Accelerating Model Development by Reducing Operational Barriers

Patrick Hayes, Cofounder & CTO, SigOpt

Talk ID: S9556

SIGOPT

SIGOPT

Accelerate and amplify the impact of modelers everywhere

UNIVERSITY OF WATERLOO

FOURSQUARE

Y

a16z

SIGOPT

SigOpt automates experimentation and optimization

Data Preparation

Transformation
Labeling
Pre-Processing
Pipeline Dev.
Feature Eng.
Feature Stores

Experimentation, Training, Evaluation

Notebook & Model Framework

jupyter · PyTorch · H_2O · dmlc mxnet · scikit learn · Java

SIGOPT Experimentation & Model Optimization SIGOPT

- Insights, Tracking, Collaboration
- Model Search, Hyperparameter Tuning
- Resource Scheduler, Management

Hardware Environment

On-Premise | Hybrid | Multi-Cloud

aws · Microsoft Azure

Model Deployment

Validation
Serving
Deploying
Monitoring
Managing
Inference
Online Testing

Model Tuning

Deep Learning Architecture Search

Training & Tuning

Hyperparameter Search

Hyperparameter Optimization

Grid Search

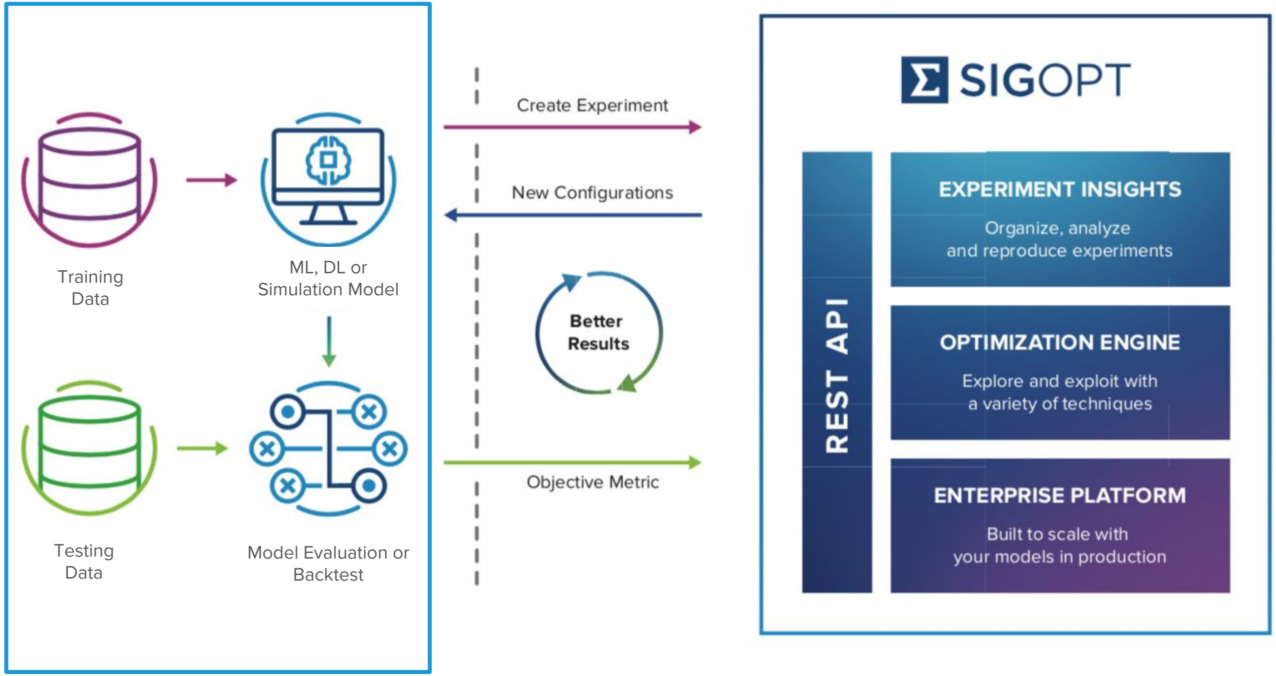
Evolutionary Algorithms

Random Search

Bayesian Optimization

How it works: Seamlessly tune any model

Never accesses your data or models



How it Works: Seamless implementation for any stack

1. Install SigOpt
2. Create experiment
3. Parameterize model
4. Run optimization loop
5. Analyze experiments

How it Works: Seamless implementation for any stack

1. Install SigOpt
2. Create experiment
3. Parameterize model
4. Run optimization loop
5. Analyze experiments

Step 1, part a: Installing sigopt

```
[ ]  pip install sigopt
```

Step 1, part b: initializing sigopt connection

```
[ ]  from sigopt import Connection
     conn = Connection(client_token="")
```

https://bit.ly/sigopt-notebook

How it Works: Seamless implementation for any stack

1. Install SigOpt
2. Create experiment
3. Parameterize model
4. Run optimization loop
5. Analyze experiments

Step 2

```
experiment = conn.experiments().create(
    name="Multi-Layer Perceptron",
    parameters=[
        dict(
            name="log_learning_rate",
            bounds=dict(min=-7, max=0),
            type="double"
        ),
        dict(
            name="activation",
            categorical_values=[
                dict(name="relu"),
                dict(name="sigmoid"),
                dict(name="tanh")
            ],
            type="categorical"
        ),
        dict(name="num_hidden_1", bounds=dict(min=1, max=6), type="int"),
        dict(name="num_hidden_2", bounds=dict(min=1, max=6), type="int"),
        dict(name="num_hidden_3", bounds=dict(min=1, max=6), type="int"),
        dict(name="batch_size", bounds=dict(min=5, max=20), type="int")
    ],
    parallel_bandwidth=4,
    observation_budget=100
)
```

https://bit.ly/sigopt-notebook

How it Works: Seamless implementation for any stack

1. Install SigOpt
2. Create experiment
3. **Parameterize model**
4. Run optimization loop
5. Analyze experiments

Step 3

```
def create_model(assignments):
    model = Sequential()
    model.add(Dense(assignments['num_hidden_1'],
                    input_dim=784,
                    activation=assignments['activation']))
    model.add(Dense(assignments['num_hidden_2'],
                    input_dim=assignments['num_hidden_1'],
                    activation=assignments['activation'])),
    model.add(Dense(assignments['num_hidden_3'],
                    input_dim=assignments['num_hidden_2'],
                    activation=assignments['activation'])),
    model.add(Dense(10, activation='softmax'))
    model.compile(
        optimizer=optimizers.RMSprop(
            lr=10**assignments['log_learning_rate']
        ),
        loss='categorical_crossentropy',
        metrics=['accuracy'],
    )

    model.fit(x_train, y_train, epochs=24,
              batch_size=assignments['batch_size'])
    return model

def evaluate_model(assignments):
    model = create_model(assignments)
    return model.evaluate(x_test, y_test)[1]
```

https://bit.ly/sigopt-notebook

How it Works: Seamless implementation for any stack

1. Install SigOpt
2. Create experiment
3. Parameterize model
4. Run optimization loop
5. Analyze experiments

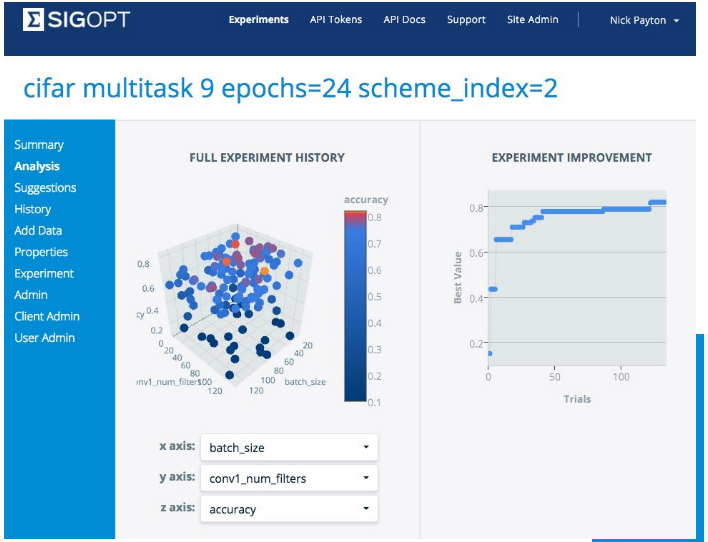
Step 4

```
while experiment.progress.observation_count < experiment.observation_budget:
    suggestion = conn.experiments(experiment.id).suggestions().create()
    assignments = suggestion.assignments
    value = evaluate_model(assignments)

    conn.experiments(experiment.id).observations().create(
        suggestion=suggestion.id,
        value=value,
    )

    experiment = conn.experiments(experiment.id).fetch()

best_assignments = conn.experiments(experiment.id)
  .best_assignments().fetch().data[0].assignments

# This is a SigOpt-tuned model
classifier = create_model(best_assignments)
```

https://bit.ly/sigopt-notebook

How it Works: Seamless implementation for any stack

1. Install SigOpt
2. Create experiment
3. Parameterize model
4. Run optimization loop
5. Analyze experiments

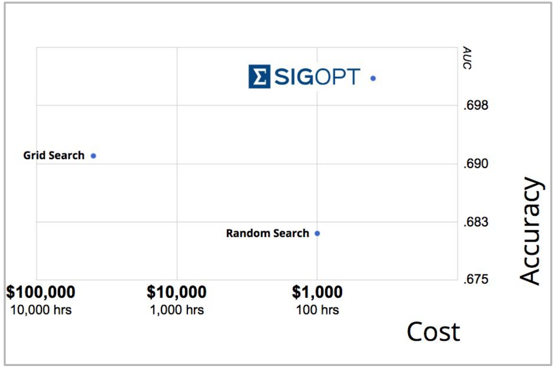


Benefits: Better, Cheaper, Faster Model Development

90% Cost Savings

Maximize utilization of compute

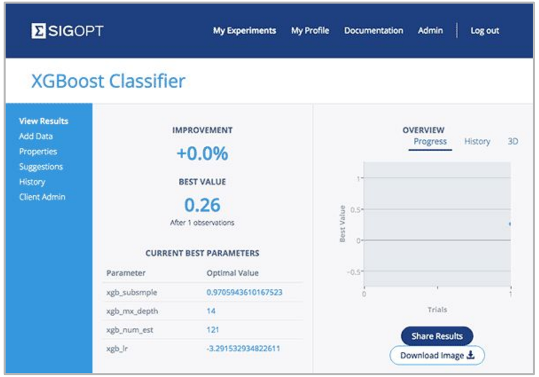
https://aws.amazon.com/blogs/machine-learning/fast-cnn-tuning-with-aws-gpu-instances-and-sigopt/



10x Faster Time to Tune

Less expert time per model

https://devblogs.nvidia.com/sigopt-deep-learning-hyperparameter-optimization/



Better Performance

No free lunch, but optimize any model

https://arxiv.org/pdf/1603.09441.pdf

Overview of Features Behind SigOpt

Experiment Insights	Intuitive web dashboards	Advanced experiment visualizations	Parameter importance analysis
	Reproducibility	Cross-team permissions and collaboration	Organizational experiment analysis
Optimization Engine	Continuous, categorical, or integer parameters	Up to 10k observations, 100 parameters	Conditional parameters
	Multimetric optimization	Constraints and failure regions	Multitask optimization and high parallelism
Enterprise Platform	REST API	Black-box interface	Libraries for Python, Java, R, and MATLAB
	Infrastructure agnostic	Model agnostic	Doesn't touch data

Key:

Only HPO solution with this capability

Applied deep learning introduces unique challenges

Failed observations

Constraints

Uncertainty

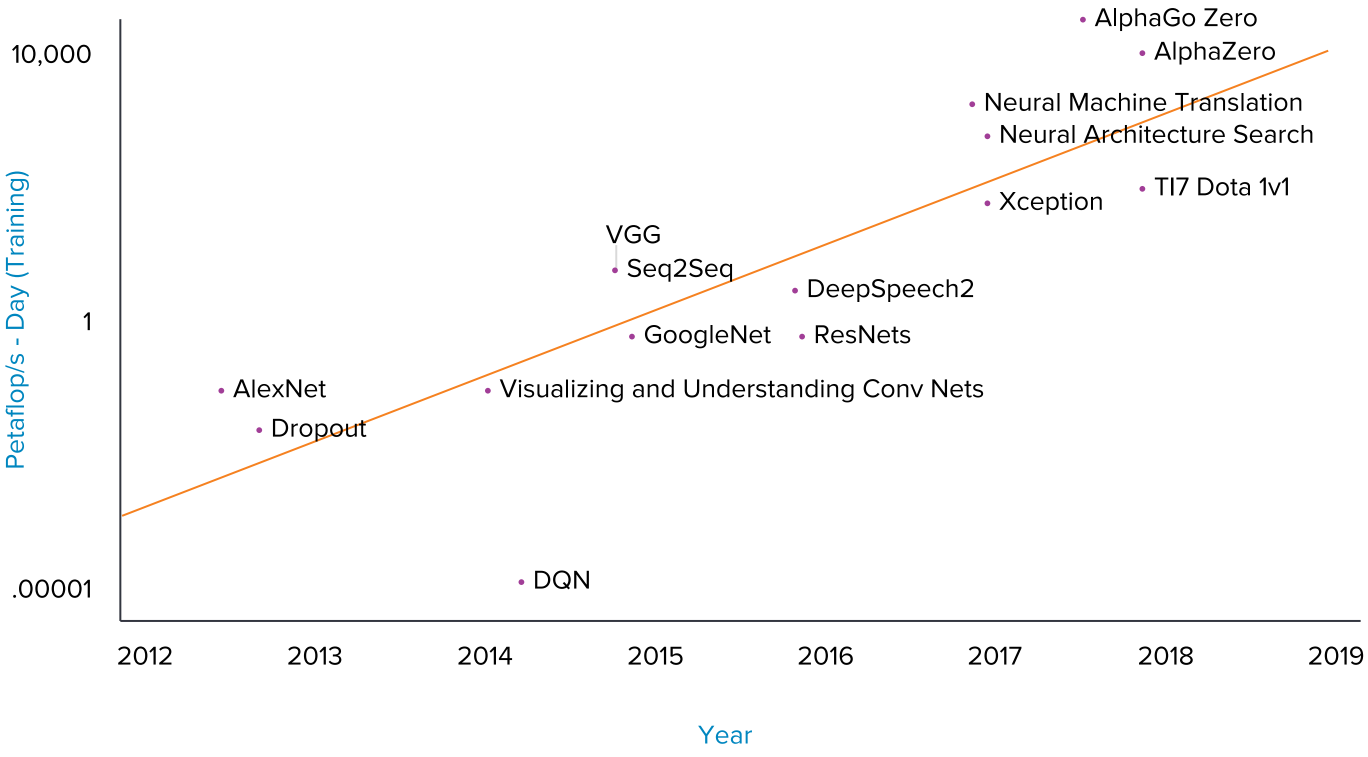
Competing objectives

Lengthy training cycles

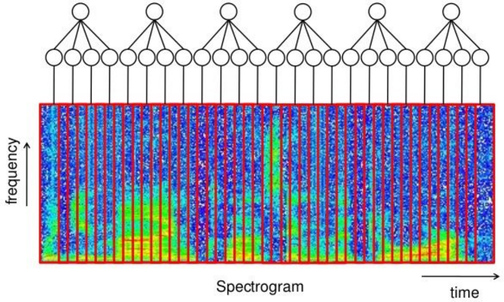
Cluster orchestration

How do you more efficiently tune models that take days (or weeks) to train?

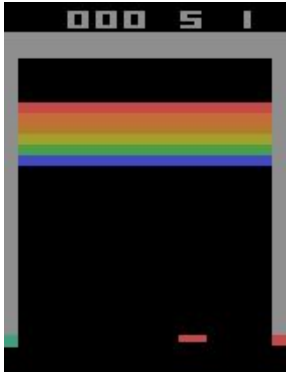
AlexNex to AlphaGo Zero: 300,000x Increase in Compute



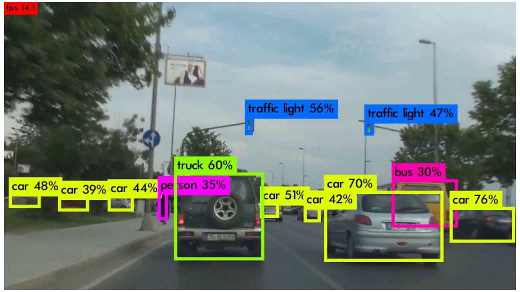
Speech Recognition



Deep Reinforcement Learning



Computer Vision



Training Resnet-50 on ImageNet takes 10 hours

Tuning 12 parameters requires at least 120 distinct models

That equals **1,200 hours, or 50 days, of training time**

Running optimization tasks in parallel is critical to tuning expensive deep learning models

Complexity of Deep Learning DevOps

Basic Case

Training One Model, No Optimization

Advanced Case

Multiple Users

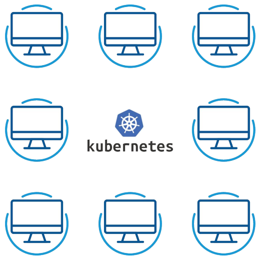
Concurrent Optimization Experiments

Concurrent Model Configuration Evaluations

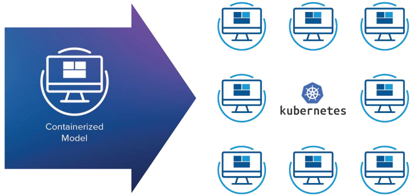
Multiple GPUs per Model

Cluster Orchestration

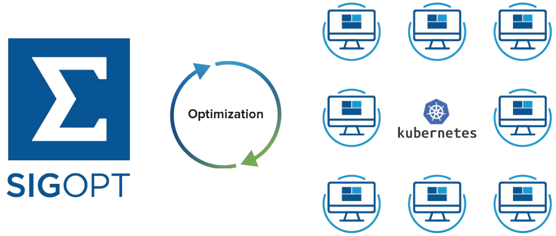
1 Spin up and share training clusters



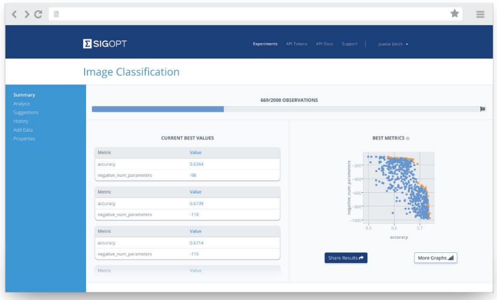
2 Schedule optimization experiments



3 Integrate with the optimization API



4 Monitor experiment and infrastructure



Problems:
Infrastructure, scheduling, dependencies, code, monitoring

Solution:
SigOpt Orchestrate is a CLI for managing training infrastructure and running optimization experiments

How it Works

```python

def evaluate_model(X, y):
    classifier = RandomForestClassifier(
        n_estimators=orchestrate.io.assignment('n_estimators', default=3),
        max_features=orchestrate.io.assignment('max_features', default=3),
        min_samples_leaf=orchestrate.io.assignment('min_samples_leaf', default=80)
    )
    cv_accuracies = cross_val_score(classifier, X, y, cv=5)
    return (numpy.mean(cv_accuracies), numpy.std(cv_accuracies))

if __name__ == "__main__":
    (X, y) = load_data()
    (mean, std) = evaluate_model(X=X, y=y)
    orchestrate.io.log_metric('accuracy', mean, std)

```

Seamless Integration into Your Model Code

```yaml
language: python3.6
name: Orchestrate Random Forest (python)
install:
  -  pip install -r requirements.txt
run:
  -  python model.py
optimization:
  metrics:
    - name: accuracy
  parameters:  # Fill in parameters to optimize
    - name: max_features
      type: int
      bounds:
        min: 1
        max: 4
    - name: n_estimators
      type: int
      bounds:
        min: 1
        max: 100
    - name: min_samples_leaf
      type: int
      bounds:
        min: 1
        max: 10
  parallel_bandwidth: 2
  observation_budget: 200
```

Easily Define Optimization Experiments

```
(venv) ➜  random_forest git:(master) ✗ sigopt run -f orchestrate.yml
/Users/benhsu/Developer/sigopt-examples/orchestrate/venv/lib/python3.7/site-packages/requests/__i
.4) doesn't match a supported version!
  RequestsDependencyWarning)
Containerizing your model and starting your experiment, this may take a few minutes...
Step 1/5 : FROM orchestrate/python-3.6:0.2.2
 ---> 623143becd6e
Step 2/5 : LABEL orchestrate-user-created=true
 ---> Using cache
 ---> 4ccb928a7baf
Step 3/5 : ADD "." "/"
 ---> Using cache
 ---> 88871a43efa4
Step 4/5 : WORKDIR /orchestrate
 ---> Using cache
 ---> aa76b5f6c620
Step 5/5 : RUN pip install -r requirements.txt
 ---> Using cache
 ---> 5185e635022d
Successfully built 5185e635022d
Successfully tagged orchestrate/random-forest:latest
Uploading the model environment to Amazon ECR, this may be limited by your connection speed...
```

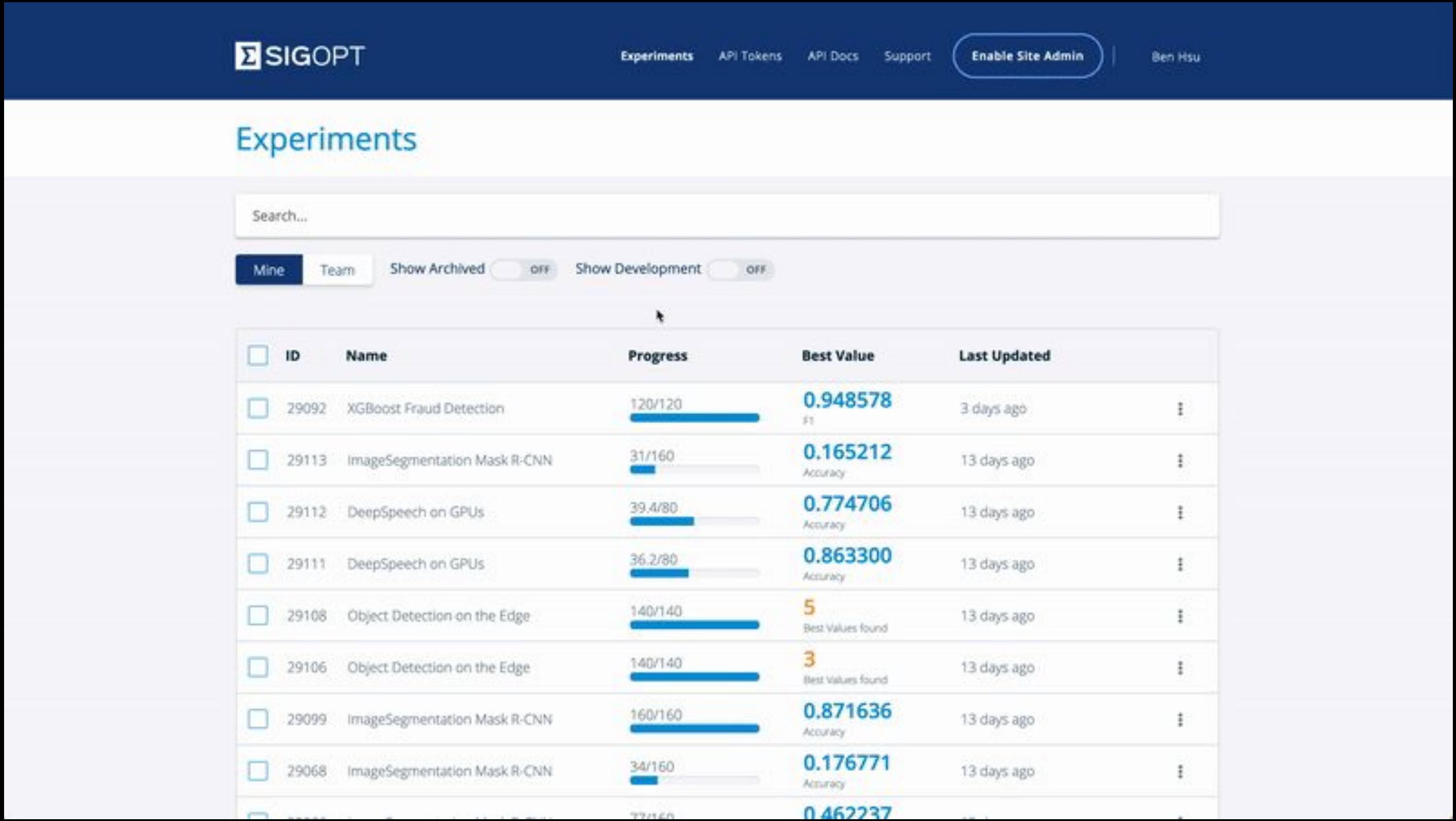
Easily Kick Off Optimization Experiment Jobs

```
(venv) ➜  random_forest git:(master) ✗ sigopt status 55471

Job Name: orchestrate-55471
Job Status: Not Complete

Experiment Name: Orchestrate Random Forest (python)
8 / 200 Observations
0 Observation(s) failed

Pod status:

Pod Name                  Status                           Success                 Failed
orchestrate-55471-cd5f2   Running                          4                       0
orchestrate-55471-pv4hs   Running                          5                       0

View more at: https://app.sigopt.com/experiment/55471
```

Check the Status of Active and Completed Experiments

```
[orchestrate-55471-cd5f2] /usr/local/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
y module and should not be imported. It will be removed in a future NumPy release.
[orchestrate-55471-cd5f2]   from numpy.core.umath_tests import inner1d
[orchestrate-55471-cd5f2] Observation data: {"suggestion": "21807868", "values": [{"name": "accuracy", "value": 0.95333333
}], "failed": false, "metadata": {"pod_name": "orchestrate-55471-cd5f2"}}
[orchestrate-55471-pv4hs] /usr/local/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
y module and should not be imported. It will be removed in a future NumPy release.
[orchestrate-55471-pv4hs]   from numpy.core.umath_tests import inner1d
[orchestrate-55471-pv4hs] Observation data: {"suggestion": "21807869", "values": [{"name": "accuracy", "value": 0.96, "val
 false, "metadata": {"pod_name": "orchestrate-55471-pv4hs"}}
[orchestrate-55471-cd5f2] /usr/local/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
y module and should not be imported. It will be removed in a future NumPy release.
[orchestrate-55471-cd5f2]   from numpy.core.umath_tests import inner1d
[orchestrate-55471-cd5f2] Observation data: {"suggestion": "21807870", "values": [{"name": "accuracy", "value": 0.94666666
}], "failed": false, "metadata": {"pod_name": "orchestrate-55471-cd5f2"}}
[orchestrate-55471-pv4hs] /usr/local/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
y module and should not be imported. It will be removed in a future NumPy release.
[orchestrate-55471-pv4hs]   from numpy.core.umath_tests import inner1d
[orchestrate-55471-cd5f2] /usr/local/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
y module and should not be imported. It will be removed in a future NumPy release.
[orchestrate-55471-cd5f2]   from numpy.core.umath_tests import inner1d
[orchestrate-55471-pv4hs] Observation data: {"suggestion": "21807871", "values": [{"name": "accuracy", "value": 0.94666666
}], "failed": false, "metadata": {"pod_name": "orchestrate-55471-pv4hs"}}
[orchestrate-55471-cd5f2] Observation data: {"suggestion": "21807872", "values": [{"name": "accuracy", "value": 0.95333333
}], "failed": false, "metadata": {"pod_name": "orchestrate-55471-cd5f2"}}
[orchestrate-55471-pv4hs] /usr/local/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
```

View Experiment Logs Across Multiple Workers



Track Metadata and Monitor Your Results

Automated Cluster Management

Training Resnet-50 on ImageNet takes **10 hours**

Tuning 12 parameters requires at least 120 distinct models

That equals **1,200 hours**, or **50 days**, of training time

While training on **20 machines**, wall-clock time is ~~50 days~~ 2.5 days

Failed Observations

Constraints

Uncertainty

Competing Objectives

Lengthy Training Cycles

Cluster Orchestration

SIGOPT

Try SigOpt Orchestrate: https://sigopt.com/orchestrate

Free access for Academics & Nonprofits: https://sigopt.com/edu

Solution-oriented program for the Enterprise: https://sigopt.com/pricing

Leading applied optimization research: https://sigopt.com/research

... and we're hiring! https://sigopt.com/careers

Thank you!

patrick@sigopt.com for additional questions.