说明文档

1. 项目概述

1.1 项目背景

目前,基于传统的规则匹配及算法的防护拦截措施无法及时发现并阻断恶意加密流量的攻击行为，可能对财产、声誉、数据造成严重损失，识别恶意加密流量已成为业界的重点难点课题，需要探索新的防护技术来提升安全防护能力。网络流量分析作为安全技术的新兴类别,是通过检测网络通信过程中的流量行为数据来分析网络中的安全威胁，帮助识别、管控风险流量，从而更快地发现漏洞，已成为各类威胁检测方案的重要组成部分。

1.2 需求描述

在 MindSpore 框架下，设计一种基于 Python 的恶意访问流量识别模型，该模型以日常业务中的正常流量和互联网公开的恶意流量为数据集,并利用相关第三方库共同实现对访问在 openEuler 上部署的业务的流量的分类、过滤的目的，从而为日常业务提供安全防护。

1.3 项目介绍

本项目是一个基于 Mindspore 框架的异常流量识别模型。该项目基于 mindspore 框架搭建了一个深度神经网络（DNN）模型，该模型先在 CICIDS2017 数据集上学习正常流量与异常流量的特征分布,然后用训练好的模型对网络流量进行行为分析，最终实现了异常流量及其攻击类型的识别与过滤。

2. 模型搭建

2.1 运行环境

（1） 硬件环境

Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

（2） 软件环境

Windows 10 操作系统 64 位
Pycharm 2022 .1
python 3.7

（3） 项目相关依赖包

pandas 1.3.5
numpy 1.21.6
mindspore 1.7

scikit-learn 1.0.2
joblib 1.1.0

2.2 目录结构

```
├── model_saved                    # 模型文件
│       ├── DNN.ckpt               # DNN 网络模型的权重参数
│       └── StandardScaler.save    # 数据标准化缩放器
│
├── data/                          # 数据文件
│   ├── processed_data/            # 预处理后的数据集
│   └── origin_data/               # 下载得到的 CICIDS2017 原始数据集文件
│
├── DataProcessing.ipynb           # 数据预处理（数据清洗及合并、特征筛选）
├── TrainDNN.ipynb                 # 模型构建及训练
├── net.py                         # 网络结构
└── apply.py                       # 应用接口
```

2.3 模型搭建流程

（1） 数据预处理（DataProcessing.ipynb）

- 数据清洗：去除原始数据中的缺失值和异常值
- 平衡数据集：调整各类流量数据在数据集中的占比
- 筛选特征：通过对数据的观察分析，筛选出 11 个典型特征进行训练
- 特征编码：对类别标签进行向量编码

最终得到含有 6 种流量类别数据的数据集。

（2） 模型构建及优化（TrainDNN.ipynb）

- 构造模型输入：划分数据集为训练集、验证集和测试集，同时对数据集进行缩放处理，然后使用 GeneratorDataset 的 API 接口，对数据集进行批处理得到最终的模型输入
- 搭建 DNN 网络结构：包含输入层、输出层和 3 个隐藏层
- 编译、训练模型：模型的损失函数选择 SoftmaxCrossEntropyWithLogits，优、化器选择 Adam，然后对模型进行训练
- 模型优化：通过观察模型训练过程中的回调结果，不断调整超参数对模型进行优化

最终得到训练好的 DNN 模型。

(3) 模型推理应用（apply.py）

- 加载训练好的 DNN 模型
- 调整输入数据的格式以满足模型输入要求
- 使用该模型对输入数据进行预测

最终得到该输入数据流量类型的识别结果。

3. 模型应用

3.1 应用说明

（1） 输入项：

含有 11 个数据的 numpy 数组，其中每个数据分别代表以下 11 个特征：

Bwd Packet Length Min	:数据包在反向上的最小值
Subflow Fwd Bytes	:子流在正向中的平均字节数
Total Length of Fwd Packets	:正向数据包的总大小
Fwd Packet Length Mean	:数据包在正向上的平均大小
Bwd Packet Length Std	:数据包反向标准偏差大小
Flow Duration	:数据流持续时间
Flow IAT Std	:两个流之间的标准差
Init_Win_bytes_forward	:在正向的初始窗口中发送的字节数
Bwd Packets/s	:每秒中后向包的数量
PSH Flag Count	:带有 PSH 的包数量
Average Packet Size	:数据包的平均大小

（2） 输出项：模型预测的流量类型

该流量识别模型通过调用 apply 文件中的 identify 接口实现，即使用时需导入该应用模块，具体使用样例如下：

```
>>> from apply import identify
>>> import numpy as np
>>> data = np.array([ 0.0, 1715.0, 1715.0, 171.5, 1547.738, 100506818.0, 24600000.0, 274.0, 0.07, 0.0, 782.941])
>>> label = identify(data)
Abnormal Traffic! Attack type: DoS Hulk
>>>
```

```
from apply import identify
import numpy as np
data = np.array([ 0.0, 1715.0, 1715.0, 171.5, 1547.738, 100506818.0, 24600000.0, 274.0, 0.07, 0.0, 782.941])
label = identify(data)
```

```
Abnormal Traffic! Attack type: DoS Hulk
```

3.2 模型效果测试：

（1） 测试代码：

```
import pandas as pd
import numpy as np
import mindspore as ms
from mindspore import Tensor
from apply import identify

dataset = pd.read_csv('./data/processed_data/dataset6classes.csv')
dataset = dataset.drop('Unnamed: 0', 1)

# 从数据集中随机抽选 20 条测试数据
np.random.seed(0)
test_data = dataset.loc[np.random.randint(low=0, high=len(dataset), size=20)]
test_data = np.array(test_data)
data = test_data[:, :11]              # 获取测试数据的特征数据
real_lables = test_data[:,-1]     # 获取测试数据的真实标签
for i in range(len(data)):
    data_flow = data[i,:]
    real_lable = class_map[int(real_lables[i])]
    np.set_printoptions(formatter={'float': '{: 0.3f}'.format})
    print( "Input_data:(Real Label is {})\n".format(real_lable), data_flow)
    # 对输入的每条数据流进行流量鉴别
    print("Output:")
    pred_label = identify(data_flow)
    print("\n")
```

（2） 输出结果:

```
Input_data:(Real Label is BENIGN)
 [ 134.000  44.000  44.000  44.000  0.000  219605.000  0.000 -1.000  4.554
  0.000  111.000]
Output:
Normal Traffic!


Input_data:(Real Label is PortScan)
 [ 6.000  0.000  0.000  0.000  0.000  25.000  0.000  29200.000  40000.000
  1.000  3.000]
Output:
Abnormal Traffic! Attack type: PortScan


Input_data:(Real Label is DoS Hulk)
 [ 0.000  343.000  343.000  57.167  3506.022  100362160.000  30200000.000
  0.000  0.060  0.000  995.333]
Output:
Abnormal Traffic! Attack type: DoS Hulk


Input_data:(Real Label is DoS Hulk)
 [ 0.000  339.000  339.000  42.375  1977.813  84949209.000  23500000.000
  251.000  0.071  0.000  852.429]
Output:
Abnormal Traffic! Attack type: DoS Hulk


Input_data:(Real Label is DoS Hulk)
 [ 0.000  0.000  0.000  0.000  0.000  1.000  0.000  274.000  0.000  0.000
```