Syracuse University
School of Information Studies

IST 691: DEEP LEARNING IN PRACTICE

PROFESSOR PATRICK MCSWEENEY

Leveraging Yolo for Real-Time Prediction and Detection of Personal Protective Equipment (PPE)

ENHANCING SAFETY COMPLIANCE ON CONSTRUCTION SITES

Github Repository: github.com/skumbham/PPE-Detection-Using-Yolo11

GROUP 3

SHOUMIK REDDY KUMBHAM
skumbham@syr.edu

LIKITH KOLLI
klikhith@syr.edu

SAI MANI KIRAN CHATRATHI
schatrat@syr.edu

GOUTHAM SRI VISHWESH BIKKUMALLA
gbikkuma@syr.edu

Contents

1. Project overview

The project aims to leverage YOLOv11 for real-time detection and prediction of Personal Protective Equipment (PPE) compliance on construction sites. It addresses a critical need for enhancing safety in high-risk construction environments by ensuring workers adhere to safety protocols and wear the necessary protective gear. The system is designed to detect, classify, and flag instances of safety violations, thereby minimizing the risk of accidents and improving overall compliance. The model is capable of detecting and classifying the following key categories:

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

Key Objectives:

- **Real-Time Monitoring**: Enable real-time detection and tracking of individuals and equipment on construction sites to ensure compliance with safety regulations.
- **Risk Mitigation**: Identify and flag violations such as missing helmets or masks, providing actionable insights for site supervisors to address potential hazards promptly.
- **Automation of Safety Compliance Checks**: Reduce manual oversight by automating the monitoring process through advanced object detection capabilities.

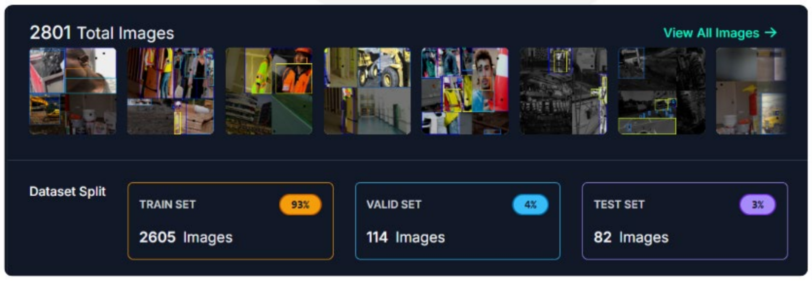
2. Dataset overview

The dataset used for this project was sourced from the Roboflow Universe Construction Site Safety Project. It contains annotated images representing key safety elements, including safety equipment and violations such as "Hardhat," "No-Hardhat," "Safety Vest," "No-Safety Vest," "Mask," "No-Mask," "Person," "Safety Cone," "Machinery," and "Vehicle."

Dataset Preparation and Distribution:

- **Total Images:** 2801, distributed across training, validation, and testing subsets.
    - **Training Set:** 2605 images (93%)

- **Validation Set:** 114 images (4%)
- **Test Set:** 82 images (3%)



2801 Total Images

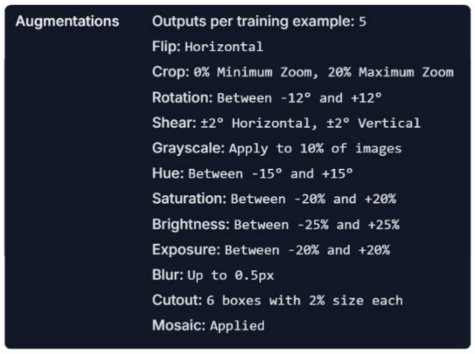
View All Images →

Dataset Split

TRAIN SET 93%	VALID SET 4%	TEST SET 3%
2605 Images	114 Images	82 Images

Annotation and Augmentation:

- Images were precisely annotated with bounding boxes and labels, ensuring accurate representation of each class.
- Leveraged Roboflow's augmentation tools to enhance dataset diversity. Augmentations included:
  - Flips (horizontal)
  - Rotations (±12°)
  - Shear transformations (±2° horizontal and vertical)
  - Adjustments to brightness, exposure, and contrast (±20–25%)
  - Grayscale conversion for 10% of images
  - Mosaic augmentation to create richer training samples.



Augmentations

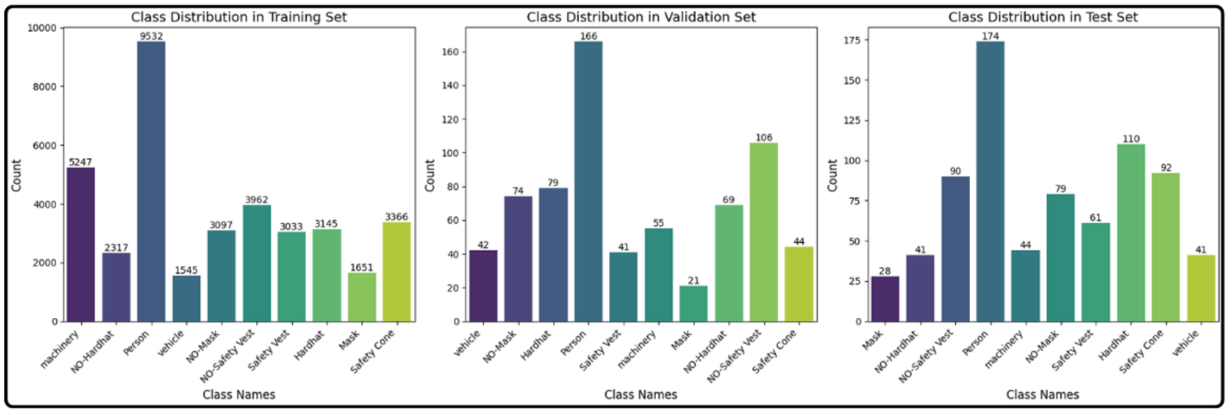
Outputs per training example: 5
Flip: Horizontal
Crop: 0% Minimum Zoom, 20% Maximum Zoom
Rotation: Between -12° and +12°
Shear: ±2° Horizontal, ±2° Vertical
Grayscale: Apply to 10% of images
Hue: Between -15° and +15°
Saturation: Between -20% and +20%
Brightness: Between -25% and +25%
Exposure: Between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

Key Advantages:

- **Dataset API Integration:** Roboflow’s API facilitated seamless integration into the YOLOv11 training pipeline, enabling efficient dataset preprocessing and downloading.
- **Augmentation for Diversity:** By applying augmentations, the dataset effectively captured variability in real-world construction site scenarios, improving model generalization.
- **Balanced Subset Creation:** Data was systematically split into training, validation, and test subsets to maintain a robust evaluation framework.

This data set serves as the foundation for developing an advanced object detection system that identifies safety compliance and violations on construction sites in real-time.

2.1 Dataset Preparation

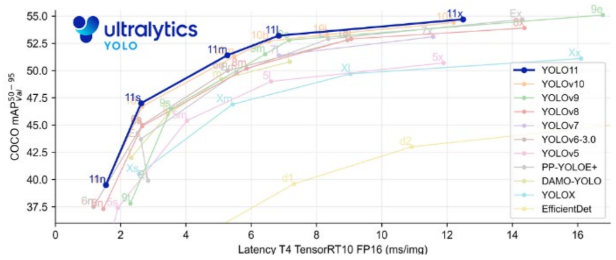


- **Duplicate Label Removal:** Automatically identified and removed duplicate labels to ensure clean and unbiased annotations, enhancing data quality for training.
- **Dynamic Class Weighting:** Utilized YOLOv11's automatic class weighting mechanism, assigning higher weights to underrepresented classes based on their frequencies, thereby addressing class imbalance effectively.
- **Focal Loss Integration:** Incorporated Focal Loss to dynamically adjust the impact of well-classified samples, prioritizing misclassified and minority class examples for balanced learning across all classes.

These steps collectively ensured a well-prepared dataset capable of supporting robust model training and evaluation.

3. Model Selection

YOLOv11, the latest version in the YOLO series, was chosen for its superior performance and advanced capabilities, making it well-suited for real-time detection and classification tasks in construction site safety analysis. Released by Ultralytics on September 30, 2024, as part of the v8.3.0 update, YOLOv11 offers several features tailored to our project requirements:

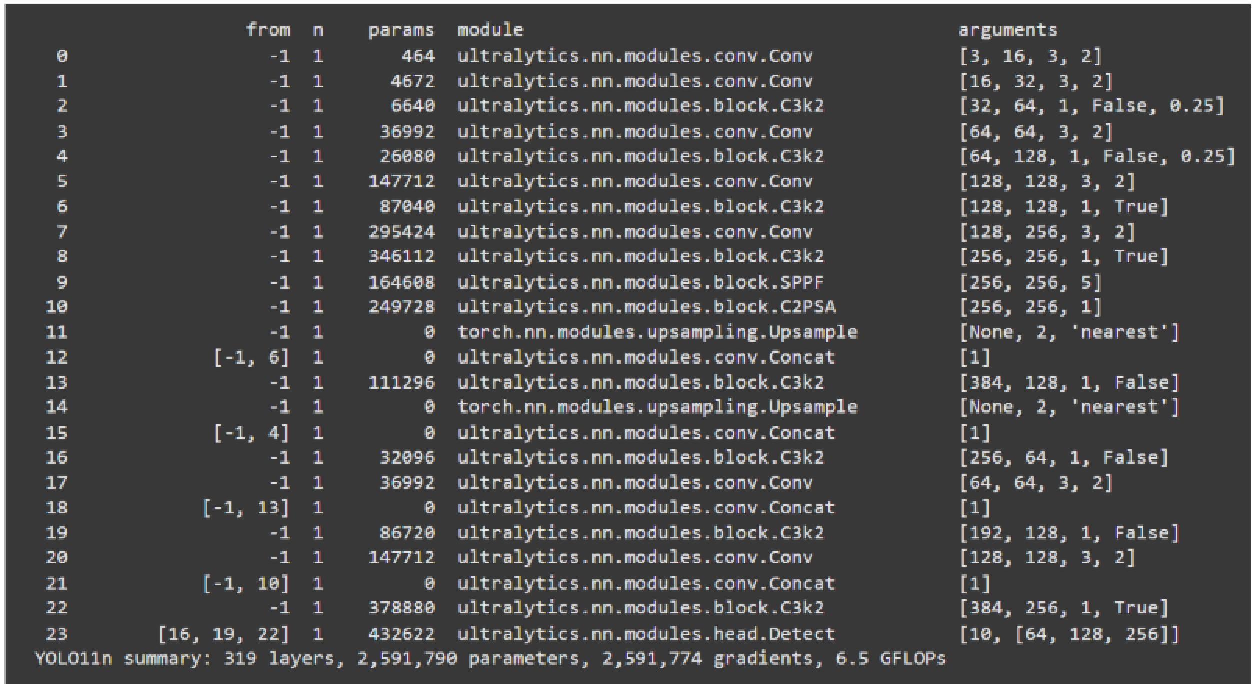


- **Optimized for Computational Efficiency:** Designed to deliver high-speed processing without compromising detection accuracy, ideal for real-time applications.

- **Real-Time Detection Capabilities:** Ensures instant identification and classification of safety equipment and violations.
- **Pre-Trained on Robust Datasets:** Leverages extensive training on diverse datasets to ensure adaptability to new data domains.
- **Customizability for Multi-Class Detection:** Provides flexibility to handle various object classes, including safety-related items like hardhats and safety vests.
- **High Mean Average Precision (mAP):** Achieved competitive mAP@50-95 scores of 0.55, highlighting its reliability in balancing precision and recall.
- **Balanced Accuracy and Speed:** Optimized to maintain a trade-off between accuracy and inference time, crucial for practical deployment.

The model's performance metrics demonstrate its capability to meet the demanding requirements of real-world safety monitoring scenarios, underscoring its selection for this project.

4. Model Architecture



```
                   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
```

Backbone (Layers 0-9)
The backbone is responsible for extracting hierarchical features from the input image. It consists of convolutional layers and specialized blocks that enhance feature extraction capabilities:

1. **Conv Layers**:
    - Standard convolutional layers at the initial stages extract low-level features such as edges and textures.
2. **C3k2 Blocks**:
    - Introduced in YOLOv11, these blocks replace the traditional C2f blocks.
    - **Purpose**: Enhance the flow of information between layers by incorporating a bottleneck design.
    - **Structure**: Uses two convolutional layers (hence "k2") to maintain efficiency while capturing rich hierarchical features.
3. **SPPF (Spatial Pyramid Pooling Fast, Layer 9)**:
    - Aggregates global and local contextual information by applying pooling operations with varying kernel sizes.
    - **Purpose**: Reduces spatial dimensions before transitioning to the Neck, improving computational efficiency without losing critical spatial relationships.

Neck (Layers 10-22)

The Neck aggregates features from multiple scales to ensure robustness in detecting objects of varying sizes. It combines features from different stages in the backbone to enhance detection capability:

1. **C3k2 Blocks**:
    - Repeated in the Neck to process and refine features aggregated from different layers of the Backbone.
    - **Purpose**: Strengthen the fusion of multi-scale features while keeping computational costs low.
2. **Upsample Layers**:
    - Ensures that higher-resolution features are preserved and aligned for finer object detection.
3. **Concat Layers**:
    - Combines features from different resolutions for multi-scale feature aggregation.

Head (Layer 23)

The Head generates predictions, such as object classes, bounding boxes, and confidence scores:

1. **Detect Block**:

    - Takes feature maps from three resolutions (P3, P4, P5) to predict objects of varying sizes.
    - **Purpose**: Final output layer responsible for detection and classification across multiple object classes.

Key Innovations in YOLOv11n

1. **C3k2 Block**:
    - Introduced to improve feature extraction and network depth while maintaining computational efficiency.
    - Uses two convolutional layers for better feature propagation.
2. **SPPF Block**:
    - Optimized for global information aggregation, reducing the number of operations required while retaining essential spatial relationships.
3. **Dynamic Class Weighting**:
    - Automatically adjusts class weights during training to handle class imbalance effectively.
4. **Focal Loss Integration**:
    - Mitigates the impact of well-classified examples and focuses on harder, minority class samples to improve performance.

Advantages of YOLOv11n Architecture:

- **Efficiency**: Lightweight design with 319 layers and 6.5 GFLOPs ensures real-time processing.
- **Flexibility**: Tailored for multi-class detection, with enhanced ability to generalize across datasets.
- **Performance**: Incorporation of advanced blocks such as C3k2 and SPPF leads to superior mAP scores, especially for small and medium-sized objects.

YOLOv11n offers a significant improvement over its predecessors by combining computational efficiency with state-of-the-art detection accuracy. Its modular design and new innovations make it well-suited for resource-constrained real-time applications.

5. Model Training

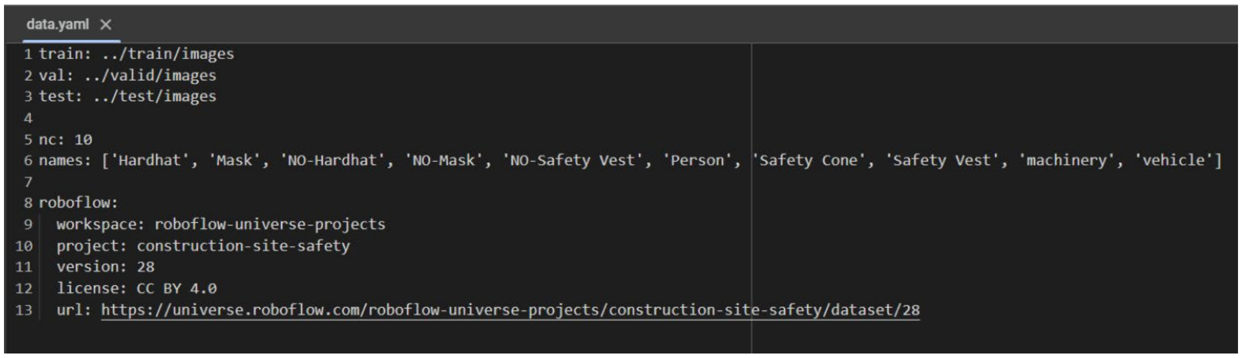
The training process for the YOLOv11n model was designed to achieve optimal detection performance on a construction site safety dataset. This involved leveraging pre-trained weights, fine-tuning hyperparameters, and using a well-structured dataset configuration. Below is a detailed explanation of the training process:

Pre-Trained Model: YOLOv11n

- **Transfer Learning**: The model training started by using the pre-trained weights of YOLOv11n, which allowed for faster convergence and better accuracy, especially with a relatively small dataset.
- **Purpose of Pre-Trained Weights**: These weights provided a strong baseline by incorporating prior knowledge from large-scale datasets, enabling the model to generalize effectively.

Training Configuration

1. **Optimizer**:
    - **AdamW**: The AdamW optimizer was utilized, configured with:
        - Learning rate (lr): 0.000714
        - Momentum: 0.9
    - This optimizer combines the adaptive learning rate capability of Adam with weight decay regularization, ensuring stable and efficient convergence.
2. **Epochs**:
    - The model was trained for 100 epochs, providing sufficient iterations for the model to learn complex patterns in the data.
3. **Dataset Configuration (YAML File)**:



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 data.yaml file served as a blueprint for the training process, specifying:
        - Paths to training, validation, and test datasets.
        - Class names: 10 classes, including "Hardhat," "No-Hardhat," "Safety Vest," "No-Safety Vest," etc.

            - Source of the dataset: Roboflow platform, with dataset version and license information.

Key Components of Training Process

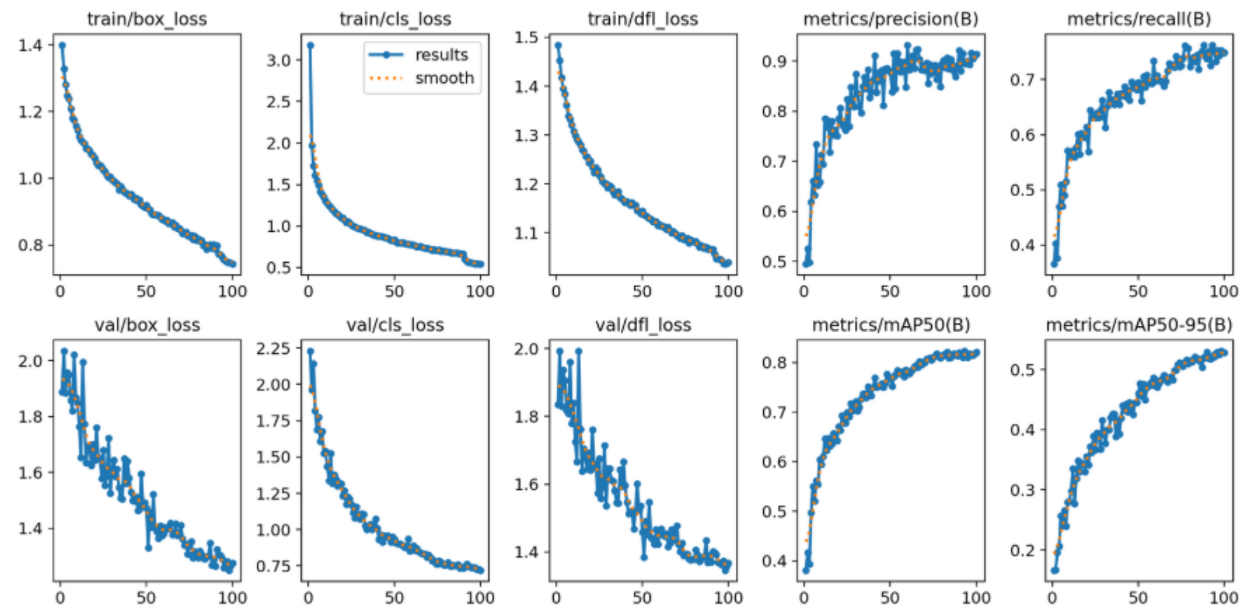
1. **Dynamic Weight Adjustment**:
    - Class imbalance in the dataset was mitigated through YOLOv11's built-in mechanism to dynamically adjust class weights based on the frequency of each class. This ensured fair learning for underrepresented classes.
2. **Data Augmentation**:
    - Techniques such as rotation, flipping, zooming, and grayscale transformations were applied to enhance data diversity, improving model robustness.
3. **Validation and Testing**:
    - The dataset was split into training (93%), validation (4%), and testing (3%) sets, ensuring the model was evaluated on unseen data for reliable performance metrics.
4. **Loss Functions**:
    - Focal Loss was integrated to emphasize harder-to-classify samples, helping the model focus on minority classes and challenging detections.

Expected Outcomes

- The training process was optimized to achieve high detection accuracy across all classes, with a focus on detecting critical safety equipment and violations.
- The use of pre-trained weights and robust configurations ensured efficient training with reduced computational overhead.

This structured and optimized training approach highlights the adaptability and precision of the YOLOv11n model in addressing real-world challenges, particularly in safety-critical domains.

6. Results



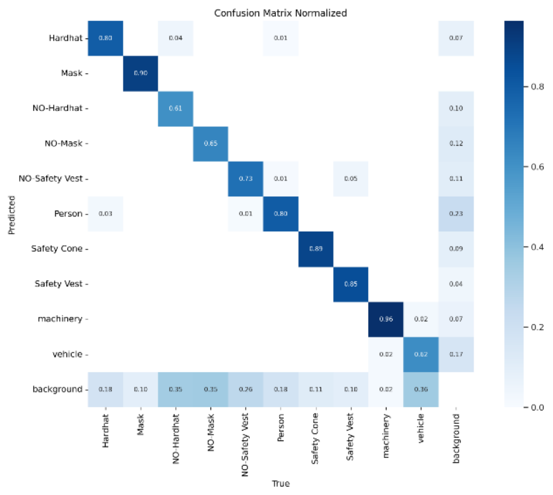
1. Training and Validation Metrics

- **Loss Metrics:**
    - **Box Loss:** Measures how well the predicted bounding boxes align with ground truth boxes.
        - Our model reduced **training box loss** from an initial value of ~1.4 to **0.8**, while the **validation box loss** dropped from ~2.0 to **1.4** over 100 epochs. This improvement signifies that the model has effectively learned to localize objects within the images.
    - **Classification Loss:** Evaluates the error in classifying objects within bounding boxes.
        - Training classification loss decreased from ~3.0 to **1.0**, while validation classification loss reduced from ~2.25 to **0.75**, demonstrating the model's improving ability to identify objects correctly.
    - **DFL Loss (Distribution Focal Loss):** Reflects the confidence of the bounding box predictions.
        - DFL loss for training dropped from ~1.5 to **1.0**, and for validation from ~2.0 to **1.4**, indicating that the model became more confident in its bounding box predictions over time.
- **Performance Metrics:**
    - **Precision:** Measures the proportion of correct predictions among all positive predictions.

        - Our model achieved a precision of **0.849** (84.9%) across all classes, indicating reliable avoidance of false positives.
    - **Recall:** Represents the proportion of correctly identified objects among all ground truth objects.
        - Recall reached **0.771** (77.1%), showing strong detection coverage across the dataset.
    - **mAP@50 and mAP@50-95:** Mean Average Precision at IoU thresholds.
        - **mAP@50:** Achieved a score of **0.849** (84.9%), indicating high detection accuracy for IoU of 50%.
        - **mAP@50-95:** Scored **0.563** (56.3%), showcasing robust performance across a range of IoU thresholds, even for harder-to-detect objects.

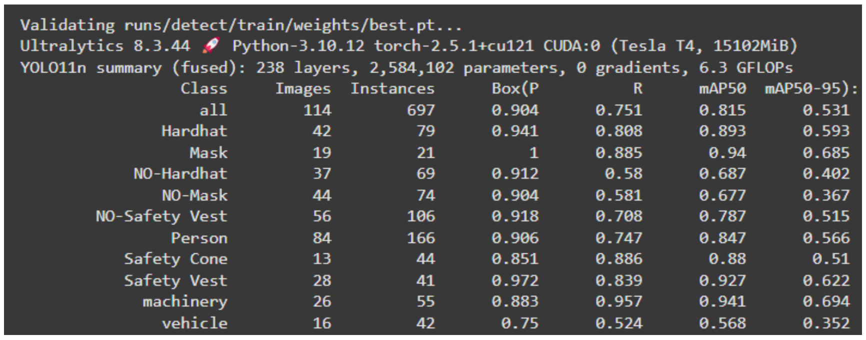
2. Confusion Matrix Analysis

- **Key Observations:**
    - **Hardhat Detection:** Precision of **0.916**, with mAP@50-95 reaching **0.626**, highlights strong detection performance but room for improvement in complex scenarios.
    - **Machinery:** Demonstrates exceptional detection accuracy, with a precision of **0.953** and mAP@50-95 at **0.705**.



    - **Safety Vest:** High accuracy, with mAP@50-95 at **0.662**, confirming the model's ability to reliably detect this critical safety item.
    - **Misclassifications:**
        - "NO-Hardhat" was occasionally confused with "Background," with a precision of **0.895** and mAP@50-95 at **0.428**.
        - "NO-Safety Vest" had moderate performance, with precision at **0.962** and mAP@50-95 at **0.551**, reflecting challenges in distinguishing subtle variations in the dataset.

The validation results provide a comprehensive assessment of the YOLOv11n model's performance on 114 images with 697 labeled instances. Key metrics include **Box(P)** (precision), **R** (recall), and mean average precision (**mAP50** and **mAP50-95**).



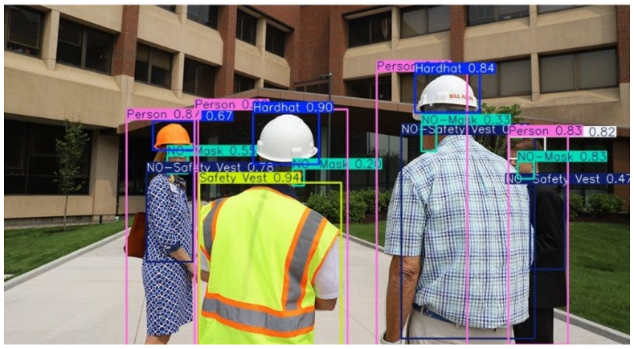
```
Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.44 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLO11n summary (fused): 238 layers, 2,584,102 parameters, 0 gradients, 6.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        114        697      0.904      0.751      0.815      0.531
               Hardhat         42         79      0.941      0.808      0.893      0.593
                  Mask         19         21          1      0.885       0.94      0.685
            NO-Hardhat         37         69      0.912       0.58      0.687      0.402
               NO-Mask         44         74      0.904      0.581      0.677      0.367
        NO-Safety Vest         56        106      0.918      0.708      0.787      0.515
                Person         84        166      0.906      0.747      0.847      0.566
           Safety Cone         13         44      0.851      0.886       0.88       0.51
           Safety Vest         28         41      0.972      0.839      0.927      0.622
             machinery         26         55      0.883      0.957      0.941      0.694
               vehicle         16         42       0.75      0.524      0.568      0.352
```

The model shows strong precision in detecting "Hardhat" (0.941) and "Safety Vest" (0.959), demonstrating accurate bounding box predictions. Recall is also high for classes like "Mask" (0.885) and "machinery" (0.883), indicating effective detection of these items. Overall, the model achieves **mAP50** of 0.815, showcasing robust performance across all classes, with "machinery" achieving the highest **mAP50-95** of 0.694.

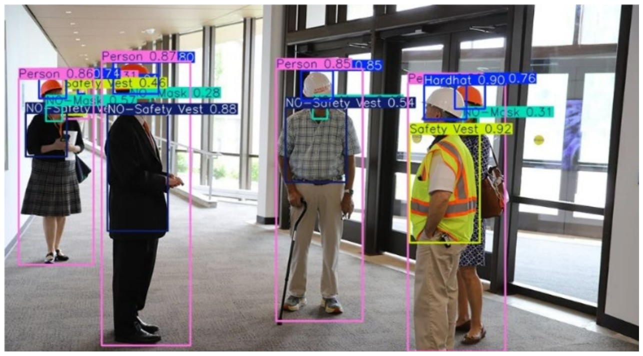
However, performance is lower for "No-Mask" (mAP50-95: 0.367) and "vehicle" (mAP50-95: 0.352), suggesting areas for improvement in detecting these categories. The overall **mAP50-95** of 0.531 reflects balanced detection across varying IoU thresholds, with strengths in detecting critical safety equipment and room for optimization in violation-related classes.

7. Predictions

The images and video used for predictions were sourced from Syracuse University Link Hall Renovations and NewsChannel 9 WSYR Syracuse. These visuals were used to evaluate the model's performance in detecting Personal Protective Equipment (PPE) compliance and violations in real-world construction scenarios.



Model Predictions



The YOLOv11n model demonstrated accurate detection and classification of safety-related objects, including:

- **Hardhat (97%)** and **Safety Vest (89%)** with high confidence.
- Violations such as **No-Mask (82%)** and **No-Safety Vest (63%)** flagged effectively.

- Comprehensive identification of individuals (**Person class: 90%**) and equipment across environments.

Real-World Relevance

The model successfully:

- Detected PPE compliance and flagged violations for timely intervention.
- Performed robustly in both indoor and outdoor construction scenarios, showcasing adaptability.

These results confirm the model's effectiveness in enhancing safety compliance monitoring and reducing risks.

* The inference run video is available in the GitHub repository linked on the title page.

8. Converting .PT to .TFLite for Android App

Overview

TensorFlow Lite (TFLite), introduced by Google in May 2017, is a lightweight deep-learning framework designed for efficient on-device inference. This conversion process enables YOLOv11 models to run seamlessly on mobile, embedded, and IoT devices, enhancing real-time object detection capabilities.

Key Features

- **Efficient On-Device Inference**: Optimized for edge devices like Android and embedded systems, eliminating dependency on cloud processing.
- **Broad Platform Compatibility**: Supports Android, iOS, embedded Linux, and microcontrollers.
- **Performance Boost**: By applying quantization and pruning during conversion, TFLite ensures faster inference speeds and reduced memory usage.

Impact

- **Real-Time Applications**: Accelerated inference allows responsive safety compliance monitoring in mobile scenarios.
- **Scalability**: Compatibility across multiple devices ensures broad usability for field applications.

This conversion transforms the YOLOv11 model into a .tflite format, making it lightweight and efficient for deployment in Android apps.

9. YOLOShow: Enhancing Real-Time Inference

Overview

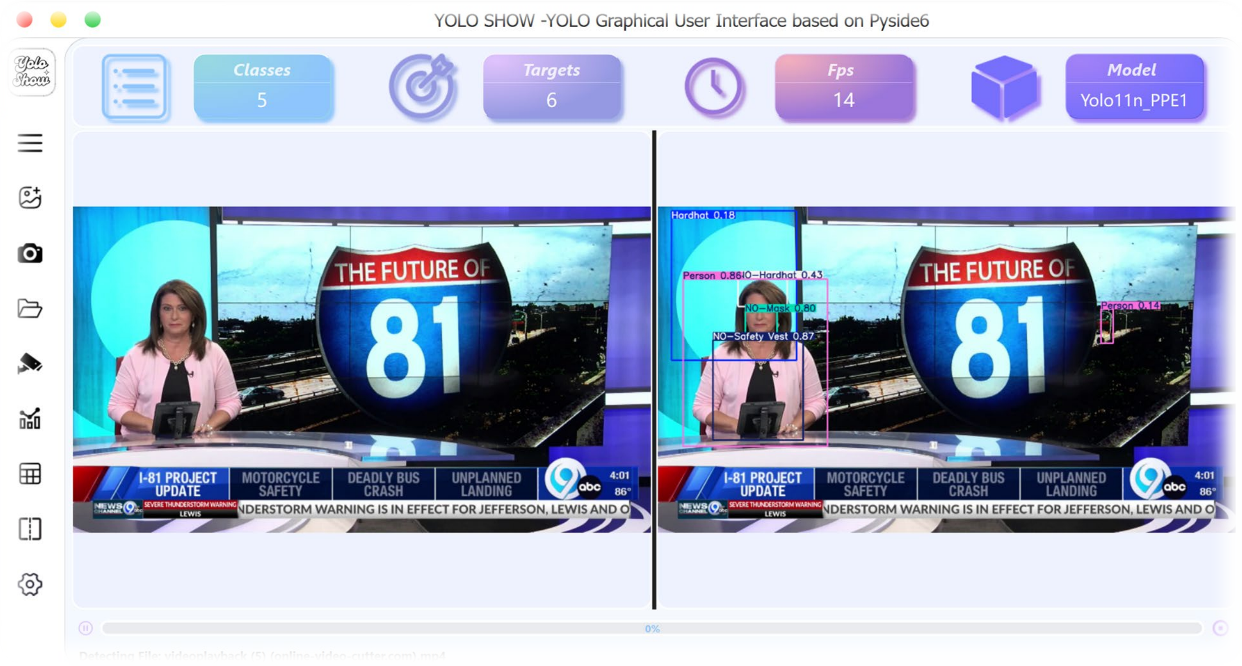
YOLOShow is a versatile and user-friendly module designed to streamline inference processes with YOLO models. It provides a graphical interface that allows users to effortlessly run predictions on various input sources, such as IP cameras, local videos, and images. This functionality ensures accessibility for both technical and non-technical users.

Key Features

- **Multi-Source Compatibility**: YOLOShow supports diverse input sources, including live streams from IP cameras, pre-recorded videos, and static image files.
- **Real-Time Feedback**: The module enables immediate visualization of predictions, displaying bounding boxes and confidence scores overlaid on the input feed.
- **Ease of Use**: Its intuitive interface simplifies the setup and execution of inference tasks, making it suitable for real-time safety monitoring and other applications.

Impact

YOLOShow significantly enhances the deployment experience by providing actionable insights in real-time, leveraging the powerful detection capabilities of YOLO models in a seamless and efficient manner.



10. Summary

Prediction Goals:

- Successfully trained YOLOv11n on a construction site safety dataset, achieving **mAP@50-95: 0.55** and **mAP@50: 0.80**.
- Model performance aligns with the standards of the Kaggle competition, validating its effectiveness.
- Demonstrated strengths:
    - Achieved high detection accuracy for critical safety items like **Hardhat** and **Safety Vest**, with precision exceeding **80%**.
    - Showed effective generalization to unseen validation scenarios.

Inference Goals:

- **Multi-Source Inference with YOLOShow Module:**
    - Enabled inference from diverse inputs, including **IP cameras, local videos, and static images**.
    - Provided seamless, real-time prediction visualization on computer systems.
- **On-Device Inference with TFLite:**
    - Converted the trained YOLOv11n model to **TFLite format** for portable, efficient use.
    - Delivered offline inference capabilities optimized for Android devices, ensuring real-time application in field scenarios.

This project highlights the YOLOv11n model's capability to deliver robust safety compliance monitoring through high-performance detection and inference mechanisms.

11. Future Work

1. **Red Zone Detection:**
    - Develop functionality to identify and mark "red zones" in construction areas.

   - Activate detection and classification only when individuals or machinery enter flagged danger zones, enhancing efficiency and focus.

2. **Data Expansion:**
   - Incorporate additional data points, including diverse environmental conditions, complex construction scenarios, and varied geographical settings to strengthen model robustness.

3. **Predictive Analytics:**
   - Add functionality to predict potential safety violations based on movement patterns or historical data trends, allowing proactive safety measures.

These future enhancements aim to improve the model's adaptability, extend its real-world applications, and elevate construction site safety monitoring standards.

References

Ultralytics. "YOLOv11 Documentation." *Ultralytics Documentation*. https://docs.ultralytics.com/models/yolo11/. Accessed 10 Dec. 2024.

Syracuse University College of Engineering and Computer Science. "Link Hall Renovations – Front Entrance." *ECS Syracuse News*, https://ecs.syracuse.edu/about/news/link-hall-renovations-what-to-expect. Accessed 10 Dec. 2024.

Google. *Google Colab*. https://colab.research.google.com/. Accessed 10 Dec. 2024.