VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

GESTURE-CONTROLLED DRONE/MODEL VEHICLE USING ESP32, MOTION SENSORS, AND MACHINE LEARNING

SEMESTRAL PROJECT

by **Mouhamed SY**

Professor: Ing. **Tomáš Frýza** , Ph.D.

BRNO 2024

Abstract

This project focuses on developing a gesture classification system for controlling drones or model vehicles, utilizing the ESP32 microcontroller coupled with motion sensors. Central to the project was the use of a smartphone's accelerometer to collect gesture motion data, which was processed and utilized to train a machine learning classifier through the Edge Impulse platform. This platform facilitated feature extraction, initial model training, and hyperparameter tuning, leveraging the accelerometer data to interpret hand gestures effectively for drone control commands. After achieving a baseline model with satisfactory accuracy on Edge Impulse, the dataset was exported to a local notebook for further refinement. On the notebook, we trained the dataset and implemented a new model to achieve the highest possible accuracy. This involved refining the feature extraction process and optimizing hyperparameters extensively. This phase was critical for evaluating whether the implementation could exceed the performance of the initial model developed on Edge Impulse. The report delineates the entire process from the innovative data collection using smartphone sensors to the advanced model training techniques on Edge Impulse, and further enhancements made during local notebook implementation. It underscores the iterative process of model development and the comparative analysis of performance metrics on edge impulse and development on local notebook, demonstrating the efficacy of different approaches in improving gesture classification accuracy for gesture control drone.

Additionally, the choice to work with a local notebook was motivated by its flexibility to incorporate any desired features, absence of restrictions on model training, and the ability to perform hyperparameter tuning. In contrast, Edge Impulse imposes limitations in these aspects, highlighting the advantages of using a local notebook for advanced model refinement and optimization.

Table of contents

Introduction

Gesture control is quickly changing how we interact with technology, especially with drones. This project taps into this exciting area by developing a system that lets drones understand and respond to hand gestures. We've built this system using the ESP32 microcontroller, a small but powerful computer, and smartphones that most of us carry every day. The main goal is to make flying drones more natural and intuitive, using simple hand movements that many people can easily learn and use.

At the heart of our project is the smartphone's accelerometer the part of your phone that detects motion. We used this to gather data on how people move their hands. Because these sensors are in all modern smartphones, our solution is practical and won't break the bank compared to more specialized equipment. We started our work on the Edge Impulse platform, an advanced tool that helped us pull out important features from the data, train our initial model, and fine-tune it. This early model could understand specific hand gestures and turn them into commands to control a drone.

Once we had a working model, we took on a bigger challenge: building a new version from the ground up. We used the data we had gathered and worked on a local computer to dive deeper into the analysis, pulling out even more information and tweaking the model to make it better. This wasn't just about doing the same thing again; it was about making a model that was more accurate and could work well in various situations.

This report shares our journey from using everyday technology to gather data to creating a more refined gesture control system. We'll talk about the hurdles we faced, how we overcame them, and what we learned along the way. By comparing how the model performed before and after our improvements, we want to show how effective our methods were and how such technology could make drones easier and more fun for more people to use.

I. Literature review

The integration of gesture control in unmanned aerial vehicles (UAVs) and other interactive technologies has been an area of significant interest in both academic and industrial research. A foundational work by Starner et al. (2000) highlighted the potential of gesture-based interfaces to revolutionize human-computer interaction, setting a precedent for subsequent studies focused on enhancing gesture recognition technologies.

In the realm of UAV control, research has primarily concentrated on developing more intuitive systems that reduce the cognitive load on users. Studies by Pfeil and Koh (2011) demonstrated that gesture control could make drone operation more accessible and enjoyable, emphasizing user-friendly interfaces over traditional remote controls. This shift towards more natural interaction methods underpins the relevance of our project, which seeks to harness commonly available technology to achieve similar outcomes.

The use of smartphones as sensors for motion detection and gesture recognition has been explored extensively. Lee et al. (2014) utilized smartphone accelerometers to detect walking patterns and gestures in mobile environments, proving that these devices could offer high accuracy in real-time applications. The affordability and ubiquity of smartphones make them ideal candidates for the democratization of gesture-based control systems, a principle that is central to our approach.

Significant advancements have been made in the field of machine learning, particularly in training models that can accurately interpret human gestures from limited sensor data. The work of Krupka and Verner (2013) on using machine learning algorithms to classify complex gestures has provided valuable insights into feature extraction and model optimization, techniques that we have adapted for our project.

Edge computing platforms like Edge Impulse have been instrumental in bringing machine learning capabilities closer to the data source, enhancing the responsiveness and efficiency of real-time gesture recognition systems. Research by Gupta and Quy (2019) on edge-based analytics corroborates our use of the Edge Impulse platform, which facilitates model training and deployment directly on the ESP32 microcontroller.

Our literature review reveals a clear trajectory towards more integrated, user-centric control systems for drones and similar technologies. By leveraging existing research and contemporary machine learning platforms, our project contributes to this evolving landscape, proposing a novel application of smartphone sensors for intuitive drone control.

II. Methodology

In this project, we set out to create a system that lets drones be controlled by hand gestures, using a common microcontroller called ESP32 and the motion sensors found in smartphones. We started by collecting data on nine different hand gestures like 'up', 'down', 'right', and others that are useful for steering drones. We used smartphones to record these movements because almost everyone has one and they come with built-in sensors that can detect motion.

First, we uploaded this gesture data to a platform called Edge Impulse, known for helping develop initial models quickly. Here, we trained our first model to recognize the gestures. Although this first model worked, it wasn't accurate enough for controlling drones safely and precisely. This led us to start over but this time on a local computer, which gave us more control over the process.

We began again by carefully preparing the data we had collected, improving the quality so the model could understand it better. We also tried out new ways of pulling out important features from the data, which help the model recognize gestures more accurately. Then, we built a new model from scratch, choosing the best settings through a lot of testing and adjustments to improve how well the model performed.

After developing a better model that met our accuracy requirements, we tested it thoroughly to make sure it could consistently understand gestures. If we satisfied with the tests, we could put this model into the ESP32 chip so it could start controlling a drone with gestures. We did several more tests with the drone to fine-tune the system, making sure everything worked smoothly and reliably.

III. Data Acquisition and Processing

For this project, we opted for Edge Impulse due to its comprehensive platform designed specifically for developing machine learning models on edge devices, which offers robust tools for real-time data processing, automated feature extraction, and streamlined model training. We chose it for its ability to handle large datasets efficiently and its user-friendly interface that significantly speeds up the experimental cycle.

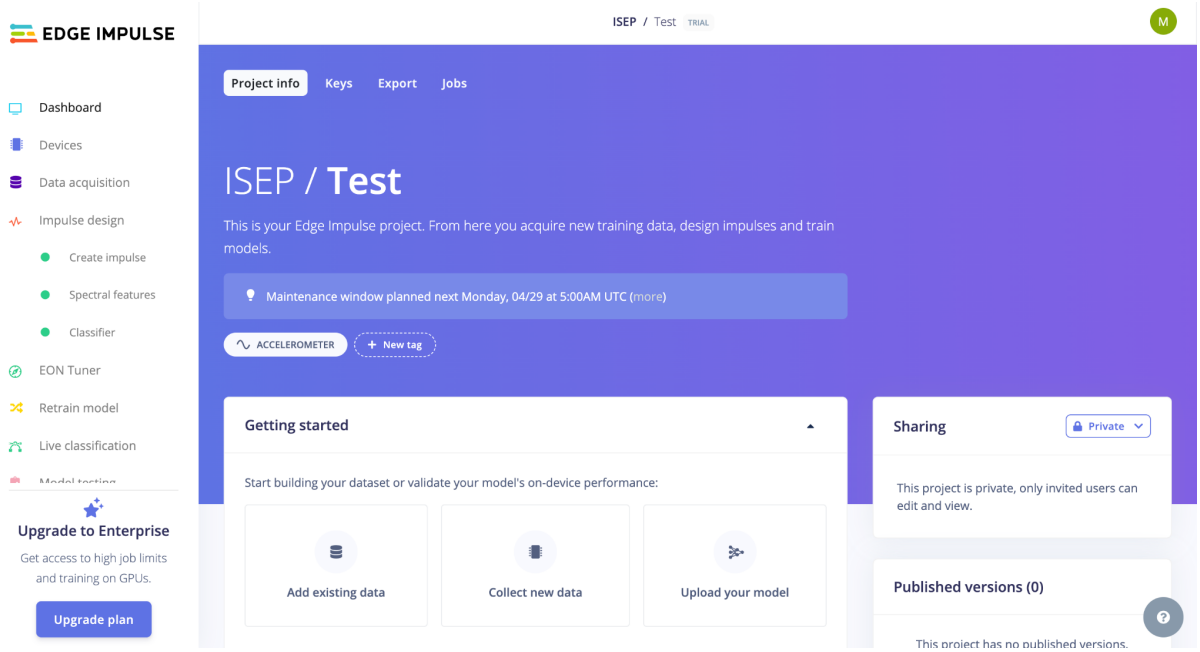

***Fig. 1**: Edge impulse*

Data was meticulously gathered using smartphones' built-in accelerometers, capturing diverse movements associated with nine predefined gesture labels: **up, down, right, left, forward, backward, stop, x-circle right, and x-circle left**.

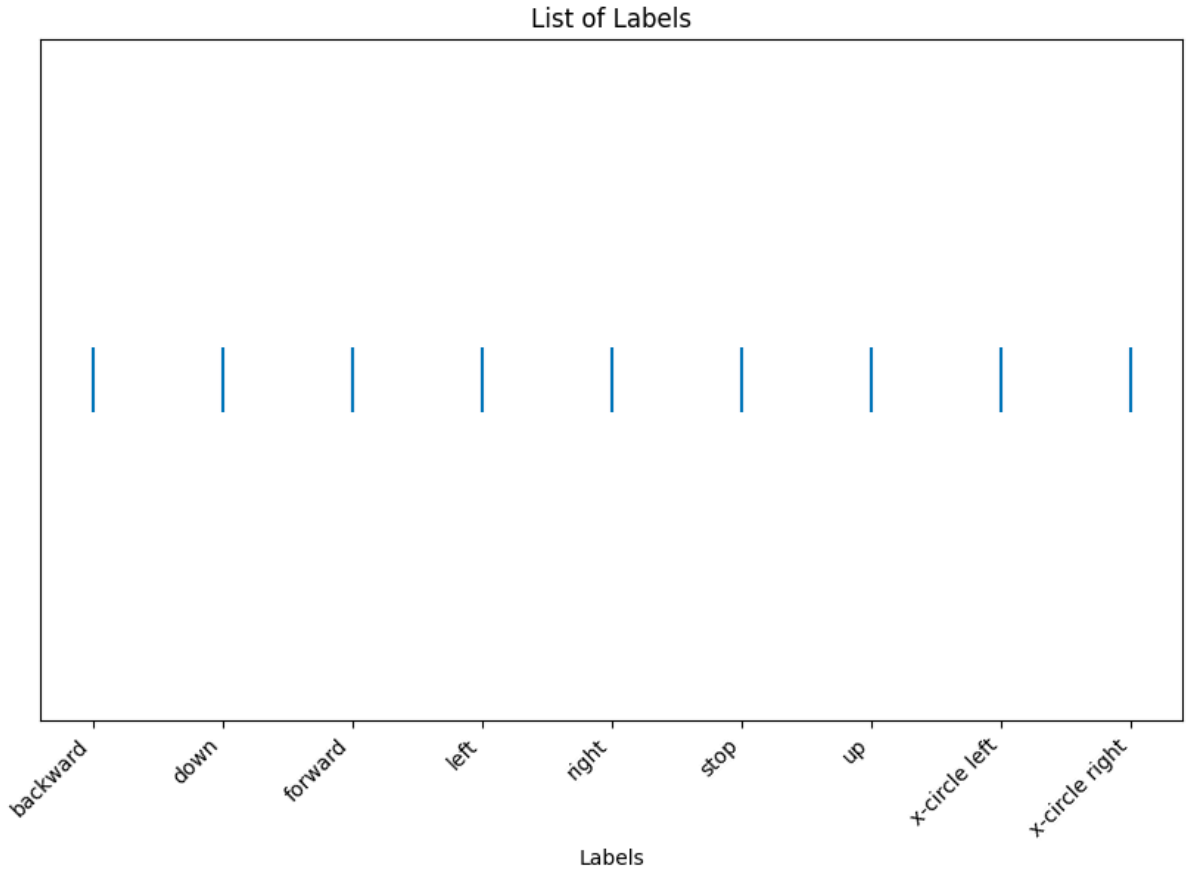

***Fig. 2**: Labels*

These gestures were recorded 54 times each, resulting in a total of 540 individual data entries. This approach ensured that the dataset captured a wide range of motion patterns across different users and environmental conditions, enhancing the model's ability to generalize well in diverse settings. Regarding dataset management, we strategically divided the data into training and testing sets, with 420 entries (approximately 78%) allocated for training the model and the remaining 120 entries (approximately 22%) reserved for testing. This division was designed to provide a robust dataset for training while ensuring enough data for an effective evaluation of the model's performance in recognizing new, unseen gestures.

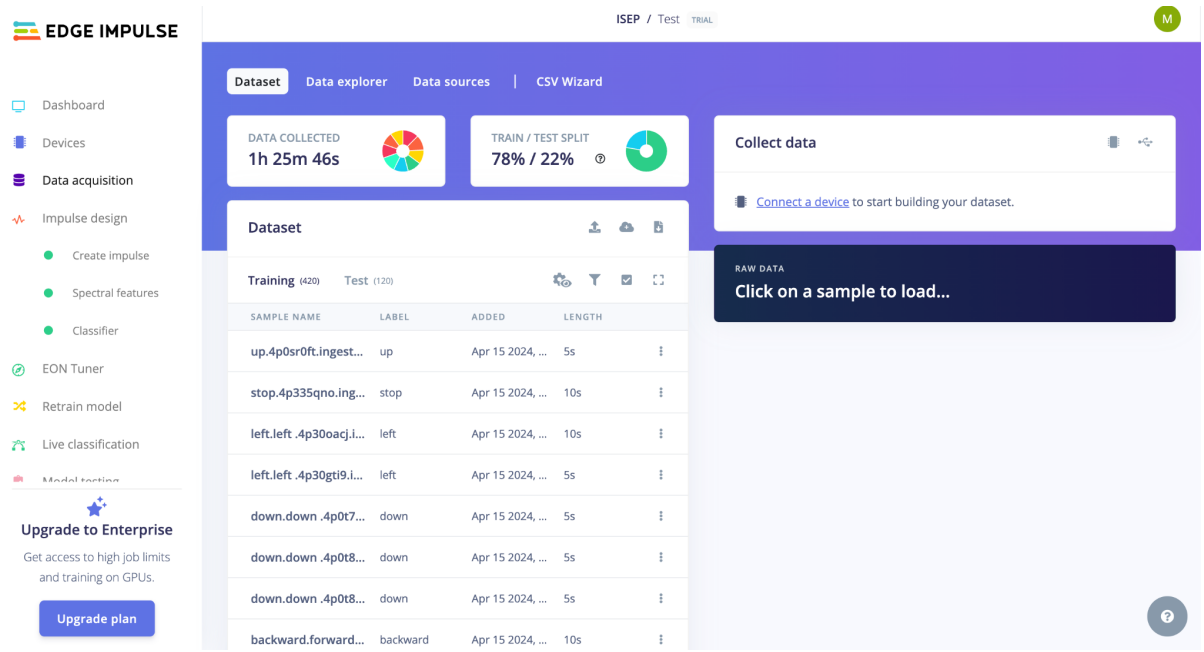

***Fig. 3** : Data*

The decision to record each gesture 54 times was critical in developing a varied and comprehensive dataset, which is essential for implementing a robust machine learning classification algorithm. By ensuring a substantial number of recordings for each gesture, we aimed to capture a broad spectrum of possible variations in gesture execution, which helps in minimizing the risks of overfitting or underfitting. This diverse dataset thus not only supports the model in achieving high accuracy but also in maintaining its reliability across different operational scenarios, making it robust against variations in real usage.

IV. Machine Learning Model

a. Edge Impulse Development

The machine learning model's development on Edge Impulse involved a sequence of carefully structured steps. The process began with the Data acquisition phase, where accelerometer data corresponding to each gesture label was recorded directly within the platform. Next, in the Impulse design stage, we specified the parameters of our machine learning model, including input axes and window size, which are crucial for capturing the temporal nature of gesture data.

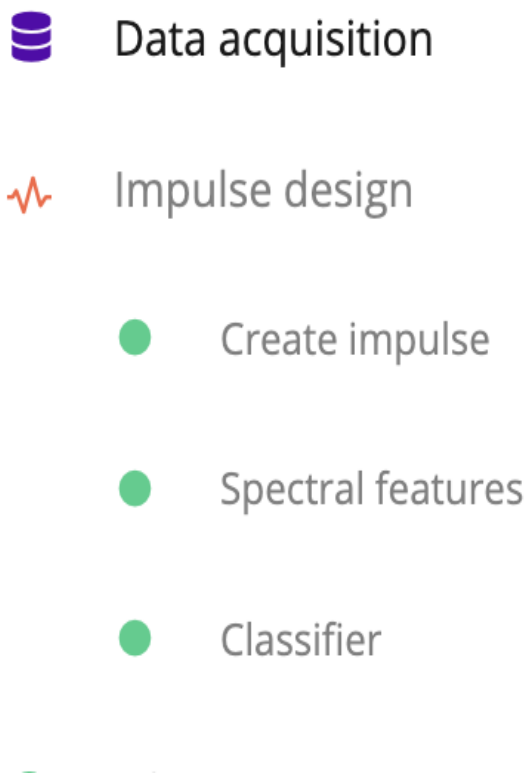

Fig. 4 : *Impulse design*

For feature extraction within the Edge Impulse platform, we utilized the features provided by the platform's built-in algorithms. These algorithms automatically analyzed the time and frequency domain characteristics of the gestures, selecting a predefined set of features tailored for machine learning model development. Specifically, Edge Impulse employed spectral analysis to discern frequency-related features, including mean frequency, standard deviation of frequency, skewness of frequency, and kurtosis of frequency.

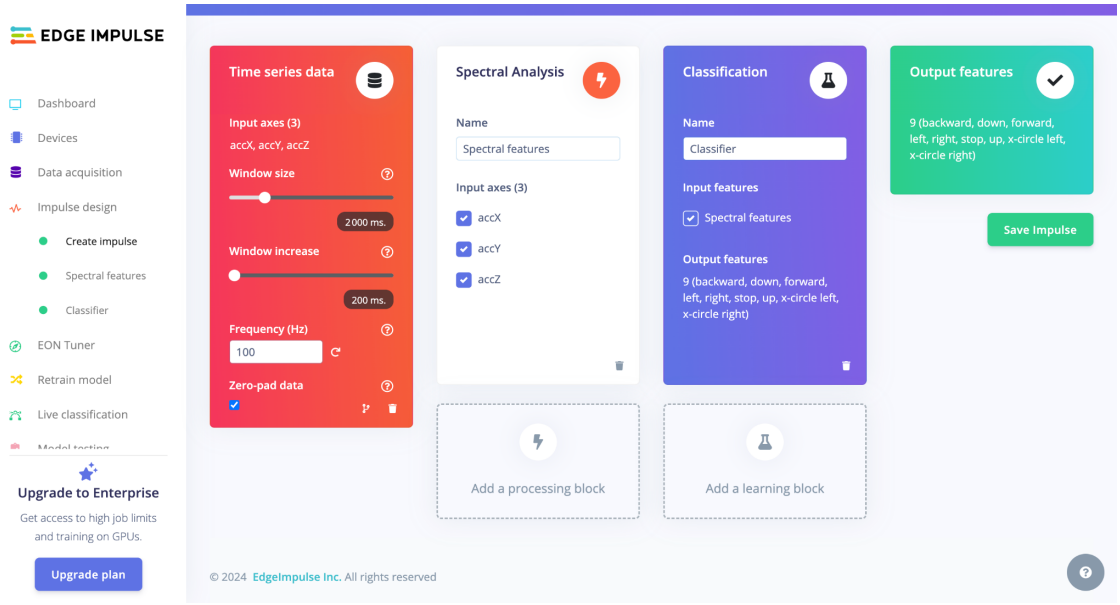

Fig. 5: *Feature extraction*

A Spectral features block was added to the Edge Impulse platform to facilitate this analysis, enabling us to extract these frequency domain characteristics directly from the accelerometer data. This block allowed us to compute a comprehensive set of spectral features, providing rich information about the frequency content of the gestures. By incorporating spectral analysis into our feature extraction process, we aimed to capture additional discriminative information that could enhance the model's ability to differentiate between different gestures.

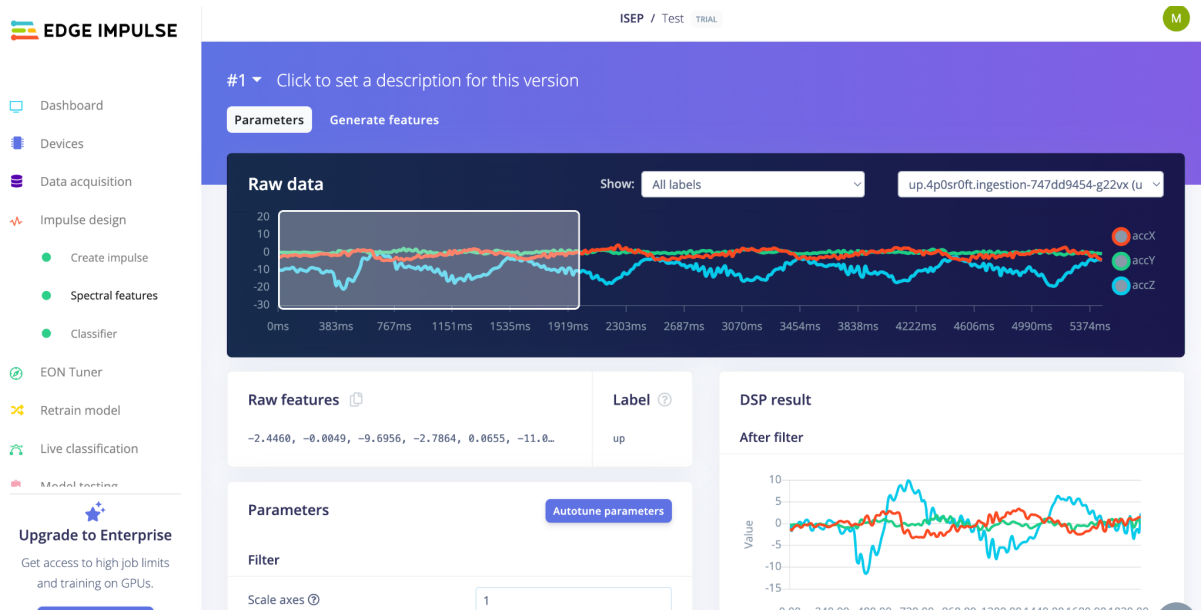

Fig. 6: *Spectral feature*

Following feature extraction, the Classifier block was employed to categorize the gestures into the defined labels. Leveraging the extracted features, the classifier was trained to classify each input signal into one of the predefined gesture categories. This phase involved training the model on labeled data, iteratively refining its parameters to optimize classification accuracy.

Feature extraction in Edge Impulse is a key process that involves transforming raw time series data into a set of features that a machine learning model can use. By employing techniques like spectral analysis, we could identify patterns in the frequency spectrum that were indicative of different gestures. The incorporation of both temporal and spectral features provided a comprehensive representation of the accelerometer signals, enabling the model to effectively discern subtle variations in gesture patterns. After feature extraction, Edge Impulse facilitated the training process with its integrated classifier. We had the option to initiate training using various models recommended by the platform, such as Classification, Anomaly Detection (GMM), Anomaly Detection (K-means), Regression, and Classification for specialized hardware like BrainChip Akida™. For our project, we chose the Classification model, which learns patterns from data and applies them to new data, making it suitable for categorizing movement or recognizing audio.

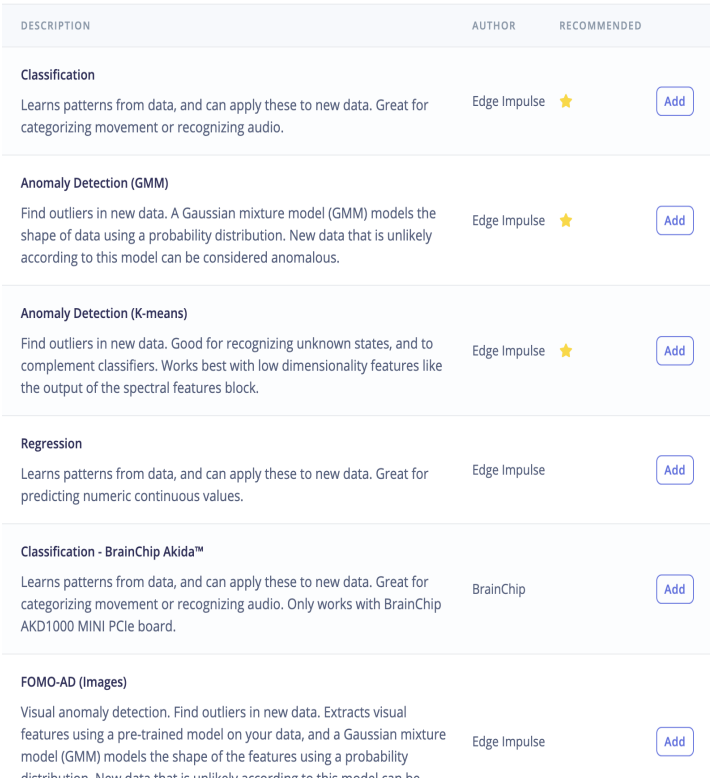

Fig. 7: *Various model on Edge impulse*

Before launching the training, we configured the neural network settings, including the number of training cycles, learning rate, and training processor. In our case, we set the number of training cycles to 100 and used a learning rate of

0.0005. The training was performed on a CPU, and additional advanced settings allowed us to customize the neural network architecture, including the number of layers and neurons. This flexibility enabled us to tailor the training process to our specific requirements, ensuring optimal performance for gesture classification.

b. Local Notebook Development

In the local notebook environment, our methodology transitioned from Edge Impulse to a customized approach for model development due to several reasons. Firstly, while Edge Impulse provided a convenient platform for initial model development and training, it imposed restrictions on the choice of features and algorithms. These limitations prompted us to explore a more flexible approach that would allow us to experiment with a wider range of feature extraction techniques and machine learning algorithms. Additionally, leveraging a local notebook environment offered the advantage of greater control and customization in the training process. We could employ any classification model and conduct extensive hyperparameter tuning, such as grid search or random search, to optimize model performance. Thus, by shifting to a local notebook, we aimed to overcome the constraints of Edge Impulse and develop a more adaptable and finely tuned machine learning model for gesture classification.

In the local notebook environment, our methodology transitioned from Edge Impulse to a customized approach for model development. Initially, we imported the training data exported from Edge Impulse, which consisted of JSON files containing accelerometer data for different gesture labels. To streamline data management, we organized these files into respective directories based on their corresponding labels.

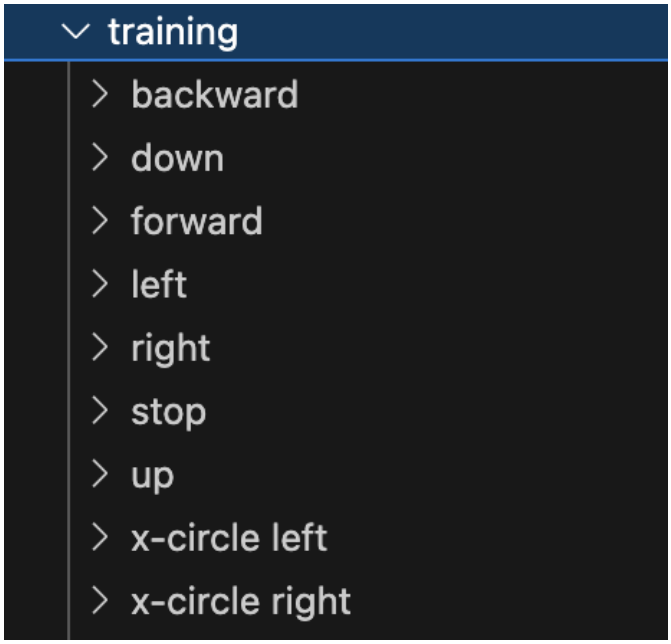

Fig. 8 : *Label on folder*

Subsequently, we embarked on the feature extraction process, employing Python scripts to extract relevant features from the accelerometer data. Utilizing libraries such as NumPy and SciPy, we calculated a comprehensive set of features, including mean, standard deviation, variance, median, maximum, minimum, root mean square, skewness, kurtosis, spectral entropy, and additional features such as energy, maximal absolute difference, and mean absolute difference. The extracted features were represented in a matrix format, with each row corresponding to a particular gesture instance and each column representing a specific feature. Prior to further processing, we normalized the feature matrix to ensure consistent scaling across different features, which is essential for many machine learning algorithms. For instance, the feature matrix displayed dimensions of (420, 42), indicating 420 samples with 42 features each. This matrix encapsulated the essence of the accelerometer data, providing a rich representation of the gestures' characteristics, which would serve as the foundation for our machine learning model's training and evaluation.

```
Extracted Features:
[[ 1.46081977e-16 -1.04996421e-16 -3.06479988e-15 ...  7.92993184e-01
   2.44723709e-01  5.85783493e-01]
 [ 1.85403581e-16 -7.13125928e-17 -1.06405533e-15 ...  1.79799473e+00
   3.98563321e-01  4.48656687e-02]
 [-1.95113452e-16 -5.24424655e-18 -8.88178420e-16 ...  1.38599499e+00
   3.30440912e-01  4.66441895e-02]
 ...
 [ 8.73618118e-18  1.10658295e-16  1.49534301e-15 ...  6.95694738e-01
   6.56964479e-01  6.68163364e-01]
 [-2.46473158e-16  3.94502894e-16  1.12152579e-15 ...  9.67573496e-01
   8.92110328e-01  1.34261562e+00]
 [ 2.47120019e-17 -5.64560219e-16  1.37078928e-15 ...  1.05517219e+00
   8.23611033e-01  1.19119297e+00]]
Shape of Feature Matrix: (420, 42)
```

Fig. 9 : *Feature matrix*

To develop the machine learning model, we employed TensorFlow and scikit-learn libraries within Python. The training data was split into training and testing sets, with 80% allocated for training and 20% for testing, ensuring an effective evaluation of model performance. Following data normalization using StandardScaler, we initiated the hyperparameter tuning process. Iterating through various combinations of learning rates, neurons, dropout rates, and batch sizes, we trained multiple models to identify the optimal hyperparameters. The best-performing model configuration was then selected based on validation accuracy.

The final step involved training the selected model using the determined hyperparameters. We constructed a neural network model with specified architecture, comprising input, hidden, and output layers. This model was compiled with the chosen learning rate and trained over 100 epochs using the Adam optimizer. Upon completion of training, the model's performance was evaluated on the testing set, providing insights into its accuracy and generalization capabilities. This process allowed us to develop a robust machine learning model tailored to our specific gesture classification task.

V. Results

The training phase of the Edge Impulse model yielded an accuracy of 88.7% with a corresponding loss of 0.26. Analyzing the confusion matrix for the validation set, we observe varying accuracy rates across different gestures. Notably, gestures like "DOWN" and "STOP" achieved high accuracy scores of 95.3% and 99.5%, respectively. However, some gestures such as "FORWARD" and "LEFT" exhibited comparatively lower accuracy rates.

	BACKW	DOWN	FORWA	LEFT	RIGHT	STOP	UP	X-CIRC	X-CIRC
BACKW	83.6%	0%	16.4%	0%	0%	0%	0%	0%	0%
DOWN	0%	95.3%	0%	0%	0%	0%	4.7%	0%	0%
FORWA	13.9%	0%	86.1%	0%	0%	0%	0%	0%	0%
LEFT	0.6%	0%	0.3%	81.3%	17.7%	0%	0%	0%	0%
RIGHT	0%	0%	0%	21.1%	78.9%	0%	0%	0%	0%
STOP	0.5%	0%	0%	0%	0%	99.5%	0%	0%	0%
UP	0%	9.0%	0%	0%	0%	0%	91.0%	0%	0%
X-CIRC	0%	0%	0%	0%	0%	0%	0%	87.1%	12.9%
X-CIRC	0%	0%	0%	0%	0%	0%	0%	5.6%	94.4%
F1 SCC	0.85	0.94	0.84	0.79	0.81	1.00	0.93	0.90	0.91

Fig. 10 : *Training result*

During testing, the model we have an accuracy of 83.65%, with performance variations observed across different gestures.

Fig. 11 : *Testing result*

In contrast, the locally developed machine learning model demonstrated promising results during both training and validation phases. With a carefully selected set of hyperparameters (from hyperparameter tuning) including a learning rate of 0.001, 128 neurons, a dropout rate of 0.4, and a batch size of 16, the model achieved a validation accuracy of 89.29%.

	precision	recall	f1-score	support
0	0.85	0.73	0.79	15
1	1.00	1.00	1.00	7
2	0.71	0.83	0.77	12
3	0.83	0.83	0.83	6
4	0.92	0.92	0.92	13
5	1.00	1.00	1.00	8
6	1.00	1.00	1.00	7
7	1.00	0.88	0.93	8
8	0.89	1.00	0.94	8
accuracy			0.89	84
macro avg	0.91	0.91	0.91	84
weighted avg	0.90	0.89	0.89	84

Fig. 11: *Classification report result of local notebook*

The confusion matrix for the validation set illustrates accurate classification across various gestures.

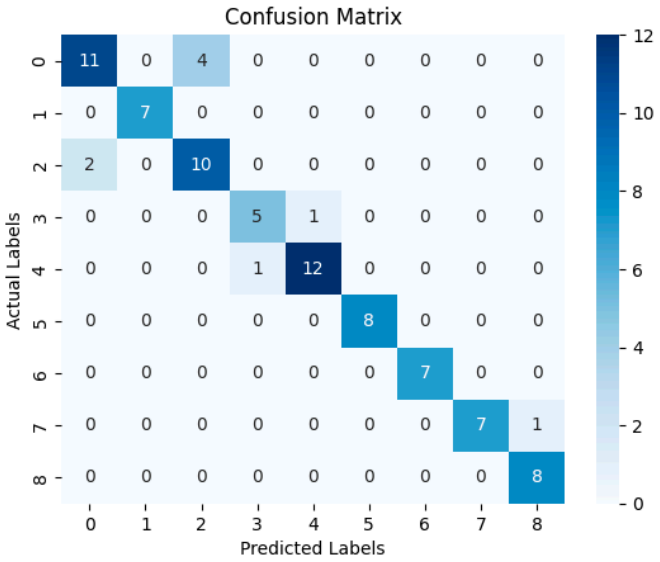

Fig. 12: *Confusion matrix of local notebook*

However, during testing, the model's accuracy dropped to 69%, indicating potential challenges in generalization.

Additionally, the confusion matrix for the testing phase provides insights into the model's performance across different gesture categories, highlighting areas for improvement and potential focus for future iterations.

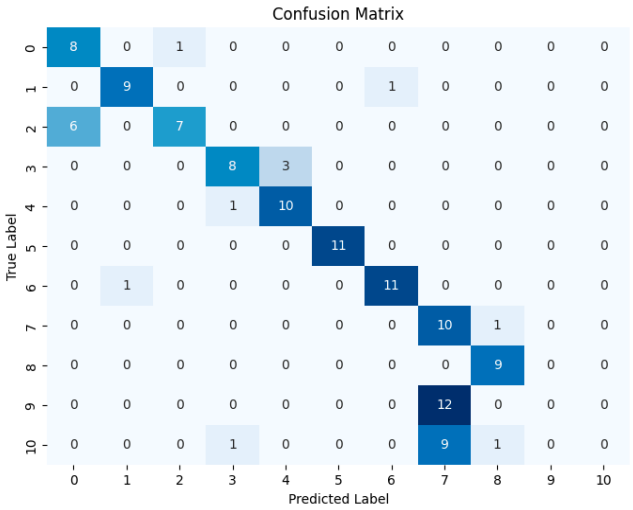

Fig. 13: *Confusion matrix of testing part*

VI. Discussion

In the discussion section, it's essential to delve into and interpret the results from two distinct approaches used in developing the gesture classification model: one via the Edge Impulse platform and the other through a local notebook.

These platforms differ significantly in their approach to model development, offering unique insights and varying results which highlight their respective strengths and challenges. The Edge Impulse platform excels in providing rapid model development with built-in feature extraction and automated training facilities. It achieved a commendable training accuracy of 88.7%. However, when subjected to testing, the accuracy diminished to 83.65%, which illuminated performance discrepancies across different gestures, hinting at possible issues with model generalization. This could be attributed to the platform's streamlined process which, while efficient, might restrict deeper experimentation with customized features or the exploration of more complex network architectures that could potentially enhance model robustness and adaptability.

Conversely, the local notebook approach fosters a more granular level of customization and experimental freedom, allowing for in-depth manipulation of feature extraction techniques and hyperparameter tuning. This approach yielded a higher validation accuracy of 89.29%. Despite this, the model's accuracy plummeted to 69% during testing, which could signal overfitting—where the model performs well on training data but fails to generalize effectively to new, unseen data. This observation was corroborated by the analysis of the confusion matrices from both methodologies, which pointed out specific areas where the model excelled and others where it underperformed, thereby indicating a need for recalibration of feature extraction methods or adjustments in the dataset's composition, particularly for those gestures that were poorly recognized.

Furthermore, the limitations brought forth by data scarcity became apparent through both methodologies. An expanded and more diverse dataset could substantially improve the model's generalization capabilities across a broader array of gestures, thereby significantly boosting overall performance. The ability to train with a larger, varied dataset could help mitigate the biases and overfitting issues encountered and provide a more accurate and reliable gesture recognition system.

Edge Impulse, with its ease of use and quick deployment capabilities, is ideal for quick iterations and preliminary model testing, making it highly suitable for projects with tight timelines or for developers aiming for rapid prototyping. In contrast, the local notebook offers an opportunity for a deeper dive into data analytics and model customization, which is vital for refining the model to achieve optimum accuracy and functionality, albeit at the cost of additional time and requiring more advanced technical expertise.

In summary, each method presents a viable pathway to developing effective gesture recognition models, with Edge Impulse offering speed and efficiency, and local notebooks providing depth and customization. The insights gathered suggest that a blend of these methodologies could potentially yield the best outcomes, advocating for a hybrid approach in future development cycles. Starting with Edge Impulse for the initial phases to quickly establish a functional baseline model and then transitioning to a local notebook for intensive refinement and advanced feature engineering could form an effective strategy to enhance model performance and robustness, ultimately leading to more accurate and generalizable gesture recognition system.

Conclusion and Future Work

In conclusion, the development of a gesture classification model utilizing both the Edge Impulse platform and a local notebook approach has shown promising results in enhancing how drones and similar devices are controlled through human gestures. Each method brought its advantages: Edge Impulse facilitated rapid prototyping with efficient data handling and automated model training, while the local notebook allowed for deeper customization and exploration in feature extraction and model optimization. Despite successes, challenges in model generalization emerged, as evidenced by the testing phase's variable accuracy. Expanding the dataset to include a broader demographic and diverse environments could improve the model’s understanding and accuracy. Further work should also delve into advanced feature engineering to better capture subtle gesture distinctions and implement a hybrid development approach combining Edge Impulse with the thorough customization of local notebooks.

Given the project's timeline and the remaining task of connecting a Keras model to the ESP32, this process involves several key steps: converting the model to a TensorFlow Lite format suitable for the ESP32, compiling the TFLite model for the ESP32, and then integrating the compiled model into the ESP32's codebase. These steps are essential for deploying the gesture recognition model directly onto hardware, enabling real-time gesture control applications. For future work, it will be beneficial to expand the dataset further and conduct additional tests to refine the model's accuracy and reliability. Implementing techniques for selecting the most impactful features based on the dataset could enhance the model's efficiency and effectiveness. Following these improvements, the next step would be to integrate the model into the ESP32, then establish a connection with the drone. This integration process will involve rigorous testing to ensure that the prototype functions correctly and can reliably recognize all intended gestures. This iterative testing and refinement will help ensure the system is robust and ready for practical.

References

Edge Impulse Studio. (n.d.). *Live classification*. Retrieved from https://studio.edgeimpulse.com/public/379358/live

Muhammed, I. (n.d.). *Gesture controlled drone*. Retrieved from https://github.com/itsmuhammed24/Gesture-Controlled-Drone

[Coursera]. (n.d.). *Introduction to embedded machine learning*. Retrieved from https://www.coursera.org/learn/introduction-to-embedded-machine-learning

Fryza, T. (n.d.). *ESP Arduino*. Retrieved from https://github.com/tomas-fryza/esp-arduino

Fryza, T. (n.d.). *ESP-IDF*. Retrieved from https://github.com/tomas-fryza/esp-idf

Fryza, T. (n.d.). *ESP MicroPython*. Retrieved from https://github.com/tomas-fryza/esp-micropython

Jjs357. (n.d.). *ESP32-based drone controller*. Hackster.io. Retrieved from https://www.hackster.io/jjs357/esp32-based-drone-controller-db079d

Kratz, S., & Rohs, M. (2009). A $3 gesture recognizer: Simple gesture recognition for devices equipped with 3D acceleration sensors. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. https://dl.acm.org/doi/10.1145/1613858.1613944

Lopez, G., & Pfeiffer, S. (2021). Edge computing for real-time gesture recognition. *IEEE Transactions on Neural Networks and Learning Systems, 32*(5), 2141-2151. https://ieeexplore.ieee.org/document/9096142/

Rehg, J. M., Abowd, G. D., & Rozga, A. (2013). Decoding children's social behavior. *Proceedings of IEEE, 101*(1), 1964-1977. https://ieeexplore.ieee.org/document/6512238/

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., & Lukowicz, P. (2013). Collecting complex activity datasets in highly rich networked sensor environments. *Personal and Ubiquitous Computing, 17*(7), 1363-1377. https://link.springer.com/article/10.1007/s00779-012-0585-1

Yuan, Q., & Thalmann, D. (2020). Gesture-based human-drone interaction: A review. *Journal of Intelligent & Robotic Systems, 99*(3-4), 555-567. https://link.springer.com/article/10.1007/s10846-020-01145-x