GALLAUDET UNIVERSITY

Sign-Language Detection: Identifying Potential Improvements

By: Allison Tate and Shane Angel

Director: Dr. Raja Kushingler

Co-Director: Dr. Christian Vogler

Mentor: **Linda Kozma-Spyetek**

Co-Mentor: Norman Williams

Date: 7/30/2021

Abstract

Due to the ongoing COVID-19 pandemic worldwide, people were stuck home, which meant working remotely using videoconferencing applications such as Zoom. DHH community found themselves struggling with the spotlight feature on Zoom since it only detects vocal sounds. Sign detection technology is being developed for this particular reason. However, the field of Sign Detection has been around since the early 2000s and has many potential applications across the internet. A group of researchers at Texas A&M's Department of Computer Science and Engineering have made significant contributions to the field of Sign Language detection since 2012. One of the newest Sign Detection applications was developed by Google Developers in 2020 called Real-Time Sign Language Detection using Human Pose Estimation. This study aims to determine what algorithms or UI can be improved or changed to minimize errors and the impact of errors in current Sign Language Detection applications. The application testing was conducted with 18 participants. Our analysis showed that body pose estimation software is responsible for low FPS because it has a high demand for processing, but the original software for determining if the participant is signing or not did not. We concluded that the user interface and algorithms need improvements that can be implemented in future research.

Introduction

In the midst of a COVID-19 pandemic, many people had to resort to working and learning remotely. The pandemic led to the increased use of technologies like Zoom, which is helpful for large group meetings without additional costs, and educational institutions and companies have been using Zoom during the pandemic. (Iqbal, 2021) However, Zoom is not deaf-friendly since its spotlight feature only detects voices and doesn't have the technology to detect signers, which leads to professors having to spotlight every signer, a time-consuming task. The solution to this problem is sign detection, which automatically spotlights any person who is signing, lessening the burden on the host, who otherwise has to manually spotlight every DHH person when it is their turn to speak using American Sign Language (ASL). Apple already had sign detection technology used in FaceTime to spotlight any signers automatically, but they did not share their coding or technology with anyone else, so that leads to another current sign detection technology which Google invented. The Google app aimed to detect sign language while improving the accuracy of detecting sign language from non-sign language. We plan to use the Google app to minimize the amount of errors and impact of errors by identifying which algorithms and UI can be improved or changed.

Background

The Texas A&M's Department of Computer Science and Engineering contributions significantly contributed to the field of Sign Language detection. They published a multitude of algorithms, testing results, and approaches that benefited the development of current detection approaches. In 2012, the Texas A&M group contributed with the developments of their five-factor classification criteria, as discussed previously. (Monteiro et al., 2012) After 2012, in 2014, they published their detection algorithms using PMP with an average F1 score of .87, which significantly improved the previous 5FC method by a .18 F1 score. (Karappa et al., 2014) Finally, with a detection approach with an F1 score of .87, the group turned their focus towards Metadata and published their results of testing in 2015. (Shipman et al., 2015) Another publishing that benefited the field of sign language detection was their recent publications in 2017 and 2019. Within both of these publications, they deeply analyzed the tradeoffs between computation times and accuracy. (Shipman, et al., 2017) (Monteiro et al., 2019) Their contributions to analyzing speed and accuracy were essential to the field of sign language detection because by examining the needed computation time, it is possible to deploy a live application that detects sign language in real-time. The difference between Google and Texas A&M is that Google used a live application to detect sign language in real-time while Texas A&M only tested sign detection with recorded videos.

A group of Google developers created a real-time sign detection application called Real-Time Sign Language Detection using Human Pose Estimation. (Moryossef et al., 2020) The goal was to create a feature that accurately monitors the user's movement by using human pose estimation. They identified four monitoring methods and tested which method was the most successful at detecting sign language. The four monitoring methods were BBOX, Pose-Hands, Pose-Body, and Pose-All. (Moryossef et al., 2020) They used techniques previously developed by Texas A&M University to detect sign language within pixels. In contrast, the background movement and non-sign language movement were filtered out by separating the foreground and background of the video input. In the end, they decided that Pose-Body was the best method for detecting sign language in their working demo.

TensorFlow.js (TFJS) is a high-level Application Programming Interface (API) that implements both machine learning and deep learning to create an end-to-end platform that makes it possible for ASL detection applications to be built and deployed. ("Real-Time Human Pose," n.d.) As a result, TensorFlow significantly improved the development of ASL detection applications because it eliminates a large portion of the development process.

A feature of TensorFlow.js is the PoseNet part of the API. PoseNet is a computer vision technology that detects human figures in images and videos using key body joints. ("Real-Time Human Pose," n.d.) Similar to TensorFlow, PoseNet also runs using JavaScript. Due to this, PoseNet strictly runs on the client's computer and does not require processing from the server for computation. Therefore, PoseNet does not share information with the server about the image and gives more privacy to the user. PoseNet can be used to estimate either a single pose or multiple poses, meaning there is a version of the algorithm that can detect only one person in an image and another version that can detect multiple individuals in an image.

Methods

The two key goals for our testing were to determine results for accuracy and latency/lag. We compared our results to the Google group's results for accuracy and then compared results for latency to the other table and speech detection results. In addition, we will be using the Google sign language detection application to determine the impact of errors and delays. Our research question is what algorithms or UI can be improved or changed to minimize errors and the impact of errors in current Sign Language Detection applications? To complete the testing, we set up a VM on Google's Cloud platform and the code set up by the Google Detection team.

We will introduce our study to participants and explain the consent form where they can sign to allow data gathering and have their faces shown on the video. However, none of the identifying features about them will be publicly shared. Next, we explain the demo and the overview of intentions for it. We will explain that there are two parts of the demo: the signing part and the non-signing/gestures part. Zoom will be the way we can communicate with participants due to the ongoing pandemic and remote work. Participants will receive the link to the demo before sharing their screen with us. The reason for this is that the demo application does not allow the camera to be on simultaneously as Zoom, so we will have the participant turn off their camera on Zoom but leave the camera on for demo while sharing the screen with us. Participants will have a chance to play with a demo for forty-five seconds to two minutes so they can get familiar with the application before we can proceed with the experiment. They will be given a list of phrases to sign after viewing the sample video of phrases being signed, and we will observe the results to determine if there are any false results or errors. If there are errors or false negatives/positives, we will document the phrase they occurred on and how often. Then, we will repeat the process but with gestures to determine if the demo is able to differentiate between signing and gestures.

Once completed, we will give the participant the link to Google Forms where they either agree or disagree with the statement provided that is related to the study. They will then answer a few open-ended questions related to the experiment as well. Finally, the last section of the survey is demographics such as age, gender, race, hearing identity, signing ability, and what kind of camera they used for the study. For example, race is necessary to be able to identify if the user's skin color impacts accuracy. Likewise, knowing the signer's dominant hand is vital to our study. It will enable us to identify if being right-handed dominant outputs better results than being left-handed dominant and vice-versa. Once the form is submitted, we will ask the participant if they got any questions or feedback about the demo or project. Lastly, we may ask the participant about their interview experience and how we may improve our process.

Testing time lasted for a maximum of 30 minutes, and participants were compensated a $15 Amazon gift card. In total, we aimed to test at least twenty participants to gather enough data for the study. We recruited participants using word of mouth, contacting students from our peers, a few from alumni at university through email or social media, and family and friends through word of mouth. Our research question asks: What algorithms or UI can be improved or changed to minimize errors and the impact of errors in current Sign Language Detection applications? We hope to find the answer to that very question through studies with participants on the application.

Results

The majority of our participants are in the age range of 18-24, two are in the 25-39 group, and two are in the 40-60 group. Nearly half of them identified as female, with the remaining identified as male. The race was the necessary factor in the experiment to see if it can affect the software. More than three-fourths of participants identified as white, nearly one-fourth were Asian, 6% were American Indian or Alaska Native, and 5.6% were Hispanic or Latino or Spanish origin. As for the hearing status, nearly half of them were hearing, one-third were hard of hearing, while almost one-fourth were deaf. We found that 61.1% are fluent in ASL, while 11.2% considered themselves beginners in ASL. 27.8% said they are comfortable using ASL but not fluent. Participants reported having from low-end to high-end computers, and the majority of them own built-in laptop cameras. Others reported having a webcam attached to the monitor, actual camera, or all-in-one.

Everyone agreed that the demo was pretty simple to set up. From our survey, we found that 72% of participants agreed in some way that the demo was able to differentiate between signs and gestures. More than three-fourths of participants reported that the demo's output or yellow bar graph works as intended. However, there were conflicting reports on whether the gestures did not cause the yellow bar graph to increase, and only one-third were not sure, with nearly 50% that the gestures did not cause the graph to increase. Participants were also conflicted on whether the phrase they signed had a similar output to other phrases, with little more than half agreeing but less than one-fourth being unsure. The majority of participants agreed that the gestures they performed would occur naturally while using videoconferencing applications such as FaceTime, Zoom, or Google Hangouts. They also agreed that the yellow bar graph was easy to understand, and they did not notice any lag while using the demo. There were conflicting agreements on whether FPS or the blue bar graph did not change during the demo, with only 50% agreeing and nearly 40% being unsure. We found that not everyone agreed that the demo was ready to be added to Zoom and Google Hangouts applications as some participants felt it needed more work or were unsure. Out of 18 participants, only two were not right-handed, with one of the remaining two being ambidextrous.

We encountered some issues with the demo. One participant reported that they had to switch a browser due to constant lag on Google Chrome. The other participant said there was no button to minimize the screen, so they had to use their keyboard to minimize the screen. It was noted that the app does not work on Safari, so Google Chrome was the preferred browser. They all reported little to no delay during the testing, and the delay was 1-2 seconds at the top. They also all agreed that the demo was easy and exciting to use, with few participants reporting the setup was confusing, or the demo thought they were signing when they were actually sitting far away from the camera. The participants offered various suggestions for the application's improvement: better layout, one prompt at a time, tweaking the gestures, and stressing the importance of bar graphs. After conducting the studies, we moved onto analyzing the data and came up with several conclusions.

Data Analysis

After the conclusion of our app testing, we had our participants complete a Google Form that asked them questions about their perception of our sign detection application, followed by a demographics section to gain general information about the participants that could pertain to our testing of the sign detection app.

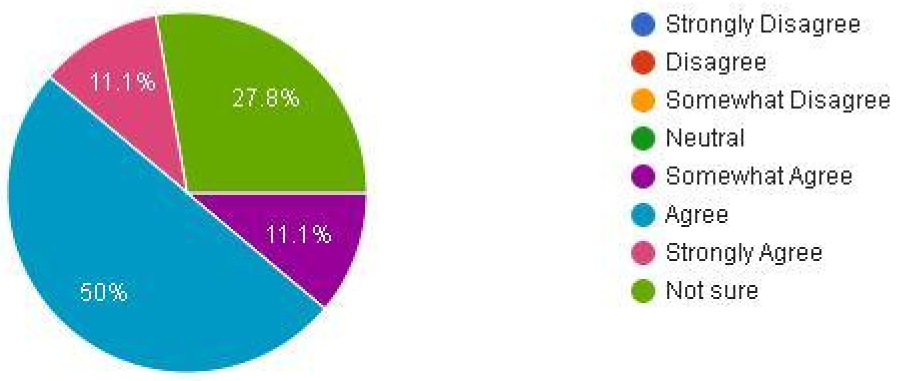

Figure XI

From these responses, we learned that nearly three-fourths of our participants either strongly agreed, agreed, or somewhat agreed that our app demo was able to tell the difference between sign language and gestures. The other one-fourth of our participants responded that they were not sure if the app was able to differentiate between the two. This highlights one issue with our app, being that the app's user interface (UI) struggles to communicate with the user whether or not the app is detecting sign language or not. Currently, the app's two outputs, sign detection value, and FPS are displayed by the two small graphs in the top left of the app. The actual output method is fine, but the size of the graphs is too small to see if you are signing and focusing on things besides solely the app's output. Therefore, the authors propose that the output of the app be made more clear for the users. We propose that either the size of the app be increased by at least 150% or by adding a log function to the app. These changes to the app are important for the usability of this app because users tend to need something that is working properly before they are willing to add it to the normal applications they use by choice.

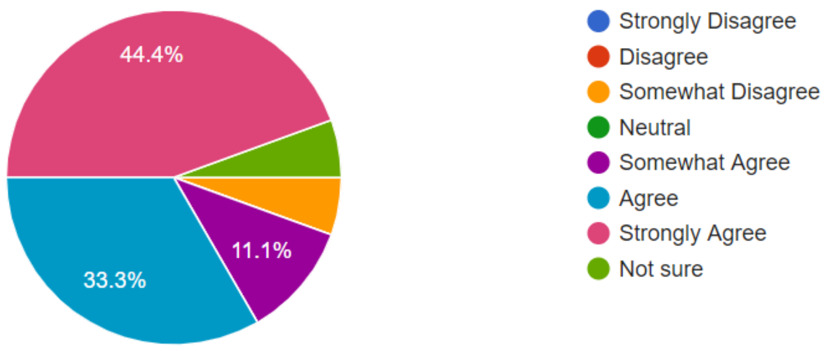

Figure X2

The next question in our Google Form that had notable results was our question asking our participants if they agreed that the gestures we had them complete would occur naturally when using a video conferencing platform. As indicated in Figure X2, 44.4% strongly agree, 33.3% agree, 11.1% somewhat agree, 5.6% neutral, and 5.6% somewhat disagree. Therefore, almost 90% of the participants either strongly agreed, agreed, or somewhat agreed that these gestures would occur naturally in a video conference platform. Based on the results, we conclude that the gestures we had our participants complete were proper and confirmed our testing procedures.

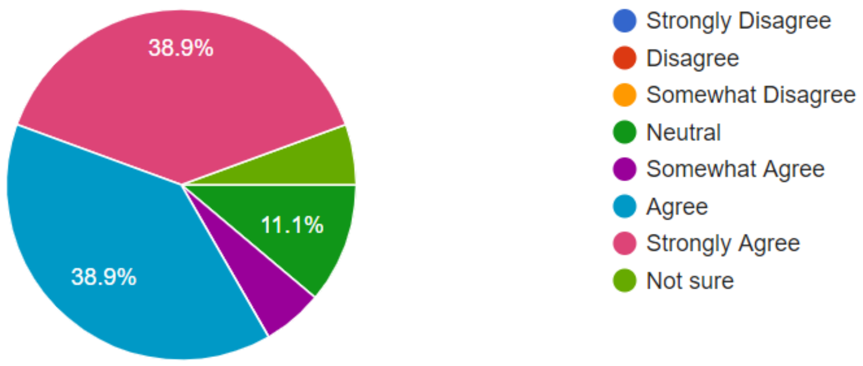

Figure X3

Another question that produced significant results was the question asking if the participants agreed or disagreed that they did not experience any lag or latency while using the app demo. Overall, an overwhelming majority of the participants stated that they either strongly agreed, agreed, or somewhat agreed that they did not notice any lag. To be exact, 38.9% strongly agreed, 38.9% agreed, and 5.6% somewhat agreed, as shown in Figure X3. With these results, it can be concluded that 83.4% of our participants did not have a noticeable issue with lag or latency. Lag or latency is not necessarily an issue with our app because instead of sending packets back and forth between the client's computer and a server while using the app, a single packet containing the code for our app is sent to the client's computer, and no information is sent back to the server. The way the app is set up, all computation occurs on the client's computer. Therefore, the client's computer can alter how well the app performs depending on the type of processing their computer possesses. For example, while doing testing, we noticed that the blue graph that displays the amount of frames per second that is used for detecting sign language would vary between each participant. Therefore, we used the FPS per participant and the peak detection for each sign phrase/gesture to examine the impact of FPS on the amount of signs detected by the app.

Before conducting our analysis of the data we collected during the study, we did a separate analysis of the data to determine if our data contained any outliers that should be removed before proceeding with the analysis. We found a total of 4 different sets of data that were excluded. These sets of data were removed for a variety of reasons. For example, some of the data was inconsistent due to poor camera quality and unexplained poor results. Another reason was that the participant had limited proficiency in ASL, so that the participant would take long pauses between each sign, and it had a negative impact on the results. The common trend between all four sets of data was that they were inconsistent. Once we finished removing outliers, we began deeply analyzing the data we documented from the recordings of the participants using the sign detection app.

Due to FPS being different for each participant, we decided to find the averages for each group of FPS ranges. FPS was separated into the following four groups: 1-10 FPS, 11-20 FPS, 21-27, and 28+. The graph demonstrating our average for each group for each sign phrase/gesture is shown below:

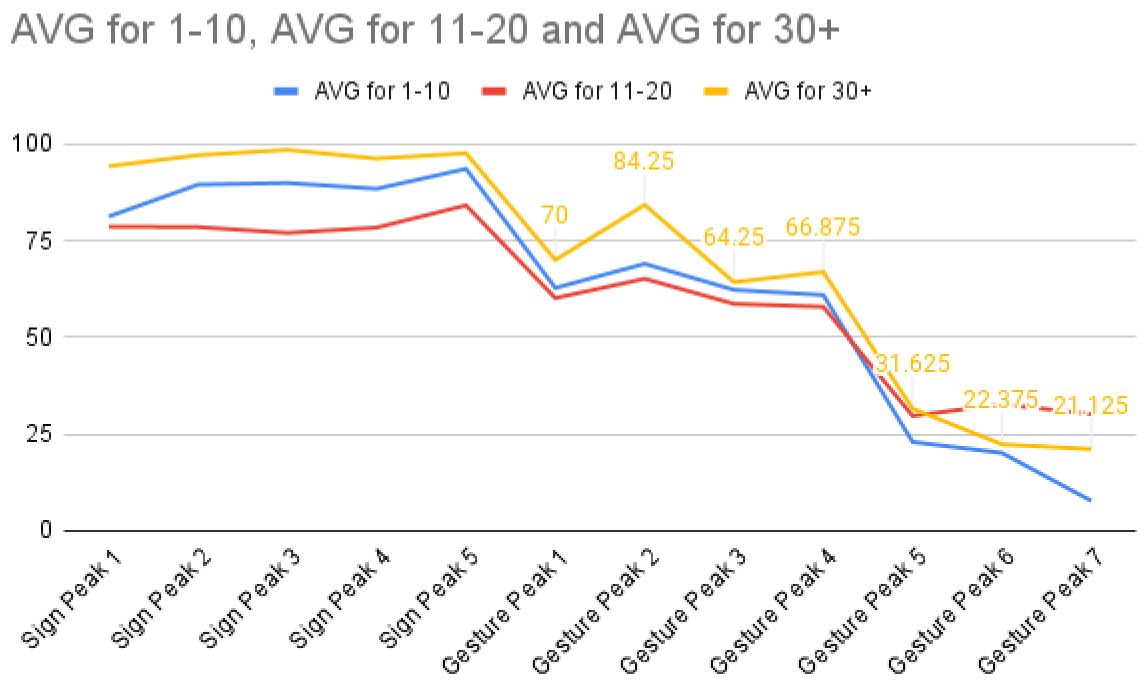

Figure X4

From Figure X4 above, it is evident that FPS can make a difference in the output for the users that use it. For example, desired FPS or 28+ FPS will typically result in higher sign detection compared to lower FPS. The main cause for higher FPS outputting a higher peak detection value is that the app detects signing by subtracting movement between frames, so by having more frames processed per second, it will cause a higher output. Likewise, as the FPS decreases, the peak sign detection will be lower compared to higher FPS computations.

For FPS impact, we did not include the data for the class of data of FPS 20-28 FPS because there was only one participant that fits that data, and we noticed their data was a bit inconsistent for testing. So we didn't want only that single line of data to make an impact. For this comparison, we found the averages of all FPS groups and found that the averages for high FPS tests outputted the highest average for the sign phrases. Also, for gestures 5, 6, and 7, the output was lower than 11-20 FPS but not lower than 1-10 FPS. One noticeable part of the graph above, 28+ FPS for gesture 2, was a vast majority higher than the others and peaked at around 84.25 detection. Gesture 2 requires the participant to scratch their arm while in the camera's view. One explanation for this peak in 28+ FPS is that scratching your arm is a relatively quick movement, so the higher FPS allows for more detection per frame. Besides the impact of FPS, our study focused on how the distance away from the camera can impact the output of sign detection by the app.

All participants completed half the testing at a shorter distance from the camera, where the camera could see about from the mid-torso to the top of the head. Once they finished this, they did the same set of testing with the app but at a farther distance away or showing from about their waist to about a foot above their head. The reason for this type of setup is to have data from a shorter distance and a longer distance to allow for comparison between the two. The resulting graph of the average between the two is as followed:

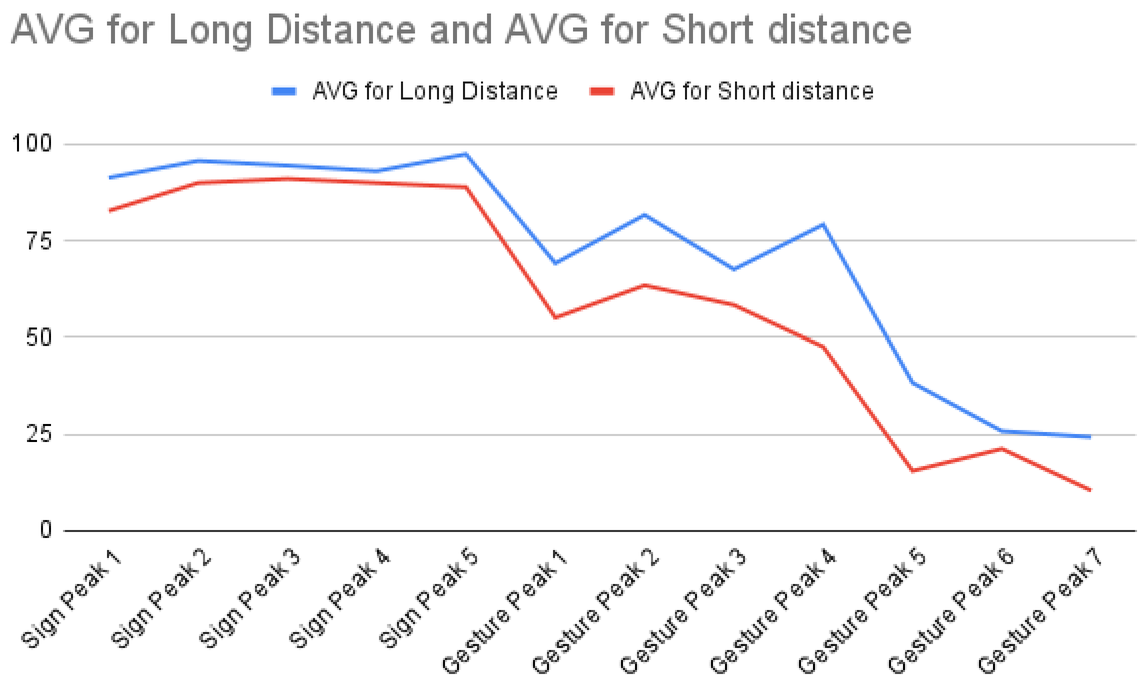

Figure X5

From Figure X5, we conclude that a longer distance from the camera, on average, results in a higher overall detection value. The results show an increase because the camera can see more of the user per frame. Therefore, the subtraction processing by the app results in a higher value because the camera is able to see more of the user. After completing both sets of comparisons, a pattern has been observed. The pattern observed is that the sign detection app's output will depend on the amount of movement observed per second of the computations. To prove this, we completed a comparison of the data by grouping the two types of distances with the amount of FPS that was used during the computation by the participant's computer.

Our last comparison is comparing distance impacts during both ends of the spectrum of FPS. So we compared the 1-10 group of FPS and the 28+ group of FPS at a short distance and long-distance. We were able to deduce that a short distance with a low FPS (1-10) resulted in the lowest and poorest amount of sign detection. At a longer distance, we found that low FPS results were better and similar to the other groups of data. The relationships between both FPS and distance are demonstrated as shown in Figure X6 below:

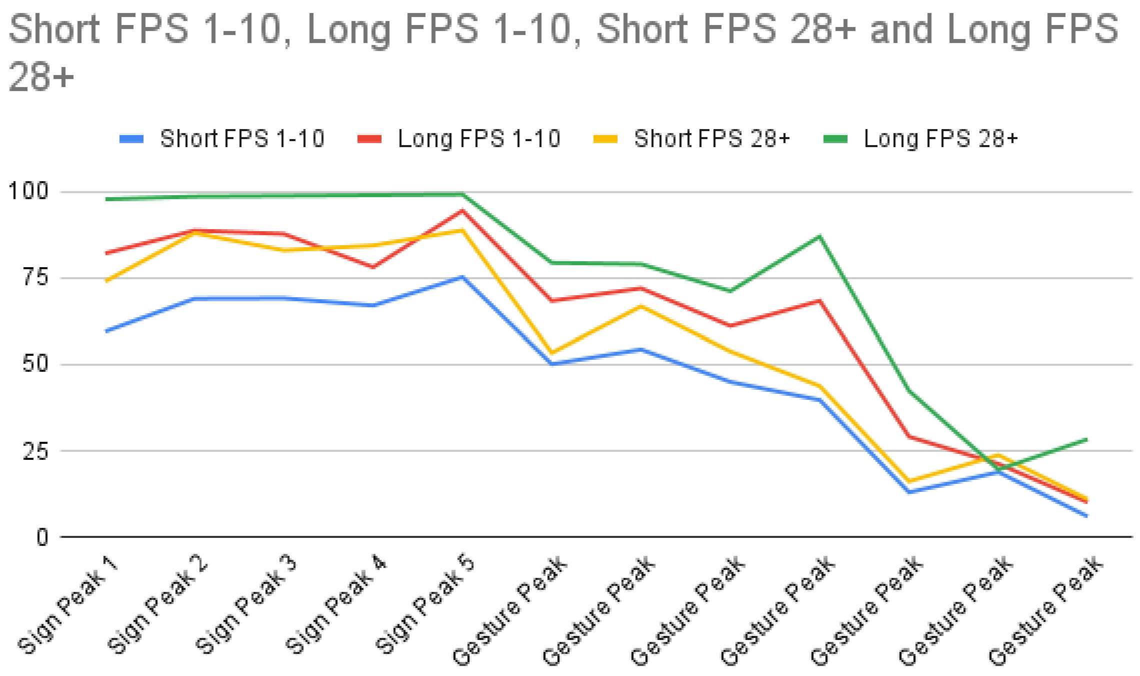

Figure X6

One thing to note is that a far distance and high fps gave the highest amount of sign detection. Similar to the comparison of only distance, again, gesture 4 gave a peak for the long distance. This gesture gave a peak for long distances because this gesture requires the participant to stretch their arms out. The software works by looking frame by frame of the video input and subtracting the movement between each frame. For the short distance, most often, the participant's hands would end out of the frame and therefore would not be detected and result in nondetection. However, with a long distance, the hands would be in the frame and therefore would result in more detection.

Discussion

After the completion of the analysis, we concluded that an FPS range of 28-30 is desirable. Due to being online and being unable to control what computers our participants used, the performance of our app depended on the participant's computer. Therefore, the computer specs varied from participant to participant. However, we were able to conclude that the minimum CPU requirement to use the detection app at an FPS of 28+ is to have at least a quad-core processor with a minimum base clock of 2.0 GHz. However, it is important to note that all of the participants completed their testing with the application while using Zoom in the background. The use of Zoom added extra processing on their CPU and took away resources from the sign detection app's processing. Although Zoom can impact the minimum CPU specs needed for desired FPS, it is important to note that one of the intended uses of our app is video conferencing, so our conclusion is that a quad-core CPU with a minimum base clock of 2.0 GHz would be justified.

Not only does CPU processing speed impact our app's results, but also the quality of camera video and the lighting can make an impact. For example, cameras capture video at different FPS or at various resolutions. If a camera captures at an FPS below 30 frames per second, then the app will only be able to analyze the number of frames provided. This can make an impact in future sign detection applications because it will cause any improvements to account for the amount of frames captured by the camera. Therefore, a possible future improvement for our app would be an interface addition that communicates the camera's FPS to the users. Additionally, higher-quality cameras capture video at a higher resolution. This factors into the effectiveness of the app because with higher resolution, the app is able to more effectively complete body pose estimation and, in theory, would generate more accurate results. Determining the impact of the camera capturing FPS and resolution is a possible factor that should be studied more in the future.

Future Work

From completing our study, we have identified four areas of our app that need to be improved in order to potentially decrease the amount of errors that occur and to improve usability for users. The first area of improvement is to change the way the app outputs its results and add a log system to track previous results. Next, the app should be changed to allow for optimization throughout the processing on the client's side of the system. This change will directly benefit users because it will enable the app to process more frames per second and result in fewer errors in detection. Besides optimization, another change in the algorithms would be to recreate the machine language part of the app but with only specific sign languages and not an array of them. Through our research of the app, we found that the app was taught using multiple different languages. A potential impact of this would be that the app is not able to differentiate between signing and gestures because of the different languages present. Lastly, we noticed that different lighting and background would seem to cause unwanted errors during testing. Therefore, we propose the addition of AI software to minimize the impact that different lighting and background have on the application.

Limitations

Reviewing our testing process and results, we have identified a few things that could improve our process in the future. Looking back, our data needed a more vigorous verification process. To do this, we would have needed to complete more interviews with participants to ensure we had more data for each group of data that we had. For example, during analysis, we had to eliminate the FPS 21-27 because we only had one group of data for this range. Therefore, we were unable to complete additional analysis because we couldn't use this group. Another change we would propose in the future would be having all interviews be conducted on computers provided by the study. By providing specific computers to participants, we would control which FPS different participants would conduct the demo testing with. Although these changes would benefit our study in the future, they do not change the takeaways of our study.

Conclusion

Our study's goal was to complete a successful study to identify possible improvements to current sign detection software. In order to accomplish our goal, our team completed a total of 18 study interviews and collected data for each. After completing our data analysis and combining our research, we were able to determine four areas of improvement for current. The proposed changes included changes in our selected app's UI and processing. Our proposed UI change would make our app's output more clear to the user. Our proposed processing changes include optimizing the body pose estimation software, training the app's machine learning process only to use American Sign Language, and adding a type of AI for better performance for different lightings and backgrounds. Therefore, our team strongly feels that we have properly completed our goal of identifying possible improvements to current sign language detection software and answered our research question.

Citations

Iqbal, M. (2021, March 10). Zoom revenue and usage Statistics (2020). https://www.businessofapps.com/data/zoom-statistics/.

Karappa, V., Monteiro, C. D. D., Shipman, F. M., & Gutierrez-Osuna, R. (2014). Detection of sign-language content in the video through polar motion profiles. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1290–1294. https://doi.org/10.1109/ICASSP.2014.6853805

Monteiro, C. D. D., Gutierrez-Osuna, R., & Shipman, F. M. (2012). Design and evaluation of classifier for identifying sign language videos in video sharing sites. Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '12, 191. https://doi.org/10.1145/2384916.2384950

Monteiro, C. D. D., Mathew, C. M., Gutierrez-Osuna, R., & Shipman, F. (2016). Detecting and Identifying Sign Languages through Visual Features. 2016 IEEE International Symposium on Multimedia (ISM), 287–290. https://doi.org/10.1109/ISM.2016.0063

Monteiro, C. D. D., Shipman, F. M., Duggina, S., & Gutierrez-Osuna, R. (2019). Tradeoffs in the Efficient Detection of Sign Language Content in Video Sharing Sites. ACM Transactions on Accessible Computing, 12(2), 1–16. https://doi.org/10.1145/3325863

Moryossef, A., Tsochantaridis, I., Aharoni, R. Y., Ebling, S., & Narayanan, S. (2020). Real-Time Sign Language Detection using Human Pose Estimation.
Real-time Human Pose Estimation in the Browser with TensorFlow.js. (n.d.). Retrieved May 27, 2021, from https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html

Shipman, F., Gutierrez-Osuna, R., Shipman, T., Monteiro, C., & Karappa, V. (2015). Towards a Distributed Digital Library for Sign Language Content. Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries, 187–190. https://doi.org/10.1145/2756406.2756945

Shipman, F. M., Duggina, S., Monteiro, C. D. D., & Gutierrez-Osuna, R. (2017). Speed-Accuracy Tradeoffs for Detecting Sign Language Content in Video Sharing Sites. Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility, 185–189. https://doi.org/10.1145/3132525.3132559