



**NEAR EAST UNIVERSITY**  
**INSTITUTE OF GRADUATE STUDIES**  
**DEPARTMENT OF ELECTRICAL & ELECTRONIC**  
**ENGINEERING**

**MULTIPLE FILTER EMBEDDED CUSTOM CNN**  
**ARCHITECTURE ON OPTIMIZED DETECTION AND**  
**CLASSIFICATION OF FACE MASK**

**PhD THESIS**

**Devrim KAYALI**

**Nicosia**

**January, 2025**

**DEVIRIM KAYALI**

**MULTIPLE FILTER EMBEDDED CUSTOM CNN ARCHITECTURE ON  
OPTIMIZED DETECTION AND CLASSIFICATION OF FACE MASK**

**PhD THESIS**

**2025**

**NEAR EAST UNIVERSITY**  
**INSTITUTE OF GRADUATE STUDIES**  
**DEPARTMENT OF ELECTRICAL & ELECTRONIC**  
**ENGINEERING**

**MULTIPLE FILTER EMBEDDED CUSTOM CNN**  
**ARCHITECTURE ON OPTIMIZED DETECTION AND**  
**CLASSIFICATION OF FACE MASK**

**PhD THESIS**






**Devrim KAYALI**

**Supervisor**  
**Prof. Dr. Kamil DİMİLİLER**


**Nicosia**  
**January, 2025**

## Approval

We certify that we have read the thesis submitted by Devrim Kayalı titled **“MULTIPLE FILTER EMBEDDED CUSTOM CNN ARCHITECTURE ON OPTIMIZED DETECTION AND CLASSIFICATION OF FACE MASK”** and that in our combined opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Examining Committee	Name-Surname	Signature
Head of the Committee:	Prof. Dr. Hasan DEMİREL	
Committee Member:	Prof. Dr. Bülent BİLGEHAN	
Committee Member:	Prof. Dr. Rahib ABİYEV	
Committee Member:	Assoc. Prof. Dr. Cem DİREKOĞLU	
Supervisor:	Prof. Dr. Kamil DİMİLİLER	

Approved by the Head of the Department

  
13./01./2025  
Prof. Dr. Bülent BİLGEHAN  
Head of the Department

Approved by the Institute of Graduate Studies

  
13./01./2025  
Prof. Dr. Kemal Hüsnü Can Başer  
Head of the Institute of Graduate Studies



### **Declaration of Ethical Principles**

I hereby declare that all information, documents, analysis and results in this thesis have been collected and presented according to the academic rules and ethical guidelines of Institute of Graduate Studies, Near East University. I also declare that as required by these rules and conduct, I have fully cited and referenced information and data that are not original to this study.

**Devrim Kayalı**

**10/01/25**

### **Acknowledgments**

I want to thank my family, especially my parents, for their constant support during this time. Even through tough moments, they kept me going. Their unwavering belief in me has been my greatest source of strength. I wish my mom was here to see, I know this achievement would have made her very proud.

A special thanks to my supervisor, Prof. Dr. Kamil Dimililer, for his guidance. His support has made this journey smoother and greatly contributed to my academic growth. I value his mentorship and the ways he has helped me through this academic journey.

I also appreciate the jury members, Prof. Dr. Bülent Bilgehan, Prof. Dr. Hasan Demirel, Prof. Dr. Rahib Abiyev, and Assoc. Prof. Dr. Cem Direkoğlu, for their time and valuable feedback. Their contributions have been important to the development of my work.

To my friends and everyone who supported me, thank you for all your support. Your encouragement and belief in me have been very important.

**Devrim Kayalı**

**Abstract****Multiple Filter Embedded Custom CNN Architecture on Optimized Detection and Classification of Face Mask****Kayali, Devrim****PhD, Department of Electrical & Electronic Engineering****01/2025, 73 pages**

When COVID-19 started to spread rapidly globally, various kinds of face masks were started to be used because one of the effective ways of protection was the proper usage of face masks until the vaccinations were found. In such an environment, it is important to check proper face mask usage to keep the spread under control, which can be effectively done by using automated control systems. This research aims to accurately classify the correct, wrong, and no mask-wearing situations. The used dataset with the three mask-wearing conditions is obtained by adding face masks to the Labeled Faces in the Wild (LFW) dataset. The research contains three main parts. In the first part, deep convolutional neural networks are trained with their minimum possible input data and make a comparison. The second part aimed to use a filter-based approach as a pre-process to extract features and feed them to a customized neural network for classification. Chosen 17 filters were applied to the input images and the network was trained at different learning rates for comparison. Also, an oscillating learning rate was used and compared with the fixed learning rate results. A weight-dropping feature was added to the training process by checking training and validation metrics after each epoch and setting random weights to zero. In the third part, the filtering process was embedded inside a custom network by using convolutional layers and initializing them with selected filter values. They were frozen to prevent them from updating by training. With this approach fixed filters were embedded inside the network architecture, and three versions of the network were implemented with 32x32, 64x64, and 128x128 input image resolutions. The final network sizes were 24.2MB, 60.8MB, and 204MB respectively. The networks were evaluated with their validation and test accuracies. At 32x32, validation and test accuracies were 99.45% and 98.45%. For the second input with 64x64 resolution, the resulting accuracies were 99.84% and 99.43%. When 128x128 input images were used the results were 99.84% and 99.49%.

When the results of the whole research are considered, it was observed that with the fixed filter-based approach, good results can be achieved with a smaller network size than the other models. Also, the results showed that using the oscillating learning rate and the weight-dropping features can effectively shorten the training time and improve the obtained accuracy.

**Key Words:** mask detection, image processing, filtering, custom cnn architecture, training optimization

## Özet

### **Yüz Maskesinin Optimize Edilmiş Algılanması ve Sınıflandırılmasında Çoklu Filtre Gömülü Özel CNN Mimarisi**

**Kayalı, Devrim**

**Doktora, Elektrik ve Elektronik Mühendisliği Anabilim Dalı**

**01/2025, 73 sayfa**

COVID-19 küresel olarak hızla yayılmaya başladığında, aşılara bulunana kadar etkili korunma yollarından biri yüz maskelerinin doğru kullanımı olduğundan çeşitli yüz maskeleri kullanılmaya başlandı. Böyle bir ortamda, yayılmayı kontrol altında tutmak için doğru yüz maskesi kullanımının kontrol edilmesi önemlidir ve bu, otomatik kontrol sistemleri kullanılarak etkili bir şekilde yapılabilir. Bu araştırma, doğru, yanlış ve maske takmama durumlarını doğru bir şekilde sınıflandırmayı amaçlamaktadır. Üç maske takma koşuluna sahip kullanılan veri seti, Labeled Faces in the Wild (LFW) veri setine yüz maskeleri eklenerek elde edilmiştir. Araştırma üç ana bölümden oluşmaktadır. İlk bölümde, derin evrişimli sinir ağları mümkün olan en az giriş verileriyle eğitilir ve bir karşılaştırma yapılır. İkinci bölümde, özellikleri çıkarmak ve sınıflandırma için özelleştirilmiş bir sinir ağına beslemek için ön işlem olarak filtre tabanlı bir yaklaşım kullanılması amaçlanmıştır. Seçilen 17 filtre giriş görüntülerine uygulanmış ve ağ karşılaştırma için farklı öğrenme oranlarında eğitilmiştir. Ayrıca, dalgalanan öğrenme oranı kullanılmış ve sabit öğrenme oranı sonuçlarıyla karşılaştırılmıştır. Eğitim sürecine, her dönemden sonra eğitim ve doğrulama metriklerini kontrol ederek ve rastgele ağırlıkları sıfıra ayarlayarak bir ağırlık düşürme özelliği eklendi. Üçüncü bölümde, filtreleme işlemi evrişimli katmanlar kullanılarak ve seçilen filtre değerleriyle başlatılarak özel bir ağına gömüldü. Eğitimle güncellenmelerini önlemek amacıyla donduruldular. Bu yaklaşımla sabit filtreler ağ mimarisinin içine gömüldü ve ağına üç versiyonu 32x32, 64x64 ve 128x128 giriş görüntü çözünürlükleriyle uygulandı. Son ağ boyutları sırasıyla 24.2 MB, 60.8 MB ve 204 MB oldu. Ağlar doğrulama ve test doğruluklarıyla değerlendirildi. 32x32'de doğrulama ve test doğrulukları %99.45 ve %98.45 idi. 64x64 çözünürlüğe sahip ikinci giriş için ortaya çıkan doğruluklar %99.84 ve %99.43 idi. 128x128 giriş görüntüleri kullanıldığında sonuçlar %99.84 ve %99.49 olmuştur. Tüm araştırmanın sonuçları göz



önüne alındığında, sabit filtre tabanlı yaklaşımla, diğer modellere göre daha küçük bir ağ boyutuyla iyi sonuçlar elde edilebileceği gözlemlenmiştir. Ayrıca, sonuçlar dalgalanan öğrenme oranı ve ağırlık düşürme özelliklerinin kullanılmasının eğitim süresini etkili bir şekilde kısaltabileceğini ve elde edilen doğruluğu artırabileceğini göstermiştir.

***Anahtar kelimeler:*** maske algılama, görüntü işleme, filtreleme, özel cnn mimarisi, eğitim optimizasyonu

## Table of Contents

Approval.....	I
Declaration of Ethical Principles .....	II
Acknowledgments.....	III
Abstract .....	IV
Özet .....	VI
Table of Contents .....	VIII
List of Tables .....	X
List of Figures .....	XI
List of Abbreviations .....	XIII
 <b>CHAPTER I</b> 	
Introduction.....	1
Statement of the Problem .....	3
Purpose of the Study.....	4
Significance of the Study .....	4
Limitations.....	4
Contributions .....	5
 <b>CHAPTER II</b> 	
Literature Review.....	6
Related Research .....	6
 <b>CHAPTER III</b> 	
Methodology .....	16
Dataset .....	16
Modification of Networks .....	19
Preprocessing.....	19
Network Models .....	20
Training .....	23
Customized Network and Training .....	24
Preprocessing.....	24
Network and Training .....	26
Oscillating Learning Rate.....	28

Weight Dropping (Set Zero) Between Epochs .....	28
Embedding Image Processing Filters into Network Architecture .....	30
<b>CHAPTER IV</b>	
Findings and Discussion .....	33
Results For Modified Networks .....	33
Results for Customized Network .....	37
<b>CHAPTER V</b>	
Conclusion and Recommendations .....	48
Conclusion.....	48
Recommendations .....	49
REFERENCES.....	50
APPENDICES .....	61

## List of Tables

	<b>Page</b>
<b>Table 1. <i>Minimum Input Resolutions of the Networks</i> .....</b>	<b>19</b>
<b>Table 2. <i>Input Normalization Requirements of the Networks.</i> .....</b>	<b>20</b>
<b>Table 3. <i>Total Parameters of each Network</i> .....</b>	<b>21</b>
<b>Table 4. <i>Network Performance Results</i> .....</b>	<b>35</b>
<b>Table 5. <i>Results Obtained from the Custom Network with 64x64 images for each learning rate</i> .....</b>	<b>37</b>
<b>Table 6. <i>Comparison of Fixed and Oscillating Learning Rate</i> .....</b>	<b>38</b>
<b>Table 7. <i>Comparison of Fixed and Oscillating Learning Rate with Weight Drop Feature</i> .....</b>	<b>39</b>
<b>Table 8. <i>Results of Filter Embedded Custom CNN Architecture</i> .....</b>	<b>42</b>
<b>Table 9. <i>Performance Comparison of Proposed Network</i> .....</b>	<b>46</b>
<b>Table 10. <i>Comparison with the Recent studies in the Literature</i> .....</b>	<b>47</b>

List of Figures

	Page
<b>Figure 1</b> <i>Test Result of the System Proposed by Yadav (2020)</i> .....	7
<b>Figure 2.</b> <i>Monitoring System of Hussain et al. (2021)</i> .....	10
<b>Figure 3.</b> <i>Cases Used by Aydemir et al. (2022)</i> .....	10
<b>Figure 4.</b> <i>Mask and Distance Detection by Singh et al. (2022)</i> .....	12
<b>Figure 5.</b> <i>System Proposed by Sheikh and Zafar (2023)</i> .....	15
<b>Figure 6.</b> <i>Dataset Preparation</i> .....	17
<b>Figure 7.</b> <i>Mask Correct Class</i> .....	18
<b>Figure 8.</b> <i>Mask Wrong Class</i> .....	18
<b>Figure 9.</b> <i>No Mask Class</i> .....	18
<b>Figure 10.</b> <i>Xception Training Graph</i> .....	23
<b>Figure 11.</b> <i>17 Filters Used</i> .....	24
<b>Figure 12.</b> <i>17 Filters Outputs of Example Mask Correct Class</i> .....	25
<b>Figure 13.</b> <i>17 Filters Outputs of Example Mask Wrong Class</i> .....	25
<b>Figure 14.</b> <i>17 Filters Outputs of Example No Mask Class</i> .....	26
<b>Figure 15.</b> <i>Proposed Custom Network Architecture</i> .....	27
<b>Figure 16.</b> <i>Block Diagram of the Process</i> .....	28
<b>Figure 17.</b> <i>Oscillating Learning Rate</i> .....	28
<b>Figure 18.</b> <i>Training and Validation Accuracy and Loss Graphs (Oscillating Learning Rate)</i> .....	29
<b>Figure 19.</b> <i>Training and Validation Accuracy and Loss Graphs (Oscillating Learning Rate + Weight Dropping)</i> .....	30
<b>Figure 20.</b> <i>Final Custom Network Architecture</i> .....	32
<b>Figure 21.</b> <i>VGG16 Confusion Matrix</i> .....	36
<b>Figure 22.</b> <i>VGG19 Confusion Matrix</i> .....	36
<b>Figure 23.</b> <i>Xception Confusion Matrix</i> .....	36
<b>Figure 24.</b> <i>Validation Loss Graphs of Fixed and Oscillating Learning Rate with Weight Drop Feature</i> .....	40
<b>Figure 25.</b> <i>Validation Accuracy Graphs of Fixed and Oscillating Learning Rate with Weight Drop Feature</i> .....	41

<b>Figure 26. Validation Loss Graphs of Filter Embedded Custom CNN Architecture</b> .....	<b>43</b>
<b>Figure 27. Validation Accuracy Graphs of Filter Embedded Custom CNN Architecture</b> .....	<b>44</b>
<b>Figure 28. Confusion Matrices of Filter Embedded Custom CNN Architecture ...</b>	<b>44</b>

**List of Abbreviations**

<b>AI:</b>	Artificial Intelligence
<b>ANN:</b>	Artificial Neural Network
<b>BPNN:</b>	Back Propagation Neural Network
<b>CNN:</b>	Convolutional Neural Network
<b>CT:</b>	Computed Tomography
<b>DCNN:</b>	Deep Convolutional Neural Network
<b>DCT:</b>	Discrete Cosine Transform
<b>DL:</b>	Deep Learning
<b>FN:</b>	False Negatives
<b>FP:</b>	False Positives
<b>IoT:</b>	Internet of Things
<b>IP:</b>	Image Processing
<b>KNN:</b>	K-Nearest Neighbors
<b>LASSO:</b>	Least Absolute Shrinkage and Selection Operator
<b>ML:</b>	Machine Learning
<b>NLP:</b>	Natural Language Processing
<b>NN:</b>	Neural Network
<b>OCT:</b>	Optical Coherence Tomography
<b>RF:</b>	Random Forest
<b>SVM:</b>	Support Vector Machine

<b>TN:</b>	True Negatives
<b>TP:</b>	True Positives
<b>WxH:</b>	Image Width x Image Height
<b>YOLO:</b>	You Only Look Once



## CHAPTER I

### Introduction

Both artificial intelligence (AI) and image processing (IP) applications have become crucial tools to solve problems in a faster, more efficient, and accurate way. Their speed, sustainability, and reliability are the main points to be used in various applications. **AI and IP** can be used for preliminary examination and information retrieval, assistance, and fully automated systems.

Image processing has different uses so it can be used for multiple purposes according to the need. As a preprocessing step, it can be used for removing background noise, intensity normalization of individual images, and enhancing or removing unwanted data in images before computational processing. Getting rid of any kind of noise before any process is very important since noise can decrease the success rate and cause increased time in calculations. To obtain a better image before any process, some image enhancement steps are used after noise reduction. Histogram equalization, thresholding, and different kinds of filtering techniques are some examples of image enhancement. Applying image enhancement increases the success rate of an operation. Because of this image enhancement is widely used in healthcare applications (Dimililer & Kayali, 2021). Image restoration on the other hand is a technique to estimate the original image which is corrupted with any kind of noise. The aim is that the estimation is as close as the original image. For images having details of different resolutions for different sizes of objects, multiresolution processing is a method used at multiple resolutions to obtain more information about different objects in an image. Image compression is another technique that is used to reduce the amount of data required to represent an image. This is important to process images faster and also efficiently store information. Besides images, it is also important for videos since videos consist of a sequence of multiple images which is also called frames. This means reducing the data required for an image efficiently at this scale saving a lot of space and also means a faster transmission rate. For extracting the interested components or features from an image morphological image processing is used. Dilation and erosion are the basis of morphological image processing and these are also the basis for image segmentation. In image segmentation, the image is divided into regions, or some objects are detected or targeted in the image. This operation can be easy or very hard depending on the task and the risk of the accuracy of the output

of the task. In some cases, a low accuracy may be enough, although in serious cases a very high accuracy can still be not enough.

Artificial intelligence applications are used in many different areas and play an important role in our lives nowadays. Machine learning (ML), neural networks (NN), deep learning (DL), and natural language processing (NLP) are technologies related to AI. Neural networks are formed by copying the human brain structure, while machine learning algorithms use statistical approaches to learn and predict from given data. In deep learning, there are many layers involved in the structure which makes it effective in complex tasks including image and speech recognition. On the other hand, natural language processing makes communication between humans and machines possible so humans can communicate with systems that contain artificial intelligence. AI is used in fields like healthcare, autonomous vehicles, entertainment, and finance. This helps to obtain sustainable, efficient, and minimal errored systems.

Artificial intelligence applications are used for both classification and regression problems. Some applications include fruit classification (Dimililer & Bush, 2017), security technologies (Ilgi et al., 2022), skin lesion classification (Dimililer & Sekeroglu, 2023), and electrochemical determination and classification (Asir et al., 2019; Kayali et al., 2023). When used for regression, the output represents a numerical output rather than a class label like in classification as in the study by Dimililer et al. (2023) 19 years of historical power demand data was used with machine learning regressor models to predict future month power demand (Dimililer et al., 2023).

In AI applications, image processing is used as a preprocessing stage. Dimililer et al. (2016), used image processing on CT images before using back propagation neural networks (BPNN) for lung tumor detection (Dimililer et al., 2016). A detailed review was done by Dimililer et al. (2024) where the image preprocessing phase was applied with artificial intelligence applications on medical images (Dimililer et al., 2024).

Human health can be influenced by many factors, one of them being biological. When COVID-19 was first introduced, various kinds of face masks with different printed figures were designed and sold to help users become familiar with wearing masks and to increase their usage. Because it was seen that one of the most effective ways of slowing down infection and ensuring individual protection is using face masks. Also in such an environment, it is important to check face mask usage in public places since no usage or improper usage can increase the spread.

When the COVID-19 pandemic was spreading quickly all over the world, global-level precautions had to be taken because of its high infection rate and severity. One of the most effective ways of slowing down the infection and individual protection was to use face masks while the vaccination was not found (Cheng et al., 2020). Checking face mask usage in public places was important since no usage or improper usage can increase the spread. Automated control systems are a reliable way of maintaining control. As an example, a smart door project was studied by Baluprithviraj et al. (2021), which was integrated into a mobile app. It detects whether a person is wearing a face mask and sends an alert message to the mobile app (Baluprithviraj et al., 2021). Pandey (2020) also worked on a system that detects whether an individual is wearing a face mask. The system raises an alarm when a person without a mask is detected (Pandey, 2020). A CNN-based gate control system was proposed by Prasad et al. (2022) which opens or closes according to body temperature and face mask detection. Arduino was used for body temperature monitoring while Raspberry Pi was used for face mask monitoring with CNN (Prasad et al., 2022).

Besides mask detection, studies on automatic COVID-19 detection and survival analysis have also been carried out. Narin et al. (2021) used X-ray images and deep CNNs for the automatic detection of COVID-19. Three different binary classifications were implemented to discriminate COVID-19 from normal (healthy) subjects and viral and bacterial pneumonia (Narin et al., 2021). For survival analysis, Atlam et al. (2021) used the Cox regression model and deep learning (Atlam et al., 2021). The usage of face masks also caused conventional face recognition technologies ineffective, so to improve existing methods some research started. According to Wang et al. (2020), face recognition systems are mostly based on deep learning with a large number of face samples (Wang et al., 2020).

### **Statement of the Problem**

Effective infection control and individual protection depend on proper face mask usage in public places. Since manual monitoring requires manpower and is prone to human error, automated control systems are a reliable way of doing this. When this type of global-level problem happens and starts to spread quickly, it can be hard to gather enough data to develop such systems. Also, for deployment in various environments, both highly accurate and resource-efficient solutions are harder to achieve, especially on datasets with more than two classes.

## **Purpose of the Study**

Developing an image processing-based artificial intelligence system can help maintain public safety. Using a fixed filter-based feature extraction approach with a selection of filters can significantly enhance the classification accuracy by a custom network architecture. Also, embedding the image filtering process within the network architecture can streamline image processing, leading to higher classification accuracy and more efficient network performance. The implementation of a custom network architecture with embedded fixed filters can achieve high accuracy while maintaining a small network size, making it practical for deployment in environments with lower specifications. To achieve effective training in a shorter time, applying adjustments between epochs can further improve classification accuracy while reducing the total training time.

## **Significance of the Study**

This study focuses on the accurate classification of three types of face mask-wearing conditions using and modifying convolutional neural networks and filters as image processing. In the first part of the research, deep convolutional neural networks are trained with minimum input data as possible. In the second part, 17 filters are selected to extract features from the images, and a customized network is used for the classification. The filter-based approach is used as a pre-processing phase for feature extraction. In the third part, these filters are embedded inside the network to achieve a custom network architecture that contains the image processing step inside the network itself.

## **Limitations**

Although a variety of masks with different designs and patterns were used in the mask pool to obtain the dataset, there may still be other designs and patterns that can affect the performance obtained in the study. Also, environmental factors that affect the captured image quality can cause performance variations. These limitations can be addressed by fine-tuning or retraining the obtained network and adding image processing for image enhancement. Implementing a fixed filter-based approach may require decent or even advanced knowledge of filter selection and application, especially in more complex situations.

## Contributions

This research focuses on the accurate classification of three face mask-wearing conditions using convolutional neural networks and filters as image processing. To achieve this, 17 filters are selected to extract features from the images and then they are given to a customized network. These filters are also embedded inside the network to achieve a custom network architecture that contains the image processing step inside the network itself. Besides the custom architecture, also training optimization was done to achieve higher accuracy in a shorter training time. All of the obtained results were evaluated and comparisons of networks were made by their performance and efficiency in the means of training times, model sizes, and accuracies.

The thesis is organized into five chapters. The first chapter is the introduction which gives an overview, which is then followed by the statement of the problem, purpose of the study, significance of the study, limitations, and contributions. The second chapter is the literature review, which includes relevant research in the literature. The third chapter explains the methodology by describing the dataset, modifications, and networks used, and the proposed customized network with the training procedure. In the fourth chapter, the obtained results are given and discussed. In the fifth chapter, the conclusion of the research is given with some recommendations for possible future directions of the study.

Throughout the research, several publications were made with the findings, which reflects the progressive development of the research (Dimililer & Kayali, 2023; Kayali et al., 2021; Kayali & Dimililer, 2023). Contributions of the thesis can be summarized as:

- Creating a three-class face mask dataset
- Classification of the three face mask-wearing conditions
- Training existing deep convolutional neural networks with minimum input resolution
- A fixed filter-based feature extraction approach with the chosen filters
- Improved training by learning rate changes and weight-dropping features
- Embedding the filtering process inside a network to obtain a custom architecture
- Development of a resource-efficient and accurate network

## CHAPTER II

### Literature Review

This chapter includes related research in the literature which explains existing methods used and gives information about the recent advances in the field that are related to image processing, filtering, and face mask classification with convolutional neural networks.

#### Related Research

For face mask detection in public areas, Suresh et al. (2021) studied a MobileNet network. If a face mask is not detected, the authorities are notified and the face is captured (Suresh et al., 2021).

A CNN-based real-time face mask detector was proposed by Chavda et al. (2021). The first stage involved face detection, and in the second stage, NasNetMobile, DenseNet121, and MobileNetV2 networks were trained. NasNetMobile network was selected as the most suitable network for real-time application according to the experimental results in terms of both accuracy and average inference time (Chavda et al., 2021).

Mohan et al. (2021) proposed a tiny CNN architecture to be used on devices with constrained resources. The proposed architecture was compared to the SqueezeNet and the Modified SqueezeNet. The proposed method outperformed the SqueezeNet and the Modified SqueezeNet in both size and accuracy according to the results (Mohan et al., 2021).

Snyder and Husari (2021) developed an automated detection model “Thor” and implemented it on a mobile robot. The detection of human existence, the detection and extraction of the human face, and the classification of masked/unmasked faces are the three components of the method. ResNet50 is used with Feature Pyramid Network (FPN) in the first component. The second component detects and extracts human faces with multitask convolutional neural networks (MT-CNNs). The last component is a trained CNN classifier for the detection phase (Snyder & Husari, 2021).

Khamlae et al. (2021) trained a CNN with a dataset consisting of 848 images with a 416x416 resolution. 81% accuracy was obtained with the model, which was then implemented at the front gate of a campus building (Khamlae et al., 2021).

For face mask detection Kodali and Dhanekula (2021) proposed a deep learning-based network that consists of two parts. The first part, the pre-processing step, converts the RGB image to grayscale and resizes and normalizes it. Then, the proposed CNN classifies the face images with and without face masks. The accuracy of the proposed model was 96% (Kodali & Dhanekula, 2021).

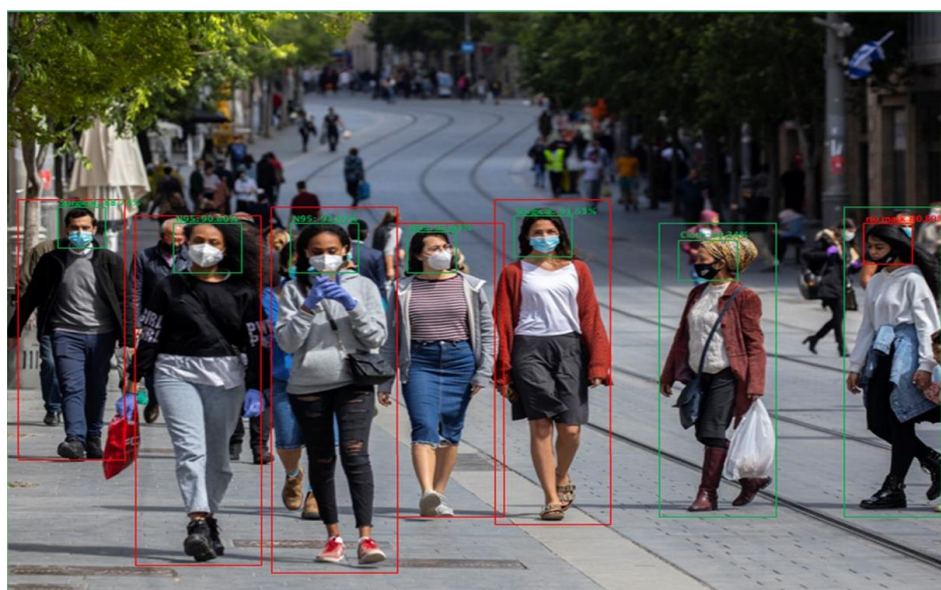
By fine-tuning MobileNetV2, Pinki and Garg (2020) developed a deep-learning model for a face mask detection system. The developed model can be used in public areas to detect if people are wearing masks. If a person is not wearing a mask, a notification is sent to the authorities (Pinki & Garg, 2020).

By building four fully connected layers on top of MobileNetV2, Sen and Patidar (2020) developed a deep learning-based method. The developed model detects people with or without face masks from images and video streams with an accuracy of 79.24% (Sen & Patidar, 2020).

A system was proposed by Yadav (2020) that uses both deep learning and geometric techniques for detection, tracking, and validation purposes. Real-time monitoring is performed to detect face masks and social distancing between people. Figure 1 shows a test result from the research. (Yadav, 2020).

Figure 1

*Test Result of the System Proposed by Yadav (2020)*



Srinivasan et al. (2021) proposed a system that uses Dual Shot Face Detector (DSFD), Density-based spatial clustering of applications with noise (DBSCAN), YOLOv3, and MobileNetV2 for an effective solution regarding face mask and social distancing detection. According to an evaluation of its performance, the system obtained accuracy and F1 scores of 91.2 and 90.79%, respectively (Srinivasan et al., 2021).

Wang et al. (2021) proposed a face mask detection solution dubbed Web-based efficient AI recognition of masks (WearMask). It is an in-browser, serverless, edgecomputing- based application that requires no software installation and can be used by any device with an internet connection and access to web browsers. This framework integrates deep learning (YOLO), a high-performance neural network inference computing framework (NCNN), and a stack-based virtual machine (WebAssembly) (Wang et al., 2021).

Dey et al. (2021) proposed a multi-phase deep learning based model for face mask detection in images and video streams, which was named MobileNet Mask. According to their experimental results, MobileNet Mask achieved about 93% accuracy with 770 validation samples, and about 100% with 276 validation samples (Dey et al., 2021).

Naufal et al. (2021) conducted a comparative study on face mask detection using support vector machines (SVM), k-nearest neighbors (KNN), and deep CNNs (DCNN). Although CNN required a longer execution time compared to KNN and SVM, it reported the best average performance, with an accuracy of 96.83% (Naufal et al., 2021).

Vijitkunsawat and Chantngarm (2020) also conducted a comparative study on the performance of SVM, KNN, and MobileNet for real-time face mask detection. According to the results, MobileNet has the best accuracy with regard to both images and real-time video inputs (Vijitkunsawat & Chantngarm, 2020).

Nagrath et al. (2021) proposed SSDMNv2, an approach based on OpenCV and deep learning. In this method, a Single Shot Multibox Detector is used as a face detector, and the MobilenetV2 network is used as a classifier. The accuracy and F1 score of the proposed system are 92.64 and 93%, respectively (Nagrath et al., 2021).

Sanjaya and Rakhmawan (2020) also used MobileNetV2 to develop a face mask detection algorithm in order to fight COVID-19. Their model can discriminate



people with and without a face mask with an accuracy of 96,85% (Sanjaya & Rakhmawan, 2020).

Sakshi et al. (2021) also used MobileNetV2 to implement a face mask detection model to be used both on static pictures and real-time videos (Sakshi et al., 2021).

Venkateswarlu et al. (2020) used MobileNet and a global pooling block to develop a face mask detection model. The global pooling layer flattens the feature vector, and a fully connected dense layer and the softmax layer utilize classification. The system was evaluated with different datasets, and the obtained accuracies were 99 and 100% (Venkateswarlu et al., 2020).

Rudraraju et al. (2020) proposed a face mask detection system to control various entrances of a facility using fog computing. Fog nodes are used for processing the video streams of the entrances. Haar-cascade-classifiers are used to detect faces in the frames, and each fog node consists of two MobileNet models. The first model classifies whether the person is wearing a mask, and, if this model gives the output that the person is wearing a face mask, then the second model classifies whether the mask-wearing condition is proper. As a result of this two-level classifier, the person is allowed to enter the facility if they are properly wearing a mask. The accuracy of this system is about 90% (Rudraraju et al, 2020).

Hussain et al. (2021) proposed a Smart Screening and Disinfection Walkthrough Gate (SSDWG), an IoT-based control and monitoring system. This system performs rapid screening, includes temperature measuring, and stores the record of suspected individuals. The screening system uses a deep learning model for face mask detection and the classification of people who are wearing face mask their properly, improperly, or wear no mask (Figure 2). A transfer learning approach was used on CNN, ResNet-50, Inceptionv3, MobileNetV2, and VGG-16. Results showed that VGG-16 achieved the highest accuracy, with 99.81%, followed by the MobileNetV2 with 99.6% (Hussain et al., 2021).

Figure 2.

*Monitoring System of Hussain et al. (2021)*



Adusumalli et al. (2021) studied a system that detects whether people are wearing face masks. The system is based on OpenCV and MobileNetV2. It also sends message to the person if they are not wearing a mask, and their face is stored in the database (Adusumalli et al., 2021).

Das et al. (2020) used OpenCV and CNNs in their research to develop a face mask detection system. They used two different datasets, and the obtained accuracies were 95.77 and 94.58% (Das et al., 2020).

Aydemir et al. (2022) used pre-trained DenseNet201 and ResNet101 for feature extraction. Then, they used an improved ReliefF selector to choose features in order to train an SVM classifier. They had three cases in their research: mask vs. no mask vs. improper mask; mask vs. no mask and improper mask; and mask vs. no mask (Figure 3). They obtained 95.95%, 97.49%, and 100% accuracy for these cases, respectively (Aydemir et al., 2022).

Figure 3.

*Cases Used by Aydemir et al. (2022)*



Wu et al. (2022) proposed an automatic facemask recognition and detection framework called FMD-Yolo. ImRes2Net-101 was used to extract features, which is a modified Res2Net structure. Then, an aggregation network En-PAN was used to merge low-level details and high-level semantic information. According to their experimental results, at the intersection over union (IoU) level 0.5, FMDYolo had average precisions of 92 and 88.4% for two different datasets (Wu et al., 2022).

Mar-Cupido et al. (2022) conducted a study on classifying face masks used by people according to their type. They used the pre-trained models ResNet101v2, ResNet152v2, MobileNetv2, and NasNetMobile to classify the classes KN95, N95, cloth, surgical, and no mask. According to their results, ResNet101v2 and ResNet152v2 both had higher accuracies (98 and 97.37%, respectively) in comparison with NasNetMobile and MobileNetv2, both with 93.24% accuracy (Mar-Cupido et al., 2022).

Agarwal et al. (2022) proposed a hybrid model with CNNs and SVMs. They used the medical mask dataset (MDD) and the real-world masked face recognition dataset (RMFD) for training and evaluating the proposed model. The obtained accuracy was 99.11% (Agarwal et al., 2022).

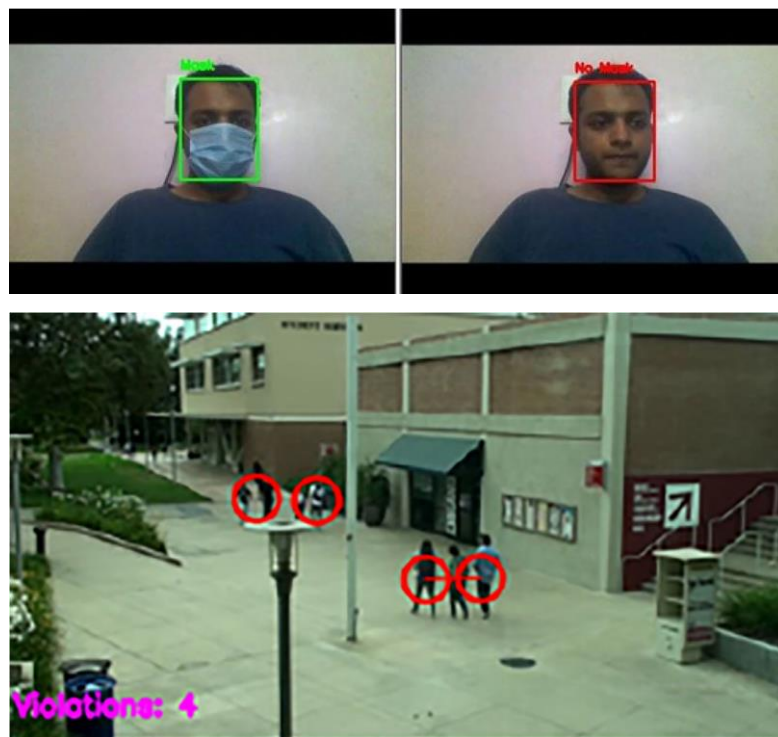
Crespo et al. (2022) carried out a study on both two and three-class face mask-wearing conditions using deep learning while focusing on ResNets. The results showed that ResNet-18 had the best accuracies for both two and three classes, with 99.6% and 99.2% respectively (Crespo et al., 2022).

Jayaswal and Dixit (2022) proposed a framework that uses a Single Shot Multibox Detector and Inception V3 (SSDIV3). They also proposed two versions of synthesized face mask datasets and compared SSDIV3 against VGG16, VGG19, Xception, and MobileNet. Their experimental results showed the higher accuracy of their proposed system, with 98% (Jayaswal & Dixit, 2022).

Singh et al. (2022) proposed a framework that uses cloud computing, fog computing, and deep neural networks. It is a smart and scalable system that can detect mask-wearing and distancing violations (Figure 4). When the proposed framework was tested on video, it achieved a 98% accuracy (Singh et al., 2022).

Figure 4.

*Mask and Distance Detection by Singh et al. (2022)*



A quarter Laplacian filter was proposed by Gong et al. (2021) which is 2x2 instead of the classical 3x3 Laplacian filter which works more locally and can preserve the corners in the image while smoothing. They used the filter in some image processing tasks such as image smoothing, low-light image enhancement, and texture enhancement to show the edge-preserving advantage (Gong et al., 2021).

Methil (2021) used various image processing methods on brain images and investigated their impact on the dataset for brain tumor detection with a convolutional neural network. The dataset consisted of different tumor sizes, textures, shapes, and locations, and an accuracy of 99.73% was obtained on the validation data (Methil, 2021). Ahmad and Dimililer (2022) also used convolutional neural network in their study for brain tumor detection (Ahmad & Dimililer, 2022).

Tayal et al. (2022) proposed a deep learning-based framework that can detect ocular diseases. Four classes which are normal, choroidal neovascularization, drusen, and diabetic macular edema were detected from optical coherence tomography (OCT) scans. Image preprocessing was also involved to remove noise, enhance contrast, do contour-based edge detection, and extract retinal layers. Three different convolutional neural networks were used for detection which had five, seven, and nine layers. The

resulting accuracy was 96.5% while having 96% sensitivity and 98.6% specificity (Tayal et al., 2022).

Jasti et al. (2022) used machine learning and image processing for the detection and classification of breast cancer using the MIAS data collection. The geometric mean filter was used for image enhancement while AlexNex was used for the feature extraction. After using the relief algorithm for feature selection, Naïve Bayes, KNN, random forest (RF), and least square support vector machine algorithms were used for detection and classification (Jasti et al., 2022).

Viknesh et al. (2023) used their designed CNN, AlexNet, VGG-16, and LeNet from existing CNNs and support vector machines (SVM) for the detection and classification of melanoma skin cancer as normal, malignant, and benign. According to the results, CNN obtained 91% of accuracy while the SVM classifier obtained 86.6%. By using Android Studio and Django, the designed CNN model was also deployed for mobile and web applications (Viknesh et al., 2023).

Hammad et al. (2023) proposed a hybrid approach in their study which includes image processing, feature extraction with CNN, feature selection, and machine learning. They used genomic image processing to convert human coronavirus (HCoV) genome sequences to genomic grayscale images by the frequency chaos game representation mapping technique. Then they used AlexNex to extract features from these images from the last convolution layer and the second fully connected layer. Feature selection was done by using ReliefF and the least absolute shrinkage and selection operator (LASSO). For classification, k-nearest neighbors (KNN) and decision trees were used. According to their results, the best hybrid combination was extracting features from the second fully connected layer, using the LASSO algorithm for feature selection, and using KNN as a classifier. This approach obtained 99.71% accuracy, 99.78% specificity, and 99.62% sensitivity (Hammad et al., 2023).

Umer et al. (2023) built a public dataset named RILFD for face mask detection by using a camera and annotating them with the two classes, with mask, and without mask. In their study, they used machine learning models, YOLOv3, Faster R-CNN, and a customized CNN with a four-step image processing for the classification. According to their results, the proposed method obtained better accuracy on the RILFD, and two other public datasets MOXA and MAFA with a 97.5% accuracy (Umer et al., 2023).

Dong et al. (2022) studied generating raw monochrome images from raw colored images, and low-light image enhancement. For the first task, they used deep neural networks based on De-Bayer-Filter simulator. For the low-light image enhancement, they proposed a fully convolutional network. For training, they proposed a dataset called Mono-Colored Raw paired dataset (MCR) which is collected by a color camera with Bayer-Filter and a monochrome camera without Bayer-Filter (Dong et al., 2022). Lv et al. (2022) proposed an adaptive bilateral filter that combines bilateral filtering and the edge detection operator to enhance infrared images. This method acts as an improved convolutional kernel for bilateral filtering and enhances the details in infrared images while suppressing the noise (Lv et al., 2022). For edge detection, Siddharth et al. (2021) mentioned that using Kalman Filter with ANN has lower calculation rates and quicker merging. They used ANN for object localization and Kalman filtering on obtained object coordinates which lowered the localization error distances and improved localization accuracy (Siddharth et al., 2021). Schmalfluss et al. (2023) studied blind image inpainting using directional filters. They used a dictionary of filters and combined them with the trainable weights of a lightweight network. Their approach had faster network convergence and improved inpainting quality (Schmalfluss et al., 2023). Dasari and Reddy (2023) applied Gabor filter to the public ILD (interstitial lung diseases) dataset to obtain texture-enhanced input images. Then they gave these input images to a Multi-scale Convolution Neural Network (M-CNN) and obtained 90.67% accuracy on lung tissue classification (Dasari & Reddy, 2023). Putra et al. (2023) studied a face mask detection system on CCTV that sends notifications on a mobile application. They used MobileNetV2 and stated that using a distance of one meter gave good results in their experiments (Putra et al., 2023). Sheikh and Zafar (2023) proposed a system called RRFMDS (Rapid Real-Time Face Mask Detection System). They fine-tuned MobileNetV2 for classification while using single-shot multi-box detector for face detection. The system detects three classes which are incorrect mask, with mask, and without mask (Figure 5). According to their results, training accuracy was 99.15% and the test accuracy was 97.81% (Sheikh & Zafar, 2023).

Figure 5.

*System Proposed by Sheikh and Zafar (2023)*



## CHAPTER III

### Methodology

Deep learning refers to using artificial neural networks (ANNs) with many layers. Since these networks have many layers, they can learn complex patterns on data which makes them effective in applications of image recognition tasks. Convolutional Neural Networks (CNNs) are a type of these models that are widely used on image-related applications because of their effectiveness in processing image data and extracting features. Layers generally included in CNNs are convolutional layers, pooling layers, flattening layers, and fully connected dense layers. Convolution layers are used to apply filters to the images by sliding over, applying convolution operation, which extracts features from the image. Pooling layers are used to reduce the spatial dimensions of the extracted feature maps, which reduces the computational load. Fully connected layers are used to make classification according to the extracted features. After the features are extracted using convolutional and pooling layers, they are flattened using a flattening layer and fed into the fully connected dense layers. Generally, these layers are used with the ReLU (Rectified Linear Unit) activation function. If the task is classification, the final layer of the network is a dense layer with units equal to the number of output classes and the “softmax” activation function. This activation function gives an output of probabilities of each class, which is then used to obtain the output class using the highest probability.

The goal of the study is to obtain an accurate face mask detection model that classifies three mask-wearing conditions. To achieve this, a custom three-class dataset was obtained to be used in this research. Then this dataset was used to obtain results from modifying existing networks, and secondly, a customized network was created to obtain results for comparison. This chapter provides detailed information about obtaining the dataset, how the existing networks were modified, and how the customized network was created and trained.

#### Dataset

The dataset used to train our networks is prepared by using the existing dataset called Labeled Faces in the Wild (LFW) (Huang et al., 2007). Originally, this dataset has face images of different people and each of them is labelled and categorized into folders with their names. But the reason for selecting this dataset is not the labelling,



the good amount of face images were suitable because the purpose was adding face masks on face images to obtain 3 classes (no mask, mask wrong and mask correct) of images required for our training process. By using OpenCV's face detection algorithm face borders in images were found and cropped. Then again OpenCV's eye detection algorithm was applied to these images to locate the eye positions. These eye positions were used as reference points for calculating the insertion point and scaling of the masks. The mask is selected randomly from a mask pool which consists of PNG images of masks that differ in design and pattern. To obtain the desired dataset, this process was applied to the images to insert face masks correctly to simulate the correct way of wearing, at a lower position to simulate the wrong mask-wearing situation, and left untouched for the no mask-wearing situation (Figure 6). With this whole preparation process, 7892 images with 3 classes of mask-wearing conditions were obtained with 2479 images for mask correct class, 2520 images for mask wrong class and 2893 images for no mask class. Figure 7, Figure 8, and Figure 9 shows examples from the obtained dataset. Obtained dataset was also used in other studies for three class mask detection and classification (Dimililer & Kayali, 2023; Kayali et al., 2021; Kayali et al., 2022; Kayali & Dimililer, 2023).

Figure 6.

*Dataset Preparation*

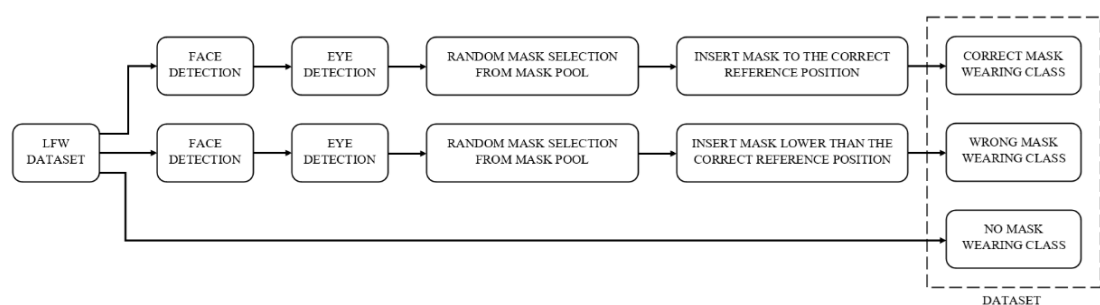


Figure 7.

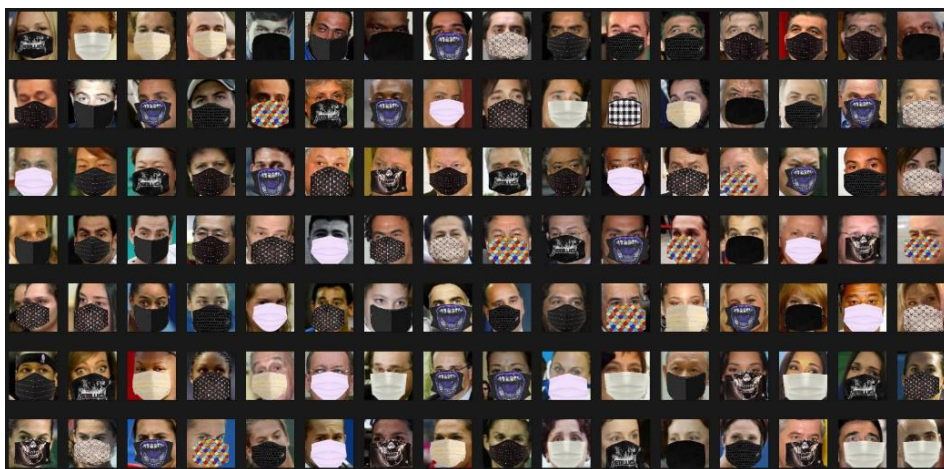
*Mask Correct Class*

Figure 8.

*Mask Wrong Class*

Figure 9.

*No Mask Class*

## Modification of Networks

### *Preprocessing*

The pre-processing stage was applied differently for each network depending on their properties. Each of the selected networks has a default input resolution, and some of them also need the input data to be scaled between specific values for training. Since the goal was to train the network with as little data as possible, the minimum input resolutions of each network were examined as shown in Table 1. The second most important thing was the rescaling of the input data in order to meet the network requirements. To this effect, each network's input pixel value requirements were found (Table 2). Finally, the image was converted to grayscale in order to train with only one channel. Although these network inputs are generally three-channel images, grayscale images were used to reduce the input data. To summarize: first, the input image was converted to grayscale; then, the input image was resized to the minimum input size of the network; and the input data were rescaled according to the network's input scale requirements. For networks with no input scale requirements, the input data were scaled between 0 and 1.

Table 1.

*Minimum Input Resolutions of the Networks*

<b>Network</b>	<b>Minimum input resolution</b>
DenseNet 121-169-201	32x32
EfficientNet B0-B7	32x32
InceptionResNetV2	75x75
InceptionV3	75x75
MobileNet - MobileNetV2	32x32
NasNetMobile - NasNetLarge	32x32
ResNet 50-101-152	32x32
ResNet 50V2-101V2-152V2	32x32
VGG16	32x32
VGG19	32x32
Xception	71x71

Table 2.

*Input Normalization Requirements of the Networks.*

*(-) Means that there is no Requirement*

<b>Network</b>	<b>Input Normalization Requirement</b>
DenseNet 121-169-201	-
EfficientNet B0-B7	between 0 and 255
InceptionResNetV2	between -1 and 1
InceptionV3	between -1 and 1
MobileNet - MobileNetV2	between -1 and 1
NasNetMobile - NasNetLarge	-
ResNet 50-101-152	-
ResNet 50V2-101V2-152V2	between -1 and 1
VGG16	-
VGG19	-
Xception	between -1 and 1

### ***Network Models***

The networks used for this study are MobileNets (Howard, 2017; Sandler et al., 2018), EfficientNets (Tan & Le, 2019), ResNets (He et al.,2016; He et al.,2016), DenseNets (Huang et al., 2017), NasNets (Zoph et al., 2018), InceptionResNetV2 (Szegedy et al., 2017), InceptionV3 (Szegedy et al., 2016), Xception (Chollet, 2017), VGG16 (Simonyan & Zisserman, 2014), and VGG19 (Simonyan & Zisserman, 2014). The total parameters of each network are given in Table 3.

Regarding the DenseNets, the models used in this research were DenseNet121, DenseNet169, and DenseNet201. The default input resolution of these networks is (224, 224, 3), which means the expected input, is a three-channel image of size 224x224. These networks can be trained with smaller images, but the size should not be smaller than 32x32.

The EfficientNet models used in this research were EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, and EfficientNetB7. The inputs expected by EfficientNet networks are the float pixel values of the images between 0 and 255. The minimum image size supported by the EfficientNets is also 32x32.

Table 3.

*Total Parameters of each Network*

<b>Network</b>	<b>Total Parameters</b>
DenseNet121	7.034.307
DenseNet169	12.641.603
DenseNet201	18.321.475
EfficientNetB0	4.052.834
EfficientNetB1	6.578.502
EfficientNetB2	7.772.216
EfficientNetB3	10.787.422
EfficientNetB4	17.678.334
EfficientNetB5	28.518.806
EfficientNetB6	40.966.046
EfficientNetB7	64.104.214
InceptionResNetV2	54.340.771
InceptionV3	21.808.355
MobileNet	3.231.363
MobileNetV2	2.261.251
NasNetMobile	4.272.311
NasNetLarge	84.927.189
ResNet50	23.587.587
ResNet101	42.658.051
ResNet152	58.370.819
ResNet50V2	23.564.675
ResNet101V2	42.626.435
ResNet152V2	58.331.523
VGG16	33.608.387
VGG19	38.918.083
Xception	20.867.051

The InceptionResNetV2 model has some differences when compared to EfficientNets and DenseNets. This network expects a default input resolution of (299, 299, 3), and the input data need to be normalized between -1 and 1. 75x75 is the minimum image size accepted by this network.

InceptionV3 network has a default input resolution of (299, 299, 3), and the input data normalization requirement is between -1 and 1, like InceptionResNetV2. InceptionV3 has a minimum input image size of 75x75.

The MobileNet and MobileNetV2 networks have a default input resolution of (224, 224, 3), and the minimum supported image size is 32x32, like DenseNets. However, these networks require the input data to be normalized between -1 and 1.

NasNets includes NasNetLarge and NasNetMobile, whose input requirements are just like DenseNets, with a default input resolution of (224, 224, 3) and a minimum image size of 32x32. As their names explain, NasNetLarge is a larger network, with more parameters to train, and NasNetMobile is the lightweight version of the network, with fewer parameters to train that can however be implemented on more devices.

ResNet networks can be grouped into two: ResNets and ResNetV2s. ResNets have a default input resolution of (224, 224, 3), and the minimum input size is 32x32. ResNetV2s also have the same properties, but their input needs to be normalized between -1 and 1. This research used ResNet50, ResNet101, ResNet152, ResNet50V2, ResNet101V2, and ResNet152V2.

VGG16 and VGG19 are also networks that have a minimum input size of 32x32 and a default input resolution of (224, 224, 3). The last network used was Xception, which has a minimum input size of 71x71 and a default input resolution of (299, 299, 3). This network also requires the input data to be normalized between -1 and 1.

All of the networks described above have the option of loading pre-trained weights from a path or from ImageNet. However, in this research, pre-trained weights were not used, which means that the weights were randomly initialized.

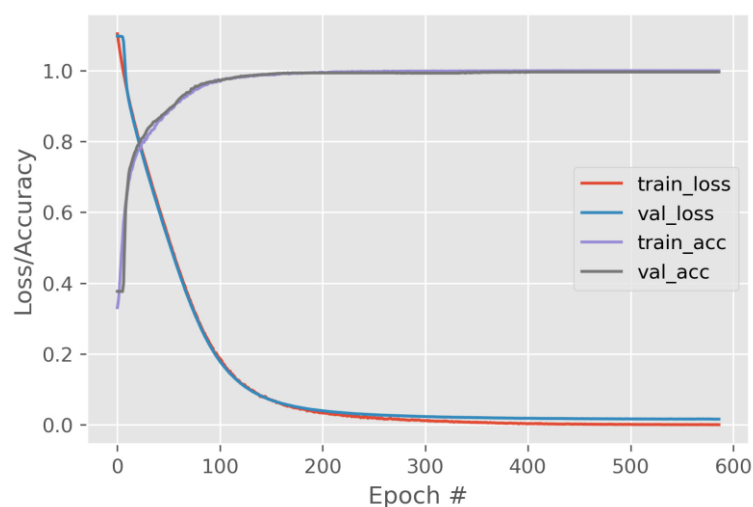
### *Training*

For the training process, training parameters were selected after several experiments and used for all the networks. A value of patience parameter was determined in order to track the learning process of the networks, aiming to make sure that the learning process continued as long as the model continued to improve. To this effect, the patience value was set at 100 epochs by experimenting, and validation loss was tracked during the whole training process of each network. This means that each network continued with the training process as long as the validation loss of the network kept decreasing (Figure 10). In other words, the network continued to learn. However, if the network stopped learning and the validation loss did not decrease for the selected patience value of 100 epochs, the training process would automatically stop, and the best weights of the training process would be restored before saving the model.

For comparison, the training of all networks was carried out with the same computer, which had 32GB DDR4 system memory, a 10th Gen. Intel Core i7-10750H processor, and an NVIDIA GeForce RTX 2070 SUPER graphics card with 8GB GDDR6 video memory. All of the code implementations were done by using Python programming language, and Spyder which is an open-source integrated development environment (IDE).

Figure 10.

*Xception Training Graph*



## Customized Network and Training

### *Preprocessing*

Before feeding the images to the convolutional neural network, image preprocessing was applied. This process was done by applying various 3x3 image filters to the images and feeding all of the output images to the network at once. To extract various features from the image, 17 different filters were chosen and applied. The idea behind choosing these filters was to have multiple filters that put forward different directional and regional details. By leaving out the center pixel, surrounding values were adjusted to keep up with different details from the image. Then the center pixel value was set so that all the numbers in the filter sum up to 0, otherwise loss of details would occur. The chosen filters are shown in Figure 11. The steps included in the whole image preprocessing phase are, resizing, converting images to grayscale, and filtering step. After the last step custom filtering, the resulting input becomes 17 images which are obtained by applying 17 different 3x3 filters to the images. Examples of images obtained from each class are given in Figure 12, Figure 13, and Figure 14.

Figure 11.

#### *17 Filters Used*

$$\begin{array}{cccc}
 \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
 \begin{bmatrix} 1 & 1 & 1 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 1 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & -3 & 1 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 1 & -3 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\
 \begin{bmatrix} 1 & 1 & 0 \\ 1 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 1 \\ 0 & -3 & 1 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 1 \\ 0 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 1 & -3 & 0 \\ 1 & 1 & 0 \end{bmatrix} \\
 \begin{bmatrix} 1 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 1 \end{bmatrix} & \begin{bmatrix} -1 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & -1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}
 \end{array}$$



Figure 12.

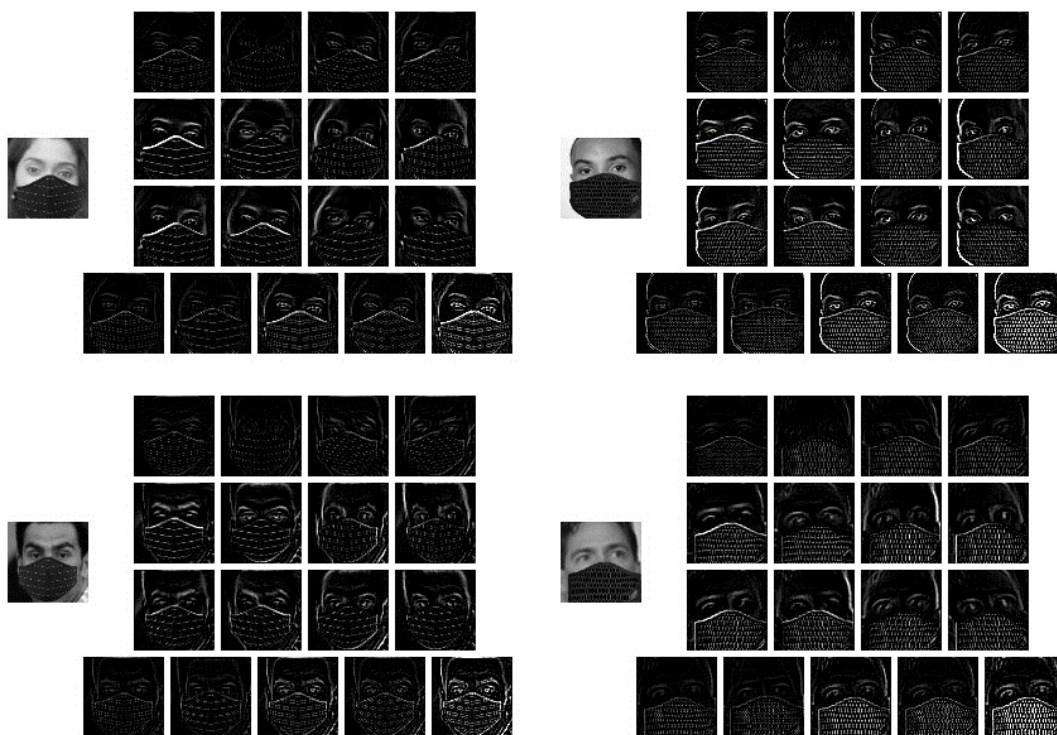
*17 Filters Outputs of Example Mask Correct Class*

Figure 13.

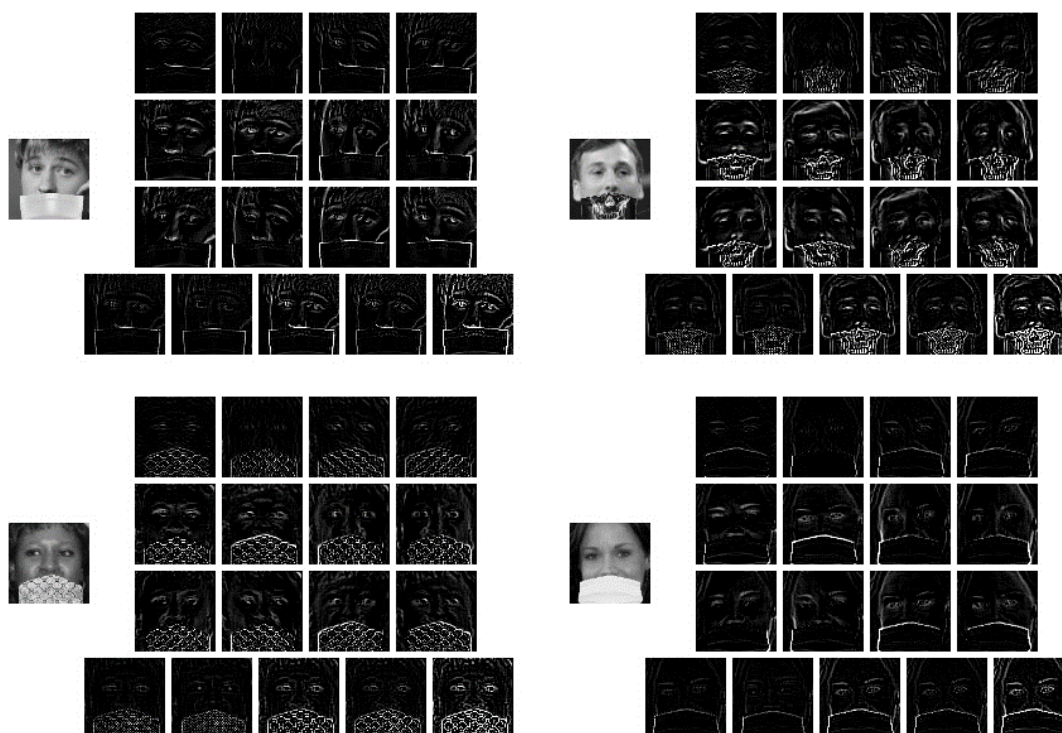
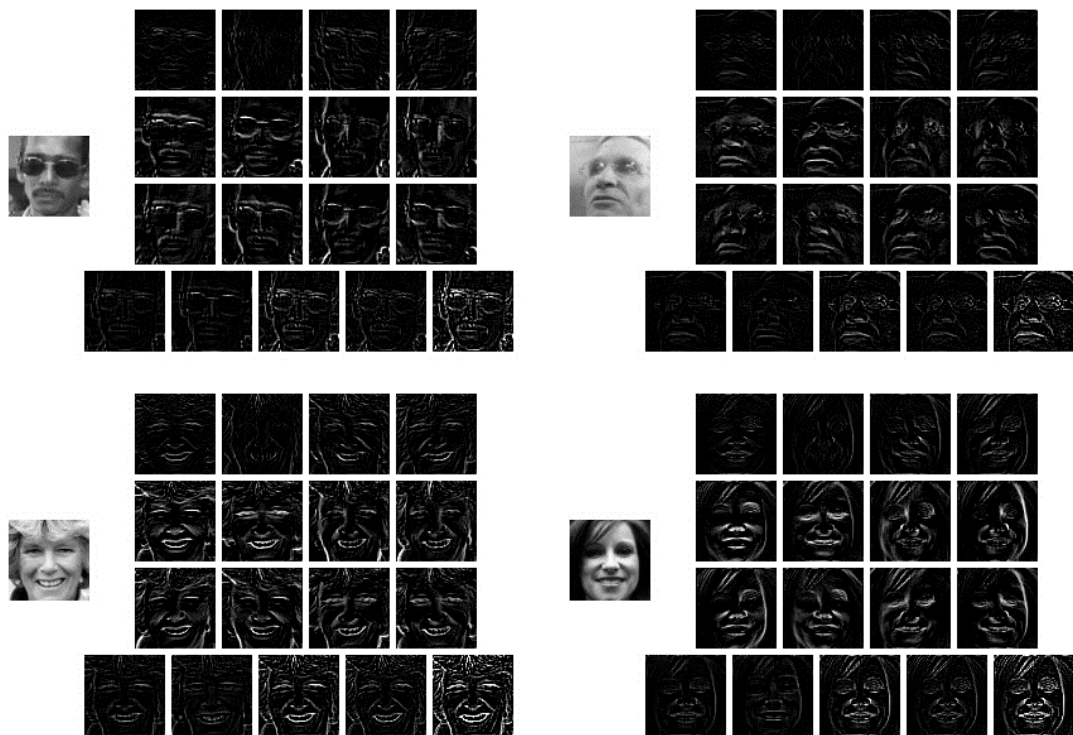
*17 Filters Outputs of Example Mask Wrong Class*

Figure 14.

*17 Filters Outputs of Example No Mask Class****Network and Training***

By applying preprocessing to the images, the input spatial resolution becomes  $W \times H \times 17$ , so the input layer of the customized convolutional neural network is specified to accept this input shape. After the input layer, a convolutional layer is added to reduce the input to  $W \times H \times 1$ , which is a single-channel image of  $W \times H$  size. So up to this point, the network actually eliminates the features from the whole 17 images and obtains a single resulting image. After this layer, a flattening layer is applied before continuing with the final dense layers. After two fully connected dense layers with “relu” activation function, the last layer is a dense layer with “softmax” function for the classification result. The custom network architecture is shown in Figure 15 and the block diagram of the process is given in Figure 16.

For training, the dataset is split into 3 which are, train, validation, and test datasets. 20% of the dataset is reserved for test. From the remaining 80%, again 20% is used for validation 80% is used for training. This results in 64% for training, 16% for validation, and 20% for test. Adam optimizer was used and different learning rates were tried for comparison. These learning rates are  $10^{-3}$ ,  $10^{-4}$ ,  $10^{-5}$ ,  $10^{-6}$ , and  $10^{-7}$ . These

learning rates were selected to have a good comparison of their effect on accuracy and training time but without being very low or very high. Very low learning rates can cause the network to have a very long training time without having a good fit on the data and remain at low accuracy. On the other hand, using a high learning rate can cause the network to overfit on the training data which will then cause generalization problems and accuracy will be low on any other data than the training data. During the training process, accuracy, precision, recall, TP, TN, FP, and FN metrics are monitored and the loss function is set as “categorical cross entropy” which is also monitored. Patience-based training is used which traces validation loss and stops the training process when it does not decrease by a given threshold value in the number of patience epochs which is 100 in this research. When the training process is stopped, the weights of the best resulting epoch are restored to the model. All of the training process is done with a computer with 32GB system memory, 10th generation Intel Core i7-10750H processor, and NVIDIA GeForce RTX2070 SUPER graphics card with 8GB memory.

Figure 15.

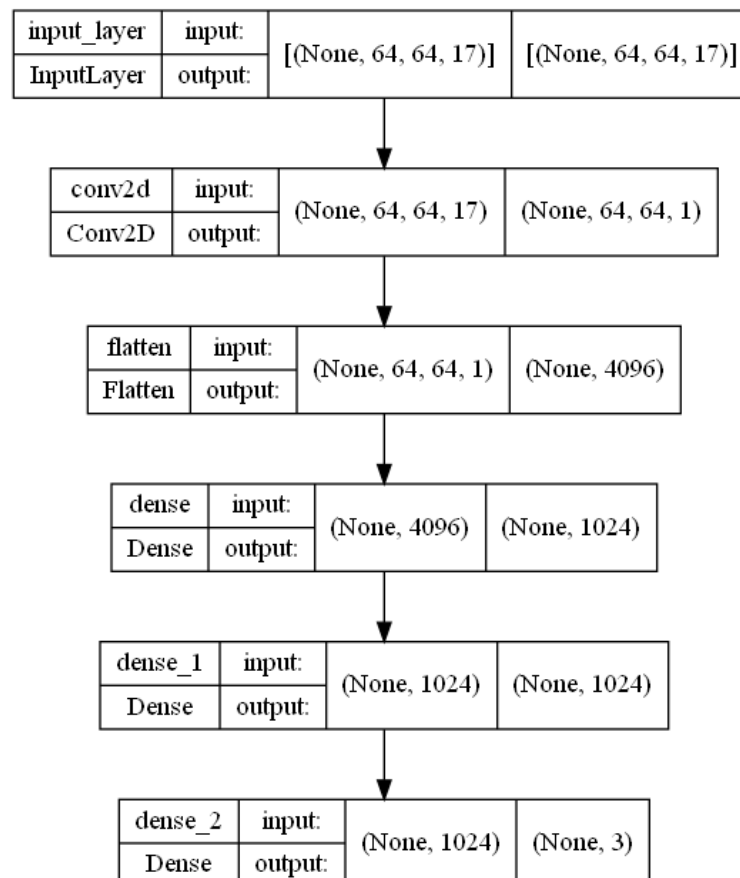
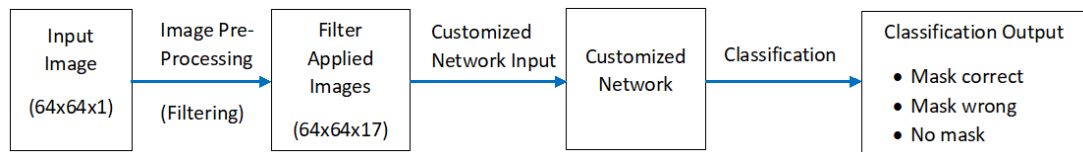
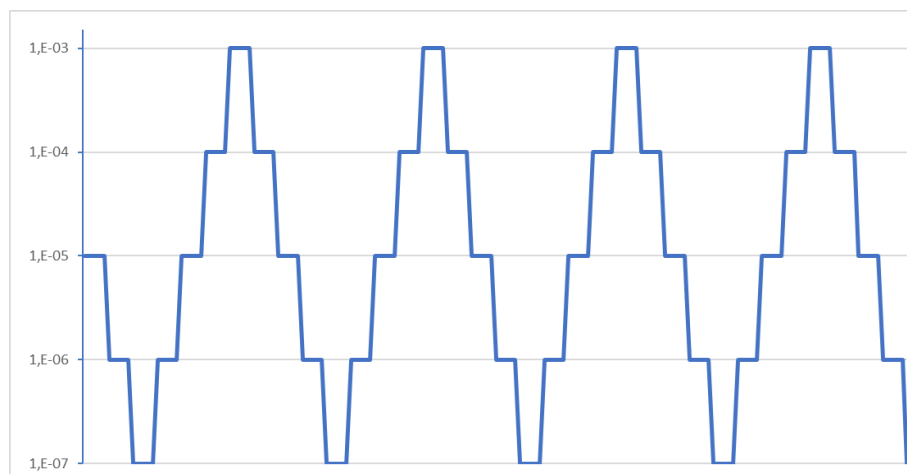
*Proposed Custom Network Architecture*

Figure 16.

*Block Diagram of the Process**Oscillating Learning Rate*

After implementing training on different learning rates to investigate the effect on the training process, the oscillating learning rate method was also implemented. In this training method, the learning rate was changed every  $n$  epochs which was chosen as 5 in this study. The learning rate was divided by 10 or multiplied by 10 every 5 epochs to keep it oscillating between the top boundary  $10^{-3}$  and the bottom boundary  $10^{-7}$ . The change in learning rate over epochs can be visually observed in Figure 17.

Figure 17.

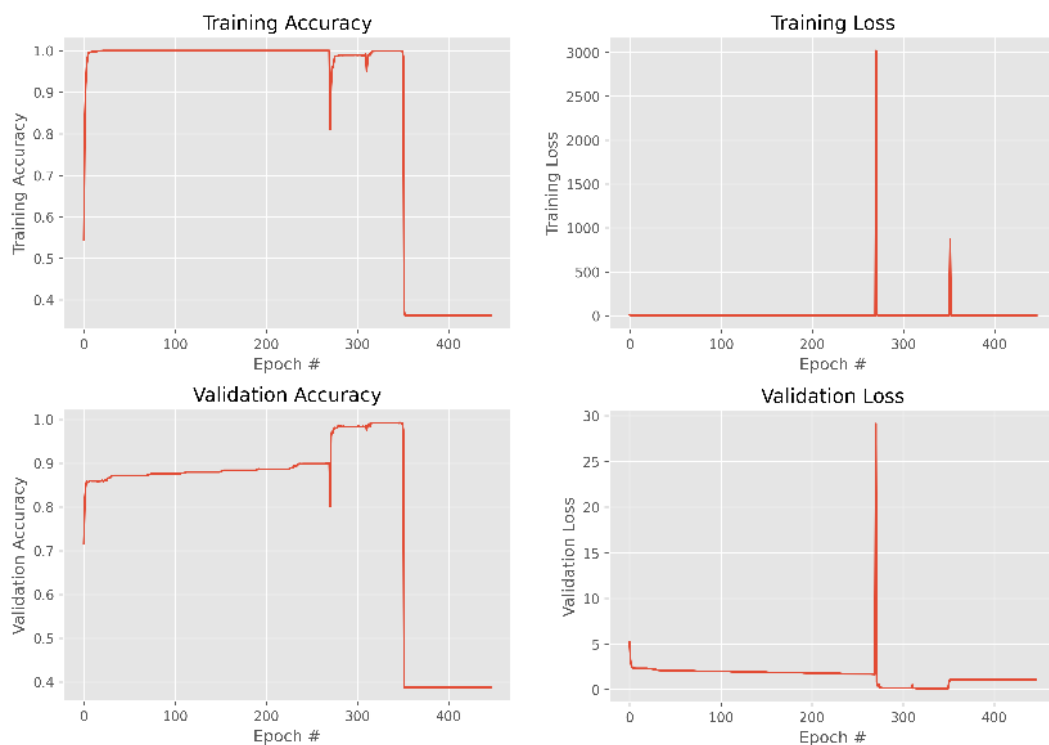
*Oscillating Learning Rate**Weight Dropping (Set Zero) Between Epochs*

When the accuracy and loss graphs of both training and validation were investigated it was seen that the network had a good improvement in achieving higher validation accuracy and lower validation loss after a specific event. By checking the numerical training history data and the graphs visually it is seen that once the training

accuracy hits 1.0 and the training loss goes close to zero the network stops improving itself as needed which means overfitting occurs. Until change in the learning rate triggers a spike on the training graphs which lowers the training accuracy and increases the training loss for a short time which is then followed by a spike in validation accuracy and validation loss which affect them positively and result in better results (Figure 18).

Figure 18.

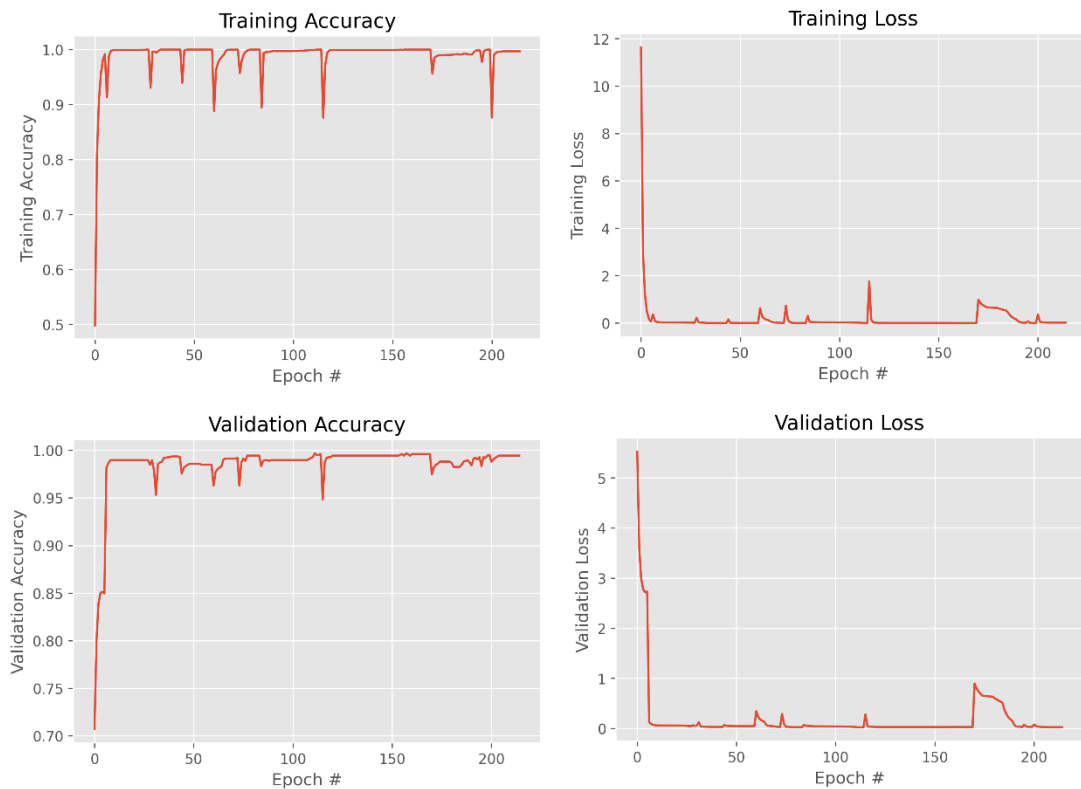
*Training and Validation Accuracy and Loss Graphs  
(Oscillating Learning Rate)*



By using this idea, a new training method was tried to see the effect on training and validation outcomes. In this method, training and validation metrics are traced between epochs, and once training accuracy hits 1.0 or the difference between training loss and validation loss increases to more than 10 times, weights are randomly set to zero. By using this method it was observed that better accuracy was obtained in fewer epochs (Figure 19).

Figure 19.

*Training and Validation Accuracy and Loss Graphs  
(Oscillating Learning Rate + Weight Dropping)*



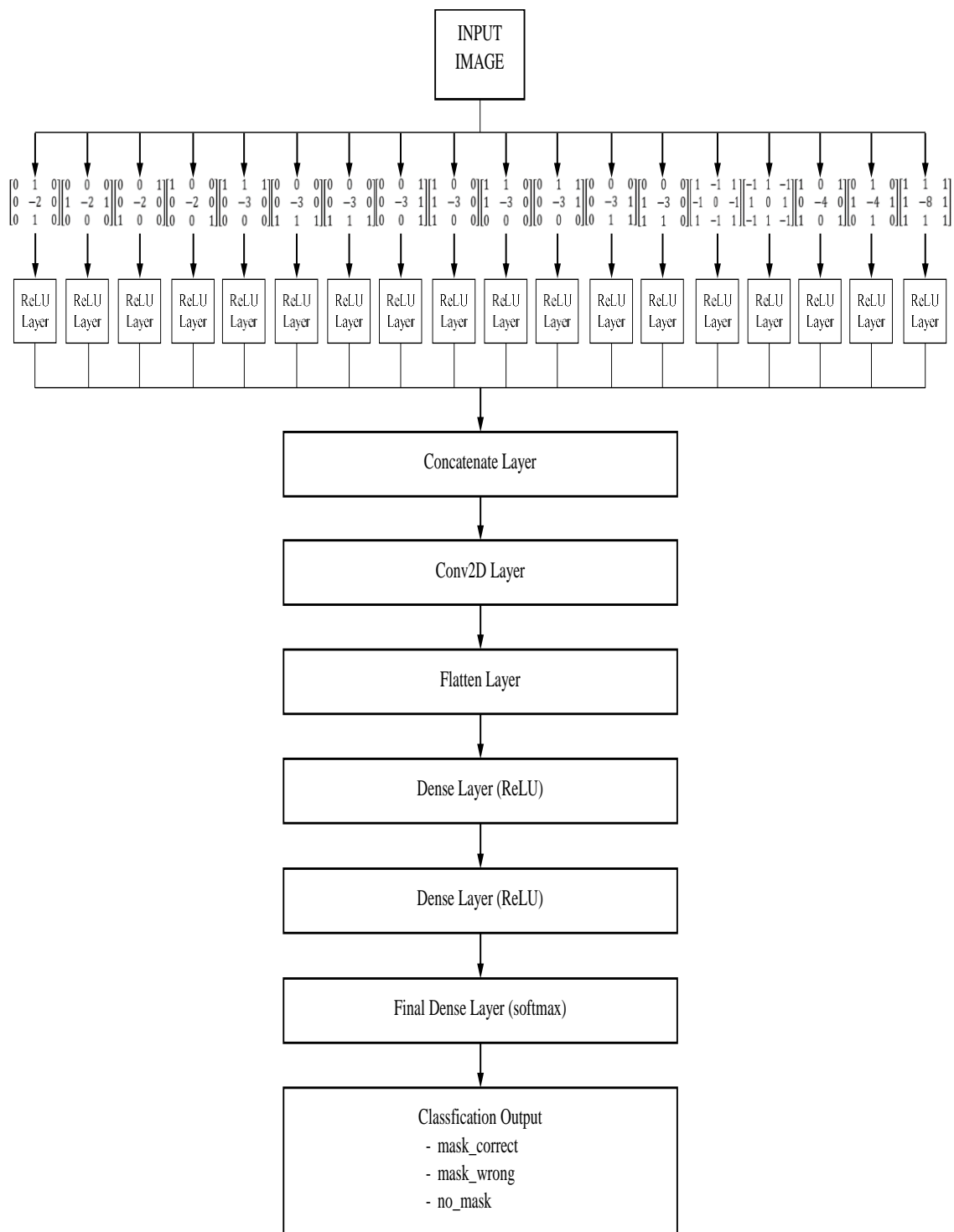
***Embedding Image Processing Filters into Network Architecture***

After making performance improvements in the training process with learning rate change and weights dropping features, the next step was embedding the filtering process into the network architecture which was used as preprocessing. To achieve this, convolutional layers were used. The input image was connected to multiple Conv2D layers having 3x3 kernel size and kernels are set to be initialized with the filter values selected. The weights of these layers were frozen by setting their “trainable” parameter to false to prevent the weights from being updated during the training process and act like fixed filters. Then the outputs of these layers are concatenated and given to another Conv2D layer to reduce the data into a single image again. So the input image with the shape  $W \times H \times 1$  becomes  $W \times H \times 17$  after the filters are applied and concatenated, then it is again reduced to the shape  $W \times H \times 1$  by using the convolutional layer. The output of this layer is then given to the second part of the network which results in the classification output. This was achieved by flattening the

image and using two dense layers with “relu” and finally a “softmax” layer to obtain the classification output.

When results were obtained by this network, they were different and not as good as the previous results where the filters were applied outside the network and then used for training. To find the cause, the outputs of filters of both cases were investigated. When the numerical outputs of embedded filters are compared to the other ones, it was seen that embedded filters give negative pixel output results which is not the case for applying filters to images. To fix this problem another layer had to be added after the filters which will act as a thresholding layer and set the negative pixel values to zero. To achieve this, a layer with “relu” function was added after the filters. After this, the resulting custom network architecture was ready (Figure 20).

Figure 20.

*Final Custom Network Architecture*



## CHAPTER IV

### Findings and Discussion

#### Results For Modified Networks

After the experimental results were obtained, networks with an accuracy above 70% were selected for comparison. ResNet50, ResNet101, and ResNet152, as well as EfficientNets, were not able to obtain a good accuracy with 32x32 grayscale images, so they were not included in the comparison.

After obtaining results with the original dataset for all of the networks, different image preprocessing techniques were evaluated for further improvements to the networks. Contrast stretching was applied to obtain low and high contrast images. Furthermore, Discrete Cosine Transform (DCT) was applied, which has improved accuracy by compressing the image in some of the studies. However, in our case, none of these approaches was able to improve the accuracy of the networks. Since grayscale images with a resolution as low as 32x32 were being used, the task became harder for the networks, even with the original images.

Although image preprocessing affects ANN applications in a good way, our experiments showed that it can negatively affect the results for small input data, which is 1024 (32x32x1) in our research. Table 4 shows the overall performances of the networks included in the comparison.

By comparing the training times of the networks, it can be seen that VGG16 required the shortest time, with 105 minutes. On the contrary, ResNet152V2 reported the highest value, with 607 minutes. The training times of Xception and MobileNet were also very close to that of VGG16, with 108 and 112 minutes, respectively.

While evaluating the networks on the test dataset, their evaluation were also recorded. An 80-20% train-test ratio was used for all of the networks, and the test dataset included 1579 images. VGG16 reported the best evaluation time, with 2,52 seconds, and VGG19 was the second best (2,68 seconds). Considering that the image size for Xception was 71x71, its evaluation time of 4,05 seconds was also a good result. The highest evaluation time was 14,2 seconds, obtained by the InceptionResNetV2 with a 75x75 test image size.

The size of the network is also an important parameter for implementation. It may not be always possible to implement a model with the highest accuracy on every

device because of the size of the network and its memory requirements. By comparing the output model sizes of all networks, it was observed that the MobileNetV2 network has the smallest size, with 29,6 MB. The MobileNet network has the second smallest size with 38,9 MB, and ResNet152V2 and InceptionResNetV2 have the two largest model sizes (679 and 638 MB, respectively).

Regarding the accuracies of the networks, InceptionResNetv2 and Xception had the highest value, with 99,6%. VGG19 and VGG16 also showed an accuracy over 99 (with 99,4% and 99,1%, respectively). As for the general performance of the networks, if the highest possible accuracy is desired, the Xception network could be selected, as it has moderate values regarding model size. If a little lower accuracy can be tolerated, still over 99%, VGG16 or VGG19 are also good options, with faster evaluation time. However, they have slightly higher model sizes, which is also a drawback. Figure 21, Figure 22, and Figure 23 are the confusion matrices of VGG16, VGG19, and Xception.

Table 4.

*Network Performance Results*

<b>Network (Input Size)</b>	<b>Epochs Run</b>	<b>Training time (mins)</b>	<b>Evaluation Time(s)</b>	<b>Model Size (MB)</b>	<b>F1-Score</b>	<b>Accuracy</b>
DenseNet121 (32x32)	1178	226	6.13	89.6	0.9968	0.9690
DenseNet169 (32x32)	1554	267	6.47	157	0.9625	0.9650
DenseNet201 (32x32)	1681	334	8.05	224	0.9613	0.9630
InceptionRes NetV2(75x75)	682	205	14.2	638	0.9950	0.9960
InceptionV3 (75x75)	1810	241	6.95	256	0.9740	0.9760
MobileNet (32x32)	3417	112	3.01	38.9	0.8112	0.8230
MobileNetV2 (32x32)	4118	198	3.69	29.6	0.7238	0.7300
ResNet50V2 (32x32)	1445	122	4.55	273	0.8917	0.8980
ResNet101V2 (32x32)	1808	315	6.73	495	0.8777	0.8820
ResNet152V2 (32x32)	2173	607	6.83	679	0.8773	0.8888
VGG16 (32x32)	2078	105	2.52	385	0.9885	0.9910
VGG19 (32x32)	2412	150	2.68	445	0.9924	0.9940
Xception (71x71)	587	108	4.05	241	0.9963	0.9960

Figure 21.

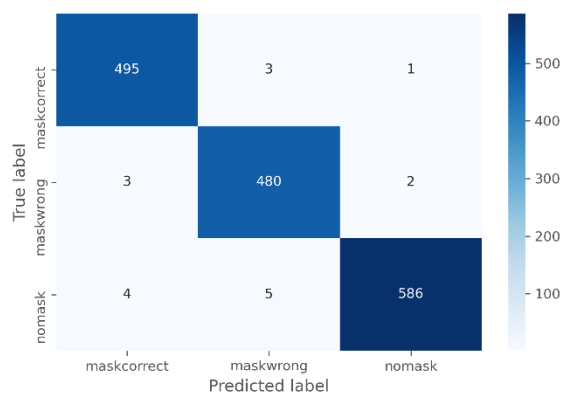
*VGG16 Confusion Matrix*

Figure 22.

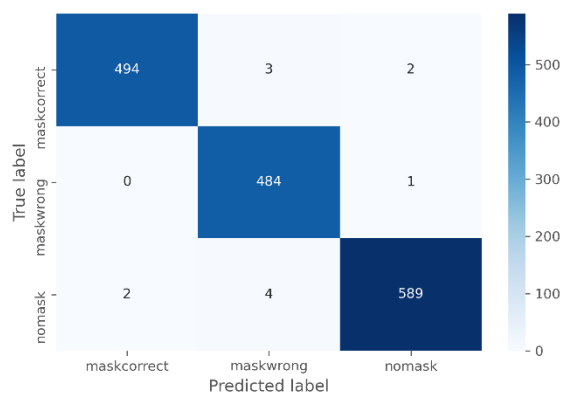
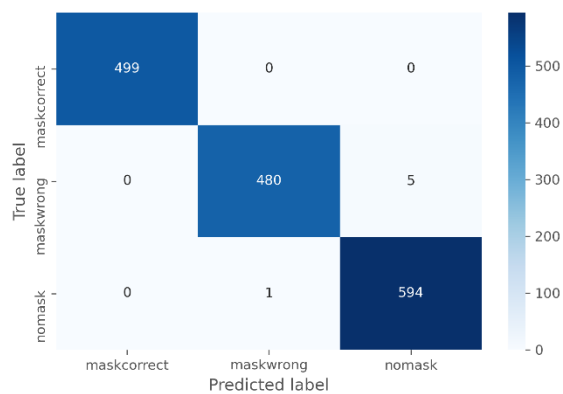
*VGG19 Confusion Matrix*

Figure 23.

*Xception Confusion Matrix*

## Results for Customized Network

Results were obtained for five different learning rates. At the start, the learning rate was set to  $10^{-7}$  and increased by 10 times for each try to compare the results. It is observed that the increase in the learning rate resulted in faster training and improved accuracy. A test dataset was used which was not involved in the training process to also check the test accuracy if any overfitting problem is caused. According to the obtained results, no overfitting problem was observed. At the  $10^{-7}$  learning rate the training process run for 4545 epochs for 180 minutes. The validation accuracy was 96.2% while the test accuracy was 95.69%. When the learning rate was increased to  $10^{-6}$ , the training process was about 4 times faster with 44 minutes with 1010 epochs. The validation accuracy was slightly better with 96.4% but the test accuracy was 95.57%. At the  $10^{-5}$  learning rate, the training time was improved and the results were obtained in 470 epochs in 20 minutes. Both validation and test accuracies were better than the previous ones with 97.4% and 96.33% respectively. When the learning rate was increased to  $10^{-4}$  a good performance increase occurred both for time and accuracy. The epochs run was 115 in 5.5 minutes. The obtained validation accuracy was 99.3% and the test accuracy was 98.16% which was a good increase. Then the last learning rate of  $10^{-3}$  was tried and the results were close to the results of  $10^{-4}$ . 114 epochs were run in 5.4 minutes. The validation accuracy was 98.8% which was lower, but the test accuracy was 98.86% which was better and the highest result obtained. Obtained results are summarized in Table 5. It is also important to mention that the final size of the customized network is about 60.4MB which is a good size to be implemented in an application and does not require devices with high performance needs.

Table 5.

*Results Obtained from the Custom Network with 64x64 images for each learning rate*

<b>Learning Rate</b>	<b>Validation Accuracy</b>	<b>Test Accuracy</b>	<b>Time (s)</b>	<b>Epochs</b>
$10^{-3}$	98.8%	98.86%	323	114
$10^{-4}$	99.3%	98.16%	326	115
$10^{-5}$	97.4%	96.33%	1170	470
$10^{-6}$	96.4%	95.57%	2629	1010
$10^{-7}$	96.2%	95.69%	10791	4545

In the next step, an oscillating learning rate approach was implemented and its effect on training time and accuracy was investigated. The results showed that, by using this approach good accuracy can be obtained in moderate time without the need to search for an effective fixed learning rate. Table 6 shows the comparison of this approach to a fixed learning rate.

Table 6.

*Comparison of Fixed and Oscillating Learning Rate*

<b>Learning Rate</b>	<b>Validation Accuracy</b>	<b>Test Accuracy</b>	<b>Time (s)</b>	<b>Epochs</b>
$10^{-3}$	98.8%	98.86%	323	114
$10^{-4}$	99.3%	98.16%	326	115
$10^{-5}$	97.4%	96.33%	1170	470
$10^{-6}$	96.4%	95.57%	2629	1010
$10^{-7}$	96.2%	95.69%	10791	4545
<b>Osc. <math>10^{-3}</math>-<math>10^{-7}</math></b>	<b>99.3%</b>	<b>98.67%</b>	<b>1101</b>	<b>447</b>
<b>Osc. <math>10^{-4}</math>-<math>10^{-7}</math></b>	<b>98.89%</b>	<b>97.53%</b>	<b>2113</b>	<b>814</b>
<b>Osc. <math>10^{-4}</math>-<math>10^{-6}</math></b>	<b>99.29%</b>	<b>97.78%</b>	<b>1993</b>	<b>774</b>
<b>Osc. <math>10^{-3}</math>-<math>10^{-6}</math></b>	<b>91.37%</b>	<b>36.35%</b>	<b>1042</b>	<b>392</b>

After the comparison of the oscillating learning rate approach, a weight-dropping feature was added to the training process which was investigated to positively effect the training both in time and accuracy. When using the weight-dropping feature, both training and validation metrics are checked after every epoch and when the training accuracy reaches 1.0 or the difference between training loss and validation loss increases to more than 10 times, random weights are set to zero. Table 7 shows the results obtained with this training method.

Table 7.

*Comparison of Fixed and Oscillating Learning Rate with Weight Drop Feature*

<b>Learning Rate</b>	<b>Validation Accuracy</b>	<b>Test Accuracy</b>	<b>Time (s)</b>	<b>Epochs</b>
$10^{-3}$	99.76%	99.18%	385	128
$10^{-4}$	99.76%	99.56%	521	193
$10^{-5}$	99.76%	99.3%	472	170
$10^{-6}$	99.6%	99.24%	614	227
$10^{-7}$	98.73%	97.97%	949	361
<b>Osc. <math>10^{-3}</math>-<math>10^{-7}</math></b>	<b>99.37%</b>	<b>98.42%</b>	<b>351</b>	<b>123</b>
<b>Osc. <math>10^{-4}</math>-<math>10^{-7}</math></b>	<b>99.76%</b>	<b>99.18%</b>	<b>1041</b>	<b>403</b>
<b>Osc. <math>10^{-4}</math>-<math>10^{-6}</math></b>	<b>99.76%</b>	<b>99.49%</b>	<b>439</b>	<b>154</b>
<b>Osc. <math>10^{-3}</math>-<math>10^{-6}</math></b>	<b>99.29%</b>	<b>98.61%</b>	<b>362</b>	<b>115</b>

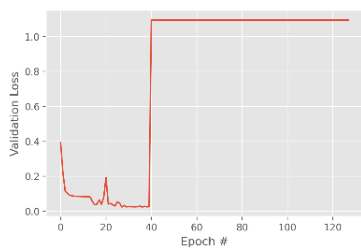
In Figure 24, validation loss graphs according to these results can be observed and applied weight drops are seen in these graphs as spikes. Some of the weight drops are then followed by a lower validation loss which means improvement while some of them cause increased validation loss which causes a negative effect. Since patience-based training approach is used, best-obtained weights are loaded back into the network when the training process is finished. Validation accuracies of the obtained results are shown in Figure 25 where the effect caused by the weight drop feature can be observed again with the spikes.

Figure 24.

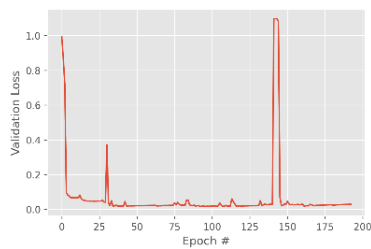
*Validation Loss Graphs of Fixed and Oscillating Learning Rate with Weight Drop Feature*

(a) $10^{-3}$ . (b) $10^{-4}$ . (c) $10^{-5}$ . (d) $10^{-6}$ . (e) $10^{-7}$ .

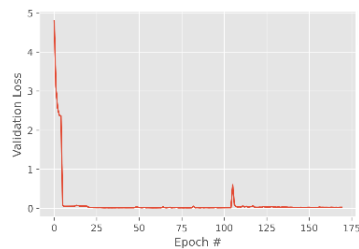
(f)Osc.  $10^{-3}$ - $10^{-7}$ . (g)Osc.  $10^{-4}$ - $10^{-7}$ . (h)Osc.  $10^{-4}$ - $10^{-6}$ .(i)Osc.  $10^{-3}$ - $10^{-6}$ .



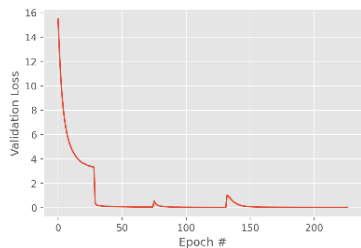
(a)



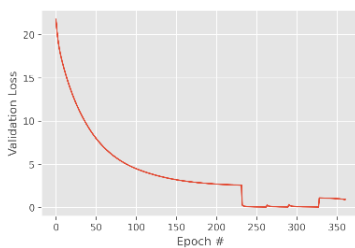
(b)



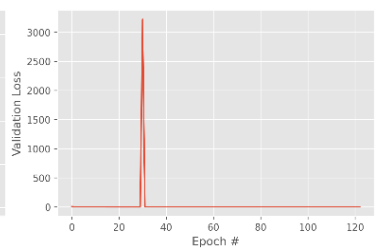
(c)



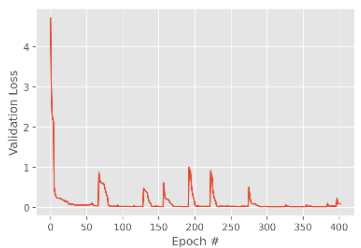
(d)



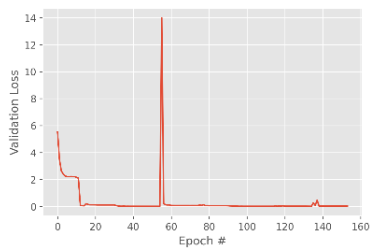
(e)



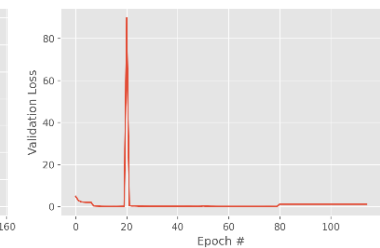
(f)



(g)



(h)



(i)

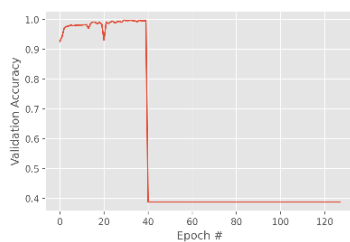


Figure 25.

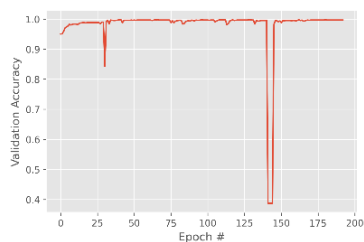
*Validation Accuracy Graphs of Fixed and Oscillating Learning Rate with Weight Drop Feature*

(a) $10^{-3}$ . (b) $10^{-4}$ . (c) $10^{-5}$ . (d) $10^{-6}$ . (e) $10^{-7}$ .

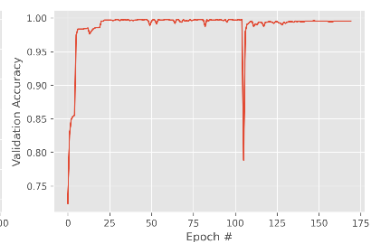
(f)*Osc.*  $10^{-3}$ - $10^{-7}$ .(g)*Osc.*  $10^{-4}$ - $10^{-7}$ . (h)*Osc.*  $10^{-4}$ - $10^{-6}$ .(i)*Osc.*  $10^{-3}$ - $10^{-6}$ .



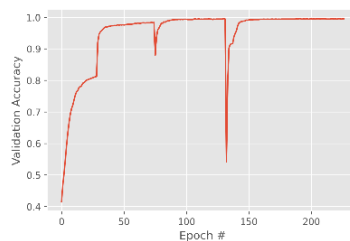
(a)



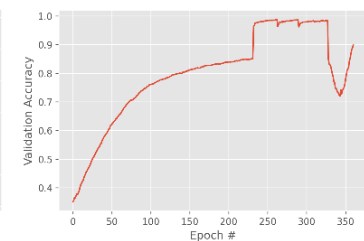
(b)



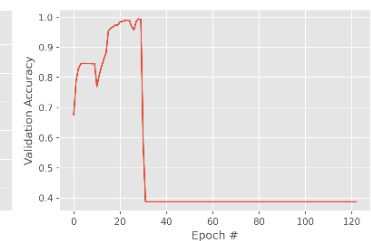
(c)



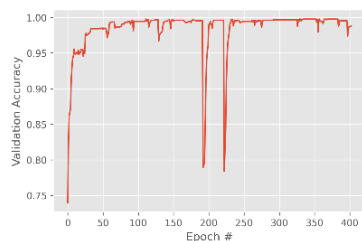
(d)



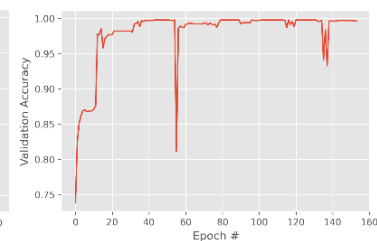
(e)



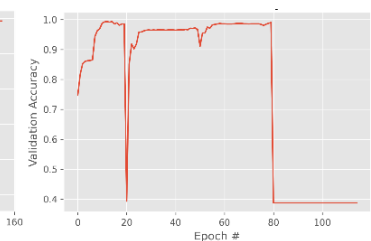
(f)



(g)



(h)



(i)

After having good results with filtering as a preprocess, this approach was embedded inside the network architecture using convolutional layers as explained in the previous chapter. Table 8 shows the obtained results using this network at different learning rates. Validation loss and validation accuracy graphs of all learning rates are shown in Figure 26 and Figure 27 respectively. 99.84% validation accuracy and 99.43% test accuracy were obtained at the  $10^{-4}$  fixed learning rate, while 99.6% validation accuracy and 99.3% test accuracy were obtained at an oscillating learning rate between  $10^{-4}$  and  $10^{-6}$ . Confusion matrices of these results are given in Figure 28.

Table 8.

*Results of Filter Embedded Custom CNN Architecture*

<b>Learning Rate</b>	<b>Validation Accuracy</b>	<b>Test Accuracy</b>	<b>Time (s)</b>	<b>Epochs</b>
$10^{-3}$	99.52%	99.11%	375	139
$10^{-4}$	99.84%	99.43%	616	241
$10^{-5}$	99.76%	98.86%	667	258
$10^{-6}$	99.45%	99.05%	930	368
$10^{-7}$	99.21%	98.54%	1922	798
<b>Osc. <math>10^{-3}</math>-<math>10^{-7}</math></b>	<b>99.13%</b>	<b>98.10%</b>	<b>335</b>	<b>123</b>
<b>Osc. <math>10^{-4}</math>-<math>10^{-7}</math></b>	<b>99.37%</b>	<b>98.61%</b>	<b>489</b>	<b>194</b>
<b>Osc. <math>10^{-4}</math>-<math>10^{-6}</math></b>	<b>99.60%</b>	<b>99.30%</b>	<b>398</b>	<b>151</b>
<b>Osc. <math>10^{-3}</math>-<math>10^{-6}</math></b>	<b>98.81%</b>	<b>98.10%</b>	<b>312</b>	<b>115</b>

Figure 26.

*Validation Loss Graphs of Filter Embedded Custom CNN Architecture*

(a) $10^{-3}$ . (b) $10^{-4}$ . (c) $10^{-5}$ . (d) $10^{-6}$ . (e) $10^{-7}$ . (f)*Osc.*  $10^{-3}$ - $10^{-7}$ .

(g)*Osc.*  $10^{-4}$ - $10^{-7}$ . (h)*Osc.*  $10^{-4}$ - $10^{-6}$ . (i)*Osc.*  $10^{-3}$ - $10^{-6}$ .

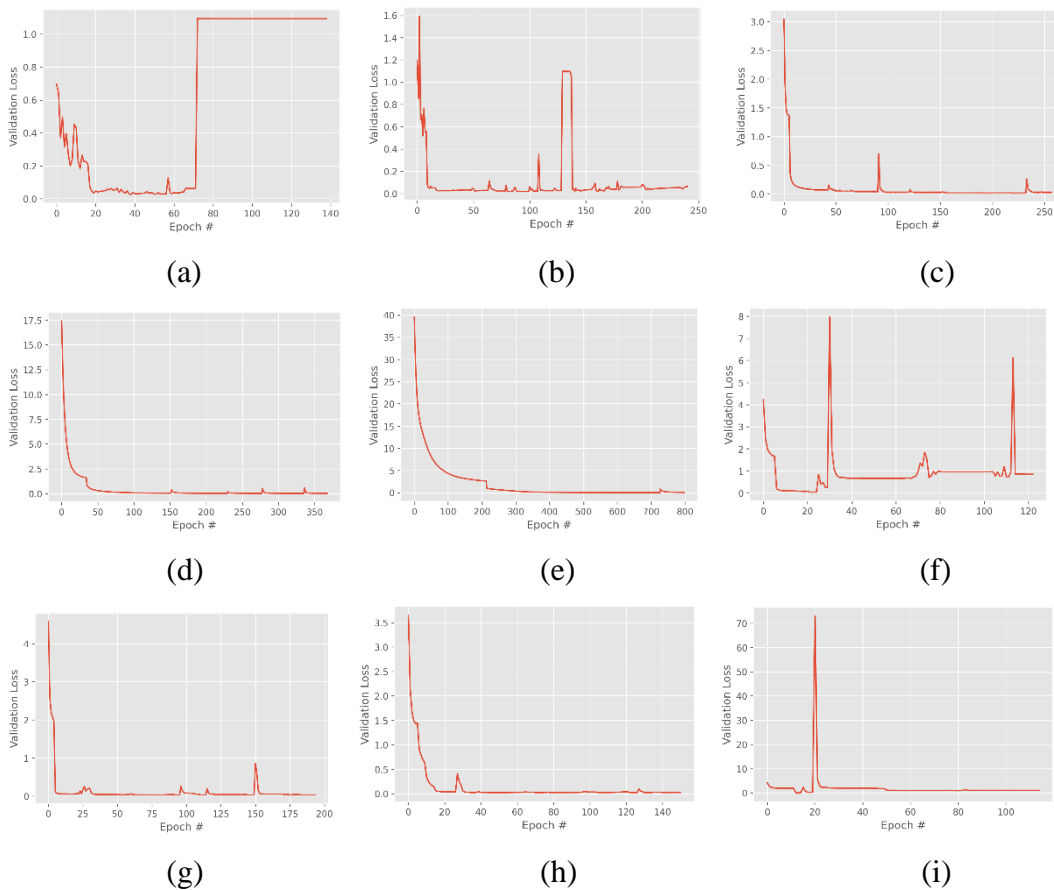


Figure 27.

*Validation Accuracy Graphs of Filter Embedded Custom CNN Architecture*

(a) $10^{-3}$ . (b) $10^{-4}$ . (c) $10^{-5}$ . (d) $10^{-6}$ . (e) $10^{-7}$ . (f)*Osc.  $10^{-3}$ - $10^{-7}$* .

(g)*Osc.  $10^{-4}$ - $10^{-7}$* . (h)*Osc.  $10^{-4}$ - $10^{-6}$* .(i)*Osc.  $10^{-3}$ - $10^{-6}$* .

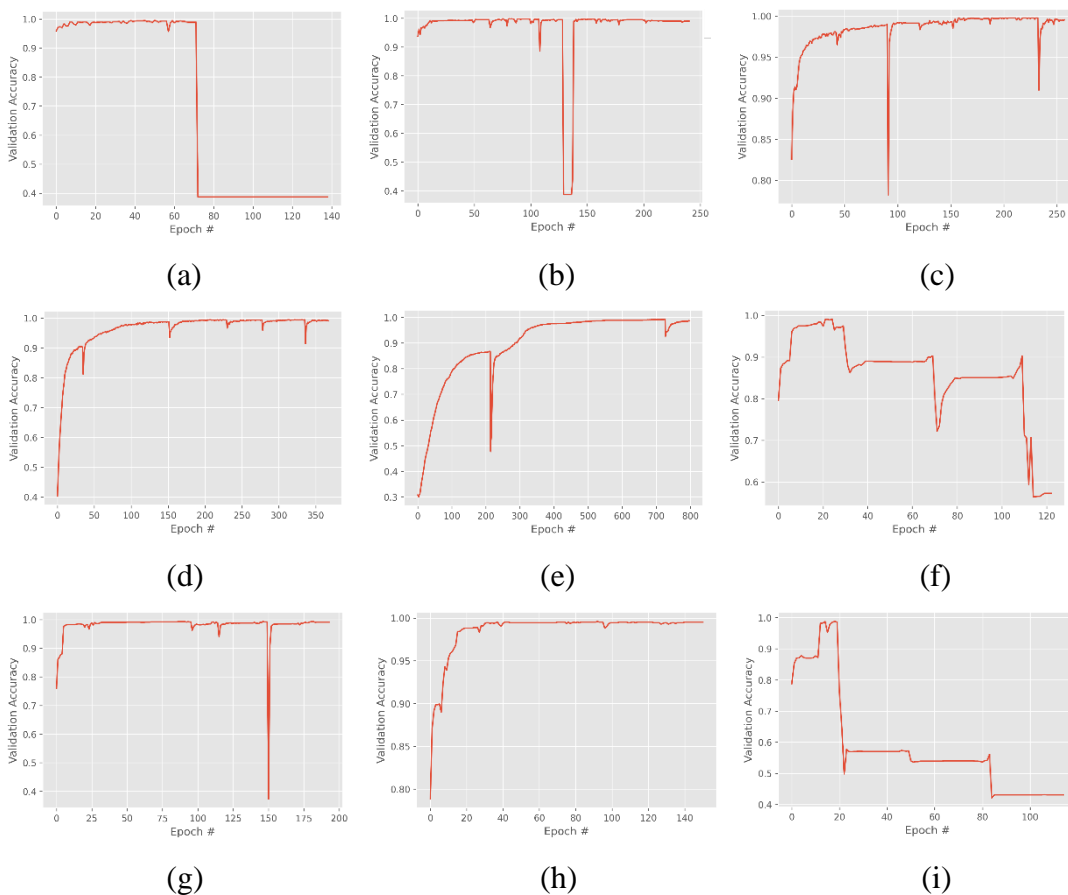
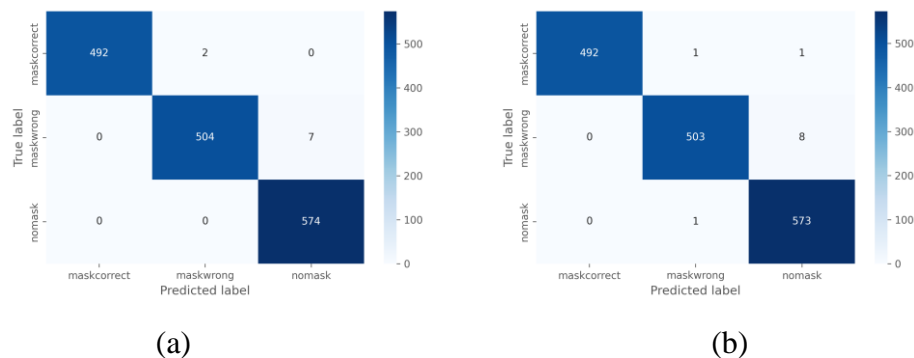


Figure 28.

*Confusion Matrices of Filter Embedded Custom CNN Architecture*

(a)99.43% Test Accuracy at  $10^{-4}$  Learning Rate. (b)99.3% Test Accuracy at

*Oscillating Learning Rate between  $10^{-4}$ - $10^{-6}$ .*



When the final results were obtained from the customized network, a final comparison with the modified networks was done which is given in Table 9. When the epochs run are considered, all versions of the proposed network have shorter epoch numbers and training times which is caused by the different learning rates used and also the training optimizations used for the customized networks during the training process.

Model size comparison shows that the 32x32 input version of the network has the smallest size with 24.7MB, which is followed by MobileNetV2 and MobileNet with 29.6MB and 38.9MB network sizes respectively. 64x64 input version of the network comes after these networks with 60.8MB network size which is then followed by DenseNet121 with 89.MB size.

When the obtained accuracies are compared, the modified networks Xception and InceptionResNetv2 had the highest value with 99.6%, and VGG19 and VGG16 followed by 99,4% and 99,1%, respectively. Customized networks were evaluated both with validation and test accuracies on separate test images that were not involved in the training process. 64x64 and 128x128 input versions of the networks both had 99.84% validation accuracy while they had 99.43% and 99.49% test accuracies respectively. With a lower input size, the 32x32 version of the network obtained 99.45% validation and 98.42% test accuracy.

When accuracy and model size are both considered, the proposed customized network with 64x64 input image size is the most optimal network to be used. By using this network, 99.43% accuracy is obtained with a network size of 60.8MB. Also obtained results from all versions of the network are compared with some recent related studies in the literature in Table 10 according to their classification accuracies with both two and three classes.

Table 9.  
*Performance Comparison of Proposed Network*

<b>Network (Input Size)</b>	<b>Epochs Run</b>	<b>Training time (mins)</b>	<b>Model Size (MB)</b>	<b>F1-Score</b>	<b>Accuracy</b>
DenseNet121 (32x32)	1178	226	89.6	0.9968	0.9690
DenseNet169 (32x32)	1554	267	157	0.9625	0.9650
DenseNet201 (32x32)	1681	334	224	0.9613	0.9630
InceptionResNetV2 (75x75)	682	205	638	0.9950	0.9960
InceptionV3 (75x75)	1810	241	256	0.9740	0.9760
MobileNet (32x32)	3417	112	38.9	0.8112	0.8230
MobileNetV2 (32x32)	4118	198	29.6	0.7238	0.7300
ResNet50V2 (32x32)	1445	122	273	0.8917	0.8980
ResNet101V2 (32x32)	1808	315	495	0.8777	0.8820
ResNet152V2 (32x32)	2173	607	679	0.8773	0.8888
VGG16 (32x32)	2078	105	385	0.9885	0.9910
VGG19 (32x32)	2412	150	445	0.9924	0.9940
Xception (71x71)	587	108	241	0.9963	0.9960
Proposed Network (32x32)	310	12	24.7	0.9845*	0.9945 0.9842*
Proposed Network (64x64)	241	10	60.8	0.9944*	0.9984 0.9943*
Proposed Network (128x128)	271	18	204	0.9950*	0.9984 0.9949*

\* Obtained by test dataset.

Table 10.

*Comparison with the Recent studies in the Literature*

<b>Reference</b>	<b>Classes</b>	<b>Accuracy</b>
Syafitri et al. (2024)	2	94.78%
Nirmaladevi et al. (2024)	2	98.1%
Kaur et al. (2024)	2	98.79%
Fazeli Ardekani et al.(2024)	2	99.02%
Parikh et al. (2024)	3	85%
Likhith et al. (2024)	3	90.74%
Mostafa et al. (2024)	3	92.2%
Aydemir et al. (2022)	3	95.95%
Tun and Myat (2024)	3	96%
Sheikh and Zafar (2023)	3	97.81%
Crespo et al. (2022)	3	99.2%
<b>Proposed Network (32x32)</b>	<b>3</b>	<b>98.42%</b>
<b>Proposed Network (64x64)</b>	<b>3</b>	<b>99.43%</b>
<b>Proposed Network (128x128)</b>	<b>3</b>	<b>99.49%</b>

## CHAPTER V

### Conclusion and Recommendations

This chapter summarizes the research and gives conclusions according to the research objectives, and experimental results, and gives recommendations for future work.

#### Conclusion

This research aimed to make an accurate classification of three conditions of face mask wearing. Correct, wrong, and no mask-wearing conditions have been simulated with the processed dataset. The dataset was created by adding face masks to the Labeled Faces in the Wild (LFW) dataset to simulate correct, wrong, and no mask-wearing conditions.

In the first part of the research, deep convolutional neural networks were trained with as little input data as possible to obtain an accurate model for face mask detection and wearing condition classification. According to the experimental results, EfficientNets and ResNet50, ResNet101, and ResNet152 networks were not able to learn with their minimum input size of 32x32 and grayscale images. Four of the trained networks were able to obtain an accuracy of over 99%, i.e., InceptionResNetv2, Xception, VGG16, and VGG19, with accuracies of 99.6%, 99.6%, 99.1%, and 99.4% respectively.

The second part of the research aimed to obtain an accurate classification of face mask usage using a filter-based approach as a pre-process for feature extraction to give them to a customized neural network. For this purpose, 17 filters were chosen and applied to input images. Then the input of the network was adjusted accordingly to accept these 17 images obtained by the filters. At the training stage of the network, different learning rates were tried to find the optimal learning rate. Results showed that at  $10^{-3}$  and  $10^{-4}$  learning rates, both training time and accuracy were better. The obtained validation and test accuracies were 98.8% and 98.86% at the  $10^{-3}$  learning rate, while 99.3% and 98.16% at  $10^{-4}$  respectively. After this step, an oscillating learning rate approach was also implemented. When the learning rate oscillated between  $10^{-3}$  and  $10^{-7}$ , 99.3% validation accuracy and 98.67% test accuracy were obtained. And between  $10^{-4}$  and  $10^{-7}$ , validation and test accuracies were 98.89% and



97.53%. At  $10^{-4}$ - $10^{-6}$ , validation and test accuracies were 99.29% and 97.78%. Between  $10^{-3}$  and  $10^{-6}$ , the network was not able to obtain a good result. After comparing fixed learning rate and oscillating learning rate approaches, the weight-dropping method was implemented during the training process which had a positive effect on both training time and accuracy. To apply the weight-dropping feature, training and validation metrics were traced after each epoch, and random weights were set to zero when the training accuracy reached 1.0 or the difference validation loss and the training loss increased more than 10 times.

The third part of the research was to embed the filtering process inside a custom network architecture after achieving improvements in training with the learning rate changes and weight-dropping feature. Convolutional layers were used to apply the filtering process inside the network. Multiple Conv2D layers having 3x3 kernel size and kernels with the selected values were connected to the input. The “trainable” parameter of the layers was set to false to freeze them and prevent them from being updated during the training process, which made them act like fixed filters. A layer with the “relu” function was also added after each filter to act as a thresholding layer because negative pixel outputs were observed. The output of these layers was then concatenated and the data was reduced to a single image using another Conv2D layer and given to the second part of the network which gives the classification result as output. The network was implemented at 3 different input sizes 32x32, 64x64, and 128x128 for comparison. At 32x32 input size, 99.45% validation accuracy and 98.42% test accuracy with a network size of 24.2MB. With 64x64 image input, the obtained validation and test accuracies were 99.84% and 99.43% respectively, with a 60.8MB network size. When a 128x128 input size was given, the network size increased to 204MB while the train and test accuracies were 99.84% and 99.49%.

### **Recommendations**

It is possible to obtain different versions of the network by using different filters and different combinations of filters. The filter-based approach is not limited to the face mask dataset used in this research, it is possible to use it for feature extraction for any application. By trying different filters and their combinations, the preprocess involving neural networks can be used on different types of datasets to check performance variation between various datasets.

## REFERENCES

- Adusumalli, H., Kalyani, D., Sri, R. K., Pratapteja, M., & Rao, P. P. (2021, February). Face mask detection using OpenCV. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)* (pp. 1304-1309). IEEE.  
<https://doi.org/10.1109/ICICV50876.2021.9388375>
- Agarwal, C., Kaur, I., & Yadav, S. (2022, July). Hybrid CNN-SVM model for face mask detector to protect from COVID-19. In *Artificial Intelligence on Medical Data: Proceedings of International Symposium, ISMM 2021* (pp. 419-426). Singapore: Springer Nature Singapore. [https://doi.org/10.1007/978-981-19-0151-5\\_35](https://doi.org/10.1007/978-981-19-0151-5_35)
- Ahmad, N., & Dimililer, K. (2022, October). Brain tumor detection using convolutional neural network. In *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1032-1037). IEEE. <https://doi.org/10.1109/ISMSIT56059.2022.9932741>
- Asir, S., Dimililer, K., Kirsal-Ever, Y., Özsöz, M., & Shama, N. A. (2019, October). Electrochemical determination of potassium ferricyanide using artificial intelligence. In *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1-4). IEEE.  
<https://doi.org/10.1109/ISMSIT.2019.8932910>
- Atlam, M., Torkey, H., El-Fishawy, N., & Salem, H. (2021). Coronavirus disease 2019 (COVID-19): Survival analysis using deep learning and Cox regression model. *Pattern Analysis and Applications*, 24, 993-1005.  
<https://doi.org/10.1007/s10044-021-00958-0>
- Aydemir, E., Yalcinkaya, M. A., Barua, P. D., Baygin, M., Faust, O., Dogan, S., ... & Acharya, U. R. (2022). Hybrid deep feature generation for appropriate face mask use detection. *International journal of environmental research and public health*, 19(4), 1939. <https://doi.org/10.3390/ijerph19041939>
- Baluprithviraj, K. N., Bharathi, K. R., Chendhuran, S., & Lokeshwaran, P. (2021, March). Artificial intelligence based smart door with face mask detection. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 543-548). IEEE.  
<https://doi.org/10.1109/ICAIS50930.2021.9395807>

- Chavda, A., Dsouza, J., Badgujar, S., & Damani, A. (2021, April). Multi-stage CNN architecture for face mask detection. In *2021 6th International Conference for Convergence in Technology (i2ct)* (pp. 1-8). IEEE.  
<https://doi.org/10.1109/I2CT51068.2021.9418207>
- Cheng, V. C. C., Wong, S. C., Chuang, V. W. M., So, S. Y. C., Chen, J. H. K., Sridhar, S., ... & Yuen, K. Y. (2020). The role of community-wide wearing of face mask for control of coronavirus disease 2019 (COVID-19) epidemic due to SARS-CoV-2. *Journal of Infection*, *81*(1), 107-114.  
<https://doi.org/10.1016/j.jinf.2020.04.024>
- Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258). <https://doi.org/10.48550/arXiv.1610.02357>
- Crespo, F., Crespo, A., Sierra-Martínez, L. M., Peluffo-Ordóñez, D. H., & Morochocayamcela, M. E. (2022). A computer vision model to identify the incorrect use of face masks for COVID-19 awareness. *Applied Sciences*, *12*(14), 6924.  
<https://doi.org/10.3390/app12146924>
- Das, A., Ansari, M. W., & Basak, R. (2020, December). Covid-19 face mask detection using TensorFlow, Keras and OpenCV. In *2020 IEEE 17th India council international conference (INDICON)* (pp. 1-5). IEEE.  
<https://doi.org/10.1109/INDICON49873.2020.9342585>
- Dasari, N., & Reddy, B. R. (2023). Multi-scale lung tissue classification for interstitial lung diseases using learned Gabor filters. *Microsystem Technologies*, *29*(4), 599-607. <https://doi.org/10.1007/s00542-023-05413-0>
- Dey, S. K., Howlader, A., & Deb, C. (2020, December). MobileNet mask: a multi-phase face mask detection model to prevent person-to-person transmission of SARS-CoV-2. In *Proceedings of International Conference on Trends in Computational and Cognitive Engineering: Proceedings of TCCE 2020* (pp. 603-613). Singapore: Springer Singapore. [https://doi.org/10.1007/978-981-33-4673-4\\_49](https://doi.org/10.1007/978-981-33-4673-4_49)
- Dimililer, K., & Bush, I. J. (2017, September). Automated classification of fruits: pawpaw fruit as a case study. In *International conference on man-machine interactions* (pp. 365-374). Cham: Springer International Publishing.  
[https://doi.org/10.1007/978-3-319-67792-7\\_36](https://doi.org/10.1007/978-3-319-67792-7_36)

- Dimililer, K., Erdem, B. D., Kayali, D., & Olawale, O. P. (2024). Image preprocessing phase with artificial intelligence methods on medical images. In *Artificial Intelligence and Image Processing in Medical Imaging* (pp. 51-82). Academic Press. <https://doi.org/10.1016/B978-0-323-95462-4.00003-0>
- Dimililer, K., Ever, Y. K., & Ugur, B. (2016). ILTDS: Intelligent lung tumor detection system on ct images. In *Intelligent Systems Technologies and Applications 2016* (pp. 225-235). Springer International Publishing. [https://doi.org/10.1007/978-3-319-47952-1\\_17](https://doi.org/10.1007/978-3-319-47952-1_17)
- Dimililer, K., & Kayalı, D. (2021). Image enhancement in healthcare applications: A review. *Artificial Intelligence and Machine Learning for COVID-19*, 111-140. [https://doi.org/10.1007/978-3-030-60188-1\\_6](https://doi.org/10.1007/978-3-030-60188-1_6)
- Dimililer, K., & Kayali, D. (2023). Mask Detection and Categorization during the COVID-19 Pandemic Using Deep Convolutional Neural Network. *Ingeniería e Investigación*, 43(3), 1. <https://doi.org/10.15446/ing.investig.101817>
- Dimililer, K., Kayali, D., & Tackie, S. N. (2023, October). Power Demand Prediction of North Cyprus using Machine Learning Regressor Models. In *2023 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE. <https://doi.org/10.1109/ASYU58738.2023.10296786>
- Dimililer, K., & Sekeroglu, B. (2023). Skin lesion classification using cnn-based transfer learning model. *Gazi University Journal of Science*, 36(2), 660-673. <https://doi.org/10.35378/gujs.1063289>
- Dong, X., Xu, W., Miao, Z., Ma, L., Zhang, C., Yang, J., ... & Shen, J. (2022). Abandoning the bayer-filter to see in the dark. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 17431-17440). <https://doi.org/10.1109/CVPR52688.2022.01691>
- Fazeli Ardekani, P., Tale, S. Z., & Parseh, M. J. (2024). Face mask recognition using a custom CNN and data augmentation. *Signal, Image and Video Processing*, 18(1), 255-263. <https://doi.org/10.1007/s11760-023-02717-6>
- Gong, Y., Tang, W., Zhou, L., Yu, L., & Qiu, G. (2021, September). Quarter Laplacian filter for edge aware image processing. In *2021 IEEE International Conference on Image Processing (ICIP)* (pp. 1959-1963). IEEE. <https://doi.org/10.1109/ICIP42928.2021.9506503>

- Hammad, M. S., Ghoneim, V. F., Mabrouk, M. S., & Al-Atabany, W. I. (2023). A hybrid deep learning approach for COVID-19 detection based on genomic image processing techniques. *Scientific Reports*, *13*(1), 4003.  
<https://doi.org/10.1038/s41598-023-30941-0>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). <https://doi.org/10.48550/arXiv.1512.03385>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14* (pp. 630-645). Springer International Publishing.  
<https://doi.org/10.48550/arXiv.1603.05027>
- Howard, A. G. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.  
<https://doi.org/10.48550/arXiv.1704.04861>
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).  
<https://doi.org/10.48550/arXiv.1608.06993>
- Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments (*tech. rep. No. 07-49*). University of Massachusetts.
- Hussain, S., Yu, Y., Ayoub, M., Khan, A., Rehman, R., Wahid, J. A., & Hou, W. (2021). IoT and deep learning based approach for rapid screening and face mask detection for infection spread control of COVID-19. *Applied Sciences*, *11*(8), 3495. <https://doi.org/10.3390/app11083495>
- Ilgı, G. S., Kayali, D., Olawale, P., Erdem, B. D., Dimililer, K., & Kirsal-Ever, Y. (2022, September). Formal verification for security technologies in the blockchain with artificial intelligence: A survey. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-6). IEEE.  
<https://doi.org/10.1109/ASYU56188.2022.9925532>

- Jasti, V. D. P., Zamani, A. S., Arumugam, K., Naved, M., Pallathadka, H., Sammy, F., ... & Kaliyaperumal, K. (2022). Computational technique based on machine learning and image processing for medical image analysis of breast cancer diagnosis. *Security and communication networks*, 2022(1), 1918379. <https://doi.org/10.1155/2022/1918379>
- Jayaswal, R., & Dixit, M. (2023). AI-based face mask detection system: a straightforward proposition to fight with Covid-19 situation. *Multimedia Tools and Applications*, 82(9), 13241-13273. <https://doi.org/10.1007/s11042-022-13697-z>
- Kaur, G., Sharma, N., Malhotra, S., Devliyal, S., & Gupta, R. (2024, July). Enhancing Public Safety through Face Mask Detection using Xception Deep Learning Architecture. In *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)* (pp. 946-952). IEEE. <https://doi.org/10.1109/AIC61668.2024.10731076>
- Kayali, D., & Dimililer, K. (2023, December). Multi-filter-Based Image Pre-processing on Face Mask Detection Using Custom CNN Architecture. In *International Symposium on Intelligent Informatics* (pp. 29-36). Singapore: Springer Nature Singapore. [https://doi.org/10.1007/978-981-97-2147-4\\_3](https://doi.org/10.1007/978-981-97-2147-4_3)
- Kayali, D., Dimililer, K., & Sekeroglu, B. (2021, August). Face mask detection and classification for COVID-19 using deep learning. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)* (pp. 1-6). IEEE. <https://doi.org/10.1109/INISTA52262.2021.9548642>
- Kayali, D., Shama, N. A., Asir, S., & Dimililer, K. (2023). Machine learning-based models for the qualitative classification of potassium ferrocyanide using electrochemical methods. *The Journal of Supercomputing*, 79(11), 12472-12491. <https://doi.org/10.1007/s11227-023-05137-y>
- Kayali, D., Olawale, P., Kirsal-Ever, Y., & Dimililer, K. (2022, September). The effect of compressor-decompressor networks with different image sizes on mask detection using Convolutional Neural Networks-VGG-16. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE. <https://doi.org/10.1109/ASYU56188.2022.9925317>

- Khamlae, P., Sookhanaphibarn, K., & Choensawat, W. (2021, March). An application of deep-learning techniques to face mask detection during the COVID-19 pandemic. In *2021 IEEE 3rd global conference on life sciences and technologies (LifeTech)* (pp. 298-299). IEEE.  
<https://doi.org/10.1109/LifeTech52111.2021.9391922>
- Kodali, R. K., & Dhanekula, R. (2021, January). Face mask detection using deep learning. In *2021 international conference on computer communication and informatics (ICCCI)* (pp. 1-5). IEEE.  
<https://doi.org/10.1109/ICCCI50826.2021.9402670>
- Likhith, S. R., Itagi, S., Nithya, B. A., Bhuvaneshwari, P., & Josephine, R. (2024, August). FaceShroud Advanced Real-Time Mask Detection using Deep Learning. In *2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON)* (pp. 1-7). IEEE.  
<https://doi.org/10.1109/NMITCON62075.2024.10699295>
- Lv, H., Shan, P., Shi, H., & Zhao, L. (2022). An adaptive bilateral filtering method based on improved convolution kernel used for infrared image enhancement. *Signal, Image and Video Processing*, *16*(8), 2231-2237.  
<https://doi.org/10.1007/s11760-022-02188-1>
- Mar-Cupido, R., García, V., Rivera, G., & Sánchez, J. S. (2022). Deep transfer learning for the recognition of types of face masks as a core measure to prevent the transmission of COVID-19. *Applied Soft Computing*, *125*, 109207. <https://doi.org/10.1016/j.asoc.2022.109207>
- Methil, A. S. (2021, March). Brain tumor detection using deep learning and image processing. In *2021 international conference on artificial intelligence and smart systems (ICAIS)* (pp. 100-108). IEEE.  
<https://doi.org/10.1109/ICAIS50930.2021.9395823>
- Mohan, P., Paul, A. J., & Chirania, A. (2021). A tiny CNN architecture for medical face mask detection for resource-constrained endpoints. In *Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2021* (pp. 657-670). Singapore: Springer Singapore. [https://doi.org/10.1007/978-981-16-0749-3\\_52](https://doi.org/10.1007/978-981-16-0749-3_52)



- Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. *Sustainable cities and society*, 66, 102692. <https://doi.org/10.1016/j.scs.2020.102692>
- Narin, A., Kaya, C., & Pamuk, Z. (2021). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, 24, 1207-1220. <https://doi.org/10.1007/s10044-021-00984-y>
- Naufal, M. F., Kusuma, S. F., Prayuska, Z. A., Yoshua, A. A., Lauwoto, Y. A., Dinata, N. S., & Sugiarto, D. (2021). Comparative analysis of image classification algorithms for face mask detection. *Journal of Information Systems Engineering and Business Intelligence*, 7(1), 56-66. <https://doi.org/10.20473/jisebi.7.1.56-66>
- Nirmaladevi, J., Poornachandra, S., Jayavardhini, P., Ragsana, R. V., & Mahalakame, R. M. (2024, April). Deep Learning Based FPN and MT-CNN Face Mask Detection System. In *2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)* (pp. 1-5). IEEE. <https://doi.org/10.1109/ICONSTEM60960.2024.10568713>
- Pandey, V. (2020). Artificial intelligence based face mask detection system. *International Journal of Innovative Science and Research Technology*, 5(8), 467-470. <https://doi.org/10.38124/IJSRT20AUG410>
- Parikh, S., Shukla, D., Parekh, J., & Kotak, M. (2024, July). Deep Learning Method for Multi-Class Face-Mask Classification in Real-Time. In *2024 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)* (pp. 385-391). IEEE. <https://doi.org/10.1109/IAICT62357.2024.10617736>
- Pinki & Garg, S. (2020). Face mask detection system using deep learning. *International Journal for Modern Trends in Science and Technology*, 6(12), 161-164. <https://doi.org/10.46501/IJMTST061231>
- Prasad, T. G., Turukmane, A. V., Kumar, M. S., Madhavi, N. B., Sushama, C., & Neelima, P. (2022). Cnn Based Pathway Control To Prevent Covid Spread Using Face Mask And Body Temperature Detection. *Journal of Pharmaceutical Negative Results*, 1374-1381. <https://doi.org/10.47750/pnr.2022.13.S04.164>



- Putra, R. M., Yossy, E. H., Saputro, I. P., Pratama, D., & Prasandy, T. (2023, May). Face mask detection using convolutional neural network. In *2023 8th International Conference on Business and Industrial Research (ICBIR)* (pp. 133-138). IEEE. <https://doi.org/10.1109/ICBIR57571.2023.10147730>
- Rudraraju, S. R., Suryadevara, N. K., & Negi, A. (2020, September). Face mask detection at the fog computing gateway. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)* (pp. 521-524). IEEE. <https://doi.org/10.15439/2020F143>
- Sakshi, S., Gupta, A. K., Yadav, S. S., & Kumar, U. (2021, March). Face mask detection system using CNN. In *2021 International conference on advance computing and innovative technologies in engineering (ICACITE)* (pp. 212-216). IEEE. <https://doi.org/10.1109/ICACITE51222.2021.9404731>
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520). <https://doi.org/10.48550/arXiv.1801.04381>
- Sanjaya, S. A., & Rakhmawan, S. A. (2020, October). Face mask detection using MobileNetV2 in the era of COVID-19 pandemic. In *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)* (pp. 1-5). IEEE. <https://doi.org/10.1109/ICDABI51230.2020.9325631>
- Sen, S., & Patidar, H. (2020). Face mask detection system for COVID\_19 pandemic precautions using deep learning method. *International Journal of Emerging Technologies and Innovative Research*, 7(10), 16-21. <https://www.jetir.org/view?paper=JETIR2010003>
- Schmalfuss, J., Scheurer, E., Zhao, H., Karantzas, N., Bruhn, A., & Labate, D. (2023). Blind image inpainting with sparse directional filter dictionaries for lightweight CNNs. *Journal of Mathematical Imaging and Vision*, 65(2), 323-339. <https://doi.org/10.1007/s10851-022-01119-6>
- Sheikh, B. U. H., & Zafar, A. (2023). RRFMDs: rapid real-time face mask detection system for effective COVID-19 monitoring. *SN Computer Science*, 4(3), 288. <https://doi.org/10.1007/s42979-023-01738-9>

- Siddharth, D., Saini, D. K. J., & Singh, P. (2021). An efficient approach for edge detection technique using kalman filter with artificial neural network. *International Journal of Engineering*, 34(12), 2604-2610.  
<https://doi.org/10.5829/ije.2021.34.12c.04>
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.  
<https://doi.org/10.48550/arXiv.1409.1556>
- Singh, A., Jindal, V., Sandhu, R., & Chang, V. (2022). A scalable framework for smart COVID surveillance in the workplace using Deep Neural Networks and cloud computing. *Expert Systems*, 39(3), e12704.  
<https://doi.org/10.1111/exsy.12704>
- Srinivasan, S., Singh, R. R., Biradar, R. R., & Revathi, S. A. (2021, March). COVID-19 monitoring system using social distancing and face mask detection on surveillance video datasets. In *2021 International conference on emerging smart computing and informatics (ESCI)* (pp. 449-455). IEEE.  
<https://doi.org/10.1109/ESCI50559.2021.9396783>
- Snyder, S. E., & Husari, G. (2021, March). Thor: A deep learning approach for face mask detection to prevent the COVID-19 pandemic. In *SoutheastCon 2021* (pp. 1-8). IEEE.  
<https://doi.org/10.1109/SoutheastCon45413.2021.9401874>
- Suresh, K., Palangappa, M. B., & Bhuvan, S. (2021, January). Face mask detection by using optimistic convolutional neural network. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)* (pp. 1084-1089). IEEE. <https://doi.org/10.1109/ICICT50816.2021.9358653>
- Syafitri, M. D., Hidayat, R., & Nugroho, H. A. (2024, June). Face Mask Detection using Gabor Filter and Gamma Transform with Localized Feature Extraction. In *2024 International Conference on Smart Computing, IoT and Machine Learning (SIML)* (pp. 208-213). IEEE.  
<https://doi.org/10.1109/SIML61815.2024.10578117>
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 31, No. 1). <https://doi.org/10.48550/arXiv.1602.07261>

- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826). <https://doi.org/10.48550/arXiv.1512.00567>
- Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR. <https://doi.org/10.48550/arXiv.1905.11946>
- Tayal, A., Gupta, J., Solanki, A., Bisht, K., Nayyar, A., & Masud, M. (2022). DL-CNN-based approach with image processing techniques for diagnosis of retinal diseases. *Multimedia systems*, 28(4), 1417-1438. <https://doi.org/10.1007/s00530-021-00769-7>
- TUN, N., & Myat, A. M. (2024). Proposed Activation Function Based Deep Learning Approach for Real-Time Face Mask Detection System. *International Journal of Electrical Engineering and Computer Science*, 6, 211-217. <https://doi.org/10.37394/232027.2024.6.25>
- Umer, M., Sadiq, S., Alhebshi, R. M., Alsubai, S., Al Hejaili, A., Nappi, M., & Ashraf, I. (2023). Face mask detection using deep convolutional neural network and multi-stage image processing. *Image and Vision Computing*, 133, 104657. <https://doi.org/10.1016/j.imavis.2023.104657>
- Venkateswarlu, I. B., Kakarla, J., & Prakash, S. (2020, December). Face mask detection using mobilenet and global pooling block. In *2020 IEEE 4th conference on information & communication technology (CICT)* (pp. 1-5). IEEE. <https://doi.org/10.1109/CICT51604.2020.9312083>
- Vijitkunsawat, W., & Chantngarm, P. (2020, October). Study of the performance of machine learning algorithms for face mask detection. In *2020-5th international conference on information technology (InCIT)* (pp. 39-43). IEEE. <https://doi.org/10.1109/InCIT50588.2020.9310963>
- Viknesh, C. K., Kumar, P. N., Seetharaman, R., & Anitha, D. (2023). Detection and classification of melanoma skin cancer using image processing technique. *Diagnostics*, 13(21), 3313. <https://doi.org/10.3390/diagnostics13213313>
- Wang, Z., Wang, G., Huang, B., Xiong, Z., Hong, Q., Wu, H., ... & Liang, J. (2020). Masked Face Recognition Dataset and Application. *arXiv preprint arXiv:2003.09093*. <https://doi.org/10.48550/arXiv.2003.09093>

- Wang, Z., Wang, P., Louis, P. C., Wheless, L. E., & Huo, Y. (2021). Wearmask: Fast in-browser face mask detection with serverless edge computing for covid-19. *arXiv preprint arXiv:2101.00784*.  
<https://doi.org/10.48550/arXiv.2101.00784>
- Wu, P., Li, H., Zeng, N., & Li, F. (2022). FMD-Yolo: An efficient face mask detection method for COVID-19 prevention and control in public. *Image and vision computing*, *117*, 104341. <https://doi.org/10.1016/j.imavis.2021.104341>
- Yadav, S. (2020). Deep learning based safe social distancing and face mask detection in public areas for covid-19 safety guidelines adherence. *International Journal for Research in Applied Science and Engineering Technology*, *8*(7), 1368-1375. <https://doi.org/10.22214/ijraset.2020.30560>
- Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8697-8710).  
<https://doi.org/10.48550/arXiv.1707.07012>

## APPENDICES

## Appendix A

## Journal Publication

INGENIERÍA E INVESTIGACIÓN VOL. 43 NO. 3, DECEMBER - 2023 (101817)

Research article / Educational Engineering

<http://doi.org/10.15446/ing.investig.101817>

## Mask Detection and Categorization during the COVID-19 Pandemic Using Deep Convolutional Neural Network

### Detección y categorización de máscaras durante la pandemia del COVID-19 utilizando una red neuronal convolucional profunda

Kamil Dimililer<sup>1,2,3</sup>, and Devrim Kayali<sup>2</sup>**ABSTRACT**

With COVID-19 spreading all over the world and restricting our daily lives, the use of face masks has become very important, as it is an efficient way of slowing down the spread of the virus and an important piece to continue our daily tasks until vaccination is completed. People have been fighting this disease for a long time, and they are bored with the precautions, so they act carelessly. In this case, automatic detection systems are very important to keep the situation under control. In this research, deep learning models are trained with as little input data as possible in order to obtain an accurate face mask-wearing condition classification. These classes are mask-correct, mask wrong, and no mask, which refers to proper face mask use, improper face mask use, and no mask use, respectively. DenseNets, EfficientNets, InceptionResNetV2, InceptionV3, MobileNets, NasNets, ResNets, VGG16, VGG19, and Xception are the networks used in this study. The highest accuracy was obtained by the InceptionResNetV2 and Xception networks, with 99,6%. When other performance parameters are taken into consideration, the Xception network is a step forward. VGG16 and VGG19 also show an accuracy rate over 99%, with 99,1 and 99,4%, respectively. These two networks also had higher FPS and the two lowest initialization times during implementation. A comparison with recent studies was also carried out to evaluate the obtained accuracy. It was found that a higher accuracy can be obtained with the possible minimum input size.

**Keywords:** COVID-19, mask detection, deep learning, classification

**RESUMEN**

Con el COVID-19 extendiéndose por todo el mundo y restringiendo nuestra vida diaria, el uso de mascarillas se ha vuelto muy importante, pues es una forma eficiente de frenar la propagación del virus, y una pieza importante para continuar con nuestras tareas diarias hasta que se complete la vacunación. La gente ha estado luchando contra la enfermedad durante mucho tiempo y se aburre con las precauciones, por lo que actúa con descuido. En este caso, los sistemas de detección automática son muy importantes para mantener la situación bajo control. En esta investigación se entrenan modelos de aprendizaje profundo con el mínimo de datos de entrada posibles para obtener una clasificación precisa de las condiciones de uso de las mascarillas. Estas clases son maskcorrect, maskwrong y nomask, que se refieren al uso adecuado, a un uso inadecuado y al no uso de la mascarilla facial, respectivamente. DenseNets, EfficientNets, InceptionResNetV2, InceptionV3, MobileNets, NasNets, ResNets, VGG16, VGG19 y Xception son las redes utilizadas en este estudio. La mayor precisión la obtuvieron las redes InceptionResNetV2 y Xception, con un 99,6 %. Cuando se tienen en cuenta otros parámetros de rendimiento, la red Xception un paso adelante. VGG16 y VGG19 presentan una tasa de precisión superior al 99 %, con 99,1 y 99,4 % respectivamente. Estas dos redes también presentaron FPS más altos y los dos tiempos de inicialización más bajos en la implementación. También se realizó una comparación con estudios recientes para evaluar la precisión obtenida. Se encontró que se puede obtener una mayor precisión con el mínimo tamaño de entrada posible.

**Palabras clave:** COVID-19, detección de mascarillas, aprendizaje profundo, clasificación

**Received:** March 25th 2022

**Accepted:** October 31st 2022

**Introduction**

Our lifestyles started changing with the advent of COVID-19, a very dangerous virus that quickly spread all over the world. Due to its high infection rate and severity, precautions were taken at a global level, and restrictions were put into effect. The use of face masks was one of the most important and effective ways for slowing down the infection from the very beginning of the pandemic (Cheng *et al.*, 2020).

Despite the fact that vaccination continues all over the world, the use of masks is still important, since sufficient vaccination has not yet been reached and the virus can mutate and change. As we are trying to return to our daily lives while coexisting with the disease, we still have to take

some precautions to avoid any rapid spread. This means that serious control measures are needed, especially in public places. Automated control is key to obtain fast, reliable, and continuously working systems. Artificial intelligence (AI) makes it possible to obtain such systems in many different areas. AI-based image processing has been applied in various cases (Dimililer, 2022; Dimililer and Kayali, 2021;

<sup>1</sup>Department of Electrical and Electronic Engineering. Near East University, Nicosia, North Cyprus, Mersin 10 Turkey. E-mail: kamil.dimililer@neu.edu.tr

<sup>2</sup>Department of Electrical and Electronic Engineering. Near East University, Nicosia, North Cyprus, Mersin 10 Turkey. E-mail: devrim.kayali@ktemo.org

<sup>3</sup>Applied Artificial Intelligence Research Centre. Near East University, Nicosia, North Cyprus, Mersin 10 Turkey.



Attribution 4.0 International (CC BY 4.0) Share - Adapt



## Appendix B

### Conference Publication

# Face Mask Detection and Classification for COVID-19 using Deep Learning

1<sup>st</sup> Devrim Kayali  
*Electrical & Electronic Engineering*  
*Near East University*  
 Nicosia, Cyprus, Mersin 10 Turkey  
 devrim.kayali@ktemo.org

2<sup>nd</sup> Kamil Dimililer  
*Electrical & Electronic Engineering*  
*Near East University*  
 Nicosia, Cyprus, Mersin 10 Turkey  
 kamil.dimililer@neu.edu.tr

3<sup>rd</sup> Boran Sekeroglu  
*Information Systems Engineering*  
*Near East University*  
 Nicosia, Cyprus, Mersin 10 Turkey  
 boran.sekeroglu@neu.edu.tr

**Abstract**—With the emergence of COVID-19, our lifestyles have changed. Because of its high infectivity and high severity, fighting with this virus has become a global problem in no time. Minimizing the speed of spread and keeping the numbers under control is a way to save more people by taking care of them better and also have more time spent on research to find a cure such as medicine or vaccine which will put an end to this situation. During this time, personal protection like using face masks is very important since it protects people and others close to them and significantly reduces the risk of infection on correct usage. Unfortunately, some people can act careless or reckless which puts many people at risk. This leads us to use automated systems in crowded places to detect those who do not follow the rules. When it comes to automated systems, artificial intelligence-related applications are favorable for supporting humans. In this paper, at first, a dataset was obtained by adding face masks to the existing Labeled Faces in the Wild (LFW) dataset to detect three mask-wearing conditions; correct, wrong and no mask. Then NASNetMobile and ResNet50 networks were trained using the considered dataset. The ResNet50 outperformed the NASNetMobile by achieving 92% detection accuracy.

**Index Terms**—COVID-19, Face Mask, Mask Detection, Deep Learning

## I. INTRODUCTION

Sustainable development is a key concept and solution in creating a promising and prosperous future for human societies. Human being always faces epidemic diseases in both predicted or unpredicted ways. COVID-19, which is the new version of Coronavirus, is one of the epidemic diseases that affect human life. Many research has been made by various authors considering both the medical side as well as image processing with artificial intelligence.

Pirouz et al. suggested a binary classification using AI and Regression analysis. The authors selected Hubei in China as a case study and used group method data handling type of neural network. Various kinds of weather factors were considered for 30 days. For the comparison of the fluctuations of daily weather parameters and the trend of confirmed cases, regression analysis was done. When the confirmed cases are considered, in the main case study relative humidity affected positively with an average of 77.9% and degrees Celsius affected negatively with an average temperature of 15.4 [1].

978-1-6654-3603-8/21/\$31.00 ©2021 IEEE

Lalmuanawma et al. reviewed the applications of machine learning and artificial intelligence for the COVID-19 pandemic. The authors discussed the AI and ML in the field of forecasting, predicting, screening, contact tracing, and drug development for SARS-CoV-2 and its related epidemic [2].

Naude W. Studied the Artificial Intelligence against COVID-19. Early evaluation of AI is considered for COVID-19. The author suggested that AI was not impactful against COVID-19. When the constraints such as lack of data or too much data are considered, a careful balance between data privacy and public health will be needed to be overcome [3].

Another research has been made by Pourhomayoun and Shakibi, to predict the Morality Risks in patients with COVID-19 using AI, for decision making. The authors developed a predictive model to determine the health risk and predict the mortality risk of the patients. The authors achieved 93% of overall accuracy in predicting the mortality rate by comparing Logistic regression, Random forest, Support vector machine, Decision tree, K-nearest neighbor and Artificial neural networks [4].

COVID-19 virus can be easily spread among people, especially in closed areas, so wearing a mask is necessary to prevent the virus reach the humans during coronavirus epidemic. This situation makes the conventional face recognition technology ineffective in many cases. The researchers started to improve the recognition performance of the existing Technologies considering the masked faces. Wang et al. mentioned that most of the face recognition systems are based on deep learning with a large number of face samples [5].

## II. LITERATURE REVIEW

Matuscheck et al. studied the benefits, as well as the risks during the COVID-19 crisis considering the face masks. The authors performed an extensive query of the most recent publications that address the prevention of viral infections while the face masks in the community prevent the spread of the infection. The authors suggested that there are some clinically relevant scenarios where the use of MNC necessitates more defined recommendations [6].

Bubbico et al. studied the community use of face masks against the spread of COVID-19. The authors suggested that

## Appendix C

### Conference Publication

# Multi Filter Based Image Pre-Processing on Face Mask Detection using Custom CNN Architecture

Devrim Kayali<sup>1</sup> and Kamil Dimililer<sup>2</sup>

<sup>1</sup> Electrical & Electronic Engineering  
Research Center for Science, Technology and Engineering (BILTEM)  
Near East University, Nicosia, N. Cyprus, Via Mersin 10, Turkey  
[devrim.kayali@ktemo.org](mailto:devrim.kayali@ktemo.org),

<sup>2</sup> Electrical & Electronic Engineering  
Applied Artificial Intelligence Research Centre (AAIRC)  
Research Center for Science, Technology and Engineering (BILTEM)  
Near East University, Nicosia, N. Cyprus, Via Mersin 10, Turkey  
[kamil.dimililer@neu.edu.tr](mailto:kamil.dimililer@neu.edu.tr)

**Abstract.** After the COVID-19 pandemic, the effectiveness of face mask usage in such an environment has been seen. This situation triggered much research and experiments on this subject. Most of this research was done by using artificial intelligence. This paper focuses on a filter-based approach using a convolutional neural network for classification of face mask usage, which are mask correct, mask wrong, and no mask. 17 chosen filters are applied to the input images as an image pre-processing phase which results in 17 input images per image. Our customized network takes these 17 images as input and eliminates them into a single image before flattening and feeding to the dense layers. For training, 5 learning rates were tried starting from  $10^{-7}$  and increasing 10 times for each try stopping at  $10^{-3}$ . Results showed that increasing the learning rate shortened the training time and increased the accuracy. The highest test accuracy was obtained at  $10^{-3}$  learning rate with 98.86%. At  $10^{-4}$  learning rate, the second good result 98.16% was obtained.

**Keywords:** Face Mask, Filtering, Image Processing, CNN, Classification

## 1 Introduction

When COVID-19 came up and spread rapidly and became a pandemic that affected the whole world, we had to take precautions to take the spread under control and protect ourselves individually. One of the effective methods for this was to use face masks. When it comes to public places, it is important to check the face mask usage since improper and no usage can increase the spread and affect other's lives. When the situation is serious and constantly running applications are needed with minimum errors, automated detection systems become

## Appendix D

### Python Code of the Proposed Network

```

class ReLU(Layer):
    def __init__(self, **kwargs):
        super(ReLU, self).__init__(**kwargs)

    def call(self, inputs):
        # Apply the relu function to the input image
        outputs = tf.nn.relu(inputs)
        return outputs

inputimagewidth,inputimageheight,inputimagechannel=inputshape
input_layer = keras.layers.Input(shape=(inputimagewidth,inputimageheight,inputimagechannel), name='input_layer')

filter_outputs=[]
# Apply each filter independently
for i in range(len(filterarray)):
    initfilter = tf.constant_initializer(filterarray[i])
    filter_output = tf.keras.layers.Conv2D(1, kernel_size=(3, 3), padding='same', kernel_initializer=initfilter
                                           ,use_bias=False,name="filter"+str(i))(input_layer)
    filter_outputs.append(filter_output)
# Concatenate the filter outputs
filter_output_layer = tf.keras.layers.Concatenate(axis=-1)(filter_outputs)

# Conv2D layer
if inputimagewidth>=64:
    convout=Conv2D(1,(32,32),(1,1),padding="same")(filter_output_layer)
else:
    convout=Conv2D(1,(int(inputimagewidth/2),int(inputimageheight/2)),(1,1),padding="same")(filter_output_layer)

x=Flatten()(convout)
x=Dense(1024, activation='relu')(x)
x=Dense(1024, activation='relu')(x)
out=Dense(categories,activation='softmax')(x)
model = Model(input_layer,out)

# Loop over the layers of the model
for layer in model.layers:
    # If the layer name contains 'filter', set it to not trainable
    if 'filter' in layer.name:
        layer.trainable = False
    # Otherwise, leave the layer as it is
    else:
        pass

```



## Appendix E

### Proposed Network with 32x32 Input Image Resolution

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 32, 32, 1)]	0	[]
filter0 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter1 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter2 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter3 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter4 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter5 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter6 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter7 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter8 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter9 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter10 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter11 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter12 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter13 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter14 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter15 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
filter16 (Conv2D)	(None, 32, 32, 1)	9	['input_layer[0][0]']
ReLU0 (ReLU)	(None, 32, 32, 1)	0	['filter0[0][0]']
ReLU1 (ReLU)	(None, 32, 32, 1)	0	['filter1[0][0]']
ReLU2 (ReLU)	(None, 32, 32, 1)	0	['filter2[0][0]']
ReLU3 (ReLU)	(None, 32, 32, 1)	0	['filter3[0][0]']
ReLU4 (ReLU)	(None, 32, 32, 1)	0	['filter4[0][0]']
ReLU5 (ReLU)	(None, 32, 32, 1)	0	['filter5[0][0]']
ReLU6 (ReLU)	(None, 32, 32, 1)	0	['filter6[0][0]']
ReLU7 (ReLU)	(None, 32, 32, 1)	0	['filter7[0][0]']
ReLU8 (ReLU)	(None, 32, 32, 1)	0	['filter8[0][0]']
ReLU9 (ReLU)	(None, 32, 32, 1)	0	['filter9[0][0]']
ReLU10 (ReLU)	(None, 32, 32, 1)	0	['filter10[0][0]']

ReLU11 (ReLU)	(None, 32, 32, 1)	0	['filter11[0][0]']
ReLU12 (ReLU)	(None, 32, 32, 1)	0	['filter12[0][0]']
ReLU13 (ReLU)	(None, 32, 32, 1)	0	['filter13[0][0]']
ReLU14 (ReLU)	(None, 32, 32, 1)	0	['filter14[0][0]']
ReLU15 (ReLU)	(None, 32, 32, 1)	0	['filter15[0][0]']
ReLU16 (ReLU)	(None, 32, 32, 1)	0	['filter16[0][0]']
concatenate (Concatenate)	(None, 32, 32, 17)	0	['ReLU0[0][0]', 'ReLU1[0][0]', 'ReLU2[0][0]', 'ReLU3[0][0]', 'ReLU4[0][0]', 'ReLU5[0][0]', 'ReLU6[0][0]', 'ReLU7[0][0]', 'ReLU8[0][0]', 'ReLU9[0][0]', 'ReLU10[0][0]', 'ReLU11[0][0]', 'ReLU12[0][0]', 'ReLU13[0][0]', 'ReLU14[0][0]', 'ReLU15[0][0]', 'ReLU16[0][0]']
conv2d (Conv2D)	(None, 32, 32, 1)	4353	['concatenate[0][0]']
flatten (Flatten)	(None, 1024)	0	['conv2d[0][0]']
dense (Dense)	(None, 1024)	1049600	['flatten[0][0]']
dense_1 (Dense)	(None, 1024)	1049600	['dense[0][0]']
dense_2 (Dense)	(None, 3)	3075	['dense_1[0][0]']

```

=====
Total params: 2,106,781
Trainable params: 2,106,628
Non-trainable params: 153

```

## Appendix F

## Proposed Network with 64x64 Input Image Resolution

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 64, 64, 1)]	0	[]
filter0 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter1 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter2 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter3 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter4 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter5 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter6 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter7 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter8 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter9 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter10 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter11 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter12 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter13 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter14 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter15 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
filter16 (Conv2D)	(None, 64, 64, 1)	9	['input_layer[0][0]']
ReLU0 (ReLU)	(None, 64, 64, 1)	0	['filter0[0][0]']
ReLU1 (ReLU)	(None, 64, 64, 1)	0	['filter1[0][0]']
ReLU2 (ReLU)	(None, 64, 64, 1)	0	['filter2[0][0]']
ReLU3 (ReLU)	(None, 64, 64, 1)	0	['filter3[0][0]']
ReLU4 (ReLU)	(None, 64, 64, 1)	0	['filter4[0][0]']
ReLU5 (ReLU)	(None, 64, 64, 1)	0	['filter5[0][0]']
ReLU6 (ReLU)	(None, 64, 64, 1)	0	['filter6[0][0]']
ReLU7 (ReLU)	(None, 64, 64, 1)	0	['filter7[0][0]']
ReLU8 (ReLU)	(None, 64, 64, 1)	0	['filter8[0][0]']
ReLU9 (ReLU)	(None, 64, 64, 1)	0	['filter9[0][0]']
ReLU10 (ReLU)	(None, 64, 64, 1)	0	['filter10[0][0]']

ReLU11 (ReLU)	(None, 64, 64, 1)	0	['filter11[0][0]']
ReLU12 (ReLU)	(None, 64, 64, 1)	0	['filter12[0][0]']
ReLU13 (ReLU)	(None, 64, 64, 1)	0	['filter13[0][0]']
ReLU14 (ReLU)	(None, 64, 64, 1)	0	['filter14[0][0]']
ReLU15 (ReLU)	(None, 64, 64, 1)	0	['filter15[0][0]']
ReLU16 (ReLU)	(None, 64, 64, 1)	0	['filter16[0][0]']
concatenate (Concatenate)	(None, 64, 64, 17)	0	['ReLU0[0][0]', 'ReLU1[0][0]', 'ReLU2[0][0]', 'ReLU3[0][0]', 'ReLU4[0][0]', 'ReLU5[0][0]', 'ReLU6[0][0]', 'ReLU7[0][0]', 'ReLU8[0][0]', 'ReLU9[0][0]', 'ReLU10[0][0]', 'ReLU11[0][0]', 'ReLU12[0][0]', 'ReLU13[0][0]', 'ReLU14[0][0]', 'ReLU15[0][0]', 'ReLU16[0][0]']
conv2d (Conv2D)	(None, 64, 64, 1)	17409	['concatenate[0][0]']
flatten (Flatten)	(None, 4096)	0	['conv2d[0][0]']
dense (Dense)	(None, 1024)	4195328	['flatten[0][0]']
dense_1 (Dense)	(None, 1024)	1049600	['dense[0][0]']
dense_2 (Dense)	(None, 3)	3075	['dense_1[0][0]']

```

=====
Total params: 5,265,565
Trainable params: 5,265,412
Non-trainable params: 153

```

## Appendix G

### Proposed Network with 128x128 Input Image Resolution

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 128, 128, 1 )]	0	[]
filter0 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter1 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter2 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter3 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter4 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter5 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter6 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter7 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter8 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter9 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter10 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter11 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter12 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter13 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter14 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter15 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
filter16 (Conv2D)	(None, 128, 128, 1)	9	['input_layer[0][0]']
ReLU0 (ReLU)	(None, 128, 128, 1)	0	['filter0[0][0]']
ReLU1 (ReLU)	(None, 128, 128, 1)	0	['filter1[0][0]']
ReLU2 (ReLU)	(None, 128, 128, 1)	0	['filter2[0][0]']
ReLU3 (ReLU)	(None, 128, 128, 1)	0	['filter3[0][0]']
ReLU4 (ReLU)	(None, 128, 128, 1)	0	['filter4[0][0]']
ReLU5 (ReLU)	(None, 128, 128, 1)	0	['filter5[0][0]']
ReLU6 (ReLU)	(None, 128, 128, 1)	0	['filter6[0][0]']
ReLU7 (ReLU)	(None, 128, 128, 1)	0	['filter7[0][0]']
ReLU8 (ReLU)	(None, 128, 128, 1)	0	['filter8[0][0]']
ReLU9 (ReLU)	(None, 128, 128, 1)	0	['filter9[0][0]']

ReLU10 (ReLU)	(None, 128, 128, 1)	0	['filter10[0][0]']
ReLU11 (ReLU)	(None, 128, 128, 1)	0	['filter11[0][0]']
ReLU12 (ReLU)	(None, 128, 128, 1)	0	['filter12[0][0]']
ReLU13 (ReLU)	(None, 128, 128, 1)	0	['filter13[0][0]']
ReLU14 (ReLU)	(None, 128, 128, 1)	0	['filter14[0][0]']
ReLU15 (ReLU)	(None, 128, 128, 1)	0	['filter15[0][0]']
ReLU16 (ReLU)	(None, 128, 128, 1)	0	['filter16[0][0]']
concatenate (Concatenate)	(None, 128, 128, 17)	0	['ReLU0[0][0]', 'ReLU1[0][0]', 'ReLU2[0][0]', 'ReLU3[0][0]', 'ReLU4[0][0]', 'ReLU5[0][0]', 'ReLU6[0][0]', 'ReLU7[0][0]', 'ReLU8[0][0]', 'ReLU9[0][0]', 'ReLU10[0][0]', 'ReLU11[0][0]', 'ReLU12[0][0]', 'ReLU13[0][0]', 'ReLU14[0][0]', 'ReLU15[0][0]', 'ReLU16[0][0]']
conv2d (Conv2D)	(None, 128, 128, 1)	17409	['concatenate[0][0]']
flatten (Flatten)	(None, 16384)	0	['conv2d[0][0]']
dense (Dense)	(None, 1024)	16778240	['flatten[0][0]']
dense_1 (Dense)	(None, 1024)	1049600	['dense[0][0]']
dense_2 (Dense)	(None, 3)	3075	['dense_1[0][0]']

```

=====
Total params: 17,848,477
Trainable params: 17,848,324
Non-trainable params: 153

```

## Appendix H

### Turnitin Similarity Report

Similarities: Abstract %0, Chapters %8, Conclusion %0

**turnitin** [All Classes](#) [Join Account \(TA\)](#) [Quick Submit](#)

NOW VIEWING HOME - QUICK SUBMIT

**About this page**  
This is your assignment inbox. To view a paper, select the paper's title. To view a Similarity Report, select the paper's Similarity Report icon in the similarity column. A ghosted icon indicates that the Similarity Report has not yet been generated.

**Yakin Dogu Universitesi**  
[QUICK SUBMIT](#) | [NOW VIEWING ALL PAPERS](#) ▼

<input type="checkbox"/>	AUTHOR	TITLE	SIMILARITY	FILE	PAPER ID	DATE
<input type="checkbox"/>	Devrim Kayali	ABS 22012025	0%		2569011910	22-Jan-2025
<input type="checkbox"/>	Devrim Kayali	ALLCH 22012025	8%		2569011915	22-Jan-2025
<input type="checkbox"/>	Devrim Kayali	CNC 22012025	0%		2569011913	22-Jan-2025

## CV

**1. Name - Surname:** Devrim Kayalı

**2. Title:** PhD

**3. Educational Background:**

Degree	Department/Program	University	Year
Bachelor's	Electrical and Electronics Engineering	Cukurova University	2012-2016
Master's	Electrical and Electronics Engineering	Cukurova University	2017-2020
PhD	Electrical and Electronic Engineering	Near East University	2020-2025

**4. Publications**

**7.1. Journal Publications**

Dimililer, K., & Kayalı, D. (2023). Mask Detection and Categorization during the COVID-19 Pandemic Using Deep Convolutional Neural Network. *Ingeniería e Investigación*, 43(3), 1.

Kayalı, D., Shama, N. A., Asir, S., & Dimililer, K. (2023). Machine learning-based models for the qualitative classification of potassium ferrocyanide using electrochemical methods. *The Journal of Supercomputing*, 79(11), 12472-12491.

AŞIR, S., SHAMA, N. A., SALEEM, N., KAYALI, D., & DİMİLİLER, K. (2025). Machine Learning Models Approach for the Quantitative Classification of Ferricyanide Compound Using Electrochemical Detection with CPE-Fe<sub>3</sub>O<sub>4</sub>NPs. *Turkish Journal of Electrical Engineering and Computer Sciences*, 33(1), 15-31.

**7.2. Conference Publications**

Kayalı, D., Dimililer, K., & Sekeroglu, B. (2021, August). Face mask detection and classification for COVID-19 using deep learning. In *2021 International Conference on INnovations in Intelligent Systems and Applications (INISTA)* (pp. 1-6). IEEE.



Ilgi, G. S., Kayali, D., Olawale, P., Erdem, B. D., Dimililer, K., & Kirsal-Ever, Y. (2022, September). Formal verification for security technologies in the blockchain with artificial intelligence: A survey. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-6). IEEE.

Kayali, D., Olawale, P., Kirsal-Ever, Y., & Dimililer, K. (2022, September). The effect of compressor-decompressor networks with different image sizes on mask detection using Convolutional Neural Networks-VGG-16. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE.

Dimililer, K., Kayali, D., & Tackie, S. N. (2023, October). Power Demand Prediction of North Cyprus using Machine Learning Regressor Models. In *2023 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE.

Kayali, D., & Dimililer, K. (2023, December). Multi-filter-Based Image Pre-processing on Face Mask Detection Using Custom CNN Architecture. In *International Symposium on Intelligent Informatics* (pp. 29-36). Singapore: Springer Nature Singapore.

Orish, G. C., Tackie, S. N., Kayali, D., Alpan, K., Vafaei, L. E., & Dimililer, K. (2024, July). Comparative Analysis of Machine Learning Models for Solar Power Generation Prediction. In *2024 10th International Conference on Smart Computing and Communication (ICSCC)* (pp. 455-459). IEEE.

### **7.3. Book Chapters**

Dimililer, K., & Kayali, D. (2021). Image enhancement in healthcare applications: A review. *Artificial Intelligence and Machine Learning for COVID-19*, 111-140.

Dimililer, K., Erdem, B. D., Kayali, D., & Olawale, O. P. (2024). Image preprocessing phase with artificial intelligence methods on medical images. In *Artificial Intelligence and Image Processing in Medical Imaging* (pp. 51-82). Academic Press.