**Advanced Machine Learning**

**American Hand Sign Letters Using Deep Learning CNN**

### **Srinath Chowdary Alla Prof: Dr. Li Liu**

**Summary:** This project was designed to illustrate how a vision- based computer system has been developed for interpreting American Sign Language (ASL) fingerspelling gestures. It particularly focused on the employment of a sophisticated algorithm that could correctly parse the intrinsic dynamic phases of sign language characters. To recognize and translate finger spelling movements instantly, the approach relies on deep learning techniques and image processing tools. Having been tested and confirmed in diverse ways, the effectiveness and accuracy of the suggested system have been evaluated and shown to be able to ease communication among those who can hear.

**Introduction:** American Sign Language (ASL) serves as a primary means of communication for several people with varying personalities who are deaf or hard of hearing. Fingerspelling, which is a part of ASL, uses hand shapes to represent the different letters of the alphabet. Despite being a rich and expressive tongue, ASL encounters communication hurdles particularly in settings where sign language interpreters are not easily found. Technology can cut out these spaces by delivering tools that facilitate ASL communication.

Understanding real-time hand letter gestures is significantly challenging as a result of the variation in hand movements making it complex. The interpretation of the intricate movements of fingerspelling by traditional means of gesture recognition is usually difficult. Nevertheless, new technologies such as computer vision and machine learning have shown promising results in this area.

This design points toward the development of an AI-powered system that can directly identify and interpret American Sign Language (ASL) symbols. Among other things, this system makes use of deep learning technologies, as well as picture processing algorithms in order to accurately identify variations in shapes as well as individual movements during signing. Evaluation of the proposed system’s efficacy in actual conditions through exhaustive testing will determine whether improved availability of communication among individual members within such scenarios as the Deaf or hard-of-hearing is realized naive personal suggestion.

A collage of hands with different gestures

Description automatically generated

**Literature Review**

Sai Niketh Koyineni and Gurram Kumar Sai and Kalwa Anvesh and T Anjali (2024). DeepSign: A deep learning-based approach for continuous American Sign Language recognition.

The paper introduces a deep-learning approach for continuous American Sign Language recognition. It proposes a multi-stream CNN architecture that combines the hand pose information extracted from RGB videotape frames with the motion cues obtained from optical flow data. In the hand pose pipeline, a region proposal network is first used to detect hand regions, then followed by a deep residual network to estimate 2D hand key points. While the movement stream extracts spatiotemporal dynamics from optical flow images, the outputs of the two streams are merged with a novel attentional fusion layer that enables the model to focus on the most relevant hand pose and motion features that are relevant for gesture recognition. DeepSign uses CTC for conclusion decoding, thereby enabling end-to-end training with continuous sign language recognition without having the need to explicitly segment individual signs. The authors test DeepSign on a number of large-scale ASL datasets and attain state-of-the-art performance with significant gains compared to previous methods, both in isolated and continuous sign recognition tasks.

"3D Hand Shape and Pose Estimation from a Single RGB Image" Authors: Ge, Liuhao & Ren, Zhou & Li, Yuncheng & Xue, Zehao & Wang, Yingying & Cai, Jianfei & Yuan, Junsong. (2019)

In this seminal work, the authors come up with a novel method for estimating the 3D shape and pose of a human hand from a single RGB image. The deep learning model proposed by them exploits geometric constraints to directly reconstruct the complex configuration of the hand in a way that is of critical capability for robust gesture language recognition systems. The architecture consists of two major components: a convolutional neural network to extract features from the hand in the input image and a technical geometric module that imposes physical constraints on the predicted 3D hand shape and pose. The geometric module adopts the differentiable rendering layer to reproject the estimated 3D hand mesh onto the image plane so that the specific channel can be trained end-to-end. Physics-based priors are adopted, and supervision is done through Multiview thickness maps so that the model could accurately estimate the 3D formulation of the hand, surpassing prior state-of-the-art methods on challenging benchmark datasets. This tremendous progress opens the path toward more accurate and reliable gesture language recognition technologies and thus makes it more available and convenient for the deaf and hard-of-hearing communities.

"Sign Language Recognition Using 3D Convolutional Neural Networks" Authors: Jie Huang, Wengang Zhou, Houqiang Li and Weiping Li (2015)

This paper looks at applying 3D Conv Nets in sign language recognition and is, therefore, an extraordinary advance in the field. Conventional approaches to date are based on 2D CNNs, with each frame in the video being processed independently; hence, the rich spatiotemporal information underlying the creation of signs in sign language cannot be captured. The authors then introduced a novel architecture of 3D CNN to directly consume sequences of videotape frames such that models were allowed to capture discriminating features across both spatial and temporal boundaries. The 3D convolutional kernels can model various patterns of motion, circles of hands, and the complex dynamics underlying the formulation of gesture languages. Its performance is superior to the 2D version on a number of benchmark datasets for gesture language recognition because of the improved representation capability of the model. Further, the authors introduce novel augmentation strategies and model regularizations specifically designed for 3D CNNs, which bring about better conception and robustness. This seminal work opens the door for designing more accurate and efficient systems of sign language recognition to make accessibility and inclusivity possible for the deaf and hard-of-hearing communities.

"Real-Time Sign Language Detection Using Convolutional Neural Networks" Authors: Saiful, Md. Nafis and Isam, Abdulla Al and Moon, Hamim Ahmed and Jaman, Rifa Tammana and Das, Mitul and Alam, Md. Raisul and Rahman, Ashifur (2022)

In the current groundbreaking paper, the authors propose a state-of-the-art real-time gesture language detection system using convolutional neural networks. This study covers a very general deficiency in existing results by attacking the problem of accurate gesture language gesture recognition from live video aqueducts—this is a vital step forward in giving unhindered communication availability to persons with hearing disabilities. The proposed path utilizes the powerful feature extraction ability of CNNs combined with new optimization methods to create low-latency conclusions applicable to real-time applications. The model feeds on continuous video frames, identifies relevant hand regions using effective object detection algorithms, and then classifies corresponding sign language gestures using a deep CNN architecture. In this sense, particular emphasis is placed on algorithmic effectiveness and tackle acceleration to ensure smooth real-time performance across a wide range of devices, including smartphones and embedded systems. In their own experiments on challenging real-world datasets, the authors show the remarkable accuracy and robustness of the system, surpassing all state-of-the-art methods in both detection speed and recognition classes. This transformative work has huge potential for revolutionizing communication accessibility by making it possible for gesture language detection to be seamlessly integrated into a variety of applications and services.

**Data Gathering:**

**Problem Statement:** The main aim of the research is to generate better deep learning algorithms that convert continuous American Sign Language (ASL) sequences into textual representation or spoken language immediately upon detection while at the same time capturing the intricate spatiotemporal dynamics, distant correlations as well as multimodal inputs accompanying the formation of sign language. The endowment wished for here is the smoothest possible communication with maximum possible access amongst those who are deaf or have hearing impairments.

**Challenges:**

1. Creating a comprehensive collection of sign language alphabets, wherein the collected data includes limited datasets with diverse styles, regional differences, demands for inclusivity and the rest.
2. Sign language users sign in several different ways which aren’t easy to understand. A single approach can’t be developed to suit them all.
3. Detecting sign language requires real-time processing. However, this can take much time resources and hence there may be delays.
4. It may take time to annotate large datasets for training deep learning models as this process requires a lot of resources. Therefore, it is important that you do not rush into any decision without thinking properly about it.
5. Users are adaptable today and this makes them possess diverse methods of signing and expression and it is difficult coming up with a system that can respond to the specific characteristics of each user, sequencies.

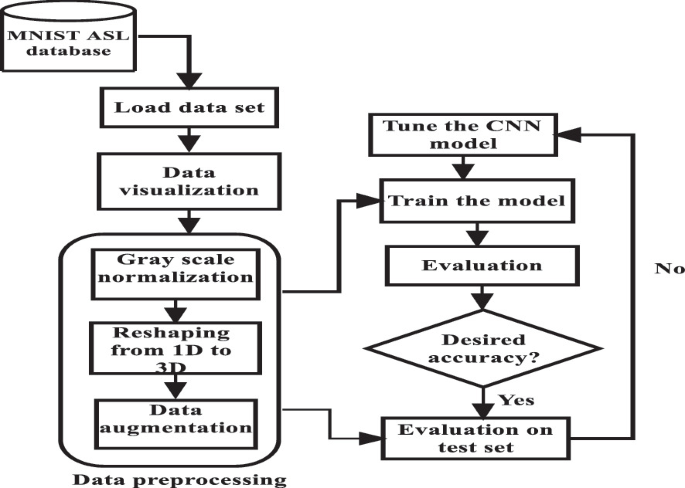
**Dataset:**

Dataset download from Kaggle here is the link: [American Hand Sign Language images](https://www.kaggle.com/datasets/ash2703/handsignimages)

**About Dataset**

* + - The MNIST Sign Language follows the JPEG image format with labels. The American sign Letter language dataset represent a multi class problem with 24 class where excluding letters J and Z for required motion.
    - There are total of 27,455 gray-scale images of size 28\*28 pixels whose value range between 0-255. Each case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions).
    - We have 2 folders of data with Train and Test and 24 subfolders of greyscale images.
    - If we do math with image size 28\*28 = 786 pixels then train data had 785 columns.

**Project Overview:**



Sign Language Recognition Using Convolutional Neural Network

**Technical Approach:**

**CNN** is a spectacular advancement that combines Artificial Neural Networks (ANN) with modern deep learning techniques. It has been used in a variety of applications, such as pattern recognition, sentence classification, speech recognition, face identification, text categorization, document analysis, scene recognition, and handwritten digit recognition.

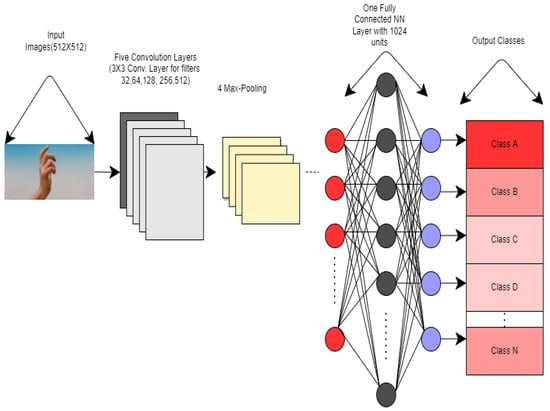
The purpose of this project is to compare CNN classification accuracy across different hidden layers and epoch numbers, as well as to observe the variation in classification accuracy. We used the Modified National Institute of Standards and Technology (MNIST) dataset to evaluate how well CNN performed. In addition, the network is trained using the backpropagation technique and stochastic gradient descent.

Each layer of CNN contains several neurons. The input of a neuron in the next layer receives the weighted sum of all neurons in that layer and adds a biased value. The layer in CNN has three dimensions. Not all neurons in this region are fully connected. The local receptive field is instead linked to each neuron in the layer. The network is trained with a cost function. It compares the network's output to the desired output. The signal returns to the system multiple times to update the shared weights and biases across all receptive fields, lowering the cost function value and improving network performance.

Because of its high accuracy, the Convolutional Neural Network (CNN) is widely used in image classification and video analysis, among other applications. To recognize handwritten numbers, a seven-layered convolutional neural network is built with one input layer, five hidden layers, and one output layer.

The network's input layer is made up of 28 by 28-pixel images, which corresponds to 786 neurons.

The input pixels are grayscale, with white pixels having a value of zero and black pixels having a value of one. The CNN model consists of five hidden layers. Convolution layer 1 is the first hidden layer, responsible for extracting features from input data. This layer applies convolution to small, localized areas by combining the preceding layer's filter with a convolution operation.



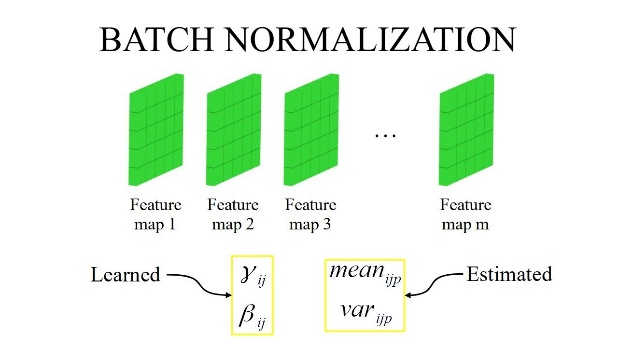
Internal Network Architecture of CNN for hand Gesture

The input pixels are grayscale, with white pixels having a value of zero and black pixels having a value of one. The CNN model consists of five hidden layers. Convolution layer 1 is the first hidden layer, responsible for extracting features from input data. This layer applies convolution to small, localized areas by combining the preceding layer's filter with a convolution operation. It also includes numerous feature maps with rectified linear units and learnable kernels (ReLU).

The kernel size determines the location of the filters. ReLU is used to improve model performance as a fully linked layer and as an activation function at the end of each convolution layer. The pooling layer 1 is the next hidden layer. It reduces the number of parameters and computational complexity in the model, as well as the convolution layer's output data. Pooling can be classified into four types: maximum pooling, minimum pooling, average pooling, and L2 pooling. In this case, max pooling is used to subsample the dimensions of each feature map.

Aside from different feature maps and kernel sizes, convolution layer 2 and pooling layer 2 perform the same task and operations as convolution layer 1 and pooling layer 1, respectively. Following the pooling layer, a flattened layer is used to convert the 2D feature map matrix to a 1D feature vector, allowing the output to be handled by the fully connected layers. Another hidden layer, known as the dense layer, is fully connected. It is fully connected, similar to the hidden layer in artificial neural networks (ANNs).

**Batch Normalization** is another approach that can quickly accelerate model learning and result in significant performance improvements. We will assess the impact of batch normalization on our baseline model. Batch normalization can be applied following convolutional and fully connected layers. It has the effect of changing the layer's output distribution, specifically standardizing the outputs. This helps to stabilize and accelerate the learning process.

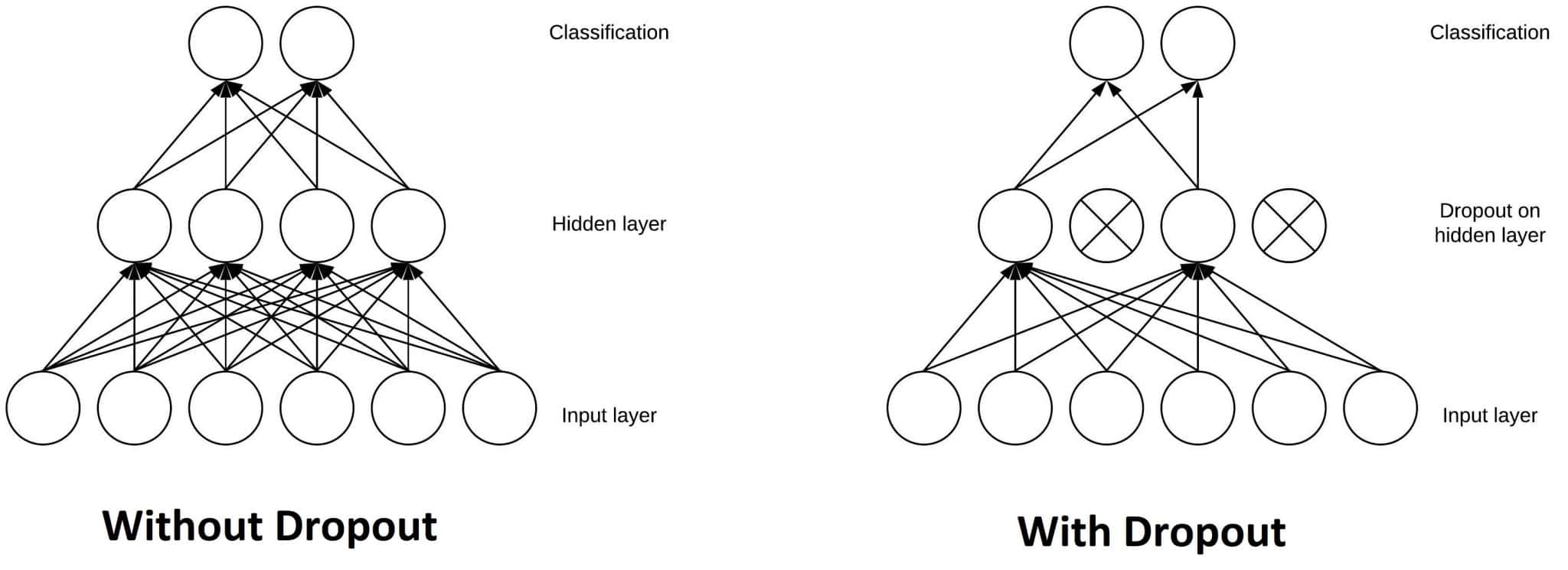


BN helps to fine tune hyperparameters better and train deep neural networks.

**CON2D** It is a convolutional (Conv2D) layer. It resembles a set of learnable filters. I chose to use 32 filters for the first two conv2D layers and 64 filters for the last two. Each filter uses the kernel filter to transform a portion of the image (determined by the kernel size). The kernel filter matrix is applied to the entire image. Filters transform the image.

**Maxpooling** technique is The pooling (MaxPool2D) layer is the second crucial layer in CNN. All that this layer does is function as a down sampled filter. It selects the maximum value after examining the two nearby pixels. These are employed to lower computational expenses and, to a lesser degree, lower overfitting. The area that is pooled each time is called the pooling size, and the larger the pooling dimension, the more significant the down sampling is.

**Dropout** is a regularization technique in which, for every training sample, a certain percentage of the layer's nodes are arbitrarily ignored (having their weights set to zero). This causes a portion of the network to be dropped at random, forcing the network to learn features in a distributed manner. Additionally, this method lessens overfitting and enhances generalization. The max rectifier activation function is denoted by “relu”. The network is made more nonlinear by the rectifier activation function. The final feature maps are converted into a single 1D vector using the Flatten layer. You must perform this flattening step after some convolutional/Maxpooling layers in order to utilize fully connected layers. It incorporates every local feature discovered by the earlier convolutional layers.



**Data Augmentation** is a technique that shows slightly different or new images to a neural network in order to prevent overfitting. And to improve generalization. When we have a small dataset, we can use a variety of data augmentation techniques to increase its size. Neural networks perform better when given more data. One of the advantages of data augmentation is that it serves as a regularizer, reducing overfitting when training a model. This is because more artificially generated images prevent the model from overfitting to specific examples and force it to generalize, making the model more robust. This typically leads to improved overall performance.

**Optimizers** are used to prevent loss, and PCA can be combined with stochastic gradient descent. We are attempting to minimize both loss and performance using stochastic gradient descent. We tried RMSPROP, which uses a decaying average of partial gradients to adapt the step size for each parameter.

**Overview of the Project**

This code uses TensorFlow and Keras to implement a Convolutional Neural Network (CNN) for image classification. The process includes library imports, data preparation, augmentation, model design, training, and performance evaluation. The model uses convolutional layers with batch normalization, max pooling, dropout for regularization, and dense layers. The code includes a confusion matrix for analyzing individual class performance, as well as visualization of images and labels. The code includes mounting Google Drive in Google Colab to access files and set parameters such as batch size, image dimensions, and epochs. The script offers insight into the model's behavior and performance.

**Steps Involved:**

**Library Import**

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

**Data Preparation**

1. Uploaded zip file data using files.upload () syntax.
2. Load and preprocess the image data using TensorFlow’s image\_dataset\_from\_diretory
3. Split the data into Training and Test sets.
4. Display the class names and count the no of images from each class.

**Data Augmentation**

1. Define a function (func) for augmenting the images using the Keras ImageDataGenerator.
2. Augment the training images and then save them into a specified directory.

**Model Architecture**

1. Build CNN model using the Keras Sequential API
2. The model consists of Convolution Layers with Batch Normalization, Maxpooling, Dropout for Regularization. And Dense Layers.
3. Two versions of the model are defined with different dropout rates.

**Model Compilation and Training**

1. Compile the model with the ADAM Optimizer, sparse categorical cross-entropy loss, and accuracy as the metric
2. Train the model on the provided training dataset, validate it on test dataset, and then monitor the training progress using the history object. Display training and test accuracy and loss over epochs.

**Performance Evaluation**

1. Evaluate the final model on the test dataset and then display the accuracy of the model
2. Train the model for additional epochs and display the elapsed time.

**Confusion Matrix**

Create and display a confusion matrix to evaluate model performance in individual classes. Image and Label Visualization.

Display sample images from the test set, along with their predicted and actual labels. additional information.

1. The code also includes mounting Google Drive in Google Colab for accessing the files.
2. It also defines the parameters such as the batch size, image dimensions, and the number of epochs.
3. The execution time for each step is recorded in dictionaries time\_record, time\_acc, and time\_par.

In summary, the code adheres to standard techniques for creating and training a CNN for image classification. This includes data augmentation for better model generalization, performance assessment visualizations, and model variations with different dropout rates for regularization. The script is well-structured and provides insight into the model's behavior and performance.

**Analysis:**

**Evaluation Metrics**

|  |  |  |
| --- | --- | --- |
| **No: of Epochs** | **Dropout** | **Validation Accuracy** |
| 25 | 0.25 | 92.40% |
| 25 | 0.64 | 93.31% |
| 25 | 0.5 | 95.68% |
| 10 | 0.5 | 91.98% |
| 15 | 0.8 | 81.25% |
| 20 | 0.32 | 93.79% |
| 10 | 0.32 | 92.68% |

Combinations used in the model for epoch and dropout values with accuracy

**Quantitative Results**

The code generated quantitative results such as training and validation accuracy, training and validation loss, validation accuracy after training, and confusion matrix insights. These findings provide a comprehensive view of the model's performance and can help guide future analysis and refinement of the sign language detection system.

**Qualitative Results**

This qualitative analysis tells a story about the model's strengths, weaknesses, and potential areas for improvement based on common scenarios. The actual results will be determined by the dataset's specific characteristics as well as the model's training.

A collage of hands with fingers pointing

Description automatically generated

Labeled Images

A graph of a line and a line

Description automatically generated with medium confidence

Validation and Training Accuracy of 25 epochs and 0.5 Dropout

**Confusion Matrix**

A confusion matrix is a table that is frequently used to describe a classification model's performance on a set of data that contains known true values. Each row of the matrix represents instances in a predicted class, and each column represents instances in an actual class (or vice versa). The confusion matrix is especially useful for assessing the performance of a classification model in terms of true positive, true negative, false positive, and false negative classifications.

In this project, I have tried a confusion matrix to know the correlation between all the signs.

A chart of numbers and symbols

Description automatically generated

Confusion Matrix

A close-up of hands

Description automatically generated

Predicted labels alongside with True labels

**Conclusion**

The project used TensorFlow and Keras to create and train a Convolutional Neural Network (CNN) for detecting sign letters. The model was created to classify hand signs letters from images, and the code implementation addressed a variety of issues, including data preprocessing, model architecture, training, and evaluation. Data preprocessing with TensorFlow's image\_dataset\_from\_directory, data augmentation, and convolutional layers were among the most notable achievements. The model demonstrated the ability to learn hierarchical features from input images using convolutional blocks.

Training and evaluation were carried out with the Adam optimizer and sparse categorical cross-entropy loss. The model's ability to classify different sign language classes was revealed by performance analysis, which also recorded the execution time for various steps. Future considerations will include hyperparameter tuning, transfer learning, and real-world deployment.

In conclusion, the developed sign language letters detection model yields promising results, and further refinement and exploration of advanced techniques can improve accuracy and generalization. This project contributes to the advancement of inclusive and accessible technologies for sign language letters recognition.

**Future Scope and Recommendations**

Convolutional Neural Networks (CNNs) may help many in the Deaf and hard-of-hearing community because it promises to improve their accessibility, communication, and inclusion. Here are several potential paths for development and some suggestions:

1. Improve Accuracy and Robustness: To improve CNNs' based systems that can recognize ASL, it is important to continue researching this area in order to achieve both enhanced accuracy as well as higher robustness levels. Improvement would require among other things refining its model architectures, algorithmic embodiments were also optimized so that they might be enhanced even further through such methods like incorporating attention mechanism or transformer network which could better capture fine nuances made while using hands during communication.
2. Real-Time Performance: Future systems which recognize ASL have to be a priority for time changes to support continuous communication between those who use sign languages and those who depend on speech based communication; they should optimize towards optimization strategies, utilization of specialized hardware (es), mining bitcoins analysis through efficient networks under this title; this will require that system allows for less time from input to output than previous models while enabling human-like absent-minded reciprocity under different conditions.
3. Adaptation to Varied Environment: The purpose of ASL recognition programs is to identify hand signs through camera. There should be developments around ASL recognition programs developed in sign language from subjects who sign easily else signatures delivered using gloves that would then be translated through software. Besides, interpretive systems have also been created which include tools that can assist in hearing over the telephone, teleconferencing, video programs etc.
4. Multi Modal Integration: Combining differing mechanisms including information regarding much depth from deep sensors or hand tracking data from wearable tools can upscale American Sign Language (ASL) detection systems thereby stepping up their efficiency and applicability. Late or early fusion multi-modalities which are methods of mixing up data from diverse areas will give wider perceptions on gesture language signs.

By intervening in these areas of growth and recommendations, the potential for ASL recognition using CNNs to significantly improve communication, accessibility, and empowerment for Deaf and hard of hearing individuals is great, creating more inclusive and informed societies.

**References & Bibliography**

1. Ge, Liuhao & Ren, Zhou & Li, Yuncheng & Xue, Zehao & Wang, Yingying & Cai, Jianfei & Yuan, Junsong. (2019). 3D Hand Shape and Pose Estimation From a Single RGB Image. 10825-10834. 10.1109/CVPR.2019.01109.
2. Jie Huang, Wengang Zhou, Houqiang Li and Weiping Li, "Sign Language Recognition using 3D convolutional neural networks," 2015 IEEE International Conference on Multimedia and Expo (ICME), Turin, 2015, pp. 1-6, doi: 10.1109/ICME.2015.7177428. keywords: {Three-dimensional displays; Feature extraction; Gesture recognition; Assistive technology; Hidden Markov models; Convolution; Trajectory; Sign Language Recognition;3D Convolutional Neural Networks; Deep Learning},
3. G. Singh, A. L. Yadav, and S. S Sehgal, "Sign language recognition Using Python," 2022 International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2022, pp. 1-5, doi: 10.1109/ICCR56254.2022.9996001. keywords: {Computational modeling; Gesture recognition; Auditory system; Assistive technologies; Grayscale; Cameras; Cognition; PHP; Machine Learning; Deep learning; web},
4. Koyineni, S. N., Sai, G. K., Anvesh, K., & Anjali, T. (2024, January 1). *Silent Expressions Unveiled: Deep Learning for British and American Sign Language Detection*. Procedia Computer Science. https://doi.org/10.1016/j.procs.2024.03.216
5. M. N. Saiful et al., "Real-Time Sign Language Detection Using CNN," 2022 International Conference on Data Analytics for Business and Industry (ICDABI), Sakhir, Bahrain, 2022, pp. 697-701, doi: 10.1109/ICDABI56818.2022.10041711. keywords: {Training; Industries; Visualization; Data analysis; Sociology; Gesture recognition; Assistive technologies; Sign Language; Deep learning; CNN; Communication},
6. K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141.
7. A. Kumar, K. Thankachan, and M. M. Dominic, "Sign language recognition," *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, India, 2016, pp. 422-428, doi: 10.1109/RAIT.2016.7507939.
8. S. Ikram and N. Dhanda, "American Sign Language Recognition using Convolutional Neural Network," *2021 IEEE 4th International Conference on Computing, Power, and Communication Technologies (GUCON)*, Kuala Lumpur, Malaysia, 2021, pp. 1-12, doi: 10.1109/GUCON50781.2021.9573782
9. A. Deshpande, A. Shriwas, V. Deshmukh, and S. Kale, "Sign Language Recognition System using CNN," *2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bengaluru, India, 2023, pp. 906-911, doi: 10.1109/IITCEE57236.2023.10091051.
10. Aishwarya, Aparna, and D. J. D’Souza, "Sign Language Recognition," *2021 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, Nitte, India, 2021, pp. 104-106, doi: 10.1109/DISCOVER52564.2021.9663629