**CloudFactory Object Detection Pipeline:**

**Installation & Setup**

**Dependencies Installation**

# Create virtual environment

python -m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

# Install dependencies

pip install -r requirements.txt

detectron2==0.6

**Important: Use numpy<2.0.0 to avoid compatibility issues with PyTorch.**

**Basic Execution**

# Run with default settings

python app.py

# Process images with custom confidence threshold

python app.py --confidence\_threshold 0.5

# Adjust NMS threshold for detection overlap

python app.py --nms\_threshold 0.3

# Enable debug logging

python app.py --debug

**To run according to the Custom Paths :**

docker run image-detection python app.py \ --images\_dir /path/to/images \ --model\_path /path/to/model \ --class\_mapping\_path /path/to/class\_mapping \ --output\_dir /path/to/output

python app.py --images\_dir ./Image --model\_path ./fish\_detector/model.pt --class\_mapping\_path ./fish\_detector/class\_mapping.json --output\_dir ./output

Readme :

# CloudFactory Fish Detector

This project is a Dockerized pipeline for detecting fish in images using PyTorch.

The pipeline reads images from the `Image/` folder, runs inference using a pre-trained model in `fish\_detector/`, and outputs results to the `output/` folder.

---

## Features

- Runs completely inside a Docker container.

- No installation of Python dependencies on the host required.

- Prepares outputs automatically in the `output/` folder.

---

## Known Dependency Warning

When running the pipeline, you may see a warning like:

**Failed to initialize NumPy: A module that was compiled using NumPy 1.x cannot be run in NumPy 2.x...**

This happens because:

- PyTorch was compiled with NumPy 1.x.

- Docker image currently has NumPy 2.2.6 (required for OpenCV 4.12+).

This is a warning only the pipeline has been tested and runs successfully. Outputs are correctly generated.

---

## 🐳 Running the Docker Container

### Build Docker image (if needed):

```bash

docker build -t fish-detector .

**Overview**

This project implements a production-ready ML pipeline for batch object detection using CloudFactory's TorchScript models. The pipeline processes images through the complete workflow from preprocessing to COCO format output generation.

**Solution Architecture**

**Core Components**

1. **Image Preprocessing Module** (CloudFactoryImageProcessor)
   * Loads and applies CloudFactory's serialized transforms
   * Handles image format conversions (RGBA → RGB)
   * Maintains preprocessing consistency between training and inference
2. **Object Detection Module** (CloudFactoryObjectDetector)
   * Supports both Faster R-CNN and FBNetv3 model formats
   * Implements CloudFactory's exact inference patterns
   * Applies Non-Maximum Suppression for duplicate removal
3. **Results Export Module** (COCOResultsExporter)
   * Generates industry-standard COCO format output
   * Includes detection metadata and confidence scores
   * Compatible with evaluation and visualization tools
4. **Pipeline Orchestrator** (MLPipelineRunner)
   * Coordinates all pipeline components
   * Handles batch processing and error recovery
   * Manages resource cleanup and logging

**Error Resilience**: The pipeline continues processing even if individual images fail, ensuring maximum throughput in production environments.

**Technical Implementation Details**

**Transform Handling**:

* Prioritizes CloudFactory's serialized transforms.json for consistency
* Falls back to standard 224x224 resize if transforms unavailable
* Handles various image formats (PNG, JPEG, RGBA) automatically

**Model Inference Pattern**:

* Follows CloudFactory's exact tensor preparation (channels-first, float conversion)
* Supports both dictionary and tuple output formats
* Implements proper batch dimension handling

**Memory Management**:

* GPU memory cleanup after processing
* Efficient tensor operations to minimize memory usage
* Resource cleanup in finally blocks

**File Structure**

project/

├── app.py # Main pipeline script

├── fish\_detector/

│ ├── model.pt # TorchScript model

│ ├── class\_mapping.json # Class definitions

│ └── transforms.json # Image transforms config

├── image/ # Input images directory

│ ├── image1.jpg

│ └── image2.jpg

├── output/ # Generated results

│ └── detection\_results\_TIMESTAMP.json

└── requirements.txt # Python dependencies

**Key Fields Explanation**

* **bbox**: [x, y, width, height] in pixels
* **score**: Model confidence (0.0 to 1.0)
* **area**: Bounding box area in square pixels
* **category\_id**: Maps to class in categories array

**Results Analysis**

Based on the provided detection results:

**Performance Summary**

* **Total Images Processed**: 5
* **Total Detections**: 12 fish detected
* **Detection Rate**: 80% (4 out of 5 images had detections)

**Confidence Distribution**

* **High Confidence (>80%)**: 3 detections
  + Best: 98.4% (great\_blue\_heron\_with\_fish)
  + Second: 93.9% (Koi\_fish\_in\_the\_water)
  + Third: 81.2% (Posidonia\_oceanica)
* **Medium Confidence (50-80%)**: 2 detections
* **Lower Confidence (<50%)**: 7 detections

**Image-by-Image Results**

1. **Koi\_fish\_in\_the\_water.jpg**: 5 fish (best: 93.9%)
2. **Madeira\_Fish.jpeg**: 5 fish (best: 61.6%)
3. **Malate\_Manila\_Aquarium\_Fish.jpg**: 0 fish (challenging image)
4. **Posidonia\_oceanica.jpg**: 1 fish (81.2%)
5. **great\_blue\_heron\_with\_fish.jpg**: 1 fish (98.4%)

**Testing**

**Running Unit Tests**

# Run all tests

python -m pytest test\_pipeline.py -v

# Run specific test class

python -m pytest test\_pipeline.py::TestCloudFactoryImageProcessor -v

# Run with coverage

python -m pytest test\_pipeline.py --cov=app --cov-report=html

**Test Coverage Areas**

* Image preprocessing with various formats
* Transform loading and application
* COCO format generation and validation
* Error handling for edge cases
* Mock model inference testing

**Docker Deployment**

**Build and Run**

# Build Docker image

docker build -t fish-detector .

# Run pipeline in container

docker run --rm -v $(pwd)/output:/app/output fish-detector

# Run with custom parameters

docker run --rm -v $(pwd)/output:/app/output fish-detector \

python app.py --confidence\_threshold 0.5 --debug

**Prerequisites**

- Docker installed: [https://www.docker.com/get-started](https://www.docker.com/get-started)

- Python dependencies are handled inside Docker, no local Python setup required.

# Build Docker Image

Run the following from the `cloudfactory/` folder:

```bash

docker build -t fish-detector .

Run The pipeline:

docker run --rm -v $(pwd)/output:/app/output fish-detector

**Troubleshooting**

**Common Issues**

**NumPy Compatibility Error**:

# Fix: Downgrade NumPy

pip install "numpy<2.0.0"

**Transform Loading Error**:

* Verify transforms.json format matches CloudFactory specification
* Check albumentations version compatibility
* Use default transforms if custom ones fail

**No Detections Found**:

* Lower confidence threshold: --confidence\_threshold 0.1
* Adjust NMS threshold: --nms\_threshold 0.3
* Verify image quality and class mapping