AI研习社

基于Java的开源深度学习框架-Deeplearning4j的介绍与实例分享

分享人:万宫玺

Outline

AI研习社

- Deeplearning4j Intro
- Deeplearning4j Eco-system
  - Deeplearning4j
  - ND4j
  - DataVec
- Deeplearning4j Modeling
  - Single
  - Parallel
  - Distributed with Spark
- Example: Develop & Deploy
- Transfer Learning & Reinforcement Learning in Deeplearning4j
- Summary
- About AI

AI研习社

Deeplearning4j Intro

- Deeplearning4j是由AI创业公司Skymind(skymind.ai/)主导开发并维护的基于Java/JVM的深度学习开源框架。包括腾讯、SVAngel、GreatPointVentures等都参与了对这家公司的投资。NASA喷气实验室、IBM、雪弗兰、埃森哲等企业都是其用户。
- Deeplearning4j最新版本：0.9.2/1.0.0
- Deeplearning4j支持CPU/GPU集群分布式大规模深度学习模型的训练
- Deeplearning4j是为数不多原生态支持Apache Spark的深度学习开源框架
- Deeplearning4j于2017.10月进入Eclipse社区
- 官网：deeplearning4j.org/
- github地址：github.com/deeplearning4j

Deeplearning4j Eco-system

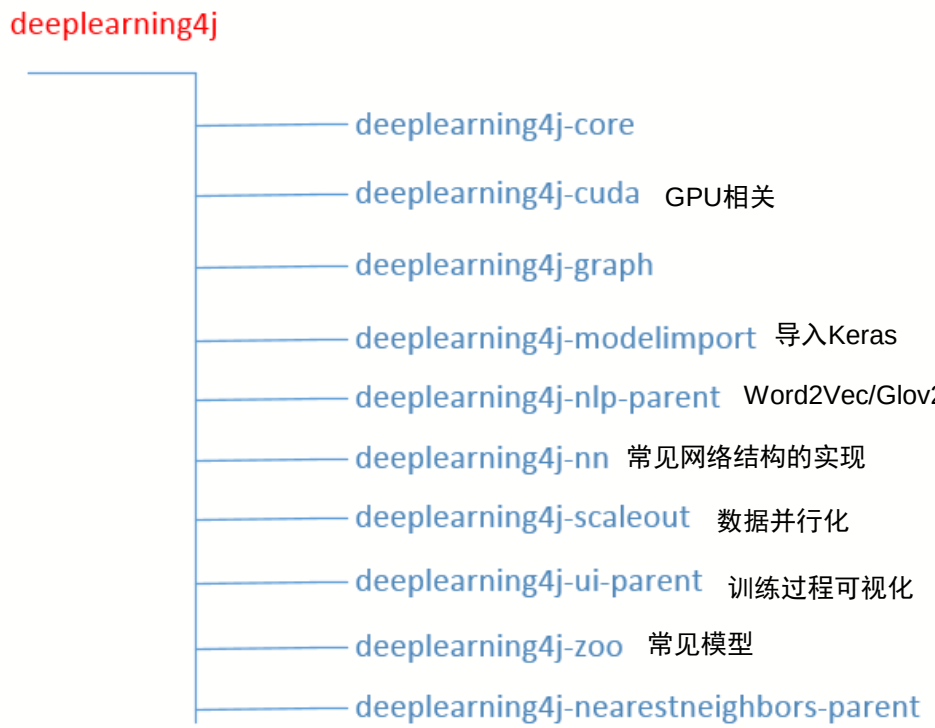
- Deep Learning for Java/JVM（dl4j）
- N-Dimensional Arrays for Java/JVM（nd4j/libnd4j）
- Data ETL Library for Machine/Deep Learning（datavec）
- Deep Reinforcement Learning for Java/JVM（rl4j）
- Hyperparameter Tuning（arbiter）
- Examples（dl4j-examples）
- Model Zoo（dl4j-model-zoo）
- Others（nd4s/scalnet/dl4j-benchmark …）

Deeplearning4j Eco-system

AI研习社

--dl4j Module

- deeplearning4j(github.com/deeplearning4j/deeplearning4j)中定



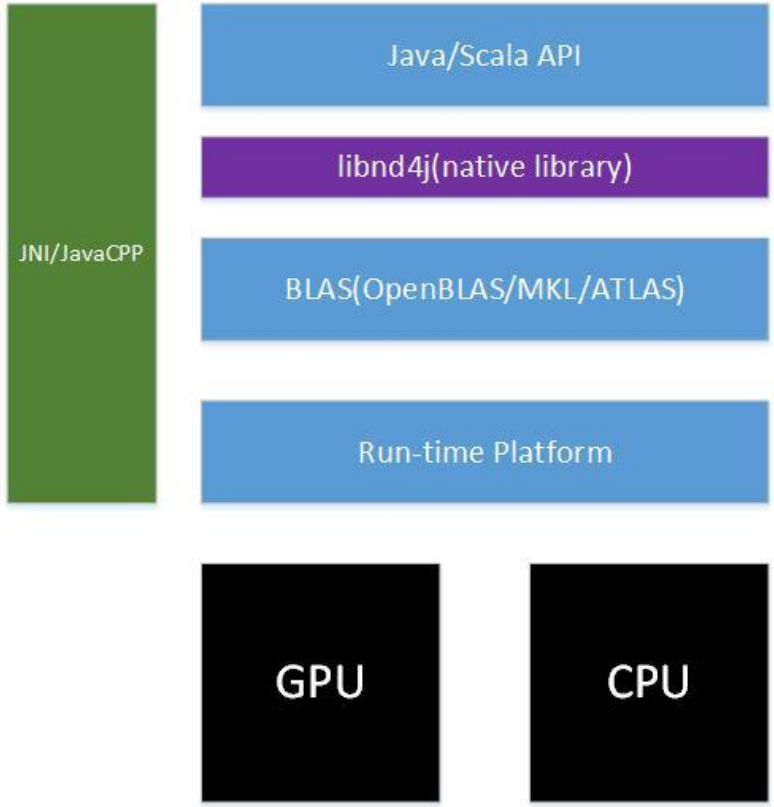

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <nd4j.version>0.8.0</nd4j.version>
  <dl4j.version>0.8.0</dl4j.version>
  <datavec.version>0.8.0</datavec.version>
  <scala.binary.version>2.11</scala.binary.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>deeplearning4j-core</artifactId>
    <version>${dl4j.version}</version>
  </dependency>
  <dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>dl4j-spark_${scala.binary.version}</artifactId>
    <version>${dl4j.version}_spark_2</version><!--2 for spark-2.x.x,1 for spark 1.5/1.6-->
  </dependency>
  <dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>deeplearning4j-ui_${scala.binary.version}</artifactId>
    <version>${dl4j.version}</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

Deeplearning4j Eco-system

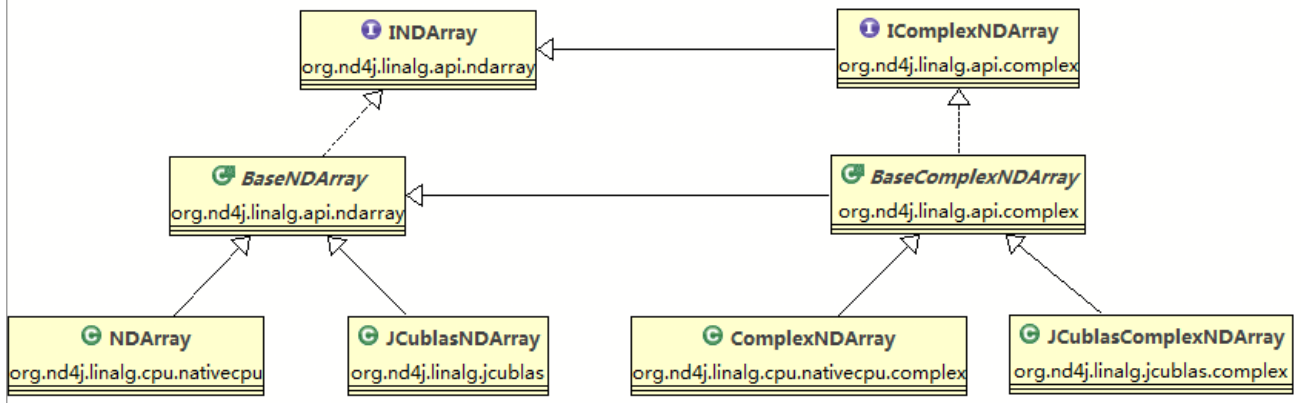
AI研习社

--nd4j module

- ND4j可当作是Java版的Numpy，定义了各种张量运算
- ND4j的后台可在BLAS的开源库中切换




```
<dependency>
  <groupId>org.nd4j</groupId>
  <artifactId>nd4j-native</artifactId>
  <version>${nd4j.version}</version>
</dependency>
<dependency>
  <groupId>org.nd4j</groupId>
  <artifactId>nd4j-cuda-8.0</artifactId>
  <version>${nd4j.version}</version>
  <scope>provided</scope>
</dependency>
```



AI研习社

Deeplerning4j Eco-system

--nd4j memory management

- ND4j利用堆外内存(off-heap memory)来存储张量对象
- ND4j中的堆上内存(on-heap memory)存储张量对象的引用/指针
- ND4j中实际的张量运算后台时BLAS的开源实现，将张量对象作为堆外内存存储，可加快运行效率
- 堆上内存：-Xms -Xmx
- 堆外内存： -Dorg.bytedeco.javacpp.maxbytes
  -Dorg.bytedeco.javacpp.maxphysicalbytes
- Spark on Yarn Executor内存调优： --executor-memory调整堆上内存
  --conf “spark.executor.extraJavaOptions”
  --conf “spark.yarn.executor.memoryOverhead”调整堆外内存

Deeplearning4j Eco-system

AI研习社

--nd4j example

```java
@Test
public void create(){
    INDArray default_order_nd = Nd4j.create(new double[][]{{1.0,2.0,3.0},{4.0,5.0,6.0}});
    double[] default_order_array = default_order_nd.data().asDouble();
    println("Default: ");
    println(Arrays.toString(default_order_array));
    //
    INDArray c_order_nd = Nd4j.create(new double[][]{{1.0,2.0,3.0},{4.0,5.0,6.0}} , 'c');
    double[] c_order_array = c_order_nd.data().asDouble();
    println("C/C++: ");
    println(Arrays.toString(c_order_array));
    //
    INDArray f_order_nd = Nd4j.create(new double[][]{{1.0,2.0,3.0},{4.0,5.0,6.0}} , 'f');
    double[] f_order_array = f_order_nd.data().asDouble();
    println("Fortran: ");
    println(Arrays.toString(f_order_array));
}
```

```
Default:
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
C/C++:
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
Fortran:
[1.0, 4.0, 2.0, 5.0, 3.0, 6.0]
```

```java
@Test
public void muli(){ //hadmard product
    INDArray data1 = Nd4j.create(new double[]{1.0,2.0});
    INDArray data2 = Nd4j.create(new double[]{3.0,4.0});
    println("hadmard product: " + data1.muli(data2));
    println("data1: " + data1);
    println("data2: " + data2);
}
```

```
hadmard product: [3.00, 8.00]
data1: [3.00, 8.00]
data2: [3.00, 4.00]
```

AI研习社

Deeplearning4j Eco-system

--DataVec

- DataVec用于将原始数据(音频、图像、文本等)转化为可用于机器学习训练的张量数据
- 图像的处理：读取/保存常见格式的图片、灰度化、反转、规整等常见操作(backend:opencv)
- 文本的处理：分词、停用词过滤、词频统计等
- 音频的处理：加窗分帧、采样、 FFT变换等
- 其他格式：支持JSON/XML/YAML，LibSVM，MAT等格式的读取

AI研习社

Deeplearning4j Modeling

- Step1：读取数据，以DataSetIterator的形式或者DataSet/INDArray的某种形式构建训练/验证/测试的数据
- Step2：定义神经网络结构和超参数，形成网络配置(MultiLayerConfiguration/ComputationGraphConfiguration)，构建模型
- Step3：训练（一般为多轮训练）神经网络，调优，评估，并保存模型（CPU/GPU）
- Step4：加载模型用于线下批量/线上实时的预测

Deeplearning4j Modeling

AI研习社

--Single

```
DataSetIterator mnistTrain = new MnistDataSetIterator(batchSize,true,12345);
DataSetIterator mnistTest = new MnistDataSetIterator(batchSize,false,12345);
```

加载数据

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
        .seed(seed)
        .iterations(iterations)
        .weightInit(WeightInit.XAVIER)
        .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
        .updater(Updater.ADAM)
        .list()
        .layer(0, new ConvolutionLayer.Builder(5, 5)
                .stride(1, 1)
                .nOut(32)
                .activation(Activation.IDENTITY)
                .build())
        .layer(1, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
                .kernelSize(2,2)
                .stride(2,2)
                .build())
        .layer(2, new DenseLayer.Builder().activation(Activation.RELU)
                .nOut(500).build())
        .layer(3, new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                .nOut(outputNum)
                .activation(Activation.SOFTMAX)
                .build())
        .setInputType(InputType.convolutionalFlat(height ,width, nchannel))
        .backprop(true).pretrain(false).build();
MultiLayerNetwork model = new MultiLayerNetwork(conf);
model.init();
```

配置模型结构以及超参数，
并初始化模型参数

```
model.setListeners(new ScoreIterationListener(1));
for( int i=0; i<nEpochs; i++ ) {
    model.fit(mnistTrain);
    log.info("*** Completed epoch {} ***", i);

    log.info("Evaluate model....");
    Evaluation eval = model.evaluate(mnistTest);
    log.info(eval.stats());
    mnistTest.reset();
}
```

训练神经网络模型，并利
用测试数据集进行评估

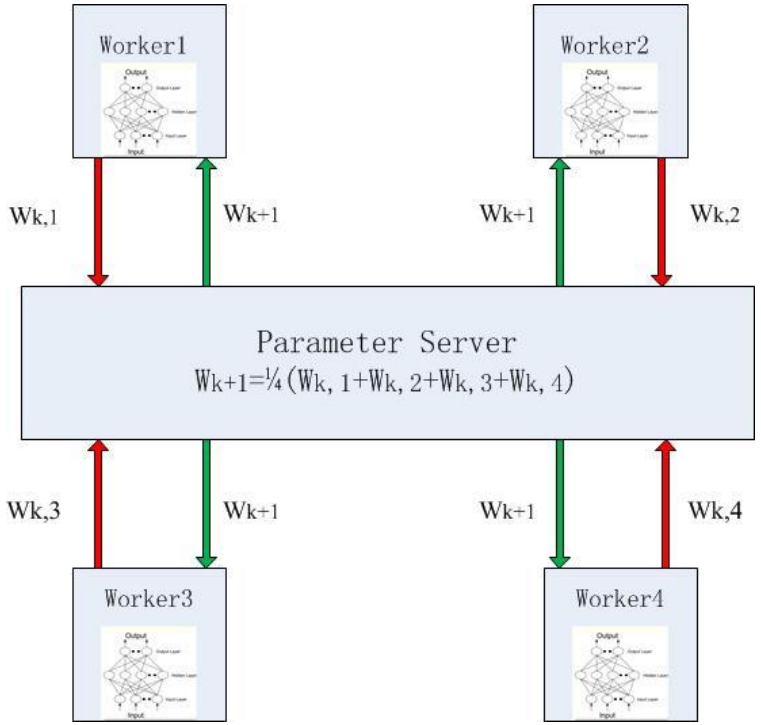
AI研习社

Deeplearning4j Modeling

--Data Parallelism

- 每一个计算节点都有完整的网络结构的拷贝
- 每一个计算节点读取并训练数据集中的不同子集
- Deeplearning4j的数据并行化训练方案：
  - 参数or梯度：参数平均
  - 同步or异步：同步
- PS：模型并行化策略信息可见

《Large scale distributed deep networks》



Deeplearning4j Modeling

AI研习社

--Parallel

```
ParallelWrapper wrapper = new ParallelWrapper.Builder<MultiLayerNetwork>(model)
                        // 预先读取训练数据
                        .prefetchBuffer(prefetch)
                        // 设备数量（core number）
                        .workers(numworker)
                        // 平均参数的频率
                        .averagingFrequency(freq)
                        // 输出loss score
                        .reportScoreAfterAveraging(true)
                        .build();
```

声明数据并行化的实例
Multi-CPU/Multi-GPU

```
wrapper.setListeners(new ScoreIterationListener(1));
for( int i=0; i<nEpochs; i++ ) {
    wrapper.fit(mnistTrain);
    log.info("*** Completed epoch {} ***", i);

    log.info("Evaluate model....");
    Evaluation eval = model.evaluate(mnistTest);
    log.info(eval.stats());
    mnistTest.reset();
}
```

并行训练

Deeplearning4j Modeling

--Distributed with Spark

AI研习社

```
JavaRDD<DataSet> trainData = sc.parallelize(trainDataList);
JavaRDD<DataSet> testData = sc.parallelize(testDataList);
```

以RDD形式加载训练数据

```
//配置参数服务器：parameterserver
TrainingMaster tm = new ParameterAveragingTrainingMaster.Builder(batchSize)
                        .averagingFrequency(aveFreq)
                        .workerPrefetchNumBatches(prefetch)
                        .batchSizePerWorker(batchSize)
                        .build();
```

Parameter Server的声明

```
//声明分布式训练的模型并传入参数，进行多轮训练
SparkDl4jMultiLayer sparkNet = new SparkDl4jMultiLayer(sc, netconf, tm);

for (int i = 0; i < numEpochs; i++) {
    sparkNet.fit(trainData);
    log.info("Completed Epoch {}", i);
}
//评估spark上训练的模型
Evaluation evaluation = sparkNet.evaluate(testData);
log.info("***** 模型评估 *****");
log.info(evaluation.stats());
//删除hdfs上的临时文件
tm.deleteTempFiles(sc);
log.info("**** spark 任务结束 *****");
```

Spark集群分布式训练

Deeplerning4j Modeling

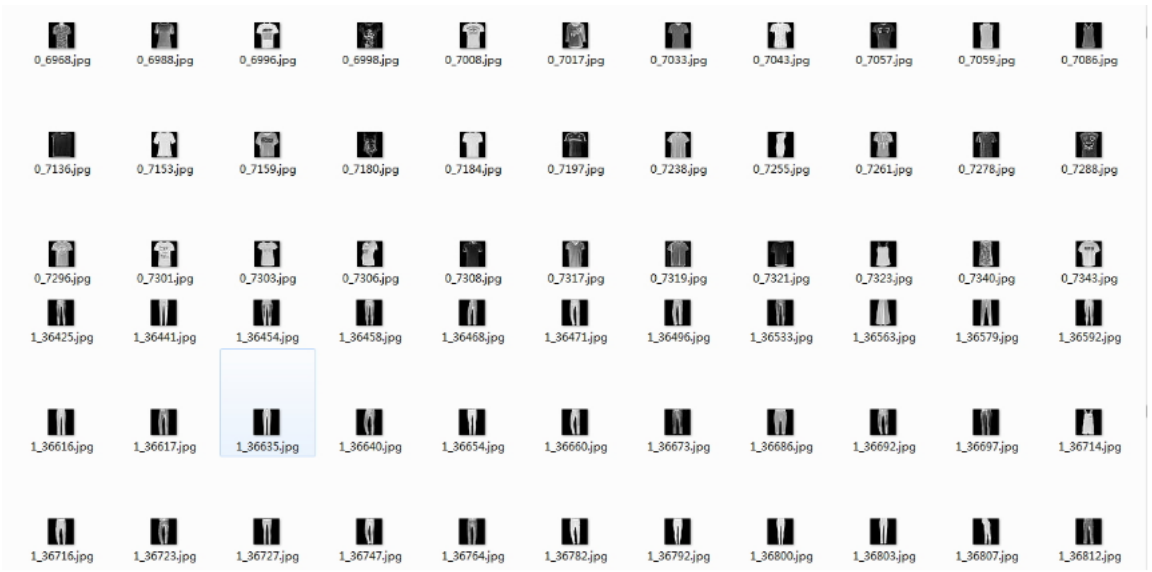
--Distributed with Spark Summary

- Step1：加载训练数据（Data Source：HDFS/Hive/Kafka/RMDB）并以DataSet/LabeledPoint形式存储。
- Step2：声明参数服务器并及配置相关信息。
- Step3：声明MultiLayer/ComputationGraph的Spark形式的包装类实例
- Step4：利用Step3中的实例进行训练、评估与保存

AI研习社

Example: Image Classification

- 开源数据集：Fashion Mnist
- 离线训练：Nvidia Telsa K80 + CUDA8.0 + Deeplearning4j + JDK1.8
- 部署应用：Spring MVC + Tomcat + JSP



Example: Image Classification

--Modeling & Train

AI研习社

```
DataTypeUtil.setDTypeForContext(DataBuffer.Type.DOUBLE);
final int numEpochs = Integer.parseInt(args[0]);
final int batchSize = Integer.parseInt(args[1]);
final String modelSavePath = args[2];
final String dataPath = args[3];
CudaEnvironment.getInstance().getConfiguration()
                // 是否允许多卡
                .allowMultiGPU(false)
                .useDevice(7)
                // 显存大小
                .setMaximumDeviceCache(11L * 1024L * 1024L * 1024L)
                // 是否允许多卡直接数据的直接访问
                .allowCrossDeviceAccess(true);
```

GPU环境声明

```
public static MultiLayerNetwork getModel(){
    MultiLayerConfiguration.Builder builder = new NeuralNetConfiguration.Builder()
                .seed(12345)
                .iterations(1)
                .learningRate(0.01).learningRateScoreBasedDecayRate(0.5)
                .weightInit(WeightInit.XAVIER)
                .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
                .updater(Updater.ADAM)
                .list()
                .layer(0, new ConvolutionLayer.Builder(5, 5)
                        .nIn(1).nOut(32).stride(1, 1).activation(Activation.LEAKYRELU)
                        .build())
                .layer(1, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
                        .kernelSize(2,2).stride(2,2).activation(Activation.IDENTITY)
                        .build())
                .layer(2, new ConvolutionLayer.Builder(5, 5)
                        .stride(1, 1).nOut(64).activation(Activation.LEAKYRELU)
                        .build())
                .layer(3, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
                        .kernelSize(2,2).stride(2,2).activation(Activation.IDENTITY)
                        .build())
                .layer(4, new DenseLayer.Builder().activation(Activation.LEAKYRELU)
                        .nOut(500).build())
                .layer(5, new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                        .nOut(10).activation(Activation.SOFTMAX)
                        .build())
                .backprop(true).pretrain(false)
                .setInputType(InputType.convolutionalFlat(28, 28, 1));
    MultiLayerConfiguration conf = builder.build();
```

卷积神经网络配置

Example: Image Classification

--Modeling & Training

AI研习社

```
DataSetIterator trainData = getData(dataPath+ "/", true, batchSize, false);
DataSetIterator testData = getData(dataPath+ "/" , false, batchSize, false);
MultiLayerNetwork model = getModel();
for( int i = 0; i < numEpochs; ++i ){
    model.fit(trainData);
    System.out.println("Epoch :" + i + " Finish");
    System.out.println("Score: " + model.score());
    Evaluation eval = model.evaluate(testData);
    System.out.println(eval.stats());
    System.out.println();
}
Evaluation eval = model.evaluate(testData);
System.out.println(eval.stats());
ModelSerializer.writeModel(model, modelSavePath, true);
```

加载数据、训练以及评估模型并保存

```
Epoch :0 Finish
Score: 0.4417036887606827

Examples labeled as 0 classified by model as 0: 766 times
Examples labeled as 0 classified by model as 1: 1 times
Examples labeled as 0 classified by model as 2: 14 times
Examples labeled as 0 classified by model as 3: 90 times
Examples labeled as 0 classified by model as 4: 11 times
Examples labeled as 0 classified by model as 5: 9 times
Examples labeled as 0 classified by model as 6: 94 times
Examples labeled as 0 classified by model as 8: 15 times
Examples labeled as 1 classified by model as 1: 938 times
Examples labeled as 1 classified by model as 2: 2 times
Examples labeled as 1 classified by model as 3: 42 times
Examples labeled as 1 classified by model as 4: 10 times
Examples labeled as 1 classified by model as 5: 1 times
Examples labeled as 1 classified by model as 6: 5 times
Examples labeled as 1 classified by model as 8: 2 times
Examples labeled as 2 classified by model as 0: 9 times
Examples labeled as 2 classified by model as 1: 1 times
Examples labeled as 2 classified by model as 2: 564 times
Examples labeled as 2 classified by model as 3: 13 times
```

训练过程

Example: Image Classification

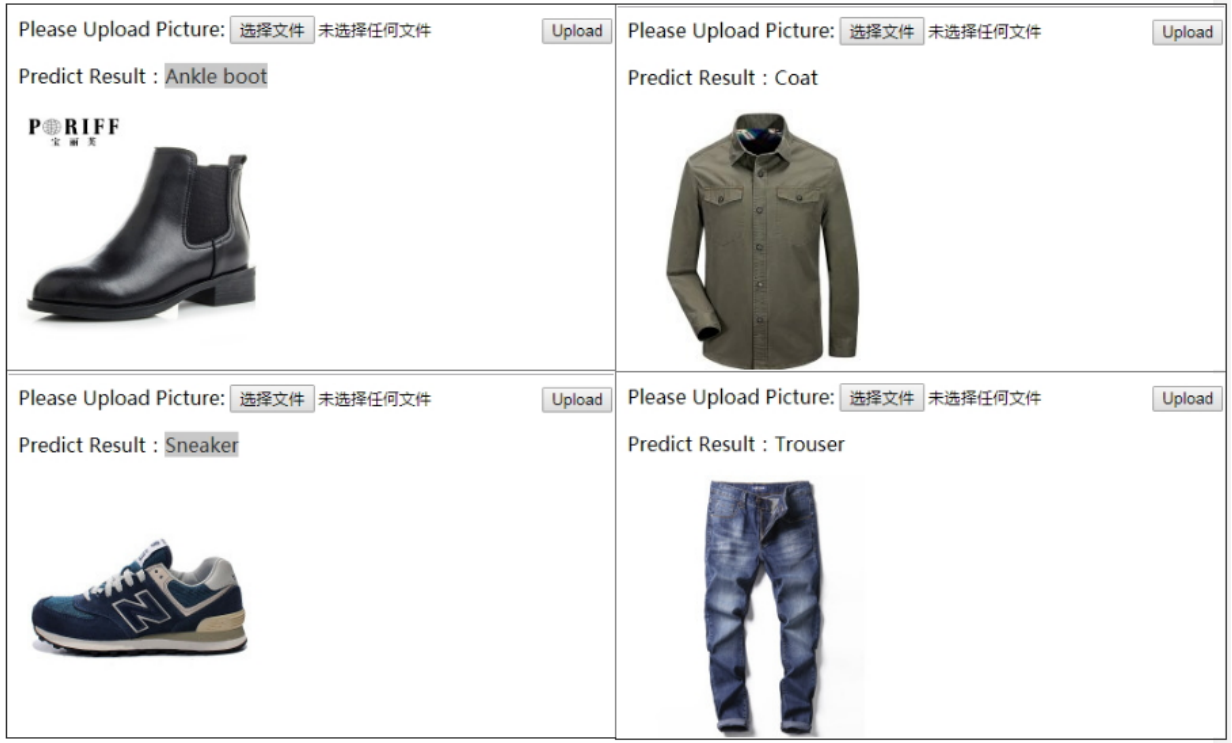
--Load & Deploy

AI研习社

图像分类的后台服务

```
@Service
public class PictureUploadService implements InitializingBean{
    private MultiLayerNetwork model;
    private NativeImageLoader imageloader;
    private Map<Integer,String> map;

    @Override
    public void afterPropertiesSet() throws Exception {
        model = ModelSerializer.restoreMultiLayerNetwork("fashionmnist/model.mod");
        imageloader = new NativeImageLoader(28, 28, 1);
        map = new HashMap<Integer,String>(){{put(0,"T-shirt");put(1,"Trouser");
                                             put(2,"Pullover");put(3,"Dress");
                                             put(4,"Coat");put(5,"Sandal");
                                             put(6,"Shirt");put(7,"Sneaker");
                                             put(8,"Bag");put(9,"Ankle boot");}};
        System.out.println("Finish Loading Model");
    }

    public String fashionReco(File pic) throws IOException{
        INDArray feature = imageloader.asRowVector(pic);
        feature = feature.div(255.0).rsub(1.0);
        int label = model.predict(feature)[0];
        return map.get(label);
    }
}
```

前端页面的效果图：

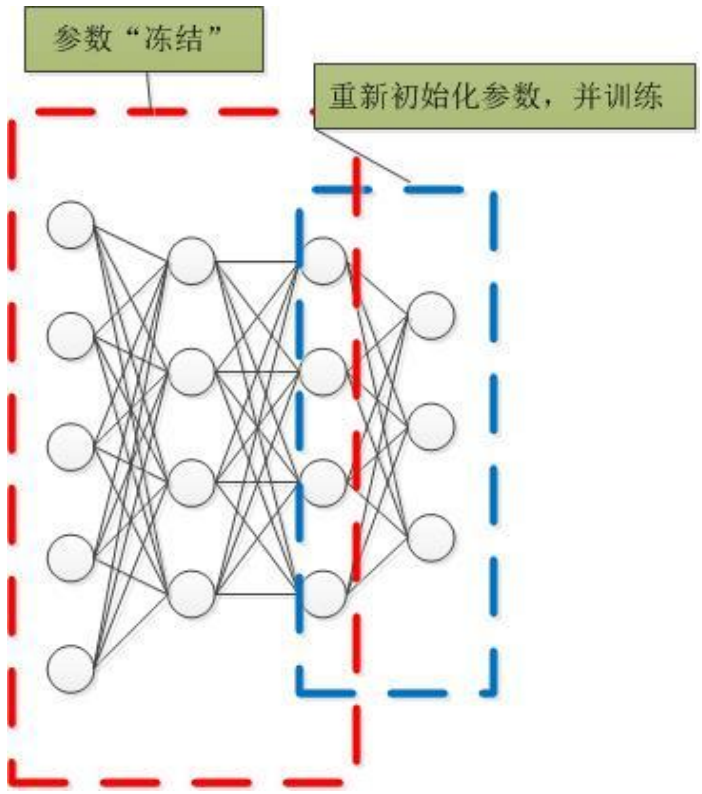


Transfer Learning in Deeplearning4j

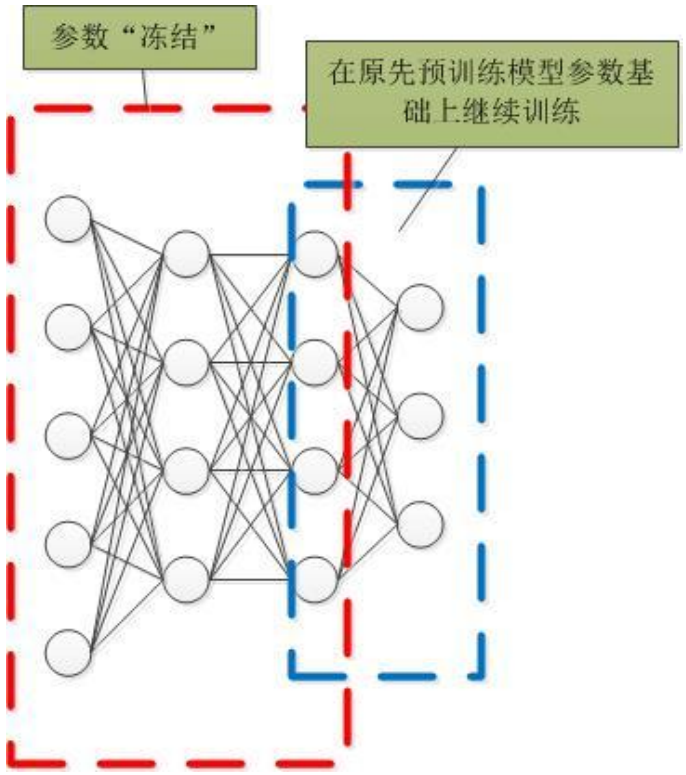
- 迁移学习一般有模型/参数迁移、特征迁移、样本迁移、关系迁移。可用于解决标注数据不足的场景下的模型训练的问题以及个性化的问题
- Deeplearning4j自0.8.0开始支持基于神经网络的迁移学习，主要支持模型/参数的迁移
- Deeplearning4j支持以下三种迁移方式
  - 重新训练预训练模型中的若干层参数
  - 继续训练预训练模型中的若干层参数
  - 修改/重构预训练模型中的部分结构并训练这些新部分间的连接参数

AI研习社

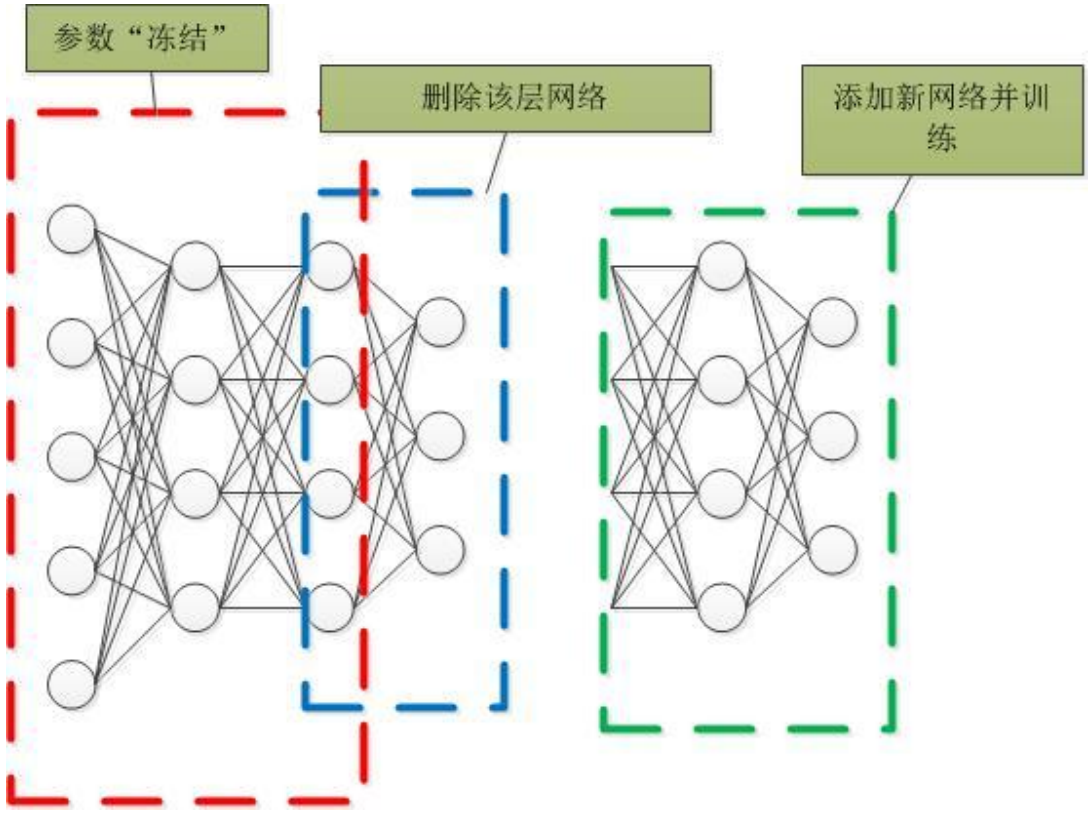
Transfer Learning in Deeplearning4j



冻结部分网络参数，
并重新训练部分



冻结部分网络参数，
并继续训练部分



改变部分网络结构并
训练

AI研习社

Reinforcement Learning in Deeplearning4j

- RL4j(https://github.com/deeplearning4j/rl4j)
- RL4j支持Deep Q-Network，A3C，Async NStepQlearning
- RL4j对OpenAI的开发环境gym的支持通过封装Java客户端来实现(https://github.com/deeplearning4j/gym-java-client)
- RL4j目前支持的强化学习都是Value-Based，离散状态空间的策略，而Policy-Based以及连续控制策略在WIP计划中
- RL4j实现的Cartpole问题以及效果：http://blog.csdn.net/wangongxi/article/details/73921083

Summary

AI研习社

- Deeplearning4j是基于Java/JVM的深度学习开源框架
- Deeplearning4j支持单机/并行/分布式的模型训练，并行的实现基于数据并行化的思想
- Deeplearning4j可以和Java生态圈中的大数据框架紧密结合，并基于Java Web的框架快速的部署上线，完成全栈式深度学习应用的开发
- Deeplearning4j生态圈中有独立的张量计算库、数据ETL框架以及其他的辅助模块

Summary

AI研习社

- 官网： https://deeplearning4j.org/
- Github： https://github.com/deeplearning4j
- Gitter英文频道：https://gitter.im/deeplearning4j/deeplearning4j
- Gitter中文频道：https://gitter.im/deeplearning4j/deeplearning4j/deeplearning4j-cn
- 官方QQ群：289058486
- 个人CSDN博客： http://blog.csdn.net/wangongxi

About AI

AI研习社

- 机器学习是AI的主要解决方案，但不是唯一方案
- 深度学习并不是万能的，对于非结构化数据：图像/文本/语音会有更出色的效果，但传统机器学习模型同样很重要
- 数据的质与量在实际的应用中共同决定了模型的泛化能力
- 转型AI同样可以从Hello World入手
- 提升AI内功必须精通原理，而不仅仅跑出demo
- 不局限于某一特定工具/框架，可以取长补短
- 迁移学习和强化学习可能代表AI的未来

AI研习社

Thanks for Watching