VISION TRANSFORMER-BASED DEEP LEARNING TECHNIQUES FOR IMPROPER FACE MASK-WEARING DETECTION

A. S. M. MUNTAHEEN (SN. 0419140003)

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DHAKA, BANGLADESH

MARCH 2024

VISION TRANSFORMER-BASED DEEP LEARNING TECHNIQUES FOR IMPROPER FACE MASK-WEARING DETECTION

M.Sc. Engineering Thesis

By

A. S. M. MUNTAHEEN (0419140003)

Approved as to style and content by the Board of Examination on 11 March, 2024:

_____ Dr Nusrat Sharmin Assistant Professor of Computer Science and Engineering Military Institute of Science and Technology, Dhaka	Chairman (Supervisor) Board of Examination
_____ Dr Md Forhad Rabbi Professor of Computer Science and Engineering Shahjalal University of Science and Technology, Sylhet	Member (External) Board of Examination
_____ Lt Col Muhammad Nazrul Islam, PhD Associate Professor (Instructor Class-A) of Computer Science & Engineering Military Institute of Science and Technology, Dhaka	Member (Internal) Board of Examination
_____ Brig Gen Mohammad Sajjad Hossain Acting Head of Computer Science and Engineering Military Institute of Science and Technology, Dhaka	Head of the Department Member (Ex-officio)

Department of Computer Science and Engineering, MIST, Dhaka

VISION TRANSFORMER-BASED DEEP LEARNING TECHNIQUES FOR IMPROPER FACE MASK-WEARING DETECTION

DECLARATION

I hereby declare that the study reported in this thesis entitled as above is my own original work and has not been submitted before (fully or partially) anywhere for any degree or other purposes. Further, I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged and/or cited in the reference section.

A. S. M. Muntaheen

Department of Computer Science and Engineering, MIST, Dhaka

ABSTRACT

VISION TRANSFORMER-BASED DEEP LEARNING TECHNIQUES FOR IMPROPER FACE MASK-WEARING DETECTION

COVID-19 pandemic causes a global catastrophe that remarkably affects individual lives and society, as well as the economy. The world has taken numerous defenses against this contagious disease and using face masks is one of the most crucial defense mechanisms. Effective prevention relies on proper face mask use, yet less than 25% of individuals adhere to correct usage. The prevalent method for face mask detection involves image processing, machine learning, and deep learning; notably, the Vision Transformer (ViT) base model has outperformed traditional deep learning models in making a significant impact in various domains. The exploration of ViT model in face mask detection is yet to be explored. This paper proposes to apply a most recent deep learning-based image classification model named ViT model to automatically detect improper face mask-wearing, i.e., whether the face masks are being worn correctly or not. The ViT base model shows a significant impact on incorrect face mask detection. The experiment has been conducted on a large custom dataset consisting of 2,03,780 digital images with 03 class labels namely 'With Mask' (when people are wearing the mask properly), 'Without Mask' (when people are not wearing the mask), and 'Incorrect Mask' (when people are not wearing the mask properly) to train, validate and test ViT model that can classify the use of face masks correctly. The results show that the accuracy achieved with the pre-trained ViT model is highly remarkable. Furthermore, the same experiment has been conducted on the Convolutional Neural Network (CNN) model with the same dataset consisting of 03 class labels. Then, the comparison has been done between the CNN model results and the ViT model results. The findings indicate that the ViT model exhibited faster training times and higher training accuracy, making it a more time-efficient option for incorrect face mask identification. This advantage in training time can be crucial for real-time applications and scenarios requiring quick response and decision-making. These findings contribute to advancing the field of image classification and offer valuable insights for future research and the development of improved image classification systems. The extended experiments involve five distinct CNN architectures—XCEPTION, MOBILENETV2, VGG16, INCEPTION, and ResNet-50—utilizing a smaller dataset consisting of 2079 digital images with the same 03 class labels. The results demonstrate that the ViT model outperforms all other models. In

conclusion, this study establishes that the ViT model achieves the highest accuracy among the other evaluated models.

সারসংক্ষেপ

VISION TRANSFORMER-BASED DEEP LEARNING TECHNIQUES FOR IMPROPER FACE MASK-WEARING DETECTION

কোভিড-১৯ (covid-19) মহামারী একটি বিশ্বব্যাপী বিপর্যয় ঘটায় যা উল্লেখযোগ্যভাবে ব্যক্তি জীবন এবং সমাজের পাশাপাশি অর্থনীতিকে প্রভাবিত করে। বিশ্ব এই সংক্রামক রোগের বিরুদ্ধে অসংখ্য প্রতিরক্ষা ব্যবস্থা নিয়েছে এবং ফেস মাস্ক (face mask) ব্যবহার করা সবচেয়ে গুরুত্বপূর্ণ প্রতিরক্ষা ব্যবস্থাগুলির মধ্যে একটি। সঠিক ফেস মাস্ক ব্যবহারের উপর কার্যকর প্রতিরোধ নির্ভর করে, তবুও ২৫% এরও কম ব্যক্তি সঠিক ব্যবহার মেনে চলে। ফেস মাস্ক সনাক্তকরণের জন্য প্রচলিত পদ্ধতির মধ্যে রয়েছে ইমেজ প্রসেসিং (image processing), মেশিন লার্নিং (machine learning) এবং ডিপ লার্নিং (deep learning); উল্লেখযোগ্যভাবে, ভিশন ট্রান্সফরমার (ভিআইটি) (Vision Transformer (ViT)) বেস মডেল যা বিভিন্ন ডোমেনে ঐতিহ্যগত ডিপ লার্নিং মডেলগুলিকে ছাড়িয়ে গেছে এবং উল্লেখযোগ্য প্রভাব ফেলতে শুরু করেছে। ফেস মাস্ক সনাক্তকরণে ভিআইটি মডেল এর অনুসন্ধান এখনও অন্বেষণ করা হয়নি। এই পেপারে, সাম্প্রতিক একটি ডিপ লার্নিং-ভিত্তিক ইমেজ ক্লাসিফিকেশন (image classification) মডেল প্রয়োগ করার প্রস্তাব করা হয়েছে যার নাম ভিআইটি মডেল যা স্বয়ংক্রিয়ভাবে অনুপযুক্ত ফেস মাস্ক পরা, অর্থাৎ, মুখের মাস্কগুলি সঠিকভাবে পরা হচ্ছে কিনা তা সনাক্ত করতে পারে। ভিআইটি বেস মডেল ভুল ফেস মাস্ক সনাক্তকরণের উপর একটি উল্লেখযোগ্য প্রভাব দেখায়। ভিআইটি মডেলকে প্রশিক্ষণ, যাচাই এবং পরীক্ষা করার জন্য যাতে ফেস মাস্কের ব্যবহার সঠিকভাবে শ্রেণীবদ্ধ করতে পারে; একটি বৃহৎ কাস্টম ডেটাসেট (dataset) নিয়ে পরীক্ষা করা হয়েছে যার মধ্যে ০৩টি ক্লাস লেবেল সহ ২,০৩,৭৮০টি ডিজিটাল ইমেজ রয়েছে যেমন 'মাস্ক সহ' (যখন লোকেরা সঠিকভাবে ফেস মাস্ক পরে থাকে), 'মাস্ক ছাড়াই' (যখন লোকেরা ফেস মাস্ক পরে না), এবং 'ভুল মাস্ক' (যখন লোকেরা সঠিকভাবে ফেস মাস্ক পরে না)। ফলাফলগুলি দেখায় যে প্রাক-প্রশিক্ষিত ভিআইটি মডেলের সাথে অর্জিত নির্ভুলতা অত্যন্ত উল্লেখযোগ্য। উপরন্তু, ০৩ শ্রেণীর লেবেল সমন্বিত একই ডেটাসেট সহ কনভোলিউশনাল নিউরাল নেটওয়ার্ক (সিএনএন) (Convolutional Neural Network (CNN)) মডেলে একই পরীক্ষা করা হয়েছে। তারপরে, সিএনএন মডেলের ফলাফলকে ভিআইটি মডেল ফলাফলের সাথে তুলনা করা হয়েছে। এই অনুসন্ধানগুলি থেকে দেখা যায় যে ভিআইটি মডেলটি দ্রুত প্রশিক্ষণের সময় এবং উচ্চতর প্রশিক্ষণের নির্ভুলতা প্রদর্শন করেছে, এটি ভুল ফেস মাস্ক সনাক্তকরণের জন্য নিজেকে আরও সময়-দক্ষ বিকল্প হিসাবে তৈরি করেছে। প্রশিক্ষণের সময় এই সুবিধাটি রিয়েল-টাইম (real time) অ্যাপ্লিকেশন এবং পরিস্থিতিগুলির জন্য অত্যন্ত গুরুত্বপূর্ণ হতে পারে যাতে দ্রুত

প্রতিক্রিয়া এবং সিদ্ধান্ত নেওয়ার প্রয়োজন হয়। এই ফলাফলগুলি ইমেজ প্রসেসিং এর ক্ষেত্রে অগ্রসর হতে অবদান রাখে এবং ভবিষ্যতের গবেষণা এবং উন্নত ইমেজ প্রসেসিং সিস্টেমের উন্নয়নের জন্য মূল্যবান অন্তর্দৃষ্টি প্রদান করে। এছাড়া বর্ধিত পরীক্ষা-নিরীক্ষায় পাঁচটি স্বতন্ত্র মডেল - এক্সসেপশন (XCEPTION), মোবাইলনেটভি২ (MOBILENETV2), ভিজিজি১৬ (VGG16), ইন্সেপশন (INCEPTION) এবং রেসনেট-৫০ (ResNet-50) মডেল - একই ০৩ শ্রেণীর লেবেল সহ ২০৭৯টি ডিজিটাল ইমেজ সমন্বিত একটি ছোট ডেটাসেট ব্যবহার করা হয়েছে। ফলাফলগুলি দেখিয়েছে যে ভিআইটি মডেলটি অন্য সমস্ত মডেলকে ছাড়িয়ে গেছে। উপসংহারে, এই গবেষণা প্রতিষ্ঠিত করে যে ভিআইটি মডেল অন্যান্য মূল্যায়নকৃত মডেলগুলির মধ্যে সর্বোচ্চ নির্ভুলতা অর্জন করেছে।

ACKNOWLEDGEMENTS

First and foremost, I want to give thanks and praise to Almighty Allah, who provided me with the courage, health and patience to complete this research work.

I would like to express my deep and sincere gratitude to my thesis supervisor *Dr. Nusrat Sharmin; Assistant Professor, Department of Computer Science and Engineering (CSE), MIST*. Her immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. She consistently allowed this research to be my own work, but steered me in the right direction whenever I needed it.

I express my sincere gratitude to Brig Gen Mohammad Sajjad Hossain; Acting Head of the Department, Department of CSE, MIST and Lt Col Muhammad Nazrul Islam, PhD; Associate Professor (Instructor Class-A), Department of CSE, MIST for their worthwhile suggestions, cordial assistance, entire support, guidance and providing all the facilities to complete this thesis work. I would like to convey my thanks to Dr Md Forhad Rabbi; Professor, Department of CSE, Shahjalal University of Science and Technology (SUST) for his kind consent of becoming my external for this thesis.

I am extremely grateful to one and all of Department of CSE, MIST; who directly or indirectly, have lent their hand in this work. Also, I am thankful to all my course mates for their kind support. Finally, I must express my very profound gratitude to my family members for providing me with unfailing motivation and continuous encouragement throughout my years of study and through the process of conducting this thesis. This endeavor would not have been possible without them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ViT	:	Vision Transformer
CNN	:	Convolutional Neural Network
COVID	:	Corona Virus Disease
SARS-CoV-2	:	Severe Acute Respiratory Syndrome Coronavirus 2
AI	:	Artificial Intelligence
WHO	:	World Health Organization
ML	:	Machine Learning
DL	:	Deep Learning
YOLOv3	:	You Only Look Once version 3
VGG16	:	Visual Geometry Group 16
RMFD	:	Real Masked Face Dataset
RNN	:	Recurrent Neural Network
ResNet-50	:	Residual Network 50 Layers
GPU	:	Graphics Processing Unit
ReLU	:	Rectified Linear Unit
FFHQ	:	Flickr Faces HQ Dataset
DeiT	:	Data-efficient Image Transformers
CV	:	Computer Vision

CHAPTER 1
INTRODUCTION

In recent years, the intersection of computer vision and deep learning has ushered in a new era of intelligent surveillance and public safety applications. Among these applications, the detection of face mask-wearing behavior has gained paramount significance due to its role in curbing the spread of infectious diseases. With the emergence of the COVID-19 pandemic, the proper utilization of face masks has become a crucial aspect of public health guidelines. However, monitoring and ensuring compliance with proper face mask-wearing in real-world scenarios pose unique challenges.

1.1 Thesis Background

In December 2019, a new contagious disease named COVID-19 broke out in Wuhan, China. The prevalence of COVID-19 had a negative impact on the financial industries, respiratory assets and numerous sectors which hindered the welfare and development of human communities. The fast spread of the coronavirus (COVID-19) has caused a significant health crisis around the world. To safeguard against the coronavirus, the World Health Organization (WHO) established a number of guidelines, however wearing a mask in crowded areas and public places is the most efficient preventive measure.

When COVID-19 first appeared, it was recommended that people wear face masks to prevent the transmission of the coronavirus (SARS-CoV-2) (Feng et al., 2020). Unfortunately, most countries didn't use face masks when covid-19 first outbreaks. Studies have manifested significant outcomes in lowering the expansion of the virus by utilizing face masks on different flu and other respiratory diseases (Cowling et al., 2010). Individuals who wear face masks properly can reduce the spread of viruses, while improper usage of masks will have the inverse result (World Health Organization, 2020) (Pedersen and Ho, 2020). Figure 1.1 demonstrates three different classes of face masks being worn: the with mask, the without mask and the incorrect mask. However, in this study, the intent is to focus on improper face mask-wearing.

Fig 1.1. Sample Face Image a) With mask b) Without mask c) Incorrect mask

The most effective and protective way to prevent the spread of the coronavirus is to wear a face mask, but persuading people to wear them in public areas is a challenging task for the government and the relevant authorities. Fortunately, face mask wearing in public areas can be ensured by employing artificial intelligence (AI) tools that use machine learning (ML) or deep learning (DL) algorithms simply by identifying face masks. It is a simple way to control society's populace, preserve social distance, and confirm that everyone has worn a face mask properly (Hussain et al., 2022).

1.2 Thesis Motivation

Face-mask detection using AI requires both a detection task as well as a classification task. First, it detected the face mask in digital images and then classified whether people are wearing a mask or not. The first task of this problem has been studied extensively in the computer vision literature because of the popularity of face detection technology (Zafeiriou, Zhang and Zhang, 2015). On the other hand, the second task - predicting whether a face mask is worn or not- has only gained necessity recently, due to the COVID-19 pandemic. Despite the fact that a lot of work has been done in the detection area over the past few years (Morciglio et al., 2021), the majority of these efforts have focused solely on determining if a mask is present in the image or not. Very few works are found in the literature where focus is given to whether the masks are correctly placed or not on the face.

Most recently in paper (Tomas, 2021), authors have proposed an incorrect face mask-wearing detection approach using CNN with Transfer Learning, where CNN based model is being trained with a small amount of dataset (3200 images). They have addressed the

issue of a small dataset and published a new dataset along with the application of improper face mask detection. However, this method has a limitation of validating their method with a small dataset. In this study, the attention-based ViT approach is used to solve the problem of improper face mask detection.

1.3 Problem Statement

The World Health Organization (WHO) and other health organizations have advised using face masks to stop the spread of the extremely dangerous COVID-19 virus. Every government authority is trying to ensure that people wear face masks in public, but in crowded areas, it can be challenging to manually identify people who aren't wearing them. Researchers are trying to create autonomous systems that can recognize and enforce face mask-wearing in public areas.

The issue at hand involves a face mask detection task where the objective is to employ a classification model for categorizing a face image based on whether the person is wearing a mask correctly or not. The input to the classification model is provided in the form of a facial picture.

Most of the ML based research papers emphasize face mask recognition rather than the proper face mask wearing which is equally important (Susanto et al., 2020). Thus, further investigation is required to explore extended ML classifiers for improving correct face mask wearing detection by incorporating different state-of-the-art ML techniques like ViT, CNN model etc. Moreover, a comparative analysis is required to identify which model achieves outstanding results while taking fewer processing time during training and evaluation phases. The objective of this study is to determine whether a person is wearing a mask correctly or not. People often make casual mistakes by wearing a face mask under the nose, below the mouth, etc. That’s why this study considers not only whether a mask is on the face or not, but also it is placed correctly or not.

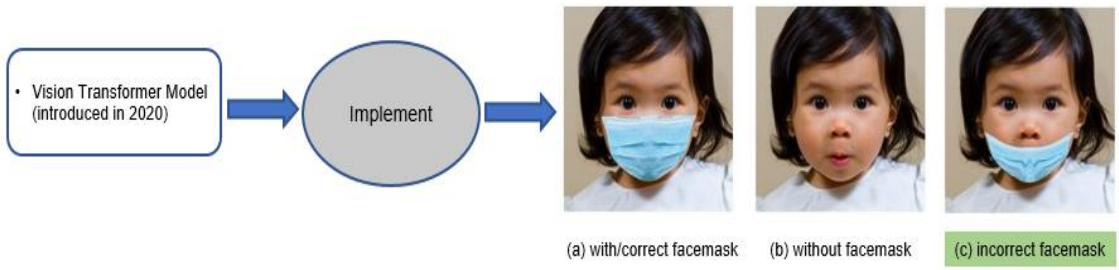

Fig 1.2. Implementation of ViT Model for Incorrect Face Mask Detection

1.4 Objectives of the Thesis

The objectives of this thesis are:

i. To explore ViT based deep learning techniques for detecting improper face mask-wearing.
ii. To develop a model based on the ViT methods to identify improper face mask-wearing.
iii. To evaluate the effectiveness of the proposed method with existing deep learning techniques.

1.5 Contribution of the Thesis

In this thesis, the most recent ViT-based deep learning techniques have been introduced for incorrect face mask-wearing detection with a large amount of dataset. The experiment has been conducted to support our claim. In this study, a pre-trained ViT model is used to classify 03 categories (with mask, without mask and incorrect mask) from the large dataset of 203K images. The ViT model has been trained, validated, and tested and achieved a high accuracy of 98 percent.

The decision to employ ViT and a considerably larger dataset in this study is strategic for several reasons:

- Novelty: ViT represents a shift in how information is processed in computer vision tasks compared to CNNs. Leveraging ViT for face mask detection introduces a novel approach that might yield more accurate and robust results.

- Performance: The anticipation is that ViT, with its ability to capture long-range dependencies in images, could potentially outperform the existing methods like CNNs when it comes to detect nuances in improper face mask wearing.

- Pioneering Work: By exploring the application of ViT in this domain, this study aims to be pioneering. Being the first to implement ViT for detecting improper face mask wearing could pave the way for future research and advancements in this field.

- Data Set: The utilization of a considerably larger dataset is crucial. More data often leads to better model generalization and performance. Training ViT on a larger dataset might enable it to learn more comprehensive representations, potentially improving its ability to recognize different types of improper face mask wearing.

Overall, the decision to employ ViT in conjunction with a larger dataset for detecting improper face mask wearing is driven by the quest for higher performance, the potential for innovation in the field, and the aspiration to fill a gap in current research methodologies.

The main contributions of this thesis are as follows:

- ViT Model: The cutting-edge ViT-based deep learning techniques have been employed, introduced in 2020, to detect improper wearing of face masks.

- Two different types of Dataset: A large dataset of 203K images of 03 class levels named with mask, without mask and incorrect mask has been addressed. A small dataset consisting of 2079 images of the same class levels has also been addressed.

- Comparative Experiment: In this research, experiment has been conducted among CNN, XCEPTION, MOBILENETV2, VGG16, INCEPTION, and ResNet-50 models with the smaller dataset and a comparative result analysis with ViT model has been made to substantiate our assertion that ViT outperforms other models.

1.6 Organization of the remaining Chapters

The rest of the thesis book has been structured in the following way:

i. Chapter 2 describes the necessary theoretical background of the study.

ii. In Chapter 3, the methodology of both ViT model and CNN model based Incorrect Face Mask Detection have been described. It also presents the methodology of other deep learning models (XCEPTION, MOBILENETV2, VGG16, INCEPTION, and ResNet-50) in Incorrect Face Mask Detection.

iii. Chapter 4 explains the results of the experiment of both ViT and CNN model, and analyses the results between these models. It also showcases why the ViT model might offer advantages in the context of detecting incorrect face mask wearing over other relevant models commonly used in computer vision tasks, specifically Xception, MobileNetV2, VGG16, InceptionV3, ResNet-50, and a generic CNN architecture.

iv. Chapter 5 of the thesis presents the discussion and conclusion of the research work with the limitations of the study as well as future plan.

CHAPTER 2
BACKGROUND

This chapter includes some fundamental concepts of this research acquainted with the background. Initially, it delves into the theoretical foundation, followed by an exploration of previous work in detecting improper face masks. Finally, it identifies the gaps in research within this domain.

2.1 Theoretical Background

2.1.1 Face Mask Recognition

The method of recognizing and classifying people according to whether or not they are wearing face masks is known as "face mask recognition." It entails analyzing pictures or video frames with human faces to see if a face mask is visible. Computer vision and machine learning techniques are used for this.

With the widespread adoption of face mask wearing as a preventive measure during the COVID-19 epidemic, the application of face mask recognition gained significance. Face mask recognition systems are developed to contribute to public health efforts by automatically identifying individuals who are complying with mask-wearing guidelines and those who may not be.

The following steps are usually involved in these systems:

- Face Detection: Locating and detecting human faces inside pictures or video frames. This is often done using face detection algorithms.
- Mask Presence Detection: Examining the affected area of the face to see if a mask is present or not. This phase entails identifying face characteristics and determining whether a mask is on or off the mouth and nose.
- Classification: Categorizing persons into groups based on the presence or absence of a face mask.
- Alerts or Notifications: Based on the findings of the recognition, sending alerts or notifications to relevant parties, law enforcement, or security personnel.

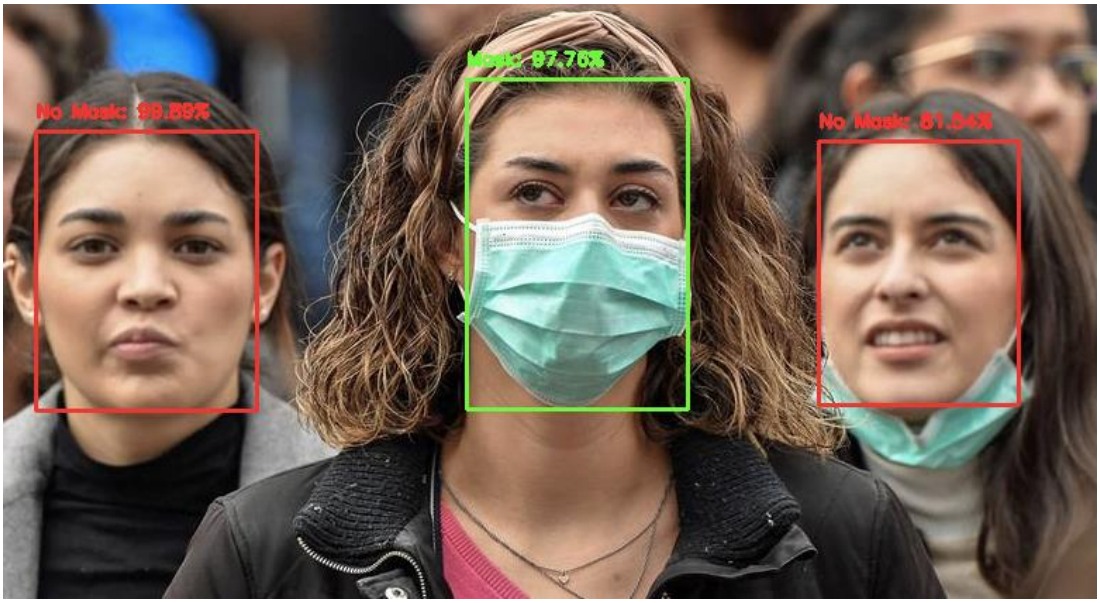

Fig 2.1. Face Mask Recognition

Applications for face mask identification technology can be found in a variety of areas, including public places, workplaces, transit hubs, and healthcare institutions, where it is crucial to enforce mask-wearing regulations for everyone's health and safety. It can support efforts to stop the spread of infectious diseases and automate the monitoring procedure.

2.1.2 Deep Learning

A branch of machine learning called "deep learning" is concerned with creating and conditioning artificial neural networks to carry out tasks without the need for explicit programming. It draws inspiration from the anatomy and physiology of the human brain, especially from the neural networks that house the brain's networked neurons.

Key characteristics of deep learning include:

- Neural Networks: Artificial neural networks, which are made up of layers of connected nodes (artificial neurons), are commonly used to build deep learning models. The word "deep" learning comes from the possibility of numerous layers in these networks.
- Learning from Data: Deep learning algorithms acquire representations and patterns straight from the data. To identify patterns, characteristics, and correlations in the input data, they are trained on big datasets.

- Representation Learning: Hierarchical data representations are automatically learned by deep learning models. The neural network's layers each acquire the ability to represent various abstraction levels, allowing the model to comprehend complicated features.
- End-to-End Learning: Deep learning models don't require human feature engineering because they can learn from raw data directly and generate output. Through a single, integrated process, they are able to execute end-to-end learning, extract pertinent information, and make predictions or choices.
- Training with Backpropagation: Backpropagation is an iterative optimization technique used in the training phase. In order to minimize the discrepancy between expected and actual outputs, the model modifies its internal parameters during training, progressively enhancing its performance.
- Versatility: Deep learning has shown promise in a variety of applications, such as computer vision, natural language processing, picture and speech recognition, and playing strategy games.

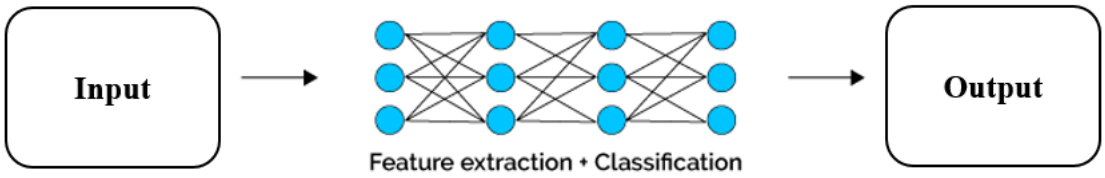

Fig 2.2. Deep Learning Technique

CNNs, recurrent neural networks (RNNs), and transformer designs are popular neural network types used in deep learning. CNNs are utilized for image-related tasks, while RNNs are used for sequence data. Recent years have seen amazing advances in deep learning, allowing machines to accomplish difficult jobs with human-like abilities and advancing numerous fields.

2.2 Related Work

With the spreading of COVID-19 to worldwide, realistic solutions to optimize the incorrect use of face mask has become essential. Deep learning, a branch of machine learning and artificial intelligence, is one of the best techniques for solving image categorization

problems to varied situations (AI). In this section, the most relevant works on deep learning techniques have been discussed in the context of COVID-19. Note that most of the paper focus on face mask detection, whereas this research present the literature based on that.

Through the years, multiple research studies have been presented on face detection. One of the most influential studies is the paper presented by Paul Viola and Michael Jones (Viola and Jones, 2001). Their model was based on rapid object detection using a boosted cascade of simple features. The Viola and Jones proposal is known worldwide for being the first CV model that employed a rudimentary machine learning (ML) technique known as boosted decision trees. The central idea was to create an algorithm capable of recognizing faces and drawing a box around them. With the advent of CNNs, the face detection error decreased significantly (Li et al., 2015).

2.2.1 Deep Learning for Face Mask Recognition

Subsequently, Deep Learning (DL) research accelerated its pace, and novel deep neural network architectures were proposed to tackle the face detection problem. The main advantage of using DL architectures is the increased accuracy of the models. In (Sun, Wu and Hoi, 2018), the authors used a faster region-based CNN framework to perform face detection. This study obtained a state-of-the-art face detection performance on the FDDB benchmark. Another important model is RefineFace, which is an improved face detection model based on ResNet, with a 6-level feature pyramidal structure used as the base of the network (Zhang et al., 2020). RefineFace is also based on the face detector RetinaNet, with five new modules: STR, STC, SML, FSM, and RFE. The results obtained by varying the modules are presented in the paper, with accuracies near and above 90%.

The necessity for a model to detect faces in a complex background and the large amount of time consumed are reasons why the authors in (Gao and Yang, 2022) presented face detection using improved TinyYOLOv3 and an attention mechanism. This is composed of an improved Tiny YOLOv3 capable of extracting significant semantic information by changing the traditional convolution to deep separable convolution, i.e., dividing a single convolution into two or more parts to obtain a model of smaller size with a high detection speed. In addition, the attention mechanism was added to the feature extractor layers to improve the detection position.

Lastly, a face detection algorithm based on a double-channel CNN with occlusion perceptron is a method that uses a VGG16 as a backbone, with the addition of a specialized unit to judge occluded areas, making it an occlusion perceptron neural network. The double channel refers to the capacity of the residual network to extract features of the whole face, while the perception neural network extracts the features of the occluded parts. Both results are combined to produce the final prediction. This method improves detection speed and accuracy (Li, 2022).

Multiple DL models were released from the beginning of the COVID-19 pandemic with different approaches to solving the face mask detection and classification challenges. For example, in (Das, Ansari and Basak, 2020), the authors propose a hybrid model using DL and classical ML. The first part uses a ResNet-50 architecture as a feature extractor, while decision trees and support vector machines are used to classify the positions of the masks. Similarly, in (Sethi, Kathuria and Kaushik, 2021), the authors present a model capable of identifying face-mask-wearing conditions by combining super-resolution images and classification networks such as SRCNet. In addition, the use of ResNet-50 as a feature extractor and YOLOv2 as a detector has been recently proposed (Nagrath et al., 2021).

In (Zhang et al., 2020), the authors proposed the so-called FMD-Yolo framework to detect whether people wear masks in public spaces correctly. This framework employs Im-Res2Net-101 and the path aggregation network En-PAN for the steps of feature extractor and feature fusion, respectively. The authors test their approach against several state-of-the art detection algorithms using two publicly available datasets. Their results on both datasets outperformed the approaches to which they were compared.

Context-Attention R-CNN is another detection method using a new framework for recognition of masked faces (Zhang et al., 2021). The method is based on multiple context feature extractor components, decoupling branch components, and attention components. In addition, the researchers created a dataset of 8635 faces under different experimental conditions. The mean average precision (mAP) for the model was 84.1% over the dataset previously mentioned, which was 6.8% mAP higher than Faster R-CNN. Finally, Face Mask Recognition Network (FMRN) is a detection model that uses an algorithm to classify images via posture recognition. Then, the network processes the images according to the classes (Lin et al., 2021).

Artificial Neural Networks (ANN) can help classify the usage of the mask through image recognition by learning the extracted feature of various images by processing the images through the layer repeatedly. ANN will deliver the best performance incrementally with the size and uniqueness of the data set (Shahinfar, Meek and Falzon, 2020). Through some research, it is found that Masked Face-Net data set (Cabani et al., 2021) is the best choice for this research. It consists of 1,37,016 digital images using masks with 4 class labels: Mask, Mask Chin, Mask Mouth Chin, and Mask Nose Mouth. Compared to the data set used by Masked Face Detection Dataset (MFDD) which consists of only 24,771 masked face images, and Real-world Masked Face Recognition (RMFRD) which consist of only 95,000 pictures, Masked Face-net has more range of variation of age, race, and ethnicity because it is gathered from Flickr-Faces-HQ dataset (Rahadika, Yudistira and Sari, 2021).

Using a person's face in images, videos, or in real time, facial recognition methods can recognize or verify a person's identity. Traditional face recognition methods named holistic methods, feature-based methods and hybrid methods are used in most of the cases to identify faces. In recent years, deep learning techniques using CNNs have supplanted conventional methods for face recognition (Trigueros, Meng and Hartnett, 2018). A recent study shows that three pre-trained deep CNNs models namely VGG-16, AlexNet and ResNet-50 are widely used to recognize masked faces (Hariri, 2020). These models are used to retrieve fundamental features, usually from the forehead and eye regions, from the obtained areas. Deep learning methods' main benefit is their ability to be trained on very large datasets to discover the most effective features for data representation. Because there are so many human faces on the internet, it has been possible to compile sizable datasets of human faces with variances found in the real world, which have then been used to train CNN-based face recognition algorithms.

2.2.2 Deep Learning for Improper Face Mask wearing Detection

A very few amounts of research work have been done over the last few years regarding the problem of improper face-mask detection. A research paper entitled "Incorrect Facemask-Wearing Detection Using CNN with Transfer Learning" uses a public dataset of 3200 images with 13 categories (Tomas et al., 2021). To increase the performance of this small dataset, multiple methods including data augmentation and transfer learning were compared by the authors. They compared the MobileNet architecture of 3.5 million

parameters and VGG16 architecture of 134.4 million parameters. This study concluded that VGG16 produces good accuracy using both transfer learning and data augmentation.

In (Loey et al., 2021), it utilized RMFD dataset which comprises 5000 with mask and 90000 without face mask images of real-world. Another dataset was presented named Labeled Faces in the Wild (LFW) which comprises 13000 simulations of with mask images. After that, the authors proposed the utilization of a hybrid and complex deep transfer learning model, with the ResNet50 as elements extractor. At last, the utilization of SVM (Support Vector Machine) and decision tree to build a learning platform which performs well with high training and testing accuracy on both RMFD and LFW datasets.

The Medical Masks Dataset (MMD), which consists of 853 images categorized into three groups: mask, without mask, and mask worn incorrectly, has been annotated in a study titled "Fighting against COVID-19: A Novel Deep Learning Model Based on YOLO-v2 with ResNet-50 for Medical Face Mask Detection" (Loey et al., 2021). Using YOLOv2 and ResNet feature extraction, this produces an accuracy result of 81%. Their results suggest that training probabilistic models on labeled images should outperform training on unannotated images. However, it is challenging to get and validate the necessary number of annotated images for deep learning research.

Table 2.1: SUMMARY OF THE LITERATURE REVIEW

Reference	**Focuses**	**Findings**	**Limitations**
Tomas et al., 2021	Authors have proposed an incorrect face mask-wearing detection approach using CNN with Transfer Learning, where CNN based model is being trained with a small amount of dataset (3200 images).	They have shown that the accuracy achieved with transfer learning slightly improves the accuracy achieved with CNN.	However, In this study they train and test their model with a small dataset.
Loey et al., 2021	Authors have proposed the utilization of a hybrid and complex	Three face masked datasets have been selected for investigation	It utilizes RMFD dataset which comprises 5000 with

	deep transfer learning model, with the ResNet50 as elements extractor. They have utilized SVM (Support Vector Machine) and decision tree to build a learning platform. They only focus on with facemask and without facemask detection.	which are the Real-World Masked Face Dataset (RMFD), the Simulated Masked Face Dataset (SMFD), and the Labeled Faces in the Wild (LFW). The SVM classifier achieved higher testing accuracy in RMFD dataset with compare to others.	mask and 90000 without face mask images of real-world. Another dataset is presented named LFW which comprises 13000 simulations of with mask images. Moreover, in this study they don't focus on incorrect facemask wearing detection.
Loey et al., 2021	Authors have used YOLOv2 and ResNet feature extraction and the Medical Masks Dataset (MMD), which consists of 853 images categorized into three groups: mask, without mask and mask worn incorrectly that has been annotated in this study.	They produce an accuracy result of 81%. Their results suggest that training probabilistic models on labeled images should outperform training on unannotated images.	However, it is challenging to get and validate the necessary number of annotated images for deep learning research. Moreover, in this study they train and test their model with a small dataset of 853 images.

2.3 Research Gap

The majority of ML based research papers emphasize face mask recognition rather than the proper face mask wearing which is equally important. Consequently, there is a need for further exploration into extended machine learning classifiers to enhance the detection of correct mask wearing by integrating various state-of-the-art machine learning techniques. As ViT model is introduced in 2020, there exists a gap in harnessing the latest strides in computer vision, notably the ViT model, for the purpose of detecting improper face mask wearing. While traditional approaches have been prevalent, the untapped potential of ViT in addressing this issue signifies an opportunity to leverage its advanced capabilities for more accurate and efficient detection of incorrect face mask usage. Integrating ViT model into the domain of face mask detection application could mark a significant step forward in improving the reliability and performance of such systems.

CHAPTER 3
METHODOLOGY

This chapter details the step-by-step process and methodology that employed to address the problem. The workflow comprises three distinct phases which are outlined in Figure 3.1.

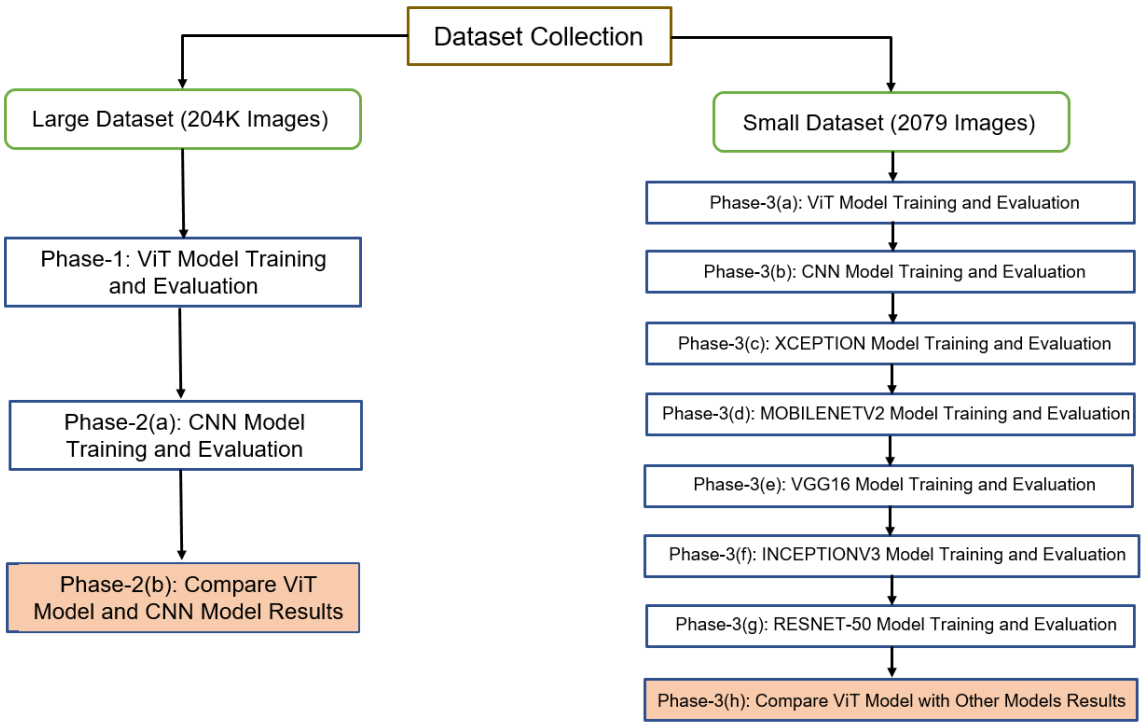

Fig. 3.1. Step-by-step Workflow Diagram

3.1 Dataset Collection and Overview

The referenced literature in the preceding chapter has established the necessity for a high resolution and large dataset to enhance accuracy, ensure image uniqueness, and excel in real-world applications. In this study a large dataset consists of around 204K images has been collected to compare results between ViT and CNN, and a small dataset consists of 2079 images has also been collected to compare results between ViT and other models (CNN, XCEPTION, MOBILENETV2, VGG16, INCEPTION, and ResNet-50).

3.1.1 Large Dataset Overview

The large dataset that used in this study is MaskedFace-Net (Cabani et al., 2021) dataset which consists of 02 categories of images named ‘with mask’ and ‘incorrect mask’. For ‘without mask’ category Flickr-Faces-HQ Dataset (FFHQ) has been used. Details of the large dataset has been reported in Table 3.1.

Table 3.1: LARGE DATASET OVERVIEW

Category Name	Image Source	Image Count
with_mask	MaskedFace-Net	67,048
without_mask	Flickr-Faces-HQ Dataset (FFHQ)	70,000
incorrect_mask	MaskedFace-Net	66,734
	Total	203,782

From the table 3.1, it is shown that 67,048 images of ‘with mask’ category, 70,000 images of ‘without mask’ category and 66,734 images of ‘incorrect mask’ category with dimensions of 224 by 224 pixels of each image.

Fig. 3.2. MaskedFace-Net Dataset Sample (with_mask and incorrect_mask category)

Fig. 3.3. Flickr-Faces-HQ Dataset sample (without_mask category)

Figure 3.2 and Figure 3.3 show the sample images of the datasets from two different sources named MaskedFace-Net dataset and Flickr-Faces-HQ Dataset respectively.

3.1.2 Small Dataset Overview

The small dataset that used in this study is consists of 2079 images with 03 categories named ‘with mask’, ‘without mask’ and ‘incorrect mask’. Details of the small dataset has been reported in Table 3.2.

Table 3.2: SMALL DATASET OVERVIEW

Category Name	Image Source	Image Count
with_mask	Kaggle	690
without_mask	Kaggle	686
incorrect_mask	Kaggle	703
	Total	2079

From the table 3.2, it is shown that 690 images of ‘with mask’ category, 686 images of ‘without mask’ category and 703 images of ‘incorrect mask’ category.

a) With Mask b) Without Mask

Fig. 3.4. Small Dataset Sample (a) with_mask and b) without_mask Category

c) Incorrect Mask

Fig. 3.5. Small Dataset Sample (c) incorrect_mask Category

Figure 3.4 and Figure 3.5 show the sample images of the small datasets from different sources.

3.2 Vision Transformer (ViT) Model

Engaging with ViT model encompasses a comprehensive workflow, involving a blend of theoretical understanding, data preparation, model training, validation, and iterative improvements to achieve robust performance. In this section, all these workflows are discussed.

3.2.1 Vision Transformer (ViT)

In 2017, the Transformer was introduced through a research paper titled "Attention Is All You Need" (Vaswani et al., 2017). Its primary aim was to address challenges present in Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) models. Particularly within the realm of Natural Language Processing (NLP), RNN and LSTM models encounter difficulties related to loss gradients and prolonged training periods. Traditionally, CNNs have been the foundational architecture for tasks such as image classification. However, transformers have demonstrated their applicability in image classification research, exhibiting a computational efficiency five times faster than the convolutional architecture, all while maintaining competitive accuracy (Dosovitskiy et al., 2020).

A model named ViT is employed for image classification. This involves applying a transformer-like architecture to segments or slices of the image. The steps are as follows:

- Generate a lower-dimensional linear embedding by flattening patches obtained after dividing and flattening an image into distinct segments.
- Transmit the resulting sequence as input to a conventional transformer encoder, incorporating positional embedding.
- Pre-train the model using image labels from an extensive dataset (fully supervised), followed by fine-tuning the model on a subsequent dataset designed for image classification.

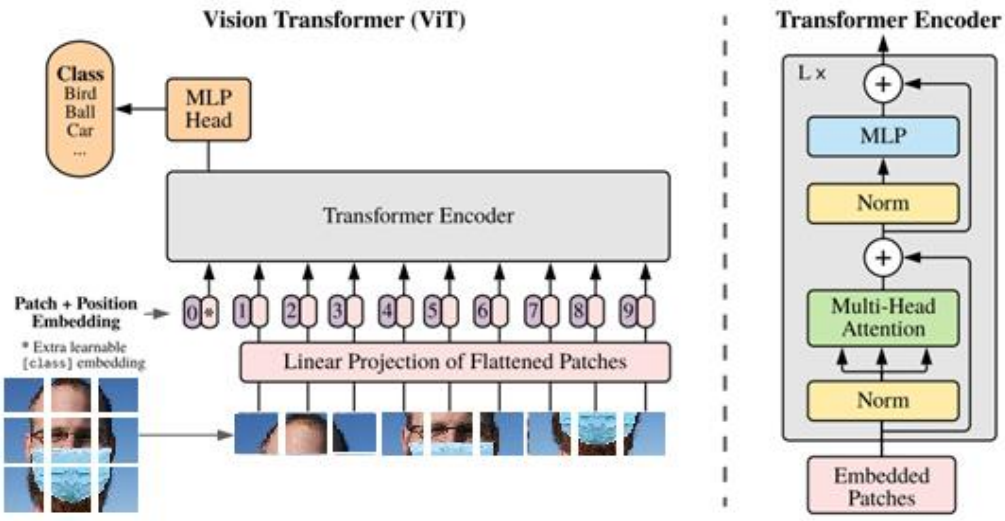

Fig. 3.6. ViT Model Overview

Figure 3.6 shows an overview of the ViT model which depicts an outline of the ViT model where input images are sliced into several pieces depended on the patches' number that have been mentioned after going through the procedure of slicing a 2-dimensional digital image. Here, the two-dimensional digital image must be converted into a one-dimensional vector. Then it will be changed P the number of patches.

After that, the submerged outcomes will pass through the transformer's encoder. After that, a neuron above the small batch of training examples is subjected to a layer normalization method that makes use of the summed input distribution. Following that, Variance and Mean are used to normalize the aggregated input, which offers a significant speed advantage over batch normalization.

The multi-head self-attention is necessary to detain critical part of the image. The most often used feature extractor to obtain the most important portion of the photos is ResNet. However, the transformers encoder will function adequately even without another feature extraction. A straightforward multilayer perceptron serves as the encoder's final layer. Its results are dependent on the classifications provided in the data set, and GeLU is used to activate the multilayer perceptron.

3.2.2 Vision Transformer (ViT) based Incorrect Face Mask Detection

In this section, the procedures that are required for incorrect face mask detection using VIT model has been presented. For this implementation, the following steps are taken that depicted in Figure 3.7. It shows the workflow diagram of classification process using ViT model phase by phase.

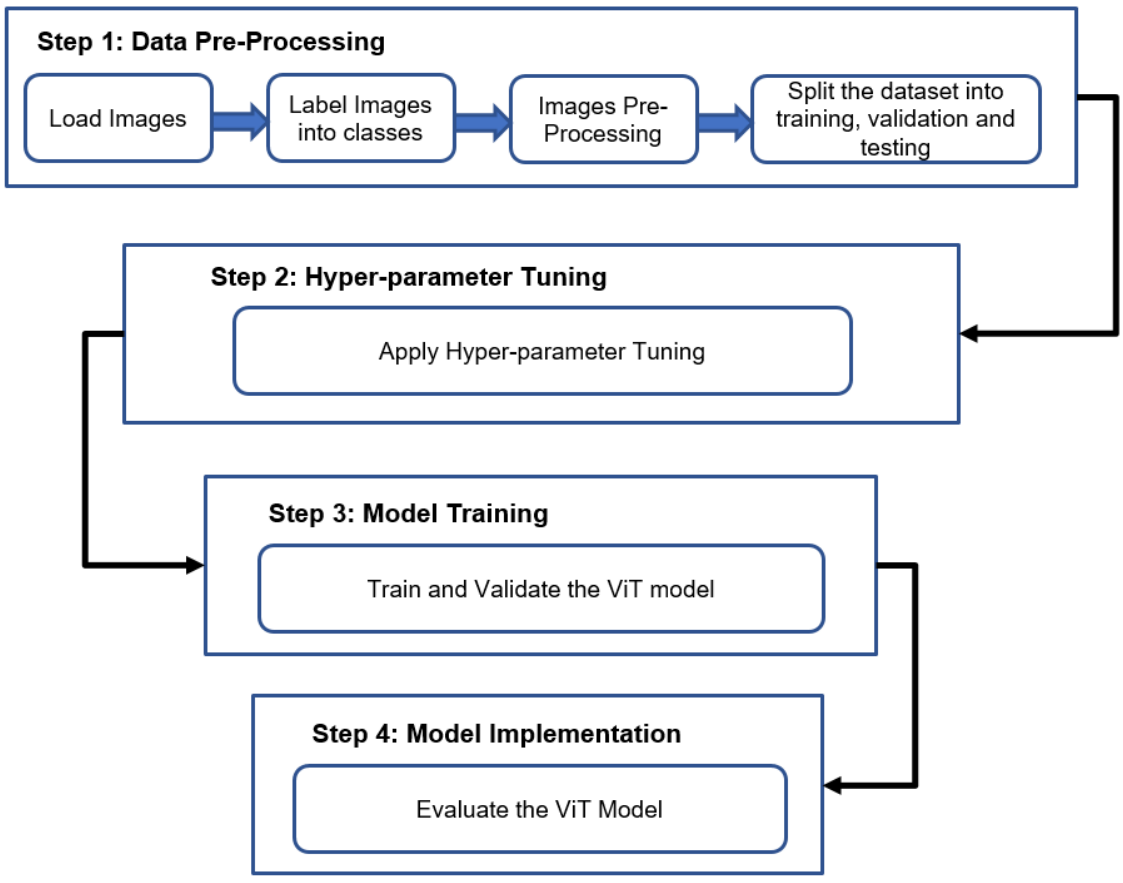

Fig. 3.7. Workflow diagram for ViT based Incorrect Face Mask Detection

In the following sections all the steps will be discussed elaborately.

1. **Step 1- Data Pre-processing:** In Data Pre-processing phase, all the images from multiple sources are gathered and labeled into three different classes. Then the raw images are preprocessed (if needed) and split into training, validation and testing folder. The whole dataset has been split into the train, validation and test data with respect to the size of 70%, 20% and 10% of total images respectively as shown in Table 3.3. The 'ratio' function of python 'splitfolders' package is used to do this splitting task.

TABLE 3.3: DATASET AFTER SPLITTING FOR ViT EXPERIMENT

Folder Name	Category Name	Image Count
train	with_mask	46,933
	without_mask	49,000
	Incorrect_mask	46,715
validation	with_mask	13,410
	without_mask	14,000
	Incorrect_mask	13,346
test	with_mask	6,705
	without_mask	7,000
	Incorrect_mask	6,673

2. **Step 2- Hyper-Parameter Tuning:** For the ViT model, a number of configurations must be defined. To avoid bias in the test outcomes across different configurations, consistent Hyper-parameters are maintained throughout training. The models are fine-tuned using specific Hyper-parameters such as the rectified linear unit (ReLU) function, label-smoothing cross-entropy function, a batch size of 32, learning rate set at 0.01, 50 epochs, and employing the Adam optimizer method.

   Data does not always come in its final processed form that is required for training ViT model and that's why data augmentation/transformation is needed which changes photos in a way that gives them a new shape in the form of digital images. Transforms module of torch vision is used to perform some manipulation of the data and make it suitable for training model. These transformations can be chained together using Compose. 'Random Horizontal Flip' function, 'Random Vertical

Flip' function, 'Normalize' function, 'Random Erasing' function etc. are also used for transforming data.

3. **Step 3- Model Training:** Following the three different layers of (ViT) are discussed:
    - **ViT Model:** In this study, Data-efficient Image Transformers (DeiT) model is used which is the core of the ViT model. The DeiT model is optimized and pre-trained at 224 x 224 resolution.
    - **Dropout:** Here dropout value is 0.3 which is utilized for regularization to avert overfitting.
    - **Linear:** The image is finally classified by this last layer. It has an output layer that is equal to the number of distinct classes and an input size that is as large as the number of hidden nodes on the ViT Model.

    To train the ViT model, a custom function has been written up which takes the ViT model as input parameter and train that model. It may take a significant amount of time. So, to do it fast, it is recommended to enable the GPU during the training. In this study, it takes around 105 minutes to train and validate the model for the dataset of 204K images.

4. **Step 4 - Model Implementation:** In this model implementation phase, the ViT model is executed for incorrect face mask detection. Here the following steps are required to do this classification task.

    Initializing Setup: This study makes utilization of the Kaggle online community platform for data scientists and deep learning aficionados, which enables users to collaborate with one another, find and share datasets, and use GPU integrated notebooks for experiment.

    Importing Python Packages: After completing the necessary setup, all the required python packages such as 'torch', 'torch vision', 'numpy', 'pandas', 'matplotlib' etc. are imported in Kaggle notebook. Then 'timm', 'split-folders' are installed and 'openpyxl' python libraries are imported in Kaggle notebook.

Importing Dataset: Dataset has been imported and split the whole dataset into the training, validation, and test data folder with respect to the size of 70%, 20% and 10% respectively. To spilt the data, ‘ratio’ function of python ‘splitfolders’ package has been used. The 'splitfolders' package is commonly used to automate the process of splitting datasets into training, validation, and test sets based on specified ratios or proportions. In this case, it was employed to allocate 70% of the data to the training set, 20% to the validation set, and 10% to the test set, ensuring a balanced distribution for model training, validation, and evaluation.

Identifying Class Labels: To get the class labels from dataset we use ‘datasets’ module of torch vision and ‘ImageFolder’ function which returns all the class labels of the dataset. Three class labels named ‘incorrect mask’, ’with mask’ and ’without mask’ has been used in this experiments.

Implementing ViT Model: To implement the ViT model, a custom function has been written up which takes the ViT model as input parameter and test that model. It may take a significant amount of time. So, to do it fast, it is recommended to enable the GPU during the training. In this study, it takes around 10 minutes to test the model for the dataset of 20K images.

Accuracy: We evaluate the accuracy of incorrect mask, with mask and without mask category individually and the overall accuracy of the ViT model. Besides, the value of test loss in the process has also been tested.

3.3 Convolutional Neural Network (CNN) Model

Engaging with CNN Model involves a range of steps and considerations which all are discussed in this section.

3.3.1 Convolutional Neural Network (CNN)

CNN is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words Recurrent Neural Networks more precisely an LSTM are used, similarly for image classification Convolution Neural networks are used (Selvaraju et al., 2016).

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which input is given to the model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).

2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon the model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear (Selvaraju et al., 2016).

3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, then the error is calculated using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, back propagation is done into the model by

calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss (Selvaraju et al., 2016).

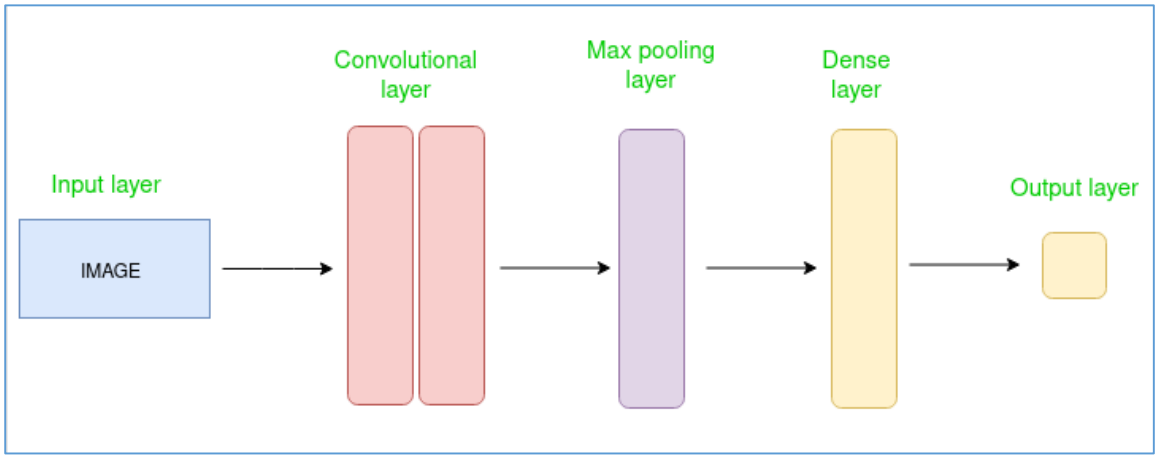

Fig. 3.8. Simple CNN Model Architecture

3.3.2 Convolutional Neural Network based Incorrect Face Mask Detection

In this section, we can see the procedures that are required for incorrect face mask detection using CNN model. For this implementation, the following steps are taken that is depicted in Figure 3.9. It shows the workflow diagram of classification process using CNN model phase by phase.

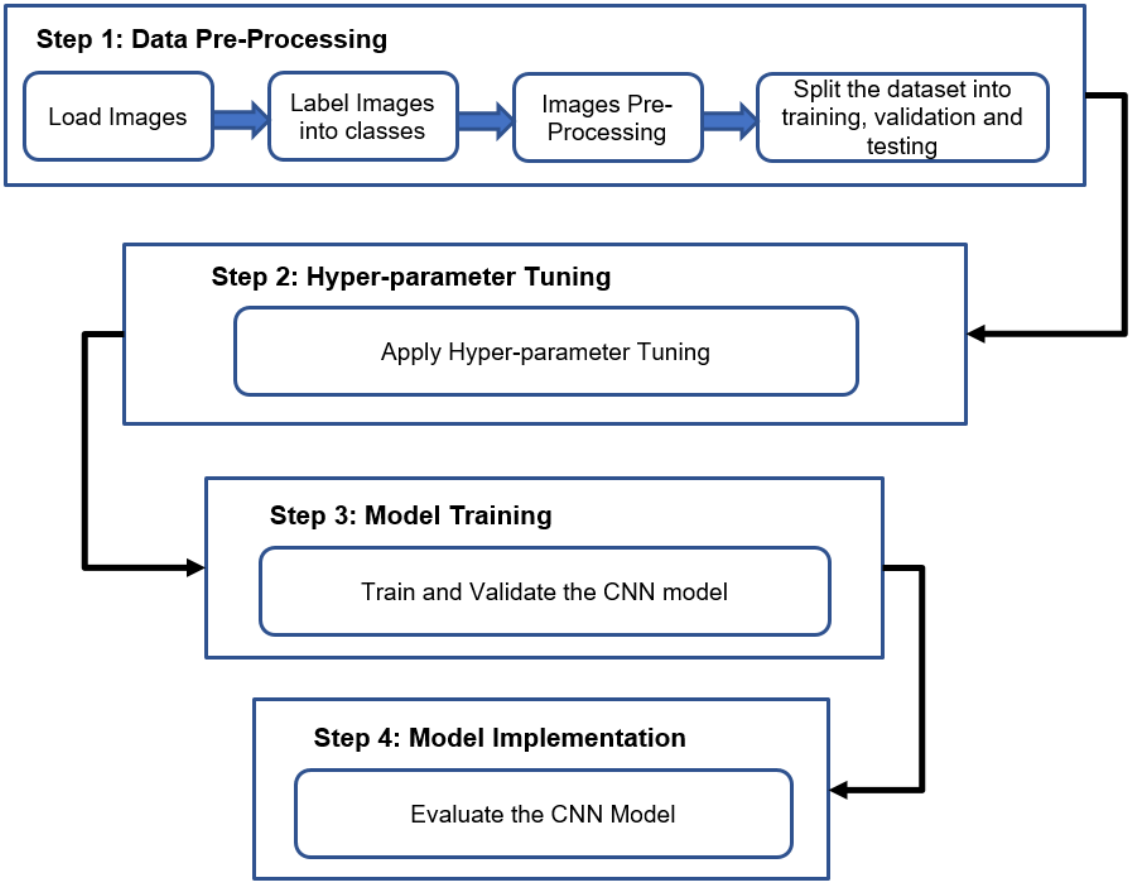

Fig. 3.9. Workflow diagram for CNN based Incorrect Face Mask Detection

In the following sections all the steps are discussed elaborately.

- **Step 1- Data Pre-processing:** In Data Pre-processing phase, all the images from multiple sources are gathered and labeled into three different classes. Then the raw images are preprocessed (if needed) and split into training, validation and testing folder. The whole dataset has been split into the train, validation and test data with respect to the size of 70%, 20% and 10% of total images respectively as shown in Table 3.4. The 'ratio' function of python 'train_test_split' package is used to do this splitting task.

TABLE 3.4: DATASET AFTER SPLITTING FOR CNN EXPERIMENT

Folder Name	Category Name	Image Count
train	with_mask	46,933
	without_mask	49,000

	Incorrect_mask	46,715
validation	with_mask	13,410
	without_mask	14,000
	Incorrect_mask	13,346
test	with_mask	6,705
	without_mask	7,000
	Incorrect_mask	6,673

- **Step 2- Hyper-Parameter Tuning:** Several configurations for the CNN model need to be specified. Constant Hyper-parameters are established during training to avoid bias in the test results of each configuration. The Hyper-parameters used for fine-tuning include the rectified linear unit (ReLU) function, categorical cross entropy function, batch size of 32, learning rate of 0.01, epochs of 50, and Adam optimizer method.

  Data augmentation, also known as data transformation, is necessary to give photos a new structure in the form of digital images because data does not always arrive in the final processed form needed to train a CNN model. The Python OpenCV package is utilized to manipulate the data and prepare it for model training. These transformations can be combined with 'Random Horizontal Flip', 'Random Vertical Flip', 'Normalize', 'Random Erasing' etc. for transforming data.

- **Step 3- Model Training:** Following the different layers of CNN model are discussed:

  - **CNN Model:** In this study the 'Sequential' class is imported from the 'models' module within Keras. The 'Sequential' class allows to create a linear stack of neural network layers, enabling to build a model layer by layer.

- **Layers:** The ‘layers’ module is imported within Keras to use Conv2D, MaxPooling2D, Flatten and Dense layers. Each layer has a distinct functionality within a CNN model.

- **Conv2D:** This represents a 2-dimensional convolutional layer used for filtering spatial data.

- **MaxPooling2D:** A pooling layer that performs a down sampling operation to reduce the spatial dimensions of the input.

- **Flatten:** This layer is used to convert the multidimensional output of the previous layer into a one-dimensional array, preparing it for the fully connected layers.

- **Dense:** Represents the fully connected layers, where each neuron is connected to every neuron in the preceding and succeeding layers.

Together, these imported components provide the essential building blocks required to create a basic CNN architecture in Keras. To train the CNN model, a custom function has been written up which takes the CNN model as input parameter and train that model. It may take a significant amount of time. So, to do it fast, it is recommended to enable the GPU during the training. In this study, it takes around 218 minutes to train and validate the model for the dataset of 204K images.

- **Step 4- Model Implementation:** In this model implementation phase, the CNN model is executed for incorrect face mask detection. Here the following steps are required to do this classification task.

  Initializing Setup: This research utilizes the Kaggle online platform tailored for data scientists and deep learning enthusiasts. It allows collaborative work, dataset discovery, sharing, and the use of GPU-integrated notebooks for experimentation.

Importing Python Packages: After completing the necessary setup, all the required python libraries such as ‘numpy’, ‘pandas’, ‘tensorflow’, ‘matplotlib’, ‘sklearn’ etc. are imported in Kaggle notebook.

Importing Dataset: The dataset has been imported and split into the training, validation, and test data folder with respect to the size of 70%, 20% and 10% respectively. To spilt the dataset, ‘train_test_split’ function of python ‘sklearn.model_selection’ package is used.

Identifying Class Labels: To get the class labels from dataset we use ‘os’ module of python and ‘listdir’ function which returns all the class labels of the dataset. Three class labels named ‘incorrect mask’, ’with mask’ and ’without mask’ has been used in this experiments.

Implementing CNN Model: To implement the CNN model, ‘evaluate’ function of ‘models’ module has been called which takes the CNN model as input parameter and tests that model. It may take a significant amount of time. So, to do it fast, it is recommended to enable the GPU during the training phase. In this study, it takes around 19 minutes to test the model for the dataset of 20K images.

Accuracy: We evaluate the accuracy of incorrect mask, with mask and without mask category individually and the overall accuracy of the CNN model. Besides, the value of test loss in the process has also been tested.

3.4 Other Deep Learning Models

In extensive experimentation, five different CNN architectures, namely XCEPTION, MOBILENETV2, VGG16, INCEPTION, and ResNet-50 have been incorporated. These architectures are assessed using a relatively smaller dataset, which comprised 2079 digital images categorized into the same three distinct class labels.

3.4.1 Xception Model

The Xception model is a deep CNN architecture designed for computer vision tasks, particularly image classification. It was proposed by François Chollet, the creator of Keras, in 2017. The name "Xception" is derived from "Extreme Inception," as it is an extension and improvement over the Inception architecture. Xception's core innovation revolves around the concept of depthwise separable convolutions.

Key Features of the Xception Model:

- Depthwise Separable Convolutions: Xception extensively uses depth wise separable convolutions instead of traditional convolutions. This involves splitting the convolution into two separate operations:
  - Depthwise Convolution: Performs convolution independently for each input channel.
  - Pointwise Convolution: Applies a 1x1 convolution to combine information across channels.
- Extreme Depthwise Separability: Xception takes this concept to an extreme level by stacking multiple layers of these depthwise separable convolutions. This architecture significantly reduces the number of parameters compared to traditional CNNs.
- Efficiency: The use of depthwise separable convolutions allows Xception to achieve better computational efficiency without compromising on the network's representational power or performance.
- Hierarchical Feature Extraction: Xception is structured into a series of convolutional blocks, enabling the extraction of hierarchical features from input images. This architecture promotes the flow of information through the network.
- Image Classification: The primary application of the Xception model is image classification, where it excels in recognizing and classifying objects within images into predefined categories or classes.

3.4.2 MobileNetV2 Model

MobileNetV2 is a neural network architecture designed primarily for mobile and edge devices where computational resources and memory are limited. It's an evolution of the original MobileNet, introduced by Google researchers in 2018.

Key Features of MobileNetV2:

- Depthwise Separable Convolutions: Similar to the original MobileNet, MobileNetV2 extensively uses depthwise separable convolutions. These convolutions are split into depthwise and pointwise convolutions, reducing computational cost and the number of parameters.
- Inverted Residuals with Linear Bottlenecks: MobileNetV2 introduces the concept of inverted residuals, which expand the network's non-linearity without significantly increasing the computational load. Linear bottlenecks are used to maintain information flow through the network efficiently.
- Linear Bottlenecks: Intermediate layers are compressed with 1x1 convolutions, reducing the number of channels before expansion and after squeezing. This helps in maintaining a balance between computational efficiency and network depth.
- Width Multiplier and Resolution Multiplier: MobileNetV2 provides flexibility through width and resolution multipliers. The width multiplier controls the size of the input and output channels of the network, while the resolution multiplier adjusts the input resolution, allowing trade-offs between accuracy and computation.
- Improved Performance: MobileNetV2 achieves better performance compared to its predecessor by incorporating these enhancements while maintaining low latency and reduced memory footprint.

3.4.3 VGG16 Model

VGG16 is a deep CNN architecture that was introduced by the Visual Graphics Group (VGG) at the University of Oxford in 2014. It's named "VGG16" because it comprises 16 layers, including 13 convolutional layers and 3 fully connected layers.

Key Features of VGG16:

- Simplicity and Uniformity: VGG16 is characterized by its simplicity. It follows a straightforward architecture design with small 3x3 convolutional filters used throughout the network. This uniformity in architecture contributes to its ease of understanding and implementation.
- Stacked Convolutional Layers: The network consists of multiple stacked convolutional layers followed by max-pooling layers. This stacking of convolutional layers allows the network to learn increasingly complex features from input images.
- Deep Representation: VGG16's depth enables it to learn intricate hierarchical representations of images, capturing both low-level features (edges, textures) and high-level features (shapes, objects).
- Fully Connected Layers: Towards the end of the network, there are three fully connected layers, followed by a softmax layer for classification. These layers perform the final mapping of features learned by the convolutional layers to the output classes.

3.4.4 InceptionV3 Model

InceptionV3 is a CNN architecture developed by Google as part of the Inception family. It's a successor to the earlier Inception models, designed to improve accuracy while keeping computational efficiency in tasks like image classification, object detection, and more.

Key Features of InceptionV3:

- Inception Modules: InceptionV3 uses a series of modules called Inception modules. These modules are composed of various convolutions (1x1, 3x3, 5x5) and pooling operations that occur in parallel. This parallel processing enables the network to capture features at multiple scales and complexities within the same layer.
- Factorization: InceptionV3 employs factorized convolutions to reduce computation. It uses 1x1 convolutions (known as bottleneck layers) to decrease the number of input channels before applying larger convolutions (3x3 or 5x5). This reduces the computational cost while retaining representational power.

- Auxiliary Classifiers: During training, InceptionV3 includes auxiliary classifiers in intermediate layers. These auxiliary classifiers help combat the vanishing gradient problem by providing additional supervision signals during training.
- Global Average Pooling: Instead of fully connected layers at the end, InceptionV3 uses global average pooling, which computes the average of each feature map. This reduces the number of parameters and helps prevent overfitting.

3.4.5 ResNet-50 Model

ResNet-50 is a deep CNN architecture introduced by Microsoft Research in 2015. It's part of the ResNet (Residual Network) family, known for its depth and the innovative use of residual connections.

Key Features of ResNet-50:

- Deep Architecture: ResNet-50 is deep, consisting of 50 layers. It comprises a series of convolutional layers, pooling layers, and fully connected layers, organized into blocks.
- Residual Connections: The core innovation of ResNet is the introduction of residual connections, also known as skip connections. These connections allow the network to bypass certain layers by adding the input of a layer to its output. This helps mitigate the vanishing gradient problem encountered in very deep networks and enables smoother gradient flow during training.
- Building Blocks: ResNet-50 uses residual blocks, each containing multiple convolutional layers with shortcut connections. These blocks enable the network to learn more abstract and intricate features by allowing the information to flow more efficiently through the network.
- Different Block Configurations: ResNet-50 specifically utilizes bottleneck blocks, which are composed of a sequence of 1x1, 3x3, and 1x1 convolutions. These bottleneck structures reduce computational complexity while still capturing complex features effectively.

3.4.6 Key Steps in Each Model Development

This section represents the steps that are required for with mask, without mask and incorrect mask detection using XCEPTION, MOBILENETV2, VGG16, INCEPTION, and ResNet-50 models. For these model implementation, the following steps are taken that is depicted in Figure 3.10 phase by phase.

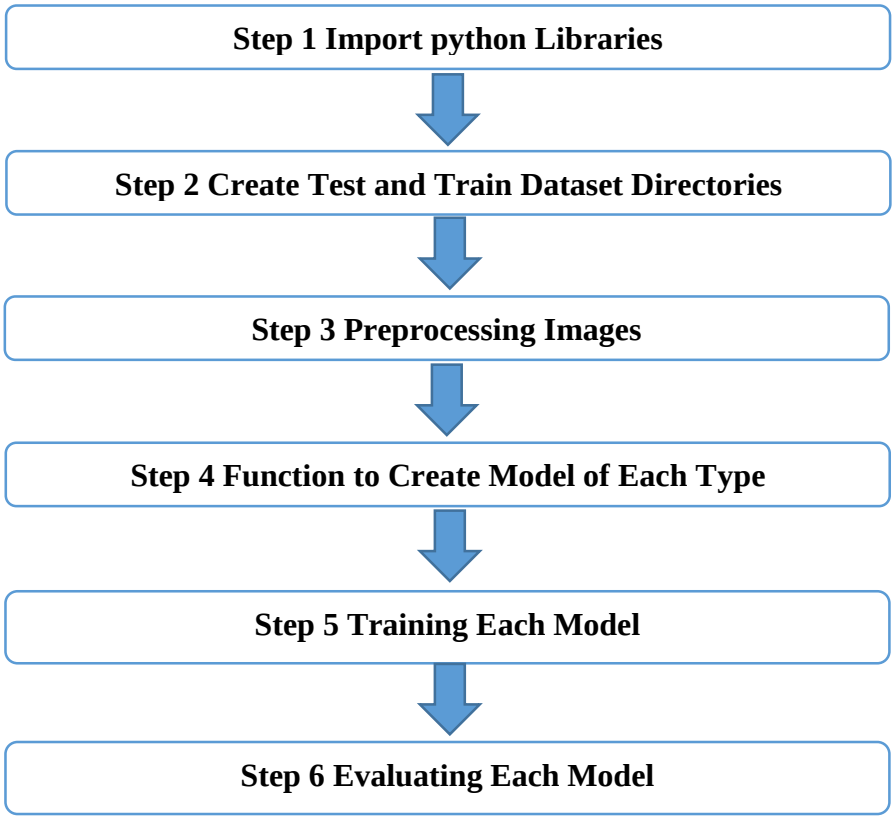

Fig. 3.10. Workflow diagram for Other Deep Learning Models

Step 1 Import python Libraries: Essential libraries and features for TensorFlow and Keras-based image categorization are imported through Python script. The first step involves loading the required modules, which include matplotlib.pyplot, glob, random, OS, Shutil, and pandas. Next, import TensorFlow and Keras to create and train deep learning models. Using Dense and Flatten layers, Keras is used to define a Sequential model. For model training, the Stochastic Gradient Descent (SGD) optimizer is used. To improve the training process, callbacks like ModelCheckpoint and EarlyStopping are also used.

For the CNN architectures, we import pre-trained models from TensorFlow's applications module, including VGG16, MobileNetV2, Xception, InceptionV3, and ResNet50.

Furthermore, metrics from Keras, such as metrics, and utilizes ImageDataGenerator for data augmentation are imported. Finally, the python library includes the ability to load pre-existing models using load_model from TensorFlow.

Step 2 Create Test and Train Dataset Directories: Creating directories and filling them with 540 train images and 60 test images of each class randomly got from input dataset. To do so ‘shutil.copy’ function is used from the Python standard library.

Step 3 Preprocessing Images: The data generator is set up for training and testing image datasets using TensorFlow's ImageDataGenerator. These include rescaling the pixel values, zooming, rotating, shifting horizontally and vertically, adjusting brightness, and enabling horizontal flipping. The flow_from_directory method is then used to generate batches of augmented images from the specified training and testing directory. Images are resized to (224, 224) pixels, and the class mode is set to 'categorical' for multi-class classification.

Step 4 Function to Create Model of Each Type: The creating_model function starts by checking to see if a directory has been set aside for storing models and, if not, generating it. The next step entails building a neural network model using a pre-trained base and adding more layers for a particular task. To preserve its learnt features, the pre-trained model's layers are frozen. After that, the model architecture—which consists of a densely connected layer with softmax activation for three classes and a flattening layer is printed for examination. The function then sets up the training parameters, including how many steps to take each epoch, and uses stochastic gradient descent (SGD) optimization with categorical crossentropy loss to train the model. During training, ModelCheckpoint and EarlyStopping callbacks are used to save the best-performing model based on validation accuracy. The function returns the training history, including metrics and losses throughout epochs, and saves the trained model in the designated directory at the end.

Step 5 Training Each Model: An instance of each model is created using the Keras library, with pre-trained weights from the ImageNet dataset. The include_top parameter is set to False, indicating that the fully connected layers responsible for classification are not included. The input_shape is specified as (224, 224, 3), defining the dimensions of the input images. Subsequently, the function creating_model is called for each model, training set (training_set), testing set (testing_set), and a string identifier ("Each Model Name"). This

function likely involves compiling and training the model on the provided datasets, and the training history is stored in a variable for further analysis or visualization.

Step 6 Evaluating Each Model: The Python function load_and_evaluate (("Each Model Name") is designed to streamline the process of loading a pre-trained model from a specified file path, generating batches of test data using an ImageDataGenerator with rescaling, and evaluating the model's performance on the test dataset. Utilizing the Keras library, the function first prepares the test dataset with the batch size of 256, target image size (224,224), and categorical class mode. It then loads the pre-trained model from the provided file path, prints a summary of its architecture, and proceeds to evaluate its performance on the test set using the evaluate method. Finally, the function prints out the various metrics obtained during the evaluation, providing a concise overview of the model's effectiveness on the given test data.

CHAPTER 4
MODEL EXPERIMENT RESULTS AND ANALYSIS

To validate the proposal, the experiments that have been carried out are described in this section. Both the ViT and CNN model experiments are done on same large custom dataset consisting of 2,03,780 digital images with 3 class labels namely 'with_mask', 'without_mask' and 'incorrect_mask'. Both the models are trained for 50 epochs with a learning rate of 0.01.

4.1 Framework and Hardware Acceleration

ML Framework

The framework that employ the design and implement the pipeline is PyTorch, since all the pre-trained models under study are implemented in the framework, and they could be easily imported through torchvision.

Hardware Acceleration

On the hardware acceleration side, Tesla P100 graphical processing unit (GPU), 16GB RAM and Intel(R) Xeon(R) @ 2.30GHz CPU with 16 core instance from Kaggle are used. The GPU plays an important role in training the models because depending on their capacity, the training phase can take more or less time.

4.2 Evaluation Metrics

To validate the performance of the model, we have calculated not only the accuracy and loss value of the model but also calculated the Precision, Recall and F1-score of the model. The performance metrics are primarily based on a comparison of anticipated and actual values that investigates number of correct and incorrect predictions from the training sample, which is divided into four categories: True Positive (TP) that is both the true and predicted values are positive; True Negative (TN) in which both the original and the anticipated values are negative.; False Positive (FP) a where the actual value is negative

but the anticipated result is positive and lastly False Negative(FN) where the actual value is positive, but the predicted result is negative.

I. Accuracy: Percentage of correct predictions.

$$\text{Accuracy} = (TP + TN) \div (TP + TN + FP + FN)$$

II. Loss: It depicts a model behavior (poor or well) after each iteration of optimization.

$$\text{Loss} = \text{running\_loss} / \text{dataset\_sizes[phase]}$$

Here, running_loss is the number of wrong predicted from the class label and dataset_sizes[phase] is the number of images in training/validation/testing phase.

III. Precision: The number of correct predictions.

$$\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive})$$

IV. Recall: The proportion of the true positives that were correct.

$$\text{Recall} = \text{True Positive} / (\text{True Positive} + \text{False Negative})$$

V. F1-score: The F1-score is the harmonic mean between the precision and recall. It is frequently used when there is an imbalance in the dataset. One crucial fact is that this metric considers how many errors the model has per class and their influence on the final performance of the model and not only the absolute number of right or wrong predictions.

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

4.3 ViT Model Experiment Results

This section will discuss about ViT model training and testing phase results.

4.3.1 Training Model Outcomes

In this section, it will discuss and analysis about ViT model's training and validation phase results. Training phase result of ViT model has been reported in table 4.1. The output is shown below:

TABLE 4.1: VIT MODEL TRAINING PHASE RESULT

Epoch No	Phase	Loss	Accuracy
1	train	0.475939756	0.9176
2	train	0.476180533	0.9223
3	train	0.478968032	0.9241
4	train	0.478741554	0.9252
5	train	0.471858906	0.9257
6	train	0.469954827	0.9259
7	train	0.476044394	0.9270
8	train	0.470223727	0.9273
9	train	0.469691821	0.9272
10	train	0.467121451	0.9276
11	train	0.466011469	0.9289
12	train	0.472262531	0.9271
13	train	0.475191999	0.9286
14	train	0.471229919	0.9294
15	train	0.468824911	0.9284
16	train	0.472513478	0.9298
17	train	0.465058104	0.9282
18	train	0.464919686	0.9293
19	train	0.470505673	0.9304
20	train	0.465607344	0.9304
21	train	0.462317548	0.9302
22	train	0.461094857	0.9299
23	train	0.46569084	0.9301
24	train	0.460432619	0.9309
25	train	0.458684348	0.9312
26	train	0.461493897	0.9317
27	train	0.461265368	0.9313
28	train	0.461978456	0.9311
29	train	0.462709434	0.9325
30	train	0.457120905	0.9312
31	train	0.456112688	0.9323
32	train	0.456547903	0.9326
33	train	0.457096238	0.9326
34	train	0.454961111	0.9322
35	train	0.455919041	0.9330
36	train	0.456583065	0.9324
37	train	0.454195875	0.9325
38	train	0.457458772	0.9328
39	train	0.45535676	0.9314
40	train	0.454719741	0.9328
41	train	0.456446203	0.9316
42	train	0.458662869	0.9326
43	train	0.452768097	0.9339

44	train	0.451778883	0.9338
45	train	0.453261572	0.9334
46	train	0.451922385	0.9333
47	train	0.450912546	0.9337
48	train	0.45420495	0.9322
49	train	0.450794311	0.9344
50	train	0.4525823	0.9330

From the Table 4.1, it clearly shows that the highest accuracy obtained by the ViT model was 0.9344 at 49th epoch of training phase.

Validation phase result of ViT model has been reported in table 4.2. The output is shown below:

TABLE 4.2: VIT MODEL VALIDATION PHASE RESULT

Epoch No	Validation Loss	Validation Accuracy
1	0.385717	0.9639
2	0.386966	0.9664
3	0.38645	0.9686
4	0.381013	0.9672
5	0.387593	0.9720
6	0.375979	0.9703
7	0.376233	0.9703
8	0.384462	0.9681
9	0.376914	0.9680
10	0.37435	0.9704
11	0.380105	0.9711
12	0.375065	0.9701
13	0.37785	0.9700
14	0.369556	0.9725
15	0.384735	0.9683
16	0.379797	0.9702
17	0.374388	0.9718
18	0.372991	0.9727
19	0.382675	0.9730
20	0.385482	0.9718
21	0.381435	0.9689
22	0.379347	0.9725
23	0.380513	0.9725
24	0.390039	0.9674
25	0.379418	0.9730
26	0.379044	0.9684
27	0.366117	0.9745
28	0.376385	0.9714

29	0.373768	0.9726
30	0.381938	0.9701
31	0.370825	0.9714
32	0.378005	0.9723
33	0.368785	0.9744
34	0.373616	0.9724
35	0.368218	0.9741
36	0.378815	0.9713
37	0.372296	0.9725
38	0.373733	0.9703
39	0.367539	0.9745
40	0.367855	0.9719
41	0.367851	0.9722
42	0.367487	0.9718
43	0.372965	0.9718
44	0.368967	0.9744
45	0.365465	0.9732
46	0.363366	0.9746
47	0.374966	0.9729
48	0.365324	0.9737
49	0.366042	0.9734
50	0.365691	0.9738

Figure 4.1 visually presents the progression of the Training and Validation phases over the course of 50 epochs in relation to their respective accuracies. It showcases how the accuracy metrics evolve across each epoch during the training and validation processes, providing insights into the model's performance and learning trends over time.

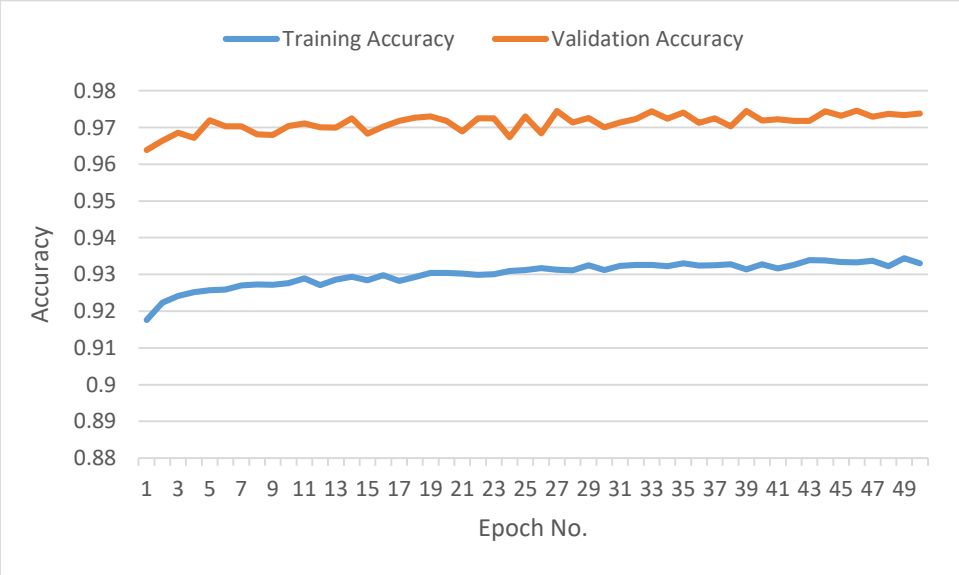

Fig. 4.1 Training and Validation Results (Epoch vs Accuracy)

Figure 4.2 visually presents the progression of the Training and Validation phases over the course of 50 epochs in relation to their respective losses. It showcases how the loss metrics regress across each epoch during the training and validation processes, providing insights into the model's performance and learning trends over time.

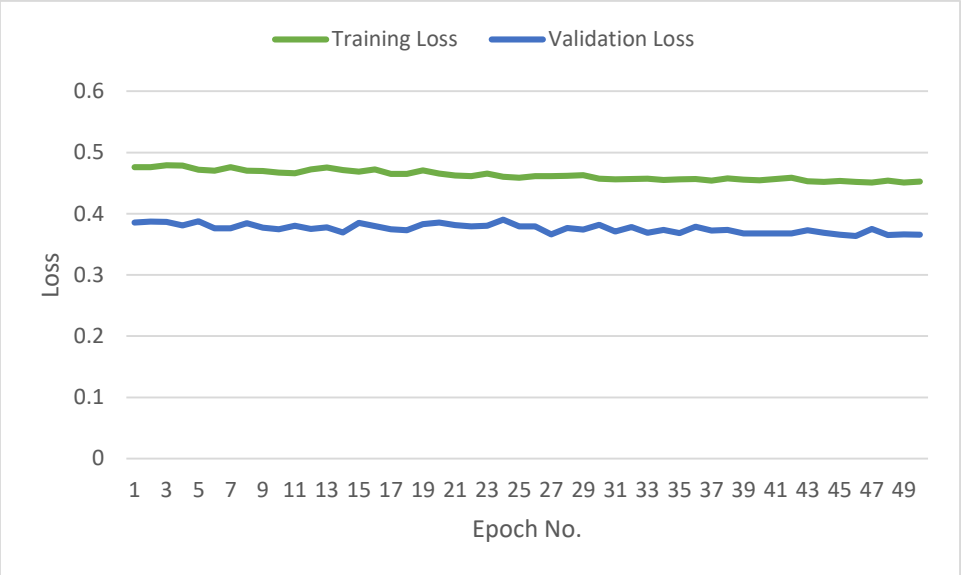

Fig. 4.2 Training and Validation Results (Epoch vs Loss)

4.3.2 Testing Model Outcomes

In testing phase, around 20,352 images of 03 categories are tested and got the accuracy result of each category which is shown in Table 4.3.

TABLE 4.3: EXPERIMENTAL RESULT OF TEST ACCURACY

Category Name	Image Count	Accuracy
with_mask	6,705	97%
without_mask	7,000	99%
incorrect_mask	6,673	95%
Total	20,378	97%

It is observed that ViT model has a great testing performance of 97% accuracy and test loss of 0.0005 in 10 minutes processing time. From the collected results of the experiment, it appears that the ViT model is able to test quickly while obtaining a high accuracy.

Precision, Recall and F1 Score:

ViT model's Precision score, Recall score and F1 score have also been calculated in three different categories as shown in Table 4.4.

TABLE 4.4: PRECISION, RECALL AND F1 SCORE OF VIT MODEL

Category Name	Precision Score	Recall Score	F1 Score
Micro	97.3063	97.3063	97.3063
Macro	97.2777	97.2760	97.2757
Weighted	97.3074	97.3063	97.3057

Grad-Cam result:

Grad-Cam (Szegedy et al., 2016) requires a gradient on a layer so that the attention of a target layer is obtained. In ViT, the gradient of the last attention layer in the architecture is used to find where the important parts are in an image. Fig. 4.3 shows the result of Grad-Cam for the with_mask, incorrect_mask and without_mask class label. Based on the visualization, image (a) shows that ViT focuses on the entire mask area of with_mask image, from image (b) ViT focuses on the part of the exposed face and the third image (c) shows that ViT model focuses on the without mask area of the face.

(a) (b) (c)

Fig 4.3 Grad-Cam results of ViT a) With mask b) Incorrect mask c) Without mask

4.4 CNN Model Experiment Results

In this research, the same experiment is also conducted on CNN model with the same dataset consisting of 03 class labels. After getting the training and testing phase results from CNN model, then a comparison is done between the ViT model results and the CNN model results. In this section, it will present the CNN model experiment results.

4.4.1 Training Model Outcomes

This section will discuss and analysis about CNN model's training and validation phase results. Training phase results of CNN model has been reported in table 4.5. The output is shown below:

TABLE 4.5: CNN MODEL TRAINING PHASE RESULT

Epoch No	Training Accuracy	Training Loss
1	0.3367	1.1044
2	0.3362	1.1018
3	0.3361	1.1019
4	0.3361	1.1018
5	0.3356	1.1019
6	0.3375	1.1017
7	0.3355	1.1018
8	0.3337	1.1019
9	0.3346	1.102
10	0.3363	1.1018
11	0.3357	1.1019
12	0.3343	1.1018
13	0.3351	1.102
14	0.3348	1.1019
15	0.3337	1.102
16	0.336	1.1019
17	0.3352	1.1018
18	0.3375	1.1018

19	0.3356	1.1021
20	0.3349	1.102
21	0.3361	1.102
22	0.336	1.1018
23	0.3369	1.1016
24	0.3366	1.1019
25	0.3354	1.1021
26	0.3357	1.1019
27	0.3353	1.1029
28	0.3365	1.1018
29	0.3349	1.102
30	0.3349	1.1019
31	0.3341	1.1021
32	0.336	1.1019
33	0.3377	1.1017
34	0.3369	1.1017
35	0.3344	1.102
36	0.3367	1.1018
37	0.3362	1.102
38	0.3376	1.1019
39	0.3365	1.1018
40	0.3336	1.102
41	0.3344	1.102
42	0.3354	1.1018
43	0.3364	1.1018
44	0.3343	1.102
45	0.3368	1.1016
46	0.3359	1.1018
47	0.3359	1.1019
48	0.3356	1.1019
49	0.3349	1.1018
50	0.3365	1.1018

From the Table 4.5, it clearly shows that the highest accuracy obtained by the CNN model was 0.3376 at 38th epoch of training phase.

Validation phase result of ViT model has been reported in table 4.6. The output is shown below:

TABLE 4.6: CNN MODEL VALIDATION PHASE RESULT

Epoch No	Validation Accuracy	Validation Loss
1	0.3282	1.1009
2	0.3388	1.0998
3	0.3388	1.1036
4	0.3388	1.1002
5	0.3388	1.1043
6	0.333	1.0988
7	0.3282	1.1156
8	0.3388	1.1028
9	0.333	1.0995
10	0.333	1.0987
11	0.3388	1.1007
12	0.3282	1.1004
13	0.333	1.1015
14	0.333	1.1005
15	0.3388	1.1009
16	0.3388	1.0989
17	0.3282	1.1032
18	0.333	1.1
19	0.3282	1.1151
20	0.333	1.0988
21	0.3282	1.1054
22	0.333	1.1
23	0.3388	1.099
24	0.3388	1.1031
25	0.3282	1.0989

26	0.3282	1.1001
27	0.3388	1.1025
28	0.3388	1.0989
29	0.3282	1.0986
30	0.333	1.0996
31	0.333	1.0998
32	0.3282	1.1005
33	0.3388	1.1083
34	0.3388	1.1013
35	0.333	1.1012
36	0.333	1.1031
37	0.3388	1.0986
38	0.333	1.0995
39	0.333	1.0994
40	0.333	1.1006
41	0.3282	1.1011
42	0.3388	1.0985
43	0.3282	1.1075
44	0.3388	1.1074
45	0.333	1.1101
46	0.333	1.1001
47	0.3389	1.0991
48	0.333	1.0998
49	0.333	1.1067
50	0.3388	1.0998

The training and validation phases are shown graphically in Figure 4.4 along with their corresponding accuracies during a 50-epoch period. It illustrates the evolution of the accuracy metrics during the training and validation phases across each epoch, offering insights into the model's performance and learning patterns over time.

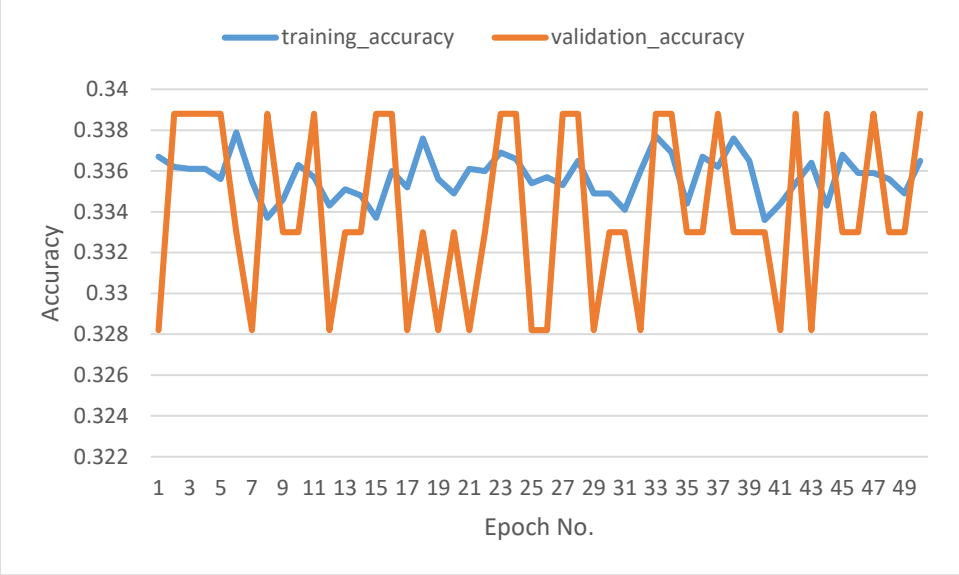

Fig. 4.4 Training and Validation Results (Epoch vs Accuracy)

Figure 4.5 visually presents the progression of the Training and Validation phases over the course of 50 epochs in relation to their respective losses. It showcases how the loss metrics behave across each epoch during the training and validation processes, providing insights into the model's performance and learning trends over time.

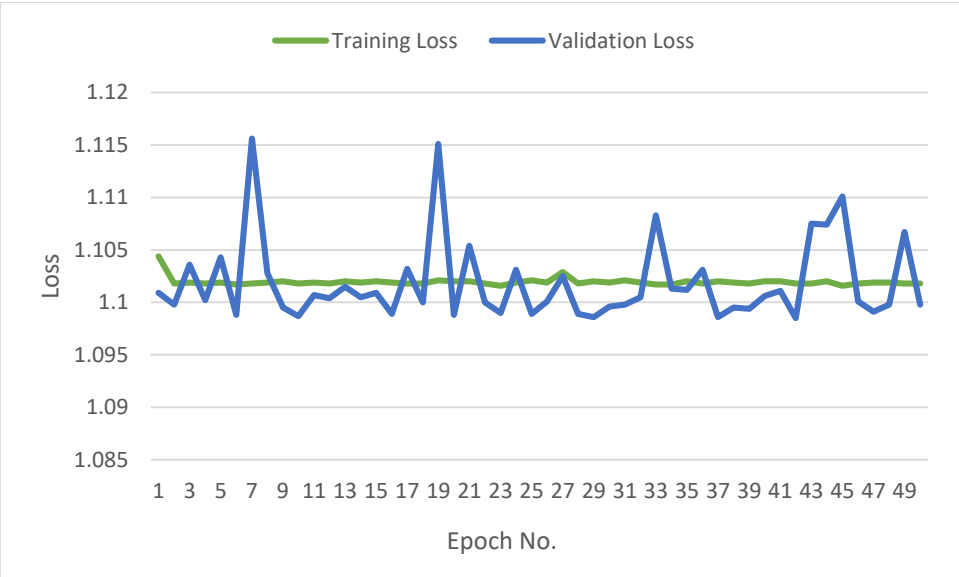

Fig. 4.5 Training and Validation Results (Epoch vs Loss)

4.4.2 Testing Model Outcomes

In testing phase, around 20,352 images of 03 categories have been tested and got the accuracy and loss result as shown in Table 4.7.

TABLE 4.7: TESTING PHASE LOSS AND ACCURACY RESULT

Phase Name	Loss	Accuracy	Time
Testing	1.0986	0.3446	19 minutes

It is observed that CNN model has a testing performance of 34.5% accuracy and test loss of 1.0986 in 19 minutes processing time. From the collected results of the experiment, it appears that the CNN model shows a significant performance.

Precision, Recall and F1 Score:

CNN model's Precision score, Recall score and F1 score have also been calculated in three different categories as shown in Table 4.8.

TABLE 4.8: PRECISION, RECALL AND F1 SCORE OF CNN MODEL

Category Name	Precision Score	Recall Score	F1 Score
Micro	34.4619	34.4619	34.4619
Macro	15.4251	38.4732	18.4734
Weighted	11.8763	34.4619	17.6649

4.5 The Outcomes Comparison between ViT and CNN Model

In the context of image classification using the ViT and CNN models, this section aims to evaluate and compare the performance of these models on the specific task of incorrect face mask detection. Undoubtedly both architecture has significant role in incorrect face mask detection.

This comparison section includes accuracy curves, loss curves, precision-recall-f1 scores and model performances. These provide insights into the model's ability to correctly identify with mask, without mask and incorrect mask wearing images.

4.5.1 Accuracy Curves

This section represents the accuracy curves for both the ViT and CNN model in training and validation phase.

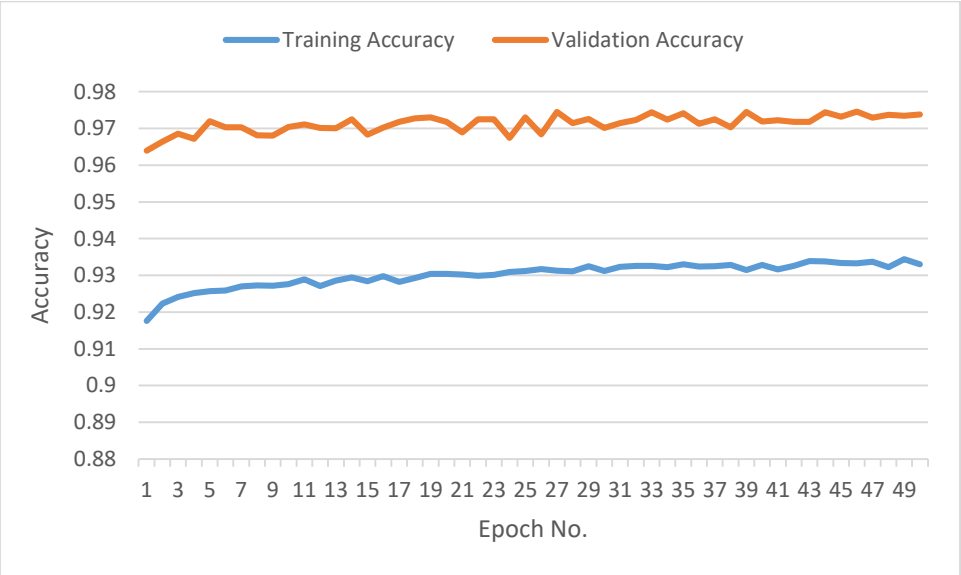

Fig. 4.6 Accuracy Curves for ViT Model

In Figure 4.6, the accuracy curves for the ViT model are presented. It is evident that the training accuracy rapidly increases and reaches almost 94% after around 50 epochs. This suggests that the ViT model quickly learns and memorizes the training data, resulting in perfect accuracy during training. The validation accuracy also shows a steady increase, indicating the model's ability to generalize well and accurately classify unseen data.

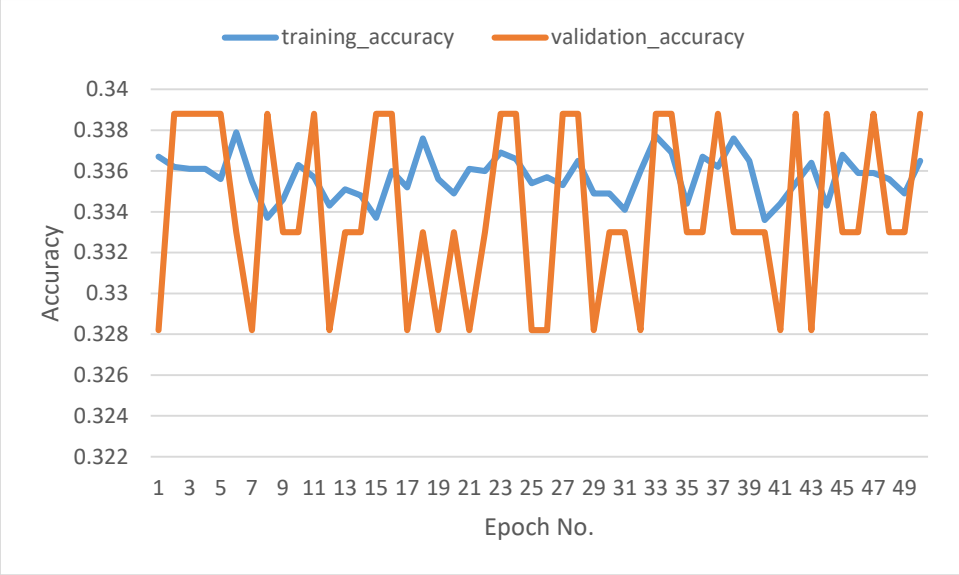

Fig. 4.7 Accuracy Curves for CNN Model

Figure 4.7 shows the training and validation accuracies over 50 epochs of CNN model. The training accuracy hovers around 0.334 to 0.337, while the validation accuracy varies between 0.3282 and 0.3388. It seems like there isn't a substantial improvement in either accuracy after the initial epochs. There's some fluctuation in validation accuracy, but it tends to stay within a certain range.

4.5.2 Loss Curves

This section represents the loss curves for both the ViT and CNN model in training and validation phase.

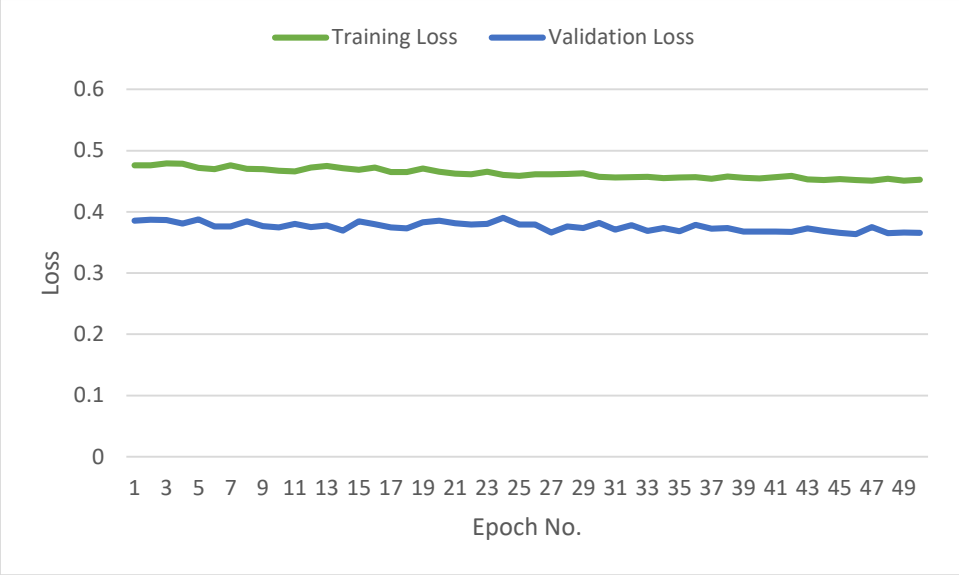

Fig. 4.8 Loss Curves for ViT Model

Figure 4.8 illustrates the loss curves for the ViT model during training and validation phase. Both the training and validation loss exhibit a significant and consistent decrease throughout the epochs. This indicates that the ViT model effectively learns the patterns and features present in the face mask images. The decreasing validation loss demonstrates the model's ability to generalize well to unseen data as it consistently improves its performance on both the training and validation sets.

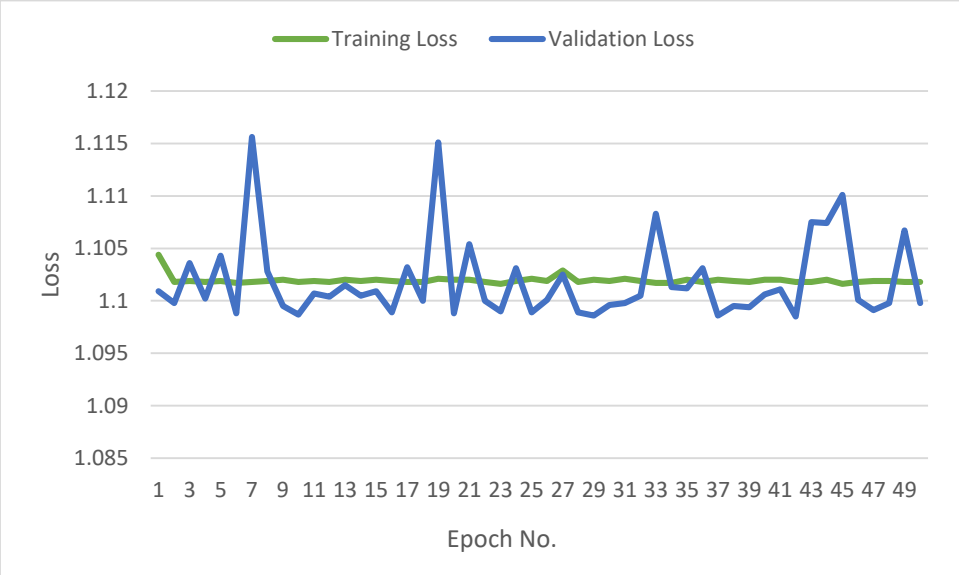

Fig. 4.9 Loss Curves for CNN Model

Figure 4.9 shows that the training and validation losses fluctuate slightly over the 50 epochs. The values of both training and validation losses seem to hover around 1.1018 to 1.102, with occasional variations. It's notable that some epochs display higher validation losses, particularly epochs 7, 19, 27, 43, and 45, suggesting potential fluctuations or challenges in generalizing the model's performance to unseen data during those specific epochs.

4.5.3 Model Performance

Model performance can be assessed using various metrics depending on the specific task and context. This section highlights the performance of both the ViT and CNN model as shown below:

TABLE 4.9: VIT AND CNN MODEL PERFORMANCE CHART

Item	ViT	CNN
Training Accuracy	94%	33.8%
Validation Accuracy	97.5%	34%
Testing Accuracy	97%	34.46%
Total Training Time (minutes)	105	218
Total Testing Time (minutes)	10	19

Computational Aspects:

Table 4.8 portrays the ViT and CNN model performances respectively for training, validation and testing phases. It takes around 105 minutes for ViT model and 218 minutes for CNN model respectively to complete the whole training and validation phase for 50 epochs. This indicates that ViT model takes lower time than CNN model to complete the whole training process and ViT model learns quickly than CNN model.

It is observed that the testing time taken by the CNN model is almost double compared to the ViT model, indicating that the ViT model has a shorter testing duration. This suggests that the ViT model has a more computationally efficient evaluation process.

Training Accuracy:

ViT model achieves a high training accuracy of 94% that means the ViT effectively learns from the training data, capturing patterns and features to a high degree of accuracy during the training phase. On the other hand, CNN model demonstrates a comparatively lower training accuracy at 33.8%. The CNN's performance on the training data is notably lower compared to the ViT, potentially indicating a less effective learning process or difficulty in capturing complex patterns within the training dataset.

Validation Accuracy:

ViT model shows exceptional performance with a validation accuracy of 97.5%. which means ViT generalizes well to unseen validation data, demonstrating a high level of accuracy and robustness in recognizing patterns beyond the training set. Unfortunately, CNN model exhibits a validation accuracy of 34%, significantly lower compared to the ViT. The CNN model's ability to generalize to new, unseen data during validation is considerably weaker than the ViT, suggesting potential limitations in its ability to capture diverse patterns.

Testing Accuracy:

ViT model maintains a high testing accuracy of 97% that implies ViT performs exceptionally well on an independent testing dataset, indicating its ability to generalize and make accurate predictions on completely new and unseen data. CNN model shows a slightly higher testing accuracy of 34.46% compared to its validation accuracy. The CNN's performance on the testing dataset is marginally better than its validation accuracy, but it still lags significantly behind from the ViT model and exhibits limited generalization capability.

In essence, the ViT displays superior performance in learning representations and generalizing to new data compared to the CNN, as evidenced by significantly higher accuracy rates in training, validation, and testing scenarios. This exceptional accuracy showcases the robustness and effectiveness of the ViT model in accurately distinguishing between with mask, without mask and incorrect face mask images.

Comparing with Precision, Recall and F1 Scores:

Table 4.10 displays precision, recall, and F1 scores for different categories (Micro, Macro, Weighted) for both ViT and CNN model.

TABLE 4.10: METRICS OF VIT AND CNN MODEL PRECISION, RECALL AND F1

Score Name	Category Name	ViT Model	CNN Model
Precision	Micro	97.3063	34.4619
	Macro	97.2777	
	Weighted	97.3074	11.8763
Recall	Micro	97.3063	34.4619
	Macro	97.2760	
	Weighted	97.3063	34.4619
F1	Micro	97.3063	34.4619
	Macro	97.2757	
	Weighted	97.3057	17.6649

ViT outperforms the CNN model significantly across all categories (Micro, Macro, Weighted) of precision score, indicating that ViT is better at making accurate positive predictions across all classes.

ViT also outperforms the CNN model across all categories (Micro, Macro, Weighted) of recall score, indicating that ViT captures a larger proportion of actual positive instances across all classes.

ViT shows better performance than the CNN model in Micro, Macro and Weighted categories, suggesting a balance between precision and recall that is superior to the CNN model.

Overall, these scores indicate that the ViT model performs notably better than the CNN model across precision, recall, and F1 score.

4.6 Compare ViT Model with other Deep Learning Models

The comparison being conducted aims to showcase why the ViT model might offer advantages in the context of detecting incorrect face mask wearing. To achieve this, the ViT model is being pitted against several other relevant models commonly used in computer vision tasks, specifically Xception, MobileNetV2, VGG16, InceptionV3, ResNet-50, and a generic CNN architecture.

The training and evaluation is being carried out on a relatively small dataset consisting of 2079 images categorized into three classes: 'with mask,' 'without mask,' and 'incorrect mask.' These classes likely represent different scenarios or states of facial mask wearing, such as correct usage, absence, or improper placement of masks.

The intent of this comparison is to highlight the strengths or superior performance of the ViT model over these other established architectures when dealing with the specific task of detecting incorrect face mask usage within these defined classes. It's essential to note that each model possesses its unique architectural features and capabilities in handling visual data.

By assessing various metrics such as accuracy, precision, recall, and possibly others relevant to this task, the goal is to demonstrate that ViT could potentially offer advantages, potentially through its ability to capture complex visual patterns or dependencies within images, enabling it to discern instances of incorrect face mask wearing more accurately or efficiently than the other models considered in this comparison.

This comparison helps in elucidating the potential suitability of ViT for this particular task and dataset, shedding light on its efficacy and potential benefits over other established models in identifying and categorizing instances of incorrect face mask usage within the specified classes.

4.6.1 Initial Configuration for Training and Evaluation

To train and evaluate all these 07 models, same number of configurations are defined. To avoid bias in the test outcomes across different configurations, consistent Hyper-parameters are maintained throughout training. The models are fine-tuned using specific Hyper-

parameters such as the rectified linear unit (ReLU) function, label-smoothing cross-entropy function, a batch size of 32, learning rate set at 0.01, total epoch 50, and employing the Adam optimizer method.

Data often arrives in formats that aren't directly suitable for training and testing all these 07 models, necessitating data augmentation or transformation. This process involves altering images to provide them with different variations that expand the dataset, aiding in better model generalization. In Python's PyTorch library, the transforms module within torchvision facilitates these manipulations, preparing the data for model training. Transformations are applied sequentially using Compose. Commonly used transformation functions include:

- Random Horizontal Flip: Flips images randomly along the horizontal axis, creating mirror images.
- Random Vertical Flip: Similar to horizontal flip but along the vertical axis.
- Normalize: Adjusts pixel values to fall within a specified range, typically by subtracting mean values and dividing by standard deviation for each channel. This helps standardize input data.
- Random Erasing: Randomly erases patches of an image, which aids in regularization and preventing overfitting by adding noise and variation to the dataset.

By chaining these transformations using Compose, the data undergoes a sequence of alterations to enhance its diversity and suitability for training, validation and testing.

The dataset has been imported and divided into training, validation, and test sets using a split ratio of 70%, 20%, and 10% respectively. This partitioning of data was achieved utilizing the 'ratio' function from the 'splitfolders' package in Python.

4.6.2 Comparing Evaluation Outcomes

This section presents a comparison of evaluation outcomes between ViT, CNN, Xception, MobileNetV2, VGG16, InceptionV3, and ResNet-50 models.

TABLE 4.11: COMPARING EVALUATION OUTCOMES AMONG VARIOUS MODELS

Model Name	Size (MB)	Accuracy	Loss	Precision	Recall
VIT	21.9	99%	0.0262	99.0542	99.0476
CNN	0.5	29%	1.1064	8.0460	28.3654
XCEPTION	80.73	52%	0.0395	0.9944	0.9944
MOBILENETV2	9.33	98.5%	0.0	1.0	1.0
VGG16	56.42	31%	0.0129	0.9944	0.9944
INCEPTIONV3	83.76	95%	0.0	1.0	1.0
RESNET-50	91.13	69%	209.30	0.5722	0.5722

Looking at the Table 4.11, here's a brief comparison across these models:

- CNN model has the smallest size (0.5 MB), while RESNET-50 is the largest (91.13 MB). VIT (21.9 MB) and MOBILENETV2 (9.33 MB) models are relatively smaller.
- VIT model shows a high accuracy of 99% but CNN (29%) and VGG16 (31%) have shown significantly lower accuracy.
- RESNET-50 has a notably high loss (209.30), while other models generally have lower loss values. MOBILENETV2 and INCEPTIONV3 have 0.0 loss, indicating better performance in this metric.
- XCEPTION, MOBILENETV2, VGG16, INCEPTIONV3, and RESNET-50 have similar precision and recall scores. VIT model demonstrates very high precision and recall values.

Overall, the VIT model distinguishes itself across various metrics, presenting commendable performance with high accuracy, low loss, and robust precision-recall values. Its efficacy in capturing intricate patterns and long-range dependencies contributes to its superior overall performance. On the contrary, traditional CNN and VGG16 models

exhibit comparatively lower accuracy, indicating potential limitations in capturing nuanced features. RESNET-50 faces challenges in terms of loss, possibly due to its depth and complexity, suggesting that while it excels in certain aspects, it may encounter difficulties in optimizing certain training objectives. These variations in performance highlight the nuanced strengths and weaknesses of each model, emphasizing the importance of selecting the most suitable approach based on specific use cases and requirements.

CHAPTER 5
DISCUSSION AND CONCLUSION

In this work, a pre-trained ViT model has been presented for classifying usage of face mask images during the COVID-19 pandemic. A novel method of ViT has been adopted in the context of incorrect face mask wearing. In this context, the images with incorrect wearing of mask have been identified. The proposed transformer-based model is able to classify the use of face masks into three different classes: With Mask (when people are wearing the mask properly), Incorrect Mask (when person are not wearing the mask according to the WHO guidelines, i.e., the mask does not completely cover the mouth and nose), and Without Mask (when people are wearing no face mask).

The different research papers have been reviewed on different approaches to the problem of the classification of face-mask-wearing images, from basic methods to those that are considered to be state-of-the-art. The research gap is also identified and finally, a very recent ViT model is introduced that is capable of performing the task assigned to this study. The vision transformer (DeiT) model is trained, validated and tested using the custom dataset of around 204K images and followed all the necessary steps that are required to classify the face mask images. Finally, this research has achieved a very high accuracy in the task using this ViT model, encouraging its use in applications requiring face mask classification stage.

Moreover, in this study a comparative analysis of ViT and CNN model has been presented for face mask image classification. Both the models have showed promising results in this specific task. However, the ViT model has demonstrated superior computational efficiency, making it a more time-efficient option for real-time incorrect face mask image classification. The study highlights the importance of choosing an appropriate model based on the specific requirements and computational constraints of the application. The findings contribute to advancing the field of image classification and provide valuable insights for researchers and practitioners working in the domain of image processing.

5.1 Limitations and Future Enhancement

Limitation:

The study has encountered certain limitations, notably the challenge in gathering face mask images across three distinct classes: ‘with_mask’, ‘without_mask’ and ‘incorrect_mask’. This task is inherently difficult, primarily due to the scarcity of accessible and high-quality face mask images. Obtaining images with good resolution, adequate quality, and clear depiction of face mask variations have posed significant challenges during data collection.

Consequently, both the ViT and CNN models are trained and tested on a restricted dataset comprising only 203,000 images. This limitation has stemmed from the scarcity of a comprehensive dataset that could adequately represent the diverse scenarios of face mask wearing across the specified classes.

Future Enhancement:

Absolutely, the envisioned development of a real-time incorrect face mask detection application integrated with the ViT model holds significant promise for various reasons.

Firstly, the integration of the ViT model into such an application enables the real-time analysis of live video feeds or images from crowded places. This would facilitate the identification of individuals whose face masks are worn incorrectly among a large crowd. By swiftly recognizing these individuals, the application could notify administrative authorities. This prompt identification allows authorities to take necessary and immediate actions. These actions might include informing or educating the individuals about the proper way of wearing face masks to mitigate the risks of COVID-19 transmission and other contagious diseases.

Moreover, this application could serve as a preventive measure in public health safety. Identifying individuals wearing face masks improperly in crowded places where the risk of disease transmission is higher could aid in containing potential outbreaks. Administrative authorities can then intervene, possibly by offering guidance, providing suitable masks, or ensuring compliance with safety protocols. The integration of this technology not only addresses the immediate concern of incorrect face mask usage but also contributes to a larger framework of disease prevention and public health management in crowded settings.

It helps in raising awareness, ensuring compliance with safety measures, and thereby reducing the risk of contagious diseases like COVID-19 spreading within communities.

However, it's important to consider ethical implications, privacy concerns, and the need for user consent in implementing such technology in public spaces. Striking a balance between public health safety and individual privacy rights will be crucial in the development and deployment of this kind of application.

REFERENCES

S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, “Rational use of face masks in the COVID-19 pandemic,” The Lancet Respiratory Medicine, vol. 8, no. 5, pp. 434-436, 2020. [Online]. Available: doi: 10.1016/s2213-2600(20)30134-x.

B. J. Cowling, Y. Zhou, D. K. Ip, G. M. Leung, and A. E. Aiello, “Face masks to prevent transmission of influenza virus: a systematic review,” Epidemiology and Infection, vol. 138, no. 4, pp. 449-456, 2010. [Online]. Available: doi: 10.1017/s0950268809991658.

S. F. Pedersen and Y. C. Ho, “SARS-CoV-2: a storm is raging,” J Clin Invest, vol. 130, no. 5, pp. 2202-2205, May 1, 2020. [Online]. Available: doi: 10.1172/JCI137647. PMID: 32217834; PMCID: PMC7190904.

D. Hussain, M. Ismail, I. Hussain, R. Alroobaea, S. Hussain, and S. S. Ullah, “Face mask detection using deep convolutional neural network and MobileNetV2-based transfer learning,” Wireless Communications and Mobile Computing, vol. 2022, Article ID 1536318, pp. 1-10, 2022. [Online]. Available: doi: 10.1155/2022/1536318.

S. Susanto, F. A. Putra, R. Analia, and I. K. L. N. Suciningtyas, “The face mask detection for preventing the spread of COVID-19 at Politeknik Negeri Batam,” in 2020 3rd International Conference on Applied Engineering (ICAE), pp. 1-5, 2020. [Online]. Available: doi: 10.1109/ICAE50557.2020.9350556.

S. Zafeiriou, C. Zhang, and Z. Zhang, “A survey on face detection in the wild: Past, present and future,” Computer Vision and Image Understanding, vol. 138, pp. 1-24, 2015. [Online]. Available: doi: 10.1016/j.cviu.2015.03.015.

A. Morciglio, B. Zhang, G. Chowell, J. M. Hyman, and Y. Jiang, “Mask-Ematics: Modeling the effects of masks in COVID-19 transmission in High-Risk Environments,” Epidemiologia, vol. 2, no. 2, pp. 207–226, May 2021. [Online]. Available: doi: 10.3390/epidemiologia2020016.

P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, “SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2,” Sustainable Cities and Society, vol. 66, p. 102692, Mar. 2021. [Online]. Available: doi: 10.1016/j.scs.2020.102692.

A. Das, M. Wasif Ansari and R. Basak, “Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV,” in 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, pp. 1-5, 2020. [Online]. Available: doi: 10.1109/INDICON49873.2020.9342585.

S. Sethi, M. Kathuria, and T. Kaushik, “Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread,” Journal of Biomedical Informatics, vol. 120, p. 103848, Aug. 2021. [Online]. Available: doi: 10.1016/j.jbi.2021.103848.

X. Liu and S. Zhang, “COVID-19: Face masks and human-to-human transmission”, Influenza and Other Respiratory Viruses, 2020. [Online]. Available: doi: 10.1111/irv.12740.

J. Tomas, A. Rego, S. Viciano-Tudela, and J. Lloret, “Incorrect facemask wearing detection using convolutional neural networks with transfer learning,” Healthcare, vol. 9, no. 8, pp. 1050, 2021. [Online]. Available: doi: 10.3390/healthcare9081050.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, “Attention is all you need,” Advances in Neural Information Processing Systems, 2017-Decem (Nips) (2017) 5999-6009. [Online]. Available: arXiv:1706.03762.

D. S. Trigueros, L. Meng, and M. Hartnett, “Face recognition: From traditional to deep learning methods,” 2018. [Online]. Available: arXiv:1811.00116v1.

W. Hariri, “Efficient masked face recognition method during the COVID-19 pandemic,” Research Square, pp. 1-8, 2020.

B. Mandal, A. Okeukwu, and Y. Theis, “Masked Face Recognition using ResNet-50,” 2021. [Online]. Available: arXiv:2104.08997v1.

E. Mbunge, S.Simelane, S. G. Fashoto, B. Akinnuwesi, and A. S. Metfula, “Application of deep learning and machine learning models to detect COVID-19 face masks – A review,” Sustainable Operations and Computers, vol. 2, pp.235–245, 2021.

M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic,” Measurement, vol. 167, p. 108288, 2021.

M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection,” Sustainable Cities and Society, vol. 65, p. 102600, 2021. [Online]. Available: doi: 10.1016/j.scs.2020.102600.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit and Neil Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” 2020. [Online]. Available: arXiv:2010.11929v2.

A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, “MaskedFace-Net - A dataset of correctly/incorrectly masked face images in the context of COVID-19,” Smart Health, vol. 19, pp. 1-5, 2021. [Online]. Available: doi: 10.1016/j.smhl.2020.100144.

F. Rahadika, N. Yudistira, and Y. Sari, “Facial expression recognition using residual convnet with image augmentations,” Jurnal Ilmu Komputer dan Informasi, vol. 14, no. 2, pp. 127-135, 2021. [Online]. Available: doi: 10.21609/jiki.v14i2.968.

E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, “AutoAugment: Learning augmentation policies from data,” CoRR, vol. abs/1805.09501, 2018. [Online]. Available: arXiv:1805.09501.

E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” 2019. [Online]. Available: arXiv:1909.13719.

G. Boesch, “Vision transformers (vit) in image recognition -2022 guide,” Aug 2022, [Online]. Available: https://viso.ai/deep-learning/vision-transformer-ViT.

P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition—CVPR, Kauai, HI, USA, vol. 1, p. I, Dec. 8–14, 2001.

H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, pp. 5325–5334, June 7–12, 2015.

S. S. Farfade, M. J. Saberian, and L. J. Li, “Multi-view face detection using deep convolutional neural networks,” in Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, Shanghai, China, pp. 643–650, June 23–26, 2015.

X. Sun, P. Wu, and S. C. Hoi, “Face detection using deep learning: An improved faster RCNN approach,” Neurocomputing, vol. 299, pp. 42–50, 2018.

S. Zhang, C. Chi, Z. Lei, and S. Z. Li, “RefineFace: Refinement neural network for high performance face detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, pp. 4008–4020, 2020.

J. Gao and T. Yang, “Face detection algorithm based on improved TinyYOLOv3 and attention mechanism,” Comput. Commun., vol. 181, pp. 329–337, 2022.

Y. Li, “Face detection algorithm based on double-channel CNN with occlusion perceptron,” Comput. Intell. Neurosci., vol. 2022, 3705581, 2022.

J. Zhang, F. Han, Y. Chun, and W. Chen, “A novel detection framework about conditions of wearing face mask for helping control the spread of covid-19,” IEEE Access, vol. 9, pp. 42975–42984, 2021.

H. Lin, R. Tse, S. K. Tang, Y. Chen, W. Ke, and G. Pau, “Near-realtime face mask wearing recognition based on deep learning,” in Proceedings of the 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, pp. 1–7, Jan. 9–12, 2021.

S. Shahinfar, P. D. Meek, and G. Falzon, “How many images do I need? Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring,” CoRR, vol. abs/2010.08186, 2020. [Online]. Available: arXiv:2010.08186.

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, “Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization,” CoRR, vol. abs/1610.02391, 2016. [Online]. Available: arXiv:1610.02391.

F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, pp. 1800–1807, July 21–26, 2017.

A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” arXiv Prepr., 2017. [Online]. Available: arXiv:1704.04861.

K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv, 2014. [Online]. Available: arXiv:1409.1556.

C. Szegedy et al., “Rethinking the Inception architecture for computer vision,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 2818–2826, June 27–30, 2016.

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 770–778, June 27–30, 2016.

APPENDIX A

Python Code for Vision Transformer Model Implementation

In [1]:
```
import numpy as np
import pandas as pd
!pip install openpyxl
import os, openpyxl
import shutil
```

```
Requirement already satisfied: openpyxl in /opt/conda/lib/python3.10/site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in /opt/conda/lib/python3.10/site-packages (from openpyxl) (1.1.0)
```

In [2]:
```
import torch
import torchvision
from torchvision import datasets
from torchvision import transforms as T
from torch import nn, optim
from torch.nn import functional as F
from torch.utils.data import DataLoader, sampler, random_split
from torchvision import models
```

In [3]:
```
!pip install timm --root-user-action=ignore
!pip install typing
!pip install typing-extensions
try:
    from typing import Literal
except ImportError:
    from typing_extensions import Literal
```

In [4]:
```
import timm
from timm.loss import LabelSmoothingCrossEntropy
```

In [5]:
```
!pip install split-folders
import splitfolders
```

```
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

In [6]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [7]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [8]:
```
import sys
from tqdm import tqdm
import time
import copy
```

In [9]:

```
epoch_data_path = '/kaggle/working/epoch_data.xlsx'
if os.path.isfile(epoch_data_path):
    os.remove(epoch_data_path)
excel= openpyxl.Workbook()
sheet= excel.active
sheet.title = 'epoch_data'
sheet.append(['epoch_no','phase','loss','accuracy'])
```

In [10]:

```
def get_classes(data_dir):
    all_data = datasets.ImageFolder(data_dir)
    return all_data.classes
```

In [11]:

```
def get_data_loaders(data_dir, batch_size, train = False):
    if train:
        #train
        transform = T.Compose([
            T.RandomHorizontalFlip(),
            T.RandomVerticalFlip(),
            T.RandomApply(torch.nn.ModuleList([T.ColorJitter()]), p=0.25),
            T.Resize(256),
            T.CenterCrop(224),
            T.ToTensor(),
            T.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)), # imagenet means
            T.RandomErasing(p=0.2, value='random')
        ])
        train_data = datasets.ImageFolder(os.path.join(data_dir, "train/"), transform = transform)
        train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True, num_workers=4)
        return train_loader, len(train_data)
    else:
        # val/test
        transform = T.Compose([ # We dont need augmentation for test transforms
            T.Resize(256),
            T.CenterCrop(224),
            T.ToTensor(),
            T.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)), # imagenet means
        ])
        val_data = datasets.ImageFolder(os.path.join(data_dir, "val/"), transform=transform)
        test_data = datasets.ImageFolder(os.path.join(data_dir, "test/"), transform=transform)
        val_loader = DataLoader(val_data, batch_size=batch_size, shuffle=True, num_workers=4)
        test_loader = DataLoader(test_data, batch_size=batch_size, shuffle=True, num_workers=4)
        return val_loader, test_loader, len(val_data), len(test_data)
```

In [12]:
```
input_dataset_path = "/kaggle/input/facemaskdatasetv3/facemask_dataset_v3"
#input_dataset_path = "/kaggle/input/face-mask-detectormask-not-mask-incorrect-mask/dataset"
output_dataset_path = "/kaggle/working/data"
if os.path.isfile(output_dataset_path):
    shutil.rmtree(output_dataset_path)
    os.remove(output_dataset_path)
```

In [13]:
```
splitfolders.ratio(input_dataset_path, output="/kaggle/working/data", seed=500, ratio=(.7, .2, .1), group_prefix=None, move=False)
dataset_path = "/kaggle/working/data"
```

```
Copying files: 203782 files [16:42, 203.22 files/s]
```

In [14]:
```
(train_loader, train_data_len) = get_data_loaders(dataset_path, 128, train=True)
(val_loader, test_loader, valid_data_len, test_data_len) = get_data_loaders(dataset_path, 32, train=False)
```

In [15]:
```
classes = get_classes("/kaggle/working/data/train/")
print(classes, len(classes))
```

```
['incorrect_mask', 'with_mask', 'without_mask'] 3
```

In [16]:
```
dataloaders = {
    "train": train_loader,
    "val": val_loader
}
dataset_sizes = {
    "train": train_data_len,
    "val": valid_data_len
}
```

In [17]:
```
print(len(train_loader), len(val_loader), len(test_loader))
```

```
1115 1274 637
```

In [18]:
```
print(train_data_len, valid_data_len, test_data_len)
```

```
142646 40755 20381
```

In [19]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
device
```

Out[19]:
```
device(type='cuda')
```

In [20]:
```
model = torch.hub.load('facebookresearch/deit:main', 'deit_tiny_patch16_224', pretrained=True)
```

In [21]:

```
for param in model.parameters(): #freeze model
    param.requires_grad = False

n_inputs = model.head.in_features
model.head = nn.Sequential(
    nn.Linear(n_inputs, 512),
    nn.ReLU(),
    nn.Dropout(0.3),
    nn.Linear(512, len(classes))
)
model = model.to(device)
print(model.head)
```

```
Sequential(
  (0): Linear(in_features=192, out_features=512, bias=True)
  (1): ReLU()
  (2): Dropout(p=0.3, inplace=False)
  (3): Linear(in_features=512, out_features=3, bias=True)
)
```

In [22]:

```
criterion = LabelSmoothingCrossEntropy()
criterion = criterion.to(device)
optimizer = optim.Adam(model.head.parameters(), lr=0.01)
```

In [23]:

```
exp_lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.97)
```

In [26]:

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=50):
    since = time.time()
    best_model_wts = copy.deepcopy(model.state_dict())
    history = []
    best_acc = 0.0
    train_loss= 0.0
    train_acc= 0.0
    valid_loss= 0.0
    valid_acc= 0.0
    for epoch in range(num_epochs):
        print(f'Epoch {epoch}/{num_epochs - 1}')
        print("-"*10)

        for phase in ['train', 'val']: # We do training and validation phase per epoch
            if phase == 'train':
                model.train() # model to training mode
            else:
                model.eval() # model to evaluate

            running_loss = 0.0
            running_corrects = 0.0

            for inputs, labels in tqdm(dataloaders[phase]):
                inputs = inputs.to(device)
                labels = labels.to(device)

                optimizer.zero_grad()

                with torch.set_grad_enabled(phase == 'train'): # no autograd makes validation go faster
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1) # used for accuracy
                    loss = criterion(outputs, labels)
```

```
                if phase == 'train':
                    loss.backward()
                    optimizer.step()
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            if phase == 'train':
                scheduler.step() # step at end of epoch

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc =  running_corrects.double() / dataset_sizes[phase]
            sheet.append([epoch+1,str(phase),float(epoch_loss),str(epoch_acc).split(",", 1)[0].replace("tensor(", "")])

            print("{} Loss: {:.4f} Acc: {:.4f}".format(phase, epoch_loss, epoch_acc))
            if phase == 'train' :
                train_loss=epoch_loss
                train_acc=epoch_acc
            if phase == 'val' :
                valid_loss=epoch_loss
                valid_acc=epoch_acc
            history.append([train_loss, valid_loss, train_acc, valid_acc])
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict()) # keep the best validation accuracy model
        print()
    time_elapsed = time.time() - since # slight error
    print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60, time_elapsed % 60))
    print("Best Val Acc: {:.4f}".format(best_acc))
    excel.save('/kaggle/working/epoch_data.xlsx')

    model.load_state_dict(best_model_wts)
    return model,history
```

In [27]:

```
model_ft,history = train_model(model, criterion, optimizer, exp_lr_scheduler)
```

In [28]:

```
test_loss = 0.0
class_correct = list(0 for i in range(len(classes)))
#print (len(classes))
class_total = list(0 for i in range(len(classes)))
model.eval()

for data, target in tqdm(test_loader):
    data, target = data.to(device), target.to(device)
    with torch.no_grad(): # turn off autograd for faster testing
        output = model(data)
        #print(output)
        #print(len(target))
        loss = criterion(output, target)
    test_loss = loss.item() * data.size(0)
    _, pred = torch.max(output, 1)
    correct_tensor = pred.eq(target.data.view_as(pred))
    correct = np.squeeze(correct_tensor.cpu().numpy())
    if len(target) == 32:
        for i in range(32):
            label = target.data[i]
            class_correct[label] += correct[i].item()
            class_total[label] += 1

test_loss = test_loss / test_data_len
print('Test Loss: {:.4f}'.format(test_loss))
for i in range(len(classes)):
    if class_total[i] > 0:
        print("Test Accuracy of %5s: %2d%% (%2d/%2d)" % (
            classes[i], 100*class_correct[i]/class_total[i], np.sum(class_correct[i]), np.sum(class_total[i])
        ))
    else:
        print("Test accuracy of %5s: NA" % (classes[i]))
print("Test Accuracy of %2d%% (%2d/%2d)" % (
            100*np.sum(class_correct)/np.sum(class_total), np.sum(class_correct), np.sum(class_total)
        ))
```

In [30]:

```
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

In [31]:

```
def mtrx(model):
    y_pred = []
    y_true = []
    model.eval()
    for data, target in tqdm(test_loader):
        y_true = y_true + target.tolist()
        data, target = data.to(device), target.to(device)
        with torch.no_grad():
            output = model(data)
            loss = criterion(output, target)
            _, pred = torch.max(output, 1)
            y_pred = y_pred + pred.tolist()
    print("Micro F1: {:.4f}".format(f1_score(y_true, y_pred, average='micro') * 100))
    print("Macro F1: {:.4f}".format(f1_score(y_true, y_pred, average='macro') * 100))
    print("Weighted F1: {:.4f}".format(f1_score(y_true, y_pred, average='weighted') * 100))
    print("----------------------------")
    print("Micro Precision: {:.4f}".format(precision_score(y_true, y_pred, average='micro') * 100))
    print("Macro Precision: {:.4f}".format(precision_score(y_true, y_pred, average='macro') * 100))
    print("Weighted Precision: {:.4f}".format(precision_score(y_true, y_pred, average='weighted') * 100))
    print("----------------------------")
    print("Micro Recall: {:.4f}".format(recall_score(y_true, y_pred, average='micro') * 100))
    print("Macro Recall: {:.4f}".format(recall_score(y_true, y_pred, average='macro') * 100))
    print("Weighted Recall: {:.4f}".format(recall_score(y_true, y_pred, average='weighted') * 100))
```

In [32]:

```
mtrx(model_ft)
```