Recognition of Romanian Sign Language Alphabet Using Convolutional Neural Networks

Emilia-Maria Nuță,
Gheorghe Cosmin Silaghi
Babeș-Bolyai University, Business Informatics Research Center
Cluj-Napoca, Romania
nuta.emilia01@gmail.com, gheorghe.silaghi@ubbcluj.ro

ABSTRACT

The recognition of sign languages is critical for facilitating communication between the deaf and hearing communities. This paper addresses the problem of recognizing the Romanian sign language (RSL) alphabet using convolutional neural networks. We trained both custom made and transfer learning models. Due to non-availability of datasets on RSL, we recorded a dataset from two different subjects performing 26 signs in 4 different lightnings. Comparative analysis demonstrates that leveraging transfer learning outperforms the custom models in terms of accuracy. Key contribution of this paper is the development of a comprehensive dataset for the RSL alphabet that was not available before, facilitating the creation of other models in this area. Our findings underscore the potential of deep learning in advancing assistive technologies.

Author Keywords

Romanian sign language; Convolutional neural networks; Image classification; Deep learning

ACM Classification Keywords

•Computing methodologies-Artificial intelligence- Computer vision •Human-centered computing- Accessibility-Accessibility systems and tools

DOI: 10.37789/ icusi.2024.5

INTRODUCTION

Sign language is the primary means of communication used by the deaf and hard of hearing population worldwide. It is a language that involves various hand movements, facial expressions and gestures to convey meaning. A common misconception about sign languages is that they are universal or just manual and facial versions of the spoken language [3]. This is far further from the truth, since sign language can vary even in the same country. According to the National Authority for the Protection of Rights of Persons with Disabilities[6], in Romania, as of Q1 2024, 23.620 persons are affected by various degrees of hearing loss. Having an intelligent system which can bridge the communication between them and the hearing population would greatly enhance their ability to interact effectively in society. Such a system, particularly one that accurately recognizes and translates the sign language into text or speech, could facilitate access to education, employment, and social services for individuals with hearing impairments.

In this paper we aim at creating a dataset for the Romanian sign language (RSL) and experiment with the recorded images in order to see how a well the classic image recognition models succeed in sign recognition.

The remainder of this paper is organized as follows: the next section reviews the existing literature regarding classification of sign language images. Section *Methodology* details the dataset creation and models development. Section *Results* presents and discusses the obtained outcomes, and section *Conclusions* outlines key finding and future directions.

RELATED WORK

Sign language recognition is a field aimed at facilitating communication for the deaf and hearing-impaired.

The main approaches for recognizing the sign language are wearable sensing modalities and vision-based methods [1]. The sensor-based approach utilizes data gloves with built-in sensors to track hand movements and articulations [9]. However, these gloves can be costly. The more commonly used vision-based approach employs cameras to capture images or video of signs, followed by image processing and recognition algorithms [5]. Vision-based methods can be further categorized into color-based and depth-based techniques, depending on the type of camera used [10].

In terms of machine learning techniques, Convolutional Neural Networks (CNNs) are the most used due to their effectiveness. These deep learning models can automatically extract discriminative features from raw video or image data, eliminating the need for handcrafted features. Various CNN architectures have been applied to recognize different sign languages demonstrating their effectiveness, such as Indian [7], Italian [6], and American Sign Language [2]. Despite the progress made by CNNs in the task of sign language recognition, several challenges are still open, as for many

[6] https://anpd.gov.ro/web/transparenta/statistici/

languages, including the RSL, image datasets are not available.

METHODOLOGY

This section introduces our RSL dataset, the architectures used for our recognition experiments and other issues considered when conducting these experiments. The code was written in Python 3.10 and executed on a Google Colab paid subscription. For implementation, we leveraged the Keras deep learning API. The datasets and the experiments are available at https://github.com/Emilia-Maria/Romanian- Sign-Language-Image-Classification/

Dataset

Unavailability of RSL datasets prompted us to create our own. RSL is composed of 31 letters, but because some of the signs are made by moving another (e.g. Î and J are a variation of I), we will keep just 26 which are purely static, ending up removing Î, J, Ș, Ț, Z. The RSL alphabet is depicted in Fig.1.

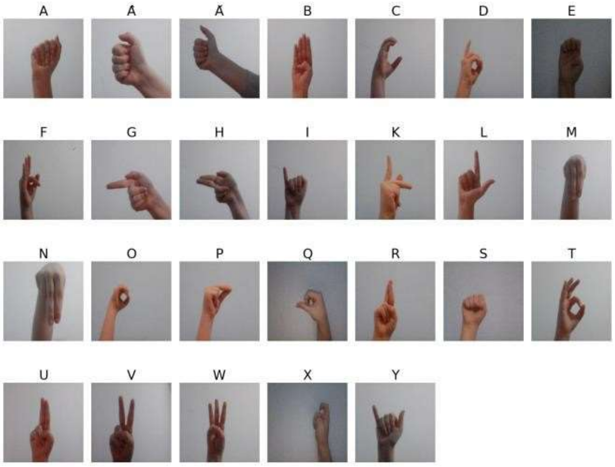


Figure 1. Romanian sign language alphabet.

The images for the dataset were captured using a VGA camera built into a personal laptop. The VGA camera has a resolution of 640x480 pixels and provides a balance between image quality and file size, which makes it suitable for our data collection needs. The decision to use the integrated camera stemmed from the ease of use, the non-existent costs involved, and the consistency in image capture.

To introduce variability and ensure robustness of the dataset, we collected images from two different individuals, namely a woman and a man. This approach will enable future models to generalize across different hand shapes and sizes, which is crucial for a real-world application of sign language recognition.

Illumination plays a significant role in the image collection tasks. This diversity of lighting conditions ensures that the model can handle variations in illumination, shadows and contrast that are common in real-world scenarios. In order to simulate various real-world conditions, the images were taken in four different lighting scenarios:

- Natural light: Images are captured during daylight, using natural sunlight as the main light source
- Artificial light: Images are captured indoor with artificial lighting, using LED bulb lights as the main light source
- Low light: Images are captured under low light conditions, simulating dimly lit environments
- Bright light: Images are captured with a bright light source directed at the hand, using the flash of a phone

The image capturing process was standardized to maintain consistency and quality across the dataset. Each participant performed sign language gestures in front of the laptop camera, ensuring that they were clearly visible. The following were considered during the image capture process:

- Background: A simple, white, uncluttered background was used to minimize distractions and focus on hand gestures
- Hand signs: Each participant performed each sign several times to capture slight variations in execution

In total, 26.000 images were collected, with 1.000 images for each of the 26 different gestures. This dataset was divided into train, validation and test with a ratio of 6:2:2.

Architecture

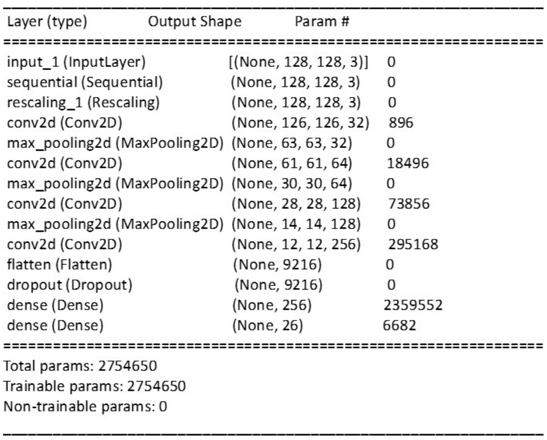
We explored two main strategies for constructing image recognition models: training a custom CNN model from scratch and employing transfer learning by leveraging the VGG16 architecture [8].

Our proposed CNN model depicted in Fig.2 is designed as a sequence of convolutional and pooling layers, finishing with two fully connected layers, the last one performing the classification. This architecture aims to efficiently extract and learn hierarchical feature representations from the input images of size 128x128.

The model comprises four convolutional layers, each with a 3x3 kernel size, a stride of 1, and valid padding to preserve the spatial dimensions of the input. The layers progressively increase in the number of filters, starting with 32, 64, 128, and 256 filters in the last layer to capture intricate patterns in the data. Each convolutional layer uses the ReLU activation function to introduce non-linearity into the model. Below each convolutional layer stays a max-pooling layer with a pooling size of 2x2 and a stride of 1. These max-pooling layers perform down-sampling operations, reducing the spatial dimensions of the feature maps while retaining the most relevant features. The pooling operation utilizes valid padding, ensuring no padding is added to the input of the pooling layers. After the final max-pooling layer, a flatten layer is used to convert the 2D feature maps into a 1D vector. The flattened vector is then fed into a fully connected dense layer with 256 units, activated by a ReLU function. The final layer with 26 units, corresponding to the number of letters is equipped with the softmax activation function and outputs a distribution of 26 probabilities, one for each letter.

The VGG16 model [8] is one of the basic architectures used for transfer learning in shallow image recognition tasks. The model consists of 5 convolutional blocks, each with 3 convolutional layers of 3x3 size kernels with a stride of 1 and padding. The dense network includes 2 fully connected layers with 4096 units each, followed by a fully connected layer with 1000 units, as VGG16 was initially constructed for recognizing 1000 image categories.

To utilize the VGG16 model available in Keras, we first learned our 26-letters classifier on top of VGG16. After the weights of the classifier where initialized, we unfroze the convolution base and trained only the last block, trying to obtain specific features
for our RSL recognition task, as none of the original VGG16 1000 classes are related to RSL.



```
_________________________________________________________________
Layer (type)              Output Shape              Param #
=================================================================
input_1 (InputLayer)            [(None, 128, 128, 3)]   0
sequential (Sequential)         (None, 128, 128, 3)     0
rescaling_1 (Rescaling)         (None, 128, 128, 3)     0
conv2d (Conv2D)                 (None, 126, 126, 32)    896
max_pooling2d (MaxPooling2D)    (None, 63, 63, 32)      0
conv2d (Conv2D)                 (None, 61, 61, 64)      18496
max_pooling2d (MaxPooling2D)    (None, 30, 30, 64)      0
conv2d (Conv2D)                 (None, 28, 28, 128)     73856
max_pooling2d (MaxPooling2D)    (None, 14, 14, 128)     0
conv2d (Conv2D)                 (None, 12, 12, 256)     295168
flatten (Flatten)               (None, 9216)            0
dropout (Dropout)               (None, 9216)            0
dense (Dense)                   (None, 256)             2359552
dense (Dense)                   (None, 26)              6682
=================================================================
Total params: 2754650
Trainable params: 2754650
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. Model summary.

Data augmentation was done to artificially expand the size of the training dataset by flipping horizontally, rotating by a factor of 0.05 and zooming by a height factor of 0.2. Each time an image is passed through this augmentation pipeline during training, it can randomly be subjected to all three of these transformations. This will make an image different each time, making the training process more robust by exposing the model to a wide variety of transformed images. These changes were chosen to preserve reality. Dropout [4] rate was set to 0.5, letting the model to randomly set half of the neurons to 0 in order to prevent overfitting and encouraging the model to learn more robust features. The last regularization technique utilized was L2 regularization or weight decay, which adds a penalty to the loss function to keep weights small. We also varied the capacity of the dense layer by testing with 64, 126, 256 and 512 nodes.

Model	Dense units	Accuracy (%)	F1
Simple	256	82,1	0.80
Augmented	256	83,3	0.84
Augmented + Dropout	256	85,7	0.83
	64	80,9	0.83
Augmented + Dropout + Regularized	128	84,2	0.82
	256	83,5	0.84
	512	81,6	0.86
VGG16	256	89,2	0.87

Table 1. Accuracy of all models.

RESULTS

In our study, we evaluated five different CNN models for the task of recognizing the RSL alphabet. These included four custom-designed models and one model based on transfer learning using the VGG16 architecture. Custom models are based on the architecture described in Fig. 2 incrementally enhanced with data augmentation, dropout and regularization techniques.

Letter	Precision	Recall	F1	Instances
A	0.78	1	0.88	200
Â	0.99	1	1	200
Ă	0.99	0.99	0.99	200
B	0.9	0.98	0.94	200
C	0.95	1	0.98	200
D	0.99	0.41	0.58	200
E	0.96	0.67	0.78	200
F	0.98	0.99	0.99	200
G	1	0.96	0.98	200
H	0.96	1	0.98	200
I	0.97	0.76	0.85	200
K	0.84	0.97	0.9	200
L	0.97	1	0.99	200
M	0.55	0.88	0.67	200
N	0.81	0.27	0.4	200
O	0.71	1	0.83	200
P	0.98	0.98	0.98	200
Q	1	0.98	0.99	200
R	0.98	0.81	0.89	200
S	0.78	0.82	0.8	200
T	0.95	0.94	0.95	200
U	0.87	0.98	0.92	200
V	1	0.88	0.93	200
W	0.88	1	0.93	200
X	0.96	0.93	0.94	200
Y	0.86	1	0.92	200

Table 2. Classification report of VGG16 model.

To comprehensively evaluate the effectiveness of the models, we used accuracy, precision, recall and F1 score. The evaluation of these models (Table 1) highlights the progressive improvements obtained through data augmentation and dropout techniques. The transfer learning approach with VGG16 significantly outperformed the custom models, underscoring the efficacy of leveraging pre- trained networks for complex image recognition tasks.

Following the VGG16 model, we computed the classification report of Table 2, which highlights the precision, recall and F1 score for each letter in the test dataset. Several classes have good performance, such as Â, Ă, G, H, L and Q indicating that the model can accurately identify and classify these letters. From the confusion matrix depicted in Fig. 3 we observe in more detail that the model had problems with the letters D, E, I and N. Letter N is the most problematic one, with the confusion matrix revealing frequent misclassifications as M. The low recall rate for N is caused by the high rate of confusion, which indicates that the visual cues that the model utilized to differentiate N from these other letters significantly overlapped with M.

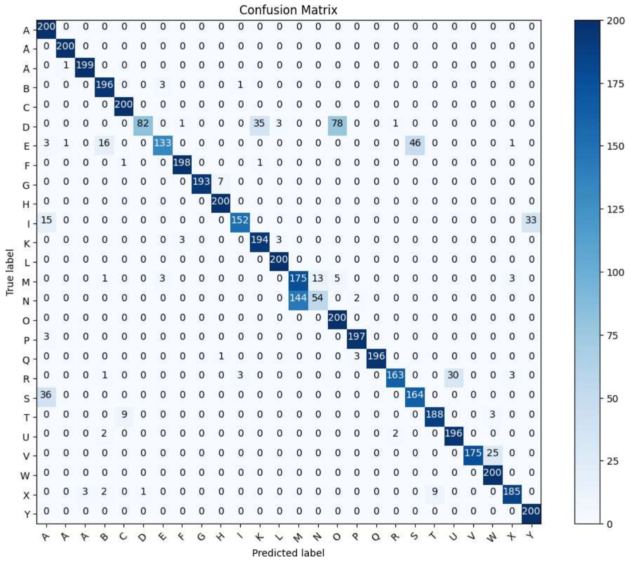


Figure 3. Confusion matrix for the VGG16 model.

CONCLUSION

The exploration into the recognition of RSL alphabet leveraging CNNs demonstrates significant potential in advancing communication accessibility for the deaf and hard of hearing communities. The implications of our research are significant for RSL, providing an extensive dataset to be used in other experiments regarding this topic.

Future work will prioritize the development of a real-time sign language recognizer. This addresses the major limitation of the research, since we used static images and thus limiting the ability to recognize dynamic hand movements and continuous signing. Recognizing letters from moving frames would advance towards solving the automated transcription task for RSL.

REFERENCES

1. Adeyanju, I.A., Bello, O.O., Adegboye, M.A. (2021) Machine learning methods for sign language recognition: A critical review and analysis. *Intelligent Systems with Applications*, 12, 200056.
2. Bantupalli, K., and Xie, Y. (2018) American Sign Language Recognition using Deep Learning and Computer Vision. *IEEE Intl. Conf. on Big Data (Big Data)*, pp. 4896-4899, IEEE Press.
3. LeMaster, B.., Monagham, L. Variation in Sign Languages (2004) In *A Companion to Linguistic Anthropology*, Blackwell Publishing, pp. 141-165
4. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R. (2012) *Improving neural networks by preventing co-adaptation of feature detectors*, CoRR abs/1207.0580
5. Nimisha, K., Jacob, A. (2020) A Brief Review of the Recent Trends in Sign Language Recognition. *Intl. Conf. on Communication and Signal Processing (ICCSP),* pp. 186–190, IEEE Press
6. Pigou, L., Dieleman, S., Kindermans, P.J., Schrauwen, B. (2014) Sign Language Recognition Using Convolutional Neural Networks. In *European Conf. on Computer Vision (ECCV), Workshops, LNCS vol. 8925, pp.* 572–578, Springer
7. Rao, G.A., Syamala, K., Kishore, P.V.V., Sastry A.S.C.S. (2018) Deep Convolutional Neural Networks for Sign Language Recognition. in *Conf. on Signal Processing and Communication Engineering Systems (SPACES),* pp. 194-197, IEEE Press
8. Simonyan, K., Zisserman, A. (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. in *Intl. Conf. on Learning Representations*
9. Taneja, M., Singla, N., Goyal, N., Jindal, R. A Comprehensive Review of Sensor-based Sign Language Recognition Models (2022) *Journal of Xi'an Shiyou University,* 19, pp. 292-302
10. Yang, L., Zhu, Y., Tao, L. (2019) Towards Computer- Aided Sign Language Recognition Technique: A Directional Review. In *Advanced Information Technology, Electronic and Automation Control Conf.* (*IAEAC)* IEEE Press