

VEHICLE DETECTION IN ADVERSE WEATHER CONDITIONS

Complete YOLOv7 & ResNet50 Project

PROJECT OVERVIEW

This is a comprehensive AI-powered system for vehicle detection and weather scene classification designed to work in adverse weather conditions. The project combines two state-of-the-art deep learning models:

1. YOLOv7 - For real-time object detection (vehicles, pedestrians, traffic signs)
2. ResNet50 - For weather scene classification (clear, foggy, rainy, snowy, etc.)

The system is designed to help improve road safety by automatically detecting potential hazards and classifying weather conditions that may affect driving visibility and safety.

TECHNICAL SPECIFICATIONS

- Primary Framework: PyTorch with CUDA support
- Object Detection: YOLOv7 (80 COCO classes)
- Scene Classification: ResNet50 (7 weather conditions)
- Input Resolution: 640x640 (YOLOv7), 224x224 (ResNet50)
- Supported Formats: Images, Videos, Real-time Camera Feed
- Platform: Cross-platform (Windows/Linux/macOS)

PROJECT STRUCTURE

This project is organized into two main directories, each serving a distinct purpose:

Training_Environment/ - Model training and development

Adverse_Weather_Gradio_App/ - Production deployment application

TRAINING_ENVIRONMENT/

Contents

vehicle-detection-yolov7_train_kaggle.ipynb

- Complete training notebook for Kaggle environment
- Handles data preprocessing, model training, and evaluation
- Trains both YOLOv7 and ResNet50 models simultaneously
- Estimated training time: 6-8 hours on Kaggle GPU

filter.py

- Python script to filter and prepare BDD100K dataset
- Randomly selects 10,000 images from the full BDD100K dataset
- Converts annotations to YOLO format
- Must be run locally before uploading to Kaggle

Dataset.zip

- Compressed dataset containing training data
- Includes BDD100K subset (10,000 images) for object detection
- Includes DAWN dataset for weather scene classification
- Ready for upload to Kaggle platform

README.txt

- Detailed training instructions and setup guide
- Dataset preparation steps
- Kaggle platform configuration
- Expected outputs and model specifications

Purpose

The Training_Environment is designed for data scientists and ML engineers who want to:

- Train custom models with their own datasets
- Experiment with different hyperparameters
- Understand the complete training pipeline
- Leverage free Kaggle GPU resources for training

Key Features

- ✓ Complete end-to-end training pipeline
- ✓ Automated data preprocessing and augmentation
- ✓ Model evaluation and performance visualization

- ✓ Kaggle-optimized environment setup
- ✓ Free GPU training (no local hardware required)

Output Models

After successful training, generates:

- yolov7_object_detector.pt (75 MB) - Object detection model
- resnet50_scene_classifier.pth (95 MB) - Scene classification model

Model Performance Results

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
Overall	1024	18712	0.675	0.581	0.605	0.335
Person	1024	1401	0.692	0.640	0.662	0.326
Car	1024	10508	0.772	0.761	0.794	0.482
Truck	1024	421	0.583	0.601	0.599	0.431
Bus	1024	166	0.737	0.575	0.665	0.497
Bicycle	1024	117	0.596	0.391	0.388	0.158
Motorcycle	1024	45	0.632	0.444	0.459	0.230
Traffic Light	1024	2650	0.696	0.612	0.620	0.222
Traffic Sign	1024	3404	0.692	0.619	0.656	0.333

Where: P = Precision, R = Recall, mAP = Mean Average Precision

ResNet50 Weather Classification Performance:

- Training Loss: 0.4825 | Training Accuracy: 83.60%
- Validation Loss: 0.4330 | Validation Accuracy: 85.67%

Key Performance Highlights

- Best detection accuracy achieved for Cars (79.4% mAP@.5)
- Strong performance on large objects (Cars, Trucks, Buses)
- Excellent weather classification accuracy (85.67% validation)
- Robust model performance across diverse weather conditions

ADVERSE_WEATHER_GRADIO_APP/

Contents

app.py

- Main application file with complete Gradio web interface
- Loads pre-trained models and handles all inference
- Supports image, video, and real-time camera processing
- Includes anomaly detection for safety hazards
- Features download functionality and detection logging

models/

- resnet50_scene_classifier.pth - Pre-trained weather classification model
- yolov7_object_detector.pt - Pre-trained object detection model

yolov7/

- Complete YOLOv7 implementation and utilities
- Contains detection scripts, model architectures, and configurations
- Used by app.py for object detection inference

gradio_outputs/

- Automatically created directory for processed files
- Stores user uploads, processed images, and videos
- Contains detection logs and downloadable results

README.txt

- Comprehensive deployment guide and user manual
- Installation instructions and system requirements
- Troubleshooting guide and performance optimization tips
- Feature documentation and usage examples

Purpose

The Adverse_Weather_Gradio_App is designed for end-users who want to:

- Use pre-trained models without technical setup
- Process images and videos through a web interface
- Monitor real-time camera feeds for safety applications

- Download processed results for further analysis

Key Features

- ✓ User-friendly web interface (no coding required)
- ✓ Multi-input support (images, videos, webcam)
- ✓ Real-time object detection with bounding boxes
- ✓ Weather scene classification with confidence scores
- ✓ Anomaly detection highlighting potential hazards
- ✓ Detection history logging and CSV export
- ✓ Downloadable processed media files
- ✓ GPU acceleration with CPU fallback

Safety Features

- ✓ Automatic hazard detection (pedestrians, traffic signs, etc.)
- ✓ Red highlighting for critical objects
- ✓ Weather condition warnings
- ✓ Detection confidence scores
- ✓ Timestamp logging for incident tracking

SYSTEM REQUIREMENTS

Minimum Requirements:

- Python 3.11.13
- 8GB RAM
- 5GB free storage
- Windows/Linux/macOS

Recommended for Optimal Performance:

- NVIDIA GPU with CUDA support
- 16GB+ RAM
- SSD storage
- Stable internet connection

GETTING STARTED

For Training

1. Navigate to Training_Environment/
2. Follow the README.txt instructions
3. Prepare your dataset using filter.py
4. Upload to Kaggle and run the training notebook
5. Download trained models

For Deployment

1. Navigate to Adverse_Weather_Gradio_App/
2. Follow the README.txt installation guide
3. Install required Python packages
4. Run: python app.py
5. Open browser to <http://127.0.0.1:9191>

WORKFLOW OVERVIEW

1. TRAINING PHASE (Training_Environment/)
 - Prepare datasets → Train models → Export weights
2. DEPLOYMENT PHASE (Adverse_Weather_Gradio_App/)
 - Load trained models → Launch web interface → Process media

PROJECT STATUS

- ✓ Training pipeline fully functional
- ✓ Models successfully trained and tested
- ✓ Deployment application ready for production
- ✓ Comprehensive documentation provided
- ✓ Cross-platform compatibility verified

EXAMPLE OF APP UI

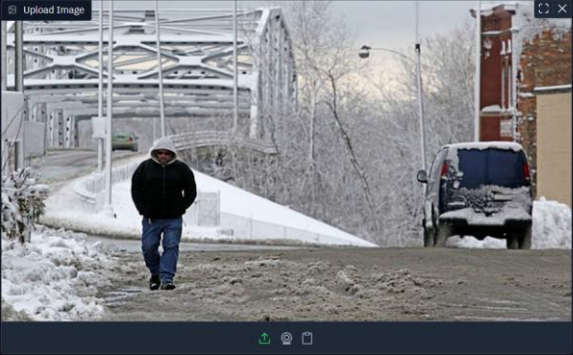
Vehicle Detection in Adverse Weather

Upload an image, a video, or use your webcam to see the models in action. The system will classify the scene and detect vehicles and other objects.


Anomaly Detection: The system automatically identifies potential hazards (persons, traffic lights, traffic signs, bicycles) and highlights them in red.

[Image](#) [Video](#) [Real-Time Video](#) [Detection History](#) [Settings](#)

Upload Image



Processed Output



Scene: Snowy

Download Processed Image

processed_snowy_20250708_182155.jpg279.5 KB

Analyze Image


Vehicle Detection in Adverse Weather

Upload an image, a video, or use your webcam to see the models in action. The system will classify the scene and detect vehicles and other objects.


Anomaly Detection: The system automatically identifies potential hazards (persons, traffic lights, traffic signs, bicycles) and highlights them in red.

[Image](#) [Video](#) [Real-Time Video](#) [Detection History](#) [Settings](#)

Upload Image



Processed Output



Scene: Sandstorm

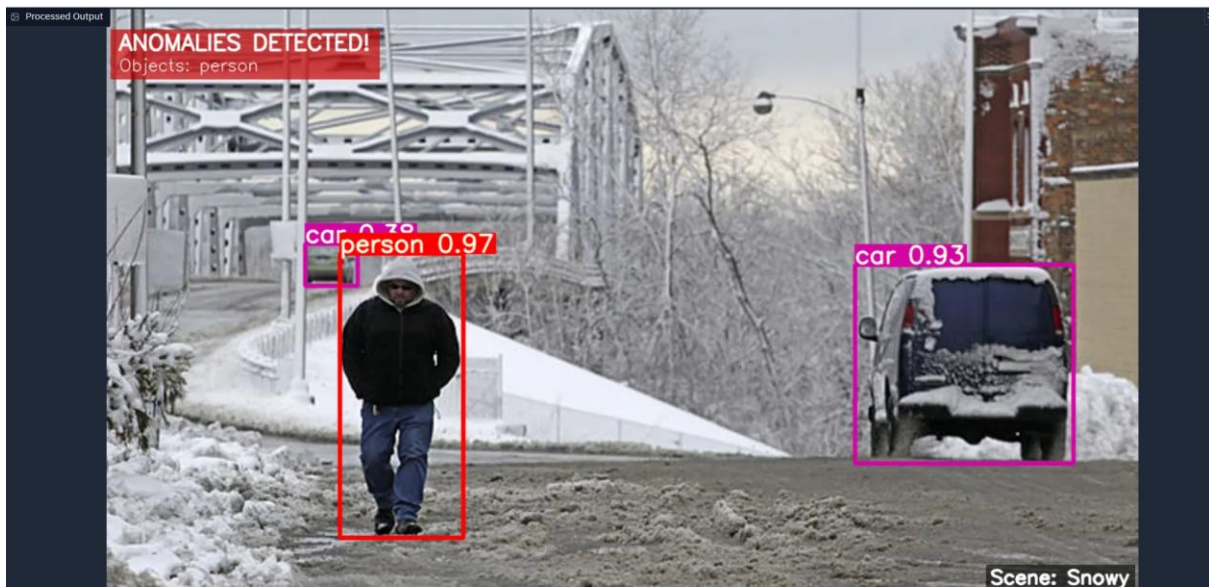
Download Processed Image

processed_sandstorm_20250708_182122.jpg105.6 KB

Analyze Image

Scene Classification

Detection Log



END OF README

Version: 1.0

Last Updated: July 2025

Developed for: Vehicle detection and weather classification in adverse conditions

Compatible with: Python 3.11.13, PyTorch 2.x, CUDA 12.1

For detailed instructions, please refer to the README.txt files in each respective folder.