RWR 4013

Digital Twins for Smart Cities

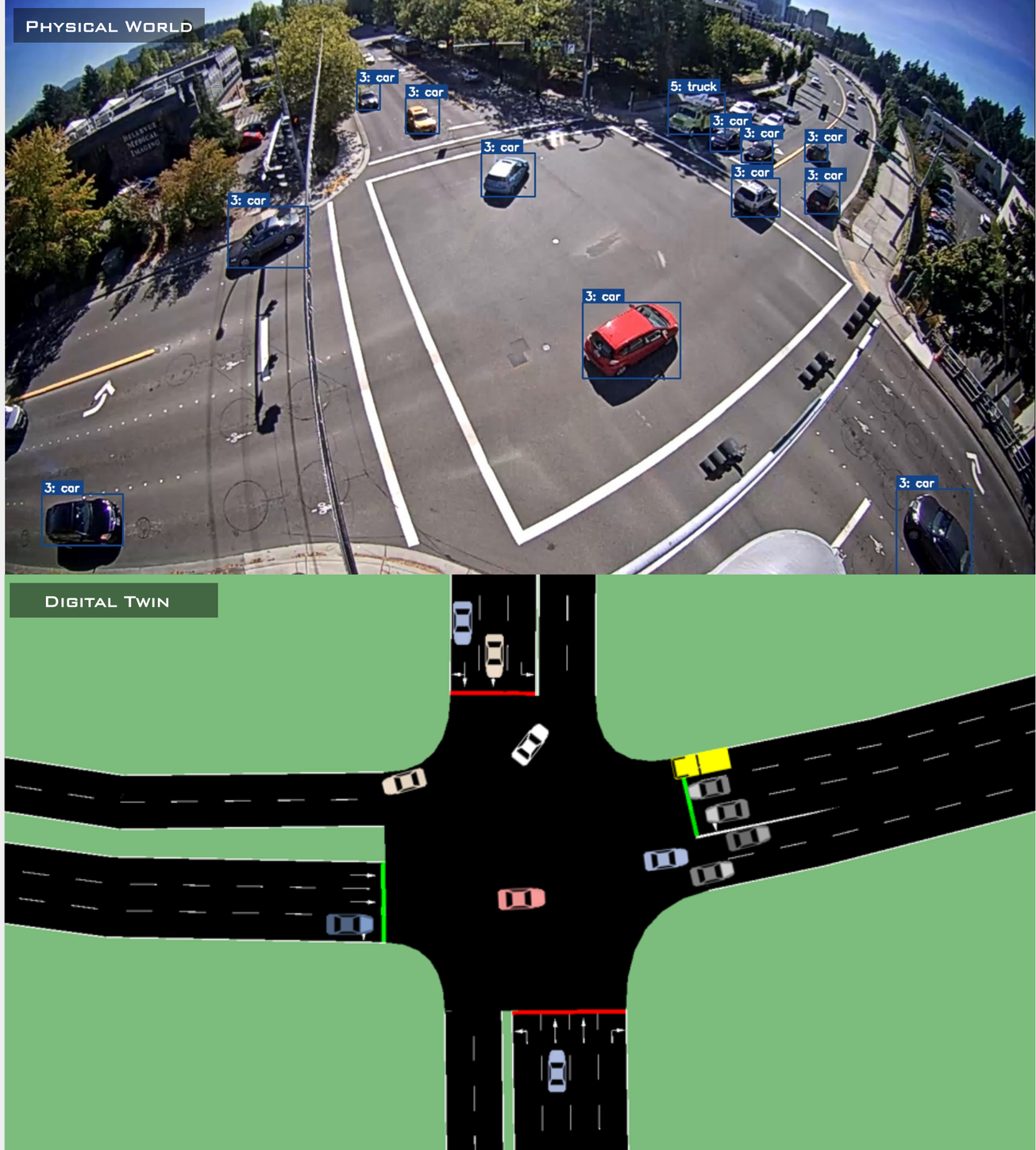
Dr. Ahmad Mohammadi

Week 2 | Session 2:
Computer Vision I
(Object Detection)

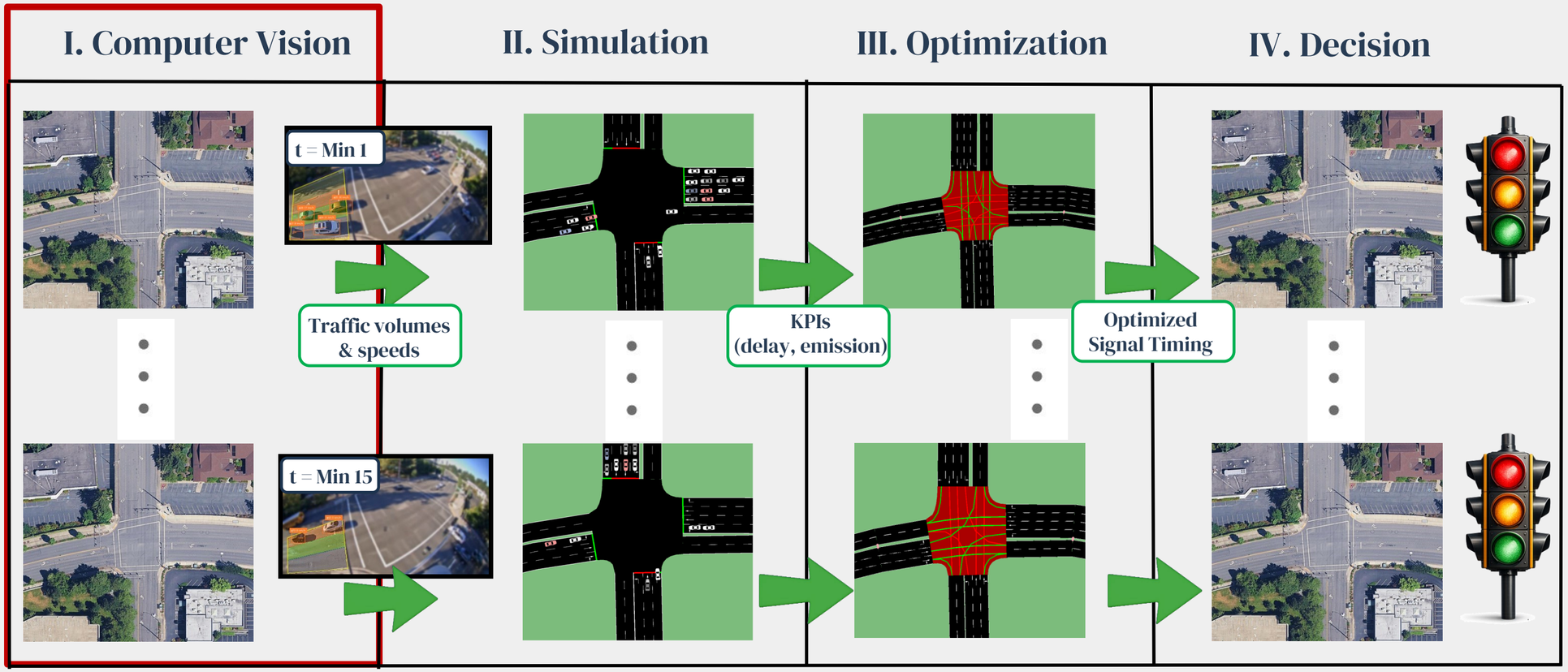
Fall 2026

RoadwayVR

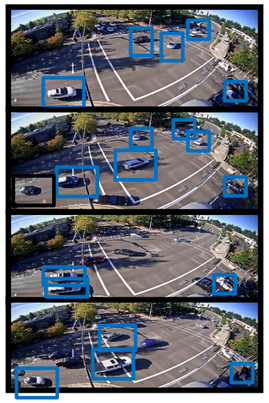
roadwayvr.github.io/DigitalTwinsforSmartCities



Overview of Course Syllabus in One Shot

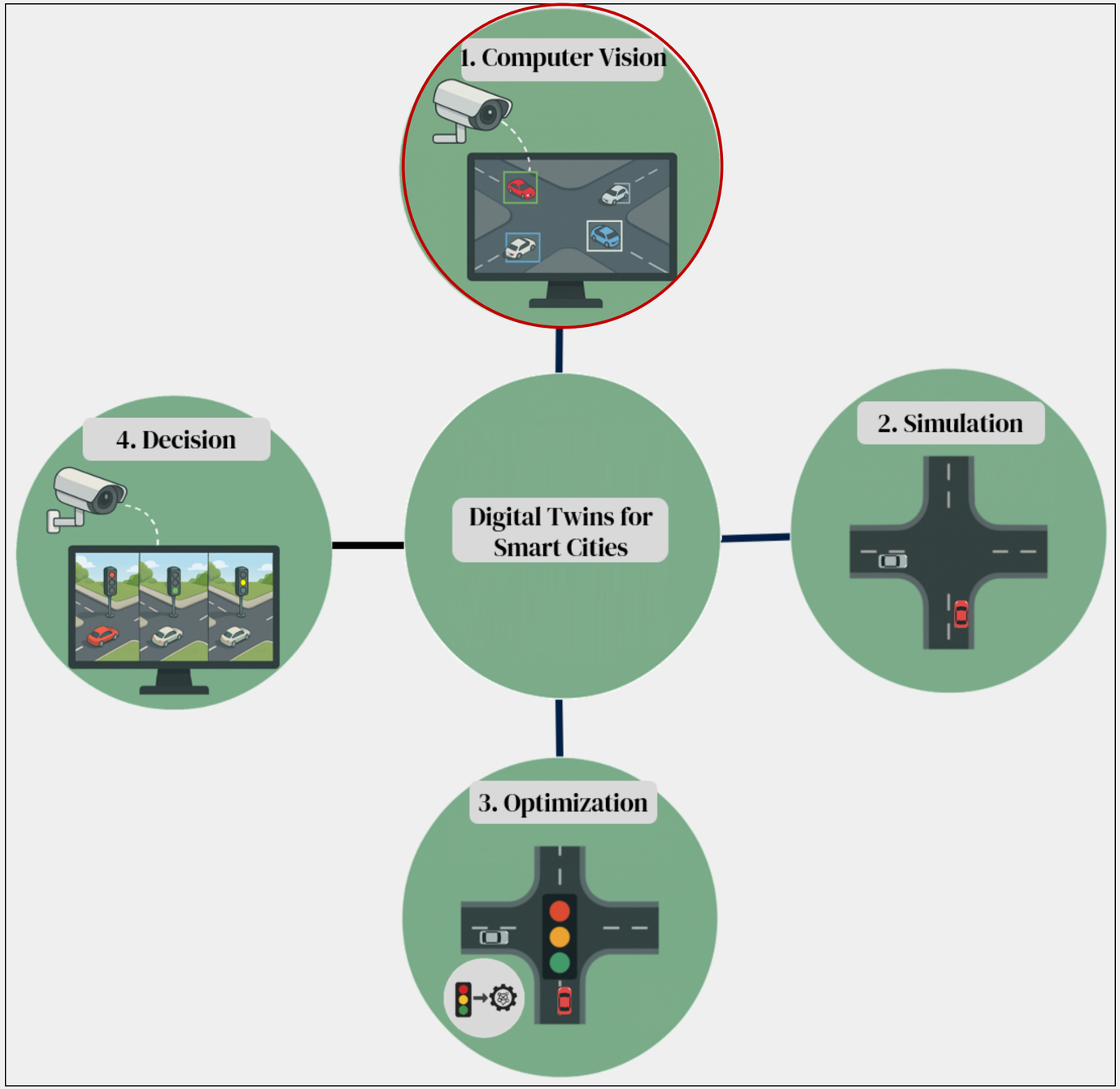


Recap: 6 Steps in Convolutional Neural Network for Object Detection

1 Input	2 Algorithm	3 Output
	**Algorithm: YOLO**	 **Class (car, bus, etc)** **Confidence score** **Bounding Box**
✅ Step 1: Study area and video recording ✅ Step 2: Frame extraction & dataset creation	Step 5: YOLO Trainer on Mini Datasets Step 6: Understanding the Result	✅ Step 3.Image annotation & class definition Step 4.Dataset partitioning (train validation test)

Agenda

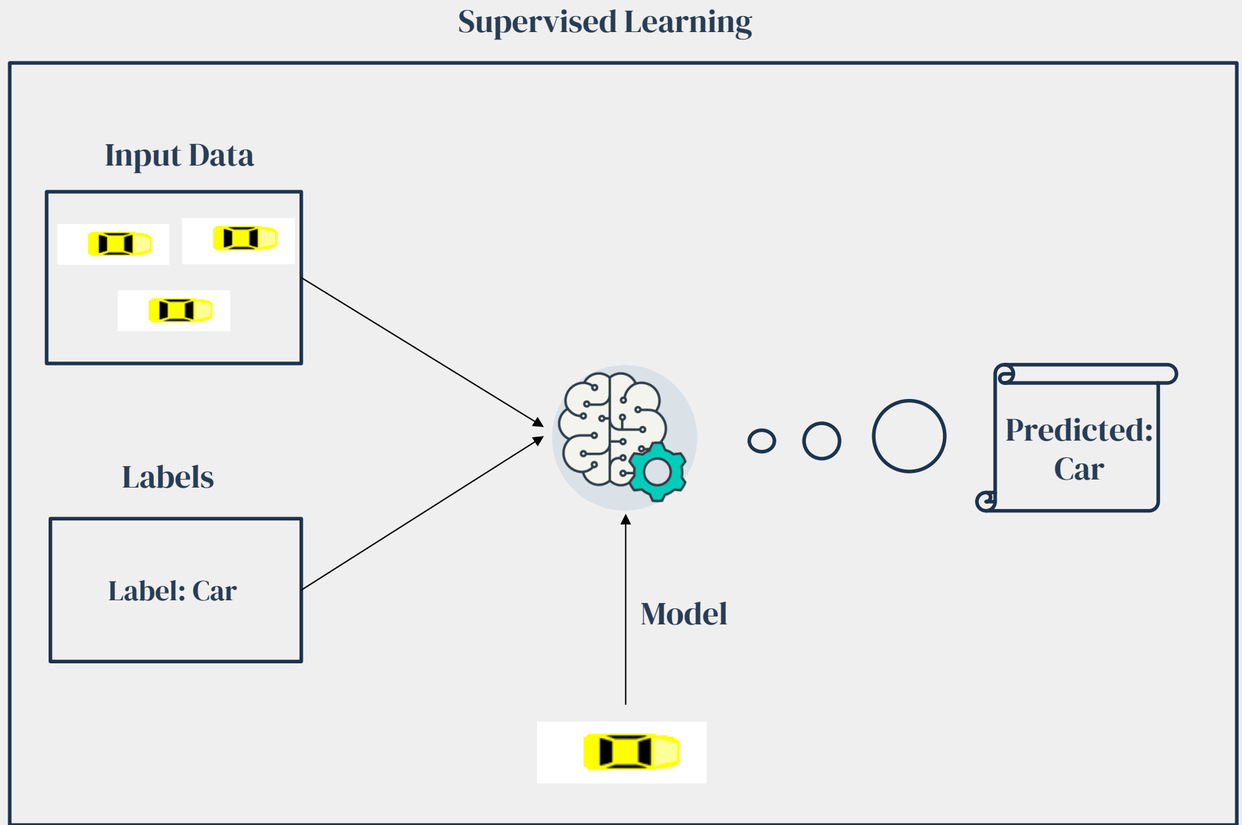
- ❑ Step 4: Dataset partitioning (Train Validation Test)
- ❑ Step 5. YOLO Trainer
- ❑ Step 6. Understanding The Result



Step 4: Dataset Partitioning (Train Validation Test)

Question: Why do we partition (split) the dataset?

- In AI (supervised learning), the model learns from labeled examples.
- To know if the model truly learned (and not just memorized), we test it on new images it has never seen.
- We split the data into **training**, **validation**, **test.**



Step 4: Dataset Partitioning (Train Validation Test)

1) Train set (learning)

- The model learns patterns from these **images + labels.**
- Usually, the largest portion of the dataset (e.g., **60%**).
- Purpose: **“Teach the model.”**

2) Validation set (tuning)

- Used during training to **monitor performance** and decide when to stop training (early stopping)
- The model does not train on the validation set (it is only evaluated on it)
- Purpose: **“Check progress”**

3) Test set (final evaluation)

- Used only after training and validation are finished.
- Provides an **unbiased final result** that best represents real-world performance.
- Purpose: **“Final exam.”**

Step 4: Dataset Partitioning (Train Validation Test)

Think of it like studying for an exam:

1) Train set (learning) = lectures + practice

You learn from it. The model updates its weights using these examples.

2) Validation set (tuning) = assignments / quizzes / midterm

You do not learn from it directly. You use it to check progress and tune decisions (e.g., early stopping)

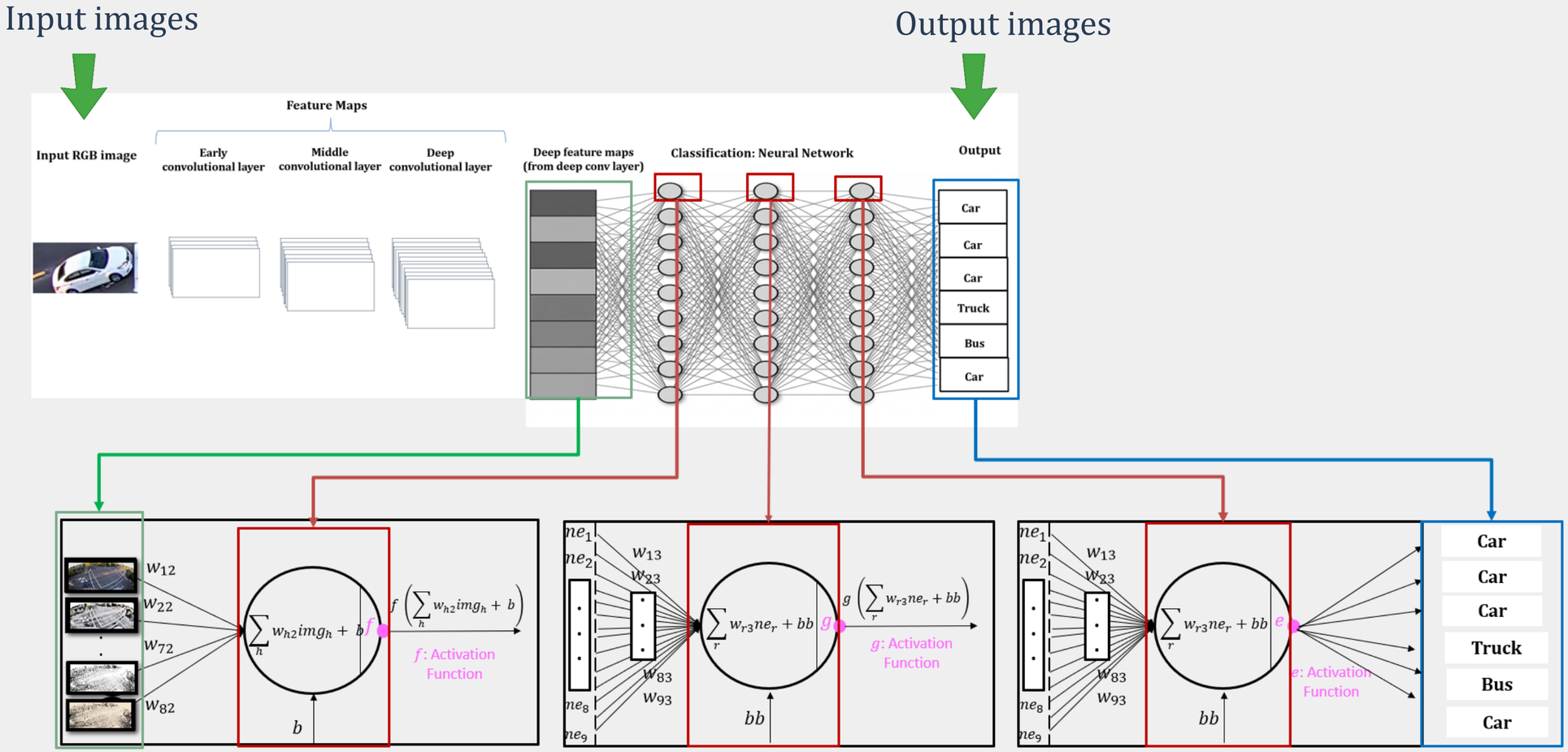
3) Test set (final evaluation)= final exam

You do not learn from it directly. You use what you learned during training (and tuned using validation) to perform

on completely unseen examples.

Step 4: Dataset Partitioning (Train Validation Test)

- **Training and validation:** By providing input images and output (through training and validation), the CNN tries to estimate and update these weights in different layers.
- **Test:** Uses completely unseen images to measure how well the trained model can predict.



Step 4: Dataset Partitioning (Train Validation Test)

1. Download Week2b.Material.zip and Extract It.
2. In Folder “4.Dataset partitioning (train validation test)”
3. Create two main sub-folders:
   - images
   - labels
4. Inside each sub-folder, create another three subfolders:

train

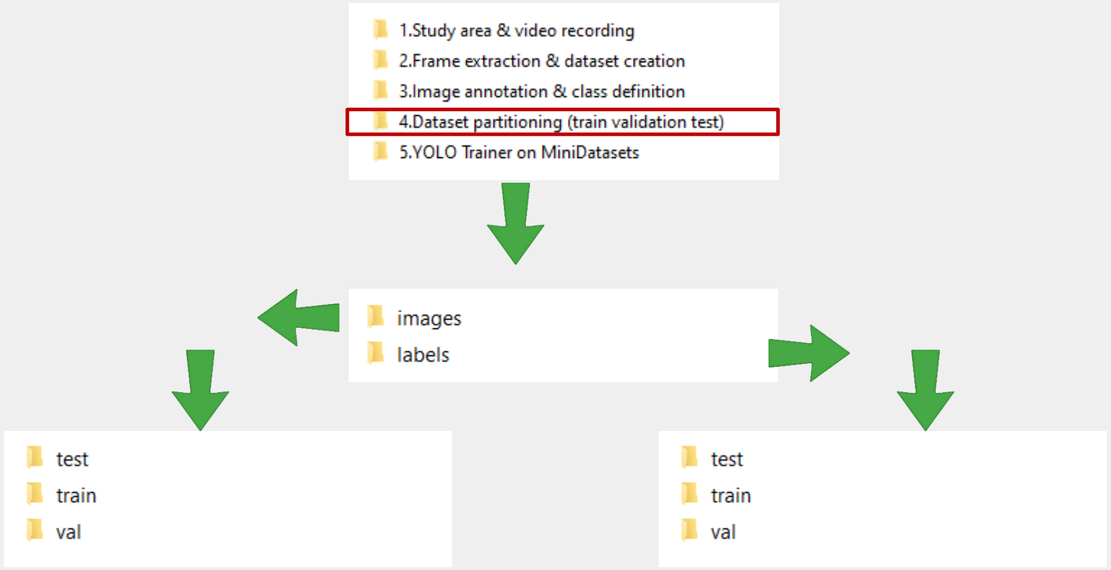
val

test

5. Final structure

images/train , images/val , images/test

labels/train , labels/val , labels/test



Step 4: Dataset Partitioning (Train Validation Test)

4. Using datasets in Folder “3.Image annotation & class definition”, Split the dataset into:

60% of images → images/train

20% of images → images/val

20% of images → images/test

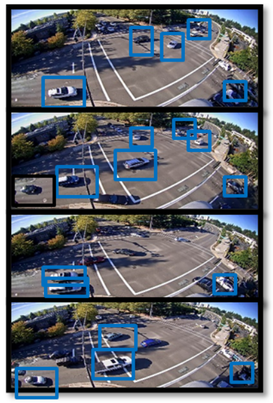
Move the corresponding label file for each image to the matching labels folder:

If images/train/12.jpg is in the training set, then labels/train/12.txt must also be in the training set.

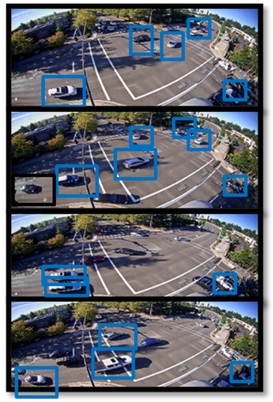
The same rule applies to val and test.

Rule: Every image and its .txt label must always stay in the same split (train/val/test).

Stage 3 Completed

1 Input	2 Algorithm	3 Output
	**Algorithm: YOLO**	 **Class (car, bus, etc)** **Confidence score** **Bounding Box**
Step 1: Study area and video recording Step 2: Frame extraction & dataset creation	Step 5: YOLO Trainer on Mini Datasets Step 6: Understanding the Result	Step 3.Image annotation & class definition Step 4.Dataset partitioning (train validation test)

Stage 2 In Progress

1 Input	2 Algorithm	3 Output
	**Algorithm: YOLO**	 **Class (car, bus, etc)** **Confidence score** **Bounding Box**
Step 1: Study area and video recording Step 2: Frame extraction & dataset creation	Step 5: YOLO Trainer on Mini Datasets Step 6: Understanding the Result	Step 3.Image annotation & class definition Step 4.Dataset partitioning (train validation test)

Step 5. YOLO Trainer on MiniDatasets

1. In Folder “5.YOLO Trainer on MiniDatasets”

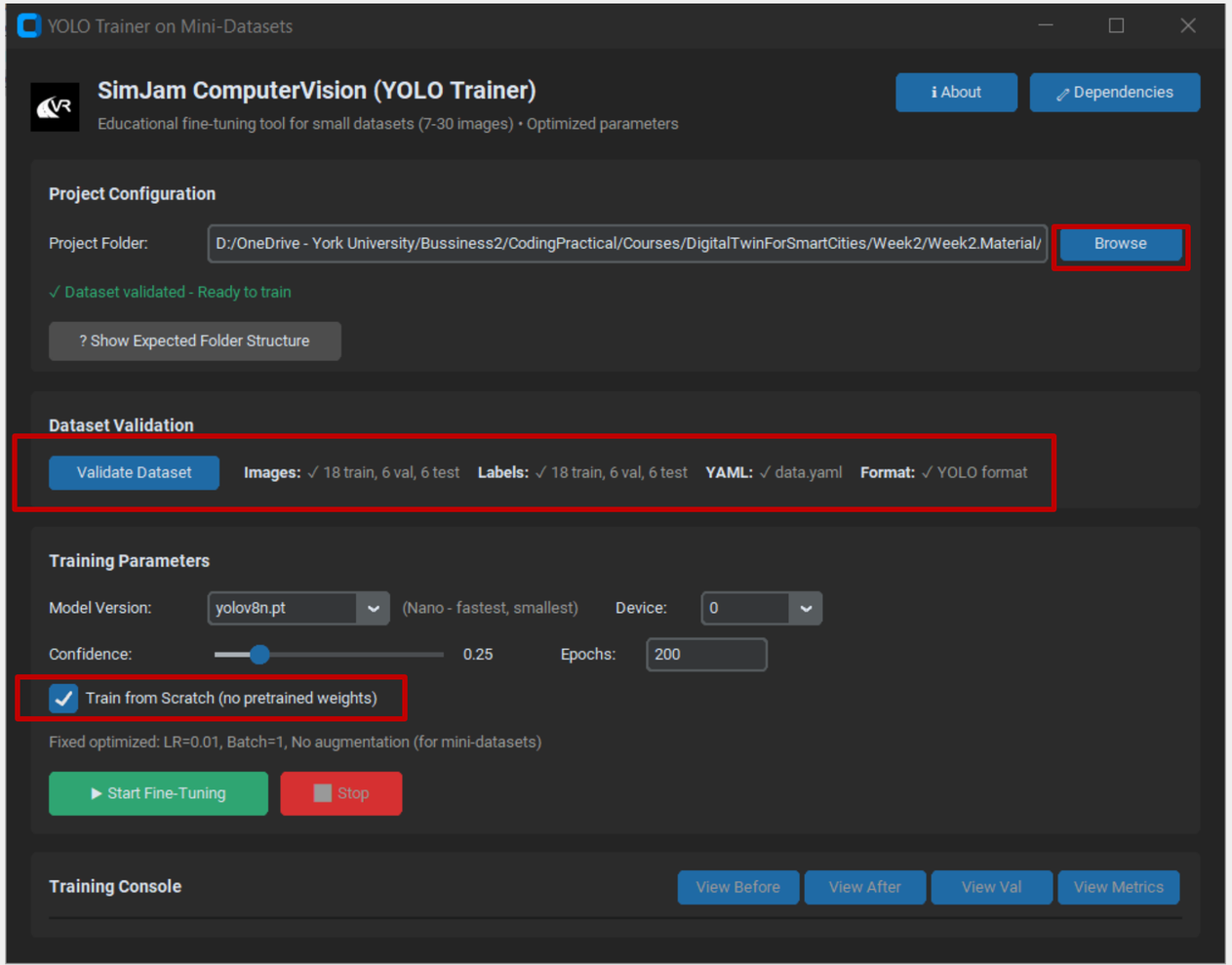
2. Copy and Paste folders “images” and “labels” from “4.Dataset partitioning (train validation test)” into “5. YOLO Trainer on MiniDatasets”

3. Run application “SimJamCVTrainer.py”

4. In Project Folder → Select “Browse” → Select Folder “5. YOLO Trainer on MiniDatasets”

5. Select “Validation Dataset” Button

6. Make Sure Button “Train From Scratch” is checked



Step 5. YOLO Trainer on MiniDatasets

7. Select ”Start Fine Tuning”
8. Observe Visual Studio Code

Epoch: which training round you’re in (e.g., 123/200 = epoch 123 out of 200).

GPU_mem: how much GPU memory is being used.

box_loss: error for bounding box location/size (lower is better).

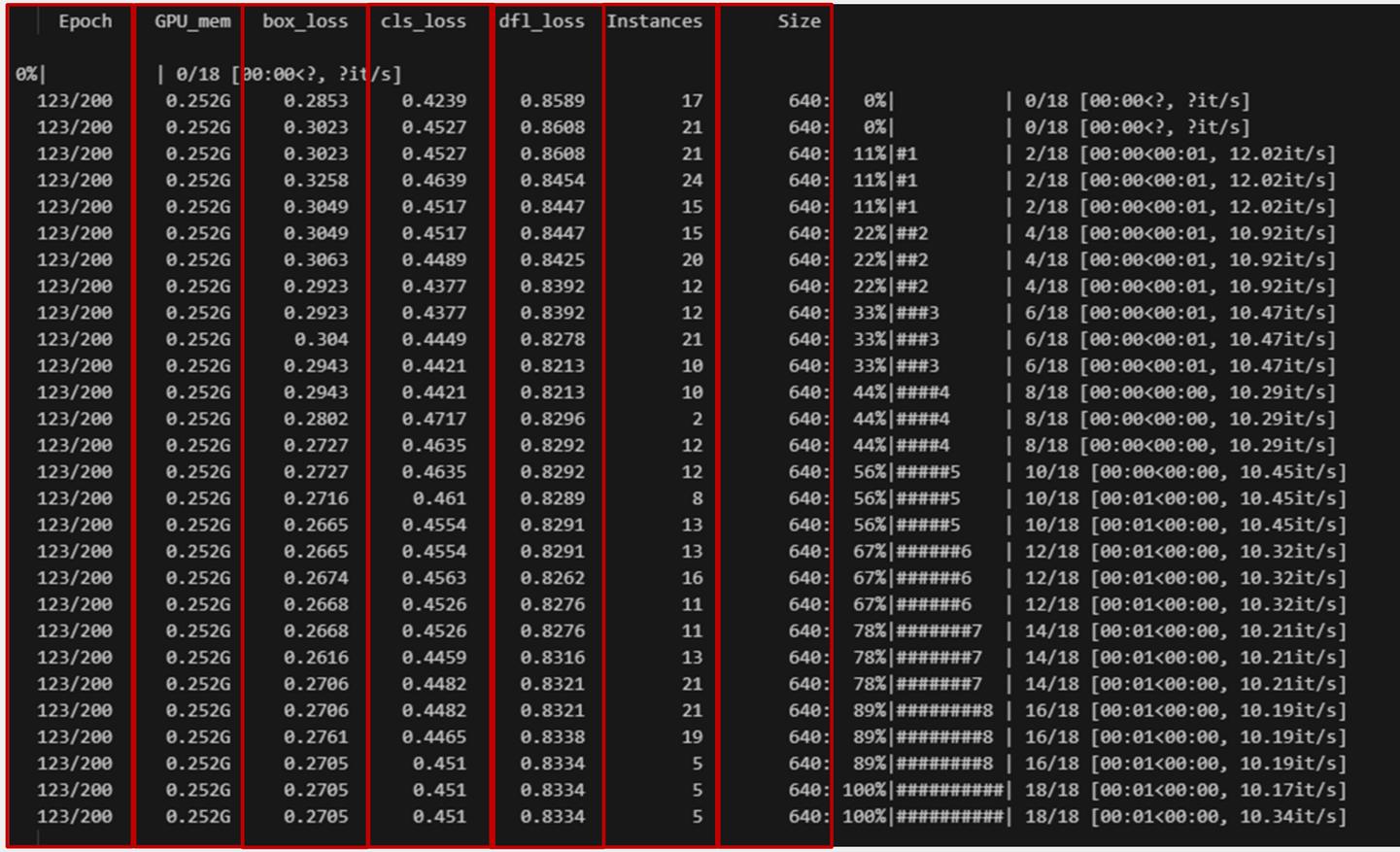
cls_loss: error for class prediction (lower is better).

dfl_loss: “Distribution Focal Loss” used in newer YOLOs for box quality (lower is better).

Instances: number of labeled objects in the current batch.

Size: input image size used for training (e.g., 640).

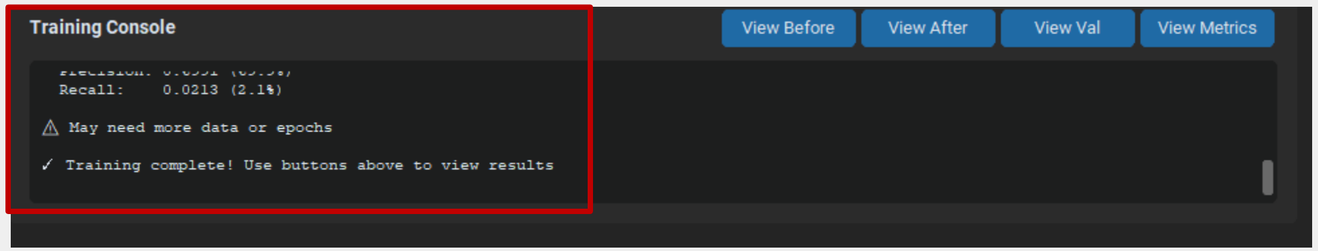
The progress bars / it/s: Training speed (iterations per second).



Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
0%|	| 0/18 [00:00<?, ?it/s]						
123/200	0.252G	0.2853	0.4239	0.8589	17	640:	0%| | 0/18 [00:00<?, ?it/s]
123/200	0.252G	0.3023	0.4527	0.8608	21	640:	0%| | 0/18 [00:00<?, ?it/s]
123/200	0.252G	0.3023	0.4527	0.8608	21	640:	11%|#1 | 2/18 [00:00<00:01, 12.02it/s]
123/200	0.252G	0.3258	0.4639	0.8454	24	640:	11%|#1 | 2/18 [00:00<00:01, 12.02it/s]
123/200	0.252G	0.3049	0.4517	0.8447	15	640:	11%|#1 | 2/18 [00:00<00:01, 12.02it/s]
123/200	0.252G	0.3049	0.4517	0.8447	15	640:	22%|##2 | 4/18 [00:00<00:01, 10.92it/s]
123/200	0.252G	0.3063	0.4489	0.8425	20	640:	22%|##2 | 4/18 [00:00<00:01, 10.92it/s]
123/200	0.252G	0.2923	0.4377	0.8392	12	640:	22%|##2 | 4/18 [00:00<00:01, 10.92it/s]
123/200	0.252G	0.2923	0.4377	0.8392	12	640:	33%|###3 | 6/18 [00:00<00:01, 10.47it/s]
123/200	0.252G	0.304	0.4449	0.8278	21	640:	33%|###3 | 6/18 [00:00<00:01, 10.47it/s]
123/200	0.252G	0.2943	0.4421	0.8213	10	640:	33%|###3 | 6/18 [00:00<00:01, 10.47it/s]
123/200	0.252G	0.2943	0.4421	0.8213	10	640:	44%|####4 | 8/18 [00:00<00:00, 10.29it/s]
123/200	0.252G	0.2802	0.4717	0.8296	2	640:	44%|####4 | 8/18 [00:00<00:00, 10.29it/s]
123/200	0.252G	0.2727	0.4635	0.8292	12	640:	44%|####4 | 8/18 [00:00<00:00, 10.29it/s]
123/200	0.252G	0.2727	0.4635	0.8292	12	640:	56%|#####5 | 10/18 [00:00<00:00, 10.45it/s]
123/200	0.252G	0.2716	0.461	0.8289	8	640:	56%|#####5 | 10/18 [00:01<00:00, 10.45it/s]
123/200	0.252G	0.2665	0.4554	0.8291	13	640:	56%|#####5 | 10/18 [00:01<00:00, 10.45it/s]
123/200	0.252G	0.2665	0.4554	0.8291	13	640:	67%|######6 | 12/18 [00:01<00:00, 10.32it/s]
123/200	0.252G	0.2674	0.4563	0.8262	16	640:	67%|######6 | 12/18 [00:01<00:00, 10.32it/s]
123/200	0.252G	0.2668	0.4526	0.8276	11	640:	67%|######6 | 12/18 [00:01<00:00, 10.32it/s]
123/200	0.252G	0.2668	0.4526	0.8276	11	640:	78%|#######7 | 14/18 [00:01<00:00, 10.21it/s]
123/200	0.252G	0.2616	0.4459	0.8316	13	640:	78%|#######7 | 14/18 [00:01<00:00, 10.21it/s]
123/200	0.252G	0.2706	0.4482	0.8321	21	640:	78%|#######7 | 14/18 [00:01<00:00, 10.21it/s]
123/200	0.252G	0.2706	0.4482	0.8321	21	640:	89%|########8 | 16/18 [00:01<00:00, 10.19it/s]
123/200	0.252G	0.2761	0.4465	0.8338	19	640:	89%|########8 | 16/18 [00:01<00:00, 10.19it/s]
123/200	0.252G	0.2705	0.451	0.8334	5	640:	89%|########8 | 16/18 [00:01<00:00, 10.19it/s]
123/200	0.252G	0.2705	0.451	0.8334	5	640:	100%|##########| 18/18 [00:01<00:00, 10.17it/s]
123/200	0.252G	0.2705	0.451	0.8334	5	640:	100%|##########| 18/18 [00:01<00:00, 10.34it/s]

Step 5. YOLO Trainer on MiniDatasets

9. When you see “Training complete! in the console → See next slide



Step 5. YOLO Trainer on MiniDatasets

10. Open Folder “test Before training” and “test after training”

- **This example shows the CV is learning but it misses almost all of the cars due to low sample sizes**
- **We need more datasets**

Test before Training

Test after Training

Train the Model on an Existing Model

(Existing YOLO V8)

Step 5. YOLO Trainer on MiniDatasets

11. Close application

12. Remove Folders “test_after_training” & “test_before_training” &

“val_after_training” & “finetuned_model”

13. Run SimJamCVTrainer.py again

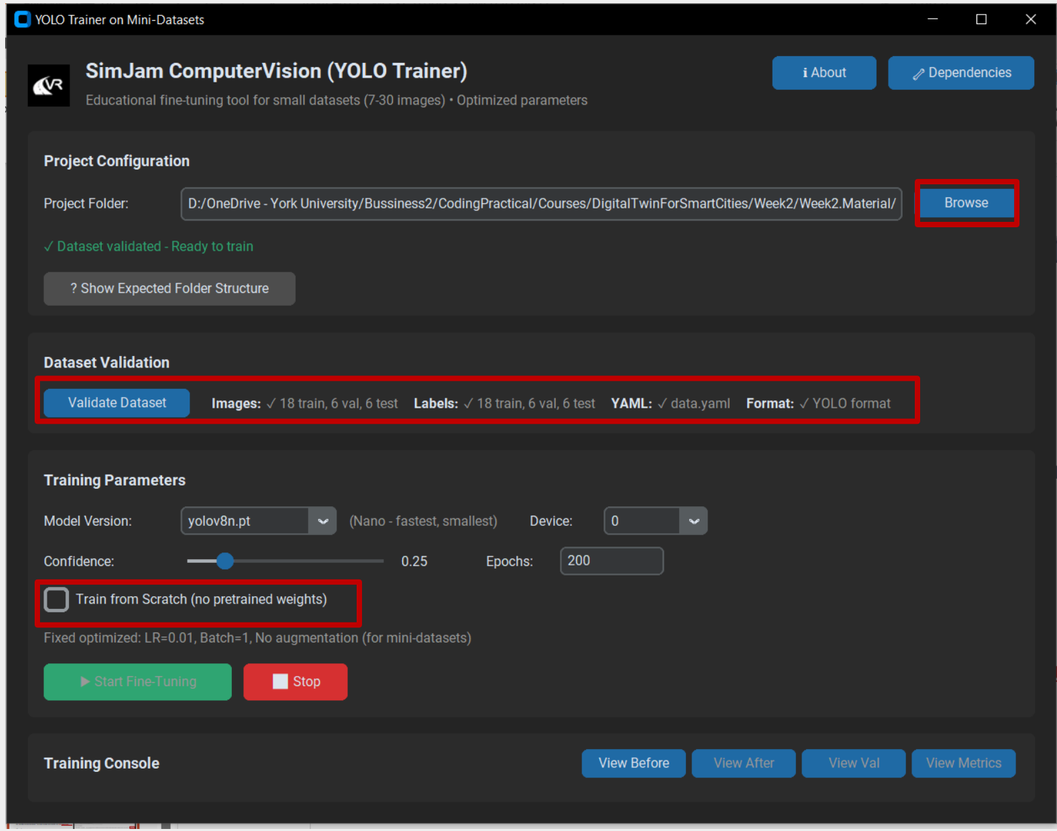
14. In Project Folder → Select “Browse” → Select Folder “5. YOLO Trainer

on MiniDatasets”

15. Select “Validate Dataset” Button

16. Make Sure Button “Train From Scratch” is unchecked

17. Select ”Start Fine Tuning”



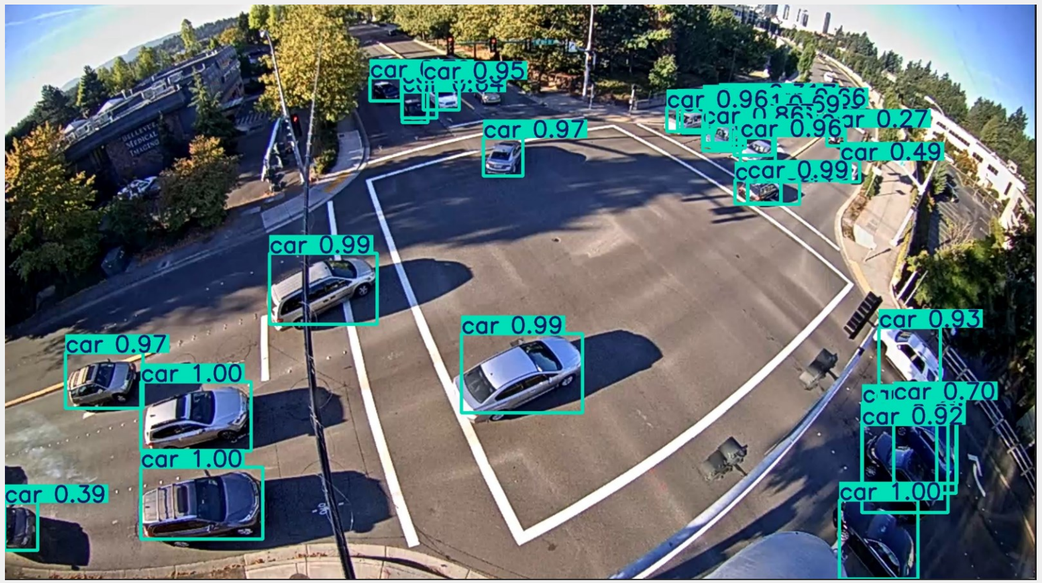
Step 5. YOLO Trainer on MiniDatasets

17. Open Folder “test before training” and “test after training”

- This example shows the CV is now performing much better than when we trained the model from scratch
- **Reason:** We trained the model on an existing YOLO v8 model

Test before Training (YOLO v8)

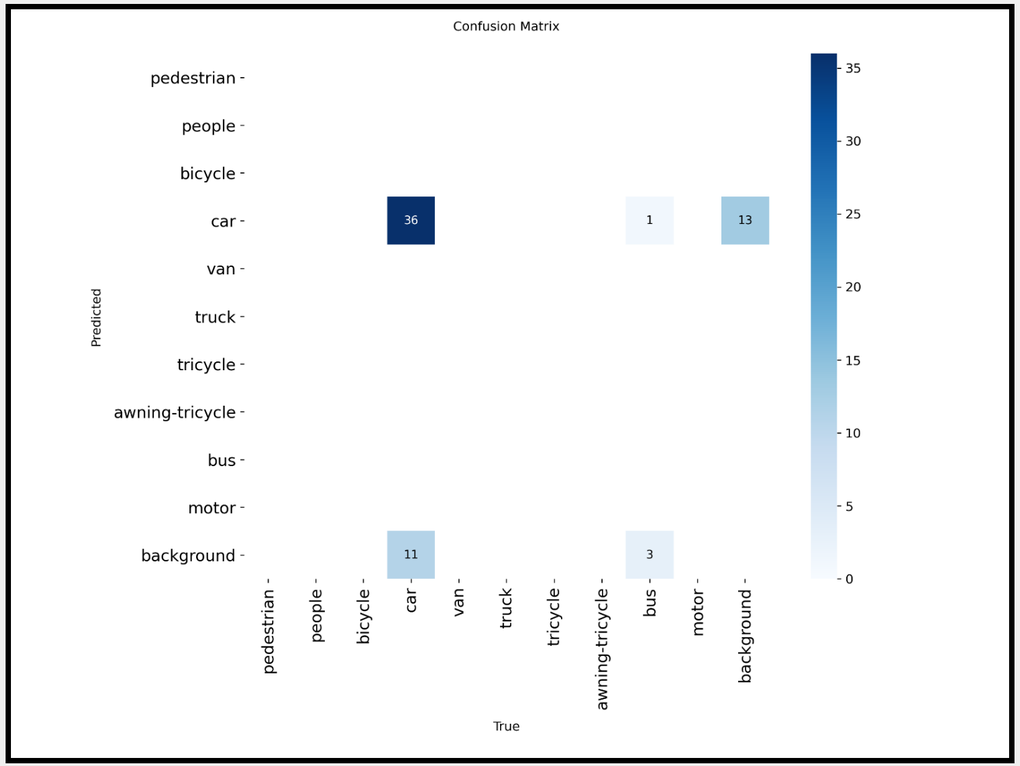
Test after Training (YOLO v8 + Our small Dataset)



Step 6. Understanding The Result

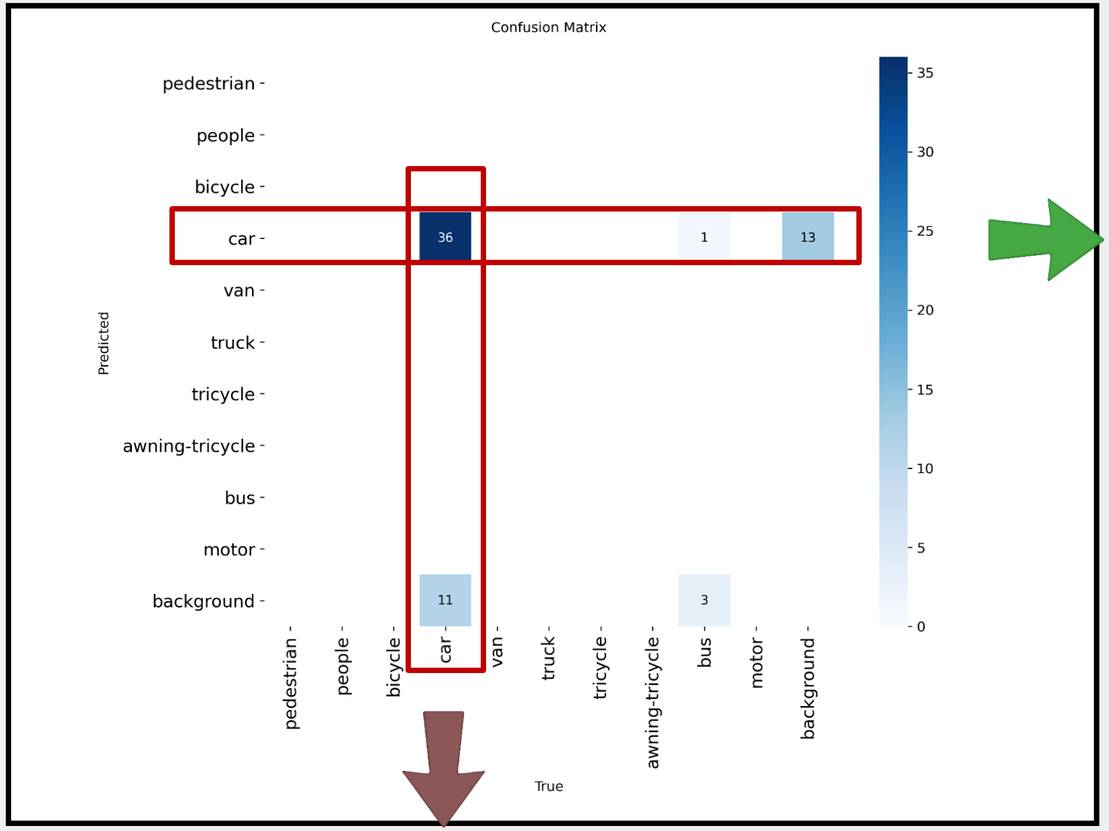
1. Open Folder “finetuned_model → open “confusion_matrix.png”

➢ How do you interpret this result?



Step 6. Understanding The Result

➢ **How do you interpret this result?**



Precision: (36)/ (36+14) = 72 %

When the model draws a box, how often is it car?

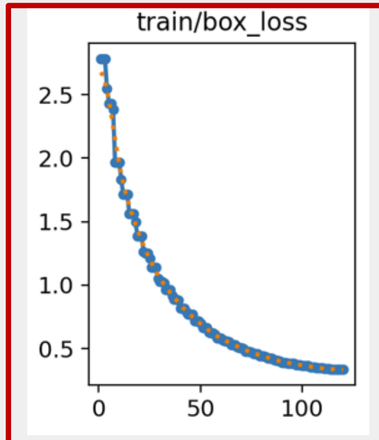
Recall: (36)/ (36+11) = 76 %

Out of all cars, how many did the model find?

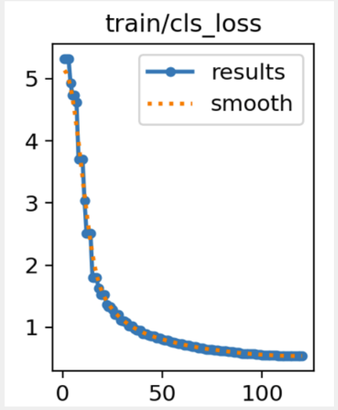
Step 6. Understanding The Result

2. Open Folder “finetuned_model → open “results.png”

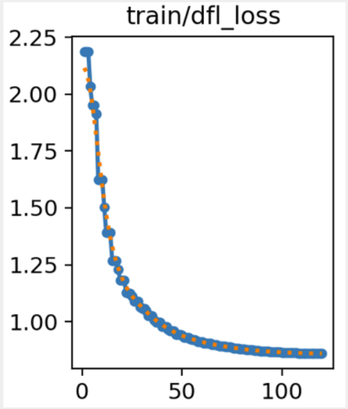
➢ Listen to Course Instructor to Interpret this



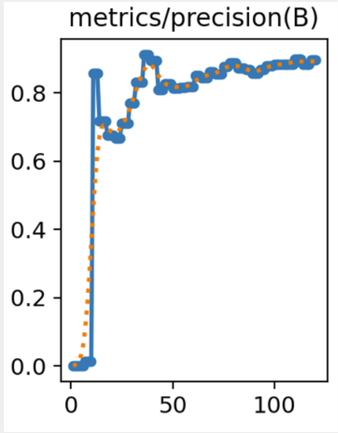
1.train/box_loss
How wrong the **predicted bounding box positions** are on the **training images** (lower = better).



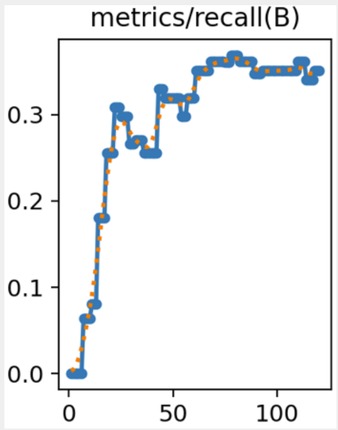
2.train/cls_loss
How wrong the predicted **class labels** are on the **training images** (lower = better).



3. train/dfl_loss
“Distribution Focal Loss” used by YOLO for **more accurate box edges** (lower = better)



4. metrics/precision(B)
When the model draws a box, how often is it right?
(higher = better - fewer wrong boxes)

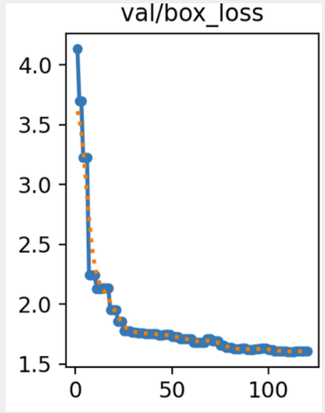


5. metrics/recall(B)
Out of all real objects, how many did the model find?
(higher = fewer missed objects.).

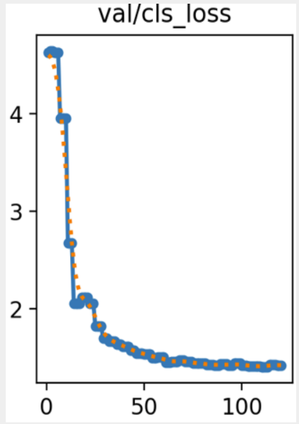
Step 6. Understanding The Result

2. Open Folder “finetuned_model → open “results.png”

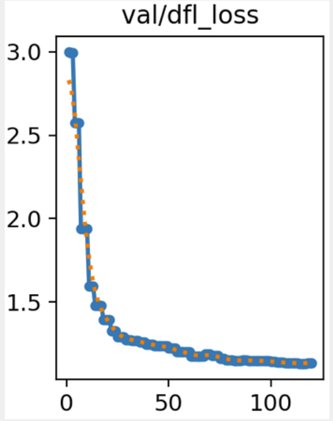
➢ Listen to Course Instructor to Interpret this



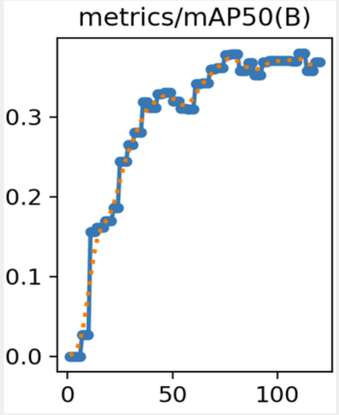
6. val/box_loss
Bounding-box localization error on the **validation set** (checks how well the model generalizes).



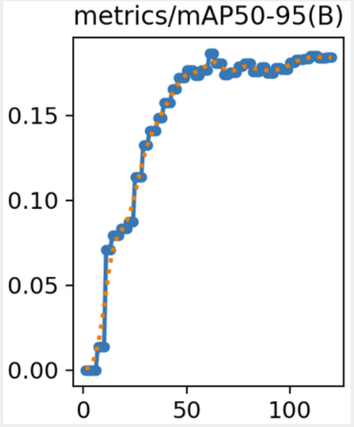
7. val/cls_loss
Class prediction error on validation set (generalization of classification).



8. val/dfl_loss
Box edge/shape quality loss on validation set (generalization of box precision).

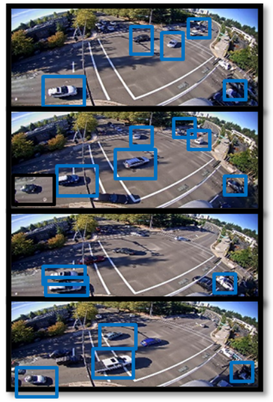


9.metrics/mAP50(B)
Overall detection accuracy when we count a prediction as correct if the box overlaps the real object by at least 50%.



10. metrics/mAP50-95(B)
mAP averaged from IoU 0.50 to 0.95 (stricter and more realistic; higher = better, but usually lower than mAP50).

All Stages are Completed

1 Input	2 Algorithm	3 Output
	**Algorithm: YOLO**	 **Class (car, bus, etc)** **Confidence score** **Bounding Box**
Step 1: Study area and video recording Step 2: Frame extraction & dataset creation	Step 5: YOLO Trainer on Mini Datasets Step 6: Understanding the Result	Step 3.Image annotation & class definition Step 4.Dataset partitioning (train validation test)