

# SIGNWRITING FOR HANDSHAPE RECOGNITION IN SIGN LANGUAGE -SUPPLEMENTARY MATERIAL-

## 1. ABLATION STUDY

To assess the contribution of each component and technique to the overall performance of our model, we conducted a series of ablation experiments. The study explores the impact of different normalization strategies, augmentation methods, and types of learning models.

### 1.1. Impact of the learning model

The first ablation experiment analyzed the impact of the learning model used in gesture classification. Table 1 summarizes the performance of various models, including Fully Connected Networks (FC), One-dimensional Convolutional Networks (CONV1D), Random Forests (RF), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Logistic Regression (LR), Gradient Boosting Classifier (GBC) and AdaBoost. Additionally, we tested an FC with residual blocks, incorporating a pre-trained embedding from Google called "FC + embedder." This model integrates a pre-trained feature representation to improve classification.

Additionally, each model was evaluated in different normalization scenarios and with multiple augmentation techniques, such as "Finger Rotation" and "Noise." These techniques were applied to increase the model's robustness against realistic variations, such as angle changes and gesture landmarks perturbations.

### 1.2. Impact of different training datasets

In this experiment, we evaluated the impact of different training datasets on the model's performance using the best configuration identified in the previous experiment (active normalization and augmentation with "Finger Rotation" and "Noise"). The results of this experiment are presented in Table 2. Three training configurations were tested:

- **SW:** Training using the original SW dataset extracted from the official online organization.
- **SW + LSWH100:** Training combining the SW dataset with the training set of LSWH100, which supports 100 SW classes.

| No normalization and no augmentation methods                            |                        |                        |
|---|------------------------|------------------------|
| Model   | Accuracy               | F1                     |
| FC  | 0.1106 ± 0.0059        | 0.0421 ± 0.0032        |
| FC + embedder   | <b>0.6564 ± 0.0119</b> | <b>0.6212 ± 0.0131</b> |
| CONV1D  | 0.0899 ± 0.0069        | 0.0234 ± 0.0029        |
| RF  | 0.1214 ± 0.0034        | 0.0756 ± 0.0033        |
| SVM   | 0.0841 ± 0.0069        | 0.0389 ± 0.0046        |
| GBC   | 0.0766 ± 0.0058        | 0.0157 ± 0.0021        |
| Adaboost  | 0.0994 ± 0.0055        | 0.0607 ± 0.0057        |
| KNN   | 0.0946 ± 0.0049        | 0.0616 ± 0.0045        |
| LR  | 0.1118 ± 0.0033        | 0.0506 ± 0.0019        |
| With normalization and without augmentation methods                     |                        |                        |
| FC  | 0.5147 ± 0.0082        | 0.476 ± 0.0104         |
| FC + embedder   | 0.5317 ± 0.0105        | 0.5081 ± 0.0107        |
| CONV1D  | 0.1591 ± 0.0086        | 0.0566 ± 0.0052        |
| RF  | 0.6023 ± 0.0158        | 0.5736 ± 0.0169        |
| SVM   | <b>0.6312 ± 0.0084</b> | <b>0.5909 ± 0.0088</b> |
| GBC   | 0.3364 ± 0.0103        | 0.3253 ± 0.0098        |
| Adaboost  | 0.1138 ± 0.0063        | 0.098 ± 0.0066         |
| KNN   | 0.2667 ± 0.0068        | 0.2362 ± 0.0106        |
| LR  | 0.5861 ± 0.0056        | 0.5388 ± 0.0067        |
| With normalization and with "Finger Rotation" augmentation              |                        |                        |
| FC  | <b>0.7305 ± 0.0098</b> | <b>0.7254 ± 0.0106</b> |
| FC + embedder   | 0.7042 ± 0.0107        | 0.7054 ± 0.011         |
| CONV1D  | 0.5279 ± 0.0078        | 0.5126 ± 0.0075        |
| RF  | 0.6433 ± 0.0131        | 0.6296 ± 0.0137        |
| SVM   | 0.6534 ± 0.0076        | 0.6534 ± 0.0081        |
| GBC   | 0.3389 ± 0.0116        | 0.3303 ± 0.0111        |
| Adaboost  | 0.1206 ± 0.0089        | 0.0684 ± 0.0069        |
| KNN   | 0.5968 ± 0.0123        | 0.5962 ± 0.0124        |
| LR  | 0.6456 ± 0.0088        | 0.6356 ± 0.0107        |
| With normalization and with "Finger Rotation" and "Noise" augmentations |                        |                        |
| FC  | <b>0.757 ± 0.0085</b>  | <b>0.7531 ± 0.0083</b> |
| FC + embedder   | 0.6895 ± 0.0117        | 0.6883 ± 0.0127        |
| CONV1D  | 0.4893 ± 0.0095        | 0.4718 ± 0.0094        |
| RF  | 0.6263 ± 0.0122        | 0.6095 ± 0.013         |
| SVM   | 0.6579 ± 0.0082        | 0.6524 ± 0.0072        |
| GBC   | 0.3142 ± 0.0126        | 0.3126 ± 0.0128        |
| Adaboost  | 0.2131 ± 0.0125        | 0.1499 ± 0.0084        |
| KNN   | 0.6002 ± 0.0091        | 0.5995 ± 0.0085        |
| LR  | 0.6489 ± 0.0092        | 0.6391 ± 0.0082        |

**Table 1:** Comparison of the performance of different models in terms of accuracy and F1-score, considering various combinations of normalization and augmentation methods in HAGRID Dataset.

- **SW + Experiment Sample:** Training with the SW dataset complemented by a sample of 25 examples of each class from the training set corresponding to the experiment, when available. For cases without an associated training set, a sample of 25 examples is separated for training, keeping the rest for testing.

| Train Dataset     | Accuracy                              | F1                                    |
|-------------------|---------------------------------------|---------------------------------------|
| SW                | $0.7247 \pm 0.0065$                   | $0.7166 \pm 0.0058$                   |
| SW + LSWH100      | $0.8705 \pm 0.0060$                   | $0.8694 \pm 0.0060$                   |
| SW + Train Sample | <b><math>0.9549 \pm 0.0022</math></b> | <b><math>0.9549 \pm 0.0022</math></b> |

**Table 2:** Model performance results in HAGRID Dataset in three training configurations, measured by accuracy and F1-score, all accompanied by confidence intervals calculated using bootstrap.

### 1.3. Impact of augmentation factors and training sample size

To evaluate the impact of the amount of training data and the level of augmentation on the model’s performance, we conducted a study varying two main factors: the augmentation factor and the number of training samples. Table 3 presents the results obtained for different combinations of these variables, measured in terms of accuracy and F1-score.

The augmentation factor refers to the multiplication of samples generated from the original data, with values tested between 5 and 25. This factor allows the model to experience more significant variability in the data, which can improve its generalization capacity. The number of training samples refers to the original examples used before augmentation, varying from 5 to 25. The analysis allows us to observe how these variables interact to optimize the model’s performance.

### 1.4. Impact of landmark detection errors

We conducted a manual analysis to quantify the impact of Mediapipe landmark detection errors on model performance. Table 4 summarizes the results, reporting accuracy, total error count, Mediapipe-related errors, and an adjusted accuracy, computed as:

$$\text{Adjusted Accuracy} = \frac{N_{\text{total}} - N_{\text{mediapipe.error}} - N_{\text{model.error}}}{N_{\text{total}} - N_{\text{mediapipe.error}}} \quad (1)$$

This metric is not intended to replace standard accuracy measures but rather to quantify the degree to which classification errors can be attributed to landmark detection inaccuracies. It provides an estimate of the model’s performance if Mediapipe errors were mitigated, helping to isolate the impact of detection quality on recognition results.

| Augmentation Factor | Train Sample | Accuracy                              | F1                                    |
|---------------------|--------------|---------------------------------------|---------------------------------------|
| 5.0                 | 5.0          | $0.9227 \pm 0.0079$                   | $0.9231 \pm 0.0077$                   |
| 5.0                 | 10.0         | $0.9419 \pm 0.0034$                   | $0.9418 \pm 0.0035$                   |
| 5.0                 | 15.0         | $0.9462 \pm 0.0045$                   | $0.9465 \pm 0.0044$                   |
| 5.0                 | 20.0         | $0.9515 \pm 0.004$                    | $0.9517 \pm 0.004$                    |
| 5.0                 | 25.0         | $0.9507 \pm 0.0052$                   | $0.9509 \pm 0.0051$                   |
| 10.0                | 5.0          | $0.9133 \pm 0.0027$                   | $0.9130 \pm 0.0027$                   |
| 10.0                | 10.0         | $0.9356 \pm 0.0058$                   | $0.9354 \pm 0.0058$                   |
| 10.0                | 15.0         | $0.9491 \pm 0.0048$                   | $0.9492 \pm 0.0048$                   |
| 10.0                | 20.0         | $0.9485 \pm 0.0049$                   | $0.9486 \pm 0.005$                    |
| 10.0                | 25.0         | $0.9538 \pm 0.005$                    | $0.9539 \pm 0.0049$                   |
| 15.0                | 5.0          | $0.9198 \pm 0.0062$                   | $0.9200 \pm 0.0062$                   |
| 15.0                | 10.0         | $0.9390 \pm 0.0058$                   | $0.9388 \pm 0.0058$                   |
| 15.0                | 15.0         | $0.9439 \pm 0.0035$                   | $0.9441 \pm 0.0035$                   |
| 15.0                | 20.0         | $0.9457 \pm 0.0056$                   | $0.9458 \pm 0.0057$                   |
| 15.0                | 25.0         | $0.9541 \pm 0.0044$                   | $0.9543 \pm 0.0044$                   |
| 20.0                | 5.0          | $0.9209 \pm 0.0053$                   | $0.9206 \pm 0.0054$                   |
| 20.0                | 10.0         | $0.9415 \pm 0.0044$                   | $0.9416 \pm 0.0043$                   |
| 20.0                | 15.0         | $0.9486 \pm 0.0055$                   | $0.9487 \pm 0.0055$                   |
| 20.0                | 20.0         | $0.9521 \pm 0.0045$                   | $0.9522 \pm 0.0045$                   |
| 20.0                | 25.0         | $0.9482 \pm 0.0037$                   | $0.9482 \pm 0.0037$                   |
| 25.0                | 5.0          | $0.9358 \pm 0.0037$                   | $0.9359 \pm 0.0037$                   |
| 25.0                | 10.0         | $0.9335 \pm 0.005$                    | $0.9337 \pm 0.0049$                   |
| 25.0                | 15.0         | $0.9501 \pm 0.0033$                   | $0.9502 \pm 0.0033$                   |
| 25.0                | 20.0         | $0.9518 \pm 0.0032$                   | $0.952 \pm 0.0032$                    |
| 25.0                | 25.0         | <b><math>0.9554 \pm 0.0056</math></b> | <b><math>0.9555 \pm 0.0056</math></b> |

**Table 3:** Model performance in terms of accuracy and F1-score in HAGRID Dataset for different combinations of augmentation factors and number of training samples. The augmentation factor, ranging from 5 to 25, represents the multiplication of samples to increase data variability, while the number of training samples (from 5 to 25) indicates the number of original examples used before augmentation.

| Dataset          | Accuracy | Total Error | Mediapipe Error | Adjusted Accuracy |
|------------------|----------|-------------|-----------------|-------------------|
| NUS II           | 0.9871   | 26          | 12              | 0.9926            |
| OUHANDS          | 0.9818   | 11          | 4               | 0.9911            |
| ASL Digits       | 0.9902   | 15          | 14              | 0.9994            |
| Indian Alphabet  | 0.9968   | 48          | 4               | 0.9971            |
| HAGRID           | 0.9549   | 544         | 437             | 0.9907            |
| HG14             | 0.9685   | 384         | 245             | 0.9885            |
| LSA16            | 0.8439   | 45          | 23              | 0.9160            |
| Pugeault         | 0.9483   | 346         | 161             | 0.9720            |
| ArSL21L          | 0.9598   | 155         | 48              | 0.9719            |
| ASL Alphabet     | 0.9167   | 2           | 2               | 1.0000            |
| KU-BdSL          | 0.9865   | 20          | 12              | 0.9932            |
| PSL              | 0.9819   | 17          | 15              | 0.9979            |
| Bengali Alphabet | 0.7258   | 393         | 201             | 0.8482            |
| PHOENIX-14       | 0.8553   | 856         | 804             | 0.9461            |
| LSWH100          | 0.9098   | 278         | 86              | 0.9377            |

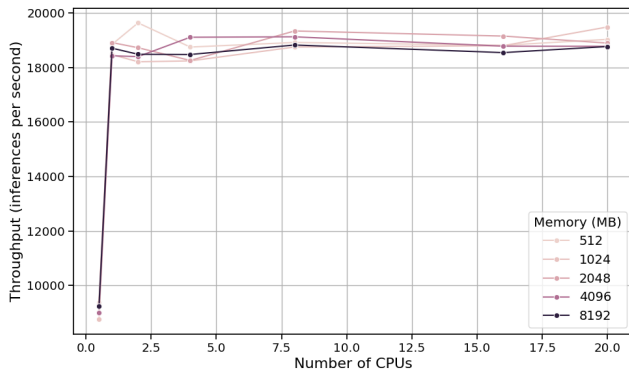
**Table 4:** Summary of model performance with adjusted accuracy for each dataset. The table includes the original accuracy, total error count, errors attributed to Mediapipe, and the recalculated adjusted accuracy, which accounts for Mediapipe-related detection errors.

While Mediapipe’s robustness enables real-time hand tracking, its errors, such as misaligned keypoints or missing detections, can significantly impact classification. To mitigate these issues, future improvements could include post-processing techniques such as outlier filtering, temporal smoothing across frames, or adaptive confidence thresholds. Additionally, incorporating alternative landmark extraction models or fine-tuning Mediapipe’s parameters could further enhance reliability.

### 1.5. Impact of resource allocation on model performance

We evaluated our model’s inference performance under different computational constraints using Docker. The model was converted to TensorFlow Lite (TFLite) format without additional optimizations, resulting in a file size of approximately 3 megabytes. It was then tested for inference time, throughput (inferences per second), and resource usage (CPU and memory). Throughput was measured as the number of inferences completed per second over batches of 1,000, repeated 30 times per configuration.

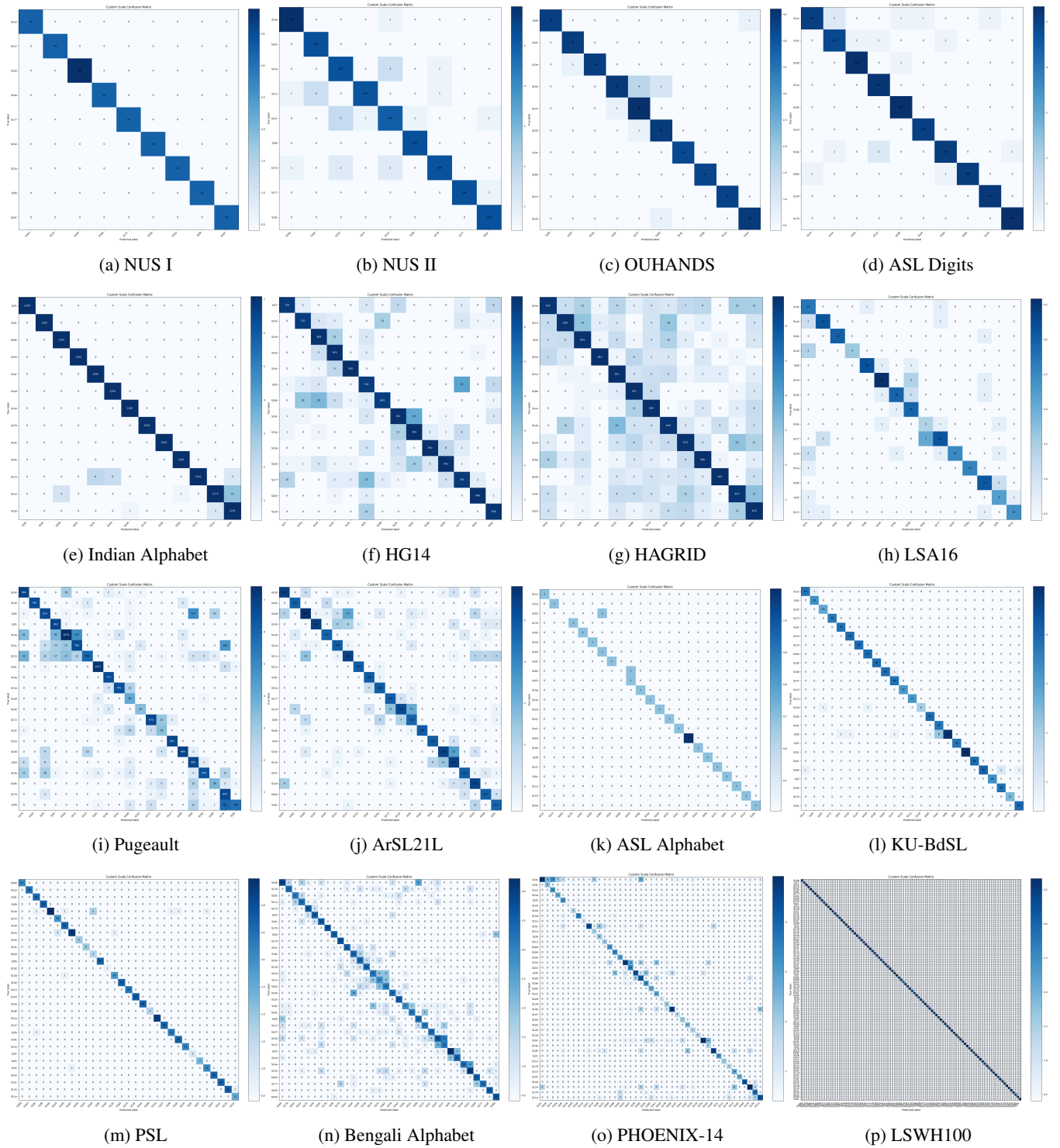
Figure 1 illustrates the relationship between the number of CPUs and throughput across various memory configurations, demonstrating the model’s scalability and stability. The results show that throughput increases with the number of allocated CPUs, stabilizing once computational bottlenecks are resolved. For reference, the fastest configuration achieved a throughput of approximately 18,500 inferences per second, corresponding to an average inference time of 0.00005 seconds per gesture. Further details on the testing procedures are provided in the supplementary materials.



**Fig. 1:** Relationship between the number of CPUs (x-axis) and throughput in inferences per second (y-axis) across various memory configurations. Each colored line represents a different memory allocation, showing how throughput changes as CPU resources increase for each memory setting.

## 2. DETAILED RESULTS

Figure 2 presents the normalized confusion matrices for the 16 experiments conducted on various gesture recognition datasets. Each matrix reflects the model’s performance in terms of classification accuracy across classes, with higher color intensity along the diagonal indicating more robust prediction accuracy. A predominance of high values along the main diagonal is observed across all datasets, suggesting a robust generalization capability of the model across diverse gesture recognition contexts.



**Fig. 2:** Normalized confusion matrices for the 16 experiments conducted on different gesture recognition datasets. Each matrix illustrates the model's performance in terms of classification accuracy between classes, with higher color intensity along the diagonal indicating better prediction accuracy.