Autonomous Deterrent and Alert Module

Kopacz, Nathan **Garibay, August** **Singh, Harshit**

1 Problem Definition

ADAM (Autonomous Deterrent and Alert Module) is a project focused on building a threat detection system using a Raspberry Pi equipped with a camera and a machine learning model. The system is designed to identify specific threats—in this case, detecting a person wearing a red hat—and trigger a deterrent, such as a buzzer and flashing LED. The aim is to train a machine learning model that is both efficient enough to run on a Raspberry Pi and robust enough to handle real-world situations, like noisy data. Additionally, the model must be interpretable so that we can understand and trust how it makes its decisions.

2 Related Work

Machine learning-based object detection is commonly used in surveillance systems. Models like YOLO (You Only Look Once) are optimized for real-time performance and are frequently used in applications requiring fast and accurate object detection. Such models, when combined with transfer learning, are well-suited for adapting to new tasks like detecting a red hat in this case. While performance is crucial, interpretability and robustness have become increasingly important in real-world AI deployments. Techniques like LIME (Local Interpretable Model-Agnostic Explanations) allow developers to understand how models make decisions, while robustness testing ensures that models are reliable under diverse conditions, such as when data is noisy or incomplete.

3 High-level Approach

- Dataset Creation: The first step will be to create a dataset specifically for detecting individuals wearing a red hat. The dataset will contain images captured from the Raspberry Pi itself to ensure that the model can generalize well to real-world data. Since the system will operate in a real-time environment, it is essential that the dataset be representative of various lighting conditions, angles, and distances. Additionally, diverse examples of people both with and without red hats will be included to improve the model's accuracy and reduce bias.
- Model Training: The machine learning model will likely be trained using transfer learning on a pre-trained model like YOLO. Transfer learning enables us to leverage a model that has already learned to detect a wide range of objects and adapt it to a specific task—red hat detection in this case. This approach allows us to significantly reduce the time and computational resources needed for training. The model will be optimized to run efficiently on the Raspberry Pi. Additionally, we will ensure that the model only triggers the deterrent when the red hat is detected in 3 out of 5 consecutive frames, minimizing false positives and ensuring reliability in real-world use.
- Model Analysis (Input Interpretability using LIME): Once the model is trained, it is important to understand how the model arrives at its decisions. For this, we will focus on input interpretability, using LIME. LIME is a technique that explains how individual predictions are made by perturbing the input data and observing how the predictions change. It provides an intuitive understanding of which parts of an image the model focuses on when making its decision. For example, when detecting a red hat, LIME can highlight which regions of the image the model considers important—whether it correctly focuses on the hat or is

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

mistakenly influenced by background elements. This interpretability is critical, as it allows us to trust the model's predictions and ensures that it is focusing on the right features. We will run LIME analyses on a range of test images to assess how well the model is interpreting the inputs and to identify any potential weaknesses in its decision-making process.

- In addition to ensuring that the model is accurate, we must also verify that it is robust to perturbation. In real-world environments, the quality of the input data may vary due to factors such as low lighting, camera noise. Robustness testing will involve intentionally introducing noise or distortions into the test images and evaluating the model's ability to maintain accuracy. For example, we may test how the model performs when the image is blurred, or when the lighting is uneven. Beyond just testing, we will explore training techniques that improve robustness, such as data augmentation. The goal is to ensure that the model can still reliably detect the red hat even when the data quality is not ideal

4 Timeline