DeepLearning4J and Spark

Table of Contents

Intro to DeepLearning4J

DeepLearning Framework built in Java

Supported by Skymind.io

Enterprise focus

Goals of the DeepLearning4J project

- Provide a Toolkit for using DeepLearning on the JVM
  - Enterprise users
  - Security
  - Flexibility

DeepLearning4J sub-projects

- DataVec
  - Tools for ETL
- ND4J
  - NUmeric Arrays
  - NumPY for the JVM
- libnd4j
  - Native Libraries for GPUs/CPUs
- DeepLearning4J
  - Tools to train Neural Networks

DataVec

- ETL Libraries purpose built for Neural Networks
  - Neural networks process numeric arrays
  - Datavec helps you get from your_data => numeric array

Data Sources Supported by DataVec

- Log files
- Text documents
- Tabular data
- Images and video
- and more !!

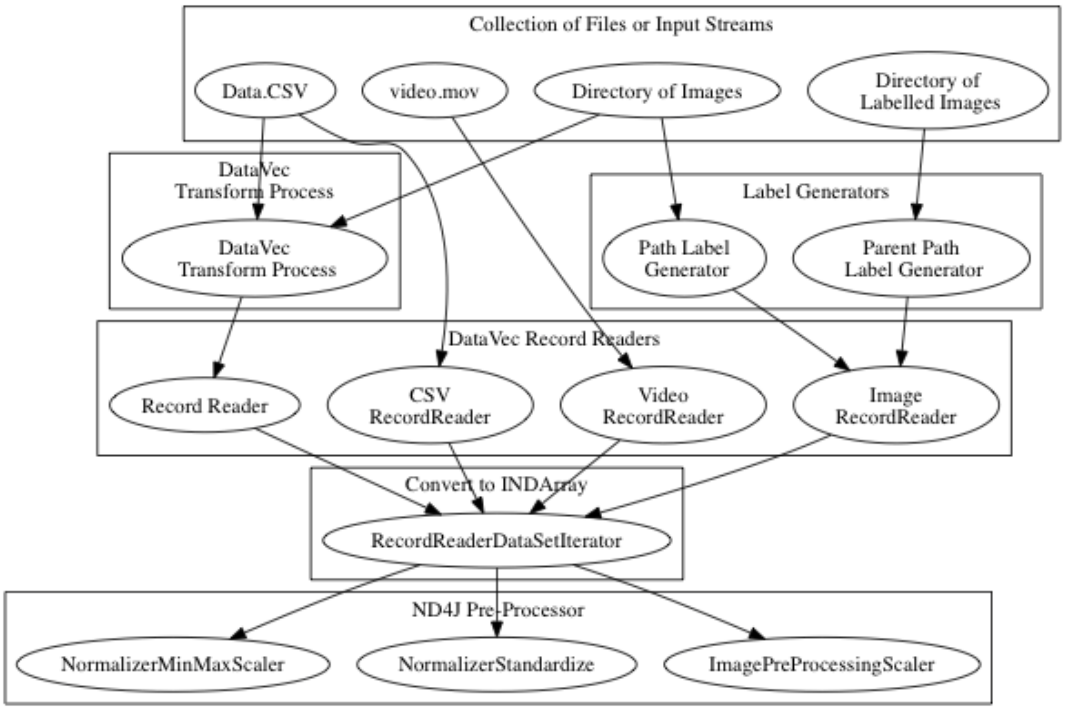
DataVec Features

- Transformation
- Scaling
- Shuffling
- Joining
- Splitting

Commonly Used Features

- RecordReaders
    - Read files or input, convert to List of Writables
- Normalizers
    - Standardize, scale or normalize the data
- Transform Process
    - Join datasets, replace strings with numerics, extract labels

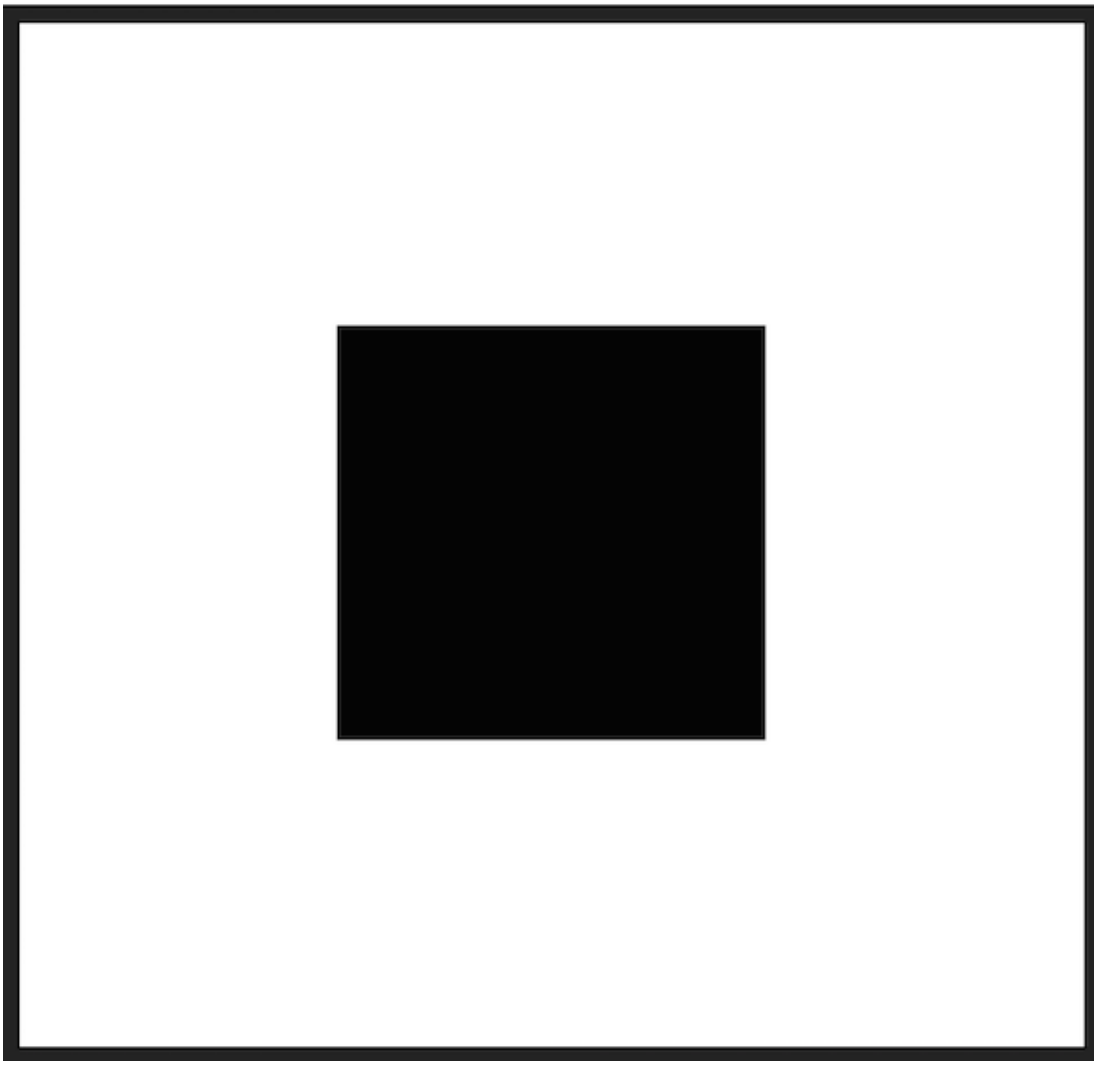
Diagram of available ETL paths



skymind

DataVec Image Basic Example

- Images are arrays of pixel values



skymind

Code Example:

- Load above image as INDArray

```
INDArray imagematrix = loader.asMatrix(image);
System.out.println(imagematrix);
```

- Output

```
[[[[255.00, 255.00, 255.00, 255.00],
[255.00, 0.00, 0.00, 255.00],
[255.00, 0.00, 0.00, 255.00],
[255.00, 255.00, 255.00, 255.00]]]]
```

Code Example:

- Scale values between 0 and 1

```
DataNormalization scaler = new ImagePreProcessingScaler(0,1);
scaler.transform(imagematrix);
```

- Output

```
[[[[1.00, 1.00, 1.00, 1.00],
[1.00, 0.00, 0.00, 1.00],
[1.00, 0.00, 0.00, 1.00],
[1.00, 1.00, 1.00, 1.00]]]]
```

Manipulating Images with DataVec

- Scale images to same height/width with RecordReader
    - Used when processing data in bulk during training

      ImageRecordReader recordReader = new ImageRecordReader(height,width,channels);
- Scale image to appropriate dimenstions with NativeImageLoader
    - Used when processing a single image for inference

      NativeImageLoader loader = new NativeImageLoader(height, width, channels); \ load and scale INDArray image = loader.asMatrix(file); \ create INDarray INDArray output = model.output(image); \ get model prediction for image

Image Data Set Augmentation

- Create "larger" training set with OpenCV/dataVec tools
  - Transform
  - Crop
  - Skew

Applying Labels

- DataVec provides the following tools for generating Labels
  - ParentPathLabelGenerator
  - PathLabelGenerator

Available Record Readers

- Table of available record readers:
  - https://deeplearning4j.org/etl-userguide

Code Example: Image Transform

- Scale pixel values from 0-255 to 0-1

```
DataNormalization scaler = new ImagePreProcessingScaler(0,1);
scaler.fit(dataIter);
dataIter.setPreProcessor(scaler);
```

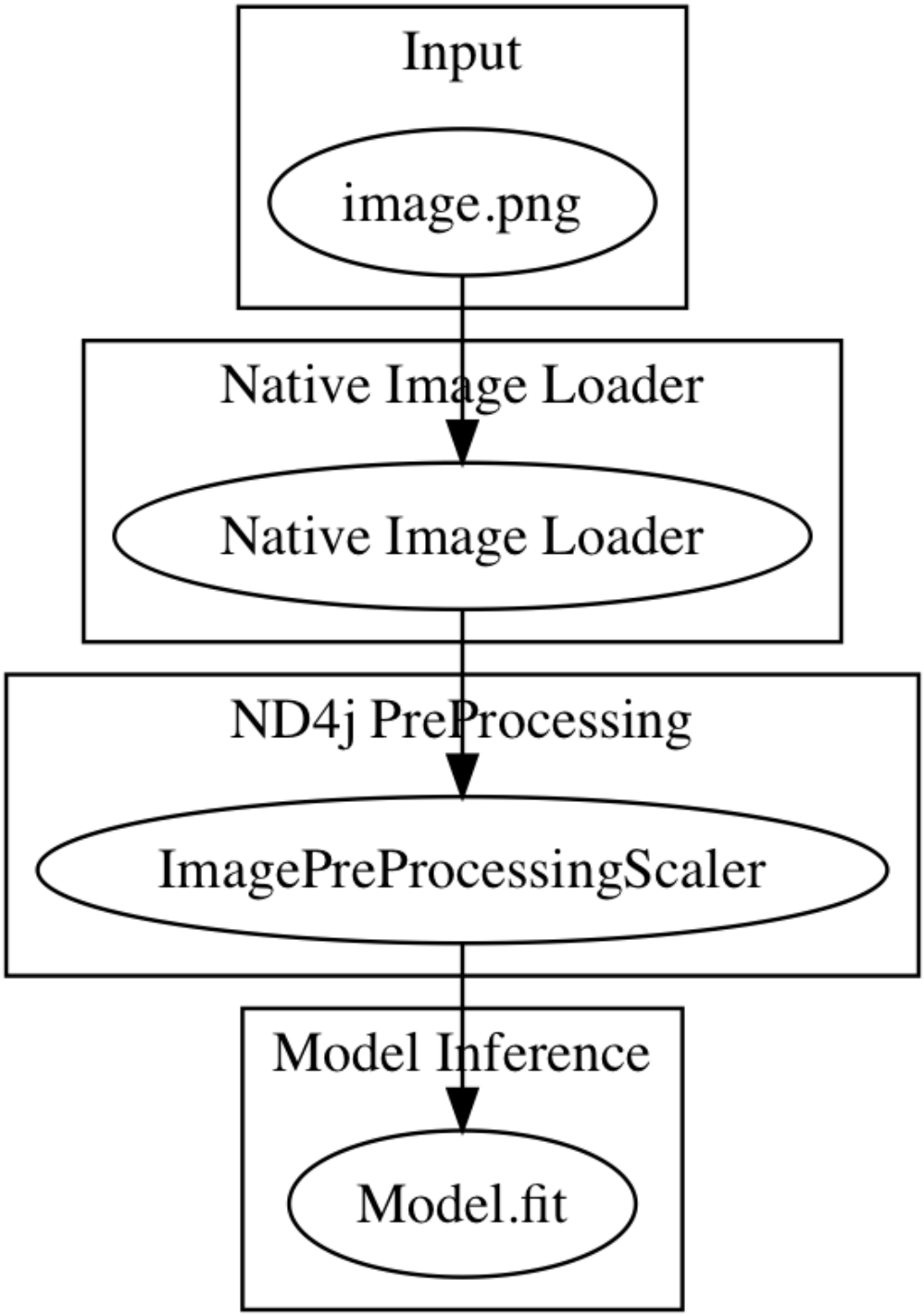
Available ND4J Pre-Processors

- ImagePreProcessingScaler
  - min max scaling default 0 + - 1
- NormalizerMinMaxScaler
  - Scale values observed min -> 0, observed max -> 1
- NormalizerStandardize
  - moving column wise variance and mean
  - no need to pre-process

Image Transforms

- Included Libraries
  - JavaCV
  - OpenCV
  - ffmpeg

Image pipeline Single Image to Pre-Trained Model



Code Example: CSV Data to INDArray

```
public class CSVExample {
private static Logger log = LoggerFactory.getLogger(CSVExample.class);

public static void main(String[] args) throws  Exception {

    //First: get the dataset using the record reader. CSVRecordReader handles loadi
    int numLinesToSkip = 0;
    String delimiter = ",";
    RecordReader recordReader = new CSVRecordReader(numLinesToSkip,delimiter);
    recordReader.initialize(new FileSplit(new ClassPathResource("iris.txt").getFile

    //Second: the RecordReaderDataSetIterator handles conversion to DataSet objects
    int labelIndex = 4;     //5 values in each row of the iris.txt CSV: 4 input fea
    int numClasses = 3;     //3 classes (types of iris flowers) in the iris data se
    int batchSize = 150;    //Iris data set: 150 examples total. We are loading all

    DataSetIterator iterator = new RecordReaderDataSetIterator(recordReader,batchSi
    DataSet allData = iterator.next();
```

DataVec Code Explained

- RecordReader recordReader = new CSVRecordReader(numLinesToSkip,delimiter);
  - A RecordReader prepares a list of Writables
  - A Writable is an efficient Serialization format
- DataSetIterator iterator = new RecordReaderDataSetIterator
  - We are in DL4J know, with DataSetIterator
  - Builds an Iterator over the list of records
- DataSet allData = iterator.next();
  - Builds a DataSet
  - INDArray of Features, INDArray of Labels

Frequently Used DataVec classes

- CSVRecordReader
  - CSV text data
- ImageRecordReader
  - Convert image to numeric array representing pixel values
- JacksonRecordReader
  - Parses JSON records
- ParentPathLabelGenerator
  - Builds labels based on directory path
- Transform, Transform Process Builder, TransformProcess
  - Conversion tools

ND4J

- Provides scientific computing libraries
- Main features
    - Versatile n-dimensional array object
    - Multiplatform functionality including GPUs
    - Linear algebra and signal processing functions

ND4J and DeepLearning

- Classes frequently Used
  - DataSet
    - Container for INDArrays of Features/Labels
  - DataSetIterator
    - Build DataSet from RecordReader

libND4J

- The C++ engine that powers ND4J
  - Speed
  - CPU and GPU support

A Simple Neural Network in DeepLearning4J

- Iris Data, 3 Species, 4 features

```
package ai.skymind.training.demos;

import org.datavec.api.records.reader.RecordReader;
import org.datavec.api.records.reader.impl.csv.CSVRecordReader;
import org.datavec.api.split.FileSplit;
import org.datavec.api.util.ClassPathResource;
import org.deeplearning4j.datasets.datavec.RecordReaderDataSetIterator;
import org.deeplearning4j.eval.Evaluation;
import org.deeplearning4j.nn.conf.MultiLayerConfiguration;
import org.deeplearning4j.nn.conf.NeuralNetConfiguration;
import org.deeplearning4j.nn.conf.layers.DenseLayer;
import org.deeplearning4j.nn.conf.layers.OutputLayer;
import org.deeplearning4j.nn.multilayer.MultiLayerNetwork;
import org.deeplearning4j.nn.weights.WeightInit;
import org.deeplearning4j.optimize.listeners.ScoreIterationListener;
import org.nd4j.linalg.activations.Activation;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.dataset.DataSet;
```

A Simple Neural Network Continued.....

```
import org.nd4j.linalg.dataset.SplitTestAndTrain;
import org.nd4j.linalg.dataset.api.iterator.DataSetIterator;
import org.nd4j.linalg.dataset.api.preprocessor.DataNormalization;
import org.nd4j.linalg.dataset.api.preprocessor.NormalizerStandardize;
import org.nd4j.linalg.lossfunctions.LossFunctions;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

A Simple Neural Network Continued.....

```
public class IrisNoImport {

private static Logger log = LoggerFactory.getLogger(IrisNoImport.class);

public static void main(String[] args) throws  Exception {

//First: get the dataset using the record reader.
//CSVRecordReader handles loading/parsing
int numLinesToSkip = 0;
String delimiter = ",";
RecordReader recordReader = new CSVRecordReader(numLinesToSkip,delimiter);
recordReader.initialize(new FileSplit(new ClassPathResource("iris.txt").getFile()))

//Second: the RecordReaderDataSetIterator handles conversion to
//DataSet objects, ready for use in neural network
int labelIndex = 4;
int numClasses = 3;
int batchSize = 150;
```

A Simple Neural Network Continued.....

```
DataSetIterator iterator = new RecordReaderDataSetIterator(recordReader,batchSize,l
DataSet allData = iterator.next();
allData.shuffle();
SplitTestAndTrain testAndTrain = allData.splitTestAndTrain(0.65);  //Use 65% of dat

DataSet trainingData = testAndTrain.getTrain();
DataSet testData = testAndTrain.getTest();
DataNormalization normalizer = new NormalizerStandardize();
normalizer.fit(trainingData);
normalizer.transform(trainingData);
//Apply normalization to the training data
normalizer.transform(testData);
```

A Simple Neural Network Continued.....

```
final int numInputs = 4;
int outputNum = 3;
int iterations = 1000;
long seed = 6;


log.info("Build model....");
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(seed)
    .iterations(iterations)
    .activation(Activation.TANH)
    .weightInit(WeightInit.XAVIER)
    .learningRate(0.1)
    .regularization(true).l2(1e-4)
    .list()
    .layer(0, new DenseLayer.Builder().nIn(numInputs).nOut(3)
        .build())
    .layer(1, new DenseLayer.Builder().nIn(3).nOut(3)
        .build())
    .layer(2, new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIK
        .activation(Activation.SOFTMAX)
        .nIn(3).nOut(outputNum).build())
    .backprop(true).pretrain(false)
    .build();
```

A Simple Neural Network Continued.....

```
        //run the model
        MultiLayerNetwork model = new MultiLayerNetwork(conf);
        model.init();
        model.setListeners(new ScoreIterationListener(100));

        model.fit(trainingData);

        //evaluate the model on the test set
        Evaluation eval = new Evaluation(3);
        INDArray output = model.output(testData.getFeatureMatrix());
        eval.eval(testData.getLabels(), output);
        log.info(eval.stats());
        }

    }
```

DataVec: Spark Transform Process

DataVec overview

- Goal of DataVec
- Data => INDArray

DataVec Spark Based Operations

- Joins
- Column Transformations
- Data Analysis

Spark Analyze

- Tool to gather statistics across a data set

Spark Analyze Example

- The Data: classic dataset of Iris flower characteristics

```
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
4.6,3.1,1.5,0.2,0
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
```

Spark Analyze Example Continued...

- Define a Schema

```
Schema schema = new Schema.Builder()
.addColumnsDouble("Sepal length", "Sepal width",
"Petal length", "Petal width")
.addColumnInteger("Species")
.build();
```

Spark Analyze Example Continued…

- Create Spark Conf

```
SparkConf conf = new SparkConf();
conf.setMaster("local[*]");
conf.setAppName("DataVec Example");
JavaSparkContext sc = new JavaSparkContext(conf);
```

Spark Analyze Example Continued…

- Read Data

```
String directory = new ClassPathResource("IrisData/iris.txt")
.getFile().getParent();
//Normally just define your directory
//like "file:/..." or "hdfs:/..."
JavaRDD<String> stringData = sc.textFile(directory);
```

Spark Analyze Example Continued…

- Parse Data

```
//We first need to parse this comma-delimited (CSV) format;
//we can do this using CSVRecordReader:
RecordReader rr = new CSVRecordReader();
JavaRDD<List<Writable>> parsedInputData =
stringData.map(new StringToWritablesFunction(rr));
```

Spark Analyze Example Continued…

- Analyze Data

```
int maxHistogramBuckets = 10;
DataAnalysis dataAnalysis =
AnalyzeSpark.analyze(schema,
parsedInputData, maxHistogramBuckets);

System.out.println(dataAnalysis);
```

Spark Analyze Output

- min,max,mean,StDev,Variance,num0,
- numNegative,numPositive,numMin,numMax,Total
- See example IrisAnalysis.java

Spark Analyze options

- Per Column Analysis

```
//We can get statistics on a per-column basis:
DoubleAnalysis da = (DoubleAnalysis)dataAnalysis.getColumnAnalysis("Sepal
double minValue = da.getMin();
double maxValue = da.getMax();
double mean = da.getMean();
```

Spark Analyze HTML output

- To Generate HTML Output

```
HtmlAnalysis.createHtmlAnalysisFile(dataAnalysis,
new File("DataVecIrisAnalysis.html"));

//To write to HDFS instead:
//String htmlAnalysisFileContents =
//HtmlAnalysis.createHtmlAnalysisString(dataAnalysis);
//SparkUtils.writeStringToFile("hdfs://your/hdfs/path/here",
//htmlAnalysisFileContents,sc);
```

DataVec Spark Transform Process

- Joining
- Transformation
- Schema alteration

Video

- https://www.youtube.com/watch?v=MLEMw2NxjxE

Code Examples

- https://github.com/deeplearning4j/dl4j-examples/tree/master/datavec-examples
- http://github.com/SkymindIO/screencasts/tree/master/datavec_spark_transform

Basic Example CSV Data

- Storm Reports Data

```
161006-1655,UNK,2 SE BARTLETT,LABETTE,KS,37.03,-95.19,
TRAINED SPOTTER REPORTS TORNADO ON THE GROUND. (ICT),TOR

//Fields are
//datetime,severity,location,county,state,lat,lon,comment,type
```

Define Schema as read

```
Schema inputDataSchema = new Schema.Builder()
.addColumnsString("datetime","severity","location","county","state")
.addColumnsDouble("lat","lon")
.addColumnsString("comment")
.addColumnCategorical("type","TOR","WIND","HAIL")
.build();
```

Define Transform process

```
TransformProcess tp = new TransformProcess.Builder(inputDataSchema)
.removeColumns
("datetime","severity","location","county","state","comment")
.categoricalToInteger("type")
.build();
// keep lat/lon convert type to 0,1,2
```

Create Spark Conf

```
SparkConf sparkConf = new SparkConf();
sparkConf.setMaster("local[*]");
sparkConf.setAppName("Storm Reports Record Reader Transform");
JavaSparkContext sc = new JavaSparkContext(sparkConf);
```

read the data file

```
JavaRDD<String> lines = sc.textFile(inputPath);
```

Convert to Writable

```
JavaRDD<List<Writable>> stormReports =
lines.map(new StringToWritablesFunction(new CSVRecordReader()));
```

Run our transform process

```
JavaRDD<List<Writable>> processed =
SparkTransformExecutor.execute(stormReports,tp);
```

Convert Writable Back to String for Export

```
JavaRDD<String> toSave=
processed.map(new WritablesToStringFunction(","));
```

Write to File

```
        toSave.saveAsTextFile(outputPath);
```

Distributed Training in Spark

- Documentation
  - https://deeplearning4j.org/spark
- Examples
  - https://github.com/deeplearning4j/dl4j-examples/tree/master/dl4j-spark-examples

Non-Distributed Training overview

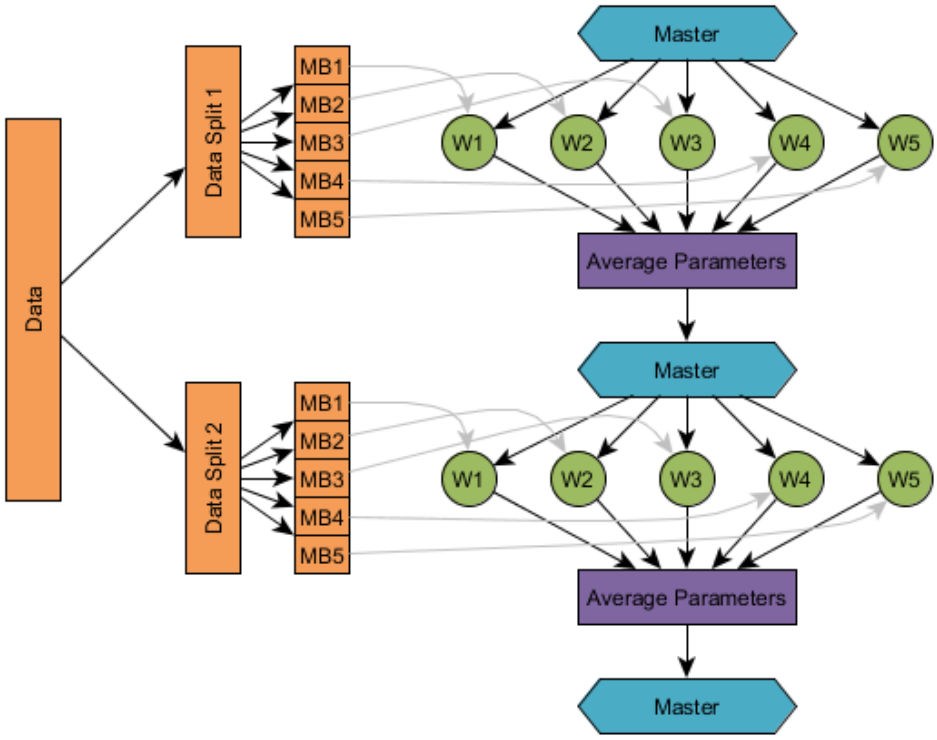
- Process minibatch
  - Calculate Loss Function
  - Calculate direction of less error
  - Update weights in that direction

Non-Distributed Training Challenges

- Number of parameters may be large
- CPU or preferably GPU intensive
- Update large multi-dimensional matrix of numeric values

Distributed Training Implementation

- Workers process minibatch
- Calculate Loss Function
- Calculate Gradient update
- Submit to Parameter server
- Parameter Server averages weights from workers
- Ships averaged weights to workers


Master
Data
Data Split 1
MB1
MB2
MB3
MB4
MB5
W1
W2
W3
W4
W5
Average Parameters
Master
Data Split 2
MB1
MB2
MB3
MB4
MB5
W1
W2
W3
W4
W5
Average Parameters
Master

skymind

DL4J Spark Examples

- https://github.com/deeplearning4j/dl4j-examples/tree/master/dl4j-spark-examples

Minimal Example

```
JavaSparkContent sc = ...;
JavaRDD<DataSet> trainingData = ...;
MultiLayerConfiguration networkConfig = ...;

//Create the TrainingMaster instance
int examplesPerDataSetObject = 1;
TrainingMaster trainingMaster = new ParameterAveragingTrainingMaster.Builder(exampl
    .(other configuration options)
    .build();

//Create the SparkDl4jMultiLayer instance
SparkDl4jMultiLayer sparkNetwork = new SparkDl4jMultiLayer(sc, networkConfig, train

//Fit the network using the training data:
sparkNetwork.fit(trainingData);
```

How to Participate and Contribute

- Chat with us on Gitter
  - https://gitter.im/deeplearning4j/deeplearning4j
- Contribute
  - https://github.com/deeplearning4j/

Interesting Challenges

- GPU aware Yarn
  - https://issues.apache.org/jira/browse/YARN-5517
- Parallelism in DeepLearning
  - https://static.googleusercontent.com/media/research.google.com/en//archive/large_deep_netwo