University of Essex

School of Computer Science and Electronic Engineering

CE903 – 7: Group Project

Final Report

Team 30 Smart Glasses Application to Avoid Accidents for Blind People

Students name and registration numbers:

Carlos	Romano Gomez	2201887
Ricardo Vladimir	Ortega Ramirez	2201649
Priscila	Montoya Beltran	2201316
Thanaphoom	Pratjaroenwanit	2209324
Rayan	Pinto	2207973
Shreya	Tirunagari	2204561
Vivek	Unni	2200426
Varun	Nittur Shekar	2202118

Module Supervisor: Dr. Manoj Thakur

Project Supervisor: Dr. Sefki Kolozali

Second Assessor: Dr. Faiyaz Doctor

Version 1.3 Date: 23 / 03 / 2023

Table of Contents

List of Figure

List of Table

I. Abstract

To help the increasing number of blind people, the project uses smart glasses with a machine learning algorithm to alert blind people of possible dangers that lay ahead. A retrained YOLOv3 structure was used after a comparison between Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) showed CNN structure performs better. The model could recognize different objects in a real-time or offline video according to their location in the frame's left, middle, or right sections; it also showed the bounding box to determine if it was detecting correctly. Depending on where the object was located in relation to these sections, some advice was sent to the user through the speakers of the smart glasses. Future approaches to the project may include to consolidate the glasses connectivity, and to implement a self-produced model, because these 2 tasks will need a lot of time, which we didn't have.

II. Introduction

In the last few years there has been an increasing number both in blind people and visual impairment around the world. This is a severe problem throughout the world because loss of vision, total or partial, represents a loss in people's quality of life because simple tasks become difficult, and the lifestyle changes abruptly compared to those with good visual ability. Many times, these problems are not given the necessary importance. For example, being that a lower hearing capacity is not considered as relevant as having a lower visual capacity, despite the fact that since the Equality Act 2010 having a hearing loss is considered a disability. According to data from the NHS, "in the United Kingdom alone there are more than 2 million people with loss of vision, of which 340,000 are registered as visually impaired. Of this group, approximately 80% are aged 65 and over and just over 60% are over 75 years of age". [1] This information represents a group at risk of losing a large part of their quality of life, which by nature has also decreased in other aspects, making their day-to-day life even more difficult.

The implementation of modern technological devices such as smart glasses with the support of effective algorithms can represent a radical change in people's lives. Several devices have been developed by several companies which aim to facilitate partial or fully blind people's lives, for some years now. Nevertheless, many of these devices stay at the development stage or are too simple to be used on a daily basis. This is why the use of deep learning techniques for objects and text recognition plays an important role in this technology, since it is the basis of its working, the better or more complex the machine learning algorithm, the more actions the device can help on. This is possible since currently, the state of art for the development of recognition algorithms is solid and it is possible to find much research that either implements new strategies or try to improve the ones that are in the field. Most of these research projects already offer the code they use, and they are mostly based on the use of Convolutional Neural Networks (CNN) architectures, which by nature meets the requirements of its application on smart glasses: fast response and high precision on reading images.

In the same way, some developers try to reach better results by implementing different training approaches for their models such as "Transfer learning" technique, which means to extract features from a model with high precision and replace the last layer of the model with the new model which will be trained in the desired task [2]. By making use of this technique, the model increases its precision and requires less training data, since it's been already trained for a similar task. It is believed that this method can make a system faster to implement and at the same time preserve the precision that a final product needs. So, in order to implement these

types of models in a device that is also accessible to the public with ease, the device needs to be hardware constrained, meaning that the hardware does not necessarily have the requirements to fulfill these tasks. That's why the device may need to have a connection with the internet or other devices that can run the complex machine learning algorithm and send back a response to the original device in a short time.

To accomplish this project, Vuzix Blade Upgraded glasses were chosen, this device offers a camera within the glasses that can record and stream video, built-in stereo speakers in the temples and a reliable connection over Bluetooth or Wi-Fi to other devices. Other features that this device offers are microphones with noise-canceling and a small screen within the glass's lens. These are mainly used to guide doctors through an operation in real time, engineers or technicians with online help through streaming what the technician is seeing, or simply for hands-free video calls [3]. Also, we are looking forward to implementing different deep learning architectures, other than CNN to compare performances, along with different approaches such as Transfer learning to improve final results. With this, we expect blind people to be assisted by our glasses following the next process: Image input, image send to external device, image recognition via deep neural network, sent back prediction, output the prediction through glasses speakers.

III. Objectives

As said before, the main goal of this project is to create an app for blind people that uses deep learning algorithms that can identify objects in real-time video, send a warning to the user via speakers, and alert them of potential danger. To fulfill the goal, the team set several milestones, this includes dividing the project into 3 workspaces: Connection glasses, Machine Learning Algorithm and Cloud (Federated Learning and response) teams. The specific objectives include:

- Connection with "Vuzix Blade Upgraded" Smart glasses hardware for real-time video input.
- Create and compare at least 2 neural networks models.
- Train the best-performance model via cloud by Federated Learning technique.
- Retrain and use transfer learning with the selected model to recognize different objects.
- Locate the recognized objects and set the output via Smart glasses' speaker.

IV. Technology use

In order to make this project possible, we needed much technology in all levels of development, from code editors to devices in its entirety. The list of all software and hardware used to complete this project will be listed below with a small description of its utility.

Visual Studio Code

The Visual Studio Code (VS Code) is a free open-source code editor that popular used by developer because the interface of VS Code was clean and modern that make it easy to use.

Jupyter Notebook

The Jupyter Notebook is an open-source web-based tool that used in data science, machine learning, and also scientific computing. This IDE was allowing to create live code, visualizations that was helpful to visualize the data for the machine learning.

Android Studio

Android Studio software will be used to develop android application for the smartphone for our desired specifications and purpose.

labelImg

LabelImg is now part of the Label Studio community. Label Studio is an open-source data labeling tool for images, text, hypertext, audio, video and time-series data [4]. This application helped us to label images that would have helped in order to retrain a premade neural network model [20].

YOLO

is an object detection algorithm that can detect objects in real-time in images and videos by dividing an image into a grid and predicting bounding boxes and class probabilities for each grid cell using a single neural network.

Google Firebase

The Google Firebase was commonly used as a database or real-time database for the mobile applications. The Firebase database was a real-time sync and reflect in all clients in real-time. Moreover, Firebase was easy to integrate in mobile application, making it easy for the project.

Mobile Phone

The cell phone was use as the processor of the system. An application was receiving an input from the smart glasses and process in cellphone to get an output and also connect to the cloud for federated learning.

Vuzix Blade Smart Glasses

A wearable device that directly interface with the blind people. This device was integrated with camera to use for as a system input and also have an integrated speaker that use as for an output of the system.

V. Related Work

Throughout the years, many companies and researchers have explored the idea of using hardware such as glasses to help blind people avoid obstacles. This devices range in their capabilities and Some of the principal projects in this area

Previous experiments, such as those conducted by UEM Jaipur [17] and SNDT University [18] in India, involved ultrasound systems that beeped when detecting obstacles within a certain distance. Another project used OpenCV, OCR, and a Raspberry Pi card to create a text-to-speech device, but it functioned optimally only in high-quality image settings. Conversely, the High Institute for Computing and Information Technology's glasses prototype [19], which closely aligns with this project's goals, had a camera for object recognition and a text-to-speech feature. However, their Visual Studio console application was not portable, and the system displayed poor accuracy in low-light images. By analyzing these projects, we can identify the right path to follow, avoid previous errors, and leverage successful features to enhance our system in the long term.

Over the past decade, there has been significant progress towards the use of machine learning in the context of object detection for real-time videos. Each year, there is a significant advance in this topic due to the development of new models and algorithms. Object detection can be explained as a computer vision technique used to locate objects in certain images or videos [5].

A large number of machine learning techniques have been explored, such as convolutional neural networks (CNNs) and region-based convolutional neural networks (R-CNNs). In 2016, Redmon et al. proposed a real-time object detection model, which is called You Only Look Once (YOLO). YOLO is an algorithm based on a single convolutional neural network that predicts bounding boxes and class probabilities directly from the images or video [6]. On the other hand, before YOLO existed, Girshick et al. (2014) developed a region-based convolutional neural network (R-CNN). Girshick's model proposes first selecting regions of interest within the image and then applying a convolutional neural network to each of these regions [7]. These R-CNNs have been widely used for several different applications, such as face and pedestrian detection.

Although the progress made in this topic has been outstanding, there is still room for improvement and much more challenges to overcome. The main challenge at the moment would be the achievement of using these models on real-time video on other devices, such as smartphones.

Federated learning has grown in popularity in recent years because of its promise for privacy-preserving machine learning. Much research has been conducted on various elements of federated learning, such as optimization algorithms, communication protocols, and security and privacy concerns. Li et al., for example, provided a safe and efficient federated learning system based on homomorphic encryption, allowing clients to encrypt their local data before transferring it to the server for model training [8]. Yang et al. created a distributed federated learning system that leverages reinforcement learning to improve federated learning communication and computation overhead [9]. These papers reflect the breadth of federated learning research and emphasize the need of solving both technical and practical difficulties in this new discipline.

FedAVG is a common federated learning approach for pooling local model changes from several clients. Using stochastic gradient descent, each client computes a gradient update on its local data and delivers it to a central server. The server then combines the changes using a simple averaging procedure and broadcasts the averaged update back to the clients as the global model update. FedAVG has been widely utilized in numerous applications of federated learning, including image classification, audio recognition, and natural language processing. McMahan et all. For example, they utilized FedAVG to train a machine learning model to predict customized language models on mobile devices [10]. They demonstrated that FedAVG provides accuracy equivalent to centralized training while protecting user privacy. Several research have demonstrated FedAVG's usefulness in lowering communication overhead and increasing convergence speed in federated learning [11][12].

VI. Literature Review

One of the most important senses for humans is vision since it allows us to understand our surroundings. Yet, millions of people around the world have visual loss. Their inability to see obstacles in their environment makes it difficult for them to navigate daily life, and one of their largest problems is recognizing individuals. Object detection is employed in a range of still unexplored applications aside from automation. A face recognition system with audible output is one such application that uses detection to help visually impaired persons identify objects ahead of them for safe navigation. It also uses detection to help visually impaired people recognize both familiar and new faces. Voice-based support. In this study, we used a deep learning-based Faster Region-Convolutional Neural Network (Faster R-CNN) to recognize and classify people and environmental objects. The audio jockey gets the recognized picture as an audio input and processes it using a faster region convolution neural network algorithm. White canes are less pleasant for visually impaired people.

Blindness and vision impairment make it difficult for sufferers to move around on their own and deal with problems in their daily life. Artificial intelligence and computer vision techniques help blind and visually impaired (BVI) persons carry out their core tasks without being overly reliant on others as a solution. Potential assistive technology for BVI citizens includes smart glasses, which can improve social comfort and safety while assisting with independent travel. Practically speaking, the BVI cannot move by themselves, especially at night and in dimly lit areas. In this work, we suggest a smart glass system for BVI individuals that makes use of deep learning models, acoustic feedback, tactile graphics, and computer vision approaches to enable autonomous movement in a dark environment. A low-light image enhancement model, an object recognition and audio feedback model, a salient object detection model, and a text-to-speech and tactile graphics generating model are the four models that make up the system. This system was created to help in the following ways: (1) improving image contrast in low-light situations using a two-branch exposure-fusion network; (2) guiding users with audio feedback using a transformer encoder-decoder object detection model that can identify 133 categories of sound, such as people, animals, cars, etc.; and (3) accessing visual information using salient object extraction, text recognition, and refreshable tactile display We assessed the system's performance and found that it performed favorably on the difficult Low-Light and ExDark datasets.

In this difficult progression, the main role of object detection necessitates computer vision that deals with both indoor and outdoor classes. This zeal has grown increasingly demanding over the years. Earlier implementation methods used a single labeling mechanism for object detection. Goals and Purpose: In this sense, a multi-label method employing vision and machine learning technology can produce an accurate response that can be recognized for its effectiveness. In the suggested work, we apply classification/clustering strategies to reduce the recognize time of multiple objects in less time with the best time complexities in order to address the system problem that currently exists. Model: The model used to help the blind can independently identify objects that are close to them. The creation of these machine learning algorithms for sight impaired people to assist with correct navigation in both indoor and outdoor settings was complicated by the reverence combined with the study. In this regard, Retina Net is used to construct an indoor and outdoor architecture for its detecting methods, and neural network technologies support this framework. ResNet and FPN function as an essential module for its accuracy based on effectiveness and implementation time.

VII. Methodology

The proposed methodology for this project involves several stages. After the primary approach to the project, where we understand the project, discuss related works and set the objectives; we will set the management structure we want to follow, in this case will be the agile methodology. This methodology consists in constant collaboration and feedback through the development. This can allow collaboration and flexibility within the team and with the supervisor. In the same way, the project can bring many advantages to the development like: Iterative development, which consists in breaking down the project into different teams that can focus in a specific task and can collaborated between each other to discuss new creative ideas, this teams were selected later; and a continuous testing and improvement of the project, which can lead to small improvements or better performances in the algorithm for identifying possible issues earlier [13].

Secondly, as said earlier, the group breaks down into 3 teams: Connectivity (Glasses group), machine learning algorithm (ML group), and server connectivity (Cloud team). Each of these teams was involved in specific tasks regarding their own goals, with weekly meetings where we can gather feedback from each other and from the supervisor. Thirdly, we will integrate the ML algorithm into the Vuzix Blade Glasses and conduct a small test simulating being visually impaired individuals to evaluate the effectiveness and usability of the glasses. Finally, we will analyze the data collected from the usability testing to refine and improve the ML algorithm prototype.

A. System Design

The original system architecture followed a linear communication process. The video input would begin from the glasses and sent to the cellphone where a simple machine learning model analyzed each frame to detect objects. The weights of the model were then sent to a cloud server, where a more complex model would finish detecting the objects and evaluate the situation's level of danger for the user. The resulting information would then be sent back to the glasses via the phone, alerting the user with an audio alert. However, due to the system's complexity and lack of resources, this architecture only remained theoretical and was not implemented.

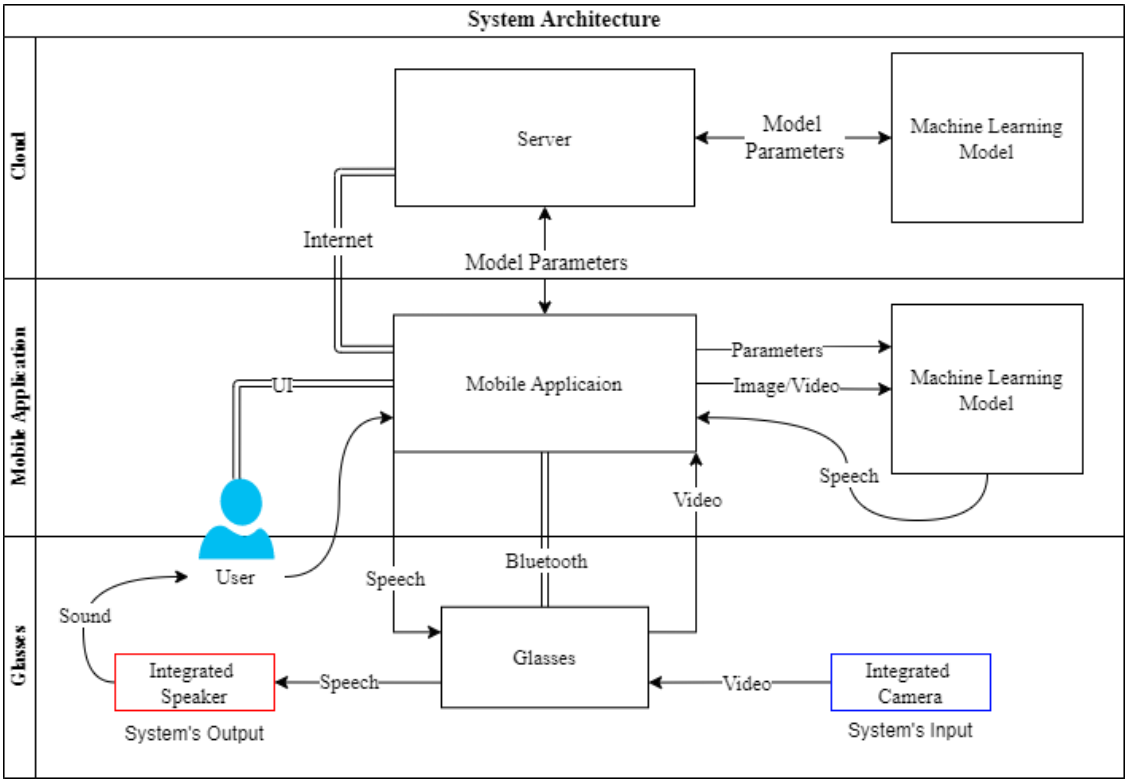

Figure 1 Initial System Structure

While managing the problems that came up while developing each group of the previous structure, and once we understood the system in a better way, we needed to adapt a new structure that could demonstrate the idea behind the structure without having to connect every part in a single one.

This would work in a similar way than the original structure, but each part in an independent way. This would help to understand the same idea without having to develop the whole system as it was planned. An explained diagram of this system is shown in figure 2.

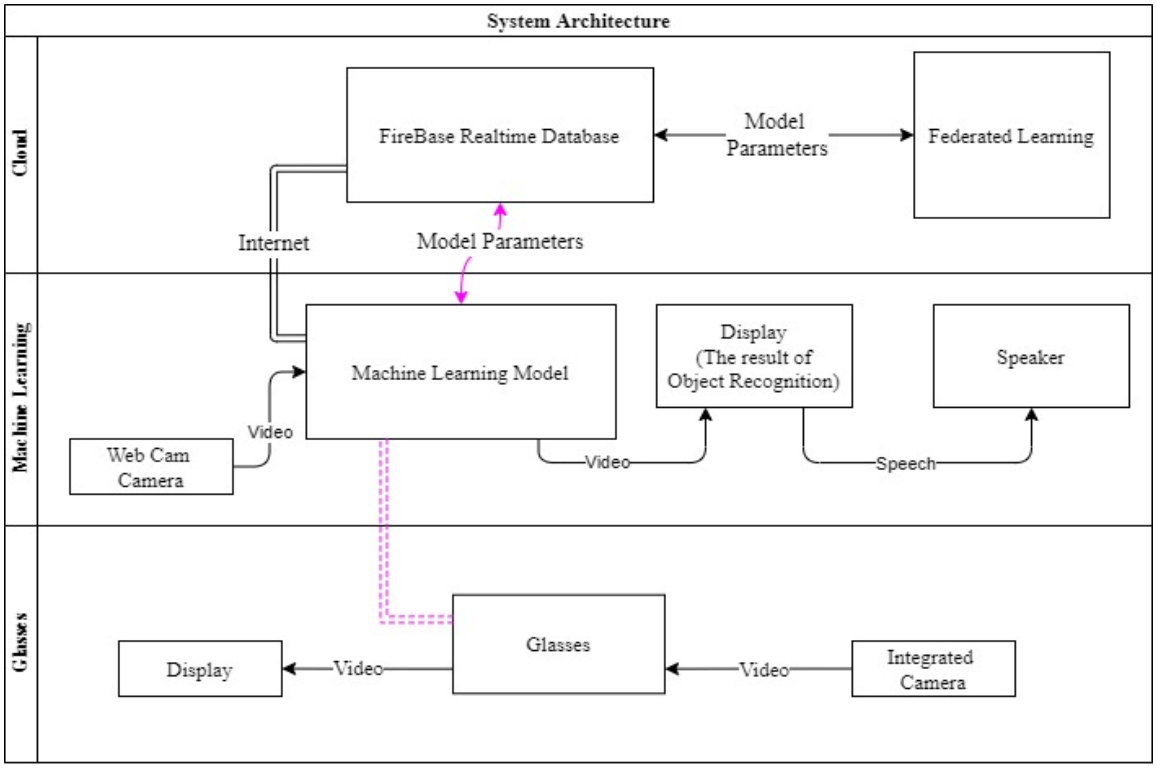

Figure 2 New System Architecture

The figure 2 was present it is necessary to understand each part separately. One part consists in the glasses, where a subsystem will be created to only try to send the live video to a web page or other display, as a streaming service, this will demonstrate that the glasses can do that. The second part will be about machine learning. This part will use pre-recorded media to train a machine learning, but will predict on live media, display the results on the same device that has the model and send a response in there too. This will demonstrate that the model works and that it can have a live input, and an output based on the location of the objects in the images. Finally, the cloud part represents a system that runs on its own: for this system we will be using federated learning with a device and dummy neural networks that will send the model parameters and receive the average of all the weights as a response from the server. The pink lines in the system would represent the ways in which each area can be connected, but that was not made in this project. This could be taken as future work for the project.

B. Object Recognition and Machine Learning

1. Data Collection

In order to make an object detection model, that its trains by us, the network will need a series of images of the objects we want to recognize, so to came up with a database large enough to separate the data in training and testing, the data collection, must be done by ourselves and label each image individually. To make this, first, the team had to come up with the ethical approval from the university to run video recordings around campus, this ethical approval was submitted by the supervisor as a formal process. Once it was approved, the team used the Vuzix Blade integrated camera to collect real-time video data for the object recognition system. The team chose to collect data around the university area to have a diverse and representative sample of objects and scenes for the object recognition system to learn from. The team recorded and saved it in the mp4 format, which facilitated the labeling and preparation of the data for the machine learning model. In the figure 3 it is shown how the members recorded around campus with a notice that it was a video recording in progress, as the ethical approval commanded.

Figure 3 Data Collection

Figure 3 in the context of the project likely depicts an individual walking around a university area while wearing a pair of glasses that can record short videos. The purpose of recording these videos is to gather data that will be used to train the machine learning algorithm to recognize different objects in the environment.

The glasses likely have a camera that records the user's point of view as they walk around. As the user moves, the camera captures the surrounding environment and records it as a short video. This video data is then fed into the machine learning algorithm, which analyzes it and learns to recognize different objects in the environment. By recording videos while walking around the university area, the project team can gather a diverse set of data that includes a variety of objects, environments, and lighting conditions. This helps ensure that the

machine learning algorithm is trained on a wide range of scenarios and can recognize objects accurately in different contexts.

2. Data Preparation

In order to prepare an image dataset to train and use into the neural network model, the images needed to be labeled. This means that the videos needed to be separated into frames and in each frame box the regions of interest (ROI's) that would be the objects we want to train the model to recognize. This process of labeling this dataset involved several steps, which were all explained in an instructions document that was elaborated by the team and uploaded to GitLab. The first step was uploading the previously recorded data onto GitLab for everyone to have access to. After the data was successfully uploaded, each member downloaded their assigned videos. Then, each member proceeded to split each video into frames by using an online converter from *mp4* to *jpg* format. Then, a specialized app, as 'LabelImg', can be used. LabelImg is an open-source image annotation tool used to label objects using bounding boxes in images. Since there's no downloadable file for an easy-install, in order to access this application, we needed to download it from the terminal, which was done by writing the command line *"pip3 install labelImg"* in the Anaconda terminal [4], which mean that the user must have Anaconda installed previously. Then, access the folder with the app and called it with *"labelImg"* command.[20]

Once the application was accessed, the directory containing all the frames of the video were uploaded to it so the process of labeling could start. To label the images, the RectBox tool was used. This tool works by making a bounding box around the object that we want to label and then adding the class name. It was important to make sure to label the images with the right classes. In this case, it was opted to use classes to detect objects that a blind person would come across in their daily life, such as Door, Trash Bin, Backpack, Chair, Person, Stairs, Table, Elevator, Bench, and Car.

After each frame was labeled correctly, it was important to ensure that the data was being saved in YOLO format, which is a .txt file with each class labeled in the image and its coordinates. This would make it possible to use it in the model. In figure 4 and 5, it is shown some screenshots using the software tool to label the images, in here it can be seen the boxes drawn around the objects, the classes that have already been called, and the interface itself.

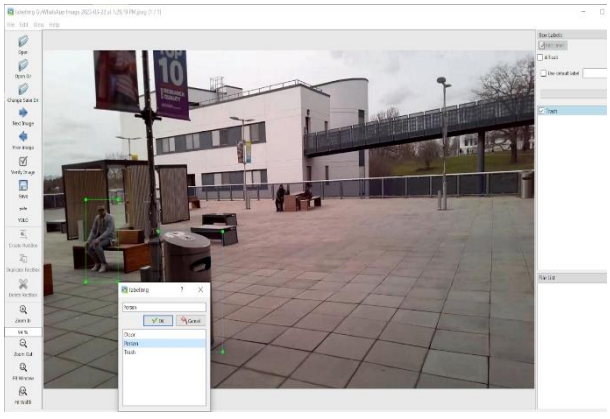

Figure 5 Data Labelling

Figure 4 Image already labeled in labelImg

With all the videos collected and labeled, they can already be used into a neural network model. Even though they can be used, just for training and comparing the models, we are going to use a premade dataset imported from the same library: Keras model Cifar10. This dataset contains 10 different classes with 60,000 images for every class, 50,000 images for training

and 10,000 for testing [14]. This dataset was chosen to have a large dataset and see the actual performance of the model. So, in order to use the dataset into our model all the images need to be reshaped first, so we can have a 32x32 image, then we can indicate this into the input layer of the model and start training with the dataset. Now that we have selected our input, the next step will be to choose the model. According to the literature, the data which you are working with will determine the type of neural network to use, in this case, since we are working with images, the best approach would be a Convolutional Neural Network (CNN), due to its ability to its dimensionality reduction feature and the effectiveness in not losing much quality in the process [15]. This is also the model widely used for image detection and classification.

On the other hand, to test the effectiveness of the model and corroborate the literature, we are going to use another model to compare with, in this case a Recurrent Neural Network (RNN). This type of network is widely used in time series information, due to the long short-term memory (LSTM), it means that information that has been read in the past, can affect the present information, unlike CNN's where the past information is forgotten. This type of model is used in image recognition, since it can remember the objects in the past and predict the actuals, leading to it being possible to predict future positions or to describe what happens in the image with ease [16], In order to use this network with images, we may have to use convolutional layers, that are the ones that the CNN model uses, but input them into a time distributed layer, to make them suitable for the LSTM layer. In figure 15 and 16 in an Appendix B a structure for each model is shown.

3. Model Comparison

The purpose of the next 2 figures is to represent the accuracy scores over time, and to visualize how the accuracy of the models changes over time as the model is trained with more epochs. The blue lines represent the accuracy score of the training dataset, while the orange lines represent the accuracy score of the validation dataset. With this, we can gain insights into the performance of the model and identify any issues such as overfitting or underfitting.

A high training accuracy with a low validation accuracy can indicate overfitting, while a low training accuracy with a high validation accuracy can indicate underfitting. Ideally, you would want to see both the training and validation accuracy increasing over time, indicating that the model is learning the underlying patterns in the data and generalizing well to new data. If the model is overfitting it means that it is performing well on the training data but poorly on the validation data, but if it is underfitting it means that it is performing poorly on both the training and validation data.

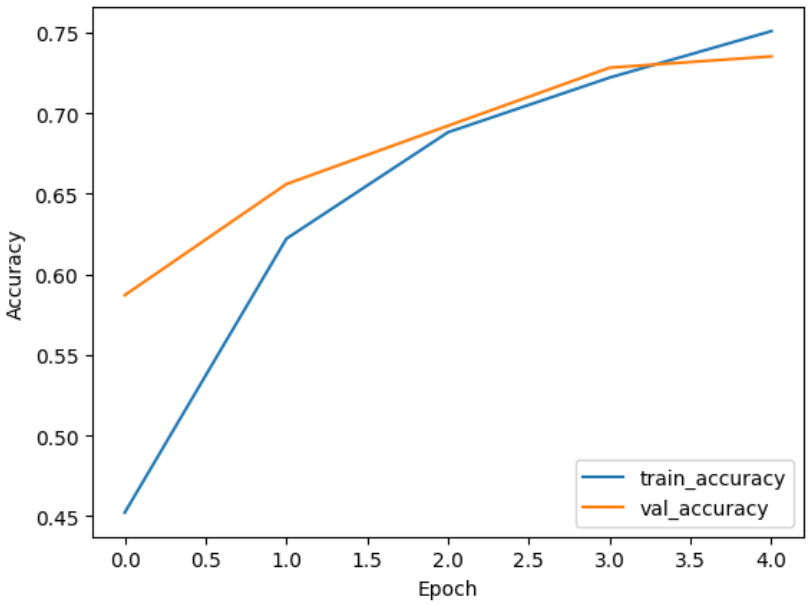

Figure 6 CNN Model Accuracy Score

As shown in the figure 6, the graph represents the accuracy score of the Convolutional Neural Network (CNN) models on the y-axis, compared to each epoch on the x-axis. In this case we can see that in the last epoch, the model may start to overfit, so the last epoch makes the performance of the model before it starts to perform bad on new data.

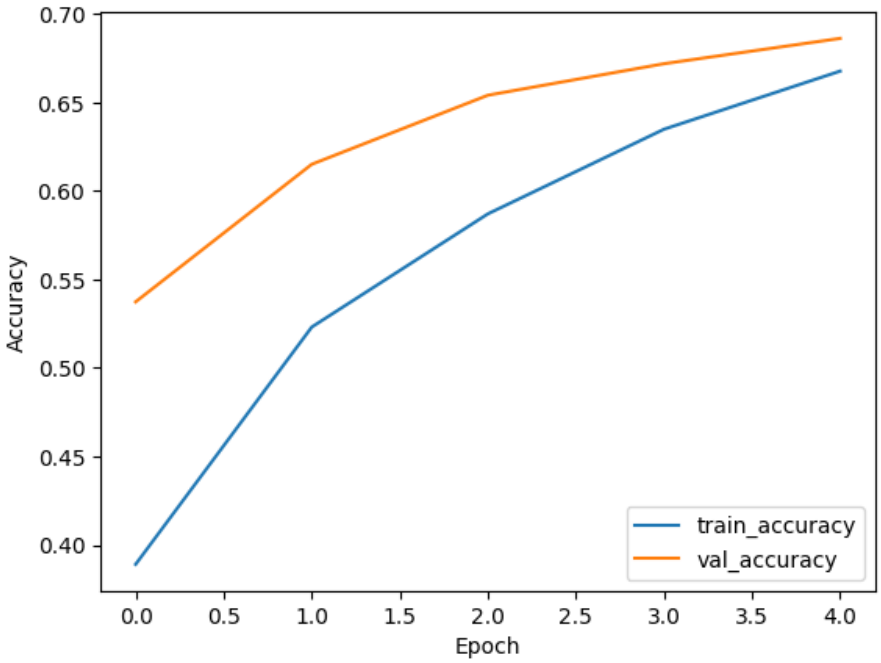

Figure 7 RNN Model Accuracy Score

The following graph in the figure 7 shows the performance of a Recurrent Neural Network (RNN) model over multiple epochs, the same as before, the graph shows how it performs over time. In this case may seem that is underfitting, but in reality, when adding more epochs it makes the model to overfit, meaning that this is the best accuracy.

	Epoch 0	Epoch 1	Epoch 2	Epoch 3	Epoch 4
CNN Training	0.45	0.62	0.67	0.71	0.74
CNN Validation	0.59	0.65	0.68	0.72	0.72
RNN Training	0.38	0.52	0.57	0.61	0.66
RNN Validation	0.54	0.63	0.65	0.68	0.69

Table 1 Accuracy Score Comparation

As table 1 presents the accuracy scores for each epoch of a machine learning model. The first column lists the epoch number, with the second column displaying the training accuracy score and the third column displaying the validation accuracy score.

Testing these 2 models with the Cifar10 dataset, we obtain the next results shown in figure 6 and 7, where we can see what it was expected: the CNN performed better than the RNN. Therefore, for the project's objective of object classification in images, the team decided to focus on using CNNs.

As the team focused on using CNN´s, one of the firsts approaches of the team, and the one that was more advanced, was to use a pretrained model that uses the CNN architecture to detect many objects in a real time video. This was achieved, and also the code could send an

auditory response, like the main plan was. The problem with this approach is that the images used for training the model were unknown, and the model had many classes that we did not need.

The pretrained model that the team came up with was the You Only Look Once, also known as YOLO version 3. This model can predict up to 80 classes, and has about 24 convolutional layers, with YOLO layers between them. This can be seen in the figure 8.

```
person          wine glass
bicycle         cup
car             fork
motorcycle      knife
airplane        spoon
bus             bowl
train           banana
truck           apple
boat            sandwich
traffic light   orange
fire hydrant    broccoli
stop sign       carrot
parking meter   hot dog
bench           pizza
bird            donut
cat             cake
dog             chair
horse           couch
sheep           potted plant
cow             bed
elephant        dining table
bear            toilet
zebra           tv
giraffe         laptop
backpack        mouse
umbrella        remote
handbag         keyboard
tie             cell phone
suitcase        microwave
frisbee         oven
skis            toaster
snowboard       sink
sports ball     refrigerator
kite            book
baseball bat    clock
baseball glove  vase
skateboard      scissors
surfboard       teddy bear
tennis racket   hair drier
bottle          toothbrush
```

```
[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear


[yolo]
mask = 6,7,8
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Figure 8 YOLO Architecture and Classes

Using this architecture, we came up with some results like the ones shown in appendix B, where we obtained an accuracy on some objects, but also some false positives. This only confirmed the necessity to retrain the model and adapt it to our necessities but speaking with each other we realized that there was some confusion and using a pretrained model in the final recognition was not permitted, so we needed to keep working with the other 2 architectures that we had developed earlier. Despite this prohibition, if this model was to be used in the final model with the glasses, it would need some improvements to run the net faster, because currently it runs in a few frames per second, and to retrained it with the labels that we already obtained with the labelImg app in order to recognize the most important objects that a blind person could encounter while in university and that were mentioned before, this code will be available with the final code zip file.

Then, to convert the CNN model that we had used before, there were many complex ways to do it, one included exporting the model in a specific file to use a function inside CV2 to detect objects based on a classifier; other included to detect all the objects in the frame, run the classifier through them all and then make the bounding boxes to determine the position of

each object; and finally the third option was to restructure the model and retrain it with the labeled images to be a recognizer instead of a classifier, but due to the confusion and lack of time management, this tests were only attempts and never finished as something concrete. All the attempt codes will be available with the zip file. The next steps after recognizing the objects in an image was to use an online video to locate the objects in real time, and depending on where they were on the frame use the speaker in the device to alert the user where to move. Both these features were implemented in the YOLO code, but since neither of the models of CNN and RNN were completely finished as recognizers, these features could not be implemented on the self-produced networks.

C. Cloud System

The system consists of an intermediate part which is identified as the “Cloud system”. This is composed of two parts: the Cloud Database, which is online, and the server (run on python) which connects to the Cloud Database.

The main purpose of this cloud system was to increase the security and privacy of the whole device. However, this purpose might not be the single use of the system, since many other ways of implementing the whole recognition device were proposed with and without the privacy approach. In other words, at some point of the development the online server was meant to serve as the main processing and recognition unit, receiving the raw images data from an Android App, storing them, and lately, being accessed and processed by the server, sending back the output in the opposite way.

For the choice of the database, the one with the greatest ease of access and the fastest reception and modification of data was considered. From a list of 3 possible databases, Microsoft Azure, Google Firebase, Amazon Web Services, only one was chosen and used. Firebase RealTime Database demonstrated to be the best option for our project since it is both python and Android friendly, providing the necessary libraries for the real time connection. This way we manage to send data from the Android App to the python server and vice versa, simulating the sending of float numbers as weights values as shown in figures 11, 10 and 11.

Figure 11 “Database” shows the current configuration of the branches inside the Firebase Realtime Database, Figure 9 “Python server” follows this configuration to connect and retrieve the data corresponding of Wx from Firebase: “Devices” - “Parametros” - “Weights”: “W1”, “W2”, “W3”, and the same applies for the “Parametros2”. It also has the function of sending data back to the database, modifying the current values. Finally, Fig X3 shows the operation of the Android App connecting with the database, reading Wx values in real time and having the option to send data to the database to modify Wx values.

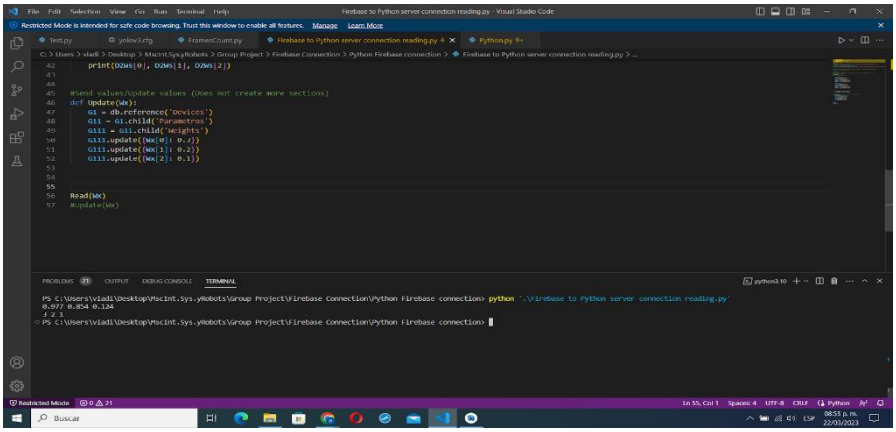

Figure 9 Python Server

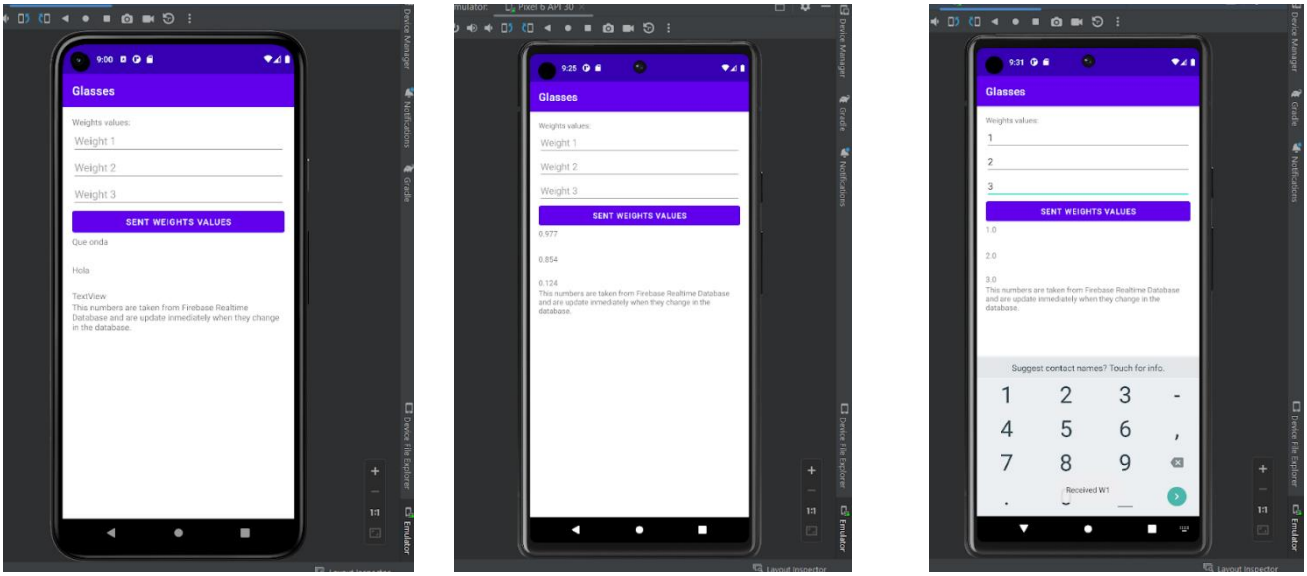

Figure 10 Android App Receive and send data.

D. Federated Learning

The system architecture presented in figure 2 shows that federated learning was implemented for the cloud part of the system. The cloud part contained the federated learning and implementation for decentralized model training, which ensures data privacy and security while still enabling the aggregation of knowledge from multiple sources. By using federated learning, the system can train models on user data without requiring the data to be transmitted to a central server. This approach also allows the system to scale up easily, as it can handle large amounts of data from many sources without requiring additional computational resources. federated learning offers a promising solution for building robust and privacy-preserving machine learning systems that can learn from decentralized data sources.

In this part are discussing federated learning, a group showcased a simulation of how federated learning could be implemented using the fedAVG methods. The fedAVG method was chosen for the project as it is a popular federated learning algorithm that allows for efficient model training across multiple decentralized devices or data sources. By using the fedAVG algorithm, the system can aggregate model updates from participating devices or nodes without sharing their raw data, ensuring privacy and security. The algorithm also ensures that the aggregated model is representative of the distributed data, thus improving the accuracy of the final model. Additionally, the fedAVG method is computationally efficient, allowing the system to handle large amounts of data and support real-time model updates.

Connection to Server

Firebase Realtime Database is a cloud-hosted NoSQL database provided by Google, which allows you to store and sync data between clients in real-time. It's an excellent choice for building real-time applications such as chat apps, social media platforms, and online games.

When it comes to Federated Learning, the Firebase Realtime Database can be used as a central server to store and aggregate the model updates sent by the participating clients. Each client trains a local model on its own data and sends the updated model parameters to the Firebase Realtime Database. The central server then aggregates the updates and returns a new set of model parameters, which the clients can then use to update their local models.

To connect to the Firebase Realtime Database using Python, you need to use the firebase-admin library, which is an official Firebase Admin SDK for Python. As in coding to config the firebase as a python file in GitLAB.

Figure 11 Real Time Database

After successfully configuring the real-time database, the next step is to create a child that will collect the parameters and weights and store them in the database. This data will be used to compute the average weight in the next step. To achieve this, it is important to ensure that the database structure is set up correctly. Figure 11 presents an example of how the database structure should look like.

It is essential to ensure that the parameters and weights are accurately recorded in the database to obtain accurate results. Therefore, it is recommended to follow the best practices for data collection and storage, such as using descriptive and meaningful labels and documenting any changes made to the data.

Model Evaluation

After implementing the federated function that includes the models, weights, and parameters, and using Federated Averaging techniques, the model is evaluated at the end of each epoch. This means that the performance of the model is measured after each round of training to ensure that it is making progress and improving in accuracy.

```
In [12]: federated_learning()

         Epoch 1/5
         Device 1/1
         1563/1563 [==============================] - 25s 16ms/step - loss: 1.8948 - accuracy: 0.3499
         Epoch 2/5
         Device 1/1
         1563/1563 [==============================] - 25s 16ms/step - loss: 1.3888 - accuracy: 0.5021
         Epoch 3/5
         Device 1/1
         1563/1563 [==============================] - 27s 17ms/step - loss: 1.2305 - accuracy: 0.5659
         Epoch 4/5
         Device 1/1
         1563/1563 [==============================] - 26s 16ms/step - loss: 1.1182 - accuracy: 0.6057
         Epoch 5/5
         Device 1/1
         1563/1563 [==============================] - 27s 17ms/step - loss: 1.0353 - accuracy: 0.6357
```

Figure 12 Federated Learing Result

In the figure 12 showing the demonstrate of the federated learning for 5 epochs on one device. This means that the device went through 5 rounds of training, with each round updating the model's parameters based on its local data set. At the end of each epoch, the model's loss score and accuracy were calculated to measure its performance. The loss score is a measure of how well the model is able to predict the correct output. It calculates the difference between the predicted output and the actual output, with the goal of minimizing this difference. A lower loss score indicates that the model is making better predictions. Accuracy, on the other hand,

measures how often the model is able to correctly predict the output. It is calculated by dividing the number of correct predictions by the total number of predictions. A higher accuracy indicates that the model is performing better.

E. Mobile Application

The application is one important aspect of the project to connect the glasses via Bluetooth to be able to view and transfer files between the glasses and the app. It acts as a medium for collecting and transferring data between glasses and the cloud. The application would also show a live preview of the camera input from the glasses. Once the data is sent to the cloud for processing it would then receive text data from the cloud on the processed data which would find the objects in the video. This text input will then be converted to an audio file and sent to the glass's speaker as an output.

The part of the project was quite difficult to achieve as few times the connectivity for the glasses was successful but after certain code changes were made to achieve the actual purpose of the application the connectivity kept failing at times.

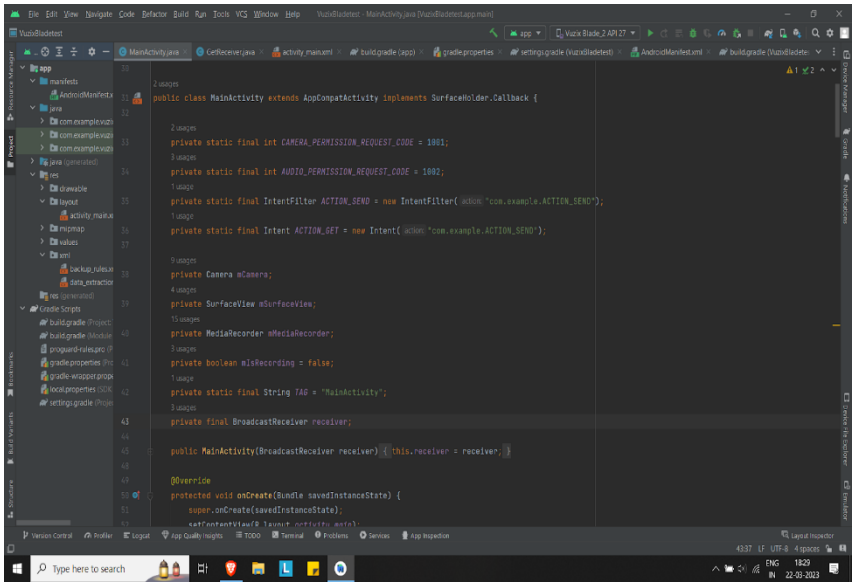

Figure 13 Mobile Application Code

The code was done using Android Studio to create an application for the smartphone and the glasses in order to try both approaches of sending video data from the glasses to the cloud. After implementing the federated function that includes the models, weights, and parameters, and using Federated Averaging techniques, the model is evaluated at the end of each epoch. This means that the performance of the model is measured after each round of training to ensure that it is making progress and improving in accuracy. The implementation application did not successfully get executed as there were complications in the code due to a lack of supportive libraries and repositories. After making multiple attempts at creating a mobile application, the team decided to try a different approach where the application would be created directly for the glasses instead of the mobile phone but still continue to try and test creating the mobile application.

F. Connection of the Glasses

By using smart glasses, it can be assured that they can be used by all people. It also offers the opportunity to use them on a daily basis with little to no struggle. With that in mind, we decided to use the Vuzix Blades Upgrades, that as told before, have an integrated camera and speakers, meaning that they suit the main utilities needed in the project. To implement them into the structure, at first, it was planned that it could send the video via streaming into the cell phone for it to process the images, nevertheless, there was a connectivity problem that could not give access to the hardware on the glasses, which slowed down the process and finally did not allow to continue with that approach. So, in order to demonstrate that this glasses could be able to do the specific task, we needed to come with another approach: To be in close contact with the company that manufactured the glasses in order to solve the problem, while we were working on accessing the video by making an application that the glasses can use and implement the model in the glasses that could send the information directly to Firebase, the server.

Despite all efforts, the application could not be done in its entirety, because of little knowledge of managing Android Studio and that the libraries for the glasses were un-updated. Now, the goals that were achieved were to make an app that could run into the glasses, even though this app was based on the code provided by the company and would only show a default template. This template can be seen in figure 14.

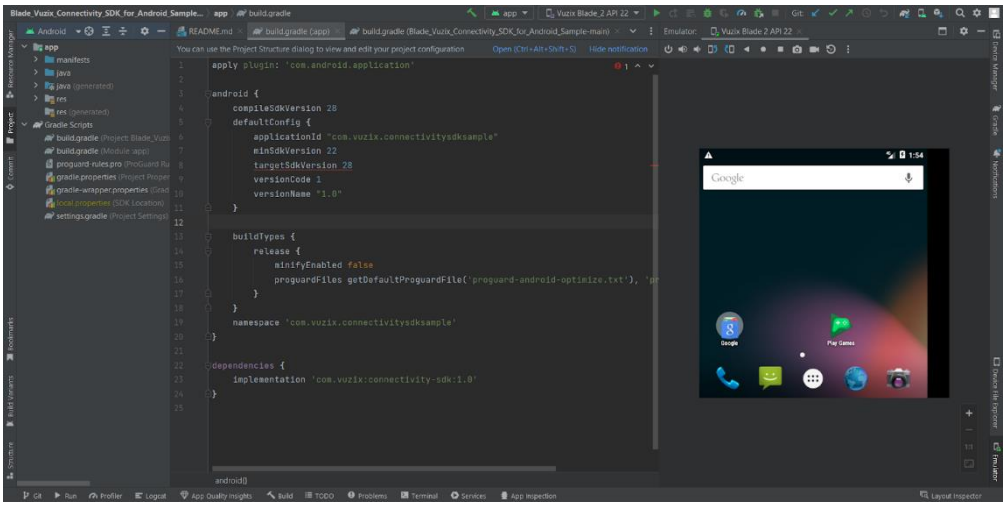

Figure 14 Template to emulate Vuzix Blade Glasses interface

VIII. Testing

Regarding the system testing, there was no testing performed for the integrated system, as the connection between groups could not be established. However, as presented in the Gantt chart from the project proposal in table 2 in an Appendix A, the chart outlined the testing schedule for each sub-team. Therefore, this section covers the testing and challenges of the sub-tasks.

A. Testing Phase I

Application

Objective: Connect the glasses with mobile phone.

Expected Result: Have a code that can send the video from the glasses to the phone in real time.

Result: The camera cannot be accessed through code, try different approaches.

Problems: the libraries of the glasses are un-updated.

Machine Learning

Objective: Have two of machine learnings model to compare each other's

Expected Result: Have the structure of 2 models and compare them through the accuracy of both.

Result: Having a model that is pretrained, and another model in progress

Problems: Bad installation of libraries to make the models, using a pretrained model is not adequate.

Cloud

Objective: The connection implementation with the central server was tested, and values were pushed into the cloud server to prepare for the FedAVG process.

Expected Result: Have a connection and communicate with the central server.

Result: Can be connect to the central server but cannot be push or pull data

Problems: take the wrong configuration to create the node for push the data

B. Testing Phase II

Application

Objective: Access the camera in the glasses

Expected Result: Use an app on the glasses or on the phone that can access the camera and speakers on the glasses.

Result: Many approaches that failed, start contacting the company to solve the problem.

Problems: The connection cannot be made; the use of other apps do not fulfill the requirement of streaming video.

Machine Learning

Objective: Have a classification model and start converting to a recognizer model.

Expected Result: Once selected the model to use, to have a code that can classify images.

Result: A model that classifies images but is too simple, need for improvement.

Problems: Simple model, trying new models but they do not work.

Cloud

Objective: The data that pushed to the cloud should be compute dur the FedAVG techniques for federated learning

Expected Result: get the list of the updated weights.

Result: cannot push all the weights to the central server.

Problems: Programming issues

C. Testing Phase III

Application

Objective: Record a video with glasses.

Expected Result: To have a real time video recording from the glasses using code or other from apart from the app in the glasses.

Result: An application developed that connects with the glasses but cannot access the integrated camera

Problems: The access of application to open the Vuzix integrated camera

Machine Learning

Objective: The Object Recognition can be detected the real time object

Expected Result: Can detect the object by using the implemented algorithms and get an accuracy in range of 90%.

Result: can detected the object by get highest accuracy in 97% and the less one is 71% as the figure 17 present in the Appendix B

Cloud

Objective: Test the implemented code for the FedAVG method and run each epoch for the simulated model

Expected Result: The running of federated learning should be successful by increasing of the accuracy in every epoch.

Result: Test was unsuccessful

Problems: the implemented code was not working and have some of an error.

IX. Conclusion

The project aimed to address a significant challenge faced by visually impaired individuals in their daily lives. The team's approach involved developing a system that incorporated glasses, machine learning, and federated learning. The glasses were intended to provide an augmented reality experience to the user, while the machine learning aspect was designed to enable the system to recognize and provide information about objects and environments. Federated learning was incorporated to enable the system to learn from data collected by users, thus improving its accuracy over time.

Despite the team's best efforts, the project did not achieve its desired outcome within the allocated timeframe. However, it is worth noting that developing technology solutions for vulnerable populations is a complex process that requires significant resources and effort. While it is disappointing that the project was not successful, the team's efforts were commendable, and the experience gained will undoubtedly be invaluable in future endeavors.

X. Appendices

A. Appendix A

	Task	Assigned member	January			Febuary					March			
			15 - 21	22 - 28	29 - 31	1 - 4	5 - 11	12 - 18	19 - 25	26 - 28	1 - 4	5 - 11	12 - 18	19 - 25
System	Glass Connection	Carlos						13 / 02 / 2023						
	Connect with cloud	Vivek								28 / 02 / 2023				
	Object Recognition	Shreya									03 / 02 / 2023			
	Situation Recognition	Varun										10 / 02 / 2023		
	Deploy	Thanaphoom												23 / 02 / 2023
	Integration System	Vladimir												
Application	Research	Vivek, Carlos												
	Design	Vivek												
	Development	Vivek												
	Test	Carlos												
	Integration	Carlos												
	Optimize	Vivek, Carlos												
ML and Cloud	Integration	Thanaphoom												
	Optimize	Priscila, Rayan												
	ML													
	Research	Shreya, Varun, Priscila												
	Design	Shreya, Varun, Priscila												
	Data Collection	Priscila												
	Development	Shreya, Varun												
	Test	Priscila												
	Cloud													
	Research	Thanaphoom , Rayan, Vladimir												
	Design	Thanaphoom , Rayan												
	Development	Vladimir, Rayan												
	Test	Vladimir												
Project Development	Requirements Document	All members					09 / 02 / 2023							
	Final Code and Reports	All members												23 / 03 / 2023
	Personal appraisal	All members (Individually)												24 / 03 / 2023
	Know the Projects	All members												
	Set teams and Objectives	All members												
	Write requirements reports	All members												
	Write Final report	All members												

Table 2 Gantt Chart

B. Appendix B

```
### -------------------------- CNN -------------------------- ###
# Initialize the model
model = keras.Sequential()
model.add(keras.layers.Conv2D(filters=16, kernel_size=3,activation='relu',padding="same"))
model.add(keras.layers.Conv2D(32, 3,activation='relu',padding="same"))
model.add(keras.layers.MaxPooling2D(pool_size=2))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Conv2D(64, 3, activation='relu',padding="same"))
model.add(keras.layers.Conv2D(64, 3, activation='relu',padding="same"))
model.add(keras.layers.MaxPooling2D(pool_size=2))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dense(num_classification_categories, activation='softmax'))
model.build(input_shape=(None,) + train_images.shape[1:]) # Build the model
model.summary() # Get a summary of the model, how many layers, neurons and parameters
```

Figure 16 CNN Models Structure

```
## -------------------------- RNN -------------------------- ##
# Reshape the data to fit the model (RNN as we are using TimeDistributed layers)
train_images = train_images.reshape(train_images.shape[0], 1, train_images.shape[1], train_images.shape[2], train_images.shape[3])
test_images = test_images.reshape(test_images.shape[0], 1, test_images.shape[1], test_images.shape[2], test_images.shape[3])

# Initialize the model
model = keras.Sequential([
        keras.layers.TimeDistributed(keras.layers.Conv2D(16, 3, activation='relu', padding="same", input_shape=(1, 32, 32, 3))),
        keras.layers.TimeDistributed(keras.layers.Conv2D(32, 3, activation='relu')),
        keras.layers.TimeDistributed(keras.layers.Conv2D(64, 3, activation='relu')),
        keras.layers.TimeDistributed(keras.layers.MaxPooling2D((2, 2))),
        keras.layers.TimeDistributed(keras.layers.Flatten()),
        keras.layers.LSTM(128, activation='relu', return_sequences=False),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(10, activation="softmax")
])
model.build(input_shape=(None,) + train_images.shape[1:]) # Build the model
model.summary() # Get a summary of the model, how many layers, neurons and parameters
```

Figure 15 RNN Models

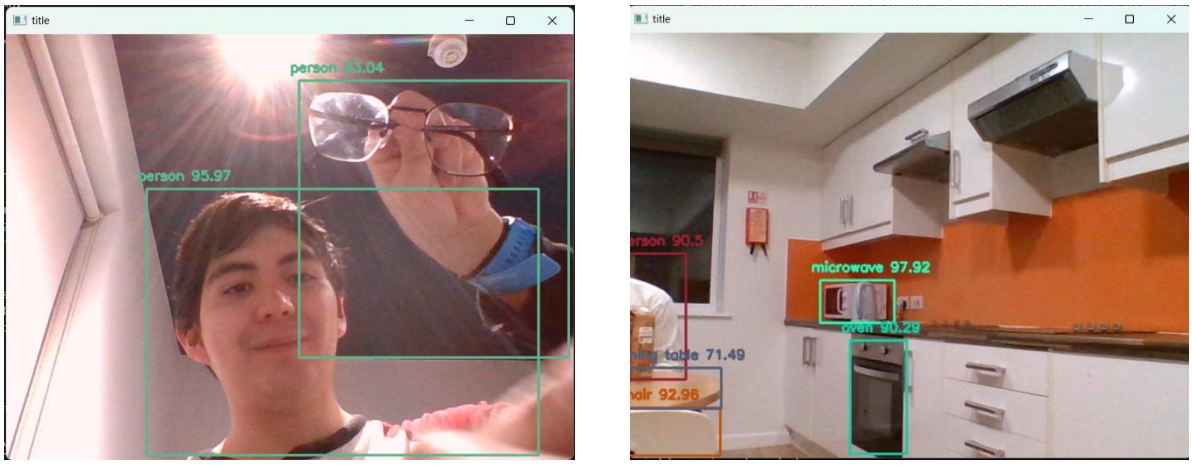

Figure 17 Prediction of Models

XI. References

[1] RNIB. (2022). Key information and statistics on sight loss in the UK. RNIB. https://www.rnib.org.uk/professionals/health-social-care-education-professionals/knowledge-and-research-hub/key-information-and-statistics-on-sight-loss-in-the-uk/

[2] Dipanjan (DJ) Sarkar. (2018). A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning. Medium; Towards Data Science. https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

[3] Vuzix. (2020). Vuzix Blade Upgrade a Step Forward for Augmented Reality Eyewear. Vuzix. https://www.vuzix.com/blogs/vuzix-blog/vuzix-blade-upgrade-a-step-forward-for-augmented-reality-eyewear

[4] Boesch, G. (2022, July 9). LabelImg for Image Annotation. viso.ai. https://viso.ai/computer-vision/labelimg-for-image-annotation/

[5] Kumari, K. (2022, March 9). A Basic Introduction to Object Detection. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/03/a-basic-introduction-to-object-detection/

[6] Redmon, J., Divvala, S. K., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2016.91

[7] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2014.81

[8] Li, H., Li, T., Yang, J., Li, F., Li, L., & Li, J. (2019). Secure and efficient federated learning via periodic model averaging. IEEE Access, 7, 11115-11127.

[9] Yang, T., Wang, S., Han, Z., Xu, X., & Ye, C. (2020). FedRL: Federated reinforcement learning for efficient and privacy-preserving collective intelligence. IEEE Transactions on Parallel and Distributed Systems, 31(11), 2445-2460.

[10] McMahan, H. B., Moore, E., Ramage, D., & Hampson, S. (2017). Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) (pp. 1273-1282).

[11] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In Advances in Neural Information Processing Systems (NeurIPS) (pp. 14880-14891).

[12] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., … & Rajkumar, K. (2021). Advances and open problems in federated learning. Foundations and Trends® in Machine Learning, 14(3-4), 221-309.

[13] Atlassian. (2019). Get started with agile project management | Atlassian. Atlassian. https://www.atlassian.com/agile/project-management

[14] Krizhevsky, A. (2009). CIFAR-10 and CIFAR-100 datasets. Toronto.edu. https://www.cs.toronto.edu/~kriz/cifar.html

[15] Mishra, P. (2019). Why are Convolutional Neural Networks good for image classification? Medium. https://medium.datadriveninvestor.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8

[16] Cohen, J. (2021). Recurrent Neural Networks (RNNs) in Computer Vision: Image Captioning. Medium. https://heartbeat.comet.ml/recurrent-neural-networks-rnns-in-computer-vision-image-captioning-ea9d568e0077

[17] Saha, Himadri & Dey, Ratul & Dey, Shopan. (2017). Low cost ultrasonic smart glasses for blind. 10.1109/IEMCON.2017.8117194. https://www.researchgate.net/publication/321288844_Low_cost_ultrasonic_smart_glasses_for_blind

[18] Kassa, S. (2021) Smart Glasses for Blind People. Vol. 7. Issue 12. SNDT University. https://www.google.com/url?q=https://ijirt.org/master/publishedpaper/IJIRT151189_PAPER.pdf&sa=U&ved=2ahUKEwjvoLaCoIb9AhXPPsAKHWNbA-AQFnoECAAQAg&usg=AOvVaw3JAJEtLyAKTaLHSr6YgdjY

[19] Ismail, W. Mostafa, A. Et al. (2017) Smart glasses for blind people. Academia. https://www.academia.edu/33796786/Smart_glasses_for_blind_people

[20] TzuTa Lin, "LabelImg", GitHub repository, 2015. [Online]. Available:https://github.com/heartexlabs/labelImg. [Accessed: Mar. 22, 2023].