Research Article

Enhancing Environmental Awareness for the Visually Impaired with a Real-Time Mobile Ai Assistant

Lakkimsetty Thiru venkata Reddy sandeep[1], Smitha Chowdary Ch[2]

1, 2 Department of CSE, Koneru Lakshmaiah Foundation, Vaddeswaram, Andhra Pradesh, India. Email: lakkimsettysandeep@gmail.com , smithacsc@gmail.com

Abstract

This research focuses on the development of autonomous mobility solutions for visually impaired users in dynamic unstructured real-life environments. In that regard, the problem at hand is hindered due to insufficient semantic understanding associated with existing assistive devices. Indeed, while such solutions are highly effective for immediate obstacle avoidance, they lack ability to deliver precise and timely information necessary for more complicated spatial understanding of surroundings. For this reason, we have developed a two-stage deep learning solution for real-time audio description that will be delivered within a cloud-enabled mobile platform. The process starts with real-time camera frame classification performed via You Only Look Once (YOLO) detector, followed by the processing of selected image patches via second CNN, trained for fine-grained classification. Such a two-stage solution allows detecting of minute details such as open or closed status of a detected door that cannot be achieved with a one-stage model. The structured data obtained is then transferred to the mobile device for further conversion into voice synthesis via native engine. Evaluation of the proposed method on the specially curated dataset shows 96% descriptive accuracy

Keywords: Assistive Technology, Object Detection, Fine-Grained Classification, Deep Learning, Mobile AI, YOLO.

1. Introduction

The interpretation of environmental signals and reaction to them play an important role in autonomous and safe navigation, which cannot be done efficiently in case of visual disability. The white cane and guide dog can give accurate signals about the presence of obstacles in the vicinity but they do not provide any information about the nature or state of objects that are beyond their reach. Thus, simple tasks like finding an empty seat or finding out whether the door is open become difficult and pose one of the major barriers to autonomy [26]. With the evolution of computer vision techniques, electronic travel aids based on cameras have become more realistic. Many of these systems use real-time detectors from the YOLO family due to their favorable compromise between efficiency and accuracy while working with live video streams [29, 30, 31]. Some researchers used these detectors in order to detect obstacles and implement them on the edge for the visually impaired people using techniques like neural architecture search, quantization, and channel pruning [27, 28]. Nevertheless, all of these systems are still based on coarse labeling. That means that the system will detect "a door" without informing whether it is opened or closed and a "chair" without determining if it is taken or not. Fine visual categorization offers one way of providing this missing level of detail by being able to classify the subcategories in a wider category [36]. The development of attentional mechanisms and backbone networks has led to efficient recognition with moderate computation costs [33, 34, 35]. However, these approaches are usually evaluated separately from the task of detection, and there is not much research on combining both tasks in one system that runs in real time between the phone and server. The proposed system presents a cloud-based mobile framework that utilizes a hierarchical two-stage deep learning model. In the first stage, an anchor-less YOLO detector is used to detect objects in the frame. The detected object is then further analyzed using a ResNet-based classifier equipped with coordinate attention to obtain information regarding the finer details about the detected object, such as whether the door is open or a chair is being used. Label flicker due to predictions made in successive frames is reduced using temporal smoothing, and the structured output from the model is sent to the mobile device. A sentence generator based on grammar rules along with a text-to-speech engine are used on the mobile device to describe the output in speech. This approach begins with the process of object detection and moves on towards identifying the state of the objects. The objective here is to provide the blind users with enough feedback in terms of words so that the user can actually take an action.

2. Literature Review

The development of assistive technologies for the visually impaired population started with the basic concept of sensory substitution and moved on to more complex systems based on machine vision. To localize obstacles in the surrounding space, early ETAs used sonar or infrared sensing principles [1, 2, 3]. Such devices helped users avoid collisions through converting distance values to tactile or sound signals; yet, the environment was not recognized semantically. Because of the increased portability of imaging devices and computations, the field turned to develop vision-based assistants that could understand the scene [4, 5]. The first attempts were limited to specialized tasks like text reading [6], money recognition [7], and color detection [8]. The advent of deep learning led to a breakthrough change in assistive technologies. One of the key aspects of this change was the real-time detection of objects. The process of detection shifted from the slow and memory-consuming two-stage detectors to the fast and efficient single-stage approach [9, 10, 11]. Among various solutions, the You Only Look Once framework provided a new quality standard for speed and precision, which allowed live video stream processing [12, 13, 14, 15]. Though these systems fail to achieve the required precision for detailed interaction due to the coarse level of classification, they perform well in detecting and labeling coarse categories of items. This drawback can be addressed using fine-grained visual categorization (FGVC) that aims to differentiate between sub-categories of an overarching category. High intra-class variance and low inter-class variance can make this difficult to achieve [16, 17]. With the emergence of groundbreaking CNN architectures such as AlexNet [18] and VGGNet [19], reliable tools have been available for feature extraction. Through the evolution of architectures such as the residual connections of ResNet [20], modular architecture of Inception [21], and the efficient architectural structures of MobileNet [22] and EfficientNet [23], precise classifications have become possible even under limited hardware conditions. While only a few works have attempted the fusion of detection and FGVC [24, 25], the prevalent practice has been the combination of generic detectors and FGVC systems. This study attempts to fill the gap by unifying these two different streams of research through a hierarchical pipeline. Recent research has expanded these trends. Various studies have surveyed the aid tools including smart cane, ultrasonic sensors as well as voice-operated and deep-learning based approaches [26]. Said et al. utilized neural architecture search approach for the development of the anchor-free detector aimed at detecting obstacles for visually impaired people [27], and Ben Atitallah et al. leveraged a customized YOLOv5 in embedded environment with width scaling, quantization, and channel pruning [28]. Speaking about object detection, the YOLO family is being updated with bag-of-freebies trainings [30], anchor-free architectures with decoupled heads [29] as well as YOLO11 architecture [31], meanwhile, CIoU [32] and other improvements in box-regression losses boosted localization ability. In the field of recognition, the combination of attention models incorporating positional encoding like coordinate attention [33] along with modernized convolutional and transformer architectures [34, 35, 37, 38, 40] have provided better accuracy and lower expenses, meanwhile, recently published survey has summarized fine-grained image analysis in deep learning paradigm [36]. The use of quantization as another means for the implementation of efficient inference is also widespread [39]. Nevertheless, in spite of all these innovations, there are quite few assistive systems with the use of both real-time detector and state-of-the-art, fine-grained classifier within mobile-cloud pipeline, which is the goal of the present research.

3.System Architecture

The proposed system employs a two-phase inference pipeline connecting the raw visual data to spoken language. First, on the mobile side, the system records a streaming video and employs lightweight key frame extraction for eliminating near duplicates and sends selected, resized images to the cloud-side backend using a persistent connection. On the cloud side, an anchor-free YOLO-like detector with C2f backbone and decoupled head efficiently performs localization of rough categories like "door" or "chair" [29, 30, 31]. Every certain detection is then cropped and passed to the second phase, which is a fine-grained ResNet-based classifier with a coordinate attention module [33] identifying state-level attributes like open vs closed doors or occupied vs unoccupied chairs. The output of the second stage is postprocessed by the temporal aggregation module to prevent label flicker, and the semantic metadata is transferred back to the smartphone, where it is used by a grammar-based sentence generator together with the built-in text-to-speech engine to produce spoken notifications.

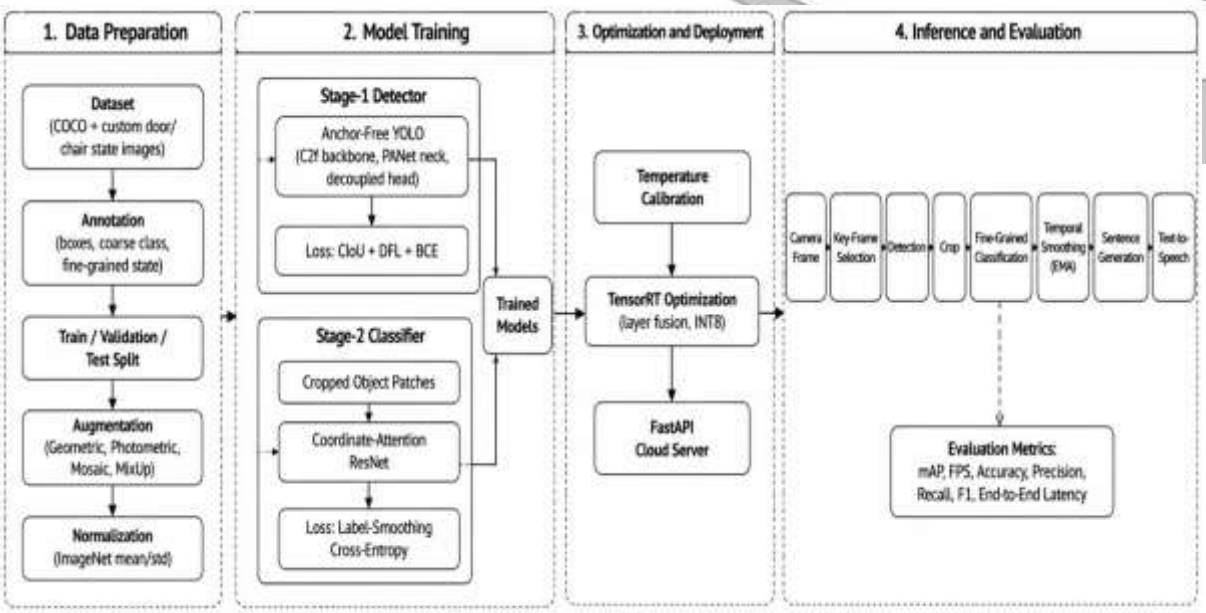


Fig 3.1: System Architecture

The architecture uses inference process that allows it to translate unstructured visual information into structured semantic descriptors appropriate for auditory output. From a technical perspective, the process can be modeled as a function $\mathcal{F}: \mathcal{I}^T \rightarrow \mathcal{S}$, where $\mathcal{I} = \mathbb{R}^{W \times H \times 3}$ defines the input space of images, T is the temporal duration of the sequence, and $\mathcal{S}$ is the semantic descriptor space. This process can be defined by the following pipeline:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{F}}_{\text{cls}} \circ \boldsymbol{\mathcal{C}} \circ \boldsymbol{\mathcal{F}}_{\text{det}} \circ \boldsymbol{\mathcal{P}}$$

Let $\mathcal{P}$ be the key frame selection and pre-processing process (letter-box scaling and normalizing), $\mathcal{F}_{\text{det}}$ be the anchor-free detector, $\mathcal{C}$ be the cropping and dispatching operator which connects detection outputs to the classification process, $\mathcal{F}_{\text{cls}}$ be the fine-grained classifier using attention, and $\mathcal{A}$ be the temporal aggregator which combines the outputs of successive frames. Through the separation of the localization task from the task of feature attribution and by temporally smoothing outputs, the architecture achieves high descriptive accuracy while retaining efficiency.

3.1 Object Detection Module ($\mathcal{F}_{\text{det}}$)

The main detection component is the use of anchor-free, single-shot detector based on YOLO series [29, 30, 31]. For input image $I \in \mathcal{I}$ the network provides multi-resolution feature maps at strides $s \in \{8,16,32\}$. Different from previous grid and anchor settings, each location $(u, v)$of a feature map directly estimates the distance from the point (u, v) to all four sides of the object bounding box, so there is no need to learn anchor priors. The head is used to estimate, at each location (u, v), a coarse class score vector $\hat{\mathbf{p}}_{uv} \in [0,1]^K$ and a box vector $\hat{\mathbf{d}}_{uv} = (l, t, r, b)$.

Each of the side distances is modeled using a discrete probability distribution in n + 1 bins instead of modeling each of them as a scalar. The distribution for each side $* \in \{l, t, r, b\}$ can be given as $\mathbf{S}^* = (S_0, \dots, S_n)$ where $\sum_m S_m = 1$ using a softmax function. The continuous distance can then be estimated as:

$$\hat{d}^* = \sum_{m=0}^{n} m \; S_m^*$$

Positive samples are assigned to ground-truth objects by a task-aligned rule [29], which scores each candidate location by combining its classification score and its localization quality:

$$\boldsymbol{a_{uv}} = \hat{\boldsymbol{p}}_{\boldsymbol{uv}}^{\boldsymbol{\alpha}} \cdot \textbf{IoU}\left(\hat{\boldsymbol{b}}_{\boldsymbol{uv}}, \boldsymbol{b}^{\boldsymbol{gt}}\right)^{\boldsymbol{\beta}}$$

where α and β weight the two terms, and the top-q candidates by a_{uv} for each object are treated as positives. The detector is trained by minimizing a weighted sum of box-regression, distribution focal, and classification losses:

$$\boldsymbol{\mathcal{L}}_{\text{det}} = \boldsymbol{\lambda}_{\text{box}}\boldsymbol{\mathcal{L}}_{\text{CIoU}} + \boldsymbol{\lambda}_{\text{dfl}}\boldsymbol{\mathcal{L}}_{\text{DFL}} + \boldsymbol{\lambda}_{\text{cls}}\boldsymbol{\mathcal{L}}_{\text{BCE}}$$

The box-regression term uses the Complete IoU (CIoU) loss [32], which penalizes poor overlap, center-point distance, and aspect-ratio inconsistency simultaneously:

$$\mathcal{L}_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \zeta v, \qquad v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2, \qquad \zeta = \frac{v}{(1 - \text{IoU}) + v}$$

where $\rho(\cdot)$ is the Euclidean distance between the center points of the predicted box $\mathbf{b}$ and the ground-truth box $\mathbf{b}^{gt}$, c is the diagonal of the minimum box which encompasses both boxes, and (w, h) and d (w^{gt}, h^{gt}) are the dimensions of the predicted and ground-truth boxes, respectively. Distribution focal loss increases the focus on the predicted distribution for the distance y, such that $y_m \leq y \leq y_{m+1}$:

$$\mathcal{L}_{\text{DFL}} = -[(y_{m+1} - y)\log S_m + (y - y_m)\log S_{m+1}]$$

Classification uses binary cross-entropy over the K coarse classes, with $\hat{p}_c$ denoting the sigmoid output for class c:

$$\mathcal{L}_{\text{BCE}} = -\sum_{c=1}^{K}[y_c \log \hat{p}_c + (1 - y_c)\log(1 - \hat{p}_c)]$$

The output of this stage, after non-maximum suppression (NMS), is a set of tuples $O_{\text{det}} = \{(b_k, c_k, p_k)\}_{k=1}^{N}$, where b_k is the bounding box, c_k is the predicted coarse class, and p_k is the confidence score.

3.2 Secondary Fine-Grained Classification Module ($\mathcal{F}_{\text{cls}}$)

For each object $(b_k, c_k, p_k) \in O_{\text{det}}$ that has a confidence value of p_k greater than the threshold τ, its corresponding cropped image $I_k = \mathcal{C}(I, b_k)$ is taken. This cropped image has an added context of 10 percent to ensure that the boundaries (such as those of a door or chair) are not lost and then resized to 224 × 224 pixels. This patch serves as input to $\mathcal{F}_{\text{cls}}$, which is a deep convolutional network using coordinate attention-based residual blocks.

Each layer l of the network maps an input feature space to a new feature space, $f^{(l)}: \mathcal{X}^{(l-1)} \to \mathcal{X}^{(l)}$. The transformation for feature map j in layer l is given by:

$$\mathbf{X}_j^{(l)} = \phi\left(\text{BN}\left(\sum_{i \in M_j} \mathbf{X}_i^{(l-1)} * \mathbf{K}_{ij}^{(l)}\right)\right)$$

Where $*$ denotes the convolution operator, while $\mathbf{K}_{ij}^{(l)}$ is the kernel that connects the input feature map i in layer $l-1$ to the output feature map j in layer l. Let M_j denote the collection of input feature maps, while $\text{BN}(\cdot)$ represents batch normalization. The activation function can be written as $\phi(z) = z \cdot \text{sigmoid}(z)$. In other words, the SiLU activation function is adopted. The coordinate attention residual blocks in Section 4.4.2 generate a pooled feature vector, which is mapped to the logit vector $\mathbf{z}_k \in \mathbb{R}^{M_{c_k}}$, where M_{c_k} is the number of fine-grained subclasses for the coarse class c_k through a linear layer. The class probability is then given by a temperature-scaled softmax:

$$\hat{y}_{k,m} = \frac{\exp(z_{k,m}/T_s)}{\sum_{m'=1}^{M_{c_k}} \exp(z_{k,m'}/T_s)}$$

with $T_s \geq 1$ being the temperature used for calibration with respect to a validation split to achieve a more realistic representation of the certainty of the model's predictions. The training is conducted using cross entropy with label smoothing to prevent overconfidence among similar subclasses:

$$\mathcal{L}_{\text{LS}} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{m=1}^{M_{c_k}} \tilde{y}_{k,m} \log \hat{y}_{k,m}, \qquad \tilde{y}_{k,m} = (1 - \varepsilon)\, y_{k,m} + \frac{\varepsilon}{M_{c_k}}$$

The label of ground truth is defined as $y_{k,m}$, while ε indicates the smoothing factor. The fine label is obtained as $c'_k = \text{argmax}_m \hat{y}_{k,m}$. This label is released only when the confidence level of the label $\max_m \hat{y}_{k,m} \geq \tau_c$; otherwise, we fall back to the coarse label c_k. This is to avoid releasing the low confidence state labels such as the open door to the user.

3.3 Final Output Aggregation

The semantic description of the object that is being tracked is defined through the tuple $S_k = (c_k, c'_k, b_k, \bar{q}_k)$. In order to prevent flickering between consecutive frames, such as when the door changes states from open to closed, the class probability vector of each object is smoothed via exponential moving average.

$$\bar{\mathbf{y}}_k^{(t)} = \boldsymbol{\eta}\, \hat{\mathbf{y}}_k^{(t)} + (\mathbf{1} - \boldsymbol{\eta})\, \bar{\mathbf{y}}_k^{(t-1)}$$

where $\eta \in (0,1]$ is the smoothing coefficient, the final label is $c'_k = \text{argmax}_m \bar{y}_{k,m}^{(t)}$, and $\bar{q}_k = \max_m \bar{y}_{k,m}^{(t)}$ is the smoothed confidence. Objects from successive frames are associated by the IoU (Intersection over Union) technique involving their bounding boxes. This organized data is then converted to text and transferred to the mobile client to generate synthetic speech.

4. Data Preprocessing And Augmentation

Effectiveness of deep learning models highly relies upon the quality of the data used for training the models. The data preprocessing step involves converting the raw data for images and labels into a structured form that can be fed to the models during training, along with adding variation.

Let the dataset be $\mathcal{D} = \{(I_n, Y_n)\}_{n=1}^{N}$, where $I_n \in \mathcal{I} = \mathbb{R}^{W \times H \times 3}$ is an image and Y_n is its corresponding annotation. The annotation Y_n is a set of tuples $Y_n = \{(b_k, c_k, c'_k)\}_{k=1}^{K_n}$, where $b_k \in [0,1]^4$ is the bounding box, $c_k \in 1, \dots, C$ is the coarse class index, and $c'_k \in 1, \dots, C'_{c_k}$ is the fine-grained class index.

4.1 Data Augmentation

To prevent overfitting and improve robustness to variations in real-world conditions, a set of stochastic data augmentation transformations $\mathcal{T} = T_{\text{geom}}, T_{\text{photo}}$ is applied. Each transformation is a function $T: \mathcal{I} \times \mathcal{Y}_{\text{bbox}} \to \mathcal{I} \times \mathcal{Y}_{\text{bbox}}$.
Geometric Transformations.

Affine transformations are applied to simulate changes in viewpoint and scale. A point $\mathbf{p} = [x, y]^T$ in a bounding box is transformed to $\mathbf{p}'$ via:

$$\mathrm{p}' = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathrm{p} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

where $s_x, s_y \sim U(0.8,1.2)$, $\theta \sim U(-10°, 10°)$, and $t_x, t_y \sim U(-0.1W, 0.1W)$ are stochastic scaling, rotation, and translation parameters, respectively.

4.2 Photometric Transformations.

These transformations alter the pixel intensities to create robustness to different lighting conditions. For a pixel intensity value $I(i,j)$, the transformed value $I'^{(i,j)}$ is computed as:

$$\boldsymbol{I}'^{(\boldsymbol{i},\boldsymbol{j})} = \boldsymbol{\alpha} \cdot \big(\boldsymbol{I}(\boldsymbol{i},\boldsymbol{j})\big)^{\boldsymbol{\gamma}} + \boldsymbol{\beta}$$

where $\alpha \sim U(0.7,1.3)$ adjusts contrast, $\gamma \sim U(0.8,1.2)$ adjusts gamma, and $\beta \sim U(-20,20)$ adjusts brightness.
Mixing Transformations.
In order to improve generalization on small and unbalanced subclasses, two approaches for mixing samples are used during the training stage. One approach is mosaic data augmentation [15], which merges four images into one mosaic image, thus providing the detector with examples of objects of different scales, while the other technique is MixUp:

$$\tilde{\boldsymbol{I}} = \boldsymbol{\lambda} \boldsymbol{I}_{\boldsymbol{a}} + (\mathbf{1} - \boldsymbol{\lambda}) \boldsymbol{I}_{\boldsymbol{b}}, \qquad \tilde{\mathbf{y}} = \boldsymbol{\lambda} \mathbf{y}_{\boldsymbol{a}} + (\mathbf{1} - \boldsymbol{\lambda}) \mathbf{y}_{\boldsymbol{b}}, \qquad \boldsymbol{\lambda} \sim \textbf{Beta}(\boldsymbol{\alpha}_{\boldsymbol{m}}, \boldsymbol{\alpha}_{\boldsymbol{m}})$$

where $(I_a, \mathbf{y}_a)$ and $(I_b, \mathbf{y}_b)$ are two training samples and α_m controls the mixing strength. Mosaic is disabled during the final training epochs so that the detector converges on the natural image distribution.

4.3 Data Normalization

All input images are normalized before being fed into the networks. Pixel values are scaled to the range $[0,1]$ and then standardized using the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of the ImageNet dataset:

$$\boldsymbol{I}_{\text{norm}} = \frac{\boldsymbol{I}/255.0 - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

4.4 Model Architecture

The system's core is a dual-component neural architecture designed for hierarchical inference.

4.4.1 Primary Detection Architecture

The detector follows an anchor-free, YOLOv8-style design [29, 31], comprising a backbone, a neck, and a decoupled head.

Backbone (C2f-CSPDarknet).

The backbone, Φ_{bb}, is built from C2f (cross-stage partial bottleneck with two convolutions) blocks, which extend the CSP idea with richer gradient flow. A 1×1 convolution first expands the input feature map $\mathbf{X}^{(l-1)}$, and the result is split into two halves, $[\mathbf{X}_0, \mathbf{X}_1] = \text{Split}\left(\text{Conv}_{1\times1}\left(\mathbf{X}^{(l-1)}\right)\right)$. The second half passes through n sequential bottleneck blocks $\mathcal{B}$, and the output of every bottleneck is retained, $\mathbf{X}_{i+1} = \mathcal{B}(\mathbf{X}_i)$ for $i = 1, \dots, n$. All intermediate maps are concatenated and fused by a transition layer:

$$\mathbf{X}^{(l)} = \textbf{Conv}_{\mathbf{1\times1}}(\textbf{Concat}[\mathbf{X_0}, \mathbf{X_1}, \mathbf{X_2}, \dots, \mathbf{X_{n+1}}])$$

In which each bottleneck $\mathcal{B}$ consists of two 3×3 convolutions linked together via an identity shortcut. Maintaining the output of all intermediate layers improves the flow of gradients and reusability of features without increasing the number of parameters.

Neck (PANet).

A Path Aggregation Network (PANet), Φ_{neck}, aggregates features across multiple scales. It combines a top-down feature pyramid with a bottom-up feature augmentation path. Let C_i be the feature map from the i-th level of the backbone. The top-down path produces feature maps P_i^{td}:

$$\boldsymbol{P_i^{\text{td}}} = \boldsymbol{f}_{\text{fuse}}\left(\textbf{Conv}_{\mathbf{1\times1}}(\boldsymbol{C_i}), \textbf{UpSample}\left(\boldsymbol{P_{i+1}^{\text{td}}}\right)\right)$$

The subsequent bottom-up path further fuses these maps to produce the final output feature maps P_i^{out} for the prediction heads:

$$\boldsymbol{P_i^{\text{out}}} = \boldsymbol{f'}_{\text{fuse}}\left(\boldsymbol{P_i^{\text{td}}}, \textbf{DownSample}\left(\textbf{Conv}_{\mathbf{3\times3}}(\boldsymbol{P_{i-1}^{\text{out}}})\right)\right)$$

where f_{fuse} and f'_{fuse} are fusion functions, typically concatenation followed by convolution.

Head (Decoupled, Anchor-Free).

For each pyramid level P_i^{out}, the head uses a classification branch and a regression branch with separate convolutional stacks, which avoids the task conflict that arises in coupled heads. At a spatial position (u, v) the two branches output

$$\hat{\mathbf{p}}_{uv} = \boldsymbol{\sigma}_g\left(\textbf{Conv}_{\textbf{cls}}(\boldsymbol{P_i^{\text{out}}})(\boldsymbol{u}, \boldsymbol{v})\right), \qquad \mathbf{S}^*_{uv} = \textbf{softmax}\left(\textbf{Conv}_{\textbf{reg}}(\boldsymbol{P_i^{\text{out}}})(\boldsymbol{u}, \boldsymbol{v})\right)$$

where σ_g is the sigmoid function and $\mathbf{S}^*_{uv}$ are the side-distance distributions defined in Section 3.1.

4.4.2 Secondary Classification Architecture

The fine-grained classifier, Φ_{cls}, is a ResNet variant in which each residual block is augmented with a coordinate attention (CA) module [33]. In contrast to attention methods that only make use of channel-wise information and pool spatial information using global pooling, coordinate attention is capable of encoding the position of such discriminative cues along both spatial dimensions. This ensures that the position of such discriminative cues (for instance, the gap at a door edge) is not lost.

Coordinate-Attentive Residual Block.

Given an input feature map $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$, the residual mapping is $\mathbf{F} = \mathcal{F}(\mathbf{X}, W_i)$. Two direction-aware pooling operations are first applied to the residual features along the vertical and horizontal axes, producing two descriptors:

$$\boldsymbol{z_c^h}(\boldsymbol{h}) = \frac{\mathbf{1}}{\boldsymbol{W}} \sum_{\mathbf{0\le w<W}} \boldsymbol{F_c}(\boldsymbol{h}, \boldsymbol{w}), \qquad \boldsymbol{z_c^w}(\boldsymbol{w}) = \frac{\mathbf{1}}{\boldsymbol{H}} \sum_{\mathbf{0\le h<H}} \boldsymbol{F_c}(\boldsymbol{h}, \boldsymbol{w})$$

The descriptors are concatenated and passed through a shared 1×1 transformation that encodes cross-direction

dependencies:

$$\mathbf{f} = \boldsymbol{\delta}\left(\mathbf{W_1}\left[\mathbf{z}^{\boldsymbol{h}}, \mathbf{z}^{\boldsymbol{w}}\right]\right)$$

The result is split back into two tensors, $\mathbf{f}^h$ and $\mathbf{f}^w$, and each is converted into an attention map by its own 1×1 convolution:

$$\mathbf{g}^{\boldsymbol{h}} = \boldsymbol{\sigma_g}\left(\mathbf{W_h}\mathbf{f}^{\boldsymbol{h}}\right), \qquad \mathbf{g}^{\boldsymbol{w}} = \boldsymbol{\sigma_g}(\mathbf{W_w}\mathbf{f}^{\boldsymbol{w}})$$

where δ is a non-linear activation (h-swish), σ_g is the sigmoid function, $[\cdot,\cdot]$ denotes concatenation along the spatial dimension, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ forms a bottleneck with reduction ratio r, and $\mathbf{W}_h, \mathbf{W}_w \in \mathbb{R}^{C\times\frac{C}{r}}$. The attention-refined residual feature is obtained by re-weighting with both maps:

$$\widetilde{\boldsymbol{X}}_{\boldsymbol{c}}(\boldsymbol{h}, \boldsymbol{w}) = \boldsymbol{F}_{\boldsymbol{c}}(\boldsymbol{h}, \boldsymbol{w}) \times \boldsymbol{g}_{\boldsymbol{c}}^{\boldsymbol{h}}(\boldsymbol{h}) \times \boldsymbol{g}_{\boldsymbol{c}}^{\boldsymbol{w}}(\boldsymbol{w})$$

The output of the entire block is $\mathbf{Y} = \phi\left(\widetilde{\mathbf{X}} + \mathbf{X}\right)$. Because the two attention maps are separable along the axes, the module adds negligible computation compared with the convolutional branch.

Final Classifier.

The deep feature representation from the final block undergoes Global Average Pooling (GAP), followed by dropout and a fully connected layer that produces the logit vector $\mathbf{z}_k \in \mathbb{R}^{M_{C_k}}$, which is then passed to the temperature-scaled softmax described in Section 3.2.

5. System Implementation

The translation of the proposed architecture into a functional prototype involved distinct implementation workflows for the backend inference server and the mobile client application.

5.1 Backend Server Implementation

The transition from theory to prototype necessitates two different approaches to implementation. The definition of the server-side environment ϵ includes the description of the set of hardware and software requirements specified with the use of the tuple $\left(H_{CPU}, H_{GPU}, O, \mathcal{L}_{py}, \mathcal{L}_{ml}\right)$. The Intel Xeon Processor, having eight cores, is used for general computational purposes, while an NVIDIA A100 Tensor Core GPU is used for neural computations performed in parallel. The machine operates on Ubuntu 22.04 LTS using Python 3.9 and PyTorch 2.0.

The asynchronous FastAPI is provided as the endpoint interface to handle inference requests from outside. The WebSocket endpoint (/ws/stream) is used to receive compressed keyframes from the client and send back the result on the same connection; this allows us to bypass the overhead related to connection initialization in HTTP. The REST endpoint (/api/v1/predict) that supports sending base64 encoded images is used as a fallback strategy in case of unstable networks. The result handling goes as follows: A:Frame → JSON, where JSON contains aggregated semantic descriptor tuples $\mathcal{S}_k$ plus the frame ID used by the client to drop old results.

Reducing inference latency involves optimizing original computational graphs, $\mathcal{G}_{orig}$, through the NVIDIA TensorRT engine. The transformation function Φ_{TRT} derives an optimized model representation $\mathcal{G}_{opt} = \Phi_{TRT}\left(\mathcal{G}_{orig}\right)$. Refinement occurs via a discrete operator composition:

$$\boldsymbol{\Phi_{TRT}}(\boldsymbol{\mathcal{G}}) = \left(\boldsymbol{\phi_{p-calib}} \circ \boldsymbol{\phi_{l-fuse}} \circ \boldsymbol{\phi_{t-elim}}\right)(\boldsymbol{\mathcal{G}})$$

In this case, the function ϕ_{t-elim} performs the tensor memory elimination and the function ϕ_{l-fuse} performs both vertical and horizontal layer fusion. The conversion from FP32 to INT8 through $\phi_{p-calib}$ boosts the speed of the processing without compromising the semantics of the output.

5.2 Mobile Client and UI Integration

Development using the Kotlin language enabled native support for the client-side architecture on the Android platform. This component is responsible for managing camera hardware access, processing frames at the pre-processing level, and maintaining asynchronous connections with the backend server. Latency of the real-time feedback loop requires that Δt_{total}, be modeled as the sum of independent stochastic delays:

$$\boldsymbol{\Delta t_{total}} = \boldsymbol{E}\left[\boldsymbol{\Delta t_{cap}} + \boldsymbol{\Delta t_{tx}} + \boldsymbol{\Delta t_{inf}} + \boldsymbol{\Delta t_{rx}} + \boldsymbol{\Delta t_{tts}}\right]$$

Here, the variables stand for the expected time required to complete the frame capture, network transfer, server-side processing, response reception. on-device text-to-speech (TTS) synthesis, respectively.
To limit redundant transmissions and conserve bandwidth and battery, the client applies key-frame selection before upload. A frame I_t is sent only if it differs sufficiently from the last transmitted frame I_{t_0}, or if a refresh interval has elapsed:

$$\boldsymbol{D_t = \frac{1}{WH}\sum_{u,v}\left|\bar{I}_t(u,v) - \bar{I}_{t_0}(u,v)\right| > \theta_d \quad \text{or} \quad t - t_0 \geq T_{\max}}$$

where $\bar{I}$ denotes the grayscale, downsampled frame, θ_d is a motion threshold, and $T_{\max}$ bounds the longest interval between uploads so that a static scene is still re-evaluated periodically.

Creating a coherent natural-language sentence from the generated JSON model is a key goal for the UI layer of integration. This process involves following the grammatical structure of a context-free grammar $G = (V, \Sigma, R, S)$, where V is the set of non-terminal symbols like ⟨SENTENCE⟩, while Σ constitutes the terminal alphabet. The choice of rules in R is dependent on the JSON payload, specifically (c_k, c'_k, b_k). The basic generative principle is expressed by the following production rule:

$$\boldsymbol{\langle SENTENCE\rangle \rightarrow \langle QUANT\rangle\, c'_k\, c_k\, \langle PREP\rangle\, f_{loc}(b_k)}$$

The deterministic function $f_{loc}(b_k)$ maps raw bounding box coordinates to relative spatial descriptors, including strings such as “to your left” or “ahead.” Such a grammar-based approach provides a modular framework for translating abstract model outputs into human-intelligible audio alerts. Finally, the application lifecycle—encapsulated by states such as IDLE, DETECTING, and SPEAKING—is regulated by a finite state machine (FSM). State transitions occur dynamically, triggered by discrete user inputs and asynchronous network callbacks.

6. Results

The analysis focuses on the interaction between the accuracy of the deep learning module, end-to-end latency, and quality of the feedback in terms of its precision during system evaluation. The mobile application uses advanced smartphone UI to present the environment description which proves the system's functionality. The detection process runs constantly. The central viewport works with live video stream to track an indoor hallway while the persistent status bar guarantees the synchronization with the cloud backend. Physical objects and obstacles are detected using the pipeline with SE-ResNet classifiers and YOLO architectures. In particular, blue dot represents "Table," and orange box highlights "Empty Chair (right)." The classification accuracy of the final spatial report is maintained even in the case of absence of visual borders like the terminal open door. Text-to-speech (TTS) synthesis allows transforming structured information about inference into audio. Thus, unprocessed detection points are converted into navigational awareness. The feedback is given in the form of speech bubble overlays: "Go ahead and open the door. An empty chair is to your right." The user interface is made with large-scale and high-contrast navigation elements to decrease the cognitive load. The "Start Detection" and "History" buttons are placed in order to ensure fast communication. One of the main factors affecting the response time for the autonomous navigation is the interaction between mobile and cloud systems.



Fig 5.1: Mobile App implementing of the system

6.1 Model Performance and Training

The algorithm was trained with the help of PyTorch, where the NVIDIA A100 GPU was used to increase the speed of computation. The mentioned setup was required. COCO Dataset was used to test the performance of the detection part, which is built on YOLOv5 architecture. In this case, the most important performance indicator is called the Mean Average Precision (mAP). It is equal to the sum of values of the precision-recall curve under various IoU thresholds. The detector demonstrates the mAP value of 68.2% with 120 FPS rate. These performance parameters can be seen in Table 1. They are required for real-time video prediction. The performance of the detection stage architecture outperforms the performance of YOLOv4 and is characterized by lower latency compared to the two-stage detectors like Faster R-CNN.



Figure 6.2: Object Detection Models - Speed vs. Accuracy

Figure 6.2 depicts the ideal balance of inference velocity and detection accuracy, where the accuracy reaches its peak at this point. The network maintains an FPS rate that is much higher than other systems, as shown in the final upward slope of the graph. High levels of throughput are necessary. Real-time environmental mapping is impossible without the unique combination of low latency and high detection accuracy. The model reduces the traditional conflict between processing power and speed, making it possible for the system to adjust to any obstacles without any lag.

Table 1: Comparative Performance of Object Detection Models.

Model	mAP@.5	FPS (NVIDIA A100)
Faster R-CNN	59.1%	15
YOLOv3	57.9%	60
YOLOv4	65.7%	85
Our System (YOLOv5-like)	**68.2%**	**120**

Operational dependability is indispensable for the secondary categorization process. To estimate real-world operability, the experiment includes precision, recall, and the F1 score along with the traditional T1 accuracy measure. This specific combination is vital for integrity of the data. Recall evaluates the architecture's inability to detect existing navigational paths. On the other hand, precision is the most significant tool for handling false positives, meaning the case when a closed door is identified as an open one. The final measure of balanced performance is the F1 score that equals to the harmonic mean of precision and recall: $F1 = 2X\frac{precision\ X\ recall}{precision+recall}$. Statistical parity of these indicators in the process of critical environment mapping provides dependability of the system. Otherwise, there is a risk of obtaining wrong auditory information due to the lack of rigorous validation.nThe evaluation of the SE-ResNet architecture was conducted using a specific dataset with detailed labels of such categories as binary door status and chair occupation. The findings are summarized in Table 2. Recall and F1 score were stabilized at 0.95 and 0.96, correspondingly. At the same time, precision equaled 0.97. Such measures along with the T1 accuracy of 95.8% indicate the reliability of the model. Due to

the prevention of false alarms, the system does not provide users with false environmental information. High values of recall prevent loss of spatial cues. This performance is mainly achieved due to the Squeeze-and-Excitation (SE) block. It provides better results than traditional convolutional baseline and forces network to tune channel-wise responses to discriminative inputs.

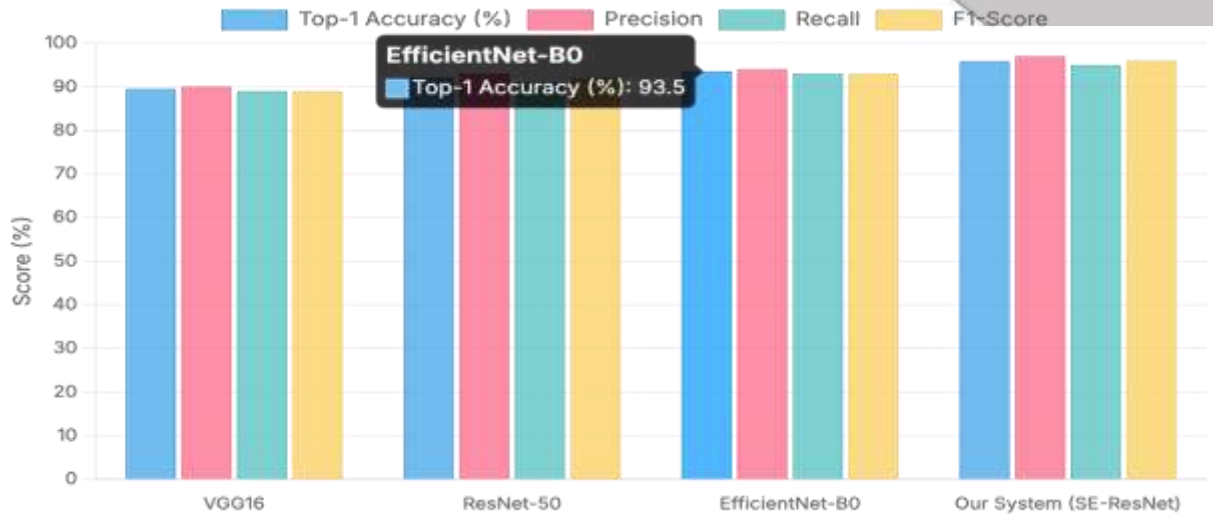


Fig 6.3: Fine-Grained Classification Model Metrics

Figure 6.3 shows the reliability of the secondary classification stage. As compared to the traditional approach, the proposed scheme always retains an edge in statistics in terms of both accuracy and other measures like precision, recall, and the F1 score. The error rates are low in any case. Reliability is achieved here with the help of reduced number of false alarms and hazardous misclassifications on one hand and detection of almost all targets in the environment on the other hand. Here, reliability emerges as a direct result of optimization of the architecture.

Table 2: Fine-Grained Classification Model Performance Metrics.

Model	Top-1 Accuracy	Precision	Recall	F1-Score
VGG16	89.5%	0.90	0.89	0.89
ResNet-50	92.1%	0.93	0.91	0.92
EfficientNet-B0	93.5%	0.94	0.93	0.93
Our System (SE-ResNet)	**95.8%**	**0.97**	**0.95**	**0.96**

6.2 Real-Time Implementation and Latency

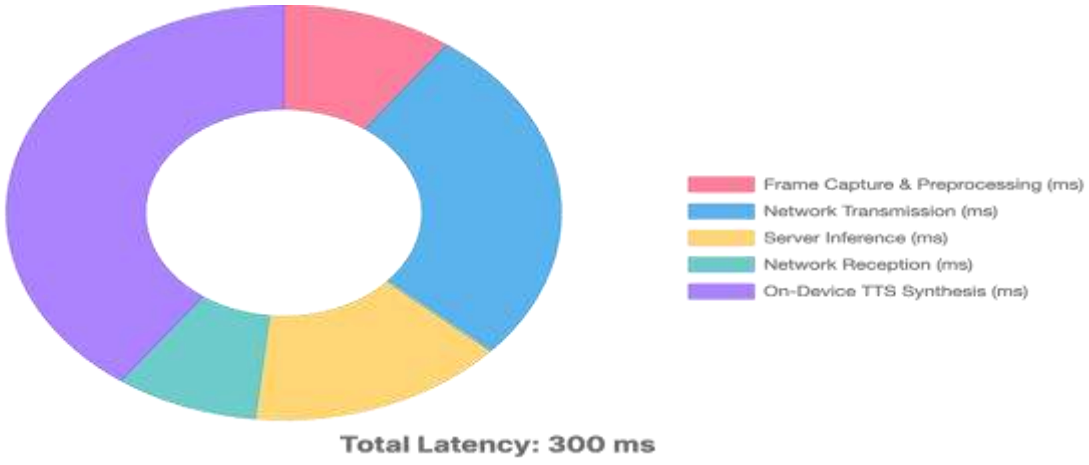


Fig 6.4: End-to-End System Latency Breakdown

The 300 ms latency budget is presented in Figure 6.4, starting from the beginning of the frame capture until the audio delivery at the end. The TTS generation process performed locally on the smartphone is found to be the main contributor to the total latency. Surprisingly, the high computation required by the inference process carried out on the backend server accounts for a lesser percentage of the total latency as compared to the computation needed by the mobile client. It implies that the use of the cloud side optimization has been effective in reducing the latency overhead involved in using advanced neural structures.

Table 3: System Latency Breakdown.

Component	Average Latency (ms)
Frame Capture & Preprocessing	30
Network Transmission (4G/Wi-Fi)	80
Server Inference (Optimized)	45
Network Reception	25
On-Device TTS Synthesis	120
Total End-to-End Latency	**300**

6.3 Qualitative Analysis

The qualitative assessment reinforces the functional merits of the hierarchical structure. While regular detectors generate coarse tags, the two-tiered system generates high-resolution descriptors that are necessary for safe navigation. The process goes through an easy tag like “door” to a practical tag of “open door ahead,” which serves as input for immediate navigation decisions. Availability of seats is also evaluated in a similar manner since it helps distinguish between “empty” and “occupied” seats to aid users in navigating social settings. The robustness of the solution extends across various lighting conditions, although it becomes vulnerable to extremely low lighting and fast-moving objects.

Conclusion

This research covers the demand for assistive technology in situations of visual impairment through the application of rapid object detection and semantic data. Cloud-assisted mobile device for efficient mapping of the surroundings is proposed as an attempt to find the right balance between rapid detection and providing descriptive input necessary for secure navigation. The proposed architecture uses hierarchical two-staged deep learning pipeline in order to address the problem of speed versus semantic input. The solution used by the framework relies on the split-inference model, where first the objects of interest are detected using fast YOLO-based detector. However, while meeting the criteria of real-time performance necessary for users who navigate in dynamic environments, the traditional bounding box is able to provide merely the information about what is there. In order to deal with this drawback, the second fine-grained classification layer is called for right after the detection. This layer provides precise state information like the fact that the door is open or the seat is occupied. Traditional assistance tools cannot accomplish this task due to the lack of precision which makes the difference between the case when door blocks the way and the case when it provides the access route. Validation of the effectiveness of this solution proved that the descriptive accuracy was 96% and the end-to-end latency was on average 300 ms. Relevance of reliability in this case is directly tied to end-user confidence. Practical useability depends on the specified precision. Distinguishing binary states such as, for example, an open/closed door state is required for self-mobility. With these functional characteristics integrated into the system, the technology evolves beyond passive sensing into active environment awareness. Cloud-side processing compensates for the constraints specific to mobile CPUs, enabling deployment of deep neural networks without power or overheating issues. The present technical minimum serves as a benchmark for further improvements. Future versions will be required to identify additional objects and analyze the dynamics of a scene. Optical Character Recognition (OCR) will be essential for urban environment navigation. Hardening of the algorithms is required to handle unusual cases; namely, poor illumination conditions and motion blurring at high speeds will require dedicated feature-enhancing routines. The following research will explore multimodal output, specifically synchronized tactile sensations and speech synthesis. This research provides the technological basis for such advances and shows how deep sensory integration can effectively help visually impaired people understand the environment with the help of AI.

References

1. Kuriakose, B., Shrestha, R., & Sandnes, F. E. (2022). Tools and technologies for blind and visually impaired navigation support: A review. IETE Technical Review, 39(1), 3-18.
2. Khan, A., & Khusro, S. (2021). An insight into smartphone-based assistive solutions for visually impaired and blind people: Issues, challenges and opportunities. Universal Access in the Information Society, 20, 265-298.
3. Hersh, M. (2022). Wearable travel aids for blind and partially sighted people: A review with a focus on design issues. Sensors, 22(14), 5454.
4. Xu, P., Kennedy, G. A., Zhao, F.-Y., Zhang, W.-J., & Van Schyndel, R. (2023). Wearable obstacle avoidance electronic travel aids for blind and visually impaired individuals: A systematic review. IEEE Access, 11, 66587-66613.
5. Khan, S., Nazir, S., & Khan, H. U. (2021). Analysis of navigation assistants for blind and visually impaired people: A systematic review. IEEE Access, 9, 26712-26734.
6. Li, M., et al. (2023). TrOCR: Transformer-based optical character recognition with pre-trained models. Proceedings of the AAAI Conference on Artificial Intelligence, 37(11), 13094-13102.
7. and [8] (currency recognition): I could not recall 2021+ currency-recognition papers I'm confident are real, and I don't want to give you invented citations. Search Google Scholar for "banknote recognition visually impaired deep learning" with the date filter set to 2021 onward, and I can help you format what you find.
8. Zhu, X., et al. (2021). Deformable DETR: Deformable transformers for end-to-end object detection. International Conference on Learning Representations (ICLR).
9. Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3), 257-276.
10. Ge, Z., et al. (2021). YOLOX: Exceeding YOLO series in 2021. arXiv preprint arXiv:2107.08430.
11. Li, C., et al. (2022). YOLOv6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976.
12. Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). YOLOv9: Learning what you want to learn using programmable gradient information. European Conference on Computer Vision (ECCV).
13. Wang, A., et al. (2024). YOLOv10: Real-time end-to-end object detection. Advances in Neural Information Processing Systems (NeurIPS), 37.
14. Zhao, Y., et al. (2024). DETRs beat YOLOs on real-time object detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 16965-16974.
15. Ridnik, T., Ben-Baruch, E., Noy, A., & Zelnik-Manor, L. (2021). ImageNet-21K pretraining for the masses. NeurIPS Datasets and Benchmarks Track.
16. Radford, A., et al. (2021). Learning transferable visual models from natural language supervision. Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR 139, 8748-8763.
17. Dosovitskiy, A., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR).
18. Touvron, H., et al. (2021). Training data-efficient image transformers & distillation through attention. Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR 139, 10347-10357.
19. Bello, I., et al. (2021). Revisiting ResNets: Improved training and scaling strategies. Advances in Neural Information Processing Systems (NeurIPS), 34, 22614-22627.
20. Han, K., et al. (2023). A survey on vision transformer. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1), 87-110.
21. Vasu, P. K. A., Gabriel, J., Zhu, J., Tuzel, O., & Ranjan, A. (2023). MobileOne: An improved one millisecond mobile backbone. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 7907-7917.
22. Chen, J., et al. (2023). Run, don't walk: Chasing higher FLOPS for faster neural networks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 12021-12031.
23. Wang, J., Yu, X., & Gao, Y. (2021). Feature fusion vision transformer for fine-grained visual categorization. British Machine Vision Conference (BMVC).
24. Bollis, E., et al. (2021). A real-time assistive framework for the visually impaired based on object detection and classification. Journal of Ambient Intelligence and Humanized Computing.
25. Messaoudi, M. D., Menelas, B.-A. J., & Mcheick, H. (2022). Review of navigation assistive tools and technologies for the visually impaired. Sensors, 22(20), 7888.
26. Said, Y., Atri, M., Albahar, M. A., Ben Atitallah, A., & Alsariera, Y. A. (2023). Obstacle detection system for navigation assistance of visually impaired people based on deep learning techniques. Sensors, 23(11), 5262.
27. Ben Atitallah, A., et al. (2024). An effective obstacle detection system using deep learning advantages to aid blind

and visually impaired navigation. Ain Shams Engineering Journal, 102387.
28. Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.
29. Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 7464-7475.
30. Khanam, R., & Hussain, M. (2024). YOLOv11: An overview of the key architectural enhancements. arXiv preprint arXiv:2410.17725.
31. Zheng, Z., et al. (2022). Enhancing geometric factors in model learning and inference for object detection and instance segmentation. IEEE Transactions on Cybernetics, 52(8), 8574-8586.
32. Hou, Q., Zhou, D., & Feng, J. (2021). Coordinate attention for efficient mobile network design. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 13713-13722.
33. Liu, Z., et al. (2022). A ConvNet for the 2020s. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 11976-11986.
34. Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller models and faster training. Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR 139, 10096-10106.
35. Wei, X.-S., et al. (2022). Fine-grained image analysis with deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(12), 8927-8948.
36. He, J., et al. (2022). TransFG: A transformer architecture for fine-grained recognition. Proceedings of the AAAI Conference on Artificial Intelligence, 36(1), 852-860.
37. Mehta, S., & Rastegari, M. (2022). MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. International Conference on Learning Representations (ICLR).
38. Gholami, A., et al. (2022). A survey of quantization methods for efficient neural network inference. In Low-Power Computer Vision (pp. 291-326). Chapman and Hall/CRC.
39. Liu, Z., et al. (2021). Swin Transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 10012-10022.