Autism Detecting Model using Image

[1]Md Tahmid Zoayed, [2]Sayma Haque Arshe, and [3]Plabon Banik *

[1]Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
[2] Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
[3] Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
2019-1-60-154@std.ewubd.edu, 2019-1-60-021@std.ewubd.edu, 2019-1-60-167@std.ewubd.edu
*Corresponding Author

Abstract. This paper brings a model which can be used in detecting autism-affected patients using their images. The images of autism-affected patients and non-autism patients are collected from the Kaggle website. The total number of images is almost 2940. We have implemented CNN(Convolution Neural Network) algorithm to build this model. We have used TensorFlow and Keras library in this process and lots of image processing functions to smooth the model. This model can be implemented in hospitals and clinics as a GUI or software to detect autism disease without any kind of medical operations.

Keywords: Autism detection, Autism Detection using an image, Autism Detection using CNN, Autism detection using CNN, TensorFlow, and Keras, Autism detection using deep learning, disease detection using CNN.

1 Introduction

Autism is a neuro-disorder in which a person's interactions and communication with others are affected for the rest of their lives. Autism can be found at any age and is referred to as a "behavioral disease" since symptoms commonly develop in the first two years of life. ASD detection is challenging since there are various mental conditions with few symptoms that are quite similar to those with ASD symptoms, making this a difficult assignment.

Early identification of this neurological condition can help keep the subject's mental and physical health in good shape. With the increased use of machine learning-based models to forecast numerous human diseases, early diagnosis based on multiple health and physiological parameters appears to be viable. Machine learning

is now being used to diagnose disorders such as depression, and ASD. The key goals of using machine learning techniques are to enhance diagnostic accuracy and minimize diagnosis time to enable faster access to health care services. Now, we wish to detect this disease using an image of a person and check how much it can contribute to the medical industry.

Recently, many efforts are done to identify ASD based on deep learning with fMRI (Koyamada et al., 2015; Anirudh and Thiagarajan, 2017; Subbaraju et al., 2017). In Koyamada et al. (2015), a deep neural network (DNN) model was implemented to build a subject-transfer decoder. The authors used principal sensitivity analysis (PSA) to construct a decoder for visualizing different features of all individuals in the dataset.

To implement the model we will use CNN(Convolution Neural Network) which is an algorithm of deep learning. Speech recognition, image classification, automotive software engineering, and neuroscience are the areas where CNN has given very great performances. This huge progress is mainly due to a combination of algorithmic improvements, computation resource enhancements, and access to a big amount of data. That's why we have chosen CNN for this work. We'll try to maximize the accuracy by creating more hidden layers and using a strong dataset.

2 Related Work

Previously, a lot of persons have made such models to identify autism by pictures. They all had taken different approaches.
youngsters with ASD, while the stereotypical motor movement of the patients stood out enough to be noticed." The SMMs are a
vital class of the abnormal and rehashed practices of kids with Autism, so it is important to create powerful and exact techniques
for the programmed location of these developments [14]. Another examination recommends the order of the rehashed examples of
strolling, which depends on the motor and kinematic attributes of strolling with the assistance of AI. Linear analysis concerning Learning classification with its positives and negatives and tried to spot the problem going with "existing ASD screening tools and the consistency of such tools using the DSM-IV instead of the DSM-5 manual". Thomas et al. notice the indicative control of Autism by utilizing the attractive tomography of the cerebrum [12]. Another Artificial neural network was made by utilizing the information base of the application ASD Tests just as a data set. As information, they have utilized ten inquiries, including the age and the sex of the member.

3 System Architecture and Design

The proposed framework is depicted in Figure. 1. The framework contains the following major steps: (1) Training the images with 21 epochs (2) validating the train set with the valid set (3) Using the test set to test the result

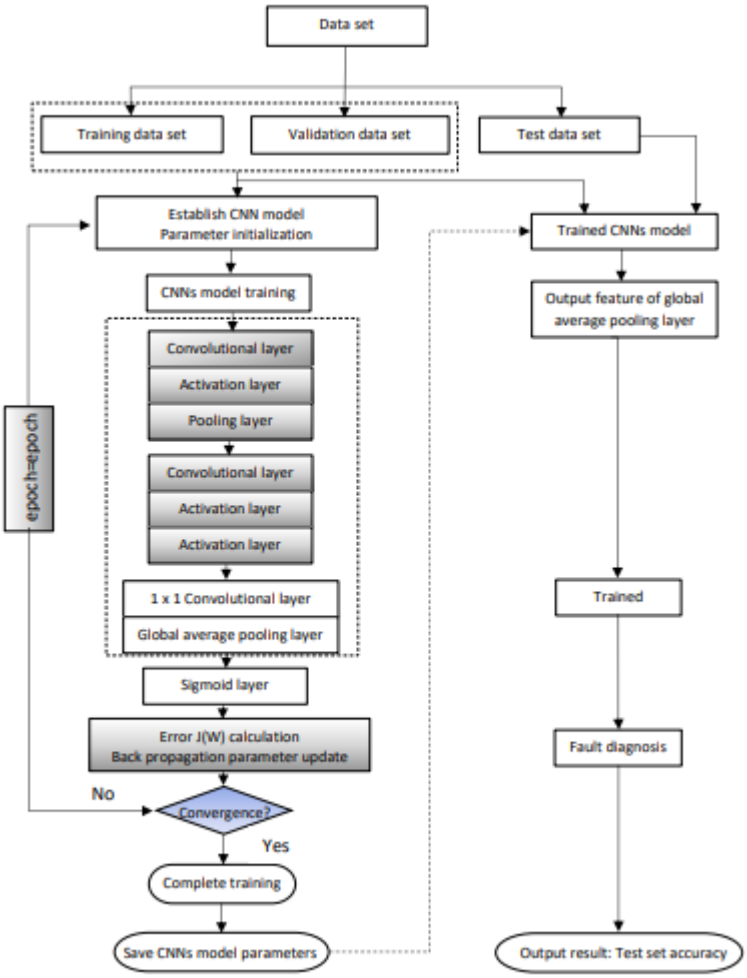

Fig. 1. The system architecture of the model

3.1 Dataset Description

The dataset used in this model is taken from the Kaggle website. This dataset consists of 3 folders named train, test, and valid. All of the folders are filled with autism and non-autism patient's image. In this dataset, the train folder has 1270 autism-affected images and 1270 non-autism images. In the valid folder, the autism and non-autism both have 50 images in each sub-folder. Finally, there are 301 files of images in the test folders. All the images are in jpg format.

Fig. 2. Sample input data set

The images of these humans are collected without any racism from each ethnicity across the whole world. The total collection of all images is almost 2940. We are going to work on this dataset and implement our model with this dataset using different folders.

3.2 Data Preprocessing

The images of the dataset folders are of different sizes and got some noise in them. To prevent the noises and make the dataset smoother we have implemented some preprocessing techniques on the images. We have preprocessed the train and valid folder using the ImageDataGenerator function. In the function, we have rescaled the image by (1/255) as the pixel values are[0,255]. We have also used 20% shear and 20% zoom-in to get clear visibility of the images. The shear will visualize the images from every human angle and detect the image better. We made the horizontal true as we need all images trained in horizontal mode. At last, we have

set all the images in (200x200) shape for the smooth work of the algorithm.

3.3 Convolutional Neural Network Module

Convolutional Neural Networks are one of the most widely used deep neural networks. It was most typically used to examine visual imagery. The human brain inspired a feed-forward neural network. Simpler patterns (lines, curves, etc.) are detected initially, then many more complex patterns (faces, objects, etc.) are detected. Typically, a CNN module includes three layers:

- Convolutional layer
- Pooling layer
- Fully connected layer

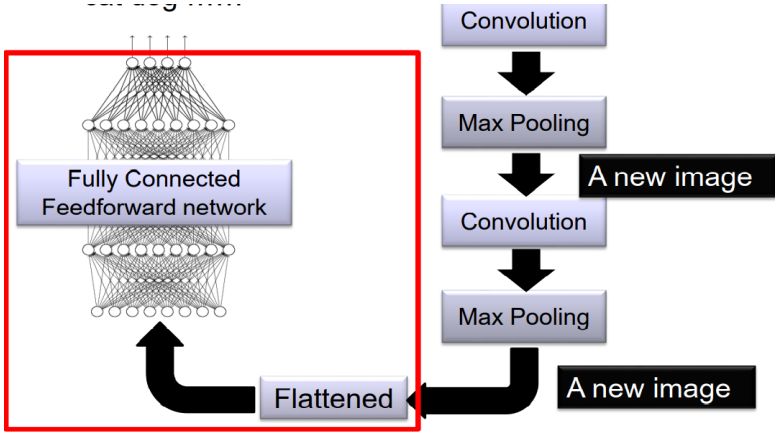

Fig. 3. CNN methods

3.3.1 Convolutional layer:

CNN implements a technique known as Convolution. A convolutional layer is made up of several filters that perform convolutional operations. Each layer creates many activation functions that are passed on to the next layer when we input an image into a ConvNet.

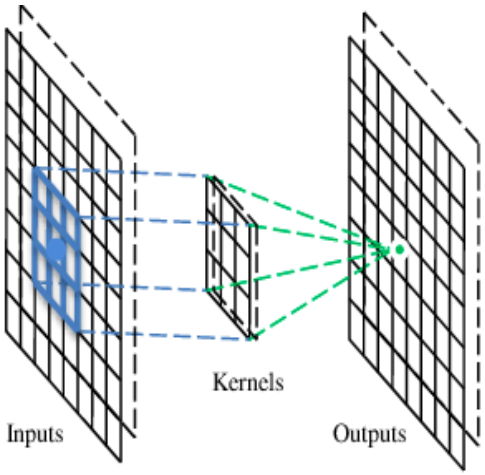

Fig. 4. CNN kernels

Algorithm: processing of CONV layers.

Input: An image
1. Begin
2. for j = 1:J {% Loop on output feature maps
3. for i = 1:I {% Loop on Input feature maps
4. for m = 1:M {% Loop on rows of output feature maps
5. for n = 1:N {% Loop on columns of output feature maps
6. for p = 1:P {% Loop on rows of filer kernel
7. for q = 1:Q {% Loop on columns of filter kernel
8. $OF\ j\ m, n = OF\ j\ m, n + F\ i,j\ p, q \times IF\ i(m + p - 1, n + q - 1)$; }}}}}}} % Loops
9. End

I and J denote the number of input and output feature maps, M and N the number of rows and columns in output feature maps, and P and Q the number of rows and columns in filter kernels, respectively. These values can be used to define additional values, such as the number of filters, which can be derived from J.

3.3.2 Pooling:

After a convolutional layer, pooling is commonly used. A pooling operation that determines the maximum value for patches of a feature map and uses it to build a down sampled feature map is known as Max Pooling. In CNN, two types of pooling are commonly utilized. The first is local pooling, which involves down sampling feature maps by pooling data from small local locations. The second is global pooling, which extracts a scalar value of a feature vector for picture representation from each of the feature maps. The fully connected layers use this representation to classify the data.

Algorithm: procedure of pooling layers.

1. Start
2. Input:
3. Input features $x_1 \ldots x_N, k_{top}$, Total Layer L
4. Output:
5. Update k_1
6. Extract Features $y_i \leftarrow$f(x;b)
7. Convolve network $y_j = \sum_{i=0}^{k} (k_{ij} * x_i)$
8. Compute activation layer y_i(l+1) = f $((k_{ij})\ y_i) + b_i$
9. Calculate k_{map}
10. Calculate $k_{map} = k_{map}$ *Gaussian R
11. Choose to pool k_1= max (k_{top}, k_{map}
12. End

3.3.3 Fully connected layer:

As in a typical FCNN, this layer has a full connection with all preceding and succeeding layers. The FC layer helps in the mapping of representations between input and output. We must flatten the final pooling k_1 and create an output size in this layer. Finally, we have our result. A matrix multiplication followed by a bias effect can be used to compute it.

3.4 Xception :

Francois Chollet proposes the Xception Model. Xception is an expansion of the Inception Architecture that uses depthwise Separable Convolutions to replace the

regular Inception modules. For the picture identification and classification job, the Xception model was trained using the ImageNet dataset [2, 3]. Xception is a deep CNN with a lot of features. new levels of inception The inception layers are made from of Layers of depthwise convolution are followed by a layer of pointwise convolution. There are two types of transfer learning: extraction and fine-tuning of features The characteristic in this study is The pretraining model was employed in the extraction process. taught to extract the feature from a standard dataset the new dataset, and to remove the model's top layers -e custom top layers were added to the model. The number of classes determines the categorization. Fine-tuning has been used to customize generic characteristics for a specific class.

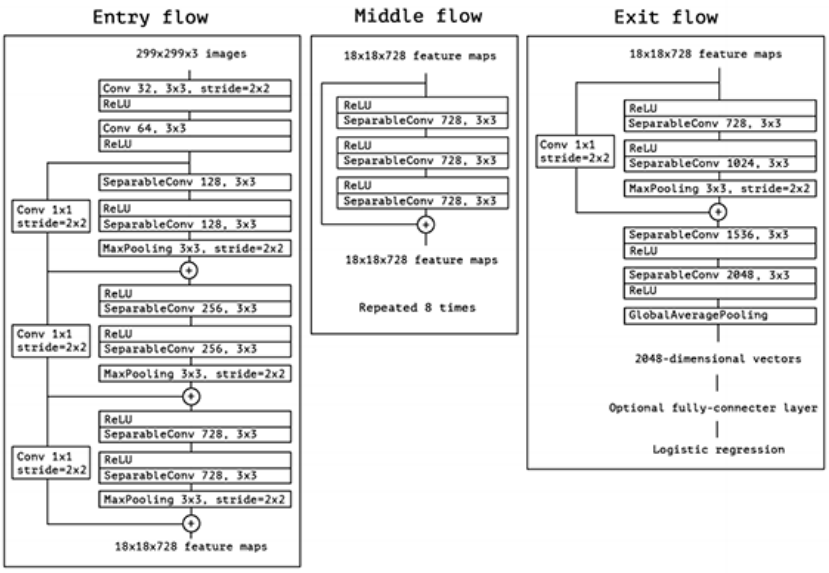

Fig. 5. Xception Architecture

3.5 VGG16 :

VGG16 is one of the most excellent vision models used convolution neural net (CNN) architecture. VGG16 focused on 3x3 filter convolution layers having stride 1 and always, and maxpool layer of 2x2 filter stride 2. Throughout the architecture, it maintains this convolution and max pool layer layout. After that, there are two completely connected layers and a softmax for output. The VGG16 stands for "weighted 16 layers."

3.6 VGG19 :

The VGG19 net was built primarily to win the ILSVRC, but it has been utilized in a variety of other ways as well. It is used simply as a suitable classification architecture for many additional datasets, and because the authors made the models public, they may be used as is or with modifications for other comparable jobs as well. Transfer learning may also be applied to facial recognition applications. Weights are easily available in other frameworks like as Keras and may be tinkered with and utilized as desired. The VGG19 is mainly upgrade version of VGG16 which have 19 layers while VGG16 has 16 layers.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 6. VGG16 & VGG19 Architecture

4 Implementation and Experimental Result

4.1 Experimental Setup

The proposed system has been implemented on a machine having Windows 10, Core i3 2.4 GHz with 16GB RAM. The work is done in Jupyter Notebook using Keras and TensorFlow packages. Python 3.6.7(version) is used for developing it.

4.2 Implementation

Images of the datasets are transformed and scaled by our preprocessing techniques. Now it's time to implement CNN on them and get a result. We have used 2 convolutions and 2 max pool layers in the model. Then, after flattening, we have used 3 dense layers where 2 are hidden and one is output. In the output, we have used the 'sigmoid activation function. Rather than that, we have used the 'ReLU' activation function in all other layers. In the compilation, we have used 'binary_entropy' as 'loss entropy' and set the Learning rate at 0.00001. As we have binary class clasification in the dataset the loss function of 'binary_entropy' is.
Loss= abs(Y_pred – Y_actual).

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_87 (Conv2D)          (None, 198, 198, 128)     3584

 max_pooling2d_36 (MaxPoolin  (None, 99, 99, 128)      0
 g2D)

 conv2d_88 (Conv2D)          (None, 97, 97, 256)       295168

 max_pooling2d_37 (MaxPoolin  (None, 48, 48, 256)      0
 g2D)

 flatten_12 (Flatten)        (None, 589824)            0

 dense_64 (Dense)            (None, 512)               301990400

 dense_65 (Dense)            (None, 100)               51300

 dense_66 (Dense)            (None, 1)                 101

=================================================================
Total params: 302,340,553
Trainable params: 302,340,553
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. Summary of the used model

After setting all the parameters, we moved to our fitting the model segment. We have used 21 epochs in this. The valid folder is used as validation in the model. We have also restored the best weights to gain the maximum result possible and assigned the patience as 3. After that we have used every model with 5 epochs to get an idea of which algorithm works best.

4.3 Performance Evaluation

We can see the best accuracy possible from 21 epochs is 77% in this model. We have run 21 epochs and got this result. But with 5 epochs with other algorithms like Xception, VGG19,VGG16 and our proposed model we get 66%,71%, 73% and 73% resepctively. This gives a visual represetation of which model works best.

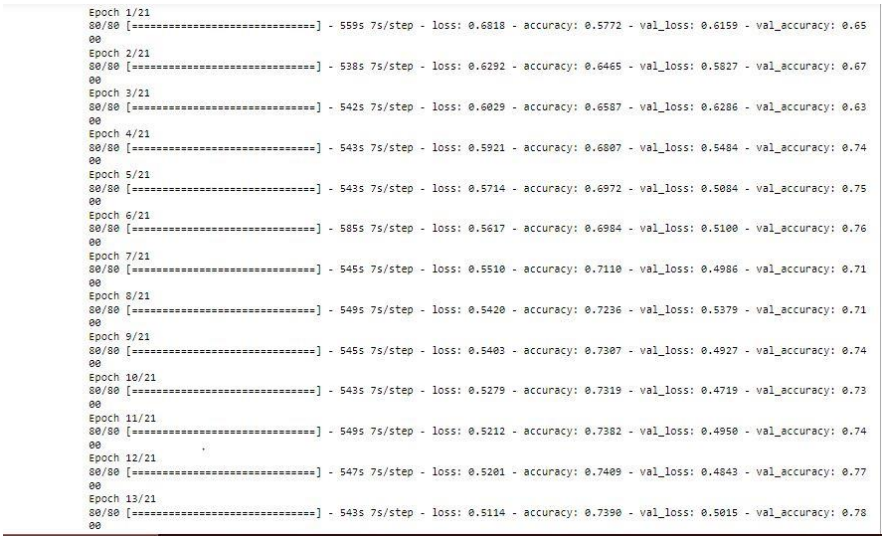

Fig. 8. Epochs running in Proposed model

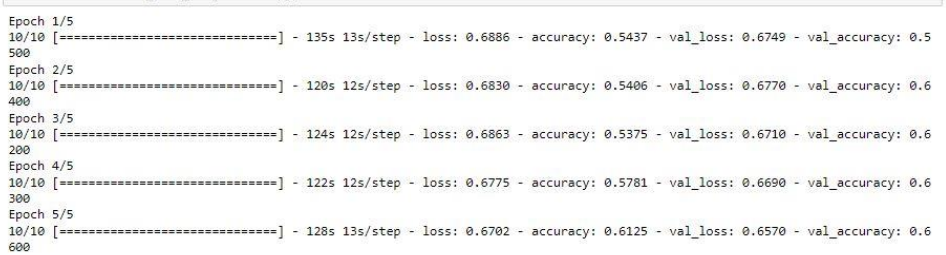

Fig. 9. Epochs running in Xception model

```
Epoch 1/5
10/10 [==============================] - 238s 24s/step - loss: 0.8126 - accuracy: 0.5031 - val_loss: 0.6578 - val_accuracy: 0.5
900
Epoch 2/5
10/10 [==============================] - 238s 24s/step - loss: 0.6959 - accuracy: 0.5750 - val_loss: 0.6446 - val_accuracy: 0.6
600
Epoch 3/5
10/10 [==============================] - 240s 24s/step - loss: 0.6458 - accuracy: 0.6250 - val_loss: 0.6045 - val_accuracy: 0.6
600
Epoch 4/5
10/10 [==============================] - 214s 22s/step - loss: 0.6229 - accuracy: 0.6667 - val_loss: 0.5962 - val_accuracy: 0.6
600
Epoch 5/5
10/10 [==============================] - 229s 23s/step - loss: 0.6235 - accuracy: 0.6625 - val_loss: 0.5828 - val_accuracy: 0.7
100
```

Fig. 10. Epochs running in VGG16 model

```
Epoch 1/5
10/10 [==============================] - 278s 27s/step - loss: 0.7264 - accuracy: 0.5813 - val_loss: 0.6531 - val_accuracy: 0.5
700
Epoch 2/5
10/10 [==============================] - 273s 28s/step - loss: 0.7206 - accuracy: 0.5562 - val_loss: 0.6117 - val_accuracy: 0.6
900
Epoch 3/5
10/10 [==============================] - 252s 25s/step - loss: 0.6070 - accuracy: 0.6733 - val_loss: 0.5750 - val_accuracy: 0.7
300
Epoch 4/5
10/10 [==============================] - 291s 30s/step - loss: 0.6386 - accuracy: 0.6531 - val_loss: 0.6320 - val_accuracy: 0.6
500
Epoch 5/5
10/10 [==============================] - 293s 30s/step - loss: 0.6262 - accuracy: 0.6625 - val_loss: 0.6229 - val_accuracy: 0.6
500
```

Fig. 11. Epochs running in VGG19 model

We have implemented a plot to visualize the accuracy more effectively. This plot will help you to understand better the model and its accuracy.

Fig. 12. Training vs Validation (Loss & Accuracy) of Proposed Model

Fig. 13. Training vs Validation (Loss & Accuracy) of Xception Model

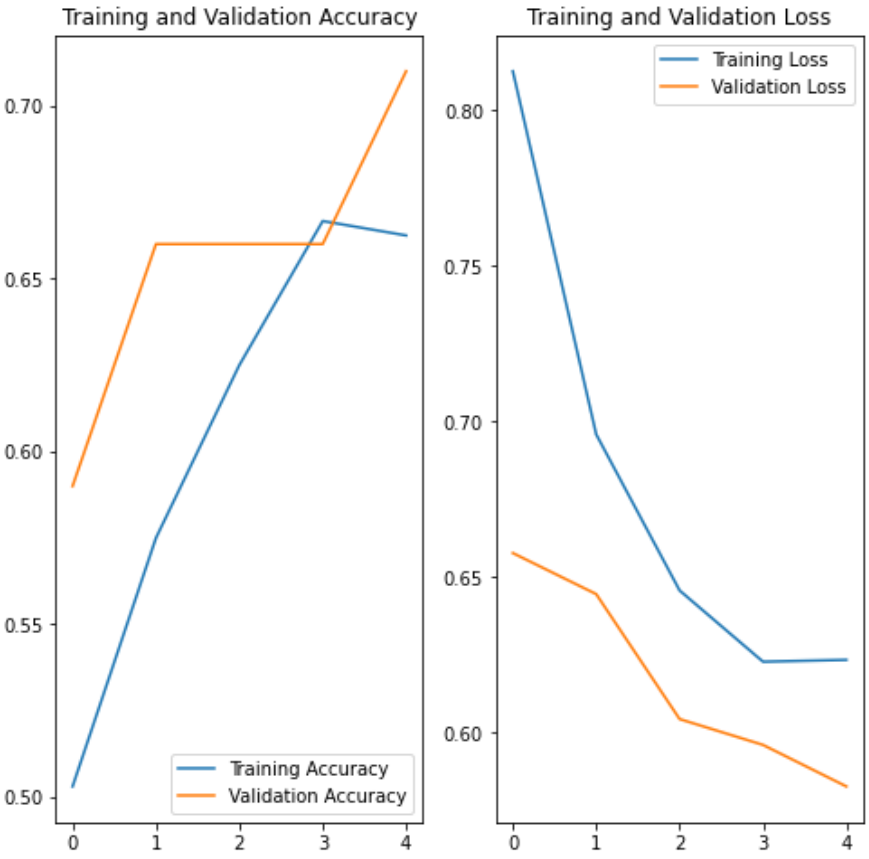

Fig. 14. Training vs Validation (Loss & Accuracy) of VGG16 Model

Fig. 15. Training vs Validation (Loss & Accuracy) of VGG19 Model

Now, we'll use this model to check our test images from the test folder. If the value we get is '0' the person is affected with Autism else the person is non-autistic. This is how we can check the accuracy of our model better.

Fig. 16. Non-Autism child

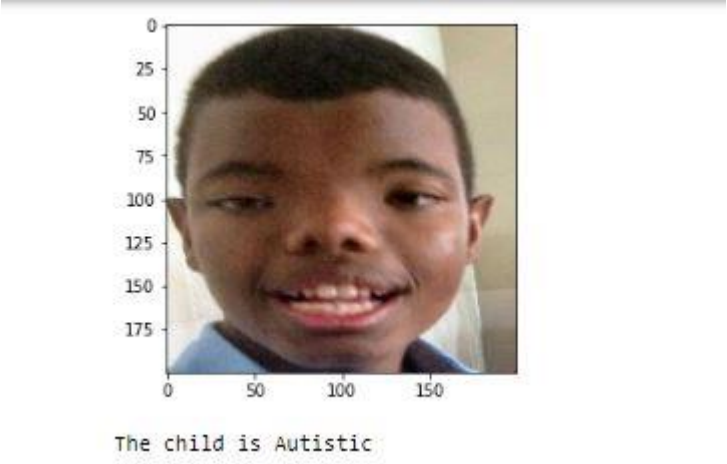

Fig. 17. Autism child

4.4 Comparison with other existing framework

The performance of each model is diffrent from others. Based on their performances some works very good as a model. From the above discussion we

have figured out that VGG19 and our proposed model works the best in this dataset, while others does an average job. If we increase the size of epochs from 5 to 21, it will give more accuracy. The list of their performance accuracy is shown in the table with 5 epochs.

Name of Models	**Image Detection Accuracy**
Xception	66%
VGG16	71%
VGG19	73%
Proposed CNN	73%

Table.1. Accuracy of Diferent models

5 Conclusion

Autistic Spectrum Disorder (ASD) is a neurodevelopmental disorder that comes with high medical costs. Autism cannot be detected with medical tests. As a result, a behavioral prediction is required. Machine learning can assist overcome this problem. In this paper, we use CNN and VGG19 the most effective machine learning algorithm to detect autism patients. Through this algorithm, we got the optimal result of prediction. For the ASD dataset, CNN-based model and VGG19 had the highest prediction accuracy. This experiment involved the use of 2940 children's images image collection is classified into autistic and non-autistic children using.CNN. This ASD diagnosis has the potential to dramatically reduce our society's nervous system problems and can diagnose a new child immediately.

References

1. C. S. Paula, S. H. Ribeiro, E. Fombonne, and
M. T. Mercadante, “Brief report: prevalence of the pervasive
developmental disorder in Brazil: a pilot study,” Journal of
Autism and Developmental Disorders, vol. 41, no. 12,
pp. 1738–1742, 2011.
2. L. C. Nunes, P. R. Pinheiro, M. C. D. Pinheiro et al., A Hybrid
Model to Guide the Consultation of Children with Autism
Spectrum Disorder, A. Visvizi and M. D. Lytras, Eds., Springer

International Publishing, View at: Google Scholar,
pp. 419–431.
3. Apa–American Psychiatric Association, “Diagnostic and
statistical manual of mental disorders (DSM –5),” 2020,
https://www.psychiatry.org/psychiatrists/practice/dsm.
4. R. Carette, F. Cilia, G. Dequen, J. Bosche, J.-L. Guerin, and
L. Vandromme, “Automatic autism spectrum disorder detection thanks to eye-tracking and neural network-based
approach,” in Proceedings of the International Conference on
IoT Technologies for Healthcare, pp. 75–81. 4, Springer, Angers, France, 24–25 October 2017.
5. L. Kanner, “Autistic disturbances of affective contact,” Nerv.
Child, vol. 2, pp. 217–250, 1943.
6. E. Fombonne, “Epidemiology of pervasive developmental
disorders,” Pediatric Research, vol. 65, no. 6, pp. 591–598,
2009.
7. “International statistical classification of diseases and related
health problems (ICD),” https://www.who.int/standards/
classifications/classification-of-diseases.
8. G. Yolcu, I. Oztel, S. Kazan et al., “Deep learning-based facial
expression recognition for monitoring neurological disorders,” in Proceedings of the 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM),
pp. 1652–1657, Kansas City, MO, USA, 13–16 November
2017.
9. G. Piosenka, “Detect autism from a facial image,” 10 December 2021, https://www.kaggle.com/cihan063/autismimage-data accessed on.
10. https://towardsdatascience.com/detecting-autism-spectrumdisorder-in-children-with-computer-vision-8abd7fc9b40arep
lace.
11. C. Garcia, J. Ostermann, and T. Cootes, “Facial image processing,” Eurasip J. Image Video Process, vol. 2007, pp. 1-2, 2008.
12. F. C. Tamilarasi and J. Shanmugam, “Convolutional Neural
Network based Autism Classification,” in 2020 5th International Conference on Communication and Electronics Systems
(ICCES), pp. 1208–1212, IEEE, (2020, June).
13. S. Jahanara and S. Padmanabhan, “Detecting autism from
facial image,” 2021
14. Marr, B. 7 Amazing Examples of Computer and Machine Vision in Practice. Forbs. 8 April 2019. Available online:
https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-
practice/?sh=4eca093d1018 (accessed on 16 August 2021).
15. de Belen, R.A.J.; Bednarz, T.; Sowmya, A.; Del Favero, D. Computer vision in autism spectrum disorder research: A systematic
review of published studies from 2009 to 2019. Transl. Psychiatry 2020, 10, 1–20. https://doi.org/10.1038/s41398-020-01015-w.
16. Rahman, O.L.U.M. A Review of Machine Learning Methods of Feature Selection and Classification for Autism Spectrum
Disorder. Brain Sci. 2020, 10, 949. https://doi.org/10.3390/brainsci10120949.
17. Piosenka, G. Detect Autism from a Facial Image. Available online: https://cutt.ly/ibIXt5a (accessed on 27 January 2021).

18. Rajaram, M. Concerns with ‘Detect Autism’ Dataset. Kaggle. Available online: www.kaggle.com/melissarajaram/concerns-
with-detect-autism-dataset (accessed on 6 August 2021).
19. Musser, M. Detecting Autism Spectrum Disorder in Children with Computer Vision. Medium. 24 August 2020. Available online:
https://towardsdatascience.com/detecting-autism-spectrum-disorder-in-children-with-computer-vision-8abd7fc9b40a
(accessed on 1 August 2021).
20. Vo, T.; Nguyen, T.; Le, T. Race Recognition Using Deep Convolutional Neural Networks. Symmetry 2018, 10, 564.
https://doi.org/10.3390/sym10110564.
21. Chaudhuri, A. Deep Learning Models for Face Recognition: A Comparative Analysis. In Deep Biometrics; Springer: Singapore,
2020; pp. 99–140.
22. Gwyn, T.; Roy, K.; Atay, M. Face Recognition Using Popular Deep Net Architectures: A Brief Comparative Study. Future Internet
2021, 13, 164. https://doi.org/10.3390/fi13070164.