EffiE: **Effi**cient Convolutional Neural Network for Real-Time **E**MG Pattern Recognition System on Edge Devices

Jimmy Lu[1], Philip Liang[2], Jin Chul Rhim[2], Xiaorong Zhang[2], *Senior Member, IEEE*, Zhuwei Qin[2], *Member, IEEE*

2023 11th International IEEE/EMBS Conference on Neural Engineering (NER) | 978-1-6654-6292-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/NER52421.2023.10123741

***Abstract*—With the advancement of deep learning (DL) technologies, applying DL methods to processing surface electromyographic (sEMG) signals for movement intent recognition has gained increasing interest in the research community. Compared to conventional non-DL methods commonly used for EMG pattern recognition (PR), DL algorithms have the advantage of automatically extracting sEMG features without the cumbersome manual feature engineering step and are especially effective in processing sEMG signals collected from 1-dimentional (1D) or 2D sensor arrays. However, a key challenge to the deployment of DL methods in sEMG-controlled neural-machine interface (NMI) applications (e.g., myoelectric controlled prostheses) is the high computational cost associated with DL algorithms (*e.g.*, convolutional neural network (CNN)) since most NMI applications need to be implemented on resource-constrained embedded computer systems and have real-time requirements. In this paper, we designed and implemented *EffiE* – an efficient CNN for real-time EMG PR system on edge devices. The development of the *EffiE* system integrated several strategies including a deep transfer learning strategy to adaptively and quickly update the pre-trained CNN model based on the user's newly collected data on the edge device, and a deep learning quantization method that can dramatically reduce the memory consumption and computational load of the CNN model without sacrificing the model accuracy. The proposed *EffiE* system has been implemented on a Sony Spresense 6-core microcontroller board as a working prototype for real-time NMIs. The embedded NMI prototype has integrated input/output interfaces as well as efficient memory management and precise timing control schemes to achieve real-time DL-based myoelectric control of a bionic arm using hand gestures. We released all the source code at: https://github.com/MIC-Laboratory/EffiE**

I. Introduction

Surface electromyography (sEMG)-based neural-machine interfaces (NMIs) measure myoelectric signals from the surface of human skin, interpret the signals to identify human movement intentions, and then make decisions to control external applications such as neural prostheses which restore function for patients with limb loss or impairment [1]. Recently, with the advancement of deep learning (DL) technologies, applying DL methods to sEMG-based movement intent recognition has gained increasing interest in the research community. However, utilizing DL often comes at a cost of high computational burdens, which typically relies on high-performance hardware, such as GPU to run DL model inference in real-time. Various deep learning methods such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and hybrid CNN-RNN methods have been exploited for sEMG-based gesture recognition and have shown promising results [2]–[4]. Compared to typical machine learning-based methods for EMG pattern recognition (PR), DL algorithms have the advantage of automatically extracting HD EMG features without the cumbersome manual feature engineering step and are especially effective in processing sEMG signals collected from 1-dimensional (1D) or 2D sensor arrays. However, a key challenge to the deployment of DL methods in sEMG-controlled NMI applications (e.g., myoelectric controlled prostheses) is the high computational cost associated with DL algorithms (e.g., CNN) since most NMI applications need to be implemented on resource-constrained embedded computer systems and have real-time requirements. Shen *et al*. designed an sEMG gesture recognition system using deep learning based on wearable device [5]. Despite having a portable sEMG acquisition system, the deep learning model inference is still computed on the desktop computer, which might not be applicable in real-world use cases for portable settings. Recently, Nguyen *et al*. designed a system of neuroprosthetic hand leveraging the capabilities of both CNN and RNN computed via the Nvidia Jetson computer with embedded GPU [6]. This system provides both accurate and portable prosthesis control in medical settings by using an implanted EMG sensor. However, the power consumption of the onboard GPU may quickly drain the battery.

To solve the conflict between the heavy computation cost of the DL model and limited on-device hardware resources, many previous works have been proposed to compress and accelerate CNNs for edge devices [7]–[9], and DL quantization is considered as one of the most hardware-friendly approaches [10]. DL quantization methods convert the DL model parameters and inputs from float point into integer data types, which can significantly reduce the computation load and memory occupation while retaining comparable prediction accuracy. Efficient and effective as they are, DL quantization is barely explored in the NMI applications.

This work is supported by the National Science Foundation (NSF #1752255)

[1]Jimmy Lu is with the Lowell High School, 1101 Eucalyptus Dr, San Francisco, CA 94132.

[2]Philip Liang, Jin Chul Rhim, Xiaorong Zhang, and Zhuwei Qin* are with the School of Engineering, San Francisco State University, 1600 Holloway Ave, San Francisco, CA 94132. *Correspondence: zwqin@sfsu.edu

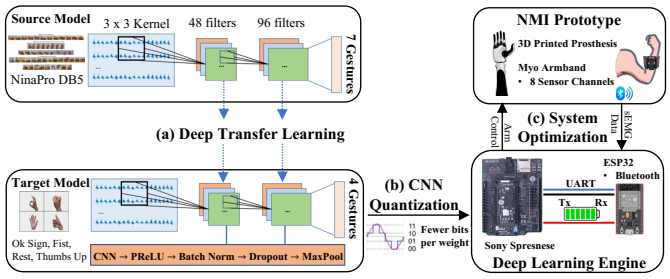

Fig. 1. *EffiE* System: (a) Deep transfer Learning, (b) CNN Quantization, (c) System Optimization.

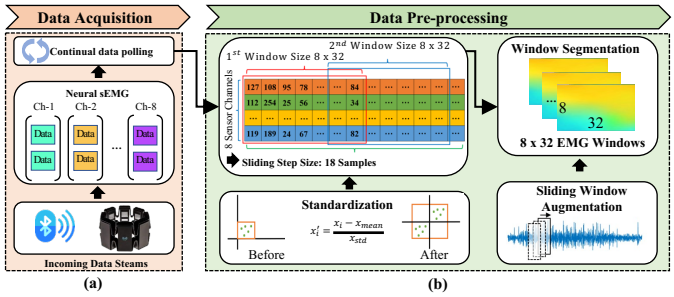

Fig. 2. sEMG Data Acquisition and Pre-processing with Sliding Window Augmentation.

In this paper, we proposed *EffiE* – an Efficient CNN for a real-time EMG pattern recognition system on an edge device by integrating deep transfer learning and quantization. For the sEMG pattern recognition, we utilized a 2-dimensional CNN to extract spatial domain features from real-time sEMG signals. To adaptively and quickly update the pre-trained CNN model based on the user's newly collected data on the edge device, we applied deep transfer learning on our CNN model, which generalizes sEMG feature learning from a larger sEMG dataset and adapts to real-time collected user data. To further boost the system performance, we introduced CNN model quantization that can dramatically reduce memory occupation and computational load without sacrificing the model's accuracy. The proposed *EffiE* system has been implemented on a Sony Spresense 6-core microcontroller as a working prototype for real-time NMIs. The embedded NMI prototype has integrated input/output interfaces as well as efficient memory management and precise timing control schemes to achieve real-time DL-based myoelectric control of a bionic arm using hand gestures. The experiments show that our proposed NMI system can meet real-time processing latency of 160 ms while retaining optimal accuracy on the edge device.

II. Method

A. System Overview

Fig. 1 shows the proposed *EffiE* – an efficient CNN for real-time EMG pattern recognition system, which contains of three major parts: (a) Adapting pre-trained CNN model on the user's newly collected data by Deep Transfer Learning, (b) Reducing the memory occupation and computation latency for edge devices by CNN Quantization, and (c) Optimizing system performance to achieve real-time DL-based myoelectric control of a bionic arm using hand gestures by efficient memory management and precise timing control schemes. In the following sections, we will introduce the *EffiE* in three parts: sEMG Data Pre-processing, sEMG Feature Extraction with Transfer Learning, and System Optimization for Real-time Performance.

B. sEMG Data Pre-processing

Fig. 2 shows an overview of the data processing program deployed on the 6-core Sony Spresense edge devices [11], which consists of two separate threads of data acquisition and pre-processing. Such multi-threading implementation maximizes the utilization of the microcontroller's computational resources while minimizing processing latency.

As shown in the Fig. 2 (a), the data acquisition thread continuously polls quantized sEMG signals from skin surfaces by using a eight-channel Myo Armband [12] sampled at a rate of 200Hz. These data streams are then simultaneously transmitted to the Sony Spresense for the data pre-processing thread. As shown in the Fig. 2 (b), the data pre-processing thread rescales and aligns raw sEMG data into appropriate sEMG images (2D-arrays) with an optimal *window size*. First, the raw sEMG signals are subtracted by the data mean and divided by the standard deviation to accelerate CNN model convergence during offline training. Second, the standardized sEMG signals are transformed into an sEMG 2D-array, with each column representing a motor unit action potential at a certain time frame under an eight-channel electrode grid. As a result, the sEMG 2D-array contains both the spatial and temporal features which can be utilized to represent human motor intents. Third, overlapping sliding window augmentation with an optimal *step size* is applied to the sEMG 2D-array, which can divide the sEMG 2D-array into individual sEMG images. Finally, the individual sEMG images can be processed by the CNN model for offline training and real-time testing.

C. sEMG Feature Extraction with Transfer Learning

Convolutional Neural Network Architecture for sEMG Feature Extraction: As mentioned above, the raw sEMG signal is converted to sEMG image that can be directly processed by CNN model. In our proposed system, a 2D-CNN is adopted for sEMG image pattern recognition by extracting spatial features from sEMG images.

As shown in the Fig. 1 (a), our 2D-CNN consists of two feature extraction layers (convolutional layers) with 3x3 convolutional kernels. The first convolutional layer consists of 48-filters, followed by a PReLU activation function, and batch normalization to accelerate model convergence [13]. To prevent overfitting, spatial 2-dimensional dropout (drop rate of 0.5) and max pooling (kernel of size 1x2) were adopted. The second convolutional layer is the same as the first layer except that it uses 96-filters. Lastly, the CNN architecture ends with a fully connected layer for multi-gesture classification.

Transfer Learning for User-specific sEMG data Adaptation: Utilizing the above CNN architecture, the CNN model can be trained for feature extraction from the sEMG image.

However, collecting training data from a single user to train a CNN model can be extremely time-consuming and impractical. In contrast, pre-training a classifier on the accumulated data of preceding subjects could potentially reduce the required amount of data for new users, while enhancing the system's accuracy [4]. As shown in Fig. 1 (a), our CNN is trained offline using a large quantity of labeled data across a source domain to generate a Source Model. Then, the system transfers the learned features from the convolutional layers of Source Model to the Target Model by fine-tuning on the user's newly collected data on the edge device.

To apply transfer learning over the source model and align with the user's specific sEMG signal pattern, the source dataset should have the same placement of EMG sensor as the target dateset. In this paper, we adopt the NinaPro DB5 [14], which also used the Myo Armband as the data acquisition sensor for the source model. During transfer learning, the offline trained model's last fully connected layer will be replaced by an N neuron fully-connected layer depending on N gestures to classify during real-time. Therefore, we aim first to train a CNN source model offline with the NinaPro DB5 by using its prerecorded sEMG from seven gestures. Then, we generalize the features learned from these seven gestures to a four user-specific gestures by applying transfer learning over the CNN Source Model with a small set of data collected in real-time.

D. System Optimization for Real-time Performance

By using transfer learning, we can significantly reduce the CNN training time, but the CNN is also computationally expensive regarding on-device inference, which poses a challenge for embedded system implementation. In this section, we will further improve the real-time performance by quantizing the CNN model and jointly optimizing sEMG data transmission and CNN processing.

CNN Quantization by TensorFlow Lite: CNN models are often assocatied with heavy computations due to the convolutional operations. Typically, the CNN model's parameters (*e.g.,* convolutional kernel) are often represented using floating point values for higher accuracy at the expense of inference latency, and memory occupation. One particular technique to accelerate CNN computation is quantization, which replaces floating point values with integers inside the neural network [10].

By converting the weights and inputs into integer types, the memory occupation and computation load can be significantly reduced and the calculations can be accelerated. Hence, affine mapping can be employed during quantization to accelerate on-device inference and achieve efficient precision for the real-time CNN model by mapping float value parameters to 8-bit integers [15]. For our proposed system, the CNN model was quantized to 8-bit integers using TensorFlow Lite [16] before deploying onto Sony Spresense, with the size of model weights decreased almost tenfold from 772 kilobytes to 73 kilobytes.

System Optimization for Data Transmission and CNN Processing: While quantization addresses on-device inference latency and memory occupation, data acquisition, and preprocessing latency can also be further optimized. To achieve parallelism of data transmission and on-device computation, the data transmission time to retrieve and process one sEMG image should ideally be equal to the latency for one CNN model inference. In our proposed framework, data preprocessing and model inference are carried out in parallel on the Sony Spresense microcontroller. An integral part of such optimization is finding an optimal *window size* and *step size* for the sEMG images.

TABLE I
OFFLINE ANALYSIS ON NINAPRO DB5 WITH 7 GESTURES PERFORMANCE

	Window Size								
Step Size	Acc.	24	28	32	36	40	44	48	52
	10	85.79%	88.5%	**91.12%**	92.14%	93.21%	94.3%	95.22%	96.12%
	14	85.54%	87.5%	90.2%	91.69%	93.08%	93.61%	94.58%	95.44%
	16	84.85%	87.46%	89.89%	91.16%	92.64%	93.69%	94.53%	95%
	18	82.66%	86.97%	88.81%	91.11%	92.51%	93.46%	93.91%	95.25%
Comput. Load (MFlops)		1.59	1.93	2.28	2.63	2.98	3.33	3.68	4.03

The length of the neural action potential distribution (row) inside each sEMG image is determined by the window size, while incorporating more temporal sEMG features per sEMG image typically increases model accuracy. This introduces higher transmission delay when collecting additional sEMG samples, computation latency when processing these features, and inference latency during the CNN model's convolutional operations due to the larger input shape during real-time.

Optimizing the step size for overlapping may address the transmission delays from a larger window size. Overlapping decreases the continual data transmission latency by reducing the subsequent sEMG image retrieval delay to be much shorter after an initially acquired sEMG image. Doing so requires an optimal step size, because a step size that's too small can potentially impair the model's responsiveness during real-time (*i.e.*, the number of predictions it takes for the CNN model to transition to a newly performed gesture). Therefore, it's imperative to find an efficient combination of sEMG window size and step size for the proposed bionic arm control system.

III. EXPERIMENT

A. Experiment Setup

System Implementation: *EffiE* is implemented with Tensorflow [17] and the offline training is performed on a desktop with an Intel i9 CPU and Nvidia RTX 3080 GPU. During offline training, we used the Adam optimizer with a learning rate of 0.2 and a step decay of drop factor 0.9 per 1.5 epochs. Furthermore, to achieve the global optimum loss for the CNN Source Model, the training iterations last 200 epochs with a batch gradient descent of size 384. We used an early stopping if the validation loss did not decrease after 60 epochs to avoid over-fitting issues that may arise during the training process.

sEMG Dataset for Offline Training: The NinaPro DB5 dataset [14], which was used in this paper for offline training. It contains sparse eight-channel sEMG recordings from ten human subjects. We obtained prerecorded sEMG signals of seven gestures from exercise B of this dataset, which is primarily

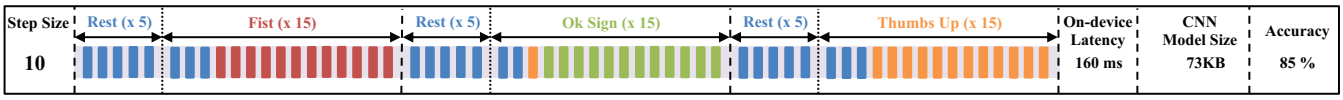

Fig. 3. On-device Performance with Transfer Learning. Each colored bar indicates the classification results for the three distinct gestures.

comprised of isometric, isotonic hand configurations, and fundamental wrist motions. Hence, the CNN Source Model first learns from these accumulated sEMG characteristics before adapting to a new subject's sEMG activity in real time.

Training Data Collection: The user participates in four distinct sessions each performing one of the four gestures: Rest, Fist, Thumbs Up, or Ok Sign. Each session will last approximately two seconds to evaluate the effects of varying sample sizes during Transfer-Learning. As a result, the acquired data consists of 15 windows of 8 sEMG channels, each containing 32 unique sEMG samples from the Myo Armband. After obtaining the real-time dataset, sliding window augmentation and standardization are applied to integrate the temporal features and accelerate model convergence.

B. Offline Classification Accuracy on NinaPro DB5

While CNN has shown promising results for extracting sEMG spatial features via the convolutional layers, this strength is highly associated with its input shape defined by the window size of an sEMG image during feed-forward propagation. However, an sEMG image with a larger window size would also introduce higher data transmission and processing latency. Therefore, this experiment aims to identify an optimal window size and step size for the Target Model, which is essential for efficient real-time gesture recognition. To achieve this, several offline-trained Source Models are examined for their validation accuracy and computational complexity across varying parameters of window and step sizes.

Comparisons of the validation accuracy between various Source Models before transfer learning is shown in the Table. I. Each column indicates the offline validation accuracy tested under different window sizes, while each row represents a step size beneath a Source Model for a fixed window size. In addition, the last row indicate the CNN computation load under a specific window size. As shown in the Table. I, the model can achieve higher accuracy when the input window size increase. However, the CNN computation load also increased because each convolutional layer needs to process more features from the sEMG input. Considering the limited computational resources on the Spresense edge board, a window size of 32 is an optimal trade-off between the CNN accuracy and computational load.

C. On-Device Performance with Transfer Learning

This experiment aims to assess the performance of the real-time target model on edge devices, which is critical for determining its usability for NMI applications such as a prosthesis arm control. This entails evaluating a target model with a fixed window size of 32 and step size of 10 (derived from the previous experiment) for on-device inference accuracy, latency, and memory occupation.

To achieve effective feature learning on user-specific sEMG data, transfer learning was applied over the Source Model using tsEMG data acquired in real-time to generate a Target Model that adapts to the user's specific sEMG signal patterns. The process of fine-tuning the target model takes ten epochs, with each feed-forward propagation accompanied by two batches of sEMG data. Following the fine-tuning of a target model from transfer learning, it is quantized and optimized to a model size of 73 KB using Tensorflow Lite before running inference entirely and efficiently on the edge device.

The comparisons of the on-device target model's real-time responsiveness and accuracy are shown in Fig. 3. The columns indicate the classification results for the three distinct gestures and are represented by different-colored bars. For example, the blue bars beneath the Rest (x5) column indicates the target model's successfully classification of the Rest gesture. Each of these bars takes 160 ms to compute, which includes the parallel inference and data acquisition latency. Twenty of these bars were generated during a single round where the participant was asked to perform the Rest gestures over the five initial predictions. When the participant visually perceives the 5^{th} prediction result on the monitor, the participant immediately transitions to another particular gesture for the subsequent fifteen predictions. Hence, three such rounds constitute one cycle, which was performed to comprehensively evaluate the Target Model's responsiveness and accuracy across the following four gestures during real-time: Rest (blue), Fist (red), Ok Sign (green), and Thumbs Up (orange). Furthermore, as seen in the last column of Fig. 3, the real-time Target Model achieved a classification accuracy of 85%. These experimental results indicate that our proposed system can meet systematic real-time constraints for NMI applications and is therefore practical for prosthesis arm control systems.

IV. Conclusion

This paper investigates the feasibility of deploying deep learning for neuroprosthetic applications using tiny-edge computing devices. We proposed and implemented *EffiE*, an end-to-end CNN based sEMG pattern precognition system for prosthesis arm control on the microcontroller. Our system achieved real-time performance by integrating several strategies including transfer learning, quantization and systematic optimization on the proposed CNN model parameters for on-device deployment. Our experimental results show that our proposed NMI system can meet real-time requirements while identifying the user's motor intents with high accuracy. In our future work, we will investigate on-device Transfer-Learning to improve user experience by enabling model adaptation to user-specific sEMG signals. We see this effort as a stepping stone to future artificial intelligence deployments for low-cost portable biomedical equipment.

REFERENCES

[1] D. Farina, N. Jiang, H. Rehbaum, A. Holobar, B. Graimann, H. Dietl, and O. C. Aszmann, "The extraction of neural information from the surface emg for the control of upper-limb prostheses: emerging avenues and challenges," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 797–809, 2014.

[2] W. Geng, Y. Du, W. Jin, W. Wei, Y. Hu, and J. Li, "Gesture recognition by instantaneous surface emg images," *Scientific reports*, vol. 6, no. 1, pp. 1–8, 2016.

[3] F. Quivira, T. Koike-Akino, Y. Wang, and D. Erdogmus, "Translating semg signals to continuous hand poses using recurrent neural networks," in *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. IEEE, 2018, pp. 166–169.

[4] U. Côté-Allard, C. L. Fall, A. Campeau-Lecours, C. Gosselin, F. Laviolette, and B. Gosselin, "Transfer learning for semg hand gestures recognition using convolutional neural networks," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 1663–1668.

[5] S. Shen, K. Gu, X. Chen, C. Lv, and R. chuan Wang, "Gesture recognition through semg with wearable device based on deep learning," *Mob. Networks Appl.*, vol. 25, pp. 2447–2458, 2020.

[6] A. T. Nguyen, M. W. Drealan, D. K. Luu, M. Jiang, J. Xu, J. Cheng, Q. Zhao, E. W. Keefer, and Z. Yang, "A portable, self-contained neuroprosthetic hand with deep learning-based finger control," *CoRR*, vol. abs/2103.13452, 2021. [Online]. Available: https://arxiv.org/abs/2103.13452

[7] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[8] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[9] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[10] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4820–4828.

[11] "Overview - Spresense - Sony Developer World," https://developer.sony.com/develop/spresense/, [Online; accessed 2022-11-18].

[12] S. Rawat, S. Vats, and P. Kumar, "Evaluating and exploring the myo armband," in *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*. IEEE, 2016, pp. 115–120.

[13] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[14] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.-G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller, "Electromyography data for non-invasive naturally-controlled robotic hand prostheses," *Scientific data*, vol. 1, no. 1, pp. 1–13, 2014.

[15] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

[16] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang *et al.*, "Tensorflow lite micro: Embedded machine learning for tinyml systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, 2021.

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.