Detecting Autism Spectrum Disorder based on Eye-tracking

Hawra A. Al Mohsin and Prof. Mohab A. Mangoud
Department of Electrical Engineering, University of Bahrain, Kingdom of Bahrain
Corresponding author : Hawra A. Al Mohsin (e-mail: 20160966@ stu.uob.edu.bh).

ABSTRACT

Autism spectrum disorder (ASD) is a developmental disorder associated with cognitive and neurobehavioral disorders. It affects the person's behavior and performance. Autism affects verbal and non-verbal communication in social interactions. Early screening and diagnosis of ASD are essential and helpful for early educational planning and treatment, the provision of family support, and for providing appropriate medical support for the child on time. Thus, developing automated methods for diagnosing ASD is becoming an essential need. In this research, we propose a practical ASD screening solution using Eye-tracking through applying several models in deep learning : CNN, ANN, ConvNetV2 and machine learning model : non neural network (Naive Bayes) . the best accuracy we reached with Our models is a 78% classification accuracy with ConvNetV2 model. we are using Eye-tracking snapshot dataset to detect ASD based on gaze point , which is an open source dataset . Our results support the clinical findings of Eye movement differences between children with ASD and NON-ASD.

KEYWORDS—***ASD , Gaze point , CNN , ANN , Naive Bayes , POG***

Introduction

Several research studies have explored the use of AI in autism diagnosis. For example, machine learning algorithms have been trained on various data sources, such as behavioral assessments, medical records, and neuroimaging data, to identify patterns that differentiate individuals with autism from neurotypical individuals. These algorithms can provide insights and assist clinicians in making more accurate and efficient diagnoses. Developmental disorders are chronic disabilities that have a critical impact on many people’s daily performance. The number of people with autism is increasing around the world, which leads families to have increased anxiety about their children and requires an examination to ensure the health of their children if they are autistic. An autistic patient needs special care in order to develop perceptual skills to communicate with their family and society. When an autistic patient is diagnosed at an early stage, the results of behavioral therapy will be more effective. Computer vision techniques can analyze facial features, eye gaze, and other visual cues to detect atypical patterns associated with autism. By analyzing videos or images of individuals during social interactions, AI algorithms can help identify potential signs of autism . Eye-tracking is the process of capturing, tracking and measuring eye movements or the absolute point of gaze (POG), which refers to the point where the eye gaze is focused in the visual scene . The eye-tracking technology received particular attention in the ASD context since abnormalities of gaze have been consistently recognized as the hallmark of autism in general. The Psychology literature is replete with studies that analyzed eye movements in response to verbal or visual cues as signs of ASD . The main objective of this study is to build a deep learning-model using a convolutional neural network with transfer learning to identify autistic children using Eye-tracking. This model is designed to help families and psychiatrists to diagnose autism easily and efficiently. The current study applied multi of the most popular deep learning and machine learning algorithms, such as CNN, ANN, ConvNetV2 and machine learning model : non neural network (Naive Bayes) , to the dataset. The dataset consists of snapshoot of eye movement divided into classes: autistic and non autistic children. Premise that visual representations of eye-tracking snapshoot can discriminate the gaze behavior of autism. At its core, the key idea is to compactly render eye movements into an image-based format while maintaining the dynamic characteristics of eye motion (e.g. velocity, acceleration) using color gradients. In this manner, the prediction problem can be approached as an image classification task.

RELATED WORK

Plentiful studies sought to take advantage of eye tracking for the study and analysis of ASD such as Duan, H., et al.[1] developed a dataset of 14 autistic children and 14 typically developing by recording their eye movement data. In this study, they divided the 300 images (animals, buildings, objects, natural scenes…) into 10 sessions by showing 30 images in each session and each image was presented in the screen for 3 seconds than one-second gray screen and so on. Tobii T120 Eye Tracker was used to record eye movement. The analysis of data collected from the fixation position they set 1 in when the map is available *otherwise 0, for both ASD and TD. The result of the study found the ASD have more fixation in the hand and objects, the TD more fixation in faces.*

in [2] this study test whether visual processing differences between adults with and without high functioning autism captured through eye tracking can be used to detect autism. the researchers recorded the eye movements of adult participants with and without autism while they look for information within web pages. then they use the recorded eye-tracking data to train machine learning classifiers to detect the condition. The data was collected as part of two separate studies involving a total of 71 unique participants (31 with autism and 40 control), which enabled the evaluation of the approach on two separate groups of participants, using different stimuli and tasks. We explore the effects of a number of gaze-based and other variables, showing that autism can be detected automatically with around 74% accuracy.

Wan, G., et al. [3] suggested a method for earlier detection of autism. The dataset was 37 with autism and 37 typically developing children between 4–6 years old. The experimental stimuli utilized the SMI RED250 portable Eye Tracking system. The 10-second silence video clip of an Asian female speaking showed to the children. The ten AOI were tested: background; body: shoulders, neck, chest and hair, outer-face: the face area excluding the mouth, nose, and eyes, nose, mouth, eyes, person: body, hair, face, face: outer-face, eyes, nose, and mouth. The classification accuracy was 85.1%, sensitivity of 86.5%, and specificity of 83.8%.

Tao, Y., et al. [4] used CNNS for ASD and TD classification based on the scanpath of the fixation point. They used a dataset provided by Saliency4ASD grand challenge which consisted of 300 images collected from 14 ASD and 14 TD. The result of the classification was 74.22% accuracy.

Dalrymple, K. A., et al.[5] used machine learning to classify two groups of the children, 37 children in 18 months old and 36 children in 30 month-old. The experimenter showed static Stimuli to the children and used a Tobii TX300 eye-tracking device to record the eye movement. They used Deep learning classification according to the number of the fixation maps. The result showed the children of ages 18 months more interested in the dark region and the children of 30 months more interested in the bright region. The classification was 0.70 accuracy.

In [6]this study, three artificial-intelligence techniques were developed, namely, machine learning, deep learning, and a hybrid technique between them, for early diagnosis of autism. The first technique, neural networks [feedforward neural networks (FFNNs) and artificial neural networks (ANNs)], is based on feature classification extracted by a hybrid method between local binary pattern (LBP) and grey level cooccurrence matrix (GLCM) algorithms. This technique achieved a high accuracy of 99.8% for FFNNs and ANNs. The second technique used a pre-trained convolutional neural network (CNN) model, such as GoogleNet and ResNet-18, on the basis of deep feature map extraction. The GoogleNet and ResNet-18 models achieved high performances of 93.6% and 97.6%, respectively. The third technique used the hybrid method between deep learning (GoogleNet and ResNet-18) and machine learning (SVM), called GoogleNet + SVM and ResNet-18 + SVM. This technique depends on two blocks. The first block used CNN to extract deep feature maps, whilst the second block used SVM to classify the features extracted from the first block.

dataset

the dataset is open source from research [7] and it's A group of 59 children took part in this study. The participants were broadly organized into two groups as: i) ASD-Diagnosed, or ii) Non-ASD.ASD-diagnosed children were examined in multidisciplinary ASD specialty clinics. The intensity of autism was estimated by psychologists based on the Childhood Autism Rating Scale(CARS. Table 1 gives summary statistics of the participants (e.g. gender distribution, age mean).[7]

Table 1: Summary statistics of participants.

Number of Participants	59
Gender Distribution (M / F)	38 (≈ 64%) / 21 (≈ 36%)
Number of Non-ASD	30
Number of ASD-Diagnosed	29
Age (Mean / Median) years	7.88 / 8.1
CARS (Mean / Median)	32.97 / 34.50

SMI Remote Eye Tracker (Red-m 250Hz) Main tools for performing eye tracking function. Devices belong to the categories A screen-based eye tracker. There are three basic categories of eye movements Intended to be caught by eye trackers, including: fixation, saccade, and blink. Fixed means A brief moment that occurs while the gaze is stationary enable the brain to function on objects process of recognition. The average duration is Fixed time was estimated at 330 ms. Also, accurate recognition Objects need to be scanned continuously at high speed Eye movements called saccades. A saccade involves 20 or more rapid ballistic jumps. Each lasts about 30-120 milliseconds. On the other hand, blinking It can often be a sign that the system is losing track opinion .

in Figure 1 examples of snapshoots visualizations corresponding to ASD and non-ASD participants. As it appears, the center of both images includes areas of high density, which probably represent one of the main points of focus in the video scenario. For example, it can be noticed that the ASD-diagnosed participant had a tendency to look at the bottom of the screen, where the eye-tracking device was placed. The visualizations resulted in an image dataset containing 547 images. Specifically, 328 images for the non-ASD participants, and 219 images for others (i.e. ASD-diagnosed). The default image dimension was set as 640x480.

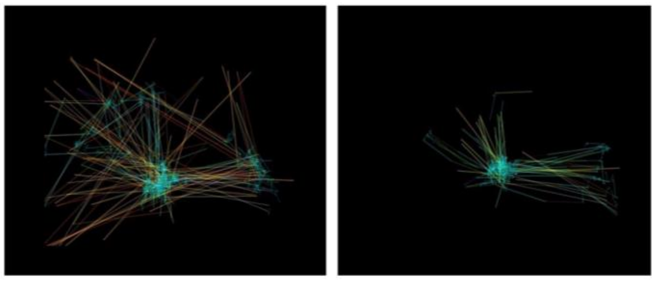


Figure1 : Visualization of eye-tracking snapshoots.

DATA AUGMENTATION AND PRE-PROCESSING

Image augmentation is a common technique for increasing the size of the dataset to generalize the model better and reduce the likelihood of overfitting. The basic idea of augmentation is to use a set of random image transformations such as rotations to generate synthetic samples. We applied augmentation to create variations in the visualization of the eye's scanning path. The dataset was expanded to include an additional 2,735 samples, generating 5 composite images per visualization. The data augmentation process has been greatly simplified by the Keras library, which contains an easy-to-use API for this purpose.

Dimension refers to the number of variables considered. In our case the first character is Dimensions are 640*480*3 (i.e. image size*) RGB components). it means more than 900,000 functions have to be considered, but this is possible Models become more difficult to train and increase significantly possibility of overfitting. A set of images is required to facilitate the learning process Sequential application of processing technology follows. First we scaled all images uniformly Maximum dimensions 100x100. The image was then transformed as follows: An even more useful grayscale format. Grayscale conversion shrinks the image Expression by removing hue and saturation Convey information while maintaining brightness. Specifically, the grayscale values were calculated from: Form a weighted sum of R, G, and B. From 100x100x3 to 100x100x1. It turned out that The grayscale spectrum was mostly sufficient We distinguish eye-tracking patterns in terms of: Velocity, acceleration, jerk. Finally principal component analysis (PCA) was implemented to convert grayscale Images in a more compressed format. use Orthogonal transform, PCA tries to transform Potentially correlated datasets (e.g. signals) images) to a linearly uncorrelated set size. PCA is one of the most popular Existing dimensionality reduction techniques Often used for data processing problems High dimensionality such as image compression . Figure 2 summarizes the pretreatment steps The dimensions output at each step are also shown.

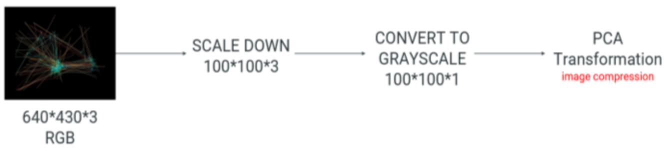


Figure 2 : The procedures of dimensionality reduction.

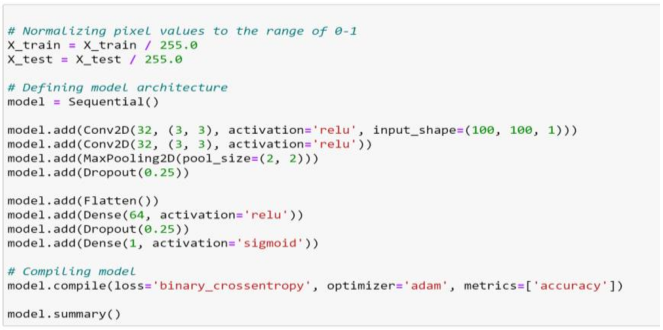
EXPERIMENTAL RESULTS

The experimental results are divided into four sections as follows:

i. CNN model
ii. ML :Naive Bayes
iii. ANN model
iv. ConvNetV2 model

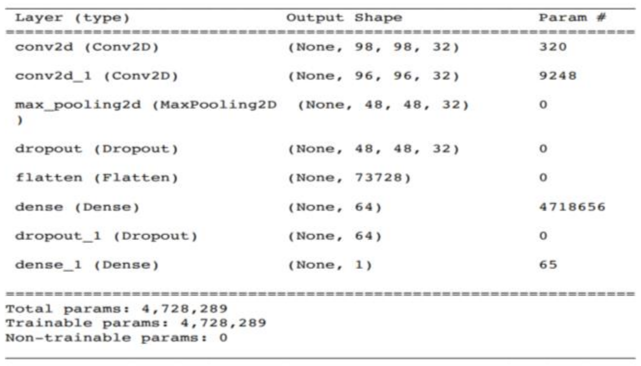
I. CNN model

The convolutional layer is an important part of the CNN model that is used for feature extraction. The first convolutional layer learns basic features: edges and lines. The next layer extracts the features of squares and circles. More complex features are extracted in the following layers, including the face, eyes, and nose. The basic components of the CNN model are the input layer, convolutional layer, activation function, pooling layer, fully connected layer, and output prediction.
In this model the database is trained in two layers 32 neurance , (3*3)kernal size ,activation function relu , adam optimizer , the accuracy in this model is 0.67%.



```
# Normalizing pixel values to the range of 0-1
X_train = X_train / 255.0
X_test = X_test / 255.0

# Defining model architecture
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 1)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(1, activation='sigmoid'))

# Compiling model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Figure 3 : model architecture



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 98, 98, 32)        320

 conv2d_1 (Conv2D)           (None, 96, 96, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 48, 48, 32)       0
 )

 dropout (Dropout)           (None, 48, 48, 32)        0

 flatten (Flatten)           (None, 73728)             0

 dense (Dense)               (None, 64)                4718656

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 4,728,289
Trainable params: 4,728,289
Non-trainable params: 0
```

Figure 4 : model summary

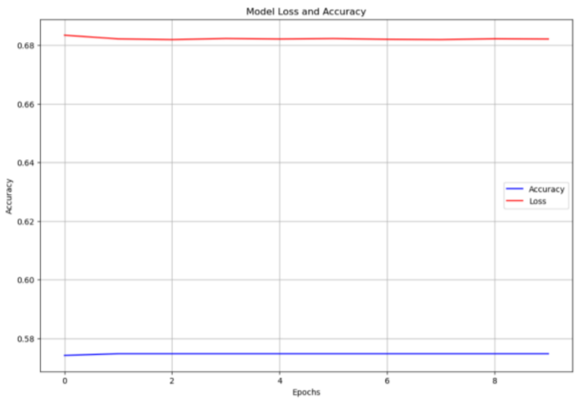


Figure 5 : model loss accuracy

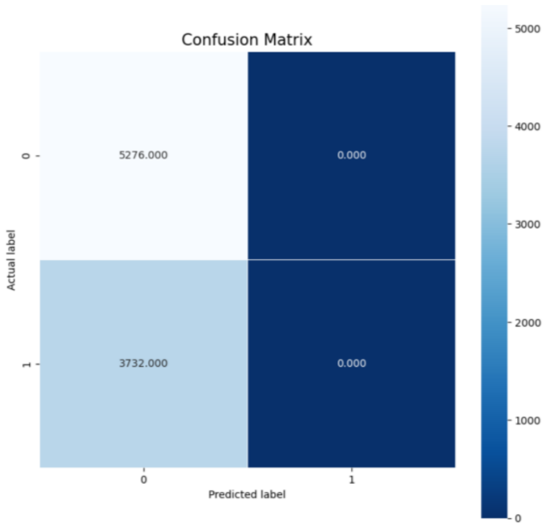
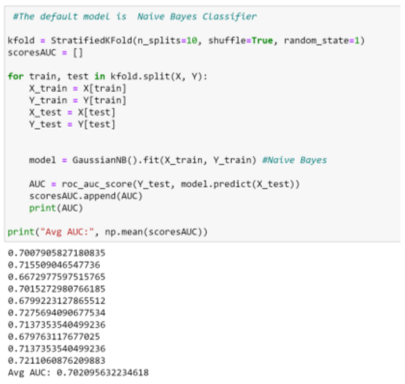


Figure 6 : CNN Confusion Matrix

II. ML : NAÏVE BAYES

We conducted our experiments using several ML model. Initially, non-neural network approaches were tested including: Naive Bayes this models were implemented using the Scikit-Learn library. Generally, the accuracy realized by that category of models was relatively fair (AUC ≈ 0.7).



```
 #The default model is  Naive Bayes Classifier

kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)
scoresAUC = []

for train, test in kfold.split(X, Y):
    X_train = X[train]
    Y_train = Y[train]
    X_test = X[test]
    Y_test = Y[test]


    model = GaussianNB().fit(X_train, Y_train) #Naive Bayes

    AUC = roc_auc_score(Y_test, model.predict(X_test))
    scoresAUC.append(AUC)
    print(AUC)

print("Avg AUC:", np.mean(scoresAUC))
```

```
0.7007905827180835
0.715509046547736
0.6672977597515765
0.7015272980766185
0.6799223127865512
0.7275694090677534
0.7137353540499236
0.679763117677025
0.7137353540499236
0.7211060876209883
Avg AUC: 0.702095632234618
```

Figure 7 : model architecture

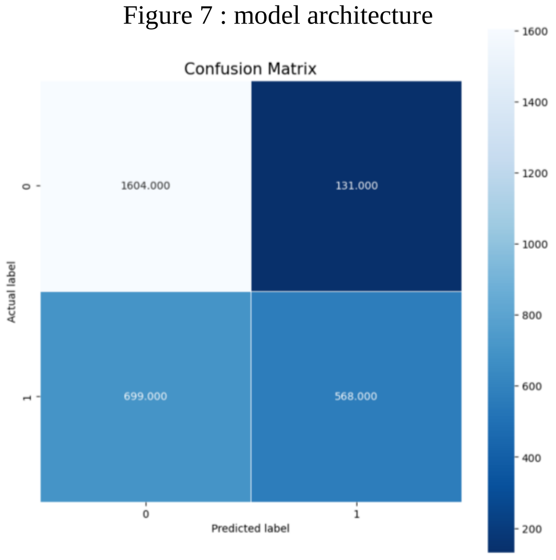


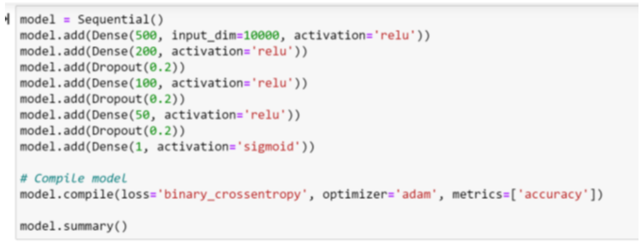
Figure 8 : ML Confusion Matrix

III. ANN

the model was experimented using various Artificial Neural Network (ANN) structures as follows. Initially, the simplest model structure included a single hidden layer of 50 neurons. The complexity of the model was gradually increased by adding more neurons (e.g. 200, 500), the accuracy in this model is 0.67%.

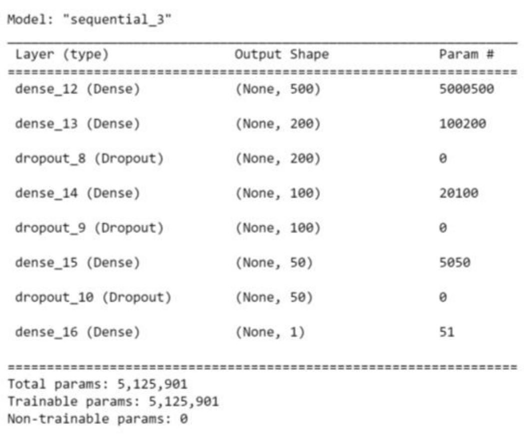
Table 2: ANN model architecture.

	Model Architecture	
	Hidden Layers	Number of Neurons
Experiment #1	Single-Layer	50
Experiment #2		200
Experiment #3		500
Experiment #4	Two-Layer	(80, 40)



```
model = Sequential()
model.add(Dense(500, input_dim=10000, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Figure 9 : model architecture



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 500)               5000500

 dense_13 (Dense)            (None, 200)               100200

 dropout_8 (Dropout)         (None, 200)               0

 dense_14 (Dense)            (None, 100)               20100

 dropout_9 (Dropout)         (None, 100)               0

 dense_15 (Dense)            (None, 50)                5050

 dropout_10 (Dropout)        (None, 50)                0

 dense_16 (Dense)            (None, 1)                 51

=================================================================
Total params: 5,125,901
Trainable params: 5,125,901
Non-trainable params: 0
```

Figure 10 : model summary

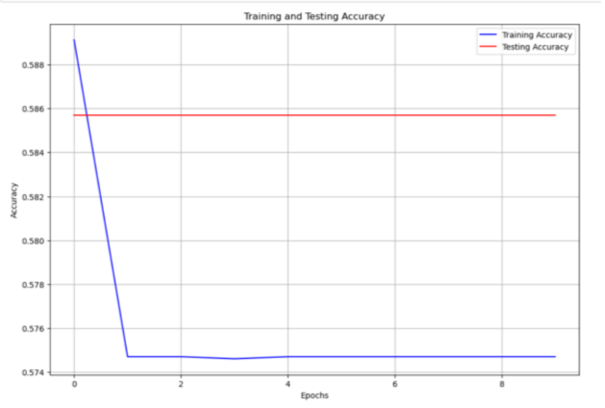


Figure 11 : model training and testing accuracy

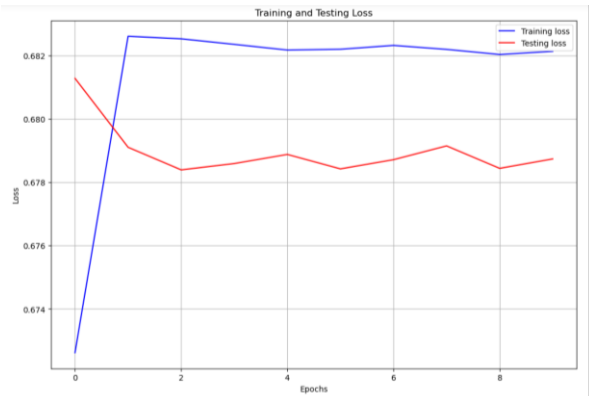


Figure 12 : model training and testing loss

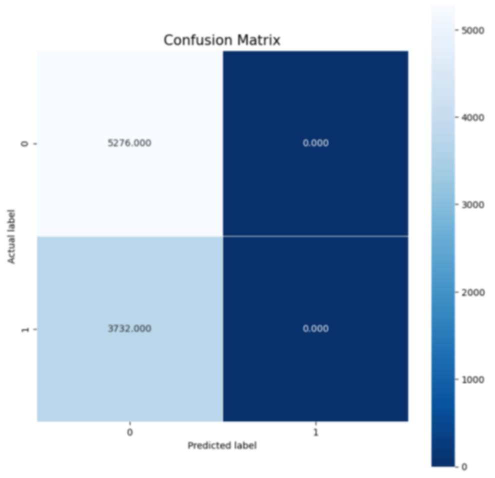
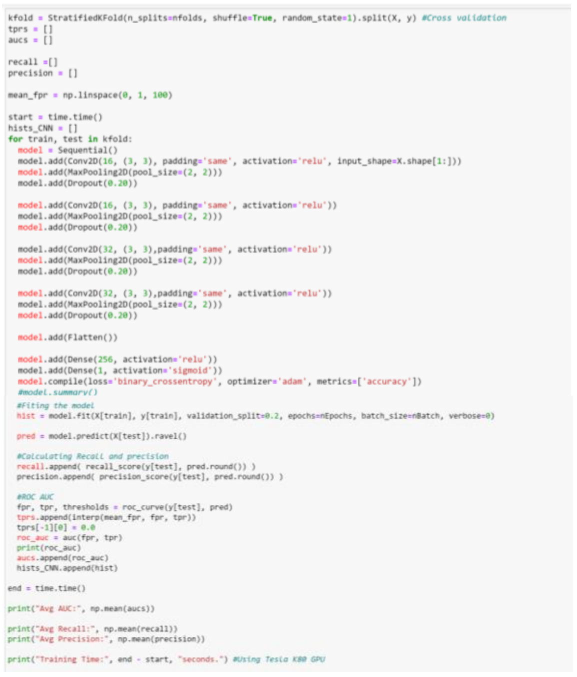


Figure 13 : ANN Confusion Matrix

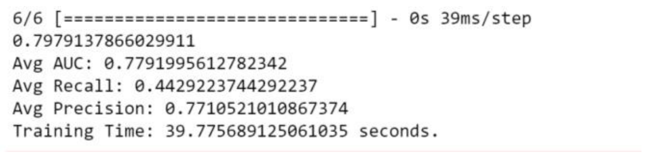
IV. ConvNetV2 model

In this model the database is trained in multi layers 16 neurance , (3*3)kernal size ,activation function relu , adam optimizer ,padding same and the accuracy in this model is the highest accuracy around 0.78%.



```
kfold = StratifiedKFold(n_splits=nfolds, shuffle=True, random_state=1).split(X, y) #Cross validation
tprs = []
aucs = []

recall =[]
precision = []

mean_fpr = np.linspace(0, 1, 100)

start = time.time()
hists_CNN = []
for train, test in kfold:
  model = Sequential()
  model.add(Conv2D(16, (3, 3), padding='same', activation='relu', input_shape=X.shape[1:]))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.20))

  model.add(Conv2D(16, (3, 3), padding='same', activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.20))

  model.add(Conv2D(32, (3, 3),padding='same', activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.20))

  model.add(Conv2D(32, (3, 3),padding='same', activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.20))

  model.add(Flatten())

  model.add(Dense(256, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  #model.summary()
  #Fiting the model
  hist = model.fit(X[train], y[train], validation_split=0.2, epochs=nEpochs, batch_size=nBatch, verbose=0)

  pred = model.predict(X[test]).ravel()

  #Calculating Recall and precision
  recall.append( recall_score(y[test], pred.round()) )
  precision.append( precision_score(y[test], pred.round()) )

  #ROC AUC
  fpr, tpr, thresholds = roc_curve(y[test], pred)
  tprs.append(interp(mean_fpr, fpr, tpr))
  tprs[-1][0] = 0.0
  roc_auc = auc(fpr, tpr)
  print(roc_auc)
  aucs.append(roc_auc)
  hists_CNN.append(hist)

end = time.time()

print("Avg AUC:", np.mean(aucs))

print("Avg Recall:", np.mean(recall))
print("Avg Precision:", np.mean(precision))

print("Training Time:", end - start, "seconds.") #Using Tesla K80 GPU
```

Figure 14 : model architecture



```
6/6 [==============================] - 0s 39ms/step
0.7979137866029911
Avg AUC: 0.7791995612782342
Avg Recall: 0.44292237442922237
Avg Precision: 0.7710521010867374
Training Time: 39.775689125061035 seconds.
```

Figure 15 : model accuracy

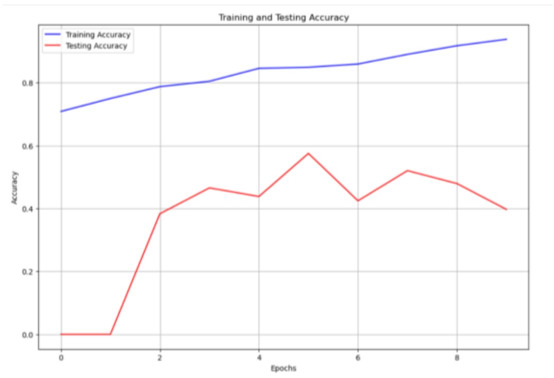


Figure 16 : model training and testing accuracy

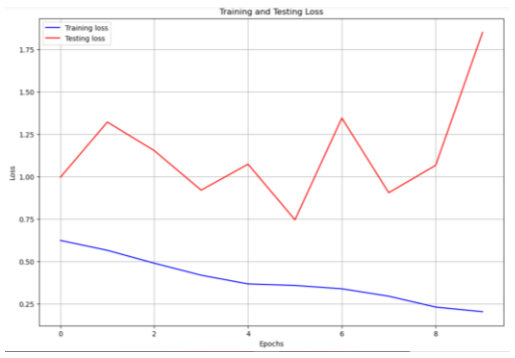


Figure 17 : model training and testing loss

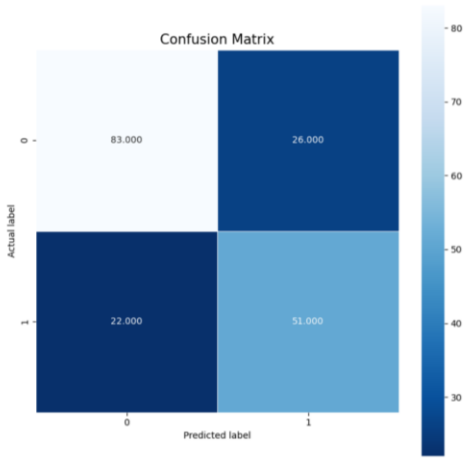


Figure 18 : convNetV2 Confusion Matrix

ROC curve

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a ROC curve of the model .

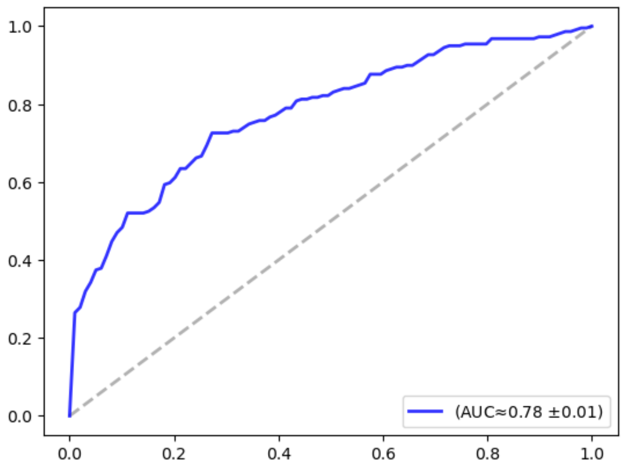
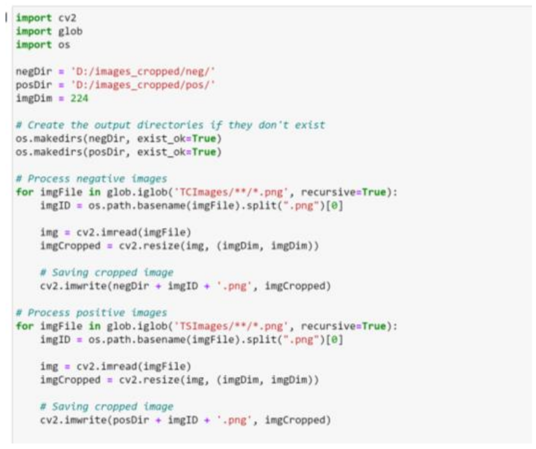


Figure 19: Receiver Operating Characteristic curve

Cropping

Image Cropping is a common photo manipulation process, which improves the overall composition by removing unwanted regions in this model we did cropping before training so this is one of the effectiveness that increasing the accuracy.



```
import cv2
import glob
import os

negDir = 'D:/images_cropped/neg/'
posDir = 'D:/images_cropped/pos/'
imgDim = 224

# Create the output directories if they don't exist
os.makedirs(negDir, exist_ok=True)
os.makedirs(posDir, exist_ok=True)

# Process negative images
for imgFile in glob.iglob('TCImages/**/*.png', recursive=True):
    imgID = os.path.basename(imgFile).split(".png")[0]

    img = cv2.imread(imgFile)
    imgCropped = cv2.resize(img, (imgDim, imgDim))

    # Saving cropped image
    cv2.imwrite(negDir + imgID + '.png', imgCropped)

# Process positive images
for imgFile in glob.iglob('TSImages/**/*.png', recursive=True):
    imgID = os.path.basename(imgFile).split(".png")[0]

    img = cv2.imread(imgFile)
    imgCropped = cv2.resize(img, (imgDim, imgDim))

    # Saving cropped image
    cv2.imwrite(posDir + imgID + '.png', imgCropped)
```

Figure 19 : cropping code

LIMITATIONS

Even though the results presented in this study are promising, a set of limitations need to be highlighted as follows. The primary limitation could be the relatively small number of participants in the dataset . Another relevant issue of concern is the duration One of the biggest obstacles in this project is the limited resources such as the computers used were very slow and unable to process the models and the limited database Despite limitations, the study is still believed to serve as a kernel for further interesting applications of the proposed approach.

Conclusion

The coupling of eye-tracking, visualization and ML and DL can hold a strong potential for the development of an objective tool to assist the diagnosis of ASD. The ML and DL experiments confirmed the core idea behind our approach, which hinges on the visual representation of eye-tracking snapshoots . The classification accuracy indicated that visualizations were able to some extent he can succeed pack the information of eye motion and its underlying dynamics. From a practical standpoint, it is noteworthy that we could reach that accuracy with largely ML and ConvNetV2 models. ML model Using non-neural network: Naive Bayes , the prediction accuracy could go beyond 70% while the ConvNetV2 is reach the highest accuracy 78% In comparison with other Deep Learning models . should be compared positively to related efforts that used different sets of features and more complex models.

I. References

[1] Duan, H., Zhai, G., Min, X., Che, Z., Fang, Y., Yang, X., & Callet, P. L. (2019, June). A dataset of eye movements for the children with autism spectrum disorder. In Proceedings of the 10th ACM Multimedia SystemsConference (pp. 255-260). ACM.

[2]Yaneva, V., Eraslan, S., Yesilada, Y., & Mitkov, R. (2020). Detecting high-functioning autism in adults using eye tracking and machine learning. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 28(6), 1254-1261.

[3]Wan, G., Kong, X., Sun, B., Yu, S., Tu, Y., Park, J., ... & Lin, Y. (2019). Applying Eye Tracking to Identify Autism Spectrum Disorder in Children. Journal of autism and developmental disorders, 49(1), 209-215.

[4]Tao, Y., & Shyu, M. L. (2019, July). SP-ASDNet: CNN-LSTM Based ASD Classification Model using Observer ScanPaths. In 2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (pp. 641-646). IEEE.

[5]Dalrymple, K. A., Jiang, M., Zhao, Q., & Elison, J. T. (2019). Machine learning accurately classifies age of toddlers based on eye tracking. Scientific reports, 9(1), 6255

[6] Ahmed, I. A., Senan, E. M., Rassem, T. H., Ali, M. A., Shatnawi, H. S. A., Alwazer, S. M., & Alshahrani, M. (2022). Eye tracking-based diagnosis and early detection of autism spectrum disorder using machine learning and deep learning techniques. Electronics, 11(4), 530.

[7] Carette, R., Elbattah, M., Cilia, F., Dequen, G., Guerin, J. L., & Bosche, J. (2019, February). Learning to Predict Autism Spectrum Disorder based on the Visual Patterns of Eye-tracking Scanpaths. In HEALTHINF (pp. 103-112).

[8] Ahmed, Z. A. T., & Jadhav, M. E. (2020, February). A review of early detection of autism based on eye-tracking and sensing technology. In 2020 International Conference on Inventive Computation Technologies (ICICT) (pp. 160-166). IEEE.

[9]] Yaneva, V., Eraslan, S., Yesilada, Y., & Mitkov, R. (2020). Detecting high-functioning autism in adults using eye tracking and machine learning. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 28(6), 1254-1261.

[10] Yaneva, V., Temnikova, I., & Mitkov, R. (2015, October). Accessible texts for autism: An eye-tracking study. In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (pp. 49-57).