VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

Midterm Progress Report

Face Detection

Course: CSC14006 - Pattern Recognition

Reported by:
22127230 - Duong Binh Nguyen Lan
22127320 - Bui Ta Phat
22127476 - Dang Trieu Kha
22127486 - Dao Ngoc Thien

Teacher:
Le Hoang Thai
Duong Thai Bao
Truong Tan Khoa

Ho Chi Minh city, 19th April 2025

Contents

I. Group members

Student ID	Student name
22127230	Dương Bình Nguyễn Lân
22127320	Bùi Tá Phát
22127476	Đặng Triệu Kha
22127486	Đào Ngọc Thiện

II. Fundamental method

1. Related works

As the first step in an automatic face recognition pipeline, the reliability of the face detection stage plays a pivotal role, directly affecting the performance and applicability of the entire system. In modern approaches, face detection is typically formulated as a binary classification problem, in which sub-windows scanned from the input image are classified into two categories: face and non-face.

Before 2001, several methods were proposed based on anthropometric measurements, such as the size and distance between the nose, eyes, and mouth, to locate faces. However, these approaches yielded low recognition performance, leading to slow progress in the field. Other methods used dimensionality reduction techniques, such as principal component analysis (PCA) combined with classifiers such as Support Vector Machines (SVM). Although they improved accuracy to some extent, these methods still suffered from slow processing speeds and limited applicability in real-time scenarios.

It was not until Viola and Jones [1] introduced their face detection framework, based on the AdaBoost algorithm combined with a cascade classifier, that the field experienced a major breakthrough. Their method struck an effective balance between speed and accuracy, laying the foundation for numerous subsequent advances in face detection research.

2. AdaBoost-based method

In machine learning, the AdaBoost algorithm constructs a strong classifier by linearly combining multiple weak classifiers. Its primary goal is to select the most effective weak classifiers, assigning them optimal weights to improve overall performance. However, AdaBoost itself does not directly learn the weak classifiers; instead, it assumes the existence of a base learning algorithm capable of training these classifiers based on the current weight distribution of the training data.

To enhance the efficiency of face detection, Viola and Jones integrated AdaBoost into a specialized structure known as a cascade of classifiers. This structure consists of a sequence of strong classifiers arranged in increasing order of complexity. Each stage in the cascade is a strong classifier built from a small set of weak classifiers. If an image window fails to pass any stage, meaning it is classified as a non-face, it is immediately discarded, significantly reducing computation time by quickly eliminating irrelevant regions.

The combination of the AdaBoost algorithm and the cascade structure resulted in a face detection method that is both fast and accurate, outperforming earlier techniques in terms of both speed and effectiveness. This method marked a breakthrough in the field and paved the way for practical applications in facial recognition, including surveillance systems, human–machine interfaces, and mobile devices.

III. State-of-the-art approach

1. Drawbacks of classical models

Classical face detection models, which are based on ensemble methods - techniques that combine multiple individual machine learning models to improve predictive accuracy over a single model - were once considered the standard, with AdaBoost being a prominent example. However, these methods now reveal significant limitations. They rely heavily on handcrafted feature selection, which leads to a lack of flexibility in modern applications where demands for accuracy, speed, and adaptability are increasingly stringent.

One of the major drawbacks of classical models lies in their high computational complexity, as they require the integration of multiple weak classifiers to construct a strong one. This not only increases processing costs but also results in slow inference times, especially when scanning images region by region.

Moreover, the handcrafted features used in these models are often insufficient to detect faces under occlusion, leading to poor performance in situations with incomplete visual information. Their accuracy also drops significantly when

faces are rotated or appear at steep angles, due to limited adaptability to geometric variations.
Training these models is another bottleneck, as they are generally not optimized to leverage modern hardware like GPUs, resulting in prolonged training times and suboptimal performance. In addition, classical models are highly susceptible to noise, particularly in scenes with complex backgrounds or varying lighting conditions, which contributes to a higher false detection rate.
Generalization is another critical issue, as model performance tends to be unstable when applied to large and diverse datasets. Notably, classical approaches struggle to detect multiple faces simultaneously in crowded scenes and show low sensitivity to small or low-resolution faces, increasing the likelihood of missing important targets during the detection process.

2. Intermediary advancements in Deep Learning-based methods and their remaining limitations

To overcome the limitations of handcrafted methods, deep learning models have gradually emerged, leveraging the power of Convolutional Neural Networks (CNNs) to automatically extract features and enhance face detection performance. Prominent architectures such as Faster R-CNN [2] have significantly improved detection accuracy by employing region proposal mechanisms to identify potential face regions, while also handling complex scenarios more effectively, such as occluded faces or those appearing at challenging angles.
However, despite these improvements in detection quality, these models still present several notable drawbacks. The face detection process requires multiple sequential steps - first generating region proposals, then performing classification - which introduces latency during inference. Computational complexity remains high, making these models less suitable for real-time applications. Additionally, their ability to detect small faces or faces in crowded scenes is still limited, as detection accuracy heavily depends on the quality of the region proposals. Training and deploying these models also demand substantial computational resources, particularly when operating on non-specialized hardware.

3. Introduction to YOLO Face V2

YOLO Face V2 [3] represents a significant advancement compared to traditional face recognition techniques, effectively overcoming the limitations of both manual methods and intermediate deep learning models. While conventional approaches often require scanning image regions sequentially or generating separate region proposals, which can result in slow processing and failure to detect small faces, YOLO Face V2 leverages convolutional neural networks to analyze the entire image in a single forward pass. This approach allows the model to quickly locate and recognize faces in real time while greatly reducing computational costs, delivering superior performance in both speed and accuracy.
From an architectural perspective, YOLO Face V2 is structured around three core components: the backbone, the neck, and the head, along with a specialized module known as SEAM (Spatial Enhancement Attention Module). The **backbone** serves as the foundation by using deep convolutional layers to automatically extract high-quality features from the input image, capturing details such as edges, shapes, and textures of the face. These features remain robust even under challenging conditions such as varying lighting or complex viewing angles. The **neck** functions as an intermediate aggregator, combining features from the **backbone** at multiple scales, from fine details like the eyes and nose to larger patterns like overall facial structure. This allows the model to accurately detect faces of various sizes, ranging from small faces in the background to larger ones in the foreground, before passing the features to the **head**. The **head** then performs the final task of prediction, generating precise bounding boxes to locate faces and determine whether the object is indeed a face, ensuring high reliability in the output.

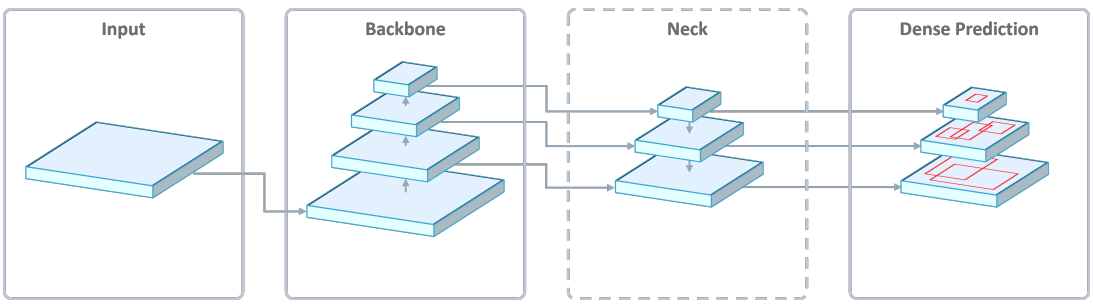

Figure 1: YOLO Face V2 flow structure

One of the most important features of YOLO Face V2 is the integration of **SEAM**, a dedicated module designed to enhance attention to critical spatial features. Positioned between the neck and the head, **SEAM** acts as a smart filter that directs the model's focus to essential facial regions such as the eyes or mouth, even when parts of the face are

occluded by elements like masks, hair, or shadows. By amplifying the recognition of the remaining visible features, **SEAM** enables YOLO Face V2 to maintain high accuracy under difficult conditions and significantly improves performance compared to models that lack such attention mechanisms.
Thanks to its well-optimized architectural design and its ability to efficiently utilize modern hardware such as graphics processing units, YOLO Face V2 achieves outstanding results in both processing speed and detection accuracy. The model can smoothly handle live video streams and deliver nearly instant outputs without compromising the quality of recognition. This makes YOLO Face V2 a highly suitable solution for real time applications, including security surveillance systems, facial recognition on mobile devices, and intelligent camera platforms where both speed and reliability are essential.

IV. Conduct the demonstration

1. Technologies used

1.1 Web

The demo system is built on a modern technological foundation, focusing on optimizing performance, maintainability, and enhancing user experience. TypeScript is chosen as the primary language instead of JavaScript due to its strict type-checking capabilities, which help reduce runtime errors and improve overall development stability. On the frontend, ReactJS plays a central role with its component-based architecture, enabling flexible UI organization, efficient code reuse, and easy feature expansion.
The development workflow is accelerated through the integration of Vite—a high-speed bundler and development server that supports instant hot reloads and significantly shortens build times. Additionally, Tailwind CSS is used to design the interface following a utility-first approach, allowing styles to be applied directly in HTML while minimizing the need for custom CSS, all without compromising consistency or visual appeal.
The synergy between these technologies forms a well-balanced architecture that ensures both performance and maintainability, meeting modern web development standards: robust, minimalist, and user-friendly.

1.2 Model

You need to clone the github repository from here and install the required libraries, which were noted in README.md.

2. User interface

2.1 Home

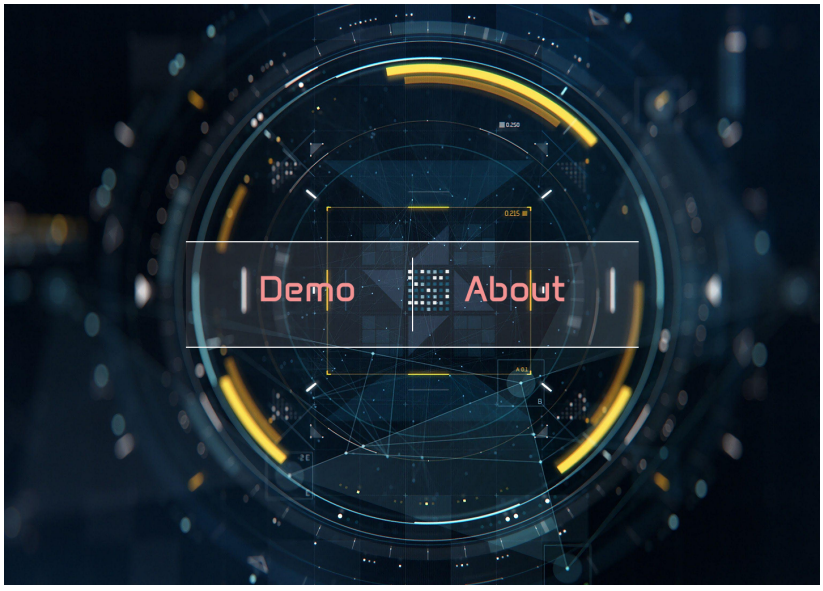

Figure 2: Figure of Home page

Once accessing the demo page, users encounter the Onboard screen, which is the first interface. Here, they can choose between experiencing the product demo or viewing detailed information about the product and the development team.

2.2 Demo

Figure 3: Figure of Demo page

After selecting the Demo option from the Onboard screen, users will be redirected to the product demo interface. Here, they can upload an image or video file, which the system will process using the YOLO FaceV2 model. The model performs face detection on the uploaded content, and the results are displayed directly on the web interface with identified faces and their corresponding confidence scores. In addition to on-screen visualization, the system allows users to download the processed image or video, enabling viewing in higher resolution or use for other purposes.

Note: Users should ensure that the uploaded files are in one of the supported formats: ".jpg", ".png", ".bmp", ".jpeg", ".gif", or ".mp4". The downloadable output files are available in ".jpg" format for images and ".mp4" for videos.

2.3 About

Figure 4: Figure of About page

This page provides detailed information about the model used in the demo, specifically YOLO FaceV2, an advanced facial recognition algorithm based on YOLO (You Only Look Once). This model is designed to detect faces quickly and accurately in real-time.

Additionally, users can learn more about the development team, including a list of members involved in researching and implementing the system. Transparency in information enhances trust and offers a clearer perspective on the people behind this project.

3. How to conduct

This section describes the steps to demonstrate the functionality of the YOLO-FaceV2 web-based face detection system.

3.1 Project setup overview

- **Frontend**: Built with **TypeScript**, **ReactJS**, and **Tailwind CSS** for styling.
- **Backend**: A **FastAPI** server handles file uploads (image/video), processes them using **YOLO-FaceV2**, and returns the detected output.
- **Model**: The YOLO-FaceV2 model is cloned from the official GitHub repository: https://github.com/Krasjet-Yu/YOLO-FaceV2

3.2 Running the backend (API server)

1. Ensure `Python`, `torch`, `opencv-python`, `FastAPI`, and all YOLO-FaceV2 dependencies are installed.
2. Place the trained model weights `best.pt` in the same directory as the backend script.
3. Start the FastAPI server using
4. The API will be hosted at: `http://127.0.0.1:8000/detect`

3.3 Running the frontend (Web interface)

1. Install Node.js and required dependencies:

   ```
   npm install
   ```

2. Start the development server:

   ```
   npm run dev
   ```

3. The frontend will be accessible at: `http://localhost:5173`

3.4 Using the web application

1. Open the web interface in your browser.
2. Click the **Upload** button to select an image (`.png`, `.jpg`, etc.) or a video file (`.mp4`).
3. The frontend sends the file to the FastAPI server via a POST request.
4. The server detects faces using YOLO-FaceV2 and returns the processed result.
5. The result is displayed directly on the web interface.

3.5 Demonstration scenarios

- **Scenario 1: Image Detection** — Upload a portrait image and show the bounding boxes and confidence scores.
- **Scenario 2: Multiple Faces** — Upload a group photo and show how the model detects multiple faces accurately.
- **Scenario 3: Video Detection** — Upload a short `.mp4` file and demonstrate the frame-by-frame detection.

3.6 Technical notes

- The API supports **CORS**, enabling seamless communication with the frontend.
- The model runs on **GPU** if available, or falls back to **CPU**.
- Unified endpoint `/detect` supports both images and videos.

3.7 Additional considerations

- Prepare sample image/video files in advance to ensure a smooth demo.
- Monitor the backend terminal for detection logs and potential errors.
- Ensure the model file `best.pt` is properly loaded before starting.

V. References

[1] Stan Z JAIN, Anil K.; LI. Handbook of face recognition. new york: springer, 2011.

[2] Ross Girshick Shaoqing Ren, Kaiming He and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks.

[3] et al YU, Ziping. Yolo-facev2: A scale and occlusion aware face detector. pattern recognition, 2024, 155: 110714.