Analytics Zoo: Building Analytics and AI Pipelines for Apache Spark with BigDL

Michael Pittaro, Dell EMC
Radhika Rangarajan, Intel

SPARK+AI SUMMIT 2018

THE JOURNEY TO PRODUCTION AI

TOTAL TIME TO SOLUTION (TTS)

Phase0 | Phase1 | Phase2 | Phase3 | Phase4

1 Frame Opportunity

- Define the problem
- Assess data needs

2 Hypotheses

- Identify software model needs
- Categorize existing resources

3 Data Preparation

- Collect data
- Store data
- Explore data
- Label data
- Process data
- Test data

4 Modeling

- Experiment with models
- Develop a proof of concept
- Train model
- Measure performance
- Evaluate system costs

5 Deployment

- Integrate into workflow
- Build organizational and system level capabilities to scale

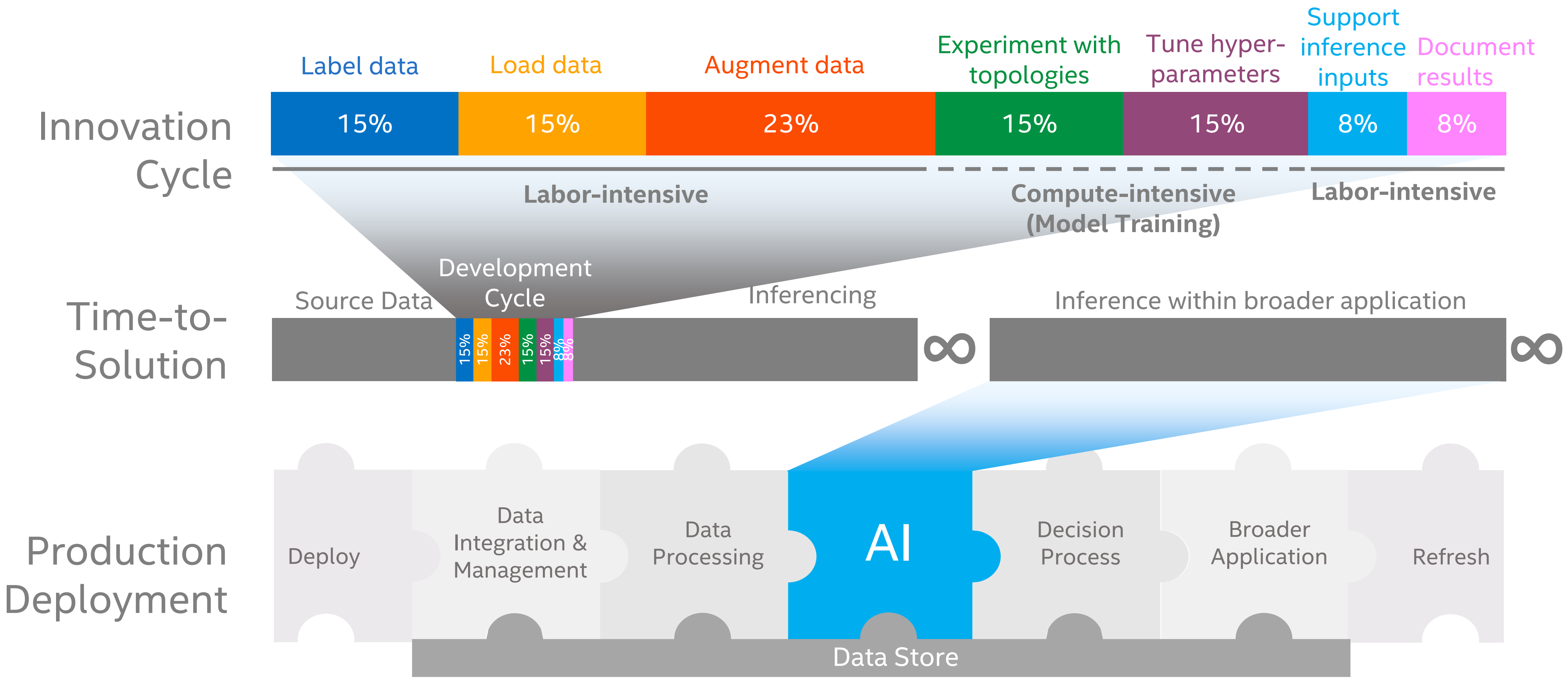
6 Iteration

- Increase robustness of model with additional data
- Re-training and refine model
- Tie inference to workflow decision making
- Optimize for end-user experience

7 Evaluation

- Measure utilization
- Explore need for additional processing hardware
- Understand bottlenecks to scaling
- Assess additional revenue generation models

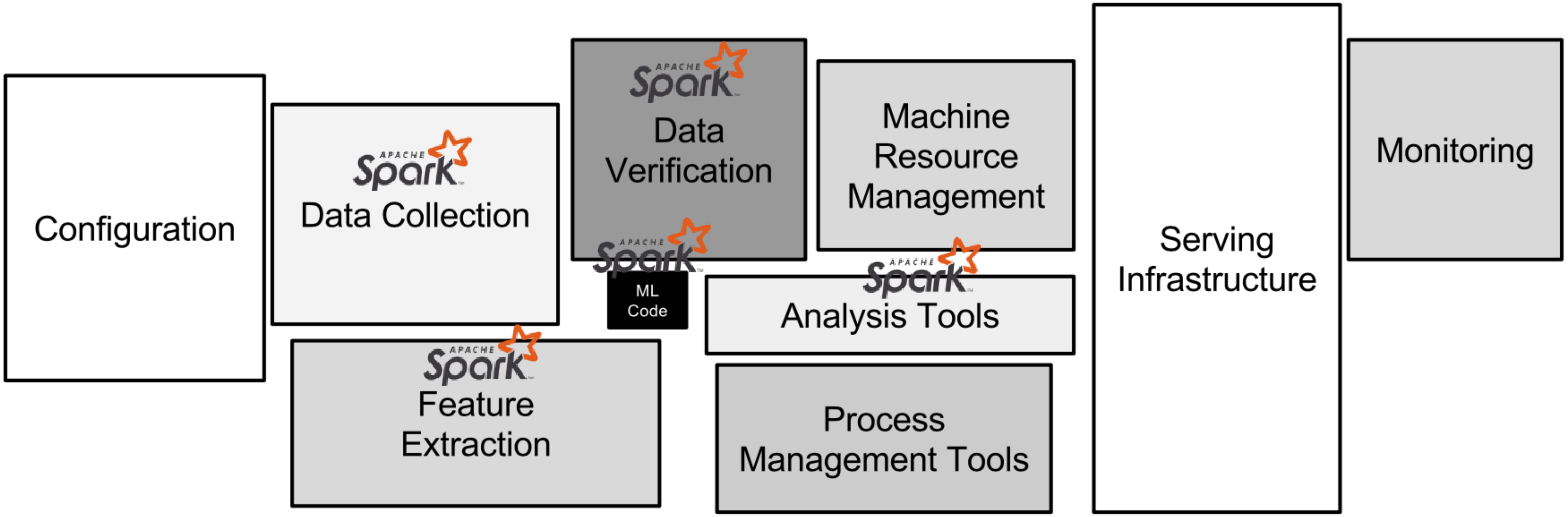
DEEP LEARNING IN PRACTICE



Time-to-solution is more significant than time-to-train

Source: Intel customer engagements

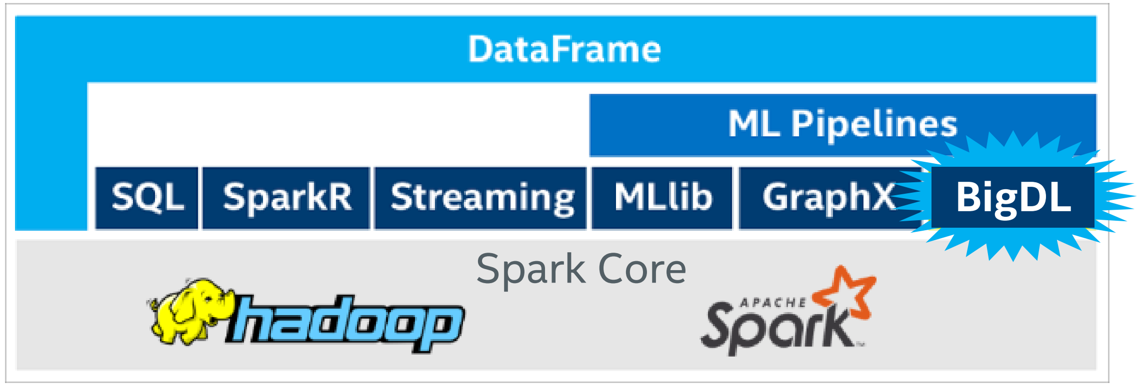
ARE YOU SEEING THE BIG PICTURE?



Supporting infrastructure is more significant than ML code

BIGDL: AI ON SPARK SOLUTION

High Performance Deep Learning on Apache Spark* on CPU Infrastructure[1]



BigDL is a distributed deep learning library for Apache Spark* that can run directly on top of existing Spark or Apache Hadoop* clusters with direct access to stored data and tool/workflow consistency!

No need to deploy costly accelerators, duplicate data, or suffer through scaling headaches!

Feature Parity with Caffe*/Torch*/ Tensorflow*

Lower TCO and improved ease of use with existing infrastructure

Deep Learning on Big Data Platform, Enabling **Efficient Scale-Out**

software.intel.com/bigdl

[1]*Open-source software is available for download at no cost; 'free' is also contingent upon running on existing idle CPU infrastructure where the operating cost is treated as a 'sunk' cost*

WHAT'S INSIDE BIGDL?

Lenet | Inception | Vgg | Resnet
Fast RCNN | SSD | RNN | LSTM
Deep Speech | Seq2Seq | Auto Encoder | Recommendation

Examples | Documents
Notebook | Tensor Board
Scala | Python

Layers
Spatial Convolution | Spatial MaxPooling | Volumetric Convolution | Volumetric MaxPooling | Fully Connected
ReLu | LRN | RNN | Batch Normalization | Other Layers

Optimization
SGD | Adagrad | LBFGS | Adamx
Adam | Adadelta | RMSprop

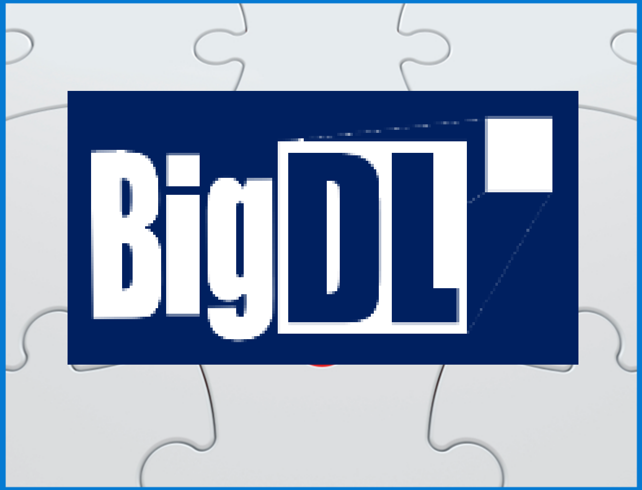
Criterion
Cross Entropy | ClassNLL | MSE
Dice Coefficient | Margin | Abs

Deep Learning Building Blocks
Layers, Optimizers, Criterion
Deep Learning Models

Scala* and Python* support
Spark ML Pipeline integration
Jupyter* notebook integration
Tensorboard* integration
OpenCV* support
Model Interoperability (Caffe*/TensorFlow*/Keras*)

ENABLING END TO END DL PIPELINES

How to Run Deep Leaning Workloads Directly on Big Data Platform?



- Integrated with Big Data ecosystem
- Massively distributed, shared-nothing
- Scale-out
- Send compute to data
- Fault tolerance
- Elasticity
- Incremental scaling
- Dynamic resource sharing
- …

Operationalizing deep learning at scale is as challenging as big data was a decade ago: steep learning curves due to complex APIs, expensive GPU infrastructures and disconnected operational/analytics feedback loop.

The combination of open source frameworks such as Spark and BigDL can make a real difference in simplifying AI adoption complexity across the enterprise.

BIGDL WORKLOADS….ACROSS THE INDUSTRY

CONSUMER

CALL CENTER ROUTING
IMAGE SIMILARITY SEARCH
SMART JOB SEARCH

HEALTH

ANALYSIS OF 3D MRI MODELS FOR KNEE DEGRADATION

FINANCE

FRAUD DETECTION
RECOMMENDATION
CUSTOMER/MERCHANT PROPENSITY

RETAIL

IMAGE FEATURE EXTRACTION

MANUFACTURING

STEEL SURFACE DEFECT DETECTION

SCIENTIFIC COMPUTING

WEATHER FORECASTING

AND OTHER EMERGING USAGES…

BIGDL…IT'S CATCHING ON…

Technology	Cloud Service Providers	End Users
bluedata	Alibaba Cloud aliyun.com	cdhi
cloudera	aws	GIGASPACES innovate with confidence
CRAY THE SUPERCOMPUTER COMPANY	Cloud Dataproc	JD.COM 京东
databricks	Azure	UnionPay 银联
DELL EMC	IBM Cloud	
KINGSOFT		
Lightbend		
Qubole		

BUT IT TAKES MORE THAN A FRAMEWORK…

AI IN PRODUCTION IS JUST BEGINNING…

AI ON SPARK IS STILL IN ITS INFANCY…

- **API Documentation is not enough**
- **Most real world examples involve proprietary trade secrets**
- **Research is great, but doesn't always map to production**
- **Where are the examples based on the real world?**

We know *what* to do, but *how* do we do it ?

ANALYTICS ZOO STACK

Reference Use Cases	Anomaly detection, sentiment analysis, fraud detection, chatbot, sequence prediction, etc.
Built-In Algorithms and Models	Image classification, object detection, text classification, recommendations, GAN, etc.
Feature Engineering and Transformations	Image, text, speech, 3D imaging, time series, etc.
High-Level Pipeline APIs	DataFrames, ML Pipelines, Autograd, Transfer Learning, etc.
Runtime Environment	Spark, BigDL, Python, etc.

Making it easier to build end-to-end analytics + AI applications

DIGGING DEEPER INTO THE ZOO

- **High level pipeline APIs**
  - ***nnframes***: native deep learning support in Spark DataFrames and ML Pipelines
  - ***autograd***: building custom layer/loss using auto differentiation operations
  - ***Transfer learning***: customizing pre-trained model for feature extraction or fine-tuning

- **Built-in deep learning models**
  - Object detection API
  - Image classification API
  - Text classification API
  - Recommendation API

- **Reference use cases:** a collection of end-to-end *reference use cases* (e.g., anomaly detection, sentiment analysis, fraud detection, image augmentation, object detection, variational autoencoder, etc.)

ANALYTICS ZOO : HIGH LEVEL PIPELINE API

Using high level pipeline APIs, users can easily build complex deep learning pipelines in just a few lines…

1. Load images into DataFrames using *NNImageReader*

```
from zoo.common.nncontext import *
from zoo.pipeline.nnframes import *
sc = get_nncontext()
imageDF = NNImageReader.readImages(image_path, sc)
```

2. Process loaded data using *DataFrames transformations*

```
getName = udf(lambda row: ...)
getLabel = udf(lambda name: ...)
df = imageDF.withColumn("name", getName(col("image"))) \
     .withColumn("label", getLabel(col('name')))
```

ANALYTICS ZOO : HIGH LEVEL PIPELINE API

3. Process images using built-in *feature engineering operations*

```
from zoo.feature.image import *
transformer = RowToImageFeature() \
        -> ImageResize(64, 64) \
        -> ImageChannelNormalize(123.0, 117.0, 104.0) \
        -> ImageMatToTensor() \
        -> ImageFeatureToTensor())
```

4. Load an existing model (pre-trained in Caffe), remove the last few layers and freeze the first few layers

```
from zoo.pipeline.api.net import *
full_model = Net.load_caffe(model_path)

# Remove layers after pool5/drop_7x7_s1
model = full_model.new_graph(["pool5/drop_7x7_s1"])

# freeze layers from input to pool4/3x3_s2 inclusive
model.freeze_up_to(["pool4/3x3_s2"])
```

ANALYTICS ZOO : HIGH LEVEL PIPELINE API

5. Add a few new layers (using *Keras-style API* and custom *Lambda* layer)

```
from zoo.pipeline.api.autograd import *
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import *

def add_one_func(x):
return x + 1.0

input = Input(name="input", shape=(3, 224, 224))
inception = model.to_keras()(input)
flatten = Flatten()(inception)
lambda = Lambda(function=add_one_func)(flatten)
logits = Dense(2)(lambda)
newModel = Model(inputNode, logits)
```

ANALYTICS ZOO : HIGH LEVEL PIPELINE API

6. Train model using Spark ML Pipelines

```
cls = NNClassifier(model, CrossEntropyCriterion(), transformer) \
      .setLearningRate(0.003).setBatchSize(40) \
      .setMaxEpoch(1).setFeaturesCol("image") \
      .setCachingSample(False)
nnModel = cls.fit(df)
```

Analytics Zoo : Built in Models

OBJECT DETECTION API

1. Download object detection models in Analytics Zoo – pre-trained on PASCAL VOC and COCO dataset

2. Load the image data and object detection model

```
from zoo.common.nncontext import get_nncontext
from zoo.models.image.objectdetection import *

spark = get_nncontext()
image_set = ImageSet.read(img_path, spark)
model = ObjectDetector.load_model(model_path)
```

ANALYTICS ZOO : BUILT IN MODELS

3. Use *Object Detection API* for off-the-shelf inference and visualization

```
output = model.predict_image_set(image_set)

visualizer = Visualizer(model.get_config().label_map(), \
                        encoding="jpg")
visualized = visualizer(output).get_image(to_chw=False) \
                               .collect()

for img_id in range(len(visualized)):
    cv2.imwrite(output_path + '/' + str(img_id) + '.jpg', \
          visualized[img_id])
```

COMMUNITY REQUESTS

- **Model serving pipelines**
  - *Web server, Spark Stream, Apache Storm, etc.*
- **Feature engineering**
  - *3D imaging, text, etc.*
- **Built-in deep learning models**
  - *Sequence-to-sequence, GAN, etc.*

THE ZOO CALLS...

LEARN

EXPLORE

ENGAGE

https://github.com/intel-analytics/analytics-zoo

https://github.com/intel-analytics/BigDL

software.intel.com/bigdl

UPCOMING SESSIONS...

JUNE 5th, TUESDAY @ 5.00 PM
Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL
Maurice Nsabimana (World Bank)Jiao Wang (Intel)
•DEEP LEARNING TECHNIQUES ROOM# 2009-2011 - 30 MINS

JUNE 6th, WEDNESDAY @ 11.00 AM
Building Deep Reinforcement Learning Applications on Apache Spark with Analytics Zoo using BigDL
Yuhao Yang (Intel)
DEEP LEARNING TECHNIQUES ROOM# 2009-2011 - 30 MINS

JUNE 6th, WEDNESDAY @ 4.20 PM
Using BigDL on Apache Spark to Improve the MLS Real Estate Search Experience at Scale
Sergey Ermolin (Big Data Technologies, Intel Corp)Dave Wetzel (MLS Listings)
SPARK EXPERIENCE AND USE CASES   ROOM# 2006-2008 - 30 MINS


intel