**Comparative Analysis Report on YOLOv8 Variants**

**1. Introduction**

This report presents a comparative analysis of three YOLOv8 model variants that I experimented with during my project work. The three variants are:

* **YOLOv8n.yaml** – the model built from a configuration file, where training starts from scratch.
* **YOLOv8n.pt** – a pre-trained Nano model with optimized weights.
* **YOLOv8l.pt** – a pre-trained Large model designed for higher detection accuracy.

**2. Model Overview**

**YOLOv8n (Configured via yolov8n.yaml)**

* **Purpose**: This model is defined using a configuration file, which gives flexibility to customize the architecture.
* **Test Metrics**:
  + **Precision**: 0.5177
  + **Recall**: 0.4436
* **Note**: Since this model is trained from scratch it shows comparatively low performance

**YOLOv8n.pt (Pre-trained Nano Model)**

* **Purpose**: This version uses pre-trained weights that have been optimized on large datasets.
* **Test Metrics**:
  + **Precision**: 0.7508
  + **Recall**: 0.6160
* **Advantage**: The pre-trained model delivers good results, making it effective for real-time applications with limited resources.

**YOLOv8l.pt (Pre-trained Large Model)**

* **Purpose**: The large model is aimed at achieving higher accuracy, especially in complex scenes.
* **Test Metrics**:
  + **Precision**: 0.6897
  + **Recall**: 0.5618
* **Observation**: Although this model is designed for higher accuracy, in my tests it showed slightly lower precision and recall compared to the pre-trained Nano model, possibly due to overfitting ,This model is used mostly for very large datasets this Is perhaps why due to the smaller size of our dataset the results were not up to the mark .

**3. Performance Metrics**

**Accuracy Metrics**

* **YOLOv8n.yaml**:
  + *Precision*: ~51.8%
  + *Recall*: ~44.4%
* **YOLOv8n.pt**:
  + *Precision*: ~75.1%
  + *Recall*: ~61.6%
* **YOLOv8l.pt**:
  + *Precision*: ~69.0%
  + *Recall*: ~56.2%
  + Despite being larger, its metrics are a bit lower than the Nano pre-trained model in this experiment.

**4. Conclusion**

Based on my experiments:

* The **YOLOv8n.yaml** model, while customizable, currently underperforms compared to the pre-trained versions. More tuning could potentially improve its metrics.
* The **YOLOv8n.pt** model stands out for its high precision and recall, making it a strong candidate for practical applications, especially when speed and efficiency are required.
* The **YOLOv8l.pt** model, although designed for higher accuracy, did not perform as well as expected in this set of tests, suggesting that its benefits may be more better in different scenarios.

Overall, my results indicate that the pre-trained YOLOv8n.pt model offers the best balance of accuracy, speed, and resource efficiency for my current application.

***OUTPUTS OF THE SELECTED MODEL***



