Syracuse University
School of Information Studies

IST 691: DEEP LEARNING IN PRACTICE | PROF. PATRICK MCSWEENEY

LEVERAGING YOLO FOR REAL-TIME PREDICTION AND DETECTION OF PERSONAL PROTECTIVE EQUIPMENT

ENHANCING SAFETY COMPLIANCE ON CONSTRUCTION SITES

GROUP 3

Shoumik Reddy Kumbham
Likith Kolli
Sai Mani Kiran Chatrathi
Goutham Sri Vishwesh Bikkumalla

PROJECT OVERVIEW

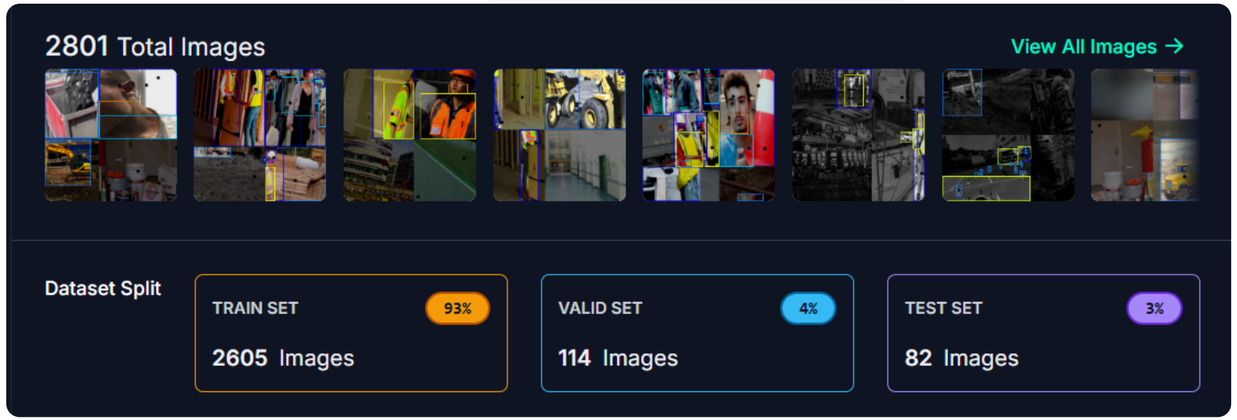
The primary goal of this project is to develop a deep learning model leveraging YOLOv11 for real-time detection and prediction of Personal Protective Equipment (PPE) compliance on construction sites. This involves identifying safety gear and flagging violations to enhance site safety and minimize risks.
The model is designed to detect and predict the following classes in real-time:

- Hardhat, No-Hardhat
- Mask, No-Mask
- Safety Vest, No-Safety Vest
- Person
- Safety Cone
- Machinery
- Vehicle

This solution aims to improve PPE compliance, instantly identify safety violations, and provide actionable insights for effective safety monitoring and accident prevention on construction sites.

DATASET

roboflow



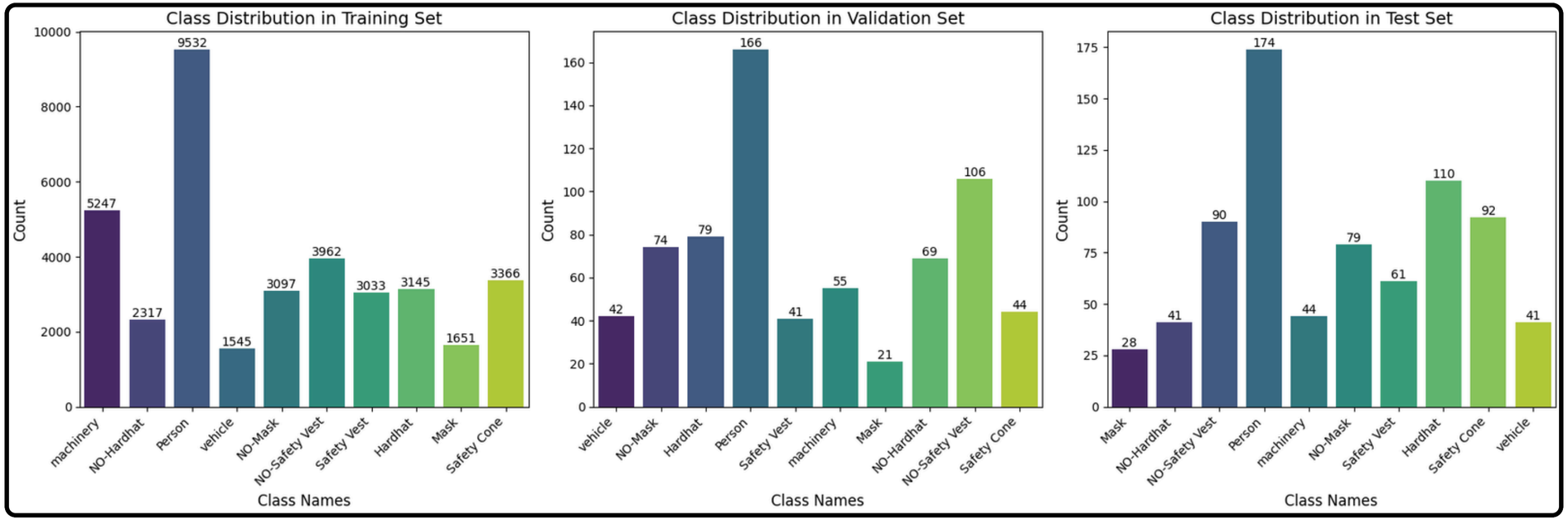
Augmentations

Outputs per training example: 5
Flip: Horizontal
Crop: 0% Minimum Zoom, 20% Maximum Zoom
Rotation: Between -12° and +12°
Shear: ±2° Horizontal, ±2° Vertical
Grayscale: Apply to 10% of images
Hue: Between -15° and +15°
Saturation: Between -20% and +20%
Brightness: Between -25% and +25%
Exposure: Between -20% and +20%
Blur: Up to 0.5px
Cutout: 6 boxes with 2% size each
Mosaic: Applied

- Leveraged Roboflow for dataset pre-processing and augmentation.
- Ensured precise annotations with accurate bounding boxes and labels.
- Distributed data effectively: majority for training, subsets for validation and testing.
- Enhanced diversity with augmentations like flips, rotations, and blurring.
- Prepared a robust dataset for training high-performing object detection models.

DATA PREPARATION

- Duplicate labels were detected and automatically removed to ensure clean and unbiased annotations for training.
- Dynamic Class Weighting: Ultralytics YOLOv11 automatically calculates and applies class weights based on the frequency of each class, ensuring minority classes receive higher priority during training.
- Focal Loss Integration: YOLOv11 integrates Focal Loss, which reduces the influence of well-classified examples and emphasizes harder, misclassified ones, effectively balancing learning across all classes.



WHAT MAKES YOLO11 IDEAL FOR OUR USE CASE?

- YOLO11 is the latest iteration in the **Ultralytics** YOLO series of real-time object detectors.
- Ultralytics released YOLO11 on **September 30, 2024**, as part of their v8.3.0 update.

Optimized for Computational Efficiency

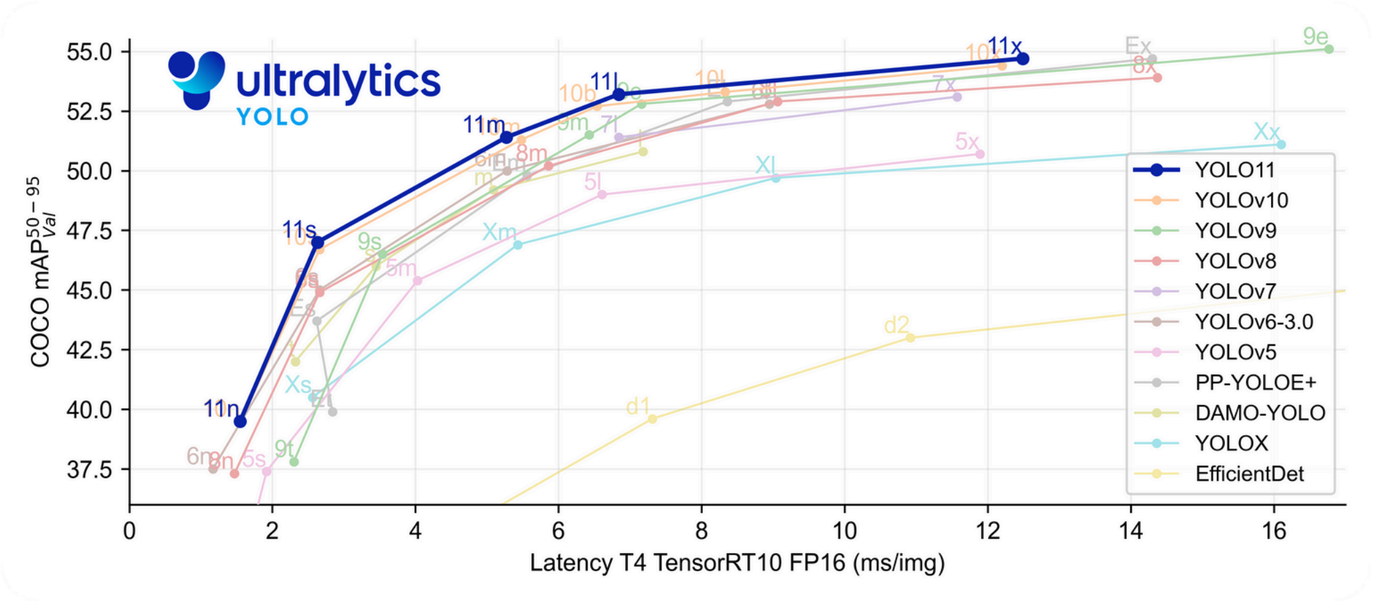
Real-Time Detection

Pre-trained on Robust Datasets

Customizability for Multi-Class Detection

High mAP (Mean Average Precision)

Balancing Accuracy and Speed



mAP@50-95 measures how well the model detects objects by averaging accuracy across different overlap thresholds (from 50% to 95%), balancing precision and recall for comprehensive performance evaluation.

YOLO11N ARCHITECTURE

BACKBONE (Extracts features from input image):

- Layers 0-9:
  - Sequential convolutions and C3k2 blocks extract hierarchical features from the input image.
  - SPPF (Layer 9):
    - Aggregates global and local information efficiently.
    - Used before transitioning to the neck.

NECK (Aggregates multi-scale features):

- Layers 10-22:
  - Combines features from different stages using Upsample, Concat, and C3k2 blocks.
  - Upsample layers:
    - Ensure higher-resolution features are preserved for finer detection.

HEAD (Performs predictions):

- Layer 23:
  - Detect block:
    - Final output layer that makes predictions for object classes, bounding boxes, and confidence scores.
    - Takes feature maps from three resolutions (P3, P4, P5) for detecting objects of various sizes.

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	6640	ultralytics.nn.modules.block.C3k2	[32, 64, 1, False, 0.25]
3	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
4	-1	1	26080	ultralytics.nn.modules.block.C3k2	[64, 128, 1, False, 0.25]
5	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
6	-1	1	87040	ultralytics.nn.modules.block.C3k2	[128, 128, 1, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	346112	ultralytics.nn.modules.block.C3k2	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 5]
10	-1	1	249728	ultralytics.nn.modules.block.C2PSA	[256, 256, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
13	-1	1	111296	ultralytics.nn.modules.block.C3k2	[384, 128, 1, False]
14	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
15	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
16	-1	1	32096	ultralytics.nn.modules.block.C3k2	[256, 64, 1, False]
17	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
18	[-1, 13]	1	0	ultralytics.nn.modules.conv.Concat	[1]
19	-1	1	86720	ultralytics.nn.modules.block.C3k2	[192, 128, 1, False]
20	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
21	[-1, 10]	1	0	ultralytics.nn.modules.conv.Concat	[1]
22	-1	1	378880	ultralytics.nn.modules.block.C3k2	[384, 256, 1, True]
23	[16, 19, 22]	1	432622	ultralytics.nn.modules.head.Detect	[10, [64, 128, 256]]

YOLO11n summary: 319 layers, 2,591,790 parameters, 2,591,774 gradients, 6.5 GFLOPs

TRAINING

AdamW(lr=0.000714, momentum=0.9) | **Epochs = 100** | **Data = data.yaml file**

```
[8]  1 # Import the Pre-Trained Model
     2 from ultralytics import YOLO
     3 model = YOLO("yolo11n.pt")

Creating new Ultralytics Settings v0.0.6 file
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ultralytics.co
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11n.pt to 'yolo11n.pt'...
100%| 5.35M/5.35M [00:00<00:00, 105MB/s]
```

The yolov11n.pt pre-trained model is loaded as the base for transfer learning. It allows leveraging pre-trained weights to improve training speed and accuracy, especially with limited data.

data.yaml

```yaml
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 10
names: ['Hardhat', 'Mask', 'NO-Hardhat', 'NO-Mask', 'NO-Safety Vest', 'Person', 'Safety Cone', 'Safety Vest', 'machinery', 'vehicle']

roboflow:
  workspace: roboflow-universe-projects
  project: construction-site-safety
  version: 28
  license: CC BY 4.0
  url: https://universe.roboflow.com/roboflow-universe-projects/construction-site-safety/dataset/28
```

The YAML file is crucial in YOLO training as it serves as a configuration file that defines the structure and parameters of the dataset, guiding the model during training and evaluation.

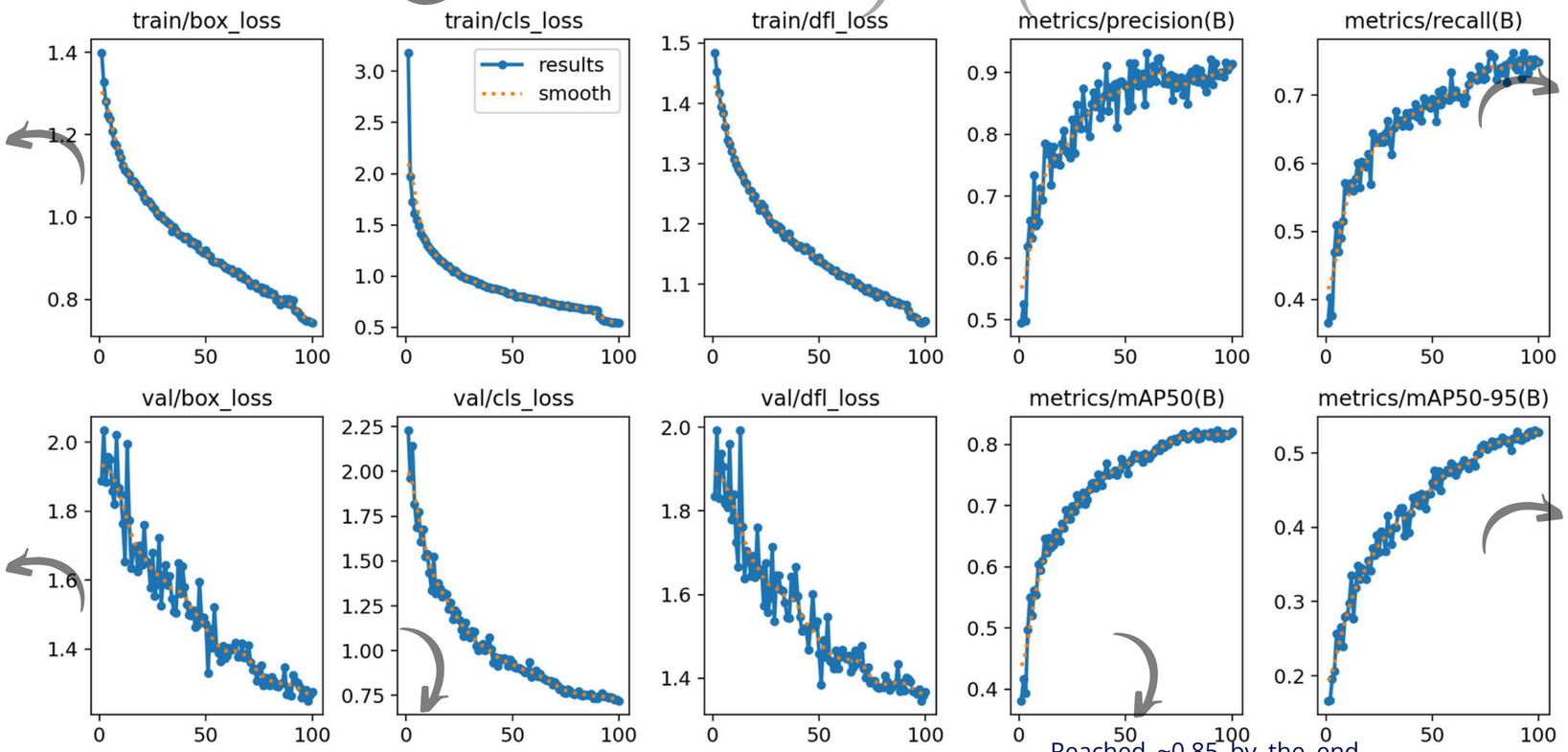
RESULTS

Reflecting improved classification of the objects within the training dataset.

Distribution Focal Loss, indicating that the model is better aligning its predicted probability distributions with the ground truth.

Precision Increased over the epochs, indicating fewer false positives in the training set.

showing the model's ability to better predict bounding box locations for objects in the training dataset



Progressively improved, signifying better detection of all objects in the dataset.

Started higher (~2.0) due to unseen data and reduced to ~1.4, showing that the model generalizes well to new data.

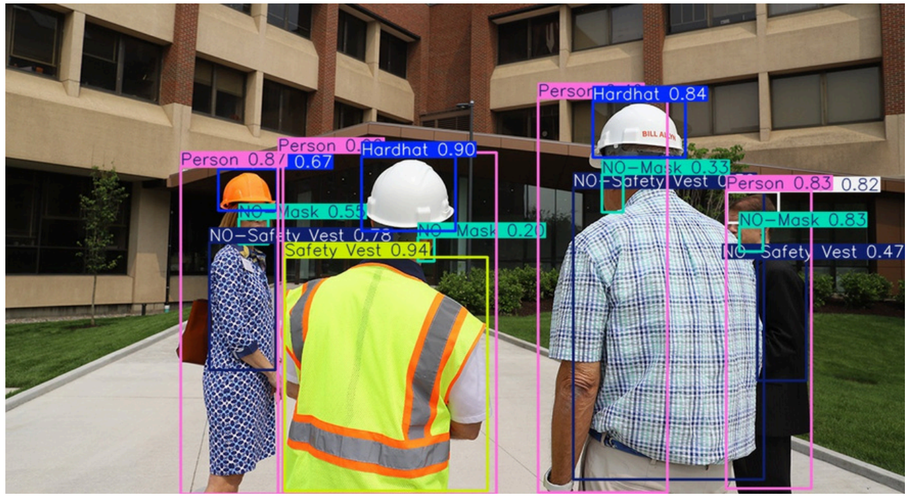
Achieved ~0.55, showing the model's robustness across stricter IoU thresholds.

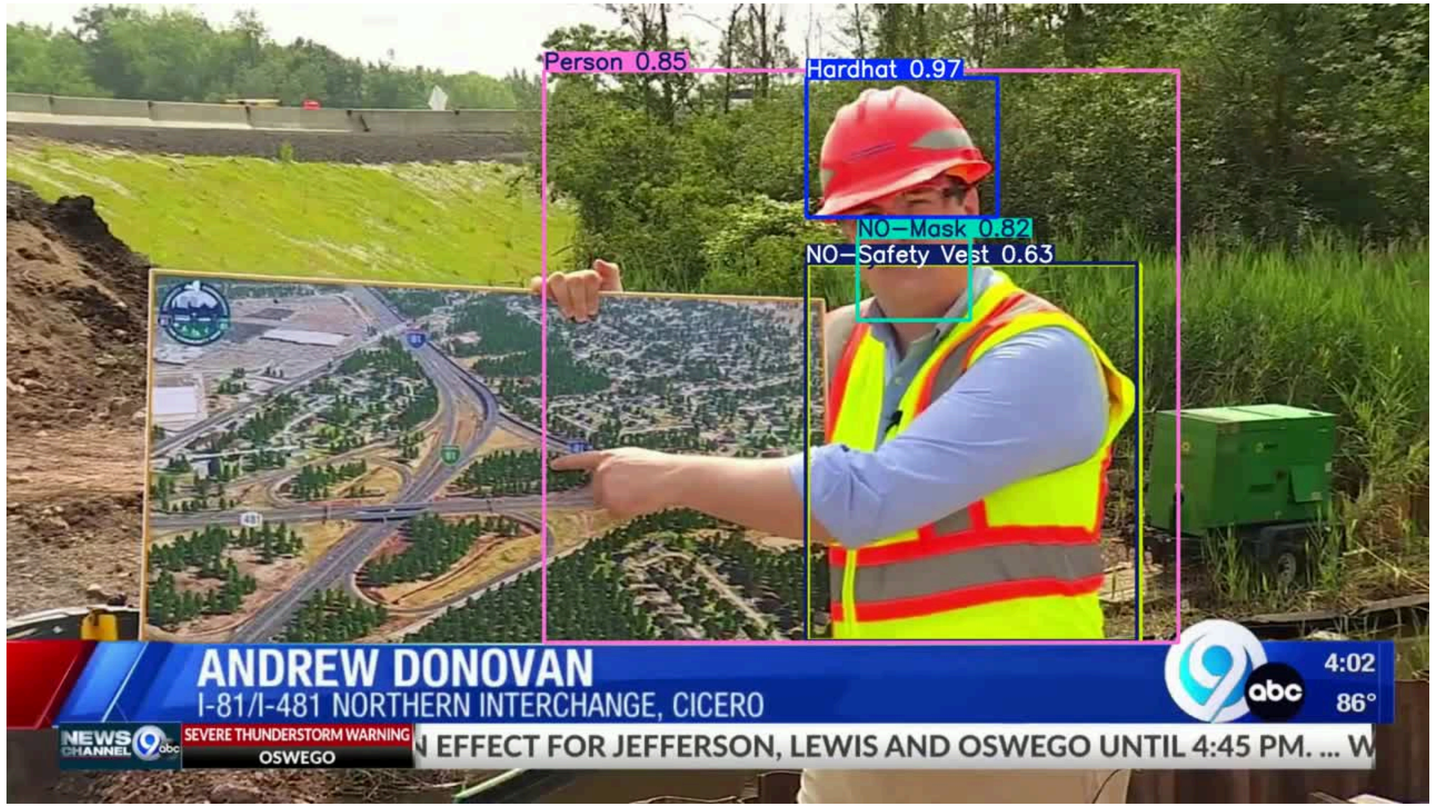
Initially high (~2.25) but dropped to ~0.75, reflecting improved performance on the validation dataset.

Reached ~0.85 by the end of training, demonstrating strong performance at detecting objects with a 50% overlap threshold.

PREDICTIONS

The images shown below are taken from **Syracuse University Link Hall Renovations**. The video footage demonstrating the future of I-81 construction was sourced from NewsChannel 9 WSYR Syracuse on YouTube, titled "**New I-81 construction update footage from Cicero**.". They were used for illustration purposes and to demonstrate the model's predictions during the construction site safety analysis.





CONVERTING .PT TO .TFLITE FOR ANDROID APP

- Introduced by Google in May 2017, TFLite (TensorFlowLite).
- Enables **efficient on-device inference** for mobile, embedded, IoT, and traditional devices.
- Platform compatibility: **Supports Android**, iOS, embedded Linux, and MCU.
- Performance Boost: TensorFlow Lite optimizes models for **faster inference by quantization and pruning**, often leading to significant speed improvements on edge devices.

Code to Convert model from .pt to .tflite

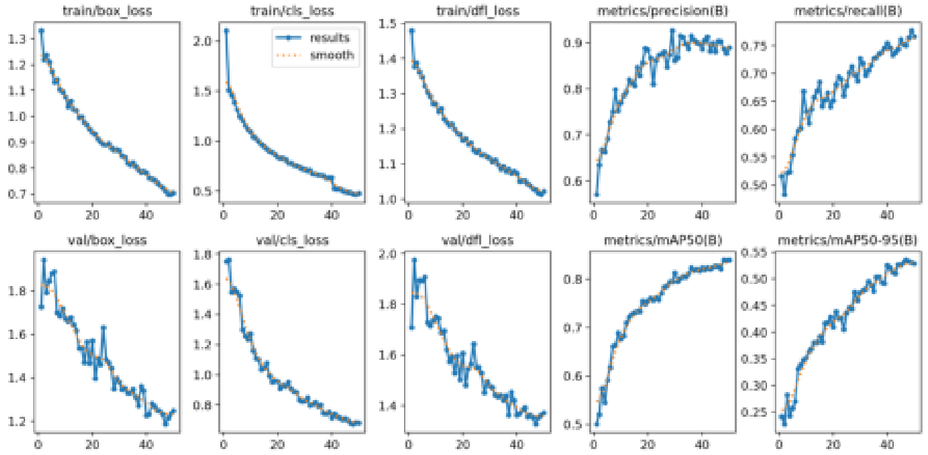
```
# Load the YOLO11 model
model = YOLO("yolo11n.pt")

# Export the model to TFLite format
model.export(format="tflite")  # creates 'yolo11n_float32.tflite'
```

SUMMARY

PREDICTION GOALS

- Trained YOLOv11n on the construction site safety dataset, achieving **mAP@50-95: 0.55** and **mAP@50: 0.80**.
- Model metrics are on par with the performance of the model showcased in the Kaggle competition, validating the effectiveness of the approach.
- Strengths:
  - High detection accuracy for key safety items such as "Hardhat" and "Safety Vest," with precision exceeding 80% for these classes.
  - Effective generalization to unseen validation scenario



INFERENCE GOALS:

- Multi-Source Inference with YOLOShow Module:
- Utilized the YOLOShow module to run inference on a computer.
- Supported inputs include IP cameras, local videos, and images..
- On-Device Inference with TFLite:
- Converted the trained YOLOv11n model to a TFLite format.
- Enabled efficient, offline inference on Android devices, ensuring portability and real-time application in the field.

FUTURE WORK

- Develop functionality to **identify and mark "red zones"** in construction areas.
- Enable detection and classification to **activate only when individuals or machinery enter the flagged danger zones**
- **Incorporate more data points**, including diverse environmental conditions and construction scenarios, to improve model robustness.
- Add functionality to **predict potential safety violations** based on movement patterns or historical data trends.

Syracuse University
School of Information Studies

THANK YOU

ANY QUESTIONS?