Robust Magic Wand Project Documentation

Developer: Krishan Mohan Patel

patel@student.tugraz.at

Project Overview:

The Robust Magic Wand project aims to implement a versatile "Magic Wand" application capable of recognizing and classifying various wizard spells, charms, and curses. The magic wand utilizes an Arduino Nano 33 BLE Sense board with an Inertial Measurement Unit (IMU) for gesture recognition. The project comprises two main components: IMU Capture and Gesture Classification.

Spells, Charms, and Curses:

- The magic wand recognizes the following magical actions:
  - Alohomora Charm: Opens closed doors.
  - Avada Kedavra: Causes instantaneous death.
  - Locomotor Charm: Moves the target.
  - Arresto Momentum Charm: Slows or stops a target's velocity.
  - Revelio: Reveals hidden objects.

Assumptions:

- Magic wand length: ~30 cm.
- Hold wand with Arduino board facing up.
- IMU sensor sampling rate: 119 Hz.
- Utilize both accelerometer and gyroscope data (6 values per sample, Total 25 Samples).
- Extract useful features on Arduino.
- Consider useful data augmentations for improved model robustness.
- Experiment with different model architectures, considering Arduino limitations.
- Model training time on Colab should be under 3 minutes.
- Optimize the model's footprint through quantization and efficient layer usage.

Robustness Challenges:

- Recognizing spells is akin to handwriting, each person having a unique style.
- IMU sensors may not be identical, posing a robustness challenge.

- Address robustness issues considering in-distribution and out-of-distribution generalization.

Efficiency Measures:

- Fix seeds for reproducibility.
- Code compatibility for both Colab (model training) and Arduino (model inference).
- Provide information on the amount of data used, model size, FLOPs, and inference time.

What I am doing?

In this Project, I will show you how to:

- Use machine learning to recognize wand gestures.
- Build an Arduino-powered magic wand.
- Record your own training data.
- Train and deploy an ML model to recognize custom spell gestures.

Project Hardware Requirements:

- Arduino Nano 33 BLE Sense board.
- USB Cable.
- Magic wand should be ~30 cm long.

Project Software Requirements:

- Download the right version for your OS at Arduino Software.
- Run the Arduino IDE.
- Install the Board Package: Boards -> Boards Manager and search for "mbed" to install the "Arduino mbed-enabled Boards" package.

Testing the Setup:

To ensure your Arduino is connected correctly, go to File -> Example -> Basics -> Blink and try uploading a sketch that blinks an LED.

IMU Capture (Arduino Code)

- **Description**
    - The IMU Capture component is responsible for capturing acceleration and gyroscope data when significant motion is detected. It utilizes an Inertial Measurement Unit (IMU) to collect sensor data and prints the data in CSV format to the Serial Monitor.

- **Key Parameters**
  - **Acceleration Threshold: 2.5 G's**
  - **Number of Samples to Capture: 119**

```
  // Print the CSV header
  Serial.println("aX,aY,aZ,gX,gY,gZ");
}

void loop() {
  float aX, aY, aZ, gX, gY, gZ;

  // Wait for significant motion
  while (samplesRead == numSamples) {
    if (IMU.accelerationAvailable()) {
      // Read the acceleration data
      IMU.readAcceleration(aX, aY, aZ);

      // Sum up the absolutes
      float aSum = fabs(aX) + fabs(aY) + fabs(aZ);
```

- **Gesture Classification (Arduino Code)**

The Gesture Classification component reads the captured data, normalizes it, and feeds it into a TensorFlow Lite model for classification. The model predicts the gesture performed and outputs the result to the Serial Monitor. The project uses TensorFlow Lite for Microcontrollers, with model details and operations resolver included in the code.

```
// Create a static memory buffer for TFLM; the size may need to
// be adjusted based on the model you are using
constexpr int tensorArenaSize = 8 * 1024;
byte tensorArena[tensorArenaSize] __attribute__((aligned(16)));

// Array to map gesture index to a name
const char* GESTURES[] = {
  "alohomora",
  "avadakedavra",
  "locomotor",
  "revelio",
  "arrestomomentum"
};

#define NUM_GESTURES (sizeof(GESTURES) / sizeof(GESTURES[0]))
```

Model Architecture

The TensorFlow Lite model used for gesture classification is embedded in the Arduino code. It has been trained to recognize five gestures: "alohomora," "avadakedavra," "locomotor," "revelio," and "arrestomomentum." The model's architecture includes input normalization and an output layer with softmax activation for multi-class classification.

Data Augmentation

Data augmentation is applied to the original sensor data to increase the diversity of the training dataset. This technique aids in improving the model's ability to recognize gestures under various conditions.

Hyperparameter Tuning

The hyperparameters of the TensorFlow Lite model, such as the number of layers, units per layer, and activation functions, are predefined and embedded in the Arduino code. Fine-tuning hyperparameters for the specific use case may require adjusting these values and retraining the model.

Regularization and Dropout

- **Regularization and dropout are implemented in the model architecture to prevent overfitting:**
    - kernel_regularizer=tf.keras.regularizers.l2(0.01): L2 regularization on weights.
    - tf.keras.layers.Dropout(0.5): Dropout applied to input.

Project Limitations and Future Work

- The project assumes a predefined set of gestures, limiting its adaptability to new gestures.
- Real-world testing and validation are necessary to assess the model's performance in different environments.
- Model optimization for memory constraints on Arduino boards could be explored.
- Continuous improvement of the model through additional training data and retraining is recommended.

Conclusion

The Magic Wand Gesture Recognition project showcases the integration of an Arduino board, IMU sensor, and TensorFlow Lite for Microcontrollers to recognize gestures. The provided code serves as a foundation for further exploration, improvements, and adaptations to specific use cases.