Classifying Bee Hive Frames with YOLOv8n

Reagan Hill

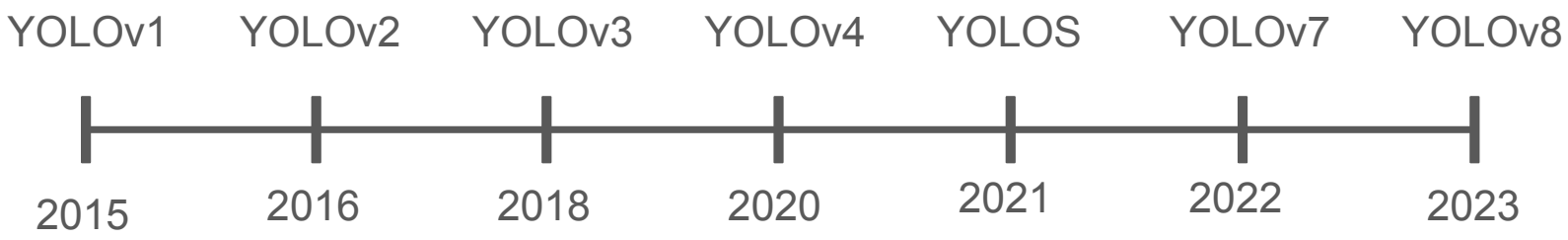
Department of Computer Science Utah State University

Overview

- YOLO Background
- YOLOv8 Installation and Setup
- YOLOv8 Training and Testing
- Comparisons
- Helpful Links
- Acknowledgements

Background

- YOLO: You only look once
- Created by Joseph Redmon and Ali Farhadi
- Real-time object detection and image segmentation



Background: YOLOv3, YOLOv4, and YOLOv7

Best mAP:

	Overall	Capped Honey Cell	Capped Worker Brood Cell	Empty Comb Cell	Pollen Cell	Uncapped Nectar Cell	Uncapped Worker Larva Cell	Bee Hive Frame
YOLOv3	13.5	21.7	12.5	11.6	11.7	19.3	18.6	25
YOLOv4	88	88.4	86.4	81.5	80.2	95.3	87.3	100
YOLOv7	1.1	1.1	2.7	1.4	0.8	0.1	3.6	0.2

Background: YOLOv8

- Created and owned by Ultralytics
- Two Licenses
  - AGPL-3.0
  - Enterprise
- Standard datasets are available (COCO, VOC, ImagNET)
- CLI and Python interface
- Multiple sizes
  - Nano: 6.5 MB
  - Small: 22.6MB
  - Medium: 52.1 MB
  - Large: 87.8 MB
  - XLarge: 136.9 MB

YOLOv8 Installation and Setup

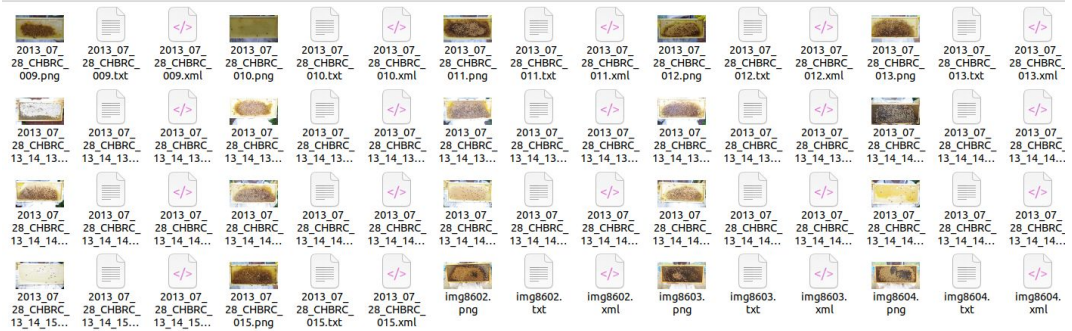
Install:

```
pip install ultralytics
```

Setup:

- YAML file
  - Set the path to the training set
  - Set the path to the validation set
  - Set the number of classes
  - Set the names of the classes
- Data set
  - Directory with the images and txt files
- Python file
  - `from ultralytics import YOLO`

```yaml
train: ../../data/obj
val: ../../data/obj

nc: 7
names: ['CappedHoneyCell', 'CappedWorkerBroodCell', 'EmptyCombCell', 'PollenCell'
 'UncappedNectarCell', 'UncappedWorkerLarvaCell', 'BeeHiveFrame']
```



YOLOv8 Training and Testing

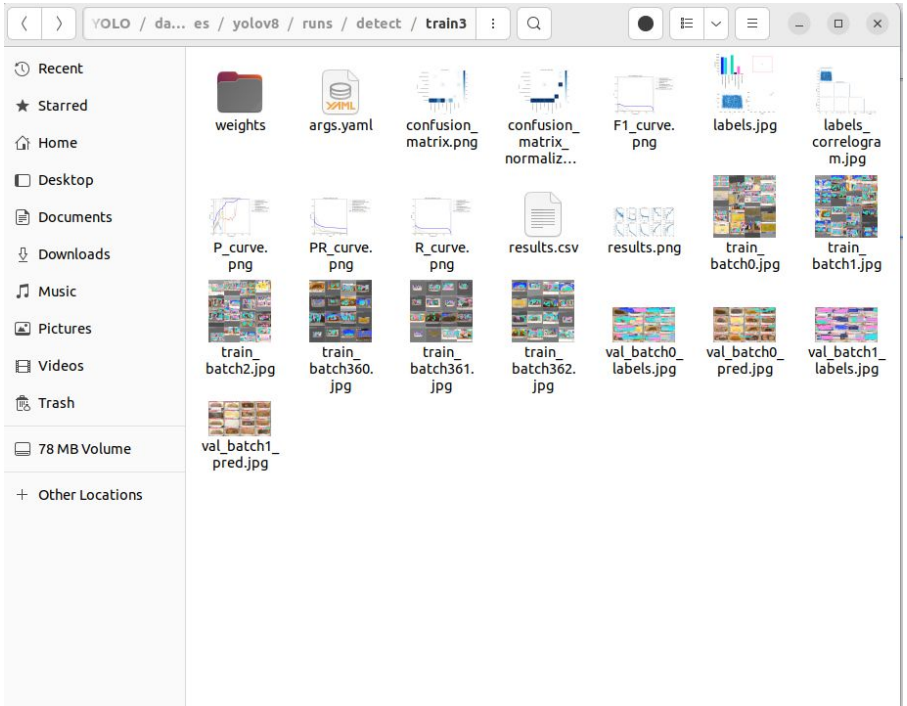
```python
from ultralytics import YOLO

model = YOLO("v8test.pt")

train_results = model.train(data='data.yaml', epochs=100, imgsz=640, device=0)

# evaluate performace on the validation set
metrics = model.val()

# perform obj detection on image
results = model("../data/obj/2013_07_28_CHBRC_010.png")
results[0].show()

# export format
export_model = model.export(format="onnx")


```

To run a trained model: Switch out the PyTorch file in the model definition

Next step: Looking into hyperparameter tuning

YOLOv8 Training and Testing: Output

- The `runs` and `detect` folders are created after the first run
  - `train` folders with run specific output
    - `weights` folder
      - `best.pt`
      - `last.pt`
      - `Best.onnx`
    - `args.yaml`
    - Confusion Matrix (normalized)
    - F1 curve
    - Percision-Recall curve
    - `results.csv`



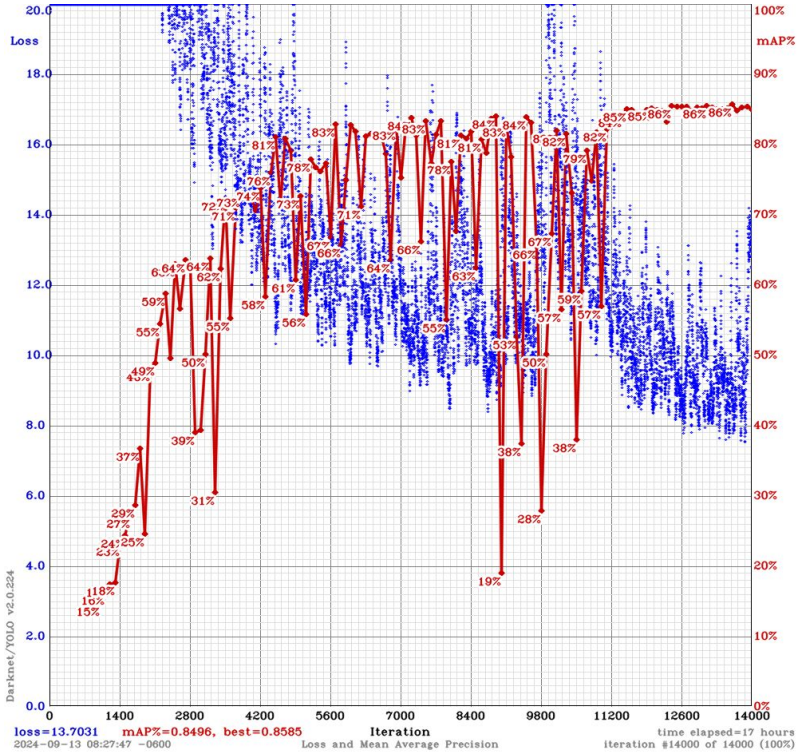
Comparisons: Object Detection on Images

YOLOv4

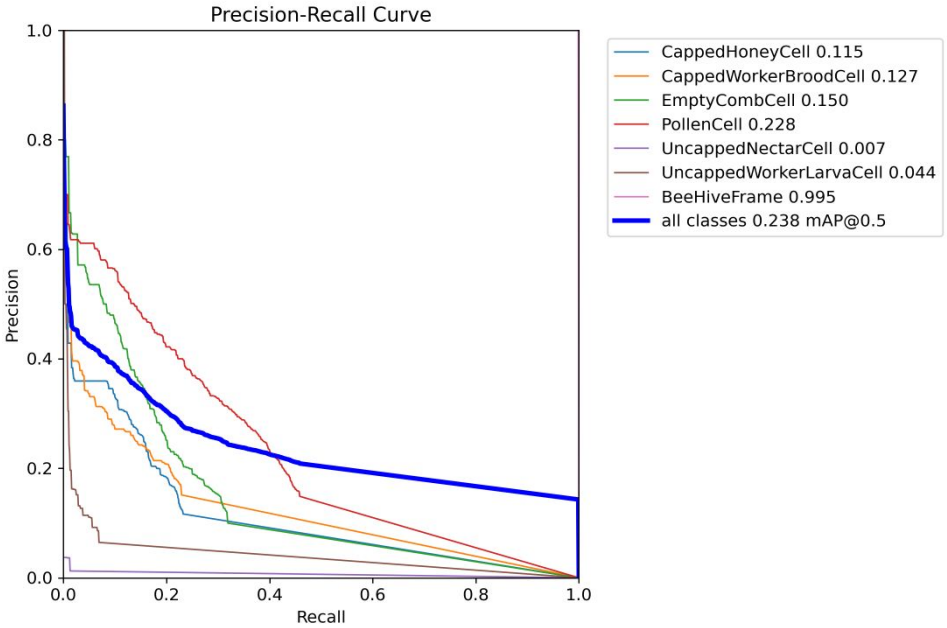
YOLOv8

Comparisons: Generated Result Graphs

YOLOv4



YOLOv8



(Potentially) Helpful Links

- Explore YOLOv8: https://yolov8.com/
- YOLOv8 GitHub: https://github.com/ultralytics/ultralytics
- YOLOv8 Python: https://docs.ultralytics.com/usage/python/
- YOLOv8 Docs: https://docs.ultralytics.com/models/yolov8/
- Model Training Ultralystics YOLO: https://docs.ultralytics.com/modes/train/
- Hyperparameter Tuning Guide: https://docs.ultralytics.com/guides/hyperparameter-tuning/

Acknowledgements

This research was supported, in part, by the intramural research program of the U.S. Department of Agriculture, National Institute of Food and Agriculture Program DSFAS A1541 Award 2024-67013-42521. The findings, conclusions, or recommendations expressed in this presentation have not been formally disseminated by the U.S. Department of Agriculture and should not be construed to represent any agency determination or policy.