16.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=1.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=230.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=2.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=32.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=350.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=4.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=64.pth
dataset=MNIST-net=VGG11_bn-lr=[0p25]-examples_per_class=5000-num_classes=10-epc_seed=0-train_seed=0-forward_class=Classification-epoch=8.pth
latest_epoch.txt
optim-net=VGG11_bn-epoch=350.pth
results_epoch=350.csv
results_epoch=350.json
training_results.csv  training results csv
training_results.json
```

intermediate state -- can resume if interrupted in middle of training

deepnet models trained

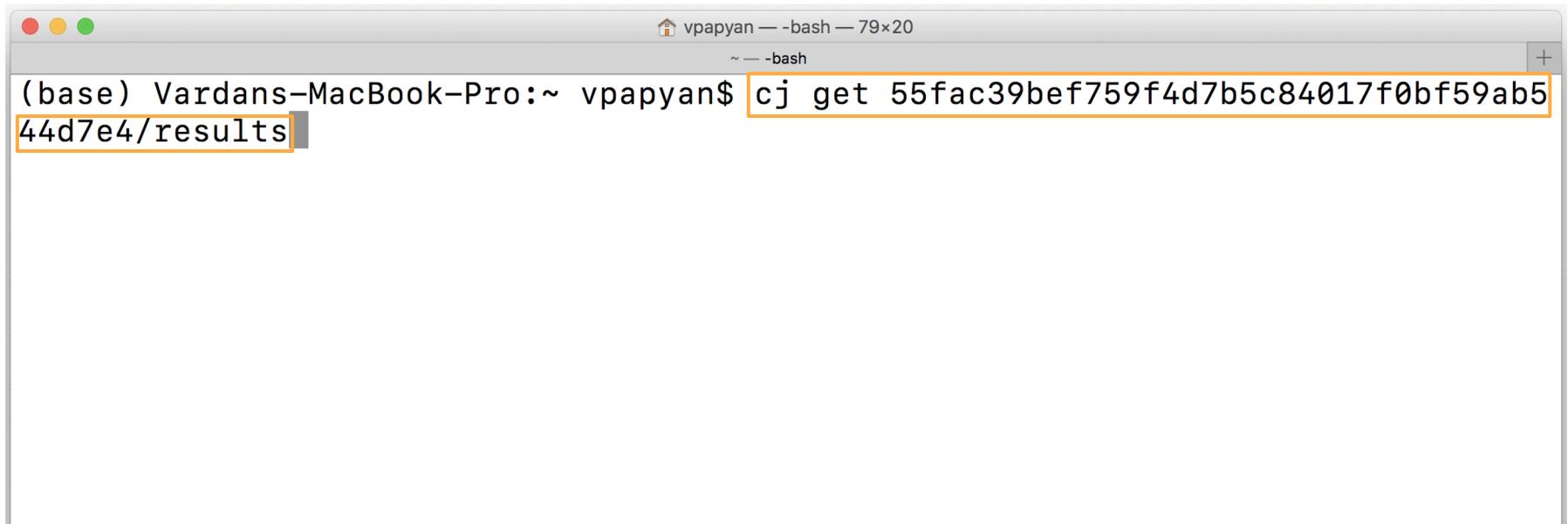
ClusterJob: reducing all results into a single csv file



```
(base) Vardans-MacBook-Pro:~ vpapyan$ cj reduce -f results/training_results.csv  
55fac39bef759f4d7b5c84017f0bf59ab544d7e4 --header=1
```

The terminal window shows a command being run in a bash shell. The command is `cj reduce -f results/training_results.csv`. The argument `results/training_results.csv` is highlighted with a red box. A blue box highlights the job ID `55fac39bef759f4d7b5c84017f0bf59ab544d7e4`. To the right of the command, a red annotation text states: "path to csv file within each job directory".

ClusterJob: downloading results to local machine



A screenshot of a macOS terminal window titled "vpapyan — -bash — 79x20". The window shows a command-line interface with the following text:

```
(base) Vardans-MacBook-Pro:~ vpapyan$ cj get 55fac39bef759f4d7b5c84017f0bf59ab5  
44d7e4/results
```

The command "cj get" and its arguments are highlighted with a yellow box.

Good way of keeping track of running jobs:

- reduce
- get
- plot locally

Elasticcluster

- During quarter Sherlock can get busy
- Two options:
 - Work nights / weekends / holidays
 - Cloud computing
- Elasticcluster allows to easily set up clusters on GCP/AWS/Azure/...
- Works seamlessly with ClusterJob

Tableau



test_results.csv



+ a b | e a u

Tableau in a nutshell

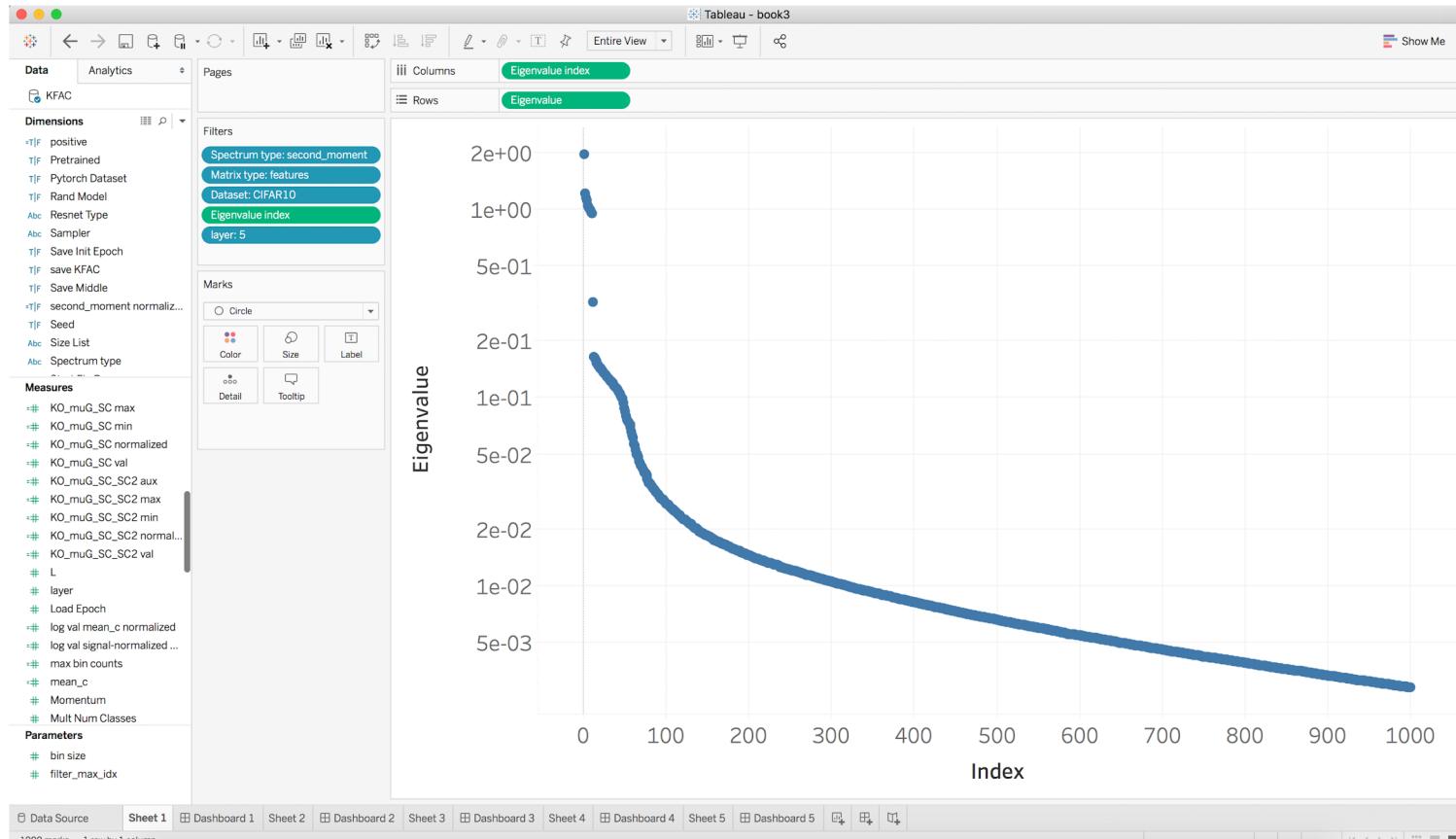


Tableau in a nutshell

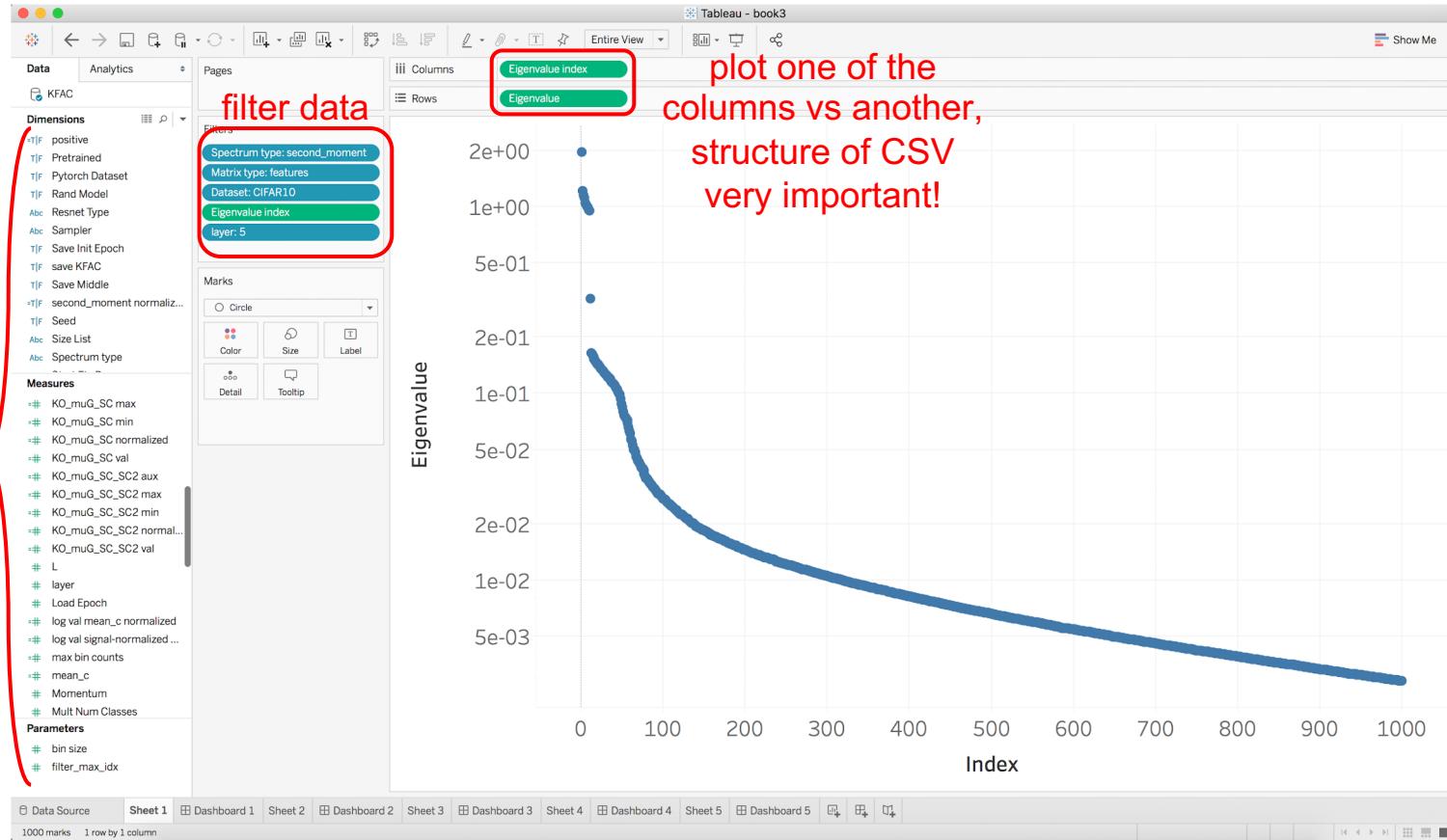


Tableau: XYZ grid

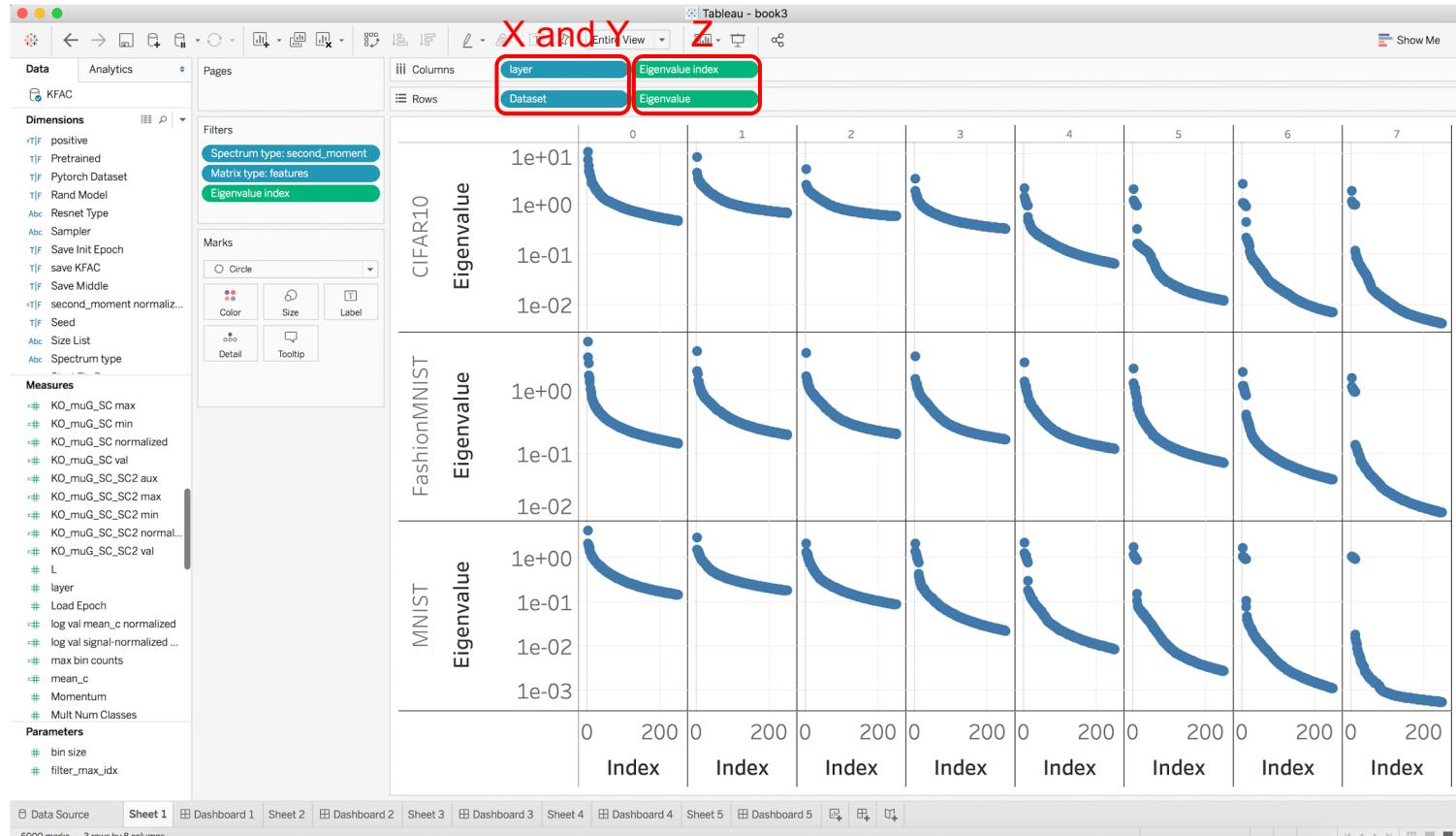


Tableau: simple functions of existing variables

Tableau - book3

Entire View | Columns | Rows | Data | Analytics | Pages | Filter | Sort | Refresh | Help | Exit

Dimensions

- Abc Size List
- Abc Spectrum type
- Abc Start Eig Range
- Abc Stop Eig Range
- Abc Test Dataset
- Test Trans Only
- top C**
- top C²
- Train Dataset
- Train Dump File
- Training Results Path
- Unnamed: 0
- Measure Names

Measures

- # second_moment max
- # second_moment min
- # second_moment normaliz...
- # second_moment val
- # second_moment_indicator
- # Spectrum Margin
- # Test Batch Size
- # Threads
- # top C mean_ccp
- # Total Params
- # Trace Est Iters
- # Train Batch Size
- # Train Seed
- # Val
- # Weight Decay
- # Number of Records
- # Measure Values

Parameters

- # bin size
- # filter_max_idx

Filters

- Spectrum type: second_moment
- Matrix type: features
- Eigenvalue index

Marks

- Circle
- Color
- Size
- Label
- Detail
- Tooltip

top C

IF [I]<[Num Classes] THEN
TRUE
ELSE
FALSE
END

The calculation is valid.

Sheets Affected: Apply OK

Tableau - book3

Columns: layer, Eigenvalue index

Rows: Dataset, Eigenvalue

Top C: False, True

Y-axis: Eigenvalue (log scale)

Subplots: CIFAR10, fashionMNIST

X-axis: Index (0 to 200)

Tableau

- Easy to analyze data -- drag and drop
- Easy to reproduce plots:
 - Delete results locally and keep only tableau sheet
 - Keep results on Sherlock2 / GCP
 - When need to recreate plot, download from cluster and open tableau sheet
- Easy to work with very large csv files using integration of tableau with the cloud
- Easy to calculate simple functions of existing columns

Summary

- **Alpha:** facilitates massive experiments by organizing code correctly
- **ClusterJob:** allows easy job parallelization
- **Sherlock2:** provides computational resources, storage, IPython notebooks
- **Elasticcluster:** creates cluster on cloud, when sherlock is not enough
- **Tableau:** easy visualization of massive data

