Lab 3: Time Series Data and Edge Impulse

Introduction

In the first two labs, you created a pipeline from data to on-device execution using *TensorFlow*, *TensorFlow Lite*, and *TensorFlow Lite for Microcontroller*. Within these labs, you used image data and predefined datasets. While image data is common for machine learning applications, it is not the most common data type for low-power embedded machine learning applications. More commonly, you will use time series data, for example from inertial sensors/inertial measurement units (IMU).

IMUs can be found in many devices including motorcycles, aircraft, and drones. They are used for, i.a., navigation and vehicle tracking purposes but also for gesture recognition or Human Activity Recognition (HAR). IMUs consist of accelerometers and gyroscopes, with some including also a magnetometer. The Arduino we use for this course contains an IMU consisting of an accelerometer, a gyroscope, and a magnetometer.

In this lab, you will collect your own data, preprocess this data and build a system that is able to recognize at least three different gestures. For this lab, you will use Edge Impulse.

This lab contains quite some questions. We don't expect you to write long answers for them.

Submission Instructions

For the submission of this lab, please document what you did (please include screenshots) and answer the questions below. Please upload a PDF with your documentation and your answers to the questions, as well as your presentation slides when submitting this lab.

Part 1 – Setup

Task 1. Create a new project in Edge Impulse.

Task 2. Install the *Edge Impulse CLI* and *Arduino CLI*[1], connect the Arduino, and verify that your device is connected to Edge Impulse.

Part 2 – Data Collection

Task 3. Collect data for at least 3 motions. Your motions can be, i.a., *side movements*, *vertical movements*, *L-shaped movements*, *circular movements*, *diagonal movements*, *waving motion*, *snake motion on your desk*, etc. Collect at least twenty 10-second-samples per

[1]Installation instructions for the CLI tools: `https://docs.edgeimpulse.com/docs/development-platforms/officially-supported-mcu-targets/arduino-nano-33-ble-sense`

motion. Each sample of your motion should contain a periodically repeating pattern of the same motion. Make sure to perform variations on the motions. For example, perform both slow and fast movements and potentially vary the orientation of the board. You'll never know how your user will use the device. Only vary the orientation if you don't have two motions that otherwise look the same, e.g., don't do it if you chose both an *up-down* and a *left-right* motion.

Task 4. Look at your data. Do all of your samples for the same class look similar? If you have outliers, you might want to re-record that sample.

Question 1. Which features/patterns are representative for each of your motions?
Question 2. Is it sufficient to record data for your classes, or do you also need a baseline without any movement? Explain why you (don't) need idle data.

Task 5. If you need idle data, record the same amount of data without moving the device as for each of the classes.

Task 6. Split your data into *train* and *test* data. You can do that in the *danger zone* on your Dashboard. Check on your data acquisition page whether the train and test sets are balanced. If not, move samples between the sets to balance them.

Part 3 – Impulse Design

Task 7. Create an Impulse. Add one of each of the three processing blocks (*Spectral Analysis*, *Flatten*, and *Raw Data*) and one classification block. If you like, you can also test the *Spectrogram* processing block. **For this lab, we want to use only the accelerometer data.** Thus, deselect all the other data sources. Save the Impulse.

Question 3. Which settings did you choose in the *Time series data* input block? Explain what these settings do and why you think that your choices are a good choice.

Task 8. Explore the different preprocessing blocks and generate features.

Question 4. What do the different preprocessing blocks do? Explain them. Why should we use preprocessing blocks instead of feeding the data directly into the neural network like in the previous labs?
Question 5. Which preprocessing block creates the best features separating the classes? Add screenshots. What kind of features did the blocks extract?

Task 9. Continue with at least the *Spectral Analysis* and the *Raw Data* blocks. If one of the other blocks looks more promising to you, continue using this one as well. For each of your processing blocks (you decided to continue with), train a classifier. You can start with the visual (simple) mode to create your neural network architecture. But make sure to switch to Keras (expert) mode to understand the model in detail. If you like, do some modifications here using your knowledge from the previous labs. Select your board in the upper-right corner before training to get the correct profiling information.

Question 6. What are 1D Convolutions? How do they differ from 2D convolutions used for images?
Question 7. Briefly explain your models. Focus in your explanation on the parts of the code deviating from the network architectures you used in the previous labs. E.g., what is `activity_regularizer=tf.keras.regularizers.l1(0.00001)`? Include the code for your models in your submission.
Question 8. What is the training performance of the models? How much memory is your model expected to need? What execution time is estimated? You can add screenshots. Check also for the memory usage and execution time of your DSP blocks.

Task 10. Head to your dashboard and download the quantized models of your classifiers. Head to `https://netron.app/` and open your models with it. Click on the input or output layer and take a look at the quantization equations.

Question 9. Are the quantization equations the same for each of your classifiers? Why (not)?

Part 4 – Model Testing

Task 11. Evaluate your models in the *Model Testing* section.

Question 10. How good do your models classify the test data? Do the models generalize well? Create plots comparing your models. Also explain *F1 Score* and *Uncertainty* displayed in the confusion matrix.

Task 12. If your models didn't perform well, head back to the data acquisition and record additional data and/or modify your models with the knowledge you have to avoid overfitting.

Task 13. After finding well-functioning models, you will use next an automated method for finding a good model. Use the EON Tuner[2] to automatically search for a good model. This might take a while.

[2]https://docs.edgeimpulse.com/docs/edge-impulse-studio/eon-tuner

Question 11. Did the EON Tuner come up with a better model than you? If so, in which regard is it better? Is it still better when you limit it to using only accelerometer data? *(To answer the latter question, first answer Question 12.)*
Question 12. If the EON tuner resulted in a better model, use this model as well in the *Model Testing* section and add its results to the plots you created in Question 10. *Please note: Before doing so, save your current version with the versioning tool on the left.*

Task 14. From this task onward, we will only use your best performing model. Save your previous state with the *Versioning* tool in Edge Impulse, and afterward remove all blocks no longer needed from your impulse design. Train your impulse one more time.

Part 5 – Live Classification

Task 15. Head to *Live classification* and test your model with the device. Try each of your motions.

Question 13. Explain the output of the classification results. Does your model work? Did it misclassify some timestamps? Is this a bad thing? Why (not)?

Task 16. Try the live classification with another Arduino board.

Question 14. Does the classification also work for another board? How does the performance compare when you use a board other than the one you used for collecting training data?

Task 17. Head to *Devices* and add your phone as an additional device. You can just scan the QR Code for this. Now perform live classification with your phone.

Question 15. Does the classification also work for your phone? If it doesn't, why not? Does the performance change when you change the orientation of your phone? In which orientation do you have to hold your phone for it to work best?

Part 6 – Deployment

Task 18. Build two Arduino Libraries for your model – one with enabling the EON Compiler and one without. For both libraries, use the quantized version.

Question 16. What is the EON Compiler, and why is the memory usage so different between the two libraries? Compare the models included in the two libraries (in `src > tflite-model`). How do they differ? What makes one of them smaller?

Task 19. Include the libraries into your Arduino IDE (`Add .ZIP Library...`). Open the accelerometer example that comes with your library and flash it to your board. Open a serial monitor.

Question 17. Explain the Arduino program and the output of the serial monitor. Also, why is there a reference to `numpy` in the Arduino program? How is that possible in C++? Evaluate how well and how fast the classification works for each of your motions. Is there a difference in performance between the two Arduino libraries?

Task 20. Open the continuous accelerometer example that comes with your library and flash it to your board. Open a serial monitor.

Question 18. Explain the Arduino program and the output of the serial monitor. How does it differ from the previous program? Does it use multithreading? If so, how? How well does the continuous mode work? Under which circumstances would you use either of the two modes? What applications would require the one or the other mode?
Question 19. Discuss the pros and cons of Edge Impulse. When should one use it and when rather not? Also elaborate on the privacy and ethical aspects of using Edge Impulse versus building a model locally.

Optional: To Go Further

Task 21 ***(optional).*** Include the other data sources (Gyroscope and Magnetometer) and train a model using the full inertial measurement unit.

Question 20 ***(optional).*** How does this model compare (regarding memory usage and performance) to your model using only accelerometer data? What are the pros and cons of including all data sources? For which applications would you use all and for which only some of them?