

*A project report on*

# **AUTISM SPECTRUM DISORDER DETECTION USING DEEP LEARNING**

*Submitted in partial fulfillment for the award of the degree of*

*Bachelor of Technology*

**Electronics and Communication Engineering (Core)**

*By*

**20BEC7002**

**Chadalawada Sree Venkateswarlu**



**SCHOOL OF ELECTRONICS  
ENGINEERING**

May, 2024

*A project report on*

# **AUTISM SPECTRUM DISORDER DETECTION USING DEEP LEARNING**

*Submitted in partial fulfillment for the award of the degree of*

*Bachelor of Technology*

## **ELECTRONICS AND COMMUNICATION ENGINEERING**

*by*

**CHADALAWADA SREE VENKATESWARLU (20BEC7002)**



**SCHOOL OF ELECTRONICS  
ENGINEERING**

Month, Year

May, 2024

## **DECLARATION**

I hereby declare that the thesis entitled “AUTISM SPECTRUM DISORDER DETECTION USING THE DEEP LEARNING” submitted by me, for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering VIT is a record of Bonafede work carried out by me under the supervision of Dr. Sucharitha. M.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Amaravati

Date: May, 2024

Signature of the Candidate

Chadalawada Sree Venkateswarlu

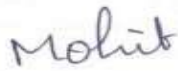
## CERTIFICATE

This is to certify that the Senior Design Project titled “**AUTISM SPECTRUM DISORDER DETECTION USING DEEP LEARNING**” that is being submitted by **Sree Venkateswarlu (20BEC7002)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of Bonafede work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.



Dr. Sucharitha. M  
Guide

**The thesis is satisfactory / unsatisfactory**



**Internal Examiner –I**



**Internal Examiner – II**



**Internal Examiner-III**

**Approved by**



**PROGRAM CHAIR**  
ECE Core



**DEAN**  
School Of Electronics  
Engineering

## ABSTRACT

Autism Spectrum Disorder (ASD) is a developmental condition that affects how people interact socially, communicate, and behave. Identifying ASD early and starting treatment as soon as possible can greatly improve the quality of life for those affected. Recently, deep learning models have become valuable in detecting ASD, utilizing detailed information from medical images to support the diagnostic process.

This project, titled "Detecting ASD with Deep Learning Models," introduces a cutting-edge method to automatically identify Autism Spectrum Disorder by analyzing medical images. It examines the effectiveness of several advanced deep learning architectures, such as VGG-16, Inception Net, EfficientNet-B0, EfficientNet-B7, and a custom hybrid model built on EfficientNet-B0.

In the initial phase of this project, the dataset—consisting of images of individuals both with and without ASD—is prepared. To increase the variety and reliability of the dataset, various data augmentation techniques are used. These include rotating, shifting, shearing, flipping, and zooming the images.

Subsequently, several deep learning models are implemented and evaluated. The VGG-16 model, known for its deep architecture, is adapted and fine-tuned to classify ASD based on image features. Similarly, the Inception Net and Efficient Net architectures are utilized, leveraging their ability to capture intricate patterns in images while mitigating computational complexity.

Additionally, the project introduces a custom hybrid model that enhances the EfficientNet-B0 base model with extra classification layers designed specifically for detecting ASD. This hybrid model aims to achieve the highest possible accuracy and efficiency in classifying ASD. The experimental results show that the proposed deep learning models are highly effective at accurately identifying individuals with ASD.

To evaluate the performance of each model, metrics such as accuracy, precision, recall, and F1-score are used, highlighting the strengths and weaknesses of each approach. Overall, this project advances the use of sophisticated machine learning techniques in healthcare, especially for neurodevelopmental disorders like ASD.

By automating the diagnostic process, these deep learning models can improve clinical workflows, reduce diagnostic errors, and enable timely interventions, ultimately enhancing the quality of life for individuals with ASD and their families.

## ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. M. Sucharitha. Associate Professor, School of Electronics Engineering, VIT-AP, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Learning.

I would like to express my gratitude to Chancellor, VPs, VC, and Dr. Umakanta Nanda, Dean, School of Electronics Engineering, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Jayendra Kumar. Associate Professor, Program Chair (ECE Core), all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Amaravati

Date: May, 202

Name of the student  
**SREE VENKATESWARLU**

## Index:

I.	Introduction -----	9
II.	Literature Survey -----	10 - 13
III.	Objective -----	14
IV.	Convolution Neural Networks	
V.	Methodology -----	15 - 29
	1. Data Pre-Processing	
	2. Data Validation	
	3. Data Argumentation	
	4. Data Normalization	
	5. Introduction to Convolution Neural Network	
	6. VGG-16	
	7. InceptionNet	
	8. Efficient Net B0	
	9. Efficient Net B7	
	10. Modified Efficient Net B0	
	11. Architectural Comparison	
	12. Activation Functions	
VI.	Project Work Flow -----	29
VII.	Architecture Work Flow of Models -----	30 - 32
	1. VGG-16 Architecture Work Flow	
	2. Inception Net Architecture Work Flow	
	3. Efficient Net Architecture Work Flow	
VIII.	Model Compilation -----	33
IX.	Pre-Trained Models -----	34
X.	Transfer Learning Techniques -----	35
XI.	Model Fitting and Batch Size -----	36
XII.	Results -----	38 – 41
XIII.	References -----	41

#### XIV. List of Tables:

1. Data Argumentation Techniques
2. Data Normalization Techniques
3. Convolution Techniques Used
4. Architectural Comparison
5. Impact of Proposed Model
6. Activation Functions Used in Output Layers
7. Description of Activation Functions
8. Optimizer Algorithms for each Model
9. Transfer Learning Techniques
10. Batch Size for each model
11. Model Accuracy Comparison Table

#### XV. List of Figures

1. Work Flow of the Project
2. VGG-16 Architecture Flow
3. Inception Net Architecture Flow
4. Efficient Net Architecture Flow
5. Modified Efficient Net Work Flow
6. VGG-16 Training Accuracy and Validation Accuracy Graph
7. VGG-16 Training Loss and Validation Loss Graph
8. Inception V3 Training Accuracy and Validation Accuracy Graph
9. Efficient Net B0 Training Accuracy and Validation Accuracy Graph
10. Efficient Net B0 Training Loss and Validation Loss Graph
11. Efficient Net B7 Training Accuracy and Validation Accuracy Graph
12. Efficient Net B7 Training Loss and Validation Loss Graph
13. Modified Efficient Net B0 Training Accuracy and Validation Accuracy Graph
14. Modified Efficient Net B0 Training Loss and Validation Loss Graph
15. Real-Time Capture Image and Output
16. Real-Time Search for ASD and Output Image



## Introduction

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental condition marked by ongoing challenges in social interaction, communication, and repetitive or restricted behaviors. Although the exact causes of ASD are not fully known, research suggests it arises from a mix of genetic, environmental, and neurological factors. Possible contributors include genetic tendencies, prenatal influences like maternal infections or exposure to harmful substances, and early brain development issues.

ASD is usually diagnosed in early childhood, often by the age of two or three, although some children may be diagnosed later. Early detection is essential for providing interventions that can significantly enhance outcomes for those with ASD. However, the timing of diagnosis can vary, depending on factors like the severity of symptoms and the availability of healthcare services.

While ASD cannot be cured, there are numerous treatments and interventions designed to help manage symptoms and support those affected. These can include behavioral therapies, speech and language therapy, occupational therapy, and medications to address specific issues such as anxiety or aggression.

Globally, the prevalence of ASD has been on the rise, with official government records reflecting this trend. In India, where healthcare infrastructure and awareness of ASD have been growing, the reported prevalence rates have increased in recent years. According to government data, the prevalence of ASD in India has risen steadily, paralleling global trends. Efforts to enhance awareness, improve diagnostic capabilities, and expand support services for individuals with the ASD and their families they undergoing.

The application of deep learning (DL) models in ASD detection offers was promising avenue for early identification & intervention. DL models utilize advanced machine learning techniques to analyze medical images and identify patterns indicative of ASD. By automating the diagnostic process, DL models can assist healthcare professionals in accurately and efficiently identifying individuals with ASD, facilitating timely intervention and support.

The accuracy of DL models in ASD detection is paramount for their clinical utility. Several state-of-the-art DL architectures, including VGG-16, Inception Net, and Efficient Net B0 and B7, have been evaluated for their efficacy in ASD classification. Notably, the hybrid EfficientNet B0 model has achieved the highest accuracy rate of 94.82% in assessment. Understanding the performance of these models is crucial for informing clinical decision-making and guiding future research endeavors aimed at improving ASD detection and intervention strategies.

## Literature Survey

In the ongoing pursuit of reliable methods for detecting (ASD) in the facial images, researchers have turned their attention to a vast array of machine learning and deep learning techniques. This review delves deeper into the methodologies and outcomes of three pivotal studies in this field, illuminating the advancements and challenges we face in detecting ASD.

### A Deep Convolutional Neural Network-Based System for Detecting Autism Spectrum Disorder from Facial Images (Study 1)

The first study, titled "A Deep Convolutional Neural Network based Detection System for Autism Spectrum Disorder in Facial Images," ventures into the machine learning. It explores a spectrum of powerful techniques ASD detection through facial image analysis. These techniques include:

- Deep Neural Network (DNN) Classifier: This classifier utilizes a multi-layered artificial neural network architecture to analyze and classify facial features potentially associated with ASD.
- (LSTM) Network: This LSTM is adept at handling sequential data, making it potentially valuable in analyzing subtle temporal dynamics of facial expressions that might be indicative of ASD.
- Random Forest Classifier: This classifier combines the predictive power of multiple decision trees, improving the overall accuracy and robustness of the detection system.
- Support Vector Machine (SVM) Classification: This technique identifies hyperplanes in multidimensional space that effectively separate data points belonging to different classes (ASD and non-ASD in this case).
- Artificial Neural Network (ANN) Classifier: Similar to DNN classifiers, ANNs are inspired by the structure of brain and also function of the human brain, enabling them to learn complex patterns within facial images that might be linked to ASD.

By leveraging this diverse toolbox of machine learning algorithms, the study achieved an impressive accuracy of 85.3% in detecting ASD from facial images. Additionally, the F-Static Score, which measures the balance between precision and recall, reached a promising value of 20.85. These results suggest that the proposed system has the potential to effectively identify individuals with ASD based on facial features.

### Autism Spectrum Disorder Detection in Children Based Transfer Learning Technique (Study – 2)

The second study, titled "Autism Spectrum Disorder Detection in Children Using Transfer Learning Techniques," explores the efficacy of DL models in ASD detection. Specifically, it investigates the power of transfer learning, a technique where a pre-trained model on a large dataset is fine-tuned for a new, specific task like ASD detection in children. This approach leverages the existing knowledge encoded within the pre-trained model, potentially accelerating the development of accurate detection systems for this specific population.

The study investigates the performance of several pre or implemented trained deep learning models, including Efficient Net B7, VGG-16, VGG-19, EfficientNet-B3, and Efficient Net B5, when applied to ASD detection in children using facial images. The results showcase a range of accuracies, varying from 46.50% to a highly promising 87.50%. These findings highlight the immense potential transfer learning for ASD detection, particularly in paediatric populations. It suggests that by adapting existing, powerful deep learning models, researchers can potentially develop accurate and efficient diagnostic tools for children with ASD.

However, a significant disparity in accuracy across the different models is observed. This emphasizes the importance of selecting appropriate pre-trained models and fine-tuning them effectively for the specific task of ASD detection in children. Further research is needed to optimize these techniques and achieve consistently high accuracy across diverse datasets.

### Autistic Spectrum Disorder and Screening: Prediction with Machine Learning Models (Study 3)

The third study, titled "Autistic Spectrum Disorder Screening: Prediction with Machine Learning Models," delves into the realm of ML algorithms and their potential for ASD screening. It focuses on the predictive capabilities of several established algorithms, including:

- **Decision Tree:** This algorithm employs a tree-like structure with branching conditions based on specific features. In this context, the features could be extracted from facial images or other data sources relevant to ASD diagnosis. By following the decision tree's branches, the model arrives at a prediction of ASD presence or absence.
- **Random Forest:** Similar to Study 1, this study utilizes the Random Forest algorithm. As mentioned before, it combines the predictions of multiple decision trees, leading to improved overall accuracy and robustness.
- **Logistic Regression:** This algorithm establishes a mathematical relationship between independent variables (facial features, for instance) and the dependent variable (presence or absence of ASD). By analyzing this relationship, the model can predict the likelihood of an individual having ASD.
- **Support Vector Machine (SVM):** As mentioned in Study 1, SVMs are powerful tools for classification tasks. They can be employed here to effectively distinguish between individuals with and without ASD based on the extracted data.

This study investigated the performance of these algorithms in predicting ASD using various datasets. These findings underscore the valuable role that machine learning models can play in the early detection and screening of Autistic Spectrum Disorder, highlighting their potential to enhance diagnostic accuracy and support clinical decision-making.

### Exploring Autism Spectrum Disorder Detection through Machine Learning (Study - 4)

(ASD) is a developmental condition known for difficulties in social interaction and communication. Although signs usually emerge within the first two years of life, ASD can be identified at any age. Diagnosis commonly relies on behavioral evaluations conducted by trained experts.

This review explores the potential of machine learning (ML) techniques for ASD prediction and analysis across different age groups (children, adolescents, and adults). The increasing application of ML in medical diagnosis motivates this investigation, with the aim of potentially streamlining or complementing traditional assessment methods.

Several prominent ML algorithms are considered for this purpose, including:

- **Naïve Bayes:** This probabilistic classifier is known for its simplicity and efficiency, making it a good candidate for initial exploration.

- Support Vector Machine (SVM): This powerful classification algorithm is adept at identifying patterns in high-dimensional data, potentially useful for analyzing complex diagnostic features.
- Logistic Regression: This technique establishes a mathematical relationship between independent variables and a binary dependent variable (ASD presence or absence). Analyzing this relationship allows for prediction of ASD likelihood.
- K-Nearest Neighbors (KNN): This method categorizes data points by comparing them to similar examples with known labels in the training dataset.
- Neural Network (NN): Inspired by the structure and function of the human brain, NNs can learn complex patterns within data, potentially uncovering hidden relationships relevant to ASD diagnosis.
- Convolutional Neural Network (CNN): A specialized type of NN particularly adept at image recognition, potentially valuable if the analysis incorporates facial image data.

The effectiveness of these techniques will be evaluated on three publicly available datasets:

- Dataset 1: ASD Screening in Children (292 instances, 21 attributes)
- Dataset 2: ASD Screening in Adults (704 instances, 21 attributes)
- Dataset 3: ASD Screening in Adolescents (104 instances, 21 attributes)

## **Objective**

The objective of the project "ASD Detection using Deep Learning Models" is to develop and evaluate advanced deep learning (DL) architectures for the accurate and efficient identification of Autism Spectrum Disorder (ASD) from medical images. This project focuses on leveraging state-of-the-art DL models to enhance early diagnosis and intervention, ultimately aiming to improve outcomes for individuals with ASD.

The first goal is to compile a comprehensive dataset of medical images from individuals with and without ASD. This dataset will undergo data augment techniques such as rotation, shifting, shearing, flipping, and zooming to increase its diversity and robustness, thereby reducing the risk of overfitting during model training.

Several prominent deep learning models will be adapted and fine-tuned for ASD classification, including VGG-16, Inception Net, and EfficientNet (B0 and B7). Each model's computational efficiency and accuracy will be assessed, considering factors such as training time, memory usage, and resource requirements. A custom hybrid model based on the EfficientNet-B0 architecture will be developed, integrating additional classification layers specifically tailored for ASD detection. This hybrid model aims to maximize both accuracy and computational efficiency, making it suitable for practical clinical & xapplications integrated limited computational resources.

Performance evaluation will utilize key metrics such same as accuracy, Theprecision, Therecall, and F1-score to determine each model's effectiveness in identifying ASD. A comparative analysis will identify the most efficient and accurate models, providing insights into their strengths and limitations for clinical use. The project will highlight the computational advantages of the hybrid EfficientNet-B0 model, which achieves highest accuracy (94.82%) But using fewer computational resources compared to

other models. This emphasis on computational efficiency is crucial for real-world clinical settings, where resource constraints can impact the feasibility of deploying advanced DL models.

Ultimately, the project aims to facilitate the adoption of DL models in healthcare by demonstrating their potential to automate and streamline the ASD diagnostic process, reducing diagnostic errors and enabling timely interventions. This work contributes to ongoing efforts to improve diagnostic capabilities and support & services for every child with ASD and their families, providing a foundation for future research aimed at enhancing ASD detection and intervention strategies through advanced machine learning techniques.

## **Convolution Neural Networks (CNN)**

### **Introduction to CNNs**

(CNNs) are a type of advanced deep learning model inspired by the structure and function of the human visual system. They excel at tasks like recognizing images, classifying objects, and performing other computer vision tasks because they can learn complex patterns and features from visual data in a hierarchical manner.

### **Neurons and Layers in CNNs**

CNNs are centered around neurons, which are the basic computational units responsible for receiving input signals, conducting transformations (usually weighted sums), and generating output signals. These neurons are structured into layers, creating a network that analyzes input data through various stages of feature extraction and abstraction. CNNs consist of different types of layers, each serving a distinct function within the network architecture.

### **Convolutional Layers**

Convolutional layers are the backbone of CNNs, providing the groundwork for their operations. In these layers, convolution operations are applied to input images using adaptable filters or kernels. These filters move across the input image, computing dot products to generate feature maps containing spatial information. Convolutional layers are pivotal in detecting local features such as edges, textures, and shapes, which are essential for understanding and interpreting visual data.

### **Pooling Layers**

Following convolutional layers, pooling layers come into play to downsample the feature maps, shrinking their spatial dimensions while retaining crucial features. This downsampling aids in reducing the computational load of the network and enhances its ability to recognize patterns regardless of their exact location. Pooling can be achieved through various methods, such as max pooling, which picks the highest value in a pooling window, or average pooling, which calculates the mean value.

Overall, pooling layers contribute significantly to the network's capacity to identify objects irrespective of their precise placement within the image.

### Fully Connected Layers

Fully connected layers, sometimes referred to as dense layers, establish connections between each neuron in one layer and every neuron in the following layer. These layers play a vital role in carrying out higher-level reasoning and decision-making processes based on the features extracted by preceding layers. In the concluding stages of a CNN, fully connected layers amalgamate all the acquired features to generate the ultimate output, commonly employed for classification endeavors.

### Training CNNs

During the training phase of CNNs, the parameters of the layers—such as weights and biases—are fine-tuned using optimization algorithms. These algorithms, like stochastic gradient descent (SGD) and its adaptations such as Adam and RMSprop, iteratively adjust these parameters to minimize a loss function. This loss function measures the disparity between the predicted outputs and the actual labels, guiding the network towards improved accuracy. Through this iterative process, the model gradually enhances its performance on the specified task.

### Impact and Applications of CNNs

CNNs have had a profound impact on the field of computer vision, driving significant advancements in image recognition systems. They have been instrumental in achieving state-of-the-art results in various applications, from autonomous vehicles to medical image analysis.

### Notable CNN Architectures

Several notable CNN architectures have emerged, each with unique characteristics and contributions to the field. Three prominent examples include VGG, InceptionNet, and EfficientNet.

#### VGG (Visual Geometry Group)

The VGG architecture stands out in the realm of deep CNNs for its straightforwardness and consistency. It comprises numerous convolutional layers paired with max-pooling layers, concluding with fully connected layers tailored for classification tasks. A pivotal aspect of VGG's design philosophy is its reliance on compact convolutional filters, typically sized at 3x3, which are layered to construct more intricate representations of visual attributes. While this approach yields remarkable accuracy by harnessing depth, it comes with heightened demands on computational power and memory resources.

#### InceptionNet (GoogLeNet)

InceptionNet, also known as GoogLeNet, introduced a novel approach with inception modules. These modules perform parallel convolutions with different filter sizes (e.g.,

1x1, 3x3, and 5x5) and concatenate the results. This design allows the network to capture features at multiple scales efficiently and reduces the number of parameters compared to traditional CNN architectures. InceptionNet's innovative structure enables it to achieve high performance with a relatively lower computational burden. EfficientNet

EfficientNet introduces a series of CNN architectures that strike a harmonious balance between model complexity and computational demands via a systematic scaling technique. In contrast to conventional scaling methods that indiscriminately augment the network's depth, width, or resolution, EfficientNet adopts a compound scaling strategy. This approach uniformly scales all dimensions in a coordinated fashion, resulting in exceptionally efficient and proficient models. By optimizing the architecture to prioritize both accuracy and efficiency, EfficientNet attains cutting-edge performance standards.

### The Role of Pretrained Models

Pretrained models are deep learning models that have already been trained on large-scale datasets for specific tasks, such as image classification, object detection, or natural language processing (NLP). These models are trained on vast amounts of labeled data to learn patterns, features, and representations useful for the task at hand.

### Advantages of Employing Pretrained Models

1. **Transfer Learning:** Pretrained models offer the opportunity for transfer learning, where they can be fine-tuned on new, often smaller datasets tailored to specific tasks. This approach drastically reduces the time and computational resources needed for training, as it capitalizes on the knowledge gained from the original task to enhance performance on the new one.
2. **Performance:** Pretrained models typically exhibit impressive performance levels on established benchmarks, thanks to their extensive training on vast and diverse datasets. This reliability makes them a preferred choice for constructing robust and precise applications.
3. **Efficiency:** Leveraging pretrained models can streamline the data collection and annotation process, as well as conserve computational resources required for training. This efficiency proves invaluable for tasks where obtaining labeled data is challenging or costly.

### Utilizing Pretrained Models Trained on ImageNet

In our project, we employ three pretrained models—VGG-16, InceptionNet, and EfficientNet—all of which have been trained on the ImageNet dataset. ImageNet is a comprehensive dataset comprising millions of annotated images spanning thousands of categories. It has significantly propelled computer vision research forward, setting new benchmarks in image classification and object recognition tasks. Models pretrained on ImageNet demonstrate strong generalization capabilities across a wide array of visual tasks, making them an ideal choice for transfer learning applications.

## Summary:

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision by providing powerful tools for image recognition, classification, and other related tasks. The hierarchical nature of CNNs, with convolutional, pooling, and fully connected layers, allows them to learn and abstract features at multiple levels, making them highly effective for visual data processing. The use of pretrained models, particularly those trained on large-scale datasets like ImageNet, further enhances their applicability and efficiency, enables the development of robust and accurate deep learning applications. Models like VGG, Inception Net, and Efficient Net exemplify the diverse approaches to CNN architecture design, each contributing to the advancement of the field in unique ways.

## Convolution Techniques:

Model	Convolution Techniques Used
VGG-16	Standard Convolution, Depthwise Separable Convolution
Inception V3	Inception Modules (1x1, 3x3, 5x5 convolutions), Deep and wide Separable Convolution
EfficientNet B0	Standard and Convolution, Depth wise Separable Convolution
EfficientNet B7	Standard Convolution, Depthwise Separable Convolution

## Methodology

We choose facial photos because, given the advancements in neural networks, using the structural information found in these images would seem to be a practical way to achieve a rapid and inexpensive test rather than relying on difficult and costly physical measurement procedures. The children's faces in the dataset are used. There are two classes in it. photographs of children with autism are found in autistic class, while photographs of the children without autism diagnosis are found in the non-autistic class.

Three segments have been created from the dataset: train, validation, and test. The test folder has two subfolders named "autistic" and "non-autistic," which hold the images required to test the model once it has been trained. A total of 100 JPG photos, each measuring 244x244x3, are present in each subfolder. Similar to the test folder, the train folder is organised into subfolders, each of which holds 1263 photos. The valid folder is organized in the same way as the test folder and contains photos used in the model's training to gauge its validation performance.

The experiment's steps are outlined in brief below:



### 1. Data Pre-Processing:

- The dataset consists 2536 images for the training, 100 images for the validation, and 300 images for the testing, distributed across two classes: "autistic" and "non\_autistic."
- The best pixel fits for each deep learning (DL) model and the dimensions used for image input are as follows:
  - VGG-16: Input size - 224x224 pixels
  - InceptionNet: Input size - 224x224 pixels
  - EfficientNet B0: Input size - 224x224 pixels
  - EfficientNet B7: Input size - 224x224 pixels

### 2. Data Validation:

- The dataset is validated by mapping the class labels to numerical values:
  - 'non\_autistic': 0
  - 'autistic': 1
- This mapping ensures uniformity in class representation across the dataset and facilitates model training.

### 3. Data Augmentation:

- Different data augmentation techniques are applied to enhance the diversity and robustness of the dataset for each DL model.
- For VGG-16, InceptionNet, and EfficientNet B0 models, the following augmentation techniques are used:
  - Rotation Range:  $\pm 40$  degrees
  - Width Shift Range:  $\pm 20\%$  of total width
  - Height Shift Range:  $\pm 20\%$  of total height
  - Shear Range:  $\pm 20\%$
  - Horizontal Flip: Enabled
  - Fill Mode: Nearest Neighbor
  - Zoom Range:  $\pm 20\%$
- These augmentation techniques help the models learn invariant features and improve generalization by exposing them to variations in the training data.
- For the EfficientNet B7 model, a more extensive set of augmentation techniques is utilized:
  - Rotation Range:  $\pm 40$  degrees
  - Width Shift Range:  $\pm 20\%$  of total width
  - Height Shift Range:  $\pm 20\%$  of total height
  - Shear Range:  $\pm 20\%$
  - Zoom Range:  $\pm 20\%$
  - Horizontal Flip: Enabled
  - Vertical Flip: Enabled
  - Brightness Range: 0.7 to 1.3
  - Channel Shift Range:  $\pm 50.0$
  - Fill Mode: Nearest Neighbor
- These augmentation techniques are more comprehensive and suitable for a deeper and more complex model like EfficientNet B7, helping to prevent overfitting and improve model performance.

For the hybrid EfficientNet B0 model, similar to the other DL models, data augmentation techniques are crucial for enhancing model performance and generalization. However, due to the hybrid nature of this model, which combines the EfficientNet B0 base with additional classification layers, the choice of augmentation techniques may slightly differ. Here's how data augmentation is applied for the hybrid EfficientNet B0 model:

#### Data Augmentation:

- The hybrid EfficientNet B0 model combines traditional augmentation techniques with regularization methods to counter overfitting and enhance model resilience.
- Like other models, rotations, shifts, shears, flips, and zooms are incorporated to diversify the training data and facilitate the model in learning invariant features.
- Additional regularization tactics like dropout and weight decay might be integrated within the classification layers to bolster the model's capacity to generalize to novel data.
- The details of data augmentation parameters, such as rotation range, shift range, and flip modes, can be fine-tuned based on the dataset's characteristics and the model's performance during training.

Data Argumentation Technique	Description
Rotation	<b>Rotate the image by a random angle.</b>
Horizontal turn or Flip	Flip an image horizontally with a certain probability.
Vertical turn or Flip	Flip the image vertically with a certain probability.
Width turn or Shift	Shift the image horizontally by a fraction of its width.
Height turn or Shift	Shift the image vertically by a fraction of its height.
Shear	Apply shear transformation to the image.
Zoom	Zoom into the image by a random factor.
Brightness of the images	Adjust the brightness of the image.
Contrast of images	Adjust the contrast of the image.

4. Data Normalization: - Data normalization is the process of rescaling input data to have an zero mean and unit variance because of which improves model stability and convergence during training in deep learning.

Below are different Normalization Techniques in ASD Detection:

Normalization Techniques Used	Description
Min-Max Scaling	Scale & pixel values to range [0,1]
Z-Score Standardization	Scale & pixel values to have zero mean and unit variance

Mean Subtraction	Subtract the mean pixel value from each pixel
Unit Variance Scaling	Scale pixel values to have input variance.

These Normalization and Argumentation techniques are applied to the input images during the preprocessing stage before feeding them into the deep learning models for training.

They plays an crucial role in improving performance and generalization ability of models.

## Proposed Method: VGG-16

### 1. Introduction to VGG-16:

- VGG-16 is a convolutional neural network (CNN) architecture developed by the Visual Geometry Group at the University of Oxford, renowned for its simplicity and effectiveness in image classification tasks.
- The "16" in VGG-16 signifies the total number of layers, consisting of 13 convolutional layers and 3 fully connected layers.
- VGG-16's appeal lies in its uniform design, where convolutional layers are stacked sequentially with small 3x3 filters, followed by max-pooling layers for downsampling.

### 2. Data Input:

- In our project, VGG-16 ingests image data categorized into "autistic" and "non\_autistic" classes from the dataset.
- Each image is represented as a pixel value matrix, typically resized to 224x224 pixels to match VGG-16's expected input size.

### 3. Model Prediction:

- VGG-16 predicts the output class (either "autistic" or "non\_autistic") for each input image through forward passes.
- During inference, input images undergo convolutional and pooling operations, followed by flattening and processing through fully connected layers.
- The final softmax layer outputs a probability distribution over classes, with the highest probability class considered the prediction.

### 4. Training Parameters:

- Backpropagation, a supervised learning method, trains VGG-16 by adjusting weights based on predicted and actual outputs.
- Convolutional layers extract hierarchical features from input images during training, followed by aggregation and classification through fully connected layers.
- Tuned training parameters include learning rate, batch size, epochs, and optimization algorithm (e.g., RMSprop or Adam) to optimize performance and prevent overfitting.

### 5. VGG Architecture and Efficiency:

- VGG-16 comprises 13 convolutional layers and 3 fully connected layers, with interleaved max-pooling layers for spatial downsampling.
- Using small 3x3 convolutional filters facilitates effective learning of spatial hierarchies, enhancing performance.
- Pooling layers reduce feature map dimensions while retaining essential features, enabling capture of both local and global information.
- VGG-16's uniform design and straightforward architecture enhance understandability and implementation, contributing to its efficiency in various image classification tasks.
- However, its extensive parameter count results in relatively high computational costs, potentially limiting deployment in resource-constrained environments. Nevertheless, its accuracy and robustness make it valuable where computational resources permit.

The VGG-16 model, with its hierarchical feature extraction and classification capabilities, is a potent tool in image classification tasks. Its simplicity, effectiveness, and ability to discern complex image patterns make it indispensable in computer vision and image analysis.

#### 6. Activation Function:

- The activation function decides whether a neuron should be activated or not after computing the weighted sum and adding bias.
- Its purpose is to introduce non-linearity to a neuron's output, enabling neural networks to learn complex patterns effectively.

#### Output Layer Activation Function:

Softmax Activation Function: Defined as:

$$(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

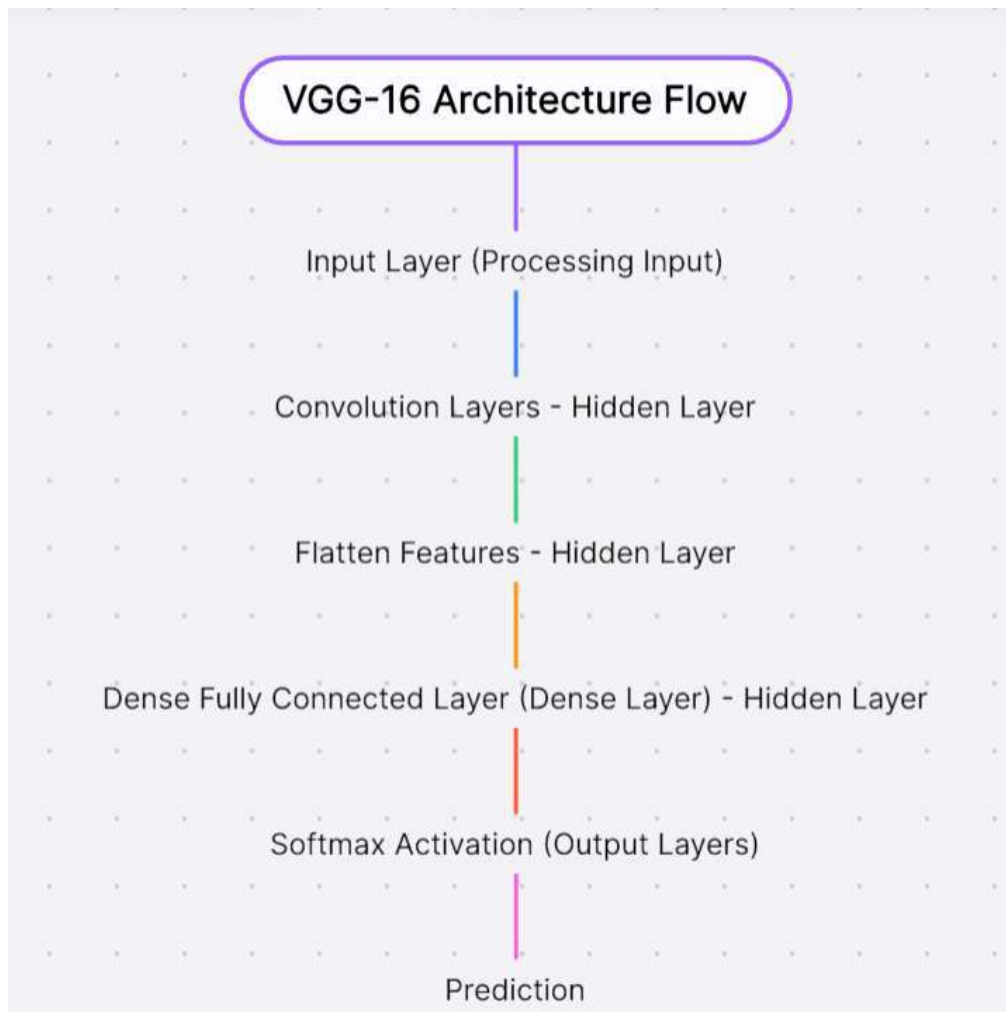
Code Snippet:

```
output = Dense(95, activation='softmax')(class1)
```

- Dense: The Dense layer, a fundamental component in neural networks, signifies a fully connected layer wherein every neuron in the layer links to each neuron in the preceding layer. This layer executes a linear operation, followed by activation function application.
- 95: In this context, 95 denotes the quantity of units or neurons in the output layer. Each unit in the output layer corresponds to a class in a multi-class classification task. For instance, if there are 95 classes, then the output layer will encompass 95 neurons, with each representing the probability of the input belonging to a specific class.
- class1: class1 designates the input to the Dense layer. It emerges as the output of a preceding layer or some intermediary computation within the neural network. The Dense layer receives this input, conducts a linear transformation, and then

applies the activation function. softmax activation function to produce the final output, which is a probability distribution over the classes.

#### **VGG-16 Architecture Flow:**



*Description: - The VGG-16 architecture initiates with an input layer dedicated to image processing. Subsequently, the model traverses multiple convolutional layers for feature extraction a flatten layer that converts to 2D feature maps into a 1D feature vector passed through dense fully connected layers for advanced processing, concluding with a softmax activation layer for classification, which yields the final prediction.*

#### **InceptionNet:**

Methodology of InceptionNet Model:

##### **1. Introduction to InceptionNet (GoogLeNet):**

- InceptionNet, also referred to as GoogLeNet, is a convolutional neural network architecture developed by researchers at Google. It stands out for its deep and broad design, meticulously crafted to capture a wide array of features across varying scales within images.

- InceptionNet introduces the innovative concept of "inception modules," comprising multiple parallel convolutional operations employing distinct filter sizes. This ingenious setup enables the network to adeptly capture both local and global features present in the input data.

- The overarching goal of this architecture is to strike a delicate balance between computational efficiency and representational prowess. By efficiently extracting features while conserving computational resources, InceptionNet ensures effective utilization in various applications requiring sophisticated image analysis.

## 2. Data Input:

- In this project, the InceptionNet model reads image data from the dataset containing images classified into "autistic" and "non\_autistic" classes.

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to 224x224 pixels to match the input size expected by the InceptionNet architecture.

## 3. Model Prediction and Training:

InceptionNet, like any other neural network model, uses both forward propagation and backward propagation during the training process.

### i. Forward Propagation:

- During forward propagation, the input data, typically images, traverses the network layer by layer in a forward direction.

- Each layer applies a series of mathematical operations to transform the input data and generate an output.

- InceptionNet's forward propagation entails guiding the input image through various inception modules and pooling layers, extracting features across different scales and spatial resolutions.

- Ultimately, the output of the final layer furnishes the predicted class probabilities for the input image.

### ii. Backward Propagation:

- Subsequent to forward propagation, during the training phase, the model's predictions are juxtaposed with the actual labels of the input images to compute the loss or error.

- Backward propagation, alias backpropagation, is subsequently employed to adjust the model's weights with the objective of minimizing this loss.

- This procedure entails computing the gradients of the loss function concerning each parameter in the network, employing the chain rule of calculus.

- These gradients are then utilized to refine the parameters (weights and biases) of the network in a direction opposite to that of the gradient, thereby effectively "backpropagating" the error throughout the network.

- By iteratively fine-tuning the model's parameters grounded on the calculated gradients, the model gradually enhances its predictive accuracy over successive iterations.

So, to clarify, InceptionNet utilizes both forward propagation (for prediction) and backward propagation (for training) during the training process. Forward propagation

is used to make predictions, while the backward propagation are used to update the model's parameters based on the computed errors.

#### 4. InceptionNet Architecture and Efficiency:

- The InceptionNet architecture consists of multiple inception modules stacked sequentially, interspersed with max-pooling layers for spatial downsampling.
- Each inception module incorporates multiple convolutional operations with different kernel size are (1x1, 3x3, and 5x5), allowing model to capture features different scales efficiently.
- The use of parallel convolutional operations within inception modules enables the model to extract diverse and informative features while minimizing computational cost.
- Inception Net's efficient architecture strikes a balance between the computational efficiency and representational of power, making it suitable for tasks where both accuracy and efficiency are important considerations.
- Despite its deep and wide architecture, Inception Net achieves competitive performance on image classification tasks with relatively fewer parameters compared to other architectures, making it a popular choice for various computer vision applications.

In summary, the InceptionNet model is an the deep and convolutional neural network architecture is designed to efficiently capture multi-scale features within images using inception modules. Its balanced architecture, efficient feature extraction, and competitive performance make it a valuable tools for image classification tasks, including the detection of the autism spectrum disorder in this project.

#### 5. Output Layer Activation Function:

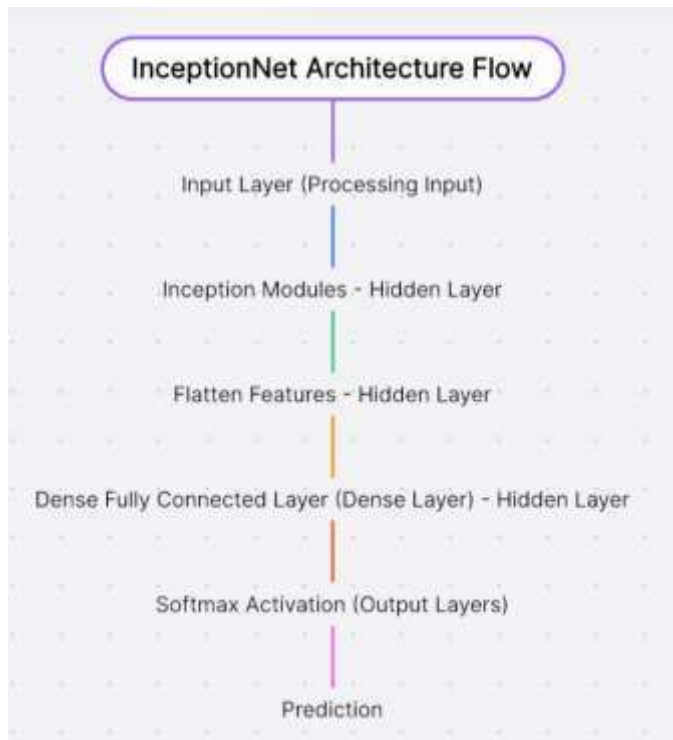
Softmax Activation Function: Defined as:

$$(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Code Snippet:

```
predictions = Dense(1, activation="softmax")(x)
```

#### **Inception Net Architecture Flow:**



*Description: - Inception v3, a convolutional neural network, starts with an input layer for images. It then utilizes multiple Inception modules, which are like mini-networks that extract features at various scales. After these modules, a flatten layer transforms the extracted features into the single vector. This vector fed into fully- connected layers for high-level reasoning, and finally a softmax layer outputs probabilities for different object categories.*

## **EfficientNet-B0**

### 1. Introduction to EfficientNet:

- EfficientNet is a family of convolutional neural network architectures developed by researchers at Google. It aims to achieve state-of-the-art performance while maintaining efficiency in terms of computational resources.
- EfficientNet introduces a novel compound scaling method that uniformly scales the network width, depth, and resolution, resulting in models that are both highly accurate and computationally efficient.
- The architecture balances between model size and accuracy by optimizing the trade-off between network depth, width, and resolution, leading to models that achieve superior performance with fewer parameters

### 2. Data Input:

- In this project, the EfficientNet model reads image data from the dataset containing images classified into "autistic" and "non\_autistic" classes.
- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to match the input size expected by the EfficientNet architecture.

### 3. Model Prediction:



- EfficientNet predicts the output class (either "autistic" or "non\_autistic") for each input image through forward propagation.
- During forward propagation, input image undergoes an series of convolutional operation within EfficientNet architecture, allowing the model to extract hierarchical features.
- The final output of the softmax layer presents a probability distribution across all classes, where each class is assigned a probability score. Subsequently, the class exhibiting the highest probability is identified as the predicted class for the input image.

#### 4. Training Parameters:

- The EfficientNet model is trained using backpropagation, similar to other deep learning architectures, where the model's weights are adjusted based on the error and between predicted the actual outputs.
- During the training, the model learns to an extract hierarchical features from input images through the convolutional layers, which are then aggregated and classified into different categories through fully connected layers.
- The training parameters include the learning rate, batch size, number of epochs, and optimization algorithm (e.g., RMSprop or Adam), which are tuned to optimize model performance on the training dataset while preventing overfitting.

#### 5. Efficient Net Architecture and Efficiency:

- The EfficientNet architecture integrates numerous convolutional layers with an innovative approach to scaling network width, depth, and resolution efficiently.
- Employing a compound scaling technique, EfficientNet uniformly adjusts the network's width, depth, and resolution to strike an optimal balance between model size and accuracy.
- This architecture adopts depthwise separable convolutions, a technique that significantly reduces parameter count and computational overhead while preserving expressive capabilities.
- EfficientNet outperforms its counterparts by delivering superior performance with fewer parameters, showcasing remarkable efficiency in computational resource utilization.
- Leveraging efficient scaling and depthwise separable convolutions, EfficientNet achieves state-of-the-art results across diverse image classification tasks, making it a favored option for environments with limited computational resources.

In summary, the EfficientNet model is a highly efficient convolutional neural network architecture that achieves state-of-the-art performance on image classification tasks with fewer parameters and computational resources.

Its compound scaling method and depthwise separable convolutions enable it to strike a balance between model size and accuracy, making it suitable for various computer vision applications, including the detection of autism spectrum disorder in this project.

#### Output Layer Activation Function:

Sigmoid Activation Function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Code Snippet:  
predictions = Dense(1, activation="sigmoid")(x)

## **EfficientNet – B7**

Methodology of Efficient Net B7 Model:

### 1. Introduction to Efficient Net B7:

- Efficient Net B7 is CNN architecture developed Google, belonging to the Efficient Net family. It is one of the largest variants in the series, EfficientNet is meticulously crafted to excel in image classification tasks while maximizing the utilization of computational resources, aiming to achieve state-of-the-art performance.

- The architecture of Efficient Net B7 is characterized by its depth, width, and resolution scaling, which are optimized using a compound & scaling method balance model size & accuracy.

- Efficient Net B7 incorporates depth wise separable convolutions and efficient feature extraction techniques, enabling it to capture complex patterns and hierarchical features from input images efficiently.

### 2. Data Input:

- The EfficientNet B7 model reads image data from the dataset containing images categorized into "autistic" and "non\_autistic" classes.

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to match the input size expected by the EfficientNet B7 architecture.

### 3. Model Prediction:

- EfficientNet B7 undertakes the prediction of output classes, namely "autistic" or "non\_autistic," for each input image through forward propagation.

- During this propagation, the input image traverses through a series of convolutional operations within the EfficientNet B7 architecture, efficiently extracting hierarchical features.

- These extracted features are subsequently amalgamated and processed across multiple network layers, progressively learning representations that are increasingly distinctive for the target classes.

- The conclusive outcome of the model manifests as a probability distribution across classes, with the predicted class being the one exhibiting the highest probability.

### 4. Training Parameters:

- The training of the EfficientNet B7 model entails the utilization of backpropagation, a standard deep learning technique for optimizing model parameters.

- Throughout the training phase, the model's weights are iteratively adjusted based on the discrepancy between predicted and actual outputs, with the intent of minimizing a designated loss function.

- Key training parameters, including learning rate, batch size, number of epochs, and optimization algorithm (e.g., RMSprop or Adam), are fine-tuned to optimize model performance while mitigating overfitting.

### 5. EfficientNet B7 Architecture and Efficiency:

- The architecture of EfficientNet B7 encompasses numerous layers of convolutional operations, interspersed with depthwise separable convolutions and efficient scaling methodologies.
- The scalability in depth and width within EfficientNet B7 enables the capture of intricate patterns and representations from input images, while resolution scaling optimizes computational efficiency by striking a balance between model size and accuracy.
- Leveraging efficient feature extraction techniques, EfficientNet B7 achieves unparalleled performance in image classification tasks while upholding computational efficiency.
- By harnessing depthwise separable convolutions and efficient scaling strategies, EfficientNet B7 attains superior performance with reduced parameter count compared to alternative architectures, rendering it exceptionally efficient for an extensive array of computer vision applications.

In summary, EfficientNet B7 is a powerful convolutional neural network architecture optimized for efficient use of computational resources while achieving state-of-the-art performance on image classification tasks. Its depth, width, and resolution scaling, combined with efficient feature extraction techniques, make it a valuable tool for various computer vision applications, including the detection of autism spectrum disorder in this project.

### Output Layer Activation Function:

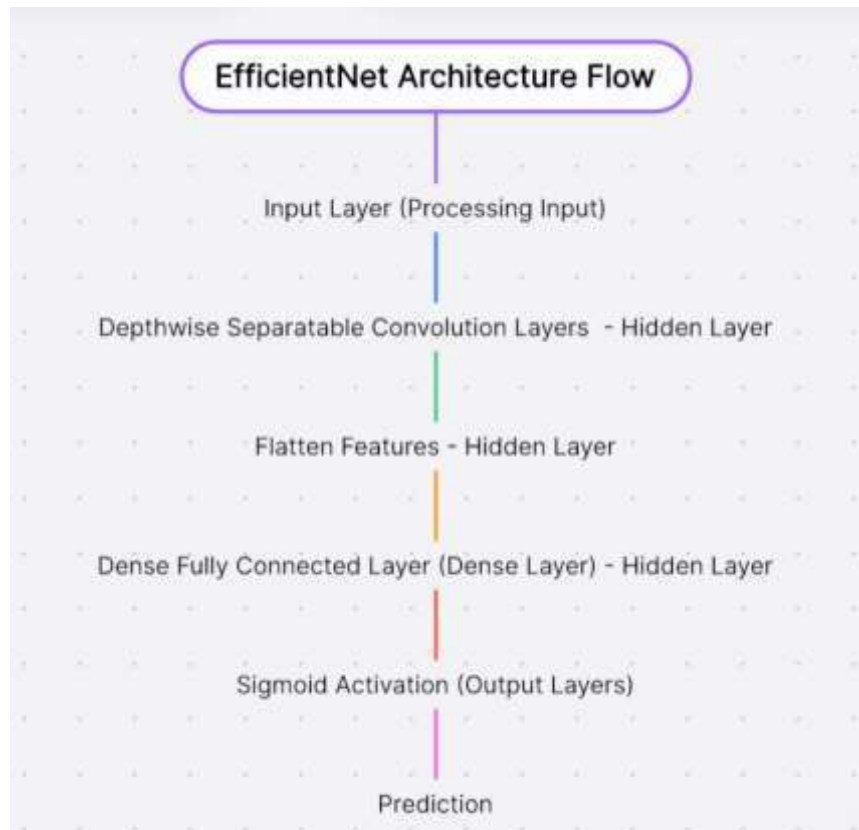
Sigmoid Activation Function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Code Snippet:

```
output = Dense(1, activation='sigmoid')(x)
```

### **Efficient Net Architecture Flow**



*Description: - The EfficientNet architecture prioritizes processing efficiency while maintaining accuracy. It starts with an input layer that receives the image data. Data then progresses through depthwise separable convolution layers, which are essential building blocks for efficient feature extraction. Next, a flatten layer transforms the extracted features into a one-dimensional vector.*

### **Modified EfficientNet-Bo (Proposed Model)**

Methodology of Modified EfficientNet B0 Model:

#### **1. Introduction to EfficientNet B0 Hybrid Model:**

- The EfficientNet B0 Hybrid Model is a custom variation of the EfficientNet B0 architecture, enhanced with additional classification layers for improved performance on specific tasks.
- This hybrid model incorporates the efficient scaling and depthwise separable convolutions of EfficientNet B0 while adding custom classification layers to adapt the model to the target task.
- By leveraging the efficient feature extraction capabilities of EfficientNet B0 and fine-tuning the classification layers, the hybrid model aims to achieve superior performance on image classification tasks with reduced computational cost.

#### **2. Data Input:**

- Similar to other models, the EfficientNet B0 Hybrid Model reads image data from the dataset containing images categorized into "autistic" and "non\_autistic" classes.

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized match the input size expected by the Efficient Net B0 architecture.

### 3. Model Prediction:

- The EfficientNet B0 Hybrid Model executes the prediction of output classes, "autistic" or "non\_autistic," for each input image via forward propagation.
- Throughout this propagation, the input image navigates through a sequence of convolutional operations within the EfficientNet B0 architecture, adeptly extracting hierarchical features.
- These extracted features then traverse through supplementary classification layers integrated into the model, which meticulously process the features and prognosticate the probabilities of output classes.
- Subsequently, the class exhibiting the highest probability is identified as the predicted class for the input image.

### 4. Training Parameters:

- The EfficientNet B0 Hybrid Model is trained using backpropagation, similar to other deep learning architectures.
- During training, the model's weights are adjusted based on the error between predicted and actual outputs, optimizing the model's performance on the training dataset.
- The training parameters include the learning rate, batch size, number of epochs, and optimization algorithm (e.g., RMSprop or Adam), which are tuned to optimize model performance while preventing overfitting.

### 5. EfficientNet B0 Hybrid Model Architecture and Efficiency:

- The EfficientNet B0 Hybrid Model architecture merges the efficiency of scaling and depthwise separable convolutions from EfficientNet B0 with supplementary classification layers.
- EfficientNet B0's efficient feature extraction capabilities enable the model to capture meaningful patterns from input images while minimizing computational resources.
- The hybrid model's additional classification layers further refine the extracted features, adapting them to the specific task of classifying autism spectrum disorder.
- By leveraging EfficientNet B0's efficiency and fine-tuning the classification layers, the hybrid model achieves superior performance on image classification tasks with reduced computational cost.
- The use of efficient scaling and depthwise separable convolutions, combined with custom classification layers, allows the EfficientNet B0 Hybrid Model to strike a balance between accuracy and efficiency, making it a powerful tool for image classification tasks, including the detection of autism spectrum disorder.

In summary, the EfficientNet B0 Hybrid Model is a custom variation of the EfficientNet B0 architecture, enhanced with additional classification layers for improved performance on specific tasks. By leveraging the efficiency of EfficientNet B0 and fine-tuning the classification layers, the hybrid model achieves superior performance on image classification tasks with reduced computational cost. Its architecture and efficiency make it a valuable tool for various computer vision applications, including the detection of autism spectrum disorder in this project.

### Output Layer Activation Function:

Sigmoid Activation Function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Code Snippet:

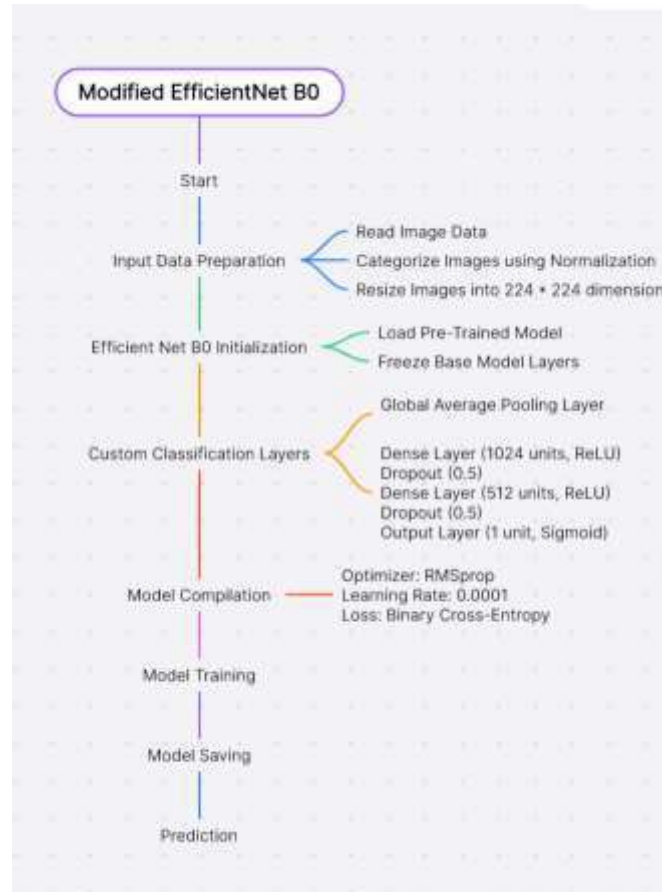
```
output = Dense(1, activation='sigmoid')(x)
```

### Impact of Proposed Model

Modification on Efficient B0	Impact
Freezing base model layers	- Reduces training time by focusing on the custom classification head. Potentially limits model capacity for complex tasks.
GlobalAveragePooling2D	- Efficiently summarizes spatial information from the base model's output.
Dense layers (1024, 512 and units with ReLU activation and Dropout)	- Introduces non-linearity and helps prevent overfitting by reducing model complexity.
Output layer (1 unit with sigmoid activation)	- Produces a single output between 0 and 1 for binary classification (e.g., ASD vs. Non-ASD).
RMSprop optimizer with low learning rate (0.0001)	- Slows down learning process but helps to find a minimum in the loss function and avoid overshooting.

The above methods will improve the overall image classification ability of the model.

### **Work Flow of the Modified EfficientNet B0**



*Description: - The Modified Efficient Net B0 model pipeline involves reading and normalizing image data, resizing it to  $224 \times 224$  pixels, and loading a pre-trained Efficient Net B0 with frozen base layers. Custom classification layers, including dense and dropout layers, culminate in an sigmoid-activated output. The model is compiled with an RMSprop, 0.0001 learning rate, and binary the cross-entropy loss, then trained, saved, and used for predictions.*

### **Architectural Comparison**

Here's a tabular comparison of the mentioned model architectures along with the number of layers in each:

Model	Architecture	Number of Layers
VGG-16	Sequential	16
InceptionNet	GoogLeNet-like	Varies
EfficientNet B0	Efficient Scaling	Varies
EfficientNet B7	Efficient Scaling	Varies
Modified Model	Custom	Varies

The number of layers in InceptionNet, EfficientNet B0, and EfficientNet B7 varies depending on the specific configuration (like pooling layers, activation layers, etc) and variant used in the implementation.

Therefore, the exact number of layers may differ between different instances of these models.

### **Activation Functions used in Output Layers**

Model	Activation Function	Usage
1. VGG-16	Softmax	Multi-Class Classification Task
2. Inception Net	Softmax	Multi-Class Classification Task
3. EfficientNet B0	Sigmoid	Binary Classification Task
4. EfficientNet B7	Sigmoid	Binary Classification Task
5. Modified EfficientNet B0	Sigmoid	Binary Classification Task

Sigmoid activation is used in models designed for binary classification tasks (EfficientNet B0, EfficientNet B7, and Hybrid EfficientNet B0), while softmax activation is used in models designed for multi-class classification tasks (VGG-16 and Inception Net).

### **Description of Activation Functions:**

Activation Function	Formula
Sigmoid	<ul style="list-style-type: none"> <li>Sigmoid activation is commonly used in binary classification tasks, where the output represents the probability of the input belonging to the positive class.</li> <li>It squashes the input values to the range [0, 1], which can be interpreted as probabilities.</li> <li>Defined as: <math>\sigma(x) = \frac{1}{1+e^{-x}}</math></li> </ul> <p>- <math>\sigma(x)</math> is the output of the sigmoid function for a given input <math>x</math>.</p> <p>- <math>e</math> is Euler's number, approximately equal to 2.71828.</p> <p>- <math>x</math> is the input to the function</p>
Softmax	<ul style="list-style-type: none"> <li>It transforms the raw output of the neural network into a probability distribution over multiple classes, enabling the model to make predictions about the likelihood of an input belonging to each class.</li> <li>Defined as: <math>(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}</math></li> </ul> <p>- <math>x_i</math> represents <i>ith element of the vector x</i></p> <p>- <math>e</math> is Euler's number, approximately equal to 2.71828.</p> <p>- The sum in the denominator is computed over all elements <math>x_j</math> of the input vector <math>x</math>.</p>

### **Workflow of the Project**





## **Model Compilation:**

Model compilation is an essential step in training a neural network model, where you configure the model with additional parameters beyond its architecture. This step involves specifying the optimizer, loss function, and optional metrics to monitor during training and evaluation.

1. **Optimizer:** The optimizer is responsible for updating the model's weights based on the gradient of the loss function. It plays a crucial role in the training process by determining how the model learns from the data.

- *Adam Optimizer:* The Adam optimizer is a popular choice for training neural networks. It adjusts the learning rates for each parameter adaptively, based on the historical gradients. This adaptive approach facilitates efficient and effective optimization, helping the model converge faster and perform better during training.

2. **Loss Function:** The loss function measures the difference between the model's predictions and the actual target values (labels) during training. It quantifies the model's performance on a given batch of data. By minimizing the loss function, the model improves its accuracy over time.

- The choice of the loss function depends on the specific task and the nature of the data. For example, in classification tasks, cross-entropy loss is commonly used, while mean squared error is often used for regression tasks.

The model compilation involves setting up the optimizer and loss function, which are critical for the training process. The optimizer, such as Adam, ensures efficient learning by adaptively adjusting learning rates, while the loss function provides a metric for evaluating the model's performance and guiding its improvements. By carefully configuring these parameters, you can enhance the effectiveness of your neural network training process.

### **Optimizer Algorithms for Each Model:**

Model	Optimizer Algorithm	Learning Rate
VGG-16	Adam (Adaptive Moment Estimation)	0.001
InceptionNet	RMSprop	0.001
EfficientNet B0	RMSprop	0.001
Modified EfficientNet B0	RMSprop	0.001
EfficientNet B7	RMSprop	0.001

### **Pre-Trained Models:**

Pretrained models are deep learning models that have already been trained on extensive datasets for specific tasks, such as image classification, object detection, natural language processing (NLP), and other machine learning applications. These models have undergone extensive training on vast amounts of labeled data, enabling them to learn patterns, features, and representations pertinent to their designated tasks.

In our project, we are leveraging three prominent pretrained models: VGG-16, InceptionNet, and EfficientNet.

- VGG-16: This model is known for its simplicity and effectiveness, featuring a deep architecture with 16 layers. It has been widely adopted for image classification tasks.
- InceptionNet: Recognized for its inception modules, this model efficiently handles various scales of information within an image. Its architectural innovations help achieve high accuracy in image classification and object detection.
- EfficientNet: This model is renowned for its efficiency and scalability. It achieves high performance by optimizing the balance between depth, width, and resolution of the network.

### **Dataset for Pre-Trained Models:**

The pretrained models we are using are trained on the ImageNet dataset. ImageNet is a large-scale dataset comprising millions of labeled images across thousands of categories. It is one of the most significant datasets in computer vision research and has played a crucial role in advancing the field of deep learning.

ImageNet encompasses a wide range of objects, scenes, and concepts, making it a comprehensive resource for training models to recognize and classify diverse images. The dataset was created with the objective of pushing the boundaries of image classification and object recognition tasks. Consequently, it has become a standard benchmark for evaluating and comparing the performance of various deep learning models.

Projects and research initiatives involving image classification, object detection, image segmentation, and other computer vision tasks frequently utilize ImageNet. The vast amount of labeled data it provides allows models to learn intricate details and

features of images, improving their accuracy and generalization capabilities.

By using pretrained models on the ImageNet dataset, we can leverage the extensive knowledge and representations these models have acquired during their training. This approach not only saves time and computational resources but also enhances the performance of our models on our specific tasks. Instead of training models from scratch, we can fine-tune these pretrained models on our dataset, adapting their learned features to our unique requirements.

In summary, pretrained models like VGG-16, InceptionNet, and EfficientNet, trained on the comprehensive ImageNet dataset, offer a robust foundation for our project. These models bring the benefits of extensive prior learning, enabling us to achieve high performance in our specific image classification and object recognition tasks with greater efficiency.

### **Transfer-Learning**

*Transfer Learning functions on the idea of reusing knowledge gained from one task to improve performance on a related but different task.*

1. VGG-16

We modify the pretrained VGG-16 model by adding new classification layers on top of the existing convolutional base.

The weights of the convolutional layers are frozen (not trained), while the new classification layers are trained on your specific dataset.

This is a form of transfer learning where the pretrained VGG-16 model serves as a feature extractor for your task of autism detection.

2. InceptionNet

Similar to VGG-16, we use a pretrained InceptionNet model as a feature extractor. We add new classification layers on top of the InceptionNet base and fine-tune them on our dataset.

Again, the weights of the pretrained layers are frozen, and only the new classification layers are trained.

3. EfficientNet B0

The same transfer learning approach is applied here, where the pretrained EfficientNet B0 model is used as a feature extractor, and new classification layers are added and trained on your dataset.

4. EfficientNet B7

Transfer learning is also used with the pretrained EfficientNet B7 model, where the existing convolutional base serves as a feature extractor, and new classification layers are added and trained on your dataset.

5. Modified EfficientNet B0

Similarly, transfer learning is applied to the hybrid EfficientNet B0 model, where the pretrained base model is used for feature extraction, and new classification layers are added and trained.

### **Transfer Learning Techniques:**

<b>Model</b>	<b>Transfer Learning Technique</b>
VGG-16	Feature Extraction
InceptionNet V3	Feature Extraction
EfficientNet B0	Feature Extraction
EfficientNet B7	Feature Extraction
Modified EfficientNet B0	Feature Extraction

### **Model Fitting**

In deep learning, "fitting" signifies the training process of a neural network on a specific dataset. This involves utilizing the `fit` method, which is a standard function in deep learning libraries like TensorFlow and Keras. The `fit` method enables you to train a model by providing the necessary training data, specifying the number of epochs, determining the batch size, and setting other relevant training parameters. The primary goal of the fitting process is for the model to fine-tune its weights and biases in response to the training data.

This fine-tuning aims to minimize a defined loss function, thereby optimizing the model's performance for the given task. Essentially, the fitting process is an iterative optimization procedure that adjusts the model parameters to improve its predictive accuracy. This optimization is conducted through a series of forward and backward passes within the neural network. During the forward pass, input data is propagated through the network, and predictions are generated. This process involves passing data through multiple layers of neurons, each applying a specific mathematical transformation to the input.

The predictions made at the end of the forward pass are then compared to the actual target values, and the difference between them is quantified using a loss function. The loss function is a critical component as it measures the model's prediction error. Commonly used loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks. The goal is to minimize this loss function, indicating that the model's predictions are becoming more accurate.

After computing the loss, the next step is the backward pass, which involves calculating the gradients of the loss function with respect to each weight in the network. This is done using a technique called backpropagation. Backpropagation applies the chain rule of calculus to propagate the gradient of the loss function backward through the network, from the output layer to the input layer. Once the gradients are computed, they are used to update the model's weights and biases.

This update is typically performed using an optimization algorithm like Stochastic Gradient Descent (SGD), Adam, or RMSprop. These algorithms adjust the weights in the direction that reduces the loss, thereby improving the model's accuracy over successive iterations. The fitting process continues for a specified number of epochs, where an epoch is defined as one complete pass through the entire training dataset. Within each epoch, the dataset is often divided into smaller batches, and the model

parameters are updated after each batch. This approach, known as mini-batch gradient descent, helps in stabilizing and speeding up the training process.

Throughout training, the model's performance is periodically evaluated on a validation set, which is a separate portion of the dataset not used for training. This evaluation helps in monitoring the model's generalization ability and in tuning hyperparameters to avoid overfitting.

Therefore, the process of fitting a neural network involves training the model by adjusting its parameters to minimize a loss function, thereby enhancing its ability to make accurate predictions. This is achieved through iterative forward and backward passes, using backpropagation and optimization algorithms to fine-tune the model's weights and biases.

### **Batch Size: -**

The batch size in neural network training refers to the number of training examples processed in one iteration. Essentially, it's the count of samples that the model handles before updating its parameters using the gradients calculated from the loss function. For instance, if you set the batch size to 32, the model processes 32 training examples at a time before updating its weights and biases.

During training, the entire dataset is divided into these smaller batches, and each batch is fed into the model sequentially. The choice of batch size can significantly influence the training process. A larger batch size often leads to faster convergence since more data is processed at once, which can make the gradient estimates more accurate. However, this comes at the cost of requiring more memory, which might not be feasible for very large datasets or limited hardware resources. On the other hand, a smaller batch size may result in slower convergence. This is because smaller batches provide noisier estimates of the gradient, which can introduce more fluctuations during training.

However, these fluctuations can sometimes help the model escape local minima and potentially generalize better to unseen data. Choosing the optimal batch size often involves a trade-off between computational efficiency and model performance. Larger batch sizes can make better use of modern hardware accelerators like GPUs and TPUs, leading to faster training times.

However, they might not always provide the best generalization performance. Conversely, smaller batch sizes might slow down training but could potentially lead to a model that performs better on new, unseen data. This balance makes the selection of an appropriate batch size a critical decision in the design of neural network training processes.

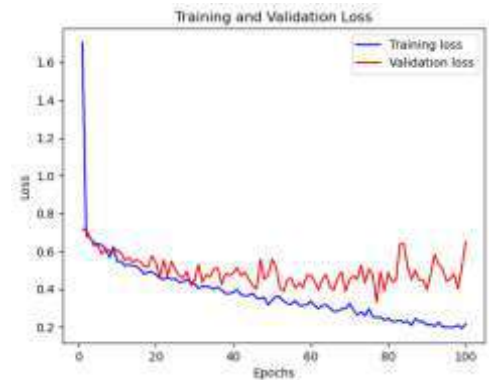
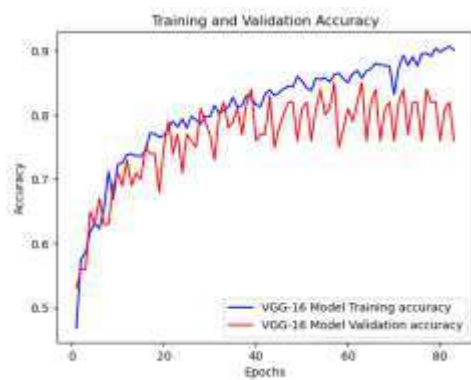
Therefore, the batch size is a crucial hyperparameter in neural network training, determining how many examples are processed before updating model parameters. While larger batch sizes can speed up training, they require more memory and might not always yield the best generalization. Smaller batch sizes, although slower, can sometimes offer better generalization to new data. Finding the right balance is key to optimizing the training process and model performance.

### **Batch Size for Each Model:**

Model	Batch Size
VGG-16	128
Inception V3	128
EfficientNet B0	128
EfficientNet B7	128
Hybrid EfficientNet B0	64

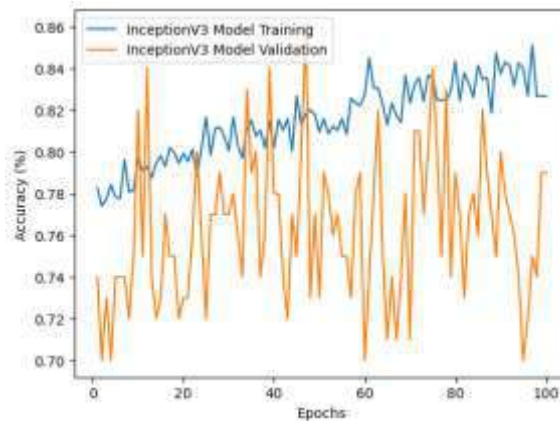
### **Results**

#### 1. VGG-16



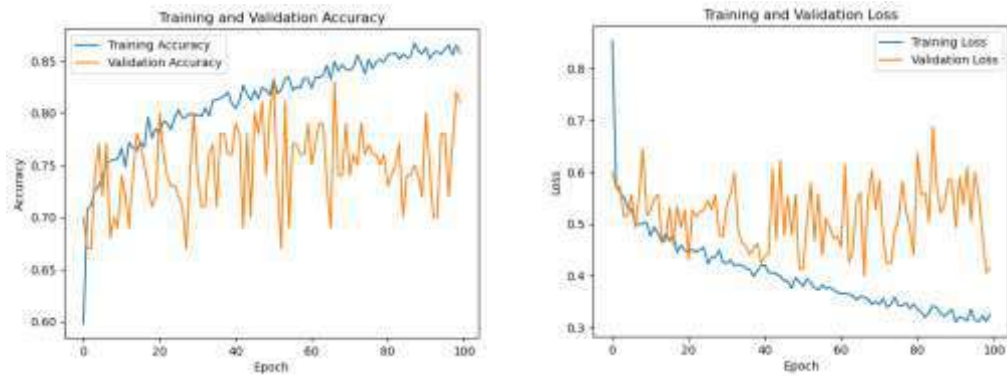
*VGG-16 Training and Validation Accuracy      VGG-16 Training Loss and Validation Loss*

#### 2. InceptionNet V3



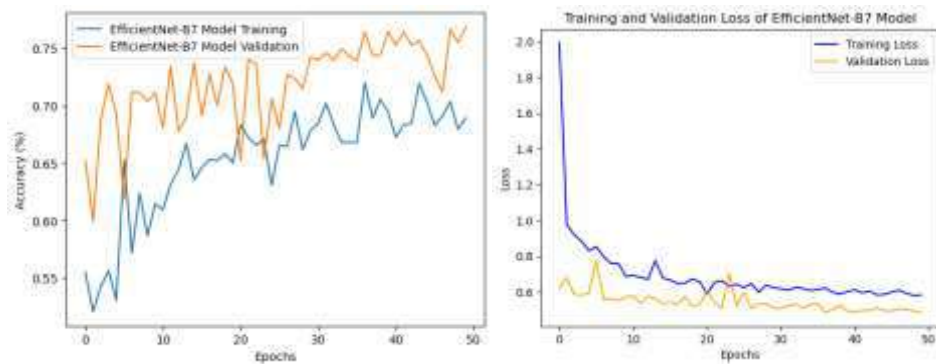
*InceptionNet Training and Validation Accuracy*

### 3. EfficientNet B0



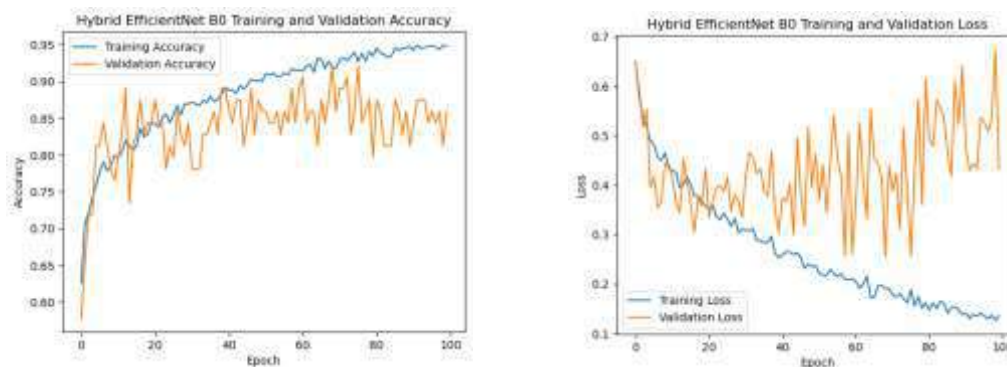
*Efficient Net B0 Training and Validation Accuracy Graph & Training and Validation Loss Graphs*

### 4. EfficientNet B7

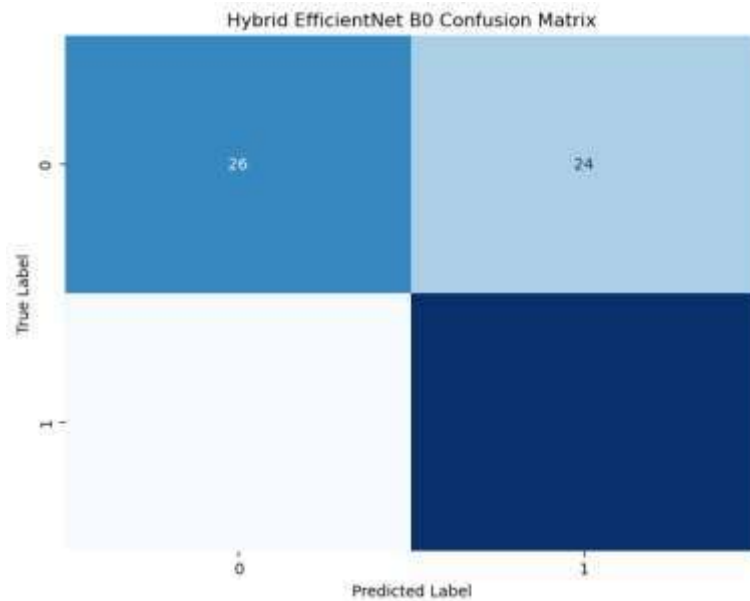


*Efficient Net B7 Training and Validation Accuracy Graph & Training Loss and Validation Loss Graph*

### 5. Modified EfficientNet B0



*Efficient Net B7 Training and Validation Accuracy Graph & Training Loss and Validation Loss Graph*



*Modified EfficientNet B0 Confusion Matrix*

### **Model Accuracy Comparison Table**

Model	Accuracy	Validation Accuracy
1. VGG-16	90.69%	76.00%
2. Inception V3	82.65%	79.00%
3. EfficientNet B0	85.73%	81.00%
4. EfficientNet B7	68.93%	76.85%
5. Modified EfficientNet B0	94.82%	85.94%

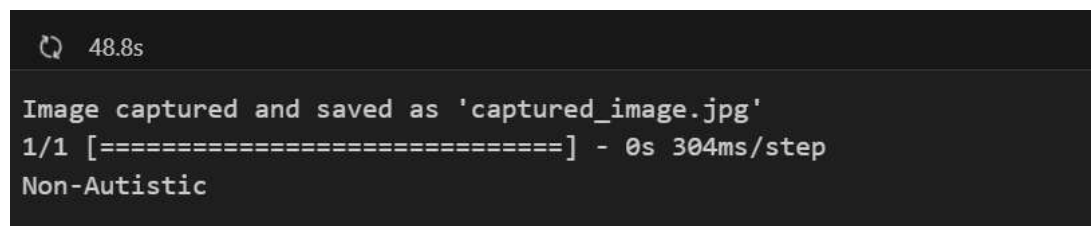
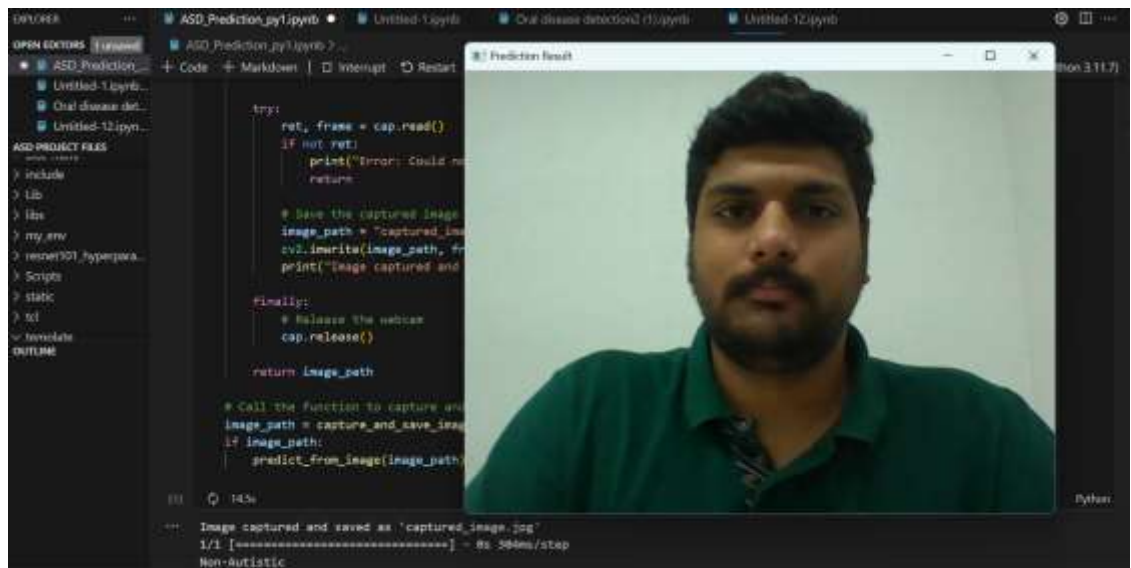
### **Draw-Backs for Each Model**

Model	Drawbacks	Solution
1. VGG-16 Model	<ul style="list-style-type: none"> <li>- Large Size &amp; High Computational Cost</li> <li>- Slow Training Time</li> </ul>	<ul style="list-style-type: none"> <li>- Transfer Learning with Pre-trained Weights</li> <li>- Layer Freezing to Reduce Trainable Parameters</li> </ul>
2. InceptionV3 Model	<ul style="list-style-type: none"> <li>- Complex Architecture</li> <li>- High Computational Cost</li> </ul>	<ul style="list-style-type: none"> <li>- Simplified Architectures for Limited Resources - Efficient Hardware (e.g., GPUs)</li> </ul>
3. EfficientNet B0 and B7	<ul style="list-style-type: none"> <li>- Requires Careful Parameter Tuning</li> <li>- B7 Can Be Time-Consuming to Train</li> </ul>	<ul style="list-style-type: none"> <li>- Automated Hyperparameter Tuning Tools</li> <li>- Model Selection Based on Resources &amp; Needs</li> </ul>
4. Modified EfficientNet B0	<ul style="list-style-type: none"> <li>- Increased Implementation Complexity</li> <li>- Longer Training Time</li> </ul>	<ul style="list-style-type: none"> <li>- Gradual Unfreezing During Training</li> <li>- Layer-wise Learning Rates</li> </ul>



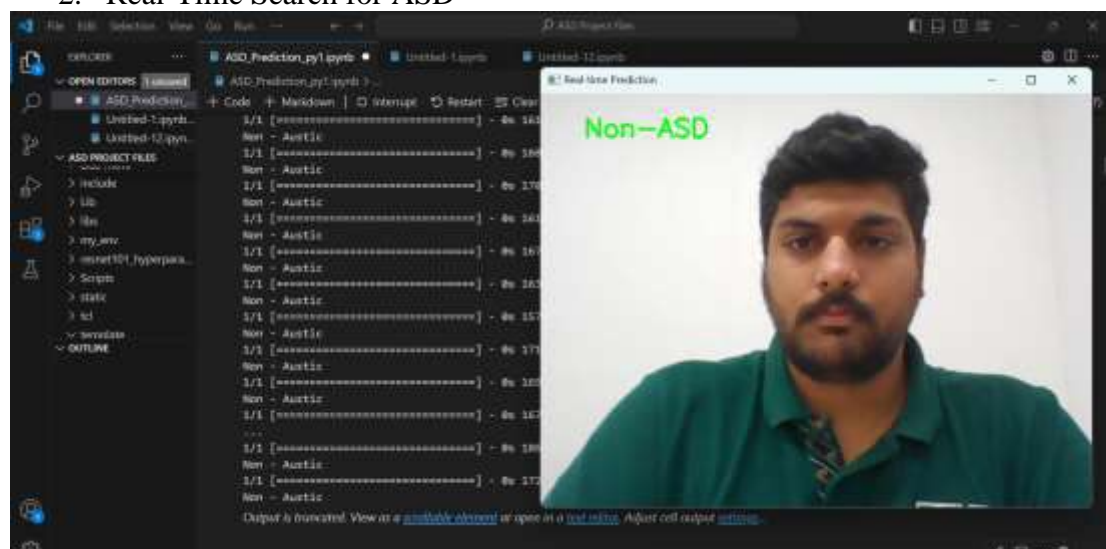
## Real-Time Outputs:

### 1. Real Time Capture



*Description: - This picture shows a real-time output of pre-trained deep neural network model. When we take a picture with the webcam, the code feeds the image to this model. The model then analyses the image, likely focusing on facial features or other visual cues, and attempts to predict the likelihood of ASD in the person depicted.*

### 2. Real-Time Search for ASD



```

1/1 [=====] - 0s 157ms/step
Non - Austic
1/1 [=====] - 0s 171ms/step
Non - Austic
1/1 [=====] - 0s 169ms/step
Non - Austic

```

*Description: It is the real-time ASD detection through webcam analysis. It loads a pre-trained deep learning model. The webcam captures frames, which are pre-processed and fed to the model. The model predicts ASD based on the image, displaying "ASD detected" or "non-ASD" on the frame and printing the result.*

## Conclusion and Future Scope

The Modified EfficientNet B0 Model is a customized version of the EfficientNet B0 architecture, optimized with additional classification layers for enhanced performance in specific tasks, such as classifying autism spectrum disorder. This model leverages EfficientNet B0's efficient feature extraction through depthwise separable convolutions and efficient scaling, while fine-tuning classification layers to better suit the target task. The hybrid approach ensures high accuracy with reduced computational cost, making it an effective tool for image classification tasks. Through forward propagation and efficient hierarchical feature extraction, the model predicts the output class for input images, with training parameters optimized to prevent overfitting and maximize performance. Overall, the EfficientNet B0 Hybrid Model achieves a balance between accuracy and efficiency, making it a powerful solution for computer vision applications, including autism detection.

The future of ASD detection using deep learning models presents a compelling path towards more comprehensive diagnosis and improved patient outcomes. One promising avenue lies in multimodal fusion. Deep learning models may evolve beyond analysing isolated data streams, such as images or eye gaze, and instead integrate information from various modalities. This could involve combining brain scans, facial expressions, and even speech patterns.

By leveraging this rich tapestry of data, researchers aim to create a more nuanced understanding of ASD, leading to more accurate and reliable diagnoses. Another key area of exploration is Explainable AI (XAI). While deep learning excels at identifying patterns in data, its decision-making process can often be opaque. XAI techniques will be crucial in demystifying how these models arrive at ASD diagnoses. This transparency is not only essential for building trust in the technology but also holds the potential to reveal new insights into the underlying mechanisms of ASD itself.

Finally, deep learning could be harnessed to develop tools for continuous monitoring and personalized care.

## **References: -**

1. Autism\_Spectrum\_Disorder\_Detection\_in\_Children\_Using\_Transfer\_Learning\_Techniques.  
<https://ieeexplore.ieee.org/abstract/document/10212257>
2. Autistic\_Spectrum\_Disorder\_Screening\_Prediction\_with\_Machine\_Learning\_Models  
<https://ieeexplore.ieee.org/document/9077881>
3. A\_Deep\_Convolutional\_Neural\_Network\_based\_Detection\_System\_for\_Autism\_Spectrum\_Disorder\_in\_Facial\_images  
<https://ieeexplore.ieee.org/document/9641046>
4. Analysis and Detection of Autism Spectrum Disorder Using Machine Learning Techniques  
[https://www.researchgate.net/publication/340711028\\_Analysis\\_and\\_Detection\\_of\\_Autism\\_Spectrum\\_Disorder\\_Using\\_Machine\\_Learning\\_Techniques](https://www.researchgate.net/publication/340711028_Analysis_and_Detection_of_Autism_Spectrum_Disorder_Using_Machine_Learning_Techniques)

# RE-2022-287075 - Turnitin Plagiarism Report

*by Sree Venkateswarlu Chadalawada*

---

**Submission date:** 29-May-2024 06:25PM (UTC+0700)

**Submission ID:** 271717001976

**File name:** RE-2022-287075.docx (2.37M)

**Word count:** 10383

**Character count:** 63179

## ABSTRACT

3

Autism Spectrum Disorder (ASD) is a developmental condition that affects how people interact socially, communicate, and behave<sup>12</sup>. Identifying ASD early and starting treatment as soon as possible can greatly improve the quality of life for those affected. Recently, deep learning models have become valuable in detecting ASD, utilizing detailed information from medical images to support the diagnostic process.

This project, titled "Detecting ASD with Deep Learning Models," introduces a cutting-edge method to automatically identify Autism Spectrum Disorder by analyzing medical images. It examines the effectiveness of several advanced deep learning architectures, such as VGG-16, Inception Net, EfficientNet-B0, EfficientNet-B7, and a custom hybrid model built on EfficientNet-B0.

58

In the initial phase of this project, the dataset—consisting of images of individuals both with and without ASD—is prepared. To increase the variety and reliability of the dataset, various data augmentation techniques are used. These include rotating, shifting, shearing, flipping, and zooming the images.

Subsequently, several deep learning models are implemented and evaluated. The VGG-16 model, known for its deep architecture, is adapted and fine-tuned to classify ASD based on image features. Similarly, the Inception Net and Efficient Net architectures are utilized, leveraging their ability to capture intricate patterns in images while mitigating computational complexity.

Additionally, the project introduces a custom hybrid model that enhances the EfficientNet-B0 base model with extra classification layers designed specifically for detecting ASD. This hybrid model<sup>32</sup> aims to achieve the highest possible accuracy and efficiency in classifying ASD. The experimental results show that the proposed deep learning models are highly effective at accurately identifying individuals with ASD.

12

To evaluate the performance of each model, metrics such as accuracy, precision, recall, and F1-score<sup>43</sup> are used, highlighting the strengths and weaknesses of each approach. Overall, this project advances the use of sophisticated machine learning techniques in healthcare, especially for neurodevelopmental disorders like ASD.

By automating the diagnostic process, these deep learning models can improve clinical workflows, reduce diagnostic errors, and enable timely interventions, ultimately enhancing the quality of life for individuals with ASD and their families.

## Introduction

42

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental condition marked by ongoing challenges in social interaction, communication, and repetitive or restricted behaviors. Although the exact causes of ASD are not fully known, research suggests it arises from a mix of genetic, environmental, and neurological factors. Possible contributors include genetic tendencies, prenatal influences like maternal infections or exposure to harmful substances, and early brain development issues.

ASD is usually diagnosed in early childhood, often by the age of two or three, although some children may be diagnosed later. Early detection is essential for providing interventions that can significantly enhance outcomes for those with ASD. However, the timing of diagnosis can vary, depending on factors like the severity of symptoms and the availability of healthcare services.

8

While ASD cannot be cured, there are numerous treatments and interventions designed to help manage symptoms and support those affected. These can include behavioral therapies, speech and language therapy, occupational therapy, and medications to address specific issues such as anxiety or aggression.

Globally, the prevalence of ASD has been on the rise, with official government records reflecting this trend. In India, where healthcare infrastructure and awareness of ASD have been growing, the reported prevalence rates have increased in recent years. According to government data, the prevalence of ASD in India has risen steadily, paralleling global trends. Efforts to enhance awareness, improve diagnostic capabilities, and expand support services for individuals with the ASD and their families they undergoing.

The application of deep learning (DL) models in ASD detection offers a promising avenue for early identification & intervention. DL models utilize advanced machine learning techniques to analyze medical images and identify patterns indicative of ASD. By automating the diagnostic process, DL models can assist healthcare professionals in accurately and efficiently identifying individuals with ASD, facilitating timely intervention and support.

The accuracy of DL models in ASD detection is paramount for their clinical utility. Several state-of-the-art DL architectures, including VGG-16, Inception Net, and Efficient Net B0 and B7, have been evaluated for their efficacy in ASD classification. Notably, the hybrid EfficientNet B0 model has achieved the highest accuracy rate of 94.82% in assessment. Understanding the performance of these models is crucial for informing clinical decision-making and guiding future research endeavors aimed at improving ASD detection and intervention strategies.

## Literature Survey

In the ongoing pursuit of reliable methods for detecting (27) (ASD) in the facial images, researchers have turned their attention to a vast array of machine learning and deep learning techniques. This review delves deeper into the methodologies and outcomes of three pivotal studies in this field, illuminating the advancements and challenges we face in detecting ASD.

53

### A Deep Convolutional Neural Network-Based System for Detecting Autism Spectrum Disorder from Facial Images (Study 1)

The first study, titled "A Deep Convolutional Neural Network based Detection System for Autism Spectrum Disorder in Facial Images," ventures into the machine learning. It explores a spectrum of powerful techniques ASD detection through facial image analysis. These techniques include:

- Deep Neural Network (DNN) Classifier: This classifier utilizes a multi-layered artificial neural network architecture to analyze and classify facial features potentially associated with ASD.
- (LSTM) Network: This LSTM is adept at handling sequential data, making it potentially valuable in analyzing subtle temporal dynamics of facial expressions that might be indicative of ASD.
- Random Forest Classifier: This classifier combines the predictive power of multiple decision trees, improving the overall accuracy and robustness of the detection system.
- Support Vector Machine (SVM) Classification: This technique identifies hyperplanes in multidimensional space that effectively separate data points belonging to different classes (ASD and non-ASD in this case).
- Artificial Neural Network (ANN) Classifier: Similar to DNN classifiers, ANNs are inspired by the structure of brain and also function of the human brain, enabling them to learn complex patterns within facial images that might be linked to ASD.

By leveraging this diverse toolbox of machine learning algorithms, the study achieved an impressive accuracy of 85.3% in detecting ASD from facial images. Additionally, the F-Static Score, which measures the balance between precision and recall, reached a promising value of 20.85. These results suggest that the proposed system has the potential to effectively identify individuals with ASD based on facial features.

95

### Autism Spectrum Disorder Detection in Children Based Transfer Learning Technique (Study – 2)

The second study, titled "Autism Spectrum Disorder Detection in Children Using Transfer Learning Techniques," explores the efficacy (19) of DL models in ASD detection. Specifically, it investigates the power of transfer learning, a technique where a pre-trained model on a large dataset is fine-tuned for a new, specific task like ASD detection in children. This approach leverages the existing knowledge encoded within the pre-trained model, potentially accelerating the development of accurate detection systems for the (13) specific population.

The study investigates the performance of several pre or implemented trained deep learning models, including Efficient Net B7, VGG-16, VGG-19, EfficientNet-B3, and Efficient Net B5, when applied to ASD detection in children using facial images. The results showcase a range of accuracies, varying from 46.50% to a highly promising 87.50%. These findings highlight the immense potential transfer learning for ASD detection, particularly in paediatric populations. It suggests that by adapting existing, powerful deep learning models, researchers can potentially develop accurate and efficient diagnostic tools for children with ASD.



However, a significant disparity in accuracy across the different models is observed. This emphasizes the importance of selecting appropriate pre-trained models and fine-tuning them effectively for the specific task of ASD detection in children. Further research is needed to optimize these techniques and achieve consistently high accuracy across diverse datasets.

#### 71 Autistic Spectrum Disorder and Screening: Prediction with Machine Learning Models (Study 3)

The third study, titled "Autistic Spectrum Disorder Screening: Prediction with Machine Learning Models," delves the realm of ML algorithms and their potential for ASD screening. It focuses on the predictive capabilities of several established algorithms, including:

- Decision Tree: This algorithm employs a tree-like structure with branching conditions based on specific features. In this context, the features could be extract from facial images or other data sources relevant to ASD diagnosis. By following the decision tree's branches, the model arrives at a prediction of ASD presence or absence.
- Random Forest: Similar to Study 1, this study utilizes the Random Forest algorithm. As mentioned before, it combines the predictions of multiple decision trees, leading to improved overall accuracy and robustness.
- Logistic Regression: This algorithm establishes a mathematical relationship between independent variables (facial features, for instance) and the dependent variable (presence or absence of ASD). By analyzing this relationship, the model can predict the likelihood of an individual having ASD.
- Support Vector Machine (SVM): As mentioned in Study 1, SVMs are powerful tools for classification tasks. They can be employed here to effectively distinguish between individuals with and without ASD based on the extracted data.

This study investigated the performance of these algorithms in predicting ASD using various datasets. These findings underscore the valuable role that machine learning models can play in the early detection and screening of Autistic Spectrum Disorder, highlighting their potential to enhance diagnostic accuracy and support clinical decision-making.

#### Exploring Autism Spectrum Disorder Detection through Machine Learning (Study -

479 (ASD) is a developmental condition known for difficulties in social interaction and communication. Although signs usually emerge within the first two years of life, ASD can be identified at any age. Diagnosis commonly relies on behavioral evaluations conducted by trained experts.

This review explores the potential of machine learning (ML) techniques for ASD prediction and analysis across different age groups (children, adolescents, and adults). The increasing application of ML in medical diagnosis motivates this investigation, with the aim of potentially streamlining or complementing traditional assessment methods.

Several prominent ML algorithms are considered for this purpose, including:

- Naïve Bayes: This probabilistic classifier is known for its simplicity and efficiency, making it a good candidate for initial exploration.



- Support Vector Machine (SVM): This powerful classification algorithm is adept at identifying patterns in high-dimensional data, potentially useful for analyzing complex diagnostic features.
- Logistic Regression: This technique establishes a mathematical relationship between independent variables and a binary dependent variable (ASD presence or absence). Analyzing this relationship allows for prediction of ASD likelihood.
- K-Nearest Neighbors (KNN): This method categorizes data points by comparing them to similar examples with known labels in the training dataset.
- Neural Network (NN): Inspired by the structure and function of the human brain, NNs can learn complex patterns within data, potentially uncovering hidden relationships relevant to ASD diagnosis.
- Convolutional Neural Network (CNN): A specialized type of NN particularly adept at image recognition, potentially valuable if the analysis incorporates facial image data.

The effectiveness of these techniques will be evaluated on three publicly available datasets:

- Dataset 1: ASD Screening in Children (292 instances, 21 attributes)
- Dataset 2: ASD Screening in Adults (704 instances, 21 attributes)
- Dataset 3: ASD Screening in Adolescents (104 instances, 21 attributes)

### Objective

The objective of the project "ASD Detection using Deep Learning Models" is to develop and evaluate advanced deep learning (DL) architectures for the accurate and efficient identification of Autism Spectrum Disorder (ASD) from medical images. This project focuses on leveraging state-of-the-art DL models to enhance early diagnosis and intervention, ultimately aiming to improve outcomes for individuals with ASD.

The first goal is to compile a comprehensive dataset of medical images from individuals with and without ASD. This dataset will undergo data augmentation techniques such as rotation, shifting, shearing, flipping, and zooming to increase its diversity and robustness, thereby reducing the risk of overfitting during model training.

Several prominent deep learning models will be adapted and fine-tuned for ASD classification, including VGG-16, Inception Net, and EfficientNet (B0 and B7). Each model's computational efficiency and accuracy will be assessed, considering factors such as training time, memory usage, and resource requirements. A custom hybrid model based on the EfficientNet-B0 architecture will be developed, integrating additional classification layers specifically tailored for ASD detection. This hybrid model aims to maximize both accuracy and computational efficiency, making it suitable for practical clinical applications integrated limited computational resources.

96

Performance evaluation will utilize key metrics such as accuracy, precision, recall, and F1-score to determine each model's effectiveness in identifying ASD. A comparative analysis will identify the most efficient and accurate models, providing insights into their strengths and limitations for clinical use. The project will highlight the computational advantages of the hybrid EfficientNet-B0 model, which achieves highest accuracy (94.82%) But using fewer computational resources compared to

other models. This emphasis on computational efficiency is crucial for real-world clinical settings, where resource constraints can impact the feasibility of deploying advanced DL models.

Ultimately, the project aims to facilitate the adoption of DL models in healthcare by demonstrating their potential to automate and streamline the ASD diagnostic process, reducing diagnostic errors and enabling timely interventions. This work contributes to ongoing efforts to improve diagnostic capabilities and support & services for every child with ASD and their families, providing a foundation for future research aimed at enhancing ASD detection and intervention strategies through advanced machine learning techniques.

## Convolution Neural Networks (CNN)

### Introduction to CNNs

(CNNs) are a type of advanced deep learning model inspired by the structure and function of the human visual system. They excel at tasks like recognizing images, classifying objects, and performing other computer vision tasks because they can learn complex patterns and features from visual data in a hierarchical manner.

### Neurons and Layers in CNNs

CNNs are centered around neurons, which are the basic computational units responsible for receiving input signals, conducting transformations (usually weighted sums), and generating output signals. These neurons are structured into layers, creating a network that analyzes input data through various stages of feature extraction and abstraction. CNNs consist of different types of layers, each serving a distinct function within the network architecture.

### Convolutional Layers

Convolutional layers are the backbone of CNNs, providing the groundwork for their operations. In these layers, convolution operations are applied to input images using adaptable filters or kernels. These filters move across the input image, computing dot products to generate feature maps containing spatial information. Convolutional layers are pivotal in detecting local features such as edges, textures, and shapes, which are essential for understanding and interpreting visual data.

### Pooling Layers

Following convolutional layers, pooling layers come into play to downsample the feature maps, shrinking their spatial dimensions while retaining crucial features. This down sampling aids in reducing the computational load of the network and enhances its ability to recognize patterns regardless of their exact location. Pooling can be achieved through various methods, such as max pooling, which picks the highest value in a pooling window, or average pooling, which calculates the mean value.

Overall, pooling layers contribute significantly to the network's capacity to identify objects irrespective of their precise placement within the image.

## 26 Fully Connected Layers

Fully connected layers, sometimes referred to as dense layers, establish connections between each neuron in one layer and every neuron in the following layer. These layers play a vital role in carrying out higher-level reasoning and decision-making processes based on the features extracted by preceding layers. In the concluding stages of a CNN, fully connected layers amalgamate all the acquired features to generate the ultimate output, commonly employed for classification endeavors.

## Training CNNs

94 During the training phase of CNNs, the parameters of the layers—such as weights and biases—are fine-tuned using optimization algorithms. These algorithms, like stochastic gradient descent (SGD) and its adaptations such as Adam and RMSprop, iteratively adjust these parameters to minimize a loss function. This loss function measures the disparity between the predicted outputs and the actual labels, guiding the network towards improved accuracy. Through this iterative process, the model gradually enhances its performance on the specified task.

## Impact and Applications of CNNs

44 CNNs have had a profound impact on the field of computer vision, driving significant advancements in image recognition systems. They have been instrumental in achieving state-of-the-art results in various applications, from autonomous vehicles to medical image analysis.

## Notable CNN Architectures

Several notable CNN architectures have emerged, each with unique characteristics and contributions to the field. Three prominent examples include VGG, InceptionNet, and EfficientNet.

### VGG (Visual Geometry Group)

The VGG architecture stands out in the realm of deep CNNs for its straightforwardness and consistency. It comprises numerous convolutional layers paired with max-pooling layers, concluding with fully connected layers tailored for classification tasks. A pivotal aspect of VGG's design philosophy is its reliance on compact convolutional filters, typically sized at 3x3, which are layered to construct more intricate representations of visual attributes. While this approach yields remarkable accuracy by harnessing depth, it comes with heightened demands on computational power and memory resources.

### InceptionNet (GoogLeNet)

InceptionNet, also known as GoogLeNet, introduced a novel approach with inception modules. These modules perform parallel convolutions with different filter sizes (e.g.,

1x1, 3x3, and 5x5) and concatenate the results. This design allows the network to capture features at multiple scales efficiently and reduces the number of parameters compared to traditional CNN architectures. InceptionNet's innovative structure enables it to achieve high performance with a relatively lower computational burden. EfficientNet

EfficientNet introduces a series of CNN architectures that strike a harmonious balance between model complexity and computational demands via a systematic scaling technique. In contrast to conventional scaling methods that indiscriminately augment the network's depth, width, or resolution, EfficientNet adopts a compound scaling strategy. This approach uniformly scales all dimensions in a coordinated fashion, resulting in exceptionally efficient and proficient models. By optimizing the architecture to prioritize both accuracy and efficiency, EfficientNet attains cutting-edge performance standards.

### The Role of Pretrained Models

Pretrained models are deep learning models that have already been trained on large-scale datasets for specific tasks, such as image classification, object detection, or natural language processing (NLP). These models are trained on vast amounts of labeled data to learn patterns, features, and representations useful for the task at hand.

### Advantages of Employing Pretrained Models

1. Transfer Learning: Pretrained models offer the opportunity for transfer learning, where they can be fine-tuned on new, often smaller datasets tailored to specific tasks. This approach drastically reduces the time and computational resources needed for training, as it capitalizes on the knowledge gained from the original task to enhance performance on the new one.
2. Performance: Pretrained models typically exhibit impressive performance levels on established benchmarks, thanks to their extensive training on vast and diverse datasets. This reliability makes them a preferred choice for constructing robust and precise applications.
3. Efficiency: Leveraging pretrained models can streamline the data collection and annotation process, as well as conserve computational resources required for training. This efficiency proves invaluable for tasks where obtaining labeled data is challenging or costly.

### Utilizing Pretrained Models Trained on ImageNet

In our project, we employ three pretrained models—VGG-16, InceptionNet, and EfficientNet—all of which have been trained on the ImageNet dataset. ImageNet is a comprehensive dataset comprising millions of annotated images spanning thousands of categories. It has significantly propelled computer vision research forward, setting new benchmarks in image classification and object recognition tasks. Models pretrained on ImageNet demonstrate strong generalization capabilities across a wide array of visual tasks, making them an ideal choice for transfer learning applications.

Summary:

35 Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision by providing powerful tools for image recognition, classification, and other related tasks. The hierarchical nature of CNNs, with convolutional, pooling, and fully connected layers, allows them to learn and abstract features at multiple levels, making them highly effective for visual data processing. The use of pretrained models, particularly those trained on large-scale datasets like ImageNet, further enhances their applicability and efficiency, enables the development of robust and accurate deep learning applications. Models like VGG, Inception Net, and Efficient Net exemplify the diverse approaches to CNN architecture design, each contributing to the advancement of the field in unique ways.

#### **Convolution Techniques:**

Model	Convolution Techniques Used
VGG-16	Standard Convolution, Depthwise Separable Convolution
Inception V3	Inception Modules (1x1, 3x3, 5x5 convolutions), Deep and wide Separable Convolution
EfficientNet B0	Standard and Convolution, Depth wise Separable Convolution
EfficientNet B7	Standard Convolution, Depthwise Separable Convolution

#### **Methodology**

We choose facial photos because, given the advancements in neural networks, using the structural information found in these images would seem to be a practical way to achieve a rapid and inexpensive test rather than relying on difficult and costly physical measurement procedures. The children's faces in the dataset are used. There are two classes in it, photographs of children with autism are found in autistic class, while photographs of the children without autism diagnosis are found in the non-autistic class.

Three segments have been created from the dataset: train, validation, and test. The test folder has two subfolders named "autistic" and "non-autistic," which hold the images required to test the model once it has been trained. A total of 100 JPG photos, each measuring 244x244x3, are present in each subfolder. Similar to the test folder, the train folder is organised into subfolders, each of which holds 1263 photos. The valid folder is organized in the same way as the test folder and contains photos used in the model's training to gauge its validation performance.

The experiment's steps are outlined in brief below:



### 1. Data Pre-Processing:

- The dataset consists of 2536 images for the training, 100 images for the validation, and 300 images for the testing, distributed across two classes: "autistic" and "non\_autistic."
- The best pixel fits for each deep learning (DL) model and the dimensions used for image input are as follows:
  - VGG-16: Input size - 224x224 pixels
  - InceptionNet: Input size - 224x224 pixels
  - EfficientNet B0: Input size - 224x224 pixels
  - EfficientNet B7: Input size - 224x224 pixels

### 2. Data Validation:

- The dataset is validated by mapping the class labels to numerical values:
  - 'non\_autistic': 0
  - 'autistic': 1
- This mapping ensures uniformity in class representation across the dataset and facilitates model training.

### 3. Data Augmentation:

- Different data augmentation techniques are applied to enhance the diversity and robustness of the dataset for each DL model.
- For VGG-16, InceptionNet, and EfficientNet B0 models, the following augmentation techniques are used:
  - Rotation Range:  $\pm 40$  degrees
  - Width Shift Range:  $\pm 20\%$  of total width
  - Height Shift Range:  $\pm 20\%$  of total height
  - Shear Range:  $\pm 20\%$
  - Horizontal Flip: Enabled
  - Fill Mode: Nearest Neighbor
  - Zoom Range:  $\pm 20\%$
- These augmentation techniques help the models learn invariant features and improve generalization by exposing them to variations in the training data.
- For the EfficientNet B7 model, a more extensive set of augmentation techniques is utilized:
  - Rotation Range:  $\pm 40$  degrees
  - Width Shift Range:  $\pm 20\%$  of total width
  - Height Shift Range:  $\pm 20\%$  of total height
  - Shear Range:  $\pm 20\%$
  - Zoom Range:  $\pm 20\%$
  - Horizontal Flip: Enabled
  - Vertical Flip: Enabled
  - Brightness Range: 0.7 to 1.3
  - Channel Shift Range:  $\pm 50.0$
  - Fill Mode: Nearest Neighbor
- These augmentation techniques are more comprehensive and suitable for a deeper and more complex model like EfficientNet B7, helping to prevent overfitting and improve model performance.

For the hybrid EfficientNet B0 model, similar to the other DL models, data augmentation techniques are crucial for enhancing model performance and generalization. However, due to the hybrid nature of this model, which combines the EfficientNet B0 base with additional classification layers, the choice of augmentation techniques may slightly differ. Here's how data augmentation is applied for the hybrid EfficientNet B0 model:

#### Data Augmentation:

- The hybrid EfficientNet B0 model combines traditional augmentation techniques with regularization methods to counter overfitting and enhance model resilience.
- Like other models, rotations, shifts, shears, flips, and zooms are incorporated to diversify the training data and facilitate the model in learning invariant features.
- Additional regularization tactics like dropout and weight decay might be integrated within the classification layers to bolster the model's capacity to generalize to novel data.
- The details of data augmentation parameters, such as rotation range, shift range, and flip modes, can be fine-tuned based on the dataset's characteristics and the model's performance during training.

Data Augmentation Technique	Description
Rotation	Rotate the image by a random angle.
Horizontal turn or Flip	Flip an image horizontally with a certain probability.
Vertical turn or Flip	Flip the image vertically with a certain probability.
Width turn or Shift	Shift the image horizontally by a fraction of its width.
Height turn or Shift	Shift the image vertically by a fraction of its height.
Shear	Apply shear transformation to the image.
Zoom	Zoom into the image by a random factor.
Brightness of the images	Adjust the brightness of the image.
Contrast of images	Adjust the contrast of the image.

4. Data Normalization: - Data normalization is the process of rescaling input data to have an zero mean and unit variance because of which improves model stability and convergence during training in deep learning.

Below are different Normalization Techniques in ASD Detection:

Normalization Techniques Used	Description
Min-Max Scaling	Scale & pixel values to range [0,1]
Z-Score Standardization	Scale & pixel values to have zero mean and unit variance

Mean Subtraction	Subtract the mean pixel value from each pixel
Unit Variance Scaling	Scale pixel values to have input variance.

These Normalization and Argumentation techniques are applied to the input images during the preprocessing stage before feeding them into the deep learning models for training.

They play a crucial role in improving performance and generalization ability of models.

Proposed Method: VGG-16

10

#### 1. Introduction to VGG-16:

- VGG-16 is a convolutional neural network (CNN) architecture developed by the Visual Geometry Group at the University of Oxford, renowned for its simplicity and effectiveness in image classification tasks.

- The "16" in VGG-16 signifies the total number of layers, consisting of 13 convolutional layers and 3 fully connected layers.

- VGG-16's appeal lies in its uniform design, where convolutional layers are stacked sequentially with small 3x3 filters, followed by max-pooling layers for downsampling.

#### 2. Data Input:

58

- In our project, VGG-16 ingests image data categorized into "autistic" and "non\_autistic" classes from the dataset.

- Each image is represented as a pixel value matrix, typically resized to 224x224 pixels to match VGG-16's expected input size.

#### 3. Model Prediction:

- VGG-16 predicts the output class (either "autistic" or "non\_autistic") for each input image through forward passes.

- During inference, input images undergo convolutional and pooling operations, followed by flattening and processing through fully connected layers.

- The final softmax layer outputs a probability distribution over classes, with the highest probability class considered the prediction.

#### 4. Training Parameters:

- Backpropagation, a supervised learning method, trains VGG-16 by adjusting weights based on predicted and actual outputs.

- Convolutional layers extract hierarchical features from input images during training, followed by aggregation and classification through fully connected layers.

- Tuned training parameters include learning rate, batch size, epochs, and optimization algorithm (e.g., RMSprop or Adam) to optimize performance and prevent overfitting.

#### 5. VGG Architecture and Efficiency:



38

- VGG-16 comprises 13 convolutional layers and 3 fully connected layers, with interleaved max-pooling layers for spatial downsampling.
- Using small 3x3 convolutional filters facilitates effective learning of spatial hierarchies, enhancing performance.
- Pooling layers reduce feature map dimensions while retaining essential features, enabling capture of both local and global information.
- VGG-16's uniform design and straightforward architecture enhance understandability and implementation, contributing to its efficiency in various image classification tasks.
- However, its extensive parameter count results in relatively high computational costs, potentially limiting deployment in resource-constrained environments. Nevertheless, its accuracy and robustness make it valuable where computational resources permit.

83

The VGG-16 model, with its hierarchical feature extraction and classification capabilities, is a potent tool in image classification tasks. Its simplicity, effectiveness, and ability to discern complex image patterns make it indispensable in computer vision and image analysis.

33

#### 6. Activation Function:

- The activation function decides whether a neuron should be activated or not after computing the weighted sum and adding bias.
- Its purpose is to introduce non-linearity to a neuron's output, enabling neural networks to learn complex patterns effectively.

Output Layer Activation Function:

Softmax Activation Function: Defined as:

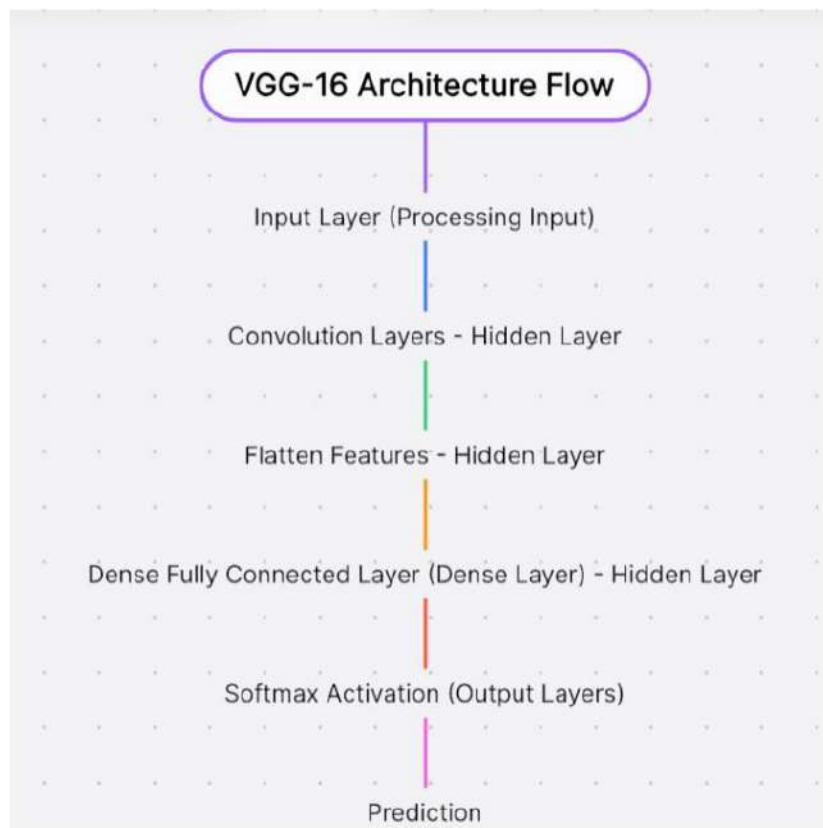
$$(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Code Snippet:

```
output = Dense(95, activation='softmax')(class1)
```

- Dense: The Dense layer, a fundamental component in neural networks, signifies a fully connected layer wherein every neuron in the layer links to each neuron in the preceding layer. This layer executes a linear operation, followed by activation function application.
- 95: In this context, 95 denotes the quantity of units or neurons in the output layer. Each unit in the output layer corresponds to a class in a multi-class classification task. For instance, if there are 95 classes, then the output layer will encompass 95 neurons, with each representing the probability of the input belonging to a specific class.
- class1: class1 designates the input to the Dense layer. It emerges as the output of a preceding layer or some intermediary computation within the neural network. The Dense layer receives this input, conducts a linear transformation, and then

applies the activation function. <sup>13</sup> softmax activation function to produce the final output, which is a probability distribution over the classes.  
**VGG-16 Architecture Flow:**



*Description: - The VGG-16 architecture initiates with an input layer dedicated to image processing. Subsequently, the model traverses multiple convolutional layers for feature extraction a flatten layer that converts to 2D feature maps into a 1D feature vector passed through dense fully connected layers for advanced processing, concluding with a softmax activation layer for classification, which yields the final prediction.*

### **InceptionNet:**

Methodology of InceptionNet Model:

#### **1. Introduction to InceptionNet (Go<sup>15</sup>LeNet):**

- InceptionNet, also referred to as GoogLeNet, is a convolutional neural network architecture developed by researchers at Google. It stands out for its deep and broad design, meticulously crafted to capture a wide array of features across varying scales within images.

- InceptionNet introduces the innovative concept of "inception modules," comprising multiple parallel convolutional operations employing distinct filter sizes. This ingenious setup enables the network to adeptly capture both local and global features present in the input data.

- The overarching goal of this architecture is to strike a delicate balance between computational efficiency and representational prowess. By efficiently extracting features while conserving computational resources, InceptionNet ensures effective utilization in various applications requiring sophisticated image analysis.

## 2. Data Input:

- In this project, the InceptionNet model reads image data from the dataset containing images classified into "autistic" and "non\_autistic" classes.

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to 224x224 pixels to match the input size expected by the InceptionNet architecture.

## 3. Model Prediction and Training:

InceptionNet, like any other neural network model, uses both forward propagation and backward propagation during the training process.

### i. Forward Propagation:

- During forward propagation, the input data, typically images, traverses the network layer by layer in a forward direction.

- Each layer applies a series of mathematical operations to transform the input data and generate an output.

- InceptionNet's forward propagation entails guiding the input image through various inception modules and pooling layers, extracting features across different scales and spatial resolutions.

- Ultimately, the output of the final layer furnishes the predicted class probabilities for the input image.

### ii. Backward Propagation:

- Subsequent to forward propagation, during the training phase, the model's predictions are juxtaposed with the actual labels of the input images to compute the loss or error.

- Backward propagation, alias backpropagation, is subsequently employed to adjust the model's weights with the objective of minimizing this loss.

- This procedure entails computing the gradients of the loss function concerning each parameter in the network, employing the chain rule of calculus.

- These gradients are then utilized to refine the parameters (weights and biases) of the network in a direction opposite to that of the gradient, thereby effectively "backpropagating" the error throughout the network.

- By iteratively fine-tuning the model's parameters grounded on the calculated gradients, the model gradually enhances its predictive accuracy over successive iterations.

So, to clarify, InceptionNet utilizes both forward propagation (for prediction) and backward propagation (for training) during the training process. Forward propagation

is used to make predictions, while the backward propagation are used to update the model's parameters based on the computed errors.

#### 4. InceptionNet Architecture and Efficiency:

- The InceptionNet architecture consists of multiple inception modules stacked sequentially, interspersed with max-pooling layers for spatial downsampling.
- Each inception module incorporates multiple convolutional operations with different kernel size are (1x1, 3x3, and 5x5), allowing model to capture features different scales efficiently.
- The use of parallel convolutional operations within inception modules enables the model to extract diverse and informative features while minimizing computational cost.

- Inception Net's efficient architecture strikes an balance between the computational efficiency and representational of power, making it suitable for tasks where both accuracy and efficiency are important considerations.

Despite its deep and wide architecture, Inception Net achieves competitive performance on image classification tasks with relatively fewer parameters compared to other architectures, making it a popular choice for various computer vision applications.

In summary, the InceptionNet model is an the deep and convolutional neural network architecture is designed to efficiently capture multi-scale features within images using inception modules. Its balanced architecture efficient feature extraction, and competitive performance make it a valuable tools for image classification tasks, including the detection of the autism spectrum disorder in this project.

#### 5. Output Layer Activation Function:

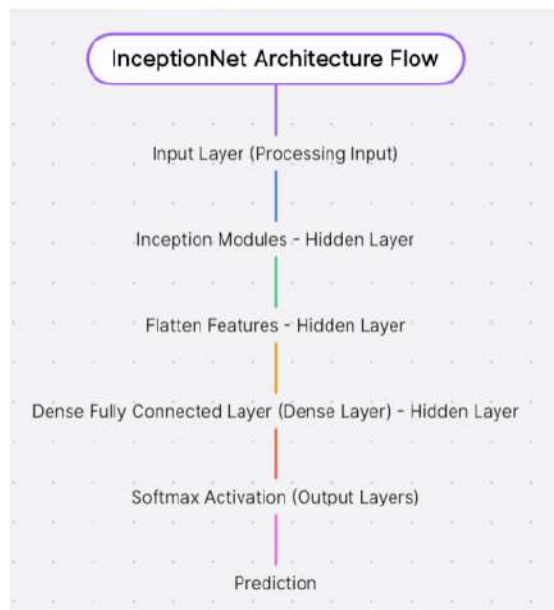
Softmax Activation Function: Defined as:

$$(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Code Snippet:

```
predictions = Dense(1, activation="softmax")(x)
```

#### Inception Net Architecture Flow:



*Description: - Inception v3, a convolutional neural network, starts with an input layer for images. It then utilizes multiple Inception modules, which are like mini-networks that extract features at various scales. After these modules, a flatten layer<sup>40</sup> transforms the extracted features into the single vector. This vector fed into fully-connected layers for high-level reasoning, and finally a softmax layer outputs probabilities for different object categories.*

## **EfficientNet-B0**

### **1.1. Introduction to EfficientNet<sup>15</sup>**

- EfficientNet is a family of convolutional neural network architectures developed by researchers at Google. It aims to achieve state-of-the-art performance while maintaining efficiency in terms of computational resources.

- EfficientNet introduces a novel compound scaling<sup>1</sup> method that uniformly scales the network width, depth, and resolution, resulting in models that are both highly accurate and computationally efficient.

- The architecture balances between model size and accuracy by optimizing the trade-off between network depth, width, and resolution, leading to models that achieve superior performance with fewer parameters

### **2. Data Input:**

- In this project, the EfficientNet model reads image data from the dataset containing images classified into "autistic" and "non\_autistic" classes.

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to match the input size expected by the EfficientNet architecture.

### **3. Model Prediction:**



- EfficientNet predicts the output class (either "autistic" or "non\_autistic") for each input image through forward propagation.

- During forward propagation, input image undergoes an series of convolutional operation within EfficientNet architecture, allowing the model to extract hierarchical features.

- The final output of the softmax layer presents a probability distribution across all classes, where each class is assigned a probability score. Subsequently, the class exhibiting the highest probability is identified as the predicted class for the input image.

#### 4. Training Parameters:

- The EfficientNet model is trained using backpropagation, similar to other deep learning architectures, where the model's weights are adjusted based on the error and between predicted the actual outputs.

- During the training, the model learns to extract hierarchical features from input images through the convolutional layers, which are then aggregated and classified into different categories through fully connected layers.

- The training parameters include the learning rate, batch size, number of epochs, and optimization algorithm (e.g., RMSprop or Adam), which are tuned to optimize model performance on the training dataset while preventing overfitting.

#### 5. Efficient Net Architecture and Efficiency:

- The EfficientNet architecture integrates numerous convolutional layers with an innovative approach to scaling network width, depth, and resolution efficiently.

- Employing a compound scaling technique, EfficientNet uniformly adjusts the network's width, depth, and resolution to strike an optimal balance between model size and accuracy.

- This architecture adopts depthwise separable convolutions, a technique that significantly reduces parameter count and computational overhead while preserving expressive capabilities.

- EfficientNet outperforms its counterparts by delivering superior performance with fewer parameters, showcasing remarkable efficiency in computational resource utilization.

Leveraging efficient scaling and depthwise separable convolutions, EfficientNet achieves state-of-the-art results across diverse image classification tasks, making it a favored option for environments with limited computational resources.

In summary, the EfficientNet model is a highly efficient convolutional neural network architecture that achieves state-of-the-art performance on image classification tasks with fewer parameters and computational resources.

Its compound scaling method and depthwise separable convolutions enable it to strike a balance between model size and accuracy, making it suitable for various computer vision applications, including the detection of autism spectrum disorder in this project.

#### Output Layer Activation Function:

Sigmoid Activation Function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Code Snippet:  
predictions = Dense(1, activation="sigmoid")(x)

## **EfficientNet – B7**

Methodology of Efficient Net B7 Model:

### 1. Introduction to Efficient Net B7:

- Efficient Net B7 is CNN architecture developed Google, belonging to the Efficient Net family. It is one of the largest variants in the series, EfficientNet is meticulously crafted to excel in image classification tasks while maximizing the utilization of computational resources, aiming to achieve state-of-the-art performance.

- The architecture of Efficient Net B7 is characterized by its depth, width, and resolution scaling, which are optimized using a compound & scaling method balance model size & accuracy.

- Efficient Net B7 incorporates depth wise separable convolutions and efficient feature extraction techniques, enabling it to capture complex patterns and hierarchical features from input images efficiently.

### 2. Data Input:

- The EfficientNet B7 model reads image data from the dataset containing images categorized into "autistic" and "non\_autistic" classes.

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to match the input size expected by the EfficientNet B7 architecture.

### 3. Model Prediction:

- EfficientNet B7 undertakes the prediction of output classes, namely "autistic" or "non\_autistic," for each input image through forward propagation.

- During this propagation, the input image traverses through a series of convolutional operations within the EfficientNet B7 architecture, efficiently extracting hierarchical features.

- These extracted features are subsequently amalgamated and processed across multiple network layers, progressively learning representations that are increasingly distinctive for the target classes.

- The conclusive outcome of the model manifests as a probability distribution across classes, with the predicted class being the one exhibiting the highest probability.

### 4. Training Parameters:

- The training of the EfficientNet B7 model entails the utilization of backpropagation, a standard deep learning technique for optimizing model parameters.

- Throughout the training phase, the model's weights are iteratively adjusted based on the discrepancy between predicted and actual outputs, with the intent of minimizing a designated loss function.

- Key training parameters, including learning rate, batch size, number of epochs, and optimization algorithm (e.g., RMSprop or Adam), are fine-tuned to optimize model performance while mitigating overfitting.

### 5. EfficientNet B7 Architecture and Efficiency:

- The architecture of EfficientNet B7 encompasses numerous layers of convolutional operations, interspersed with depthwise separable convolutions and efficient scaling methodologies.
- The scalability in depth and width within EfficientNet B7 enables the capture of intricate patterns and representations from input images, while resolution scaling optimizes computational efficiency by striking a balance between model size and accuracy.
- Leveraging efficient feature extraction techniques, EfficientNet B7 achieves unparalleled performance in image classification tasks while upholding computational efficiency.
- By harnessing depthwise separable convolutions and efficient scaling strategies, EfficientNet B7 attains superior performance with reduced parameter count compared to alternative architectures, rendering it exceptionally efficient for an extensive array of computer vision applications.

In summary, EfficientNet B7 is a powerful convolutional neural network architecture optimized for efficient use of computational resources while achieving state-of-the-art performance on image classification tasks. Its depth, width, and resolution scaling, combined with efficient feature extraction techniques, make it a valuable tool for various computer vision applications, including the detection of autism spectrum disorder in this project.

### Output Layer Activation Function:

Sigmoid Activation Function:

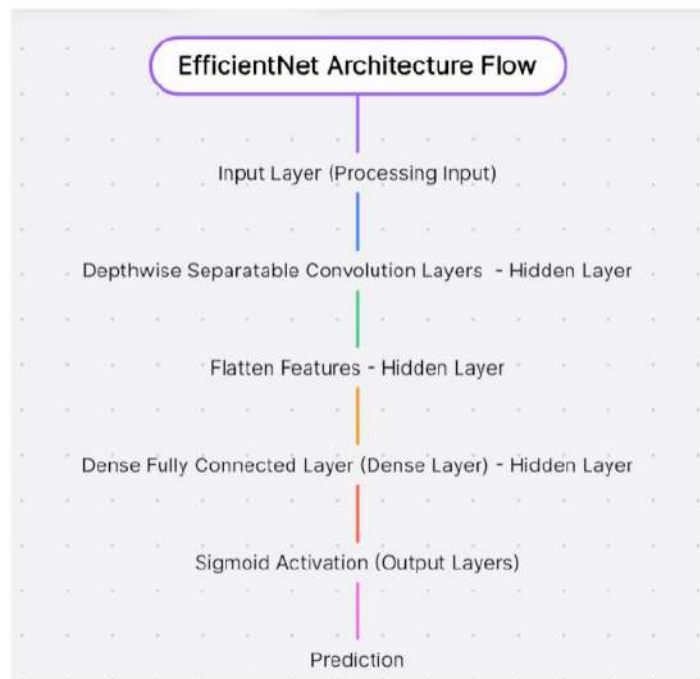
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Code Snippet:

```
output = Dense(1, activation='sigmoid')(x)
```

### **Efficient Net Architecture Flow**





*Description: - The EfficientNet architecture prioritizes processing efficiency while maintaining accuracy. It starts with an input layer that receives the image data. Data then progresses through depthwise separable convolution layers, which are essential building blocks for efficient feature extraction. Next, a flatten layer transforms the extracted features into a one-dimensional vector.*

### **Modified EfficientNet-B0 (Proposed Model)**

Methodology of Modified EfficientNet B0 Model:

#### **1. Introduction to EfficientNet B0 Hybrid Model:**

- The EfficientNet B0 Hybrid Model is a custom variation of the EfficientNet B0 architecture, enhanced with additional classification layers for improved performance on specific tasks.
- This hybrid model incorporates the efficient scaling and depthwise separable convolutions of EfficientNet B0 while adding custom classification layers to adapt the model the target of task.
- By leveraging the efficient feature extraction capabilities of EfficientNet B0 and fine-tuning the classification layers, the hybrid model aims to achieve superior performance on image classification tasks with reduced computational cost.

#### **2. Data Input:**

- Similar to other models, the EfficientNet B0 Hybrid Model reads image data from the dataset containing images categorized into "autistic" and "non\_autistic" classes.

2

- Each image in the dataset is represented as a matrix of pixel values, with dimensions typically resized to match the input size expected by the EfficientNet B0 architecture.

### 3. Model Prediction:

- The EfficientNet B0 Hybrid Model executes the prediction of output classes, "autistic" or "non\_autistic," for each input image via forward propagation.
- Throughout this propagation, the input image navigates through a sequence of convolutional operations within the EfficientNet B0 architecture, adeptly extracting hierarchical features.
- These extracted features then traverse through supplementary classification layers integrated into the model, which meticulously process the features and prognosticate the probabilities of output classes.
- Subsequently, the class exhibiting the highest probability is identified as the predicted class for the input image.

### 4. Training Parameters:

- The EfficientNet B0 Hybrid Model is trained using backpropagation, similar to other deep learning architectures.
- During training, the model's weights are adjusted based on the error between predicted and actual outputs, optimizing the model's performance on the training dataset.
- The training parameters include the learning rate, batch size, number of epochs, and optimization algorithm (e.g., RMSprop or Adam), which are tuned to optimize model performance while preventing overfitting.

25

### 5. EfficientNet B0 Hybrid Model Architecture and Efficiency:

- The EfficientNet B0 Hybrid Model architecture merges the efficiency of scaling and depthwise separable convolutions from EfficientNet B0 with supplementary classification layers.
- EfficientNet B0's efficient feature extraction capabilities enable the model to capture meaningful patterns from input images while minimizing computational resources.
- The hybrid model's additional classification layers further refine the extracted features, adapting them to the specific task of classifying autism spectrum disorder.
- By leveraging EfficientNet B0's efficiency and fine-tuning the classification layers, the hybrid model achieves superior performance on image classification tasks with reduced computational cost.
- The use of efficient scaling and depthwise separable convolutions, combined with custom classification layers, allows the EfficientNet B0 Hybrid Model to strike a balance between accuracy and efficiency, making it a powerful tool for image classification tasks, including the detection of autism spectrum disorder.

In summary, the EfficientNet B0 Hybrid Model is a custom variation of the EfficientNet B0 architecture, enhanced with additional classification layers for improved performance on specific tasks. By leveraging the efficiency of EfficientNet B0 and fine-tuning the classification layers, the hybrid model achieves superior performance on image classification tasks with reduced computational cost. Its architecture and efficiency make it a valuable tool for various computer vision applications, including the detection of autism spectrum disorder in this project.

Output Layer Activation Function:

Sigmoid Activation Function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Code Snippet:

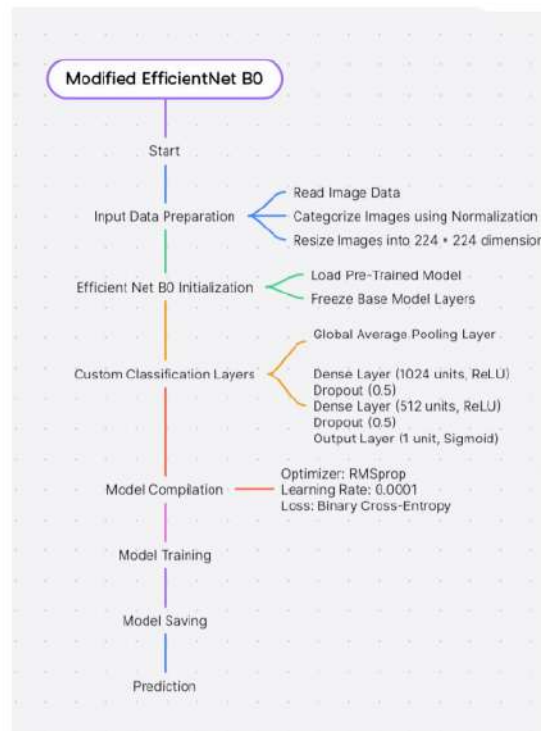
```
output = Dense(1, activation='sigmoid')(x)
```

### **Impact of Proposed Model**

Modification on Efficient B0	Impact
Freezing base model layers	- Reduces training time by focusing on the custom classification head. Potentially limits model capacity for complex tasks.
GlobalAveragePooling2D 24	- Efficiently summarizes spatial information from the base model's output.
Dense layers (1024, 512 and units with ReLU activation and Dropout)	- Introduces non-linearity and helps prevent overfitting by reducing model complexity.
Output layer (1 unit with sigmoid activation) 77	- Produces a single output between 0 and 1 for binary classification (e.g., ASD vs. Non-ASD).
RMSprop optimizer with low learning rate (0.0001)	- Slows down learning process but helps to find a minimum in the loss function and avoid overshooting.

The above methods will improve the overall image classification ability of the model.

### **Work Flow of the Modified EfficientNet B0**



*Description: - The Modified Efficient Net B0 model pipeline involves reading and normalizing image data, resizing it to  $224 \times 224$  pixels, and loading a pre-trained Efficient Net B0 with frozen base layers. Custom classification layers, including dense and dropout layers, culminate in a sigmoid-activated output. The model is compiled with an RMSprop, 0.0001 learning rate, and binary the cross-entropy loss, then trained, saved, and used for predictions.*

### **Architectural Comparison**

Here's a tabular comparison of the mentioned model architectures along with the number of layers in each:

Model	Architecture	Number of Layers
VGG-16	Sequential	16
InceptionNet	GoogLeNet-like	Varies
EfficientNet B0	Efficient Scaling	Varies
EfficientNet B7	Efficient Scaling	Varies
Modified Model	Custom	Varies

The number of layers in InceptionNet, EfficientNet B0, and EfficientNet B7 varies depending on the specific configuration (like pooling layers, activation layers, etc) and variant used in the implementation. Therefore, the exact number of layers may differ between different instances of these models.

### **Activation Functions used in Output Layers**

Model	Activation Function	Usage
1. VGG-16	Softmax	Multi-Class Classification Task
2. Inception Net	Softmax	Multi-Class Classification Task
3. EfficientNet B0	Sigmoid	Binary Classification Task
4. EfficientNet B7	Sigmoid	Binary Classification Task
5. Modified EfficientNet B0	Sigmoid	Binary Classification Task

Sigmoid activation is used in models designed for binary classification tasks (EfficientNet B0, EfficientNet B7, and Hybrid EfficientNet B0), while softmax activation is used in models designed for multi-class classification tasks (VGG-16 and Inception Net).

#### Description of Activation Functions:

Activation Function	Formula
Sigmoid	<ul style="list-style-type: none"> <li>• Sigmoid activation is commonly used in binary classification tasks, where the output represents the probability of the input belonging to the positive class.</li> <li>• It squashes the input values to the range [0, 1], which can be interpreted as probabilities.</li> <li>• Defined as: <math>\sigma(x) = \frac{1}{1 + e^{-x}}</math></li> <li>- <math>\sigma(x)</math> is the output of the sigmoid function for a given input <math>x</math>.</li> <li>- <math>e</math> is Euler's number, approximately equal to 2.71828.</li> <li>- <math>x</math> is the input to the function</li> </ul>
Softmax	<ul style="list-style-type: none"> <li>• It transforms the raw output of the neural network into a probability distribution over multiple classes, enabling the model to make predictions about the likelihood of an input belonging to each class.</li> <li>• Defined as: <math>(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}</math></li> <li>- <math>x_i</math> represents <math>i</math>th element of the vector <math>x</math></li> <li>- <math>e</math> is Euler's number, approximately equal to 2.71828.</li> <li>- The sum in the denominator is computed over all elements <math>x_j</math> of the input vector <math>x</math>.</li> </ul>

#### Workflow of the Project





### Model Compilation:

Model compilation is an essential step in training a neural network model, where you configure the model with additional parameters beyond its architecture. This step involves specifying the optimizer, loss function, and optional metrics to monitor during training and evaluation.

1. **Optimizer:** The optimizer is responsible for updating the model's weights based on the gradient of the loss function. It plays a crucial role in the training process by determining how the model learns from the data.

- **Adam Optimizer:** The Adam optimizer is a popular choice for training neural networks. It adjusts the learning rates for each parameter adaptively, based on the historical gradients. This adaptive approach facilitates efficient and effective optimization, helping the model converge faster and perform better during training.

2. **Loss Function:** The loss function measures the difference between the model's predictions and the actual target values (labels) during training. It quantifies the model's performance on a given batch of data. By minimizing the loss function, the model improves its accuracy over time.

- The choice of the loss function depends on the specific task and the nature of the data. For example, in classification tasks, cross-entropy loss is commonly used, while mean squared error is often used for regression tasks.

The model compilation involves setting up the optimizer and loss function, which are critical for the training process. The optimizer, such as Adam, ensures efficient learning by adaptively adjusting learning rates, while the loss function provides a metric for evaluating the model's performance and guiding its improvements. By carefully configuring these parameters, you can enhance the effectiveness of your neural network training process.

### Optimizer Algorithms for Each Model:

Model	Optimizer Algorithm	Learning Rate
VGG-16	Adam (Adaptive Moment Estimation)	0.001
InceptionNet	RMSprop	0.001
EfficientNet B0	RMSprop	0.001
Modified EfficientNet B0	RMSprop	0.001
EfficientNet B7	RMSprop	0.001

11

### Pre-Trained Models:

Pretrained models are deep learning models that have already been trained on extensive datasets for specific tasks, such as image classification, object detection, natural language processing (NLP), and other machine learning applications. These models have undergone extensive training on vast amounts of labeled data, enabling them to learn patterns, features, and representations pertinent to their designated tasks.

In our project, we are leveraging three prominent pretrained models: VGG-16, InceptionNet, and EfficientNet.

- VGG-16: This model is known for its simplicity and effectiveness, featuring a deep architecture with 16 layers. It has been widely adopted for image classification tasks.

- InceptionNet: Recognized for its inception modules, this model efficiently handles various scales of information within an image. Its architectural innovations help achieve high accuracy in image classification and object detection.

- EfficientNet: This model is renowned for its efficiency and scalability. It achieves high performance by optimizing the balance between depth, width, and resolution of the network.

Dataset for Pre-Trained Models:

18

The pretrained models we are using are trained on the ImageNet dataset. ImageNet is a large-scale dataset comprising millions of labeled images across thousands of categories. It is one of the most significant datasets in computer vision research and has played a crucial role in advancing the field of deep learning.

43

ImageNet encompasses a wide range of objects, scenes, and concepts, making it a comprehensive resource for training models to recognize and classify diverse images. The dataset was created with the objective of pushing the boundaries of image classification and object recognition tasks. Consequently, it has become a standard benchmark for evaluating and comparing the performance of various deep learning models.

69

Projects and research initiatives involving image classification, object detection, image segmentation, and other computer vision tasks frequently utilize ImageNet. The vast amount of labeled data it provides allows models to learn intricate details and features of images, improving their accuracy and generalization capabilities.

By using pretrained models on the ImageNet dataset, we can leverage the extensive knowledge and representations these models have acquired during their training. This approach not only saves time and computational resources but also enhances the performance of our models on our specific tasks. Instead of training models from scratch, we can fine-tune these pretrained models on our dataset, adapting their learned features to our unique requirements.

In summary, pretrained models like VGG-16, InceptionNet, and EfficientNet, trained on the comprehensive ImageNet dataset, offer a robust foundation for our project. These models bring the benefits of extensive prior learning, enabling us to achieve high performance in our specific image classification and object recognition tasks with greater efficiency.

### **Transfer-Learning**

Transfer Learning functions on the idea of reusing <sup>50</sup> knowledge gained from one task to improve performance on a related but different task.

1. VGG-16 <sup>32</sup>  
We modify the pretrained VGG-16 model by adding new classification layers on top of the existing convolutional base. The weights of the convolutional layers are frozen (not trained), while the new classification layers are trained on your specific dataset. This is a form of transfer learning where the pretrained VGG-16 model serves as a feature extractor for your task of autism detection.
2. InceptionNet  
Similar to VGG-16, we use a pretrained InceptionNet model as a feature extractor. We add new classification layers on top of the InceptionNet base and fine-tune them on our dataset. Again, the weights of the pretrained layers are frozen, and only the new classification layers are trained.
3. EfficientNet B0 <sup>88</sup>  
The same transfer learning approach is applied here, where the pretrained EfficientNet B0 model is used as a feature extractor, and new classification layers are added and trained on your dataset.
4. EfficientNet B7  
Transfer learning is also used with the pretrained EfficientNet B7 model, where the existing convolutional base serves as a feature extractor, and new classification layers are added and trained on your dataset.
5. Modified EfficientNet B0  
Similarly, transfer learning is applied to the hybrid EfficientNet B0 model, where the pretrained base model is used for feature extraction, and new classification layers are added and trained.

### **Transfer Learning Techniques:**



Model	Transfer Learning Technique
VGG-16	Feature Extraction
InceptionNet V3	Feature Extraction
EfficientNet B0	Feature Extraction
EfficientNet B7	Feature Extraction
Modified EfficientNet B0	Feature Extraction

### Model Fitting

In deep learning, "fitting" signifies the training process of a neural network on a specific dataset. This involves utilizing the `fit` method, which is a standard function in deep learning libraries like TensorFlow and Keras. The `fit` method enables you to train a model by providing the necessary training data, specifying the number of epochs, determining the batch size, and setting other relevant training parameters. The primary goal of the fitting process is for the model to fine-tune its weights and biases in response to the training data.

This fine-tuning aims to minimize a defined loss function, thereby optimizing the model's performance for the given task. Essentially, the fitting process is an iterative optimization procedure that adjusts the model parameters to improve its predictive accuracy. This optimization is conducted through a series of forward and backward passes within the neural network. During the forward pass, input data is propagated through the network, and predictions are generated. This process involves passing data through multiple layers of neurons, each applying a specific mathematical transformation to the input.

The predictions made at the end of the forward pass are then compared to the actual target values, and the difference between them is quantified using a loss function. The loss function is a critical component as it measures the model's prediction error. Commonly used loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks. The goal is to minimize this loss function, indicating that the model's predictions are becoming more accurate.

After computing the loss, the next step is the backward pass, which involves calculating the gradients of the loss function with respect to each weight in the network. This is done using a technique called backpropagation. Backpropagation applies the chain rule of calculus to propagate the gradient of the loss function backward through the network, from the output layer to the input layer. Once the gradients are computed, they are used to update the model's weights and biases.

This update is typically performed using an optimization algorithm like Stochastic Gradient Descent (SGD), Adam, or RMSprop. These algorithms adjust the weights in the direction that reduces the loss, thereby improving the model's accuracy over successive iterations. The fitting process continues for a specified number of epochs, where an epoch is defined as one complete pass through the entire training dataset. Within each epoch, the dataset is often divided into smaller batches, and the model parameters are updated after each batch. This approach, known as mini-batch gradient descent, helps in stabilizing and speeding up the training process.

Throughout training, <sup>85</sup> model's performance is periodically evaluated on a validation set, which is a separate portion of the dataset not used for training. This evaluation helps in monitoring the model's generalization ability and in tuning hyperparameters to avoid overfitting.

Therefore, the process of fitting a neural network involves training the model by adjusting its parameters to minimize a loss function, thereby enhancing its ability to make accurate predictions. This is achieved through iterative <sup>12</sup> forward and backward passes, using backpropagation and optimization algorithms to fine-tune the model's weights and biases.

<sup>56</sup>

#### **Batch Size: -**

The batch size in neural network training refers to the number of training examples processed in one iteration. Essentially, it's the count of samples that the model handles before updating its parameters using the gradients calculated from the loss function. For instance, if you set the batch size to 32, the model processes 32 training examples at a time before updating its weights and biases.

<sup>5</sup>

During training, the entire dataset is divided <sup>68</sup> into these smaller batches, and each batch is fed into the model sequentially. The choice of batch size can significantly influence the training process. A larger batch size often leads to faster convergence since more data is processed at once, which can make the gradient estimates more accurate. However, this comes at the cost of requiring more memory, which might not be feasible for very large datasets or limited hardware resources. On the other hand, a smaller batch size may result in slower convergence. This is because smaller batches provide noisier estimates of the gradient, which can introduce more fluctuations during training.

However, these fluctuations can sometimes help the model escape local minima and potentially generalize better to unseen data. Choosing the optimal batch size often involves a trade-off between computational efficiency and model performance. Larger batch sizes can make better use of modern hardware accelerators like GPUs and TPUs, leading to faster training times.

However, they might not always provide the best generalization performance. Conversely, smaller batch sizes might slow down training but could potentially lead to a model that performs better on new, unseen data. This balance makes the selection of an appropriate batch size a critical decision in the design of neural network training processes.

<sup>6</sup>

Therefore, the <sup>90</sup> batch size is a crucial hyperparameter in neural network training, determining how many examples are processed before updating model parameters. While larger batch sizes can speed up training, they require more memory and might not always yield the best generalization. Smaller batch sizes, although slower, can sometimes offer better generalization to new data. Finding the right balance is key to optimizing the training process and model performance.

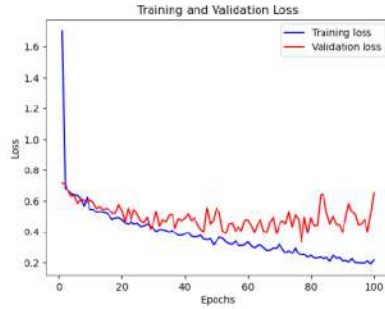
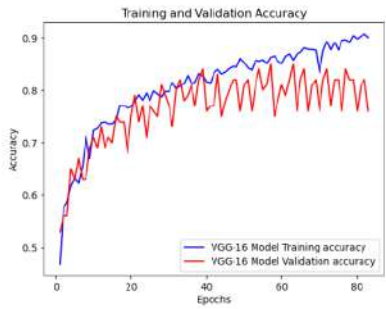
#### **Batch Size for Each Model:**

Model	Batch Size
VGG-16	128
Inception V3	128
EfficientNet B0	128
EfficientNet B7	128
Hybrid EfficientNet B0	64

55

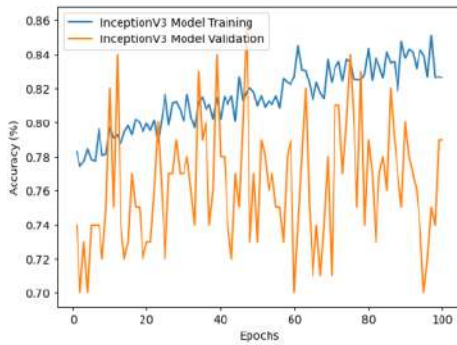
## 1. VGG-16

## Results



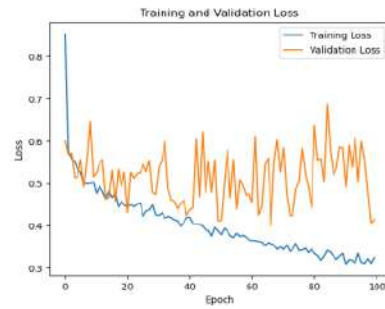
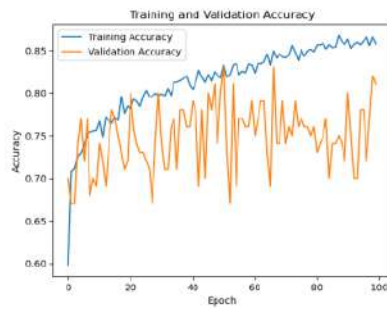
VGG-16 Training and Validation Accuracy VGG-16 Training Loss and Validation Loss

## 2. InceptionNet V3



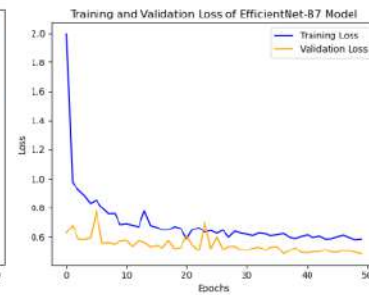
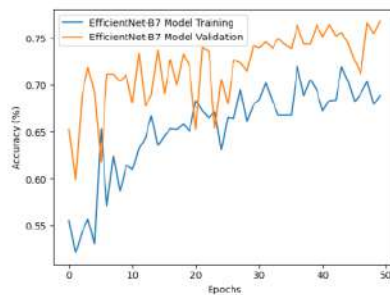
InceptionNet Training and Validation Accuracy

## 3. EfficientNet B0



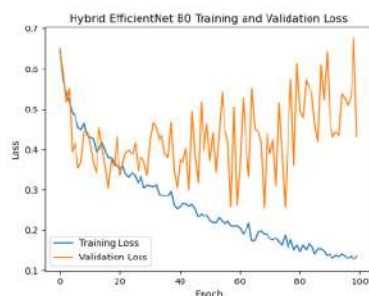
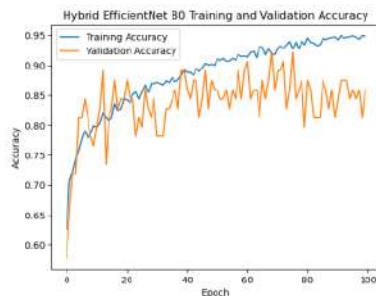
97  
Efficient Net B0 Training and Validation Accuracy Graph & Training and Validation Loss Graphs

#### 4. EfficientNet B7

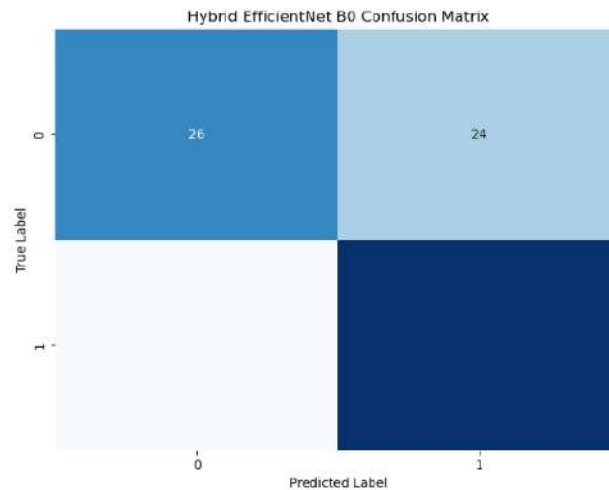


29  
Efficient Net B7 Training and Validation Accuracy Graph & Training Loss and Validation Loss Graph

#### 5. Modified EfficientNet B0



29  
Efficient Net B7 Training and Validation Accuracy Graph & Training Loss and Validation Loss Graph



*Modified EfficientNet B0 Confusion Matrix*

### Model Accuracy Comparison Table

Model	Accuracy	Validation Accuracy
1. VGG-16	90.69%	76.00%
2. Inception V3	82.65%	79.00%
3. EfficientNet B0	85.73%	81.00%
4. EfficientNet B7	68.93%	76.85%
5. Modified EfficientNet B0	94.82%	85.94%

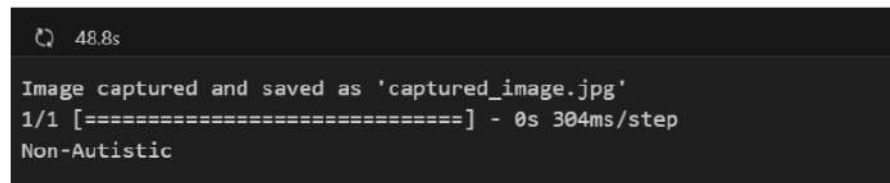
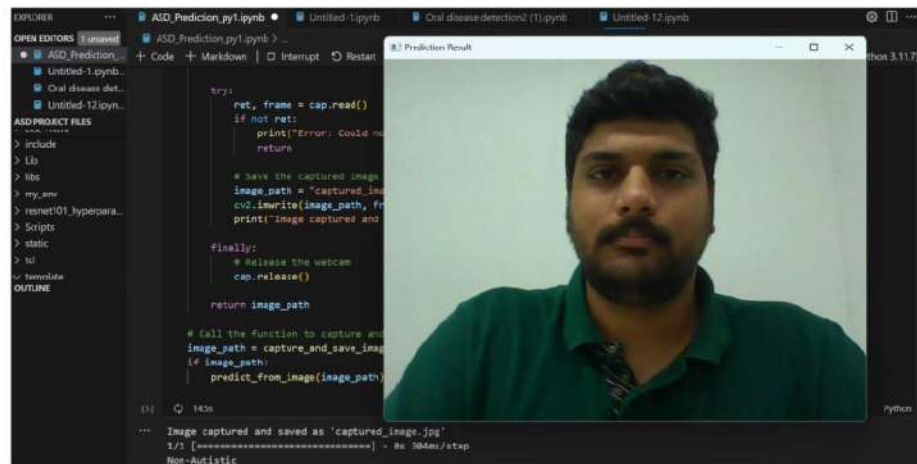
### Draw-Backs for Each Model

Model	Drawbacks	Solution
1. VGG-16 Model	- Large Size & High Computational Cost - Slow Training Time	- Transfer Learning with Pre-trained Weights - Layer Freezing to Reduce Trainable Parameters
2. InceptionV3 Model	- Complex Architecture - High Computational Cost	- Simplified Architectures for Limited Resources - Efficient Hardware (e.g., GPUs)
3. EfficientNet B0 and B7	- Requires Careful Parameter Tuning - B7 Can Be Time-Consuming to Train	- Automated Hyperparameter Tuning Tools - Model Selection Based on Resources & Needs
4. Modified EfficientNet B0	- Increased Implementation Complexity - Longer Training Time	- Gradual Unfreezing During Training - Layer-wise Learning Rates



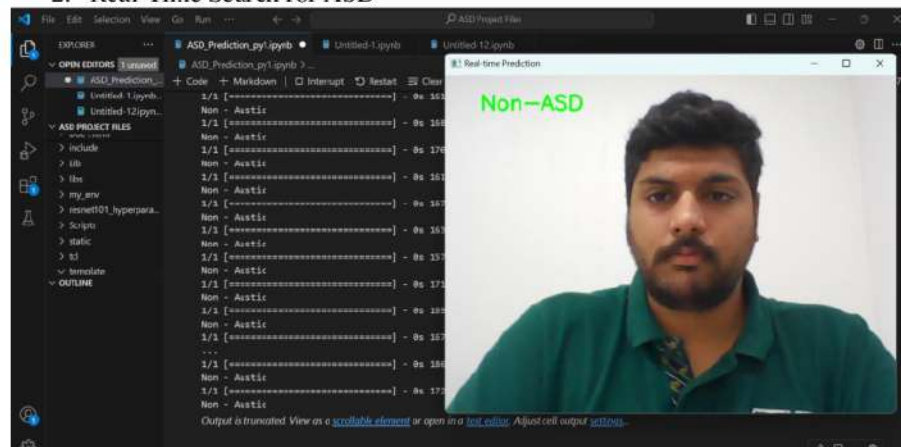
## Real-Time Outputs:

### 1. Real Time Capture



*Description: - This picture shows a real-time output of pre-trained deep neural network model. When we take a picture with the webcam, the code feeds the image to this model. The model then analyses the image, likely focusing on facial features or other visual cues, and attempts to predict the likelihood of ASD in the person depicted.*

### 2. Real-Time Search for ASD



```

1/1 [=====] - 0s 157ms/step
Non - Austic
1/1 [=====] - 0s 171ms/step
Non - Austic
1/1 [=====] - 0s 169ms/step
Non - Austic

```

*Description: It is the real-time ASD detection through webcam analysis. It loads a pre-trained deep learning model. The webcam captures frames, which are pre-processed and fed to the model. The model predicts ASD based on the image, displaying "ASD detected" or "non-ASD" on the frame and printing the result.*

### Conclusion and Future Scope

The Modified EfficientNet B0 Model is a customized version of the EfficientNet B0 architecture, optimized with additional classification layers for enhanced performance in specific tasks, such as classifying autism spectrum disorder. This model leverages EfficientNet B0's efficient feature extraction through depthwise separable convolutions and efficient scaling, while fine-tuning classification layers to better suit the target task. The hybrid approach ensures high accuracy with reduced computational cost, making it an effective tool for image classification tasks. Through forward propagation and efficient hierarchical feature extraction, the model predicts the output class for input images, with training parameters optimized to prevent overfitting and maximize performance. Overall, the EfficientNet B0 Hybrid Model achieves a balance between accuracy and efficiency, making it a powerful solution for computer vision applications, including autism detection.

The future of ASD detection using deep learning models presents a compelling path towards more comprehensive diagnosis and improved patient outcomes. One promising avenue lies in multimodal fusion. Deep learning models may evolve beyond analysing isolated data streams, such as images or eye gaze, and instead integrate information from various modalities. This could involve combining brain scans, facial expressions, and even speech patterns.

By leveraging this rich tapestry of data, researchers aim to create a more nuanced understanding of ASD, leading to more accurate and reliable diagnoses. Another key area of exploration is Explainable AI (XAI). While deep learning excels at identifying patterns in data, its decision-making process can often be opaque. XAI techniques will be crucial in demystifying how these models arrive at ASD diagnoses. This

<sup>41</sup> transparency is not only essential for building trust in the technology but also holds the potential to reveal new insights into the underlying mechanisms of ASD itself.

Finally, deep learning could be harnessed to develop tools for continuous monitoring and personalized care.

**References: -**

1. Autism\_Spectrum\_Disorder\_Detection\_in\_Children\_Using\_Transfer\_Learning\_Techniques.  
<https://ieeexplore.ieee.org/abstract/document/10212257>
2. Autistic\_Spectrum\_Disorder\_Screening\_Prediction\_with\_Machine\_Learning\_Models  
<https://ieeexplore.ieee.org/document/9077881>
3. A\_Deep\_Convolutional\_Neural\_Network\_based\_Detection\_System\_for\_Autism\_Spectrum\_Disorder\_in\_Facial\_images  
<https://ieeexplore.ieee.org/document/9641046>
4. Analysis and Detection of Autism Spectrum Disorder Using Machine Learning Techniques  
[https://www.researchgate.net/publication/340711028\\_Analysis\\_and\\_Detection\\_of\\_Autism\\_Spectrum\\_Disorder\\_Using\\_Machine\\_Learning\\_Techniques](https://www.researchgate.net/publication/340711028_Analysis_and_Detection_of_Autism_Spectrum_Disorder_Using_Machine_Learning_Techniques)



## ORIGINALITY REPORT

19%

SIMILARITY INDEX

12%

INTERNET SOURCES

11%

PUBLICATIONS

10%

STUDENT PAPERS

## PRIMARY SOURCES

1

medium.com

Internet Source

1%

2

Singh, Shekhar. "Facial Expression Recognition Using Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) for Data Augmentation and Image Generation", University of Nevada, Las Vegas, 2024

Publication

1%

3

fastercapital.com

Internet Source

1%

4

Nimpa Fondje, Cedric A.. "Bridging Domain Gaps for Cross-Spectrum and Long-range Face Recognition Using Domain Adaptive Machine Learning", The University of Nebraska - Lincoln, 2023

Publication

1%

5

Submitted to University of Surrey

Student Paper

1%

6

www.internationaljournalssrg.org

Internet Source

<1%

7	Submitted to Heriot-Watt University Student Paper	<1 %
8	Submitted to Liverpool John Moores University Student Paper	<1 %
9	Submitted to University of Derby Student Paper	<1 %
10	Submitted to B.V. B College of Engineering and Technology, Hubli Student Paper	<1 %
11	iabac.org Internet Source	<1 %
12	www.mdpi.com Internet Source	<1 %
13	"Image and Video Technology", Springer Science and Business Media LLC, 2024 Publication	<1 %
14	Submitted to University of Portsmouth Student Paper	<1 %
15	Submitted to Yakin Doğu Üniversitesi Student Paper	<1 %
16	Submitted to University of North Texas Student Paper	<1 %
17	Submitted to University of Hertfordshire Student Paper	<1 %

18

Submitted to City University of Seattle

Student Paper

&lt;1 %

19

Submitted to I.I.S. University, Jaipur

Student Paper

&lt;1 %

20

Jui-Sheng Chou, Hoang-Minh Nguyen.  
"Simulating long-term energy consumption  
prediction in campus buildings through  
enhanced data augmentation and  
metaheuristic-optimized artificial intelligence",  
Energy and Buildings, 2024

Publication

&lt;1 %

21

Umberto Michelucci. "Fundamental  
Mathematical Concepts for Machine Learning  
in Science", Springer Science and Business  
Media LLC, 2024

Publication

&lt;1 %

22

Submitted to University of Information  
Technology, Yangon

Student Paper

&lt;1 %

23

dspace.bracu.ac.bd

Internet Source

&lt;1 %

24

Aachal Modak, Apurva Padamwar, Shivani  
Pokale, Roshani Raut, Anita Devkar, Sonali  
Patil. "Disease Detection in Rice leaves using  
Convolutional Neural Network", 2023  
International Conference on Applied

&lt;1 %

# Intelligence and Sustainable Computing (ICAISC), 2023

Publication

- 
- |   |   |                |
|---|---|----------------|
| <div style="background-color: red; color: white; padding: 2px 5px; display: inline-block;">25</div> | <p>Adi Dwifana Saputra, Djarot Hindarto, Ben Rahman, Handri Santoso. "Comparison of Accuracy in Detecting Tomato Leaf Disease with GoogleNet VS EfficientNetB3", SinkrOn, 2023</p> <p>Publication</p> | <p>&lt;1 %</p> |
|---|---|----------------|
- 
- |   |  |                |
|---|--|----------------|
| <div style="background-color: magenta; color: white; padding: 2px 5px; display: inline-block;">26</div> | <p>Submitted to University of Cape Town</p> <p>Student Paper</p> | <p>&lt;1 %</p> |
|---|--|----------------|
- 
- |  |  |                |
|--|--|----------------|
| <div style="background-color: purple; color: white; padding: 2px 5px; display: inline-block;">27</div> | <p>V. R. Azhaguramyaa, K. Srinivasan, S. Oswalt Manoj, M. Rohini, R. Rama Abirami. "chapter 8 Decoding Parkinson's Disease", IGI Global, 2024</p> <p>Publication</p> | <p>&lt;1 %</p> |
|--|--|----------------|
- 
- |  |  |                |
|--|--|----------------|
| <div style="background-color: teal; color: white; padding: 2px 5px; display: inline-block;">28</div> | <p>Submitted to West Visayas State University</p> <p>Student Paper</p> | <p>&lt;1 %</p> |
|--|--|----------------|
- 
- |   |   |                |
|---|---|----------------|
| <div style="background-color: green; color: white; padding: 2px 5px; display: inline-block;">29</div> | <p><a href="https://joiv.org">joiv.org</a></p> <p>Internet Source</p> | <p>&lt;1 %</p> |
|---|---|----------------|
- 
- |   |   |                |
|---|---|----------------|
| <div style="background-color: brown; color: white; padding: 2px 5px; display: inline-block;">30</div> | <p>Submitted to University of Technology, Sydney</p> <p>Student Paper</p> | <p>&lt;1 %</p> |
|---|---|----------------|
- 
- |   |   |                |
|---|---|----------------|
| <div style="background-color: brown; color: white; padding: 2px 5px; display: inline-block;">31</div> | <p><a href="https://publications.eai.eu">publications.eai.eu</a></p> <p>Internet Source</p> | <p>&lt;1 %</p> |
|---|---|----------------|
-

32	"Neural Information Processing", Springer Science and Business Media LLC, 2017 Publication	<1 %
33	Submitted to University of Pretoria Student Paper	<1 %
34	Submitted to Nanyang Technological University Student Paper	<1 %
35	Submitted to Coventry University Student Paper	<1 %
36	Submitted to The University of Manchester Student Paper	<1 %
37	<a href="http://ijsrp.org">ijsrp.org</a> Internet Source	<1 %
38	<a href="http://www.arxiv-vanity.com">www.arxiv-vanity.com</a> Internet Source	<1 %
39	<a href="http://www.cscml.org">www.cscml.org</a> Internet Source	<1 %
40	<a href="http://www.researchsquare.com">www.researchsquare.com</a> Internet Source	<1 %
41	Wong, Shi Wen. "A Study of the Local Deep Galerkin Method for the Modified Cahn Hilliard Equation", South Dakota State University, 2023 Publication	<1 %

42	<a href="https://cdn.techscience.cn">cdn.techscience.cn</a> Internet Source	<1 %
43	<a href="https://www.ijert.org">www.ijert.org</a> Internet Source	<1 %
44	<a href="https://www.ijraset.com">www.ijraset.com</a> Internet Source	<1 %
45	Submitted to Clemson University Student Paper	<1 %
46	Derrick Roy Edgar Rajappan, S Sanith, S Subbulakshmi. "Malnutrition Detection using Deep Learning Models", 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), 2023 Publication	<1 %
47	Submitted to Mepco Schlenk Engineering college Student Paper	<1 %
48	<a href="https://aisberg.unibg.it">aisberg.unibg.it</a> Internet Source	<1 %
49	<a href="https://essay.utwente.nl">essay.utwente.nl</a> Internet Source	<1 %
50	Submitted to Buckinghamshire Chilterns University College Student Paper	<1 %
51	Submitted to National College of Ireland Student Paper	

<1 %

52

Tanvi Shetty, Vidya Zope, Maitraiya Dandekar, Anmol Devnani, Puneet Meghrajani. "sha- Early Intervention for children at risk of ASD", 2022 International Conference on Industry 4.0 Technology (I4Tech), 2022

Publication

<1 %

53

[journals.iium.edu.my](https://journals.iium.edu.my)

Internet Source

<1 %

54

[sungood.shop](https://sungood.shop)

Internet Source

<1 %

55

Submitted to Babes-Bolyai University

Student Paper

<1 %

56

Submitted to California State University, San Bernadino

Student Paper

<1 %

57

Submitted to Southampton Solent University

Student Paper

<1 %

58

[link.springer.com](https://link.springer.com)

Internet Source

<1 %

59

[www.c-sharpcorner.com](https://www.c-sharpcorner.com)

Internet Source

<1 %

60

Submitted to Jagran Lakecity University

Student Paper

<1 %

61	Min Feng, Juncai Xu. "Electroencephalogram-Based ConvMixer Architecture for Recognizing Attention Deficit Hyperactivity Disorder in Children", Brain Sciences, 2024 Publication	<1 %
62	Submitted to University Politehnica of Bucharest Student Paper	<1 %
63	Submitted to University of Hong Kong Student Paper	<1 %
64	123dok.org Internet Source	<1 %
65	Submitted to Kingston University Student Paper	<1 %
66	Sridhar Chandramohan Iyer, Narendra M. Shekokar. "Identifying the Optimal Deep Learning Based Image Recognition Technique for Dog Breed Detection", 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023 Publication	<1 %
67	Submitted to The University of the West of Scotland Student Paper	<1 %
68	Submitted to University of Leeds Student Paper	<1 %



69	<a href="http://www.researchgate.net">www.researchgate.net</a> Internet Source	<1 %
70	Submitted to University of Bristol Student Paper	<1 %
71	<a href="http://journals.grdpublications.com">journals.grdpublications.com</a> Internet Source	<1 %
72	<a href="http://www.ijisae.org">www.ijisae.org</a> Internet Source	<1 %
73	"Advanced Intelligent Computing Technology and Applications", Springer Science and Business Media LLC, 2023 Publication	<1 %
74	"Advances in Neural Networks - ISNN 2017", Springer Science and Business Media LLC, 2017 Publication	<1 %
75	Shuting Zhao, Lifeng Wu, Youzhen Xiang, Fucang Zhang. "Forecasting short-term methane based on corrected numerical weather prediction outputs", Journal of Cleaner Production, 2024 Publication	<1 %
76	<a href="http://aitechtrend.com">aitechtrend.com</a> Internet Source	<1 %
77	<a href="http://dlibrary.univ-boumerdes.dz:8080">dlibrary.univ-boumerdes.dz:8080</a> Internet Source	<1 %

78

[elibrary.tucl.edu.np](http://elibrary.tucl.edu.np)

Internet Source

<1 %

79

[ijhs.qu.edu.sa](http://ijhs.qu.edu.sa)

Internet Source

<1 %

80

[web.archive.org](http://web.archive.org)

Internet Source

<1 %

81

[www.hindawi.com](http://www.hindawi.com)

Internet Source

<1 %

82

[www.techscience.com](http://www.techscience.com)

Internet Source

<1 %

83

"Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications", Springer Science and Business Media LLC, 2019

Publication

<1 %

84

Allam, Mary Akshara. "Emotion Detection Using Deep Learning", California State University, Sacramento, 2021

Publication

<1 %

85

EzzatiKarami, Mahtab. "Automatically Classifying Non-Functional Requirements with Feature Extraction and Supervised Machine Learning Techniques", The University of Western Ontario (Canada), 2023

Publication

<1 %

86	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1 %
87	Lotfi Mhamdi, Oussama Dammak, François Cottin, Imed Ben Dhaou. " Deep learning for - 19 contamination analysis and prediction using images on ", International Journal of Imaging Systems and Technology, 2023 Publication	<1 %
88	Miroslav Valan, Karoly Makonyi, Atsuto Maki, Dominik Vondráček, Fredrik Ronquist. "Automated Taxonomic Identification of Insects with Expert-Level Accuracy Using Effective Feature Transfer from Convolutional Networks", Systematic Biology, 2019 Publication	<1 %
89	Nadeem Akhtar, U. Ragavendran. "Interpretation of intelligence in CNN-pooling processes: a methodological survey", Neural Computing and Applications, 2019 Publication	<1 %
90	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
91	arxiv.org Internet Source	<1 %
92	cheatography.com	

93

[commons.und.edu](https://commons.und.edu)

Internet Source

&lt;1 %

94

[ebin.pub](https://ebin.pub)

Internet Source

&lt;1 %

95

[ieeexplore.ieee.org](https://ieeexplore.ieee.org)

Internet Source

&lt;1 %

96

[www.geeksforgeeks.org](https://www.geeksforgeeks.org)

Internet Source

&lt;1 %

97

Endang Sugiharti, Riza Arifudin, Dian Tri Wiyanti, Arief Broto Susilo. "Integration of convolutional neural network and extreme gradient boosting for breast cancer detection", Bulletin of Electrical Engineering and Informatics, 2022

Publication

&lt;1 %

98

Huang, Zhewen. "Engagement Prediction in YouTube Educational Videos", Miami University, 2023

Publication

&lt;1 %

99

Suleyman Yildirim, Zeeshan Rana, Gilbert Tang. "Autonomous Ground Refuelling Approach for Civil Aircrafts using Computer Vision and Robotics", 2021 IEEE/AIAA 40th

&lt;1 %

# Digital Avionics Systems Conference (DASC), 2021

Publication

---

---

Exclude quotes      On

Exclude bibliography      On

Exclude matches      Off

# RE-2022-287075-plag-report

## GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

---

PAGE 23

---

PAGE 24

---

PAGE 25

---

PAGE 26

---

PAGE 27

---

PAGE 28

---

PAGE 29

---

PAGE 30

---

PAGE 31

---

PAGE 32

---

PAGE 33

---

PAGE 34

---

PAGE 35

---

PAGE 36

---