

DetactaX AI

(AI Vision Platform)

DEPI Round 3 Graduation Project

11/29/2025

DetactaX – Image Classification & Object Recognition
Reimagined.

eyouth®

Project team:

- **Zyad Gamal** (team leader): Azure Integration & API Deployment.
- **Basel Mohamed**: Both Classification Models Designer, Mlflow, and Project Documentation.
- **Omar Yasser**: Project Models Interface GUI using Streamlit.
- **Mohamed Hamada**: Object Recognition Model.
- **Zyad Ahmed**: Dataset Acquisition & Augmentations.
- **Abdulrahman Kamal**: Baseline Classifier Transfer Learning.



Abstract:

DetactaX is a ***unified AI vision platform*** that integrates image **classification**, **object detection**, **real-time analytics**, interpretability tools, and **cloud deployment** into a single cohesive system.

Unlike traditional vision solutions that focus on isolated tasks, DetactaX delivers an **end-to-end workflow** that spans data ingestion, preprocessing, model inference, visualization, and results export.

- The platform is designed to be **accessible**, **efficient**, and **deployable at scale**, enabling both technical and non-technical users to generate high-quality visual insights with minimal setup.

Introduction:

- Existing tools are complex or slow and require technical expertise.
- **Users need:**
 1. Easy-to-use interface
 2. Quick predictions
 3. High accuracy
 4. No technical setup required
- **Project Idea:** Making a website with both Image Classification & Object Recognition capabilities for multi-use cases.
- **Goal:** Quickly and accurately classify the main object in an image & recognize multiple objects using top of the line models.

Introduction:

Our project is **not just a classifier or a detector.**

We provide a complete **Solution platform** that combines image classification, object detection, real-time analysis, Grad-CAM visualization, reporting, and Azure endpoints.

The main goal is to offer a **general visual data preparation system** that any AI team can build advanced applications on top of, such as:

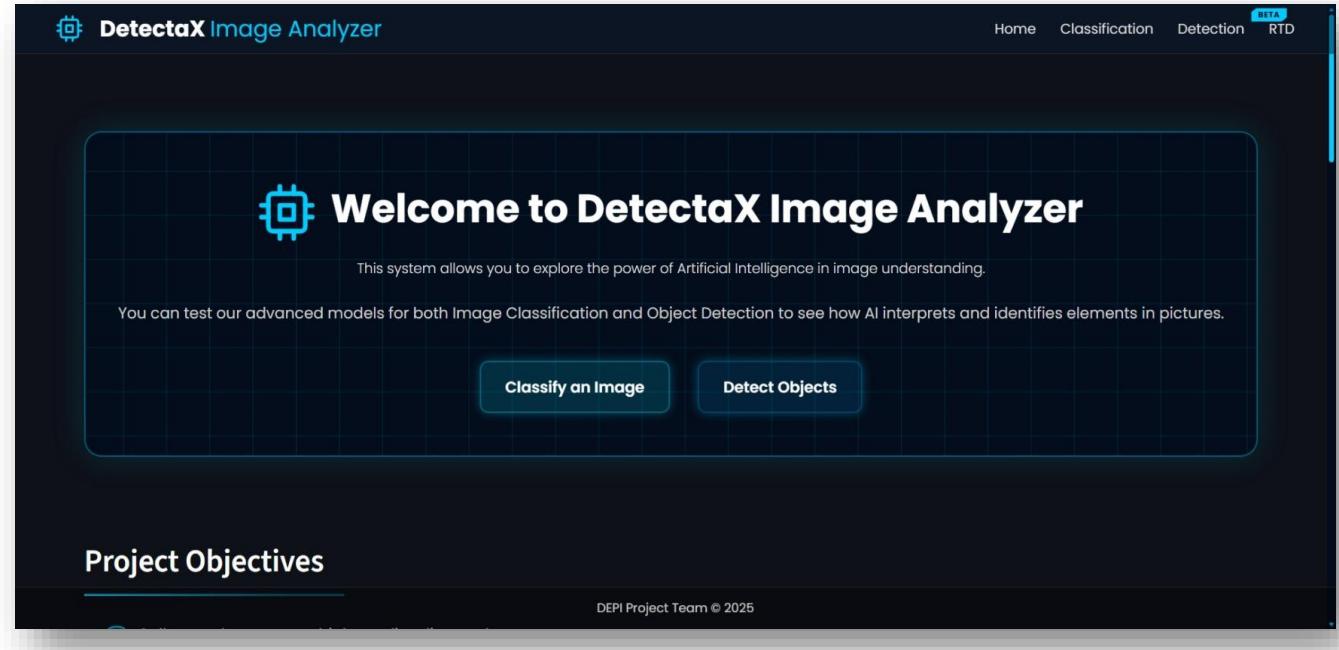
1. Security and surveillance
2. Traffic analysis
3. Product analysis
4. Sorting systems
5. Research dataset generation



Services Offered:

1. Image upload
2. Data cleaning
3. Object extraction
4. Classification
5. Result exporting
6. Grad-CAM visualization
7. PDF reports
8. JSON / CSV export
9. Real-time feed

This makes the platform **ready for downstream tasks.**



The screenshot shows the homepage of the DetectaX Image Analyzer. The header features the logo and name "DetectaX Image Analyzer" along with navigation links for Home, Classification, Detection, and RTD. The main content area has a dark background with a light blue grid. It displays a large button with the text "Welcome to DetectaX Image Analyzer" and a subtext explaining the system's purpose: "This system allows you to explore the power of Artificial Intelligence in image understanding. You can test our advanced models for both Image Classification and Object Detection to see how AI interprets and identifies elements in pictures." Below this are two buttons: "Classify an Image" and "Detect Objects". At the bottom, there is a section titled "Project Objectives" and a copyright notice: "DEPI Project Team © 2025".

CIFAR-10 Classifier Dataset

11/29/2025

DetactaX – Image Classification & Object Recognition
Reimagined.

eyouth®

General Classifier (Base-line/Control):

Full Name: Canadian Institute For Advanced Research (CIFAR) - **10 classes.**

- **Creators:** Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton (University of Toronto).
- **Purpose:** A standard benchmark for evaluating image classification algorithms, specifically Convolutional Neural Networks (CNNs).

Data Specifications:

- **Total Volume:** 60,000 labeled images.
 - **Training Set:** 50,000 images (5 batches of 10k).
 - **Test Set:** 10,000 images (1 batch of 10k).
- **Dimensions:** Low-resolution **32 x 32 Pixels.**
- **Format:** Color images with **3 channels (RGB).**

General Classifier (Base-line/Control):

Augmentations Applied on the Dataset:

1. Resizing to 224x224.
2. Random Flip Horizontal and Vertical.
3. Random Brightness.
4. Random Contrast.
5. Random Rotation.
6. Random Zoom.
7. Rescaling (1/255).

```
data_augmentation = keras.Sequential([
    layers.Resizing(224, 224),
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomBrightness(0.1),
    layers.RandomContrast(0.1),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.Rescaling(1.0 / 255.0)
])
```

Applied Transfer Learning

Transfer Learning

- Transfer learning is a machine learning technique where a model trained on one task is repurposed as the starting point for a related task. Instead of training from scratch, you leverage pre-existing knowledge.
- **How it works:**
 - Start with a model pre-trained on a large, general dataset (e.g., ImageNet for images).
 - Remove the final classification layer.
 - Add new layers tailored to your specific task.
 - Fine-tune the model on your smaller, specific dataset.
- **Why we apply it:**
 - 1. Saves Time & Resources.**
 - 2. Better Performance with Less Data.**
 - 3. Improves Accuracy.**

Transfer Learning

Applied transfer learning on
Baseline/model (MobileNetV2).

Where we unfroze the last 50
Layers of the Model to be able
To apply much finer training of
The model on our dataset.

3.5. Training the 2nd iteration of the model

In this section, we will train the second iteration of the model.

- 50 Layers unfrozen (trainable).
- epochs 20.

```
# Unfreezing the last 50 layers of the models
base_model.trainable = True
fine_tune = 50

# Freeze all the layers before the `fine_tune` layer
for layer in base_model.layers[:-fine_tune]:
    layer.trainable = False
```

Classification Models

Technology Stack & Versions

- Python 3.12
- Streamlit 1.28.0
- OpenCV 4.8.1.78
- Pillow 10.0.1
- NumPy 1.24.3
- Requests 2.31.0
- Matplotlib 3.7.2
- Plotly 5.15.0
- Kaleido 0.2.1
- Ultralytics 8.1.0
- Torch 2.1.0

Classifiers:

- **Models used:**
 1. MobileNetV2.
 2. EfficientNet-B0.

Both models are employed for different use-cases in our project, where:

- MobileNetV2 is used as a baseline test **general classifier**.
- EfficientNet-B0 used for **Intelligent Transportation systems**.

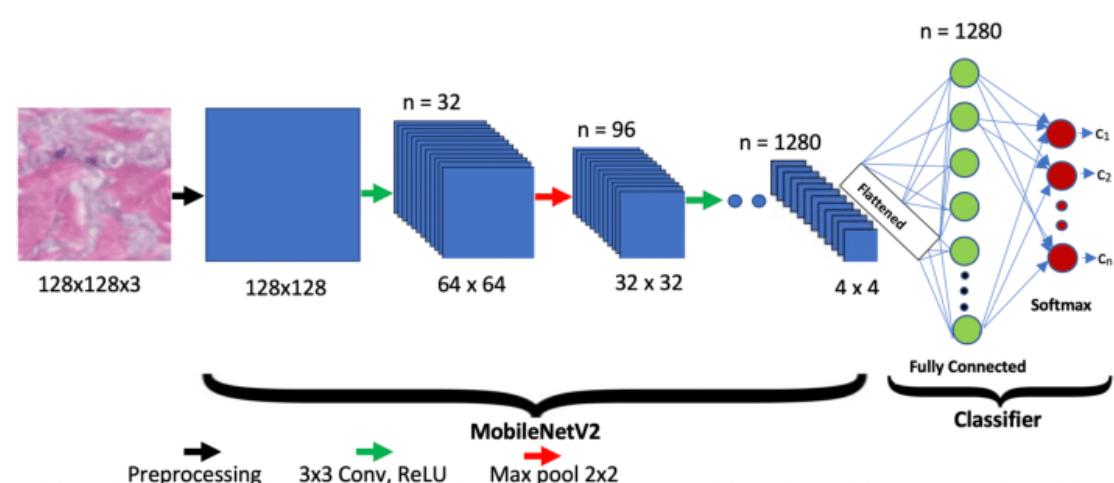
Baseline General Classifier:

Model Used: MobileNetV2 trained on **CIFAR-10 Dataset.**

- MobileNetV2 released **by google** is a very light weight yet efficient model released mainly for Mobile & Embedded systems applications.

Important three points:

1. Inverted Residuals.
2. Linear Bottlenecks.
3. Highly Efficient.





Baseline General Classifier Training:

Trained the model two times, each time with different hyperparameters and more aggressive training cycles.

First Training Cycle:

1. 10 epochs.
2. All layers frozen.
2. Accuracy: 0.6610.
3. Loss: 0.9548.
4. Val. Accuracy: 0.7584.

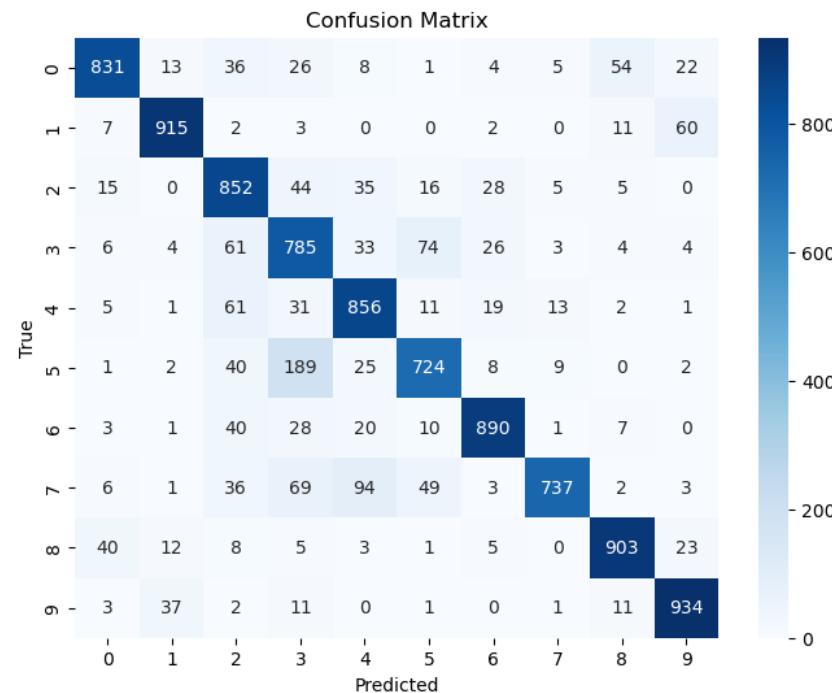


Baseline General Classifier Training:

Trained the model two times, each time with different hyperparameters and more aggressive training cycles.

Second Training Cycle:

1. 50 epochs.
2. Unfreeze last 50 layers.
3. Accuracy: 0.8451
4. Loss: 0.4446
5. Validation Accuracy: 0.8404



Intelligent Transportation System (Main Classifier):

Model Used: EfficientNet-B0 trained on the CUHK-Compcars dataset.

Dataset Details:

- Contains **163 car makes** with **1,716 car models**. There are a **total of 136,726 images**. The full car images are labeled with bounding boxes and viewpoints.

Augmentations Applied:

1. Image Resize.
2. Color Jitter.
3. Normalization
4. Random Horizontal Flipped.

```
# Train Transforms
train_transform = v2.Compose([
    v2.Resize((224, 224)),
    v2.RandomHorizontalFlip(p=0.5),
    v2.ColorJitter(brightness=0.15, contrast=0.15),
    v2.ToImage(),
    v2.ToDtype(torch.float32, scale=True),
    v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Validation Transforms
val_transforms = v2.Compose([
    v2.Resize((224, 224)),
    v2.ToImage(),
    v2.ToDtype(torch.float32, scale=True),
    v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```



Car Classification Results

- Make: ['Volkswagen'] (100.0%)
- Model: ['Tiguan abroad version'] (57.7%)
- Year: 2015 (28.3%)
- Overall Confidence: 62.0%



Car Classification Results

- Make: ['Daihatsu'] (99.5%)
- Model: ['MATERIA'] (100.0%)
- Year: 2008 (95.5%)
- Overall Confidence: 98.3%



Car Classification Results

- Make: ['Hyundai'] (100.0%)
- Model: ['i10'] (99.9%)
- Year: 2011 (97.1%)
- Overall Confidence: 99.0%



Car Classification Results

- Make: ['Volkswagen'] (100.0%)
- Model: ['Golf'] (54.3%)
- Year: 2009 (58.2%)
- Overall Confidence: 70.8%



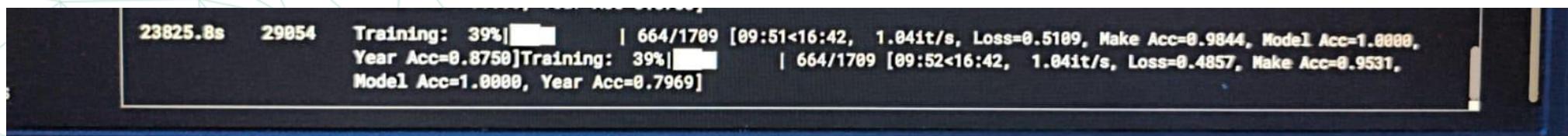
Models Specifications:

Training Cycles:

- **50 Epochs** training cycle with Validation stages in-between for Baseline Classifier (MobileNetV2), Training Took about 6 Hours.
- **18 Epochs** training cycle with Validation stages in-between for Main Car Classifier (EfficientNet-B0), Training took about 11 Hours.

Final Models Results:

- Accuracy: 0.8451, Loss: 0.4446, Val Accuracy: 0.8404.



Object Recognition Model

Model Details:

- **Object Detection system** using a **pre-trained YOLO model (yolov12x.pt)** from the Ultralytics library. The goal was to build a model capable of detecting and classifying multiple objects in images with high accuracy and fast inference speed.
- **Model Setup and Evaluation**
- to measure its performance. The evaluation automatically generated detection results and key metrics including:
 - **mAP@0.5** – Measures the overlap quality between predicted and ground-truth bounding boxes.
 - **mAP@0.5:0.95** – A stricter score averaged across 10 IoU thresholds.
 - **Precision** – Percentage of predicted boxes that are correct.
 - **Recall** – Percentage of actual objects successfully detected.
 - These metrics provide a full understanding of both accuracy and robustness of the object detection model.



Model Details:

- Key Code Operations:

1) Load YOLO model:

```
model = YOLO("yolov12x.pt")
```

2) Run validation on COCO128:

```
metrics = model.val(data="coco128.yaml", imgsz=640)
```

3) Display sample detection results:

```
display(Image(filename=img_path))
```

4) Export structured performance report:

```
with open("model_evaluation_report.txt", "w") as f:  
    f.write(report)]
```

Summary

The YOLOv12x model successfully completed object detection on the validation data, generating reliable performance scores and visual outputs. The evaluation report summarizes the model's strength in identifying objects with strong precision the project requirements for a trained and evaluated object detection system.

and recall, meeting



MIflow

DetactaX – Image Classification & Object Recognition
Reimagined.

11/29/2025

eyouth®

26



Key MLflow Components:

- **MLflow**: Central experiment tracker.
- **Local SQLite**: Development tracking database.
- **Custom Monitors**: Production performance tracking and metrics documenting.
- **Azure Integration**: Cloud deployment bridge for easy model redeployment to Azure.

Results & Metrics:

```
mlruns\3
  a00c2fe554214d10b129a3c61969e92c
  d698b168da4f4072bd7a5fa0ba9db668
  e4de11a5b34a46faa6fbe0c4e98063b8
  f6cd3ae6570347059a9f3e3bdfdf4fca
phase1_artifacts
  class_names.txt
  training_history.png
phase2_artifacts
  class_names.txt
  training_history.png
  mlflow.db
  model_registry.py
  requirements.txt
```

- Classification Model Performance

1. Final Test Accuracy: 82%
2. Training Time: ~60 minutes (with MLflow tracking)
3. Model Versions: 3+ registered versions
4. Experiments Tracked: 10+ complete runs

- Monitoring System

1. Prediction Tracking: Real-time with <100ms overhead
2. Data Collected: Parameters, metrics, artifacts, predictions



mlflow 3.6.0

+ New

Home Experiments Models Prompts

Welcome to MLflow

Get started

Log traces
Trace LLM applications for debugging and monitoring.

Run evaluation
Iterate on quality with offline evaluations and comparisons.

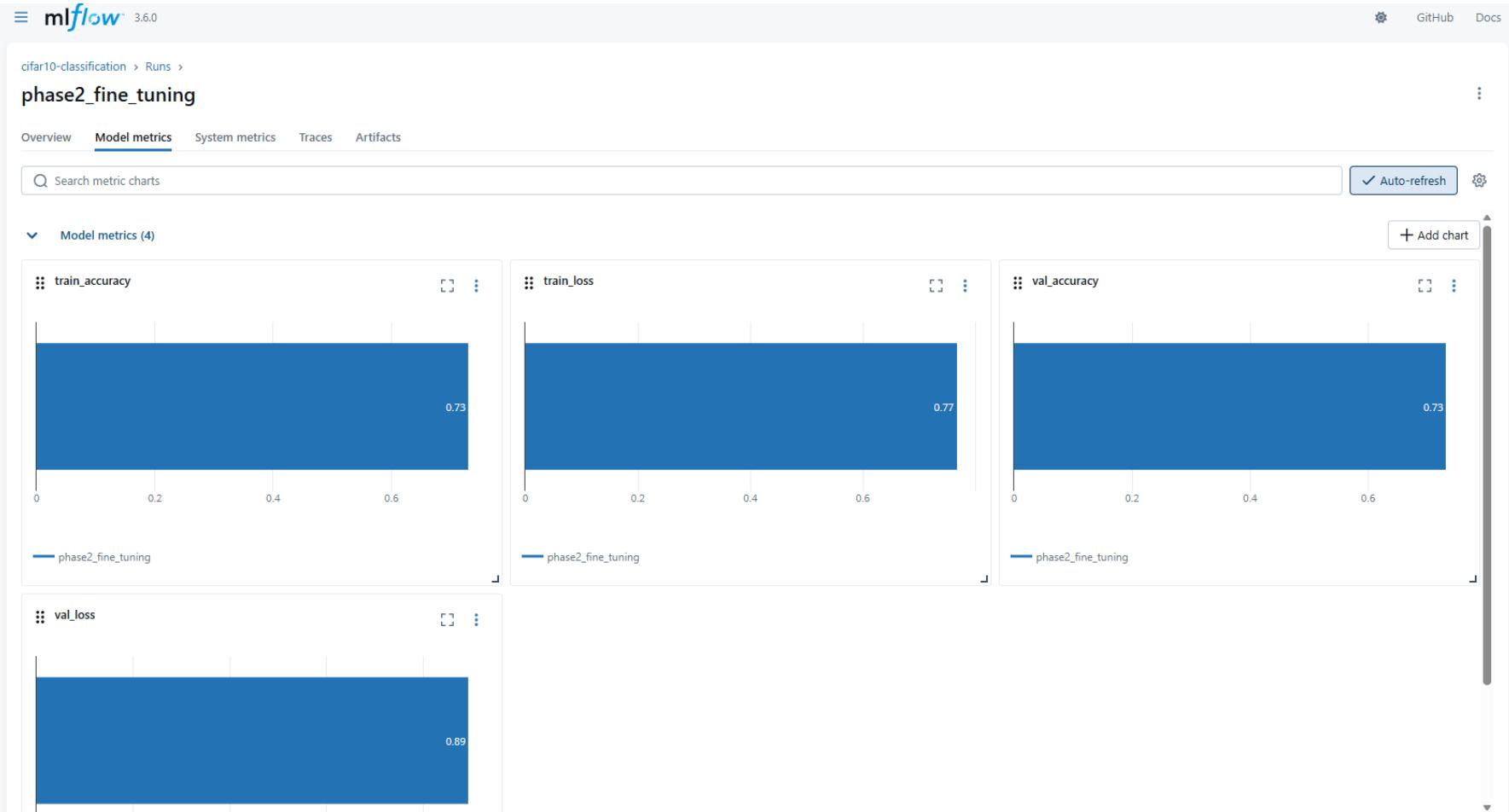
Train models
Track experiments, parameters, and metrics throughout training.

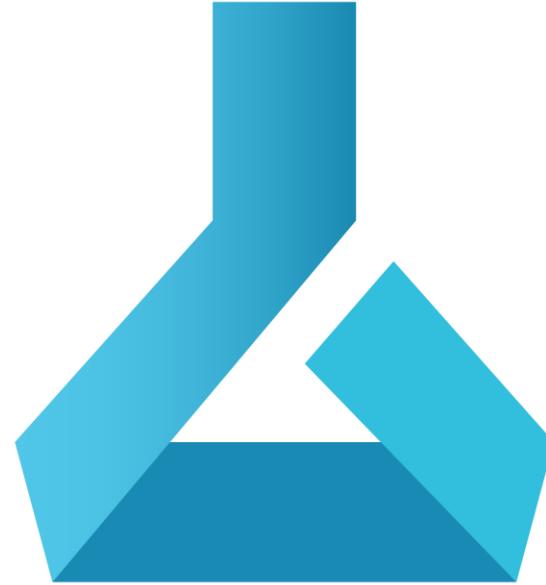
Register prompts
Manage prompt updates and collaborate across teams.

Experiments

[View all](#)

<input type="checkbox"/> Name	Time created	Last modified	Description	Tags
<input type="checkbox"/> cifar10-classification	11/15/2025, 12:07:54 PM	11/15/2025, 12:07:54 PM	-	
<input type="checkbox"/> test-detection-experiment	11/14/2025, 08:06:26 PM	11/14/2025, 08:06:26 PM	-	
<input type="checkbox"/> test-classification-experiment	11/14/2025, 08:04:48 PM	11/14/2025, 08:04:48 PM	-	
<input type="checkbox"/> Default	11/14/2025, 08:04:48 PM	11/14/2025, 08:04:48 PM	-	





Model Deployment

Why Deploy?

- Make the **model accessible to real users.**
- Deliver predictions instantly and at scale.
- Integrate the system into real workflows.
- Ensure consistent, reliable performance.

Why Azure?

- Reliable cloud infrastructure.
- Easy model deployment with Azure ML.
- Scalable compute for heavy workloads.
- Built-in security and compliance.



Azure Deployment Metrics:

depi-cars-cnn

Details Test Consume **Monitoring** PREVIEW Logs

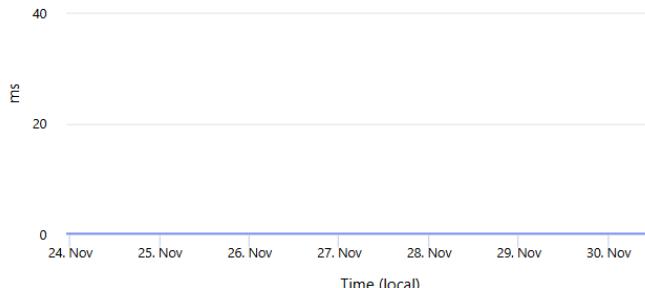
Monitoring

11/23/2025 - 11/30/2025 Last day 7D 1M 3M 1Y

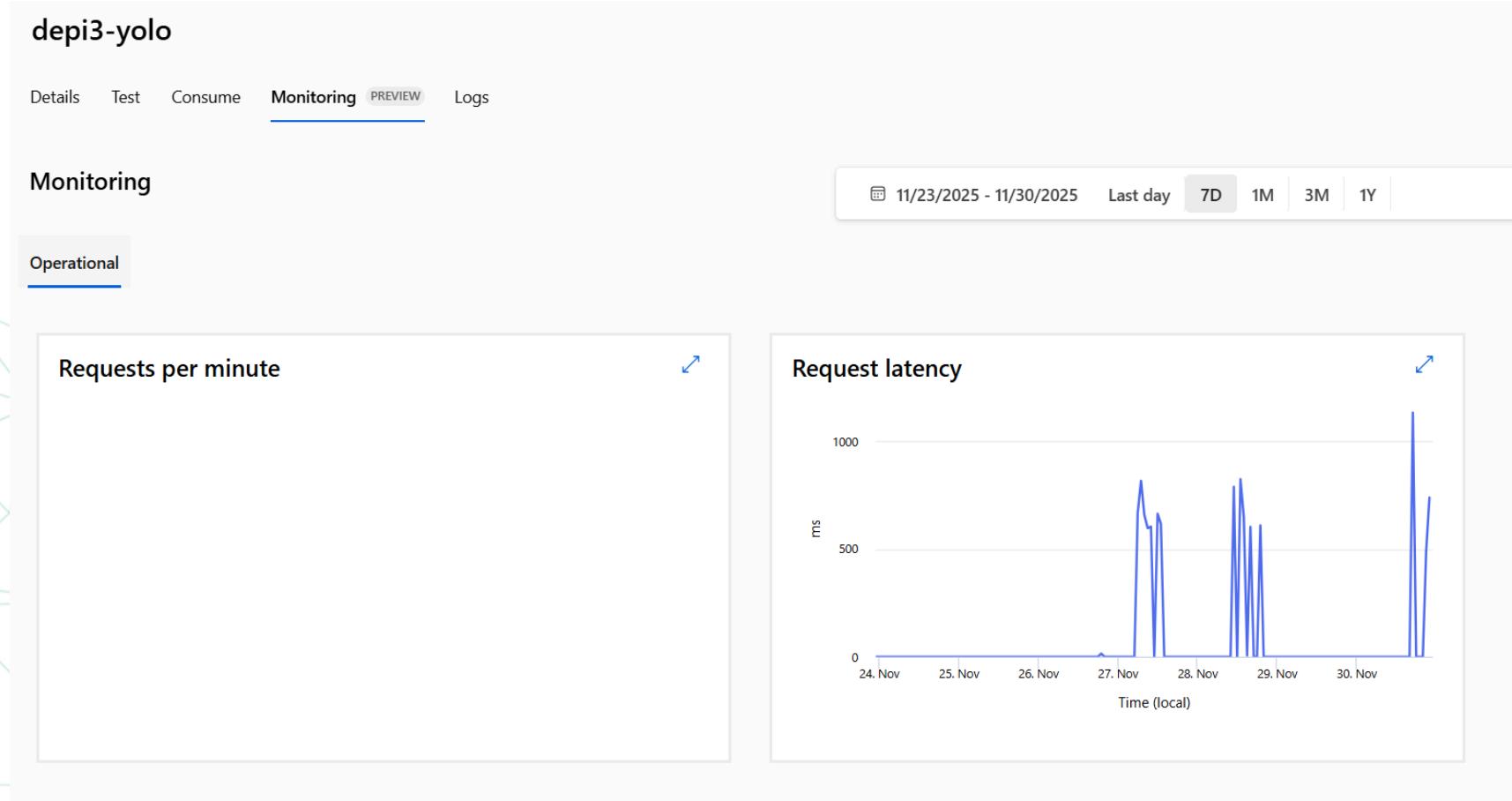
Operational

Requests per minute

Request latency



Azure Deployment Metrics:

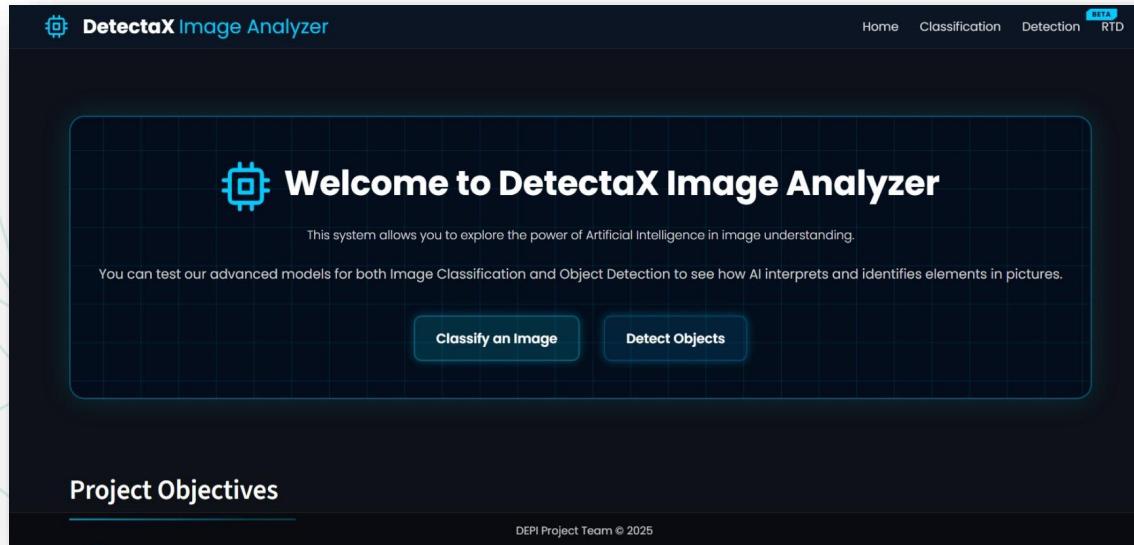


11/29/2025

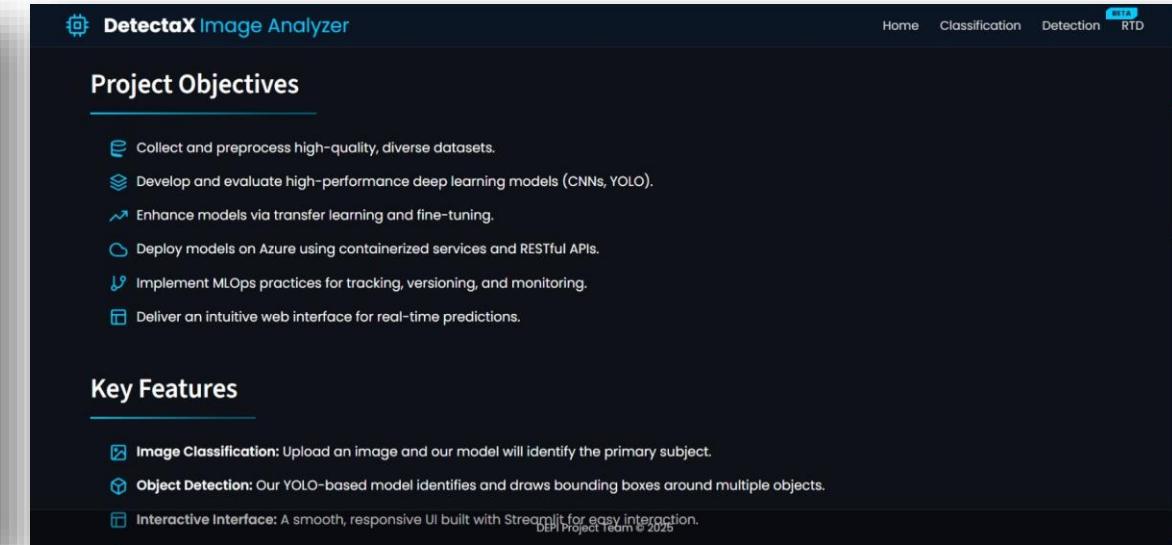
Web Interface (Frontend & API Integration)



Web Interface Overview



The screenshot shows the homepage of the DetectaX Image Analyzer. At the top, there is a navigation bar with links for Home, Classification, Detection, and RTD. Below the navigation bar is a large central box with a blue border containing the text "Welcome to DetectaX Image Analyzer". Below this text is a sub-section titled "Project Objectives". At the bottom of the main content area are two buttons: "Classify an Image" and "Detect Objects".



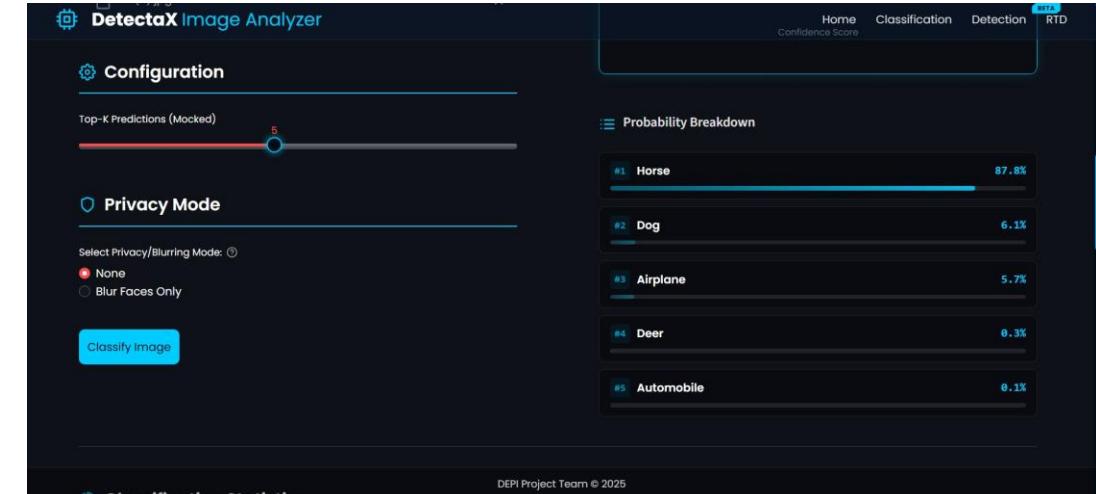
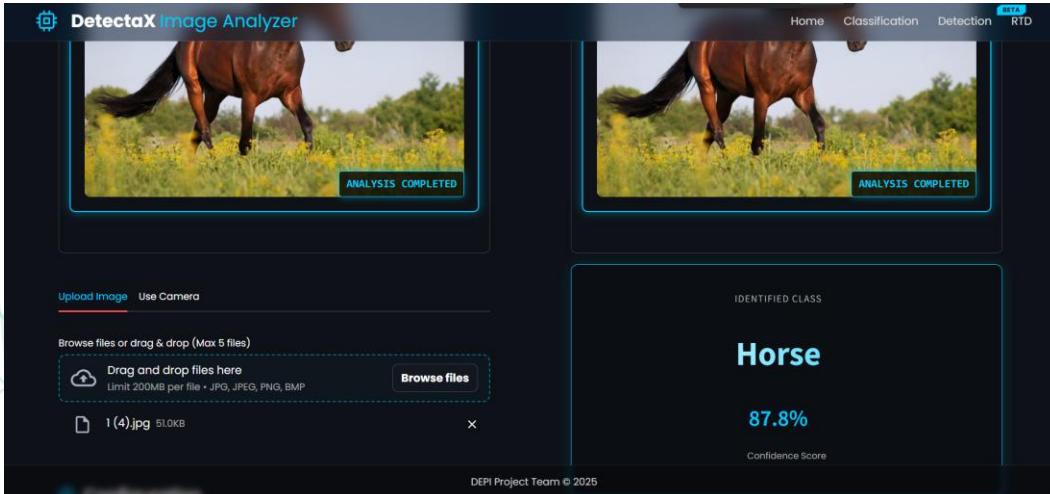
The screenshot shows the "Project Objectives" page of the DetectaX Image Analyzer. It features a title "Project Objectives" followed by a list of six items, each with an icon and a brief description. Below this is another title "Key Features" followed by three items. At the bottom right of the page, there is a small copyright notice: "DEPI Project Team © 2025".

- Collect and preprocess high-quality, diverse datasets.
- Develop and evaluate high-performance deep learning models (CNNs, YOLO).
- Enhance models via transfer learning and fine-tuning.
- Deploy models on Azure using containerized services and RESTful APIs.
- Implement MLOps practices for tracking, versioning, and monitoring.
- Deliver an intuitive web interface for real-time predictions.

- Image Classification:** Upload an image and our model will identify the primary subject.
- Object Detection:** Our YOLO-based model identifies and draws bounding boxes around multiple objects.
- Interactive Interface:** A smooth, responsive UI built with Streamlit for easy interaction.

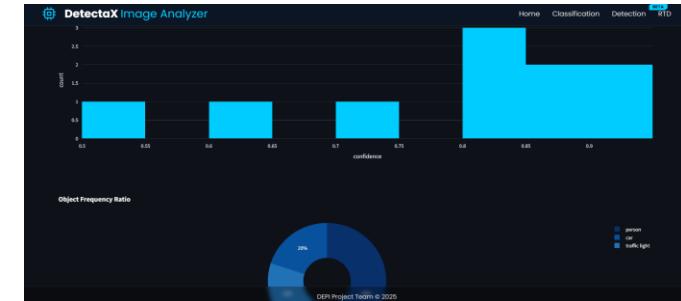
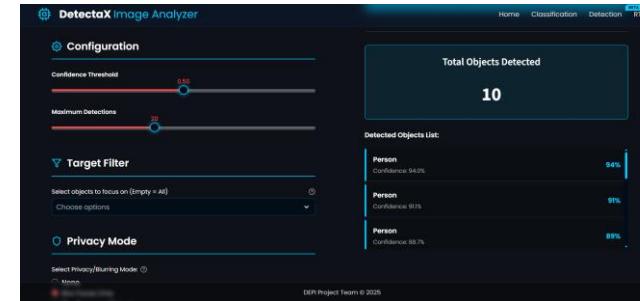
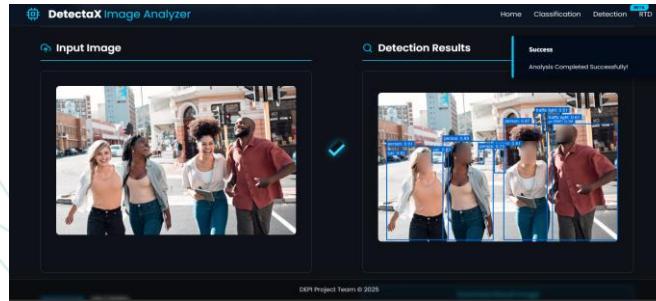
- Full interactive web interface built with **Streamlit**.
- Supports image upload, batch processing, camera streaming, and real-time detection.
- Unified dark **glassmorphism** theme with custom CSS.
- Smooth navigation and consistent UI design across all modules.

Image Classification Module



- Upload single or multiple images (**Batch Processing**).
- Automatic preprocessing and validation of all images.
- Sends images to Azure **CNN** classification endpoint.
- Displays predicted class, confidence score, and analytical charts.
- Session history maintained for all processed images.
- Stable fallback logic for handling API downtime.

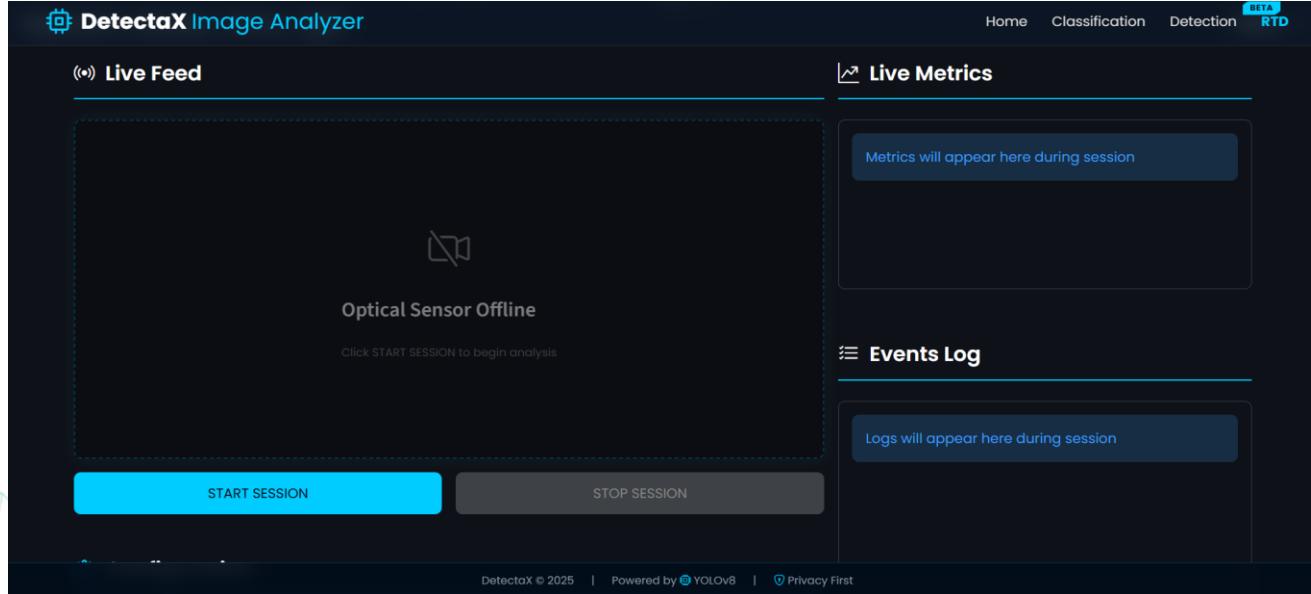
Object Detection Module



- Supports single-image and multi-image detection (**Batch Processing**).
- Performs detection using **YOLOv12x** via Azure cloud endpoint.
- Adjustable confidence threshold and class filtering.
- Custom bounding-box visualization with class-based colors.
- Object **cropping**, ZIP export, PDF reporting, and detection analytics.
- Clean, guided workflow for processing multiple images sequentially.



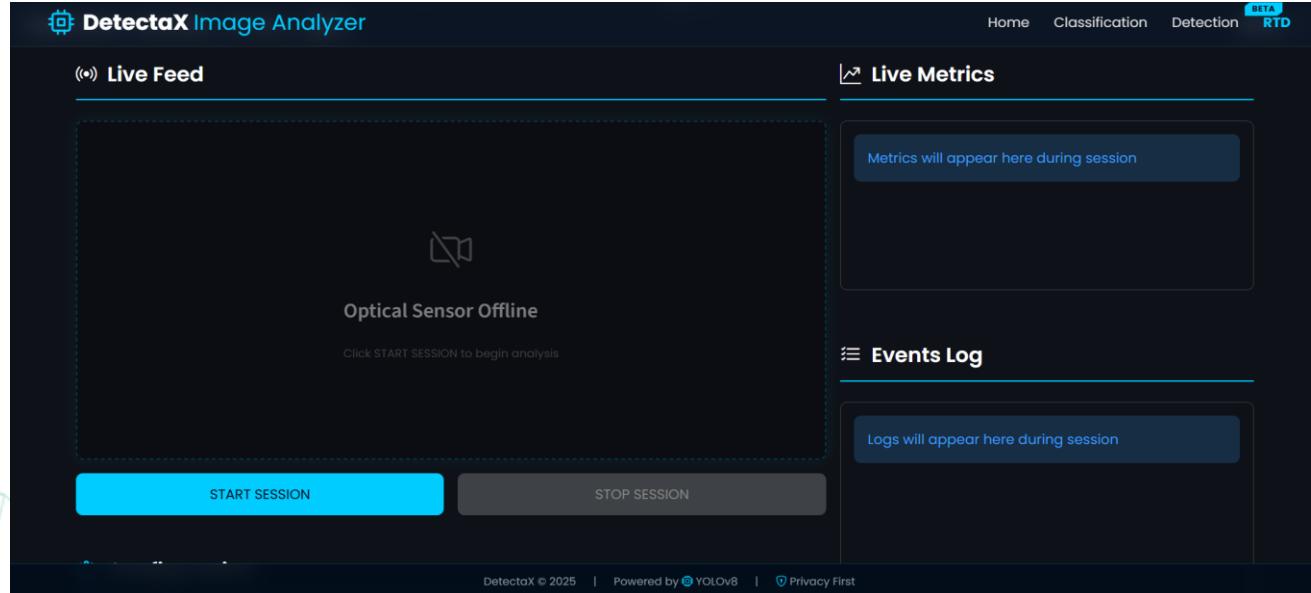
Real-Time Detection (RTD)



The screenshot shows the DetectaX Image Analyzer interface in RTD mode. The top navigation bar includes Home, Classification, Detection, and RTD (BETA). The main area is divided into two sections: 'Live Feed' on the left and 'Live Metrics' on the right. The 'Live Feed' section displays a placeholder message 'Optical Sensor Offline' with a speaker icon and a button to 'START SESSION'. The 'Live Metrics' and 'Events Log' sections both have placeholder messages: 'Metrics will appear here during session' and 'Logs will appear here during session' respectively.

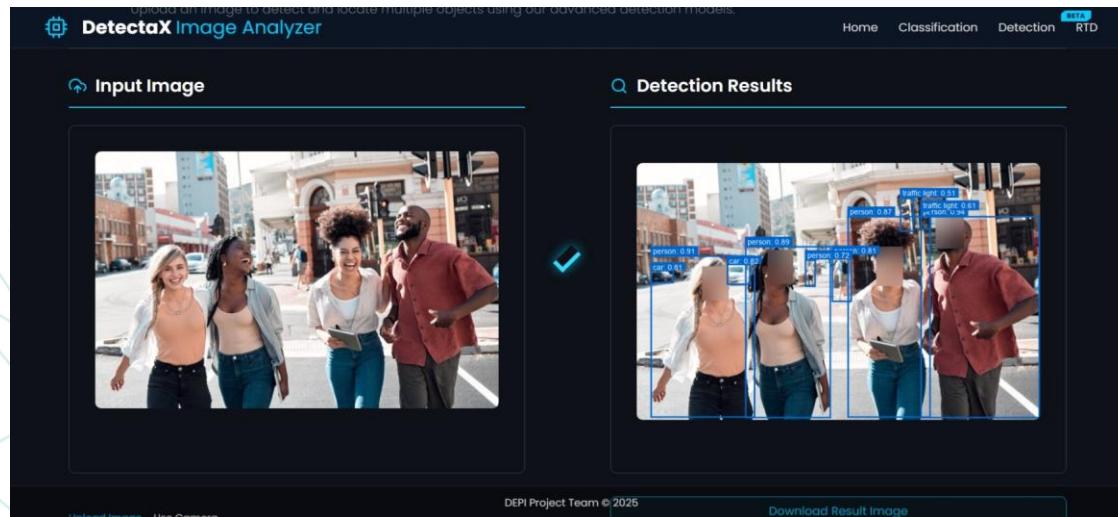
- Real-time camera feed processing using **OpenCV**.
- Continuous YOLOv12x inference per frame with **FPS** calculation.
- Auto-snapshot capturing during the session.
- Live statistics and dynamically updated charts.

Real-Time Detection (RTD)

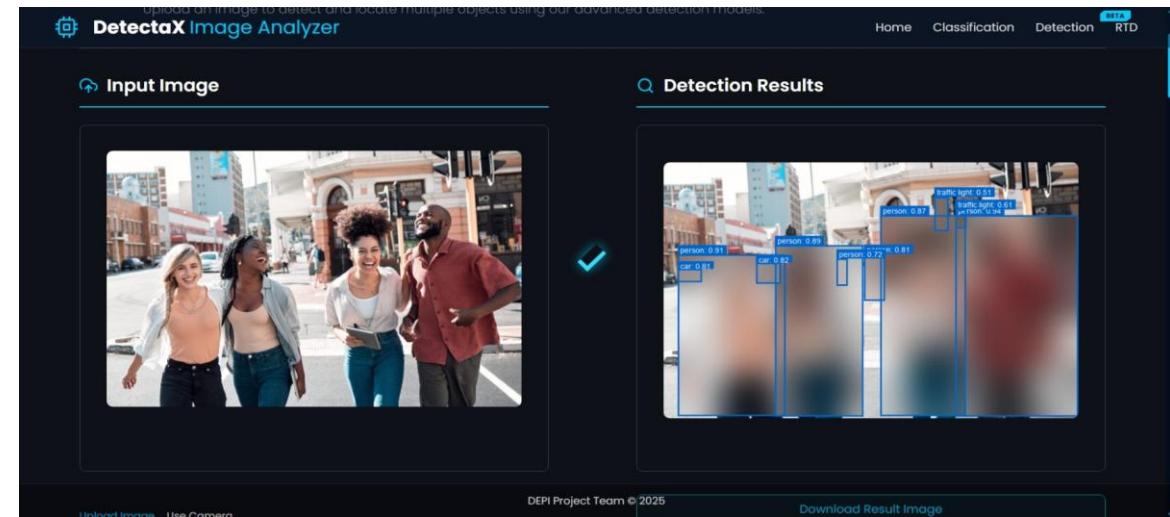


- Real-time camera feed processing using **OpenCV**.
- Continuous YOLOv12x inference per frame with **FPS** calculation.
- Auto-snapshot capturing during the session.
- Live statistics and dynamically updated charts.

Smart Privacy System



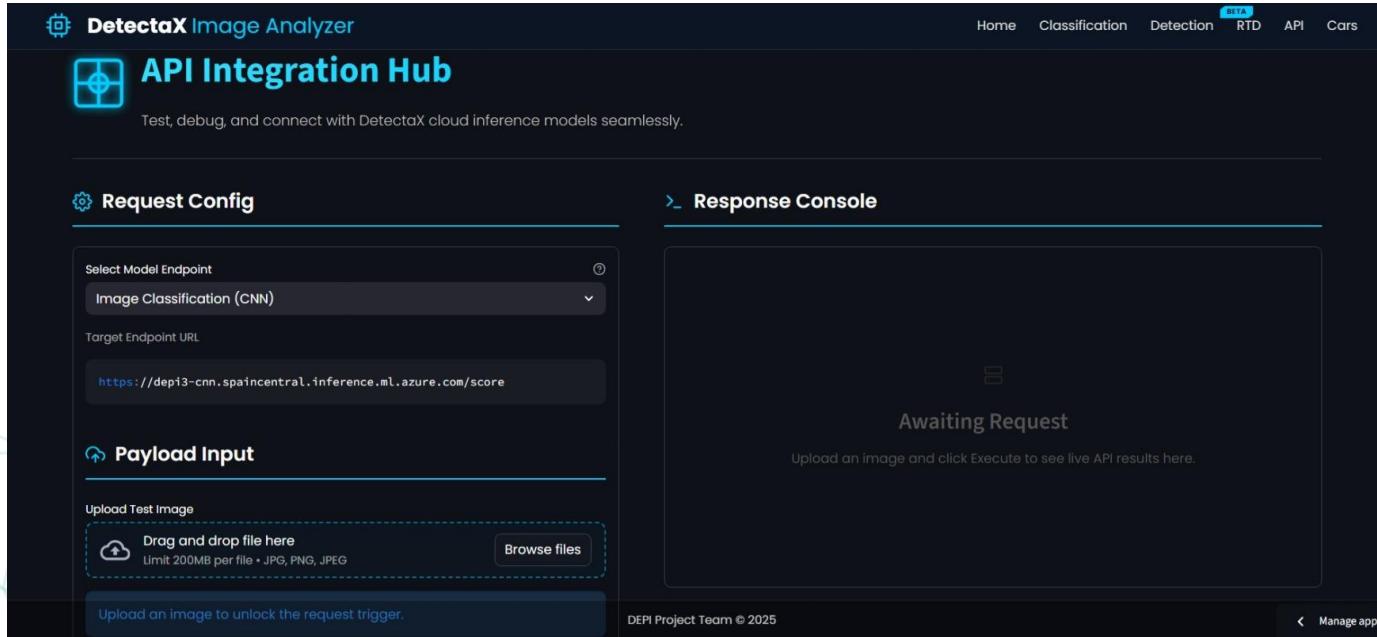
(Blur Faces Only)



(Blur Detected Objects)

- Optional privacy modes: **face blur** and full-person masking.
- Utilizes **YOLO-Face** for accurate face detection.
- Selective blurring without affecting full frame.
- Available in both static image detection and real-time mode.

API Integration:



- **Key Points** About the API Integration Hub Allows testing and debugging of the **DetectaX** cloud AI models.
 1. Supports **image upload for real-time inference**.
 2. Provides a **clean interface** to **send requests** and view API responses.
 3. Enables **selection of different model endpoints** (e.g., Image Classification CNN).
 4. **Helps developers** validate the model output before integrating it into applications.
 5. Displays the **live response** in the console after executing the request.

Project Demonstration

Thank You!