FSL Detection Prototype Documentation

Overview

This is a prototype desktop application for real-time hand sign detection using a YOLO model. The application provides a simple graphical user interface (GUI) built with Tkinter. It uses the webcam feed, applies the trained YOLO model, and displays annotated results live.

Features

- Real-time hand sign detection via webcam.
- User-friendly GUI built with Tkinter.
- Menu navigation with buttons for detection, settings, and information.
- Settings panel to adjust camera resolution and background color.
- About page describing the project.
- Background model loading to avoid freezing the interface.

Application Flow

1. **Launch Application**
    - Tkinter window initializes.
    - A loading screen appears while the YOLO model is loaded in a background thread.
2. **Menu Screen**
    - Options: Start Detection, Settings, About, Quit.
3. **Start Detection**
    - Activates webcam with selected resolution.
    - Runs YOLO model predictions on frames.
    - Displays annotated frames in real time.
    - Option to return to menu.
4. **Settings Panel**
    - Change camera width and height (predefined values).
    - Select background color (predefined color options).
    - Save changes and return to menu.
5. **About Page**
    - Displays project description, dataset size, and prototype status.

Key Components

Globals

- **video_label**: Tkinter label for displaying video frames.
- **running**: Boolean flag to control detection loop.
- **model**: YOLO model instance loaded from `best.pt`.
- **cap**: OpenCV video capture object.

Preferences (Tkinter Variables)

- **cam_width (IntVar)**: Camera frame width.
- **cam_height (IntVar)**: Camera frame height.
- **BG_COLOR (StringVar)**: Background color.
- **BTN_COLOR (StringVar)**: Button background color.
- **BTN_HOVER (StringVar)**: Button hover color.
- **TEXT_COLOR (StringVar)**: Button text color.
- **NOTE_COLOR (StringVar)**: Label note color.

Functions

- **on_enter / on_leave**: Handle button hover effects.
- **styled_button(master, text, command)**: Create styled buttons with hover bindings.
- **start_detection()**: Start video capture and detection loop.
- **update_frame()**: Capture webcam frame, run YOLO prediction, display annotated frame.
- **show_about()**: Display About screen with project details.
- **show_settings()**: Display Settings screen for adjusting preferences.
- **show_menu()**: Main menu screen with navigation options.
- **show_loading()**: Display loading screen while model loads.
- **load_model()**: Load YOLO model in a background thread.

Model Training

The YOLO model used in this app was trained to detect hand signs. Training steps:

1. **Dataset**
    - More than 13,000 labeled images of hand signs.
    - Images annotated in YOLO format (bounding boxes with class labels).
    - Dataset split into training, validation, and testing sets.
2. **Training Process**
    - Framework: Ultralytics YOLOv8.

- Command example:

  ```
  yolo detect train data=hand_signs.yaml model=yolov8n.pt
  epochs=100 imgsz=416 batch=16
  ```

- `hand_signs.yaml` contains dataset paths and class definitions.

- Model trained for 100 epochs with image size 416.

3. **Output**
   - Best weights saved as `best.pt`.
   - This file is loaded by the application for real-time inference.

Dependencies

- **Python 3.9+**
- **Tkinter**: GUI framework (comes with Python)
- **OpenCV (cv2)**: Webcam access and image processing
- **Ultralytics YOLO**: Object detection model
- **Pillow (PIL)**: Image handling for Tkinter display
- **Threading**: Background model loading

Requirements

The full environment snapshot used in development:

```
asttokens==3.0.0
certifi==2025.8.3
charset-normalizer==3.4.3
colorama==0.4.6
comm==0.2.3
contourpy==1.3.3
cycler==0.12.1
debugpy==1.8.16
decorator==5.2.1
executing==2.2.1
filelock==3.19.1
filetype==1.2.0
fonttools==4.59.2
fsspec==2025.9.0
idna==3.7
ipykernel==6.30.1
ipython==9.5.0
ipython_pygments_lexers==1.1.1
jedi==0.19.2
Jinja2==3.1.6
jupyter_client==8.6.3
jupyter_core==5.8.1
```

```
kiwisolver==1.4.9
MarkupSafe==3.0.2
matplotlib==3.10.6
matplotlib-inline==0.1.7
mpmath==1.3.0
nest-asyncio==1.6.0
networkx==3.5
numpy==2.2.6
opencv-python==4.12.0.88
opencv-python-headless==4.10.0.84
packaging==25.0
parso==0.8.5
pi_heif==1.1.0
pillow==11.3.0
pillow-avif-plugin==1.5.2
platformdirs==4.4.0
polars==1.33.0
prompt_toolkit==3.0.52
psutil==7.0.0
pure_eval==0.2.3
py-cpuinfo==9.0.0
Pygments==2.19.2
pyparsing==3.2.3
python-dateutil==2.9.0.post0
python-dotenv==1.1.1
pywin32==311
PyYAML==6.0.2
pyzmq==27.0.2
requests==2.32.5
requests-toolbelt==1.0.0
roboflow==1.2.7
scipy==1.16.1
setuptools==80.9.0
six==1.17.0
stack-data==0.6.3
sympy==1.14.0
torch==2.8.0
torchvision==0.23.0
tornado==6.5.2
tqdm==4.67.1
traitlets==5.14.3
typing_extensions==4.15.0
ultralytics==8.3.191
ultralytics-thop==2.0.16
urllib3==2.5.0
wcwidth==0.2.13
```

Minimal requirements for running the app:

```
ultralytics==8.3.191
opencv-python==4.12.0.88
pillow==11.3.0
numpy==2.2.6
torch==2.8.0
torchvision==0.23.0
```

Optional (for dataset management and training):

```
roboflow==1.2.7
```

Usage

1. **Create and activate a virtual environment**

   ```
   python -m venv venv
   # Windows
   venv\\Scripts\\activate
   # Linux / Mac
   source venv/bin/activate
   ```

2. **Install dependencies**

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

3. **Download the model file**

   - The trained YOLO model `best.pt` is available here: Google Drive Link
   - Place `best.pt` in the project directory.

4. **Run the application**

   ```
   python main.py
   ```

5. **Use the menu to:**

   - Start detection.
   - Configure settings.
   - View about info.
   - Quit the app.

6. **Adjust settings**

   - Change camera resolution and background color before starting detection.

Notes

- Always activate the virtual environment before running the app.
- Performance depends on device specifications.
- This application is in prototype stage.
- Default YOLO model file: `best.pt` must be available in the project directory.

Demo Video

Watch a sample demo of the application here:
https://drive.google.com/file/d/1vW9_DRKFWgcMaozMfNGqQQumto_BxfBt/view?usp=drive_link

Datasets

The hand sign dataset used to train this model contains over 13,000 annotated images. You can access the dataset here:
https://drive.google.com/file/d/1jAwcFYX-oHFQ46cYXTO02bPSaB3L2UeV/view?usp=sharing