REAL TIME FACE DETECTION IN VIDEOS

Project Report for
Computer Vision DS5602 Course

Submitted by
Jagadeesh Thogata 112001045
Ninad Chavan 142202021

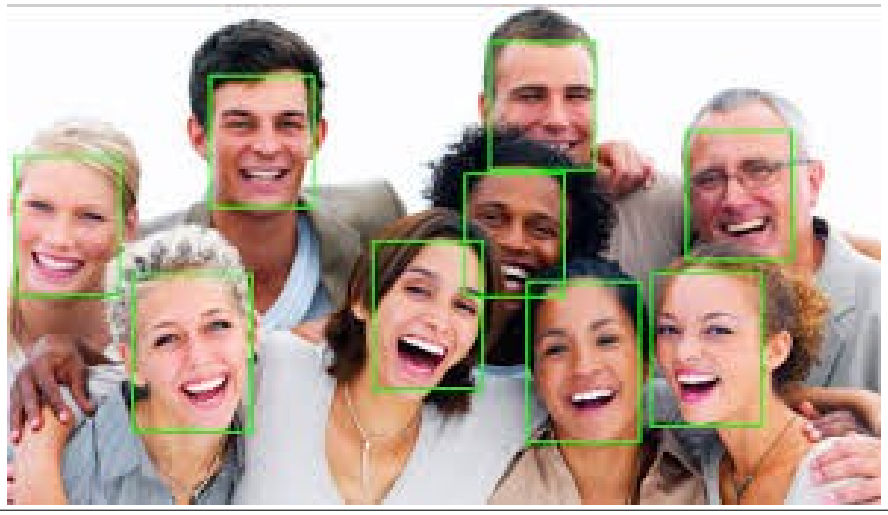
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

Contents

1 Introduction

- Task is to build a model for the Object Detection Model and deploy it in real time. Here the object to be detected is face. Object Detection involves identifying the object along with their locations. Real-time face detection finds extensive use in general surveillance systems for identifying and tracking individuals in public spaces, enhancing security and public safety.
- For localization, the model uses bounding boxes to locate the faces in the image. In our case, only Single class object detection. No other class other than face should be detected. But Models are built for detecting both single and multiple faces in the image. Dataset used is the Wider Face Dataset.
- We have used different models,including Regional Based Convolutional Neural Networks and Single Stage Networks. We used both approaches which includes training pretrained models from scratch and also directly using pretrained models weights.
- Finally, based on evaluation metrics we have compared build models on different approaches and ranked them in this particular task of face detection.

Figure 1 Face Detection



2 About the Dataset

- Dataset used is Wider face Dataset - A Benchmark Dataset for face detection. This dataset contains 32203 images and 393703 faces. This dataset contains images and faces which have a high degree of variability in scale, pose and occlusion.
- The diversity of the dataset is huge. The wider dataset includes faces from 60 event categories. e.g. Tennis, Parade, Greeting, Swimming, Group etc. These event categories are further divided into 3 sets Easy, Medium and Hard, based on the ease of detection.

- We will be only working with Easy set for the scope of this project, as large number of computational resources will be required as we are also training the models from scratch. The dataset has train, validation and annotation files. The count of train and validation images are 2307 and 577 respectively.

- **Wider Face Dataset Annotation Format**
  Filename - tennis.png
  No. of faces in the image – 1
  Bounding Box Annotation – 616,116,214,306,0,0,0,0,0,0
  format is as [xmin, ymin, width, height, blur, expression, illumination, invalid, occlusion, pose]

2.1 Link to the DataSet

Link to Wider Face Dataset

2.2 Link to Python Files

Implementation of Faster-RCNN and RetinaNet Model.
Implementation of YOLOv8 Model.

3 DataPreprocessing

We are using two libraries Detectron2 and Ultralytics for run different models. Datapreprocessing techniques to import data in for both of them are different.

3.1 Data Preprocessing for Detectron2

- Detectron2 is a platform for object detection and segmentation created by Facebook AI Research team (FAIR). It implements state of the art architectures such as Faster R-CNN, RetinaNet, Mask R-CNN All the pretrained models used in Detectron2 are trained on the COCO dataset (Common Objects in Context) – a large scale image dataset.

- **Steps to solve the Face Detection Problem using Detectron2**
  - Install Dependencies – Detectron2 Library and prerequisites required for it
  - Loading and preprocessing the wider face dataset
  - Creating annotations as per Detectron2
  - Register the Dataset in Detectron2
  - Fine Tuning the model
  - Evaluating model performance

- **Format for Detectron2**

- **annotations**:{bbox: [xmin, ymin, xmax, ymax], category_id: category_id of image}
- **file_name**: name of image.jpg
- **height**: height of entire image
- **image_id**: number
- **width**: width of image

3.2 Data Preprocessing for Ultralytics

- Ultralytics YOLOv8 is a state-of-the-art object detection algorithm that combines YOLO (You Only Look Once) architecture with the power of Ultralytics deep learning framework. Excels in real time object detection tasks, offering high accuracy and speed for various applications
- **Steps to implement YOLO**
  - Install Dependencies
  - Update the architecture file based on the dataset
  - Load and Preprocess Data
  - Arrange data in required format
  - Train the model
- **Dataset structure and Annotations Format for YOLO8**
  - Changing the number of filters and classes to 1, as we have only 1 class i.e.face, in the YOLO cfg file. Creating 2 folders one for storing the images and other for storing the labels. For each image file create a separate labels file which contains the bounding box coordinates and targets for the respective image file.
  - Bounding box coordinates should be in x-centre, y-centre, height and width format. Box coordinates must be normalized in xywh format (from 0 to 1) for that we need to divide the x-centre and box width with image width and y-centre and box height with image height.
  - Finally, we need to create 2 separate folders for train and validation.
  - Also, create a DataFile which contains the path for train and validation sets and the names file which contains the number of classes and the total number of classes.

4 Model Development

We have built both categories of model:
Regional Based Convolutional Neural Networks: Faster RCNN
Single Stage Networks: RetinaNet, YOLOv8

4.1 Models developed in Detectron2

- Pre trained models RetinaNet, Faster RCNN trained on COCO dataset are available in Detectron2 library. The trained weights are found by training layers of all the pretrained models from Scratch, on custom wider face dataset, as the COCO dataset does not contains the class face.The threshold value is set to 0.8.
- Setting the hyperparameters before the model training:
  images_per_batch = 2, learning_rate = 0.001,
  no. of iterations = 1000, no. of classes = 1 (as only 1 class i.e. face)

4.1.1 Faster R-CNN

- Instead of using Selective Search algorithm as in R-CNN and Fast R-CNN, convolutional blocks are used for generating region proposals using Region Proposal Network(RPN). The features extracted from feature maps are used to create 9 different anchor boxes for each element.
- Objective of the RPN network is to determine whether there is an object or not irrespective of class. RPN has its convolution layers for classification and regression

4.1.2 RetinaNet

- Although single stage networks are faster than 2 stage networks, two stage networks give higher performance than single stage networks. It is because of class imbalance between the background class and object class. To address this problem, a new focal loss is introduced.
- High penalty is assigned for misclassifying the object class using hyperparameters α and γ.
  α is used to assign weightage to the background class and object class, while
  γ is used to differentiate between objects based on ease of classification.
- Uses ResNet as the backbone architecture. Uses Feature pyramid network for multiscale prediction. Also used the anchor boxes concept and used 3 different aspect ratios for these anchor boxes.

4.2 Models developed in Ultralytics

- Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility.
- We trained two variations small and nano of yolo v8 on our wider face data set using pretrained weights on the large COCO dataset. We also fine-tune hyperparameters like learning rate, batch size, and epochs for optimal performance.
- We observed more mAP value for yolov8s model but the inference time less in realtime compared to yolo v8n model.

4.2.1 YOLOv8

- Ultralytics YOLOv8 not only excels in performance but also features a well-designed architecture. This architecture is an evolution of the YOLO family, known for its efficiency in processing images in a single pass, hence the name "You Only Look Once."
- One of the key aspects of YOLOv8's architecture is the use of deeper and more complex convolutional neural networks (CNNs). These CNNs are adept at extracting and learning features from images, making the model more effective in recognizing a wide range of objects.
- Additionally, YOLOv8 employs various techniques to optimize its performance, such as batch normalization and skip connections, which help in faster convergence and improved accuracy.
- Another notable feature is its scalability and flexibility in handling different sizes and types of objects. YOLOv8 can adjust its receptive field dynamically, enabling it to detect small objects with the same efficiency as larger ones. This makes it particularly useful in scenarios where object sizes can vary greatly, such as in aerial imagery or crowded urban scenes.

5 Model Comparison

Evaluating the model with the trained weights stored in model_final.pth file.
Using the COCO evaluator for evaluating, the models are compared based on inference time and mean Average precision for various IOU values on test data.

5.1 Evaluation Metrics

IOU (Intersection Over Union) : The overlap is considered between the predicted bounding box and the actual bounding box. The range of IOU is between 0 to 1.

$$IOU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

Threshold: A numerical value between 0 to 1.
Object is not present in the predicted bounding box and classified as
- True Positive prediction when $IOU >$ Threshold
- False Positive prediction when $IOU <$ Threshold

Precision:

$$Precision = \frac{TP}{TP + FP}$$

Average Precision:

$$Average\ Precision = \frac{1}{N} \sum Precision$$

Mean Average Precision (mAP):

$$mAP = \frac{1}{K} \sum Average\ Precision$$

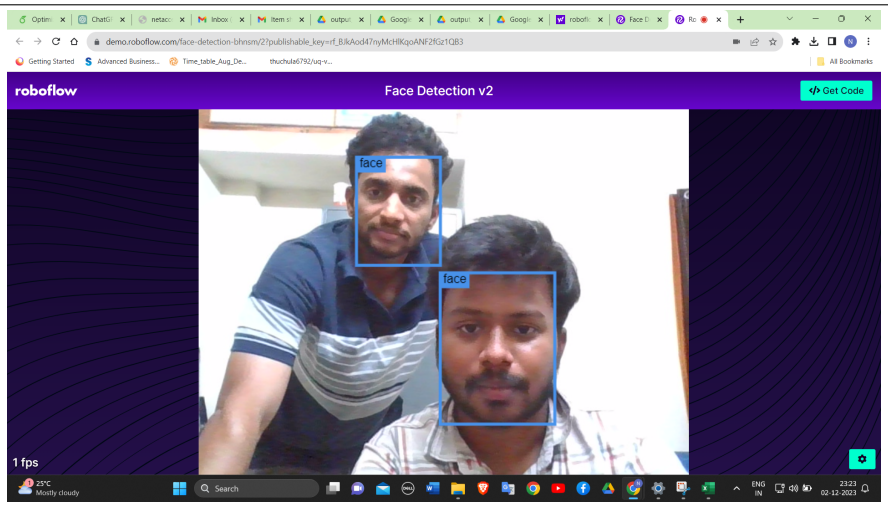
Gives the mean of the average precision across all the classes (K number of classes).

5.2 Inference Time

- Inference time, a critical evaluation metric for object detection models, measures the speed at which a model processes data and makes predictions. Faster inference times are crucial for real-time applications. Minimizing inference time while maintaining accuracy is a key goal in optimizing these models for practical use.
- The general process involves loading the model, feeding it input data, and then measuring the time it takes to obtain the output predictions.

6 Model Deployment in Real Time

Figure 2 Real Time Face Detection



Roboflow is a platform for computer vision tasks like object detection. Steps to use your trained YOLOv8 models for real-time face detection in Roboflow:

- **Model Export**: Export your trained YOLOv8 and YOLOv8s models in formats compatible with Roboflow, like Darknet's YOLO format or ONNX.
- **Roboflow Deploy**: Upload your exported models to Roboflow's deployment service, Roboflow Deploy. This makes your models accessible through API endpoints.

Roboflow simplifies the deployment process, making it easier to integrate your models for real-time face detection into various applications and services

Models	AP50 (%)	InferenceTime (sec/iteration)
Faster R-CNN	62.65	0.28
RetinaNet	67.65	0.27
YOLOv8n	80.9	0.0035
YOLOv8s	82.95	0.0040

Table 1: Results Table

7 Conclusion

- From Results Table, it can be seen that,In the context of wider face detection, YOLOv8 typically has a greater inference speed (lower inference time) compared to Faster R-CNN and RetinaNet. This is because YOLOv8 is optimized for real-time object detection and processes images more efficiently in a single pass through the network
- Mean Average Precision at 50% Intersection over Union (mAP50) for YOLOv8 is higher than Faster R-CNN and RetinaNet on wider face dataset, it suggests that YOLOv8 is performing exceptionally well in terms of accuracy for your particular dataset and task.
- The reasons for YOLOv8 achieving a higher mAP50 could be:
  Training Data: The quality and quantity of training data can significantly impact model performance. If your YOLOv8 model has been trained on a well-annotated and diverse dataset of wider faces, it may have learned to detect faces effectively.
  Hyperparameter Tuning: Proper hyperparameter tuning, including learning rates, batch sizes, and model architecture modifications, can enhance a model's performance.

8 References

1. WIDER FACE: A Face Detection Benchmark
2. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
3. Focal Loss for Dense Object Detection
4. You Only Look Once: Unified, Real-Time Object Detection