# Spy Your Mate

Project Report — 054327 - Wireless Internet | 054323 - Internet of Things

Davide Giacomini
*Polytechnic University of Milan (PoliMi)*
*Deparment of Electronics, Information and Bioengineering*
*Milan (MI), Italy*
*August 20, 2022*

***Abstract*—After a global pandemic, video streaming calls have become the normality in every situation. Even though the major providers usually encrypt the traffic, some information can be extracted by the packets that are sent between the parties through the internet. By aggregating the packets of a conversation and analyzing some of their features through a ML algorithm, we can infer if somebody is present in front of the camera or there is nobody.**

## I. Introduction

In this report I am going to illustrate the work done for the joint project of Wireless Internet and Internet of Things.

I was required to extract some information from a video call with a provider of my choice. In particular, I was requested to analyze the traffic generated by a webcam call and, from such traffic, identify whether a person or only the background is recorded in the video. I was asked to capture the network traffic and filter all the video call packets. Subsequently, I had to extract some important features from the captured packets, and use those features to train a model of Machine Learning. Finally, I had to predict the presence of a person in the background using the trained model and compare the prediction with the ground truth. For the comparison, I had to use a Node-RED chart.

## II. Method

The first step was to choose a software for network traffic, hence I chose Wireshark[1]. I recorded three vide calls. In the first one I was alone, in the other two I was with another person (different each time). The captures duration is more or less between 300 seconds and 420 seconds. For the video calls I used my laptop (Asus) and my smartphone (Samsung Galaxy S10). The provider I used was Zoom[2], as I am comfortable with it.

I needed both the IP addresses and the UDP ports to filter the packets of the video call. In Wireshark there is a convenient window under `Statistics -> Conversations` which shows you the conversations based on the chosen protocol. Clicking on `UDP`, I ordered the conversations in order to highlight the one with the highest number of packets (and bytes) transferred. It was easy to see which one is the actual video call, because there is a difference of several orders of magnitude in number of bytes exchanged. The addresses and the ports that I extracted from Wireshark can be seen at Table I.

### A. Python Code

After extracting the addresses, I used them to filter out only the packets regarding the conversation between the parties. I used PyShark[3] to manipulate the captures. The Python scripts are used to extract the packets, organize them in chunks of half a second, and save the chunks and their features on a csv file, later used for training or testing. The scripts are called in order by the `prepare_*.py` scripts. I divided them for convenience during debugging.

The training and testing is done by two scripts called `script_ML_*.py`. The cross validation is only used to understand if the features chosen are good to make an accurate prediction, and the

[1] wireshark.org

[2] zoom.us

[3] pypi.org/project/pyshark/

| | Zoom IP Address | Zoom UDP Port | Local IP Address | Local UDP Port |
|---|---|---|---|---|
| Capture 1 | 149.137.11.203 | 8801 | 192.168.1.15 | 60235 |
| Capture 2 | 149.137.11.142 | 8801 | 192.168.1.15 | 51541 |
| Capture 3 | 149.137.10.223 | 8801 | 192.168.1.15 | 52409 |

TABLE I: Addresses for each capture.

second script is used to actually predict the results. I used the second capture (`capture2_test.csv`) to test the model and the other two captures to train it.

I incrementally added features to get a model each time more accurate, and I finally ended up with (TODO NUMBER OF FEATURES) features. Each of them represents a chunk of half a second packets. I am going to describe them below. Notice that each point represents three features, i.e. inbound packets, outbound packets and all packets:

- Average packet interval: the average time between one packet and the next.
- Average packet size: the average packet size in bytes.
- Packets number: the total number of packets.
- Bit rate: the number of bytes per half a second.
- TODO: more features

I used two different algorithms to train the model: logistic regression and random forest. The predictions are saved as `csv` files to be later used in Node-RED.

### *B. Node-RED Flow*

In Node-RED I was required to compare the ground truth with the actual prediction done in Python.

In order to do that, I read with Node-RED two different csv files (the capture file exported in csv and the prediction of python) and compared the results with the Chart of Node-RED. For comparing the results, I had to filter out the ground truth taking only the multiples of half a second. The prediction file, instead, had already been prepared with Python.

### *C. ML Training and Testing*

For the training and the testing of models (both logistic regression and random forest), I divided the work in two main parts. The former one is the cross validation, to understand more or less the accuracy of my predictions, and the latter one is the actual training and prediction with the features I chose during the cross validation. Both the cross validation and the actual training and prediction take as input the `capture*.csv` files that I prepared with PyShark. I used `scikit-learn`[4] for the ML methods.

The cross validation file (`script_ML_cross_validation.py`) takes as input the csv files thought for the training part (I chose capture 1 and 3 for the training part) and applies a K fold cross validation with $k = 5$. I set the maximum training set size to the 80% of the training data, hence the minimum validation set is 20% of the training data. I am basing the terminology on the slide number 5 from the set of slides [1]. After the cross validation, a learning curve shows the training and validation accuracy in function of the number of samples (training size).

After the cross vaildation, we can proceed with the actual training of the model and the subsequent prediction with the trained model. The steps are two: the training and then the testing. For the training, as before, the file takes as input the csv training files and trains the two models (logistic regression and random forest) with those two files. Then, for the testing, the file takes as input the csv file of capture 2 and predicts the presence or absence of a person in front of the webcam. The prediction is then exported in a csv file for Node-RED and is also compared to the ground truth, showing the confusion matrix and the accuracy of the prediction.

## III. Experimental Results

I ran several trials, but I will report here only the ones that I found interesting for the sake of the document.

[4] scikit-learn.org/stable/

### A. First attempt

I started with five features: average packets interval, average packets size, inbound and outbound packets number, total packets number.

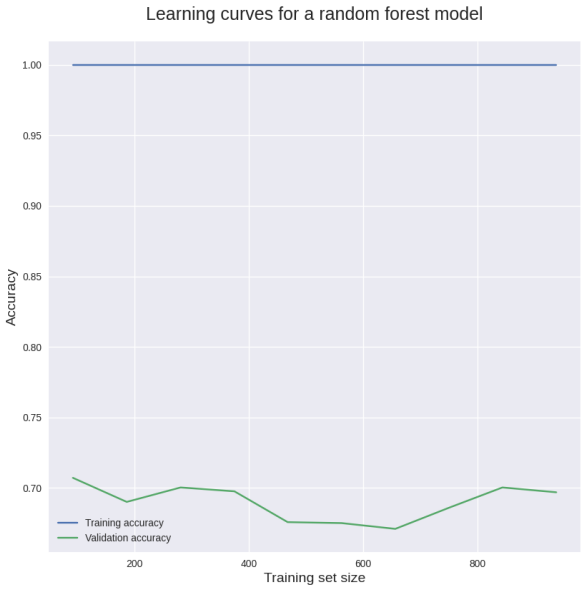
*1) Cross validation results:* The learning curve of the cross validation for both models can be seen at Fig. 1. Notice that the training curve of the random forest is always equal to 100% accuracy, and that is not exactly the best behaviour required (Fig. 1a). Nonetheless, the validation accuracy is quite consinstent with the testing results, which I will show later. In particular, the validation accuracy stands at around 70%. On the other hand, the logistic regression curve is quite "normal" (Fig. 1b). The training accuracy is about 77% and the validation accuracy is about 76%.

*2) Training and testing results:*

## REFERENCES

[1] Matteo Matteucci. Neural networks training and overfitting, 2022. URL http://chrome.ws.dei.polimi.it/images/8/8a/AN2DL_03_2122_NeuralNetwroksTraining.pdf. Accessed: August 20, 2022.


(a) Logistic Regression Curve




(b) Random Forest Curve

Fig. 1: Learning curves in the first attempt