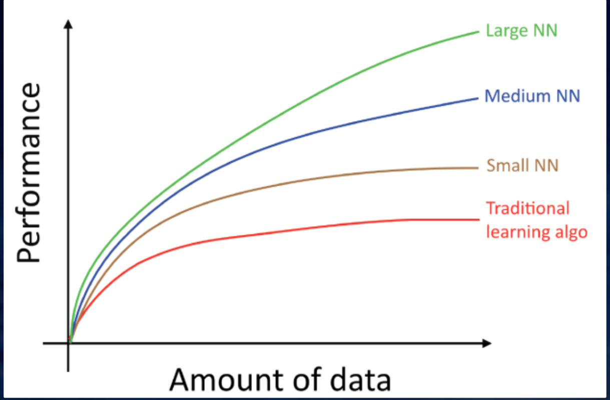
ANALYTICS ZOO: UNIFYING BIG DATA ANALYTICS AND AI FOR APACHE SPARK

Motivations

Data Scale Driving Deep Learning Process



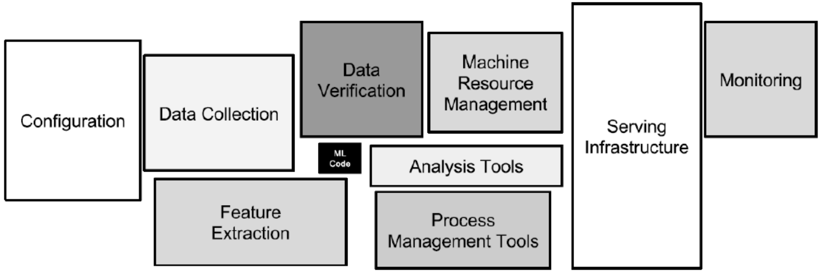


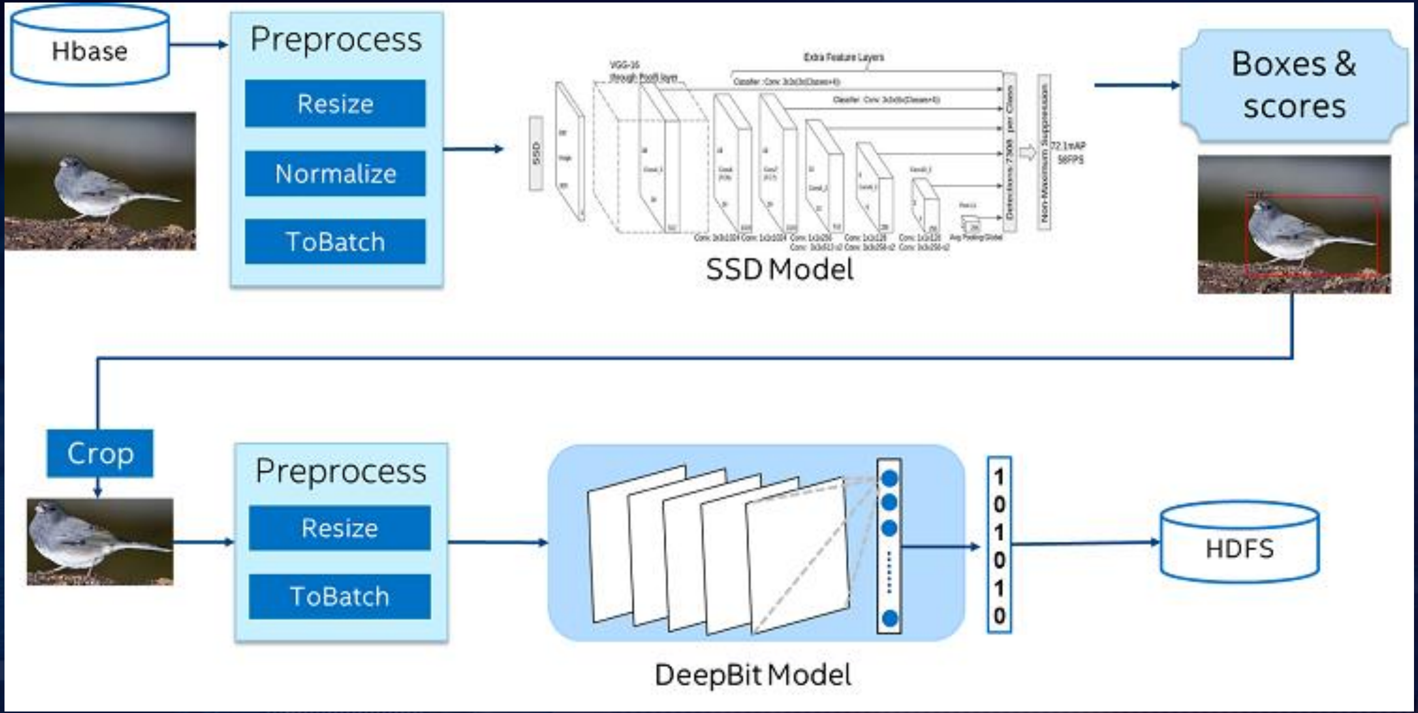
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

"Machine Learning Yearning", Andrew Ng, 2016

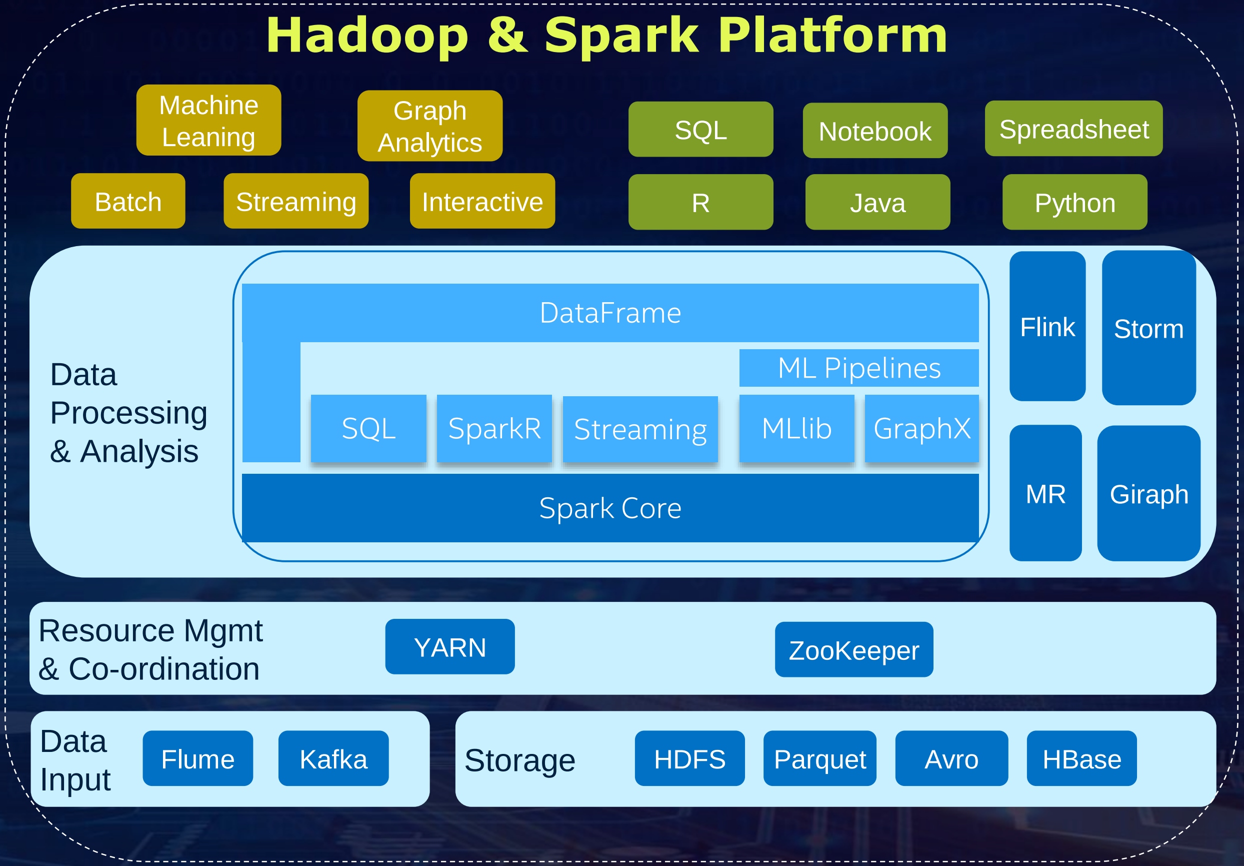
"Hidden Technical Debt in Machine Learning Systems", Google, NIPS 2015 paper

Chasm b/w Deep Learning and Big Data Communities

Large-Scale Image Recognition at JD.com



Unified Big Data Analytics Platform

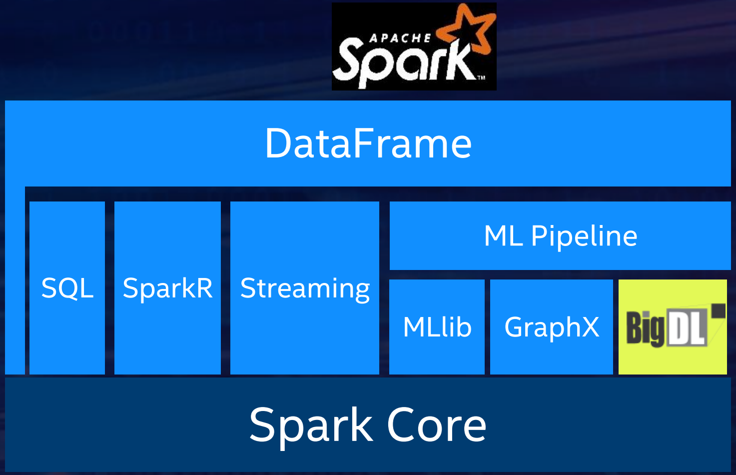


Overview

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

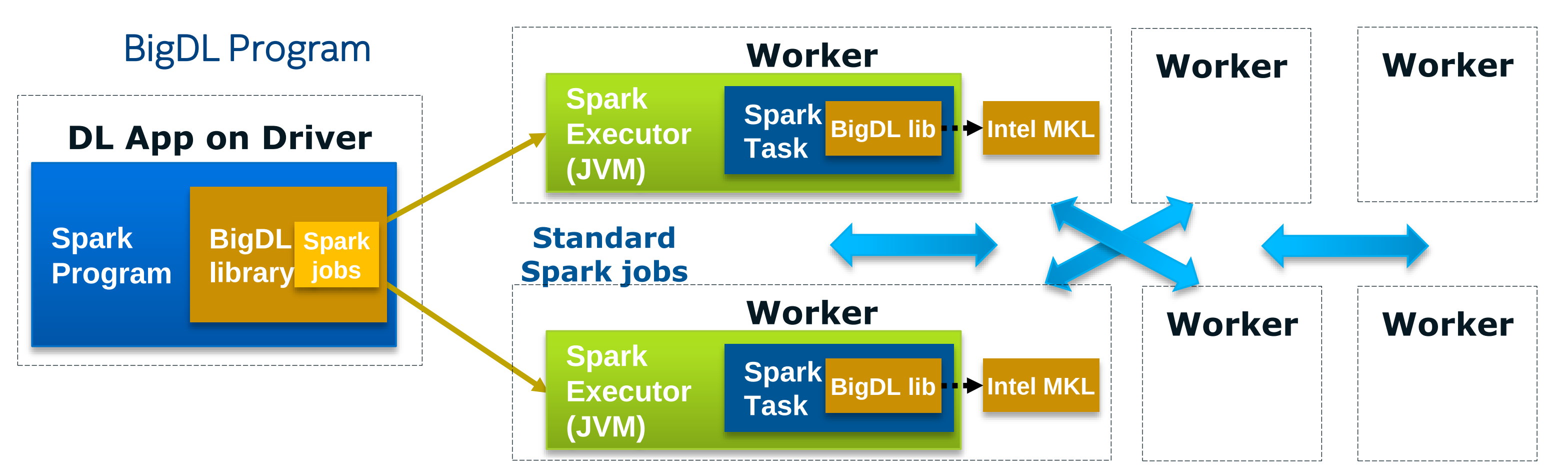
https://bigdl-project.github.io/

BigDL Run as Standard Spark Programs

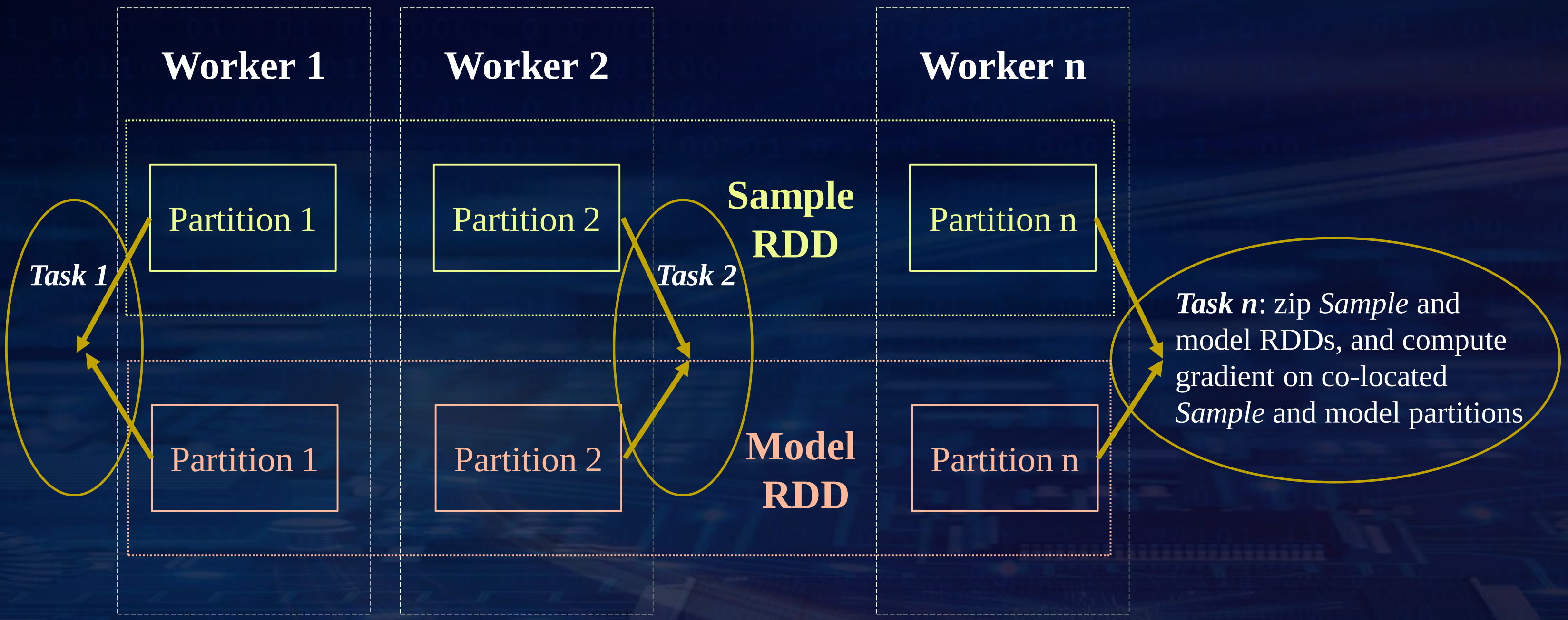
Standard Spark jobs
- No changes to the Spark or Hadoop clusters needed

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)



Data Parallel Training



"Model Forward-Backward" Job

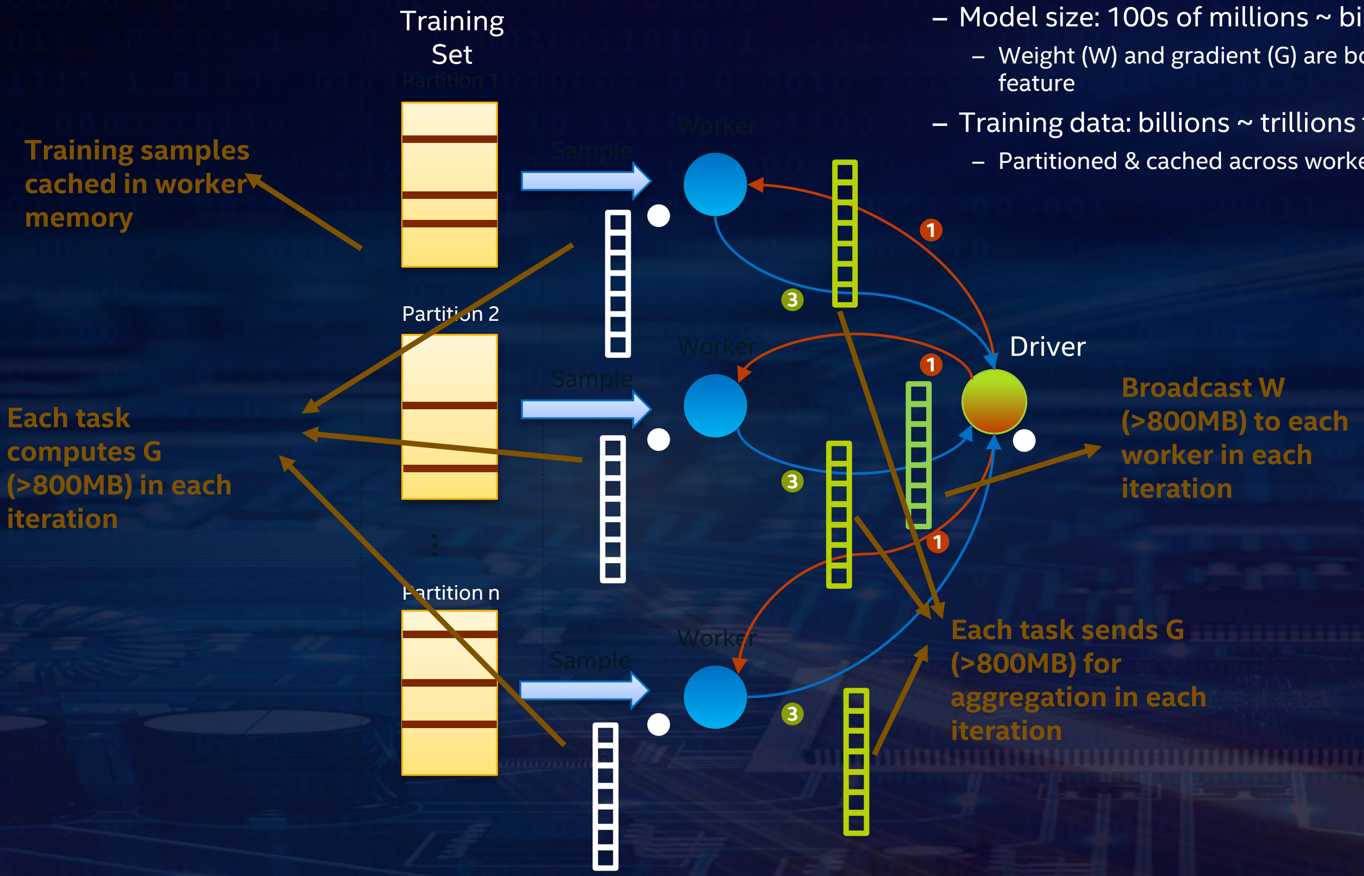
Distributed Training in BigDL

Data Parallel, Synchronous Mini-Batch SGD

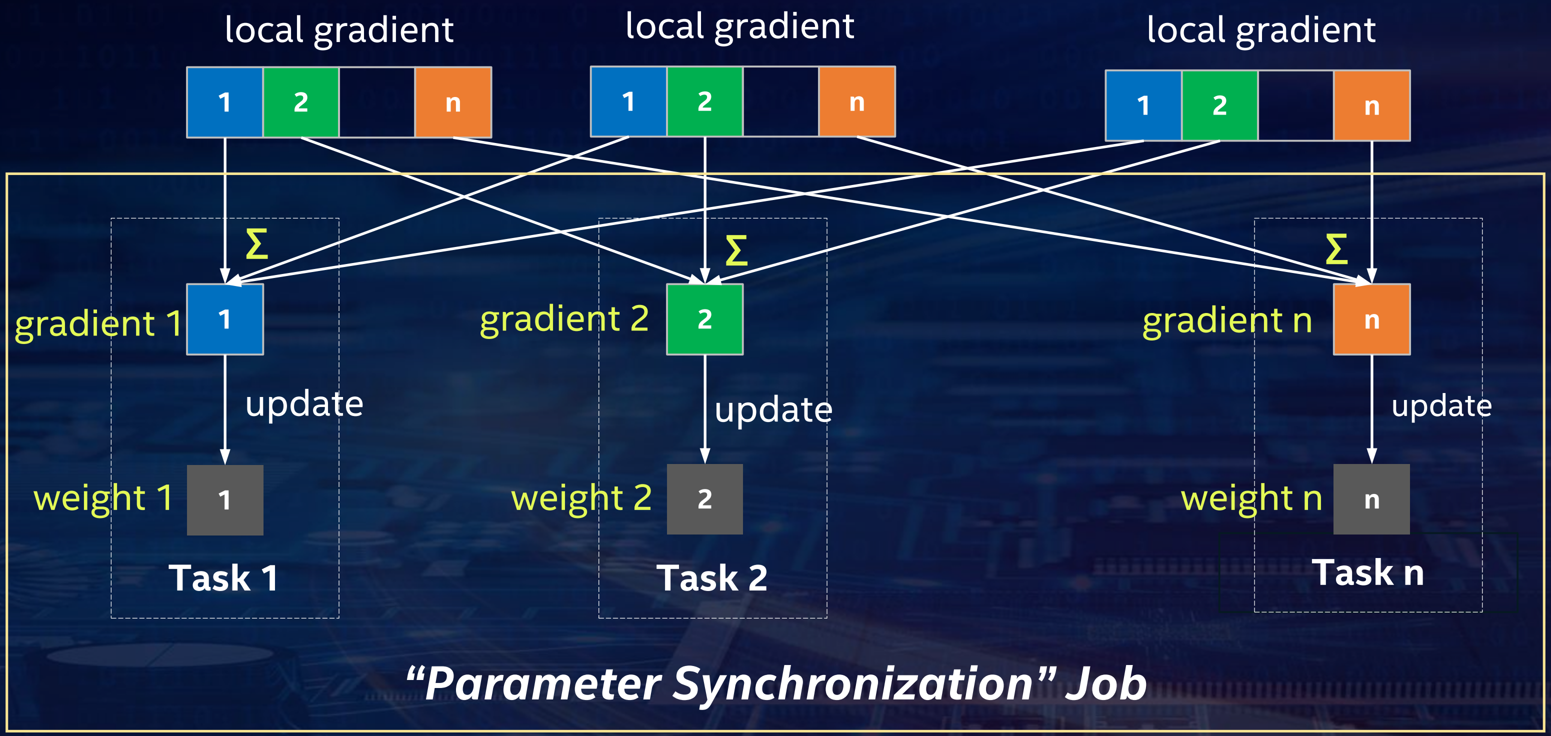
```
Prepare training data as an RDD of Samples
Construct an RDD of models (each being a replica of the original model)

for (i <- 1 to N) {
  //"model forward-backward" job
  for each task in the Spark job:
     read the latest weights
     get a random batch of data from local Sample partition
     compute errors (forward on local model replica)
     compute gradients (backward on local model replica)

  //"parameter synchronization" job
  aggregate (sum) all the gradients
  update the weights per specified optimization method
}
```

NETWORK AND MEMORY BOTTLENECKS

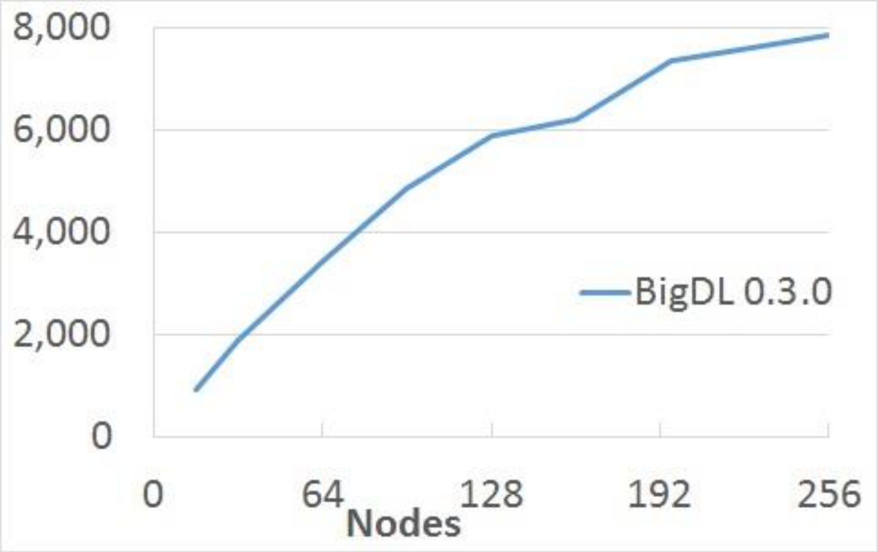
- Large-scale problem settings
  - Model size: 100s of millions ~ billions unique features
    - Weight (W) and gradient (G) are both double vector, one entry for each unique feature
  - Training data: billions ~ trillions training samples
    - Partitioned & cached across workers



Parameter 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

Build E2E Deep Learning Applications for Big Data at Scale

Analytics + AI Platform for Apache Spark and BigDL

Reference Use Cases	• Anomaly detection, sentiment analysis, fraud detection, image generation, etc.
Built-In Deep Learning Models	• Image classification, object detection, text classification, recommendations, etc.
Feature Engineering	Feature transformations for • Image, text, 3D imaging, time series, speech, etc.
High-Level Pipeline APIs	• Native deep learning support in Spark DataFrames and ML Pipelines • Autograd, Keras and transfer learning APIs for model definition • Model serving API for model serving/inference pipelines
Backbends	Spark, BigDL, TensorFlow, Python, etc.

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

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

Feature Engineering

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)

Autograd, Keras &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"]).to_keras())
# 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

Autograd, Keras &Transfer Learning APIs

2. Use *autograd and Keras-style* 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

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)
```

Built-in Deep Learning Models

- ***Object detection API***
  - High-level API and pretrained models (e.g., SSD, Faster-RCNN, etc.) for object detection
- ***Image classification API***
  - High-level API and pretrained models (e.g., VGG, Inception, ResNet, MobileNet, etc.) for image classification
- ***Text classification API***
  - High-level API and pre-defined models (using CNN, LSTM, etc.) for text classification
- ***Recommendation API***
  - High-level API and pre-defined models (e.g., Neural Collaborative Filtering, Wide and Deep Learning, etc.) for recommendation

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

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

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

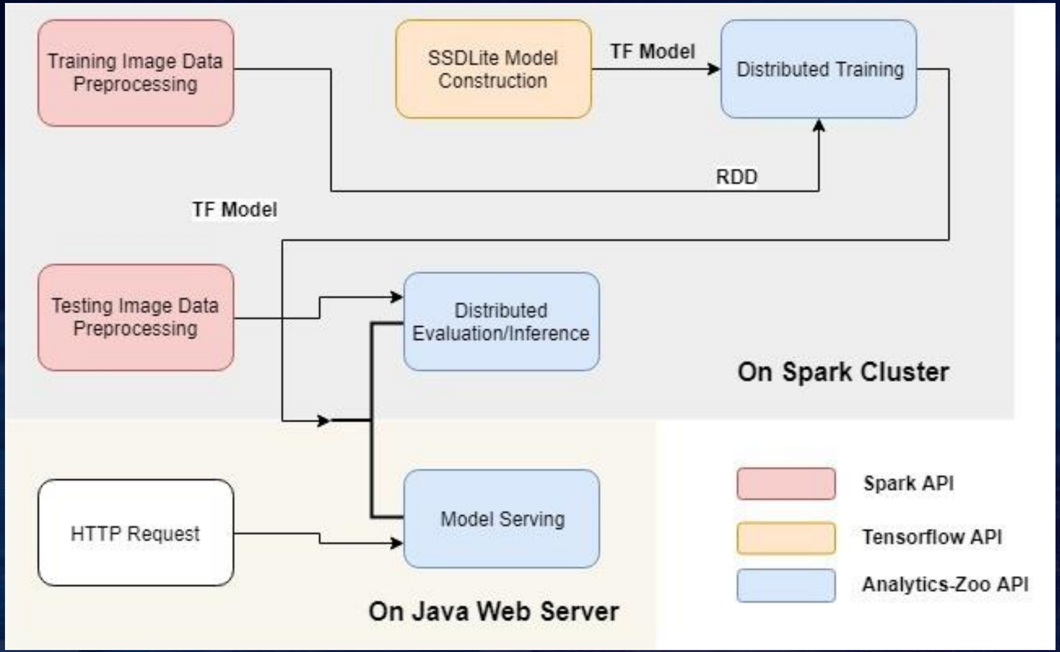
Models Interoperability Support
(e.g., between TensorFlow, Keras, Caffe, Torch, BigDL models)

- Load existing TensorFlow, Keras, Caffe, Torch Model
  - Useful for inference and model fine-tuning
  - Allows for transition from single-node for distributed application deployment
  - Allows for model sharing between data scientists and production engineers

Industrial Inspection Platform in Midea* and KUKA*

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

TensorFlow Object Detection: SSDLite+MobileNet V2



More details

```
import tensorflow as tf

slim = tf.contrib.slim

images, labels = dataset.tensors
squeezed_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=squeezed_labels))
```

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

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

```
model = Net.loadCaffe("/tmp/def/path", "/tmp/model/path") //load from local fs
model = Net.loadCaffe("hdfs://def/path", "hdfs://model/path") //load from hdfs
model = Net.loadCaffe("s3://def/path", "s3://model/path") //load from s3
```

https://analytics-zoo.github.io/master/#ProgrammingGuide/tensorflow/

Use Cases

Public Cloud Deployment

Optimized for Amazon* EC2* C5 instanced, and listed in AWS* Marketplace*

https://aws.amazon.com/blogs/machine-learning/leveraging-low-precision-and-quantization-for-deep-learning-using-the-amazon-ec2-c5-instance-and-bigdl/

Listed in Microsoft* Azure* Marketplace*

https://azure.microsoft.com/en-us/blog/bigdl-spark-deep-learning-library-vm-now-available-on-microsoft-azure-marketplace/

Available on Google* Cloud Dataproc*

https://cloud.google.com/blog/big-data/2018/04/using-bigdl-for-deep-learning-with-apache-spark-and-google-cloud-dataproc

Deployed on AliCloud* E-MapReduce*

https://yq.aliyun.com/articles/73347

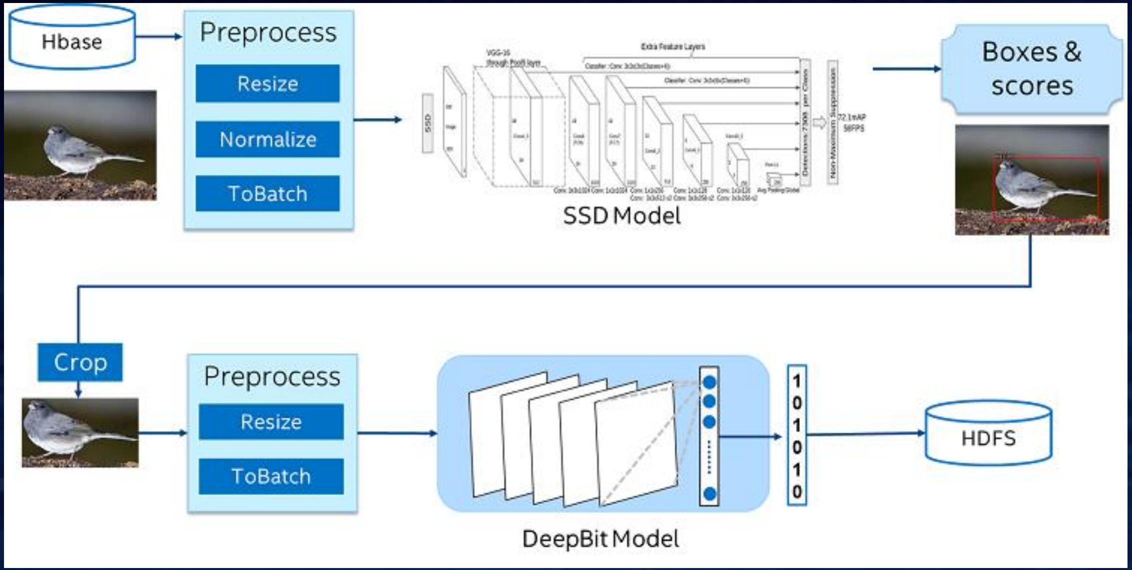
Deployed on IBM* Data Science Expetience*

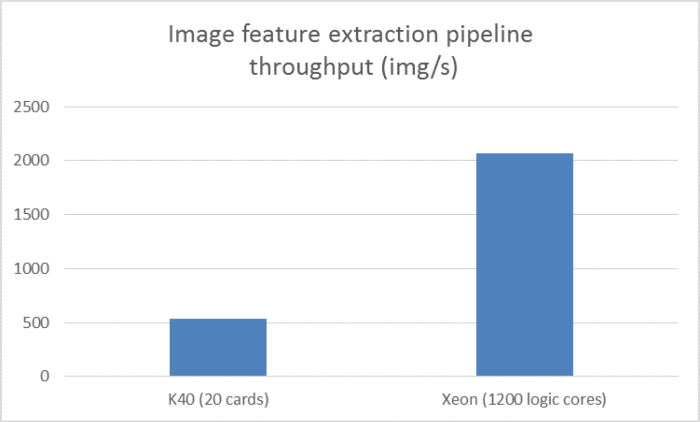
https://medium.com/ibm-data-science-experience/using-bigdl-in-data-science-experience-for-deep-learning-on-spark-f1cf30ad6ca0

Available on Telefonica* Open Cloud*

https://support.telefonicaopencloud.com/en-us/ecs/doc/download/20180329/20180329111611_166372a698.pdf

Object Detection and Image Feature Extraction in JD





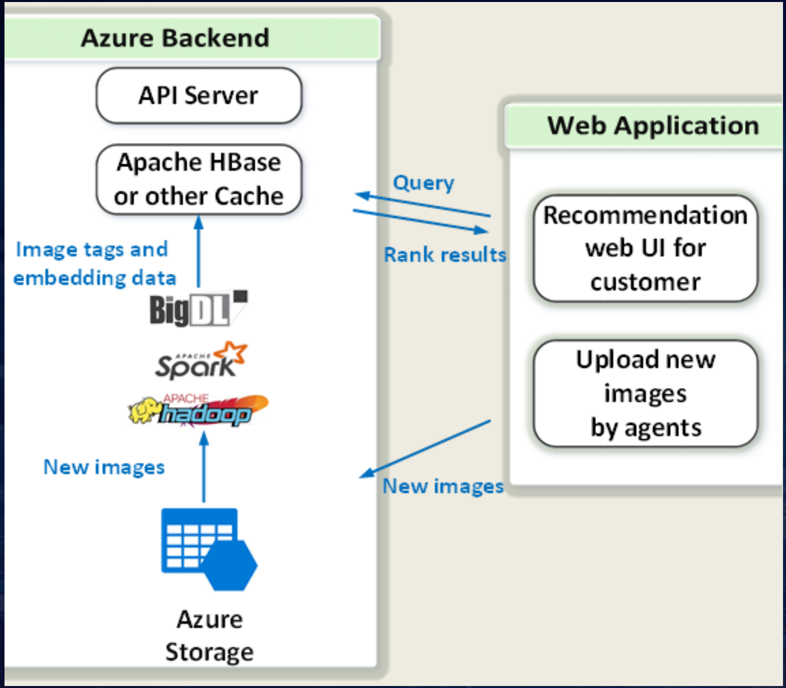
- Reuse existing Hadoop/Spark clusters for deep learning with no changes (image search, IP protection, etc.)
- Efficiently scale out on Spark with superior performance (***3.83x*** speed-up vs. GPU severs) as benchmarked by JD

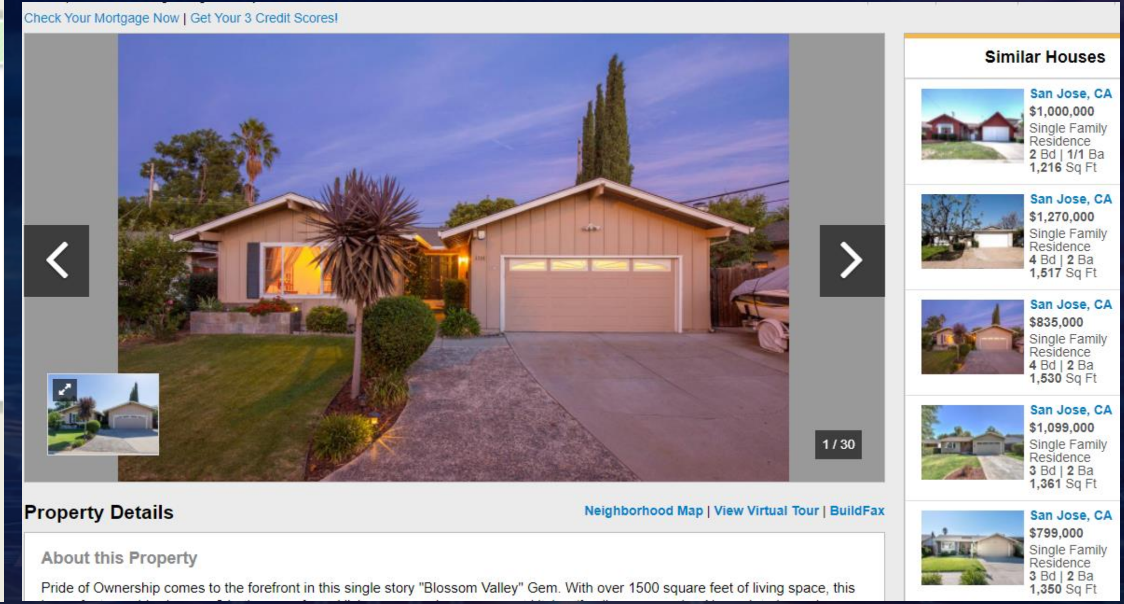
http://mp.weixin.qq.com/s/xUCkzbHK4K06-v5qUsaNQQ
https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Image Similarity Based House Recommendation for MLSlistings

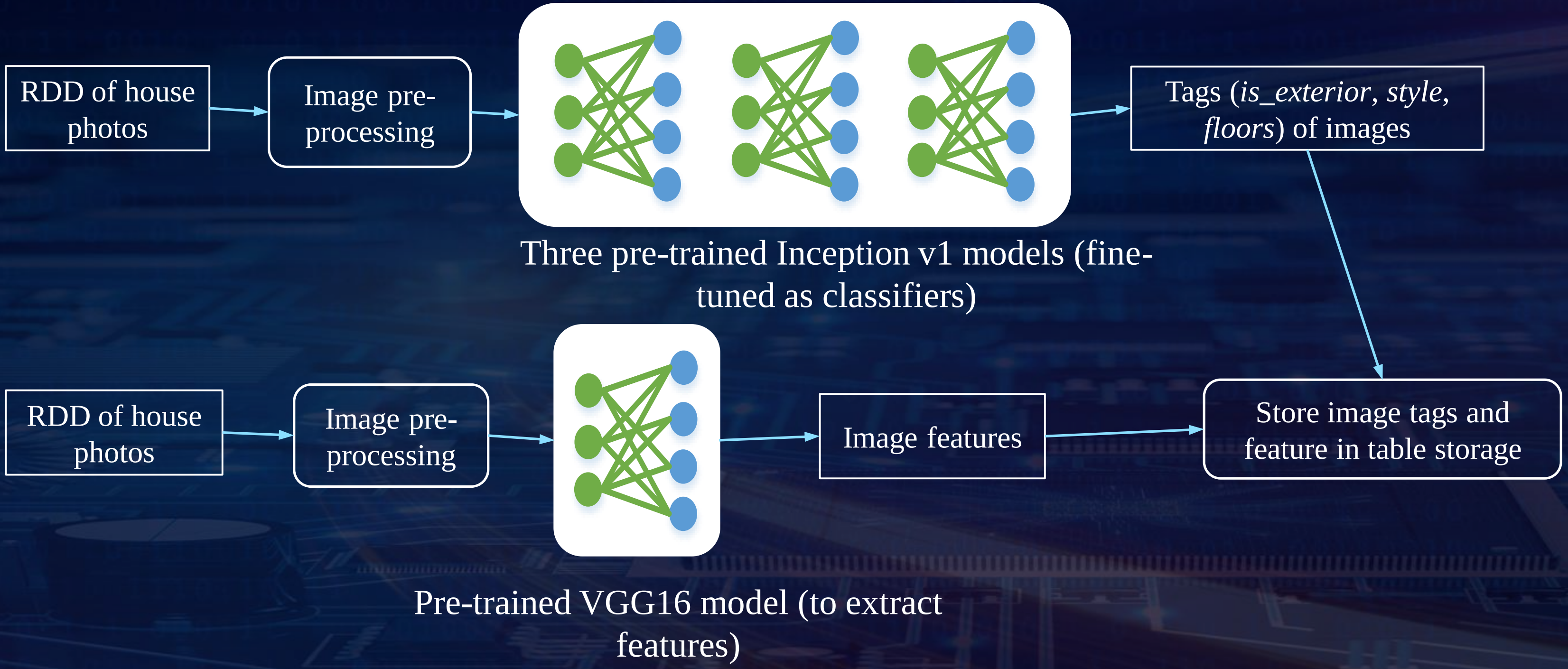
MLSlistings built image-similarity based house recommendations on Microsoft Azure



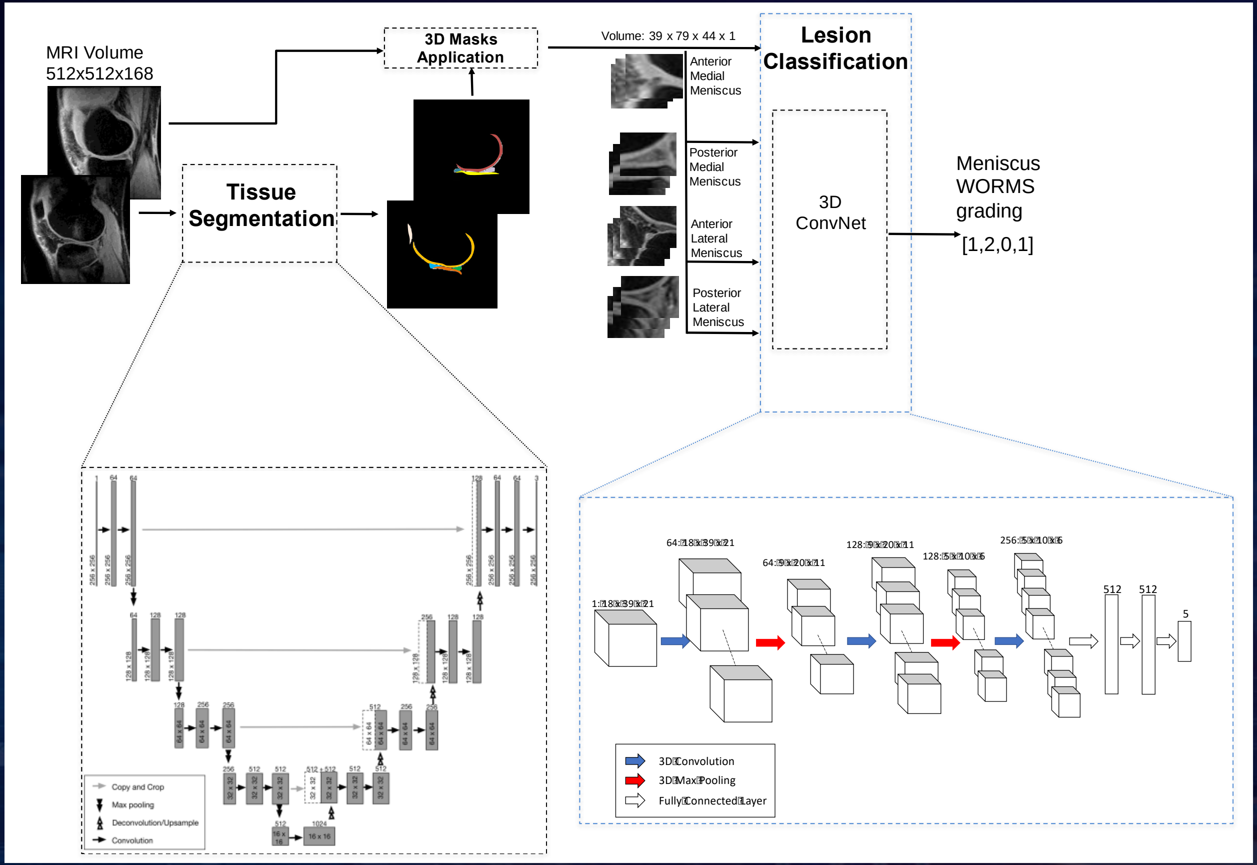


https://software.intel.com/en-us/articles/using-bigdl-to-build-image-similarity-based-house-recommendations

Image Similarity Based House Recommendation for MLSlistings

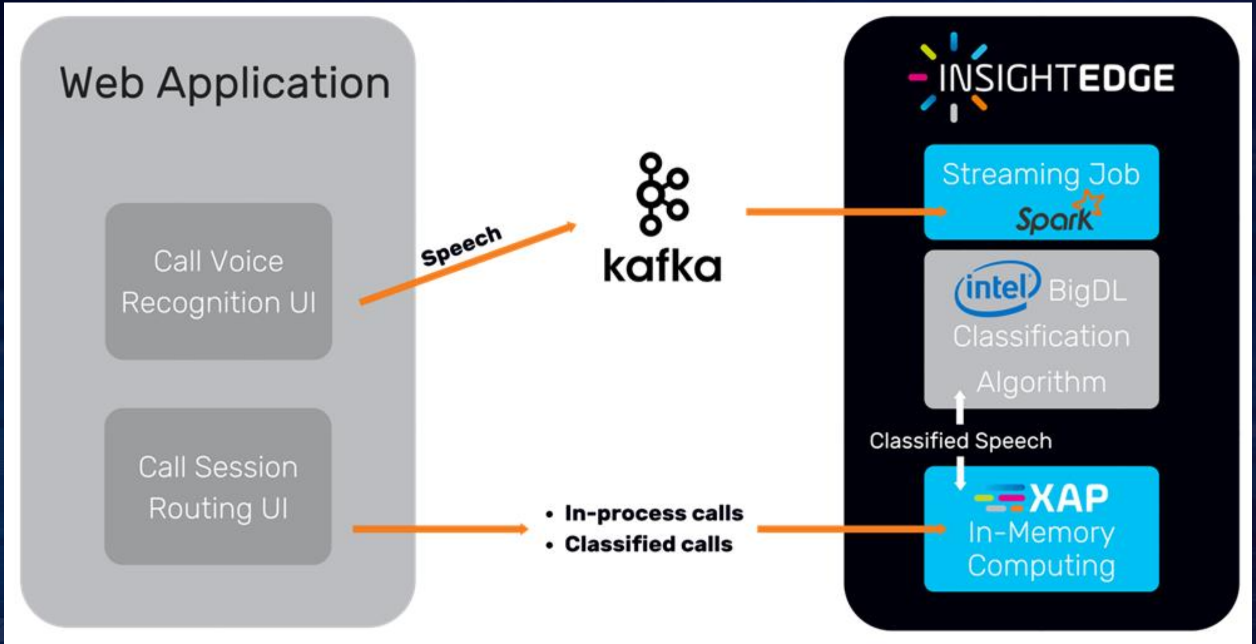


3D Medical Image Analysis in UCSF



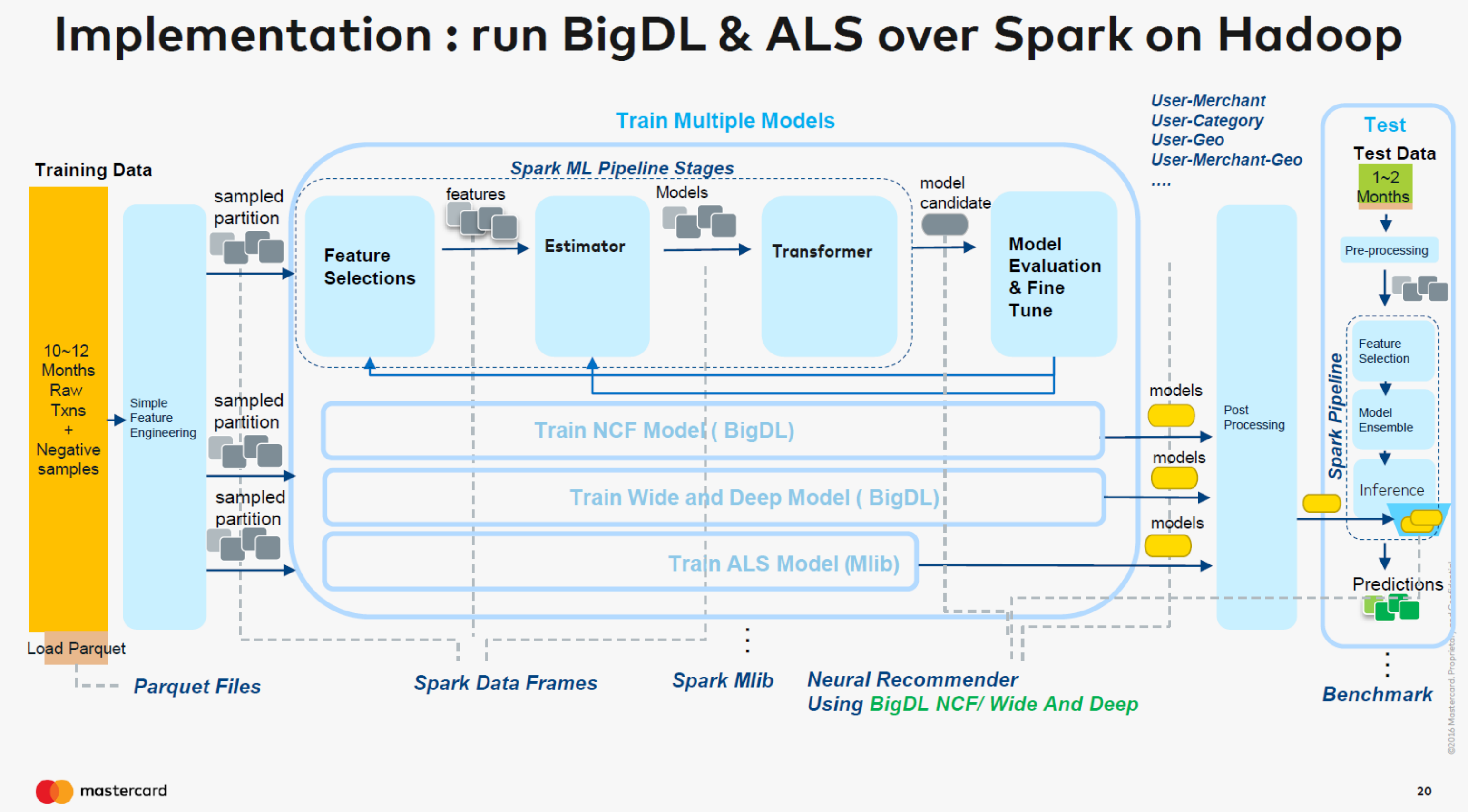
https://conferences.oreilly.com/strata/strata-ca/public/schedule/detail/64023

NLP Based Call Center Routing in GigaSpaces



https://blog.gigaspaces.com/gigaspaces-to-demo-with-intel-at-strata-data-conference-and-microsoft-ignite/

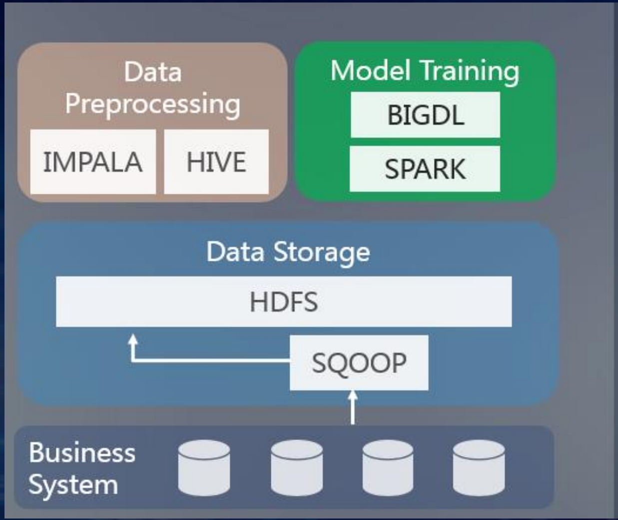
User-Merchant Propensity Modeling in MasterCard



https://conferences.oreilly.com/strata/strata-ca/public/schedule/detail/63897

Neural Recommendation Engine in China Life

Realize re-discovery of life insurance business, accurately and effectively recommend products.



BigData Platform：

- CDH 5.10
- Through the Sqoop multiple library data access HDFS
- Data cleaning and partial preprocessing using Hive/Impala
- Use Spark On Hive to read data in structured form, and call BigDL for model training

- https://strata.oreilly.com.cn/strata-cn/public/schedule/detail/59722?locale=en

Partner With Us

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

Documents: https://analytics-zoo.github.io/

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

intel®

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.