

# Enhancing Security: Facial Comparison for Student ID Verification

Tim Nguyen  
UNC Charlotte Graduate  
Charlotte, NC 28223  
hnguye91@charlotte.edu

Naveen Vellaturi  
UNC Charlotte Graduate  
Charlotte, NC 28223  
nvellatu@charlotte.edu

Aaron Zhang  
UNC Charlotte Graduate  
Charlotte, NC 28223  
azhang7@charlotte.edu

Ansh Sawant  
UNC Charlotte Graduate  
Charlotte, NC 28223  
asawant4@charlotte.edu

Gage Markgraf  
UNC Charlotte Undergraduate  
Charlotte, NC 28223  
gmarkgra@charlotte.edu

Ria Banerjee  
UNC Charlotte Graduate  
Charlotte, NC 28223  
rbanerj2@charlotte.edu

## Abstract

*For this semester's project, our team aims to develop an innovative solution for a student verification system by leveraging face detection techniques to scan and compare students' ID photographs (labeled with the student's ID), which are stored in a database, with live camera-captured images of the students themselves. The primary objective is to provide a matching percentage between the person in the live image and the individual on the stored ID image, facilitating efficient and accurate attendance-taking and student verification processes. The result of this project would enable usage in various domains, such as educational institutions, examinations, corporate environments, and access-controlled facilities.*

## 1. Background

Siamese networks in facial recognition offer a potent solution by leveraging pairs of facial images to learn robust representations. Comprising twin neural networks, these architectures are trained to minimize the distance between embeddings of similar facial pairs while maximizing the gap between dissimilar ones. This approach, facilitated by contrastive or triplet loss functions, enables the networks to extract discriminative facial features, making them adept at handling variations in illumination, pose, and expression. Despite their effectiveness with limited labeled data, challenges such as data-set size, and imbalanced distributions, prompts ongoing research to further enhance their performance and generalization in facial similarity recognition tasks across various applications, including authentication systems and surveillance technologies. In our case, we will be leveraging this technology and system to assist in higher education facial recognition.

## 2. Literature Reviews

The literature presents a diverse array of advancements in facial recognition and image matching techniques. Chen et al. showcase the effectiveness of Convolutional Neural Networks (CNN) by achieving a 4% increase in accuracy through training for multiple facial expressions [Chen et al., 2017]. Assran et al.'s Masked Siamese Networks offer an innovative approach to label-efficient learning, employing masking techniques for image denoising without pixel reconstruction, particularly valuable in facial recognition tasks [Assran et al., 2022].

In the context of real-world scenarios, Zhang et al.'s work on Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks addresses challenges such as poses and illuminations, achieving superior accuracy in face detection benchmarks with real-time efficiency [Zhang et al., 2016]. Koch et al.'s exploration of Siamese Networks for one-shot image recognition highlights the adaptability of paired structures and weight sharing between identical neural networks, proving effective in recognizing objects from limited examples [Koch et al., 2015]. Lastly, Melekhov et al.'s Siamese Network Features for Image Matching introduce a novel approach for identifying matching and non-matching image pairs, demonstrating enhanced matching performance in various scenarios [Melekhov et al., 2016]. These findings collectively contribute to the evolving landscape of facial recognition and image matching technologies.

## 3. Proposal

Our proposed topic includes the use of feature matching using object detection with cameras to find a percentage match with student IDs stored in a database. Object detection is the technology that allows us to identify and locate items in a

given scene. The use of this is to find the students, specifically their faces. In practice, the student will first take a picture of their face, and then a picture of their ID. The algorithm will detect and read the ID number on the student's ID card and use it to run a search on the database and find the stored student ID image. The algorithm will then locate the student's face on the picture they have taken as well as the face on the stored ID image. We will use feature matching to compare the features detected within the scene element of their face and the image on the ID. The system will generate a match or percentage score based on how closely the detected face in the image scene matches the reference ID. This score will be used as a measure of accuracy and confidence for a match result.

## 4. Our Data

The dataset employed for implementing the Siamese network in our study for facial recognition and similarity rating comprises a collection of 400 images distributed among 40 individuals, with each person represented by 10 distinct images. Our dataset is partitioned into training and testing subsets, with 37 sets, totaling 370 images, allocated for training the Siamese model. The remaining 30 images, constituting three sets, are reserved specifically for evaluating and testing the model's accuracy. This dataset distribution allows for a comprehensive training regimen while ensuring a robust assessment of the model's performance on unseen data, enabling a thorough validation of the Siamese network's efficacy in facial recognition and similarity assessment tasks.

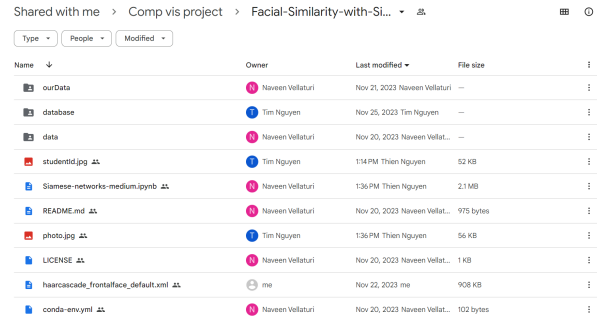
## 5. Solution Method

### 5.1. Usage of Google Drive

We seamlessly transitioned our database of images to Google Drive, a versatile cloud storage solution, streamlining our image management process. By leveraging Google Drive's robust features and accessibility, we transformed it into a repository for our image dataset. The platform's scalability allowed us to efficiently organize, store, and retrieve images, while its collaborative capabilities enabled multiple team members to access and work on the dataset simultaneously. Additionally, Google Drive's integration with various applications and APIs facilitated easy connectivity, empowering us to seamlessly incorporate the images into our workflow, conduct analyses, and utilize them in our machine learning models or other applications with convenience and reliability.

### 5.2. Google Colab

Google Colab offers an incredibly convenient and versatile environment for our code base. Its integration with Google Drive allows seamless collaboration and easy access



Name	Owner	Last modified	File size
ourData	Naveen Vellaturi	Nov 21, 2023 Naveen Vellaturi	—
database	Tim Nguyen	Nov 25, 2023 Tim Nguyen	—
data	Naveen Vellaturi	Nov 20, 2023 Naveen Vellaturi	—
studentid.jpg	Tim Nguyen	1:14 PM Thien Nguyen	52 KB
Siamese-networks-medium.ipynb	Naveen Vellaturi	1:36 PM Thien Nguyen	2.1 MB
README.md	Naveen Vellaturi	Nov 20, 2023 Naveen Vellaturi	975 bytes
photo.jpg	Tim Nguyen	1:36 PM Thien Nguyen	56 KB
LICENSE	Naveen Vellaturi	Nov 20, 2023 Naveen Vellaturi	1 KB
haarcascade_frontalface_default.xml	me	Nov 22, 2023 me	908 KB
conda-env.yml	Naveen Vellaturi	Nov 20, 2023 Naveen Vellaturi	102 bytes

Figure 1. An image of our google drive

to datasets, code files, and libraries. With Colab's cloud-based infrastructure, we can execute complex computations using powerful hardware accelerators like GPUs and TPUs, speeding up our workflow significantly. The ability to write and execute code in Python within a browser, along with the pre-installed libraries like TensorFlow, PyTorch, and others allow us to develop and experiment with our code. Furthermore, Colab notebooks are easily accessible and shareable among the members of a team without having to constantly push and pull from a centralized source.

### 5.3. Siamese Network

A Siamese network was effectively leveraged in our solution to tackle image comparison tasks. Initially, pairs of images were constructed by iterating through our labeled dataset, forming associations or pairs between individual pictures. This approach enabled the model to discern unique features between images. Moreover, due to limitations in the dataset size, strategies were devised to augment the available images. Techniques such as brightness adjustments, random rotations, and horizontal flips were employed to expand the dataset, enhancing the variety and quantity of training samples.

We trained our network using a pre-trained model called Visual Geometry Group(VGG). VGG allowed us to utilized a pre-trained model that has learned to extract features and categorized them in certain images using ImageNet.

### 5.4. Take Picture using Camera

A feature allowing image capture through the device's built-in webcam was introduced. To isolate the face of interest within captured images, a method for facial cropping was integrated, focusing solely on the facial region while excluding extraneous elements.

### 5.5. Scan Student ID for ID Number

An image processing function was devised to extract and name images based on the embedded student ID, ensuring

```

import torch.nn as nn
import torchvision.models as models

class SiameseNetwork(nn.Module):
    def __init__(self):
        super(SiameseNetwork, self).__init__()

        # Use pre-trained VGG16 as the base model
        self.vgg16 = models.vgg16(pretrained=True)

        # Modify the first convolutional layer to accept one channel
        self.vgg16.features[0] = nn.Conv2d(1, 64, kernel_size=3, padding=1)

        # Modify the fully connected layers to match your output size
        in_features = self.vgg16.classifier[6].in_features
        self.vgg16.classifier[6] = nn.Sequential(
            nn.Linear(in_features, 500),
            nn.Linear(500, 5)
        )

    def forward_once(self, x):
        output = self.vgg16(x)
        return output

    def forward(self, input1, input2):
        output1 = self.forward_once(input1)
        output2 = self.forward_once(input2)
        return output1, output2

```

Figure 2. An image of our Siamese network using a pre-trained VGG model

```

def take_photo(filename='photo.jpg', quality=0.8, flip=False, rotate=False):
    js = Javascript("""
    async function takePhoto(quality) {
        const div = document.createElement('div');
        const capture = document.createElement('button');
        capture.textContent = 'Capture';
        div.appendChild(capture);

        const video = document.createElement('video');
        video.style.display = 'block';

        const devices = await navigator.mediaDevices.enumerateDevices();
        console.log("Available video devices:", devices);

        //const stream = await navigator.mediaDevices.getUserMedia({video: true});
        const stream = await navigator.mediaDevices.getUserMedia({video: {
            deviceId: "aed8266c6a448710f819ee8d6599b70e9cfc1ff8651d6fbca77f215e1881f344",
        }});

        document.body.appendChild(div);
        div.appendChild(video);
        video.srcObject = stream;
        await video.play();

        // Resize the output to fit the video element.
        google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

        div.style.width = document.documentElement.scrollHeight + 'px';
        div.style.height = document.documentElement.scrollWidth + 'px';
        div.style.transform = "rotate(90deg)";

        // Wait for Capture to be clicked.
        await new Promise((resolve) => capture.onclick = resolve);

        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        canvas.getContext('2d').drawImage(video, 0, 0);
        stream.getVideoTracks()[0].stop();
        div.remove();
    });
    """);

```

Figure 3. A picture taking algorithm using embedded javascript

a seamless correspondence between the image and the respective student identifier.

## 5.6. Use ID Number for Database Search

This Python code segment uses Tesseract, an OCR tool, to extract text information from an image (thresh). It then employs a regular expression pattern to find and extract an ID number from the extracted text. The code prints the raw text and the extracted ID. Subsequently, it runs a loop to check if the corresponding image file linked to the extracted ID ex-

```

extractedInformation = pytesseract.image_to_string(thresh)
import re

def extract_id(input_string):
    # Define a regular expression pattern to match the ID
    pattern = r'ID# (\d+)'

    # Use re.search to find the match in the input string
    match = re.search(pattern, input_string)

    # Check if a match is found
    if match:
        # Extract the ID from the match
        id_number = match.group(1)
        return id_number
    else:
        return None

print('raw extraction: ', extractedInformation)
result = extract_id(extractedInformation)
print('result:', result)

while True:
    # Check if the file does not exist in the database
    if len(extractedInformation) == 0 or cv2.imread("./database/{}.png".format(result)) is None:
        print("Your data was not found. Please enter your ID manually.")
        # Set a custom value when the file doesn't exist in the database
        result = input("Enter Student ID: ")
    else:
        break # Exit the loop if the condition is met

```

Figure 4. An ID reading algorithm using embedded javascript

ists in our database. If the extracted information is empty or if the associated image file is not found, it prompts the user to manually input their student ID for further verification. Once the file is located in the database, the loop concludes, allowing the code to proceed..

## 5.7. Compare Picture with Stored Image

This code snippet executes a Siamese network to generate embeddings for a pair of input images (final tensors[0] and final tensors[1]). The network computes the outputs output1 and output2, representing the embeddings for each image. Utilizing these embeddings, the code calculates the Euclidean distance between the two outputs through F.pairwise distance(). The resultant distance is then displayed visually by concatenating the input images and showing them using the imshow function. This function not only presents the concatenated images but also annotates the dissimilarity score (computed Euclidean distance) between the paired images. The visualization aids in understanding the network's assessment of the dissimilarity between the image pairs.

## 6. Results

Following the establishment of our Siamese network utilizing a pre-trained VGG model, our experimentation led to remarkable results, showcasing an almost perfect accuracy of nearly 96 percent on our testing dataset. However, despite this outstanding performance within the controlled testing environment, our model encountered challenges when exposed to real-world data. This discrepancy in performance could potentially be attributed to the limited quantity and diversity of images used for training. The impact of an insufficiently diverse and representative dataset might have hindered the model's ability to generalize effectively to real-world scenarios. Consequently, while our model excelled within the confines of the testing data, its limitations in han-

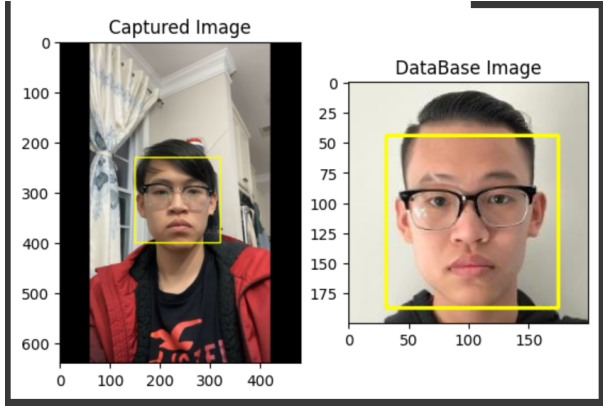


Figure 5. The two inputs for our test demo. Left image is the webcam image taken for the demo and the right consists of the image within our student database to be extracted for comparison.

dling real-world data underscore the significance of data diversity and volume in training robust and adaptable models for practical applications. We believe with proper training and more diverse image data we can train our models to be effective in real-world situations.



Figure 6. The resulting score of one demo run. The value of 0.83 shows that the model works and that the two images are of the same person.

## 7. Limitations

The proposed project faces two primary concerns: overfitting and challenges related to real-world application reliability. Overfitting, the phenomenon where a model becomes too tailored to its training data, jeopardizes its ability to generalize to new, unseen data. To address this, the project suggests transitioning to a more complex and diverse data set, introducing a broader range of scenarios and variations. Utilizing stronger GPUs for faster training allows for extensive experimentation, enhancing the model's adaptability to real-world conditions. Simultaneously, the unreliability of the system in real-world scenarios is acknowledged as a significant limitation. The strategy to improve real-world application reliability involves

focusing on enhancing the model's robustness to diverse and dynamic conditions. This includes expanding the dataset to encompass a representative range of real-world scenarios, enabling the model to better handle variations in lighting, backgrounds, and other environmental factors, ultimately improving its performance beyond controlled environments.

## 8. References

- **GitHub:** <https://github.com/harveyslash/Facial-Similarity-with-Siamese-Networks-in-Pytorch/tree/master>
- X. Chen, X. Yang, M. Wang, and J. Zou, "Convolution neural network for automatic facial expression recognition," 2017 International Conference on Applied System Innovation (ICASI), Sapporo, Japan, 2017, pp. 814-817, doi: 10.1109/ICASI.2017.7988558.
- Assran, M., Caron, M., Misra, I., Bojanowski, P., Bordes, F., Vincent, P., Joulin, A., Rabbat, M., & Ballas, N. (2022). Masked Siamese Networks for Label-Efficient Learning. ArXiv. /abs/2204.07141
- Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016a, April 11). Joint face detection and alignment using multi-task cascaded convolutional networks. arXiv.org. <https://arxiv.org/abs/1604.02878>
- Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML deep learning workshop. Vol. 2. No. 1. 2015.
- I. Melekhov, J. Kannala and E. Rahtu, "Siamese network features for image matching," 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 2016, pp. 378-383, doi: 10.1109/ICPR.2016.7899663.