**Object Detection Model Building Assignment for AI Interns Report**

**Overview**

In this assignment, you will build an object detection model by adding detection layers to an existing CNN backbone. This project will give you hands-on experience with computer vision architectures while allowing you to explore how to effectively use AI assistance in your development workflow.

**Introduction**

This project leverages the KITTI dataset, a widely used benchmark in autonomous

driving research, to train a Convolutional Neural Network (CNN) for object

classification. Bounding box annotations from the dataset are parsed to extract and

crop individual objects from images. These cropped regions are resized to 64x64

pixels and normalized to serve as input to the CNN. The model is trained using

categorical crossentropy loss with a softmax activation in the output layer to classify

objects into distinct categories. The performance of the model is evaluated using key

metrics including precision, recall, and F1-score. While YOLOv8 offers an end-to-end

solution for detection, this approach focuses on classification using cropped regions,

allowing for more controlled training and evaluation of class discrimination.

**Data PreProcessing** **& Steps**

**Components**

1. Dataset loaded using KaggleHub from klemenko/kitti-dataset.

2. Image and label files parsed to extract bounding boxes.

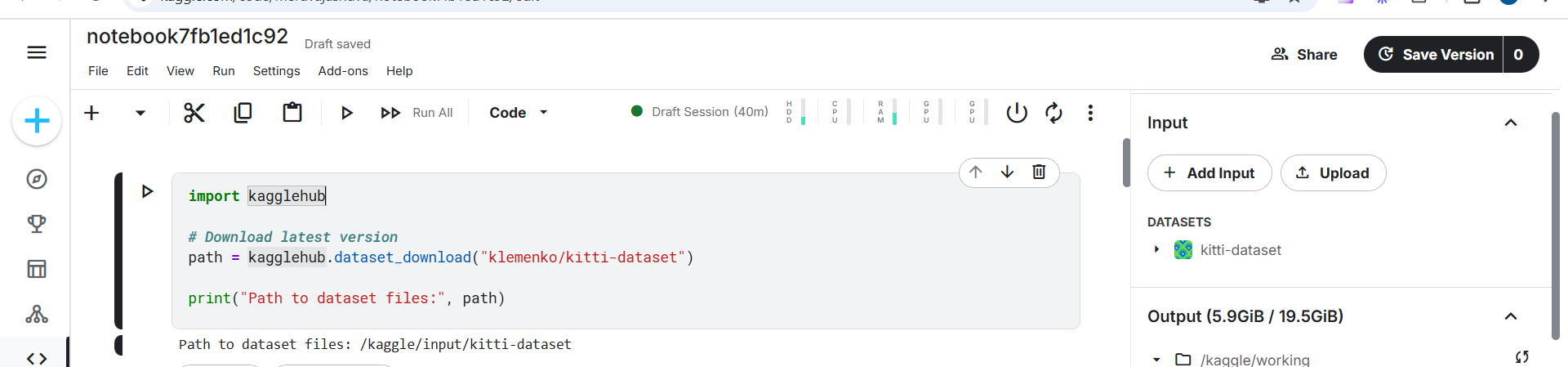
3. Objects cropped and resized (64x64), then normalized for CNN input.

4. CNN trained using categorical crossentropy and softmax output.

5. Model evaluated using precision, recall, and F1-score

**Code Snippets:**

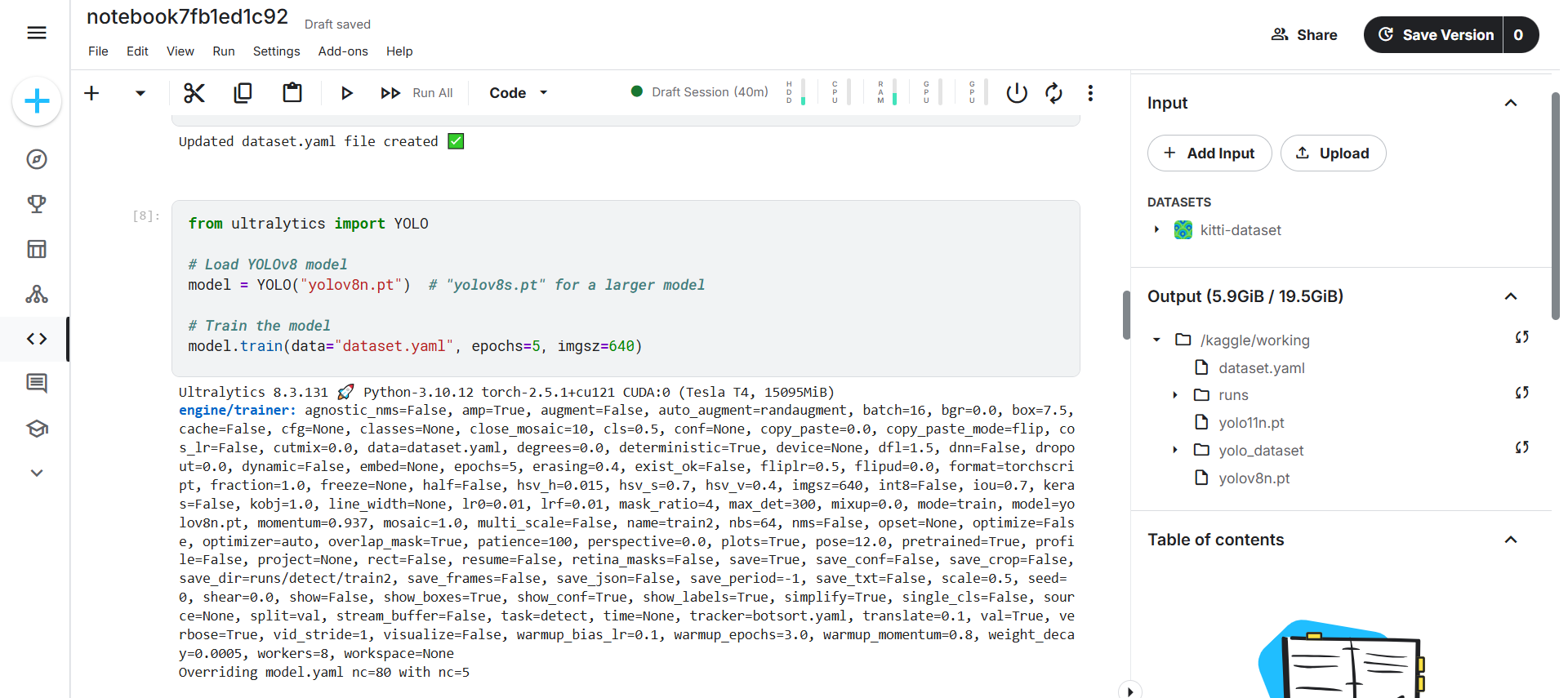
Steps1:Download the dataset

****

Step2: CNN Approach



Step3: YOLO Approach:





**Challenges and Problem Solving**

* **KITTI labels contain many object types**; the code filters only specific classes (Car, Pedestrian, Cyclist, Truck, Van) to simplify classification
* **Invalid bounding boxes** (e.g., negative width/height, out-of-bounds coordinates) had to be manually checked and skipped during preprocessing.
* **Class imbalance**: Some classes like 'Van' or 'Truck' had significantly fewer samples than 'Car', which affected the model's ability to generalize across all classes.
* **Small image crops (64x64)** limit the amount of visual detail, making it harder for the CNN to learn meaningful features, especially for fine-grained class differences.
* **Full object detection was not implemented**; the approach was limited to classification on cropped bounding boxes, which limits real-world applicability
* **Large dataset size** increased the preprocessing and training time significantly. Working with thousands of high-resolution images required careful memory and storage management.
* **GPU usage was restricted** (e.g., using Kaggle free tier or local CPU training), making training slow and sometimes leading to timeouts or session disconnections.
* **Model training took substantial time** even with a simple CNN, due to the large number of training samples and small batch size needed to prevent memory overload.
* **Data augmentation was skipped or limited** due to time/resource constraints, which could otherwise improve model generalization.
* **Error tracing was difficult** when handling a pipeline with many moving parts (label parsing, cropping, resizing, training), especially with occasional mismatches between image and label counts.
* **Model evaluation on the validation set** showed performance drops on minority classes, suggesting the need for techniques like weighted loss or oversampling.
* **Manual verification** of cropped images and label mappings was needed to ensure data integrity before training.

**What I Learned**

* How to work with complex real-world datasets like KITTI, which have rich annotation formats and diverse object classes.
* How to build a custom image classification pipeline from scratch, including image cropping, resizing, label encoding, and model evaluation.
* Gained experience in training a CNN model on small object crops and understanding its limitations.
* Learned to evaluate model performance using metrics like precision, recall, and F1-score, and how class imbalance can affect these.
* Understood the importance of preprocessing—cropping images correctly, handling invalid bounding boxes, and maintaining consistent label mappings.
* Learned the need to balance class distribution, and how underrepresented classes can degrade performance if not handled properly.
* Realized that model generalization can be improved with data augmentation and more robust sampling strategies.
* Recognized that YOLO (You Only Look Once) is better suited for real-time object detection, as it predicts bounding boxes and class probabilities directly from full images, instead of just classifying crops.
* Understood the difference between classification vs. detection: our CNN model classifies cropped images, while YOLO could locate and classify multiple objects in one frame.
* Learned that YOLOv5 or YOLOv8 could be used to convert KITTI to YOLO format, train an end-to-end detector, and overcome limitations of small image crops.
* Explored how GPU acceleration is essential for deep learning tasks on large datasets, and how limited GPU access (on free tiers) creates time bottlenecks.
* Understood how to split data into training and validation sets thoughtfully to avoid overfitting and to ensure fair performance evaluation.
* Learned to debug issues when the number of labels didn’t match the number of images, and how such mismatches can affect training.

**How I Feel About the Balance Between Writing Code Myself vs. Using AI Assistance**

* I found that writing code myself helped reinforce foundational knowledge, especially in preprocessing, file handling, and model training logic.
* Using AI assistance was very helpful for debugging, quickly finding correct syntax, and exploring better ways to structure the code.
* The balance was effective when I used AI as a guide or mentor, not a solution provider—first trying things myself, and then asking for feedback or improvements.
* For time-consuming tasks like repetitive directory handling or file conversions, AIhelped speed up boilerplate work, allowing me to focus on logic and learning.
* Relying too heavily on AI without understanding the problem would have limited my learning, so I made sure to validate and experiment with AI suggestions.
* Overall, combining both approaches gave me a deeper understanding and improved my confidence in solving computer vision tasks.

**Suggestions for Improving This Assignment**

* Provide GPU access or allow Colab usage, since training CNNs or YOLO models is computationally intensive and difficult to do efficiently on CPU.
* Consider scoping down the dataset size e.g., use a smaller KITTI subset or synthetic dataset to help students iterate and debug faster.
* Add a starter notebook or clear structure for beginners to understand the expected data flow and outputs.
* Provide optional challenges to implement full object detection with YOLO for students who want to go beyond classification.
* Offer a comparison section that explains when to use classification vs. detection vs. segmentation for different real-world problems.
* Include some guidance on visualizing predictions to better interpret the model's strengths and weaknesses.