Revision Summary

We would like to thank the reviewers for their careful reading of our manuscript *"EdgeEMG: On-Device Neural Network Training for Real-Time EMG Pattern Recognition"* and for their constructive feedback. Their comments have been very helpful in improving the quality, clarity, and rigor of our paper. Below, we address each reviewer's concerns point by point and summarize the revisions we have made. A tracked changes PDF is attached after this summary.

Reviewer 3YKA

Weakness 1: The paper frames EdgeEMG as a compact, low-power system, but does not provide a power analysis of the devices used. Placement of devices (ESP32 and Sony Spresense) is unclear.

Revision: We thank the reviewer for pointing out the lack of power analysis and clarification of device placement. We now report power consumption for the Myo Armband (40.1 mW) and Sony Spresense (24.79 mW), totaling ~65 mW. We also clarified that the devices were kept in participants' pockets during use, demonstrating both portability and suitability for wearable applications.

Weakness 2: The paper compares EdgeEMG only to an LDA baseline, which is outdated compared to state-of-the-art deep learning methods.

Revision: We appreciate the reviewer's concern and have expanded the introduction to explain that models pretrained on GPUs lose significant accuracy when fine-tuned on individuals, making generalized deployment impractical. This underscores the need for user-specific on-device training. We also justified our use of LDA as a baseline by citing its prevalence in sEMG research. Since we are the first group doing on-device training of sEMG signal with deep-learning model, it is reasonable to compare it with the LDA model under similar hardware constraints rather than the model trained on powerful GPUs.

Reviewer rSGf

Weakness 1: Only four gestures and four participants were used; scalability and generalization are not addressed.

Revision: We thank the reviewer for this suggestion. We will try to expand our system to more gestures and larger user groups in our future research to further explore the possibilities of on-device training applications.

Weakness 2: No statistical analysis was included to validate performance improvements.

Revision: We agree and thank the reviewer for this observation. We have now included a two-sample z-test comparing our FNN to LDA, which yielded a z-score of 4.53 and p-value of 6×10^{-6}, demonstrating that our results are statistically significant.

Weakness 3: Power consumption metrics, critical for wearable applications, were not reported.

Revision: We thank the reviewer for highlighting this important omission. We now provide detailed runtime power measurements (Myo Armband: 40.1 mW, Sony Spresense: 24.79 mW), which validate the system's low-power suitability for wearable applications.

Reviewer hYW9

Weakness 1: Unclear whether the model was trained per participant or on combined data.

Revision: We thank the reviewer for this helpful clarification request. We have updated the evaluation section to state explicitly that each model was trained and tested independently per participant, which emphasized our system's focus on user-specific adaptation.

Weakness 2: The choice of LDA as a baseline was not well justified.

Revision: We thank the reviewer for raising this concern. We added justification in the methods section, explaining that LDA is widely used in sEMG research as a lightweight benchmark, making it an appropriate baseline for demonstrating the value of our on-device training approach.

Weakness 3: Results presentation could be improved; offline and real-time accuracies were difficult to compare.

Revision: We appreciate the reviewer's suggestion. However since real-time testing involves hyperparameter tuning, the variables are not controlled compared to the offline testing and therefore can not be put into a single chart for comparison.

Weakness 4 : Model can be tested on users whom it did not train on to assess its real-world adaptation to new users.

Revision: We thank the reviewer for highlighting potential experiments. Our model mainly focuses on on-device training, which indicates that new users are supposed to retrain the model with their own sEMG signals rather than using a model that is trained on someone else. Due to the lightweight and portable nature of the model, it takes little computational resource to finish the training process, which allows new users to use the personalized model as soon as they want to.

We sincerely thank all reviewers for their detailed and constructive feedback. Their comments have helped us improve the clarity, rigor, and presentation of our work. We believe the revisions strengthen the paper and better highlight the significance of EdgeEMG as a practical, low-power, user-specific solution for sEMG-based gesture recognition.

EdgeEMG: On-Device Neural Network Training for Real-Time EMG Pattern Recognition

Jiahua Tang[1], Philip Liang[2], Zhuwei Qin[2]
[1]Lynbrook High School, San Jose, USA
[2]School of Engineering, San Francisco State University, San Francisco, USA

***Abstract*—Surface electromyography (sEMG) is a non-invasive technique that records bioelectrical signals generated by muscle activity via electrodes placed on the skin. Its ability to capture a user's motor intent in real time has enabled a wide range of applications, including prosthetic control, rehabilitation robotics, and human-computer interaction. Recent advances in machine learning (ML), particularly deep learning (DL), have enabled automated processing of complex biosignals. While DL-based approaches for sEMG gesture recognition have shown strong performance on embedded systems, they typically rely on pre-training models on high-performance computing platforms (e.g., PCs or supercomputers) before deployment to low-power devices. This off-device pretraining limits portability and adaptability, as it requires prior collection and processing of sEMG data on non-portable hardware. In this paper, we present EdgeEMG, the first fully on-device training approach for sEMG gesture recognition using a deep neural network. Our approach is implemented using the AIfES library, developed by the Fraunhofer Institute, which provides efficient operations for feedforward neural networks and supports deployment on resource-constrained microcontrollers. We validate our system on the Sony Spresense MCU and benchmark it against a conventional Linear Discriminant Analysis (LDA) classifier. Experimental results demonstrate that our approach achieves an average real-time accuracy of 70% across different hand gestures. These findings highlight the feasibility of real-time, user-adaptive EMG decoding entirely on embedded hardware, without reliance on external compute resources.**

***Index Terms*—Gesture Recognition, sEMG, Machine Learning, Neural Network**

I. Introduction

Surface Electromyography (sEMG) is a widely used non-invasive technique that records the electrical potentials generated by muscle fibers during contraction. Unlike intramuscular EMG, which requires needle electrodes, sEMG uses electrodes placed on the surface of the skin, making it more practical and less intrusive for continuous and wearable use. The signals reflect the underlying motor unit activity and have been shown to correlate strongly with intended motion and muscular force. Due to these properties, sEMG has become a critical sensing modality in a range of applications, including prosthetic control, neuromuscular rehabilitation, exoskeleton actuation, and gesture-based human-computer interaction [1].

Interpreting sEMG signals for real-time control is challenging due to their stochastic, non-stationary, and subject-specific nature. Historically, sEMG pattern recognition systems have relied on traditional machine learning (ML) techniques, such as Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), etc. These models typically operate on handcrafted features extracted from time or frequency domains, such as root mean square (RMS), waveform length, zero crossings, and autoregressive coefficients. While effective in controlled settings, these approaches often struggle to generalize across users or varying recording conditions. They also require manual feature engineering, which limits scalability and adaptability.

In recent years, deep learning (DL) has emerged as a powerful alternative, enabling end-to-end learning of features and representations directly from raw or lightly processed sEMG data. Architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers have demonstrated superior performance in decoding motor intent, particularly when large, high-quality datasets are available. These models learn hierarchical, task-specific features through gradient-based optimization, typically using backpropagation and variants of stochastic gradient descent to minimize a supervised loss function. However, DL-based approaches introduce substantial computational demands. Training involves iterative updates to millions of parameters through large-scale matrix operations, making them computationally intensive and memory-hungry. Consequently, model training is usually performed on high-performance GPUs or cloud platforms. Then, trained models are often compressed using techniques like quantization and model pruning for deployment.

For example, Nguyen et al. developed a portable neuroprosthetic hand with deep learning-based finger control, deploying an RNN on the NVIDIA Jetson Nano [2]. Lu et al. demonstrated the feasibility of deploying a CNN-based EMG recognition model on the microcontroller, utilizing 8-channel EMG data along with 8-bit quantization and transfer learning to enable efficient inference on resource-constrained hardware [3]. Although the models achieved high accuracy, initial training was conducted on a desktop GPU, limiting the system's portability and autonomy.

Beyond hardware limitations, a persistent challenge in sEMG-based systems is robust generalization across users and recording conditions. Performance often deteriorates when a model trained on data from a specific subject is applied to a new user or when the recording setup changes due to electrode displacement, muscle fatigue, perspiration, or external noise. This issue arises because sEMG signals are highly sensitive to individual physiological characteristics, such as muscle com-

Supported by Sony Sensing Solution University Program.

position, skin impedance, and electrode placement. Additionally, signal properties may vary within the same user over time, a phenomenon known as intra-subject variability. Even GPU-pretrained models degrade significantly after user-specific fine-tuning, wasting computational resources and yielding accuracy too unreliable for practical use

Addressing this robustness issue has become a focal point in recent research. EMGSense [4] for instance, adopts a self-supervised domain adaptation strategy combined with data augmentation to account for temporal and inter-user variability. By leveraging unlabeled EMG data from new users, the system learns user-specific representations that adapt the model without requiring manual labeling. RoHDE [5] uses a generative adversarial network (GAN) to synthesize realistic, noise-augmented sEMG signals that simulate common artifacts like electrode shift and signal dropout. This data augmentation strategy improves model tolerance to signal distortions and enhances generalization in practical settings. Despite these advances, many of these techniques still depend on access to high-performance training infrastructure and extensive calibration data. Moreover, fully user-independent generalization, where a model trained on data from one group of users performs reliably on unseen users without fine-tuning remains an unsolved problem. These limitations motivate the development of more robust, efficient, and adaptive sEMG decoding systems.

In this work, we introduce *EdgeEMG*, the first fully self-contained and truly portable deep learning system capable of performing both on-device training and real-time inference on embedded hardware. Unlike prior approaches that rely on external high-performance computing resources for model training, EdgeEMG implements a complete training and inference pipeline directly on the microcontroller using a lightweight feedforward neural network (FNN). This approach enables autonomous, real-time adaptation to new users without requiring connectivity to a host device or server.

To realize this capability, we leverage the AIfES (Artificial Intelligence for Embedded Systems)-Express library developed by Fraunhofer IMS [6], which provides an optimized neural network (NN) runtime tailored for ultra-low-power microcontrollers. Our model is deployed on the Sony Spresense platform, a compact embedded system featuring a multi-core ARM Cortex-M4F processor. The EdgeEMG system supports end-to-end operation—from sEMG signal acquisition to NN training and inference—entirely on-device, with no reliance on external computational infrastructure. This enables rapid, user-specific calibration in the field and opens new opportunities for deploying adaptive neural-machine interfaces in resource-constrained environments.

Experimental results demonstrate that EdgeEMG achieves an average offline classification accuracy of 75% and a real-time accuracy of 70% across four hand gestures, with a total training time of just 30 seconds. To assess the effectiveness of our approach, we benchmarked EdgeEMG against a conventional on-device ML baseline employing feature-based classification. Our FNN model outperformed the baseline in both accuracy and adaptability, despite operating directly on raw sEMG signals without the need for handcrafted feature extraction. This end-to-end learning capability significantly reduces system complexity and enables faster user adaptation, making EdgeEMG a practical, low-cost solution for real-world assistive and wearable technologies.

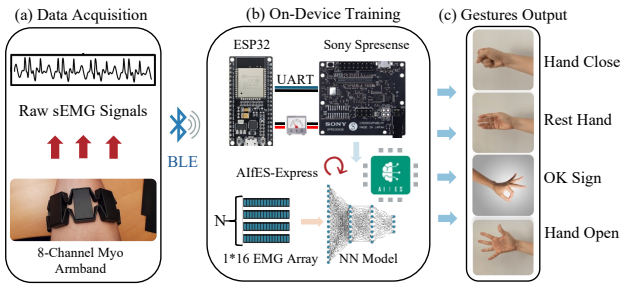

Fig. 1. *EdgeEMG* System Achitecture. The Sony Spresense serves as the primary computing platform for executing the on-device training algorithm.

II. Methodology and System Overview

A. Data Acquisition and Pre-processing

During the data acquisition phase, raw sEMG signals are sampled at 200 Hz using the Myo Armband. As illustrated in Fig. 1(a), the Myo Armband features eight dry electrodes arranged radially around the user's forearm to capture muscle activity from multiple channels. The sensor streams 8-channel sEMG data via Bluetooth Low Energy (BLE) in a compact, low-power format suitable for wearable applications.

The sEMG data packets, each containing two consecutive timestamps of 8-channel readings, are received by an ESP32 microcontroller, which acts as a BLE receiver. The ESP32 then forwards the decoded data via UART communication protocol with a shared ground reference to maintain signal integrity. The Sony Spresense microcontroller handles all downstream signal processing, model training, and inference tasks, enabling a fully embedded system architecture without external computation. The wearable system consumes only around 65 mW (Myo armband 40.1 mW, Sony Spresense 24.79 mW), making it orders of magnitude more energy-efficient than training equivalent models on a high-performance GPU such as the NVIDIA A100 (around 350 W).

Upon reception, the sEMG signals undergo a pre-processing stage before being passed to the NN. Each channel is standardized using its corresponding mean and standard deviation, calculated from a short calibration buffer. Specifically, the standardized signal x_i^{new} is computed as: $x_i^{\text{new}} = \frac{x_i - x_{\text{mean}}}{x_{\text{std}}}$, where x_i^{new} denotes the standardized value of the raw input x_{mean} is the mean, and x_{std} is the standard deviation of the corresponding sEMG channel. These normalization parameters are initially computed using data from the publicly available Ninapro DB5 sparse sEMG database [7], which provides a representative prior distribution for initialization.

B. Model Training

To prepare training inputs, we recorded 20 sets of standardized sEMG data per gesture. Each data sample consists of a

temporally flattened 1 × 16 array: 8 channels × 2 timestamps. These fixed-size vectors are sequentially fed into the NN model for training. This compact representation ensures compatibility with the limited memory and computational capacity of the embedded hardware, while still capturing sufficient spatiotemporal information to discriminate between gestures.

The NN model used in EdgeEMG is implemented using the AIfES-Express library, a lightweight NN framework written in Embedded-C specifically designed for resource-constrained microcontrollers. AIfES-Express provides highly optimized APIs for matrix operations, activation functions, and learning algorithms, enabling the deployment of otherwise compute-intensive deep learning models on ultra-low-power platforms such as the Sony Spresense.

As shown in Fig 1(b), our NN model architecture is a 4-layer fully connected FNN. The input layer consists of 16 nodes, corresponding to a flattened vector of 8 channels × 2 timestamps of standardized sEMG data. This is followed by two hidden layers, each with 8 neurons and sigmoid activation functions, selected for their compact representation and compatibility with fixed-point operations. The final output layer contains 4 neurons, each representing one of the four gesture classes. A softmax function is applied to the output layer to compute class-wise prediction probabilities, providing a confidence distribution over the gesture classes.

The model is trained entirely on-device using the Adam optimizer, which combines momentum and adaptive learning rate updates for efficient convergence. The training objective is to minimize the categorical cross-entropy loss, a standard choice for multi-class classification tasks. A fixed learning rate of 0.1 is used, which was empirically selected to balance convergence speed and model stability given the limited computational precision on embedded hardware. Once all training samples have been processed, the model parameters are stored in memory, enabling immediate transition to real-time inference.

C. Real-Time Inference

In the inference phase, the EdgeEMG system performs gesture recognition in real time using the trained on-device model. For each prediction cycle, the system collects a new set of 16 raw sEMG values from the Myo Armband (i.e., 8 channels × 2 timestamps), which are then standardized using the previously computed mean and standard deviation values for each channel.

The resulting normalized input vector is passed through the trained FNN model, which computes a set of prediction confidence scores. The system then selects the gesture associated with the highest confidence score as the final predicted output. This classification result is available with minimal latency, enabling real-time, closed-loop interaction for downstream applications such as gesture-controlled prosthetic devices or wearable user interfaces.

III. Experiments and Results

A. Experiment Setup

We evaluated the EdgeEMG system using the Myo Armband, which was securely positioned around the thickest part of the user's forearm to ensure consistent electrode contact. The Sony Spresense and ESP32 can be placed in the user's pockets due to their portable sizes. During the training phase, the user was prompted via the serial monitor to perform four predefined hand gestures. For each gesture, the system collected 20 sets of 16 raw sEMG samples (8 channels × 2 timestamps per sample). Following training, the system automatically transitioned into real-time inference mode, in which it continuously collected and standardized 16 sEMG samples per prediction cycle and passed them to the trained model to obtain class-wise prediction confidence scores. Experiments were conducted with four able-bodied participants (2 male, 2 female; ages 23–68). Each participant was instructed to perform and hold each gesture for 5 seconds, followed by a 5-second rest interval. The model was trained and tested on each of the four users independently to better demonstrate how on-device training tackles inter-user variability. In addition, we performed hyperparameter tuning experiments to assess the trade-offs between model complexity, training duration, and accuracy. Parameters varied included the optimizer, number of training epochs, input window size, number of hidden layers, and number of nodes per layer.

B. Offline Recognition Accuracy and Model Comparison

To evaluate the model's offline performance, we collected additional test data for each gesture during the training session. These test samples were withheld from training and later used to evaluate classification accuracy.

Table I shows the average offline testing accuracies of the NN model across four gesture classes: Rest, Hand Close, Hand Open, and OK Sign. Each row represents the ground truth label for the test set, while each column indicates the model's predicted class. Diagonal entries reflect correct classifications, with the highest accuracy observed for Rest (80%), followed by Hand Close (77%), OK Sign (76%), and Hand Open (67%). The model achieved an overall average offline testing accuracy of 75%, with a corresponding training accuracy of 83%. Misclassifications, indicated by the off-diagonal entries, remain relatively low across all gesture pairs. These results confirm the

TABLE I
Deep Learning Model's Average Offline Testing Accuracies

	Rest	Hand Close	Hand Open	OK Sign
Rest Test Set	80%	3%	10%	7%
Hand Close Test Set	4%	77%	6%	13%
Hand Open Test Set	11%	10%	67%	12%
OK Sign Test Set	8%	7%	9%	76%

TABLE II
LDA Model's Average Offline Testing Accuracies

	Rest	Hand Close	Hand Open	OK Sign
Rest Test Set	72%	8%	11%	9%
Hand Close Test Set	11%	53%	16%	20%
Hand Open Test Set	17%	10%	58%	15%
OK Sign Test Set	11%	14%	18%	57%

model's effectiveness in learning and distinguishing discrete sEMG patterns during offline evaluation.

We also compared its performance against a conventional ML model baseline: Linear Discriminant Analysis (LDA), which is widely used in sEMG-related research [8]. The LDA model was implemented in Python and trained on the same dataset. As shown in Table II, the LDA model achieved a lower average test accuracy of 60%. To evaluate whether this difference was statistically significant, we performed a two-sample z-test (H_0: the FNN model does not improve cross-user accuracy; H_A: the FNN model improves cross-user accuracy). The test yielded a z-score of 4.53 and a p-value of $6 * 10^{-6}$, which is well below the 0.05 significance threshold. We therefore reject the null hypothesis, confirming that the improvements achieved by our FNN model are statistically significant. This highlights the advantage of end-to-end learning directly from raw sEMG.

C. Real-Time Accuracy and Hyperparameter Tuning

Real-time inference accuracy across the four participants is shown in Table III. Each row corresponds to an individual participant, and each column reflects the model's classification accuracy for a specific gesture during real-time inference. The Rest gesture consistently achieved the highest accuracy across all subjects, peaking at 85% for Subject 1. The Hand Open gesture showed slightly lower and more variable performance, with accuracies ranging from 59% to 67%. Overall, the model maintained an average real-time accuracy of approximately 70%, demonstrating reliable on-device prediction performance across users in real-world conditions.

Fig. 2 illustrates the impact of various NN architecture configurations on training time and classification accuracy over 200 training epochs. Four combinations of hidden layers and node counts were evaluated: 1 hidden layer with 4 nodes, 1 hidden layer with 8 nodes, 2 hidden layers with 4 nodes, and 2 hidden layers with 8 nodes. The blue bars represent training time (left y-axis), while the red line denotes classification accuracy (right y-axis). As expected, increasing model complexity—either through additional hidden layers or more nodes—results in longer training durations. The configuration with 2 hidden layers and 8 hidden nodes yielded the highest accuracy, at the cost of the longest training time (∼30 seconds). In contrast, the smallest configuration (1 layer, 4 nodes) trained fastest (∼15 seconds) but achieved lower accuracy. These results highlight a trade-off between model expressiveness and computational efficiency, with the 2-layer, 8-node architecture offering the best balance for real-time, on-device training under constrained hardware conditions.

TABLE III
AVERAGE REAL-TIME PREDICTION ACCURACIES FOR EACH GESTURE

	Rest	Hand Close	Hand Open	OK Sign
Test Subject 1	85%	72%	59%	67%
Test Subject 2	74%	71%	65%	72%
Test Subject 3	77%	70%	62%	68%
Test Subject 4	73%	68%	67%	70%

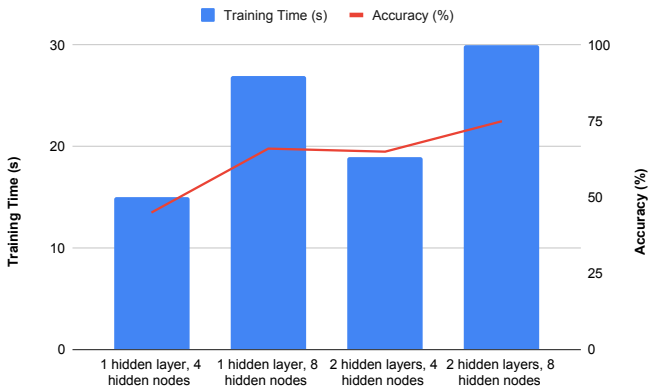

Fig. 2. Experimental Results of Hyperparameter Combinations for 200 Training Epochs

IV. CONCLUSION AND FUTURE WORK

In this work, we have developed the first on-device NN training and inference for sEMG gesture recognition entirely in Embedded-C, without reliance on external computational resources. Our approach demonstrates the feasibility of deploying a fully self-contained, real-time, and adaptive gesture recognition system on low-power microcontrollers. Through a series of offline and real-time experiments, we verified that our on-device NN model outperforms a conventional ML baseline in classification accuracy. Notably, our model operates directly on raw sEMG signals, eliminating the need for handcrafted feature extraction and reducing both implementation complexity and training time.

REFERENCES

[1] Chiang Liang Kok, Chee Kit Ho, Fu Kai Tan, and Yit Yan Koh, "Machine Learning-Based Feature Extraction and Classification of EMG Signals for Intuitive Prosthetic Control," Applied sciences, vol. 14, no. 13, pp. 5784–5784, Jul. 2024, doi: https://doi.org/10.3390/app14135784.

[2] A. T. Nguyen et al., "A portable, self-contained neuroprosthetic hand with deep learning-based finger control," Journal of Neural Engineering, vol. 18, no. 5, p. 056051, Oct. 2021, doi: https://doi.org/10.1088/1741-2552/ac2a8d.

[3] J. Lu, et al., "EffiE: Efficient Convolutional Neural Network for Real-Time EMG Pattern Recognition System on Edge Devices," Apr. 2023, doi: https://doi.org/10.1109/ner52421.2023.10123741.

[4] D. Duan et al., "EMGSense: A low-effort self-supervised domain adaptation framework for EMG Sensing," 2023 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 160–170, Mar. 2023. doi:10.1109/percom56429.2023.10099164

[5] Z. Lin, Z. Qiu, J. Tang, L. Liu, and S. Zhang, "RoHDE: A robust high-dimensional sEMG representation enhanced via GAN-based data augmentation," in Proceedings of the IEEE International Conference on Medical Computing and Bioinformatics (IMCB), 2022, pp. 19–24. [Online]. Available: https://doi.org/10.1109/IMCB55806.2022.00011

[6] Fraunhofer, "AIFES®-Express Tutorial (Feedforward Neural Network-FLOAT32)," *Arduino Project Hub*, Oct. 15, 2021. [Online]. Available: https://projecthub.arduino.cc/aifes_team/6cbb7e4d-4a70-4790-bc4e-3baca76bef00?ref=424_recent___&offset=1

[7] S. Pizzolato, L. Tagliapietra, M. Cognolato, M. Reggiani, H. Müller, and M. Atzori, "Comparison of six electromyography acquisition setups on hand movement classification tasks," PLOS ONE, vol. 12, no. 10, p. e0186132, Oct. 2017, doi: https://doi.org/10.1371/journal.pone.0186132.

[8] Donovan, Ian M., et al. "Simple space-domain features for low-resolution sEMG pattern recognition." 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2017.