CHAIN SNATCHING DETECTION USING DEEP LEARNING

---

1. INTRODUCTION

CHAIN SNATCHING IS A COMMON STREET CRIME IN URBAN AREAS WHERE OFFENDERS SUDDENLY SNATCH CHAINS OR VALUABLES FROM PEDESTRIANS AND ESCAPE QUICKLY. DUE TO THE SUDDEN NATURE OF THE CRIME, MANUAL SURVEILLANCE USING CCTV CAMERAS BECOMES INEFFECTIVE AND REQUIRES CONTINUOUS HUMAN MONITORING.

WITH THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE AND COMPUTER VISION, IT IS POSSIBLE TO AUTOMATE THE DETECTION OF SUCH CRIMINAL ACTIVITIES. THIS PROJECT PROPOSES A DEEP LEARNING–BASED SYSTEM THAT DETECTS CHAIN SNATCHING EVENTS FROM VIDEO STREAMS.

THE SYSTEM COMBINES:

- A CONVOLUTIONAL NEURAL NETWORK (CNN) FOR DETECTING CHAIN SNATCHING EVENTS
- A YOLOV8 OBJECT DETECTION MODEL FOR DETECTING AND VISUALIZING HUMANS IN THE SCENE

THE MAIN OBJECTIVE IS TO ACCURATELY IDENTIFY WHETHER A CHAIN SNATCHING EVENT HAS OCCURRED WHILE PROVIDING VISUAL INTERPRETATION USING BOUNDING BOXES AND IDS.

---

2. PROBLEM STATEMENT

EXISTING CCTV SURVEILLANCE SYSTEMS CAN RECORD EVENTS BUT CANNOT AUTOMATICALLY UNDERSTAND OR DETECT CRIMINAL ACTIVITIES SUCH AS CHAIN SNATCHING. OBJECT DETECTION MODELS CAN IDENTIFY PEOPLE BUT CANNOT RECOGNIZE INTERACTIONS, WHILE CLASSIFICATION MODELS CAN DETECT EVENTS BUT LACK LOCALIZATION CAPABILITY.

THE PROBLEM IS TO:

- DETECT CHAIN SNATCHING EVENTS ACCURATELY
- MAINTAIN HIGH ACCURACY DURING MODEL INTEGRATION
- PROVIDE CLEAR VISUALIZATION
- AVOID FALSE POSITIVES
- DESIGN A SCALABLE AND EXPLAINABLE SYSTEM

---

3. OBJECTIVES

- DETECT CHAIN SNATCHING EVENTS FROM VIDEO DATA
- USE DEEP LEARNING FOR SCENE-LEVEL EVENT DETECTION
- DETECT AND VISUALIZE ALL HUMANS USING BOUNDING BOXES

- Track humans using unique IDs
- Ensure high accuracy and robustness
- Build a system suitable for real-world surveillance

---

4. Dataset Preparation

4.1 Chain Snatching Dataset (CNN)

- Frames extracted from videos containing chain snatching and normal activities
- Images classified into:
  - Snatch
  - Not-Snatch
- Dataset represents scene-level context
- Used for training a binary classification CNN model

4.2 Human Detection Dataset (YOLO)

- Dataset consists of images containing humans
- Bounding boxes annotated using YOLO format
- Annotated using CVAT / LabelImg
- YOLOv8 trained to detect only humans

Both datasets were prepared independently to avoid domain mismatch.

---

5. Model Architecture

5.1 CNN for Chain Snatching Detection

- Input size: 128 × 128
- Convolutional layers for feature extraction
- ReLU activation functions
- Max pooling layers
- Fully connected layers for classification
- Output: Probability of chain snatching

Loss Function:

- Binary Cross-Entropy

Optimizer:

- Adam Optimizer

5.2 YOLOv8 for Human Detection

- Model: YOLOv8
- Detects only human class
- Produces bounding boxes
- Integrated with ByteTrack for assigning unique IDs
- Used only for visualization, not event decision

---

6. Integration Strategy

Initial Approach

Initially, YOLO was tightly integrated with CNN by cropping detected humans and passing them to the CNN. This caused poor accuracy due to domain mismatch.

Final Approach – Loose Coupling

Model	Function
CNN	Event detection (chain snatching)
YOLO	Human visualization with bounding boxes and IDs

The CNN and YOLO models work independently to preserve accuracy.

---

7. System Workflow

1. Video frames are read using OpenCV
2. Each full frame is passed to CNN
3. CNN predicts chain snatching probability
4. Temporal smoothing is applied
5. YOLO detects all humans
6. Bounding boxes and IDs are drawn
7. Final result is displayed

---

8. Temporal Event Detection

To improve robustness:

- Sliding window smoothing is used
- Chain snatching is confirmed only after consecutive frames exceed a threshold

This reduces false positives and improves stability.

---

9. Tools and Technologies Used

9.1 AI & Deep Learning Tools

- PyTorch – CNN training and inference
- Ultralytics YOLOv8 – Human detection
- ByteTrack – Human tracking and ID assignment

9.2 Computer Vision Tools

- OpenCV – Video processing
- Torchvision – Image transformations

9.3 Dataset & Annotation Tools

- CVAT – Bounding box annotation
- LabelImg – YOLO annotation format

9.4 Development Tools

- Python 3.12
- VS Code
- GitHub (optional)

---

10. System Requirements

10.1 Software Requirements

- Windows / Linux OS
- Python 3.8 or above
- PyTorch
- Ultralytics YOLOv8
- OpenCV

- CUDA (for GPU acceleration)

10.2 Hardware Requirements

- NVIDIA GPU (RTX series recommended)
- Minimum 8 GB RAM
- Webcam / CCTV footage
- Storage for datasets and videos

---

11. Results

- CNN-alone testing accurately detects chain snatching events
- Integrated system maintains high accuracy
- Bounding boxes with IDs improve interpretability
- Works effectively on real-world videos

---

12. Advantages of the System

- Automated crime detection
- Reduces manual monitoring
- Real-time capability
- Scalable architecture
- Clear visualization

---

13. Limitations

- Performance depends on video quality
- Extreme occlusion may affect detection
- Requires sufficient training data

---

14. Conclusion

This project successfully demonstrates an AI-based approach for detecting chain snatching events using deep learning. By decoupling event detection and object detection, the system achieves high accuracy, robustness, and scalability. The system can be extended to other surveillance-based crime detection tasks.

15. Future Enhancements

- CNN + LSTM for temporal learning
- Integration with alert systems
- Real-time deployment on CCTV networks
- Crime analytics dashboards
- Multi-camera tracking

---

16. Final Summary

The proposed system uses a CNN for scene-level chain snatching detection and YOLOv8 for human visualization. Loose coupling ensures reliable performance, making the system suitable for surveillance and academic applications.