EchoBand: A Voice-Activated, Gesture-Controlled Wristband

Aria Sharifnia
Department of Computer Science, University of Calgary
Calgary, Canada
aria.sharifnia@ucalgary.ca

Nathan Campbell
Department of Computer Science, University of Calgary
Calgary, Canada
nathan.campbell@ucalgary.ca

1 Introduction

Wearable devices have become extremely common in recent years, ranging from the simple FitBit to smart watches and meta glasses, changing the way we interact with the world around us. One limitation that we spotted is the lack of hands-free control that these devices allow you to exert. Many devices that currently exist offer either voice commands or gesture recognition tied directly to one specific tool, like the new Meta glasses. None of these solutions offers something openly flexible with device control for many different tools at the same time, without relying on specific device software to handle the computation. The biggest technical challenge with this is keeping the edge computing, which is the true thing that allows the flexibility to function so well. Running a machine learning model on a tiny chip that will also handle gesture recognition and Bluetooth control is more challenging to get an accurate reading from due to the small size with fewer parameters. To address this, we used a board with all of the required sensors built in for a smaller device footprint and developed a self-contained, wrist-worn wearable that integrates a 9-axis IMU, a microphone, and BLE capabilities. Our device detects keywords describing the tool you want to affect, and then watches for gestures to control that tool more precisely. One of the best ways to show it working is with our implementation of phone music control, whereby connecting to an Android phone, the watch will listen for the keyword "Spotify" before allowing the user to swipe their hand left and right to skip and go back through their song list. Ultimately this report demonstrates how using offline TinyML voice recognition with intuitive motion gestures can create a seamless interface for controlling smart devices without the latency or security risks of the cloud.

2 Related Work

Our project combines voice and gesture content into a single usable device. To understand where our project fits within previous work, we reviewed research across multiple areas of this project.

2.1 Multimodal Interaction

Using more than one way to communicate with a device, such as what we are doing with combining speech and gestures, makes technology easier to use, and also increases possibilities [14]. Research seems to divide these systems into 2 types, one being simultaneous and the other being sequential. Simultaneously would be trying to understand speech and gestures at the same time. While this could be powerful, it uses much more computing power and doesn't let you combine 2 senses, which isn't great for our goal [10]. Our project uses the sequential method instead meaning one action after another. For example, you say a word first that the device recognizes, and then the device listens for a gesture to act on that keyword. Studies by Buddi et al. [3] and Smith et al. [13] show that this step after a step method solves a major problem called the "Midas Touch," where a device accidentally gets triggered by a random command. By using a voice keyword to kind of "wake up" the device first, then act on it, we practically remove these accidental triggers. However, implementing this sequential approach requires defining a specific integration window, which is the maximum timeframe in which a gesture must occur after a keyword. Research by Gupta and Anastasakos indicates that while adaptive windows are optimal, fixed durations effectively balance avoiding accidental actions and still being able to perform your action after you say the keyword.

2.2 Recognizing Gestures on the Wrist

Previously, gesture recognition relied on cameras, which often suffered from occlusion and raised privacy concerns. Today, most wearables use Inertial Measurement Units (IMUs) because they are inexpensive, compact, and avoid privacy issues [18]. More advanced devices, such as the Mudra Band, use sensors that detect muscle electricity (EMG) [16]. While EMG sensors offer high precision, they require extra setup, must be strapped tightly to the skin, can malfunction if the user is, for example, sweaty, and are also more expensive [11]. Since our project focuses on simple movements like swiping and shaking, IMUs are the best fit. Recent research shows that simple sensors can recognize these hand motions with over 90% accuracy using basic machine learning models [8, 20]. This allows us to achieve our project goals without adding expense, discomfort, or complexity to the hardware. Notably, recent studies have also explored fusing IMU data with acoustic sensing, using wrist-worn microphones to detect mechanical vibrations of tendons for improved gesture recognition [7, 17]. Although this research is promising, our aim is to use the microphone strictly for voice commands to minimize processing overhead.

2.3 TinyMLs

Most systems with voice assistants, such as Siri or Alexa, send your voice to the internet to be processed. This can cause delay, requires an internet connection, and also raises privacy concerns [4]. Our device, on the other hand, uses TinyML to process everything directly on the Arduino in the watch. New software designs, such as Depthwise Separable Convolutional Neural Networks (DS-CNN), allow us to run keyword detection on small microcontrollers like the Cortex-M4 we used [15, 19]. We used EdgeImpulse to create a model for detecting the keywords that we added and then uploaded it onto the Arduino for our watch. Running the machine learning model on the device saves power because the radio doesn't have to be on all the time [1, 12]. Also keeps data private as it never has to send or save what you said, your voice, and gesture data never have to leave the wristband.

2.4 Universal Wireless Control

Finally, we need to put this all together. To be a universal remote, it needs to talk to many different systems. Bluetooth Low Energy (BLE) is the standard for this because it uses very little power [5]. Instead of trying to program the watch to know about every smart light or music source, we used a generic method. The Arduino in the watch sends a simple code to the smartphone over BLE, and the phone handles the logic of what to do with that code. This approach, where your smartphone acts as the bridge between the wearables input and then actions themselves, is a common and effective strategy in smart home design [9]. It allows our device to control almost anything it is programmed to control without needing complex software updates [2, 6].

3 Design Goals/Space

Among hands-free wearable devices we noticed they usually either have fine control with gestures for 1 specific tool, or broad control of many tools using voice controls. Our main goal was to develop a hands-free wearable device that uses voice commands to select a tool, and then gesture controls for easy control of that tool. A key consideration when designing this was trying to decrease the cognitive load and repetition normally found in voice assistants; the design incorporates a timed "active state" that allows for rapid, repeated gestures without redundant word triggers. The system was also designed to be computationally self-sufficient; the sensors and neural network inference need to occur on the wearable itself, rather than offloading data for processing in a specialized app. By processing commands locally the device can output standard BLE messages, enabling it to interface with a wide variety of external hardware without requiring the user to download software specific to our wearable device for processing. While this architecture was chosen for its flexibility and speed, it has significant secondary benefits in data privacy and reduced latency as raw audio and motion data are never transmitted wirelessly. Finally, the physical realization of the device is constrained by the need for a minimal footprint, necessitating the use of an efficient TinyML model that can operate accurately within the power and memory limitations of a compact, embedded system. When creating a case to contain this system with a sleek, minimalistic design we needed to make sure it would fit the battery and board snugly while allowing for sound to pass through to the microphone.

4 System Design And Implementation

Our system required a 9-axis IMU, a microphone, and BLE capabilities. We used an Arduino Nano 33 BLE Sense Rev2 so that we could stay within the direct Arduino ecosystem as well as contain all the necessary sensors onboard to avoid any issues. We powered the system with a 3.7V Lithium-Ion battery so that the watch would be rechargeable and maintain a small footprint while being able to stay powered for an extended period of time without BLE power fluctuation issues. For the microphone, we used a sampling rate of 16kHz, which was the maximum recording rate for the microphone; for the IMU, we sampled at ~66.7Hz to capture hand motions without excessive noise. For our voice recognition model, we used Edge Impulse to develop the full end-to-end training pipeline to build a lightweight neural network compatible with the microcontroller's memory constraints. For data collection, we used several different methods; we used a small amount of synthetic data of voice generated by OpenAI to get clean cuts of our keywords, and we also recorded our own voices using the onboard microphone within our 3D printed case, saying each word close to a hundred times, with light background noise to avoid the model overfitting. In order to gain good noise data, we made many recordings also using the onboard microphone, within a separate training version of our case on NBA game sounds, podcasts, and our own background conversation. When creating the model we decided to do some processing on the raw microphone data in order to get better results; we used a MFCC(Mel-Frequency Cepstral Coefficients layer to extract the relevant features from the raw data, and that is what the model would use as input. Our parameters can be seen in Figure 1. We used the Edge Impulse autotune function to find the best parameters without spending hours doing trial and error for better results. For the neural network we used a Classification model with the details of our convolutional neural network architecture and training settings visible in Figure 2 & 3.

For our gesture recognition we used a threshold-based system using the gyroscope and magnetometer, with the gyroscope detecting the motions themselves and the magnetometer was used to find the user's hand orientation, allowing for only specific gestures in specific orientations to trigger the detection. We chose to use 3 motions: a swipe to the left, a swipe to the right, and a shaking motion that can be used as a toggle. For the swiping motion, we would anaylze every "frame" of input to see if a swiping motion was detected and then require 2 of the same motions in a row in order to flag the motion as detected. This greatly improved our accuracy and prevented false positives. For the shaking motion, the magnetometer is used to ensure the hand is down by the side to avoid false positives, then we use a window approach where a shaking detection must be detected 10 times across 1 second for the detection to trigger, this is 15% of the frames within the second.

Putting everything together we have a visual explanation in Figure 4 showing a representation using a Finite State Machine diagram. When the wristband is on, it sits in an idle mode where it exclusively listens for keywords to move into the next detection stage. Once a keyword is detected, it will move into a gesture recognition mode, sending a BLE message stating that it will watch for positive gesture recognitions and stop listening for voice commands. The device will stay in this mode until an 8-second timeout period has been reached, where no gestures were recognized, then it will move back into the idle position. When a gesture has been recognized the wristband will send a BLE message indicating what mode and gesture the detection should be triggering, this is `sp_next, sp_prev, sp_toggle, vol_up, vol_down, vol_mute, br_up, br_down, br_toggle`. Using these commands the device will be able to interpret the commands using any functionality the user chooses with their own device apps. For testing purposes, we used an app called "Tasker" for Android, which allowed us to connect BLE messages to phone commands. A basic show of our commands is available in Figure 5, while the actual code for the commands is available with the submission. Lastly we wrapped everything together by 3D printing a case for the watch using Tinkercad, along with a purchased strap to attach to our wrists.

The 3D printed case went through multiple iterations where we narrowed in the width, length, and height to be a perfect fit for our internal components without crushing anything. We also used a mostly hollow lid with 3 holes above the microphone to aid in the sensor readings, which vastly improved performance. We also made a second version of the case with a hole in the side so that we could have the device plugged into a computer train the voice detection using the onboard microphone inside the case for the best possible accuracy.

5 Evaluation

As a baseline we were able to accurately gain solid metrics for the voice detection, as we had a data set aside for testing and gaining metrics on data from the neural network. We saw an accuracy of 99.4% on test data with all of the metrics being available in figures 6-7. For the gesture recognition and full usage testing this was much more challenging to gain accurate data on, we performed 21 trials where the user would attempt to activate the keywords on the device, by just speaking the keyword, we tested each word 7 times and had a 100% success rate for each word except for volume and spotify which each had 1 failure leading to a 90% success rate from this test. For the swipe gestures, we bundled them into one group as the swipe left is just the negated version of the swipe right as far as thresholds go, while the shake we measured separately, as it is distinct on its own. For the swiping gestures on 30 trials, we had 2 readings come back as the opposite swipe, where 2 'swipe-left' actions were mistaken as 'swipe-right actions, giving a result of 93%.

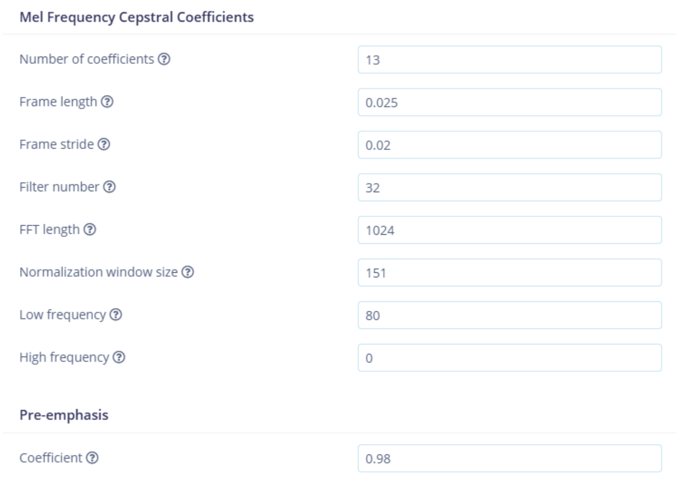
6 Discussion, Limitations, and Future Work

For this project there were several issues that we had to overcome or work around, partially involving our own personal skills, time constraints, and technological limitations. One of the limitations we had was involving the physical case for the watch itself. We had never done 3D modelling for printing before, and the designs that we were able to come up with were the best we could do at our skill level, but many improvements could be made there to allow for better functionality as well as a more appealing design. In the future, using a design where the board is embedded within a rubber band, like the original FitBit, the device would be much more comfortable for users. Another limitation that we faced was the size of our board. We made a choice at the beginning of the project to use a board that was an official Arduino board to avoid complications. This board, however, is much larger than what would be ideal for this project and forced our design to use a much larger footprint. We could have chosen to use smaller boards or build our own microcontroller that was designed specifically for our needs, allowing for a much better size and less power consumption. Within the constraints of time, we ran into issues with the battery, as we originally planned on using 3 CR2032 batteries, but we had issues with the BLE system, causing power spikes that the battery couldn't handle, which then forced us to find a last-minute alternative in a 3.7V Lithium Ion battery, which we had to connect directly to the board itself. We would have liked to introduce a full recharging system where the device itself could be plugged in to enable charging without having to take the battery out to charge on its own. One of the limitations is the choice to put the machine learning model on the board itself. With a much smaller model, it is challenging to have an accurate and diverse number of keywords. We did not have any problems with our current state of the project, but this could have become an issue if we had introduced more keywords. This is something that would be possible to improve by using a board that has more onboard memory to contain a larger model, allowing for better performance. Lastly, one of the improvements we considered was a better way to indicate that a keyword had been recognized; currently, the system we implemented sends a notification to the phone that indicates on screen which mode the phone has entered. This could be improved via haptics with a buzzing sensation when a keyword is triggered or a flashing light somewhere on the case to indicate the device is now listening for gestures.

7 Conclusion

In this project we presented the design and implementation of a self-contained, multi-modal wearable device that addresses the limitations of externally dependent and single-input devices. By fusing offline TinyML voice recognition with deterministic gesture controls, we developed a system that offers flexible, privacy-centric control over external digital systems without the latency or security risks associated with network transmission. Our implementation on the Arduino Nano 33 BLE Sense Rev2 demonstrated that complex sensor fusion and neural network inference can be effectively executed on resource-constrained hardware. The evaluation results, showing over 90% accuracy for both keyword spotting and gesture recognition, validate the effectiveness of our hybrid "voice-to-context, gesture-to-action" interaction model in reducing user cognitive load. Ultimately, this project highlights the viability of edge computing in creating universal, hands-free interfaces that prioritize user data sovereignty while maintaining high responsiveness and intuitive control.

8 Apendix



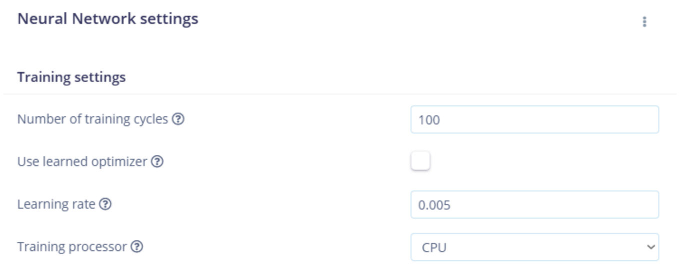
Mel Frequency Cepstral Coefficients

Number of coefficients	13
Frame length	0.025
Frame stride	0.02
Filter number	32
FFT length	1024
Normalization window size	151
Low frequency	80
High frequency	0

Pre-emphasis

Coefficient	0.98

Figure 1



Neural Network settings

Training settings

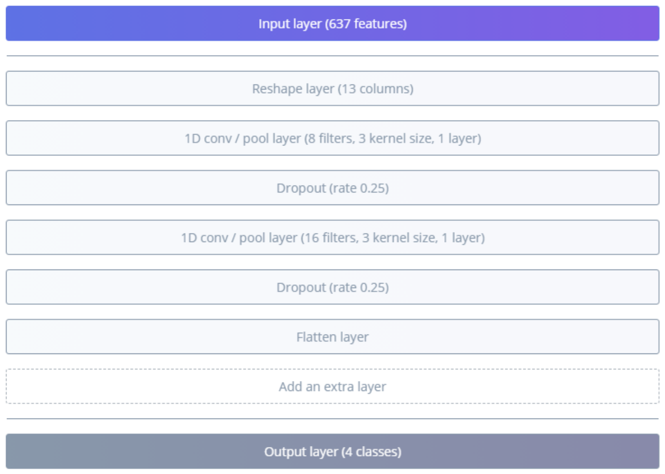
Number of training cycles: 100

Use learned optimizer: ☐

Learning rate: 0.005

Training processor: CPU

Figure 2



Input layer (637 features)

Reshape layer (13 columns)

1D conv / pool layer (8 filters, 3 kernel size, 1 layer)

Dropout (rate 0.25)

1D conv / pool layer (16 filters, 3 kernel size, 1 layer)

Dropout (rate 0.25)

Flatten layer

Add an extra layer

Output layer (4 classes)

Figure 3

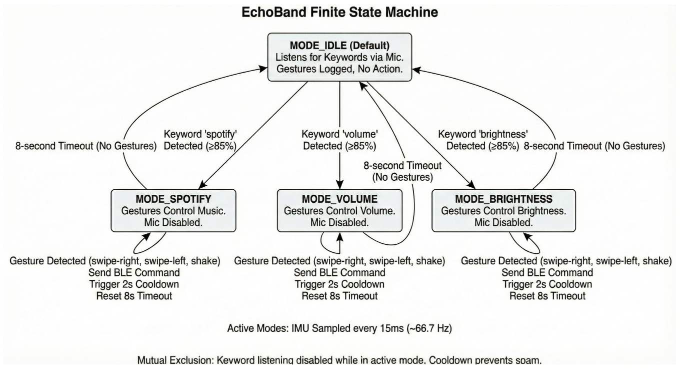
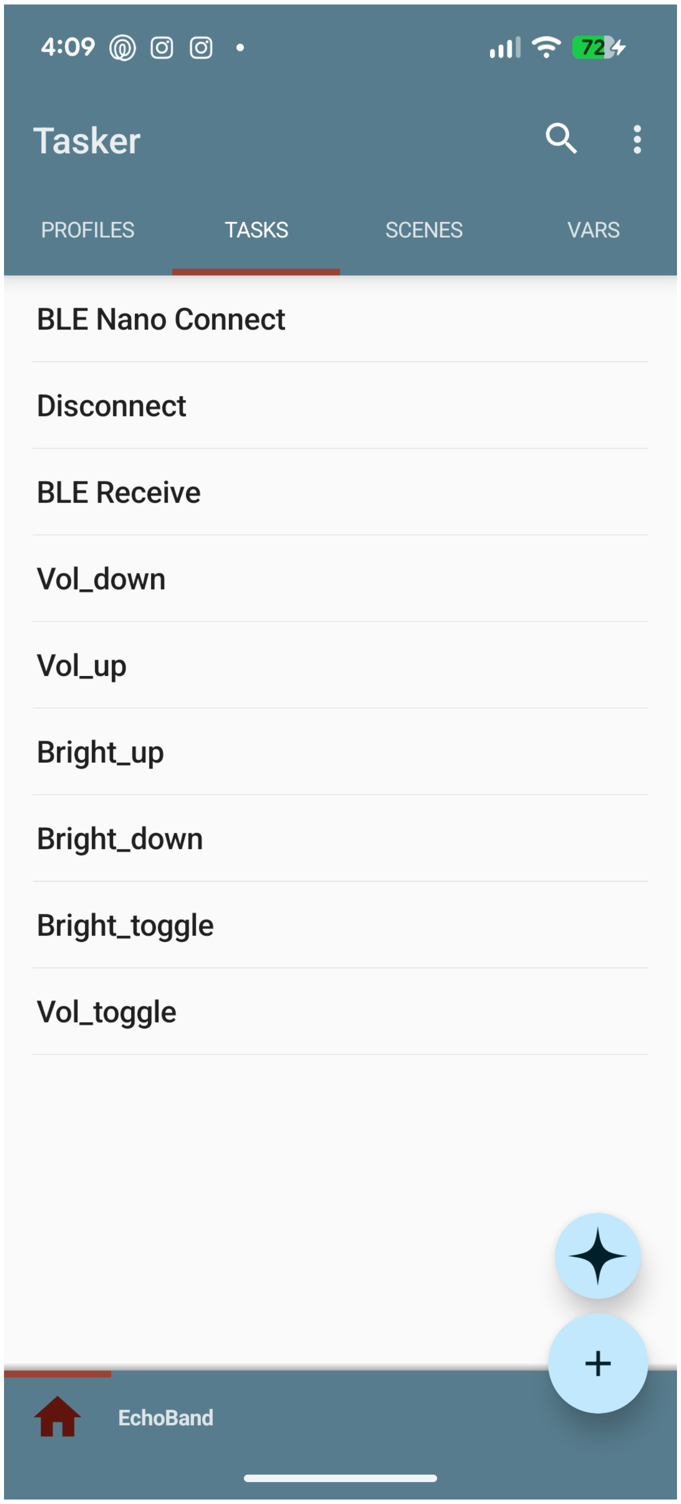

EchoBand Finite State Machine
MODE_IDLE (Default) Listens for Keywords via Mic. Gestures Logged, No Action.
8-second Timeout (No Gestures)
Keyword 'spotify' Detected (≥85%)
Keyword 'volume' Detected (≥85%)
8-second Timeout (No Gestures)
Keyword 'brightness' Detected (≥85%)
8-second Timeout (No Gestures)
MODE_SPOTIFY Gestures Control Music. Mic Disabled.
MODE_VOLUME Gestures Control Volume. Mic Disabled.
MODE_BRIGHTNESS Gestures Control Brightness. Mic Disabled.
Gesture Detected (swipe-right, swipe-left, shake) Send BLE Command Trigger 2s Cooldown Reset 8s Timeout
Gesture Detected (swipe-right, swipe-left, shake) Send BLE Command Trigger 2s Cooldown Reset 8s Timeout
Gesture Detected (swipe-right, swipe-left, shake) Send BLE Command Trigger 2s Cooldown Reset 8s Timeout
Active Modes: IMU Sampled every 15ms (~66.7 Hz)
Mutual Exclusion: Keyword listening disabled while in active mode. Cooldown prevents spam.

Figure 4



BLE Nano Connect

Disconnect

BLE Receive

Vol_down

Vol_up

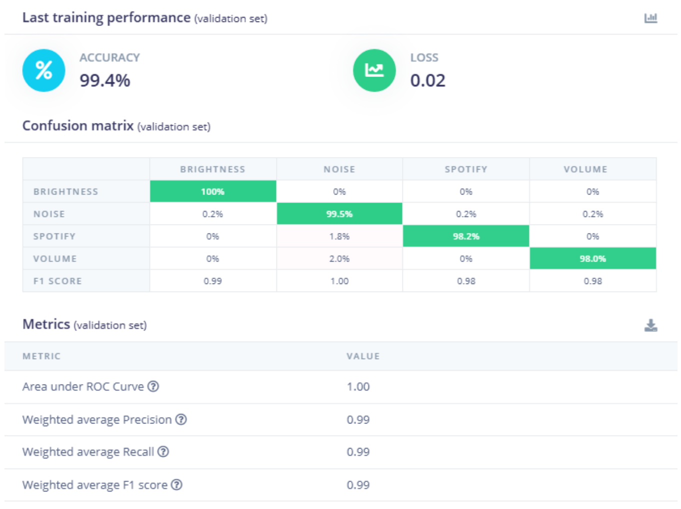
Bright_up

Bright_down

Bright_toggle

Vol_toggle

Figure 5



Last training performance (validation set)

ACCURACY 99.4%

LOSS 0.02

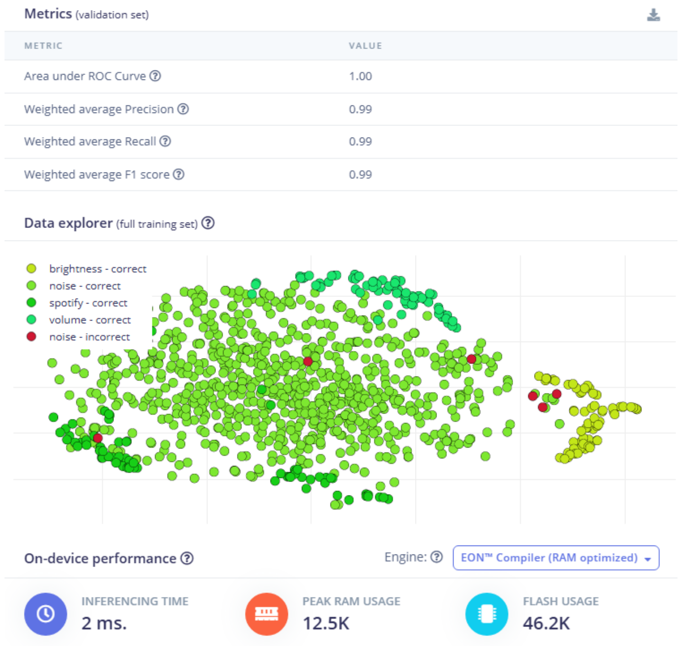
Confusion matrix (validation set)

	BRIGHTNESS	NOISE	SPOTIFY	VOLUME
BRIGHTNESS	100%	0%	0%	0%
NOISE	0.2%	99.5%	0.2%	0.2%
SPOTIFY	0%	1.8%	98.2%	0%
VOLUME	0%	2.0%	0%	98.0%
F1 SCORE	0.99	1.00	0.98	0.98

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.99
Weighted average Recall	0.99
Weighted average F1 score	0.99

Figure 6



Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.99
Weighted average Recall	0.99
Weighted average F1 score	0.99

Data explorer (full training set)

On-device performance

Engine: EON™ Compiler (RAM optimized)

INFERENCING TIME 2 ms.

PEAK RAM USAGE 12.5K

FLASH USAGE 46.2K

Figure 7

References

[1] C. Banbury et al. 2021. Micronets: Neural Network Architectures for Deploying TinyML Applications. *MLSYS (Proceedings?)* (2021). https://arxiv.org/abs/2010.11267

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. ACM. https://dl.acm.org/doi/10.1145/2342509.2342513

[3] Sai Srujana Buddi et al. 2023. Efficient Multimodal Neural Networks for Triggerless Voice Assistants. In *Proc. INTERSPEECH 2023*. https://www.isca-speech.org/archive/interspeech_2023/buddi23_interspeech.pdf

[4] J. Chen et al. 2021. Deep Learning on Edge: A Review. *IEEE Access* (2021). https://ieeexplore.ieee.org/document/8763885

[5] C. Gomez et al. 2012. Bluetooth Low Energy for the Internet of Things. *IEEE Wireless Communications* (2012). https://www.mdpi.com/1424-8220/12/9/11734

[6] Andreas Jacobsson, Martin Boldt, and Bengt Carlsson. 2016. A Risk Analysis of Simple Smart Home Automation Systems. *Future Generation Computer Systems* (2016). https://www.diva-portal.org/smash/get/diva2:1419600/FULLTEXT01.pdf

[7] Athanasios Katsamanis et al. 2017. Multimodal gesture recognition. *IEEE Access* (2017). https://dl.acm.org/doi/10.1145/3015783.3015796

[8] G. Laput et al. 2019. Sensing Fine-Grained Hand Activity with Smartwatches. In *Proc. CHI 2019*. https://dl.acm.org/doi/10.1145/3290605.3300568

[9] S. Lee et al. 2016. Point-n-Press: An Intelligent Universal Remote Control System. *IEEE Transactions on Automation Science and Engineering* (2016). https://ieeexplore.ieee.org/document/7442594

[10] Vassilis Pitsikalis et al. 2015. Multimodal Gesture Recognition via Multiple Hypotheses Rescoring. *Journal of Machine Learning Research* (2015). https://jmlr.org/papers/v16/pitsikalis15a.html

[11] Stefano Pizzolato, Luca Tagliapietra, Matteo Cognolato, Monica Reggiani, Henning Müller, and Manfredo Atzori. 2017. Comparison of six electromyography acquisition setups on hand movement classification tasks. *PLOS ONE* (2017). https://doi.org/10.1371/journal.pone.0186132

[12] P. P. Ray. 2022. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University — Computer and Information Sciences* (2022). https://doi.org/10.1016/j.jksuci.2021.11.019

[13] M. Smith et al. 2020. Evaluating the Scalability of Non-Preferred Hand Mode Switching in Augmented Reality. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI)*. https://dl.acm.org/doi/pdf/10.1145/3399715.3399850

[14] Matthew Turk. 2014. Multimodal interaction: A review. *Pattern Recognition Letters* (2014). doi:10.1016/j.patrec.2013.07.003

[15] P. Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint* arXiv:1804.03209 (2018). https://arxiv.org/abs/1804.03209

[16] Wearable Devices Ltd. 2024. *Interaction Beyond Boundaries: Integrating Wearable Neural Interfaces.* Technical Report. Wearable Devices Ltd. https://www.wearabledevices.co.il/post/interaction-beyond-boundaries-integrating-gesture-recognition-wearables-and-cameras-for-enhanced-ux

[17] C. Zhang et al. 2018. FingerPing: Recognizing Fine-grained Hand Poses using Active Acoustic Sensing. In *Proc. MobiCom 2018*. https://dl.acm.org/doi/10.1145/3173574.3174011

[18] X. Zhang et al. 2024. Emerging Wearable Interfaces and Algorithms for Hand Gesture Recognition: A Survey. *IEEE Reviews in Biomedical Engineering* (2024). https://ieeexplore.ieee.org/document/9426433

[19] Y. Zhang et al. 2017. Hello Edge: Keyword Spotting on Microcontrollers. *arXiv preprint* arXiv:1711.07128 (2017). https://arxiv.org/abs/1711.07128

[20] Y. Zhang et al. 2022. Hand Gesture Recognition on a Resource-Limited Microcontroller. *Sensors* (2022). https://www.mdpi.com/1424-8220/21/17/5713