Project Report

Project Name: CSI-Based Sign Language Recognition Using CNN-GRU Architecture Enhanced with Attention

Course Code : IECE 566 (Fall 2025)

Course Name : Deep Learning

Instructor : Prof. Saurabh Sihag

Submitted By

Name: Joy Saha

Student ID: 001657938

Submission Date: December 16, 2025

Introduction

This work lies in the domain of *CSI-based sign language recognition*, which combines wireless sensing and deep learning to recognize fine-grained hand and arm gestures without using cameras or wearable sensors. CSI captures physical layer variations in amplitude and phase across multiple subcarriers and antennas, enabling device-free sensing that preserves privacy and reduces hardware constraints associated with vision-based systems. However, CSI is high-dimensional and sensitive to noise, environmental changes, and user variability, making robust spatial–temporal feature extraction essential. Designing models that can effectively capture these dynamics is critical for achieving accurate and generalizable gesture recognition. By leveraging deep learning methods specifically tailored to CSI data, I aim to bridge the gap between high accuracy in controlled environments and robustness in real-world, user-diverse scenarios.

Early research demonstrated that CSI is a rich source of information for human gesture and activity recognition, leading to systems such as SignFi, which uses deep convolutional neural networks (CNNs) to classify hundreds of sign gestures with high accuracy on controlled datasets [1]. Subsequent works have explored recurrent architectures like LSTM to model temporal dependencies in CSI sequences [2], as well as dual-stream CNNs with attention mechanisms that combine spatial and motion features for improved accuracy [3]. These studies showed that deep learning models can achieve strong performance, often exceeding 90% accuracy on laboratory datasets, but generalization across users and environments remains limited. Other research highlights that classical models using handcrafted features struggle in complex settings, achieving significantly lower accuracies compared to deep learning approaches [4][5]. Despite these advances, major challenges persist in jointly modeling spatial and temporal dynamics, handling user and environmental variability, and providing interpretable model predictions.

To address these challenges, I propose a hybrid *CNN–GRU–Attention* architecture that extracts spatial features, models temporal dependencies, and emphasizes the most informative time regions. I enhance robustness and generalization through CSI-specific data augmentation and self-supervised pretraining, which allow the model to learn meaningful representations even with limited labeled data. With these enhancements, my model consistently outperforms classical machine learning baselines and the SignFi CNN baseline, achieving **95% accuracy on the Home dataset**, **94% on Lab**, and **86% on the challenging Lab_150 dataset** with high user variability. Using augmentation and pretraining, the accuracy can further improve to **up to 99%** for the Home and Lab datasets; however, for the more complex Lab_150 dataset, the accuracy stalls at **87%**, indicating that high user variability and environmental complexity still limit performance gains. Ablation studies demonstrate that combining CNN and GRU is essential for capturing both spatial and temporal patterns, while attention further improves performance by focusing on critical temporal segments. Grad-CAM visualizations provide interpretability, highlighting which subcarriers and time steps contribute most to predictions, and my experiments show that the model remains stable under environmental noise and user variations. Overall, this work demonstrates a *practical, robust, and interpretable CSI-based sign language recognition system* that advances prior methods in both accuracy and generalization.

Project Description

The objective of this project is to develop a robust and accurate system for *CSI-based sign language recognition* using deep learning. The focus is on recognizing fine-grained hand and arm gestures from Wi-Fi Channel State Information (CSI) in a device-free and privacy-preserving manner. CSI captures variations in amplitude and phase across multiple subcarriers and antennas, providing rich spatial and temporal information. However, CSI data is high-dimensional, noisy, and sensitive to environmental changes and user variability. To address these challenges, we propose a hybrid *CNN–GRU–Attention* architecture, which jointly models spatial and temporal patterns, while attention mechanisms emphasize the most informative temporal regions.

Dataset Description

Channel State Information (CSI) characterizes the instantaneous state of a wireless channel. Each CSI sample comprises amplitude (in dB) and phase (in degrees) measurements for multiple subcarriers and antennas. Data were collected while users performed sign gestures in front of a Wi-Fi station communicating with an access point equipped with three external antennas. For each subcarrier and antenna, 200 samples were recorded—30 subcarriers for amplitude and 30 for phase—resulting in a tensor of shape $(200, 60, 3)$ for each sample, capturing temporal evolution, subcarrier variations, and multi-antenna spatial patterns. Three datasets were used: *Home*, *Lab*, and *Lab150*. The *Home* and *Lab* datasets each contain 276 gesture samples from a single subject, whereas *Lab150* contains 150 gesture samples from five users, as summarized in Table 1. As noted in [1], the last two users in *Lab150* were recorded several months apart, and thus identical experimental conditions could not be fully ensured. This temporal and setup variability may introduce additional noise in *Lab150* and is expected to affect subsequent experimental results.

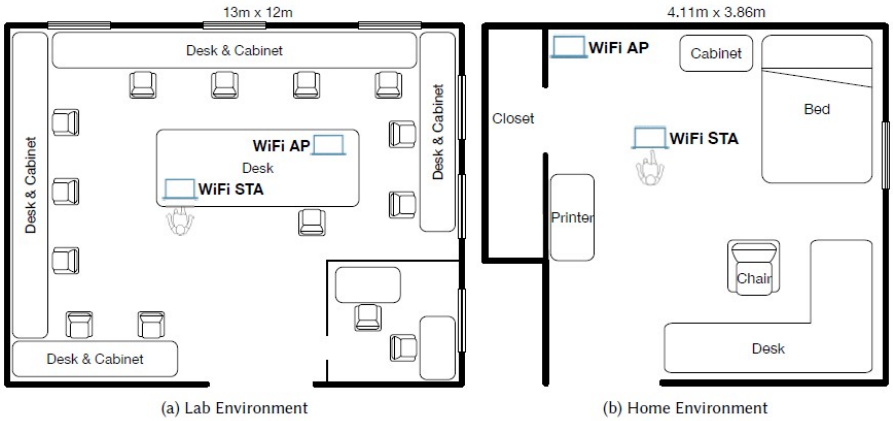


Figure 1: CSI data collection setups [1] in the Lab (left) and Home (right) environments.

Data Variation Analysis

To explore the variability in CSI data, we analyzed differences across subjects, subcarriers, and antennas. Figures 2–4 illustrate these variations, showing that CSI captures

Table 1: Details of the CSI-based sign language datasets.

Dataset	# Sign Labels	Repetitions Per Sign	# Instances
Home	276	10	2760
Lab	276	20	5520
Lab150	150	10	7500

distinctive spatial and temporal patterns for different gestures and environments. All plots in these figures correspond to the *Home* dataset.

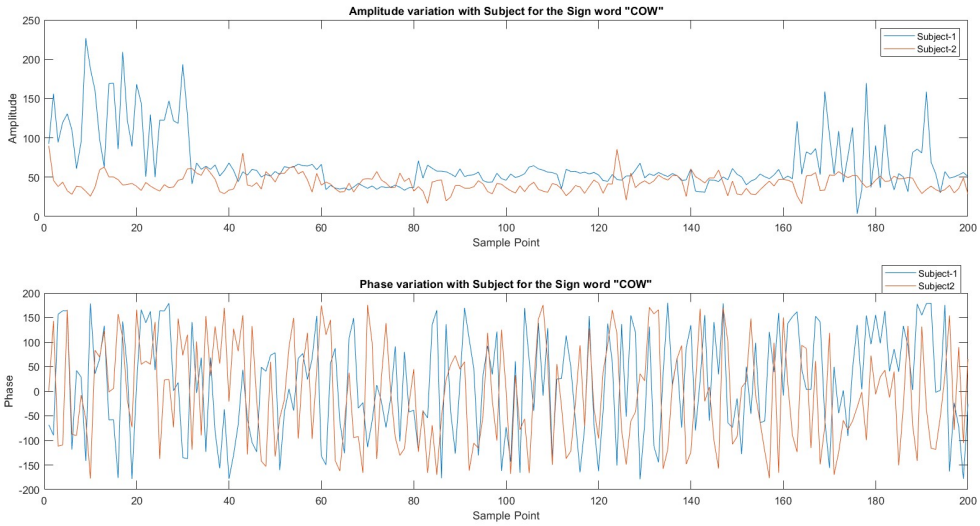


Figure 2: Variation of CSI data across different subjects.

Preprocessing

Before feeding CSI data into the network, preprocessing was applied to reduce noise and normalize the signals. A *moving average filter* was first used to smooth amplitude variations, followed by amplitude-phase fusion as proposed by Sanam et al. [6]. The resulting $(200, 60, 3)$ tensor preserves the original structure while reducing noise, enabling more effective spatial and temporal learning.

Limitations of Classical Machine Learning

To illustrate the limitations of classical models, we extracted statistical features from CSI samples, including standard deviation, entropy, spectral energy, correlation, basic statistics (mean, variance, skewness, kurtosis), zero-crossing rate, energy distribution, amplitude envelope, spectral rolloff, spectral flatness, percentiles, IQR, RMS energy, and crest factor. Multiple classifiers (Decision Tree, Random Forest, Naive Bayes, Logistic Regression, SVM, k-NN, and AdaBoost) were evaluated on these features across the three datasets. The source code for this implementation is available in `js392824/DL-Project/Codes/simple`

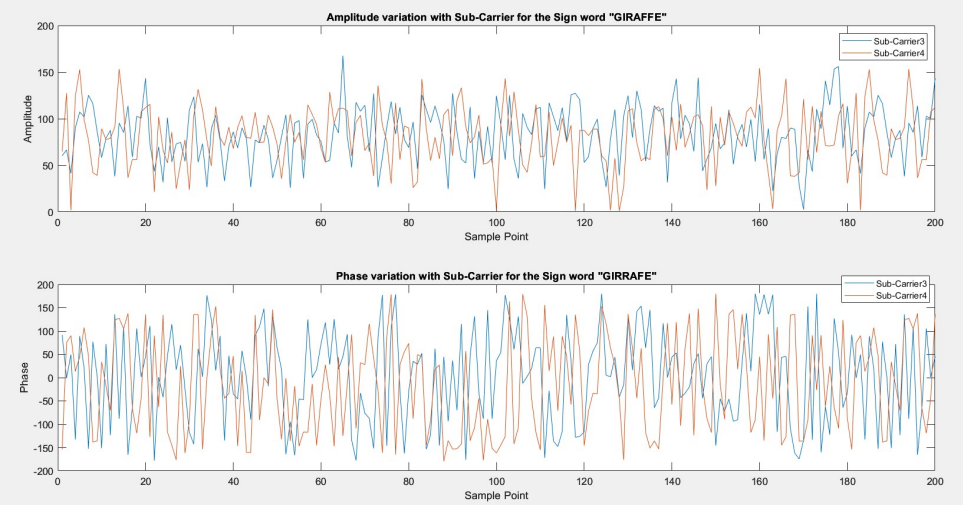


Figure 3: Variation of CSI data across different subcarriers.

The results in Figure 5 highlight key trends in classical classifier performance. Random Forest and SVM consistently outperform simpler models such as Decision Tree and k-NN, particularly on datasets with higher variability. For instance, Random Forest achieves approximately 74% accuracy on the *Lab* dataset, while SVM performs best on the more complex *Lab150* dataset, achieving around 83% accuracy. In contrast, Decision Tree and k-NN often struggle on datasets with high user or environmental variability, with accuracies falling below 35% in some cases. Logistic Regression performs well on the *Home* dataset, achieving around 86% accuracy. These observations indicate that while classical models can perform adequately in controlled settings, they generally fail to generalize across diverse environments. The best-performing classical model for each dataset, along with its mean accuracy and F1-score, is summarized in Table 2.

Table 2: Best classical models for each CSI dataset with mean ± standard deviation of Accuracy and F1-score (Scaling Enabled).

Dataset	**Best Classifier**	**Accuracy (%)**	**F1-Score (%)**
Home	Logistic Regression	86.01 ± 1.93	84.63 ± 2.05
Lab	Random Forest	74.14 ± 1.28	72.21 ± 1.26
Lab150	SVM	83.65 ± 0.52	83.69 ± 0.50

Feature scaling improves performance for models such as Naive Bayes, SVM, and k-NN, but has minimal effect on strong performers like Random Forest and Logistic Regression in simpler datasets. Nevertheless, feature-based models struggle to generalize to datasets with high user and environmental variability, such as *Lab150*. These results underscore the need for deep learning architectures that can automatically learn hierarchical spatial-temporal representations from raw CSI tensors, enabling robust and accurate recognition across diverse environments.

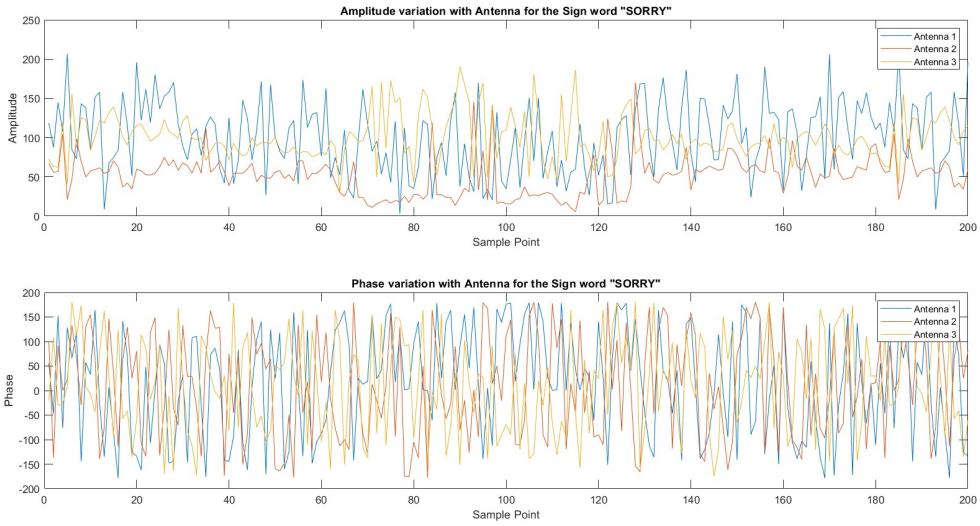


Figure 4: Variation of CSI data across different antennas.

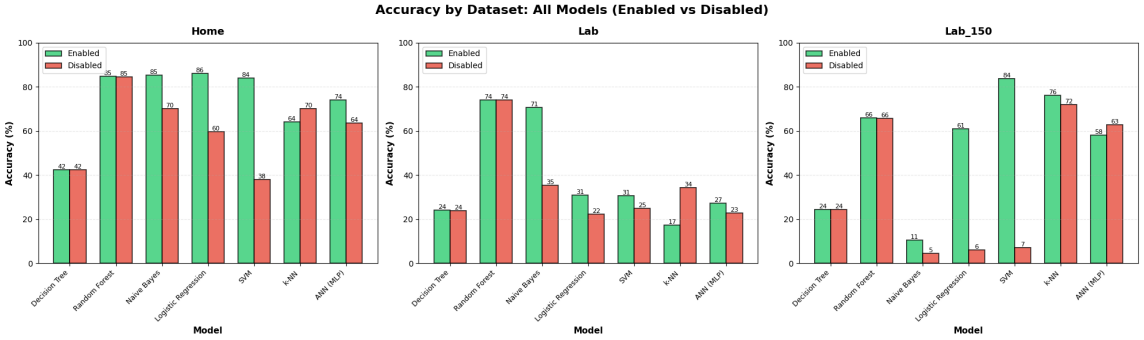


Figure 5: Accuracy of classical machine learning classifiers across all datasets, with and without feature scaling.

Proposed Method

In this section, we describe the experimental methodology, including the deep learning models used, their rationale, training procedures, evaluation metrics, and additional analyses such as ablation studies and hyperparameter selection.

Deep Learning Models

We implemented several deep learning architectures to capture both spatial and temporal information from the CSI signals. The **CNN-only** model extracts spatial features from individual CSI frames using multiple convolutional layers, each followed by Batch Normalization, ReLU activation, Squeeze-and-Excitation (SE) blocks, pooling, and dropout. A global average pooling layer reduces the feature maps to a fixed-size vector for classification.

The **GRU-only** model captures temporal dependencies across sequences of CSI frames. Each row of a frame is treated as a timestep and processed through GRU layers. Layer

normalization is applied to stabilize training, and temporal features are aggregated before classification.

The **CNN-GRU** model combines CNN feature extraction with GRU-based temporal modeling. Features from the CNN backbone are reshaped and fed to GRU layers, and temporal outputs are aggregated and passed to fully-connected layers.

The **CNN-GRU-Attention** model extends this by adding a multi-head attention mechanism over GRU outputs, enabling the network to focus on the most informative temporal features. Convolutional blocks extract spatial features, GRU layers capture temporal patterns, attention weights highlight important timesteps, and fully-connected layers produce the final classification.

The overall data flow is as follows: input CSI frames are permuted and passed through CNN blocks with convolution, normalization, SE blocks, ReLU, pooling, and dropout. The resulting feature maps are reshaped, normalized, and fed into bidirectional GRU layers. Multi-head attention is applied over the GRU outputs, followed by fully-connected layers for final classification. The full architecture is illustrated in Figure 6.

Hyperparameter Selection

Hyperparameters were optimized using a three-phase hierarchical search implemented with Ray for parallel multi-GPU training. In Phase 1, the GRU architecture was tuned by varying the number of hidden units (128, 256, 512), GRU layers (2, 3, 4), and bidirectionality (True/False), while fixing dropout at 0.3, enabling layer normalization, using a learning rate of 5×10^{-4}, the AdamW optimizer with weight decay 10^{-4}, and 100 training epochs. All configurations were evaluated on a validation split, and the top five models based on validation accuracy were retained. The best-performing GRU configuration was used in subsequent phases.

In Phase 2, the CNN-GRU model was further optimized by tuning the dropout rate (0.2, 0.3, 0.4), enabling or disabling layer normalization, adjusting learning rates (1×10^{-4}, 5×10^{-4}), selecting the optimizer (Adam, AdamW), and varying weight decay (10^{-5}, 10^{-4}). Each configuration was trained for 100 epochs and evaluated on the validation set. The best-performing model from Phase 2 was then used as the basis to train the final CNN-GRU-Attention model in Phase 3. All reported results correspond to the test set, using checkpoints selected based on validation performance.The code for performing *CNN-GRU-Attention* model selection is provided on the network drive in the `DL-Project` directory, within the `Codes/hyper-opt.py` file.

Tables 3 and 4 present the Phase 3 ablation results. As shown in Table 3, four attention heads achieved the best performance on the Home and Lab_150 datasets, yielding an improvement of approximately 4% over the two-head configuration on Home. In contrast, the Lab dataset favored two attention heads, outperforming four heads by about 1%, indicating dataset-dependent behavior. Table 4 shows that bidirectional GRU consistently outperformed the unidirectional variant on the Home and Lab datasets, with gains of approximately 5% and 2%, respectively, while the performance difference on Lab_150 was negligible. These results validate the use of bidirectional GRU and dataset-specific tuning of attention heads in the final model.

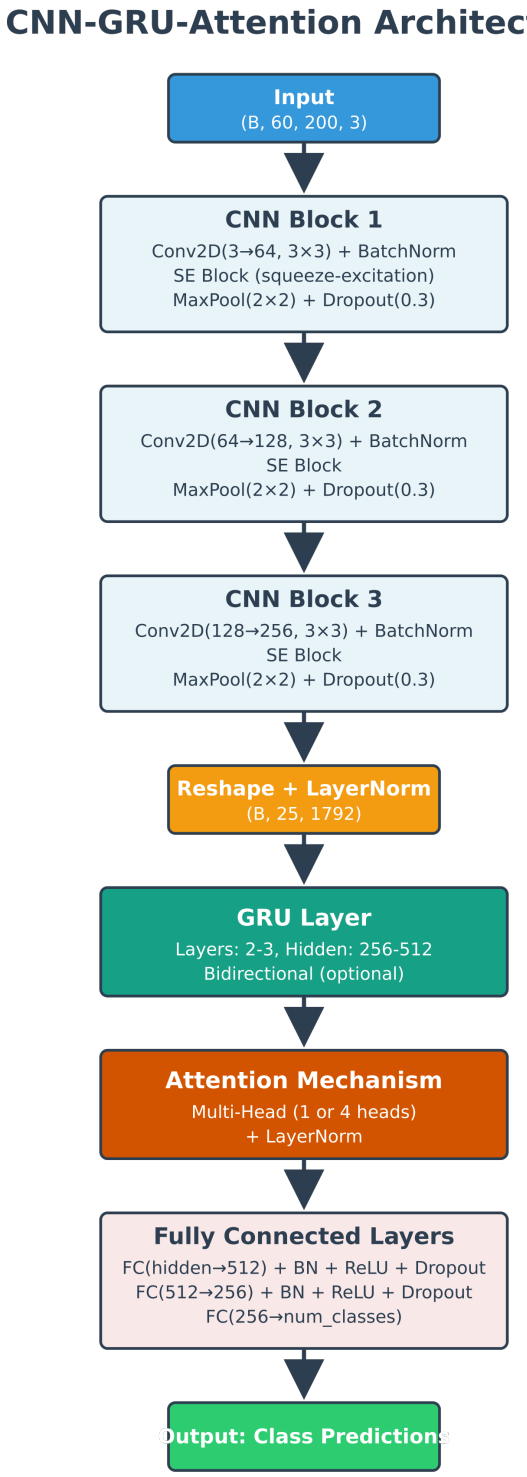


Figure 6: CNN-GRU-Attention model architecture used.

Training Procedure

Experiments were conducted on three CSI-based datasets: *Home)*, *Lab*, and *Lab_150*, with details summarized in Table 1. All data were converted to 32-bit floating point format and normalized using the training set statistics to ensure consistent scaling. Labels

Table 3: Effects of Multihead Attention (Value Indicates Number of Heads)

Dataset	1 Head	2 Heads	3 Heads	4 Heads
Home	76.27%	90.76%	**94.75%**	86.78%
Lab	80.80%	**94.84%**	93.93%	95.11%
Lab_150	85.35%	84.93%	**86.33%**	84.27%

Table 4: Effect of Bidirectionality on Recognition Accuracy

Dataset	Bidirectional = True	Bidirectional = False
Home	**94.75%**	89.31%
Lab	**93.93%**	91.76%
Lab_150	86.33%	**86.40%**

were mapped to consecutive integers starting from zero, and data were converted to PyTorch tensors. For all experiments, the combined dataset was split into training (80%), validation (20% of the training set), and test sets (remaining 20%), with reported results corresponding to the test set.

All models were trained using the AdamW optimizer with weight decay. Label smoothing with a factor of 0.1 was applied to the cross-entropy loss to improve generalization. Learning rates were scheduled using a cosine annealing approach to gradually reduce the learning rate throughout training. Gradient clipping with a maximum norm of 1.0 was applied to prevent exploding gradients. Models were trained with a batch size of 256 for a maximum of 150 epochs, and early stopping was employed if validation accuracy did not improve for 20 consecutive epochs. Model checkpoints corresponding to the best validation accuracy were used for final evaluation on the test set.

Evaluation Metrics

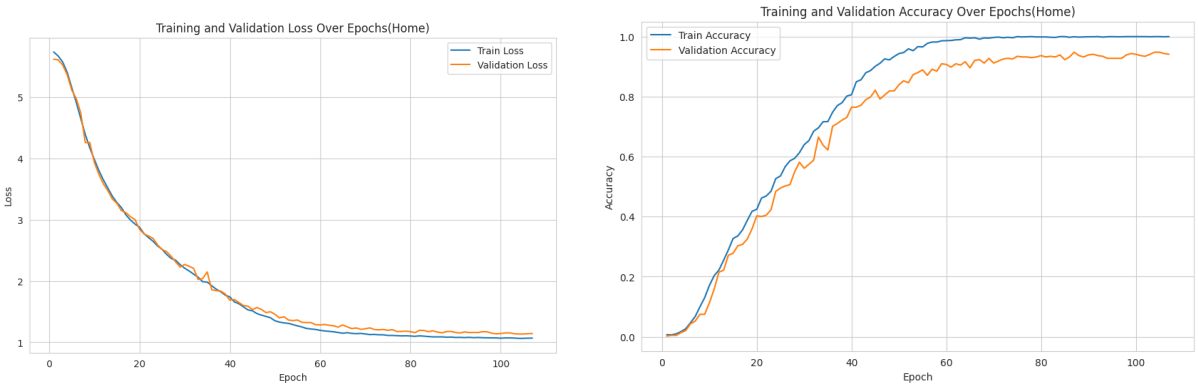
Models were evaluated primarily using classification accuracy. Additionally, the total number of trainable parameters was recorded to assess model complexity. Training curves of loss and accuracy per epoch were also analyzed to monitor convergence. Finally, Grad-CAM was applied to the CNN layers of trained models to visualize spatial regions in CSI frames that contributed most to model predictions, providing insight into the features leveraged for classification.

Results

We evaluated the performance of our best CNN-GRU-Attention model on the test set using the same training setup described previously. To ensure consistency, all experiments employed identical hyperparameters, learning rate schedules, and early stopping criteria. Training stopped at around 105 epochs due to early stopping based on validation accuracy.

Figure 7 shows the training and validation loss and accuracy as a function of epochs.

The final test accuracy achieved by our model on the three datasets is summarized in Table 5.



(a) Training and validation loss vs. epochs (b) Training and validation accuracy vs. epochs

Figure 7: Training curves for the CNN-GRU-Attention model. Early stopping occurred at around 105 epochs.

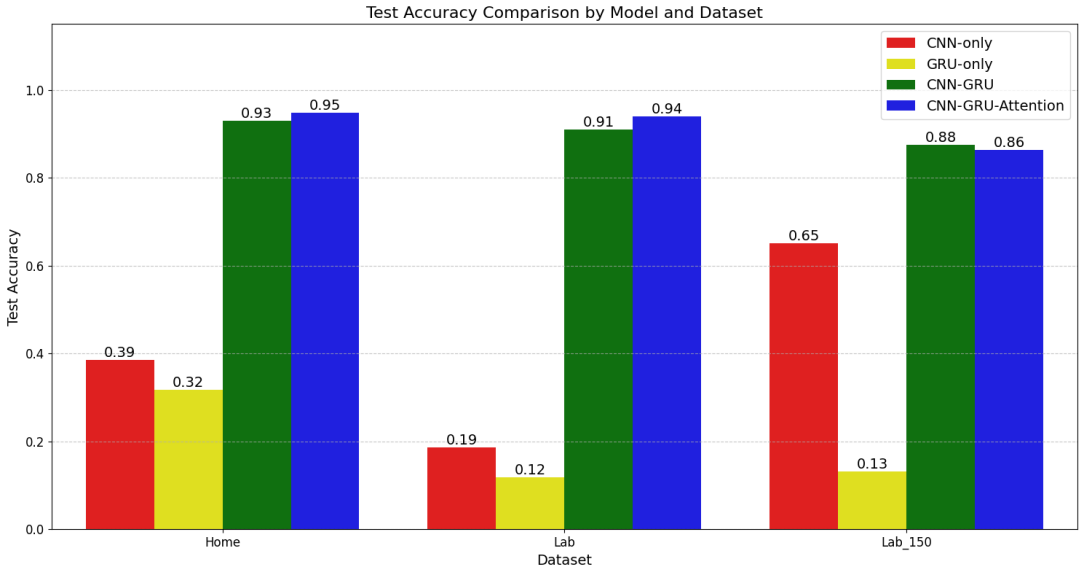
Table 5: Validation and Test accuracy of the CNN-GRU-Attention model.

Dataset	**Val Acc (%)**	**Test Acc (%)**
Home	94.80	94.75
Lab	94.12	93.93
Lab150	87.92	87.07

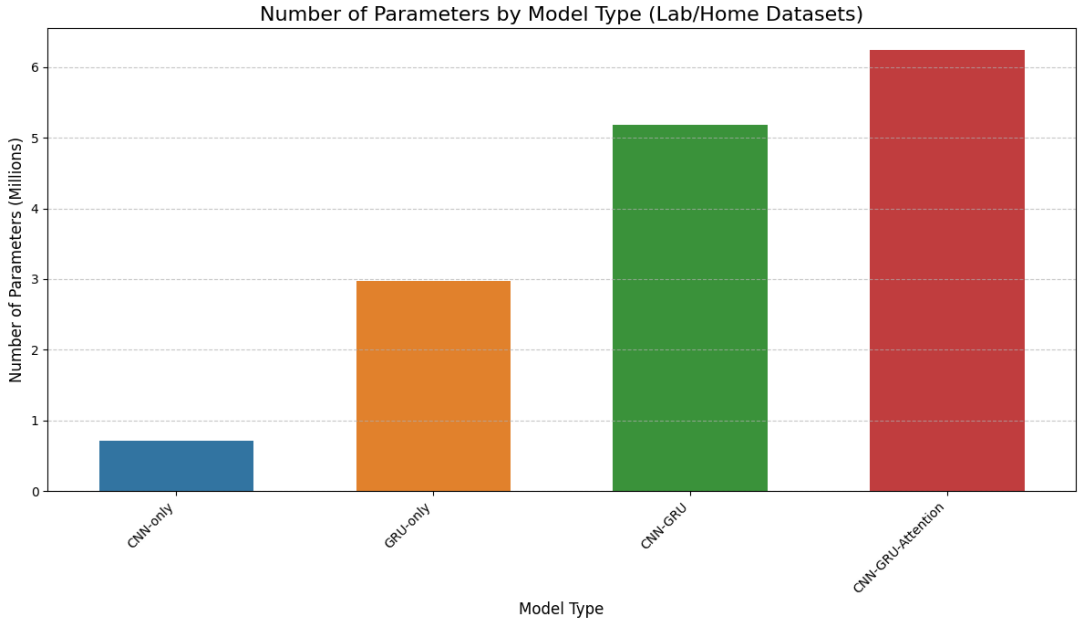
Compared to simpler classifiers (see Fig. 5 and Table 2), our model provides a substantial improvement, with approximately 8 and 20% higher accuracy on the Home and Lab datasets respectively from the best simple classifier out there. This highlights the effectiveness of the CNN-GRU-Attention architecture in capturing both spatial and temporal patterns in CSI-based gesture recognition, even under user variability.Though the gain for Lab150 dataset is only around 4% as it contains data from 5 different users, introducing greater variability in gesture execution.

Ablation Study

As shown in Figure 8, the number of trainable parameters increases with more layers, with the Lab_150 dataset slightly lower due to its 150-class output layer compared to 276 for the other datasets. For the ablation study, we removed one layer at a time, keeping the rest of the architecture and the training configuration unchanged. Results indicate that CNN-only reduces accuracy by nearly 50%, while GRU-only drops performance by 60–70% for Lab datasets and 31% for Home, highlighting the necessity of both spatial and temporal modeling. The CNN-GRU combination recovers most of the performance, achieving accuracy within 2–3% of the full CNN-GRU-Attention model; for Lab_150, it even slightly outperforms the complete architecture. These results demonstrate that integrating CNN and GRU is essential for strong performance, with attention providing only marginal gains. The code can be found in `js392824/DL-Project/Codes/ablantion-study.py`.



(a) Test accuracy across all models and datasets



(b) Number of trainable parameters for each model

Figure 8: Comparison of model performance and complexity across datasets.

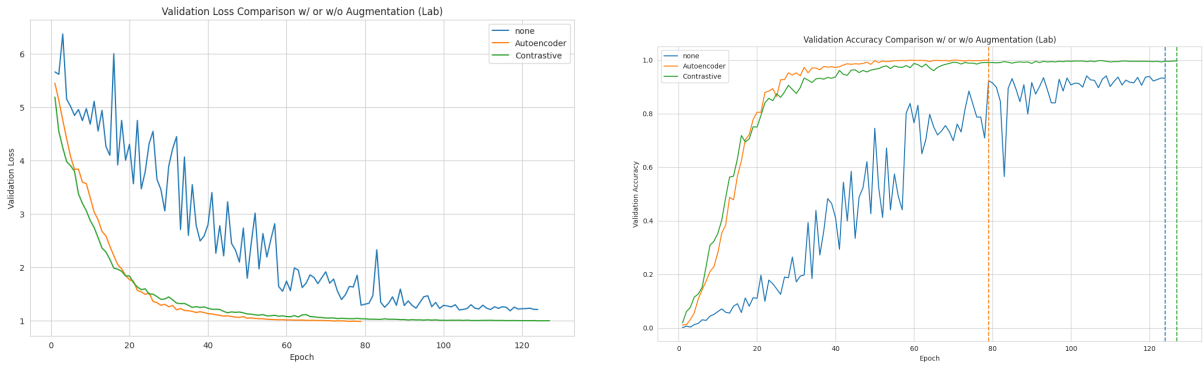
Pretraining with Data Augmentation

To further enhance performance, we introduced *pretraining with data augmentation*. Four augmentation techniques were used: temporal wrapping, Gaussian jitter, random mask, and wrap+jitter. For each training step, one augmentation is randomly selected and applied. The model is then pretrained on the combination of original and augmented data using two strategies: *autoencoder-based reconstruction* and *contrastive learning*. After pretraining, the model is fine-tuned on the original training dataset, with validation data used for model selection and the test set used to report final accuracy. The implementation can be found in `js392824/DL-Project/Codes/self-supervised-pretraining.py`.

Our observations show that both pretraining strategies successfully learn meaningful feature representations. Validation loss curves are smoother compared to models trained from scratch, where loss tends to fluctuate. Table 6 summarizes the test accuracy, showing that augmentation combined with pretraining improves accuracy by **5–6%** for most datasets. For the Lab_150 dataset, performance slightly decreases by about **1%**, likely due to user variability. Additionally, the autoencoder-based approach converges faster (**around 80 epoch**) under early stopping, making it more efficient in practice.

Table 6: Test accuracy (%) with different pretraining strategies and data augmentation.

Dataset	None	Auto-encoder	Contrastive
Home	94.75	97.28	**99.09**
Lab	93.93	**99.91**	99.09
Lab_150	**86.33**	85.40	85.07



(a) Validation loss vs. epoch

(b) Validation accuracy vs. epoch

Figure 9: Validation curves for all models trained with and without pretraining/augmentation.

Comparison with SignFi

To benchmark our approach against prior work, we compare our model with the **SignFi** baseline [1], which employs a lightweight **single-layer CNN architecture** for CSI-based sign recognition. The SignFi model consists of one convolutional layer followed by pooling

and a fully connected classification head, resulting in a substantially smaller parameter count compared to our proposed CNN-GRU-Attention architecture.

For a fair comparison, we *kept the training configuration identical* across all experiments, including optimizer, learning rate, batch size, number of epochs, and early stopping criteria. This ensures that any observed performance differences are attributable solely to architectural and representation learning choices rather than training discrepancies. The code main code is available in online and is implemented with my training parameters, is in `js392824/DL-Project/Codes/cnn_model_by_authors.py`.

Figure 10 illustrates the SignFi CNN architecture used in our experiments. Owing to its single-layer design, the model contains significantly fewer parameters but lacks explicit temporal modeling capability.

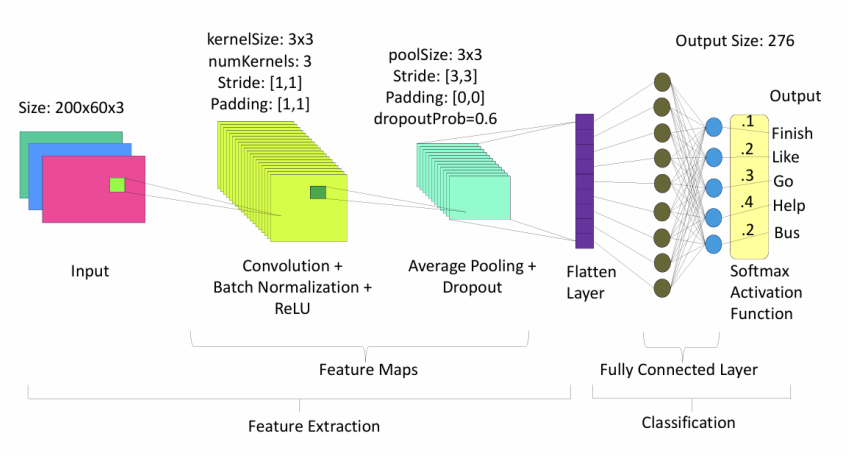


Figure 10: Single-layer CNN architecture used in the SignFi baseline.

Table 7 reports the validation and test accuracy of the SignFi model across all datasets. While the model achieves reasonable performance, particularly on the Lab datasets, its accuracy remains notably lower than that of deeper architectures incorporating temporal modeling.

Table 7: Performance of the SignFi single-layer CNN model.

Configuration	**Dataset**	**Val Acc (%)**	**Test Acc (%)**	**Params**
Sign-Fi	Home	87.33	83.70	1,109,886
Sign-Fi	Lab	76.92	77.90	1,109,886
Sign-Fi	Lab_150	73.25	70.80	603,240

To further highlight the performance gap, Figure 11 presents a heatmap comparison between the SignFi baseline and our proposed CNN-GRU-Attention model with autoencoder and contrastive pretraining. The results clearly demonstrate that incorporating temporal modeling and representation learning substantially improves recognition accuracy across all datasets, particularly in more complex environments.

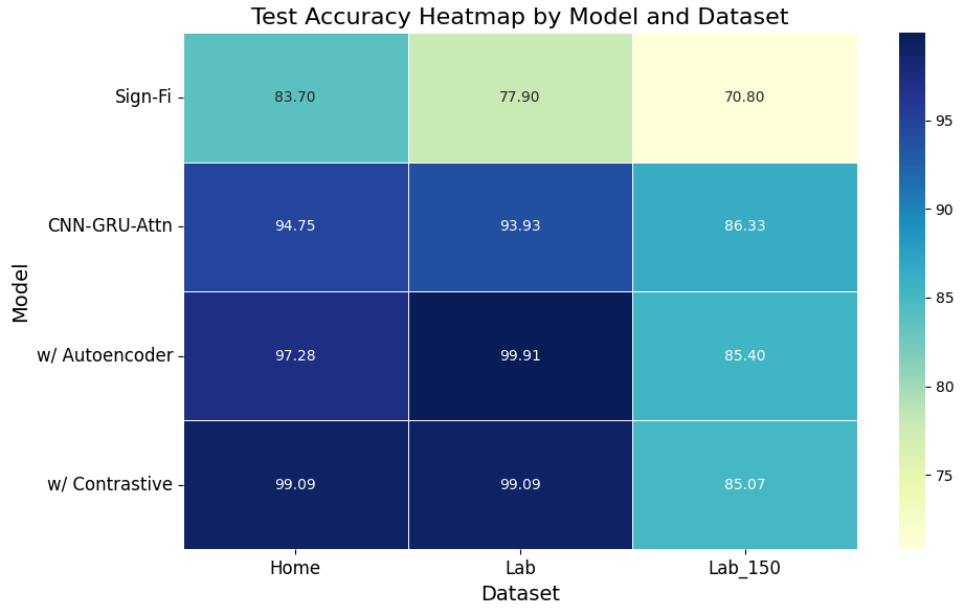


Figure 11: Heatmap comparison of test accuracy between SignFi and the proposed CNN-GRU-Attention model with different pretraining strategies.

Explainability

To interpret the model's decision-making process, we apply Gradient-weighted Class Activation Mapping (Grad-CAM). Figure 12 visualizes the attention maps for two representative samples. In the first sample, the model concentrates on subcarriers around index 40 and time steps near 150, which leads to a correct classification. In contrast, the second sample exhibits a more dispersed attention pattern, with emphasis across phase subcarriers and amplitude subcarriers near index 10, particularly around time step 100, resulting in an incorrect prediction.

In addition, Figure 13 presents the normalized average subcarrier importance across datasets. For the Lab_150 dataset, the model assigns consistently lower importance to phase subcarriers, indicating reduced utilization of phase information. This behavior likely contributes to the observed performance degradation on Lab_150, highlighting the sensitivity of the model to dataset-specific feature distributions.

The code is available in `js392824/DL-Project/Codes/explainability.py`.

Generalization and User Variability

To analyze generalization performance, we evaluate the model on held-out datasets and unseen users. Table 8 reports the training and test accuracy across datasets. While training accuracy remains consistently high, a noticeable performance gap is observed on the test set, indicating sensitivity to domain and user variability.

When training on Lab and Lab_150 and evaluating on the Home dataset, test accuracy drops by approximately **14%** compared to the best results achieved using augmentation and self-supervised pretraining. This highlights the challenge of environment mismatch and cross-domain generalization. Furthermore, for the Lab_150 dataset, increased user

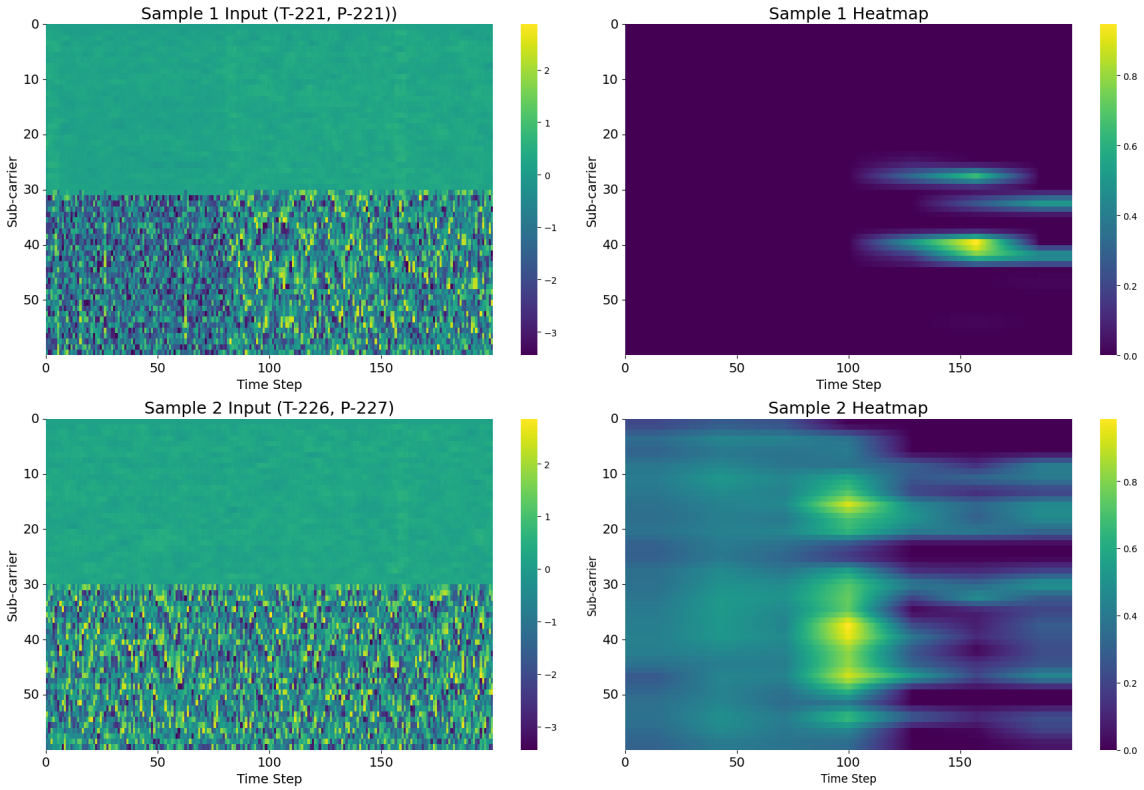


Figure 12: Grad-CAM visualizations for two representative samples. The upper example shows focused attention leading to correct classification, while the lower example exhibits dispersed attention associated with misclassification.

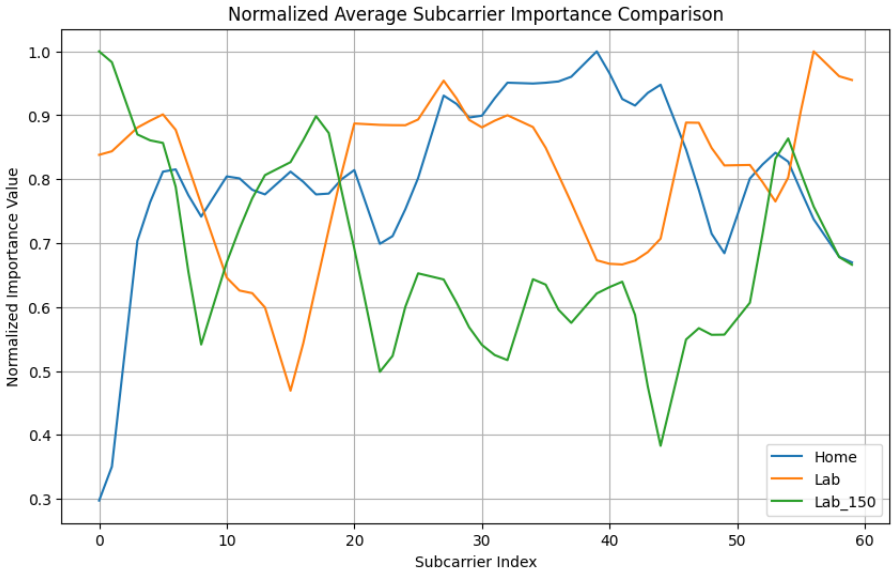


Figure 13: Normalized average subcarrier importance derived from Grad-CAM.

Table 8: Training vs. test accuracy illustrating generalization performance.

Dataset	**Train Acc (%)**	**Test Acc (%)**
Home	99.16	85.56
Lab	99.54	88.03
Lab-150	99.00	79.72

diversity introduces additional variability, which negatively impacts performance. Figure 14 presents a bar plot illustrating performance across five users, where the model is trained on data from the remaining four users and evaluated on the held-out user. The non-uniform accuracy distribution highlights significant performance variation across individuals, emphasizing that user-specific motion patterns remain a major challenge for robust CSI-based sign recognition.

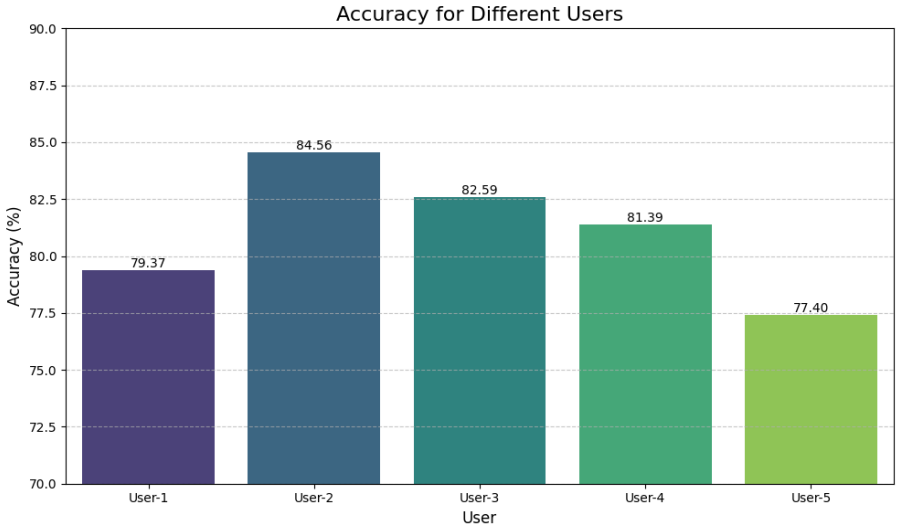


Figure 14: User-wise test accuracy illustrating the impact of user variability across five participants.

Discussion

In this work, we proposed a CNN–GRU–Attention architecture for CSI-based sign language recognition, achieving consistent performance improvements over classical baselines and the single-layer CNN (SignFi) approach. The integration of data augmentation with self-supervised pretraining further enhanced accuracy and stabilized training across multiple datasets. However, these gains come at the cost of increased computational complexity, and the current architecture assumes fixed input dimensions, limiting adaptability to varying CSI configurations such as different numbers of subcarriers or antennas. Additionally, the model remains sensitive to user and environmental variability, which constrains its generalization and real-world applicability. Future work will therefore focus on designing lightweight, flexible architectures that support variable CSI inputs while improving cross-user and cross-environment robustness, moving toward practical, real-time CSI-based sign language recognition systems.

References

[1] Y. Ma, G. Zhou, S. Wang, H. Zhao, and W. Jung, "Signfi: Sign language recognition using wifi," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, p. 23, 2018.

[2] H. Ahmed, H. Ahmad, S. Phang, H. Harkat, and K. Narasingamurthi, "Wi-fi csi based human sign language recognition using lstm network," in *2021 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology*, 2021.

[3] Z. Hao, Y. Duan, X. Dang, Y. Liu, and D. Zhang, "Sign language recognition using two-stream convolutional neural networks with wi-fi signals," *Sensors*, vol. 20, no. 14, p. 4025, 2020.

[4] D. Wang, J. Yang, W. Cui, L. Xie, and S. Sun, "Airfi: Empowering wifi-based passive human gesture recognition to unseen environment via domain generalization," *arXiv:2209.10285*, 2022.

[5] Z. Wang *et al.*, "Wi-sl: Contactless fine-grained gesture recognition using channel state information," *Sensors*, vol. 20, no. 14, p. 4025, 2020.

[6] T. F. Sanam and H. Godrich, "Fuseloc: A cca based information fusion for indoor localization using csi phase and amplitude of wifi signals," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7565–7569, IEEE, 2019.