A breakthrough in fingerspelling to text translation using an RNN

Zhouxin Ge 4272358 | Keith Klein 4298292 | Robert van Wijk 4313968 | Bram Harleman 4361105

1. Introduction

Nowadays, numerous applications of neural networks are popping up and are being improved upon. A domain that is less straightforward for the application of neural networks stems from a world that is less known to the most us. It is the world of people who cannot communicate using speech. These people use sign language. Sign language is a very important means of communication for people with a hearing disability [1]. Finger-spelling is part of learning sign language, it is mainly useful for spelling names and places. It might also be helpful for people who want to learn sign language in a fast and interactive way.

For the scope of this course, a research by four students will be performed on the implementation and performance of a convolutional neural network on classifying sign language letters. If the desired accuracy is met with the classifier, the team will implement an word predictor using a recurrent neural network on top of the classifier. In the analysis of the classifier's performance, insightful information of the hidden layers will be presented.

A possible application of this research is in teaching children to spell words and names of place by giving feedback on how to improve in having clear and distinct finger gestures.

In this article, firstly a description will be given on the two data sets that have been used in Section 3. Secondly, an elaboration on the technical approach of designing the classifier and predictor will be given in Section 4. Thirdly, the intermediate results and evaluation of the these results in Section 5. Fourthly, a reflection on the topic and approach will be given in Section 6.

2. Problem statement

An extensive research was performed on the implementation and performance of a convolutional neural network on classifying sign language letters. The overarching goal is to build a word predictor using a recurrent neural network on top of this classifier.

Hypothesis for the CNN: the classifier will have lower accuracy among the letters in the sets [A, S], [E, M, N] and [G,H] due to the similarity of the sign, see images in Figure 1 respectively. A filter should be learned and designed by tuning hyperparameters to accentuate the subtle differences between the letters.

Hypothesis for sequence modeling: the sequence modeling network will be able to predict subsequent characters with reasonable to high accuracy when the user tries to spell common words.

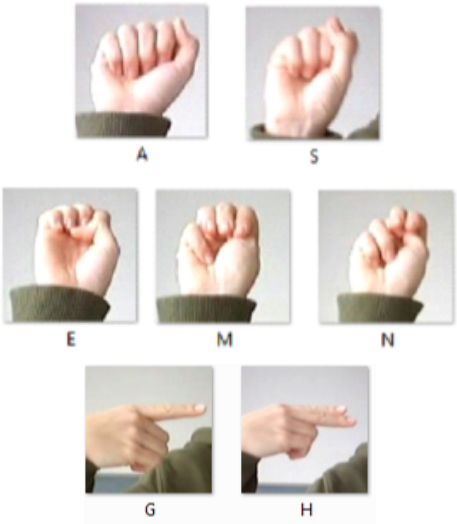

Figure 1. Some examples of ambiguous hand signs from which is hypothesized that these will be hard to distinct for the convolutional neural network.

3. Data

Two data sets have been used for the purpose of this article. One for the sign language letters in American Sign Language (ASL). The second data set is a list of words that are commonly used in the English language according to Google [4].

3.1. Sign language Data-set

For this project, the MNIST Sign Language data-set will be used [9]. The data-set is based on 1704 color images and data-augmentation is used to create a cropped, gray-scaled, and resized training set, consisting of 27.455 28-by-28 pixel images, and test set, consisting of 7.172 28-by-28

pixel images. The data is labeled from 0 to 25 for all the letters in the alphabet. There are no cases for labels 9 and 25 corresponding to the letters J and Z, respectively, since these letters require motion. Some examples of the data can be seen in figure 2.

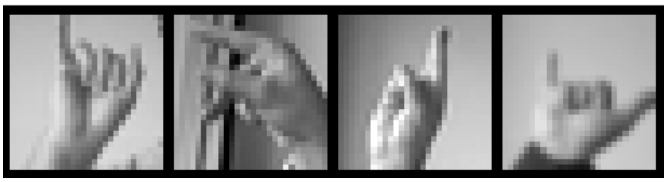

Figure 2. These figures show 4 examples of the data-set corresponding to the letters I, P, R, Y, from left to right respectively

3.2. Data-set used for sequence model

The data-set used to train the language model can be either a list of words or sequences of text from books or articles. There are a number of available data-sets of which the following are most applicable:

1. The data-set provided in [7] consists of over 466k English words.
2. The google-10000 data-set provided in [4] consists of the 10000 most common English words sorted by frequency of use.
3. The brown corpus data-set provided on the Kaggle website [2] consists of 1 million words of English text.

Since predictions will be done on the character level, a list of words is a more appropriate data-set. Texts are better suited for word-level predictions. The drawback of the first data-set is that the frequency of use for each word is not taken into account. This can lead to predictions that are less likely in everyday use. Therefore, the google-10000 data-set is the most suitable for this application.

4. Technical Approach

4.1. Classification

The data-set consists of images, therefore a convolutional neural network is used as a classifier. The starting point of the CNN's design was based on a PyTorch tutorial on CNN's this was soon after tailored to the needs of the research team. A common layout of a CNN architecture (as described by [3] and [5]) starts with a sequence of convolutional (Conv) layers and ReLu activation (normally not more than 3), followed by a pooling layer. This layer sequence is then repeated until the input image is reduced to a spatially small size. The output of this sequence is fed to a sequence of fully connected (FC) layers and ReLu activations (not more than 3), with the final FC layer being the output.

The architecture used in this classifier uses two Conv-ReLu sequences followed by a pooling layer, again two Conv-ReLu sequences followed by a pooling layer, and concludes with three FC-ReLu sequences. For each Conv layer a 3×3 filter is used because it is preferable to use smaller filters with more layers over larger receptive field with less layers [3]. Also, padding is set to 1 to let the pooling layers deal with shrinking the image. The consecutive Conv layers produce 12, 24, 48, and 96 feature maps, respectively. The three FC layers have 120, 84, and 26 nodes. The loss is calculated using Cross Entropy, and Stochastic Gradient Descent is used to do the backwards pass with a learning rate of 0.001 and a momentum of 0.9.

Until now only the architecture (playing with the amount of channels, adding / deleting layers), the batch size, and the amount of epochs were tuned to improve the accuracy of the network. The influence of changing the loss function, backward pass technique, learning rate or momentum has not yet been evaluated.

4.2. Sequence modeling

There are multiple ways to perform natural language processing. One way is to use n-gram modeling as discussed in [8]. Using this method, the probabilities of a group of 2 - n arbitrary letters is determined according to some data-set. These probabilities can then be used to make predictions based on the likelihood of a certain letter combination. n-grams are very efficient to implement and even perform well on small data-sets.

It is also possible to use a recurrent neural network (RNN) to perform character level predictions. The main advantage of using an RNN is its ability to learn longer sequences compared to an n-gram model. The drawback of using an RNN is that it requires a larger data-set and requires longer training times [6].

As the language modelling will be part of the second goal, its progress is discussed in Section 6 Approach.

5. Intermediate results and evaluation

The model can be divided into two main parts. The first part is the classification of individual letters. For the classifier, an accuracy of at least 90% is expected. This can be evaluated using the provided test data. Another way to test the performance of the classifier is by creating new test data. This approach could provide a more accurate representation of actual performance.

5.1. Performance

The most recent result was trained on the provided augmented training data with a batch size of 8 and passed 50 epochs. Tested on the provided augmented test data resulted in a total test accuracy of 90% with a lowest class accuracy

of 66% on the letter N. This confirms the hypothesis that some letters in the sign language alphabet look alike and probably because the images are in a relatively low quality. Figure 3 shows the course of the loss during training this network. To visualize the course better, the y-axis is in log-scale. As can be seen the loss drops fairly quickly in the first few epochs and although the loss gets less, the network struggles to do so in the final epochs.

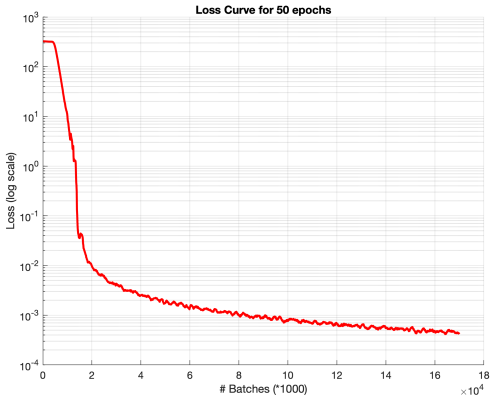

Figure 3. Course of training loss over 50 epochs with batch size 8.

Figure 4 shows the confusion matrix after testing the network on the test data. As can be seen, the network overall is performing fairly well. If individual letters are highlighted, such as the letter R, it can be seen that the network often falsely classifies the letter D as the letter R. This is because these two letters look very much alike in the ASL Sign Language alphabet.

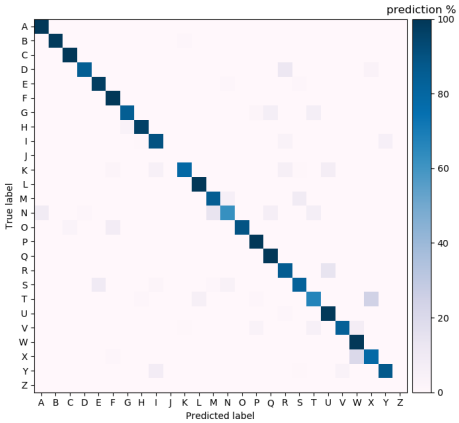

Figure 4. Confusion matrix for the predicted letters.

To get a sense of what the network is actually doing, figures 5 - 8 show the activation maps for Conv layer 1, 2, 3, and 4, respectively. As can be seen, the network has learned certain filters to activate when for example the letter C is given as input. The plots show a good representation that each filter looks at different things. An attempt to interpret the activations of the first Conv layer can be that the network sees the background, the contour edges of the hand, the edges of the hand itself and the C shape of the fingers.

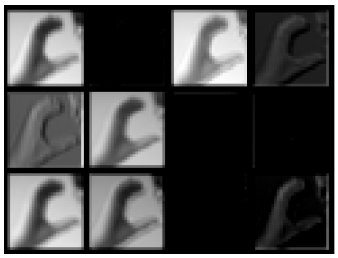

Figure 5. The 12 activation maps of the first Conv layer for an instance of the letter C.

Looking at the second, third, and fourth Conv layer activation maps, it is harder for a human to interpret the activation maps but it can be concluded that for a certain image of a letter, different filters are addressed, resulting in other activations.

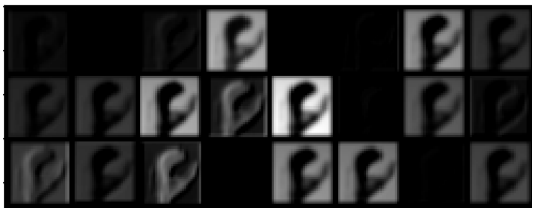

Figure 6. The 24 activation maps of the second Conv layer for an instance of the letter C.

Figure 7. The 48 activation maps of the third Conv layer after the first pooling layer for an instance of the letter C.

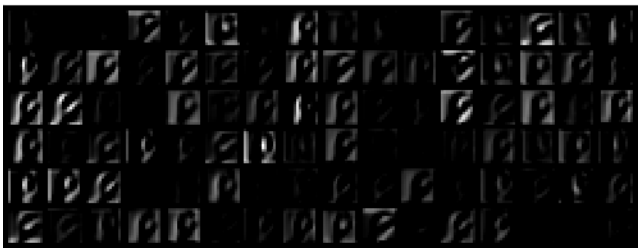

Figure 8. The 96 activation maps of the fourth Conv layer for an instance of the letter C.

The second part of the model is word prediction based on a sequence of classified letters using a sequential network. As of now there are no results for this part. The performance of this language model will be evaluated based on the accuracy of the predictions. The expected result is that the network will learn to predict the most likely character based on the sequence of previous inputs. Whether these predictions will create actual words remains to be seen.

6. Reflection of topic and approach

The original proposal stated that this project would focus on developing an RNN by comparing different possible architectures. However, it turned out a CNN was a better starting point, therefore the project was split up into two parts.

For the first part, primary results were acquired in time and look promising. The current time-frame allows for further improvements to the CNN. The results show that the classifier has trouble discriminating between certain letters. It is expected that this problem can be tackled by tuning the parameters that have not been tuned as of now, especially learning rate and momentum. The results of this tuning will be visualized to be able to pinpoint the cause of the wrong classifications and improve the classification performance. Also, a different loss function and backward pass technique will be tried. It will be interesting to see what results are obtained after tuning, but the main focus of the project will shift to the second part, sequence modeling.

Two approaches have been presented previously in section 4. The project will first focus on using n-gram modeling, because this can quickly provide baseline results. However, the main focus will lie on training an RNN. Literature research has shown that RNN's can be very powerful, but that it is also a daunting topic. Therefore, work on the RNN will start simultaneously with the n-gram modeling, so preliminary results should follow soon. Based on these first results, improvements will be made, which will likely yield better results than n-gram modeling. It is expected that all parts of the project will lead to satisfactory results within the time that is left.

References

[1] American sign language — nidcd. https://www.nidcd.nih.gov/health/american-sign-language. (Accessed on 05/26/2019). 1

[2] Brown corpus — kaggle. https://www.kaggle.com/nltkdata/brown-corpus. (Accessed on 05/26/2019). 2

[3] Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/. (Accessed on 05/26/2019). 2

[4] first20hours/google-10000-english: This repo contains a list of the 10,000 most common english words in order of frequency, as determined by n-gram frequency analysis of the google's trillion word corpus. https://github.com/first20hours/google-10000-english. (Accessed on 05/26/2019). 1, 2

[5] Vivek Bheda and Dianna Radpour. Using deep convolutional networks for gesture recognition in american sign language. *CoRR*, abs/1710.06836, 2017. 2

[6] Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. N-gram language modeling using recurrent neural network estimation. *CoRR*, abs/1703.10724, 2017. 2

[7] dwyl/english-words: A text file containing 479k english words for all your dictionary/word-based projects e.g: auto-completion / autosuggestion. https://github.com/dwyl/english-words. (Accessed on 05/21/2019). 2

[8] Angga Rahagiyanto, Achmad Basuki, Riyanto Sigit, Aditiya Anwar, and Moh Zikky. Hand gesture classification for sign language using artificial neural network. pages 1–5, 11 2017. 2

[9] Sign language mnist — kaggle. https://www.kaggle.com/datamunge/sign-language-mnist. (Accessed on 05/16/2019). 1