CSE4001

Parallel and Distributed Computing

Project report

Problem statement:

Real-time translation of Indian Sign Language to English text and speech using Neural Networks and parallel computing

Team Members

19BCE1449 - Chandhru K
19BCE1482 - Bettina Shirley R
19BCE1717 - Makesh Srinivasan

Table of contents

Problem Statement

When one loses or is born without the ability to hear, they are faced with the inability to communicate with persons who do not understand their language - sign language. Another issue is that there exist around 150 different sign languages that often increase the complexity of one person being unable to learn all of them to be able to communicate with deaf or hard-of-hearing people. Moreover, networking emerges as a major issue for people who can communicate *only* through sign language, thus limiting their abilities in that area.

Proposed Solution

We intend to bridge this gap by employing a technological solution to this age-old problem by bringing Deep Learning into the picture. A convenient sign language translator needs to be accurate and produce fast results in a real-time environment, for sophisticated translation of Sign Language. Neural Networks have gained popularity over the years for tasks like image segmentation, and mimicking brain features; however, training neural networks on images requires tremendous resources and time to run on traditional CPU architectures.

The key aspect to note is that Convolutional Neural Networks (CNN) are trained on images by passing a set of filters over the image, clearly performing this operation with a 3x3 matrix on a 1024x1024 image will consume more time on CPU. However, passing the filter over the first 3x3 section of the image is *independent* of passing the filter over the next 3x3 pixels of the image. Hence it can clearly be **parallelized** with the use of **Graphics Processing Units (GPUs).** Moreover, increasing accuracy while maintaining a low run-time can also be done by **running multiple Neural Network models in parallel** on separate threads and then combining the output in the master thread.

Selected Area/ Concept from the syllabus:

- Parallelization in training the Deep Learning Model
- Programming the Graphical Processing Unit for Deep Learning using Compute Unified Device Architecture (CUDA)
- Multithreading for running multiple Neural Network models in parallel.

Abstract

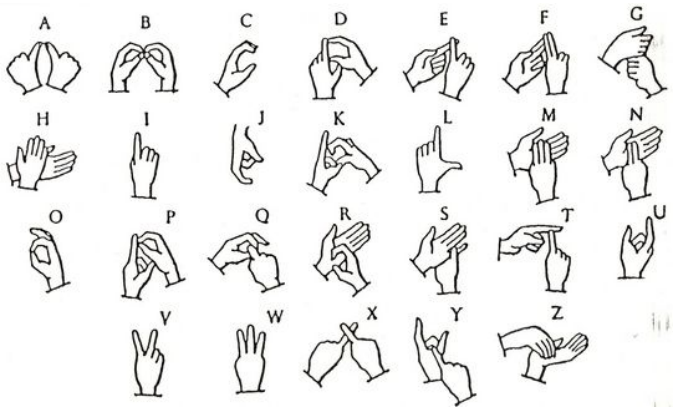
The problem at hand here falls under the paradigm of classification, more specifically, a multiclass classification. Sign language contains not just hand gestures for letters, but also for commonly used words and phrases such as ‘Hello’ and ‘Thank You. The starting point for the same begins at developing a model to help deal with categorization of the signs to the corresponding alphabets of the English language. To do this, we first manually create the dataset for the various signs, since the dataset that is manually created is not large enough to train a model, the dataset is augmented with noise added to create a more diverse and larger pool from which the model could now be trained.

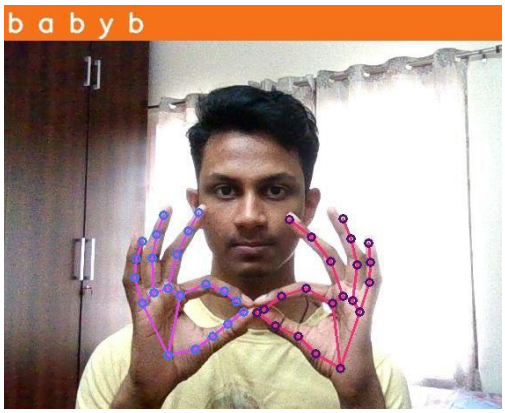
At a high level of abstraction, the idea is to feed the model with the visual input and await the classification of the image to a particular letter of the alphabet. To get into the workings of this model the input image is flattened into an array of numbers and this helps the model understand the image. The Convolutional Neural Network is trained on many images of such with their labels. The process is supervised. Patterns are recognized using filters within the model which help it differentiate different gestures.

The CNN works on each kernel (standard matrix of the array of numbers) independent of the other and many kernels can be worked on in tandem. GPU plays the major role of aiding in this parallel computation. This helps in faster computation thus making the entire process of training the model and the subsequent classification tasks simpler.

Motivation

Anyone who has read Helen Keller’s “The Story of My Life” has empathized and understood the challenge that people with special needs face. The very normal and mundane things of life such as answering a question that the professor asks or simply communicating for purchasing a ticket in a bus or train becomes troublesome. With the





amount of technology we have and the knowledge we possess, we attempt to bridge the gap and smoothly aid in their communication.

Image 1: ISL alphabets and detection/translation to English

The finished product of this project is envisioned to be an application that can be set up (in tablets) in public places and also used on phones, so whenever they feel the need to communicate, they can just as simply open the app and sign in to it to give them their voice. This could help them socially and help them to completely transition into normal society.

Literature survey

Considering parallelization on CPUs, the number of computation threads are recommended to be not larger than the number of physical CPU cores. Since it needs additional CPU resource to do the thread scheduling during the computing progress, if the CPU resources are all exploited for computation, it is difficult to achieve higher performance [1] GPUs on the other hand have large parallelization capacity, however the batch size in training has to be varied accordingly as shown in [1] by Shaohuai Shi et al.

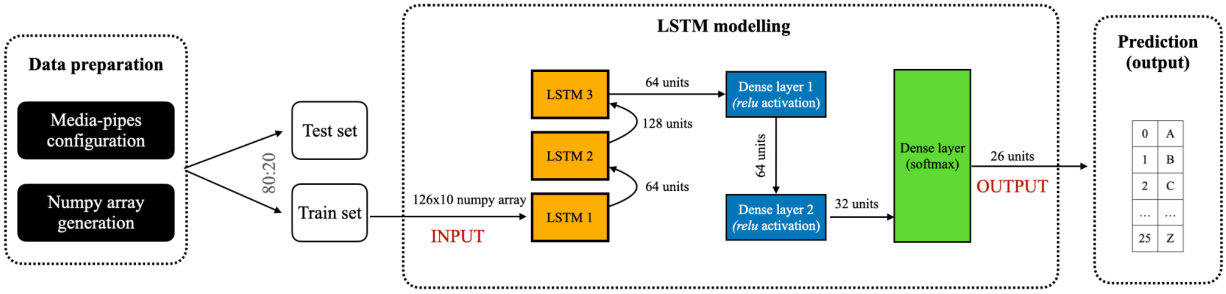


Image 2: LSTM architecture

A mini-batch size of 128 training records gave the highest Time per mini batch size (in secs). CPU training of LSTM resulted in the following: "The lower the better" mini-batch size wise, Alex-Net showed optimal training time at batch size of 16 & for FCN-S under similar conditions the batch size was 64. Hence their results made it clear how useful GPU training can be in Neural Networks and also the inference process is much faster.

```
def LSTMCell (prev_ct, prev_ht, input)
    combine = prev_ht + input
    ft = forget_layer(combine)
    candidate = candidate_layer(combine)
    it = input_layer(combine)
    Ct = prev_ct * ft + candidate
    ot = output_layer(combine)
    ht = ot * tanh(Ct)
    return ht, Ct

ct = [0, 0, 0]
ht = [0, 0, 0]

for input in inputs:
    ct, ht = LSTMCell(ct, ht, input)
```

Image 3: LSTM algorithm

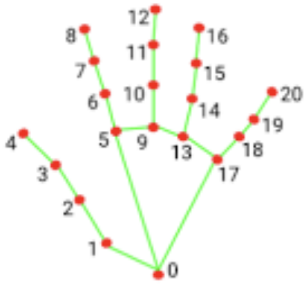
Sign language yields a complexity of postures which makes the detection a complex problem, to break it down and derive the meaning from the correct generation of static postures is a challenge. The training of deep neural networks takes place in a layer-wise manner, the abstract features from the collected signs are grouped as primary features in the first layer and then passed on to the next layer. This process continues until the distinct features are solidified over many layers and thus used in the detection of different signs [2]. The convolution neural network depends on the three major attributes - local receptive field, weight sharing, and sub-sampling. The convolution and sub-sampling layers possess 2D feature maps. In [3], max-pooling was applied for extraction from feature maps; this was done by getting the maximum activation over non-overlapping kernel regions. Max pooling is also beneficial because it samples the input image by a factor of the dimensions of the kernel thus leading to a faster convergence rate. In comparison to other pre-existing models, this model performed better and gave a 96% accuracy rate.

In [4], an ambitious model for both hand gestures recognition as well as upper body recognition where two CNNs are deployed for each of the purposes and combined using an ANN. Max-pooling is applied for faster convergence and the model yields an accuracy of 91.86%. Kinect by Microsoft has been beneficial for the development in this field, it uses an RGB camera, a depth sensor, and a multi-array microphone. It has been a vital part of [5] which uses CNN with subsampling done by performing average-pooling on the feature map in the previous layer, evidently reducing the dimensionality. The CNN used is LeNet-5 with 1024 input units (resolution of images - 32×32 pixels) and the accuracy of classifying the hand postures is 94.17%.

In another CNN model[6] developed using 35000 input images of resolution 128 x 128, the model was built using max-pooling and ReLU function was also applied, this yielded an accuracy of 98.80%. The ReLU function is simply $f(x) = \max(x,0)$ which helps introduce non-linearity in the data. In [7] the data is preprocessed using HSV colour space and background elimination setting the background to black before further processing of the image. The activation function at each step is ReLU and the one at the last layer is Softmax, the pooling is done by applying the max-pooling function, this gives an accuracy of greater than 90%.

Methodology

The problem at hand here falls under the paradigm of classification, more specifically, a multiclass classification. Sign language contains not just hand gestures for letters, but also for commonly used words and phrases such as ‘Hello’ and ‘Thank You. A good starting point is to develop a model to categorize alphabets of the English language. Thus, we have a class of 26 now. Due to the lack of usable data, we are manually creating this data by setting up the media pipe configuration and using it to generate the Numpy arrays for the different images. The points identified by the Numpy array are shown in Image 3.



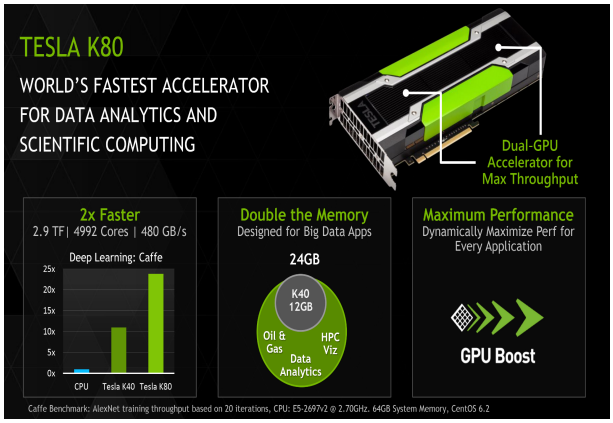
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

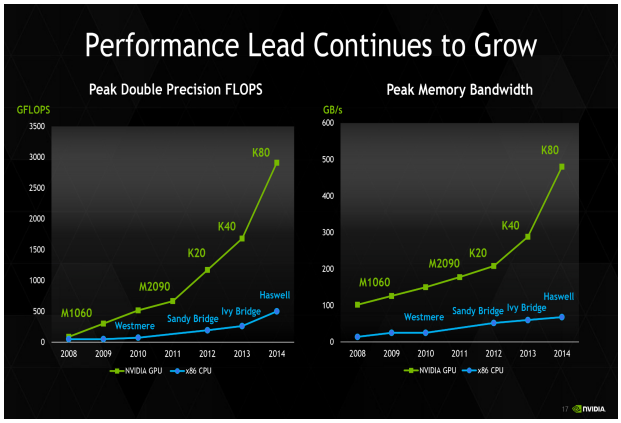
The idea is to feed the model with visual input and then decide what the subject is trying to convey. This is image classification and LSTMs are made for exactly this purpose. We flatten the image into a linear array of numbers which helps the model understand the image. The LSTM is trained on many images of such with their labels. The process is supervised. Patterns are recognized using filters within the model which help it differentiate different gestures.

Once the model is trained we can use it and check in real time as the letters are shown one after another and the corresponding output is shown on the screen. This model is more effective than using a CNN model as it doesn't have to be sliced at discrete time intervals and works as such. It works by generating the Numpy array when hands are held up and thereby determines the alphabet.

The model was trained on cloud service:Kaggle. They provide free access to NVidia K80 GPUs in kernels. This benchmark shows that enabling a GPU to your Kernel results in a nearly 13X+ speedup during training of our chosen CNN deep learning model, requiring way less time as the inputs were small Numpy arrays marking coordinates of hand landmarks. We tested this kernel on Kaggle where it was run with a GPU. We compared the run-times of the same to a kernel training with the same model on a CPU here. The total run-time with a GPU is nearly 950 seconds for a full epoch set. The total run-time for the kernel with only a CPU was around 13k(in seconds). The overall speedup was 13.6842X(meaning total run-time with only a CPU is 13.6842X as long)

We limited the comparison to model training only and not the testing section of the model to compare runtime for 1 input & output classification, we see a reduction from 13k seconds on CPU to 950 seconds with a GPU. So the model training speed-up is a little over 13X. Precise speed-up changes based on different variables involved like the pipeline complexity of the input, the structure of the model itself, dimensions of the images, size of batches in an epoch of training and so on.





Roadmap

- Defining ETP framework for the project
- Creating the Dataset of Numpy array generation landmarks(of 21 points per hand) for training.
- Splitting the dataset into a train—test—validation split with enough random samples for training under each class of alphabets and numbers.
- Deciding on a LSTM Architecture to train and classify the Sign Language Images Choosing the appropriate metrics, Loss functions, Hyperparameters for the opted model Fixing the number of Epochs to train the model.
- Building and Training LSTM model.
- Save the weights after training and predict on the test set then calculate the test metrics Repeat steps 9 and 10 till optimal performance is achieved.
- Calculate the prediction time on one image and explore the possibility of real-time prediction with a video input by slicing the video at discrete time frames and predicting on a series of image frames

---

Results and Discussion

8.4.1 Experiment 1

Title: Train CNN model and perform testing & validation on the test data.

Design of the experiment:

80% of the total data was used for the purpose of training and the remaining 20% was used for testing. The algorithm chosen was CNN and the model was created using the same. The testing dataset was used to evaluate the performance of the model. The details pertaining to the CNN model is described in the figure below.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 60, 60, 32)        2432
_________________________________________________________________
activation (Activation)      (None, 60, 60, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
activation_1 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 128)               295040
_________________________________________________________________
dense_1 (Dense)              (None, 28)                3612
=================================================================
Total params: 356,508
Trainable params: 356,508
Non-trainable params: 0
```

Dataset

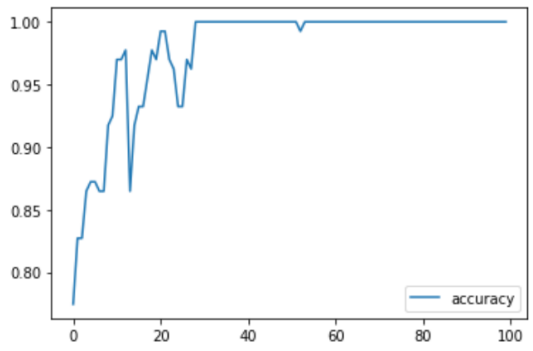
The dataset for this model involves static images of the signs for the various alphabets. All the 26 letters of the alphabet (and 2 extra characters) were generated by ourselves in accordance with the standards specified for the Indian Sign language. The dataset used for training comprised around 190 images in total.

Inferences from the experiment:

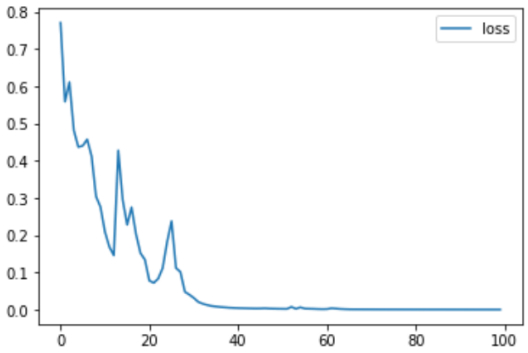
The training on the CNN model shows good accuracy when static images are given as input. Figure a shows the epoch wise classification accuracy and Figure b shows the epoch wise loss. The x axis shows the number of epochs the model has been trained and is tested against and the corresponding y axis value shows the accuracy or loss respectively. The accuracy and loss are optimum and stabilised around 100 epochs and thus is accepted as the final model after 100 epochs.

However on working with real time, the camera input should be divided into frames and then passed as input to the CNN, this would once again include exhaustive predictive modelling because the speed of the person communicating through sign language is not constant for every individual and also may vary for the same person.

After exploring the use of CNN in this project, we concluded that it will not be the most suitable Ma- chine Learning model to apply in our problem statement (Justifications provided in section 8.3). After further research, we decided to explore LSTM and thus, we began experiment 2 which is described as follows.



(a) Epoch-wise categorical accuracy value



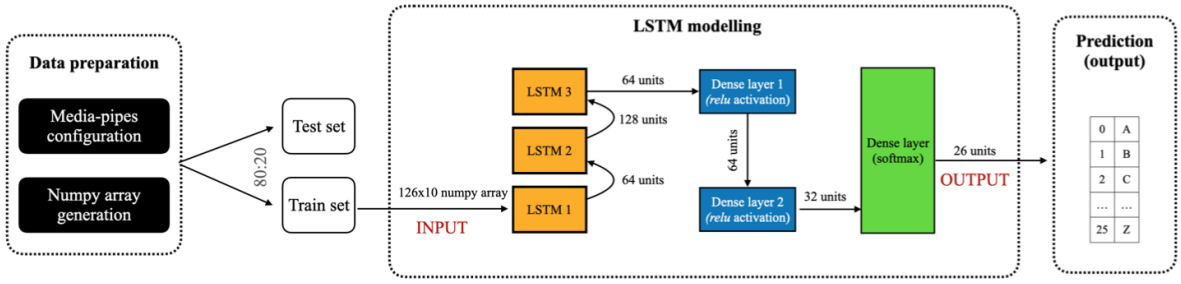
(b) Epoch-wise loss-value

8.4.2 Experiment 2

Title: Train LSTM model and perform testing & validation on the test data and real-time data

Design of the experiment:

The below figure illustrates the design of the experiment 2. We generated 10 samples for

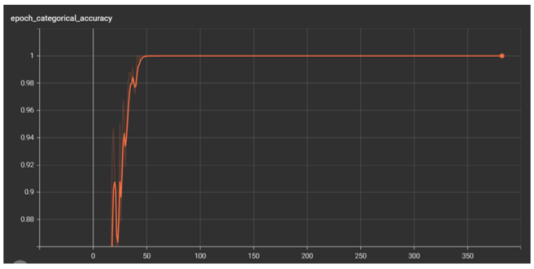


each letter and stored the coordinates of the landmarks in a Numpy array of dimension 126x10. We have 26 letters in the English alphabet, so we needed 26 of this array - one for each letter. The dataset is then split into train and test sets with the ratio of 80:20 respectively.

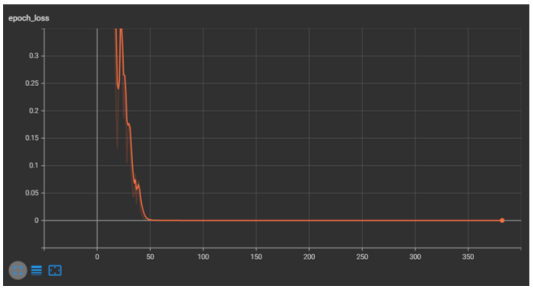
The LSTM model takes an input as a 126x10 array to train using 3 layers, 2 dense layers with Relu activation, and one softmax activation. The output of the model is 26 units (0-25) each corresponding to a letter in the English alphabet. This is depicted in the ”LSTM modelling” section in Figure 5. The first LSTM layer takes the Numpy array as input and the output of this cell (64 units) is the input to the next LSTM cell. Similarly, the output of the second LSTM cell (128 units) is the input to the third LSTM cell; the output of the third LSTM cell (64 units) is the input to the dense layer 1 (relu activation) which in turn gives 64 units as output to dense layer 2 (relu activation). Finally, the softmax layer takes the output from this layer (32 units) as input and provides the prediction as output (0-25).

Inferences from the experiment:

The training of the model using LSTM shows good real-time accuracy. It is able to detect and identify the ISL sign and show the confidence of prediction for the same. The Figure below shows the epoch- wise categorical accuracy and the epoch-wise loss value. The x-axis (independent variable) shows the epochs while the y-axis (dependent variable) shows the accuracy and loss in Figure a and Figure b respectively. The training was done over 300 epochs, and the accuracy reached 1.00 around the 50th iteration. Consequently, the loss value decreased to 0 at the same iteration. Following this epoch, the graph is saturated in both cases indicating that the maximum accuracy in training is reached. We also performed some real-time detection testing, and Figure 8 shows the prediction - the English translation of the ISL sign and the prediction confidence. The letter signed is ”b” and the prediction on the console is also ”b” which is predicted with 0.99 confidence.



(a) Epoch-wise categorical accuracy value



(b) Epoch-wise loss-value

(a) ISL input via camera

(b) English translation on console

Drawbacks

1. Convolutional Neural Networks were used to train a model that could detect sign language and translate them to English; however, it could take only static signs as input. This limits the real-world application of the software as most of the signs in ISL are gesture based and there is constant movement involved while communicating. After doing some research to overcome this drawback, we finalised on LSTM (Long short-term memory). The use of LSTM instead of CNN can help avoid the long-term dependency problem. The innate nature of LSTM to remember information for a long period of time helps us with tracking the landmarks on the hand, and detect the alphabet being shown with much higher accuracy and lower computations. The use of media-pipes is the ideal solution for real-time gesture recognition as the training data consists of only the coordinates of the landmarks, which in turn means we can train the models using only Numpy arrays and matrices. This can drastically decrease the computation time and would be suitable for real-time ISL to English translation. By parallelizing this, we can speed up the process even further.
2. The accuracy of the translation is highly dependent on the quality of the input, hence the camera on the phone must be able to acquire good quality video to be translated. One way to overcome this would be to use image enhancement techniques in case of bad quality images.
3. The speed and accuracy of translation is also dependent on the GPU and hardware of the mobile device.
4. Generic drawbacks:
    a. The application only performs translation into English. The user must be capable of understanding the language otherwise the software would be redundant.
    b. It is not very practical to pull up a phone to record or capture him/her when trying to have a conversation with a user who communicates using ISL. The most practical way to translate ISL to English would be to embed the application on a device that is seamless and organic, that does not affect the nature of the conversation. The ideal solution would be to use smart glasses that are capable of this translation in real time.

5. Transformers can be used in place of RNN, CNN or LSTM as well but the performance is not guaranteed to be better. Hence, this is not a drawback but in general, transformers are considered to be a good alternative to LSTM. Exploring this can also be useful in building a good model, but due to time constraints, we have implemented only LSTM.

Scope of improvement

- The model built currently is trained and used for the classification of the 26 English alphabets, however, ISL also uses gestures for various common phrases. The Numpy array should be generated individually for each of these phrases and thereby it should be used for the classification task.
- The GPUs in mobile devices may not be equipped to run the software necessary for the classification task, thus we need to look for a tradeoff between speed and versatility and create a software more suitable for commercial use.
- Image enhancement can help get better images as input in case of cameras with low resolutions.
- Include more regional languages: English is not a very commonly used language in India, especially in the rural areas. Including translation to regional languages such as Hindi or Tamil can be more useful to the users.
- As mentioned in the previous section, implementing the software in a device that does not affect the organic and natural conversations such as a smart glass can provide better experience to the users.

Likely Impact of the solution of the chosen problem in other related environments

The gap of communication between the disabled (deaf and mute) and others involve writing or typing. Sign Language is not exactly in the curriculum of the general population. In a world where languages are no longer a barrier, why should this be one? The project helps unlock a new sector of communication. Just like how text written in a foreign language is made comprehensible using an application, the same can be achieved using a smartphone application with a camera. Gesture recognition is not limited to just

Sign Language classification. One can program certain gestures to run different tasks like typing using an alternative to shorthand.

Even though the percentage of the population that would use this technology is low, the social impact it would have is high. Communicating to the receptionist at a hotel, ordering takeout at a fast-food chain, or just communicating with someone you meet at the park, form the very mundane portion of our life that can now be expanded to those who cannot speak or hear.

References:

[1]Benchmarking state-of-the-art deep learning software toolsProceedings Of The 7th International Conference On Cloud Computing And Big Data, IEEE, Macau, China, 2016.2016 S. ShiQ. WangP. XuX. Chu

[2] Oyedotun, O K and Khashman, A 2016 Deep learning in vision-based static hand gesture recognition. Neural Computing and Applications, 28(12): 3941–3951. DOI: https://doi.org/10.1007/s00521-016-2294-8

[3] Nagi J, Ducatelle F, Di Caro GA, Cires¸an D, Meier U, Giusti A, Gambardella LM (2011) Max-pooling convolutional neural networks for vision-based hand gesture recognition. In: IEEE international conference on signal and image processing applications (ICSIPA), pp 342–347

[4] Pigou L, Dieleman S, Kindermans PJ, Schrauwen B (2014) Sign language recognition using convolutional neural networks. In: Workshop at the European conference on computer vision. Springer, Cham, pp 572–578

[5] Tang A, Lu K, Wang Y, Huang J, Li H (2015) A real-time hand posture recognition system using deep neural networks. ACM Trans Intell Syst Technol (TIST) 6(2):21

[6] Wadhawan, A., Kumar, P. Deep learning-based sign language recognition system for static signs. Neural Comput & Applic 32, 7957–7968 (2020). https://doi.org/10.1007/s00521-019-04691-y

[7] Mehreen Hurroo, Mohammad Elham, 2020, Sign Language Recognition System using Convolutional Neural Network and Computer Vision, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 12 (December 2020)

Appendix:

The code is pushed into the GitHub repository and the request to join as a collaborator was shared with you, sir. Kindly accept to view the code.