Sign Language to Text Translation

by

Aniket Sharma

2019BCS-008

A report submitted for Summer Project

Bachelor of Technology

in

CSE

विश्वजीवनामृतं ज्ञानम्

ATAL BIHARI VAJPAYEE-

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

GWALIOR - 474015, MADHYA PRADESH, INDIA

Report Certificate

I hereby certify that the work, which is being presented in the report, entitled Sign Language to Text Translation, for Summer Project in Computer Science Engineering and submitted to the institution is an authentic record of my/our own work carried out during the period *May-2021* to *August-2021* under the supervision of **Prof. Shashikala Tapaswi**. I also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Date: 10 August 2021 Aniket Sharma

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 10 August 2021 Prof. Shashikala Tapaswi

Candidate's Declaration

I hereby certify that I have properly checked and verified all the items as prescribed in the check-list and ensure that my thesis is in the proper format as specified in the guideline for thesis preparation.
I declare that the work containing in this report is my own work. I understand that plagiarism is defined as any one or combination of the following:

(1) To steal and pass off (the ideas or words of another) as one's own

(2) To use (another's production) without crediting the source

(3) To commit literary theft

(4) To present as new and original idea or product derived from an existing source.

I understand that plagiarism involves an intentional act by the plagiarist of using someone else's work/ideas completely/partially and claiming authorship/originality of the work/ideas. Verbatim copy as well as close resemblance to some else's work constitute plagiarism.
I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programmes, experiments, results, websites, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.
I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report/dissertation/thesis are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable. My faculty supervisor(s) will not be responsible for the same.

Name: Aniket Sharma

Roll. No: 2019BCS-008

Date: 10 August 2021

Abstract

Sign languages are natural languages, with their grammar and lexicon, expressed using visual-manual modality. Out of more than 150 sign languages worldwide, ASL is the most widely studied. The task of Sign Language translation is ongoing research.

This project focuses on **Fingerspelling** component of ASL.

There are three different startegies for Sign Language Translation:

- using specialized hand-tracking tools
- using depth map
- using Computer Vision

The main objective of this project is to develop an AI system capable of translating Sign Language without requiring any specialized hardware.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor **Prof. Shashikala Tapaswi**, for her esteemed mentorship, and allowing me to explore and experiment with my ideas in the course of making this project a reality. The leeway I was given went a long way towards helping cultivate a genuine thirst for knowledge and keeping up the motivation to achieve the best possible outcome.

This project motivated me to explore many areas of Computer Science that are new to me and kindled an interest to further follow up on many of these areas. Moreover, the successful completion of this project has brought great satisfaction and confidence in my ability to produce more high-quality, non-trivial work that can be helpful for society.

I would also like to thank the institution for allowing me to pursue the project despite the current pandemic.

Aniket Sharma

Contents

Tables

Figures

Chapter 1

Introduction

This chapter provides a brief introduction to the project. Section 1.1 provides context regarding the project. Next, Section 1.2 discusses the problem and motivation. The project objective is discussed in Section 1.3. In the end, all of these are used to formulate a workflow in Section 1.4.

1.1 Context

Sign languages are natural languages, with their own grammar and lexicon,[18] expressed using visual-manual modality. Although there are more than 150 sign languages worldwide[5], the **American Sign Language (ASL)** is the most widely studied.

In ASL, there are distinct symbols for many characters, words, and phrases. The use of so many symbols add complexity to the language, which can be both a good thing and a bad thing depending on the proficiency of the language user. **Fingerspelling** is a way of expressing Sign Language by making use of alphabet characters and some other representational characters to spell-out the text. Fingerspelling in ASL accounts for 12% to 35% of discourse by signers regardless of their age, gender, class, and ethnicity.[13]

1.2 Problem/Motivation

The large number of symbols in ASL results in huge computational and memory costs if each of these symbols is to be taught to an AI system. Although this problem cannot be avoided while developing a full-fledged translation system, for research purposes this work makes use of a dataset containing only 28 symbols, each for 26 English alphabets, 1 for ‘space’, and 1 for ‘nothing’. Thus, the system developed will be able to translate fingerspelled text.

Most of the prominent work in this area, as discussed in Chapter 2, relies on specialized devices to accurately determine the hand’s shape and motion. The data gathered through such devices helps in creating accurate Translation systems. However, it is not feasible to provide such devices to all signers worldwide, and thus there is a need for an AI system capable of solving the problem using technology that is already widely available and used.

The goal of this project is to use computer vision for solving the Sign Language Translation problem. Therefore, various Multi-Layer Perceptron (MLP) are suggested using Convolutional Neural Network (CNN), Transfer Learning, and Ensembling techniques.

1.3 Objectives

The project aims to build a Computer Vision-based AI system that can translate real-time Sign Language to English by designing, testing, and studying various Deep Learning modelling approaches. This will make it easier for members of the Deaf community to connect with the rest of society, giving them an equal opportunity in this technological age.

Briefly, the main objective of the project can be summarized as to develop an AI system capable of translating Sign Language without requiring any specialized hardware.

1.4 Work flow

In regards to the objectives discussed in Section 1.3, the workflow prepared is stated as:

Step 1: Finalizing a dataset for training.

Step 2: Designing a basic CNN.

Step 3: Training the basic CNN using ASL Dataset described in Section 4.2.1 and evaluating its performance on the same data.

Step 4: Designing another CNN employing Transfer Learning to improve the performance of the

AI system.

Step 5: Applying Data Augmentation to the ASL Dataset such that the model trained on the data is capable of classifying unobserved signers.

Step 6: Training the basic CNN and Transfer Learning CNN again from scratch using augmented data.

Step 7: Create Bootstrap Aggregated data sets for Ensemble model.

Step 8: Design an Ensemble CNN and train each model of the ensemble using a different set of bootstrap aggregated data.

Step 9: Evaluating the performance of all designed and trained models.

Chapter 2

Literature review

This chapter provides detailed information regarding the current progress in the area of designing Sign Language Translator AI systems.

2.1 Background

The task of Automatic Sign Language recognition and translation is an ongoing research area. Many works provide different strategies to tackle the problem: using **specialized hand tracking devices**[4][15], using **depth map**[2][8][14][16], and using **Computer Vision**[6][9][12][11].

This work suggests a new strategy for the Computer Vision approach. In the upcoming section, the reason for taking this approach, and the discussion regarding each of the approaches mentioned above is provided.

2.2 Key related research

In the previous section, three different research areas for Sign Language translation were mentioned. In this section, each of these areas will be discussed in-depth.

2.2.1 Using Specialized Hand Tracking Tools

Some of the most notable advancements in this area are made using a combination of hand-tracking devices and supervised learning algorithms. The use of these devices provide accurate and relatively structured data[7] compared images captured using a camera. Thus, enabling the development of fast and accurate Sign Language Translation systems with up to 100% accuracy[15].

The main drawback of such an approach is that they require the installation of costly hand tracking devices. For the same reason, such approaches cannot be used on the wildly available Sign Language data.

2.2.2 Using Depth map/images

Most of the development in this approach for Sign Language Translation using **depth images or depth maps** use specialized 3D sensors capable of taking depth images like Microsoft Kinect Cameras. These depth images, along with the color images taken using cameras, are used to train a supervised learning algorithm[14][16] or a CNN[2].

As opposed to the previously discussed approach, the main advantage of this approach is the relatively cheap cost and ease of installing 3D sensors. Still, this approach does not entirely address the limitations of using Specialized Hand Tracking Tools.

2.2.3 Using Computer Vision

This approach can be further divided into two categories: using dynamic properties of hand movements[6] and using static hand shapes[9][11][12]. Both of these approaches make use of data collected using cameras and thus have no extra setup installation costs, enabling them to be used in real-world settings easily.

While the dynamic hand movement approach is generally better for Sign Language translation due to the dynamic nature of Sign Languages, the static hand shape approach is computationally cheaper. Since ASL alphabets are primarily static, the latter approach is an obvious choice for fingerspelling. As discussed in [17], we can combine the benefits of both approaches by using two different models, one for fingerspelling and one for other character recognition and translation.

Some other approaches combining the benefits of both Computer Vision approach and Hand Tracking Device approach, by using colored gloves instead of specialized devices, also exist and are under research[10].

2.3 Analysis

All the approaches discussed in the previous section can translate Sign Language with reasonable accuracy, but only the Computer Vision approach can be used in production where the systems are required to be economically and computationally feasible.

As shown in [8], the depth image approach can achieve a near-perfect accuracy of 99.99% in observed signers and 83.58% to 85.49% accuracy for new signers by using CNN.

In this project, I will try to translate Sign Language using Computer Vision. Various CNN models are suggested in Chapter 3 and the evaluation of the performance of these models is discussed in Chapter 4.

2.4 Conclusion

This chapter describes the three major approaches to the Sign Language Translation problem, their advantages and limitations. In the end, the direction this work is going to take is stated.

Chapter 3

Methodology

The approach to developing and testing a Sign Language Translation AI system is described in this chapter.

3.1 Models and Approaches

The numerous techniques to constructing and training a CNN for Sign Language Translation are described in this section.

The training data consists of 84,000 images and 28 classes. Section 4.2.1 has a more thorough explanation of the data, and some example images of the data are shown in Figure 3.1.

To make the pictures compatible with the criteria of the pre-trained models used in the Transfer Learning based method, Section 3.1.2, the training images have been scaled to 224x224 pixels before training each of the models.

3.1.1 Basic CNN Model

The first model designed for this problem is a CNN. The number of layers and the features of each layer were determined through rigorous analysis and, trial and error.

Notebook Link: Basic CNN training

The layers in this model are as described below:

- Convolution Layers: 4

  Features:

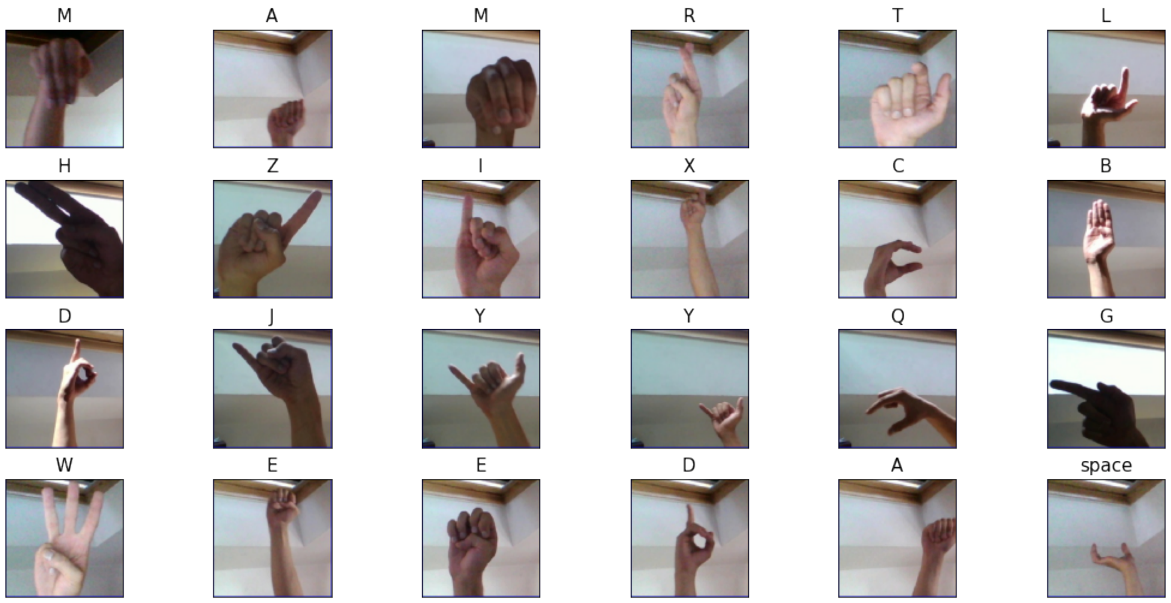

Figure 3.1: Visualization of data used for training the models

  * Kernel Size: 9x9
  * Filters: 64
  * Activation Function: ReLU

- MaxPooling Layers: 4

  Features:

  * Pool Size: 2x2

- Dense Layers: 4

  Features:

  * Units: 128
  * Activation Function: ReLU

- Output Layer

  Features:

  * Units: 28

∗ Activation Function: Softmax

To decrease the model parameters and therefore the training time, each convolutional layer is followed by a maxpooling layer. The output layer is made up of 28 units, one for each of the 28 classes, with the softmax activation function applied to the output to indicate the likelihood that a given picture belongs to that class. Regularization is not used in any part of the MLP since it wasn't found to provide much benefit, and regularised models were found to be prone to becoming trapped in local minima.

Pixel values of all the three channels of all images has been normalized by diving each value by 255.

A visualization of the architecture of the model has been depicted in Figure 3.2.

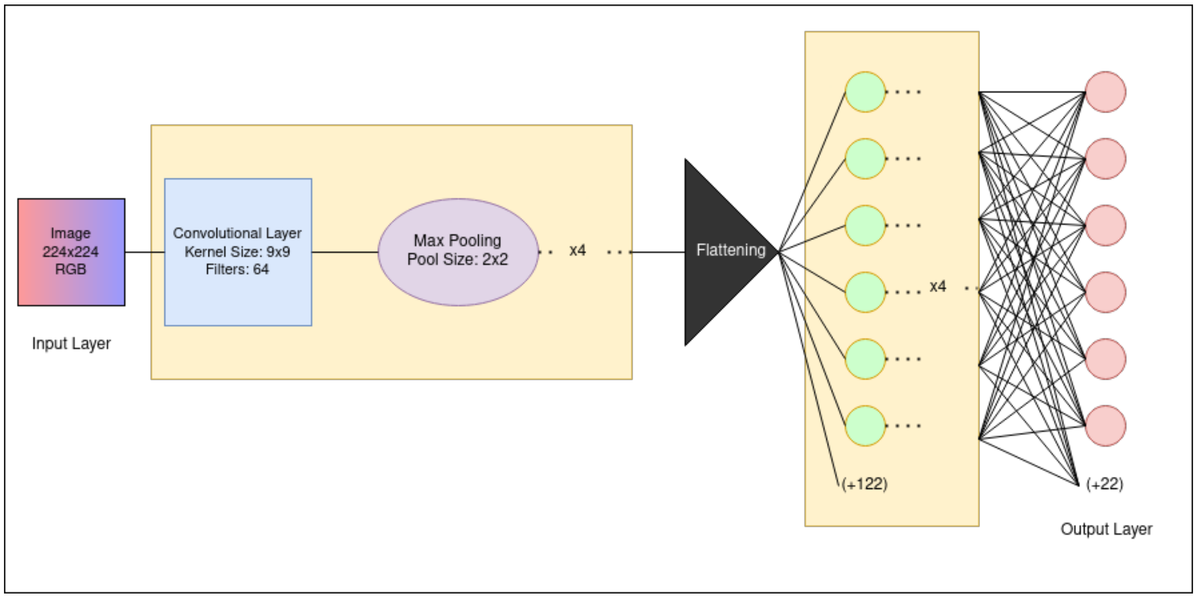

Figure 3.2: Baseline Model Architecture

3.1.2 Transfer Learning

Transfer Learning is an algorithmic process in which the knowledge gained from one problem is used in a different problem. Various models pre-trained on ImageNet data has been trained again for one epoch each and validation accuracy of these models has been compared in Table 3.1. The model is created by adding two Dense layers, each containing 64 units and LeakyReLU activation function, and one Dense layer for output containing 28 units, each representing one character in the data, and softmax activation function on top of the convolutional part of the pre-trained models.

Notebook Link: Performance evaluation of pre-trained models

Model	**Validation Accuracy**	**Training Time (sec.)**
EfficientNetB0	89.00%	177
DenseNet201	88.73%	452
EfficientNetB5	88.62%	575
DenseNet169	88.45%	307
EfficientNetB7	88.23%	999
ResNet152	88.15%	688
EfficientNetB3	88.08%	331
MobileNet	87.99%	203
EfficientNetB1	87.93%	225
EfficientNetB4	87.78%	452
EfficientNetB2	86.90%	270
MobileNetV2	85.98%	167
ResNet101	83.98%	507
ResNet50	83.90%	325
ResNet152V2	83.77%	605
ResNet50V2	83.74%	265
EfficientNetB6	83.29%	713
VGG16	83.03%	396
ResNet101V2	81.99%	448
DensityNet121	81.14%	265
Xception	80.14%	398
VGG19	79.66%	446
InceptionV3	76.33%	206
InceptionResNetV2	75.11%	453
NASNetMobile	73.06%	202

Table 3.1: Performance of models created using pre-trained models. Each of these models were trained for one epoch.

EfficientNetB0 was found to be the most efficient for the application, as evident from Table 3.1 and has been used to design the model for the Sign Language Translator. Figure 3.3 shows the architecture of the Transfer Learning model.

Notebook Link: Transfer Learning CNN training

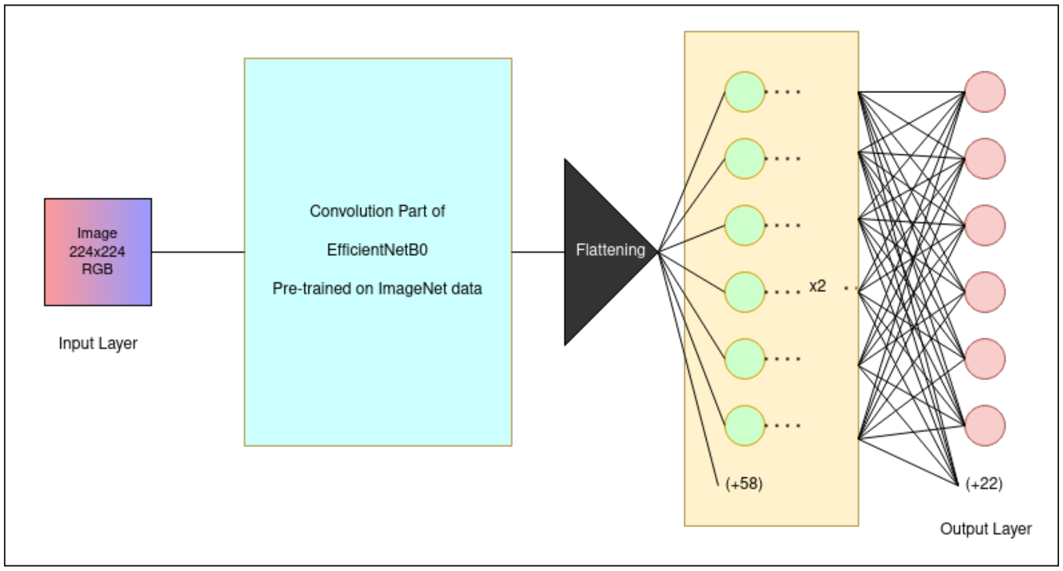

Figure 3.3: Transfer Learning CNN with EfficientNetB0

3.1.3 Data Augmentation

To make an attempt to improve the performance for new signers without requiring more data. The models designed and trained in the previous sections has been trained again using augmented data.

Table 3.2 shows the augmentation applied to the data, and the augmented data is shown in Figure 3.4.

Augmentation Type	**Value**
Rotation	20°
Width Shift	10%
Height Shift	10%
Brightness	20%-100%
Shear	45°
Zoom	50%-150%
Channel shift	100 px

Table 3.2: Augmentations applied to the data

Notebook Link: Basic CNN retrained with augmented data

All the pre-trained models described in Section 3.1.2 has been evaluated again after training

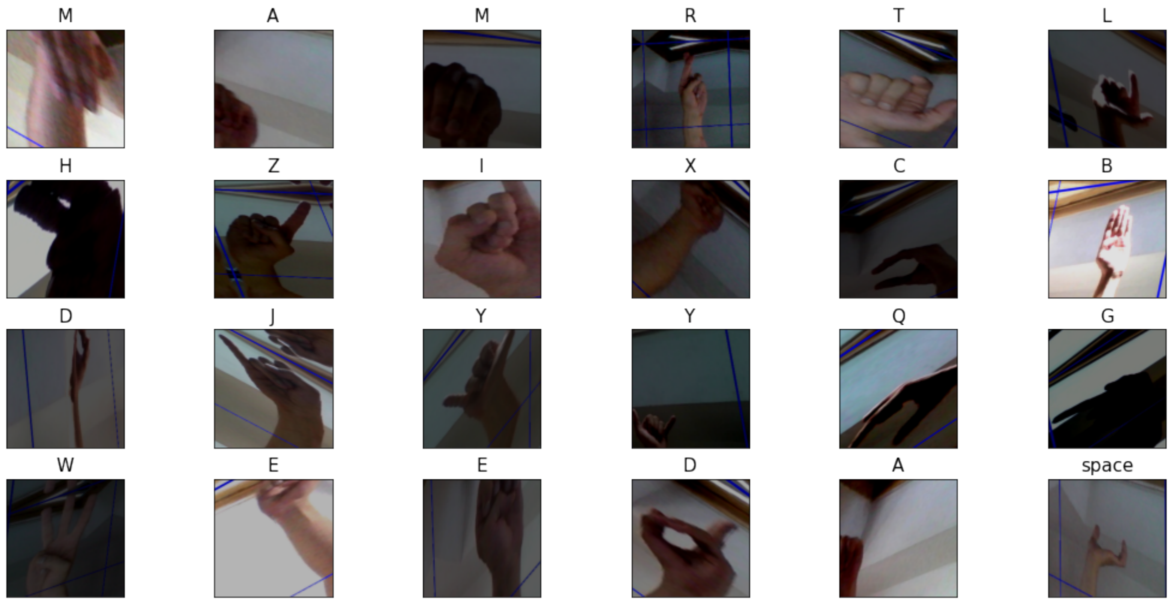

Figure 3.4: Data after applying Data Augmentation (Table 3.1)

for one epoch each on the augmented data. Table 3.3 shows the validation accuracy and training time for all such models. Again, the best performing model, MobileNet in this case, is used for full training and further evaluation.

Notebook Link: Performance evaluation of pre-trained models with augmented data

Notebook Link: Tranfer Learning CNN training with augmented data

3.1.4 Ensemble Model

The last approach for efficient Sign Language translation is to use an ensemble of models. A majority vote ensemble of five basic CNN, Section 3.1.1, has been created. **Bootstrap Aggregated** data of the same size as the training data has been created for training the basic CNNs. Thus, the variance of the models on the data is high which is expected from high performing ensemble models. Each of the models are trained separately and the final prediction of the model is the majority vote of all the models.

The creation of Bootstrap Aggregated data and training of the Ensemble Model is show in the corresponding notebook.

Model	**Validation Accuracy**	**Training Time (sec.)**
MobileNet	65.95%	933
ResNet152	62.62%	1177
EfficientNetB5	62.57%	1082
EfficientNetB7	61.69%	1251
DenseNet201	61.49%	1067
ResNet101	61.01%	1100
ResNet101V2	60.46%	1056
EfficientNetB1	60.18%	970
ResNet50	59.51%	1033
DenseNet169	59.43%	1028
ResNet50V2	59.26%	987
EfficientNetB3	58.72%	1051
EfficientNetB0	58.38%	959
ResNet152V2	58.36%	1171
EfficientNetB2	57.10%	972
EfficientNetB4	55.95%	1032
EfficientNetB6	55.65%	1176
MobileNetV2	52.28%	933
VGG16	51.86%	1080
VGG19	51.35%	1086
DensityNet121	50.92%	1015
Xception	50.70%	1058
InceptionResNetV2	48.34%	1062
InceptionV3	46.31%	966
NASNetMobile	43.38%	982

Table 3.3: Performance of models created using pre-trained model and trained on augmented data. Each of these models were trained for one epoch.

Notebook Link: Ensemble Model training

3.2 Evaluation Metrics

The evaluation of all the models designed and trained is done in the next chapter with detailed explanation of the datasets used for evaluation.

Metrics used for evaluation of the models are:

- Classification Accuracy
- Precision Score

- Recall Score
- F-score

3.3 Conclusion

In this chapter, the models designed and trained for the Sign Language Translator have been discussed along with their architecture. The Data Augmentation steps taken in an attempt to improve model performance for unobserved signers are also described. In the end, the evaluation metrics used to check the performance of the models have been mentioned. In the next chapter, the model evaluation procedure is discussed in detail.

Chapter 4

Experiments and results

The models are evaluated in this chapter using the metrics described in Section 3.2. The setup for the tests is described in the first section of this chapter, Section 4.1. Next, in Section 4.2, the models will be tested on the ASL dataset, which was used to train the models, on a wild video extracted from YouTube containing ASL alphabets and a video captured using a Webcam. The chapter comes to a close in Section 4.3.

Notebook Link: Evaluation

4.1 Setup

Three distinct datasets from three different sources are used to evaluate the models.

(1) ASL Dataset[1]

(2) ASL Alphabets[3]

(3) Webcam video

All trained models require images with three channels, to represent colored images, and a resolution of 224x224 pixels. The images are required to be normalized before they are provided to the model.

To satisfy the above-mentioned criteria for input, the following operations are applied to the datasets before making predictions:

- **Cropping** to make the height to width ratio 1:1

- **Resizing** to size 224x224 pixels
- **Normalizing** the pixel values of images

4.2 Evaluation

4.2.1 ASL Dataset

This is the dataset that the models were trained on. The models are evaluated on this dataset to help determine how well they perform for signers they have previously observed. Throughout this report, this dataset is referred to as the 'ASL Dataset.'

4.2.1.1 Dataset description

Colored pictures with a dimension of 200x200 pixels make up the dataset (see Figure 3.1). The source of the dataset is Kaggle[1]. There are 84,000 images in all, divided into 28 classes: 26 alphabets, 1 'space' class, and 1 'nothing' class. Each class has the same number of images as the other, with a total of 3,000 images in each.

4.2.1.2 Results

The values of all metrics for the dataset are shown in Table 4.1.

Model	**Accuracy**	**Precision**	**Recall**	**F-Score**
Basic CNN Model	95.71%	0.958	0.957	0.957
Transfer Learning CNN	98.12%	0.982	0.981	0.981
Basic CNN Model with Data Augmentation	95.72%	0.961	0.957	0.957
Transfer Learning CNN with Data Augmentation	94.95%	0.951	0.949	0.949
Ensemble Model	99.99%	0.999	0.999	0.999

Table 4.1: Metric Evaluation of ASL Dataset

4.2.1.3 Conclusion

On this observed dataset, all of the models performed well, with the Ensemble model (Section 3.1.4) achieving near-perfect accuracy. Training on augmented data affected the model performance for the Transfer Learning CNN and gave similar performance as model trained on non-augmented data for the Basic CNN model.

4.2.2 ASL Alphabets

This dataset was produced from a YouTube video[3] intended to teach ASL alphabets to beginners. This was chosen so that the models could be evaluated on a professional signer who had not been observed throughout training. This dataset is referred to as 'ASL Alphabets'.

4.2.2.1 Dataset description

The dataset consists of 32 colored images of size 224x224 for each alphabet. Figure 4.1 presents a glimpse of the dataset.

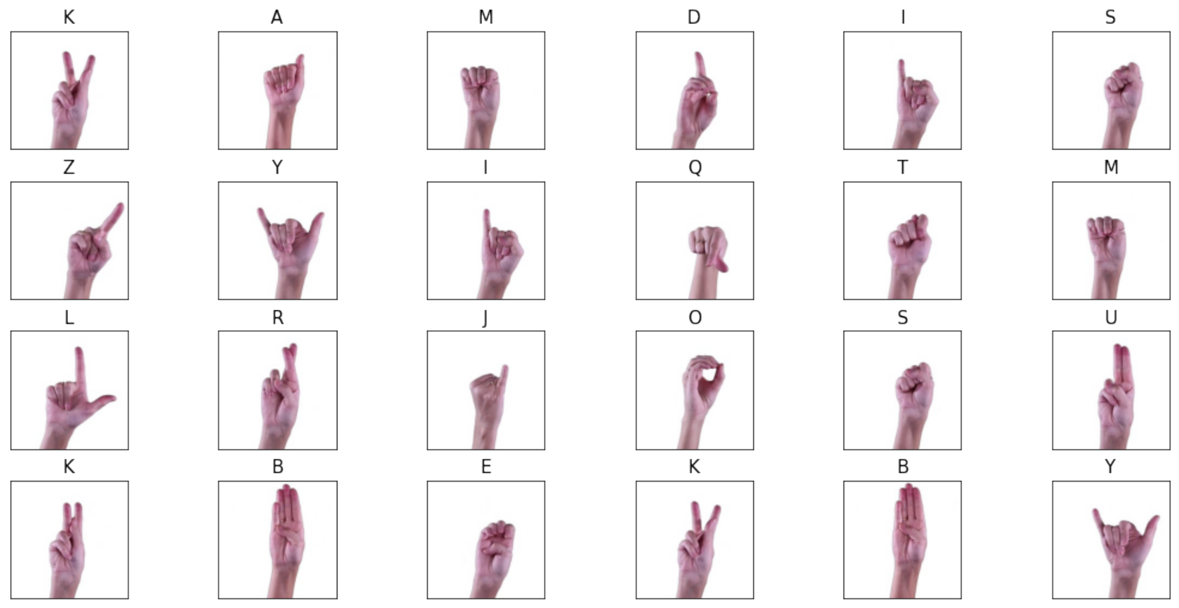

Figure 4.1: ASL Alphabets (Source: YouTube)

4.2.2.2 Results and discussion

The values of all metrics for the dataset are shown in Table 4.2.

Model	Accuracy	Precision	Recall	F-Score
Basic CNN Model	37.02%	0.230	0.344	0.257
Transfer Learning CNN	42.31%	0.323	0.407	0.315
Basic CNN Model with Data Augmentation	36.78%	0.259	0.368	0.269
Transfer Learning CNN with Data Augmentation	43.39%	0.422	0.434	0.380
Ensemble Model	44.11%	0.365	0.441	0.353

Table 4.2: Metric Evaluation of ASL Alphabets

As seen from Table 4.2, none of the models performed too well on this data but the Ensemble model performed better than others. The basic model trained with augmented data has the same accuracy as without, but the Precision, Recall, and F-Score are significantly better. This behavior can be better understood using Table 4.3. From the table, it can be observed that the model trained on augmented data has a higher Recall score and F-Score and approximately equal Precision score compared to the one trained on non-augmented data.

Figure 4.2 shows why the model is confused for characters A, M, and N. The different ways in which characters are represented in the two datasets is the reason for the bad performance. A similar problem also arises in the Handwriting Recognition AI system.

4.2.2.3 Conclusion

The Ensemble model worked the best in this dataset also. The performance gained by using Data Augmentation was evident but small.

4.2.3 Webcam video

The last evaluation of the models was done on a video captured using a Webcam. The translation of the captured video was done in real-time and the translated text was shown as subtitles. The results of this evaluation will help determine the usability of the developed AI

	Basic CNN Model			Basic CNN Model with Data Augmentation		
Alphabet	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
A	0	0	0	0.157	1	0.271
B	0	0	0	0.381	1	0.552
C	0	0	0	0.317	1	0.481
D	0.5	1	0.667	0.289	0.344	0.314
E	0	0	0	0.395	1	0.566
F	1	1	1	0.5	1	0.667
G	0	0	0	0	0	0
H	0	0	0	0	0	0
I	1	1	1	0.865	1	0.928
J	0	0	0	0	0	0
K	0	0	0	0.273	0.375	0.316
L	0	0	0	1	0.188	0.316
M	0.031	0.031	0.031	0	0	0
N	0	0	0	0	0	0
O	0.842	1	0.914	1	1	1
P	0	0	0	0	0	0
Q	0	0	0	0	0	0
R	0.238	0.312	0.270	0.227	0.156	0.185
S	0.199	1	0.332	0	0	0
T	0	0	0	0	0	0
U	0.421	1	0.592	0	0	0
V	0.5	1	0.667	0	0	0
W	0.592	1	0.744	0	0	0
X	0	0	0	0	0	0
Y	0	0.625	0.769	1	1	1
Z	0.118	0.656	0.2	0.333	0.5	0.4

Table 4.3: Character-wise Metric Evaluation of ASL Alphabets

system in real-world scenarios.

Video Link: Live Fingerspelling Sign Language Translation

The live translation using the best performing Ensemble model is perfomed in the above video. As evident the AI system was able to accurately translate the Fingerspelled word ‘ANIKET’.

Scripts for live webcam translation and video translation has been provided in the project repository. The scripts also give the option to choose the model to use for translation.

Project Repository Link: sign-language-to-text-translator

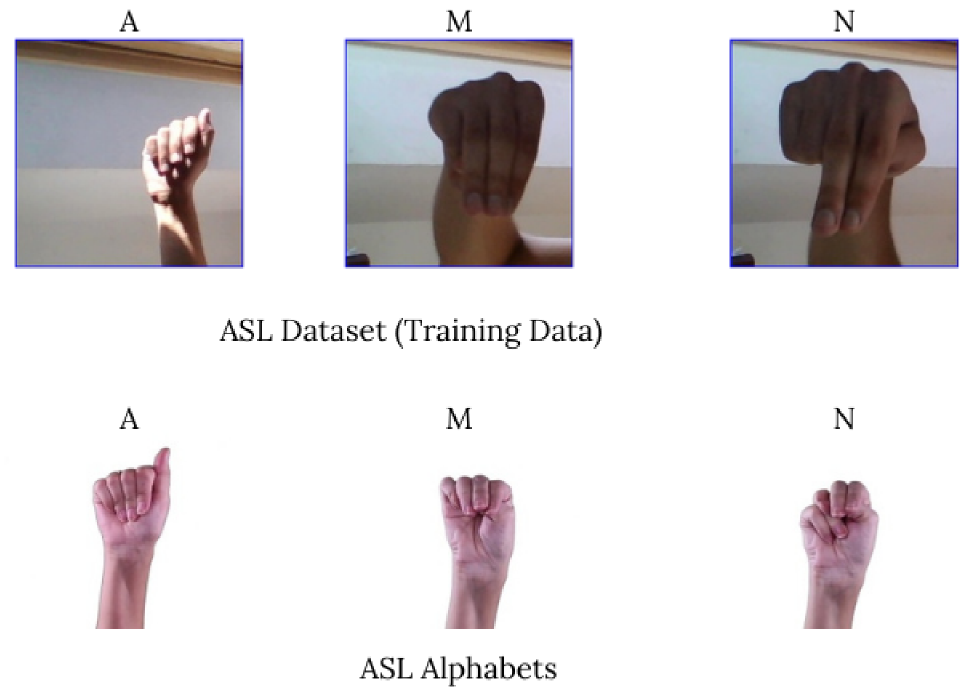

Figure 4.2: The different ways to represent same character. This is equivalent to having different hand-writings.

4.3 Overall conclusion

The Ensemble model performed best on all the datasets followed by Transfer Learning CNN. A large and diverse dataset is required to create a reliable AI Sign Language Translator. This is evident from the near-perfect accuracy on observed signers and less than 50% accuracy on unobserved signers.

Chapter 5

Discussions and conclusion

The Project Repository contains all the notebooks training notebooks, evaluation notebook and video translation scripts.

Project Repository Link: sign-language-to-text-translator

The below link contains a demo video running the *live_translation.py* script and depicting translation as subtitles.

Video Link: Live Fingerspelling Sign Language Translation

5.1 Limitations

The limitations of the project are as follows:

(1) The video to translate must contain only the hand of the signer with a white empty background.

(2) The dataset used for training was not diverse so the models are not able to accurately predict signers with different styles of representing the same symbol.

5.2 Future scope

The possible future improvements:

(1) Training the models on a diverse and large data containing different ways signers use to represent a same character.

(2) Inclusion of Object Detection techniques to detect hands in a image such that the background of the video does not affect the predictions.

(3) Adding more character like numbers and other commonly used characters.

Bibliography

[1] Akash. Asl alphabet dataset. Kaggle. Image data set for alphabets in the American Sign Language.

[2] Salem Ameen and Sunil Vadera. A convolutional neural network to classify american sign language fingerspelling from depth and colour images. Expert Systems, 34(3):e12197, 2017. e12197 EXSY-Feb-16-038.R1.

[3] Laura Berg. Learn asl alphabet video. YouTube. Video containing all ASL English alphabets.

[4] Ching-Hua Chuan, Eric Regina, and Caroline Guardino. American sign language recognition using leap motion sensor. In 2014 13th International Conference on Machine Learning and Applications, pages 541–544, 2014.

[5] David M. Eberhard, Gary F. Simons, and Charles D. Fennig. Sign language. Ethnologue: Languages of the World (24th ed.), SIL International, retrieved 2021-05-15. eds. (2021), retrieved 2021-05-15.

[6] Paul Goh. Automatic recognition of auslan finger-spelling using hidden markov models. 2005.

[7] Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, and Jaka Sodnik. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. Sensors, 14(2):3702–3720, 2014.

[8] Byeongkeun Kang, Subarna Tripathi, and Truong Q. Nguyen. Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pages 136–140, 2015.

[9] Ali Karami, Bahman Zanj, and Azadeh Kiani Sarkaleh. Persian sign language (psl) recognition using wavelet transform and neural networks. Expert Syst. Appl., 38(3):2661–2667, March 2011.

[10] M.V. Lamari, M.S. Bhuiyan, and A. Iwata. Hand alphabet recognition using morphological pca and neural networks. In IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), volume 4, pages 2839–2844 vol.4, 1999.

[11] Stephan Liwicki and Mark Everingham. Automatic recognition of fingerspelled words in british sign language. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 50–57, 2009.

[12] R. Lockton and A. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In Proceedings of the British Machine Vision Conference, pages 80.1–80.10. BMVA Press, 2002. doi:10.5244/C.16.80.

[13] CAROL A. PADDEN and DARLINE CLARK GUNSAULS. How the alphabet came to be used in a sign language. Sign Language Studies, 4(1):10–33, 2003.

[14] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time asl fingerspelling recognition. pages 1114–1119, 11 2011.

[15] Luis Quesada, Gustavo López, and Luis Guerrero. Automatic recognition of the american sign language fingerspelling alphabet to assist people living with speech or hearing impairments. Journal of Ambient Intelligence and Humanized Computing, 8(4):625–635, Aug 2017.

[16] Lucas Rioux-Maldague and Philippe Giguère. Sign language fingerspelling classification from depth and color images using a deep belief network. pages 92–97, 05 2014.

[17] Bowen Shi, Diane Brentari, Greg Shakhnarovich, and Karen Livescu. Fingerspelling detection in american sign language. CoRR, abs/2104.01291, 2021.

[18] Kristin Snoddon. Wendy sandler & diane lillo-martin, sign language and linguistic universals. cambridge: Cambridge university press, 2006. pp. xxi, 547. pb $45.00. Language in Society, 37(4):628–628, 2008.

Appendix A

Abbreviations

The Appendix contains a list of all the Abbreviations used throughout this report.

Abbreviation	**Full Form**
ASL	American Sign Language
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
LeakyReLU	Leaky Rectified Linear Unit
MLP	Multi Layer Perceptron