A breakthrough in fingerspelling-to-text translation using CNN classification and Ngram modelling

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

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 of 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]. However, most people do not understand sign language, so the hearing impaired are unable to communicate with the largest part of the population in a way that is natural for them. Research has shown that this leads to lower self esteem amongst hearing impaired students [9], overall social isolation of people with a hearing disability and even depression[12].

Finger-spelling is an integral part of learning sign language and is mainly used for spelling names and places. It forms a good basis for a deep learning classification project, since the outcome is bounded to only 26 possibilities.

The authors of this paper have researched the implementation and performance of a convolutional neural network on classifying sign language letters. In the analysis of the classifier's performance, insightful information of the hidden layers will be presented. A word modelling technique in the form of Ngram language model will be added to show that word modelling can improve the classification performance. This performance analysis will be done on both a character-level and a word-level.

This paper starts with the problem statement in Section 2. A description of the three used data-sets is given in Section 3. An elaboration on the technical approach of designing the classifier and word model will be given in Section 4. The results are covered in Section 5, followed by their evaluation and discussion in Section 6. Finally, a conclusion is drawn in Section 7.

2. Problem statement

For hearing impaired people, the natural way of speaking is by means of sign language. However, most people are able to hear, so they use the spoken word as a means of communicating. This research aims to lay the basis for a tool that directly translates sign language to spoken or written text, so that the hearing impaired can communicate with people with good hearing in a way that is natural for both parties. For this project, the scope is limited to the task of translating complete words in the form of a sequence of sign language letters to text with a performance unparalleled by any single letter classifier. On its own, a possible application of this research lies in teaching children to spell words and names of places by giving feedback on how to improve in making clear and distinct finger gestures.

Hypothesis for the classifier: the classifier will have lower accuracy among the letters in the sets [A, S], [E, M, N] and [G,H] due to the visual similarities of these signs, as can be seen from the respective images in Figure 1. A filter should be learned and designed by tuning hyper-parameters to accentuate the subtle differences between the letters.

Hypothesis for sequence modeling: the word modelling network will be able to predict subsequent characters with reasonable to high accuracy when the user tries to spell common words. By doing this, the word modeller will be able to correct mistakes made by the classifier.

3. Data-sets

Three data-sets have been used for the purpose of this research. For the sign language letters the MNIST ASL Dataset is used [15]. For language modeling, the brown-corpus [2] data-set is used to train the model and the Google-10000 [4] data-set is used to test prediction performance of the model.

3.1. Sign language

For this research, the MNIST Sign Language data-set will be used [15]. The data-set is based on 1704 color images and data-augmentation is used to create a cropped, gray-scaled and re-sized training set consisting of 27.455 28-by-28 pixel images, and a test set consisting of 7.172 28-by-28 pixel images. The data distribution of the training- and test-set can be seen in figure 2 and 3, respectively. Resizing the images to only 28-by-28 pixels can be seen as a way of regularizing the data-set by feature reduction.

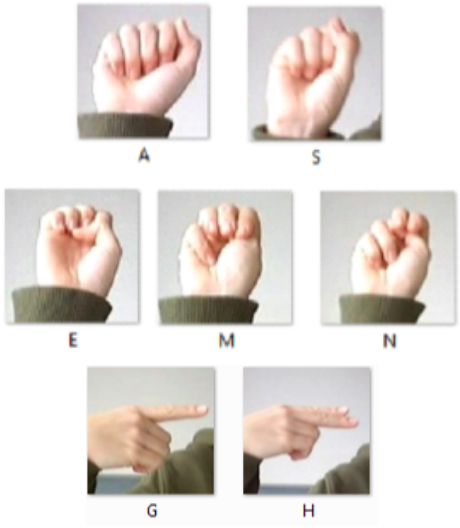

Figure 1. Examples of ambiguous hand signs for which is hypothesized that they will be hard to distinguish for the convolutional neural network.

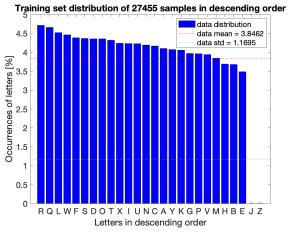

Figure 2. ASL MNIST Training set distribution

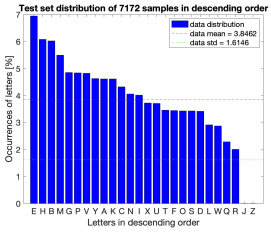

Figure 3. ASL MNIST Test set distribution

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 4.

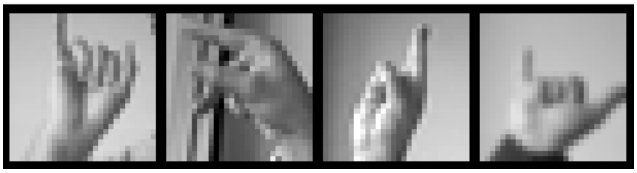

Figure 4. These figures show 4 examples of the data-set corresponding to the letters I, P, R, Y, from left to right respectively

3.2. Language model

To train the language model, the Brown-corpus data-set is used [2]. The brown-corpus data-set is a corpus of present-day American English. It contains 500 samples of English-language texts with approximately 1 million words in total.

In order to test the performance of the model on the word-level, the Google-10000 data-set is used [4]. This data-set contains a list of the 10000 most frequently used words in the English language according to Google.

4. Technical Approach

4.1. Classification

The MNIST data-set consists of images, therefore a convolutional neural network is used for classification. A common layout of a CNN architecture (as described by [3] and [7]) starts with a sequence of convolutional 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.

This common network architecture was used as a basis for this classifier. The final architecture resulted from an optimization process where the number of layer sequences, number of filters per convolutional layer, the learning rate, momentum, and loss function were varied. The optimal network that resulted from this optimization is a classifier that is made up of two Conv-ReLu layers followed by a pooling layer, again two Conv-ReLu layers followed by a pooling layer, and concludes with three FC-ReLu layers, depicted in figure 5. For each convolutional 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 loss is calculated using `CrossEntropyLoss`, which is the preferred loss function for multi-class classification problems [10]. The backward pass is done by Stochastic Gradient Descent with a decaying learning rate of 0.005 and a momentum of 0.95. The consecutive convolutional layers produce 48, 96, 144, and 192 feature maps, respectively. The three fully-connected layers have 120, 80, and 26 nodes, respectively. The batch-size during training was set to 8 and the network was trained for 50 epochs.

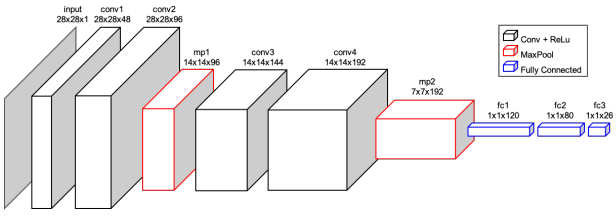

Figure 5. The final classifier architecture

4.2. Language modeling

There are multiple ways to perform natural language processing. One way is to use Ngram modeling as discussed in

[13]. Using this method, the probabilities of a sequence of two to N arbitrary characters is determined according to some data-set. These probabilities can then be used to make predictions based on the likelihood of certain letter combinations. Ngrams 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 Ngram model. The drawback of using an RNN is that it requires a larger data-set and much more training time [8].

4.2.1 Ngram Modeling

Since the language model is used to predict characters on a word-level, the sequences that need to be learned are fairly short. As a result, the use of an RNN might not provide much benefit. For this reason, it was decided to use Ngrams for language modeling.

For calculating Ngrams, the ngramModelTrainer by Github user Sfikas is used[5]. The ASL data-set contains only images of letters, the Ngram trainer is altered to only calculate Ngrams over the letters present in the ASL data-set. Before calculating the Ngrams, all words containing out of alphabet characters are removed from the Brown-Corpus data-set using a pre-processing script. The Ngrams are then calculated up to N=4, and stored in their respective tensors.

In order to predict the next letter in a sequence, the current sequence is taken into account and the corresponding Ngram tensor is used. For example, if the current sequence length is two, trigram probabilities are used to predict the next letter. The labels of the letters in the sequence are used as indices to obtain the probability distribution for the next letter. An example can be seen in figure 6.

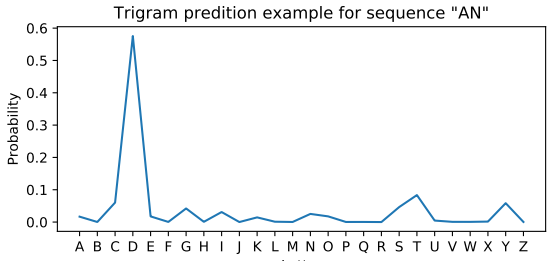

Figure 6. An example of the probability distribution that is calculated by the Ngram model for the next letter in a sequence, when "AN" are the previous letters. In this example, the most probable next letter is "D", resulting in the sequence "AND". The second most probable letter would be "T".

4.3. Combining Classification and Prediction

In order to combine the classifier with the Ngram predictor, the probability distributions are combined using Bayesian inference.

$$\frac{P(E|H)P(H)}{P(E)} \tag{1}$$

Where E represents the letter estimate made by the classifier and H is the prior hypothesis of the Ngram model. Thus, the final estimate is based on prior information from the English language and on probability data for the classifier estimate. The probability distributions for the classifier are obtained from the confusion matrix generated when testing the classifier individually. The model is tested by individually classifying every letter in the google-10000 data-set. For each letter in the google-10000 data-set, a random image with the correct label is chosen as the input to the classifier. The performance is then calculated based on the number of fully correct words. Accuracy on the letter level is also calculated. A diagram of the model can be seen in figure 7

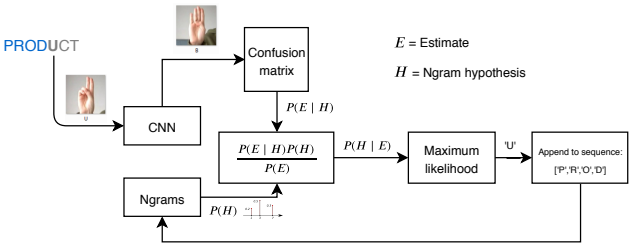

Figure 7. This figure shows a diagram of the total model. To illustrate the correction of misclassified letters, an example from testing is used. The corresponding probability distributions can be seen in figure 8

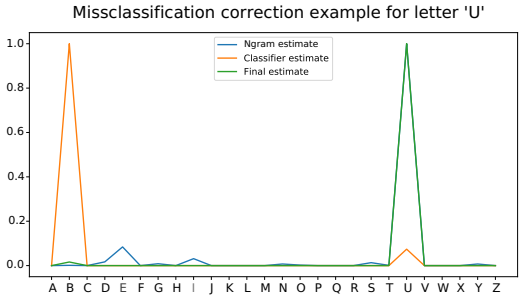

Figure 8. This figure shows the estimates for the classifier, Ngram predictor and total model based on the example in figure 7. In this example, to aid visualization, the letter with highest probability is normalized to 1.

In order to improve prediction accuracy, letters are estimated in pairs when possible. This means that instead of

only considering the probability for the next letter, a probability tree is constructed containing the probabilities of all possible combinations for the next two letters. The probabilities are calculated based on the classifier output and Ngram probabilities. The branch corresponding to the two letters with the highest probability is then selected. An illustration of this process can be seen in figure 9.

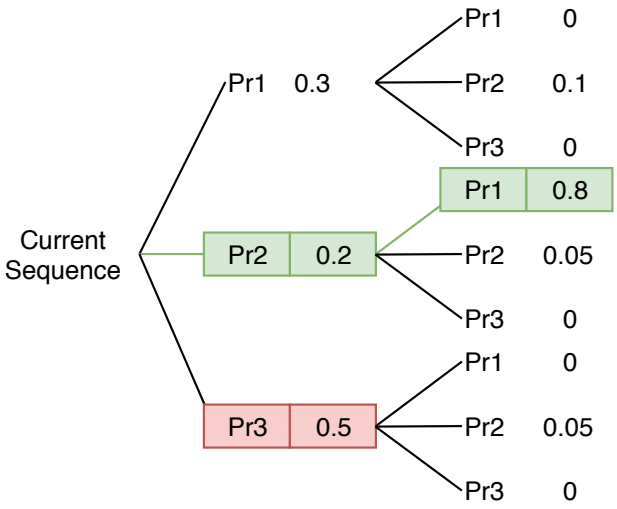

Figure 9. This figure illustrates the principle of prediction based on a probability tree. Possible predictions for the next item in the sequence are given by Pr. The probability for each prediction is also given. It can be seen that, even though Pr3 is more likely when considering only the first branch, The branch Pr2 → Pr1 is more likely than any branch starting with Pr3.

5. Results

The results are split up into two main parts. The first part is the classification of individual letters by purely using the proposed CNN. These results are evaluated using the test data that is provided by the MNIST data-set. The second part is the classification of each letter in the google-10000 data-set using the combination of the language model and CNN.

5.1. Classification using CNN

Testing the network as discussed in Section 4 on the provided augmented test set resulted in an overall test accuracy of 94.88% with a lowest class accuracy of 83% on the letter T. Figure 10 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 quickly in the first ten epochs and although the loss gets less, the network struggles to do so in the final epochs.

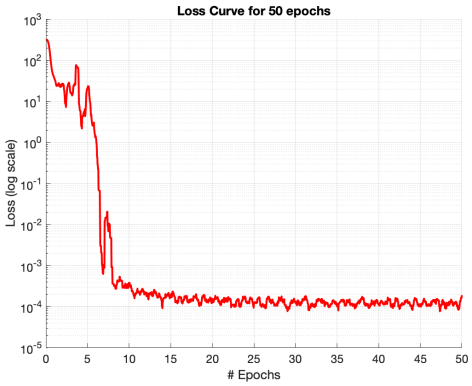

Figure 10. Training loss over 50 epochs with batch size 8.

Figure 11 shows the confusion matrix after testing the network on the test data. As can be seen, the network overall is performing fairly well. A good look at the confusion matrix shows that the network has the most trouble with letter combinations such as the following: an image with true label R sometimes gets misclassified as an L, true label S as an E, true label U with a B, and true label M with an N.

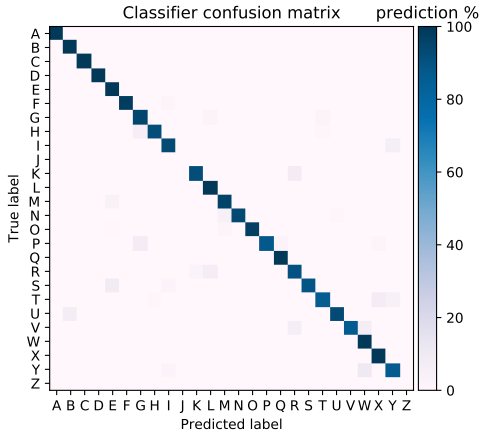

Figure 11. Confusion matrix for the predicted letters.

To get a sense of what the network is actually doing, figure 12 shows the filter maps for convolutional layer 1 prior to (left) and after activation (right). 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 convolutional 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.

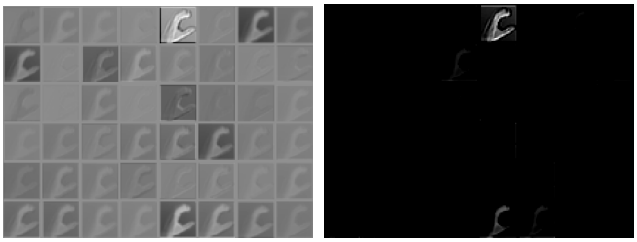

Figure 12. On the left, the output of the first convolutional layer prior to activation for an instance of the letter C. On the right the output after activation.

Not shown here, but looking at the second, third, and fourth convolutional 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 13, showing the activation of the last fully-connected layer, shows that for the letter C the network gives the highest probability to label C, classifying the image correctly.

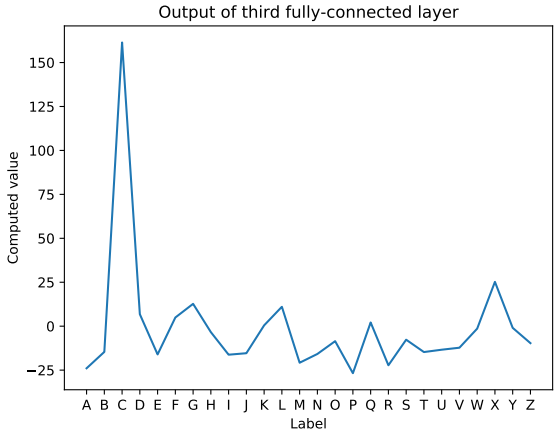

Figure 13. The 26 activation maps of the third fully connected layer for an instance of the letter C. The highest activation corresponds to the classified label.

5.2. Combined model performance

Without the Ngram model, the classifier has a word-level accuracy of 69 % over the entire test-set. For words containing more than 9 letters, accuracy drops to 55 %. When adding the Ngram model, performance increases significantly. With a word-level accuracy of 82 % over the entire test-set and an accuracy of 80 % for words containing more than 9 letters. This is illustrated in figure 14

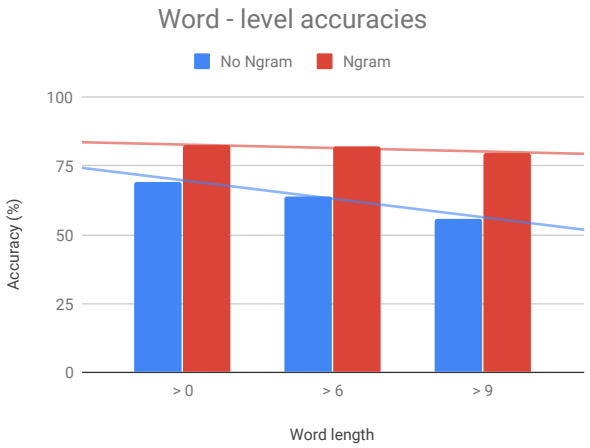

Figure 14. The performance of the total model compared to the classifier-only model for different minimum word-lengths.

It can be seen that in addition to providing an absolute increase to performance, the Ngram predictions also ensure that the accuracy is less dependent on the length of the word being spelled.

An increase in performance is also observed on the letter-level. With accuracies increasing with word length. This is due to the fact that the Ngram is more effective on longer sequences.

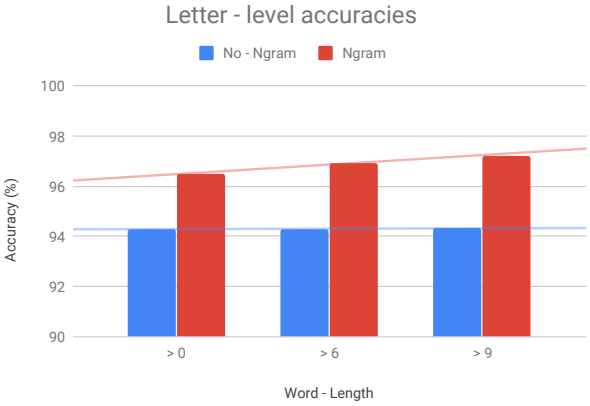

Figure 15. The performance of the total model compared to the classifier-only model on the letter-level. This is done for different minimum word-lengths.

6. Discussion and recommendations

The results look promising, but it is important to create insight into how they came to be and in which ways performance might be improved even further.

6.1. Reflection on results

The CNN has surpassed the initial expectations by achieving an accuracy of almost 95%. However, the classification performance differs remarkably per letter. For example, the letter A is always correctly classified as an A, but the letter R is only correctly recognized for 83% of test cases. This difference can be explained by the unbalanced test set. The letter R is underrepresented in the test set, as can be seen in data-set distribution in figure 3. Therefore one mistake on the overall class accuracy is penalized harder relative to other classes. The hypothesis that some letters are misclassified because they have a similar appearance can be confirmed. This phenomenon does occur, but is not solely responsible for all misclassifications. For example, the most common misclassification was the prediction of the letter X when a T was presented. figure 16 shows an example of this misclassification. At first glance this is surprising because the finger gestures are very different. However, the positioning of the hand shows similarities, so that could partially explain the error. The figure shows another possible explanation for wrongly classifying signs for hand signs that are not similar at first glance, namely the feature reduction that was performed by down-scaling the images to 28-by-28 pixels. Both images now show similar features, as indicated by the coloured circles in figure 16. The left image is a bit lighter, but the features have similar shape and location. However, this remains speculative as it can not be verified whether the CNN has used these features for classification.

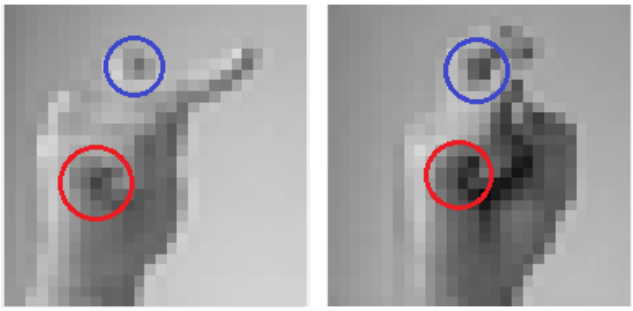

Figure 16. The image on the left shows a sample for the letter T that has been misclassified for an X, a sample of which can be seen in the image on the right. The coloured circles indicate similar features that appear in both images and might have caused the misclassification.

As can be seen in figure 14, the proposed combination of the two methods has lead to a great increase of performance on word level translation. This is because the performance of the Ngram model increases as the sequence grows, like shown in figure 15, whereas the CNN remains equally likely to misclassify each individual letter, leading to a decreased classification performance on the whole sequence for growing sequence length.

6.2. Reflection on approach

The original proposal stated that this research would focus on developing an RNN by comparing different possible architectures. However, it turned out a CNN was a better starting point, therefore the research was split up into the design of a CNN for classifying and the design of a sequence modelling framework to aid classification. The role of sequence modelling has been taken over by an Ngram model instead of an RNN for the sake of simplicity. In the end it can be concluded that choosing for this approach has been successful in the sense that it yielded satisfactory results. However, it cannot be said that this was the best approach. It is likely that combining the CNN with an RNN would have lead to even better results since it would have allow the word modeller to predict the complete word based on previous words the user has spelled. The presented Ngram model has less knowledge of the past compared to an RNN.

6.3. Recommendations

In the future an RNN is recommended for better generalization and increase in learning range, when a longer sequence of letters is desired. A long short term memory (LSTM) architecture would be a natural starting point for this problem statement. Though a convolutional neural network with a dilated and residual connections combined with causal convolutions will be a better starting point according to [14]. Another alternative to a LSTM would be an attention based sequence model as described in [6], though this network requires a considerable amount of computing power and is mostly applicable for complex application which require large amounts of memory.

Next to the choice of the architecture, considerations about data augmentation and training data should also be made. Firstly, the data-set that was provided by [15] was already augmented and down-scaled to a 28-by-28 image, introducing a dimensionality reduction. Using the full and original image would render this application more useful in live application of this algorithm. As was observed in figure 16, the pose of the hand is of great influence on the classification of the image. This means the classifier is not robust to spatial difference in position of the hand sign with respect to the camera image sensor plane. This suggests an adversarial training and adding information about 3D positions of the hand pose would greatly improve the robustness and likely the accuracy of the classifier in real-life applications [11].

7. Conclusion

An initial analysis of the data-set resulted in a educated guess that a filter should be learned by the convolutional neural network that accentuate the subtle differences between the letters [A, S], [E, M, N] and [G,H]. Though as the

results indicated that the CNN has the most difficulty with letter combinations R being classified as L, S with an E, U with an B, and finally M with an N. This partially confirms our initial analysis and confirms the general understanding of the innerworkings of convolutional neural networks.

In addition, the CNN had an overall test accuracy of 94.88% with a lowest class accuracy of 83% on the letter T and this is mainly the consequence of the most common misclassification of the letter X when a T was presented. This was not expected since the finger gestures are very different, but the positioning of the hand shows similarities. This suggest that the classifier is sensitive to positioning and it is recommended to investigate possibilities for adversarial data augmentation.

The second hypothesis stated that the Ngram word modeller can correct mistakes made by the classifier by predicting subsequent characters. This is true. As the classifier without the Ngram model, has a word-level accuracy of 69 % over the entire test-set. For words containing more than 9 letters, accuracy drops to 55 %. When adding the Ngram model, performance increases significantly. With a word-level accuracy of 82 % over the entire test-set and an accuracy of 80 % for words containing more than 9 letters.

The simplicity of Ngram models combined with a well performing CNN classifier makes it a powerful combination to achieve fingerspelling-to-text translation for possible future applications for hearing impaired children to learn fingerspelling.

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). 1, 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] sfikas/ngrammodeltrainer: Train an n-gram model given a corpus of words. https://github.com/sfikas/ngramModelTrainer. (Accessed on 06/25/2019). 3

[6] et al Ashish Vaswan, Noam Shazeer. Attention is all you need. *arXiv*, 2017. 6

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

[8] Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. N-gram language modeling using recurrent neural network estimation. *CoRR*, abs/1703.10724, 2017. 3

[9] David Farrugia and Gary F. Austin. A study of social-emotional adjustment patterns of hearing-impaired students in different educational settings. *American Annals of the Deaf*, 125(5):535–541, 1980. 1

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 2

[11] et al Michael A. Alcorn, Qi Li. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. *arXiv*, 1811.11553v2, 2019. 6

[12] Mary Pat Moeller. Current state of knowledge: Psychosocial development in children with hearing impairment. *Ear & Hearing*, (28(6)):729–739. 1

[13] 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. 3

[14] Vladlen Koltun Shaojie Bai, J. Zico Kolter. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv*, 2018. 6

[15] Sign language mnist — kaggle. https://www.kaggle.com/datamunge/sign-language-mnist. (Accessed on 05/16/2019). 1, 6