VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

FINAL PROJECT REPORT

SUBJECT: COMPUTATIONAL THINKING

TOPIC
FAKE FACE DETECTION

Lecturer:	Ph.D. Ngô Đức Thành
Students:	Trần Hữu Lộc - 22520796
	Trần Gia Bảo - 22520119
	Nguyễn Tiến Huy - 22520567
	Trần Tuấn Khoa - 22520692
Class:	CS117.O21.KHTN

Ho Chi Minh City, July 2024

Table of Contents

I. Introduction:

Fake face detection is a complex problem that requires a multi-pronged approach, combining computer vision methods with computational theory methods. Computational thinking provides a systematic framework for breaking down problems into manageable chunks, analyzing them and solving them. The application of mechanical reasoning to fake face recognition allows us to design effective ways to recognize altered or synthetic facial images.

Image 1: Image from Phys.org

The foremost objective of this document is to analyze the application of computational simulation in fake face recognition cases. We explore how computational layout standards, including decomposition, pattern recognition, abstraction, and algorithmic design can be used to overcome challenges related to fake face recognition. Using these techniques, we aim to develop a potential solution that could effectively understand and distinguish between real and altered facial images.

In addition, this document will look at the algorithms and evaluation strategies in our solution. We used superior algorithms and imaging techniques to create a system that can accurately identify and distinguish between real and fake faces in images. This crucial functionality allows users to control hazards put into effect and efficaciously manipulate the risks associated with deepfake technology and other digital identities issues. With false identification technologies now, businesses and individuals can protect their identity and digital property from malicious activity.

We thank Ph.D. Ngo Duc Thanh for his priceless contributions to our information of computational theory and its programs in computer vision. Ph.D. Ngo Duc Thanh's understanding, steerage, and coaching had been essential to our technique to fake face detection using strategies and his determination and help enriched our getting to know enjoy and knowledgeable our studies the attempt became a hit.

II. Problem Identification:

The goal of Fake Face Detection problem is to develop methods and systems that can accurately identify whether a given facial image as input is real or generated by artificial intelligence, such as deepfake technology.

Effective fake face detection helps safeguard personal and societal interests by ensuring that visual content is reliable and trustworthy.

- **Input**: A single image containing one or multiple faces.

- **Output**:
  - **Face Coordinates**: The coordinates of detected faces within each image.
  - **Classification Labels**: A label for each detected face indicating whether it is "Real" or "Fake".
  - **Confidence Scores**: The probability or confidence score associated with each classification label.
  - **Annotated Image**: The original image with rectangles drawn around each detected face.

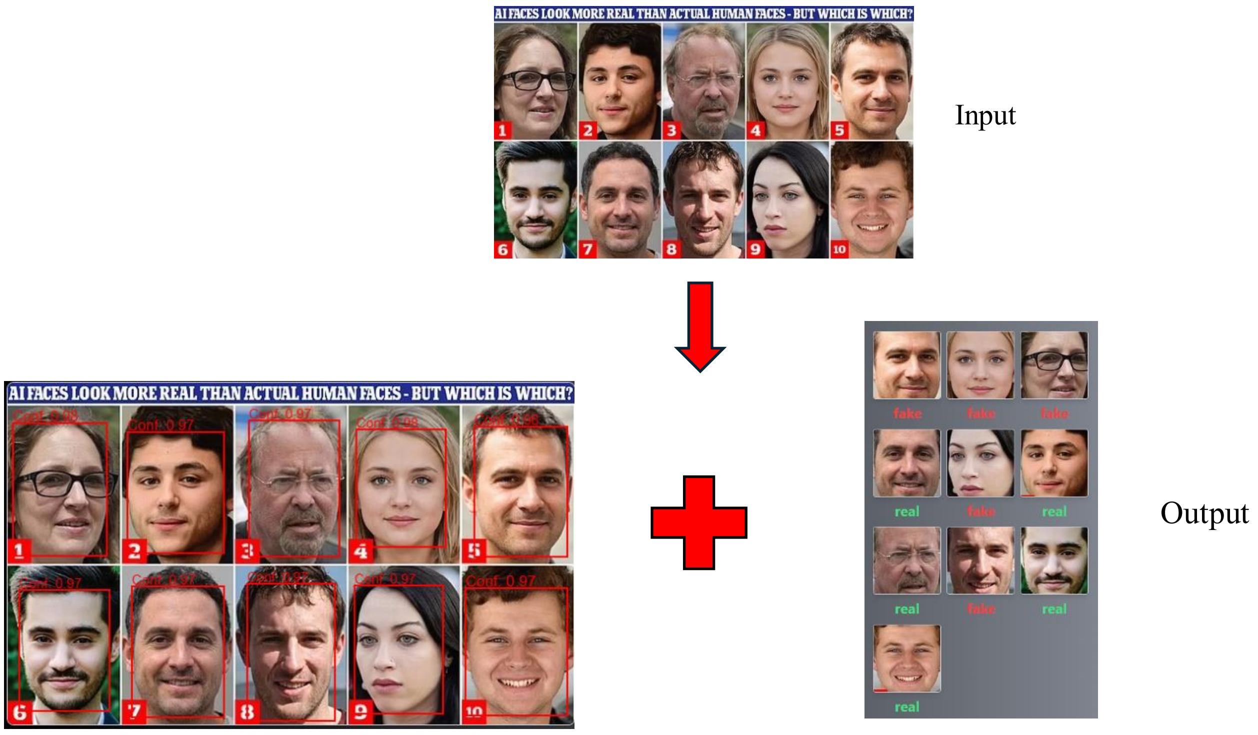

AI FACES LOOK MORE REAL THAN ACTUAL HUMAN FACES - BUT WHICH IS WHICH?
Input
AI FACES LOOK MORE REAL THAN ACTUAL HUMAN FACES - BUT WHICH IS WHICH?
fake
fake
fake
real
fake
real
real
fake
real
real
Output

Image 2: Input and output visualization

III. Abstraction:

1. Input Data:

- **Image Format:** Input images must be in JPEG or PNG format.
- **Face Visibility:** The faces in the images must be clearly visible and unobstructed.
- **Lighting Conditions:** Input images should have consistent and moderate lighting conditions, avoiding extremes of very low or very high illumination.

2. Preprocessing:

- **Face Detection:** Detect and isolate faces within the provided images.
- **Confidence Scores:** Provide confidence scores for each face detected within the provided images.

3. Feature Extraction:

- **Facial Features:** Extract significant features from the face, such as texture, shape, and color patterns.
- **Anomalies Identification:** Identify potential indicators of fake faces, such as inconsistencies in texture, unnatural lighting effects, and irregular facial landmarks.

4. Classification:

- **Binary Classification:** Develop a model that outputs whether a face is real or fake.

5. Evaluation:

- **Accuracy Metrics**: Use metrics such as accuracy, precision, recall, F1_score, and mean average precision (mAP) to evaluate the model's performance

IV. Decomposition and Pattern Recognition:

1. Decomposition:

The overall task can be divided into two main sub-tasks: face detection and fake face classification. The result of the face detection sub-task acts as the input for the fake face classification sub-task.

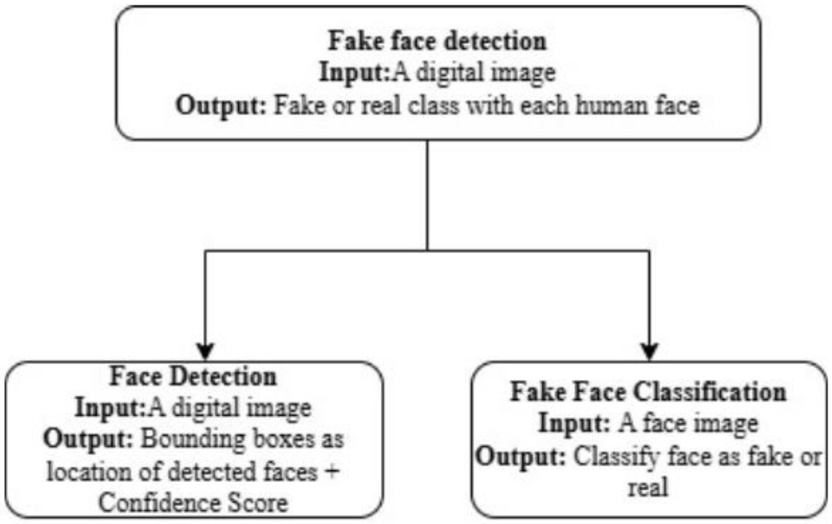

Fake face detection
Input:A digital image
Output: Fake or real class with each human face
Face Detection
Input:A digital image
Output: Bounding boxes as location of detected faces + Confidence Score
Fake Face Classification
Input: A face image
Output: Classify face as fake or real

Image 3: Decomposition Tree

2. Pattern Recognition:

2.1. Face Detection:

- **Load Face Detection Model:**
  - Use a pre-trained model for detecting faces.
- **Detect Faces:**
  - Apply the model to an image to locate faces.
  - Extract the coordinates of detected face regions.
- **Input:** A digital image
- **Output:**
  - Face Regions: A list of cropped face images.
  - Original Image with Face Coordinates: The original image with rectangles drawn around detected faces.
  - Face Coordinates: A list of tuples, each containing the coordinates (x, y, w, h) of detected faces.

2.2. Face Classification:

- **Preprocess Detected Faces:** Resize the extracted face regions to match the input size expected by the classification model.
- **Load Classification Model:** Load a pre-trained deep learning model that can classify faces as real or fake.
- **Classify Faces:**
  - Use the model to classify each detected face as real or fake.
  - Output the classification results.
- **Input:** A list of cropped face images.
- **Output:**
  - Classification Results: A list of predictions for each face region, indicating whether each face is real or fake.

V. Dataset and Metric:

3. Dataset:

We have meticulously created a dataset comprising 1000 faces, ensuring a high level of diversity in terms of age, gender, and other demographic factors. This dataset includes 506 real faces and 494 fake faces, all carefully verified for authenticity. It is divided into three sets: training, validation, and testing, with ratios of 75%, 15%, and 10%, respectively. By structuring our dataset in this way, we aim to develop robust and accurate models for fake face detection, capable of performing well across various demographic groups.

4. Metric:

4.1. Precision:

- **Definition**: Precision is the ratio of correctly identified fake faces to the total number of faces identified as fake by the model.
- Precision measures the accuracy of the model in correctly identifying fake faces without mistakenly classifying real faces as fake. High precision indicates a low false positive rate.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Image 4: Formula to calculate precision

- In there:
  - **True Positives**: correctly identifies a fake face as fake.
  - **False Positives**: incorrectly identifies a real face as fake.

4.2. Recall:

- **Definition**: Recall is the ratio of correctly identified fake faces to the total number of actual fake faces.
- Recall measures the model's ability to detect fake faces among all fake faces present in the dataset. High recall indicates a low false negative rate.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Image 5: Formula to calculate recall

- In there:
  - **False Negatives**: incorrectly identifies a fake face as real.
  - **True Positives**: correctly identifies a fake face as fake.

4.3. F1_score:

- **Definition**: The F1_score is the harmonic mean of precision and recall.
- The F1 score provides a balanced measure of the model's accuracy, considering both precision and recall. It is particularly useful when there is an imbalance between false positives and false negatives.

$$\text{F1 Score} = \frac{2 \text{ x } (\text{Precision x Recall})}{\text{Precision + Recall}}$$

Image 6: Formula to calculate F1_score

4.4. Accuracy:

- **Definition**: Accuracy is the ratio of correctly classified faces (both real and fake) to the total number of faces.
- Accuracy gives an overall measure of the model's performance, but it can be misleading in the case of class imbalance (e.g., more real faces than fake faces).

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{True Positives + False Positives + False Negatives + True Negatives}}$$

Image 7: Formula to calculate accuracy

- In there:
  - **True Positives**: correctly identifies a fake face as fake.
  - **True Negatives**: correctly identifies a real face as real.
  - **False Positives**: incorrectly identifies a real face as fake.
  - **False Negatives**: incorrectly identifies a fake face as real.

4.5. Mean Average Precision (mAP):

- **Definition:** mAP is the meaning of the Average Precision (AP) calculated for each class in the dataset. AP summarizes the precision-recall curve for a specific class.

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_i$$

Image 8: Formula to calculate mAP

4.6. Inference Time:

- **Definition**: Inference time is the amount of time used to infer the model's prediction.

VI. Algorithm and Evaluation:

1. Algorithm:

For each sub-task in the decomposition step, we search for feasible solutions for each of them.

1.1. Face Detection Model:

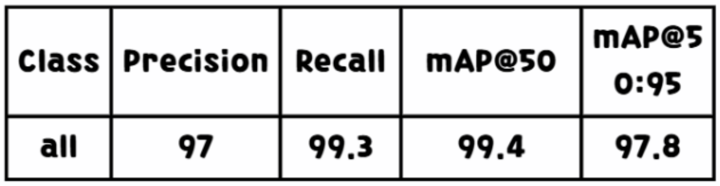
Among current object detection algorithms that give good results, our team considers performance and inference speed. Therefore, the team decided to use **Yolov5** as the face detection algorithm to solve the problem the team was solving.

1.2. Fake Face Classification Model:

We use ShuffleNetv2 as a model to classify images as fake or real face, a model pre-trained on the ImageNet1k dataset, with the ability to extract high-level features, in addition to a relatively small number of parameters. (2.1M). Therefore, my group decided to use this model.

2. Evaluation:

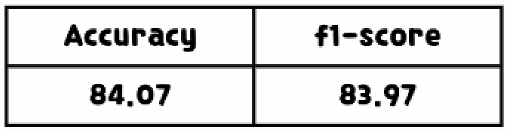
- Table 1 shows the results of the face detection problem about locating and classifying human faces used by our group.
  - Through the metrics, most of them reach over 97%, which proves that the algorithm our team used can detect most cases of face detection on the test set that our team collected with cases in 3 different views, with diverse skin colors, races, genders, etc.
  - In addition, the team measures inference time, with 0,31s, the speed is quite fast and stable, contributing to the ability to use the application in practice.



Class	Precision	Recall	mAP@50	mAP@50:95
all	97	99.3	99.4	97.8

Table 1: Experiment results of YOLOv5

- Table 2 shows the binary classification results of the problem of classifying human face images as real of fake images.
  - The experimental results demonstrate that the algorithm used by the group gives stable results, quite competitive with other methods.

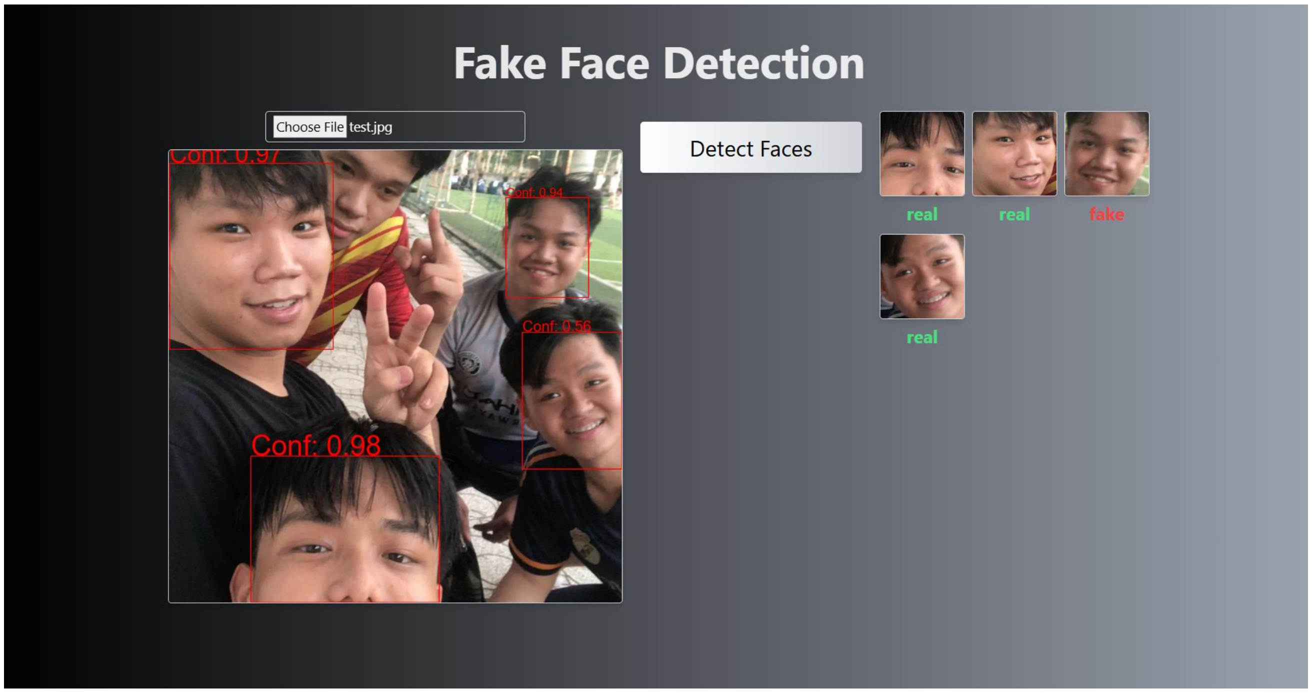


Accuracy	f1-score
84.07	83.97

Table 2: Experiment results of ShuffleNetv2

VII. Demo:

- To demonstrate that we have developed a solution to address the problem of fake face detection, we have created a website that provides a user-friendly interface. Users can upload an image file and click the "Detect" button to receive results. The system will analyze the uploaded image, extract all detected faces, and label each face as "Real" or "Fake". This interface showcases our detection model's capability to efficiently and accurately classify faces, making it easy for users to verify the authenticity of faces in their images.
- Below is the preview image of the application:



- You can view our project details via GitHub repository. There will be full source code, documents, setup methods and other related information resources that showcase the implementation of our solution.
- Click on this link https://github.com/trhuuloc/ComputationalThinking to access our GitHub repository.