Final project report.

Gesture Recognition. Sign Language.

Dias Zhagaparov IT-2207
Insaniyat Tungushpayeva IT-2207
Nadir Kuat IT-2207

Introduction

Gesture recognition for sign language is a critical area within computer vision and machine learning, aiming to bridge communication gaps between deaf and hearing individuals. Sign languages, characterized by their completeness and expressiveness, serve as vital means of communication for the deaf community. However, challenges persist in environments where others may lack proficiency in sign language, hindering effective interaction.

In response to these challenges, our project focuses on the development of an American Sign Language (ASL) recognition frame app. It utilizes advanced algorithms and machine learning techniques to accurately interpret and translate ASL gestures into text or letters, facilitating seamless communication between deaf individuals and those who may not be proficient in sign language.

By leveraging the power of computer vision, our ASL recognition frame app aims to empower deaf individuals by providing them with a tool to effectively communicate with the broader community. Additionally, it serves as a valuable resource for hearing individuals seeking to learn and understand sign language, promoting inclusivity and accessibility in communication.

Through this project, we aim to contribute to the advancement of assistive technologies and foster greater understanding and integration between deaf and hearing communities.

Literature Review

Pigou, L., Dieleman, S., Kindermans, P., & Schrauwen, B. (2015). Sign language recognition using convolutional neural networks. In *Lecture Notes in Computer Science* (pp. 572–578). https://doi.org/10.1007/978-3-319-16178-5_40

https://www.ijeiat.com/images/sliders/92dc4915b9487f589cfe29a2d410cc0a.pdf

Prakash, K. B. (2020). Accurate Hand Gesture Recognition using CNN and RNN Approaches. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(3), 3216–3222. https://doi.org/10.30534/ijatcse/2020/114932020

Current Work

The idea of our project is to create a model that recognizes sign language, in particular American Sign Language (ASL), using video. The code we wrote in Python collectdata.py , allows us to capture and save images of hand gestures or signs by pressing certain keys corresponding to different letters of the alphabet. So we created our dataset using our hands and various backgrounds to better recognize the model. Then we trained our model in Google Colab using Convolutional Neural Network (CNN) for image recognition. After that, we imported the finished model into our file realtimedetection.py, where it runs the frame in the beginning, in which our created model works.

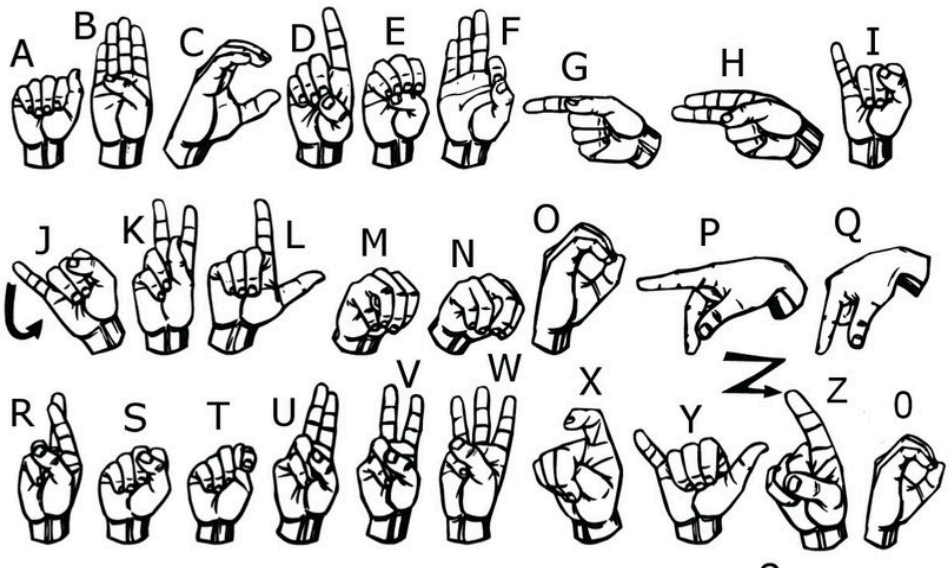
In our project, for machine learning, we used libraries such as Tensorflow and Keras, as well as OpenCV for image analysis, classification and processing.

To run the program, you need to do the following:

1) Download all libraries from requierments.txt :
   - opencv (pip install opencv-contrib-python)
   - tensorflow (pip install tensorflow)
   - keras (pip install keras)

2) Run realtimedetection.py through the debugger

3) In the window that opens, namely in the frame, indicate the letters ASL with hand gestures, here is an alphabet we used as a referense:



Data and Methods

We created the data for the dataset personally, by photographing our hands using a program that we wrote in Python. There is a file in our program collectdata.py when you start it, 27 folders are automatically created, including all the letters of the alphabet and the background. Then, after the launch, two frames "data" and "ROI" are released. Using OpenCV (cv2) to capture images, we created our dataset by showing a certain gesture to a certain letter in the ROI frame, and clicking on the same letter on the keyboard, it saves the image to the appropriate folder. The three of us made the dataset, using different backgrounds and different hands to better recognize the model. In our SignImage48x48 folder, we have distributed all the photos to the training set and validation set by 80% and 20%, respectively.

In our program, we used a convolutional neural network (CNN) model designed for image classification tasks. At the input level, we have defined the shape of the input data that expects the input of 48x48 pixel grayscale image packets.

The model starts with a series of convolutional layers (Conv2D). These layers are responsible for studying the objects in the input images.

Each convolutional layer applies a set of trainable filters to the input image to create a set of feature maps. These filters are 3x3 in size and use the ReLU activation function.

After each convolutional layer, there is a max-pooling layer (MaxPooling2D). Max-pooling reduces the spatial dimensions of the feature maps, helping to decrease computational complexity and prevent overfitting.

Dropout layers (Dropout) are added after each max-pooling layer to randomly drop a fraction of neurons during training, which helps to prevent overfitting.

Flatten Layer:

After several convolutional and max-pooling layers, the feature maps are flattened into a one-dimensional vector to be fed into a fully connected layer.

Fully Connected Layers:

Following the flatten layer, there are several fully connected (dense) layers (Dense). These layers perform classification based on the learned features.

Each dense layer is followed by a dropout layer to prevent overfitting.

The number of neurons in the last dense layer matches the number of classes in the classification task, which is 27 in this case (26 letters of the alphabet plus one for blank).

Output Layer:

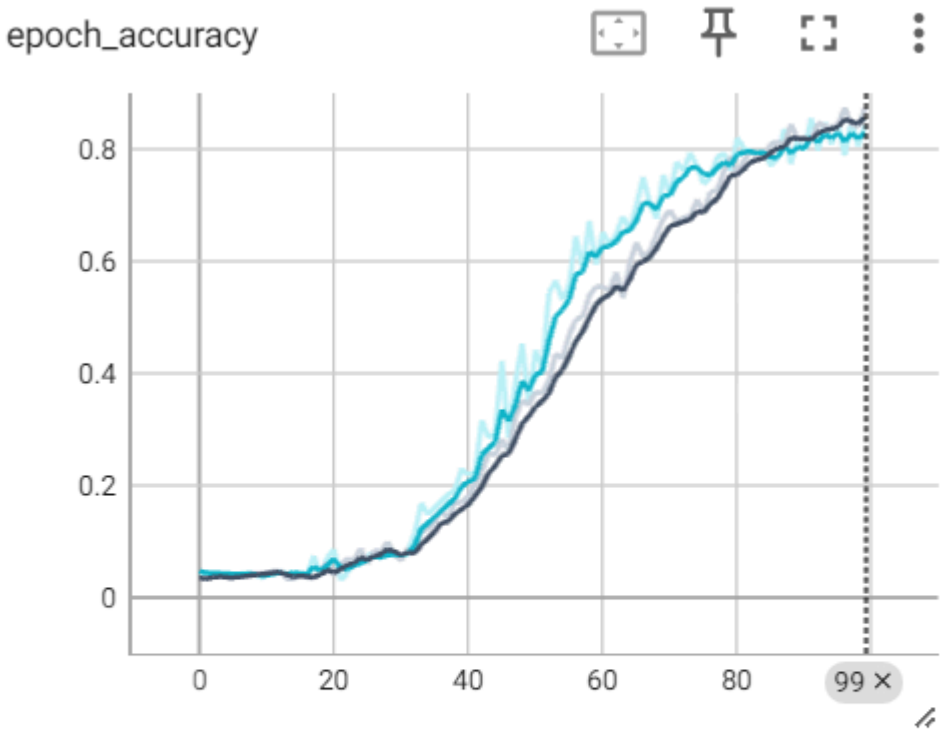
The output layer uses the softmax activation function to output probabilities for each class. It predicts the probability distribution of each class given the input.

Compilation:

The model is compiled using categorical cross-entropy as the loss function, which is suitable for multi-class classification problems.The Adam optimizer is used for optimization. The performance metric used for evaluation is accuracy.

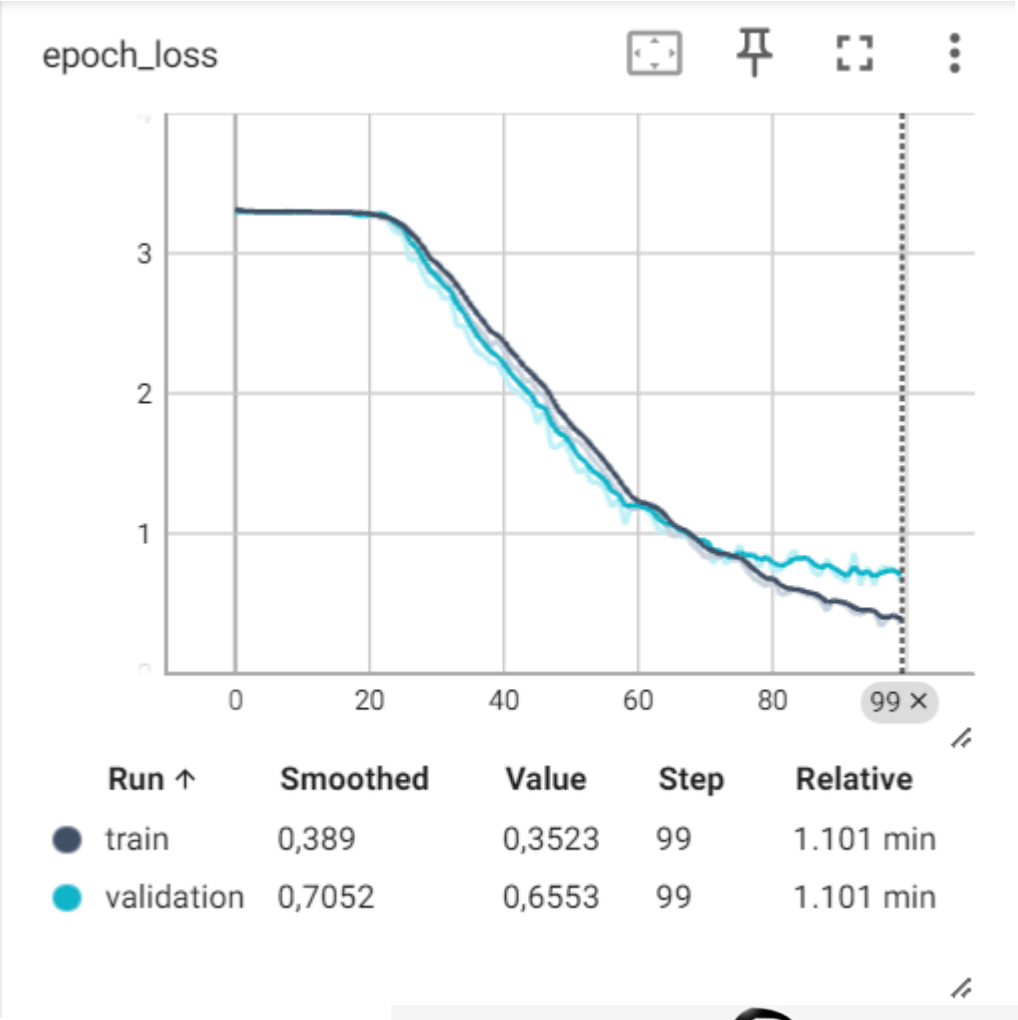
Results

Epoch accuracy time series:



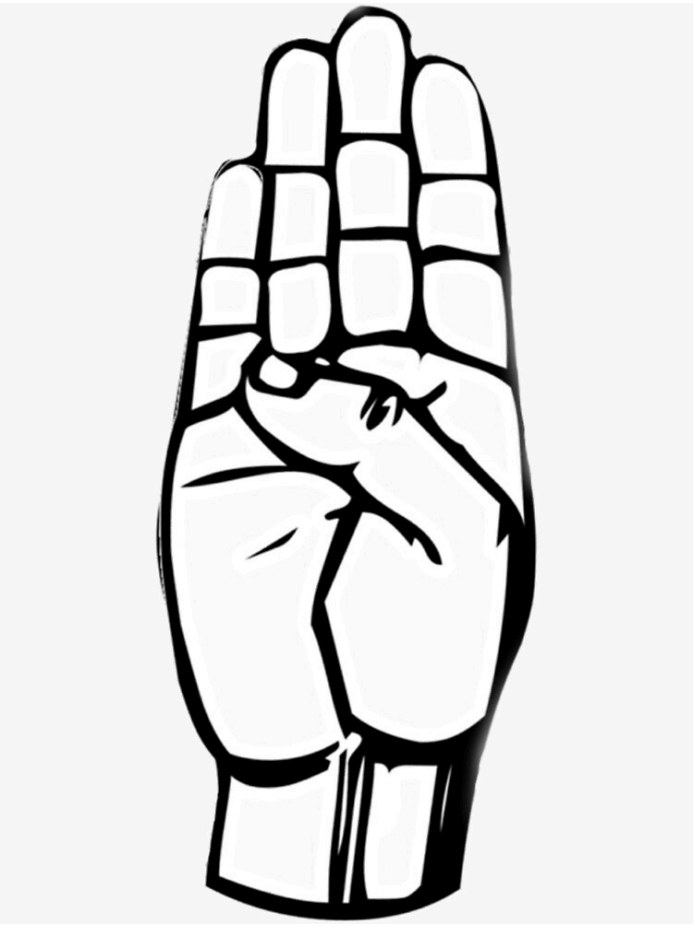
Run ↑	Smoothed	Value	Step	Relative
train	0,8566	0,873	99	1.101 min
validation	0,8289	0,8438	99	1.101 min

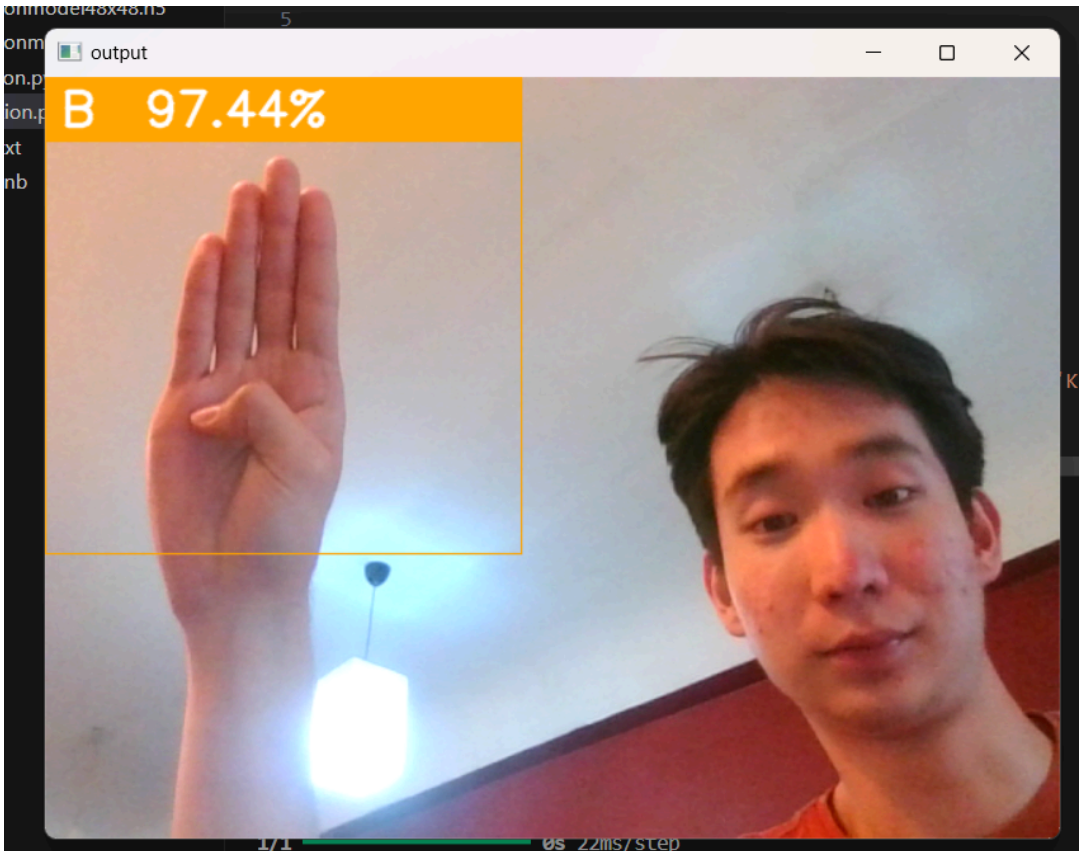
Epoch loss time series:



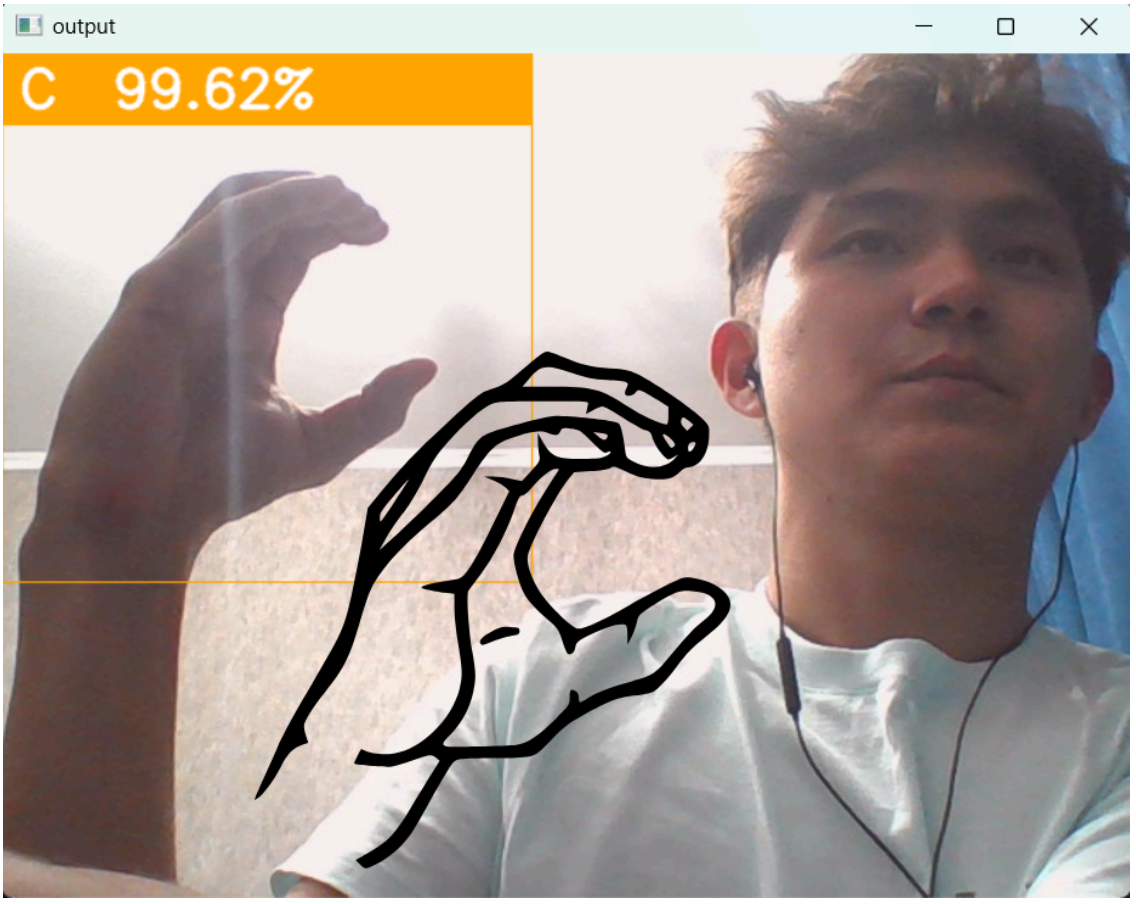
And how it works:

For example, B letter in ASL

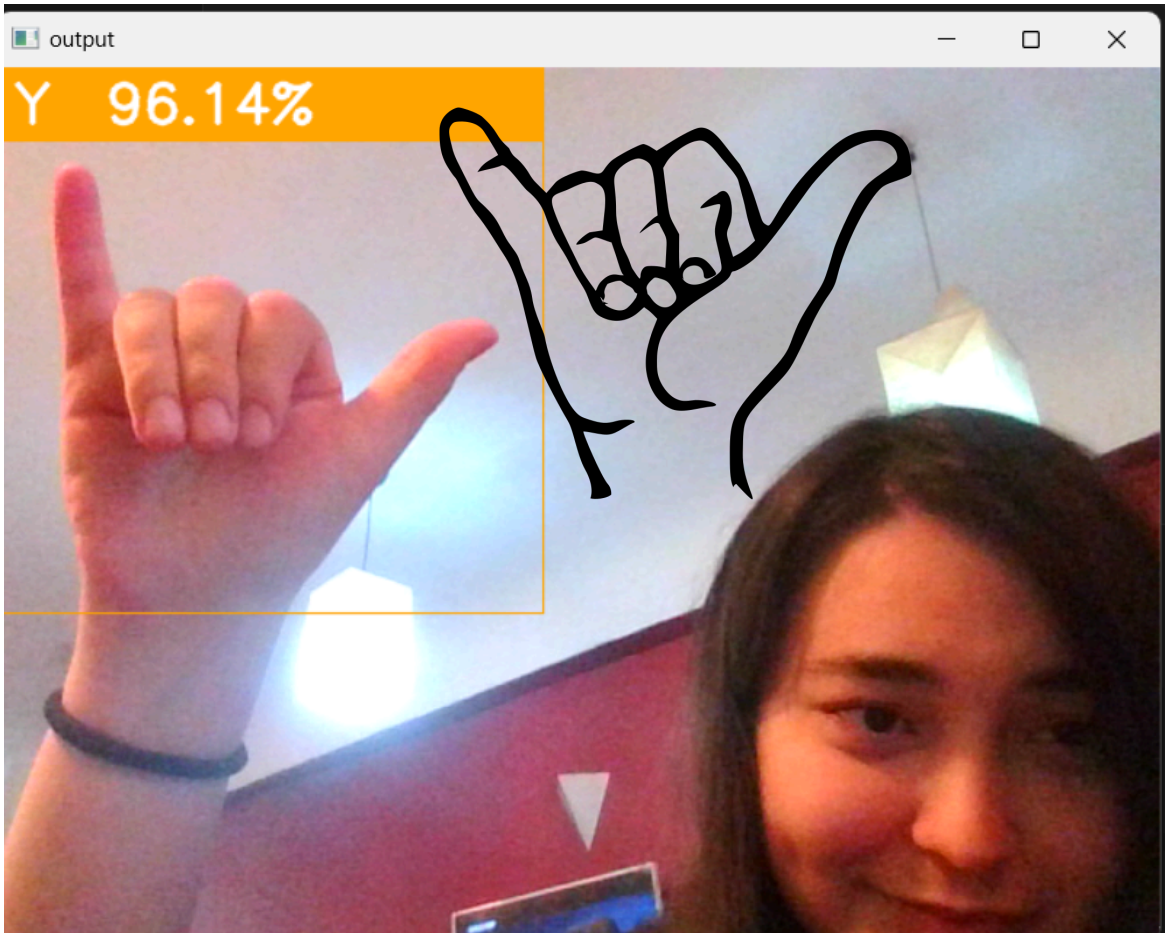




Or C letter



or Y letter:



Critical review of results for our final project

We are enthusiastic about integrating the complete Kazakh alphabet into our repertoire of linguistic gestures. Our application possesses the remarkable capability to dynamically incorporate additional datasets, ensuring that each letter's distinct gesture is accurately captured and represented. Through the power of Python's machine learning algorithms, we will meticulously analyze and interpret gesture data, allowing users to seamlessly communicate using the rich linguistic diversity of the Kazakh alphabet. This integration signifies a significant step forward in bridging language barriers and fostering inclusive communication within our application.

We exhibit the results of our efforts and devotion in the field of Gesture Recognition and Sign Language utilizing machine learning. Our path has been marked

by significant progress, but **it is critical to acknowledge the challenges we have faced along the way.**

One of the most significant issues we encountered was the dependence of our outcomes on lighting conditions. As we explored deeper into the complexities of our system, we discovered that under low lighting, our results degraded significantly, resulting in mistakes in gesture detection. This problem is a substantial hurdle to real-world applications, as illumination conditions are frequently unexpected. Furthermore, our work entailed the categorization of 27 separate classes, including a backdrop class critical for correct identification. However, we quickly noticed that the quality and amount of our background information had a significant impact on our system's performance. Acquiring and curating such a dataset proved to be a time-consuming effort, emphasizing the difficulty of our project.

Furthermore, we had difficulty differentiating between specific gestures, such as "s" and "t", "u" and "v", "i" and "j", "m" and "n". These subtle distinctions constituted a daunting obstacle that could only be addressed by extensive model training and refining.

Despite the severe hurdles we faced, our voyage was not in vain. We have gained crucial insights that will pave the way for future developments in Gesture Recognition and Sign Language. As we reflect on our journey, we must acknowledge the influence of these disadvantages on our outcomes.

However, difficulty fosters ingenuity, and we do not lack solutions. Moving forward, we must investigate strategies to reduce the influence of lighting circumstances on our system, maybe by augmenting our dataset with photographs recorded under other lighting settings. Furthermore, we must invest in strengthening our model to better distinguish between difficult motions, utilizing advanced machine learning algorithms and approaches.

To summarize, while our road has been difficult, it has been made possible by patience and drive in the field of Gesture Recognition and Sign Language. As we chart our road forward, let us see these problems as possibilities for development and innovation, paving the way for a world where communication knows no boundaries.

Next steps

Since we have not found the Kazakh alphabet in sign language, we plan to add the Kazakh language in the future. To do this, we need to create a dataset, and we have the capabilities for this, but it is quite time-consuming work.
In the future, we would like to create this, as it is relevant and develops both Kazakh sign language and AI.

It is critical that we set a roadmap for our future steps, employing our knowledge and goals to push the frontiers of Gesture Recognition and Sign Language even further.

One interesting potential on the horizon is the inclusion of the Kazakh alphabet into our sign language detection system. Recognizing the significance of inclusiveness and diversity, we acknowledge the lack of the Kazakh alphabet in current sign language statistics. As a result, we foresee a future in which the Kazakh language finds a place in our system, reducing communication gaps and promoting inclusion within the Kazakh community.

However, this undertaking needs rigorous preparation and execution. It includes creating a thorough dataset that captures the subtleties of the Kazakh sign language. While we have the ability to complete this assignment, we recognize its time-consuming nature. Nonetheless, we remain committed to this cause, acknowledging its importance and potential influence on both the growth of Kazakh sign language and the evolution of artificial intelligence.

Sources

https://www.youtube.com/watch?v=6Bn0PY_ouBY
https://www.youtube.com/watch?v=pDXdlXlaCco
https://www.youtube.com/watch?v=doDUihpj6ro
https://research.nu.edu.kz/en/projects/kazakh-sign-language-automatic-recognition-system-k-slars