System of sign language recognition for smart home application

Sofia Cerè; 0001057906
Laboratory of Making
Master's Degree in Computer Science, Department of Computer Science and Engineering
Alma Mater Studiorum - University of Bologna

***Abstract*—The goal of the project is to implement a recognition system for some elements of American Sign Language fingerspelling. To achieve the goal, a neural network model was developed and then loaded onto a microcontroller.**

***Index Terms*—Arduino, TinyML, TensorFlow Lite Micro, American Sign Language, Digital Accessibility, Making.**

I. Introduction

There are many smart home display, but none include sign language as a means of interaction.
The goal of the project is to realise a simple system of sign language recognition trough camera for smart home application. In particular, through the recognition of a letter "B" of sign language, a smart LED can be switched on.
The sign language used in the project is American Sign Language since several fingerspelling alphabet dataset are available for it.
American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from American English. ASL is expressed by movements of the hands and face.
Fingerspelling (or dactylology) is the representation of the letters of a writing system, and sometimes numeral systems, using only the hands.
The following report will describe in detail the project architecture (Section 2), the implementation carried out (Section 3) and the results obtained (Section 4).

II. Project's Architecture

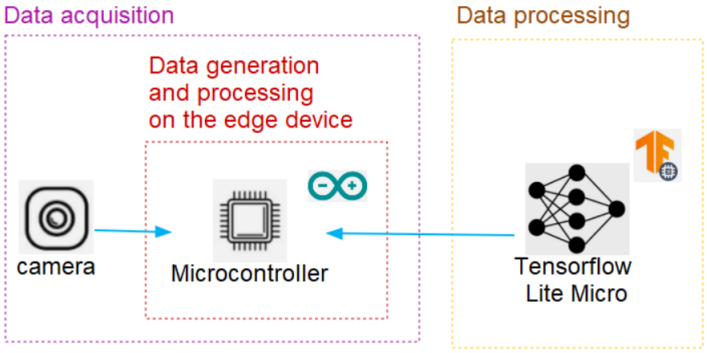

Fig. 1. Project Architecture

As shown in Figure 1, the project architecture consists of the use of a microcontroller with a camera, which is responsible for capturing frames, employed for data processing with the help of the neural network.

A. Hardware

Arduino Tiny Machine Learning Kit[1] composed by Arduino Nano 33 BLE Sense Lite, a camera module (OV7675) and custom Arduino shield were used for the project objective, i.e. use sign language for turn on LED.
It has an ARM Cortex M4 MCU running at 64MHz clock and 256 KB SRAM.

Fig. 2. Arduino Tiny Machine Learning Kit[1]

B. Software

TensorFlow [2] : used for developing neural network model for image classification;
TensorFlow Lite for Microcontrollers [3]: used to run neural network model on Arduino Nano 33 BLE Sense;
Language used:

- Python: for develope neural network model;
- Arduino language(C/C++): for programming Arduino microcontrollers.

III. Project's Implementation

A. Neural Network

Through the use of Edge Impulse platform[4]
, the neural network (NN) model was created, trained and tested. The neural network performs a two-class classification: sign and non-sign. To simplify the problem, when you identify the letter sign "B", you detects the sign. Whereas when one identifies the sign letter "A", one detects the non-sign.
The procedure of developing of the neural network follows these steps:

1) **Creation of training and validation dataset**
From the American Sign Language Letters Dataset[5] was extracted the group of images of letter ”A” and letter ”B” of sign language.

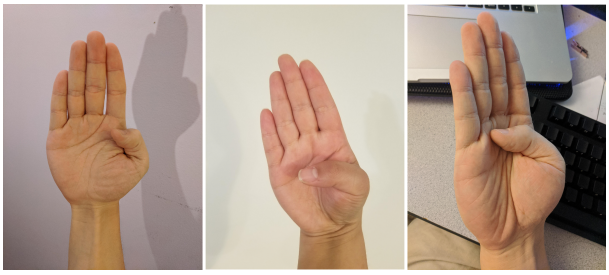

Fig. 3. Example of letter ”B” of ASL dataset

All images with size 96x96 in the dataset were converted to grayscale in order to further lighten the model.
Also some samples 4 by OV7675 camera module of microcontroller was added: The images in the dataset have a different background and brightness level: this increment the possibility of images that the Neural Network can be recognised.
A limitation of the dataset is that the samples used consist of photographs of a right hand, making it more challenging to recognize letters formed by a left-handed person’s hand.
In total , the dataset have 6,084 images that has been split in 80% for training set and 20% for validation set. Splitting a dataset into an 80/20 ratio for training and testing strikes a balance between providing ample data for model learning (80%) and ensuring rigorous evaluation on unseen data (20%). This approach helps prevent overfitting and facilitates the development of accurate and generalizable machine learning models.

2) **Creation of the model**
The model of NN is a model compatible with TensorFlow Lite Micro [6]
Since the neural network was intended to be installed in a microcontroller, it was decided to produce a low-complexity one in order to avoid memory overload and the possible consequent overheating of the device.
The second reason is that not all type of layers of Tensorflow was supported by microcontroller.
After several training sessions, an accuracy of 96.55% was achieved.

3) **Conversion in TensorFlow Lite for Microcontrollers Model**

Fig. 4. Example of letter ”B” of camera samples

TensorFlow Lite for Microcontrollers (TFLM) is designed to run machine learning models on microcontrollers and other devices with only a few kilobytes of memory.
For this reason, it was necessary to convert the model to TFLM.
TensorFlow Lite has two key features namely Converter and Interpreter. The TensorFlow Lite Converter uses the TensorFlow graph file or saved model to generate a TensorFlow Lite FlatBuffer based file which is then used by the TensorFlow Lite Interpreter for inference. The generated *.tflite* file after the conversion process is used at the client-side for an on-device inference. The entire process is shown in the diagram below:

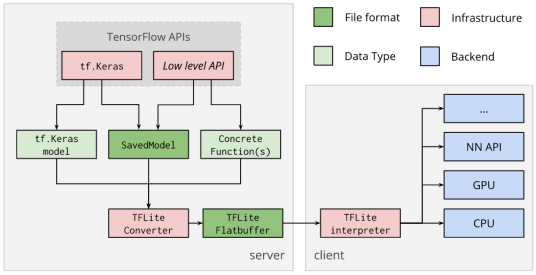

Fig. 5. TensorFlow Lite Converter and Interpreter [7]

The TensorFlow Lite Converter took a TensorFlow model and generated a TensorFlow Lite for Microcontroller model which was saved in the *.tflite* file.
After that, the TFLM model was tested in order to compare the result of prediction: it must be similar to TF model result of prediction.

4) **Export of TF Lite Model in cc format**
In order to upload the model in Arduino platform is necessary convert TF lite model in cc format so this is last steps done.
Post-training quantization is a conversion technique that can reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy. However, the choice was made to convert the TF lite model without quantization since the size compared to the TF lite model with quantization was not larger and since the library used in Arduino supported it.
The file of cc format model was inserted in the Arduino Sketch in order to start the recognition.

B. Arduino Sketch

The implementation of a model developed with Tensorflow technology turned out to be a trial-and-error process.
Unfortunately, as the Tensorflow_Lite library for Arduino is not supported to date, it was very difficult to create a model that could be compatible with the latest version of

the developed library. Furthermore, the aforementioned library does not support the OV765 camera that is present in the Arduino TinyML kit. Should the Tensorflow Lite libraries be supported again, the system could be updated accordingly.
For this reason, we had to resort to an additional library, namely Harvard_TinyMLx [8].
Specifically, we chose to use the sketch that implements a image classification, namely person detection. The sketch deals with either person detection or non-person detection.
In the case of the present project, the sketch was modified so that it could detect a sign and non-sign.
First, the camera is initialised to process images in grayscale. Once the camera is in action, the frames are captured and cropped to the size 96x96.
Subsequently, they are analysed by the TF Lite interpreter that runs the model.
When a valid gesture is detected, i.e. when it falls within the sign or non-sign class, the behaviour of the LED embedded within the microcontroller is executed accordingly.
The figure shows the workflow that the device performs:

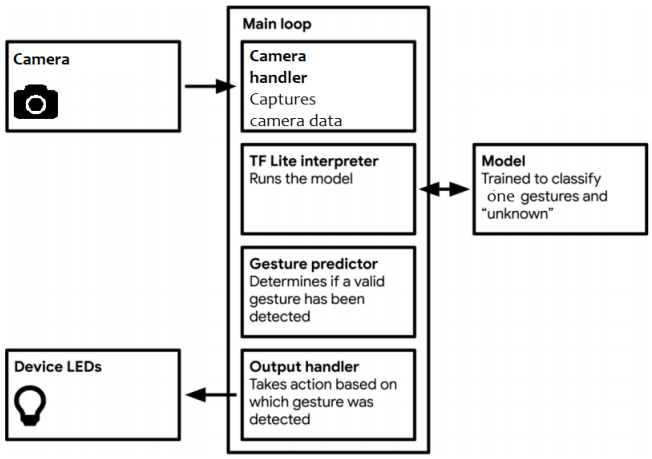

Fig. 6. Workflow of Arduino

IV. Results

In this project, I was able to develop a model that could be hosted by the Arduino 33 Nano BLE Sense microcontroller.
As explained in section III there were multiple limitations in creating a model that could be supported by the Arduino, and as a result a not very efficient system of recognition was developed because it cannot support a high degree of complexity.
In any case, the project produced good results as the accuracy of the neural network model is about 97% while maintaining a reduced size.
Possible extensions to the project could be improving accuracy of system of sign language recognition for smart home application could be to expand the image dataset used for training the neural network by also including images with letter from left hand and more different type of hand with various background and brightness. In the future, further neural network model structures could be tested for compatibility and, subsequently, the level of accuracy that can be achieved and the corresponding model size could be tested.

References

[1] Arduino Tiny Machine Learning Kit. url: https://store.arduino.cc/products/arduino-tiny-machine-learning-kit .
[2] TensorFlow. url: https://www.tensorflow.org/?hl=en.
[3] TensorFlow Lite for Microcontrollers. url: ttps://www.tensorflow.org/lite/microcontrollers?hl=en .
[4] Edge Impulse. url: https://edgeimpulse.com/ .
[5] David Lee. (2022). American Sign Language Letters Dataset.
[6] Pete Warden. Magic Wand example for TensorFlow Lite Micro. url: https://github.com/petewarden/magic_wand/
[7] Model conversion overview. url: https://www.tensorflow.org/lite/models/convert?hl=itfrom_model_training_to_device_deployment .
[8] Harvard_TinyMLx Arduino Library. url: https://github.com/tinyMLx/arduino-library .

V. License