Analytics Zoo: Distributed Tensorflow, Keras and BigDL in production on Apache Spark

Jennie Wang, Big Data Technologies, Intel

Agenda

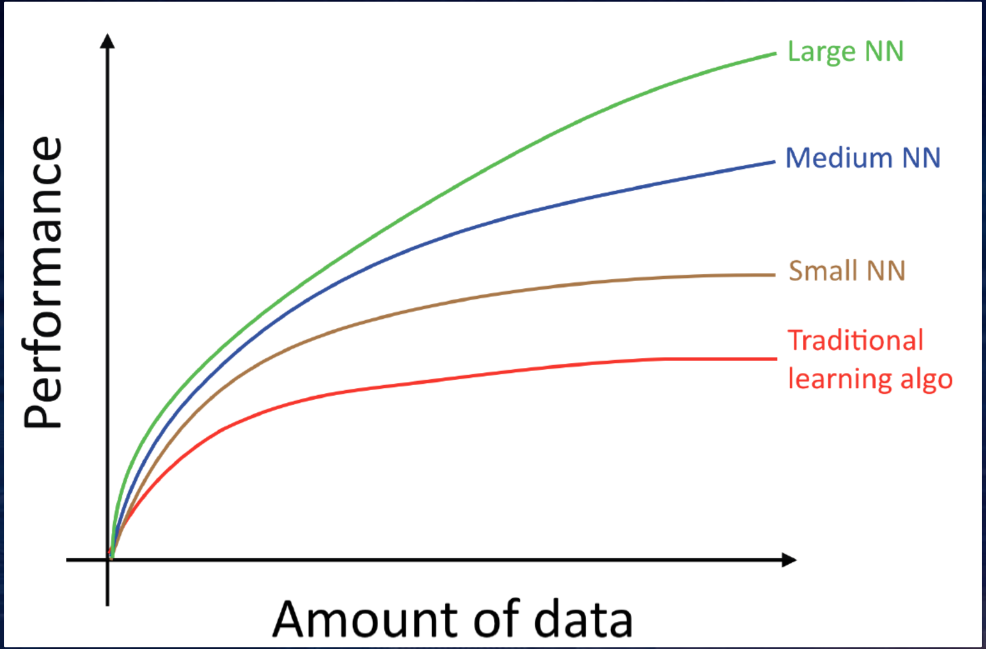
- Motivation
- BigDL
- Analytics Zoo
- Real-world applications
- Conclusion and Q&A

Motivations

Technology and Industry Trends

Real World Scenarios

Trend #1: Data Scale Driving Deep Learning Process



"Machine Learning Yearning", Andrew Ng, 2016

Trend #2: Real-World ML/DL Systems Are Complex Big Data Analytics Pipelines

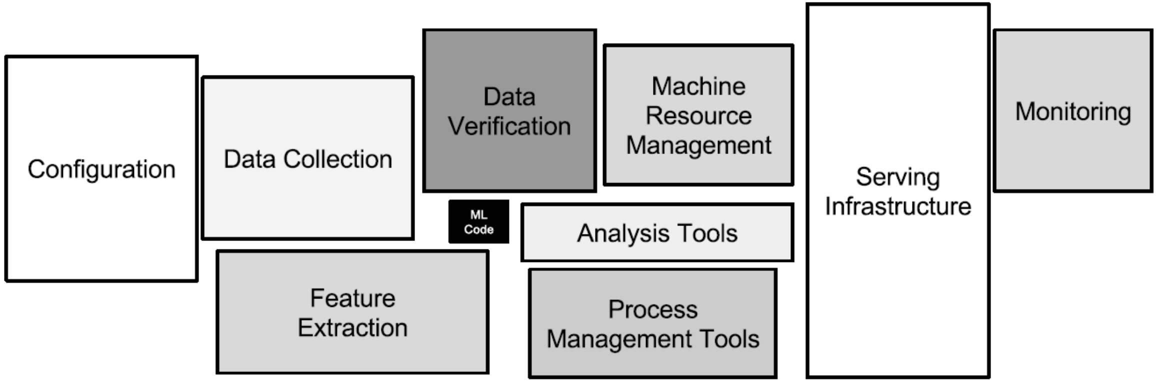


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

"Hidden Technical Debt in Machine Learning Systems", Sculley et al., Google, NIPS 2015 Paper

Trend #3: Hadoop Becoming the Center of Data Gravity

Why an Enterprise Data Hub ?

- Single place for all enterprise data... (unedited hi-resolution history of everything)
- Reduces Application Integration Costs
  - Connect once to Hub (N vs N^2 connections)
- Lowest unit cost data processing & storage platform
  - Open source S/W on commodity H/W (reliability in S/W not H/W)
  - Can mix H/W vendors means every expansion is competitively tendered
- Fast Standardised Provision
  - No custom design task, re-use Active Directory account/password processes
  - Reduces Shadow IT
- Secure (audited, E2E visibility/auditing, encryption)
  - Eliminate need for one off extracts

#StrataHadoop

Strata + Hadoop WORLD

Everyone is building Data Lakes

Goldman Sachs

- Universal data acquisition makes all big data analytics and reporting easier
- Hadoop provides a scalable storage with HDFS
- How will we scale consumption and curation of all this data?

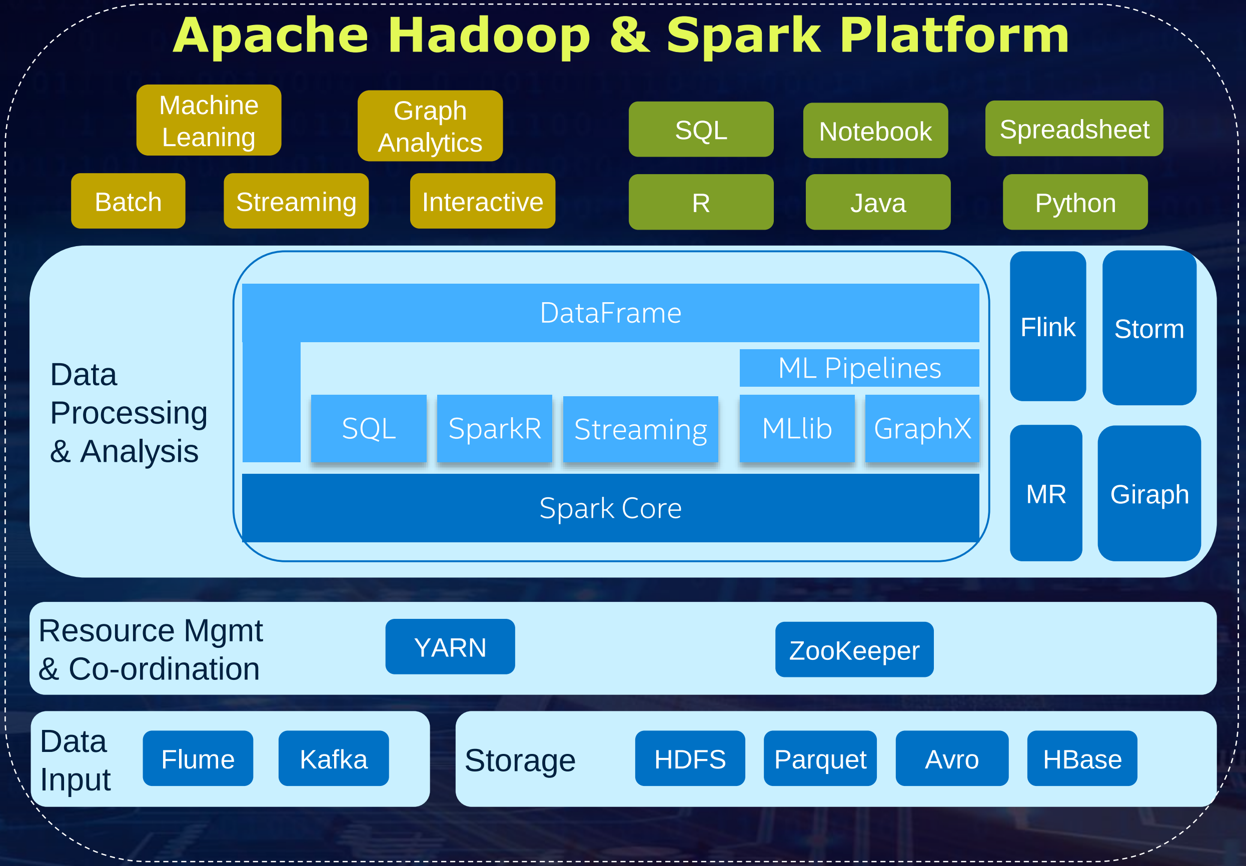
7

we BUILD

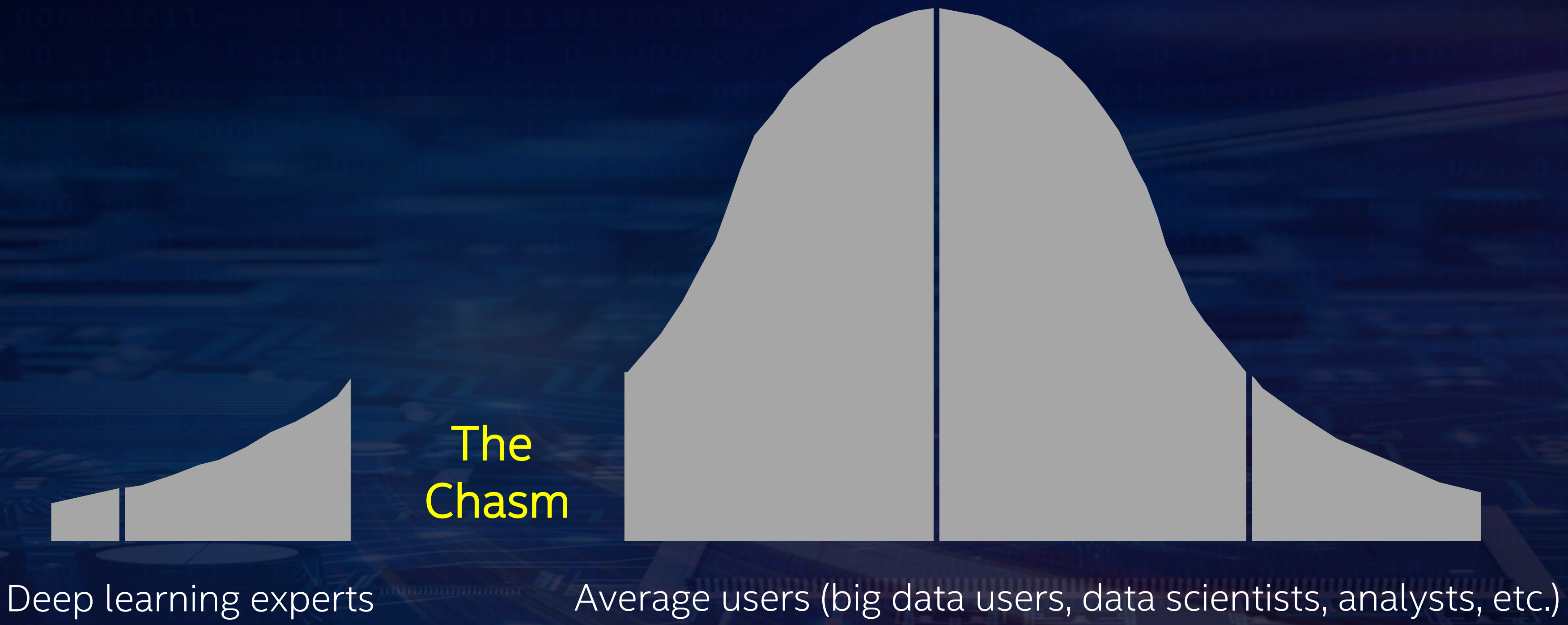
Phillip Radley, BT Group
Strata + Hadoop World 2016 San Jose

Matthew Glickman, Goldman Sachs
Spark Summit East 2015

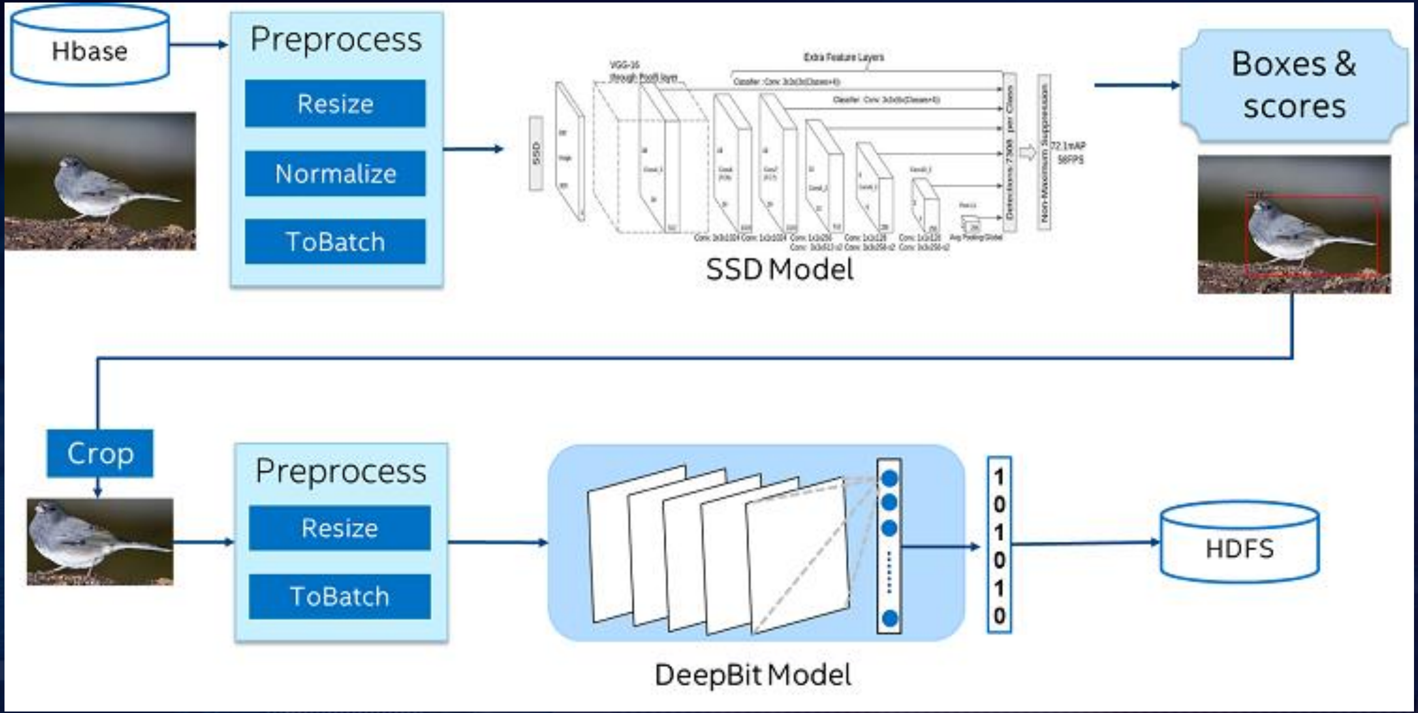
Unified Big Data Analytics Platform



Chasm b/w Deep Learning and Big Data Communities



Large-Scale Image Recognition at JD.com



Bridging the Chasm

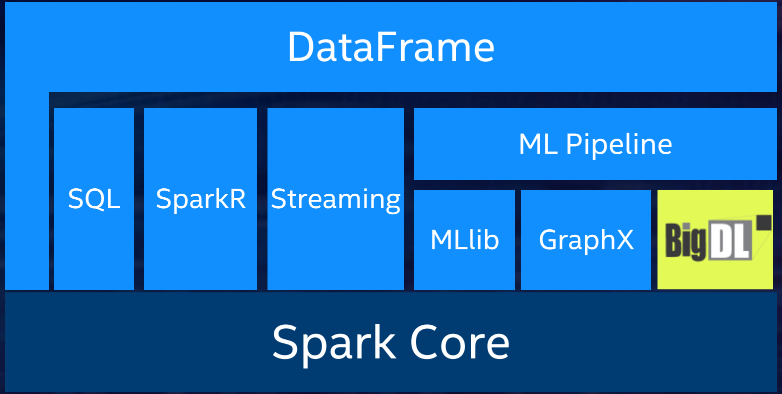
Make deep learning more accessible to big data and data science communities

- Continue the use of familiar SW tools and HW infrastructure to build deep learning applications
- Analyze “big data” using deep learning on the same Hadoop/Spark cluster where the data are stored
- Add deep learning functionalities to large-scale big data programs and/or workflow
- Leverage existing Hadoop/Spark clusters to run deep learning applications
    - Shared, monitored and managed with other workloads (*e.g., ETL, data warehouse, feature engineering, traditional ML, graph analytics, etc.*) in a dynamic and elastic fashion

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://bigdl-project.github.io/

BigDL Run as Standard Spark Programs

Standard Spark jobs

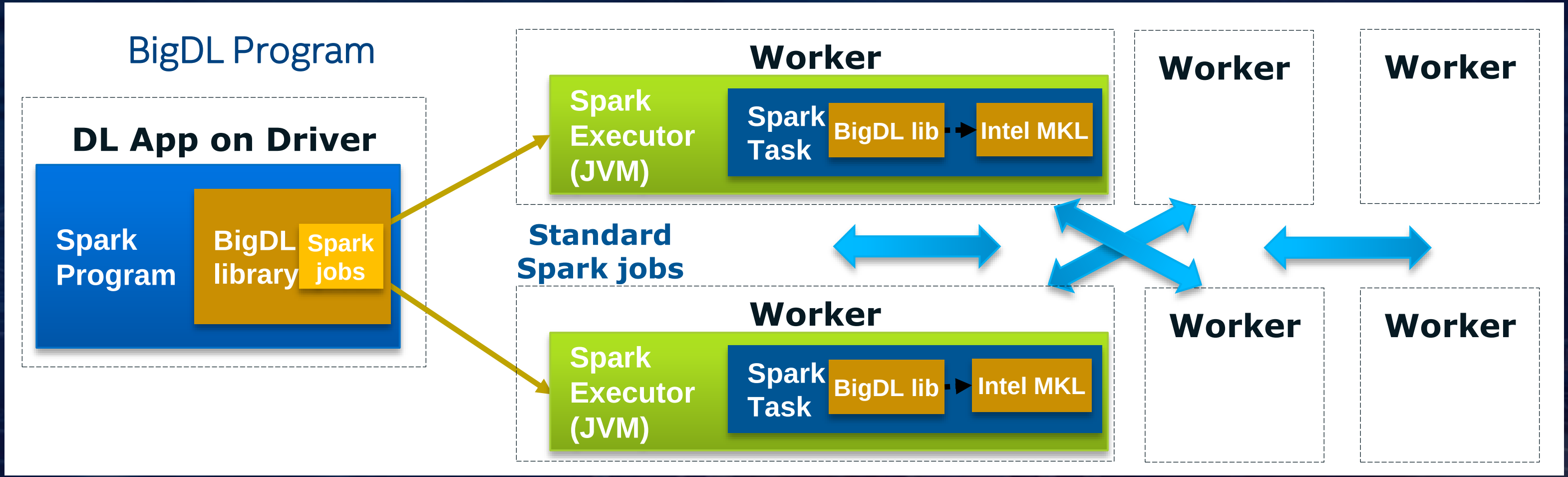
- No changes to the Spark or Hadoop clusters needed

Iterative

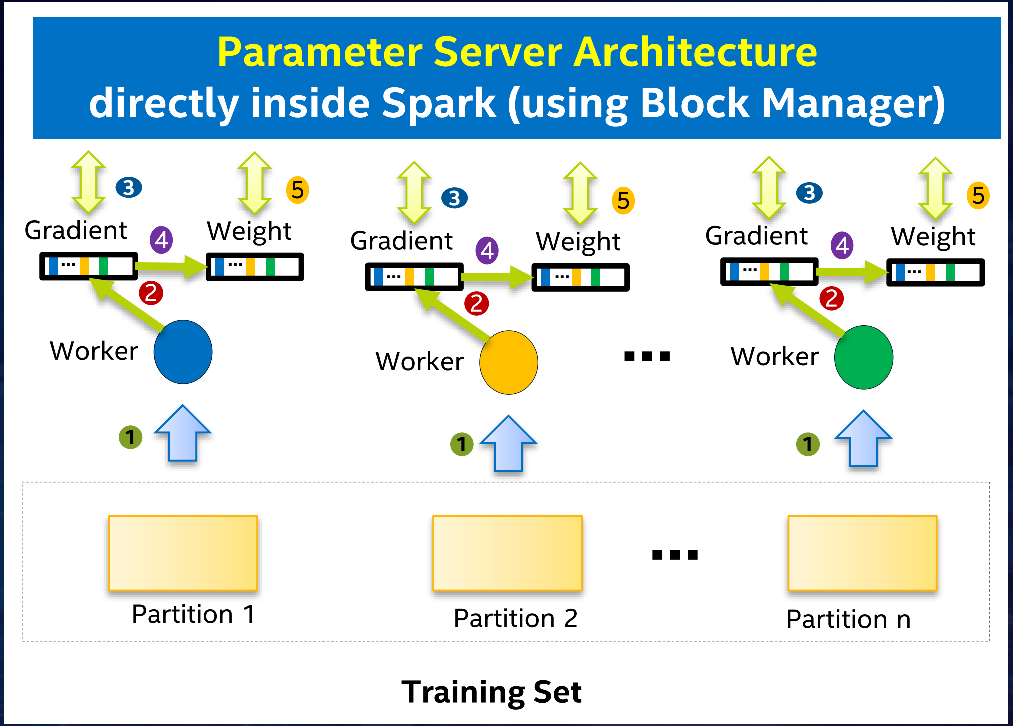
- Each iteration of the training runs as a Spark job

Data parallel

- Each Spark task runs the same model on a subset of the data (batch)

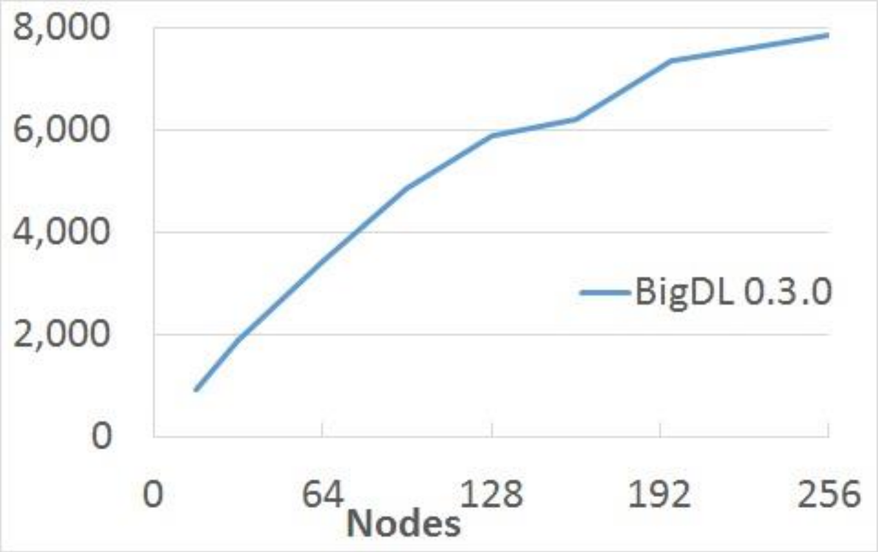


Distributed Training in BigDL



Peer-2-Peer **All-Reduce** synchronization

Training Scalability



Throughput of ImageNet Inception v1 training (w/ BigDL 0.3.0 and dual-socket Intel Broadwell 2.1 GHz); the throughput scales almost linear up to 128 nodes (and continue to scale reasonably up to 256 nodes).

Source: Scalable Deep Learning with BigDL on the Urika-XC Software Suite
(https://www.cray.com/blog/scalable-deep-learning-bigdl-urika-xc-software-suite/)

Analytics Zoo

A unified analytics + AI platform for distributed TensorFlow, Keras and BigDL on Apache Spark

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

Analytics Zoo

Unified Analytics + AI Platform for Big Data

Distributed TensorFlow, Keras and BigDL on Spark

Reference Use Cases	• Anomaly detection, sentiment analysis, fraud detection, image generation, chatbot, etc.
Built-In Deep Learning Models	• Image classification, object detection, text classification, text matching, recommendations, sequence-to-sequence, anomaly detection, etc.
Feature Engineering	Feature transformations for • Image, text, 3D imaging, time series, speech, etc.
High-Level Pipeline APIs	• Distributed TensorFlow and Keras on Spark • Native support for transfer learning, Spark DataFrame and ML Pipelines • Model serving API for model serving/inference pipelines
Backbends	Spark, TensorFlow, Keras, BigDL, OpenVINO, MKL-DNN, etc.

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

Analytics Zoo

Use Cases

- Anomaly Detection
- Sentiment Analysis
- Fraud Detection
- Image Generation
- Chatbot
- Recommedation
- Variational Autoencoder(VAE)
- Web services

High-Level Pipeline APIs

Build-in Deep Learning models

- Object Detection
- Image Classification
- Text Classification
- Recommenda tion
- Anomaly Detection
- Sequence-to-Sequence

- Distributed Tensorflow
- Keras-style APIs
- DataFrame and ML pipeline support
- Model Serving pipeline

Feature Engineering

- Image
- 3D Image
- Text
- Speech
- Time Series

Backends

- Spark
- Tensorflow
- Keras
- BigDL
- OpenVINO
- MKLDNN

Analytics Zoo

Build end-to-end deep learning applications for big data
- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engineering* operations for data preprocessing

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

Out-of-the-box solutions
- Built-in deep learning *models* and reference *use cases*

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engineering* operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models* and reference *use cases*

Distributed TensorFlow on Spark in Analytics Zoo

1. **Data wrangling and analysis using PySpark**

```
from zoo import init_nncontext
from zoo.pipeline.api.net import TFDataset

sc = init_nncontext()

#Each record in the train_rdd consists of a list of NumPy ndrrays
train_rdd = sc.parallelize(file_list)
  .map(lambda x: read_image_and_label(x))
  .map(lambda image_label: decode_to_ndarrays(image_label))

#TFDataset represents a distributed set of elements,
#in which each element contains one or more TensorFlow Tensor objects.
dataset = TFDataset.from_rdd(train_rdd,
                             names=["features", "labels"],
                             shapes=[[28, 28, 1], [1]],
                             types=[tf.float32, tf.int32],
                             batch_size=BATCH_SIZE)
```

Distributed TensorFlow on Spark in Analytics Zoo

2. **Deep learning model development using TensorFlow**

```
import tensorflow as tf

slim = tf.contrib.slim

images, labels = dataset.tensors
labels = tf.squeeze(labels)
with slim.arg_scope(lenet.lenet_arg_scope()):
     logits, end_points = lenet.lenet(images, num_classes=10, is_training=True)

loss = tf.reduce_mean(tf.losses.sparse_softmax_cross_entropy(logits=logits,
labels=labels))
```

Distributed TensorFlow on Spark in Analytics Zoo

3. Distributed training on Spark and BigDL

```
from zoo.pipeline.api.net import TFOptimizer
from bigdl.optim.optimizer import MaxIteration, Adam, MaxEpoch, TrainSummary

optimizer = TFOptimizer.from_loss(loss, Adam(1e-3))
optimizer.set_train_summary(TrainSummary("/tmp/az_lenet", "lenet"))
optimizer.optimize(end_trigger=MaxEpoch(5))
```

More Examples:

https://github.com/intel-analytics/analytics-zoo/blob/master/apps/tfnet/image_classification_inference.ipynb

https://github.com/intel-analytics/analytics-zoo/blob/master/pyzoo/zoo/examples/tensorflow/distributed_training/train_lenet.py

https://github.com/intel-analytics/analytics-zoo/blob/master/pyzoo/zoo/examples/tensorflow/distributed_training/train_mnist_keras.py

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engi*neering operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models* and reference *use cases*

Keras, Autograd &Transfer Learning APIs

1. Use transfer learning APIs to
   - Load an existing Caffe model
   - Remove last few layers
   - Freeze first few layers
   - Append a few layers

```
from zoo.pipeline.api.net import *
full_model = Net.load_caffe(def_path, model_path)
# Remove layers after pool5
model = full_model.new_graph(outputs=["pool5"])
# freeze layers from input to res4f inclusive
model.freeze_up_to(["res4f"])
# append a few layers
image = Input(name="input", shape=(3, 224, 224))
resnet = model.to_keras()(image)
resnet50 = Flatten()(resnet)
```

Build Siamese Network Using Transfer Learning

Keras, Autograd &Transfer Learning APIs

2. Use *Keras-style and autograd* APIs to build the Siamese Network

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

input = Input(shape=[2, 3, 226, 226])
features = TimeDistributed(layer=resnet50)(input)
f1 = features.index_select(1, 0) #image1
f2 = features.index_select(1, 1) #image2
diff = A.abs(f1 - f2)
fc = Dense(1)(diff)
output = Activation("sigmoid")(fc)
model = Model(input, output)
```

Build Siamese Network Using Transfer Learning

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engineering* operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models* and reference *use cases*

nnframes

Native DL support in Spark DataFrames and ML Pipelines

1. **Initialize *NNContext* and load images into *DataFrames* using *`NNImageReader`***

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

2. **Process loaded data using *`DataFrame`* transformations**

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

3. **Processing image using built-in *feature engineering* operations**

```
from zoo.feature.image import *
transformer = ChainedPreprocessing(
        [RowToImageFeature(), ImageChannelNormalize(123.0, 117.0, 104.0),
         ImageMatToTensor(), ImageFeatureToTensor()])
```

nnframes

Native DL support in Spark DataFrames and ML Pipelines

4. Define model using *Keras-style API*

```
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import *
model = Sequential()
          .add(Convolution2D(32, 3, 3, activation='relu', input_shape=(1, 28, 28))) \
          .add(MaxPooling2D(pool_size=(2, 2))) \
          .add(Flatten()).add(Dense(10, activation='softmax')))
```

5. Train model using *Spark ML Pipelines*

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

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engineering* operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models* and reference *use cases*

Working with Image

1. **Read images into local or distributed *ImageSet***

```
from zoo.common.nncontext import *
from zoo.feature.image import *
spark = init_nncontext()
local_image_set = ImageSet.read(image_path)
distributed_image_set = ImageSet.read(image_path, spark, 2)
```

2. **Image augmentations using built-in *ImageProcessing* operations**

```
transformer = ChainedPreprocessing([ImageBytesToMat(),
                     ImageColorJitter(),
                     ImageExpand(max_expand_ratio=2.0),
                     ImageResize(300, 300, -1),
                     ImageHFlip()])
new_local_image_set = transformer(local_image_set)
new_distributed_image_set = transformer(distributed_image_set)
```

Image Augmentations Using Built-in Image Transformations (w/ OpenCV on Spark)

Working with Text

1. **Read text into local or distributed *`TextSet`***

```
from zoo.common.nncontext import *
from zoo.feature.text import *
spark = init_nncontext()
local_text_set = TextSet.read(text_path)
distributed_text_set = TextSet.read(text_path, spark, 2)
```

2. **Build text transformation pipeline using built-in operations**

```
transformedTextSet = textSet.tokenize()            \
                            .normalize()           \
                            .word2idx()            \
                            .shapeSequence(len)    \
                            .generateSample()      \
```

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engineering* operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models* and reference *use cases*

POJO Model Serving API

```
import com.intel.analytics.zoo.pipeline.inference.AbstractInferenceModel;

public class TextClassification extends AbstractInferenceModel {
  public RankerInferenceModel(int concurrentNum) {
    super(concurrentNum);
  }
  ...
}

public class ServingExample {
  public static void main(String[] args) throws IOException {
    TextClassification model = new TextClassification();
    model.load(modelPath, weightPath);

    texts = …
    List<JTensor> inputs = preprocess(texts);
    for (JTensor input : inputs) {
      List<Float> result = model.predict(input.getData(), input.getShape());
      ...
  }
}
```

OpenVINO Support for Model Serving

```python
from zoo.common.nncontext import init_nncontext
from zoo.feature.image import ImageSet
from zoo.pipeline.inference import InferenceModel

sc = init_nncontext("OpenVINO Object Detection Inference Example")
images = ImageSet.read(options.img_path, sc,
                       resize_height=600, resize_width=600).get_image().collect()
input_data = np.concatenate([image.reshape((1, 1) + image.shape) for image in images], axis=0)

model = InferenceModel()
model.load_tf(options.model_path, backend="openvino", model_type=options.model_type)
predictions = model.predict(input_data)

# Print the detection result of the first image.
print(predictions[0])
```

Transparently support OpenVINO in model serving,
which deliver a significant boost for inference speed

Analytics Zoo

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras*-style APIs (with autograd & transfer learning support)
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines
- Built-in *feature engineering* operations for data preprocessing

Productionize deep learning applications for big data at scale

- POJO *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Storm, Flink, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models* and reference *use cases*

Built-in Deep Learning Models

- *Object detection*
  - E.g., SSD, Faster-RCNN, etc.
- *Image classification*
  - E.g., VGG, Inception, ResNet, MobileNet, etc.
- *Text classification*
  - Text classifier (using CNN, LSTM, etc.)
- *Recommendation*
  - E.g., Neural Collaborative Filtering, Wide and Deep Learning, etc.
- *Anomaly detection*
  - Unsupervised time series anomaly detection using LSTM
- *Sequence-to-sequence*

Object Detection API

1. **Load pretrained model in *Detection Model Zoo***

```
from zoo.common.nncontext import *
from zoo.models.image.objectdetection import *
spark = init_nncontext()
model = ObjectDetector.load_model(model_path)
```

2. **Off-the-shell inference using the loaded model**

```
image_set = ImageSet.read(img_path, spark)
output = model.predict_image_set(image_set)
```

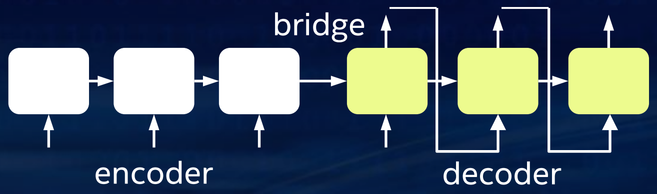
3. **Visualize the results using utility methods**

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

Off-the-shell Inference Using Analytics Zoo Object Detection API

https://github.com/intel-analytics/analytics-zoo/tree/master/pyzoo/zoo/examples/objectdetection

Sequence-to-Sequence API



Sequence to sequence model

```
encoder = RNNEncoder.initialize(rnn_type, nlayers, hidden_size, embedding)
decoder = RNNDecoder.initialize(rnn_type, nlayers, hidden_size, embedding)
seq2seq = Seq2seq(encoder, decoder)
```

Reference Use Cases

- ***Anomaly Detection***
  - Using LSTM network to detect anomalies in time series data
- ***Fraud Detection***
  - Using feed-forward neural network to detect frauds in credit card transaction data
- ***Recommendation***
  - Use Analytics Zoo Recommendation API (i.e., Neural Collaborative Filtering, Wide and Deep Learning) for recommendations on data with explicit feedback.
- ***Sentiment Analysis***
  - Sentiment analysis using neural network models (e.g. CNN, LSTM, GRU, Bi-LSTM)
- ***Variational Autoencoder (VAE)***
  - Use VAE to generate faces and digital numbers
- ***Web services***
  - Use Analytics Zoo model serving APIs for model inference in web servers

https://github.com/intel-analytics/analytics-zoo/tree/master/apps

Real-World Applications

Object detection and image feature extraction at JD.com

Produce defect detection using distributed TF on Spark in Midea

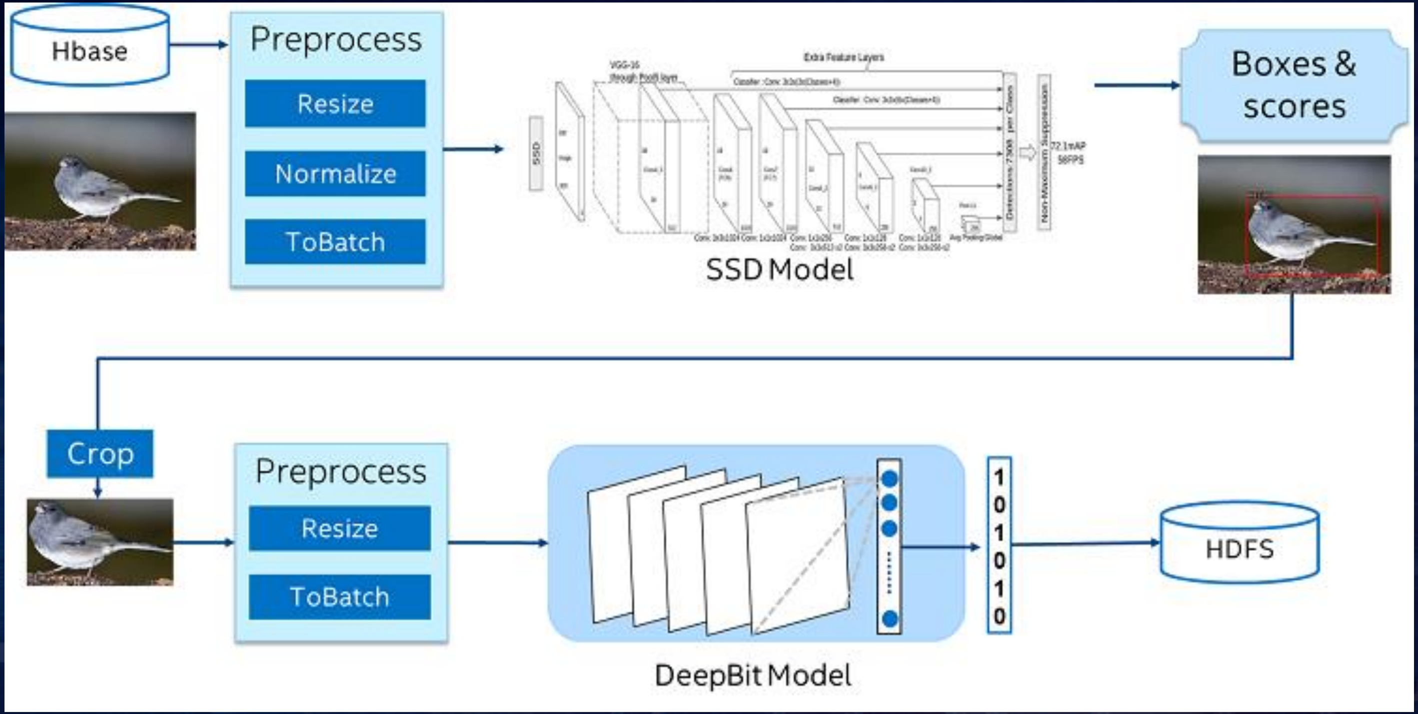
NLP based customer service chatbot for Microsoft Azure

Image similarity based house recommendation for MLSlisting

LSTM-Based Time Series Anomaly Detection for Baosight

Fraud Detection for Payment Transactions for UnionPay

Object Detection and Image Feature Extraction at JD.com



Applications

Large-scale image feature extraction

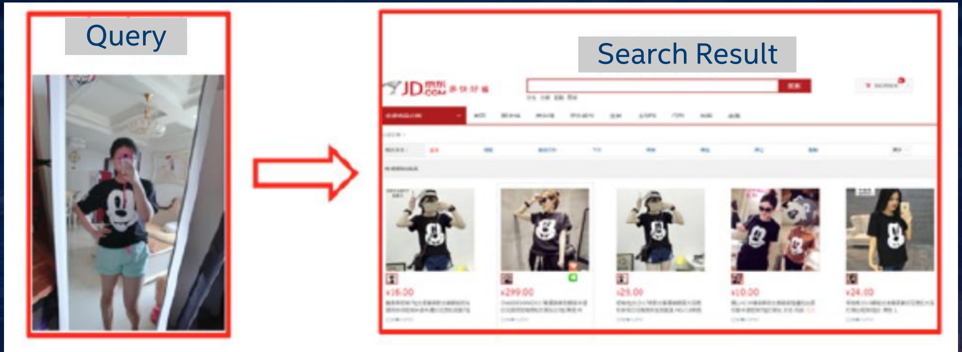
- Object detect (remove background, optional)
- Feature extraction

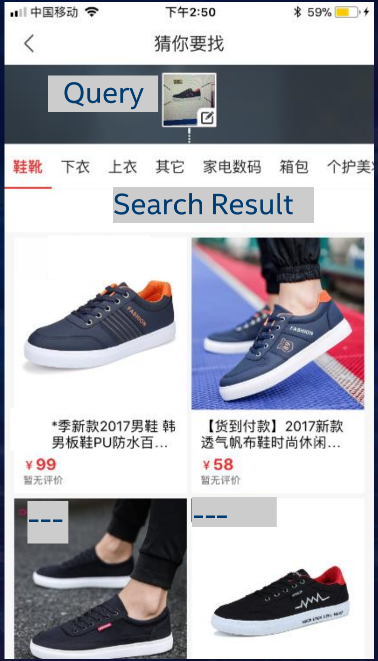
Application

- Similar image search
- Image Deduplication
  - Competitive price monitoring
  - IP (image copyright) protection system

Source: “Bringing deep learning into big data analytics using BigDL”, Xianyan Jia and Zhenhua Wang, Strata Data Conference Singapore 2017

Similar Image Search





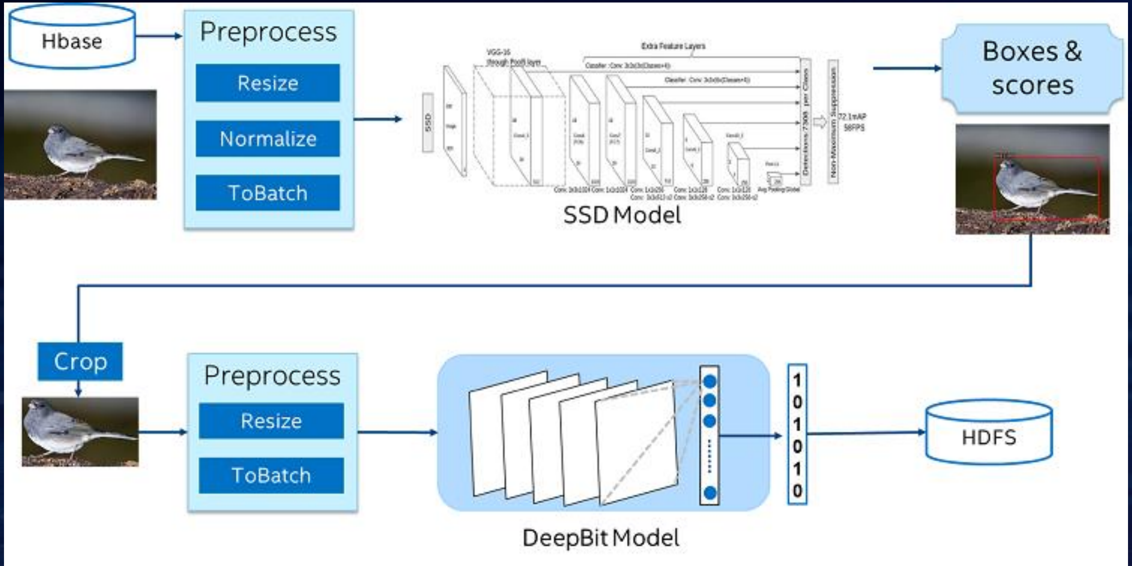
Source: "Bringing deep learning into big data analytics using BigDL", Xianyan Jia and Zhenhua Wang, Strata Data Conference Singapore 2017

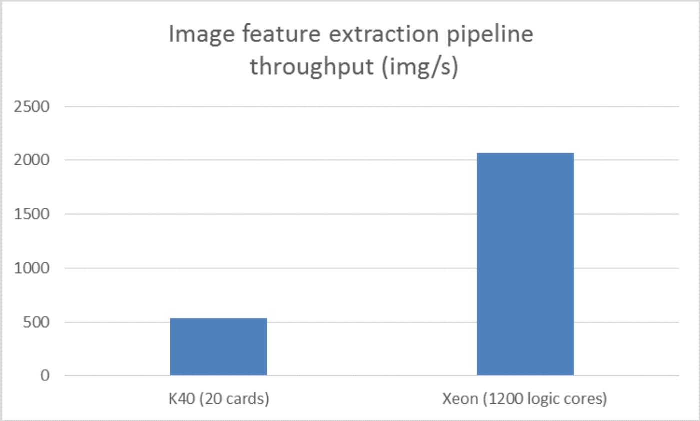
Challenges of Productionizing Large-Scale Deep Learning Solutions

Productionizing large-scale seep learning solutions is challenging

- Very complex and error-prone in managing large-scale distributed systems
  - E.g., resource management and allocation, data partitioning, task balance, fault tolerance, model deployment, etc.
- Low end-to-end performance in GPU solutions
  - E.g., reading images out from HBase takes about half of the total time
- Very inefficient to develop the end-to-end processing pipeline
  - E.g., image pre-processing on HBase can be very complex

Production Deployment with Analytics Zoo for Spark and BigDL





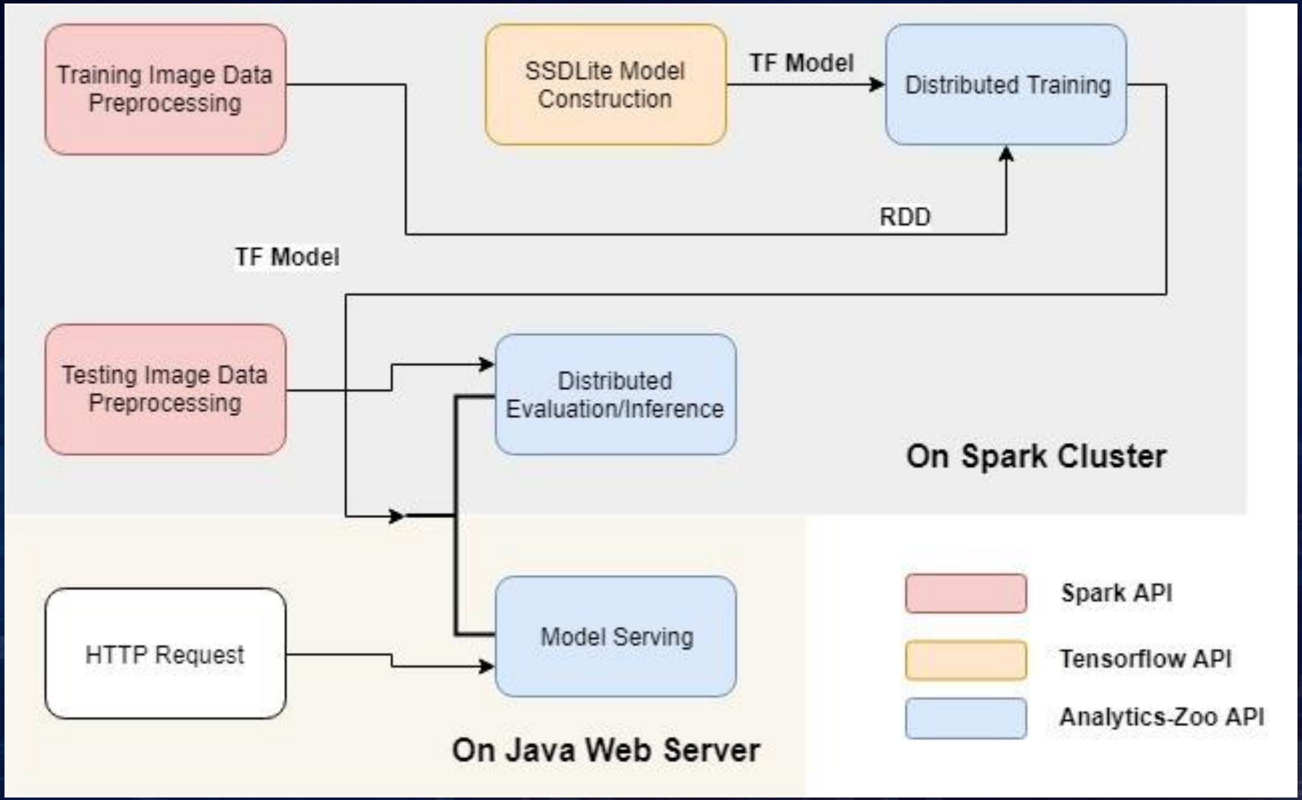
- Reuse existing Hadoop/Spark clusters for deep learning with no changes (image search, IP protection, etc.)
- Efficiently scale out on Spark with superior performance (***3.83x*** speed-up vs. GPU severs) as benchmarked by JD

  http://mp.weixin.qq.com/s/xUCkzbHK4K06-v5qUsaNQQ
  https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

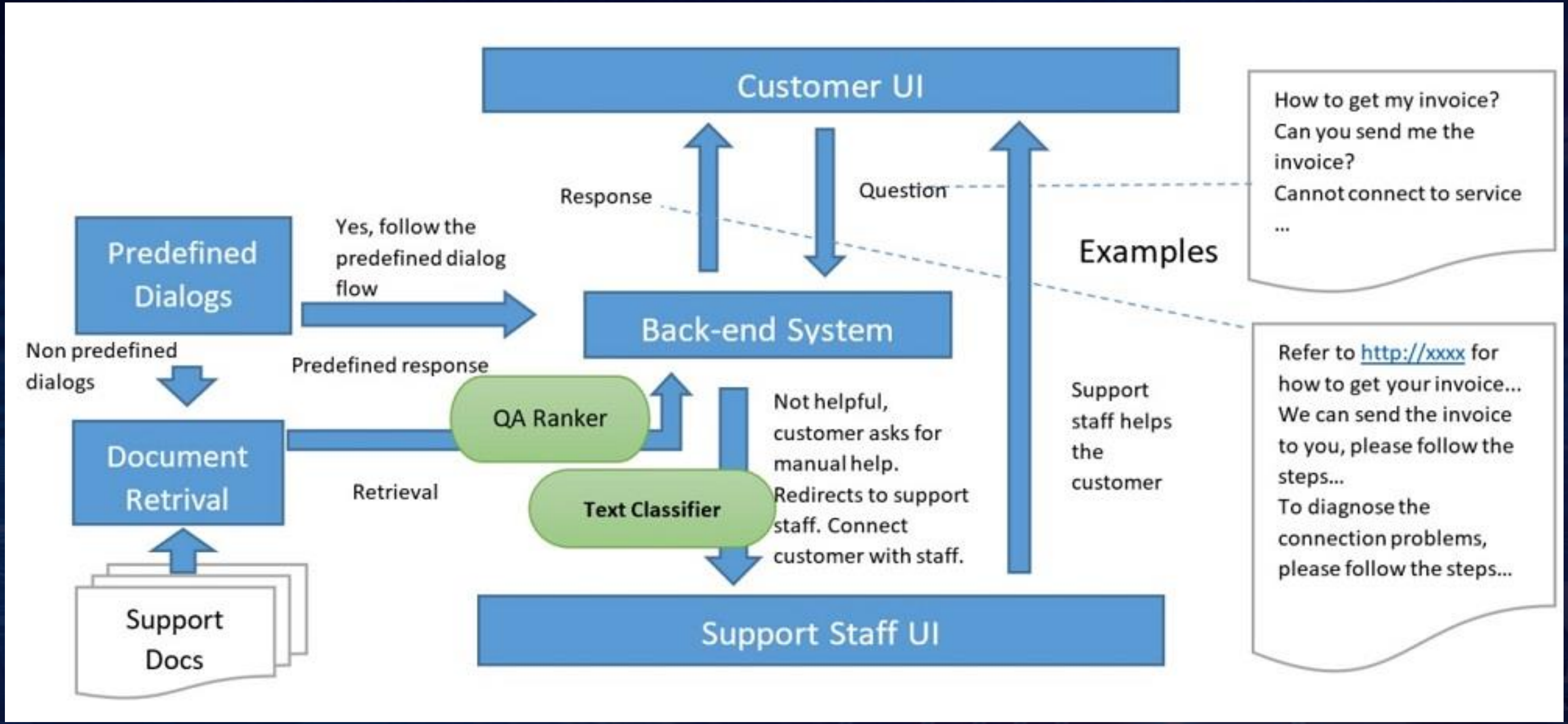
Produce Defect Detection using Distributed TF on Spark in Midea

https://software.intel.com/en-us/articles/industrial-inspection-platform-in-midea-and-kuka-using-distributed-tensorflow-on-analytics

Produce Defect Detection using Distributed TF on Spark in Midea



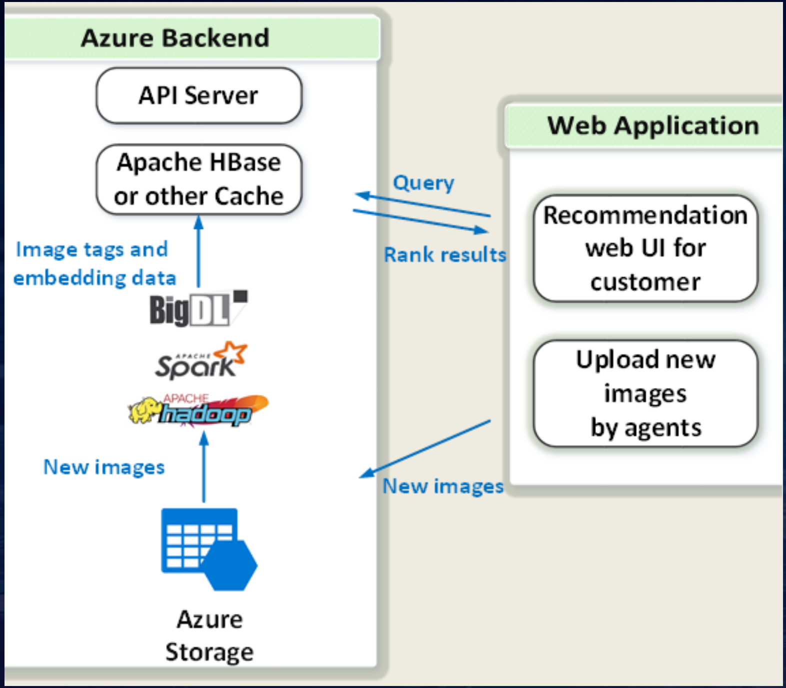
NLP Based Customer Service Chatbot for Microsoft Azure

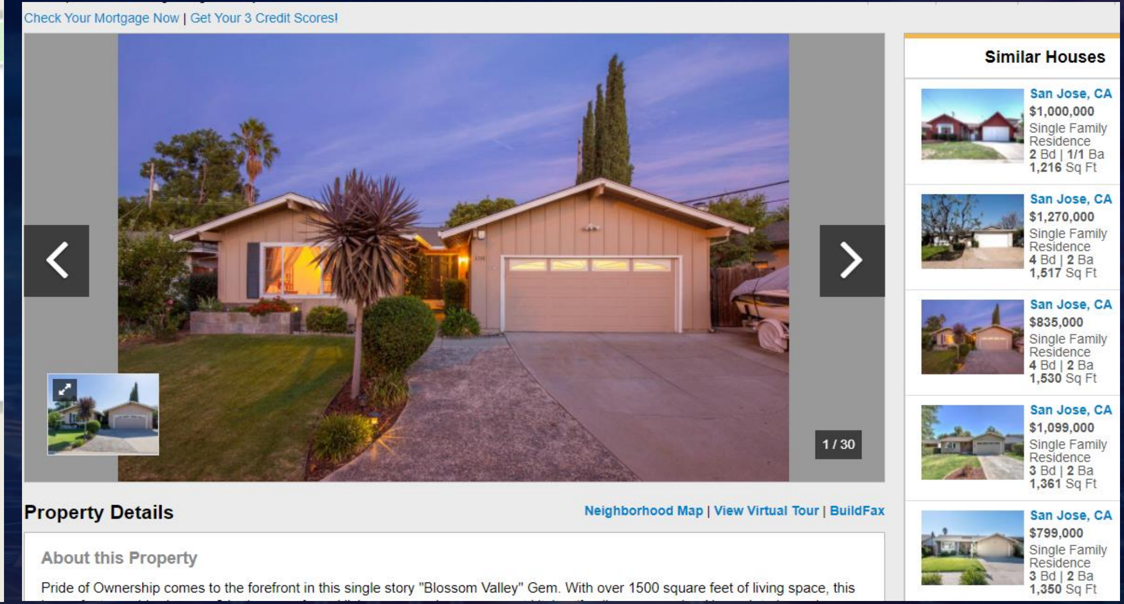


https://software.intel.com/en-us/articles/use-analytics-zoo-to-inject-ai-into-customer-service-platforms-on-microsoft-azure-part-1

Image Similarity Based House Recommendation for MLSlistings

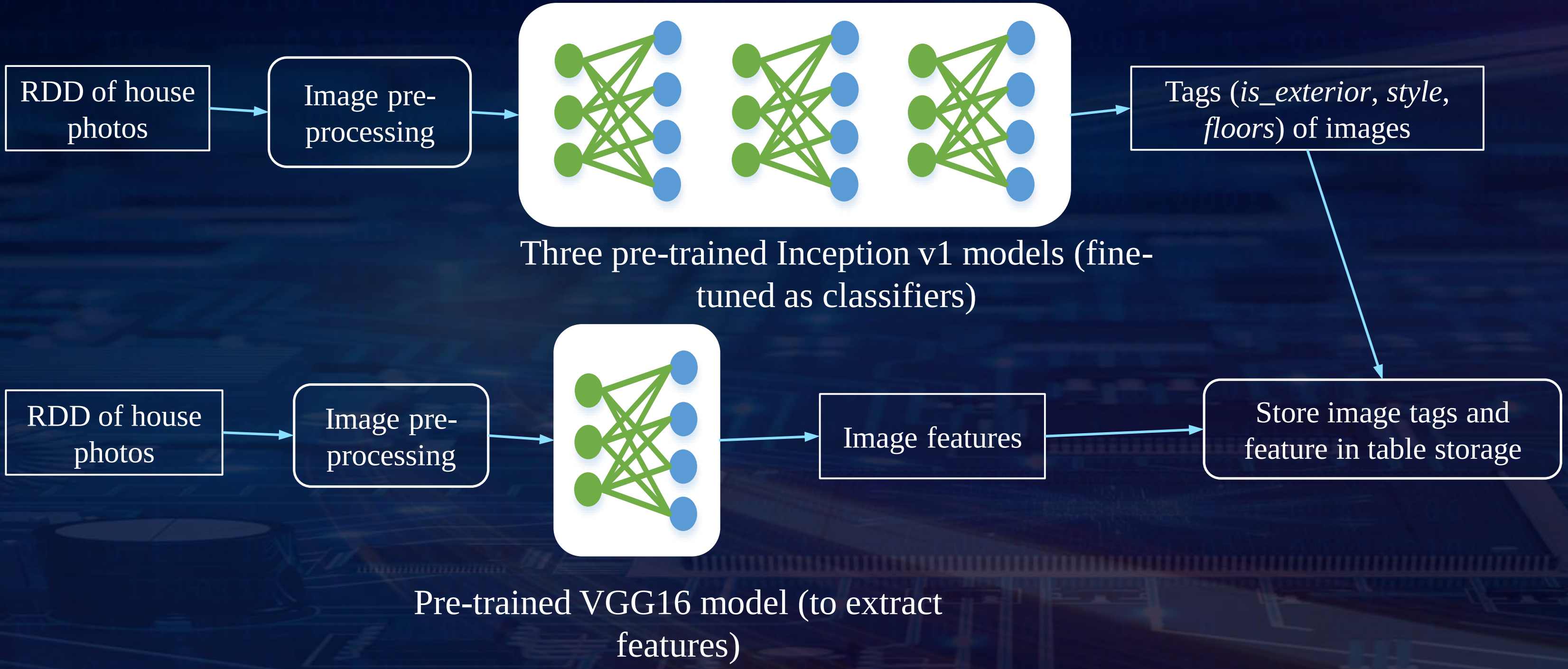
MLSlistings built image-similarity based house recommendations using BigDL on Microsoft Azure



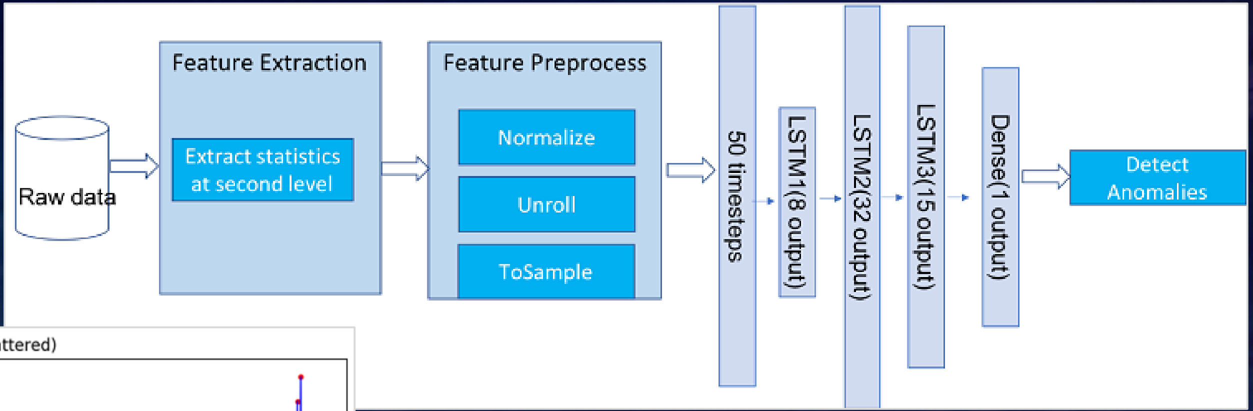


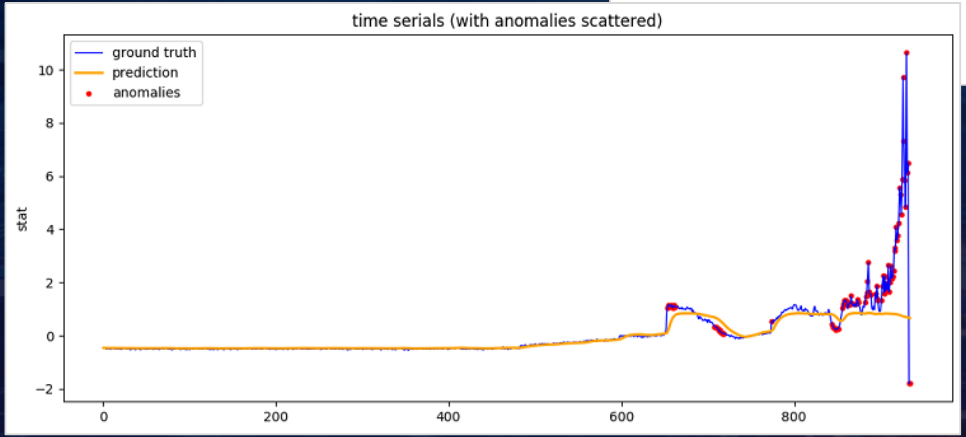
https://software.intel.com/en-us/articles/using-bigdl-to-build-image-similarity-based-house-recommendations

Image Similarity Based House Recommendation for MLSlistings



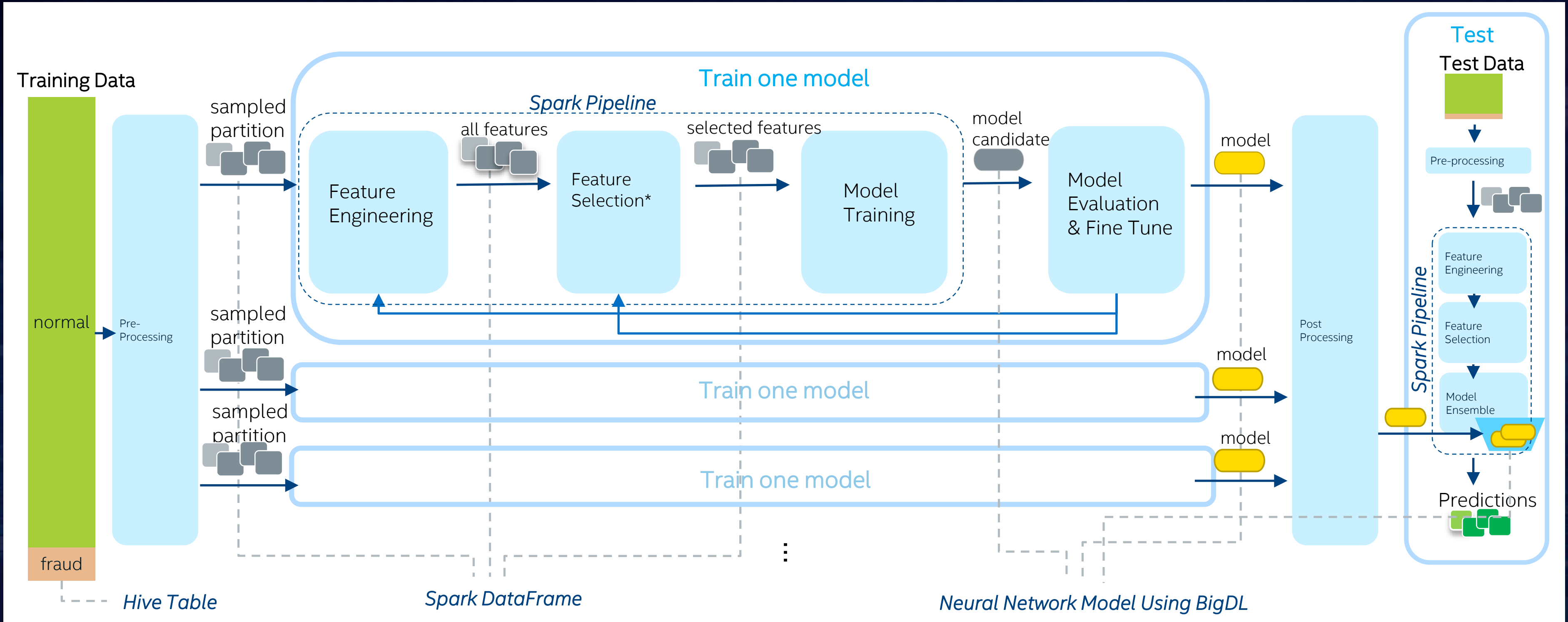
LSTM-Based Time Series Anomaly Detection for Baosight





https://software.intel.com/en-us/articles/lstm-based-time-series-anomaly-detection-using-analytics-zoo-for-apache-spark-and-bigdl

Fraud Detection for Payment Transactions for UnionPay



https://mp.weixin.qq.com/s?__biz=MzI3NDAwNDUwNg==&mid=2648307335&idx=1&sn=8eb9f63eaf2e40e24a90601b9cc03d1f

ANALYTICS ZOO

Unified Analytics + AI Platform

Distributed TensorFlow, Keras and BigDL on Apache Spark

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

Strata2019

LEGAL DISCLAIMERS

- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- No computer system can be absolutely secure.
- Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **http://www.intel.com/performance**.