SN Computer Science (2021) 2:207
https://doi.org/10.1007/s42979-021-00612-w

ORIGINAL RESEARCH

Interpretation of Swedish Sign Language Using Convolutional Neural Networks and Transfer Learning

Gustaf Halvardsson[1] · Johanna Peterson[1] · César Soto-Valero[1] · Benoit Baudry[1]

Received: 21 October 2020 / Accepted: 22 March 2021 / Published online: 11 April 2021

Abstract
The automatic interpretation of sign languages is a challenging task, as it requires the usage of high-level vision and high-level motion processing systems for providing accurate image perception. In this paper, we use Convolutional Neural Networks (CNNs) and transfer learning to make computers able to interpret signs of the Swedish Sign Language (SSL) hand alphabet. Our model consists of the implementation of a pre-trained InceptionV3 network, and the usage of the mini-batch gradient descent optimization algorithm. We rely on transfer learning during the pre-training of the model and its data. The final accuracy of the model, based on 8 study subjects and 9400 images, is 85%. Our results indicate that the usage of CNNs is a promising approach to interpret sign languages, and transfer learning can be used to achieve high testing accuracy despite using a small training dataset. Furthermore, we describe the implementation details of our model to interpret signs as a user-friendly web application.

Keywords Sign language interpretation · Machine learning · Convolutional neural networks · Transfer learning

Introduction

In 2018, it was estimated that 466 million people worldwide had a disabling hearing loss [1]. When a person in a family turns deaf or is born with impaired hearing, several problems might emerge [2]. In particular, deaf people who often use sign language to communicate are in many cases dependent on interpreters when, for example, seeking care. People who need to use sign language are often unable to communicate effectively with people who are not familiar with sign language [3]. In this context, an application that automatically translates sign language is beneficial since it can improve deaf people's quality of life, especially in terms of increased social inclusion and individual freedom.

The problem is that, just as with spoken languages, all sign languages differ. There is no global sign language shared over the world [3]. Therefore, a generic translating solution is not enough to address this problem for all deaf people in the world. To the best of our knowledge, there is today no application to help them interpret generic sign language to text. If there would be only one, for example, a tool interpreting only American Sign Language (ASL) would not work on Swedish Sign Language (SSL). Therefore, a generic software application based on a solution that can easily be adjusted for interpreting many different sign languages would be a preferable solution. This solution should work automatically, translating sign language to text independent of the physical characteristics of the user.

An application to interpret sign language could benefit from novel Artificial Intelligence (AI) techniques. AI is the science and engineering of building intelligent machines that can be fed raw data to learn on. The machines can make decisions in situations that they have not encountered before. Machine Learning (ML) is a large sub-area of AI that specializes in recognizing patterns in data to continuously learn from feedback [4]. One commonly used ML architecture is neural networks. A neural network is built up of several layers of artificial neurons [5] that try to mimic the function and behavior of biological neurons [6]. Each layer of neurons specializes in detecting different features. One layer could, for example, learn to detect edges when analyzing images.

"This article is part of the topical collection "Recent Trends in Computer Vision" guest edited by P. Nagabhushan, Balasubramaniyan Raman, Satish Kumar Singh and Subrahmanyam Murala"

✉ César Soto-Valero
cesarsv@kth.se

[1] KTH Royal Institute of Technology, Stockholm, Sweden

Since the differences between signs in sign languages largely consist of different patterns like hand movements and shiftings, using artificial neural networks that learn from datasets of sign images is useful to develop a model able to detect and interpret signs [7].

Training a neural network to perform this type of image recognition adequately requires a large amount of data [8, 9]. Regarding sign language, the only datasets available are based on the Sign Language MNIST image dataset,[1] which is based on the American Sign Language.[2] There are none based on the Swedish Sign Language (SSL), and thus, not enough SSL data are available to train a model on. While a few databases of SSL exist (i.e., the SSL dictionary provided by Stockholm's University[3]), their data only include one or two examples per sign. This is not sufficient to train an accurate ML model. A solution to the problem of having a small amount of data is to use a pre-built and pre-trained model and applying transfer learning [10]. Transfer learning is a technique that uses models trained on one quantitatively large dataset, and then the first layers of this model are reused to personalize new models based on a set of limited and specific data.

Several studies on sign language interpretation are based on AI solutions, as well as AI solutions with transfer learning. One study focuses on gesture recognition using a CyberGlove and has received an accuracy of up to 100% [11]. Even though this solution has resulted in high accuracy, using a CyberGlove device might not be possible at all times, and therefore, a solution based on using only a camera can be more adaptable. Another study using AI generated a dataset using YouTube videos but did not use transfer learning [12]. This made their model highly dependent on their dataset and might not generalize well to new situations. A study focusing on using both AI and transfer learning, however, did not receive higher accuracy than 66% due to a too small dataset (1200 images) [13]. Based on these studies, generating a larger dataset to use with transfer learning, as well as accurately interpret sings using a single camera, is the focus of this paper.

In this paper, we investigate to what extent Convolutional Neural Networks (CNNs) and transfer learning are effective techniques to interpret the hand alphabet of SSL. To find the network architecture with the highest validation accuracy, three different pre-trained models, two optimization algorithms, and three values of the number of frozen layers and step-size are tested. Our final network structure is based on the pre-trained model InceptionV3, the optimization algorithm of mini-batch gradient with a step-size factor of 1.2, and with five frozen layers. The final model consists of 25 488 698 parameters and 316 layers. Based on this model, we build an application that translates the signs into their corresponding text form. The final network has a testing accuracy of 85%.

In summary, this paper makes the following contributions:

- A CNN that translates images from the SSL hand alphabet;
- An original approach based on transfer learning, which adapts the model to perform well using a limited training dataset;
- An evaluation of the model on 8 study subjects, obtaining 85% of overall accuracy;
- A publicly available implementation of our approach[4] and a web application where a static sign is entered as an input image by the user, and its corresponding text form is displayed on the screen.[5]

The rest of this paper is structured as follows. Section 2 presents a background of CNNs and transfer learning. Section 3 describes the methodology, the dataset employed, and the architecture of our CNN model. Section 4 presents the evaluation procedure and the results obtained when measuring the performance of the model. Section 5 discusses the challenges associated with learning from images when translating sign language. Section 6 presents the related work, and Section 7 concludes the paper.

Background

Image recognition is the technology of analyzing patterns in images to classify the image as a particular object [14]. An image recognition method generally includes four steps: (1) *image acquisition*: retrieves unprocessed images from a source [14] and defines class belonging for each image, these images and the corresponding classes will represent the dataset for the model; (2) *image processing*: performs processing on the image, for example, reduces the colour of the background and finally represents every digital image frame as matrices of pixels; (3) *feature analysis*: is the step where the method is chosen comes in at most in analyzing the features of the images [15], and finding patterns, and (4) *image classification*: classifies new, unseen, images to a class among the predefined classes.

There are several image recognition methods, e.g., statistical pattern recognition [14], fuzzy mathematical method [16], syntactic pattern recognition method [14], and

[1] https://www.kaggle.com/datamunge/sign-language-mnist.

[2] https://www.kaggle.com/grassknoted/asl-alphabet.

[3] https://teckenspraakslexikon.su.se/sok/handalfabetet.

[4] https://github.com/gustafvh/SignInterpreterSSL.

[5] https://sign-interpreter-ssl.herokuapp.com.

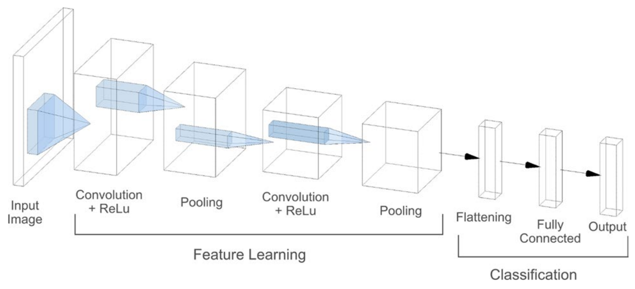


Fig. 1 General architecture of a Convolutional Neural Network. Starting from the left in the figure, the input image is split into several smaller matrices that are used as input to the first convolutional layer. This layer performs convolving and the method of Rectified Linear Unit. The next layer performs pooling on matrices. These two steps are repeated several times and comprise the feature learning in the network. The final part, classification, consists of the methods of flattening, fully connected, and finally, providing the output

CNNs [14]. These three methods have been widely used for image recognition. However, a promising line of methods used today are those that use Deep Learning (DL) and are based on CNNs [15]. The use of CNNs is particularly suitable for image recognition since they automate the process of feature extraction from the images efficiently.

On the other hand, conventional Machine Learning Models for computer vision rely on feature extraction techniques, which are often done using hand-crafted features based on color, edges, and corners. These features may be difficult to define, which often result in poor generalization and performance [17]. In CNNs, the feature extraction and data engineering are done automatically by the network itself, usually resulting in better performance for most computer vision tasks [18].

Another advantage of using CNNs, as opposed to the rest of methods mentioned, is the fact that the other methods include feature vectors extracted using algorithms made by researchers. The patterns are thus determined by the researchers and might not actually represent the real nature of the image. CNNs, on the other hand, does this extraction independent of the researcher, and can, therefore, decrease bias introduced by the researchers by deriving meaning from patterns too complex to be noticed by humans or traditional algorithms [19]. CNNs also outperform other image recognition methods available today [15].

Convolutional Neural Networks (CNNs)

CNNs are a class of neural networks that are particularly accurate when applied to image recognition tasks. The training of a CNN is performed using the backpropagation method [20]. The general structure of a CNNs is described in Fig. 1. The figure was made with the help of the tool NN-SVG[6]. As seen in the figure, CNNs perform feature extraction and classification. First, the input images are split into small matrices of pixels. The feature extraction consists of the two processes of convoluting (and ReLu) and pooling that is applied repeatedly several times [15]. This enables the possibility to recognize specific geometrical patterns, with little data to train on [20]. Further on, the classification is performed through the three-layer of flattening, full connection, and output. All of these steps are further described below.

Convolutional Layer. The convolutional layer consists of a layer of neurons, each connected to a filter, allowing different neurons to activate on different patterns in the images [20]. The filters perform dot multiplication with its assigned section of the input matrix, and then perform ReLu to increase non-linearity and reduce unwanted pixels. These values are put in a new matrix which is the new feature map that is transferred to the next layer [15].

Pooling Layer. The pooling layer takes a specific value from every small patch of the input image and places it in a new matrix. This layer downsizes the matrices and thus works as a regularizer for the network. This new matrix is the new pooled feature map that is transferred to the next layer [15].

Flattening, Fully Connected Layer, and Output Layer. When all the steps of feature learning have taken place, the final pooled feature map needs to be flattened by transforming the feature map matrix into a one-column matrix [21]. The penultimate layer of the network is the fully connected layer, which performs a calculation to classify the images into the predefined classes [20]. The final output layer consists of the probability of class-belonging to each class [15].

Transfer Learning

One problem with CNNs is that big networks need high GPU performance, which is often difficult to achieve on personal computers [22]. Without this, the learning will be slow. Another problem is the need for data. A possible solution for this is using a pre-trained model with the technique of transfer learning [23]. Transfer learning leverages knowledge from one source to improve learning on another [23]. In the following, we describe the general method and the concept of a pre-trained model, together with several models considered for this paper.

General Method. Transfer learning firstly uses a pre-trained DL model on a problem [10]. It does not have to be based on the same type of input data as the new source [23]. However, the performance of the new network will vary depending on which pre-trained model is used. The first layers

[6] http://alexlenail.me/NN-SVG

Table 1 The eleven pre-trained models available from Keras considered in this paper for transfer learning. All models are trained on the ImageNet dataset. Each model is presented with its accuracy on the ImageNet dataset, and the number of parameters in the pre-trained model. The table is sorted on the highest accuracy

Model	Accuracy [%]	Parameters
InceptionResNetV2	80.3	55,873,736
Xception	79.0	23,910,480
InceptionV3	77.9	23,851,784
ResNet50V2	76.0	25,613,800
DenseNet121	75.0	8,062,504
ResNet50	74.9	25,636,712
NASNetMobile	74.4	5,326,716
MobileNetV2	71.3	3,538,984
VGG16	71.3	138,357,544
VGG19	71.3	143,667,240
MobileNet	70.4	4,253,864

from this network are then frozen and put in front of new layers that have not been trained on any data. The learned parameters from the pre-trained source are saved as a vector $\boldsymbol{\theta} = \left\{\theta_1, \theta_2, ..., \theta_n\right\}$ which is transferred to the new model together with new, specific data, for the model to train on. Transfer learning eliminates the need to train the entire network by transferring the knowledge of the pre-trained model and, thus, reducing the need for large quantities of data.

Pre-Trained Models. The pre-trained models considered for this paper are all available for use with Keras[7], which is the DL API that is used for this paper. All models are specialized in image classification, trained on the dataset ImageNet[8].

Table 1 presents the models considered in this paper. The first column shows the name of the model. The second column presents the models' accuracies on the ImageNet dataset. The accuracy is an important indicator to take into consideration when choosing the model since the better the model performs on the ImageNet dataset, the better starting conditions for the new model [24]. The third column in the table presents the number of parameters representing the model. The higher this number is, the more time and space it will take to train the network. Too many parameters can lead to a slow and memory-expensive process on modern computers. However, too few parameters will likely make the pre-trained model less fit for use on new data.

Evaluation Criteria for CNNs

Evaluating an ML model is about finding the difference between the predicted output by the model and the actual output [25]. This defines the model's accuracy. Two metrics are tested in the paper to evaluate which one of them performs best on the data. The metric of misclassification error is used to maximize accuracy. The metric of using a loss function is used to minimize the loss on the data. These two metrics are chosen since they are commonly used by other researchers [20, 21, 23, 26].

The metric of misclassification error is calculated as the average number of correct classifications [27]:

$$err(f, D) = \frac{1}{N} \sum_{i=1}^{N} lnd(f(\mathbf{x_i} \neq y_i),$$

where $lnd(f(\mathbf{x}) = 1)$ if x is true, otherwise $lnd(f(\mathbf{x}) = 0)$. The metric of using a loss function is commonly used on image recognition problems based on a DL network [15]. A distance is here the distance between the feature vectors of the current pattern and the input image. The distance between two images of the same object is considered small, whilst the distance between two images of different objects is considered large.

The two metrics are evaluated at the beginning of the testing process before the more specific details of the network is tuned, and the metric producing the highest accuracy on the data is the one used for the rest of the paper.

Materials and Methods

The modeling, as presented in Figure 2, consists of several activities, all of which include quality assurance. The first activity, highlighted in blue, is to handle the pre-trained model. The model is imported and the last classification layer is detached from it. The first layers of the pre-trained model are frozen to keep its knowledge. Three convolutional layers are then added at the end of the network. At the same time as this, the SSL dataset is generated, highlighted in red in the figure. This includes steps of image acquisition and image processing. Further on, the model is retrained, yellow color in the figure, on the pre-trained model and the new training dataset. This model is then tested and its accuracy is improved in several cycles, as presented in green in the figure. The focus of the tuning of the network is to find the combination of pre-trained model, optimization algorithm, and tuning of hyper-parameters, with the highest resulting validation accuracy. Finally, the final network is evaluated using the testing dataset, highlighted in orange in the figure. The rest of this section is dedicated to these five phases of the methodology.

Pre-trained Models

The first step of modeling is to choose and integrate a pre-trained model. A pre-trained model is used as the basis of

[7] https://keras.io/applications

[8] http://www.image-net.org

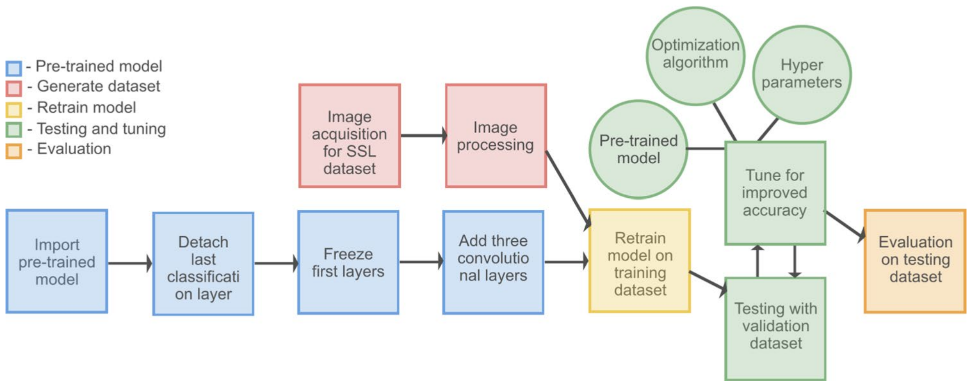

- Pre-trained model
- Generate dataset
- Retrain model
- Testing and tuning
- Evaluation
Image acquisition for SSL dataset
Image processing
Optimization algorithm
Hyper parameters
Pre-trained model
Tune for improved accuracy
Import pre-trained model
Detach last classification layer
Freeze first layers
Add three convolutional layers
Retrain model on training dataset
Testing with validation dataset
Evaluation on testing dataset

Fig. 2 General overview of our approach, including building the network and generating the dataset. The first activities include changing the imported pre-trained model. Simultaneously the dataset is generated. Following this, the new model based on the pre-trained model is trained using the training dataset. The model is then retrained, and the accuracy is improved in several cycles focusing on choosing the combination with the highest validation accuracy. Lastly, the final network is evaluated using the testing dataset

the program to make better predictions on a small dataset. Another alternative of using the architecture of pre-trained models as the base, is to utilize few or one-shot learning techniques [28], such as Siamese Neural Networks, which are best suited when samples per class are extremely low (fewer than 10). Thanks to the multiple volunteers combined with the video tool for training data capture, the number of samples per class was high enough to consider using a conventional CNN structure with transfer learning that generally achieves better performance if the data requirement is met [29].

The data for the pre-trained model are already collected, processed, and the model is trained on it. There are, thus, several variables, such as the type and characteristics of images sued for training the model, that cannot be controlled. This is the part of the data collection that is based on experiments since the pre-trained models considered all are based on a large dataset. The pre-trained models used in this research are the ones presented in Section 2.2. They are trained on the ImageNet dataset.

Since training the models is a very time-consuming task: approximately three hours per training session on Colabs GPUs, three out of all of the pre-trained models presented in Table 1 are tested. Table 1 presents eleven pre-trained models that are available via Keras and ordered by their accuracy on the ImageNet Dataset. The three pre-trained models used are decided upon based on the highest accuracy received on the ImageNet dataset. All three included the Inception-module in the network, which suited this studied problem type. This was due to the variable filter size of the Inception module, which allows the model to identify signs despite large differences in its spatial size in the input, something very common for this study's application [30, 31]. Thus, InceptionResNetV2, Xception, and InceptionV3 are tested. The process of deciding which one of these three models to use in the final network is presented in Section 3.5.

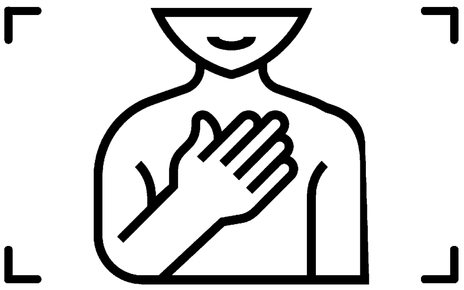


Fig. 3 Illustration representing how the signing videos are staged regarding the position of the person's body and hand

The pre-trained models are imported as a pre-existing module through Keras and are then possible to use in the code when testing the model on data. The last classification layer is detached from the model, and the 20 first layers are frozen to keep that knowledge. Finally, three convolutional layers are added to specialize in the new data. The process of choosing and using pre-trained models is further described in Section 3.5.

Swedish Sign Language Dataset

The next step is to collect and process the data of SSL with a focus on the generation of a generic dataset. Image acquisition and image processing are conducted in this step of the process. Feature analysis and image classification are performed in a subsequent step when the model is retrained and tested on the new dataset.

Image Acquisition

The first step, image acquisition, is performed by filming all the letters in the hand alphabet for 20 seconds, for a total of eight times by five subjects. All 26 letters are recorded eight times, and every recording is performed signing with the person's right hand. The recordings are performed as presented in Figure 3. The figure is created using assets from Flat Icon[9]. Only a part of the body and face are included, the person is centered, and their hand is centered in front of the chest. The webcam used is the built-in FaceTime HD camera from Apple Macbooks, which is rated at 1.2 megapixels and records 720p video in 30 or 24 frames per second.

The eight recordings are the basis of the training, validation, and testing datasets. The signs are based on the SSL dictionary provided by Stockholm's University. The conditions between the recordings are varied by background and clothes. When filming, the hands are slightly rotated to allow for more variance. Each sign of a letter is then classified as belonging to that specific letter.

The datasets are collected by five different subjects. This part of the method connects to the consciousness of ethics in society and technology. This is one part of the investigation where the technology could become an inhibitor and not an enabler for sustainable development. By collecting diverse data, it could improve the ML models' ability to generalize and different types of signers and hands.

Image Processing

The second step, image processing, is now performed on each video. A function is created to read all videos per letter and create one frame per 0.1 seconds, so every recording becomes 200 images. These images are then resized to a size of 224 × 224 pixels. 100 of these images are placed in the training set, 50 in the validation set, and 50 in the testing set. All images are put into repositories on GitHub, cloned into the code, and then merged once on the Colab Virtual Machine to be used as data[10]. To prevent memory overflow, the images in the training dataset are split up into batch sizes of 32 that are sent into the model in batches.

The training images are then augmented to allow for more noise in the data. The testing images are also augmented to create a more realistic real-life testing experience. The steps performed for both datasets are rotation, shifting height and width, zooming, and tilting, to allow for more variations of signing. When shifting and tilting, the points now outside of the boundaries of the image are filled with the nearest pixel. The testing images are also altered on brightness. No image augmentation is performed on the validation dataset. The image augmentation is performed with the Keras class ImageDataGenerator[11].

Model Retraining

When the pre-trained model is imported and the new dataset generated, the training of the new model begins. This is done by adding extra layers. Three convolutional layers are added. The first two consist of 1024 nodes each, the third consists of 512 nodes, and they all use ReLu. These values of the numbers of nodes are chosen since they are frequently used in other studies and yielded satisfying results. The pooling layers use global average pooling. Finally, softmax is used to ensure that the probabilities end up between zero and one. The first 20 layers of the pre-trained model are frozen (model weights became immutable) in order to not retrain them but keep their knowledge. Then the model is trained on the new data with the new layers in order to specialize in the new data. This is the step of feature analysis in the image recognition framework. The final step of image classification then occurs when testing the model and finally towards the end of the implementation when using the application.

Model Testing

Before starting to improve the accuracy of the model, the evaluation method needs to be decided upon. This focused on the research's approach, deductive, which ends with specific accuracies of the model's performance. As the model used is a neural network, the outcome is a generalization based on the collected data over several runs explaining the results as relationships between several variables.

The model is trained on 50% of the data, validated on 25%, and tested on 25%. This split of the data is done on a per person and sign level, meaning the training, validation and test datasets each consist of an even distribution of each sign and person. The testing data, therefore, contains samples from every person and sign and consists of 25% of the full dataset. The evaluation of accuracy on the validation data is performed both through the loss function and the total misclassification error on the validation data. The models are non-deterministic, and thus each run of the model generates a slightly different accuracy. Each run is repeated three times to get an average accuracy. Several training sessions on ten epochs, with different parameters, are tested,

[9] Figure 3 uses assets from the icon made by Eucalyp from https://www.flaticon.com

[10] https://colab.research.google.com/drive/1oxKUHDykfQOsG0VpP_pblMf9waEy8pq7?usp=sharing

[11] https://keras.io/api/preprocessing/image/#image-data-preprocessing

as will be presented in Section 3.5. These executions are used when comparing different networks. Therefore, the differences between the networks are of importance, not their absolute performance. To present the final accuracy of the model, the final network architecture is trained during 30 epochs. This run is then tested on the testing data to present the final accuracy.

A dynamic video tool is developed to visualize the model's performance on single signs. This is also developed to be able to detect several signs in a row and help evaluate the different models' accuracies and behavior in real-life situations and with fast feedback. It works by making predictions continuously on every frame from the webcam video stream and showing the result live. Basic helper functions, including backspace, delete, and reset, are also added. The tool is only developed to be run locally for testing purposes as deploying that functionality on the web is out of scope for this investigation.

Accuracy Improvement

One part of the research that focused on the experimental aspect of our methodology is the choice of the pre-trained model to use, the optimization algorithms, and the tuning of hyper-parameters. The methodology for improving the model's accuracy is conducted in two steps. First, the InceptionResNetV2, Xception, and InceptionV3 pre-trained models are tested with different optimization algorithms. Second, two hyper-parameters are tuned based on the combination of pre-trained model and optimization algorithm that has the highest accuracy from the previous step. In this section, we give details regarding those improvements.

Model Pre-training and Optimization

The three pre-trained models are then tested on two optimization algorithms. Due to the same reasoning as above, not all algorithms are tested. The two methods tested are mini-batch gradient descent and Adam since they are commonly used by other researchers. Thus, all three pre-trained models are tested on the two optimization algorithms. The tests are conducted using a fixed batch size for the optimization algorithm of 32, and on ten epochs. Based on these tests, the architecture with the highest accuracy is chosen, according to the Wilcoxon signed-rank test [32]. This test is appropriate to use since it can determine, with statistical significance, if one model combination can be used over another based on its overall performance. The combinations of different pre-trained models are used in the tests to determine if the models perform as an equal distribution or not and, thus, which validation accuracies are the most reliant. The final combination with the most statistical significance is chosen as the final combination. With the final combination determined, the tuning of the rest of the parameters is performed.

Hyper-parameters Tuning

The tuning of hyper-parameters begins with one combination of pre-trained model and optimization algorithm. This is done with the help of the tool Weights & Biases [33]. It allows sweeping of several parameters to help find the parameter values that maximize the performance of the network. The following two parameters are tuned: the number of frozen layers of the pre-trained model, and the step size factor. The step size factor is a factor that determines the relationship between the total number of training data examples. The number of frozen layers determines the impact the pre-trained model should have on the final outcome. The step size factor is tested on three values, namely 0.2, 0.7, and 1.2. Furthermore, the number of frozen layers is tested on three values, namely, 5, 20, and 50.

Experimental Results

The goal of the experiments is tuning a network with the highest resulting validation accuracy, which allows the network to interpret as many words as possible correctly. Figures 4 and 5 show the web application's architecture and the user flow for using it, respectively. The objective of each experiment is, thus, to find the combination of parameters that provide the highest accuracy.

All code and data for this investigation are stored in three different GitHub repositories, where each repository contains a `README` file with more information on its content. The code for the model, data generator, and front end for the application can be found in one GitHub repository.[12] The code for the back end API for the application is stored separately and can be found in another GitHub repository.[13]

Our dataset is collected using five different subjects, in a total of eight recordings. Figure 6 presents one representative image per person, or recording, that is included in the dataset. More information about each recording is shown in Table 2. Two factors that might change how an image of a signer looks visually are age and gender which is why it is important to have a varied selection of this in the dataset. Signers included people with different ethnicity, different ages (between 22 and 53 years old),

[12] The following link contains the GitHub repository for the model, data generator, and front end for the application: https://github.com/gustafvh/SignInterpreterSSL.

[13] The following link contains the GitHub repository for the back end API code: https://github.com/gustafvh/SignInterpreterSSL_API.

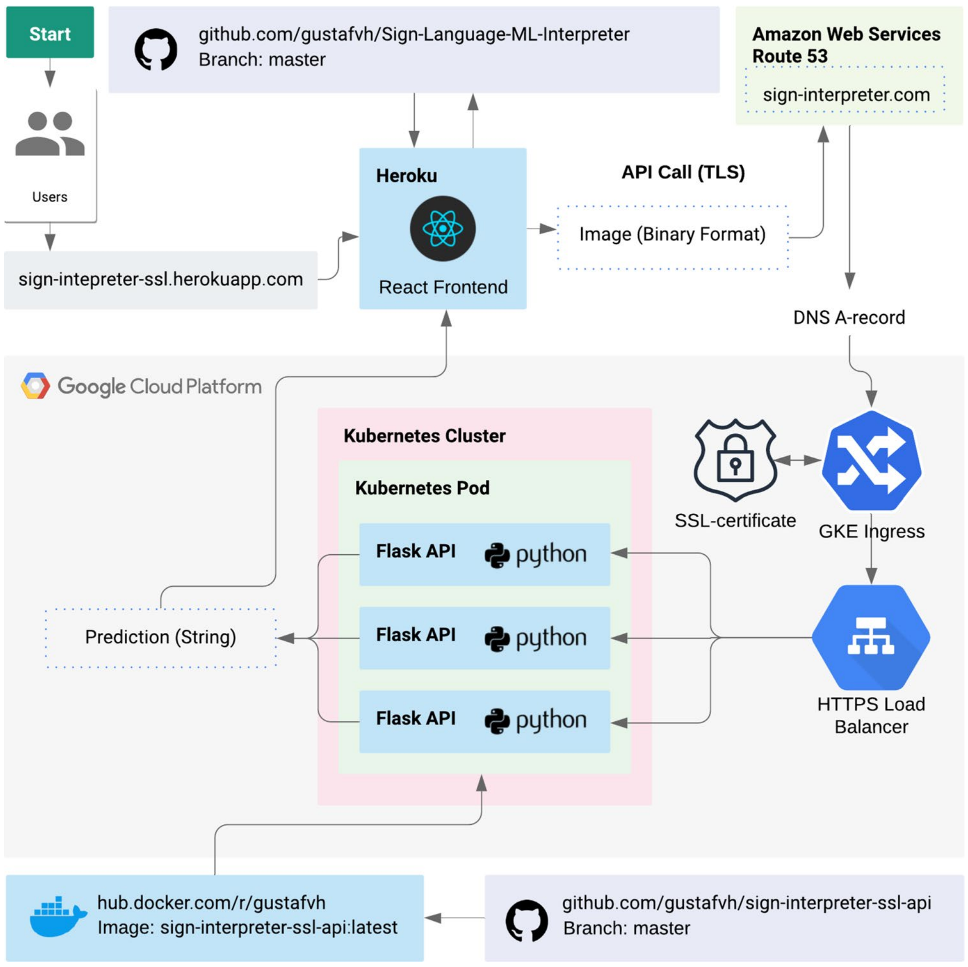


Fig. 4 Cloud architecture of the web application. The arrows represent the flow of data for an API request when a user wants to analyze a sign. Starting at the top left corner, the user enters the URL in a browser to the Heroku-hosted application, which pulls its source code from GitHub. The front end (client) makes a POST request with the sign image now binary encoded, to the sign-interpreter.com domain, which points to the cloud services. That entry point is an ingress that validates the identity with a certificate to be able to handle the HTTPS protocol. The request is then forwarded to the load-balancer, which sends it to one of three Flask Python back ends that return a prediction to be returned to the client and displayed in the application. The Python back ends are based on Docker images living on Docker Hub and GitHub

different genders (Male and Female), and different visual background conditions during recording This is to ensure the model becomes better at generalizing and as signer independent as possible which could result in an improved performance. As previously mentioned, the exact number of images each person contributes with depends on how long each recording of each sign is and, therefore, varies slightly.

The testing data are generated from the recordings of these persons (but never overlaps the training or validation data) and are later further augmented to make them even more significantly different to the training and validation-data. These image augmentations include rotations, width and height shifts, shearing, zoom and brightness alterations to ensure the model is fairly evaluated using new and previously unseen data.

Accuracy Testing

This section focuses on the tests conducted to find the tuning of the network with the highest resulting accuracy. This is performed in two steps, first with three different pre-trained models and two different optimization algorithms, presented in Section 4.1.1, and then by tuning the hyper parameters, presented in Section 4.2.

Optimization Algorithm and Pre-Trained Model

These first tests are conducted using two different metrics as stated in Subsection 2.3. First, the parameter for using a loss function is activated, aiming to minimize the validation loss. Second, the metric of misclassification error is used, aiming to maximize the validation accuracy. Minimizing the validation loss results on average in 20% lower accuracy on the data. Therefore, maximizing the validation accuracy is used for the rest of the investigation.

The results of the tests of the pre-trained models and optimization algorithms are presented in Table 3. The table shows that the differences in accuracy between the different combinations are small. Thus, the Wilcoxon signed-rank test is used to validate which combination proves to perform best on the data.

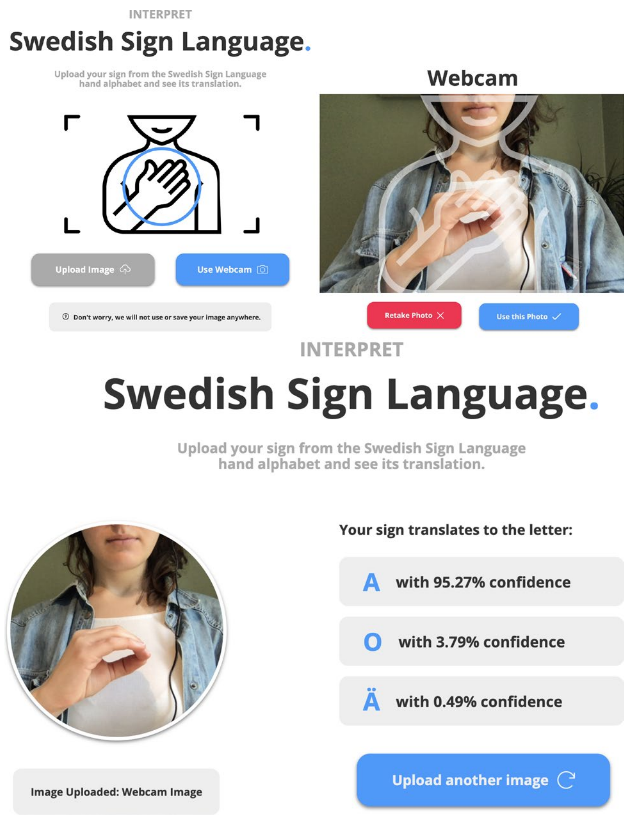

INTERPRET
Swedish Sign Language.
Upload your sign from the Swedish Sign Language hand alphabet and see its translation.
Upload Image
Use Webcam
Don't worry, we will not use or save your image anywhere.
Webcam
Retake Photo
Use this Photo
INTERPRET
Swedish Sign Language.
Upload your sign from the Swedish Sign Language hand alphabet and see its translation.
Your sign translates to the letter:
A with 95.27% confidence
O with 3.79% confidence
Ä with 0.49% confidence
Upload another image
Image Uploaded: Webcam Image

Fig. 5 User flow of the web application when using the webcam to take a picture. The first image presents the home page where a button for using your webcam is placed on the bottom right. The second image presents the page of the webcam where the user can take, and re-take a picture. The final image presents the results page that shows the sign with the highest confidence on the top

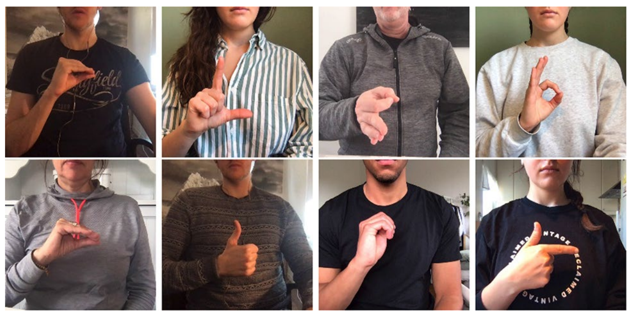


Fig. 6 One image per recording in the dataset for SSL. The images are shown in the same order as the recordings shown in Table 2. The images represent the letters A, L, P, H, A, B, E, and T

Statistical Validation

Since the runs with InceptionResNetV2 produce significantly lower accuracies than the other pre-trained models, they are excluded from the Wilcoxon test. However, Xception and InceptionV3 produce similar results, and thus, the

Table 2 Information about the eight recordings included in the data generation. Each row represents a recording. The columns represent the person's age, gender, the total number of generated images as well as the total time of video material for that recording. The eight recordings consisted of five different subjects, meaning recording one and six, as well as two and eight, were done by the same two persons respectively

Rec.	Age	Gender	#Imgs	Rec. Time (s)
1	22	Male	5 191	519
2	23	Female	5 200	520
3	53	Male	5 200	520
4	23	Female	4 401	440
5	53	Female	5 200	520
6	22	Male	5 077	508
7	24	Male	5 200	520
8	23	Female	5 134	513

test is used to see if there are any statistical significance in the data that could be used to determine which combination performs better.

The first test keeps Xception static while altering the two optimization algorithms. The second test keeps InceptionV3 static while altering the two optimization algorithms. The data used for the tests are the validation accuracies received per epoch of training for the different combinations. The null hypotheses for both tests are that the combinations are of the same distribution.

The p-value of the test with InceptionV3 is approximately 0.0069, and thus, the null hypothesis can be rejected at a confidence level of over 99%. This meant that with InceptionV3, the optimization algorithms are not of the same distribution with a certainty of over 99%. This means that the combination that provides the highest accuracy can be used. Since mini-batch gradient descent produces higher accuracy than Adam for InceptionV3 as shown in Table 3, it is passed on to the next step of the evaluation. Approximately, the same reasoning can be used regarding the test with Xception since it produces a p-value of approximately 0.0469. Its null hypothesis can also be rejected, however, here at a confidence level of over 95%. Since Adam performs a higher accuracy than mini-batch gradient descent as shown in Table 3, it is passed on to the next step of the evaluation.

The two final combinations to evaluate are, thus, InceptionV3 with mini-batch gradient descent, and Xception with Adam. As seen in Table 3, they produce a validation accuracy of 94.78% and 94.77% respectively. Thus, further tests are needed to show which performs better on the data. To evaluate this, a box plot is used to show the variance and consistency in validation accuracies for the different epochs.

Table 3 Result, measured in accuracy, of the testing of pre-trained models and optimization algorithms. The columns correspond to the three pre-trained models tested, and the rows correspond to the two optimization algorithms. The results were obtained by basic layers added to the pre-trained models, and the new Swedish Sign Language data set

Optimization algorithms	Pre-trained models		
	InceptionResNetV2	Xception	InceptionV3
Adaptive Moment Estimation (Adam)	88.51%	94.77%	91.77%
Mini-batch gradient descent	87.98%	91.63%	94.78%

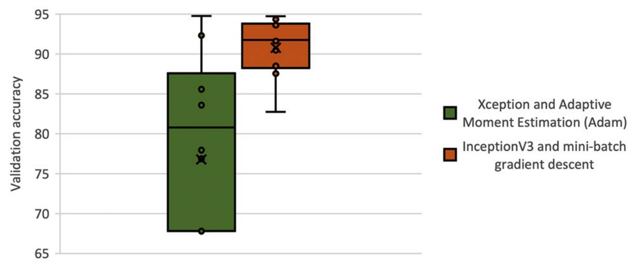


Fig. 7 Box plot of the validation accuracies of the two combinations, Xception and Adaptive Moment Estimation (Adam), as well as InceptionV3 and mini-batch gradient descent, varied over the ten epochs. The boxes include 50% of the data points and, thus, shows the main distribution. The lines over and under the boxes represent the highest and lowest validation accuracies received for the different combinations

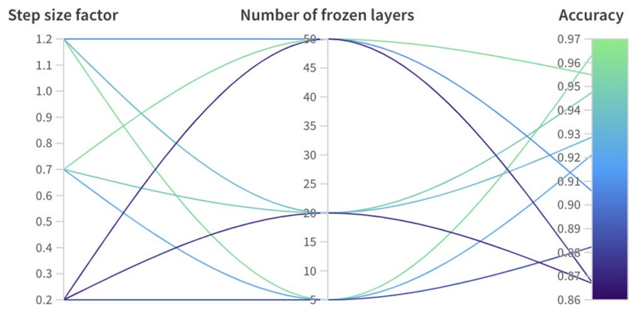


Fig. 8 Results from the tuning of hyper-parameters based on InceptionV3 and mini-batch gradient descent. Two hyper-parameters are tuned, the step size factor, and the number of frozen layers of the pre-trained model. The graph includes nine different combinations of the hyper-parameters and presents the combinations' validation accuracies after training. The figure is made automatically by the tool Weights & Biases [33]

Evaluation of Spread and Consistency of Validation Accuracy

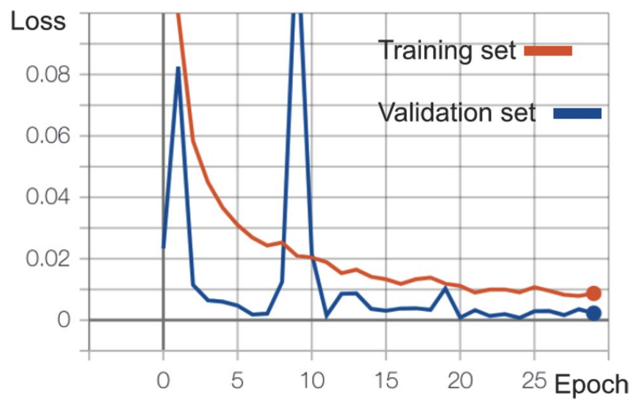
As seen in Fig. 7, InceptionV3 with mini-batch gradient descent has a low spread in the validation accuracies per epoch and produces consistently high accuracies. Xception with Adam, however, show a large variance in the results as well as consistently lower accuracies than the other combination. This shows that Adam is more prone to get stuck in local minima. Thus, even though the two combinations perform equally as stated in Table 3, InceptionV3 and mini-batch gradient descent are shown to be more reliant since the validation accuracies per epoch are more consistent. This increased reliance is an advantage since it simplifies the tuning of hyper-parameters as every run is more prone to deliver good results. This combination also increases the replicability of the experiments. Thus, the combination of InceptionV3 and mini-batch gradient descent is used as the basis for the final network architecture.

Hyper-parameters Tuning

The hyper-parameters for the network based on InceptionV3 and mini-batch gradient descent are now tuned. Two hyper-parameters are in focus: the step size factor, and the number of frozen layers of the pre-trained model. The results from these tests are presented in Fig. 8. The figure shows that the two combinations with the highest validation accuracy are with step size factor 1.2 and five frozen layers, and with step size factor 0.7 and 50 frozen layers. These two combinations are trained two times more each to find the average accuracy over several runs. This results in an accuracy of 96.46% for the combination of a step-size factor of 1.2 and five frozen layers, and an accuracy of 93.90% for the combination of a step size factor of 0.7 and 50 frozen layers. Thus the final values of the hyper-parameters are five frozen layers and a step-size factor of 1.2. The value of the step size factor, corresponds to a learning rate close to, but bigger than, the number of training data samples divided by the batch size.

Final Network Architecture

The accuracy tests end with one final network architecture with the highest validation accuracy. It consists of the following properties: the pre-trained model is InceptionV3, the optimization algorithm is mini-batch gradient descent, the batch size is 32, the step-size factor is 1.2, and the number of frozen layers of the pre-trained model is five. To come up with the final architecture, several decisions are made.



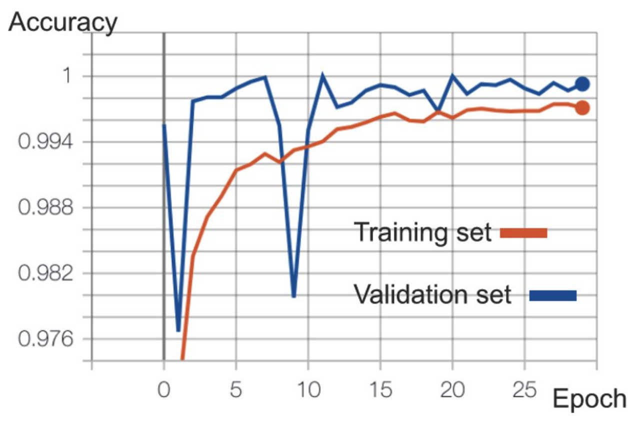


Fig. 9 Loss and accuracy of the training and validation set per epoch for the final model. The first figure presents the loss, and the second figure presents the accuracy of the training and validation set. The figures are made automatically by the Tensorboard tool in the tool Weights and Biases [33]

More specifically decisions regarding which pre-trained model, optimization algorithms and hyper-parameters to be tested and evaluated. First, only three pre-trained models are tested, based on the highest accuracy, as presented in Table 1. Then, mini-batch gradient descent and Adam are chosen as leads on optimization algorithms. Mini-batch gradient descent is chosen since it is a well-balanced combination of the basic approaches of Stochastic Gradient Descent and batch gradient descent and is more stable than the others to converge to the global minimum. Furthermore, Adam is chosen since it generally converges fast and is well suited for problems with a large number of parameters. The combination of InceptionV3 and mini-batch gradient descent is then chosen since it produces the most reliable results based on the Wilcoxon signed-rank test as presented in Section 4.1.1. Three values of step-size factor and number of frozen layers are tested. These numbers are based on the recommendation from Keras, and then one smaller number and one bigger. The hyper-parameter values of a step-size factor of 1.2 and five frozen layers are chosen since they performed better than other values on the combination of pre-trained model and optimization algorithm as presented in Section 4.2. The final network architecture is thus based on five frozen layers of InceptionV3. Thus, transfer learning is used, however, not to the extent that the network becomes too specialized on the pre-trained model and its data.

On top of the pre-trained model, three convolutional layers are added. The first two with 1024 nodes, the third with 512 nodes, and they all use ReLu as the activation function. The pooling layers use global average pooling. Finally, softmax is used to ensure that the probabilities end up between zero and one. To present the final accuracy of the network, it is trained one last time, this time during 30 epochs. The final model, as it is a CNN, consists of 25 488 698 parameters and 316 layers[14].

Figure 9 shows the training and validation loss and accuracy over the 30 epochs. As we observe in the figure, the training and validation loss soften after approximately twelve epochs. The overfitting that appears in epoch two and nine is eliminated after more epochs. The final model is then tested on the testing dataset. The final Top-1 testing accuracy is 85%.

The performance is also evaluated using the confusion matrix of the test dataset, presented in Fig. 10. We observe which classes (signs) the model performed better and worse on. As seen in the figure, the number of correct predictions were significantly less on the letter `P` (where 0.75 of labels were predicted correctly), `H` (.76), and `Z` (.78). For these cases, it can be observed that the model is finding these inputs more challenging, resulting in worsened performance. The reason is that these signs are very similar visually, and even more so due to the exclusion of certain parts of the hand. This explanation can also be observed in the particularly high performance of the visually distinct letter `V` (.90). Without a designated depth sensor, the model will have more problems with recognizing the signs more dependent on the z-axis (depth) due to exclusion.

The dynamic video tool that is used when testing the models' performances can be seen in Fig. 11. This particular picture is taken based on the final model.

Discussion

The delimitations can be constricted to those regarding the sign language and the model interpreting the signs.

[14] All layers and parameters for the network can be found at https://app.wandb.ai/sign-interpretor/sign-interpreter/runs/y11of78x/files/model-best.h5

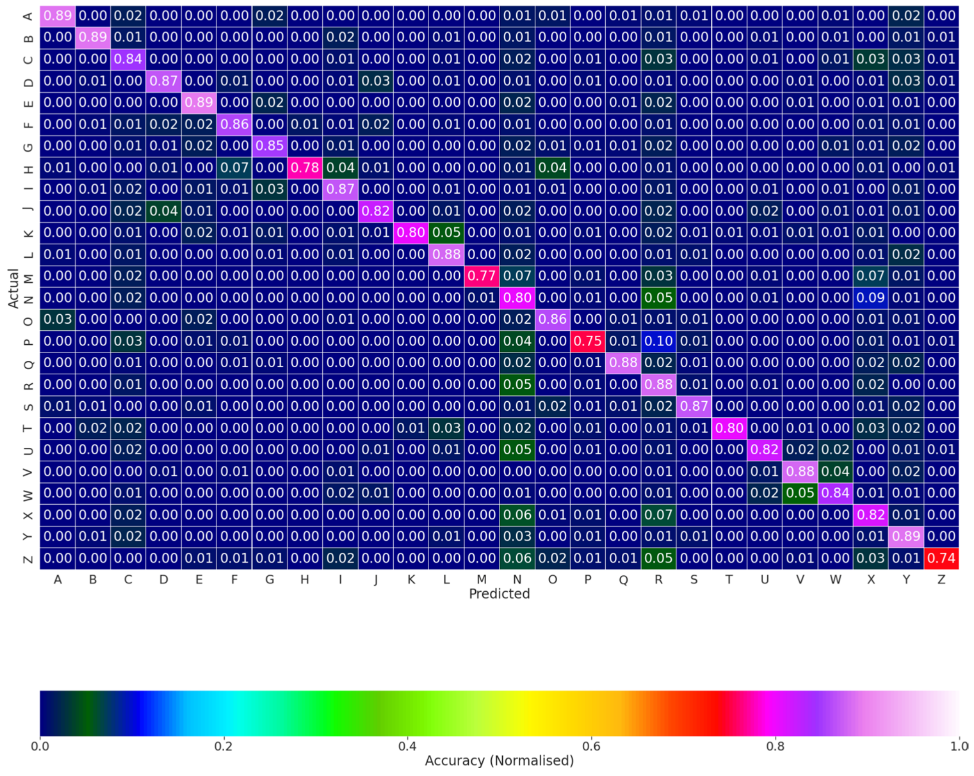


Fig. 10 Confusion Matrix showing the performance of the model in regards to each of its classes on the test dataset. The x-axis is the predicted values. The y-axis is the actual ground truth. Both axes correspond to the signed letter (class labels). The values are normalized, meaning the value in each box represents the percentage of instances when the predicted sign corresponded to the actual sign. Thus, a perfect model with 100% accuracy would only have 1.0 across the diagonal, and all remaining cells being 0.0

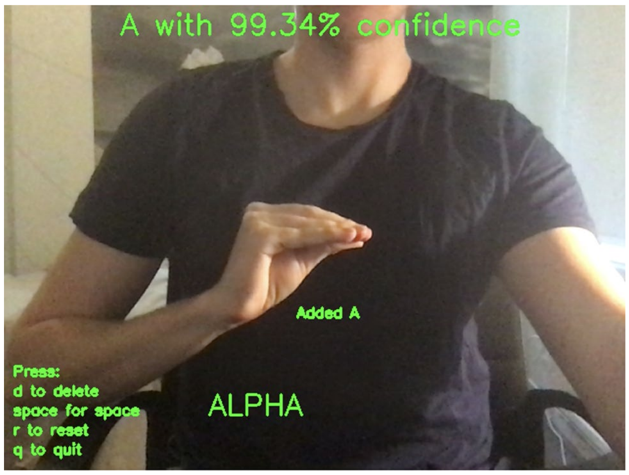


Fig. 11 A caption of the video tool used for testing. This picture presents how a list of letters was signed and added to a word. The bottom left of the picture shows the basic helper functions included to be able to sign words

Sign Language Delimitations

One delimitation regarding the sign language is that only the static signs of the SSL hand alphabet is used. Words, and four of the letters, in SSL are dynamic. They consist of several signs conducted in a specific order after each other. The remaining 25 letters in the hand alphabet consist of only one still hand gesture and are thus static. The four letters that are dynamic are: Y, Å, Ä, and Ö. Å is, for example, the same sign as A but moved around in a full circle. Any individual frame extracted from the sign Å would be interpreted as *A*. *Y*, is a sign shaped like a boat moved vertically down. Even though *Y* is dynamic, it can be interpreted statically since no other letter is shaped like a boat causing ambiguities, and *Y* is, therefore, included in the paper. Å, Ä, and Ö, on the other hand, will not be included in the paper.

When interpreting words and sentences, facial expressions are often involved when signing and this is not of focus in this paper. The focus is only on hand gestures which work well for the hand alphabet. Another factor when signing is that the room and objects around the signer are often used as a reference and commonly used to point at. This is not a critical factor for interpreting the hand alphabet, and thus, these references are not taken into consideration in this paper.

Model Delimitations

One delimitation regarding the model used to interpret the signs is the use of transfer learning. This paper focuses on using transfer learning for interpreting the hand alphabet of SSL. Thus, it does not necessarily suggest that the same thing can be done for the rest of the sign language or for the hand alphabets of other sign languages. Furthermore, the paper focuses on using transfer learning particularly, and thus other models with or without the basis of ML will not be tested. Finally, the dataset used for SSL is specifically developed for this paper based on us signing, and thus, other possible data sources will not be used.

Related Work

There have been several studies aimed at interpreting sign languages. Some of them have been based on gloves that can interpret signs, while others rely on computer vision and statistical comparisons. Furthermore, there are studies focusing on both SSL and CNNs. This section will present some of these studies. Section 6.1 focuses on studies performed without neural networks, while Section 6.2 focuses on studies with neural networks. Finally, Section 6.3 presents studies on sign language and transfer learning.

Sign Language Interpretation Using Human Supervision

Hern et al. [34] study the signer wearing an AcceleGlove when signing. In this work, the different angles per finger are used as input to a computer program that analyses the positions. It is tested on different subjects and is able to recognize 30 one-handed signs with an accuracy of 98%.

Glenn et al. [35] is based on computer vision and statistical comparisons. Each sign is filmed in a controlled environment, the background is extracted, the image is cropped, resized and edge detected, and finally placed in an adaptive statistical database. To classify a sign as a particular word, the image is processed and then compared to all images in the statistical database.

Akram et al. [36] used a Kinect sensor to recognize SSL signs. The signer uses a RGB-D Kinect sensor placed in the hand. This allows the backgrounds to be removed, helps with the resolution when the hand was placed in front of the face, and simplifies the use of 3D signs. The classification is done through a statistical database. The results are different depending on who signs the signs. One signer received an accuracy of 77% while one had 94%.

On the other hand, the work of Segundo et al. [37] is conducted with the opposite goal to this paper, to translate spoken words into sign language. The system uses speech recognition, a natural language translator, and a 3D avatar to show translations for Spanish Sign Language. It achieves a 31.6% Sign Error Rate. This significantly differs from this paper as one of the most challenging aspects of this paper is to correctly identify the sign from an image, which is not needed during this direction of translation.

Most of the related work presented includes several forms of assisting equipment, such as depth cameras and gloves, to simplify isolation and identification of the hands performing the signs. In this paper, the focus is on making an accurate interpretation possible without any specialized equipment besides a basic RGB-camera.

Sign Language Interpretation Using CNNs

The studies presented in this section are based on ANN. The benefits of using neural networks rely on their ability to derive meaning from patterns too complex to be noticed by humans or traditional algorithms [19].

Weis et al. [11] focuse on gesture recognition. The experiments are conducted with a CyberGlove, a glove with virtual reality sensors. The analysis is conducted by a multi-layered ANN, and the accuracy is close to 100% for some gestures. Pugeault et al. [38] present an interactive finger-spelling graphical user interface for ASL, which shows good performance and robustness for multiple users. Nagendraswamy et al. [39] present a model for learning symbolic representation of sign language from video shots, achieving sign recognition accuracy of 96% using a dataset of signs made by communication impaired people of Mysore.

Koller et al. [40] exploited the discriminative power of CNNs with application to hand shape classification in the scope of sign language. Moc et al. [41] perform a study that aims at recognizing a stream of continuous signs in a video. The computer architecture used is RNNs. The network has feedback connections that are suitable for video processing. The paper reaches an accuracy of 80% on a continuous stream of video data.

Quirk et al. [12] translate signs filmed with a webcam. The study aims to translate the Auslan Sign Language alphabet. The dataset for the Auslan alphabet is generated by extracting signs from YouTube videos and drawing boxes over the hands. They use CNNs and the final accuracy is 86%. This paper can utilize the fact that there were sign language instructional videos available on YouTube for the Auslan Alphabet and thus be used as data, which is something that is not currently available for SSL. Therefore, they do not utilize transfer learning to be less dependent on large datasets.

Cam et al. [42] focus on translating a continuous stream of sign language sentences. Specifically, they improve interpretations when it comes to grammar. The paper is built on CNNs and attention-based encoders and results in several sentences being correctly interpreted. The paper focuses more on aspects of grammar, something this paper chooses to have as a delimitation.

Sign Language Interpretation Using Transfer Learning

The studies presented in this section are based on ANN and transfer learning. The datasets used on the pre-trained models have all been limited.

Moci et al. [43] build a system to interpret British Sign Language. The datasets of British Sign Language used are a corpus for standard English with transcriptions to sign language and a pre-processed corpus called Penn Treebank. The videos from the corpora are split into sentences. The ML architectures used are both based on RNNs and CNNs. The study shows good results in words, but sentences are not interpreted grammatically correct. This study can also utilize the fact that a British corpus existed with more than just a few examples per sign (something not available for SSL). This eliminates the need to create new data for the transfer learning process.

Dhi et al. [13] use transfer learning to interpret Indian Sign Language. A 3D camera is used to help with depth interpretation, which differs from this paper as it aims to only use a basic RGB-webcam without that assistance. The data used are based both on images per sign but also depth images, which reduces the pre-processing time and also allows for better 3D processing. Further on, a pre-trained model based on the ImageNet dataset is used to increase the accuracy of the limited dataset. Then, several methods, including CNNs, are applied to the pre-trained model. The optimization algorithms AdaDelta and Adam are used. The model achieves an accuracy of 66%. However, when applying the pre-trained model, the accuracy becomes lower, and they conclude that they would need a larger dataset (>1200). Because of the lack of data, the authors therefore not conclude whether the use of transfer learning was successful, something this paper manages to do thanks to a larger dataset.

Conclusion

In this paper, we proposed an end-to-end machine learning model based on CNNs to translate images from the hand alphabet of SSL. We demonstrate that the problem of having a small dataset of SSL data is solvable using transfer-learning with a pre-trained model: our model is able to classify sign images with an accuracy of 85%.

Our approach of using transfer-learning to boost the model accuracy might be replicable to many other sign languages. However, using neural networks, the impact of human supervision is minimized and some patterns that are too complex for human's perception can be derived meaning from. However, the use of neural networks also adds an abstraction layer in which humans have no control in understanding or changing. This means that the algorithm might do things it is not supposed to or derives too much meaning out of a binary situation. In this regard, the use of more traditional methods must not be neglected on the basis of new and more complex techniques.

In the long run, this research could benefit deaf people who have access to technology and enhance good health, quality education, decent work, and reduced inequalities. Suggestions for future work include integrating dynamic signing data to interpret words and sentences, evaluating the method on another sign language's hand alphabet, and integrate dynamic interpretation in the web application for several letters or words to be interpreted in tandem.

Acknowledgements This work is based on the thesis of Peterson and Halvardsson [44], which was conducted in collaboration with Prevas AB, a Swedish technical IT consulting firm focusing on several areas of industry such as energy, defense, and life science. It has also been partially supported by the Wallenberg Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

Funding Open access funding provided by Royal Institute of Technology.

Declarations

Conflict of Interest The authors declare that there is no conflict of interest regarding the publication of this article.

References

1. World Health Organization. Addressing the rising prevalence of hearing loss. https://apps.who.int/iris/bitstream/handle/10665/260336/9789241550260-eng.pdf?sequence=1&isAllowed=y, 2018. [Online; accessed 30-April-2020].
2. Paul PV. What's it like to be deaf? reflections on signed language, sustainable development, and equal opportunities. American Annals of the Deaf Gallaudet University Press, 163, 2018.
3. Emmorey K. Language, Cognition, and the Brain: Insights From Sign Language Research. New Jersy: Lawrence Erlbaum Associates Inc; 2002.
4. Ray S. A quick review of machine learning algorithms. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), 2019.
5. Kaur T. Implementation of backpropagation algorithm: A neural network approach for pattern recognition. International Journal of Engineering Research and Development, 1, 2012.
6. Rosenblatt F. Principles of Neurodynamics. New York: Spartan; 1962.
7. Shi B, Martinez Del Rio A, Keane J, Brentari D, Shakhnarovich G, Livescu K. Fingerspelling recognition in the wild with iterative visual attention. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 5399–5408, 2019.

8. Atkinson PM, Tatnall ARL. Introduction neural networks in remote sensing. Int J Remote Sens. 1997;18.
9. Kim T, Keane J, Wang W, Tang H, Riggle J, Shakhnarovich G, Brentari D, Livescu K. Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation. Comput Speech Lang. 2017;46:209–32.
10. Liang H, Fu W, Yi F. A survey of recent advances in transfer learning. 2019 IEEE 19th International Conference on Communication Technology (ICCT), 2019.
11. Weissmann J, Salomon R. Gesture recognition for virtual reality applications using data gloves and neural networks. IJCNN'99. Int Jt Conf Neural Netw. 1999;3:2043–6.
12. Quirk T, Kamaal K. How we used ai to translate sign language in real time. https://blog.coviu.com/2018/09/21/how-we-used-ai-to-translate-sign-language-in-real-time/, 2018. [Online; accessed 30-April-2020].
13. Dhiman M. Sign language recognition, 2017.
14. Shu Y, Chen Y, Xiong C. Application of image recognition technology based on embedded technology in environmental pollution detection. Microprocessors and Microsystems. 2020;75.
15. Fujiyoshi H, Hirakawa T, Yamashita T. Deep learning-based image recognition for autonomous driving. IATSS Research. 2019;43:244–52.
16. Jovanović I, Miljanović I, Jovanović T. Soft computing-based modeling of flotation processes - a review. Miner Eng. 2015;84:34–63.
17. O'Mahony Niall, Campbell Sean, Carvalho Anderson, Harapanahalli Suman, Hernandez Gustavo Velasco, Krpalkova Lenka, Riordan Daniel, Walsh Joseph. Deep learning vs. traditional computer vision. In Science and Information Conference, pages 128–144. Springer, 2019.
18. Bui DT, Tsangaratos P, Nguyen V-T, Van LN, Trinh PT. Comparing the prediction performance of a deep learning neural network model with conventional machine learning models in landslide susceptibility assessment. Catena, 188:104426, 2020.
19. Stergiou C, Siganos D. Neural networks. urveys and Presentations in Information Systems Engineering (SURPRISE), 96, 1996.
20. Shanthi T, Sabeenian RS, Anand R. Automatic diagnosis of skin diseases using convolution neural network. Microprocessors and Microsystems. 2020.
21. Xu W, Zhang X, Yao L, Xue W, Wei B. A multi-view cnn-based acoustic classification system for automatic animal species identification. Ad Hoc Netw. 2020;102.
22. Ahmadvand P, Ebrahimpour R, Ahmadvand P. How popular cnns perform in real applications of face recognition. 24th Telecommunications Forum, 2016.
23. Shen S, Sadoughi M, Li M, Wang Z, Hu C. Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries. Applied Energy. 2020;260.
24. Canziani A, Paszke A, Culurciello E. An Analysis of Deep Neural Network Models for Practical Applications. arXiv:1605.07678, May 2016.
25. Hecht-Nielsen R. Neural Networks for Perception - Computation, Learning, and Architectures, chapter III.3 - Theory of the Backpropagation Neural Network, Elsevier Inc, 1992.
26. Christiansen NH, Torbergsen Voie PE, Winther O, Hogsberg J. Comparison of neural network error measures for simulation of slender marine structures. Journal of Applied Mathematics, 2014.
27. Maki A. Lecture 3 in course dd2421 machine learning.
28. Manjunath AVN, Mufti M, Basant A, Guru DS, Shamim . One shot cluster based approach for the detection of covid-19 from chest x-ray images. 2020.
29. Horiguchi S, Ikami D, Aizawa K. Significance of softmax-based features in comparison to distance metric learning-based features. IEEE Trans Pattern Anal Mach Intell. 2019;42(5):1279–85.
30. Christian S, Wei L, Yangqing J, Pierre SScott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew R. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
31. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.
32. L. Liu. Biostatistical basis of inference in heart failure study. *Heart Failure: Epidemiology and Research Methods*, pages 43–82, 2018.
33. L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
34. J. Hernandez-Rebollar, N. Kyriakopoulos, R. Lindeman. A new instrumented approach for translating American sign language into sound and text. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004.
35. Glenn CM, Mandloi D, Sarella K, Lonon M. An image processing technique for the translation of asl finger-spelling to digital audio or text. Rochester, NY: In Instructional Technology and Education of the deaf Symposium; 2005. p. 1–7.
36. S. Akram, J. Beskow, and H. Kjellstrom. Visual Recognition of Isolated Swedish Sign Language Signs. arXiv:1211.3901, November 2012.
37. San-Segundo R, Barra R, Córdoba R, D'Haro LF, Fernández F, Ferreiros J, Lucas JM, Macías-Guarasa J, Montero JM, Pardo JM. Speech to sign language translation system for spanish. Speech Commun. 2008;50(11–12):1009–20.
38. N. Pugeault, R. Bowden. Spelling it out: Real-time asl fingerspelling recognition. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pages 1114–1119, 2011.
39. Nagendraswamy HS, Guru DS, Naresh YG, et al. Symbolic representation of sign language at sentence level. International Journal of Image, Graphics and Signal Processing. 2015;7(9):49.
40. O. Koller, H. Ney, and R. Bowden. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3793–3802, 2016.
41. B. Mocialov, G. Turner, K. Lohan, and H. Hastie. Towards continuous sign language recognition with deep learning. 2017.
42. N.C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. Neural sign language translation. IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7784–7793, 2018.
43. B. Mocialov, H. Hastie, G. Turner. Transfer learning for british sign language modelling. In Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018), pages 101–110, 2018.
44. Halvardsson G, Peterson J. Interpretation of swedish sign language using convolutional neural networks and transfer learning, 2020.