SOFTWARE REQUIREMENT SPECIFICATION

1. Introduction

1.1 Purpose

The project aims to develop an AI-powered people counting system that can: - Process images and videos to detect and count people - Handle overlapped scenarios in crowded scenes - Provide real-time video processing capabilities - Support multiple deep learning models for accurate counting

1.2 Scope

The system provides: - Image and video processing capabilities - Support for multiple ML models (VGG16, ResNet50, ResNet18, YOLO) - Real-time video stream processing - Ground truth comparison - Web-based user interface - REST API endpoints

1.3 Definitions, Acronyms, and Abbreviations

- VGG16: Visual Geometry Group 16-layer CNN
- ResNet: Residual Neural Network
- YOLO: You Only Look Once (object detection model)
- CNN: Convolutional Neural Network
- API: Application Programming Interface
- SLA: Service Level Agreement

1.4 References

- PyTorch Documentation
- Flask Documentation
- React.js Documentation
- Bootstrap Documentation

2. System Overview

2.1 Overall Description

A web-based application that utilizes deep learning models to count people in images and videos, featuring both standard and overlapped counting capabilities.

2.2 Key Features

- Multiple model support (VGG16, ResNet50, ResNet18, YOLO)
- Real-time video processing
- Image processing with bounding boxes

- Overlapped people detection
- Ground truth comparison
- Progress tracking for video processing
- Base64 image encoding/decoding
- CSV-based ground truth validation

3. Operating Environment

3.1 Software Requirements

- Python 3.8+
- Flask web framework
- PyTorch ML framework
- React.js frontend
- Node.js runtime
- Modern web browser
- CUDA support (optional)

3.2 Hardware Requirements

- CPU: Multi-core processor
- RAM: 8GB minimum, 16GB recommended
- GPU: NVIDIA GPU with CUDA support (optional)
- Storage: 500MB for application, 2GB+ for models
- Network: Broadband internet connection

4. Functional Requirements

4.1 Data Collection and Storage

- Image input: JPEG, PNG formats
- Video input: MP4, AVI formats
- URL input for video streams
- CSV storage for ground truth data
- Model weights storage

4.2 Data Preprocessing

- Image resizing to (224, 224)
- Normalization using ImageNet stats
- Video frame extraction
- Base64 encoding/decoding
- Tensor transformation
- Channel manipulation for overlapped detection

4.3 Model Development

Models implemented: - VGG16 for standard counting - ResNet50 for standard counting - ResNet18 for overlapped counting - ResNet50 for overlapped counting - YOLOv5 for video processing - FasterRCNN for object detection

4.4 Visualization

- Bounding box rendering
- Confidence score display
- Progress bar for video processing
- Split screen view for original/processed
- Real-time frame updates

4.5 Deployment

- Flask backend deployment
- React frontend serving
- Model serving
- REST API endpoints
- CORS configuration
- Error handling

5. Non-Functional Requirements

5.1 Performance

- Real-time video processing at 30 FPS
- Image processing under 2 seconds
- API response time under 1 second
- Support for multiple concurrent users
- Efficient memory management

5.2 Security

- CORS policy implementation
- Input validation
- Error handling
- Secure file uploads
- Safe model loading

5.3 Scalability

- Support for multiple ML models
- Horizontal scaling capability
- Modular architecture
- Containerization support
- Resource optimization