Election Watch AI

Computer Vision System for Electoral Integrity

DS-FT12 - Group 4: Bernice, Lilian, Vanessa, Eric and Tim (17/07/2025)

Project Overview

- A real-time computer vision system to monitor ballot boxes
- Detects potential electoral fraud and irregularities
- Prioritises **voter privacy** while maintaining **transparency**
- Uses **deep learning models** and **synthetic datasets**

System Tasks

Task	Model	Purpose
Ballot Drop Detection	YOLOv8	Detect Ballot Drops
Tampering Detection	CNN + LSTM	Detect suspicious box activities
Voter Repetition Detection	YOLOv8 + DeepSort	Track re-entry of individuals
Voter Spike Detection	LSTM	Detect abnormal ballot drop rates
Face Blurring	MTCNN + OpenCV	Blur faces to protect voter privacy

Ballot Drop Detection

- **Model**: YOLOv8
- **Dataset**:
  - Leap Hand Gesture Dataset
  - Synthetic Ballot Dataset
- **Goal**:
  - Detect when hands drop ballots
  - Count valid ballot submissions

Tampering Detection

- **Model**: CNN + LSTM
- **Dataset:** Synthetic Ballot + UCF Crime
- **Detects**:
  - Ballot box shaking
  - Unauthorized access/opening
  - Ballot stuffing
  - Other unusual activity

Voter Re-entry Detection

- **Models**: YOLOv8 + DeepSort
- **Dataset**: Synthetic Ballot Dataset
- **Goal**:
  - Track and identify individuals
  - Detect multiple entries by the same person
  - Use outfit changes for re-ID scenarios

Voter Spike Pattern Detection

- **Model**: LSTM (Anomaly Detection)
- **Dataset**: Generated CSV event logs
- **Goal**:
  - Analyze drop-rate over time
  - Flag suspicious spikes (e.g., ballot stuffing attempts)

Face Blurring for Privacy

- **Model**: MTCNN + OpenCV Gaussian Blur
- **Dataset**: LFW Face Dataset
- **Goal**:
  - Detect & blur voter faces
  - Ensure anonymity & privacy
  - Maintain transparency in footage

Technical Stack

- **Languages**: Python 3.8+
- **GPU Recommended**: NVIDIA + CUDA/cuDNN
- **Libraries**:
  - OpenCV
  - PyTorch/TensorFlow
  - YOLOv8 (Ultralytics)
  - DeepSort, LSTM, MTCNN

System Architecture

Flow:

1. Video Input
2. Ballot Drop Detection
3. Tampering + Voter Tracking
4. Anomaly Detection
5. Face Blurring
6. Processed Video Output

Next Steps

- Conduct full model evaluation in Jupyter Notebook
- Train on real-world data (with permissions)
- Improve accuracy in low-light and occlusion scenarios
- Partner with election bodies for pilot testing

Further Steps

- Improved Data Labeling: Use semi-supervised learning or anomaly detection to handle unlabeled or imbalanced data.
- Behavioral Analysis: Incorporate voter flow patterns and time-based voting trends to detect unusual activity.
- Edge Deployment: Deploy the model to Raspberry Pi or embedded devices for real-time tampering alerts without reliance on cloud infrastructure.
- Security Enhancements: Integrate with blockchain or secure logging systems for traceability of predictions.
- AutoML Techniques: Experiment with automated model tuning (e.g., Keras Tuner or Optuna) for hyperparameter optimization.

Questions?

Contact

Project Lead: Bernice Wakarindi

📧 bernicewakarindi@gmail.com

🔗 GitHub: Election_Watch_AI