Simplify AI Solutions on Big Data with BigDL / Analytics Zoo - Part 2

Sajan Govindan, Jason Dai, Radhika Rangarajan
Intel

December 2018

Legal notices & disclaimers

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

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.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon, Movidius, Saffron and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

Agenda

- **Building & Deploying**
  - Applications, Partnerships, Cloud, Intel Select Solutions
- **Distributed Training in BigDL**
  - Data parallel training, parameter synchronization, scaling & convergence, task scheduling, etc.
- **Real-world Applications**
  - Object detection and image feature extraction at *JD.com*
  - Image similarity based house recommendation for *MLSlistings*
  - Transfer learning based image classifications for *World Bank*
- **Conclusion**

Building & Deploying

Applications
End to end Deep Learning examples

Consumer

[Sentiment Analysis]
[Image Similarity And Search]
[Image Transfer Learning]
[Image Generation]

Health

[3D Image Support]

Finance

[Fraud Detection]
[Anomaly Detection]

Retail

[Recommendation]
[NCF]
[Wide n Deep]

Manufacturing

[Object Detection]

Infrastructure

[Tensorflow support]
[Low latency serving]

Check how easy it is to use BigDL and Analytics-Zoo

Building and Deploying with BigDL/Analytics Zoo

Technology

bluedata
cloudera
CRAY
databricks
DELL EMC
KINGSOFT
Lightbend
Qubole

Cloud Service Providers

Alibaba Cloud
aliyun.com
aws
Cloud Dataproc
Azure
IBM Cloud
Telefónica

End Users

cdhi
中国电信 CHINA TELECOM
GIGASPACES
JD.COM 京东
Midea
UnionPay 银联

http://software.intel.com/bigdl/build

*Other names and brands may be claimed as the property of others.

Not a full list

Intel® Select Solution for BigDL on Apache Spark

Turnkey Analytics/AI solution integrated with Intel Xeon Scalable processors and Intel SSDs

intel select solution

Simplified evaluation — Tightly-specified HW and SW components, eliminating guesswork

Fast and easy to deploy — Pre-defined settings and system-wide tuning, enabling smooth deployment

Workload optimized — Designed and benchmarked to perform optimally for specific workloads

All Intel® Select Solution configurations and benchmark results are verified by Intel

https://www.intel.com/content/www/us/en/architecture-and-technology/intel-select-solutions-overview.html

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 Experience*

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

Distributed Training In BigDL

Data parallel training

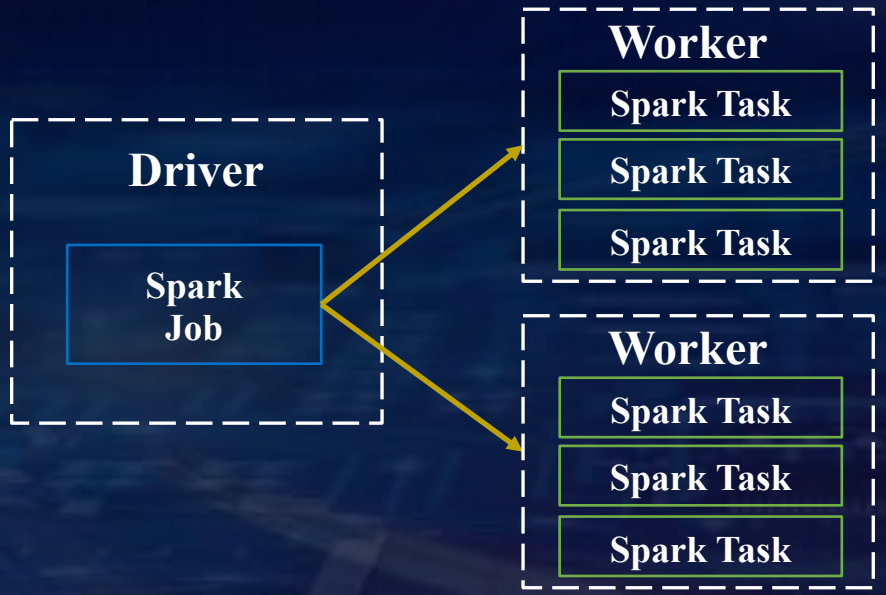
Parameter synchronization

Scaling and Convergence

Task scheduling

"BigDL: A Distributed Deep Learning Framework for Big Data", https://arxiv.org/abs/1804.05839

Apache Spark

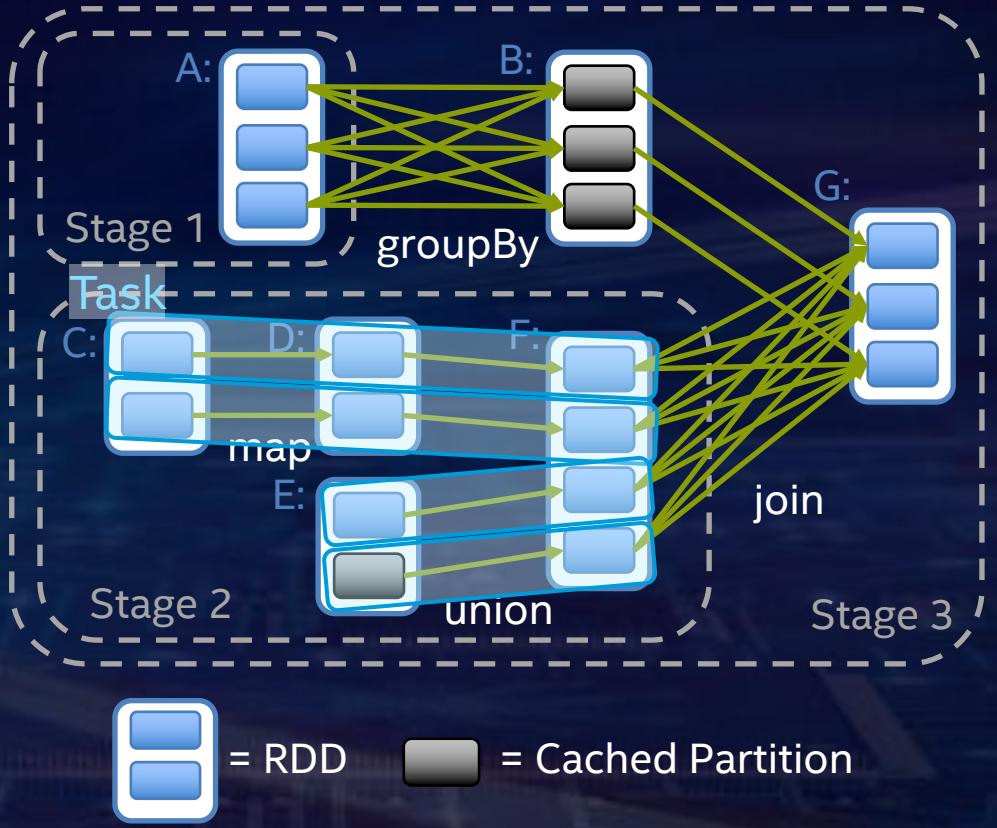


Single master (driver), multiple workers

Apache Spark

Spark compute model

- **Data parallel**
- **Functional, coarse-grained operators**
  - **Immutable RDDs**
  - **Applying the same operation (e.g., `map`, `filter`, etc.) to all data items**



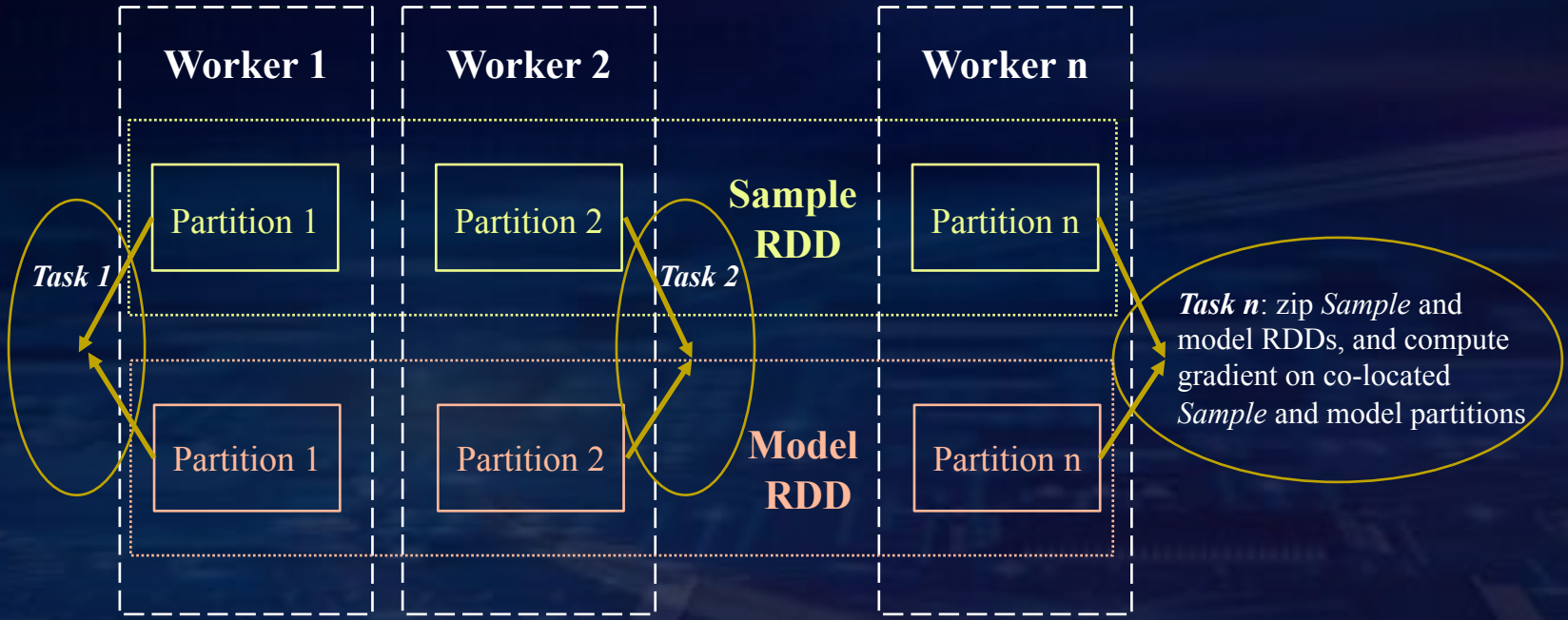
Source: "Parallel programming with Spark", Matei Zaharia, AMPCamp 3

Distributed Training in BigDL

Data Parallel, Synchronous Mini-Batch SGD

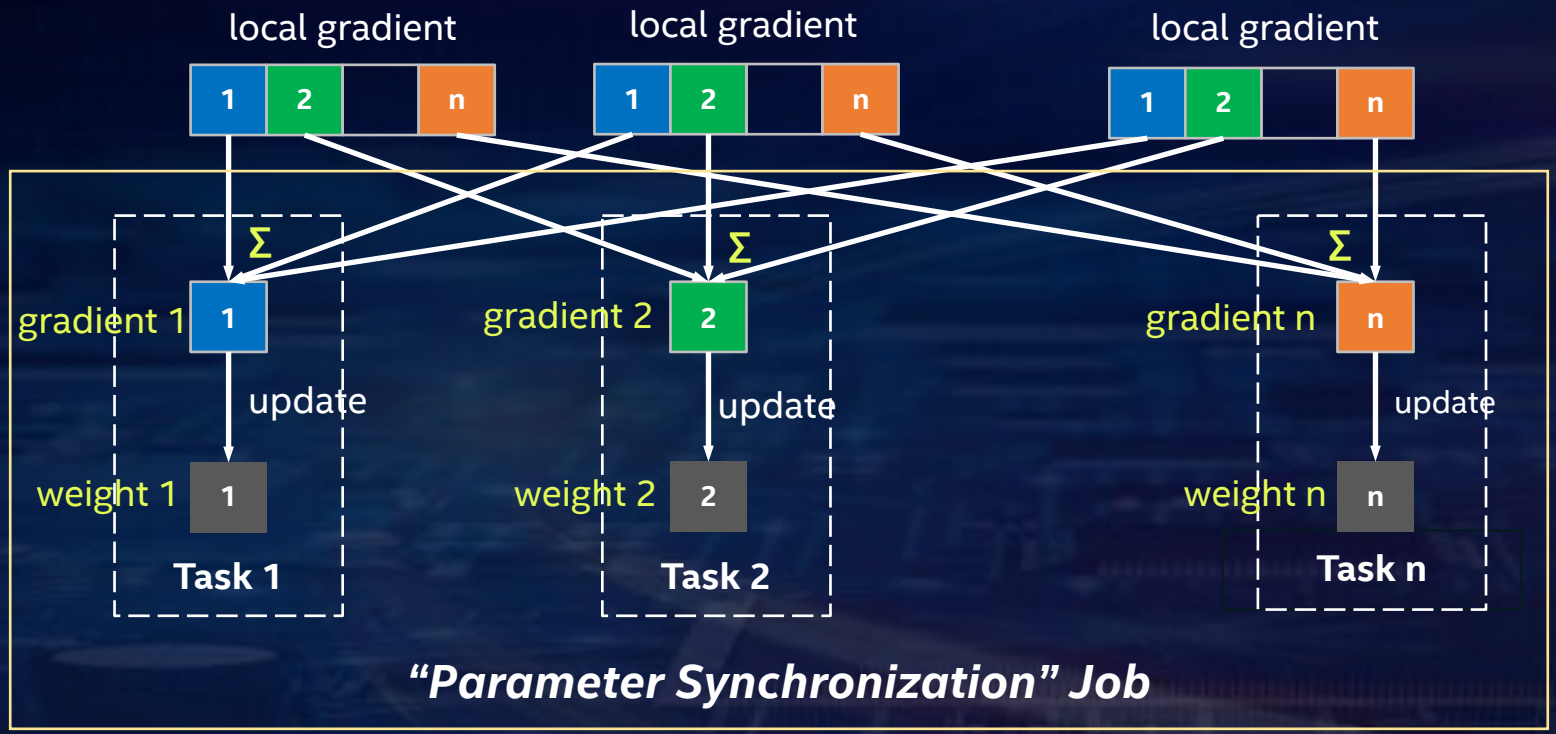
```
Prepare training data as an RDD of Samples
Construct an RDD of models (each being a replica of the original model)

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
}
```

Data Parallel Training



"Model Forward-Backward" Job

Parameter Synchronization



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

Parameter Synchronization

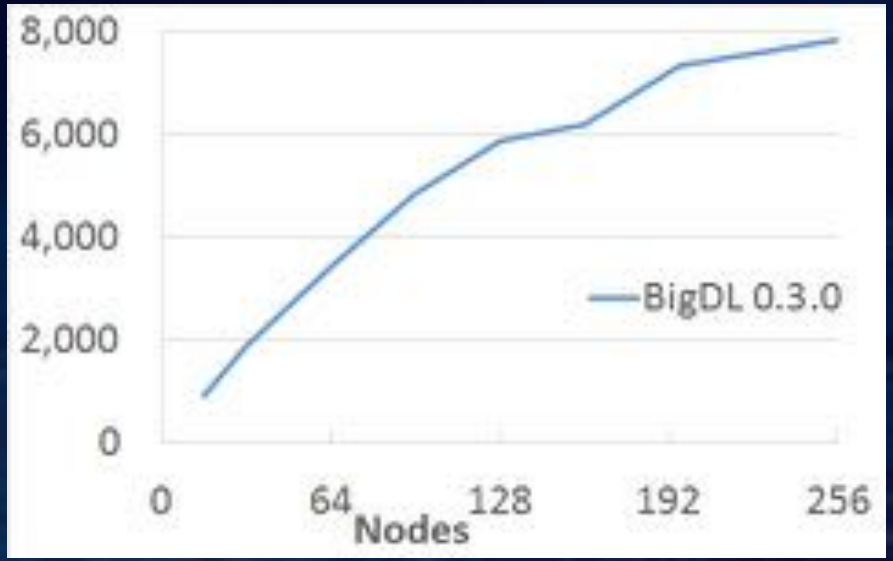
```
For each task n in the "parameter synchronization" job {
   shuffle the n^th partition of all gradients to this task
   aggregate (sum) the gradients
   updates the n^th partition of the weights
   broadcast the n^th partition of the updated weights
}
```

"Parameter Synchronization" Job
(managing n^{th} partition of the parameters - similar to a parameter server)

"Parameter Server" style architecture (directly on top of primitives in Spark)

- **Gradient aggregation: *shuffle***
- **Weight sync: *task-side broadcast***
- ***In-memory* persistence**

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/)

Difference vs. Classical PS Architecture

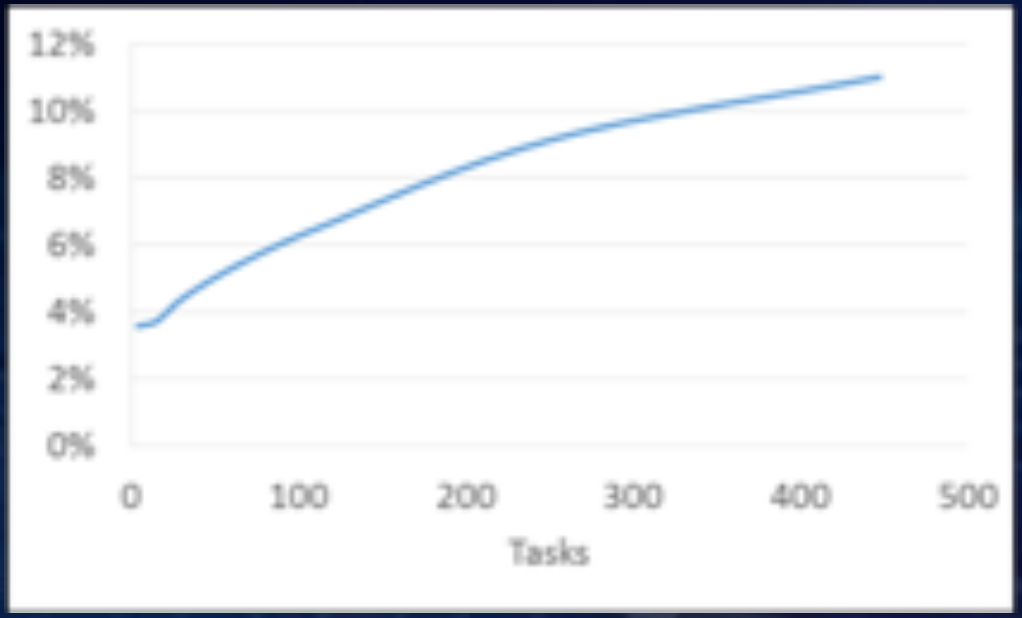
Classical PS architecture

- Multiple long-running, potentially stateful tasks
- Interact with each other (in a blocking fashion for synchronization)
- Require fine-grained data access and in-place data mutation
- Not directly supported by existing big data systems

BigDL implementations

- Run a series of short-lived Spark jobs (e.g., two jobs per mini-batch)
- Each task in the job is stateless and non-blocking
- Automatically adapt to the dynamic resource changes (e.g., *preemption*, *failures*, *resource sharing*, etc.)
- Built on top of existing primitives in Spark (e.g., *shuffle*, *broadcast*, and *in-memory data persistence*)

Task Scheduling Overheads



Spark overheads (task scheduling & task dispatch) as a fraction of average compute time for Inception v1 training

BigDL implementations

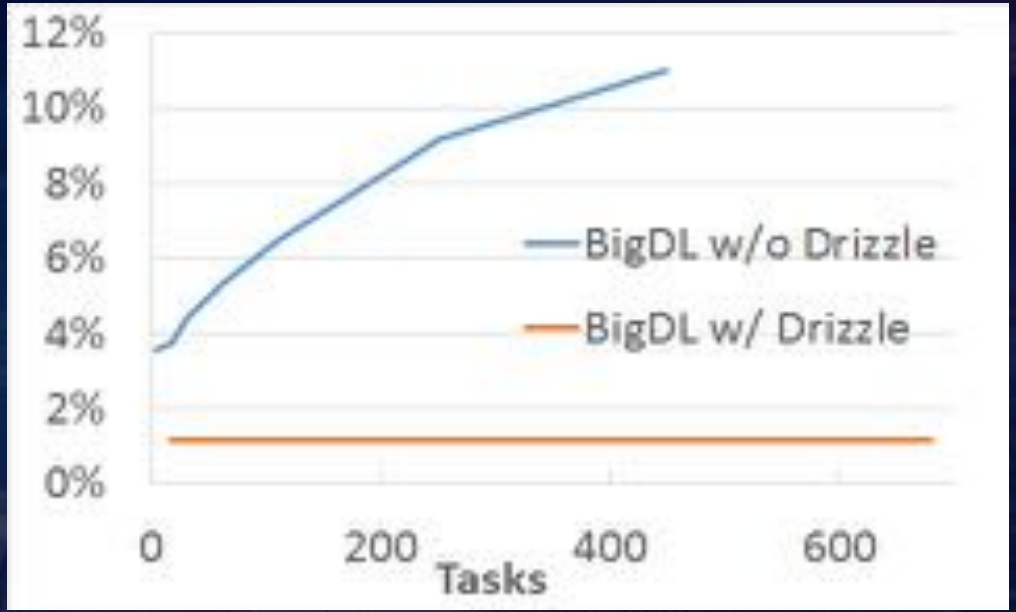
- **Run a single, multi-threaded task on each worker**
- **Achieve high scalability on large clusters (e.g., up to 256 servers**

Source: Accelerating Deep Learning Training with BigDL and Drizzle on Apache Spark, Shivaram Venkataraman, Ding Ding, and Sergey Ermolin, (https://rise.cs.berkeley.edu/blog/accelerating-deep-learning-training-with-bigdl-and-drizzle-on-apache-spark/)

Reducing Scheduling Overheads Using Drizzle

Scaling to even larger (>500) workers

- **Iterative model training**
  - **Same operations run repeatedly**
- **Drizzle**
  - **A low latency execution engine for Spark**
  - ***Group scheduling* for multiple iterations of computations at once**



Source: Accelerating Deep Learning Training with BigDL and Drizzle on Apache Spark, Shivaram Venkataraman, Ding Ding, and Sergey Ermolin, (https://rise.cs.berkeley.edu/blog/accelerating-deep-learning-training-with-bigdl-and-drizzle-on-apache-spark/)

Real-World Applications

Object detection and image feature extraction at JD.com

Image similarity based house recommendation for MLSlisting

Transfer learning based image classifications for World Bank

Case Study: Image Recognition
JD.Com*

Result
4X GaiN

in performance with Intel® Xeon® CPU cluster, per JD.com. Processing ~380M images with Intel Xeon CPU E5-2650 v4 @ 2.20GHz with 1200 YARN cores.

"We found BigDL using Intel Xeon® processors as the best platform for production deployment of our SSD (single-shot multi-box detector) solution on our Spark cluster"

Dennis Weng, VP of JD.com
Head of JD Big Data Platform Division

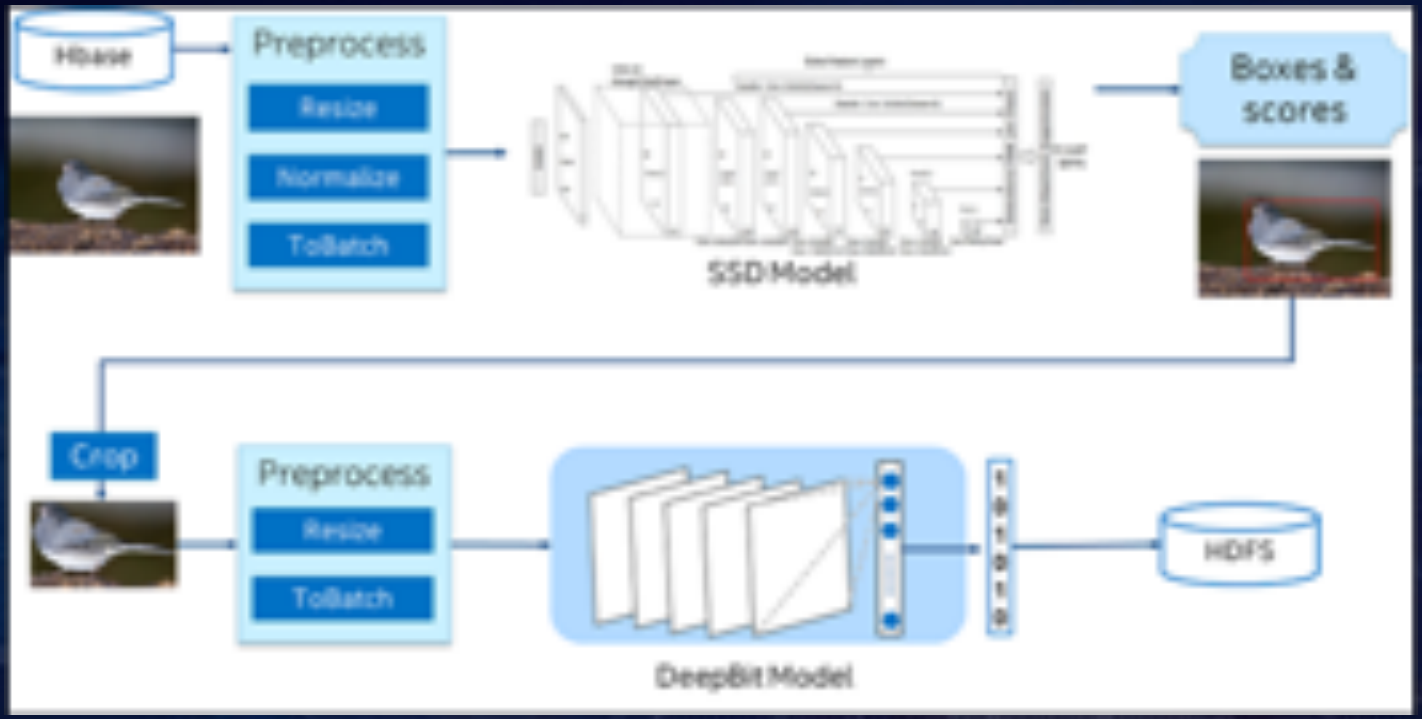
Client: JD.Com*, second largest online retailer in China, with approximately 25 million registered users.

Challenge: Building deep learning applications such as image similarity search on GPU cluster was costly & complex. Technical issues included high latency when downloading graphic data from Apache Hbase* & complicated data pre-processing in GPU environment.

Solution: Switched from GPU to CPU cluster. Using Apache Spark* with BigDL, running on Intel® Xeon® processors. Intel delivered an image detection & extraction pipeline. BigDL used to build deep learning models for image recognition & feature extraction.

https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

Object Detection and Image Feature Extraction at JD.com



Applications

Large-scale image feature extraction

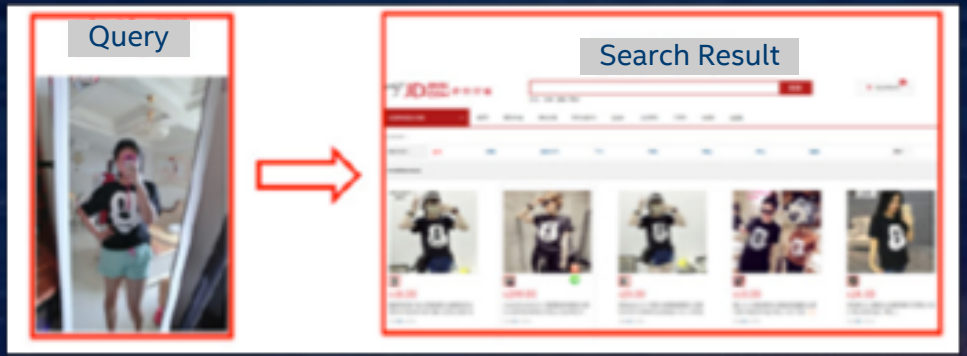
- Object detect (remove background, optional)
- Feature extraction

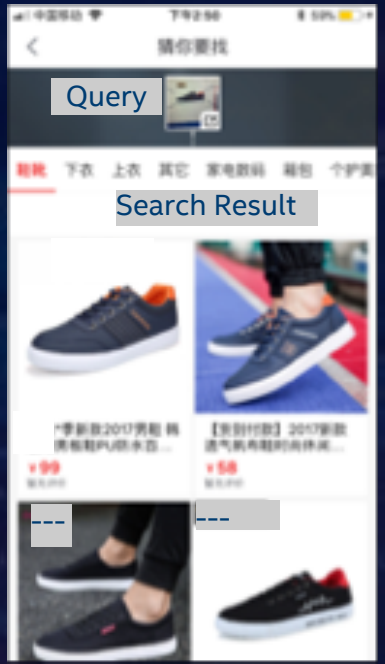
Application

- 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





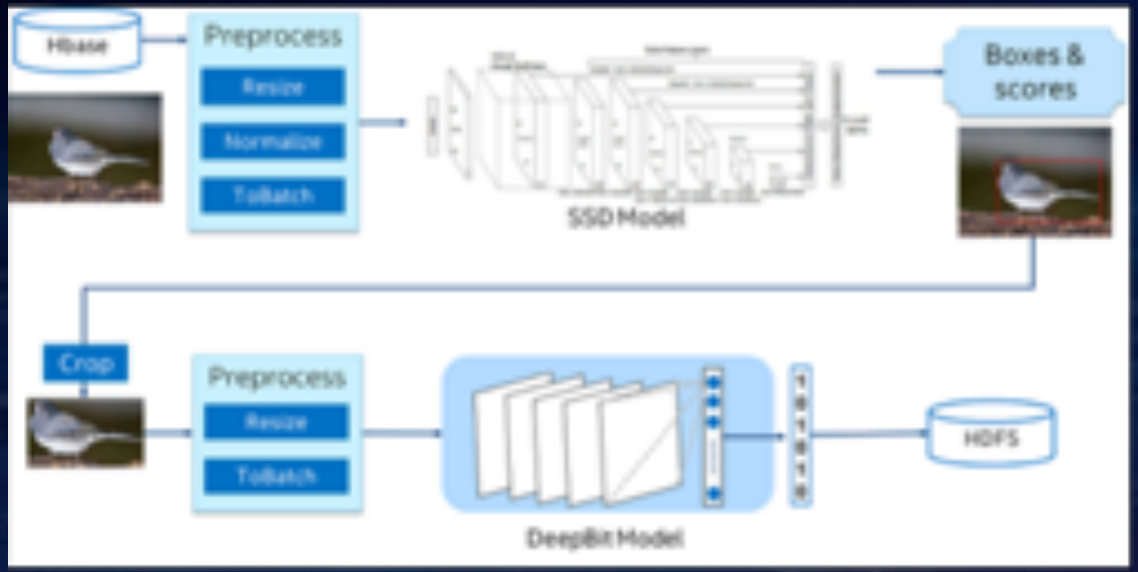
Source: “Bringing deep learning into big data analytics using BigDL”, Xianyan Jia and Zhenhua Wang, Strata Data Conference Singapore 2017

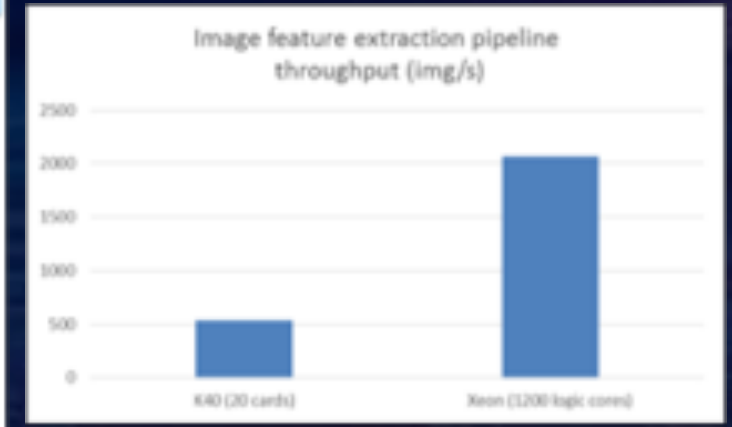
Challenges of Productionizing Large-Scale Deep Learning Solutions

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

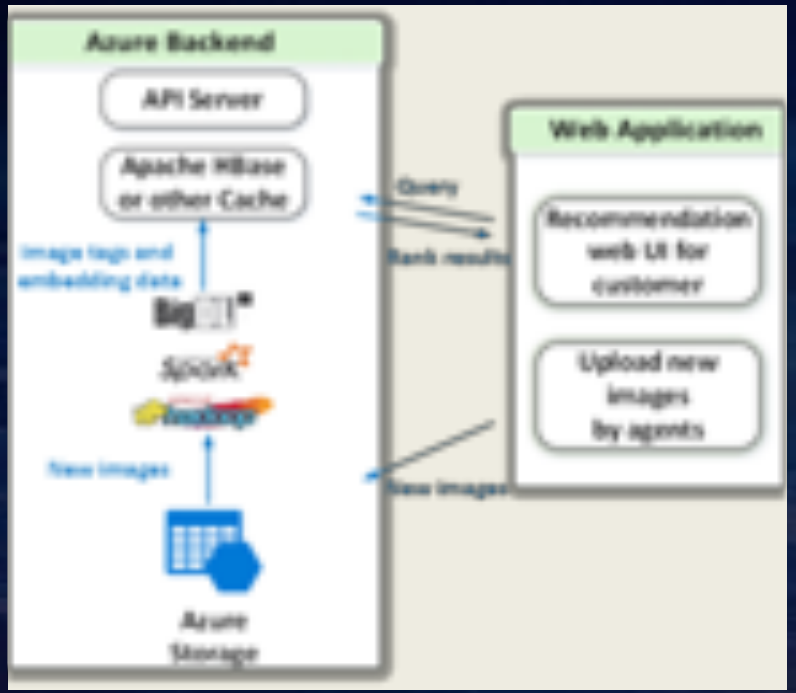




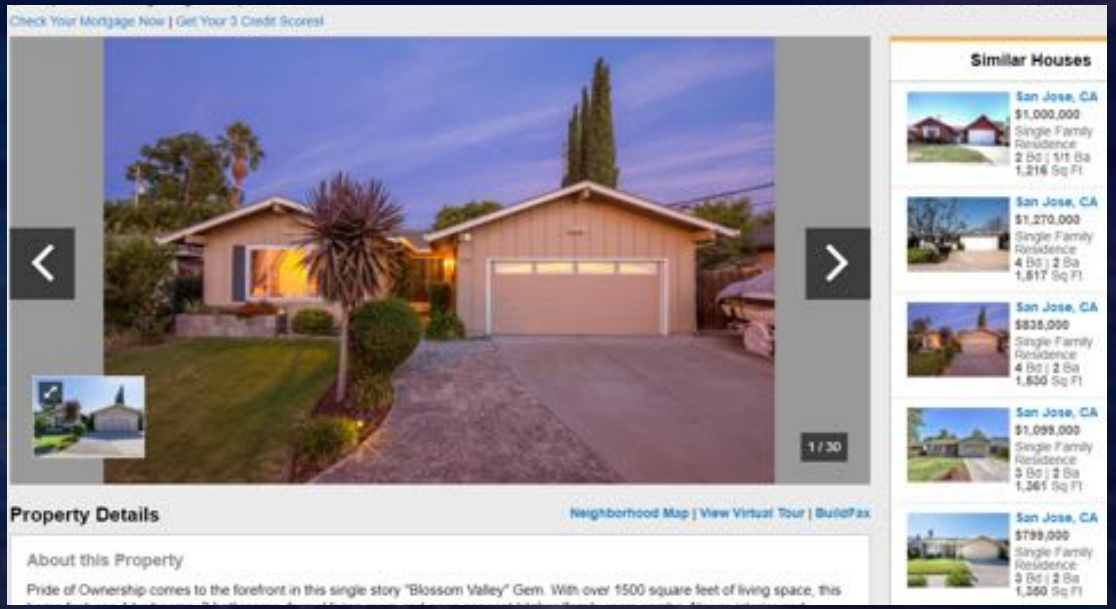
- Reuse existing Hadoop/Spark clusters for deep learning with no changes (image search, IP protection, etc.)
- Efficiently scale out on Spark with superior performance (***3.83x*** speed-up vs. GPU severs) as benchmarked by JD

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

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

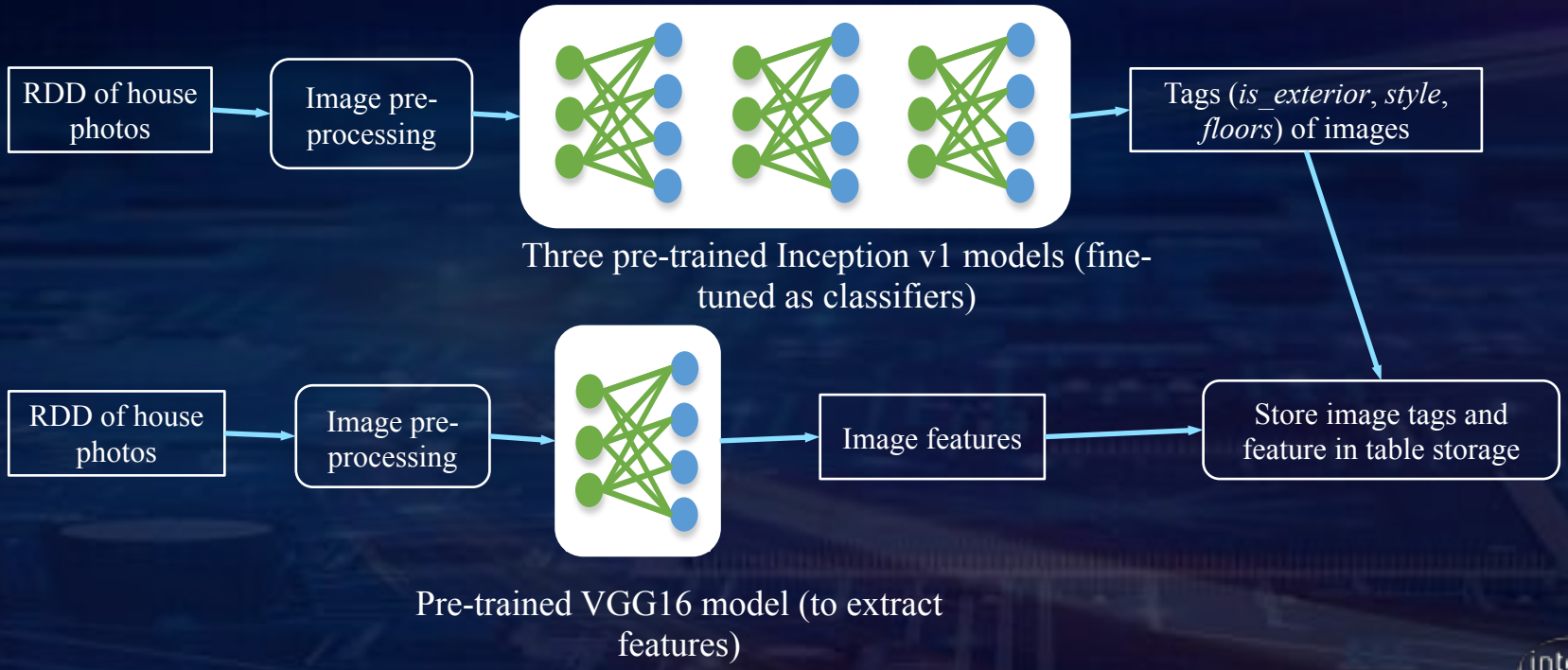


Image Similarity Based House Recommendation for MLSlistings

Notebook:

https://github.com/intel-analytics/analytics-zoo/blob/master/apps/image-similarity/Image%20similarity.ipynb

Case Study: Image Classification

World Bank

Result

High accuracy results using an AWS Databricks platform to train a dataset consisting of almost 1 million images in 69 categories, with near linear scaling on a partial dataset*

https://databricks.com/session/using-crowdsourced-images-to-create-image-recognition-models-with-bigdl

https://itpeernetwork.intel.com/artificial-intelligence-world-bank-image-recognition/

Client: The International Comparison Program (ICP) in the World Bank Development Data Group

Challenge: The World Bank team needed to automate the process of confirming that the crowd-sourced photos, gathered from cellphone contributors from 15 countries, were accurately classified into one of 162 categories ranging from food to footwear, and to remove personally identifiable information (PII) from the photos.

Solution: Utilized Intel's BigDL framework (a distributed deep-learning library for Apache Spark*) and an AWS Databricks* platform running on Intel® Xeon® Processors (AWS R4.8xlarge instance with 20 nodes) to help classify more than 1 million crowdsourced photos before sharing the dataset with the public.

Transfer Learning Based Image Classifications for World Bank



Classifying Real Food Images is not a Cat vs. Dog Problem

Source: Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL, Maurice Nsabimana and Jiao Wang, Spark Summit 2018

Project Layout

Phase 1:

- Image preprocessing (eliminate poor quality images and invalid images)
- Classify images (by food type) to validate existing labels

Phase 2:

- Identify texts in the image and make bounding box around them
- Text recognition (words/sentences in the image text)
- Determine whether text contains PII (personal identifiable information)
- Blur areas with PII text

Source: Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL, Maurice Nsabimana and Jiao Wang, Spark Summit 2018

Code – Phase 1

Fine-tuning Training

```
# get model
pretrained_model_path = path.join(MODEL_ROOT,"bigdl_inception_v1_imagenet_0_4_0.model")
n_classes = len(label_dict)# label categories
full_model = Net.load_bigdl("dbfs:" + pretrained_model_path)
# create a new model by remove layers after pool5/drop_7x7_s1
model = full_model.new_graph(["pool5/drop_7x7_s1"])

inputNode = Input(name="input", shape=(3, 224, 224))
inception = model.to_keras()(inputNode)
flatten = Flatten()(inception)
logits = Dense(n_classes)(flatten)

lrModel = Model(inputNode, logits)
```

```
creating: createZooKerasInput
creating: createZooKerasFlatten
creating: createZooKerasDense
creating: createZooKerasModel
```

Command took 4.74 seconds -- by Jiao.Wang@intel.com at 6/2/2018, 8:03:40 PM on 28-node-cluster

```
# train model
classifier = NNClassifier(lrModel, CrossEntropyCriterion(), train_transformer) \
        .setLearningRate(learning_rate)\
        .setBatchSize(batch_size)\
        .setMaxEpoch(no_epochs)\
        .setFeaturesCol("image")\
        .setValidation(EveryEpoch(), val_image, [Top1Accuracy()], batch_size)
start = time.time()
trained_model = classifier.fit(train_image)
end = time.time()
print("Optimization Done.")
print("Training time is: %s seconds" % str(end-start))
# + dt.datetime.now().strftime("%Y%m%d-%H%M%S")
```

Prediction and Evaluation

```
#predict
predict_model = trained_model.setBatchSize(batch_size)
predictionDF = predict_model.transform(test_image)
predictionDF.cache()
```

```
'''
Measure Test Accuracy w/Test Set
'''
evaluator = MulticlassClassificationEvaluator(labelCol="label",
                                              predictionCol="prediction",
                                              metricName="accuracy")
accuracy = evaluator.evaluate(predictionDF)
# expected error should be less than 10%
print("Accuracy = %g " % accuracy)
predictionDF.unpersist()
```

Source: Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL, Maurice Nsabimana and Jiao Wang, Spark Summit 2018

Result – Phase 1

- Fine tune with Inception v1 on a full dataset
- Dataset: 994325 images, 69 categories

Nodes	Cores	Batch Size	Epochs	Training Time (sec)	Throughput (images/sec)	Accuracy (%)
20	30	1200	12	61125	170	81.7

* This model training was performed using multinode cluster on AWS R4.8xlarge instance with 20 nodes

Source: Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL, Maurice Nsabimana and Jiao Wang, Spark Summit 2018

Next Steps – Phase 2

- Image Quality Preprocessing
  - Filter with print text only
  - Rescaling, Binarisation, Noise Removal, Rotation / Deskewing (OpenCV, Python, etc.)
- Detect text and bounding box circle text
- Recognize text
- Determine whether text contains PII (personal identifiable information)
  - Recognize PII with leading words
- Blur areas with PII text
  - Image tools

Source: Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL, Maurice Nsabimana and Jiao Wang, Spark Summit 2018

Case Study: Product Defect Detection

Midea*

Result

Midea

Working closely with Intel's Analytics Zoo team, Midea built a highly-optimized defect detection solution, and chose Intel® Xeon® Scalable 6130/6148 over GPU-based servers as it met their latency requirements and more easily integrated into their existing infrastructure

Spark

TensorFlow*

"Analytics Zoo from Intel provides a great tool for developing the end-to-end AI solutions, building pipelines across cloud and edge computing, and optimizing the hardware resources."

Zheng Hu, Director of Computer Vision Research Institute, Midea

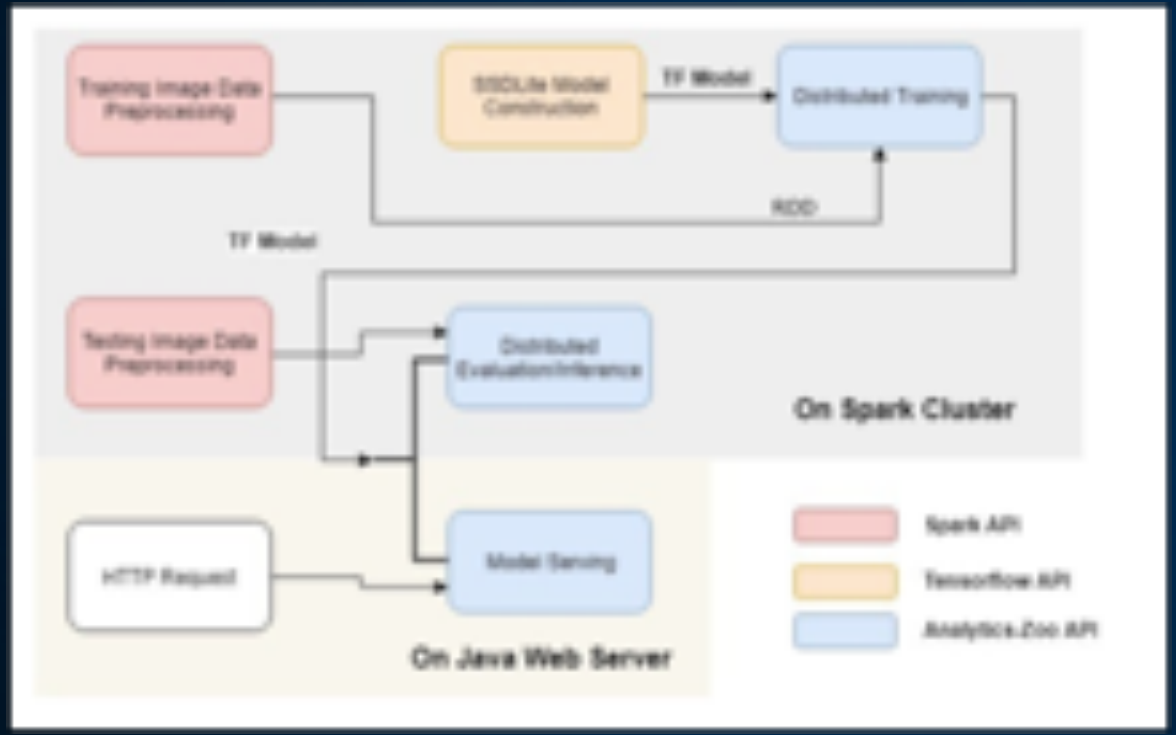
Client: Midea Group is a Chinese electrical appliance manufacturer with 21 manufacturing plants and 260 logistics centers across 200 countries

Challenge: Midea needed to eliminate defects caused by scratched surfaces, missing bolts, misaligned labeling on surfaces (glass, polished metal, painted), and human inspection was not able to meet target quality metrics or detection rate requirements.

Solution: An advanced defect inspection system built on top of Analytics Zoo, which provides a unified analytics + AI platform that seamlessly unites Spark, BigDL and TensorFlow* programs into an integrated pipeline. The system was based on Intel® Xeon Scalable 6130/6148 servers and Core i7 edge devices.

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

Midea*
End-to-end solution



*

Case Study: Customer Service Platforms

Microsoft Azure

Result

The deep learning model successfully interact with customer with ideal accuracy, and can learn from real data and evolve through time.

Client: Microsoft Azure China team

Challenge: The Azure documentation library contains a lot of useful materials and it need an AI-based customer support platform that is capable of saving human efforts as well as improving user experiences to some extent.

Solution: provides real-time chat functionality to the customer, our solution is to build two new, intelligent modules into the basic system (implemented using Analytics Zoo), the text classifier module and the QA ranker module.

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

Case Study: Customer Service Platforms

Microsoft Azure

Summary

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

- 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, managed and monitored with other workloads (*ETL, data warehouse, traditional ML, etc.*)

Partner With Us

learn

explore

engage

Great Resources at:

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

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

software.intel.com/ai

software.intel.com/bigdl

ai.intel.com

Use Intel's performance-optimized libraries & frameworks

ai.intel.com/framework-optimizations/

Contact your Intel representative for help and POC opportunities


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**.