Computer Vision Portfolio 2

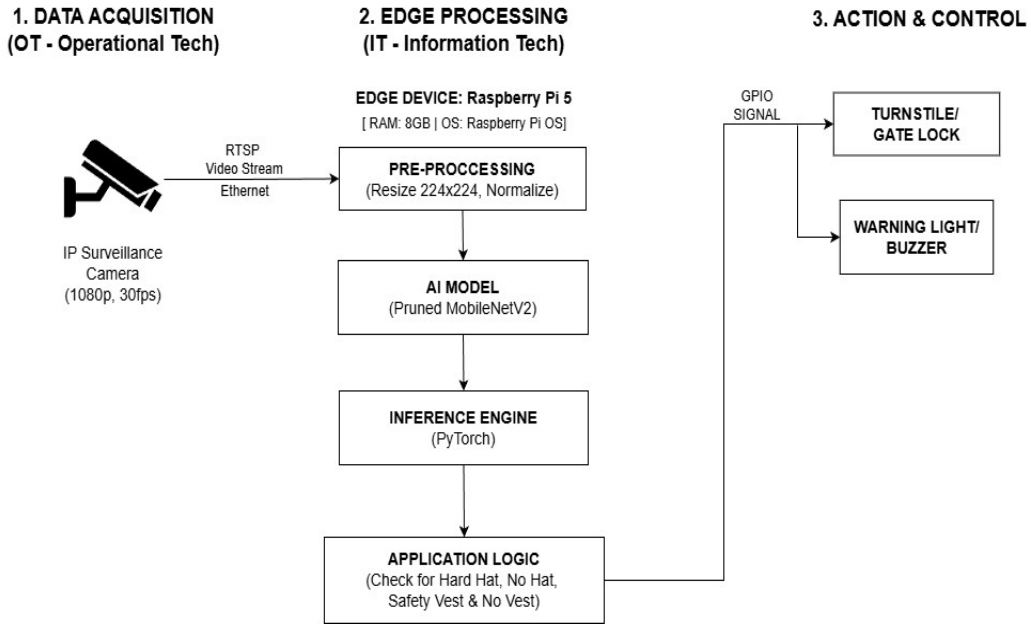
Submitted by:
Riya Biju, Harsha Sathish, Harshith Babu Prakash Babu

Assignment 2: Edge AI for Industry 4.0: Optimizing Models for On Device Inference

Task 2: Designing the Edge AI Concept

1. System Architecture Design:

The system follows a standard three-stage pipeline: **See, Think and Act**.



Hardware Components:

- **Data Acquisition:** A standard **IP Surveillance Camera** mounted at the gate. It sends video via RTSP (Real-Time Streaming Protocol) over Ethernet so that the connection is always fast and stable.
- **Edge Device:** A single-board computer (RPi 5) running Raspberry Pi OS. We chose this because it is small and its processor is powerful enough to run the AI model without needing extra graphics cards.
- **Actuator Interface:** The RPi connects directly to the **Turnstile Lock** and a **Warning Light** using wires (GPIO). This allows the code to physically block the door or flash a red light instantly.

Software Stack & CV Pipeline:

- **Preprocessing:** The incoming image is resized to 224x224 pixels and normalized (adjust the colors) to match the AI model's input requirements.
- **AI Model:** A light weight CNN pruned MobileNetV2 as the architecture.
- **Inference Engine:** The model runs on PyTorch.

- **Application Logic:** A Python script checks the model's detections. The rule is simple: IF a Person is seen and Helmet is missing or Vest is missing then the Alarm will be triggered.

Implementation Details:

MobileNetV2 model is used here because it is faster unlike models like ResNet which are heavy and slow. MobileNetV2 uses depth wise separable convolutions which breaks one big calculation into two small ones and reducing the workload. This allows the Raspberry Pi to process over 30 frames every second.

First the images are resized into 224x224 pixels so maintain the shape of the helmet. Then the colors are normalized. This keeps the math inside the AI stable when the factory lights are bright or dim.

The model is trained to detect specific objects:

1. **Helmet:** Is the person wearing a helmet?
2. **Safety Vest:** Is the person wearing a safety vest?

Logic: If the person is not wearing a helmet or vest, it triggers a alarm**!**

2. Analysis of Model Optimization Strategies:

Using MobileNet v2 model, to support Edge Deployment, we explored 3 pruning strategies and discussed a quantization approach.

Optimization 1: Pruning Heuristics Analysis:

We experimented with three different strategies to identify and remove redundant connections (weights) from the neural network.

- **Magnitude Pruning:** We removed the weights with the lowest absolute values (closest to zero) and assuming they contributed the least to the decision. This was the most effective method. Even at **68.9% sparsity** (removing around 70% of the model), our accuracy actually improved to **96.00%** (compared to the baseline of 94.64%). This proves the original model contained significant redundancy.
- **Random Pruning:** We removed weights randomly to serve as a baseline comparison. This severely damaged the model. At the same sparsity level (68.9%), accuracy dropped to **91.75%**. This confirms that which weights we remove is critical for performance.
- **Layer-wise Pruning:** We removed a fixed percentage of weights from every layer equally. This achieved a stable accuracy of **94.81%**, but it ultimately underperformed compared to the global magnitude approach.

Selected Method & Impact: Based on these results, **Magnitude Pruning** (specifically the **One-shot** approach which achieved the highest accuracy of **96.68%**).

- **Impact on Energy:** By physically deleting nearly 70% of the parameters, the Raspberry Pi has significantly less data to fetch from RAM. Memory access is the power operation on an edge device which is reducing it directly lowers thermal output and energy consumption.
- **Impact on Latency:** With fewer parameters to process, the CPU requires fewer distinct calculation cycles. This allows the inference to complete faster, reducing the time between the camera seeing a person and the gate locking.

Optimization 2: Post-Training Quantization:

While our current implementation focuses on pruning, we propose adding Post-Training Quantization as a complementary optimization technique. Quantization compresses the model by reducing the precision of numerical values by converting weights from 32-bit floating-point (Float32) to 8-bit integer (Int8) format.

Working Principle: Instead of storing precise decimal values (e.g., 0.00234567), the method maps weights to simple integer values (0-255) while maintaining a scaling factor to preserve model accuracy.

Expected Impact:

- **Model Size**: Reduces storage requirements from approximately 9MB to 2.3MB (4x compression) which is critical for deployment on resource-constrained edge devices
- **Energy Consumption**: Integer arithmetic consumes significantly less power than floating-point operations. Additionally, transferring 8-bit values from memory requires 75% less energy compared to 32-bit values, directly reducing power consumption and heat generation
- **Inference Latency**: Expected to deliver the largest performance improvement, potentially tripling inference speed by utilizing the Raspberry Pi's efficient integer processing capabilities
- **Accuracy**: MobileNetV2 architecture demonstrates resilience to quantization, with accuracy degradation typically limited to less than 1%, representing an acceptable trade-off for substantial performance gains

3. Risk Assessment: The Dangers of Aggressive Compression

Risk 1: The Small Object Detection Failure:
If we prune the model too much like removing 95% of weights. the model may lose its ability to generalize. While a model might still easily detect large objects like a "Safety Vest" but it may fail to recognize smaller accessories like a "Helmet".

Risk 2: Loss of Robustness:
A heavily compressed model suffers from a reduced capacity to learn invariant features and effectively making its perception thinner and less adaptable. While a full-sized model retains enough redundancy to handle variable conditions, a hyper-compressed model might function perfectly under better lighting but fail unpredictably when environmental factors change. In a safety system, such inconsistency is critical. If the gate lock behaves randomly then, there is a chance that the workers may attempt to bypass the safety protocols entirely.