Using Crowdsourced Images to Create Image Recognition Models with Analytics Zoo using BigDL

Maurice Nsabimana,
World Bank Development Data Group

Jiao(Jennie) Wang,
Big Data Technologies, Intel Corporation

#DL8SAIS

Agenda

- Use Case Overview
- Solution Architecture
- Model Development and Results
- Next Steps
- Summary

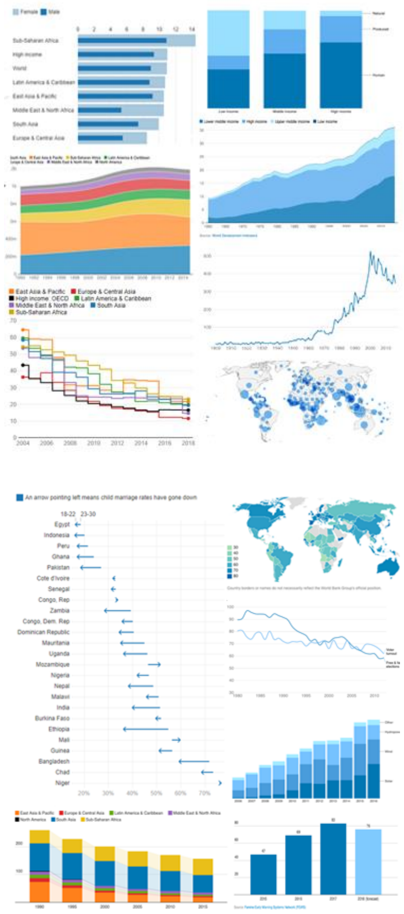
Use Case Overview

World Bank

The World Bank is a vital source of financial and technical assistance to developing countries around the world. It is not a bank in the ordinary sense but a unique partnership to reduce poverty and support development. The World Bank Group comprises five institutions managed by their member countries.

Established in 1944, the **World Bank Group** is headquartered in Washington, D.C. It has more than 10,000 employees in more than 120 offices worldwide.

The Development Data Group provides high-quality national and international statistics to clients within and outside the World Bank and to improve the capacity of member countries to produce and use statistical information.



Problem Statement

- The International Comparison Program team in the World Bank Development Data Group collected crowdsourced images for a pilot data collection study through a privately-operated network of paid on-the-ground contributors that had access to a smartphone and a data collection application designed for the pilot.
- Nearly 3 million labeled images were collected as ground truth/metadata attached to each price observation of 162 tightly specified items for a variety of household goods and services. The use of common item specifications aimed at ensuring the quality, as well as intra- and inter-country comparability, of the collected data.
- **Goal** is to reduce labor intensive tasks of manually moderating (reviewing, searching and sorting) the crowd-sourced images before their release as a public image dataset that could be used to train various deep learning models.

Our Challenges

- Crowdsourced images are of different quality (resolution, close-up, blur, etc.)
- Crowdsourced images are not very focused.(Multiple items, item is small, etc.)
- Some images are possibly corrupted (cannot be resized or transformed)
- Images sourced from 15 different countries – different language groups represented in the text example of images
- Some text is typed, some text is handwritten

Classifying Real Food Images is not a Cat vs Dog Problem

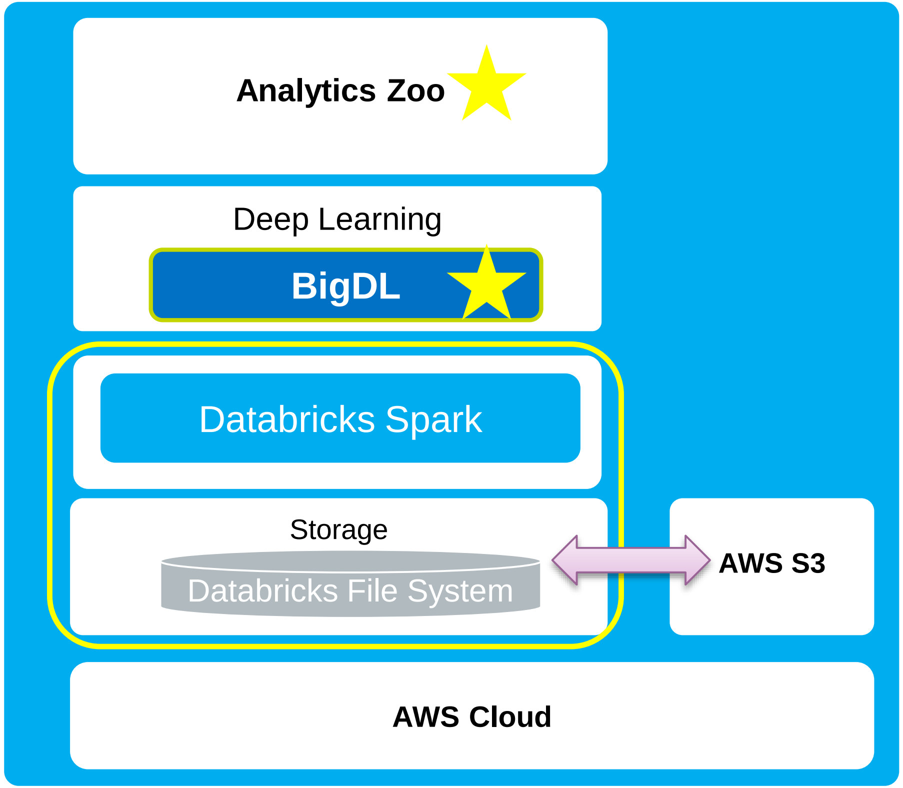
Project Layout

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

Solution Architecture

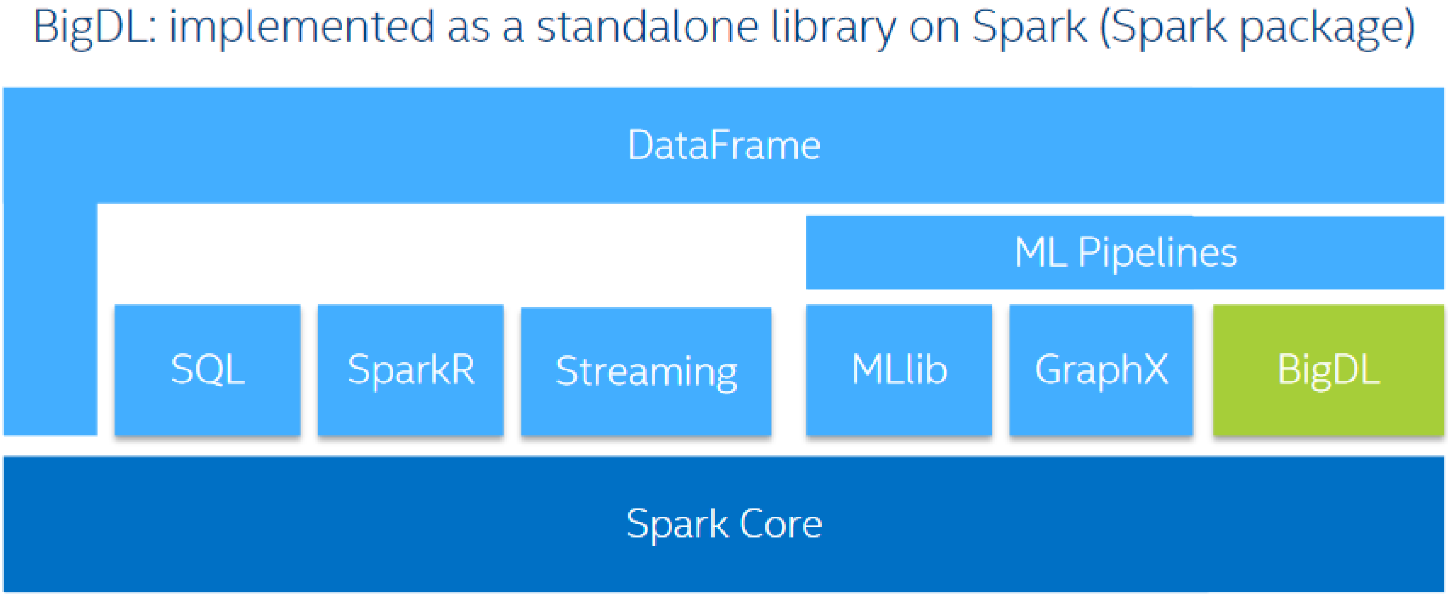
Solution Architecture



- BigDL 0.5
- Databricks Spark
- AWS S3
- AWS R4 instance

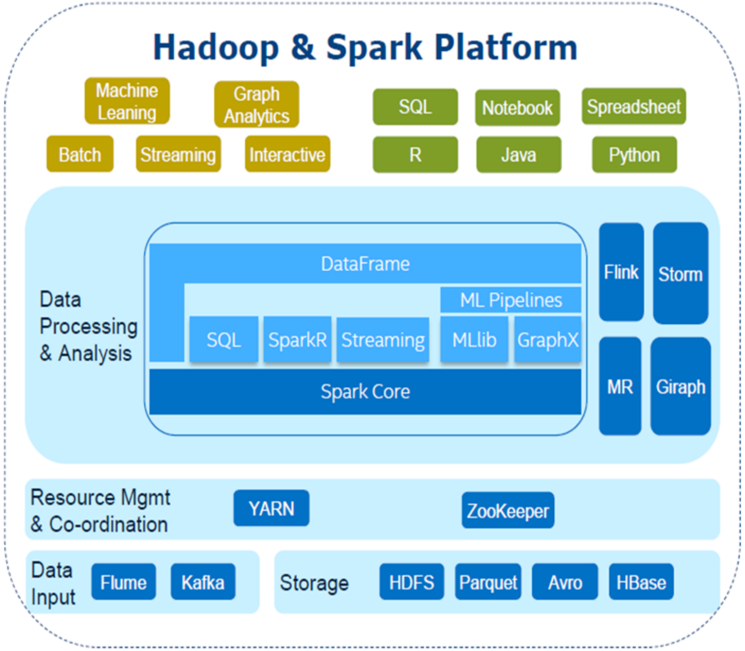
What is BigDL?

BigDL is a distributed deep learning library for Apache Spark*



BigDL is Designed for Big Data

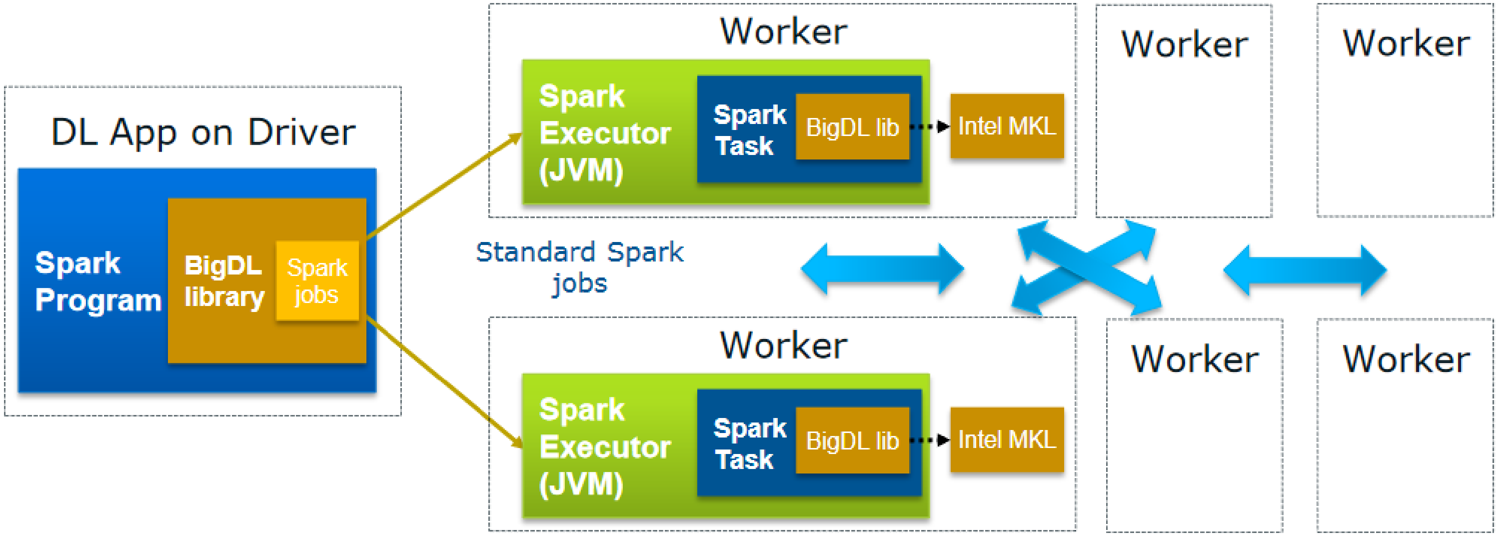
- Distributed deep learning framework for Apache Spark
- Make deep learning more accessible to big data users and data scientists
  - Write deep learning applications as ***standard Spark programs***
  - Run on existing Spark/Hadoop clusters (***no changes needed***)
- Feature parity with popular deep learning frameworks
  - E.g., Caffe, Torch, Tensorflow, etc.
- High performance
  - Powered by Intel MKL and multi-threaded programming
- Efficient scale-out
  - Leveraging Spark for distributed training & inference



BigDL as a Standard Spark Program

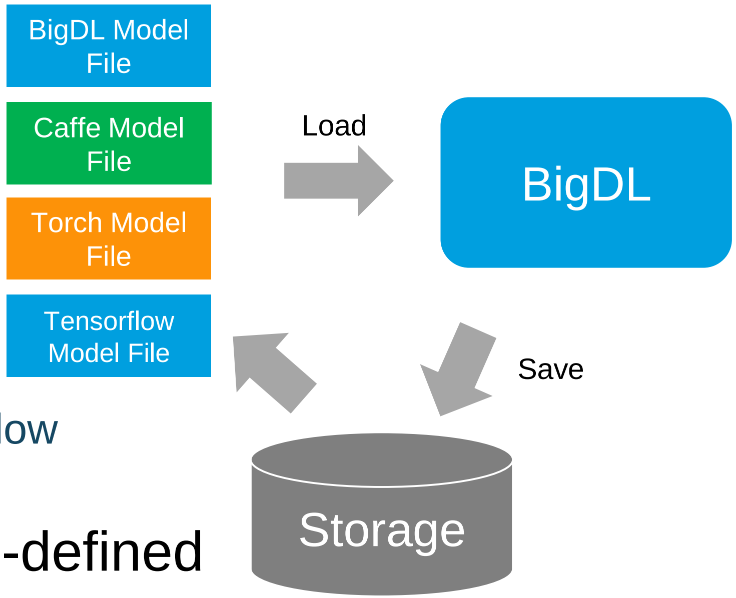
Distributed Deep learning applications (training, fine-tuning & prediction) on Apache Spark*

- No changes to the existing Hadoop/Spark clusters needed



Models Interoperability Support

- Model Snapshot
  - Long training work checkpoint
  - Model deployment and sharing
  - Fine-tune
- Caffe/Torch/Tensorflow Model Support
  - Model file load
  - Easy to migrate your Caffe/Torch/Tensorflow work to Spark
- NEW - BigDL supports loading pre-defined Keras models (Keras 1.2.2)



BigDL: Python API

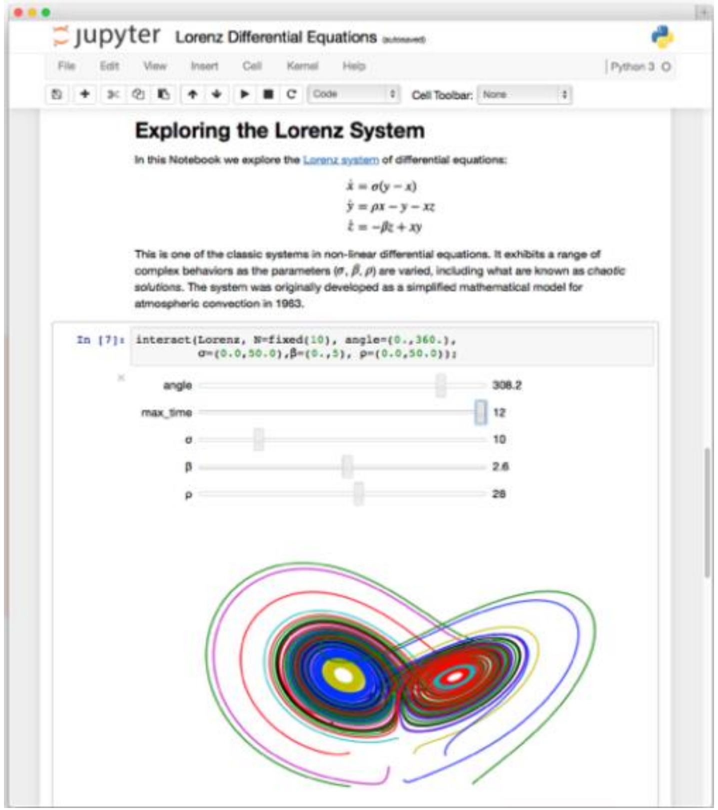
- Support deep learning model training, evaluation, inference
- Support Spark v1.5/1.6/2.X
- Support **Python 2.7/3.5/3.6**
- Based on PySpark, **Python API** in BigDL allows use of existing Python libs (Numpy, Scipy, Pandas, Scikit-learn, NLTK, Matplotlib, etc)

```
train_data = get_minst("train").map(
    normalizer(mnist.TRAIN_MEAN, mnist.TRAIN_STD))
test_data = get_minst("test").map(
    normalizer(mnist.TEST_MEAN, mnist.TEST_STD))
state = {"batchSize": int(options.batchSize),
         "learningRate": 0.01,
         "learningRateDecay": 0.0002}
optimizer = Optimizer(
    model=build_model(10),
    training_rdd=train_data,
    criterion=ClassNLLCriterion(),
    optim_method="SGD",
    state=state,
    end_trigger=MaxEpoch(100))
optimizer.setvalidation(
    batch_size=32,
    val_rdd=test_data,
    trigger=EveryEpoch(),
    val_method=["top1"]
)
optimizer.setcheckpoint(EveryEpoch(), "/tmp/lenet5/")
trained_model = optimizer.optimize()
```

Works with Interactive Notebooks

Running BigDL applications directly in Jupyter, Zeppelin, Databricks notebooks, etc.

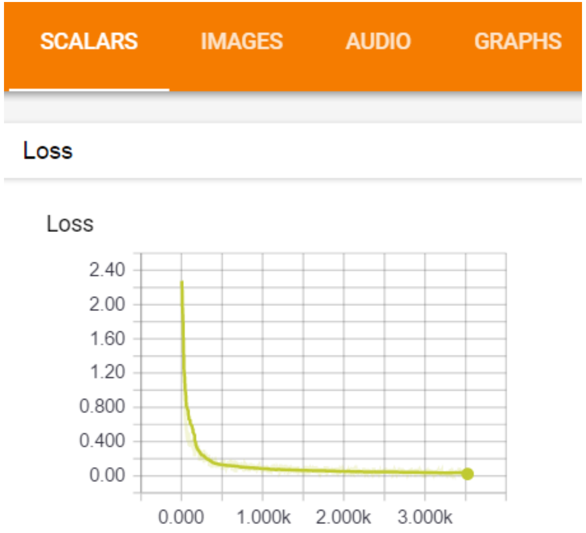
- ✓ Share and Reproduce
  - Notebooks can be shared with others
  - Easy to reproduce and track
- ✓ Rich Content
  - Texts, images, videos, LaTeX and JavaScript
  - Code can also produce rich contents
- ✓ Rich toolbox
  - Apache Spark, from Python, R and Scala
  - Pandas, scikit-learn, ggplot2, dplyr, etc

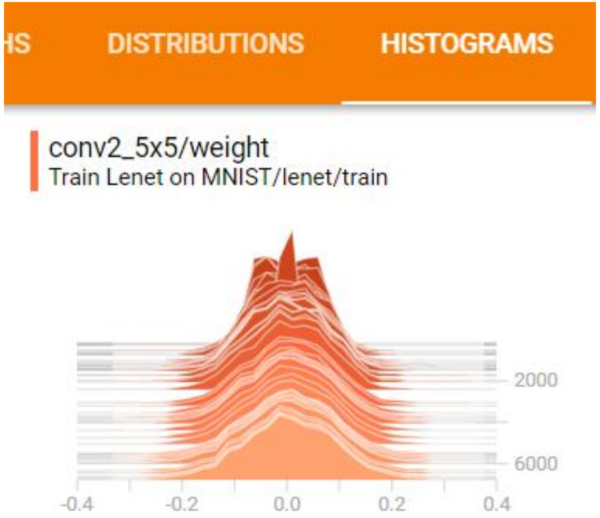


Visualization for Learning

BigDL integration with TensorBoard

- TensorBoard is a suite of web applications from Google for visualizing and understanding deep learning applications





Analytics Zoo

Analytics + AI Pipelines for Spark and BigDL

“Out-of-the-box” ready for use

- Reference use cases
  - Fraud detection, time series prediction, sentiment analysis, chatbot, etc.
- Predefined models
  - Object detection, image classification, text classification, recommendations, etc.
- Feature transformations
  - Vision, text, 3D imaging, etc.
- High level APIs
  - DataFrames, ML Pipelines, Keras/Keras2, etc.

Model Development & Results

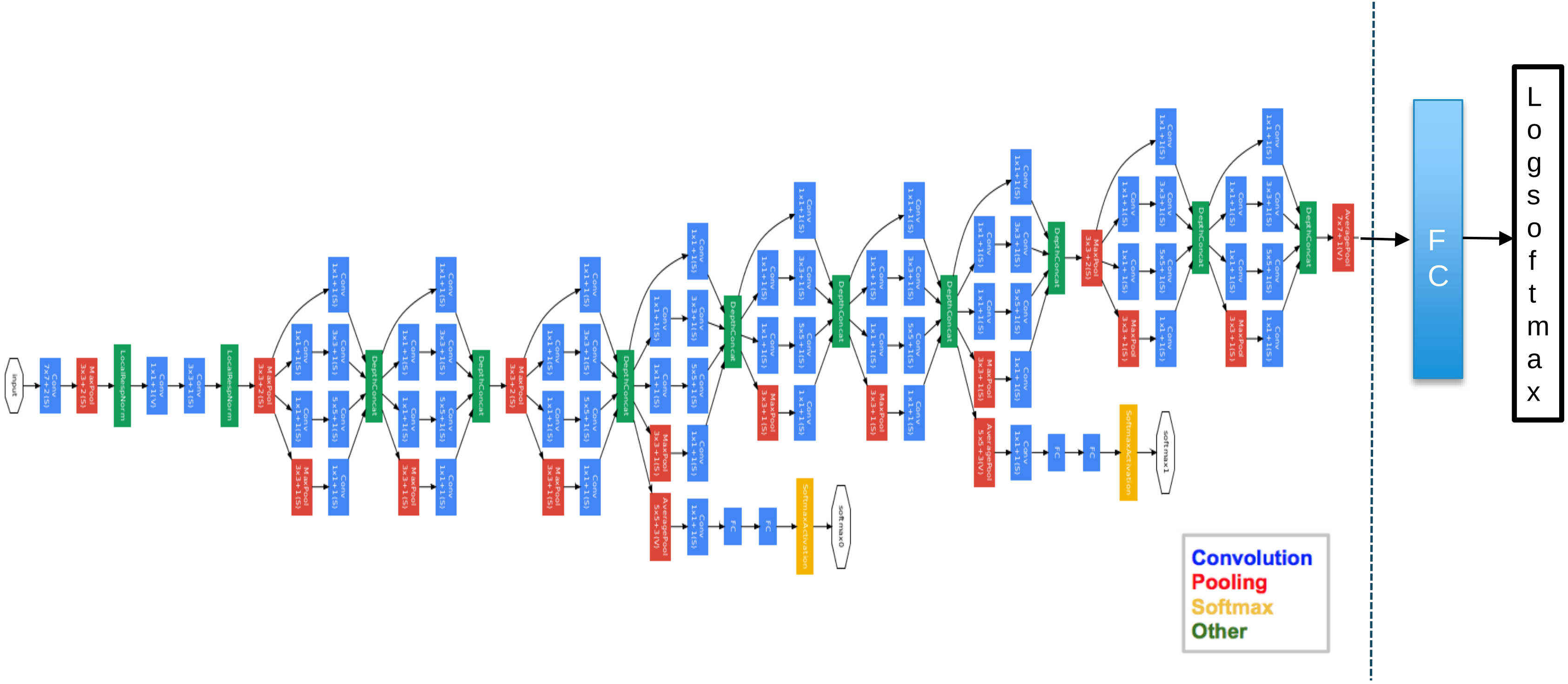
Model Development - Phase 1

- Transfer learning/Fine tuning from pre-trained Inception model to do classification
- Load pre-trained Inception-v1 model to BigDL
- Add FC layer with SoftMax classifier
- Train on partial dataset with pre-trained weights using Analytics Zoo on Spark Cluster
- Scale training on multi-node cluster in AWS Databricks to train large whole dataset (1M images)

Model Development - Phase 1

Inception v1

Customized Classifier



Code- Phase 1

Transfer Learning Training

Cmd 32

```
# get embedding
pretrained_model_path = path.join(MODEL_ROOT,"bigdl_inception_v1_imagenet_0_4_0.model")
n_classes = len(label_dict)# label categories
preTrainedNNModel = NNModel(Model.loadModel("dbfs:" + pretrained_model_path), train_transformer)\
.setFeaturesCol("image")\
.setPredictionCol("embedding")\
.setBatchSize(batch_size)

train_embedding = preTrainedNNModel.transform(train_image)
val_embedding = preTrainedNNModel.transform(val_image)

# train transfered model
lrModel = Sequential().add(Linear(1000, n_classes)).add(LogSoftMax())
classifier = NNClassifier(lrModel, ClassNLLCriterion(), SeqToTensor([1000]))\
.setLearningRate(learning_rate)\
.setBatchSize(batch_size)\
.setMaxEpoch(no_epochs)\
.setFeaturesCol("embedding")\
.setValidation(EveryEpoch(), val_embedding, [Top1Accuracy()], batch_size)

start = time.time()
trained_model = classifier.fit(train_embedding)
end = time.time()
print("Optimization Done.")
print("Training time is: %s seconds" % str(end-start))
```

Predict and Evaluation

```
#predict
test_embedding = preTrainedNNModel.transform(test_image)
predict_model = trained_model.setBatchSize(batch_size)
predictionDF = predict_model.transform(test_embedding)
predictionDF.show()
```

Cmd 37

```
'''
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)
```

Code- Phase 1

Fine Tune Training

Cmd 32

```
# 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, 8:03:46 PM on 20-node-cluster

Cmd 33

```
# 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")
```

Predict 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()
```

Results - Phase 1

- Training from scratch vs Transfer Learning vs Fine Tuning (on a partial dataset)
- Dataset: 1927 images, 9 categories

Training Type	Epochs	Training Time (sec)	Accuracy (%)
Training from scratch	40	1266	23.9
Transfer Learning	40	210	65.4
Fine Tuning	15	489	81.5

** Accuracy numbers are in that range due to a small part of dataset being used*

Results - Phase 1

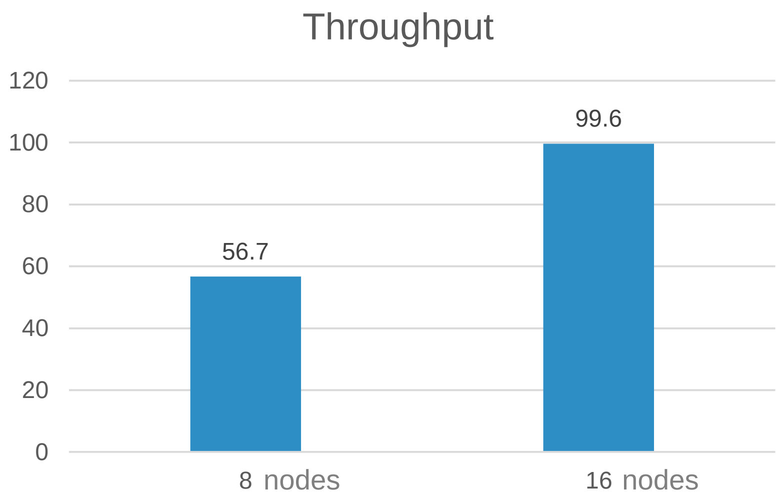
- Fine tune with Inception v1 on a full dataset
- Dataset: 994325 images, 69 categories

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

Scaling Results - Phase 1

Nodes	Batch Size	Epochs	Through put	Training Time
8	256	20	56.7	745.6
16	256	20	99.6	424.7
8	128	20	55.3	764.9
16	128	20	80.7	524.4



Scaling test was run on AWS R4 instance that include the following features:

- dual socket Intel Xeon E5 Broadwell processors (2.3 GHz)
- DDR4 memory
- Hardware Virtualization (HVM) only

Model	vCPUs	Memory (GiB)	Networking Performance
r4.xlarge	4	30.5	Up to 10 Gigabit

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
  - Reference model:
    - Object Detection (Fast-RCNN, SSD)
    - Tesseract(https://github.com/tesseract-ocr/tesseract)
    - CTPN(https://github.com/tianzhi0549/CTPN)

Next Steps – Phase 2

- Recognize text
  - Tesseract (https://github.com/tesseract-ocr/tesseract)
  - CRNN (https://github.com/bgshih/crnn)
- Determine whether text contains PII (personal identifiable information)
  - Recognize PII with leading words (RNN)
- Blur areas with PII text
  - Image tools

Summary

Key Takeaways

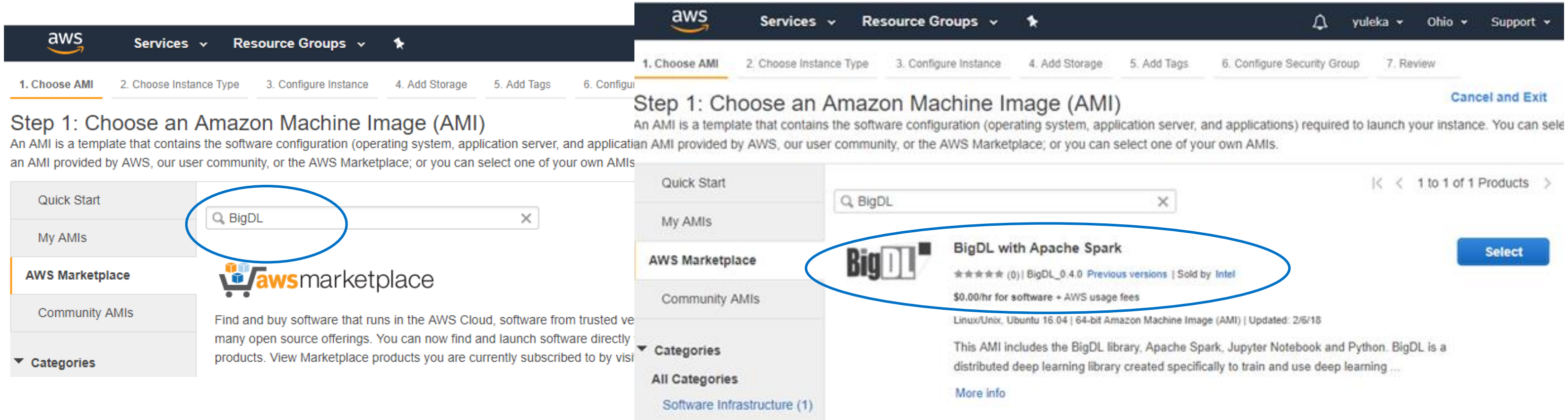
- AI on Apache Spark is a reality with use cases like World Bank
- BigDL makes distributed deep learning and AI more accessible both for big data users and data scientists
- Analytics Zoo provide high level APIs more accessible to Data Scientists and Spark Users (Keras, Spark DataFrame, etc.)
- BigDL can leverage on existing Spark/Hadoop infrastructure and also runs deep learning applications in the cloud (AWS, Azure, GCP, …)
- Join in and contribute to projects:

Analytics Zoo: github.com/intel-analytics/analytics-zoo

BigDL: github.com/intel-analytics/BigDL

Call to Action

- Try BigDL on AWS - lookup BigDL AMI on AWS Marketplace



- Try image classification with Analytics Zoo using BigDL– this use case code is shared on https://github.com/intel-analytics/WorldBankPoC

Questions?

Notices and Disclaimers

- The findings, interpretations, and conclusions expressed in this document do not necessarily reflect the views of the World Bank Group.
- 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 visit: **http://www.intel.com/performance**.
- 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 **www.intel.com/benchmarks**.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.
- 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.
- Intel, the Intel logo, Xeon, Xeon Phi and Nervana 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
-