INTRODUCTION TO BIGDL ON APACHE SPARK* : BRINGING DEEP LEARNING TO THE BIG DATA PLATFORM

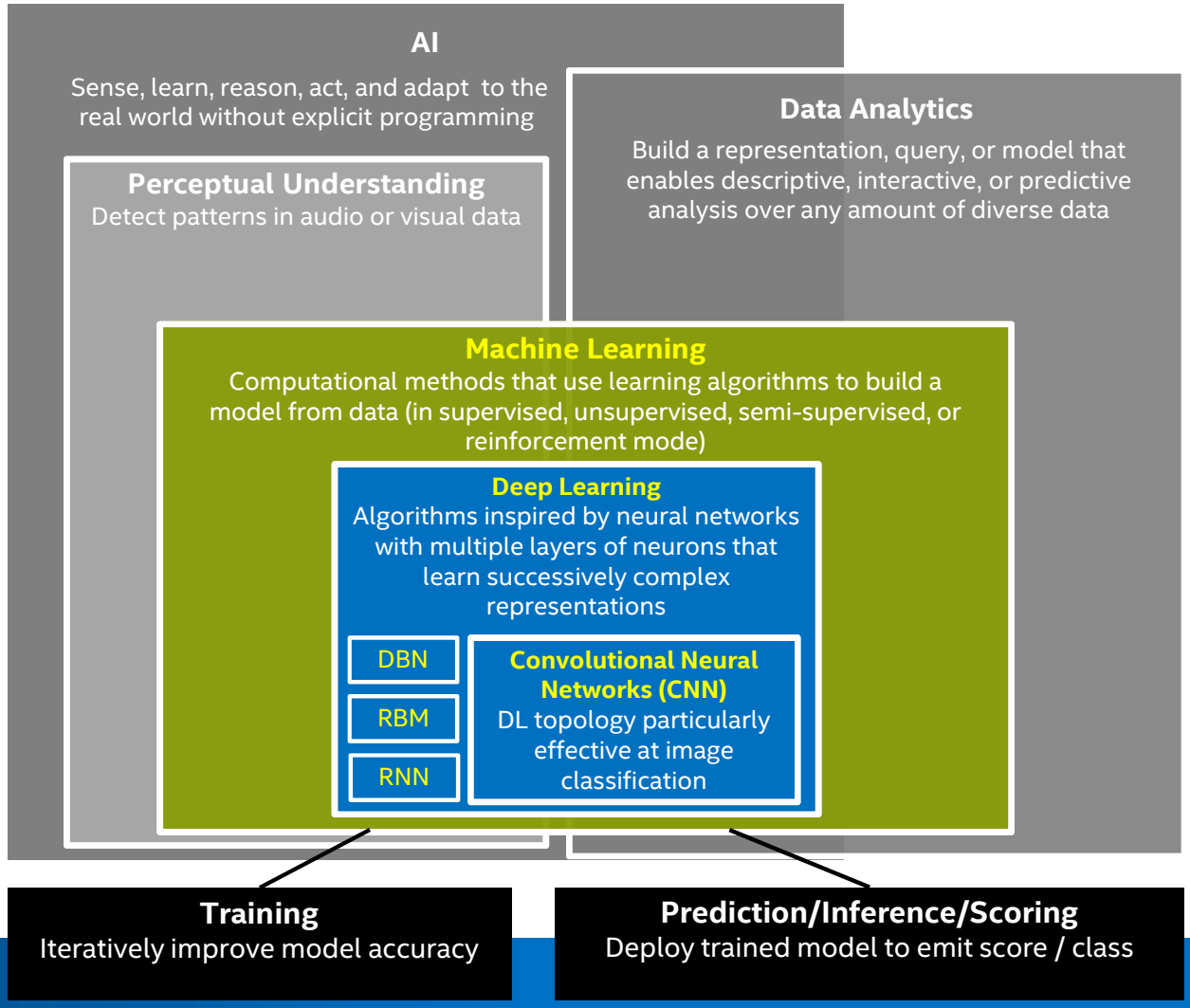
Radhika Rangarajan, Engineering PM

Ack: Jason Dai, Chief Architect, Big Data Technologies

CONTENT OUTLINE

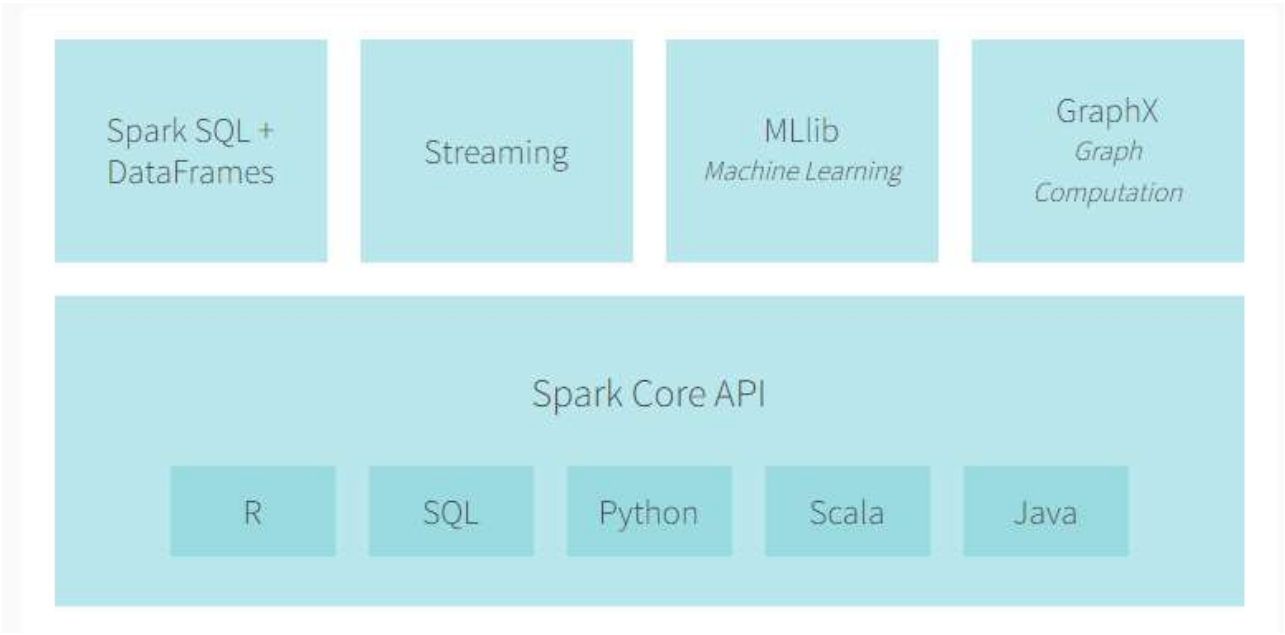
- Concept Refresh
- **WHY:** Motivation for BigDL
- **WHAT:** Architecture , Features and Code
- **WHERE**: Use Cases
- **HOW**: Get Started with BigDL

Concept Refresh



Concept Refresh

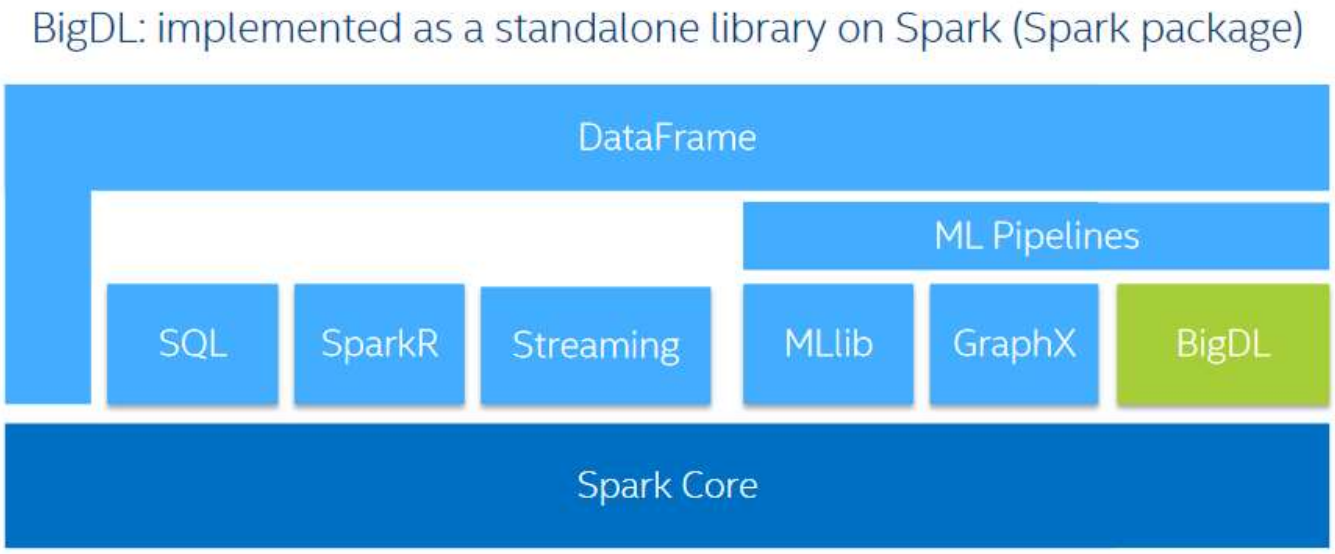
Apache Spark* 101



http://spark.apache.org/

What is BigDL?

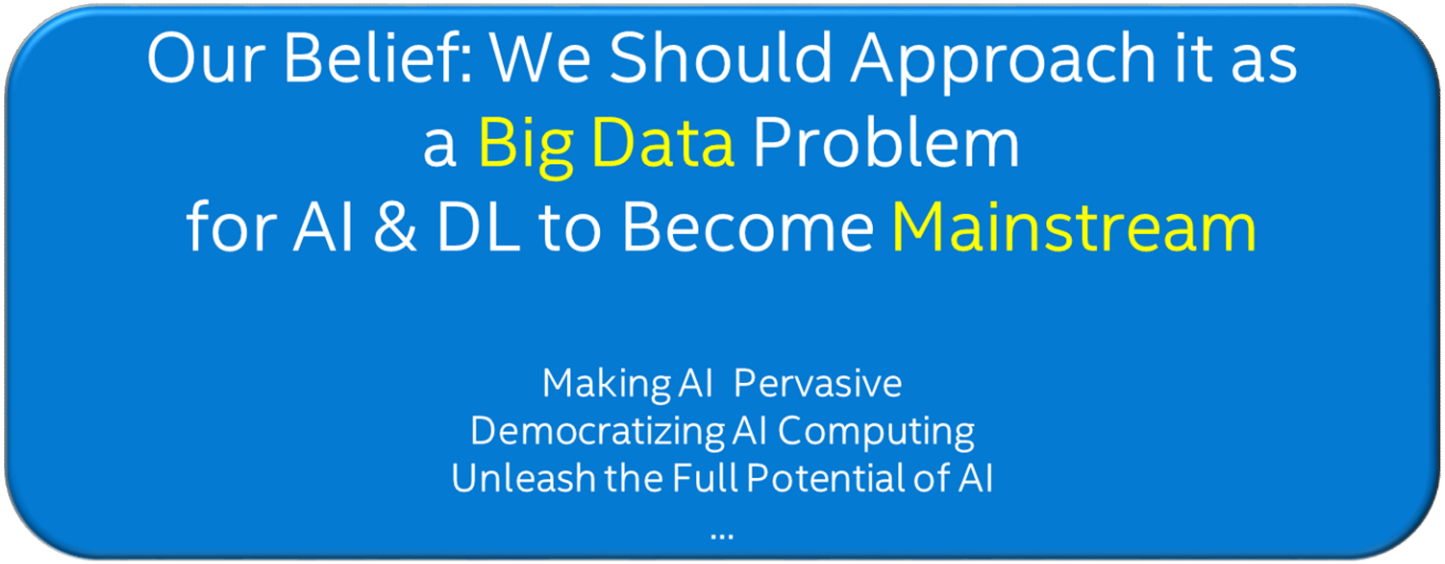
BigDL is a distributed deep learning library for Apache Spark*



WHY:
Motivation for BigDL

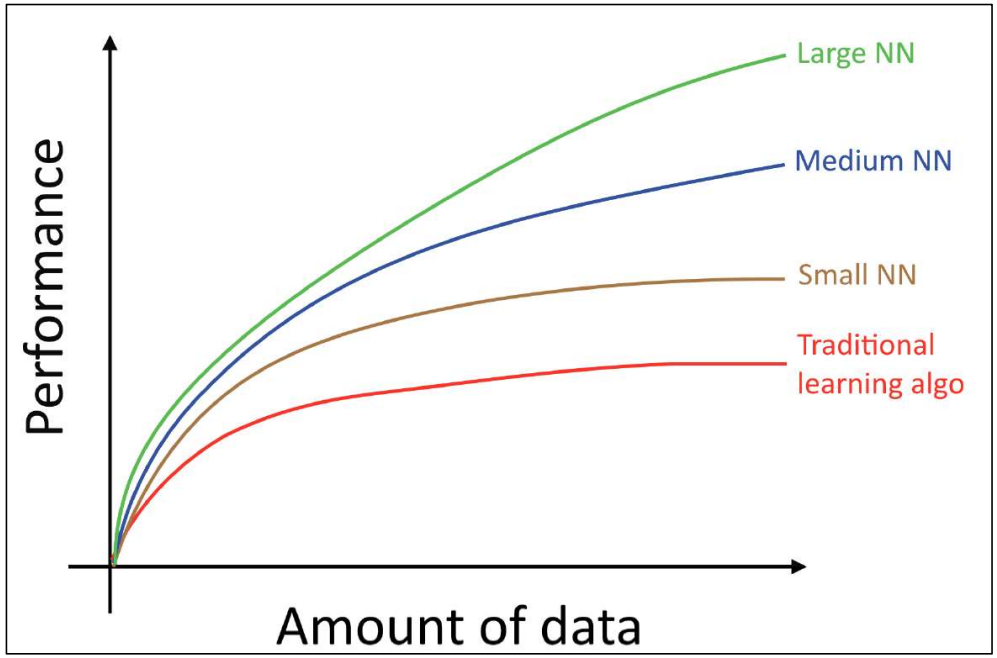
Motivation for BigDL

AI: HPC or Big Data?



Motivation for BigDL

Data Scale Driving Deep Learning Process



Andrew NG, Baidu.
NIPS 2016 Tutorial

Motivation for BigDL

Production ML/DL system is **Complex**

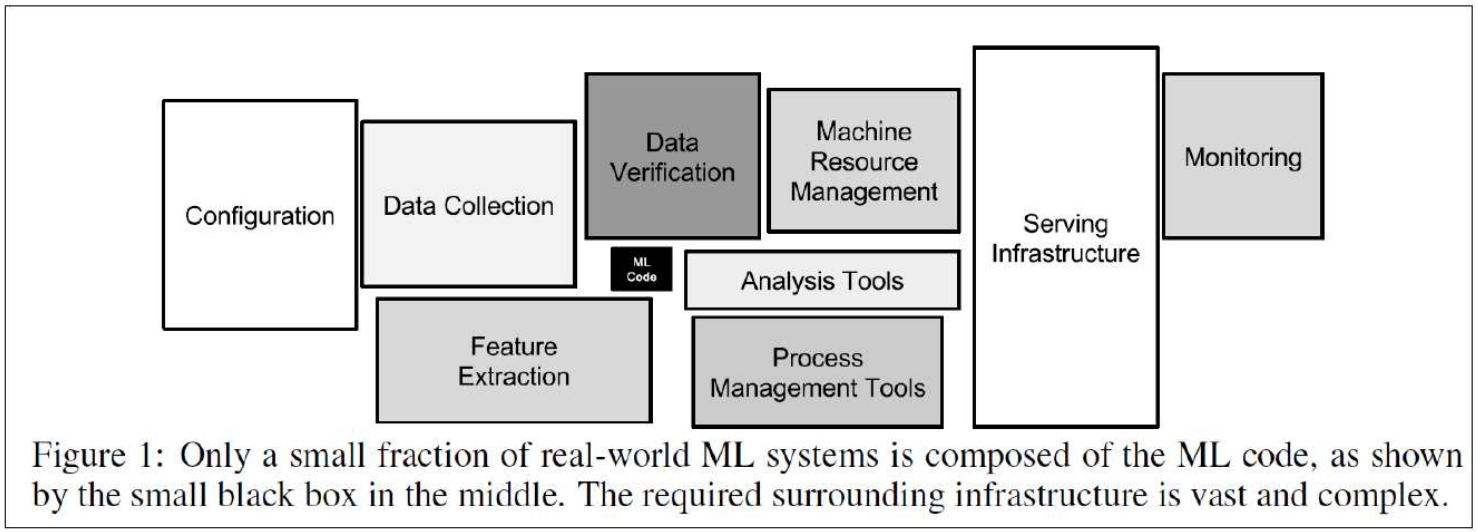
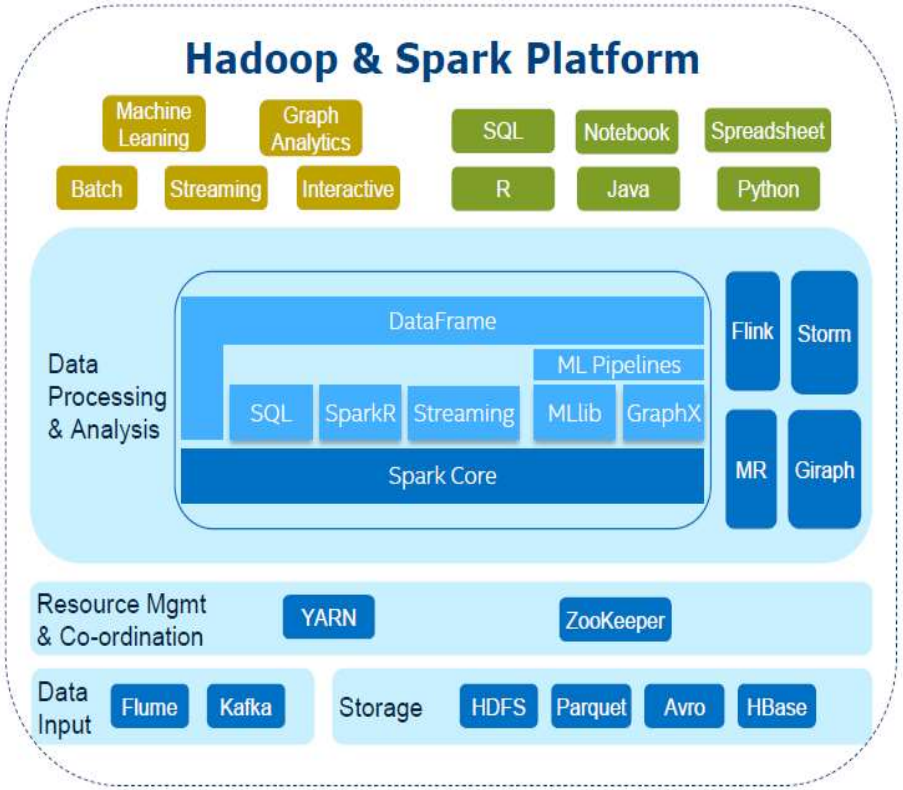


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

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

Motivation for BigDL

Unified Big Data Platform Driving Analytics and Data Science



Motivation for BigDL

The Missing Piece ?

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
- ...

BigDL – It is a BIG DEAL!

Open sourced on Dec 30, 2016

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

BigDL: Distributed Deep learning on Apache Spark

What is BigDL?

BigDL is a distributed deep learning library for Apache Spark; with BigDL, users can write their deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters.

- **Rich deep learning support.** Modeled after Torch, BigDL provides comprehensive support for deep learning, including numeric computing (via Tensor) and high level neural networks; in addition, users can load pre-trained Caffe or Torch models into Spark programs using BigDL.
- **Extremely high performance.** To achieve high performance, BigDL uses Intel MKL and multi-threaded programming in each Spark task. Consequently, it is orders of magnitude faster than out-of-box open source Caffe, Torch or TensorFlow on a single-node Xeon (i.e., comparable with mainstream GPU).
- **Efficiently scale-out.** BigDL can efficiently scale out to perform data analytics at "Big Data scale", by leveraging Apache Spark (a lightning fast distributed data processing framework), as well as efficient implementations of synchronous SGD and all-reduce communications on Spark.

Why BigDL?

You may want to write your deep learning programs using BigDL if:

- You want to analyze a large amount of data on the same Big Data (Hadoop/Spark) cluster where the data are stored (in, say, HDFS, HBase, Hive, etc.).
- You want to add deep learning functionalities (either training or prediction) to your Big Data (Spark) programs and/or workflow.
- You want to leverage existing Hadoop/Spark clusters to run your deep learning applications, which can be then dynamically shared with other workloads (e.g., ETL, data warehouse, feature engineering, classical machine learning, graph analytics, etc.)

How to use BigDL?

- To learn how to install and build BigDL (on both Linux and macOS), you can check out the Build Page
- To learn how to run BigDL programs (as either a local Java program or a Spark program), you can check out the Getting Started Page
- To try BigDL out on EC2, you can check out the Running on EC2 Page
- To learn how to create practical neural networks using BigDL in a couple of minutes, you can check out the Tutorials Page
- For more details, you can check out the Documents Page (including Tutorials, Examples, Programming Guide, etc.)

Support

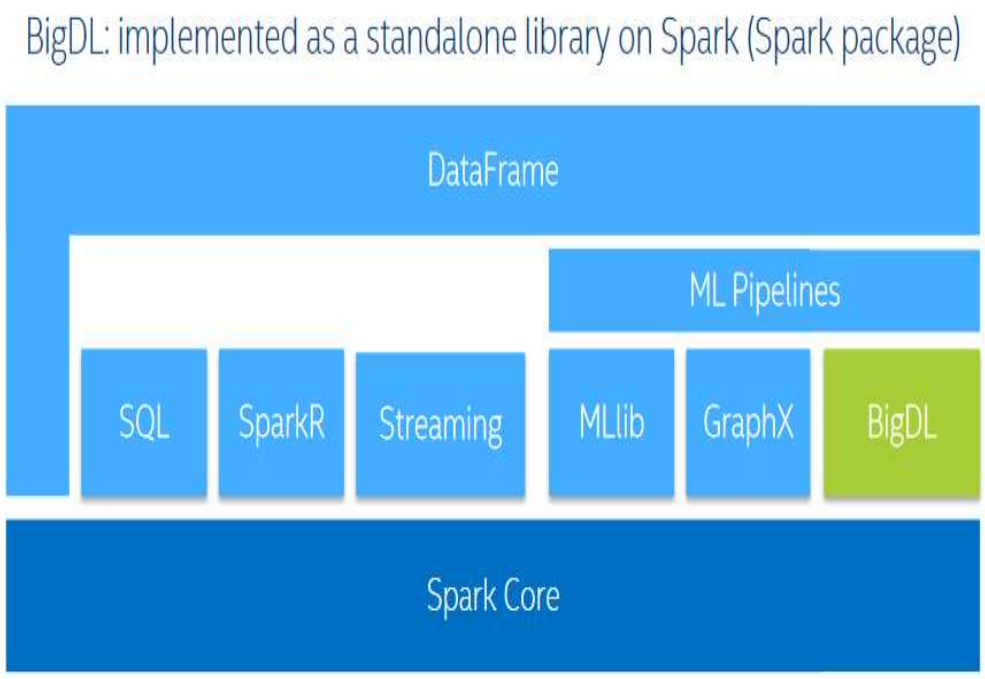
- You can join the BigDL Google Group (or subscribe to the Mail List) for more questions and discussions on BigDL
- You can post bug reports and feature requests at the Issue Page


WORKFLOW TOOLS/SDK
Intel® Deep Learning SDK (Training & Deployment)
BIG DATA ANALYTICS
hadoop
Spark
BigDL
DEEP LEARNING FRAMEWORK OPTIMIZATIONS
neon
TensorFlow
theano
Caffe
torch
Chainer
LOW LEVEL SOFTWARE PRIMITIVES
Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)
INTEL® ARCHITECTURE (IA)
XEON
XEON PHI
XEON
Arria 10
XEON
LAKE CREST

WHAT:
Architecture, Features and Code

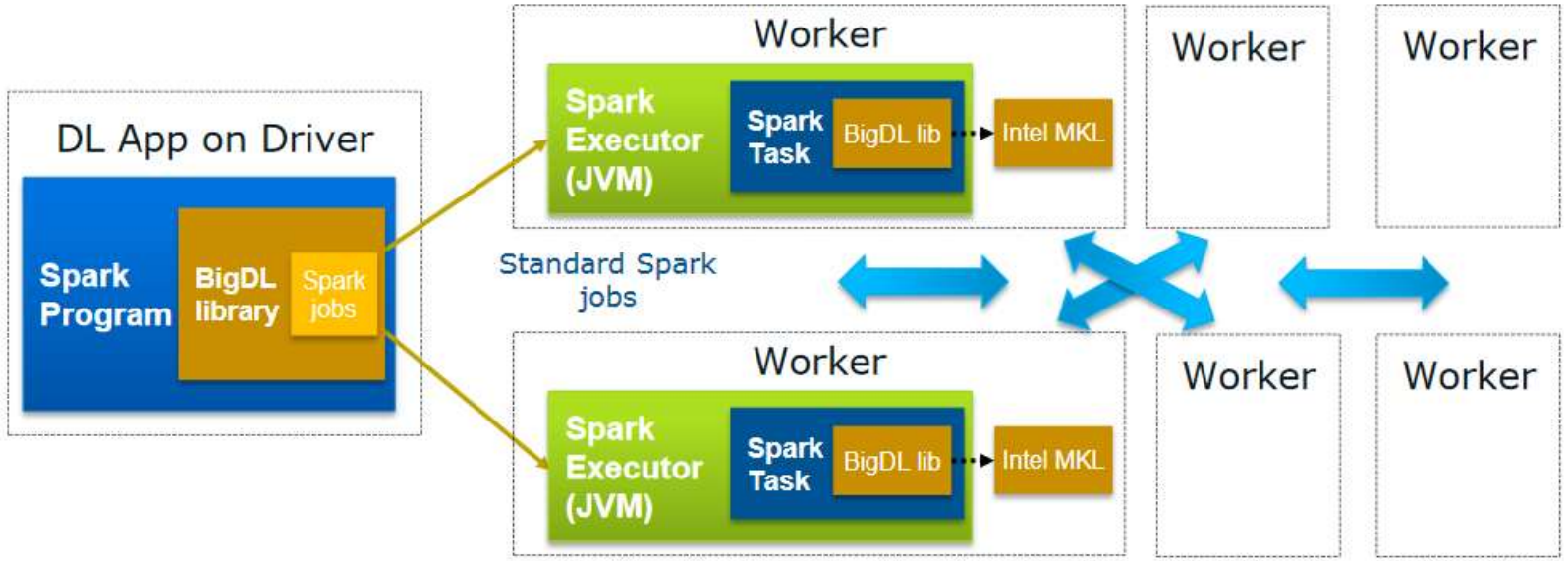
Architecture, Features and Code

- Write deep learning applications as standard Spark programs
- Run on top of existing Spark or Hadoop clusters (No change to the clusters)
- Rich deep learning support
- High performance powered by Intel MKL and multi-threaded programming
- Efficient scale-out with an all-reduce communications on Spark



Architecture, Features and Code

- Training using synchronous mini-batch SGD
- Distributed training iterations run as standard Spark jobs

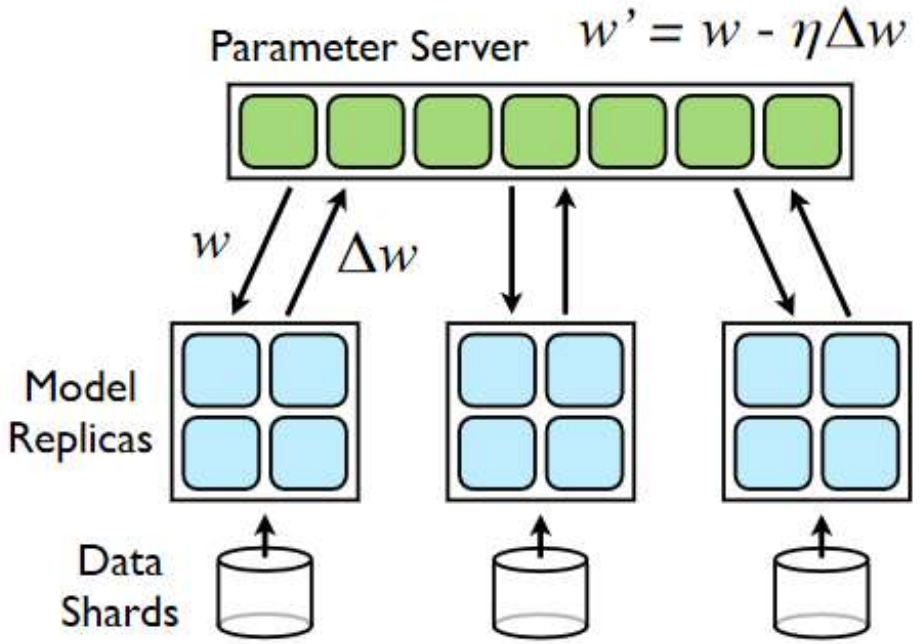


Architecture, Features and Code

Distributed Training

- Model Parallelism
- **Data Parallelism**

A distributed training example,
(Jeff Dean, NIPS 2012)

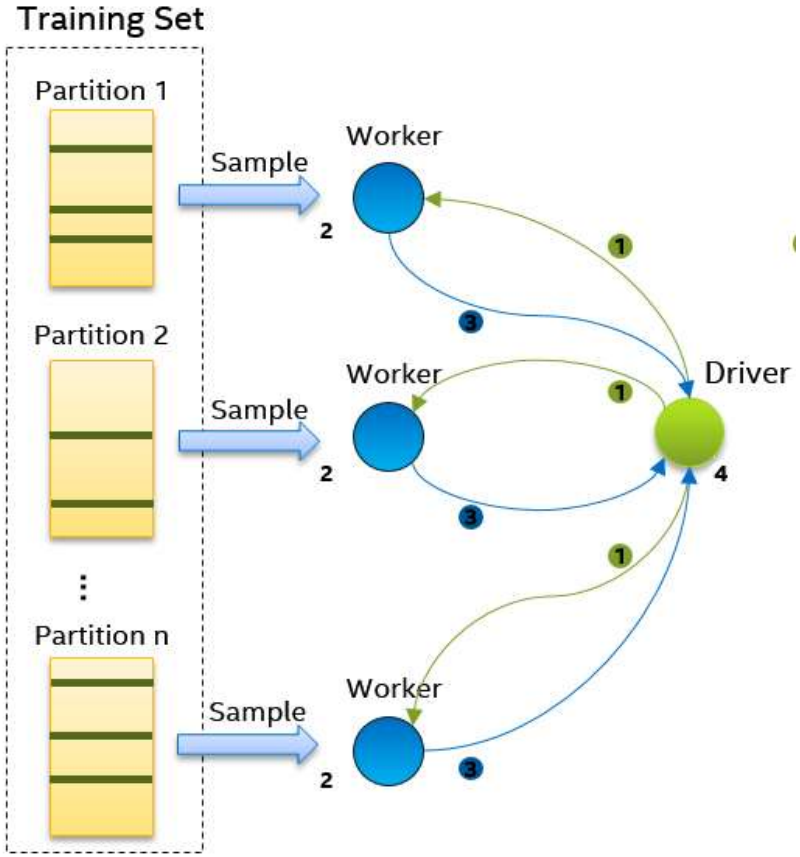


Architecture, Features and Code

"Canonical" implementation on Apache Spark

Driver becomes the bottleneck

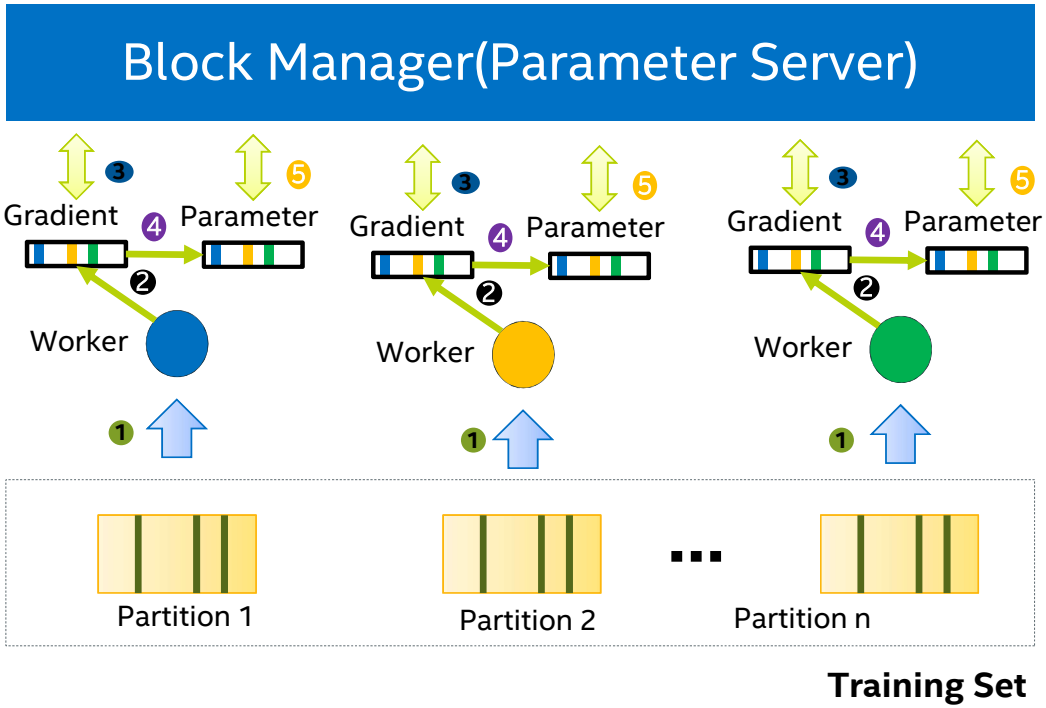
- RDD.reduce / aggregate
- RDD.treeAggregate (shuffle)



Architecture, Features and Code

BigDL distributed training:

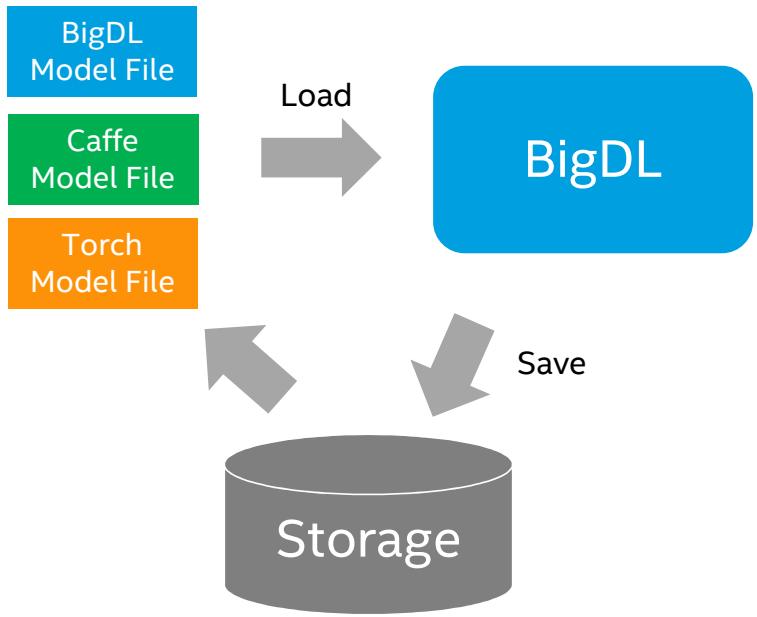
- Implement an P2P All Reduce Algorithm on Apache Spark
- Spark block manager as parameter server (handle different APIs of Spark 1.x/2.x)
- Compress float32 parameter to float16 parameter



Architecture, **Features** and Code

Rich deep learning support

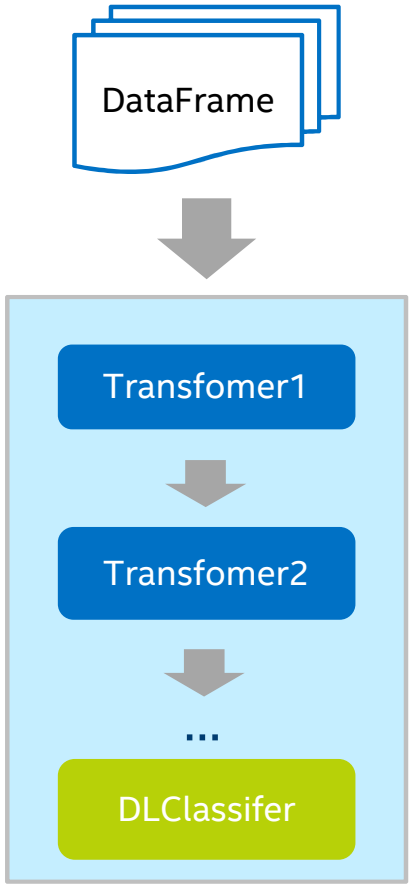
- Feature parity with existing DL frameworks (e.g., Caffe and Torch)
- Load pre-trained Caffe or Torch model into Spark programs
- Model Snapshot
  - Useful in a long training
  - Used in inference later
  - Share your model with others
  - Fine-tune the model



Architecture, **Features** and Code

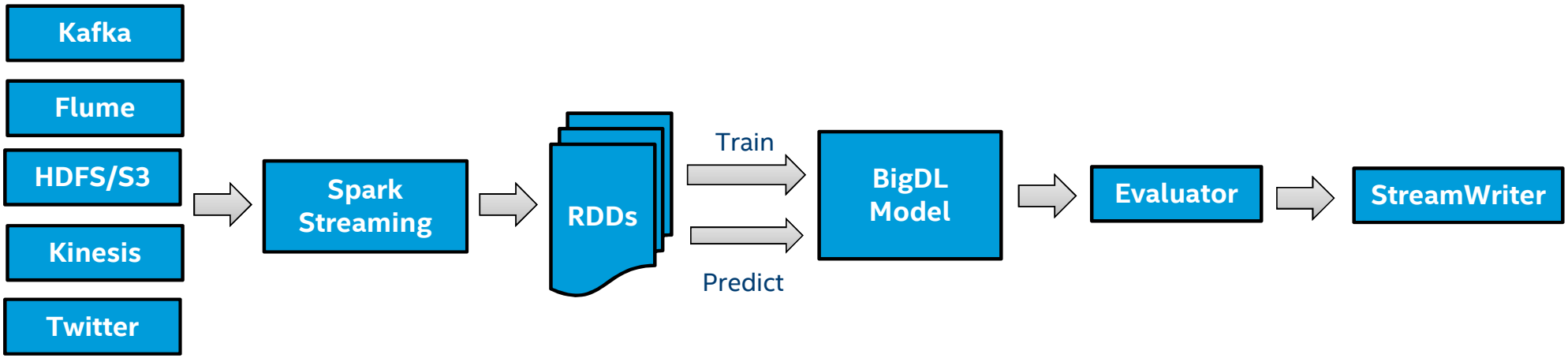
Integrates with Spark-ML Pipeline

- Wrapper with Spark ML Transformer
- Plug into Spark ML pipeline
- Support 1.5/1.6/2.0



Architecture, **Features** and Code

BigDL integrates with Spark Streaming for runtime training and prediction



Architecture, **Features** and Code

Layers

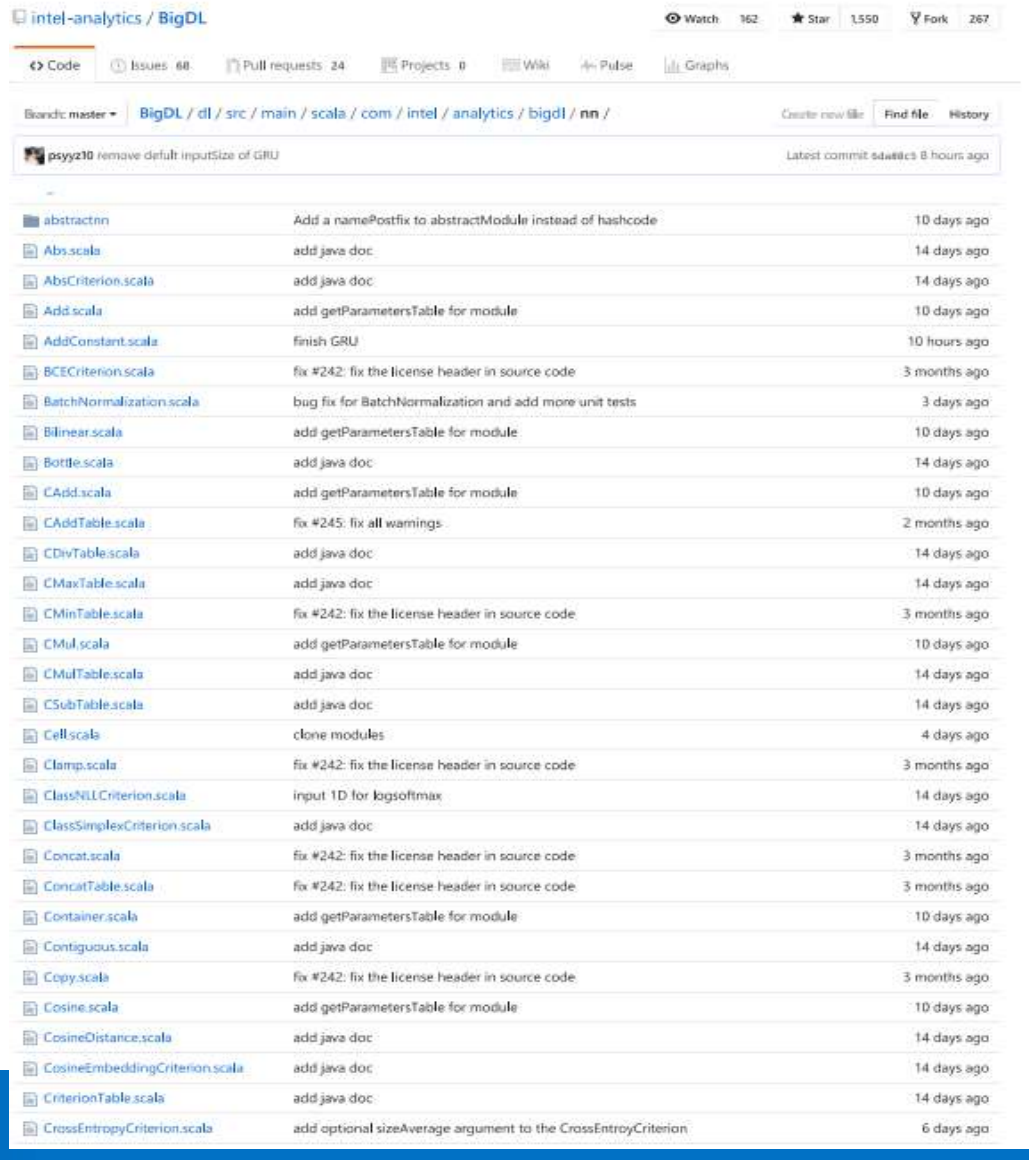
- 90+ Layers (Yiheng)

Criterion

- 10+ loss functions

Optimization

- SGD, Adagrad, LBFGS



Architecture, **Features** and Code

BigDL provide examples to help developer play with BigDL and start with popular models.

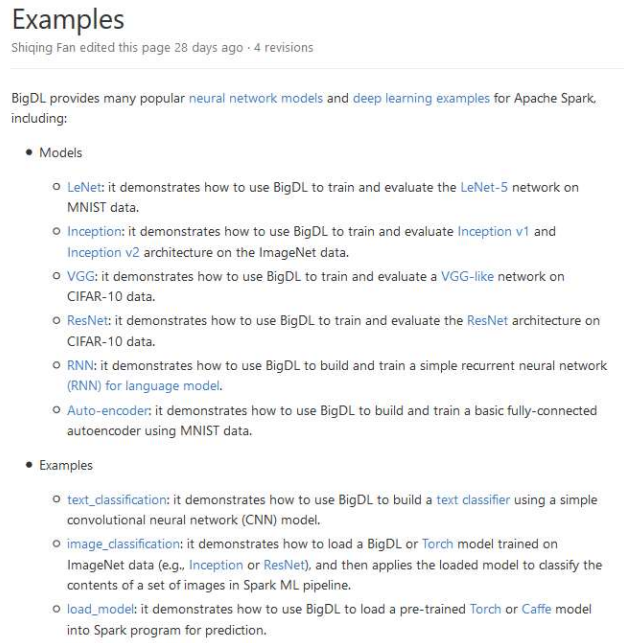
https://github.com/intel-analytics/BigDL/wiki/Examples

Models (Train and Inference Example Code):

- LeNet, Inception, VGG, ResNet, RNN, Auto-encoder

Examples:

- Text Classification
- Image Classification
- Load Torch/Caffe model



Examples

Shiqing Fan edited this page 28 days ago · 4 revisions

BigDL provides many popular neural network models and deep learning examples for Apache Spark, including:

- Models
  - LeNet: it demonstrates how to use BigDL to train and evaluate the LeNet-5 network on MNIST data.
  - Inception: it demonstrates how to use BigDL to train and evaluate Inception v1 and Inception v2 architecture on the ImageNet data.
  - VGG: it demonstrates how to use BigDL to train and evaluate a VGG-like network on CIFAR-10 data.
  - ResNet: it demonstrates how to use BigDL to train and evaluate the ResNet architecture on CIFAR-10 data.
  - RNN: it demonstrates how to use BigDL to build and train a simple recurrent neural network (RNN) for language model.
  - Auto-encoder: it demonstrates how to use BigDL to build and train a basic fully-connected autoencoder using MNIST data.
- Examples
  - text_classification: it demonstrates how to use BigDL to build a text classifier using a simple convolutional neural network (CNN) model.
  - image_classification: it demonstrates how to load a BigDL or Torch model trained on ImageNet data (e.g., Inception or ResNet), and then applies the loaded model to classify the contents of a set of images in Spark ML pipeline.
  - load_model: it demonstrates how to use BigDL to load a pre-trained Torch or Caffe model into Spark program for prediction.

Architecture, **Features** and Code

Model on Data Set	Top-1 Accuracy
LeNet5 on MNIST	99%
Vgg on Cifar10	90%
AlexNet OWT on ImageNet	56%
GoogleNetV1 on ImageNet	68%

Architecture, **Features** and Code

- **High Single Node Performance**
  - Powered by Intel MKL.
  - Benchmarked best on Xeon E5-26XX v3 or E5-26XX v4

- **Efficient Scale-out**
  - Efficiently scale out to several 10s of Xeon servers on Spark to distributed train some deep learning model on ImageNet dataset

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

Architecture, Features and **Code**

BigDL APIs:

Tensor

- Multi-dimensional array of numeric types (e.g., Int, Float, Double, etc.)

Module

- Individual layers of the neural network (such as ReLU, Linear, SpatialConvolution, Sequential, etc.)

Criterion

- Given input and target, computing gradient per given loss function

DataSet

- Training, validation and test data
- *Transformer*: series of data transformations (->)

Engine

- Runtime environment for the training (e.g., node#, core#, spark vs. local, multi-threading, etc.)

Optimizer

- Local & distributed optimizer (synchronous mini-batch SGD)
- *OptimMethod*: SGD, AdaGrad, etc.

Architecture, Features and **Code**

Tensor

- A powerful ndarray data structure
- Generic data type
- Data manipulate / math APIs, model after torch

```
scala> import com.intel.analytics.bigdl.tensor.Tensor
import com.intel.analytics.bigdl.tensor.Tensor

scala> val tensor = Tensor[Float](2, 3)
tensor: com.intel.analytics.bigdl.tensor.Tensor[Float] =
0.0     0.0     0.0
0.0     0.0     0.0
[com.intel.analytics.bigdl.tensor.DenseTensor of size 2x3]
```

Architecture, Features and **Code**

```
scala> import com.intel.analytics.bigdl.numeric.NumericFloat // import global float tensor
import com.intel.analytics.bigdl.numeric.NumericFloat

scala> import com.intel.analytics.bigdl.nn._
import com.intel.analytics.bigdl.nn._

scala> val f = Linear(3,4) // create the module
mlp: com.intel.analytics.bigdl.nn.Linear[Float] = nn.Linear(3 -> 4)

// let's see what f's parameters were initialized to. ('nn' always inits to something reasor
scala> f.weight
res5: com.intel.analytics.bigdl.tensor.Tensor[Float] =
-0.008662592    0.543819        -0.028795477
-0.30469555     -0.3909278      -0.10871882
0.114964925     0.1411745       0.35646403
-0.16590376     -0.19962183     -0.18782845
[com.intel.analytics.bigdl.tensor.DenseTensor of size 4x3]
```

Architecture, Features and **Code**

Build a simple model

```
scala> val g = Sum()
g: com.intel.analytics.bigdl.nn.Sum[Float] = nn.Sum

scala> val mlp = Sequential().add(f).add(g)
mlp: com.intel.analytics.bigdl.nn.Sequential[Float] =
nn.Sequential {
  [input -> (1) -> (2) -> output]
  (1): nn.Linear(3 -> 4)
  (2): nn.Sum
}
```

Architecture, Features and **Code**

A full example

```
val model = Sequential()
  .add(SpatialConvolution(3, 64, 11, 11, 4, 4, 2, 2, 1))
  .add(ReLU(true))
  .add(SpatialMaxPooling(3, 3, 2, 2))
  .add(SpatialConvolution(64, 192, 5, 5, 1, 1, 2, 2))
  .add(ReLU(true))
  .add(SpatialMaxPooling(3, 3, 2, 2))
  .add(SpatialConvolution(192, 384, 3, 3, 1, 1, 1, 1))
  .add(ReLU(true))
  .add(SpatialConvolution(384, 256, 3, 3, 1, 1, 1, 1))
  .add(ReLU(true))
  .add(SpatialConvolution(256, 256, 3, 3, 1, 1, 1, 1))
  .add(ReLU(true))
  .add(SpatialMaxPooling(3, 3, 2, 2))
  .add(View(256 * 6 * 6))
  .add(Linear(256 * 6 * 6, 4096))
  .add(ReLU(true))
  .add(Dropout(0.5))
  .add(Linear(4096, 4096))
  .add(ReLU(true))
  .add(Dropout(0.5))
  .add(Linear(4096, 1000))
  .add(LogSoftMax())
```

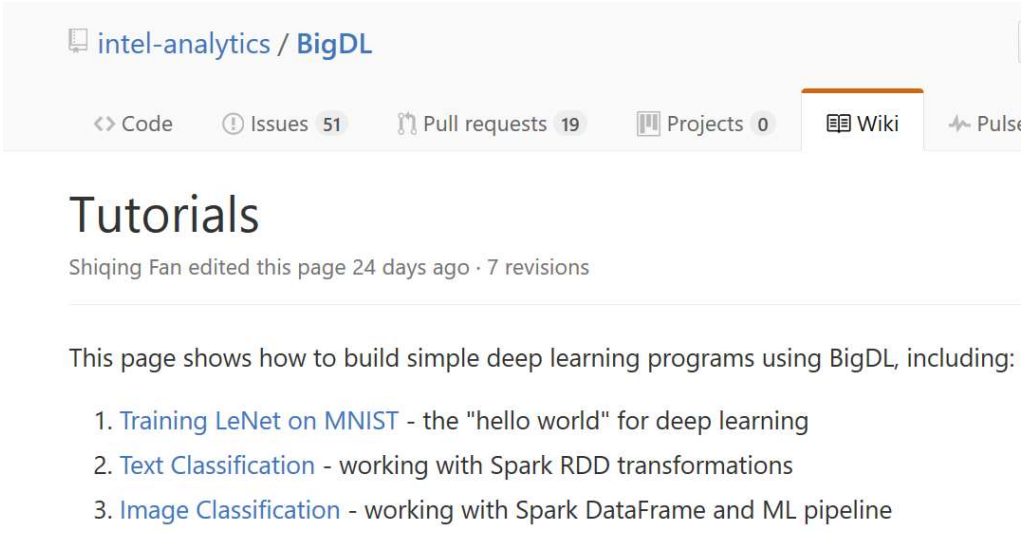
```
val optimizer = Optimizer(
  model = model,
  dataset = trainSet,
  criterion = new ClassNLLCriterion[Float]()
)
```

```
optimizer
  .setState(state)
  .setValidation(Trigger.severalIteration(620),
   valSet, Array(new Top1Accuracy[Float], new Top5Accuracy[Float]))
  .setEndWhen(Trigger.maxIteration(62000))
  .optimize()
```

Architecture, Features and **Code**

Start with tutorials

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



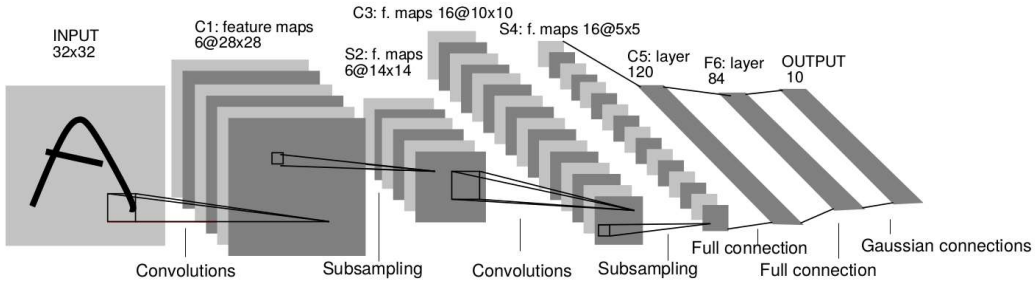
Architecture, Features and **Code**

Training LeNet on MNIST

"Hello World" for Deep learning

MNIST dataset (images of handwritten digits)

http://yann.lecun.com/exdb/mnist/



LeNet-5

http://yann.lecun.com/exdb/lenet/

Architecture, Features and **Code**

"*Hello World*" for Deep learning

```
 2* * Licensed to Intel Corporation under one or more
17
18 package com.intel.analytics.bigdl.models.lenet
19
20 import java.nio.file.Paths
21
22 import com.intel.analytics.bigdl._
23 import com.intel.analytics.bigdl.dataset.DataSet
24 import com.intel.analytics.bigdl.dataset.image.{GreyImgNormalizer, GreyImgToBatch, BytesToGreyImg}
25 import com.intel.analytics.bigdl.nn.{ClassNLLCriterion, Module}
26 import com.intel.analytics.bigdl.numeric.NumericFloat
27 import com.intel.analytics.bigdl.optim._
28 import com.intel.analytics.bigdl.utils.{Engine, T}
29 import org.apache.log4j.{Level, Logger}
30 import org.apache.spark.SparkContext
31
32
33 object Train {
34   Logger.getLogger("org").setLevel(Level.ERROR)
35   Logger.getLogger("akka").setLevel(Level.ERROR)
36   Logger.getLogger("breeze").setLevel(Level.ERROR)
37   Logger.getLogger("com.intel.analytics.bigdl.optim").setLevel(Level.INFO)
```

Architecture, Features and **Code**

"*Hello World*" for Deep learning

- *sc* is ***None*** for local mode;
- Program will run as either local program or Spark program based on *sc* value.

```
40
41  def main(args: Array[String]): Unit = {
42    trainParser.parse(args, new TrainParams()).map(param => {
43      val sc = Engine.init(param.nodeNumber, param.coreNumber, param.env == "spark").map(conf => {
44        conf.setAppName("Train Lenet on MNIST")
45          .set("spark.akka.frameSize", 64.toString)
46          .set("spark.task.maxFailures", "1")
47        new SparkContext(conf)
48      })
49
50      val trainData = Paths.get(param.folder, "/train-images-idx3-ubyte")
51      val trainLabel = Paths.get(param.folder, "/train-labels-idx1-ubyte")
52      val validationData = Paths.get(param.folder, "/t10k-images-idx3-ubyte")
53      val validationLabel = Paths.get(param.folder, "/t10k-labels-idx1-ubyte")
54
55      val model = if (param.modelSnapshot.isDefined) {
56        Module.load[Float](param.modelSnapshot.get)
57      } else {
58        LeNet5(classNum = 10)
59      }
```

Architecture, Features and **Code**

"*Hello World*" for Deep learning

```
40
41  def main(args: Array[String]): Unit = {
42    trainParser.parse(args, new TrainParams()).map(param => {
43      val sc = Engine.init(param.nodeNumber, param.coreNumber, param.env == "spa
44        conf.setAppName("Train Lenet on MNIST")
45          .set("spark.akka.frameSize", 64.toString)
46          .set("spark.task.maxFailures", "1")
47        new SparkContext(conf)
48      })
49
50      val trainData = Paths.get(param.folder, "/train-images-idx3-ubyte")
51      val trainLabel = Paths.get(param.folder, "/train-labels-idx1-ubyte")
52      val validationData = Paths.get(param.folder, "/t10k-images-idx3-ubyte")
53      val validationLabel = Paths.get(param.folder, "/t10k-labels-idx1-ubyte")
54
55      val model = if (param.modelSnapshot.isDefined) {
56        Module.load[Float](param.modelSnapshot.get)
57      } else {
58        LeNet5(classNum = 10)
59      }
60
61      val state = if (param.stateSnapshot.isDefined) {
62        T.load(param.stateSnapshot.get)
63      } else {
```

```
val model = Sequential()
model.add(Reshape(Array(1, 28, 28)))
  .add(SpatialConvolution(1, 6, 5, 5))
  .add(Tanh())
  .add(SpatialMaxPooling(2, 2, 2, 2))
  .add(Tanh())
  .add(SpatialConvolution(6, 12, 5, 5))
  .add(SpatialMaxPooling(2, 2, 2, 2))
  .add(Reshape(Array(12 * 4 * 4)))
  .add(Linear(12 * 4 * 4, 100))
  .add(Tanh())
  .add(Linear(100, classNum))
  .add(LogSoftMax())
```

Architecture, Features and **Code**

"*Hello World*" for Deep learning

```
60
61      val state = if (param.stateSnapshot.isDefined) {
62        T.load(param.stateSnapshot.get)
63      } else {
64        T(
65          "learningRate" -> param.learningRate
66        )
67      }
68
69      val trainSet = (if (sc.isDefined) {
70        DataSet.array(load(trainData, trainLabel), sc.get)
71      } else {
72        DataSet.array(load(trainData, trainLabel))
73      }) -> BytesToGreyImg(28, 28) -> GreyImgNormalizer(trainMean, trainStd) -> GreyImgToBatch(
74        param.batchSize)
75
76      val optimizer = Optimizer(
77        model = model,
78        dataset = trainSet,
79        criterion = ClassNLLCriterion[Float]())
80      if (param.checkpoint.isDefined) {
81        optimizer.setCheckpoint(param.checkpoint.get, Trigger.everyEpoch)
82      }
```

Architecture, Features and **Code**

"*Hello World*" for Deep learning

```
60
61      val state = if (param.stateSnapshot.isDefined) {
62        T.load(param.stateSnapshot.get)
63      } else {
64        T(
65          "learningRate" -> param.learningRate
66        )
67      }
68
69      val trainSet = (if (sc.isDefined) {
70        DataSet.array(load(trainData, trainLabel), sc.get)
71      } else {
72        DataSet.array(load(trainData, trainLabel))
73      }) -> BytesToGreyImg(28, 28) -> GreyImgNormalizer(trainMean, trainStd) -> GreyImgToBatch(
74        param.batchSize)
75
76      val optimizer = Optimizer(
77        model = model,
78        dataset = trainSet,
79        criterion = ClassNLLCriterion[Float]())
80      if (param.checkpoint.isDefined) {
81        optimizer.setCheckpoint(param.checkpoint.get, Trigger.everyEpoch)
82      }
83
```

Architecture, Features and **Code**

"*Hello World*" for Deep learning

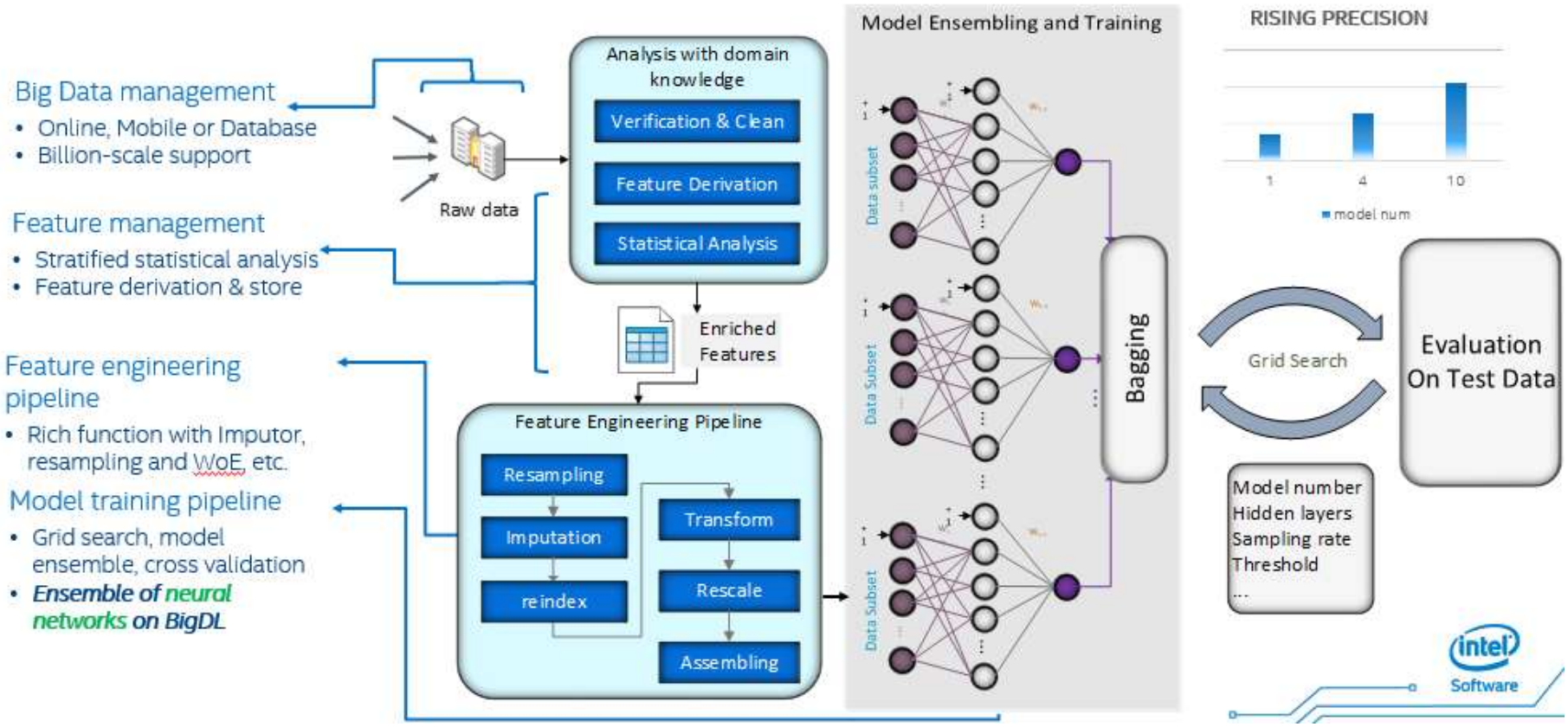
```
83
84        val validationSet = (if (sc.isDefined) {
85          DataSet.array(load(validationData, validationLabel), sc.get)
86        } else {
87          DataSet.array(load(validationData, validationLabel))
88        }) -> BytesToGreyImg(28, 28) -> GreyImgNormalizer(testMean, testStd) -> GreyImgToBatch(
89          param.batchSize)
90
91        optimizer
92          .setValidation(
93            trigger = Trigger.everyEpoch,
94            dataset = validationSet,
95            vMethods = Array(new Top1Accuracy))
96          .setState(state)
97          .setEndWhen(Trigger.maxEpoch(param.maxEpoch))
98          .optimize()
99      })
100   }
101 }
```

WHERE:
Use Cases

WHERE CAN I USE BIGDL?

- Analyze “big data” using deep learning on the same Hadoop/Spark cluster where the data are stored
- Add deep learning functionalities to the Big Data (Spark) programs and/or workflow
- Leverage existing Hadoop/Spark clusters to run deep learning applications
  - Dynamically sharing with other workloads (e.g., ETL, data warehouse, feature engineering, classical machine learning, graph analytics, etc.)
- Making deep learning more accessible for ***Big Data users*** and ***Data Scientists***.

FRAUD DETECTION Using BigDL and SparkML



Product Defect Detection and Classification

Big Data management

- High resolution images
- Large volume of data

Proposal Extraction

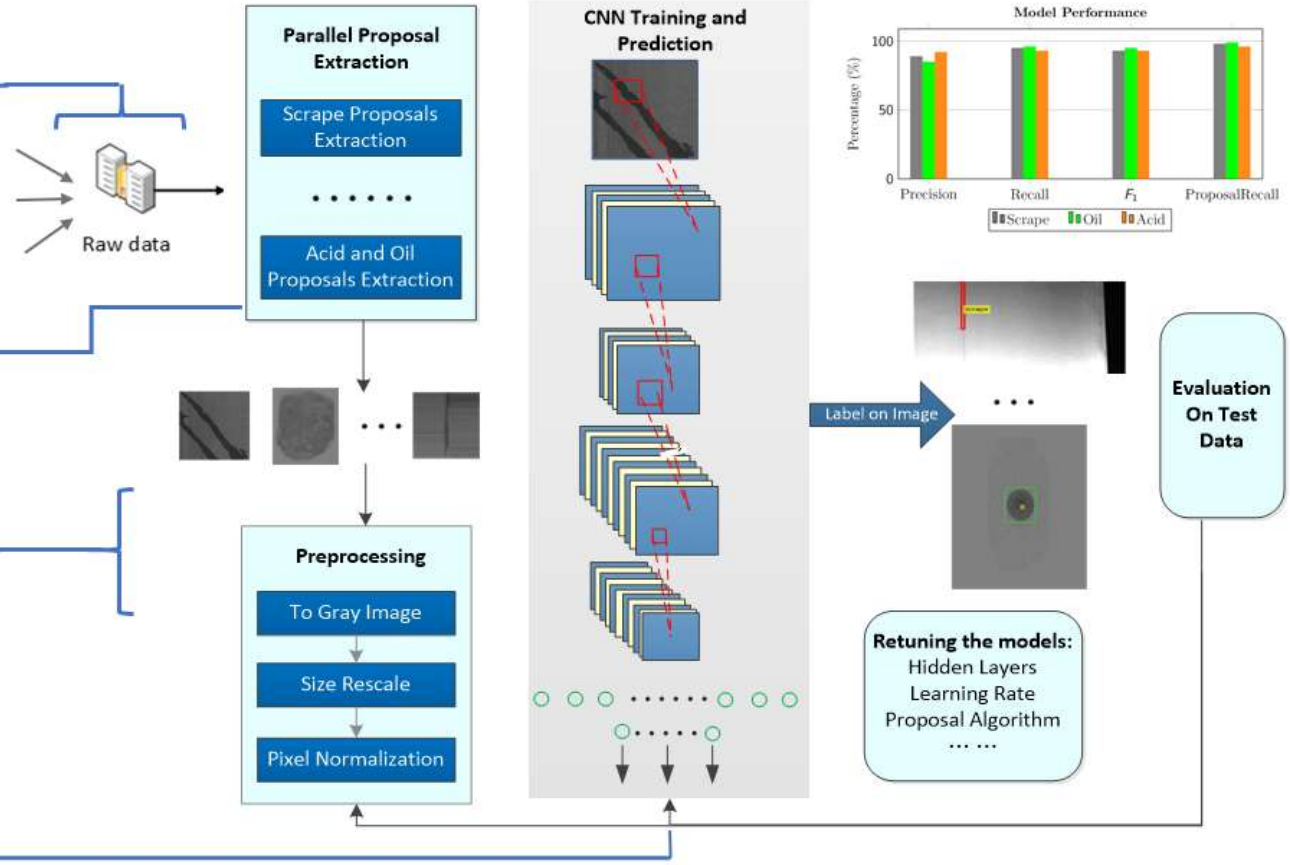
- Extract proposals for each defect
- Parallel pipeline **(KeyStone ML Pipeline)**
- Running **on Spark**

Preprocessing

- Preprocess the proposals before model training or testing

Model training pipeline

- Train ***Convolutional Neural Networks*** **on Spark**
- Parameter tuning, optimize proposal extraction algorithms

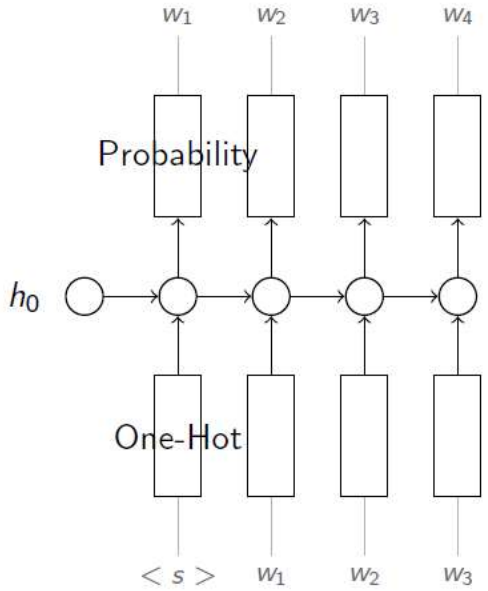


Language Model with RNN

- Sentence Tokenizer
- Dictionary Building
- Input Document Transformer

Generated sentences with regard to trigger words.

RNN Model




```
1   /.../
17
18  package com.intel.analytics.bigdl.models.rnn
19
20  import ...
22
23  object SimpleRNN {
24    def apply(
25    inputSize: Int,
26    hiddenSize: Int,
27    outputSize: Int,
28    bpttTruncate: Int = 4)
29    : Module[Float] = {
30      val model = Sequential[Float]()
31      model.add(Recurrent[Float](hiddenSize, bpttTruncate)
32        .add(RnnCell[Float](inputSize, hiddenSize))
33        .add(Tanh[Float]()))
34        .add(Linear[Float](hiddenSize, outputSize))
35      model
36    }
37  }
38
```

See the code at:
https://github.com/intel-analytics/BigDL/tree/master/dl/src/main/scala/com/intel/analytics/bigdl/models/rnn

Learn from Shakespeare Poems

Output of RNN:

Long live the King . The King and Queen , and the Strange of the Veils of the rhapsodic . and grapple, and the entreatments of the pressure .

Upon her head , and in the world ? `` Oh, the gods ! O Jove ! To whom the king : `` O friends !

Her hair, nor loose ! If , my lord , and the groundlings of the skies . jocund and Tasso in the Staggering of the Mankind . and

Fine-tune Caffe/Torch Model on Spark

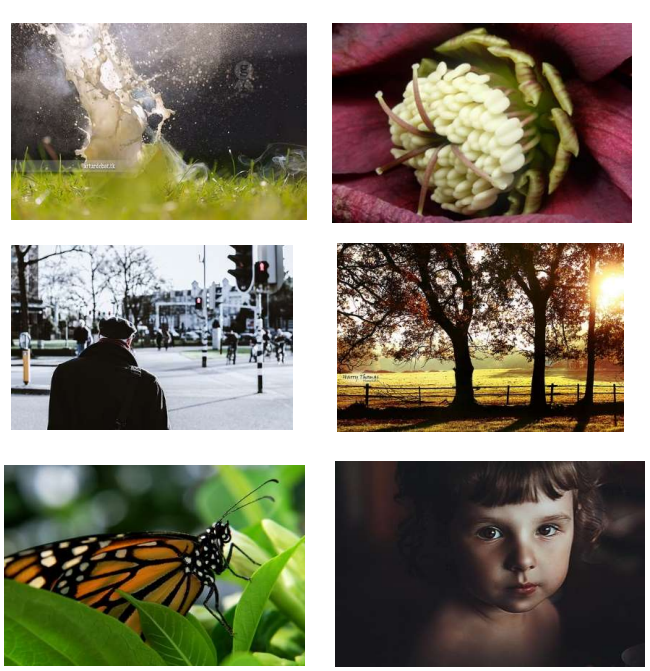
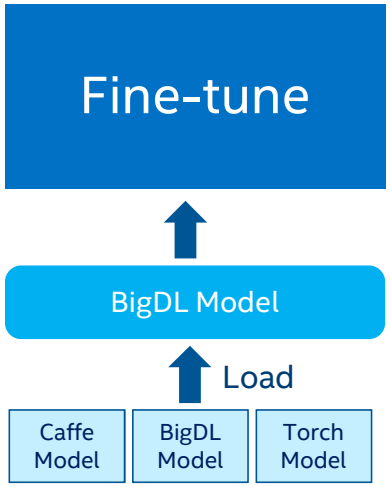
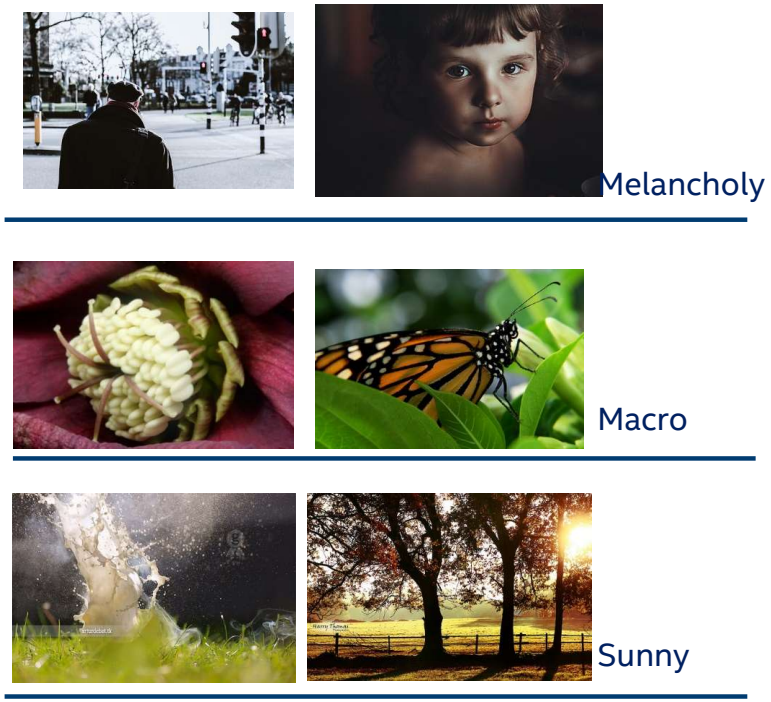


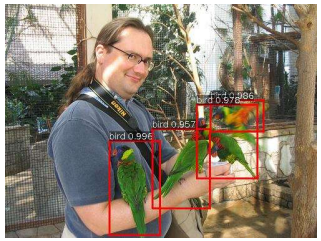
Image source: https://www.flickr.com/photos/

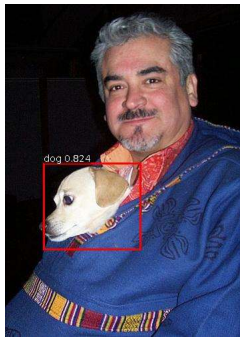


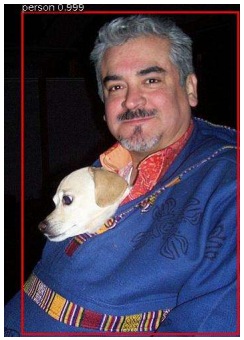


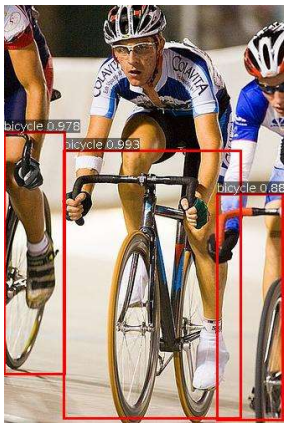
- Train on different dataset based on pre-trained model
- Predict image style instead of type
- Save training time and improve accuracy

Object Detection on PASCAL(http://host.robots.ox.ac.uk/pascal/VOC/)



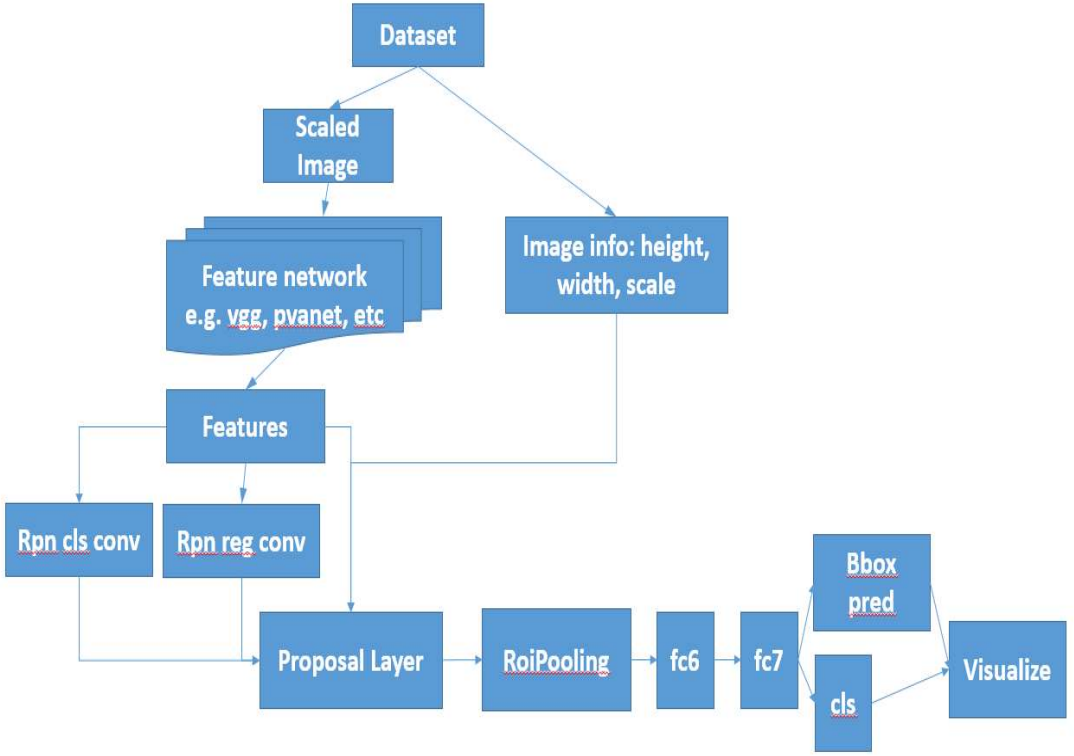






Fast-RCNN

- Faster-RCNN is a popular object detection framework
- It share the features between detection network and region proposal network



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems*. 2015.

Object Detection with Fast-RCNN

```
val classes = Array[String](
  "__background__", "aeroplane", "bicycle", "bird", "boat", "bottle", "bus",
  "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike",
  "person", "pottedplant", "sheep", "sofa", "train", "tvmonitor"
)

val net = FasterRcnn(params.modelType,
  caffeModel = Some((params.caffeDefPath, params.caffeModelPath)))

val dataSet = Preprocessor(params.imageFolder, net.param, sc)

val validator = FrcnnValidator(net, classes, dataSet = dataSet)
val output = validator.test()
output.foreach(x => {
  (1 until classes.length).foreach(cls => {
    VisualizeTool.visDetection(x._4, classes(cls), x._1(cls),
      thresh = 0.8f, outPath = params.outputFolder)
  })
})
```

HOW:
Get Started with BigDL

Get Started with BigDL

- BigDL programs
  - Run as either local Scala/Java programs, or Spark programs

- Interactive Scala shell (REPL) is a great way to get started

Type:

```
$ source PATH_To_BigDL/scripts/bigdl.sh
$ scala -cp bigdl-0.1.0-SNAPSHOT-jar-with-dependencies-and-spark.jar
```

Scala shell:

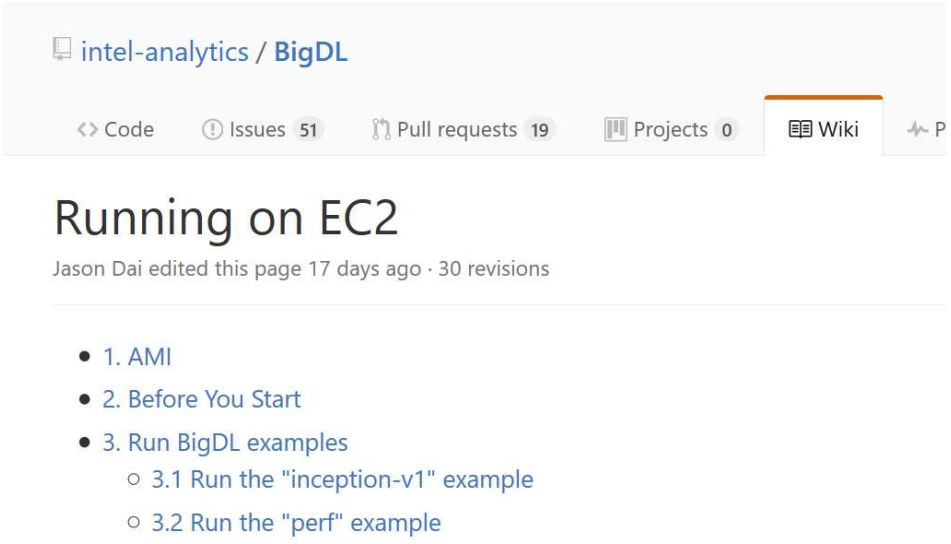
```
Welcome to Scala 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_91).
Type in expressions for evaluation. Or try :help.

scala>
```

Get Started with BigDL

BigDL Out-of-box run scripts on AWS

https://github.com/intel-analytics/BigDL/wiki/Running-on-EC2



BigDL On Github

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

Community

Mail List

bigdl-user-group+subscribe@googlegroups.com

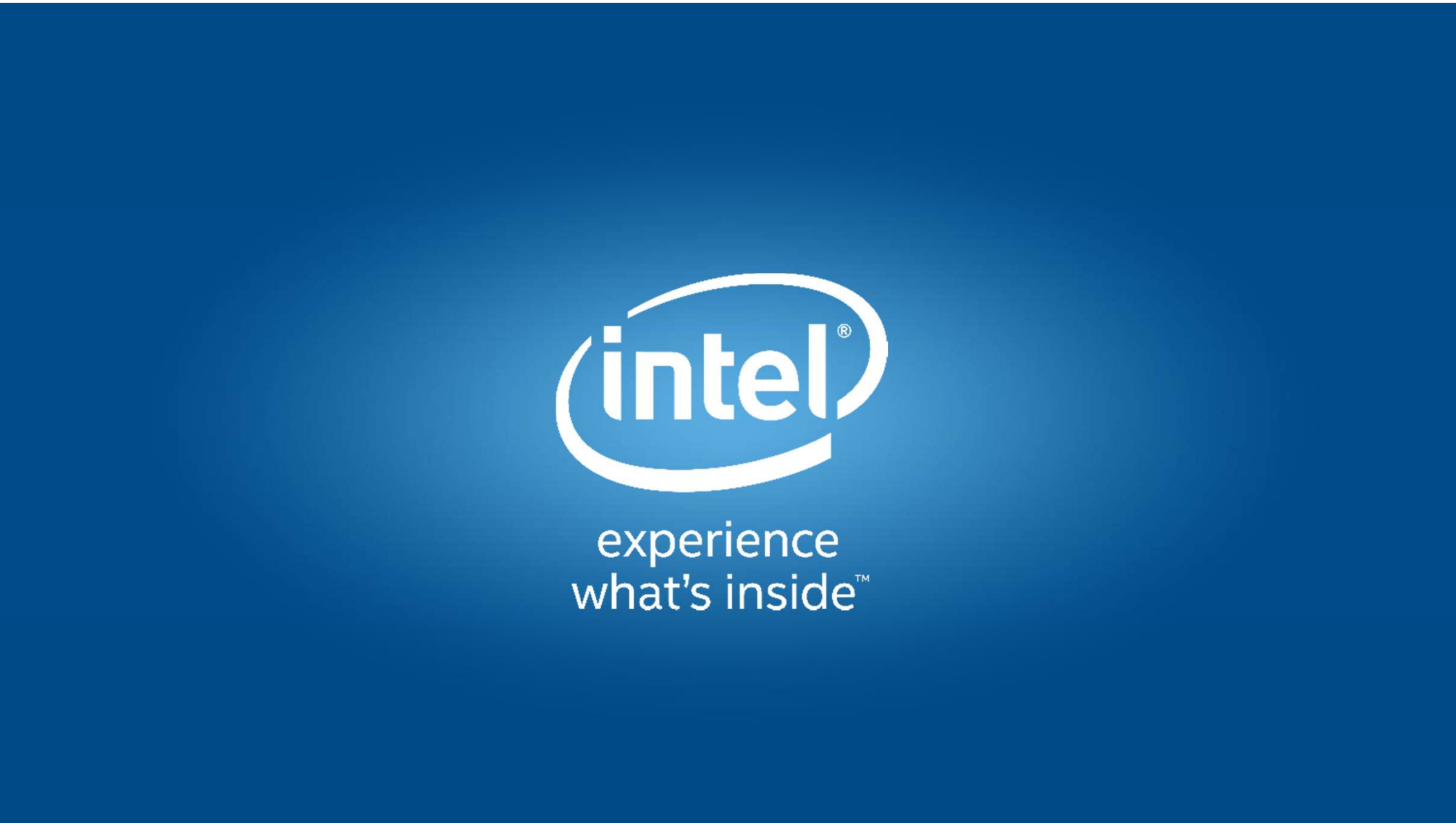
Report bugs and feature request

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

Key Contacts & Additional Resources

- Email: Jason.dai@intel.com (Chief Architect, BigDL)
- O'reilly Podcast/ Jason Dai
  - https://soundcloud.com/oreilly-radar/deep-learning-for-apache-spark
- Databricks Blog
  - https://databricks.com/blog/2017/02/09/intels-bigdl-databricks.html
- Spark Summit Talks
  - https://spark-summit.org/east-2017/events/bigdl-a-distributed-deep-learning-library-on-spark/
  - https://spark-summit.org/east-2017/events/building-deep-learning-powered-big-data/
- Intel Developer Zone for AI
  - Video: https://software.intel.com/en-us/videos/bigdl-distributed-deep-learning-on-apache-spark
  - Blog: https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark

Q & A


intel®
experience
what's inside™