# Chapter 1: Introduction

## 1.1 Overview

This chapter will introduce the background and concepts regarding the project. That includes proposed work, expectations and features of the final product. Besides that, the methodology and software tools adopted will also be discussed here.

## 1.2 Background of the Project

Transign, an artificial intelligence-based real-time sign language to speech translation application, has the main features of detecting hand signs and then converting it them to speech.

This project aims to ease the difficulty of communication for deaf/mute people in everyday life as the skillset for understanding sign language is a skill rarely found among people within society in the current times. ASL (American Sign Language) will be the go-to sign language for this project due to its popularity. The project will go through a few phases to accomplish this objective. Firstly, custom data of hand signs are produced through a webcam. Following that, the dataset is cleaned and processed before it is fed to the model for training. Once a feasible model has been produced, the inference will go through a few confidence tests before it is passed as legible sign language and then converted to text for speech synthesizing. The resulting translation can also be translated to other spoken languages to further bridge the gap of communication between deaf/mute people and other ethnics around the world.

## 1.3 Key Concepts

At the base of this system is the hardware which will be the mobile handheld devices used daily by almost everyone known as smartphones. A device can be considered a smartphone when it has the processing power and memory to run applications, make phone calls and text messages, access to the internet, sound input and output, and lastly camera capabilities(*What is a smartphone? | Digital Unite*, no date).

With that said, the system will be using Android-based smartphones. Android is an operating system for smartphones developed by Google and is the operating system of more than 70% mobile handheld devices worldwide to date(*Mobile Operating System Market Share Worldwide | StatCounter Global Stats*, no date).

Moving along, a few keywords must be known that will be used throughout this proposal. The first one being **image processing.** The definition of image processing is to manipulate images by processing it through computers(Young *et al.*, 2004). Images can be broken down into a matrix of precise numbers that can be quantized and manipulated to do tasks such as image enhancement, restoration, and analysis(da Silva and Mendonça, 2005). The system will be mainly focused on the enhancement and analysis aspect of image processing to process then extract useful data from the video feedback given through the smartphone camera.

Besides that is **machine learning,** itis the core of what people of the current era knows as artificial intelligence(Shalev-Shwartz and Ben-David, 2014). It has the power to learn autonomously without any manipulation from a human being through the data fed into it to look for similarities or patterns through computing power and models made by human beings. It uses what it has learned to make decisions when provided input and output a desirable outcome(Decencière *et al.*, 2013).

Moving along is **deep learning**, a subset of machine learning algorithms mainly inspired by the function of a human brain called artificial neural networks(Socher, Bengio and Manning, 2012). Deep learning models achieve immaculate accuracy and speed, sometimes better than human-levels when it comes to classification tasks directly from images, text, or sound. These models train through enormous sets of labeled data and neural network architectures containing many layers similar to a human brain(Perez and Wang, 2017).

Last of all we have **real-time object recognition,** which has the aim of producing decisions on the spot when given live data on-site like video from a camera feed(Chadalawada, no date). It requires a robust deep learning model that can process information as quickly and as accurately as possible due to the time-sensitive nature of tasks object recognition is used for such as autopilot systems(Gavrila and Philomin, 1999).

## 1.4 Methodology

For this system, the Prototyping Model is the methodology the system’s development will follow. This methodology is the most suitable for this project as the ideas behind this system are fairly new and never done before so requirements for it may change as functions are developed(Al-Husseini and Obaid, 2018).

Firstly, the system’s requirements will be analyzed by doing more research on what deaf and mute people would like to see in an application that will mainly affect them or be used by them. Not only that, the frameworks and models practical usefulness will also be analyzed in this phase.

After that, once all the requirements have been detailed and documented, a feasibility study will be done to see whether the project is feasible to be developed. An example of this is checking what Android devices have adequate resources to run this system once it is developed.

Continuing on, a feasibility study will be conducted when the first design of the system is created. This design is considered the first prototype and once it is created, it will go through an evaluation to see what other requirements or enhancements should be done. This will repeat until a satisfying product is created hence the name prototyping. The first prototype may be functional and able to recognize sign gestures but does not have a friendly UI to be used. It will go through the second prototyping to redesign the UI with its sign gesture recognition functions and be reviewed again(Carr, 1998).

Lastly, once a satisfying prototype has been created, it will go through final testing and maintenance before becoming a finalized product. This is where all the test code and straggling bugs are cleaned up so that the final product used does not have any problems(‘SDLC - Software Prototype Model’, no date).

## 1.5 Proposed Work

The proposed system is to create a mobile application in Android that will be able to recognize sign language gestures and translate it into speech. This application is mainly targeted to assist deaf and/or mute people with communicating with the general society when text or writing does not suffice.

As its nature states, this system will focus on Android handheld devices. This means that the application will only be able to run on devices that use the Android operating system. The device must have a camera which will not be a problem as the majority of Android devices nowadays come equipped with one.

The system will be designed with Kotlin, a language being pushed by the developers of Android, Google, as the new mainstay for Android application development(*Kotlin and Android  |  Android Developers*, no date). It will work in tandem with libraries such as OpenCV, TensorFlow, and Keras to recognize objects within the live video feed of the camera and train the data into suitable models.

The system will not have any processes that run in the background and will only be an application that can be run on the main thread. This will comfort the user’s privacy is maintained as the system does not have anything to do if it is not currently being used.

As far as cloud connectivity goes, the requirement for it will be decided as development goes during the prototyping. Whether the model will perform better as a live service on the cloud or run locally on the device will be analyzed and evaluated during the developing phase.

The system will have its model created by TensorFlow and Keras and then trained with data found on the web or data given to it through the OpenCV controlled camera video input from the Android device(Zebin, 2017). The system will learn to recognize the gestures through the model and link them to the correct translations.

There are a lot of basic American Sign Language datasets that can be found on Kaggle, an open-source dataset website, containing the datasets such as the alphabet and digit gestures which will become the foundation of the data used to train the model here(*Interpret Sign Language with Deep Learning*, no date). As for the image recognition techniques, TensorFlow has APIs with trained models to recognize objects and classify them into objects. For this system, once hands have been recognized then it can go deeper and look for what gestures are being performed to recognize what gestures are being made.

## 1.6 Summary

After identifying the aims and objectives for this proposed system, it is concluded that this system requires more research before implementation. All the other areas of research will be discussed in next chapters with proper justifications and examples.

# Chapter 2: Literature Review

## 2.1 Overview

In this chapter, three areas of research will be discussed that includes image recognition, machine learning and object detection. This analysis was aimed to help in studying all the methodology and complexity in the implementation of the proposed system. Each of them will be supported by at least two existing system literature.

## 2.2 Image Recognition

The purpose of this study is to analyse all the existing image recognition projects to get some understanding and insights about their approach and methodology.

### 2.2.1 Face Recognition

Another important area of research in image recognition is face detection system. The threshold function for a system will extract the information from the human face. The first existing system in this field was by Fares Jalled in 2017 who build a face recognition system using EigenFace which discriminating image input into multiple classes for person identification. The researcher’s proposed system structure starts from the image input to the pre-processing process of the image. In the image pre-processing stage, the image resolution, environment elimination, image rotation as well as illumination to the image were taken care of before the next step, feature extraction (Jalled, 2017). In the feature extraction, the system will retrieve all the distinctive elements from a person's face for classification purpose. The classification purpose will classify the person by their name and features on their face through the database. Another face recognition project considered for this area was from Priyanka Wagh and her colleague in 2017. This proposed system was aimed to use face recognition to help in taking attendance. This system achieves face recognition by setting up a static camera where the entire class is covered to periodically take a photo. The taken photos were converted to a grayscale image with histogram equalization to improve the contrast in the image (Wagh et al., 2015). For the student recognition in the classroom, Viola and Jones framework were used. Each recorded face was tested through EigenFace and Principal Component Analysis (PCA) for random value reduction as EigenFace often maximize variation so with PCA it helps the system to combine the variations based on a specific principle (Wagh et al., 2015).

### 2.2.2 Crop and Weeds Recognition

In the 1990s, a research was conducted in conjunction between related agricultural departments from the McGill University and Cornell University which was built on the rapid improvement of machine vision and image processing technology at the time to distinguish weeds from actual crops by leveraging the power of artificial neural networks (ANNs) (YANG, et al., 2000). The images were taken in bird’s-eye view at random locations within the fields of the in campus farms of McGill University with slightly varied zooms but similar aspect ratios. For training and testing purposes, common weeds were put into one category and separated from the target crop which in this case were corn plants. The images would be processed into 8-bit color bitmaps and then further cropped into 100x100 pixels from the original image’s resolution. This ensured that the crops and weeds were properly seen in each cropped section while optimizing it for processing within the ANN where PC memory was inadequate (YANG, et al., 2000). The images would then be fed into the ANN where the color index of a cluster of pixels would go through the processing elements (PEs) of the ANN and return 1 or 0s depending on whether a crop or weed was predicted for that cluster. The results from all cluster predictions would then be gathered and inferenced from a confidence of 0…1 to predict whether the plant seen within the cropped image is a crop or weed. Multiple configurations of PEs were tested for the recognition of crop and weeds which returned a noteworthy result. The ANN’s prediction result of weeds accuracy rose from 40 to 50% to 70 to 80% when 200 PEs instead of 160 were incorporated to its hidden layer but the accuracy for the prediction of corn plants decreased when more PEs were added onto the ANN’s hidden layer compared to the previous 200 PEs (YANG, et al., 2000).

## 2.3 Deep Learning Neural Network

The proposed project uses a specific neural network architecture specifically known as convolutional neural network (CNN).

CNNs primarily focus on input made up of images, ensuring that the architecture is set up best suited for dealing with that type of data. The reason for this is a key difference between CNNs and artificial neural networks (ANNs). The neurons within the layers of CNNs are organized in three dimensions comprising of spatial attributes such as height, width and depth (O'Shea & Nash, 2015). The depth mentioned in this case does not refer to the outstanding number of layers in the ANN but the third dimension of an activation volume. Compared to usual ANNs, the neurons within any given layer in the CNN only connect to a small region of the layer before it.

CNNs are made up of three forms of layers. These layers are known as convolutional layers, pooling layers, and fully-connected layers. A CNN architecture is formed when these layers are stacked together (O'Shea & Nash, 2015).

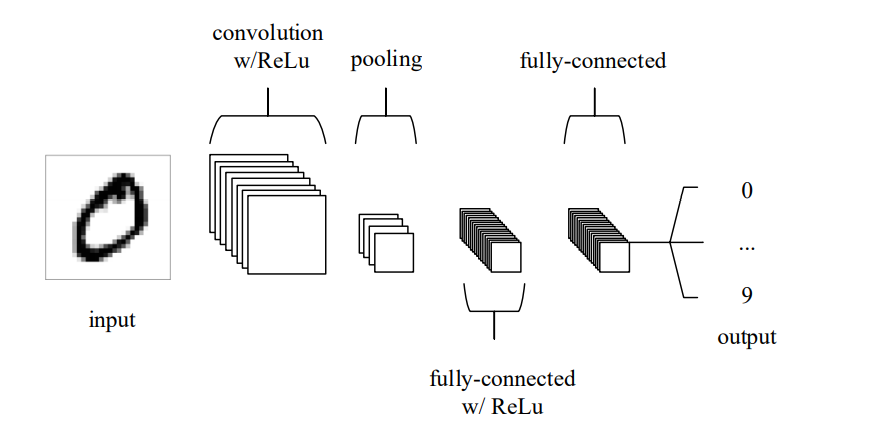


Figure Simple CNN architecture (O'Shea & Nash, 2015)

Seen in figure 1 above is a simple CNN architecture made of only 5 layers for MNIST classification. The input layer contains the pixel values of the image fed into the CNN. Following that, the convolutional layer produces the output of neurons that are joined to local regions of the input through calculating the scalar product between their weights and the region joined to the input volume. The rectified linear unit (ReLu) applies an ‘elementwise’ activation to the output produced by the previous layer. The pooling layer has the role of downsampling along the input’s spatial dimensionality, lowering the number of parameters within that activation. Lastly, the fully-connected layers perform similar roles as normal ANNs to produce class scores for classification (O'Shea & Nash, 2015).

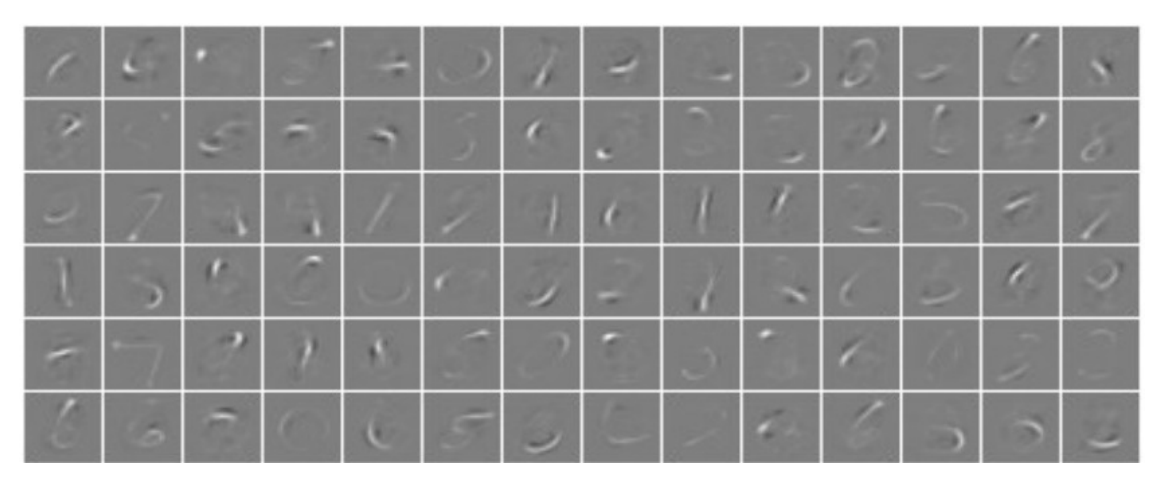


Figure Activations from convolutional layer (O'Shea & Nash, 2015)

As seen in figure 2 above, the network picks up on unique characteristics of specific numeric digits from MNIST’s database of handwritten digits.

## 2.4 Object detection

For object detection, there are a few algorithms that must be discussed and compared that are based on the CNN architecture mentioned in the previous subtopic. These algorithms will be discussed in the subtopic below.

### 2.4.1 Faster R-CNN

The R-CNN technique trains CNN end-to-end to do classification for proposal regions into object background or categories. R-CNN mostly act as a classifier, and the object bounds cannot not be predicted. The performance of the region proposal module defines the accuracy. Pierre Sermanet and his team proposed a paper in 2013 under the title of “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks” which shows the ways of using deep networks for predicting object bounding boxes (Sermanet et al., 2013). In the OverFeat method, a fully-connected layer is trained to predict the coordinates of the box for the localisation task that consider a single object. Then, the fully-connected layer is turned into a convolutional layer for multiple class object detection (Sermanet et al., 2013). Figure 3 shows the architecture of Faster R-CNN which is a single, unified network for object detection.

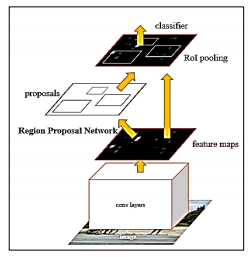


Figure architecture of Faster R-CNN (Sermanet et al., 2013)

Fast R-CNN implement end-to-end detector training on shared convolutional features and display decent accuracy and speed. On the other hand, R-CNN fails to do real time detection because of its two step architecture.

### 2.4.2 YOLO

YOLO stands for “You only look once”. According to Redmon (2016), it is an object detection algorithm that runs quicker than R-CNN because of its simpler architecture. Classification and bounding box regression will be done at the same time. Figure 4 shows how YOLO detection system works.

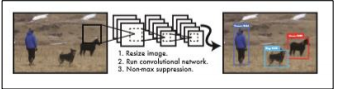


Figure YOLO Detection

A single convolutional network predicts several bounding boxes and class probabilities for those boxes simultaneously. YOLO trains on full images and the detection performance will be optimised directly. This ensures YOLO to run extremely fast, hence producing real-time predictions.

### 2.4.3 SSD

Lastly, we have SSD which stands for “Single shot detection”. SSD is another algorithm with similar performance metrics to YOLO where SSD wins in overall accuracy and speed but loses out in number of true positives and recall values (Morera, et al., 2020).

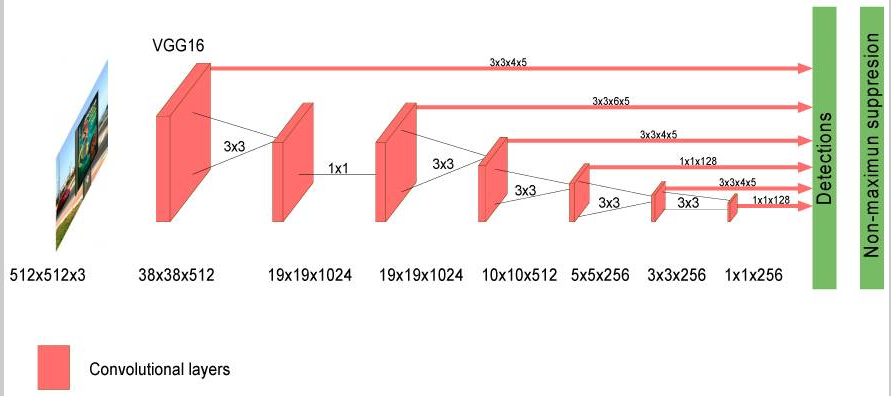


Figure SSD Algorithm (Morera, et al., 2020)

As seen in figure 5 above, a CNN based on SSD defines a domain of anchor boxes with varying sizes and aspect ratios at multiple locations of an image. The detection box prediction is then done based on these references. The SSD algorithm does not resample features for bounding box hypotheses unlike R-CNN and is compatible with real-time applications.

## 2.5 Summary

In conclusion, the CNN architecture is chosen due to its performance and compatibility with processing image data. Moving on, the CNN-based SSD algorithm will be used due to the cost-performance ratio in terms of computing resources, making it suitable for the purpose of the project without sacrificing the accuracy of the results produced.