A picture containing company name

Description automatically generated

ISLT

(Irish Sign Language Translator)

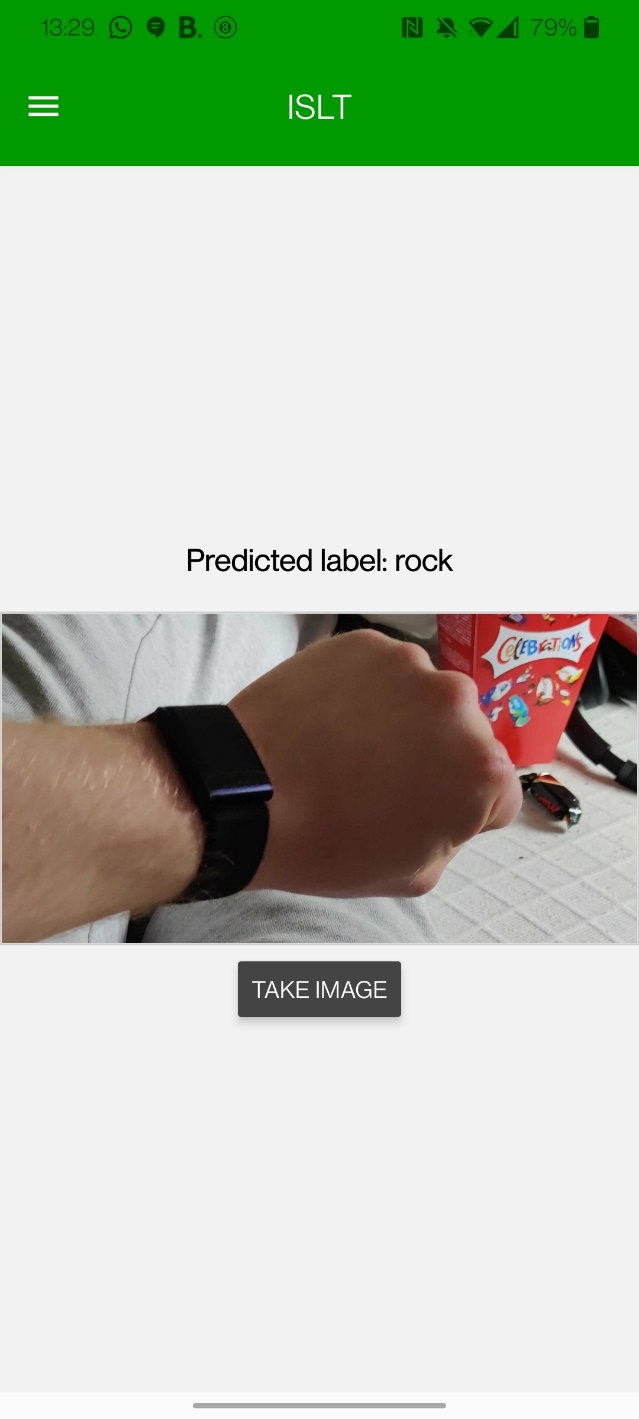
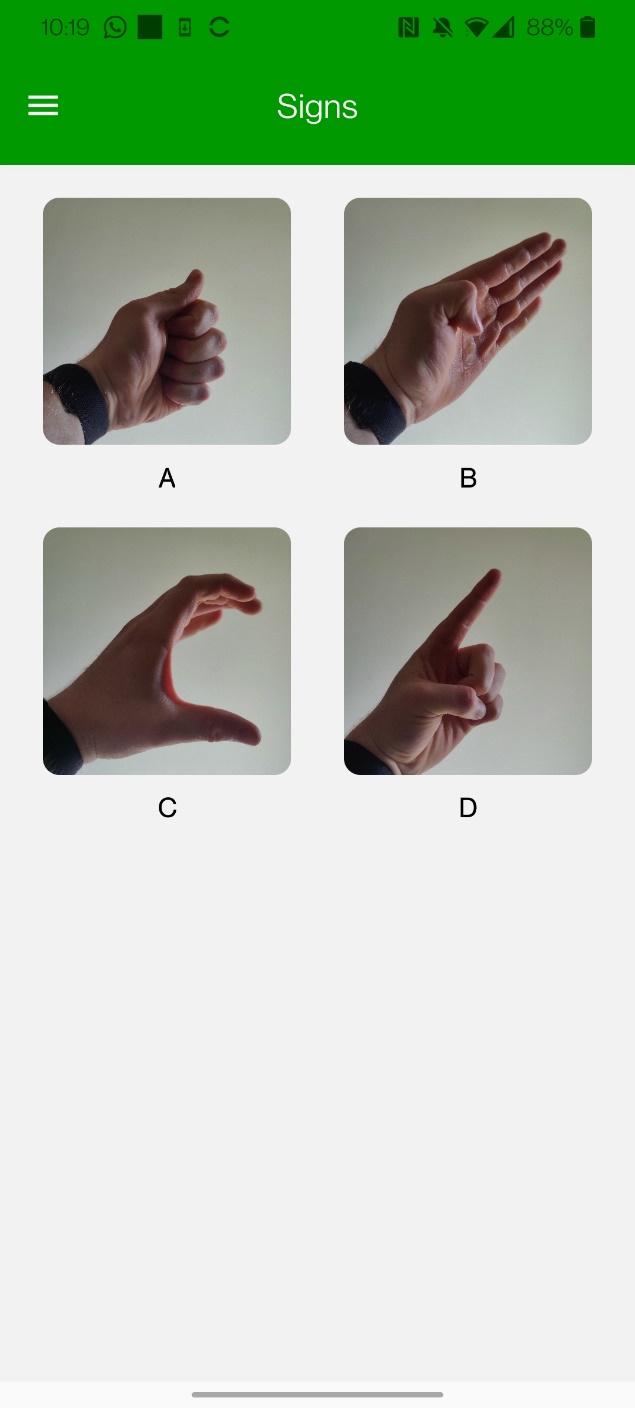
Project Engineering

Year 4

Darragh Coen

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2022/2023

**Declaration**

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at the Atlantic Technological University.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

\_\_\_\_\_\_\_\_Darragh Coen 01/05/23\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgements**

I would like to thank my project supervisor, Paul Lennon, for his assistance and guidance throughout the duration of my final year project.

I would also like to thank my Project Engineering coordinators Brian O’Shea, Michelle Lynch, and Niall O’Keeffe for their feedback and tutelage on project work throughout the year.

I would also like to thank my classmates for their feedback on all my submissions and also for their help when I ran into blockers throughout this project

**Table of Contents**

[1 Summary 7](#_Toc134012360)

[2 Poster 9](#_Toc134012361)

[3 Introduction 10](#_Toc134012362)

[4 Technologies 11](#_Toc134012363)

[4.1 Google Colab 11](#_Toc134012364)

[4.2 TensorFlow 11](#_Toc134012365)

[4.3 Flask 11](#_Toc134012366)

[4.4 Npm 11](#_Toc134012367)

[4.5 React Native 11](#_Toc134012368)

[4.6 Expo 11](#_Toc134012369)

[4.7 Keras 12](#_Toc134012370)

[4.8 AWS 12](#_Toc134012371)

[5 Project Architecture 13](#_Toc134012372)

[5.1 Mobile App 13](#_Toc134012373)

[5.2 Flask Server 13](#_Toc134012374)

[5.3 AWS 13](#_Toc134012375)

[6 Project Plan 14](#_Toc134012376)

[7 Convolution Neural Network 15](#_Toc134012377)

[7.1 Dataset and Pre-processing 15](#_Toc134012378)

[7.2 Architecture and Tuning 16](#_Toc134012379)

[7.3 Keras Tuner 16](#_Toc134012380)

[7.4 Model Compilation 16](#_Toc134012381)

[7.5 Training and Validation 16](#_Toc134012382)

[7.6 Evaluation 17](#_Toc134012383)

[8 Mobile App 18](#_Toc134012384)

[8.1 Navigation 18](#_Toc134012385)

[8.1.1 Navigation Drawer 18](#_Toc134012386)

[8.2 Components 19](#_Toc134012387)

[8.2.1 Hamburger Button 19](#_Toc134012388)

[8.2.2 Image Picker 19](#_Toc134012389)

[8.2.3 Sign Image Gridview 21](#_Toc134012390)

[8.3 Screens 23](#_Toc134012391)

[8.3.1 Home Screen 23](#_Toc134012392)

[8.3.2 Sign Screen 23](#_Toc134012393)

[9 Sever 23](#_Toc134012394)

[9.1 Setup & Requirements 23](#_Toc134012395)

[9.2 Loading the Model 23](#_Toc134012396)

[9.3 API 24](#_Toc134012397)

[9.4 Prediction 24](#_Toc134012398)

[10 Data Set 24](#_Toc134012399)

[10.1 Creating a Dataset 24](#_Toc134012400)

[10.2 Processing the Images 25](#_Toc134012401)

[10.3 Where to improve. 25](#_Toc134012402)

[11 Ethics 26](#_Toc134012403)

[12 Conclusion 26](#_Toc134012404)

[13 Bibliography 26](#_Toc134012405)

[14 Git Hub Code 28](#_Toc134012406)

# Summary

The goal of ISLT (Irish Sign Language Translator) was to develop a mobile app which would better the lives of individuals in the deaf community. It would do this by reading in an image from the user’s mobile phone and sending the image to a server where a CNN model would predict the image read in and send the prediction back to the user and tell the user what the sign is.

The project’s scope focused primarily on the image prediction model which is a Convolutional Neural Network. With this model, the app will be able to predict the most likely sign that the user is trying to read.

The project began with researching different machine learning technologies. This then lead me to explore different datasets and even attempted to try and create my own as I could not find a dataset for Irish Sign Language(ISL). After these steps, the process of creating the mobile app began followed by creating and training the model. Then a server was created to run the training model on which the mobile app could send the image to and receive the response of what the sign means.

Unfortunately, due to the limited time and the unavailability of an ISL dataset, I was only able to create a proof-of-concept demonstration using a tensor flow rock paper scissors dataset [1]. The Mobile App is fully functional, and its appearance is very minimalist and clean.

During the development of this, I gathered a greater understanding of the areas related to my project such as Image Processing, Deep Learning especially Convolution Neural Networks, deploying a Flask server also the creation and deployment of a React Native App. The Project has a huge opportunity for growth which will be pursued.

This is a final year project for the Bachelor of Engineering Honors in Software and Electronic Engineering at the Atlantic Technological University. This course includes a study of Cloud Computing, Mobile App Development, Digital Signal Processing, C++, Java, Continuous Integration and Continues Development, Embedded Real-Time Operation, Enterprise and Innovation and Medical Electronics.

# Poster

Graphical user interface, application

Description automatically generated

Figure 2‑ Project Poster

# Introduction

The way we communicate is changing. Communicating with people who speak different languages is becoming easier and easier. This is due to apps and webpages like Google Translate. Unfortunately, 45,000 people in Ireland use a language that cannot be translated. With this in mind, I decided to make ISLT my final year project. After speaking to a member who is part of this community, I began creating the mobile app for ISLT.

ISLT aims to help the communication between the deaf community and those who do not know ISL The project required creating a mobile app using the camera on the mobile phone, creating a Machine Learning (ML) model to read the image and converting the predictions to text. Create a server that the ML agent can run on and send and receive the data between the server and the app. The main feature of ISLT is the ML agent it is using a Convolutional Neural Network (CNN). CNN is an ML model that processes images by using convolution layers to detect local patterns.

ISLT has a second page which is designed to help the user learn ISL. This page has a selection of images of Irish sings which the user can learn. This is done by connecting to an Amazon Web Service (AWS) S3 bucket which stores the image.

# Technologies

## Google Colab

Google Colab is a free cloud-based service that allows you to run Jupyter Notebooks. It gives access to powerful computing resources, allowing the machine-learning model to train quicker. Also being a cloud-based environment, the notebook could be accessed anywhere.

## TensorFlow

TensorFlow is an open-source machine-learning library developed by Google. It provides a flexible and efficient platform for building, training and deploying machine learning models. It is widely used in the deep learning field mainly for neural networking.

## Flask

Flask is a web application framework written in Python [2]. It is used to create a simple API for servicing predictions from the machine learning model. The Flask server receives an image and returns the predicted labels.

## Npm

Node Package Manager is a package manager and registry for JavaScript and node.js projects. It provides a convenient way to manage dependencies. Npm is used to download, install and manage packages that are used in ISLT.

## React Native

React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces. [3]

## Expo

Expo is a set of tools and services built around react native to help you build and deploy a mobile app. It simplifies the development process of building an app by having several different libraries and tools.

## Keras

Keras is a high-level deep-learning library for Python that simplifies the process of building, training, and evaluating neural networks. It’s built on TensorFlow(see 4.2). Keras provides a flexible interface for constructing complex deep learning models such as a CNN model.

## AWS

Amazon Web Service (AWS) [4] is a comprehensive cloud computing platform provided by Amazon. It offers a wide range of cloud-based services and infrastructure. In this project, we use Amazon S3 (bucket) which is an object storage device for storing and retrieving data.

# Project Architecture

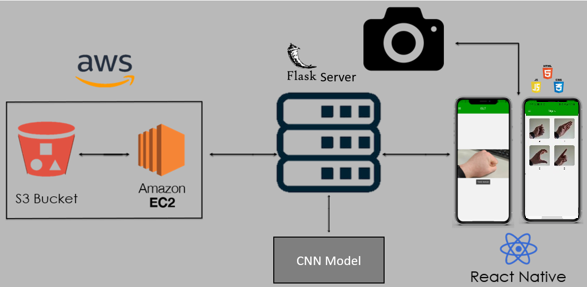


Figure 5‑ Architecture Diagram

## Mobile App

The mobile app in ISLT is built using React Native which accesses the phone’s camera roll which selects an image on the home screen. The app will then send the image to the Flask server. The Sign screen connects to the S3 bucket and displays different images of ISL signs.

## Flask Server

The Flask server receives the image from the mobile app. It applies the CNN model to the image and sends back the prediction to be viewed on the app.

## AWS

The AWS bucket stores the ISL images so that the app does not need to. The images are stored inside an S3 bucket called isltbucket. The app calls these images down using an access key and the bucket and path to the images in the bucket and this then gets displayed on the signs page of the app.

# Project Plan

Jira was used as the project management tool for this project. It was updated regularly and helped with breaking down big tasks into smaller doable tasks. Jira was chosen as according to my research it was the management tool as it was the main tool used in industry and would be vital to get experience with it. Tasks were created each week after every stand-up and would try to get completed for the next stand-up. When I task was not completed it would then be added to the next sprint.

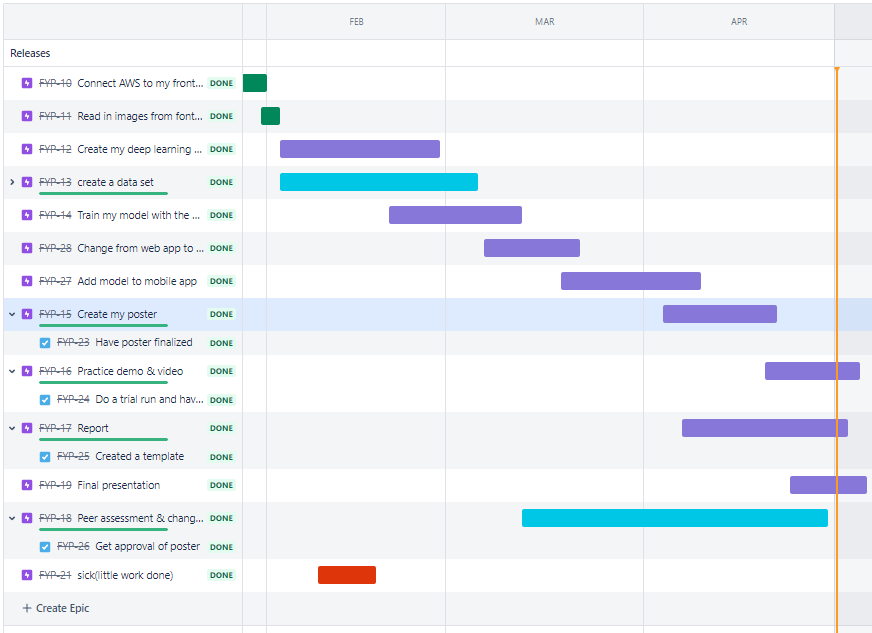


Figure 6‑ Project Timeline

# Convolution Neural Network

CNN is a class of artificial neural networks most commonly applied to analyse visual imagery [5]. It is a type of deep learning algorithm. Convolutional layers are used by CNNs to identify local patterns in the input data and to capture spatial elements such as edges, forms, and textures. These networks are often combined with pooling layers to reduce dimensionality, enabling the model to recognize increasingly complex patterns as data passes through the layers. The Tensor Flow Rock Paper Scissors [1] dataset is used for this project. The model is created in Google Colab using a Jupiter notebook. Google Colab was a good choice for this notebook as it allowed free access to powerful computing resources, allowing my model to train quicker. Also being a cloud-based environment, the notebook could be accessed anywhere which made working on it a lot easier.

## Dataset and Pre-processing

Pre-processing the dataset is crucial as it prepares the data for effective and efficient training. In this project, we are using the TensorFlow Dataset Library which simplifies the process of obtaining the data. The dataset is split between a training set and a testing set.

ds\_train = tfds.load(name="rock\_paper\_scissors", split="train")

ds\_test = tfds.load(name="rock\_paper\_scissors", split="test")

The images and labels are then stored in their NumPy arrays. The images are then normalized to the range of 0 to 1. Normalizing the pixel values helps improve the model's training process by reducing the variations in the input data and speeding up convergence.

train\_images = train\_images.reshape(2520, 300, 300, 1).astype('float32') / 255

test\_images = test\_images.reshape(372, 300, 300, 1).astype('float32') / 255

The images are also reshaped to match what is expected by the CNN model. They are reshaped to a 4D tensor with dimensions(number of samples. Height, width, channels). As the dataset contains greyscale images there is only one channel.

## Architecture and Tuning

The Project model is built using various layers, including AvergaePooling2D, Conv2D, BatchNormalization, MaxPool2D, Dropout, Flatten, and Dense. Each layer has its own purpose. The AvergaePooling2D layer is used to first reduce the input image dimensions, followed by several Conv2D and MaxPool2D layers for feature extraction. BatchNormalization is used to improve convergence during training. The Dropout layer is used to prevent overfitting which can occur from portions of neurons randomly dropping. The Flatten layer just converts the 3D map into a 1D Vector which is passed through the Dense layer.

## Keras Tuner

Keras tuner [6]is used to find the optimal hyperparameters for the model, such as the number of convolutional layers, the number of filters in each layer and the size of the dense layer.

tuner = RandomSearch(build\_model, objective='val\_accuracy', max\_trials=32)

tuner.search(train\_images, train\_labels, validation\_data=(test\_images, test\_labels), epochs=10, batch\_size=32)

The tuner searches by training the model with different hyperparameters and selecting the best based on the validation data.

## Model Compilation

The Adam optimizer [7] which is an adaptive learning rate optimization algorithm, is used for compiling the model. This helps the model converge faster and achieve better performance. The loss function used is SparseCategoricalCrossentropy as it can be used for multiclass classification. The evaluation metric is accuracy which will measure how accurate the predictions made by the model are.

model.compile(optimizer=Adam(), loss=SparseCategoricalCrossentropy(),

                  metrics=['accuracy'])

## Training and Validation

The model is fitted using the training dataset while the performance is evaluated by the validation dataset. The epochs and batch size are set during the training this controls the number of iterations through the dataset and the number of samples processed.

## Evaluation

After the training is complete the best model is evaluated on the test dataset to see how it performs on unseen data. The final architecture is displayed including the configuration of each layer and the number of trainable parameters.

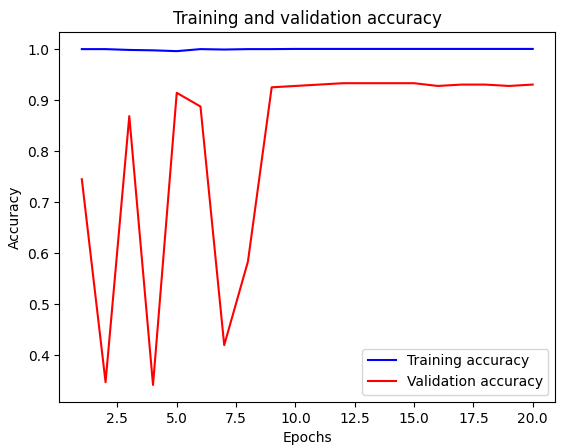
best\_model = tuner.get\_best\_models()[0]

best\_model.evaluate(test\_images, test\_labels)

best\_model.summary()

This model is then saved as a .h5 file so that it can be loaded later.

best\_model.save(“Location/to/save/file”+”.h5”)



This graph is generated using the Matplot library. It shows the trained model running against unseen data the model has high rising and falling peaks up to the 8 epochs until it stabilises at around 90% accuracy.

The fluctuation at the initial epochs can be due to several factors. This can be because the learning rate could be too high may cause the model to overshoot the weights. The model could also not have learned all the patterns in the data which shows why CNN is such a good model as it can continue to learn even after testing.

# Mobile App

The mobile app was built using Expo [8], which is a powerful platform for developing, building and deploying React Native Applications. As an expo App, it allows deployment for both Android and IOS platforms without the need for platform-specific code. The app allows the user to select an image from their device, predict the hand sign in the image and display the recognized sign. It uses a custom-trained deep learning model to classify the image between Rock, Paper or Scissors. The app consists of two main screens the home screen which the user selects the image and also displays the predication and the Sign Screen which will show the user different signs as a learning tool. The app uses a navigation drawer to change between the two screens.

## Navigation

The app uses React Navigation [9], a popular library for organizing and managing screen navigation. It provided a seamless user experience and helped structure the layout of the app. We utilized both createStackNavigator and createDrawerNavigator from the React library to create a hybrid navigation structure.

### Navigation Drawer

The app has a Draw Navigator that makes it simple to navigate between the ISLT home screen and the Sign screen. By using a hamburger button [10] in the header or swiping from the left side the user can quickly transition between the two pages.

The createStackNavigator allows the header and screen transitions in a stack-like manner while maintaining the drawer functionality. With this hybrid, the app has the ease of a drawer navigator and the versatility of a stack navigator. To customize the appearance and behaviour of the drawer ‘DrawerContentScrollView’, ’DrawerItemList’ and ‘DrawerItem’ components are used from the React Navigation library [9]. This Custom drawer content provides a consistent look and feel throughout the app.

## Components

ISLTs interface is built using various custom components. These include ‘HanbugerButton’, ‘ImagePicker’, and ‘SignImageGridView’. The hamburger component displays the hamburger icon and handles the user’s interaction to open the drawer. The Image Picker component allows the user to select images from their device gallery and processes them for prediction. The Sign Image Gridview component presents a visually appealing grid layout of images with their respective labels. Altogether this provides the user with a smooth and engaging experience.

### Hamburger Button

The hamburger button allows the user to open the drawer menu. This component uses the ‘Ionics’ icon library [11] and is integrated into the navigation header using the ‘onPress’ event handler.

|  |
| --- |
| <TouchableOpacity onPress={onPress}> |
|  | |
|  | |
|  |  | |

When pressed, it triggers the drawer menu’s opening, allowing the user to access the other screen.

### Image Picker

The Image Picker allows the user to select images from their device’s gallery using the expo-image-picker library [12]. This uses this library to access the device’s camera roll and display the selected image on the screen. This component also contains a camera and camera roll access permission handler.

const verifyPermissions = async () => {

    const result = await Permissions.askAsync(

      Permissions.CAMERA\_ROLL,

      Permissions.CAMERA

    );

    if (result.status !== 'granted') {

      Alert.alert(

        'Insufficient permissions!',

        'You need to grant camera permissions to use this app.',

        [{ text: 'OK' }]

      );

      return false;

    }

    return true;

  };

If the app does not have permissions, it will make an alert and request the permissions. When attempting to access the camera roll the verifyPermissions function is checked each time this is for the user’s protection.

Once an image is selected it is passed to the server so that the ML agent can make a prediction(see section 9). This function resizes the image using the expo-image-manipulation library. This is important as the model expects an image of 300x300 and will not be able to predict the image if it is a different size. The image is then converted into base 64 format so that it can be transmitted to the server as a type json file. A POST request is then made to the server containing the base64 encoded image. The server’s response to the predicted image is then displayed for the user to view on the screen.

const predict = async (imageUri) => {

    const resizedImage = await ImageManipulator.manipulateAsync(

      imageUri,

      [{ resize: { width: 300, height: 300 } }],

      { format: ImageManipulator.SaveFormat.JPEG, compress: 1, base64: true }

    );

    const base64Content = resizedImage.base64;

    const imagePayload = {

      data: base64Content,

    };

    try {

      const response = await fetch('http://192.168.1.9:5000/predict', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify(imagePayload),

      });

      if (!response.ok) {

        console.error(`Server responded with status ${response.status}`);

        const text = await response.text();

        console.error(`Server response: ${text}`);

        return null;

      }

      const result = await response.json();

      console.log(result);

      return result.label;

    } catch (error) {

      console.error("Error in predict function: ", error);

      return null;

    }

  };

### Sign Image Gridview

This component displays a grid of images with their respective labels. It uses the ‘Flatlist’ components from React Native [9] to create a responsive grid layout. The purpose of this component is to showcase a set of images of ISL in an organized and visually appealing.

The data array contains the image data to be displayed in the grid as seen below.

const data = [

  { id: 1, imageKey: 'ISL\_pic/A\_isl.jpg', text: 'A' },

  { id: 2, imageKey: 'ISL\_pic/B\_isl.jpg', text: 'B' },

  { id: 3, imageKey: 'ISL\_pic/C\_isl.jpg', text: 'C' },

  { id: 4, imageKey: 'ISL\_pic/D\_isl.jpg', text: 'D' },

];

The id is a unique identifier, and the imageKey presents the path to the image which is stored in AWS.

The GridItem is a separate functional component created for each item in the grid. It handles fetching the URL of the image from the AWS Bucket using the getSignedURL.

const GridItem = ({ item }) => {

  const [imageUrl, setImageUrl] = useState(null);

  useEffect(() => {

    const fetchImage = async () => {

      const signedUrl = await getSignedUrl(item.imageKey);

      setImageUrl(signedUrl);

    };

    fetchImage();

  }, [item.imageKey]);

The getSignedUrl function can be seen below this function sets the parameters to pull the image from the S3 bucket. It is done by setting the bucket name which will match with the name on the S3 bucket. The imageKey is the path to the individual image inside the bucket.

const getSignedUrl = async (imageKey) => {

  const s3 = new AWS.S3();

  const params = {

    Bucket: 'isltbucket',

    Key: imageKey,

    Expires: 60 \* 5,

  };

The size of the grid can be adjusted by modifying the ‘FlatList’.

<FlatList

      data={data}

      renderItem={({ item }) => <GridItem item={item} />}

      keyExtractor={item => item.id.toString()}

      numColumns={2} // You can adjust the number of columns here

      contentContainerStyle={styles.container}

    />

## Screens

Users can interact with the application, see the photographs, and use the image recognition feature on ISLT’s two main screens. These screens are designed with the user’s experience in mind and provide a smooth navigation flow between the different features of the app.

### Home Screen

The Home Screen is the initial screen of the app where users can access the image recognition feature. The primary function of this screen is to allow users to select an image from their camera roll and view the prediction of what sign the image is.

### Image previewSign Screen

The Sign Screen was designed as a learning tool so the user can view different signs in a clear and easy-to-understand format.

# Sever

The server plays a crucial role in ISLT. It is specially designed for image recognition in the context of ISLT. The server is a Flask web server responsible for handling the incoming image data from the app, processing the data, and using the pre-trained model to make predictions. The server uses API endpoints for seamless interactions between the client app and the server, allowing the clients to obtain real-time predictions.

## Setup & Requirements

To set up the server I use the Spyder IDE [13]. This is a very popular development environment and is included with the Anaconda distribution. Anaconda [14] is a powerful package and environment manager for Python and R programming languages. It comes pre-installed in Spyder. Open the server file in Spyder in the console you will need to install the required packages i.e., Flask, TensorFlow and any others. Once completed run the code by pressing f5.

## Loading the Model

To Load the pre-trained model you will have to update the following line of code:

model = keras.models.load\_model('Where/ever/you/saved/your/model/model.h5')

## API

An API route is defined with the endpoint ‘/predict’. This route accepts posts when the endpoint is called.

## Prediction

The predict function extracts the base 64 image data from the request. It then decodes the base64 image which returns the binary image data.

image\_data = request.json['data']

decoded\_image\_data = base64.b64decode(image\_data)

The image binary is then converted into an image object this is done using the PIL library. The image is then converted to greyscale using the .convert(‘L’) method and resized into what the model expects a (300,300-pixel image.

image = Image.open(BytesIO(decoded\_image\_data)).convert('L')

image = image.resize((300, 300))

The image is then converted into a NumPy array and passed into the model which makes its prediction. This then saves the prediction as a Python int. The class\_name which is the label of the predicted image is then sent as a json response to the app as the prediction.

# Data Set

This project faced a challenge because there was no readily available data set for ISL, so we attempted to make one from scratch. To complete this work, we had to learn how to establish a dataset, process the photographs, and make sure that each image was crystal clear and followed a consistent format.

## Creating a Dataset

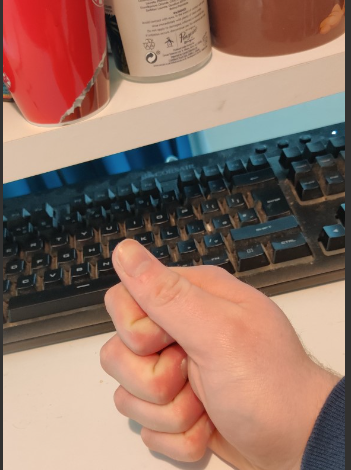
To create the dataset, we first researched various signs and gestures for ISL. The letters A, B and C were decided to be used for the beginning. Then images were gathered of each sign from a variety of angles and backgrounds.

## Processing the Images

Once the images were collected, they were processed to create a uniform dataset. This involved resizing the image so that all images are the same dimensions. This is important so that the model has consistent input data for the model. Then grey scaling was attempted on the images this is important as it reduces the complexity of the image and puts more focus on the hand gesture. Then removing the background was attempted this is done to reduce the noise of the image for the model.

## Where to improve.

Since I was creating the dataset alone it was a very small dataset. This means the model would not be extremely accurate. The backgrounds for the image were very inconsistent which made the removal of the background extremely difficult. Also, some of the images had awkward shadowing which made the greys scaling very inconsistent. Although this dataset did not make it into the project it was an invaluable learning experience and I know where to improve next time. Example from my dataset as of A in ISL before and after grey scaling below.

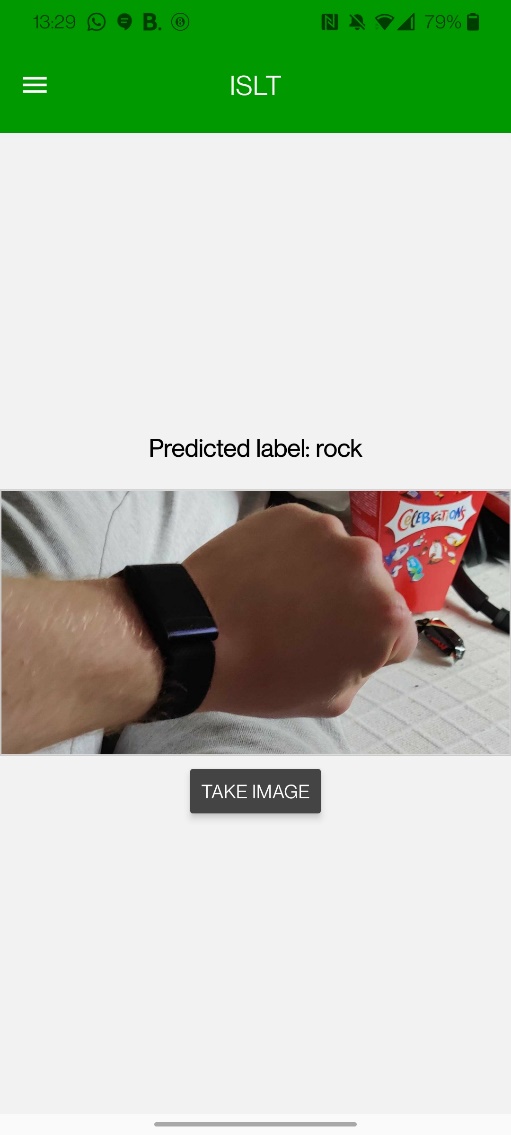
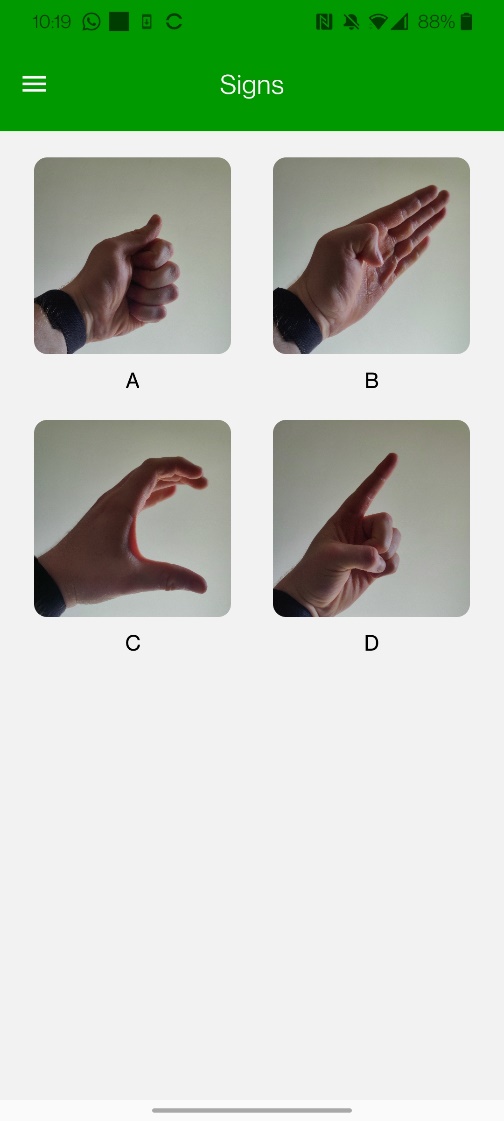
# Ethics

Ethics is an essential aspect of any project or profession. It’s the principles that guide us on our path to making responsible and moral decisions while working on projects and interacting with others. I believe honesty and integrity are hugely import in one’s work. Not only concerning others’ intellectual property but also one’s boss or team members with what they are capable of. I also believe that one should thrive to deliver the highest quality of work they can by following the best practise of the industry.

# Conclusion

During the development of ISLT, I gathered a greater understanding of the areas related to my project such as Image Processing, Deep Learning especially Convolution Neural Networks, building and creating a dataset, deploying a Flask server also the creation and deployment of a React Native App. The Project has a huge opportunity for growth which will be pursued.

ISLT is a fully functional mobile app which detects where the image inputted is Rock, Paper or Scissors. The image is then sent to a server where the prediction is calculated and sent back to be displayed for the user. Although the original plan of the project was to be able to read Irish Sign Language I believe Rock paper scissors are a good alternate and will be easily transferable once there is an ISL dataset created.



# Bibliography

|  |  |
| --- | --- |
| [1] | TensorFlow, "Rock Paper Scissors dataset," TensorFlow, [Online]. Available: https://www.tensorflow.org/datasets/catalog/rock\_paper\_scissors. [Accessed 30 01 2023]. |
| [2] | pythonbasics, "what is flask python," [Online]. Available: https://pythonbasics.org/what-is-flask-python/. |
| [3] | R. Native, "React Native Webpage," [Online]. Available: https://reactnative.dev/. |
| [4] | Amazon, "AWS," [Online]. Available: https://aws.amazon.com/free/?trk=d5254134-67ca-4a35-91cc-77868c97eedd&sc\_channel=ps&ef\_id=CjwKCAjwjMiiBhA4EiwAZe6jQ8aZuRd-mbYaFWup\_ZuAvmKGmDpJqpJHQQUa32gpRL59OeRCPd3F-xoCO2EQAvD\_BwE:G:s&s\_kwcid=AL!4422!3!433803620858!e!!g!!aws!1680401428!67152600164&all-f. |
| [5] | "CNN WIKI," [Online]. Available: https://en.wikipedia.org/wiki/Convolutional\_neural\_network. |
| [6] | Keras, "Keras Tuner," [Online]. Available: https://keras.io/keras\_tuner/. |
| [7] | Keras, “Keras Adam Optimizer,” [Online]. Available: https://keras.io/api/optimizers/adam/#:~:text=Adam%20optimization%20is%20a%20stochastic,order%20and%20second%2Dorder%20moments.. |
| [8] | expo, "expo website," [Online]. Available: https://expo.dev/?utm\_source=google&utm\_medium=cpc&utm\_content=search&gclid=CjwKCAjwxr2iBhBJEiwAdXECw6-k3Q0tUkkX7vMpna-lx\_BQn1EyZBaH0sw9N73UzW1-IXuDwX06\_xoCY8UQAvD\_BwE. |
| [9] | R. Navigation, "GitHub for React Navigation," [Online]. Available: https://github.com/react-navigation/react-navigation/tree/main/example. |
| [10] | K. Balasubramanian, "Medium Hamburger button," [Online]. Available: https://medium.com/timeless/building-a-hamburger-menu-interaction-in-react-native-1c9012396d6b. |
| [11] | Ionic, "Ionic Icons," [Online]. Available: https://ionic.io/ionicons. |
| [12] | Expo, "Expo image picker lib," [Online]. Available: https://docs.expo.dev/versions/latest/sdk/imagepicker/. |
| [13] | S. IDE, "Spider IDE," [Online]. Available: https://www.spyder-ide.org/. |
| [14] | Anaconda, "Anaconda Website," [Online]. Available: https://www.anaconda.com/. |

# Git Hub Code

<https://github.com/DCoen23/FYP-ISLT>

Qr code

Description automatically generated