August 1, 2025

University of Sciences and Arts in Lebanon

Faculty of Sciences and Arts

SafeWalk AI

Computer Science Final Year Project – Data Science

Mohamad Moallem – 1122430

Ali Moallem – 1122431

Nour Rammal – 1122365

Supervisor: Dr. Ali Ezzeddine

Abstract

This project aims to develop an **AI-powered accessibility tool** to assist visually impaired individuals in navigating their environment independently. Leveraging **Computer Vision (CV)** and **Natural Language Processing (NLP)**, the system detects objects, extracts text, and identifies obstacles in real-time, providing audio feedback. The tool integrates YOLOv8 for object detection, Tesseract OCR for text recognition, and depth sensing for obstacle avoidance. Key innovations include:

- **Real-time processing** optimized for low-latency deployment on edge devices.
- **Multi-modal feedback** combining object, text, and obstacle detection.
- **User-centric design** with intuitive audio output.

The project addresses a critical gap in assistive technology, combining state-of-the-art models with practical usability. Evaluation metrics include accuracy (mAP for object detection, CER for OCR) and real-world usability testing with visually impaired users.

NOTE: Please refer to midas_selection_criteria.pdf, Indoor_object_detection_report.pdf, report_scene_classification.pdf, and outdoor-object-detection-report.pdf for more info on the experiments done using various datasets leading to the chosen models mentioned and explained in this report

Table of Contents

Table of Figures

1 Introduction

1.1 Background

Over **285 million people** worldwide live with visual impairments, yet many existing assistive technologies lack real-time, multi-modal feedback. Solutions like Seeing AI and Envision focus on specific tasks (e.g., text reading) but do not integrate comprehensive object and obstacle detection. This project bridges this gap by combining **YOLOv8**, **OCR**, and **depth sensing** into a unified system.

1.2 Objective

1. **Classify the environment** as **indoor** or **outdoor** using a deep learning-based scene classifier.
2. **Detect and localize common objects** (e.g., chairs, doors) and provide **real-time audio feedback**.
3. **Identify obstacles** and estimate their distance using **depth estimation models** like **MiDaS**.
4. Deliver **intuitive, spoken navigation cues** using a **text-to-speech (TTS)** engine for enhanced spatial awareness.

1.3 Scope

- **Included**:
  - ✓ Real-time object/text detection on smartphones for indoor/outdoor environments.
  - ✓ Depth estimation of obstacles
  - ✓ Open-source frameworks (PyTorch, OpenCV).
  - ✓ Offline mode.
  - ✓ Initial deployment prioritizes English for global accessibility, with a design-ready framework for adding languages (e.g., Spanish, Arabic) via plug-in datasets and TTS engines.

- **Excluded**:
  - OCR(optional can be added to the final pipeline).

- NLP models(using Rule-based instructions since including models like chatgpt mini would be overkill to generate simple instructions avoiding high cost and complexity of LLMs).

1.4 Methodology

1. **Data Collection**: COCO (objects), ICDAR (text), Cityscapes (obstacles).
2. **Models**:
   - **YOLOv8n:** Two models — one trained for indoor environments and another for outdoor object detection.
   - **Easy-OCR:** For optional text recognition in scenes (e.g., signs, labels).
   - **MiDaS (Small):** Lightweight depth estimation model for obstacle detection and distance calculation.
   - **MobileNetV3Large + Custom CNN**: Scene classifier to distinguish between **indoor** and **outdoor** environments.
3. **Evaluation**: mAP (objects), CER (text), FPS (real-time performance).

1.5 Limitations

- Dependency on camera quality.
- Real-time latency on low-end devices.
- User personalization.

2 Related Research

Weak Competitor — Strong Competitor

2.1 Similar Existing Products

2.1.1 Be My Eyes

- **Description**: A mobile app that connects visually impaired users with sighted volunteers through video calls to help them navigate daily tasks.
- **Strengths**: Real-time human assistance; very supportive community.
- **Weaknesses**: Requires human availability; depends heavily on internet connectivity; not fully autonomous.

2.1.2 Aira

- **Description**: Professional agents provide on-demand remote assistance to visually impaired users via smart glasses or smartphone.
- **Strengths**: High-quality professional support; integration with smart wearables.
- **Weaknesses**: Subscription cost is expensive; limited by human agent availability; privacy concerns.

2.1.3 Microsoft Seeing AI

- **Description**: A mobile app that uses AI to describe people, text, and objects for visually impaired users.
- **Strengths**: AI-based, automatic scene description; offline text recognition.
- **Weaknesses**: Scene analysis is basic; obstacle avoidance and spatial understanding are limited; no live path guidance.

2.1.4 OrCam MyEye

- **Description**: A wearable device that reads text, recognizes faces, and identifies products in real-time.
- **Strengths**: Portable, discreet, real-time operation.
- **Weaknesses**: Very expensive; limited obstacle detection; mostly focused on object and text reading, not navigation.

2.1.5 Cane-Based Smart Systems (e.g., SmartCane)

- **Description**: Modified canes with sensors to detect nearby obstacles and alert the user via vibrations.
- **Strengths**: Simple, affordable, durable.
- **Weaknesses**: Limited range; does not offer rich environmental understanding; lacks spatial or language reasoning.

2.1.6 Lookout by Google

- **Description**: A Google app that helps visually impaired users identify objects, read text, and explore surroundings using the phone camera.
- **Strengths**:Offline support, multiple modes (text, currency, food), user-friendly.
- **Weaknesses**: No depth or obstacle detection, limited real-time navigation support.

2.2 Project's Added Value

2.2.1 Unique Contributions

This project proposes a **next-generation assistive navigation system** by **integrating two powerful AI models**:

- **YOLOv8:** Enables real-time detection of critical objects and obstacles (e.g., people, doors, stairs) tailored for both indoor and outdoor environments.
- **MiDaS_small:** Provides lightweight yet powerful depth estimation for understanding obstacle distance, enabling spatial awareness without additional hardware(like *intel RealSense*).
- **Rule-Based Spatial Logic:** Converts detection and depth data into intuitive spoken instructions (e.g., "Move slightly right to avoid the chair"), avoiding the need for heavy NLP models.

Key Innovations:

- Real-time **obstacle identification AND spatial reasoning** combined.
- Generates intuitive, rule-based navigation instructions rather than just simple alerts—without relying on heavy NLP models.
- Uses **deep learning object detection**, unlike cane sensors or traditional image recognition.
- **Hands-free** — can be integrated into wearable devices like smart glasses or phones.

- **Cost-effective** compared to commercial solutions like OrCam.
- **offline capabilities** after model optimization.

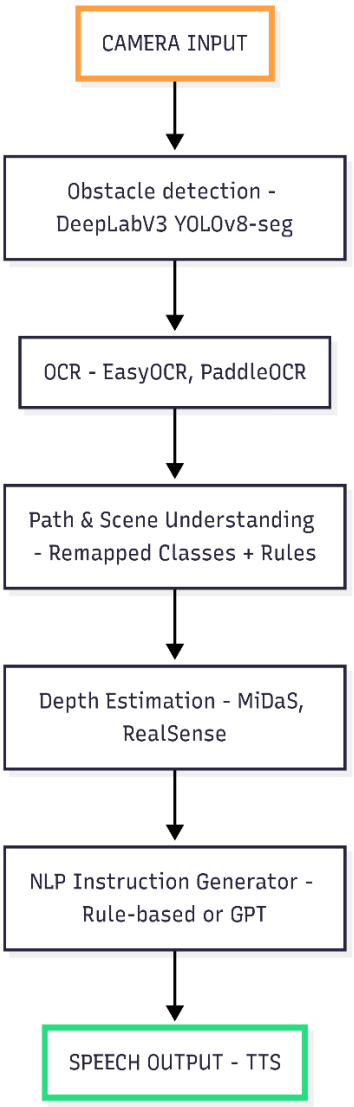
2.2.1 Improvement Over Existing Solutions

Aspect	**Existing Products**	**This Project**
Obstacle Detection	Limited or basic	YOLOv8 high-accuracy detection
Depth estimation	Rare or non-existent	Light weight depth estimation for distance calculation.
Navigation Guidance	Human-dependent or minimal	Automated rule based instructions guidance
Cost	Expensive (OrCam, Aira)	Potentially affordable with smartphones(free to download mobile app)
Autonomy	Needs human agents or limited	Fully AI-driven, autonomous
Customizability	Fixed functionality	Can be fine-tuned for specific environments and languages

3 Technical Discussion

To design a reliable and efficient assistive navigation system for our users, we explored and iteratively refined several AI based pipeline architectures. This section outlines the evolution of our system's technical design from a unified, multi task model to a modular, scene aware architecture highlighting the motivations behind each shift and the performance trade offs observed.

The original pipeline took the following shape:

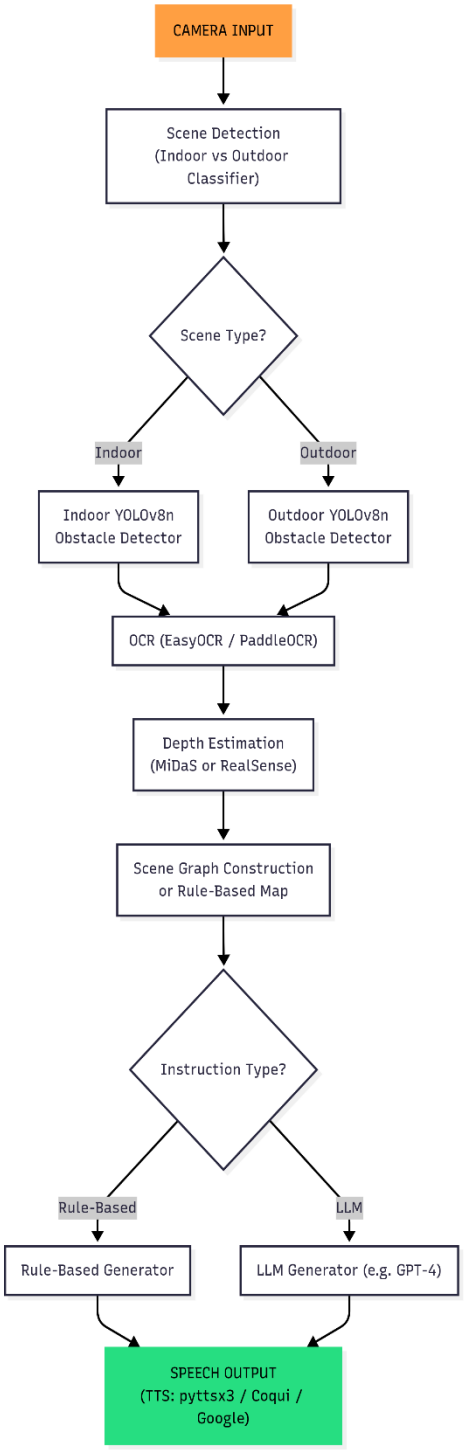


We initially began development with a unified pipeline that relied on a single object detection model trained to identify both indoor and outdoor obstacles. However, this approach quickly revealed several critical limitations:

1. **Dataset limitations:** It was difficult to find a single dataset in YOLO format that contained both relevant and helpful indoor and outdoor objects specifically suited for assisting visually impaired individuals in real-life scenarios.
2. **Dataset merging challenges:** To address this, we attempted to combine multiple datasets. However, differences in annotation formats, inconsistencies in object class definitions, and complications in converting everything into a unified YOLO format introduced significant challenges. These issues led to either poor performance or complete failure during training.
3. **Performance degradation:** Even after overcoming some of the merging and format issues, the resulting model performed poorly. The presence of irrelevant object classes, severe class imbalance between indoor and outdoor scenes, and imbalance within object categories negatively impacted detection accuracy and overall model reliability.
4. **Domain inconsistency:** A further issue was the stark variation in image characteristics across datasets. Images differed in resolution, lighting, angle, and context—often because they were captured for different purposes (e.g., surveillance, driving datasets, or product detection), which didn't translate well to our use case of wearable, human-perspective navigation assistance.

To overcome these limitations, we shifted to a modular, scene aware pipeline. By first classifying the environment as indoor or outdoor, we were able to route input to specialized object detectors optimized for each scene type. This approach improved accuracy, reduced class imbalance, and made better use of available datasets.

The updated pipeline is shown below:



The second proposed pipeline successfully avoided many of the critical issues encountered in the original design. However, new challenges still emerged during implementation, including:

1. **Lack of a suitable dataset** for indoor vs. outdoor scene classification. Most existing datasets were either too small, imbalanced, or not intended for this specific task. (Details on how we addressed this are provided in the *Model Paradigm* section.)

2. **Pipeline instability** due to limited computational resources. Our system faced frequent crashes caused by high memory consumption and the complexity of components like LLMs (used initially for instruction generation).

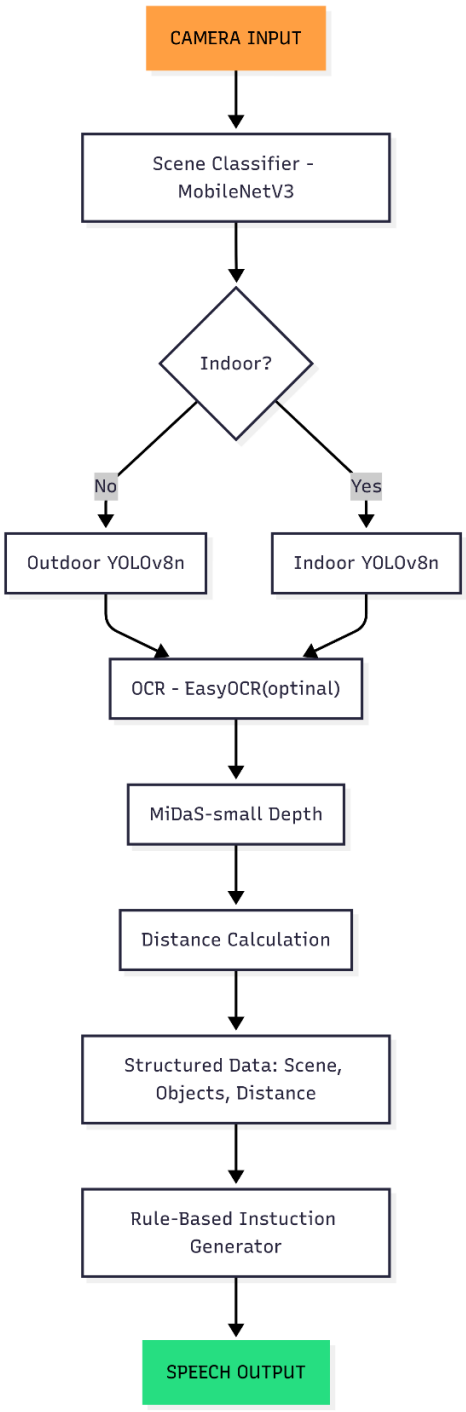
Despite these challenges, we were able to overcome them through extensive experimentation and iterative improvements. The key solutions were:

1. **Scene classifier improvements:** After testing over eight different datasets, we built a reliable scene classification model with minimal error. This was achieved using a custom-labeled version of ADE20K, which contains over 150 object classes. Though it wasn't explicitly designed for indoor/outdoor classification, we leveraged its object annotations to derive accurate labels. Full details are discussed in the *Model Paradigm* section.

2. **Reducing computational complexity:** To lower the pipeline's resource demands, we removed the NLP model used for generating instructions. LLMs were ultimately unnecessary for our use case, where simple, real-time navigation cues were sufficient. We replaced them with a **rule-based instruction generator**, which was significantly lighter and easier to run on edge devices. Additionally, the **OCR component was dropped** from the final pipeline—while useful, it was not essential for core navigation functionality and is now included as an optional feature that can be easily added to the app.

Building on the improvements of the second pipeline, the final design further streamlined the system by replacing the NLP based instruction generator with a lightweight, rule-based module and making OCR an optional feature, as well as incorporating some minor changes to enhance overall stability and performance. This resulted in a more efficient, robust, and practical assistive navigation solution.

The final pipeline is shown as follows:



The final pipeline effectively capitalized on the strengths of the second design while addressing its limitations through targeted refinements. The key changes included:

1. **OCR Repositioned as Optional**: The OCR module was made optional rather than core, as it provided supplementary benefits (e.g., reading signs or text) but was not essential to the primary functionality of navigation and obstacle avoidance. Removing it from the main flow reduced processing load and improved responsiveness.
2. **Exclusion of NLP-based Instruction Generation**: The NLP (LLM-based) instruction generator was removed entirely due to its high computational demands, frequent crashes on low-resource devices, and impracticality for generating simple navigation cues. It was replaced with a lightweight, rule-based instruction engine, which significantly improved execution speed, reduced memory consumption, and ensured reliable performance on edge and mobile hardware.

These changes, along with smaller stability and usability enhancements, made the final pipeline more efficient, robust, and well-suited for real-time deployment in assistive scenarios.

Sample result:

We are outdoors. There are 18 persons, 11 cars, 1 bike, 1 traffic light, and 2 riders. The closest is bike at about 0.7 meters very close, located to your right. You may continue straight ahead, but be mindful of objects in your path.(the photo might be unclear)

3.1 Model Paradigm

Our project addresses a complex **computer vision** problem designed to support **real-time navigation assistance for visually impaired users**. To solve this, we implemented a modular system composed of several distinct model paradigms, each targeting a specific sub-task within the pipeline. Below, we describe the paradigms applied and how they map to the problems we encountered:

1- Scene Classification

- **Paradigm:** Binary Classification (Indoor vs Outdoor)
- **Model:** MobileNetV3Large (TensorFlow, Transfer Learning)
- **Description:**
  This module is responsible for classifying each incoming video frame as either *indoor* or *outdoor*. The classification result determines which specialized object detection model to use for that frame. Although the task is binary, the model was trained using a customized version of the ADE20K dataset, which originally includes over 150 object classes.

  To tailor it for our needs, we manually re-labeled each image based on the object masks it contained. For example:

  - Images featuring *sky*, *mountain*, *tree*, *sea*, *road*, etc., were labeled as **outdoor**.
  - Images with *wall*, *ceiling*, *floor*, *chair*, *table*, etc., were labeled as **indoor**.

  Surprisingly, the resulting dataset was well-balanced, with approximately **10,000 indoor** and **9,700+ outdoor** images. This near equal distribution helped prevent model bias and contributed to stable and accurate performance during both training and evaluation.

- **Why it matters:**
  This classification step significantly improves downstream object detection accuracy by routing frames to models that are specifically optimized for either indoor or outdoor scenes. It also addresses earlier challenges with domain mismatch and class imbalance found in the unified detection approach.
- For more details about this model and the experiments done to reach it we highly recommend going through report_scene_classification.pdf

2-Indoor Object Detection:

- **Paradigm:** Object Detection (Supervised Computer Vision)
- **Model:** YOLOv8n (Fine-tuned on a merged indoor dataset)
- **Description:**
  The indoor object detection module is a key part of the SafeWalk AI system, activated

after the scene is classified as *indoor*. Its role is to detect and localize relevant obstacles within indoor environments such as *chairs*, *tables*, *beds*, *mirrors*, *toilets*, *people*, and other household or facility related objects. These detections are later used in conjunction with depth estimation to generate navigation instructions.

We adopted the **YOLOv8n architecture** due to its optimal trade off between detection accuracy, speed, and model size crucial for real time inference on mobile devices. The model was trained on a **custom-merged dataset** that combined three high quality sources:

- **HomeObjects-3K**
- **Roboflow Indoor-Spoor**
- **Roboflow Washroom-RF1FA**

All selected datasets were already formatted for **YOLOv8**, allowing seamless integration. We unified these sources into a single dataset with **19 indoor-specific object classes**, cleaned for consistency by removing duplicate or irrelevant labels. This merged dataset ensured broader object coverage, improved generalization, and reduced class imbalance.

To prepare the model for deployment, training involved various optimizations including:

- Class mapping and balancing
- Data augmentation (e.g., flipping, color jittering, scaling)
- Layer freezing to preserve general object knowledge from COCO
- Early stopping and mixed-precision training for efficiency

One major challenge was **catastrophic forgetting**, which occurred when fine-tuning on smaller datasets led to the model forgetting previously learned classes. This was resolved by retraining on the fully merged dataset from the beginning, ensuring consistent class exposure throughout the training process.

- **Why it matters:**
  Indoor environments are typically cluttered, variable in lighting, and densely packed with obstacles. These conditions require a highly specialized object detector that can recognize a wide range of object types in close proximity. The indoor model ensures reliable detection of such objects, enhancing spatial awareness and enabling the system to issue more accurate and context-aware navigation instructions. This is especially critical in homes, hospitals, offices, and other enclosed spaces where safe path planning is essential for visually impaired users.
- For more details about this model and the experiments done to reach it we highly recommend going through Indoor_object_detection_report.pdf

3-Outdoor Object Detection:

- **Paradigm:** Object Detection (Supervised Computer Vision)

- **Model:** YOLOv8n (Fine-tuned on BDD100K and custom outdoor data)
- **Description:**
  The outdoor object detection module is activated when the scene classifier identifies an outdoor environment. Its purpose is to detect critical obstacles and elements found in urban or street level environments, such as *vehicles*, *traffic lights*, *traffic signs*, *pedestrians*, *bicycles*, and *riders*. These detections are essential for helping visually impaired users navigate outdoor spaces safely and avoid hazards in real time.

For this task, we fine tuned the **YOLOv8n** architecture on the **BDD100K dataset**, a well established dataset commonly used for autonomous driving research. It includes high resolution images labeled with 10 key object classes relevant to outdoor navigation scenarios. The model was further refined using a small set of custom collected outdoor images to increase generalizability and adaptability to real-world conditions outside the dataset's original domain.

The training process followed standard object detection procedures using the Ultralytics YOLO framework. Performance was evaluated through both quantitative metrics (e.g., mAP, precision, recall) and qualitative results (visualizations of prediction accuracy on test images). Example predictions demonstrate consistent detection of multiple overlapping objects with reasonable inference times, even under variable lighting and crowd density.

- **Why it matters:**
  Outdoor navigation introduces challenges such as variable lighting, moving obstacles, and more complex scene composition. By fine-tuning on a domain-specific dataset like BDD100K, this model becomes better equipped to handle real-world street scenes. It enables the SafeWalk system to identify and localize outdoor obstacles with high confidence, supporting the generation of timely and context-aware navigation instructions for the user.
- For more details about this model and the experiments done to reach it we highly recommend going through outdoor_object_detection_report.pdf

4- Optical Character Recognition (Optional Module)

- **Paradigm:** OCR (Computer Vision — Pretrained Text Recognition)
- **Model:** EasyOCR (Pretrained PyTorch-based model)
- **Description:**
  EasyOCR was integrated as an optional module in the SafeWalk AI pipeline to recognize and extract textual information from signs, posters, doors, or other real world objects. It supports multi language recognition and performs well in diverse lighting conditions without requiring extensive retraining. Since it is based on a pretrained deep learning model, it can directly output text given an image region.

While the OCR functionality is not essential to the core navigation task, it was included to enhance the system's situational awareness in text-rich environment such as identifying room numbers, restroom labels, or directional signs.

- **Why it matters:**
  Although not part of the core pipeline, OCR provides valuable contextual information that can improve the user's understanding of their surroundings. Keeping it as an optional module allows the app to remain lightweight and responsive while offering additional accessibility features when needed.
- Note: We used a pretrained model no training nor experimentation was undergone.

5- Depth Estimation

- **Paradigm:** Dense Regression (Pixel-wise Depth Prediction)
- **Model:** MiDaS v2.1 Small (midas_v21_small_256.pt — PyTorch)
- **Description:**
  Depth estimation is a critical component of the SafeWalk AI system, enabling the perception of spatial distance from a monocular image. We used **MiDaS v2.1 Small**, a lightweight version of Intel's monocular depth estimation model. It predicts relative depth values for each pixel in a frame, allowing us to determine how far detected objects are from the user.

This model was chosen specifically for its **efficiency on CPU devices**, making it ideal for mobile deployment. It performs robustly across both indoor and outdoor environments and was integrated to work in tandem with object detection output for accurate distance-aware navigation.

To improve stability, only the central region of each object's bounding box is used for depth estimation, helping reduce the effect of noisy edges or overlapping objects. The resulting depth map is normalized and inverted to approximate real world distances.

- **Why it matters:**
  Depth estimation provides the crucial "third dimension" in navigation, allowing the system not just to detect obstacles, but to prioritize them based on proximity. This enables SafeWalk to generate meaningful guidance like *"a person is 1.5 meters ahead"* or *"a chair is close to your left"*, supporting smarter path planning for the visually impaired.
- Note: We used a pretrained model no training but some experimentations were done vs other existing models for more details and comparisons we highly recommend going through midas_selection_criteria.pdf

3.2 Data

Our project utilizes multiple datasets, each tailored to a specific subtask in our SafeWalk AI pipeline: scene classification, indoor object detection, and outdoor object detection. The diversity and complexity of the data play a significant role in the system's overall accuracy, reliability, and generalizability. This section outlines the types of data used, their sources, the preprocessing steps taken, challenges encountered, and how they were addressed.

1. Scene Classification Dataset

- **Data Type:** Image classification
- **Source:** ADE20K Dataset (re labeled)
- **Size:** ~19,700 images (10,000 indoor, 9,700+ outdoor)
- **Description:**
  To train our binary indoor vs. outdoor scene classifier, we used a modified version of the ADE20K dataset. Originally designed for semantic segmentation, it includes over 150 object categories across a wide range of environments. We manually re labeled each image based on the presence of certain object masks:
  - *Indoor labels* were assigned to images containing masks like *wall*, *ceiling*, *lamp*, *table*, *sofa*.
  - *Outdoor labels* were assigned to scenes containing *sky*, *mountain*, *tree*, *car*, *road*, etc.
- **Cleaning & Preprocessing:**
  - Converted segmentation format to classification labels
  - Normalized and resized all images to 224x224
  - Split into training (80%) and validation (20%) sets
  - Balanced the dataset to ensure equal class distribution
- **Challenges:**
  - **Label noise** due to mixed-object images (e.g., balconies)
  - Solved through manual visual inspection and thresholding object mask proportions
  - **Model impact:** Label ambiguity can slightly reduce classifier confidence, but performance remained stable (>80% accuracy)

2. Indoor Object Detection Dataset

- **Data Type:** Object detection (bounding boxes in YOLOv8 format)
- **Sources Merged:**
  - HomeObjects-3K (Ultralytics)
  - Roboflow Indoor-Spoor
  - Roboflow Washroom-RF1FA

- **Size:** ~4,200 images (combined)
- **Classes:** 19 unified indoor object classes
- **Description:**
  These datasets were selected for their focus on common indoor objects critical to visually impaired users (e.g., *chair*, *table*, *sink*, *mirror*, *TV*, *bed*). All datasets were already in YOLOv8 format, simplifying integration.
- **Preprocessing & Cleaning:**
  - Unified class labels (e.g., merged "TV screen" and "monitor")
  - Removed irrelevant or duplicate classes (e.g., *geyser*, placeholder labels)
  - Verified annotation coverage and corrected malformed labels
  - Augmented training data using flips, rotations, and brightness shifts
- **Data Split:**
  - 80% training, 20% validation
  - Maintained class distribution balance during split
- **Challenges:**
  - **Class imbalance:** Some objects (e.g., *sink*, *toilet*) appeared far less frequently → Solution: Marked underrepresented classes for future dataset expansion
  - **Catastrophic forgetting:** When fine tuning separately, some classes were lost → Solved by training on the fully merged dataset from the beginning

3. Outdoor Object Detection Dataset

- **Data Type:** Object detection (YOLOv8 format)
- **Dataset Used:** solesensei_bdd100k (Kaggle version of BDD100K)
- **Size:** ~70,000 images total (subset used for training)
- **Classes Used:** 10 (e.g., *person*, *car*, *bike*, *traffic light*, *traffic sign*, *rider*)
- **Description:**
  We used the **solesensei version of BDD100K**, which is a Kaggle hosted version of the original BDD100K dataset preprocessed into **YOLOv8 compatible format**. This saved significant time on label conversion and allowed us to focus on training and experimentation.

The dataset includes high-resolution street-level images with bounding box annotations for common urban objects, making it ideal for outdoor navigation support. We fine tuned the YOLOv8n model on a **filtered subset** of images containing the most relevant classes for assisting visually impaired users in street-level scenarios.

- **Preprocessing:**
  - No format conversion needed (already YOLOv8-ready)

- Filtered out irrelevant or rare classes
- Standardized image input size to 640x640
- Validated annotation consistency

- **Challenges:**
  - **High variation in lighting (day/night)**: Addressed with image augmentation
  - **Complex scenes with occlusion**: Managed through longer training and increased exposure to crowded frames
  - **Inference lag on low-end devices**: Justified the selection of the **YOLOv8n** (nano) variant
- **Impact:**
  Using this preprocessed dataset allowed for a streamlined training workflow and enabled high quality outdoor object detection with strong generalization to real world scenes.

4. Depth Estimation Input Data (Note: input not training data)

- **Data Type:** Raw RGB frames
- **Source:** Frames from test images and live camera feeds
- **Description:**
  The MiDaS model uses raw RGB input without annotations. No external dataset was needed. We tested the model on diverse scenes to assess visual depth consistency and object-relative distance accuracy.
- **Preprocessing:**
  - Resized to 256x256
  - Normalized pixel values to [0, 1]
  - Converted BGR (OpenCV default) to RGB

3.3 Evaluation

Robust evaluation is essential to ensure that each component of our SafeWalk AI system performs effectively under real world conditions. Because our pipeline includes **multiple models** across different paradigms classification, object detection, and depth estimation each model is evaluated using **metrics appropriate to its task**.

We selected evaluation criteria that reflect both **technical accuracy** and **practical usefulness**, given the system's goal of assisting visually impaired users with real-time navigation. Below, we describe the evaluation strategy for each model, why it was chosen, and any potential limitations.

1. Scene Classification (Indoor vs. Outdoor)

- **Evaluation Metrics:**
  - **Accuracy**
  - **Precision, Recall, F1 Score** (for class-level insights)
  - **Confusion Matrix**
- **Why these metrics?**
  Since this is a binary classification task, overall accuracy gives a high-level view, while precision and recall provide insights into model performance on each class (e.g., avoiding misclassification of outdoor scenes as indoor, which would route the frame to the wrong detector).
- **Limitations:**
  Slight label ambiguity due to mixed-scene images (e.g., indoor spaces with windows facing outdoors) may reduce precision.
  → **Mitigation:** Manual label review and model tuning with dropout and regularization.

2. Object Detection (YOLOv8n / Indoor and Outdoor Models)

- **Evaluation Metrics:**
  - **mAP@0.5 (Mean Average Precision at IoU 0.5)**
  - **mAP@0.5:0.95** (Average Precision across multiple thresholds)
  - **Precision & Recall**
  - **Visual Inspection of Predictions** (qualitative)
- **Why these metrics?**
  mAP is the industry standard for object detection and reflects both **localization** and **classification accuracy**. The split at IoU thresholds (0.5 to 0.95) gives a more comprehensive view of how well the model handles different levels of overlap, especially in cluttered scenes.

- **Limitations:**
  mAP does not directly reflect the **user impact**—for example, mislabeling a minor object may not affect navigation, while missing a *person* may be critical.
  → **Mitigation:** We prioritize qualitative testing in real-world-like environments and implement **class weighting** for high-priority objects (e.g., *person*, *pole*, *chair*).

3. Depth Estimation (MiDaS v2.1 Small)

- **Evaluation Metrics:**
  - **Qualitative Visual Validation**
  - **Relative Depth Consistency** (manual test cases)
  - **Percentile-based Region Depth Sampling** (in bounding box center)
- **Why these metrics?**
  Since the MiDaS model predicts **relative depth** (not absolute in meters), standard numeric metrics like RMSE are less useful unless ground-truth depth maps are available (which we don't have). Instead, we focus on **how consistently it ranks objects** by depth and how well the estimates align with visual expectations.
- **Limitations:**
  - Depth maps can be sensitive to texture and lighting.
  - No ground-truth absolute depth for quantitative comparison.
    → **Mitigation:** We use percentile sampling inside bounding boxes to reduce noise and compare multiple frames to check consistency.

3.4 Baseline Model

The initial prototype served as the first practical implementation of the SafeWalk AI assistive system. It followed the structure of the **first proposed pipeline**, aiming to combine object detection, text recognition, depth estimation,natural language description, and audio feedback in real time. While conceptually sound, this approach encountered significant computational and architectural limitations, prompting a shift toward a more efficient and specialized pipeline design.

Prototype Pipeline Structure

The initial design was intended to include the following modules:

Camera Input → Object Detection → OCR → Depth Estimation → NLP-Based Instruction Generator → TTS Output

However, during testing, the system experienced frequent instability and performance bottlenecks on CPU-based hardware. As a result, the **depth estimation module was never integrated** into the working prototype. Because the pipeline crashed without it being there why

would it not crash when it is so we decided to move on and search for a more practical and advanced approach leading to the second pipeline where we had out break through.

All models used in this version were **pretrained**, as the goal of this phase was to validate the overall concept, not to achieve optimal performance or accuracy. Fine tuning or custom model training had where still in development and no usable models where produced we were still starting and the planning phase.

Component Overview

- **Object Detection:** Used a **pretrained YOLOv8n model** (trained on the COCO dataset) to identify basic objects in the environment.
- **Optical Character Recognition (OCR): Tesseract OCR** was used to extract visible text from the scene (e.g., signs or labels).
- **Instruction Generation (NLP):** A **pretrained T5-small model** was used to generate navigational descriptions from detected object labels and OCR text. Despite its capability, this model introduced unnecessary computational complexity.
- **Depth estimation(Midas):** The inclusion of **MiDaS** was originally planned to provide **relative depth maps** for spatial awareness we were still in the experimentation phase and still did not now of any light versions until later on.
- **Text-to-Speech (TTS):** The **Google Text-to-Speech (gTTS)** library provided real-time speech output based on the NLP-generated descriptions.

Key Limitations

Area	Limitation
Scene Awareness	No indoor vs. outdoor classification — same model used universally
Instruction Logic	NLP model was overkill for simple scene description
Hardware Constraints	Frequent crashes, frame delays, and lag on standard hardware
No Customization	All models were off-the-shelf and not fine-tuned for our task/domain

Outcomes & Insights

Despite its limitations, the prototype:

- Demonstrated basic integration of CV, NLP, OCR, and audio components
- Validated the concept of a wearable, camera-based assistant for visually impaired users
- Provided key insights into system bottlenecks and the need for computational efficiency

This baseline model ultimately served its purpose — not as a deployable solution, but as an experimental platform. The limitations identified here directly influenced the design of the second and final pipelines, where specialization, optimization, and model fine-tuning became central to achieving real-world performance and reliability.

3.5 Advanced Model

Building upon the initial prototype, the advanced model phase focused on developing a more efficient, robust, and context-aware assistive navigation system tailored for visually impaired users. This involved replacing generic pretrained models with fine-tuned, specialized components and carefully evaluating state-of-the-art (SOTA) architectures to maximize real-time performance and accuracy within resource constraints.

Planned Improvements and Model Specialization

- **Scene-specific Object Detection:**
  Instead of relying on a single general-purpose object detector, we implemented separate YOLOv8 models fine-tuned for indoor and outdoor environments. This specialization was expected to improve detection accuracy by reducing domain mismatch and enabling each model to focus on relevant object classes and contextual features.
- **Scene Classification Module:**
  A lightweight scene classifier was introduced to automatically route frames to the appropriate object detection model based on the environment (indoor vs. outdoor). This approach improved overall system efficiency and relevance of detected objects.
- **Rule-based Instruction Generator:**
  The computationally expensive NLP-based instruction generation was replaced with a rule-based system that utilizes detected object labels and depth information. This change significantly simplified the processing pipeline, reduced latency, and produced clear, interpretable navigation instructions.
- **Depth Estimation Integration:**
  Incorporating the MiDaS_small_v21_256 depth estimation model enabled relative spatial awareness, essential for understanding obstacle proximity and generating accurate guidance.

Evaluation of SOTA Models

Throughout this phase, we evaluated several SOTA models across different system components:

- **Object Detection:**
  YOLOv8 variants (from nano to small) were fine-tuned on curated datasets. Fine-tuning yielded measurable improvements in accuracy and precision compared to off-the-shelf pretrained models, justifying the additional effort.
- **Depth Estimation:**
  Various depth estimation models were benchmarked. MiDaS_small_v21_256 emerged as the optimal choice, offering a favorable trade-off between depth quality and

computational efficiency. Larger models provided marginal accuracy gains but imposed impractical latency on the target hardware.

- **Instruction Generation (NLP):**
  Transformer-based NLP models such as T5-small were initially considered for sophisticated natural language descriptions. However, their high computational demands caused unacceptable latency and instability in real-time use. This reinforced the decision to adopt a lightweight, rule-based instruction generator that met functional needs without compromising responsiveness.

Outcomes and Learnings

- **Performance Gains:**
  Specialized object detection models improved obstacle recognition by tailoring to indoor and outdoor environments, reducing false positives and missed detections.
- **Efficiency Improvements:**
  The removal of the NLP module and integration of a rule-based generator drastically reduced computational load and inference time, enabling smoother real-time operation.
- **Depth Awareness:**
  Depth integration enhanced spatial understanding, crucial for safe navigation, while maintaining system responsiveness.
- **Pragmatic Trade-offs:**
  Although SOTA models offer impressive capabilities, their practical benefits depend heavily on the application context. For an assistive navigation system running on limited hardware, prioritizing lightweight, specialized models and simpler logic provided a more reliable user experience.

Conclusion

The advanced model approach successfully validated the project's feasibility with a practical, deployable pipeline. Selective use of SOTA models — particularly for object detection and depth estimation — yielded clear improvements without overwhelming system resources. Conversely, heavier NLP models were found unnecessary given the task's simplicity and real-time requirements. This balance between sophistication and efficiency underpins the final design's success, providing a solid foundation for future enhancements such as multi-modal data fusion and optimized edge deployment.

4 User Interface

Given the target users are visually impaired, a traditional visual UI is neither practical nor accessible. Instead, the SafeWalk app is designed as a voice assisted, blind friendly interface that minimizes user effort and maximizes immediate assistance.

The ideal user experience allows the app to be activated via a simple voice command (e.g., “SafeWalk, guide me”), instantly launching the camera and starting the navigation pipeline without any manual interaction. This voice first, hands free approach reduces barriers and ensures seamless real time guidance.

Additional enhancements to improve usability include:

- **Physical Shortcut or Wearable Integration:** Offering an optional physical button or wearable remote to start the app quickly when voice activation is unavailable or inconvenient.
- **Haptic Feedback:** Integrating subtle vibration alerts to complement voice instructions, especially helpful in noisy environments.

The best approach would be integrating SafeWalk directly into the system firmware or core OS for several reasons:

- **Seamless user experience:**
  The assistive feature would be instantly available on device startup without requiring any downloads or manual activation.
- **Deep hardware-software optimization:**
  Direct integration allows for better resource management, lower latency, and tighter coupling with device sensors (camera, depth sensors) and system services.
- **Accessibility consistency:**
  It can be made part of the core accessibility suite, ensuring it works reliably across all apps and system states (lock screen, emergency mode, etc.).
- **System-wide permissions:**
  The app wouldn’t face the usual app-store restrictions, permission prompts, or background limitations.
- **Better support & updates:**
  Updates can be rolled out with system updates, improving stability and integration over time.

Overall, SafeWalk focuses on simplicity, immediacy, and accessibility by leveraging existing mobile OS voice assistants while providing specialized, real time guidance tailored for visually impaired users.

5 Challenges

Throughout the SafeWalk AI project, several key challenges arose that significantly influenced the development process and final design choices. Addressing these obstacles provided valuable lessons and helped shape a more robust and practical solution.

1. Computational Resource Constraints
The initial prototype combined multiple complex models, including object detection, OCR, NLP-based instruction generation, and depth estimation. Running all these modules in real time on standard CPU-based hardware resulted in frequent crashes, significant lag, and overall instability. This challenge was identified early during prototype testing, where the system was unable to maintain smooth performance.

Solution & Lesson:
To overcome this, the pipeline was streamlined by removing the computationally expensive NLP model and making the depth estimation module optional. Additionally, models were specialized and fine-tuned for specific environments (indoor vs. outdoor) to reduce unnecessary processing. This taught the importance of balancing model complexity with hardware limitations, especially in assistive applications requiring real-time feedback.

2. Dataset Acquisition and Preparation
Obtaining and merging diverse datasets suitable for both indoor and outdoor object detection presented a significant obstacle. Datasets varied in format, class definitions, image perspectives, and annotation quality. Some datasets, while large, contained many irrelevant classes or perspectives that did not align with the use case.

Solution & Lesson:
Custom scripts were developed to unify dataset formats and class mappings. Careful selection and merging of relevant datasets ensured coverage of critical obstacle classes. This challenge highlighted the importance of domain-specific data curation and preprocessing in achieving effective model training.

3. Model Generalization vs. Specialization
Using generic pretrained models initially led to suboptimal detection accuracy due to domain mismatch—outdoor objects being misclassified indoors and vice versa. Balancing model

generalization with the need for specialized detection in different environments was a key design challenge.

Solution & Lesson:
Introducing a lightweight scene classifier to route frames to environment-specific object detection models improved accuracy and reduced false positives. This reinforced that adaptive model selection based on context can greatly enhance performance in real-world applications.

4. Instruction Generation Complexity
The original approach employed a transformer-based NLP model to generate natural language navigation instructions. While powerful, this introduced excessive computational overhead and latency for a task that largely required simple, direct commands.

Solution & Lesson:
Replacing the NLP model with a rule-based instruction generator drastically improved system responsiveness and reduced resource use. This shift underscored the value of simplicity and task-appropriate complexity in assistive technology design.

5. Segmentation vs. Bounding Box for Depth Estimation

One notable challenge encountered during the project was deciding whether to retrain our object detection models to perform segmentation instead of simple bounding box detection. While bounding boxes are easier to obtain and faster to process, they provide only coarse spatial information that limits the precision of depth fusion when combined with MiDaS depth maps. Pixel-accurate segmentation masks would allow much more precise spatial awareness and improve obstacle localization, which is critical for assistive navigation.

However, retraining a segmentation-capable YOLOv8 model required significant additional effort, including dataset preparation, format conversion of segmentation masks, and extended training time. Given these constraints, we initially opted to use a script to estimate obstacle distances by fusing bounding box detections with depth estimations from MiDaS. This approach simplified implementation and allowed us to proceed with system integration and testing.

Note: We explored the option of training YOLO segmentation models to improve depth estimation accuracy by using precise object masks instead of bounding boxes. However, YOLO requires segmentation annotations in specific formats either COCO JSON or YOLO segmentation format (.txt). Most of the available datasets had segmentation masks in PNG format, which necessitated conversion. While tools like pycocotools or labelme2coco could facilitate this conversion, we faced a steep learning curve and limited time to fully explore and integrate these tools. Consequently, this constrained our ability to retrain segmentation models and pushed us to adopt a workaround using bounding box based depth approximation scripts to move forward.

Summary
These major challenges collectively guided the project toward a leaner, more focused, and user-centric solution. The key lessons learned include the importance of resource-aware design, domain-specific data preparation, adaptive model deployment, simplicity in instruction logic, and accessible user experience—each vital for creating effective assistive AI systems there more minor challenges mentioned in the side models reports mentioned at the beginning of the paper.

6 Future Work

Building on the insights and outcomes of this project, several promising directions for future research and development emerge:

1. Segmentation-Based Object Detection:
To improve spatial accuracy and depth estimation, future efforts could focus on training or fine-tuning segmentation models (e.g., YOLOv8n-Seg) using properly formatted datasets. Developing or converting datasets with pixel-accurate masks would enhance obstacle delineation, enabling more precise navigation guidance.

2. Dataset Expansion and Custom Annotation:
Expanding the dataset to include a broader variety of indoor and outdoor objects relevant to visually impaired navigation—such as handrails, curbs, or smaller obstacles—would improve detection robustness. Custom annotation tools and pipelines could be developed to efficiently label new classes tailored to the assistive use case.

3. Lightweight, Real-Time Depth Estimation:
Further optimization of depth estimation models to reduce computational load while maintaining accuracy will be critical for deployment on low-power mobile devices. Research into novel architectures or model compression techniques could yield significant gains.

4. Adaptive and Personalized Instruction Generation:
Replacing rule-based navigation instructions with adaptive systems that learn from user feedback and preferences could improve usability. Incorporating reinforcement learning or personalized natural language generation models may offer more intuitive assistance.

By pursuing these avenues, the SafeWalk AI system can evolve into a robust, reliable, and user-centric assistive technology with meaningful real-world impact.

7 Conclusion

This project set out to develop an assistive navigation system, SafeWalk AI, aimed at enhancing the mobility and independence of visually impaired users through real-time obstacle detection and guidance. By exploring multiple pipelines and leveraging pretrained models, the project successfully demonstrated the feasibility of integrating computer vision, depth estimation, OCR, and audio feedback into a cohesive system.

Key outcomes include the development of specialized object detection models tailored for indoor and outdoor environments, the selection of an efficient depth estimation model (MiDaS_small_v21_256), and the design of a streamlined pipeline that balances computational demands with functional effectiveness. The replacement of a resource-intensive NLP model with a rule-based instruction generator significantly improved system responsiveness and stability, highlighting the importance of domain-specific optimization.

Challenges such as dataset format mismatches, computational limitations, and the complexities of segmentation-based depth fusion provided valuable learning experiences, informing decisions to prioritize lightweight models and practical implementation strategies. The project also underscored the need for future work in areas like dataset expansion, and user-centered interface design.

Overall, SafeWalk AI contributes a practical framework for assistive navigation that bridges advanced AI techniques with real-world constraints. Its modular design and focus on computational efficiency make it well-suited for deployment on mobile devices, offering promising potential to improve quality of life for visually impaired individuals. The insights and methodologies developed here pave the way for further research and refinement toward accessible, intelligent mobility solutions.

8 Bibliography

A. Sources Used in Final Product

[1] A. W. Safwan, “BDD100K Dataset,” *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/awsaf49/bdd100k-dataset

[2] Ultralytics, “HomeObjects-3K Dataset,” *Ultralytics Docs*, 2023. [Online]. Available: https://docs.ultralytics.com/datasets/detect/homeobjects-3k/

[3] Thesis S260n, “Indoor-Spoor Dataset,” *Roboflow Universe*, 2023. [Online]. Available: https://universe.roboflow.com/thesis-s260n/indoor-spoor

[4] Roboflow-100, “Washroom-RF1FA Dataset,” *Roboflow Universe*, 2023. [Online]. Available: https://universe.roboflow.com/roboflow-100/washroom-rf1fa

[5] Intel ISL, "MiDaS: Monocular Depth Estimation," *GitHub*, 2023. [Online]. Available: https://github.com/isl-org/MiDaS

[6] Ultralytics, "YOLOv8 Documentation," *Ultralytics Docs*, 2023. [Online]. Available: https://docs.ultralytics.com

B. Sources Used During Research & Exploration

[7] M. Moallem, "Outdoor Object Detection (WIP)," *Roboflow Universe*, 2025. [Online]. Available: https://app.roboflow.com/mohamadmoallem/outdoor_object_detection_mm-arpd7

[8] A. W. Safwan, "ADE20K Dataset," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/awsaf49/ade20k-dataset

[9] Thepbordin, "Indoor Object Detection Dataset," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/thepbordin/indoor-object-detection

[10] Ultralytics, "Open Images V7 Dataset," *Ultralytics Docs*, 2023. [Online]. Available: https://docs.ultralytics.com/datasets/detect/open-images-v7/

[11] Soumik Rakshit, "NYU Depth V2 Dataset," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/soumikrakshit/nyu-depth-v2

[12] Intelligent Digital Systems Lab, "Imperial UAV Indoor Dataset," *Imperial College London*, 2023. [Online]. Available: https://www.imperial.ac.uk/intelligent-digital-systems/indoor-uav-data/

[13] P. Renaisan, "MIT Indoor Scene Classification Dataset," *Roboflow Universe*, 2023. [Online]. Available: https://universe.roboflow.com/pasha-renaisan-gmail-com/indoor-mit-scene-classification

[14] L. Ashraf, "SUN397 (50-50 Split)," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/lash45/sun397-50-50

[15] "CIFAR-10 Dataset," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/c/cifar-10/

[16] D. Schettler, "LabelMe 12_50k Dataset," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/dschettler8845/labelme-12-50k

[17] *Caltech101 Dataset*, *Voxel51 Dataset Zoo*, 2023. [Online]. Available: https://docs.voxel51.com/dataset_zoo/datasets.html#dataset-zoo-caltech101

[18] *Caltech256 Dataset*, *Voxel51 Dataset Zoo*, 2023. [Online]. Available: https://docs.voxel51.com/dataset_zoo/datasets.html#dataset-zoo-caltech256

[19] *ImageNet Sample Dataset*, *Voxel51 Dataset Zoo*, 2023. [Online]. Available: https://docs.voxel51.com/dataset_zoo/datasets.html#imagenet-sample