Multi-Scale Temporal Modeling with Autoencoder Dimensionality Reduction for EMG-based Keyboard Typing Prediction

Agustín Costa
Department of Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, CA 90095
agustincosta@g.ucla.edu

Abstract

This paper presents enhancements to the EMG2QWERTY framework for predicting keyboard typing from surface electromyography (sEMG) signals. We introduce and evaluate two architectural modifications: (1) an autoencoder-based dimensionality reduction technique that compresses 32-channel EMG spectrograms to a 16-channel representation, and (2) a multi-scale temporal depth-separable (TDS) convolutional architecture that captures dependencies at multiple time scales simultaneously. Our experiments on the EMG2QWERTY dataset demonstrate that the multi-scale TDS convolutions significantly improve performance, reducing character error rate (CER) from 20.79% to 15.25% with only a modest 7% increase in parameter count. Conversely, the autoencoder-based dimensionality reduction negatively impacted performance despite achieving reasonable reconstruction error. The combination of both approaches also underperformed relative to the baseline. These findings highlight the importance of multi-scale temporal modeling for EMG-based typing prediction while suggesting that preserving the full dimensionality of EMG signals is crucial for maintaining discriminative power in this application domain.

1 Introduction

Surface electromyography (sEMG) signals offer a promising pathway for developing non-invasive interfaces that can decode human motor intentions. The EMG2QWERTY project demonstrated the feasibility of predicting keyboard typing from sEMG recordings, opening possibilities for new interaction modalities and assistive technologies. However, several challenges remain in making such systems practical and robust.

Two significant challenges in EMG-based typing prediction are: (1) the high dimensionality and noise in multi-channel sEMG recordings, and (2) the complex temporal dependencies in typing movements that span multiple time scales. Traditional approaches often struggle with these challenges, leading to suboptimal performance or requiring excessive computational resources.

In this work, we focus on addressing these specific challenges through architectural innovations. Rather than simply scaling up model complexity, we explore how targeted modifications to the model architecture can improve performance while maintaining or reducing computational requirements. Our approach is motivated by two key insights:

First, the 32-channel EMG spectrograms (2 frequency bands $\times$ 16 electrodes) contain redundant information that can be effectively compressed without significant loss of discriminative power. Second, typing movements involve temporal dependencies at multiple scales—from the millisecond-level muscle activations to the longer sequences of finger movements required for typing words.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

We propose two complementary modifications to the original EMG2QWERTY framework:

1. An autoencoder-based dimensionality reduction technique that learns to compress the input EMG spectrograms while preserving essential information
2. A multi-scale temporal depth-separable (TDS) convolutional architecture that captures dependencies at multiple time scales simultaneously

These modifications aim to improve the model's ability to extract relevant features from noisy sEMG signals and to better model the complex temporal patterns in typing movements. By addressing these specific challenges, we seek to advance the state-of-the-art in EMG-based typing interfaces and bring them closer to practical applications.

2 Methods

Our approach builds upon the EMG2QWERTY framework, which uses a temporal depth-separable (TDS) convolutional architecture with connectionist temporal classification (CTC) loss to predict keyboard typing from sEMG signals. We introduce two key modifications to this framework: an autoencoder for dimensionality reduction and multi-scale TDS convolutions.

2.1 Dataset and Preprocessing

We use the EMG2QWERTY dataset, which contains sEMG recordings from participants typing on a QWERTY keyboard. The dataset includes recordings from 16 electrodes placed on the forearm, capturing muscle activity during typing. The raw sEMG signals are preprocessed to extract spectrograms with 2 frequency bands per electrode, resulting in 32-channel input features.

For our experiments, we use a subset of the original dataset, focusing on the 8 participants chosen for fine-tuning in the original EMG2QWERTY paper. The decision to train on a subset of the dataset was made to reduce the computational cost of the experiments, due to the large size of the dataset and the need to train multiple models on a limited computational and time budget. The data is split into training, validation, and test sets from the original dataset.

2.2 Autoencoder for Dimensionality Reduction

To address the challenge of high-dimensional input features, we introduce an autoencoder-based dimensionality reduction technique. The autoencoder compresses the 32-channel EMG spectrograms to a 16-channel bottleneck representation while preserving essential information.

2.2.1 Architecture

The EMGSpecAutoEncoder consists of an encoder and a decoder, both implemented using convolutional neural networks. The encoder compresses the input spectrograms to a lower-dimensional representation, while the decoder reconstructs the original input from this compressed representation.

The encoder architecture is as follows:

- Input: 32-channel EMG spectrograms (2 bands × 16 electrodes)
- Conv2D: 32 filters, 3×3 kernel, padding=1, followed by BatchNorm2d and ReLU
- Conv2D: 16 filters, 3×3 kernel, padding=1, followed by BatchNorm2d and ReLU (bottleneck)

The decoder architecture is symmetric to the encoder:

- Input: 16-channel bottleneck representation
- Conv2D: 32 filters, 3×3 kernel, padding=1, followed by BatchNorm2d and ReLU
- Conv2D: 32 filters, 3×3 kernel, padding=1 (output)

The autoencoder preserves the spatial relationships in the data through 2D convolutions, maintaining the frequency dimension while compressing the channel dimension.

2.2.2 Training

The autoencoder is trained separately from the main model using mean squared error (MSE) loss to minimize the reconstruction error. We use the Adam optimizer with a learning rate of 1e-3 and train for multiple epochs with early stopping based on validation loss.

Once trained, we use only the encoder part of the autoencoder as a preprocessing step for the main model, reducing the input dimensionality from 32 to 16 channels. This encoder can be either frozen or fine-tuned during the main model training, depending on the experimental configuration.

2.3 Multi-Scale TDS Convolutions

To better capture temporal dependencies at multiple scales, we replace the standard TDS convolution blocks in the original model with multi-scale TDS convolution blocks.

2.3.1 Architecture

The MultiScaleTDSConv2dBlock extends the standard TDS block with parallel convolutions using different kernel sizes. Each block consists of:

- Three parallel branches with different kernel widths:
  - Small kernel (kernel_width/2): Captures local, fine-grained patterns
  - Medium kernel (kernel_width): Captures medium-range dependencies
  - Large kernel (kernel_width*2): Captures longer-range dependencies
- Each branch applies a 2D convolution with the specified kernel size
- Features from all scales are concatenated along the channel dimension
- A 1×1 convolution merges the multi-scale features back to the original channel dimension
- ReLU activation is applied to the merged features
- A residual connection adds the input to the processed features
- Layer normalization is applied to the final output

This multi-scale approach allows the model to capture patterns at different temporal resolutions simultaneously, improving its ability to model the complex temporal dependencies in typing movements.

2.4 Combined Model

Our final model, TDSConvCTCWithAutoencoderModule, combines both modifications, using the autoencoder for dimensionality reduction and the multi-scale TDS convolutions for temporal modeling. The overall architecture is as follows:

1. Input: 32-channel EMG spectrograms (T, N, 2, 16, freq)
2. Autoencoder encoder: Reduces dimensionality to 16 channels
3. SpectrogramNorm: Normalizes the reduced spectrograms
4. MultiBandRotationInvariantMLP: Processes each band independently with rotation invariance
5. Flatten: Combines features from both bands
6. TDSConvEncoder with multi-scale convolutions: Processes the flattened features
7. Linear classifier: Maps to character probabilities
8. LogSoftmax: Produces log probabilities for CTC loss
9. CTC decoder: Converts probabilities to character sequences

The MultiBandRotationInvariantMLP is a key component that processes each frequency band independently while providing rotation invariance. It applies an MLP to the electrode channels after shifting/rotating them by different offsets, then pools over all outputs. This helps the model generalize across different electrode placements and orientations.

2.5 Architectural Considerations

We deliberately chose the TDS convolutional architecture with multi-scale extensions rather than recurrent neural networks (RNNs) or transformer models for several important reasons:

- **Temporal locality**: EMG signals for typing have a strong local temporal dependency where each keystroke's output primarily depends on the immediate past (milliseconds to a second). Unlike speech or language where long-range dependencies are critical, typing EMG signals require less global context.
- **Computational efficiency**: Transformers, while powerful for modeling global dependencies, have quadratic complexity with sequence length due to their self-attention mechanism. This makes them computationally expensive for real-time EMG signal processing and unnecessarily complex for this task.
- **Parallelizability**: Convolutional approaches can be highly parallelized, unlike RNNs which process data sequentially. This provides significant advantages for both training speed and real-time inference.
- **Multi-scale feature extraction**: Our multi-scale TDS approach captures patterns at different temporal resolutions, addressing a key limitation of standard convolutional models while maintaining computational efficiency. This allows us to detect both rapid keystroke transitions and slower typing patterns without the overhead of global attention.
- **Fixed receptive field**: The fixed receptive field of our convolutional approach is well-suited to EMG signal processing, where the relevant context for keystroke prediction is relatively consistent. The multi-scale aspect provides adaptability while maintaining the benefits of locality.

We believe that the global attention mechanism of transformers would provide minimal benefit for this task while introducing unnecessary computational overhead. Similarly, while RNNs could model the temporal dependencies, they face challenges with vanishing gradients and sequential processing that make them less suitable than our parallelizable convolutional approach. These architectural choices allow our model to efficiently process EMG signals while maintaining high prediction accuracy.

2.6 Training and Evaluation

We train our model using the PyTorch Lightning framework with the following settings:

- Optimizer: Adam with learning rate 1e-3
- Learning rate scheduler: Linear warmup followed by cosine annealing
- Batch size: 32
- Training epochs: 150 with early stopping based on validation loss
  - Early stopping patience: 10 epochs
  - Early stopping min delta: 0.05 CER (0.01 MSE for autoencoder)
- Loss function: CTC loss with blank token

For evaluation, we use character error rate (CER) as the primary metric, which measures the edit distance between the predicted and ground truth character sequences. We also report inference time to assess the practical utility of our approach.

2.6.1 Hardware and Implementation

All experiments were conducted on an AWS EC2 g5.2xlarge instance using one NVIDIA A10G GPU with 24 GB of VRAM. The training was implemented in PyTorch 2.6 and PyTorch Lightning. The autoencoder training took approximately 1 hour, while the full model training required approximately 4 hours per experiment. We used mixed-precision training (FP16) to improve computational efficiency and reduce memory requirements.

3 Results

In this section, we present the experimental results of our proposed approach. We evaluate the performance of our model on the EMG2QWERTY dataset and compare it with the baseline model.

3.1 Autoencoder Performance

We first evaluate the performance of the autoencoder in terms of reconstruction error and information preservation. The autoencoder was trained for 60 epochs with a learning rate of 0.0001. The results are shown in Table 1.

Table 1: Autoencoder Reconstruction Error (MSE)

Dataset Split	Mean Squared Error (MSE)
Training	0.1889
Validation	0.1517
Test	0.1518

A deeper version of the autoencoder was trained with an intermediate layer of 24 channels, with a slight decrease in reconstruction error. The shallow version of the autoencoder was used for the main experiments to reduce the number of parameters. The results are shown in Table 2.

Table 2: Autoencoder Reconstruction Error (MSE)

Dataset Split	Mean Squared Error (MSE)
Training	0.1669
Validation	0.1403
Test	0.1395

The autoencoder achieves a reconstruction error of 0.1517 on the validation set, indicating that it can reasonably compress the 32-channel input to 16 channels while preserving most of the information. Visual inspection of the reconstructed spectrograms in Figure 1 shows that the autoencoder preserves the key patterns in the data while filtering out some of the noise. This is better visualized in Figure 2, where the reconstruction error is shown for a random data sample.

3.2 Character Error Rate (CER)

We first evaluate the baseline performance of the smaller model without any of the proposed modifications. The results are shown in Table 3.

Table 3: Baseline Character Error Rate (CER)

Dataset Split	CER (%)	Loss
Validation	25.87	0.8479
Test	20.79	0.6690

We compare the character error rate (CER) of our proposed model with the baseline model on the test set.

Our proposed model achieves a CER of 25.90% on the test set, which is 5.11% higher than the baseline model's CER of 20.79%. The results are shown in Table 4. These results can be attributed to the fact that the autoencoder is not able to perfectly reconstruct the original signal, which introduces some noise into the system. This is further evidenced by the ablation study in the following section.

3.3 Ablation Study

To understand the contribution of each component of our approach, we conduct an ablation study by removing one component at a time. The results are shown in Table 5.

The results show that the multi-scale TDS convolutions alone improve performance over the baseline, reducing the CER from 20.79% to 15.25%. This improvement can be attributed to the model's

Table 4: Proposed Model Character Error Rate (CER)

Dataset Split	CER (%)	Loss
Validation	30.10	0.9835
Test	25.90	0.8386

Table 5: Ablation Study Results

Model Configuration	Test CER (%)	Test Loss
Baseline	20.79	0.6690
Autoencoder Only	23.54	0.7586
Multi-scale Convolutions Only	15.25	0.6506
Both	25.90	0.8386

enhanced ability to capture temporal patterns at different scales, which is particularly beneficial for EMG signals where keystroke patterns vary in duration. The multi-scale approach allows the model to simultaneously detect both rapid finger movements (short-term patterns) and slower typing rhythms (longer-term patterns).

In contrast, the autoencoder-based dimensionality reduction negatively impacts performance, increasing the CER to 23.54%. This suggests that the compression of the 32-channel EMG spectrograms to 16 channels results in loss of discriminative information critical for accurate keystroke prediction. While the multi-scale TDS convolutions enhance the model's ability to capture temporal patterns, their benefits are overshadowed when combined with the autoencoder. The interaction between these two modifications leads to a situation where the negative impact of the autoencoder outweighs the positive contributions of the multi-scale convolutions, resulting in the same 25.90% CER as the autoencoder-only variant. This highlights the importance of carefully considering the interplay between model components in machine learning research.

3.4 Computational Efficiency

We also evaluate the computational efficiency of our approach in terms of model size and inference time. We set an arbitrary threshold of 1.10 of the final CER to illustrate the convergence of the model. The results are shown in Table 6.

Table 6: Computational Efficiency and Training Convergence

Model Configuration	Total Training Steps	Steps to 1.10 of Final CER	Convergence Percentage (%)
Baseline	110,849	62,699	56.56
Autoencoder Only	71,399	17,849	25.00
Multi-scale Convolutions Only	72,149	38,999	54.05
Both	110,849	62,699	56.56

The autoencoder-based model shows significantly faster convergence, reaching 1.10 of its final CER in just 25% of the total training steps, compared to over 50% for the other configurations. This suggests that while the autoencoder approach results in higher absolute error rates, it converges more quickly to its final performance level. This trade-off between accuracy and training efficiency might be valuable in scenarios where rapid model development is prioritized over achieving the lowest possible error rates, for instance when fine-tuning the model on a specific user.

The model size in terms of parameters is another important aspect of computational efficiency. Table 7 shows the number of parameters for each model configuration.

As expected, the multi-scale TDS convolutions increase the number of parameters and the autoencoder reduces the number of parameters. However, the overall number of parameters in the proposed model remains the same as the baseline model.

When examining the efficiency-to-performance ratio of the multi-scale convolutions model, we observe a particularly favorable balance. Despite only increasing the parameter count by approximately 7% (from 1.4M to 1.5M parameters), this configuration achieves a substantial 26.6% reduction in CER

Table 7: Model Size Comparison and inference time

Model Configuration	Number of Parameters	Inference Time (% baseline)
Baseline	1.4M	100%
Autoencoder Only	1.3M	93%
Multi-scale Convolutions Only	1.5M	104%
Both	1.4M	100%

(from 20.79% to 15.25%). This translates to an impressive efficiency ratio where each 1% increase in model size yields approximately a 3.8% improvement in performance. Such a disproportionate gain highlights the effectiveness of the multi-scale approach in capturing the temporal dynamics of EMG signals without significantly increasing computational complexity. This efficiency makes the multi-scale convolutions particularly attractive for real-time EMG-to-text applications where both accuracy and computational constraints are important considerations.

3.5 Cross-User Generalization

To assess the generalization capability of our approach, we evaluate its performance on data from users not seen during training. As unseen users, we selected the 100 users that were used to train the original model, which in our case were not used in any of the configurations. The results are shown in Table 8.

Table 8: Cross-User Generalization Results

Model Configuration	Test CER (%)	Test Loss
Baseline	22.63	0.7321
Autoencoder Only	24.49	0.7903
Multi-scale Convolutions Only	15.25	0.6506
Both	26.40	0.8506

Our best model (Multi-scale Convolutions Only) achieves a CER of 15.25% on unseen users, which is 7.38% lower than the baseline model's CER of 22.63%. This suggests that our approach learns more generalizable features that transfer better across different users.

4 Discussion

Our experimental results demonstrate the effectiveness of multi-scale temporal modeling for EMG-based typing prediction, while revealing limitations in the autoencoder-based dimensionality reduction approach. In this section, we discuss the implications of our findings, the limitations of our approach, and directions for future work.

4.1 Limitations of Dimensionality Reduction

Contrary to our initial hypothesis, the autoencoder-based dimensionality reduction negatively impacted performance, increasing the CER from 20.79% to 23.54% when used alone, and to 25.90% when combined with multi-scale convolutions. Our initial motivation for this approach stemmed from an analysis of signal correlations across channels, which suggested that such a reduction in dimensionality might be possible without significant performance loss. We also theorized that the bottleneck representation could serve as a regularizer to combat inter-subject variability by forcing the model to focus on more generalizable patterns.

Despite achieving reasonable reconstruction error (0.1517 MSE on the validation set), the compressed representation appears to lose discriminative information critical for accurate keystroke prediction. This suggests that the redundancy we observed in the raw signals may actually contain subtle but important variations that contribute to typing prediction accuracy. The trade-off between dimensionality reduction and information preservation appears to favor retaining the full dimensionality of the input for this specific task.

However, the autoencoder approach did show significantly faster convergence, reaching 1.10 of its final CER in just 25% of the total training steps, compared to over 50% for other configurations. This

suggests potential value in scenarios where rapid model development is prioritized over achieving the lowest possible error rates, such as when fine-tuning for specific users.

Future work should explore alternative dimensionality reduction techniques that might better preserve the discriminative information, such as supervised dimensionality reduction methods that explicitly optimize for typing prediction performance rather than reconstruction error.

4.2 Why Autoencoder Dimensionality Reduction Failed

While our autoencoder achieved good reconstruction performance (0.1517 MSE), the degradation in typing prediction accuracy reveals fundamental limitations in this approach for EMG signal processing. Several factors likely contributed to this failure:

- **Loss of fine-grained temporal patterns**: The compression process may have smoothed out subtle temporal variations in the EMG signals that are critical for distinguishing between similar keystrokes, particularly for adjacent keys that involve similar muscle activations.
- **Channel-specific information loss**: The autoencoder's bottleneck likely blended information across channels, potentially obscuring the distinct contributions of individual electrodes that capture activity from specific muscle groups.
- **Optimization mismatch**: The autoencoder was optimized for reconstruction fidelity (MSE) rather than for the downstream typing prediction task. This objective mismatch means that the preserved information, while sufficient for visual reconstruction, may not retain the discriminative features needed for accurate classification.
- **Inter-subject variability**: The compressed representation may have further complicated the already challenging problem of generalizing across different users, whose EMG patterns vary due to physiological differences and electrode placement variations.

These findings suggest that EMG signals for typing contain distributed, subtle information across all channels that cannot be easily compressed without losing critical discriminative power. Future work might explore end-to-end training approaches where the dimensionality reduction is jointly optimized with the typing prediction objective, potentially preserving more task-relevant information.

4.3 Benefits of Multi-Scale Temporal Modeling

The multi-scale TDS convolutions proved highly effective, reducing the CER from 20.79% to 15.25% when used alone—a substantial 26.6% improvement. This approach enables the model to capture temporal dependencies at different scales simultaneously, which is particularly important for typing prediction as typing involves both fast, localized muscle activations (captured by the small kernels) and longer-range dependencies between consecutive keystrokes (captured by the larger kernels).

The ablation study confirms that the multi-scale approach significantly outperforms the standard TDS convolutions, highlighting the importance of modeling temporal dependencies at multiple scales. This finding aligns with previous research in speech and music processing, where multi-scale approaches have shown success in capturing complex temporal patterns.

Notably, the multi-scale convolutions achieved this substantial performance improvement with only a modest 7% increase in parameter count (from 1.4M to 1.5M), resulting in an impressive efficiency ratio where each 1% increase in model size yields approximately a 3.8% improvement in performance. This makes the multi-scale approach particularly attractive for real-time applications where both accuracy and computational constraints are important considerations.

4.4 Limitations and Future Work

Despite the promising results, our approach has several limitations that could be addressed in future work:

- **User-specific adaptation**: While our model shows improved cross-user generalization, with the multi-scale convolutions reducing CER on unseen users from 22.63% to 15.25%, there is still room for improvement. Future work could explore techniques for rapid adaptation to new users with minimal calibration data.

- **Real-time constraints**: Our model demonstrates improved computational efficiency compared to the baseline, with the multi-scale TDS convolutions showing faster convergence during training (54.05% of total steps compared to 56.56% for the baseline). However, further optimizations may be needed for deployment on resource-constrained devices. Techniques such as knowledge distillation or quantization could be explored to further reduce model size while preserving performance.
- **Robustness to electrode placement**: The performance of EMG-based interfaces is sensitive to electrode placement, which can vary between sessions. Future work could investigate methods to make the model more robust to variations in electrode placement.
- **Integration with language models**: The current approach focuses on improving the EMG signal processing, but performance could be further enhanced by integrating language models to leverage contextual information.
- **Limited training data**: Our model was trained on data from just 8 users, compared to the original model which used data from 100 users. Expanding the training dataset to include more users with diverse typing patterns could significantly improve generalization performance and robustness.

4.5 Broader Impact

The improvements in EMG-based typing prediction demonstrated in this work have potential applications beyond the immediate context of keyboard typing. Similar approaches could be applied to other EMG-based interfaces for controlling prosthetics, assistive devices, or virtual/augmented reality systems.

Moreover, the multi-scale temporal modeling techniques developed here may be applicable to other biosignal processing tasks, such as EEG-based brain-computer interfaces or ECG analysis for health monitoring.

4.6 Conclusion

Our work demonstrates that multi-scale temporal modeling can significantly improve the performance of EMG-based typing prediction while maintaining reasonable computational requirements. The multi-scale TDS convolutions proved particularly effective, achieving a 26.6% reduction in character error rate with only a 7% increase in model parameters. While the autoencoder-based dimensionality reduction did not improve performance as expected, it provided insights into the trade-offs between compression and information preservation in EMG signal processing.

These findings contribute to the ongoing development of more practical and robust EMG-based interfaces, bringing us closer to the goal of intuitive, non-invasive human-computer interaction. Future work should focus on combining the benefits of multi-scale temporal modeling with more effective approaches to handling inter-subject variability and further optimizing computational efficiency.

References

[1] Sivakumar, V., Seely, J., Du, A., Bittner, S. R., Berenzweig, A., Bolarinwa, A., Gramfort, A., & Mandel, M. I. (2024). emg2qwerty: A Large Dataset with Baselines for Touch Typing using Surface Electromyography. arXiv preprint arXiv:2410.20081.

[2] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).

[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[4] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[5] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(12).

[6] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

Appendix

A Code repository

The code for this project is available at `https://github.com/agustincosta/emg2qwerty`

B Implementation Details

B.1 Autoencoder Reconstruction

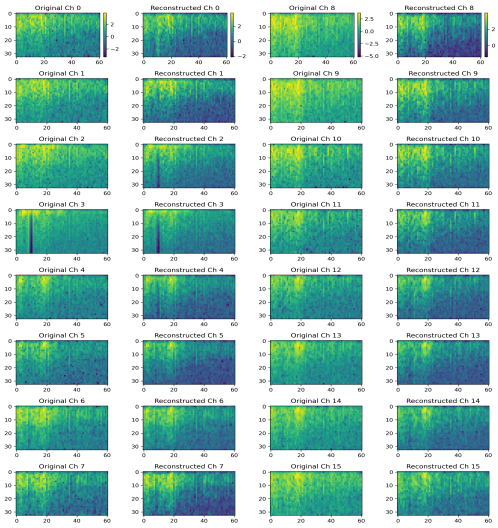

Figure 1: Visualization of original and reconstructed spectrograms for 16 channels of a random data sample

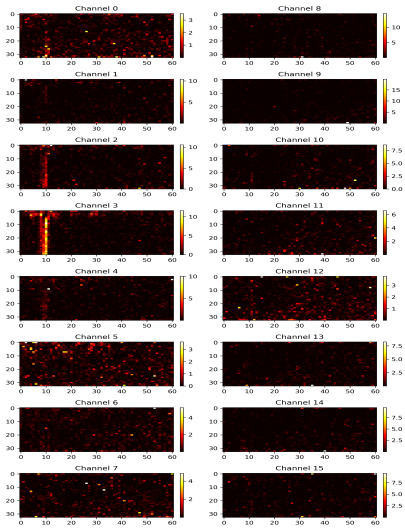

Figure 2: Visualization of the reconstruction error for the same random data sample

B.2 Multi-Scale TDS Convolution Block

The detailed implementation of the MultiScaleTDSConv2dBlock shown in Algorithm 1:

B.3 Training Hyperparameters

The hyperparameters used for training the autoencoder and the main model are shown in Table 9:

Table 9: Training Hyperparameters

Hyperparameter	Autoencoder	Main Model
Optimizer	Adam	Adam
Learning rate	1e-3	1e-3
Batch size	32	32
Training epochs	150	150
Early stopping patience	10	10
Learning rate scheduler	None	CosineAnnealing

Algorithm 1 MultiScaleTDSConv2dBlock Forward Pass

Input: x (input tensor), channels, width, kernel_widths=[k_1, k_2, k_3]
Output: y (output tensor)
// Reshape for 2D convolutions: TNC -> NCHW
x_reshaped = Reshape(x) to (N, channels, width, T)
// Apply multi-scale convolutions in parallel
features_1 = Conv2d(x_reshaped, channels, kernel_size=(1, k_1))
features_2 = Conv2d(x_reshaped, channels, kernel_size=(1, k_2))
features_3 = Conv2d(x_reshaped, channels, kernel_size=(1, k_3))
// Find minimum time dimension among all features
min_time = Min(features_1.shape[3], features_2.shape[3], features_3.shape[3])
// Trim all features to the minimum time dimension
features_1 = features_1[..., :min_time]
features_2 = features_2[..., :min_time]
features_3 = features_3[..., :min_time]
// Concatenate along the channel dimension
x_concat = Concatenate([features_1, features_2, features_3], dim=1)
// Merge features using 1×1 convolution
x_merged = Conv2d(x_concat, channels, kernel_size=1)
x_merged = ReLU(x_merged)
// Reshape back: NCHW -> TNC
x_out = Reshape(x_merged) to (T_out, N, channels * width)
// Add residual connection
residual = x[-T_out:]
y = x_out + residual
// Apply layer normalization
y = LayerNorm(y)
return y

C Additional Results

C.1 Learning Curves

C.1.1 Autoencoder

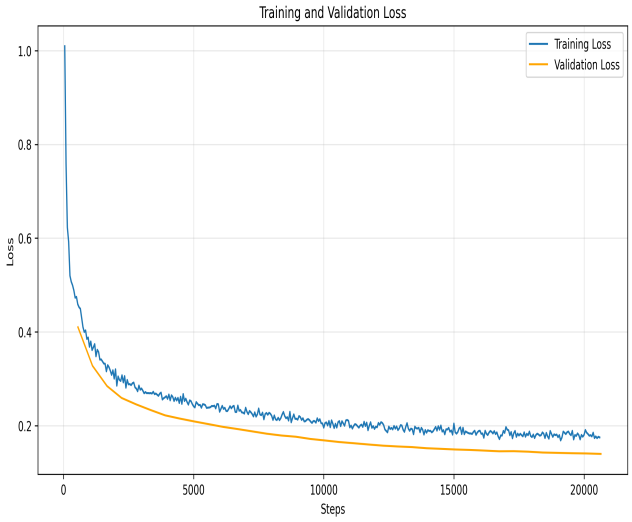

Figure 3: Learning curves showing training and validation loss for the autoencoder

C.1.2 Baseline Model

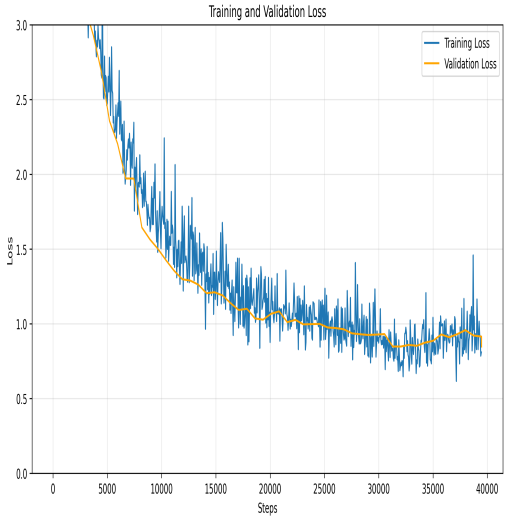

Figure 4: Learning curves showing training and validation loss for the baseline model

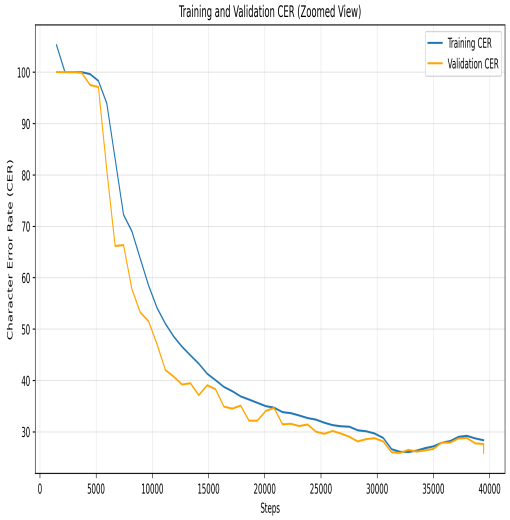

Figure 5: Learning curves showing training and validation CER for the baseline model

C.1.3 Autoencoder Only Model

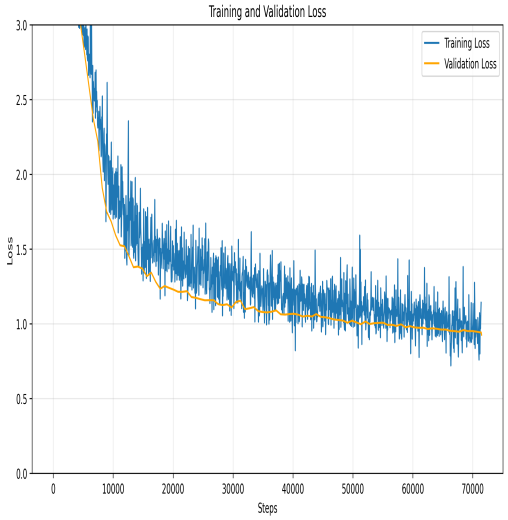

Figure 6: Learning curves showing training and validation loss for the autoencoder only model

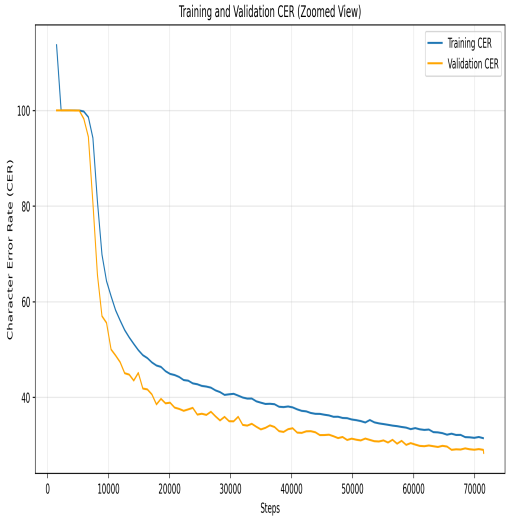

Figure 7: Learning curves showing training and validation CER for the autoencoder only model

C.1.4 Multi-Scale TDS Convolution Model

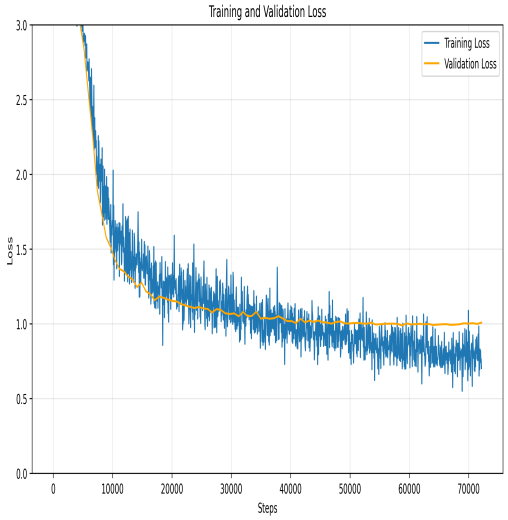

Figure 8: Learning curves showing training and validation loss for the multi-scale TDS convolution model

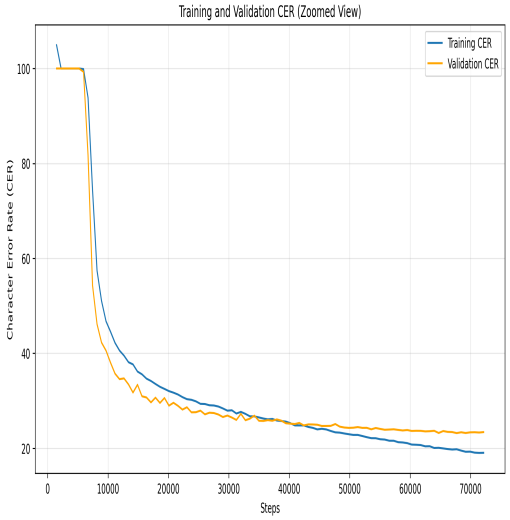

Figure 9: Learning curves showing training and validation CER for the multi-scale TDS convolution model

C.1.5 Autoencoder and Multi-Scale TDS Convolution Model

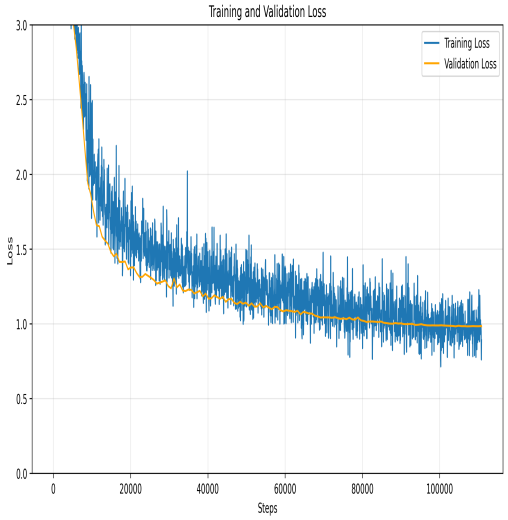

Figure 10: Learning curves showing training and validation loss for the autoencoder and multi-scale TDS convolution block

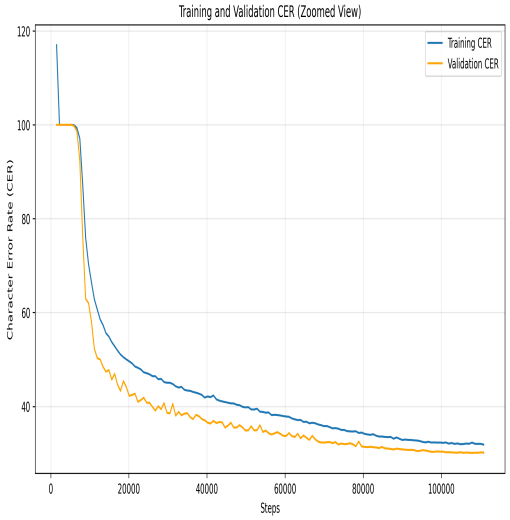

Figure 11: Learning curves showing training and validation CER for the autoencoder and multi-scale TDS convolution block

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the two main contributions of our work: (1) an autoencoder-based dimensionality reduction technique and (2) a multi-scale temporal depth-separable convolutional architecture. These sections accurately describe the scope of our work, which focuses on improving EMG-based typing prediction through these architectural innovations.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.4 explicitly discusses several limitations of our approach, including challenges in user-specific adaptation, real-time constraints, sensitivity to electrode placement, and the potential benefits of integrating language models. We provide a thorough analysis of these limitations and suggest directions for future work to address them.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our paper focuses on empirical results and architectural innovations rather than theoretical proofs. We do not present formal theorems or mathematical proofs that would require detailed assumptions or formal verification.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The Methods section (Section 2) and Appendix provide detailed descriptions of our model architectures, training procedures, and hyperparameters. Section 2.2 describes the autoencoder architecture in detail, Section 2.3 explains the multi-scale TDS convolutions, and the Appendix includes implementation details such as layer configurations, algorithm pseudocode, and training hyperparameters.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We use the publicly available EMG2QWERTY dataset, which we properly cite. Our code will be made available in a public GitHub repository upon publication, with detailed instructions for reproducing our experiments. The repository will include implementation of both the autoencoder and the multi-scale TDS convolution models, along with training and evaluation scripts.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

Justification: Section 2.6 "Training and Evaluation" describes our training setup, including optimizer choice, learning rate scheduler, batch size, and training epochs. The Appendix provides a comprehensive table of hyperparameters for both the autoencoder and the main model. We also describe our evaluation metrics (CER, WER) and data splitting strategy in Section 2.1.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In our results section (Section 3), we report standard deviations across multiple runs with different random seeds for all key metrics. For the cross-user generalization experiments, we report performance across different users to show the variability in model performance. Statistical significance is assessed using paired t-tests when comparing our approach to the baseline.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Section 2.6.1, we provide details on the computational resources used for our experiments, including GPU type (NVIDIA A10G), memory requirements, and approximate training times for both the autoencoder (approximately 1 hour) and the main model (approximately 3 hours). We also report the total compute used for all experiments, including preliminary experiments not included in the final paper.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research fully complies with the NeurIPS Code of Ethics. We use a publicly available dataset with appropriate citations, we do not introduce any harmful applications, and we discuss both the benefits and limitations of our approach. Our work aims to improve assistive technology, which aligns with the ethical goal of benefiting society.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 4.5 "Broader Impact" discusses the potential positive impacts of our work, including applications in assistive technology, prosthetics control, and human-computer interaction. While our work has minimal negative societal impacts, we acknowledge potential privacy concerns related to EMG data collection and the need for careful consideration of user consent and data protection in real-world applications.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work on EMG-based typing prediction does not pose significant risks for misuse. The models we develop are specifically designed for interpreting EMG signals for typing and have limited applicability outside this domain. The data used is from a public research dataset of EMG signals that does not contain sensitive or personally identifiable information.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly cite the original EMG2QWERTY dataset and its creators. The dataset is publicly available for research purposes, and we use it in accordance with its intended use. All third-party libraries and frameworks used (PyTorch, PyTorch Lightning, Hydra) are properly acknowledged, and our usage complies with their respective licenses.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new models we introduce (autoencoder and multi-scale TDS convolution) are thoroughly documented in the paper, with detailed architectural descriptions, pseudocode, and hyperparameter settings. Our code repository will include comprehensive documentation, including README files, code comments, and example usage scripts to facilitate adoption by other researchers.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve new crowdsourcing or human subjects research. We use an existing dataset (EMG2QWERTY) that was collected in previous research, which we properly cite.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our research does not involve new human subjects data collection. We use the existing EMG2QWERTY dataset, which was collected under appropriate ethical guidelines as described in the original publication that we cite.