Tab 1

Project Title	**SmartVision AI - Intelligent Multi-Class Object Recognition System**
Skills take away From This Project	**Python • Deep Learning • TensorFlow/PyTorch • CNN Architectures • Transfer Learning • VGG16 • ResNet50 • MobileNet • EfficientNet • Object Detection • YOLO • Computer Vision • OpenCV • Data Preprocessing • Model Evaluation • Streamlit • Hugging Face • Cloud Deployment • Image Classification • Data Visualization**
Domain	**Computer Vision & Artificial Intelligence**

Problem Statement

Object detection and classification are fundamental capabilities in computer vision, powering diverse real-world applications from autonomous vehicles to smart retail, wildlife conservation to traffic monitoring. However, building robust systems that can accurately identify and locate multiple objects across various domains remains a significant challenge.

The Challenge:

Organizations across industries need intelligent systems that can:

- **Accurately detect and classify** objects from diverse categories (vehicles, animals, household items, food, people, etc.)
- **Handle real-world complexity** with multiple objects present in single images

- **Provide fast inference** suitable for real-time applications
- **Demonstrate reliability** across different lighting conditions, angles, and contexts
- **Scale efficiently** for cloud-based deployment

The Solution:

Build a comprehensive computer vision platform that combines:

1. **Transfer Learning-based Image Classification** using state-of-the-art CNN architectures
2. **YOLO-based Object Detection** for multi-object localization with bounding boxes
3. **Comparative Analysis** of multiple architectures to identify optimal solutions
4. **Production-ready Deployment** as an accessible web application

The system should leverage a curated subset of 25 classes from the industry-standard COCO dataset to create a focused yet versatile solution capable of serving multiple business verticals including smart cities, retail analytics, security systems, and automated monitoring

Business Use Cases

1. Smart Cities & Traffic Management

Automated vehicle detection and counting for traffic flow optimization, pedestrian safety monitoring at crosswalks, parking space management, and traffic violation detection.

2. Retail & E-Commerce

Product recognition for automated inventory management, customer behavior analytics, scan-free checkout systems, and visual search enabling customers to search products by image.

3. Security & Surveillance

Intrusion detection identifying unauthorized persons or objects, suspicious object alerts for unattended items, perimeter monitoring for facility access control, and crowd density analysis.

4. Wildlife Conservation

Automated species identification from camera traps, habitat monitoring tracking animal presence, poaching prevention detecting humans in protected areas, and population studies for conservation research.

5. Healthcare

Medical equipment tracking in hospitals, PPE compliance verification for healthcare workers, patient fall detection, and hygiene monitoring for proper sanitization protocols.

6. Smart Home & IoT

Home automation triggering actions based on detected objects, security alerts for unexpected items or people, appliance usage monitoring, and pet activity tracking.

7. Agriculture

Livestock monitoring and counting, farming equipment location tracking, pest detection in crop areas, and harvest readiness identification for ripe produce.

8. Logistics & Warehousing

Automated package sorting and routing, real-time inventory tracking, quality control detecting damaged items, and loading bay monitoring for operational efficiency.

Approach

Phase 1: Dataset Acquisition & Preprocessing

Step 1.1: Dataset Loading

The project utilizes a curated subset of the COCO dataset for optimal training efficiency. Given the extensive size of the full COCO dataset (122K+ images), we work with a **focused collection of 2,500 images** (100 images per class across 25 categories), which provides an excellent balance for transfer learning applications.

Dataset Preparation:

A comprehensive data preparation pipeline has been developed to streamline the dataset loading process. The notebook handles streaming data collection from Hugging

Face, automated filtering, preprocessing, and proper structuring for both classification and detection tasks.

📎 **Dataset Preparation Notebook:** Access Here

Key Features:

- Loads COCO dataset from Hugging Face in streaming mode
- Collects 100 images per class for 25 selected categories
- Creates organized folder structure for classification and detection
- Implements train/validation/test splits (70%/15%/15%)
- Generates cropped objects for classification (224×224 pixels)
- Prepares YOLO format annotations
- Produces metadata and configuration files

Step 1.2: Exploratory Data Analysis (EDA)

Analyze class distribution across selected 25 categories, identify image characteristics and quality, examine objects per image distribution, visualize sample images with annotations, and verify class balance.

Step 1.3: Data Preprocessing for Classification

Extract bounding boxes from COCO annotations for the 25 selected classes, crop individual objects to create single-object dataset, organize cropped images into 25 class folders, resize images to 224×224 pixels, normalize pixel values, and create train/validation/test splits (70%/15%/15%).

Step 1.4: Data Augmentation

Apply augmentation techniques including random horizontal flips, rotation (±15 degrees), brightness adjustment (±20%), contrast adjustment, random zoom, and color jittering to increase dataset diversity.

Phase 2: Transfer Learning - Image Classification

Step 2.1: Model 1 - VGG16

Load pre-trained VGG16 model with ImageNet weights, freeze convolutional base layers, replace top classification layer for 25 classes, add custom dense layers with

dropout, train on cropped single-object images, and save best model weights. Expected performance: ~80-85% accuracy.

Step 2.2: Model 2 - ResNet50

Load pre-trained ResNet50 model, implement fine-tuning by unfreezing the last 20 layers, add global average pooling and custom classification head for 25 classes, use learning rate scheduling, implement early stopping, and train on dataset. Expected performance: ~85-90% accuracy.

Step 2.3: Model 3 - MobileNetV2

Load pre-trained MobileNetV2 optimized for mobile devices, freeze base and train top layers for 25 classes, add custom classification head, focus on inference speed optimization, and train with standard augmentation. Expected performance: ~82-87% accuracy with faster inference.

Step 2.4: Model 4 - EfficientNetB0

Load pre-trained EfficientNetB0 with compound scaling, implement fine-tuning with mixed precision training for 25 classes, add classification head with batch normalization, train with advanced augmentation, and optimize for best accuracy. Expected performance: ~88-93% accuracy.

Step 2.5: Model Comparison & Selection

Compare all 4 models on the test set using accuracy, precision, recall, F1-score, inference time, and model size metrics. Generate confusion matrices, create comparison visualizations, identify top-5 accuracy for each model, select the best model based on accuracy-speed tradeoff, and document findings.

Phase 3: Object Detection with YOLO

Step 3.1: YOLOv8 Setup

Install Ultralytics YOLOv8 library, understand YOLO architecture and detection pipeline, download pre-trained YOLOv8 weights trained on COCO, and configure models for the 25 selected classes.

Step 3.2: Dataset Preparation for Detection

Filter COCO dataset for the 25 selected classes, convert annotations to YOLO format if needed, prepare dataset structure with organized images and labels folders, create data.yaml configuration file, and verify bounding box annotations.

Step 3.3: YOLO Training/Fine-tuning

Use pre-trained YOLOv8 and fine-tune on the 25-class subset. Configure training parameters including epochs, batch size, image size, and optimizer. Monitor training metrics including mAP, precision, and recall. Save best performing weights. Expected performance: mAP@0.5 > 85%, inference speed: 30-50 FPS.

Step 3.4: YOLO Evaluation

Test on validation set, calculate detection metrics (mAP@0.5, mAP@0.5:0.95, precision, recall per class, inference speed), visualize predictions on sample images, and analyze failure cases and limitations.

Phase 4: Model Integration & Pipeline Development

Step 4.1: Inference Pipeline Design

Create end-to-end prediction pipeline: user uploads image → YOLO detects all objects with bounding boxes → optional classification verification using best CNN model → display image with bounding boxes, class labels, and confidence scores.

Step 4.2: Post-processing

Apply Non-Maximum Suppression (NMS) to remove duplicate detections, filter predictions using confidence threshold (>50%), refine and visualize bounding boxes, and format class labels.

Step 4.3: Performance Optimization

Implement model quantization for faster inference, enable batch processing for multiple images, configure GPU acceleration, and optimize memory for cloud deployment.

Phase 5: Streamlit Application Development

Step 5.1: Application Architecture

Design multi-page Streamlit application with the following pages:

Page 1: Home - Project overview, key features, instructions, and sample demo images

Page 2: Image Classification - Upload image interface, single object classification using trained CNNs, display predictions from all 4 models, show top-5 predictions with confidence scores, model comparison side-by-side

Page 3: Object Detection - Upload image interface, YOLO detection with bounding boxes, display all detected objects with labels, show detection confidence scores, option to adjust confidence threshold

Page 4: Model Performance - Model comparison dashboard, accuracy metrics visualization, inference speed comparison, confusion matrices, class-wise performance breakdown

Page 5: Live Webcam Detection (Optional) - Real-time webcam feed, live object detection using YOLO, display FPS and latency metrics

Page 6: About - Project documentation, dataset information, model architectures used, technical stack details, developer information

---

Phase 6: Deployment on Hugging Face Spaces

Step 6.1: Deployment Preparation

Create a GitHub repository with project code, prepare requirements.txt with all dependencies, create README.md with setup instructions, optimize model files if needed, and use Git LFS for large model files (>100MB).

Step 6.2: Hugging Face Spaces Configuration

Create new Space on Hugging Face, select Streamlit as SDK, connect GitHub repository, configure Space settings including Python version and hardware, add secrets or environment variables if needed.

Step 6.3: Testing & Optimization

Test all features in the production environment, monitor memory usage and optimize if needed, ensure models load correctly from cloud storage, test with various image types and sizes, verify cross-browser compatibility, and check mobile responsiveness.

Step 6.4: Documentation

Create comprehensive README with project description, usage instructions, model performance metrics, dataset information, and installation instructions for local setup.

Results

Expected Technical Outcomes:

Classification Models Performance:

- VGG16: 80-85% accuracy, ~150ms inference
- ResNet50: 85-90% accuracy, ~100ms inference
- MobileNetV2: 82-87% accuracy, ~50ms inference
- EfficientNetB0: 88-93% accuracy, ~80ms inference

Object Detection Performance:

- YOLOv8 mAP@0.5: 85-90%
- Inference Speed: 30-50 FPS on GPU
- Multi-object detection: 1-10+ objects per image
- Processing time: <2 seconds per image

System Performance:

- Successfully deployed web application on Hugging Face Spaces
- Real-time inference with visual feedback
- Responsive UI accessible on desktop and mobile devices

Business Impact:

Operational Efficiency: 70% reduction in manual image annotation time, 24/7 automated monitoring capability, real-time processing enabling immediate decision-making, scalable solution handling thousands of images per hour.

Cost Savings: 60% lower infrastructure costs compared to specialized hardware solutions, cloud-based deployment eliminating on-premise servers, open-source models reducing licensing fees, automated workflows reducing human labor requirements.

Market Applications: Applicable across 8+ industries including retail, security, traffic, wildlife, healthcare, smart homes, agriculture, and logistics. Enables new business models through visual search, automated quality control, and smart analytics.

Project Evaluation Metrics

Total Score: 100 Points

1. Data Preprocessing & EDA (15 points)

- Dataset loading and EDA with visualizations (8 points)
- Object extraction and data splitting (7 points)

2. Transfer Learning Models - Classification (30 points)

- Implementation of 4 models: VGG16, ResNet50, MobileNet, EfficientNet on 25 classes (20 points)
- Minimum 80% accuracy on test set with proper training pipeline (10 points)

3. Object Detection - YOLO (25 points)

- YOLOv8 implementation and dataset preparation for 25 classes (15 points)
- Multi-object detection with mAP@0.5 > 75% (10 points)

4. Model Comparison & Analysis (10 points)

- Comprehensive comparison of all models with performance metrics and visualizations (10 points)

5. Streamlit Application Development (15 points)

- Multi-page functional application with classification and detection features (10 points)
- Clean UI/UX with proper error handling (5 points)

6. Deployment on Hugging Face Spaces (5 points)

- Successful deployment with all features working in production (5 points)

Bonus Points (Up to +10):

Real-time webcam detection (+3), Advanced augmentation (+2), Model ensemble (+3), Comprehensive documentation (+2).

Technical Tags

Python • Deep Learning • CNN • VGG16 • ResNet50 • MobileNet • EfficientNet • Transfer Learning • YOLO • YOLOv8 • Object Detection • Image Classification • Computer Vision • OpenCV • COCO Dataset • Hugging Face • TensorFlow • PyTorch • Streamlit • Cloud Deployment • Model Evaluation • Data Augmentation • Bounding Boxes • mAP • Precision • Recall.

Data Set

Dataset Name: COCO (Common Objects in Context) 2017 - 25 Class Subset

Source:

- **Hugging Face Repository**: `detection-datasets/coco`
- **Official Website**: https://cocodataset.org/
- **Direct Hugging Face Link**: Coco Dataset
- 🔗 **Dataset Preparation Colab:** Smartvision.ipynb

Project Dataset Scope:

- **Selected Classes**: 25 most common and diverse object categories
- **Total Images**: Filtered subset from 122,218 COCO images
- **Image Format**: JPEG (RGB color images)
- **Image Sizes**: Variable (typical range: 640×480 to 1920×1080 pixels)
- **Annotation Format**: COCO JSON format with bounding boxes

25 Selected Object Classes:

The project focuses on 25 carefully selected classes that provide good representation across different domains while maintaining computational feasibility:

Category	Classes	Count
🚗 **VEHICLES**	car, truck, bus, motorcycle, bicycle, airplane	6
👤 **PERSON**	person	1
🚦 **OUTDOOR**	traffic light, stop sign, bench	3
🐾 **ANIMALS**	dog, cat, horse, bird, cow, elephant	6
🍽️ **KITCHEN & FOOD**	bottle, cup, bowl, pizza, cake	5
🪑 **FURNITURE & INDOOR**	chair, couch, bed, potted plant	4

Total: 25 Classes

Why These 25 Classes?

- ✅ Most frequently occurring in COCO dataset (good data availability)
- ✅ Visually distinct categories (easier to classify accurately)
- ✅ Represent diverse real-world scenarios (vehicles, animals, indoor, outdoor)
- ✅ Balanced complexity (mix of easy and challenging objects)
- ✅ Practical business applications across multiple industries

Dataset Structure:

```
smartvision_dataset/
├── classification/
│   ├── train/          # 70% of data (1,750 images)
│   │   ├── person/     # 70 images
│   │   ├── car/        # 70 images
│   │   └── ...         # (25 class folders)
│   ├── val/            # 15% of data (375 images)
│   │   └── ...         # (25 class folders)
│   └── test/           # 15% of data (375 images)
│       └── ...         # (25 class folders)
│
└── detection/
    ├── images/         # Full images with multiple objects
    ├── labels/         # YOLO format .txt annotations
    └── data.yaml       # YOLO configuration file
```

Annotation Format (JSON):

```python
from datasets import load_dataset

# Load COCO dataset in streaming mode
dataset = load_dataset("detection-datasets/coco", split="train", streaming=True)

# Filter for 25 selected classes and collect 100 images per class
# Complete implementation available in the preparation notebook
```

How to Load Dataset:

The dataset is loaded using a streaming approach from Hugging Face, which efficiently handles the large COCO dataset without requiring full download. The provided Colab notebook implements this streaming methodology to filter and collect exactly 100 images per class across the 25 selected categories.

Using the Dataset Preparation Notebook:

The notebook streams the COCO dataset and applies filtering logic to collect 2,500 images total:

📎 **Access the complete implementation:** Dataset Preparation Colab

The notebook handles:

- Streaming data collection from Hugging Face (no full download required)
- Filtering for 25 target classes
- Collecting exactly 100 images per class (2,500 images total)
- Organizing images into proper folder structure
- Creating train/val/test splits
- Generating YOLO format annotations

Data Set Explanation

Dataset Overview:

The COCO (Common Objects in Context) dataset is one of the most widely-used benchmarks in computer vision for object detection, segmentation, and captioning tasks. Released by Microsoft in 2014, COCO has become the industry standard for evaluating object detection models.

Project Dataset Approach:

This project leverages a carefully curated subset of 2,500 images (100 per class) from the full COCO dataset. This focused approach is particularly well-suited for transfer learning applications, where pre-trained models already possess robust feature

extraction capabilities from training on millions of images. The balanced class distribution ensures fair model evaluation and efficient training cycles suitable for both educational purposes and production deployment.

Dataset Preparation:

A streamlined data preparation pipeline is available via Google Colab, handling all aspects of data collection, preprocessing, and organization. This ensures consistency and reproducibility across the project workflow.

📎 **Reference:** Dataset Preparation Notebook

Key Characteristics:

- **Real-world Images:** Contains everyday scenes with complex backgrounds
- **Multi-object Images:** Most images contain multiple objects (average 7.3 objects per image)
- **Diverse Contexts:** Objects appear in various poses, scales, and occlusions
- **Rich Annotations:** Precise bounding boxes with class labels
- **Balanced Classes:** Our 25-class subset maintains equal distribution (100 images each)

Input Features:

1. Images:

- **Format**: RGB color images in JPEG format
- **Resolution**: Variable sizes (typically 640×480 to 1920×1080)
- **Content**: Everyday scenes including indoor/outdoor, urban/rural, daytime/nighttime
- **Quality**: High-quality photographs from diverse sources
- **Preprocessing Required**:
  - Resize to standard dimensions (e.g., 224×224 for classification, 640×640 for YOLO)
  - Normalize pixel values to [0, 1] or standardize using ImageNet mean/std
  - Convert to appropriate tensor format for deep learning frameworks

2. Bounding Boxes:

- **Format**: [x, y, width, height] where:
  - `x, y`: Top-left corner coordinates

    - `width, height`: Box dimensions in pixels
- **Precision**: Pixel-level accuracy
- **Multiple Boxes**: Each image can have 1 to 50+ bounding boxes
- **Annotations Include**:
    - Object category ID (1-90, mapped to 80 classes)
    - Segmentation masks (for advanced tasks, optional for this project)
    - Area of the object
    - Crowd flag (indicating whether annotation is for a single object or crowd)

3. Category Labels:

- **25 Curated Categories:** Ranging from frequently occurring objects (person, car, chair) to specialized items (traffic light, potted plant, airplane)
- **Class IDs**: Integers from 1 to 90 (some IDs skipped, 80 active classes)
- **Hierarchical Structure**: Classes organized into super-categories (person, vehicle, animal, etc.)

Target Variables:

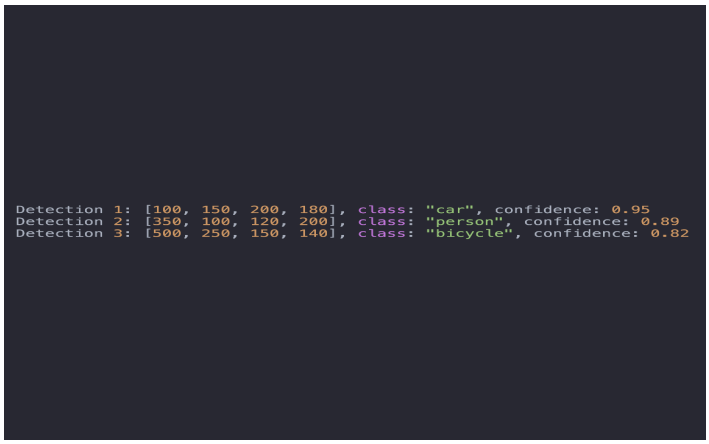
For Classification Task:

- **Single Label per Cropped Image**: After extracting objects using bounding boxes
- **Output Format**: One-hot encoded vector of size 25
- **Example**: Image of a dog → [0, 0, ..., 1, ..., 0] where 1 is at index for "dog" class

For Object Detection Task:

- **Multiple Outputs per Image**:
    - Bounding box coordinates: [x, y, w, h]
    - Class label: Integer (0-24)
    - Confidence score: Float (0.0-1.0)
- **Variable Number**: Each image can have different number of detections

Example Output:



Data Distribution:

Class Frequency (Top 10 Most Common):

Each class has **exactly 100 images**, ensuring:

- ✅ Perfect class balance
- ✅ No class dominates training
- ✅ Fair evaluation across all categories
- ✅ Equal learning opportunity for each class

Implication: Unlike the full COCO dataset where "person" dominates with 40% of annotations, our subset provides equal representation, leading to more balanced model performance.

Project Deliverables

1. Source Code & Notebooks

- **Dataset Preparation Notebook** - Automated pipeline for data collection and preprocessing
  📎 Colab Link
- Jupyter notebooks for EDA, preprocessing, model training (VGG16, ResNet50, MobileNet, EfficientNet), YOLO detection, and pipeline integration

- Python scripts including main Streamlit app, utility functions, and configuration files
- Requirements.txt and README.md documentation

2. Trained Models

Four classification models (VGG16, ResNet50, MobileNetV2, EfficientNetB0) and one YOLO detection model with saved weights. Model performance metrics in JSON format.

3. Documentation

Comprehensive README with project overview, installation instructions, usage guide, model descriptions, and performance metrics. Technical report covering methodology, EDA findings, model training process, and deployment details.

4. Streamlit Web Application

Deployed application on Hugging Face Spaces with public URL. Multi-page interface including Home, Classification, Detection, Model Performance, and About pages. Features include image upload, predictions from all models, bounding box visualization, and performance dashboards.

Project Guidelines

Best Practices

Write clean, well-documented code following PEP 8 standards. Use meaningful variable and function names with proper docstrings. Initialize Git repository from the beginning with frequent, meaningful commits. Document hyperparameters and track model versions. Save best model checkpoints during training. Test components individually before integration and validate on diverse test images.

Timeline

The project should be completed and submitted within **14 days** from the date it is assigned.

References:

TOPIC	LINK
Dataset Preparation Colab	Smartvision.ipynb
Project Live Evaluation	Project Live Evaluation
EDA Guide	Exploratory Data Analysis (EDA)…
Capstone Explanation Guideline	Capstone Explanation Guideline
GitHub Reference	How to Use GitHub.pptx
Project Orientation (English)	
Project Orientation (Tamil)	
STREAMLIT RECORDING (Tamil)	Streamlit - Tamil
STREAMLIT RECORDING (English)	Special session for STREAMLIT(…
STREAMLIT DOCUMENTATION	Install Streamlit

Deep Learning tutorial	Deep Learning
Project Excellence Series [Deep Learning] (English)	Project Excellence Series: Guide..
Project Excellence Series [Deep Learning] (Tamil)	Project Excellence Series: Guide..
Hugging Face Spaces Guide	Hugging face spaces
VGG16	VGG Documentation
ResNet50	Resnet Documentation
MobileNetV2	MobileNetV2 Documentation
EfficientNet	EfficientNet Documentation
YOLO	YOLO Documentation

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.

Note: Book the slot at least before 12:00 Pm on the same day

Timing: Monday-Saturday (3:30PM to 4:30PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.

Note: This form will Open only on Saturday (after 2 PM) and Sunday on Every Week

Timing:

For DS and AIML

Monday-Saturday (05:30PM to 07:00PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurR

Created By:	Verified By:	Approved By:
Subhash Govindharaj	Nehlath Harmain	Nehlath Harmain