Webology (ISSN: 1735-188X)
Volume 19, Number 2, 2022

Enhancement Of Eye Tracking Based On Deep Learning Using Manta Ray Foraging Optimization (MRFO)

Ahmed A. Salman[1] , Mohammed H. Ali[2]

[1]Department of Electronic and Communication Engineering, Al-Nahrain University, Baghdad-Iraq.

[2]Department of Electronic and Communication Engineering, Al-Nahrain University, Baghdad-Iraq.

Received November 21.03.2022 ; Accepted December 30.04.2022

Abstract

Smart surveillance, decision making and automated response systems are trending topics nowadays, especially after performance's growth for a wide range of control systems, which is a normal result of the noticeable evolution in the areas of Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) techniques made dramatically increasing in performance in terms of accuracy and results that have been reached for various systems and algorithms used for different applications in all fields. This development enabled human beings to make a lot of tasks that were done within manual manners much easier than before, not only easier but less time and power-consuming with reducing the numbers of essential operators.

The produced application is Eye-tracking based on CNN; the dataset used for training was about 5568 images, gathered using the camera of laptop based on software designed as shown later. The maximum accuracy achieved by this model 97.57%.

Many tools have been used for various types of necessary tasks which will be parts of the desired application such as: Python 3.7; which was used for building the essential algorithms, KERAS framework; which provided the deep learning algorithms, Visual studio code (VSC) as an Integrated Development Environment (IDE) and Anaconda navigator for downloading the different compatible libraries.

The proposed models were trained on Processor Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz, RAM 16.0 GB, 64-bit operating system, x64-based processor, NVIDIA GeForce GTX 1650 GPU.

Keywords

Eye tracking, Artificial intelligence, Machine learning, Deep learning, CNN, MRFO.

Introduction

Artificial Intelligence (AI)

The capability of distinguishing objects by vision is naturally congenital owned by humans, but for machines, it is an issue. The continuously developing of object detection software techniques, made the performance reach its peak ability using the routine programming procedures, so the need for developing the ways of designing the frameworks has been grown day after day till the time of using Artificial Intelligence, which is the behavior and the properties that the program owns, in order to make the computer simulate the human being's intelligence like the ability of learning, conclusion and react to situations that never been programmed to deal with, that the regular program does not have. AI can be considered as the main idea and the Foundation stone of Machine Learning (ML). ML can be defined as making the machine learn by its own capabilities without the intervention of humans using previous experiments datasets for the specific task and developing itself. This developing in the AI field led to reach the final limits of ML models performance, which meant that there is a necessity for developing this idea for more complex tasks, and that led the way to what it called Deep Learning (DL). The word "Deep" refers to complexity increasing than it was known in ML, which it was a natural result for increasing abilities more than it was before (Chollet, 2021).

Eye-Tracking

Eye-tracking in simple words is tracking the movement of the eyes using different techniques and equipment, such as using DL based on the vision provided by the camera. In the last two decades, the interest in the Eye-tracking technology field dramatically grew up as a tool for different applications such as how his pupils and irises are reacting for a variety of actions, recording the positions that a person is looking at, detecting some nervous illnesses by tracking the movement of the patient eye, etc. An extraordinary amount of information that concerns the user can be provided from the resulted data, in case it was interlocked through data analysis systems, such as biometric identity, age, etc. Large opportunities to develop algorithms and applications were produced by the improvements in this area of Eye-tracking using deep learning (Chollet, 2021) .

Literature Survey

In 1999, K. N. Kim and S. R. Ramakrishnan (K.-N. Kim & Ramakrishna, 1999) presented a method for tracking the gaze, in which feeding eye gaze as an input to a computer interface for eye movements with high efficiency.

In 2002, A. E. Kaufman, A. Bandyopadhyay, and B. D. Shaviv (Kaufman, Bandopadhay, & Shaviv, 1993) proposed that the rotation of the eyeballs will vary the electric field which is measured by the electrodes placed with different locations close to the eye and the resulting estimations are then amplified and analyzed for eye position prediction.

In 2006, A. Poole and L. Ball (Poole & Ball, 2006) proposed that the position that the user is looking at can be predicted using the position of the reflections relative to the pupil.

J. Griffin and A. Ramirez (Griffin & Ramirez, 2018) proposed that applying a dataset to a CNN as a classification method, using different interpolation methods can produce high accuracy and speed.

In 2016, D. Venugopal, A. J, and C. Jyotsna (Venugopal, Amudha, & Jyotsna, 2016) illustrated the usable fields of eye-tracking, its advantages and applications, and practical methods for applications developing using a commercial eye tracker.

In 2017, P. A. Punde, Dr. M. E. Jadhav and Dr. R. R. Manza (Punde, Jadhav, & Manza, 2017) discussed eye-tracking, its wide range of possible implementations, and the fact that it is being used in important fields such as computers that have techniques for human interactions, marketing, medical modern infrastructure, psychology, etc.

In 2020, I. Rakhmatulin and A. T. Duchowski (Rakhmatulin & Duchowski, 2020) presented an analysis about new webcam-based techniques for gaze-tracking with the practical implementation details of the populist methods and an introduction of a new approach that increases the effectiveness of using a DL significantly.

In 2020, A. F. Klaib, N. O. Alsrehin, W. Y. Melhem, H. O. Bashtawi, and A. A. Magableh (Klaib, Alsrehin, Melhem, Bashtawi, & Magableh, 2021) made a study aiming for exploring and reviewing eye-tracking algorithms, methods, techniques, procedures and types with details on the most efficient and effective methods.

In 2021, M. F. Ansari, P. Kasprowski, and M. Obetkal (Jyotsna & Amudha, 2018) presented an article concerned about using a CNN for gaze estimation that can be used without additional special hardware requirements and beneath various circumstances and platforms, this article is the basis of the comparison mentioned later.

Interesting has been increased in this field for the last few years, many methods of image processing have been used, but most of them were not DL-based for tracking and just like any of other different fields many types of research and experiments had their chances for enhancing the performance.

Aim of the Thesis

This thesis aims to Achieve better performance than that could be achieved by predefined models, by adjusting their parameters using an optimization method without depending on

trial-and-error method. Also, it aims to develop a method for eye-tracking dataset gathering that can be used by any user for his own eye in order to train the model.

Theoretical Background

Neural Network (NN)

The main idea of the neural networks was first investigated in the 1950s in toy forms. The first practical neural networks successful application came in 1989, which was done by combining the earlier thoughts of Convolutional Neural Networks (CNN) with the algorithm of backpropagation (Chollet, 2021).Each Neural Network is built up from a number of small units called Neurons. Each one has a number of inputs and only one output. The output of the Neuron is equal to the summation of each input multiplied by a parameter called Weight, along with another term called the Bias. Then it passed through a function called the Activation function. Each one of the weights is dedicated to its input and does not interfere with other inputs' weights. Inputs are fed into the input layer and each Neuron produces an output. The output of the input layer is used as an input to the next layer, which is a hidden layer, then the output of the hidden layer will be used as an input to the next layer, which will be either another hidden layer or the output layer. As the main task of the Neural network is to find the optimal weights, it achieves this task by repeating two processes called Forward propagation and Backpropagation for each example in the training dataset. The learning process requires a number of epochs, while each epoch is achieved when all of the observations contained in the training dataset pass through the forward and backward propagation and the parameters are updated with each epoch (Albon, 2018).

Activation Function

It is a function as any other function has a task of mapping the input into the output (Sultana, Sufian, & Dutta, 2020). An activation function is used in order to make the main function of the network non-linear, as this behavior of those layers enables the CNN model to learn more complex things (Moolayil, Moolayil, & John, 2019; Raschka & Mirjalili, 2019). There is a number of activation functions such as tanh, Relu and Soft max (Moolayil et al., 2019; Rao & McMahan, 2019; Sadoon & Ali, 2021).

Cost Function

It is a mathematical term that contributes with the learning procedure, which is the metric of how far the expected output to the actual output from the output layer of the neural network (P. J. W. m. l. Kim, neural networks & intelligence, 2017).

Optimizer

An optimizer is a mathematical algorithm that depends on derivatives, chain rules and partial derivatives to produce the amount of change in the loss function by making small changes in the values of nodes weights. The loss function variation is used to determine the correct direction of the required change in the amounts of weights (Moolayil et al., 2019).

Convolutional Neural Network (CNN)

It is, in general, a specialized and popular type of neural network for data processing that has proved to be very effective at computer vision. It is evident from its name that this type utilizes one time at least a process called the convolution, which is a particular type of linear mathematical operation, in place of the general multiplication of the matrix (Goodfellow, Bengio, & Courville, 2016). CNN has the ability to extract features from the input data using the particular types of layers that it contains. Three main types of layers are used for building CNN architectures: Convolutional Layer, Pooling Layer and Fully-Connected Layer (Shukla & Fricklas, 2018).

DNN models

Different models were developed with different structures and parameters, but all of the layers consisted of are divided into a small number of types such as convolution layer, pooling layer, batch normalization, dropout, fully connected, etc. Here, three models were used, and they are:

ALEXNET

This model was introduced by Alex Krizhevsky in 2012. It won the ImageNet LSVRC-2012 competition. It is a famous model that the paper contained the proposal was cited about 30000 times in 2019 (Aggarwal, 2018; Zou, Shi, Guo, & Ye, 2019).

VGG16

For a while, many efforts have been made for working on improving vision understanding by the computers using DNN. This model has proved to be a significant development in this field that led the way for a number of followed innovations was paved by it. It is a CNN model proposed in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition” by Karen Simonyan and Andrew Zisserman at the University of Oxford, its idea was proposed at first in 2013, while the implementation was introduced in 2014 at the ILSVRC ImageNet Challenge (Chollet, 2021).

Model1

This model is not actually a standard model like the previous models. It was inspired by the architectures of both LENET5 and ALEXNET and designed in a trial-and-error manner. It contains four convolutional layers and two fully connected layers. A 5-by-5 kernel size was chosen for each convolutional layer with some strides equal to 1. Convolutional and max-pooling layers used" Re LU" as an activation function. 2-by-2 kernel size and a stride of 1 was used for Max pooling layers. The last fully-connected layer used the activation function" Soft max". The architecture of the model is shown in Figure 6. It also shows the chosen number of filters, the shapes of inputs and outputs for each layer and the chosen number of neurons of the fully connected layers. It was titled as" Model1" to be distinguished from other models.

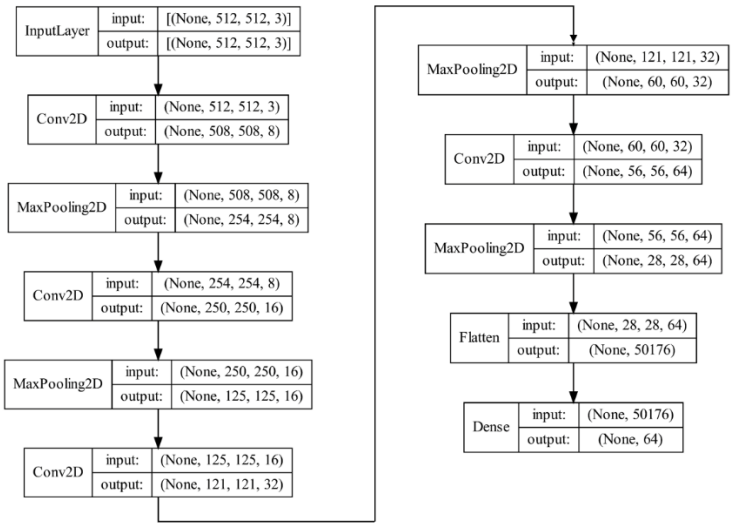


Figure 1 Model1 architecture

ILSVRC

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was an annual competition that evaluated algorithms for image classification (and object detection) at a large scale.

Confusion matrix

The confusion matrix can be simply defined as a square matrix which reports the number of the True positive (TP), False positive (FP), True negative (TN) and False negative (FN) predictions of a model. This helps to give a full picture for the performance of the model.

KERAS

Keras is a popular library written in Python used for building, training, and evaluating NNs. It is a high-level library, running on top of the machine learning platform TensorFlow with using other libraries like Theano as an "engine". The advantage of Keras is that the designer can focus on network design and training, leaving the specifics of the tensor operations to other libraries. As it provides great performance for industry, it takes part in large companies all over the world, such as YouTube and NASA.

MRFO

Designing a model is a procedure that can end with a wide range of performance, but choosing a specific set of values for its parameters may achieve the best performance. Tuning the model's parameters using trial-and-error is an exhausting procedure. A large number of real-world optimization problems are becoming challenging because of the large number of variables, objective functions and nonlinear constraints. At the same time, using traditional approaches such as numerical methods for optimization have been less effective as the objective functions may contain multiple peaks. While meta-heuristic
algorithms are powerful tools that are used for handling challenging optimization issues and their popularity are continuously growing. Alternatively, MRFO has been used for obtaining accurate parameters for the model that maximizes its accuracy for its robustness, ease of implementation and simplicity. Manta rays are creatures that belong to a family of rays that belong to the Manta genus. These creatures have pectoral fleshy fins that are similar to wings projecting as cephalic fins that lie from the head reaching to the large mouth. They achieve different and successful strategies of foraging for feeding on plankton, including the chain, somersault and cyclone foraging techniques. MRFO has three techniques of foraging:

1- CHAIN FORAGING: Chain foraging techniques is the first stage of foraging. It begins when not less than 50 manta rays' line up, each one behind another, which forms an orderly line. Smaller males are piggybacked upon female's swim on top of their backs in order to match the female's pectoral fins beats, as shown in figure (2) with the black color. Consequently, the missed plankton by the previous manta ray will be caught up by the manta rays behind him. Using this method of cooperating can funnel the biggest possible number of plankton into their gills, which improves their food consumption.

2- CYCLONE FORAGING (CF): When a group of Manta rays finds in deep water a spot containing plankton, they will head close to the spot of the food with spiralling after achieving a long forage chain, as shown in figure (2) with magenta color. Along with this technique, each Manta ray will then move in the direction of food ahead individually. In essence, each one of them follows not only the food ahead but also travels in a spiralling manner to the direction of food.

3- SOMERSAULT FORAGING (SF): Every single one of Manta rays will try to swim back and forth from one place to another around the pivot and somersault to a new location. So, Manta rays will update their locations continuously around the best-obtained location so far, as shown in figure (2) with blue color.

The three strategies can be shown together in the figure (2) (Alturki, Omotoso, Al-Shamma'a, Farh, & Alsharabi, 2020; Rai & Tyagi, 2013; Zhao, Zhang, & Wang, 2020).

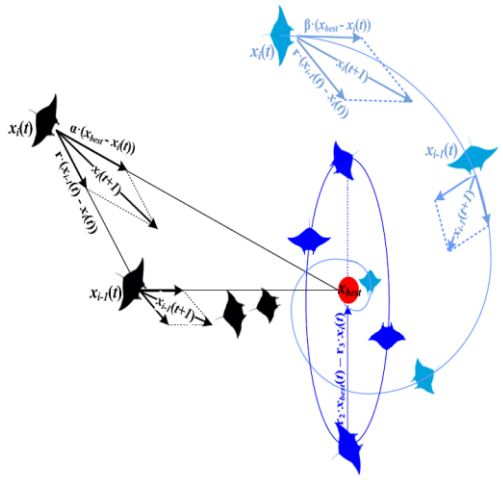


Figure 2 Manta ray's three foraging strategies

Automatic Systems Preparations and Implementing Dataset

This application differs from other applications in terms of the dataset, as collecting the dataset takes other types of processes. As the model is based on the computer vision which is implemented by the video frames captured by the camera. The main part is reading the frames consequently from the camera video input to our model as shown in Figure (3).

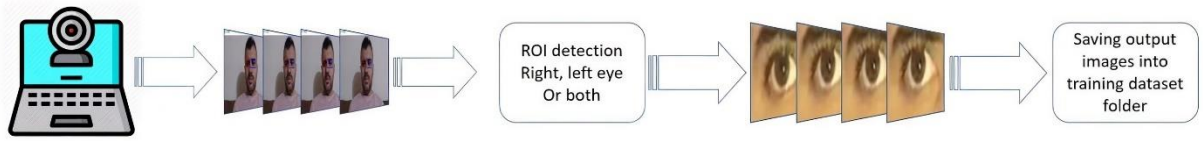


Figure 3 Reading video from the camera, detect and extract ROI(s)

For making it easy to handle, the procedure needs to be simple for using it with different applications. To understand it in a simple way, a flowchart of the algorithm parts was drawn and explained. The main part that concerns with reading frames consequently from the

video captured by the camera to the model is shown in figure(4a). The RoI is detected using an open-source library was used called" shape-predictor-68-face-landmarks", which makes a mask on the human being face using 68 points as shown in figure(5a), then choosing the required points to extract the RoI. Then each or both of the eyes have been detected from the resulted coordinates of the surrounding points of the eye(s). Various algorithms might be using right, left eye, or even both eyes. The (RoI) regions can be titled as" R RoI" or" LRoI" respectively. The desired accuracy can be achieved by choosing number of spots which lying on the screen that the user needs to be interacted with or according to the requirements of the user application. Here a number of 16 spots was chosen as shown in figure(5b).

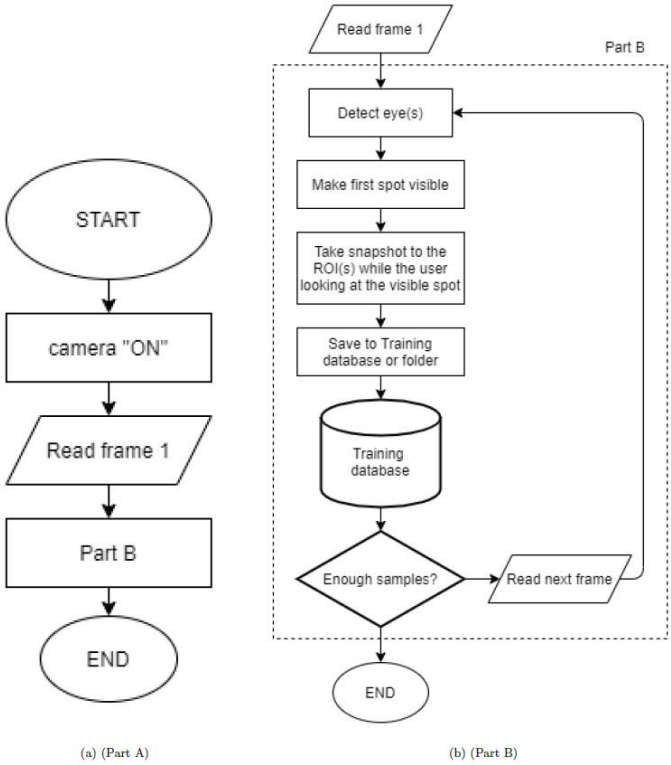


Figure 4 Flowchart of data gathering process

These spots on the screen in the figure(5b) only one of them will be visible through collecting the training data in practice. However, they are visible here together only for explaining the process. Dataset collecting process Firstly, the training dataset collecting code is opened and executed. Then, the training pad and the first spot was visible. At this step, the user must focus on the visible spot and a specific period of time for visibility is assigned for each spot. When the period ends; the current spot loses its visibility while the next one gains it. For instance, in this experiment, a period of time equal to (5) seconds of time was chosen and in the middle of it a snapshot will be captured for the RoI(s) only and

not for the entire face. Finally, the snapshot was saved into the training dataset with a name formed by gathering more than one piece as shown in the equation (1).

Name Format = CONCATENATE (RoI, Spot, Random) (1)

where:

A-RoI character: which may be either “L” for the left eye or “R” for the right eye.

B-Spot number: it ranges from (1) to maximum chosen number of spots (16 in this experiment).

C-Random number: in order to not making any similar names, it is generated and added to the name.

(a) Face landmark points map

(b) Eye Tracking training spots, no. of spots=16

Figure 5 Data-collecting tools

This procedure is repeated till gathering the required amount of data by user. This procedure is shown in both of figure(4a) and figure(4b). They are connected to each other but they were separated into two parts to be clearer for understanding. The gathered dataset was about (5568) for training. After that, the generated data were arranged in folders for each spot as a type of data preparation. Data augmentation is a technique used for improving model performance by increasing the amount of the training data. This results in reducing overfitting if it happens. Data amount increasing is done by generating new data from the original ones using different procedures [18, 12, 19, 21]. Various methods have been used for augmentation. First, in order to take the movement of the head randomly into account, a rotation from (-5) to (+5) degree was used. Second, to give the system more flexibility for the distance between the face and the camera while working, random zooming was used with a range of (0 - 0.1). The amounts of the dataset after augmentation are summarized in table (1).

Training the model

It can be briefly said that the model is fed with the gathered data and as a result, this procedure provides the output model. Choosing the model is done through two scenarios. The first scenario, predefined models were used and trained by the dataset. While the second scenario is like the first scenario but with initializing at first then modifying the parameters of the model using MRFO, which will depend on the previous results and try to calibrate the parameters till reaching the best performance.

Table 1 Summary of numbers of images after augmentation

	Number of images
Original	5568
Exposing	5493
Rotating	5561
Zooming	5568
Total	22190

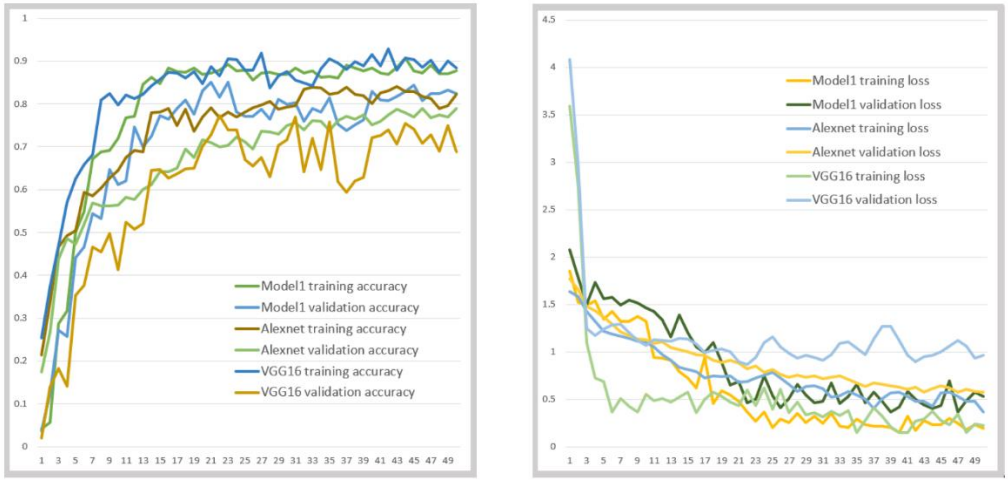
Eye-tracker implementation

Mouse-controlling based on Eye-tracking is used as an example of controlling and decision making for this model, the system was built up. Then, this extracted RoI is fed into the model in order to make the decision which represents the number of the spot that the eye is focusing on, which is a pure classification process. After that, the coordinates of the spot are called using the number of spots predicted by the model and then move the mouse into the specific coordinates after importing a library called" pyautogui" using the commands written below:

import pyautogui

pyautogui.move To (x,y)

where x and y are the target coordinates, these coordinates are calculated with different aspects based on the application requirements like being uniformly distributed or not, being resized as the screen size change and even for a specific part of the screen.



(a) Accuracy (b) Loss

Figure 6 Accuracy and Loss of Eye-tracking models before applying MRFO

Table 2 Accuracies and losses for Eye-tracking before applying MRFO

	Accuracy		Loss	
	Training	Validation	Training	Validation
Model1	0.906857581	0.850544982	0.145768775	0.367458447
ALEXNET	0.840279536	0.790014684	0.36465	0.573872527
VGG16	0.928395576	0.772082459	0.146414363	0.866128207

Systems implementation

The next step of implementing the system is training the model. For this application, three models were used, ALEXNET, VGG16, and Model1.

First scenario

The three models have been fed with the training dataset and trained. After completing the trainings, the final results are tabled in the table (2) and plotted as shown in figure (6). It is clear that the best accuracy (92.83%) was achieved by VGG16 and the lowest loss was about (14.57%) achieved by the model named (Model1). Although it seems that these results may be satisfying. However, it is very important to remind that this model was based on 16 spots only. In order to generalize the whole operation for even larger number of spots, it need to be more accurate, which is done using MRFO as shown in the next section.

Enhancing the models using MRFO (second scenario)

After achieving some good results, it is necessary to show that better results can be achieved in order to be a pathfinder for other researchers to increase the number of spots, add some more features like eye-blinking detection, etc., as will be shown in the future work. After that, the second scenario was done, the results implemented by the parameters values of the model, which can be seen in tables (3, 4, and 5). Applying the final parameters to the models achieves the results shown in table (6), then plotting these results on the same graph together as shown in figure (7).

Table 3 Model1 for Eye-Tracking feasible ranges of hyperparameter values, and the final values

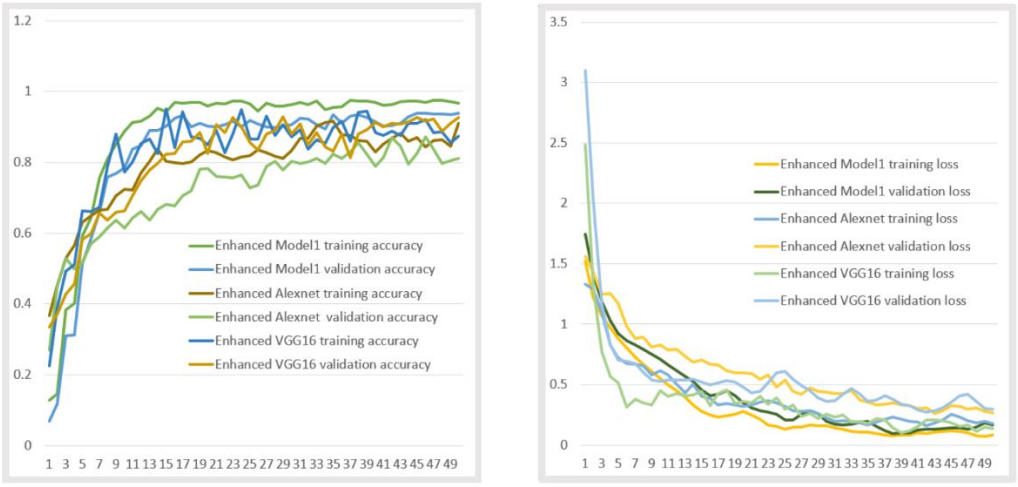
Model1 parameters	Range	choosed value
No. of filters in C1	8 - 128	12
No. of filters in C2	16 - 128	24
No. of filters in C3	32 - 128	44
No. of filters in C4	32 - 128	67
Kernal size in C1	7 - 15	7
Kernal size in C2	5 - 11	7
Kernal size in C3	5 - 11	9
Kernal size in C4	3 - 11	3
No. of neurons in FC1	64 - 256	115
No. of neurons in FC2	32 - 256	39
Dropout rate	0.0 - 0.5	0.4

Table 4 ALEXNET for Eye-Tracking feasible ranges of hyperparameter values, and the final values

ALEXNET parameters	Range	choosed value
No. of filters in C1	32 - 128	78
No. of filters in C2 and C5	64 - 128	96
No. of filters in C3 and C4	64 - 128	82
Kernal size in C1	5 - 15	5
Kernal size in C2	5 - 11	5
Kernal size in C3, C4 and C5	3 - 11	3
No. of neurons in FC1	64 - 256	232
No. of neurons in FC1	64 - 256	142
Dropout rate	0.0 - 0.5	0.3

Table 5 VGG16 for Eye-Tracking feasible ranges of hyperparameter values, and the final values

VGG16 parameters	Range	choosed value
No. of filters in C1	32 - 128	72
No. of filters in C2	32 - 128	73
No. of filters in C3	32 - 128	36
No. of filters in C4	32 - 128	124
Kernal size in C1	3 - 13	3
Kernal size in C2	3 - 13	7
Kernal size in C3	3 - 13	5
Kernal size in C4	3 - 13	5
No. of neurons in FC1	64 - 256	210
No. of neurons in FC2	32 - 256	112
Dropout rate	0.0 - 0.5	0.4



(a) Accuracy (b) Loss

Figure 7 Accuracy and Loss of Eye-tracking models after applying MRFO

Table 6 Accuracies and losses for Eye-tracking after applying MRFO

	Accuracy		Loss	
	Training	Validation	Training	Validation
Model1	0.975777779	0.939039489	0.073739739	0.094580183
ALEXNET	0.916821957	0.871173358	0.16209537	0.258937709
VGG16	0.951637846	0.92856612	0.100888013	0.26971397

Testing the system

For studying the performance of the system, it was tested by applying the system to a number of samples and producing the confusion matrix. For generating the confusion

matrix, a number of samples about (640) were applied to the system, and the result was observed and drawn as a confusion matrix as shown in figure (8).

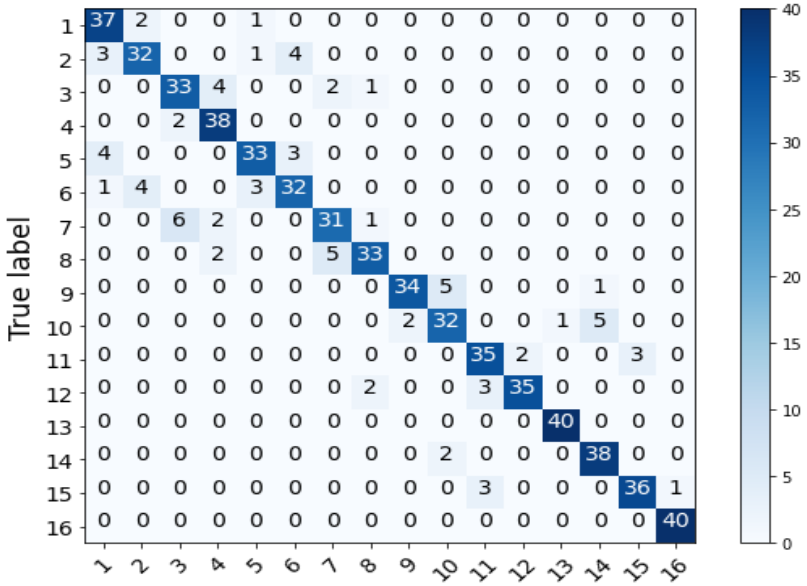


Figure 8 Eye-Tracking model confusion matrix

Table 7 Summary of confusion matrix outputs and model performance measuring metrics for the best model for Eye-Tracking

Spot	TP	TN	FP	FN	Sensitivity	specificity	Accuracy	Precision
1	37	592	8	3	0.925	0.9867	0.9828	0.93
2	32	594	6	8	0.8	0.99	0.9781	0.80
3	33	592	8	7	0.825	0.9867	0.9766	0.82
4	38	592	8	2	0.95	0.9867	0.9844	0.95
5	33	588	5	7	0.825	0.9915	0.9813	0.82
6	32	593	7	8	0.8	0.9883	0.9766	0.80
7	31	588	7	9	0.775	0.9882	0.975	0.78
8	33	594	4	7	0.825	0.9933	0.9828	0.82
9	34	596	2	6	0.85	0.9966	0.9875	0.85
10	32	590	7	8	0.8	0.9883	0.9766	0.80
11	35	583	6	5	0.875	0.9898	0.9828	0.88
12	35	583	2	5	0.875	0.9966	0.9891	0.88
13	40	585	1	0	1	0.9983	0.9984	1
14	38	588	6	2	0.95	0.9899	0.9875	0.95
15	36	598	3	4	0.9	0.995	0.9891	0.9
16	40	597	1	0	1	0.9983	0.9984	1

The result was written in tabular form as shown in table (7).
Where:
TP: Represents the number of samples that the eye is looking on this spot and the predicted is the same spot.
TN: The number of samples that the eye focusing on other spots and the predicted spot is not this spot.
FP: Number of samples that the eye is focusing on other spots, while the predicted spot is this spot.
FN: The number of samples that the eye is focusing on this spot, while the predicted spot is not.
Sensitivity, specificity, Accuracy, and Precision discussed earlier in 2.7.
The overall accuracy equals 87.34% which was calculated as shown below:
Accuracy = (TP+TN)/(TP+FP+TN+FN) = 559/640 = 0.8734

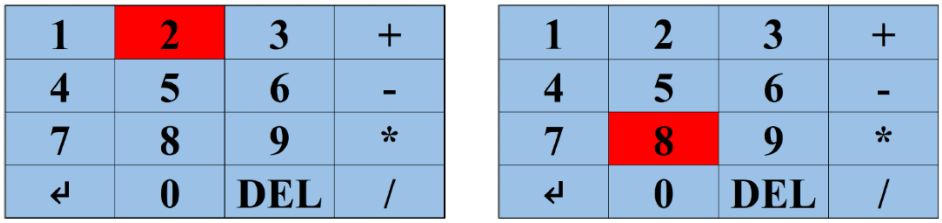
Implementing Applications Based on The Eye-Tracking System

Implementing the system was within a basic application represented by choosing a specific folder using the movement of the eye (with recalling the window), as shown in figure (9). This example can be used with a calculator with big size as shown in figure (10).

Comparison with other previous works

Many previous works tried to attempt Eye-tracking system with various equipment and algorithms, one of them used the same technique with same level of simplicity but also has some different parameters such as the number of spots, number of training dataset images, etc. The comparison is obvious in the table (8).

Figure 9 An example of controlling the mouse using Eye-tracker



(a) Focusing on number "2" (b) Focusing on number "8"

Figure 10 An example of controlling the mouse with a calculated using Eye-tracker

Table 8 Details table of Eye tracking performance comparison

Reference	No. of spots	Training dataset	Model	Accuracy %
M. F. Ansari, P. Kasprowski, and M. Obetkal [10]	20	6000	ARCH-L	88.55
			ARCH-R	82.30
This work	16	22190	E-Model1	97.57

Conclusion and future work

Conclusion

Completing this thesis was accompanied by some conclusions drawn from the results especially and from all of the thesis generally. VGG16 is a powerful model but requires higher device specifications than other models do. Applying MRFO as a hybrid-parameter tuner has a significant influence on the model's performance. The enhanced model is dedicated to work for the specific application that it was enhanced with. Increasing the dataset requires increasing the model's ability and making the model more general with different circumstances. Training datasets for many applications need to be augmented accurately depending on the nature of the application and not randomly. An Eye-tracking system was designed for disabled people based on enhanced predefined CNN models and achieved an accuracy (97.57%). A data collector algorithm was built for collecting data for the user by himself.

Future work

For Eye-Tracking, a larger number of spots can be used in order to make more use for the system and increase its abilities by adding some more features, such as mouse-clicking using eye blinking, making the pc be controlled by its user's eyes only, etc. Also, its performance enhancement of the system can be done using a larger dataset. MRFO abilities

can be increased by making the objective function has more parameters such as the number of parameters, number of convolution layers, etc.

References

Aggarwal, C. C. (2018). An introduction to neural networks. In Neural Networks and Deep Learning (pp. 1-52): Springer.

Albon, C. (2018). Machine learning with python cookbook: Practical solutions from preprocessing to deep learning: " O'Reilly Media, Inc.".

Alturki, F. A., Omotoso, H. O., Al-Shamma'a, A. A., Farh, H. M., & Alsharabi, K. J. I. A. (2020). Novel Manta Rays Foraging Optimization Algorithm Based Optimal Control for Grid-Connected PV Energy System. 8, 187276-187290.

Chollet, F. (2021). Deep learning with Python: Simon and Schuster.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning: MIT press.

Griffin, J., & Ramirez, A. (2018). Convolutional Neural Networks for Eye Tracking Algorithm. In: Stanford University.

Jyotsna, C., & Amudha, J. (2018). Eye gaze as an indicator for stress level analysis in students. Paper presented at the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

Kaufman, A. E., Bandopadhay, A., & Shaviv, B. D. (1993). An eye tracking computer user interface. Paper presented at the Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium.

Kim, K.-N., & Ramakrishna, R. (1999). Vision-based eye-gaze tracking for human computer interface. Paper presented at the IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028).

Kim, P. J. W. m. l., neural networks, & intelligence, a. (2017). Matlab deep learning. 130(21).

Klaib, A. F., Alsrehin, N. O., Melhem, W. Y., Bashtawi, H. O., & Magableh, A. A. J. E. S. w. A. (2021). Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and Internet of Things technologies. 166, 114037.

Moolayil, J., Moolayil, J., & John, S. (2019). Learn Keras for deep neural networks: Springer.

Poole, A., & Ball, L. J. (2006). Eye tracking in HCI and usability research. In Encyclopedia of human computer interaction (pp. 211-219): IGI global.

Punde, P. A., Jadhav, M. E., & Manza, R. R. (2017). A study of eye tracking technology and its applications. Paper presented at the 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM).

Rai, D., & Tyagi, K. J. A. S. S. E. N. (2013). Bio-inspired optimization techniques: a critical comparative study. 38(4), 1-7.

Rakhmatulin, I., & Duchowski, A. T. J. P. C. S. (2020). Deep neural networks for low-cost eye tracking. 176, 685-694.

Rao, D., & McMahan, B. (2019). Natural language processing with PyTorch: build intelligent language applications using deep learning: " O'Reilly Media, Inc.".

Raschka, S., & Mirjalili, V. (2019). Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2: Packt Publishing Ltd.

Sadoon, T. A., & Ali, M. H. J. I. J. A. A. S. I. (2021). Deep learning model for glioma, meningioma and pituitary classification. 2252(8814), 8814.

Shukla, N., & Fricklas, K. (2018). Machine learning with TensorFlow: Manning Greenwich.

Sultana, F., Sufian, A., & Dutta, P. J. I. c. i. p. b. a. (2020). A review of object detection models based on convolutional neural network. 1-16.

Venugopal, D., Amudha, J., & Jyotsna, C. (2016). Developing an application using eye tracker. Paper presented at the 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT).

Zhao, W., Zhang, Z., & Wang, L. J. E. A. o. A. I. (2020). Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. 87, 103300.

Zou, Z., Shi, Z., Guo, Y., & Ye, J. J. a. p. a. (2019). Object detection in 20 years: A survey.