WIX3001 – SOFT COMPUTING

INDIVIDUAL ASSIGNMENT

FACE DETECTION AND ANONYMIZATION

AMIR FIRDAUS BIN ABDUL HADI

17204620/2

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2022

Table of Contents

1 Abstract

Face recognition techniques improves at a steady rate with the recent deep learning technique developed and the access to large training dataset made available in the Internet. This project aims to be able to detect faces in images and apply censorship as an anonymization feature for privacy using deep learning. The deep learning that is used is Multi-task Cascaded Convolutional Neural Network (MTCNN), which contains three sub-networks together to learn to recognize human faces after several stages of decomposition and filtering.

2 Introduction

Fogging, also known as blurring, is used for censorship or privacy. A visual area of a picture or video is blurred to obscure it from sight. This form of editing also appears in television programs where an individual's face may not be shown due to legal or privacy concerns. Another example is Google Maps Street View that blurred out any faces, vehicle plates and others privacy related area to ensure that they are not responsible for any legal or privacy issues that can occur. With the era of social media, some of us do not realize the important of anonymity of others while we are taking pictures and capturing videos in public. Hence, this project approach to achieve anonymization in images by detecting faces using deep learning model MTCNN.

2.1 Objectives

Objective 1: To be able to detect faces in images.

Objective 2: To apply blur to the detected faces to anonymize it.

2.2 Dataset

For the MTCNN model, the creator used Face Detection Data Set and Benchmark (FDDB) and Annotated Facial Landmarks in the Wild (AFLW) for their training and testing. But for this project, as the model is already pre-trained, a sample of 20 images with its corresponding labelled key points will be used to evaluate the MTCNN model to find the success detection rate.

3 Literature Review

Face detection are essential to many face applications, such as face recognition and facial expression analysis. However, the large visual variations of faces, such as occlusions, large pose variations and extreme lightings, impose great challenges for these tasks in real world applications. The cascade face detector proposed by Viola and Jones (Viola and Jones 2004) utilizes Haar-Like features and AdaBoost to train cascaded classifiers, which achieves good performance with real-time efficiency.

However, quite a few works (Yang et al. 2014; Pham et al. 2010) indicate that this kind of detector may degrade significantly in real-world applications with larger visual variations of human faces even with more advanced features and classifiers. Besides the cascade structure (Zhu and Ramanan 2012), introduce deformable part models (DPM) for face detection and achieve remarkable performance. However, they are computationally expensive and may usually require expensive annotation in the training stage.

Recently, convolutional neural networks (CNNs) achieve remarkable progresses in a variety of computer vision tasks, such as image classification and face recognition (Sun, Wang, and Tang 2014). Inspired by the significant successes of deep learning methods in computer vision tasks, several studies utilize deep CNNs for face detection. Yang et al. (Yang et al. 2016) train deep convolution neural networks for facial attribute recognition to obtain high response in face regions which further yield candidate windows of faces. However, due to its complex CNN structure, this approach is time costly in practice. Li et al. (Li et al. 2015) use cascaded CNNs for face detection, but it requires bounding box calibration from face detection with extra computational expense and ignores the inherent correlation between facial landmarks localization and bounding box regression.

Using deep neural networks to learn effective feature representations has become popular in face recognition (Sun, Wang, and Tang 2013). With better deep network architectures and supervisory methods, face recognition accuracy has been boosted rapidly in recent years.

4 Methodology

4.1 MTCNN framework

The method that is used is called Multi-task Cascaded Convolutional Networks. Different from other algorithms, MTCNN is cascading three CNN with different structures together for face detection. Figure X is the pipeline of MTCNN, and figure X is the detailed structure of MTCNN. Before we put the testing image into classifier, we need to resize the image in different scales, and stack it into an image pyramid. By doing these steps, we can generate the same face in different scales, which increases the ability of the network. After that, a sliding window will be applied to the pyramid, and break the image into regions, which are the inputs to the network.

4.1.1 Stage 1 (P-Net):

Stage-1 is called P-Net, which references to Proposal network. It is a shallow network which will roughly decide which region contains faces. For each proposed bounding box, there will be three different classifications applied to it, which are face classification, bounding box regression and facial landmark localization. I will introduce these three algorithms in the later section. The output of P-Net are some proposed boxing boxes and their classification scores of faces. None maximum suppression will be applied in order to clean the bounding boxes that are overlapping with each other. Those remaining boxes will be resized to 24x24x3, and used as the input to the next net.

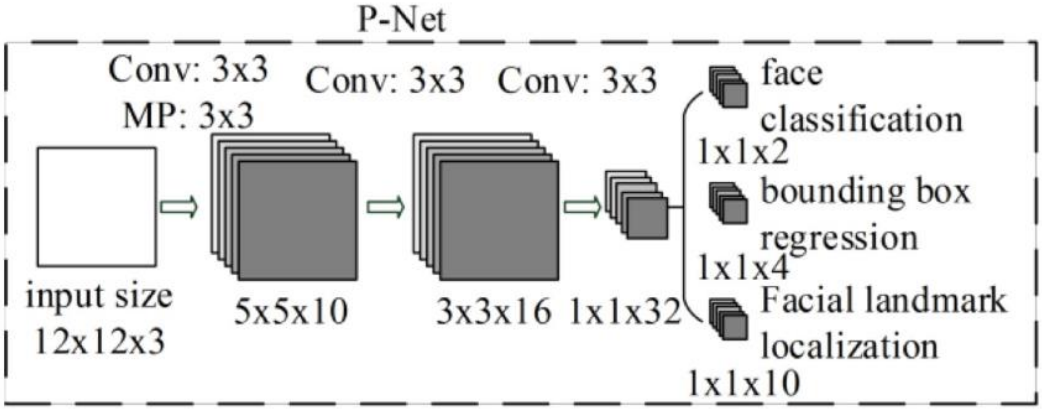

Figure 1: P-Net of MTCNN

4.1.2 Stage 2 (R-Net):

Stage-2 is called R-Net, which references to Refine network. This Net has convolution with larger kernel and a fully connected layer, which is more powerful than the previous network. The goal of this net is to refine the results from previous net. The bounding boxes which have low face classification scores will be discarded. Again, for the regions with high classification

scores, bounding box regression and facial landmark localization will be applied. The output of this net is still bounding boxes and their classification scores. Those boxes will be resized to 48x48x3 to be used as the input to the next net.

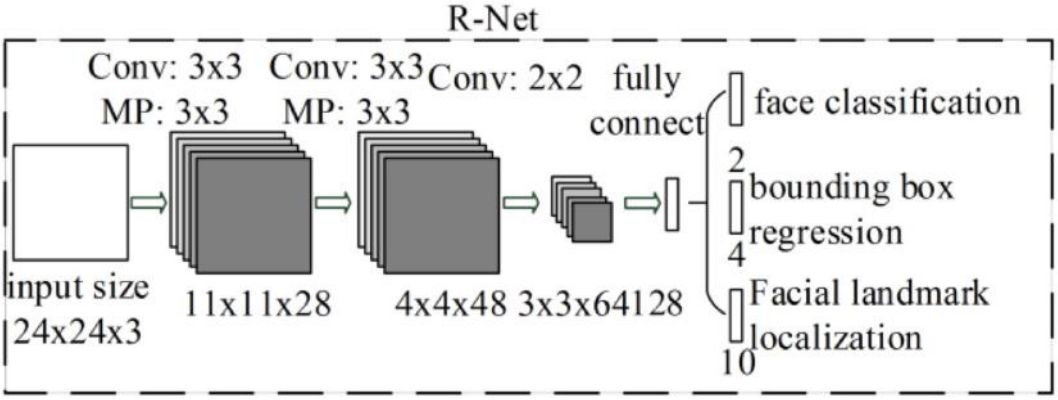

Figure 2: R-Net of MTCNN

4.1.3 Stage 3 (O-Net):

Stage-3 is called O-Net, which references to Output Net. This network is deeper with larger convolution kernel comparing with previous nets. This powerful network will make the final decision about where the face is and what the size of bounding box should be.

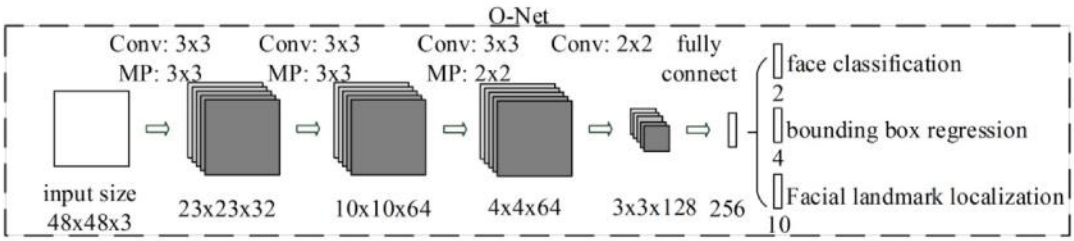

Figure 3: O-Net of MTCNN

4.1.4 Classification and Regression:

At the end of three stages, there are the computation of face classification, bounding box regression and facial landmark localization. Their loss functions are different, I will introduce them one by one.

4.1.5 Face Classification

While doing face classification, cross entropy is used as the Loss function.

$$L_i^{det} = -(y_i^{det} log(p_i) + (1 - y_i^{det})(1 - log(p_i))) \quad (1)$$

where p_i is the probability produced by the network. $y_i^{det} \subset 0, 1$ is the ground-truth label of the faces.

4.1.6 Bounding Box Regression

For the bounding box regression, Euler distance (L2 loss) is used as the loss function.

$$L_i^{box} = ||\hat{y}_i^{box} - y_i^{box}||_2^2 \quad (2)$$

where $\hat{y}_i^{box}$ is the regression target generated by the network and y_i^{box} is the ground-truth location.

4.1.7 Facial Landmark Localization

L2 loss is also used as the loss function for this part. There are five landmarks: left eye, right eye, nose, left mouth corner and right mouth corner.

$$L_i^{landmark} = ||\hat{y}_i^{landmark} - y_i^{landmark}||_2^2 \quad (3)$$

where $\hat{y}_i^{landmark}$ is the location of facial landmarks generated by the network and $y_i^{landmark}$ is the ground-truth location.

4.2 Project Flowchart

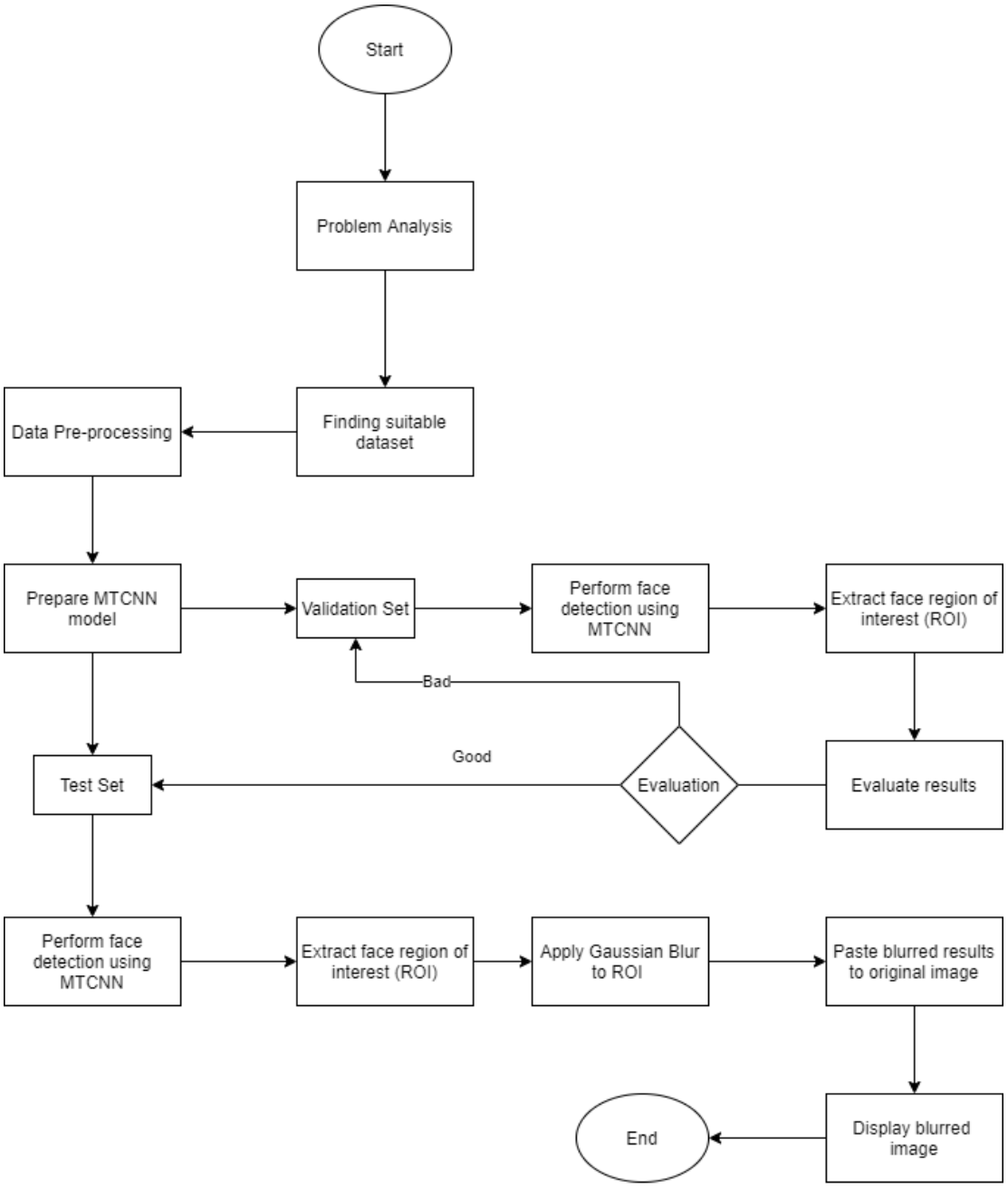

Figure 4: Project Workflow

As we are using a pre-trained model, this project only needs to prepare the dataset to put into the model and evaluate it. The main flow of this project is

1. Performing face detection using the model
2. Extract face region of interest (ROI)
3. Apply blur to ROI and
4. Paste blurred ROI to original image.

4.3 Software and Tools

4.3.1 Python

	Libraries	Description
1	**MTCNN**	Face detecting model. Implementation of the MTCNN face detector for Keras in Python3.4+. It is written from scratch, using as a reference the implementation of MTCNN from David Sandberg (FaceNet's MTCNN) in Facenet.
2	**Matplotlib**	Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. **Pyplot:** a plotting library used for 2D graphics **Patches:** A function to draw a patch such as rectangle around the bounding box.
3	**Numpy**	To perform a wide variety of mathematical operations on arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
4	**Pandas**	Pandas is a software library written for the Python programming language for data manipulation and analysis. Used for data processing and preparation.
5	**Pillow**	Contains all the basic image processing functionality. Allows image resizing, rotation and transformation. support for opening, manipulating, and saving many different images file formats. Used for modifying image to blurred image.

Table 1: Python Libraries Used in the project and its description

4.3.2 Google Collaboratory Notebook

Google Colab/Colaboratory is a free, cloud-based environment similar to Jupyter's network environment. Colab notebooks are Jupyter notebooks that run in the cloud and are highly integrated with Google Drive, making them easy to set up, access, and share. It does not require a setup and the notebooks that you create can be simultaneously edited by others. Colab supports many popular machine learning libraries which can be easily loaded in the notebook. Functions of Google Colab:

- Write and execute code in Python
- Document code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

5 Experiment and Results

5.1 Model Evaluation using Sample Images

The MTCNN model is a pre-trained model. So, there is no need to train and test large dataset for the project. However, we can evaluate the model using small sample of 20 images with labeled key points. The key points are the coordinates of left and right eyes (Table 2). If the predicted key points from the model is within 30 pixels of labelled key points, then it will be detected as successful. Based on table 3, the result of the evaluation is the model manage to successfully detect 18 out of 20 images which is 90% success rate.

Figure 5: 001.jpg from 20 sample images to be evaluated

	left_eye_x	left_eye_y	right_eye_x	right_eye_y
1	280	161	348	160
2	384	270	495	266
3	290	326	429	328
4	307	257	427	255

Table 2: Corresponding key points coordinates of sample images

Image	Success	Image	Success
001	1	**011**	0
002	1	**012**	1
003	1	**013**	1
004	1	**014**	0
005	1	**015**	1
006	1	**016**	1
007	1	**017**	1
008	1	**018**	1
009	1	**019**	1
010	1	**020**	1

Table 3: Evaluation results of 90% detection rate from sample images

5.2 Extracting Region of Interest (ROI)

Now we have determined the model works well, we can start detecting region of interest of images and then apply blur to it. Face detector will give the bounding box (x, y)-coordinates of a face in an image. These coordinates represent: starting x (lower left), starting y (lower left), width and height of a given bounding box. Figure 6 below shows the result of extracting ROI of ‘test1.jpg’ image and draw bounding box to it.

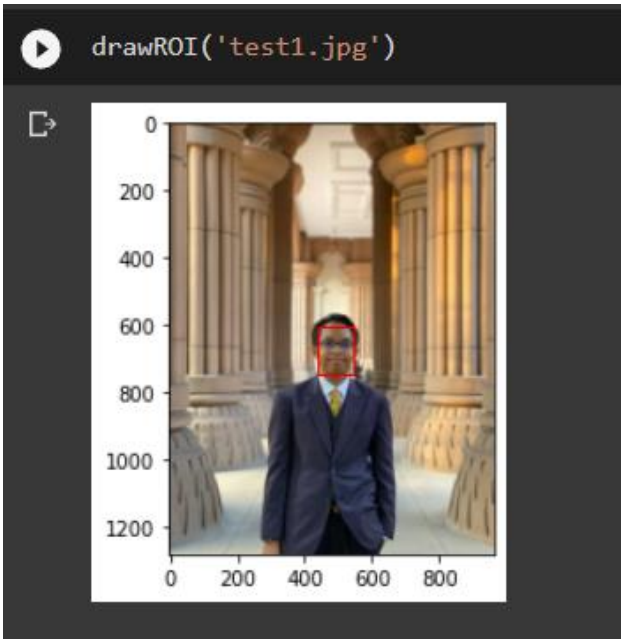

Figure 6: Red bounding box on detected ROI

5.3 Anonymize the Image based on ROI Bounding Box

For the blurring of the image, Gaussian blur is applied to the ROI. First, we cropped out the ROI and then apply the blur to the cropped image. Lastly, we paste back the blurred cropped image to original image. Low-level radius blur (r = 3) and high-level radius blur (r = 30) is experimented. Figure 7 (left) used low level blur while figure 7 (right) used high-level. From the result, low level blur still has the transparency and not much anonymity produced. Hence, high-level blur will be used for testing images.

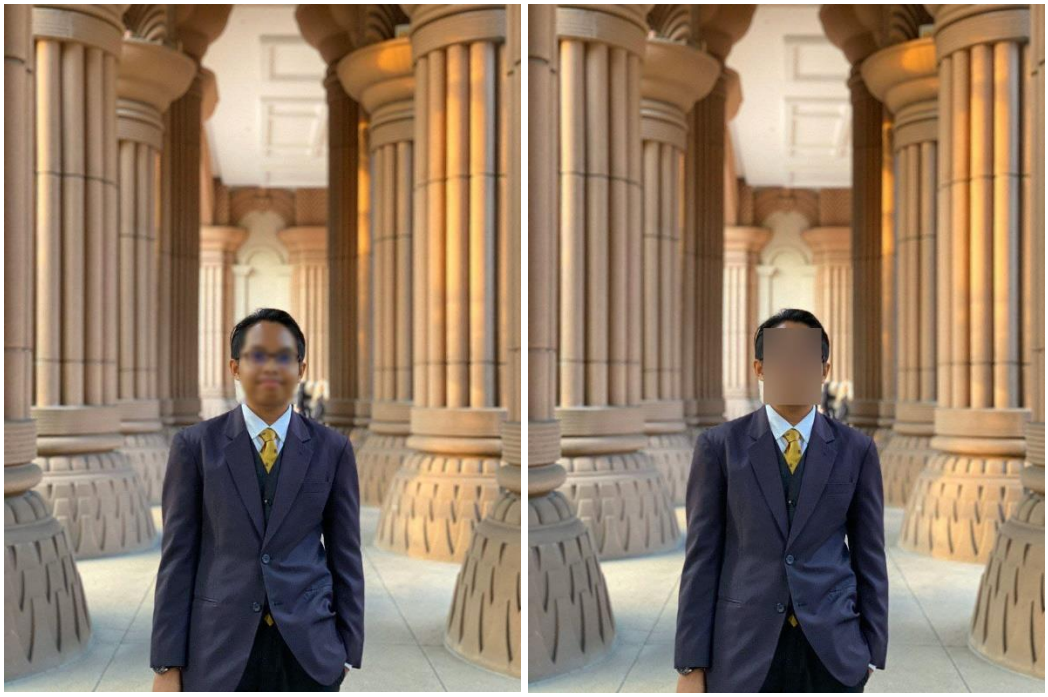

Figure 7: Application of low-level blur (left) and high-level blur (right)

5.4 Testing the Model with High-Level Blur

Testing Image with Frontal Pose:

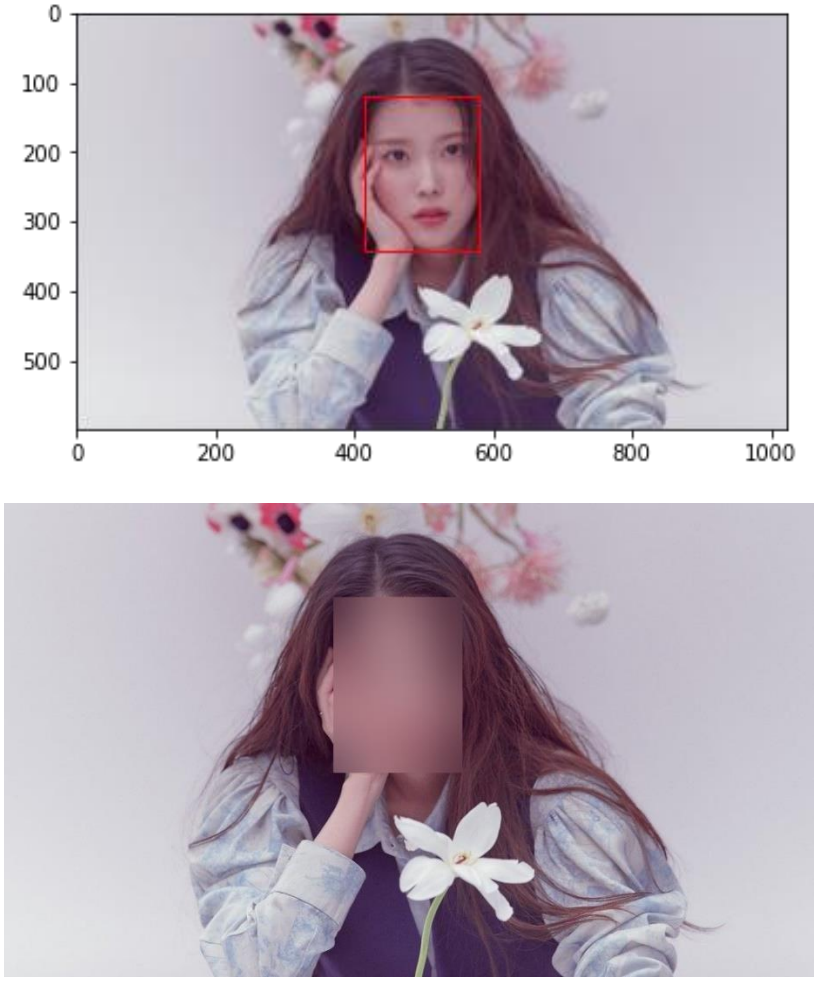

Figure 8: ROI and blurred face using frontal pose

Testing Image with Slightly Angled Pose:

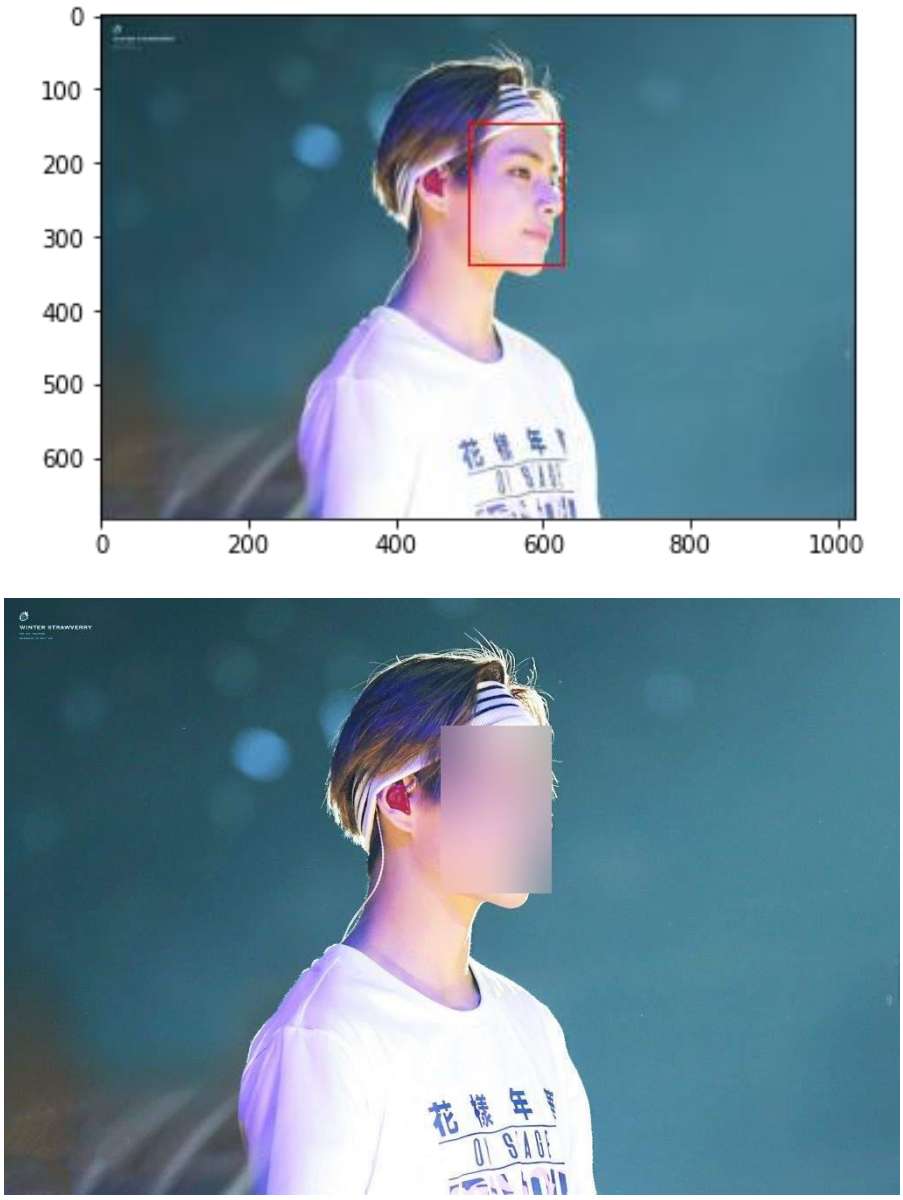

Figure 9: ROI and blurred face using slightly angled pose

Testing Image with Multiple Faces:

Figure 10: ROI and blurred face using image with multiple faces

6 Conclusion

By using deep learning MTCNN model that uses 3 cascaded convolutional neural networks; P-Net, R-Net and O-Net, this project is able to successfully detect faces in images. The strength of the model is that the detected images covered frontal pose, slightly angled pose and multiple faces. And by being able to detect the ROI, this project also achieves to anonymize images by blurring the faces ROI and modify the original image.

6.1 Limitation of Study

Although the project managed to achieve the objectives, there are some limitations to this project:

- The code for detecting ROI is only viable for JPG/JPEG images. PNG images are unable to be read and needed to converted to JPG/JPED images first.
- The model cannot detect face with completely showing side profile only but works with slightly angled pose.

6.2 Future Works

For future works, we can try to make algorithm for detecting ROI with any images format. Because this project did not have own interface to receive image, creating an UI interface to anonymize face can be considered. Other than that, due to this is pre-trained model, it is limited to its functionality. So, for future works we can create and train own deep learning model to detect faces using larger dataset with labeled key points. This self-train model might can detect images with side profile. And to note is to have extra GPU or using software that provide GPU for processing power for self-train deep learning model as it can be time consuming.

7 References

Website sources:

Adamczyk, J. (2021, February 17). Robust face detection with MTCNN. Retrieved from https://towardsdatascience.com/robust-face-detection-with-mtcnn-400fa81adc2e

Blur and anonymize faces with OpenCV and Python. (2021, April 17). Retrieved from https://pyimagesearch.com/2020/04/06/blur-and-anonymize-faces-with-opencv-and-python/

Gradilla, R. (2020, July 27). Multi-task cascaded Convolutional networks (MTCNN) for face detection and facial landmark alignment. Retrieved from https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923

Jason Brownlee. (2020, August 23). How to perform face detection with deep learning. Retrieved from https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/

Mtcnn/mtcnn at master · ipazc/mtcnn. (n.d.). Retrieved from https://github.com/ipazc/mtcnn/tree/master/mtcnn

Wider face: A face detection benchmark. (n.d.). Retrieved from https://shuoyang1213.me/WIDERFACE/

LFW face database : Main. (2018, January 9). Retrieved from https://vis-www.cs.umass.edu/lfw/

Journal sources:

K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10):1499–1503, Oct 2016.

P. Viola and M. J. Jones, “Robust real-time face detection. International journal of computer vision,” vol. 57, no. 2, pp. 137-154, 2004

M. T. Pham, Y. Gao, V. D. D. Hoang, and T. J. Cham, “Fast polygonal integration and its application in extending haar-like features to improve object detection,” in IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 942-949.

Q. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan, “Fast human detection using a cascade of histograms of oriented gradients,” in IEEE Computer Conference on Computer Vision and Pattern Recognition, 2006, pp. 1491-1498.

Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representa-tion by joint identification-verification,” in Advances in Neural Infor-mation Processing Systems, 2014, pp. 1988-1996.

J. Yan, Z. Lei, L. Wen, and S. Li, “The fastest deformable part model for object detection,” in IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2497-2504

H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection,” in IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5325-5334.

X. Zhu, and D. Ramanan, “Face detection, pose estimation, and landmark localization in the wild,” in IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2879-2886.

8 Appendix

Link to Code (Google Collaboratory Notebook):

https://colab.research.google.com/drive/11HM9Pr0V7A2MhhudS8mc7davmSzWJCLV?usp=sharing

Link to Google Drive (image sample data sets, csv file, code):

https://drive.google.com/drive/folders/1vQkEgZqBJKfUc-PFdZhe_CkOEeaoZQ7s?usp=sharing