American Sign Language Understanding

Machine Learning Topics
Professor Susana Brás

Eduardo Lopes 103070
Aveiro, Portugal
eduardolplopes@ua.pt

João Neto 113482
Aveiro, Portugal
jneto04@ua.pt

Rodrigo Abreu 113626
Aveiro, Portugal
rodrigo.abreu@ua.pt

***Abstract*—Communication barriers between individuals who use American Sign Language (ASL) and those unfamiliar with it present significant challenges in daily interactions. This project aims to bridge this gap by developing a machine learning model capable of recognizing and interpreting ASL letters, facilitating smoother communication between signers and non-signers. The proposed model leverages advanced image recognition techniques, classic classification machine learning models, and deep learning with Convolutional Neural Networks to accurately identify and translate ASL fingerspelling. This research advances the field of computer vision and signal recognition by exploring effective methods for interpreting visual language. It underscores the potential of machine learning to model complex human signs with precision, contributing to broader developments in human-computer interaction.**

I. Introduction

Bridging the communication gap between deaf or hard-of-hearing individuals and non-signers has become an increasingly relevant challenge in the field of human-computer interaction. The automatic recognition of American Sign Language (ASL), particularly fingerspelling through static gestures, has been the focus of extensive research, benefiting from advances in machine learning (ML) and deep learning (DL) techniques.

This project focuses on the development of an ML-based system capable of recognizing static ASL letters, specifically the 24 alphabetic characters represented through hand gestures (excluding J and Z, which require dynamic motion).

In order to evaluate different approaches to ASL recognition, we implemented a variety of ML algorithms ranging from traditional methods to DL techniques. The models used in this study include Softmax Logistic Regression, One-vs-All Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Convolutional Neural Networks (CNNs) developed using the Keras library.

The results provide insights into the trade-offs between model complexity and classification performance and contribute to the broader field of accessible and inclusive human-computer interaction.

II. State of Art

A seminal contribution in this space was made [1], who implemented a Convolutional Neural Network (CNN) architecture augmented with multiview data generation and inference fusion strategies. Their work addressed variability in hand orientation and lighting conditions by training the model on synthetically generated perspectives of static hand gestures. This strategy significantly enhanced the model's generalization capabilities, yielding improved performance across previously unseen data. Importantly, their model was trained on depth images, which increased accuracy but introduced limitations due to the need for specialized depth sensors, thereby reducing applicability on everyday devices that rely on RGB input.

Expanding the applicability of ASL recognition to real-world environments, [2] developed a real-time recognition system leveraging neural networks and skin color segmentation. Their method isolated hand gestures from video streams using webcams, enabling gesture detection in dynamic settings without expensive equipment. By incorporating background subtraction and hand segmentation techniques, their system improved classification accuracy in noisy visual environments. However, the reliance on skin tone-based segmentation posed challenges under inconsistent lighting or with users of diverse skin tones, potentially limiting the robustness of the system in uncontrolled scenarios.

Another line of research investigates the comparative effectiveness of ML and DL approaches. [3] conducted a head-to-head evaluation between Multilayer Perceptron (MLP) and CNN models on a public ASL dataset, demonstrating the clear superiority of CNNs in both accuracy and F1-score (96.93% and 96.97%, respectively). Their findings emphasized that CNNs' layered convolutional operations are particularly suited for extracting spatial hierarchies and nuanced patterns from static images of hand gestures—capabilities that shallow networks like MLPs lack. They also highlighted how preprocessing steps such as Gaussian low-pass filtering helped improve image quality, positively affecting classification performance.

While vision-based systems dominate the field, some researchers have explored alternative modalities to capture ASL gestures. [4] introduced a recognition system using a wearable inertial motion capture device to detect and classify dynamic signs. This approach is particularly valuable for recognizing letters such as "J" and "Z," which involve motion and are therefore excluded from widely used static datasets like Sign Language MNIST. Their use of time-series data and motion trajectories enabled the accurate modeling of dynamic ges-

tures, achieving high classification rates. However, the dependency on specialized wearable hardware introduces scalability and cost concerns, and the model's real-world performance remains less validated in uncontrolled settings.

[5] conducted a broad evaluation of both classical ML and modern DL techniques, comparing the performance of Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and CNNs. Their results confirmed that CNNs consistently outperformed traditional classifiers across key metrics. The study also emphasized that architectural choices—such as the inclusion of dropout layers, batch normalization, and optimized learning rate schedules—were critical in fine-tuning CNN performance. Despite the robust experimental results, the paper lacked detailed explanations of the dataset used and the specific preprocessing pipeline, making reproducibility a challenge.

These studies underscore several converging trends and best practices in ASL letter recognition. CNNs are widely acknowledged as the most effective models for interpreting static ASL gestures due to their ability to learn spatial hierarchies and local features directly from image data. Pre-processing techniques, including multiview augmentation, segmentation, filtering, and normalization, play a central role in enhancing model robustness and accuracy.

III. Data description, visualization and statistical analysis

A. Dataset

The American Sign Language letter database of hand gestures represents a multi-class problem with 24 classes of letters (excluding J and Z, which require motion). To develop the American Sign Language Understanding project, we used the Sign Language MNIST dataset.

The dataset has two files: 1 training file and a test file. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data contains 27455 cases and the test data contains 7172 cases. Both have a header row of label pixel1,pixel2....pixel784, which represents a single 28x28 pixel image with grayscale values between 0-255. Each data image contains a hand expressing the corresponding letter sign with different angles, backgrounds, and light exposures.

B. Data Visualization

It was possible to get a reconstruction of the original image using the dataset information for each class, using the pixel values. Fig. 1, was obtained by this process, and allowed an example of each class to be visualized by the study team.

Each example was labeled according to the corresponding alphabet letter and corresponding class label after a small filtering to remove gaps in the class labels. The classes 9 (J) and 25 (Z) were removed, and the rest of the classes were adapted to remove the class label gap accordingly.

After analyzing each example, the study identified that some hand signs are similar and may be difficult to distinguish. For example, the hand signals from classes 0 (A), 4 (E), 11 (M), 12 (N), and 17 (S) are all represented by a closed hand, with slight differences that can cause problems in some machine learning models' results.

An aspect that all image examples have in common is that each of them has much redundant information because of the pixels representing the background of the image. Only the pixel information representing the hand is important. This much noise on each dataset example may compromise model performance and cause overfitting. Both this problem and the problem mentioned above must be considered during implementation, because it may be impactful in the results.

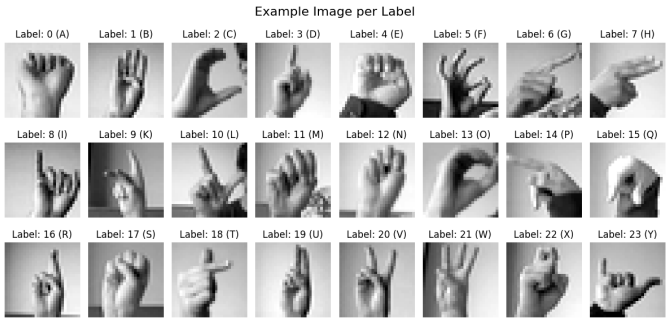

Fig. 1. Example of each class of the dataset with respective letter and class number labeled.

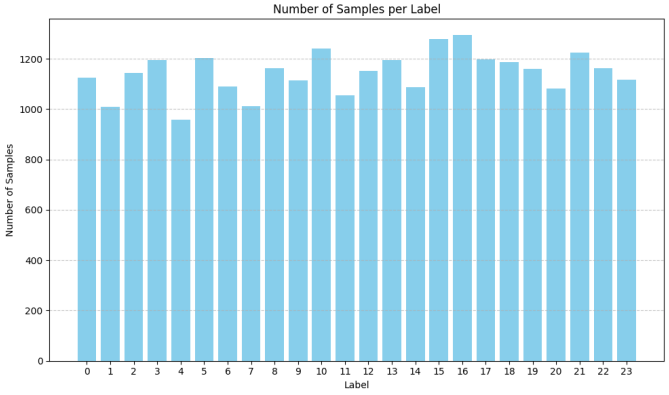

Fig. 2. Plot representing the number of examples for each class.

C. Statistical Analysis

Some aspects about the dataset are important to be addressed such as the number of training and testing examples and the balancement of these examples for each class. The number of training examples, 27455 cases, may not be optimal for model training, since the image features are each pixel value, totaling 784 features.

Even though 784 features for each example is high, the number of training examples is even higher, 27455 cases, which gives a number of examples ratio divided by the number of features ratio of 35. This ratio is higher than the

recommended 10, so this dataset may have a sufficient number of examples to prevent overfitting.

To evaluate the balancement of the class examples, a script to count the number of examples for each class was made, and the results are presented in the Fig. 2 plot. It can be concluded that, even though each class does not have the same number of samples, it looks balanced enough to exclude balancing techniques from the project implementation.

IV. Data Preprocessing

Since in this study different machine learning models will be implemented, different pre-processing approaches were also executed, but some steps were common for each.

When preprocessing the data for each algorithm, the first step was loading the train and test data in respective variables, so that for each one, the header row could be removed and example labels could be separated from the train and test examples. After this, normalization on the train and test examples was implemented by dividing each pixel value by 255, which is the value limit. This scales the data to a [0,1] range. This process is important to improve the performance and convergence speed of many machine learning algorithms. These steps are common for each algorithm implementation. Each will be aborded individually in this document, and the next preprocessing steps, which differ between them, will be explored ahead.

V. Softmax Logistic Regression

In this study, the implementation started by trying simpler algorithms like this Softmax Logistic Regression implementation and then went further to more complex ones, until model accuracy results were satisfied.

This algorithm is a generalization of the classic logistic regression for multiclass problems using the softmax function. It is suitable for linearly separable multiclass problems. Even though the dataset does not meet this characteristic, it serves as a solid starting point.

For this algorithm implementation, the training data was split with a ratio of 80% for training and 20% for training evaluation. After this, an algorithm was applied in order to tune the C hyperparameter for the Softmax Logistic Regression model. This algorithm applied L2 Regularization on the training data, which enhances generalization by penalizing large weights, improving this model implementation's results. The best value of C found was 4.6416 which made the model have an accuracy of approximately 99% on training evaluation data.

Classification Report:

	precision	recall	f1-score	support
0	0.85	1.00	0.92	331
1	0.96	0.91	0.93	432
2	0.92	0.87	0.90	310
3	0.88	0.78	0.83	245
4	0.87	0.88	0.87	498
5	0.67	0.91	0.78	247
6	0.82	0.78	0.80	348
7	0.78	0.71	0.74	436
8	0.63	0.66	0.65	288
9	0.61	0.39	0.47	331
10	0.52	0.80	0.63	209
11	0.65	0.63	0.64	394
12	0.66	0.56	0.60	291
13	0.98	0.66	0.79	246
14	0.86	0.99	0.92	347
15	0.68	0.75	0.71	164
16	0.21	0.43	0.29	144
17	0.30	0.36	0.33	246
18	0.33	0.34	0.33	248
19	0.45	0.46	0.45	266
20	0.83	0.54	0.65	346
21	0.48	0.61	0.54	206
22	0.46	0.37	0.41	267
23	0.71	0.57	0.63	332
accuracy			0.68	7172
macro avg	0.67	0.67	0.66	7172
weighted avg	0.71	0.68	0.69	7172

Fig. 3. Classification Report for Softmax Logistic Regression.

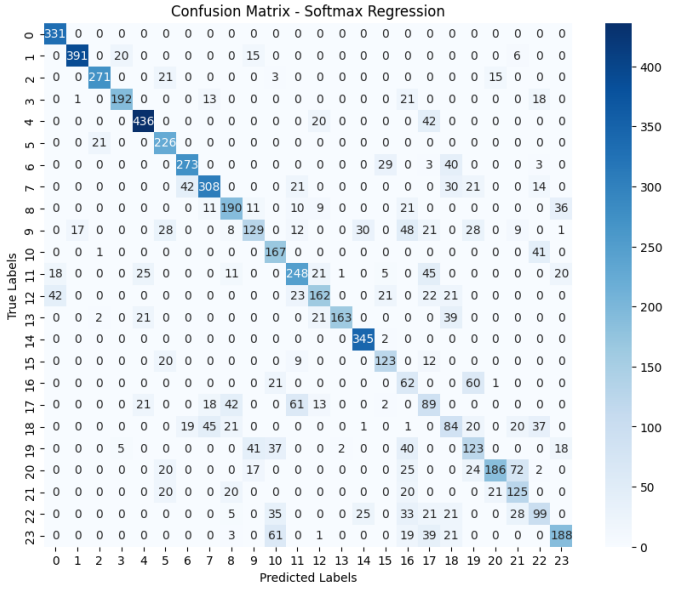

Fig. 4. Confusion Matrix for Softmax Logistic Regression application on test set.

Despite the promising results obtained during the validation phase, the performance of the Softmax Logistic Regression model significantly dropped when evaluated on the test data, with an accuracy of approximately 68%.

This performance degradation underscores some of the key

limitations of Softmax Logistic Regression for this type of task. American Sign Language (ASL) gestures, especially when captured as images or high-dimensional sensor data, often exhibit complex, nonlinear relationships that this model is not equipped to capture.

Moreover, the confusion matrix highlights substantial misclassification between visually or anatomically similar gestures (e.g., V vs. W, S vs. M, and Y vs. L), suggesting that the model struggles to distinguish between subtle spatial differences.

These challenges indicate that while Softmax Logistic Regression offers a strong and interpretable baseline, it lacks the representational power needed for high-fidelity gesture classification tasks.

VI. One-vs-All Logistic Regression

The One-vs-All (OvA) logistic regression classifier extends binary logistic regression to multiclass scenarios by training a separate classifier for each class. Each classifier treats its designated class as the positive case and all others as negative, effectively turning the multiclass problem into multiple binary problems.

Initially, logistic regression was applied directly to the raw pixel data from the Sign Language MNIST dataset. However, this approach yielded suboptimal performance, with the training, validation, and testing accuracies stagnating around 60%. This result indicated that the raw images did not provide sufficiently discriminative features for the logistic regression model.

To address this challenge, edge detection was introduced as a crucial preprocessing step. Edge detection highlights structural details of images, such as hand shapes and contours, which are more informative for classification tasks.

Specifically, the Sobel operator was used to extract edge features, which effectively emphasized the key characteristics of hand gestures while suppressing less relevant pixel variations.

With edge detection integrated into the preprocessing pipeline, the model's performance improved significantly. The training and validation accuracies increased dramatically to nearly 99%, underscoring the importance of preprocessing in feature extraction. Despite this leap, the testing accuracy reached 70%, which is lower than the training and validation results. This gap suggests a certain degree of overfitting, as the model adapts closely to the training data but does not generalize perfectly to unseen samples.

The edge-enhanced OvA logistic regression model consistently achieved high precision and recall for most gesture classes, reflecting its robustness in distinguishing key patterns. However, similar to the SVM model, certain classes exhibited lower F1-scores, particularly for ambiguous or less frequent gestures. This observation points to potential improvements, such as data augmentation or more expressive models, to better capture subtle variations across all sign language classes.

Fig. 5. Comparison of raw images and Sobel-processed images for signs A-Z. The Sobel filter emphasizes hand contours, which improves the discriminative power of the logistic regression model.

	precision	recall	f1-score	support
0	0.76	0.90	0.82	331
1	0.98	0.77	0.86	432
2	0.93	0.80	0.86	310
3	0.62	0.67	0.65	245
4	0.81	0.85	0.83	498
5	0.77	0.83	0.80	247
6	0.78	0.65	0.71	348
7	0.86	0.58	0.70	436
8	0.58	0.82	0.68	288
9	0.45	0.43	0.44	331
10	0.68	1.00	0.81	209
11	0.65	0.46	0.54	394
12	0.52	0.50	0.51	291
13	0.56	0.51	0.54	246
14	0.93	0.82	0.87	347
15	0.69	0.96	0.80	164
16	0.27	0.42	0.33	144
17	0.47	0.67	0.55	246
18	0.53	0.60	0.56	248
19	0.41	0.31	0.36	266
20	0.55	0.46	0.50	346
21	0.63	0.82	0.71	206
22	0.68	0.61	0.65	267
...				
accuracy			0.67	7172
macro avg	0.66	0.67	0.66	7172
weighted avg	0.69	0.67	0.67	7172

Fig. 6. Classification Report for OvA Logistic Regression.

In conclusion, the One-vs-All logistic regression approach, when combined with effective image preprocessing via edge detection, demonstrates strong performance on the Sign Language MNIST dataset. The edge-detected features proved essential for boosting the classifier's effectiveness, laying the groundwork for future enhancements and more sophisticated model architectures.

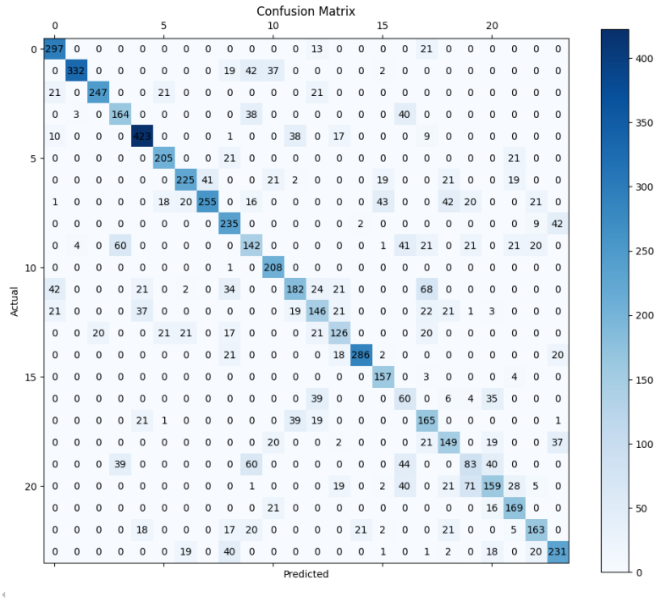

Fig. 7. Confusion Matrix for OvA Logistic Regression.

VII. K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm assigns class labels based on the majority vote of the K closest neighbors in the feature space. This method is particularly effective for image-based data, as it leverages the similarity of pixel patterns to distinguish between classes.

Prior to training,the optimal value for K was determined through cross-validation on the training data. Accuracy was evaluated across multiple K values, and the model achieved its best performance with K=1, yielding 100% accuracy on the training validation set.

The final KNN model was then trained using K=1 and evaluated on the test set, achieving an accuracy of 81%. To further enhance performance, Principal Component Analysis (PCA) was applied to reduce dimensionality by removing noise and redundant features. This approach aimed to reduce overfitting, speed up training and inference, and improve generalization to unseen data.

PCA was fitted on the training set and used to transform the training, validation, and test sets. A grid search was conducted to find the optimal combination of PCA variance retention and K value. The best performance was achieved with 95% retained variance and K=1. The corresponding classification results and confusion matrix are presented in Figures 8 and 9, respectively.

Despite the use of PCA, the overall accuracy only improved marginally by 1%, which was below expectations. Notably, the model struggled with certain classes: the letters ‘R’ and ‘U’ had particularly low precision, at 35% and 44%, respectively. The confusion matrix revealed that the letter ‘V’ was frequently misclassified (101 times), as another class ’U’. This confusion likely stems from the subtle visual difference between the letters, particularly the gap between the two raised fingers in the sign for ‘V’, a feature that may be difficult to detect reliably (see Fig. 1).

Classification Report for Best k:

	precision	recall	f1-score	support
0	0.84	1.00	0.91	331
1	0.97	0.94	0.95	432
2	0.96	1.00	0.98	310
3	0.80	0.92	0.86	245
4	0.79	0.97	0.87	498
5	0.95	0.91	0.93	247
6	0.90	0.95	0.92	348
7	0.98	0.94	0.96	436
8	0.86	0.67	0.75	288
9	0.83	0.56	0.67	331
10	0.93	0.91	0.92	209
11	0.81	0.52	0.63	394
12	0.78	0.64	0.70	291
13	1.00	0.92	0.96	246
14	1.00	1.00	1.00	347
15	0.94	1.00	0.97	164
16	0.32	0.58	0.41	144
17	0.66	0.86	0.74	246
18	0.71	0.71	0.71	248
19	0.43	0.70	0.54	266
20	0.65	0.52	0.58	346
21	0.65	0.70	0.68	206
22	0.81	0.70	0.75	267
23	0.93	0.68	0.78	332
accuracy			0.81	7172
macro avg	0.81	0.80	0.80	7172
weighted avg	0.83	0.81	0.81	7172

Fig. 8. Classification Report for KNN.

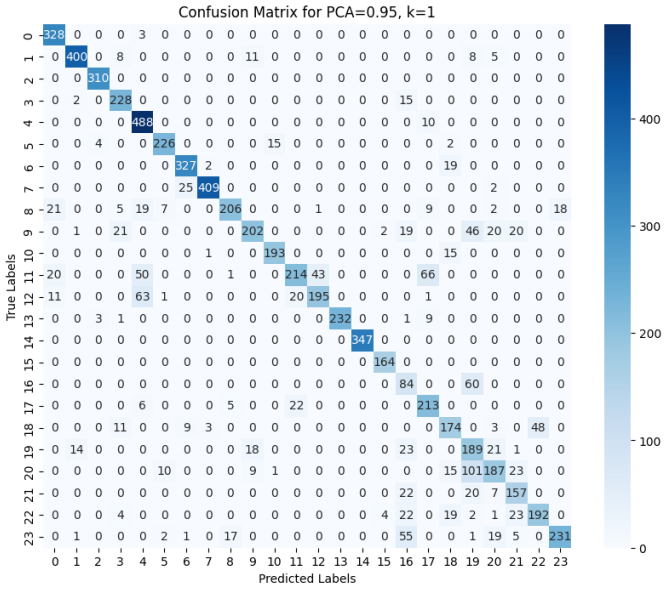

Fig. 9. Confusion Matrix for KNN.

In summary, although this implementation of KNN outper-

formed previous models, its performance remains suboptimal for deployment, particularly due to the misclassification of visually similar hand signs.

VIII. Support Vector Machine

SVM is a supervised learning algorithm known for its effectiveness in high-dimensional classification problems. It operates by identifying the optimal hyperplane that separates data points from different classes with the largest possible margin. In scenarios involving more than two classes, as is the case with the Sign Language MNIST dataset, SVM requires an extension strategy. This project employed the One-vs-All approach, where a separate binary classifier is trained for each class, distinguishing it from all others.

To identify the most effective SVM configuration, a grid search was performed over a range of values for the regularization parameter C and the kernel coefficient gamma, both of which critically influence model complexity and decision boundary flexibility. Specifically, values tested for C included 1, 10, and 100, while gamma was varied over 0.001, 0.01, 0.035, and 0.1. The best validation performance was achieved with a C value of 100 and a gamma value of 0.035, which were then used to train the final model.

	precision	recall	f1-score	support
0	0.92	1.00	0.96	331
1	1.00	0.99	1.00	432
2	0.86	0.99	0.92	310
3	0.93	1.00	0.96	245
4	0.93	0.99	0.96	498
5	0.72	0.83	0.77	247
6	0.93	0.93	0.93	348
7	0.99	0.93	0.96	436
8	0.80	0.84	0.82	288
9	0.75	0.54	0.63	331
10	0.91	1.00	0.95	209
11	0.87	0.70	0.78	394
12	0.89	0.68	0.77	291
13	1.00	0.82	0.90	246
14	1.00	1.00	1.00	347
15	1.00	1.00	1.00	164
16	0.30	0.50	0.37	144
17	0.71	0.83	0.76	246
18	0.84	0.69	0.76	248
19	0.52	0.66	0.58	266
20	0.82	0.62	0.70	346
21	0.60	0.77	0.67	206
22	0.78	0.77	0.78	267
23	0.84	0.76	0.80	332
accuracy			0.84	7172
macro avg	0.83	0.83	0.82	7172
weighted avg	0.85	0.84	0.84	7172

Fig. 10. Classification Report for SVM.

When evaluated, the trained SVM achieved perfect accuracy on both the training and validation sets (100%), while test accuracy reached 83.8%. This discrepancy suggests a certain degree of overfitting, where the model generalizes less effectively to unseen data. Despite this, the overall performance remains strong for many classes. The model's strengths were evident in its high precision and recall across most labels, particularly for well-represented gestures such as classes 0, 1, 2, 4, 14, and 15, where F1-scores often exceeded 0.90.

However, classes 19 and 21 showed relatively weak results, with F1-scores of 0.58 and 0.67, respectively. These outcomes likely reflect challenges related to ambiguous hand shapes or inherent noise in the data. Such limitations highlight the fact that while SVM is a powerful tool for structured data, it may struggle with subtler visual distinctions in image classification tasks without feature engineering or additional preprocessing.

In conclusion, the SVM model proved to be a robust baseline for classifying hand signs in the Sign Language MNIST dataset. Its success largely depended on careful tuning of hyperparameters and proper data normalization. While the results are promising, the observed weaknesses in class-specific performance indicate opportunities for future improvements through enhanced data augmentation, more expressive models such as CNNs, or ensemble approaches.

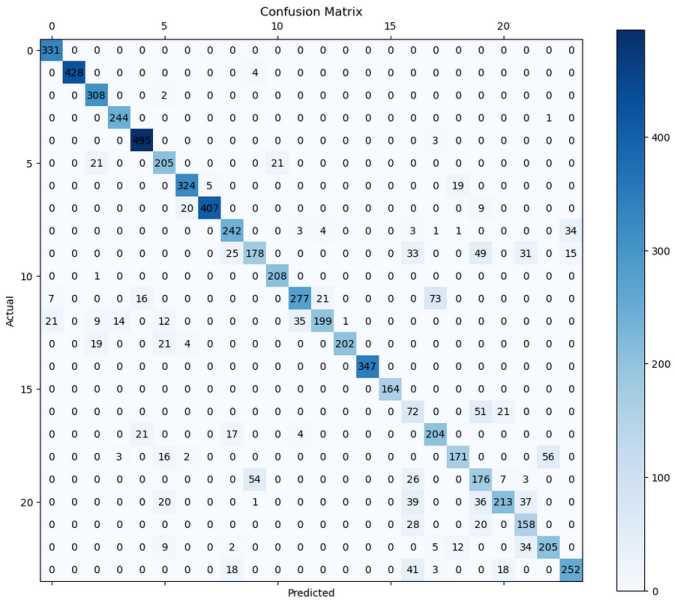

Fig. 11. Confusion Matrix for SVM.

IX. Deep Learning - Convolutional Neural Networks

Since previous implementation results were below expectations, deep learning methodologies were adopted in order to obtain better results. For this, Convolutional Neural Networks using Keras seemed like a suitable implementation for this problem. This algorithm automatically learns spatial hierarchies of features from raw images, making it ideal for visual recognition tasks.

After the pre-processing described previously, there was also the need to add a few more steps: reshape the training and test data from 1D to 3D in order to be applied in the algorithm, and artificially generate new training data, applying

small transformations in the original data, in order to avoid the overfitting problem.

It includes regularization techniques (Dropout, BatchNorm) and learning rate adaptation to improve generalization and training efficiency. The performance is visualized using accuracy and loss plots over 20 epochs.

This implementation got an impressive accuracy result of 99.83%. The corresponding confusion matrix is presented in Fig. 13. From these results, we can conclude that this model almost got a perfect accuracy score, with the only limitation being a slight confusion of letter ”G” with ”E” (12 times).

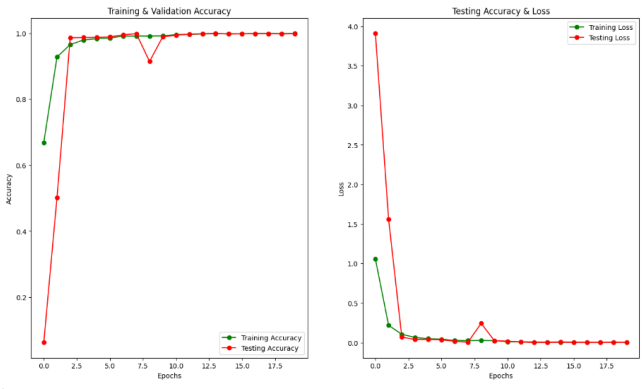

Fig. 12. Training and testing accuracy and loss curves for the CNN model.

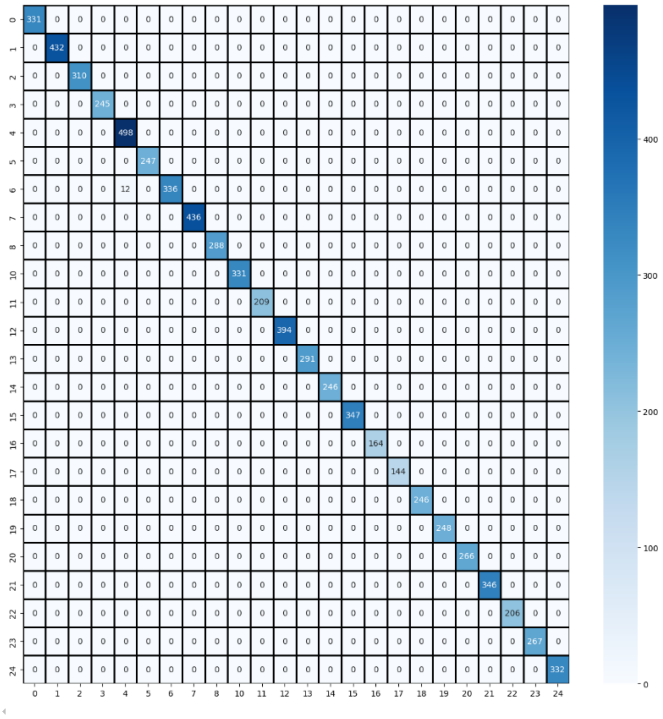

Fig. 13. Confusion Matrix for CNN.

X. Conclusion

A critical observation throughout this project is the controlled nature of the dataset: all images feature centered hands of consistent size and scale. While this consistency simplifies the learning task for the models (especially deep learning), it may also present an unrealistic view of model performance. Traditional ML models, which interpret each pixel column as an individual feature, are particularly sensitive to this issue. When a hand is slightly shifted in the image, the same gesture may produce a completely different pattern of pixel values across columns, potentially leading to misclassification.

Moreover, our analysis revealed that certain letters (such as ’M’, ’N’, ’S’, and ’E’) were more frequently misclassified. These signs share visual similarities, especially when represented as static grayscale images, making them difficult to distinguish using purely pixel-based features. This confusion was consistent across several models, highlighting the need for more discriminative features.

To enhance the effectiveness of traditional ML algorithms, more research is warranted on image preprocessing and feature extraction. As discussed in [6], feature extraction techniques such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), and edge detection (e.g., Sobel filters) can help emphasize structural characteristics of hand gestures, like finger contours and edges, that are more meaningful than raw pixel intensities. Incorporating such methods can significantly improve the generalization capability of simpler models by reducing the dimensionality and increasing the semantic relevance of features.

In conclusion, while CNNs offer state-of-the-art performance due to their capacity to learn hierarchical spatial patterns, there remains value in improving the pipeline for classic ML models through more intelligent feature engineering. Future work should explore hybrid approaches that combine traditional classifiers with image-derived features, and evaluate model robustness on less constrained datasets where hands vary in position, size, and lighting (closer to real-world usage conditions).

Acknowledgment

This paper was assisted by AI tools, including reference tracking and literature review supported by ”consensus.app” and ”elicit.com”, which provided citation analysis and relevant academic sources.

References

[1] W. Tao, M. C. Leu, and Z. Yin, “American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion,” Engineering Applications of Artificial Intelligence, vol. 76, pp. 202–213, Nov. 2018, doi: https://doi.org/10.1016/j.engappai.2018.09.006.

[2] D. Bala, Mohammad Alamgir Hossain, Mohammad Anwarul Islam, M. Mynuddin, Md. Shamim Hossain, and Md. Ibrahim Abdullah, “Effective Recognition System

of American Sign Language Alphabets using Machine Learning Classifiers, ANN and CNN," Dec. 2022, doi: https://doi.org/10.1109/iatmsi56455.2022.10119336.

[3] Mohammad Farid Naufal et al., "Analisis Perbandingan Algoritma Klasifikasi MLP dan CNN pada Dataset American Sign Language," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 5, no. 3, pp. 489–495, Jun. 2021, doi: https://doi.org/10.29207/resti.v5i3.3009.

[4] A. A. Hosain, P. S. Santhalingam, P. Pathak, H. Rangwala, and J. Kosecka, "FineHand: Learning Hand Shapes for American Sign Language Recognition," 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), pp. 700–707, Nov. 2020, doi: https://doi.org/10.1109/fg47880.2020.00062.

[5] A. A. Abdulhussein and F. A. Raheem, "Hand Gesture Recognition of Static Letters American Sign Language (ASL) Using Deep Learning," Engineering and Technology Journal, vol. 38, no. 6A, pp. 926–937, Jun. 2020, doi: https://doi.org/10.30684/etj.v38i6a.533.

[6] GeeksforGeeks, "Feature Extraction in Image Processing: Techniques and Applications," GeeksforGeeks, Jun. 10, 2024. https://www.geeksforgeeks.org/feature-extraction-in-image-processing-techniques-and-applications/