Machine Learning Engineer Technical Challenge

STROMA VISION

SYSTEM TO DETECT, CLASSIFY AND TRACK FALLING NUTS AND BOLTS

By

Bahadır TATAR

FEBRUARY - 2023

TABLE OF CONTENTS

1. First Impressions And First Ideas For The Challenge

When I received the Technical Challenge files, I first examined the annotations and images to understand how to proceed. When I browsed the annotation files, I saw that they were created with the COCO JSON format. Since I have not worked with the COCO JSON format before, I immediately researched the COCO JSON format.

When I got enough information about the COCO JSON format, I examined the JSON files in detail. Seeing that the (Width x Height) values of the images are (640x640) and the Ground Truth values of each object are given in the file reminded me of YOLO models.

I did a literature review on academic articles on Nuts and Bolts. In an article [1] published on May 25, 2022, I saw the YOLOv5-s and YOLOv5-1 models used when working with Nuts and Bolts. Seeing in this article that the YOLO models outperformed the ResNet50 models, which have great popularity in the literature, I decided to use a YOLO model in this Technical Challenge. Another article [2] on Nuts and Bolts published in February 2023 also influenced this decision.

Considering all these conditions, it was inevitable for me to use a newer technology, YOLOv7 [3].

2. Problems Encountered and Solutions After Deciding to Use a YOLO Model

The first problem I faced after deciding to use YOLOv7 was that I couldn't directly use the COCO JSON format in the YOLOv7 model. To solve this problem, I examined the structures of COCO JSON and YOLO formats and saw a difference as in Figure 2.1.

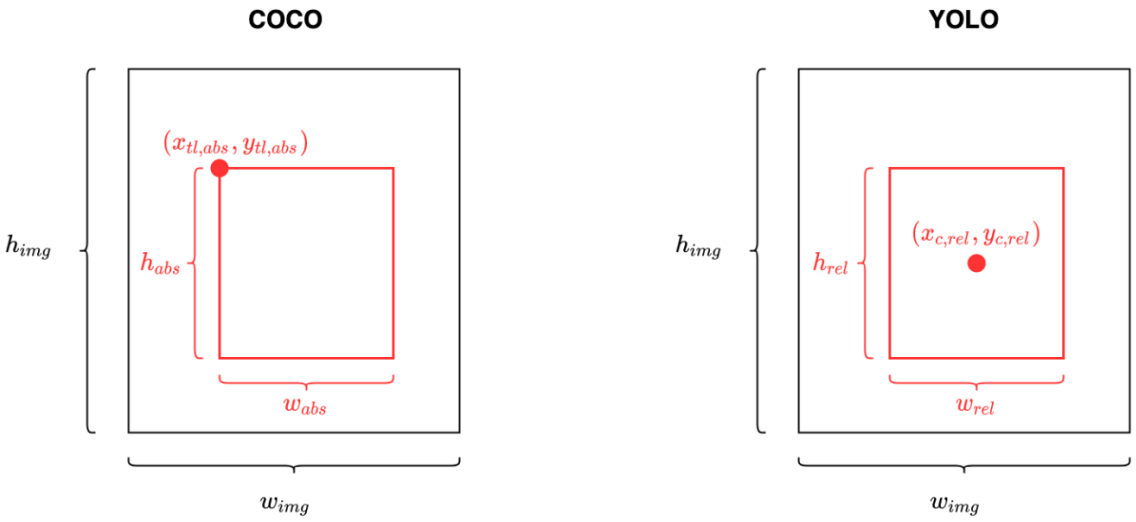

Figure 2.1: COCO vs YOLO format.

In this case, I used the following formulas to convert the COCO JSON format I have to the YOLO format.

$$x_{yolo} = (x_{coco} + \frac{w_{coco}}{2})/w_{img}$$

$$y_{yolo} = (y_{coco} + \frac{h_{coco}}{2})/h_{img}$$

$$w_{yolo} = {}^{w_{coco}}/_{w_{img}}$$

$$h_{yolo} = {}^{h_{coco}}/_{h}$$

While researching the YOLO format, I found that for each image in the dataset, there must be a YOLO format file with the same name in .txt format. For this, I prepared a conversion function in the python file named "COCO JSON to YOLO Format.ipynb". Thus, I have obtained a YOLO format file in .txt format with the same name for each image I have.

After this process, it was time for the second problem I encountered. Although there is information about images with .jpg extension in the COCO JSON files given to me, I did not have these JPG images. Instead, I had .mp4 video files named "test", "train", and "val". Since I couldn't use the videos directly, I had to split these MP4 files into frames as many as the number of .jpg images provided to me in the COCO JSON datasets. So, while I needed to get 7200 frames from the "train.mp4" file, I needed to get 1800 frames from the "val.mp4" file. When I examined the explanations about the Technical Challenge, I saw that each video was 30 FPS. This meant that I could get 240x30=7200 frames from the "train.mp4" file which is 4 minutes long, ie 240 seconds. With the same logic, I could get 60x30=1800 frames from a 60 second long "val.mp4" file.

I prepared a conversion function in a python file named "MP4 to JPG FRAMES.ipynb" to split MP4 files into required frames. Thus, I was able to obtain the required number of frames.

3. Computer Setup and Integrated Development Environment (IDE)

The setup of my personal computer that I used during the whole Technical Challenge is Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz with 16.0 GB of RAM and NVIDA GeForce GTX 950M 4GB of GPU. Unfortunately, my PC was very old and very slow, which cost me a lot of time. I decided to use Google Colab to provide myself with the healthiest environment. This is how I decided to solve both my PC's low memory issue and low GPU issue. Thanks to Google Colab, I was able to access the Tesla T4 GPU.

4. Python Libraries and Versions Used in YOLOv7's Training

In Table 4.1 you can review the libraries and their versions required to train YOLOv7.

Table 4.1.: Version control table

Python Library	Version
matplotlib	>= 3.2.2
numpy	>=1.18.5,<1.24.0
opencv	>=4.1.1
Pillow	>=7.1.2
PyYAML	>=5.3.1
requests	>=2.23.0
scipy	>=1.4.1
torch	>=1.7.0,!=1.12.0
torchvision	>=0.8.1,!=0.13.0
tqdm	>=4.41.0
protobuf	<4.21.3
tensorboard	>=2.4.1
pandas	>=1.1.4
seaborn	>=0.11.0

5. Preparation for YOLOv7's Training

118287 training images and 5000 validation images from a total of 80 classes were used in the training of original YOLOv7 from the article. In this Technical Challenge, 7200 training images and 1800 validation images from 2 classes were used.

Figure 5.1 is the coco.yaml file showing the 80 classes used in the original YOLOv7 article. But to train YOLOv7 for this Technical Challenge, the coco.yaml file should be updated as in Figure 5.2.

Figure 5.1: coco.yaml file of original YOLOv7 from the article.

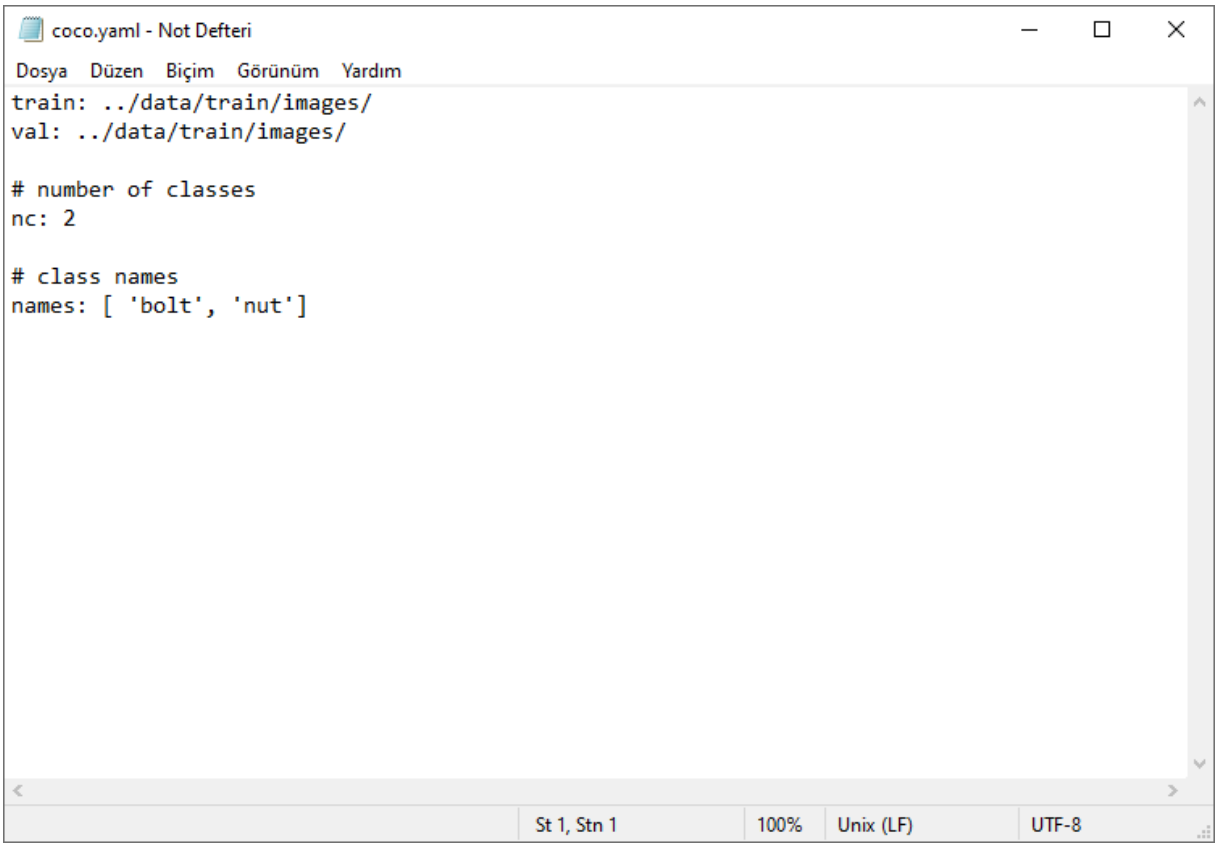

Figure 5.2: coco.yaml file used for Technical Challenge to train YOLOv7.

6. Model Training and Results

With all the conditions met for training YOLOv7 using the Nuts and Bolts Dataset, it's time to determine the required Epoch and Batch Size values. Although I wanted to train the model in 64 Batch Size, the setup provided by Google Colab allowed me to train YOLOv7 in 16 Batch Size. However, the Epoch value was set to 25. Considering that the model with 25 Epochs takes about 4 hours to train, a higher Epoch value at the same Batch Size would result in a much longer training session.

The results at the end of the training are as in the Figures below.

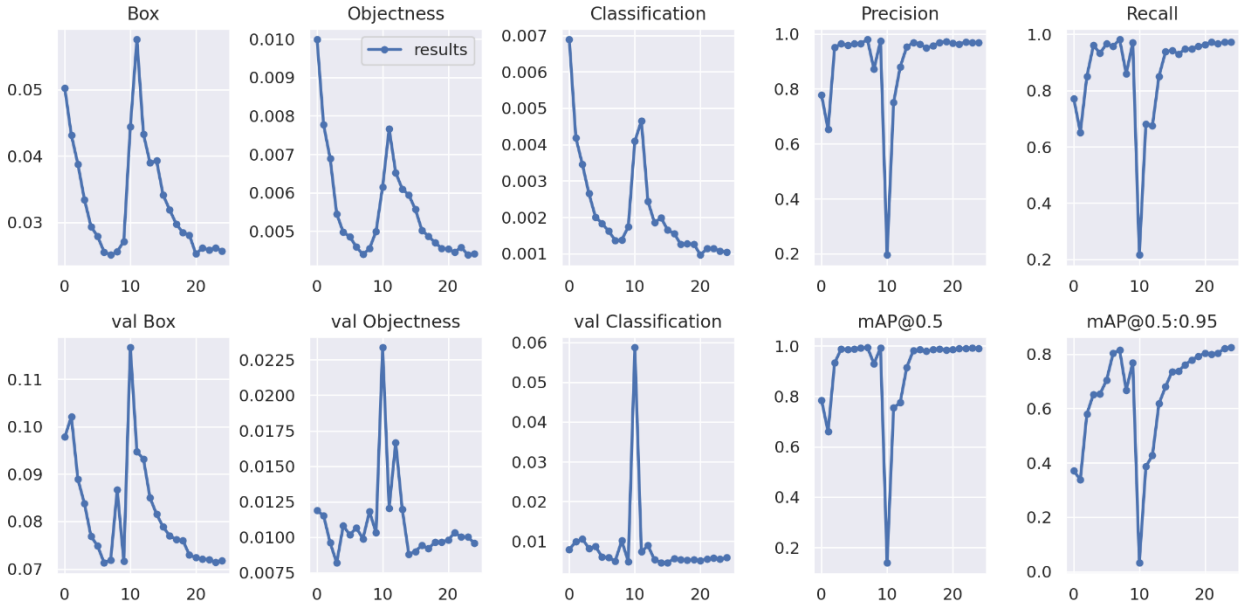

Figure 6.1: Overall training results.

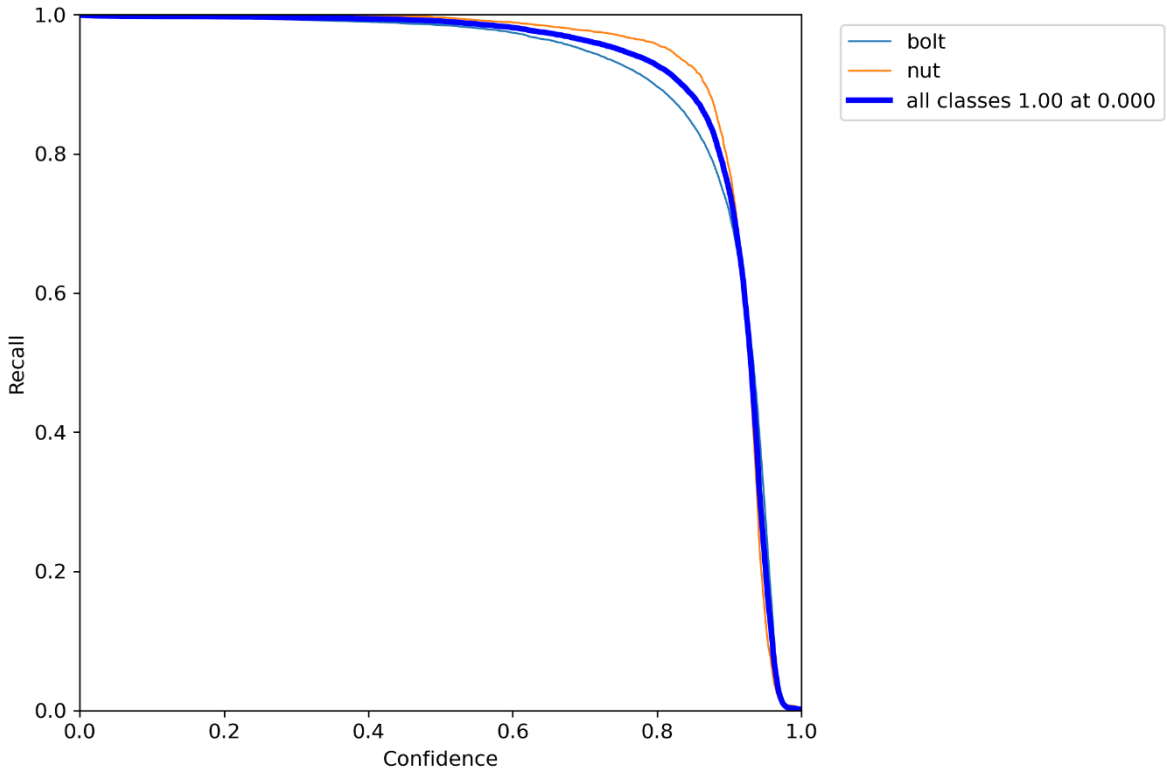

Figure 6.2: Recall curve.

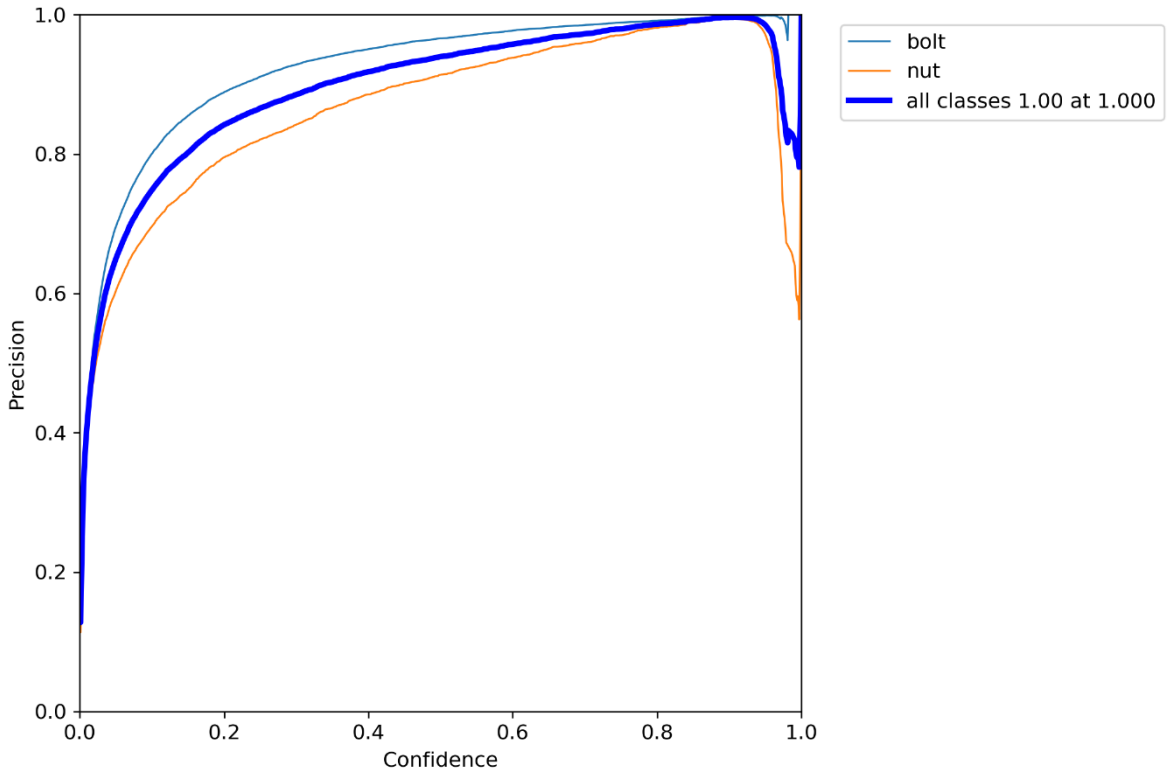

Figure 6.3: Precision curve.

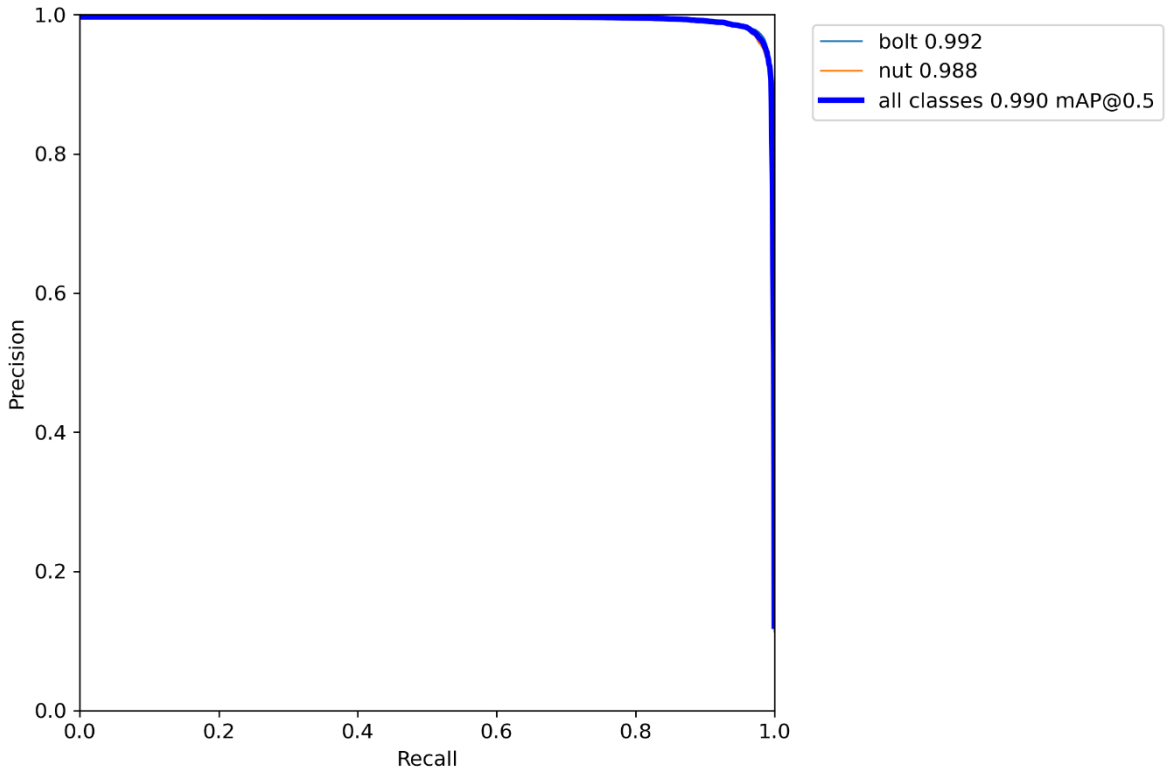

Figure 6.4: Precision- Recall curve.

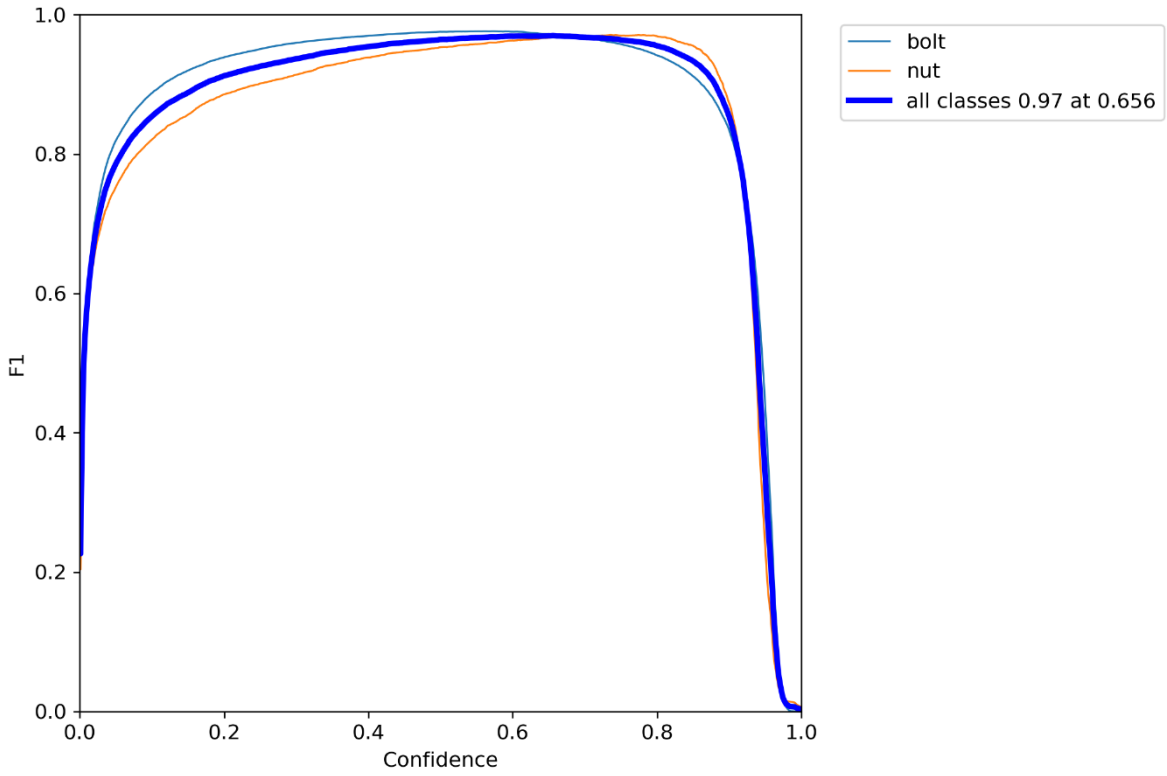

Figure 6.5: F1-Score curve.

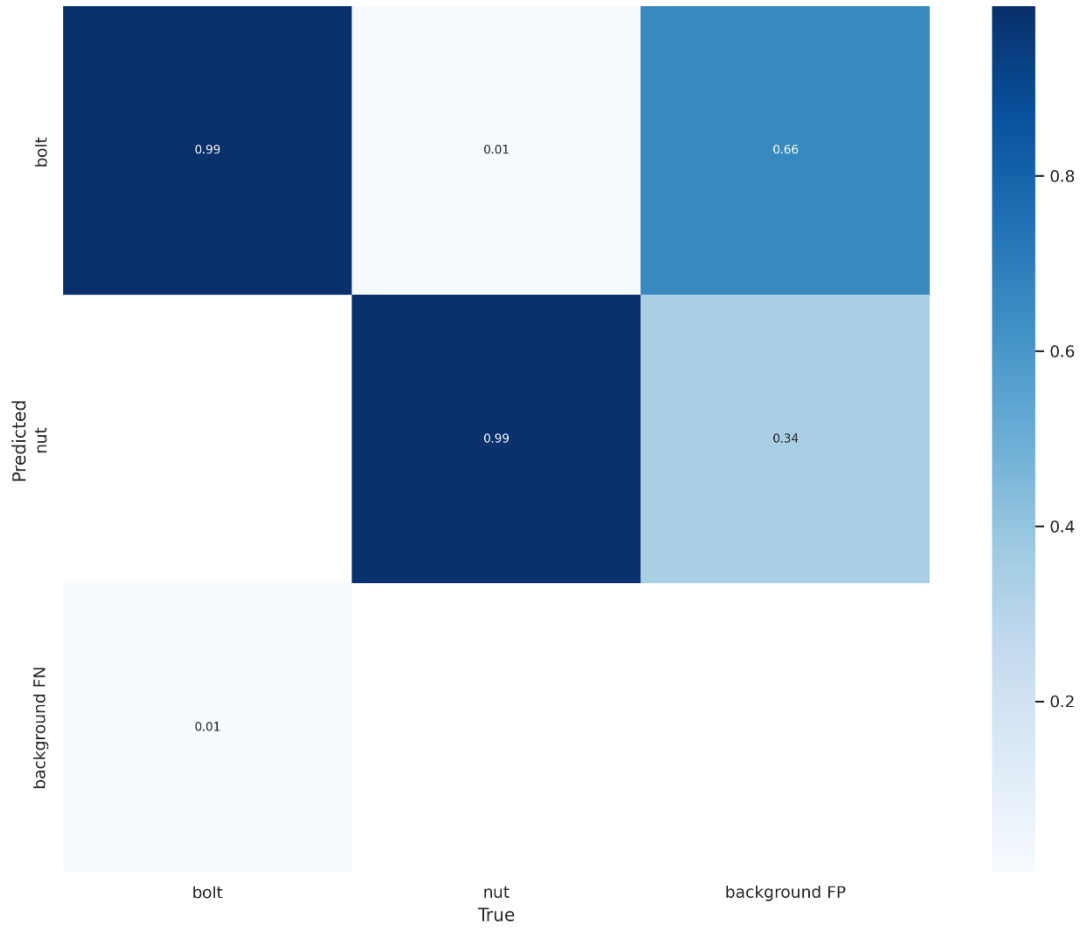

Figure 6.6: Confusion matrix.

Considering all these training results, although the YOLOv7 model trained with Nuts and Bolts Dataset under current conditions proves that it can handle simple tasks, it is seen that it needs a better training in much more difficult tasks.

7. Detection and Classification

Figure 7.1 and Figure 7.2 show the actual labels and predictions of the first test batch, respectively.

Figure 7.1: Actual labels of the first test batch.

Figure 7.2: Predictions of the first test batch.

8. Tracking and Counting

Figure 8.1, Figure 8.2, and Figure 8.3 show the tracking and counting capabilities of the YOLOv7 model trained with the Nuts and Bolts Dataset.

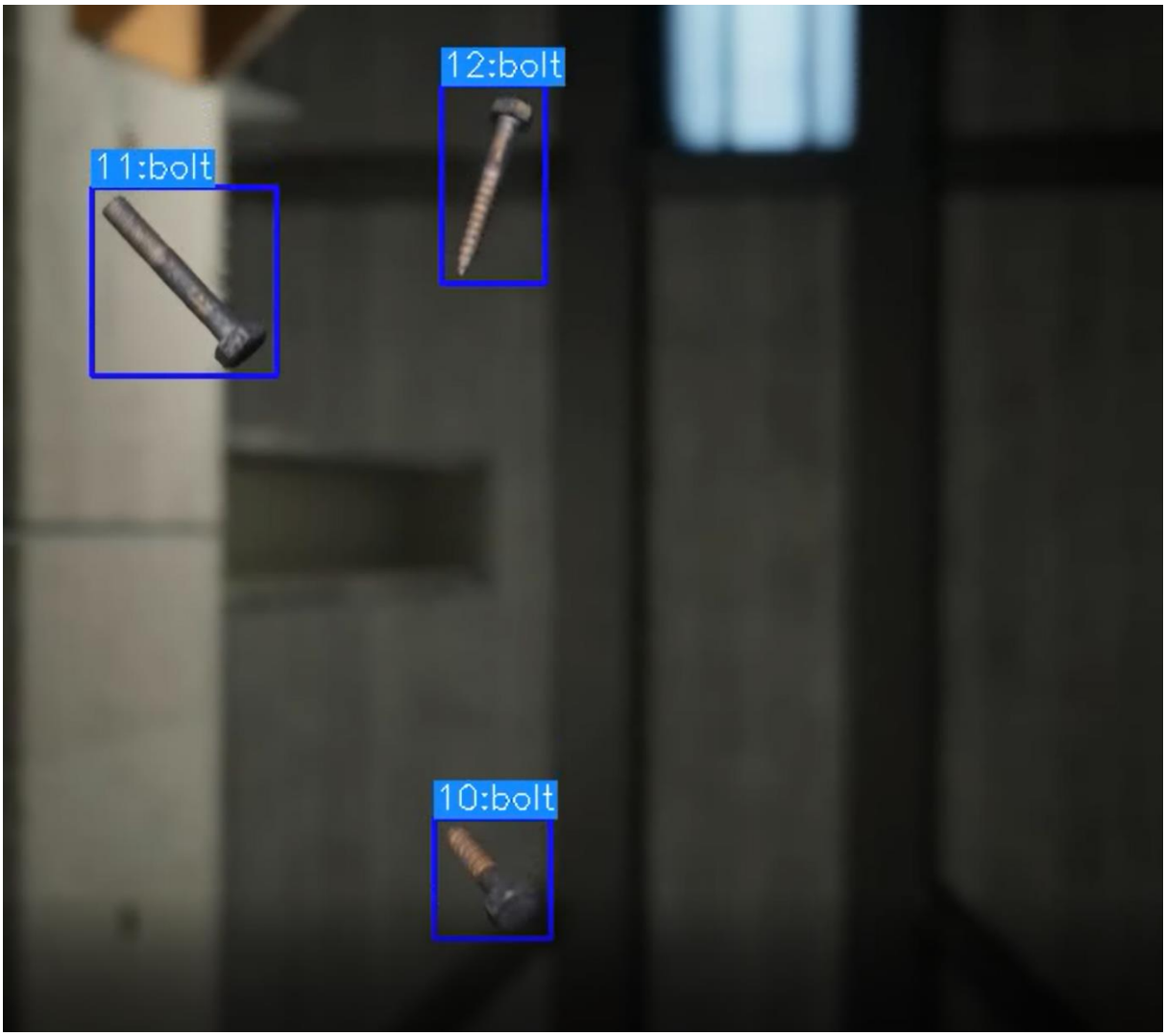

Figure 8.1: Counting capabilities of the YOLOv7 – 3 Bolts.

Figure 8.2: Counting capabilities of the YOLOv7- 2 Bolts – 1 Nut.

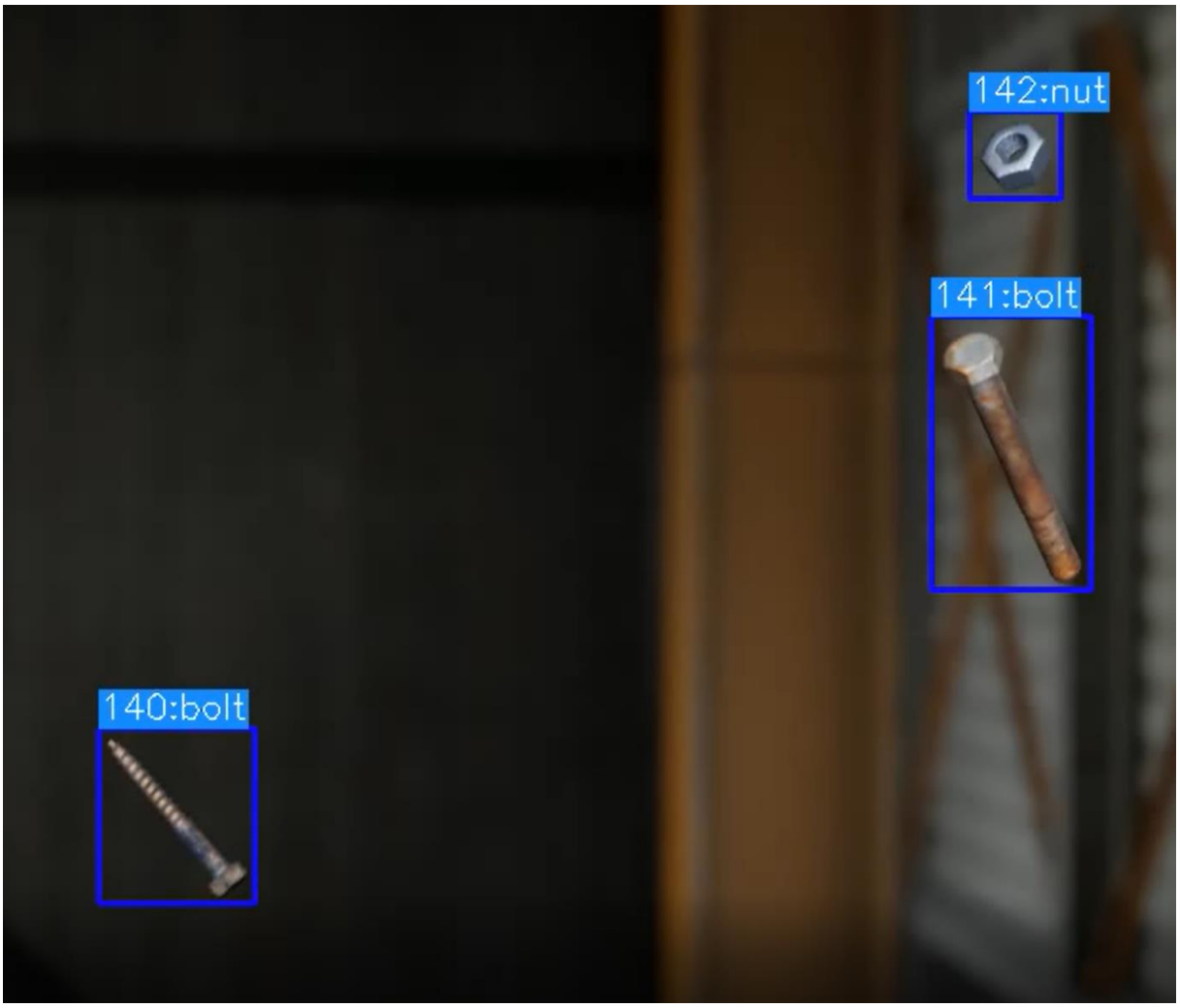

Figure 8.3: Counting capabilities of the YOLOv7- 2 Bolts – 1 Nut.

9. Conclusion

When the whole process of Technical Challenge is evaluated, it can be said that YOLOv7 gives good results. From an objective point of view, it is seen that the YOLOv7 model is a model open to development. In my opinion, it can be said that the detection and classification capabilities of the created model are better than the counting capabilities.

As a result, it is seen that the YOLOv7 model trained using Nuts and Bolts Dataset has the desired features.

10. References

[1] Y. Zhao, Z. Yang and C. Xu, "NPU-BOLT: A Dataset for Bolt Object Detection in Natural Scene Images," 2022.

[2] F. Mushtaq, K. Ramesh, S. Deshmukh, T. Ray, C. Parimi, P. Tandon and P. Kumar Jha, "Nuts&bolts: YOLO-v5 and image processing based component identification system," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105665, 2023.

[3] C.-Y. Wang, A. Bochkovskiy ve H.-Y. M. Liao, «YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,» 2022.