“Masked Face Recognition Using Deep Learning”

A PROJECT REPORT

SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted By

Rachit Mann

(2K19/SWE/09)

Under the supervision of

Dr. Manoj Kumar
Associate Professor
Department of Computer Science & Engineering
Delhi Technological University, Delhi

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

JUNE, 2021

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Rachit Mann, 2K19/SWE/09 student of M.Tech (SWE), hereby declare that the project entitled **"Masked Face Recognition Using Deep Learning"** which is submitted by me to the Department of Software Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Software Engineering, has not been previously formed the basis for any fulfilment of the requirement in any degree or other similar title or recognition.

This report is an authentic record of my work carried out during my degree under the guidance of Dr. Manoj Kumar.

Rachit Mann

Place: Delhi **Rachit Mann**

Date: 9th June, 2021 **(2K19/SWE/09)**

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the project entitled **"Masked Face Recognition Using Deep Learning"** which is submitted by Rachit Mann (2K19/SWE/09) to the Department of Software Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Software Engineering, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this university or elsewhere.

[illegible]

Place: Delhi

Date:

Dr. Manoj Kumar

SUPERVISOR

Associate Professor

Dept. of Computer Science & Engineering

ACKNOWLEDGEMENT

I am very thankful to **Dr. Manoj Kumar** (Associate Professor, Department of Computer Science) and all the faculty members of the Department of Computer Science and Software Engineering at DTU. They all provided us with immense support and guidance for the project.

I would also like to express my gratitude to the University for providing us with the laboratories, infrastructure, testing facilities, and environment which allowed us to work without any obstructions.

I would also like to appreciate the support provided to us by our lab assistants, seniors, and our peer group who aided us with all the knowledge they had regarding various topics.

Rachit Mann

(2K19/SWE/09)

Abstract

The COVID-19 crisis has brought significant changes in our lives. Social distancing has become a norm. Masks are part of our daily life. We cannot leave out homes without wearing masks. These masks have become like an integral organ for survival. But these masks have caused problems with our computer models for face recognition. Face recognition is the sub-field of computer science in which the computer matches an input face to a corresponding set of output images to deduce the identity of the face provided as input to the system. But with the masks, the face is covered from nose till neck. Only the eyes and forehead region is visible with a mask on the face. This creates a problem with existing techniques available in the domain of face recognition using computers. The computer models achieving accuracy over 95% for a face drops to a very low level when the same face is given as input with a mask on it. The predictions made by these models are no better than random predictions made by untrained models.

In this project, the performance of different state-of-the-art models has been studied. A modified version of the existing dataset is utilized for training and testing these models. The modification is done by augmenting the masks on faces in the chosen part of the VGGface dataset. For faster training and testing, the concept of transfer learning plays a big role. The pre-trained models are being adapted to the modified dataset. Apart from this, a new model is also introduced which is somewhat a hybrid of the best performing models. This new model architecture is defined and then trained and tested on the modified dataset. The new model is thrown against the state-of-the-art models. The aim of developing this new model is to improve over the existing baselines present. Based on the closed observations, the research questions are answered.

CONTENTS

List of Figures

List of Tables

List of symbols and abbreviations

Abbreviations	**Full Form**
AUC	Area Under ROC Curve
COVID-19	Coronavirus disease 2019
CNN	Convolutional Neural Network
DenseNet201	Densely Connected Convolutional Networks
FFHQ	Flicker Faces HQ3 Dataset
FN	False Negative
FP	False Positive
FPGA	Field Programmable Gate Arrays
FPS	Frames Per Second
InceptionResNet	Inception Residual Neural Network
InceptionV3	Inception Network Version 3
JSON	JavaScript Object Notation
LBP	Local Binary Patterns
LNMF	Local Non-negative Matrix Factorization
MCF	Masked Correlation Filters
MobileNetV2	Mobile Neural Network Version 2
MTCNN	Multi-Task Cascaded Convolutional Neural Network
NMF	Non-Negative Matrix Factorization

PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
ResNet50V1	Residual neural Network (50 deep layers) Version 1
ResNet50V2	Residual neural Network (50 deep layers) Version 2
RGB	Red-Green-Blue
RGGNet	Residual Geometric Group Network
ROC Curve	Receiver Operating Characteristic Curve
SAE	Stacked Auto Encoder
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SFNMF	Spatially Confined Non-negative Matrix Factorization
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
VGG16	Visual Geometry Group Neural Network (16 deep layers)
Xception	Extreme Inception
YOLOV3	You Only Look Once Neural Network - Version 3

CHAPTER 1

INTRODUCTION

1.1 General

Neural networks and Artificial Intelligence have played a great role in influencing our day-to-day activities. AI has shown its influence on various domains such as computer vision, biomedical research, etc. AI models are being embedded into appliances ranging from large-scale appliances such as Smart TV, Refrigerators, etc. to sophisticated computing devices such as mobile phones. In the year 2020, the SARS-CoV-2 alias COVID-19 has brought the development to the halt. Day-to-day activities needed contactless working in the COVID-19 crisis. AI showed its potential for helping in these day-to-day activities. Robots controlled by AI neural network models are helping doctors to monitor and treat patients in a contactless manner. AI neural networks are also being utilized by scientists to run simulations for the spread of infection and diseases. These are also helping the people at the helpdesk by assisting the customers. AI also helped the government to track the infection and make predictions so that preventive measures can be taken at the right moment.

AI is slowly changing the conventional ways of computing. In the conventional programming paradigm, the problem is divided into sub-problems and then these sub-problems are assigned computing resources (programmers finding pattern and coding) for generating solutions. But, in the case of AI, an enormous volume of data of a problem is feed to a neural network that deduces patterns based on inputs to predict output with high accuracy and efficiency. In the AI field, deep learning is the most influential sub-field of machine learning which itself is a sub-domain of AI (as shown in Fig. 1).

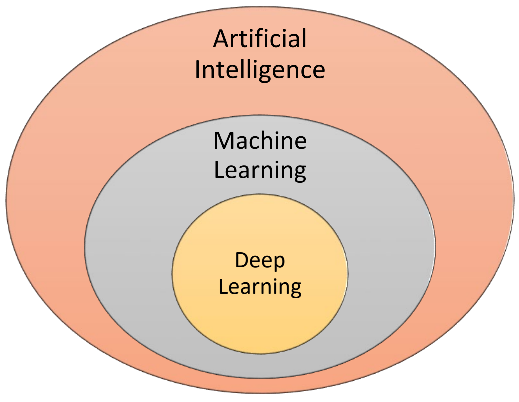


Fig. 1: AI and Deep Learning

Deep learning is being used to solve a variety of problems in various domains such as image pattern analysis. Face recognition is one such domain. Face recognition is currently being the most popular whether it be a traffic challan system or personal device authentication. Face recognition techniques vary from simple Machine learning techniques such as PCA, SVM to complex neural networks. In AI, machine learning such as SVM, decision trees, etc. involves input, feature extraction, and classification for outputs. But in deep learning, feature extraction and classification are combined for predicting output (as shown in Fig. 2).

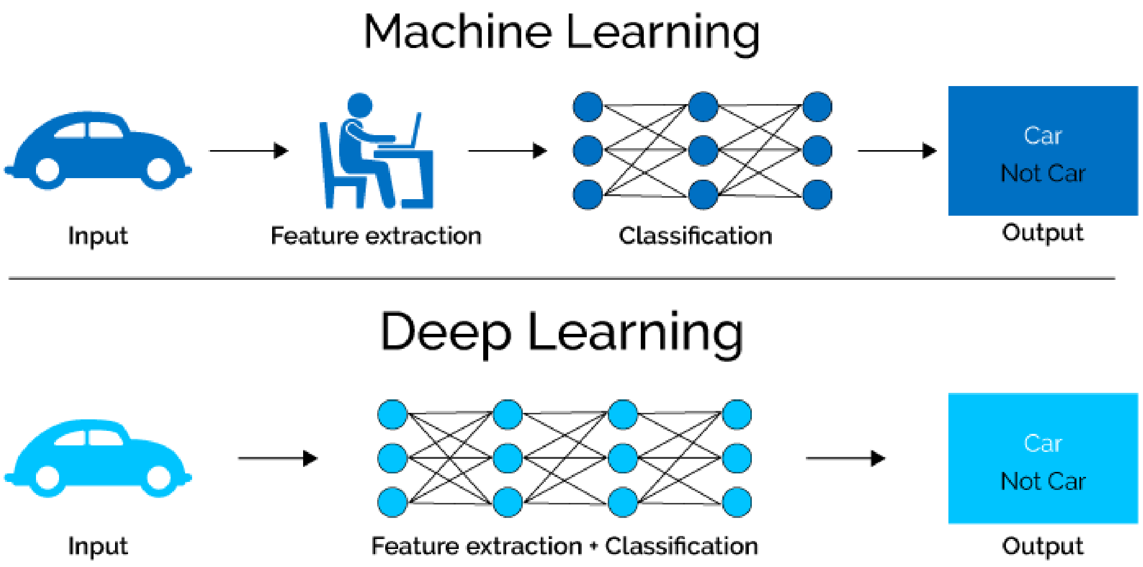


Fig. 2: Machine learning vs Deep Learning

In this project, we are dealing with face recognition using neural networks. Due to the COVID-19 crisis, face masks have become mandatory for humans to protect themselves but these face masks have an unseen side-effect on the existing face recognition system. The majority of a face recognition system requires the complete face for identification of an individual and face masks have covered the majority of facial features that act as unique identifiers for the existing face recognition systems.

1.2 Problem Formulation

When visiting banks, individuals need to be identified to issue the amount against the cheque. If the account holder itself has come, he/she need to show his/her face so that his/her identity can be verified but due to COVID-19, masks have become part of the daily routine. For such identification, one needs to put away his/her mask exposing him/her to the risk of getting the virus. To avoid this problem, a need is found to develop a face recognition technology that can accommodate masks. This face recognition technique must utilize the limited facial features available to it and based on that individuals must be identified. Based on this problem following questions has been identified:

1. What is the current scenario of face recognition with the masks?
2. What are the existing models available for deep learning?
3. Which datasets are available for face recognition particularly those involving masks?
4. What is the accuracy achieved by a face recognition system with masks?
5. How can the achieved accuracy be retained while reducing the time to upgrade the existing systems?

1.3 Objectives of the Project

In this project, the following objectives need to be achieved:

1. Generating a custom dataset for training and validating the deep learning model.
2. Training and validating the performance of deep learning models.
3. Comparing the performance of different deep learning models available.
4. Establish a baseline for improvement of the deep learning models in the future.
5. Develop a new model that combines the capabilities of top-performers based on the baseline developed.
6. Comparing the performance of the developed model with other models.

CHAPTER 2

LITERATURE REVIEW

In this section, detailed information related to the variety of research papers concentrating on face recognition using different methodologies, challenges present in face recognition, and which techniques can perform better, etc. has been listed. This section also provides insight into the deep learning models currently being used in face recognition. Furthermore, the effect of occlusion specifically the mask on face recognition and available methods to tackle these problems have also been discussed in this section. This section provides a brief insight into the previous works done in the field of face recognition.

- *Neo, H.F. at al. (2010)* [1]: This paper presents the idea of occlusions on the face recognition techniques. This paper aims to develop a framework for partial face recognition, i.e., faces occluded by objects like sunglasses, face masks, etc. This proposes a framework utilizing PCA/NMF/LNMF/SFNMF for feature extraction and then using an L2 classifier for genuine/ imposter identification. This framework was tested on approx. 2000 images of 100 subjects with each having 20 images. This framework was able to achieve 95.17% accuracy by using SFNMF for the bottom region of the face, i.e., faces with occlusion on the eyes region.
- *Su, Y. et al. (2015)* [2]: This paper presents the occlusion detection method by fusing the raw images with residual images using an SVM classifier. This method utilizes NMF to generate reconstructed images and residual images which are then divided into the upper and lower region. The generated regions are then fed to PCA for feature extraction and dimensionality reduction. The extracted features are fed to SVM at two-level. At the first level extracted features are fed to the SVM and then the result is fed to SVM at the second level which detects the occlusion on faces in each component. The experiment utilized a dataset of 960 normal faces, 720 images of faces with sunglasses, and 720 images of faces with a scarf. The proposed method was able to achieve 91.9% accuracy on images with sunglass but accuracy dropped to 79.4% with scarfs.
- *He, E. J. et al. (2016)* [3]: This paper presented the ideas discussed above but using three larger datasets with one consisting of over 4000 images of 126 subjects. It

proposes the use of MCF and for face recognition. This methodology achieved higher accuracy against the observed datasets but required prior knowledge of the occluded region of the face for achieving higher accuracy. This prior knowledge provides compensation for the presence of occlusion on the face.

— *Lin, S. et al. (2016)* [4]: This paper proposes the use of a deep learning model for partially occluded faces in the videos. The proposed method utilized a 5 layer each consisting of convolution and sub-sampling operations. The dataset consisted of 1140 images with 240 positive instanced and 960 negative instances. To reduce the overfitting problem due to an imbalanced dataset, the training instances were increased to twice using horizontal reflections. The experiment is used for the binary classification problem, i.e., to detect masked faces in the dataset. If a masked face is available then it is detected in the video stream. The proposed methodology achieved an f1-score of 0.803 with recall at 0.925 and precision at 0.71 which are reported higher than other techniques under consideration such as Adaboost. This paper presented the idea that deep learning models can achieve high performance compared to the traditional machine learning methods such as SVM, Adaboost, etc.

— *Wang, M. et al. (2017)* [5]: Earlier papers discussed using a deep learning model for occlusion detection on the face whereas this paper proposes the use of a deep learning model for face recognition. This paper doesn't consider occlusion for faces but discusses the performance of deep learning models on faces compared to traditional techniques while also proposing an own deep learning model for face recognition. The proposed method achieved the highest recognition rate of 96.6 whereas traditional methods only achieved a recognition rate of 83.5 (PCA), 89.7 (LBP), and 92.3 (SAE). Thus, this paper opens the approach of deep learning towards face recognition. The proposed MaskNet has advantages of powerful generalization ability and brilliant performance combined with the classification advantage of the softmax regression model. This model also reduces the cost of training.

— *Coskun, M. et al. (2017)* [6]: This research document provides details of an experimental evaluation of CNN based face recognition system. The noticeable qualities of the proposed algorithm are that for the initial and final layers of convolution, batch normalization is utilized which allows the neural network to achieve higher accuracy. Apart from this, the fully connected layer is connected to a softmax classifier. Georgia Tech Face database, composing 15 images each for 50 individuals, has been utilized as the testing dataset for the proposed model. The model results have shown significant recognition accuracy when compared to the literature studies presented in the paper. This paper shows how optimization can increase the achievable accuracy of CNN architecture and the potential of CNN in the field of face recognition.

— *Wan, W., & Chen, J. (2017)* [7]: This research document proposed the idea of using deep learning methods for face recognition on occluded faces. The paper proposed a trainable module called MaskNet which optimized the existing CNN architecture. MaskNet provided a clear distinction between occluded areas and non-occluded areas of the face. MaskNet improved the accuracy of ResNet by 1.4% to 3.0%. A fixed-size image is provided as input to MaskNet, followed by a low convolutional network. Then a regression layer is fully connected to the last convolution output in MaskNet. For facial recognition, occluded areas corrupt the training process, so MaskNet assigns more weight to non-occluded areas thus, decreasing the importance of the occluded area in the training and therefore, improves the existing CNN architectures.

— *Guo, G., & Zhang, N. (2018)* [8]: This paper presents the various challenges of deep learning in the face recognition field particularly in an unrestricted environment (non-ideal environment). The paper investigates the roles of quality on the performance of deep learning models. Face images with different qualities were assembled. The quality of images varied into three categories: low, medium, and high. The assembled images are then fetched to state-of-the-art deep learning techniques to investigate the role of quality and its impact on performance on those learning techniques. Two publicly available dataset of 21,230 images (High: 1,543 – Medium 13,941 – Low: 6,196) and 106,863 images (out of these 20,895 images consisting 10089 high, 10444 medium and 362 low quality were selected) are used. VGGFace, FaceNet, and other CNN-based architectures have been observed and found that recognition rate is greatly influenced by quality. For high-quality images, the recognition rate is on the higher side compared to low and medium-quality images.

— *Elmahmudi, A., & Ugail, H. (2018)* [9]: This paper presented the idea of determining the performance of deep learning methods when partial faces are used, i.e., either nose, eyes, or their combination or top-half or ¾ face, etc. The paper validates that a low recognition rate is achieved when utilizing partial face parts such as cheek, forehead, and mouth whereas higher accuracy rates can be achieved using left or right

½ of the face, top half. A 100% accuracy rate is possible for the ¾ of the face. The paper utilizes VGG-Face architecture in combination with SVM and Cosine Similarity classifier for conducting the experiments to validate the hypothesis. This paper acts as a baseline for state-of-the-art models to improve upon and thus providing direction for further research in this field of face recognition using deep learning architectures.

— *Qu, X. et al. (2018)* [10]: This research paper promises a fast face recognition approach using parallelization and networking for faster training of the deep neural networks. The proposed method separates the understanding process of CNN into two parts: network training on the PC and network implementation on the FPGA. After software simulation and board measurement, the speed of the face recognition system was 400 FPS with a 99.25% recognition rate along with good vigor under various light environments.

— *Tomodan, E. R., & Caleanu, C. D. (2018)* [11]: This research manuscript provided a comparative study between the traditional machine learning approach of the bag of features and deep CNN. The basic difference between these two is that in deep CNN, the features are automatically extracted from low-level to high-level whereas, in the bag of features, these features need to be hand-tuned. The manuscript presents the problems in CNN and providing research directions. It provided details on how datasets size, background variations, and images per class influence the learning process of deep neural networks.

— *Wu, G. et al. (2019)* [12]: This proposed paper implements the InceptionResnetV1 model for occlusion-based face recognition. The dataset utilized for this model has about 30% occlusion to the faces. This dataset consists of 4,234 images and the model was able to 98.6% recognition rate. The model is robust to the occlusions while retaining a high recognition rate for practical applications.

— *Ejaz, M. S. et al. (2019)* [13]: This paper performs analysis of face recognition using the PCA method for both masked faces and non-masked faces. The non-masked face has a better recognition rate in PCA based face recognition system whereas, for a person who is wearing a mask, a poor recognition rate is observed. Extracting features from a masked face is less than a non-masked face which causes a reduction in features ultimately degrading the recognition rate. The paper concludes that PCA is good for normal face recognition but not for masked face recognition.

— *Khan, S. et al. (2019)* [14]: This research paper presents the use of transfer learning on deep learning models for face recognition. The paper proposes the modification of existing layers of pre-trained models according to the dataset on which face recognition is required. This process of modification is known as transfer learning. The transfer learning is performed on pre-trained AlexNet. This paper only studies face recognition without occlusions. The transferred model was able to achieve an accuracy of 97.95%.

— *Ejaz, M. S., & Islam, M. R. (2019)* [15]: This paper proposes the use of FaceNet CNN in combination with SVM. The proposed framework utilized the MTCNN to detect the face in the dataset and then use the extracted faces as input to the FaceNet to create face embedding (face vector). The generated face vector is used by SVM to recognize the face under test with the existing vectors. This model proposed face verification, i.e., one-to-one mapping of the face. One-to-one mapping means the test face is cross-matched a single identity using L2 normalization by SVM.

— *Bhuiyan, M. R. et al (2020)* [16]: This paper proposed the use of YOLOv3 architecture for detecting masks on the faces, i.e., a person is wearing a mask or not. The model achieved a 17 FPS average for detection on video stream and a precision score of 0.96.

— Adjabi, I. *et al (2020)* [17]: This review paper provided insight into the face recognition methodologies. The document presented the techniques utilized in the past for face recognition, the present approaches, and trends in face recognition, and the future challenges that are needed to be tackled in the future.

— *Cabani, A. et al (2020)* [18]: This paper proposed a dataset of correctly and incorrectly masked faces. The dataset was created while considering the COVID-19 crisis. Apart from the dataset, it also provided the details of generating a similar dataset with a different set of images. The proposed dataset has been generated using the FFHQ dataset available online by NVIDIA. The proposed dataset consisted of 137,016 good-quality images. The dataset is proposed to act as a benchmark for the models that are utilized for detecting the masked faces specifically whether the masks are correctly worn or not by a person.

— *Li, Y. et al (2021)* [19]: This paper proposed an optimization module (CBAM) to crop the masked faces for better recognition. The CBAM module is created to focus on the areas surrounding the eye region. This helped to achieve better performance for masked face recognition compared to the other attention-based models. An optimal value of 0.7L is utilized for cropping the faces and this allows CBAM to focus on eyes and their surrounding regions. An increase of 17.427% and 18.507% has been achieved using this approach.

Out of all the paper reviewed in this section following concluding points are noted:

1. Initial papers were limited to traditional machine learning techniques. As the deep learning techniques evolved, their practicality for face recognition has become apparent. No analysis has been performed among different deep learning techniques for occlusion-based face recognition.
2. Most of the papers use occlusion up to 60-70% in the dataset for analyzing the techniques. Only one paper studied face recognition with just individual parts such as eyes, mouth, etc. but the accuracy achieved was quite low.
3. The dataset size in most cases is limited to 5-20 images per subject and a size varying from 100 images to 4000 images. The dataset size is quite low if considered from the deep learning point of view. So a need for a larger dataset is required for extensive training and testing of the deep learning models.
4. Apart from the small dataset, one paper presented the use of optimization by cropping for better results. This proposed approach can be utilized for generating better models.
5. Transfer learning is studied from the surface only in deep learning models utilized for face recognition. Transfer learning provides a way to use pre-trained models and re-training pre-trained models require relatively less computational power than training a deep learning model from scratch. Generally, 10 epochs are more than enough for transfer learning.

CHAPTER 3

THEORETICAL CONCEPTS

This section presents the basic theoretical concepts required to understand the key processes and working of the experiment studied in this project. This section familiarizes the concept of deep learning, transfer learning, and pre-trained models available. It also induces the idea of working on different kinds of layers utilized by a variety of pre-trained models. The concepts introduced in this section help to understand the proposed architecture for masked face recognition.

3.1 Face Recognition

The process of cross-matching a face against one or more faces is described as face recognition. The face recognition task can be explained as the problem of a supervised learning task that uses predictive modeling. A model is trained on a set of inputs and outputs in a predictive modeling task. The model tries to learn the relation between inputs and outputs and is required to give predictions / expected output for a given input.

The 2011 book titled "Handbook of Face Recognition" explains the following categories of face recognition task (as shown in Fig. 3):

- Face Verification: A one-to-one matching of face against an identified face, i.e., a person's face under test is matched against a particular face available in the records present. Example: A photo id verification.
- Face Identification: A one-to-many matching of face against all the identified faces, i.e., face under test is matched against every known face to identify the actual identity. Example: Who is the person?

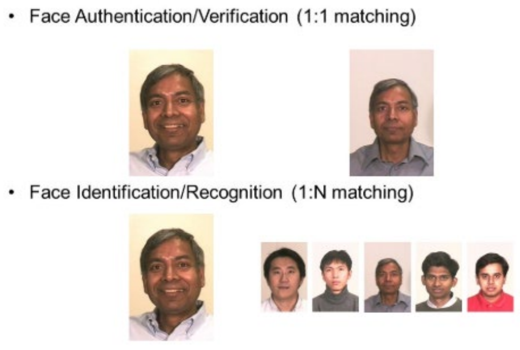

• Face Authentication/Verification (1:1 matching)
• Face Identification/Recognition (1:N matching)

Fig. 3: Face Verification vs Face Identification

"A face recognition system is expected to identify faces present in images and videos automatically. It can operate in either or both of two modes: (1) face verification (or authentication), and (2) face identification (or recognition)."

– (Li and Jain, 2011) [20]

In this project, we are dealing with the face identification task. This task is selected since the literature review revealed that masked face recognition paper consisted of two categories for classification namely, masked or not-masked. Furthermore, the masked category was further divided into correctly worn or incorrectly worn in some cases. The challenge of face identification was not explored in such cases.

3.2 Basic Concepts of Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is an algorithm in deep learning taking images as a form of input, allocating some weights/ the significance factor to the different features/aspects present in the image, and then use the assigned factor value to differentiate the various aspects from each other. CNN pre-computation is on the far lower side compared to other classification learning techniques. In classical techniques, the feature filtering methods are manually provided, i.e., manually constructed whereas, in CNN, CNN itself learns those filters using enough training, and this is termed as an epoch.

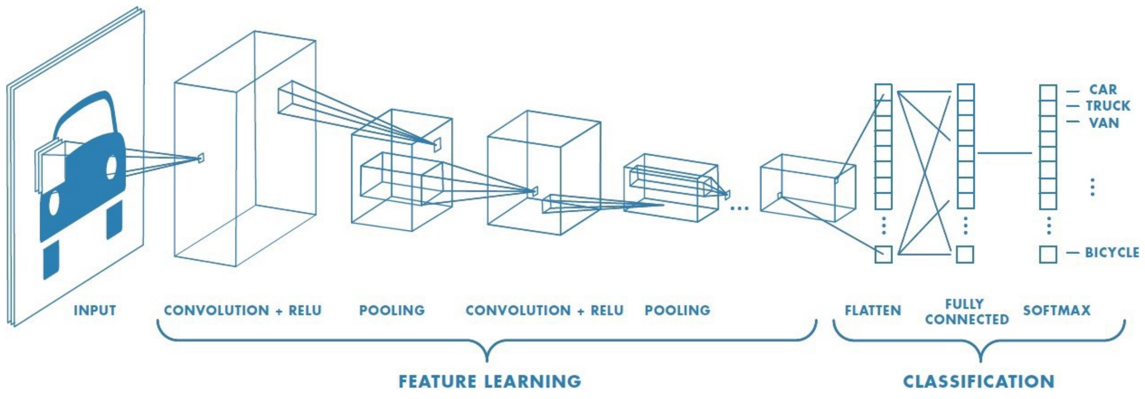


Fig. 4: A Typical CNN Network

Basically, an image is a collection of pixels. CNN can easily detect the spatial and temporal relations of the pixels present in the image by applying relevant filters. CNN performs better fitting of the images by reduction of parameters and reusability of weights. CNN converts the image into a much smaller form which is easier to process without losing critical features as shown in Fig. 4. This leads to high prediction accuracy with the added advantage of scalability.

Just like neural networks, CNN consist of a set of neurons that memorizes features and biases as weights and filters. Instead of learning each pixel, each neuron receives a collection of pixels as inputs (typically convolved pixels), takes a weighted sum over them, uses the resultant output as input to the activation function, and then generates corresponding output. The whole network has a loss function to adjust the weight according to the training epoch.

3.3 The Architecture of CNN:

A typical CNN has the following layers:

- Convolutional layers (Acting as feature maps for the model)
- ReLU layers (Optimizes the feature map using ReLU activation function)
- Pooling layers (Groups together the features map to reduce the size of the input)
- A Fully connected layer (Predicts the output based on the input and feature map)

These layers are placed in the stacked form to form a CNN. The arrangement can be like this:

“ *Input → Convolution → ReLU → Convolution → ReLU → Pooling → ReLU → Convolution → ReLU → Pooling → Convolution → ReLU → Pooling → Fully Connected → Output* ”

3.3.1 Layers of CNN:

1. Convolutional Layer

   A convolutional layer in a CNN has defined feature maps which are utilized to recognize the patterns and shapes in the input in a stratified/ranked manner.

   A convolution operation is a simple multiplication operation performed using a set of weights and an input. The 2-D array of weights is known as a filter or a kernel. This filter has a smaller size compared to the input data. A dot product multiplication is performed between the set of weights (filter) and the input set.

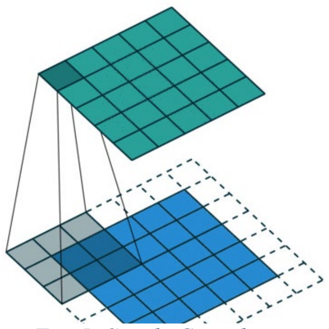
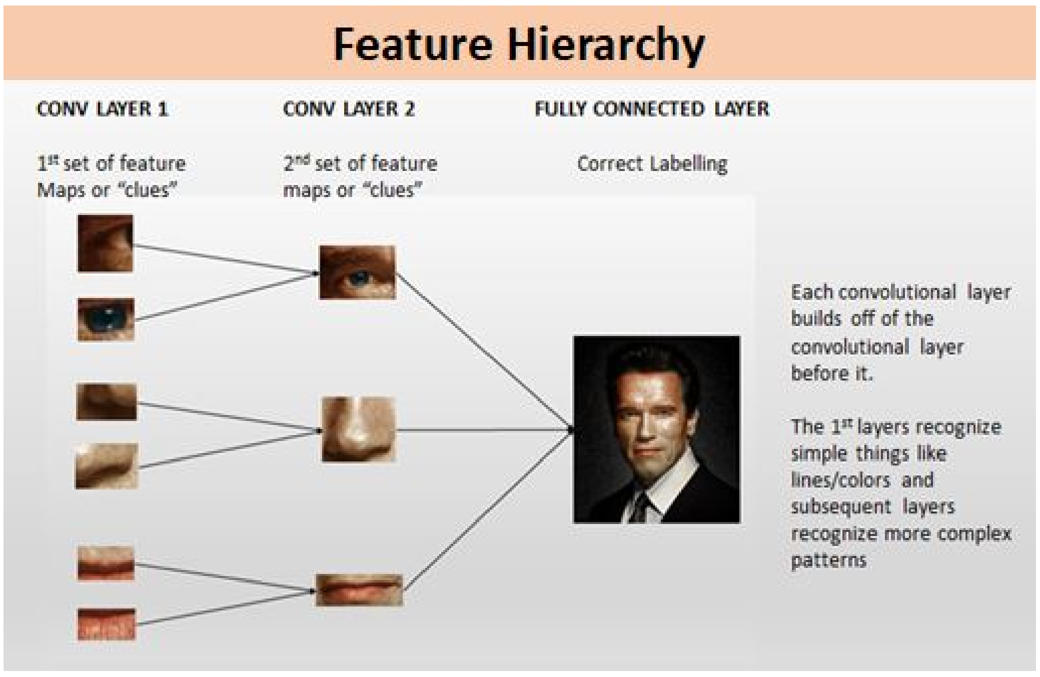


Fig. 5: Simple Convolution Operation

This filter multiplication is applied symmetrically over the input image as shown in Fig. 5.

CNN pinpoints the most useful features of an input image using the pattern recognition of the numbers. CNN piles up these patterns in form of a heap which progressively builds the complex feature maps as shown in Fig. 6.



Feature Hierarchy
CONV LAYER 1
1st set of feature Maps or "clues"
CONV LAYER 2
2nd set of feature maps or "clues"
FULLY CONNECTED LAYER
Correct Labelling
Each convolutional layer builds off of the convolutional layer before it.
The 1st layers recognize simple things like lines/colors and subsequent layers recognize more complex patterns

Fig. 6: Feature Hierarchy in CNN

Fig. 7 is representing the multiplication operation performed by the Convolution layer on different channels(dimensions such as RGB) of the image to generate a convoluted matrix.

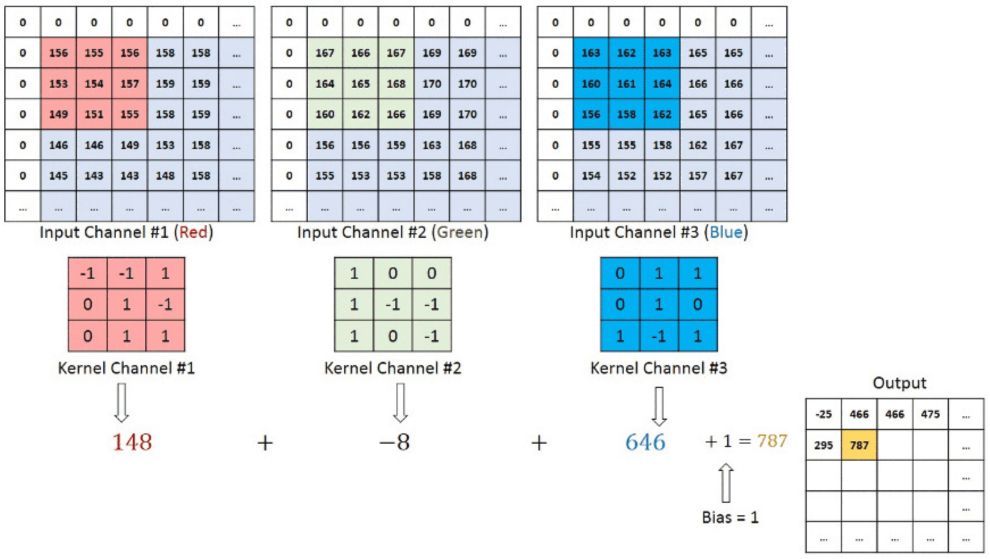


Fig. 7: Typical Convolution operation on MxNx3 image matrix with 3x3x3 Kernel

2. ReLU (Rectified Linear Unit) layer

   ReLU layer is an on/off switch in CNN (shown in Fig. 8). The output (feature map) of the convolution layer is moved through the ReLU. Typically, if the value is negative, it is changed to 0 else remains unchanged. The convolution layer processes the input linearly but this layer introduces non-linearity in the feature map. This ReLU layer corresponds to the activation function in a simple neural network. ReLU is the most popular function. Other functions such as tanh, sigmoid are also available for use.

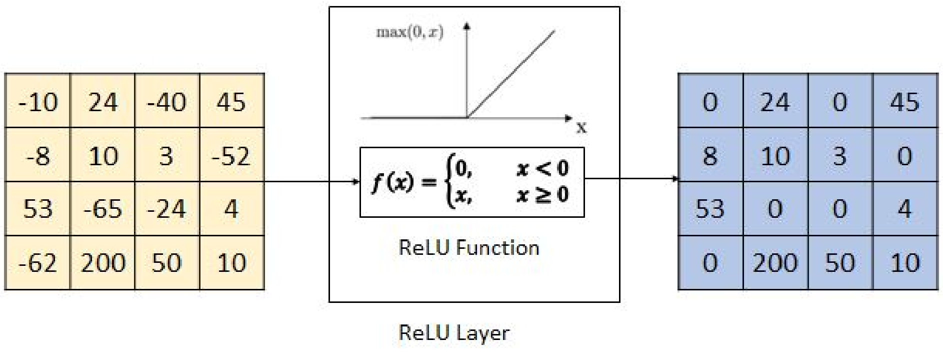


Fig. 8: ReLU Layer

3. Pooling Layer

   Feature map (output of convolutional layer + ReLU layer) keeps note of the specific position of features in the input. This leads to a problem in case the input is altered, i.e., rotated, cropped, or shifted.

   To overcome this problem, a downsampling technique is applied. In downsampling, the input is downsized to lower resolution input but still maintaining a high amount of fundamental information. This technique is applied by the pooling layer. The pooling layer is added after the ReLU operation has taken place.

   The pooling layer executes symmetrically over each feature map generated to create a new set of feature maps.

   Pooling consists of choosing a pooling operation (average or maximum) to be applied on feature maps. The result of the pooling operation will be downsized version of feature maps generated from the convolution layer.

   Pooling operation has two types of function (shown in Fig. 9):

- Average Pooling: For each subset of the feature map, the average value is calculated.
- Max Pooling: For each subset of the feature map, the maximum value is calculated.

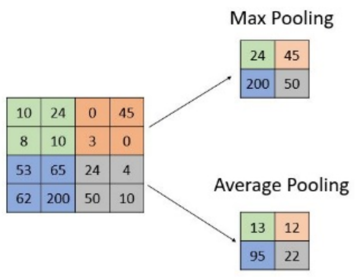


Fig. 9: Types of Pooling

Typically, max pooling is the most commonly used operation.

4. Fully Connected Layer

In simple words, the process of joining the features with each other to deduce some result is the fully connected layer. The feature map obtained from the earlier layer is flattened to vector, i.e., changed into a 1-D vector. The fully connected layer (corresponds to classical neural network) takes the flattened vector is as input. This layer then captures the high-level links between these complex features obtained and deduce some results. This layer returns a 1-D feature vector. This obtained vector typically represents the output of the CNN (shown in Fig. 10).

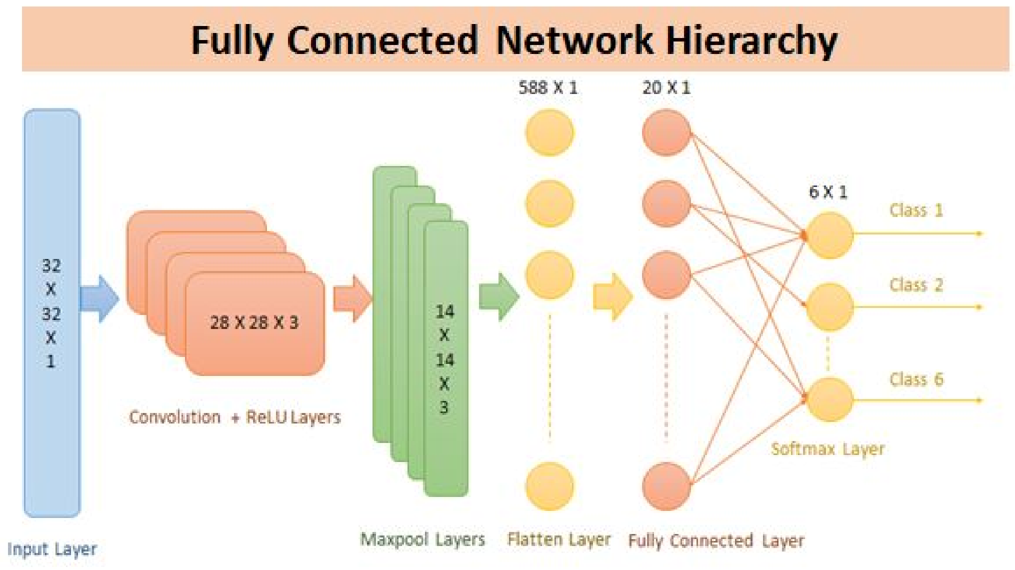


Fig. 10: Fully Connected Layer connected after a sequence of convolution, ReLU, and max pool layers

Using feed-forward network and back-propagation, the model is trained for a series of iterations (epochs). This back-propagation helps the model to change its weights and improve the distinguishing ability of the model between features using the Softmax classification technique.

The functioning of CNN layers in the union is as follows:

1. An image is provided as input to the first layer.
2. The input is fed to the convolution layer which generates a feature matrix/map.
3. The output of the convolution layer is fetched to the ReLU function to obtain a feature map to intensify the irregularities.
4. Pooling operation is applied to every batch of feature matrix by the pooling layer.
5. The pooled map is flattened into a long 1-D vector.
6. The vector is fed to the fully connected neural network which provides the probability of the classes.
7. This process is repeated for several epochs along with back-propagation for generating a high-quality model.

3.4 Concept of Transfer Learning in the field of Deep Learning

In transfer learning, pre-trained models are utilized to train new models. Existing knowledge (weights, feature matrix, etc.) from already trained models are utilized for training (as shown in Fig. 11).

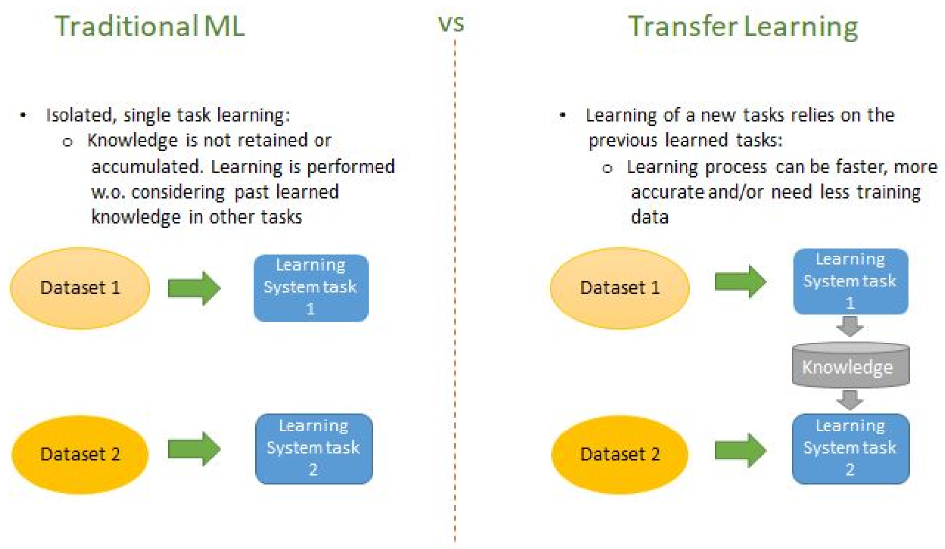

Traditional ML
VS
Transfer Learning
• Isolated, single task learning:
○ Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
• Learning of a new tasks relies on the previous learned tasks:
○ Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System task 1
Dataset 2
Learning System task 2
Knowledge

Fig. 11: Traditional ML vs Transfer Learning

Deep learning prototypes are instances of inductive learning techniques where the goal is to deduce relations from training data, i.e., try to infer mapping using the examples provided as input during the training of the model.

To deduce these relations, the techniques work on a set of assumptions termed inductive bias. These biases are influenced by hypothesis, search strategy, and various other factors.

The inductive bias is the one that impacts the learning of a task in a domain, i.e., what should be learned, how it should be learned from given task data of a domain.

For applying the concept of transfer learning to deep learning, inductive transfer techniques are utilized which use the original target task biases to guide the model for the new target task. The details of the architecture of different deep learning models are publicly accessible as pre-trained models architecture. These pre-trained models are utilized for transfer learning and this form of transfer learning is termed deep transfer learning.

Deep transfer learning can exist in two forms of implementation (either Freezing or Fine-Tuning):

1. In the freezing implementation, the input layer and final fully connected layer are adjusted according to the destination dataset and freeze the hidden layers of the network (as shown in Fig. 12).

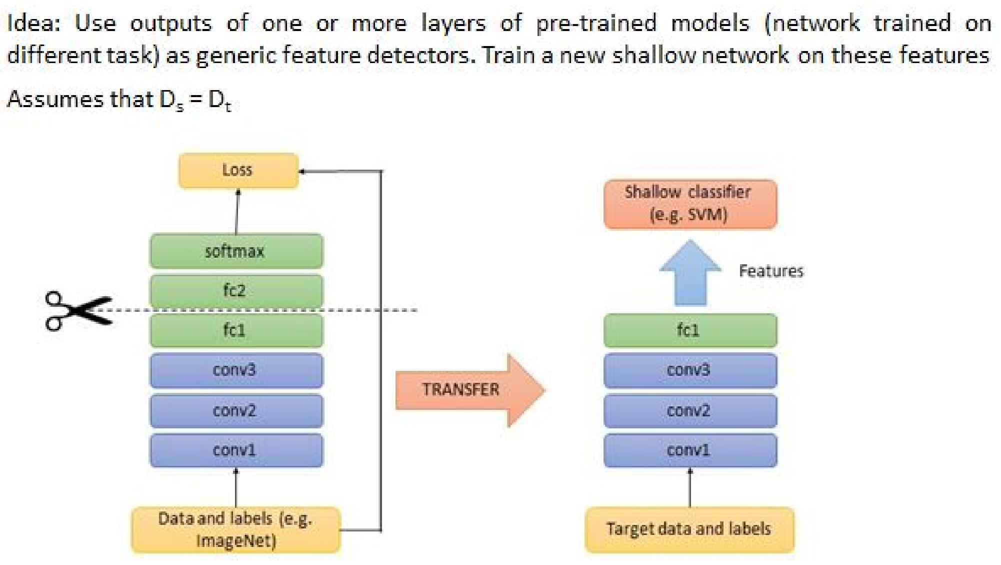
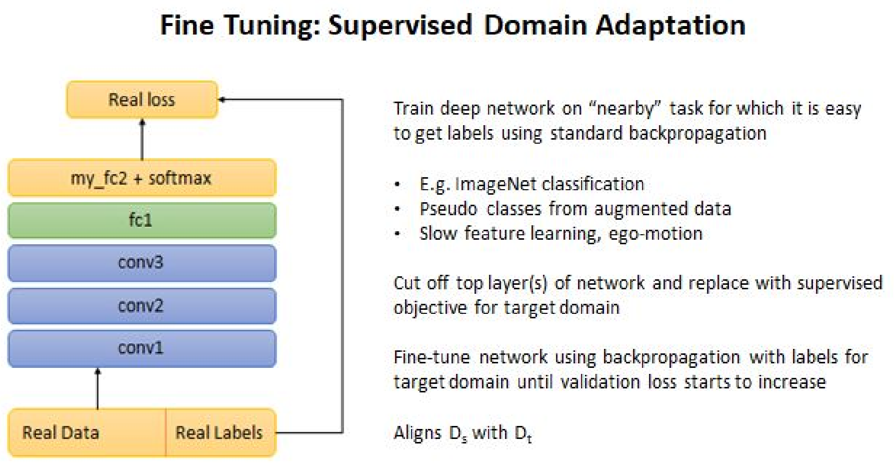

Idea: Use outputs of one or more layers of pre-trained models (network trained on different task) as generic feature detectors. Train a new shallow network on these features
Assumes that $D_s = D_t$
Loss
softmax
fc2
fc1
conv3
conv2
conv1
Data and labels (e.g. ImageNet)
TRANSFER
Shallow classifier (e.g. SVM)
Features
fc1
conv3
conv2
conv1
Target data and labels

Fig. 12: Transfer learning via freezing

2. In the fine-tuning implementation, adjustment of the input layer and final fully connected layer according to the destination dataset as well as training of the hidden layers of the network for finer adjustments is done (as shown in Fig. 13).



Fine Tuning: Supervised Domain Adaptation

Train deep network on "nearby" task for which it is easy to get labels using standard backpropagation

- E.g. ImageNet classification
- Pseudo classes from augmented data
- Slow feature learning, ego-motion

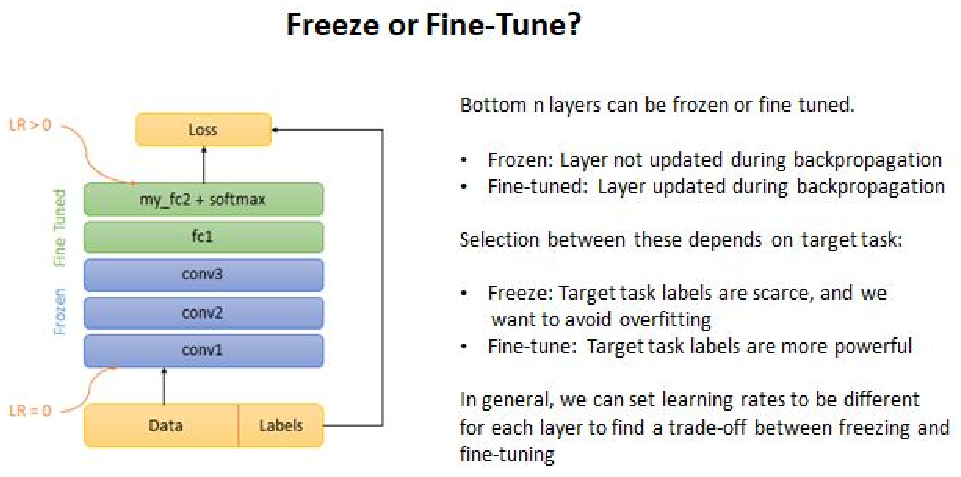
Cut off top layer(s) of network and replace with supervised objective for target domain

Fine-tune network using backpropagation with labels for target domain until validation loss starts to increase

Aligns D_s with D_t

Fig. 13: Transfer learning via fine-tuning

Choosing the strategy for transfer learning depends on the type of task and the domain on which transfer learning is required. The process for choosing the strategy for transfer learning is shown in Fig. 14.



Freeze or Fine-Tune?

Bottom n layers can be frozen or fine tuned.

- Frozen: Layer not updated during backpropagation
- Fine-tuned: Layer updated during backpropagation

Selection between these depends on target task:

- Freeze: Target task labels are scarce, and we want to avoid overfitting
- Fine-tune: Target task labels are more powerful

In general, we can set learning rates to be different for each layer to find a trade-off between freezing and fine-tuning

Fig. 14: Freeze or Fine-Tune

For the field of computer vision, some of the popular pre-trained models available are VGG16, DenseNet121, ResNet-50, InceptionV3, and Xception.

CHAPTER 4

PROPOSED MODEL FOR FACE RECOGNITION

This section presents the proposed model, i.e., RGGNet that is being utilized in this project. This section familiarizes the architecture of RGGNet. It also helps to understand the relation of the RGGNet with the ResNet50V2 and VGG16. It presents the details of the layer and the position at which the layers are introduced.

4.1 RGGNet Architecture

RGGNet architecture is based on the architecture of ResNet50V2 along with the features of VGG16 along the shortcut paths. RGGNet model is composed of three blocks (CONV, IDENTITY*, and POOL block) clubbed together. Each block has two paths: The main path and the shortcut path. The main path represents standard convolution, ReLU, and pooling operations whereas the shortcut path represents the unique attribute that differs each of these blocks. The detailed representation of these blocks is as follows:

1. CONV Block

   A standard CONV block is utilized in the RGGNet architecture. A standard CONV block (as shown in Fig. 15) represents the case when the input has a different shape compared to the output activation. CONV block is utilized to resize the dimensions of the image.

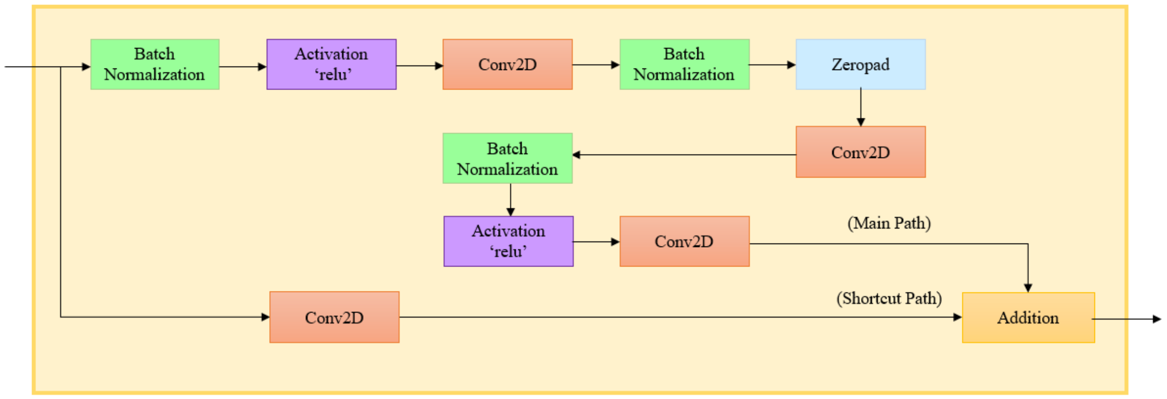


Fig. 15: CONV Block

2. IDENTITY* Block

   A standard identity block represents the case when the input function has the same dimensionality as the output function. A standard Identity block has two paths:

Main Path and the skip connection path. The skip connection directly connects the input to the block with the output of the main path.

But in this case, this identity block has been modified. Instead of skip connection, we have utilized a shortcut path with a single convolution layer as shown in Fig. 16. This allows additional feature extraction in the skip connection. The input to the identity block is processed on the main path that helps in retrieving complex features whereas the shortcut path utilizes the Conv2D layer to deduce simpler features. These simple and complex features together help to understand the image properties better.

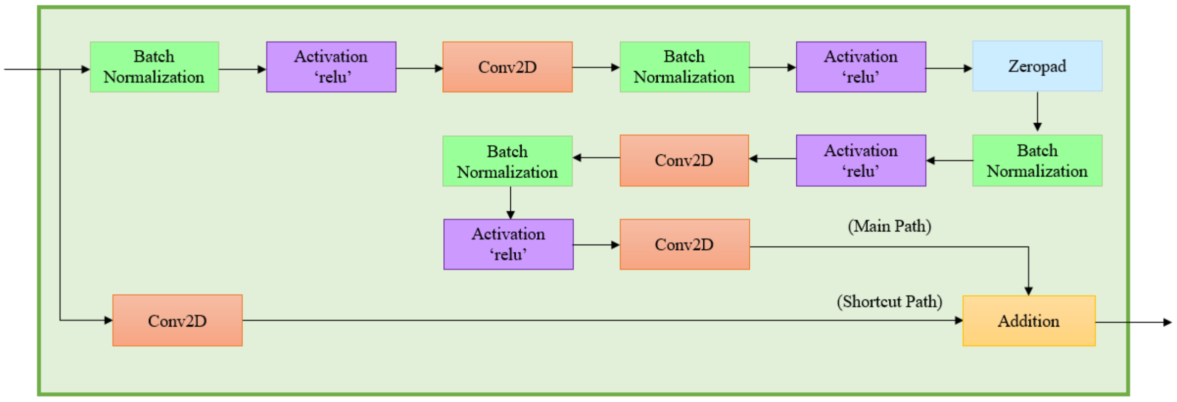


Fig. 16: IDENTITY Block*

3. POOL Block

A standard POOL block is utilized to gather the most prominent features from the input image. A POOL block (as shown in Fig. 17) performs the max pooling operation on the image.

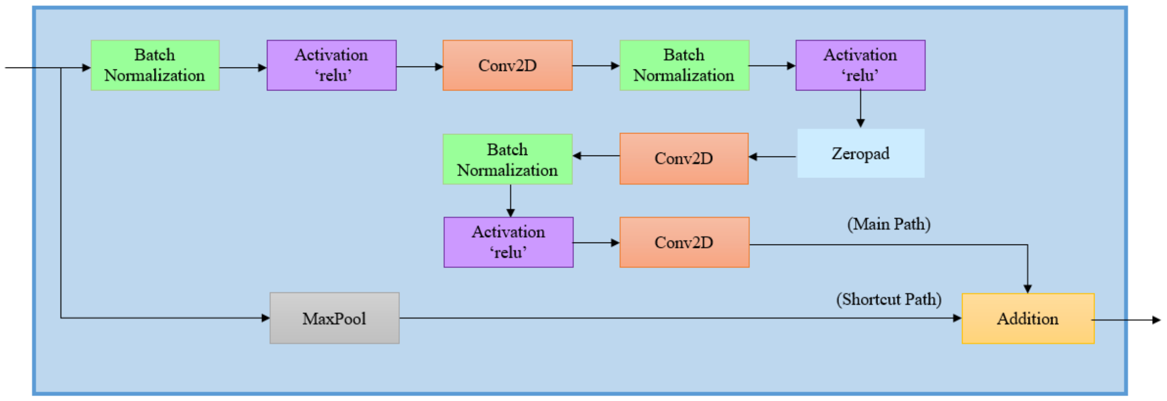


Fig. 17: POOL Block

By combining these three blocks, the shortcut path in the RGGNet has layers in the same manner as that of VGG16 thus giving the higher capabilities compared to the standard architectures. The combined architecture is shown in fig. 18.

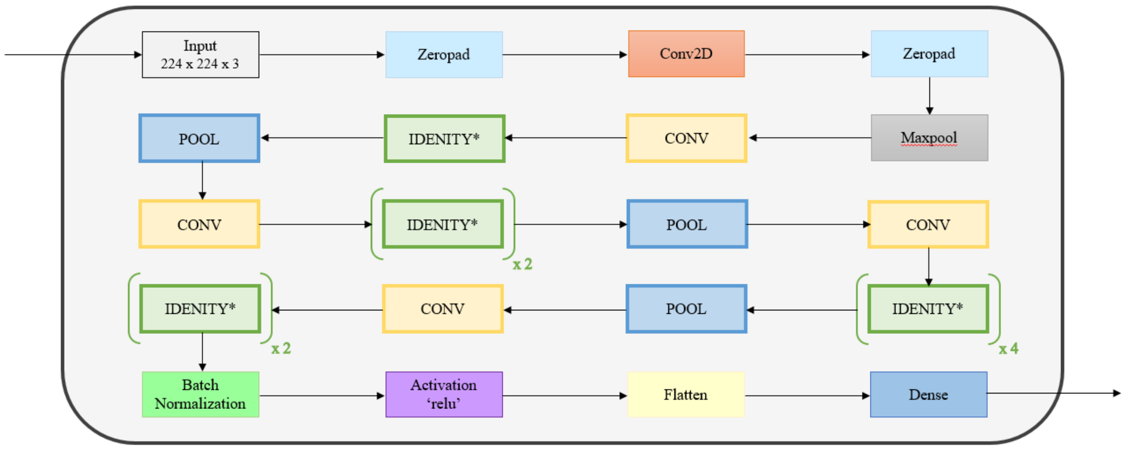


Fig. 18: RGGNet Architecture

The proposed RGGNet architecture is designed for face identification with more than 43 million parameters as shown below:

Table I: RGGNet Parameters

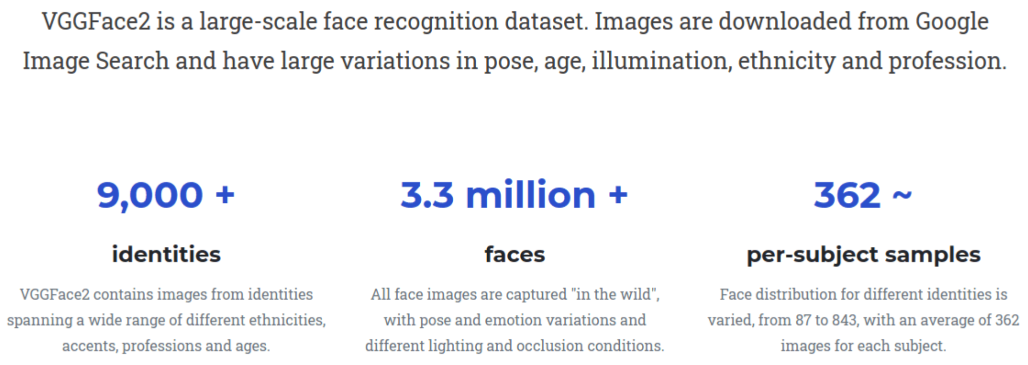
Parameters	ResNet50V2	VGG16	RGGNet
Total params	26,360,385	15,180,673	43,269,953
Trainable params	2,795,585	465,985	43,224,513
Non-trainable params	23,564,800	14,714,688	45,440

CHAPTER 5

EXPERIMENTAL SETUP

5.1 Dataset

In the experiment, a modified version of the vggface2 dataset[1] is utilized. The description of the vggface2 dataset is shown in Fig. 19.



VGGFace2 is a large-scale face recognition dataset. Images are downloaded from Google Image Search and have large variations in pose, age, illumination, ethnicity and profession.

9,000 +

identities

VGGFace2 contains images from identities spanning a wide range of different ethnicities, accents, professions and ages.

3.3 million +

faces

All face images are captured "in the wild", with pose and emotion variations and different lighting and occlusion conditions.

362 ~

per-subject samples

Face distribution for different identities is varied, from 87 to 843, with an average of 362 images for each subject.

Fig. 19: VGGFace2 Description

Pre-processing of part of the dataset is done in which masks are augmented on the face of individuals and then the faces are cropped. The pre-processing of the dataset is taking place in four steps:

a. Find Face Landmarks

Python library "face-alignment" is utilized to generate coordinates of features of the face and then use those obtained features for extracting mask coordinates (Fig. 20).

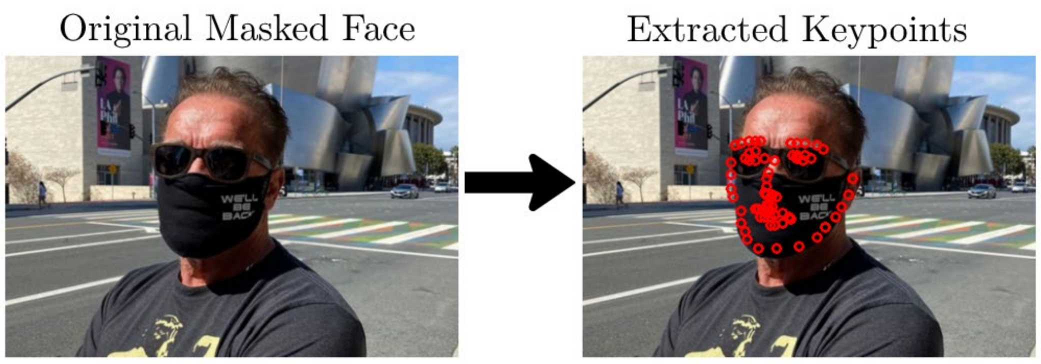


Fig. 20: Extracting facial landmarks using "face-alignment" python library

[1] Original dataset available at https://www.robots.ox.ac.uk/~vgg/data/vgg_face2/

b. Triangulation Process

Using the existing masks database (approx. 250 masks with different orientations and styles), masks are extracted from the generated features using the triangulation process (Fig. 21). The extracted masks are then saved into JSON format.

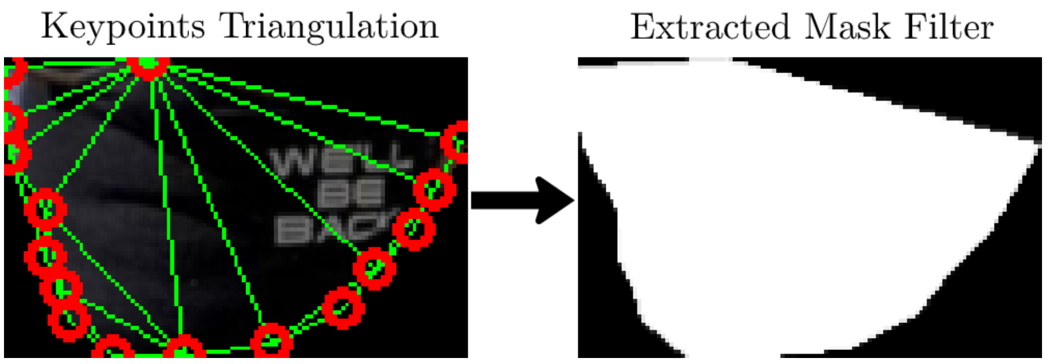


Fig. 21: Triangulation Process to extract face mask

c. Mask Matching (Augmentation)

Using the extracted masks, the VGGFace2 dataset is modified. The extracted masks are added to the faces available in the dataset by finding the facial landmarks and estimating the face pose for augmenting the mask (Fig. 22).

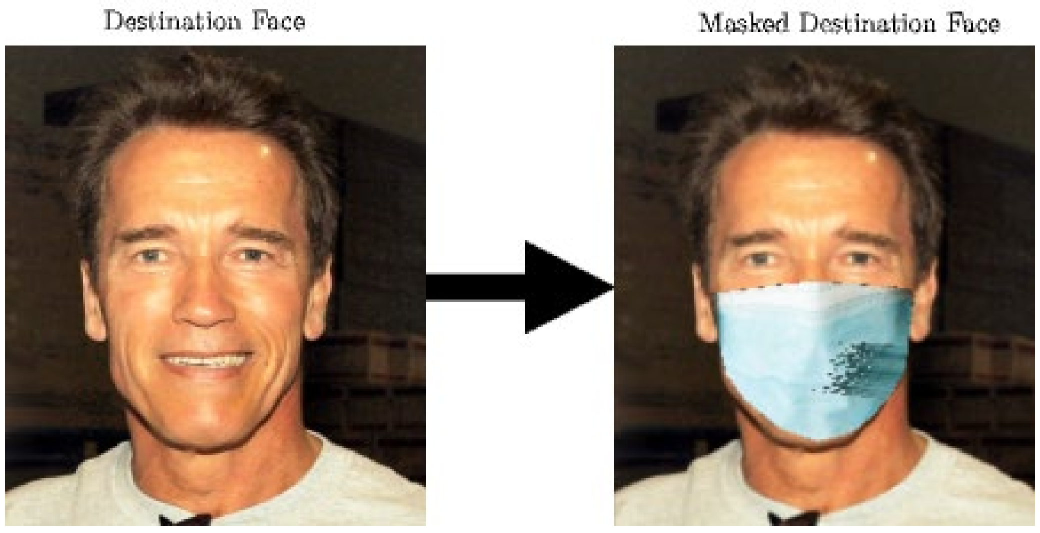


Fig. 22: Augmenting the extracted mask on a new face

d. Masked Face Cropping (Optimization)

The masked images are cropped to extract the eyes and forehead region by using the optimization (cropping) ratio of 0.7L [18] as shown in Fig. 23.

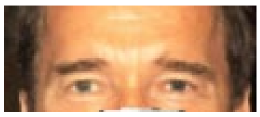


Fig. 23: Area of interest for deep learning models

Finally, the dataset consists of 22,647 belonging to 65 classes out of which 18,092 images belonging to the training dataset and 4,555 images belonging to the validation dataset.

5.2 Deep Learning Models under Observation

In this project, pre-trained deep learning models are used using transfer learning for masked face recognition as shown in Fig. 24.

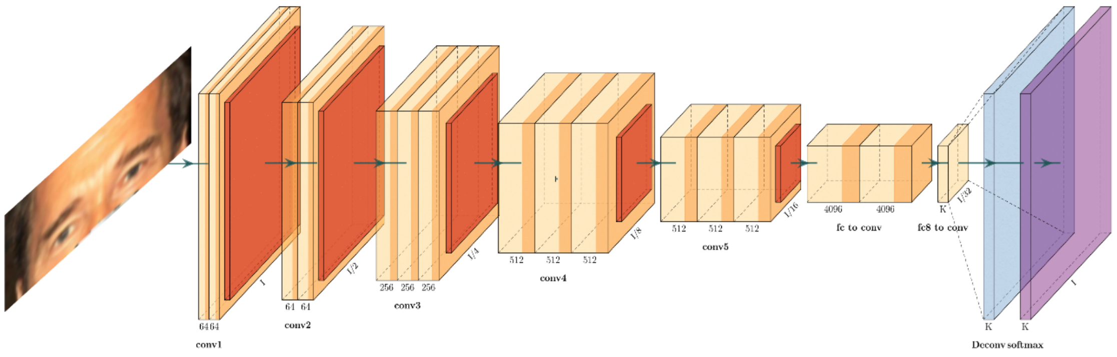


Fig. 24: Processes input to the deep neural network

Following is the list of pre-trained deep learning models under observation along with their description:

Table II: Pre-trained Models Description[2]

Model	**Size**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
VGG16	528 MB	0.713	0.901	138,357,544	23
InceptionV3	92 MB	0.779	0.937	23,851,784	159
Xception	88 MB	0.790	0.945	22,910,480	126

[2] Available at https://keras.io/api/applications/

InceptionResnetV2	215 MB	0.803	0.953	55,873,736	572
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet201	80 MB	0.773	0.936	20,242,984	201

The top-1 and top-5 accuracy signifies the ImageNet validation dataset performance of the model.

Depth represents the depth of the model which includes activation layers, batch normalization layers, etc.

Table III: Hyper-parameters and validation strategy for DL models

Validation Strategy	Hold-out validation in ratio 8:2 for training and testing
Pre-Training Dataset	ImageNet
Batch Size	32
Optimizer	Adam Optimizer
Loss Function	Categorical Cross Entropy
Maximum number of Training epochs	25
Early Stopping Patience	3
Early Stopping Criteria	Validation Accuracy

5.3 Metrics for performance

The following metrics are used for analyzing the performance of the models that are calculated using confusion matrix (as shown in Fig. 25) generated on the dataset:

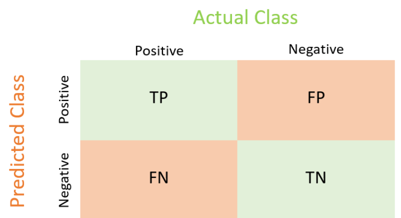


Fig. 25: Confusion Matrix

1. Accuracy

   Accuracy is the ratio of correct outcomes to the total outcomes of the experiment. Accuracy is calculated using the following formula:

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \tag{4.1}$$

   Accuracy represents the correctness of predictions made by the model.

2. Top k Categorical Accuracy

   It represents the percentage of records for which the target predictions are in the top K predictions.

   For a record:

   a. The predictions for an instance are ranked in the descending order of probability values.

   b. If the rank of the prediction is less than or equal to K for the actual class, it is considered accurate.

   Here for the deep learning models, k is chosen as 5.

3. AUC (Area Under ROC Curve)

   AUC represents the worthiness of model predictions (Fig. 26). It is the degree of how superior a model is capable to discern between positive and negative occurrences.

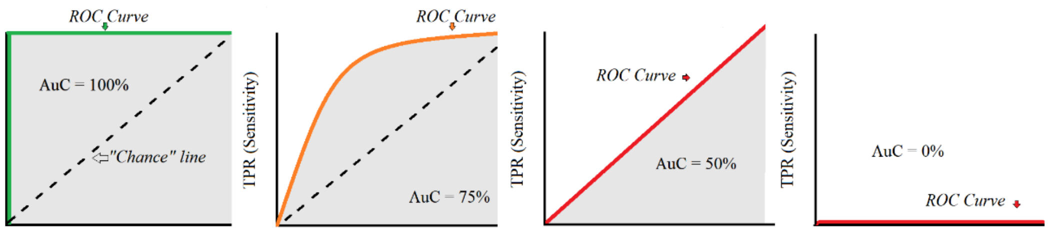


Fig. 26: AUC and ROC

   1.0 value in AUC means model prediction is 100% accurate and 0.5 means model prediction is worthless for unknown instances prediction.

4. Precision

   Precision is the ratio of correct positive outcomes to the total positive outcomes for a class. Precision is calculated using the following formula:

$$Precision = \frac{TP}{TP+FP} \tag{4.2}$$

Precision signifies how many positive outcomes are actually correct for a class.

5. Recall

Recall or sensitivity is defined as the ratio of correctly classified positive instances to the total number of actual positive instances. The recall is calculated using the following formula:

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

Recall signifies the positive predictions that are classified incorrectly.

The following figure (Fig. 27) represents the confusion matrix along with the metrics calculations:

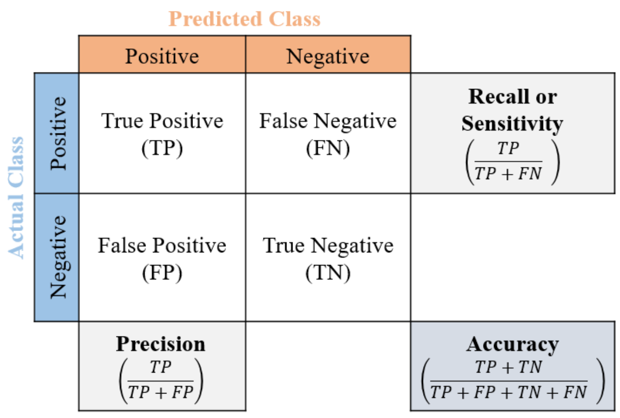

Predicted Class
Positive
Negative
Actual Class
Positive
Negative
True Positive (TP)
False Negative (FN)
False Positive (FP)
True Negative (TN)
Recall or Sensitivity $\left(\frac{TP}{TP+FN}\right)$
Precision $\left(\frac{TP}{TP+FP}\right)$
Accuracy $\left(\frac{TP+TN}{TP+FP+TN+FN}\right)$

Fig. 27: Confusion Matrix and different Metrics

CHAPTER 6

RESULTS AND DISCUSSION ON RESULTS

Q What is the performance of different deep learning models on masked faces?

The performance of different models is shown in the following table:

Table IV: Performance Metrics for different models

Model	**Accuracy**	**Top-5 Accuracy**	**AUC**	**Precision**	**Recall**
RGGNet (Proposed Model)	***0.608***	***0.841***	***0.953***	***0.777***	***0.519***
ResNet50V2	*0.5060*	*0.7737*	0.7704	0.5073	*0.5060*
VGG16	0.4648	0.7396	*0.9146*	*0.6940*	0.3495
InceptionV3	0.3574	0.6571	0.7767	0.3850	0.3460
Xception	0.3627	0.6428	0.7256	0.3713	0.3603
InceptionResnetV2	0.4360	0.7135	0.8009	0.4617	0.4250
MobileNetV2	0.4004	0.6777	0.7264	0.4029	0.3991
DenseNet201	0.4665	0.7396	0.7846	0.4768	0.4641

Q What is the highest accuracy achieved by different deep learning models on masked faces?

Out of all the pre-trained models under evaluation, ResNet50V2 achieved the highest accuracy (50.6%) as well as the highest top-5 accuracy (77.37%). Overall, RGGNet achieved the highest accuracy.

The accuracy plot for the different model is shown in the following figure:

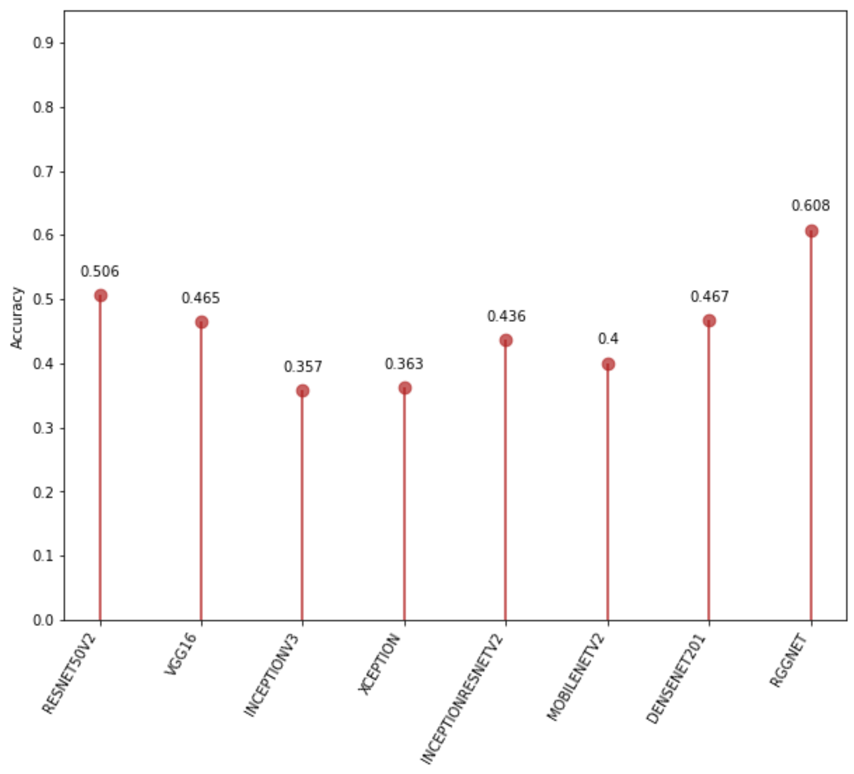


Fig. 28: Accuracy of different models

The top-5 accuracy plot for the different model is shown in the following figure:

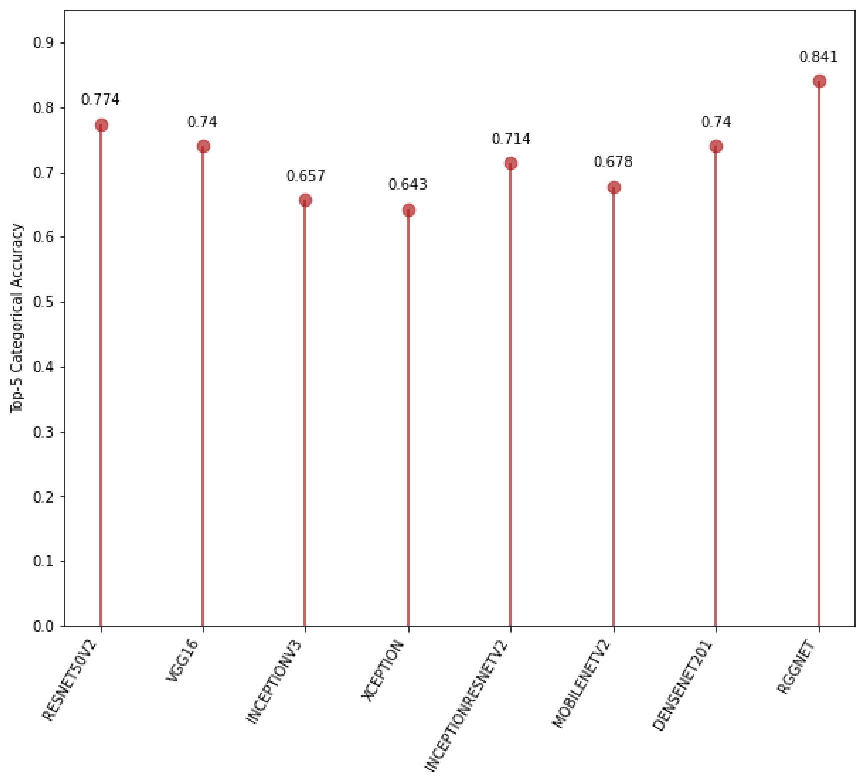


Fig. 29: Top-5 Accuracy of different models

The AUC plot for the different model is shown in the following figure:

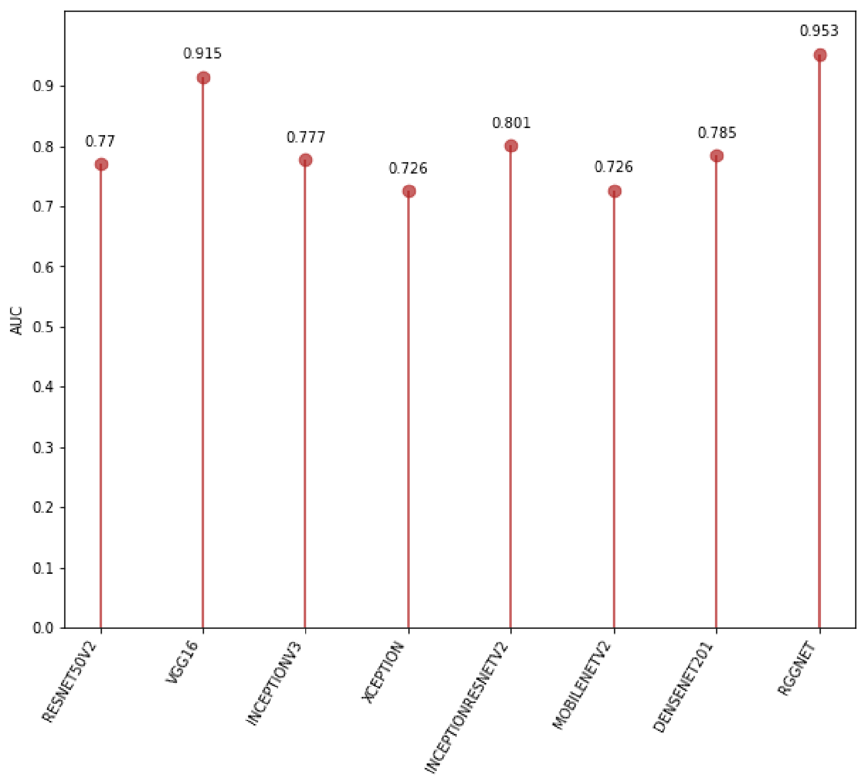


Fig. 30: AUC of different models

The precision plot for the different model is shown in the following figure:

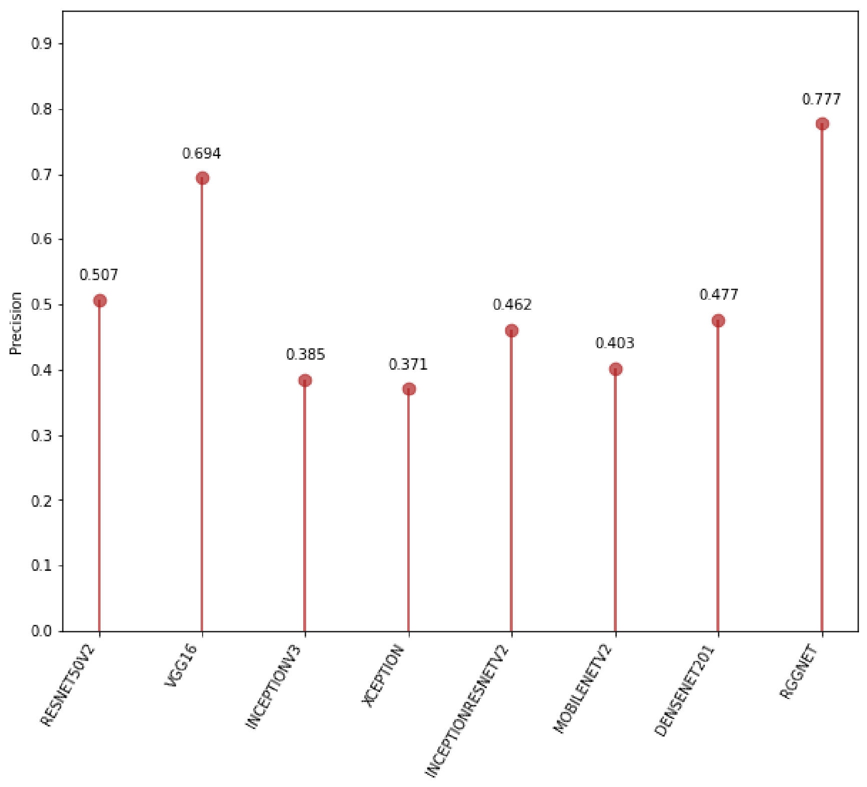


Fig. 31: Precision of different models

The recall plot for the different model is shown in the following figure:

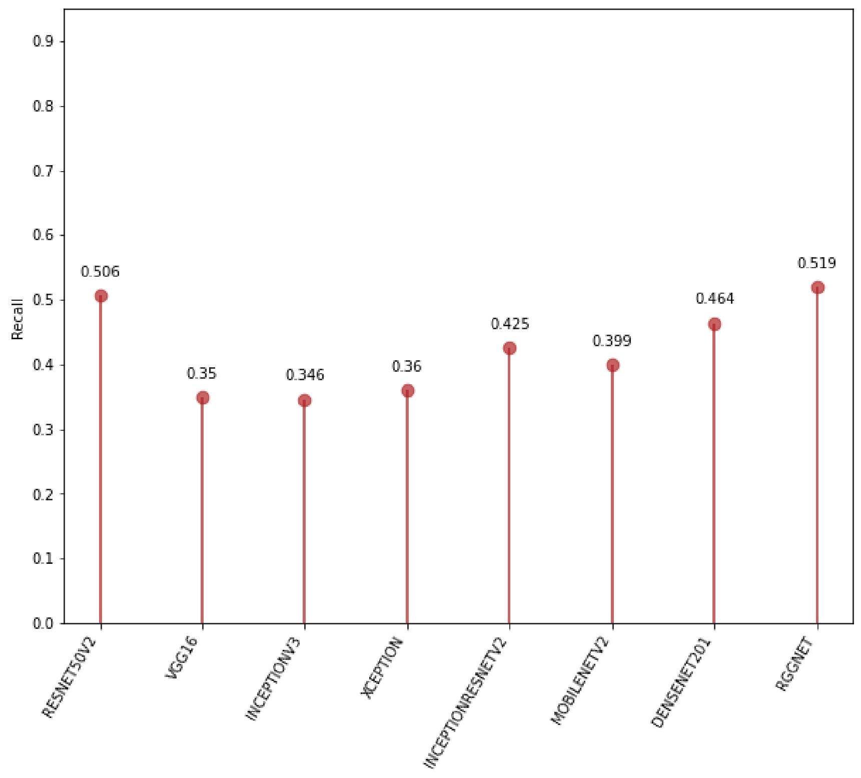


Fig. 32: Recall of different models

All these plots represent the variation of performance for different models.

Q What are observations made for different deep learning models on masked face recognition?

The following observations are made for pre-trained models:

- ResNet50V2 outperforms every other model in terms of accuracy and top-5 accuracy while also maintaining high precision and recall.
- In terms of precision, VGG16 has the highest value, i.e., VGG16 has shown a very low number of false predictions of positive instances. The AUC value of VGG16 is very high (>0.9), thus stating it is very much capable of differentiating between positive and negative instances or in this case, identify the person more easily from the crowd.
- DenseNet201 has shown accuracy and top-5 accuracy at the same level as VGG16 but it lacks in terms of performance on other performance metrics.
- ResNet50V2 is found to be more capable of masked face recognition compared to VGG16 when all the performance factors are taken into consideration as it is

performing highest in 3 out 5 performance criteria and 2nd highest in one of the remaining criteria.

The following observations are made for the proposed models:

- RggNet has been advantages of both the ResNet50V2 and the VGG16 model.
- RggNet outperforms every other model in every performance criteria.
- The top-5 accuracy of RggNet is more than 0.8 which means that the actual identity of the person under observation is in the top-5 predictions made by our model.
- In terms of precision, RggNet has a value greater than 0.75, i.e., RggNet has a very low false prediction rate over the positive instances.
- The AUC of 0.95 for RggNet throws the light that the prediction made by the model is not just some random prediction but the model has identified the features to predict the identity of the person.

Q Perform an investigation of whether the performance of the models differs significantly or not.

Statistical tests are used to determine whether the performance of the models differ significantly or not:

Applying Statistical test:

Step 1: Hypothesis formation

The null hypothesis (H_0) and alternative hypothesis (H_a) are as follows:

H_0: There is no statistical difference between the performance of the models

H_a: There is a statistically significant difference between the performance of the models.

Step 2: Selecting the statistical test

Since an evaluation of the difference between the performance of different methods is to be evaluated, the Friedman test is selected.

Step 3: Applying test on performance results and calculating p-value.

$$\chi 2 = \frac{12}{nk(k+1)} \sum_{i=1}^{k} R_i^2 - 3n(k+1) \quad (6.1)$$

where R_i is the individual rank of ith iteration, n is no of data instances and k is no. of groups

$$\text{Degree of Freedom } (DOF) = k - 1 \quad (6.2)$$

Table V: Computation of Ranks for Friedman Test

	ResNet 50V2	VGG16	InceptionV3	Xception	InceptionResnetV2	MobileNetV2	DenseNet201	RGGNet
Accuracy	2	4	8	7	5	6	3	1
Top-5 Accuracy	2	3.5	7	8	5	6	3.5	1
AUC	6	2	5	8	3	7	4	1
Precision	3	2	7	8	5	6	4	1
Recall	2	7	8	6	4	5	3	1
Rank total	15	18.5	35	37	22	30	17.5	5
Average Rank	3	3.7	7	7.4	4.4	6	3.5	1

On applying the Friedman test, Friedman's chi-square value (χ^2) is found to be 27.615752.

Degree of Freedom (DOF) = 7

Based on the calculated χ^2 value, the p-value is 0.000258.

Step 4: Defining the level of significance

As the calculated p-values <0.01, the results are significant at significance level (α) = 0.01.

Step 5: Deriving Conclusion

Since the calculated χ^2 value is greater than the tabulated value, the null hypothesis is rejected. Thus, it is concluded that the performance of the models differs significantly.

Applying a Post-adhoc test to determine whether there is a significant difference between RGGNet and other models:

Step 1: Hypothesis formation

The null hypothesis (*H_0*) and alternative hypothesis (*H_a*) are as follows:

H_{01}: The performance of the RGGNet model and ResNet50V2 model do not differ significantly.

H_{a1}: The performance of the RGGNet model and ResNet50V2 model differ significantly.

H_{02}: The performance of the RGGNet model and VGG16 model do not differ significantly.

H_{a2}: The performance of the RGGNet model and VGG16 model differ significantly.

H_{03}: The performance of the RGGNet model and InceptionV3 model do not differ significantly.

H_{a3}: The performance of the RGGNet model and InceptionV3 model differ significantly.

H_{04}: The performance of the RGGNet model and Xception model do not differ significantly.

H_{a4}: The performance of the RGGNet model and Xception model differ significantly.

H_{05}: The performance of the RGGNet model and InceptionResnetV2 model do not differ significantly.

H_{a5}: The performance of the RGGNet model and InceptionResnetV2 model differ significantly.

H_{06}: The performance of the RGGNet model and MobileNetV2 model do not differ significantly.

H_{a6}: The performance of the RGGNet model and MobileNetV2 model differ significantly.

H_{07}: The performance of the RGGNet model and DenseNet201 model do not differ significantly.

H_{a7}: The performance of the RGGNet model and DenseNet201 model differ significantly.

Step 2: Selecting the statistical test

Since the evaluation of the difference between the performance of different methods (with equal sample size) using the Friedman test leads to rejection of the null hypothesis and pairwise comparison is required, the Nemenyi test is selected.

Step 3: Applying test and calculating CD.

$$CD = q_{\alpha}\sqrt{\frac{k(k+1)}{6n}} \tag{6.3}$$

The value of q_{α} for eight subjects at $\alpha = 0.01$ is 3.526. The calculated CD value is found to be 5.462455711.

The following table is computed using average ranks calculated during the Friedman test:

Table VI: Computation of Differences for Nemenyi Test

Pair	Pair Difference	CD Difference
RGGNet – ResNet50V2	2	2 < 5.46
RGGNet – VGG16	2.7	2.7 < 5.46
RGGNet – InceptionV3	6	**6 > 5.46**
RGGNet – Xception	6.4	**6.4 > 5.46**
RGGNet – InceptionResnetV2	3.4	3.4 < 5.46
RGGNet – MobileNetV2	5	5 < 5.46
RGGNet – DenseNet201	2.5	2.5 < 5.46

Step 4: Defining the level of significance

The rank difference of pairs (RGGNet – InceptionV3 and RGGNet – Xception) is higher than the computed critical distance at α =0.01. The rank difference for other pairs is not significant at α =0.01.

Step 5: Deriving Conclusion

As the rank difference of pairs (RGGNet – InceptionV3 and RGGNet – Xception) is higher than the computed critical distance, it can be concluded that the RGGNet model significantly outperforms the InceptionV3 model and Xception model. The difference is not significant for the other techniques. All the null hypotheses except H_{03} and H_{04} cannot be rejected.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The following conclusion has been made:

- Existing pre-trained deep learning models have been studied and compared in this study.
- It has been found that ResNet50 has the most potential to solve the problem of masked faces compared to any other pre-trained deep learning model available.
- The proposed architecture that is derived from ResNet has shown better accuracy and other performance parameters compared to all the pre-trained models under observation.
- Along with this, the performance of different deep learning models has been studied.
- A baseline has been generated for the state-of-the-art pre-trained models to work upon.

The future scope of this project is as follows:

- Optimization of the existing deep learning models for performance improvement can be performed.
- The proposed model architecture can be enhanced further by the use of layer optimization in future iterations.
- Currently, transfer learning is utilized for faster training. Fine-tuning of layers can be performed, i.e., the further scope of fine-tuning can be studied for these models in the future.
- The results can act as a baseline for developing a better deep learning model for the masked face recognition problem without using the actual face knowledge.
- The proposed model architecture can be utilized in different domains of deep learning for further determining the capabilities of the model.

APPENDICES

Appendix 1: Training and Testing Results of Different Models

This appendix contains the training and testing results in detail that are reported in Chapter 5. The various models for which plots are included in this appendix are as follows:

- ResNet50V2 (Fig. 33)
- VGG16 (Fig. 34)
- InceptionV3 (Fig. 35)
- Xception (Fig. 36)
- InceptionResNetV2 (Fig. 37)
- MobileNetV2 (Fig. 38)
- DenseNet (Fig. 39)
- RGGNet (Fig. 40)

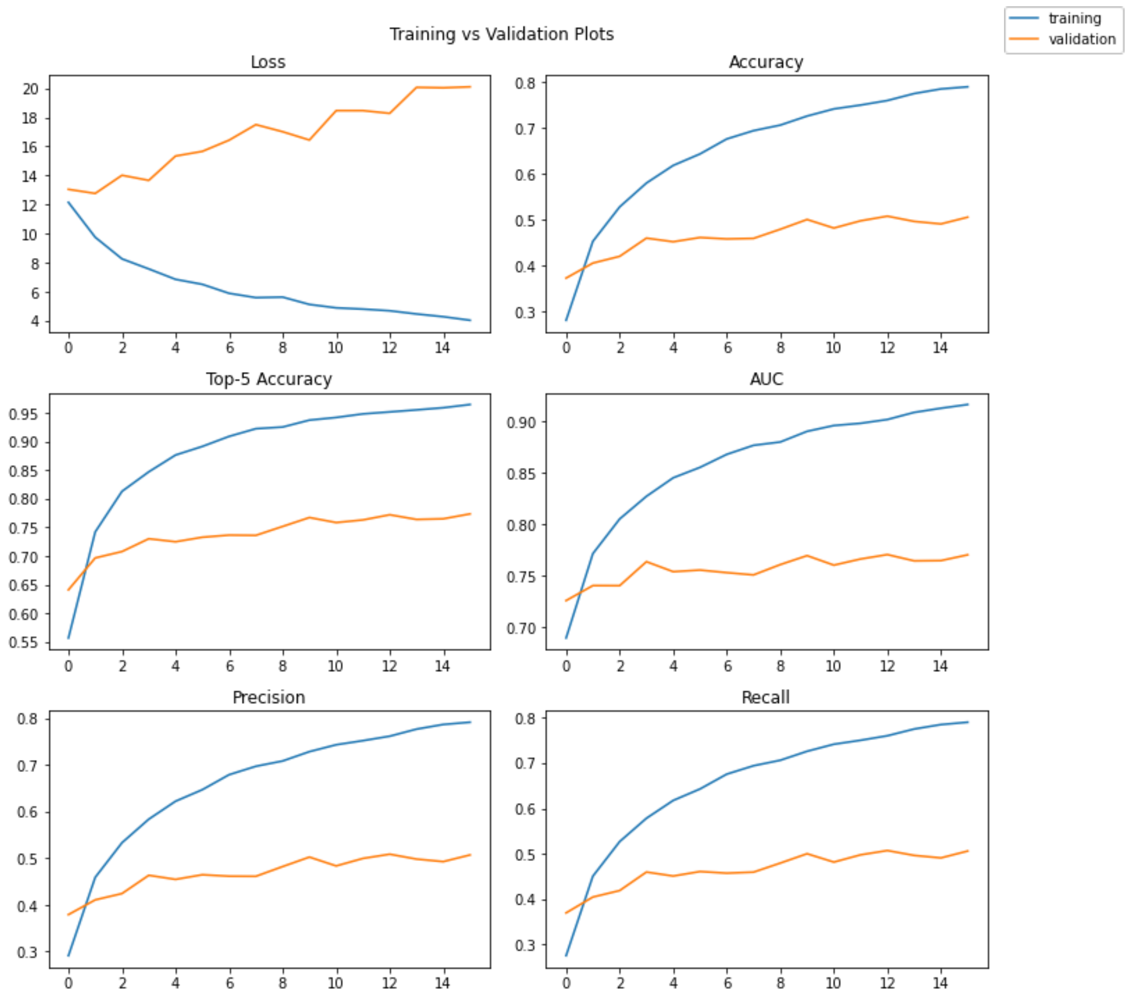


Fig. 33: ResNet50V2 Plots

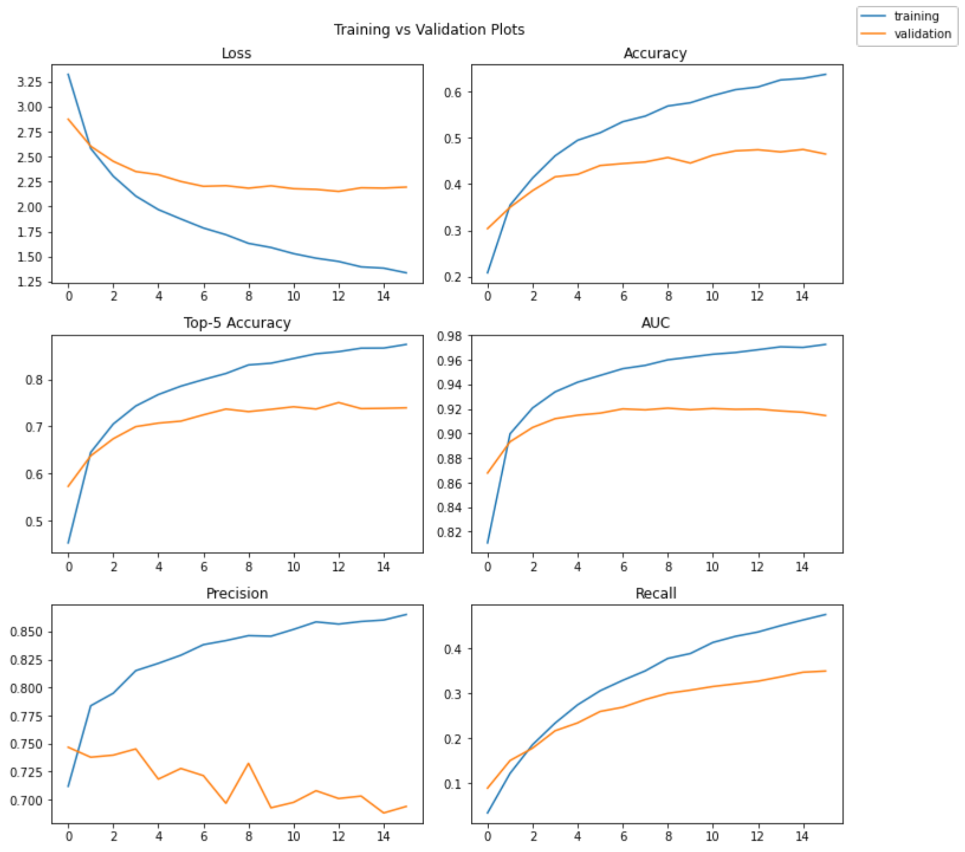


Fig. 34: VGG16 Plots

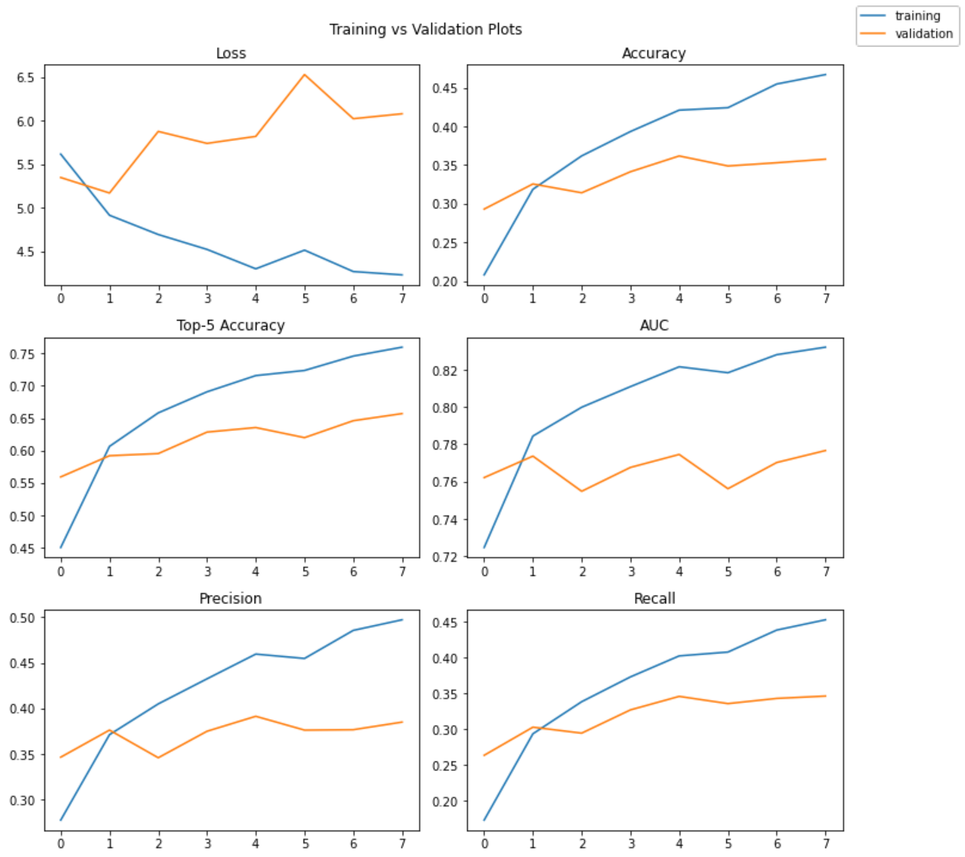


Fig. 35: InceptionV3 Plots

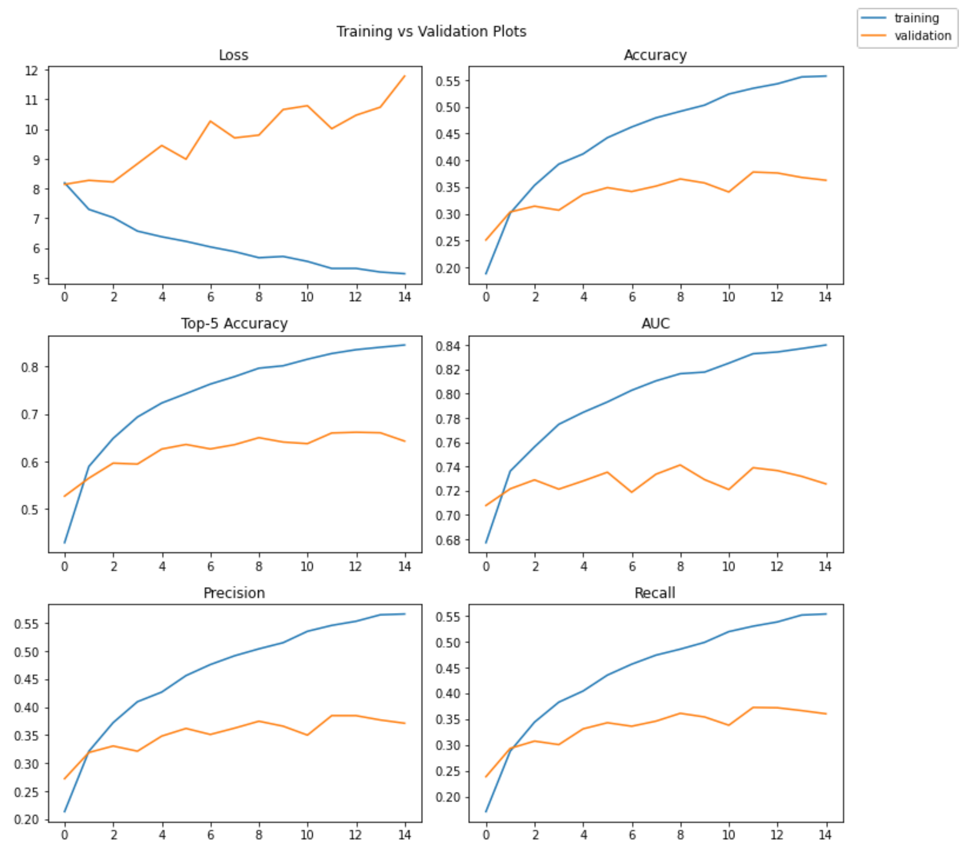


Fig. 36: Xception Plots

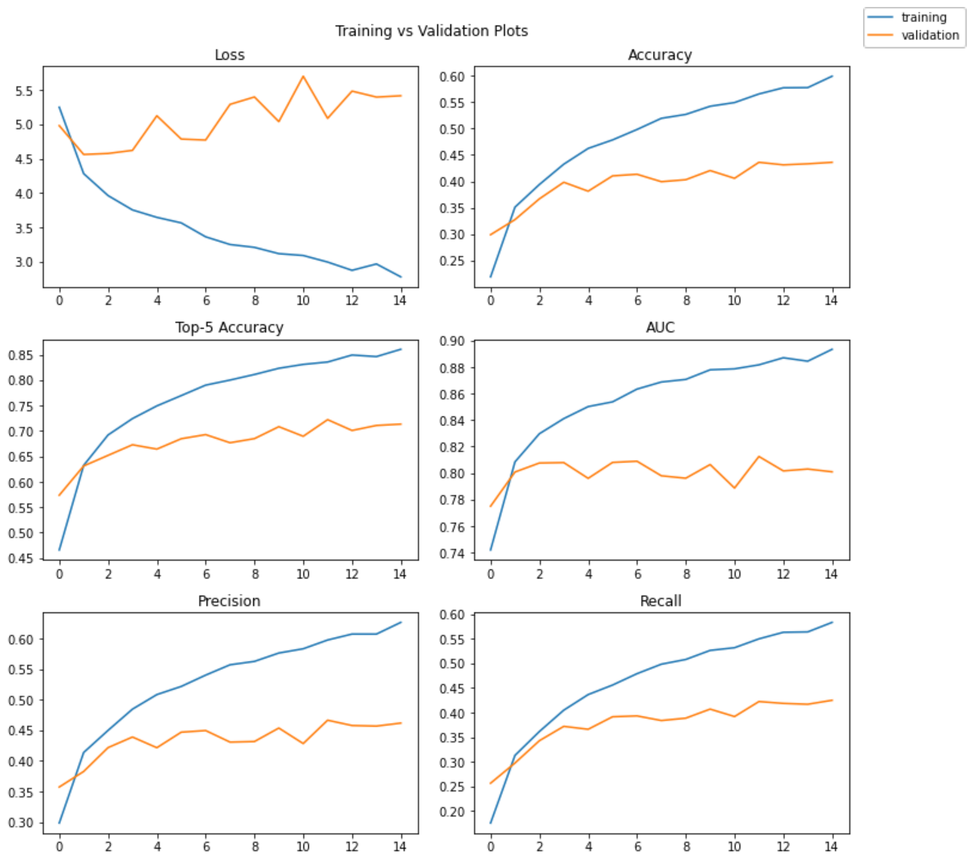


Fig. 37: InceptionResNetV2

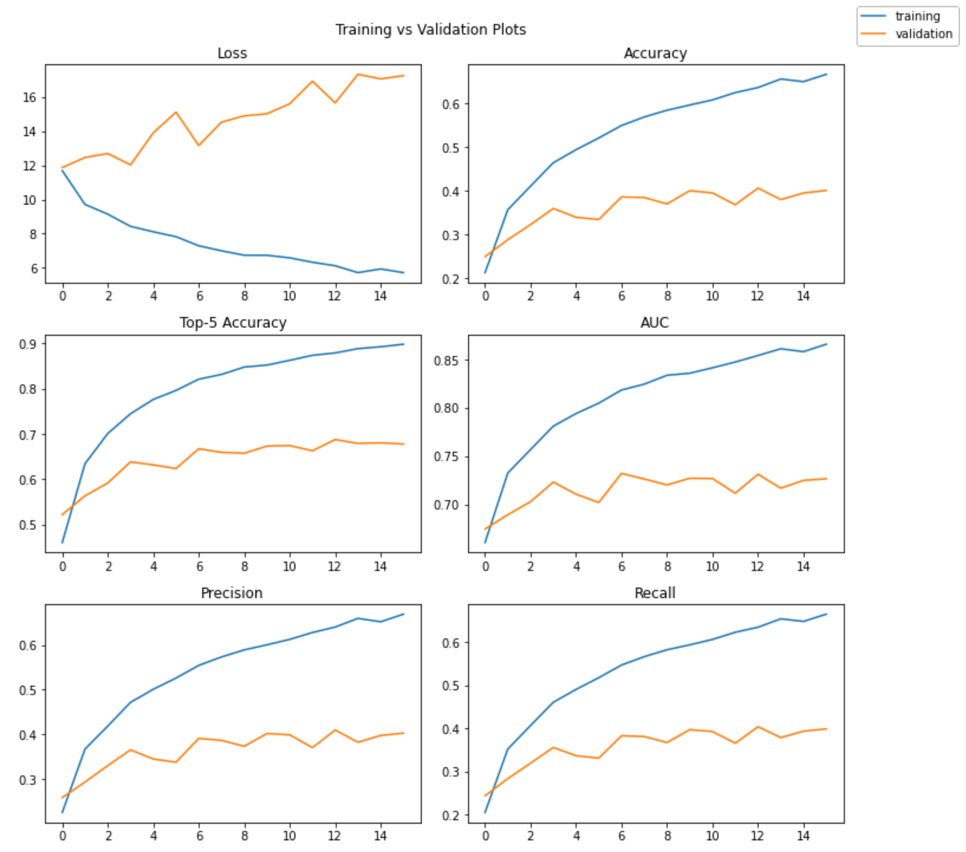


Fig. 38: MobileNetV2 Plots

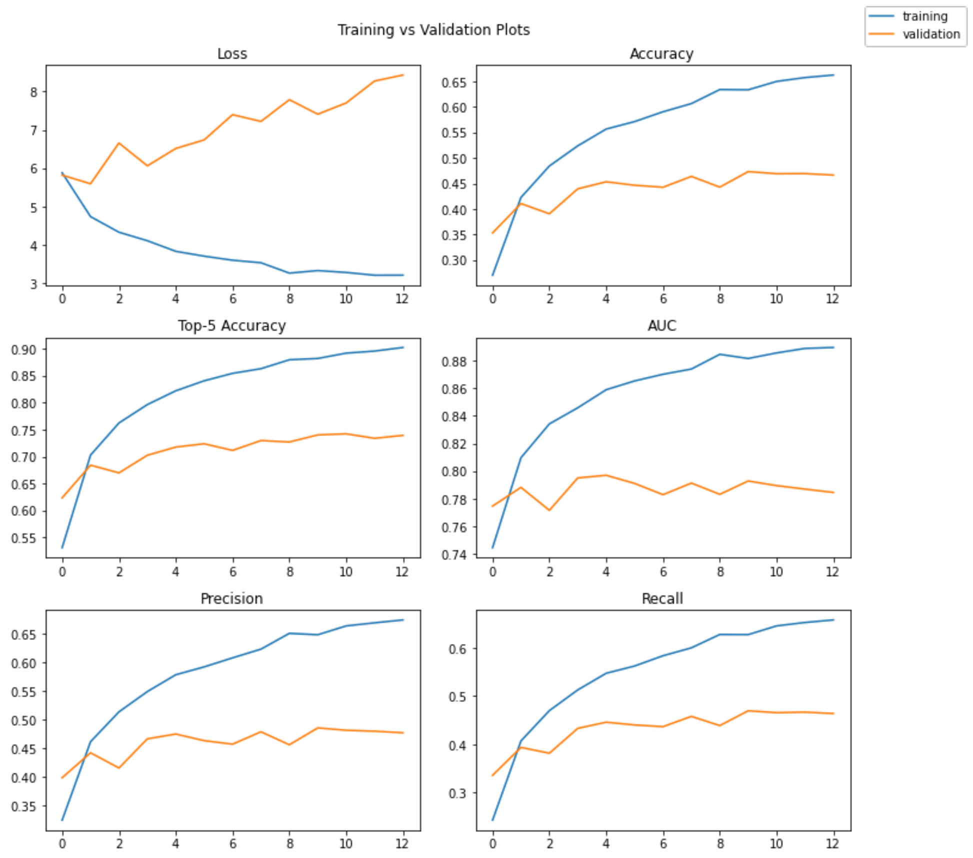


Fig. 39: DenseNet201 Plots

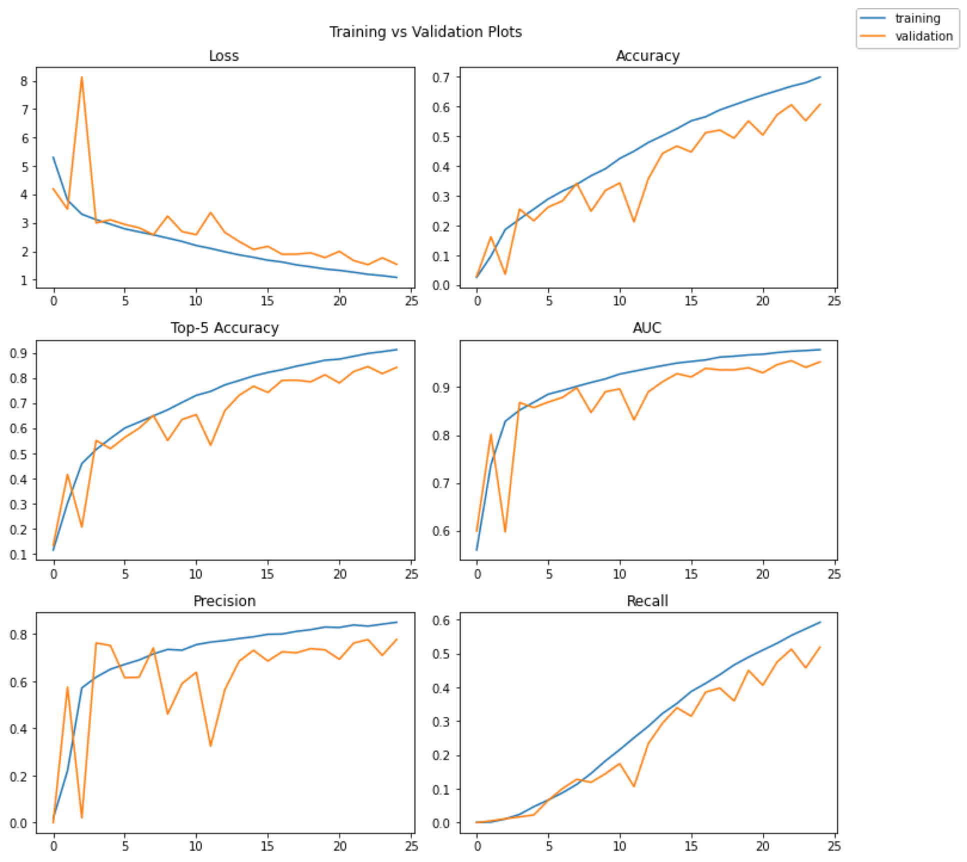


Fig. 40: RGGNet Plots

Appendix 2: Python Code

This appendix contains the python code divided into sections that can be used to replicate the project.

A 2.1 Importing Libraries

```
from glob import glob as gl
import keras as ks
from keras import metrics as met
from keras.preprocessing.image import ImageDataGenerator as IDG
from keras.callbacks import EarlyStopping as ES
from keras.applications.resnet_v2 import ResNet50V2 as RNETV2
from keras.applications.vgg16 import VGG16 as VGGNET
from keras.applications.inception_v3 import InceptionV3 as INCV3NET
from keras.applications.xception import Xception as XCPNET
from keras.applications.inception_resnet_v2 import InceptionResNetV2 as
INCRNETV2
from keras.applications.mobilenet_v2 import MobileNetV2 as MOBNETV2
from keras.applications.densenet import DenseNet201 as DENSENET
from keras.layers import Input as inp, Lambda as lbd, Dense as den,
Flatten as flt
from keras.models import Model as MdlObject
import pickle as pkl
```

A 2.2 Defining Constants

```
# re-size all the images to this
img_size = [224, 224]
bat_size = 32
mode = 'categorical'
#init constants
training_path = '/content/test_large'
validating_path = '/content/test_small'
EpochCount = 25
early_stop = ES(
    monitor='val_accuracy',
    patience=3,
    min_delta=0.001
)

preweights='imagenet'
permit = False
lossfunc = 'categorical_crossentropy'
optimfunc = 'adam'
actfunc = 'softmax'
```

A 2.3 Defining Training and Testing Dataset

```
#Defining training and testing dataset
train_gen = IDG(rescale = 1./255, shear_range = 0.2,
                zoom_range = 0.2, horizontal_flip = True)

test_gen = IDG(rescale = 1./255)

train_dataset = train_gen.flow_from_directory('/content/test_large',
                                                 target_size = img_size,
                                                 batch_size = bat_size,
                                                 class_mode = mode)

test_dataset = test_gen.flow_from_directory('/content/test_small',
                                            target_size = img_size,
                                            batch_size = bat_size,
                                            class_mode = mode)
```

A 2.4 Defining functions for RGGNet

```
from keras import layers
from keras.layers import Input as Inp, ZeroPadding2D as Zpd
from keras.layers import Conv2D as Con, MaxPool2D as MXPool
from keras.layers import BatchNormalization as BatNor, Activation as Act
from keras.layers import Add

def blockfunc(input_a, filterlen, kernels=3, stridesize=1,
           short_con=False, block_name=None):
    #Generate Residual Block
    axis_no = 3

    preactivation = BatNor(axis=axis_no, epsilon=1.001e-5,
                                  name=block_name +
'_preact_batchnormal')(input_a)
```

```
    preactivation = Act('relu', name=block_name +
'_preact_reluactivation')(preactivation)

    if short_con is True:
        shortcut_path = Con(4 * filterlen, 1, strides=stridesize,
                                 name=block_name + '_0con')(preactivation)
    else:
        shortcut_path = MXPool(1, strides=stridesize)(input_a) if
stridesize > 1 else Con(4 * filterlen, 1, strides=stridesize,
                                 name=block_name + '_0con')(input_a)

    input_a = Con(filterlen, 1, strides=1, use_bias=False,
                      name=block_name + '_1con')(preactivation)
    input_a = BatNor(axis=axis_no, epsilon=1.001e-5,
                                  name=block_name +
'_1batchnormal')(input_a)
    input_a = Act('relu', name=block_name + '_1reluactivation')(input_a)

    input_a = Zpd(padding=((1, 1), (1, 1)), name=block_name +
'_2padding')(input_a)
    input_a = Con(filterlen, kernels, strides=stridesize,
                      use_bias=False, name=block_name + '_2con')(input_a)
    input_a = BatNor(axis=axis_no, epsilon=1.001e-5,
                                  name=block_name +
'_2batchnormal')(input_a)
    input_a = Act('relu', name=block_name + '_2reluactivation')(input_a)

    input_a = Con(4 * filterlen, 1, name=block_name + '_3con')(input_a)
    input_a = layers.Add(name=block_name + '_output')([shortcut_path,
input_a])
    return input_a

def stacking_func(input_a, filterlen, blocklen, strfunc=2,
blocks_name=None):
    #Generate Stack of Repeated Blocks
    input_a = blockfunc(input_a, filterlen, short_con=True,
block_name=blocks_name + '_block1')
    for x in range(2, blocklen):
        input_a = blockfunc(input_a, filterlen, block_name=blocks_name +
'_block' + str(x))
    input_a = blockfunc(input_a, filterlen, stridesize=strfunc,
block_name=blocks_name + '_block' + str(blocklen))
    return input_a

def rggnet_module(layer_input_value,
                    use_bias = True,
                    weights = 'None',
                    topmost_layer_include = False,
                    pooltype=None,
                    **kwargs):

  layer_img_input = layer_input_value

  axis_no = 3

  input_a = Zpd(padding=((3, 3), (3, 3)),
name='con1_padding')(layer_img_input)
```

```
  input_a = Con(64, 7, strides=2, use_bias=use_bias,
name='con1_convolution')(input_a)

  input_a = Zpd(padding=((1, 1), (1, 1)),
name='MaxPooling1_padding')(input_a)
  input_a = MXPool(3, strides=2, name='MXPooling1_pooling')(input_a)

  input_a = stacking_func(input_a, 64, 3, blocks_name='con2')
  input_a = stacking_func(input_a, 128, 4, blocks_name='con3')
  input_a = stacking_func(input_a, 256, 6, blocks_name='con4')
  input_a = stacking_func(input_a, 512, 3, strfunc=1, blocks_name='con5')

  input_a = BatNor(axis=axis_no, epsilon=1.001e-5,
name='post_batchnormalizing')(input_a)
  input_a = Act('relu', name='post_reluactivation')(input_a)

  if topmost_layer_include:
    input_a =
layers.GlobalAveragePooling2D(name='average_pooling')(input_a)
    input_a = layers.Dense(1000, activation='softmax',
name='probs')(input_a)
  else:
    if pooltype == 'max':
      input_a = layers.GlobalMaxPooling2D(name='maximum_pooling')(input_a)
    elif pooltype == 'avg':
      input_a =
layers.GlobalAveragePooling2D(name='average_pooling')(input_a)

  input_layer = layer_img_input

  # Generate model object
  modelobj = MdlObject(input_layer,input_a, name='My_RNet')

  return modelobj
```

A 2.5 Defining, Training, Testing and Saving Various Models

A 2.5.1 ResNet50V2 Model

```
# Preprocess layer added to the face of ResNet50V2
rnet = RNETV2(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
for fold in rnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(rnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)
```

```
#Defining Resnet50V2 model for 25 epochs
rnetmod = MdlObject(inputs=rnet.input, outputs=inferlayer)

# Summary of the model
rnetmod.summary()

# Defining loss and optimization function
rnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
           met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
RNetFit = rnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)

model_namevalue='ResNet50V2__25epoch_model.h5'
model.save(model_namevalue)

model_history='ResNet50V2_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(RNetFit.history, file_pi)
```

A 2.5.2 VGG16 Model

```
# Preprocess layer added to the face of VGG16
vggnet = VGGNET(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
for fold in vggnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(vggnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining VGG16 model for 25 epochs
vggnetmod = MdlObject(inputs=vggnet.input, outputs=inferlayer)

# Summary of the model
vggnetmod.summary()

# Defining loss and optimization function
vggnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
           met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
VGGNetFit = vggnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)

model_namevalue='VGG16_25epoch_model.h5'
model.save(model_namevalue)

model_history='VGG16_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(VGGNetFit.history, file_pi)
```

A 2.5.3 InceptionV3 Model

```
# Preprocess layer added to the face of InceptionV3
incnet = INCV3NET(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
for fold in incnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(incnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining InceptionV3 model for 25 epochs
incnetmod = MdlObject(inputs=incnet.input, outputs=inferlayer)

# Summary of the model
incnetmod.summary()

# Defining loss and optimization function
incnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
           met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
IncNetFit = incnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)

model_namevalue='InceptionV3_25epoch_model.h5'
model.save(model_namevalue)

model_history='InceptionV3_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(IncNetFit.history, file_pi)
```

A 2.5.4 Xception Model

```
# Preprocess layer added to the face of Xception
xpnet = XCPNET(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
for fold in xpnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(xpnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining Xception model for 25 epochs
xpnetmod = MdlObject(inputs=xpnet.input, outputs=inferlayer)

# Summary of the model
xpnetmod.summary()

# Defining loss and optimization function
xpnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
          met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
XpNetFit = xpnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)
```

```
model_namevalue='Xception_25epoch_model.h5'
model.save(model_namevalue)

model_history='Xception_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(XpNetFit.history, file_pi)
```

A 2.5.5 InceptionResNetV2 Model

```
# Preprocess layer added to the face of InceptionResNetV2
incrnet = INCRNETV2(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
for fold in incrnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(incrnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining InceptionResNetV2 model for 25 epochs
incrnetmod = MdlObject(inputs=incrnet.input, outputs=inferlayer)

# Summary of the model
incrnetmod.summary()

# Defining loss and optimization function
incrnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
           met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
IncRNetFit = incrnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)

model_namevalue='InceptionResNetV2_25epoch_model.h5'
model.save(model_namevalue)

model_history='InceptionResNetV2_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(IncRNetFit.history, file_pi)
```

A 2.5.6 MobileNetV2 Model

```
# Preprocess layer added to the face of MobileNetV2
mnet = MOBNETV2(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
for fold in mnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(mnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining MobileNetV2 model for 25 epochs
mnetmod = MdlObject(inputs=mnet.input, outputs=inferlayer)

# Summary of the model
mnetmod.summary()

# Defining loss and optimization function
mnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
          met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
MNetFit = mnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)

model_namevalue='MobileNetV2_25epoch_model.h5'
model.save(model_namevalue)

model_history='MobileNetV2_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(MNetFit.history, file_pi)
```

A 2.5.7 DenseNet201 Model

```
# Preprocess layer added to the face of DenseNet201
dnet = DENSENET(input_shape=img_size + [3],
              include_top=permit,
              weights=preweights)

# To keep existing weights
```

```
for fold in dnet.layers:
  fold.trainable = permit

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(dnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining DenseNet201 model for 25 epochs
dnetmod = MdlObject(inputs=dnet.input, outputs=inferlayer)

# Summary of the model
dnetmod.summary()

# Defining loss and optimization function
dnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
           met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
DNetFit = dnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset),
  callbacks=[early_stop]
)

model_namevalue='DenseNet201_25epoch_model.h5'
model.save(model_namevalue)

model_history='DenseNet201_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(DNetFit.history, file_pi)
```

A 2.5.8 RGGNet Model

```
# Generating RGGNet Model Object

visible = Input(shape=(224, 224, 3))
# Adding RGGNet model module
rggnet = rggnet_module(visible, 64)

# To find number of classes in the dataset
imgfolders = gl('/content/test_large/*')

#Custom Layers
a = flt()(rggnet.output)
inferlayer = den(len(imgfolders), activation=actfunc)(a)

#Defining RGGNet model for 25 epochs
```

```
rggnetmod = MdlObject(inputs=rggnet.input, outputs=inferlayer)

# Summary of the model
rggnetmod.summary()

# Defining loss and optimization function
rggnetmod.compile(
  loss=lossfunc,
  optimizer=optimfunc,
  metrics=['accuracy','top_k_categorical_accuracy',
           met.AUC(), met.Precision(), met.Recall()]
)

#Performing training and testing for 25 epoches
RGGNetFit = rggnetmod.fit_generator(
  train_dataset,
  validation_data=test_dataset,
  epochs=EpochCount,
  steps_per_epoch=len(train_dataset),
  validation_steps=len(test_dataset)
)

model_namevalue='RGGNet_25epoch_model.h5'
model.save(model_namevalue)

model_history='RGGNet_25epoch_history.h5'
with open(model_history, 'wb') as file_pi:
        pkl.dump(RGGNetFit.history, file_pi)
```

REFERENCES

[1] Neo, H. F., Teo, C. C., & Teoh, A. B. J. (2010). *Development of Partial Face Recognition Framework. 2010 Seventh International Conference on Computer Graphics, Imaging and Visualization.* doi:10.1109/cgiv.2010.29

[2] Su, Y., Yang, Y., Guo, Z., & Yang, W. (2015). *Face recognition with occlusion. 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR).* doi:10.1109/acpr.2015.7486587

[3] He, E. J., Fernandez, J. A., Kumar, B. V. K. V., & Alkanhal, M. (2016). *Masked correlation filters for partially occluded face recognition. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* doi:10.1109/icassp.2016.7471885

[4] Lin, S., Cai, L., Lin, X., & Ji, R. (2016). *Masked face detection via a modified LeNet. Neurocomputing, 218, 197–202.* doi:10.1016/j.neucom.2016.08.056

[5] Wang, M., Wang, Z., & Li, J. (2017). *Deep convolutional neural network applies to face recognition in small and medium databases. 2017 4th International Conference on Systems and Informatics (ICSAI).* doi:10.1109/icsai.2017.8248499

[6] Coskun, M., Ucar, A., Yildirim, O., & Demir, Y. (2017). *Face recognition based on convolutional neural network. 2017 International Conference on Modern Electrical and Energy Systems (MEES).* doi:10.1109/mees.2017.8248937

[7] Wan, W., & Chen, J. (2017). *Occlusion robust face recognition based on mask learning. 2017 IEEE International Conference on Image Processing (ICIP).* doi:10.1109/icip.2017.8296992

[8] Guo, G., & Zhang, N. (2018). *What Is the Challenge for Deep Learning in Unconstrained Face Recognition? 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018).* doi:10.1109/fg.2018.00070

[9] Elmahmudi, A., & Ugail, H. (2018). *Experiments on Deep Face Recognition Using Partial Faces. 2018 International Conference on Cyberworlds (CW).* doi:10.1109/cw.2018.00071

[10] Qu, X., Wei, T., Peng, C., & Du, P. (2018). *A Fast Face Recognition System Based on Deep Learning. 2018 11th International Symposium on Computational Intelligence and Design (ISCID).* doi:10.1109/iscid.2018.00072

[11] Tomodan, E. R., & Caleanu, C. D. (2018). *Bag of Features vs Deep Neural Networks for Face Recognition. 2018 International Symposium on Electronics and Telecommunications (ISETC).* doi:10.1109/isetc.2018.8583846

[12] Wu, G., Tao, J., & Xu, X. (2019). *Occluded Face Recognition Based on the Deep Learning. 2019 Chinese Control And Decision Conference (CCDC).* doi:10.1109/ccdc.2019.8832330

[13] Ejaz, M. S., Islam, M. R., Sifatullah, M., & Sarker, A. (2019). *Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition. 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT).* doi:10.1109/icasert.2019.8934543

[14] Khan, S., Ahmed, E., Javed, M. H., A Shah, S. A., & Ali, S. U. (2019). *Transfer Learning of a Neural Network Using Deep Learning to Perform Face Recognition. 2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE).* doi:10.1109/icecce47252.2019.8940754

[15] Ejaz, M. S., & Islam, M. R. (2019). *Masked Face Recognition Using Convolutional Neural Network. 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*. doi:10.1109/sti47673.2019.9068044

[16] Bhuiyan, M. R., Khushbu, S. A., & Islam, M. S. (2020). *A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT).* doi:10.1109/icccnt49239.2020.9225384

[17] Adjabi, I., Ouahabi, A., Benzaoui, A., Taleb-Ahmed, A. (2020). *Past, Present, and Future of Face Recognition: A Review. Electronics 2020, 9, 1188.* doi:10.3390/electronics9081188

[18] Adnane Cabani, Karim Hammoudi, Halim Benhabiles, and Mahmoud Melkemi (2021). *MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19. Smart Health, Volume 19, 2021.* doi:10.1016/j.smhl.2020.100144

[19] Li, Y., Guo, K., Lu, Y., and Lui L. (2021). *Cropping and attention-based approach for masked face recognition, Appl Intell, 2021.* doi:10.1007/s10489-020-02100-9

[20] Li, S. and Jain, A., 2011. "*Handbook of Face Recognition*". 2nd ed. London: Springer-Verlag London Limited, p.1.