百度点石

比赛链接地址：http://dianshi.baidu.com/dianshi/pc/competition/22/submit

1. 数据简单处理

PathFilter

可以用于对我们的数据进行采样

1. RandomPathFilter：随机采样数据，起到了一个 shuffle 的作用
2. BalancedPathFilter：随意采样数据，并且解决数据不平衡

PipelineImageTransform

```
List<Pair<ImageTransform, Double>> pipeline = Arrays.asList(new Pair<>(
cropTransform, 0.9),
                new Pair<>(filpTransform, 0.9),
                new Pair<>(rotateTransform0, 1.0),
                new Pair<>(rotateTransform30, 0.9),
                new Pair<>(rotateTransform90, 0.9),
                new Pair<>(rotateTransform120, 0.9),
                new Pair<>(warpTransform, 0.9));
```

后面的 Double 类型是用于标注前面的图像增强方法的执行概率。

```
// {@link
org.datavec.image.transform.PipelineImageTransform.doTransform}
 @Override
protected ImageWritable doTransform(ImageWritable image, Random random
) {
    if (shuffle) {
        Collections.shuffle(imageTransforms);
    }
```

```
7.
8.      currentTransforms.clear();
9.
10.     // execute each item in the pipeline
11.     for (Pair<ImageTransform, Double> tuple : imageTransforms) {
12.         if (tuple.getSecond() == 1.0 || rng.nextDouble() < tuple.getSec
    ond()) { // probability of execution
13.             currentTransforms.add(tuple.getFirst());
14.             image = random != null ? tuple.getFirst().transform(image,
    random)
15.                     : tuple.getFirst().transform(image);
16.         }
17.     }
18.
19.     return image;
20. }
```

2. 内存管理

官方文档：https://deeplearning4j.org/docs/latest/deeplearning4j-config-memory

```
1.  -Xms2G -Xmx2G -Dorg.bytedeco.javacpp.maxbytes=10G -Dorg.bytedeco.javacp
    p.maxphysicalbytes=10G
```

3. 模型训练早停法

1. 创建 ModelSaver

用于在模型训练过程中，指定最好模型保存的位置：

1. InMemoryModelSaver：用于保存到内存中
2. LocalFileModelSaver：用于保存到本地目录中，只能保存 `MultiLayerNetwork` 类型的网络结果
3. LocalFileGraphSaver：用于保存到本地目录中，只能保存 `ComputationGraph` 类型的网

络结果

2. 配置早停法训练配置项

1. epochTerminationConditions：训练结束条件
2. evaluateEveryNEpochs：训练多少个epoch 来进行一次模型评估
3. scoreCalculator：模型评估分数的计算者
   i. org.deeplearning4j.earlystopping.scorecalc.RegressionScoreCalculator 用于回归的分数计算
   ii. ClassificationScoreCalculator 用于分类任务的分数计算
4. modelSaver：模型的存储位置
5. iterationTerminationConditions：在每一次迭代的时候用于控制

3. 获取早停法信息

```
1.  //Conduct early stopping training:
2.  EarlyStoppingResult result = trainer.fit();
3.  System.out.println("Termination reason: " +
    result.getTerminationReason());
4.  System.out.println("Termination details: " +
    result.getTerminationDetails());
5.  System.out.println("Total epochs: " + result.getTotalEpochs());
6.  System.out.println("Best epoch number: " +
    result.getBestModelEpoch());
7.  System.out.println("Score at best epoch: " + result.getBestModelScore(
    ));
8.
9.  //Print score vs. epoch
10. Map<Integer,Double> scoreVsEpoch = result.getScoreVsEpoch();
11. List<Integer> list = new ArrayList<>(scoreVsEpoch.keySet());
12. Collections.sort(list);
13. System.out.println("Score vs. Epoch:");
14. for( Integer i : list){
15.     System.out.println(i + "\t" + scoreVsEpoch.get(i));
16. }
```

4. 迁移学习

1. 获取原有的网络结构

```
 // 构造数据模型
ZooModel zooModel = VGG16.builder().build();
ComputationGraph vgg16 = (ComputationGraph) zooModel.initPretrained();
```

2. 修改模型的训练部分超参数

1. updater
2. 学习率
3. 随机数种子：用于模型的复现

```
 FineTuneConfiguration fineTuneConf = new
FineTuneConfiguration.Builder()
                .updater(new Nesterovs(0.1, 0.9))
                .seed(123)
                .build();
```

3. 修改网络架构

3.1 setFeatureExtractor

用于指定那个层以下为非 frozen 层，非冻结层。

3.2 结构更改

1. 一般只有不同网络层之间才会出现 shape 异常：需要根据异常信息调整我们的网络层结构和参数
2. `removeVertexKeepConnections` 和 `addLayer` 或者是 `addVertex` 进行网络结构的更改

迁移学习思路

1. 抛弃全连接层 -> Global average Pooling -> 替代全连接层进行分类
2. 对部分卷积层进行非冻结训练 -> 优化模型本身的特征提取能力