Deep Learning on Apache Spark at CERN’s Large Hadron Collider with Analytics Zoo

September 26, 2019

Sajan Govindan, Intel
Luca Canali, CERN
Outline

Analytics Zoo & BigDL
• Deep Learning toolkit and framework for Apache Spark

CERN
• Data pipelines for High Energy Physics
Building End-to-End, Integrated Data Analytics & AI Solutions

BigDL
Distributed, High-Performance Deep Learning Framework for Apache Spark*
https://github.com/intel-analytics/bigdl

Analytics + AI Platform
Distributed TensorFlow*, Keras*, PyTorch* and BigDL on Apache Spark*
https://github.com/intel-analytics/analytics-zoo

Accelerating Data Analytics + AI Solutions At Scale

*Other names and brands may be claimed as the property of others.
Real-World ML/DL Applications Are Complex Data Analytics Pipelines

Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

BigDL
Bringing Deep Learning To Big Data Platform

- **Distributed** deep learning framework for Apache Spark
- Make deep learning more accessible to **big data users** and **data scientists**
  - Write deep learning applications as **standard Spark programs**
  - Run on existing Spark/Hadoop clusters (**no changes needed**)
- Feature parity with popular deep learning frameworks
  - E.g., Caffe, Torch, Tensorflow, etc.
- High performance (on CPU)
  - Powered by Intel MKL and multi-threaded programming
- Efficient scale-out
  - Leveraging Spark for distributed training & inference

[https://github.com/intel-analytics/BigDL](https://github.com/intel-analytics/BigDL)
[https://bigdl-project.github.io/](https://bigdl-project.github.io/)
# Analytics Zoo

## Unified Analytics + AI Platform for Big Data

<table>
<thead>
<tr>
<th>Use case</th>
<th>Recommendation</th>
<th>Anomaly Detection</th>
<th>Text Classification</th>
<th>Text Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Image Classification</td>
<td>Object Detection</td>
<td>Seq2Seq</td>
<td>Transformer</td>
</tr>
<tr>
<td>Feature Engineering</td>
<td>image</td>
<td>3D image</td>
<td>text</td>
<td>Time series</td>
</tr>
<tr>
<td>High Level Pipelines</td>
<td>tfpark: Distributed TF on Spark</td>
<td>Distributed Keras w/ autograd on Spark</td>
<td>nnframes: Spark Dataframes &amp; ML Pipelines for Deep Learning</td>
<td>Distributed Model Serving (batch, streaming &amp; online)</td>
</tr>
<tr>
<td>Backend/Library</td>
<td>TensorFlow</td>
<td>Keras</td>
<td>BigDL</td>
<td>NLP Architect</td>
</tr>
<tr>
<td></td>
<td>MKLDNN</td>
<td>OpenVINO</td>
<td>NLP Architect</td>
<td>Intel® Optane™ DCPMM</td>
</tr>
</tbody>
</table>

[https://github.com/intel-analytics/analytics-zoo](https://github.com/intel-analytics/analytics-zoo)
End-to-End Big Data Analytics and AI Pipeline

Seamless Scaling from Laptop to Production with

Prototype on laptop using sample data

Experiment on clusters with history data

Production deployment w/ distributed data pipeline

- "Zero" code change from laptop to distributed cluster
- Directly access production data (Hadoop/Hive/HBase) without data copy
- Easily prototype the end-to-end pipeline
- Seamlessly deployed on production big data clusters
Analytics Zoo
Unified Analytics + AI Platform for Big Data

Build end-to-end deep learning applications for big data
• Distributed *TensorFlow* on Spark
• *Keras* API (with autograd & transfer learning support) on Spark
• *nnframes*: native DL support for Spark DataFrames and ML Pipelines

Productionize deep learning applications for big data at scale
• Plain Java/Python *model serving* APIs (w/ OpenVINO support)
• Support Web Services, Spark, Flink, Storm, Kafka, etc.

Out-of-the-box solutions
• Built-in deep learning *models*, *feature engineering* operations, and reference *use cases*
Distributed TF & Keras on Spark

- Data wrangling and analysis using PySpark
- Deep learning model development using TensorFlow or Keras
- Distributed training / inference on Spark

Write TensorFlow code inline in PySpark program

```python
# pyspark code
train_rdd = spark.hadoopFile(...).map(...)
dataset = TFDataset.from_rdd(train_rdd, ...)

# tensorflow code
import tensorflow as tf
images, labels = dataset.tensors
with slim.arg_scope(lenet.lenet_arg_scope()):
    logits, end_points = lenet.lenet(images, ...)
loss = tf.reduce_mean(
    tf.losses.sparse_softmax_cross_entropy(logits=logits, labels=labels))

# distributed training on Spark
optimizer = TFOptimizer.from_loss(loss, Adam(...))
optimizer.optimize(end_trigger=MaxEpoch(5))
```
Spark Dataframe & ML Pipeline for DL

#Spark dataframe transformations
parquetfile = spark.read.parquet(…)
train_df = parquetfile.withColumn(…)

#Keras API
model = Sequential()
    .add(Convolution2D(32, 3, 3, activation='relu', input_shape=…)) \
    .add(MaxPooling2D(pool_size=(2, 2))) \
    .add(Flatten()).add(Dense(10, activation='softmax'))

#Spark ML pipeline
Estimater = NNEstimater(model, CrossEntropyCriterion()) \
    .setLearningRate(0.003).setBatchSize(40).setMaxEpoch(5) \
    .setFeaturesCol("image")
nnModel = estimater.fit(train_df)
Distributed model serving in **Web Service**, **Flink**, **Kafka**, **Storm**, etc.

- Plain Java or Python API, with OpenVINO and DL Boost (VNNI) support
Upcoming Analytics Zoo Features

• Distributed PyTorch on Spark

• Ray on Spark
  • Run Ray programs directly on standard Hadoop/YARN clusters

• AutoML support
  • Automatic feature generation, model selection and hyper-parameter tuning for time series prediction

• Cluster serving
  • Distributed, real-time (streaming) model serving with simple pub-sub interface
Growing Ecosystem & Use Cases

TECHNOLOGY
- bluedata
- cloudera
- CRAY THE SUPERCOMPUTER COMPANY
- databricks
- DELL EMC
- GIGASPACES Innovate with confidence
- Lightbend
- Qu bole

CLOUD SERVICE PROVIDERS
- Alibaba Cloud aliyun.com
- AWS
- Azure
- Tencent 腾讯
- Baidu 百度
- IBM Cloud
-キングソフト KINGSOFT Cloud Dataproc

END USERS
- cdhi
- Telefonica
- China Telecom
- THE WORLD BANK
- JD. COM
- CERN openlab
- Midea
- 韵达
- Cisco
- UnionPay 银联

software.intel.com/AlonBigData

Not a full list
*Other names and brands may be claimed as the property of others.
Outline

Analytics Zoo & BigDL
• Deep Learning toolkit and framework for Apache Spark

CERN
• Data pipelines for High Energy Physics
Largest machine in the world
27 km-long Large Hadron Collider (LHC)

Fastest racetrack on Earth
Protons travel at 99.9999991% of the speed of light

Emptiest place in the solar system
Particules circulate in the highest vacuum

Hottest spot in the galaxy
Lead ion collisions create temperatures 100,000x hotter than the hearth of the sun
CERN
Particle accelerators (LHC)
High Energy Physics Experiments
Key Data Processing Challenge

- Proton-proton collisions at LHC experiments happen at 40MHz.
  - Hundreds of TB/s of electrical signals that allow physicists to investigate particle collision events.
- Storage, limited by bandwidth
  - Currently, only 1 every 40K events stored to disk (~10 GB/s).

2018: 5 collisions/beam cross
Current LHC

2026: 400 collisions/beam cross
Future: High-Luminosity LHC upgrade
This can generate up to a petabyte of raw data per second. Reduced to GB/s by filtering in real time. Key is how to select potentially interesting events (trigger systems).
Deep Learning Pipeline for Physics Data

- R&D to improve the **quality of filtering systems**
  - Develop a “Deep Learning classifier” to be used by the filtering system
  - Goal: Reduce false positives -> do not store nor process uninteresting events
Analytics Platform at CERN

Integrating new “Big Data” components with existing infrastructure:
- Software distribution
- Data platforms
Hadoop and Spark Clusters at CERN

- Clusters:
  - YARN/Hadoop
  - Spark on Kubernetes
- Hardware: Intel based servers, continuous refresh and capacity expansion

<table>
<thead>
<tr>
<th>Accelerator logging (part of LHC infrastructure)</th>
<th>Hadoop - YARN - 30 nodes (Cores - 800, Mem - 13 TB, Storage – 7.5 PB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Purpose</td>
<td>Hadoop - YARN, 65 nodes (Cores – 1.3k, Mem – 20 TB, Storage – 12.5 PB)</td>
</tr>
<tr>
<td>Cloud containers</td>
<td>Kubernetes on Openstack VMs, Cores - 250, Mem – 2 TB Storage: remote HDFS or EOS (for physics data)</td>
</tr>
</tbody>
</table>
Extending Spark to Read Physics Data

- Physics data is stored in EOS system, accessible with xrootd protocol: extended HDFS APIs
- Stored in ROOT format: developed a Spark Datasource

- Currently: 300 PBs
- ~90 PBs per year of operation

https://github.com/cerndb/hadoop-xrootd
https://github.com/diana-hep/spark-root
Deep Learning Pipeline for Physics Data

Data Ingestion
- Read physics data and feature engineering

Feature Preparation
- Prepare input for Deep Learning network

Model Development
1. Specify model topology
2. Tune model topology on small dataset

Training
- Train the best model

Built with Apache Spark + Analytics Zoo + Python Notebooks
The Dataset

- Software simulators generate events and calculate the detector response
- Every event is a 801x19 matrix: for every particle momentum, position, energy, charge and particle type are given

```python
features = [  
    'Energy', 'Px', 'Py', 'Pz', 'Pt', 'Eta', 'Phi',  
    'vtxX', 'vtxY', 'vtxZ', 'ChPFIso', 'GammaPFIso', 'NeuPF Iso',  
    'isChHad', 'isNeuHad', 'isGamma', 'isEle', 'isMu', 'Charge'
]```
Data Ingestion

- Read input files (4.4 TB) from custom format
- Compute physics-motivated features
- Store to parquet format

54 M events
~4TB

Physics data storage

950 GBs
Stored on HDFS
Features Engineering

• From the 19 features recorded in the experiment, 14 more are calculated based on domain specific knowledge: these are called High Level Features (HLF)
• A sorting metric to create a sequence of particles to be fed to a sequence based classifier
Feature Preparation

- All features need to be converted to a format consumable by the network
- One Hot Encoding of categories
- Sort the particles for the sequence classifier with a UDF
- Executed in PySpark using Spark SQL and ML
Models Investigated

1. Fully connected feed-forward DNN with High Level Features
2. DNN with a recursive layer (based on GRUs)
3. Combination of (1) + (2)
Once the network topology is chosen, hyper-parameter tuning is done with scikit-learn+Keras and parallelized with Spark.
Model Development – DNN

- Model is instantiated with the Keras-compatible API provided by Analytics Zoo

```python
# Create keras like zoo model.
# Only need to change package name from keras to zoo.pipeline.api.keras

from zoo.pipeline.api.keras.optimizers import Adam
from zoo.pipeline.api.keras.models import Sequential
from zoo.pipeline.api.keras.layers.core import Dense, Activation

model = Sequential()
model.add(Dense(50, input_shape=(14,), activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))

creating: createZooKerasSequential
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasDense
```
A more complex topology for the network

```python
from zoo.pipeline.api.keras.models import Sequential
from zoo.pipeline.api.keras.layers.core import *
from zoo.pipeline.api.keras.layers.torch import Select
from zoo.pipeline.api.keras.layers.normalization import BatchNormalization
from zoo.pipeline.api.keras.layers.recurrent import GRU
from zoo.pipeline.api.keras.engine.topology import Merge

## GRU branch
gruBranch = Sequential()
.add(Masking(0.0, input_shape=(801, 19)))
.add(GRU(output_dim=50,
        return_sequences=True,
        activation='tanh'))
.add(Select([1, -1])

## HLF branch
hlfBranch = Sequential()
.add(Dropout(0.0, input_shape=(14,)))

## Concatenate the branches
branches = Merge(layers=[gruBranch, hlfBranch], mode='concat')

## Create the model
model = Sequential()
.add(branches)
.add(BatchNormalization())
.add(Dense(2, activation='softmax'))
```

![Diagram of a more complex topology for the network]
Distributed Training

Instantiate the estimator using Analytics Zoo / BigDL

```python
# Create SparkML compatible estimator for deep learning training

from bigdl.optim.optimizer import EveryEpoch, Loss, TrainSummary, ValidationSummary
from zoo.pipeline.nnframes import *
from zoo.pipeline.api.keras.objectives import CategoricalCrossEntropy

estimator = NNMultiClassEstimator[nn.Module, CategoricalCrossEntropy]()
    .setOptMethod(Adam())
    .setBatchSize(BDLbatch)
    .setMaxEpoch(numEpochs)
    .setFeaturesCol(“MLF_input”)  
    .setLabelCol(“encoded_label”)  
    .setValidation(EveryEpoch(), val_df=testDF,  
    val_method=[Loss(CategoricalCrossEntropy())], batch_size=BDLbatch)
```

The actual training is distributed to Spark executors

```python
%%time
trained_model = estimator.fit(trainDF)
```

Storing the model for later use

```python
modelDir = logDir + ‘/nnmodels/MLFClassifier’
trained_model.save(modelDir)
```

![Graph showing HLF classifier loss over iterations](image)
Performance and Scalability of Analytics Zoo & BigDL

Analytics Zoo & BigDL scales very well in the range tested.
Results

- Trained models with Analytics Zoo and BigDL
- Met the expected accuracy results
Model Serving

- Using Apache Kafka and Spark
- In FPGA replacing/integrating current rule-based algorithms
Summary

• We have successfully developed a Deep Learning pipeline using Apache Spark and Analytics Zoo on Intel Xeon servers
  • The use case developed addresses the needs for higher efficiency in event filtering at LHC experiments
  • Spark, Python notebooks and Analytics Zoo provide intuitive APIs for data preparation at scale on existing Hadoop cluster and cloud
  • Analytics Zoo & BigDL successfully address the problems of scaling DL on Spark clusters running on Intel Xeon servers
• Code and data at: https://github.com/cerndb/SparkDLTrigger
Acknowledgements

• CERN Colleagues
  • Matteo Migliorini, Riccardo Castellotti, Viktor Khristenko
  • Hadoop, Spark and Streaming service
  • CERN openlab

• Analytics Zoo & BigDL team, Jiao(Jennie) Wang, Jason Dai

• Authors of “Topology classification with deep learning to improve real-time event selection at the LHC”, https://arxiv.org/abs/1807.00083, Thong Nguyen, Maurizio Pierini
Thank You

References:
BigDL: https://software.intel.com/bigdl
software.intel.com/AIonBigData

HEP use case, code and data at: https://github.com/cerndb/SparkDLTrigger