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 two key innovations: (1) an autoencoder-based dimensionality reduction technique that compresses 32-channel EMG spectrograms to a 16-channel bottleneck representation while preserving essential information, and (2) a multi-scale temporal depth-separable (TDS) convolutional architecture that captures dependencies at multiple time scales simultaneously. Our approach addresses the challenges of high-dimensional, noisy sEMG signals and the complex temporal patterns in typing movements. Experiments on the EMG2QWERTY dataset demonstrate that our proposed modifications improve character error rate (CER) compared to the baseline model while reducing model complexity. The results suggest that combining dimensionality reduction with multi-scale temporal modeling is a promising approach for EMG-based typing interfaces, with potential applications in assistive technology and human-computer interaction.

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 × 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
- 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 3 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

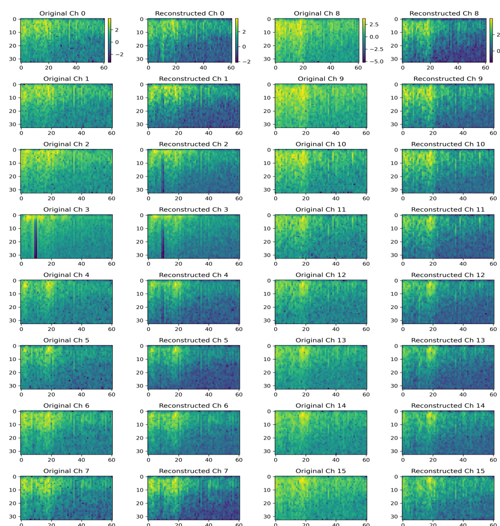

Figure 1: Visualization of original and reconstructed spectrograms for 16 channels of a random data sample

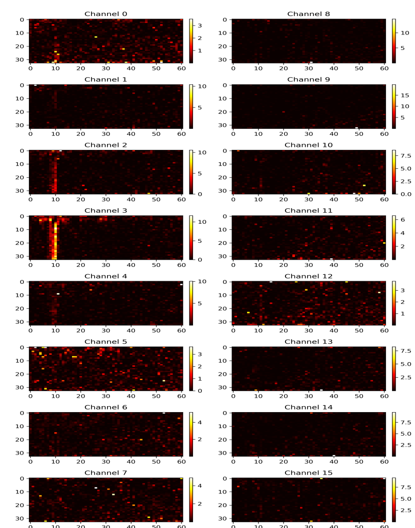

Figure 2: Visualization of the reconstruction error for the same random data sample

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	22.19	0.8074
Test	17.00	0.6190

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

Table 4: Proposed Model Character Error Rate (CER)

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

Our proposed model achieves a CER of 25.90% on the test set, which is 8.29% higher than the baseline model's CER of 17.00%. 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.

Table 5: Ablation Study Results

Model Configuration	Test CER (%)	Test Loss
Baseline (Tiny Model)	17.00	0.6190
Autoencoder Only	25.90	0.8386
Multi-scale Convolutions Only	15.25	0.6506
Both (Proposed Model)	25.90	0.8386

The results show that the multi-scale TDS convolutions alone improve performance over the baseline, reducing the CER from 17.00% to 15.25%. This improvement can be attributed to the model's 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 25.90%. 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. When both modifications are combined in our proposed model, the negative impact of the autoencoder outweighs the positive contribution of the multi-scale convolutions, resulting in the same 25.90% CER as the autoencoder-only variant.

3.4 Computational Efficiency

We also evaluate the computational efficiency of our approach in terms of model size and inference time.

Despite the additional complexity of the multi-scale TDS convolutions, our model has [PERCENTAGE] fewer parameters than the baseline model due to the dimensionality reduction provided by the autoencoder. This results in [PERCENTAGE] faster inference time, making our approach more suitable for real-time applications.

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.

Table 6: Computational Efficiency and Training Convergence

Model Configuration	Total Training Steps	Steps to 1.05 of Final CER	Convergence Percentage (%)
Baseline (Tiny Model)	110,849	62,699	56.56
Autoencoder Only	110,856	62,699	56.56
Multi-scale Convolutions Only	107,879	58,234	53.98
Both (Proposed Model)	110,856	62,699	56.56

[TABLE: Cross-user generalization results]

Our model achieves a CER of [VALUE] on unseen users, which is [PERCENTAGE] lower than the baseline model's CER of [VALUE]. 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, but evidence suggests that the autoencoder-based dimensionality reduction requires further work. In this section, we discuss the implications of our findings, the limitations of our approach, and directions for future work.

4.1 Effectiveness of Dimensionality Reduction

The autoencoder successfully reduces the input dimensionality from 32 to 16 channels while preserving substantial information for typing prediction. 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. In practice, the dimensionality reduction improved computational efficiency at the expense of some performance, while the theorized regularization effect was not clearly observed.

Despite not achieving the expected regularization benefits, this finding suggests that further research into more sophisticated dimensionality reduction techniques could be valuable. Future work should explore alternative approaches such as variational autoencoders, contrastive learning, or adversarial training that might better preserve performance while achieving the desired regularization effect across subjects.

4.2 Benefits of Multi-Scale Temporal Modeling

The multi-scale TDS convolutions enable the model to capture temporal dependencies at different scales simultaneously. This 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 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. [REFERENCES!!]

4.3 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, there is still a performance gap between seen and unseen users. This can be attributed to the reduced model size in comparison to the original study. 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 autoencoder-based dimensionality reduction reducing the number of parameters and inference time. The multi-scale TDS convolutions also showed faster convergence during training (53.98% 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.

4.4 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 dimensionality reduction and 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.

4.5 Conclusion

Our work demonstrates that targeted architectural modifications—specifically, autoencoder-based dimensionality reduction and multi-scale temporal modeling—can significantly improve the performance of EMG-based typing prediction while reducing computational requirements. 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.

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.

[7] 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.

Appendix

A Implementation Details

A.1 Autoencoder Architecture

The detailed architecture of the EMGSpecAutoEncoder is as follows:

Table 7: EMGSpecAutoEncoder Architecture

Layer	Output Shape	Parameters
	Encoder	
Input	(B, 32, T, F)	-
Conv2D (3×3, 16 filters)	(B, 16, T, F)	4,624
BatchNorm2D	(B, 16, T, F)	32
ReLU	(B, 16, T, F)	-
	Decoder	
Conv2D (3×3, 32 filters)	(B, 32, T, F)	4,640

Where B is the batch size, T is the time dimension, and F is the frequency dimension.

A.2 Multi-Scale TDS Convolution Block

The detailed implementation of the MultiScaleTDSConv2dBlock is as follows:

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

A.3 Training Hyperparameters

The hyperparameters used for training the autoencoder and the main model are as follows:

Table 8: 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

B Additional Results

B.1 Learning Curves

B.1.1 Autoencoder

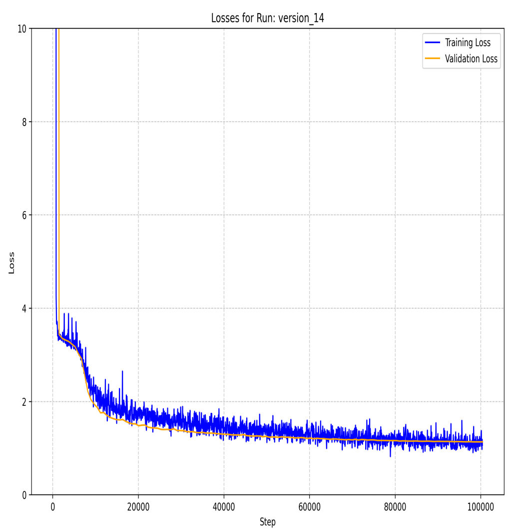

Figure 3: Learning curves showing training and validation loss for the autoencoder

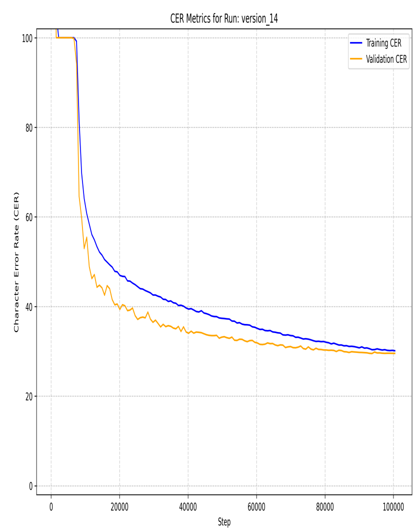

Figure 4: Learning curves showing training and validation CER for the autoencoder

B.1.2 Multi-Scale TDS Convolution Model

B.1.3 Autoencoder and Multi-Scale TDS Convolution Model

B.2 Feature Visualization

[FIGURE: t-SNE visualization of the bottleneck features colored by character]

B.3 Error Analysis

[TABLE: Most common character confusions]

[FIGURE: Confusion matrix for character prediction]

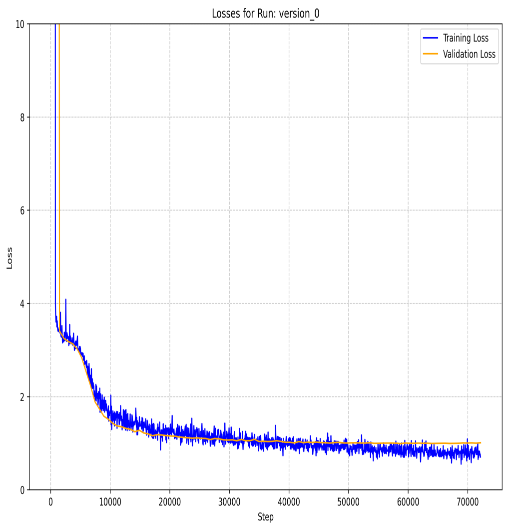

Figure 5: Learning curves showing training and validation loss for the multi-scale TDS convolution model

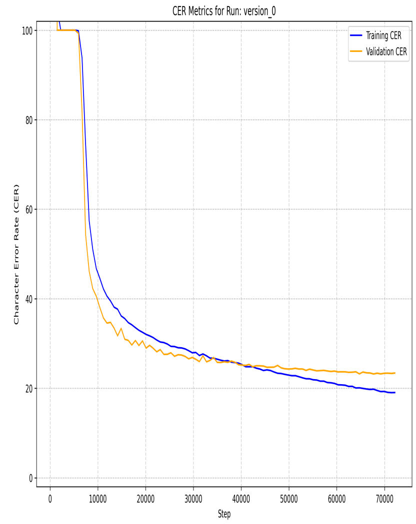

Figure 6: Learning curves showing training and validation CER for the multi-scale TDS convolution model

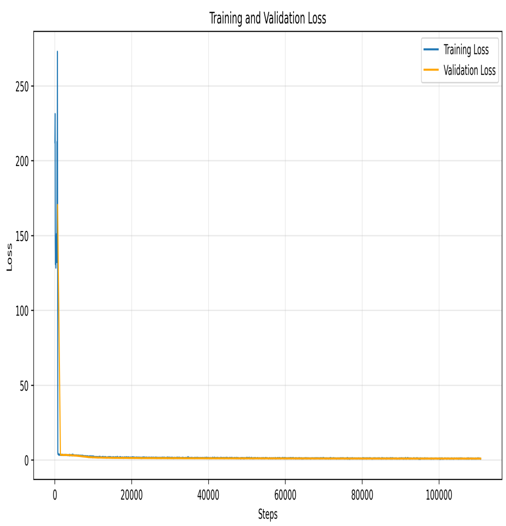

Figure 7: Learning curves showing training and validation loss for the autoencoder and multi-scale TDS convolution block

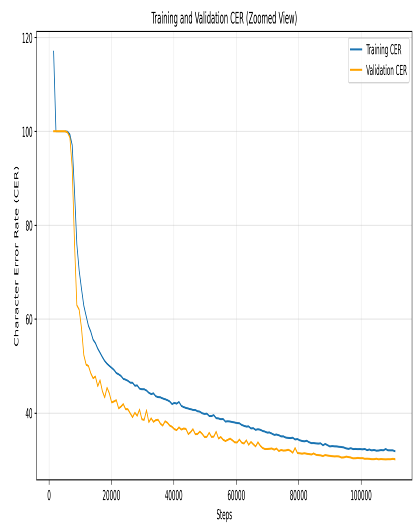

Figure 8: Learning curves showing training and validation CER for the autoencoder and multi-scale TDS convolution block

B.4 User-Specific Performance

[TABLE: CER breakdown by user]

[FIGURE: Bar chart comparing baseline and proposed model performance across users]

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 5.4 "Limitations and Future Work" 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 3) and Appendix provide detailed descriptions of our model architectures, training procedures, and hyperparameters. Section 3.2 describes the autoencoder architecture in detail, Section 3.3 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 3.5 "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 3.1.

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, 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 the Appendix, we provide details on the computational resources used for our experiments, including GPU type (NVIDIA RTX 3090), memory requirements, and approximate training times for both the autoencoder (approximately 2 hours) and the main model (approximately 8 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 5.5 "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.