Statistical Analysis of Sign Language Recognition with Simple CNN Models

Julia Xu, 293857[1]

[1]Supervisor: Maciej Huk, Ph.D
[2]Institute: Wroclaw University of Science and Technology - Faculty of Information and Communication Technology
[3]Lecture: Advanced Topics in Artificial Intelligence - Project

January 2026

Contents

1 Research

The purpose of this paper is to perform a statistical and effect analysis of a selected problem related to Artificial Intelligence (AI). The chosen task is to perform Sign Language Recognition (SLR) in form of image classification. Further details about the experiment(s) are explained in subsection 1.1. The implementation is published and visible on GitHub.[1]

The motivation behind this project is to analyze whether observed model performance differences occur due real and systematic given a same baseline rather than due to randomness be it from initialization or splits. Naturally, it is to proof whether the training of model variants result in significantly different outcomes when running them as independently and randomized as possible. Hence, the null hypothesis H_0 here is that there is no difference in performance among three selected models. Rejection of this hypothesis would indicate at least one model performs differently. This allows to verify whether the performance differences of three models are statistically significant.

To provide as much randomness as possible, the seed management is conducted by producing MD5-hashed values first which deal as basis for creating randomized seeds used for training and splitting purposes.

Furthermore, Demšar's paper in 2006 documents thoroughly how to cleanly perform a statistical and effect analysis by providing multiple methods to compare various models [1]. This authors recommendations are taken into consideration to construct the research plan for the statistical analysis of SLR through image classification with simple Convolutional Neural Networks (CNNs).

It is assumed that DeeperCNN or RegularizedCNN result the best accuracies while all models flag significant differences. This assumption is researched and explained in the followoing sections.

1.1 Research plan

Figure 1 visualizes the experimental pipeline. There is a chronological nature integrated from left to right. There are three model variants defined: a two-layered CNN (BaselineCNN), a three-layered CNN (DeeperCNN), and a three-layered CNN with a higher regularization factor (RegularizedCNN). All three models use dropout to introduce some kind of regularization while the latter model increases the dropout probability to 0.5 in contrast to 0.3 for the other two models. All three model architectures are visualized in sections 1.5.1, 1.5.2 and section 1.5.3. The activation function is a simple ReLU and max pooling and adapted average pooling are applied in all model variants.

Then the training pipeline is defined with a 5-5-3-5-5-principle: 5 global seeds for training initialization, 5 training epochs, 3 model variants, and 5 times repeated 5-fold CV per seed and model. This results in 25 measurements (5 repetitions for each of the 5 seeds) per variant. To conform to the mentioned 5-5-3-5-5-principle, the training is run with five epochs per fold while doing a 5-fold CV repeated five times per seed.

To evaluate the results, metrics such as accuracy, macro F1-score, and training duration are collected.

These 25 metric values per model are taken to perform the statistical analysis on. First, Friedman ANOVA is done to check whether the experiment results show any significant difference overall. Then, the significant pairs are taken and run through the Wilcoxon test to realize between which pairs does a significant difference exist. Finally, the p-values are run through Hommel for post-hoc correction.

The effect analysis follows after the statistical one to measure how big the significant difference between models is and in what directions they go. This is done through running Friedman ANOVA

[1] `https://github.com/xuju-dev/sign-language-recognition.git`

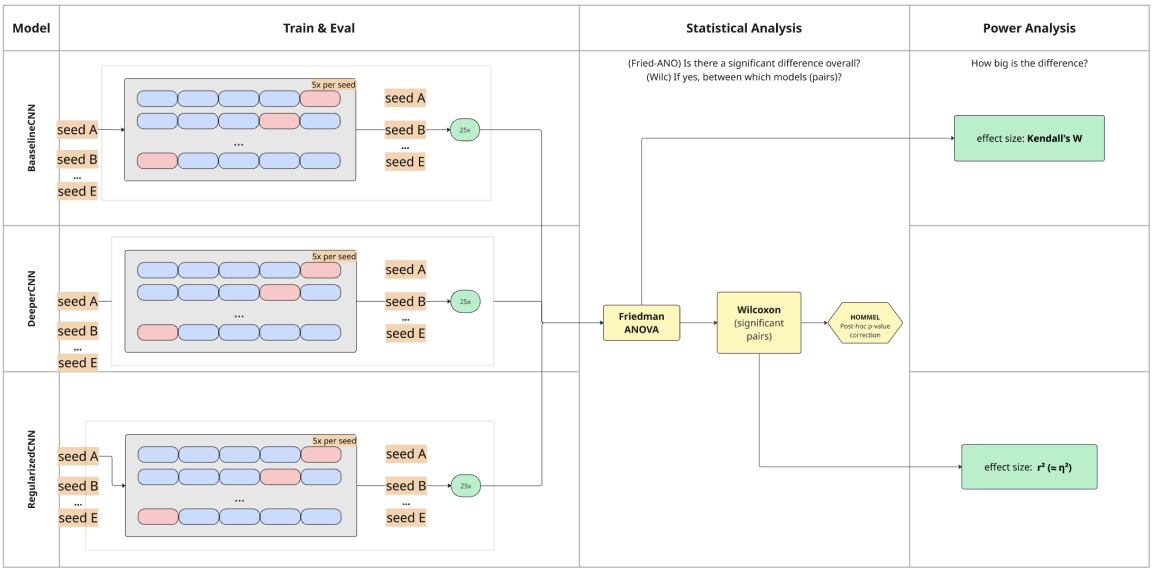


Figure 1: Flow chart of experiment pipeline. Data preprocessing is not included and assumed done already.

results through Kendall's W while the Wilcoxon test results are the inputs for calculating η^2 for effect size.

1.2 Datasets

The chosen dataset for this task of SLR contains in total 87,000 training images in 29 classes [6]. The classes are defined as the alphabetic letter with special characters such as SPACE, DELETE, and NOTHING. Each image contains a hand signature signing one specific character.

A subset of 10% of this dataset is taken to perform the experiments on. This results in 8,700 training samples. From that, the data is merged into one development set (training and validation set) with the original test set with size of 29. The images are originally in 200x200 format and pre-processed to have an image size of 224 to conform to our first draft of experimental pipeline which was with a pretrained model (MobileNet v3 [4, 10, 5]). This route was soon discarded once early training results showed that the pre-trained model was too strong for this simple task. Nonetheless, the same preprocessing pipeline can be used for the second approach with the simpler own constructed CNNs. No data augmentation is added after confirming that early training rounds of the simpler CNNS where showing accuracies low enough. This is further discussed in the later section when we evaluate and discuss the training results.

Although the original dataset contains 29 classes, all experiments were conducted using a 28-class label space due to a configuration oversight. All models were trained and evaluated under identical conditions ensuring the validity of this analysis.

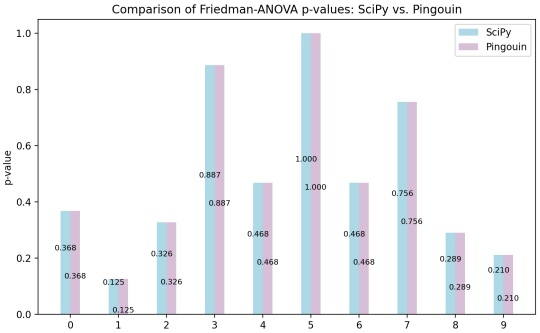
1.3 Research environment

The training is run on a Macbook Air with Apple M2 chip, 8-core CPU and 8-core GPU. This Apple hardware does not contain any CUDA supported GPU and provides its own version called

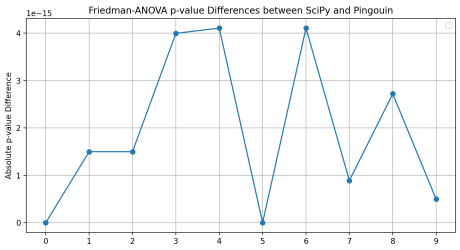
'mps' which is used for this research's experiments. General implementation is done through the programming language Python 3.11.14 with classic libraries such as but not limited to `torch`, `matplotlib`, `numpy`, `scipy`, `pingouin` and `pandas`.

1.3.1 Statistical Analysis

The statistical analysis is implemented with a Python library called `scipy`. Before deciding on using `scipy`, a comparison of libraries is conducted to see whether different implementations result in different outcomes given a same simple data input. The comparison was done between the libraries `scipy` and `pingouin`. For Friedman test three vectors are generated with 25 random values. The test is run ten times and the results are shown in Figure 2.



(a) Overview of test results. The x-axis corresponds to the numbered test repetitions. No difference when looking at three decimal places.



(b) Absolute p-value differences. The x-axis corresponds to the numbered test repetitions. Absolute differences are very small ranging from 0 to 4.108×10^{-15}.

Figure 2: Comparison of statistical test results and absolute p-value differences of ten times repeated statistical test implementation from `scipy` and `pingouin` with three example 1x25 vectors.

Figure 2a shows the overall test results. The p-value is plotted on the y-axis and the repeated test runs are numbered on the x-axis. The `scipy` implementation of Friedman-ANOVA computes a lower or p-value for 80% of the test runs where two out of ten runs resulted in identical p-values for both implementations. Figure 2a shows that the differences cannot be seen when observing the p-values up to three decimal places. The numeric differences are visualized in Figure 2b where the absolute difference is better highlighted. The highest difference in p-value was 4.108×10^{-15} for test run 6 while the lowest was 0 for test runs 0 and 5. The average absolute p-value difference is 1.932×10^{-15}. All numbers here are rounded to three decimal places.

The `statsmodels` library is used for conducting post-hoc Hommel correction.

1.3.2 Effect Analysis

For performing effect analysis on our task, manual Python implementation is used. No other established library was found such that the comparison of different implementations is omitted here. The effect sizes are calculated through Kendall's W/η^2 and r. The direction of the differences computed through estimators are omitted in this research due to time constraints.

1.4 Measures

This section gives a quick overview and definition of the used metrics in this research. There are metrics to measure model performances and formulas for statistical tests and their effect size.

1.4.1 Model Performance Metrics

Accuracy (Eq. (1)) is a general measure for how good a prediction is based on true positives (TP) and true negatives (TN) in relation to the full set of samples where FP denotes false positives, and FN false negatives [7].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{1}$$

F1 score (macro) is a multi-class based metric using precision and recall to measure F1 score for each class independently and averaging them to equally weight each class independent on their class representation in a given dataset.

$$\text{F1}_{\text{macro}} = \frac{1}{C} \sum_{c=1}^{C} \frac{2\,\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}. \tag{2}$$

The training duration is logged in minutes and records the time per fold. This time is then added up to conclude the duration for a full run of 5-fold CV (here may be also referred as 'repeat/repetition') which is taken to compare with other experiment groups.

1.4.2 Statistical Tests

The statistical analysis consists of Friedman ANOVA test and Wilcoxon signed-ranks test [8, 9]. Friedman ANOVA is used to test on more than two models. Once the Friedman ANOVA test returns a significant difference in the compared models the Wilcoxon signed-ranks test is run to compare two specific models to explore which model/group pair has a statistically significant difference. After that, Hommel is applied to conduct a post-hoc p-value correction [3].

1.4.3 Effect sizes

To determine the power of the statistical tests, the effect sizes are calculated. For Friedman ANOVA Kendall's W (Eq. (3)) is used where χ^2 is the Friedman ANOVA test statistics, n the number of conducted values per metric, and k the number of models/conditions.

The effect size W_K [2] is considered large when ≥ 0.5, medium when ≥ 0.3, and small when ≥ 0.1 [2].

$$W_K = \frac{\chi^2}{n\,(k-1)} \tag{3}$$

For Wilcoxon test the equations (4)-(6) are used. Eq. (4) extracts the number of non-zero differences, Eq. (5) computes the expected value and standard deviation of the observed Wilcoxon signed-rank statistics W. Finally, the Z-score s approximated and used to compute the effect size r (Eq. (6)) where N is the total number of paired observations.

[2]Here W_K is referred to Kendall's W to avoid confusion with Wilcoxon's test statistics W

$$n = \sum_{i=1}^{N} \mathbf{1}(x_i - y_i \neq 0) \tag{4}$$

$$\mu_W = \frac{n(n+1)}{4}, \sigma_W = \sqrt{\frac{n(n+1)(2n+1)}{24}} \tag{5}$$

$$Z = \frac{W - \mu_W}{\sigma_W}, r = \frac{|Z|}{\sqrt{N}} \tag{6}$$

The effect size r has the same interpretation thresholds as for W_K [2].

1.5 Experiment Methodology

The three experimental groups are equivalent to the model variants: BaselineCNN, DeeperCNN, RegularizedCNN. All experimental groups conduct the same training pipeline with the same preprocessed dataset. Each group gets the same five seeds. The difference in these experiments is limited to the used model variant to ensure a stable environment for the upcoming statistical analysis.

The statistical analysis contains a Friedman ANOVA test which is then followed by pairwise Wilcoxon test. The final layer provides a post-hoc p-value correction through Hommel.

Finally, the effect sizes are computed for the given statistical tests to show which model comparisons conclude a meaningful difference.

1.5.1 BaselineCNN

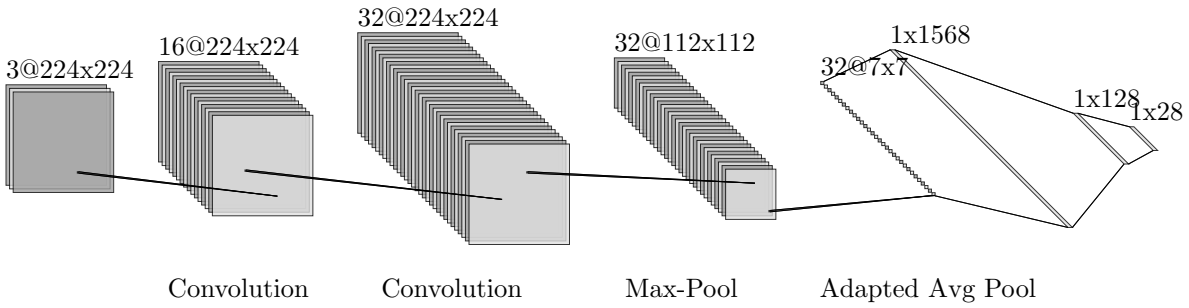


Figure 3: Model architecture of BaselineCNN illustrating size changes throughout the pipeline. Activation and dropout layers are omitted since they do not change any sizes. At the most left is the input image of size 224x224. Convolutional layers have a kernel size of 3 and padding of 1, hence not changing the spatial size. Max-Pool layers have size 2 and stride 2 and the average adapted pooling layer compresses its input into a 7x7 spatial size.

BaselineCNN consists of a 2-layered CNN with one layer of max pooling and a second resizing layer using adapted average pooling. All activations are using ReLU. All convolutional kernels have size 3 with padding of 1 and the max pooling layers half the spatial size with the last (adapted) pooling layer reducing the spatial space to 7x7. A dropout layer is applied once at the end before flattening and has the probability of 0.3.

Figure 3 shows the model architecture of BaselineCNN and illustrates how each layer reduces the spatial size.

Figure 4: Full model architecture pipeline of BaselineCNN. This model has 2 convolutional layers and 1 maximum pooling layer. Convolutions have kernel size 3 and padding 1. Max pooling layers have a filter of size 2 with stride 2 and the adapted average pooling reduces the spatial space to 7x7. Dropout probability is 0.3.

1.5.2 DeeperCNN

DeeperCNN takes the beginning of BaselineCNN and includes an additional convolutional layer with a following max pooling layer before the adapted average pooling layer ending with an adapted average pooling space of 64@7x7. The dropout probability stays the same at 0.3 and the filter sizes and padding values stay consistent with the simpler CNN model.

Figure 5: Full model architecture pipeline of DeeperCNN. This model has 3 convolutional layers and 2 maximum pooling layers. Convolutions have kernel size 3 and padding 1. Max pooling layers have a filter of size 2 with stride 2 and the adapted average pooling reduces the spatial space to 7x7. Dropout probability is 0.3.

1.5.3 RegularizedCNN

RegularizedCNN has the same architecture as DeeperCNN but with an increased dropout probability of 0.5 elevating the regularization factor in this model.

Figure 6: Full model architecture pipeline of RegularizedCNN. Convolutions have kernel size 3 and padding 1. Max pooling layers have a filter of size 2 with stride 2 and the adapted average pooling reduces the spatial space to 7x7. Dropout probability is 0.5.

2 Results

2.1 Training Performance

As shown in Figure 7, DeeperCNN results higher accuracy values and F1-scores compared to the other two models. However, DeeperCNN also takes the longest to train out of these three models. BaselineCNN shows the shortest training duration which conforms to expected outcomes since it is the smallest model out of these three. RegularizedCNN shows shorter but similar training durations

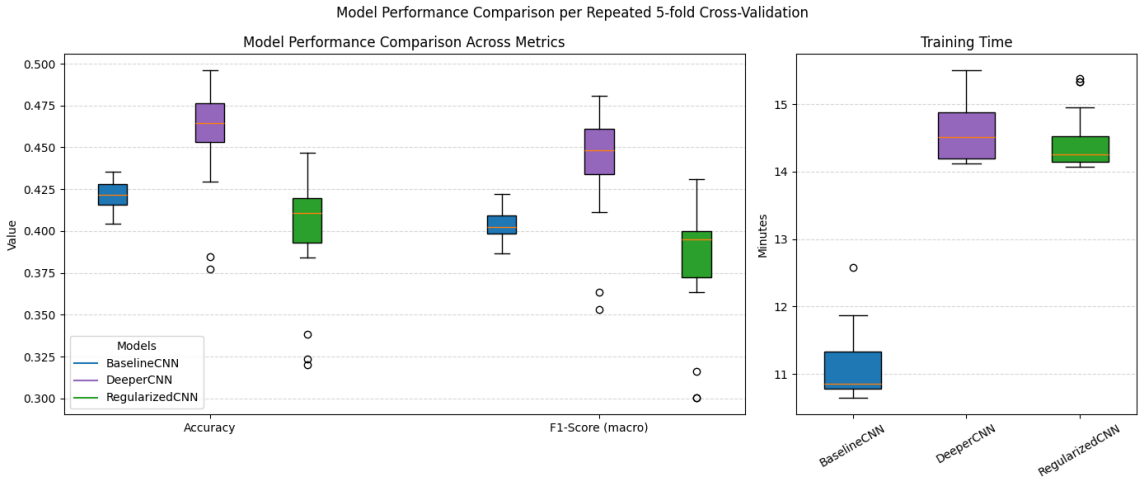


Figure 7: Overall model comparison per repeated 5-fold CV. In total there are 25 values per model (5 seeds times 5 repetitions) [Values add up to 1].

compared to DeeperCNN but the accuracy values and F1-scores underperform the base model. This shows that added regularization such as higher dropout probability does not increase a model's performance in the task of SLR with out given dataset. On the contrary, a simpler model such as BaselineCNN results in better metrics regarding correct classifications.

The model with an added convolutional layer and 0.3 dropout rate shows the best overall performance in terms of model performance metrics.

Table 1: Model performance averaged over repeated 5-fold cross-validation ($n = 25$ runs). Values are reported as mean ± standard deviation (rounded to four decimal places). [Values add up to 1 (except for time)]

Metric	**Baseline**	**Deeper**	**Regularized**
Accuracy	0.4214 ± 0.0073	$\mathbf{0.4602 \pm 0.0290}$	0.4015 ± 0.0319
F1-Score (Macro)	0.4036 ± 0.0073	$\mathbf{0.4422 \pm 0.0309}$	0.3820 ± 0.0330
Training Time (minutes)	$\mathbf{11.09 \pm 0.48}$	14.61 ± 0.48	14.44 ± 0.40

Next, the statistical analysis indicates whether these models are significantly different and if so which model pairs are significantly different and how likely so.

2.2 Statistical Analysis

2.2.1 Friedman ANOVA

The Friedman-ANOVA test shows that the models are significantly different in both metrics accuracy as well as F1-score. Both p-values are well below the set α-threshold of 5% where the metric accuracy has a lower p-value.

Table 2: Friedman ANOVA test p-values (rounded to two decimal places).

Metric	p-value	Significant ($\alpha = 0.05$)
Accuracy	6.96×10^{-8}	✓
F1-score (macro)	1.17×10^{-7}	✓

2.2.2 Wilcoxon

The Wilcoxon paired signed-rank test result show that a significant difference can be found on all model pairs given the same set α-threshold of 0.05. The highest p-value results from comparing BaselineCNN with RegularizedCNN. The post-hoc Hommel correction shows the biggest effect in the BaselineCNN-DeperCNN model pair where the correction increased the Wilcoxon p-value by 3.20×10^{-3}. Interestingly, BaselineCNN-RegularizedCNN resulted the same p-value even after Hommel. Nonetheless, these test results show that there exists a statistically significant difference between all three model pairs and confirms our initial assumption.

Table 3: (Accuracy) Wilcoxon signed-rank test with Hommel correction for pairwise comparisons of CNN models (rounded to two decimal places).

Comparison	Wilcoxon p-value	Hommel-corrected p	Significant(α=0.05)
BaselineCNN vs DeeperCNN	3.19×10^{-5}	6.39×10^{-5}	✓
BaselineCNN vs RegularizedCNN	3.78×10^{-3}	3.78×10^{-3}	✓
DeeperCNN vs RegularizedCNN	1.19×10^{-7}	3.58×10^{-7}	✓

Table 4: (F1-score) Wilcoxon signed-rank test with Hommel correction for pairwise comparisons of CNN models (rounded to two decimal places).

Comparison	Wilcoxon p-value	Hommel-corrected p	Significant(α=0.05)
BaselineCNN vs DeeperCNN	4.54×10^{-5}	9.08×10^{-5}	✓
BaselineCNN vs RegularizedCNN	1.63×10^{-3}	1.63×10^{-3}	✓
DeeperCNN vs RegularizedCNN	1.19×10^{-7}	3.58×10^{-7}	✓

2.3 Effect Analysis

The following effect sizes show how powerful the resulted differences are. Table 5 shows the effect size for Friedman ANOVA. Both accuracy and F1-score have a large effect size showing a strong and significant difference in models.

Table 6 shows the effect size for the Wilcoxon test. The largest effect size occurs when comparing DeeperCNN with RegularizedCNN while the smallest effect size conforms to the BaselineCNN-RegularizedCNN model pair. Overall, all computed effect sizes here indicate a strong difference in model pairs where the biggest difference is found in higher regularization factor with the same amount of convolutional layers. Interestingly enough, the smallest difference is found between our simplest and most tweaked model with an effect size of 0.56.

Table 5: Friedman ANOVA effect size (Kendall's W / η^2) for evaluation metrics (rounded to two decimal places).

Metric	Kendall's W (η^2)
Accuracy	0.66
F1-score	0.64

Table 6: (Accuracy) Wilcoxon signed-rank effect size (r) for pairwise CNN model comparisons (rounded to two decimal places).

Comparison	Effect size r
BaselineCNN vs DeeperCNN	0.76
BaselineCNN vs RegularizedCNN	0.56
DeeperCNN vs RegularizedCNN	0.87

Table 7: (F1-score) Wilcoxon signed-rank effect size (r) for pairwise CNN model comparisons (rounded to two decimal places).

Comparison	Effect size r
BaselineCNN vs DeeperCNN	0.75
BaselineCNN vs RegularizedCNN	0.61
DeeperCNN vs RegularizedCNN	0.87

3 Summary of research

Overall, this research shows comparing multiple models need a certain statistical pipeline. Here, three models were compared to show if a significant difference exists at all. The Friedman ANOVA test shows that there indeed exists a statistically significant difference.

Furthermore, the Wilcoxon signed rank test is applied to evaluate which model pairs conduct a significant difference and all model pair permutations achieved a p-value that confirms a statistically significant difference between them. The highest difference is found between BaselineCNN and RegularizedCNN which conforms to our initial assumption.

Finally, the effect size of both statistical tests show that the extracted difference is to be seen with large importance. While accuracy and F1-score have similar effect sizes when comparing all three models the power of the paired tests shows that DeeperCNN and RegularizedCNN have the largest effect size. This confirms our initial assumptions that all three models are statistically different. A new finding is that the power of Wilcoxon test is highest between both 3-layered CNNs and not between BaselineCNN with the other models.

To sum up, DeeperCNN performs the best in terms of simple model performance metrics and shows significant difference to all other model variants with the largest effect with RegularizedCNN.

References

[1] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[2] Fernanda Fiel Peres. Effect sizes for nonparametric tests. *Biochemia Medica*, 36(1), December 2025.

[3] G. HOMMEL. A stagewise rejective multiple test procedure based on a modified bonferroni test. *Biometrika*, 75(2):383–386, 1988.

[4] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[5] Hugging Face. Model card for mobilenetv3_small_100.lamb_in1k. `https://huggingface.co/timm/mobilenetv3_small_100.lamb_in1k`. [Accessed 21-11-2025].

[6] Khan, Anas and Nagaraj, Akash. Asl alphabet. `https://www.kaggle.com/datasets/grassknoted/asl-alphabet`. [Accessed 03-11-2025].

[7] Oona Rainio, Jarmo Teuho, and Riku Klén. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14(1), March 2024.

[8] scipy. friedmanchisquare SciPy v1.16.2 Manual. `https://docs.scipy.org/doc/scipy-1.16.2/reference/generated/scipy.stats.friedmanchisquare.html`. [Accessed 27-11-2025].

[9] scipy. wilcoxonSciPy v1.16.2 Manual. `https://docs.scipy.org/doc/scipy-1.16.2/reference/generated/scipy.stats.wilcoxon.html`. [Accessed 27-11-2025].

[10] Ross Wightman. Pytorch image models. `https://github.com/huggingface/pytorch-image-models`, 2019.