Fixing traffic on the edge

Creativity, Science and Innovation
Giulia Blanchetti, Giovanni Stanghellini, Gilles Vismara

January 10, 2025

1 Introduction

1.1 The issue of traffic

Intersections are some of the most critical points for road safety and particularly concerning is the increasing number of injured pedestrians, as a matter of fact in 2022 the number registered was 19,062 compared to the 16,693 registered in 2021. **Crosswalks** are becoming increasingly **unsafe**; according to the data provided by ISTAT [3], in 2022 there were 485 recorded pedestrian fatalities, 181 of which occurred on crosswalks; meanwhile, accidents caused by pedestrians' **irregular behavior** totaled 3,460, resulting in 106 deaths and over 5,000 injuries.

Adding to this is the problem of **traffic congestion**, which, especially in large cities, leads to significant air and noise pollution, alongside reduced efficiency of public transport and mobility. Therefore, improvements to the traffic system become increasingly necessary.

1.2 Existing solutions

In particular, we will be focusing on the city of Milan, where traffic represents a real problem given that, during a workday, it takes on average 27 minutes and 6 seconds to drive 10 kilometers by car. [4] In this city, the traffic system is centralized, meaning that traffic flow across multiple intersections is coordinated from a single central system that can make real-time decisions about the timing of traffic lights. At the moment, in agreement with the Local Police, the traffic system is controlled by AMAT (Agency for Mobility, Environment, and Territory) which focuses on calibrating traffic lights using **statistical models** and combinatorial optimization based on data acquisition of car and pedestrian congestion in various hours of the day.

This provides on average a good improvement compared to having fixed timings, but since this model is based on a statistic evaluation, if we focus on every single intersection, this solution may not be the optimal one.

Meanwhile, ongoing research is focusing on another type of solution that includes real-time traffic monitoring through data acquisition from microcontrollers. These devices are programmed to regularly **send images** to a server that will then analyze them. Through computer vision, the server extracts the number of cars and pedestrians queued up on the road and then proceeds to set the optimal traffic light configuration.

1.3 Our solution

An issue with the above mentioned solution is that, although functional, it comes out to be very **expensive** in terms of energy and bandwidth usage because it requires continuous image transmission to a server.

Instead of having controllers programmed to periodically send pictures out to be analyzed, our idea is to process those images directly on the device which captures them. This way instead of sending to the server an entire image, we will just have to send integers representing the count of vehicles and pedestrians.

Working on the edge, besides drastically reducing the amount of data transmitted will also avoid the **privacy** problems derived from having thousands of pictures of people taken and sent to a central data collection point. Since all the data from the picture will be analyzed on the device there will not be sensitive information saved anywhere, with this resulting in a safer and lighter way of monitoring traffic congestion. All the data obtained from the multiple microcontrollers will then be sent to a server where a reinforcement learning algorithm will elaborate them and consequently regulate the timing of stoplights in order to make the traffic flow as fluent as possible.

The choice of using **microcontrollers** also has its specific reason: aside from consuming less energy compared to more sophisticated devices, they are also cheaper, and since a city is full of intersections for which more than a single device is required for correct data acquisition, the price of each microcontroller has a major impact on the feasibility of the solution.

2 Detecting cars and pedestrians

2.1 The hardware

Currently, the methods used for detecting traffic on a road include infrared sensors, radar and lidar sensors, and, most recently, computer vision. The first kind of sensor is limited to counting moving traffic, not queued up vehicles, so it is not suitable for our aim. Radar and lidar sensors can be very precise in counting when in optimal conditions but come with a very expensive price tag (∼€1000). The final option seems to counter all the drawbacks of the previous ones at an astonishingly competitive price: small and cheap cameras with a good resolution are an easy-to-find commodity nowadays, and combined with object detection models, they are capable of recognizing both cars and pedestrians either moving or staying still. For all this reasons, **we chose computer vision** as the tool for counting cars and pedestrians.

Since we had decided to do all object detection on the edge, the choice of hardware was one of our first concerns. Existing literature on the topic often uses the Raspberry Pi or even more powerful boards [6] [5], but this comes with the disadvantage of greater costs and most importantly size, which leads to having a separate cabinet for doing the object detection, furthering the price of the installation. For these reasons we decided to look for a smaller and simpler microcontroller. After looking at various options we settled on the **XIAO ESP32S3 Sense**, which has enough PSRAM to fit the model we planned on using initially. One of the options we considered was the Arduino Nicla Sense Me, but noticed a significant lack of external RAM (only 2MB compared to 8MB of the XIAO ESP32S3 Sense).

2.2 The software

Our initial choice for object detection was the YOLOv5 model, but after a lot of trouble with the implementation, we found a very valid alternative in the **FOMO** model (Faster Objects, More Objects). FOMO is a much more lightweight model compared to YOLO, perfect for doing inference on the edge. It's strength is counting objects, which is exactly what we need, and it also comes with the nice advantage of requiring up to 30 times less memory and processing power than our initial alternative [1]. Another interesting characteristic about it is that it si very good at finding the center of objects rather than the precise bounding box, which is enough for our application.

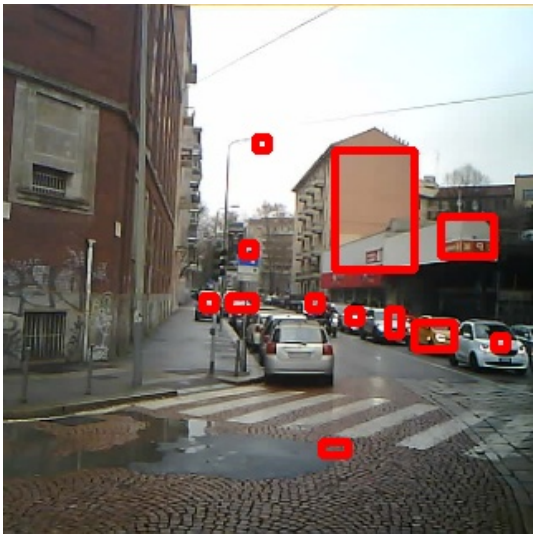

Figure 1: image labeled by our model

To build our model, we used an online platform called Edge Impulse, which allows users to train and test FOMO on our chosen dataset and create a library specifically for our microcontroller. After making a

few adjustments to record the video from the camera to check the proper workings of the model, we uploaded the code to our board and went to the street to test it.

2.3 Performance and costs

The **F1 score** of our model ended up being **88.0%** according to our testing dataset, with only 4 seconds of inference, which, keeping in mind that the minimum duration of a traffic light cycle is 10 seconds, would allow us to make at least 2 observations per cycle. On a less positive note, when testing the model in the real world, we found a few inconsistencies in the counting of cars: it often identified objects outside of the road as cars and sometimes missed a few obvious cars in the shot. We believe this to be the cause of our training process: the platform we used to train our model only allowed us to train for up to a certain amount of time on a free plan, which pushed us to trimming down our dataset. The fact that despite these limitations we managed to get a model which can give usable results, means that it's only a matter of acquiring better training data and train for a longer time to get a sound and reliable system.

Regarding the price of the microcontroller, although we paid around €10 for the one we sourced from Amazon, we found listings on Alibaba that sell the same board for as little as **€0.97** [2], provided that they are ordered in bulk. The incredible affordability of this component allows us to use multiple cameras per intersection basically for free, guaranteeing more reliable results.

Considering the ESP32 runs on 3.3V and draws 142mA of current in the worst case scenario, the total power consumptions ends up being only 11.1672W/h, which, considering the average cost of electricity to be 0.13506€/kWh, we end up finding an **annual cost of only €13** per camera.

3 Correcting traffic light timings

After collecting data on the number of vehicles and people queued up using our microcontroller, the goal becomes interpreting this data to manage traffic light states and optimize traffic flow. To achieve this, we have opted for a reinforcement learning algorithm [7] .

Reinforcement Learning is a type of machine learning where an agent learns to make decisions through trial and error to achieve the best possible outcome. The agent operates in an **environment** and follows this cyclic process:

1. **Observes** the state of the environment.
2. **Decides an action** to perform.
3. **Receives a reward** (positive or negative) that evaluates how useful the action was.
4. **Transitions to a new state** in the environment.
5. **Learns** by updating its strategy to improve future decisions.

The agent's objective is to **maximize the cumulative reward** over time by balancing exploration of new actions and exploitation of what it has already learned. Lets look at what this means for our project.

3.1 The Environment

To represent the intersections we wanted to work on we used Simulation of Urban MObility (SUMO). SUMO is an open-source and highly versatile traffic simulation platform designed to model and analyze traffic scenarios in both urban and non-urban environments. One of its most distinctive features is the ability to create fully customizable road networks, allowing users to define every aspect of the infrastructure. This includes the design of roads, lanes, intersections, traffic lights, and vehicle routes, ensuring the simulation can accurately represent specific real-world scenarios. This flexibility makes SUMO the perfect tool for optimizing traffic flow.

In our project, the state of the environment is defined by the data collected from microcontrollers. Specifically, this includes the number of vehicles and pedestrians waiting in queues at each intersection. Based on this information, the reinforcement learning agent determines the most appropriate actions

to optimize traffic flow. These actions involve adjusting the state of the traffic lights, such as changing the duration of green or red phases for different directions.

To guide the learning process, we used a reward function that evaluates the effectiveness of these actions. The reward is calculated as the negative cumulative waiting time of all vehicles in the system. By minimizing this value, the model ensures smoother traffic flow and reduces overall delays at intersections.

3.2 Training

We trained the model on a four-way intersection with three lanes on each road. We did 100 training episodes, each of which consisted of generating 10.000 cars randomly.

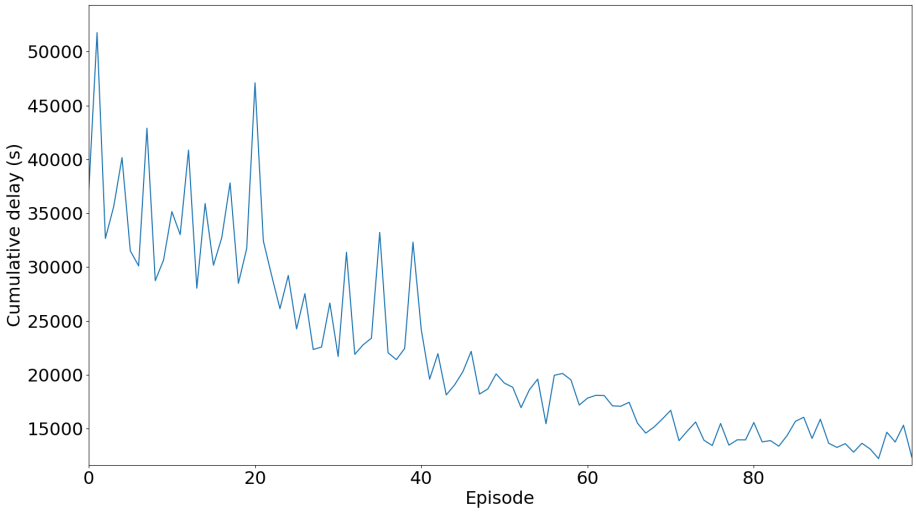

Figure 2: training: cumulative delay for each episode

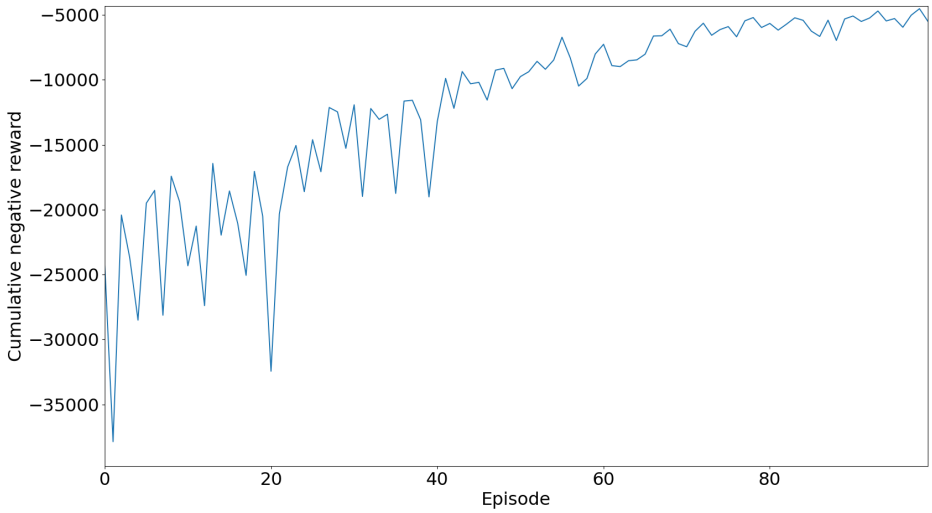

Figure 3: training: Reward for each episode

3.3 Testing

after Training the model (blue in the graphs) we tested it on different traffic scenarios and compared it to a fixed-time stoplight strategy (orange) which does not take into account the real-time state of traffic. As you can see in [fig 4] and [fig 5] the model improves greatly the traffic flow.

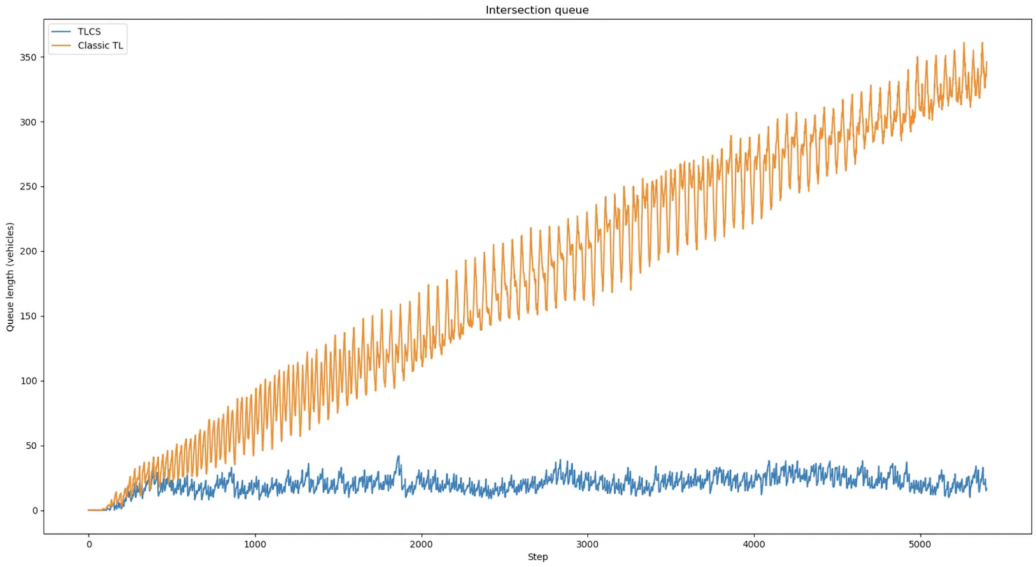

Figure 4: queue length each second for one single testing episode

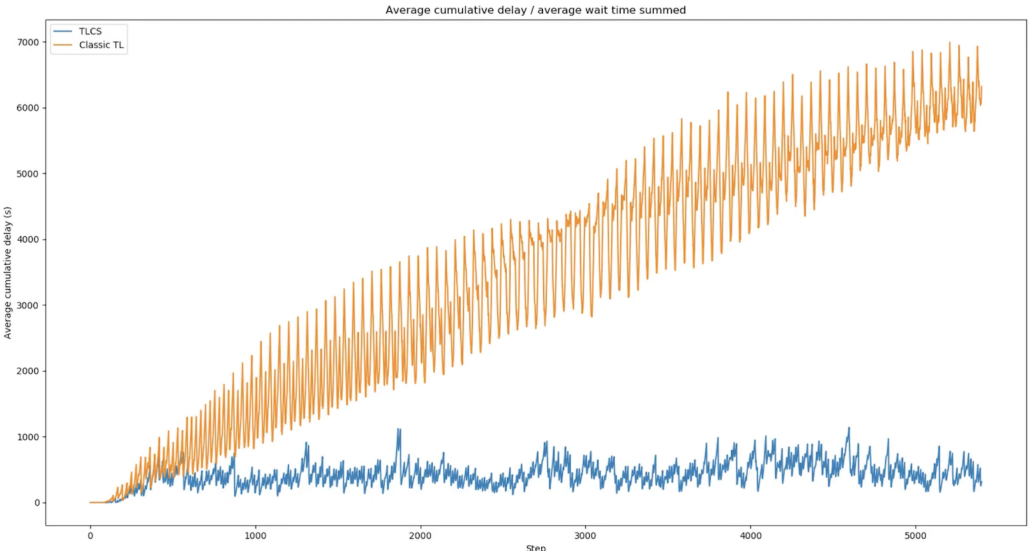

Figure 5: avg cumulative delay for one single testing episode

4 Conclusion

In conclusion, our strategy has proven promising on the side of object detection and already effective for the actual traffic control. It overcomes several limitations of the current state of the art, including privacy concerns, high costs, excessive energy consumption and high bandwidth usage. This approach offers a promising solution for traffic management in large cities, with significant potential for further improvement.

References

[1] FOMO: Object detection for constrained devices.

[2] Seeed Studio XIAO ESP32S3 Sense Alibaba listing.

[3] ISTAT road accident report.

[4] Based on a TomTom traffic index ranking.

[5] Manoj Bharadhwaj, Gitakrishnan Ramadurai, and Balaraman Ravindran. Detecting Vehicles on the Edge: Knowledge Distillation to Improve Performance in Heterogeneous Road Traffic. 2022.

[6] Asif Khan, Khurram S. Khattak, Zawar H. Khan, Thomas Aaron Gulliver, and Abdullah. Edge Computing for Effective and Efficient Traffic Characterization. 2023.

[7] Andrea Vidali. Traffic Light Control System.