目录结构：

Classification Model

│ End-to-end\_encrypted\_traffic\_classification\_with\_one-dimensional\_convolution\_neural\_networks.pdf

│ Readme.docx

│

├─1.DataSet

│ DataSet.txt

│

├─2.PreprocessedTools

│ │ 1\_Pcap2Session.ps1

│ │ 2\_ProcessSession.ps1

│ │ 3\_Session2png.py

│ │ 4\_Png2Mnist.py

│ │

│ └─0\_Tool

│ │ finddupe.exe

│ │

│ └─SplitCap\_2-1

│ NetworkWrapper.dll

│ PacketParser.dll

│ PcapFileHandler.dll

│ SplitCap.exe

│

└─3.TrainAndTest

└─1d\_cnn\_25+3

│ encrypt\_traffic\_cnn\_1d.py

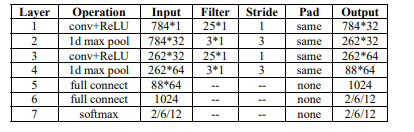
首先进行数据处理，从http://www.unb.ca/cic/datasets/vpn.html上下载对应的数据集，从下载下来的数据集中，挑选出对应类别的数据（即pcapng文件），在2.PreprocessedTools中创建1\_Pcap文件夹，在该文件下创建对应类别的文件夹（如Audio\_Vedio、Chat\_Voip）并将所有需要的数据样本放在对应类别的文件夹下。

然后可以编写脚本1\_Pcap2Session.ps1来调用开源工具splitcap根据TCP和UDP将数据集其拆分为多个文件，每个session对应一个文件。

接着调用2\_ProcessSession.ps1对上面存储起来的数据进行处理，论文中只用到了session的784个字节，因此对于超过的部分进行填充，不足的部分补0，最终存储为字节流。

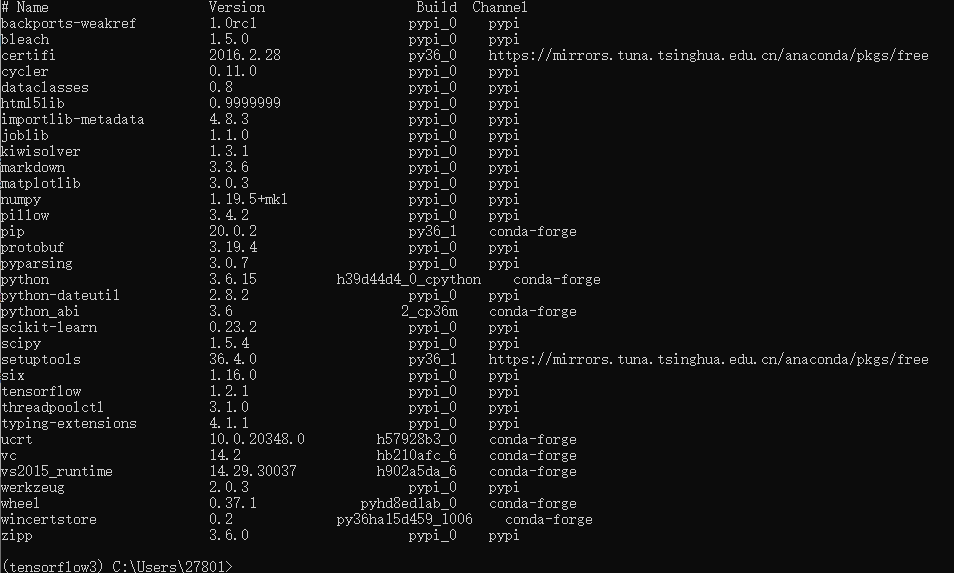
论文中的步骤是将上述会话流转为png图片，然后将PNG图片转为MNIST格式。调用3\_Session2png.py将session会话流转为png图片时，输入就是字节序，将字节转为16进制为一个单位的数组，然后将两个数组成员拼接为一个字节，字节转为Int存到numpy的array当中，然后reshape一下，让一行为28个数。调用Image.fromarray方法最终生成PNG图片。

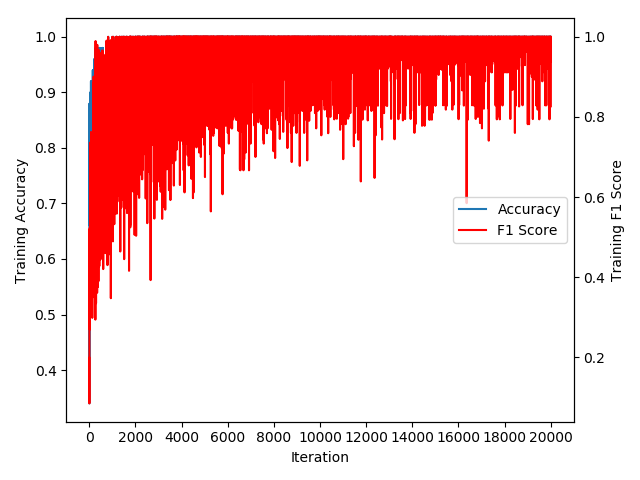
将PNG数据处理完，再调用4\_Png2Mnist.py将PNG图片转为MNIST格式，设置文件头和数据格式，生成IDX-Ubyte格式的数据集，然后便可以根据神经网络来训练，搭建的神经网络模型为encrypt\_traffic\_cnn\_1d.py，输入为（待处理数据地址，分类个数，训练次数），其中采用input\_data.read\_data\_sets进行数据读入。神经网络的参数为：



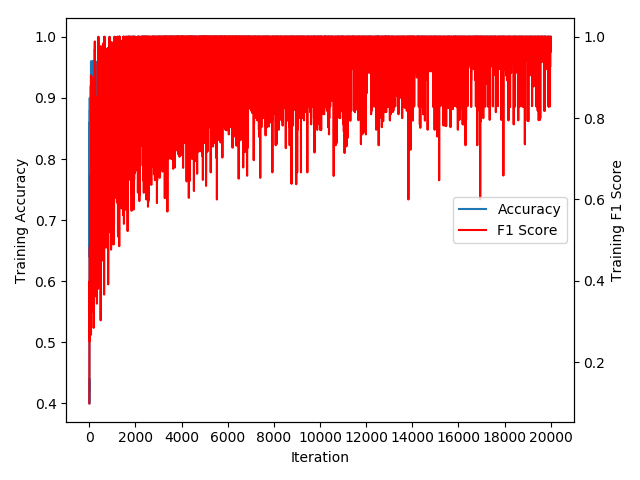
可以参考《End-to-end Encrypted Traffic Classification with One-dimensional Convolution Neural Networks》。

运行程序已安装的包：





QoS Classification test accuracy: 96.9673%



Application Classification test accuracy: 97.249%