Sign Language Recognition

Geetesh Gupta

May 2020

1 Introduction

Sign languages are used by the deaf and hard hearing people to exchange information with others. Recognition of these signs and gestures by the computer would provide an easier communication between the non audible and audible people. In this assignment, we will take a step forward in understanding how different models are used to classify different signs into American alphabets. This assignment involves comparing a pre-designed CNN model with three other models namely CNN with fine tuning, LSTM and CNN-LSTM.

Figure 1: American sign language

2 Related Work

This assignment is based on the findings of the paper(Masood, Thuwal, & Srivastava, 2018) where a similar approach was implemented in recognition of the American alphabets from different signs using the VGG16 model(Simonyan & Zisserman, 2014) of deep convolutional neural networks. An advanced version of this approach was shown in Paper(Pigou, Dieleman, Kindermans, & Schrauwen, 2014) where the authors have used a skeleton approach of extracting the hand gesture features from video sequences. Recently, one more interesting work of recognising real time American gestures using the surface Electromyography(Savur & Sahin, 2015) was published. Visually representing content, producing sign language annotation and generalisation to the unseen environment are some of the biggest challenges in this field.

3 Methodology

A model was given in the github repository which consists of 3 convolution layers, 3 maxpooling layers and 2 fully connected layers for the convolution neural networks. Fig.2 shows the architecture of the model. The filter size was kept as 3x3 to extract the details from the images.

3.1 Dataset

Initial work has been done on the sign language MNIST dataset[1]. This dataset consists of around 34k images having signs which have been classified into 24 categories. Each category represents an American alphabet(A-Z) corresponding to their signs. An important thing about this dataset is, it doesn't contain signs for alphabet J and Z since they require

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

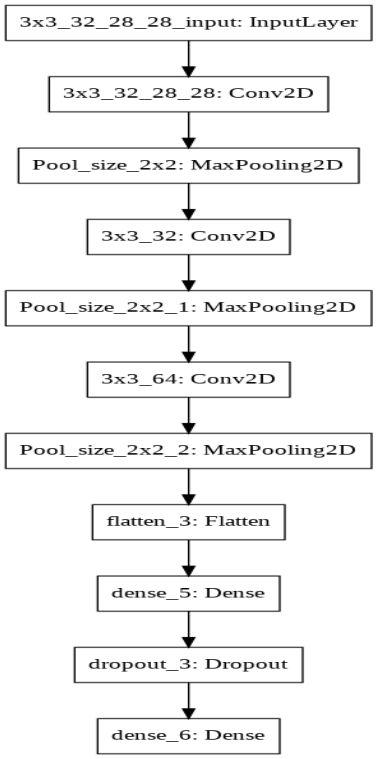


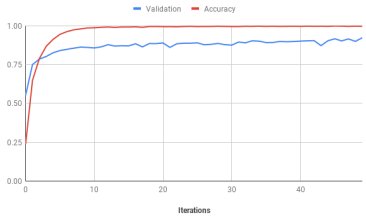
Figure 2: Architecture

hand motions. This dataset has been split into training and testing sets containing 27k and 7k images respectively. When the said model is trained on this dataset, it gives a validation accuracy of 89%. Figure3 shows the accuracy and the loss for this dataset. The weights of this model are stored which will be used later to fine tune a similar model on a different dataset.

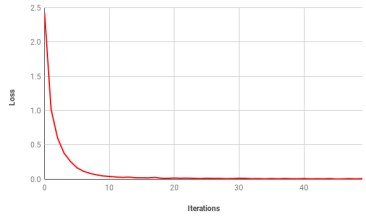
3.2 New dataset

The next dataset is taken from a github repository[2]. It contains 65k images belonging to 29 categories. The 3 extra categories correspond to signs apart from alphabets are removed resulting in 26 categories rather than 24 as previous. Since this dataset is also

[2]https://github.com/loicmarie/sign-language-alphabet-recognizer



(a) Accuracy



(b) Loss

Figure 3: Training results

very large, working with the CNN models would result in good accuracy which would make it harder to analyse. Therefore, I have selected about 100 images from each category for training and 50 for testing which is 3k and 1k set of images respectively. Also, a dictionary was maintained for keeping track of the images belonging to different categories.

3.3 Intentions

The main intention of this assignment is to try different models with different parameters on new dataset for the classification of signs and compare their performance. I have designed and included 3 more models apart from the CNN model mentioned earlier.

3.4 Other models

3.4.1 CNN with fine tuning

The idea implemented in this model was to take the weights of upper layers of the CNN model trained for MNIST dataset and freeze those layers which makes them untrainable. The architecture2 was manipulated by adding an extra layer of fully connected layer

and the last layer was modified to classify the output into 26 categories. It was observed that when the upper layers are frozen, the validation accuracy was slightly(79%) less than, when the layers were not frozen(83.5%). The best one is selected for comparison.

3.4.2 LSTM

A simple LSTM model was implemented with only one LSTM layer and one fully connected layer for classification along with a dropout layer. I wanted to design such a model because the processing time of LSTM is drastically larger when compared with CNN. A simpler model wouldn't be efficient but it will help to understand the overcoming with respect to other models. The dropout layer is important in these types of models to fight overfitting. The model was tested for 2 cases. First, when the hidden nodes are set to 50 and Second, for 100 nodes. The accuracy obtained with 100 hidden nodes was better than 50 which was selected for further observations.

3.4.3 CNN+LSTM

The design of this model is also quite similar to fig.2 where one of the fully connected layer is replaced with an LSTM layer with 64 hidden nodes and the rest of the layers are replaced with time distributed layers. As this is a deeper network than above, early stoppage is used with a patience of 10 for training. The performance was expected to be better than LSTM but would be interesting to see how it performs against fined tuned CNN.

4 Results

4.1 Model performance

Table 4 shows the overall performance of all the models. As mentioned earlier, the training set and testing set had 3k and 1k images respectively(100 and 50 from each category). As expected, LSTM performs the worst while fine tuned CNN is the best one.

Model	Validation	Recall	F1 Scores	Precision
Fine tuned CNN	83.5%	80.3%	80.6%	81.4%
CNN-LSTM	81.5%	78.4%	78.6%	79.5%
LSTM	74.3%	71.5%	72.7%	75.3%

Figure 4: Performance analysis

4.2 K-fold cross validation

This technique is used to test the model by splitting the dataset into k groups and training the model on them. For this assignment, K=5 has been taken. The data used for this validation is 80% in size of the above sets used. Fig.5 is the boxplot showing the maximum, minimum and average accuracy of the model. The further decrease in accuracy than shown in the table above is due the reduction in the size of the data. The loss (fig.6) remains very close between CNN-LSTM and LSTM. On an average, the CNN-LSTM performs almost similar to the LSTM model.

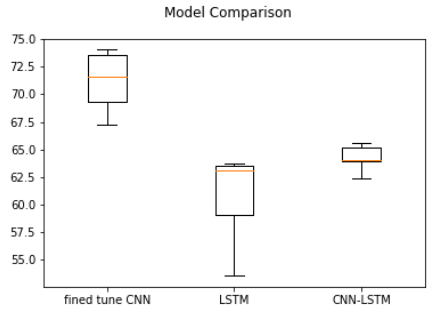


Figure 5: Accuracy

4.3 Best model

Based on the performance of above models, fine tuned CNN is selected as the best one. A confusion matrix is shown in fig.7 which gives the performance of the model in classifying the signs into alphabets. The model had a hard time in classifying the signs for M and O.

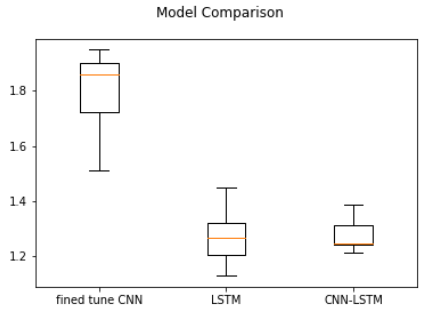


Figure 6: Loss

Figure 7: Confusion Matrix

5 Discussions

By looking at the performance of these models, we can say that they fail to extract in-depth details from the images due to the data being small. When the same fine tuned CNN model was trained on the whole dataset, it gave an accuracy of 97%. This tells us the importance of data augmentation in providing good accuracy when the dataset is small. For the small dataset, a deeper CNN model with multiple stacked convolution layer would be helpful in bringing the accuracy up. The LSTM and CNN-LSTM model can be improved by training the models and changing the number of hidden nodes.

6 Conclusion

We have implemented and observed how different models work and perform when the data is small. We have also observed that CNN models are faster than LSTM models and can reach the same accuracy in a couple of epochs. A deeper CNN model with multiple stacked convolution layers would be helpful in bringing the accuracy up. Further, the same approach can be tried on recognising words and sentences from gestures in video sequencing. One more interesting thing to test out is when both the hands are involved in signs with different motions.

References

Masood, S., Thuwal, H. C., & Srivastava, A. (2018). American sign language character recognition using convolution neural network. In *Smart computing and informatics* (pp. 403–412). Springer.

Pigou, L., Dieleman, S., Kindermans, P.-J., & Schrauwen, B. (2014). Sign language recognition using convolutional neural networks. In *European conference on computer vision* (pp. 572–578).

Savur, C., & Sahin, F. (2015). Real-time american sign language recognition system using surface emg signal. In *2015 ieee 14th international conference on machine learning and applications (icmla)* (pp. 497–502).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.