Dynamic Recognition: Extending Faster RCNN Capabilities for Facial Identification in Media Stream

Qian Tang
Department of Electrical and Computer Engineering,
University of Toronto
qianqian.tang@mail.utoronto.ca

Abstract—Prosopagnosia, also called face blindness, is a condition where you have difficulty recognizing people's faces. To help people solve this problem, I would like to develop an advanced face detection and tracking system to analyze video actors/actresses. My approach includes preprocessing techniques like pseudo-HDR and noise reduction and employs Faster R-CNN for face identification and tracking. The dataset is diversified for robust model training, focusing on various visual scenarios. With an overall mean average precision of 87.81%, the system effectively addresses the challenges in face recognition and tracking, demonstrating high accuracy even in complex scenes. This technology holds significant potential for aiding individuals with prosopagnosia and enhancing security and media cataloging.

Index Terms—Computational Photography, Image Sharpen, Brightness Balance, Image Denoise, Deep Learning

1 Introduction

In the evolving landscape of digital media and security, facial recognition technology has become pivotal. However, challenges persist in dynamic, real-world scenarios, notably for individuals with prosopagnosia (face blindness). This paper addresses these challenges through an advanced face detection and tracking system, integrating Faster R-CNN with sophisticated preprocessing techniques like pseudo-HDR and noise reduction. My approach aims to enhance accuracy and adaptability in diverse environments. The key contributions of this work include significant improvements in facial recognition for aiding individuals with face blindness and applications in security and media cataloging, leveraging the advancements in computational photography and deep learning.

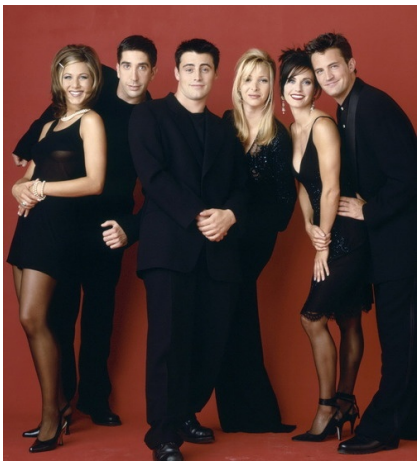

Fig. 1. Aim to recognize faces from pictures above

2 Related Work

Previous work has laid a strong foundation for my project. They have shown a significant step in face detection but still remain primarily as a basic detection model that lacks the ability to do an identification or to solve complicated scenarios. But anyway, each of these methods brings us closer and also highlights the gap my work aims to fill. Faster R-CNN framework: [1] In 2017, H. Jiang and E. Learned-Miller investigated the application of the Faster R-CNN model, known for its impressive results in object detection, to the realm of face detection. This approach marks a significant shift from traditional methods predominantly based on the R-CNN framework, which were limited in both accuracy and processing speed. By training the Faster R-CNN on the extensive WIDER face dataset, they not only achieve state-of-the-art results on this dataset but also demonstrate marked improvements on other key face detection benchmarks, such as FDDB and IJB-A.

Face Recognition: A CNN architecture [2] includes three convolution layers, two pooling layers, two fully connected layers, and a Softmax regression layer, all trained using the stochastic gradient descent algorithm. It's adept at autonomously extracting and classifying facial features, with the Dropout method integrated to mitigate over-fitting issues and the Caffe framework employed during training and testing phases. Remarkably, this method achieves impressive face recognition rates of 99.82% on the ORL face database and 99.78% on the AR face database.

Face Mesh and 3D reconstruction: A novel model [3] addresses the limitations of existing face recognition systems, particularly under varying conditions such as illumination, background, and non-frontal images of individuals across

different ages and races. They use Face Mesh technology to reconstruct the 3D face after face detection. Then, use a deep Deep Neural Network to recognize the 3D face. The model demonstrates proficiency in accurately identifying and detecting faces, even in challenging scenarios. The Face Mesh enables the model to operate effectively in diverse conditions and handle images where faces are not directly front

3 Proposed Method

3.1 Data Processing

3.1.1 DSFD

For my project, I handpicked data from Season 6 of 'Friends'. A total of 1400+ pictures is collected for further training. I captured multiple videos and an algorithm called DSFD [4] is used to accurately locate faces in these video frames. It has four parts:
1. Load Image: The first step is to load the image in which faces are to be detected.
2. First Shot (Detection): In this stage, the model performs the initial face detection. It is designed to identify faces in various conditions and scales, making it effective in detecting faces that might be missed by simpler models.
3. Second Shot (Refinement): The detection from the first shot is refined in this stage. It aims to improve the accuracy of the detection by reducing false positives and adjusting the detected face regions for better precision.
4. Mark Faces: Once the faces are detected and refined, the model marks them, typically by drawing bounding boxes around the faces in the image.

3.1.2 Automation Python Script

I wrote a specialized Python script to annotate each frame and label them, making sure I could keep track of each frame's actor and actress accurately.

```
def save_face_clips(frame, bboxes):
    global img_counter
    global frame_counter
    for bbox in bboxes:
        x0, y0, x1, y1 = [int(_) for _ in bbox]
        # Crop and save face image
        frame_copy = frame.copy()
        try:
            x0 = x0-20
            y0 = y0-20
            x1 = x1+20
            y1 = y1+20
            face_img = frame_copy[y0:y1, x0:x1]
            img_path = os.path.join(save_dir, f"
                face_{img_counter}_frame_{
                frame_counter}@3_x0_{x0}_y0_{
                y0}_x1_{x1}_y1_{y1}.jpg")
            cv2.imwrite(img_path, face_img)
            img_counter += 1
            cv2.rectangle(frame, (x0, y0), (x1,
                y1), (0, 0, 255), 2)
        except:
            pass

while True:
    frame_counter += 1
    ret, frame = vs.read()
    if not ret:
        break
    dets = detector.detect(frame[:, :, ::-1])[:,
        :4]
    save_face_clips(frame, dets)
    key = cv2.waitKey(1)
    if key == ord('q'):
        break
    if writer is None and args["output"] is not
        None:
        fourcc = cv2.VideoWriter_fourcc(*"MJPG"
            )
        writer = cv2.VideoWriter(args["output"],
            fourcc, 20, (frame.shape[1], frame.
            shape[0]), True)
    if writer is not None:
        writer.write(frame)
```

3.1.3 Distribution of Dataset

To make sure my model learns effectively, I've organized my dataset into different categories. This includes 90% of the images where the subjects have their eyes open and 10% where they're closed. I also made sure to balance the types of images I used – half of them are front-facing shots, and the other half are side profiles. This mix is meant to help my model perform better, no matter what kind of image it's analyzing.

3.2 Model

For this project, I will evaluate the performance of face detection and recognition of three types of Faster Faster R-CNN. One is the base Faster R-CNN model with backbone ResNet 50 [5]. The second model is using the ResNet 152. In the third model, I apply some pre-processing to the image set before passing into the Faster R-CNN model with backbone ResNet 152 and use a validation set to train the hyperparameters. Below is the final configuration of my model.

```
Loaded IMAGENET1K_V1 pre-trained weights
    of Torchvision ResNet50 backbone
Training Parameters
--------------------
 Initial weights   : IMAGENET1K_V1
Dataset            : VOCdevkit/VOC2007
Training split     : trainval
Evaluation split   : test
Backbone           : resnet50
Epochs             : 3
Learning rate      : 0.001000
Momentum           : 0.900000
Weight decay       : 0.000500
Dropout            : 0.000000
Augmentation       : enabled
Edge proposals     : included
CSV log            : none
Checkpoints        : disabled
```

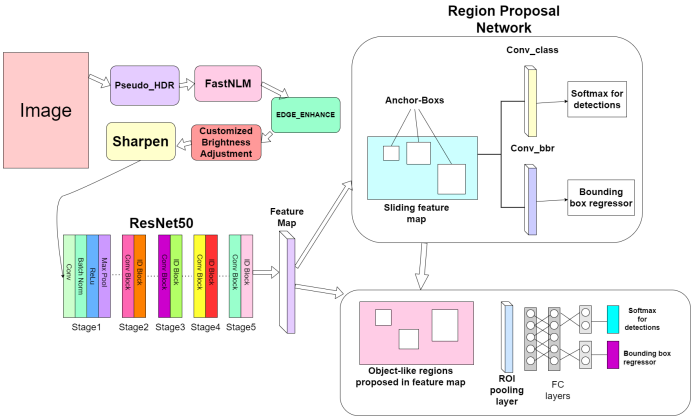

Fig. 2. Modified Fast RCNN model

3.2.1 *Pre-processing*

My model includes a pre-processing part before doing model training. It can be divided into five parts:
CLAHE, FastNLM, Edge Enhancement, Brightness Balance, Image Sharpen.
I will show you a sample image and its change after each process.

Fig. 3. Sample raw image to be processed

3.2.1.1 CLAHE: My project's initial phase involves enhancing the images' dynamic range using Contrast Limited Adaptive Histogram Equalization (CLAHE). This technique is a variant of adaptive histogram equalization, specifically designed to prevent over-amplification of noise in relatively homogeneous regions of an image. CLAHE improves the contrast of the images, thereby extending the range between the lightest and darkest areas. This enhancement is crucial for revealing more details in the highlights and shadows, ensuring that facial features are visible under varying lighting conditions.

Fig. 4. After Contrast Limited Adaptive Histogram Equalization

3.2.1.2 Non-Local Mean: Following the enhancement of the dynamic range, I employed the Fast Non-Local Means algorithm for noise reduction.

The essence of the Fast Non-Local Means algorithm lies in its unique approach to averaging pixel values. Unlike local denoising methods that only consider nearby pixels, Non-Local Means [6] look at the entire image to find similar patches. The denoising process is based on the assumption that patches of the image repeat themselves within the image. The pixel's value in the denoised image is computed as a weighted average of all pixels in the image, where the weights depend on the similarity of pixel neighbourhoods.

$$I_{Denoised}(i) = \frac{1}{Z(i)} \sum_{j \in i} W(i,j) I_{noisy}(j)$$

where

- $I_{Denoised}(i)$ is the intensity of pixel i in the denoised image
- $I_{noisy}(j)$ is the intensity of pixel j in the noisy image
- $w(i,j)$ is the weight reflecting the similarity between the neighbourhoods of pixel i and pixel j. It is typically computed using the Gaussian function of the Euclidean distance between the neighbourhoods.
- $Z(i)$ is a normalization factor defined as $\sum_{j \in i} w(i,j)$, ensuring the weights sum up to one.

However, Non-Local Means has a time complexity of $O(m^2, n^2)$. This causes a long delay during training and testing. Therefore I decided to run this NLM in GPU mode. This makes a huge improvement in efficiency. This advanced denoising method effectively reduces image noise while preserving important details, which is crucial for accurate face detection and recognition. This step is particularly important for low-light images where noise is more prevalent.

Fig. 5. After NLM

3.2.1.3 Edge Enhancement: The EDGE ENHANCE technique was then applied to the images. This process makes the edges within the images more pronounced, which aids in the distinction of facial features from the background. Edge enhancement is crucial for highlighting subtle features in the face, which are critical for accurate recognition.

Fig. 6. After Edge Enhancement

3.2.1.4 Brightness balance: Subsequently, each image underwent a brightness balancing process with Customized Brightness Adjustment using the YCbCr approach. Convert the image to YCbCr format set the Y channel to 80 and then convert back. This step ensures uniform illumination across all images, addressing any issues of underexposure or overexposure. By normalizing the brightness, I improve the consistency of the input data, which is essential for the effectiveness of subsequent processing steps.

Fig. 7. After YCbCr

3.2.1.5 Sharpen: After that, I implemented an image-sharpening step. This was necessary to counteract any softening effects that might have been introduced in the earlier stages of processing. The sharpening process fine-tunes the image details, further improving the clarity of facial features.

Fig. 8. After Sharpening

3.2.2 Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Networks) is a state-of-the-art object detection framework that is widely used due to its efficiency and accuracy [7]. It is an extension of the original R-CNN and Fast R-CNN, further improving the speed and detection performance. This model is particularly effective in tasks like object detection and recognition, where precise localization and identification are crucial.

1. **Convolution Layer**:
The input image is first processed through several convolutional layers. These layers, borrowed from a pre-trained model like ResNet or VGG16 [8], are used to extract a feature map of the input image. These convolutional layers capture various aspects of the image, such as edges, textures, and other relevant features.

2. **Region Proposal Network (RPN)**:

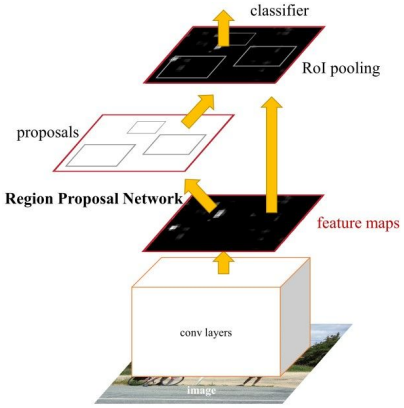

Fig. 9. RPN model

Faster R-CNN introduces a Region Proposal Network (RPN) that directly works on the feature maps from the convolutional layers to propose candidate object bounding boxes. The RPN uses a sliding window approach on the feature map and for each location, it generates multiple region proposals (bounding boxes) of various scales and aspect ratios. Each region proposal is scored based on how likely it is to contain an object. This step significantly reduces the number of candidate regions, improving efficiency.

3. **RoI Pooling**:
The Region of Interest (RoI) Pooling layer takes the feature map and the proposed regions from the RPN. It then extracts fixed-size feature vectors from each proposed region. This process involves resizing the features from each region proposal to a fixed size so that they can be processed by a fully connected layer, regardless of the region's original size.

4. **Classification and Bounding Box Regression**:
The extracted fixed-size feature vectors are then fed into a set of fully connected layers. These layers serve two purposes: they classify the region into different categories (such as face or no face in the context of face detection) and refine the bounding box coordinates of the region.

4 Experimental Results

For six performers in the "Friends", I get an overall average result of 87.81% accuracy.

```
Average Precisions
------------------
Jennifer Aniston: 100.0%
Courteney Cox   : 92.0%
Matthew Perry   : 90.6%
David Schwimmer : 90.1%
Matt LeBlanc    : 90.0%
lisa Kudrow     : 64.1%
------------------
Mean Average Precision = 87.81%
```

Fig. 10. Raw Result

My proposed model consistently outperforms the base models across all individuals. Therefore the pre-processing and hyperparameter tuning have a positive impact on the model's ability to detect and recognize faces. The base model with ResNet152 generally outperforms the one with ResNet50, because ResNet152 has deeper layers than ResNet50.

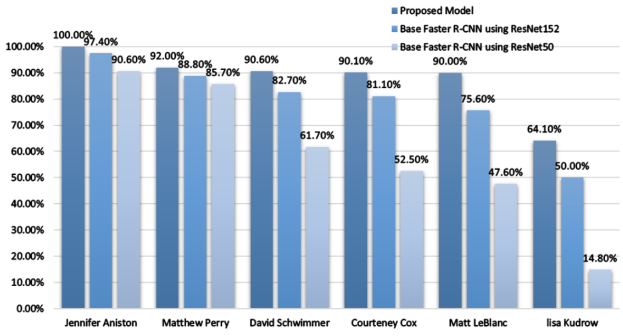

Fig. 11. Result Comparison

My model's performance exhibits significant variability across different individuals. This inconsistency may stem from a range of factors such as the diversity and volume of the training data, the uniqueness of each individual's facial features, and the variations in lighting and pose in the images. For instance, Jennifer Aniston's dataset, which includes a variety of poses, is more comprehensive.

The significantly lower recognition rates for Lisa Kudrow across all models can be attributed to the discrepancy between the training and test datasets. Specifically, episodes 3 and 4, which were used for training, feature images of Lisa Kudrow with her hair down. In contrast, the test set, derived from episode 2, predominantly contains images where her hair is tied up. This variation in hairstyle introduces a substantial visual difference that the Faster R-CNN model fails to generalize, leading to reduced softmax probabilities that fall below the necessary threshold for accurate detection. As a result, numerous instances of Lisa Kudrow in the test set are either not detected or incorrectly classified. This highlights the model's sensitivity to changes in appearance and underscores the importance of incorporating a diverse range of training data to improve the robustness of the recognition system. Two examples are shown below: My model still shows good accuracy when encountering complicated cases which contain three or four actors/actresses in the same frame.

Fig. 12. Complicated Case 3 actors

When three actresses dress the same, having the same hairstyle and face make-up, my model identifies each of the actresses correctly.

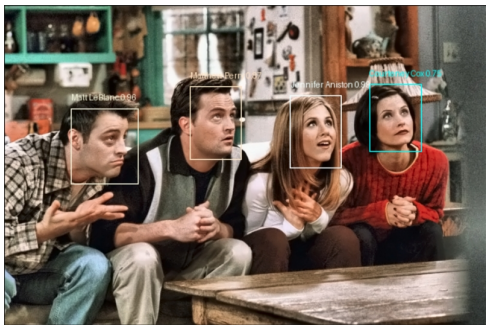

Fig. 13. Complicated Case 4 actors

I also include a part of a short video in the code folder for you to get a better intuitive understanding of the tracking process.

5 Limitation

My model exhibits notable accuracy in complex environments, effectively identifying multiple actors within a single frame, particularly in scenes with three to four individuals. However, it encounters difficulties in scenarios where actors display highly exaggerated expressions or when dealing with small facial features, resulting in a decrease in performance. These challenges underscore the need for further refinement to enhance the model's capability in more diverse and challenging conditions.

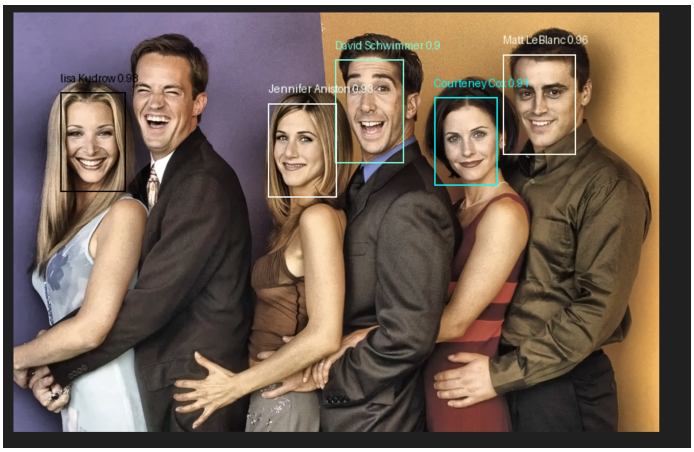

Fig. 14. Failed in detection of exaggerated expressions

Fig. 15. Failed in detection of small face

The current limitations of my model predominantly stem from a restricted range of expressions in my training dataset. To address this, I am diligently working to broaden the dataset's scope by incorporating a more diverse array of facial expressions and configurations. My efforts are focused on enhancing the robustness of the model, enabling it to consistently achieve high accuracy across an extensive range of facial expressions and complex group interactions. By introducing scenarios that more closely resemble real-life situations, I aim to refine the model's capability to accurately detect and recognize faces across various media streams.

6 Conclusion

My project has successfully implemented the Faster R-CNN framework, complemented by advanced image preprocessing and meticulous hyperparameter adjustments, leading to significant improvements in accuracy. However, I encountered challenges when dealing with substantial variations in appearances and expressions, particularly in test scenarios divergent from my training conditions. Moving forward, my emphasis will be on enriching my training dataset with a wider spectrum of real-life images. This strategic enhancement is aimed at surmounting the current limitations, thereby augmenting the model's resilience and adaptability to diverse real-world situations.

References

[1] H. Jiang and E. Learned-Miller, "Face detection with the faster r-cnn," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, Washington, DC, USA, 2017, pp. 650–657.

[2] K. Yan, S. Huang, Y. Song, W. Liu, and N. Fan, "Face recognition based on convolution neural network," in *2017 36th Chinese Control Conference (CCC)*, Dalian, China, 2017, pp. 4077–4081.

[3] S. Hangaragi, T. Singh, and N. N, "Face detection and recognition using face mesh and deep neural network," *Procedia Computer Science*, vol. 218, pp. 741–749, 2023.

[4] J. Li *et al.*, "Dsfd: Dual shot face detector," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 5055–5064.

[5] S. Mukherjee, "The annotated resnet-50," https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758, aug 2022.

[6] J.-M. M. Antoni Buades, Bartomeu Coll, "Title of the article," *Image Processing On Line*, Sep. 2011. [Online]. Available: https://doi.org/10.5201/ipol.2011.bcm_nlm

[7] "Faster r-cnn explained: Object detection," https://blog.paperspace.com/faster-r-cnn-explained-object-detection/.

[8] K. Le, "A quick overview of resnet models," https://medium.com/mlearning-ai/a-quick-overview-of-resnet-models-f8ed277ae81e, mar 2021.