Image Classification Using Convolutional Neural Networks

Sebastian Wilke[1], Mirsad Sukilovic[2], and Mariia Melnik[3]

[1] Department of Computer Science; Humboldt-Universität zu Berlin; Berlin, Germany
[2] Department of Business and Economics; Humboldt-Universität zu Berlin; Berlin, Germany
[3] Department of Physics and Mathematics; Humboldt-Universität zu Berlin; Berlin, Germany

March 1, 2021

Contents

Abstract

This project report presents our work on image classification using convolutional neural networks (CNNs). During the course of this semester we built a machine learning model that classifies images from the Sign Language MNIST dataset. The dataset contains American Sign Language letters of hand gestures. Additionally, we created three test datasets with different lighting and background settings to further assess the performance in a real world setting. We conclude with an evaluation of the model and point out possible limitations as well as steps needed to successfully deploy a model like this in a real application.

Keywords
machine learning, robot, ASL, MNIST, classification, CNN, sign language

1 Introduction

The objective of this project is to train a machine learning model that classifies letters from the American Sign Language (ASL). It is a complete natural language which serves as the predominant sign language of deaf communities in the United States and most of Anglophone Canada. It is expressed by movements of hands and facial expressions. ASL is expected to have $250,000 - 500,000$ native speakers in the United States[4].
For training and testing of our model we use the MNIST Sign Language dataset[5]. Our model could be deployed in a number of different computer/robot vision tasks like assisting humans in understanding or translating sign language.
We think this task is important to solve because humanoid robots should be able to communicate with deaf and mute people. Especially since only few non-deaf people know sign language.

2 Related work

Research on sign language recognition has already been conducted worldwide using various sign languages. This section aims to provide an overview of methods used in this area to date and to present some work by other authors.

Munib et al.[1] presented a system for automatic translation of static gestures of alphabets and signs in ASL. They used Hough transform and neural networks. A dataset of 300 samples of hand sign images with 15 images for each sign was used.
Aryanie and Heryadi[2] developed a camera based fingerspelling recognition system. In their system, features were extracted using color histogram and principal component analysis was employed to reduce dimensions of the extracted feature set.
Oz and Leu[3] developed an ASL recognition system using sensory glove and 3D motion tracker to convert ASL words into English. They collected static single handed samples from 50 words and trained an artifical neural network as a classifier.

3 Materials and Methods

3.1 Dataset specifications and preprocessing steps

The Sign Language MNIST dataset matches closely with the classic MNIST. It consists of $27,455$ training and $7,172$ test cases. Each case represents a 28×28 gray-scale image that is cropped to hands-only. The labels are in range $[0;24]$ and represent an alphabetic letter $A-Z$. The letters J and Z and their corresponding label are excluded since they require motion. The dataset is given in CSV file format. Both training and testing set contain a header row label,pixel1,…,pixel784. Each of the following rows describes an image with grayscale values in range $[0;255]$.

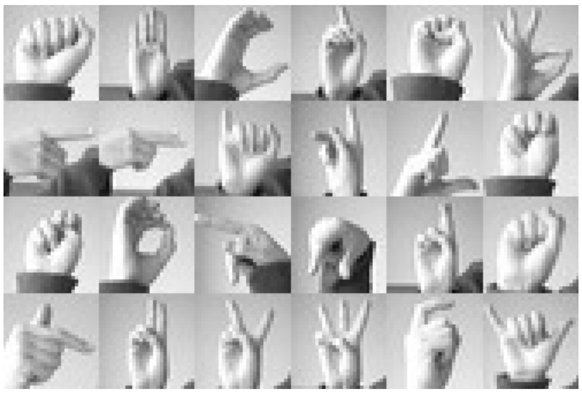

Figure 1: Shows examples of ASL hand signs from the original Sign Language MNIST dataset.

To prevent overfitting and accurately evaluate our models performance, we split the training data into a 80% (21,964 images) training and a 20% (5,491 images) validation set. We then normalized the pixel values to $[0,1]$. We used keras ImageDataGenerator to augmented the dateset by flipped, shifted, and zoomed in images.
To further test our model in a real world setting, we created three test datasets consisting of 24 images for each of the 24 letters. Of these datasets one was cropped to hands only.
For each of these test datasets $T1-T3$ we created a CSV file matching closely with the format of the original dataset. This meant gray-scaling and resizing all of images to 28×28.

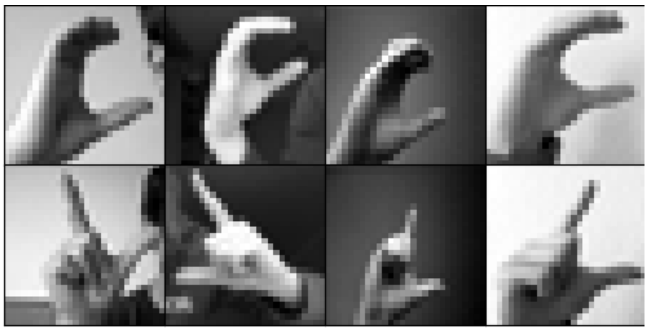

Figure 2: The letters C (top) and L (bottom). From left to right: original dataset, $T1$, $T2$, $T3$.

3.2 Machine learning model

To build the model we first did a couple of experiments to figure out the best hyperparameters. The code and plots of these experiments can be found in the repository to this report. Since the input consists of 28×28 images, our InputLayer has an input shape of $(28,28,1)$. As can be seen in figure 3 (page 3), our model has 9 Hidden Layers. It consists of two convolution-subsampling pairs both made up of a *Conv2D*, *MaxPool2D* and *Dropout* layer. The first convolutional layer has of 32 units, the second 64 units. After that we use a *Flatten* and a *Dense* layer. We use *max pooling* since we want to extract the most salient features. For activation we apply the *Rectified Linear Unit* (ReLU) function $F(x) = \max(0,x)$ but use *Softmax* function in the output layer. *Categorical Crossentropy* is used as loss function and *Adam* as optimizer. Since there are 24 labels to predict, we also have 24 neurons in the output layer. We trained the model with 10 epochs using a batch size of 64.

4 Results and discussion

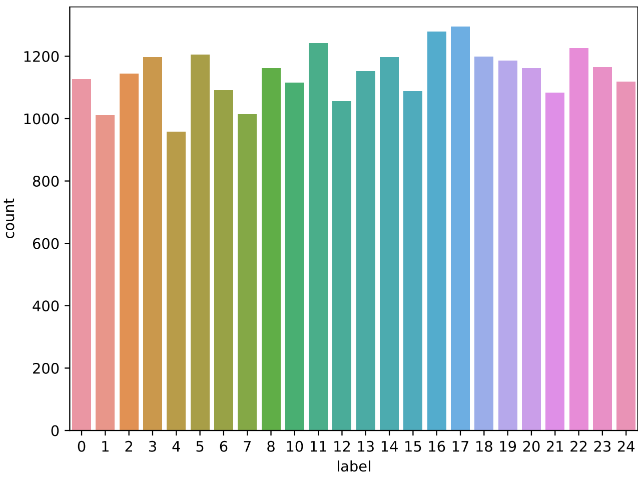

Figure 4: Countplot of training dataset

As can be seen from figure 4, the classes in the training set are evenly distributed. It is thus enough to focus on accuracy as the evaluation metric. Using our model, we achieve an evaluation accuracy of 99%. Additional evaluation metrics as well as a confusion matrix (figure 5) can be found near the end of this section.

4.1 Description of additional test sets

As mentioned in the previous section, we created three additional test sets, each consisting of 24 photos with different background and lighting settings:

- **Test Set 1** ($T1$): The images in this testset are not cropped and contain objects like doors and lights. This was done to check the influence of background noise on the prediction. Additionally some of the images do not have the same orientation like in the training set (see letter L of testset $T1$ in figure 2) The result is evident: The model performed very bad with an accuracy of only 8%. This corresponds to a correct classification of 2 out of the 24 letters.

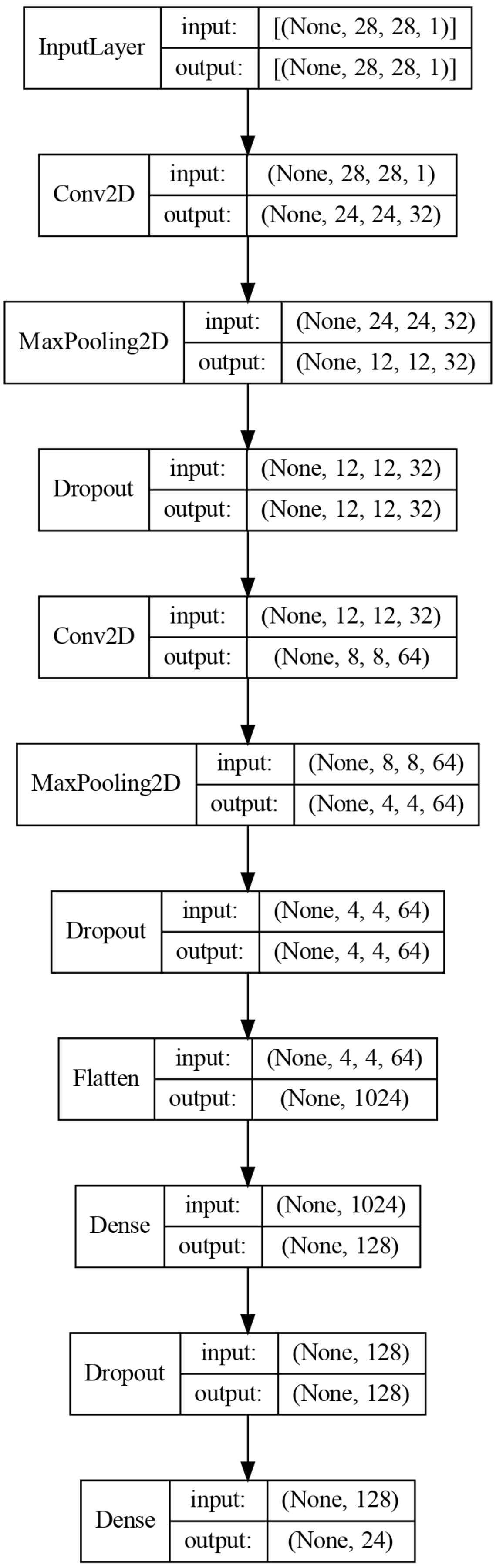

Figure 3: Final model

- **Test Set 2** ($T2$): The images in this testset have no other objects in them but are not cropped either. In addition, the hue of the white background is slightly different from that of the images used for training the model. As expected, the result is better than for $T1$ with an accuracy of 16%. This corresponds to a correct classification of 4 out of the 24 letters. This shows that background noise has a significant effect on the prediction result.

- **Test Set 3** ($T3$): The images in this testset resemble the images in the original training set the closest. They are cropped to hands only and only have white background. The goal was to check whether our model achieves a high accuracy when the images are reproduced in detail (i.e. as similar to the training images as possible). Not suprisingly the results were much better than for $T1$ and $T2$. The model achieved an accuracy of 75% which corresponds to a correct classification of 18 letters for 24 letters.

4.2 Real world application and Limitation

We briefly mentioned one possible application in the introductory section of this paper. A robot equipped with a camera could take in image data and output desired results via speaker or display. We observe that although our model correctly classified almost every letter from the original test set, it did not perform particularly well when classifying images from the test sets $T1 - T3$. It might seem like our model is only performing well, if the input images look similar to the images the model was trained on. This is of course not always the case and should not be required of any sufficient model.

4.3 Possible solutions

One solution is to augment the training dataset with thousands of additional images taken in the application setting. During the course of the model being in use it would also get better the more data is being collected. From our observations of test sets $T1 - T3$ it is clear that any real world application should implement an effective object recognition algorithm. The images could then be cropped accordingly, thus removing background noise and passed to our model.

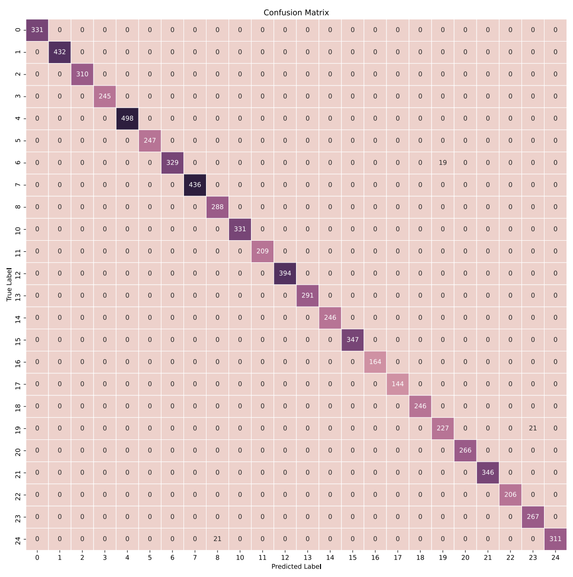

Figure 5: Confusion Matrix

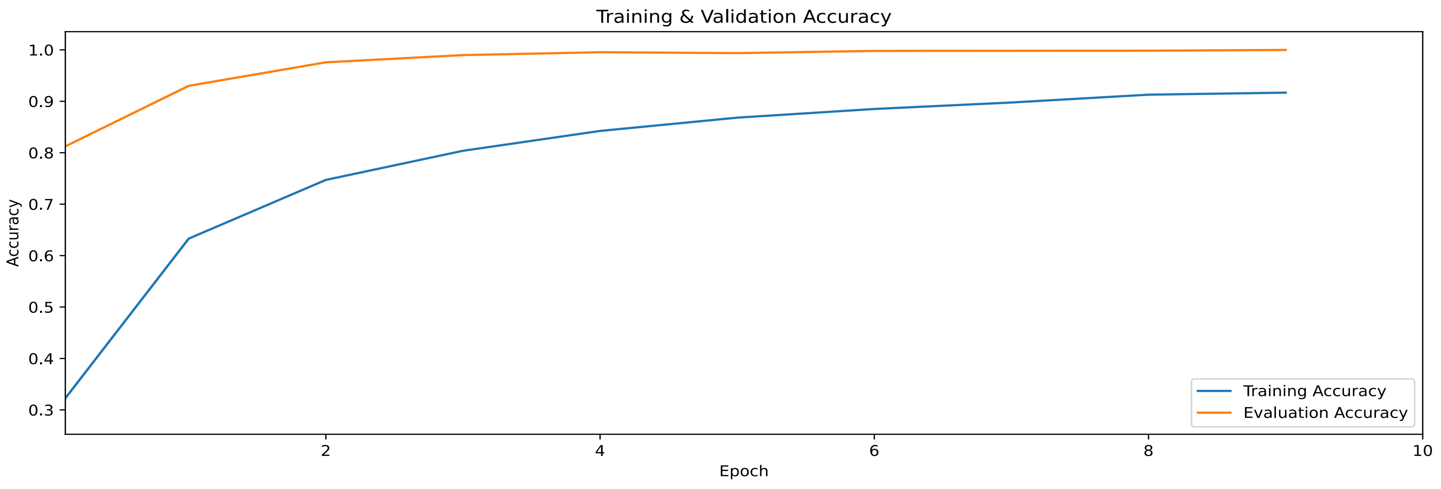

Figure 6: Accuracy of our model

	precision	recall	f1-score	support
Class 0	1.00	1.00	1.00	331
Class 1	1.00	1.00	1.00	432
Class 2	1.00	1.00	1.00	310
Class 3	1.00	1.00	1.00	245
Class 4	1.00	1.00	1.00	498
Class 5	1.00	1.00	1.00	247
Class 6	1.00	0.95	0.97	348
Class 7	1.00	1.00	1.00	436
Class 8	0.93	1.00	0.96	288
Class 10	1.00	1.00	1.00	331
Class 11	1.00	1.00	1.00	209
Class 12	1.00	1.00	1.00	394
Class 13	1.00	1.00	1.00	291
Class 14	1.00	1.00	1.00	246
Class 15	1.00	1.00	1.00	347
Class 16	1.00	1.00	1.00	164
Class 17	1.00	1.00	1.00	144
Class 18	1.00	1.00	1.00	246
Class 19	0.92	0.92	0.92	248
Class 20	1.00	1.00	1.00	266
Class 21	1.00	1.00	1.00	346
Class 22	1.00	1.00	1.00	206
Class 23	0.93	1.00	0.96	267
Class 24	1.00	0.94	0.97	332
accuracy			0.99	7172
macro avg	0.99	0.99	0.99	7172
weighted avg	0.99	0.99	0.99	7172

5 Reproducibility of the study

We built our model using the keras Sequential model API since we only have one input, one output and we do not need layer sharing. For plots and visualization we used matplotlib. All code was written in python 3.8.
The interested reader can find datasets $T1 - T3$, all the experiments, plots and code in the github repository[6] to this report.

6 Conclusion

In this report, we proposed a machine learning pipeline, which utilizes a convolutional neural network for image classification. We achieved high accuracy and are quite content with the results. Next steps could involve inclusion of facial expressions, two-handed signs or realtime classification. We also pointed out possible limitations and steps needed to be able to deploy a model like this in a real application.

Author contributions

Research, coding and analysis was evenly distributed among the group members.

References

[1] Munib, Qutaishat & Habeeb, Moussa & Takruri, Bayan & Al-Malik, Hiba. (2007). American sign language (ASL) recognition based on Hough transform and neural networks. Expert Systems with Applications. 32. 24-37. 10.1016/j.eswa.2005.11.018.

[2] Dewinta, & Heryadi, Yaya. (2015). *American Sign Language-Based Finger-spelling Recognition using k-Nearest Neighbours Classifier.* 10.1109/ICoICT.2015.7231481.

[3] Cemil Oz, Ming C. Leu. (2011). *American Sign Language word recognition with a sensory glove using artificial neural networks*, Engineering Applications of Artificial Intelligence, Volume 24, Issue 7, Pages 1204-1213.

[4] Mitchell, Ross & Young, Travas & Bachleda, Bellamie & Karchmer, Michael. (2006). *How Many People Use ASL in the United States? Why Estimates Need Updating.* Sign Language Studies. 6. 10.1353/sls.2006.0019.

[5] The Sign Language MNIST dataset: `https://www.kaggle.com/datamunge/sign-language-mnist`

[6] `https://github.com/SebastianWilke/ASL-Classification-Using-CNNs`