FACE MASK DETECTION SYSTEM USING DEEP LEARNING MODEL

CHAI YONG XIN

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

2022

FACE MASK DETECTION SYSTEM USING DEEP LEARNING MODEL

CHAI YONG XIN

SUBMITTED TO THE FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY UNIVERSITI MALAYA, IN PARTIAL FULFLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF DATA SCIENCE

2022

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Chai Yong Xin** (I.C/Passport No: **950711-08-6218**)

Matric No: **17219908**

Name of Degree: **MASTER OF DATA SCIENCE**

Title of Research Report: **Face Mask Detection System Using Deep Learning**

Field of Study: **Computer Vision & Machine Learning**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature [illegible] Date: 24 June 2022

Subscribed and solemnly declared before,

Witness's Signature [illegible] Date: 24/6/2022

DR ERMA RAHAYU MOHD FAIZAL
PENSYARAH KANAN
JABATAN KEPINTARAN BUATAN
FAKULTI SAINS KOMPUTER DAN TEKNOLOGI MAKLUMAT
UNIVERSITI MALAYA
50603 KUALA LUMPUR

Name: Dr. Erma Rahayu Binti Mohd Faizal Abdullah

Designation: Supervisor

FACE MASK DETECTION SYSTEM USING DEEP LEARNING

ABSTRACT

Coronavirus disease (COVID-19) is an infectious disease that can cause respiratory illness. The first reported case was in 2019, and until now, in 2022, the coronavirus still exists. The World Health Organization (WHO) encourages us to always wear masks in public places, as wearing them is our last layer of protection. Many mask detection systems have been deployed recently, but the detectors failed to recognize their faces because the faces were too blurry. Therefore, super-resolution (SR) methods are used to improve the resolution of images. This study uses Enhanced Deep Super-Resolution (EDSR) and Super-Resolution Generative Adversarial Networks (SRGAN) to reconstruct images. EDSR won first place in the NTIRE2017 SR challenge because it is an enhancement to ResNet. EDSR removes the batch normalization layer, which not only improves super-resolution performance but also reduces GPU memory usage by 40%. While SRGAN can reconstruct images without compromise it. Reconstruct the low-resolution dataset to a high-resolution dataset using SRGAN. The low and high-resolution datasets will be trained by MobileNetV2 which prioritizes speed, and EfficientNet-B0 and ResNet50 which prioritize high performance. The model will be measured using a confusion matrix. MobileNetV2 + SR is the best model with 100% specificity and accuracy, 98% accuracy and F1 score, and 97% sensitivity.

Keywords: Image Recognition, SRGAN, ResNet50, MobileNetV2, EfficientNet-B0

PENGESANAN PELITUP MUKA MENGGUNAKAN DEEP LEARNING

ABSTRAK

Penyakit koronavirus (COVID-19) ialah penyakit berjangkit yang boleh menyebabkan masalah pernafasan. Kes pertama yang dilaporkan adalah pada tahun 2019, dan sehingga sekarang, tahun 2022, koronavirus masih wujud. Pertubuhan Kesihatan Sedunia (WHO) menggalakkan kita untuk sentiasa memakai pelitup muka di tempat awam kerana pemakaian pelitup muka adalah lapisan perlindungan terakhir untuk kita. Sehingga sekarang, telah banyak sistem diperkenalkan untuk mengesan pelitup muka, akan tetapi, pengesan gagal untuk mengenal wajah mereka kerana ia terlalu kabur. Oleh itu, kaedah super-resolution (SR) telah digunakan untuk menambah baik resolusi gambar. Pilih dan uji prestasi resolusi oleh Enhanced Deep Super-Resolution (EDSR) dan Super-Resolution Generative Adversarial Networks (SRGAN). EDSR telah memenangi tempat pertama dalam cabaran NTIRE2017 SR kerana ia adalah penambahbaikan kepada ResNet. EDSR telah mengeluarkan lapisan normalisasi secara kumpulan yang tidak hanya menambah baik prestasi super-resolution malahan turut mengurangkan penggunaan memori GPU sebanyak 40%. Sementara SRGAN pula boleh membina semula gambar tanpa menjejaskannya. Pembinaan semula set data resolusi rendah kepada set data resolusi tinggi menggunakan SRGAN. Data set resolusi rendah dan tinggi akan dilatih oleh MobileNetV2 lebih mementingkan kelajuan, dan EfficientNet-B0 dan ResNet50 pula mementingkan prestasi tinggi. Model ini akan diukur menggunakan matriks kekeliruan (confusion matrix). MobileNetV2 + SR ialah model terbaik dengan 100% kekhususan dan ketepatan, 98% ketepatan dan skor F1, dan 97% kepekaan.

Keywords: Pengecaman Imej. SRGAN, ResNet50, MobileNetV2, EfficientNet-B0

ACKNOWLEDGEMENTS

At the end of my research report, I would like to thank everyone who made this research report possible and gave me a pleasant experience.

First and foremost, I would like to thank the University of Malaya and the Faculty of Computer Science and Information Technology for this great research opportunity.

Also, I would like to sincerely thank my supervisor, Dr. Erma Rahayu Binti Mohd Faizal Abdullah, who helped and guided me. Thanks for her invaluable guidance and professional advice throughout the research. It was her inspiration and encouragement that kept me focused on my research.

Lastly, I would like to express my deepest appreciation and gratitude to my parents for their constant support, emotional understanding, and forbearance throughout my study. I am also very grateful to all my course mates and colleagues that help me so much not only in the moral support and help but also lend their mouths to comment on my research. With their help, I successfully completed my research report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ACC	:	Accuracy
AI	:	Artificial Intelligence
AP	:	Average Precision
BN	:	Batch Normalization
CNN	:	Convolutional Neural Network
DDAN	:	Deep Dual Attention Network
EDSR	:	Enhanced Deep Super-Resolution Network
FN	:	False Negative
FP	:	False Positive
FxSR	:	Flexible Style of SISR
HR	:	High-Resolution
LapSRN	:	Laplacian Pyramid Super-Resolution Network
LR	:	Low-Resolution
MCNet	:	Motion Compensation Network
MDSR	:	Multi-Scale Deep Super-Resolution System
MSE	:	mean squared error
PSNR	:	peak signal-to-noise ratio
ReconNet	:	Reconstruction Network
ResNet	:	Residual Neural Network
ROI	:	Region of Interest
SC	:	Sparse Coding
SISR	:	Single Image Super-Resolution
SICNN	:	single-image convolutional neural network
SR	:	Super-Resolution

SRCNet	:	Super-Resolution and Classification Network
SRCNN	:	Super-resolution convolutional neural network
SRGAN	:	Super-Resolution Generative Adversarial Network
SSIM	:	structural similarity index
TN	:	True Negative
TP	:	True Positive
VSR	:	Video Super Resolution
VSRnet	:	Video Super Resolution Network
WHO	:	World Health Organization

CHAPTER 1: INTRODUCTION

1.1 Research Background

Coronavirus disease (COVID-19) is an infectious illness that can cause severe respiratory illness, so infected people will be felt hard to breathe. The seriously ill patient will be needed mechanical ventilation to support the respiratory process. In 2020, World Health Organization (WHO) reported the coronavirus disease was first identified in China in 2019 and declared an international health emergency in January 2020 (Tomás et al., 2021).

To prevent the spread of the disease, many countries have enforced lockdown for the whole country. According to the WHO, the most effective precaution is wearing a face mask, staying at least 1-meter social distancing apart from others, and washing their hands or using hand sanitizer frequently. With the vaccine being developed and the majority of the population vaccinated, the government and health agencies are decided to reopen the country in order to solve the economic crisis and recession (Sethi et al., 2021). Whereas it is very challenging to monitor whether all the people have worn a face mask in the public area manually. Therefore, the face mask detection system which can do live checking is highly recommended and required, in order to restrict the people is wearing a face mask correctly all the time in the public area during this pandemic.

Face mask detection system is artificial intelligence (AI) that uses machine learning techniques and deep learning algorithms to identify whether the people are wearing a face mask. It will send out a notification to alert the in-charge person to take action if detected the person is not wearing a mask. It is useful to detect those who do not wear a face mask in public transportation, airports, and hospitality.

1.2 Statement of the Problem

Face mask detection systems have been widely used in society, but it is difficult to detect and identify blurred faces from different angles in video streams. (Ahmed et al., 2021). When the system is failed to predict indistinct faces, and the infected people are entered into the crowded area, then a new cluster of diseases will be triggered.

1.3 Research Questions

The research questions that need to be addressed

(a) Which super-resolution method with scale up the resolution of the image and video features for face mask detection?

(b) How to identify a suitable super-resolution method to combine with the convolutional neural network algorithm for face mask detection?

(c) How to evaluate and compare the model performance between the model with and without a super-resolution method for face mask detection?

1.4 Research Objectives

The objectives of this research are

(a) To investigate the super-resolution method with scale up the resolution of the image and video features for face mask detection.

(b) To identify a suitable super-resolution method to combine with the convolutional neural network algorithm for face mask detection.

(c) To evaluate and compare the model performance between the model with and without a super-solution method for face mask detection.

1.5 Research Significant

The significance of this research is to increase the resolution of the image by using the super-resolution (SR) method combined with the convolutional neural network (CNN) algorithm to do face recognition. The accuracy will be increased as the high-resolution image is available.

1.6 Scope of Study

In this research, two types of datasets will be used as image and video, the image dataset consists of three classes that are with mask, without a mask, and mask worn incorrectly. Two types of super-resolution methods will be studied and combined with the CNN for mask detection. Moreover, two python scripts will be prepared for image and video detection.

1.7 Organization of the Research Proposal

This research report contains 5 chapters including introduction, literature review, methodology, results, discussion, and conclusion. References will be followed these 6 chapters.

Chapter 1 is the research introduction. A brief introduction to the research background, articulating the problem statement, research question, and research objectives. Describe the organization of the research scope and research plan.

Chapter 2 is a literature review of super-resolution methods and convolutional neural network (CNN) models for mask detection.

Chapter 3 is the methodology of super-resolution combined with convolutional neural network (CNN) models. We will discuss and understand the method.

Chapter 4 is the results and discussion. To do comparison and discussion of the results of different types of super-resolution methods combined with convolutional neural network (CNN) models for mask detection.

Chapter 5 is the conclusion of this study on the most suitable super-resolution methods combined with convolutional neural network (CNN) models.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Several super-resolution (SR) methods have been introduced to effectively increase the resolution of images and videos. Convolutional Neural Networks (CNN) consists of three layers, an input layer, multiple hidden layers, and an output layer, and therefore widely used in image and video recognition. In this chapter, the features and performance of SR and CCN methods will be investigated and compared.

2.2 Super Resolution

Super resolution is the process of increasing the resolution of an image, meaning that the pixels in the image are denser and can be displayed more flexibly. This process helps us reduce noise and deblur the image by appropriately combining several separately acquired images. Therefore, we can find important features and identify objects efficiently. Super resolution can be divided into single image super resolution (SISR) and video super resolution (VSR). Figures 2.1 and 2.2 show a comparison of sample images before and after super-resolution processing. It clearly shows that the image after super-resolution processing is sharper.

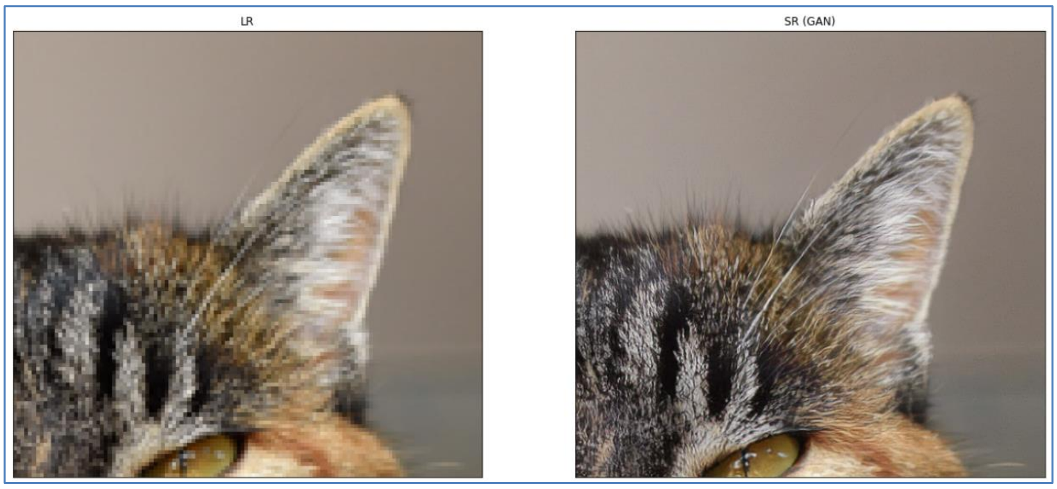


Figure 2.1: Image Comparison Before (left) and After Super Resolution Processing (1)

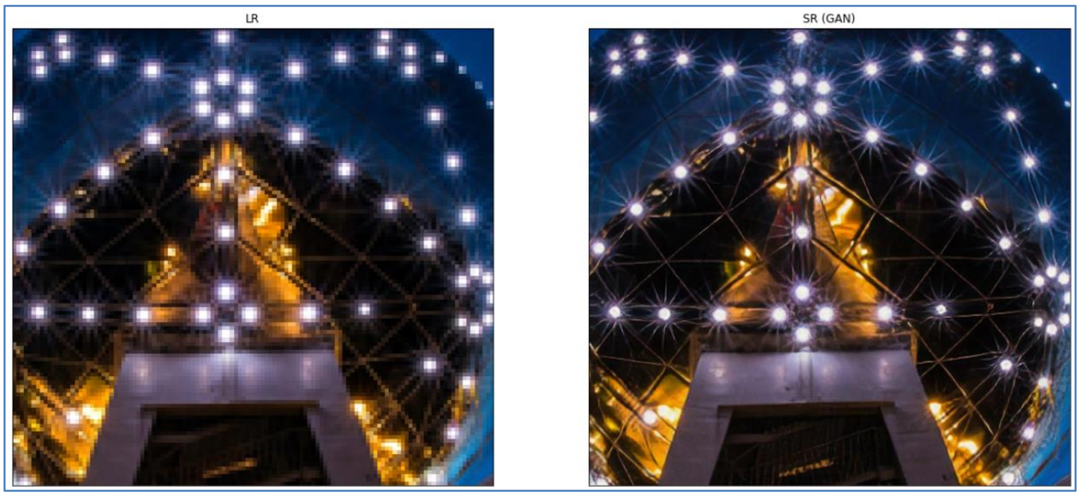


Figure 2.2: Image Comparison Before (left) and After (right) Super Resolution Processing (2)

2.2.1 Single Image Super Resolution (SISR)

Single Image Super Resolution (SISR) is converting low-resolution (LR) images to high-resolution (HR) images. The super resolution convolutional neural network (SRCNN) is a simple, robust, fast, and highly accurate method (Dong et al., 2014). It is developed by a fully convolutional neural network for image super-resolution, which directly learns the end-to-end mapping between low- and high-resolution images.

SRCNN has several attractive properties. First, SRCNN has a simpler structure and higher accuracy than state-of-the-art example-based methods. Second, SRCNN is fully feed-forward and consists of a moderate number of filters and layers, which makes it faster than many example-based methods. Third, the recovery quality of the network can be further improved when dealing with larger datasets. Furthermore, SRCNN can process three-channel color images simultaneously to improve super-resolution performance. SRCNN outperforms the bicubic baseline and sparse coding-based methods (SC) with a few and moderate training, respectively. Referring to Figure 2.3, we can observe that the butterfly image is sharper than bicubic and SC. Dong et al. (2014) pointed out that SRCNN surpasses the bicubic baseline with only a few training iterations and outperforms sparse-coding-based methods (SC) with moderate training.

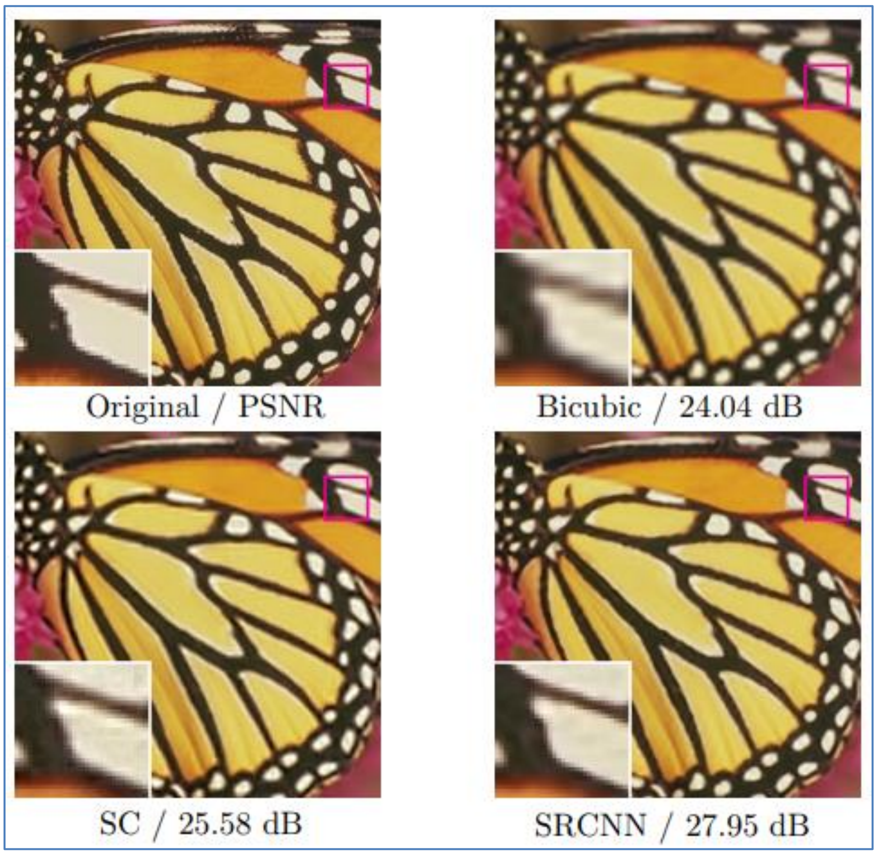


Figure 2.3: Comparison of Butterfly Images at 3x Magnification in Original, Bicubic, SC, and SRCNN (Dong et al., 2014)

In 2017, four super-resolution networks were proposed: Laplacian Pyramid Super-Resolution Network (LapSRN), Enhanced Deep Super-Resolution Network (EDSR), Multi-Scale Deep Super-Resolution System (MDSR), and Super-Resolution Generative Adversarial Networks (SRGAN). Laplacian Pyramid Super-Resolution Network (LapSRN) was proposed by Lai et al. (2017), progressively reconstruct SR images at real-time speed through a large-capacity deep network. Figure 2.4 shows the network architecture of LapSRN, with red arrows representing convolutional layers, blue arrows representing transposed convolutions that upsample the input image by a factor of 2, and green arrows representing element-wise addition operators. According to Lai et al. (2017), the architecture of LapSRN naturally accommodates deep supervision, where the supervision signal can be simultaneously applied to each layer of the pyramid. The LapSRN model has two branches, such as feature extraction and image reconstruction, and its computational complexity is significantly reduced compared to other networks that perform all feature extraction and reconstruction at finer resolutions.

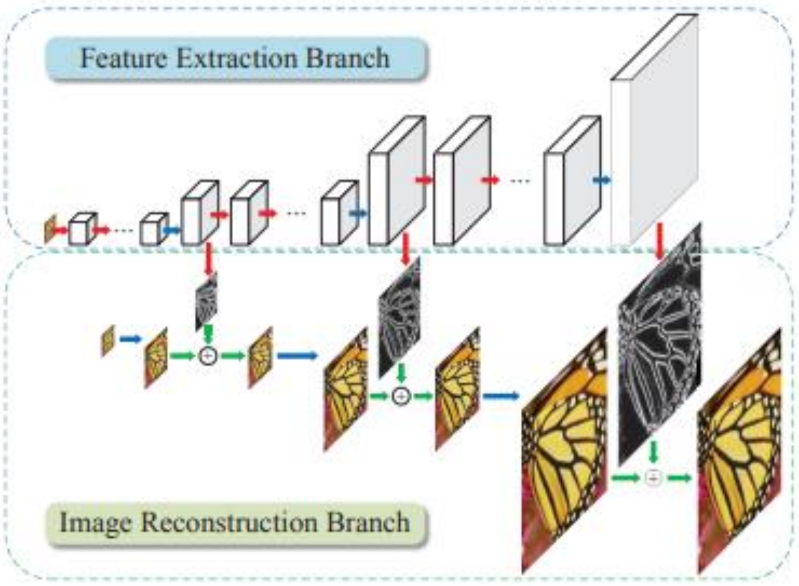


Figure 2.4: LapSRN Architecture (Lai et al. 2017)

Additionally, Lim et al. (2017) developed an Enhanced Deep Super-Resolution Network (EDSR) that outperformed current state-of-the-art SR methods and proposed a new Multi-Scale Deep Super-Resolution System (MDSR) to reconstruct various high-resolution images of various scales in a single model. The Batch Normalization (BN) layer is removed along with the final ReLU activation, as shown on the right side of Figure 2.5. According to Lim et al. (2017), batch normalization loses the scale information of the image and reduces the range flexibility of activations. Removing the batch normalization layer not only improves super-resolution performance but also reduces GPU memory usage by up to 40% so that significantly larger models can be trained with EDSR models.

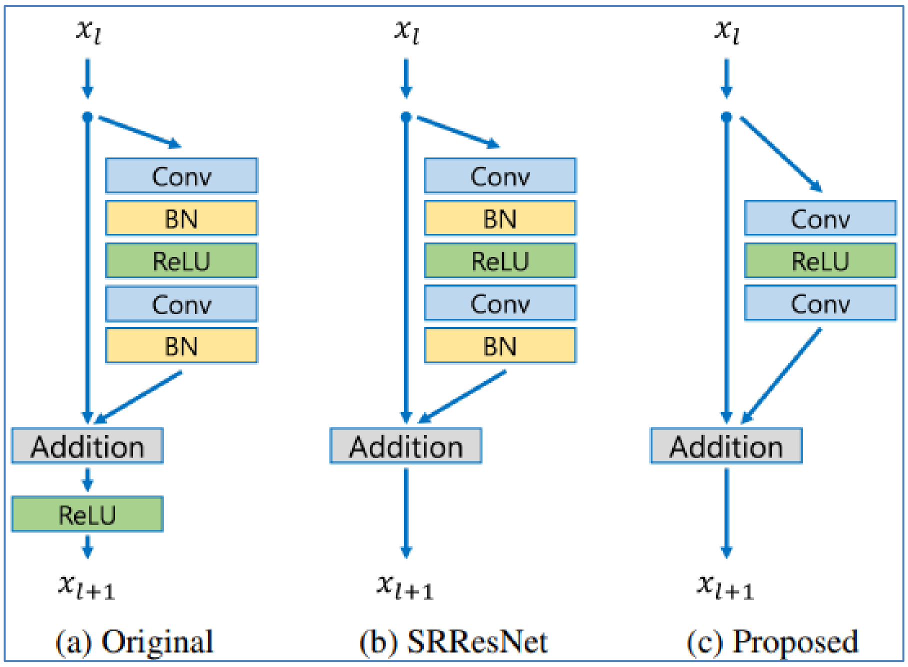


Figure 2.5: Comparison of Residual Blocks in Original ResNet, SRResNet, and EDSR (c)

On the other hand, MDSR is a multi-scale model, and while still compact compared to a set of single-scale models, it can reduce model size and training time. The NTIRE 2017 SR challenge aims to propose single-scale and multi-scale super-resolution networks with the highest peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM). Therefore, EDSR and MDSR won first and second place respectively in the challenge. Figures 2.6 and 2.7 are an overview of the EDSR and MDSR architectures:

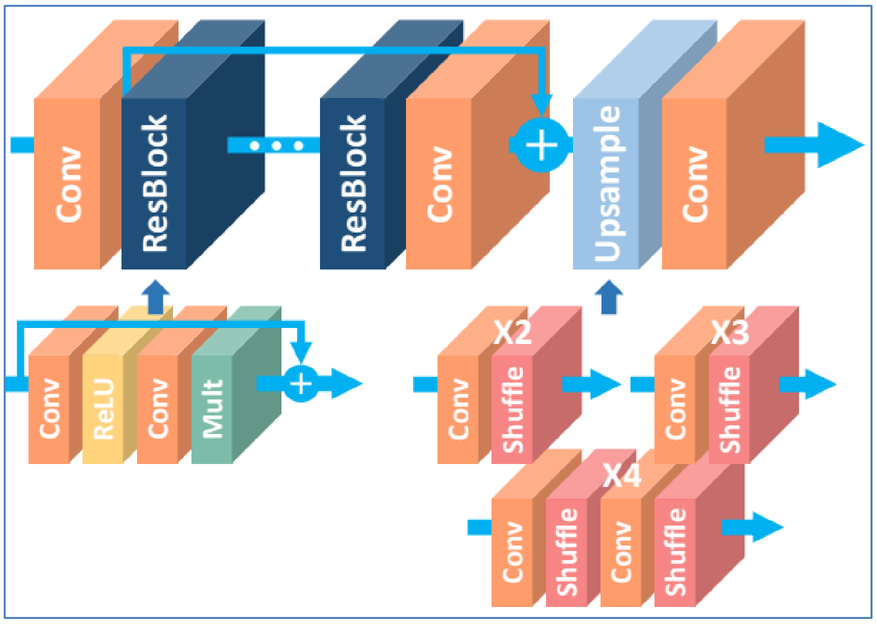


Figure 2.6: EDSR Architecture (Lim et al. 2017)

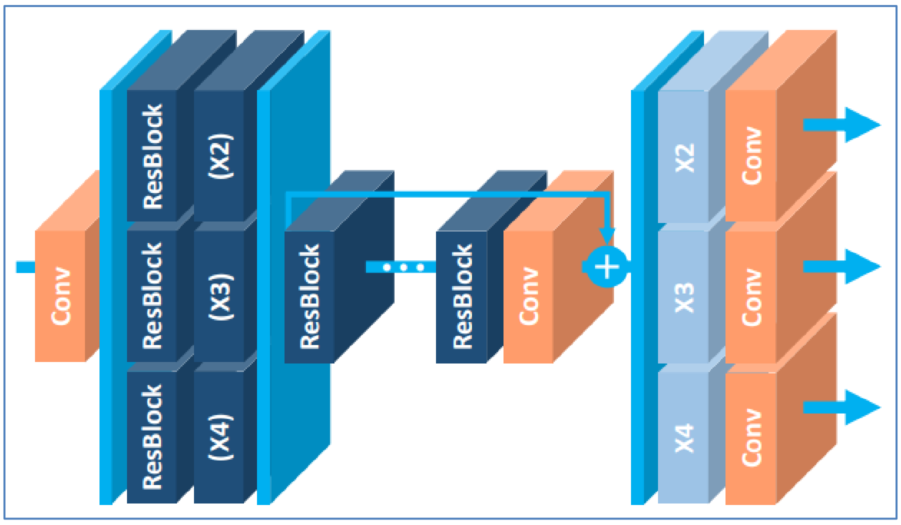


Figure 2.7: MDSR Architecture (Lim et al. 2017)

Furthermore, Ledig et al. (2017) introduced a Super-Resolution Generative Adversarial Network (SRGAN), which is used to recover finer textures from an image without compromising its quality when we scale up the image. They use a perceptual loss function consisting of a content loss and an adversarial loss. The content loss compares the deep features extracted from the SR and HR images with the pre-trained VGG network.

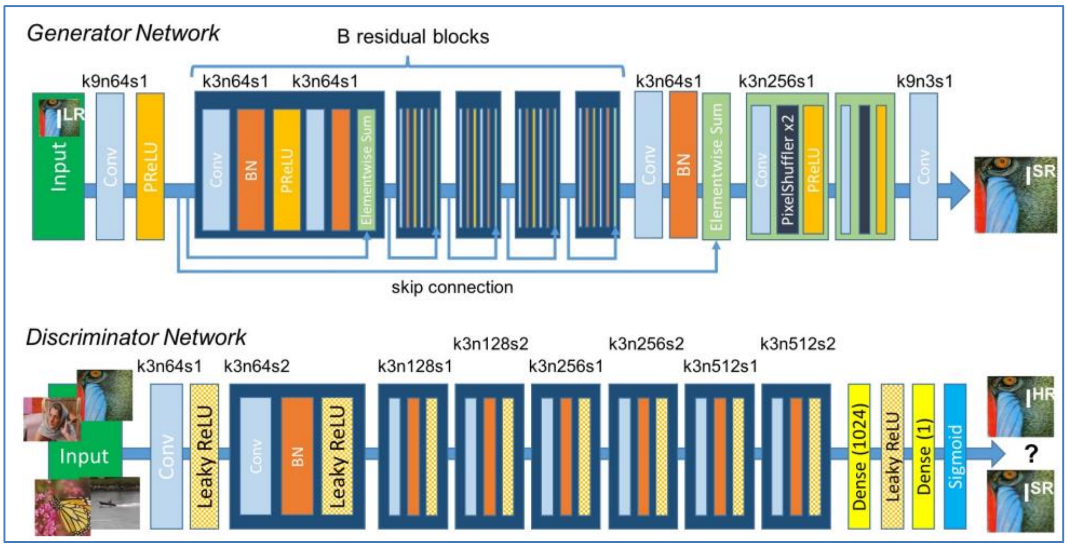


Figure 2.8: SRGAN Architecture (Ledig et al. 2017)

Figure 2.8 shows the overall architecture of SRGAN, which consists of a generator network and a discriminator network. The generator network generates some data according to the probability distribution, and the discriminator network is used to distinguish the real HR image and the generated SR image. k is the kernel size, n is the number of feature maps, and s is the stride. The generator network consists of 16 identical (B = 16) residual blocks and uses connection types such as skip connections. Two convolutional layers are used in the residual block, "k3n64s1" represents a 3 X 3 kernel filter, outputting 64 channels with a stride of 1. While the discriminator network consists of 8 convolutional layers and 3 X 3 filter kernels, the number of feature maps will increase from 64 kernels to 512 kernels by a factor of 2. Stride convolution reduces image resolution when the number of features is doubled. The resulting 512 feature maps are followed by two dense layers and a leakyReLU applied in between, and a final sigmoid activation function to obtain the probability of sample classification.

In 2020, Liu et al. has proposed single-image convolutional neural network (SICNN) method, which upscales low-resolution images to a suitable size and directly reconstructs the features of SR images. As the CNN network gets deeper and deeper, the number of parameters also increases. Therefore, they decided to design a small CNN to reconstruct SR images and train using the internal dataset, which makes SICNN better adaptable and shorter training time. It consists of two branches, such as the small-scale feature branch and the large-scale feature branch. Small-scale feature maps train feature maps from down sampled image patches to upscaled image patches, while large-scale feature maps train feature maps from low-resolution image patches to upscaled image patches. Figure 2.8 is an overview of the SICNN architecture:

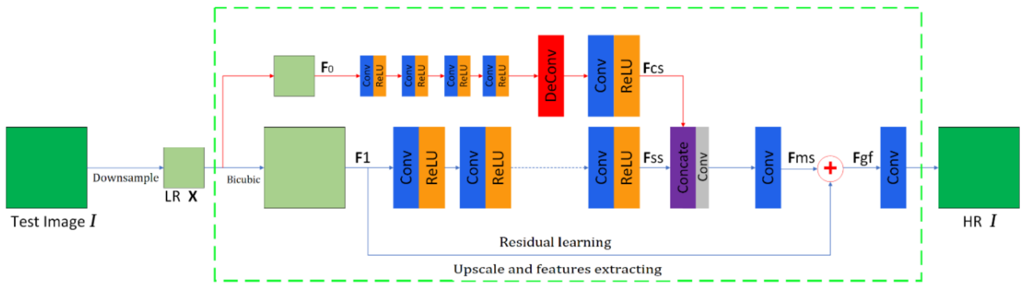


Figure 2.9: SICNN Architecture (Liu et al. 2020)

The test image is denoted I, then it will down sample the image denoted (LR X). For the small-scale feature map branch, the input is denoted as F_0. The image will go through 4 convolutional layers and 8 Rectified Linear Units (ReLU) and the output is denoted as F_{cs}. For the large-scale feature map branch, the input is denoted F_1, which is obtained from bicubic interpolation, and then it goes through 8 convolutional layers and 8 rectified linear units, and the output is denoted as F_{ss}. Afterward, the outputs are combined into a concatenation of convolutional layers and mapped to a multi-scale feature F_{ms}. The final output of SICNN is denoted as F_{gf}. The advantage of SICNN is that it can reduce network training time as it does not require external datasets for training and can reconstruct image details clearly by capturing image features at different scales.

Furthermore, the Super-Resolution and Classification Network (SRCNet) proposed by Qin and Li (2020) has been used to identify mask conditions with 98.7% accuracy. Super-resolution crack networks (SrcNet) can improve the ability to automatically detect cracks based on computer vision. The use of unmanned robots to obtain digital images for crack detection from large civilian infrastructure is often affected by motion blur and insufficient pixel resolution, which may reduce the corresponding crack detection capabilities. According to Bae et al. (2021), the proposed SrcNet can improve the pixel resolution of the original digital image through deep learning, thereby significantly improving the crack detection capability. SrcNet basically consists of two phases: the first phase is the deep learning-based super-resolution (SR) image generation, and the second phase is the deep learning-based automated crack detection. Once the raw digital image is obtained from the target bridge deck, the first stage of SrcNet generates the corresponding SR image for the raw digital image. The second stage then automatically detects cracks in the generated SR images, significantly improving crack detection capabilities. SRFlow is a flow-based normalized SR method trained with negative log-likelihood, which is a single loss. Thus, it addresses ill-posed natural problems and learns to predict a variety of realistic HR images (Lugmayr et al., 2020). Park et al. (2022) proposed a flexible style of SISR and named it FxSR. It can produce different styles corresponding to the feature losses employed and can also generate intermediate results between different styles. In addition, we can control local areas in different ways by providing control charts to the network. If the user focuses on the salient regions or foregrounds rather than backgrounds, they are able to generate more natural SR outputs. Users are also able to edit or auto-generate the segmentation map and others such as saliency or depth maps. Users can remediate unnaturally generated areas by controlling the parameters in the post-processing step. It flexibly reconstructs images between perception-oriented and distortion-oriented images to generate images with the desired restoration style for each region. Therefore, FxSR can be classified into two types: perceptual distortion (FxSR-PD) and diversity (FxSR-DS).

2.2.2 Video Super Resolution (VSR)

Video Super Resolution (VSR) is a combination of SISR and converting low-resolution video to high-resolution video. There are two deep learning methods for VSR, such as Video Super-Resolution Network (VSRnet) and Deep Dual Attention Network (DDAN). VSRnet was proposed by Kappeler et al. (2016) for estimating motion information between input frames via optical flow. Figure 2.10 shows three different video super-resolution architectures. In architecture (a), the three input frames are concatenated before applying the first convolutional layer. In architecture (b), the frames will be combined between the first and second convolutional layers, and after the second convolutional layer in architecture (c), the frames will be concatenated.

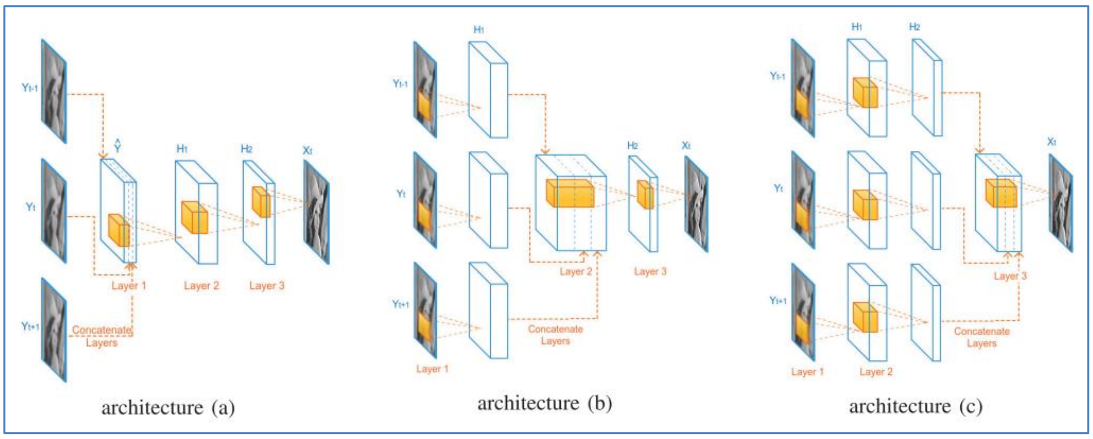


Figure 2.10: Video Super Resolution (VSR) Architecture (Kappeler et al., 2016)

According to Li et al. (2020), Deep Dual Attention Network (DDAN) includes a motion compensation network (MCNet) that progressively learns optical flow representations to synthesize motion information between adjacent frames in a pyramidal fashion, and an SR reconstruction network (ReconNet), for generating high-resolution frames from low-resolution input. In general, deep dual attention networks are used for accurate spatiotemporal informative features of video SR with high quantitative performance. The DDAN architecture is shown in the following figure:

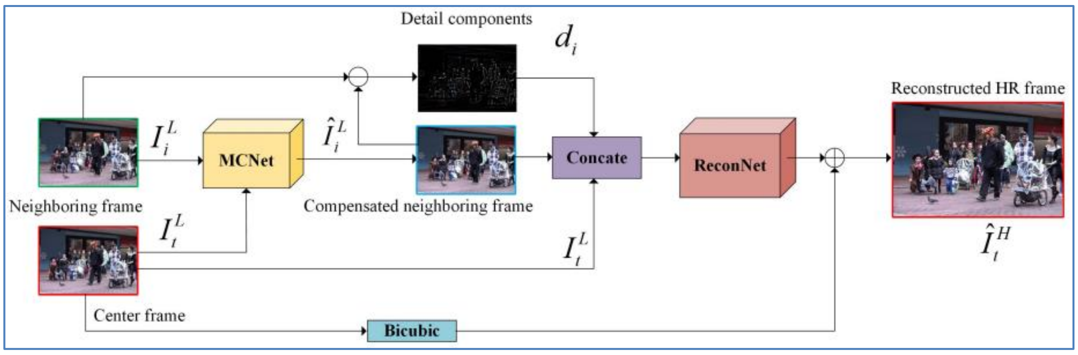


Figure 2.11: Deep Dual Attention Network (DDAN) Architecture (Li et al., 2020)

2.2.3 SR Quality Assessment

The evaluation metric used to measure SR performance is the peak signal-to-noise ratio (PSNR), which is defined by the maximum pixel value and mean squared error (MSE) between images and the structural similarity index (SSIM) used to measure the structural similarity between images. LapSRN and DDAN are outperformed compared to the SICNN, SRCNN, and VSRnet methods which are tested by the public datasets such as Set 5 and Set 14 for the SISR method and Vid4 and Myanmar for the VSR method (Athirasree, 2021). Compared to SRFlow, FxSR has higher perceptual quality but a lower diversity score (Park et al. 2022).

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a supervised deep-learning algorithm best suited for image recognition and computer vision. CNN consists of multiple hidden layers that are used to process and extract facial regions of interest (ROI) in face mask detection systems.

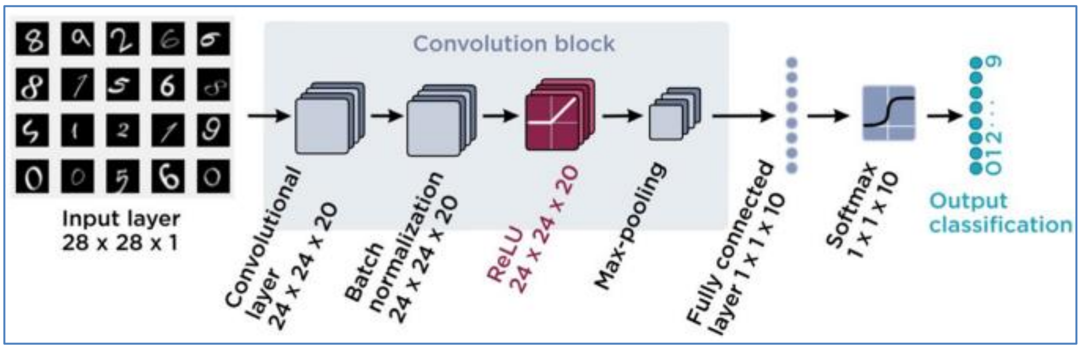


Figure 2.12 Convolutional Neural Network (CNN) Architecture (Surekcigil Pesch et al., 2022)

It consists of three layers, an input layer, multiple hidden layers, and an output layer. Multiple hidden layers are mainly used for convolution to detect and find features in images, pooling is used to reduce image overfitting, and full connections are used to identify and classify images from pooling layers. Existing mask detection algorithms can be divided into two categories: real-time algorithms and high-performance algorithms.

Various pre-trained models based on CNN architecture and their achievements are shown in Figure 2.11.

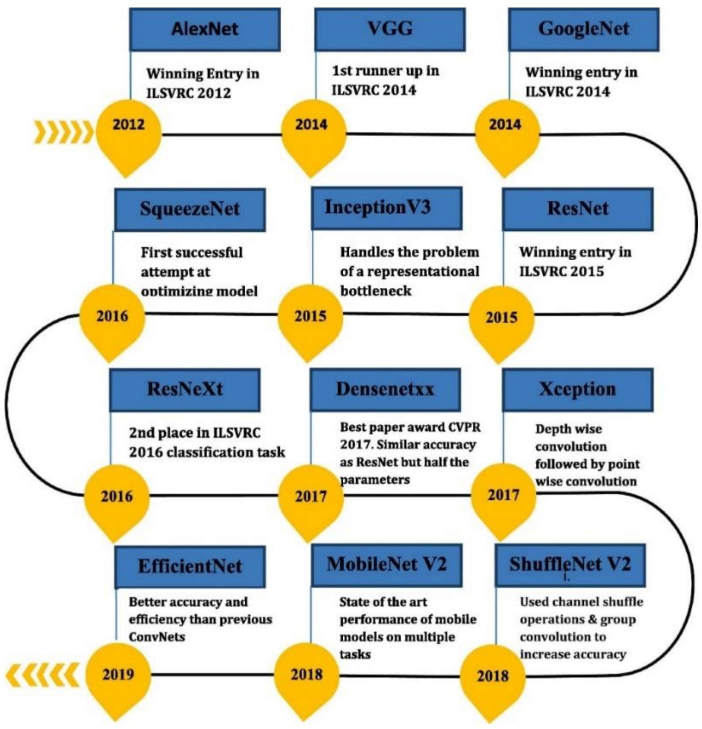

AlexNet
Winning Entry in ILSVRC 2012
2012
VGG
1st runner up in ILSVRC 2014
2014
GoogleNet
Winning entry in ILSVRC 2014
2014
SqueezeNet
First successful attempt at optimizing model
2016
InceptionV3
Handles the problem of a representational bottleneck
2015
ResNet
Winning entry in ILSVRC 2015
2015
ResNeXt
2nd place in ILSVRC 2016 classification task
2016
Densenetxx
Best paper award CVPR 2017. Similar accuracy as ResNet but half the parameters
2017
Xception
Depth wise convolution followed by point wise convolution
2017
EfficientNet
Better accuracy and efficiency than previous ConvNets
2019
MobileNet V2
State of the art performance of mobile models on multiple tasks
2018
ShuffleNet V2
Used channel shuffle operations & group convolution to increase accuracy
2018

Figure 2.13 Various Pre-trained Models based on CNN Architectures with its Achievement (Sethi et al., 2021)

2.3.1 Real-Time Algorithm

The main feature of real-time algorithm is to prioritize fast inference speed. One of the real-time CNN algorithms is MobileNet proposed by Howard et al. (2017). It is designed for mobile devices and developed by using depthwise separable convolutions based on a streamlined architecture. Also, it improves the user experience as it can run anytime, anywhere. In the following 2018, they introduced 2 new features to add linear bottlenecks between layers and shortcut connections between bottlenecks to MobileNet, called

MobileNetV2. By adding these 2 new features, MobileNetV2 has higher accuracy and lower latency compared to MobileNetV1.

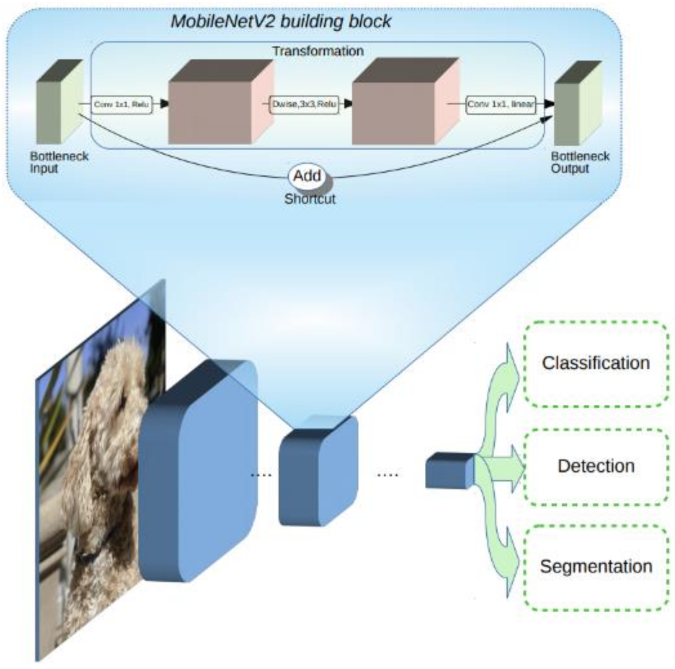
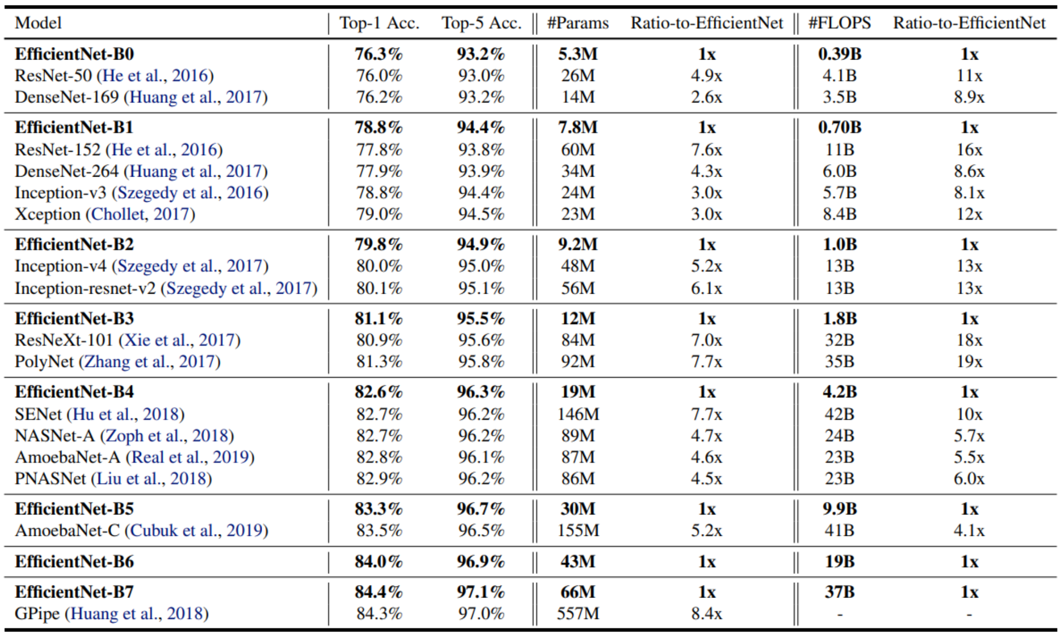


Figure 2.14 Overview of MobileNetV2 Architecture (Sandler & Howard, 2018)

2.3.2 High Performance Algorithm

The main characteristic of high-performance algorithm is to prioritize accuracy. Tan & Le (2019) introduced a new family of neural networks, EfficientNets. As the name suggests, EfficientNets are very computationally efficient and have also achieved state-of-the-art results on the ImageNet dataset with a top-1 accuracy of 84.4%. Meanwhile, they proposed a new scaling method, called compound scaling, and advise against scaling only one model attribute's depth, width, and resolution; strategically combining the three can lead to better results. They have demonstrated the effectiveness of this approach in extending MobileNets and ResNet. For example, ResNet can be extended from Resnet18 to ResNet200, which increases the number of residual blocks from 18 to 200, and ResNet200 provides better performance than ResNet18. However, there is a problem with the traditional manual scaling method, that is, the performance will not improve after reaching a certain level but will start to have a negative impact as the performance decreases. EfficientNet models are developed using a multi-objective neural architecture search optimized for accuracy and floating-point operations. EfficientNet-B0 was used as the baseline model because its architecture was not developed by engineers, but by the neural network itself. Furthermore, they extend the EfficientNet model by applying the compound scaling method to the baseline EfficientNet-B0 with different compound coefficients. Finally, they successfully developed a complete family of EfficientNet models from B1 to B7, achieving state-of-the-art accuracy on ImageNet while being very effective against competitors mentioned in Table 2.1.

Table 2.1: EfficientNet Performance Results on ImageNet (Tan & Le, 2019)



Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**78.8%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**79.8%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.1%**	**95.5%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

Residual Neural Networks (ResNet) use multiple layers and apply data augmentation techniques to improve the effectiveness of predictions (Basha et al., 2021). Although ResNet50 is one of the versions of ResNet, it consists of 50 layers deep (Loey et al., 2021). To reduce training layer time and improve accuracy, the bottleneck stack is 3 layers instead of 2 (Boesch, 2021). Loey et al. (2021) introduced the medical masked face detection model, YOLOv2 with the ResNet-50 model for classification and detection functions, with an average precision (AP) value of 81%.

2.4 Conclusion

According to the research, Enhanced Deep Super-Resolution Network (EDSR) optimizes the network architecture by analyzing and removing unnecessary modules. Reduced GPU memory usage by 40% and improved super-resolution performance. Additionally, it was the winner of the NTIRE 2017 Super-Resolution Challenge. Super-Resolution Generative Adversarial Network (SRGAN) consists of a generator network and a discriminator network. The advantage of the Super-Resolution Generative Adversarial Networks (SRGAN) is to recover finer textures without compromising image quality.

Moreover, Convolutional Neural Network (CNN) is a supervised deep-learning algorithm consisting of multiple hidden layers, which is the most suitable algorithm when we need to extract the facial region of interest (ROI) while doing image recognition. CNN algorithms can be divided into two categories: real-time algorithms and high-performance algorithms. MobileNetV2 is developed based on a streamlined architecture. Compared with MobileNetV1, it has higher accuracy and lower latency. EfficientNet is very computationally efficient and achieves state-of-the-art results on the ImageNet dataset with a top-1 accuracy of 84.4%. ResNet50 also has high accuracy and low training time.

Therefore, the Enhanced Deep Super-Resolution Network (EDSR) and Super-Resolution Generative Adversarial Network (SRGAN) will be tested using existing datasets, and a better super-resolution network will be selected by comparing their test outputs. Then choose MobileNetV2, EfficientNet-B0, and ResNet50 algorithms to classify the image dataset.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter discusses the methodology of super-resolution (SR) methods such as Enhanced Deep Super-Resolution Network (EDSR) and Super-Resolution Generative Adversarial Network (SRGAN) will be tested, and the best super-resolution network will be selected. Convolutional Neural Network (CNN) algorithms such as MobileNetV2, EfficientNet-B0, and ResNet50 will be used for mask detection and classification.

3.2 CRISP-DM

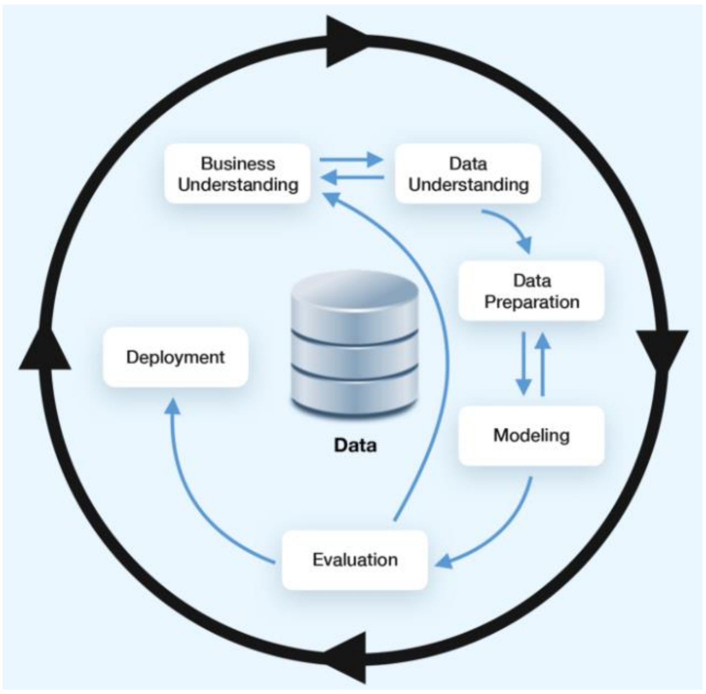


Figure 3.1 CRISP-DM Methodology (Quantum, 2019)

This research is conducted based on the methodology of CRISP-DM, there have 6 steps which are Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Figure 3.2 is showing the general flowchart of the face mask detection system.

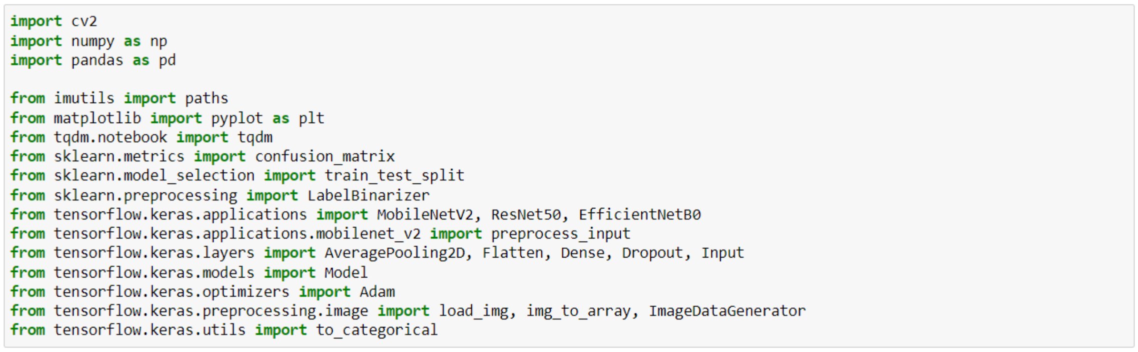
3.2.1 Business Understanding

The mask detection model has three objectives. First, investigate the super-resolution (SR) method with scale up the resolution of the image and video features for face mask detection. An Enhanced Deep Super-Resolution Network (EDSR) and a Super-Resolution Generative Adversarial Network (SRGAN) were selected for further testing.

Second, identify a suitable super-resolution method to combine with the convolutional neural network algorithm for face mask detection. EDSR and SRGAN will be tested using the DIV2K dataset and we will evaluate their outputs. The most efficient super-resolution network will be selected.

Third, evaluate and compare the model performance between the model with and without a super-solution method for face mask detection. A super-resolution network will be selected from the previous stage and then used to convert the mask image dataset to high resolution. In the end, we will have 2 sets of datasets, such as low-resolution (without going through the SR network) and high-resolution image datasets. The three convolutional neural network (CNN) algorithms are MobileNetV2, EfficientNet-B0, and ResNet50, which will be trained and tested on low- and high-resolution datasets. Therefore, we will have 6 models and we will evaluate and compare all of them. The best models will be deployed and used for image and video mask detection.

On the other hand, the model will be written in Python using Jupyter notebooks. Jupyter notebook is a free and open-source web-based interactive computing platform. It contains many modules for us to use, so we can use the function directly by just importing the library from the specific module. It helps us save a lot of effort by calling the function instead of writing the whole function from scratch. Figure 1 shows all the necessary packages and libraries that need to be used in the model. For example, MobileNetV2, ResNet50, and EfficientNetB0 algorithms are available in the Keras library of the TensorFlow module.



```
import cv2
import numpy as np
import pandas as pd

from imutils import paths
from matplotlib import pyplot as plt
from tqdm.notebook import tqdm
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.applications import MobileNetV2, ResNet50, EfficientNetB0
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.layers import AveragePooling2D, Flatten, Dense, Dropout, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import load_img, img_to_array, ImageDataGenerator
from tensorflow.keras.utils import to_categorical
```

Figure 3.2: All the Necessary Packages and Libraries

3.2.2 Data Understanding

Two types of datasets have been collected such as image and video.

Table 3.1: Meta data of Image Dataset

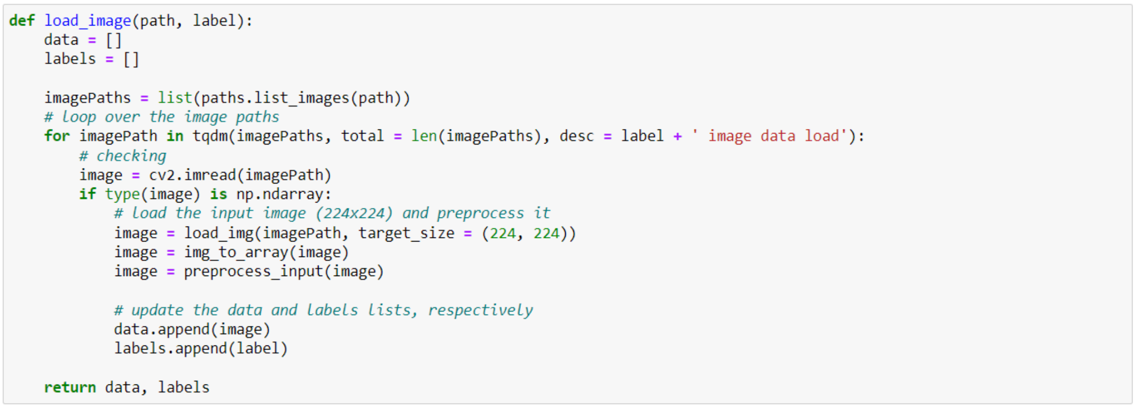
Type of Dataset	Image
Source	https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection
File Size	226MB
Total of Images	8982
Classes	With Mask, Without Mask, Mask Worn Incorrectly

Table 3.2: Meta data of Video Dataset

Type of Dataset	Video
Source	https://data.mendeley.com/datasets/v3kry8gb59/1
File Size	8891MB
Total Video Frames	4357
Classes	Mask (MW), No Mask (NM)

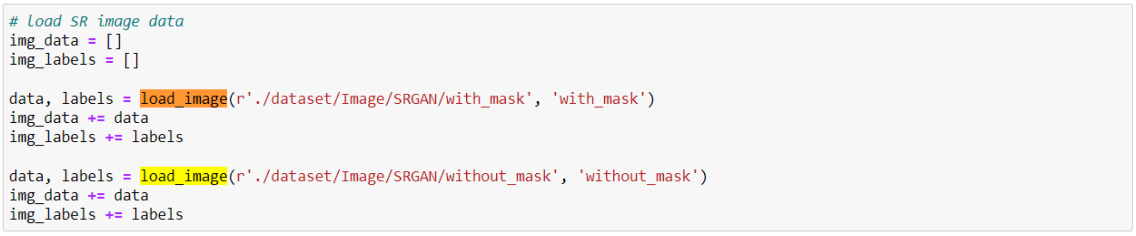
At this stage, we need to ensure that the collected image dataset can be successfully loaded into the notebook.

First, we load the image dataset into a Jupyter notebook using the path in the imutils module. It appends all images and labels as data and labels. Note that we need to check the image for corruption before appending it, otherwise it will cause problems later.



```python
def load_image(path, label):
    data = []
    labels = []

    imagePaths = list(paths.list_images(path))
    # loop over the image paths
    for imagePath in tqdm(imagePaths, total = len(imagePaths), desc = label + ' image data load'):
        # checking
        image = cv2.imread(imagePath)
        if type(image) is np.ndarray:
            # load the input image (224x224) and preprocess it
            image = load_img(imagePath, target_size = (224, 224))
            image = img_to_array(image)
            image = preprocess_input(image)

            # update the data and labels lists, respectively
            data.append(image)
            labels.append(label)

    return data, labels
```

Figure 3.3: Define a load_image Function



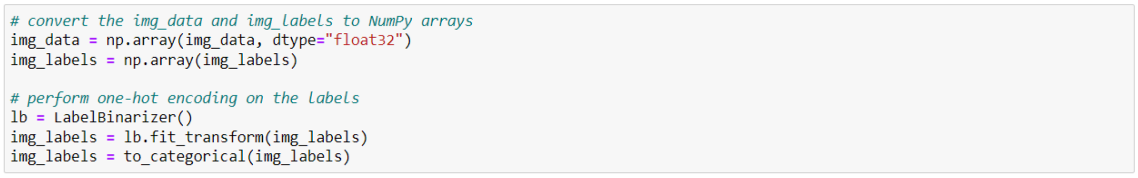
```python
# load SR image data
img_data = []
img_labels = []

data, labels = load_image(r'./dataset/Image/SRGAN/with_mask', 'with_mask')
img_data += data
img_labels += labels

data, labels = load_image(r'./dataset/Image/SRGAN/without_mask', 'without_mask')
img_data += data
img_labels += labels
```

Figure 3.4: Call the load_image Function

3.2.3 Data Preparation

At this stage, we need to prepare and modify the data and make sure the dataset is ready for modeling.

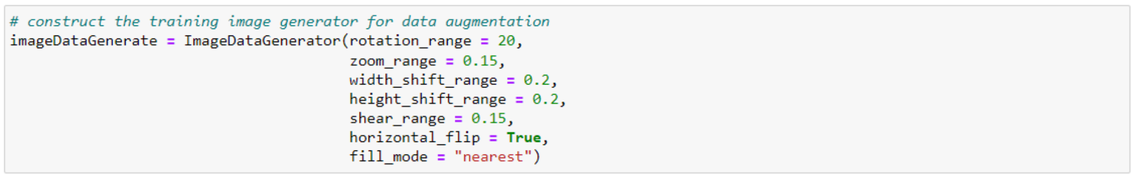
After successfully loading the dataset into Jupyter, we need to convert the data and labels to NumPy arrays. Next, we need to use the LabelBinarizer function of the preprocessing layer of the Scikit-learn module. It is used to perform one-hot encoding of categorical variables such as labels. The Python code is shown in Figure 3.5.



```python
# convert the img_data and img_labels to NumPy arrays
img_data = np.array(img_data, dtype="float32")
img_labels = np.array(img_labels)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
img_labels = lb.fit_transform(img_labels)
img_labels = to_categorical(img_labels)
```

Figure 3.5: Convert Data and Labels to NumPy Array and Encode on the Labels

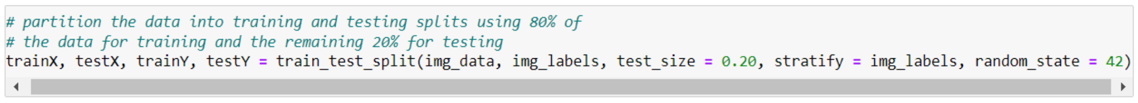
Data augmentation is widely used in computer vision because it increases the variety and amount of training data by applying random transformations. For example, resizing, rotating, and flipping images.



```python
# construct the training image generator for data augmentation
imageDataGenerate = ImageDataGenerator(rotation_range = 20,
                                       zoom_range = 0.15,
                                       width_shift_range = 0.2,
                                       height_shift_range = 0.2,
                                       shear_range = 0.15,
                                       horizontal_flip = True,
                                       fill_mode = "nearest")
```

Figure 3.6: Data Augmentation in Jupyter

The train_test_split function is used to split the image dataset into 80% training set and 20% testing set.

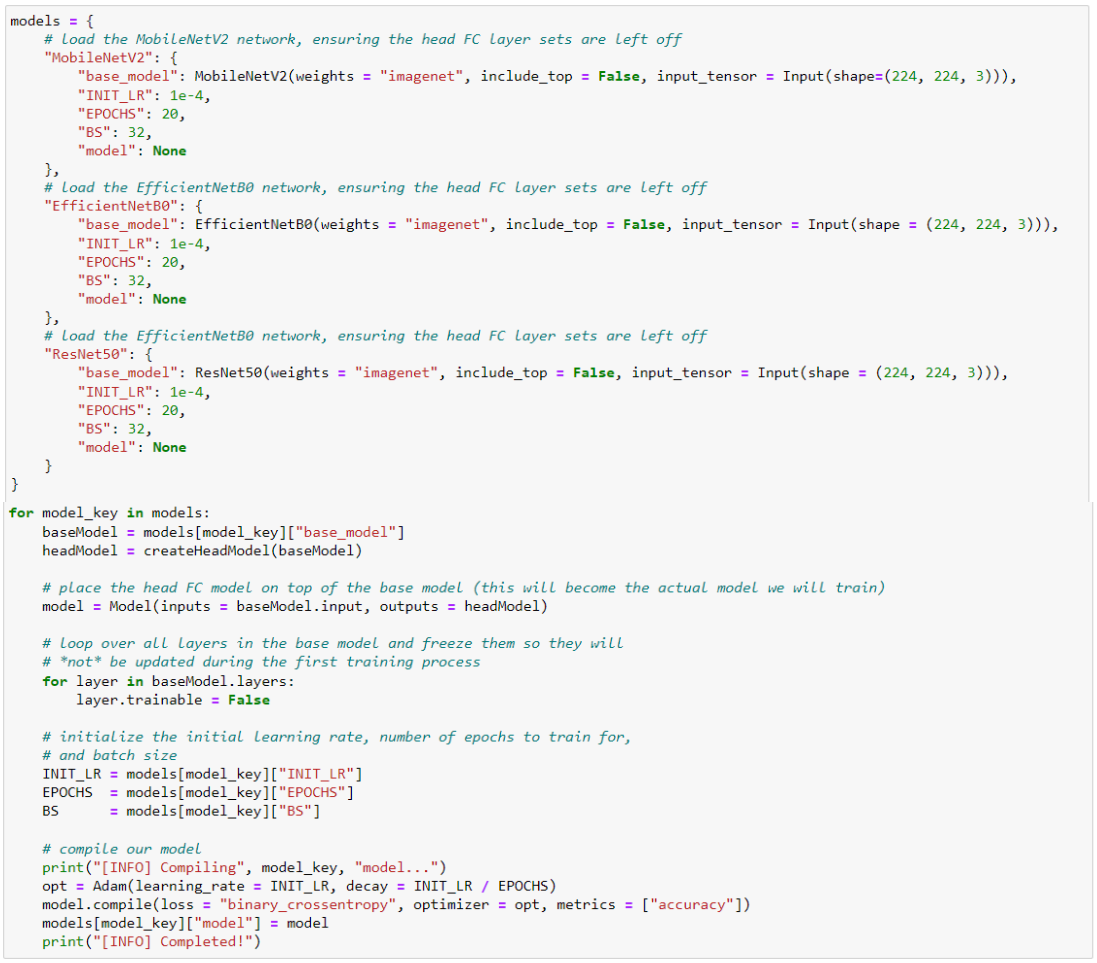


```python
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
trainX, testX, trainY, testY = train_test_split(img_data, img_labels, test_size = 0.20, stratify = img_labels, random_state = 42)
```

Figure 3.7: Use the train_test_split Function to Split the Dataset into 80% Training and 20% Testing

3.2.4 Modeling

During the modeling phase, we will use MobileNetV2, EfficientNetB0, and ResNet50 algorithms to train the model. Figure 3.8 shows the modeling python code.



```
models = {
    # load the MobileNetV2 network, ensuring the head FC layer sets are left off
    "MobileNetV2": {
        "base_model": MobileNetV2(weights = "imagenet", include_top = False, input_tensor = Input(shape=(224, 224, 3))),
        "INIT_LR": 1e-4,
        "EPOCHS": 20,
        "BS": 32,
        "model": None
    },
    # load the EfficientNetB0 network, ensuring the head FC layer sets are left off
    "EfficientNetB0": {
        "base_model": EfficientNetB0(weights = "imagenet", include_top = False, input_tensor = Input(shape = (224, 224, 3))),
        "INIT_LR": 1e-4,
        "EPOCHS": 20,
        "BS": 32,
        "model": None
    },
    # load the EfficientNetB0 network, ensuring the head FC layer sets are left off
    "ResNet50": {
        "base_model": ResNet50(weights = "imagenet", include_top = False, input_tensor = Input(shape = (224, 224, 3))),
        "INIT_LR": 1e-4,
        "EPOCHS": 20,
        "BS": 32,
        "model": None
    }
}
for model_key in models:
    baseModel = models[model_key]["base_model"]
    headModel = createHeadModel(baseModel)

    # place the head FC model on top of the base model (this will become the actual model we will train)
    model = Model(inputs = baseModel.input, outputs = headModel)

    # loop over all layers in the base model and freeze them so they will
    # *not* be updated during the first training process
    for layer in baseModel.layers:
        layer.trainable = False

    # initialize the initial learning rate, number of epochs to train for,
    # and batch size
    INIT_LR = models[model_key]["INIT_LR"]
    EPOCHS  = models[model_key]["EPOCHS"]
    BS      = models[model_key]["BS"]

    # compile our model
    print("[INFO] Compiling", model_key, "model...")
    opt = Adam(learning_rate = INIT_LR, decay = INIT_LR / EPOCHS)
    model.compile(loss = "binary_crossentropy", optimizer = opt, metrics = ["accuracy"])
    models[model_key]["model"] = model
    print("[INFO] Completed!")
```

Figure 3.8: Modeling Python Code in the Jupyter

From what we understand, super-resolution (SR) methods will increase the resolution of images. To demonstrate whether the SR method is useful in our mask detection model, hence, we need to use two image datasets to train and test the model, such as a low-resolution dataset and a high-resolution dataset. Therefore, 6 models will be generated.

Table 3.3 Generate 6 Models from Modeling

No.	Model
1	EfficientNetB0_face_mask_detector_with_SR
2	EfficientNetB0_face_mask_detector_without_SR
3	MobileNetV2_face_mask_detector_with_SR
4	MobileNetV2_face_mask_detector_without_SR
5	ResNet50_face_mask_detector_with_SR
6	ResNet50_face_mask_detector_without_SR

3.2.5 Evaluation

The main goal of this study is to identify the deep learning model with the highest accuracy and sensitivity. Therefore, evaluation metrics are needed to properly select the classification problem.

The confusion matrix is used for performance evaluation and is a performance measure of the classification model. The output can be two or more classes by comparing the predicted value with the true known value. A confusion matrix is a 2X2 table of two class models.

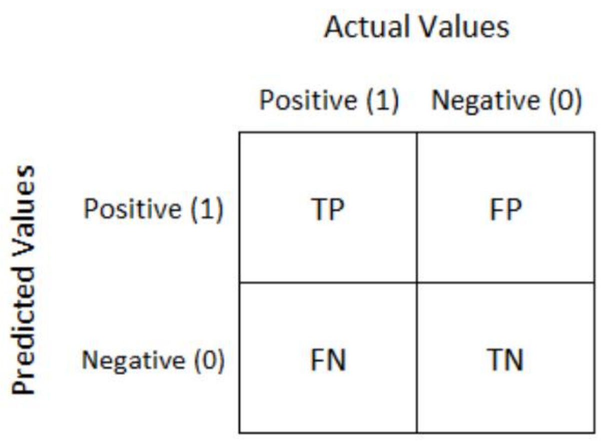


Figure 3.9: Confusion Matrix Table (Agrawal, 2021)

There have 4 possible output of confusion matrix that are two types of errors and two types of correct predictions.

Two types of errors:

(a) FP (False Positive): Predicting positive while the actual value is negative

(b) FN (False Negative): Predicting negative while the actual value is positive

Two types of correct predictions:

(a) TP (True Positive): Predicting positive and the actual value is positive

(b) TN (True Negative): Predicting negative and the actual value is negative

The most common performance metrics to be calculated are as follows: Accuracy, Misclassification, Sensitivity, Specificity, Precision and F1 Score.

$$Accuracy\ (ACC) = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

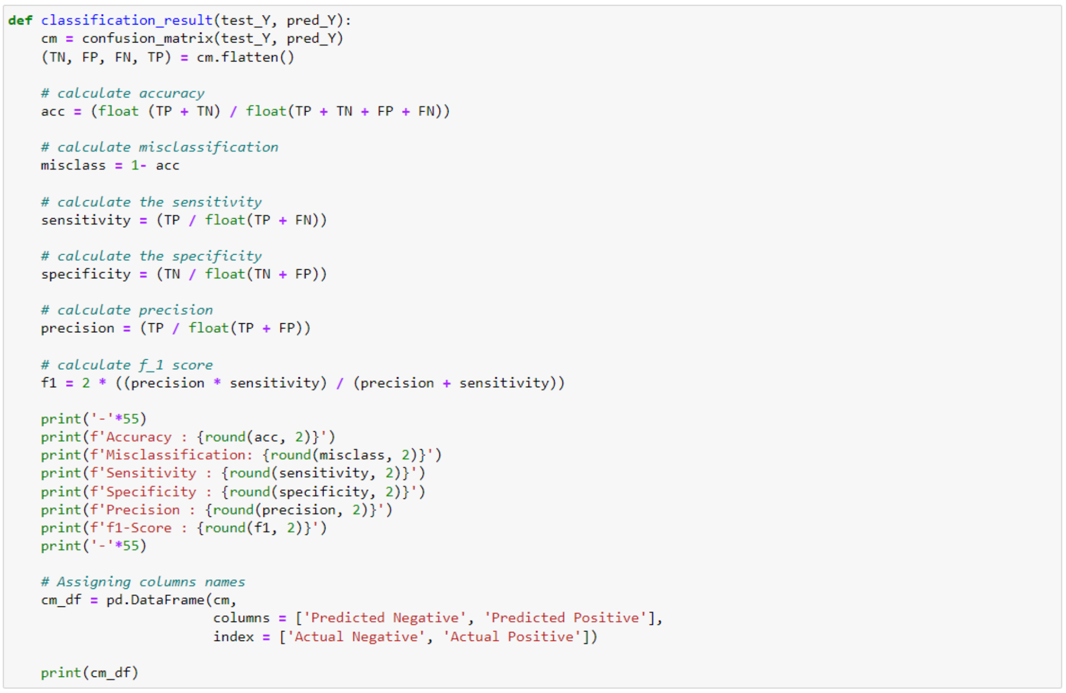
$$Misclassification\ (misclass) = 1 - ACC \tag{2}$$

$$Sensitivity = \frac{TP}{TP+FN} \tag{3}$$

$$Specificity = \frac{TN}{TN+FP} \tag{4}$$

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

$$F1\ Score\ (F1) = 2 * \left(\frac{Precision*Sensitivity}{Precision+Sensitivity}\right) \tag{6}$$



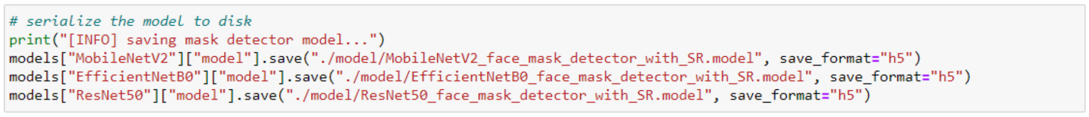
```python
def classification_result(test_Y, pred_Y):
    cm = confusion_matrix(test_Y, pred_Y)
    (TN, FP, FN, TP) = cm.flatten()

    # calculate accuracy
    acc = (float (TP + TN) / float(TP + TN + FP + FN))

    # calculate misclassification
    misclass = 1- acc

    # calculate the sensitivity
    sensitivity = (TP / float(TP + FN))

    # calculate the specificity
    specificity = (TN / float(TN + FP))

    # calculate precision
    precision = (TP / float(TP + FP))

    # calculate f_1 score
    f1 = 2 * ((precision * sensitivity) / (precision + sensitivity))

    print('-'*55)
    print(f'Accuracy : {round(acc, 2)}')
    print(f'Misclassification: {round(misclass, 2)}')
    print(f'Sensitivity : {round(sensitivity, 2)}')
    print(f'Specificity : {round(specificity, 2)}')
    print(f'Precision : {round(precision, 2)}')
    print(f'f1-Score : {round(f1, 2)}')
    print('-'*55)

    # Assigning columns names
    cm_df = pd.DataFrame(cm,
                         columns = ['Predicted Negative', 'Predicted Positive'],
                         index = ['Actual Negative', 'Actual Positive'])

    print(cm_df)
```

Figure 3.10: Confusion Matrix Calculation in Python code in Jupyter

3.2.6 Deployment

One of the best-performing models will be selected from Table 3.3. The model will then be invoked and used to perform mask image classification and detect whether the person is wearing a mask from the video stream.

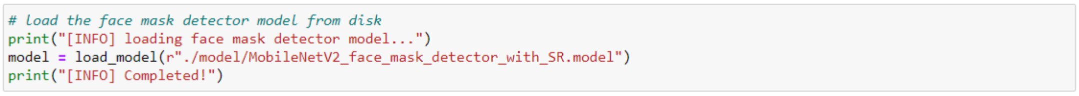
First, we need to save the best-performing model to a specific directory, as shown in Figure 3.11.



```python
# serialize the model to disk
print("[INFO] saving mask detector model...")
models["MobileNetV2"]["model"].save("./model/MobileNetV2_face_mask_detector_with_SR.model", save_format="h5")
models["EfficientNetB0"]["model"].save("./model/EfficientNetB0_face_mask_detector_with_SR.model", save_format="h5")
models["ResNet50"]["model"].save("./model/ResNet50_face_mask_detector_with_SR.model", save_format="h5")
```

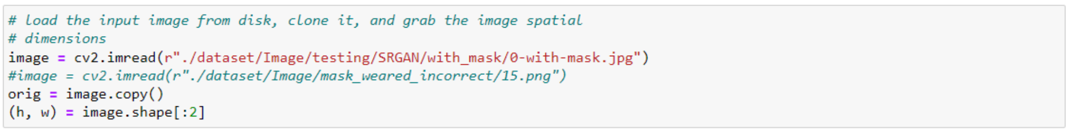
Figure 3.11: Save the Best Model

For mask detection on images, we need to call the best model into the image detection notebook. Then you need to load the image into the notebook and run the image detection model. The image detection model will loop through the image to extract the face region of interest (ROI) and convert it from BGR to RGB channels. The bounding box will be calculated, and the color determined. If the face in the image has a mask, the bounding box will be displayed in green, and if the face in the image has no mask, the bounding box will be displayed in red. After that, it will prompt you with the result. The code and process are shown in Figures 3.12 to 3.14.



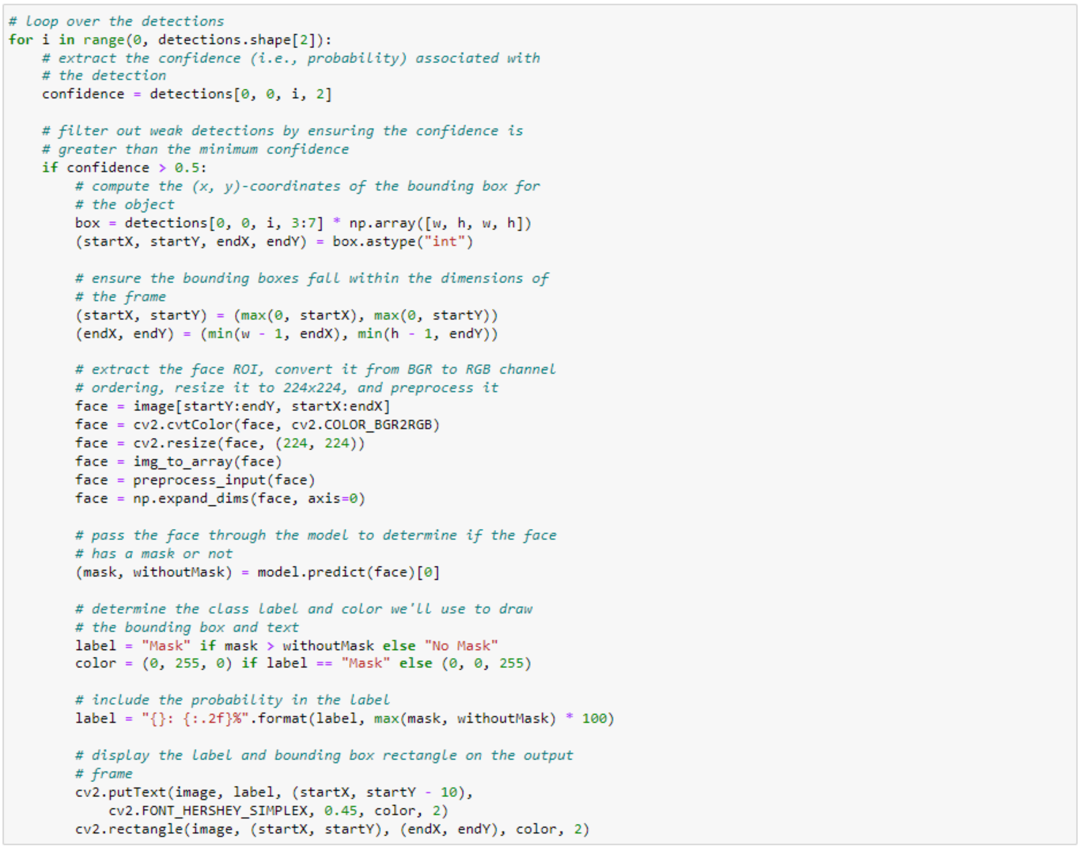
```python
# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
model = load_model(r"./model/MobileNetV2_face_mask_detector_with_SR.model")
print("[INFO] Completed!")
```

Figure 3.12: Load the Best Face Mask Detection Model into the Image Detection Notebook



```python
# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(r"./dataset/Image/testing/SRGAN/with_mask/0-with-mask.jpg")
#image = cv2.imread(r"./dataset/Image/mask_weared_incorrect/15.png")
orig = image.copy()
(h, w) = image.shape[:2]
```

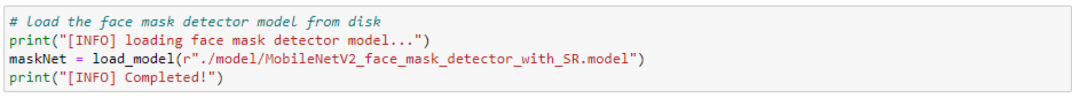
Figure 3.13: Load the Image to be Detected



```python
# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with
    # the detection
    confidence = detections[0, 0, i, 2]

    # filter out weak detections by ensuring the confidence is
    # greater than the minimum confidence
    if confidence > 0.5:
        # compute the (x, y)-coordinates of the bounding box for
        # the object
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        # ensure the bounding boxes fall within the dimensions of
        # the frame
        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

        # extract the face ROI, convert it from BGR to RGB channel
        # ordering, resize it to 224x224, and preprocess it
        face = image[startY:endY, startX:endX]
        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (224, 224))
        face = img_to_array(face)
        face = preprocess_input(face)
        face = np.expand_dims(face, axis=0)

        # pass the face through the model to determine if the face
        # has a mask or not
        (mask, withoutMask) = model.predict(face)[0]

        # determine the class label and color we'll use to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # display the label and bounding box rectangle on the output
        # frame
        cv2.putText(image, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
```

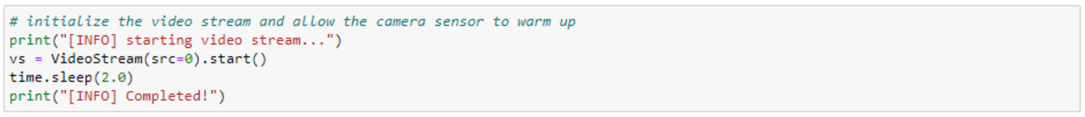
Figure 3.14: Image Detection Model

For mask detection on video, we need to call the best model trained and tested in the previous stage into the video detection model. After that, we will initialize the video stream and let the camera sensor warm up. It will loop the frames in the video stream and display a green or red bounding box along with the percentage on our faces. The percentage is calculated based on the coverage of the face region of interest (ROI). If you are wearing the mask correctly then it will show 100% green and if we are wearing the wrong mask it will show 60% to 70% the border color will change between red and green otherwise it will show on your face showing 100% red bounding box. The Python code and flow of video detection are shown in Figures 3.15 to 3.17.



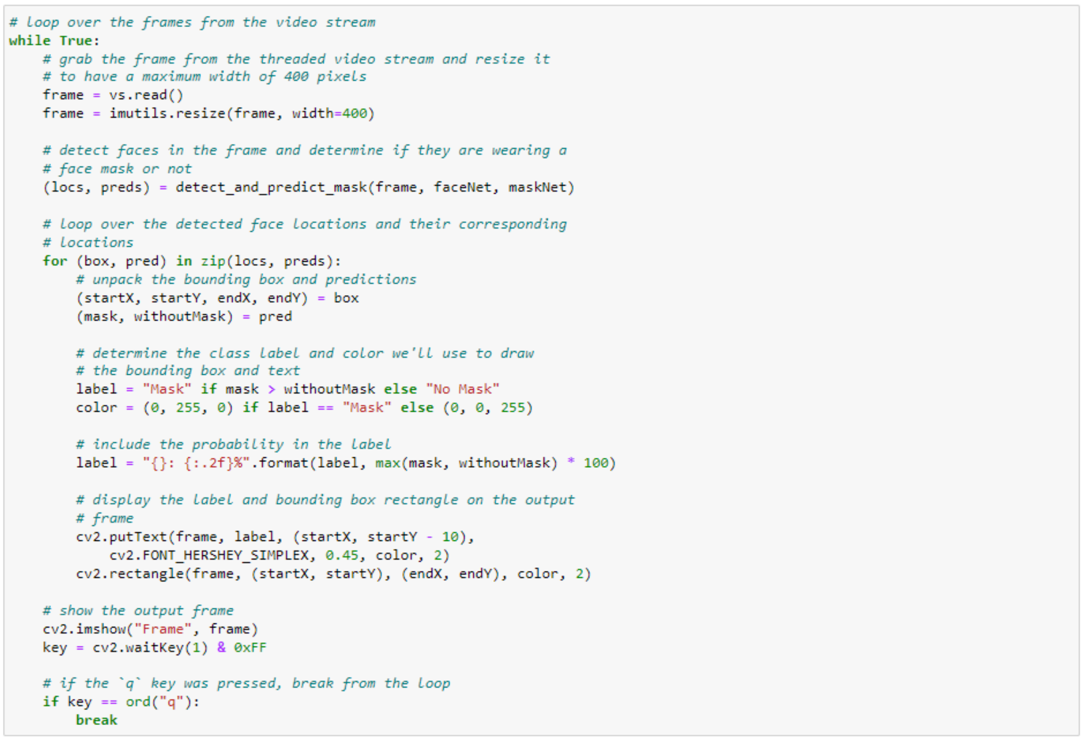
```python
# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
maskNet = load_model(r"./model/MobileNetV2_face_mask_detector_with_SR.model")
print("[INFO] Completed!")
```

Figure 3.15: Load the Best Face Mask Detection Model into the Video Detection Notebook



```python
# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
print("[INFO] Completed!")
```

Figure 3.16: On the Camera Sensor



```python
# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # detect faces in the frame and determine if they are wearing a
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    # loop over the detected face locations and their corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        # determine the class label and color we'll use to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # display the label and bounding box rectangle on the output
        # frame
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break
```

Figure 3.17: Video Detection Mode

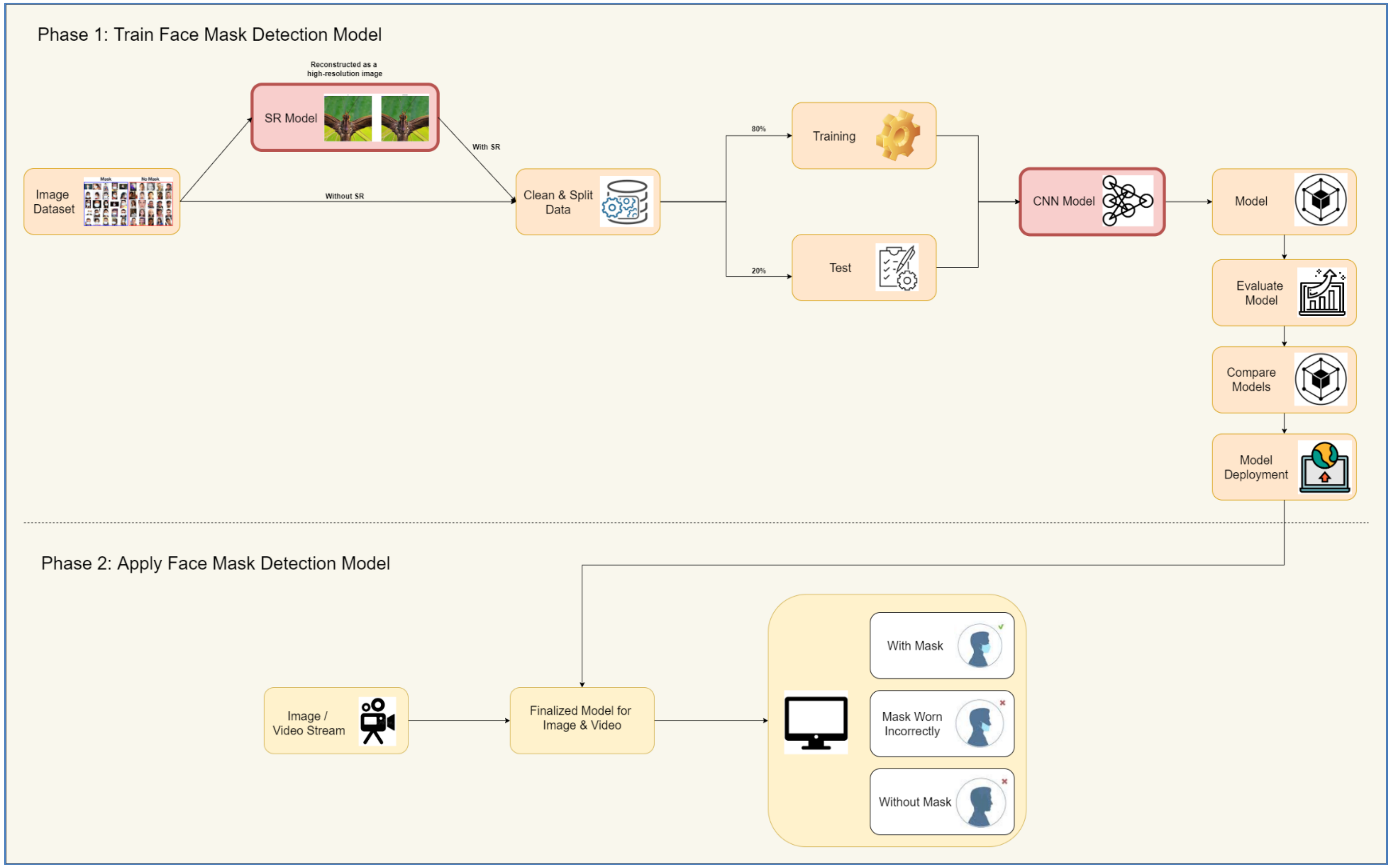


Figure 3.18 General Flowchart of Face Mask Detection System

CHAPTER 4: RESULTS AND DISCUSSION

Find the source code in GitHub:

https://github.com/syncyx/face_mask_detection_system.git

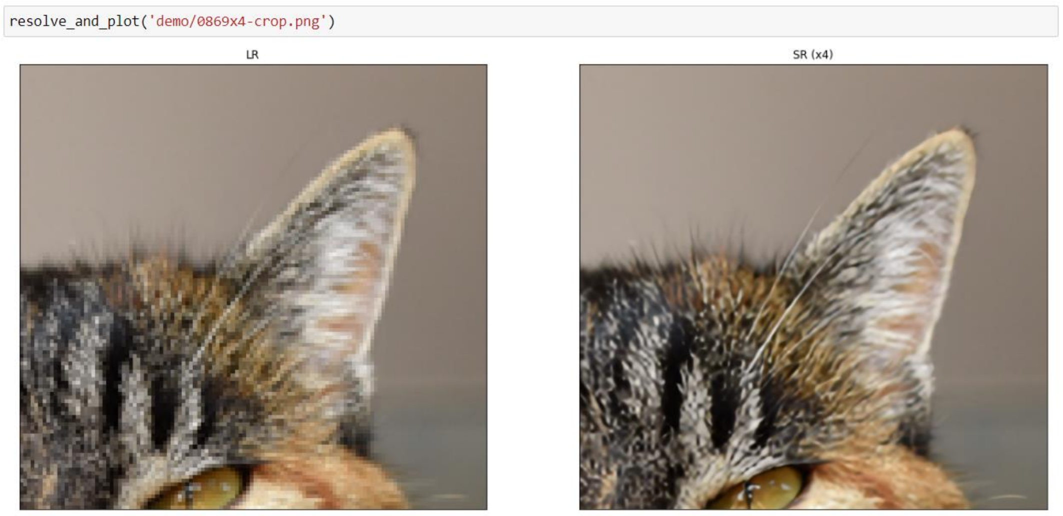
The overall flow chart of the mask detection system is shown in Figure 3.18, which is divided into two phases. Phase 1 is used to train the mask detection model. It starts by collecting an image dataset. The metadata of the image dataset is shown in Table 3.1. The image dataset has a total of 8982 images, which are divided into three categories: wearing a mask, not wearing a mask, and wearing the wrong mask.

Second, a super-resolution (SR) method is used to increase the resolution of the image. Our goal is to develop a high-accuracy mask detection model. Therefore, we need to test whether SR methods can improve the accuracy of the model when extracting face regions of interest (ROI).

Enhanced Depth Super-Resolution (EDSR) improves super-resolution performance and reduces GPU memory usage by 40% by removing batch normalization layers. First, we need to define the number of residual blocks to be 16, the super-resolution factor to be 4, and the downgrade operator to use bicubic. Where the model weights need to be calculated and will be used in the demo section. During training, the Enhanced Deep Super-Resolution model will be trained for 300,000 steps and evaluated every 1,000 steps on the first 10 images of the DIV2K validation set. It saves checkpoints only when the evaluation peak signal-to-noise ratio (PSNR) has improved. We need to save the weights to a separate location and use it later in the demo section. Afterward, for demonstration, we need to define an EDSR model consisting of 16 residual blocks of 4 factors during the super-resolution process. Then, import the DIV2K dataset into the model.

Figures 4.1 to 4.3 show the output of the EDSR model. The left image is the low resolution (LR) and the right image is the super-resolution model (SR). In Figure 4.1, the cat's hair is observed to be very clear, and the colors appear sharper compared to the low-resolution (original) image. The butterfly in Figure 4.2 has become clearer and the texture of the wings is clearly rendered. Compared to the image on the left of the original image in Figure 4.3, the radiance of the lights and the color of the buildings are sharper and brighter.

Therefore, we can conclude that the Enhanced Depth Super-Resolution method successfully reconstructs the image.



```
resolve_and_plot('demo/0869x4-crop.png')
```

Figure 4.1: Image Number 0869 from DIV2K Dataset in EDSR Model

```
resolve_and_plot('demo/0829x4-crop.png')
```

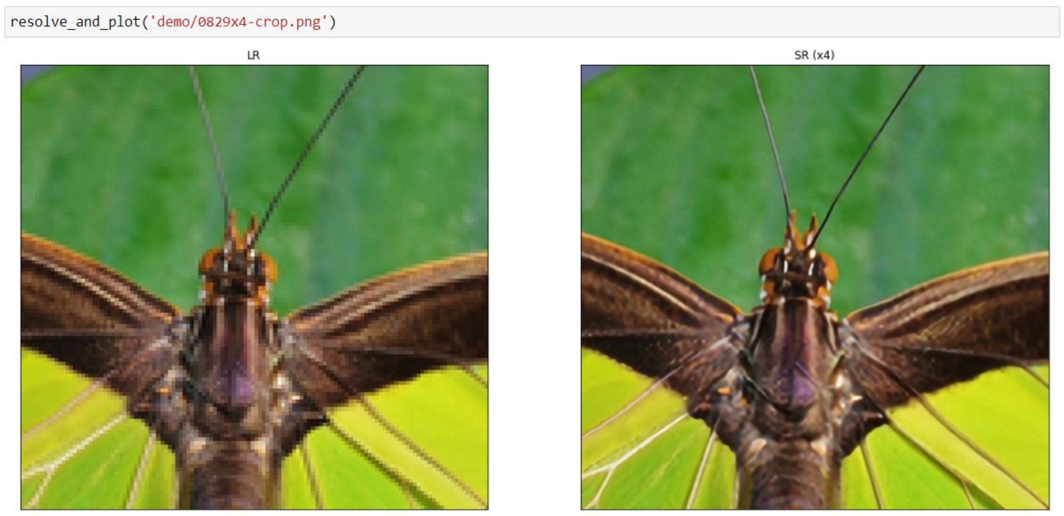


Figure 4.2: Image Number 0829 from DIV2K Dataset in EDSR Model

```
resolve_and_plot('demo/0851x4-crop.png')
```

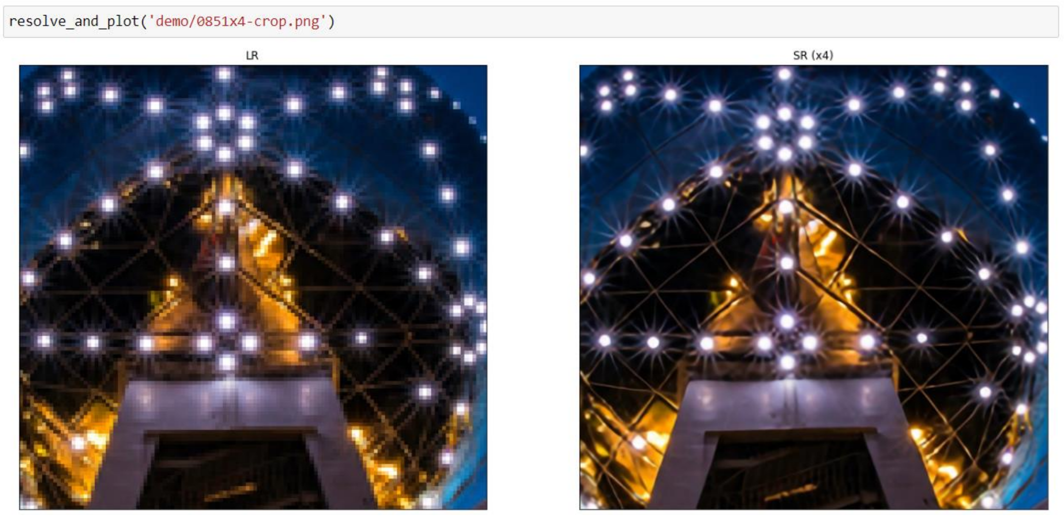
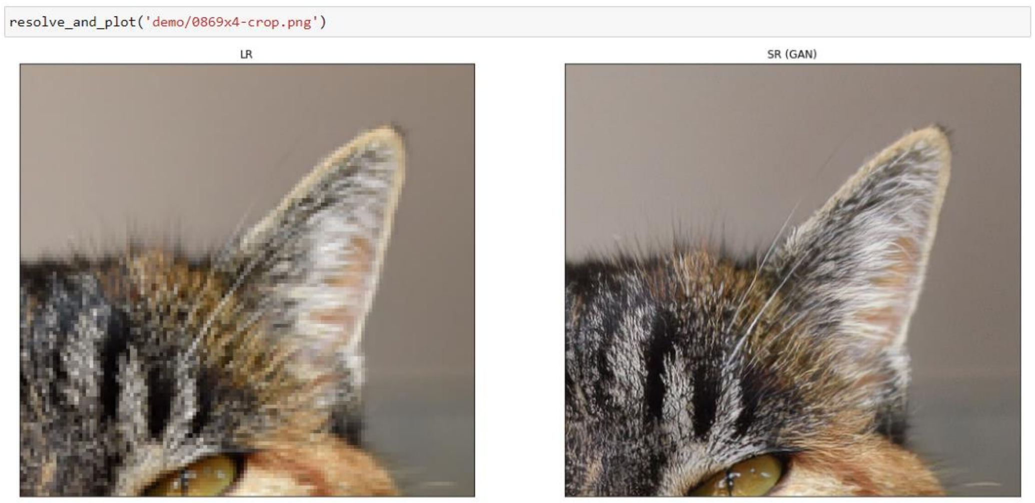


Figure 4.3: Image Number 0851 from DIV2K Dataset in EDSR Model

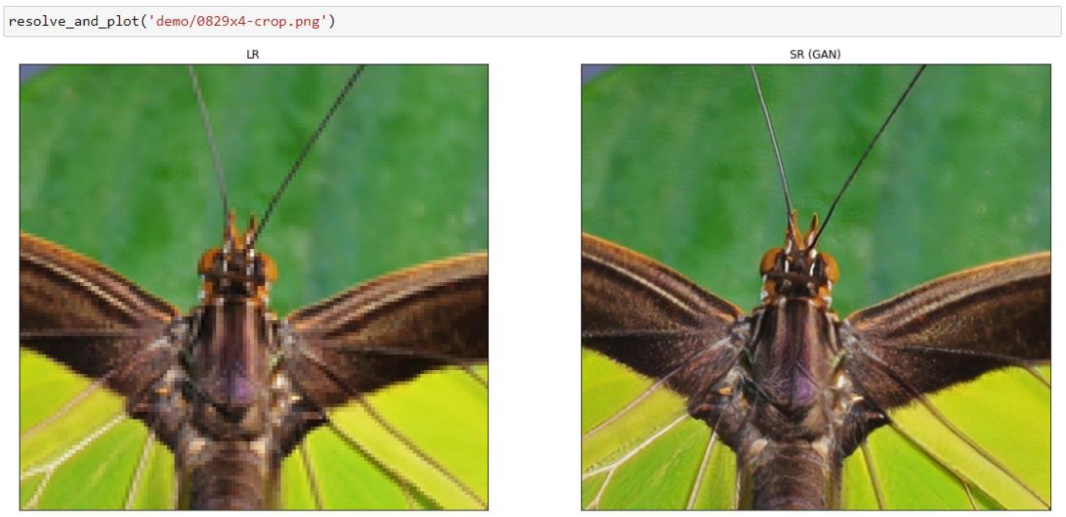
Super-Resolution Generative Adversarial Network (SRGAN) consists of generator networks and discriminator networks that reconstruct images without compromising image quality. In the training of the SRGAN model, we will pre-train the generator by training 1,000,000 steps and evaluate the pre-train model every 1,000 steps on the first 10 images of the DIV2K validation set in the generator pretraining. The weights file will be saved and used for the next fine-tuning of the generator. In generator fine-tuning (GAN), the SRGAN model is tuned for 200,000 steps. Since SRGAN consists of two networks, there are two sets of weights that are the generator set and the discriminator set. Whereas SRGAN will only use weights from the pre-generator and GAN generator in the demo.

Figures 4.4 to 4.6 show the output for images 0869, 0829, and 0851 from the DIV2K dataset. The low-resolution images as raw images are shown on the left, and the high-resolution images passed through the Super-Resolution Generative Adversarial Networks (SRGAN) model are placed on the right in each figure. We can conclude that the Super-Resolution Generative Adversarial Network also successfully improves the resolution of low-resolution images. Since the difference between the original image and the converted image is very noticeable.



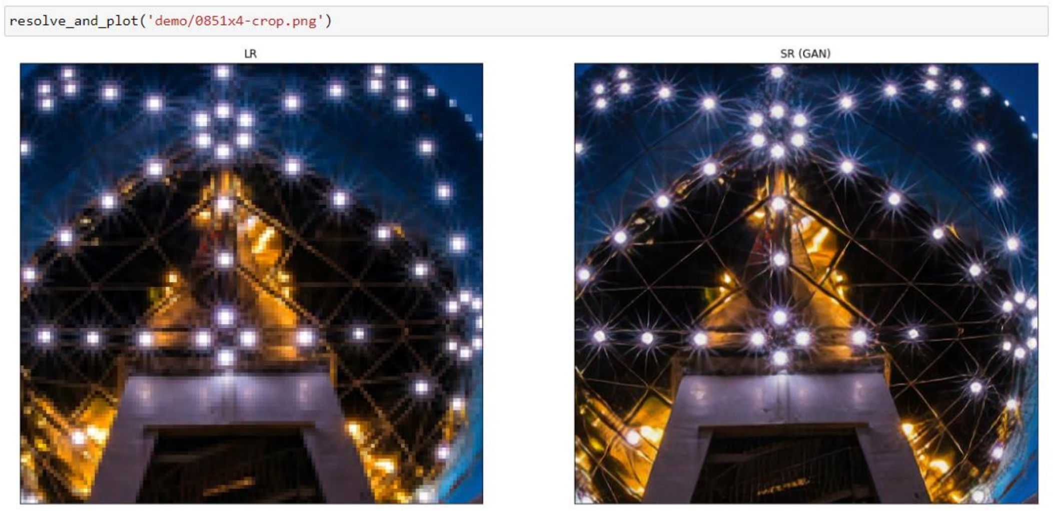
```
resolve_and_plot('demo/0869x4-crop.png')
```

Figure 4.4: Image 0869 from DIV2K Dataset in SRGAN Model



```
resolve_and_plot('demo/0829x4-crop.png')
```

Figure 4.5: Image 0829 from DIV2K Dataset in SRGAN Model



```
resolve_and_plot('demo/0851x4-crop.png')
```

Figure 4.6: Image 0851 from DIV2K Dataset in SRGAN Model

Since both super-resolution models successfully increased the resolution of the images, we will compare the outputs between EDSR and SRGAN. Compared with SRGAN, the cat hair of EDSR is not clear and the color is blurred, as shown in Figure 4.7. Compared with the EDSR in Figure 4.8, the structure of the butterfly and the texture of the leaf in SRGAN show more detail and refinement. Referring to Figure 4.9, the light radiation in SRGAN is clearer and brighter than in EDSR.

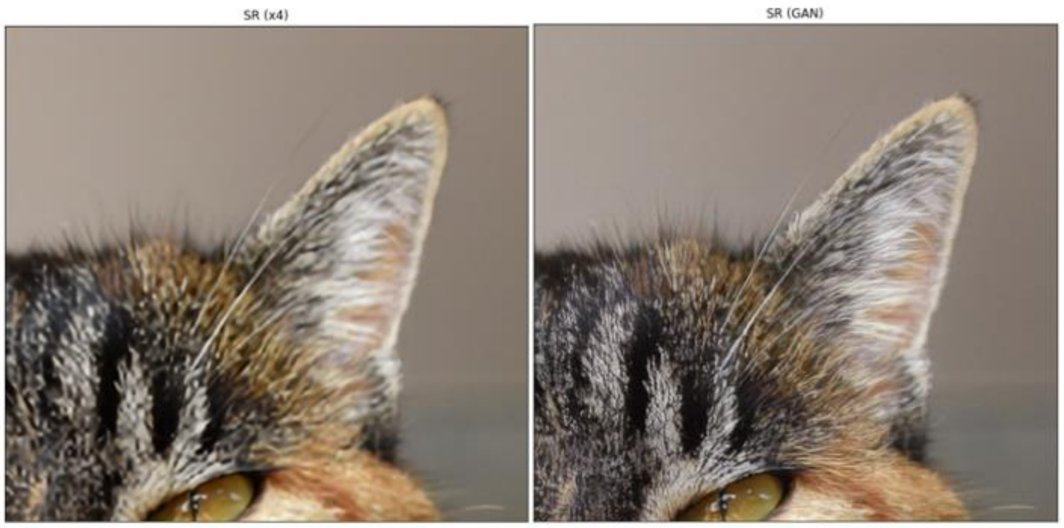


Figure 4.7: Compare Image 0869 Between EDSR and SRGAN

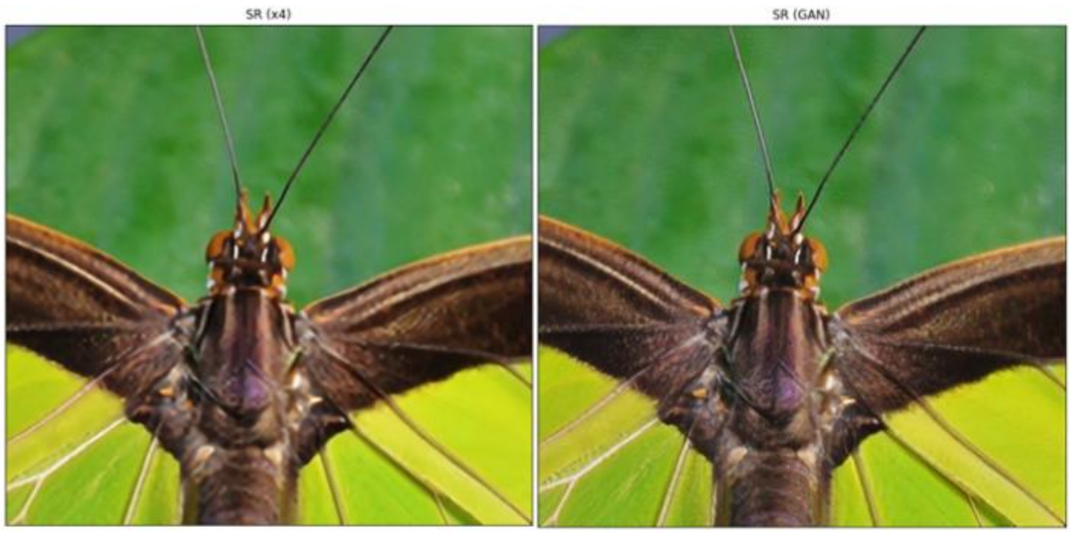


Figure 4.8: Compare Image 0829 Between EDSR and SRGAN

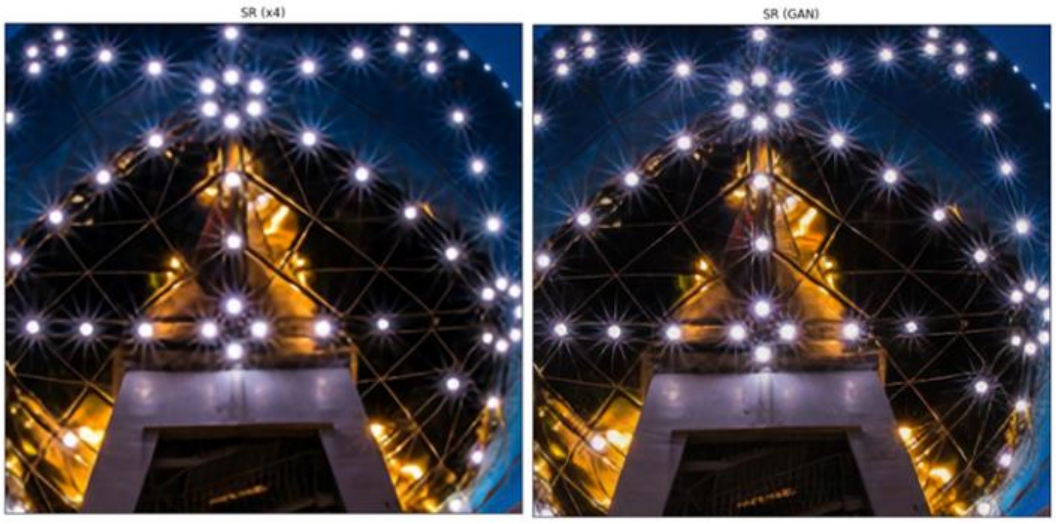
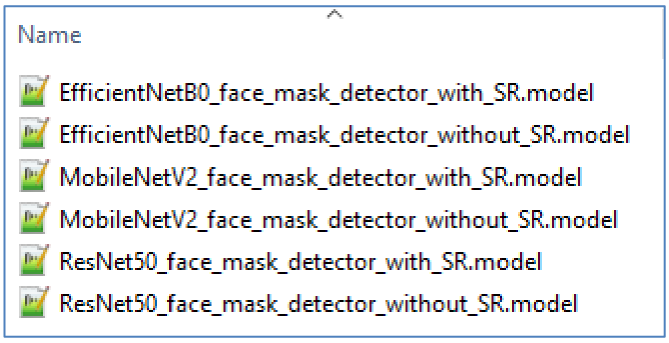


Figure 4.9: Compare Image 0851 Between EDSR and SRGAN

Based on the image comparison between EDSR and SRGAN, SRGAN outperforms EDSR completely. Therefore, SRGAN will be chosen to reconstruct the low-resolution face mask image dataset into a high-resolution face image dataset.

Third, clean and split data. During the cleaning process, we need to remove those corrupted images to ensure our model can train and test successfully. Next, data preprocessing needs to be performed, for example, resizing, normalization and data augmentation in our model. It is important to resize the images to the same size because if the images are of different sizes, it will cause data splitting to fail and be incompatible. Once the dataset is cleaned and preprocessed, we can split it into 80% and 20% as training and test sets respectively.

The next step is to train and test the Convolutional Neural Network (CNN) algorithm using the split data. MobileNetV2 was chosen because its main features prioritize fast inference speed as it was developed with a streamlined architecture, while EfficientNetB0 and ResNet50 prioritize accuracy. Each algorithm has its own advantages. Therefore, we will train and test all three algorithms on low- and high-resolution datasets, resulting in a total of 6 models.


Name

EfficientNetB0_face_mask_detector_with_SR.model
EfficientNetB0_face_mask_detector_without_SR.model
MobileNetV2_face_mask_detector_with_SR.model
MobileNetV2_face_mask_detector_without_SR.model
ResNet50_face_mask_detector_with_SR.model
ResNet50_face_mask_detector_without_SR.model

Figure 4.10: A total of 6 Models were Developed

Moreover, the confusion matrix is used to evaluate the model. In contrast, MobileNetV2 with super-resolution (SR) is the model that outperforms the others. MobileNetV2 + SR has the highest 98% accuracy, 100% specificity, 100% accuracy and 98% F1-Score. Misclassification was also the lowest at 2%. Although the sensitivity of MobileNetV2+SR is not the highest, it reaches 97%, which is a good sensitivity. The model has 6 measurements, with MobileNetV2 + SR scoring the highest out of 5 out of 6. Therefore, MobileNetV2 + SR is the best model. Now, let's see that the least efficient model is EfficientNetB0. It had the highest sensitivity but scored the lowest among the rest of the measurements. Table 4.1 shows all the models and their scores on each measurement and Figure 4.11 shows the accuracy histograms for all models.

Table 4.1: Model Evaluation

	EfficientNetB0	EfficientNetB0 + SR	MobileNetV2	MobileNetV2 + SR	ResNet50	ResNet50 + SR
Accuracy	0.5	0.51	0.98	0.98	0.82	0.83
Misclassification	0.5	0.49	0.02	0.02	0.18	0.17
Sensitivity	1	1	0.97	0.97	0.84	0.81
Specificity	0	0.02	0.99	1	0.8	0.85
Precision	0.5	0.51	0.99	1	0.81	0.84
F1-Score	0.67	0.67	0.98	0.98	0.82	0.83

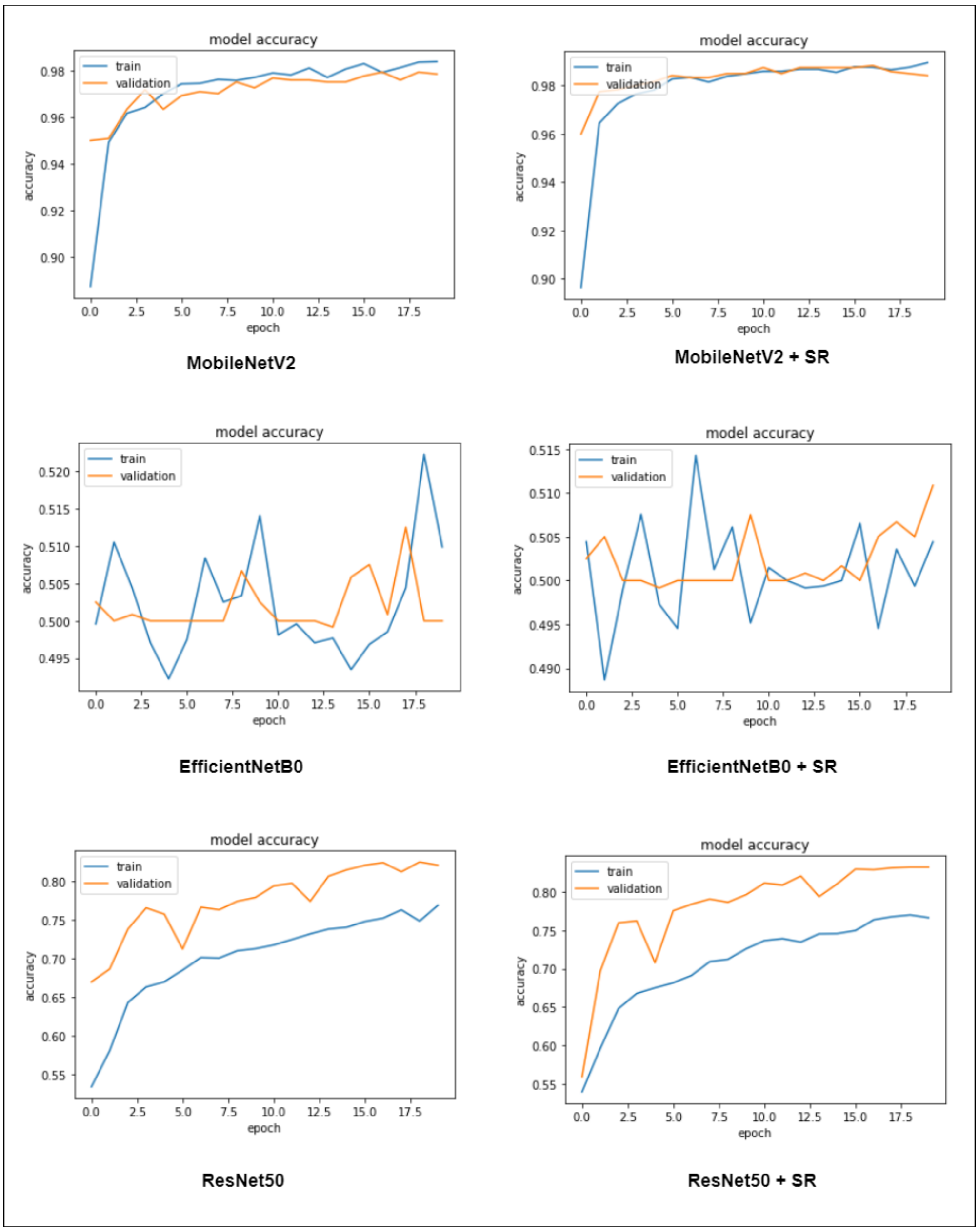


Figure 4.11: Histogram of the 6 Generated Model

To conclude Phase 1, a super-resolution generative adversarial network (SRGAN) approach was used to successfully convert the face mask image dataset to a high-resolution image dataset. Three convolutional neural network (CNN) algorithms are trained and tested on low- and high-resolution image datasets. Using a confusion matrix to measure the model, the MobileNetV2 + SR model is the best model compared to other models. Since the MobileNetV2 + SR model is already deployed, we can call it anytime for mask detection in image and video streams.

Furthermore, we can move to Phase 2: Applying the face mask detection model. By running MobileNetV2 + SR into an image detection model, it loops through images and extracts facial regions of interest (ROIs). It then detects whether the face in the image is wearing a mask. Figures 4.12 and 4.13 show the output of the image detection model.

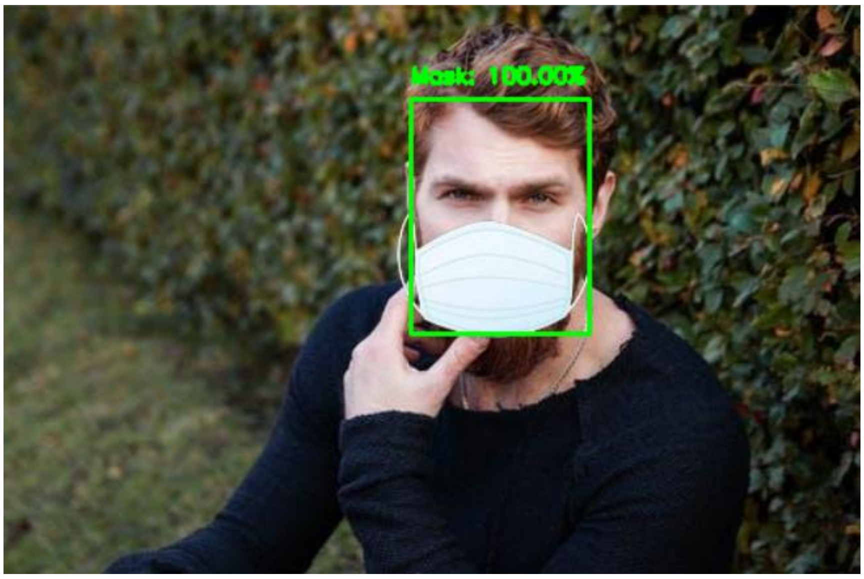


Figure 4.12: Face in the Image with Mask

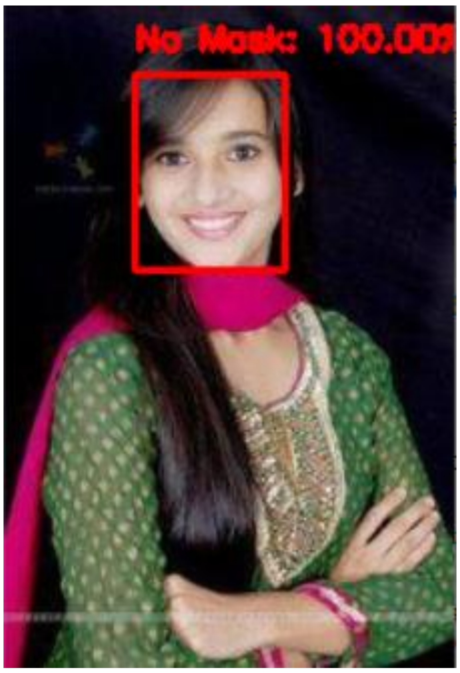


Figure 4.13: Face in Image without Mask

In the video detection model, we only need to call the MobileNetV2+SR model, the camera sensor will be turned on and start extracting the face region of interest (ROI). The results are then displayed directly on the screen as shown in Figures 4.14 to 4.16. Figure 4.14 shows no mask at all, the percentage is 100%, and the bounding box is red. Figure 4.15 shows that the mask is worn incorrectly, then the bounding box is still red, but the percentage drops as part of the face is covered. In Figure 4.16 the bounding box is green, and the percentage is 100%, which means that the person in the video is wearing the mask correctly.

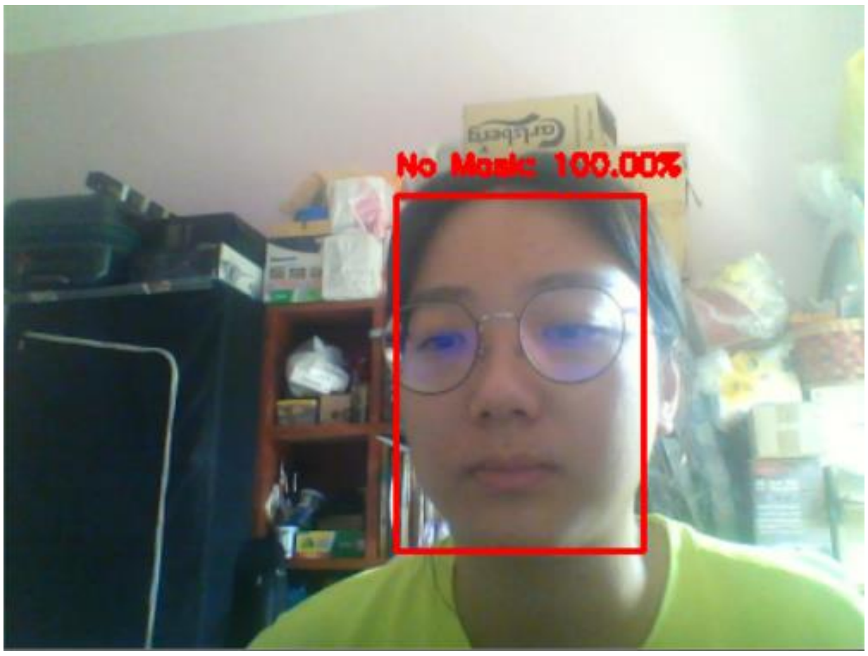


Figure 4.14: Without Mask

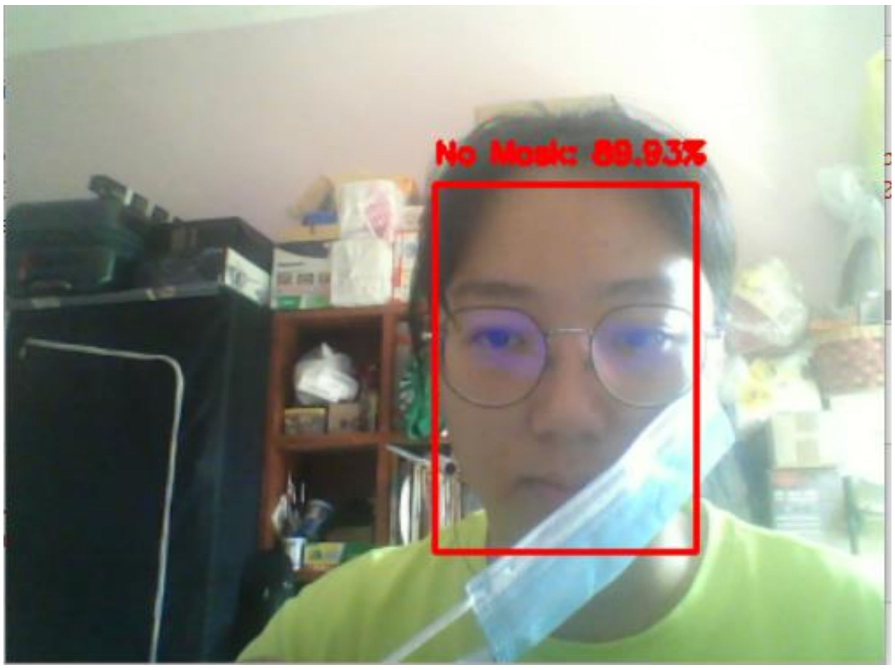


Figure 4.15: Wearing a Mask Incorrectly

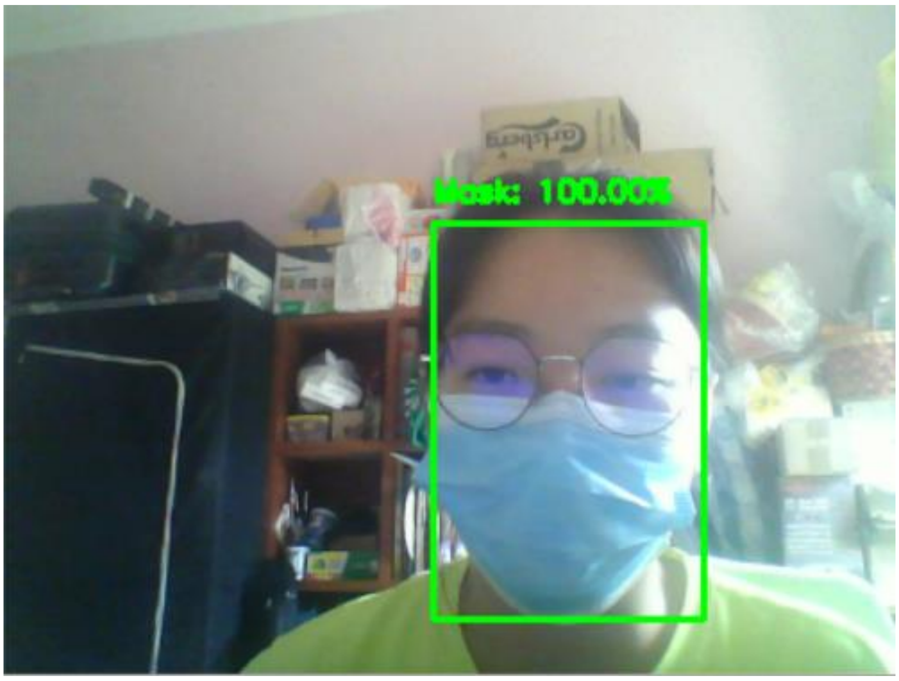


Figure 4.16: With Mask

Therefore, the detection results are displayed accurately in both image and video mask detection systems. After developing image and video streams for a mask detection system. We can say that the video dataset is not necessary to train the model. Based on the procedure in this study, we just trained the model using the image dataset. Once the model is trained and deployed, we can directly invoke the model for real-time mask detection. But if needed, we might try to use the video dataset for model validation.

CHAPTER 5: CONCLUSION

Coronavirus disease (COVID-19) was first detected in China in 2019, and it is now 2022, and the coronavirus is still among us. Malaysia has been on a nationwide lockdown for about two years. The Malaysian government is encouraged to stay at home, wear a mask, maintain a one-meter social distance, and wash your hands frequently. Types of coronaviruses are constantly evolving, and infection rates are rising sharply, such as Omicron. Therefore, we need to always maintain a high level of awareness of the disease, and masks are our last layer of protection.

Based on social needs, image and video mask detection models have been successfully developed. To achieve high accuracy and precision, we train the model using the 3 most efficient convolutional neural network (CNN) algorithms, MobileNetV2, which is built on a streamlined architecture and prioritizes processing speed. In addition to this, the accuracy of the model is also important. EfficientNetB0 and ResNet50 have high performance in terms of computation.

Many mask detection models have been introduced and deployed recently, but they also face some difficulties, such as difficulty in detecting and recognizing blurred faces. To solve this problem, we have done a lot of research on how to increase the resolution of the image, the most important point is the uncompromising image after converting to a high-resolution image.

The main purpose of this study is to investigate super-resolution (SR) methods to improve the resolution of image and video features for face mask detection. We found two effective SR methods that meet our requirements, such as Enhanced Deep Super-Resolution (EDSR) and Super-Resolution Generative Adversarial Networks (SRGAN). Enhanced Depth Super-Resolution refers to ResNet, but it has removed batch normalization from the architecture. Surprisingly, removing batch normalization from the architecture not only improves super-resolution performance but also reduces GPU memory usage by 40%. It also successfully placed first in the NTIRE2017 SR challenge. Therefore, compared to other methods, Enhanced Deep Super-Resolution (EDSR) is the most effective SR method we found. But we are also concerned that details or information in the images may be altered or removed after going through the SR process. Lediger et al. (2017) introduced Super-Resolution Generative Adversarial Networks (SRGAN). The advantage of the SRGAN method is that it does not affect the quality of the image during restoration.

The second goal is to identify a suitable super-resolution method to combine with convolutional neural network algorithms for mask detection. We chose Enhanced Deep Super-Resolution (EDSR) and Super-Resolution Generative Adversarial Networks (SRGAN) from the final stage, which we then need to test with the existing dataset DIV2K. The DIV2K dataset is a low-resolution image dataset. Since we chose EDSR and SRGAN based on the findings of others, we need to test them ourselves to see if it is suitable to implement into our model. Referring to Figures 4.7 to 4.9 comparing the outputs of EDSR and SRGAN, we can clearly see that SRGAN outperforms EDSR in improving image resolution. Therefore, SRGAN was chosen.

The final goal of this study is to evaluate and compare model performance between models with and without super-resolution methods for mask detection. At this stage, we will use SRGAN to reconstruct low-resolution images into high-resolution images. Then, we will have two sets of datasets: the original image dataset and the SR image dataset. All convolutional neural network (CNN) algorithms will be trained on both datasets. In the end, we will have 6 models which will be evaluated using the confusion matrix. The model with the highest score will be the best model.

In conclusion, MobileNetV2 + SR (SRGAN) will be presented, which achieves the highest in terms of accuracy, specificity, precision, and F1-Score, but the model has the lowest sensitivity. One of the future tasks is to increase the sensitivity of the model by compound scaling MobileNetV2 or finding other super-resolution methods or convolutional neural network algorithms.

REFERENCES

Agrawal, S. K. (2021, July 20). *Evaluation metrics for classification model: Classification model metrics*. Analytics Vidhya. Retrieved January 24, 2022, from https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/#:~:text=There%20are%20many%20ways%20for,used%20metrics%20for%20classification%20problems.

Ahmed, Abd El-Aziz & Azim, Nesrine & Mahmood, Mahmood & Alshammari, & Hamoud. (2021, October 13). *A Deep Learning Model for Face Mask Detection*. Research Gate. Retrieved from https://www.researchgate.net/publication/355827770_A_Deep_Learning_Model_for_Face_Mask_Detection

Athirasree Das, Dr.K.S Angel Viji, Linda Sebastian, 2021, A Survey on Image and Video Super Resolution with Deep Learning Models, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCREIS – 2021 (Volume 09 – Issue 13),

Bae, H., Jang, K., & An, Y. K. (2021). Deep super resolution crack network (SrcNet) for improving computer vision–based automated crack detectability in in situ bridges. Structural Health Monitoring, 20(4), 1428-1442.

Basha, C. M. A. K., Pravallika, B. N., & Shankar, E. (2021). An efficient face mask detector with Pytorch and Deep Learning. *EAI Endorsed Transactions on Pervasive Health and Technology*, *7*(25), 167843. https://doi.org/10.4108/eai.8-1-2021.167843

Boesch, G. (2021, August 29). *Deep residual networks (ResNet, RESNET50) - guide in 2022*. viso.ai. Retrieved January 24, 2022, from https://viso.ai/deep-learning/resnet-residual-neural-network/

Dong, C., Loy, C. C., He, K., & Tang, X. (2014, September). Learning a deep convolutional network for image super-resolution. In European conference on computer vision (pp. 184-199). Springer, Cham.

Elengoe, A. (2020). COVID-19 outbreak in Malaysia. Osong public health and research perspectives, 11(3), 93.

Falk, G. (2020). Unemployment Rates During the COVID-19 Pandemic:. Congressional Research Service.

Hassan, M. ul. (2018, November 20). *VGG16 - convolutional network for classification and detection*. VGG16 - Convolutional Network for Classification and Detection. Retrieved January 24, 2022, from https://neurohive.io/en/popular-networks/vgg16/

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Jones, L., Palumbo, D., &amp; Brown, D. (2021, January 24). Coronavirus: How the Pandemic Has Changed the World Economy. Retrieved January 26, 2022, from https://www.bbc.com/news/business-51706225.

Kappeler, A., Yoo, S., Dai, Q., & Katsaggelos, A. K. (2016). Video super-resolution with Convolutional Neural Networks. *IEEE Transactions on Computational Imaging*, *2*(2), 109–122. https://doi.org/10.1109/tci.2016.2532323

Lai, W.-S., Huang, J.-B., Ahuja, N., & Yang, M.-H. (2017). Deep laplacian pyramid networks for fast and accurate super-resolution. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2017.618

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., &amp; Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2017.19

Li, F., Bai, H., & Zhao, Y. (2020). Learning a deep dual attention network for Video Super-Resolution. *IEEE Transactions on Image Processing*, *29*, 4474–4488. https://doi.org/10.1109/tip.2020.2972118

Lim, B., Son, S., Kim, H., Nah, S., &amp; Lee, K. M. (2017). Enhanced deep residual networks for single image Super-Resolution. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw.2017.151

Liu, J., Xue, Y., Zhao, S., Li, S., & Zhang, X. (2020). A convolutional neural network for Image Super-Resolution using internal dataset. *IEEE Access*, *8*, 201055–201070. https://doi.org/10.1109/access.2020.3036155

Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. Sustainable cities and society, 65, 102600.

Loey, M., Manogaran, G., Taha, M. H., & Khalifa, N. E. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement*, *167*, 108288. https://doi.org/10.1016/j.measurement.2020.108288

Lugmayr, A., Danelljan, M., Van Gool, L., & Timofte, R. (2020, August). Srflow: Learning the super-resolution space with normalizing flow. In European Conference on Computer Vision (pp. 715-732). Springer, Cham.

MathWorks. (2021). *Deep Network designer*. MobileNet-v2 convolutional neural network - MATLAB. Retrieved January 24, 2022, from https://www.mathworks.com/help/deeplearning/ref/mobilenetv2.html#:~:text=MobileNet%2Dv2%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals.

MOBILENETV2 classification tensorflow 2 classification model. Roboflow. (n.d.). Retrieved January 24, 2022, from https://models.roboflow.com/classification/mobilenetv2-classification

Page, J., Hinshaw, D., &amp; McKay, B. (2021, February 26). In Hunt for covid-19 origin, patient zero points to second Wuhan Market. The Wall Street Journal. Retrieved January 23, 2022, from https://www.wsj.com/articles/in-hunt-for-covid-19-origin-patient-zero-points-to-second-wuhan-market-11614335404

Park, S. H., Moon, Y. S., & Cho, N. I. (2022). Flexible Style Image Super-Resolution using Conditional Objective. arXiv preprint arXiv:2201.04898.

Perumanoor, T. J. (2021, September 23). *What is VGG16? - introduction to VGG16*. Great Learning. Retrieved January 24, 2022, from https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615

Prajna Bhandary. (2020, April 15). *Prajnasb/observations*. GitHub. Retrieved January 24, 2022, from https://github.com/prajnasb/observations

Qin, B., & Li, D. (2020). Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19. Sensors, 20(18), 5236.

Quantum. (2019, August 20). *Data Science Project Management Methodologies*. Medium. Retrieved January 24, 2022, from https://medium.datadriveninvestor.com/data-science-project-management-methodologies-f6913c6b29eb

Rosebrock, A. R. (2020, May 4). Covid-19: Face mask detector with opencv, Keras/TensorFlow, and Deep Learning. PyImageSearch. Retrieved January 26, 2022, from https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

Sandler, M., & Howard, A. (2018, April 3). *MobileNetV2: The next generation of on-device computer vision networks*. Google AI Blog. Retrieved January 24, 2022, from https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

Sethi, S., Kathuria, M., & Kaushik, T. (2021). Face mask detection using Deep learning: An approach to reduce risk of coronavirus spread. *Journal of Biomedical Informatics*, *120*, 103848. https://doi.org/10.1016/j.jbi.2021.103848

Statistical language - quantitative and qualitative data. Australian Bureau of Statistics. (n.d.). Retrieved January 24, 2022, from https://www.abs.gov.au/websitedbs/D3310114.nsf/Home/Statistical+Language+-+quantitative+and+qualitative+data#:~:text=Quantitative%20data%20are%20measures%20of,and%20are%20expressed%20as%20numbers.&text=Qualitative%20data%20are%20measures%20of,variables%20(e.g.%20what%20type).

Surekcigil Pesch, I., Bestelink, E., de Sagazan, O., Mehonic, A., &amp; Sporea, R. A. (2022). Multimodal transistors as ReLU activation functions in physical neural network classifiers. Scientific Reports, 12(1). https://doi.org/10.1038/s41598-021-04614-9

Swapna. (2022, January 25). *Convolutional Neural Network: Deep learning*. Developers Breach. Retrieved January 29, 2022, from https://developersbreach.com/convolution-neural-network-deep-learning/

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

Tomás, J., Rego, A., Viciano-Tudela, S., & Lloret, J. (2021). Incorrect facemask-wearing detection using convolutional neural networks with transfer learning. *Healthcare*, *9*(8), 1050. https://doi.org/10.3390/healthcare9081050

WHO. (2021, December 9). *Introduction to covid-19: Methods for detection, prevention, response and control*. OpenWHO. Retrieved January 24, 2022, from https://openwho.org/courses/introduction-to-ncov

WHO. (2022). *Coronavirus*. World Health Organization. Retrieved January 24, 2022, from https://www.who.int/health-topics/coronavirus#tab=tab_1

World Health Organization. (2020). (rep.). *Novel Coronavirus (2019-nCoV)*. Retrieved January 26, 2022, from https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200121-sitrep-1-2019-ncov.pdf.