

1 Cuda安装

1.1 win10版本链接

1.2 linux版本链接

1.3 win7版本链接(尚未验证)

1.4 注意事项

安装检测

2 Dl4j项目依赖

2.1 CPU支持

2.2 cuda7.5支持

2.3 cuda8.0支持

3 代码配置

3.1 配置数据类型

3.2 配置GPU选项

3.3 使用GPU训练模型

1 Cuda安装

dl4j不支持8.0以上的版本，为此首先需要对cuda进行安装

1.1 win10版本链接

[cuda_8.0.44_win10-exe](#)

1.2 linux版本链接

[cuda_8.0.61_375.26_linux-run](#)

1.3 win7版本链接(尚未验证)

[百度云cuda地址](#) , [cuda_8.0.44_windows.exe](#)

1.4 注意事项

1. GTX1050ti以下版本显卡可能安装失败
2. 安装时候不要使用默认安装，应当选择自定义安装。将dirver勾选消除。（有可能cuda带的驱动版本和电脑自带的驱动版本有冲突）

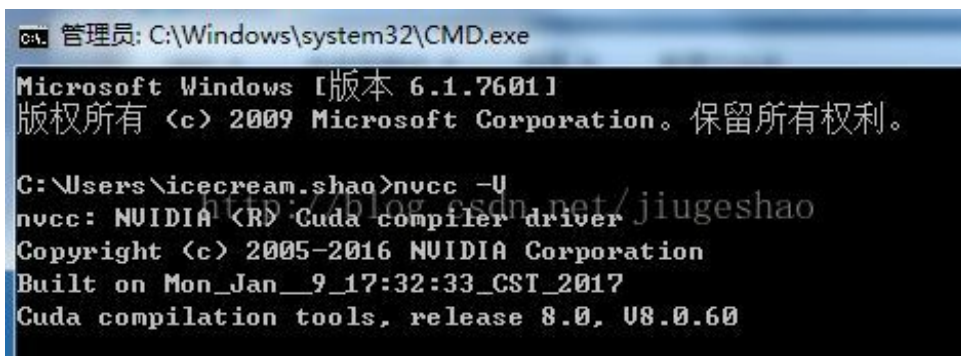
安装检测

安装完成之后，如果是windows平台则需要在命令行中输入以下命令

```
1. nvcc -V
```

如果命令行能够显示cuda的版本信息则表示安装成功。

例如博客中一个图片：



```
C:\Users\icecream.shao>nvcc -U
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2016 NVIDIA Corporation
Built on Mon_Jan_9_17:32:33_CST_2017
Cuda compilation tools, release 8.0, V8.0.60
```

2 DI4j项目依赖

```
1. <dependencies>
2.   <dependency>
3.     <groupId>org.nd4j</groupId>
4.     <artifactId>nd4j-native-platform</artifactId>
5.     <version>${nd4j.version}</version>
6.   </dependency>
7.   <dependency>
8.     <groupId>org.nd4j</groupId>
9.     <artifactId>nd4j-cuda-7.5-platform</artifactId>
10.    <version>${nd4j.version}</version>
11.  </dependency>
```

```
12.     <dependency>
13.         <groupId>org.nd4j</groupId>
14.         <artifactId>nd4j-cuda-8.0-platform</artifactId>
15.         <version>${nd4j.version}</version>
16.     </dependency>
17. </dependencies>
```

dl4j目前提供的平台依赖jar包主要有以上三个

2.1 CPU支持

```
1.     <dependency>
2.         <groupId>org.nd4j</groupId>
3.         <artifactId>nd4j-native-platform</artifactId>
4.         <version>${nd4j.version}</version>
5.     </dependency>
```

2.2 cuda7.5支持

```
1.     <dependency>
2.         <groupId>org.nd4j</groupId>
3.         <artifactId>nd4j-cuda-7.5-platform</artifactId>
4.         <version>${nd4j.version}</version>
5.     </dependency>
```

2.3 cuda8.0支持

```
1.     <dependency>
2.         <groupId>org.nd4j</groupId>
3.         <artifactId>nd4j-cuda-8.0-platform</artifactId>
4.         <version>${nd4j.version}</version>
5.     </dependency>
```

3 代码配置

3.1 配置数据类型

```
1.    DataTypeUtil.setDTypeForContext(DataType.Type.HALF);
```

这句代码提供的是GPU运算时的数据类型，还提供如下选择

```
1.    enum Type {
2.        DOUBLE, FLOAT, INT, HALF, COMPRESSED
3.    }
4.
5.    enum TypeEx {
6.        FLOAT8, INT8, UINT8, FLOAT16, INT16, UINT16, FLOAT, DOUBLE
7.    }
```

3.2 配置GPU选项

If you have several GPUs, but your system is forcing you to use just one, there' s a solution. Just

add `CudaEnvironment.getInstance().getConfiguration().allowMultiGPU(true);` as first line of your `main()` method.

```
1.    CudaEnvironment.getInstance().getConfiguration()
2.        // 如果有多个GPU则可以开启
3.        .allowMultiGPU(true)
4.
5.        //设置最大的显存分配了，取决于显卡的显存大小
6.        .setMaximumDeviceCache(2L * 1024L * 1024L * 1024L)
7.
8.        // cross-device access is used for faster model averaging over
   pcie
9.        .allowCrossDeviceAccess(true);
```

3.3 使用GPU训练模型

```
1.    // ParallelWrapper will take care of load balancing between GPUs.
2.    ParallelWrapper wrapper = new ParallelWrapper.Builder(model)
```

```
3. // DataSets预取选项。 根据实际设备的数量设置此值
4. .prefetchBuffer(24)
5.
6. // 设置数量等于或高于可用设备的数量。 x1-x2是很好的值
7. .workers(4)
8.
9. // 少量的平均可以提高性能，但可能会降低模型精度
10. .averagingFrequency(3)
11.
12. // 如果设置为TRUE，则会报告每个平均模型得分
13. .reportScoreAfterAveraging(true)
14.
15. // 可选参数，如果您的系统支持跨PCIe的P2P内存访问，则设置为false（提示：AWS不
    支持P2P）
16. .useLegacyAveraging(true)
17.
18. .build();
```

更多文档可以查看 <https://github.com/sjsdfg/deeplearning4j-issues>。

欢迎star