Swedish sign language classifier

Moa Burevik[1], Salma Gabot[2], Alice Helander[3]

Abstract
This report describes a project in the course TNM114 - ”Artificial Intelligence for Interactive Media” at Linköping University. The goal of the project was to develop a classifier for the swedish sign language using the deep learning model Convolutional Neural Networks (CNN). The system was designed to classify both static images representing letters in the Swedish sign languages of hand signs and translate hand gestures into text in real-time. The results showed a good result in both tasks despite the challenges with the custom build database.

Source code: https://github.com/salmagabot/realtime-sign-language-classifyer

Database images: https://drive.google.com/drive/folders/1APitc1cqkeyZ0PRgaSEUjOn0kwENCfHz

Authors
[1] *Media Technology Student at Linköping University, moabu603@student.liu.se*
[2] *Media Technology Student at Linköping University, salga107@student.liu.se*
[3] *Media Technology Student at Linköping University, alihe684@student.liu.se*

Keywords: Classification — Convolutional Neural Networks — Swedish sign language

Contents

1. Introduction

This report describes a project that looks into how to classify the Swedish sign language alphabet (STS) and translating different “hand letters” into text in real-time. Sign language is primarily used by the deaf and hard of hearing, but it is also used by others with disabilities. It is a language that uses hand gestures and body language for communication. While previous studies have worked with the American Sign Language (ASL), this report will focus on STS.

CNNs is a deep learning model designed to automatically learn and recognize patterns from visual data, which makes it effective for tasks like image classification and recognition. Moreover CNN is a effective tool that can automatically learn and recognize important features in images without needing specific instructions on what to identify. [1]

2. Theory

The CNN model was created using TensorFlow and Keras libraries. The model helps to find important features from images and make classifications in the following layers: Convolutional Layers, MaxPooling Layer, Flattening Layer and Dense Layers.

2.1 Convolutional Layers

The first layer in CNN is responsible for extracting features from the input images. It uses a process called convolution where a small filter moves over an image. At each position, the filter calculates a dot product between the filter and the part of the image that is covered, creating an output called a feature map.

The feature map highlights important elements like edges and corners, which help identify important parts of the image. The convolutional layers in the CNN keep the pixels intact to detect patterns like edges and shapes in the image. After the

convolution operation, the results were sent to the next layer.

2.2 MaxPooling layer

The Pooling Layer usually comes after a Convolutional Layer. Its main goal is to downsample the feature maps. This was done by reducing the connections between layers. Each feature map was processed separately.

There are different types of pooling operations. In max pooling, the largest value is chosen from the feature map within a specific area. Average pooling calculates the average value of all elements in a specific section of the image. Sum pooling computes the total sum of the values in that same section. In this project max pooling was used.

The pooling layer connects the Convolutional Layer to the Fully Connected (FC) Layer. This helps the CNN model to simplify the features from the convolution layer and making it easier for the network to recognize them. This process also lowers the amount of calculations needed in the network.

2.3 Flattening Layer

The Flattening Layer was used to transform the 2D feature maps into a 1D vector. This step is important because the next layer in the model needs a single list of values instead of a two-dimensional structure.

2.4 Dense Layers

The Fully Connected (Dense) layers are important for making predictions based on the previous layers. In these layers, every neuron is connected to every neuron in the previous layers,. This helps the model to understand the features.

The first Dense layer uses an activation function, ReLU, which helps the network to learn complex patterns. The final Dense layer represents the output classes, giving scores for each class to make the classification more simple. Overall, Dense layers play an important role in turning learned features and delivering final predictions. [2]

3. Method

This paragraph describes the four steps in the methodology, creating a database of images, converting the images into CSV files, creating and training the convolutional neural network model and lastly using the CNN model to classify the real time translator.

3.1 Database

The first step in the method was to develop a database. While it was relatively easy to find similar projects online featuring the American sign language, it was much more challenging to find relevant projects for the Swedish sign language. The images in the databases in the Swedish sign languages projects were often unclear and varied in sizes which made them hard to work with. Because of this the best solution was to create a new database.

The database was made by capturing 46 images of each letter against a clear white background which resulted in a database consisting of 20x46 images of hand gestures representing the alphabet a-z. The letters å, ä and ö were not included in the database since those letters were represented by different movements. Each image was resized to a standard size of 200x200 pixels and converted to RGB format. Moreover the dataset was split into training and testing sets, with 80 procent of the data used for training and 20 procent for testing.

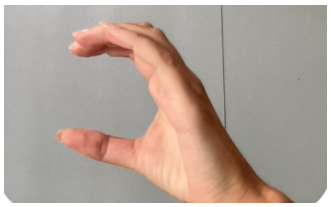

Figure 1. Example of the letter C

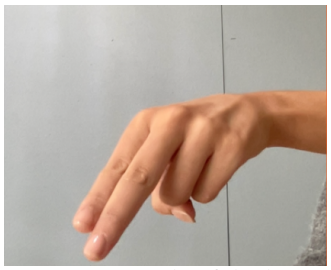

Figure 2. Example of the letter N

3.2 CSV file

After the database was created the next step involved converting the images into a CSV file, where each image contained pixel values between 0-255. The next step included loading the data from the CSV files and converting them into images. To prepare the images for training the CNN model, the pixels were normalized to a range of [0, 1]. All images were also labeled from 0-25 to easily keep track of each letter.

3.3 CNN

The CNN model was created using the library's TensorFlow, Matplotlib, NumPy and Pandas. The model consists of four different layers, three convolutional layers, three pooling layers, one flattening layer followed by two fully connected dense layers. See paragraph 2 for more details. The CNN model was compiled using the Adaptive Moment Estimation (Adam) optimizer and sparse categorical cross entropy as the loss function. The model was trained for 30 epochs, meaning it went through the entire training dataset 30 times. This helps the model to learn the patterns in the data and improve its accuracy.

After training the CNN model, the model accuracy was evaluated using test data. That was a big help for understanding how well the model can recognize new hand gestures it hasn't seen before. The accuracy was displayed in a graph that showed the training and validation accuracy over time.

Lastly the trained model was saved in a keras file so it would be easy to use in the real-time recognition.

3.4 Real time translator

The real time translator was created using the libraries Keras, OpenCV, NumPy and PIL. PIL was included to handle image processing and model loading.

A classification function was implemented to preprocess the images by resizing and normalizing them. The function also made predictions using the model by returning the letter with the highest probability. Moreover a video stream was initiated from the webcam where the webcam captured images in a loop and defined a region of interest (ROI). The ROI is defined as the relevant area on the screen for detecting the hand gestures.

Additionally the ROI were converted to grayscale and pre-processed before being sent to the classification function. This setup allows the predicted letter to appear on the screen.

4. Result

This section presents the result from the CNN classification and the models used in the real-time translator

4.1 Training and validation accuracy for diffrent image sizes

As seen in *Figure 1*, *Figure 2* and *Figure 3* an increasing number of pixels in the images, ranging from 32 to 128, generally leads to improved accuracy. More pixels provide more detailed information about the hand gestures, allowing the model to make more accurate predictions.

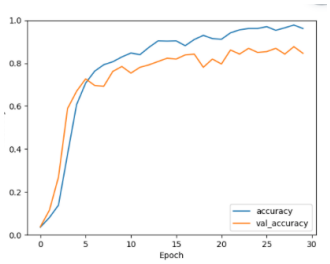

Figure 3. 32x32 pixels

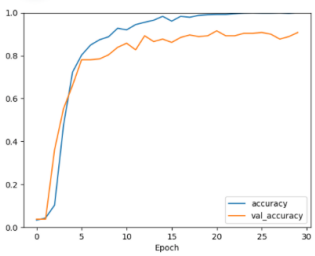

Figure 4. 64x64 pixels

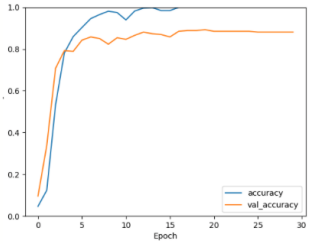

Figure 5. 128x128 pixels

4.2 Real-time translator

The hand gesture classification model was successfully implemented into a real-time translator to recognize different sign language gestures, as shown in *Figure 4*. In the translator, the green box shows the ROI, where the classifier was able to detect hand gestures. The classification was effective for most letters in the STS. However, letter e, g and o were harder to classify because they have more similar structure. The letters d and j were also hard for the classifier to distinguish between.

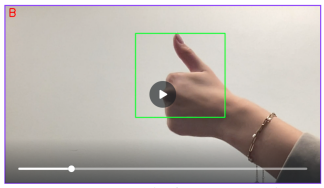

Figure 6. Real-time translator

5. Discussion

The results from the CNN classification and real-time translator implementation showed the relationship between image quality and image classification accuracy. As shown in *Figure 1*, *Figure 2* and *Figure 3*, more pixels in the image generally leads to better accuracy. However, it is important to note that even if higher pixels result in a more detailed image, there is also a higher risk of overfitting, which makes it harder for the realtime model to verify.

The model successfully classified several letters from the STS. However, the model had difficulty classifying the letters e, g, o, because of their similar structure. Likewise the model struggled to distinguish between the letters d and j. This indicates that we need to improve the way the model understands these gestures.

It is also interesting to compare our results with prior studies that have used CNN to classify ASL gestures. These studies used large datasets with thousands of imagies per letter and focused on regognizing static hand gestures. However, despite the large dataset, they still faced problems recognizing simular structure. [3], [4] Since ASL include diffrent letters than the STS, the issue were not with the same letters. For exemple, Zhang et al. (2019) faced the challenge to distinguishing between the letters m and n, since they have a very similar structure in the ASL. [4] This highlights the importance of having a consistent data set and better model training

to improve accuracy. [3]

Overall, we were not completely satisfied with our model's accuracy. The lower accuracy might be because of the small number of images in our dataset. However, we tried using a Kaggle dataset with around 1,600 images per letter, but this did not improve our results. The reason for that could be the low quality of those images, which is another reason that can affect our result.

To improve our model, we would need a larger dataset with high-quality images. This will help the model to both recognize the different gestures better but also to reduce confusion between similar letters. Future work should focus on collecting more high-quality images to train our model effectively.

6. Conclusion

Overall we are pretty happy with our result despite the low accuracy. We successfully developed a classifier for the STS using the deep learning model CNN. This model can classify static images of the STS and translates hand gestures into text in real-time. To improve our results in the future, we need a larger dataset with higher-quality images, which would help the model to better recognize the different gestures better but also to reduce confusion between similar letters.

References

[1] Raul Awati, *Convolutional Neural Network (CNN)*, 2023, Accessed: 2024-10-22.

[2] MK Gurucharan, *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network*, 2022, Accessed: 2024-10-23.

[3] Tomlinson, D., & Hinton, G, *Deep Learning for Hand Gesture Recognition in ASL*, 2015, Accessed: 2024-12-01.

[4] Zhang, X., et al, *A Large-Scale Dataset for American Sign Language Recognition*, 2019, Accessed: 2024-12-01.