POLITECNICO
MILANO 1863

Dipartimento Elettronica, Informazione e Bioingegneria

Iot + WI joint project Report

Spy Your Mate

Authors:
Giulio Cristofaro - 10555172
Yazan Matar - 10455438

August 31, 2022

Contents

1 Introduction

The aim of this project is to analyze the traffic generated by a videocall and, from such traffic, identify whether a person or only the background is recorded in the video, using a Machine Learning approach.

Even if the packets are encrypted, we can gather significant features as input for a Machine Learning algorithm, as we will see in the next section.

2 Pipeline

In this section we describe the various steps followed throughout the project, which are the following:

- Provider choice
- Dataset creation
- ML algorithms
- node-RED flow

2.1 Provider choice

We started the provider selection by taking into consideration the following providers: Skype, Meet, Teams and Discord.

- **Skype**
- Google Meet
- Microsoft Teams
- Discord

We analyzed the traffic streams with each provider, and we concluded that the most consistent in terms of differences of streams in the two scenarios (Background/Moving person) was **Skype**. Therefore we chose Skype for capturing the packets and for our further analysis.

2.2 Dataset creation

To create the dataset from scratch, we captured the videocall packets using **Wireshark**.
We used two different captures, one for training and one for prediction, with the same duration but different ground truth patterns, as shown here below:

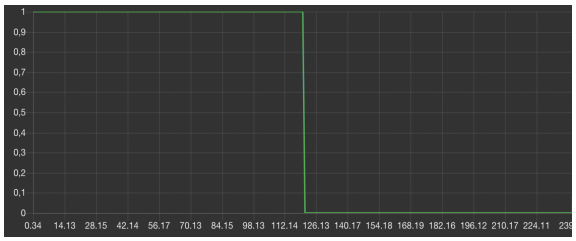

Figure 1: Training pattern

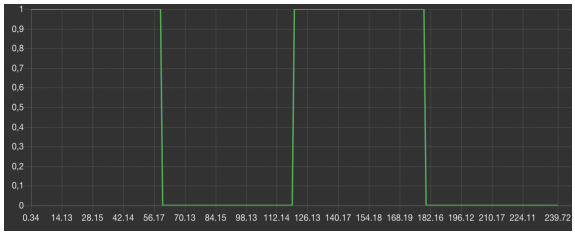

Figure 2: Prediction pattern

The captures that we then used to create our dataset and train our model have a duration of 4 minutes, which we considered sufficient to extrapolate enough data to train a successful model.

We decoded the UDP stream to an RTP stream, and then we did a first-look analysis of the stream using the statistical tools provided by Wireshark.
After filtering the RTP packets, we aggregated packets with a time window of 500ms (and 1000ms) and then we extrapolated features which we deemed as significant from our previous analysis.

The exploitable features we selected for our classification task are:

- Packet Length
- Delta Time (known as Inter-arrival Time)
- Bytes/ms
- Bit-rate (packets/second from Wireshark)

Then, for each feature we differentiated between **inbound** and **outbound** packets.

The final dataset looks like this:

DateTime	Length	Delta Time	Bytes/ms	Bit-Rate	Length (Inbound)	Delta Time (Inbound)	Bytes/ms (Inbound)	Bit-Rate (Inbound)	Length (Outbound)	Delta Time (Outbound)	Bytes/ms (Outbound)	Bit-Rate (Outbound)	Person
1900-01-01 11:53:01	866.200000	0.003475	0.866200	0.070	1022.206897	0.002684	1.022207	0.058	112.166667	0.007301	0.112167	0.012	1.0
1900-01-01 11:53:02	944.723404	0.003036	0.944723	0.188	968.038251	0.003014	0.968038	0.183	91.400000	0.003825	0.091400	0.005	1.0
1900-01-01 11:53:03	972.168539	0.003106	0.972169	0.178	992.344828	0.002937	0.992345	0.174	94.500000	0.010468	0.094500	0.004	1.0
1900-01-01 11:53:04	917.197970	0.003668	0.917198	0.197	943.041885	0.003429	0.943042	0.191	94.500000	0.011282	0.094500	0.006	1.0
1900-01-01 11:53:05	872.033493	0.003663	0.872033	0.209	898.698020	0.003318	0.898698	0.202	102.571429	0.013620	0.102571	0.007	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1900-01-01 11:56:57	446.227848	0.005112	0.446228	0.158	521.255814	0.005064	0.521256	0.129	112.482759	0.005327	0.112483	0.029	0.0
1900-01-01 11:56:58	430.455090	0.004563	0.430455	0.167	509.000000	0.004344	0.509000	0.134	111.515152	0.005453	0.111515	0.033	0.0
1900-01-01 11:56:59	502.571429	0.005822	0.502571	0.140	514.573529	0.005875	0.514574	0.136	94.500000	0.004031	0.094500	0.004	0.0
1900-01-01 11:57:00	482.838235	0.005789	0.482838	0.136	500.761538	0.005677	0.500762	0.130	94.500000	0.008218	0.094500	0.006	0.0
1900-01-01 11:57:01	508.250000	0.002309	0.508250	0.004	508.250000	0.002309	0.508250	0.004	NaN	NaN	NaN	NaN	0.0

Figure 3: Dataset

2.3 ML algorithms

For our task we tried a variety of Classification algorithms:

- Logistic Regression
- Naive Bayes
- K-Nearest Neighbour
- Decision Tree
- Random Forest
- Bagging Decision Tree
- Boosting Decision Tree
- Voting Classifier

Among all these we selected the best two for our consideration, **Decision Tree** and its Bagging version (**Bagging Decision Tree**), which have both yielded the best results with a sampling frequency of 1000 ms and a test size on the dataset split respectively of 0.4 and 0.25.

The *Decision Tree* algorithm is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules (if-else) inferred from the data features.

Bagging, an acronym for bootstrap aggregation, creates and replaces samples from the dataset. In other words, each selected instance can be repeated several times in the same sample. We seem to increase our training data with bootstraps, which are each created and then used to create a classifier model. The final prediction is the average of all predictive models.

Of course, another good result was obtained by the *Voting Classifier*, which combines conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities to predict the class labels.

Here below we can see the final results of our work considering only the two most performing algorithms, with the plot of our prediction against the ground-truth, the confusion matrices and the learning rates:

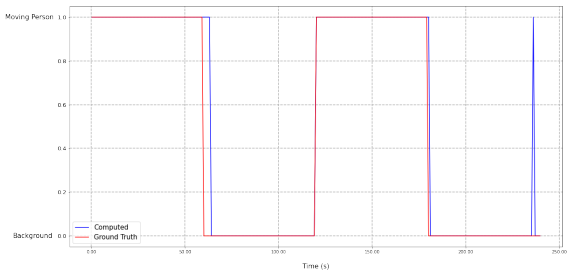

Figure 4: Decision Tree Plot

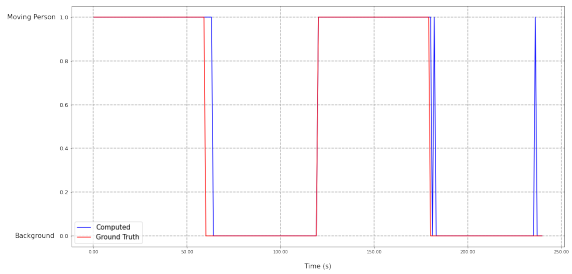

Figure 5: Bagging Plot

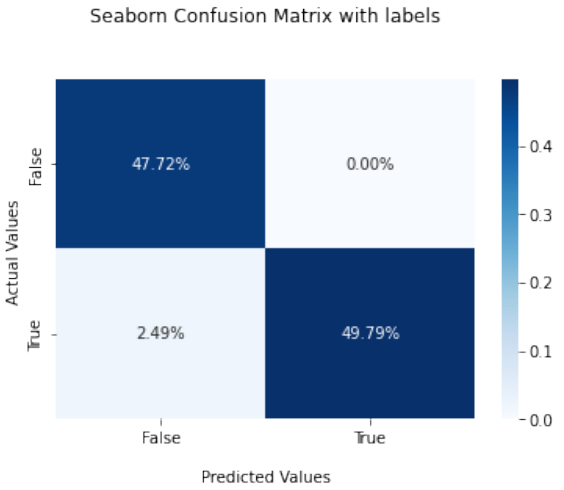

Figure 6: Decision Tree Confusion Matrix

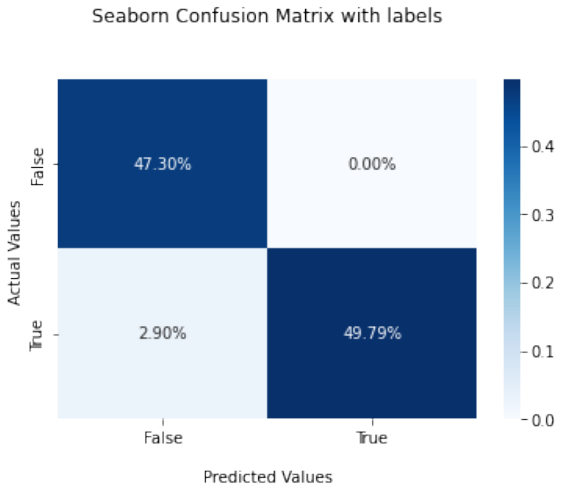

Figure 7: Bagging Confusion Matrix

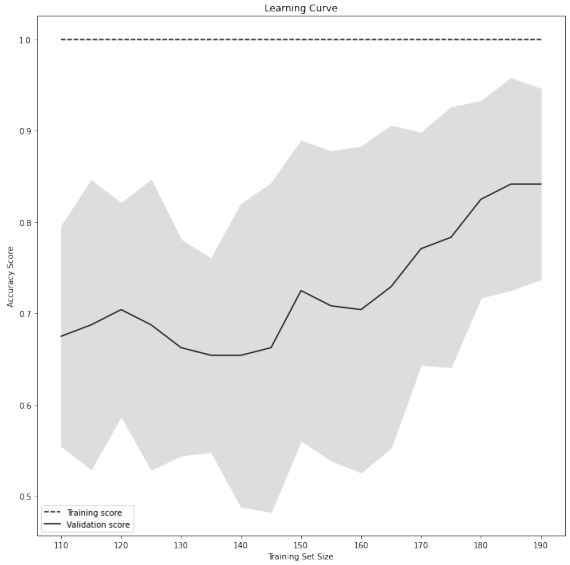

Figure 8: Decision Tree Learning Rate

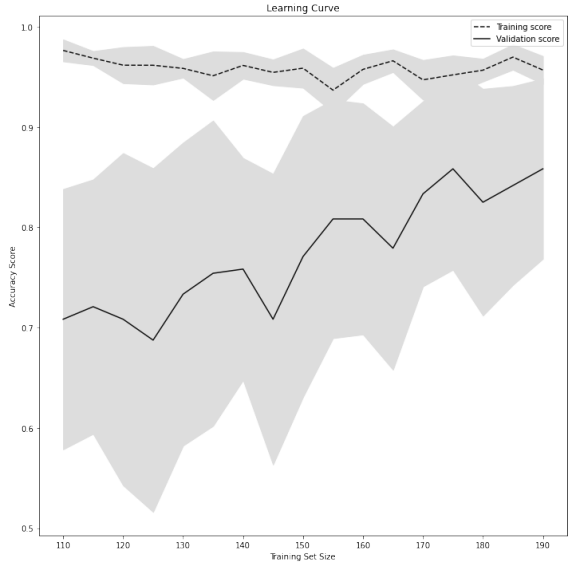

Figure 9: Bagging Learning Rate

2.4 Node-RED flow

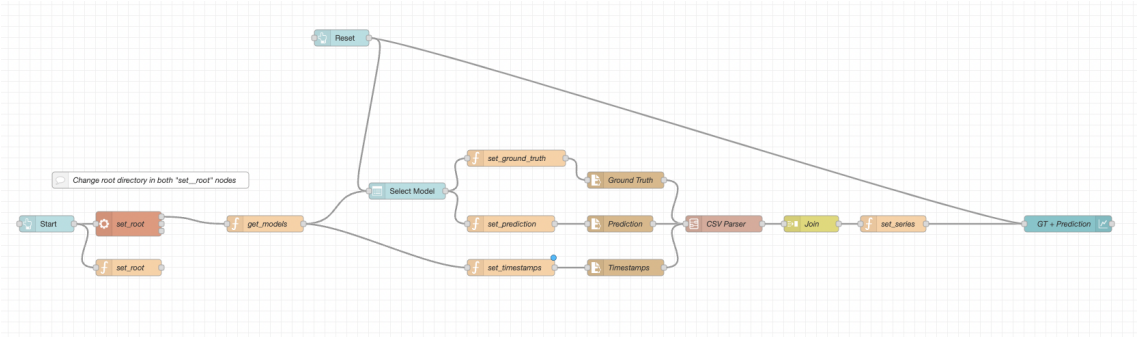

Figure 10: node-RED Flow

We used node-RED mainly to report graphically our inferences obtained through machine learning algorithms.

For each model, we exported from our colab notebook the following parameters (as CSVs):

- Ground Truth labels
- Prediction labels
- Timestamps

Then we parsed them through a custom CSV parser and compressed those values in a Series object which was needed to plot multiple lines in the Chart node.

Finally, we made a simple dashboard in which you can choose a model through a drop-down menu and it plots a chart with ground truth and prediction labels put in comparison, as shown in the picture below:

Figure 11: node-RED Dashboard

3 Conclusion

This project covered the development of a system which could differentiate between a moving person and a static background.

We believe that we obtained good results, but of course we could improve a lot our work.

For example, we could increase the duration of the captured videocall, in order to have more data to compose our training dataset.

We could also combine different videocall captures to better fit our models and to obtain predictions with higher accuracies.

Finally, we could use a neural network and add more scenarios to our datasets, so that we could differentiate even with videocalls in different places.