Analog Time Detector

Ana Pinto 202105085
Pedro Leitão 202107852

INTRODUCTION

This project is centered on detecting and interpreting the time displayed on analog clock faces using computer vision techniques. It focuses on processing images of clocks to identify key elements such as the clock's circle, hour hand, minute hand, second hand (if present), and the position of the number 12. The ultimate goal is to accurately determine and report the time shown on the clock.

DATASETS

Watche Image Dataset

A Dataset of Watches

Simple Analog Clock (Monochrome)

Custom images from watch stores and personal photographs.

TECHNOLOGY STACK

Python

YOLOv8/YOLOv11n

OpenCV

Tkinter

MOTIVATION

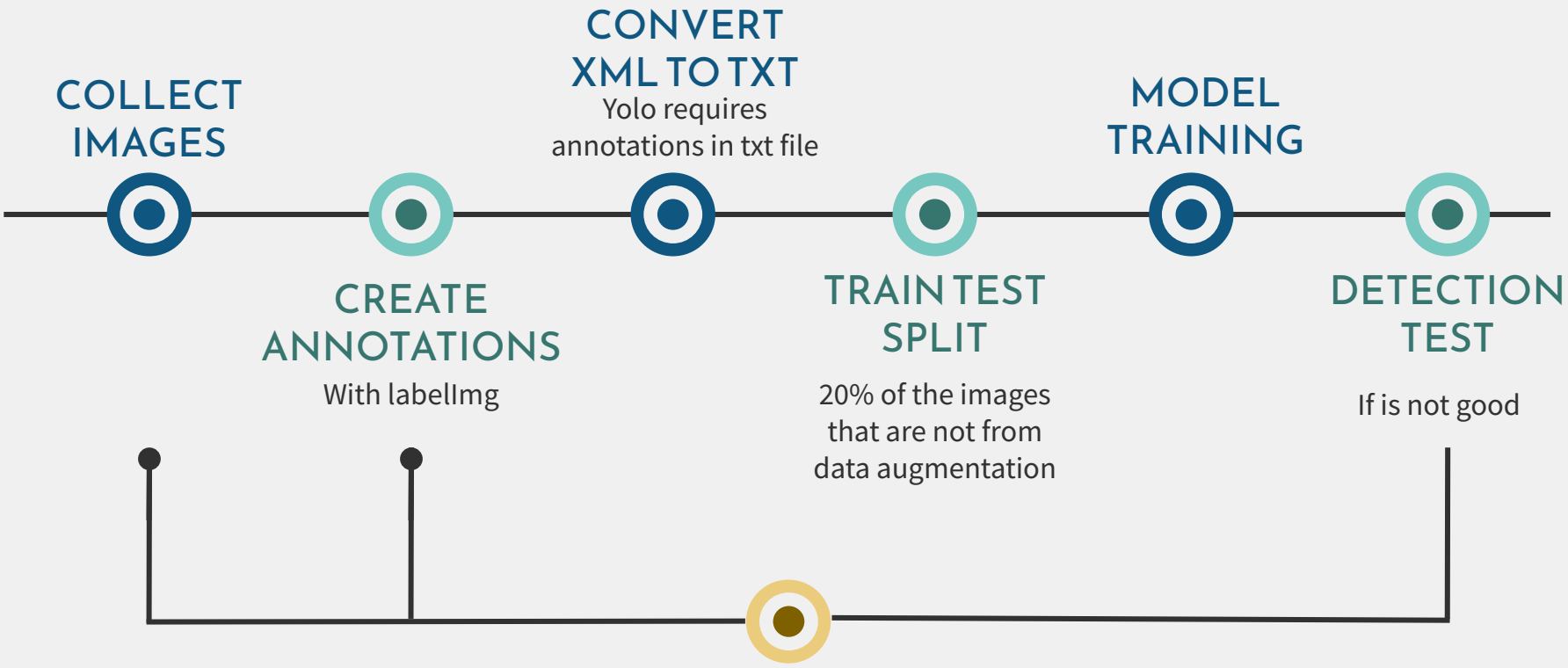
This project provides an excellent opportunity to apply the knowledge and skills acquired in the Computer Vision course to a practical and challenging problem: detecting and interpreting the time displayed on analog clocks. It bridges theory and practice by integrating key concepts and techniques such as:

- **Object Detection**: Implementing YOLO models to identify and extract key components of the clock.
- **Hyperparameter Tuning:** Optimizing model performance by adjusting parameters like learning rate, momentum, and weight decay to improve detection accuracy.
- **Data Augmentation:** Enhancing the dataset by applying transformations like rotations, scaling, and brightness adjustments to improve model robustness and generalization.

WHY YOLOv8?

We opted for YOLOv8 because traditional CNNs not only require manual design and tuning but also carry a higher risk of overfitting, especially when dealing with datasets tailored to specific tasks. Building a custom network would have been even more challenging, as our problem involves only 5 classes, making it harder to generalize effectively. While newer versions like YOLO11 exist, our tests showed that YOLOv8 provided the best balance of speed, accuracy, and generalization for our needs, making it the ideal choice.

PIPELINE - TRAINING



DATA AUGMENTATION

01 | ROTATION

02 | BLUR

03 | RESIZE

DATA AUGMENTATION - ROTATION

The first few times we trained the model with 500 images, we saw that the model always detected the 12 vertically even if the clock was in another direction. To solve this problem, we rotated the images we had by 90 degrees, -90 degrees, 180 degrees and some at different angles.

Before

Image 1. Detection without data augmentation

After

Image 2. Detection with data augmentation

DATA AUGMENTATION - BLUR

We experimented with applying blur to one image to test the model's robustness, but it performed poorly in detecting the watch elements. After applying data augmentation, the model's detection capabilities improved significantly.

Before

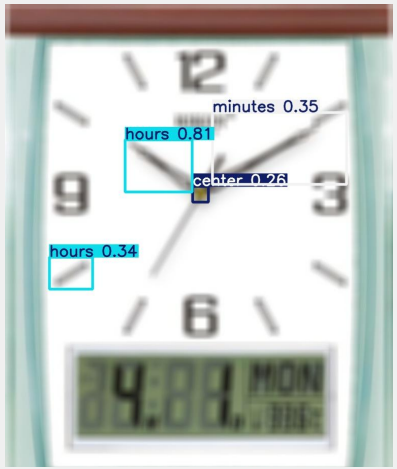


Image 3. Detection without data augmentation

After

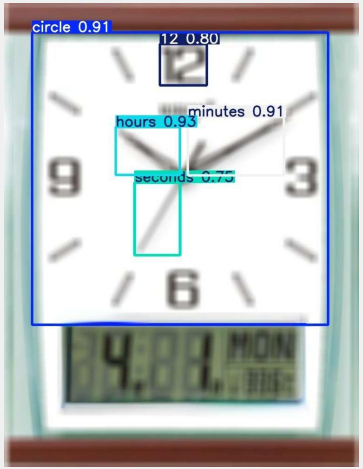


Image 4. Detection with data augmentation

DATA AUGMENTATION - RESIZE

To address the model's difficulty in identifying the watch hands in images where they appeared more distant, we applied targeted data augmentation. By resizing the images to a standard size and then padding them with a black border, we ensured consistent input dimensions and improved the model's ability to generalize across varying scales and distances.

Before

Image 5. Detection without data augmentation

After

Image 6. Detection with data augmentation

EXPANDING DATASET

We noticed that most of our images had clocks set to similar times, like 10:10:00 and 10:10:30, which introduced bias. To fix this, we added more images with varied clock times, ensuring a more diverse and balanced dataset.

Before

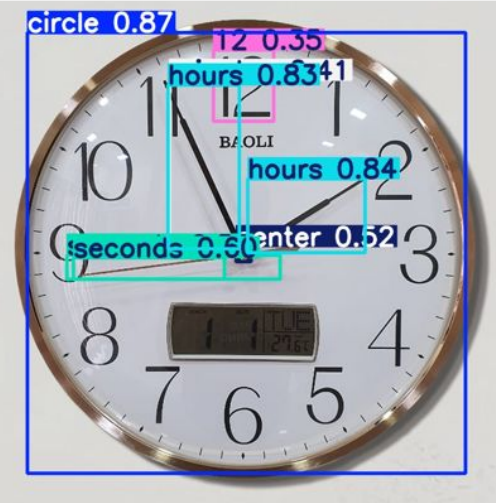


Image 7. Detection without the new images.

After

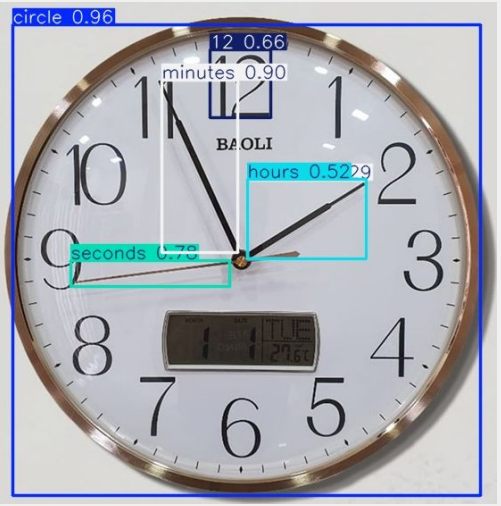


Image 8. Detection with new images

HYPERPARAMETER TUNING

We used YOLO's tune function to test various hyperparameters, conducting 10 iterations with 25 epochs using the AdamW optimizer, in order to evaluate the impact of these settings on the model's performance.

Best hyperparameters:

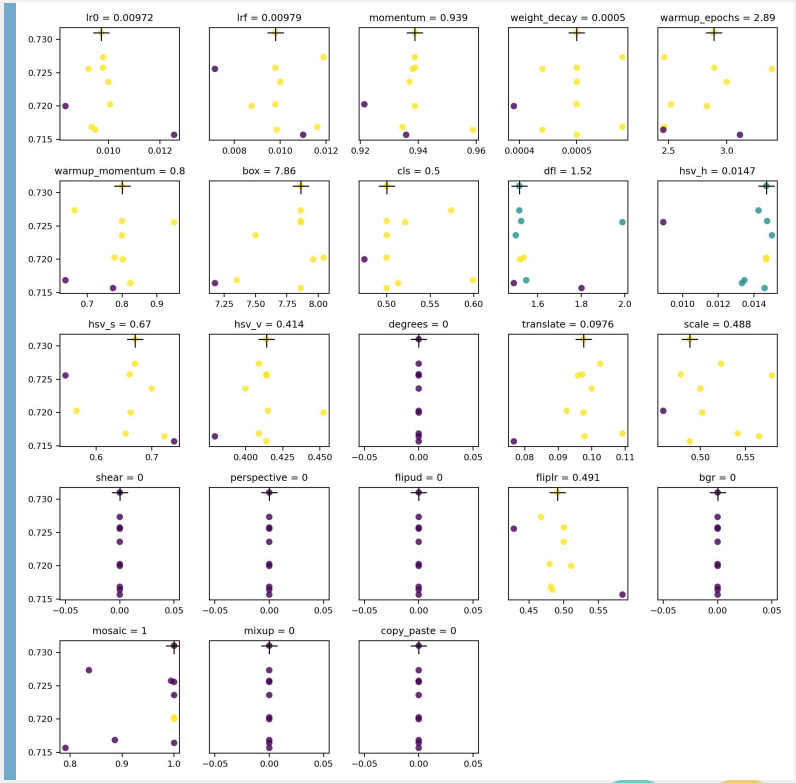


Image 9.Results of the hyperparameter tuning.

BEST GRID

We tested 3 different grids to optimize the YOLO model performance, focusing on: Image Size (imgsz), Batch Size (batch).

To extract personalised Anchors, we carried out a statistical analysis of 16,071 bounding boxes from the dataset, using K-means clustering to generate 9 anchors that optimally represent the various sizes and proportions of the objects, normalising the dimensions to avoid distortions.

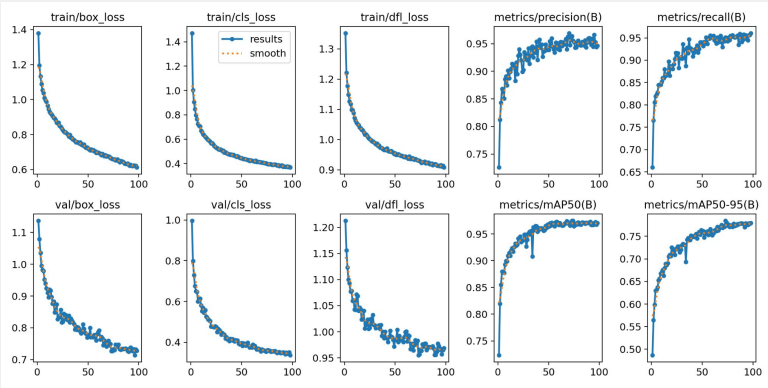


Image 10.Results with the imgsz:640 , batch:16

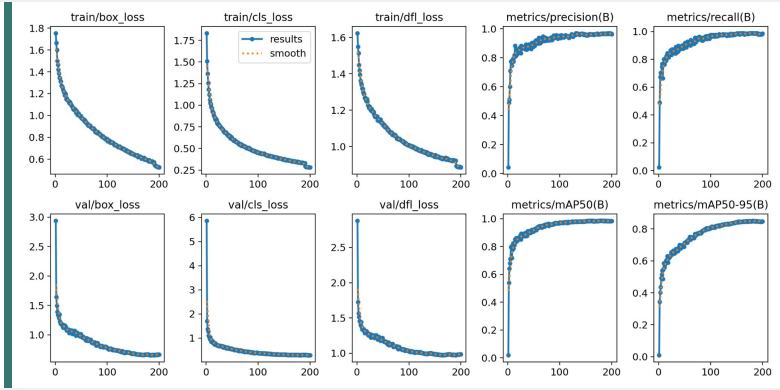


Image 11.Results with the imgsz:768, batch:16

FINAL MODEL

MODEL:	YOLOv8s
IMGSZ:	768
BATCH:	16
OPTIMIZER:	AdamW
EPOCHS:	200
PATIENCE:	20
LR0:	0.00975
MOMENTUM:	0.93883
WEIGHT_DECAY:	0.0005

ANCHORS:
- [0.60,0.58] - [0.47,0.47]
- [0.05,0.05] - [0.14,0.08]
- [0.24,0.15] - [0.07,0.13]
- [0.73,0.72]
- [0.10,0.25]
- [0.36,0.35]

WARMUP_EPOCHS:	2.88938
WARMUM_MOMENTUM:	0.8

Ultralytics 8.3.40 🚀 Python-3.8.9 torch-2.2.2+cu118 CUDA:0 (NVIDIA GeForce RTX 2060, 6144MiB)

FINAL MODEL

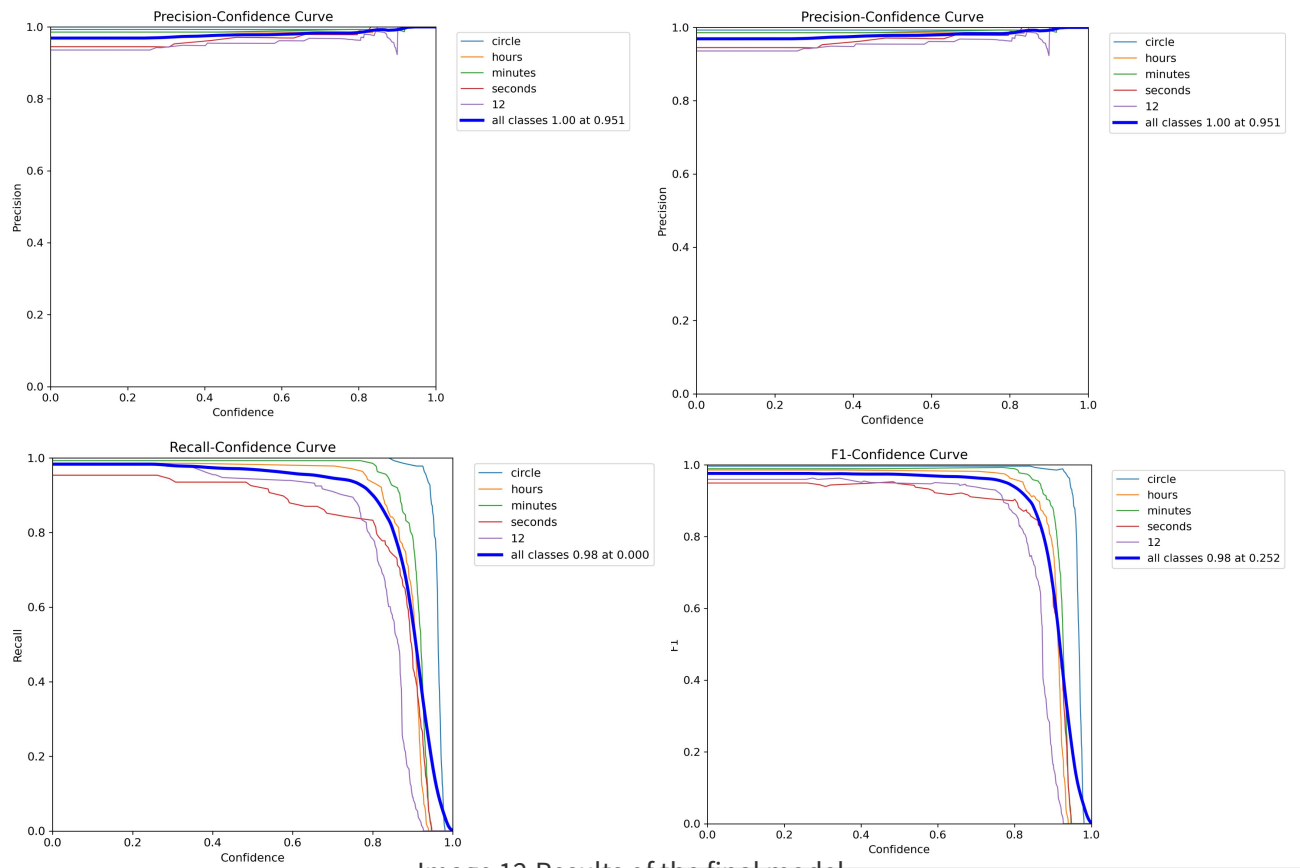


Image 12.Results of the final model.

FINAL MODEL

Before Best Grid

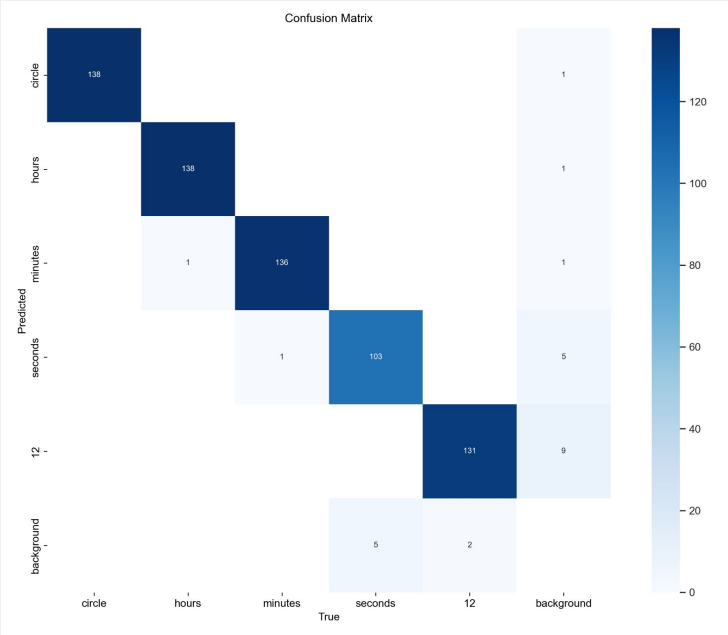


Image 13. Validation Confusion Matrix before the best grid.

After Best Grid

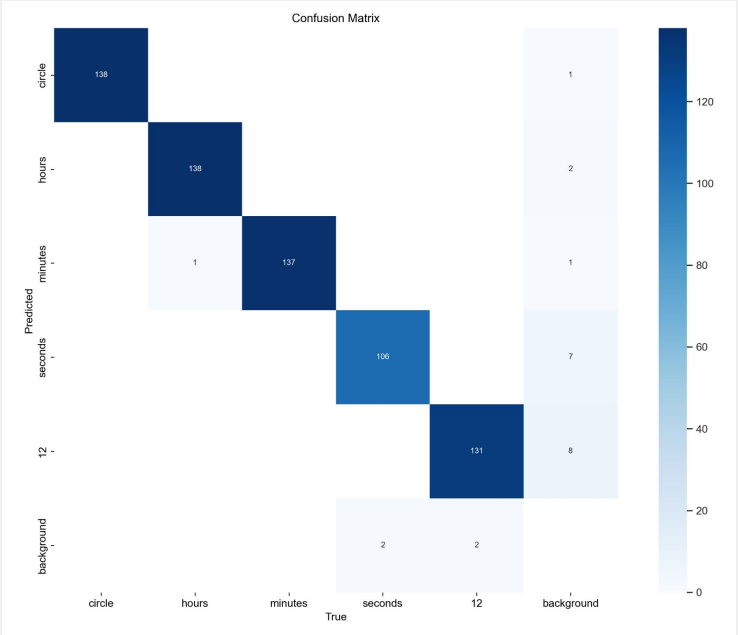


Image 14. Validation Confusion Matrix after the best grid.

FINAL MODEL

After training with the best hyperparameters and the best grid, the model's detections were better.

Before

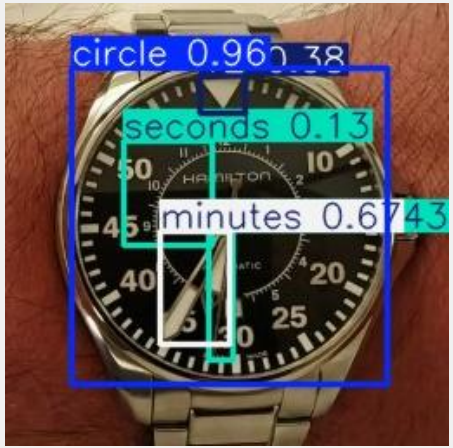
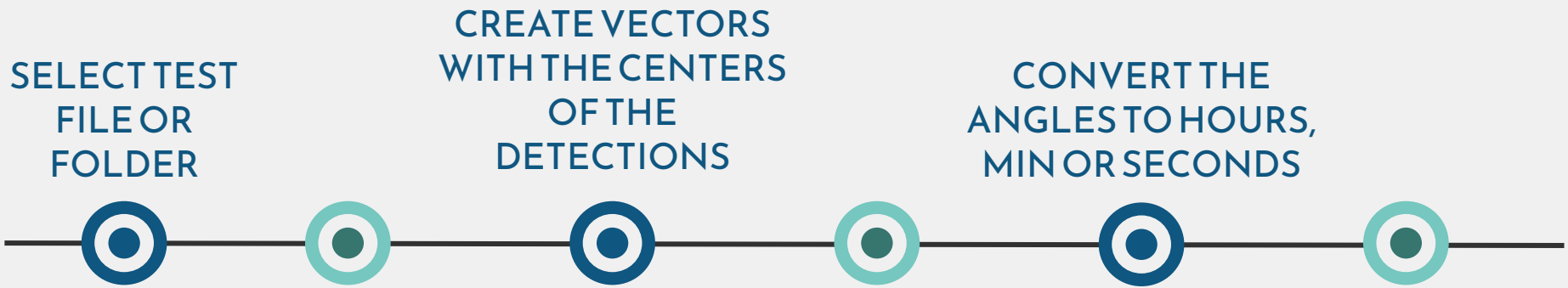


Image 15. Detection Example before the final model.

After

Image 16. Detection Example with the final model.

FINAL PIPELINE



TRANSFORM DETECTIONS TO TIME

01 CENTER POINT OF BOUNDING BOXES

$x = x\_min + x\_max / 2$

$y = y\_min + y\_max / 2$

02 VECTORS FROM THE CENTER

vector=(pointx−centerx, pointy−centery)

03 ANGLE OF EACH VECTOR ACCORDING TO '12'

angle=atan2(vectory,vectorx)

relative_angle=angle_hand−angle_12

04 CONVERT ANGLE TO TIME

$$h = \frac{h\_angle}{30} \qquad m = \frac{m\_angle}{6}$$

TO GET THE DETECTED TIME, WE NEED

CIRCLE

To calculate the center of the watch.

HOURS

To have some result.

‘12’

To calculate the angles.

When it doesn't find one of these labels, it zooms in on the image from the circle. If it still can't find it, it returns fail.

EXAMPLES OF DETECTIONS

Nice Detection

Image 17. Example of a good detection.

Mid Detection

Image 18. Example of a detection with error.

Failed Detection

Image 19. Example of a failed detection.

RESULTS - BOUNDING BOXES

Deviation=
$|T_{real} - T_{detected}|$

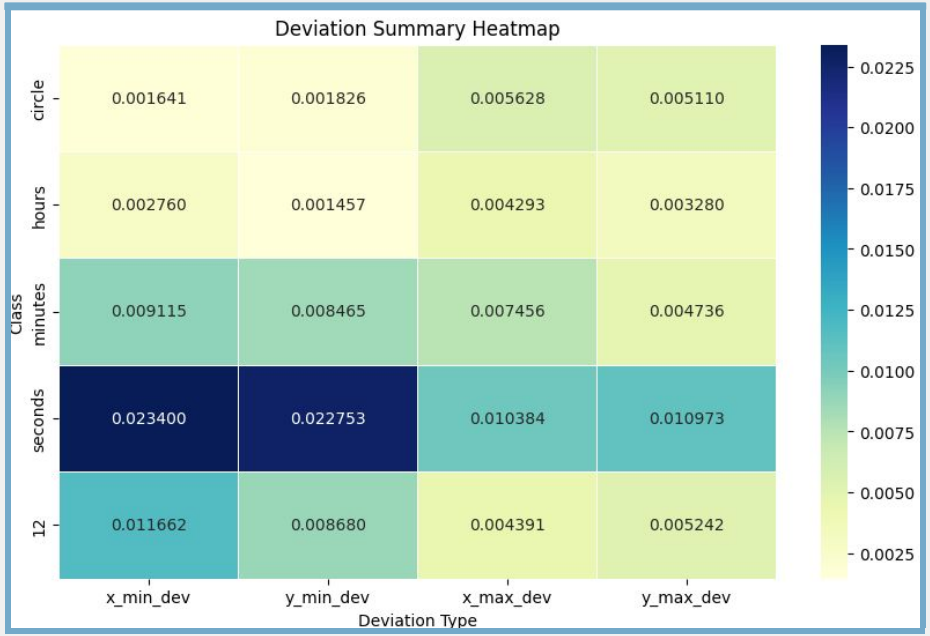


Image 20. Heatmap of the average deviation of each class.

RESULTS - BOUNDING BOXES

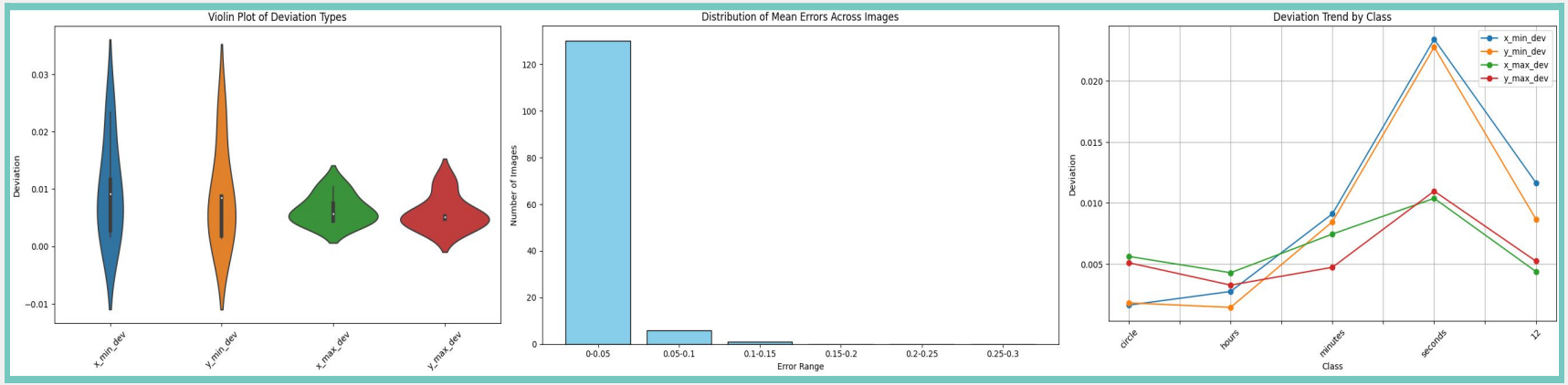


Image 21.The graph on the left shows the distribution of variations (x min, y min, x max, y max) by class. In the centre, the histogram indicates that most of the errors are between 0 and 0.05. On the right, the line graph shows greater variations in the seconds class.

RESULTS - TIME DETECTED

Validation set with 138 images:

Perfect Detection	Good Detection	Mid Detection	Bad Detection	Detection Failed
23	97	2	11	0

Higher deviation compared to ground truth: 3660 s -> 1.01 h

According to the confusion matrix, the model failed to detect the label "12" in 7 instances. Ideally, this should have resulted in 7 instances of non-detection. However, by using a lower confidence threshold during the detection process, the model produced detections that may not have been entirely reliable.

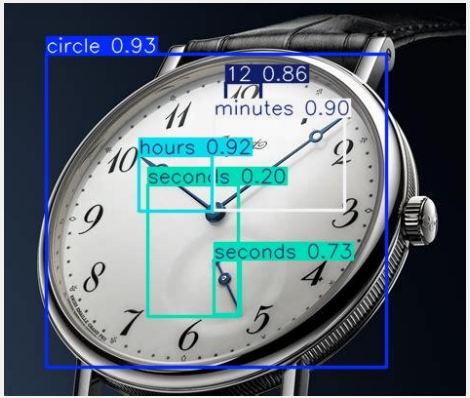


Image 22. Image with the highest devio in the validation set

RESULTS - TIME DETECTED

Test set with 118 images taken from reddit, X and some taken by us:

Perfect Detection	Good Detection	Mid Detection	Bad Detection	Detection Failed
7	55	20	24	17

Higher deviation compared to ground truth: 31436 s -> 8.73 h

Image 23. Detection and real image of the image with the higher deviation.

Detection Failed vs Good Detection

Image 24. Example of the error that the same watch at a different angle can cause to the model

FINAL THOUGHTS AND FUTURE WORK

Our initial exploration into analog watch time detection revealed the complexities of computer vision challenges. By leveraging YOLOv8 and implementing strategic data augmentation, we developed a model that can interpret watch faces across varied orientations and designs. Future work will focus on expanding the dataset to deal with our current problems and optimizing the detections with watches in different angles.

References

ULTRALYTICS. **YOLOv8 Overview**. https://docs.ultralytics.com/pt/models/yolov8/#overview. Accessed 07th December

ROBOFLOW. **How YOLO Works**. https://blog.roboflow.com/how-yolo-works/. Accessed 25th November.

EESA, Tarek. **Latest YOLOv8 & YOLOv9 Guide for Hyperparameter Tuning and Data Augmentation (2024)**. Medium.https://medium.com/@tarekeesa7/latest-yolov8-yolov9-guide-for-hyperparameter-tuning-and-data-augmentation-2024-469c69f295e0. Accessed 5th December