|  |
| --- |
| **Edge AI教程** |
| **ML-SSD-PASCAL，来自Caffe-教程** |

以下是SSD对象检测器的教程，该对象使用Caffe对[PASCAL VOC](http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf)数据集（包含20个类）进行了培训。

本教程涉及的主题包括如何使用带有Caffe框架和DNNDK工具的PASCAL VOC 2007/2012数据集训练，量化和编译SSD，然后部署在Xilinx®ZCU102目标板上。

如果您不[熟悉](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/PDF/SSD_0_Background.pdf) SSD网络，[SSD\_0\_Background.pdf中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/PDF/SSD_0_Background.pdf)编译的SSD网络背景信息应提供有用的介绍。

本教程的结构如下：

1.）安装SSD的Caffe工具

2.）准备数据集和数据库

3.）训练SSD网络

4.）评估浮点网络

5.）量化和编译网络

6.）在ZCU102板上运行应用程序

# **1.0 Caffe for SSD的预安装注意事项**

SSD需要来自Caffe的特殊分支：

本教程使用的SSD Caffe的公共版本由[Wei Liu](http://www.cs.unc.edu/~wliu/)发布。此版本的SSD Caffe可从以下[网址](https://github.com/weiliu89/caffe/tree/ssd)获得：[https](https://github.com/weiliu89/caffe/tree/ssd)：[//github.com/weiliu89/caffe/tree/ssd](https://github.com/weiliu89/caffe/tree/ssd)

可以在诸如Docker之类的环境中安装caffe或使用虚拟python环境，但是，本教程中包含的说明仅涵盖在具有GPU卡的Ubuntu 16.04主机PC上的直接安装。

以下说明包括：

* 安装ZCU102映像
* 配置ZCU102映像
* 安装推荐的SSD Caffe（直接在主机上分发）

可以在1.1环境设置和安装一节中查看这些说明。

无论您使用哪种解决方案/环境，在1.1节之后，本教程中的内容都假定您在Ubuntu 16.04上安装了推荐的Caffe fork（直接在您的计算机上，在python虚拟环境中或在Docker映像中） Linux文件系统，您已使用以下命令将**$ CAFFE\_ROOT**环境变量设置到**〜/ .bashrc**文件中（修改caffe-ssd所在位置的文件路径）：

export CAFFE\_ROOT=/absolute\_path\_to\_caffe\_install/caffe-ssd

如果您已成功安装正确的Caffe前叉，DNNDK工具，并使用Debian拉伸图像配置ZCU102板，则继续执行数据集准备步骤。

# **1.1环境设置和安装**

您将需要一台带有Nvidia GPU卡的Ubuntu 16.04或14.04主机以及正确的CUDA，CuDNN和NCCL库。本节将进一步介绍这些依赖关系。

本教程假定您具有以下内容：

* ZCU102 1.0或更新版本
* Displayport监视器
* USB键盘和鼠标
* USB集线器带USB微转换器
* 适用于ZCU102的Deephi Demo Image：[http](http://www.deephi.com/assets/2018-10-11-ZCU102-desktop-stretch.img.zip)： [//www.deephi.com/assets/2018-10-11-ZCU102-desktop-stretch.img.zip](http://www.deephi.com/assets/2018-10-11-ZCU102-desktop-stretch.img.zip)。需要使用[Win32DiskImager](https://sourceforge.net/projects/win32diskimager/)或同等程序将此映像写入SD卡。

DNNDK用户指南ZCU102和主机x86的工具：[http](http://www.deephi.com/technology/dnndk)： [//www.deephi.com/technology/dnndk](http://www.deephi.com/technology/dnndk)

## **1.1第1部分：Deephi DNNDK和ZCU102图像设置：**

可以通过以下步骤设置Deephi工具和图像：

1.）下载[ZCU102的Deephi演示图像](http://www.deephi.com/assets/2018-10-11-ZCU102-desktop-stretch.img.zip)。

2.）从zip文件中提取ISO映像。

3.）如果使用Windows，请安装[Win32DiskImager](https://sourceforge.net/projects/win32diskimager/)并将映像刷新到16GB或更大的SD卡（本教程使用[SanDisk Ultra 80MB / s 32GB SD卡](https://www.amazon.com/SanDisk-Ultra-Class-Memory-SDSDUNC-032G-GN6IN/dp/B0143RT8OY)进行验证。

4.）使用USB键盘和监视器连接，从SD卡启动ZCU102。如果您不确定如何配置ZCU102跳线和开关，请参阅“ [reVISION入门指南”操作说明](https://github.com/Xilinx/reVISION-Getting-Started-Guide/blob/master/operating-instructions.md)。

5.）如果使用Windows，可以使用[PSCP](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/pscp.exe)将文件从ZCU102传输到主机PC。您只需将PSCP从（ref\_training /）复制到所需目录，使用Windows命令提示符cd到该目录并使用它。

6.）在Windows中打开命令提示符/终端并cd到PSCP应用程序的目录（如果使用Linux，则可以使用SCP）。

7.）将以太网电缆从ZCU102板连接到主机，并使用终端中的ifconfig获取板的以太网IP地址（我的是192.168.0.102）。

8.）在同一子网（例如192.168.0.101）上将主机的IP设置为静态。如果主机上有防病毒防火墙，则可能需要先禁用它，然后再继续执行步骤9。

9.）需要将[DNNDK工具和样品图像](http://www.deephi.com/technology/dnndk)复制到电路板上。这可以通过PSCP使用如下命令（密码是root）来完成： pscp –r c:\pathtodnndkfiles root@192.168.0.102:/root/

10.）将ZNU2文件夹和DNNDK工具中包含的公共文件夹以及步骤9中下载的样本图像从主机复制到ZCU102板。

11.）通过运行提供的install.sh脚本（./install.sh）在ZCU102上安装DNNDK工具。

12.）可以将x86主机工具复制到linux x86主机以量化/编译模型。

12.a）注意：要更改Ubuntu linux中的显示分辨率，请执行以下命令：

* xrandr – q 列出支持的mo des
* xrandr –s 1920x1080 设置模式

## **1.1第2部分：安装主机依赖项**

提供以下指南，以便在具有GPU的主机x86 Ubuntu计算机上进行直接设置：

1.）安装依赖项：

sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libboost-all-dev libhdf5-serial-dev python-numpy python-scipy python-matplotlib python-sklearn python-skimage python-h5py python-protobuf python-leveldb python-networkx python-nose python-pandas python-gflags ipython protobuf-c-compiler protobuf-compiler libboost-regex-dev libyaml-cpp-dev g++ git make build-essential autoconf libtool libopenblas-dev libgflags-dev libgoogle-glog-dev libopencv-dev libprotobuf-dev protobuf-compiler libleveldb-dev liblmdb-dev libhdf5-dev libsnappy-dev libboost-system-dev libboost-thread-dev libboost-filesystem-dev libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libboost-all-dev libhdf5-serial-dev python-numpy python-scipy python-matplotlib python-sklearn python-skimage python-h5py python-protobuf python-leveldb python-networkx python-nose python-pandas python-gflags ipython protobuf-c-compiler protobuf-compiler python-opencv python-numpy libyaml-dev libyaml-cpp-dev

2.）安装NVidia库/驱动程序。本教程使用以下配置进行了验证：

* NVidia GTX 1080ti显卡
* NVidia 390图形驱动程序

2.a）[CUDA v8.0](https://developer.nvidia.com/cuda-80-ga2-download-archive)（遵循Nvidia指令并从运行文件和补丁一起安装）。

2.b）[CuDNN v7.0.5](https://developer.nvidia.com/rdp/cudnn-archive)使用“用于Linux的cuDNN v7.0.5库”选择CUDA 8.0以下步骤用于安装CuDNN：

sudo tar -xzvf cudnn-9.1-linux-x64-v7.tgz

sudo cp cuda/include/cudnn.h /usr/local/cuda/include

sudo ln –s cuda/lib64/libcudnn\* /usr/local/cuda/lib64

sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn\*

2.c）[NCCL v1.2.3](https://github.com/NVIDIA/nccl/releases) - 使用以下步骤进行安装：

下载NCCL 1.2.3（与CUDA 8.0链接）源代码（tar.gz）。

tar –xvf nccl-1.2.3-1-cuda8.0.tar.gz

cd nccl-1.2.3-1-cuda8.0

sudo make install -j

sudo ldconfig /usr/local/cuda/lib64

2.d）接下来为hdf5创建符号链接和环境变量：

cd /usr/lib/x86\_64-linux-gnu

sudo ln –s libhdf5-serial.so.x.x.x libhdf5.so

sudo ln –s libhdf5\_serial\_hl.so.x.x.x libhdf5\_hl.so

export CPATH=“/usr/include/hdf5/serial”

最后一行也可以添加到〜/ .bashrc以在启动时配置它。

2.e）重新启动机器。

### **注意：如果在需要安装图形驱动程序时遇到问题，请使用以下说明进行安装：**

一个。首先通过以下方法删除其他安装：

sudo apt-get purge nvidia-cuda\*

sudo apt-get purge nvidia-\*

湾 使用ctrl + alt + F2输入终端会话

C。停止lightdm：sudo service lightdm stop

d。在/etc/modprobe.d/blacklist-nouveau.conf中创建一个文件，其中包含以下内容：

* 黑名单新贵
* options nouveau modeset = 0

即 然后做：sudo update-initramfs –u

F。添加图形驱动程序PPA：

sudo add-apt-repository ppa:graphics-drivers

sudo apt-get update

G。现在安装并激活最新的驱动程序（对于本教程，版本390）：

sudo apt-get install nvidia-390

## **1.1第3部分：安装SSD Caffe**

1.）克隆或下载[SSD Caffe](https://github.com/weiliu89/caffe/tree/ssd)的分布。

2.）解压缩包并制作Makefile.config.example的副本并将其重命名为“Makefile.config”。此时，您还应该使用类似“export CAFFE\_ROOT = / home / Caffe-SSD”的命令导出$ CAFFE\_ROOT环境变量（假设用户已将“caffe-ssd.zip”解压缩到文件夹“/ home / Caffe-” SSD“）。

3.）编辑makefile.config，如下所示：

3.a）取消注释第5行以启用CUDNN。

3.b）确保第28行设置CUDA\_DIR：= / usr / local / cuda（或者您的CUDA安装所在的位置）。

3.c）在第35行，有一个CUDA\_ARCH的定义。删除以sm\_20和sm\_21结尾的行（这些行与正在使用的cuda版本不兼容）。

3.d）在makefile.config中编辑第90行和第91行到以下内容：

INCLUDE\_DIRS:=$(PYTHON\_INCLUDE) /usr/local/include /usr/include/hdf5/serial

LIBRARY\_DIRS:=$(PYTHON\_LIB) /usr/local/lib /usr/lib /usr/lib/x86\_64-linux-gnu /usr/lib/x86\_64-linux-gnu/hdf5/serial

如果您使用PIP安装了许多python包，您可能还需要在Makefile.config的第64行周围编辑PYTHON\_INCLUDE变量，以将路径更改为/ usr / local / lib，如下所示： PYTHON\_INCLUDE := /usr/include/python2.7 /usr/local/lib/python2.7/dist-packages/numpy/core/include

4.）编辑makefile以确保以下内容：

第409行：添加-D\_FORCE\_INLINES

5.）此时，您将能够cd到您的caffe-master目录（这是makefile所在的目录 - 以下称为$ CAFFE\_ROOT）并运行以下命令：

make clean

make –j8

make py

6.）接下来运行类似于以下的命令以导出PYTHONPATH变量：

export PYTHONPATH= $CAFFE\_ROOT/python:$PYTHONPATH

此时，现在安装了Caffe-SSD，您可以继续阅读有关数据集准备的部分。

# **2.0准备SSD的VOC数据库**

要准备VOC0712数据库，请使用以下详细说明：

1.）下载VGG网络的预先训练的：[VGG\_ILSVRC\_16\_layers\_fc\_reduced\_deploy.prototxt](https://gist.github.com/weiliu89/2ed6e13bfd5b57cf81d6)（描述）和[VGG\_ILSVRC\_16\_layers\_fc\_reduced.caffemodel](http://cs.unc.edu/~wliu/projects/ParseNet/VGG_ILSVRC_16_layers_fc_reduced.caffemodel)（权重）文件。这些将用作我们的SSD网络的主干（特征提取层），并且由于它们是预先训练的，因此将有助于减少培训时间。

1.a）将这些文件复制到**$ CAFFE\_ROOT / models / VGGNet /**文件夹中。

2.）从以下三个链接下载PASCAL VOC数据集：

* <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar>
* <http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar>
* <http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtest_06-Nov-2007.tar>

2.a）现在创建一个名为**$ CAFFE\_ROOT / data / VOC0712**的文件夹，并将上述3个tar文件复制到该目的地。从3个档案中提取所有文件，并将它们合并到一个VOCdevkit文件夹下。

2.b）在此过程结束时，您应该有一个名为**$ CAFFE\_ROOT / data / VOC0712 / VOCdevkit**的新文件夹，其中包含VOC2007和VOC2012子文件夹。注意：此数据集包含训练，验证和测试图像以及作为.xml格式中包含的边界框位置的注释。

3.）对于下一步，您将把注释（标签）和训练图像处理成LMDB文件，然后Caffe可以直接将其用于训练过程。

3.a）对于此步骤，您需要两个shell脚本[create\_list.sh](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/dataset_files/voc0712/create_list.sh)和[create\_data.sh](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/dataset_files/voc0712/create_data.sh)，这些脚本可从此存储库中获得：[https](https://github.com/intel/caffe/tree/master/data/VOC0712)：[//github.com/intel/caffe/tree/master/data/VOC0712](https://github.com/intel/caffe/tree/master/data/VOC0712)。

3.b）这些文件已经为您下载并放入此子文件夹[dataset\_files / voc0712](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/dataset_files/voc0712)以及另一个名为[labelmap\_voc.prototxt的](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/dataset_files/voc0712/labelmap_voc.prototxt)文件中。注意：已经编辑了两个shell脚本，以便指向**$ CAFFE\_ROOT / data / VOC0712**子文件夹，以便脚本中的第一个命令如下：

export DATAPATH=$CAFFE\_ROOT/data/VOC0712

这使用先前为$ CAFFE\_ROOT设置的环境变量，您可以从Linux文件系统中的任何文件夹启动它们，但在下一条指令中，您可以将它们复制到特定目录以使指令更容易理解。

将所有这3个文件复制到**$ CAFFE\_ROOT / data / VOC0712中**。如果这些目录不存在，则必须使用**mkdir** Linux命令创建它们。

4.）执行以下命令并更改目录位置：

cd $CAFFE\_ROOT/data/VOC0712

source create\_list.sh

source create\_data.sh

cp -r examples/\* ../../examples

在遵循最近的步骤结束时，您应该看到类似于此示例中所示的内容：

注意：已创建名为**examples**的新目录以及**3个txt文件**，如以下示例所示：

注意：在**$ CAFFE\_ROOT / examples / VOC0712**中创建的此过程有两个软链接，分别存储在**$ CAFFE\_ROOT / data / VOC / VOCdevkit / VOC0712 / lmdb中**的实际LMDB数据库中，大小约为1.8GB的训练验证数据库和445MB的测试数据库，如以下示例所示：

此时，训练数据集已经准备就绪，可以用来训练SSD模型，您可以继续进行下一步训练模型。

# **3.0培训SSD模型**

## **3.0第1部分：准备SSD原型文件**

1.）使用文本编辑器[$ CAFFE\_ROOT / examples / ssd / ssd\_pascal.py](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ssd_pascal.py)打开以下文件，并更改第332行以提供可用于培训的GPU的ID号。对于多个GPU，该行将读取gpus =“0,1,2”

gpus = "0"

请注意，此python脚本将图层添加到现有VGGNet：这些图层是SSD框架图层，而您之前下载的VGGNet将是特征提取层或主干。

注意：此python脚本将使用所有必需参数为SSD网络创建[solver.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/default_solver.prototxt)，[deploy.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/deploy.prototxt)，[test.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/test.prototxt)和[train.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/train.prototxt)文件。这些都将在**$ CAFFE\_ROOT / models / VGGNet / VOC0712 / SSD\_300x300下创建**。查看原型文件并注意它们都假设将LDMB数据库放在 **$ CAFFE\_ROOT / examples / VOC0712 / VOC0712\_test\_lmdb**文件夹中，因此[数据集准备中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ssd2_VOC_dataset.md)的先前步骤确保您有一些软链接指向您的实际位置在[ssd2\_VOC\_dataset.md](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ssd2_VOC_dataset.md)描述的步骤中首次创建数据库时，可能已放置数据库。

要开始培训并创建原型文件，请使用以下命令：

cd $CAFFE\_ROOT

python examples/ssd/ssd\_pascal.py

输出快照将在以下目录下逐步创建：**$ CAFFE\_ROOT / models / VGGNet / VOC0712 / SSD\_300x300**，但此时您需要通过在终端中输入**CTRL + C**来尽早停止训练过程。 **基本原理是您可以暂停培训过程，对原型文件进行一些手动修改，以便与DPU和DNNDK工具兼容。**

## **3.0第2部分：修改SSD Prototxt文件以与DPU / DNNDK兼容**

1.）此时，您应该在$ CAFFE\_ROOT / jobs / VGGNet / VOC0712 / SSD\_300x300下有原型文本和求解器文件。您需要复制“train.prototxt”并将其重命名为“train\_test.protoxt”。以下步骤将对此train\_test.prototxt文件进行一些编辑，以使其与DPU / DNNDK兼容。如果您想跳到第7步，只需将提供的[train\_test.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/train_test.prototxt)文件复制到$ CAFFE\_ROOT / jobs / VGGNet / SSD\_300x300目录即可。

2.）接下来打开test.prototxt文件并将第一层（名为“data”）复制到train\_test.prototxt作为第二层。确保省略复制test.prototxt的第一行，即名称。您应该在“train\_test.prototxt”文件的第137和138行之间插入此图层。

3.）接下来从“test.prototxt”复制最后5层（从名为“mbox\_conf\_reshape”的层开始，该层应该在第1591行附近，在最后一层结束）并将它们附加到“train\_test.prototxt”的末尾。

4.）接下来，您需要将参数添加到刚添加的**mbox\_conf\_reshape**，**mbox\_conf\_softmax**和**mbox\_conf\_flatten**图层，以确保这些图层仅用于测试。需要添加的条目如下：

include{

phase: TEST

}

这可以插入**top：**声明之后和层其他参数之前的每个层中。

5.）在“train\_test.prototxt”中，您现在需要使用BatchNorm和Scale图层替换Normalize图层，因为DPU不支持Normalize。在原始SSD模型中，这称为“conv4\_3\_norm”。可以使用以下步骤替换图层：

5.a）删除名为“conv4\_3\_norm”的Normalize层，该层应从第972行开始并在第985行附近结束。

5.b）在“conv4\_3”和“relu4\_3”层之间插入Batchnorm层和比例层。

5.c）将名为“conv4\_3\_norm”的原型文本中的其他底层替换为“conv4\_3”（应该有三个），它们显示为bottom: "conv4\_3\_norm"。

以下插图应该有助于这些修改：

6.）接下来通过将num\_output更改为682而不是1024来修改“fc6”层。原因是DPU仅支持12位来描述参数的数量。扩张为6时，输出参数的数量将为6143，这对于DPU来说太多了。

**由于对图层的这种更改，您需要在图层本身以及它们所连接的图层中重命名fc6和fc7图层。为此，我建议只需使用“fc6\_682”查找/替换名称“fc6”，使用“fc7\_682”替换名称“fc7”。**

注意：包含[train\_test.prototxt的](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/train_test.prototxt)预修改版本供您参考。请注意，修改后的train\_test.prototxt与提供的train\_test.prototxt之间的差异将是来自detection\_output图层的output\_directory。本教程不使用此目录，但如果需要，您可以将其更改为其他位置。

7.）打开存在于同一目录中的“solver.prototxt”。该文件具有训练超参数，并且还指向正在训练的网络。在此文件中，删除前两行：

train\_net: "models/VGGNet/VOC0712/SSD\_300x300/train.prototxt"

test\_net: "models/VGGNet/VOC0712/SSD\_300x300/test.prototxt"

并用以下行替换它们：

net: "jobs/VGGNet/VOC0712/SSD\_300x300/train\_test.prototxt"

请注意snapshot\_prefix位置，该位置是训练将生成输出模型的位置。这可以根据需要进行修改，以将快照输出模型放置在所需的目录中，但是对于本教程，将快照保留在此位置并使用建议的前缀，因为这样可以更轻松地评估浮点模型而无需修改其他脚本。

snapshot\_prefix: "models/VGGNet/VOC0712/SSD\_300x300/VGG\_VOC0712\_SSD\_300x300"

快照的数量/频率是根据“solver.prototxt”中指定的参数确定的，并注意默认情况下第一个快照将在80K迭代后发生。

我还提供了一个示例[solver.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/solver.prototxt)供您参考。

8.）接下来在文本编辑器中打开[VGG\_VOC0712\_SSD\_300x300.sh](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/VGG_VOC0712_SSD_300x300.sh)脚本（应存在于$ CAFFE\_ROOT / jobs / VGGNet / SSD\_300x300目录中），并修改--solver路径以指向jobs目录中的solver.prototxt文件。

--solver="jobs/VGGNet/VOC0712/SSD\_300x300/solver.prototxt"

9.）在当前配置中，训练过程需要大约10GB的GPU内存。如果GPU没有足够的内存，则可以修改train\_test.prototxt文件，并将输入图层中的batch\_size参数更改为较小的值，例如8,4,2或1。

由于训练通常使用的batch\_size为32，如果修改此值，则还应更改solver.prototxt iter\_size参数，使iter\_size x batch\_size = 32的乘积。例如，如果将batch\_size更改为8，则应该将iter\_size更改为4，以便用于训练过程的有效批量大小为4x8 = 32。

10.）cd到$ CAFFE\_ROOT并运行以下命令：

./jobs/VGGNet/VOC0712/SSD\_300x300/VGG\_VOC0712\_SSD\_300x300.sh

这将调用./build/tools/caffe train指向新创建的“train\_test.prototxt”文件的命令。

使用GTX 1080 ti显卡的Xeon机器上的120K迭代训练过程大约需要20-30小时，使用Quadro P6000 GPU的DELL5820机器上大约需要28小时。

以下是整个培训过程的示例日志文件：[logfile\_ssd\_pascal.txt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/logfile_ssd_pascal.txt)，如果一切顺利，您将获得非常相似的内容。

该模型已经过培训，现已准备好进行评估。如果您想跳过评估​​浮点模型并直接跳到量化，那也是可能的。

# **4.0评估主机上的浮点SSD模型**

模型训练完成后，您将看到.caffemodels和.solverstates出现在[“solver.prototxt”中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/solver.prototxt)定义的snapshot\_prefix目录中。如果保持不变，则此目录位于以下位置，文件名前缀为“VGG\_VOC0712\_SSD\_300x300”+迭代编号：

snapshot\_prefix: "models/VGGNet/VOC0712/SSD\_300x300/"

此时，可以使用网络摄像头测试模型，对网络进行评分，并根据测试数据集中的单个图像对其进行评估。

我已经包含了一些可用于评估浮点网络模型的示例脚本。可以在[ref\_training / evaluation文件夹中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation)找到这些脚本。

**重要提示：这些脚本将在120K迭代训练模型上向前推进。如果您没有此模型，只需打开脚本（**[**score.sh**](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation/score.sh)**和**[**webcam.sh**](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation/webcam.sh)**）并将模型名称更改为您可用的模型。**

**重要信息：在运行脚本之前，需要将评估文件夹复制到$ CAFFE\_ROOT目录**

以下说明显示了如何运行两个不同的脚本：一个通过连续从Web摄像头抓取帧，另一个使用测试数据集对网络进行评分。

请注意，在带有一块GTX 1080ti GPU卡的Xeon机器上，网络摄像头版本大约为30FPS。

## **4.0第1部分：使用网络摄像头进行测试**

请注意，网络摄像头评估使用我创建的原型文件，该文件应与您的train\_test.prototxt匹配，但输入和输出层的差异除外。如果此prototxt与您的train\_test.prototxt不匹配，则您需要创建一个与我指定的输入/输出层匹配的新[webcam.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation/webcam.prototxt)文件，并使用train\_test.prototxt中包含的体系结构层。

1.）将网络摄像头连接到主机PC。此步骤已通过[eCON USB SEECam 3.4 MP摄像头](https://www.e-consystems.com/ar0330-lowlight-usb-cameraboard.asp)和[Logitech C920摄像头进行了测试](https://www.logitech.com/en-us/product/hd-pro-webcam-c920)。

2.）将目录更改为$ CAFFE\_ROOT，然后输入以下命令：

./evaluation/webcam.sh

注意：如果您更改了快照的位置或想要评估VGG\_VOC0712\_SSD\_300x300\_iter\_120000.caffemodel以外的模型，则还需要修改上面的脚本并指定caffemodel的正确路径。

## **4.0第2部分：对网络进行评分**

1.）为了对网络进行评分，将目录更改为$ CAFFE\_ROOT，然后输入以下命令：

./evaluation/score.sh

注意：如果您在培训期间更改了快照的位置或名称，或者想要评估其他模型，则还需要修改上面的脚本并指定模型的正确路径。

我在下面的屏幕截图中显示了我机器上的评分输出0.764852：

## **4.0第3部分：单图像评估**

1.）为了运行单个图像评估，我已经包含了可用于评估单个图像的[detect.sh](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation/detect.sh)脚本和[deploy.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation/deploy.prototxt)文件。我还从PASCAL数据集中包含了一个示例[输入图像](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/evaluation/001107.jpg)。要在单个映像上运行检测，只需将目录更改为$ CAFFE\_ROOT并运行以下命令：

./evaluation/detect.sh

此过程调用$ CAFFE\_ROOT / examples / ssd目录下的python脚本ssd\_detect.py。检测到的结果输出图像应写入$ CAFFE\_ROOT目录

注意：如果您在培训期间更改了快照的位置或名称，或者想要评估其他模型，则还需要修改上面的脚本并指定模型的正确路径。

您可以在下面看到输入图像：

结果输出图像：

。

此时，您已在主机上测试了经过训练的SSD模型，并可以继续进行[量化步骤](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ssd5_Quantization.md)。

## **4.0第4部分：其他评估**

$ CAFFE\_ROOT / examples / ssd目录中包含许多其他脚本，可以使用这些脚本。您通常可以使用-h参数调用每个脚本，以获取有关如何为该命令行构造参数的帮助。

注意：运行这些python脚本时需要注意，因为其中一些脚本会重新创建$ CAFFE\_ROOT / jobs / VGGNet / VOC0712 / SSD\_300x300目录下的原型文件，默认情况下它们也可能没有指向正确的文件caffemodel / prototxt。在尝试使用它来评估模型之前，请仔细检查每个脚本。

# **5.0量化SSD网络以实现DPU**

## **5.0第1部分：安装DNNDK工具**

在[第1节中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ssd1a_Install_Caffe.md)，我们下载了[DNNDK工具](http://www.deephi.com/technology/dnndk)并将其复制到ZCU102板上。这些工具的一部分还需要安装在主机x86机器上，以便量化和编译模型。

所需的工具包含在host\_x86工具目录下。

1.）将host\_x86文件夹及其子目录复制到主机。

2.）cd进入host\_x86目录运行以下安装主机工具：

sudo ./install.sh ZCU102

注意：本教程的目标是每个命令行的ZCU102，但可以指定其他硬件目标，如相应版本的[DNNDK用户指南所示](http://www.deephi.com/assets/ug1327-xilinx-dnndk-user-guide.pdf)。有关DNNDK工具的更多详细信息，请参阅用户指南。

## **5.0第2部分：配置文件以进行量化和编译：**

1.）[ref\_training / DNNDK\_Project中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/DNNDK_Project)包含一个示例工作空间，以显示如何调用DNNDK工具以及对prototxt文件进行必要的修改。 **将此DNNDK文件夹复制到$ CAFFE\_ROOT中**。

2.）打开[float.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/DNNDK_Project/float.prototxt)被包括在如示例[DNNDK\_Project](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/DNNDK_Project)目录。

这个[float.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/DNNDK_Project/float.prototxt)应该与你的[train\_test.prototxt](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/train_test.prototxt)几乎相同，[但](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/train_test.prototxt)有一些修改：

已修改输入层以指定校准数据的路径以及启用DNNDK中的自动测试功能。以相对方式指定了以下路径，以便在复制到$ CAFFE\_ROOT时指向正确的位置：

line 55: source: "../examples/VOC0712/VOC0712\_test\_lmdb"

line 62: label\_map\_file: "../data/VOC0712/labelmap\_voc.prototxt"

line 1706: output\_directory: "./test\_results"

line 1709: label\_map\_file: "../data/VOC0712/labelmap\_voc.prototxt"

line 1710: name\_size\_file: "../data/VOC0712/test\_name\_size.txt"

如果您希望确保它与您的原型文本基本匹配，您可以使用命令行或文本编辑器（如VSCode）来区分两者。

3.）将最新训练的模型从Caffe复制到**DNNDK\_Project**目录并将其重命名为“float.caffemodel”。假设您运行了完整的120K训练迭代，此模型应具有以下位置和名称：

"$CAFFE\_ROOT/models/VGGNet/VOC0712/SSD\_300x300/VGG\_VOC0712\_SSD\_300x300\_iter\_120000.caffemodel"

4.）通过将目录更改为**DNNDK\_Project**并运行以下命令来运行量化工具：

./decent\_ssd.sh

如果您打开脚本，您将看到以下内容，这些内容表明正在调用的命令是decent quantize并且正在向其传递许多参数，包括用于测试的GPU ID，浮点原型文本和caffemodel的位置，以及auto\_test选项，它将强制量化后的测试来测量mAP％。

work\_dir=$(pwd)

model\_dir=${work\_dir}

output\_dir=${work\_dir}/decent\_output

decent quantize \

-model ${model\_dir}/float.prototxt \

-weights ${model\_dir}/float.caffemodel \

-output\_dir ${output\_dir} -gpu 0 -auto\_test

作为参考，包含一个[体面的日志文件](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/decent_log.txt)，显示运行此命令时的控制台输出。在计算机上运行命令后，您应该会看到类似的内容。请注意，在量化过程中，损耗报告为“0”。实际上，该过程正常运行，但对于检测和分段网络，这些体面的工具无法正确报告丢失情况。我从量化过程中得到的报告精度是**0.78364**，你应该看到你的网络在这个数字的+/- 5％范围内，假设120K训练迭代使用相同的训练设置/数据集。

5.）在第4步之后，您应该在**decent\_output**目录中看到“deploy.prototxt”和“deploy.caffemodel” 。此时，您可以调用以下命令将模型编译为可在DPU上运行的可执行文件。

./dnnc\_ssd.sh

如果您打开脚本，您将看到以下内容，这些内容表明正在调用的命令是，dnnc并且正在向其传递许多参数，包括部署原型和caffemodel的位置，DPU硬件目标（在本例中为4096FA，用于ZCU102图像），cpu架构（在本例中为arm64用于cortex-A53核心），以及网名。

net=ssd

model\_dir=decent\_output

output\_dir=dnnc\_output

dnnc --prototxt=${model\_dir}/deploy.prototxt \

--caffemodel=${model\_dir}/deploy.caffemodel \

--output\_dir=${output\_dir} \

--net\_name=${net} \

--dpu=4096FA \

--cpu\_arch=arm64

作为参考，包含一个[dnnc日志文件](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/dnnc_log.txt)，显示运行此命令时控制台的输出。在计算机上运行命令后，您应该会看到类似的内容。

此时，应在**dnnc\_output**目录中创建一个elf文件，该文件可用于在ZCU102上运行模型的最后一步。

# **6.0在ZCU102上运行SSD模型**

本教程的最后一步是在ZCU102板上运行经过训练和编译的模型。为了使此步骤快速简便，包含[ZCU102\_Application](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application)目录中包含的示例应用程序。

1.）用户需要完成的第一步是将ssd编译的.elf文件复制到**模型**目录中，并将其命名为**dpu\_ssd.elf**。

注意：目录结构包含以下内容：

* [Makefile](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application/Makefile)：用于在ZCU102上构建应用程序，并将软件应用程序.elf与已编译的SSD模型链接.elf。
* [run.sh](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application/run.sh)文件：用于通过source run.sh在ZCU102上运行来在目标上执行应用程序。
* [stop.sh](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application/stop.sh)文件：用于通过source stop.sh在ZCU102上运行来停止目标上的执行。
* [model](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application/model)目录：这是您需要复制已编译的模型可执行文件并确保它名为**dpu\_ssd.elf的目录**。
* [src](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application/src)目录：此目录包含将执行以读取视频文件，调整其大小并调用DPU API以处理图像，然后在displayport监视器上使用叠加显示输出的ARM源代码。

建议详细查看源文件，因为这些将帮助您了解如何处理SSD模型的输出。它在软件中执行了几个步骤，包括softmax，它根据0到1之间的概率（在较新版本的DPU中，这可以在DPU而不是软件中执行）和非最大值抑制（NMS）来改变类输出。 ）它基本上丢弃了低概率的重叠检测。

[视频](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ref_training/ZCU102_Application/video)目录：由于github文件大小限制，此目录当前为空，但用户应添加视频文件以评估网络。

2.）有一个视频是DNNDK图像的一部分，它存在于deephi\_dnndk\_v2.08\_beta / ZCU102 / samples / adas\_detection / video / adas.avi下。将此视频文件复制到ZCU102\_Application的视频目录中。如果主机上不存在DNNDK映像，则可以在步骤3之后在ZCU102目标上执行此步骤。

3.）下一步是使用scp或pscp通过以太网连接将此应用程序文件夹复制回ZCU102板。这可以与[安装指南中](https://github.com/Xilinx/Edge-AI-Platform-Tutorials/blob/master/docs/ML-SSD-PASCAL/ssd1a_Install_Caffe.md)列出的步骤类似地完成。注意：这假定您已在主机和目标上设置了IP地址，如准则中所示。

pscp –r path\_to\_ZCU102\_Application root@192.168.0.102:/root/

4.）在ZCU102上，将目录更改为ZCU102\_Application目录并运行 make

5.）您可以输入以下命令来运行应用程序：

source run.sh

此时，您应该看到视频加载和回放，并且在SSD模型输出的检测周围绘制了边界框。请注意，提供给模型的视频来自Cityscapes数据集，因此准确性不如使用PASCAL数据库中的数据那么好（这是模型之前未见过的不同视频数据）。

控制台中可能还存在一些其他软件错误/警告，但出于演示目的，可以安全地忽略这些错误/警告。

我的ZCU102上运行的一个示例如下所示：

6.）您可以通过在终端上按**CTRL + C**并输入以下命令来停止应用程序：

source stop.sh

总之，本教程演示了如何训练SSD并从头到尾将其部署在Xilinx SoC中，包括安装用于SSD的Caffe框架，准备PASCAL VOC数据集，培训SSD网络，评估浮点模型，量化和编译该模型，最后在ZCU102板上部署应用程序。可以用来显着增强SSD模型性能的另一个步骤是修剪模型，尽管该步骤未包含在本教程中，因为它需要软件许可证。