University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Counter Detection with ESP32

Presenters : Dereck Alpizar, Neeranjan JK, Nisha Kumari

10. Januar 2023

10 Next Steps

Introduction I

- This project includes use of TinyML model on an ESP32 microcontroller to digitize an analog meter
- Goal is to use machine learning to accurately interpret the readings of the analog meter and display them digitally
- Useful in monitoring energy usage or measuring the output of a device
- ESP32 is well-suited for this task due to its low power consumption and powerful processing capabilities

Challenges I

- ESP32 used for image processing in ML might be short on resources and storage.
  - Adapting the model to work with the limited resources of the microcontroller, such as memory and processing power.
  - Optimizing the model for speed and low latency processing.
- ESP32 is designed to be low-power, but running ML models can still consume a lot of power. Need to consider appropriate power source.

Challenges II

- Data Acquisition and Monitoring.
  - Data Labelling and Collection (multiple analog meters) is a time consuming activity.
  - Data Transformation to be recognized by the ESP32 - ROI (Region Of Interest).
- Model training is complex and computationally intensive process, particularly if you are working with a large dataset.
- Troubleshooting and Code Updates after Deployment.

Hardware I

ESP32-CAM technical specifications:

ESP32-CAM is a small size, low power consumption camera module based on ESP32 microcontroller and OV2640 image sensor.

- 802.11b/g/n Wi-Fi SoC Module with a speed of 2.4 GHz.
- Bluetooth 4.2 with BLE.
- Low-power dual-core 32-bit CPU for application processing.
- Clock speed up to 160 MHz.
- Computing power goes up to 600 DMIPS.
- Built-in 520 KB SRAM plus 4 MB PSRAM.

Hardware II

ESP32-CAM technical specifications:

- Supports Wi-Fi Image Upload.
- Supports multiple sleep modes.
- Firmware Over the Air (FOTA) upgrades are possible.
- 9 General-Purpose Input/Output (GPIO) ports are available.
- Built-in Flash LED.
- Supports SD card.
- Has an onboard PCB antenna.
- Has embedded Free real-time operating system (FreeRTOS) and lightweight IP.

Hardware III

ESP32-CAM technical specifications:

ESP32 CASE:

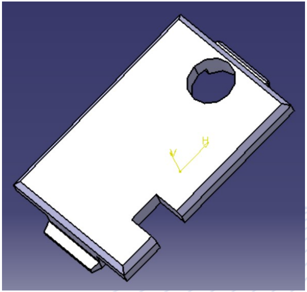

Abbildung: ESP32 Case Top Part

Hardware IV

ESP32-CAM technical specifications:

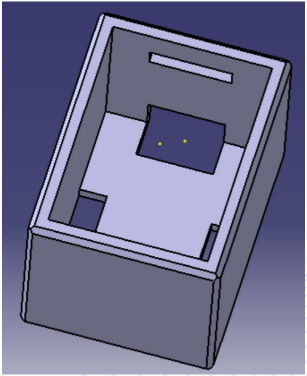

Abbildung: ESP32 Case Bottom Part

Software I

Visual Studio Code and Plug-in

Chosen IDE for the project is: VS Code:

- VS Code is a popular and widely used integrated development environment (IDE).
- VS Code has a number of features that make it particularly well-suited for developing TinyML projects:
  - Code completion and IntelliSense.
  - Debugging tools.
  - Source control integration.

Abbildung: Visual Studio Code Logo.

Software II

Visual Studio Code and Plug-in

VS Code is highly customizable and can be extended with a wide range of plugins and extensions **ESP-IDF**

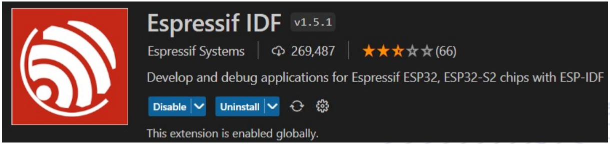

Abbildung: ESP-IDF Plug in in VS Code.

Algorithm I

Convolution Neural Network

- A CNN is a type of artificial neural network specifically designed to process data with a grid-like topology, such as an image.
- CNNs are preferred for image classification tasks because they are able to automatically learn and extract features from the input data, which makes them well-suited to tasks where the features of interest may not be easily defined by humans.
- In a digit recognition task, the input data is usually an image of a digit, and the goal is to classify the image into one of the 10 possible digits (0 through 9)

To perform this task, a CNN would typically take the following steps:

- Preprocessing: The input image is resized and possibly normalized to a standard size and format.
- Convolution: The input image is passed through one or more convolutional layers, which apply filters to the image and extract features.

Algorithm II

Convolution Neural Network

- Pooling: The output of the convolutional layers is passed through one or more pooling layers, which downsample the data and reduce the dimensions of the feature maps.
- Classification: The output of the pooling layers is passed through one or more fully connected layers, which use the extracted features to make a prediction about the digit in the image.

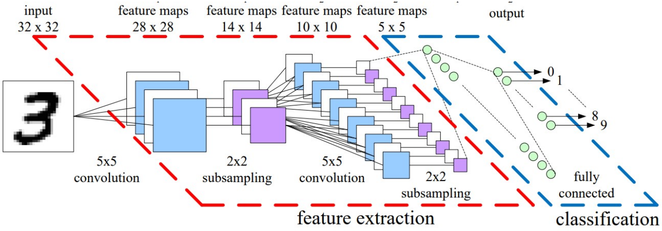

Abbildung: CNN architecture for digit recognition

Algorithm III

Convolution Neural Network

- The weights and biases of the CNN are learned through training, which involves showing the CNN a large number of labeled images of digits and adjusting the weights and biases to minimize the errors.
- Once trained, the CNN can then be used to classify new images of digits it has not seen before.

KDD Process Implementation I

- Collect a dataset of images of analog meters in various states.
- Preprocess by cropping and resizing the images
- Model training with preprocessed data
- Evaluate model to see how accurately it can interpret analog meter readings
- Model optimization to acheive desired level of accuracy
- Deploy the model on the device

Data Description I

The dataset of images for training the model have following details:

- Before data transformation:
  Image size : 703 bytes - 8KB
  Image dimensions : 32X59 - 37X67
  Color space : RGB
- After data transformation:
  Image size : 795 bytes - 923 bytes
  Image dimensions : 20X32
  Color space : RGB

Current Results I

Model Generation

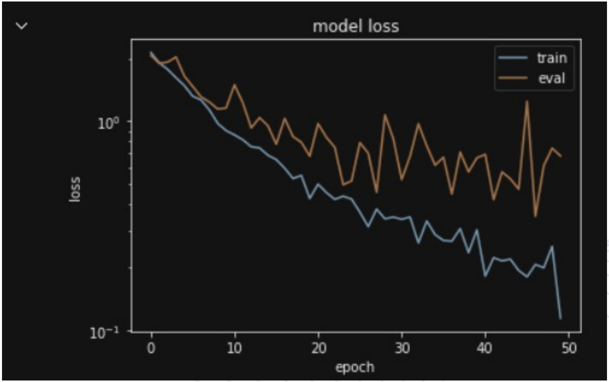

Abbildung: CNN Model

Current Results II

Model Generation

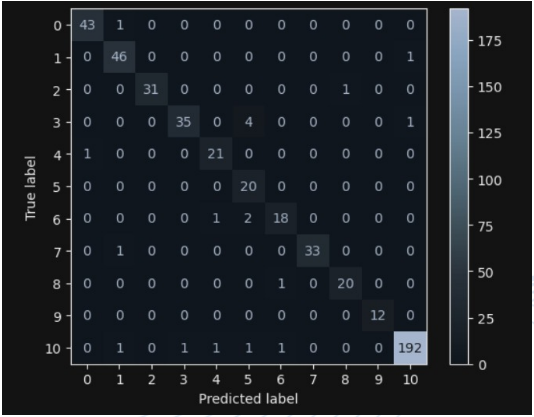

Abbildung: Confusion Matrix

Current Results I

TensorFlowLite File Generation

Abbildung: TnesorFlowLite Output file

Next Steps I

- Load the .tfl to ESP32-CAM.
- Test the model under proposed conditions.
- Put the ESP32 CAM in the case for user handling.

```
I (3590) BLINK: Blinken - start
Start Server ...
I (3610) server-main: Starting server on port: '80'
I (3610) server-main: Registering URI handlers
Load and initialize neural network ...

File does not exist.

Model file could not be loaded (/sdcard/dig-01.tfl).
I (13600) BLINK: Blinken - done
uri: /
1 uri: /, filename: , filepath: /sdcard
```

Abbildung: Current Error Message to debug.

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Thank You

for your time