GOLF TRACER — คู่มือการเทรนโมเดลตรวจจับ

คู่มือการเทรนและเตรียมข้อมูล สำหรับทีมพัฒนา AI

ชุดข้อมูล 2 คลาส (club, golf_ball) · ที่มา สถานะ วิธีเทรน และกับดักที่เคยทำพลาดจริง

จัดทำโดยทีมพัฒนา Golf Tracer

YOLO (Ultralytics 8.4.106) · PyTorch 2.13.0 · ONNX opset 12 · imgsz 960

ชุดข้อมูลต้นทาง: `labelhub` บนเครื่อง go-ai · สร้างด้วย `build_dataset.py` จาก `labelhub.db`

31 ก.ค. 2569 · v1.0 · ตัวเลขทุกตัวในเอกสารนี้วัดจากข้อมูลจริง ณ วันที่จัดทำ

บทคัดย่อ.

เอกสารนี้เป็นคู่มือส่งมอบงานเทรนโมเดลตรวจจับของ Golf Tracer ให้ทีมพัฒนา AI ทำต่อได้ด้วยตัวเอง ประกอบด้วยสี่ส่วน คือ (1) ของที่ส่งมอบและวิธีตรวจว่าครบจริง (2) ที่มาและตัวเลขจริงของชุดข้อมูล รวมถึงกติกาการแบ่ง train/val/test ที่ต้องรักษาไว้ (3) ขั้นตอนการเทรนพร้อมคำสั่งที่ใช้ได้ทันที และตัวเลขพื้นฐานที่ต้องเอาชนะ (4) กับดักที่โครงการนี้เคยเสียเวลาไปกับมันจริง ๆ พร้อมวิธีเลี่ยง ข้อสรุปสำคัญสามข้อ ข้อแรก ชุดข้อมูลที่ส่งมอบผ่านการตรวจครบทุกข้อแล้ว 8,224 ภาพ ป้ายกำกับตรงกันหนึ่งต่อหนึ่ง และจำนวนกล่องตรงกับใบสรุปที่เครื่องมือสร้างชุดข้อมูลเขียนไว้ ข้อสอง ตัวเลขที่ต้องเอาชนะไม่ใช่ mAP แต่เป็น recall ของลูกกอล์ฟ ซึ่งปัจจุบันอยู่ที่ 0.665 ต่ำกว่าของหัวไม้ที่ 0.819 อย่างเห็นได้ชัด เพราะลูกที่โมเดลมองไม่เห็นคือรอยขาดของเส้นวิถี ข้อสาม มีกับดักหนึ่งที่พร้อมทำให้เสียเวลาเทรนทั้งชั่วโมงโดยไม่รู้ตัว คือค่าตั้งต้นของ `train.py` ชี้ไปที่ไฟล์กำหนดชุดข้อมูลที่มีคลาสเดียว รายละเอียดอยู่ในหัวข้อที่ 6 และต้องอ่านก่อนเทรนครั้งแรก

คำสำคัญ— *object detection, YOLO, dataset preparation, train/val split, golf ball tracking*

สารบัญ

1. ของที่ส่งมอบ และวิธีตรวจว่าครบ

ทั้งหมดอยู่ที่ `~/Desktop/golf-tracer-training` บนเครื่องนี้ ขนาดรวมประมาณ 3.3 กิกะไบต์ ไฟล์ภาพเป็นไฟล์จริงทั้งหมด ไม่ใช่ทางลัด (symlink) ประเด็นนี้สำคัญกว่าที่เห็น เพราะชุด

ข้อมูลฉบับบนเครื่อง go-ai เก็บภาพเป็นทางลัดชี้กลับไปที่โฟลเดอร์ labelhub/data/frames การคัดลอกแบบธรรมดาจะได้ทางลัดที่ชี้ไปในที่ที่ไม่มีอยู่บนเครื่องนี้ 8,224 อัน และความผิดพลาดแบบนั้นจะไม่แสดงตัวตอนคัดลอก แต่จะแสดงตอนเทรนไปแล้ว การส่งมอบครั้งนี้จึงคลายทางลัดทุกอันให้เป็นไฟล์จริง และตรวจยืนยันว่าไม่มีทางลัดเหลืออยู่เลย

ตำแหน่ง	เนื้อหา	ขนาด
dataset/	ภาพและป้ายกำกับ 3 ชุด (train, val, test) พร้อม data.yaml และ summary.json	3.2 GB
models/yolo_birdiex_2c_1.pt	โมเดลพื้นฐานที่ใช้เป็นจุดเริ่มการ fine-tune และเป็นตัวเปรียบเทียบ (md5 a2eee2f…)	18 MB
models/yolo_birdiex_2c_20260728-004149.*	โมเดลที่เทรนล่าสุด ทั้ง .pt, .onnx และไฟล์ตัวเลขผล .json	54 MB
labelhub/labelhub.db	ฐานข้อมูลการ label ใช้ตรวจว่ากล่องแต่ละกล่องใครวาด เมื่อไร และสร้างชุดข้อมูลใหม่ได้	2.1 MB
scripts/	เครื่องมือทั้งหมด: ตรวจชุดข้อมูล, เทรน, สร้างชุดข้อมูลใหม่, ประเมินผลหลังเทรน	40 KB
docs/	เอกสารฉบับนี้	—

ตารางที่ 1 โครงสร้างของที่ส่งมอบ

> ก่อนเทรนทุกครั้ง ให้รัน python3 scripts/verify_dataset.py คำสั่งนี้ตรวจเจ็ดเรื่องที่เคยผิดพลาดมาแล้วจริงในโครงการนี้ และจะคืนค่าไม่เป็นศูนย์ถ้ามีข้อใดไม่ผ่าน ใช้เวลาไม่ถึงนาที ขณะที่ทางเลือกอีกทางคือรู้ตัวหลังเทรนไปหนึ่งชั่วโมง

2. ชุดข้อมูล: ตัวเลขจริง

ชุดข้อมูลนี้ชื่อ clean สร้างขึ้นโดยอ่านจากฐานข้อมูลการ label โดยตรง ไม่ได้อ่านจากไฟล์ที่ส่งออกไว้ก่อนหน้า เหตุผลของการออกแบบเช่นนี้อยู่ในหัวข้อที่ 9 ซึ่งเป็นบทเรียนจากความผิดพลาดจริง สองคลาสคือ 0 = club (หัวไม้) และ 1 = golf_ball (ลูกกอล์ฟ) ลำดับนี้เปลี่ยนไม่ได้ เพราะเป็นลำดับเดียวกับที่โมเดลที่ใช้งานอยู่คาดหวัง คอลัมน์สุดท้าย "ลูกช่วงลอย" คือจำนวนกล่องลูกกอล์ฟที่อยู่ในเฟรมหลังจังหวะอิมแพกต์ของคลิปนั้น เป็นตัวเลขที่เครื่องมือสร้าง

ชุดข้อมูลนับเอาไว้เอง ไม่ใช่ตัวเลขที่คำนวณย้อนหลังจากไฟล์ป้ายกำกับ เพราะการนับต้องรู้เฟรมอิมแพกต์ของแต่ละคลิปซึ่งอยู่ในฐานข้อมูล ความสำคัญของมันคือใช้เป็นเกณฑ์ว่าชุด val และ test วัด recall ของลูกได้จริงหรือไม่

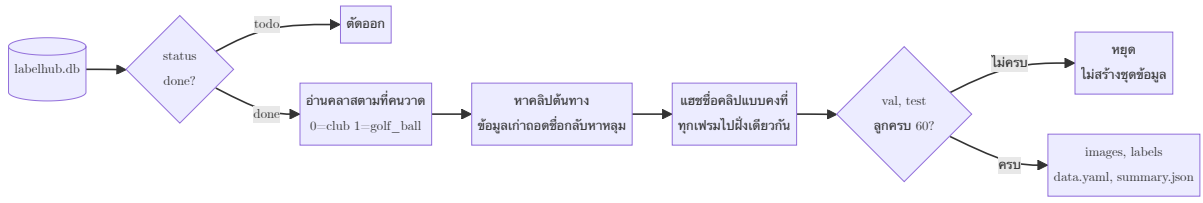
ชุด	ภาพ	ป้ายกำกับ	กล่อง club	กล่อง golf_ball	ลูกช่วงลอย
train	6,727	6,727	4,717	3,629	627
val	791	791	490	525	120
test	706	706	91	283	219
รวม	8,224	8,224	5,298	4,437	966

ตารางที่ 2 จำนวนภาพและกล่องต่อชุด ตรวจยืนยันแล้วตรงกับ summary.json ที่เครื่องมือสร้างชุดข้อมูลเขียนไว้

เกณฑ์ที่เครื่องมือบังคับไว้: ชุด val และ test ต้องมีกล่องลูกกอล์ฟอย่างน้อย 60 กล่อง ถ้าน้อยกว่านั้นเครื่องมือจะปฏิเสธการสร้างชุดข้อมูล เพราะ recall ของลูกคือตัวเลขเดียวที่บอกว่าเส้นวิถีจะขาดหรือไม่ และวัดมันไม่ได้ถ้าชุดประเมินไม่มีลูก ปัจจุบัน val มี 525 กล่อง และ test มี 283 กล่อง ผ่านเกณฑ์ทั้งคู่

3. กติกาการแบ่ง train / val / test ที่ต้องรักษาไว้

การแบ่งชุดข้อมูลนี้ไม่ได้สุ่มเป็นภาพ ๆ แต่สุ่มเป็นคลิป และนี่คือจุดที่พลาดง่ายที่สุดในงานลักษณะนี้ เฟรมที่อยู่ติดกันในคลิปเดียวกันแทบจะเหมือนกัน ถ้าแบ่งแบบสุ่มรายภาพ เฟรมที่ 100 อาจไปอยู่ชุด train ขณะที่เฟรมที่ 101 ไปอยู่ชุด val ผลที่ได้คือตัวเลข val สวยงามแต่ไม่มีความหมาย เพราะมันวัดการจำ ไม่ใช่การรู้จำแนก เครื่องมือจึงคำนวณค่าแฮชจากชื่อคลิป แล้วส่งทุกเฟรมของคลิปนั้นไปอยู่ฝั่งเดียวกันทั้งหมด และเพราะเป็นการแฮชแบบคงที่ การรันซ้ำจะได้การแบ่งเดิมทุกครั้ง สำหรับข้อมูลชุดเก่าที่ชื่อไฟล์มีรหัสสุ่มนำหน้า เครื่องมือมีขั้นตอนถอดชื่อกลับไปหาหลุมต้นทางก่อน เพื่อให้เฟรมจากหลุมเดียวกันไม่กระจายข้ามชุด



ภาพ 1. ลำดับการสร้างชุดข้อมูลจากฐานข้อมูลการ label

ถ้าต้องสร้างชุดข้อมูลใหม่ ให้ใช้ `scripts/build_dataset.py` เท่านั้น อย่าเขียนสคริปต์แบ่งชุดขึ้นใหม่เอง กติกาสามข้อที่ต้องไม่หายไปคือ แบ่งเป็นคลิปไม่ใช่เป็นภาพ อ่านคลาสจากฐานข้อมูลไม่ใช่จากไฟล์ที่ส่งออกไว้ และมีเกณฑ์จำนวนลูกขั้นต่ำในชุดประเมิน

4. สถานะการ label ปัจจุบัน

ความคืบหน้า 99.47 เปอร์เซ็นต์ จากทั้งหมด 11,803 เฟรม มีคนดูและบันทึกแล้ว 11,741 เฟรม เหลือ 62 เฟรม นิยามที่ฐานข้อมูลใช้ระบุไว้ชัด สถานะ `todo` หมายถึงยังไม่มีคนเห็นเฟรมนั้นเลย สถานะ `done` หมายถึงมีคนกดบันทึกแล้ว โดยนับรวมกรณีที่ตรวจแล้วไม่มีอะไรต้องวาด และนับรวมกรณีที่คนรับข้อเสนอของโมเดลด้วยการกดปุ่ม เพราะการยอมรับก็คือคำตอบของคนเช่นกัน ผู้ที่ลงแรง label เรียงตามจำนวนเฟรม คือ win 8,832 เฟรม, Tanupat Panyasantikul 1,912 เฟรม, admin 705 เฟรม และ siam 292 เฟรม

รายการ	จำนวน	หมายเหตุ
เฟรมที่บันทึกแล้ว	11,741 / 11,803	99.47 เปอร์เซ็นต์
เฟรมที่เหลือ	62	อยู่ในคลิปเดียว ดูหัวข้อที่ 9
คลิปทั้งหมด	50	ทำเส้นวิถีเสร็จแล้ว 49 คลิป
กล่องที่คนวาด	9,462	หัวไม้ 5,525 · ลูก 3,937
กล่องที่โมเดลเสนอและยังไม่มีคนยืนยัน	125	หัวไม้ 92 · ลูก 33
เฟรมที่ผ่านการรีวิวซ้ำ	108	คิดเป็น 0.9 เปอร์เซ็นต์ของที่บันทึกแล้ว

ตารางที่ 3 สถานะการ label ณ 31 ก.ค. 2569 อ่านจาก labelhub.db

ข้อควรระวังสำหรับการวางแผนงานต่อ อัตราการรีวิวซ้ำอยู่ที่ 0.9 เปอร์เซ็นต์ หมายความว่าป้ายกำกับเกือบทั้งหมดผ่านสายตาคนเพียงคนเดียวแล้วเข้าสู่การเทรน ถ้าจะยกระดับคุณภาพโมเดล การสุ่มรีวิวซ้ำน่าจะให้ผลตอบแทนสูงกว่าการเพิ่มจำนวนเฟรมใหม่

5. โมเดลพื้นฐาน และตัวเลขที่ต้องเอาชนะ

โมเดลที่เทรนล่าสุดคือ `yolo_birdiex_2c_20260728-004149` เทรนที่ขนาดภาพ 960 พิกเซล 120 รอบ ส่งออกเป็น ONNX opset 12 ตัวเลขทั้งหมดด้านล่างมาจากไฟล์ผลที่ระบบเขียนไว้

เอง ไม่ใช่การประมาณ ตัวเลขที่ควรใช้ตัดสินคือ recall ของลูกกอล์ฟ ไม่ใช่ mAP เหตุผลตรงไปตรงมา ลูกที่โมเดลมองไม่เห็นในเฟรมหนึ่งคือรอยขาดหนึ่งจุดบนเส้นวิถีที่ผู้ใช้เห็น ขณะที่ precision ที่ลดลงเล็กน้อยมักถูกตัวติดตามกรองออกได้ ปัจจุบันช่องว่างระหว่าง recall ของหัวไม้ที่ 0.819 กับของลูกที่ 0.665 คือประมาณ 15 จุด และเป็นเป้าหมายที่ชัดที่สุดของการเทรนรอบต่อไป

ตัวชี้วัด	ค่า	อ่านอย่างไร
mAP50	0.7956	ค่ารวม ใช้ดูภาพกว้าง แต่ไม่ใช่ตัวตัดสินสำหรับงานนี้
mAP50-95	0.4680	เข้มงวดกว่า สะท้อนความแม่นของขอบกล่อง
recall หัวไม้	0.8191	หาหัวไม้เจอ 82 เปอร์เซ็นต์ของที่มีจริง
precision หัวไม้	0.9525	ที่ทายว่าเป็นหัวไม้ ถูก 95 เปอร์เซ็นต์
recall ลูกกอล์ฟ	0.6648	จุดอ่อนที่แท้จริง — ดูคำอธิบายด้านล่าง
precision ลูกกอล์ฟ	0.8924	ที่ทายว่าเป็นลูก ถูก 89 เปอร์เซ็นต์

ตารางที่ 4 ผลการวัดของโมเดลล่าสุด บนชุด val เดียวกับที่ใช้เปรียบเทียบ

ทิศทางที่มีเหตุผลรองรับ กรณีที่โมเดลพลาดคือลูกเบลอจากการเคลื่อนที่เร็วที่ 60 เฟรมต่อวินาที ลูกสีขาวบนท้องฟ้าที่มีเมฆ และลูกที่อยู่ไกลจนเล็กมาก การเพิ่มข้อมูลที่เจาะสามกรณีนี้ให้ผลมากกว่าการเพิ่มเฟรมทั่วไปในจำนวนเท่ากัน

6. ขั้นตอนการเทรน

การเทรนไม่ได้เริ่มจากศูนย์ แต่ fine-tune ต่อจากโมเดลที่ใช้งานอยู่ เหตุผลคือชุดข้อมูลที่มีป้ายกำกับยังเล็ก หน้าที่ของมันคือสอนกรณีที่โมเดลปัจจุบันพลาด ไม่ใช่สอนใหม่ว่าลูกกอล์ฟหน้าตาเป็นอย่างไร คำสั่งเดียวที่ต้องใช้ และควรใช้แทนการเรียก `train.py` ตรง ๆ คือ

- `bash scripts/train.sh` — เทรน 60 รอบจากโมเดลพื้นฐาน โดยตรวจชุดข้อมูลให้ก่อนเสมอ
- `bash scripts/train.sh --epochs 120` — ตัวเลือกทุกตัวของ `train.py` ส่งผ่านได้ตามปกติ
- `python3 scripts/verify_dataset.py` — ตรวจชุดข้อมูลอย่างเดียว ไม่เทรน

ตัวเลือก	ค่าตั้งต้น	เหตุผล
`--epochs`	60	ปรับได้ รอบล่าสุดที่ให้ผลดีที่สุดใช้ 120
`--imgsz`	960	ต้องตรงกับที่แอปใช้ตอนอนุมาน ถ้าไม่ตรงผลจะเพี้ยน
`--batch`	8	เหมาะกับหน่วยความจำของเครื่อง Mac Studio
`--device`	mps	เร่งความเร็วด้วย GPU ของ Apple Silicon
`--patience`	8	หยุดเองเมื่อไม่ดีขึ้น 8 รอบติดกัน
`--opset`	12	รุ่นของ ONNX ที่ฝั่งแอปรองรับ
`--cache`	ram	อ่านภาพเข้าหน่วยความจำ เร็วขึ้นชัดเจน

ตารางที่ 5 ค่าตั้งต้นของการเทรน จาก train.py

> อ่านก่อนเทรนครั้งแรก ค่าตั้งต้นของตัวเลือก `--data` ใน `train.py` ชี้ไปที่ `data/datasets/current/data.yaml` ซึ่งเป็นไฟล์ที่ประกาศไว้เพียงคลาสเดียว การรัน `python train.py` เปล่า ๆ จึงเป็นการเทรนโมเดลสองคลาสด้วยพื้นที่ป้ายกำกับคลาสเดียว ผลคือกล่องลูกกอล์ฟทั้งหมดถูกย้ายไปเป็นคลาส 0 คือหัวไม้ โดยไม่มีข้อความเตือนใด ๆ ความผิดพลาดนี้เกิดขึ้นจริงแล้วหนึ่งครั้ง สคริปต์ `scripts/train.sh` มีอยู่เพื่อเรื่องนี้เรื่องเดียว คือส่ง `--data` ให้ชัดเจนทุกครั้ง

7. การตรวจผลหลังเทรน

ไม่มีการเลื่อนโมเดลขึ้นใช้งานอัตโนมัติ ทุกรอบการเทรนจะพิมพ์ตัวเลข val ของโมเดลใหม่วางข้างตัวเลขของโมเดลพื้นฐานบนชุดแบ่งเดียวกัน แล้วให้คนตัดสินว่าดีขึ้นจริงหรือไม่ การออกแบบเช่นนี้ตั้งใจไว้ตั้งแต่ต้น นอกจากตัวเลข val ยังมีการตรวจอีกสองชั้นที่ควรรันทุกครั้ง

- `scripts/eval_clips.py` — ลองโมเดลกับคลิปที่ไม่เคยอยู่ในชุดเทรน ค่าตั้งต้นคือหลุม H2G1, H4G1, H4G2 และ H8G1 ตัวเลข val ดีขึ้นแต่คลิปจริงแย่ลง เป็นสัญญาณที่ต้องหยุดพิจารณา
- `scripts/score_labels.py` — ให้คะแนนโมเดลกับเฟรมที่คนวาดไว้ เทียบกันระหว่างโมเดลเก่ากับใหม่
- `scripts/after_train.sh` — รอให้การเทรนจบแล้วรันทั้งสองอย่างข้างบนให้เอง พร้อมสรุปตัวเลขไว้ในไฟล์เดียว

เกณฑ์การตัดสินใจที่แนะนำ ให้ดู recall ของลูกกอล์ฟเป็นหลัก และดูผลจากคลิปที่ไม่เคยเห็นเป็นตัวยืนยัน อย่าตัดสินจาก mAP อย่างเดียว และอย่าตัดสินจากส่วนต่างที่เล็กกว่าความผันผวนของชุดประเมิน รายละเอียดเรื่องนี้อยู่ในหัวข้อถัดไป

8. กับดักที่โครงการนี้เคยเสียเวลาไปกับมันจริง

หัวข้อนี้ไม่ใช่ทฤษฎี ทุกข้อคือเหตุการณ์ที่เกิดขึ้นแล้ว หรือเป็นสภาพที่ยังค้างอยู่ในระบบและพร้อมจะทำให้พลาดซ้ำ

เรื่อง	สภาพปัจจุบัน	สิ่งที่ต้องทำ
ไฟล์กำหนดชุดข้อมูลที่มีคลาสเดียว ทำให้ลูกกลายเป็นหัวไม้	ยังอยู่ในระบบ และเป็นค่าตั้งต้นของ `train.py` เคยทำให้เทรนเสียเปล่าไปหนึ่งชั่วโมง โดยชุด val ของรอบนั้นมีกล่องหัวไม้ 47 กล่องและไม่มีลูกเลย	ส่ง `--data` ให้ชัดเจนทุกครั้ง หรือใช้ `scripts/train.sh` และควรแก้ค่าตั้งต้นในต้นทางให้ไม่ชี้ไปที่ไฟล์นั้น
ชุดข้อมูล `merged` ที่ใช้เทรนโมเดลที่ส่งมอบ มีชุด val เพียง 53 ภาพ จาก 2 คลิป	ยังเป็นอย่างนั้น ส่วนต่างที่รายงานในรอบล่าสุดอยู่ระดับ 0.0004 ถึง 0.008 ซึ่งเล็กกว่าความผันผวนที่ชุดขนาดนี้ให้ได้	ใช้ชุด `clean` ที่ส่งมอบครั้งนี้แทน ซึ่งมี val 791 ภาพ และอย่าสรุปผลจากส่วนต่างระดับพันส่วน
คลิปซ้ำในฐานข้อมูล H8G1 อยู่ฝั่ง train และ H8G1 copy อยู่ฝั่ง val จำนวนเฟรมเท่ากันคือ 273	ตรวจแล้วว่ายังไม่หลุดเข้าไปในชุด `clean` หรือ `merged` จึงยังไม่กระทบโมเดลที่มีอยู่	ลบคลิปที่ซ้ำออกจากฐานข้อมูลก่อนสร้างชุดข้อมูลใหม่ ไม่เช่นนั้นการแบ่งครั้งถัดไปจะทำให้คลิปเดียวกันอยู่ทั้งสองฝั่ง และตัวเลข val จะวัดการจำ
เฟรม 62 เฟรมที่ยังไม่ label อยู่ในคลิป H8G1 copy ทั้งหมด	เป็นงานที่ไม่ควรทำ เพราะเป็นสำเนา	ลบสำเนาออก แล้วความคืบหน้าจะเป็น 100 เปอร์เซ็นต์ทันทีโดยไม่ต้อง label เพิ่ม
คลิป h6g1_3 และ h6g1_3 copy อยู่ฝั่ง train ทั้งคู่ จำนวนเฟรมเท่ากันคือ 350	ไม่ทำให้ข้อมูลรั่วข้ามชุด แต่ทำให้คลิปนั้นมีน้ำหนักเป็นสองเท่าในการเทรน	ลบสำเนาออกเช่นกัน
อัตราการรีวิวซ้ำ 0.9 เปอร์เซ็นต์	ป้ายกำกับเกือบทั้งหมดผ่านสายตาคนเดียว	สุ่มรีวิวซ้ำอย่างน้อยในชุด val และ test ซึ่งเป็นชุดที่ความผิดพลาดหนึ่งจุดกระทบตัวเลขมากที่สุด

ตารางที่ 6 กับดักที่ตรวจพบ พร้อมสถานะและวิธีเลี่ยง

สาระร่วมของทุกข้อข้างบน ความผิดพลาดของชุดข้อมูลไม่ส่งเสียงเตือน มันไม่ทำให้โปรแกรมพัง แต่ทำให้ตัวเลขดูดีขึ้นหรือแย่ลงโดยไม่มีเหตุผลที่แท้จริง นี่คือเหตุผลที่ `verify_dataset.py` ถูกเขียนขึ้น และควรรันก่อนเทรนทุกครั้ง

9. การสร้างชุดข้อมูลใหม่

เมื่อมีการ label เพิ่ม หรือแก้ป้ายกำกับเดิม ต้องสร้างชุดข้อมูลใหม่ ไม่ใช่แก้ไฟล์ในโฟลเดอร์ `dataset` โดยตรง เพราะโฟลเดอร์นั้นเป็นผลลัพธ์ ไม่ใช่แหล่งข้อมูล แหล่งข้อมูลจริงคือ `labelhub.db` ขั้นตอนคือรัน `scripts/build_dataset.py` ซึ่งจะอ่านฐานข้อมูล คัดเฉพาะเฟรมที่มีคนยืนยันแล้ว แบ่งชุดตามคลิป ตรวจเกณฑ์จำนวนลูกในชุดประเมิน แล้วเขียนโฟลเดอร์ชุดข้อมูลใหม่พร้อมใบสรุป

- ตัวเครื่องมือจะไม่เขียนทับของเดิมถ้าระบุ `--out` เป็นชื่ออื่น และมีโหมด `--dry-run` ให้ดูผลก่อน
- หลังสร้างเสร็จ ต้องแก้บรรทัด `path:` ใน `data.yaml` ให้ชี้ที่เครื่องที่จะเทรน เพราะ Ultralytics อ้างอิงตำแหน่งของ train, val และ test จากบรรทัดนี้
- จากนั้นรัน `verify_dataset.py` ทับอีกครั้งก่อนเทรน

> ข้อควรรู้เรื่องรูปภาพ ชุดข้อมูลบนเครื่องต้นทางเก็บภาพเป็นทางลัดเพื่อไม่ให้เปลืองเนื้อที่ ถ้าจะย้ายชุดข้อมูลข้ามเครื่องอีกครั้ง ต้องคลายทางลัดให้เป็นไฟล์จริง เช่นด้วย `tar --dereference` หรือ `rsync -L` มิฉะนั้นจะได้ทางลัดที่ชี้ไปในที่ที่ไม่มีอยู่ และจะรู้ตัวตอนเทรนไปแล้ว

10. ข้อจำกัดของเอกสารและงานที่ยังไม่ได้ทำ

เพื่อให้ผู้อ่านประเมินความน่าเชื่อถือของแต่ละส่วนได้เอง ส่วนนี้ระบุสิ่งที่ยังไม่ได้ตรวจและสิ่งที่ยังไม่ได้ทำ

- ยังไม่ได้ทดลองเทรนบนเครื่องนี้ ชุดข้อมูลผ่านการตรวจความครบถ้วนแล้ว แต่การเทรนจริงยังไม่ได้รัน จึงยังไม่ยืนยันว่าเครื่องนี้มีสภาพแวดล้อมครบ ต้องมี Ultralytics 8.4.106 และ PyTorch 2.13.0 ตามที่เครื่องต้นทางใช้
- ตัวเลขผลของโมเดลทั้งหมดคัดลอกมาจากไฟล์ที่ระบบเขียนไว้ ไม่ได้รันวัดซ้ำเพื่อยืนยัน
- ยังไม่ได้ลบคลิปซ้ำในฐานข้อมูล เพราะเป็นการแก้ข้อมูลบนเครื่องที่ให้บริการอยู่ ต้องได้รับการยืนยันก่อน
- ยังไม่ได้แก้ค่าตั้งต้น `--data` ใน `train.py` ที่ต้นทาง เอกสารนี้จึงใช้วิธีห่อด้วย `train.sh` แทน ซึ่งแก้ที่ปลายทางไม่ใช่ที่ต้นเหตุ

- จำนวนกล่องที่โมเดลเสนอและยังไม่มีคนยืนยัน 125 กล่อง ยังไม่ได้ตรวจว่าถูกนับรวมเข้าชุดข้อมูลหรือไม่ในทุกกรณี

ข้อแนะนำลำดับงานถัดไป หนึ่ง ลบคลิปซ้ำสองคลิปออกจากฐานข้อมูล สอง แก้ค่าตั้งต้นของ `--data` ที่ต้นทางให้ไม่ชี้ไปที่ไฟล์คลาสเดียว สาม สร้างชุดข้อมูลใหม่แล้วเทรนเทียบกับตัวเลขในหัวข้อที่ 5 โดยดู recall ของลูกกอล์ฟเป็นตัวตัดสิน

เอกสารอ้างอิง

[1] ชุดข้อมูลต้นทาง: labelhub บนเครื่อง go-ai · /Users/siammacstudio/labelhub/data/datasets/clean

[2] ฐานข้อมูลการ label: labelhub.db · ตาราง clips, frames, boxes, jobs, landings

[3] เครื่องมือสร้างชุดข้อมูล: build_dataset.py · เกณฑ์ MIN_BALLS_PER_EVAL_SPLIT = 60

[4] เครื่องมือเทรน: train.py · Ultralytics 8.4.106 · PyTorch 2.13.0 · imgsz 960 · ONNX opset 12

[5] ผลการวัดของโมเดล: models/yolo_birdiex_2c_20260728-004149.json