



Deep Learning-Based Industrial Vision Inspection Platform

[Features](#) • [Architecture](#) • [Getting Started](#) • [Project Structure](#) • [Roadmap](#)

.NET .NET 8.0 | WPF Modern UI | ONNX Runtime GPU | PyTorch 2.0+ | License MIT

📘 Overview

AI Vision Inspector is a full-stack AI vision inspection software designed for industrial manufacturing scenarios. It integrates **data management**, **model training**, **real-time inference**, **camera acquisition**, and **incremental learning** into a complete workflow.

AI视觉检测工具是一款专为工业制造场景设计的全栈式AI视觉检测软件。它将**数据管理**、**模型训练**、**实时推理**、**相机采集**和**增量学习**整合到一个完整的工作流程中。

This project addresses the pain points of traditional industrial vision inspection systems:

- ❌ Traditional approach: Requires specialized algorithm engineers, complex environment setup, expensive commercial software licenses
❌ 传统方法：需要专业的算法工程师、复杂的环境设置以及昂贵的商业软件许可
- ✅ This solution: **No-code training, one-click deployment, on-site adaptive learning**

🎯 Use Cases

Scenario	Description	Algorithm
Defect Detection	Surface scratches, stains, cracks, missing parts	PatchCore / STFPM
Product Classification	Model identification, OK/NG sorting	ResNet / MobileNet
Object Counting	Part counting, assembly completeness check	YOLOv8 / YOLOv11

🌟 Features

🧠 AI Inference Engine

- **Multi-task Support:** Anomaly detection, image classification, object detection
- **GPU Acceleration:** ONNX Runtime + CUDA, inference latency < 50ms
- **Heatmap Visualization:** Intuitive display of anomaly region localization
- **Adaptive Thresholds:** Strict / Balanced / Loose three-tier automatic threshold recommendation

Camera Acquisition System

- **Multi-brand Support:** Hikvision (MVS SDK), Daheng, Basler (reserved interfaces)
- **Real-time Preview:** 30+ FPS continuous acquisition
- **Parameter Control:** Real-time adjustment of exposure, gain, gamma
- **Auto Inference:** Capture-and-detect, seamless AI pipeline integration

Model Training Pipeline

- **No-code Training:** GUI-based configuration, one-click training start
- **Real-time Monitoring:** Training progress, loss curves, log streaming
- **Auto Deployment:** Automatic ONNX export and system registration upon completion
- **Training History:** Complete records of training parameters and metrics

Incremental Learning System

- **On-site Feedback:** One-click labeling of misdetected samples (OK/NG)
- **Smart Updates:** Automatic model fine-tuning based on feedback samples
- **Version Management:** Automatic backup of historical versions, one-click rollback support
- **Auto Validation:** Post-update automatic validation, auto-rollback on performance degradation

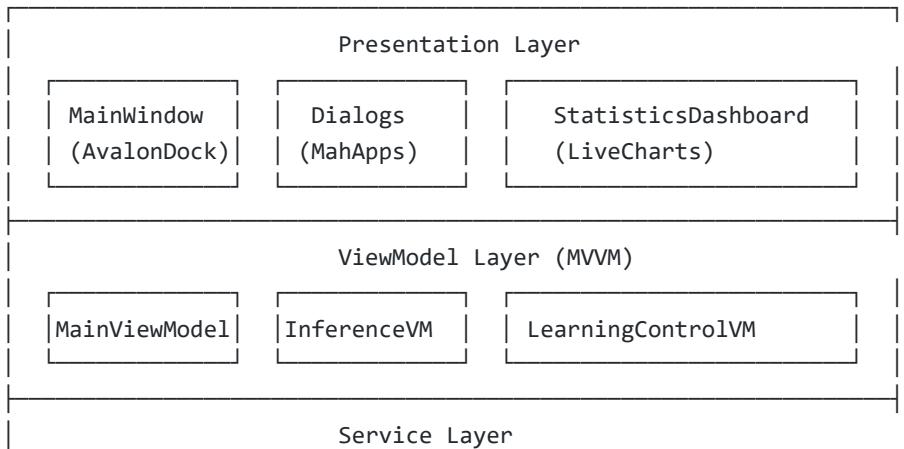
Data Management

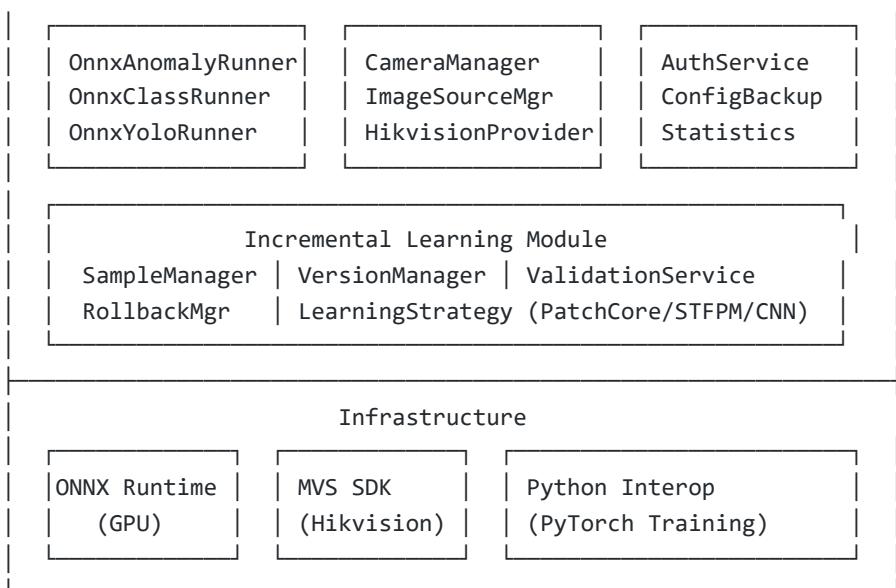
- **Dataset Wizard:** Visual creation of MVTec / ImageFolder / YOLO format datasets
- **Batch Import:** Drag-and-drop import with automatic filename normalization
- **Statistics Dashboard:** Real-time statistics, trend charts, yield analysis

Enterprise Features

- **Permission Management:** Admin / Operator two-level permissions
- **Configuration Backup:** One-click export/import of all configurations
- **History Tracing:** Automatic archiving of detection results, date-based queries

Architecture





Tech Stack

Layer	Technology	Description
Frontend Framework	WPF + .NET 8	Modern desktop application
UI Components	MahApps.Metro + AvalonDock	VS2022-style theme
Charting Library	LiveChartsCore	Real-time data visualization
Inference Engine	ONNX Runtime GPU	CUDA-accelerated inference
Training Framework	PyTorch + Anomalib	Anomaly detection algorithm library
Camera SDK	Hikvision MVS SDK	Industrial camera acquisition
Configuration	YamlDotNet	YAML format configuration

🚀 Getting Started

Requirements

- **Operating System:** Windows 10/11 64-bit
- **Runtime:** .NET 8.0 Runtime
- **GPU (Optional):** NVIDIA GPU + CUDA 11.x (for accelerated inference and training)
- **Python (Training):** Python 3.10+ or Anaconda

Installation

```

# 1. Clone the repository
git clone https://github.com/yourusername/ai-vision-inspector.git
cd ai-vision-inspector

# 2. Install Python dependencies (for training features)

```

```
pip install -r scripts/requirements.txt
```

```
# 3. Build and run  
dotnet build WpfAnomalyMvp.sln  
dotnet run --project WpfAnomalyMvp
```

Default Login

- **Username:** admin
- **Password:** admin123

📁 Project Structure

```
ai-vision-inspector/  
├── WpfAnomalyMvp/          # Main application project  
│   ├── Views/              # XAML views  
│   ├── ViewModels/          # MVVM view models  
│   ├── Services/             # Business service layer  
│   │   ├── IncrementalLearning/ # Incremental learning module  
│   │   └── Interfaces/        # Service interface definitions  
│   ├── Models/              # Data models  
│   └── Themes/              # UI theme styles  
├── WpfAnomalyMvp.Tests/    # Unit test project  
├── scripts/                # Python training scripts  
│   ├── train_anomaly.py     # Anomaly detection training  
│   ├── train_classifier.py  # Classification model training  
│   └── finetune_*.py       # Incremental learning scripts  
├── configs/                # Configuration files  
│   ├── registry.yaml        # Model registry  
│   └── thresholds.yaml      # Threshold configuration  
└── models/                 # ONNX model files  
    └── data/                  # Dataset directory
```

📊 Performance Metrics

Metric	Value	Test Environment
Inference Latency (GPU)	15-30 ms	RTX 3060, 224×224
Inference Latency (CPU)	80-150 ms	i7-12700
Camera Capture FPS	30+ FPS	Hikvision MV-CS050-10GC
Training Speed	~2 min/epoch	RTX 3060, 100 images
Memory Usage	500-800 MB	Single model loaded

📅 Roadmap

Completed

- [x] Core inference engine (PatchCore/STFPM/YOLO)
- [x] Camera acquisition system (Hikvision)
- [x] Model training pipeline
- [x] Incremental learning system
- [x] Statistics dashboard
- [x] Permission management system
- [x] Configuration backup/restore

In Progress

- [] Adaptive threshold learning
- [] Daheng/Basler camera adapters

Planned

- [] OCR character recognition
- [] Web remote monitoring
- [] Industrial protocol support (Modbus/OPC UA)
- [] Multi-camera synchronization

Key Design Patterns

MVVM Architecture

Separation of views and business logic for better testability and maintainability.

Strategy Pattern

Extensible learning strategies for different model types (PatchCore, STFPM, Classification).

Interface Abstraction

`IImageSource` and `ICameraProvider` interfaces enable easy extension for new camera brands.

Service Locator

Centralized service registration and resolution for dependency management.

Contributing

Contributions are welcome! Please see [CONTRIBUTING.md](#) for guidelines.

1. Fork the repository

2. Create a feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

License

This project is licensed under the MIT License - see the [LICENSE](#) file for details.

Contact

- **Author:** Lynn Yan
- **Email:** yanxue6886@163.com
- **LinkedIn:** [Lynn](#)