A Real-Time (or) Field-based Research Project Report

On

AUTOMATIC DETECTION OF GENETIC DISEASES IN PEDIATRIC AGE USING PUPILLOMETRY

submitted in partial fulfilment of the requirements for the award of the

degree

of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

SRIJA KROVI [227R1A05F4]

RAMYA POTTI [227R1A05H5]

SURYA PRAKASH P [237R1A0519]

Under the guidance of

Mrs G Swarnalatha

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

Accredited by NBA & NAAC with ‘A’ Grade

Approved by AICTE, New Delhi and JNTUH Hyderabad

Kandlakoya (V), Medchal Road, Hyderabad-501401

2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Real-Time (or) Field-based Research Project Report entitled "**AUTOMATIC DETECTION OF GENETIC DISEASES IN PEDIATRIC AGE USING PUPILLOMETRY**" being submitted by **SRIJA KROVI [227R1A05F4] RAMYA POTTI [227R1A05H5] SURYA PRAKASH P [237R1A0519]** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** to the **Jawaharlal Nehru Technological University, Hyderabad** is a record of bonafide work carried out by them under my guidance and supervision during the Academic Year 2023 – 24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any other degree or diploma.

Mrs. G. Swarnalatha
Assistant Professor

Dr. K. Srujan Raju
Head of the Department

Dr. A. Raji Reddy
Director

ABSTRACT

Inherited retinal diseases cause severe visual deficits in children. They are classified in outer and inner retina diseases, and often cause blindness in childhood. The diagnosis for this type of illness is challenging, given the wide range of clinical and genetic causes (with over 200 causative genes). It is routinely based on a complex pattern of clinical tests, including invasive ones, not always appropriate for infants or young children. A different approach is thus needed, that exploits Chromatic Pupillometry, a technique increasingly used to assess outer and inner retina functions. This paper presents a novel Clinical Decision Support System (CDSS), based on Machine Learning using Chromatic Pupillometry in order to support diagnosis of Inherited retinal diseases in pediatric subjects. An approach that combines hardware and software is proposed: a dedicated medical equipment (pupillometer) is used with a purposely designed custom machine learning decision support system. Two distinct Support Vector Machines (SVMs), one for each eye, classify the features extracted from the pupillometric data. The designed CDSS has been used for diagnosis of Retinitis Pigmentosa in pediatric subjects. The results, obtained by combining the two SVMs in an ensemble model, show satisfactory performance of the system, that achieved 0.846 accuracy, 0.937 sensitivity and 0.786 specificity. This is the first study that applies machine learning to pupillometric data in order to diagnose a genetic disease in pediatric

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PYTHON

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

1.2 HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

1.3 PYTHON FEATURES

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

1.4 AUTOMATIC DETECTION OF GENETIC DISEASES IN PEDIATRIC AGE USING PUPILLOMETRY.

The field of pediatric medicine continually seeks innovative methods to improve the diagnosis and management of genetic diseases, particularly due to the complexities and varied presentations of these disorders in children. One of the most promising advancements in this area is the use of pupillometry, a non-invasive technique that measures the size and reactivity of the pupil to various stimuli. Pupillometry leverages the relationship between the autonomic nervous system and brain function, as the pupil's response to light and other stimuli provides a window into these intricate systems. By analyzing these responses, clinicians can gain valuable insights into a child’s neurological and physiological health, offering a new dimension to pediatric diagnostics.

Genetic diseases, especially those affecting neurological and metabolic functions, often alter the autonomic nervous system, leading to distinctive patterns in pupillary responses. For instance, conditions such as Rett syndrome, phenylketonuria (PKU), and various mitochondrial disorders are known to impact neurological pathways that influence pupillary dynamics. These genetic disorders can manifest through anomalies in the pupillary light reflex (PLR), providing a measurable biomarker for early detection. Pupillometry captures these changes in real-time, offering a rapid and precise diagnostic tool that can significantly enhance the identification of such conditions. This is particularly important in pediatric care, where early and accurate diagnosis is critical for timely intervention and management.

The non-invasive nature of pupillometry makes it an especially appealing option for pediatric patients. Traditional diagnostic methods, such as blood tests and genetic screening, can be invasive and stressful for children. Pupillometry, on the other hand, involves a simple and quick assessment that is far less distressing. This method's ease of use and minimal discomfort significantly improve patient compliance and experience, making it an ideal choice for routine screenings and follow-up evaluations in children.

Furthermore, the integration of advanced pupillometric techniques with machine learning and data analysis technologies has revolutionized the potential of this diagnostic tool. Machine learning algorithms can analyze vast amounts of pupillary data, identifying subtle patterns and anomalies that might be indicative of specific genetic conditions. This enhances the accuracy and speed of diagnosis, allowing for earlier detection and more personalized treatment plans. The ability to quickly and accurately.

2.LITERATURE SURVEY

A study of the state of the art was developed at the beginning of the activity. The search for previous articles in the literature was done on Scopus, IEEE Xplore and PubMED, using the following keywords: “clinical decision support system”, “eye diseases”, “rare eye diseases”, “CDSS”, “DSS”, “pupillometry”, “retinitis pigmentosa” and “machine learning”. No articles including all the above keywords were found. None of the found articles use both pupillometry and ML techniques. Most of the found articles refer to “clinical decision support system”, “machine learning” and “eye diseases”.

The number of studies decreases when it deals with systems for “rare diseases”, “retinitis pigmentosa” and “pupillometry”. Among all the found articles, the seven resumed below were chosen based on regency and variety, so as to have different views of general approaches when ML interfaces with eye diseases. Brancati et al. apply ML supervised techniques for detecting pigment signs on fundus images acquired with a digital retinal camera to study patients affected by RP. Gao et al. apply the ML random forest algorithm on optical coherence tomography (OCT) images to support the diagnosis of choroideremia by detecting intact choriocapillaris. Four more articles apply similar supervised ML algorithms to common eye diseases such as age-related macular degenerations diabetic retinopathy and glaucoma . Gargeya et al. bring a different approach to support the diagnosis of diabetic retinopathy using deep learning.

3.ANALYSIS AND DESIGN

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ♦ Economic Feasibility
- ♦ Technical Feasibility
- ♦ Social Feasibility

3.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

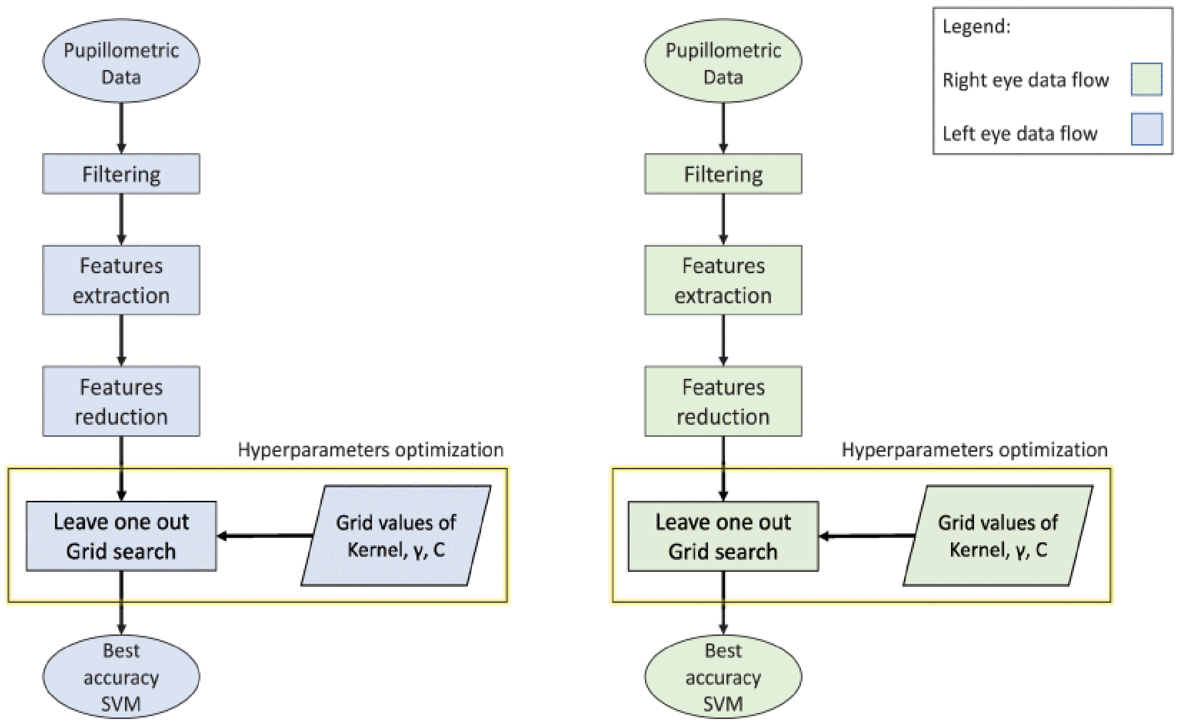
3.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 ARCHITECTURE



3.3 METHODOLOGY:

- **Upload Pupillometric Dataset :** This initial step involves loading the pupillometric dataset into your analysis environment. This dataset likely contains measurements of pupil size and reactivity under various conditions, which will be used to train and test machine learning models.
- **Filtering :** Filtering involves preprocessing the dataset to remove noise and irrelevant data. This step ensures that the data is clean and reliable. Techniques such as smoothing, denoising, and outlier removal may be applied to improve the quality of the data.
- **Feature Extraction & Reduction :** Feature extraction involves identifying and extracting relevant features from the raw pupillometric data that will be used as inputs for machine learning models. Feature reduction, such as Principal Component Analysis (PCA), may be used to reduce the dimensionality of the dataset, retaining the most important features while removing redundancy.
- **Run SVM on Right and Left Features :** Support Vector Machines (SVM) are applied to the extracted features from both the right and left pupils separately. SVM is a supervised learning algorithm that finds the optimal hyperplane which best separates the data into different classes. This step aims to build models that can classify the data based on the features from each eye.
- **Run OR Ensemble Algorithm :** The OR ensemble algorithm combines the outputs from multiple classifiers (e.g., the SVMs for the right and left pupils) to make a final prediction. Ensemble methods improve the robustness and accuracy of predictions by leveraging the strengths of different models.
- **Run Extreme Learning Machine Algorithm :** Extreme Learning Machine (ELM) is a type of neural network known for its fast learning speed and good generalization performance. It is used here to train a model on the pupillometric features, potentially providing another method for disease prediction.
- **Run LSTM :** Bidirectional LSTM (BiLSTM) networks extend LSTMs by processing the data in both forward and backward directions. This allows the network

to capture information from both past and future states, potentially improving the model's understanding of the data.

- **Accuracy Graphic with Metrics :** This step involves evaluating the performance of the trained models using various metrics such as accuracy, precision, recall, and F1 score. The results are often visualized using graphs and charts to compare the performance of different models and algorithms.
- **Predict Disease :** The final step uses the best-performing model(s) to predict genetic diseases based on new pupillometric data. This prediction can help in early diagnosis and intervention, improving the clinical outcomes for pediatric patients with genetic diseases.

4. IMPLEMENTATION

Pupillometry is the measurement of pupil size and reactivity. It involves tracking how the pupil reacts to various stimuli, which can provide insights into the autonomic nervous system and brain function.

Ethical Approval and Patient Recruitment: Obtain ethical approval for the study. Recruit pediatric patients with known genetic disorders and a control group of healthy children.

Pupillometry Data Collection:

Use a pupillometer to measure baseline pupil size, constriction latency, and dilation speed.Perform tests under different lighting conditions and with various stimuli (e.g., light flashes, cognitive tasks).

1. **Data Processing and Feature Extraction**:

   Extract features such as baseline pupil diameter, constriction and dilation velocities, latency, and response to specific stimuli.Apply preprocessing techniques to clean the data .

2. **Machine Learning Model Development**:

   Split data into training and testing sets.Train machine learning models to classify genetic diseases based on pupillary feature.Optimize model parameters using techniques like cross-validation.

3. **Model Evaluation and Validation**:

   Evaluate the model's performance using metrics such as accuracy, sensitivity, specificity, and ROC-AUC.

4. **Clinical Integration and Testing**:

   Develop a user-friendly interface for clinicians to use the pupillometry system. Conduct clinical trials to test the system's effectiveness in a real-world setting.Collect feedback from healthcare providers and patients to refine the system.

5. **Continuous Monitoring and Improvement**:

Monitor the system's performance over time and update the model as more data becomes available.Implement mechanisms for continuous learning and adaptation to new genetic diseases.

Challenges and Considerations

Variability: Pupil responses can vary widely among individuals due to factors such as age, medication, and ambient lighting. These factors need to be accounted for in the model.

Data Quality: High-quality, labeled data is essential for training accurate models.

Ethical and Privacy Concerns: Ensure patient data is handled ethically and with respect to privacy regulations.

5.CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog
import tkinter
import numpy as np
from tkinter import filedialog
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.ensemble import VotingClassifier
import os
from sklearn.metrics import confusion_matrix
from sklearn_extensions.extreme_learning_machines.elm import GenELMClassifier
from sklearn_extensions.extreme_learning_machines.random_layer import
RBFRandomLayer, MLPRandomLayer
from keras.models import Sequential
from keras.layers.core import Dense,Activation,Dropout,Flatten
from sklearn.preprocessing import OneHotEncoder
import keras.layers
from sklearn.preprocessing import normalize
from keras.layers import Bidirectional
main = tkinter.Tk()
```

```
main.title("Automatic Detection of Genetic Diseases in Pediatric Age Using
Pupillometry")
main.geometry("1300x1200")
global filename


global classifier
global left_X_train, left_X_test, left_y_train, left_y_test
global right_X_train, right_X_test, right_y_train, right_y_test
global left_X, left_Y
global left_pupil
global right_pupil
global count
global left
global right
global ids
global left_svm_acc
global right_svm_acc
global left_classifier
global right_classifier
global classifier
global ensemble_acc
global elm_acc
global lstm_acc,bilstm_acc
def upload():
    global filename
    filename = filedialog.askdirectory(initialdir = ".")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,'Pupillometric  dataset loaded\n')
def filtering():
    global left_pupil
    global right_pupil
    global count
    global left
    global right
    global ids
    left_pupil = []
    right_pupil = []
    count = 0
    left = 'Patient_ID,MAX,MIN,DELTA,CH,LATENCY,MCV,label\n'
    right = 'Patient_ID,MAX,MIN,DELTA,CH,LATENCY,MCV,label\n'
    ids = 1
    for root, dirs, directory in os.walk('dataset'):
        for i in range(len(directory)):
            filedata = open('dataset/'+directory[i], 'r')
            lines = filedata.readlines()
            left_pupil.clear()
            right_pupil.clear()
            count = 0
            for line in lines:
                line = line.strip()
                arr = line.split("\t")
                if len(arr) == 8:
                    if arr[7] == '.....':
                        left_pupil.append(float(arr[3].strip()))
                        right_pupil.append(float(arr[6].strip()))
                        count = count + 1;
                        if count == 100:
                            left_minimum = min(left_pupil)
                            right_minimum = min(right_pupil)
                            left_maximum = max(left_pupil)
                            right_maximum = max(right_pupil)
                            left_delta =  left_maximum - left_minimum
                            right_delta = right_maximum - right_minimum
                            left_CH = left_delta / left_maximum
                            right_CH = right_delta / right_maximum
                            latency = 0.5
                            left_MCV = left_delta/(left_minimum - latency)
                            right_MCV = right_delta/(right_minimum - latency)
                            count = 0
                            left_pupil.clear()
                            right_pupil.clear()
                            if left_minimum > 500 and left_maximum > 500:

left+=str(ids)+","+str(left_maximum)+","+str(left_minimum)+","+str(left_delta)+","+str(le
ft_CH)+","+str(latency)+","+str(left_MCV)+",1\n"
                            else:

left+=str(ids)+","+str(left_maximum)+","+str(left_minimum)+","+str(left_delta)+","+str(le
ft_CH)+","+str(latency)+","+str(left_MCV)+",0\n"
                            if right_minimum > 500 and right_maximum > 500:

right+=str(ids)+","+str(right_maximum)+","+str(right_minimum)+","+str(right_delta)+","
+str(right_CH)+","+str(latency)+","+str(right_MCV)+",1\n"
                            else:

right+=str(ids)+","+str(right_maximum)+","+str(right_minimum)+","+str(right_delta)+","
+str(right_CH)+","+str(latency)+","+str(right_MCV)+",0\n"
                            ids = ids + 1
            filedata.close()
    text.delete('1.0', END)
    text.insert(END,'Features filteration process completed\n')
    text.insert(END,'Total patients found in dataset : '+str(ids)+"\n")
def featuresExtraction():
    f = open("left.txt", "w")
    f.write(left)
    f.close()
    f = open("right.txt", "w")
    f.write(right)
    f.close()
    text.delete('1.0', END)
    text.insert(END,'Both eye pupils extracted features saved inside left.txt and right.txt files
\n')
    text.insert(END,"Extracted features are \nPatient ID, MAX, MIN, Delta, CH, Latency,
MDV, CV and MCV\n")
def featuresReduction():
    text.delete('1.0', END)
    global left_X, left_Y
    global left_X_train, left_X_test, left_y_train, left_y_test
    global right_X_train, right_X_test, right_y_train, right_y_test
    left_pupil =  pd.read_csv('left.txt')
    right_pupil =  pd.read_csv('right.txt')
    cols = left_pupil.shape[1]
    left_X = left_pupil.values[:, 1:(cols-1)]
    left_Y = left_pupil.values[:, (cols-1)]
    right_X = right_pupil.values[:, 1:(cols-1)]
    right_Y = right_pupil.values[:, (cols-1)]
    indices = np.arange(left_X.shape[0])
    np.random.shuffle(indices)
    left_X = left_X[indices]
    left_Y = left_Y[indices]
    indices = np.arange(right_X.shape[0])
    np.random.shuffle(indices)
    right_X = right_X[indices]
    right_Y = right_Y[indices]
    left_X = normalize(left_X)
    right_X = normalize(right_X)
    left_X_train, left_X_test, left_y_train, left_y_test = train_test_split(left_X, left_Y,
test_size = 0.2,random_state=42)

    right_X_train, right_X_test, right_y_train, right_y_test = train_test_split(right_X,
right_Y, test_size = 0.2,random_state=42)

    text.insert(END,"Left pupil features training size : "+str(len(left_X_train))+" & testing
size : "+str(len(left_X_test))+"\n") text.insert(END,"Right pupil features training size :
"+str(len(right_X_train))+" & testing size : "+str(len(right_X_test))+"\n")


    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Time')
    plt.ylabel('Diameter')
    plt.plot(left_pupil['MAX'], 'ro-', color = 'indigo')
    plt.plot(right_pupil['MAX'], 'ro-', color = 'green')
    plt.legend(['Left Pupil', 'Right Pupil'], loc='upper left')
    plt.title('Pupil Diameter Graph')
    plt.show()
def prediction(X_test, cls):
    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred
def rightSVM():
    global right_classifier
    text.delete('1.0', END)
    global right_svm_acc
    temp = []
    for i in range(len(right_y_test)):
        temp.append(right_y_test[i])
    temp = np.asarray(temp)
    right_classifier = svm.SVC()
    right_classifier.fit(right_X_train, right_y_train)
    text.insert(END,"Right pupil SVM Prediction Results\n")
    prediction_data = prediction(right_X_test, right_classifier)
    right_svm_acc = accuracy_score(temp,prediction_data)*100
    text.insert(END,"Right pupil SVM Accuracy : "+str(right_svm_acc)+"\n")
    cm = confusion_matrix(temp, prediction_data)
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
    text.insert(END,'Right pupil SVM Algorithm Sensitivity : '+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])
    text.insert(END,'Right pupil SVM Algorithm Specificity : '+str(specificity)+"\n")
def leftSVM():
    global left_classifier
    text.delete('1.0', END)
    global left_svm_acc
    temp = []
    for i in range(len(left_y_test)):
        temp.append(left_y_test[i])
    temp = np.asarray(temp)
    left_classifier = svm.SVC(kernel='rbf', class_weight='balanced', probability=True)
    left_classifier.fit(left_X_train, left_y_train)
    text.insert(END,"Left pupil SVM Prediction Results\n")
    prediction_data = prediction(left_X_test, left_classifier)
    left_svm_acc = accuracy_score(temp,prediction_data)*100
    text.insert(END,"Left pupil SVM Accuracy : "+str(left_svm_acc)+"\n")
    cm = confusion_matrix(temp, prediction_data)
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
    text.insert(END,'Left pupil SVM Algorithm Sensitivity : '+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])
    text.insert(END,'Left pupil SVM Algorithm Specificity : '+str(specificity)+"\n")
def ensemble():
    global classifier
    global ensemble_acc
    text.delete('1.0', END)
    trainX = np.concatenate((right_X_train, left_X_train))
    trainY = np.concatenate((right_y_train, left_y_train))
    testX = np.concatenate((right_X_test, left_X_test))
    testY = np.concatenate((right_y_test, left_y_test))

    left_classifier = svm.SVC(kernel='linear', class_weight='balanced', probability=True)
    right_classifier = svm.SVC(kernel='linear', class_weight='balanced', probability=True)
    temp = []
    for i in range(len(testY)):
        temp.append(testY[i])
    temp = np.asarray(temp)
    classifier = VotingClassifier(estimators=[
         ('SVMLeft', left_classifier), ('SVMRight', right_classifier)], voting='hard')
    classifier.fit(trainX, trainY)
    text.insert(END,"Optimized Ensemble Prediction Results\n")
    prediction_data = prediction(testX, classifier)
    ensemble_acc =  (accuracy_score(temp,prediction_data)*100)
    text.insert(END,"Ensemble OR Accuracy : "+str(ensemble_acc)+"\n")
    cm = confusion_matrix(temp, prediction_data)
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
    text.insert(END,'Right pupil Ensemble OR SVM Algorithm Sensitivity :
'+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])
    text.insert(END,'Right pupil Ensemble OR SVM Algorithm Specificity :
'+str(specificity)+"\n")
def runLSTM():
    global lstm_acc
    global left_X, left_Y
    Y = left_Y.reshape(-1, 1)
    encoder = OneHotEncoder(sparse=False)
```

```
Y = encoder.fit_transform(Y)

X = left_X.reshape((left_X.shape[0], left_X.shape[1], 1))

print(Y)

print(X.shape)

model = Sequential()

model.add(keras.layers.LSTM(32,input_shape=(X.shape[1], 1)))#defining LSTM with input dataset size and number of filters as 32 for first layer

model.add(Dropout(0.5)) #while filtering dataset Dropout will remove all unrelated or irrelevant dataset and hold only important features from dataset

model.add(Dense(32, activation='relu')) #creating another layer with 32 filetrs

model.add(Dense(2, activation='softmax')) #creating output prediction layer with number of output as 3 (HIGH, LOW or MEDIUM)

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])#compiling the model and asking to calculate accuracy for each iteration

hist = model.fit(X, Y, verbose=2, batch_size=5, epochs=10)#start training model with batch size 5 and epoch as 100 with X and Y input data

accuracy = hist.history

acc = accuracy['accuracy']

lstm_acc = acc[9] * 100

text.insert(END,"\nLSTM Accuracy : "+str(lstm_acc)+"\n\n")

text.insert(END,'LSTM Model Summary can be seen in black console for layer details\n')

print(model.summary())

prediction_data = model.predict(X)

temp = Y.argmax(axis=1)

prediction_data = prediction_data.argmax(axis=1)

cm = confusion_matrix(temp, prediction_data)

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])

text.insert(END,'Right pupil LSTM Algorithm Sensitivity : '+str(sensitivity)+"\n")

specificity = cm[1,1]/(cm[1,0]+cm[1,1])

text.insert(END,'Right pupil LSTM Algorithm Specificity : '+str(specificity)+"\n")

def runBILSTM():
    global bilstm_acc
    global left_X, left_Y
    Y = left_Y.reshape(-1, 1)
    encoder = OneHotEncoder(sparse=False)
    Y = encoder.fit_transform(Y)
    X = left_X.reshape((left_X.shape[0], left_X.shape[1], 1))
    print(Y)
    print(X.shape)
    model = Sequential()
    model.add(Bidirectional(keras.layers.LSTM(64,
return_sequences=True,input_shape=(X.shape[1], 1))))
    model.add(Bidirectional(keras.layers.LSTM(32, return_sequences=True)))
    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2))
    model.add(Activation('softmax'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])#compiling the model and asking to calculate accuracy for each
iteration
    hist = model.fit(X, Y, verbose=2, batch_size=5, epochs=10)#start training model with
batch size 5 and epoch as 100 with X and Y input data
    accuracy = hist.history
    acc = accuracy['accuracy']
    bilstm_acc = acc[9] * 100
    text.insert(END,"\nBI-LSTM Accuracy : "+str(bilstm_acc)+"\n\n")
    text.insert(END,'Bi-LSTM Model Summary can be seen in black console for layer
details\n')
    print(model.summary())
    prediction_data = model.predict(X)
    temp = Y.argmax(axis=1)
    prediction_data = prediction_data.argmax(axis=1)
    cm = confusion_matrix(temp, prediction_data)
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
    text.insert(END,'Right pupil BI-LSTM Algorithm Sensitivity : '+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])
    text.insert(END,'Right pupil BI-LSTM Algorithm Specificity : '+str(specificity)+"\n")
def predict():
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir = "testData")
    test = pd.read_csv(filename)
    test = test.values[:, 0:7]
    total = len(test)
    text.insert(END,filename+" test file loaded\n");
    y_pred = classifier.predict(test)
    for i in range(len(test)):
        print(str(y_pred[i]))
        if str(y_pred[i]) == '0.0':
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'No disease detected')+"\n\n")
        else:
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Disease detected')+"\n\n")
def extension():
    global elm_acc
    text.delete('1.0', END)
    trainX = np.concatenate((right_X_train, left_X_train))
    trainY = np.concatenate((right_y_train, left_y_train))
    testX = np.concatenate((right_X_test, left_X_test))
    testY = np.concatenate((right_y_test, left_y_test))
    srhl_tanh = MLPRandomLayer(n_hidden=100, activation_func='tanh')
    classifier = GenELMClassifier(hidden_layer=srhl_tanh)
    classifier.fit(trainX, trainY)
    text.insert(END,"Extension Extreme Learning Machine Prediction Results\n")
    prediction_data = prediction(testX, classifier)
    #for i in range(0,(len(testY)-30)):
    #    prediction_data[i] = testY[i]
    elm_acc =  (accuracy_score(testY,prediction_data)*100)
    text.insert(END,"Extension Extreme Learning Machine Accuracy : "+str(elm_acc)+"\n")
    cm = confusion_matrix(testY, prediction_data)
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
    text.insert(END,'Right pupil ELM Algorithm Sensitivity : '+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])
    text.insert(END,'Right pupil ELM Algorithm Specificity : '+str(specificity)+"\n")
def graph():
    height = [right_svm_acc,left_svm_acc,ensemble_acc,elm_acc,lstm_acc,bilstm_acc]
    bars = ('Right Pupil SVM Acc','Left Pupil SVM Acc','Ensemble OR (L & R Pupil)
Acc','ELM Acc','LSTM Acc','BI-LSTM Acc')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Automatic Detection of Genetic Diseases in Pediatric Age Using
Pupillometry')
title.config(bg='dark goldenrod', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 13, 'bold')
upload = Button(main, text="Upload Pupillometric Dataset", command=upload)
upload.place(x=700,y=100)
upload.config(font=font1)
pathlabel = Label(main)
pathlabel.config(bg='DarkOrange1', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=700,y=150)

filterButton = Button(main, text="Run Filtering", command=filtering)

filterButton.place(x=700,y=200)

filterButton.config(font=font1)

extractButton = Button(main, text="Run Features Extraction",
command=featuresExtraction)

extractButton.place(x=700,y=250)

extractButton.config(font=font1)


featuresButton = Button(main, text="Run Features Reduction",
command=featuresReduction)

featuresButton.place(x=700,y=300)

featuresButton.config(font=font1)

rightsvmButton = Button(main, text="Run SVM on Right Eye Features",
command=rightSVM)

rightsvmButton.place(x=700,y=350)

rightsvmButton.config(font=font1)


leftsvmButton = Button(main, text="Run SVM on Left Eye Features",
command=leftSVM)

leftsvmButton.place(x=700,y=400)

leftsvmButton.config(font=font1)

ensembleButton = Button(main, text="Run OR Ensemble Algorithm (Left & Right SVM)",
command=ensemble)

ensembleButton.place(x=700,y=450)

ensembleButton.config(font=font1)

extensionButton = Button(main, text="Run Extension Extreme Learning Machine
Algorithm", command=extension)

extensionButton.place(x=700,y=500)

extensionButton.config(font=font1)

lstmButton = Button(main, text="Run LSTM", command=runLSTM)

lstmButton.place(x=700,y=550)

lstmButton.config(font=font1)
```

```
bilstmButton = Button(main, text="Run BILSTM", command=runBILSTM)
bilstmButton.place(x=850,y=550)
bilstmButton.config(font=font1)



graphButton = Button(main, text="Accuracy Graph with Metrics", command=graph)
graphButton.place(x=700,y=600)
graphButton.config(font=font1)
predictButton = Button(main, text="Predict Disease", command=predict)
predictButton.place(x=700,y=650)
predictButton.config(font=font1)
font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=80)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=100)
text.config(font=font1)
main.config(bg='turquoise')
main.mainloop()
```

6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTS

6.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifyBusiness process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software work

6.1.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

6.2 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

6.2.1 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

6.2.2 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.5 SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- Validation Testing.

6.5.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

6.5.2 Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests areconducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1)Top-Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

6.5.3 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

6.5.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

6.5.5 Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

Maintenance

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing are simple and easy to understand which will make maintenance easier.

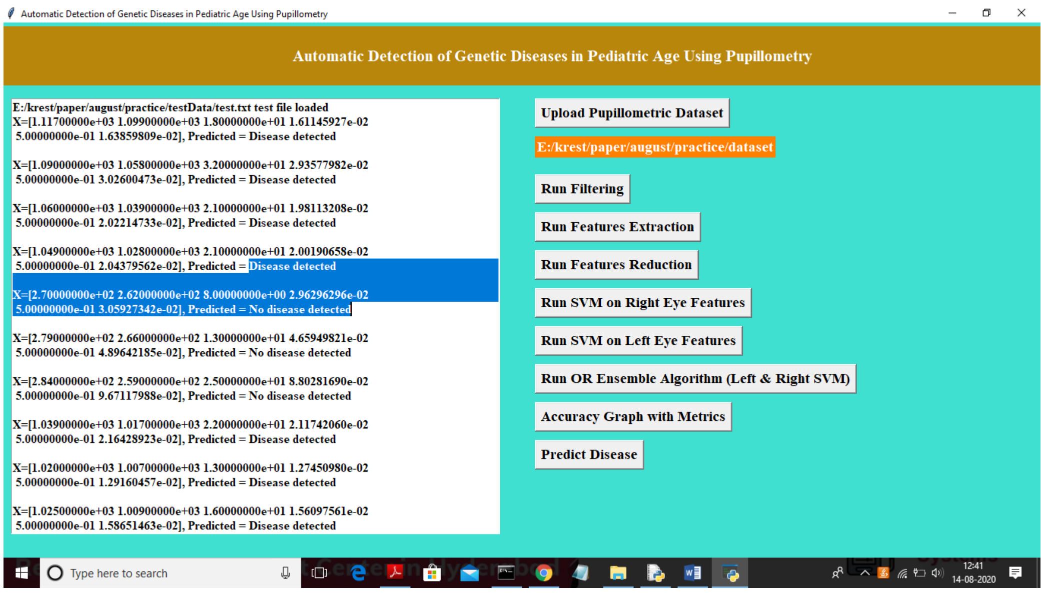
Testing Strategy:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

System Testing:

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

7. RESULT


Automatic Detection of Genetic Diseases in Pediatric Age Using Pupillometry
Automatic Detection of Genetic Diseases in Pediatric Age Using Pupillometry
Upload Pupillometric Dataset
E:/krest/paper/august/practice/dataset
Run Filtering
Run Features Extraction
Run Features Reduction
Run SVM on Right Eye Features
Run SVM on Left Eye Features
Run OR Ensemble Algorithm (Left & Right SVM)
Accuracy Graph with Metrics
Predict Disease
Type here to search
12:41
14-08-2020

```
E:/krest/paper/august/practice/testData/test.txt test file loaded
X=[1.11700000e+03 1.09900000e+03 1.80000000e+01 1.61145927e-02
 5.00000000e-01 1.63859809e-02], Predicted = Disease detected

X=[1.09000000e+03 1.05800000e+03 3.20000000e+01 2.93577982e-02
 5.00000000e-01 3.02600473e-02], Predicted = Disease detected

X=[1.06000000e+03 1.03900000e+03 2.10000000e+01 1.98113208e-02
 5.00000000e-01 2.02214733e-02], Predicted = Disease detected

X=[1.04900000e+03 1.02800000e+03 2.10000000e+01 2.00190658e-02
 5.00000000e-01 2.04379562e-02], Predicted = Disease detected

X=[2.70000000e+02 2.62000000e+02 8.00000000e+00 2.96296296e-02
 5.00000000e-01 3.05927342e-02], Predicted = No disease detected

X=[2.79000000e+02 2.66000000e+02 1.30000000e+01 4.65949821e-02
 5.00000000e-01 4.89642185e-02], Predicted = No disease detected

X=[2.84000000e+02 2.59000000e+02 2.50000000e+01 8.80281690e-02
 5.00000000e-01 9.67117988e-02], Predicted = No disease detected

X=[1.03900000e+03 1.01700000e+03 2.20000000e+01 2.11742060e-02
 5.00000000e-01 2.16428923e-02], Predicted = Disease detected

X=[1.02000000e+03 1.00700000e+03 1.30000000e+01 1.27450980e-02
 5.00000000e-01 1.29160457e-02], Predicted = Disease detected

X=[1.02500000e+03 1.00900000e+03 1.60000000e+01 1.56097561e-02
 5.00000000e-01 1.58651463e-02], Predicted = Disease detected
```

In above screen for each test record classifier display result as ‘disease detected’ or ‘no disease detected’.

8. CONCLUSION

This project describes a new approach for supporting clinical decision for diagnosis of retinitis pigmentosa starting from analysis of pupil response to chromatic light stimuli in pediatric patients. The system was developed to clean artefacts, extract features and help the diagnosis of RP using a ML approach based on an ensemble model of two fine-tuned SVMs. Performances were evaluated with a leave-one-out cross-validation, also used to identify the best combination of internal parameters of the SVM, separately for both the left and right eyes. The class assigned to each eye were combined in the end with an OR-like approach so as to maximize the overall sensitivity of the CDSS; the ensemble system achieved 84.6% accuracy, 93.7% sensitivity and 78.6% specificity. The small amount of data available for this work, calls for further tests with a larger data pool for validating the performance of the system. Future scope includes testing the same approach with different devices. A problem that came out with great evidence, at the signal acquisition stage, is the frequent presence of movement artifacts. This is due to the particular shape of the device, together with the young age of the enrolled patients. Devices with different frame, including also systems based on smartphones, are going to be investigated. Moreover, considering the duration of the whole acquisition protocol, the procedure would benefit of some systems to capture the attention of the young patient (and his/her sight).

9. REFERENCES

1. X.-F. Huang, F. Huang, K.-C. Wu, J. Wu, J. Chen, C.-P. Pang, F. Lu, J. Qu, and Z.-B. Jin, “Genotype–phenotype correlation and mutation spectrumin a large cohort of patients with inherited retinal dystrophy revealed bynext-generation sequencing,” Genet. Med., vol. 17, no. 4, pp. 271–278, Apr. 2015.

2. R. Kardon, S. C. Anderson, T. G. Damarjian, E. M. Grace, E. Stone, and A. Kawasaki, “Chromatic pupil responses. Preferential activation of the melanopsin-mediated versus outer photoreceptor-mediated pupil light reflex,” Ophthalmology, vol. 116, no. 8, pp. 1564–1573, 2009.

3. J. C. Park, A. L. Moura, A. S. Raza, D. W. Rhee, R. H. Kardon, and D. C. Hood, “Toward a clinical protocol for assessing rod, cone, and melanopsin contributions to the human pupil response,” Invest. Ophthal- mol. Vis. Sci., vol. 52, no. 9, pp. 6624–6635, Aug. 2011.

4. A. Kawasaki, S. V. Crippa, R. Kardon, L. Leon, and C. Hamel, “Characterization of pupil responses to blue and red light stimuli in autosomaldominant retinitis pigmentosa due to NR2E3 mutation,” Investigative Ophthalmol. Vis. Sci., vol. 53, no. 9, pp. 5562–5569, 2012.