IoT and Edge Computing (CS3100)
Project Report

Occupancy Detection

Submitted by

Lohith R Gowda

1RVU22CSE093

School of Computer Science

RV University

Submitted to

Chandramouleeswaran Sankaran

Professor

School of Computer Science

RV University

14-11-2024

IoT and Edge Computing (CS3100)
Project Report

1. Abstract

This project applies machine learning for real-time occupancy detection using environmental factors like temperature, humidity, light, and CO_2 levels. The model leverages data from the UCI Machine Learning Repository's Occupancy Detection Dataset, with ground truth data obtained via time-stamped images. Our goal is to improve occupancy monitoring in indoor spaces, which has applications in energy management, smart building systems, and automated facility control. The outcomes demonstrate the model’s accuracy in detecting occupancy with minimal sensor input.

2. Introduction

Project Background and Relevance

Monitoring occupancy in real-time enhances energy efficiency and optimizes building management systems. Traditional occupancy detection methods involve video monitoring, which can be intrusive and expensive. This project explores an alternative approach that uses environmental data to detect occupancy, thus offering a privacy-friendly, cost-effective solution for smart buildings and IoT applications.

Objectives

- Implement a machine learning model to classify room occupancy based on environmental sensor data.
- Use features such as temperature, humidity, light, and CO_2 concentration to determine occupancy status.
- Evaluate the model's accuracy and integrate the model with ESP32 to detect occupancy.

3. System Overview

System Architecture

The system comprises environmental sensors connected to an ESP32 microcontroller, which processes the data using a trained occupancy detection model.

- **Data Collection:** Environmental data is gathered from connected sensors.
- **Model Inference:** The ESP32 processes sensor data using the on-device model to predict occupancy.
- **Response:** The ESP32 outputs occupancy status, which could be used to control building systems like lighting or HVAC.

4. Details of the Dataset:

The UCI Occupancy Detection Dataset includes five features: temperature, humidity, light, CO_2 levels, and humidity ratio. Each data entry is labeled with occupancy status (1 for occupied, 0 for unoccupied), enabling binary classification. Ground-truth labels are verified through time-stamped images taken every minute.

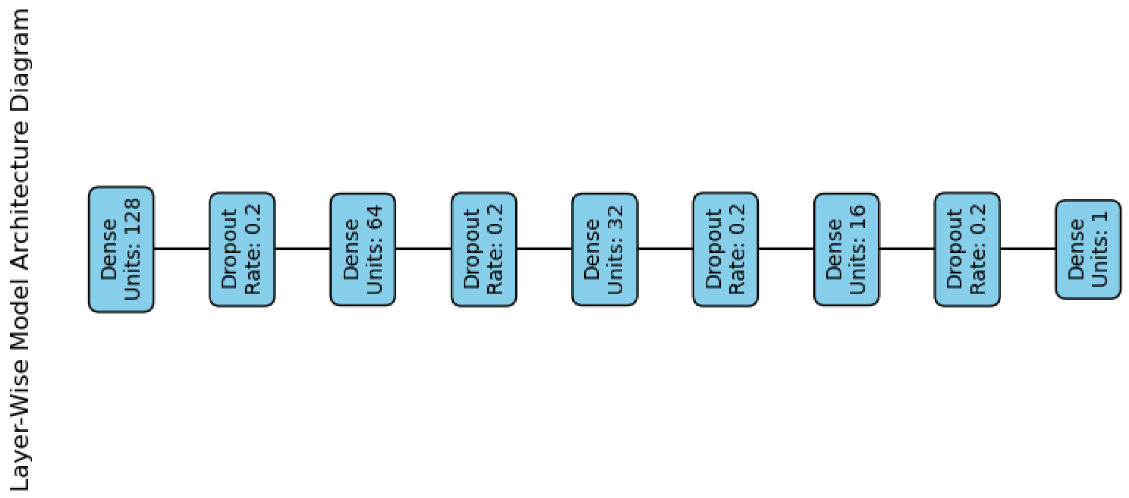
5. Model Design:

The model is a neural network designed in Python, converted to a lightweight TFLite format for deployment on ESP32.

Architecture Overview:

- **Input Layer:** Receives 4 inputs (e.g., Temperature, Humidity, Light, CO_2).
- Dense Layers: Each dense layer has ReLU activation, with dropout layers in between to prevent overfitting.
- Output Layer: A single neuron with sigmoid activation for binary classification.

Diagram:



6. Requirements

Hardware

- ESP32 microcontroller

Software

- Python with scikit-learn and TensorFlow for training and model conversion
- Arduino IDE or PlatformIO for deploying and testing the model on ESP32
- TensorFlow Lite Micro for ESP32 compatibility

7. Methodology

Data Preprocessing

Data preprocessing involved normalization and cleaning to prepare the dataset for optimal model training. Feature selection focused on environmental variables most predictive of occupancy.

Model Training and Conversion

The model was trained on environmental data to predict occupancy status. After achieving satisfactory performance, the model was converted to TensorFlow Lite format. This conversion enables the model to run on the ESP32 using TensorFlow Lite Micro, designed for low-memory devices.

Deployment to ESP32

The converted TensorFlow Lite model was deployed to the ESP32. The ESP32 continuously reads sensor data, processes it with the model, and outputs occupancy predictions.

8. Implementation and Testing

Key code excerpts include TensorFlow Lite model loading, sensor data input handling, and prediction output on the ESP32.

Testing

- **Accuracy Evaluation:** The model's prediction accuracy was evaluated on test data before deployment.

- **Real-Time Inference on ESP32:** Response times were measured on the ESP32 to ensure the model could handle real-time data inputs effectively.

9. Results

Data Output

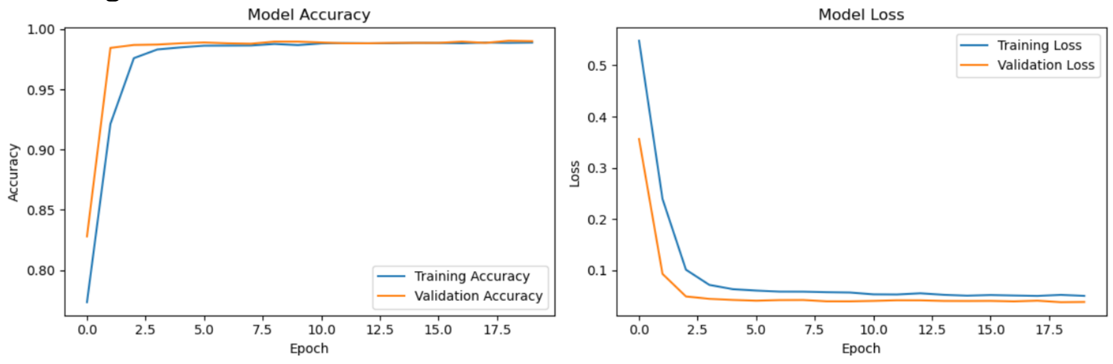
The model successfully classified occupancy in real-time on the ESP32, achieving an accuracy of 98.69% during tests with environmental data inputs.

Performance Notes

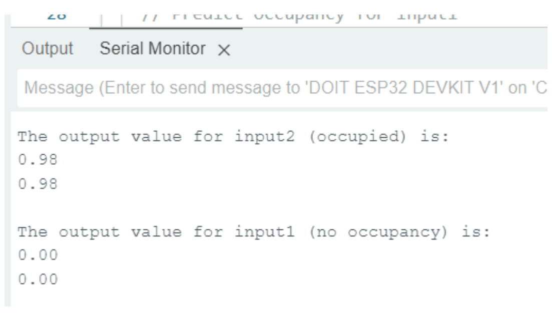
- **Latency:** Minimal delay in predictions on ESP32, suitable for real-time applications.
- **Efficiency:** The model operated within the memory and power constraints of the ESP32, demonstrating low power consumption ideal for IoT settings.

Visual Output

- **Training Results:**



- **ESP32 Serial Monitor Output:**


Output
Serial Monitor
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'C

```
The output value for input2 (occupied) is:
0.98
0.98

The output value for input1 (no occupancy) is:
0.00
0.00
```

10. Conclusion

In this project, I worked on integrating machine learning with IoT by using a dataset for occupancy detection to train a neural network model. I then converted the model into TensorFlow Lite format to run on an ESP32 microcontroller. This allowed the ESP32 to make real-time predictions based on sensor data, detecting whether a space was occupied or not. Through this project, I learned how to optimize machine learning models for low-power devices and gained hands-on experience with IoT system design. Overall, it deepened my understanding of combining AI and IoT for practical applications.

****