BigDL DNN for Apache Spark

- BigDL makes Deep Learning without GPUs **POSSIBLE**
- BigDL *embraces* distributed and fault-tolerant Deep Learning
- BigDL code runs **without changes** on a single machine or on a cluster.
- BigDL leverages tried-and-true Apache Spark framework
- BigDL has the same **features and capabilities** as TF, Keras, PyTorch, etc
- BigDL can **exchange models** with TF, Keras, Torch
- High performance via:
  - Written in Apache Spark’s native language, Scala (Python API as well)
  - Intel’s latest-and-greatest MLK library (pre-bundled with BigDL)
- Open-sourced

[bigdl-project.github.io](https://bigdl-project.github.io)  [software.intel.com/bigdl](https://software.intel.com/bigdl)
BigDL is an open-source distributed deep learning library for Apache Spark* that can run directly on top of existing Spark or Apache Hadoop* clusters. It is ideal for DL Models TRAINING and INFERENCE.

Feature Parity & Model Exchange with TensorFlow*, Caffe*, Keras, Torch*

Lower TCO and improved ease of use with existing infrastructure

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

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

BigDL offers high performance deep learning for Apache Spark* on CPU infrastructure. It is designed and optimized for Intel® Xeon® processors. It is ideal for DL Models TRAINING and INFERENCE.

Powered by Intel® MKL and multi-threaded programming

BigDL is available at bigdl-project.github.io and software.intel.com/bigdl.
Jupyter, Zeppelin notebooks and TensorBoard support
WHY BIGDL?

• Does BigDL offer lower TCO? **YES**
• **It is the story of efficient enterprise scale-out and resource leveraging:**
  - Reuse existing Spark clusters
    - Leverage existing infra of deployment, monitoring, support, etc.
  - Reuse existing Spark data pipelines
    - Incrementally add AI capabilities to existing flow rather than develop brand-new pipelines
  - Compute where your data is (data efficiency).
    - In FinTech, often you can’t even move data outside of “datalake”
  - Written in Scala – leverage expertise of Spark engineers (+Python API)
  - No need to change Spark infrastructure: “spark-submit bigdl_app.jar”
  - That’s How Customers Want It
Customer Testaments (based on their own framework comparison):

- TensorFlow/Caffe runs on specialized HW+interconnect - $$$
- Open MP Implementation of TensorFlow/Caffe-on-Spark conflicts with Spark’s JVM threading – lower performance
- TensorFlow/Spark can only interact with the rest of the analytics pipelines in a very coarse-grained fashion

bigdl-project.github.io  
software.intel.com/bigdl
WHO IS BUILDING WHAT WITH BIGDL?

CONSUMER
- Gigaspaces
- MLS Listings
- Jobs Search Engine
- Call center routing
- Image similarity search
- Smart job search

HEALTH
- UCSF
- UnionPay
- ChinaLife (Insurance)
- Major credit card issuer
- Analysis of 3D MRI models for knee degradation
- Fraud detection
- Recommendation
- Customer/Merchant Propensity

FINANCE
- JD.Com
- Steel manufacturing
- Image feature extraction (Inference)

RETAIL

MANUFACTURING
- Steel manufacturing

SCIENTIFIC COMPUTING
- Cray
- Steel Surface defect detection
- Weather forecasting
If you looked at this house.....

Non-real time indexing/bucketizing of similar images in the database.

Image similarity ("distance") becomes an extra parameter in addition to area, location, size, price, etc.

You will want to look at this one, too

Runs periodically on the refreshed database.

Needs to be scalable nationwide. Distributed compute solution needed (Spark)

MLS Listings image similarity PoC demo is ready and available here: https://homes-prod-homes-poc.azurewebsites.net/
Real estate: Search ranking

Parametric search results are ranked based on image similarity.
Data Governance - House Style in MLS entries

Business Need: validate a standardized “house style” DB entry made by a realtor.

- Ranch
- Traditional
- Mediterranean
- Modern

Trained GoogleNet Model → 1024-long embeddings vector → SoftMax Linear Classifier → ‘Ranch’ ‘Traditional’ ‘Mediterranean’ ‘Modern’
RecEngine: Customer-Merchant Purchase Propensity

• How likely is a customer to return to the merchant in the next week/month?
• How to entice ($$) an existing customer to get back in the door (targeted offer)?
• What does it take ($$) to get a new customer to in the door (re-targeted offer)?

• SCALABILITY is the key:
• 100s of banks, 1000s of merchants, but need only a few models!

<table>
<thead>
<tr>
<th>Mlib AIS</th>
<th>BigDL NCF</th>
<th>BigDL WAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUROC: A</td>
<td>AUROC: A+23%</td>
<td>AUROC: A+20% (3 % down)</td>
</tr>
<tr>
<td>AUPRCs: B</td>
<td>AUPRCs: B+31%</td>
<td>AUPRCs: B+30% (1% down)</td>
</tr>
<tr>
<td>recall: C</td>
<td>recall: C+18%</td>
<td>recall: C+12% (4 % down)</td>
</tr>
<tr>
<td>precision: D</td>
<td>precision: D+47%</td>
<td>precision: D+49% (2 % up)</td>
</tr>
<tr>
<td>20 precision: E</td>
<td>20 precision: E+51%</td>
<td>20 precision: E+54% (3% up)</td>
</tr>
</tbody>
</table>
Degenerative joint disease or “wear and tear” arthritis, osteoarthritis (OA) is the most common chronic condition of the joints. It occurs when the cartilage or cushion between joints breaks down leading to pain, stiffness and swelling.

- **52.5 million** (22.7%) adults - self-reported doctor-diagnosed OA
- **22.7 million** (9.8% adults) have arthritis-attributable activity limitation.
MEDICAL IMAGING - CLASSIFY OSTEOARTHRITIS DEGENERATIVE FEATURES

Phase 1

Knee MRI

Multi-tissue Detection and Segmentation

- Cartilage
- Meniscus
- Bone
- Ligaments

Morphometry

Image Feature Extraction (3D)

Phase 2

Classifier

Multimodal Data Integration

Meta Data

Age, Gender, BMI, PROMS...

BMEL

Full-Thickness Lesion

Subchondoral cyst

HTTPS://GITHUB.COM/INTEL-ANALYTICS/BIGDL
MEDICAL IMAGING - RESULTS

Intra-class correlation coefficients for intra-observer agreement for WORMS grading of the meniscus ranges from 0.801 to 0.928

Binary Model:

- Top model training accuracy: **No lesion: 98.5%, lesion: 99.7%**
- Top model testing accuracy: **No lesion: 94.1%, lesion: 67.6%**

Three Class Model:

- Top model training accuracy: No lesion: 99.9%, mild lesion: 100%, severe lesion: 99.9%
- Top model testing accuracy: **No lesion: 96.0%, mild lesion: 52.6%, severe lesion: 50.0%**
BIGDL 2018 ROADMAP
2018 – THE YEAR OF “ANALYTICS ZOO”

**Historical Background**
- BigDL has achieved feature-parity with other major frameworks.
- Customer are no longer asking for new features (not as much, anyway)
- Customer conversations have shifted towards deployments

**Customer conversations in 2018: “We know BigDL. Help us deploy it”**
- Deployment of BigDL in production.
- End-to-end data pipeline with BiGDL at its core
- Operationalizing the deployment.
- Docker/Kubernetes
- High-level APIs (Dataframes, ML Pipelines, Keras)
- Analytics Zoo with pre-built BigDL Spark pipelines and ‘reference designs’
- Marketplace deployments in Azure, AWS, GCP

[bigdl-project.github.io](https://bigdl-project.github.io)  [software.intel.com/bigdl](https://software.intel.com/bigdl)
**“Analytics Zoo” and BigDL 0.5.0 Release**

**BigDL 0.5.0 (see release notes for more info)**
- Keras-like API (Scala and Python). Run your Keras code on Apache Spark through BigDL.
- Loading Tensorflow dynamic models (e.g. LSTM, RNN). More TF operations.
- Combining data preprocessing and neural network layers in the same model (to make model deployment easy)
- BigDL perf tuning (eg. BCECriterion, rmsprop, LeakyRelu, etc.)
- Add DataFrame-based image reader and transformer

**BigDL “Analytics Zoo”**
- Reference designs
  - Fraud detection, time series prediction, sentiment analysis, chatbot, etc.
- Predefined models
  - Object detection, image classification, text classification, recommendations, etc.

bigdl-project.github.io  software.intel.com/bigdl
// Load caffe model
val modelCaffe = Module.loadCaffeModel(caffeDefPath, caffeModelPath)

// Load tensorflow model
val modelTF = Module.loadTF(graphFile, inputs, outputs, byteOrder, binFile)

// Load torch model
val modelTorch = Module.loadTorch(path)

# compose a pipeline that includes feature transform, pretrained model and Logistic Regression
transformer = NNImageTransformer(
  image.Pipeline([Resize(256, 256), CenterCrop(224, 224), ChannelNormalize(123.0, 117.0, 104.0)])
).setInputCol("image").setOutputCol("features")

# Set up a linear classifier with SoftMax
lrModel = Sequential().add(Linear(1024, numClasses)).add(LogSoftMax())
classifier = NNClassifier(lrModel, ClassNLLCriterion(), [1024]) \
  .setLearningRate(1e-3).setBatchSize(40).setMaxEpoch(100).setFeaturesCol("embedding")

# Set up Spark pipeline and Launch fit optimizer
pipeline = Pipeline(stages=[transformer, pretrainedNNModel, classifier])
HouseStyleModel = pipeline.fit(trainingDF)
DEEP LEARNING WITH BIGDL/SPARK

Node Scaling with BigDL
- Intel® Xeon® Scalable 8180 Platinum Processor - 8 nodes
- Intel® Xeon® Scalable 8180 Platinum Processor - 16 nodes

Generational performance increase with BigDL
- E5-2699v4 (16 nodes)
- Platinum 8180 (16 nodes)

Optimization Notice: Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit: http://www.intel.com/performance

Source: Intel measured as of August 2017.

GET EXCELLENT MULTI-NODE SCALING AND GENERATIONAL PERFORMANCE WITH YOUR EXISTING HARDWARE

bigdl-project.github.io software.intel.com/bigdl
BUILDING & DEPLOYING WITH BIGDL

PLATFORMS
- bluedata
- cloudera
- databricks
- CRAY
- KINGSOFT
- Dell
- Quasar
- Lightbend

CLOUD SERVICE PROVIDERS
- Alibaba Cloud
- AWS
- Azure
- IBM Cloud
- Google Cloud Platform

SOLUTIONS
- JD.COM
- UnionPay
- Gigaspaces
- MLS Listings
- UCSF
- And Many More...

Open Source Community support:
2270+ Stars | 500+ Forks | 50 Contributors

https://bigdl-project.github.io

*Other names and brands may be claimed as the property of others.
**BIGDL IN THE CLOUD**

- **Microsoft Azure**
  - [Azure Marketplace](https://marketplace.windowsazure.com)
  - BigDL Spark Deep Learning

- **AWS Marketplace**
  - BigDL with Apache Spark
  - [BigDL](https://aws.amazon.com/marketplace/seller?sellerId=INTEL)
  - Latest Version: BigDL_0.5.0

- **Google Cloud**
  - BigDL for deep learning with Apache Spark and Google Cloud
  - [Google Cloud Big Data and Machine Learning Blog](https://cloud.google.com/blog)

- **Bigdl-project.github.io**
- **software.intel.com/bigdl**
CONCLUSION

• BigDL makes Deep Learning without GPUs POSSIBLE
• BigDL embraces distributed and fault-tolerant Deep Learning
• BigDL code runs without changes on a single machine or on a cluster.
• BigDL leverages tried-and-true Apache Spark framework
• BigDL has the same features and capabilities as TF, Keras, PyTorch, etc
• BigDL can exchange models with TF, Keras, Torch
• High performance via:
  • Written in Apache Spark’s native language, Scala (Python API as well)
  • Intel’s latest-and-greatest MLK library (pre-bundled with BigDL)
• Open-sourced
Notices and Disclaimers

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.

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

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.

The cost reduction scenarios described are intended to enable you to get a better understanding of how the purchase of a given Intel based product, combined with a number of situation-specific variables, might affect future costs and savings. Circumstances will vary and there may be unaccounted-for costs related to the use and deployment of a given product. Nothing in this document should be interpreted as either a promise of or contract for a given level of costs or cost reduction.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

No computer system can be absolutely secure.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel processors of the same SKU may vary in frequency or power as a result of natural variability in the production process.

© 2017 Intel Corporation. Intel, the Intel logo, Xeon, Xeon Phi, Xeon Phi logos and Xeon logos 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.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION, INCLUDING LIABILITY FOR INFRINGEMENT OR MISUSE OF ANY THIRD PARTY PATENT RIGHT, OTHER THAN AS EXPRESSLY PROVIDED FOR IN THE EXPRESS LIMITED WARRAN


### BigDL Configuration Details

<table>
<thead>
<tr>
<th>Benchmark Segment</th>
<th>AI/ML/DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark type</td>
<td>Training</td>
</tr>
<tr>
<td>Benchmark Metric</td>
<td>Training Throughput (images/sec)</td>
</tr>
<tr>
<td>Framework</td>
<td>BigDL master trunk with Spark 2.1.1</td>
</tr>
<tr>
<td>Topology</td>
<td>Inception V1, VGG, ResNet-50, ResNet-152, 50, 152</td>
</tr>
<tr>
<td># of Nodes</td>
<td>8, 16 (multiple configurations)</td>
</tr>
<tr>
<td>Platform</td>
<td>Purley</td>
</tr>
<tr>
<td>Sockets</td>
<td>2S</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel ® Xeon ® Scalable Platinum 8180 Processor (Skylake): 28-core @ 2.5 GHz (base), 3.8 GHz (max turbo), 205W</td>
</tr>
<tr>
<td></td>
<td>Intel ® Xeon ® Processor E5-2699v4 (Broadwell): 22-core @ 2.2 GHz (base), 3.6 GHz (max turbo), 145W</td>
</tr>
<tr>
<td>Enabled Cores</td>
<td>Skylake: 56 per node, Broadwell: 44 per node</td>
</tr>
<tr>
<td>Total Memory</td>
<td>Skylake: 384 GB, Broadwell: 256 GB</td>
</tr>
<tr>
<td>Memory Configuration</td>
<td>Skylake: 12 slots * 32 GB @ 2666 MHz Micron DDR4 RDIMMs, Broadwell: 8 slots * 32 GB @ 2400 MHz Kingston DDR4 RDIMMs</td>
</tr>
<tr>
<td>Storage</td>
<td>Skylake: Intel® SSD DC P3520 Series (2TB, 2.5in PCIe 3.0 x4, 3D1, MLC), Broadwell: 8 * 3 TB Seagate HDDs</td>
</tr>
<tr>
<td>Network</td>
<td>1 * 10 GbE network per node</td>
</tr>
<tr>
<td>OS</td>
<td>CentOS Linux release 7.3.1611 (Core), Linux kernel 4.7.2.el7.x86_64</td>
</tr>
<tr>
<td>HT</td>
<td>On</td>
</tr>
<tr>
<td>Turbo</td>
<td>On</td>
</tr>
<tr>
<td>Computer Type</td>
<td>Dual-socket server</td>
</tr>
<tr>
<td>Framework Version</td>
<td><a href="https://github.com/intel-analytics/BigDL">https://github.com/intel-analytics/BigDL</a></td>
</tr>
<tr>
<td>Topology Version</td>
<td><a href="https://github.com/google/inception">https://github.com/google/inception</a></td>
</tr>
<tr>
<td>Dataset, version</td>
<td>ImageNet, 2012; Cifar-10</td>
</tr>
<tr>
<td>Performance command (Inception v1)</td>
<td>spark-submit --class com.intel.analytics.bigdl.models.inception.TrainInceptionV1 --master spark://$master_hostname:7077 --executor-cores=36 --num-executors=16 --total-executor-cores=576 --driver-memory=60g --executor-memory=300g $BIGDL_HOME/dist/lib/bigdl-*SNAPSHOT-jar-with-dependencies.jar --batchSize 2304 --learningRate 0.0896 -f hdfs:///user/root/sequence/ --checkpoint $check_point_folder</td>
</tr>
<tr>
<td>Data setup</td>
<td>Data was stored on HDFS and cached in memory before training</td>
</tr>
<tr>
<td>Java</td>
<td>JDK 1.8.0 update 144</td>
</tr>
<tr>
<td>MKL Library version</td>
<td>Intel MKL 2017</td>
</tr>
</tbody>
</table>