



Machine Learning for CI

Kyra Wulffert

kwulffert@yahoo.com

Matthew Treinish

mtreinish@kortar.org

Andrea Frittoli

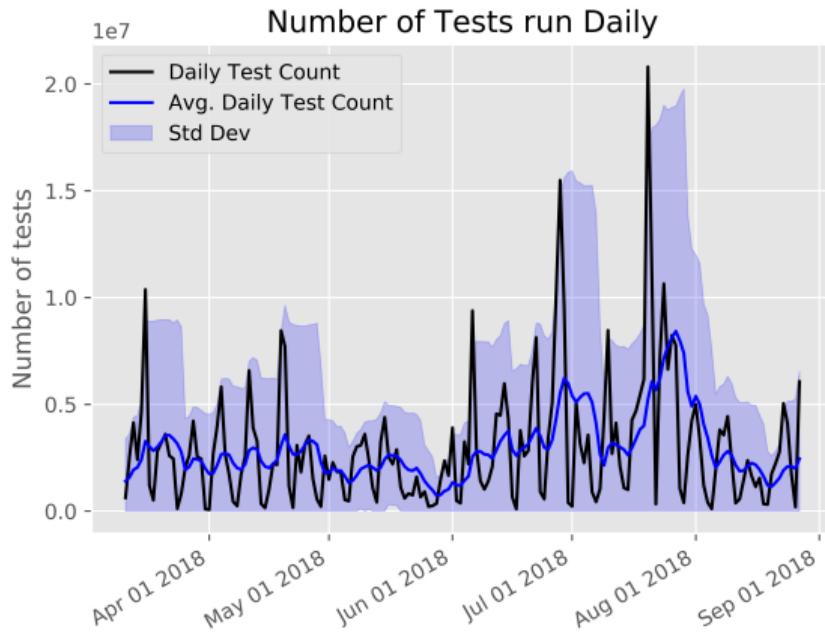
andrea.frittoli@gmail.com

August 29, 2018

https://github.com/afrittoli/ciml_talk



CI at Scale



Source: subunit2sql-graph dailycount

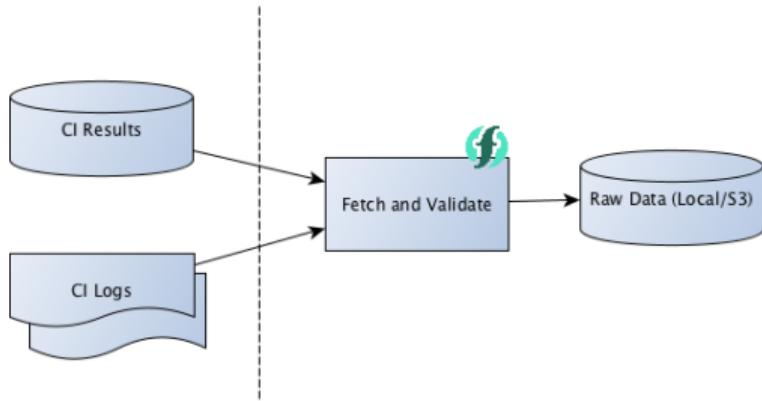
- ▶ Continuous Integration
- ▶ Continuous Log Data
- ▶ Lots of data, little time
- ▶ Triaging failures?
- ▶ AI to the rescue!

The OpenStack use case

- ▶ Integration testing in a VM
- ▶ System logs, application logs
- ▶ Dstat data
- ▶ Gate testing
- ▶ Not only OpenStack

Normalized system average load for different examples

Collecting data



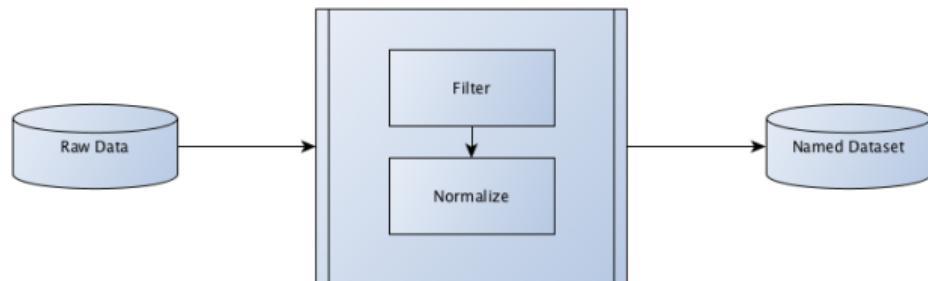
Data caching diagram

- ▶ Automation and repeatability
- ▶ Light-weight data validation
- ▶ Object storage for data
- ▶ Periodic Action on OpenWhisk

Experiment Workflow

- ▶ Visualize data
- ▶ Define a dataset
- ▶ Define an experiment
- ▶ Run the training
- ▶ Collect results
- ▶ Visualize data

```
# Build an s3 backed dataset
ciml-build-dataset --dataset cpu-load-1min-dataset \
--build-name tempest-full \
--slicer :2000 \
--sample-interval 10min \
--features-regex "(usr|1min)" \
--class-label status \
--tdt-split 7 0 3 \
--data-path s3://cimlrawdata \
--target-data-path s3://cimldatasets
```



Dataset preparation diagram

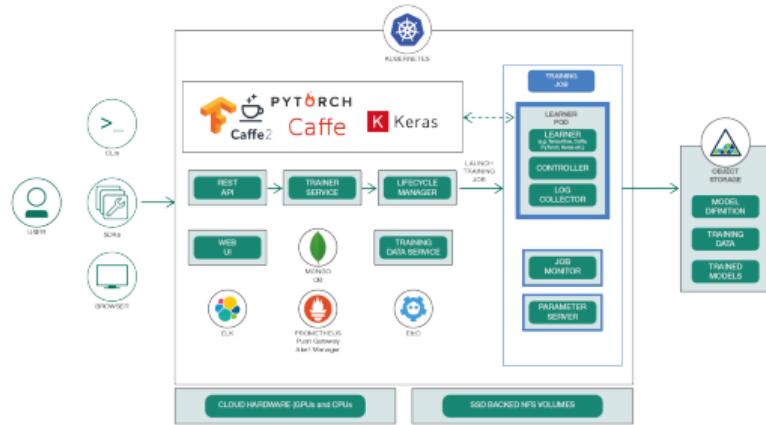
Experiment Workflow

- ▶ Visualize data
- ▶ Define a dataset
- ▶ **Define an experiment**
- ▶ Run the training
- ▶ Collect results
- ▶ Visualize data

```
# Define a local experiment
ciml-setup-experiment --experiment dnn-5x100 \
--dataset cpu-load-1min-dataset \
--estimator tf.estimator.DNNClassifier \
--hidden-layers 100/100/100/100/100 \
--steps $(( 2000 / 128 * 500 )) \
--batch-size 128 \
--epochs 500 \
--data-path s3://cimldatasets
```

```
# Train the model based on the dataset and experiment
# Store the evaluation metrics as a JSON file
ciml-train-model --dataset cpu-load-1min-dataset \
--experiment dnn-5x100 \
--data-path s3://cimldatasets
```

Training Infrastructure



FfDL Architecture - Source:

<https://developer.ibm.com/code/>

- ▶ TensorFlow Estimator API
- ▶ CIML wrapper
- ▶ ML framework interchangeable
- ▶ Training Options:
 - ▶ Run on a local machine
 - ▶ Helm deploy CIML, run in containers
 - ▶ Submit training jobs to Ffdl
 - ▶ Kubeflow

Prediction

- ▶ Event driven: near real time
- ▶ No request to serve the prediction to
- ▶ MQTT Trigger from the CI system
- ▶ CIML produces the prediction
- ▶ Trusted Source: Continuous Training

- ▶ CIML kubernetes app components:
 - ▶ MQTT Client receives events
 - ▶ Data module fetches and prepares data
 - ▶ TensorFlow wrapper issues the prediction
 - ▶ Example: comment back on Gerrit/Github

Data Selection

- ▶ What is dstat data
- ▶ Experiment reproducibility
- ▶ Dataset selection
 - ▶ Dstat feature selection
 - ▶ Data resolution (down-sampling)

Sample of dstat data

time	usr	used	writ	1m
16/03/2018 21:44	6.1	$7.36 \cdot 10^8$	$5.78 \cdot 10^6$	0.97
16/03/2018 21:44	7.45	$7.43 \cdot 10^8$	$3.6 \cdot 10^5$	0.97
16/03/2018 21:44	4.27	$7.31 \cdot 10^8$	$4.01 \cdot 10^5$	0.97
16/03/2018 21:44	1	$7.43 \cdot 10^8$	4,096	0.97
16/03/2018 21:44	0.5	$7.44 \cdot 10^8$	$1.5 \cdot 10^7$	0.97
16/03/2018 21:44	1.75	$7.31 \cdot 10^8$	4,096	0.97
16/03/2018 21:44	0.88	$7.43 \cdot 10^8$	4,096	0.9
16/03/2018 21:44	1.39	$7.31 \cdot 10^8$	$4.51 \cdot 10^5$	0.9
16/03/2018 21:45	1.01	$7.44 \cdot 10^8$	4,096	0.9
16/03/2018 21:45	0.75	$7.46 \cdot 10^8$	61,440	0.9
16/03/2018 21:45	1.26	$7.31 \cdot 10^8$	4,096	0.9
16/03/2018 21:45	1.13	$7.44 \cdot 10^8$	4,096	0.82
16/03/2018 21:45	5.77	$7.77 \cdot 10^8$	$1.72 \cdot 10^5$	0.82
16/03/2018 21:45	9.85	$8.31 \cdot 10^8$	$4.99 \cdot 10^6$	0.82
16/03/2018 21:45	3.88	$8.46 \cdot 10^8$	$8.25 \cdot 10^7$	0.82

Data Normalization

► Unrolling

Sample of unrolled data

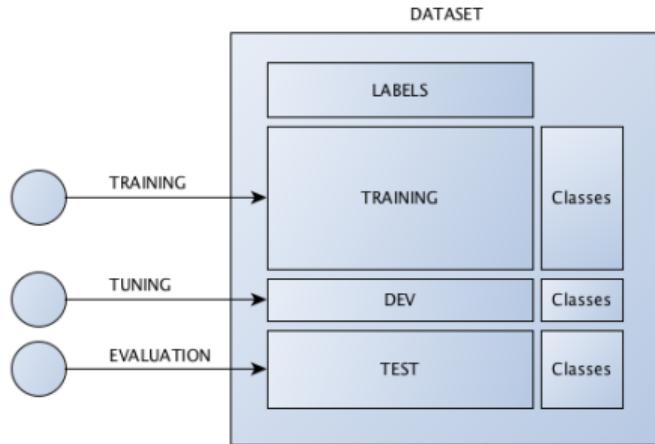
usr1	usr2	usr3	1m1	1m2	1m3
6.1	1.75	1.26	0.97	0.97	0.9
5.9	1.5	3.1	0.9	0.92	0.97
5.8	1.76	2.2	0.89	0.91	0.94

► Normalizing

Sample of normalized data

usr1	usr2	usr3	1m1	1m2	1m3
0.6	0.3	-0.5	0.6	0.6	-0.5
-0.1	-0.7	0.5	-0.3	-0.2	0.5
-0.4	0.3	0	-0.4	-0.4	0

Building the dataset

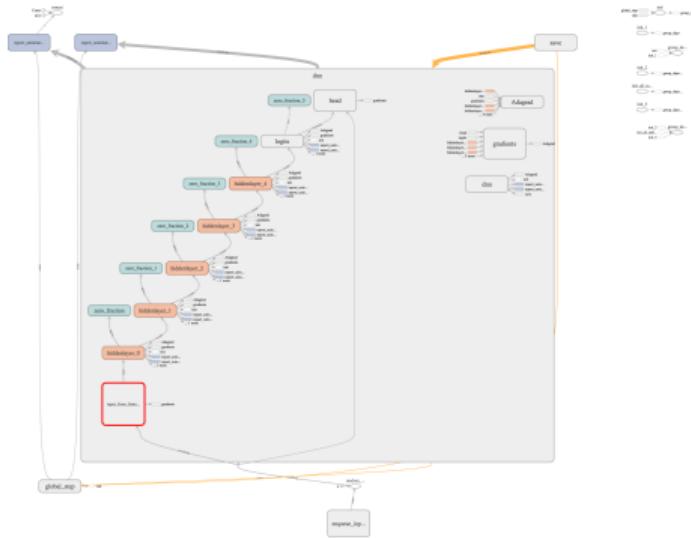


Structure of a dataset

- ▶ Split in training, dev, test
- ▶ Obtain classes
- ▶ Store normalized data on s3
- ▶ Input function for training
- ▶ Input function for evaluation

DNN - Binary Classification

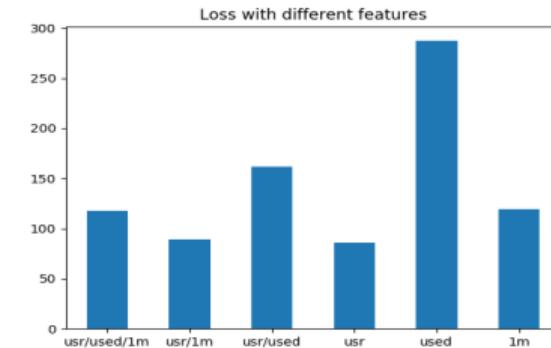
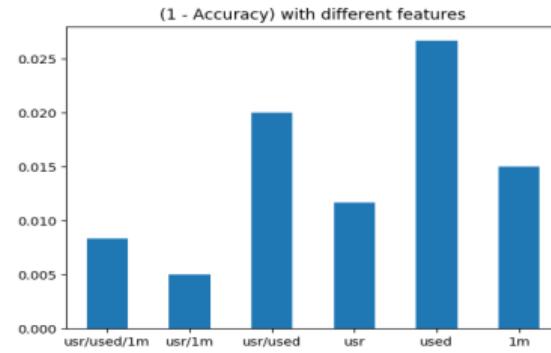
- ▶ Classes: Passed or Failed
- ▶ Supervised training
- ▶ TensorFlow *DNNClassifier*, classes=2
- ▶ Dataset:
 - ▶ CI Job "tempest-full"
 - ▶ Gate pipeline only
 - ▶ 2000 examples, 1400 training, 600 test
- ▶ Hyper-parameters:
 - ▶ Activation function: ReLU
 - ▶ Output layer: Sigmoid
 - ▶ Optimizer: Adagrad
 - ▶ Learning rate (initial): 0.05
 - ▶ 5 hidden layers, 100 units per layer
 - ▶ Batch Size: 128, Epochs: 500



Network Graph - Source: TensorBoard

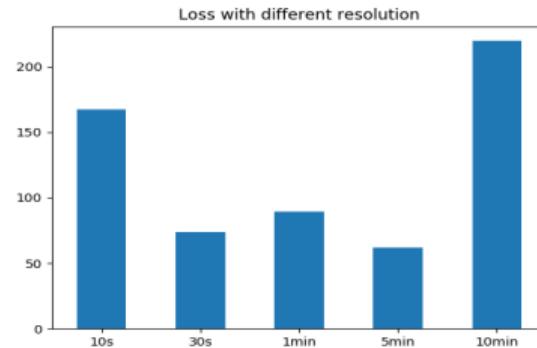
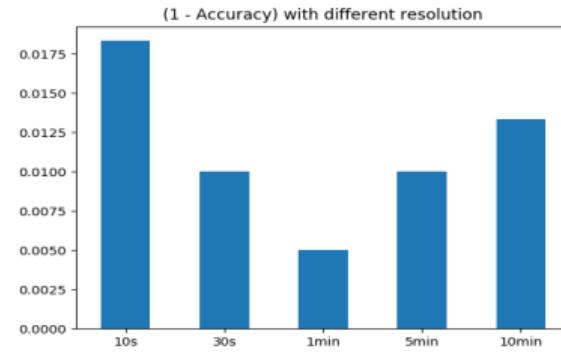
DNN - Binary Classification

- ▶ Selecting the best feature set
- ▶ Primary metric: accuracy
- ▶ Aim for lower loss, caveat: overfitting
- ▶ Key:
 - ▶ **usr**: User CPU
 - ▶ **used**: Used Memory
 - ▶ **1m**: System Load - 1min Average
 - ▶ Data Resolution: **1min**
 - ▶ Source: TensorFlow evaluation
- ▶ Winner: (**usr**, **1m**) tuple
- ▶ Accuracy achieved: **0.995**
- ▶ 3 mistakes on a 600 test set



DNN - Binary Classification

- ▶ Selecting the data resolution
- ▶ Primary metric: accuracy
- ▶ Aim for lower loss, caveat: overfitting
- ▶ Note: careful with NaN after down-sampling
- ▶ Key:
 - ▶ Original data frequency: 1s
 - ▶ x-axis: new sampling rate
 - ▶ Features: (**usr**, **1m**)
 - ▶ Source: TensorFlow evaluation
- ▶ Winner: **1min**
- ▶ Accuracy achieved: **0.995**
- ▶ 3 mistakes on a 600 test set



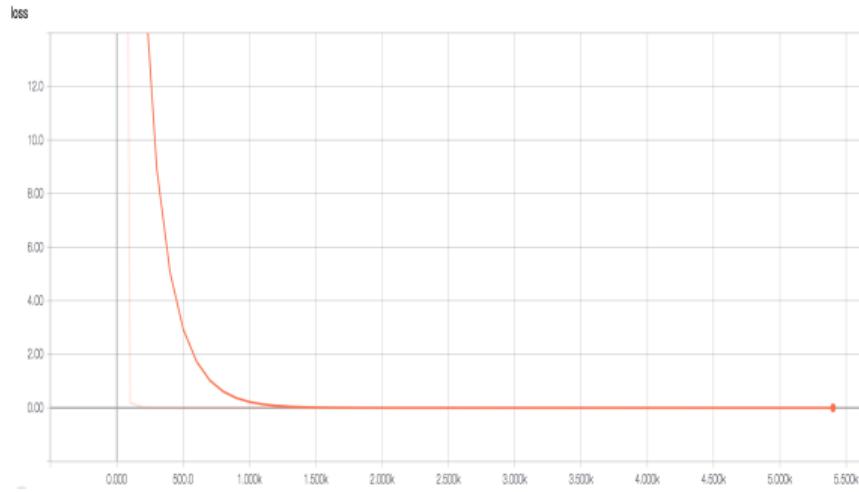
Changing test job

metric	tempest-full	tempest-full-py3
accuracy	0.997	0.943
loss	65.497	242.369
auc_precision_recall	0.965	0.608

- ▶ Train with "tempest-full"
- ▶ Evaluating with "tempest-full-py3"
 - ▶ Similar setup, uses python3
 - ▶ It does not include swift and swift tests
 - ▶ 600 examples evaluation set
- ▶ Dataset and training setup:
 - ▶ Features: (usr, 1m)
 - ▶ Resolution: 1min
 - ▶ Same hyper-parameters

Binary Classification - Summary

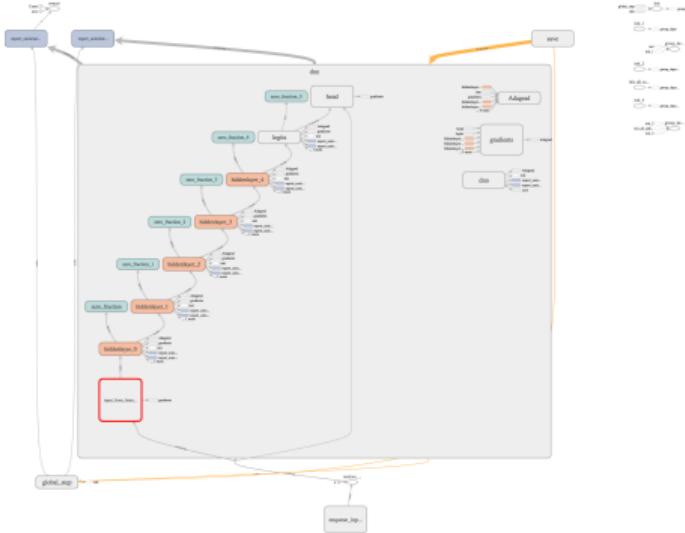
- ▶ User CPU and 1min Load Avg
- ▶ Resolution: 1 minute is enough
- ▶ High accuracy: **0.995**
- ▶ High auc_precision_recall: **0.945**
- ▶ A trained model might be applicable to similar CI jobs



Training Loss - usr/1m, 1min - Source: TensorBoard

DNN - Multi Class

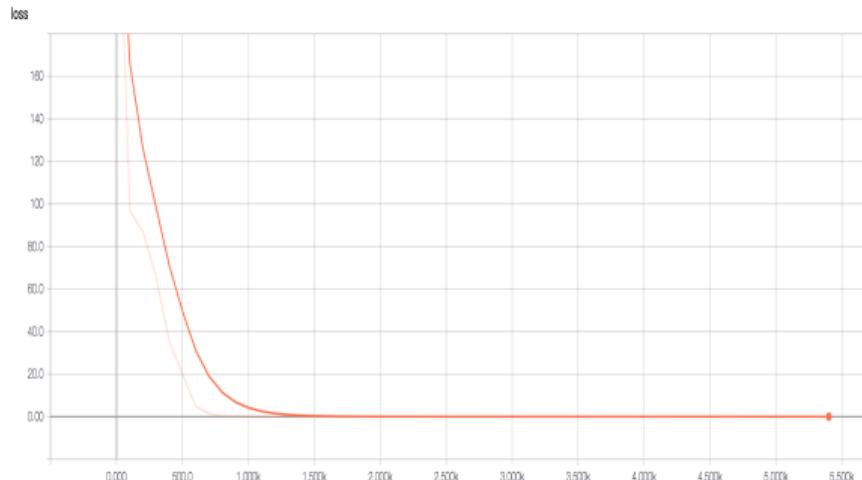
- ▶ Classes: Hosting Cloud Provider
- ▶ Supervised training
- ▶ TensorFlow *DNNClassifier*, classes=9
- ▶ Dataset:
 - ▶ CI Job "tempest-full"
 - ▶ Gate pipeline only
 - ▶ 2000 examples, 1400 training, 600 test
- ▶ Hyper-parameters:
 - ▶ Activation function: ReLU
 - ▶ Output layer: Sigmoid
 - ▶ Optimizer: Adagrad
 - ▶ Learning rate (initial): 0.05
 - ▶ 5 hidden layers, 100 units per layer
 - ▶ Batch Size: 128, Epochs: 500



Network Graph - Source: TensorBoard

DNN - Multi Class

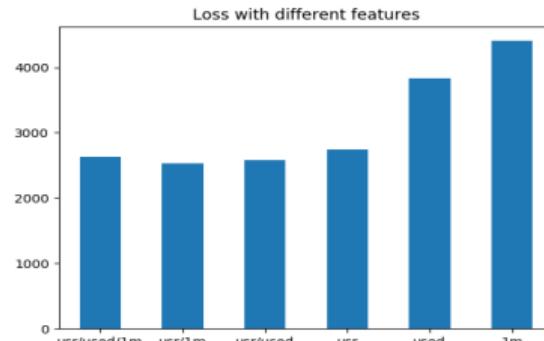
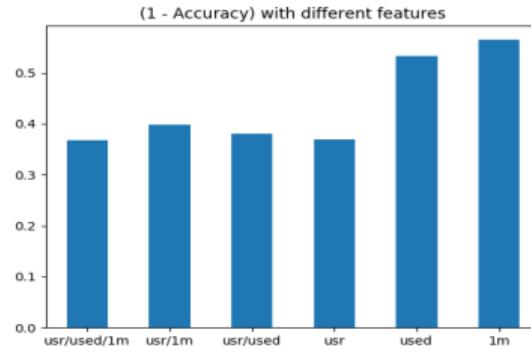
- ▶ Features: (usr, 1m)
- ▶ Resolution: 1min
- ▶ Loss converges, but...
- ▶ Evaluation accuracy achieved:
0.668
- ▶ Not good!



Training Loss - usr/1m, 1min - Source: TensorBoard

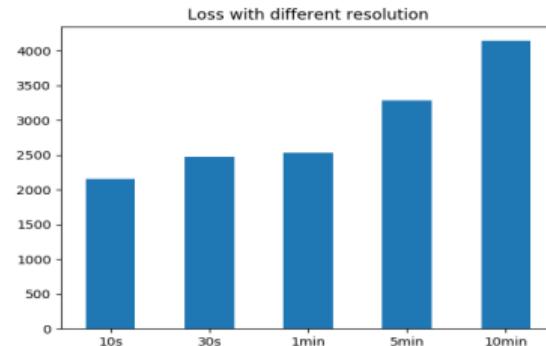
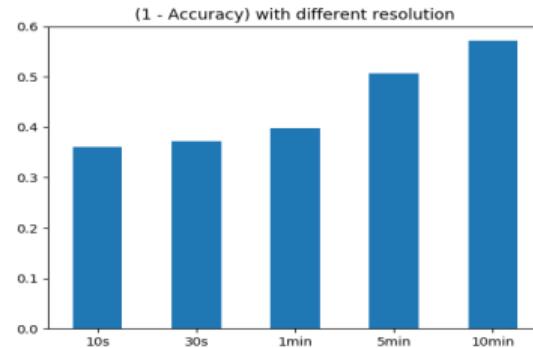
Multi Class - Different Features

- ▶ Try different combinations of features
- ▶ Primary metric: accuracy
- ▶ Aim for lower loss, caveat: overfitting
- ▶ Key:
 - ▶ **usr**: User CPU
 - ▶ **used**: Used Memory
 - ▶ **1m**: System Load - 1min Average
 - ▶ Data Resolution: **1min**
 - ▶ Source: TensorFlow evaluation output
- ▶ No real improvement
- ▶ Best accuracy achieved: **0.668**
- ▶ Adding Disk I/O or process data does not help either



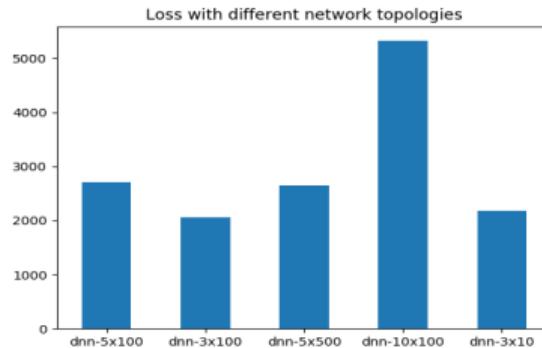
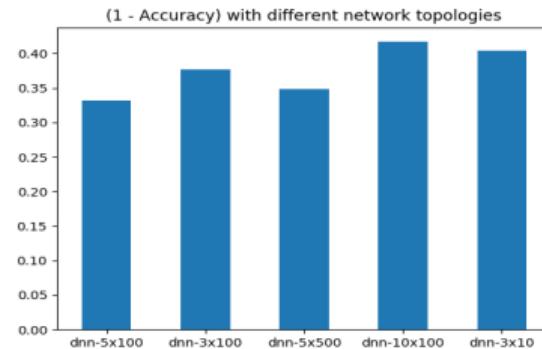
Multi Class - Changing Resolution

- ▶ Trying to change the data resolution
- ▶ Primary metric: accuracy
- ▶ Aim for lower loss, caveat: overfitting
- ▶ Key:
 - ▶ Original data frequency: 1s
 - ▶ x-axis: new sampling rate
 - ▶ Features: (**usr, 1m**)
 - ▶ Source: TensorFlow evaluation
- ▶ No real improvement
- ▶ Best accuracy achieved: **0.668**



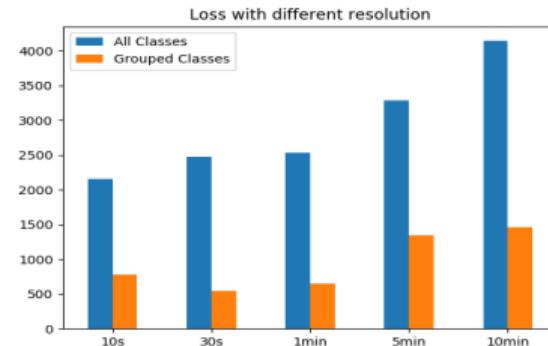
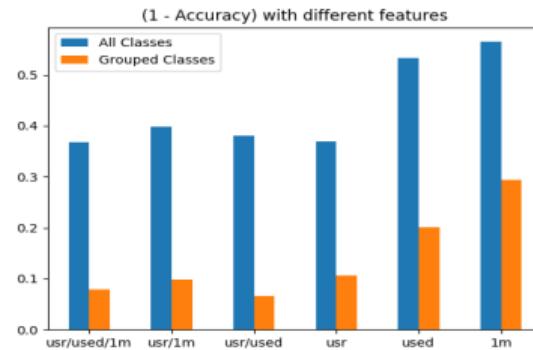
Multi Class - Network topology

- ▶ Trying to change the network depth
- ▶ Trying to change number of units per layer
- ▶ Primary metric: accuracy
- ▶ Aim for lower loss, caveat: overfitting
- ▶ Key:
 - ▶ x-axis: units and hidden layers
 - ▶ Features: (**usr**, **1m**)
 - ▶ Resolution: **1min**
 - ▶ Source: TensorFlow evaluation
- ▶ No real improvement
- ▶ Best accuracy achieved: **0.668**



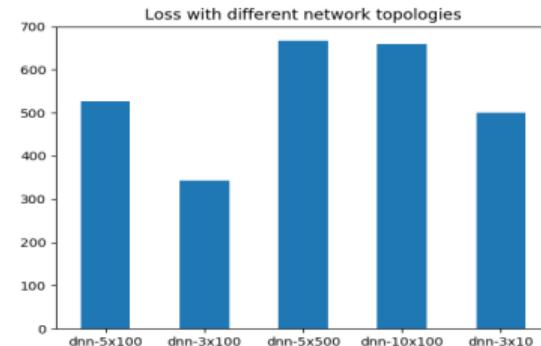
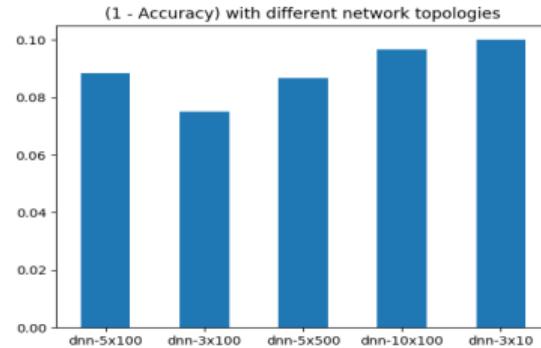
Multi Class - Reducing the number of classes

- ▶ Reducing the number of classes
 - ▶ Different regions from a Cloud Operator
 - ▶ Consider as a single class
 - ▶ New number of classes is **6**
- ▶ Experiments:
 - ▶ Train with different feature sets
 - ▶ Train with different resolutions
 - ▶ Source: TensorFlow evaluation
- ▶ Significant improvement!
- ▶ Best accuracy achieved: **0.902**
- ▶ What does that mean?



Multi Class - Tuning network topology

- ▶ Tuning network topology
- ▶ Experiments:
 - ▶ x-axis: units and hidden layers
 - ▶ Features: (**usr**, **1m**)
 - ▶ Resolution: **1min**
- ▶ Some improvement
- ▶ Winner: 3x100. Accuracy: **0.925**



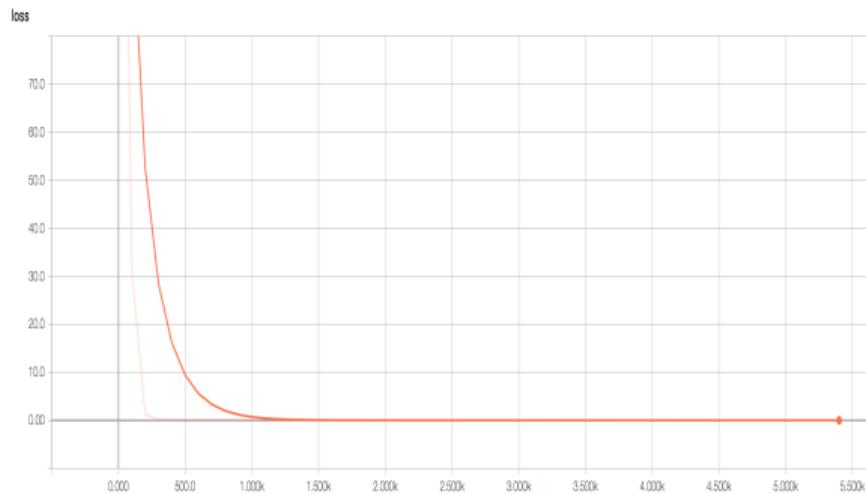
Multi Class - Changing test job

metric	tempest-full	tempest-full-py3
accuracy	0.925	0.775
average_loss	0.978	3.271
loss	586.713	1,962.447

- ▶ Train with "tempest-full"
- ▶ Evaluating with "tempest-full-py3"
 - ▶ Similar setup, uses python3
 - ▶ It does not include swift and swift tests
 - ▶ 600 examples evaluation set
- ▶ Dataset and training setup:
 - ▶ Features: (usr, 1m)
 - ▶ Resolution: 1min
 - ▶ Same hyper-parameters (dnn-3x100)

Multi Class - Summary

- ▶ User CPU and 1min Load Avg
- ▶ Resolution: 1 minute is enough
- ▶ Hyperparameters: 3 hidden layers, 100 units each
- ▶ Reasonable accuracy: **0.925**
- ▶ A trained model is not applicable to similar CI jobs



Training Loss - usr/1m, 1min, dnn3x100 - Source: TensorBoard

Conclusions

- ▶ Summary on DNN binary classification
- ▶ Summary on DNN multi class
- ▶ Collect data
- ▶ Know your data
- ▶ Work with cloud tools

Future Work

- ▶ Complete setup of the pipeline
- ▶ Human curated dataset for supervised training
- ▶ Making our life easier
- ▶ Integrate with real life CI system
- ▶ Explore job portability

Thank you!
Questions?

NATURAL DISASTERS ARE AMONG THE WORLD'S GREATEST CHALLENGES

HOW WOULD
22 MILLION
DEVELOPERS SOLVE
THESE GLOBAL
ISSUES IF GIVEN A
CHANCE TO **ANSWER
THE CALL?**

FIND OUT HOW AT
developer.ibm.com/callforcode



CALL FOR CODE

IBM

Call for Code Founding Partner



CALL FOR CODE



① REGISTER FOR THE CHALLENGE

callforcode.org/oss

Learn about the competition scope, prizes, rules and schedule



② SIGN UP FOR AN IBM ACCOUNT

ibm.biz/engagewithdeveloper

Get an account granting free tier access for an unlimited time



③ START BUILDING WITH CODE

developer.ibm.com/callforcode

Use code patterns in six tech areas to inspire your submission



④ ENGAGE THE COMMUNITY

Through the IBM Coder community and Slack

Meet advocates, brainstorm ideas, find a team, hear about updates



⑤ SUBMIT YOUR SOLUTION

ibm.biz/callforcodesubmit

Describe your app, name your team, link to your code and demo