ECEC147/247 Project Write Up

Abhimanyu Borthakur
MS Electrical and Computer Engineering
abhimanyu911@ucla.edu

Iryna Iziumska
Mathematics of Computation Undergraduate Program
iziumska@ucla.edu

Caitlin Tang
Computational and Systems Biology Undergraduate Program
ctang04@g.ucla.edu

Julie Tran
Computational and Systems Biology Undergraduate Program
juliemt@g.ucla.edu

Abstract

This paper explores an existing dataset of surface electromyography (sEMG) signals collected from a single individual while typing letters in order to determine the most effective deep learning model for typed character classification. Various architectures were tested, including convolutional neural networks (CNNs), CNNs with data augmentation techniques, and hybrid CNN-recurrent models. Among these, a CNN-BiLSTM architecture with squeeze-and-excitation blocks achieved the highest classification accuracy, effectively capturing both spatial and temporal dependencies in the sEMG signals. Link to code

1 Introduction

sEMG electrodes can noninvasively capture electrical signals associated with muscle contraction and relaxation [1]. Wearable sEMG sensors have the potential to provide highly personalized monitoring of muscle health and performance to assess physical exertion, injury rehabilitation progress, and symptom detection or progression of neuromuscular disorders such as Parkinson's Disease [1, 2].

sEMG data varies between individuals due to inherent differences ranging from precise muscle composition in the limbs to sEMG electrode placement. Thus, developing a prediction model that can accurately interpret sEMG signals for a single individual using that individual's recorded data is crucial to furthering precision medicine and health technologies. In this study, we explored an sEMG dataset representing signal recorded during an individual's keystrokes to create a prediction model that would be useful for creating better assistive devices for individuals with disabilities or limitations and could be extrapolated to recording muscle contractions for other minor, everyday tasks. By leveraging temporal and spatial representations of sEMG data, it is possible to exploit the advantages of different deep learning architectures to extract the nuanced, individual-specific features needed for making accurate, medically relevant, and personalized predictions.

2 Methods

2.1 Preprocessing

In this section, all models were evaluated after training with 30 epochs due to compute limitations. Although greater epochs for model training would have provided more interpretable results, the observed improvement in character error rate (CER) from the baseline model for 30 epochs provided confidence that this limited testing could still reflect meaningful learning and lead to relevant findings about model performance.

2.1.1 Data Augmentation

Our model is prone to overfitting because of its reliance on limited EMG data from one individual. Even though the model is only making predictions for that same individual, minor variability in factors like electrode placement between sample measurements can still negatively impact performance of an overfitted model. Data augmentation can improve generalizability and robustness by introducing noise or adjustments to the input that are analogous to real-world variability [3, 4, 5]. The model consequently learns parameters that can more reliably extract relevant features from heterogeneous input data.

Spectrogram Augmentation via SpecAugment The baseline model implements a variety of standard data augmentation techniques that include band rotation and temporal jitter. For the last preprocessing step, the default model performs blackout data on log spectrograms via the SpecAugment transform, which encourages learning on more limited training data to discourage overfitting. However, it is possible that the default hyperparameters are not optimal. We thus evaluated model performance with greater blackout (less overfitting at the risk of underfitting due to a lack of input features from which to predict a nuanced signal) as well as reduced blackout (better learning of the single individual's data at the cost of worse generalizability). This was accomplished through changing the quantity of frequency and time masks applied to the spectrograms. Quantitative results can be found in Table 1.

Magnitude Warping Band rotation and temporal jitter are meant to imitate real-world variability seen with time series sEMG data, such as discrepancies in relative electrode positioning on different individuals' wrists and latency with sEMG signal readouts. Past studies forprediction based on sEMG data from wearable electrodes have implemented these techniques, but also explore a data transformation for magnitude warping that is not included in the baseline model [4, 6]. Magnitude warping applies smoothly varying noise to samples based on a cubic spline fit to a series of scaling factors randomly sampled from a normal distribution [4, 6, 3, 5, 7, 8]. This approach to scaling can be interpreted as stochastically adjusting for extrema in recorded movements, and can be tuned with hyperparameters for "knot" quantity and sigma. The number of knots determines how many scaling factors will be sampled for fitting the cubic spline such that more knots result in a more complex cubic spline for the noise curve and greater variance in the noise curve applied to the original input. Sigma defines the variance of the normal distribution for sampling the scaling factors [4, 6, 7].

Because prior studies observed enhanced model performance with magnitude warping, we adapted implementations of this method that were originally created for their datasets and Keras library frameworks [4, 6, 7]. Due to the similar nature of our input data, we tested similar hyperparameters as well as hyperparameters that would apply less severe noise (less knots and reduced sigma). Results can be observed in Table 1.

2.1.2 Dimensionality Reduction

The first set of learnable parameters is encountered when the "Rotation Invariant" MLP applies a linear transformation and ReLU activation to the input spectrograms before data from both hands is combined for the rest of the architecture. Although this reduces the input features, it is possible that the quantity of features chosen may still be excessive such that the model is extracting nuances that are hyperspecific to the training data [9, 10]. We thus tested further reducing the input features to 576 total rather than 768 across both hands via a 1D convolution as well as testing the effects of performing a subsequent transposed convolution back to the original 768 total features. The latter retains the benefits of extracting minimal important features while also allowing for greater flexibility

in learning and prediction through returning to a larger feature space. Kernel sizes of 16 and 32 were tested, and results can be observed in Table 1.

2.2 CNN Architecture

The implemented Convolutional Neural Network (CNN) architecture is designed to process sEMG signals for keystroke prediction using a residual deep CNN-based encoder. The pipeline begins with basic data preprocessing with spectrogram normalization applied to standardize the input features, followed by a multi-band rotation-invariant multilayer perceptron (MLP) that extracts relevant features. The resulting input is then reshaped for efficient processing by the convolutional layers.

The core of the architecture is a deep residual 1D CNN encoder, which consists of multiple stacked residual convolutional blocks. Each block contains a one-dimensional convolutional layer with a specified kernel size of nine followed by a ReLU activation function. Dropout and layer normalization were added for regularization and to ensure stable training. The residual connections within these blocks help mitigate vanishing gradients and improve gradient flow during backpropagation. In cases where the input and output channel dimensions differ, a projection layer is applied to maintain consistency.

We also incorporated the TDSConv CNN baseline [11] into our experimentation pipeline. In order to improve its performance on the test set, we added dropout, squeeze and excitation blocks (which recalibrate channel-wise features, effectively improving residual networks) [12] and a BiLSTM gradually and benchmarked each augmented model. These additions showed an improvement in performance compared to the baseline CNN (see Table 2).

2.3 Exploring CNN+RNN Architectures

Despite its strengths, the CNN architecture has limitations. A primary drawbacks is its reliance on local receptive fields, which can make it challenging to capture long-term dependencies in sequential data without additional processing, such as incorporating recurrent neural networks (RNN) or attention mechanisms. Additionally, CNNs often require a large number of parameters, which can lead to high computational costs and potential overfitting if not properly regularized. To address this, RNN-based layers were used to refine the extracted features and better represent sequential patterns in the data [13] [14] [15]. Following feature extraction, the CNN generates feature maps that are further processed by a recurrent neural network (RNN) architecture to capture temporal dependencies in the sEMG signals. This hybrid architecture ensures that the local feature extraction from the CNN and long-range temporal dependencies from the RNN layers collectively enhance model performance. We explored the two primary RNN architectures used in processing sEMG signals: Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). Both architectures are effective in capturing long-term dependencies in sequential data and mitigate the vanishing gradient that standard RNNs have. The primary difference between GRUs and LSTMs is that GRUs have only two update gates whereas LSTMs have three [16]. Although both are effective, due to its lower computational cost, the GRU is often preferred over LSTM networks. After feature processing by the RNN layers, the encoded representations are passed through a fully connected linear layer that maps the extracted features to the number of output classes corresponding to possible keystroke predictions. The final output is processed through a LogSoftmax layer to generate probability distributions over the predicted characters. The model is trained using Connectionist Temporal Classification (CTC) loss, which enables sequence prediction without requiring explicit alignment between input signals and target labels. A decoder module further translates the model's predictions into readable text sequences.

2.3.1 CNN + GRU

A GRU is a RNN designed to handle long-term dependencies by controlling the flow of information in making predictions. Since our task involves a sequence of signals, the added GRU layers capture temporal relationships that CNNs struggle with as CNNs primarily focus on local dependencies [13][15]. We implemented a CNN+GRU Hybrid model by incorporating two bidirectional GRU layers that process the feature maps extracted by the CNN [13][17][15].

To process the sEMG data, this CNN has 2 convolutional layers, each with 24 filters, all using a 9×1 kernel size and ReLu activation. Each convolutional layer is followed by a dropout layer to regularize

the model, with the dropout rate set to 0.3. The model does not use max-pooling; instead, it relies on the convolutional layers with increasing receptive fields to capture temporal dependencies across the EMG signal. The output from the convolutional stack is passed through an MLP layer with 480 hidden units to further process the learned features.

The CNN output is then passed to a GRU as input. This model also utilizes a bidirectional GRU which consists of two layers, each with 256 hidden units, resulting in a total of 512 hidden units due to the bidirectional setup. This enhances the model's ability to capture contextual information from both past and future time steps. A 0.3 dropout rate is applied between the GRU layers for regularization. Finally, the output from the GRU is passed through 1 fully connected linear layer of 512 neurons with dropout of 0.3 applied before a LogSoftmax layer for classification.

The model is trained using Adam optimization with a learning rate of 0.001, CTC loss, and a batch size of 32 for 150 epochs or 400 epochs (when computational resources were not as limited).

2.3.2 CNN + BiLSTM

For the second CNN+RNN model implementation, a Bidirectional Long Short-Term Memory (BiLSTM) was used. While the LSTM is a popular choice in learning long-term dependencies, it only extracts the forward features whereas BiLSTM is more comprehensive, processing the extracted features in both forward and backward directions [18] [19]. In the CNN+BiLSTM architecture, the features extracted in the CNN layers were passed through a BiLSTM block. By concatenating the outputs from both directions, it created a richer representation of the sequence at each time step. Similar to the GRU, the BiLSTM block is configured with a specified hidden size and number of layers, and dropout was applied for regularization to prevent overfitting and help the model generalize.

2.4 CNN+RNN Model Hyperparameter Exploration and Model Evaluation

CNN + GRU Hyperparameter Exploration: To optimize the performance of the CNN+GRU model, hyperparameters including the hidden sizes for both the CNN and GRU, the number of layers in the network, and the feature size of the MLP were explored. All other parameters were left as their default values.
CNN + BiLSTM Hyperparameter Exploration: After training the baseline for 400 epochs, a bigger MLP feature size resulted in improvement over said baseline. Gradual addition of elements such as dropout with a rate of 0.3, squeeze and excitation and the BiLSTM unit continued to show improvements eventually resulting in the best model.
Model Evaluation: Throughout training and evaluation, the model tracked loss and error metrics, specifically Character Error Rate (CER), to assess performance. The architecture is implemented using PyTorch Lightning, ensuring an organized and scalable framework for training, validation, and testing. By integrating residual convolutions, dropout regularization, and layer normalization, the CNN-RNN-based models provides a robust and efficient approach to mapping sEMG signal sequences to corresponding typed characters.

2.5 Transformers

Transformers have demonstrated remarkable success in sequence-based tasks, particularly in natural language processing and speech recognition which are similar to the key recognition from muscle contractions[20, 21]. Their ability to capture long-range dependencies efficiently through self-attention mechanisms makes them an attractive choice for our problem, especially due to their success with text and sequential tasks. Unlike recurrent architectures, which process sequences sequentially and suffer from vanishing gradient issues, transformers allow for parallelization, reducing training time while enhancing the model's capacity to capture global context. Given these advantages, we explored transformer-based architectures for our task.

2.5.1 Standalone Transformer Model

Initially, we implemented a standalone transformer encoder model with a classifier at the end to evaluate its performance due to the rich representations that they are able to create, taking inspiration from BERT and using only the encoder [22]. We experimented with various hyperparameters, including the number of transformer layers, embedding dimensions, and attention heads. However,

despite extensive tuning, the model struggled to converge effectively, yielding a Character Error Rate (CER) of nearly 100 even after 30 epochs. The lack of locality bias, which CNNs inherently provide, likely hindered the model's ability to extract low-level patterns from the input data.

2.5.2 CNN + Transformer Hybrid Model

To address the shortcomings of the standalone transformer, we integrated it with a convolutional neural network (CNN) to leverage both local feature extraction from CNNs and long-range dependency modeling from transformers. Existing transformer models have added CNN-style layers like the Swin Transformer made for more visual data, which seemed to be compatible as our data was text but signals [23]. As models that work well with visual data worked well on the dataset, we decided to combine the CNN models of visual and spacial focus with the transformer encoder that is well equipped to interpret and deal with textual data. The CNN encoder processes the input features and extracts spatial hierarchies before passing them to the transformer layers for sequential modeling. This hybrid approach mitigates the transformer's difficulty in capturing local dependencies while still allowing it to model long-range interactions.

3 Results

A summary of our test results is provided in Tables 1 and 2. We observe gradual improvements in our performance when incorporating architectural improvements in our baseline TDSConv CNN. It is clear that the best performing model comprises the baseline with MLP features increased to 480, a dropout of 0.3, a squeeze and excitation block and a bidirectional LSTM (see index 5, Table 2).

Index	Preprocessing Modification	Test CER
1	Baseline (3-run average) [11]	67
2	Data Augmentation with SpecAug (4 time masks, 3 frequency masks, 3-run average)	**63**
3	Data Augmentation with SpecAug (2 time masks and 1 frequency mask)	88
4	Data Augmentation with MagWarp (knot = 4, sigma = 0.2)	89
5	Data Augmentation with MagWarp (knot = 3, sigma = 0.1)	89
6	Data Augmentation with SpecAug + Magwarp	69
7	MLP Dimensionality Reduction (576 features, kernel = 32)	95
8	MLP Dimensionality Reduction (576 features, kernel = 16)	91
9	MLP Dimensionality Reduction (576 features, return to 768 features, kernel = 16)	89

Table 1: Comparison of various approaches to data augmentation and preprocessing. All other aspects of the model architecture were unchanged from the baseline, and all results are from training runs with 30-epochs.

Index	Model Type	Test CER	Loss Plot
1	Baseline [11]	22	Fig 1 a
2	Baseline (MLP = 480)	21	Fig 1 b
3	Baseline (MLP = 480) + Dropout (0.3)	17	Fig 1 c
4	Baseline (MLP = 480) + Dropout (0.3) + BiLSTM	15	Fig 1 d
5	Baseline (MLP = 480) + Dropout (0.3) + BiLSTM + Squeeze-and-Excitation	**13**	Fig 1 e
6	Custom CNN + GRU, Boosted MLP (480), Dropout (0.3)	21	Fig 1 f
7	Custom CNN + GRU, Boosted MLP (480), No Dropout	21	Fig 1 g
8	Custom CNN + GRU, Boosted MLP (600), Dropout (0.3)	22	Fig 1 h
9	Frequency and time mask augmented baseline (30 epochs)	62	Fig 1 i 'aug_val'
10	Transformer Encoder (50 epochs)	98	–
11	Custom CNN + Transformer Encoder (1 layer) (50 epochs)	22	Fig 1 i
12	Custom CNN + Transformer Encoder (4 layers) (50 epochs)	23	–

Table 2: Comparison of various model configurations and their Test CER: Custom CNN uses 2 layers with hidden size 128. GRU uses 2 layers with hidden size 256 and LSTM uses 2 layers with hidden size 128.

4 Discussion

Preprocessing In our limited tests for data augmentation, reducing blackout applied by the SpecAugment function had greater testing error relative to the baseline model, but increasing blackout showed

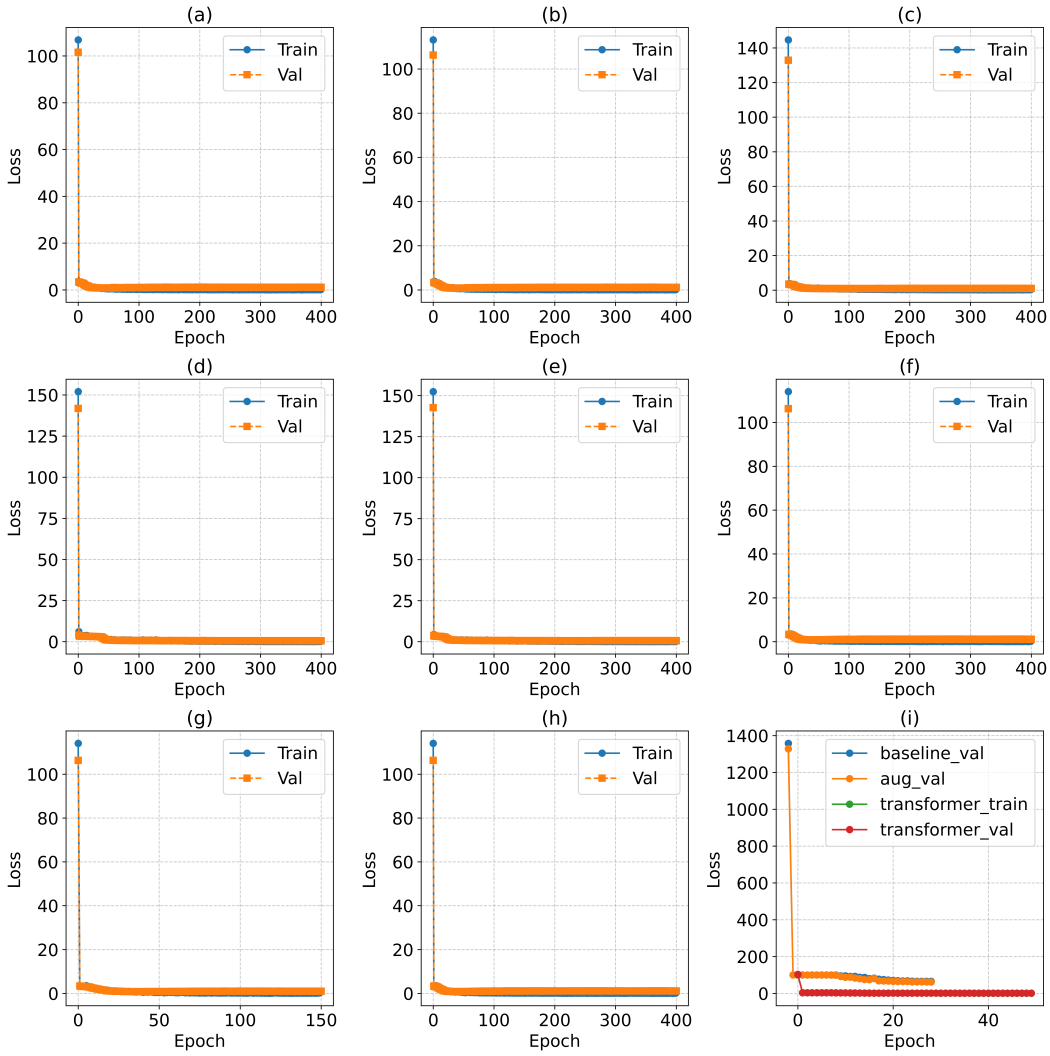

Figure 1: Evolution of loss metrics for all model types. Refer Table 2 for the model type corresponding to each subplot.

a slight decrease that remained consistent three trials for both the baseline model and the baseline model with greater SpecAugment blackout. Even though the magnitude of CER difference is not particularly large and the worse performance of the baseline is largely due to one poorly performing run, the nonzero decrease in the models with higher blackout suggests that there is potential for further testing with greater resources to more definitively verify if increasing blackout causes a true empirical improvement with statistical signficance. Implementing other methods for withholding data (e.g. random cropping) could also enhance this effect.

For magnitude warping, while our hyperparameters attuned for greater noise matching those of previous studies' resulted in a model that outperformed relative to training on data with less magnitude warping, neither outperformed the baseline after 30 epochs. The modest reductions in test error from greater magnitude warping also did not improve upon the test error reduction when combined with the greater spectrogram blackout discussed in the above section. We thus conclude that magnitude warping adds noise that does not reflect helpful real-world intersample variance for learning the invariant features from the input sEMG data.

Regarding dimensionality reduction, a dimensionality reduction to 576 features that was followed by a broadcast back to 768 features outperformed a transformation that only reduced input features to 576, but both still performed substantially worse than the baseline. It is thus likely that further

reducing dimensions past the baseline architecture's encoding leads to a loss in useful information for model training and prediction.

CNN+GRU Hyperparameter Exploration The primary issue with the CNN+GRU model during hyperparameter exploration was that the model showed improvements in training accuracy, however, test error either stagnated or worsened. This discrepancy strongly indicated overfitting, meaning that the model had memorized patterns in the training data rather than learning generalizable features. The first adjustment was to explore different hidden sizes and number layers. We found that it was best to have a GRU hidden size greater than the CNN's and only 2 layers for each. The hidden size affects how much information each layer retains and passes forward, while the number of layers determines the depth of feature extraction and sequence modeling [24]. Specifically, decreasing the hidden size of the CNN layer forces the model to learn more robust features, while increasing the hidden size of the GRU layer allows it to store and process more information from the input sequence. The number of layers for each was limited to two as the primary concern was overfitting.

To further mitigate overfitting, dropout was implemented as a regularization technique to force the model to learn more robust features rather than relying on training data patterns [25] [26]. Although dropout helped the model generalize a little better, the most significant improvement came from increasing the number of MLP features. The MLP feature size controls the complexity of the final prediction layers, impacting the model's ability to capture abstract representations of the data. Increasing the number of mlp features to 480 significantly improved the model test error, but increasing to 600 did not improve it much further. Although increasing the MLP feature size helped reduce test error, further improvements were limited, suggesting that additional model adjustments, such as tuning the learning rate or exploring alternative regularization techniques, may be necessary.

Transformer Hyperparameter Exploration We systematically explored several key hyperparameters to optimize the CNN+Transformer architecture. The primary factors included:
Number of Transformer Layers: We tested configurations with 1, 2, and 4 transformer layers. Both 1-layer and 4-layer models performed similarly, but the 2-layer model underperformed, likely due to an ineffective balance between feature extraction and sequence modeling.
Depth of CNN Backbone: Deeper CNNs extracted richer feature representations but significantly increased training time. We balanced model complexity and efficiency by selecting an optimal depth that preserved performance while maintaining reasonable training duration with the original baseline parameters being ideal of the tested values.
Training Duration: Training beyond 30 epochs led to diminishing returns with the validation loss stagnating a bit above 0.6 while the training loss kept falling down to around 0.4, but extending to 50 epochs allowed the model to reach a final testing CER of 22.2 and observe the plateau of the validation loss. Due to limited compute available and the size of the model, experimentally testing on more than 50 epochs was not feasible.

The CNN+Transformer hybrid model substantially outperformed the standalone transformer, demonstrating the importance of combining convolutional feature extraction with self-attention mechanisms. The final optimized model achieved a CER of 20.2 after 50 epochs, marking a significant improvement over initial transformer-only attempts. However, it failed to reach the same performance as other models, indicating that the heavy weighting that LSTMs and RNNs place on the data is important for the type of data used. The movements made by participants in short time frames, which are better identified by LSTMs, are likely more telling of the intended key pressed than the long-term patterns which the transformer is better at identifying.

Given the constraints on computational resources and available time, several potential avenues for improvement remain unexplored - One promising direction is the integration of decoder-based transformer architectures, which may enhance sequence generation and output modeling. Additionally, experimenting with a hybrid Transformer+LSTM+CNN model could leverage the respective strengths of each component—local feature extraction from CNNs, sequential pattern retention from LSTMs, and global attention mechanisms from transformers. Other model combinations relying on the CNN later could help creating a richer embedding for the text before classifying it. Beyond architectural modifications, further refinements in preprocessing techniques could improve performance. A more exhaustive mix-and-match approach between different models and preprocessing strategies could yield potentially better configurations.

References

[1] Wearable emg sensor – revolutionizing muscle monitoring, Dec 2024.

[2] Jie Zhang, Yan Xing, Xiuli Ma, and Liqun Feng. Differential diagnosis of parkinson disease, essential tremor, and enhanced physiological tremor with the tremor analysis of emg. *Parkinson's Disease*, 2017(1):1597907, 2017.

[3] Guillermo Iglesias, Edgar Talavera, Ángel González-Prieto, Alberto Mozo, and Sandra Gómez-Canaval. Data augmentation techniques in time series domain: a survey and taxonomy. *Neural Computing and Applications*, 35(14):10123–10145, March 2023.

[4] Terry T. Um, Franz M. J. Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ICMI '17. ACM, November 2017.

[5] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, IJCAI-2021, page 4653–4660. International Joint Conferences on Artificial Intelligence Organization, August 2021.

[6] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *PLOS ONE*, 16(7):e0254841, jul 2021.

[7] Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. Tsgm: A flexible framework for generative modeling of synthetic time series. *arXiv preprint arXiv:2305.11567*, 2023.

[8] Roman Panarin. Basic data augmentation method applied to time series, Oct 2024.

[9] Francois Chollet. Building autoencoders in keras, May 2016.

[10] Alireza Keshavarz. Image denoising using autoencoders (improved version), Jul 2023.

[11] 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.

[12] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[13] Shouliang Song, Anming Dong, Jiguo Yu, Yubing Han, and You Zhou. A multichannel cnn-gru hybrid architecture for semg gesture recognition. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 4132–4139, 2023.

[14] Naveen Kumar Karnam, Shiv Ram Dubey, Anish Chand Turlapaty, and Balakrishna Gokaraju. Emghandnet: A hybrid cnn and bi-lstm architecture for hand activity classification using surface emg signals. *Biocybernetics and Biomedical Engineering*, 42(1):325–340, 2022.

[15] N. Dua, S.N. Singh, and V.B. Semwal. Multi-input cnn-gru based human activity recognition using wearable sensors. *Computing*, 103:1461–1478, 2021.

[16] I.H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2:420, 2021.

[17] Ankit Vijayvargiya, Bharat Singh, Rajesh Kumar, Usha Desai, and Jude Hemanth. Hybrid deep learning approaches for semg signal-based lower limb activity recognition. *Mathematical Problems in Engineering*, 2022:3321810, 2022.

[18] Y. Sun, J. Zhang, Z. Yu, Y. Zhang, and Z. Liu. Bidirectional long short-term neural network based on the attention mechanism of the residual neural network (resnet-bilstm-attention) predicts porosity through well logging parameters. *ACS Omega*, 8(26):24083–24092, 2023.

[19] Davi Guimarães da Silva and Anderson Alvarenga de Moura Meneses. Comparing long short-term memory (lstm) and bidirectional lstm deep neural networks for power consumption prediction. *Energy Reports*, 10:3315–3334, 2023.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[21] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888, 2018.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, 2019.

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[24] Stefano Marrone, Cristina Papa, and Carlo Sansone. Effects of hidden layer sizing on cnn fine-tuning. *Future Generation Computer Systems*, 118:48–55, 2021.

[25] Pavel V. Matrenin, Vadim Z. Manusov, Alexandra I. Khalyasmaa, Dmitry V. Antonenkov, Stanislav A. Eroshenko, and Denis N. Butusov. Improving accuracy and generalization performance of small-size recurrent neural networks applied to short-term load forecasting. *Mathematics*, 8(12), 2020.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.