REAL-TIME SIGN LANGUAGE RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

Dilmurod R. Rakhmatov
Department of Mathematics
University of Padua
Padova, Italy
dilmurodrustamugli.rakhmatov@studenti.unipd.it

Alessandro Breccia
Department of Physics and Astronomy
University of Padua
Padova, Italy
alessandro.breccia@studenti.unipd.it

Abstract

In this article we present our final report for the project of the Cognitive Science and Computer Vision course. This project is a comprehensive study and implementation of a real-time sign language translation system, which utilizes advanced computer vision techniques. The aim is to close the gap between the hearing-impaired and hearing societies by efficiently translating sign language to text with precision and accuracy. In our research, we have designed and constructed a translator for American Sign Language (ASL) fingerspelling, anchored in a Convolutional Neural Network (CNN) framework and drawing upon the strengths, previously trained on the MNIST dataset. Our efforts have culminated in a robust model adept at accurate first-time letter classification for new users. Despite the constraints imposed by the current scope of our datasets, the promising outcomes we've observed bolster our confidence. We anticipate that with continued research and data augmentation, our endeavors will lead to the creation of a universally effective ASL letter translation tool.

Keywords: *Sign Language Translation, Computer Vision, Convolutional Neural Networks (CNN), ResNet, Real-Time Processing, Inclusive Communication, Machine Learning, Image Processing*

1. Introduction

1.1. Background and Motivation

The development of real-time sign language translation systems has become increasingly vital, with over 5% of the world's population (466 million people) being deaf or hard-of-hearing [1]. This project is motivated by the pressing need to enhance communication accessibility for this community, as emphasized in studies [2].

1.2. Scope of the Project

The project encompasses the development of a system capable of processing and translating American Sign Language (ASL) gestures into text using a CNN-based framework. It specifically targets the recognition of a comprehensive set of ASL signs, considering factors such as varying gesture speed and environmental conditions, as explored in similar research [3]. While focusing on ASL, the methodologies and technologies developed could be adapted for other sign languages, laying groundwork for future expansions in this field.

2. Literature Review

This section provides an analysis of existing research and developments in the field of sign language recognition and computer vision. Key studies on dynamic hand gesture recognition [4], and on the advancements in sign language recognition using deep learning [5], form the foundation of our understanding and approach. This section also explores the evolution of Convolutional Neural Networks (CNNs) and their application in real-time translation systems [6].

2.1. History of Sign Language Recognition

The history of sign language recognition in the realm of computer vision dates back to the early 1980s. Initial efforts were primarily focused on glove-based systems, which tracked hand movements using sensors [7]. However, the advent of more sophisticated image processing techniques and the rise of machine learning algorithms in the late 1990s shifted the focus towards vision-based systems [8]. These systems utilized cameras to capture sign language gestures, eliminating the need for wearable devices. The introduction of deep learning, especially Convolutional Neural Networks (CNNs), in the 2010s marked a significant milestone, greatly enhancing the accuracy and efficiency of sign language recognition systems [9]. Recent advancements have

been focusing on real-time translation, integrating both the linguistic complexity of sign languages and the technical challenges of computer vision [10].

2.2. Previous Works in Computer Vision for Sign Language

Recent advancements in computer vision have significantly impacted sign language recognition. Studies [4] have explored dynamic hand gesture recognition, laying the groundwork for current technologies. The integration of deep learning, work on using CNNs for sign language recognition [5], has marked a notable shift in the accuracy and efficiency of these systems. Furthermore, the application of real-time processing techniques [11], has opened new avenues for practical, real-time sign language interpretation.

3. Methodology

The real-time sign language translation system was developed through a comprehensive approach. We cover the whole pipeline starting from data collection and preprocessing to model development and system implementation. Special emphasis is placed on the use of CNNs for gesture recognition, discussing the architectural choices, training process, and optimization strategies employed. Furthermore, we illustrate our means of capturing and processing sign language gestures in real-time, to ensure a high level of accuracy and efficiency in translation.

3.1. Overview of the Proposed System

The proposed system for real-time sign language translation is built on a CNN framework, optimized for interpreting ASL. Our models architecture draws inspiration from seminal works in the field [13]. The system's foundation is the Sign Language MNIST dataset from Kaggle [15], offering a comprehensive collection of sign language images for training and validation. We employ a convolution operation, integral to CNNs for feature extraction. This is followed by a series of pooling and fully connected layers to classify each sign into corresponding alphabet letters (Fig. 1).

The Sign Language MNIST dataset, inspired by the Fashion-MNIST and Sreehari's gesture machine learning pipeline, offers a challenging benchmark for image-based machine learning methods, particularly in the realm of visual recognition. This dataset, designed in the format of the classic MNIST with 27,455 training and 7,172 test cases, represents ASL letters through 28x28 pixel grayscale images, excluding letters J and Z. It aims to not only test the capabilities of advanced techniques like CNNs but also to assist in developing practical computer vision applications for the deaf and hard-of-hearing community, recognizing ASL as a vital communication tool in North America and Europe.

Figure 1: American Sign Language Alphabet.

3.2. Data Visualization and Preprocessing

The preprocessing stage involves converting the raw dataset into image tensors suitable for CNN processing. We confirm the dataset's balance by visualizing label distribution, ensuring no class bias with nearly equal examples per class (Fig. 2).

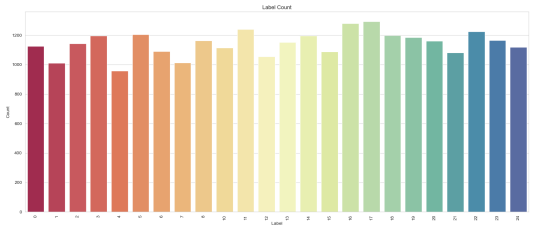


Figure 2: Demonstrating Dataset Balance for Each Training Label.

This process involves converting the image pixel values from the range $[0, 255]$ to a distribution with mean 0 and standard deviation 1. Mathematically, once obtained the mean and std for each sample, it is straightforward to perform this transformation by doing:

$$\text{Normalized Pixel Value} = \frac{\text{Original Pixel value - mean}}{\text{std}} \tag{1}$$

This normalization serves two primary purposes. First, it reduces the impact of illumination differences in the images. Illumination variance can significantly affect the performance of image processing algorithms [16]. Second, CNNs are known to converge faster on data scaled to $[0, 1]$ compared to $[0, 255]$. This is due to the effect of scaling on the gradient descent optimization process, where smaller scales lead to smoother and more stable convergence [17].

In our data normalization step, we transform the original pixel values, which range from $[0, 255]$, to a normalized range of $[0, 1]$. This normalization is achieved by (1) for-

mula. Normalization is crucial for optimizing the performance and convergence speed of our CNN model.

Following normalization, we reshape the data from a 1-Dimensional format to a 3-Dimensional format required by CNNs. The original 1-D array of $28 \times 28 = 784$ pixels is transformed into a 3-D array with dimensions $(28, 28, 1)$, where 1 represents the number of colors channels. This step is essential for the CNN to process the image data correctly:

$$\text{Reshaped Data Dimensions} = (28, 28, 1) \tag{2}$$

Figure 3 is a preview of the first 10 images after preprocessing, illustrating the results of normalization and reshaping:

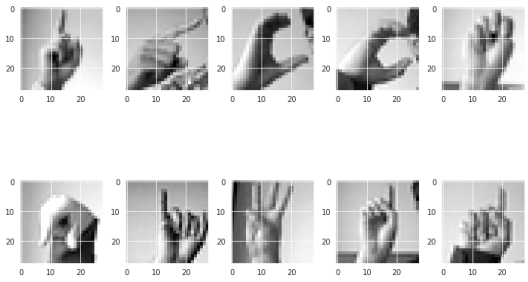


Figure 3: Preview of the First 10 Images after Preprocessing

3.3. Data Augmentation

To mitigate the risk of overfitting and enhance the robustness of our model, we employed data augmentation techniques on our training dataset. Data augmentation involves making minor modifications to the training images, thereby creating additional training data from the existing dataset. This approach maintains the label but changes the array representation of the image [18]. Popular data augmentation techniques include rotations, translations, zooming, and more.

In our approach, we implemented the following transformations:

- *Random rotation of images by 10 degrees.*
- *Random zoom on images by 10%.*
- *Random horizontal shift of images by 10% of the width.*
- *Random vertical shift of images by 10% of the height.*

Mathematically, a rotation transformation can be represented as:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3}$$

where θ is the rotation angle. Such transformations significantly increase the diversity of the training data, leading to a more versatile and robust model. Notably, we avoided vertical and horizontal flips due to the potential risk of misclassification in our specific context.

In another way, to cope with different photometric settings, such as different illumination or contrast, we tried to augment the data by varying also this aspect. But since we have not found any pre implemented method in both Keras and PyTorch, we raised to the power of P each augmented sample, where P is random number taken from the range [1, 1.5]. This results in enhancing contrast and give more variance to the training set.

4. Experiments

The primary objective of this experiment is to compare the performance of two deep learning models, Convolutional Neural Networks (CNN) and Residual Networks (ResNet), in the context of sign recognition. We aim to evaluate the models based on their accuracy, efficiency, and scalability of the dataset.

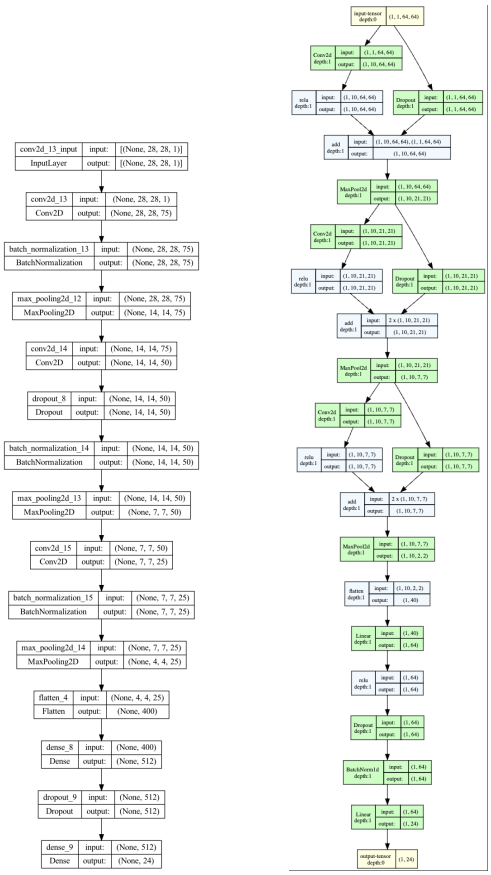
4.1. Convolutional Neural Networks (CNN)

In this phase, we focus on the development of CNN, a specialized form of Deep Learning that processes image data through a mathematical operation known as convolution. Convolution involves multiplying two matrices to produce a third, smaller matrix, effectively capturing the features of the input image [19]. The CNN employs a filter or kernel to scan over the input image, performing convolution operations to generate feature maps.

Our approach to building the CNN model involved several layers, including convolutional layers (conv2d), batch normalization, dropout for regularization, followed by flatten and dense layers. This architectural design aligns with established practices in CNN model construction [19]. The process of model development and optimization was a hyperparameter tuning tool, which allowed for the exploration of various hyperparameter configurations to identify the most effective model [20].

The Keras Tuner explored different combinations of hyperparameters, selecting the best performing model based on validation accuracy.

The architecture of the CNN developed for image classification is depicted as follows (Fig. 4 (a)). The model begins with an input layer accommodating images of size 28×28 pixels with a single color channel. It progresses through a series of convolutional layers with ReLU activation and batch normalization, each followed by max pooling layers to reduce spatial dimensions. To mitigate overfitting, dropout layers are interspersed after convolutional layers. After flattening the final feature maps, the network transitions to a dense layer with 288 neurons, followed by another dropout layer, and culminates in a final dense layer with 24 neurons corresponding to the classification categories.



(a) CNN (b) ResNet

Figure 4: Models Architecture

4.2. Residual Networks (ResNet)

Inspired by milestones in the history of CNNs, we developed a model similar to ResNet[22]: to avoid exploding or vanishing gradient issues, residual connection are added through the model architecture, right after each convolutional layer, enabling the gradient to flow fluently. The architecture follows (Fig. 4 (b)): from a 28x28 input layer we feed our samples to a series a of 3 2-D filters, of size (3, 3), with 10 different kernels each. To regularise the model, as aforementioned, residual connections have been added, together with dropout and maxpooling. The output of the last ConvLayer, after maxpooling, is then flattened and a linear layer of size 64 has been added, followed by a batch normalisation layer. The output layer is a 24 size linear layer. The only activation function used is Relu, since we want to avoid negative contributions that could arise from using different activation functions.

5. Results and Analysis

Once we had trained our two different models, we proceeded to evaluate them. Subsequently, we mainly considered the models' accuracy, or the number of correct predictions, together with precision, recall, and F1-scores extracted . A summary of the accuracy obtained values is available in Table 1.

Model	Epochs	Batch size	Accuracy
CNN	20	128	99.7 %
ResNet	3	32	97.0 %

Table 1: Results from the different experiments with models

Post-training analysis reveals significant insights into the model's performance. The Training & Validation Accuracy graph shows a rapid convergence, with the training accuracy quickly reaching a plateau, indicative of the model's capacity to learn efficiently from the data (Fig. 5 & Fig. 6.). The Testing Accuracy & Loss graph displays a stark reduction in loss, signifying a high degree of model generalization. Both graphs indicate that the model did not overfit, as the validation accuracy remains high throughout the training epochs.

From the classification report in Table 2 we can confirm that the total accuracy of the CNN model, 99%, is also reflected in the F1-scores, the harmonic mean between precision and recall, of all 24 classes.

	Precision	Recall	F1-score	Support
A	1.00	1.00	0.99	331
B	1.00	1.00	1.00	432
C	1.00	1.00	1.00	310
D	0.98	1.00	0.99	245
E	0.99	1.00	1.00	498
F	1.00	1.00	1.00	247
G	1.00	1.00	1.00	348
H	1.00	1.00	1.00	436
I	1.00	1.00	1.00	288
K	1.00	0.99	1.00	331
L	1.00	1.00	1.00	209
M	0.99	1.00	1.00	394
N	1.00	0.99	1.00	291
O	1.00	1.00	1.00	246
P	1.00	1.00	0.99	347
Q	1.00	1.00	1.00	164
R	1.00	1.00	1.00	144
S	0.98	1.00	0.99	246
T	1.00	1.00	1.00	248
U	0.99	0.97	0.98	266
V	1.00	1.00	1.00	346
W	1.00	1.00	1.00	206
X	1.00	1.00	1.00	267
Y	1.00	0.98	0.99	332
Accuracy			0.99	7172
Macro avg	0.99	0.99	0.99	7172
Weighted avg	0.99	0.99	0.99	7172

Table 2: Classification report (CNN)

ResNet model, despite the much smaller size, it has been able to obtain results comparable with the previous one. The loss shows a good training procedure, with almost no overfitting. The results of the classification are shown in the confusion matrix and in the report, both showing how the majority of signs are robustly detected and that similar signs

Class	Precision	Recall	F1-Score	Support
A	1.00	1.00	1.00	331
B	1.00	1.00	1.00	432
C	1.00	0.99	1.00	310
D	0.97	0.91	0.93	245
E	1.00	0.96	0.98	498
F	1.00	1.00	1.00	247
G	0.86	1.00	0.92	348
H	1.00	0.87	0.93	436
I	0.94	1.00	0.97	288
K	1.00	0.95	0.97	331
L	1.00	1.00	1.00	209
M	1.00	1.00	1.00	394
N	1.00	0.98	0.99	291
O	0.97	1.00	0.99	246
P	1.00	1.00	1.00	347
Q	0.99	1.00	0.99	164
R	0.80	0.99	0.88	144
S	0.93	0.99	0.96	246
T	1.00	0.92	0.95	248
U	0.98	1.00	0.99	266
V	0.94	0.93	0.94	346
W	1.00	0.99	0.99	206
X	0.87	1.00	0.93	267
Y	1.00	0.89	0.94	332
Accuracy = 0.97 Macro Avg = 0.97 Weighted Avg = 0.97				

Table 3: Classification report (ResNet)

are occasionally misinterpreted by the model (for example 'G' and 'H', or 'X' and 'D').

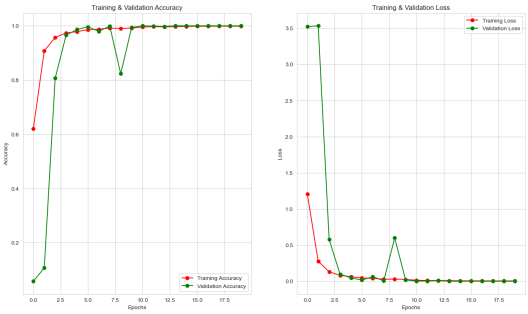


Figure 5: Training & Validation Accuracy and Training & Validation Loss (CNN)

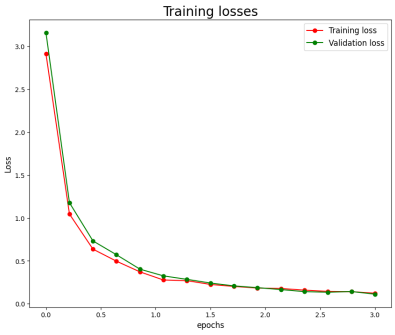


Figure 6: Training Losses (ResNet)

The confusion matrix further substantiates the model's efficacy, with a high concentration of true positives along the diagonal, and minimal misclassifications, as shown in the visualization (Fig. 7 & Fig. 8.). This uniformity demonstrates the model's proficiency in classifying images across various classes with high reliability.

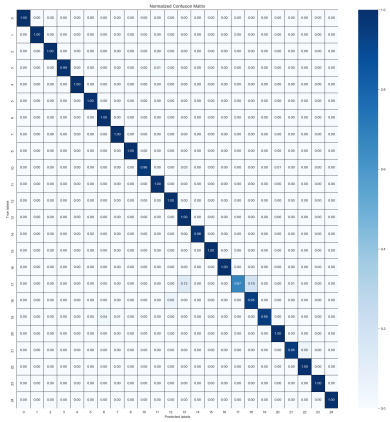


Figure 7: Confusion Matrix of the CNN Model Predictions

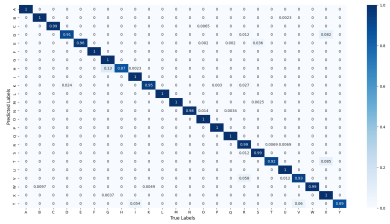


Figure 8: Confusion Matrix of the ResNet Model Predictions

Further scrutiny of the model's predictions showcases examples of correct classifications, which are crucial for evaluating the practical utility of the CNN (Fig.9). The sample images provided illustrate instances where the model has accurately predicted various classes, matching the true labels. This alignment between the predicted and actual classes across a diverse set of inputs underscores the model's robustness and its ability to generalize well to new data.

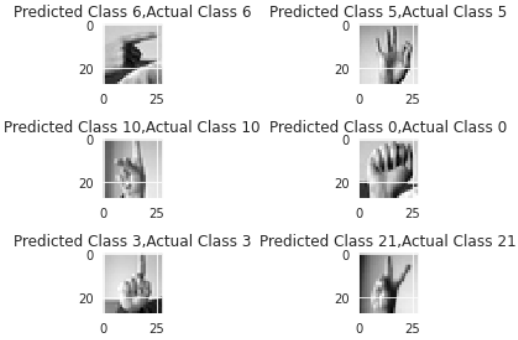


Figure 9: Sample Images of Correctly Predicted Classes

These examples of correct predictions serve as a testa-

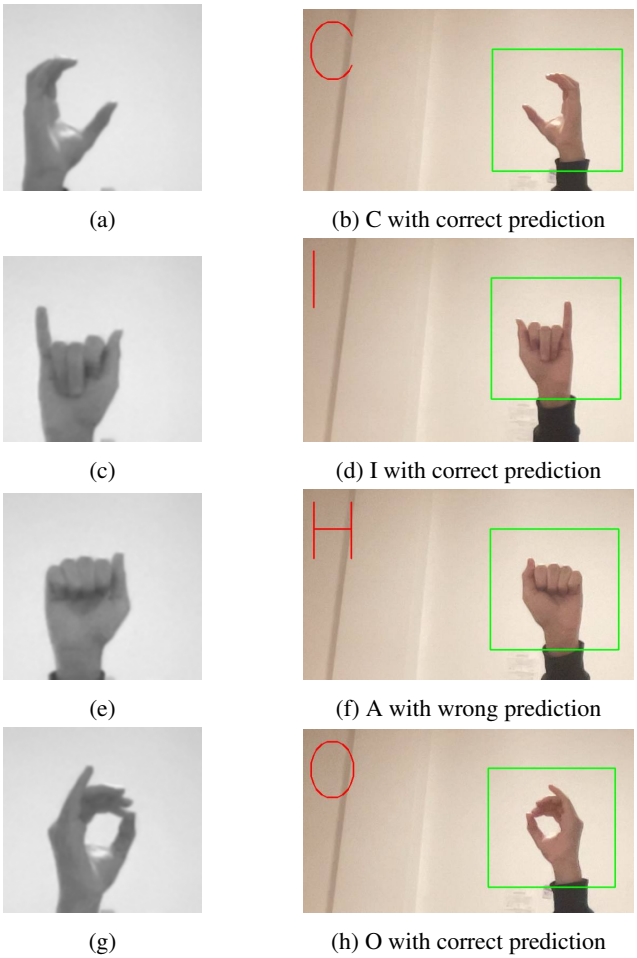
ment to the model's precision in interpreting the sign language gestures, a critical aspect for the system's intended use in real-world applications.

5.1. Real-Time Translation with OpenCV

To verify the functionality of our model in recognizing ASL alphabet signs, we employed OpenCV in a Human-Computer Interaction (HCI) setting. We developed a script to access the camera of the user's PC for real-time image recognition. The system captures images on the user's screen. To differentiate the sign from the background, the accumulated weighted average was computed for subtraction, and a threshold of 25 was used for contour detection. The images, once captured and processed (resized, flipped, and converted to grayscale). The image, in RGB colour format, was then passed to our pre-trained model.

The predictions of the latter appeared on the user's screen in the window. The most accurate prediction was represented with a reed character, corresponding to the orthographic sign of the finger-spelled ASL sign. For this experiment, a user signed each of the 22 letters of the dataset 50 times in three different settings with diverse backgrounds and lighting conditions. Note that the user is not a highly proficient signer but a learner. Figure 10 below displays some examples

Upon evaluating the accuracy for each sign, it was found that the system correctly identified signs with an overall accuracy rate of 77.2%, taking an average time of 3.46 seconds for each accurate recognition. Notably, the signs A, Q, X, and M were most frequently misidentified, often being confused with H, P, C, and W, respectively. This misclassification is likely due to the visual similarities between these signs, coupled with the user's limited experience in signing. The model, trained on images captured from various perspectives including front, right, left, top, and bottom, showcased its effectiveness in recognizing most signs through the implementation of the OpenCV library. This included accurate detection of signs even when performed at atypical or imprecise angles by the user, differing from the representations shown in Figure 1.



(a) (b) C with correct prediction

(c) (d) I with correct prediction

(e) (f) A with wrong prediction

(g) (h) O with correct prediction

Figure 10: Examples of ASL alphabet signs recognition

6. Discussion

The main disadvantage in tackling the problem of the real time implementation is the great difference between the samples contained in the original dataset and the frames taken from the live camera. To improve the model performance we collected personally a new dataset (Fig. 11.), capturing 64x64 pixels images of the left hand of one of the two authors: 200 example of the 24 categories have been collected. To make the acquisition easier, the pre-implemented hand detection tool from Google (*mediapipe*) was used to detect hands and isolate them from the rest, enabling to have well centred images and quick acquisition. The data augmentation technique applied is the same as aforementioned. Training the model with the 80% of this new samples and testing on the remaining 20% shows good results in terms of accuracy. Consequently also the real time implementation resulted in a better precision and accuracy, but given the small size of the dataset and the lack of experience in performing signs by the author, we consider this experience as a starting point for future developments. The pipeline followed for the acquisition of the images and training the model could lead, with an expert in the field of sign language, to develop recognition tools also for other sign languages, such as Spanish, French, Italian etc.

7. Conclusions and Future Work

7.1. Conclusion

In our project, we developed an American Sign Language (ASL) translation tool using a CNN-based classifier. This model showed proficiency in recognizing ASL letters, except for J and Z. However, our testing revealed that the

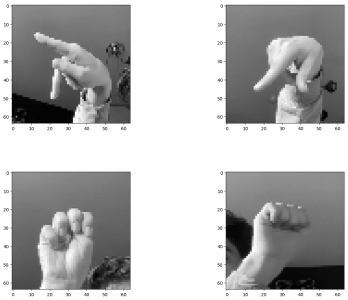


Figure 11: Examples of new dataset

model's validation accuracies during training didn't fully translate to its real-world application. We believe that enriching our dataset with more varied environmental conditions will enhance the model's ability to generalize and accurately recognize all ASL letters.

7.2. Future Work

Additional Datasets: Using the MNIST dataset for real-time sign language translation applications is possible but with some limitations. The original MNIST dataset is designed for handwritten digit recognition, so it doesn't directly contain sign language data. For a comprehensive ASL translation system, additional datasets that include a wider range of signs and dynamic gestures would be necessary.

8. References

1. World Health Organization. (2022). Deafness and hearing loss. Retrieved from https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

2. Pathan, R.K., Biswas, M., Yasmin, S. et al. Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network. Sci Rep 13, 16975 (2023).

3. Garcia B., Viesca S.A. et al. Real-time American Sign Language Recognition with Convolutional Neural Networks. Stanford University (2016).

4. Cooper, H. M., Ong, E. J., Pugeault, N., & Bowden, R. (2012). Sign language recognition using sub-units. Journal of Machine Learning Research, 13, 2205-2231.

5. Rastgoo, R., Kiani, K., & Escalera, S. (2021). Sign language recognition: A deep survey. Expert Systems with Applications, 164, 113794.

6. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep Learning for Visual Understanding: A Review. Neurocomputing, 187, 27-48.

7. Tamura, S., & Kawasaki, S. (1988). Recognition of sign language motion images. Pattern Recognition, 21(4), 343-353.

8. Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American Sign Language Recognition Using Desk and Wearable Computer-Based Video. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12), 1371-1375.

9. Koller, O., Zargaran, S., & Ney, H. (2015). Deep Learning of Mouth Shapes for Sign Language. In Proceedings of the 3rd Workshop on Recognition and Action for Scene Understanding (REACTS2015).

10. Cui, R., Liu, H., & Zhang, C. (2019). A Deep Neural Network for Real-Time Detection of Sign Language. IEEE Access, 7, 129902-129911.

11. Kadhim, R. A., & Khamees, M. (2020). A Real-Time American Sign Language Recognition System using Convolutional Neural Network for Real Datasets. Tem Journal, 9(3).

12. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

13. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

14. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

15. https://www.kaggle.com/datasets/datamunge/sign-language-mnist/data

16. Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (2nd ed.). Prentice Hall.

17. LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (2002). Efficient backprop. In Neural networks: Tricks of the trade (pp. 9-50). Berlin, Heidelberg: Springer Berlin Heidelberg.

18. Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.

19. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

20. O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., & others. (2019). Keras Tuner. https://github.com/keras-team/keras-tuner

21. Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. Neural Networks, 11(4), 761-767.

22. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).