AI-Powered Smart Glasses to Assist the Blind

FYP–I REPORT
BSCS Fall 2024

Name: Dinesh Dhanjee
21K-3459

Name: Ameer Hamza
21K-3463

Name: Aneeq Muneer
21K-4582

Supervisor: Dr. Anam Qureshi
Co-supervisor: Dr. Farrukh Hasan

Department of Computer Science

FAST-National University of Computer & Emerging Sciences, Karachi

Contents

1 Introduction **3**

2 Related Work **3**

3 Methodology **5**
- 3.1 Navigation Module 5
  - 3.1.1 Object Detection 5
  - 3.1.2 Distance Calculation 6
  - 3.1.3 Navigation System 6
- 3.2 Image Description Module 6
  - 3.2.1 Feature Extraction 6
  - 3.2.2 Region Proposal Network (RPN) 7
  - 3.2.3 ROI Align and Classification 7
  - 3.2.4 Output Generation 8
- 3.3 Face Recognition Module 8
  - 3.3.1 Dataset Preprocessing 8
  - 3.3.2 Model Selection 8
  - 3.3.3 Model Architecture 8
  - 3.3.4 Model Functionality 9
  - 3.3.5 Model Optimization and Performance Improvement 10

4 Test & Result **11**
- 4.1 Navigation Module 11
  - 4.1.1 YOLO Model Version Comparison 11
  - 4.1.2 Distance Calculation 11
- 4.2 Image Description Module 11
- 4.3 Face Recognition Module 12

5 System Diagram **12**

6 Goals For FYP II **12**

7 Conclusion **13**

1 Introduction

Navigating the world and accessing information independently remain significant challenges for blind and visually impaired individuals. These challenges not only affect daily life but also hinder opportunities in education, employment, and social interactions. Existing assistive technologies such as white canes, guide dogs, and certain electronic devices provide limited solutions but often fail to address dynamic and multifaceted scenarios. Advanced assistive devices like smart glasses or specialized AI tools offer more functionality but are usually expensive, resource-intensive, or dependent on external processing, making them less practical for widespread use.

The rapid advancements in computer vision, edge computing, and artificial intelligence (AI) present an opportunity to develop innovative solutions tailored to the needs of visually impaired individuals. Technologies like real-time object detection, Optical Character Recognition (OCR), and audio-based feedback systems hold immense potential to enhance accessibility and independence. However, integrating these technologies into a unified, cost-effective, and lightweight system has yet to be fully realized.

This project addresses these gaps by proposing a lightweight, multifunctional assistive system designed to operate on low-cost hardware like Raspberry Pi. The system features three primary modes: a Navigation Module for obstacle detection and guidance, an Image Description Module for environmental awareness, and a Text Reading Module for reading printed text. Additionally, a Face Recognition Feature is included to identify familiar individuals, further enhancing the system's usability in social scenarios.

2 Related Work

Navigation assistance for visually impaired individuals has been a focal area of research, leveraging advancements in object detection and distance estimation technologies. Real-time navigation systems aim to detect obstacles, estimate their distances, and provide actionable guidance to users. While traditional tools such as canes or guide dogs remain common, the integration of computer vision technologies has shown immense potential in creating efficient, cost-effective, and accessible navigation solutions.

Object detection forms the foundation of many assistive navigation systems. YOLO (You Only Look Once) models have been widely recognized for their ability to perform real-time detection with high accuracy. The YOLOv11 framework by Ultralytics (1) improves upon its predecessors by enhancing detection precision and computational efficiency. Designed to operate on edge devices, YOLOv11n is ideal for low-power platforms such as Raspberry Pi, making it a suitable choice for lightweight navigation systems.

Previous studies have demonstrated the application of object detection for assistive technologies. For instance, (2) integrated deep learning-based detection methods into a smart glass system, which provided obstacle avoidance and environmental feedback to users. While the system showcased significant improvements in aiding navigation, its reliance on external processing units highlighted a gap in real-time performance on low-power devices.

The challenge of achieving real-time object detection on resource-constrained platforms has been addressed in multiple studies. (3) explored deploying SSD models on Raspberry Pi 4 for real-time detection tasks. By fine-tuning and quantizing the models, the study achieved substantial reductions in inference time while maintaining acceptable accuracy. Similarly, (4) used background subtraction combined with lightweight CNNs for object detection on Raspberry Pi. These studies highlighted the trade-off between model complexity and device capabilities, emphasizing the need for optimized frameworks like YOLOv11.

(5) conducted a comparative analysis of various neural network models for real-time detection on low-power devices, emphasizing that model selection should prioritize computational efficiency alongside accuracy. YOLO models consistently performed well due to their single-pass architecture, which significantly reduced processing overhead.

Building upon the reviewed literature, this project employs YOLOv11 for robust and real-time object detection while integrating distance estimation using the method described by Fulton and Anderson [6]. The navigation module dynamically divides the camera's field of view into regions, providing feedback to guide users away from obstacles. By focusing on computational efficiency and seamless integration, this work aims to address gaps in scalability and real-time performance for assistive navigation systems.

Image captioning technologies have emerged as transformative tools for aiding visually impaired individuals in understanding and navigating their environment. These systems rely on artificial intelligence to generate textual descriptions of images, often employing deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

(6) proposed a CNN-LSTM-based encoder-decoder architecture with Bahdanau attention, utilizing ResNet101 for feature extraction and achieving high BLEU scores, highlighting the role of attention mechanisms in improving captioning accuracy. Similarly, (7) incorporated beam search into an attention-based CNN-LSTM model for enhanced caption quality. Their approach enabled multi-modal functionalities such as face recognition and text-to-speech on wearable platforms, demonstrating the practical utility of portable devices in real-world scenarios.

Recent studies have emphasized the integration of image captioning into cost-effective and real-time systems. (8) developed a VGG16-LSTM framework deployed on Raspberry Pi 4B, utilizing a NoIR camera for real-time operations. Their results indicated significant improvements in user mobility and situational awareness, achieved through rigorous evaluation metrics such as BLEU and ROUGE. (9) highlighted the flexibility of modular systems by combining text recognition and color detection with captioning, delivering comprehensive assistance via Raspberry Pi. Their modularity allows for scalability and adaptability across different tasks, enhancing the inclusivity of such systems. Proposed Methodology This project leverages a pre-trained Faster R-CNN with ResNet-50 FPN model [11] for efficient object detection and region proposal, following (10). To enhance computational efficiency, ResNet-50 replaces ResNet-101 while maintaining robust detection accuracy. The system generates precise region proposals, forming the basis for image captioning tasks, enabling detailed and contextually accurate descriptions. This streamlined architecture supports assistive technologies by optimizing detection accuracy and computational demands, ensuring practical deployment for visually impaired users.

Face recognition has become a crucial technology in assistive systems, enabling visually impaired individuals to identify people in their surroundings. Advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have significantly improved feature extraction and recognition accuracy. These developments have also facilitated the deployment of lightweight models on constrained devices like Raspberry Pi for real-time applications. Despite progress, challenges persist in optimizing these systems for resource-limited environments and maintaining consistent performance under varying conditions. This review examines key studies to explore CNN-based solutions, effective methodologies, and areas for further improvement in assistive face recognition systems.

(11) presents a real-time face detection and recognition framework optimized for Raspberry Pi using Convolutional Neural Networks (CNNs). The system trains on a labeled dataset, using a voting-based mechanism for recognition, and achieves up to 98.3% accuracy across datasets such as Virtual Makeup and Face Recognition. Despite the Raspberry Pi's limited resources, the model performs robustly under challenging conditions like masked faces and varying lighting. This research demonstrates the feasibility of deploying lightweight CNNs on edge devices, providing valuable insights for assistive technologies and real-time surveillance.

(12) introduces a CNN-based approach to face recognition, outperforming traditional methods like PCA, LBPH, and KNN. Using the ORL face database of 400 images across 40 classes, the CNN, with two convolutional layers, pooling layers, and fully connected layers, achieved a top accuracy of 98.3%. The model's ability to learn complex hierarchical features makes it superior in handling variations in lighting, pose, and expressions, highlighting its effectiveness for face recognition and potential for lightweight, practical applications.

3 Methodology

3.1 Navigation Module

The proposed system is a navigation aid for visually impaired individuals, designed to detect objects, estimate their distances, and provide dynamic navigation guidance. The system is structured into three primary modules: object detection, distance calculation, and navigation.

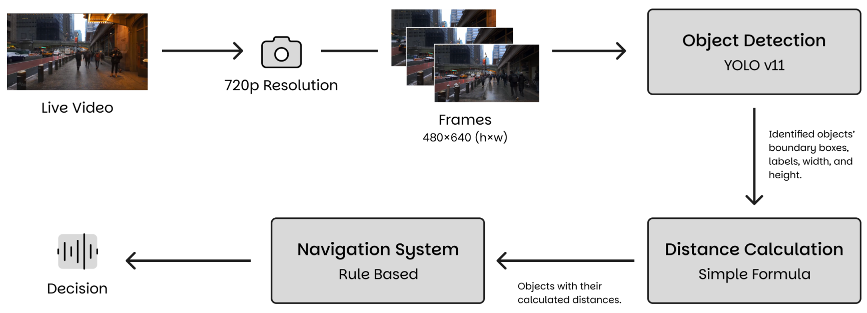


Figure 1: Navigation Module

3.1.1 Object Detection

The object detection module uses the YOLOv11 (You Only Look Once, version 11) framework (1), a state-of-the-art algorithm for real-time object detection. YOLOv11 divides the input image into a grid and predicts bounding boxes, class labels, and confidence scores for objects within each grid cell. This module outputs the detected objects' labels, bounding box coordinates, and dimensions (width and height). The choice of YOLOv11 ensures high accuracy and low latency, making it suitable for real-time deployment on resource-constrained hardware.

3.1.2 Distance Calculation

The distance calculation module estimates the distance between the detected objects and the camera. Using the bounding box dimensions provided by YOLOv11, the module leverages a formula derived from the relationship between an object's perceived size in an image and its actual size. Predefined average heights for common objects are stored in the system. These values are used to calculate distances using the formula:

$$d = \frac{H_{real} \cdot f}{H_{pixel}}$$

where:

- d is the distance to the object (in meters),
- H_{real} is the actual height of the object (in meters),
- H_{pixel} is the height of the object in the image (in pixels), and
- f is the camera's focal length in pixels.

The methodology for this calculation is inspired by the principles of photographic distance estimation (13). The accuracy of this module depends on the reliability of predefined object sizes and the camera calibration.

3.1.3 Navigation System

The navigation module provides guidance based on the detected objects and their positions relative to the camera's field of view. The camera's view is divided into three regions: left, center, and right. Each region is monitored for potential obstacles that exceed a predefined distance threshold. The rule-based system processes the data as follows:

If an object enters a region and its distance falls below the threshold, the region is marked as blocked. Guidance is provided to the user to navigate toward unblocked regions. The integration of these modules allows for a lightweight yet effective navigation system, capable of operating in real-time on hardware like the Raspberry Pi.

3.2 Image Description Module

The first phase of the image description module focuses on implementing the object detection component, which is crucial for enabling functionalities like object identification to provide environmental understanding for visually impaired individuals. The methodology for this module is detailed below:

3.2.1 Feature Extraction

The object detection process begins with feature extraction. The pretrained Faster R-CNN employs ResNet50 (14) as its backbone network, enhanced with a Feature Pyramid Network (FPN). ResNet50 extracts hierarchical features from input images, capturing both low-level and high-level features. The FPN complements this by combining features across multiple scales, enabling the detection of objects of varying sizes.

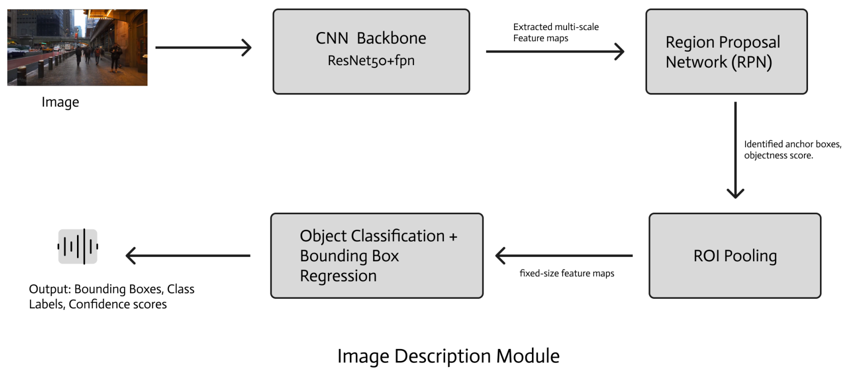


Image
CNN Backbone
ResNet50+fpn
Extracted multi-scale Feature maps
Region Proposal Network (RPN)
Identified anchor boxes, objectness score.
ROI Pooling
fixed-size feature maps
Object Classification + Bounding Box Regression
Output: Bounding Boxes, Class Labels, Confidence scores
Image Description Module

Figure 2: Image Description Module

3.2.2 Region Proposal Network (RPN)

The extracted features are processed through the Region Proposal Network (RPN), which generates potential object regions:

- Anchor Boxes: The RPN uses predefined anchor boxes to scan the image for objects.
- Objectness Score: Each anchor is scored to predict the likelihood of containing an object.
- Bounding Box Refinement: Predicted bounding boxes are adjusted to improve alignment with objects.

3.2.3 ROI Align and Classification

After region proposals are generated by the Region Proposal Network (RPN), they undergo further processing in the ROI Align and Classification stage to detect and classify objects accurately.

ROI Align

- The Region of Interest (ROI) Align operation ensures that the proposed regions align accurately with the feature maps, correcting for spatial misalignment that may arise from resizing or rounding during feature extraction.
- Unlike its predecessor, ROI Pooling, which approximates feature values, ROI Align uses bilinear interpolation to calculate feature values, preserving spatial details.
- The output of ROI Align is a fixed-size feature map for each proposed region, regardless of its original dimensions.

Classification

Once the fixed-size feature maps are obtained, they are fed into a fully connected neural network (classifier head) for the following tasks:

- Object Classification: Each ROI is classified into one of the predefined object categories or marked as "background" (if no object is detected in the region).
- Bounding Box Regression: Simultaneously, the network predicts offsets to refine the bounding boxes further for precise localization.

3.2.4 Output Generation

The final output after classification are:

- Bounding Boxes: Accurate coordinates for detected objects.
- Class Labels: Identified categories for each detected object.
- Confidence Scores: Probabilities for each prediction.

3.3 Face Recognition Module

The face recognition module is designed to identify individuals stored in a database by analyzing input images and determining whether they belong to the database, functioning as a binary classification system. In the future, this module is intended to run on a Raspberry Pi, necessitating lightweight and efficient implementation.

3.3.1 Dataset Preprocessing

The dataset used for training and testing the model is the Labeled Faces in the Wild (LFW), sourced from Kaggle. This dataset contains multiple folders, each named after the individual whose images it contains. To simulate real-world usage, a selected number of folders were chosen to represent "DB people" (known individuals stored in the database), while others were randomly selected as "Non-DB people" (unknown individuals). These two groups were combined into a single dataset, ensuring an almost equal representation of both classes for balanced testing.

3.3.2 Model Selection

For the model selection, options such as VGG16 and ResNet were considered due to their robust performance in image recognition tasks. However, these models are computationally heavy and unsuitable for deployment on a Raspberry Pi, which has limited processing power.
A custom Convolutional Neural Network (CNN) was chosen instead, as it offers automatic feature extraction, where the network learns to identify important features (like edges, textures, or shapes) directly from the data without requiring manual input, and parameter sharing, which reduces the number of parameters by using the same filters across different parts of the input image, making the model more efficient and lightweight.

3.3.3 Model Architecture

The model contains two hidden layers and an output layer. The structure of each layer is as follows:

- **Hidden Layer 1:**

  - **Convolutional Layer:**
    - **Filters:** 32
    - **Kernel Size:** 3×3 / 5×5
    - **Activation Function:** ReLU
    - **Input Shape:** $(250, 250, 3)$
  - **Pooling Layer:**
    - **Type:** Max Pooling / Average Pooling
    - **Pool Size:** 2×2

- **Hidden Layer 2:**
  - **Convolutional Layer:**
    - **Filters:** 64
    - **Kernel Size:** 3×3 / 5×5
    - **Activation Function:** ReLU
  - **Pooling Layer:**
    - **Type:** Max Pooling / Average Pooling
    - **Pool Size:** 2×2

- **Output Layer:**
  - **Flatten Layer**
  - **Dense Layer:**
    - **Type:** Fully Connected
    - **Units:** 1
    - **Activation Function:** Sigmoid

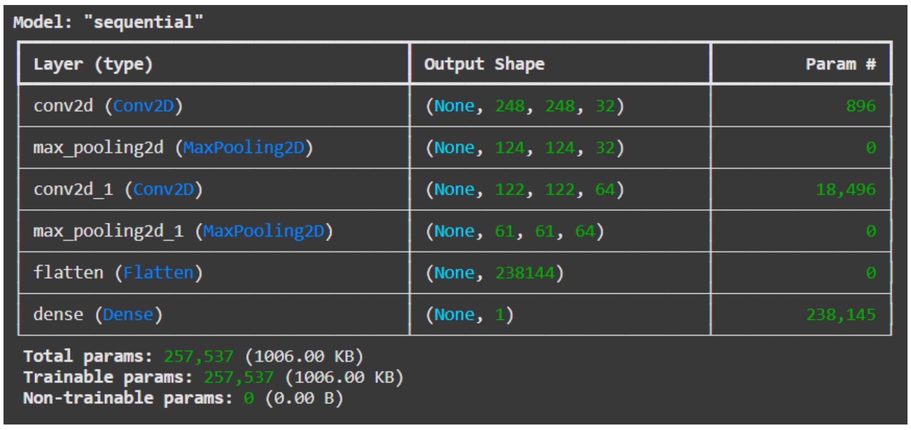
Training Configuration:

- **Optimizer:** Adam
- **Loss Function:** Binary Cross Entropy
- **Metrics:** Accuracy

The model summary is provided in 3.

3.3.4 Model Functionality

The CNN model begins with an input layer designed to process images with dimensions 250×250×3 (the size of images in our dataset). This input layers contains a Conv2D layer which extracts low-level features such as edges and textures by scanning the image. This is followed by a MaxPooling2D layer which reduces the spatial dimensions of the feature maps, thereby lowering computational complexity and capturing dominant features. The second Conv2D layer extracts more complex patterns like shapes or contours relevant to face detection. This layer is again followed by MaxPooling2D, further downsampling the feature maps while retaining critical information.



```
Model: "sequential"
┌──────────────────────────────┬────────────────────────┬──────────────┐
│ Layer (type)                 │ Output Shape           │      Param # │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ conv2d (Conv2D)              │ (None, 248, 248, 32)   │          896 │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ max_pooling2d (MaxPooling2D) │ (None, 124, 124, 32)   │            0 │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ conv2d_1 (Conv2D)            │ (None, 122, 122, 64)   │       18,496 │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ max_pooling2d_1 (MaxPooling2D)│ (None, 61, 61, 64)    │            0 │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ flatten (Flatten)            │ (None, 238144)         │            0 │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ dense (Dense)                │ (None, 1)              │      238,145 │
└──────────────────────────────┴────────────────────────┴──────────────┘
 Total params: 257,537 (1006.00 KB)
 Trainable params: 257,537 (1006.00 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 3: Face Recognition Model Summary

Next, the Flatten layer converts the 2D feature maps into a 1D vector, enabling the features to be fed into the fully connected layer. Finally, a Dense layer with a single neuron and a sigmoid activation function outputs a binary classification result. This indicates whether the input image belongs to a person stored in the database or not. The model uses the Adam optimizer for efficient weight updates, binary crossentropy as the loss function to calculate the prediction error for binary classification, and accuracy as the evaluation metric.

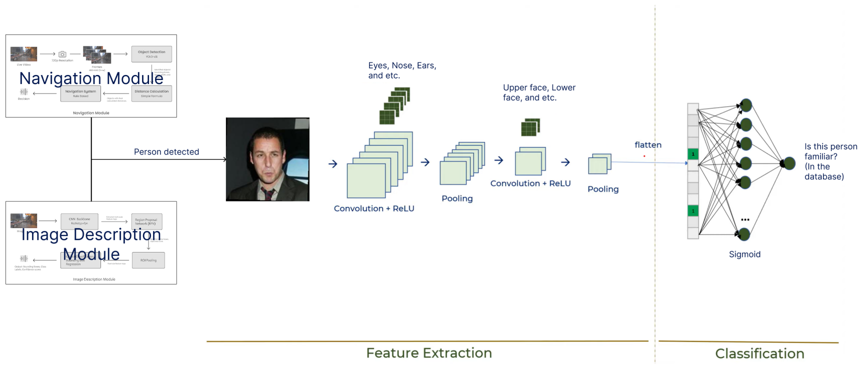


Figure 4: Face Recognition Module

3.3.5 Model Optimization and Performance Improvement

After experimenting with different configurations, a two-layer CNN architecture was selected to strike a balance between computational efficiency and functionality. This architecture ensures the model remains lightweight while maintaining its accuracy. Hyperparameter tuning played a crucial role in improving performance. By using a 3x3 kernel size and incorporating Average Pooling, the model's accuracy improved significantly, rising from an initial 50-60% to a more reliable 70-80%.

4 Test & Result

4.1 Navigation Module

4.1.1 YOLO Model Version Comparison

In this section, we present a comparison of various YOLO (nano and small) models evaluated for real-time object detection performance on a Raspberry Pi 4. We used the small and nano versions of YOLO because they are significantly less complex compared to the primary model. The benchmark results demonstrate the performance across different formats (PyTorch, TorchScript, ONNX, OpenVINO, PaddlePaddle, MNN, and NCNN), with key metrics including model size, mean Average Precision (mAP50-95), inference time, and Frames Per Second (FPS). These evaluations aim to identify the most suitable YOLO model for use in assistive navigation systems, balancing detection accuracy and computational efficiency.

The table 1 outlines the benchmark results for multiple YOLO models, including YOLOv8, YOLOv9, YOLOv10, YOLOv11, and their respective formats. Among the models tested, YOLOv11 emerged as the most efficient choice in terms of inference time and FPS when processed in the ONNX format. YOLOv11 demonstrated a significant advantage in FPS, reaching 2.13 FPS with an inference time of 469.67 ms, which is suitable for real-time applications. Additionally, YOLOv11 also offered a good balance between accuracy (mAP50-95 of 0.6082) and computational efficiency.

Thus, based on these benchmarks, YOLOv11 (in ONNX format) was selected as the optimal model for integration into the navigation module of the assistive system, offering the best combination of detection performance and real-time processing capabilities.

4.1.2 Distance Calculation

The distance calculation module was tested for accuracy using common objects, such as persons and chairs, with the system employing a simple formula based on the real-world height and perceived size in the image. The system successfully estimated distances with an average error margin of ± 0.5 meters, which is acceptable for real-time navigation assistance.

However, the accuracy of the system was sensitive to camera calibration and object detection. Inaccuracies in detecting object boundaries or misidentifying objects led to errors in height estimation, which impacted the distance calculation. To improve performance, proper camera calibration and accurate object detection are critical.

4.2 Image Description Module

In this section, we compare the performance of four object detection models VGG16 (used in R-CNN Framework), YOLOv3, YOLOv11, and Faster R-CNN with ResNet50 FPN to evaluate their suitability for accurate and efficient object detection from an input image. These models were assessed based on key performance metrics such as mean Average Precision(mAP), Precision, Recall, and Inference Time resource requirements on a CPU-only system.

The table 2 outlines the benchmark results for multiple models, including VGG16, YOLOv3, YOLOv11, and Faster R-CNN with ResNet50-FPN. Among the models tested, Faster R-CNN with ResNet50-FPN emerged as the most efficient choice in terms of object detection model for image description module requiring high accuracy and robust object detection but with fewer time constraints.

Based on this comparison, Faster R-CNN ResNet50 FPN was selected as the object detection model for this phase of the project due to its superior accuracy and robust performance in detecting objects of varying scales. Its ability to generate precise region proposals and leverage ResNet50's feature extraction capabilities ensures reliable detection, even in challenging environments. This aligns well with the project's objective of creating a high-accuracy object detection module.

4.3 Face Recognition Module

During the testing phase, the focus was on evaluating the model's performance in terms of accuracy and computational efficiency, aligning with the goal of deploying the face recognition system on a Raspberry Pi. A two-layer CNN was initially selected for experimentation based on prior research and knowledge of lightweight architectures.
The model was subsequently tested with various hyperparameter combinations, focusing primarily on kernel size and pooling type. As shown in Table 3, the top-performing model was identified as the one with a **3x3 kernel size** and **Average pooling type**, achieving an **average accuracy of 64.61%** and a **max accuracy of 75.14%**.

5 System Diagram

In the current stage of development, each module of the system is functioning independently. While the face recognition module, navigation module, and other components are designed to work together as part of a unified system in the future, they are currently being tested and optimized separately. Therefore, a system diagram that illustrates the integration of all modules is not applicable at this point.

6 Goals For FYP II

1. **Decrease Model Inference Speed and Increase System Performance:** Optimize the model's inference speed through techniques like pruning, quantization, and hardware-specific tuning to ensure real-time performance on constrained devices. Experiment with various methods to enhance the system performance.

2. **Implement Caption Generation for Image Description:** Develop an automated caption generation feature to describe the content of images, helping visually impaired users gain a clearer understanding of their surroundings and navigate more effectively.

3. **Extend Model Functionality for Specific Identity Recognition** Upgrade the face recognition module to identify specific individuals from a database, transitioning from binary classification to multi-class recognition.

4. **Implement Text Reading Mode (Optional)** Introduce a text-reading mode using OCR technology to recognize and read aloud text from books and other printed materials, to assist users.

7 Conclusion

This project presents a significant step forward in developing assistive technologies for visually impaired individuals by integrating AI-powered solutions into a lightweight, multifunctional system. The proposed system is designed to address the challenges of navigation, environmental awareness, and accessibility by incorporating three key modules: a navigation module, an image description module, and a face recognition feature.

The navigation module effectively employs YOLOv11 for real-time object detection and integrates a distance calculation method based on object dimensions and camera properties. Testing demonstrated its capability to calculate distances with an average error margin of ± 0.5 meters, which is suitable for obstacle avoidance and user guidance. The system divides the camera's field of view into regions and provides audio-based feedback for effective navigation, proving its functionality for real-time applications on low-power hardware like Raspberry Pi.

The image description module leverages advanced object detection techniques, such as Faster R-CNN with ResNet50, to generate meaningful contextual descriptions of the environment. This capability enhances situational awareness for users, helping them better understand their surroundings.

The face recognition feature adds another layer of utility by identifying familiar individuals in the user's vicinity. This functionality, powered by a lightweight convolutional neural network, ensures real-time performance even on resource-constrained devices.

Throughout development, the system's design remained focused on minimizing computational demands while maintaining functionality. However, challenges such as sensitivity to camera calibration, variations in object detection accuracy, and environmental complexity highlight areas for future improvement. Potential enhancements include advanced model optimization techniques, such as pruning and quantization, and the integration of additional features like text reading (OCR).

In conclusion, this project achieves its primary goal of creating a lightweight, multifunctional assistive system that operates efficiently on constrained devices. By bridging the gap between affordability and functionality, it provides a scalable solution for improving the independence and quality of life for visually impaired individuals. This work contributes to both academic research and practical innovation, paving the way for more accessible and effective assistive technologies.

References

[1] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[2] J. C. Mukhriddin Mukhiddinov, "Smart glass system using deep learning for the blind and visually impaired," *Electronics*, 2021, available at: https://www.mdpi.com/2079-9292/10/22/2756. [Online]. Available: https://www.mdpi.com/2079-9292/10/22/2756

[3] O. Ferm, "Real-time object detection on raspberry pi 4: Fine-tuning a ssd model using tensorflow and web scraping," *Independent*, 2020, available at: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1451946&dswid=3416. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1451946&dswid=3416

[4] J. L. Torbjørn Grande Østby, Ola Marius Lysaker, "Object detection and tracking on a raspberry pi using background subtraction and convolutional neural networks," *usn*, 2018, available at: https://openarchive.usn.no/usn-xmlui/bitstream/handle/11250/2567353/Master%C3%98stby%202018.pdf?sequence=1. [Online]. Available: https://openarchive.usn.no/usn-xmlui/bitstream/handle/11250/2567353/Master%C3%98stby%202018.pdf?sequence=1

[5] T. A. M. E. Zagitov A, Chebotareva E, "Comparative analysis of neural network models performance on low-power devices for a real-time object detection task," *http://www.computeroptics.ru*, 2024, available at: https://pdfs.semanticscholar.org/8b4f/028a5374c73a0c04eb9dc9e78209842332b5.pdf. [Online]. Available: https://pdfs.semanticscholar.org/8b4f/028a5374c73a0c04eb9dc9e78209842332b5.pdf

[6] A. V. I. A. R. Faurina, A. Jelita, "Image captioning to aid blind and visually impaired outdoor navigation," *https://ijai.iaescore.com/index.php/IJAI/index*, 2023, available at: https://ijai.iaescore.com/index.php/IJAI/article/view/22587/13681. [Online]. Available: https://ijai.iaescore.com/index.php/IJAI/article/view/22587/13681

[7] N. B. N. K. A. Hengle, A. Kulkarni and R. Udyawar, "Smart cap: A deep learning and iot based assistant for the visually impaired," *https://ieeexplore.ieee.org/xpl/conhome/9203793/proceeding*, 2020, available at: https://ieeexplore.ieee.org/abstract/document/9214140. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9214140

[8] K. Safiya and R. Pandian, "A real-time image captioning framework using computer vision to help the visually impaired," *https://link.springer.com/journal/11042*, 2023, available at: https://link.springer.com/article/10.1007/s11042-023-17849-7. [Online]. Available: https://link.springer.com/article/10.1007/s11042-023-17849-7

[9] A. D. S. S. S.Dighe, S. Revshette, "Caption-speak: Empowering sight with pi," *https://ijnrd.org/1index.php*, 2024, available at: https://ijnrd.org/papers/IJNRD2407437.pdf. [Online]. Available: https://ijnrd.org/papers/IJNRD2407437.pdf

[10] K. B. J. S. S. Herdade, A. Kappeler, "Image captioning: Transforming objects into words," *https://proceedings.neurips.cc/*, 2019, available at: https://proceedings.neurips.cc/paper_files/paper/2019/file/680390c55bbd9ce416d1d69a9ab4760d-Paper.pdf. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/680390c55bbd9ce416d1d69a9ab4760d-Paper.pdf

[11] R. A. T. S. Khan, M. U. Islam, "Face detection recognition from images videos based on cnn raspberry pi," *https://www.mdpi.com/2079-3197/10/9/148*, 2019, available at: https://www.mdpi.com/2079-3197/10/9/148. [Online]. Available: https://www.mdpi.com/2079-3197/10/9/148

[12] T. M. R. R. P. Kamencay, M. Benco, "A new method for face recognition using convolutional neural network," *https://advances.vsb.cz/index.php/AEEE/article/view/2389/pdf*, 2017, available at: https://advances.vsb.cz/index.php/AEEE/article/view/2389/pdf. [Online]. Available: https://advances.vsb.cz/index.php/AEEE/article/view/2389/pdf

[13] K. A. Wayne Fulton. Calculate distance or size of an object in a photo image. [Online]. Available: https://www.scantips.com/lights/subjectdistance.html

[14] R. G. J. S. S. Ren, K. He, "fasterrcnn$_{r}esnet50_{f}pn.''[Online].Available:$

Model	Format	Size (MB)	mAP50-95 (B)	Inference Time (ms/im)	FPS
yolov8n.pt	PyTorch	6.2	0.6381	1280.00	0.78
	TorchScript	12.4	0.6092	1623.27	0.62
	ONNX	12.2	0.6092	511.97	1.95
	OpenVINO	12.3	0.6092	395.34	2.53
	PaddlePaddle	24.4	0.6092	1797.31	0.56
	MNN	12.2	0.6092	557.10	1.80
	NCNN	12.2	0.6092	437.87	2.28
yolov8s.pt	PyTorch	21.5	0.6967	3744.15	0.27
	TorchScript	43.0	0.7136	4575.28	0.22
	ONNX	42.8	0.7136	1543.48	0.65
	OpenVINO	0.0	NaN	NaN	NaN
	PaddlePaddle	85.5	0.7136	6949.87	0.14
	MNN	42.7	0.7136	1538.85	0.65
	NCNN	42.7	0.7136	960.83	1.04
yolov9t.pt	PyTorch	4.7	0.6568	1541.02	0.65
	TorchScript	9.0	0.6428	2199.16	0.45
	ONNX	8.3	0.6428	794.17	1.26
	OpenVINO	0.0	NaN	NaN	NaN
	PaddlePaddle	16.6	0.6428	2041.70	0.49
	MNN	8.2	0.6428	573.30	1.74
	NCNN	8.2	0.6428	461.64	2.17
yolov9s.pt	PyTorch	14.7	0.6973	3767.02	0.27
	TorchScript	28.5	0.6981	4837.02	0.21
	ONNX	27.8	0.6981	1643.18	0.61
	OpenVINO	0.0	NaN	NaN	NaN
	PaddlePaddle	55.6	0.6981	5013.48	0.20
	MNN	27.6	0.6974	1520.52	0.66
	NCNN	27.7	0.6981	1232.31	0.81
yolov10n.pt	PyTorch	5.6	0.6981	1503.30	0.67
	TorchScript	11.1	0.5715	1619.54	0.62
	ONNX	9.0	0.5715	576.33	1.74
	OpenVINO	0.0	NaN	NaN	NaN
	PaddlePaddle	0.0	NaN	NaN	NaN
	MNN	8.9	0.5715	535.33	1.87
	NCNN	9.0	NaN	NaN	NaN
yolo11n.pt	PyTorch	5.4	0.6100	1185.99	0.84
	TorchScript	10.5	0.6082	1616.99	0.62
	ONNX	10.2	0.6082	469.67	2.13
	OpenVINO	0.0	NaN	NaN	NaN
	PaddlePaddle	20.4	0.6082	1498.27	0.67
	MNN	10.1	0.6082	527.07	1.90
	NCNN	10.2	0.6082	429.72	2.33
yolo11s.pt	PyTorch	18.4	0.7526	3352.92	0.30
	TorchScript	36.5	0.7400	4151.69	0.24
	ONNX	36.3	0.7400	1382.89	0.72
	OpenVINO	36.4	0.7400	1201.86	0.83
	PaddlePaddle	72.5	0.7400	4125.28	0.24
	MNN	36.2	0.7400	1522.44	0.66
	NCNN	36.2	0.7400	1164.74	0.86

Table 1: Benchmark Results for YOLO Models on coco8.yaml at imgsz=640 on Raspberry Pi 4

Model	Pretrained Dataset	mAP (%)	Precision (%)	Recall (%)	Inference Time (CPU)	Strengths	Limitations
VGG16 (as R-CNN)	ImageNet	66-70	70-75	65-70	2.5-3 s/img	High accuracy on small datasets	Very slow inference, computationally expensive
YOLOv3	COCO	57-63	85-90	75-80	300-500 ms/img	Real-time processing, good balance of speed and accuracy	Struggles with very small objects, slightly outdated for modern tasks
YOLOv11	COCO	54.7	75-80	68-74	300-500 ms/img	Optimized for speed and efficiency, making it suitable for real-time applications	Struggles with complex and overlapping objects
Faster R-CNN ResNet50 FPN	COCO	37.4-42.5	85-88	75-80	5-6 s/img	Excellent for high-quality object detection, flexible with complex architectures	High computational cost, slower compared to YOLO for real-time applications

Table 2: Benchmark Results for Object Detection Models on CPU

Kernel Size	Pooling Type	Epochs	Max Accuracy	Average Accuracy
3 x 3	Max	10	67.00%	50.09%
yellow 3 x 3	Average	10	75.14%	64.61%
5 x 5	Max	10	68.99%	52.40%
5 x 5	Average	10	70.00%	50.14%

Table 3: Model Performance with Different Kernel Sizes and Pooling Types