Bridging the Generalization Gap in sEMG Keystroke Recognition with LSTM-Based Architectures

Adit Jain
UCLA
aditjain@g.ucla.edu

Charley Sanchez
UCLA
sanchez98@g.ucla.edu

Abstract

Surface electromyography (sEMG) presents significant challenges for neural network-based decoding due to its susceptibility to noise [8] and difficulties in generalization to unseen users. The EMG2QWERTY dataset, collected by Meta [14], contains over 346 hours of sEMG data from 108 users; however, models trained on this dataset exhibit poor generalization, with a Character Error Rate (CER) exceeding 55%. This work proposes an improved neural network architecture leveraging a bidirectional LSTM [12] with residual connections [9] layer. Our approach demonstrates a 12% reduction in CER on the EMG2QWERTY dataset compared to the baseline, while also adapting to lower-channel, lower-sampling-rate datasets with minimal fine-tuning. These improvements pave the way for more robust, out-of-the-box sEMG-based keystroke decoding, reducing the need for extensive user-specific retraining.

1 Introduction

Surface Electromyography (sEMG) struggles to be used in Neural Networks due to its high susceptibility to noise [8] and difficulty in procuring large datasets [17]. Additionally, a challenge for sEMG decoding is maintaining its robustness when generalizing to real-world use cases. The emg2qwerty project by Meta collected over 346 hours of sEMG data from a 16-electrode wristband on each hand from 108 users. However, despite the large quantity of data, the model generalized poorly to the test dataset, reporting a Character Error Rate (CER) of over 55%. For widespread adoption of sEMG based innovation such as keystroke decoding, it is critical for the underlying deep learning models to generalize well to unseen users. The goal of this paper is to present alternative neural network architectures which increase generalization performance on unseen users, while also even generalizing well to datasets collected from different neural wristbands that contain 8 sEMG channels at a lower sampling rate (200Hz). Improving generalization with minimal fine-tuning would open the door for more widespread, "out-of-the-box" usage of sEMG products for keystroke decoding since the model would not have to extensively re-train on the new user's data.

2 Methods

Both Meta's [14] EMG2QWERTY dataset and Professor J. C. Kao's dataset required preprocessing, custom architecture, and hyperparameter tuning to create a model capable of generalizing well.

2.1 EMG2QWERTY

Dataset

The EMG2QWERTY dataset we used for pretraining consists of 18 subjects in the training set, 6 in the validation set, and 6 in the test set. Each set contained separate users to ensure more reliable evaluation metrics. This allocation was chosen as a balance between computational and storage constraints while maximizing data inclusion to improve generalization.

For strict comparison to Meta's baseline, we trained a single model on the full "generic" dataset. The full dataset and its details can be found in Meta's paper [14].

Preprocessing

In addition to the transformations included in Meta's model, we applied random scaling between 0.8 and 1.2, as well as Gaussian noise with a standard deviation between 0.01 and 0.03, to the raw sEMG signals. Initially, we hypothesized that adding Gaussian noise could improve generalization and reduce overfitting, a common technique in deep learning [4]. However, further research revealed that sEMG signals inherently contain significant noise, suggesting that reducing noise rather than introducing additional variability might be more beneficial [3]. This remains an open question for future research: investigating whether Gaussian noise could enhance model robustness for sEMG data.

Model architecture

The EMG2QWERTY architecture closely follows that of the original Meta [14] paper. The key difference is that where Meta uses a traditional 2D convolutional layer, we implemented a residual LSTM. Since the goal is to perform transfer learning from EMG2QWERTY to the Kao dataset, the model's parameters must work well on both. To achieve this, we selected the parameters to minimize complexity while still achieving high performance. The need to minimize complexity arises from the risk of overfitting on the smaller Kao dataset.

We implemented a two-layer bidirectional LSTM with a hidden layer size of 96. The bidirectional component is essential for increasing performance on datasets containing continuous movements and temporal dependencies [12]. A dropout rate of 0.3 was applied between hidden layers to reduce the risk of overfitting and improve generalization [7].

Next, we incorporated layer normalization after the LSTM layers to stabilize training by reducing internal covariate shift, which is crucial for improving the convergence speed, especially when training on noisy data like sEMG [1]. A fully connected block follows the LSTM layers to allow the network to learn high-level abstractions from the sequence output. More dropout is applied here to further combat overfitting, as this layer serves as a bottleneck for the model's learned representations.

Finally, a linear layer is included to reshape the output for our prediction task, ensuring it matches the given input shape. The residual connection to the input data is incorporated to mitigate the potential for gradient vanishing or explosion, which is especially important when training deep networks with sequential data like sEMG [9]. The full LSTM framework is shown in figure 1.

Training setup

The full training structure used can be viewed in the original Meta paper [14]. Please note that instead of the 4 TDS convolutional blocks, the LSTM above was implemented.

Domain Adversarial Neural Network

We investigated the use of Domain Adversarial Neural Networks (DANN) to increase the generalization performance of our model to new users. Liu et al. implemented a DANN in their project "WR Hand" which used sEMG data to estimate hand pose [10]. The motivation behind the implementation was to encourage the model to learn non user-specific features in the neural network, which in turn should increase generalization performance to unseen users. In the emg2qwerty dataset, we suspected that the baseline model may have also learned user-specific features from the EMG spectrograms. In this case, the model would try to classify keystrokes in part by first determining which user, or class of users, the data belongs to. This is impractical for applications in which there may be hundreds and thousands of test users. The original dataset split used the same users for both validation and training, and since we observed low validation error rates but high test error rates, we had reason to believe that the model was learning user-specific features.

The feature extractor passes its activations to both the label predictor (keystroke classification) as well as a domain classifier (user classification). The domain classifier and label classifier backpropagate their loss to the feature extractor, but the domain classifier loss is multiplied by a factor -lambda, effectively creating a "gradient reversal layer". In this way, the weight updates in the feature extractor

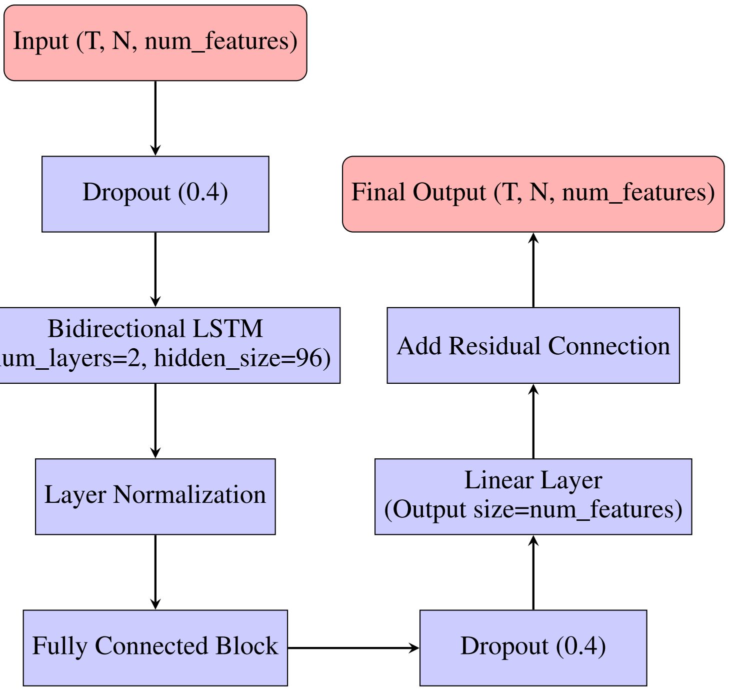
Input (T, N, num_features)
Dropout (0.4)
Bidirectional LSTM (num_layers=2, hidden_size=96)
Layer Normalization
Fully Connected Block
Dropout (0.4)
Linear Layer (Output size=num_features)
Add Residual Connection
Final Output (T, N, num_features)

Figure 1: LSTM architecture used.

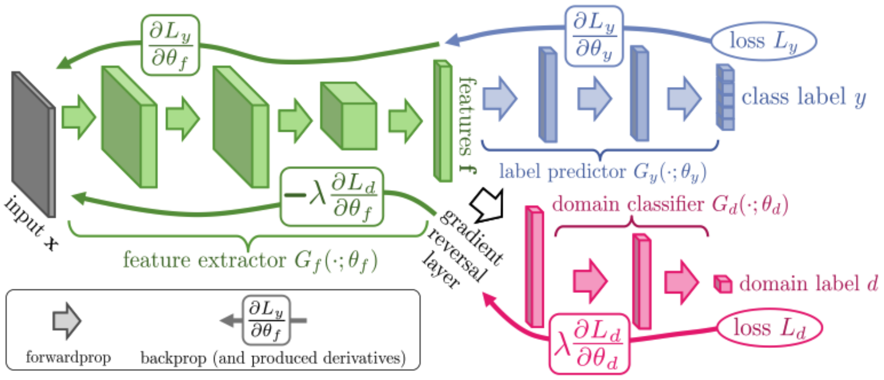
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
domain classifier $G_d(\cdot;\theta_d)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 2: Conceptual example of a DANN architecture. [5]

are updated such that the performance of the domain classifier worsens, while the label predictor continues to improve.

Empirically, our implementation of the DANN had difficulty converging to a low label classification loss, likely due to the competing gradients of the domain classifier and the label classifier that were backpropagated through the feature extractor layers.

2.2 Kao Dataset

Dataset

The Kao dataset consists of two separate subjects. The training set comprises the first 90% of data from the first subject, while the validation set consists of the remaining 10%. The test set includes all data from the second subject, ensuring that training and testing occur on different users to prevent overfitting. Additionally, the first subject has twice as many data points as the second, making it the better of the two for training purposes.

The dataset includes 14 features: 8 corresponding to the separate EMG electrodes from the armband and 6 representing the activation of each finger and the rest state. The features are recorded at 200Hz with an integer representing the signal from each EMG electrode and a 0/1 binary indicating the activation state of each finger (or rest state).

Preprocessing

Since the Kao dataset contains a different structure, there are several preprocessing steps to enable transfer learning from EMG2QWERTY to occur effectively. First, the data must be converted to an HDF5 file with an identical formatting to Meta's dataset. This was done by translating finger activations into key presses, as this was how the data was collected.

Table 1: Finger to character representations used in preprocessing of Kao dataset.

Finger	Character
Thumb	'Space'
Index	'j'
Middle	'k'
Ring	'l'
Pinky	';'
Rest State	'r'

The EMG channels from the Kao dataset were duplicated to align with the 16-electrode configuration used in the original EMG2QWERTY dataset. Additionally, the window length and padding were reduced by a factor of 10 to account for differences in sampling frequency. Another key modification was refining the model's character set. Instead of including all keyboard characters as before, the model was restricted to only the six characters listed in the table above. This adjustment ensures that the model outputs only the characters relevant to this dataset.

Model architecture

The model for the Kao dataset contains the same architecture as the model for the EMG2QWERTY dataset.

Training setup

Following the data splits defined in Section 2.2 **Dataset**, we train a baseline model on a single session from one user, with validation performed on the same session since each user has only one. This approach ensures that test users remain independent of the hyperparameter tuning process, preventing any influence on test performance while optimizing for validation accuracy.

Using the same data splits, we also train a model leveraging transfer learning. Specifically, we initialize the model with the weights learned from the EMG2QWERTY model and fine-tune it over 300 epochs using a gradual unfreezing strategy. Gradual unfreezing is a technique that enhances transfer learning performance [6] by allowing the model to adapt to the new dataset incrementally while preserving the stability of learned representations. This strategy mitigates catastrophic forgetting, a phenomenon where a model, when fine-tuned too aggressively on a new dataset, loses critical information acquired during pretraining. By progressively introducing trainable parameters, the model can retain important structural knowledge while refining its feature representations to better align with the new dataset.

Initially, all layers except the output layer remain frozen, ensuring that the model retains the high-level features learned from the pretrained network while adapting only the final classification layer to the new task. This prevents abrupt shifts in feature distributions and ensures that early updates do not distort well-established representations. After 50 epochs, we unfreeze the LSTM layer, which is responsible for modeling temporal dependencies in the data. This allows it to refine sequential patterns specific to the new dataset without destabilizing lower-level feature extraction. The process continues with the MultiLayer Perceptron (MLP) layers at 100 epochs, enabling the network to progressively fine-tune its intermediate representations. Finally, at 150 epochs, we unfreeze the batch normalization layers at the beginning of the network, allowing them to recalibrate their statistics based on the new dataset's distribution.

To further optimize performance, we apply layer-specific learning rates, ensuring that newly unfrozen layers adapt at a controlled pace. This approach prevents abrupt parameter updates that could lead to instability and overfitting. By carefully tuning the learning rate for each layer as it is unfrozen, we enable a smooth transition from the pretrained model to a fully fine-tuned model suited for the target dataset. This structured adaptation ensures that the model effectively leverages prior knowledge while achieving optimal performance on the new task.

All models utilizing the Kao dataset were implemented on a single NVIDIA RTX 3080 GPU with a consistent batch size of 32. We selected AdamW as our optimizer due to its generalization capabilities in the presence of overfitting tendencies [11]. To streamline our training process, we implemented a one-cycle learning rate scheduler, which effectively reduced the need for extensive hyperparameter optimization [15]. Both the baseline and pretrained models shared identical hyperparameters, with the notable exception of gradient clipping thresholds. The baseline model achieved optimal performance with a gradient clipping value of 0.7, whereas the pretrained model demonstrated superior results with a higher threshold of 1.0. We incorporated gradient clipping specifically to mitigate the gradient explosion issue commonly encountered in LSTM-based architectures [13].

3 Results

3.1 LSTM Performance on EMG2QWERTY

Table 3.1 shows the results of the LSTM model trained on all 108 subjects, using the same data split as in [14]. The results can be directly compared to the original EMG2QWERTY study [14] and its reported baseline performance. Our new LSTM architecture reduces CER from 55.38 to 48.41, a 12% improvement in CER.

Table 2: Validation and Test Metrics for LSTM model trained on entire EMG2QWERTY Dataset

Metric	**Validation**	**Test**
Loss	1.4339	1.6569
CER (%)	43.532	48.413
IER	15.4153	13.5959
DER	1.7777	1.2866
SER	26.3394	33.5307

3.2 Evaluation of LSTM on Kao Dataset

The immediate improvement in performance on unseen users with the LSTM architecture narrowed our investigation to LSTM only. Specifically, we now investigated the performance of the LSTM on the Kao dataset, and whether pre-training an LSTM on the larger EMG2QWERTY dataset leads to better generalization performance on the Kao dataset.

Table 3 shows the performance of an LSTM model when trained solely on the Kao dataset. Of the two users in the Kao dataset, User 1 was used for training and validation, and User 2 was used for testing. Table 4 shows the results for when an LSTM model was trained on the EMG2QWERTY dataset, and then transferred to the Kao dataset. The pre-trained model was fine-tuned on User 1, and evaluated on User 2. The two experiments show that pre-training on the EMG2QWERTY dataset and fine-tuning with User 1's data improved generalization to the test user compared to when the model was trained only on User 1.

Table 3: Validation and Test Metrics for LSTM trained and evaluated on Kao Dataset

Metric	Validation	Test
Loss	0.1320	1.083
CER (%)	2.089	22.913
IER	0.5222	1.3093
DER	0.3916	0.2455
SER	1.1749	21.3584

Table 4: Validation and Test Metrics for LSTM pre-trained on EMG2QWERTY and fine-tuned on Kao dataset.

Metric	Validation	Test
Loss	0.1548	0.6763
CER (%)	4.178	21.522
IER	1.3055	2.2368
DER	0.5222	0.7365
SER	2.3499	18.5488

4 Discussion

4.1 EMG2QWERTY

The LSTM architecture significantly outperformed Meta's baseline CNN on EMG2QWERTY, likely due to its ability to capture temporal dependencies in the EMG signals [16]. This improved performance suggests that the model has learned useful and generalizable features. Given this, we expect transfer learning to be even more effective when applied to our custom dataset, which is much smaller. By leveraging the strong feature representations learned from EMG2QWERTY, we aim to enhance generalization performance on our dataset as well.

4.2 Kao dataset

Table 3 presents the optimal performance achieved through hyperparameter tuning on our architecture when training the Kao dataset. Due to data limitations, the validation set consisted of entries from the same user the model was trained on, which contributed to a significant discrepancy between validation and test character error rates (CER). The extremely low validation CER indicates that the model overfit to the specific user, learning user-specific features rather than generalizable patterns. As a result, the test CER was nearly 11 times higher than the validation CER, highlighting severe overfitting. This overfitting hindered generalization, leading to strong performance on validation data that closely resembled the training set but relatively poor performance on unseen test data.

Our implementation of transfer learning yielded significant improvements in model performance. By first training EMG2QWERTY on our custom LSTM architecture and subsequently fine-tuning it on the Kao dataset, we achieved a test Character Error Rate (CER) of 21.522%. This represents a 6% relative (1.4% absolute) performance improvement compared to models trained exclusively on the Kao dataset. While the CER improvement is noteworthy, the more significant finding is the substantial reduction in the disparity between validation and test performance. The test CER is now only 5.15 times higher than the validation CER, less than half the disparity observed in our previous model. This marked reduction in the validation-test gap indicates substantially decreased overfitting. We conclude that the transfer learning approach has enabled our model to extract more generalizable features from the EMG data, reducing its dependence on user-specific patterns that were prevalent in our earlier implementations.

This is an important finding as it provides empirical evidence that transfer learning can effectively leverage large datasets used to train similar models when fine-tuning new models on smaller, limited datasets. Our results demonstrate that knowledge transfer between related EMG processing tasks is not only feasible but advantageous, particularly in domains where data collection is costly or constrained.

4.3 Limitations

Computation Constraints

All training was performed locally on a single RTX 3080 GPU. This approach was adopted to mitigate the costs associated with hosting large volumes of data on cloud platforms required for training the EMG2QWERTY model. As a consequence, the training time for the EMG2QWERTY model extended to approximately 13 hours for a mere 50 epochs. This significant time investment rendered comprehensive hyperparameter optimization infeasible within our project timeline.

Architectural Restrictions

The computational limitations directly impacted our architectural decisions. Despite evidence suggesting that additional LSTM layers could enhance performance, the increased complexity would have resulted in prohibitively extended training times with our large dataset. Consequently, we were constrained to implement a relatively modest LSTM architecture. We hypothesize that the Character Error Rate (CER) could be further reduced with more sophisticated network configurations and thorough hyperparameter tuning.

Dataset Limitations

The Kao dataset only contained 2 subjects. This meant we had the choice of splitting user 1 into train/val or user 2 into val/test splits. We chose the former to keep test data completely separate and avoid data leakage. However, this meant that we were more prone to overfitting as the validation data still contains user-specific features that are seen in the training data as well. Minimizing the validation error would not necessarily lead to the most generalizable results in this case. We believe that including a third validation subject would be able to improve generalization performance.

4.4 Future Directions

The chosen architecture has room for improvement. A deeper LSTM could be helpful in capturing more generalizable patterns found in the EMG2QWERTY dataset. Additionally, there is evidence that a hybrid CNN-LSTM network could lead to even higher performance than an LSTM alone on sEMG data [2]. We are hopeful that with more computational resources or more time to develop a model, that one of these design choices could result in an even lower CER for both the EMG2QWERTY and Kao datasets.

Another potential architecture to run experiments on is a transformer based network. Previous studies have shown transformer networks to perform effectively on sEMG data for gesture recognition [18], which indicates it could be helpful in maximizing performance on our tasks as well.

Future work includes experimenting with several different architectures to further boost the effect of transfer learning on our model. Additionally, fine-tuning the model on less data, 30 to 60 seconds specifically, could allow for more real-time applications. This could drastically reduce the amount of onboarding time for new users of a potential product and allow for future use of different tasks where data collection is limited.

A Appendix / supplemental material

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[2] Dianchun Bai, Tie Liu, Xinghua Han, and Hongyu Yi. Application research on optimization algorithm of semg gesture recognition based on light cnn+lstm model. *Cyborg and Bionic Systems*, 2021, 2021.

[3] R. H. Chowdhury, M. B. Reaz, M. A. Ali, A. A. Bakar, K. Chellappan, and T. G. Chang. Surface electromyography signal processing and classification techniques. *Sensors (Basel, Switzerland)*, 13(9):12431–12466, 2013.

[4] Aadil Gani Ganie and Samad Dadvadipour. Exploring noise-induced techniques to strengthen deep learning model resilience. *Pollack Periodica*, 19(3):8 – 13, 2024.

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016.

[6] Federica Gerace, Diego Doimo, Stefano Sarao Mannelli, Luca Saglietti, and Alessandro Laio. Optimal transfer protocol by incremental layer defrosting, 2023.

[7] Lu Hou, Jinhua Zhu, James Kwok, Fei Gao, Tao Qin, and Tie-Yan Liu. Normalization helps training of quantized lstm. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[8] Ahmet Doğukan Keleş, Ramazan Tarık Türksoy, and Can A. Yucesoy. The use of nonnormalized surface emg and feature inputs for lstm-based powered ankle prosthesis control algorithm development. *Frontiers in Neuroscience*, 17, 2023.

[9] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. Residual lstm: Design of a deep recurrent architecture for distant speech recognition, 2017.

[10] Yang Liu, Chengdong Lin, and Zhenjiang Li. Wr-hand: Wearable armband can track user's hand. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):1–27, 2021.

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[12] Chenfei Ma, Chuang Lin, Oluwarotimi Williams Samuel, Weiyu Guo, Hang Zhang, Steve Greenwald, Lisheng Xu, and Guanglin Li. A bi-directional lstm network for estimating continuous upper limb movement from surface electromyography. *IEEE Robotics and Automation Letters*, 6(4):7217–7224, 2021.

[13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2013.

[14] Viswanath Sivakumar, Jeffrey Seely, Alan Du, Sean R Bittner, Adam Berenzweig, Anuoluwapo Bolarinwa, Alexandre Gramfort, and Michael I Mandel. emg2qwerty: A large dataset with baselines for touch typing using surface electromyography, 2024.

[15] Leslie N. Smith. Cyclical learning rates for training neural networks, 2017.

[16] Ravi Suppiah, Noori Kim, Anurag Sharma, and Khalid Abidi. Fuzzy inference system (fis) - long short-term memory (lstm) network for electromyography (emg) signal analysis. *Biomedical Physics Engineering Express*, 8(6):065032, nov 2022.

[17] Chuheng Wu, S. Farokh Atashzar, Mohammad M. Ghassemi, and Tuka Alhanai. An lstm feature imitation network for hand movement recognition from semg signals, 2024.

[18] Wenli Zhang, Tingsong Zhao, Jianyi Zhang, and Yufei Wang. Lst-emg-net: Long short-term transformer feature fusion network for semg gesture recognition. *Frontiers in Neurorobotics*, 17, 2023.