The next Frontier: Distributed Deep Learning

@jim_dowling
“Methods that scale with computation are the future of AI”*
- Rich Sutton (Founding Father of Reinforcement Learning)

* https://www.youtube.com/watch?v=EeMCEQa85tw
Massive Increase in Compute for AI*

Distributed Systems

3.5 month-doubling time

*https://blog.openai.com/ai-and-compute
Model Parallelism

One Big Model on 3 GPUs

Copy of the Model on each GPU

Data Parallelism

Gradients

Batch Size of 3200

Distributed Filesystem

32 files

Aggregated

New Model

Barrier
ImageNet Challenge

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

[Image from https://www.slideshare.net/xavigiro/image-classification-on-imagenet-d114-2017-upc-deep-learning-for-computer-vision]
Improved Accuracy for ImageNet
Improvements in ImageNet – Accuracy

Top-1 Accuracy for ImageNet

- More Compute
- 1 Bn Images
- More Data

AutoML with RL

AutoML with GA

Google-1 (May '17)
Google-2 (March '18)
Facebook (May '18)

ImageNet Top-1

Facebook: [https://goo.gl/ERpJyr](https://goo.gl/ERpJyr)
Google-1: [https://goo.gl/EV7Xv1](https://goo.gl/EV7Xv1)
Google-2: [https://goo.gl/eidnyQ](https://goo.gl/eidnyQ)
Methods for Improving Model Accuracy

Reduced Generalization Error

Better Regularization Methods

Better Optimization Algorithms

Hyper Parameter Optimization

Design Better Models

Larger Training Datasets

Distribution
Faster Training of ImageNet

Single GPU

Clustered GPUs
Reduction in Training Time

| Table 1: Training time and top-1 1-crop validation accuracy with ImageNet/ResNet-50 |
|-------------------------------|-----------------|----------------|----------|-------|--------------|
|                               | Batch Size | Processor        | DL Library | Time  | Accuracy     |
| He et al. [7]                 | 256       | Tesla P100 x8    | Caffe    | 29 hours | 75.3%       |
| Goyal et al. [1]              | 8K        | Tesla P100 x256  | Caffe2   | 1 hour  | 76.3%       |
| Smith et al. [4]              | 8K→16K    | full TPU Pod     | TensorFlow | 30 mins | 76.1%       |
| Akiba et al. [5]              | 32K       | Tesla P100 x1024 | Chainer | 15 mins | 74.9%       |
| Jia et al. [6]                | 64K       | Tesla P40 x2048  | TensorFlow | 6.6 mins | 75.8%       |
| **This work**                 | 34K→68K   | Tesla V100 x2176 | NNL      | 224 secs | 75.03%      |

[From https://arxiv.org/abs/1811.05233 ]

300X in 3 years
## Scaling Efficiency

[From https://arxiv.org/abs/1811.05233 ]

<table>
<thead>
<tr>
<th>Processor</th>
<th>Interconnect</th>
<th>GPU scaling efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goyal et al. [1]</td>
<td>Tesla P100 x256</td>
<td>~90%</td>
</tr>
<tr>
<td>Akiba et al. [5]</td>
<td>Tesla P100 x1024</td>
<td>80%</td>
</tr>
<tr>
<td>Jia et al. [6]</td>
<td>Tesla P40 x2048</td>
<td>87.9%</td>
</tr>
<tr>
<td>This work</td>
<td>Tesla V100 x1088</td>
<td>91.62%</td>
</tr>
</tbody>
</table>

**Table 2: GPU scaling efficiency with ImageNet/ResNet-50 training**

- Network I/O Bound at 56 Gb/s
- Not Network I/O Bound at 200 Gb/s
ImageNet – Files/Sec Processed

<table>
<thead>
<tr>
<th></th>
<th>Files Processed (K/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook (June '17)</td>
<td></td>
</tr>
<tr>
<td>Google-1 (Dec '17)</td>
<td></td>
</tr>
<tr>
<td>Google-2 (March '18)</td>
<td></td>
</tr>
</tbody>
</table>

HDFS ~80K/s

FB - https://goo.gl/ERpJyr
Google-1: https://goo.gl/EV7Xv1
Google-2: https://goo.gl/eidnyQ
Remove Bottleneck and Keep Scaling

Files Processed (K/s)

Facebook  Google-1  Google-2  HopsFS

PLUG for Hops

Scale Challenge Winner (2017)

HopsFS - https://goo.gl/yFCsGc
All AI Roads Lead to Distribution

Distributed Deep Learning

- Parallel Experiments
- Hyper Parameter Optimization
- Larger Training Datasets
- AutoML
- Elastic Model Serving
- (Commodity) GPU Clusters
- Distributed Training
Data may be the new oil, but refined data is the fuel for AI
Machine Learning is a Data Distillery

- Distributed Training
- Feature Pipelines
- ETL Pipelines
Hopsworks: Data Distiller for ML
Frameworks for Distributed ML

• Managed Cloud Platforms
  - Google Cloud ML
  - Microsoft Batch AI
  - AWS Sagemaker

• On-Premise/Cloud Platforms
  - KubeFlow
  - Hopsworks
Challenges in Moving to “Distributed” Python
Classic Python Experience

pip install
conda install

Conda Env

PC

jupyter
“Cloud-Native” with KubeFlow
Python on Hopsworks

A Conda Environment per Project
Scalable Machine Learning Pipelines
Scalable ML Pipeline

Data Collection → Data Transformation & Verification → Feature Extraction → Experimentation → Training → Test → Serving

Single-Host TensorFlow

Single GPU

Standalone

Distributed Storage

Object Stores (S3, GCS), HDFS, Ceph

Potential Bottlenecks
Scalable ML Pipeline in Hopsworks

Data Collection → Data Transformation & Verification → Feature Extraction → Experimentation → Training → Test → Serving

PySpark → TensorFlow → Kubernetes

HopsFS
Spark for Distribution, HopsFS for State

Diagram:
- Driver
- Executor (with Conda Envs)
- Executor (with Conda Envs)
- TensorBoard/Logs
- HopsFS
- Model Serving
Hops Small Data ML Pipeline

Data Collection → Data Transformation & Verification → Feature Extraction → Experimentation → Training → Test → Serving

TensorFlow/Keras → TfServing

Kafka → HopsYARN → Kubernetes → HopsFS

Project Teams (Data Engineers/Scientists)
Hops Big Data ML Pipeline

Data Collection → Data Transformation & Verification → Feature Extraction → Experimentation → Training → Test → Serving

PySpark → TensorFlow/PyTorch → TfServing

Kafka → HopsYARN → Kubernetes → HopsFS

Project Teams (Data Engineers/Scientists)
Google TFX + Facets

- Jupyter Plugin
- Visualize data distributions
- Min/max/mean/media values for features
- Missing values in columns
- Test/train datasets

Visualize the relationship between the data points across the different features of a dataset.
features = ["Age", "Occupation", "Sex", ..., "Country"]

h = hdfs.get_fs()
with h.open_file(hdfs.project_path() + "/TestJob/data/census/adult.data", "r") as trainFile:
    train_data = pd.read_csv(trainFile, names=features,
                             sep=r\s*,\s*, engine='python', na_values="?")

test_data = ...

facets.overview(train_data, test_data)
facets.dive(test_data.to_json(orient='records'))
Now we want to pre-process some Images...
Small Data Preparation with tf.data API

def input_fn(batch_size):
    files = tf.data.Dataset.list_files(IMAGES_DIR)

def tfrecord_dataset(filename):
    return tf.data.TFRecordDataset(filename,
                                   num_parallel_reads=32, buffer_size=8*1024*1024)

dataset = files.apply(tf.data.parallel_interleave
                        (tfrecord_dataset, cycle_length=32, sloppy=True)
                        )
dataset = dataset.apply(tf.data.map_and_batch(parser_fn, batch_size,
                                             num_parallel_batches=4))
dataset = dataset.prefetch(4)
return dataset
Big Data Preparation with PySpark

images = spark.readImages(IMAGE_PATH, recursive = True, numPartitions=10, sampleRatio = 0.1).cache()

tr = (ImageTransformer().setOutputCol("transformed")
    .resize(height = 200, width = 200)
    .crop(0, 0, height = 180, width = 180) )
smallImages = tr.transform(images).select("transformed")

# Output .tfrecords using TensorFlowOnSpark utility
dfutil.saveAsTFRecords(smallImages, OUTPUT_DIR)
Estimator APIs in TensorFlow
# Estimators log to the Distributed Filesystem

```python
# over-simplified code – see notebook for full example
tf.estimator.RunConfig(
    'CollectiveAllReduceStrategy', model_dir,
tensorboard_logs,
checkpoints)
)
experiment.collective_all_reduce(…)
```

- HopsFS (HDFS)
  - `/Experiments/appId/run.ID/<name>`
  - `/Experiments/appId/run.ID/<name>/eval`
  - `/Experiments/appId/run.ID/<name>/checkpoint`
  - `/Experiments/appId/run.ID/<name>/*.ipynb`
  - `/Experiments/appId/run.ID/<name>/conda.yml`
# over-simplified code – see notebook

def distributed_training():
    def input_fn(): # return dataset
        model = …
        optimizer = …
        model.compile(…)
        rc = tf.estimator.RunConfig(‘CollectiveAllReduceStrategy’)
        keras_estimator = tf.keras.estimator.model_to_estimator(…)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)

    experiment.collective_all_reduce(distributed_training)
def distributed_training():
    from hops import tensorboard
    model_dir = tensorboard.logdir()
    def input_fn(): # return dataset
        model = …
        optimizer = …
        model.compile(…)
        rc = tf.estimator.RunConfig('CollectiveAllReduceStrategy')
        keras_estimator = keras.model_to_estimator(model_dir)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)
    experiment.allreduce(distributed_training)
def distributed_training():
    from hops import devices
    def input_fn(): # return dataset
        model = …
        optimizer = …
        model.compile(…)
        est.RunConfig(num_gpus_per_worker=devices.get_num_gpus())
        keras_estimator = keras.model_to_estimator(…)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)

    experiment.allreduce(distributed_training)
def distributed_training():
    def input_fn():  # return dataset
        model = …
        optimizer = …
        model.compile(…)
        rc = tf.estimator.RunConfig("CollectiveAllReduceStrategy")
        keras_estimator = keras.model_to_estimator(…)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)

    notebook = hdfs.project_path()+'/Jupyter/Experiment/inc.ipynb'
    experiment.allreduce(distributed_training, name='inception',
                         description='A inception example with hidden layers',
                         versioned_resources=[notebook])
Experiments/Versioning in Hopsworks
<table>
<thead>
<tr>
<th>id</th>
<th>user</th>
<th>name</th>
<th>start</th>
<th>finished</th>
<th>status</th>
<th>module</th>
<th>function</th>
<th>hyperparameter</th>
<th>metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>application_1538115949913_0002_1</td>
<td>Admin</td>
<td>fashion mnist grid search</td>
<td>September 29th 2018, 16:22:34.296</td>
<td>September 29th 2018, 16:30:55.242</td>
<td>SUCCEEDED</td>
<td>experiment</td>
<td>grid_search</td>
<td>learning_rate=0.001.drop out=0.7</td>
<td>0.832961797 714</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>t_id</code></td>
<td>application_1538115949913_0002_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>t_index</code></td>
<td>demo_tensorflow_admin000_experiments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>_score</code></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>t_type</code></td>
<td>experiments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>app_id</code></td>
<td>application_1538115949913_0002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>cuda</code></td>
<td>9.0.176.384.81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>description</code></td>
<td>Demonstration of running gridsearch hyperparameter optimization with fashion mnist</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>executors</code></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>finished</code></td>
<td>September 29th 2018, 16:30:55.242</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>function</code></td>
<td>grid_search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>gpus_per_executor</code></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>hops</code></td>
<td>2.8.2.5-SNAPSHOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>hops_py</code></td>
<td>2.7.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>hopsworks</code></td>
<td>0.6.0-SNAPSHOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Data Layer (Foundations)
Feeding Data to TensorFlow

Filesystem

Wrangling/Cleaning

DataFrame

Model Training

Filesystem

CPUs

CPUs

CPUs

CPUs

Project Hydrogen: Barrier Execution mode in Spark: JIRA: SPARK-24374, SPARK-24723, SPARK-24579
Existing Filesystems are not good enough?

Uber on Petastorm:
“[Using files] is hard to implement at large scale, especially using modern distributed file systems such as HDFS and S3 (these systems are typically optimized for fast reads of large chunks of data).”

https://eng.uber.com/petastorm/
with Reader('hdfs://myhadoop/dataset.parquet') as reader:
    dataset = make_petastorm_dataset(reader)
    iterator = dataset.make_one_shot_iterator()
    tensor = iterator.get_next()
with tf.Session() as sess:
    sample = sess.run(tensor)
    print(sample.id)
NVMe Disks – Game Changer

• HDFS (and S3) are designed around large blocks (optimized to overcome slow random I/O on disks), while new **NVMe hardware supports orders of magnitude faster random disk I/O.**

• Can we support faster random disk I/O with HDFS?  
  - Yes with HopsFS.
Small files on NVMe disks?*

• At Spotify’s HDFS:
  - 33% of files < 64KB in size
  - 42% of operations are on files < 16KB in size

• Similar statistics from Yahoo!’s Hadoop clusters.

• **Solution**: Keep the same large block size, but store the small files in HopsFS’ metadata layer.

*Size Matters: Improving the Performance of Small Files in Hadoop, Middleware 2018. Niazi et al
HopsFS – NVMe Performance for Small Files*

- HopsFS is HDFS with Distributed Metadata

- Small files stored replicated in the metadata layer on NVMe disks*
  - Read 10s of 1000s of images/second from HopsFS

*Size Matters: Improving the Performance of Small Files in Hadoop, Middleware 2018. Niazi et al
Model Serving
Model Serving on Kubernetes
Training-Serving Skew

• Monitor differences between performance during training and performance during serving.
  - Differences in how you process data in training vs serving.
  - Differences in the training data and live data for serving.
  - A feedback loop between your model and your algorithm.

• When to retrain?
  - If you look at the input data and use **covariant shift** to see when it deviates significantly from the data that was used to train the model on.
Model Serving

Features:
- Canary
- Multiple Models
- Scale-Out/In

Frameworks:
- TensorFlow Serving
- MLeap for Spark
- scikit-learn

Hopworks

REST API

Logging

kafka

Notifications

Retrain

Streaming

External Apps

Internal Apps

Load Balancer

Model Serving Containers

External Apps

Internal Apps

kubernetes

Frameworks:
- TensorFlow Serving
- MLeap for Spark
- scikit-learn

Notifications

Retrain

Streaming
Orchestrating ML Pipelines with Airflow

- Spark ETL
- Dist Training TensorFlow
- Test & Optimize
- Kubernetes ModelServing
- Spark Streaming Monitoring
HopsML

- Experiments
  - Dist. Hyperparameter Optimization
  - Versioning of Models/Code/Resources
  - Visualization with Tensorboard
  - Distributed Training with checkpointing
- Feature Store
- Model Serving and Monitoring
Want to try Hopsworks?

1. Register for an account at:
   www.hops.site

   Use an email address from a Swedish university or company.
Summary

- The future of Deep Learning is Distributed
  https://www.oreilly.com/ideas/distributed-tensorflow

- Hopsworks is a new Data Platform with first-class support for Python / Deep Learning / ML / Data Governance / GPUs

- Hopsworks is open-source

*https://twitter.com/karpathy/status/972701240017633281
The Team

Active:
Jim Dowling, Seif Haridi, Gautier Berthou, Salman Niazi, Mahmoud Ismail, Theofilos Kakantousis, Ermias Gebremeskel, Antonios Kouzoupis, Alex Ormenisan, Fabio Buso, Robin Andersson, August Bonds.

Alumni:

www.hops.io
@hopshadoop