FACE DETECTION LIBRARY FOR BLURRING FOR PRIVACY CONTROL USING YOLO V7

Mark Christian Avila

John Remmon Castor

Neil Mathew Lacsamana

James Garcia

Harvey Alonday

A Project
Presented to the Faculty of the
Computer Studies Department
College of Science
Technological University of the Philippines
Ayala Boulevard, Manila

INTRODUCTION

Face detection library system would tackle the need for a face detection library. The system would use object tracking to detect and track the movement of the faces of a person in a video. The proposed system would also incorporate an interactive user interface for casual users. The library would use python, pytorch and a cloud based Machine Learning platform to build and train the model to accurately detect the faces. The project aims to develop a system library that incorporates privacy andblurring controls, such as specific and exclusion detection. The system strives to provide ample data and resources to assist users in efficiently and effectively blurring faces when necessary. The project will be able to blur images, videos, and live video feeds withfully configurable settings to the model and detections. The system is not tailored for low-end hardware or older computers. The primary audience for this system is developers or technical users, at least at first, who are interested in face detection and its applications. The study will develop a library that respects the principles of privacy and consent. Face detection is a technology that identifies human faces in digital images. The project will be able to take in input from local, online hosted, and even HTTP or RTSP URLs. The library will incorporate a user-friendly interface with video input for showcasing the library?s capabilities. It will be developed using:. The haar cascade technique uses features such as edges, lines and rectangles to detect faces in images (Siddhi, 2023) Test and improve the functional suitability, performance efficiency and reliability.

METHOD

The library was developed in Python along with other necessary dependencies. The core methods rely on face detection and blurring, albeit with variations in how they handle the detection output. We set out a goal to collect a minimum of 10,000 images of different people's faces, which we accomplished thanks to open-source datasets like WIDER FACE. The central processing stage initiates with the retrieval of a frame from the video source, followed by its submission to the face detection model. The results of these tests are discussed in the next chapter. The library uses a face detection model to identify people's faces. The library is called Occultus, which is latin for ?hidden? or ?secret? It accepts a video source as its input and processes the video to obscure faces. It then generates an output in the form of a video with blurred faces. A crucial aspect of the model's output is the extraction of bounding box coordinates. These coordinates serve as the primary data for implementing blurring. It is determined as the ratio of true positives to the total number of false positives. Face detection is a standard part of computer vision tasks. It uses the ?SORT? (Simple Online Realtime Tracking) algorithm for object tracking. The face detection process is the first process, which encompasses the pre-trained model used in detecting faces. The video-handling process effectively functions as a controller for overseeing the face detection and blurring procedures. The system then exports the video, complete with blurred faces, for the user to use. It also displays the processed video, with detected and blurred elements, back to the user or displays it on the stream if the user has opted for a live feed option. The YOLO dataset is a vital resource for maintaining high performance and reliability. It also provides clear instructions for managing and troubleshooting the system. The system initiates the post-processing steps, such as converting the original video into a video re-integrating the original audio, and transforming the output into the specified format. The minimum and recommended system specification requirements for running the occultus library are outlined in the table below. The device?s specifications that will be used during executing the testing procedures are also outlined. We present a comprehensive outline of the approach, methods, and techniques that will be employed to investigate and address

the research questions and objectives that were laid out in the previous chapter. The API structure of the library is outlined as follows: The Occultus library (or Python module) will embrace anobject-oriented programming (OOP) structure, where functionalities are encapsulated within objects containing methods and attributes. The ?detect_video? and ?Detect_device? methods are designed for the direct andcomplicated detection of their respective input sources. The Python library is used for face detection in the Occultus package. The ONNX model is exported to a PyTorch-based model. It is then subjected to the preprocessing process, which manages the video frames, and original original. The postprocessing process manages the frames, original original, and bounding boxes and censoring to the final frame. The system architecture is relatively straightforward, primarily. the overall system architecture for our overall system is relatively. straightforward. We plan to have the 'detect_video' method accept a video in a compatible format, identify faces, apply.blurring, and then save the modified video. The system is fundamentally structured as astraightforward input-process-output model. It is designed to handle multiple face-detections and censorings simultaneously. The library is designed with Python in mind. It was designed to be consistent and efficient with software development. It has been designed to work with both live and recorded video. It can be used to test the library's performance and reliability. It will also be used for training and post-processing of video. The system will be available in a variety of formats. The main detection methods proposed include ?detect?, ?Detect_image? and ? Detect_video? This method serves as the core detection mechanism, with all other methods leveraging it for various purposes. We partitioned 40% of the data for training, 10% for validation, and 50% for testing. The standard version utilizes only the CPU for executing inference, while the latter utilizes the GPU for inference. Blurring options include an inference mode that blurs all detected faces, an exclude mode thatBlurs all faces except for a selected one, and a specific mode thatblurs only a particular face. -processing while harnessing the capabilities of the library. Users can simply import the library, granting them access to all the essential methods, functions, and attributes they require. The same object utils refers to the same object directory that contains utility

functions or modules that will provide auxiliaryfunctionality used throughout the library (Python Documentation, n.d.) We opted for the Python package directory structure, with a primary main file with directories for subdirectories. We used object tracking algorithms to detect and manage Face IDs associated with each. The most common approach in evaluating pre-trained object.detection models is with Mean Average Precision (mAP) Given the large.dataset size, manual correction was not feasible. We tackled the first issue by creating custom Python scripts 0.0-0.29 paste_in. The image copy paste (probability) 0.05-0, 0.30 loss_ota, 1. Use ComputeLossOTA, 1, or YOLOv7-tiny hyperparameters for training.

RESULTS

Run detection and. blur on a video using the following methods. Set censor type to "blur" using the 'change_censor' method of the face detection library. Connect to the web via the 'Occultus Websocket' server using the given address. Use the'multipart' function to connect to different video sources. The'multibit' function is used to add and remove faces from a video. The face detectionlibrary's accuracy is highly regarded by the users, with 62.5% of respondents rating it as "Highly Acceptable" The survey results for the functional completeness indicates that the facedetection library is performing exceptionally well in meeting its respective objectives. There were no ratings for "Fairly Acceptable" or "Not Acceptable", which indicates that all users are satisfied with the accuracy of the face detection. The AP for faces in the hard category is 77%, which says that the model struggles in detecting faces under difficult conditions, but it still maintains a reasonablelevel of accuracy. The model is probably due to the model, YOLOv7-tiny, which is a smaller and lighter version of the YOLov7 model. We present the results and discuss the findings of our softwaresystem, as well the components of our system. The software system is based on a model of the face detection library. The system is designed to provide reliable and relevant functionality for face censoring tasks. The results show that users generally view the library's performance positively, but with some room for improvement. We also show that the library effectively caters to the specific user requirements, providing reliable functionality. We conclude by discussing the results of the software system and components of the system. Our model, which we named ?kamukuk?, scored an impressive 92.5% Average Precision (AP) in blur conditions. Our model is effective across a range of difficulty levels, with particularly high accuracy in easy and medium conditions. The CMS-RCNN?s test size is not specified. Our "Face Detection Library with Privacy Controls" overall shows betterperformance in average precision (AP), especially in the easy and Medium categories. A significant majority of the survey votes, 75% of respondents rated it as "Highly Acceptable" Our system is structured in an object oriented manner where most of its object orientedfunctionality comes from the methods or functions of an instance of the Class Useful if

further processing is needed when detecting videos. Our face detection library with privacy controls demonstrates strongperformance across all evaluated criteria, with particular strengths in Functional strengths. All areas of Functional strengths received ?Highly Acceptable? ratings, indicating that our.library is highly capable in its methods and functions, as well as reliable in error. handling. The face detection model was trained for approximately 30 hours using Saturn Cloud's Tesla T4 GPU. The training process is so complex that only users with advanced technical skills will be able to accomplish it. The total number of frames of the video frame was calculated using the mobiledevice camera and the mobilecamera's blurred feed. The validation set represented by the 'val box' and 'val objectness' metrics in the validation set shows consistent improvement in the model's performance. A substantial majority, 70.83%^of respondents, rated the storage and memory usage as "Highly Acceptable", while the remaining 29.17% rated it as "Very Acceptable" Cccultus has developed a face detection and blurring library. It is capable of giving robust methods for both face.detection and privacy control. It can be used to integrate with popular online streaming platforms like Youtube or Facebook. The results of these metrics after training are displayed below. The library is confident in its predictions with absolute precision. It identifies faces accurately with precision with an absolute precision of 89.7% when the model is configured to use the GPU, taking roughly an average of 12 seconds when it is used. The WIDER Face dataset is a popular benchmark dataset for face detection models in the space of computer vision. ?kamukha? achieves the highest Average Precision at IoU threshold 0.5 (mAP@0.5) It outperforms LFFD (89.6%%) and 86.5% of CMS-RCNN (90% and 87% respectively) Only one respondent (4.17%) rated it as "Fairly Acceptable", and there are no ratings of "Not Fully Acceptable"

DISCUSSION

The Python-based face detection and blurring library, named ?Occultus?, has proven to be highly effective and efficient. Users have rated the library highly, with the majority rating it as "Highly Acceptable" and no respondents rating it lower. The following recommendations are made for the enhancement of the developed python system. The recommendations include further optimization, and fine-tuning of the object detection model to significantly increase accuracy and precision in detecting faces. A method or function for detecting and. blurring faces in internal or external video sources and streaming the output as a JPG Image.URI through. websocket technology. Face detection package for python is free from notable errors or bugs. Users find the library's speed satisfactory, with most rating it as "Very Very Acceptable" and a significant portion rating it "Highly Acceptable." The library encountersunexpected errors or malfunctions very infrequently, with a majority of users indicating it "Never" happens and the rest stating it "Rarely" happens. The "kamukha" model, based on the YOLOv7-tiny architecture, has proven to be highly accurate in easy and medium conditions.