SWINBURNE UNIVERSITY OF TECHNOLOGY

COS30082

ASSIGNMENT 2

(DUE 29/11/24 - 23:59P.M)

NAME & STUDENTID	:	**Nguyen Gia Binh - 104219428** **Nguyen Dang Huy - 103836512** **Bui Tran Gia Bao - 104177225**
CLASS	:	**COS30082 - Applied Machine Learning**
LECTURER	:	**Mr. Duong Trung Tin** **(**tinduong@swin.edu.au**)**
TUTOR	:	**Mr. Duong Trung Tin** **(**tinduong@swin.edu.au**)**

Hanoi, November 27, 2024

TABLE OF CONTENT

I) Executive Summary

1) Project Overview

This project focuses on developing a Face Recognition Attendance System with an integrated anti-spoofing mechanism. It utilizes advanced neural network architectures, including ResNet50 for face position detection and VGG-Face for face recognition, alongside liveness detection to prevent spoofing attempts. The system emphasizes computational efficiency, real-time performance, and security, making it ideal for practical applications such as attendance tracking and access control.

2) Objective

The main objective is to leverage existing models and techniques, apply them to create a lightweight face detection/tracking, recognition, and anti-spoofing capability. Furthermore, more techniques and models like GAN will be tested to see if they can improve parts of the system that require improvement.

II) Introduction

Assignment 2 focuses on applying a combination of techniques and models to develop a face attendance system**.** This system must fulfill two key requirements: recognizing known faces that have been registered beforehand and incorporating an anti-spoofing mechanism. In this assignment, we achieved not only face recognition and anti-spoofing but also face tracking and recognition for multiple detected faces within a frame. The completed system is a combination of models from OpenCV and DeepFace. Additionally, we experimented with Real-ESRGAN and GFPGAN for image enhancement before feeding the images into the model for detection and recognition over longer ranges. Although none of the GANs made the final cut, the data and experience gained from this experimentation were invaluable.

III) Methodology

1) Face registration

a. Workflow

The face registration process begins with users entering their name into an input field and pressing the "Open Camera" button. Users are instructed to press the spacebar to capture an image. Each captured image is temporarily saved in a directory named "buffer". A custom-trained model, using ResNet-50 as its base, analyzes the image to determine the face's orientation (e.g., up, down, left, right, front). The detected position is cross-referenced with a predefined face_positions array containing the orientations ["up", "down", "left", "right", "front"]. If a valid position is detected, the position is removed from the face_positions array, the image is moved from the “buffer” folder to a directory labeled with the user’s name, located in the "Face testing" folder. The process repeats until the user successfully captures images representing all five predefined positions. After capturing these five images, the user is allowed to take two additional images without orientation checking. These images are typically captured with the user's face tilted down to the left and down to the right. Including these positions enhances recognition performance, particularly for ceiling-mounted cameras commonly found in environments like apartments.

b. Face position model

To detect face positions during registration, we experimented with multiple custom-trained models. The dataset consisted of five classes (up, down, left, right, front), with images organized into directories for each class. To enhance dataset variability and improve generalization, we applied data augmentation techniques, such as rescaling, using Keras' ImageDataGenerator. While we explored additional augmentation methods, they did not yield improvements, and the resulting models exhibited poorer predictive performance.

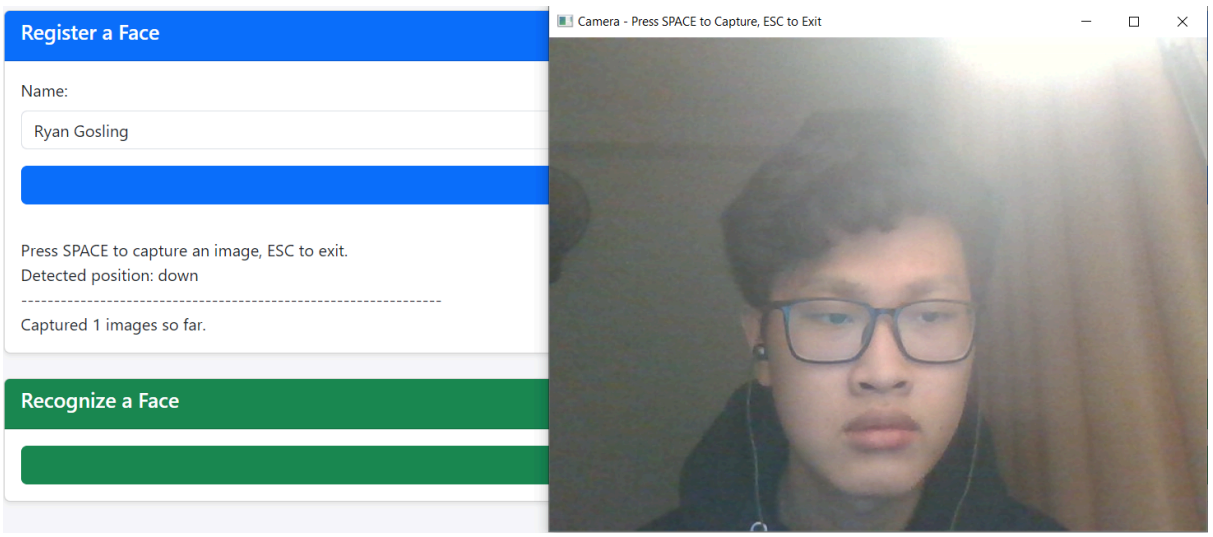
```
img_size = (224, 224)
batch_size = 32

# Data Generators
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    classes=['up', 'down', 'left', 'right', 'front'],
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    classes=['up', 'down', 'left', 'right', 'front'],
    subset='validation'
)
```

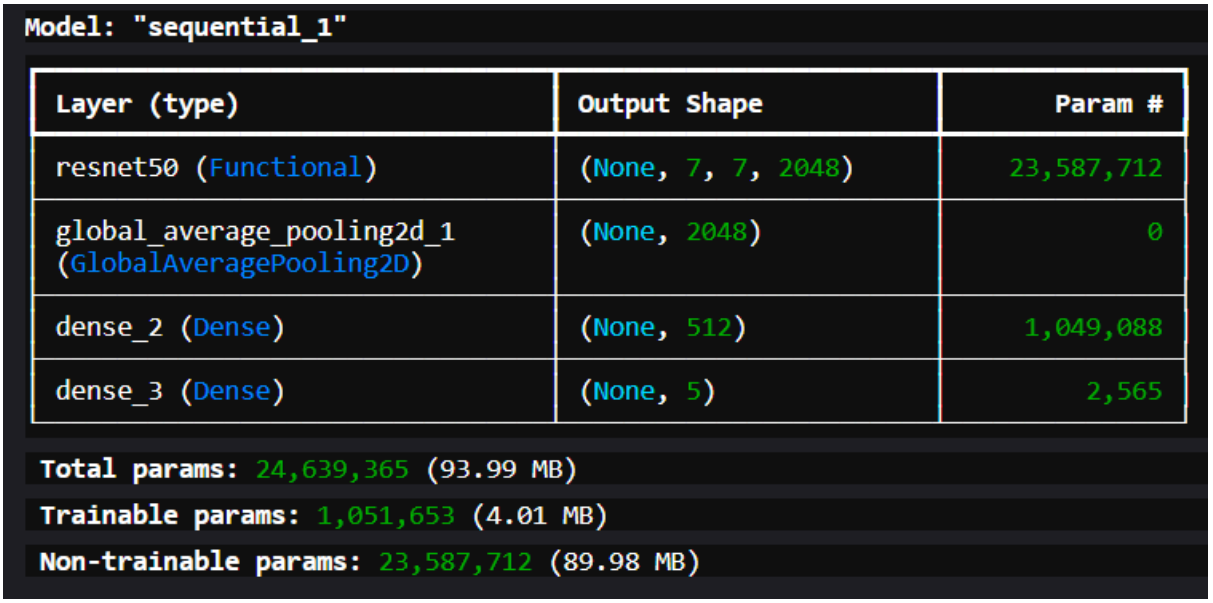
- **VGG16 Model (head_orientation_model.h5):** Trained on a smaller dataset we initially collected online, this model struggled to distinguish between the "front" and "down" positions due to dataset limitations.



- **ResNet-152 Model:** Although we attempted to expand the dataset by collecting additional images online, the increase was minimal, and the dataset remained relatively small. The complexity of the ResNet-152 architecture required a significantly larger dataset to perform effectively. As a result, the model struggled to generalize and exhibited extremely poor performance during validation. Given these limitations, we decided to exclude this model from the final submission as it did not align with the performance requirements for our system.
- **ResNet-50 Model (head_orientation_model_ResNet50.keras):**

  To address the issue of dataset size, one of our team members contributed by taking approximately 500 images for each of the five face positions, resulting in a total of 4330 images for training and 1079 images for validation, split with an 80-20 ratio.

  We selected ResNet-50 (pre-trained on ImageNet) as our final base model for its balance between complexity and performance. Unlike ResNet-152, ResNet-50 was more suitable for our smaller dataset. Additionally, we had prior experience training ResNet-50 models in our previous assignment, which made it a practical choice. The top layers of ResNet-50 were replaced with a global average pooling layer to reduce the spatial dimensions of feature maps while retaining essential features, followed by a dense layer with 512 units and ReLU activation to introduce non-linearity and enhance feature extraction, and a final softmax layer for 5-class classification. The base layers of ResNet-50 were frozen to preserve the pre-trained features and focus training on the new layers.



Model: "sequential_1"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 2048)	0
dense_2 (Dense)	(None, 512)	1,049,088
dense_3 (Dense)	(None, 5)	2,565

Total params: 24,639,365 (93.99 MB)

Trainable params: 1,051,653 (4.01 MB)

Non-trainable params: 23,587,712 (89.98 MB)

To optimize the training process, we implemented the several callbacks:

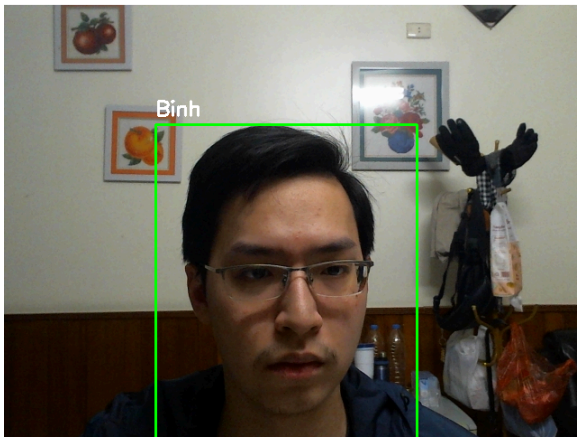
- Early Stopping: Configured with a patience of 60 epochs to prevent overfitting, as shorter patience (e.g., 20 epochs) in previous experiments caused training to terminate prematurely.
- ReduceLROnPlateau: Dynamically reduced the learning rate from 0.001 to a minimum of 1e-6 when validation performance plateaued.
- ModelCheckpoint: Saved the best-performing model and periodic checkpoints to ensure training progress was preserved, especially in cases where our Colab or Kaggle session was terminated during training.

2) Face Detection

The face detection module employs the SSD ResNet-10 model which is a lightweight architecture designed for face detection by OpenCV. This model was selected for the balance of speed and accuracy that it brings to the table, this is particularly important because all 3 of our laptops can only run anything on the CPU alone.

The process begins by preprocessing the input image frame. The frame is resized to 300x300 pixels and normalized using mean subtraction, creating a 4D blob suitable for the SSD model. The blob is then passed through the model which outputs a list of detected bounding boxes and their associated confidence scores. Bounding boxes with confidence scores exceeding 0.5 are selected, and their coordinates are scaled back to match the original image dimensions. Bounding boxes are dynamically adjusted to include margins for better visibility and every 2 seconds the next

bounding box drawn will be used to extract the face region and be used for face recognition and spoofing check.

Full image	Bounding box extracted
	

3) Face Recognition

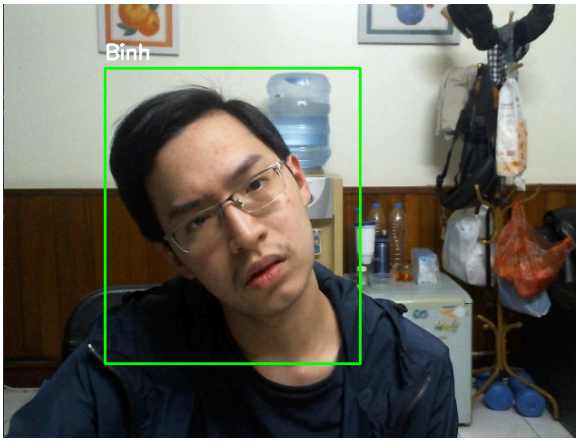
The face recognition module utilizes 2 different models provided when installing the Deepface framework which are VGG-Face for recognition and MTCNN as its backend detector.

VGG-Face was selected for its robust feature extraction capabilities, trained on a diverse dataset of over 2.6 million faces spanning 2622 identities (Parkhi et al., 2015), which allows it to generalize well in real-world scenarios. Its architecture is efficient enough to run on CPU-only environments like ours, striking a balance between computational demand and performance.

MTCNN was chosen as the face detection model because of its accuracy and reliability in detecting facial landmarks and bounding boxes. Its cascaded architecture processes images in stages, allowing it to handle challenging conditions such as variations in lighting, facial angles, or partial occlusions moderately effective.

The VGG-Face model computes embeddings - a numerical representation of facial features and compares these embeddings against a database of pre-stored embeddings for known identities. If the similarity score between the input and stored embeddings is in range, the system labels the face with the corresponding name and If no match is found, the face is marked as "Unknown".

The use of MTCNN and VGG-Face ensures that the system remains efficient and optimized for CPU-only environments like ours. MTCNN ability to handle detection with minimal computational overhead, combined with VGG-Face's pre-trained embeddings allows the system to achieve real-time performance without requiring high-end hardware.

Input image	Output
	

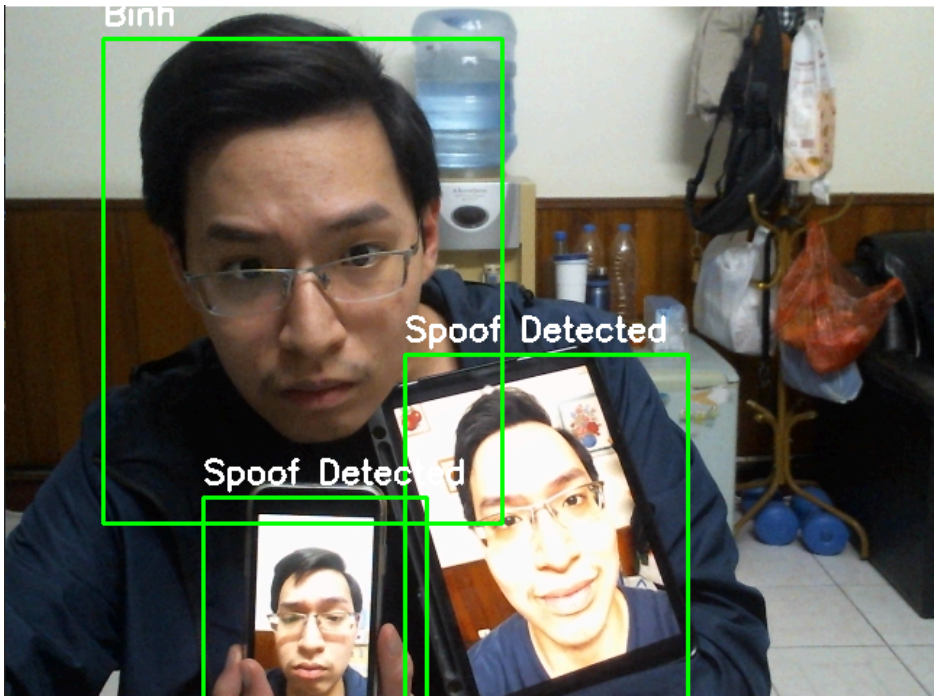
4) Spoofing Detection

The Silent Face Anti-Spoofing framework was chosen for its effectiveness in identifying spoofed faces, such as those generated using photos and videos. The system uses multiple pre-trained models to analyze features like texture, depth, and though unreliable even motion inconsistencies, ensuring accurate detection of spoofed inputs.

The anti-spoofing process begins with the extraction of the face region from the detected bounding box. The cropped face is analyzed by the Silent Face Anti-Spoofing framework which utilizes multiple pre-trained models pre-provided to do this. Each model independently evaluates the input, generating predictions for whether the face is "Real" or "Fake." These predictions are aggregated to provide a final classification.

The anti-spoofing module is integrated directly into the face recognition system. After a face is detected and recognized, it undergoes a spoof detection step to verify its authenticity. If the face is classified as "Fake," it is flagged and excluded from further processing. The results of the anti-spoofing step, along with the recognition output, are displayed in real-time on the video feed, ensuring a secure and reliable user experience.

In summary, the Silent Face Anti-Spoofing framework provides an effective and efficient solution for detecting spoofed faces, significantly enhancing the security and reliability of the face recognition system. Its lightweight and CPU-friendly design makes it well-suited for real-time applications in our CPU-only environment, offering a robust defense against spoofing attempts.



IV) Results and Discussion

1) Face registration

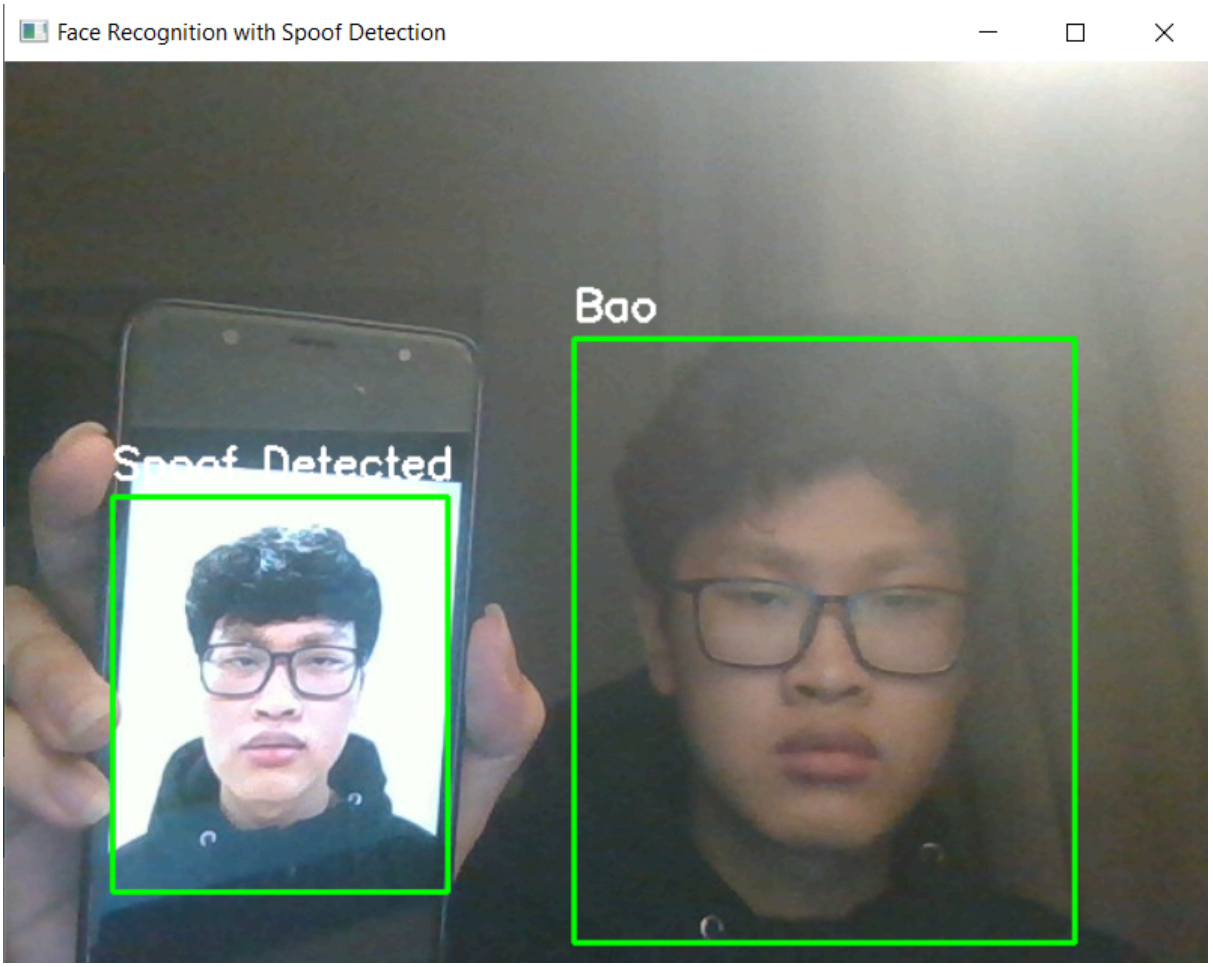
The best-performing ResNet-50 model for face position detection was saved at epoch 80, with a validation accuracy of 78.26% and a validation loss of 41,18%.

```
Epoch 40: val_loss improved from 0.53193 to 0.41184, saving model to /kaggle/working/new_best_
model.keras
135/135 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - accuracy: 0.7812 - loss: 0.6512 - val_accuracy: 0.78
26 - val_loss: 0.4118 - learning_rate: 1.0000e-06
```

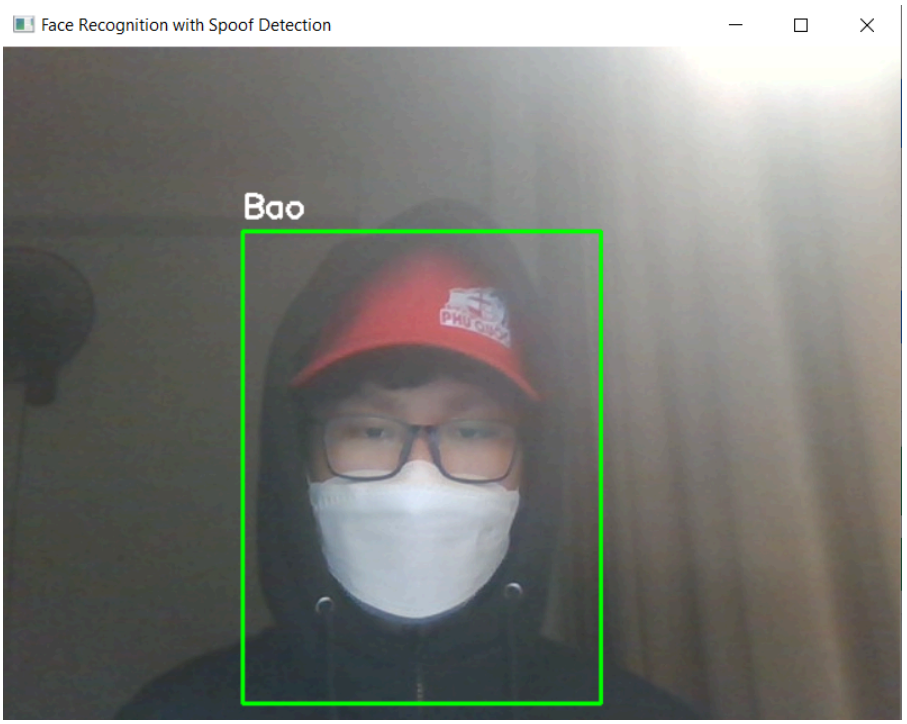
Despite its reasonable performance, the model exhibited flaws, often predicting the "right" position excessively. To provide an alternative for situations where the model's performance is too frustrating to use, a dummy function was incorporated. This dummy function can be used as a fallback by commenting out the model inference code. It allows the system to operate without checking the face position, cycling through predefined face positions in a sequential order.

2) Face recognition

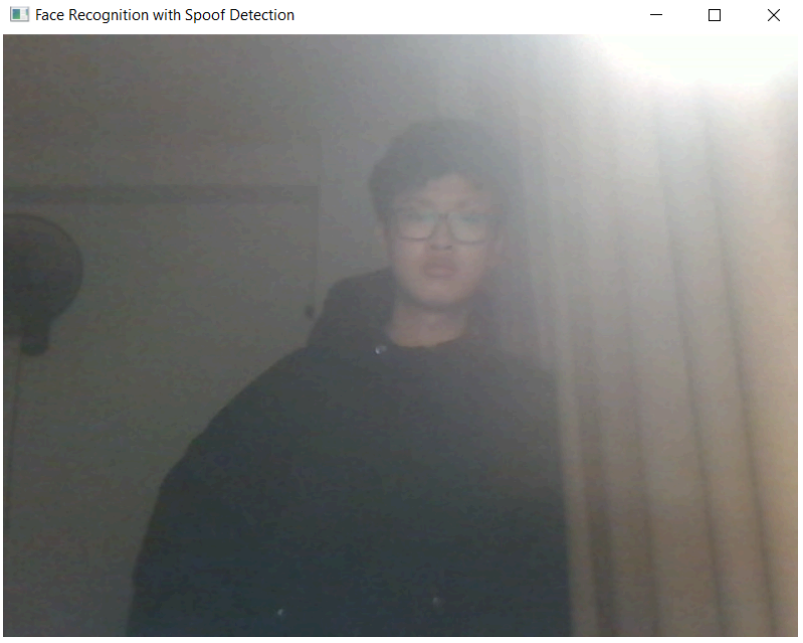
The face recognition system demonstrated a strong capability to distinguish between real faces and spoofed attempts, a critical feature for ensuring the system's security. When the system detects a spoofed face (e.g., a photo or video replay), it promptly labels it as "Spoof Detected" This anti-spoofing mechanism helps prevent unauthorized access and maintains the integrity of the recognition process.



Another highlight of the system is its ability to recognize users wearing accessories such as masks or hats. This robustness ensures that users can be authenticated even under varying facial conditions, which is particularly important in scenarios like post-pandemic environments where face masks are widely used.



However, the system exhibited limitations under challenging lighting conditions. For example, in situations with overexposure or dim lighting, the system struggled to detect faces. This limitation is likely due to the insufficient quality of the captured frames, which hinders the face detection and recognition model from extracting meaningful features. Such conditions can result in no face being detected, impacting the system's reliability.



V) Experiment

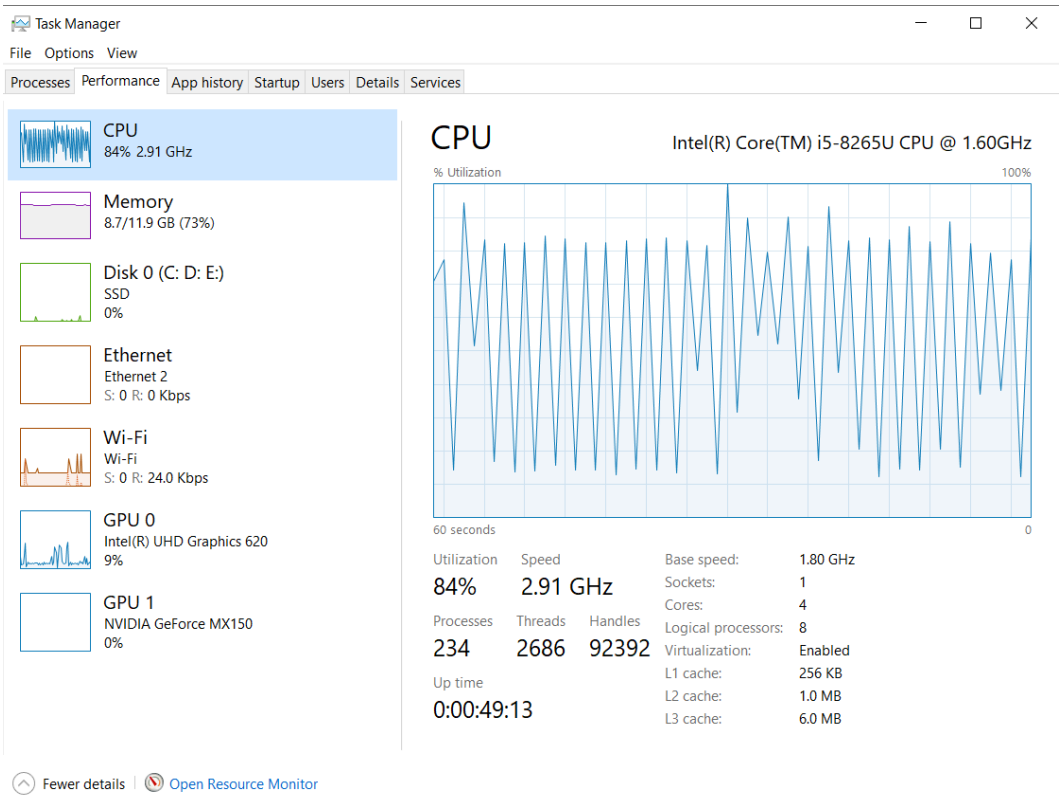
1) Real-ESRGAN

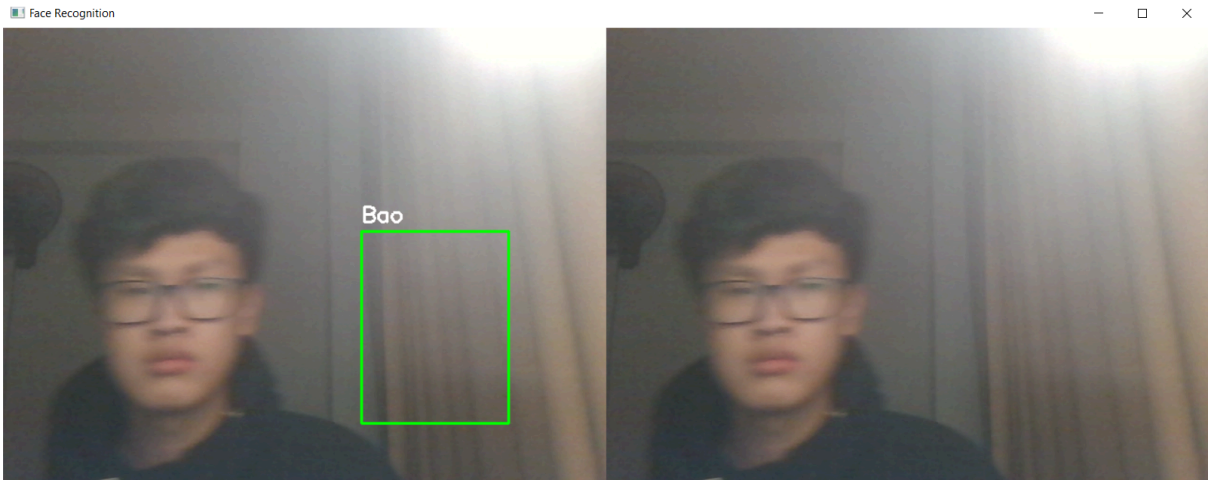
To enhance the recognition ability of our system, we experimented with Real-ESRGAN, a GAN-based model designed for image enhancement. In this approach, each camera frame was first processed through the Real-ESRGAN model to improve image quality before passing it into the face detection function.

The enhanced frames were expected to improve the system's ability to accurately detect and recognize faces in challenging conditions, especially when dealing with long distances, motion blur, dim lighting, or low-resolution faces.

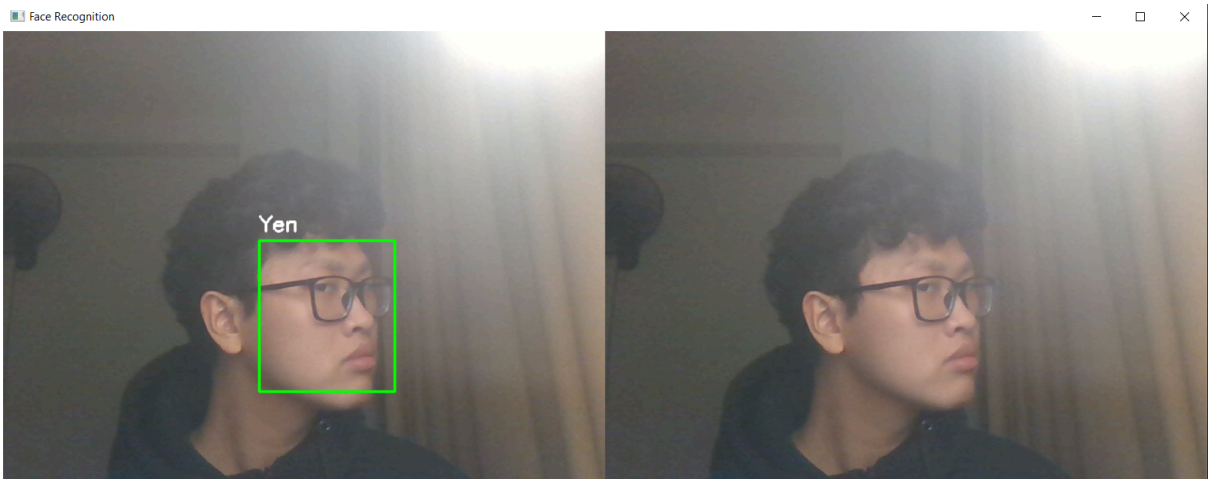
Contrary to our expectations, the implementation of Real-ESRGAN led to a decline in overall system performance:

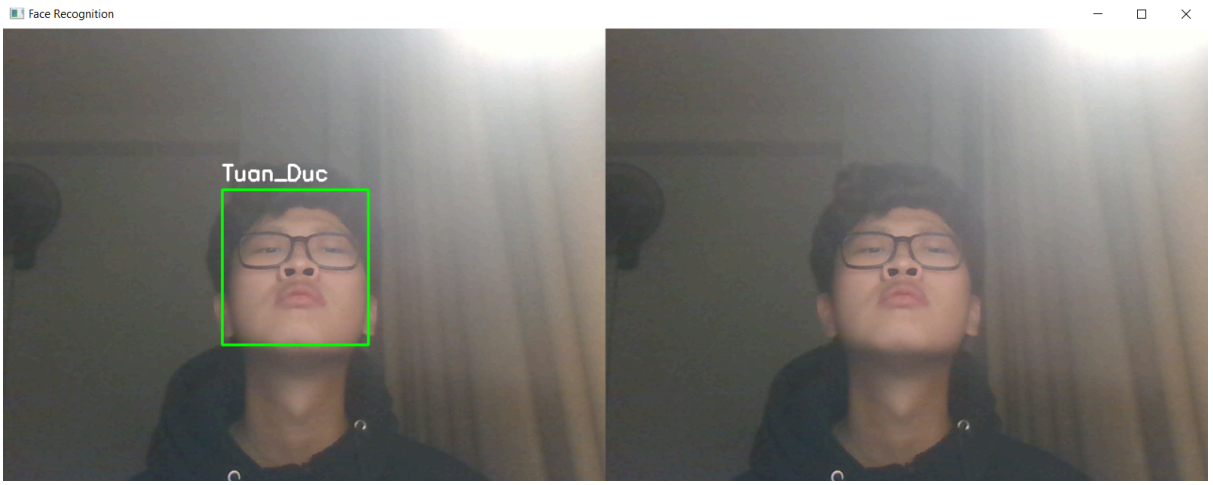
- **Performance Delay:** We evaluated the system's performance with and without the Real-ESRGAN model by analyzing CPU usage and response times during execution. When using Real-ESRGAN, the average CPU utilization increased to 84%, with periodic spikes, and the base speed rose to 2.91 GHz. However, the additional processing required for frame enhancement introduced significant computational overhead, resulting in slower overall system performance. This delay impacted face detection, causing the bounding box updates to become less responsive and unable to track faces effectively in real time.

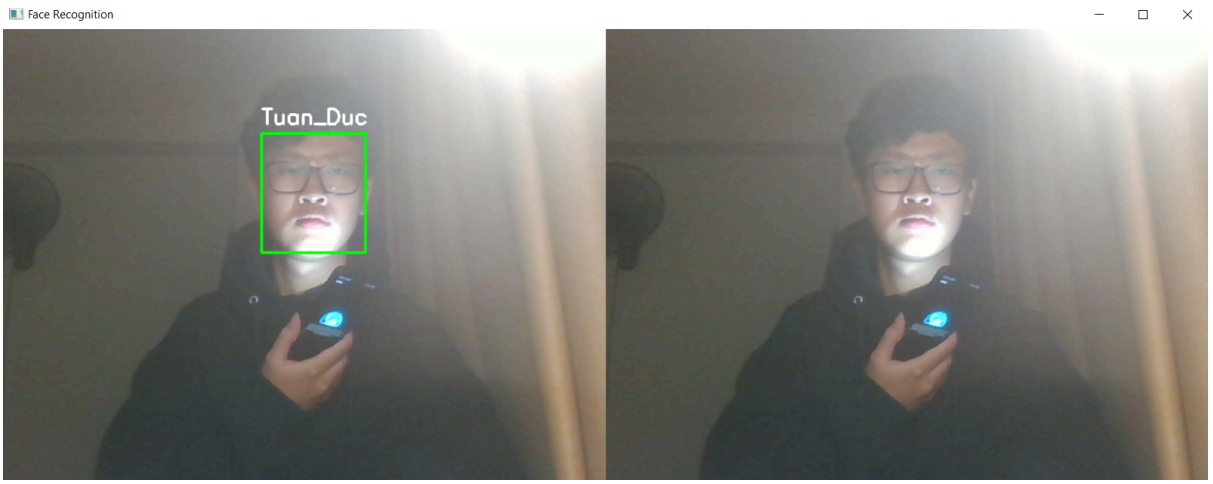




- **Lower Recognition Accuracy:** In cases where users turned their heads to the side, looked upward, or encountered unusual lighting conditions, the system occasionally misclassified users. This raises potential security concerns, such as the risk of granting access to unauthorized individuals in sensitive applications like bank authentication or apartment check-ins.





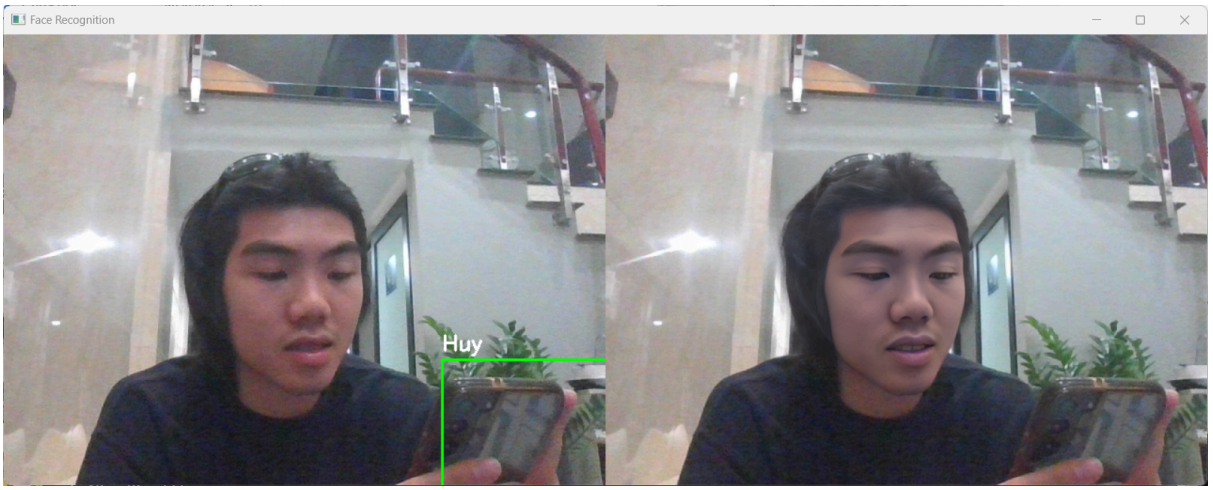


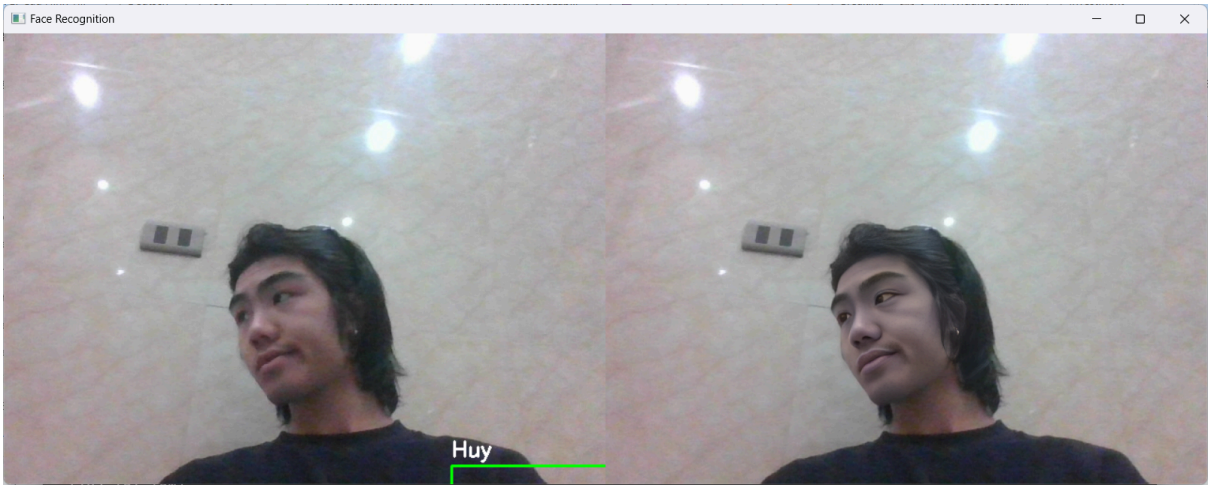
- **Inadequate Face Enhancement:** Real-ESRGAN is designed for general image and video restoration and is trained with synthetic data to handle various real-world degradations. However, it is not specifically optimized for face enhancement. Consequently, the enhanced frames did not show significant improvements in facial features compared to the original frames, limiting the expected benefits for face recognition tasks.

2) GFPGAN

Following our initial experiments, we experimented with GFPGAN, a model specifically designed for real-world face restoration. It utilizes diverse and robust priors derived from a pre-trained face GAN, such as StyleGAN2, to restore and enhance facial details in low-quality or degraded images (Wang et al., 2021).

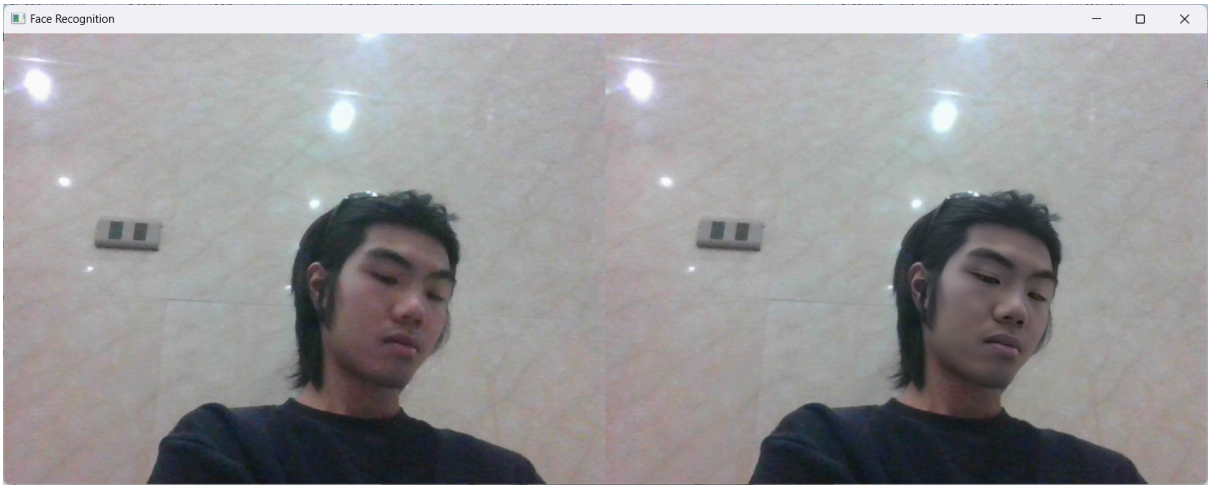
Ideal Conditions:

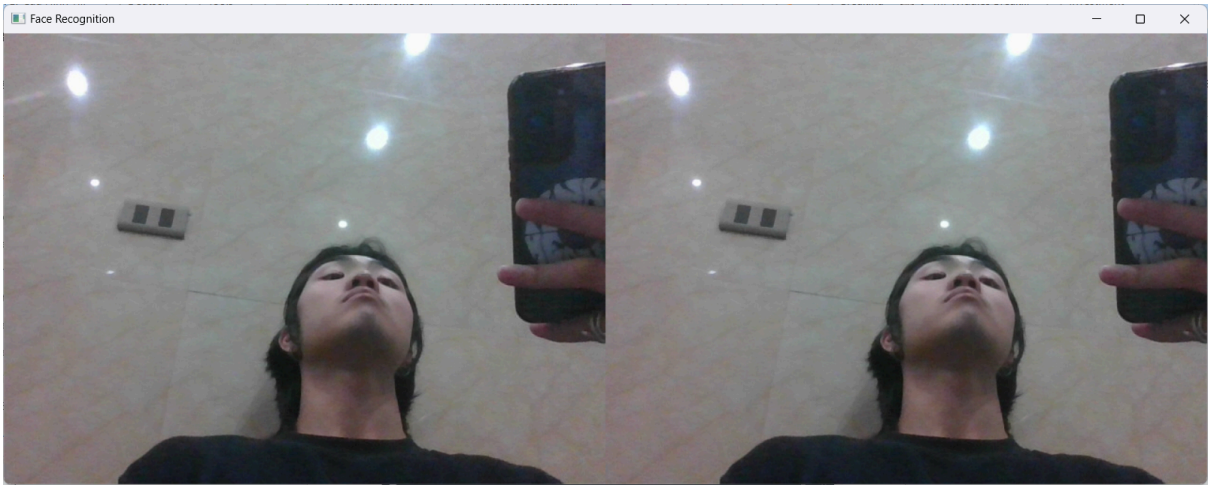




- GFPGAN performed well in ideal scenarios, such as when faces were captured directly from the front or at a slight side angle. The restored frames showed noticeable improvements in clarity, aiding face detection and recognition accuracy.

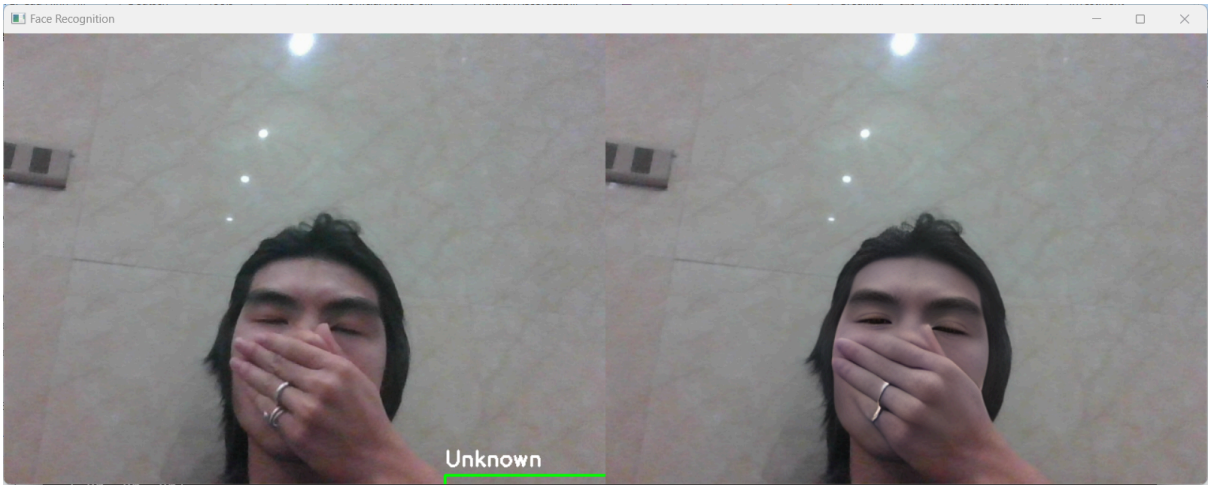
Slightly more challenging conditions:





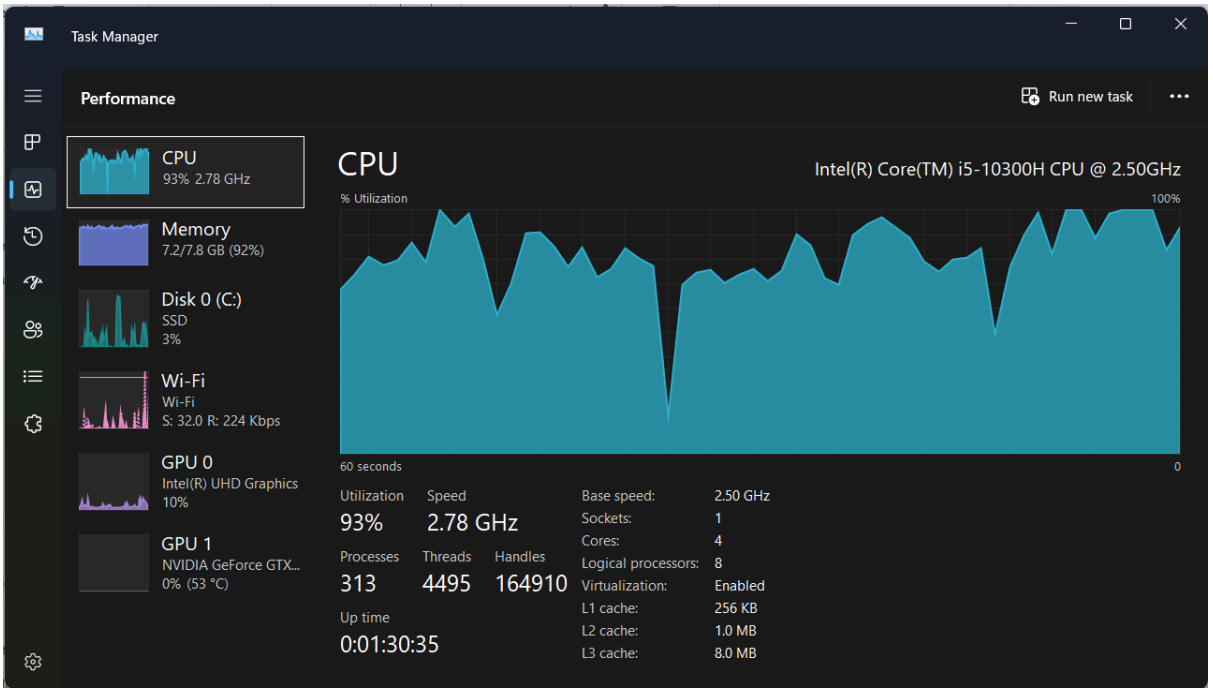
- For scenarios involving head poses where users looked upward or downward, GFPGAN struggled to provide reliable enhancements. The restored images exhibited minimal improvement in facial features, resulting in lower recognition accuracy.

Difficult cases:



- In cases of covered face, adverse lighting conditions or with significant motion blur, GFPGAN failed to produce substantial gains, with the quality of the enhanced images remaining comparable to the original frames.

CPU Utilization:



- Integrating GFPGAN led to moderate computational overhead, with CPU utilization averaging around 75% during processing, compared to 60% without the model.
- Spikes in CPU usage up to 88% were observed during periods of high activity, such as when multiple faces were detected in a frame.
- The processing speed reached up to 2.85 GHz, which, while slightly less intensive than Real-ESRGAN, still posed challenges for real-time processing in multi-threaded environments.

System Delays:

- While the delays introduced by GFPGAN were less pronounced compared to Real-ESRGAN, the additional computational load still caused minor latency. This resulted in bounding box updates being slightly slower, especially when handling frames with multiple faces or complex backgrounds.

VI) Contribution

Group member	Role/Responsibility
Nguyen Gia Binh	- Implemented and fine-tuning the main face detection - Implemented and fine-tuning the main face recognition - Implemented and fine-tuning the Silent face anti-spoofing - Helped implementing Real-ESRGAN
Nguyen Dang Huy	- Developed model for detecting face orientation - Conducted experiments with the GFPGAN model
Bui Tran Gia Bao	- Created the UI - Implemented the face registration feature - Assisted in collecting additional data for face orientation detection and trained the face orientation models - Conducted experiments with the two GAN models

VII) Appendix

1) Acknowledgment

- Deepface framework: https://github.com/serengil/deepface
- Caffe Model: Caffe Model
- Real-ESRGAN: https://github.com/xinntao/Real-ESRGAN
- GFPGAN: https://github.com/TencentARC/GFPGAN

2) Project resources

- Github repository: https://github.com/nguyengiabinh/Mini-Face-recognition-using-Deepface-and-GAN
- Project Setup and Demonstration: https://youtu.be/WWOcxMMynsM
- Face Orientation Model Training Notebook: https://www.kaggle.com/code/solar11781/faceorientationdetector
- Face Orientation Data: https://drive.google.com/drive/folders/1bcY4_lVWX8plJS7dwVjOHq4RRJGSz4SY?usp=sharing

VIII) References

Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep face recognition. *British Machine Vision Conference (BMVC)*. https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf

Wang, X., Li, Y., Zhang, H., & Shan, Y. (2021). Towards real-world blind face restoration with generative facial prior. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9168–9178. https://doi.org/10.48550/arXiv.2101.04061