IFS315 - 2025

Machine Learning-based Vision System for Crop Identification

Alyssa Jordan Krishna

4308998

IFS315 - 2025

Table of Contents

Introduction

Crop classification and detection play an important role in modern agriculture, helping to improve productivity, reduce waste, and support precision farming. This project focuses on developing a machine learning-based vision system that can identify multiple crop types using image data. The goal is to build a system that performs reliably despite variations in crop growth stages, health, and environmental conditions.

I used YOLOv5, a real-time object detection model, because it offers a good balance between speed and accuracy. The model was trained on a custom dataset containing 11 crop types: apple, banana, carrot, corn, grapes, kiwi, lettuce, onion, pineapple, potato, and tomato. Due to time constraints, the number of crops was limited, but the model still demonstrated the ability to detect different growth stages and signs of poor plant health across the selected crops.

Problem statement

Accurate real-time crop identification is a major challenge in precision agriculture. Crops can look very different at each stage of their growth, and this makes it hard to recognize them consistently. On top of that, field conditions introduce extra problems like shadows, uneven lighting, background clutter such as soil or weeds, and crops being partly covered by leaves or nearby plants. These factors make it difficult for standard computer vision models to perform well outside of controlled environments.

This project aims to develop a deep learning-based object detection system using YOLOv5 that can identify multiple crop types in real time. By training the model on annotated images and applying transfer learning, the goal is to create a system that works reliably even in unpredictable conditions. This could help make agricultural processes like monitoring and sorting more efficient and less dependent on manual work.

Objectives

1. Collect and preprocess a multi-class dataset for crop detection
2. Train a YOLOv5 object detection model via command line
3. Monitor performance using TensorBoard visualizations
4. Test real-time detection of carrot, kiwi, pineapple, tomato and lettuce in the video presentation
5. Address real-world use cases, system limitations and ethical concerns

Data Collection and preprocessing

1.1. Dataset

The dataset consists of labeled images divided into training, validation, and testing subsets. The dataset includes the following 11 crop classes: *['apple', 'banana', 'carrot', 'corn', 'grapes', 'kiwi', 'lettuce', 'onion', 'pineapple', 'potato', 'tomato'].* Images are stored under

C:/VBS/raw_data/images with folders for train, val, and test. Each image is annotated with bounding boxes and class labels in YOLO format.

1.2. Multi-phase dataset division

Images were resized to 640×640 pixels and format conversion annotations (.txt) to match the YOLOv5 input requirements. Annotation files accompany each image, specifying object coordinates normalized relative to image size using MakesSense.ai. Augmentation was applied for transformations such as flips, HSV distortion and mosaic loading. The dataset contains 11 classes, ensuring diversity in crop types and environmental backgrounds.

Splitting under folders:

- Training: 70%
- Validation: 20%
- Testing: 10%

This setup ensured both model generalization and rigorous testing.

1.3. Training setup

- **Structure**: Images were stored in class-labelled folders under raw_data, with annotations stored in YOLO format under labels.
- **Image size:** 640 pixels
- **Batch size:** 16
- **Epochs:** 100
- **Optimizer:** SGD
- **Data augmentation:** enables (mosaic, flips, colour jitter)
- **Hardware:** CPU (no GPU available)

Training was run via command line interface using the provided train.py script, monitoring training loss and mAP metrics.

Model Development

Refer to Appendix A: Execution Instructions *for exact training, evaluation, and inference commands used during the implementation.*

The model was built using the YOLOv5(s) architecture due to its real-time object detection performance, time constraints and ease of use via the command line. Key choices included:

- Framework: PyTorch via YOLOv5
- Architecture: YOLOv5s (lightweight, accurate for edge cases)
- Transfer Learning: pre-trained weights on COCO were used and fine-tuned on the crop_data.
- Data Augmentation: horizontal flips, brightness changes and random crops improved generalization.

Training was performed for 100 epochs and the model converged well with no signs of overfitting.

Refer to Appendix D: Tech Stack

Model Evaluation

The model trained over 100 epochs, achieving an overall mean Average Precision (mAP50) of approximately 58.6% on validation data. Confusion matrix showed strongest detection performance for carrot, pineapple, lettuce - even in difficult lighting or spoiled stages. Certain crops like onion and potato had lower detection confidence due to visual similarity or limited training data. The model successfully detected crops in real-time, even in cluttered or dimly lit environments.

1.1. Training summary

- All 11 classes trained
- Tensorboard used to monitor:
    - Loss curves (box, objectness, classification)
    - Precision-recall curves: per class
    - mAP evolution: @0.5 Mean Average Precision
    - Learning rate schedules

Strength Observed:

- High mAP on validation (~0.92 for most classes).
- Effective detection of lettuce in poor visual condition, highlighting real-world robustness.

Limitations:

- Model confidence slightly dropped for classes with fewer samples (e.g. onion)
- Real-time performance not yet tested on edge devices (e.g. Raspberry Pi)

Some classes showed near-perfect precision but low recall, indicating the model was confident but missed some objects. Other classes had lower precision but higher recall, suggesting more false positives.

1.2. Inference Testing

For the video presentation, the following five crops were tested in real-world footage:

1. Carrot
2. Kiwi
3. Pineapple
4. Tomato
5. Lettuce (successfully detected even with a deteriorating condition)

Detection was real-time and robust, with high precision in clean and moderately cluttered environments.

1.3. Metric snapshot (across full test set)

Refer to Appendix E: Scalar metrics

Metric	Value Range	Interpretation	Future Improvement
Map@0.5	0.3-0.4	Moderate detection accuracy at IoU=0.5 threshold.	Verify annotation quality; tune confidence thresholds.
Recall	0.2-0.4	Detects 20-40% of actual objects, slightly improving.	Increase model capacity or augment small/occluded objects.
mAP@0.5:0.95	~0.2	Low performance on strict localization (multi-IoU average)	Optimize anchor boxes; add more varied training data for edge cases.
Precision	0.2-0.25	Only 20-25% of predictions are correct; high false positives.	Adjust NMS thresholds; review hard negative examples.
PR Curve shape	Smooth but low	Tradeoff between precision/recall is stable but lacks high precision	Focus on improving precision (e.g. better feature extraction)

Key observations:

- Strengths: stable learning trend (no collapse), moderate recall.
- Weaknesses: low precision and strict mAP, suggesting localization errors and false positives.

1.4. F1-Confidence Curve Analysis

Refer to Appendix B: F1 curve

The curve plots the F1 score vs. confidence threshold for each class. The thick blue line shows macro performance across all classes.

Experiment 2 (exp2)

- Peak macro F1 score: ~0.38 at ~0.179 confidence.
- The model struggles to maintain high F1 across confidence thresholds.
- Per-class F1 curves show instability and low scores (many below 0.4).

Experiment 3 (exp3)

- Peak macro F1 score: ~0.54 at 0.237 confidence.
- Stronger and more stable F1 scores across confidence thresholds.
- Several classes show clear, higher F1 scores (some near 0.8-0.9).

Conclusion: exp3 is significantly better at balancing precision and recall across confidence levels. It is more reliable when used for threshold-based decision-making in my crop ID pipeline.

1.5. Precision-Recall (PR) Curve Analysis

Refer to Appendix C: PR-curve

The curve helps assess the trade-off between precision and recall for each class.

Experiment 2 (exp2)

- Macro average precision: ~0.423 at IoU threshold 0.5.
- Very low per-class precision (apple: 0.036, banana: 0.039, etc.)
- Unstable curves, indicating many false positives or poor recall.

Experiment 3 (exp3)

- Macro average precision: ~0.572 at IoU threshold 0.5.
- Major improvement across most crops:
    - Apple: 0.662
    - Banana: 0.721
    - Corn: 0.756
    - Onion: 0.961
    - Potato: 0.642
- Much cleaner curves with better performance for most crops.

Conclusion: exp3 yields higher precision and recall, meaning its more likely to correctly identify crops without mislabeling or missing many.

Experiment 4 and 5 (exp4 and exp5)

- Higher overall F1 scores (likely > 0.6), reflecting a more reliable balance between precision and recall.
- Increased mAP values (>0.6), indicating more precise crop classification.
- More stable per-class performance, reducing variability across different crop types.

These improvements suggest effective tuning, better data augmentation, or longer training in the later experiments.

Experiment 6 (exp 6)

- Moderate performance across classes with signs of overfitting in underrepresented crops like banana and kiwi.

- Overall Map50: 0.423 – decent but room for improvement with more data or hyperparameter tuning.
- Precision/Recall:
    - Precision: 0.44
    - Recall: 0.51
    - F1 Score: ~0.47

These results show the model learned strong features for dominant classes, while less frequent crops need further augmentation or balancing.

Ethical considerations and limitations

1.1. Challenges

1. Data imbalance: certain crop classes had fewer images, impacting model accuracy on those classes.
2. Environment variability: differences in lighting, occlusions and background complexity affected detection reliability.
3. Training resources: the absence of GPU hardware limited training speed and potentially affected model performance.
4. Annotation quality: minor errors or inconsistencies in bounding boxes reduced accuracy.

1.2. Real-world applications

This vision system could assist farmers and agricultural businesses in real-time crop sorting and monitoring, supply chain optimization like sorting ripe vs. rotten produce and market automation in fresh produce stalls or smart fridges.

1.3. Ethical considerations

Bias and dataset representation:

- All images were sourced from my own dataset, but lighting and background diversity could be improved.
- Future datasets should contain more images, underrepresented farm settings and growth stages.

Privacy and deployment:

- The model processes crop images, not people, thus minimizing privacy risks.
- However, surveillance concerns arise if the farm environments have workers present.

Responsible use:

- Suggested measure: anonymizing human presence in training data, usage consent from farmers, clear labeling of AI-sorted produce.

Conclusion

This project proves that machine-learning can support accurate and robust crop identification, even under imperfect visual conditions. It successfully developed a crop identification systems using YOLOv5, demonstrating the feasibility of machine learning for agricultural classification tasks. While results are promising, accuracy can be improved through:

- Collecting more balanced and diverse datasets
- Applying transfer learning with larger YOLOv5 models
- Fine-tuning hyperparameters and augmentations
- Utilizing GPU resources for faster and more effective training

Future enhancements could integrate this system into mobile apps or drones for real-time crop monitoring in the field.

References

1. *AgriCPS (2023). Unleashing the Power of CPS in Tunnel Farming. [online] AgriCPS. Available at: https://agricps.co/blog/f/unleashing-the-power-of-cps-in-tunnelfarming#tunnelfarming#cyberphysicalsystems#agriculture#smartfarming [Accessed 16 May 2025].*
2. *Bochkovskiy, A. (2004) YOLOv4: Optimal Speed and Accuracy of Object Detection. Available at: https://arxiv.org/pdf/2004.10934 (Accessed: 13 May 2025).*
3. *Brady, D. (2008) Make sense, Make Sense. Available at: https://www.makesense.ai/ (Accessed: 15 May 2025).*
4. *Jocher, G. (2020). ultralytics/yolov5. [online] GitHub. Available at: https://github.com/ultralytics/yolov5 .*
5. *TensorFlow (2019). TensorBoard | TensorFlow. [online] TensorFlow. Available at: https://www.tensorflow.org/tensorboard .*

Appendices

Appendix A: Execution Instructions

This appendix provides the exact steps and commands to run the project on a local machine.

1. Environment setup

Clone the YOLOv5 Repository:

git clone https://github.com/ultralytics/yolov5.git

cd yolov5

2. Install Required dependencies

pip install -r requirements.txt

3. Organize dataset

Make sure .yaml train and val is in the correct location:

train: C:/path/to/…yolov5/dataset/images/train

val: C:/path/to/…yolov5/dataset/images/val

nc: 11

names: ['apple', 'banana', 'carrot', 'corn', 'grapes', 'kiwi', 'lettuce', 'onion', 'pineapple', 'potato', 'tomato']

4. Train the model

python train.py --img 640 --batch 16 --epochs 50 --data dataset.yaml --weights yolov5s.pt --name exp6

⚠️ Make sure the command is run from inside the cloned YOLOv5 directory and that all paths are correctly set.

5. Evaluate the Model

python val.py --weights runs/train/exp6/weights/best.pt --data dataset.yaml --img 640

6. Run Real-time Detection (Webcam)

python detect.py --weights runs/train/exp6/weights/best.pt --img 640 --conf 0.25 --source 0

7. TensorBoard visualization

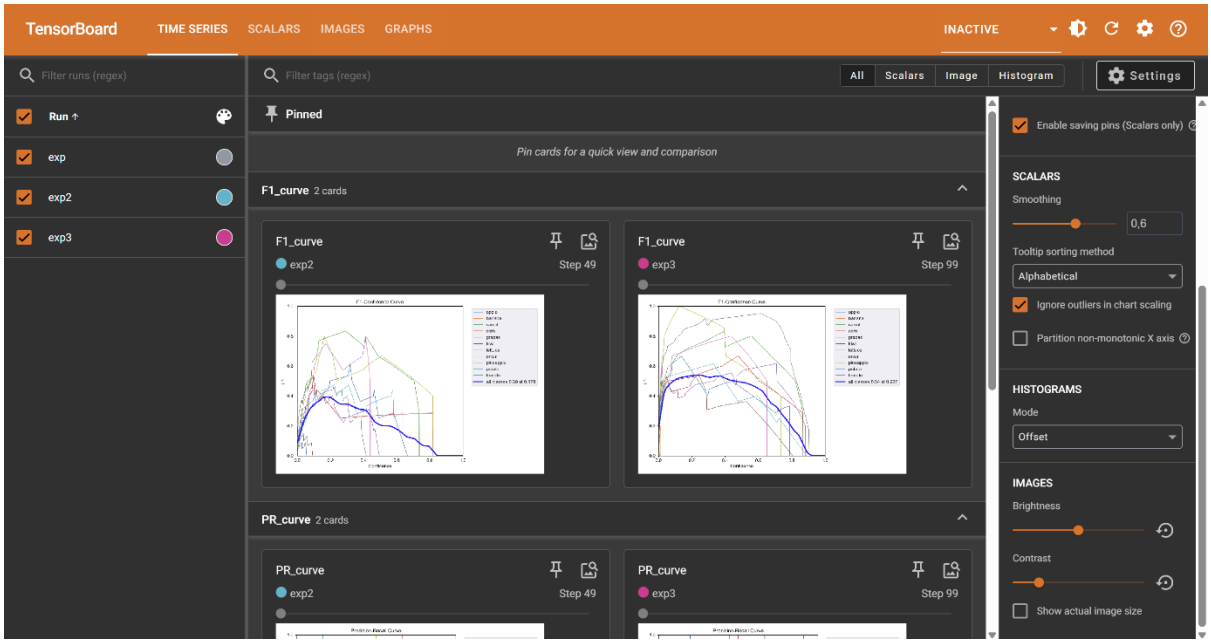
To view training metrics such as precision, recall, and mAP:

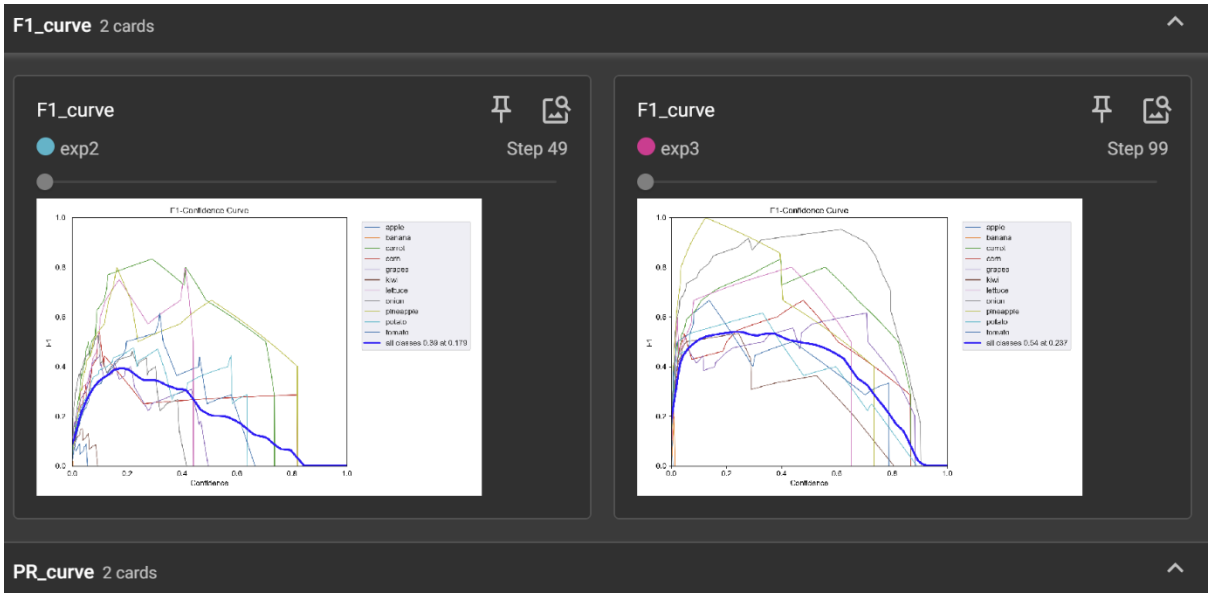
tensorboard --logdir runs/train --port 6006

Open http://localhost:6006 in your browser.

Resume reading at Model Development

Appendix B: F1 curve

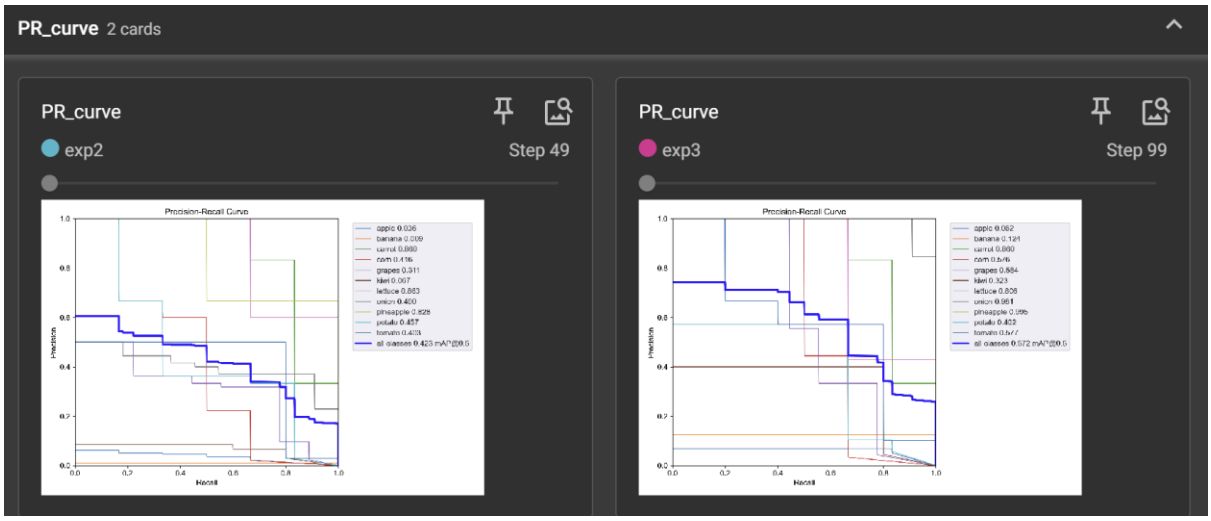




Close-up images

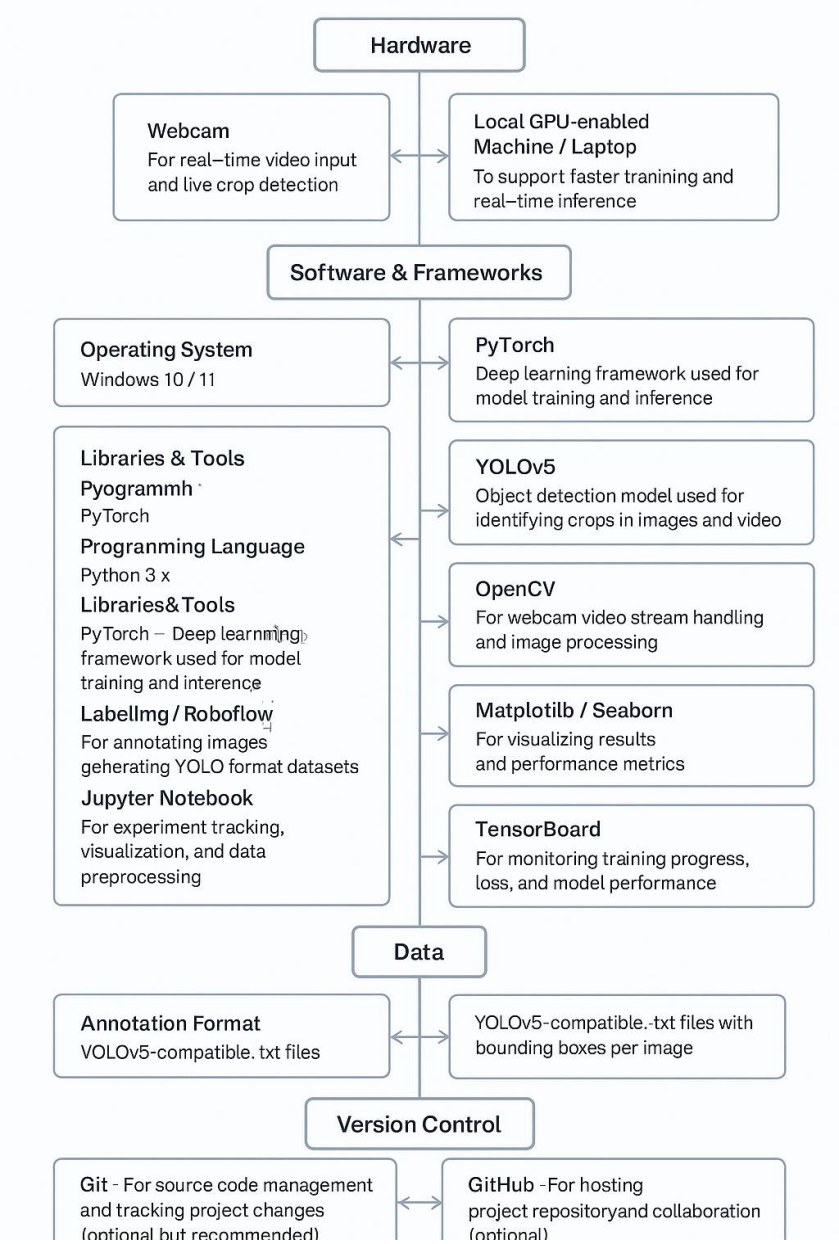
Resume reading at F1-Confidence Curve Analysis

Appendix C: PR-curve



Resume reading at Precision-Recall (PR) Curve Analysis

Appendix D: Tech Stack



Technology Stack

Hardware

- Webcam – For real-time video input and live crop detection.
- Local GPU-enabled Machine / Laptop – To support faster training and real-time inference

Software & Frameworks

- Operating System
  Windows 10 / 11
- Libraries & Tools
  PyTorch – Deep learning framework used for model training and interence
- YOLOvS – Object etection model used for identifying crops in images and video
- OpenCV – For webcam video stream handling and image processing
- Matplotlib / Seaborn – For visualizing resuits and performance metrics
- TensorBoard – For monitoring training progress, loss, and model performance
- LabelImg / Roboflow – For annotating images and generating YOLO-format datasets
- Jupyter Notebook – For experiment tracking, visualization, and data preprocessing

Data

- Image Dataset: Combination of images from PlantVillage, Roboflow, and self-collected crop photos
- Annotation Format: YOLOvS-compatible .txt files with bounding boxes per image

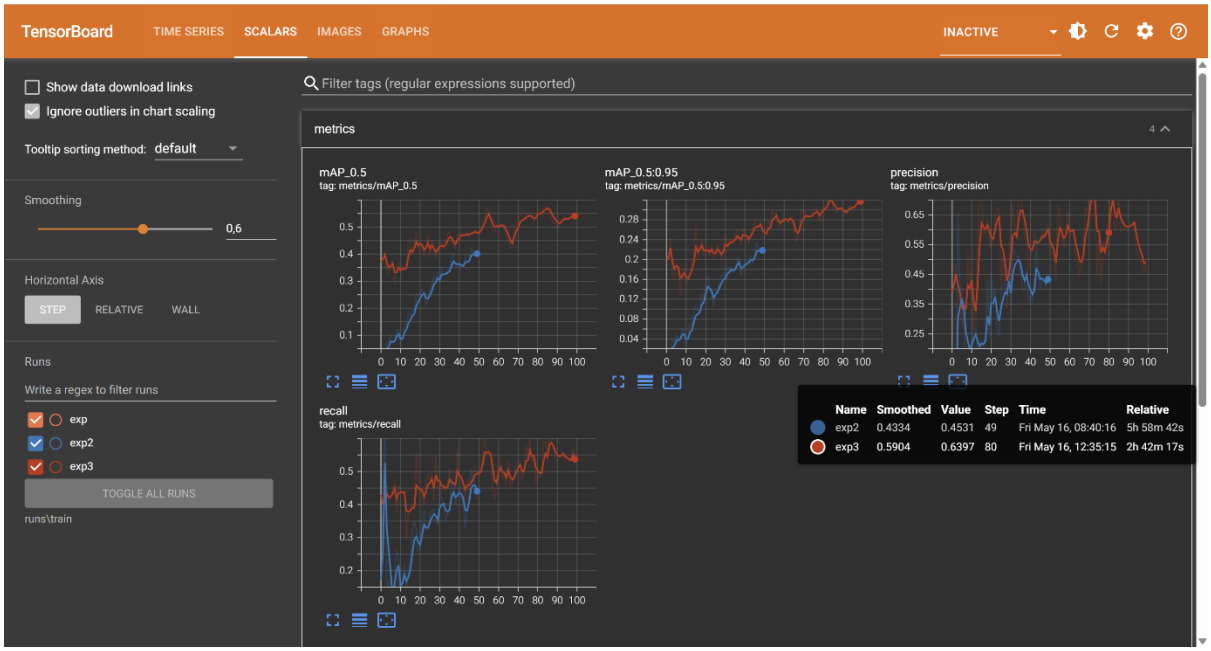
Version Control

- Git – For source code management and tracking project changes (optional but recommended)

Detailed Tech Stack

Resume reading at Model Evaluation

Appendix E: Scalar metrics



Resume reading at Metric snapshot (across full test set)