

DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING
KUDLU GATE, BANGALORE – 560068



**Bachelor of Technology
In**

COMPUTER SCIENCE AND ENGINEERING

Major Project Phase-II Report

Autism Spectrum Disorder

By

Deepthi S-ENG20CS0088

Likitha H M-ENG20CS0174

Monisha S-ENG20CS0210

Meghana G N-ENG20CS0195

Batch No: 15

Under the supervision of

Dr. Pramod Kumar Naik

Associate professor, Department of Computer Science and Engineering

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY
(2023-2024)**



DAYANANDA SAGAR UNIVERSITY

School of Engineering

Department of Computer Science & Engineering

Kudlu Gate, Bangalore –560068
Karnataka, India

CERTIFICATE

This is to certify that the Major Project Stage-2 work titled “Autism Spectrum Disorder” is carried out by **DEEPTHI S (ENG20CS0088), LIKHITHA H M (ENG20CS0174), MEGHANA G N (ENG20CS0195), MONISHA S (ENG20CS0210)** bonified students of seventh semester of bachelor of technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfilment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year 2023-2024.

Dr. Pramod Kumar Naik

Associate Professor
Dept. of CS&E,
School of Engineering
Dayananda Sagar University

Date:

Dr. Girisha G S

Chairman CSE
School of Engineering
Dayananda Sagar University

Date:

**Dr. Uday Kumar Reddy
K R**

Dean
School of Engineering
Dayananda Sagar University

Date:

Name of the Examiner

- 1.
- 2.

Signature of Examiner

DECLARATION

We, **DEEPTHI S (ENG20CS0088), LIKHITHA H M (ENG20CS0174), MEGHANA G N (ENG20CS0195), MONISHA S (ENG20CS0210)** are students of the eight semester BTech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Major Project titled “**AUSTISM SPECTRUM DISORDER DETECTION USING ML**” has been carried out by us and submitted in partial fulfillment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2023-2024**.

Student	Signature
Deepthi S ENG20CS0088	
Likitha H M ENG20CS0174	
Meghana G N ENG20CS0195	
Monisha S ENG20CS0210	

Place: Bangalore

Date:

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

*We would like to thank **Dr. Udaya Kumar Reddy K R, Dean, School of Engineering & Technology, Dayananda Sagar University, Bangalore** for his constant encouragement and expert advice.*

*It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Department Chairman, Computer Science and Engineering, Dayananda Sagar University, Bangalore** for providing right academic guidance that made our task possible.*

*We would like to thank our guide **Dr. Pramod Kumar Naik, Associate Professor, Dept. of Computer Science and Engineering, Dayananda Sagar University, Bangalore** for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

*We would like to thank our Project Coordinator **Dr. Meenakshi Malhotra and Prof. Mohammed Khurram J** as well as all the staff members of Computer Science and Engineering for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the Project work.

Signature of students

USN: ENG20CS0088

Name: Deepthi S

USN: ENG20CS0195

Name: Meghana G N

USN: ENG20CS0174

Name: Likitha H M

USN: ENG20CS0210

Name: Monisha S

TABLE OF CONTENTS

	Page
List of ABBREVIATIONS.....	vii
LIST OF FIGURES.....	viii
LIST OF TABLES.....	ix
ABSTRACT.....	x
 CHAPTER 1 INTRODUCTION.....	 1
1.1. INTRODUCTION.....	2
1.2. SCOPE.....	3
1.3 DESIGN WORKFLOW.....	3
1.4 ORGANISATION OF REPORT.....	4
CHAPTER 2 PROBLEM DEFINITION	5
2.1 PROBLEM DEFINITION.....	6
2.2 OBJECTIVE.....	7
CHAPTER 3 LITERATURE SURVEY.....	8
CHAPTER 4 PROJECT DESCRIPTION.....	14
4.1. AUTISM CLASSIFICATION ARCHITECTURE.....	15
4.2. ASSUMPTIONS AND DEPENDENCIES.....	32
CHAPTER 5 REQUIREMENTS	33
5.1. FUNCTIONAL REQUIREMENTS.....	34
5.2. NON FUNCTIONAL REQUIREMENTS.....	34
5.3 HARDWARE AND SOFTWARE REQUIREMENTS.....	35
CHAPTER 6 METHODOLOGY	36
CHAPTER 7 EXPERIMENTATION.....	40
7.1 WORKING.....	41
CHAPTER 8 RESULTS AND DISCUSSION.....	57
8.1 RESULTS.....	58

CHAPTER 9 CONCLUSION AND FUTURE WORK.....	61
9.1 CONCLUSION.....	62
9.2 SCOPE FOR FUTURE WORK.....	63
REFERENCES... ..	64

LIST OF ABBREVIATIONS

ASD	Autism Spectrum Disorder
AI	Artificial Intelligence
ML	Machine learning
DP	Deep Learning
AUC	Area Under Curve
TL	Transfer Learning
CNN	Cconvolutional Neural Network
IGAE	Info Gain Attribute Evaluator
GRAE	Gain Ratio Evaluate Attribute
CAE	Attribute evaluator

LIST OF FIGURES

Fig. No.	Description of the Figure	Page No.
1	Design work flow	4
4	Block diagram of ASD	15
4.1	a) ResNet architecture b)ResNet modules	17
4.2	AlexNet Architecture	19
4.3	DenseNet Architecture	21
4.4	InceptionNet Architecture	23
4.5	EfficientNet Architecture	24
4.6	XceptionNet Work Flow	26
4.7	VGG 16 Layers	27
4.8	NaasNet workflow	30
6.1	Design diagram	37
6.2	Process flow chart and diagram	38
7.1	Dataset	42
7.2	Resnet epoch training	44
7.3	Alexnet epoch training	45
7.4	Densenet epoch training	45
7.5	NasNet epoch training	46
7.6	Confusion Matrix for Nasnet	46
7.7	Xceptionnet	47
7.8	Confusion Matrix for XceptionNet	47
7.9	Efficientnet epoch training	48
7.10	Inceptionnet epoch training	49
7.11	Training and validation loss	49
7.12	Training and validation accuracy	50

7.13	VGG 16 epoch training	50
7.14	Training and validation accuracy	52
7.15	Training and validation loss	53
7.16	Confusion matrix for VGG16	53
7.17	Website	53
8.1	Prediction of autistic or non-autistic	58
8.2	Website Photo capture	60
8.3	Website Image upload	60

LIST OF TABLES

Table No.	Description of the table	Page.NO
3	Comparison between different research papers	10
7.1	Pre trained parameter summary	41
7.2	Validation loss and accuracy	51
7.3	Prediction of autistic or non-autistic	52
7.4	Comparison table of different CNN model	54
8.1	Comparison of eight different models	58

ABSTRACT

Autism Spectrum Disorder (ASD) diagnosis poses significant challenges due to its diverse presentation. ASD is a neurodevelopmental condition characterized by challenges in social interaction, communication, and repetitive behaviors. Previous studies underscore the different features on which a model can depend on for Prediction under various circumstances. Our work seeks to advance the detection methodologies on focusing on certain features and explore the real-time capabilities in this domain. This project explores a machine learning and deep learning driven approach using transfer learning in conjunction with the VGG-16, DenseNet model, NASNet, Alexnet, Inception net, Exception net, Resnet, Efficient net convolutional neural network architecture for ASD detection through image dataset.

The study focuses on leveraging a pre-trained VGG-16 model, DenseNet model, NASNet, Alexnet, Inception net, Exception net, Resnet, Efficient net CNN models, initially trained on a diverse image dataset and fine-tuning it to recognize ASD-related patterns from images. Utilizing transfer learning, the network's early layers capture general features, while deeper layers adapt to learn ASD-specific patterns from the image dataset. In parallel, a feature dataset is constructed, extracting pertinent image features using techniques like convolutional neural network activations and texture analysis.

All these pre-trained models were trained separately for the considered Image dataset to have complete understanding of the texture and expression in children so that when the models encounter new images, they could be able to make accurate prediction. The performance of all these eight models were recorded. Comparison based on the performances of all the eight models were made and then tabulated.

The tabulation represents the number of layers and parameters considered for experimentation and the models' performances. Each of these models' performance is measured through validation accuracy which shows that NasNet having 50%, Xception Net being 81%, AlexNet provides 77%, ResNet having accuracy of 53%, Vgg16 being 75%, Inception Net yielded 79%, DenseNet provided 78% and finally EfficientNet being 80% of accuracies. Off the best performed models Xception Net with 81% validation accuracy and VGG 16 with 81% Training accuracy were integrated along with NasNet which is taken as backend for the web development.

Web development is done using HTML, CSS for styling and Javascript for handling various events within the web page. Flask is used as Backend that acts as bridge for connecting the web application with machine learning and deep learning models. The web application enables any individual to upload or capture picture that on submitting gives the output predicted by the model as Autistic or Non-Autistic. If an image is predicted as autistic then some common treatment methods are advised along with a caution to have a further diagnosis.

The ML model's performance is rigorously evaluated using cross-validation techniques, assessing accuracy, precision, recall, and F1-score metrics to validate its efficacy in ASD detection. With an Accuracy range of 80% to 90% the models proved to be efficient in Autism Detection. This project aims to contribute to the advancement of ASD detection methodologies, providing a foundation for accurate, efficient, and potentially automated diagnosis, ultimately facilitating timely interventions and support for individuals on the autism spectrum.

CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION:

Autism, a complex neuro-developmental disorder, presents considerable challenges to individuals' social growth and development, affecting both children and adults. Despite ongoing research, a definitive cure remains elusive, highlighting the critical importance of early diagnosis for effective treatment interventions. Traditionally, diagnosing autism spectrum disorder (ASD) relies heavily on behavioral observations conducted in clinical settings, a process often time-consuming and prone to subjectivity. While ASD is typically identified around age 2, diagnosis may occur later depending on the complexity of symptoms, potentially delaying crucial interventions.

The etiology of ASD is multifaceted, involving a combination of genetic predispositions and environmental influences. These factors intricately affect not only the nervous system but also social and cognitive skills, resulting in a diverse range of symptoms that vary in intensity among affected individuals. Common manifestations include communication difficulties, obsessional interests, and repetitive behaviors, underscoring the heterogeneous nature of the disorder.

Accurate detection of ASD necessitates thorough evaluations by healthcare professionals and psychologists, involving comprehensive assessments of behavioral, developmental, and medical history factors. Early intervention strategies, tailored to individual needs, have shown promise in alleviating symptoms and improving overall quality of life. However, delays in diagnosis remain a prevalent challenge, emphasizing the urgent need for innovative approaches to expedite the assessment process.

Machine learning (ML) emerges as a promising avenue for enhancing ASD detection and diagnosis, offering the potential for quicker and more accurate assessments. ML techniques leverage algorithms to analyze vast datasets, extracting meaningful patterns and insights that may not be readily apparent to human observers. By harnessing various ML classification models, such as convolutional neural networks (CNNs) like Visual Geometry Group (VGG), DenseNet, NasNet, AlexNet, InceptionNet, ResNet, ExceptionNet, and EfficientNet, the intricate visual patterns associated with ASD can be discerned and analyzed with unprecedented efficiency.

In this project, our objective is to leverage the power of machine learning and deep learning methodologies to develop a robust system capable of detecting ASD using visual cues. Central to our approach is the assembly of a diverse and meticulously labeled dataset encompassing both ASD-affected individuals and neurotypical counterparts. Through rigorous data preprocessing, model training, and evaluation using metrics such as accuracy, precision, recall, and F1 score, we aim to optimize the performance of our detection system.

Ultimately, our vision is to create a user-friendly interface tailored for clinicians, empowering them with timely and accurate diagnostic insights. By facilitating early intervention and support, our system not only aims to improve individual outcomes but also raise awareness about the transformative role of technology in augmenting healthcare practices. Through collaboration and innovation, we strive to make meaningful strides towards enhancing the lives of individuals affected by ASD and their families.

1.2 SCOPES

Our model aims to expedite the diagnosis of autism, since traditional practice requires significant time and expertise from medical practitioners for therapies and detection. Leveraging facial features from images, our system accurately predicts whether a child exhibits characteristic associated with autism spectrum disorder or not. Through our website, users can upload images or capture a photo to determine whether a child is autistic or non-autistic. The objective of the website is to provide a user-friendly platform for users to quickly determine whether target inputs exhibit characteristics indicative of autism spectrum disorder (ASD) or not, facilitating early detection and intervention.

1.3 DESIGN WORKFLOW

In the following figure, as per the sequence begins with acquiring a dataset from Kaggle. This dataset undergoes preprocessing, including dimensionality reduction and normalisation, followed by feature extraction. Deep learning models are then employed to determine the best performing model among them. Finally, the chosen model is deployed into web application for classifying whether a child is autistic or non-autistic.

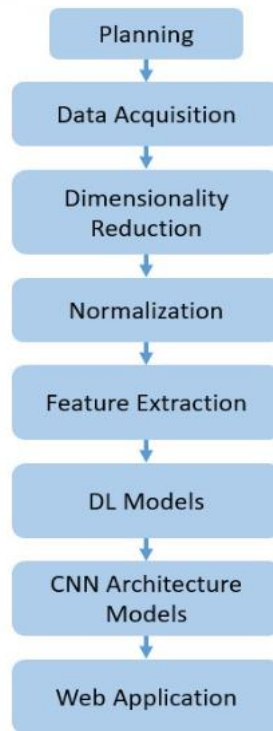


Figure 1: Design workflow

1.4 ORGANISATION OF REPORT

- In Chapter 1, introduction and idea of the project has been elaborated.
- In Chapter 2, problem statement and objectives of the project has been specified.
- In Chapter 3, based on the research papers inferred, data has been gathered and comparisons have been made between them in tabular form.
- In Chapter 4, description of the project along with details about DL algorithms have been elaborated
- In Chapter 5, functional and non-functional requirements to support the project have been specified.
- In Chapter 6, methodology and working of the project has been explained.
- In Chapter 7, we have compared the experimented results of each algorithm and gave a full overview in the form of tables, graphs, etc.
- In Chapter 8, accuracy percentages of the deployed models have been found and based on that, the best ones among them have been selected.
- In Chapter 9, conclusion has been given.
- Finally, the paper referred to support the project idea have been mentioned.

CHAPTER 2

PROBLEM DEFINITION

CHAPTER 2 PROBLEM DEFINITION

2.1 PROBLEM DEFINITION:

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental disorder that affects communication, behaviors, and social interaction. ASD encompasses a wide range of symptoms, skills, and levels of impairment, making it a "spectrum" disorder. Early detection and diagnosis of ASD are crucial for initiating appropriate interventions and support, which can significantly improve the quality of life for individuals with ASD and their families. Individuals with ASD often have difficulties in understanding and interpreting social cues, gestures, facial expressions, and emotions. They may struggle with initiating and maintaining conversations, making eye contact, and understanding the nuances of social interactions. Repetitive behaviors, routines, and interests are common in individuals with ASD. This can manifest as repetitive movements (e.g., hand-flapping, rocking), insistence on sameness, and intense focus on specific topics or activities. Verbal and non-verbal communication difficulties are prevalent in ASD. Some individuals may have delayed language development, limited vocabulary, or difficulty expressing their needs, feelings, and thoughts effectively, leading to frustration and misunderstandings. People with ASD may experience challenges in regulating their emotions and behavior's. They may exhibit impulsive behavior's, meltdowns, anxiety, depression, or difficulties in understanding and managing their own emotions and those of others. Executive functions such as planning, organizing, problem-solving, and managing time can be challenging for individuals with ASD. They may struggle with tasks that require multitasking, flexibility, and abstract thinking. Many individuals with ASD face difficulties in educational settings due to learning differences, sensory sensitivities, and social communication challenges. As adults, they may encounter challenges in finding and maintaining employment due to these differences and potential stigmas associated with ASD. The transition from adolescence to adulthood can be particularly challenging for individuals with ASD, as they navigate increased independence, vocational training, higher education, relationships, and potential changes in support services. Accessing appropriate services, therapies, and support can be challenging for individuals with ASD and their families due to limited resources, long waitlists, and varying availability of specialized professionals and programs.

By addressing these challenges our project tries to amend the existing research in classification or prediction of autism spectrum disorder.

2.2 OBJECTIVES:

Objectives of the project:

- The project is developed for early identification of ASD symptoms to facilitate timely intervention and support.
- Our Project aims to improve the accuracy and reliability of ASD diagnosis by leveraging advanced screening, algorithms, and diagnostic criteria.
- Leverage advancements in technology, such as machine learning, artificial intelligence, and digital health platforms, to enhance the efficiency, accuracy, and accessibility of ASD detection methods.
- Using Deep Learning architectures like Res Net, Nas Net, Alex Net, Dense Net, Inception Net, Efficient Net, Xception Net, VGG-16 where we compare their accuracy percentages.
- Our project can analyse individual characteristics and patterns in ASD-related behaviors, facilitating the development of personalized treatment plans tailored to each individual's needs and symptoms.

CHAPTER 3

LITERATURE REVIEW

CHAPTER 3 LITERATURE REVIEW

The paper presents a machine learning approach to detecting Autism Spectrum Disorder (ASD) based on analyzing specific behaviors from videos of infants aged 6 to 36 months. It utilizes a large dataset of 2000 videos manually coded for behaviors like directed gaze, positive affect, and vocalization. Initially, deep learning models are developed for behavior classification from raw video frames and facial features. The results show promising accuracies for smile, look face, and look object detection. Subsequently, the study explores ASD diagnosis prediction using statistical features of these behaviors. Feature selection techniques and class rebalancing methods are employed to improve classification accuracy. The proposed framework achieves an 82% accuracy for ASD diagnosis prediction. Future work includes incorporating temporal information, utilizing audio data, and enhancing object detection for further improvements.[3]

The paper presents a convolutional neural network (CNN) model for the early detection of Autism Spectrum Disorder (ASD) in children using facial images. Traditional methods rely on behavioral observations, which can be time-consuming and prone to errors. The proposed CNN model achieved an impressive accuracy of 92.31%, outperforming other machine learning algorithms like Random Forest and Gradient Boosting Machine. The study utilized a dataset of 2940 children's face images, evenly distributed between autistic and non-autistic classes. Through rigorous evaluation measures including accuracy, ROC AUC, F1-score, precision, and recall, the CNN model consistently demonstrated superior performance. The CNN architecture involved preprocessing images to a standard size, applying convolution and pooling layers, and utilizing dropout layers to prevent overfitting. The model was trained using Adam optimizer and binary-cross entropy loss function, achieving a testing accuracy of 92.31% after 60 epochs. The research underscores the potential of CNN-based models as effective tools for aiding medical practitioners in early ASD detection, offering a quicker and more accurate alternative to traditional diagnostic methods.[1]

This research focuses on using deep neural networks to classify ASD in children aged 4 to 11 years. The study utilizes data from the UCI Machine Learning repository, employing standardization and dimension reduction techniques before classification. The proposed approach achieves promising results, outperforming some traditional methods. The DNN model, especially when trained with missing data, shows higher accuracy and sensitivity.

Though performance slightly decreases with complete data, it remains clinically acceptable. This method emphasizes early ASD detection in children to enhance their quality of life.[12]

The paper explores using facial features as a potential biomarker for predicting Autism Spectrum Disorder (ASD) through a deep learning Convolutional Neural Network (CNN) algorithm. It emphasizes the importance of early detection and intervention for individuals with ASD and discusses the potential of CNNs in accurately predicting ASD based on facial images. The study uses a publicly available dataset, applies preprocessing techniques to enhance data quality, and achieves high accuracy, sensitivity, and specificity in ASD prediction.[10]

Table 3: Comparison between different research papers

Author's Name/Paper Title	Conference/Journal Name and Year	Technology/Design	Results shared by author	What you infer
Paper Title: A Convolutional Neural Network Model for Early-Stage Detection of Autism Spectrum Disorder Author's Name: Md. Fazle Rabbi, S. M. Mahedy Hasan	International Conference on Information and Communication Technology, 2021	CNN (Convolutional Neural Network)	CNN for the early-stage detection of ASD in children, which can significantly impact the quality of life for affected children and their families	CNN can be used to achieve highest accuracy and could be a helpful tool for doctors to diagnose autism more accurately and quickly
Paper Title: SP-ASDNET: CNN-LSTM BASED ASD CLASSIFICATION MODEL USING OBSERVER SCANPATHS Author's Name: Yudong Tao, Mei-Ling Shyu	IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2019	SP-ASDNET	The proposed model achieved 74.22% accuracy on the validation dataset, but performance on the test dataset was lower, indicating potential overfitting.	The SP-ASDNet model provides a new and hopeful way to identify Autism Spectrum Disorder (ASD) by analysing the way people look at images.

Paper Title: Machine Learning Based Autism Spectrum Disorder Detection from Videos, Author's Name: Chongruo Wu, Sidrah Liaqat, Halil Helvaci	IEEE International Conference on E-health Networking Application & Services (HEALTHCOM)	Synthetic Minority Over-sampling Technique (SMOTE) and Tomek Links for class rebalancing	Deep learning, in automated ASD diagnosis using video data of infant behaviors.	The use of machine learning and deep learning techniques on video data offers a promising approach to address the challenges associated with current ASD screening methods.
Paper Title: A Machine Learning Approach to Predict Autism Spectrum Disorder Author's Name: Kazi Shahrukh Omar, Prodipta Mondal, Nabila Shahnaz Khan	International Conference on Electrical, Computer and Communication Engineering (ECCE), 7-9 February, 2019	Decision Tree-CART Algorithm, Random Forest-CART Algorithm, Random Forest-CART and Random Forest-ID3	Prediction model provided better performance in terms of accuracy, specificity, sensitivity, precision, and false positive rate compared to the other existing models	The performance slightly decreased when tested on a real-world dataset, indicating the need for more diverse and larger datasets for training the prediction model to improve its accuracy.
Paper Title: Autism Spectrum Disorder Prediction by Facial Recognition Using Deep Learning Author's Name: Kanimozhi A, Dhanasri A	International Journal of Creative Research Thoughts (IJCRT), 2024	<u>ResNet50 for ASD</u>	ResNet50 can provide a quick and efficient diagnosis of ASD, enabling early intervention and treatment to improve the child's quality of life.	The use of ResNet50, a deep learning algorithm effective in image classification tasks, enhances the accuracy of ASD prediction.

Paper Title: A Machine Learning Framework for Early-Stage Detection of Autism Spectrum Disorders Author's Name: MUHAMMAD IMRAN SHARIF, ANWAAR ULHAQ	2023	Machine learning algorithms for classification and Four Feature Selection Techniques: Info Gain Attribute Evaluator (IGAE), Gain Ratio Attribute Evaluator (GRAE), Relief F Attribute Evaluator (RFAE), Correlation Attribute Evaluator (CAE)	Use of four feature scaling methods and eight machine learning classifiers to classify the data will improve ASD detection and other neuro-developmental disorders.	Analyzed the performance of each scaled dataset and determined the best feature scaling and classification techniques.
Paper Title: Efficient Net-based Transfer Learning Technique for Facial Autism Detection Author Name: TARIQ SAEED MIAN	2023	EFFICIENT NET-BASED TRANSFER LEARNING	This research illustrates that predictive analysis based on transfer learning using EfficientNetB0 on autism images is highly effective in ASD prediction	Addressed the challenge of transfer learning by utilizing the last three layers of EfficientNetB0 to detect autism disorder due to limited training data.
Paper Title: Classification and Detection of Autism Spectrum Disorder Based on Deep Learning Algorithms Author Name: Fawaz Waselallah Alsaade and Mohammed Saeed Alzahrani	28 February 2022	CNN model Xception, NASNET Mobile, and VGG19	This research underscores the potential of deep learning models in revolutionizing the early detection and diagnosis of autism through facial features.	The utilization of deep learning models enhance the accuracy and efficiency of autism diagnosis.

Paper Title: Performance Analysis of Deep Learning Models for Detection of Autism Spectrum Disorder from EEG Signals Author's Name: Menaka Radhakrishnan, Karthik Ramamurthy	International Information and Engineering Technology Association, 2021	ResNet50	The ResNet50 based deep convolutional network was identified as the most effective in detecting ASD from EEG signals	fine-tuning the ResNet50 architecture with larger datasets to improve accuracy further.
Paper Title: A Deep Convolutional Neural Network Based Prediction System For Autism Spectrum Disorder In Facial Images Author's Name: Prof. Jyothi S, Sneha S, Preetha P	International Journal of Research in Engineering and Science (IJRES)	Deep Learning CNN algorithm	The precision and recall scores were also calculated to be 80% and 83%, respectively, indicating a good balance which aids in identifying ASD.	Using AI on facial images can better detect ASD, making screening easier, faster, and more accessible.
Paper Title: Autism Spectrum Disorder Detection Using Facial Images and Deep Convolutional Neural Networks Author's Name: Lalitha Kumari Gaddala, Koteswara Rao Kodepogu	International Information and Engineering Technology Association, 2023	Visual Geometry Group models (VGG16 and VGG19)	VGG16 and VGG19 models on face images to classify autism. VGG16 showed better performance.	VGG16 model performed much better than the VGG19 model in our investigation on autism classification.

CHAPTER 4

PROJECT DESCRIPTION

CHAPTER 4 PROJECT DESCRIPTION

4.1 AUTISM CLASSIFICATION ARCHITECTURE

This block diagram holds the experimentation workflow of the process of the project. The first step is data collection which holds distinct images for training and testing, after the collection of images pre-processing of dataset is performed by using various techniques such as resizing of images, feature extraction, labelling the images, noise removal, and finally converting it to numpy array. Utilization of pre-trained models such as RES NET, NAS NET, ALEX NET, DENSE NET, INCEPTION NET, EFFICIENT NET, XCEPTION NET, VGG-16. These models are functioning by connecting model to layers, calculating loss function, performing hyper parameter tuning, calculation of learning rate, based on these characteristics the model is trained and testing is performed. Once the testing is completed the model predicts and its validated using metrics such as accuracy, training and testing validation loss, confusion matrix, roc-curve. The models that perform the best and provides higher efficiency is used at the backend for classification and prediction results can be seen at the frontend.

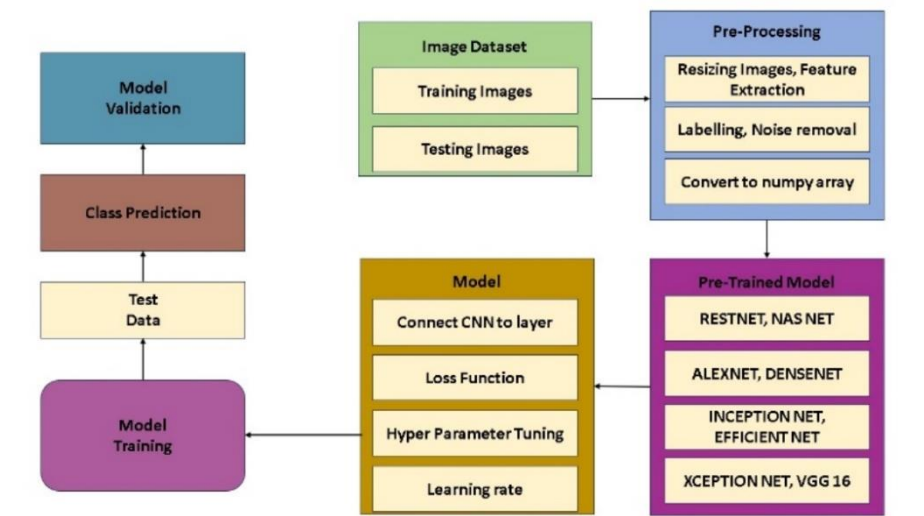


Figure 4 Block Diagram of ASD

4.1.1 DATA COLLECTION

The dataset autism detection project is structured into four categories: training, testing, validation and consolidation. The total images of the dataset include 4250. Each category contains images categorized as autism or non-autism. The training dataset is utilized to train our CNN models, while the testing dataset is employed to evaluate their performance on unseen data.

4.1.2 DATA PREPROCESSING

4.1.2.1 Resizing of images

Convolutional neural networks (CNNs) commonly used in image classification, require a fixed-size input. Resizing ensures that all images fed into the model have the same dimensions, which is essential for the model to process the data correctly.

4.1.2.2 Feature Extraction

Feature extraction helps reduce the dimensionality of the data by transforming it into a smaller set of meaningful features that capture the essential information required for the model to learn effectively. Reduced feature sets require less computational power and memory to process, making model training and inference faster and more efficient. This efficiency is particularly important for large datasets and complex models.

4.1.2.3 Labelling

Labelling images in data pre-processing is crucial for creating a high-quality, labelled dataset that can be used to train and evaluate machine learning models effectively.

4.1.2.4 Noise removal

Overall, noise removal is a critical pre-processing step that helps to clean and prepare the data for analysis, removing noise helps to produce clearer and more informative visualizations, aiding in data exploration and interpretation. Hence, making it more reliable for analysis and modelling.

4.1.2.5 Convert list to NumPy array

By converting lists to NumPy arrays and leveraging NumPy's capabilities, we can efficiently pre-process and prepare data for machine learning tasks such as NumPy allows for vectorized operations, which means that operations can be applied to entire arrays without the need for explicit loops. This can significantly speed up preprocessing tasks.

4.1.3 PRE-TRAINED MODELS

In our project we made use of pre-trained models such as RES NET, NAS NET, ALEX NET, DENSE NET, INCEPTION NET, EFFICIENT NET, XCEPTION NET, VGG-16 and

obtained the results of each model and further involved the best efficient model for further classification of autism.

4.1.3.1 RESNET

ResNet architecture is used to learn features that are useful for prediction. During training, the weights of the network are adjusted to minimize the prediction error. ResNet is used to address the problem of vanishing gradients indicating that hardly any change comes to weights which makes it difficult for the network to learn and optimize the model parameters. The description of ResNet is provided below.

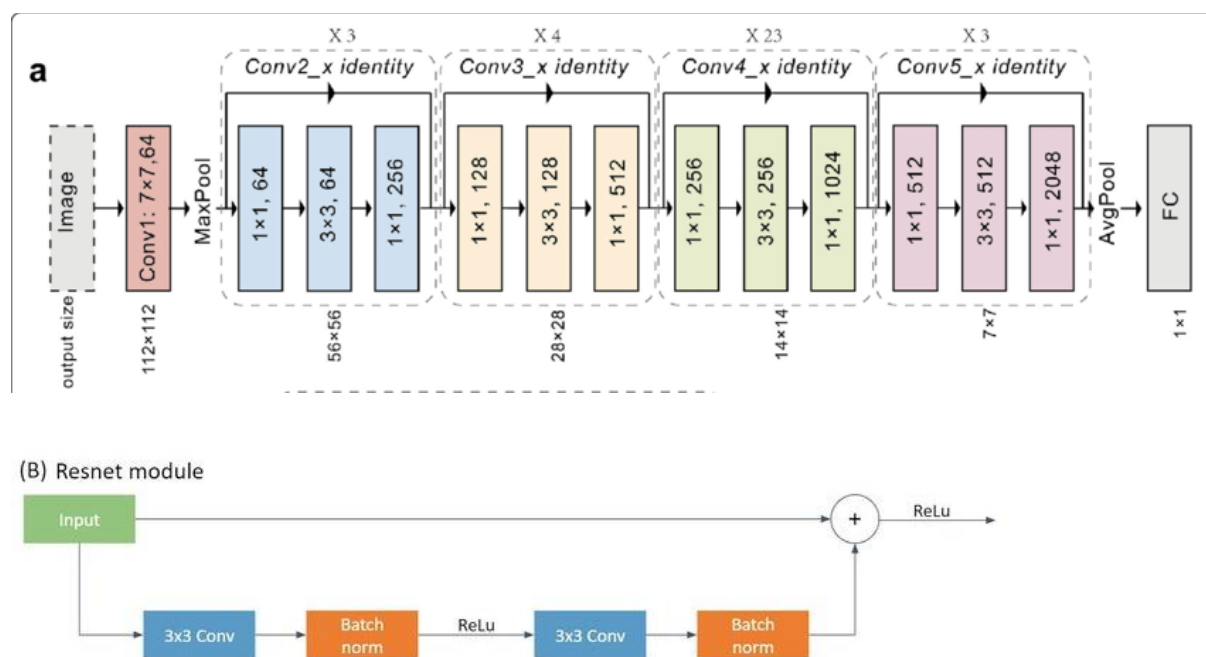
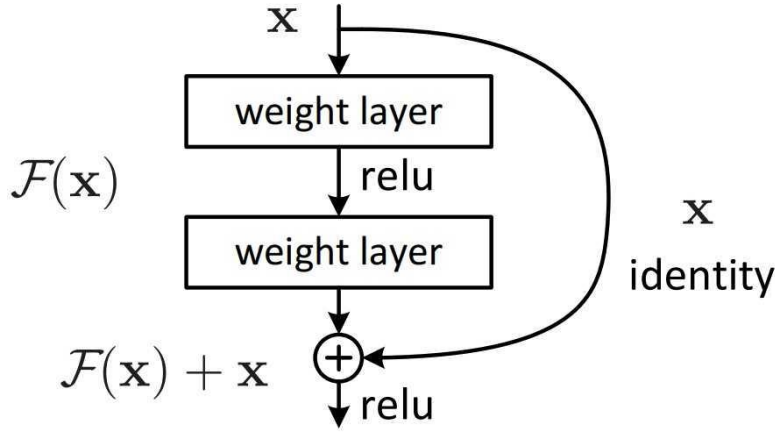


Figure 4.1 a) ResNet Architecture; B) ResNet Modules

The ResNet architecture typically begins with a 7x7 convolutional layer with a stride of 2, followed by batch normalization and ReLU activation, and then a max pooling operation. Following this initial setup, the network is organized into four stages, each comprising multiple residual blocks. The number of blocks varies based on the ResNet configuration chosen (i.e. ResNet-50). Throughout these stages, the size of feature maps is halved, while the number of filters is doubled to maintain computational efficiency per layer. The network concludes with a global average pooling layer, which averages the feature maps spatially, followed by a fully connected layer with 1000 units, corresponding to the 1000 classes in the ImageNet dataset. A softmax activation function is applied to generate the final probability distribution across the classes.



The residual connection creates a shortcut path by adding the value at the beginning of the block, x , directly to the end of the block ($F(x) + x$) [1].

a. Convolutional Layers:

Res-Net begins with initial convolutional layers to extract low-level features from the input data. These convolutional layers are represented by the equation:

$$H_l = \sigma(W_l * H_{l-1} + b_l). \quad (1)$$

Where H_l is the output feature map of layer l .

W_l represents the weights of the convolutional filters.

b_l represents the biases.

$*$ denotes the convolution operation.

σ represents the activation function, typically ReLU.

b. Residual Blocks:

The core of Res-Net architecture consists of residual blocks. These blocks enable the network to learn residual mappings, facilitating the training of very deep networks. The output of a residual block can be represented as:

$$H_{l+1} = F(H_l) + H_l \quad (2)$$

Where H_{l+1} is the output of the residual block.

F represents the residual function learned by the block.

H_l is the input to the residual block.

c. Fully Connected Layers:

After several layers of convolution and pooling, the feature maps are flattened and passed through fully connected layers for classification. The output of the fully connected layer can be represented by:

$$Z = W_{FC} + F_{flat} + b_{FC} \quad (3)$$

Where Z is the output of the fully connected layer.

W_{FC} represents the weights of the fully connected layer.

F_{flat} represents the flattened feature maps.

b_{FC} represents the biases.

By adding the input to the changed output, Res Net efficiently learns the residual mapping, allowing the network to focus on learning the “difficult” parts of the mapping rather than the complete mapping from scratch.

4.1.3.2 ALEXNET

AlexNet architecture is used for robust Image variation. In AlexNet, the number of filters increases as we go deeper hence extracting more features into the model. Also, the filter size decreases as we go ahead resulting in a decrease in the feature map shape.

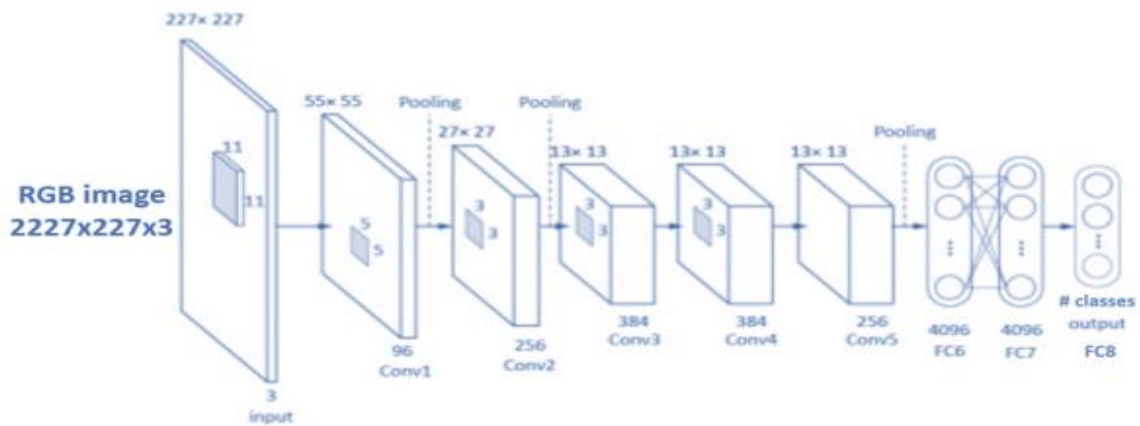


Figure 4.2 AlexNet architecture

The AlexNet architecture comprises eight layers with weights, including five convolutional layers and three fully-connected layers. The initial convolutional layer serves as an input layer, filtering 227x227x3 input images with 96 kernels sized 11x11x3, employing a stride of 4 pixels. Subsequent convolutional layers (second, fourth, and fifth) are locally connected to kernel maps on the same GPU. However, the third convolutional layer connects to all kernel maps of the preceding layer. Neurons in the fully-connected layers establish connections with all neurons in the previous layer, culminating in a comprehensive network structure.

Let's discuss the math behind Alex Net's framework:

a. Convolutional Layers (Conv):

AlexNet consists of five convolutional layers. Each convolutional layer applies a set of learnable filters to the input image to produce feature maps. These feature maps capture different aspects of the input image.

$$\text{Equation for a convolutional layer: } Z^l = W^l * A^{l-1} + b^l \quad (4)$$

$$A^l = \text{ReLU}(Z^l) \quad (5)$$

where Z^l is the output of the convolutional layer, W^l is the filter weights, A^{l-1} is the input feature map from the previous layer, and b^l is the bias term.

b. Max Pooling Layers:

$$A^l = \text{MaxPooling}(A^{l-1}, \text{filtersize}, \text{stride})$$

where, MaxPooling is the max pooling operation.

c. Fully Connected Layers (FC):

$$Z^l = W^l \cdot A^{l-1} + b^l \quad (6)$$

$$A^l = \text{ReLU}(Z^l) \quad (7)$$

$$A_{\text{Dropout}}^l = \text{Dropout}(A^l, \text{dropoutrate}) \quad (8)$$

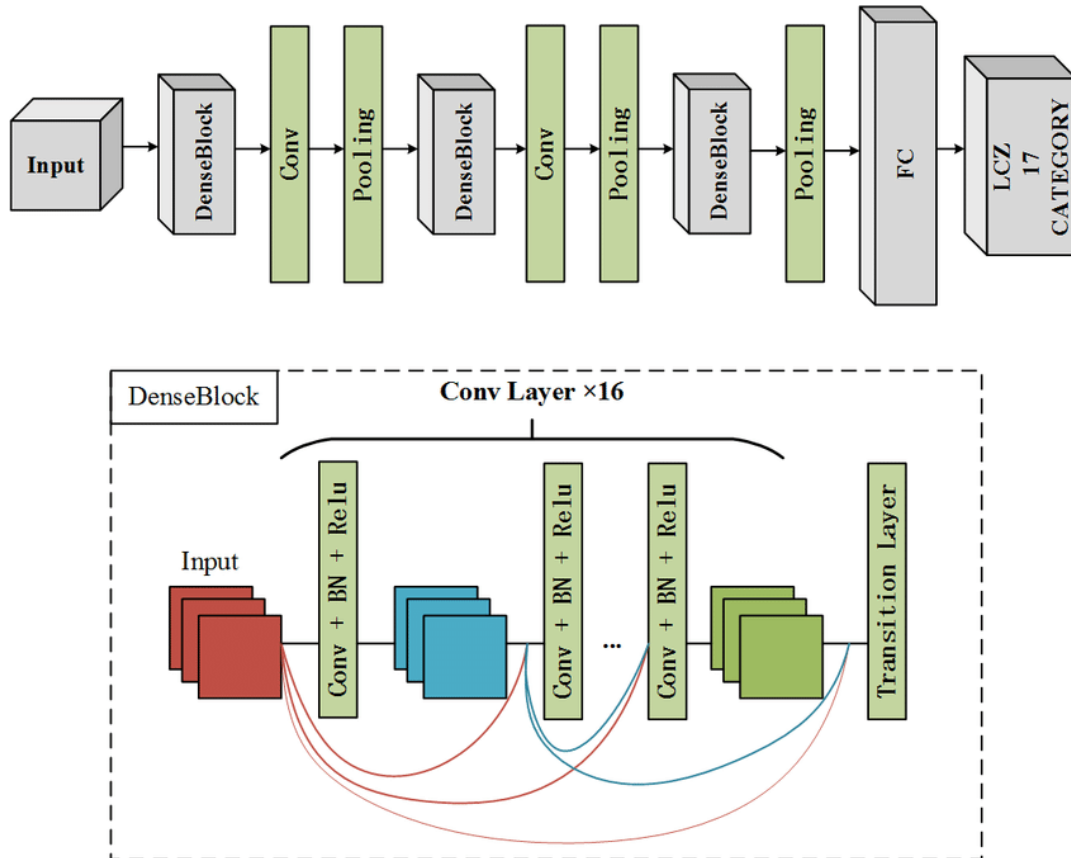
where Z^l is the output of the convolutional layer, W^l is the filter weights, A^{l-1} is the input feature map from the previous layer, and b^l is the bias term, and dropout is dropout operation.

d. Soft-max Activation:

$$P(y = i|x) = \frac{e^{z_i}}{\sum_{j=1}^{1000} e^{z_j}} \quad (9)$$

4.1.3.3 DENSENET

DenseNet architecture densely connect the various layers and flows through one by one to reduce the size of feature maps, thus making the model parameter efficient and impicit deep supervision for improved flow of gradient. DensNet takes all previous output as an input for a future layer due to the long distnace between input and output kayers and information may vanish brfore reaching its destination.



Figur 4.3 DenseNet Architecture

The DenseNet architecture features dense connectivity patterns, with each layer receiving input from all preceding layers, enhancing feature reuse and gradient flow. It consists of dense blocks, where convolutional layers are densely connected within each block, promoting

feature diversity. Transition layers, composed of batch normalization, convolution, and pooling operations, are inserted between dense blocks to manage feature map sizes. Within each dense block, feature maps from previous layers are concatenated before passing through convolutional layers. This dense connectivity fosters feature propagation and encourages feature reuse, leading to highly discriminative representations. Finally, global average pooling and a fully connected layer with softmax activation enable effective classification of complex data distributions.

Following is the description of DenseNet.

a. Dense Block:

The key innovation in Dense Net is the dense block. In a dense block, each layer receives feature maps from all preceding layers and passes its own feature maps to all subsequent layers.

Let's denote the input to the $hlth$ layer in a dense block as xl . The output of the $hlth$ layer, denoted as $Hl(xl)$, is computed as follows:

$$H_l(xl) = \sigma(W_l * [\text{Concat}(X_0, X_1, X_2, \dots, X_{l-1})] + b_l) \quad (10)$$

where:

* represents convolution operation.

Concat denotes concatenation of feature maps.

σ is the activation function (typically ReLU).

When deconstructing the i th layer received all the feature maps from previous layer as input: $[X_0, X_1, X_2, \dots, X_{l-1}]$ Where $X_l = H_l([X_0, X_1, X_2, \dots, X_{l-1}])$ is considered as a single tensor.

b. Transition Layer:

Between dense blocks, transition layers are used to reduce the number of feature maps and control the model's complexity.

Transition layer operation:

$$Hl(xl) = \sigma(Wl * xl + bl) \quad (11)$$

Followed by average pooling to reduce the spatial dimensions.

It performs three different consecutive operations:

Batch- Normalization (BN), followed by a ReLU and a 3×3 convolution operation. In the transaction block, 1×1 convolutional operations are performed with BN followed by a 2×2 average pooling layer.

4.1.3.4 INCEPTION NET

Inception Net is designed for image classification, segmentation and detection. The idea behind this model is to allow the network to learn spatial and temporal features at different scales and then concatenate them together to form a more comprehensive representation of the input data. The model is used to train more efficiently and faster than other models. The discription of Inception is given below.

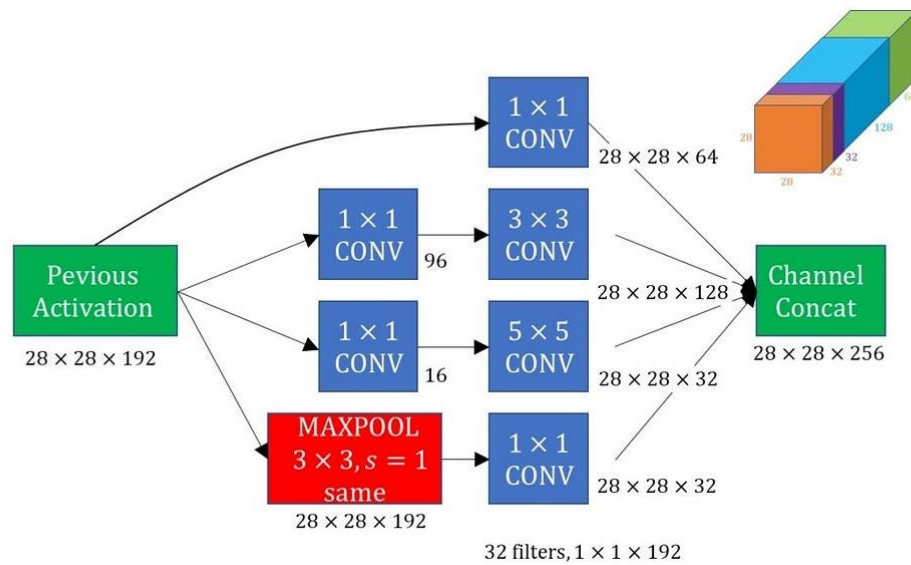


Figure 4.4 Inception Net Architecture

In GoogLeNet (Inception v1), a deep convolutional network with 22 layers, computational costs are mitigated through the strategic use of 1×1 convolutions to limit input channels before 3×3 and 5×5 convolutions. The addition of 1×1 convolutions after max-pooling reduces computational overhead while maintaining effectiveness. All channels are concatenated, resulting in a unified feature map size of $28 \times 28 \times 256$. GoogLeNet's architecture incorporates 9 inception modules linearly, contributing to its depth. Fully connected layers are replaced by global average pooling, drastically reducing parameter count and computational complexity.

a. Inception Module:

The fundamental building block of Inception is the inception module, which performs parallel convolutions of different sizes on the input feature maps and concatenates the outputs.

Let's denote the input to the inception module as x .

The output $\text{Inception}(x)$ is computed as follows

$$\text{Inception}(x) = \text{Concat}(\text{Conv } 1 \times 1(x), \text{Conv } 3 \times 3(x), \text{Conv } 5 \times 5(x), \text{MaxPool } 3 \times 3(x)) \quad (12)$$

where:

- $\text{Conv } 1 \times 1, \text{Conv } 3 \times 3, \text{Conv } 5 \times 5$ are convolutional operations with filter sizes 1×1 , 3×3 , and 5×5 respectively.
- $\text{MaxPool } 3 \times 3$ is max-pooling with a 3×3 filter size.

b. Factorization:

To reduce computational cost, 5×5 convolutions are factorized into two 3×3 convolutions.

$$\text{Conv } 5 \times 5(x) = \text{Conv } 3 \times 3(\text{Conv } 3 \times 3(x)) \quad (13)$$

4.1.3.5 EFFICIENT NET

Efficient Net is considered as one of the powerful CNN architecture. The model scaling method uses a compound coefficient to uniformly scale the depth/width/resolution dimensions instead of standard practice, which scales these factors arbitrarily. Thus, as image resolution improves, the network's depth and width also improve.

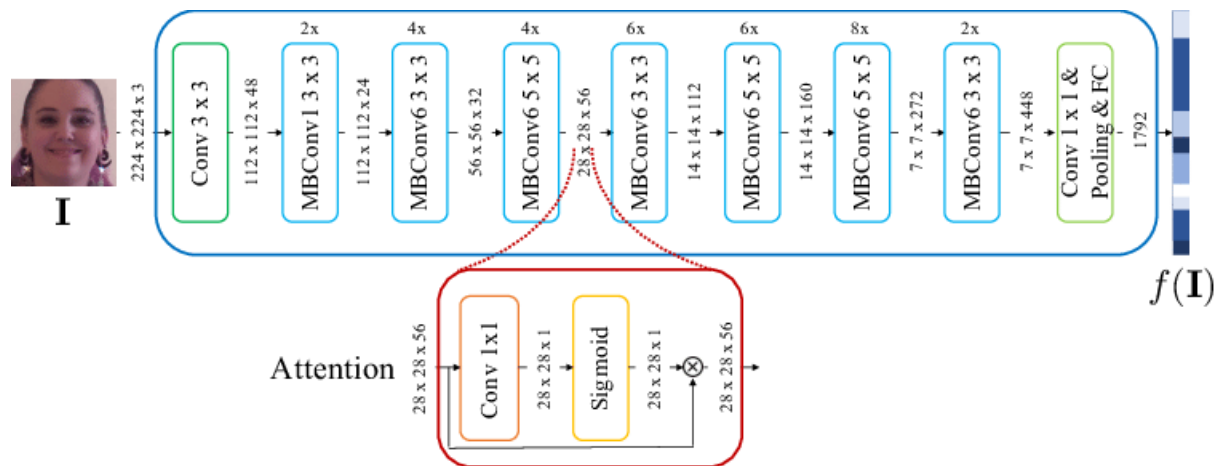


Figure 4.5 Efficient Net architecture

a. Depth wise Separable Convolution:

Efficient Net primarily uses depth wise separable convolutions, which consist of a depth wise convolution followed by a pointwise convolution. This reduces the number of parameters and computational cost while maintaining representation power. The operation of depth wise separable convolution can be represented as follows:

$$\text{ConvDepthwise}(x) = \text{DepthwiseConv}(x) \quad (14)$$

$$\text{ConvPointwise}(x) = \text{PointwiseConv}(\text{ConvDepthwise}(x)) \quad (15)$$

b. Efficient Net Scaling:

Efficient Net introduces compound scaling to uniformly scale network width, depth, and resolution. It uses coefficients to scale the width, depth, and resolution of the baseline network architecture

$$\text{Width}_{\text{new}} = \alpha \times \text{Width}_{\text{base}} \quad (16)$$

$$\text{Depth}_{\text{new}} = \beta \times \text{Depth}_{\text{base}} \quad (17)$$

$$\text{Resolution}_{\text{new}} = \gamma \times \text{Resolution}_{\text{base}} \quad (18)$$

Where α, β, γ are scaling coefficients.

4.1.3.6 XCEPTION NET

Xception model captures features at multiple spatial scales by using filters of different sizes within the same layer. Xception perform separate convolution for spatial and depth information which leads to reduction in the number of parameters.

The data in the XceptionNet goes through entry level, then through the middle flow and finally through the exit flow. Convolution and separable function follows Batch Normalisation.

a. Depthwise Separable Convolutions: Xception primarily uses depthwise separable convolutions, which consist of a depthwise convolution followed by a pointwise convolution. This factorization reduces the number of parameters and computational cost while maintaining representation power.

$$\text{ConvDepthwise}(x)=\text{DepthwiseConv}(x) \quad (19)$$

$$\text{ConvPointwise}(x)=\text{PointwiseConv}(\text{ConvDepthwise}(x)) \quad (20)$$

4.1.3.7 VGG-16

VGG 16 make sure that for every two or three convolutional layers, filters and max-pooling is applied to reduce the spatial dimensions. The increased depth allows the VGG16 to learn more complex features from the input images which leads to improved performance. The description of VGG16 is provided below.

VGG 16 characterized by its depth, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. The model's architecture features a stack of convolutional layers followed by max pooling layers, with progressive increasing depth.

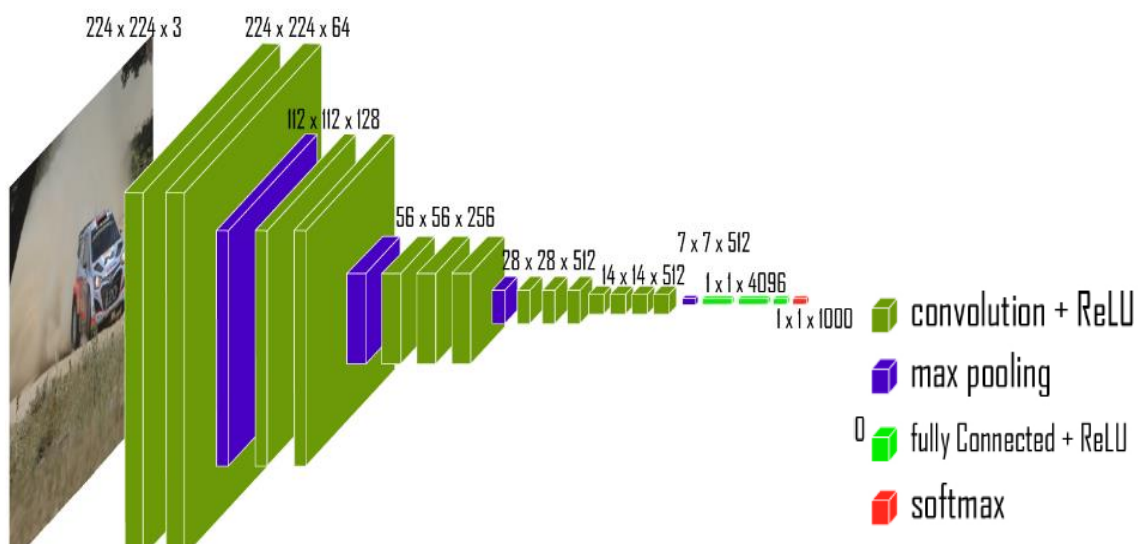


Figure 4.7 VGG 16 Layers

In the above figure consists of:

Input layer and its dimensions (224,224,3)

Convolutional layers (64 filters, 3*3 filters, same padding)

Two consecutive convolutional layers with 64 filters each and a filter size of 3*3

Max pooling Layer (2*2, stride 2): Max pooling layer with a pool size of 2*2 and a stride of 2.

Flattening: Flatten the output feature map (7*7*512) into a vector of size 25088.

Fully connected Layers: Three fully connected layers with Relu activation.

SoftMax activation function is applied to the output of the third fully connected layer of classification.

a. Convolutional Layers (Conv):

The VGG-16 model consists of 13 convolutional layers, each followed by a Rectified Linear Unit (ReLU) activation function. The convolutional layers perform feature extraction by applying a set of learnable filters to the input image. Mathematically, the operation of a convolutional layer can be represented as:

$$Z^l = W^l * A^{l-1} + b^l \quad (21)$$

Where Z^l is the output of the l^{th} layer.

W^l is the set of learnable filters (weights) for the l^{th} layer.

A^{l-1} is the output of the previous layer.

b^l is the bias term.

$*$ denotes the convolution operation.

b. ReLU Activation Function:

ReLU (Rectified Linear Unit) is applied element-wise to the output of each convolutional layer to introduce non-linearity:

$$A^l = ReLU(Z^l) \quad (22)$$

$$ReLU(x) = \max(0, x) \quad (23)$$

c. Max Pooling Layers:

After some of the convolutional layers, max-pooling layers are applied to reduce the spatial dimensions of the feature maps while retaining the most important information. It selects the maximum value from each window. The mathematical representation of max pooling is:

$$A_{[l]} = MaxPool(A_{[l-1]}, stride = 2, filter_{size} = 2)$$

d. Fully Connected (Dense) Layers:

The last three layers of the VGG-16 model are fully connected layers. These layers take the flattened output of the preceding convolutional layers and perform classification. Mathematically, the operation of a fully connected layer can be represented as:

$$Z^l = W^l A^{l-1} * b^l \quad (24)$$

Where Z^l is the output of the l^{th} layer.

W^l is the weight matrix.

A^{l-1} is the output of the previous layer.

b^l is the bias term.

e. Softmax Activation:

The final layer of the VGG-16 model uses a softmax activation function to produce probabilities for each class. It converts the raw scores into probabilities:

$$\hat{y} = \text{Softmax}(Z^l)$$

Softmax

$$(z_i) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \quad (25)$$

Where \hat{y} is the output probability vector.

K is the number of classes.

Z_i is the raw score for class i.

4.1.3.8 NAS NET

NAS NET (Neural Architecture Search Network) is an architecture designed using Neural Architecture Search (NAS), which automates the design of deep neural network architectures.

The architecture of NAS NET is composed of a stack of cells, each of which contains a sequence of operations (e.g., convolution, pooling, and skip connections). The connections and operations within each cell are determined through the neural architecture search process.

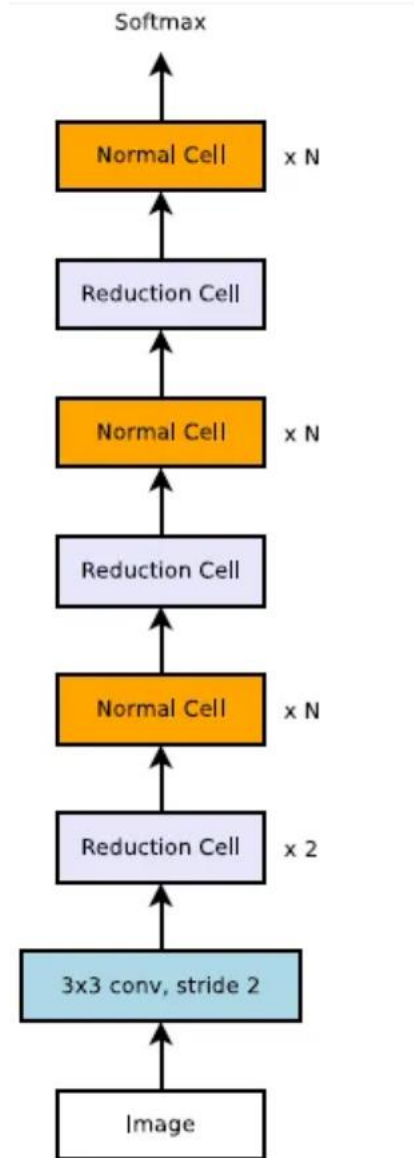


Figure 4.8 NaasNet work Flow

a. Cell Structure

A cell in NAS Net consists of a series of operations that are applied to the input feature maps. The operations include:

i. 3x3 Depth wise Convolution: This operation performs depth wise convolution on the input feature maps.

$$D(X) = \text{DepthwiseConv}(X, \text{filters}, \text{kernel_size}) \quad (26)$$

Where,

X as the input feature maps to the cell

$D(X)$ as the output from the depth wise convolution

ii. 1x1 Pointwise Convolution: This operation performs pointwise convolution (or 1x1 convolution) to combine the output feature maps from the depth wise convolution.

The depth wise convolution and pointwise convolution are followed by batch normalization and activation functions like ReLU.

$$P(D(X)) = \text{PointwiseConv}(D(X), \text{filters}) \quad (27)$$

b. NAS Net Architecture

In NAS Net, the cell structure is repeated multiple times to form the full architecture. The number of cells and the connections between them are determined through the neural architecture search process to optimize the performance on a given task.

4.1.4 MODEL FUNCTIONS

4.1.4.1 Connect CNN to layers

Connecting layers allows the network to progressively extract and combine features at different levels of abstraction. The initial layers of a CNN typically capture low-level features like edges and textures, while deeper layers capture higher-level features like shapes and objects. Connecting these layers enables the network to build a hierarchical representation of the input data, where features learned at lower layers serve as building blocks for more complex features at higher layers. The connections between the layers allow the network to transform the input data through a series of operations, gradually building a rich and abstract representation that can be used for image classification.

4.1.4.2 Loss Function

A loss function is used in Convolutional Neural Network (CNN) models to quantify the difference between the predicted outputs and the actual labels or targets. The loss function provides a measure of how well (or poorly) the model is performing on a given task, and it serves as the basis for optimizing the model's parameters during training.

4.1.4.3 Hyper parameter tuning

Hyper parameter tuning is a crucial step in the machine learning workflow aimed at finding the optimal hyper parameters for a given model. Hyper parameters are parameters that are set prior to training and control the learning process of the model, such as learning rate, batch size, number of hidden layers, and regularization strength. Tuning these hyper parameters can significantly impact the performance of the model, including its accuracy, convergence speed, and generalization ability.

Including all the above mentioned parameters the model is built and it is rigorously trained , tested and validated using performance metrics such as training and validation accuracy, training and validation loss, confusion matrix, Roc – curve.

4.2 ASSUMPTION AND DEPENDENCIES:

ASSUMPTIONS

- 1) Assuming this approach will be a better methodology in detection of autism disorder.
- 2) Assuming that this project helps us to collect more collective features in solving the issue of detection.
- 3) Assuming this project helps the mankind.

DEPENDENCIES

- 1) Deep Learning Frameworks:

TensorFlow and pyTorch: These are popular deep learning frameworks that provide high level abstractions for building and training neural networks. Availability of high-performance computational resources, such as GPUs or TPUs, for training the deep learning model efficiently.

- 2) Access to experts with knowledge in transfer learning techniques and neural network architectures for fine-tuning the model.

- 3) Availability of suitable tools and resources for data preprocessing, ensuring uniformity and compatibility of the image datasets for model training.

CHAPTER 5

REQUIREMENTS

CHAPTER 5 REQUIREMENTS

5.1 FUNCTIONAL REQUIREMENTS

- The functional requirements of this project outline the core capabilities necessary to facilitate effective ASD detection. These requirements encompass:
- Image Capture: The system enables users to capture the input image and at the same time also upload images as files for detection.
- Feature Extraction: Utilizing the Image preprocessing techniques, the system should accurately identify and extract features from the captured images or uploaded images.
- Compatibility: The system should support the input image formats and provide accurate detection to enable users to access the application for a wide range of networks.
- Reliability and Accuracy: It must consistently deliver accurate detection results to maintain user trust and confidence in the system's capabilities.

5.2 NON-FUNCTIONAL REQUIREMENTS:

Secured Performance:

- Response time: The system should have low latency in processing the captured images and generating the prediction results to ensure a smooth and uninterrupted user experience.
- Performance Requirements: It is mainly dependent on the accuracy of ML model's internal working. The chosen classification model will provide accurate results while classifying. The application is reliable in terms of taking input and displaying results as soon as possible when users click the submit button. The application will be accessible in any device with internet connection and browser.
- Responsiveness: The application should respond promptly to user interactions, ensuring that results are displayed swiftly upon submission of images. Users expect minimal delay between input submission and result display.

Certainty:

- Safety Requirements: Safety measures must be taken to make sure that server won't face downtime error while serving the web page.
- Data integrity: The system should maintain integrity of the users input and ensure that the prediction results are accurate and reliable.

- **Privacy:** The system should comply with the privacy regulations and protect user inputs, ensuring that captured images and processed results are not stored or transmitted.

5.3 HARDWARE AND SOFTWARE REQUIREMENTS:

5.3.1 HARDWARE REQUIREMENTS:

- **Server:** Local host server capable of running the necessary software for user input acceptance, image processing, detection and also the memory to handle these tasks efficiently.
- **Camera Model:** A high resolution camera module compatible with the server which is capable of capturing clear and detailed images.
- **Display:** A compatible display interface (i.e. frontend) which is connected with the server that enables user interface for real-time implementation of the project.
- **CPU:** A multicore CPU with sufficient processing power to handle data preprocessing tasks, model training, and inference. While modern CPUs can handle basic deep learning tasks, for larger datasets and more complex models, a CPU with multiple cores (e.g., Intel Core i7 or higher) is recommended.
- **RAM:** Sufficient RAM is necessary to store the dataset, model parameters, and intermediate computations during training. For image classification tasks, a minimum of 16 GB RAM is recommended, although larger datasets may require even more.

5.3.2 SOFTWARE REQUIREMENTS:

- **Server-side:** Flask will be used for backend. It provides a development server and a debugger.
- **Client-side:** Any popular web browsers which supports JavaScript, HTML 5, CSS.
- **Flask:** It is a micro web framework written in Python because it does not require any particular tools or libraries. It supports extensions that can add application features as if they were implemented in Flask itself. Applications like Pinterest and Flask use the Flask framework.
- **Google Collaboratory:** It is a cloud-based Jupyter notebook environment provided by Google. It functions as a server-client application, enabling users to create, edit, and execute notebook documents directly through a web browser. Colab notebooks can be accessed from any device with an internet connection, making it highly accessible for collaborative work and remote usage.

CHAPTER 6

METHODOLOGY

CHAPTER 6 METHODOLOGY

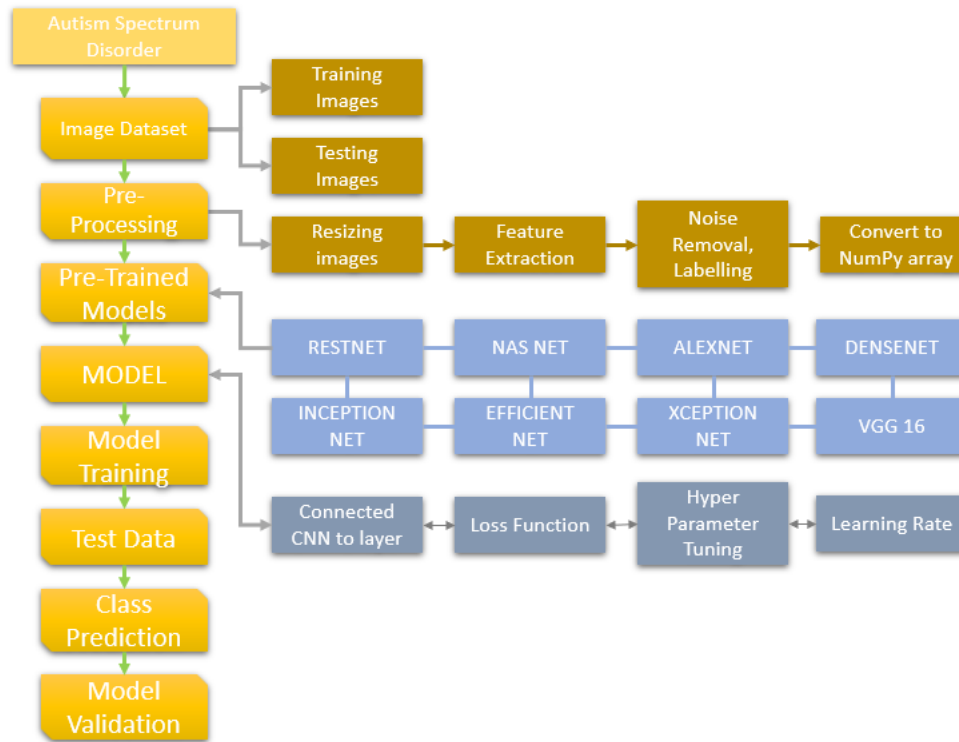


Figure 6.1: Design Diagram

Firstly, we obtained dataset from Kaggle. Where we considered images of the children to detect whether a child is autistic or non-autistic.

For data preprocessing, converting list into NumPy arrays gives indexing to the images that makes it easier for the model to assess and train on these indexed images.

Image resizing has been applied to standardize the dimensions across all images, ensuring uniformity and facilitating computational efficiency during training.

Using color contouring on facial images facilitates easier training of the image model because it enhances the contrast and delineates the facial features, making them more distinguishable.

one-hot encoding has been employed to transform categorical labels into a format suitable for machine learning algorithms, particularly beneficial for multi-class classification tasks like autism detection.

60:40 split was used for training and testing and model built using keras library. 20 epochs were chosen as optimal to train the model and to predict the accuracy.

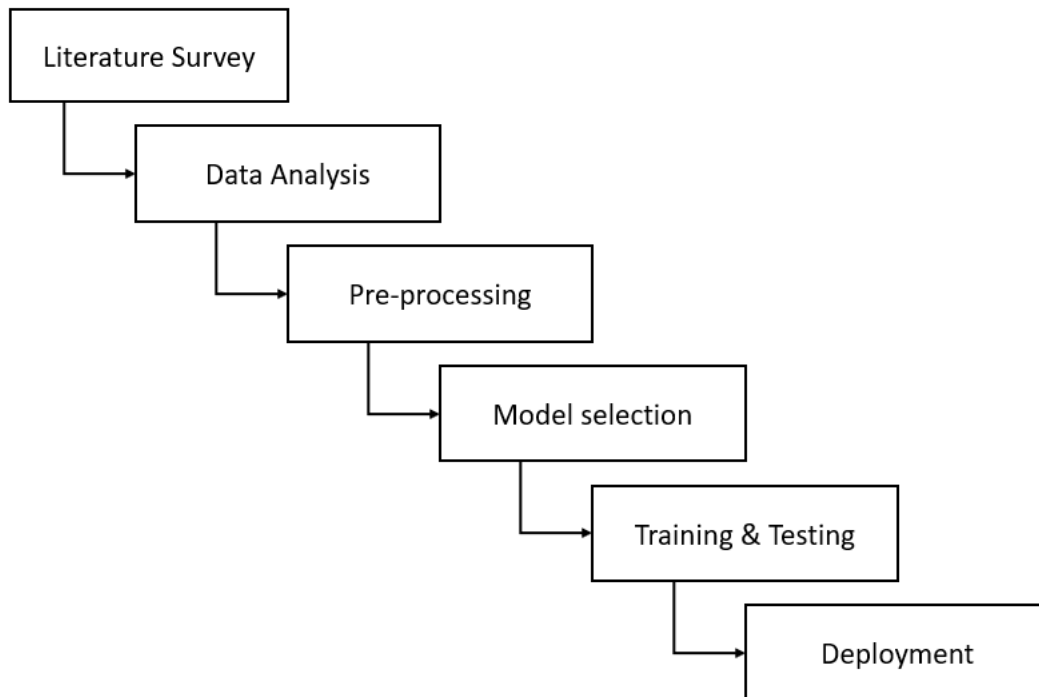


Figure 6.2 Process Flow Chart Diagram

The following tasks have been done in our project

1. Data acquisition: Our project deals with image dataset that contains images of children between the age 2 to 6 years. The models are trained based on facial expression, behaviors and alignment. The dataset autism detection project is structured into four categories: training, testing, validation, and consolidation. The dataset consists of a total of 4250 images. Each image has been resized to dimensions of 224 pixels for the X-axis, 224 pixels for the Y-axis, and 3 channels. Where each category contains images categorized as autism or non-autism. The training dataset is utilized to train our CNN models, while the testing dataset is employed to evaluate their performance on unseen data. The validation dataset aids in fine-tuning hyperparameters and preventing overfitting during training, while the consolidation dataset may serve various purposes such as refining predictions or additional evaluation.
2. Data preprocessing: For data preprocessing, several steps have been implemented. Firstly, image resizing has been applied to standardize the dimensions across all images, ensuring uniformity and facilitating computational efficiency during training. Using color contouring on facial images facilitates easier training of the image model because it enhances the contrast and delineates the facial features, making them more distinguishable. The list into NumPy arrays

representing the images have been converted into lists, likely to facilitate easier manipulation and compatibility with certain operations or libraries.

Converting list NumPy arrays gives indexing to the images that makes it easier for the model to assess and train on these indexed images.

one-hot encoding has been employed to transform categorical labels into a format suitable for machine learning algorithms, particularly beneficial for multi-class classification tasks like autism detection.

3. Model selection: CNN architectures are widely used for image classification tasks due to their ability to learn from hierarchical features from images. Architectures like Alex Net, Vgg Net, Res Net, Inception Net, Efficient Net, Dense Net, Vgg16 and Xception Net.

4. Training and Testing:

- Training procedure: The selected architecture was trained using a predefined number of epochs where epoch represents one complete pass through the entire training dataset.
- Model Training: The training process consisted of iteratively feeding batches of preprocessed image data into the model and adjusting the model's parameters based on computed loss.
- Epochs: For each architecture, training was conducted over a fixed number of epochs. Early stopping criteria were employed to halt training if the validation performance did not improve over a predicted number of epochs.
- Model Evaluation: The performance of each model was assessed using standard metrics for classification tasks, such as accuracy, precision and confusion matrix.

5. User Interface: The frontend of the web application includes a user interface where users can interact with the system. This interface allows users to upload an image or access the webcam to capture an image for diagnosis.

CHAPTER 7

EXPERIMENTATION

Experimentation

7.1 Working:

This table represents the summary of pretrained eight different CNN architecture models with each architecture offering unique design principles and trade-offs in terms of model depth, parameter efficiency, and computational requirements. Thus, each network has a specific number of layers and parameters. The number of layers and the number of learnable parameters for each model, which provides insight into the number of filters used by the models. By considering this information, the study aims to provide comprehensive understanding on the architectural complexities and the capacity inherit in the models.

Table 7.1 pre-trained networks, number of convolutions layers summary.

Architecture	Layers	Trainable parameters
NAS Net	4	19.9M
Xception Net	5	4.91M
Alex Net	21	56.5M
Res Net	15	28.9M
VGG -16	16	138.4M
Inception Net	9	65.5M
Efficient Net	5	71.6M
Dense Net	17	20.2M

7.11 Dataset

Our project deals with image dataset that contains images of children between the age 2 to 6 years. The models are trained based on facial expression, behaviors and alignment. The project works by employing binary classification methodology where ‘0’ represents Non-Autistic Images and ‘1’ represents Autistic images. All the images considered for the experimentation are categorized as the two classes.

The dataset autism detection project is structured into four categories: training, testing, validation, and consolidation. The dataset consists of a total of 4250 images. Each image has been resized to dimensions of 224 pixels for the X-axis, 224 pixels for the Y-axis, and 3 channels. Where each category contains images categorized as autism or non-autism. The training dataset is utilized to train our CNN models, while the testing dataset is employed to evaluate their performance on unseen data. The validation dataset aids in fine-tuning hyperparameters and preventing overfitting during training, while the consolidation dataset may serve various purposes such as refining predictions or additional evaluation. This organization ensures a systematic approach to model development, allowing for robust training, accurate evaluation, and effective refinement of our autism detection models.





 consolidated	File folder
 test	File folder
 train	File folder
 valid	File folder

Figure 7.1 Dataset

7.12 Preprocessing of Data

For data preprocessing, several steps have been implemented. Firstly, image resizing has been applied to standardize the dimensions across all images, ensuring uniformity and facilitating computational efficiency during training. Using color contouring on facial images facilitates easier training of the image model because it enhances the contrast and delineates the facial features, making them more distinguishable.

```
# Iterate through each directory (assuming each directory represents a class)
for label in os.listdir(dataset_dir):
    label_dir = os.path.join(dataset_dir, label)
    # Iterate through each image in the directory
    for image_name in os.listdir(label_dir):
        img_path = os.path.join(label_dir, image_name)
        # Read and preprocess the image
        img = cv2.imread(img_path)
        if img is not None: # Check if the image was read successfully
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (224, 224))
            X.append(img)
            y.append(label) # Use directory name as label
        else:
            print(f"Warning: Unable to read image '{img_path}'")
```

The list into NumPy arrays representing the images have been converted into lists, likely to facilitate easier manipulation and compatibility with certain operations or libraries. Converting

list NumPy arrays gives indexing to the images that makes it easier for the model to assess and train on these indexed images.

```
# Convert lists to numpy arrays
X = np.array(X)
y = np.array(y)

y = label_encoder.fit_transform(y)
```

one-hot encoding has been employed to transform categorical labels into a format suitable for machine learning algorithms, particularly beneficial for multi-class classification tasks like autism detection.

```
# Convert labels to one-hot encoding
# Assuming you have two classes, if more, adjust num_classes accordingly
num_classes = 2 # Change it according to your dataset
y = to_categorical(y, num_classes=num_classes)

# Split dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

The dataset is split into train and test with the test size of 0.3 and 0.5 at random state=42 to obtain 2 classes.

All these preprocessing techniques helps prepare the data for input into the CNN models, ensuring consistency and compatibility while enhancing the effectiveness of subsequent training and evaluation processes.

7.13 Deep learning model for building and testing

After preprocessing of the data, the dataset is divided into training and testing subsets, typically with split of 70% for training and 30% for testing, to assess model performance on unseen data. Different CNN architectures are implemented and compared for their efficacy in classifying images related to autism spectrum disorder.

7.1.3.1 ResNet

The ResNet architecture achieved a validation accuracy of a 70% after training for 20 epochs. This indicates that the model correctly classified around 70% of the test data instance.

In the classification report below it can be observed that the precision, recall and F-1 score for the autistic class are 0.82, 0.80 and 0.63 respectively. While for the non-autistic class the precision, recall and F-1 score are 0.57, 0.26, and 0.36 respectively.

Precision of 0.52 for the Autistic class means that out of all instances predicted as Autistic, 52% were correctly classified, while a recall of 0.80 indicates that 80% of actual Autistic instances were correctly classified by the model. Similarly, a precision of 0.57 for the Nonautistic class means that out of all instances predicted as Nonautistic, 57% were correctly classified, while a recall of 0.26 indicates that only 26% of actual Nonautistic instances were correctly classified by the model. The accuracy and performance metrics demonstrate the model's ability to differentiate between Autistic and Nonautistic instances.

```

Epoch 14/20
91/91 [=====] - 1026s 11s/step - loss: 0.5132 - accuracy: 0.7397 - val_loss: 0.6996 - val_acc
Epoch 15/20
91/91 [=====] - 1035s 11s/step - loss: 0.5047 - accuracy: 0.7438 - val_loss: 0.8712 - val_accuracy: 0.5833
Epoch 16/20
91/91 [=====] - 1003s 11s/step - loss: 0.5011 - accuracy: 0.7510 - val_loss: 0.5196 - val_accuracy: 0.7396
Epoch 17/20
91/91 [=====] - 1001s 11s/step - loss: 0.5038 - accuracy: 0.7548 - val_loss: 0.6825 - val_accuracy: 0.6667
Epoch 18/20
91/91 [=====] - 1038s 11s/step - loss: 0.4883 - accuracy: 0.7572 - val_loss: 0.5115 - val_accuracy: 0.7917
Epoch 19/20
91/91 [=====] - 1029s 11s/step - loss: 0.4779 - accuracy: 0.7665 - val_loss: 0.9153 - val_accuracy: 0.5729
Epoch 20/20
91/91 [=====] - 1002s 11s/step - loss: 0.4732 - accuracy: 0.7672 - val_loss: 0.5998 - val_accuracy: 0.7083
4/4 [=====] - 8s 2s/step - loss: 0.6482 - accuracy: 0.6900
Test Accuracy: 0.6899999976158142
4/4 [=====] - 9s 2s/step

```

	precision	recall	f1-score	support
Autistic	0.52	0.80	0.63	50
Non_Autistic	0.57	0.26	0.36	50
accuracy			0.53	100
macro avg	0.54	0.53	0.49	100
weighted avg	0.54	0.53	0.49	100

Figure 7.2 Resnet Epoch training

7.1.3.2 Alex Net

Alex Net model achieved a training loss of 0.6019 and an accuracy of approximately 67.47% for 20 epochs. This indicates that, on average, around 67.47% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.5082 and an accuracy of 76%. The 76 of the validation data instances were correctly classified by the model during this epoch.

The phrase "Training complete in 122m 42s" indicates that the entire training process took 122 minutes and 42 seconds to complete. This information provides insights into the model's performance during the epoch of training.

```

train Loss: 0.5962 Acc: 0.6828
val Loss: 0.5086 Acc: 0.7500

Epoch 16/19
-----
train Loss: 0.6036 Acc: 0.6764
val Loss: 0.5088 Acc: 0.7500

Epoch 17/19
-----
train Loss: 0.6038 Acc: 0.6716
val Loss: 0.5086 Acc: 0.7700

Epoch 18/19
-----
train Loss: 0.6019 Acc: 0.6686
val Loss: 0.5095 Acc: 0.7700

Epoch 19/19
-----
train Loss: 0.6019 Acc: 0.6747
val Loss: 0.5082 Acc: 0.7600

Training complete in 122m 42s
Best val Acc: 0.770000

```

Figure 7.3 AlexNet Epochs training

7.1.3.3 DenseNet

In epoch 20 of the training process, the model achieved a training loss of 0.4335 and an accuracy of approximately 79.62%. This indicates that, on average, around 79.62% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.4224 and an accuracy of 78.12%. This suggests that, on average, around 78.12% of the validation data instances were correctly classified by the model during this epoch. Overall, this information provides insights into the model's performance during epoch 20, both in terms of training and validation accuracies.

```

Epoch 11: saving model to /content/drive/MyDrive/training_1/cp.ckpt
45/45 [=====] - 1406s 31s/step - loss: 0.4614 - accuracy: 0.7827 - val_loss: 0.3668 - val_ac
Epoch 12/20
45/45 [=====] - ETA: 0s - loss: 0.4474 - accuracy: 0.7914
Epoch 12: saving model to /content/drive/MyDrive/training_1/cp.ckpt
45/45 [=====] - 1414s 31s/step - loss: 0.4474 - accuracy: 0.7914 - val_loss: 0.4676 - val_accuracy: 0.7344 - lr: 2.0000e-04
Epoch 13/20
45/45 [=====] - ETA: 0s - loss: 0.4560 - accuracy: 0.7907
Epoch 13: saving model to /content/drive/MyDrive/training_1/cp.ckpt
45/45 [=====] - 1416s 31s/step - loss: 0.4560 - accuracy: 0.7907 - val_loss: 0.4566 - val_accuracy: 0.7812 - lr: 2.0000e-04
Epoch 14/20
45/45 [=====] - ETA: 0s - loss: 0.4566 - accuracy: 0.7855
Epoch 14: saving model to /content/drive/MyDrive/training_1/cp.ckpt
45/45 [=====] - 1424s 32s/step - loss: 0.4566 - accuracy: 0.7855 - val_loss: 0.4816 - val_accuracy: 0.7188 - lr: 2.0000e-04
Epoch 15/20
45/45 [=====] - ETA: 0s - loss: 0.4355 - accuracy: 0.8042
Epoch 15: saving model to /content/drive/MyDrive/training_1/cp.ckpt
45/45 [=====] - 1422s 32s/step - loss: 0.4355 - accuracy: 0.8042 - val_loss: 0.4490 - val_accuracy: 0.7812 - lr: 2.0000e-04
Epoch 16/20
45/45 [=====] - ETA: 0s - loss: 0.4335 - accuracy: 0.7962
Epoch 16: saving model to /content/drive/MyDrive/training_1/cp.ckpt
45/45 [=====] - 1420s 31s/step - loss: 0.4335 - accuracy: 0.7962 - val_loss: 0.4224 - val_accuracy: 0.7812 - lr: 2.0000e-04

```

Figure 7.4 DenseNet Epoch training

7.1.3.4 NaasNet

In epoch 8 of the training process, the model achieved a training loss of 0.6025 and an accuracy of approximately 73.89%. This indicates that, on average, around 73.89% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.5437 and an accuracy of approximately 74.49%. On an average, around 74.49% of the validation data instances were correctly classified by the model during this epoch.

```

Epoch 1/30
74/74 [=====] - 2208s 29s/step - loss: 1.1642 - accuracy: 0.4932 - val_loss: 0.9459 - val_accuracy: 0.5204
Epoch 2/30
74/74 [=====] - 2095s 28s/step - loss: 0.9913 - accuracy: 0.5668 - val_loss: 0.8132 - val_accuracy: 0.5646
Epoch 3/30
74/74 [=====] - 2154s 29s/step - loss: 0.8847 - accuracy: 0.6097 - val_loss: 0.7128 - val_accuracy: 0.6429
Epoch 4/30
74/74 [=====] - 2212s 30s/step - loss: 0.7974 - accuracy: 0.6599 - val_loss: 0.6439 - val_accuracy: 0.6633
Epoch 5/30
74/74 [=====] - 2084s 28s/step - loss: 0.7150 - accuracy: 0.6815 - val_loss: 0.6103 - val_accuracy: 0.6803
Epoch 6/30
74/74 [=====] - 2079s 28s/step - loss: 0.6641 - accuracy: 0.7020 - val_loss: 0.5863 - val_accuracy: 0.7007
Epoch 7/30
74/74 [=====] - 2100s 28s/step - loss: 0.6149 - accuracy: 0.7156 - val_loss: 0.5635 - val_accuracy: 0.7279
Epoch 8/30
74/74 [=====] - 2084s 28s/step - loss: 0.6025 - accuracy: 0.7389 - val_loss: 0.5437 - val_accuracy: 0.7449
Epoch 9/30
30/74 [=====>.....] - ETA: 19:48 - loss: 0.5262 - accuracy: 0.7646

```

Figure 7.5 NaasNet Epoch training

Confusion matrix measures the performance of the model by predicting the images over the true labelled images. The number of true non autistic that is correct prediction for non-autistic dataset that is $1.3e+02$. The number of false non autistic indicating non autistic data predicted as autistic is 21. The number of false autistics for this model is 90. The number of autistic data predicted as autistic is 57. $1.3e+02$ is true non autistic

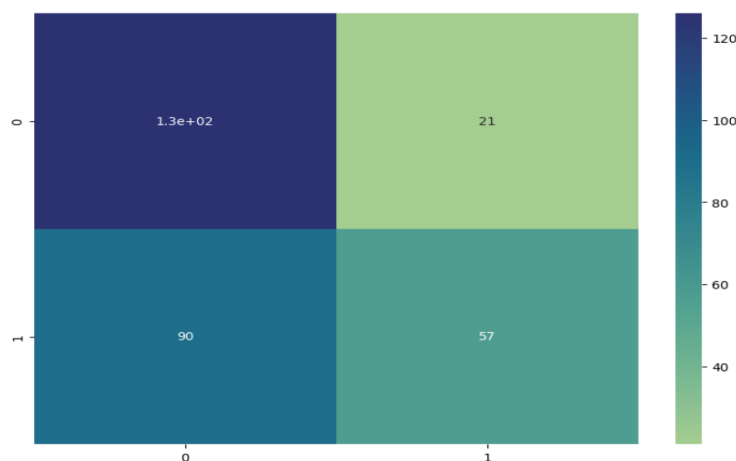


Figure 7.6 Confusion Matrix for NaasNet

7.1.3.5 XceptionNet

In epoch 11 of the training process, the model achieved a training loss of 0.1181 and an accuracy of approximately 95.66%. This indicates that, on average, around 95.66% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.5856 and an accuracy of approximately 74.83%. On an average, around 74.83% of the validation data instances were correctly classified by the model during this epoch.

```
Epoch 1/20
74/74 [=====] - 2216s 30s/step - loss: 1.0474 - accuracy: 0.5548 - val_loss: 0.9314 - val_accuracy: 0.5272
Epoch 2/20
74/74 [=====] - 2173s 29s/step - loss: 0.7040 - accuracy: 0.6841 - val_loss: 0.7854 - val_accuracy: 0.6429
Epoch 3/20
74/74 [=====] - 2170s 29s/step - loss: 0.5461 - accuracy: 0.7568 - val_loss: 0.7152 - val_accuracy: 0.6905
Epoch 4/20
74/74 [=====] - 2179s 29s/step - loss: 0.4431 - accuracy: 0.8070 - val_loss: 0.6420 - val_accuracy: 0.7007
Epoch 5/20
74/74 [=====] - 2173s 29s/step - loss: 0.3731 - accuracy: 0.8410 - val_loss: 0.6012 - val_accuracy: 0.7279
Epoch 6/20
74/74 [=====] - 2168s 29s/step - loss: 0.3068 - accuracy: 0.8690 - val_loss: 0.5877 - val_accuracy: 0.7313
Epoch 7/20
74/74 [=====] - 2182s 29s/step - loss: 0.2609 - accuracy: 0.8920 - val_loss: 0.5748 - val_accuracy: 0.7449
Epoch 8/20
74/74 [=====] - 2177s 29s/step - loss: 0.2106 - accuracy: 0.9107 - val_loss: 0.5668 - val_accuracy: 0.7517
Epoch 9/20
74/74 [=====] - 2150s 29s/step - loss: 0.1683 - accuracy: 0.9349 - val_loss: 0.5716 - val_accuracy: 0.7449
Epoch 10/20
74/74 [=====] - 2177s 29s/step - loss: 0.1491 - accuracy: 0.9396 - val_loss: 0.5764 - val_accuracy: 0.7449
Epoch 11/20
74/74 [=====] - 2158s 29s/step - loss: 0.1181 - accuracy: 0.9566 - val_loss: 0.5856 - val_accuracy: 0.7483
Epoch 12/20
12/74 [==>.....] - ETA: 29:28 - loss: 0.0965 - accuracy: 0.9740
```

Figure 7.7 XceptionNet Epoch training

Confusion matrix measures the performance of the model by predicting the images over the true labelled images. The number of true non autistic that is correct prediction for non-autistic dataset that is 1.2×10^2 . The number of false non autistic indicating non autistic data predicted as autistic is 30. The number of false autistics for this model is 27. The number of autistic data predicted as autistic is 1.2×10^2 . 1.2×10^2 is true non autistic

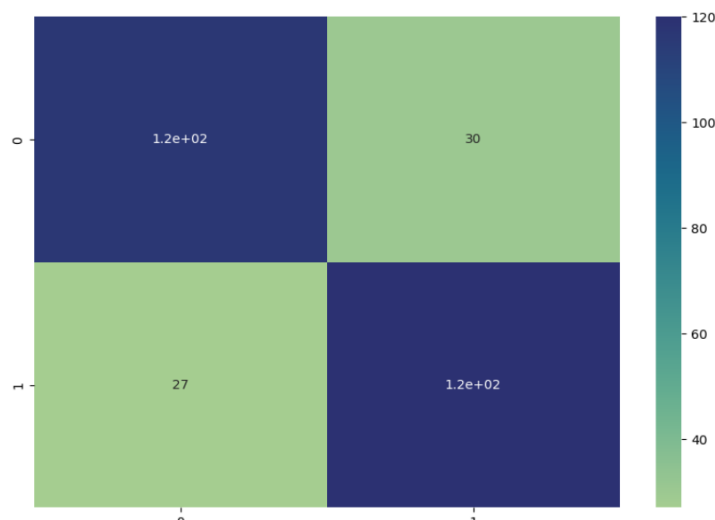


Figure 7.8 Confusion Matrix for XceptionNet

1.7.3.6 Efficient Net

In epoch 8 of the training process, the model achieved a training loss of 0.6058 and an accuracy of approximately 73.68%. This indicates that, on average, around 73.68% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.6446 and an accuracy of approximately 71.43%. This suggests that, on average, around 71.43% of the validation data instances were correctly classified by the model during this epoch.

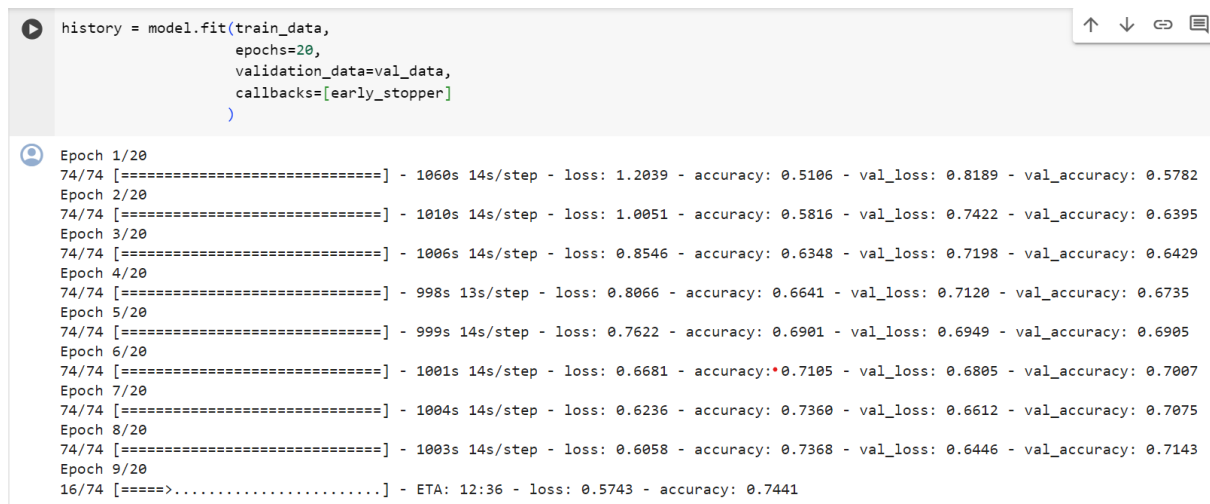


Figure 7.9 Efficient Net Epoch training

1.7.3.7 Inception Net

In epoch 30 of the training process, the model achieved a training loss of 0.0818 and an accuracy of approximately 97.12%. This indicates that, on average, around 97.12% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.8572 and an accuracy of approximately 79.69%. This suggests that, on average, around 79.69% of the validation data instances were correctly classified by the model during this epoch.

```

35/35 [=====] - 30s 852ms/step - loss: 0.1873 - accuracy: 0.9246 - val_loss: 0.3688 - val_accuracy: 0.8542
Epoch 20/30
35/35 [=====] - 30s 858ms/step - loss: 0.1785 - accuracy: 0.9329 - val_loss: 0.4389 - val_accuracy: 0.8438
Epoch 21/30
35/35 [=====] - 30s 838ms/step - loss: 0.1773 - accuracy: 0.9307 - val_loss: 0.5983 - val_accuracy: 0.8229
Epoch 22/30
35/35 [=====] - 30s 846ms/step - loss: 0.1510 - accuracy: 0.9356 - val_loss: 0.4767 - val_accuracy: 0.8542
Epoch 23/30
35/35 [=====] - 30s 856ms/step - loss: 0.1258 - accuracy: 0.9491 - val_loss: 0.5400 - val_accuracy: 0.8073
Epoch 24/30
35/35 [=====] - 30s 847ms/step - loss: 0.1327 - accuracy: 0.9478 - val_loss: 0.6571 - val_accuracy: 0.8125
Epoch 25/30
35/35 [=====] - 30s 849ms/step - loss: 0.1708 - accuracy: 0.9311 - val_loss: 0.4065 - val_accuracy: 0.8750
Epoch 26/30
35/35 [=====] - 30s 851ms/step - loss: 0.1177 - accuracy: 0.9531 - val_loss: 0.8614 - val_accuracy: 0.7760
Epoch 27/30
35/35 [=====] - 30s 846ms/step - loss: 0.1153 - accuracy: 0.9532 - val_loss: 0.7986 - val_accuracy: 0.7969
Epoch 28/30
35/35 [=====] - 30s 840ms/step - loss: 0.0845 - accuracy: 0.9680 - val_loss: 0.6140 - val_accuracy: 0.8281
Epoch 29/30
35/35 [=====] - 30s 848ms/step - loss: 0.1167 - accuracy: 0.9563 - val_loss: 0.5157 - val_accuracy: 0.8438
Epoch 30/30
35/35 [=====] - 30s 843ms/step - loss: 0.0818 - accuracy: 0.9712 - val_loss: 0.8572 - val_accuracy: 0.7969

```

7.10 InceptionNet Epoch training

The below graph represents the accuracy of training and validation loss upon epochs. The curve of the training loss shows the good trend and loss minimizes as the number of epochs increases. whereas the curve of validation loss shows noise around the training loss and validation loss tends to increase after 25 to 30 epochs. Validation loss fluctuates a lot.

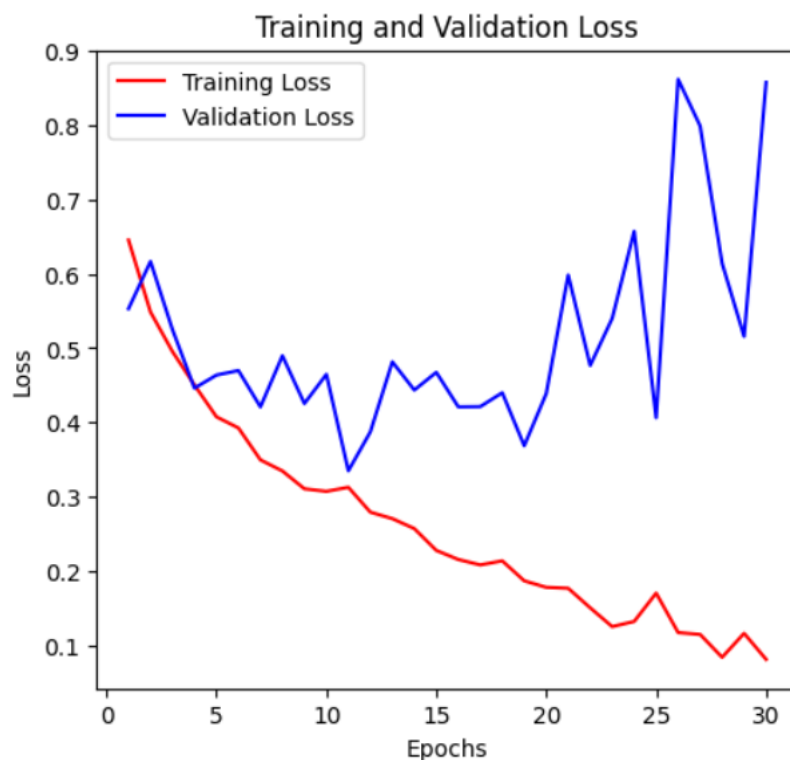


Figure 7.11 Training and validation Loss

The below graph represents the accuracy of training and validation upon epochs increases the accuracy of the training data increases from 0.65 to 0.85 whereas the validation accuracy shows lot of deflection as the number of epochs increases.

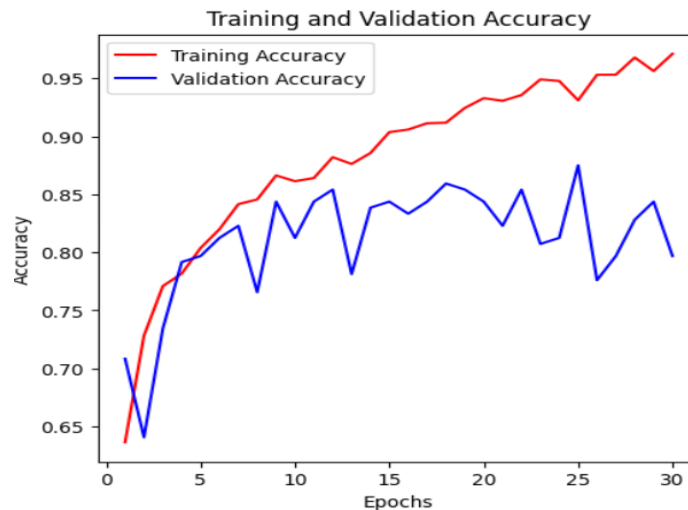


Figure 7.12 Training and validation accuracy

1.7.3.8 VGG 16

In epoch 20 of the training process, the model achieved a training loss of 0.5006 and an accuracy of approximately 74.92%. This indicates that, on average, around 74.92% of the training data instances were correctly classified by the model during this epoch. Additionally, on the validation dataset, the model achieved a validation loss of 0.4854 and an accuracy of approximately 75.00%. This suggests that, on average, around 75.00% of the validation data instances were correctly classified by the model during this epoch.

Epoch 8/20	719s	9s/step	loss: 0.5217	acc: 0.7376	val_loss: 0.4933	val_acc: 0.7500
Epoch 9/20	715s	9s/step	loss: 0.4868	acc: 0.7528	val_loss: 0.4862	val_acc: 0.7708
Epoch 10/20	720s	9s/step	loss: 0.5030	acc: 0.7472	val_loss: 0.4956	val_acc: 0.7500
Epoch 11/20	714s	9s/step	loss: 0.5062	acc: 0.7492	val_loss: 0.5212	val_acc: 0.7500
Epoch 12/20	715s	9s/step	loss: 0.4866	acc: 0.7588	val_loss: 0.4848	val_acc: 0.7188
Epoch 13/20	713s	9s/step	loss: 0.4879	acc: 0.7540	val_loss: 0.4914	val_acc: 0.8125
Epoch 14/20	712s	9s/step	loss: 0.4878	acc: 0.7528	val_loss: 0.4870	val_acc: 0.7708
Epoch 15/20	713s	9s/step	loss: 0.5011	acc: 0.7520	val_loss: 0.5198	val_acc: 0.7500
Epoch 16/20	711s	9s/step	loss: 0.4722	acc: 0.7699	val_loss: 0.4976	val_acc: 0.7292
Epoch 17/20	710s	9s/step	loss: 0.4900	acc: 0.7636	val_loss: 0.5324	val_acc: 0.7083
Epoch 18/20	729s	9s/step	loss: 0.4987	acc: 0.7552	val_loss: 0.5282	val_acc: 0.7083
Epoch 19/20	747s	9s/step	loss: 0.4972	acc: 0.7456	val_loss: 0.4761	val_acc: 0.7396
Epoch 20/20	742s	9s/step	loss: 0.5006	acc: 0.7492	val_loss: 0.4854	val_acc: 0.7500

Figure 7.13 VGG16 Epoch training

The below table explains how data is organized into columns representing different epochs (indexed from 0 to 15) and various performance metrics such as loss and accuracy for both the training and validation datasets. The "loss" column indicates the value of the loss function (e.g., categorical cross-entropy) calculated during each epoch. Lower values indicate better

performance. The "acc" column represents the accuracy achieved on the training dataset during each epoch, while "val_acc" represents the accuracy on the validation dataset. Similarly, higher accuracy values signify better model performance. Analysing the trends over epochs can provide insights into how the model's performance evolves during training.

	loss	acc	val_loss	val_acc
0	0.566206	0.706938	0.539233	0.781250
1	0.553547	0.711722	0.520341	0.770833
2	0.537694	0.716906	0.537232	0.729167
3	0.526629	0.733652	0.517915	0.729167
4	0.527585	0.729665	0.524255	0.739583
5	0.520902	0.728469	0.546041	0.718750
6	0.506999	0.747608	0.531905	0.750000
7	0.521709	0.737640	0.493284	0.750000
8	0.486839	0.752791	0.486177	0.770833
9	0.503000	0.747209	0.495553	0.750000
10	0.506220	0.749203	0.521174	0.750000
11	0.486561	0.758772	0.484794	0.718750
12	0.487863	0.753987	0.491355	0.812500
13	0.487835	0.752791	0.486965	0.770833
14	0.501067	0.751994	0.519838	0.750000
15	0.472232	0.769936	0.497602	0.729167

Table 7.2 validation loss and accuracy

This table presents a data frame containing predictions made by a model on a test dataset, alongside the corresponding filenames and true labels.

File Name: The filename of the test image. This column contains the names of the individual images that were used for testing.

Test Labels: The true labels of the test images. This column indicates whether each test image belongs to the "Autistic" class (1) or the "Nonautistic" class (0).

Predictions: The predictions made by the model for each test image. This column contains the model's predictions for the corresponding test images, where a prediction of 1 typically represents the "Autistic" class, and a prediction of 0 represents the "Nonautistic" class.

	filename	test_labels	predictions
0	Autistic.47.jpg	1	1
1	Non_Autistic.69.jpg	0	0
2	Non_Autistic.52.jpg	0	0
3	Autistic.92.jpg	1	1
4	Autistic.38.jpg	1	1
...
295	Autistic.25.jpg	1	1
296	Non_Autistic.22.jpg	0	0
297	Non_Autistic.89.jpg	0	1
298	Non_Autistic.83.jpg	0	0
299	Autistic.100.jpg	1	1

300 rows × 3 columns

Table 7.3 prediction of autistic or non autistic

The below graph represents the accuracy of training and validation upon epochs increases the accuracy of the training data increases from 64% to 80% whereas the validation accuracy shows lot of deflection as the number of epochs increases leading from 70% accuracy to 75% accuracy.

Text(0.5, 1.0, 'Training and Validation Accuracy')



Figure 7.14 Training and validation accuracy

The below graph represents the training and validation loss upon epochs. The curve of the training loss shows the good trend and loss minimizes as the number of epochs increases.

Whereas the curve of validation loss shows noise around the training loss and validation loss tends to increase after 15-17 epochs. Validation accuracy fluctuates a lot.

```
Text(0.5, 1.0, 'Training and Validation Loss')
```

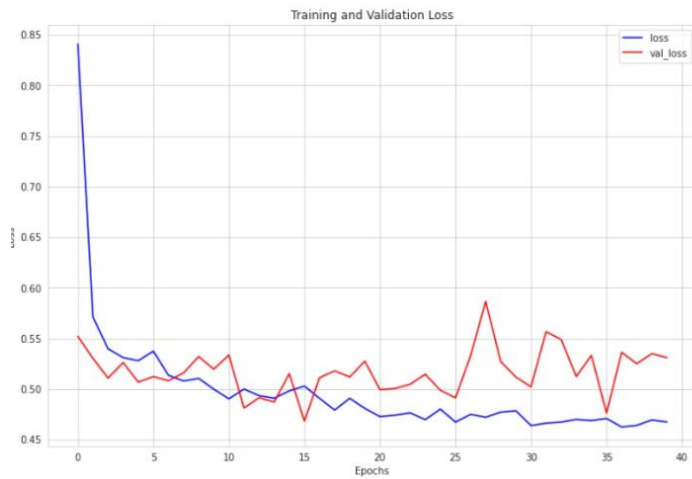


Figure 7.15 Training and validation Loss

Confusion matrix measures the performance of the model by predicting the images over the true labelled images. The number of true non-autistic that is correct prediction for non-autistic dataset that is 1.1×10^2 . The number of false non autistic indicating non-autistic data predicted as autistic is 40. The number of false autistics for this model is 45. The number of autistic data predicted as autistic is 1×10^2 . 1.1×10^2 is true non-autistic.

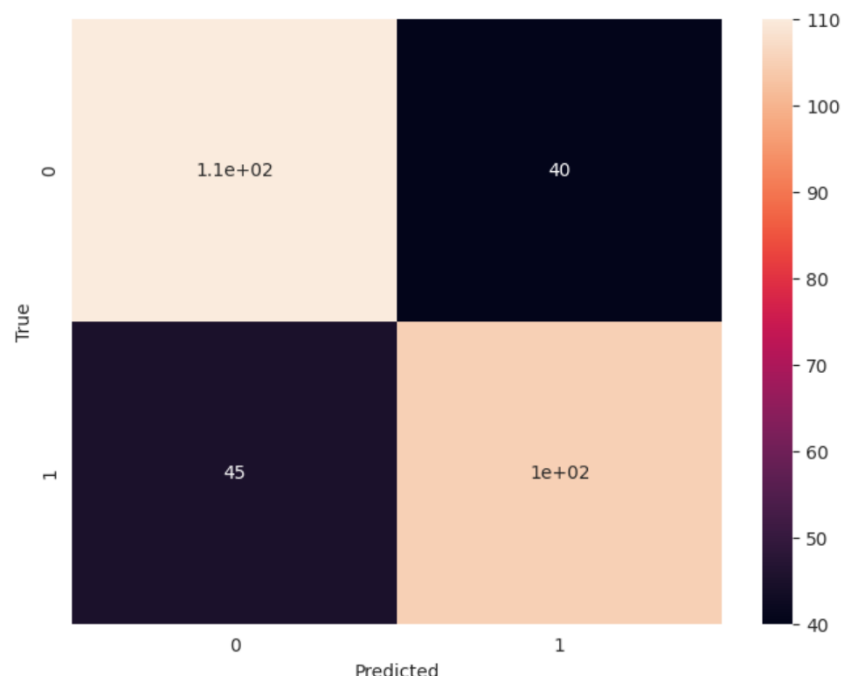


Figure 7.16 Confusion matrix for VGG 16

1.7.3.9 Training of the model:

Initially for the training of various models, image size of 224X224X3 were utilized with channel being 3. Image dataset with two classes and batch size of 34 were employed for model training. A constant of 20 epochs for all the models were maintained till the end of performance where each epoch represents entire dataset being passed through the model once. Thus, allowing the model to extract the features from input dataset and trained based on those patterns. All model were trained with a selective learning rate and all the frozen layers from the base models which later become unfrozen along with few custom additional layers were utilized for model training.

Table 7.4 comparison of different CNN model

Comparison between different models in terms of Training accuracy, Validation accuracy, Test loss

Model	Layers	Time/Epoch	Training Accuracy	Validation Accuracy	Test Loss
Nas Net	4	177s	0.755	0.50	0.7364
XceptionNet	5	38s	0.94	0.81	0.5192
Alex Net	21	122m	0.76	0.77	0.67
Res Net	15	991s	0.76	0.53	0.6482
VGG -16	16	720s	0.81	0.7500	0.50
Inception Net	9	30s	0.97	0.79	0.7806
Dense Net	17	23m 47s	0.7962	0.7812	0.44
Efficient Net	5	506ms	0.98	0.80	0.442

Models' comparison on ASD

On observation from the above table the study indicates that NASNet, AlexNet, ResNet, and DenseNet show accuracy score in the range of 75%-79% which is comparatively less than others models as number of layers various greatly thus models demanded increased epoch for better training. VGG16 being one of the best models in general yielded an accuracy score of 81% for the data and Xception model gave an accuracy of 94% which is considered as good performance. On contrary Inception model gave a great accuracy score of 97% however it's also the model with highest loss which of 78% that made the model unfit. Among all the

models, DenseNet produced very less loss of 44% due to its reduced parameters and it's skip technique between the densely-connected layers. Finally Efficient Net outperformed all the models with a Training accuracy of 98% and Validation accuracy of 80%.

7.1.3.10. USER INTERFACE

For this Machine learning models our project deals with integrating web development that enables user to discover whether the individual is autistic or non-autistic by either providing input image or accessing the webcam to capture an image. If the image is said to be autistic the web application provides description of autistic condition and further treatment that may be required.

The following components have been utilized in the development of the web application:

CNN Models: Various CNN models are been tested, such as ResNet, NaasNet, VGG16, XceptinNet, EfficientNet, AlexNet, DenseNet and Inception Net have been trained on a dataset of ASD images to classify them into autistic or non-autistic categories.

Web Development Framework: We have employed a web development framework, such as Flask, to build the backend of the web application. This framework enables us to handle user requests, process image inputs, and communicate with the machine learning models for classification.

User Interface: The frontend of the web application includes a user interface where users can interact with the system. This interface allows users to upload an image or access the webcam to capture an image for diagnosis.

Image processing: The uploaded images or images captured from the webcam are processed using image processing techniques to prepare them for input into the machine learning models.

Model Integration: The trained machine learning models are integrated into the web application backend to perform real-time classification of the input images. The predicted results are then returned to the frontend for display to the user. In our project, meticulously assessed the performance and accuracy of eight different convolutional neural network (CNN) models for classifying autism spectrum disorder (ASD) images, ultimately selecting three models to serve as the backend for our website. These chosen models, labeled VGG 16, NaasNet and

InceptionNet have demonstrated superior performance and accuracy in accurately categorizing ASD images. InceptionNet stands out for its exceptional accuracy and robustness, while NaasNet model provides mobile compatibility and VGG16 showcases balanced performance and accuracy, making it a reliable choice. Integrated into our website architecture, these CNN models will process user-provided images or webcam captures in real-time, allowing our website to deliver accurate ASD classification results and provide relevant information and support resources accordingly, thus ensuring a high-quality user experience.

Information Retrieve: Based on the classification results, the web application provides users with relevant information about the autistic condition and further treatment options that may be required. This information is displayed to the user to raise awareness and provide support.

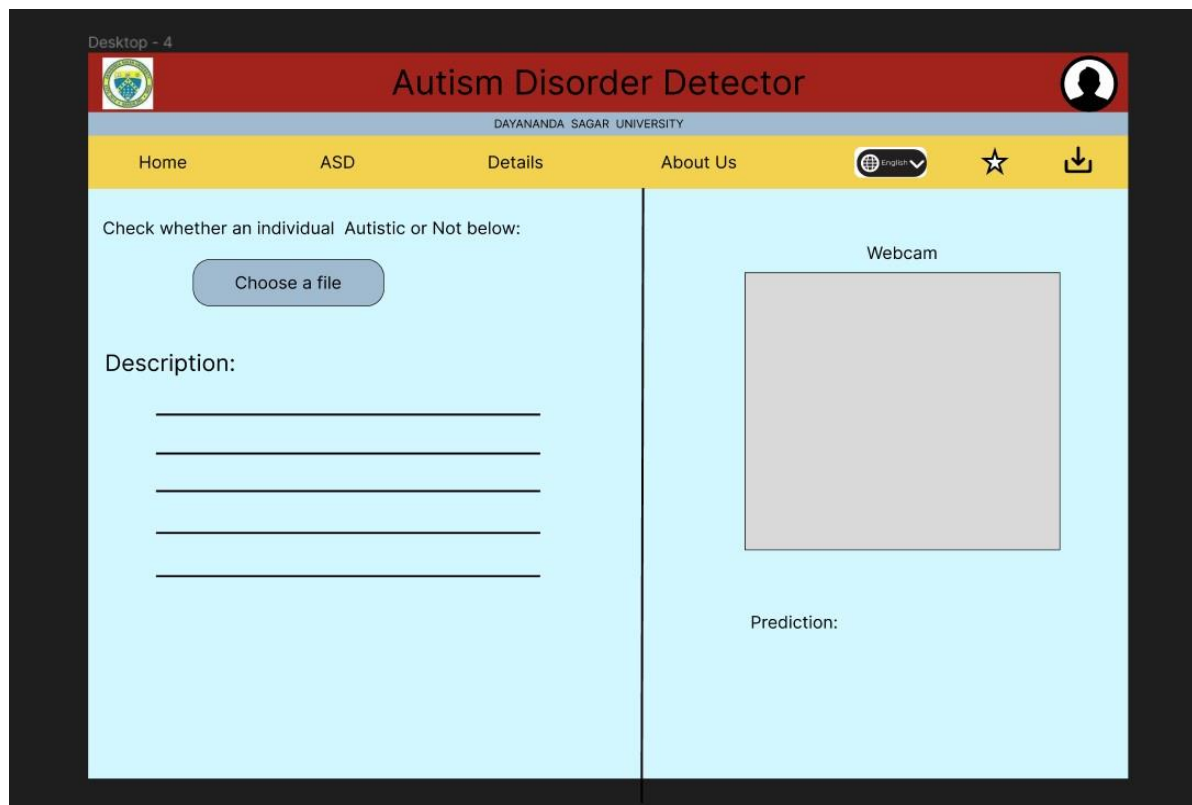


Figure 7.17 website

CHAPTER 8

RESULTS AND DISCUSSION

CHAPTER 8 RESULTS AND DISCUSSIONS

8.1 RESULTS

The ML model was able to predict the given input image as autistic or non-autistic as shown in the figure below. The resized images are subdivided into small grids in order to analyze the texture and expression of the images with trained knowledge to classify them as autistic or not. The figure shows the prediction made next to the file name at the top and all the images are accurately predicted by the model.

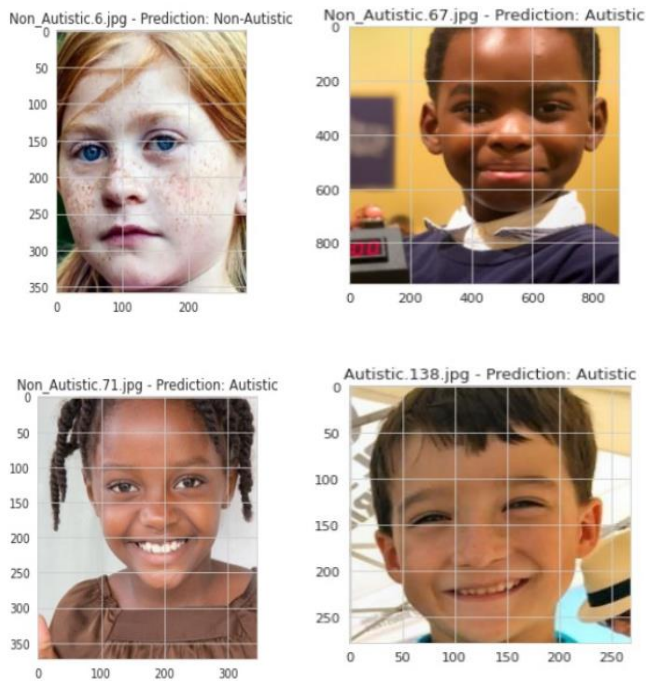


Figure 8.1 Autistic or non-autistic

Table 8.1 Comparison of the eight different CNN models based on their performances

Model	Layers	Time/Epoch	Training Accuracy	Validation Accuracy	Test Loss
Nas Net	4	177s	0.755	0.50	0.7364
XceptionNet	5	38s	0.94	0.81	0.5192
Alex Net	21	122m	0.76	0.77	0.67
Res Net	15	991s	0.76	0.53	0.6482
VGG -16	16	720s	0.81	0.7500	0.50
Inception Net	9	30s	0.97	0.79	0.7806
Dense Net	17	23m 47s	0.7962	0.7812	0.44
Efficient Net	5	506ms	0.98	0.80	0.442

On observation from the above table the study indicates that NASNet, AlexNet, ResNet, and DenseNet show accuracy score in the range of 75%-79% which is comparatively less than others models as number of layers varies greatly thus models demanded increased epoch for better training. VGG16 being one of the best models in general yielded an accuracy score of 81% for the data and Xception model gave an accuracy of 94% which is considered as good performance. On contrary Inception model gave a great accuracy score of 97% however it's also the model with highest loss which of 78% that made the model unfit. Among all the models, DenseNet produced very less loss of 44% due to its reduced parameters and its skip technique between the densely-connected layers. Finally Efficient Net outperformed all the models with a Training accuracy of 98% and Validation accuracy of 80%.

The below two figures represent the performance of the web application developed for this machine learning model that detects whether an individual is autistic or not and provide description on further treatment.

For this Machine learning models our project deals with integrating web development that enables user to discover whether the individual is autistic or non-autistic by either providing input image or accessing the webcam to capture an image. If the image is said to be autistic the web application provides description of autistic condition and further treatment that may be required. The web page allows users to input their image file in two different forms.

The former being the performance of webcam in the web application that enables user to capture the image and submit it. Upon evaluation the predicted output is displayed at the Prediction area.

The Latter being the section dedicated for the users to upload images by clicking choose the file button and the selected images displayed next to it for confirmation. Upon Submitting the prediction and Treatment are displayed in the Description area.

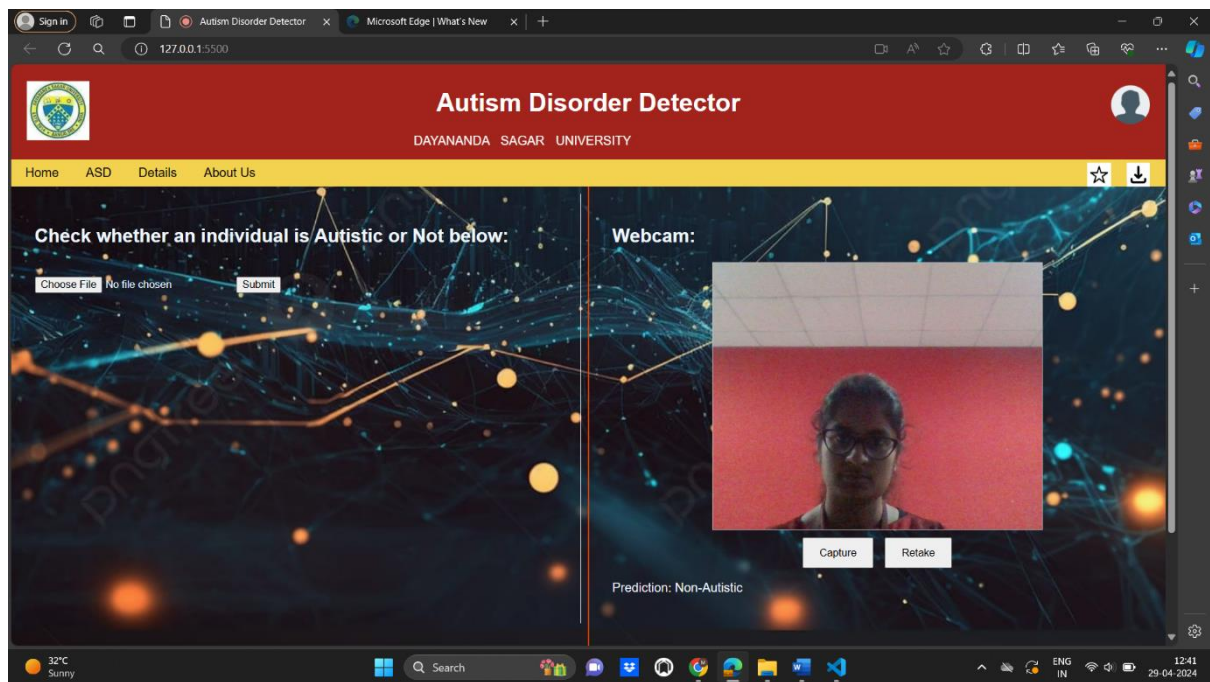


Figure 8.2 Website photo capture

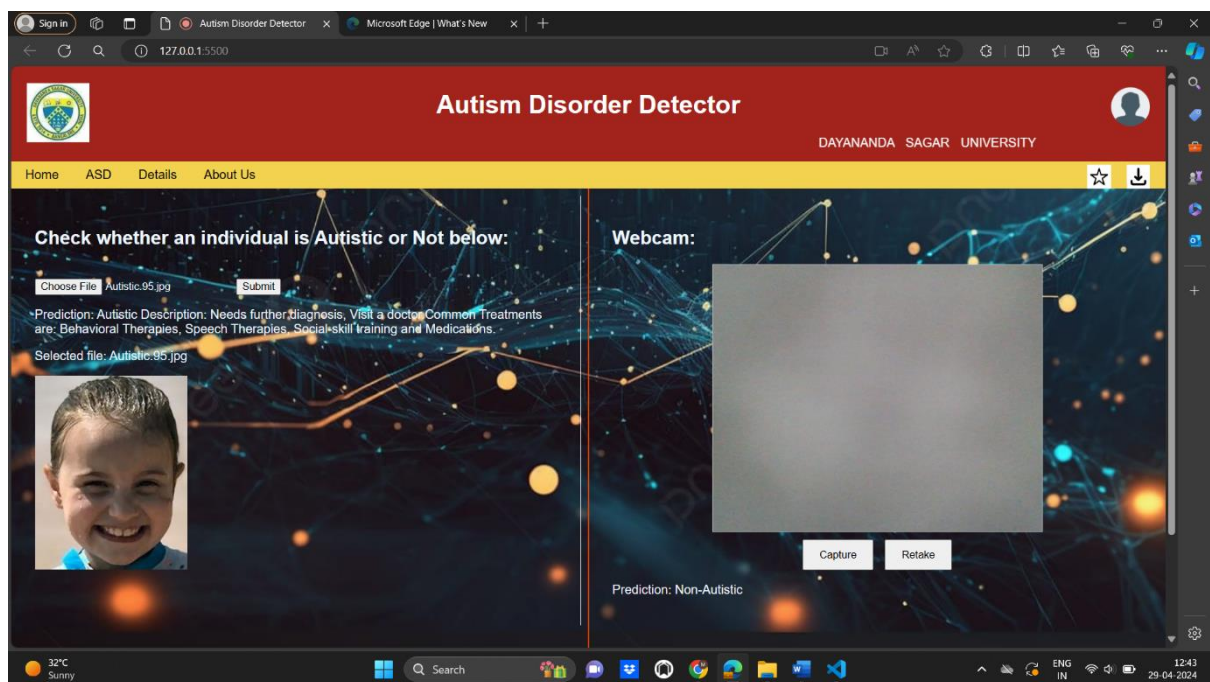


Figure 8.3 Website Image upload

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

The performance of well-established pre-trained models for the task of autism spectrum disorder detection was evaluated and the characteristics of these models, such as the relationship between resolution and performance, the number of CNN layers, the epochs, and the learning parameters, were investigated [24]. In conclusion, this project comprehensively evaluated the performance of eight Convolutional Neural Network (CNN) models, namely AlexNet, EfficientNet, ResNet, InceptionNet, XceptionNet, DenseNet, NASNet, across various metrics including Training and validation accuracy, Training and validation loss, and confusion matrix. Through rigorous experimentation and analysis, valuable insights were gained into the strengths and weaknesses of each model in handling complex image classification tasks. The results highlighted the diverse capabilities of these CNN architectures. Obtained that the EfficientNet, Xception and VGG 16 CNN architecture models provides the best accuracy for training and validating. This project contributes to the understanding of CNN model performance in image classification tasks and provides a basis for informed model selection based on specific requirements and constraints.

In addition to evaluating different CNN models, our project also delved with integration of these models into web application for real-world deployment. Building upon these findings, Xception and VGG16 were selected to be integrated into the web application as the backend models. Their superior performance and robustness make them well-suited for real-time ASD detection using visual cues. The web application interface, incorporates features such as webcam access and file upload functionality for input images, which enhance its usability and accessibility for clinicians and end-users.

The results demonstrate the successful implementation of our machine learning model for autism detection through image analysis. The model accurately predicts whether the input images depict autistic or non-autistic individuals. Utilizing small grid subdivisions, it analyzes textures and expressions to make precise classifications. Our web application further enhances accessibility, offering two convenient methods for users to interact with the model. The webcam feature allows real-time image capture, while the file upload option enables users to select images from their devices. Both methods provide prompt predictions, displayed alongside the uploaded or captured images. Additionally, the application offers insights into further treatment options based on the prediction, enhancing its practical utility for users.

Overall, these results highlight the efficacy and user-friendliness of our approach in autism detection and management.

Our project stands out in its approach to Autism disorder detection by leveraging the strengths of multiple CNN models and combining their performances to identify the best one. By meticulously tabulating the results of the top eight CNN models, we aim to achieve unparalleled accuracy and reliability in our detection system. This method allows us to capitalize on the unique strengths of each model, effectively mitigating biases and enhancing overall performance. Furthermore, our project goes beyond just the technical aspect by extending into the development of a user-friendly web interface for autism detection. This interface will make the detection process accessible to a wider audience, potentially revolutionizing the way autism is diagnosed and managed. Through this innovative combination of advanced technology and user-centric design, our project promises to make significant strides in autism detection and contribute to improving the lives of individuals affected by this disorder.

By leveraging advancements in both machine learning techniques and web development, this project not only contributes to the understanding of CNN model performance in image classification tasks but also paves the way for practical implementations in healthcare setting. Future research could explore further optimization techniques, ensemble methods, and transfer learning strategies to enhance the performance and generalization capabilities of CNN models in diverse application domains.

9.2. SCOPE FOR FUTURE WORK:

The future work could be the improvisation on speed while keeping the accuracy as high as possible and building a complete working software for the healthcare industries. Another thing could be that since only one language, i.e., English, has been used by us, it can be extended to other languages also for better reachability among the diverse population. Many other elements in the web application can be added to increase the user experience for clinicians and users.

REFERENCES

REFERENCES

- [1] Md. Fazle Rabbi¹, S. M. Mahedy Hasan², Arifa Islam Champa³, Md. Asif Zaman,” A Convolutional Neural Network Model for Early-Stage Detection of Autism Spectrum Disorder”. IEEE,2020
- [2] Nabila Saman, Jannatul Ferdus, Abdus Sattar, “Autism Spectrum Disorder Detection Using Machine Learning Approach”, IEEE,2021
- [3] Naouel Boughattas and Hanen Jabnoun, "Autism Spectrum Disorder (ASD) Detection Using Machine Learning Algorithms”, SPINGER,2022
- [4] Nastaran Mohammadian Rad "Applying Deep Learning to Stereotypical Motor Movement Detection in Autism Spectrum Disorders", international conferences of IEEE,2017
- [5] F. Catherine Tamilarasi Dr. J. Shanmugam “Convolutional Neural Network based Autism Classification”, IEEE,2020
- [6]"Google Colaboratory," [Online]. Available: colab.research.google.com.
- [7] Anshu Sharma and Dr.Poonam Tanwar “Deep Analysis of Autism Spectrum Disorder Detection Techniques”ICIEM,2020
- [8] Dadang Eman and Andi W.R Emanuel “Machine Learning Classifiers for Autism Spectrum Disorder: A Review” ICITSEE, 2019
- [9] A. S. Heinsfeld, A. R. Franco, R. C. Craddock, A. Buchweitz and F. Meneguzzi, “Identification of autism spectrum disorder using deep learning and the ABIDE dataset.” *Neuro Image. Clinical* vol. 17 16- 23. 30 Aug. 2017
- [10] Sajeev Ram Arumugam, Sankar Ganesh Karuppasamy” A Deep Convolutional Neural Network based Detection System for Autism Spectrum Disorder in Facial images” IEEE,2021
- [11] W. Jamal, S. Das, and I. Oprescu, “Classification of autism spectrum disorder using supervised learning of brain connectivity measures extracted from synchro states,” *J. Neural Eng.*, vol. 046019, 2014.
- [12] X. Bi, Y. Wang, Q. Shu, Q. Sun, and Q. Xu, “Classification of Autism Spectrum Disorder Using Random Support Vector Machine Cluster,” *Front. Genet.*, vol. 9, no. FEB, p. 18, Feb. 2018.
- [13] Zeinab Sherkatghanad, Mohammadsadegh Akhondzadeh, Soorena Salari, Mariam ZomorodiMoghadam, Moloud Abdar, U Rajendra Acharya, Reza Khosrowabadi, and Vahid Salari, “Automated detection of autism spectrum disorder using a convolutional neural network,” *Frontiers in neuroscience*, vol. 13, 2020.

-
- [14] Rajat Mani Thomas, Selene Gallo, Leonardo Cerliani, Paul Zhutovsky, Ahmed El-Gazzar, and Guido van Wingen, "Classifying autism spectrum disorder using the temporal statistics of resting-state functional mri data with 3d convolutional neural networks," *Frontiers in psychiatry*, vol. 11, 2020.
- [15] Catherine Lord, Susan Risi, Linda Lambrecht, Edwin H Cook, Bennett L Leventhal, Pamela C DiLavore, andrew Pickles, and Michael Rutter, "The autism diagnostic observation schedule—generic: A standard measure of social and communication deficits associated with the spectrum of autism," *Journal of autism and developmental disorders*, vol. 30, no. 3, pp 2000.
- [16] A. S. Heinsfeld, A. R. Franco, R. C. Craddock, A. Buchweitz and F. Meneguzzi, "Identification of autism spectrum disorder using deep learning and the ABIDE dataset." *NeuroImage. Clinical* vol. 17 16- 23. 30 Aug. 2017.
- [17] Y. Song, T. M. Epalle and H. Lu, "Characterising and predicting autism spectrum disorder by performing resting-state functional network community pattern analysis," 2019.
- [18] F. N. Büyükoflaz and A. Öztürk, "Early autism diagnosis of children with machine learning algorithms," 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2018, pp. 1-4.
- [19] A. Sharma, A. Khosla, M. Khosla and Y. Rao, "Fast and Accurate Diagnosis of Autism (FADA): a novel hierarchical fuzzy system-based autism detection tool." *Australasian physical & engineering sciences in medicine* vol. 41,3 (2018).
- [20] "Early Diagnosis of Autism Disease Based on Deep Convolutional Neural Networks and fMRI Imaging" by Chen et al. 2019, IEEE
- [21] "A Deep Learning Approach for Autism Spectrum Disorder Detection from fMRI Data" by Zhou et al. 2019, *Frontiers in Neuroscience*
- [22] "Automatic Detection of Autism Spectrum Disorder in Children's MRI Data Using Deep Learning and the Importance of Normalized Input" by Kassani et al. 2020, *Sensors (MDPI)*
- [23] "CNN-Based Framework for Autism Spectrum Disorder Diagnosis from fMRI Data: Abnormality-Aware and Transferable Feature Learning" by Sarraf and Tofighi. 2016, *Scientific Reports (Nature Publishing Group)*.
- [24] "Autism Spectrum Disorder Detection from MRI Data Using Convolutional Neural Networks" by Thabtah et al.2019, IEEE
- [25] "Identification of Autism Spectrum Disorder Using Deep Learning and the ABIDE Dataset" by Heinsfeld et al.2018, *NeuroImage*.

- [26] "Multi-Modal Deep Learning Models for Autism Spectrum Disorder Diagnosis Using Magnetic Resonance Imaging Data" by Wang et al. 2023, IEEE
- [27] "A Deep Learning Approach for Predicting Autism Spectrum Disorder from Structural MRI Data" by Tan et al.2021, IEEE.
- [28] "Autism Spectrum Disorder Prediction by Facial Recognition Using Deep Learning " by Kanimozhi and Dhanasri 2024, IJCRT
- [29] "Autism spectrum disorder detection using facial images: A performance comparison of pretrained convolutional neural networks " by Israr Ahmad and Rashid 2024, IET