Attendance Monitoring System for Academic Institutions using Nvidia Jetson Nano

Laksh C, Gururaja H.S

[1]Student, Dept. of Electrical and Electronics Engineering, B.M.S College of Engineering, Bengaluru, India
[2]Assistant Professor, Dept. of Information Science and Engineering, B.M.S College of Engineering, Bengaluru, India

Abstract - The task of monitoring attendance in a classroom setting can be a tedious and time consuming one. It also gives rise to chances of proxy attendance and other mistakes made while marking a person's presence. This project aims to tackle this problem by recognizing and marking people's attendance either through a live video stream (surveillance camera) or by uploading pictures of the people present. The computation for this process is satisfied when using a Jetson Nano.

Index Terms - Nano, facial recognition, dataset, encoding, Graphical Processing Unit (GPU)

I. INTRODUCTION

The methods used for monitoring attendance generally revolve around keeping a person in charge of marking an individual's presence either through roll call or by circulating sheet(s) of paper. To cut short the time spent and also reduce the chances of proxies, the idea of implementing facial recognition for this task came into light. The in-charge person would only require a photo(s) of the students present and feed it as input to the system wherein the students who constitute the class's strength or whose names are on the attendance list would be identified and marked while the proxies would not be recognized.

II. LITERATURE SURVEY

The field of face recognition primarily stems from the foundations of deep learning. The initial stages of face recognition and object detection in general, was primarily spent in finding out the most (or close to) efficient methods to identify the boundaries of a face present within a given image. The approaches included window sliding methods with the major drawback being that one image would have to sampled with windows of many different dimensions and find out the most suitable window which encapsulates the object. While this was a great start, this method was certainly not efficient at processing large chunks of images or capturing multiple objects of the same type in an image consistently. Since then, efficient models such as Single Shot Detector (SSD), have been developed with uses grid cells and convolution neural networks to make this task easier as explained in [1]

Many applications also use this model or other variants of it depending on their needs and incoming data. For instance, the CCTV cameras monitoring the streets could be hooked onto a model which detects license plate numbers of any vehicle appearing to break the rules. For this, the camera would have to be placed at a certain angle to cover both or all the lanes present at the location [2]. The possibility of the model being deployed using a webcam fixed at such a calculated position in a classroom to recognize faces could be explored in a similar fashion. Having a good model architecture is not enough if it has not trained over a certain number of epochs and hence it is necessary to adjust the number of epochs and observe differences in the results. This is made evident in [3].

Several research works have also presented methods on comprehending body language as seen in [5] wherein an image of the person is given and the model extracts the crucial features from the image, which in this case would be the hand positions, and pass them for further processing thereby saving computational resources.

Even when the models tick the boxes of being efficient and accurate, they can often get fooled by simple actions. Such actions include but are not limited to, wearing a hat, yawning wearing large pairs of shades, etc [6]. Thus, it is essential that the students in the classroom remain still without any facial accessory on their face when the photo/stream is being captured.

III. COMPARISON WITH OTHER PAPERS

(1) Barcode Scanning

Barcode scanning is possible by either handing out passes/tokens to people beforehand or by asking them to scan a certain QR code/ Bar code before entering the classroom. While this is an efficient method, it does cause queues to form at the entrance which could be annoying to deal with especially in a classroom environment. Students in most schools aren't allowed to carry mobile phones either and thereby this solution cannot be implemented in a classroom

(2) Fingerprint Recognition

Fingerprints are unique for everyone, and this is also a smart way of marking attendance. However, this solution too faces the same problem of making queues while students enter and wait while their fingerprint gets recognized. Poor quality fingerprint readers can develop smudges on the lens easily and thereby cause further delays in recognizing a fingerprint.

In comparison with both technologies mentioned above, facial recognition is faster as the entire class gets examined at one go and every recognizable face is mentioned and marked. The only effort required from the student's side is to just make sure that they are facing the camera from where the picture is being taken. The teacher/invigilator has to make sure that the photo(s) is/are being taken properly.

IV. THEORETICAL FRAMEWORK

The overview of the steps taken chronologically are as follows:

1. Like with any other Machine Learning related approach, the first step would be to collect data about the students expected to turn up for the class. Data needs to be collected such that the image of the student corresponds to the name of the student mentioned in the filename of the image saved.

2. Next step would be to implement the facial recognition algorithm on every image which would give rise to encodings in the form of a multi-dimensional vector. Since these encodings are in the form of vectors, the distance between a known face's encoding vector and an unseen facial image can be computed.

3. During deployment however, it is necessary to first detect the location of faces (expressed through coordinate system) present in the image received as input by the user. Only on those locations would the facial recognition take place thereby saving computational resources as well as reducing the time taken.

4. Once the locations of the faces present in the image is computed, the faces are encoded and the distance between the corresponding encoding vectors and the encodings of known faces are found and a match is declared only if the distance is within a certain threshold set by the user.

(1) Use Case Scenario I:

Consider the example of a large conference taking place inside a suitable conference hall present in the academic institute. A conference hall would typically require modern ways of monitoring attendance as it is easy for mistakes to happen during manual entry. So here, the person in charge of marking attendance must move around the conference hall (preferably during a short break or towards the end of the conference) and click photos of groups of students/staff sitting at every section of the hall. Another option would be to use image data from the surveillance cameras present in the hall and monitor the presence of a person. This is more complex as the people would then need to look directly at a camera to ensure that the camera recognized them properly.

(2) Use Case Scenario II:

The other common use case would be in classrooms wherein, the crowd is not as big as a conference but attendance for every period during the day needs to be monitored. For this scenario, the teacher needs to take just one or two pictures of the whole class while making sure that the photos are without any distortions such as glare from the windows or blur.

V. ARCHITECTURE, DESIGN AND MODELLING

(1) Hardware Setup:

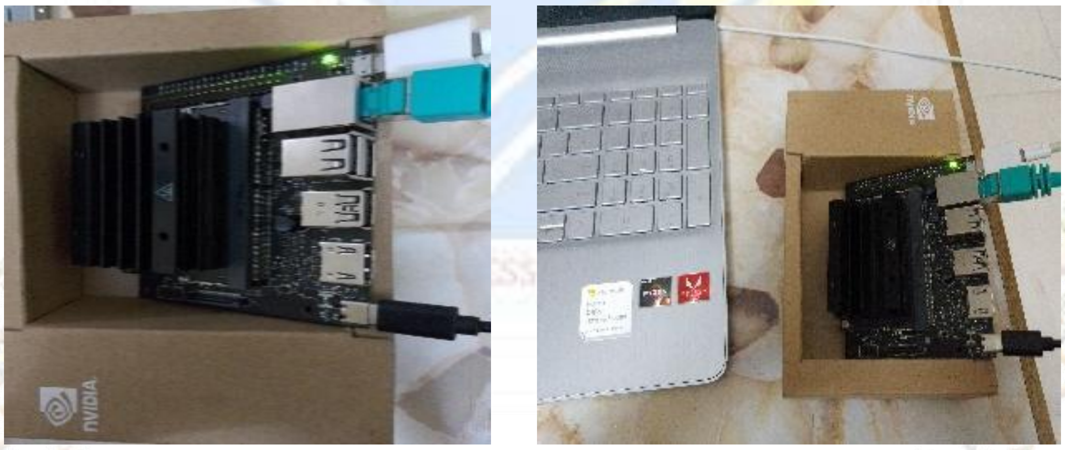


Fig.1 Connections made to the Jetson Nano

The connections made to the Jetson Nano are as shown in Fig 1. On the left, a type C cable is connected to supply power to the module while on the right, the ethernet and micro-USB cables are connected to the router and the PC respectively. As the Jetson Nano is not connected to an external monitor or keyboard separately, it is said to operate in 'headless' mode. Code is written on the PC and is made to run using the module's GPU through the communication given via the micro-USB cable.

(2) About The Model:

The code for the program is written using python. The detection algorithms are taken from the face recognition library from python whose network is based on Resnet-34 but with fewer layers and half the number of filters which was trained over 3 million images to reach 99.38% accuracy in face detection.

The method used is a technique called "deep metric learning." To elaborate, deep metric learning tries to output a real-valued vector instead of outputting a single label for the faces detected within the boundaries of the face located. This quantifies a face. Two of these images are example faces of the same person. The third image is a random face from our dataset and is not the same person as the other two images. From there on, the general idea is that we'll tweak the weights of our neural network so that the 128-d measurements of the two known faces will be closer to each other and farther from the measurements for the random face chosen. Fig.2 shows the data set chosen for the task of identifying 15 people from a class.

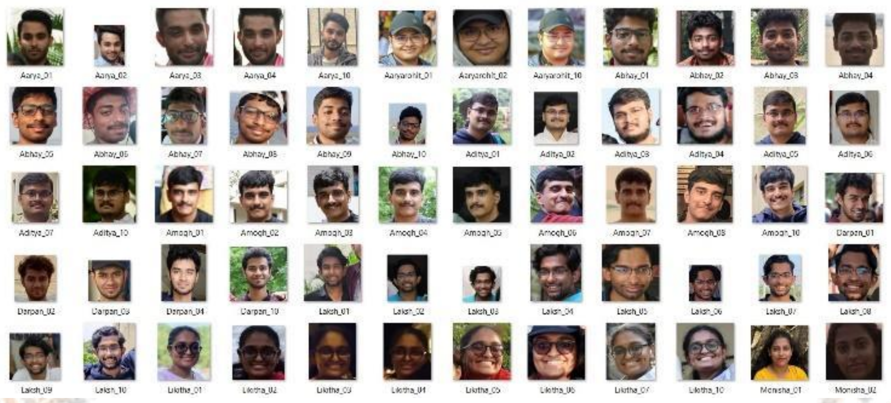


Fig.2 Dataset taken for the project

(3) Code Structure:

Two python files were written for this project. The first one reads through all the images listed under the directory of 'known images' and stores the encodings of each image along with their names in the same chronological order in two separate arrays. This is done using 'cv2' and 'os' libraries where cv2 helps us to perform operations on the images and os gives the ability to read through directories and its files. The 'face recognition' library is then used to detect the face present in the known sample and subsequently run the encoding algorithm to encode the face. The name of the file gets pushed into the array kept for storing names sequentially while the encoding gets pushed to the other array. The two arrays are then pickled in a file from which, the data can be read by other files, thereby saving us the trouble of doing this process again and again.

The second python file is where the recognition of unseen images takes place. The program first gets the data regarding the encodings and the names from the pickled file and stores them in two separate arrays as followed. The code then runs through the directory of unknown images and the face detection library detects the location of every single face present in the image. As and when it detects a face, the library encodes it and compares it to the array of known encodings obtained from the pickled file. A match is decided based on a threshold value that is set by the user and this threshold value indicates the maximum distance between the encoding vectors of the current face and the recognized faces. If the distance falls under this value, the algorithm recognises this as a match.

If the unseen face somehow matches with an image from one person and an image from another, then the person whose images have been matched with the unseen image on multiple occasions would be given preference. That is, out of 5 images of a recognized person A and person B, the unseen image is declared to belong to person A if 4 out of 5 images of his matches in contrast to 1 or 2 matching images out of 5 of person B. This process keeps happening until all the faces have been checked and verified. An output of the faces present is also sent to the user.

VI. EXPERIMENTATIONS AND DATA ANALYSIS

For a model dealing with predictions of multiple subjects, it is important to have a suitable metric for evaluation. For this project, a standard combination of accuracy and precision were chosen to quantify the model's success.

As face recognition is essentially a classification task, we can use the confusion matrix parameters to gauge our model. The people present in each of the test images were manually detected and stored in an array. Subtracting the names in this array from the names in the main dataset, gives us the list of people who are absent. True Positives (TP) were calculated based on the number of people identified by the model and verified using the manually created array. True Negatives (TN) were the people who were not identified by the model and were verified to be absent. False Positives (FP) are the number of people who were identified to be present, either by labelling a wrong face with an absentee's name or by labelling two or more faces with the same name. Lastly, False Negatives (FN) are the people who were present but were not identified by the model. Using these parameters, accuracy and precision can be calculated.

Accuracy was calculated using the formula:

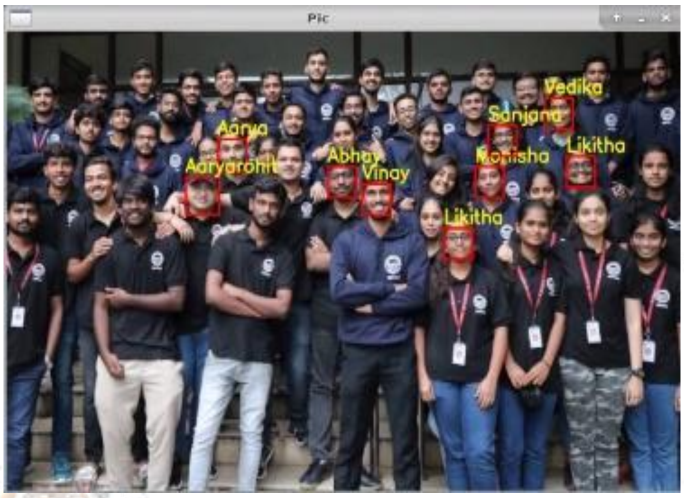
$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \qquad (1)$$

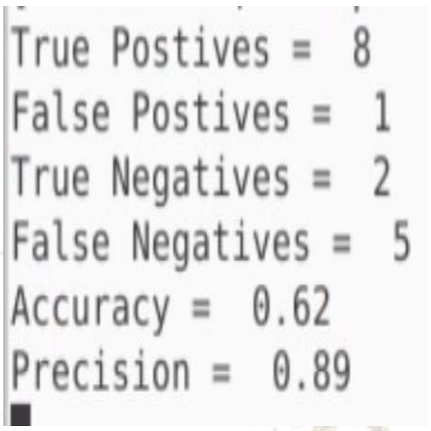
Precision was calculated using the formula:

$$\text{Precision} = TP / (TP + FP) \qquad (2)$$

A total of 6 images were chosen as test images with different resolutions and sizes. The average accuracy obtained using a dataset consisting of original pictures of the students is 0.8534 and average precision is 0.9517. The average accuracy obtained using artificially generated images as the dataset is 0.8072 and the average precision is 0.7591.

Fig 3 shows the result obtained after passing one of the test images through the trained network.





```
True Postives =  8
False Postives =  1
True Negatives =  2
False Negatives =  5
Accuracy =  0.62
Precision =  0.89
```

Fig.3 Results of the Network

The model identified everyone present and was accurate in recognizing each face in Fig 3. The network does tend to find difficulties in distinguishing similar looking people especially when one among the two is not from the known dataset thereby instantiating false positives.

VII. CONCLUSIONS

The results of the model gave us insights about the strengths and weaknesses behind such a model. The dataset played a crucial role in enhancing the model's success and a dataset consisting of original pictures performed significantly better than an artificially created one using data augmentation. Thus, this method proves to be an effective way of attendance monitoring.

VIII. REFERENCES

[1] Saif Ur Rehman, Muhd. Rashid Razzaq and Muhd. Hadi Hussian, "Training of ssd(single shot detector) for facial detection using Nvidia Jetson Nano", unpublished, arXiv:2105.13906v1

[2] Md. Imran Uddin, Md. Shahriar Alamgir, Md. Mahabubur Rahman, Muhammed Shahnewaz Bhuiyan and Md Asif Moral, "AI Traffic Control System Based on Deepstream and IoT Using NVIDIA Jetson Nano", 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), doi: 10.1109/ICREST51555.2021.9331256

[3] Kacper Podbucki and Tomasz Marciniak, "Aspects of autonomous drive control using NVIDIA Jetson Nano microcomputer", 17th Conference on Computer Science and Intelligence Systems pp. 117–120, doi: 10.15439/2022F89

[4] Wuttichai Vijitkunsawat and Peerasak Chantngram, "Comparison of Machine Learning Algorithm's on Self-Driving Car Navigation using Nvidia Jetson Nano", 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)

[5] Faculty of Management Science, Chandrakasem Rajabhat University, Thailand, "The Performance of Thai Sign Language Recognition with 2D Convolutional Neural Network Based on NVIDIA Jetson Nano Developer Kit", TEM Journal. Volume 11, Issue 1, pages 411-419, doi: 10.18421/TEM111-52

[6] Petchara Inthanon and Surasak Mungsing, "Detection of Drowsiness from Facial Images in Real-Time Video Media using Nvidia Jetson Nano", 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)

[7] S. J. Matthews and A. S. Leger, "Energy-Efficient Analysis of Synchrophasor Data using the NVIDIA Jetson Nano," 2020 IEEE High Performance Extreme Computing Conference (HPEC), 2020, pp. 1-7, doi: 10.1109/HPEC43674.2020.9286226.

[8] J. Fontaine, A. Shahid, R. Elsas, A. Seferagic, I. Moerman and E. De Poorter, "Multi-band sub-GHz technology recognition on NVIDIA's Jetson Nano," 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), 2020, pp. 1-7, doi: 10.1109/VTC2020-Fall49728.2020.9348566.

[9] Nvidia, "Getting Started with Jetson Nano 2GB Developer Kit NVIDIADeveloper", htttps://developer.nvidia.com/embedded/ embedded/learn/get-started-jetson-nano-2gb-devkit