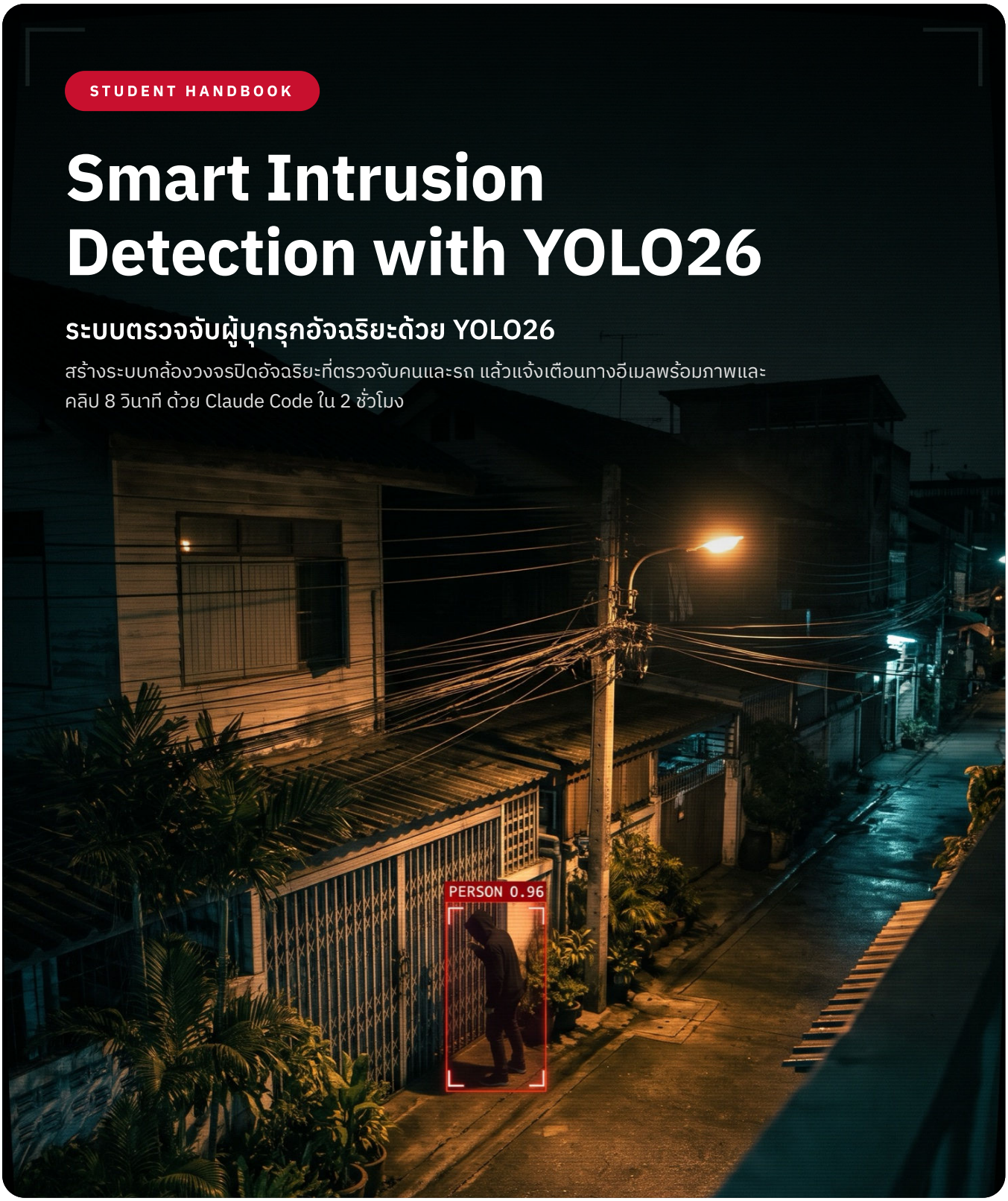
STUDENT HANDBOOK

Smart Intrusion Detection with YOLO26

ระบบตรวจจับผู้บุกรุกอัจฉริยะด้วย YOLO26

สร้างระบบกล้องวงจรปิดอัจฉริยะที่ตรวจจับคนและรถ แล้วแจ้งเตือนทางอีเมลพร้อมภาพและคลิป 8 วินาที ด้วย Claude Code ใน 2 ชั่วโมง



PHASE 1-2
ตรวจจับ
คน รถ มอเตอร์ไซค์ ด้วย YOLO26

PHASE 3-4
เฝ้าระวัง
เลือกช่วงเวลา โซน และคลาส

PHASE 5
เก็บหลักฐาน
ภาพ + คลิป 8 วินาที

PHASE 5
แจ้งเตือน
อีเมลพร้อมภาพถึงมือถือ

OVERVIEW

ภาพรวมเวิร์กช็อป

วันนี้เราจะสร้าง **ระบบกล้องวงจรปิดอัจฉริยะ** ที่มองออกว่าในภาพมี "คน" หรือ "รถ" เข้ามาในโซนที่ห้ามเข้า ในช่วงเวลาที่ไม่ควรมีใคร แล้วส่งอีเมลแจ้งเตือนพร้อมภาพและคลิปหลักฐานทันที ทั้งหมดทำด้วยการพิมพ์สั่ง **Claude Code** เป็นภาษาไทย ไม่ต้องเขียนโค้ดเอง

01	02	03	04
วาง Prompt	**AI เขียนโค้ด**	**รันดูผลทันที**	**ไปเฟสถัดไป**
คัดลอกจากคู่มือไปวางใน Claude Code	Claude Code เขียนและติดตั้งให้	เห็นกรอบตรวจจับบนวิดีโอจริง	เพิ่มความสามารถทีละขั้น จนครบระบบ

วิธีใช้คู่มือเล่มนี้

PROMPT

ตัวอย่างกล่องคำสั่ง

ข้อความในกล่องนี้คือคำสั่งสำเร็จรูป คัดลอกไปวางใน `Claude Code` ได้ทันที
ส่วนที่เป็น **[ตัวสีแดงในวงเล็บ]** ให้เปลี่ยนเป็นข้อมูลของตัวเอง

ปุ่มคัดลอก
ในคู่มือฉบับเว็บ ทุกกล่องมีปุ่ม "คัดลอก Prompt" มุมขวาบน

สิ่งที่ควรเห็น
ภาพผลลัพธ์จริงจากระบบไว้เทียบกับหน้าจอของเรา

เคล็ดลับ
กล่องแดงอ่อน คือจุดที่มักพลาดหรือทางลัดที่ช่วยได้

ดูภาพเต็มจอ
ฉบับเว็บ แตะที่ภาพเพื่อขยายเต็มจอได้ทุกภาพ

LEARNING OUTCOMES

ผลลัพธ์หลังเรียน

- อธิบายได้ว่า **AI Object Detection** คืออะไร และ **YOLO26** ทำงานอย่างไร
- ใช้ **Claude Code** สร้างระบบตรวจจับด้วย Python + OpenCV + YOLO26 ได้เองทีละเฟส
- **เลือกขนาดโมเดล** YOLO26 ให้เหมาะกับเครื่องและงานได้
- กำหนด **ช่วงเวลา โซน และคลาส** ที่ต้องการเฝ้าระวังได้
- ได้ระบบที่ **ส่งอีเมลแจ้งเตือนพร้อมภาพ + คลิป 8 วินาที** (ก่อนเหตุ 3 วิ + หลังเหตุ 5 วิ) ผ่าน Gmail App Password

TIMETABLE

ตารางสอน 120 นาที

เวลา	หัวข้อ	ทำอะไร
0:00–0:10	**เปิดคลาส**	แนะนำผู้สอน · ดูผลลัพธ์ปลายทาง (อีเมลแจ้งเตือนจริง) · เป้าหมายวันนี้
0:10–0:25	**AI Detection & YOLO26**	Detection คืออะไร · YOLO26 คืออะไร ทำไมต้องใช้ ทำงานอย่างไร · ช่วยกันขโมยอย่างไร
0:25–0:35	**เครื่องมือ + Setup**	Claude Code · Python · OpenCV · Ultralytics/YOLO26 · ffmpeg · Gmail — Prompt S0
0:35–0:50	**Phase 1 ตรวจจับคน**	Prompt P1 · ไฟล์ / webcam / RTSP
0:50–1:00	**Phase 2 เลือกขนาดโมเดล**	Prompt P2 (+P2B seg/pose) · เทียบ n / s / m
1:00–1:15	**Phase 3 ช่วงเวลา + โซน**	Prompt P3 · 22:00–06:00 · วาดโซน
1:15–1:25	**Phase 4 กรองคลาส**	Prompt P4 · 80 คลาส · กันแจ้งเตือนหลอก
1:25–1:50	**Phase 5 หลักฐาน + อีเมล**	Prompt P5A + สร้าง App Password + P5B
1:50–2:00	**รันระบบเต็ม + ต่อยอด**	Prompt RUN · Bonus RTSP / แผงควบคุมเว็บ · Q&A

> **หมายเหตุถึงผู้เรียน**
> ตารางเวลาเป็นแนวทาง ผู้สอนอาจปรับเวลาแต่ละช่วงให้เหมาะกับความคืบหน้าของผู้เรียน โดยเนื้อหาและผลลัพธ์หลักยังครบถ้วน

BEFORE CLASS

ก่อนเริ่ม — สิ่งที่ต้องเตรียม

เตรียม 6 อย่างนี้ให้พร้อมก่อนเข้าคลาส จะเริ่มลงมือได้ตั้งแต่ Prompt แรก

- ✓ **โน้ตบุ๊ก Mac หรือ Windows**
  RAM 8 GB ขึ้นไป · 1 คน 1 เครื่อง
- ✓ **ติดตั้ง Claude Code และล็อกอินแล้ว**
  แพ็กเกจ Pro ขึ้นไป · ลองพิมพ์สั่งงานได้ก่อนวันเรียน
- ✓ **Python 3.10 ขึ้นไป**
  ติดตั้งไว้ก่อน ไลบรารีที่เหลือ AI ติดตั้งให้ใน Setup
- ✓ **บัญชี Gmail ที่เปิด 2-Step Verification**
  ใช้สร้าง App Password สำหรับส่งอีเมลแจ้งเตือน (Phase 5)
- ✓ **คลิปทดสอบ**
  ดาวน์โหลดจากกล่องด้านล่าง แล้ววางในโฟลเดอร์โปรเจกต์
- ✓ **อินเทอร์เน็ต**
  ครั้งแรกระบบต้องดาวน์โหลดไลบรารีและไฟล์โมเดล

> **ไม่ต้องมี GPU** — YOLO26n รันบน CPU ได้ · **ไม่ต้องมีกล้องจริง** — ในคลาสใช้คลิปวิดีโอ · **ไม่ต้องเขียนโค้ดเป็น** — Claude Code เขียนให้

สำคัญ ดาวน์โหลดคลิปทดสอบ

คลิปกล้องวงจรปิด 7 คลิปที่ใช้ทดสอบทุกเฟส (ทั้งกลางวัน กลางคืน คน รถยนต์ มอเตอร์ไซค์)

ดาวน์โหลดคลิปทดสอบ

yolo26-test-videos.zip · 7 คลิป

แตกไฟล์ไว้ในโฟลเดอร์โปรเจกต์ ให้ได้โฟลเดอร์ชื่อ `videos/`

ดาวน์โหลด .zip

1. **สร้างโฟลเดอร์ว่างสำหรับโปรเจกต์**
   ตั้งชื่อเป็นภาษาอังกฤษ ไม่มีเว้นวรรค เช่น `yolo26-detection`

2. **ดาวน์โหลด .zip แล้วแตกไฟล์ไว้ในโฟลเดอร์นั้น**
   ให้ได้ `yolo26-detection/videos/` ที่มีไฟล์ .mp4 อยู่ข้างใน

3. **เปิดโฟลเดอร์นี้ใน Claude Code**
   ทุก Prompt ในคู่มือวางในโฟลเดอร์นี้ เริ่มจาก Prompt S0

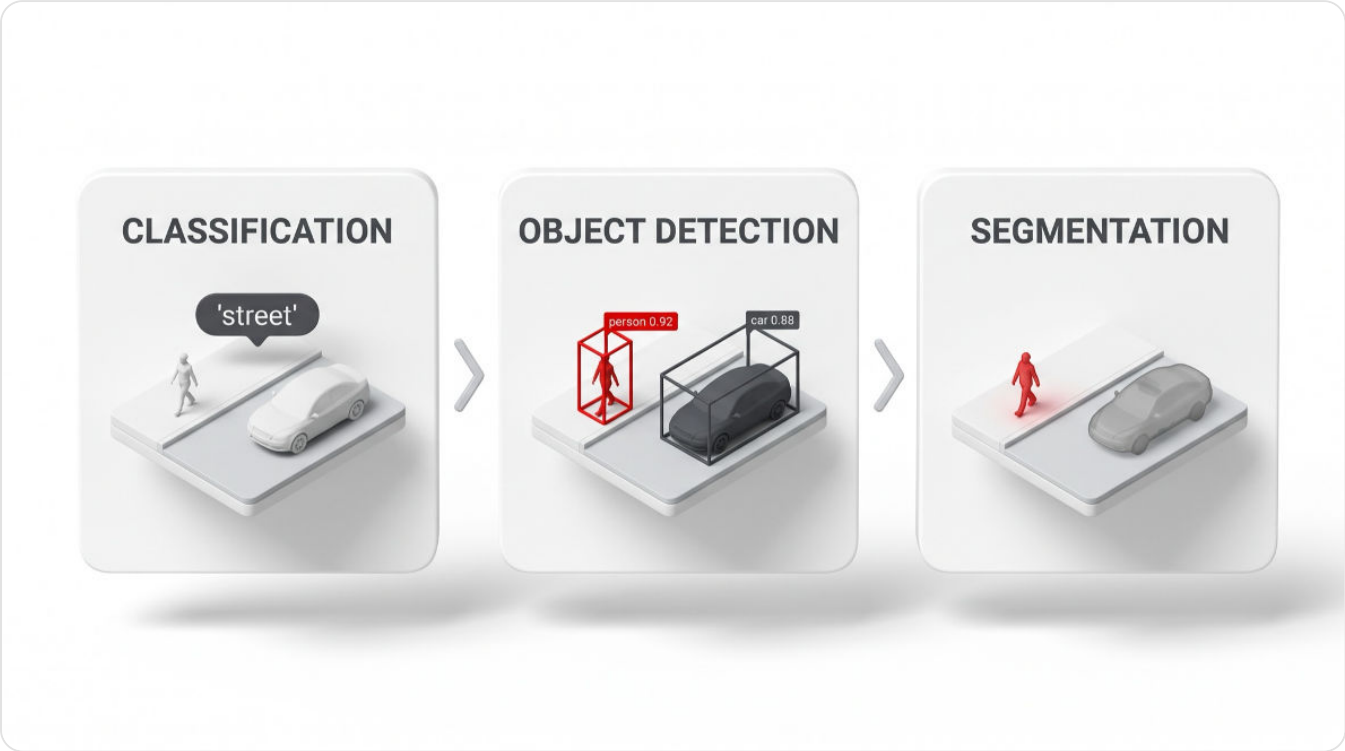
คลิปในชุดนี้ใช้ตอนไหน

ไฟล์ในโฟลเดอร์ videos/	ภาพในคลิป	ใช้ใน
`01_door_entry.mp4`	คนเดินเข้าประตู	**P1**
`02_hallway_door.mp4`	คนเดินในทางเดิน เห็นเต็มตัว	**P2B**
`04_office_wide.mp4`	ออฟฟิศมุมกว้าง	**P2**
`11_cctv_office_day.mp4`	กล้องวงจรปิดในออฟฟิศ กลางวัน	**P5A**
`12_cctv_office_night_ir.mp4`	กล้องวงจรปิดกลางคืน (อินฟราเรด)	**RUN**
`13_cctv_carport_motorcycle.mp4`	โรงจอดรถหน้าบ้าน มีมอเตอร์ไซค์	**P3 · P4**
`14_cctv_parking_car.mp4`	ลานจอดรถ มีรถยนต์เข้ามา	**P4**

CONCEPT · 0:10–0:25

AI Detection คืออะไร

Computer Vision คือการให้คอมพิวเตอร์ "เข้าใจภาพ" แบ่งได้ 3 ระดับ — ระบบกันขโมยของเราใช้ระดับกลาง คือ **Object Detection**



Computer Vision 3 ระดับ — จากบอกว่าภาพนี้คืออะไร ไปจนถึงรูปร่างของวัตถุทีละพิกเซล

ระดับ 1

Classification

บอกว่า "ภาพนี้คืออะไร" ได้คำตอบเดียวทั้งภาพ

ระดับ 2 · ใช้วันนี้

Object Detection

บอกว่า "มีอะไร อยู่ตรงไหน กี่ชิ้น" ตีกรอบให้ทุกวัตถุ

ระดับ 3

Segmentation

บอก "รูปร่าง" ของวัตถุละเอียดทีละพิกเซล

ผลลัพธ์ต่อ 1 วัตถุ

กรอบ `(x, y, w, h)` + คลาส `person` + ความมั่นใจ `confidence 0.87`

วิดีโอ = ภาพนิ่งหลายภาพ

วิดีโอคือภาพนิ่งหลายสิบภาพต่อวินาที AI จึงตรวจได้ทุกเฟรมแบบ real-time

YOLO26 คืออะไร

- **YOLO = You Only Look Once** — มองภาพครั้งเดียว ทายทุกวัตถุพร้อมกัน จึงเร็วพอสำหรับวิดีโอสด
- **YOLO26** = รุ่นล่าสุดของ Ultralytics (ออก มกราคม 2026) ออกแบบมาให้เร็วบนเครื่องทั่วไปและอุปกรณ์ edge
- ฝึกมากับชุดข้อมูล **COCO 80 คลาส**: คน รถยนต์ มอเตอร์ไซค์ รถบรรทุก จักรยาน กระเป๋า สุนัข ฯลฯ — **ใช้ได้ทันทีไม่ต้องเทรน**
- ทำได้หลายงาน: `Detection` `Segmentation` `Pose` `Classification` `OBB (กรอบเอียง)`

YOLO26 มี 5 ขนาด — ใหญ่ขึ้นแม่นขึ้น แต่ช้าลง

รุ่น	ความแม่นยำ mAP50-95	ความเร็ว CPU ONNX (ms/ภาพ)	ขนาด Params (M)
YOLO26n · ใช้ในคลาส	40.9	38.9	2.4
YOLO26s	48.6	87.2	9.5
YOLO26m	53.1	220.0	20.4
YOLO26l	55.0	286.2	24.8
YOLO26x	57.5	525.8	55.7

ทดสอบบน COCO ขนาดภาพ 640 px · ข้อมูลจาก Ultralytics · ms ยิ่งน้อยยิ่งเร็ว · mAP ยิ่งสูงยิ่งแม่น

ทำไมต้อง YOLO26



จุดเด่นของ YOLO26 — ไม่ต้อง NMS · เร็วบน CPU · จับวัตถุเล็ก · นำไปใช้ได้ทุกอุปกรณ์

End-to-end ไม่ต้อง NMS
ทายกรอบสุดท้ายได้เลย ไม่ต้องมีขั้นกรองกรอบซ้ำ ขั้นตอนน้อยลง เร็วขึ้น นำไปใช้ง่าย

เร็วบน CPU
งานวิจัยของผู้พัฒนารายงานว่า YOLO26n เร็วกว่า YOLO11n บน CPU สูงสุดราว 43%

เบา นำไปใช้ได้ทุกที่
ตัด DFL ออก หัวทำนายเรียบง่าย export ไป ONNX / TensorRT / CoreML / OpenVINO ได้ง่าย

จับวัตถุเล็กดีขึ้น (STAL)
เหมาะกับภาพกล้องวงจรปิดมุมสูงที่คนในภาพดูตัวเล็ก

ใช้ง่ายแค่ 3 บรรทัด

```
from ultralytics import YOLO
model = YOLO("yolo26n.pt")
model(frame)
```

ไม่ต้องจำโค้ดนี้ — Claude Code เขียนให้ แค่รู้ว่า "สมอง" ของระบบเรียกใช้ง่ายแค่นี้

YOLO26 ทำงานอย่างไร



ภาพเดียวจบ — ภาพเข้า ผ่าน 3 ส่วนของโมเดล แล้วได้กรอบ คลาส และความมั่นใจกันที

01 **ภาพเข้า** เฟรมจากกล้องหรือวิดีโอ	02 **Backbone** ดึงลักษณะเด่น ขอบ รูปร่าง พื้นผิว	03 **Neck** รวมลักษณะหลายขนาด ของใหญ่และของเล็ก	04 **Head** ทายกรอบ + คลาส + confidence	05 **ผลลัพธ์** ได้ทันที ไม่ต้อง NMS

ช่วยกันขโมยอย่างไร



ปัญหาเดิม — กล้องบันทึกตลอด แต่ไม่มีใครดูทันเวลา



AI Detection — แจ้งเฉพาะคน/รถในโซน ในเวลาที่ไม่ควรมีใคร

กล้องวงจรปิดทั่วไป
"บันทึก" แต่ไม่มีใครดู รู้ตัวก็เมื่อเสียหายไปแล้ว

Motion Detection
แจ้งเตือนทุกอย่างที่ขยับ (ใบไม้ เงา แมว) เตือนหลอกจนคนเลิกสนใจ

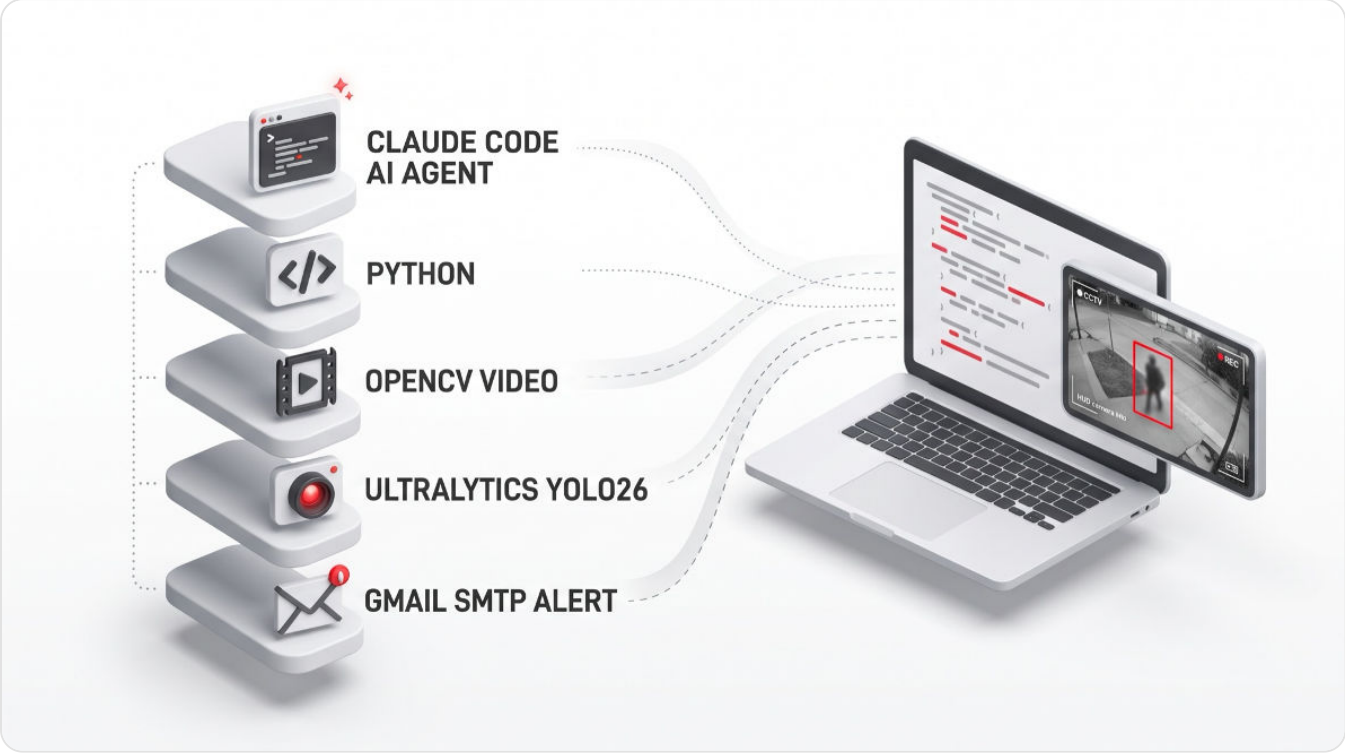
AI Detection
รู้ว่า "อะไร" ขยับ แจ้งเฉพาะคน/รถ ในโซนที่สนใจ ในเวลาที่ไม่ควรมีใคร

ส่งหลักฐานถึงมือถือทันที
ภาพ + คลิป ก่อนเหตุ 3 วินาที และหลังเหตุ 5 วินาที · ใช้ได้กับ บ้านกลางคืน · ร้านค้าหลังปิด · โกดัง · ลานจอดรถ · ห้อง server · แปลงเกษตร

TOOLS · 0:25–0:35

เครื่องมือวันนี้

ทุกเครื่องมือฟรีหรือใช้บัญชีที่มีอยู่แล้ว — Claude Code เป็นคนติดตั้งและต่อทุกอย่างเข้าด้วยกันให้



เครื่องมือ 5 ชั้นที่ประกอบเป็นระบบตรวจจับผู้บุกรุก

เครื่องมือ	บทบาท	ทำไมใช้
Claude Code	AI Agent เขียนโค้ด รันโค้ด แก้ error ให้ในโฟลเดอร์โปรเจกต์	สั่งเป็นภาษาไทย ไม่ต้องเขียนโค้ดเอง ได้ระบบจริงใน 2 ชม.
Python 3.10+	ภาษาหลักของ AI	ไลบรารี AI ครบที่สุด
OpenCV	อ่านวิดีโอ/กล้อง วาดกรอบ บันทึกคลิป	มาตรฐานงานภาพ ต่อ webcam/RTSP ได้
Ultralytics + YOLO26	"สมอง" ตรวจจับวัตถุ	โมเดลล่าสุด เร็ว ใช้ 3 บรรทัด
imageio-ffmpeg	แปลงคลิปเป็น H.264	เปิดได้ทุกเครื่องและมือถือ ไม่ต้องลง ffmpeg เอง
Gmail SMTP + App Password	ส่งอีเมลแจ้งเตือน	ฟรี ใช้บัญชีที่มีอยู่ ปลอดภัยกว่าใช้รหัสจริง
python-dotenv (.env)	เก็บรหัสแยกจากโค้ด	รหัสไม่หลุดไปกับโค้ด

> **จำง่าย ๆ**
> Claude Code = วิศวกรที่เราสั่งงาน · Ultralytics = กล่องเครื่องมือ (เหมือนปลั๊กอิน) · YOLO26 = สมองที่มองเห็น · OpenCV = ตาและมือ · Gmail = ผู้ส่งสาร

ระบบทั้งหมดต่อกันอย่างไร



เส้นทางของข้อมูล ตั้งแต่กล้องจนถึงอีเมลแจ้งเตือน

01	02	03	04	05	06
กล้อง / วิดีโอ OpenCV อ่านเฟรม + เก็บ buffer ย้อนหลัง	**YOLO26** ตรวจจับวัตถุในทุกเฟรม	**ตัวกรอง** คลาส · โซน · ช่วงเวลา · เจอติดกัน 3 เฟรม	**เหตุการณ์** INTRUSION DETECTED	**หลักฐาน** snapshot + คลิป 8 วิ (H.264)	**อีเมล Gmail** ภาพในตัว + ไฟล์แนบ · พัก 60 วิ

SETUP · 0:25–0:35

S0 เตรียมโปรเจกต์

เป้าหมาย: เครื่องพร้อมรันระบบ — มี .venv ติดตั้งไลบรารีครบ ตรวจคลิปทดสอบแล้ว และมีไฟล์ CLAUDE.md ให้ AI จำโปรเจกต์ได้

PROMPT · S0

Setup · เตรียมโปรเจกต์

```
ช่วยเตรียมโปรเจกต์ "ระบบตรวจจับผู้บุกรุกด้วย YOLO26" ในโฟลเดอร์นี้ให้หน่อย
1. ตรวจว่าเครื่องมี Python 3.10 ขึ้นไปหรือยัง ถ้ายังไม่มีให้บอกวิธีติดตั้งสำหรับเครื่องฉัน (อย่าติดตั้งโปรแกรมระดับเครื่องเองโดยไม่ถาม)
2. สร้าง virtual environment ชื่อ .venv แล้วติดตั้ง ultralytics, opencv-python, pyyaml, python-dotenv, imageio-ffmpeg
3. สร้าง requirements.txt และ .gitignore (ไม่เอา .venv, .env, events/, *.pt)
4. ดูว่าในโฟลเดอร์ videos/ มีคลิปอะไรบ้าง สรุปเป็นตาราง: ชื่อไฟล์ · ความละเอียด · ความยาว · fps
5. สร้างไฟล์ CLAUDE.md บันทึกว่าโปรเจกต์นี้คืออะไร ให้ตอบเป็นภาษาไทย และรัน Python ผ่าน .venv ทุกครั้ง
เสร็จแล้วสรุปว่าพร้อมเริ่มเฟส 1 หรือยัง
```

✓ สิ่งที่ควรเห็น

- Claude Code สรุป **ตารางคลิป 7 คลิป** ในโฟลเดอร์ `videos/` พร้อมความละเอียด ความยาว และ fps
- ในโฟลเดอร์มี `.venv/` `requirements.txt` `.gitignore` และ `CLAUDE.md`
- ข้อความสรุปว่า **พร้อมเริ่มเฟส 1**

เคล็ดลับ

- ถ้าเครื่องยังไม่มี Python 3.10+ Claude Code จะบอกวิธีติดตั้งและรอให้เราตอบก่อน ไม่ติดตั้งโปรแกรมระดับเครื่องเอง
- ต่อจากนี้ให้รัน Python ผ่าน `.venv` เสมอ — ถ้าเจอ `ModuleNotFoundError` แปลว่ารันด้วย Python นอก .venv

PHASE 1 · 0:35–0:50

01 ตรวจจับคนด้วย YOLO26

เป้าหมาย: ได้ไฟล์ `detector.py` + `config.yaml` ที่ตีกรอบแดงรอบคนในวิดีโอ พร้อมจำนวนคนและ FPS — รองรับแหล่งภาพ 3 แบบ: ไฟล์วิดีโอ · webcam (`0`) · กล้องวงจรปิด (`rtsp://`)

PROMPT · P1

Phase 1 · ตรวจจับคนด้วย YOLO26

```
สร้างไฟล์ detector.py ตรวจจับ "คน" จากวิดีโอด้วย YOLO26
- ใช้ ultralytics โหลดโมเดล yolo26n.pt (ระบบดาวน์โหลดให้เองครั้งแรก)
- อ่านค่าจาก config.yaml: source (เริ่มที่ videos/01_door_entry.mp4), model, confidence 0.5
- source รองรับ 3 แบบ: path ไฟล์วิดีโอ · ตัวเลข 0 = webcam · rtsp://... = กล้องวงจรปิด
- ตรวจเฉพาะคลาส person
- วาดกรอบสีแดงรอบคน พร้อมป้าย เช่น person 0.87 และแสดงจำนวนคนที่เจอ + FPS มุมซ้ายบน
- เปิดหน้าต่างแสดงผล กด q เพื่อออก
- เลือก device อัตโนมัติ: cuda > mps (Mac) > cpu
- เพิ่มตัวเลือก --no-window ให้รันแบบไม่เปิดหน้าต่าง (พิมพ์ผลเป็นข้อความแทน) และ --source ไว้เปลี่ยนแหล่งภาพจากคำสั่ง
ทดสอบเองด้วย --no-window ก่อน แล้วบอกคำสั่งให้ฉันรันเองแบบเปิดหน้าต่าง
สุดท้ายอธิบายโค้ดทีละส่วนสั้น ๆ ให้คนเพิ่งเริ่มเข้าใจ
```

✓ สิ่งที่ควรเห็น



ผลจริงจากคลิป 01 — กรอบแดง **person 0.90** และจำนวนคน + FPS ที่มุมจอ

เคล็ดลับ

- ครั้งแรกระบบดาวน์โหลด `yolo26n.pt` ให้เอง ถ้าเน็ตช้าหรือไฟร์วอลล์บล็อก ผู้สอนแจกไฟล์ .pt ให้วางในโฟลเดอร์โปรเจกต์
- รันแบบเปิดหน้าต่างตามคำสั่งที่ Claude Code บอก แล้วกด `q` เพื่อออก · เปลี่ยนแหล่งภาพได้ด้วย `--source`

PHASE 2 · 0:50–1:00

02 เลือกขนาดโมเดล YOLO26

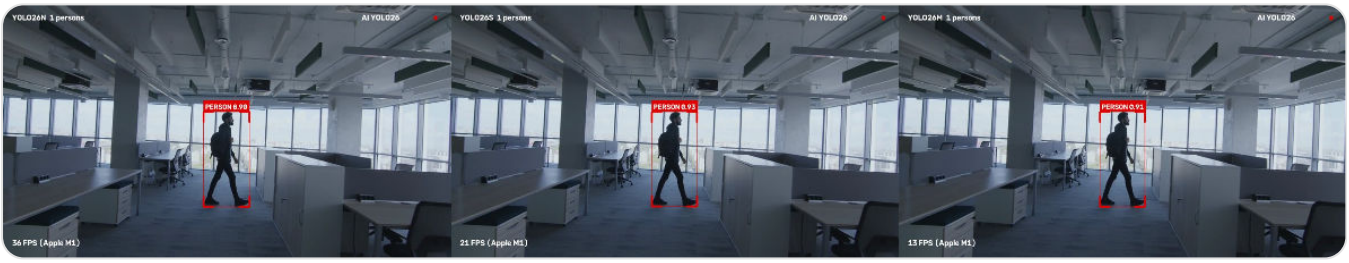
เป้าหมาย: สลับขนาดโมเดลได้ด้วย `--model` และเห็นด้วยตาว่า n / s / m ต่างกันอย่างไร ทั้งความเร็วและความแม่น เพื่อเลือกขนาดที่เหมาะกับเครื่องตัวเอง

PROMPT · P2

Phase 2 · เลือกขนาดโมเดล YOLO26

```
YOLO26 มีหลายขนาด (n, s, m, l, x) ช่วยให้ระบบสลับโมเดลได้ และเปรียบเทียบให้ดูหน่อย
- เพิ่มตัวเลือก --model ให้สลับ yolo26n.pt / yolo26s.pt / yolo26m.pt ได้จากคำสั่ง (ค่าเริ่มต้นอ่านจาก config.yaml)
- สร้าง benchmark.py รันโมเดล n, s, m กับ videos/04_office_wide.mp4 อย่างละ 150 เฟรม
  วัด FPS เฉลี่ย · จำนวนคนที่เจอต่อเฟรมเฉลี่ย · confidence เฉลี่ย แล้วสรุปเป็นตาราง
- บันทึกภาพเปรียบเทียบเฟรมเดียวกัน (เลือกเฟรมที่มีคน) ของทั้ง 3 โมเดลเรียงต่อกันเป็น benchmark.jpg
- แนะนำว่าเครื่องของฉันควรใช้ขนาดไหนสำหรับกล้องวงจรปิดแบบ real-time พร้อมเหตุผล
```

✓ สิ่งที่ควรเห็น



benchmark.jpg — เฟรมเดียวกันจาก 3 โมเดล เรียงจากซ้าย n · s · m

โมเดล	FPS บน Apple M1 (ผลทดสอบจริง)	เหมาะกับ
YOLO26n	~36–46	กล้อง real-time บนโน้ตบุ๊กทั่วไป / CPU
YOLO26s	~21–30	เครื่องแรงขึ้น ต้องการความแม่นเพิ่ม
YOLO26m	~11–13	มีการ์ดจอ หรือไม่ต้องการ real-time

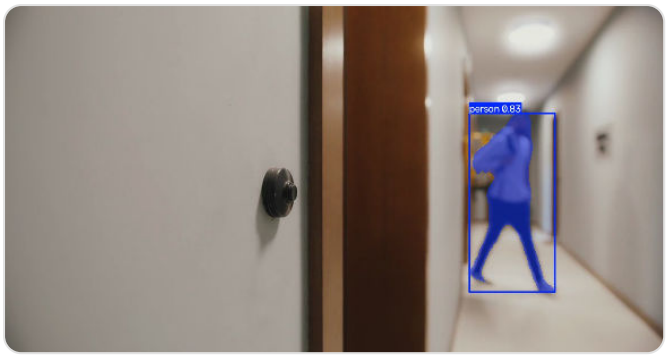
ตัวเลขจากการทดสอบ Prompt ของผู้สอน (Apple M1 · MPS) เครื่องของแต่ละคนจะได้ไม่เท่ากัน

เสริม ลองโมเดลงานอื่นของ YOLO26

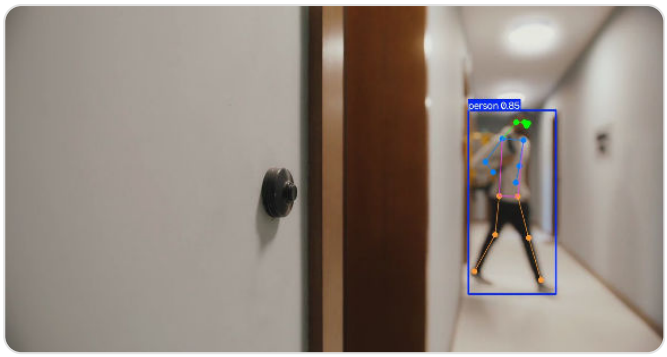
PROMPT · P2B

Phase 2 (เสริม) · ลองโมเดลงานอื่นของ YOLO26

```
YOLO26 ไม่ได้ทำได้แค่ตีกรอบ ลองให้ดูอีก 2 งาน
- สร้าง demo_tasks.py รัน yolo26n-seg.pt (แยกรูปร่างวัตถุทีละพิกเซล) และ yolo26n-pose.pt (จับจุดโครงร่างร่างกาย)
  กับ videos/02_hallway_door.mp4 เลือกเฟรมที่เห็นคนเต็มตัวชัดที่สุด แล้วบันทึกเป็น task_seg.jpg และ task_pose.jpg
- อธิบายสั้น ๆ ว่าแต่ละงานเอาไปใช้กับระบบรักษาความปลอดภัยแบบไหนได้
```



task_seg.jpg — Segmentation แยกรูปร่างคนทีละพิกเซล



task_pose.jpg — Pose จับจุดโครงร่างร่างกาย

เคล็ดลับ

- โน้ตบุ๊กทั่วไปที่ไม่มีการ์ดจอ ใช้ **YOLO26n** ดีที่สุดสำหรับงาน real-time
- FPS ยังต่ำเกินไป ให้สั่ง Claude Code ลด `imgsz` เป็น 480

PHASE 3 · 1:00–1:15

03 กำหนดช่วงเวลาและโซนตรวจจับ

เป้าหมาย: ระบบเฝ้าระวังเฉพาะ **22:00–06:00** (ข้ามเที่ยงคืนได้) ขึ้นป้าย ARMED / DISARMED และนับเฉพาะวัตถุที่เข้ามาใน **โซนที่เราวาดเอง**

PROMPT · P3

Phase 3 · กำหนดช่วงเวลาและโซนตรวจจับ

```
เพิ่มการกำหนด "ช่วงเวลา" และ "โซน" ที่ระบบจะเฝ้าระวัง
ช่วงเวลา
- config.yaml เพิ่ม schedule: enabled, start "22:00", end "06:00" (ต้องรองรับช่วงที่ข้ามเที่ยงคืน)
- ในช่วงเวลา ขึ้นป้าย ARMED สีแดง · นอกช่วงเวลา ขึ้น DISARMED สีเทา ยังแสดงภาพแต่ไม่นับเป็นเหตุการณ์
- เพิ่มตัวเลือก --now HH:MM ไว้จำลองเวลา แล้วทดสอบให้ดูทั้งกรณีอยู่ในช่วง และอยู่นอกช่วง
โซน
- เพิ่มตัวเลือก --draw-zone เปิดเฟรมแรกให้คลิกเมาส์วาดรูปหลายเหลี่ยม (Enter = บันทึก, c = ล้าง, Esc = ยกเลิก)
  บันทึกพิกัดแบบสัดส่วน 0-1 ลง config.yaml แยกตามคลิป/กล้อง (zones: {ชื่อไฟล์: [...]}) ใช้ได้กับทุกความละเอียด
  แก้เฉพาะส่วนโซน อย่าลบคอมเมนต์อื่นใน config
- นับเฉพาะวัตถุที่ "จุดเท้า" (กึ่งกลางขอบล่างของกรอบ) อยู่ในโซน · ในโซนกรอบสีแดง นอกโซนกรอบสีเทา
- วาดเส้นโซนบนภาพ ถ้าคลิป/กล้องไหนยังไม่มีโซนให้ถือว่าทั้งภาพเป็นโซน
บอกคำสั่งให้ฉันลองวาดโซนเองกับ videos/13_cctv_carport_motorcycle.mp4
```

✓ สิ่งที่ควรเห็น



ผลจริงจากคลิป 13 — เส้นโซน **CARPORT** ป้าย **ARMED 22:00–06:00** และกรอบเฉพาะวัตถุในโซน

เคล็ดลับ

- เรียนตอนกลางวันระบบจะขึ้น DISARMED — รันด้วย `--now 23:00` เพื่อจำลองเวลากลางคืน ไม่ต้องแก้ config
- วาดโซนด้วย `--draw-zone` : คลิกเป็นจุด ๆ · `Enter` บันทึก · `c` ล้าง · `Esc` ยกเลิก — ระบบนับ "จุดเท้า" ของวัตถุ จึงวาดโซนบนพื้น
- โซนบันทึก **แยกตามกล้อง/คลิป** วาดของคลิปหนึ่งแล้วไม่ไปทับอีกคลิป · คลิปที่ยังไม่มีโซนจะใช้ทั้งภาพ

PHASE 4 · 1:15–1:25

04 กรองคลาสที่ต้องการตรวจจับ

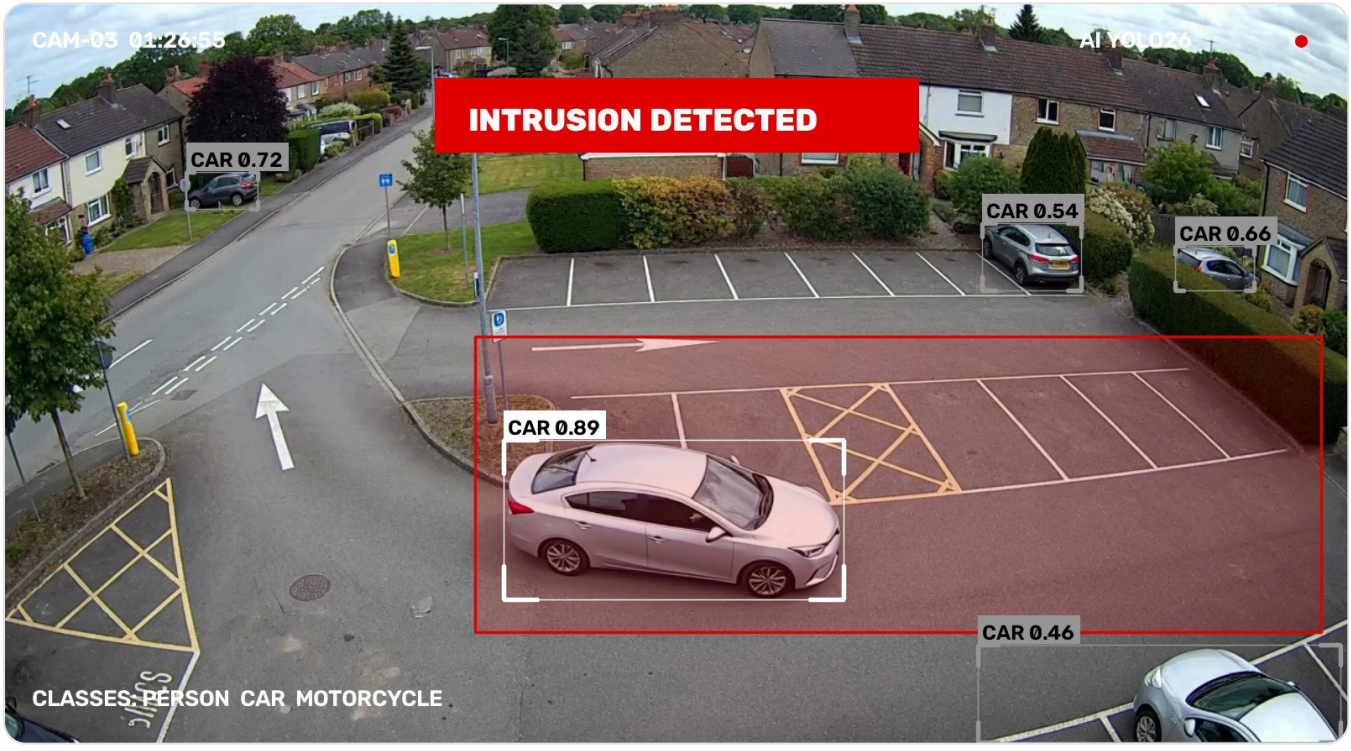
เป้าหมาย: เลือกได้ว่าจะเฝ้าอะไรจาก 80 คลาส (เช่น คน รถยนต์ มอเตอร์ไซค์) สีกรอบต่างกันตามคลาส และ **กันแจ้งเตือนหลอก** ด้วยเงื่อนไขเจอติดกัน 3 เฟรม + พัก 60 วินาที

PROMPT · P4

Phase 4 · กรองคลาสที่ต้องการตรวจจับ

```
ตอนนี้ตรวจแค่คน ช่วยให้เลือกคลาสได้จาก 80 คลาสของ YOLO26 (ชุดข้อมูล COCO)
- config.yaml: classes: [person, car, motorcycle] ใช้ชื่อคลาสแทนตัวเลข ถ้าพิมพ์ผิดให้แนะนำชื่อที่ใกล้เคียง
- เพิ่มตัวเลือก --list-classes แสดงทั้ง 80 คลาสพร้อมชื่อภาษาไทย
- แต่ละคลาสใช้สีกรอบต่างกัน และนับจำนวนแยกตามคลาสบนจอ
- กันแจ้งเตือนหลอก: ต้องเจอติดกันอย่างน้อย min_consecutive_frames: 3 เฟรม ถึงนับเป็น "เหตุการณ์"
  แล้วพักการนับ cooldown_seconds: 60 วินาที (คลิปทดสอบยาวราว 12 วินาที จึงได้ 1 เหตุการณ์ต่อการรัน)
- เมื่อเกิดเหตุการณ์ ขึ้นข้อความ INTRUSION DETECTED บนจอ และพิมพ์ log: เวลา · สิ่งที่เจอ · confidence
- ทดสอบกับ videos/14_cctv_parking_car.mp4 (รถยนต์) และ videos/13_cctv_carport_motorcycle.mp4
(มอเตอร์ไซค์)
  ลานจอดมีรถจอดอยู่แล้ว ให้ใช้โซนเฉพาะทางเข้า ไม่งั้นจะแจ้งเตือนตั้งแต่เฟรมแรก
  ถ้าตอนนี้อยู่นอกช่วงเวลา ให้ทดสอบด้วย --now 23:00 (ไม่ต้องแก้ config)
```

✓ สิ่งที่ควรเห็น



ผลจริงจากคลิป 14 — รถยนต์เข้าโซนทางเข้า ระบบขึ้น **INTRUSION DETECTED** ส่วนรถที่จอดอยู่นอกโซนเป็นกรอบเทา

เคล็ดลับ

- ลานจอดมีรถจอดอยู่แล้ว ถ้าใช้ทั้งภาพจะแจ้งเตือนตั้งแต่เฟรมแรก — วาดโซนเฉพาะ **ทางเข้า**
- ใช้ชื่อคลาสภาษาอังกฤษตาม COCO ดูรายชื่อทั้ง 80 คลาสพร้อมคำแปลด้วย `--list-classes`
- ยังเตือนหลอกบ่อย ให้เพิ่ม confidence เป็น 0.6 หรือ `min_consecutive_frames` เป็น 5

PHASE 5 · 1:25–1:50

05 เก็บหลักฐาน + ส่งอีเมลแจ้งเตือน

เป้าหมาย: เมื่อเกิดเหตุ ระบบเก็บ **snapshot + คลิป 8 วินาที** (ก่อนเหตุ 3 วิ + หลังเหตุ 5 วิ) แล้วส่ง **อีเมลภาษาไทยพร้อมภาพและไฟล์แนบ** ผ่าน Gmail — ทำ 3 ขั้น: P5A → สร้าง App Password → P5B

5.1 เก็บหลักฐาน ภาพ + คลิป 8 วินาที

PROMPT · P5A

Phase 5 (1/2) · เก็บหลักฐาน ภาพ + คลิป 8 วินาที

```
เมื่อเกิดเหตุการณ์ ให้ระบบเก็บหลักฐานอัตโนมัติ
- เก็บเฟรมย้อนหลังไว้ในหน่วยความจำตลอดเวลา (buffer) เพื่อให้มีภาพก่อนเกิดเหตุ
- เมื่อเกิดเหตุ บันทึก snapshot.jpg และ clip.mp4 (ทั้งคู่มีกรอบตรวจจับ) ยาว 8 วินาที
  = ก่อนเกิดเหตุ 3 วินาที + หลังเกิดเหตุ 5 วินาที (ตั้งค่า pre_seconds / post_seconds ใน config.yaml)
- ใส่วันเวลาและชื่อกล้องบนคลิป แล้วแปลงเป็น H.264 ด้วย ffmpeg จากแพ็กเกจ imageio-ffmpeg ให้เปิดได้ทุกเครื่องและในมือถือ
- ถ้า source เป็นไฟล์วิดีโอ ให้เล่นตามความเร็วจริงและใช้เวลาของวิดีโอ คลิปจะได้ยาวถูกต้อง
- เก็บไว้ที่ events/[วันที่_เวลา]/ พร้อม event.json (เวลา · สิ่งที่เจอ · confidence · ชื่อกล้อง)
- การบันทึกคลิปต้องไม่ทำให้การตรวจจับสะดุด (ใช้ thread แยก)
ทดสอบกับ videos/11_cctv_office_day.mp4 (ถ้าอยู่นอกช่วงเวลา ใช้ --now 23:00) แล้วเช็คให้ดูว่าคลิปยาวประมาณ 8 วินาทีจริง
```

✓ สิ่งที่ควรเห็น



ผลจริง — ระบบตรวจพบคน แล้วบันทึก **snapshot.jpg + clip.mp4 (8 วินาที)** ลงโฟลเดอร์ `events/`

ในโฟลเดอร์ `events/[วันที่_เวลา]/` จะมี `snapshot.jpg` `clip.mp4` และ `event.json` — คลิปเปิดได้ทั้งบนคอมและมือถือ

5.2 สร้าง Google App Password (ทำเองทีละขั้น)

ห้ามวางรหัสในแชท AI — App Password คือกุญแจเข้าอีเมลของเรา ให้ **พิมพ์ลงไฟล์ .env ด้วยตัวเอง** เท่านั้น ไม่ต้องบอก Claude Code และไม่ต้องส่งให้ใคร

1. **เปิดการยืนยันแบบ 2 ขั้นตอน (2-Step Verification)**
   ไปที่ `https://myaccount.google.com/security` แล้วเปิด ถ้ายังไม่เปิด
2. **เปิดหน้า App Password**
   ไปที่ `https://myaccount.google.com/apppasswords` (ล็อกอินอีกครั้งถ้าถูกถาม)
3. **ตั้งชื่อแอป แล้วกด "สร้าง"**
   เช่น `YOLO26 Alert`
4. **คัดลอกรหัส 16 ตัวที่แสดง**
   รหัสแสดงครั้งเดียว · จะเว้นวรรคหรือไม่ก็ได้
5. **เปิดไฟล์ .env ในโปรเจกต์ แล้วพิมพ์เอง**
   `SMTP_USER=อีเมลของคุณ` · `SMTP_PASSWORD=รหัส16ตัว` · `EMAIL_TO=อีเมลผู้รับ` · `EMAIL_SEND=true`
6. **ทดสอบส่งอีเมล แล้วเช็คกล่องจดหมาย**
   Mac: `.venv/bin/python detector.py --test-email` · Windows: `.venv\Scripts\python detector.py --test-email`

⚠ ถ้าทำไม่ได้

- **ไม่เห็นเมนู App Password** — ยังไม่เปิด 2-Step หรือเป็นบัญชีองค์กร (Google Workspace) ที่แอดมินปิดไว้
- **ใช้รหัส Gmail ปกติไม่ได้** ต้องเป็น App Password เท่านั้น · ยกเลิกรหัสได้ทุกเมื่อที่หน้าเดิม
- ทำ P5B ก่อนก็ได้ (ทดสอบแบบ `EMAIL_SEND=false` ยังไม่ต้องใช้รหัส) แล้วค่อยกลับมาใส่รหัสใน .env ทีหลัง

5.3 ส่งอีเมลแจ้งเตือน

PROMPT · P5B

Phase 5 (2/2) · ส่งอีเมลแจ้งเตือน

```
เมื่อเก็บหลักฐานเสร็จ ให้ส่งอีเมลแจ้งเตือนผ่าน Gmail
- ใช้ smtp.gmail.com พอร์ต 465 (SSL) อ่านค่าจากไฟล์ .env:
  SMTP_USER · SMTP_PASSWORD (App Password 16 ตัว) · EMAIL_TO (หลายคนคั่นด้วยจุลภาค) ·
EMAIL_SEND=false/true
- สร้าง .env.example ให้ด้วย ห้ามใส่รหัสจริงในโค้ด และห้ามแสดงรหัสในแชท
- หัวเรื่อง เช่น [แจ้งเตือน] พบ คน 1 · รถยนต์ 1 — CAM-01 22:15:08
- เนื้อหา HTML ภาษาไทย: สิ่งที่ตรวจพบ · เวลา · ชื่อกล้อง/สถานที่ · ภาพ snapshot แสดงในตัวอีเมล
- แนบไฟล์ snapshot.jpg และ clip.mp4
- ถ้า EMAIL_SEND=false ให้บันทึกเป็น email.eml ในโฟลเดอร์เหตุการณ์แทน (ไว้ทดสอบโดยไม่ส่งจริง)
- ส่งอีเมลใน thread แยก ถ้าส่งไม่สำเร็จให้บอกสาเหตุเป็นภาษาไทย (เช่น รหัสผิด ยังไม่เปิด 2-Step) แต่การตรวจจับต้องทำงานต่อ
- เพิ่มตัวเลือก --test-email ส่งอีเมลทดสอบ 1 ฉบับ
ทดสอบแบบ EMAIL_SEND=false ให้ดูก่อน ฉันจะใส่รหัสใน .env เอง แล้วบอกขั้นตอนที่ฉันต้องทำต่อ
```

✓ สิ่งที่ควรเห็น

- ทดสอบแบบ `EMAIL_SEND=false` ได้ไฟล์ `email.eml` ในโฟลเดอร์เหตุการณ์ — ดับเบิลคลิกเปิดดูหน้าตาอีเมลได้ มีหัวเรื่องภาษาไทย ภาพในตัว และไฟล์แนบครบ
- ใส่รหัสใน .env และตั้ง `EMAIL_SEND=true` แล้ว อีเมลจริงเข้ากล่องจดหมาย หัวเรื่องเช่น `[แจ้งเตือน] พบ คน 1 — CAM-01 22:15:08`

> **เคล็ดลับ**
>
> - ขั้นส่งอีเมลจริง Claude Code จะ **ถามสิทธิ์ก่อนรัน** — อ่านคำสั่งแล้วกดอนุญาตได้
> - ขึ้น `535 Username and Password not accepted` แปลว่าใช้รหัส Gmail ปกติหรือพิมพ์ผิด — สร้าง App Password ใหม่แล้วใส่ใน .env

---

FULL RUN · 1:50–2:00

รันระบบเต็ม

> **เป้าหมาย:** รวมทุกอย่าง (ช่วงเวลา · โซน · คลาส · หลักฐาน · อีเมล) รันกับคลิปกลางคืนอินฟราเรด แล้วให้ AI บันทึกวิธีใช้ระบบลง CLAUDE.md

PROMPT · RUN — รันระบบเต็ม

```
รันระบบเต็มกับ videos/12_cctv_office_night_ir.mp4 โดยใช้ทุกอย่างที่ทำมา (ช่วงเวลา · โซน · คลาส · หลักฐาน · อีเมล)
ถ้าตอนนี้อยู่นอกช่วงเวลาที่ตั้งไว้ ให้ใช้ --now 23:00 · ถ้าคลิปนี้ยังไม่มีโซน ให้ใช้ทั้งภาพ
สรุปผล: เกิดเหตุการณ์กี่ครั้ง · ไฟล์อยู่ที่ไหน · ส่งอีเมลสำเร็จหรือไม่
แล้วอัปเดต CLAUDE.md ว่าระบบทำอะไรได้บ้าง และรันด้วยคำสั่งอะไร
```

✓ หน้าตาโฟลเดอร์เมื่อจบคลาส

```
yolo26-detection/
├── videos/          คลิปทดสอบ 7 คลิป
├── detector.py      ระบบหลัก (Phase 1–5)
├── config.yaml      source · model · schedule · zones · classes
├── benchmark.py     เทียบโมเดล n / s / m (Phase 2)
├── demo_tasks.py    ตัวอย่าง seg / pose (Phase 2 เสริม)
├── .env             รหัส Gmail ที่เราพิมพ์เอง (ไม่แชร์)
├── .env.example     ตัวอย่างไฟล์ .env ไม่มีรหัสจริง
├── events/          snapshot.jpg · clip.mp4 · event.json · email.eml
├── requirements.txt · .gitignore
└── CLAUDE.md        บันทึกโปรเจกต์ให้ AI จำได้
```

> **เคล็ดลับ**
>
> - ไม่มีเหตุการณ์เลย ส่วนใหญ่เพราะอยู่นอกช่วงเวลา (DISARMED) — ใช้ `--now 23:00`
> - เปิด session ใหม่วันหลัง Claude Code จะอ่าน CLAUDE.md แล้วรู้ทันทีว่าระบบทำอะไรได้และรันด้วยคำสั่งอะไร

BONUS · ต่อยอดหลังเรียน

ต่อยอด: กล้องจริงและแผงควบคุมเว็บ

สองขั้นนี้ไม่บังคับในคลาส ผู้สอนสาธิตให้ดู แล้วผู้เรียนนำกลับไปทำต่อกับกล้องที่บ้านหรือร้านของตัวเองได้

B1 ต่อกล้องวงจรปิด (RTSP) และ webcam

PROMPT · B1

Bonus · ต่อกล้องวงจรปิด (RTSP) และ webcam

```
ฉันจะต่อระบบกับกล้องจริง ช่วยเตรียมให้หน่อย
- อธิบายรูปแบบ URL RTSP ของกล้องยอดนิยม (Hikvision, Dahua, TP-Link Tapo) แบบ main stream และ sub stream
  และแนะนำว่าควรใช้ sub stream เพราะอะไร
- ให้ config.yaml อ่าน source จาก .env (CAMERA_URL) เพื่อไม่ให้รหัสกล้องอยู่ในไฟล์ config
- ถ้ากล้องหลุด ให้ต่อใหม่อัตโนมัติ แต่ถ้าต่อไม่ได้ตั้งแต่แรกให้แจ้ง error ทันที
- ทดสอบกับ webcam ของเครื่องนี้ (source: 0) ให้ดู
กล้องของฉัน: [ยี่ห้อ] · IP [192.168.1.xx] · user [admin]  (รหัสผ่านฉันจะใส่ใน .env เอง)
```

เคล็ดลับ

- ทดสอบ URL กล้องในโปรแกรม VLC ก่อน ถ้า VLC เปิดไม่ได้ ระบบก็เปิดไม่ได้ · คอมกับกล้องต้องอยู่วง LAN เดียวกัน · ใช้ sub stream
- webcam บน Mac เปิดไม่ได้ → System Settings → Privacy → Camera → อนุญาตแอปที่ใช้รัน

B2 แผงควบคุมบนเว็บ

PROMPT · B2

Bonus · แผงควบคุมบนเว็บ

```
สร้างแผงควบคุมบนเว็บด้วย Flask (webapp.py) ที่ใช้ detector.py ตัวเดิม
- เลือกคลิป/webcam/RTSP · เลือกคลาสจากรายการ · เลือกขนาดโมเดล · ปรับ confidence
- ดูภาพสดพร้อมกรอบตรวจจับ (MJPEG stream) · ปุ่มเริ่ม/หยุด
- รายการเหตุการณ์ล่าสุด กดดูภาพ คลิป และตัวอย่างอีเมลได้
- ดีไซน์เรียบแบบหน้าจอห้องควบคุมกล้อง ใช้ไอคอน SVG ห้ามใช้อีโมจิ รองรับมือถือ
เปิดที่ http://127.0.0.1:8000 แล้วบอกวิธีใช้
```

✓ เปิดเบราว์เซอร์ที่ `http://127.0.0.1:8000` จะเห็นภาพสดพร้อมกรอบตรวจจับ ปุ่มเริ่ม/หยุด และรายการเหตุการณ์ล่าสุด — ผู้สอนโชว์แผงควบคุมตัวอย่างในคลาส

TROUBLESHOOTING

แก้ปัญหาที่พบบ่อย

อาการ	สาเหตุ	วิธีแก้
`ModuleNotFoundError: ultralytics`	รันด้วย Python นอก .venv	ใช้ `.venv/bin/python` (Windows `.venv\Scripts\python`)
ดาวน์โหลด yolo26n.pt ไม่ได้	อินเทอร์เน็ต/ไฟร์วอลล์	ผู้สอนแจกไฟล์ .pt วางในโฟลเดอร์โปรเจกต์
FPS ต่ำมาก	ใช้โมเดลใหญ่บน CPU	ใช้ yolo26n · ลด imgsz เป็น 480
เตือนหลอกบ่อย	confidence ต่ำ / ไม่มีโซน	เพิ่ม confidence 0.6 · วาดโซน · min_consecutive_frames 5
ไม่มีเหตุการณ์เลย	อยู่นอกช่วงเวลา (DISARMED)	รันด้วย `--now 23:00` เพื่อจำลองเวลา หรือ `enabled: false`
แจ้งเตือนตั้งแต่เฟรมแรก	มีรถจอดอยู่ในภาพแล้ว	วาดโซนเฉพาะทางเข้า (โซนแยกตามกล้อง)
คลิปเปิดในมือถือไม่ได้	ยังเป็น mp4v	แปลง H.264 ด้วย imageio-ffmpeg (P5A)
`535 Username and Password not accepted`	ใช้รหัส Gmail ปกติ / พิมพ์ผิด	สร้าง App Password ใหม่ ใส่ใน .env
webcam เปิดไม่ได้บน Mac	ยังไม่ให้สิทธิ์กล้อง	System Settings → Privacy → Camera → อนุญาต Terminal/แอปที่รัน
RTSP ต่อไม่ได้	URL/รหัสผิด หรือคนละวง LAN	ทดสอบ URL ใน VLC ก่อน · ใช้ sub stream

QUICK REFERENCE

Prompt Cheat Sheet

Prompt ทั้งหมดของวันนี้เรียงตามลำดับ — ฉบับเว็บกดคัดลอกได้ที่บทของแต่ละ Prompt

ID	Prompt	ช่วงเวลา
S0	Setup · เตรียมโปรเจกต์	0:25–0:35
P1	Phase 1 · ตรวจจับคนด้วย YOLO26	0:35–0:50
P2	Phase 2 · เลือกขนาดโมเดล YOLO26	0:50–1:00
P2B	Phase 2 (เสริม) · ลองโมเดลงานอื่นของ YOLO26	0:50–1:00
P3	Phase 3 · กำหนดช่วงเวลาและโซนตรวจจับ	1:00–1:15
P4	Phase 4 · กรองคลาสที่ต้องการตรวจจับ	1:15–1:25
P5A	Phase 5 (1/2) · เก็บหลักฐาน ภาพ + คลิป 8 วินาที	1:25–1:50
P5B	Phase 5 (2/2) · ส่งอีเมลแจ้งเตือน	1:25–1:50
RUN	รันระบบเต็ม	1:50–2:00
B1	Bonus · ต่อกล้องวงจรปิด (RTSP) และ webcam	ต่อยอด
B2	Bonus · แผงควบคุมบนเว็บ	ต่อยอด

ผู้สอน



เอกรัตน์ ใจชนะ (นิว) · AI Instructor, ZenityX

สอนจากประสบการณ์ทำระบบนี้จริง — ผลงาน "ระบบกล้องวงจรปิดตรวจจับอัจฉริยะ" ได้รางวัลชนะเลิศ Oral Presentation (ด้านนวัตกรรม) การประกวดผลงานสหกิจศึกษา CWIE ดีเด่น เครือข่ายพัฒนา CWIE ภาคเหนือตอนล่าง

ติดต่อสถาบัน ZenityX

โทร
062-262-9644

อีเมล
admin@zenityxai.com

ออกแบบหลักสูตรโดยสถาบัน ZenityX · Smart Intrusion Detection with YOLO26 · Workshop 2 ชั่วโมง