Report: Baseline Results

Obstacle detection

Dataset Collection

We manually collected our dataset by capturing images from various locations across the IIIT Delhi campus, focusing on objects that visually impaired individuals may encounter at different body levels. To simulate real-life scenarios and improve the model's ability to detect obstacles at varying heights, we categorized the data collection process into two distinct perspectives:

Head-Level Perspective: For objects that appear directly in front of the upper body, particularly at eye level or above. This includes obstacles such as door frames, signs, branches, or other high-standing objects that a visually impaired person might encounter directly in their line of sight while walking. Images for this category were captured by holding the camera at head height to simulate a natural viewpoint.

Example:

Knee-Level Perspective: This involves capturing objects that are located closer to the ground, which might impact the lower body. Examples include obstacles like chairs, tables, steps, small animals (like dogs or cats), or other low-lying objects that could obstruct a person’s path. Images were taken from knee height to simulate a lower-level perspective, ensuring the dataset accounts for obstacles that may not be visible from the head-level view but are crucial for safe navigation.

Example:

Description:

Training Dataset: 50 images of IIITD

Class 0 (Obstacle): 26

Class 1 (No obstacle) : 24

Testing Dataset: 22 images of IIITD

Model architecture and approach:

Model A: Scratch CNN-based model

Input (224×224×3) → 3 CNN layers → Output (2 classes)

The preprocessing steps involve:

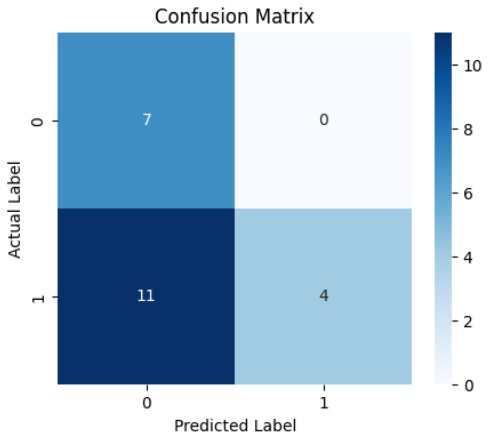
1. **Resizing Images:** Convert all input images to 224×224 dimensions to match the model's expected input size.
2. **Normalization:** Scale pixel values to the range [0, 1] by dividing by 255, ensuring consistent input to the model.
3. **Splitting Dataset:** Divide the dataset into training, validation, and test sets to train and evaluate model performance.

It is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the metric.

This is an attempt to use a very rudimentary CNN-based model to give a sanity check that this problem requires an acute understanding of the object in the model.

Results:

Confusion matrix:



Accuracy: 0.5000
Total Inference Time: 0.9496 seconds
Average Inference Time: 0.0432 seconds
False Negative Rate: 0.7333
Recall: 0.2667

The scratch model neither enough data nor model complexity to learn what is an obstacle.

Next we tried CNN based pre-trained model

Model B. Microsoft/CvT

CvT model to classify an image of the COCO 2017 dataset into one of the 1,000 ImageNet classes.

This model is generalized to learn about the object. We tried this to fine-tune and test the small dataset defined above.

As it is trained to predict 1000 classes, we tried with

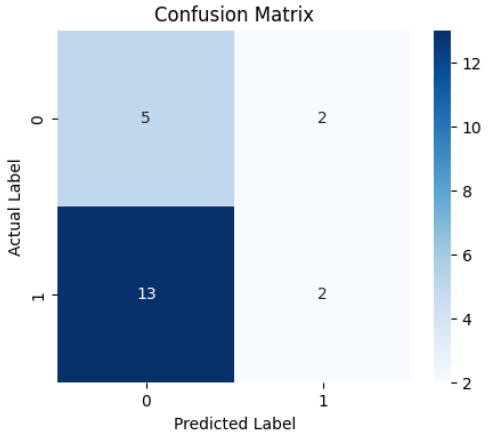
1. CvT trimmed to 2 classes
2. CvT + MLP to bring to 2 classes

Model B.1 - CvT trimmed

The model is based on the pre-trained `CvT-13` (Convolutional Vision Transformer) from Microsoft, adapted for a two-class obstacle detection task. The classifier layer is modified to output 2 labels instead of the original 1000. The model is fine-tuned using `Trainer` from the Hugging Face library, leveraging the original pre-trained weights to quickly adapt to the new task while retaining the learned image features.

Results:

Confusion matrix:



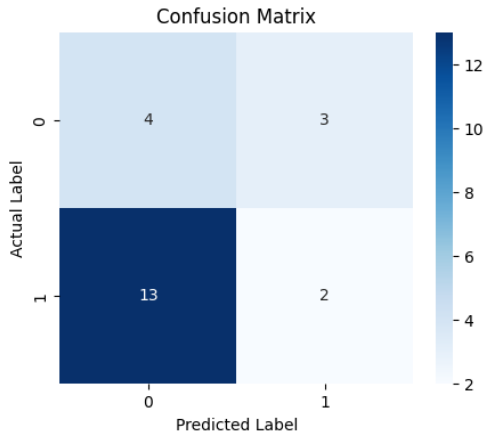
Accuracy: 0.3182
Total Inference Time: 2.2380 seconds
Average Inference Time: 0.1017 seconds
False Negative Rate: 0.8667
Recall: 0.1333

Model B.1 - CvT + MLP (1000 → 256 → 2)

The architecture is based on `CvT-13` (Convolutional Vision Transformer) from Microsoft, followed by an added MLP consisting of a linear layer reducing the output from 1000 to 256 dimensions, a ReLU activation, and a final linear layer reducing to 2 output classes for binary classification. The pre-trained convolutional transformer extracts features, while the custom MLP layer is responsible for adapting the model to the obstacle detection task.

Results:

Confusion matrix:



Accuracy: 0.2727
Total Inference Time: 1.5903 seconds
Average Inference Time: 0.0723 seconds

False Negative Rate: 0.8667
Recall: 0.1333

Model C.

Data Collection & Description

1. We collected some high-quality photos and annotated them with respect to the possible questions a blind person can asked during the Visual Question-Answering task, such as “What is in front of me?” or “Where am I?” etc.
2. We manually annotated each (Image, Question) pair and generated three answers per pair, each provided by a different annotator. These annotations were made in the context of IIIT Delhi.
3. We later used this dataset to generate baseline results for our Visual Question-Answering task, by feeding it into the models.
4. Calculated the LAVE Score on the predicted answers.

Input (224×224×3) → [Neural Network Layers] → Output (2 classes)

Image pre-processing:

Resizing: Each image is resized to a standard dimension of 224×224 pixels to ensure uniformity in input size for the model.

Color Conversion: The image color format is converted from BGR (used by OpenCV) to RGB, aligning with the expected input format for the model.

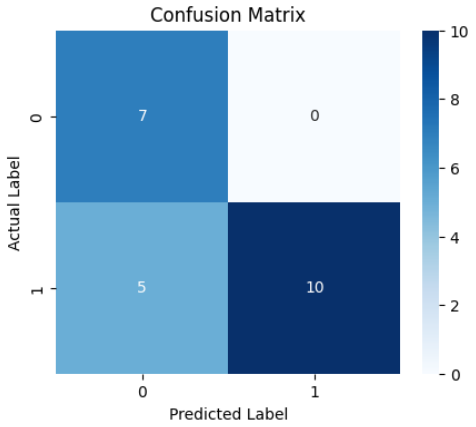
Normalization: Pixel values are scaled to a range between 0 and 1 by dividing by 255.0, which helps improve model performance and training stability.

Transposition: The image array is rearranged from a shape of (height, width, channels) to (channels, height, width) to meet the model's input requirements.

Batching: An additional dimension is added to create a batch size of 1, making the final input shape compatible for inference.

Results:

Confusion matrix:



Accuracy: 0.7727 (17 correct predictions out of 22)

Total Inference time: 0.2312 seconds

Average inference time: 0.0105 seconds

According to heatmap:

True Positives (TP) = 10
False Negatives (FN) = 5
True Negatives (TN) = 7
False Positives (FP) = 0

False negative rate: FN/(TP + FN) = 5/15 = 0.333

Recall: TP/(TP + FN) = 10/15 = 0.6667

Visual Question-Answering

LAVE Score

Given the LLM's generated text for the test example, we extract the rating r from the last character (either 1, 2 or 3) and linearly map it to a score s in the range [0, 1]: s = (r−1)/2.

Reference - https://arxiv.org/pdf/2310.02567

VizWiz Dataset

4319 (Image, Question) Pairs

	Inference Time	%Incorrect Predictions	%Ambiguous/IncompletePredictions	%Correct Predictions	LAVE Score (%Corr + %Amb * 0.5)/100
BLIP	7.85 itr/sec	76.93%	0.09%	22.96%	0.230
Llama 3.2 11B Vision Instruct (Together.ai)	8.65 sec/itr	27.55%	11.06%	61.37%	0.669

IIITD Dataset (Manually Annotated)

24 (Image, Question) Pairs

	Inference Time	%Incorrect Predictions	%Ambiguous/Incomplete Predictions	%Correct Predictions	LAVE Score (%Corr + %Amb * 0.5)/100
BLIP	5.43 itr/sec	66.66%	8.33%	25%	0.291
Llama 3.2 11B Vision Instruct (Together.ai)	8.48 sec/itr	29.16%	33.33%	37.5%	0.541

Recognize and identify faces relevant to the user

For this part our approach to tackle is to first convert an image into a vector using various vision transformers and do this for all the images of different people in the databse,

During inference phase we found out various distance metrics from the training phase and based on which vector had the “closest” distance inferred which class a certain image belongs to.

We tested the following models for vector embedding creation:

Model Name	Model ID	Description
ViT (Vision Transformer)	`google/vit-base-patch16-224`	Leverages transformers for image classification by splitting images into patches.
Swin (Swin Transformer)	`microsoft/swin-base-patch4-window7-224`	Uses shifted windows to efficiently capture local and global image features.
ConvNeXT	`facebook/convnext-base-224`	A modern CNN inspired by transformers while retaining the simplicity of convolutional layers.
RegNet	`facebook/regnet-y-400mf`	Balances performance and efficiency, offering flexible design for classification tasks.
ResNet50	`microsoft/resnet-50`	Introduces residual connections to enable deep network architectures.
ResNet101	`microsoft/resnet-101`	A deeper version of ResNet50, with 101 layers for enhanced task performance.
ResNet152	`microsoft/resnet-152`	The deepest ResNet variant with 152 layers for superior accuracy.
DEIT (Data-efficient Image Transformer)	`facebook/deit-base-patch16-224`	A ViT variant that performs well with less data and optimized training.

In order to calculate distance we used the following distance metrics:

Distance Metric	Description
Euclidean Distance	Measures the straight-line distance between two points in Euclidean space.
Cosine Similarity	Measures the cosine of the angle between two vectors, reflecting their orientation.
Manhattan Distance	Computes the sum of absolute differences between the coordinates of two points.
Chebyshev Distance	Measures the maximum absolute difference along any coordinate dimension.
Minkowski Distance	A generalized distance metric; Euclidean and Manhattan distances are special cases (p=2, p=1).

The goal is to choose the ideal combination of the embedding model and distance metric to be used.

We used 2 datasets to calculate all the metrics,

1. IDEAL DATASET: was taken from an existing dataset which has comprehensive number of images in both training and testing set, the training set for each person has 30+ images of the subject from various angles and head positions, the dataset is quite comprehensive and covers almost all possible angles thus resulting in very good matching results across models, this dataset serves as an ideal case for us and we would be trying to replicate its comprehensiveness going forward

Model	Top-1 Accuracy (%)	Top-3 Accuracy (%)	Top-5 Accuracy (%)
ViT	99.64	99.88	99.88
Swin	99.88	100.00	100.00
ConvNeXT	100.00	100.00	100.00
RegNet	99.88	100.00	100.00
ResNet50	100.00	100.00	100.00
ResNet101	100.00	100.00	100.00
ResNet152	99.88	99.88	100.00
DEiT	100.00	100.00	100.00

2. COLLECTED DATASET: this dataset has been collected by us mimicking the easiest possible scenario implementable at the university level, in the training we only took 1 sample of a formal photograph similar to an ID-card photograph of the person and tested out with real life scenario photographs in the testing set ie: when the person may not be necessarily posing to the camera similar to instances which will be faced in real life, this dataset gave poor results as displayed below and acts as a baseline system for this part.
   Please note that the models are giving near perfect results when the dataset is comprehensive hence our aim is to make improvements in the dataset going forward to go over and above the baseline results

Model	Top-1 Accuracy (%)	Top-3 Accuracy (%)	Top-5 Accuracy (%)
ViT	26.77	53.54	71.65
Swin	27.56	61.42	83.46
ConvNeXT	29.92	54.33	76.38
RegNet	22.05	51.97	77.95
ResNet50	23.62	42.52	58.27
ResNet101	29.92	49.61	65.35
ResNet152	27.56	55.12	70.08
DEiT	25.98	53.54	81.89

USING DIFFERENT DISTANCE METRICS:

- -- ViT ---
  Euclidean Distance: Top-1: 27.56%, Top-3: 54.33%, Top-5: 71.65%
  Cosine Similarity: Top-1: 30.71%, Top-3: 58.27%, Top-5: 76.38%
  Manhattan Distance: Top-1: 25.98%, Top-3: 53.54%, Top-5: 70.87%
  Chebyshev Distance: Top-1: 27.56%, Top-3: 50.39%, Top-5: 65.35%
  Minkowski Distance (p=3): Top-1: 28.35%, Top-3: 54.33%, Top-5: 73.23%

- -- Swin ---
  Euclidean Distance: Top-1: 27.56%, Top-3: 61.42%, Top-5: 83.46%
  Cosine Similarity: Top-1: 32.28%, Top-3: 66.93%, Top-5: 85.04%
  Manhattan Distance: Top-1: 29.13%, Top-3: 60.63%, Top-5: 84.25%
  Chebyshev Distance: Top-1: 25.98%, Top-3: 57.48%, Top-5: 77.95%
  Minkowski Distance (p=3): Top-1: 28.35%, Top-3: 62.20%, Top-5: 84.25%

- -- RegNet ---
  Euclidean Distance: Top-1: 22.05%, Top-3: 51.97%, Top-5: 77.95%
  Cosine Similarity: Top-1: 25.20%, Top-3: 51.18%, Top-5: 79.53%
  Manhattan Distance: Top-1: 22.05%, Top-3: 50.39%, Top-5: 74.02%
  Chebyshev Distance: Top-1: 20.47%, Top-3: 48.82%, Top-5: 69.29%
  Minkowski Distance (p=3): Top-1: 25.20%, Top-3: 53.54%, Top-5: 79.53%

- -- ResNet50 ---
  Euclidean Distance: Top-1: 23.62%, Top-3: 42.52%, Top-5: 58.27%
  Cosine Similarity: Top-1: 25.98%, Top-3: 40.94%, Top-5: 61.42%
  Manhattan Distance: Top-1: 25.98%, Top-3: 43.31%, Top-5: 58.27%
  Chebyshev Distance: Top-1: 10.24%, Top-3: 29.92%, Top-5: 44.09%
  Minkowski Distance (p=3): Top-1: 22.83%, Top-3: 40.94%, Top-5: 57.48%

- -- ResNet101 ---
  Euclidean Distance: Top-1: 29.92%, Top-3: 48.82%, Top-5: 64.57%
  Cosine Similarity: Top-1: 28.35%, Top-3: 52.76%, Top-5: 65.35%
  Manhattan Distance: Top-1: 26.77%, Top-3: 50.39%, Top-5: 65.35%
  Chebyshev Distance: Top-1: 14.17%, Top-3: 26.77%, Top-5: 37.80%
  Minkowski Distance (p=3): Top-1: 25.98%, Top-3: 51.18%, Top-5: 61.42%

- -- ResNet152 ---
  Euclidean Distance: Top-1: 27.56%, Top-3: 55.91%, Top-5: 70.08%
  Cosine Similarity: Top-1: 29.92%, Top-3: 53.54%, Top-5: 69.29%
  Manhattan Distance: Top-1: 31.50%, Top-3: 55.12%, Top-5: 70.08%
  Chebyshev Distance: Top-1: 14.96%, Top-3: 30.71%, Top-5: 39.37%
  Minkowski Distance (p=3): Top-1: 28.35%, Top-3: 52.76%, Top-5: 64.57%

- -- DEiT ---
  Euclidean Distance: Top-1: 27.56%, Top-3: 54.33%, Top-5: 82.68%
  Cosine Similarity: Top-1: 23.62%, Top-3: 58.27%, Top-5: 83.46%
  Manhattan Distance: Top-1: 25.98%, Top-3: 52.76%, Top-5: 80.31%
  Chebyshev Distance: Top-1: 22.83%, Top-3: 49.61%, Top-5: 71.65%
  Minkowski Distance (p=3): Top-1: 28.35%, Top-3: 55.12%, Top-5: 81.10%