APS360 FINAL REPORT
RECOGNIZING AND TRANSLATING SIGN LANGUAGE

Chahit Uppal
Student# 1006917400
chahit.uppal@mail.utoronto.ca

Lavanya Mehndiratta
Student# 1006819310
lavanya.mehndiratta@mail.utoronto.ca

Gunin Wasan
Student# 1007147749
gunin.wasan@mail.utoronto.ca

Shreyaansh Dadoo
Student# 1006904101
shreyaansh.dadoo@mail.utoronto.ca

ABSTRACT

Machine learning is the process of a computer learning about real-life data using various models and algorithms. Part of it includes the classification of the input data which essentially means that it divides the input into different categories of outputs and guesses what the data might be. For example, if the input data is images of animals, a possible classification could be into mammals or birds. To allow for these classifications, the model uses neural networks or a series of algorithms that implicitly understands the relationships between the data, much similar to our brain. The following sections of this paper are in relation to the progress of our project, which is to output individual sign language letters based on a live video cam.

—-Total Pages: 9

1 INTRODUCTION

We plan on building an inclusive society that allows everyone to communicate effortlessly. Using the power of machine learning, we wish to provide a meaningful product to help disadvantaged sections of our society(Khalifa, 2021). The goal of our project is to recognize sign language and translate it into written text for easier interpretation. It is important to bridge the communication gap and collectively grow as a community.

As sign language is the sole form of communication for def and mute people, we need a translator since the majority of the people don't know sign language. Our motivation is to remove the need for a translator and make communication seamless(Johnny & Nirmala, 2021). Using vision-based machine learning(Herazo, 2020), we can make sign language translation readily available in all parts of the world(Alarcon, 2019).

2 ILLUSTRATION

Figure 1 demonstrates the architectural overflow of the model. The input captures images from the live feed when it recognizes a hand gesture and processes it through the CNN Model to give the predicted letter as an output.

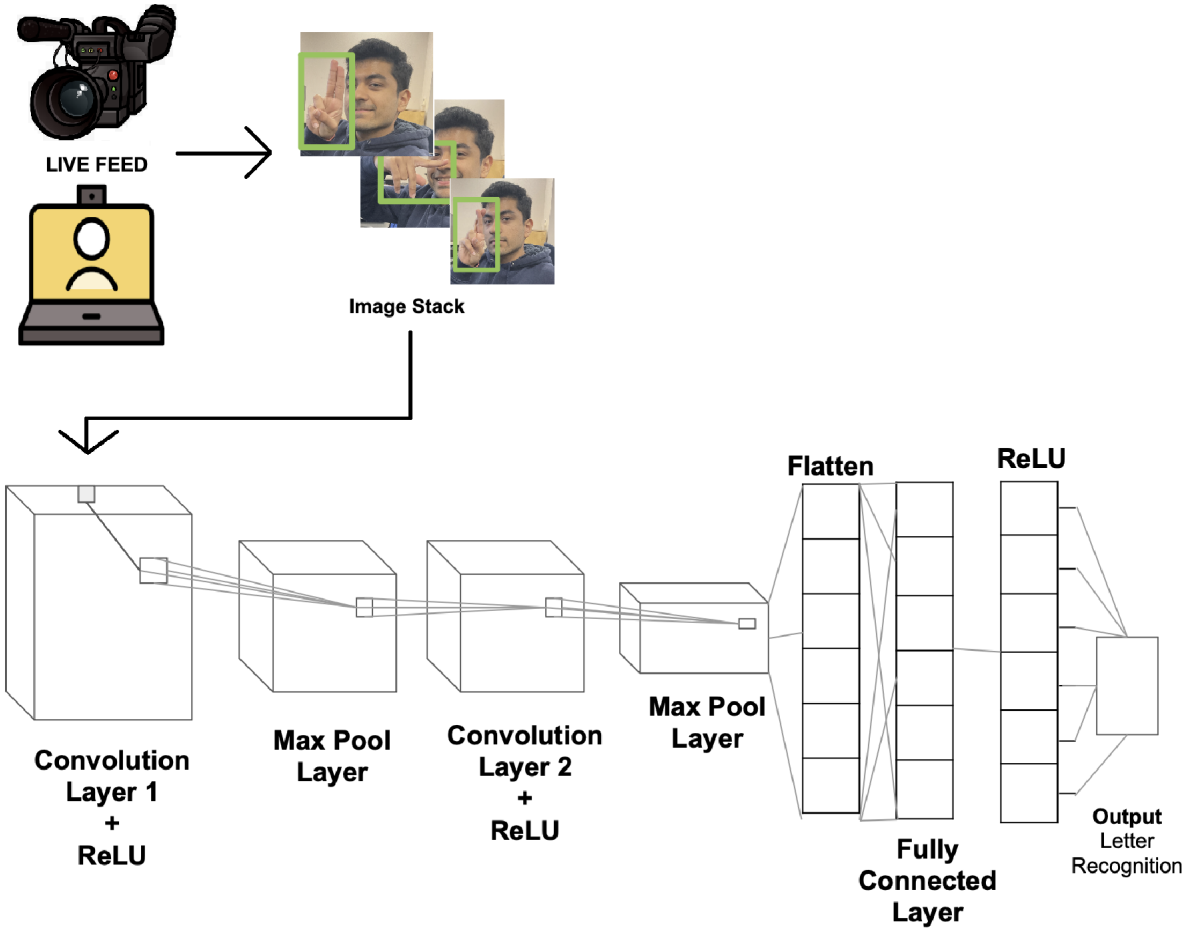

Figure 1: Architectural overflow of the model

3 Background & Related Work

Prior to conducting our project, we researched various papers to grasp a deeper understanding of the concepts. Out of those, the following five articles are the most relevant to our project.

3.1 Research Paper 1: Indian Sign Language Recognition

Machine learning was used to create a model for Indian Sign Language (ISL) recognition (Shirbhate et al., 2020). Multi-class Support Vector Machines (SVM) and the Random Forest algorithm were used for model training, along with the application of a linear kernel over each feature vector extracted from image data. Overall, this model performed at 53.23% accuracy.

3.2 Research Paper 2: Hand Gesture Recognition

A real-time, video, 3D hand gesture detection model was created using segmentation techniques and LeNet-5 which includes convolutional, pooling, fully connected, and Softmax layers (Sun et al., 2019). This research paper uses segmentation techniques to process the video frame, and identify the hand by modelling the Gaussian skin colour and using an AdaBoost Classifier (Sun et al., 2019). The average accuracy this model was able to achieve was 98.3%.

3.3 Research Paper 3: Real-Time Motion Detection

This project needed to detect and recognize the moving human body in real-time. They used R-CNN (region-based CNN), and spatial pyramid pooling to distinguish moving data and fully connected layers (Gong & Shu, 2020). After several iterations, the human-in-motion detection model works with 95.6% accuracy (Gong & Shu, 2020).

3.4 Research Paper 4: Hand Gesture Detection and Conversion

This paper discusses the detection of hand gestures and conversion to speech and text. They first capture the hand by creating a threshold of the image (Manikandan et al., 2020). They use OpenCV to create these binary images and identify the outline of the hand. This provided a "convex hull" of the hand contour, which is passed into the model for letter prediction (Manikandan et al., 2020). This way of image segmentation is very similar to our data processing.

3.5 Research Paper 5: Hand Detection from Single Colour Images

This paper discusses the detection of hands by reconstructing their appearances. An R-CNN (comprises ResNet) is used to detect the hand, followed by an asymmetric variational autoencoder (VAE) to reconstruct the hand image (Xu et al., 2019). These images are then compared to those in the original image to recognize hands.

4 Data Processing

For our data collection, we have two sources: MNIST data sets and our own hands displaying the individual sign language letters. Each sample in the data set has slightly different orientations for the same letters and is also in different backgrounds. Since a live webcam has a different frame size, our model will continuously take images from live video and resize those images to 224 x 224 pixels to match the previously collected online data sets. This is done so that there is no discrepancy between our data and the pre-processed data.

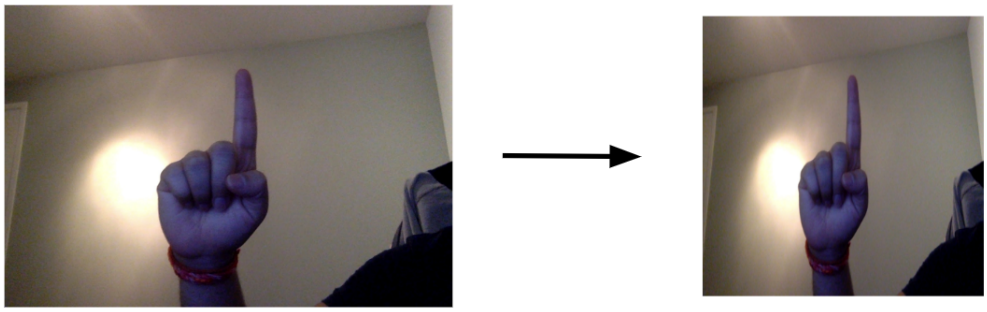

Figure 2: Resizing the user's images to 224 x 224 pixels for consistency

These photos will then be put into their respective classes and mixed with the online datasets in the training/testing folder (Figure 3). A note to make is that we are not considering letters J and Z in our data sets as they are rotational gestures and we will not be able to identify them with still images.

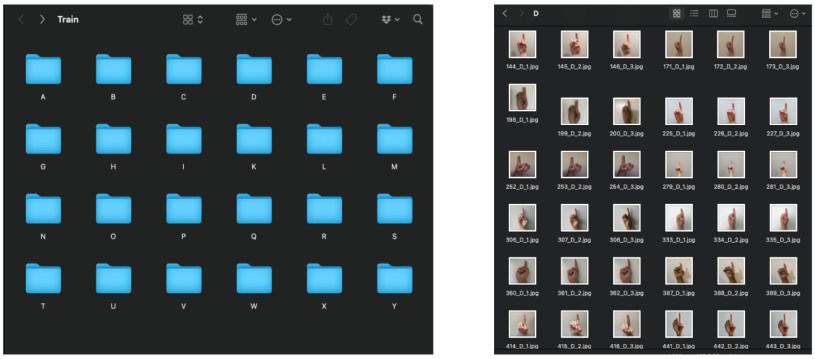

Figure 3: Example of how our data is divided into their respective classes

Each individual letter (class) in the training data set currently contains about 200 samples, making our total training samples to be about 4,800. Furthermore, to have a completely new set of testing data our team members and colleagues will be showing letters in the webcam, which will then be transformed into images until we have a total of 1,200 samples. In the end, this will give us an 80 to 20 percent ratio between the training and testing data respectively, which is a common split in machine learning today.

5 Architecture

We have chosen convolutional neural networks (CNN) to train our model as it is proven to be accurate for image classification. In our CNN model, we are using a total of 6 layers: 2 convolutional layers, 2 pooling layers (each after a convolutional layer), and 2 fully connected layers at the end.

Our Gesture Classification model can be broken into two main components: a Convolution Neural Network and a Fully Connected Neural Network. Our first convolutional layer takes in the image with 3 dimensions, performs convolution with 5 kernels of size 10x10, and outputs a feature map with 5 dimensions. The max pool layer is applied to this feature map with a size of 2x2 to extract the maximum feature values. This feature map then goes through a second convolutional layer with 5 kernels of size 5x5 which outputs a feature map that also goes through the max pool layer. We flatten the final feature map and pass it into a fully connected network with one hidden layer to classify our 9 classes of letters A to I. The logits (raw output) are then passed into the reLU function to obtain probabilities as outputs for each letter A to I.

6 Baseline Model

A baseline model is used to provide a basic accuracy test with about 10% of effort for 90% of value (Li et al., 2020). To ensure that our CNN network is providing adequate results, we will be comparing it with a simple Artificial Neural Network (ANN) model. Essentially, ANN acts similarly to CNN in terms of its forward-passing functions, but it only uses fully connected layers. Given 2-dimensional inputs, the network outputs a 1-dimensional vector. A figure representation is given below.

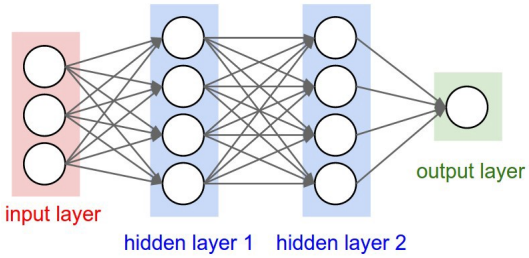

Figure 4: A simple outlook of how our baseline model works (Dabbura, 2018)

This model provided us with a basic expectation of how CNN will work. Since CNN is a subset of ANN, it should perform better hypothetically. Having only two fully connected layers in our ANN baseline model, we achieved a training accuracy of about 60% which seems plausible even if it is not necessarily preferred for image classification (see training curve below). A noticeable feature of the graph was that the curve was very smooth and overfitting, meaning our learning rate was quite small. By increasing our learning rate, we expected to have more noise in our curve and an increase in accuracy, however, that was not the case.

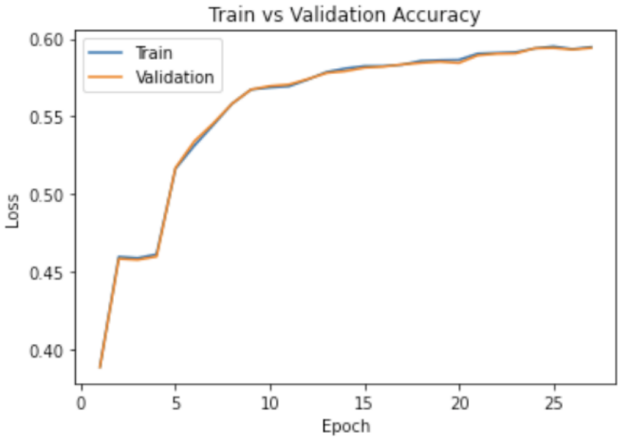

Figure 5: One of our baseline model's training curve

7 Quantitative Results

The results of the Gesture Classification CNN model can be justified quantitatively by observing the obtained loss and accuracy curves. The model will also be compared to the baseline model on the basis of their respective accuracies.

The network was trained on the MNIST data sets and the resulting Training and Validation Loss Curves can be seen in Figure 6.

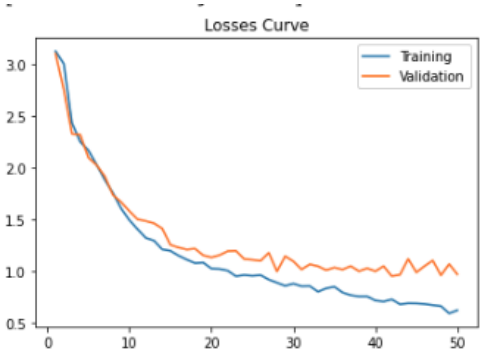

Figure 6: Training vs Validation Loss Curves Accuracy

From the graph, it is evident that as the number of epochs are increasing, the loss is decreasing; resulting in better accuracy. Aside from observing the validation-loss curves, the team compared the results against the ANN baseline model. It can be observed from the graph in Figure 5 that the accuracy of the baseline model is roughly 60% and our model outperforms the baseline model with approximately 75% accuracy (as seen in Figure 7).

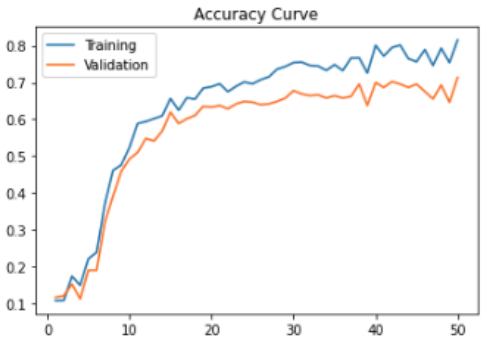

Figure 7: CNN Model's training curve

8 Qualitative Results

The results of the model can be measured quantitatively by displaying some of the sample outcomes obtained while testing input data in the model. The team believes this data will help understand and correlate with the results obtained quantitatively. Hence, the team passed some unseen testing data into the model as shown in Figure 8.

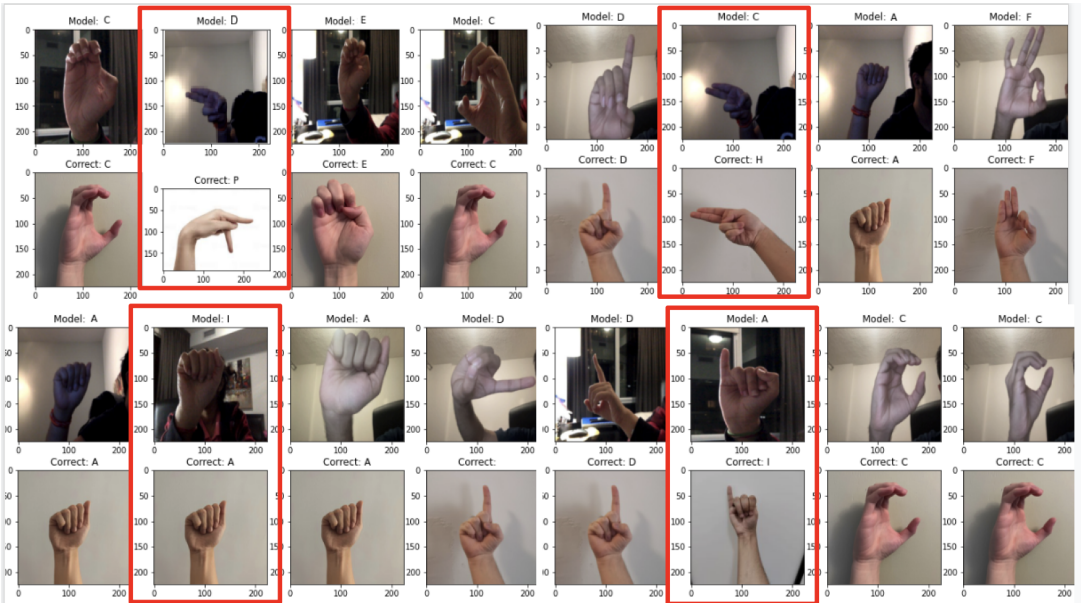

Figure 8: Incorrect Predictions by the model

It can be observed from Figure 8 that the model correctly predicted the letters three out of four times, giving us the 75% accuracy that we obtained while analyzing results quantitatively.

So, the results we observed from the sample outputs and the accuracy curves complement each other and help us correctly identify the model's accuracy and connect the qualitative results with the quantitative results.

9 Evaluate model on new data

We used a live webcam to collect our testing data. By capturing live images (from video), we were able to generate fresh, unseen data samples with variable backgrounds and lighting. As we

were generating our own data for testing, we made slight yet significant changes to the images by displaying the sign symbols in different orientations. Our newly created data was generated under average room lighting and with average camera quality to replicate the environment in which our model will be working. The generation of our test data is shown below:

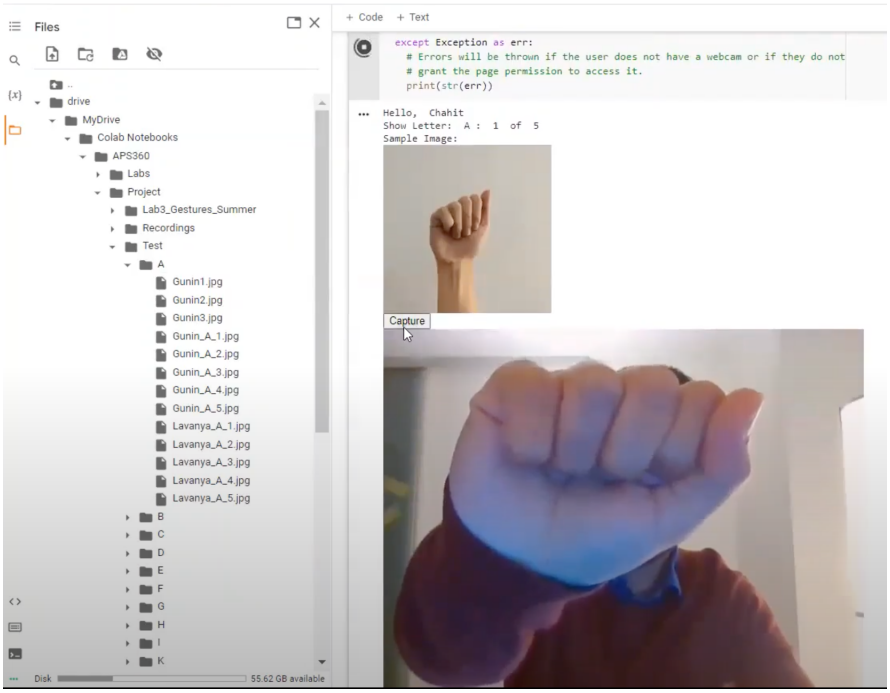

Figure 9: Example of how testing data is taken

We used OpenCV to capture live images from our bedrooms under average household conditions. We created test dataset folders where the captured image was directly imported. Our code will iterate through the different classes (letters) and display which sign language letter we have to show. We decided to capture five images per letter for each member of the team. By building an easy-to-run code, we decided to generate more samples by asking our friends and family to run OpenCV from their workstations. We generated about 1,200 fresh images in addition to the MNIST dataset.

We tested our models' accuracy using the get_accuracy function. However, this did not give us a visual representation of what our model is performing so we implemented a 'predict' function. The prediction function prints the test image, the model's prediction, the correct prediction and the correct sign for that letter. By looking at which images the model fails at, we can understand the weaknesses of our model.

In addition to capturing new data for testing, users can also use it to capture new live data and store it in our prediction folder. This will allow users to create and predict data on the go. Our model will make a prediction on the captured image in real-time. This allows the users to evaluate the models' accuracy in real-time and take note of how the model is performing.

By creating our own test data, we made sure that the model had not seen our test images earlier and that the hyper-parameters were not tuned accordingly. We reported an average of 75% accuracy on our fresh test images which align with our realistic expectations for the model. We need to improve the accuracy before we can take our project to the market and our next steps are briefly described below under Discussion.

10 Discussion

The team believes the CNN model performs relatively well on unseen testing data compared to the ANN baseline model. The results obtained quantitatively reported an accuracy of approximately

75%. While observing results qualitatively, the sample predictions made by the model match up with the quantitative results to give us an accuracy of 75% on average (Figure 8). Moreover, when testing the model against the test data, the team found out that there were about five iterations which showed an accuracy of 100% and four iterations with 50% accuracy. This disparity is likely due to the fact that in our training data set, the samples are almost perfect in the sense that the hand gestures are all centred, have good lighting and are clear images. On the other hand, the testing data contained issues like lighting, blurry pictures and unrealistic rotations on the sign symbols (i.e. rotating our hand more than +-30 degrees when creating testing data). Such differences would greatly affect the accuracy, which was seen in quantitative results. With that said, the team believes the reported accuracy rate needs to be improved. The ASL sign language project is now a well-known model with a surplus of online data sets such as the MNIST data set, and hence, with an abundance of data in our training model, we expected our accuracy to be 90% and above on average. However, our results had a great discrepancy from this expectation. Given the differences between our training and testing data, we can improve the accuracy rate in two possible ways:

We can ask a sign language expert to guide us in making our test data set. An ASL sign language expert can help avoid issues like showing inaccurate hand gestures or the model imprecisely capturing the letter to be predicted. Another solution is that we can expand our training data set to have pictures that incorporate the different backgrounds, dim lighting, and quality issues mentioned before.

11 Ethical Considerations

It is important to ensure that the data set we create does not contain any signs that might be inaccurate or offensive to the marginalized deaf-mute community. The input to this model will be a live video from which we will take still images to process and produce an output. This raises a privacy concern, as we must ask for consent before using the video in our modelBragg et al. (2021). We will not use the videos/images of those who do not provide consent for it. Overall, these ethical considerations are important as the model makes generalizations from the data set we create, which needs to be unbiased and accurate.

12 Project Difficulty / Quality

The overall difficulty of this project can be considered medium. It is not necessarily the training model or the data collection that causes the added complexity but rather the video translation of the hand signs. Our original goal was to perform predictions on a live feed where the results would show on the hand itself. However, this was difficult since our program runs on the cloud, Google Colab and connecting to the webcam is not as straightforward as it is locally. To overcome this issue, we had to combine JavaScript and Python together in order to access the user's webcam. Then, instead of having simple images as input to the model, we used the live video to detect the hand gesture and capture it into an image which was then forwarded into our model. Hence, understanding how to use OpenCV to perform the recognition and then use that data caused complexities in the project.

Additionally, even though the project is of medium difficulty, the quality in which it is done is high. Giving the user sample images of letters they can show and stating when the data is being inputted, allows them to understand where they are in our data processing. Overall, hand gesture recognition is a difficult project in the sense that it is difficult to abstain high accuracies. Hence, with attempts of multiple models (e.g. ANN, SVM, CNN) and continuous tuning of various hyper-parameters (e.g.learning rate, batch size, optimizers, number of epochs, etc), our model was able to achieve an accuracy of 75% which we consider to be high quality.

13 Link to GitHub or Colab Notebook

`https://colab.research.google.com/drive/18vmaova11vL0HpefB6HYIz4tY0_AOCyd?usp=sharing`

REFERENCES

Nefi Alarcon. Ai can interpret and translate american sign language sentences. *Technical Blog*, 2019. URL https://developer.nvidia.com/blog/ai-can-interpret-and-translate-american-sign-language-sentences/.

Danielle Bragg, Naomi Caselli, Julie A. Hochgesang, Matt Huenerfauth, Leah Katz-Hernandez, Oscar Koller, Raja Kushalnagar, Christian Vogler, and Richard E. Ladner. The fate landscape of sign language ai datasets, Jun 2021. URL https://dl.acm.org/doi/abs/10.1145/3436996?sid=SCITRUS.

Imad Dabbura. Coding neural network — forward propagation and back-propagtion, Mar 2018. URL https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76.

Meimei Gong and Yiming Shu. Real-time detection and motion recognition of human moving objects based on deep learning and multi-scale feature fusion in video. *International Research Journal of Engineering and Technology*, 8:25811–25822, 2020.

José Herazo. Sign language recognition using deep learning. *Towards Data Science*, 2020. URL https://towardsdatascience.com/sign-language-recognition-using-deep-learning-6549268c60bd.

S. Johnny and S.J. Nirmala. Sign language translator using machine learning. *Intelligent Systems*, 2021. URL https://rdcu.be/c0Frg.

Ahmed Khalifa. Is deafness a disability? *HearMeOut*, 2021. URL https://hearmeoutcc.com/deafness-as-a-disability/.

Dennis Li, Euxhen Hasanaj, and Shuo Li. Baselines. *Machine Learning Blog — ML@CMU — Carnegie Mellon University*, Aug 2020. URL https://blog.ml.cmu.edu/2020/08/31/3-baselines/.

K. Manikandan, Ayush Patidar, Pallav Walia, and Aneek Barman Roy. Hand gesture detection and conversion to speech and text. *Department of Information Technology, SRM Institute of Science and Technology*, 7:1–5, 2020.

Prof. Radha S. Shirbhate, Mr. Vedant D. Shinde, Ms. Sanam A. Metkari, Ms. Pooja U. Borkar, and Ms. Mayuri A. Khandge. Sign language recognition using machine learning algorithm. *International Research Journal of Engineering and Technology*, 7:2122–2125, 2020.

Jing-Hao Sun, Jia-Kui Yang, Ting-Ting Ji, Guang-Rong Ji, and Shu-Bin Zhang. Research on the hand gesture recognition based on deep learning. *2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*, 2019.

Chi Xu, Wendi Cai, Yongbo Li, Jun Zhou, and Longsheng Wei. Accurate hand detection from single-color images by reconstructing hand appearances. *MDPI*, 20:2122–2125, 2019.