Student Counter Using IoT and Face Detection Algorithm

Tushar Dubey
UG Scholar
Computer Science & Engineering
Shri Shankaracharya Institute of Professional Management And Technology
tushar.dubey@ssipmt.com

Dr. J. P. Patra
Professor
Computer Science & Engineering
Shri Shankaracharya Institute of Professional Management And Technology
jp.patra@ssipmt.com

Kanha Agrawal
UG Scholar
Computer Science & Engineering
Shri Shankaracharya Institute of Professional Management And Technology
kanha.agrawal@ssipmt.com

Vishal Rathi
UG Scholar
Computer Science & Engineering
Shri Shankaracharya Institute of Professional Management And Technology
vishal.rathi@ssipmt.com

***Abstract*—There is always a need to count how many people are in a room (conference room or classroom) and it is not an easy task for a human to count them without using any technology. If a person tries to count, there is a very high possibility of inaccurate data, moreover this will take a lot of time when it comes to counting a large number of people, and that too in different rooms. Hence to overcome this hectic work, we have a project that works on the principles of a tiny face detection algorithm. This project explores three phases: Capturing, Processing, and Display. In the first phase, we have an HD camera which is used to capture the image of the room via a microcontroller. This microcontroller can be either Raspberry Pi or NVIDIA Jetson Nano. The second phase consists of applying the pre-trained deep network to the image to detect tiny faces, achieving the desired count of persons, and storing that in the database. After storing the data in the database we come to the third phase of our project which is to display the number of people in a room. The results can be viewed via a mobile application or on the website along with data analysis of previous results. You can see the number of people seating in that room in previous days, anytime you want.**

Keywords*—Tiny face detection algorithm, Microcontroller, Raspberry Pi, NVIDIA Jetson Nano, Pre-trained deep network, Data analysis.

I. Introduction

Student Counter helps to count the number of students sitting at that particular instance in the classes. The system will generate the count of students and display the result at the required place.Before this, there wasn't any technology that counted the students in a classroom, hence the process was done manually by any person. He/she has to visit every classroom and either count the students sitting in that room or count the students through the attendance sheet. It was a very time-consuming process and required human resources which may lead to a high risk of error and sometimes disturb the running classes.Real-Time Tiny Face Detection is a useful technique for managing the daily attendance of many pupils. Numerous methods and algorithms have been created to enhance face recognition performance, but our suggested model uses Residual Networks (ResNet)[4] to identify the positive and negative features of the face. All of these methods and algorithms are implemented in Python and use the TensorFlow and OpenCV libraries. We make use of the pyQt5 GUI interface for user interface needs. The goal of the research is to identify small objects in images, primarily faces, utilising scale-specific detectors and features defined over a single (deep) feature hierarchy: scale invariance, picture resolution, and contextual reasoning. The method, which mimics human vision, is based on "foveal" descriptors for blurring the peripheral image in order to encode and send just the right amount of background information. Given that the subject is still up for debate, we wish to test this method in a variety of settings. We will thus focus on a real-world application, which is counting individuals at a public demonstration, after introducing the approach and the impact of its parameters. The final section will concentrate on potential future developments for this work. Our code (using Python and TensorFlow) is available at https://github.com/vishal6789/Student-Counter. The necessity to safeguard gadgets that store both our private and business information is crucial as long as humans are making technical progress. By using ID cards, passwords, passphrases, and puzzles, some conventional approaches succeeded in accomplishing this. The use of human biometrics, such as the face, voice, and fingerprints, is now widely accepted in contemporary security verification programmes, however, as a result of the quick advancements in deep learning and high-performance computers. These human biometrics are used widely because they are distinctive and challenging to duplicate. Similar to this, face recognition software offers a quick yet effective framework for identifying a person. Face detection software can be found in commonplace gadgets like laptops and mobile phones, as well as in physical security equipment installed in workplaces. They have never been more successful at correctly detecting faces. But in order to complete this stage, pre-trained models used for human face detection, like FaceNet, ResNets, SesNets, and their variants, are trained on human faces..It is very difficult - if not impossible - for a human user, to recognize very small faces. So, we must decide on the most effective context encoding method. A fixed-size (291px) receptive field s used in the paper for context modeling. The "hypercolumn" features, which are useful "foveal" descriptors and are retrieved from many layers of a deep model, are then defined as templates over these features. Through the use of this approach, it is possible to record both coarse low-resolution stimuli and high-resolution detail across vast receptive fields. Then, it defines templates over "hypercolumn" features, which are useful "foveal" descriptors and are extracted from several layers of a deep model (vector of activations of all units for each pixel). With this method, it is possible to record both fine low-resolution detail and broad high-resolution cues across sizable receptive fields. We need to identify each face and then match them in the following frame to count people just once. Of course, the initial step was to find faces in each frame. To do this, we used the Tiny Faces algorithm to identify every face, and as a result, for each frame, we received a list of anticipated faces that had been spotted. The next step is to match them between frames. Of course, the initial step was to find faces in each frame. To do this, we used the Tiny Faces algorithm to identify every face, and as a result, for each frame, we received a list of anticipated faces that had been spotted. The next step is to match them between frames. To make the matching process simpler, we developed a face embedding for each face by wrapping each image such that the faces always face the same way. A pre-trained face landmarks estimation technique and affine transformations are used to align the face. We used a pre-trained CNN for face embedding to get over the fact that we only have a few images of each face and to have a faster (in terms of time and memory) algorithm comparison (generating 128 basic measurements for each face).

II. Literature Review

In tiny face detection, the process flow is initiated by capturing images of classes in a frequent interval through a camera by using an IoT device (Raspberry Pi 4 / Nvidia Jetson Nano kit). Then the captured image will be sent to the server for computing the count face of a face in the image. This will be done through an algorithm that processes the image captured and identifies the number of faces in the image by analyzing the learned or pre-trained model. Then the result of that Image will be sent to the required places where it will display by GUI on a Screen. There are times when an authority wants to know how many students are present in a classroom so that they can make any changes or pass any decision. Now, it's a hectic task to go to every class physically and count the number of students every time. It will not only disturb the running classes but also will be a not-so-efficient way. So we wanted to build a tool that can count the number of students in a classroom at any particular time and display the count to the authority seating at that cabin. The motivation behind this project is to simplify the means of counting students in each classroom which is taken during lectures and by reducing human power and time. With the tiny face detection system in place, it will be easy to tell the count of students which are present in the classrooms. One of the most productive fields of research in machine learning is computer vision[5]. Within the last 20 years, significant advancements in this area have been made. A model to train deeper neural networks to the required depth was put forth by Hinton et al. in 2006. Then, in 2012, a breakthrough was made possible by the creation of an 8-layer neural network that performed better than all other algorithms at identifying photos in ImageNet. They used convolutional layers to correct their architecture's linear activation and dropout. Their findings paved the door for stronger computer vision algorithms to be created. Computer vision includes face detection as a subset. Face detection has great detection accuracy thanks to the employment of algorithms similar to those used in computer vision. This is significant since it allows for the identification and authentication of people. In this section, we concentrate on the deep learning methods that helped advance the field of face detection research. FaceNet's accuracy on the LFW database in 2015 was 99.63% utilizing the GoogleNet-24 framework. On the LFW database, Parkhi et alwork .'s on the VGGFace led to an accuracy of 98.95%. The VGGNet16 architecture was employed in this instance. After Microsoft Research successfully used this architecture for picture classification in 2015, most typical strategies now rely on ResNet and its variations. Many studies have been undertaken to detect faces with occlusion, and Cao et al. used ResNet-50 architecture in their work, which is analogous to our own. The restoration technique and the discard occlusion-based strategy are the two main strategies employed in this context. Based on the photos used for training, the restoration approach attempts to restore the portions of the images that were obscured. In order to restore the occluded portions of the face, Bachi et al. deploy a 3D face detection method. Iterative Closest Point Algorithm is used by the system to record a 3D input of a person's face. Next, occluded areas of the face are found, and then a restoration method employing Principal Component Algorithm is used (PCA). Similar to this, Drira et al. employ a 3D-based statistical method to identify and estimate the portion of the face that is obscured. The PCA method is employed in the recovery of occluded facial areas.

III. Methodology

Our system will capture the image of the classroom using a webcam and then send that to the server and after computing the image, send the desired result to the respected authority. Typically, this process can be divided into three stages.

A. Image Capturing:

Images of students sitting in a classroom are captured using a webcam which will be connected by an IoT device which can be either Raspberry Pi 4 or NVidia Jetson Nano kit..

Figure 1: Image of Students

Multiple images will be captured by the devices in desired intervals (suppose 2 images in each period of a class). Then the image will be sent to the server for further computation

B. Server

A local server is created using Node.Js. The time when it receives an image, it will run the python script that contains the main code of tiny face detection. Once the python file starts executing, it will detect the number of faces present in the image which was received. For the first time, it will create a JSON file containing the count of the students present in each classroom. The JSON file will be updated when the new image will be received

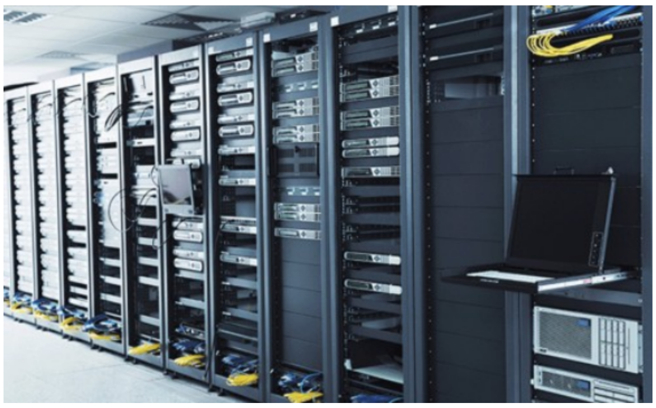
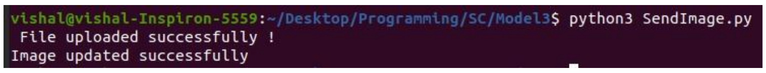


Figure 2 Server Room



```
vishal@vishal-Inspiron-5559:~/Desktop/Programming/SC/Model3$ python3 SendImage.py
 File uploaded successfully !
Image updated successfully
```

Figure 3: Uploading Images To Server

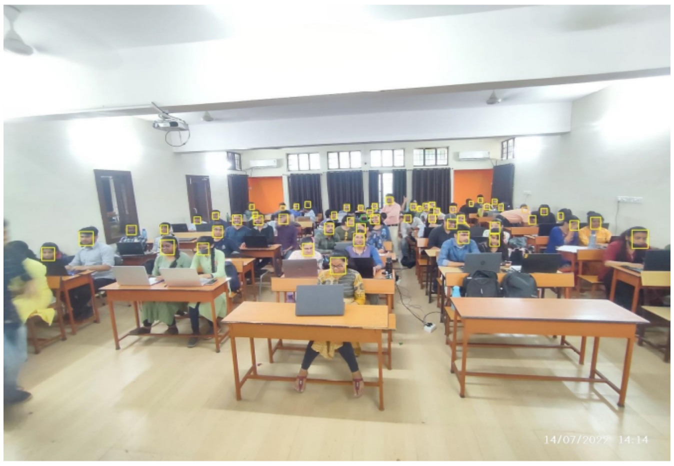


Figure 4: Detecting Faces

C. Display Unit

Once the data is updated in the JSON file, our display unit (Jetson Nano Nvidia kit)[2][3] will request the server for the JSON file and after receiving the file, it will read the file and shows the desired output in the Screen using GUI created in python using pyQt5 library.

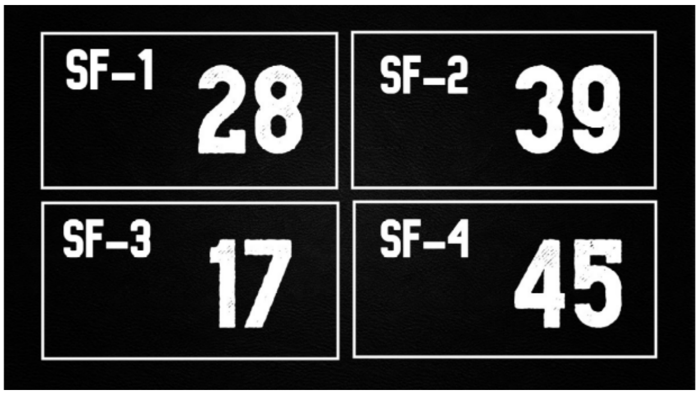


Figure 5: User Interface Display

D. Software Requirements

- *Python3*
- Numpy
- Pandas
- Tensorflow
- matplotlib

E. Hardware Requirements

- 64-bit CPU (Intel /AMD architecture)
- 2 GHz above processor
- 8 GB Ram recommended
- Nvidia Jetson Nano Kit
- Raspberry Pi 4 Kit
- High-Quality Webcam
- Display Monitor

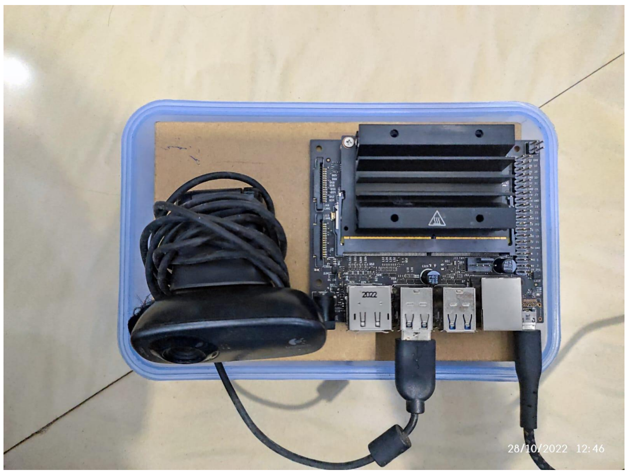


Figure 6: Nvidia Jetson Nano Kit With High-Quality Webcam

F. SDLC MODEL (Rapid Application Development)

The Software Development Life Cycle (SDLC) is a conceptual model for project management that describes the stages of developing an information system, from the initial phase of a feasibility study to the continuing maintenance of the finished application. The phases of the SDLC give a shared knowledge of the software development process. Rapid Application Development (RAD), a method of software development, prioritizes quick feedback and rapid prototyping over protracted development and testing cycles. Developers may swiftly iterate and update software without having to start from scratch thanks to fast application development. As a result, the final product is more quality-focused and in line with the needs of the end user. In our project as per the RAD model, we have to identify and analyze the system functional requirements, and based on requirements, the design of the system functionality and the development method is finished with success. The RAD approach makes a distinction between rapid application development and traditional software development methods right on. It merely requests a broad request; it doesn't demand that you meet with end users and compile a detailed list of requirements. Due to the broad nature of the criteria, you can segment specific requirements at different phases of the development cycle.

IV. Working Principle

The student counter works on the very simple principle of Input and output. As an input, it takes the image of a classroom or seminar hall and returns the student count as an output.For input (image) it uses a web camera for capturing images, which is connected to raspberry pi for controlling its operation. The captured image is then sent to the centralized server system to which the raspberry module is connected using the local network. On the server, the image gets processed using the tiny face algorithm, which first removes the unwanted facial features to make the face count processing easy and further reduces the number of pixels for making the comparison easy pixel by pixel. After collecting the number of face counts in an image, the algorithm returns the count to the server, which is then displayed using the Display Unit.

V. Conclusion

This system aims to ease the task of counting the students sitting in each classroom. The proposed system will be able to detect the number of students sitting in a classroom. For Tiny Face Detection, a pre-trained Residual Network is used. It will capture faces via webcam. After capturing, it will send the image for computation (i.e. detecting the number of students sitting in a classroom or lab) and then send the count to authorities and display them via GUI. The Student Counter System helps in increasing the accuracy and speed ultimately achieving the real-time calculations of students present in a classroom. It not only reduces the chances of errors that can be made by humans but also reduces human interaction and at the same time increases its efficiency. It reduces the time taken to complete the task too. Face detection is an emerging technology that can provide many benefits. Face detection can save resources and time, and even generate new income streams, for companies that implement it right.

References

1. Raspberry Pi 4 For Beginners And Intermediates: A Comprehensive Guide for Beginner and Intermediates to Master the New Raspberry Pi 4 and Set up Innovative Projects by Craig Berg https://singleboardbytes.com/647/install-opencv-raspberry-pi-4.htm
2. NVIDIA Jetson nano official website: https://forums.developer.nvidia.com/t/official-tensorflow-for-jetson-nano
3. Raspberry pi official website: https://www.raspberrypi.org/
4. Dependencies on Jetson Nano https://qengineering.eu/install-tensorflow-2.4.0-on-jetson-nano.html
5. For Research paper reference: https://scholar.google.com/