银河麒麟服务器操作系统 V4

Caffe 软件适配手册

天津麒麟信息技术有限公司

2019 年 5 月

目 录

1 概述

1.1 系统概述

银河麒麟服务器操作系统主要面向军队综合电子信息系统、金融系统以及电力系统等国家关键行业的服务器应用领域，突出高安全性、高可用性、高效数据处理、虚拟化等关键技术优势，针对关键业务构建的丰富高效、安全可靠的功能特性，兼容适配长城、联想、浪潮、华为、曙光等国内主流厂商的服务器整机产品，以及达梦、金仓、神通等主要国产数据库和中创、金蝶、东方通等国产中间件，满足虚拟化、云计算和大数据时代，服务器业务对操作系统在性能、安全性及可扩展性等方面的需求，是一款具有高安全、高可用、高可靠、高性能的自主可控服务器操作系统。

1.2 环境概述

服务器型号	长城信安擎天 DF720 服务器
CPU 类型	飞腾 2000+处理器
操作系统版本	Kylin-4.0.2-server-sp2-2000-19050910.Z1
内核版本	4.4.131
caffe 版本	1.0

1.3 Caffe 软件简介

Caffe 是一个清晰而高效的深度学习框架，是纯粹的 C++/CUDA 架构，支持命令行、python 和 matlab 接口；可以在 CPU 和 GPU 之间无缝切换，其作者是贾扬清，目前 caffe2 已经发布。

Caffe 的全称是 Convolutional Architecture for Fast Feature Embedding（译为：快速特征嵌入的卷积体系结构），核心语言是 C++。Caffe 的基本工作流程是设计建立在神经网络的一个简单假设，所有的计算都是层的形式表示的，网络层所做的事情就是输入数据，然后输出计算结果。比如卷积就是输入一幅图像，然后和这一层的参数（filter）做卷积，最终输出卷积结果。每层需要两种函数计算，一种是 forward，从输入计算到输出；另一种是 backward，从上层给的 gradient 来计算相对于输入层的 gradient。这两个函数实现之后，我们就可以把许多层连接成一个网络，这个网络输入数据（图像，语音或其他原始数据），然后计算需要的输出（比如识别的标签）。在训练的时候，可以根据已有的标签计算 loss 和 gradient，然后用 gradient 来更新网络中的参数。

1.4 Caffe 特点

1. 模块化

Caffe 设计之初就做到了尽可能的模块化，允许对数据格式、网络层和损失函数进行扩展。

2. 表示和实现分离

Caffe 的模型定义是用 Protocol Buffer（协议缓冲区）语言写进配置文件的，以任意有向无环图的形式，Caffe 支持网络架构。Caffe 会根据网络需要正确占用内存，通过一个函数调用实现 CPU 和 GPU 之间的切换。

3. 测试覆盖

每一个单一的模块都对应一个测试。

4. python 和 matlab 接口

同时提供两种接口。

5. 预训练参考模型

针对视觉项目，Caffe 提供了一些参考模型，这些模型仅应用在学术和非商业领域。

1.5 Caffe 架构

- 数据存储

Caffe 通过“Blobs”即以四维数组的方式存储和传递数据。Blobs 提供了一个统一的内存接口，用于批量图像（或其他数据）的操作和参数更新，Models 是以 Google Protocol Buffers 的方式存储在磁盘上的。大型数据存储在 LevelDB 数据库中。

- 层

一个 Caffe 层（Layer）是一个神经网络层的本质，它采用一个或多个 Blobs 作为输入，并产生一个或者多个 Blobs 作为输出。层有两个关键的职责，前向传播，需要输入并产生输出；反向传播，取梯度作为输出通过参数和输入计算梯度。

- 网络和运行方式

Caffe 保留所有的有向无环图，保证前向传播和反向传播正确进行。Caffe 模型是终端到终端的机器学习系统。一个典型的网络开始于数据层，结束于 loss 层。在 CPU 和 GPU 上，层会产生相同的结果。

1.6 Caffe 运行依赖（硬件）

- CPU 的选择

Caffe 支持 CPU 和 GPU 训练。如果采用 CPU 训练，CPU 支持的线程越多越好，因为 Caffe 本身显性的使用两个线程。一个线程用来读取数据，另一个线程用来执行前向传播和反向传播；如果采用 GPU 训练，则大量运算由 GPU 完成，CPU 只运行 Caffe 的两个线程，因此即使选用更多的 CPU 也无法大幅度加速训练，训练时效取决于 GPU。

- GPU 的选择

因为 Caffe 只支持 cuda（Computer Unified Device Architecture）库，而 cuda 库是 NVIDIA 显卡专用的，所以选择 Caffe 作为深度学习框架一定要选用 NVIDIA 显卡。如果电脑使用两个不同显卡的版本，则训练速度是两张低速卡一起训练的速度。

- 内存的选择

选择支持双通道的内存以及高频率的内存有利于训练，GPU 训练下，内存频率不是重要影响因素。

- 硬盘选择

Caffe 采用单独线程异步方式从硬盘中顺序读取数据，需要根据实际情况看是否考虑固态硬盘（SSD），硬盘容量和数据集密切相关。

1.7 Caffe 运行依赖（软件）

Boost 库：一个可移植、提供源代码的 C++库，作为标准库的后备，是 C++标准化进程的开发引擎之一。Caffe 采用 C++作为主开发语言，其中大量的代码依赖于 Boost 库。

GFlags 库：Google 的一个开源的处理命令行参数的库，使用 C++开发。Caffe 库采用 GFlags 库开发 Caffe 的命令行。

GLog 库：一个应用程序的日志库，提供基于 C++风格的流日志 API，Caffe 运行时的日志依赖于 GLog 库。

LevelDB 库：Google 实现的一个非常高效的 Key-Value 数据库。单进程服务，性能非常高。是 Caffe 支持的两种数据库之一。

LMDB 库：是一个超级小、超级快的 Key-Value 数据存储服务，使用内存映射文件，因此在读取数据的性能跟内存数据库一样，其大小受限于虚拟地址空间的大小。是 Caffe 支持的两种数据库之一。

ProtoBuf 库：Google Protocol Buffer，一种轻便高效的结构化数据存储格式，可用于结构化数据的串行化（序列化），适合做数据存储或 RPC 数据交换格式。可用于通信协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构

数据格式。Caffe 使用起来非常方便很大程度上是因为采用.proto 文件作为用户的输入接口。用户通过编写.proto 文件定义网络模型和 Solver。按序排列时二进制字符串尺寸最小，高效序列化，易读的文本格式与二进制版本兼容，可用多种语言实现高效的接口，尤其是 C++和 Python。这些优势造就了 Caffe 模型的灵活性与扩展性。

HDF5 库：Hierarchical Data File，一种高效存储和分发科学数据的新型数据格式，可存储不同类型的图像和数码数据的文件格式，可在不同的机器上进行传输，同时还有统一处理这种文件格式的函数库。Caffe 支持 HDF5 格式。

snappy 库：一个 C++库，用来压缩和解压缩的开发包。旨在提供高速压缩速度和合理的压缩率。Caffe 在数据处理时依赖于 snappy 库。

2 Caffe 软件适配

2.1 安装编译所需依赖

```
$ apt-get install libopencv-dev python-opencv
$ apt-get install libopenblas-dev
$ apt-get install libboost-all-dev
$ apt-get install python-pip python-dev build-essential
$ pip install --upgrade pip
$ cp /usr/local/bin/pip /usr/bin/
$ pip install protobuf
$apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler
$ apt-get install python-dev
$ apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
$ apt-get install git
```

2.2 下载 Caffe 源码

```
$ wget https://github.com/BVLC/caffe/archive/1.0.tar.gz
$ tar xvf 1.0.tar.gz
$ cd caffe-1.0/
```

2.3 修改 Makefile.config

```
$ cp Makefile.config.example Makefile.config
```

2.4 编译及安装

```
$ make all
$ make test
$ make runtest
$ make pycaffe
```

添加环境变量：

```
$ export PYTHONPATH=/opt/caffe-1.0/python/:$PYTHONPATH
```

3 Caffe 软件功能验证

3.1 验证 Caffe 安装

安装 Caffee 运行依赖：

```
$ apt-get install python-skimage
```

验证安装：

```
>>> import caffe
>>> import sys
>>> sys.path.append('/opt/caffe-1.0/python')
>>>
```

3.2 测试 mnist 数据集

1. 下载数据

```
$ ./data/mnist/get_mnist.sh
```

2. 转换数据格式

```
$ ./examples/mnist/create_mnist.sh
```

3. 修改默认参数

```
$ vim examples/mnist/lenet_solver.prototxt
```

修改为 CPU 模式。

4. 进行模型训练

```
$ ./examples/mnist/train_lenet.sh
```

5. 对训练好的模型进行测试

```
$ ./build/tools/caffe.bin test -model examples/mnist/lenet_train_test.prototxt -weights examples/mnist/lenet_iter_10000.caffemodel -iterations 100
```

测试结果如下：

```
I0525 15:46:27.309380 10297 caffe.cpp:313] Batch 89, loss = 4.23028e-05
I0525 15:46:27.447259 10297 caffe.cpp:313] Batch 90, accuracy = 0.97
I0525 15:46:27.447618 10297 caffe.cpp:313] Batch 90, loss = 0.0850854
I0525 15:46:27.524670 10297 caffe.cpp:313] Batch 91, accuracy = 1
I0525 15:46:27.524973 10297 caffe.cpp:313] Batch 91, loss = 3.26528e-05
I0525 15:46:27.602172 10297 caffe.cpp:313] Batch 92, accuracy = 1
I0525 15:46:27.602479 10297 caffe.cpp:313] Batch 92, loss = 0.000198825
I0525 15:46:27.687508 10297 caffe.cpp:313] Batch 93, accuracy = 1
I0525 15:46:27.687813 10297 caffe.cpp:313] Batch 93, loss = 0.000240378
I0525 15:46:27.764644 10297 caffe.cpp:313] Batch 94, accuracy = 1
I0525 15:46:27.764955 10297 caffe.cpp:313] Batch 94, loss = 0.000741976
I0525 15:46:27.842741 10297 caffe.cpp:313] Batch 95, accuracy = 1
I0525 15:46:27.843051 10297 caffe.cpp:313] Batch 95, loss = 0.00549091
I0525 15:46:27.844667 10302 data_layer.cpp:73] Restarting data prefetching from start.
I0525 15:46:27.921262 10297 caffe.cpp:313] Batch 96, accuracy = 0.98
I0525 15:46:27.921571 10297 caffe.cpp:313] Batch 96, loss = 0.0387434
I0525 15:46:27.999233 10297 caffe.cpp:313] Batch 97, accuracy = 0.98
I0525 15:46:27.999577 10297 caffe.cpp:313] Batch 97, loss = 0.0816107
I0525 15:46:28.075662 10297 caffe.cpp:313] Batch 98, accuracy = 1
I0525 15:46:28.075973 10297 caffe.cpp:313] Batch 98, loss = 0.0019469
I0525 15:46:28.153616 10297 caffe.cpp:313] Batch 99, accuracy = 1
I0525 15:46:28.153939 10297 caffe.cpp:313] Batch 99, loss = 0.00185568
I0525 15:46:28.154057 10297 caffe.cpp:318] Loss: 0.0278542
I0525 15:46:28.154181 10297 caffe.cpp:330] accuracy = 0.9912
I0525 15:46:28.154306 10297 caffe.cpp:330] loss = 0.0278542 (* 1 = 0.0278542 loss)
```

准确率 99%，用时 18s 左右。