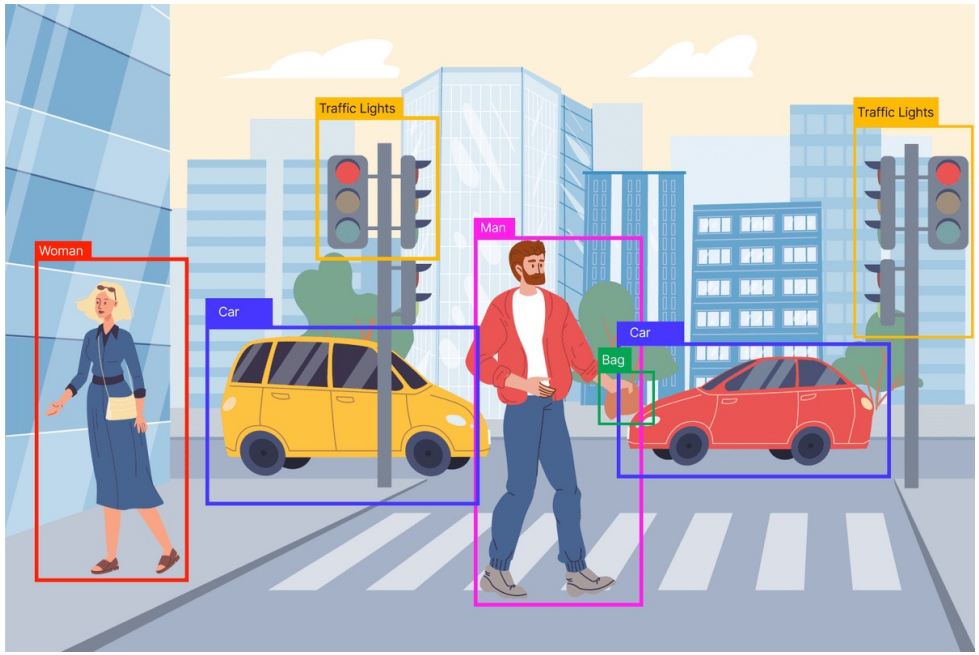
MTH 423

Object Detection



Burak EREN
Yavuz SAYAR
Okan BULGUR

Introduction

In this project, we worked on detecting three types of objects: traffic signs, pedestrians, and vehicles.

The dataset includes .jpg images and labels stored in a JSON file. Our main goal is to train a deep learning model that can find and classify these objects correctly in different images.

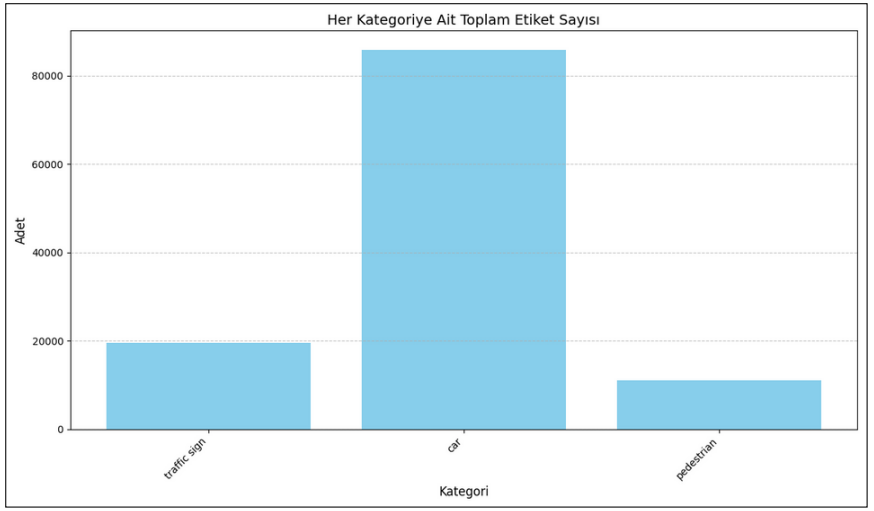
Data Exploration and Understanding

Dataset Analyzes

Our dataset consists only of images in .jpg format. The images are available in two different resolutions: 1280x720 and 1280x960. The label information is in JSON format in a file named “train_data.json”. There are 10,000 images in the training folder and 1,990 images in the test folder.

Class Distribution and Visualization

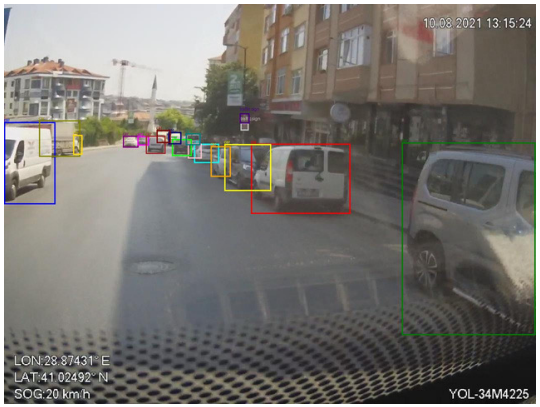
We wrote a Python script to analyze the distribution of class labels. We calculated the number of labels for each class and visualized the results with a graph. As a result of this analysis, we observed that some classes have fewer instances.

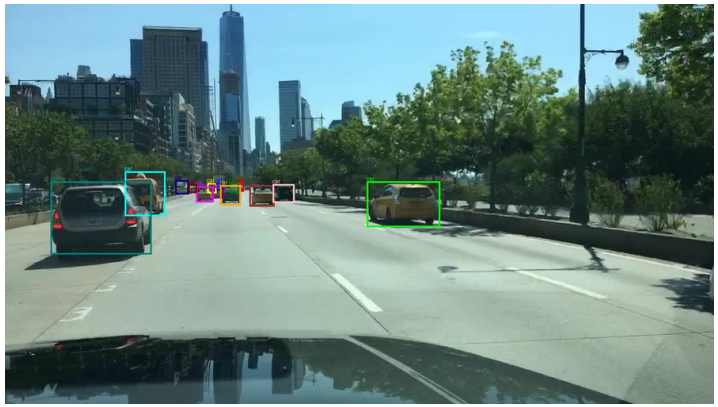


Data Exploration and Understanding

Data Quality and Observations

In our visual inspection, we observed that the labels were correctly placed and the bounding boxes matched the objects. We did not detect blurring or any other quality issues in the images. We also printed out the bounding boxes in several sample images to verify that the classification names were correct and consistent.

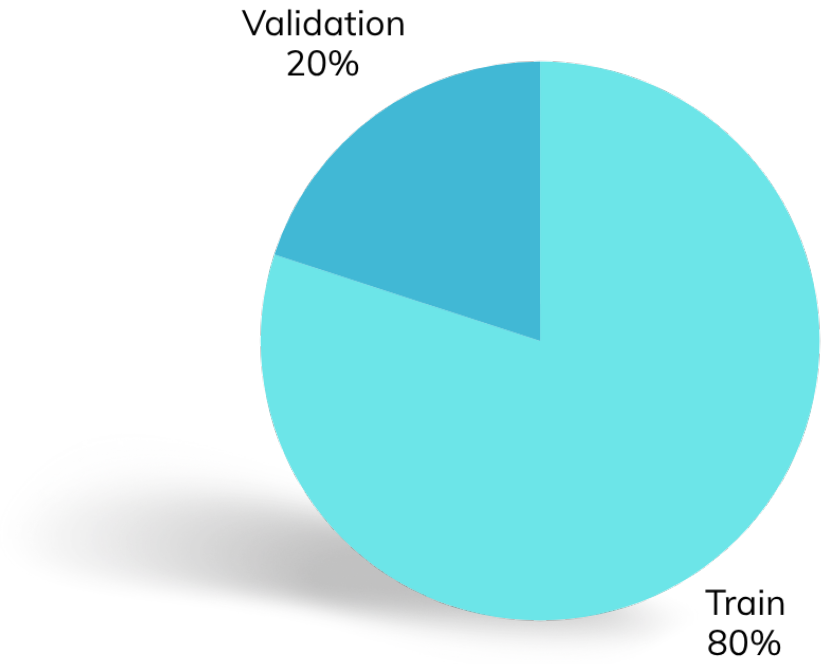




Data Cleaning

- Removed all unlabeled images (“unlabeled images”)
- It avoids the “empty scene” bias, increasing both training time and accuracy.
- This helped the model to focus only on meaningful data.

Data Splitting

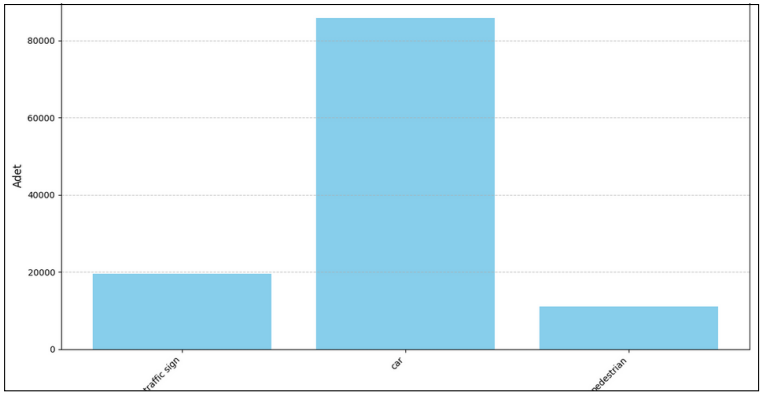


- **7702 images in train**
- **1926 images in validation**

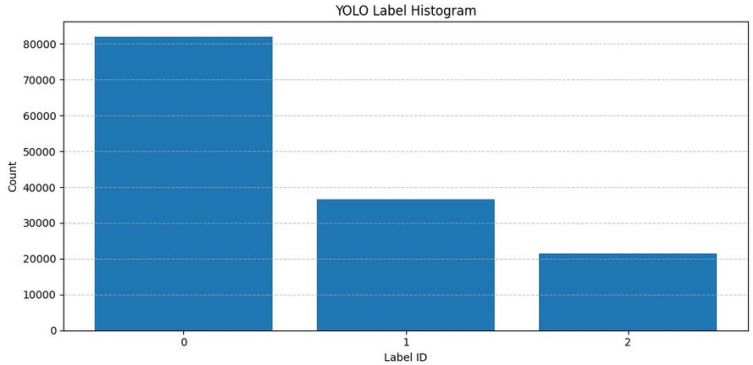
Before data agumentation

Data Augmentation

- HorizontalFlip(p=0.3),
- RandomScale(scale_limit=0.2, p=0.3),
- RandomBrightnessContrast(p=0.3),
- Rotate(limit=10, p=0.3),
- Blur(blur_limit=3, p=0.3),
- RandomGamma(p=0.3),



Object distributions before data agumentation



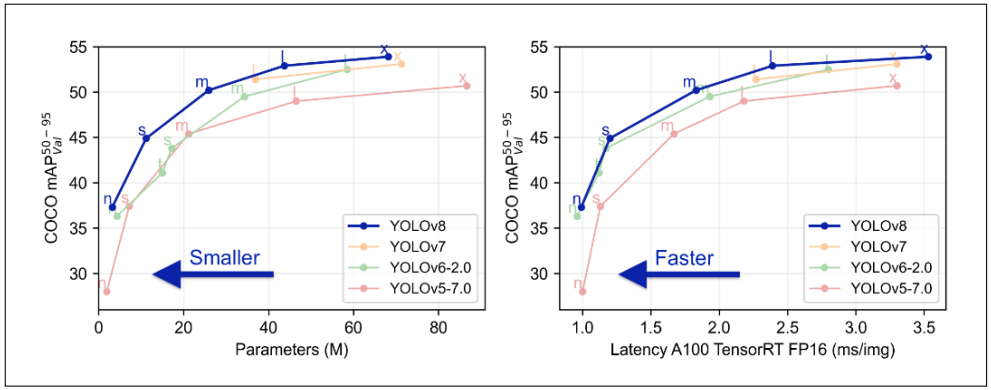
Object distributions after data agumentation

Model Selection

Selected Model and Reasons for Preference

In our project, we used the YOLOv8m model to detect traffic signs, pedestrians and vehicles. The main reasons for choosing the model are as follows:

- High accuracy and successful object detection performance
- Simple and fast training process with its single-stage structure
- Its advanced architecture enables more accurate and detailed object detection than previous versions.

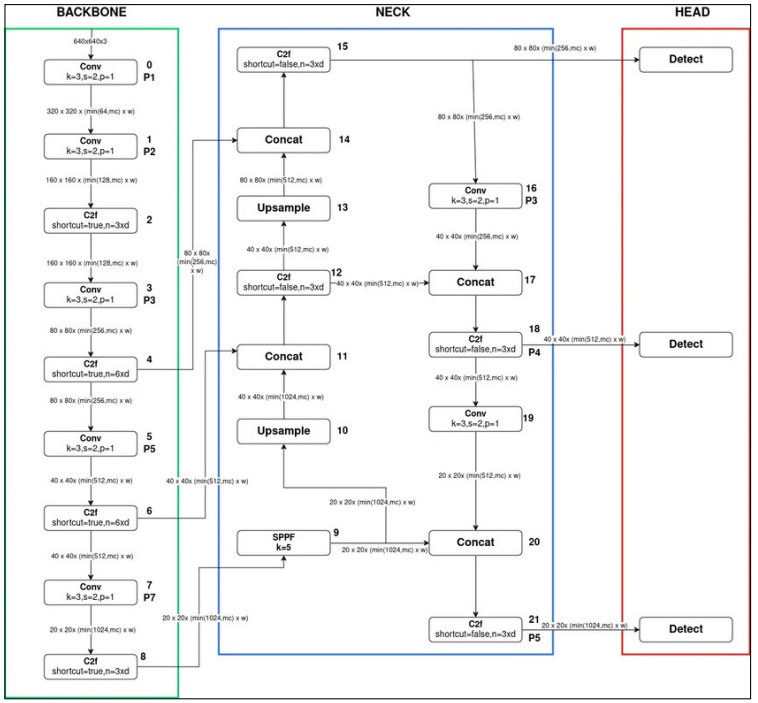


Model Selection

Model Architecture

The YOLOv8m model consists of three main components:

- **Backbone:** This part uses a CNN-based structure to extract basic feature maps from the image.
- **Neck:** It combines different levels of feature maps extracted by the Backbone and enriches them to make them more meaningful.
- **Detection Head:** In the final stage, it uses the processed feature maps to estimate bounding box coordinates, object classes and entity probabilities.

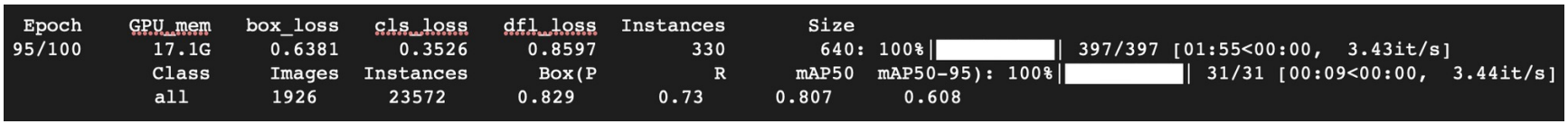


Model Selection

Model Training and Performance Measures

During the training process, we used the default loss functions of YOLOv8. We monitored the learning process of the model by tracking **box_loss** (for bounding boxes), **cls_loss** (for class prediction) and **dfl_loss** (for precision of box coordinates).

We trained the model for **100 epochs**, with a learning rate of **0.001** and a batch size of **32**. We chose Adam as the optimization algorithm. At the end of each epoch, we used the metrics **precision**, **recall**, **mAP@0.5** and **mAP@0.5:0.95** to evaluate the performance of the model.

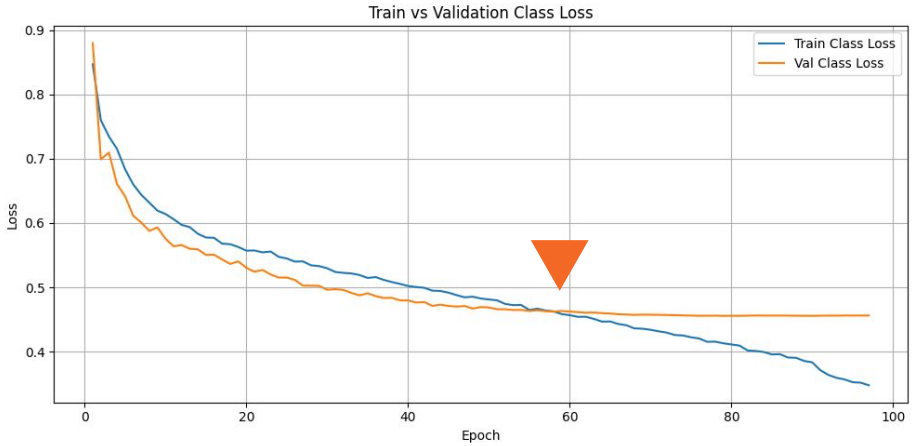


Post-processing Techniques

After processing the model outputs, we took several steps to get the results in the appropriate format and more reliable:

- **Set a confidence score threshold:** We only considered forecasts with a confidence score above 25%, thus eliminating low-probability and incorrect boxes.
- **Limit the Number of Predictions:** For each image, we selected the highest scoring boxes based on the number of predictions expected in the submission format.
- **Normalized Coordinates:** We normalized the bounding box coordinates to the range [0, 1] based on the size of each image.
- **Results are Updated:** We updated the detected class labels and box coordinates by matching them to the corresponding fields in the submission file.

Model Performance Evaluation

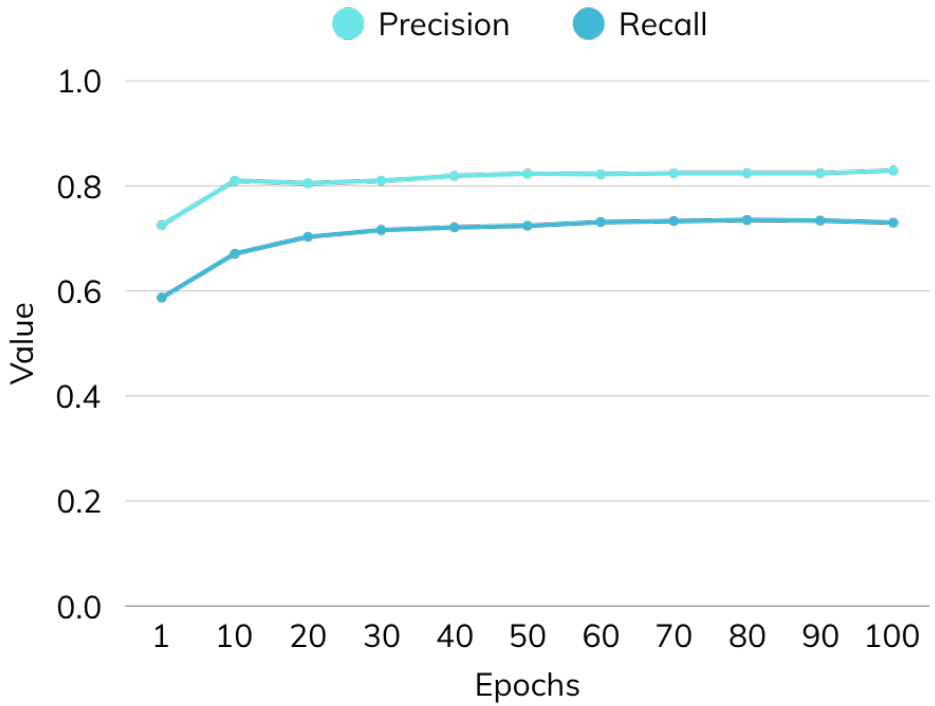


10	**Burak,Okan,Yavuz**		0.56074	20	12h

0.5607

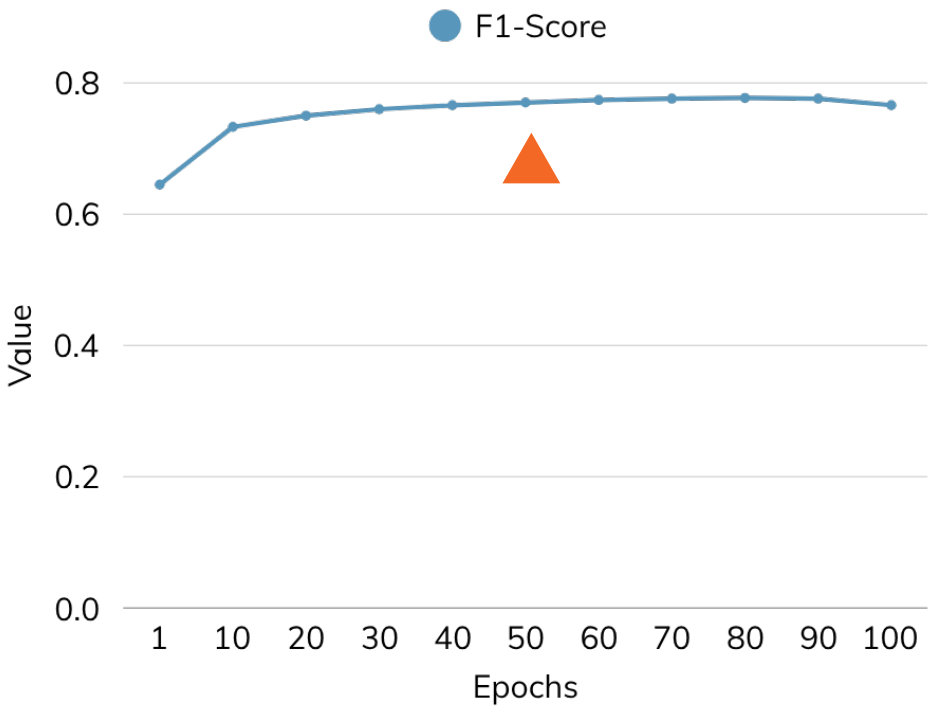
4

Precision & Recall

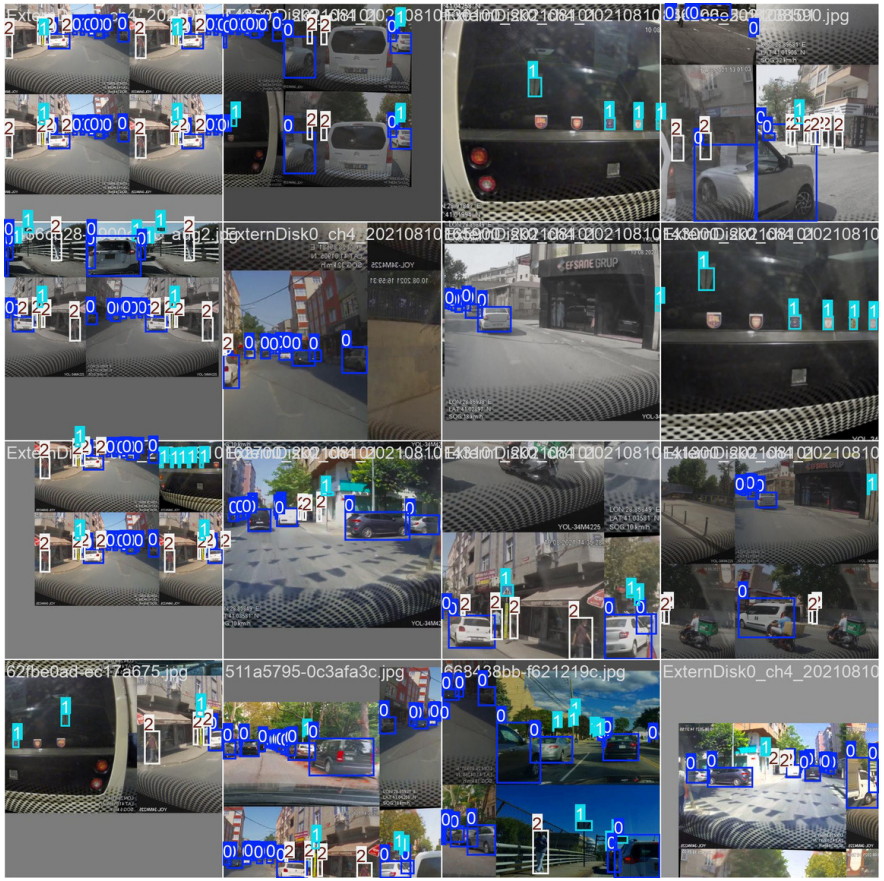


- **recall ≈ 0.73**
- **precision ≈ 0.83**
- Precision and recall values increased steadily over time.
- Both metrics reached a certain level and stabilized.
- No fluctuations observed (no high variance)
- The model has learned in a balanced and stable way.

F1-Score


F1-Score
0.8
0.6
0.4
0.2
0.0
Value
1
10
20
30
40
50
60
70
80
90
100
Epochs

Output Visualization



Prediction: c38cd3f6-bd22367b.jpg

Prediction: c5796880-a7005e6e.jpg

Prediction: ExternDisk0_ch4_20210810162600_20210810162700_2.jpg

Prediction: ExternDisk0_ch4_20210810154800_20210810154900_25.jpg