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

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

Week 1 (10/19 - 10/25): Research Techniques for Model Analysis

During the first week, we will focus on researching model interpretability and robustness techniques that we want to apply to analyze the machine learning model.

- Review literature on techniques such as LIME (Local Interpretable Model-agnostic Explanations) to explain model predictions.
- Explore robustness testing strategies, including noise injection, low-light simulation, and model perturbation testing.
- Identify performance benchmarks and real-time constraints relevant to our application on the Raspberry Pi.

Week 2 (10/26 - 11/1): Create Small Dataset (Red Hat Detection)

The second week will involve creating a small dataset tailored for detecting individuals wearing a red hat.

- Capture diverse images using the Raspberry Pi in real-world environments.
- Ensure the dataset includes variations in lighting, angles, and distances to improve generalization.
- Label data appropriately and split it into training and testing sets.

Week 3 (11/2 - 11/8): Train the Model (Fine-tuning YOLO)

In the third week, we will fine-tune the YOLO model with the newly created dataset.

- Load a pre-trained YOLO model and apply transfer learning techniques.
- Modify the model to classify a single output class (red hat detection).
- Ensure the model is optimized for both accuracy and speed, considering the limitations of the Raspberry Pi.

Week 4 (11/9 - 11/15): Test the Model Using Selected Techniques

During this week, the trained model will be tested using the research techniques selected in Week 1.

- Apply interpretability techniques (e.g., LIME) to analyze the model's decision-making process.
- Perform robustness testing by introducing noisy or perturbed data and measure the model's accuracy.
- Fine-tune hyperparameters based on results from the testing.

Week 5 (11/16 - 11/22): Port the Model to the Raspberry Pi and Test Speed

Once the model is tested, we will port it to the Raspberry Pi to evaluate real-world performance.

- Deploy the model on the Raspberry Pi and assess its inference speed.
- Test the model in real-time to ensure that it meets performance requirements.
- Optimize the model further to balance speed and accuracy.

Week 6 (11/23 - 11/29): Finalize Pipeline for Model Training and Testing

This week focuses on finalizing the pipeline for training and testing models.

- Document the entire training and testing process.
- Ensure reproducibility by packaging the code and defining clear steps for running the pipeline.
- Automate the pipeline for faster iteration on future model versions.

Week 7 (11/30 - 12/6): Create New Dataset with Perturbations

In Week 7, we will create a new dataset that introduces various types of perturbations to test the model's robustness.

- Simulate adverse conditions such as blurred images, low-light environments, and occluded objects.
- Label and prepare the dataset for model retraining.

Week 8 (12/7 - 12/8): Train a New Model and Send It Through the Pipeline

During this week, we will train a new model on the perturbed dataset and test it using the previously established pipeline.

- Fine-tune the model to handle the newly introduced perturbations.
- Run the updated model through the pipeline to evaluate performance, robustness, and interpretability.

Week 8 (12/7 - 12/8): Write and Submit the Paper

Finally, we will compile the project findings and results into a formal research paper.

- Write the final paper, summarizing our methodology, experiments, and results.
- Submit the paper by the deadline (12/8).