Median American Sign Language Interpretation Software

Joshua Anderson*, Augustana College, USA
Rebecca Casad, Augustana College, USA
Charles Cathcart, Augustana College, USA
Anh Nguyen, Augustana College, USA
Tauheed Khan Mohd, Augustana College, USA

Abstrat: The Median American Sign Language Interpretation Software (ASL) Interpretation Software is a web application that is capable of interpreting American Sign Language in real-time, utilizing an internet connection and a primary web camera, complete with basic phrases and letters. Extensive use of Deep Learning and Neural Networks, specifically Convoluted Neural Networks, enables Median to interpret video inputs and generate accurate results directly displayed to the user in text format. The ultimate goal for Median is to have it act as a bridge between hearing people and members of the deaf community, allowing deaf people to communicate with non-signing people using American Sign Language. Furthermore, Median has been designed to benefit people who lack access to a human ASL Translator, as its format as a website allows it to be accessible anywhere at any time, giving increased availability over human interpreters. Median is designed to be a very versatile program with great potential for growth and expansion.

CCS Concepts: • **Computing methodologies** → **Activity recognition and understanding**; • **Software and its engineering** → *Open source model.*

Key Words: Deep learning; Convoluted Neural Networks; LSTM; MediaPipe; Google Cloud; Object detection; Classification

1 INTRODUCTION

Historically, people who are deaf or hard of hearing have long been fighting an uphill battle to communicate their thoughts and ideas in our society. Going as far back as the eighteenth century, the deaf often found themselves isolated from the hearing community, living in special districts and cities with other deaf and hard of hearing people, with a limited ability to communicate beyond their community. There were some hearing people who were familiar with sign language, but they were far and few between. Gradually, the enlightenment of the eighteenth century and the nineteenth century saw increased opportunities for deaf and hard of hearing people, and it saw the rise of the first official signed languages, namely French sign language and American sign language.

*All authors contributed equally to this research.

Authors' addresses: Joshua Anderson, joshuaanderson18@augustana.edu, Augustana College, 907 17th St, Port Byron, Illinois, USA, 61275; Rebecca Casad, Augustana College, 5920 West River Drive, Davenport, Iowa, USA, 52802, rebeccacasad18@augustana.edu; Charles Cathcart, Augustana College, 10093 N 1750th Ave, Geneseo, Illinois, USA, 61254, charlescathcart19@augustana.edu; Anh Nguyen, Augustana College, 489 Adams Way, Pleasanton, California, USA, 94566, anhnguyen17@augustana.edu; Tauheed Khan Mohd, Augustana College, 639 38th St, Rock Island, Illinois, USA, 60201, tauheekhanmohd@augustana.edu.

Manuscript submitted to ACM

However, the twentieth century and the rise of industrialism created serious issues for those of the deaf community. Deaf and hard of hearing people now had to compete for employment against those who were hearing and with a sudden shift away from teaching sign language in public schools in America, deaf and hard of hearing people had to fight to maintain their voice in our culture. The deaf and hard of hearing banded together to create leagues and organizations to promote the use of signed language and to help other deaf and people with hearing loss succeed in the changing world. As the century progressed, Deaf culture and sign language saw incredible growth starting in the 1960s up to the present day, as Deaf education formally switched to a signed-dominant style as opposed to an oral or lip-reading tradition. This factor, combined with a growing and widespread acceptance of sign language in colleges and universities across the country, has helped create a comfortable place for the deaf and hard of hearing in today's world.

However, while today is still a far cry from the past, there is still room for improvement. American Sign Language is perfectly suited to communication between deaf and hard-of-hearing people, but what about communicating with hearing people? The number of ASL signers is relatively unknown in the United States, but it is estimated to be a small portion of the population. As such, deaf people who frequently utilize ASL to communicate cannot expect many people they encounter in day-to-day life to know sign language. But, should this mean that they are forced to give up sign language and rely on a text-based format to communicate with the world? We would argue that the deaf and hard-of-hearing should always have the choice of their preferred medium, and Median is our attempt to give an option for deaf people and other ASL signers to use sign language to communicate with hearing people and other non-signers. By creating a web application that can be accessed and utilized anywhere with a simple web or smartphone camera, ASL signers can have access to translation services with a drastically higher ability than what a traditional human translator could ever provide. Median is built in Python and makes use of multiple frameworks, notably OpenCV, MediaPipe, and Flask, to lay the foundations of the application. The most critically-important piece of this application by far is the usage of deep learning and neural networks, which powers the image recognition and pattern detection. Our neural network is trained with images of team members signing letters and words from multiple angles, giving Median a higher margin for user input error and a high degree of accuracy. With this trained neural network at its core, Median will be able to view user video input and match signs detected in the video to our training dataset and display a text output with high speed and efficiency. The outcomes and use cases for Median are myriad, such as potential usage as a tool for training in American Sign Language, but Median in its current form is satisfactory to us as a translator with high availability and accuracy, a perfect tool for ASL translation in times and places where a language barrier exists between those who wish to communicate.

2 RELATED WORK

The communication between humans is really simple and straightforward. However, it is extremely difficult to communicate between people and machines because, while machines are aware of all the languages that humans can speak and comprehend, they are unable to communicate with that knowledge or data. Therefore, Hand gestures recognition has become an important and difficult task in the realm of human computer interaction. [23] demonstrates that a slightly better precision of the facial landmark detection is achieved using the deep learning based method. Different deep learning methods, such as Convoluted Neural Networks or LSTM (Long short-term memory) had made this process easier and more efficiently [21]. In the past, researchers have tried implemented several other Machine Learning models such as K nearest neighbors (KNN), Logistic Regression, Naïve Bayes Classification, Support vector machine (SVM) but have found that a CNN models provide the best accuracy [5]

Also using a CNN approach, [4] looks at hand gestures that have a certain uniformity to them — they focus on gestures that can be comprehended by any human without any expertise. Using TensorFlow and Keras, [4] build a CNN model to classify human emotions based only on hand gestures, and was able to achieve a higher than 90 percent accuracy. [29] also attempted to use CNN for hand recognition tasks.

The study of hand gestures for human-computer interaction does not, however, lay exclusively within just comprehension of signed languages. [8] Rempel et al, in an article for the International Journal of Human-Computer Studies, demonstrates extensive research done into the relationships between human cognitive and motor processes and computer gesture recognition, for the purpose of communicating directly with a computer, and found that a significant concern is the differences in the bio-mechanical abilities within each individuals range of motion. The more frequently a sign is formed with the hand, the more likely it is that the individuals ability to produce that same sign will decrease.

[9] took a novel solution to this challenge when they developed smart gloves that can recognize hand motions and transcribe them into text that can be read on smartphones or LCD screens. These gloves can recognize 20 of the 26 alphabets with a 96 percent accuracy rate. However, using this technique, both the system's complicated hardware and software must be considered. The gloves' hardware includes five flex sensors, five contact sensors, one three-dimensional accelerometer, and one three-dimensional gyroscope, as well as a Bluetooth module for wireless signal transmission and an Arduino microcontroller. These technology are expensive and are not easily accessed by the general public

Many other works have utilized MediaPipe for it's high-end body tracking. Similarly to our project, [15] used MediaPipe and Python as a way to track the body. While we used it to track hand movement, they created a AI body language decoder. Their project is able to help detect facial expressions, hand gestures, and body poses. MediaPipe provided them with 33 2D body landmarks , 21 3D hand landmarks, and 468 3D face landmarks. Applications of the body detection they mention are as follows, driver drowsiness detection, sign language detection, market research companies can use this technology to analyze data, and body language detection in interviews. Their results successfully contained hand, body, and face tacking. They were able to use the facial detection to help interpret emotions based on facial expressions. Their project shows how we useful this technology can be when successfully developed while also bringing to light how much more can be done with this field.

The *Mobile Information Systems* journal released an article in December of 2021 that explored the possibilities of using CNN networks with advanced training techniques to further increase the accuracy of sign language detection in real time. [14] By using a convolution layer to separate the layers that focus on detecting edges, corners, and backgrounds, they were able to detect the values of the non-important elements of an image. This creates a frame around the hand sign itself during detection, allowing the model to more accurately and consistently train and predict during real-time detection. They were able to produce a 91.07 percent accuracy of detection and prediction for still gestures and signs with unique hand positioning.

The greatest obstacle for a sign language interpreter, according to [10], has always been recognizing when the complete body movement is involved instead of just the two hands . When dealing with these extremely important yet overlooked societal concerns," [2] simply experiment with static hand signals. We want to address this issue in our application by transforming all bodily parts into points and evaluating these points.

In a similar line of work, a group, at Silicon Valley AI Lab, in 2015 had worked on a deep learning speech recognition in English and Mandarin. Their goal was to use deep learning to create an end-to-end speech recognition software that was able to recognize either English or Mandarin Chinese speech due to the vast differences in these languages. End-to-end learning allowed them to accomplish speech recognition in a variety of circumstances such as noisy environments, accents, and different languages [3].

3 EXPERIMENTAL SETUP

The proposed project focuses on the areas found to be lacking in other similar projects found during research. For example, [17] the Loeding et. al paper on computer sign language recognition focuses exclusively on finger spelling, or using specific, primarily stationary hand signs to identify individual letters. The recognition of finger spelling of individual letters is significantly less technically difficult, as the hand mapping is stationary, and does not require facial mapping information in order to be recognized. The technical ease, however, requires a trade off in usability, finger spelling makes up only a small portion of sign language.

Fig. 1. Median Key Points

In studying Pakistan Sign Language analysis through the lense of computer vision, Bilal Hassan et al. addresses the common concerns related to the most accurate computer vision sign language interpretation programs that have been developed. Namely, the most accurate sign language analysis comes from the use of physical hardware- in most cases, a data glove, to measure hand and finger distances in order to accurately measure the data points of the hand. Hassan

discusses further the lack of consistency through computer vision specific sign language interpretation development, which has a large margin of error with each individual hand sign. [12]

Numerous sources have combined the finger spelling abilities of their predecessors with more complicated and common signs, which involve multi or single hand movement for individual, non-character communication. The hand sign 'yes', for example, requires not only the recognition of finger placement and location, but the also requires analysis of movement and direction, in order to be accurately identified. This, however, still fails to address an important element of sign language recognition that is essential to sign language speakers and translators- the use of facial and body movement to identify a word.

Seeing a signed language can easily host multiple similar signs for vastly unique terms, the addition of hand and body movement allows the intent behind the word to be understood more clearly. To address the previous analysis of required data for the sign 'yes', the 'yes' sign can be signed with the addition of a nod, to indicate the term used is yes. However, the 'yes' sign can also be displayed by the shaking of the head, to indicate a sarcastic or unsure use of the term. Without the addition of facial tracking, the intricacies of signed language are not only lost, but are completely unavailable, producing communication barriers which would be significantly worse than that of a human interpreter.

To reduce these barriers, should the goal be to allow computer recognition through the use of LSTM to accurately serve as a replacement for traditional sign-language interpreters, the recognition of body movements, finger placement, hand placement, and facial recognition should all be included in the recognition and processing of sign language in real time. Although many of these elements have been attempted or succeeded for small data sets, many open source projects attempting similar work are reduced to small dictionaries of recognizable words.

The addition of the top one hundred most used sign language words and signs should therefore be the bare minimum in the assessment of sign language recognition usability, when paired with hand, facial and body movement. Additionally, the ability to improve the recognition from within the application is a secondary, but important aspect to be added to the development of *Median*. This is a two fold process, starting with the ability to add a training feature, allowing not only sign language users to practice their signs, but also allowing them to add their personal sign information to be analysed, allowing for more accurate and all encompassing recognition across the software, but also reducing the rate of human error for which the software must account. By providing additional data to a vast network of information, and creating consistency among users, it allows a multi-step solution to help address in-accuracy when it arises. The secondary positive of this feature is that it allows the user to feel an element of control over the accuracy of a system that they likely do not understand, or inherently trust to address potential concerns.

4 FRAMEWORK

4.1 Overview of Python Frameworks

During our research process, we found that Tryono et al. have used the OpenCV library and Support Machine Learning to build an Android application to interpret sign language [27]. In 2019, a research team used computer vision and deep neural networks to capture hand sign digits and translate them to Bangla [2]. This team also used the Translator and Google Text to Speech API to create a similar environment as spoken language.

MediaPipe offers a customized Machine Learning solution across multiple platforms for live and streaming data [18]. The image below shows the 21 hand landmarks that can easily be extracted using the MediaPipe package in Python.

As mentioned before, Tryono et al. have utilized the OpenCV library and Support Machine Learning to build an Android mobile application to translate sign language [27]. They examined their app with two different light settings

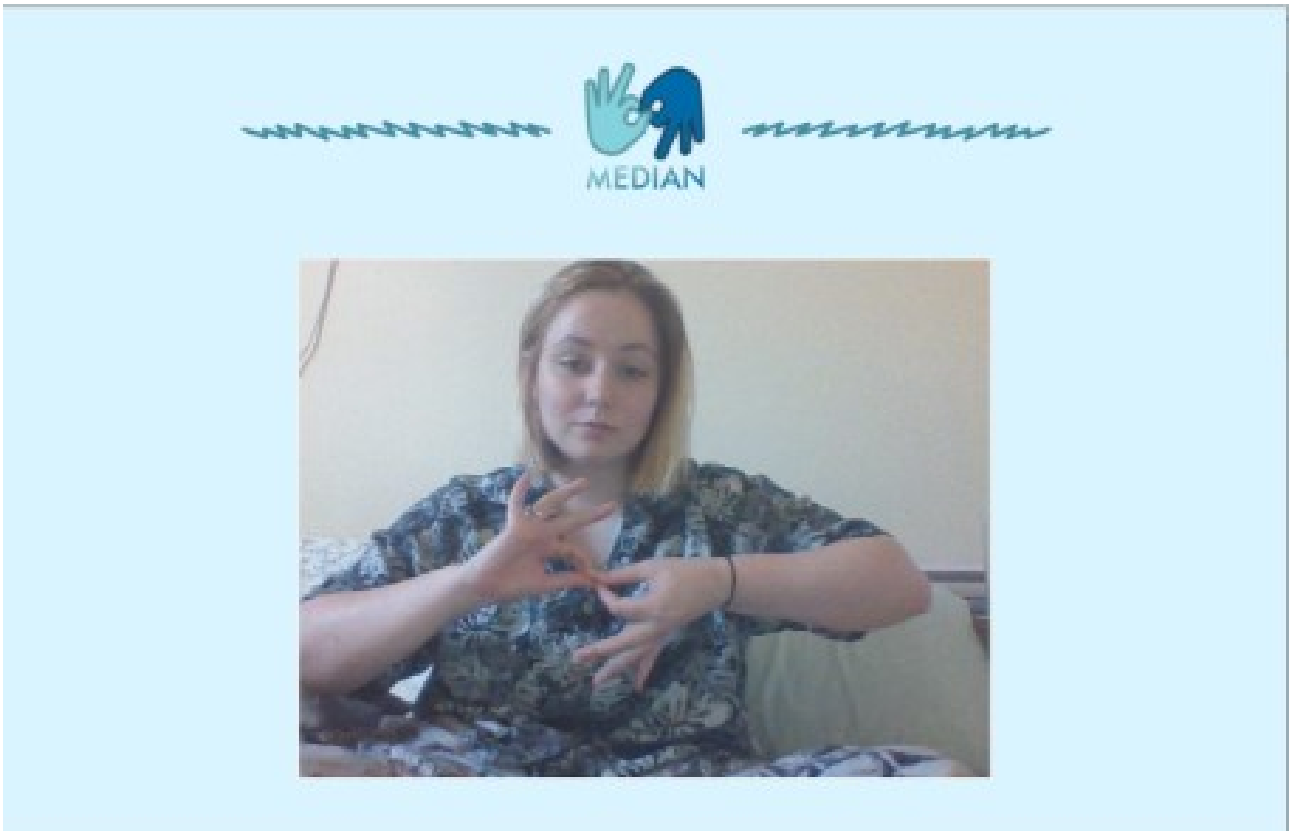

Fig. 2. Median UI Wireframe

but failed miserably. Furthermore, since they used a color difference method, the application needs some specific color background to function and does not recognize signs with too many similarities, like letters M and N. The Motion Templates on OpenCV was also highly recommended by this paper as this technique can detect motion in the smallest regions of a frame. Furthermore, OpenCV provides a conventional API for computer vision-based applications like ours [19] [7].

4.2 Overview of LSTM

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture that was created to better precisely predict temporal sequences and their long-range relationships than traditional RNNs [22] [30]. In the recurrent hidden layer of the LSTM model, there are memory blocks, that are special units [22]. Memory cells with self-connections store the network's temporal state, while special multiplicative units known as gates govern the stream of data in the memory blocks[22]. In the original architecture, each memory block had an input gate and an output gate[22]. The flow of input activations into the memory cell is controlled by the input gate[22]. The output gate regulates the flow of cell activations across the network[22]. The forget gate was later added to the memory block in order to fix a flaw in LSTM models that prevented them from processing continuous input streams without being divided into sub-sequences[22].

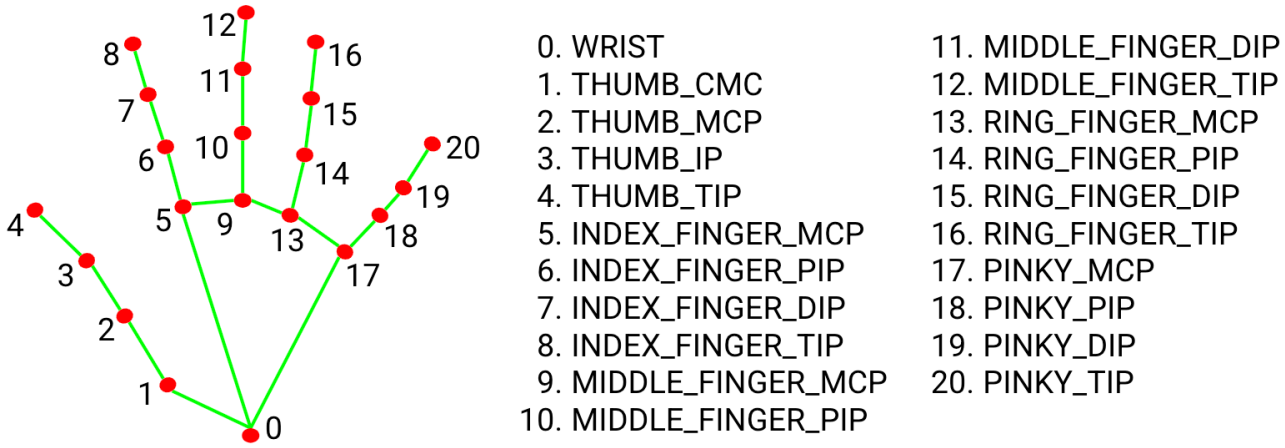

Fig. 3. 21 Hand landmarks

Early research showed that STM could be used to learn a variety of previously incredibly difficult tasks [24] and allows different architectural and hyper-parameters tuning [6]. This included retaining high precision real numbers across long noisy sequences, learning context-free languages, and a variety of timing and counting challenges [26].

4.3 Overview of Neural Network

"Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. ... Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected" [20]. Machine learning systems are used for a wide variety of different fields, and their usefulness to modern society contiunes to expand as technology develop. "Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search" [13]. Nowadays, deep learning is used in all aspects of life, some outstanding examples include self-driving cars, medical imaging, etc. More personally, deep learning is used to train virtual assistants such as Apple's Siri and Amazon's Alexa.

The introduction of Computer Vision has significantly progressed how computer engineers and scientists handle image processing problems [16]. Computer vision started in the 1960s, only around 50 years ago and has since extensively emerged into our daily lives. The main purpose of computer vision systems is to autonomously perform human visual tasks and occasionally outperform them. The rich history of computer vision, and it's growth historically, demonstrates not only the usefulness and accuracy that it has shown, but also ensures the future development of it's usefulness to expand humanities concept of automated visual identification and other similar tasks.

4.4 Overview of TensorFlow

TensorFlow was created as an interface to express machine learning algorithms and to execute them. TensorFlow can be executed on a variety of different systems and the system is flexible. The flexibility of the system allows for TensorFlow to be used in a variety of ways, a common use is in deep learning algorithms and interfaces. The implementation of a TensorFlow system has the client use the Session interface to communicate with the master and worker processes. [25] Each worker process is responsible for allowing access to different devices such as GPU cards and CPU cores.

If multiple GPU cards are installed on a device it is still possible for the system to still one as one operating system process as long as the client, master, and workers all run one machine. TensorFlow has been useful as a tool in the research and deployment of machine learning systems in a variety of fields. Some of the machine learning systems that TensorFlow has shown promise in are computer vision, speech recognition, robotics, natural language processing, information retrieval, geographic information extraction, and computational drug discovery. [1].

Our project uses TensorFlow to establish a learning algorithm that would allow for the digital interpretation of American Sign Language that is input through a camera on a given device. The TensorFlow learning algorithm is used in tangent with data from MediaPipe's hand tracking system to help accurately interpret what someone signs on camera.

4.5 Overview of MediaPipe

Our project will utilize Google's recent, rapidly-growing and open-source application MediaPipe. MediaPipe is an end-to-end hand tracking system that works in real time across many platforms. This pipeline predicts 2.5D landmarks without the use of specialist hardware, making it simple to install on common devices. They made the pipeline open source to allow academics and engineers to use it to create gesture control and imaginative virtual reality apps [31]. A hand tracking application is included with this package, which detects a hand and follows its movement using 21 predetermined landmarks on the detected hand. Each of these landmarks is a three-dimensional coordinate that has been normalized to a certain range.

We constructed a Python framework on top of MediaPipe to integrate MediaPipe into our project, and we also included hand landmark output to the MediaPipe Hand detection function.

4.6 Overview of Flask

Flask is a web microframework that is designed for the creation of web applications in Python. In a simple sense, Flask allows for the integration of both Python source code and HTML documents in order to create web applications that are simple in structure but are easy to scale up to very large applications. Flask in and of itself is very sparse in terms of extensions/additional packages, but many additional packages enabling functionality with SQL and utilization of Jinja are available, which results in Flask being extremely flexible and adaptable to any user's development needs. [11]

The use of Flask with Python as a way of web development has technical advantages to it. Flask is a robust system which helps it cope with any errors that may occur, whether that is during execution or abnormalities in input. Another advantage is that both Flask and Python are open source languages which allows the general public to make modifications to the design when necessary. Open source code also tends to have a collaborative effort which allows for other developers to improve upon and share changes with the general public. [28]

In reality, Flask has been a major boon to our development process. The above-listed advantages to using Flask have been borne out in practice, as Flask contains support through additional packages for both OpenCV and MediaPipe, removing the need to find complex and convoluted workarounds to incompatibilities between Flask and our main project's dependencies. In specific to this project, our development process has been very much an attack from both ends. Both the core programming used to detect, track and interpret American Sign Language and the eventual website hosting and user interface solutions have been in development simultaneously, with the eventual plan being to join both parts together once they have reached completion or a more stable/workable state. This marriage between both parts would not be possible without the usage of Flask. Due to Flask's high degree of flexibility and compatibility with available software packages, we will be able to essentially place our main hand-tracking source code directly into our working Flask directory, which offers us as developers a very streamlined and time-efficient process for turning

Python source code into a web application. As it stands now, this web application once complete will not be hosted nor publicly accessible from the internet, and as such will require the use of a third-party hosting solution for websites and web-applications. We have plans to make use of Heroku, one such third-party service that will allow us to host our application for free with limited features, which will for our purposes suffice in creating a real web-environment in which our project will run and be subject to rigorous tests.

5 METHODS

5.1 Overview of neural network training process

Currently, we are training a Convoluted Neural Networks to predict the hand digits. Since a CNN model is famous for its robustness, fast prediction and early predictive power, it is arguably one of the most used deep learning models This sequential model is built using TensorFlow/ Keras, consisting of 8 layers. The figure below demonstrate our layers and its parameter. The dropout layers are used to prevent this model from overfitting.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 160)               6880

 dropout_2 (Dropout)         (None, 160)               0

 dense_10 (Dense)            (None, 80)                12880

 dense_11 (Dense)            (None, 40)                3240

 dense_12 (Dense)            (None, 20)                820

 dropout_3 (Dropout)         (None, 20)                0

 dense_13 (Dense)            (None, 10)                210

 dense_14 (Dense)            (None, 26)                286

=================================================================
Total params: 24,316
Trainable params: 24,316
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. CNN Model Parameter

5.2 Overview of LSTM training process

Instead of using all the dense layers, we replaced some with LSTM layers. The LSTM model consists of 6 layers that provide a better results on this dataset

6 RESULTS

6.1 Training with alphabet-only dataset:

For the time span of this project, our application will only be able to recognize American Sign Language and translate it into the English alphabets as well as some basic words. We were able to set up a code to recognize most of the 26

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 42, 64)            16896

 lstm_1 (LSTM)               (None, 42, 128)           98816

 lstm_2 (LSTM)               (None, 64)                49408

 dense_6 (Dense)             (None, 64)                4160

 dense_7 (Dense)             (None, 32)                2080

 dense_8 (Dense)             (None, 26)                858

=================================================================
Total params: 172,218
Trainable params: 172,218
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. LSTM Model Parameter

Epoch Number	CNN	LSTM
Epoch 10	0.6877	0.9222
Epoch 80	0.8677	0.9812
Early Stopping Epoch	0.9161	0.9818

Table 1. Results summary

alphabets and numbers from 0-9. This result can provide real-time finger spelling words. Table 1 below shows the results of training effort on the 26 alphabets with 2 different types of models. Details about these results will be discussed in the next section.

The graphs below the model output for the CNN model trained to recognized 26 alphabets.

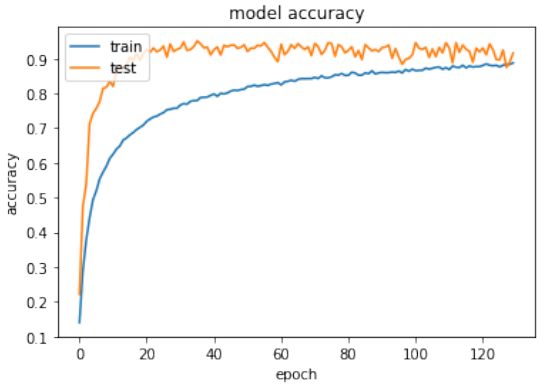

Fig. 6. CNN Model Accuracy

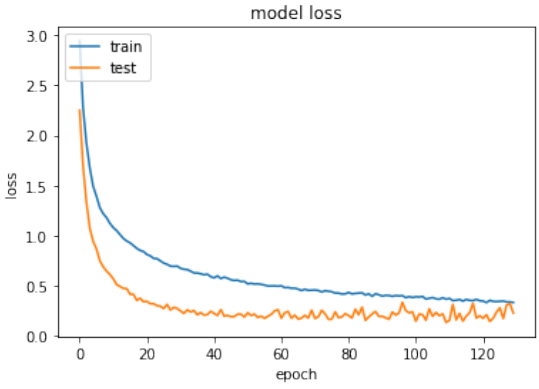

Fig. 7. CNN Model Loss

```
107/107 [==============================] - 0s 3ms/step - loss: 0.3462 - accuracy: 0.8797 - val_loss: 0.1787 - val_accuracy: 0.9257
Epoch 125/500
107/107 [==============================] - 0s 3ms/step - loss: 0.3442 - accuracy: 0.8810 - val_loss: 0.2390 - val_accuracy: 0.8975
Epoch 126/500
107/107 [==============================] - 0s 3ms/step - loss: 0.3473 - accuracy: 0.8762 - val_loss: 0.2820 - val_accuracy: 0.8968
Epoch 127/500
107/107 [==============================] - 0s 4ms/step - loss: 0.3489 - accuracy: 0.8812 - val_loss: 0.1745 - val_accuracy: 0.9253
Epoch 128/500
107/107 [==============================] - 0s 3ms/step - loss: 0.3407 - accuracy: 0.8844 - val_loss: 0.3111 - val_accuracy: 0.8747
Epoch 129/500
107/107 [==============================] - 0s 3ms/step - loss: 0.3407 - accuracy: 0.8837 - val_loss: 0.3242 - val_accuracy: 0.8907
Epoch 130/500
107/107 [==============================] - 0s 3ms/step - loss: 0.3353 - accuracy: 0.8878 - val_loss: 0.2299 - val_accuracy: 0.9161
Epoch 130: early stopping
```

Fig. 8. CNN Model Accuracy at Epoch 130

Towards the end of the training period, the CNN model has achieved a train set accuracy of approximately 0.8878 and test set accuracy of 0.9161.

The graphs below the model output for the LSTM model trained to recognized 26 English alphabets.

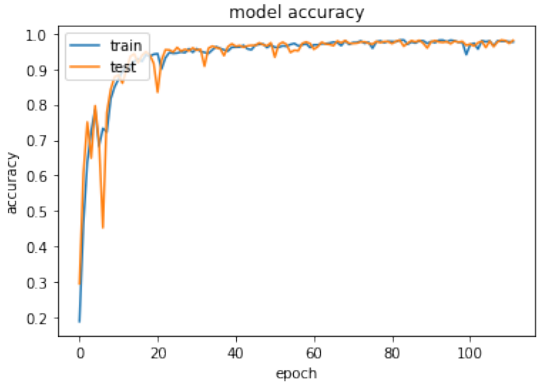

Fig. 9. LSTM Model Accuracy

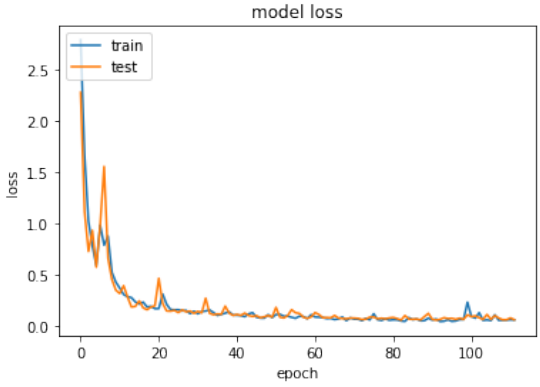

Fig. 10. LSTM Model Loss

6.2 Training with words and alphabet dataset:

Additional to the alphabets, we also record 50 words and use them to 2 models: 1 CNN and 1 LSTM. The original tests were done using only 6 sign language words which involved hand sign and movement. This was later expanded to fifty words which were the most common words used in sign language. The results of the two models are recorded in the table below:

7 DISCUSSION

After extensive training of the model with the alphabet data set, it is shown that LSTM outperformed the CNN model on this task. Therefore, we will proceed using LSTM model for our application. For future work, we will be recording a total of 100 ASL words and use this LSTM model on that dataset, as well as completely integrating the use of finger

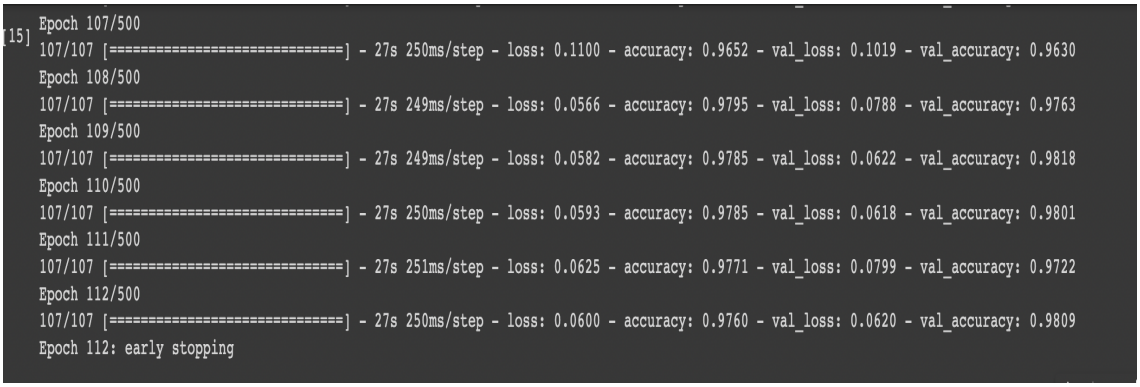

Fig. 11. LSTM Model Accuracy at Epoch 130

Epoch Number	CNN	LSTM
Epoch 100	0.5966 0.1712	
Epoch 5000	0.8596	0.2489

Table 2. Results summary

spelling individual letters when necessary to allow for proper use of the interpreter. The addition of training features to increase the accuracy of alphabet and dictionary specific signs, although not complete at the writing of this paper, are continuing to be developed, with hopes to integrate it into the LSTM model, and allow users to train themselves, and the application. The increase in accuracy with the LSTM model, with the use and functionality of a complex and distinct dictionary of usable phrases, allows the possible creation of a usable, hardware free, computer vision sign language interpreter. Although the application of this software in a marketable sense is far from the current reality, the functionality and development of *Median* aims to open the door for further accessibility and availability in the near future.

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

[2] Ahmed, S., Islam, M. R., Hassan, J., Ahmed, M. U., Ferdosi, B. J., Saha, S., and Shopon, M. Hand sign to bangla speech: A deep learning in vision based system for recognizing hand sign digits and generating bangla speech. *SSRN Electronic Journal* (2019).

[3] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G. F., Elsen, E., Engel, J., Fan, L. J., Fougner, C., Hannun, A. Y., Jun, B., Han, T., LeGresley, P., Li, X., Lin, L., Narang, S., Ng, A., Ozair, S., Prenger, R. J., Qian, S., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Sriram, A., Wang, C.-J., Wang, Y., Wang, Z., Xiao, B., Xie, Y., Yogatama, D., Zhan, J., and Zhu, Z. Deep speech 2 : End-to-end speech recognition in english and mandarin. *ArXiv abs/1512.02595* (2016).

[4] Arjun, A., Sreehari, S., and Nandakumar, R. The interplay of hand gestures and facial expressions in conveying emotions a cnn–based approach. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)* (2020), pp. 833–837.

[5] Bhavana, D., Kishore Kumar, K., Chandra, M. B., Sai Krishna Bhargav, P., Joy Sanjanaa, D., and Mohan Gopi, G. Hand sign recognition using cnn. *International Journal of Performability Engineering 17*, 3 (2021), 314.

[6] Breuel, T. M. Benchmarking of LSTM networks. *CoRR abs/1508.02774* (2015).

[7] Culjak, I., Abram, D., Pribanic, T., Dzapo, H., and Cifrek, M. A brief introduction to opencv. In *2012 Proceedings of the 35th International Convention MIPRO* (2012), pp. 1725–1730.

[8] David Rempel, Matt J. Camilleri, D. L. L. The design of hand gestures for human-computer interaction: Lessons from sign language interpreters. *International Journal of Human-Computer Studies 72* (2014), 728–735.

[9] Elmahgiubi, M., Ennajar, M., Drawil, N., and Elbuni, M. S. Sign language translator and gesture recognition. *2015 Global Summit on Computer; Information Technology (GSCIT)* (2015).

[10] Escudeiro, P., Escudeiro, N., Reis, R., Barbosa, M., Bidarra, J., and Gouveia, B. Automatic sign language translator model. *Advanced Science Letters 20*, 2 (2014), 531–533.

[11] Grinberg, M. *Flask web development: Developing web applications with python.* O'Reilly, 2018.

[12] Hassan, B., Farooq, M. S., Abid, A., and Sabir, N. Pakastan sign language: Computer vision analysis recommendations. *VFAST Transactions on Software Engineering 4* (2016).

[13] Ian J.Goodfellow, Yoshua Bengio, A. C. C. Deep learning. *Nature* (2015).

[14] Juneja, S., Juneja, A., Dhiman, G., Jain, S., Dhankhar, A., and Kautish, S. Computer vision-enabled character recognition of hand gestures for patients with hearing and speaking disability. *Mobile Information Systems 2021* (2021).

[15] Karem, S. R., Kanisetti, S. P., Soumya, D. K., Seelamanthula, J. S. G., and kalivarapu, M. Ai body language decoder using mediapipe and python.

[16] Li, Y., and Zhang, Y. Application research of computer vision technology in automation. In *2020 International Conference on Computer Information and Big Data Applications (CIBDA)* (2020), pp. 374–377.

[17] Loeding, B. L., Sarkar, S., Parashar, A., and Karshmer, A. I. Progress in automated computer recognition of sign language.

[18] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., and Grundmann, M. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019* (2019).

[19] Mahamkali, N., and Ayyasamy, V. Opencv for computer vision applications.

[20] Marr, B. What is deep learning ai? a simple guide with 8 practical examples. *Forbes* (2021).

[21] Pariselvam, S., Dhanuja, N., Divya, S., and Shanmugapriya, B. An interaction system using speech and gesture based on cnn. In *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)* (2020), pp. 1–5.

[22] Sak, H., Senior, A. W., and Beaufays, F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR abs/1402.1128* (2014).

[23] Savin, A. V., Sablina, V. A., and Nikiforov, M. B. Comparison of facial landmark detection methods for micro-expressions analysis. In *2021 10th Mediterranean Conference on Embedded Computing (MECO)* (2021), pp. 1–4.

[24] Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR abs/1808.03314* (2018).

[25] Shukla, N., and Fricklas, K. Manning Publications, 2018.

[26] Staudemeyer, R. C., and Morris, E. R. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *CoRR abs/1909.09586* (2019).

[27] Triyono, L., Pratisto, E. H., Bawono, S. A. T., Purnomo, F. A., Yudhanto, Y., and Raharjo, B. Sign language translator application using opencv. *IOP Conference Series: Materials Science and Engineering* (2018), 333.

[28] Vangala Rama Vyshnavi, A. M. Efficient way of web development using python and flask.

[29] Varun, K. S., Puneeth, I., and Jacob, T. P. Hand gesture recognition and implementation for disables using cnn's. In *2019 International Conference on Communication and Signal Processing (ICCSP)* (2019), pp. 0592–0595.

[30] Wang, Y. A new concept using lstm neural networks for dynamic system identification. In *2017 American Control Conference (ACC)* (2017),

pp. 5324–5329.

[31] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., and Grundmann, M. Mediapipe hands: On-device real-time hand tracking.