AML FINAL PROJECT REPORT

A Deep Learning Approach to Sign Language Detection

1. Abstract:

Deafness and mutism tend to draw attention to those who are unable to hear or communicate with the outside world, aiming to differentiate them from the rest of society. Most of the communication is done in ASL (American Sign Language) or ISL (Indian Sign Language), with various motions representing a certain letter, number, or sentence. Not all deaf people use sign language; some use hearing aids; however, few can afford one. Sign language addresses all these concerns because it is more efficient, easy, and adaptable. The research focuses on developing a system for hand gesture recognition that conveys the message of hope for the entire society and its significance. Because it is more efficient, easy, and adaptable, the research focuses on developing a system for hand gesture recognition that conveys the message of hope for the entire society and its significance. The purpose of this work is to use deep learning to detect sign movements and to improve existing systems to achieve efficient software with proficient and notable software.

2. Introduction:

The most widely recognized and understood method of communication is gestures, which involve employing body language or hand movement in sign language. Sign language is a kind of gesture communication used by the deaf and mute communities that brings light into the darkness by allowing people to express themselves. To appreciate these points of view, however, one must be aware of the signs and gestures used to express them. In today's world, everyone is so focused on themselves that they don't take the time to think about others. In such circumstances, the deaf and mute communities are left to deal with the implications. Several advancements are being achieved in numerous areas (robotics, healthcare, etc.) because of recent developments in artificial intelligence, pushing us to strive for a better society. Human-Computer Interaction, or HCI, is one such area of research. HCI is concerned with application domains and facial recognition technologies. The goal of picture analysis in this system is to help and train the system to decode the indicators utilizing CNN (Convolutional Neural Network) to open the door to larger discoveries.

Communication reflects a person's fundamentals of exchanging ideas, thoughts, and views, which is why it is constantly emphasized that communication is the key to all opinions. However, a sizable fraction of the global population suffers from hearing loss, speech difficulty, or both. This project aims to bridge the communication gap between us and the deaf-mute community by allowing them to experience a hopeful environment. The system's main highlights include converting the acquired data images into grayscale format, correctly processing them, and ensuring that the images collected are accurate to the Sign Language. The main idea revolves around the Deep Learning Neural Network recognition system.

3. Background:

Sign language plays a crucial role in facilitating communication for individuals with hearing impairments. However, existing solutions for sign language detection often face challenges in terms of accuracy and real-time processing. This project addresses these issues by employing deep-learning neural networks to enhance the efficiency and reliability of sign-language detection systems.

4. Motivation:

Our Aim of the project is to improve the communication of deaf and dumb people. In this manner, a system that can make predictions of hand gestures of deaf and dumb people. Here we are making a model for forecasting the outcomes of various hand a model that forecasts the outcomes of various hand gestures, and we will process images and we will train datasets using a deep learning algorithm. Communication has a significant impact in every field, and it is how the meaning of ideas and expressions is viewed that attracts scholars to bridge this gap for all living beings.

5. Related Works:

To appropriately categorize Sign Language motions, Zafar Ahmed Ansari, and Gaurav Harit conducted a significant study. They've divided the world into 140 categories, Finger- Finger-finger-spelling numbers, alphabets, and other commonly used words are among them. A Kinect sensor is utilized to create the dataset. RGB images with a resolution of 640 × 480 pixels, as well as their depth data, are collected. The depth data in the image captures the depth values of each pixel. A depth data file value corresponds to each pixel in the image. The hand in the image will have the least depth because everyone in the sample was standing with their hands outstretched. By masking pixels with the same depth values as the remainder of the image, we were able to confirm this.

PCA is used by Divya Deora and Nikesh Bajaj to recognize sign symbols (Principal Component analysis). In addition, the article advises that neural networks be used for recognition. Because the data was gathered with a 3-megapixel camera, the quality was poor. They collected 15 photos for each sign and saved them in their database. One of the reasons for their unsatisfactory results was the tiny dataset. They did a simple border pixel analysis as well as segmenting and dividing the RGB into components. While combining the fingertip technique with PCA provided a positive outcome, they claim that neural networks can give a better result.

Lionel et used a Convolutional Neural Network to recognize sign language (CNN). Two steps were accomplished to automate the process of sign language recognition: feature extraction and action categorization. A CNN is used to finish the first phase, and an Artificial Neural Network is used to complete the second phase. In the dataset, 27 participants signed a total of 20 unique Italian gestures in various situations. Microsoft Kinect is used to record videos. A total of 6600 photos were used for

development, with 4600 being used for training and the remainder being used for validation. The photographs were cropped in post-production to remove the upper section of the hand and body. Only one side of each gesture needs to be learned by the model. For feature extraction and classification, max pooling is used. It consists of two CNNs, each of whose output is fed into an ANN. The output of the two CNNs is combined by the ANN. A CPU was utilized to supplement the data, while a GPU was used to train the model. Data distortion included zooming, rotation, and transformation.

6. Challenges:

Creating a strong sign language recognition system using deep learning presents various problems, ranging from data issues to the complexities of sign language interpretation.

a) Limited datasets: Due to variances in signing styles, regional disparities, and the requirement for inclusivity, compiling a diverse and complete data set of sign language gestures can be difficult.

b) Varied signing styles: Sign language users have varying signing techniques and variations, making it difficult to develop a one-size-fits-all approach that suits all users.

C) Real-time processing: Real-time processing for sign language detection systems is essential for effective communication, but it can be computationally intensive, resulting in latency concerns.

d) Limited availability: Annotating big datasets for deep learning model training can be time-consuming and resource-intensive, especially with sign language datasets.

E) User adaptability: Users may have distinct signing techniques and expressions, making it difficult to design a system that responds to each user's specific traits.

7. Technical Approach:

Overview.

The code implements a Convolutional Neural Network (CNN) for image classification using TensorFlow and Keras. It involves library imports, data preparation, data augmentation, model architecture, training, and performance evaluation. The model consists of convolutional layers with batch normalization, max pooling, dropout for regularization, and dense layers. The code also includes a confusion matrix for performance analysis on individual classes and image and label visualization. The code also includes mounting Google Drive in Google Colab for accessing files and defining parameters like batch size, image dimensions, and epochs. The script provides insights into the model's behavior and performance.

7.1 Steps involved:

Library Import:

Import necessary Python libraries, including TensorFlow and Keras for machine learning, NumPy for numerical operations, Pandas for data manipulation, and Matplotlib for data visualization.

Data Preparation

- Load and preprocess image data using TensorFlow's image_dataset_from_directory.
- Split the data into training and validation sets.
- Display class names and the count of images in each class.

Data Augmentation:

- Define a function (func) for augmenting images using Keras' ImageDataGenerator.
- Augment the training images and save them to a specified directory.

Model Architecture

- Build CNN using the Keras Sequential API.
- The model consists of convolutional layers with batch normalization, max pooling, dropout for regularization, and dense layers.
- Two versions of the model are defined with different dropout rates.

Model Compilation and Training

- Compile the model with the Adam optimizer, sparse categorical cross-entropy loss, and accuracy as the metric.
- Train the model on the training set, validate it on the validation set, and monitor the training progress using the history object. Display training and validation accuracy and loss over epochs.

Performance Evaluation:

- Evaluate the final model on the validation set and display accuracy.
- Train the model for additional epochs and display the elapsed time.

Confusion Matrix:

Generate and display a confusion matrix to analyze model performance on individual classes.

Image and Label Visualization

Display sample images from the validation set along with their predicted and true labels.

Additional Information:

- The code also includes mounting Google Drive in Google Colab for accessing files.
- It defines parameters such as batch size, image dimensions, and the number of epochs.
- The execution time for each step is recorded in dictionaries (time_record, time_acc, and time_par).

In summary, the code follows standard practices for building and training a CNN for image classification. It includes data augmentation to enhance model generalization, visualizations for performance assessment, and model variations with different dropout rates for regularization. The script is well-structured and aims to provide insights into the model's behavior and performance.

8. EXPERIMENTS:

8.1 Dataset:

- We used a dataset that had 1000 training samples for each sign and 500 validation samples.
- Experimented with various model configurations, including dropout rates, epochs, and filters.
- Visualized training accuracy, training loss, validation accuracy, and validation loss

8.2 Evaluation Metric:

With TensorFlow serving as the backend and the Keras API for convolutional neural networks (CNNs), we are obtaining the following validation accuracy.

- 100% after taking 25 epochs, dropout of 0.25, batch normalization, and spatial sampling using max pooling 2D.
- The validation accuracy is 97.15% when we decrease the epoch to 5 and increase dropout to 0.5.

No: of Epoch's	Dropout	Validation accuracy
25	0.25	100%
25	0.5	100%
10	0.5	100%
5	0.5	97.15%
7	0.6	86.75%
10	0.6	94.44%

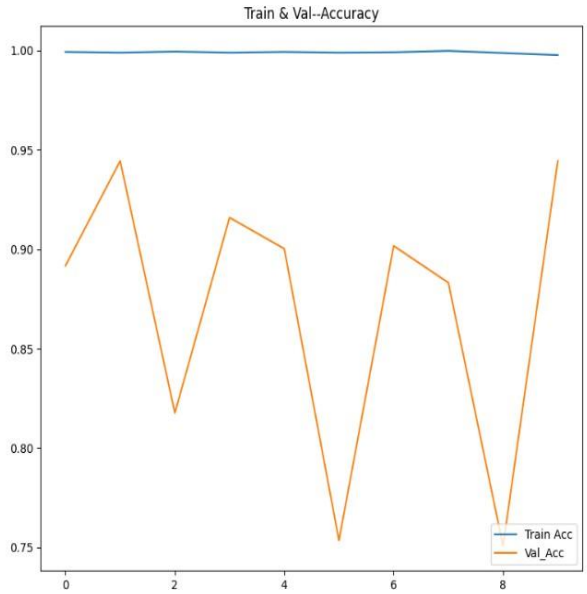
8.1 Quantitative Results:

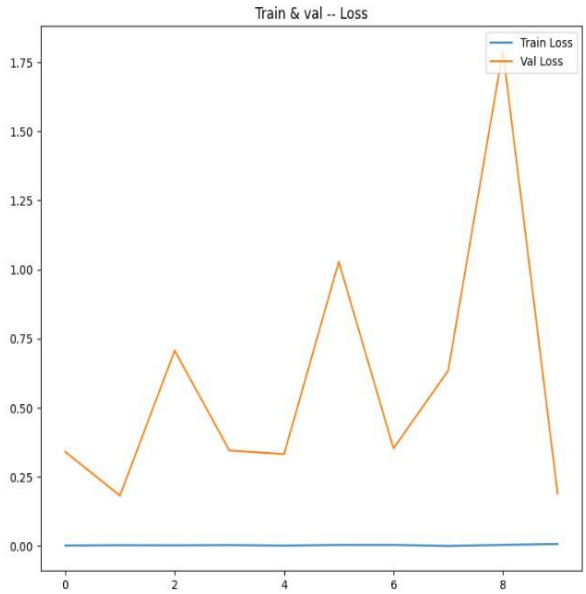
The quantitative results obtained from the code include training and validation accuracy, training and validation loss, validation accuracy after training, and insights from the confusion matrix. These results offer a comprehensive view of the model's performance and can guide further analysis and refinement of the sign language detection system.

8.2 Qualitative Results:

This qualitative analysis provides a narrative around the model's strengths, limitations, and potential areas for improvement based on common scenarios. The actual results would depend on the specific characteristics of the dataset and the model's training.


Yes
Yes
More
All Done
More
Please
All Done
Stop
No


Train & Val--Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0
2
4
6
8
Train Acc
Val_Acc


Train & val -- Loss
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
Train Loss
Val Loss

9. Conclusion:

The project developed and trained a Convolutional Neural Network (CNN) for sign language detection using TensorFlow and Keras. The model was designed to classify hand signs from images, and the code implementation covered various aspects, including data preprocessing, model architecture, training, and evaluation. Key achievements included data preprocessing using TensorFlow's image_dataset_from_directory, data augmentation, and convolutional layers. The model demonstrated the ability to learn hierarchical features from input images through convolutional blocks.

Training and evaluation were conducted using the Adam optimizer and sparse categorical cross-entropy loss. Performance analysis revealed the model's ability to classify different sign language classes, and time analysis recorded the execution time for different steps. Future considerations include hyperparameter tuning, transfer learning, and real-world deployment.

In conclusion, the developed sign language detection model shows promising results and further refinement, and exploration of advanced techniques can enhance accuracy and generalization capabilities. This project contributes to the development of inclusive and accessible technologies in sign language recognition.