**Work on Autism:**

Akhter et al [400] conducted on an analytic application of a method to justify the best

Intervention Technology treating Autism-related difficulties. Based on the learning obtained from the training data, a model was built that was capable of efficiently identifying and detecting treatment technologies for a given deficit or collection of deficiencies. The first model Decision Tree Classifier had 67% accuracy and the

second model named the Ensemble Vote Classifier, contained 75% of accuracy. As a result, it was applicable to autistic people’s medical, nursing, mentoring, and daily

requirements.The findings of the study can be used to compare and contrast

survey-based research trends in the field of results, users, and disorders targeted.

[400]

@inproceedings{akhtar2020predictive,

title={Predictive analytics using a machine learning model to recommend the most suitable intervention technology for autism related deficits},

author={Akhtar, Nishat and Feeney, Mairead},

booktitle={2020 31st Irish Signals and Systems Conference (ISSC)},

pages={1--6},

year={2020},

organization={IEEE}

}

In another study, Altay et al [401] did a study where kids aged 4 to 11 years were tested to be analyzed by the ASD disorder classification approach. The LDA and the KNN algorithms were applied in the classification approach. As a result of the application, the LDA algorithm provides a better result than the KNN algorithm at the accuracy rate.

[401]

@inproceedings{altay2018prediction,

title={Prediction of the autism spectrum disorder diagnosis with linear discriminant analysis classifier and K-nearest neighbor in children},

author={Altay, Osman and Ulas, Mustafa},

booktitle={2018 6th international symposium on digital forensic and security (ISDFS)},

pages={1--4},

year={2018},

organization={IEEE}

}

Dutta et al [402] diagnosed a method that takes contributions from numerous sensors and cell phones.The study included sensor data that was brought through an application running on a portable hub. In Autism Disease diagnosis, the method showed that the most elevated exactness is 70% using Machine learning Association Rule with minimum Redundancy Maximum Relevance (mRMR) strategy.

[402]

@inproceedings{dutta2017machine,

title={A machine learning-based method for autism diagnosis assistance in children},

author={Dutta, Sushama Rani and Giri, Soumyajit and Datta, Sujoy and Roy, Monideepa},

booktitle={2017 International Conference on Information Technology (ICIT)},

pages={36--41},

year={2017},

organization={IEEE}

}

In a brief study on ASD, Tamilarasi et al [403] initiated a deep learning-based system with ResNet-50 design and executed the examination for face image classification of ASD.The proficient deep learning strategy additionally distinguished the specific expected opportunity to erase the highlights for each image from the data put away.

[403]

@inproceedings{tamilarasi2020convolutional,

title={Convolutional neural network based autism classification},

author={Tamilarasi, F Catherine and Shanmugam, J},

booktitle={2020 5th International Conference on Communication and Electronics Systems (ICCES)},

pages={1208--1212},

year={2020},

organization={IEEE}

}

Zhou et al [404] tested machine learning models to identify ASD established on multiparametric MRI data. To lessen classification error in the study of estimation bias, Cross validation was operated. These discoveries suggested that machine learning methods of multi-parametric MRI data can be advantageous in the classifying of ASD. Moreover, Particular characteristics were connected with conclusions of behavioral analysis that can also demonstrate effectiveness in keeping an eye on symptoms covering time.

[404]

@article{zhou2014multiparametric,

title={Multiparametric MRI characterization and prediction in autism spectrum disorder using graph theory and machine learning},

author={Zhou, Yongxia and Yu, Fang and Duong, Timothy},

journal={PloS one},

volume={9},

number={6},

pages={e90405},

year={2014},

publisher={Public Library of Science San Francisco, USA}

}

**Work on meltdown :**

A study conducted by Rane et al [405] proposed a Logistic Regression model in order to create an Open Source based classifier for predicting ASD meltdowns. 539 samples were collected from ABIDE database. Later, LASSO regression and 5-fold cross-validation were used basically to get rid of over-fitting in the final stage, achieving an accuracy of 62%

[405]

@article{rane2017developing,

title={Developing predictive imaging biomarkers using whole-brain classifiers: Application to the ABIDE I dataset},

author={Rane, Swati and Jolly, Eshin and Park, Anne and Jang, Hojin and Craddock, Cameron},

journal={Research Ideas and Outcomes},

volume={3},

pages={e12733},

year={2017},

publisher={Pensoft Publishers}

}

.

Hyde et al [406] used a Decision tree to examine a new way for predicting ASD with the help of employment power. The proposed model was successfully able to figure out the ASD rate within all hired employers and it specifically showed an accuracy rate of 75% and 82% that shows a high hope in predicting ASD meltdowns.

[406]

@inproceedings{hyde2018predicting,

title={Predicting employer recruitment of individuals with autism spectrum disorders with decision trees},

author={Hyde, Kayleigh and Griffiths, Amy-Jane and Giannantonio, Cristina and Hurley-Hanson, Amy and Linstead, Erik},

booktitle={2018 17th IEEE international conference on machine learning and applications (ICMLA)},

pages={1366--1370},

year={2018},

organization={IEEE}

}

Gong et al [407] continued his existing work with the help of Association rules. The author drew out genes associated with ASD from 16,869 publication abstracts. With both candidate genes and identified genes, Association rules were applied. This research also highlights the potential of text classification for predicting undiscovered genes in ASD meltdowns in the future.

[407]

@article{gong2012prediction,

title={Prediction of autism susceptibility genes based on association rules},

author={Gong, Lejun and Yan, Yunyang and Xie, Jianming and Liu, Hongde and Sun, Xiao},

journal={Journal of neuroscience research},

volume={90},

number={6},

pages={1119--1125},

year={2012},

publisher={Wiley Online Library}

}

| Reference | Author | Based on | Approach | Year | Limitations |
| --- | --- | --- | --- | --- | --- |
| [1] | Lisa Feldman Barrett, Batja Mesquita, Kevin N. Ochsner, and James J. Gross | Emotion Recognition | facial action coding  system (FACS), | 2007 | The algorithm has trouble  categorizing facial  expressions and is unable  to recognize and detect any  underlying emotions that a  viewer may feel when they  view the image |
| [2] | Derick Leony a, Pedro J. Muñoz-Merino a, Abelardo Pardo b, Carlos Delgado Kloos | Emotion Recognition | Visualizations  In a computer interaction based environment | 2013 | Small dataset (limited scale) |
| [3] | Magali Batty, Margot J. Taylor | Facial Emotional expression | intensity  (10%, 55%, 90%); kind of  expression (anger, disgust,  fear, joyful, sad, surprise);  and observer gender  (woman, man) as a  subject factor | 2016 | Neutral replies that were  90% neutral and 10%  expressive for extremely  low intensity expressions  were recorded as erroneous |
| [4] | Md Inzamam Ul Haque; Damian Valles | Facial Expression recognition on autistic children | CNN model | 2018 | Uncontrolled environment |
| [5] | Sara Karim; Nazina Akter; Muhammed J. A. Patwary | ASD Meltdown | Fuzzy semi-supervised learning | 2022 | Mid-fuzziness segments were more likely to misclassified ASD Meltdown |
| [6] |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

[1]

@article{barrett2007experience,

title={The experience of emotion},

author={Barrett, Lisa Feldman and Mesquita, Batja and Ochsner, Kevin N and Gross, James J},

journal={Annu. Rev. Psychol.},

volume={58},

pages={373--403},

year={2007},

publisher={Annual Reviews}

}

[2]

@article{leony2013provision,

title={Provision of awareness of learners’ emotions through visualizations in a computer interaction-based environment},

author={Leony, Derick and Mu{\~n}oz-Merino, Pedro J and Pardo, Abelardo and Kloos, Carlos Delgado},

journal={Expert Systems with Applications},

volume={40},

number={13},

pages={5093--5100},

year={2013},

publisher={Elsevier}

}

[3]

@article{batty2003early,

title={Early processing of the six basic facial emotional expressions},

author={Batty, Magali and Taylor, Margot J},

journal={Cognitive brain research},

volume={17},

number={3},

pages={613--620},

year={2003},

publisher={Elsevier}

}

[4]

@inproceedings{haque2018facial,

title={A facial expression recognition approach using DCNN for autistic children to identify emotions},

author={Haque, Md Inzamam Ul and Valles, Damian},

booktitle={2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)},

pages={546--551},

year={2018},

organization={IEEE}

}

[5]

@inproceedings{karim2022predicting,

title={Predicting autism spectrum disorder (ASD) meltdown using fuzzy semi-supervised learning with NNRW},

author={Karim, Sara and Akter, Nazina and Patwary, Muhammed JA},

booktitle={2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)},

pages={367--372},

year={2022},

organization={IEEE}

}

[6]

**CHAPTER III**

**BACKGROUND TOOLS AND TECHNOLOGIES**

**3.1 Overview**

In this chapter, an overview of numerous commonly used neural networks and the related terminology are presented, as well as an introduction to the libraries that are used in the building of networks. The purpose of this part is to acquaint the readers with the many resources and ideas that will be referred to and employed over the whole of the book.

**3.2 Machine Intelligence Libraries**

The models were built using code taken from a number of different libraries; we will go through some of these libraries in the next section.

**3.2.1 TensorFlow**

TensorFlow is a popular open-source machine learning framework developed by Google. It is designed to support a wide range of tasks, including image and speech recognition, natural language processing, and time-series analysis. TensorFlow uses a dataflow graph to represent mathematical computations, making it easy to parallelize and distribute computations across multiple CPUs or GPUs. The framework also includes a range of tools for data preprocessing, model selection, and evaluation, as well as a powerful and flexible API for building and training deep neural networks. With its scalable architecture and extensive functionality, TensorFlow has become a widely used tool for researchers, developers, and businesses across a range of industries.

**3.2.2 Keras**

Keras is a high-level deep learning framework that provides a user-friendly interface for building and training deep neural networks. It was developed with the aim of making it easier for researchers and developers to experiment with deep learning models without requiring extensive knowledge of the underlying mathematical concepts. One of the key features of Keras is its modularity, which allows users to easily construct and combine various layers to form complex neural network architectures.Keras supports a wide range of layer types, including convolutional layers for image processing, recurrent layers for sequence data, and dense layers for general purpose modeling. Keras also provides a number of pre-built models for common deep learning tasks, such as image classification and language processing. These models are easily accessible and can be customized to fit specific needs, making it easy to quickly develop and test new ideas. Another strength of Keras is its support for multiple backend frameworks, including TensorFlow, Theano, and CNTK. This allows users to choose the backend that best fits their needs, while still taking advantage of Keras’ high-level API.

**3.2.3 Scikit Learn**

Scikit Learn is a popular Python library for machine learning that provides a range of tools for data preprocessing, classification, regression, clustering, and more. It is designed to be simple and efficient, making it easy for both beginners and experienced data scientists to work with large datasets and develop models quickly. One of the strengths of Scikit Learn is its focus on consistency and usability. The library provides a uniform interface for a wide range of machine learning algorithms, allowing users to easily switch between different models and compare their performance. The library also includes a range of preprocessing tools, such as feature scaling and dimensionality reduction, which can help improve the accuracy of machine learning models.

**3.2.4 CV2**

The cv2 module is the main module in OpenCV that provides developers with an easy-to-use interface for working with image and video processing functions.[100]

In Python, OpenCV is accessed through the cv2 module, which provides a user-friendly interface to leverage the powerful features of the OpenCV library. With cv2, developers can easily read, manipulate, and analyze images and videos, performing tasks that range from simple operations like resizing and cropping to complex ones like object detection and image segmentation.cv2 empowers developers to create innovative solutions that leverage the full potential of computer vision technology.

**3.2.5 Numpy**

NumPy is a fundamental Python library for scientific computing, providing powerful support for n-dimensional arrays (ndarray), which are efficient, fixed-size, and homogeneous. It enables advanced mathematical operations, including element-wise computations, linear algebra, statistics, and random number generation, all optimized for performance with pre-compiled C code. Key features like vectorization and broadcasting simplify and speed up array operations. Widely used in data science and machine learning, NumPy is essential for efficient manipulation and analysis of large datasets.

**3.2.6 Mediapipe**

MediaPipe is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio. Using MediaPipe, such a perception pipeline can be built as a graph of modular components.[101]

It is a versatile framework that leverages the power of GPUs for faster processing of multimedia tasks. It utilizes parallel processing to handle multiple tasks simultaneously, such as processing multiple video streams or running several computer vision models. Additionally, MediaPipe integrates with OpenCV, a robust library for computer vision, to enhance its capabilities for tasks like video capture, processing, and rendering. By teaming up with TensorFlow, Google's machine learning tool, MediaPipe simplifies the integration of pre-trained or custom models for tasks such as face recognition and speech understanding. With support for popular programming languages like C++, Java, and Python, MediaPipe is easy to incorporate into various projects, making it a valuable tool for multimedia processing and machine learning applications.

**3.2.7 Scipy Spatial**

Scipy Spatial Distance is a module in the Scipy library that provides functions for calculating distances between points in n-dimensional space. It also includes functions for computing distance matrices, which are matrices that contain the distances between all pairs of points in a given set.

The Scipy Spatial Distance module offers a wide range of distance metrics, including Euclidean distance, Chebyshev distance, Hamming distance, and many more. Each metric has its own mathematical formula for calculating distances between points.It also provides functions for working with data sets that have missing or invalid values. These functions can help ensure that your calculations are accurate even when dealing with imperfect data.Overall, Scipy Spatial Distance is a powerful tool for anyone working with spatial data in Python. Whether you’re analyzing geographic data, clustering data points, or performing machine learning tasks, this module can help you accurately measure distances and make informed decisions based on your results.[102]

**3.3 Preprocessing Strategies**

Throughout the preprocessing stages of the work we have reshaped and normalized the pixel values.Facial images typically have only one channel (grayscale). Our code adds a new dimension using np.expand\_dims to make it compatible with the Convolutional Neural Network (CNN) model, which expects a format like (number of images, height, width, channels).The code uses to\_categorical from tensorflow.keras.utils to convert the emotion labels (likely integer values representing different expressions) into one-hot encoded vectors which is a common technique for representing categorical data in neural networks, where each category has its own binary vector with a single "1" and all other values as "0".

**3.4 Neural Networks Related to Our Work**

In this section, a brief overview is provided of the neural networks that pertain to the study. This summary aims to familiarize the neural networks that are most relevant to the study and set the stage for the subsequent discussions on their applications and performance.

**3.4.1 Convolutional Neural Network**

A convolutional neural network (CNN) is a special kind of artificial neural network that was developed for the sole purpose of processing pixel input. CNNs are used in image recognition and digital processing. Convolutional neural networks make use of a variety of filters in order to identify picture characteristics that allow for the classification of objects. The extraction of these features is being handled by a great number of kernels inside the CNN system. It is common practice to call attention to noteworthy differences in pixel values by using the edge kernel. CNNs are used to translate two dimensional or multi- dimensional input to a single dimensional output variable. They have proved to be so beneficial that they are currently the technique of choice for solving any kind of prediction problem that involves picture data. The components of a convolutional neural network are as follows: an input layer, an output layer, and hidden layers. Any layers in a feedforward neural network that are located in the middle are referred to be hidden since the activation function and the final convolution both conceal the inputs and outputs of those layers. As compared to a standard neural network, a convolutional network is highly distinct due to the fact that the neurons in each of its layers are organized according to height, weight, and depth.

**3.4.2 Dense**

When one layer in a neural network is densely coupled to the layer above it, this means that each neuron in the layer is connected to each and every other neuron in the layer above it. This layer is the one that is used in artificial neural network networks the most often, hence it is the one with the highest prominence. In a model, each neuron in the layer below it transmits signals to the neurons in the layer above it, which are responsible for multiplying matrices and vectors. During matrix vector multiplication, the row vector of the output from the layers that came before it is the same as the column vector of the layer that contains the dense information. In order to multiply matrices with vectors, the row vector and the column vector must both have the same number of columns. The general formula for a matrix-vector product is:

Ax=*a*11 *a*12 *... a*1*n*

*a*21 *a*22*... a*2*n*

*. . . .. . . .*

*am*1 *am*2 *... amn*

*x*1

*x*

2

*.*

*.*

*xn*

=

*a*11*x*1+*a*12*x*2+*...*+*a*1*nxn*

*a*21*x*1+*a*22

*x*2+

*...*+*a*2*nxn*

*........*

*........*

*am*1*x*1+*am*

2*x*2+

*...*+*amnxn*

Where x is a matrix that has a diagonal value of 1, and A is a matrix that has the dimensions M x N. The values that are included inside the matrix are the learnt parameters from the layers that came before it, and backpropagation may also be used to update these values. Backpropagation is by far the most common and widely used method for training feedforward neural networks. Backpropagation is the process that, in a neural network, generally determines how to compute the gradient of the loss function in relation to the network weights for a single input or output. The theory that came before this one states that the output of the dense layer will be an N-dimensional vector. It has come to our attention that the dimensions of the vectors are being shrunk. As a consequence of this, each neuron in a dense layer is responsible for modifying the dimension of the vectors. According to what was said before, information is sent to each neuron in the thick layer from each neuron in the layers that came before it. This output then travels through the dense layer, which should contain N neurons if it is to be considered a dense layer. If the layer above creates a M x N matrix by adding up the results of each neuron, this output will go via the dense layer. It’s possible that we’ll build it using Keras; in the next portion of this article, we’ll have a look at some of the most important parameters of the dense layer and the definitions of those parameters.

**3.4.3 Flatten**

It is common practice to employ a neural network’s Flatten layer, which is a simple but necessary layer, to convert the output of a convolutional or pooling layer into a feature vector that is two-dimensional in shape. This layer is responsible for taking the output of a multidimensional array from the layer below it and turning it into a single vector so that it may be sent on to a fully linked layer for further processing. The Flatten layer is essential because the majority of machine learning methods, including logistic regression, need for the input data to be in the form of a flat vector. This makes the Flatten layer an essential component. Neural networks are able to readily integrate with these algorithms and take benefit of the predictive potential they provide if they make use of the Flatten layer. Moreover, the Flatten layer may assist minimize the number of parameters in the model by consolidating numerous feature maps into a single vector. This is accomplished via the use of vectorization. In general, the Flatten layer is a simple yet essential component of many different architectural designs for neural networks. It allows these networks

to handle complicated input data in an effective manner and attain high levels of accuracy.

**3.4.4 Convolution Operation**

The convolutional operation is an essential component in the construction of convolutional neural networks (CNNs). The process entails applying a filter, which is often referred to as a kernel, to an input picture in order to extract features that are significant. The filter iteratively applies itself to the picture, calculating a dot product at each place along the way, and output a feature map at the end. The size of the input picture, the size of the filter, and the stride, which defines the distance the filter travels between each calculation, all influence the size of the feature map that is produced as the output of the algorithm. By altering the weights of the filters

while the convolutional layer is being trained, it is possible for the layer to learn how to recognize certain characteristics, such as edges, corners, and textures. The network is able to learn hierarchical representations of the input data as a result of this,with lower levels identifying basic characteristics and higher layers identifying more complex ones. After a series of convolutional layers, a network will often go on to a series of pooling layers. These layers downsample the feature maps, hence reducing the overall size of the maps and assisting the network in becoming more generalized.

**3.4.5 MaxPooling Operation**

It is common practice to do maxpooling after a layer of convolutional filters has been applied. This technique is helpful for lowering the spatial dimensions of the feature maps while still preserving the characteristics that are most important to the analysis. In order to carry out the maxpooling operation, the input feature map has to be cut up into a number of pooling windows first. These windows are non overlapping rectangular regions, and they do not overlap with one another. The maxpooling technique results in an output that is the maximum value that was found to be present inside each pooling window. This output is produced by the maxpooling procedure. maxpooling greatly reduces the spatial dimensions of the feature maps by picking the feature with the highest value and pooling it with the other features. This is done while keeping the features that are most essential to the analysis.

**3.4.6 Dropout Layer**

Regularization strategies like the dropout layer are often employed in neural networks to reduce overfitting and improve accuracy. It does this by randomly removing a certain proportion of neurons from the layer that is being trained. This compels the network to acquire redundant representations and lessens the likelihood that it would become too dependent on certain characteristics. During the process of inference, all neurons are active, but the output of each neuron is scaled according to the dropout probability. This is done to maintain the constant value that is predicted from each neuron. Dropout layers may be introduced to any region of a neural network and are most successful when used in combination with large or deep networks. Dropout

layers can also be removed from a neural network. They have been found to increase neural networks’ generalization performance on a variety of tasks, including image recognition, natural language processing, and voice recognition, among others. The dropout approach is frequently used in the deep learning field and has developed into an important instrument for the construction of neural networks that are precise and resilient.

**3.4.7 Activation Function**

An activation function is a mathematical function that is applied to the output of each neuron in a layer of a neural network.It’s job is to provide the model some non-linearity so that it may learn to recognize intricate patterns in the data. This is accomplished via the model’s ability to learn. There are a wide variety of activation functions accessible, and each one has both advantages and disadvantages unique to itself. The rectified linear unit(ReLU),softmax functions are the types of activation functions that are used in our code. The ReLU function is frequently utilized in the hidden layers of a neural network due to its ease of use and its capacity to circumvent the vanishing gradient problem. On the other hand,By incorporating the softmax activation function, multi-class classification models can generate more meaningful and interpretable outputs, allowing for better decision-making and evaluation.These activation functions are all examples of what are known as "enhancement functions." The selection of an activation function is an essential factor to take into account while developing and perfecting a model since it has the potential to have a major influence on the performance and learning capabilities of a neural network

**3.4.8 Adam Optimizer**

Adam optimizer is a popular optimization technique in deep neural networks that

combines the advantages of Adaptive Gradient Algorithm (AdaGrad) and Root Mean

Square Propagation. Adam optimizer is used extensively in the field of artificial

intelligence research (RMSProp). It is a technique for calculating individual adaptive

learning rates based on the parameters, and it is referred to as an adaptive learning

rate method. An ongoing estimate of the first and second moments of the gradient is

kept by the Adam optimizer. This estimate is then utilized to update the parameters.

This makes it a very efficient approach for improving the neural network since it

helps to adapt the learning rate depending on the gradients. The capacity of the

Adam optimizer to converge in a speedy and effective manner is one of the primary

advantages offered by this tool. It has been shown to be very successful in a broad

variety of deep learning tasks, such as the categorization of images, the processing of

natural languages, and the identification of voices. The algorithm’s user-friendliness

and adaptability have contributed to its rise in popularity among developers and

academics. These qualities enable the algorithm’s users to customize the optimization

process to better suit their own requirements.

**3.5 Evaluation Metrics**

**3.5.1 Accuracy**

Accuracy is a commonly used metric to evaluate the performance of a deep learning

model. It measures the proportion of correctly classified instances out of the total

number of instances. However, it has limitations in certain scenarios, such as when

the classes are imbalanced or when the cost of misclassifying one class is much higher

than the other. In these cases, other metrics such as precision, recall, and F1 score

may be more appropriate. It is also important to note that accuracy alone may

not provide enough information about the performance of the model and should be

complemented by other metrics and visualizations such as confusion matrix, ROC

curve, and precision-recall curve.

*Accuracy* =*T P* + *T N*

*T P* + *F P* + *T N* + *F N*

*∗* 100%

References:

[100] <https://konfuzio.com/en/cv2/#:~:text=The%20cv2%20module%20is%20the,commonly%20used%20functions%20in%20cv2>.

[101]

<https://viso.ai/computer-vision/mediapipe/#:~:text=for%20Enterprise%20Teams-,What%20is%20MediaPipe%3F,currently%20in%20alpha%20at%20v0>.

[102]

<https://pieriantraining.com/measuring-distance-with-scipy-spatial-distance/#:~:text=Scipy%20Spatial%20Distance%20is%20a,points%20in%20a%20given%20set>.