

Interactive, visual learning-based tool for hearing impaired children to improve language skills.

22_23-J 18

DRAFT- Final Report

B.W.E.K. Senarathna

IT19142838

B.Sc. (Hons) Degree in Information Technology Specialized
Data Science

Department of Information Technology


Sri Lanka Institute of Information Technology

Sri Lanka

May 2023

DECLARATION

I declare that this is my own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
B.W.E.K. Senarathnna	IT19142838	

Signature of the Supervisor
(Dr. Janaka Wijekoon)

Date

.....

.....

ABSTRACT

Hearing impairment is mostly seen in many countries in the world and it also can be easily cured if correct attention and relevant medication is done at the early stages of childhood.

Keywords – Explorative learning, YOLOv5, Image Processing

ACKNOWLEDGEMENT

First.....

TABLE OF CONTENTS

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	ix
1. INTRODUCTION	1
1.1 General Introduction	1
1.2 Background literature	2
1.2 Research Gap	10
2. RESEARCH PROBLEM	13
3. RESEARCH OBJECTIVES	15
3.1 Main Objectives	15
3.2 Specific Objectives	15
4. METHODOLOGY	17
4.1 Materials and methods	17
4.1.1 Problem statement	18
4.1.2 Component System Architecture (Solution Design)	19
4.1.3 Data Acquisition and processing	20
4.1.5 Identification	
4.1.6 Design Diagrams	41
4.2 Commercialization aspects of the product	43
5. Testing & Implementation	44
5.1 Implementation	44
5.1.1 Data Preprocessing	44
5.2 Testing	57
5.2.1 Test Plan and Test Strategy.....	57
5.2.2 Test Case Design	58

6. RESULTS AND DISCUSSIONS	64
6.1 Results	64

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
CEHIC	Centre for Education of Hearing Impaired Children
CNN	Convolutional Neural Network
ML	Machine Learning
RPN	Region Proposal Network
FCN	Fully Convoluted Neural Network
YOLO	You only look once

1. INTRODUCTION

When considering the world population amount of people are suffering from hearing impairment. This hearing impairment can be mainly divided in to several types of hearing loss. Mild hearing loss, moderate hearing loss, and profound hearing loss. Out of the mentioned categories the mild and moderate hearing impairment can be cured at certain level if treatments and audio therapy were started at early childhood. There many approaches taken to address the hearing impaired children and from them mostly the Sign language is promoted in Sri Lanka to hearing impaired children. Many research has proved the exposure to sounds and audio therapy can enhance the cognitive and hearing skills. The process of exposing hearing impaired children to sounds frequently makes them normalize to normal sounds. The teaching of Sign language makes hearing impaired children refrain from speaking, so it is not recommended to learn sign language unless the child is fully hearing impaired.

Our research mainly focus on the process of making hearing impaired children listen and speak. We proposed system consist of a implementation where child can interact with the his/her environment. A process where sign language is demotivated, and the speaking and hearing skills are prioritized. The child learns words at early childhood by interacting with the environment by physically engaging with the objects around him. This concept is used hear to make the child engage with his environment and learn new words with their phonetical pronunciations.

1.2 Background literature

1.2 Research Gap

2. RESEARCH PROBLEM

There are many research and implementations to enhance the learning experience of hearing-impaired children. The base of most of this research are titled to improve the sign language skills of hearing-impaired children. There are many research which show the need of improving the hearing impaired Childs' speaking and hearing abilities by early exposure. The stimuli which trigger

the audible nerve repetitively can make the child an audible person as a normal person. The parents need to speak with the children at early childhood. Apart from the parents speaking the child needs additional support to repetitively track and improve his learning skills. A proper mechanism for this is needed to monitor considering the factors of child knowledge growth, word exposure, word limitations, and the phonetical pronunciations. A proper platform is need for the integration of all the above factors.

3. RESEARCH OBJECTIVES

3.1 Main Objectives

The main objective is to make the child interact with the surroundings and learn new words and improve the vocabulary. Implementing a mechanism to make the child learn new word from the surrounding environment.

Specific objectives

1. Creating an approach to capture objects in the environment to the learning environment.

The objects around the child environment should be captured and should be used in the learning process of the child and make the child user understand the underlying concepts related to objects surround him.

2. Creating an algorithm to make the child's learning experience less complicated.

The exposure of objects to the child needs to done on a limited basis of words based on his learning and the complexity of exposing a the child to various objects need to be minized.

3. Creating a mechanism to introduce new objects to the learning curve of the child.

If a new object which is not captured via the implemented algorithms in the mobile, these new words need to be added into the application. With specific mechanism to identify these new objects

4. METHODOLOGY

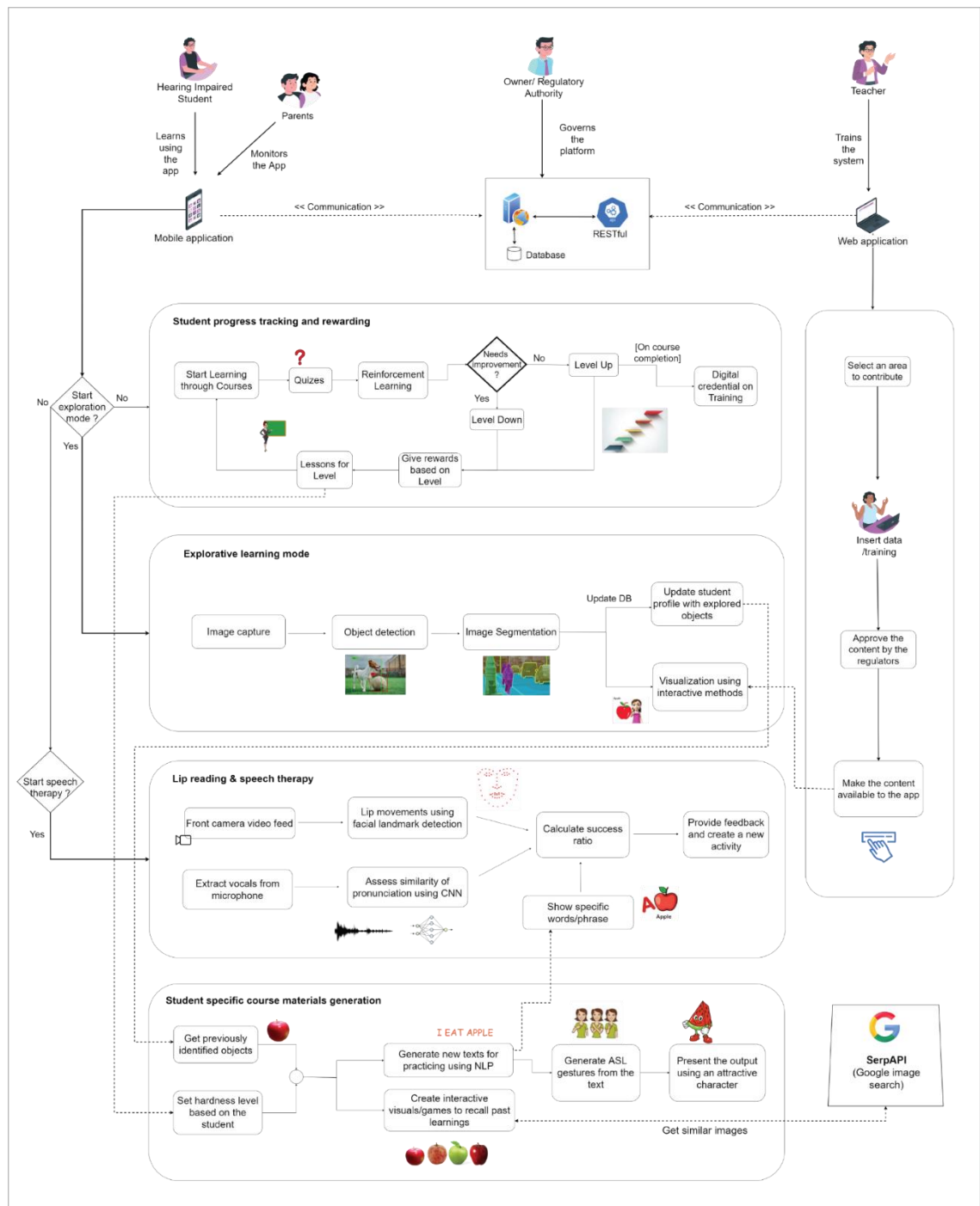
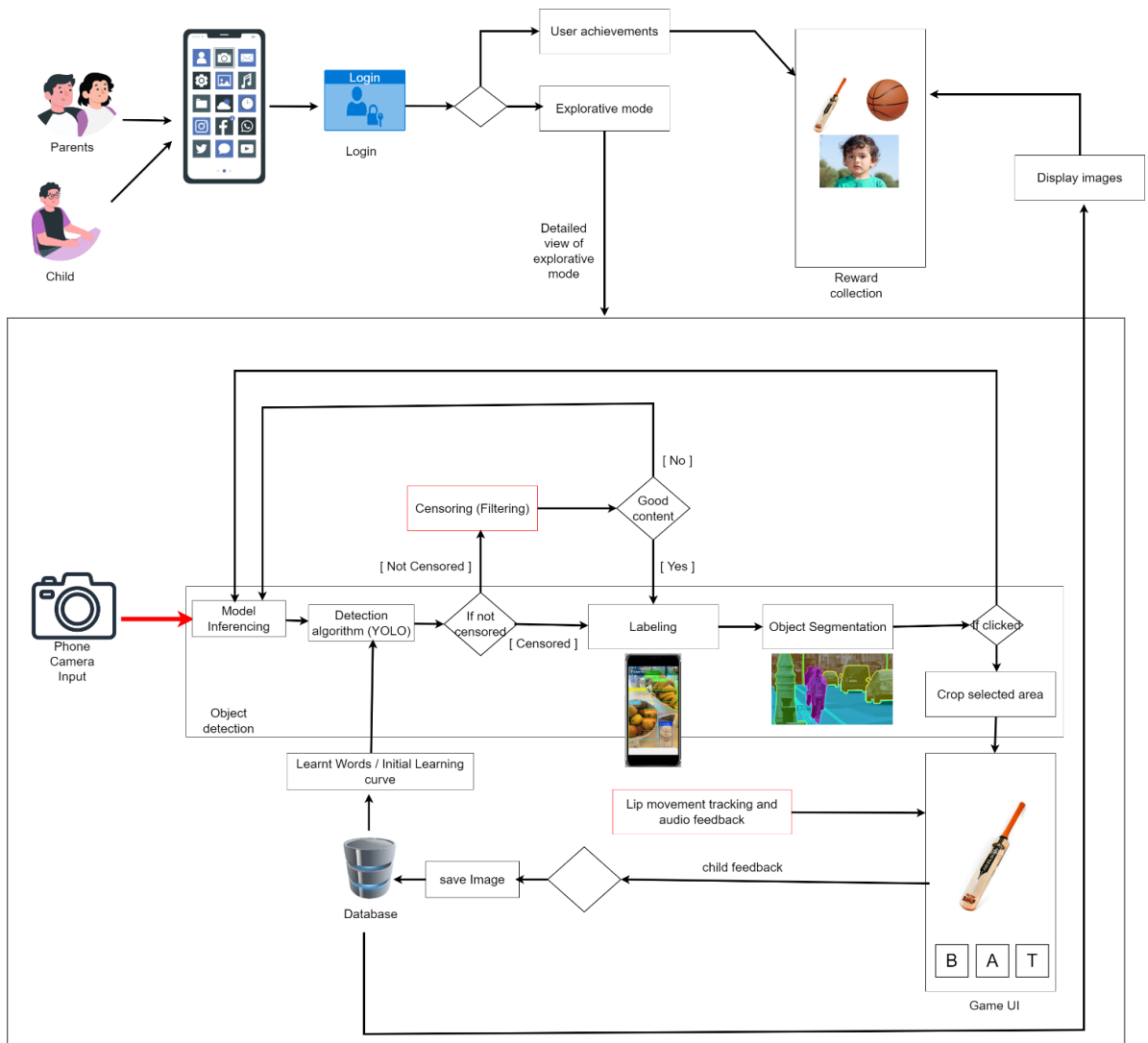


Figure 1 System Overview Diagram

The implemented system consists of several components targeting different linguistic skill which a hearing-impaired child should learn at early childhood. The full system can be divided into several components named, object explorative mode, learning material generation, lip reading and speech therapy, progress tracking and level determination. The object explorative mode focuses on increasing the learning experience of the child through his/her environment. The content generation targets the generation of new content based on the learnt content through object exploration and pre created content. The lip tracking and speech therapy component targets on the phonetic aspect of learning in the child. The combined application gives a mobile based solution where the child can enhance linguistic and cognitive skills in terms of using a mobile application under the supervision of a teacher/parent.



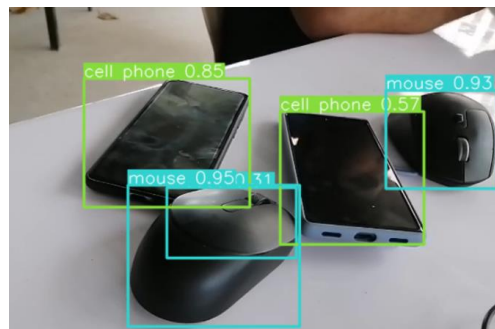
Component overview diagram

The above diagram shows the in detailed view of the object explorative mode. The environment around the child is captured using a camera. The captured objects are subjected to a labeling in the next layer using object detection algorithm (YOLOv5) and based on that the child is directed to a game where he/she can learn the spelling and the wording of the word.

A specific object detection algorithm is used to track the objects captured by the camera. The object detection was tested based to several object detection algorithms and YOLOv5 showed the highest accuracies



The amount of objects captured in a single frame can be in different numbers. Different objects from the same objects class can be detected bounding boxes are drawn upon them. Multiple objects from different class also can be detected. When considering all these objects are tracked and bounding boxes are drawn it gets complicated to a frame. This results in complications in selecting the most ideal object in the frame.



Object bounding boxes overlapping each other

Several implementations were tested to minimize the number of objects tracked within a captured frame from the environment. Initially all the objects with IOU (Intersection over union)

value greater than 0.5 was taken into consideration as a base value. Intersection over union is primarily used to track and find the most suitable bounding box for a relevant object.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

This results in having multiple bounding boxes from and overlapping of each other. When 2 bounding boxes overlap each other a specific algorithm was developed to get the best bounding box with the most precise bounding box

```
def get_iou(box1, box2):
    y11, x11, y21, x21 = box1
    y12, x12, y22, x22 = box2

    yi1 = max(y11, y12)
    xi1 = max(x11, x12)
    yi2 = min(y21, y22)
    xi2 = min(x21, x22)
    inter_area = max(((xi2 - xi1) * (yi2 - yi1)), 0)
    # Calculate the Union area by using Formula: Union(A,B) = A + B - Inter(A,B)
    box1_area = (x21 - x11) * (y21 - y11)
    box2_area = (x22 - x12) * (y22 - y12)
    print("box1_area: ", box1_area)
    union_area = box1_area + box2_area - inter_area
    # compute the IoU
    iou = inter_area / union_area
    return iou
```

Method to get the IOU of 2 different class objects

Figure shows implementation to add the IOU value to 2 different objects in a situation where bounding boxes are overlapping and get the bounding box with the highest IOU value. This results in getting the object



Final resulted model to filter the objects based on IOU values.

Imported libraries

```
import cv2
import torch
from tracker import *
import numpy as np
from shapely.geometry import Polygon
```

Open CV, Pytorch

```
cv2.namedWindow('FRAME')
cv2.setMouseCallback('FRAME', POINTS)

tracker = Tracker()
countingRegion = [(151,159),(933,173),(933,437), (127,413)]

objectCountSet = set()
while True:
    ret,frame=cap.read()
    frame=cv2.resize(frame,(1020,500))

    cv2.polylines(frame, [np.array(countingRegion,np.int32)], True, (0,255,0), 2)
    # Make detections
    results = model(frame)
    # a = results.pandas().xyxy[0]
    # print(a)

    # creating a tracker list
    trackerList = []
    iterateCounter, overCrowdCounter = 0, 0
    for index, row in results.pandas().xyxy[0].iterrows():
        x1 = int(row['xmin'])
        y1 = int(row['ymin'])
        x2 = int(row['xmax'])
        y2 = int(row['ymax'])
        objectName = str(row['name'])

        # print("trackerlist: ",trackerList)
        # tracking overlapping bounding boxes

        if len(trackerList) == 0:
            trackerList.append([x1,y1,x2,y2])
        else:
            for i in trackerList:
                value = get_iou([x1,y1,x2,y2], i)
                if value > 0.8:
                    print("the iou value: ",value)
                    trackerList.remove(i)
            else:
                trackerList.append([x1,y1,x2,y2])

        # creating box id
        boxIds = tracker.update(trackerList)
        # print("boxids: ",boxIds)
        print("Object name: ",objectName)
```



```

print("Object name: ",objectName)
print("points: [",x1,y1,x2,y2,"]")
for id in boxIds:
    x,y,w,h,id = id
    cv2.rectangle(frame, (x,y), (w,h), (0,0,0),2)
    cv2.putText(frame, str(row['name']), (x,y), cv2.FONT_HERSHEY_PLAIN,1, (255,255,255))
    count=cv2.pointPolygonTest(np.array(countingRegion,np.int32), (int(w), int(h)), False)
    if count > 0:
        objectCountSet.add(id)
    # print(len(objectCountSet))
    # cv2.putText(frame, str(objectCountSet), (78,52), cv2.FONT_HERSHEY_PLAIN,2, (255,0,0))
if len(objectCountSet) > 5:
    cv2.putText(frame, "Too many objects", (78,52), cv2.FONT_HERSHEY_PLAIN,2, (255,0,0))
    print("Too many objects")
    overCrowdCounter += 1
    if overCrowdCounter > 5:
        cv2.putText(frame, "Over Crowded too long", (78,90), cv2.FONT_HERSHEY_PLAIN,2, (255,0,0))
        print("Over Crowded too long")

objectCountSet = set()
# cv2.rectangle(frame, (x1,y1), (x2,y2), (0,0,0),2)
# cv2.putText(frame, objectName, (x1,y1), cv2.FONT_HERSHEY_PLAIN,1, (255,255,255))

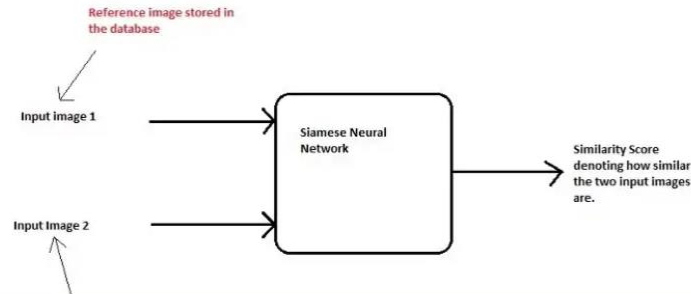
cv2.imshow('FRAME',frame)
if cv2.waitKey(10) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()

```

One-shot learning.

The need of adding objects to the detection model is very important. The process involved to add new objects to a object detection takes a long time and techniques related to transfer learning can be used to add new custom objects. Even transfer learning needs an extent of many objects (minimum of 200 images per class) from a new specific class to train a model to detect objects in real time.

To overcome the model training barrier a new implementation can be used to expand an existing ML(machine learning) model with new objects with less number of data in a dataset. This can be achieved using Siamese neural networks. Siamese neural network is a CNN which can train classification models using less number of images. With a minimum of 10 image for a class it can train a simple one shot classification model



Based on the images given to the training data set it trains the model by calculating a similarity score for images. When inferencing the model it fetches an image from the database and creates a similarity score with the base images in the database. Based on the similarity score (figure) it predicts the class for which the object can be categorized.

Data preprocessing and dataset.

The Siamese classification network works as a classifier so all the images in the data set need to be in same size. For the training purposes we used 150x150 dimensions for images. This is required to make a similar size vector when training using the neural network. The similarity score is created based on the black and white features of the image.



Sample images used for training of one shot learning model

When creating the dataset the preprocessed images should be labeled under relevant class name.

Libraries installed to train the model

```
[ ] import numpy as np
import os
import random
import pickle
import time
from skimage.io import imread
import tensorflow as tf
from keras.layers import Input, Conv2D, Lambda, Dense, Flatten, MaxPooling2D, BatchNormalization
from keras.models import Model, Sequential
from keras.optimizers import Adam
from keras import backend as K
from keras.regularizers import l2
import numpy.random as rng
from sklearn.utils import shuffle
from PIL import Image
```

Methods used to preprocess the data.

```
def loadings(path,n = 0):
    ...
    path => Path of train directory or test directory
    ...
    X=[]
    y = []
    cat_dict = {}
    lang_dict = {}
    curr_y = n

    # we load every alphabet separately so we can isolate them later
    for alphabet in os.listdir(path):
        print("loading : " + alphabet)
        print("current y: ", curr_y)
        lang_dict[alphabet] = [curr_y, None]
        alphabet_path = os.path.join(path, alphabet)
        print(alphabet_path)

        category_images=[]
        print("length category names", len(category_images))
        # every letter/category has it's own column in the array, so load separately
        for letter in os.listdir(alphabet_path):
            print("current y: ", curr_y)
            cat_dict[curr_y] = (alphabet, letter)
            print(cat_dict)

            letter_path = os.path.join(alphabet_path, letter)
            print(letter_path)

            p = Image.open(letter_path).convert('L')
            j = p.resize((105, 105))
            j.show()
            print(f"resized size : {j.size}")
            j.save('img.jpeg')
            k = Image.open('img.jpeg')
            category_images.append(k)

            y.append(curr_y)
        try:
            X.append(np.stack(category_images))
            # print(X)
            # edge case - last one
        except ValueError as e:
            print(e)
            print("error - category_images:", category_images)
            curr_y += 1
            print(cat_dict)
            lang_dict[alphabet][1] = curr_y - 1
    y = np.vstack(y)
    X = np.stack(X)

    return X,y,lang_dict
```

Trained Siamese CNN

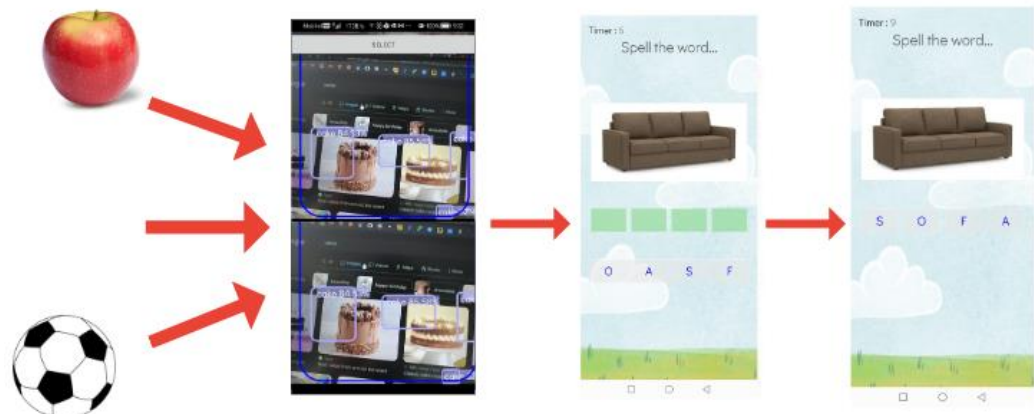
```
[ ] model = get_siamese_model((105, 105, 1))
model.summary()

Model: "model"
-----
Layer (type)                Output Shape         Param #       Connected to
-----
input_1 (InputLayer)         [(None, 105, 105, 1 0
                        )]
input_2 (InputLayer)         [(None, 105, 105, 1 0
                        )]
sequential (Sequential)      (None, 4096)         38947648      ['input_1[0][0]',
                        'input_2[0][0]']
lambda (Lambda)              (None, 4096)         0             ['sequential[0][0]',
                        'sequential[1][0]']
dense_1 (Dense)              (None, 1)            4097          ['lambda[0][0]']
-----
Total params: 38,951,745
Trainable params: 38,951,745
Non-trainable params: 0
```

Interactive Game development

The child's interaction with the learning process successful when he is enjoying the process of learning through the environment. The child gets to capture objects from the environment and it is directed to a game where he gets to involve in the process of learning the pronunciation (Phonetical pronunciation) of the word. Hear at the mean time the child gets to experience of the real time object in the environment and also the wording of the object.

The sounds of the objects are exposed to the child in the manner of phonetical sounds are learnt by the child.



The game UI showing the the path of child learning the pronunciation and the sounding of the words.

5. Testing & Implementation

5.1 Implementation

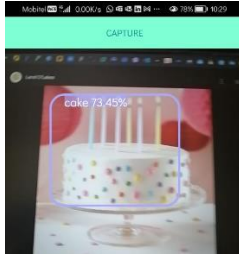
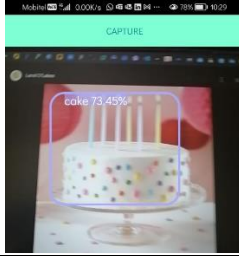
The product implementation is mainly targeted to an mobile application integrated with all the features mentioned above. A mobile application which a hearing impaired child can use with the inspection of an adult/ teacher. The front end of the application is developed using React Native. The main purpose of using react native is to accomplish cross platform compatibility. The object detection model is executing as an on device model where it can detect objects on real time. All other models will be executed into the backend flask server which is running on a Flask server. The one shot learning model needs realtime learning and the computing resources in a mobile are limited to run a the model training as a background job. This model training will happen in the backend server.

All the models are trained using Google Colab. The object detection model was trained initially as on device to check the compatibility of running a training model on device. Due lack of performance it was migrated to Google Colab. The YOLOv5 object detection model was converted to TFLite (TensorFlow lite) version since it was the only compatible model to run on mobile environment. The Python version used to train model was 3.9.2

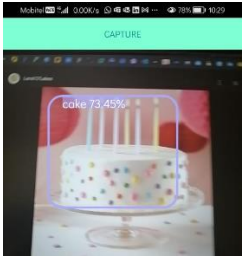

5.2 Testing

All train models and app flows need to be tested against the expected outputs. The predictions of the object detection models are tested with the accuracies To get the best output of it. The testing is process which check the procedure and the flows which helps us to identify the risky areas in the final product.

Test Case Id	01
Test Case	Verify objects detected in the mobile app
Test Scenario	Verify the realtime objects show in camera show actual object names.
Input	Camera feed

Expected Output	
Actual Result	
Status (Pass/Fail)	Pass

Test Case Id	02
Test Case	Directing to relevant game after capturing the object
Test Scenario	Capture an object from the environment The relevant gaming environment should be loaded.
Input	Capture a cake image from the camera
Expected Output	

		
Actual Result		
Status (Pass/Fail)	Pass	