Introduction

For my project, I am exploring ways to improve the YOLOv7 object detection model. My primary approaches involve leveraging a SHAP (SHapley Additive exPlanations) framework to analyze what YOLOv7 focuses on when classifying objects, as well as conducting an in-depth examination of the training process, including data preprocessing and class distribution analysis.

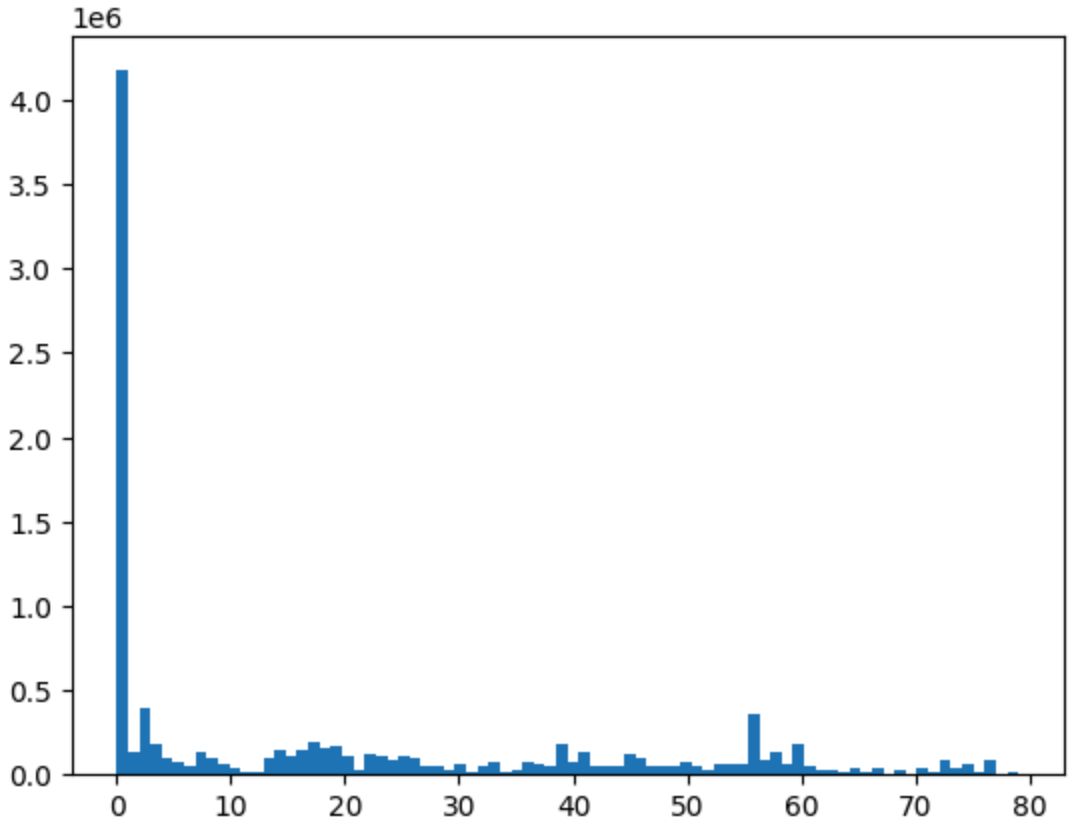
YOLOv7

The first phase of the project involved downloading the YOLOv7 source code and setting it up on my machine. This included:

- Downloading the COCO dataset, which contains over 100,000 images.
- Installing the required dependencies for YOLOv7.

I successfully got the real-time YOLOv7 model working with my webcam. This allowed me to observe YOLOv7's performance in real-time, analyzing how it classified objects as I introduced images into the frame. The YOLOv7 model I used was trained on 80 different object classes.

Class Distribution



One of the initial insights came from analyzing the class distributions in the training data. As seen in the chart above, there is a significant bias toward the class labeled `0`, which represents *person*. This imbalance was evident during real-time testing, as the model showed greater confidence in bounding boxes placed around people compared to those placed around other objects.

It would be interesting to apply data-balancing methods to even out the class distributions and evaluate whether this improves the model's performance across all object classes.

SHAP

SHAP is a tool for analyzing machine learning models, providing insights into how models make decisions. For this project, I installed the SHAP Python library and set up all its dependencies.

The next stage of my project will involve using SHAP to analyze YOLOv7, aiming to identify its weaknesses and determine potential improvements.