A Deep Learning-Based Algorithm for American Sign Language Recognition

Sai Prashanthi Gumpili[1]-saipg@udel.edu,
Murali Nagulla[1]-muralik@udel.edu,
Rakesh Emuru[1]-rakeshe@udel.edu,
Fatima Moeini Pour[1]-fmoeini@udel.edu,
Vasavi Prasanna Kurapati[1]-vasavipr@udel.edu,
Kishore Kumar Reddy Madithati[1]-kmaditha@udel.edu
[1]MS Student in Data Science-University of Delaware

***Abstract*—This project aims at developing a robust Convolutional Neural Network (CNN) for the efficient categorization of American Sign Language (ASL) signs. The primary dataset for this project is sourced from Kaggle and comprises 2,515 high-resolution images. These images represent 36 distinct classes, including the digits 0-9 and the letters A-Z as expressed in ASL. The project plans to leverage the power of deep learning tools such as Keras, TensorFlow, and Python to train a CNN model capable of multi-class image classification. Initially, a prototype CNN model is developed to assess the overall accuracy of the dataset. The process involves several stages, including data reprocessing, model development, hyper-parameter tuning, model evaluation, and prediction. Next, various data augmentation techniques such as rotation, flipping, and zooming are incorporated to enrich the dataset and ensure that the model is trained on a diverse set of images, thereby increasing its generalization capabilities. The objective of this project is to narrow the communication divide between individuals with speech impairments and the broader community, via a more robust ASL recognition model, intending to foster a more inclusive society.**

I. Introduction

Developed in the early 19th century, American Sign Language (ASL) serves as a crucial visual-gestural language for the Deaf population in the US and certain areas of Canada. Its roots trace back to a collaboration between a hearing minister and a Deaf educator, leading to a unique linguistic structure distinct from spoken languages, expressed through handshapes, facial expressions, and body movements [1]. ASL fosters communication and community within the Deaf population, enabling them to express themselves, share experiences, and access information. Recognizing ASL's importance, advancements in technology like image processing as well as artificial intelligence (AI) are propelling the growth of ASL recognition systems. These systems bridge communication gaps between the Deaf and hearing communities, allowing seamless engagement with the wider world. This intersection of technology and sign language marks a significant milestone in enhancing accessibility and inclusivity for the Deaf population.

In this project, we address ASL recognition using a convolutional neural network (CNN)-based approach. Our objective is to assess and refine a deep learning technique capable of recognizing ASL signs from provided images, relying on models trained using a publicly available ASL dataset [2].

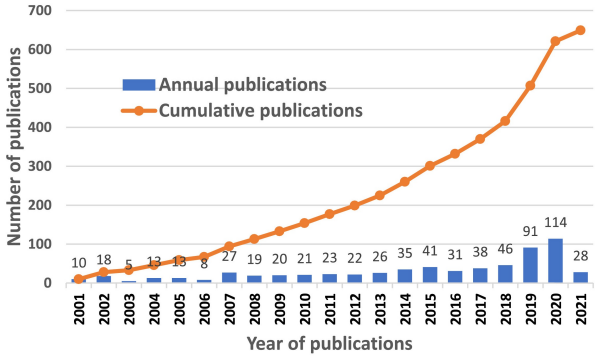

Fig. 1. Distribution of publications on AI-based ASL recognition from 2001-2021 [4].

II. Related work

The ASL dataset has been widely explored and extensively used in various computer vision tasks. Over the past two decades, researchers have shown a keen interest in integrating AI into ASL, resulting in a wealth of literature discussing advanced methods and techniques [3]. Figure 1, illustrates the escalating trend in publications within this field from 2001 to early 2021, encompassing over 600 articles during this timeframe.

Tracing the literature from the earlier publications, it can be seen that in the past, researchers experimented with hardware-based solutions like sensor-equipped gloves to interpret sign language. While effective to some extent, these methods were often uncomfortable and restrictive, limiting natural communication [3]. With advancements in computer vision, researchers have shifted their focus to camera-based recognition systems. This allowed for a more natural and unobtrusive way to interpret sign language.

With the inclusion of AI and machine learning, traditional machine learning algorithms, like Hidden Markov Models (HMMs), were initially used to decipher the sequential nature of sign language. However, these models struggled to handle the complexity and variations inherent in real-world sign language communication [4].

The emergence of deep learning, particularly CNNs, brought a revolution to image recognition tasks. This paved the way for researchers to apply these powerful techniques to ASL

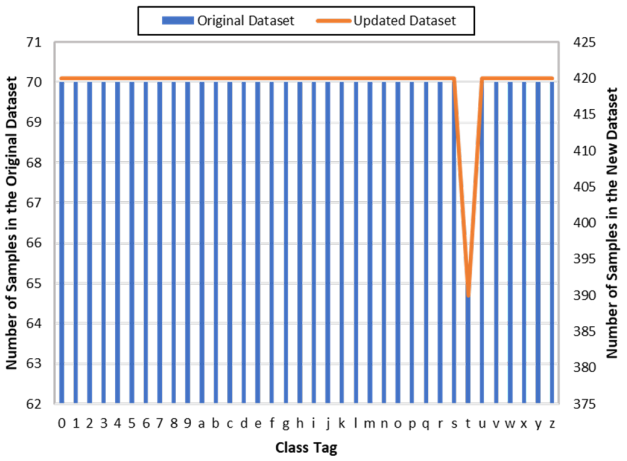

Fig. 2. Distribution of samples in different classes.

recognition with remarkable success. Studies like [5] have demonstrated how CNNs can be effectively trained to recognize hand gestures and postures, forming the core of modern ASL recognition systems.

Training deep learning models requires a vast amount of diverse data. However, obtaining sufficient ASL data can be challenging. To overcome this hurdle, researchers have employed data augmentation techniques, such as image mirroring, rotation, and cropping, to artificially increase the size and diversity of the training data. As shown in [3], these techniques significantly boost the robustness and accuracy of deep learning models for ASL recognition. These studies have paved the way for our approach to identifying ASL.

III. Data and Methods

A. The Original Dataset

The initial dataset comprised 2,515 images of ASL hand signs. These images were organized into 36 directories, labeled 'a' through 'z' and '0' through '9', corresponding to the hand sign they represent. Each directory contained approximately 70 images, except 't' which included 65 images, showcasing the respective hand sign shot from various angles. This diversity in perspectives aimed to capture the complexity and variability inherent in ASL. Figure 2 represents the distribution of images in different classes.

B. The Updated Dataset

One of the primary challenges associated with the original dataset was its relatively small size, potentially restricting the deep learning model's capacity to learn and generalize effectively. More specifically, with only around 70 images per sign, there were concerns about the representational sufficiency of the dataset, particularly in capturing the nuances of each sign. Consequently, we implemented data augmentation techniques, like rotating, mirroring, etc, to augment the database by six times. The resulting dataset comprises close to 15090 images, and the distribution of images in each class is illustrated in Fig. 2. As can be seen in this figure, each directory contained around 420 images in the updated dataset which is a significant increase in both size and diversity.

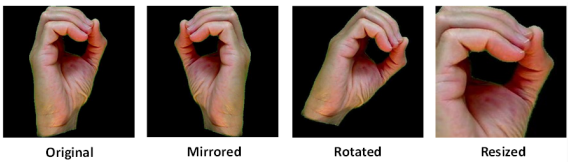

Fig. 3. Application of augmentation techniques to an original image of sign '0'.

The data augmentation techniques employed and the rationale behind each are explained below in more detail.

1) Mirroring: All the images in the original dataset were taken from a right-handed person. Studies have shown that approximately 10% of the population is left-handed [6]. With this respect, images were mirrored horizontally to simulate the appearance of left-hand signs from right-hand sign images, effectively doubling the dataset size.

2) Rotation: All the images in the original dataset were taken in a fully vertical orientation with zero rotation. However, in actual situations, people may not always position their hands vertically. To cover this aspect, all the images were rotated at different angles, up to 180 degrees. This process aimed to introduce variability in the dataset by presenting the signs in different orientations, thus enhancing the model's ability to recognize signs irrespective of their orientation.

3) Resizing: The hand position in the original dataset is situated at the center of the image. However, in actual situations, if someone raises their hand to show an ASL sign, they might be closer or farther from the camera, affecting hand size in the picture and potentially impacting prediction accuracy. To address this issue, the images were resized to maintain consistency in input size for the deep learning model, ensuring that all images were of uniform dimensions.

Figure 3 depicts the application of the above-mentioned augmentation techniques to an original image showing the sign '0'.

4) Additional of Custom Images: To further enrich the dataset, 10 custom images of hand signs were added to each directory. These images were personally captured and added to the dataset. The addition of these custom images served to introduce new variations and real-world scenarios to the dataset, potentially aiding the model in learning more realistic and practical representations of each sign.

C. The Employed Methods

In order to tackle the ASL recognition problem, a CNN-based technique is developed. The architecture of the CNN Classification model is composed of three main layers as shown in Fig. 4 and explained below in detail:

1. *Convolution Blocks*: The model consists of three convolution blocks, each with an increasing number of filters (32, 64, and 128, respectively). Each block utilizes the Rectified Linear Unit (ReLU) activation function and includes dropout layers with rates of 0.2, 0.3, and 0.4, respectively. Each convolution block also contains a Max Pooling layer with a pool size of 2.

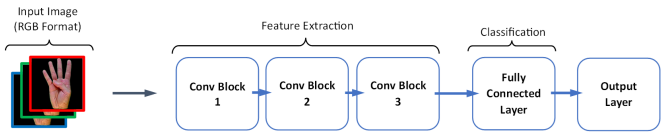

Fig. 4. The architecture of the developed CNN classification.

2. *Fully Connected Layers*: The fully connected layers include a flatten layer, followed by a Dense layer with 512 units. This is accompanied by a dropout layer with a rate of 0.2. Subsequently, another Dense layer with 128 units is included, with a dropout rate of 0.3. The ReLU activation function is used in these layers.

3. *Output Layer*: The output layer is a Dense layer with 36 units and employs the Softmax activation function.

In addition, just for the sake of comparison, a decision tree (DT))-based algorithm is also developed. To initialize the DT model 'max_depth', 'min_samples_split', and 'min_samples_leaf' are all set to 1.

D. The Solution Process

The new dataset is randomly shuffled and then split into the training (80%), validation (10%), and testing (10%) sets. Each image file is reduced to 200*200 pixels, and saved in an RGB format.

Initially, Decision Tree and CNN methods are applied to both datasets, and the results are compared against each other. Once the performance of the CNN method is approved, the hyper-parameter tuning is carried out via manual sensitivity study checks. During this step, the number and size of the layers as well as the activation functions are tested. It is observed in emperical simulations that the best parameters for our dataset include optimizer adam, categorical cross entropy as loss, batch size of 32 and 50 epochs. The architecture depicted in Fig. 4 and the details shared in Section III.C are associated with the best parameters obtained from this process.

IV. Experimental Results

A. Initial Machine Learning Models

1) Approach and Rationale: Before diving into deep learning, we explored simpler machine learning (ML) models. This was driven by the understanding that ML models generally require less computational complexity, time, space, and energy compared to deep learning models.

B. Models and Performance:

1) K-Nearest Neighbors (KNN): We implemented a KNN model, which yielded an accuracy of 71%. This model was chosen for its simplicity and effectiveness in classification tasks.

2) Decision Tree: A Decision Tree model was also tested, achieving a slightly higher accuracy of 75%. Decision Trees were selected for their interpretability and ease of implementation.

3) Contextualizing the Results: It's important to note that the task at hand is a multi-class classification problem, involving 36 classes (26 letters and 10 digits). Therefore, the baseline probability of a correct guess is 1/36, making the obtained accuracies appreciable for such simple models.

C. Deep Learning Approach - Convolutional Neural Network (CNN)

1) Transition to CNN: Based on the limited success of the ML models, we shifted our focus to a more complex but potentially more accurate approach using Convolutional Neural Networks (CNN).

2) Experimenting with Optimizers and Learning Rates: We experimented with various optimizers and learning rates to find the optimal configuration. Some of the optimizers tested included Adam, SGD (Stochastic Gradient Descent), and RMSprop. The learning rates varied from 0.001 to 0.01 in our experiments.

TABLE I
Obtained Accuracy Results on the Validation Set

Method	Accuracy
Decision Tree	75.94%
Initial CNN	95.42%
Tuned CNN	96.75%

D. Hyper-parameter Sensitivity Assessment

Hyperparameter tuning is conducted to enhance the accuracy of our model. We explored various batch sizes (16, 32, 64, and 128), among which a batch size of 32 demonstrated the optimal balance between learning speed and stability. Empirical observations revealed that the 'Adam' optimization algorithm is the most suitable choice for our specific dataset, attributed to its adaptive learning rate and efficient handling of sparse gradients. Furthermore, 'ReLU' emerged as the most suitable activation function, owing to its simplicity and effectiveness in preventing vanishing gradients.

The choice of callbacks was varied; we assessed alternatives such as 'ModelCheckpoint' and 'LearningRateScheduler', in addition to 'ReduceLROnPlateau' and 'EarlyStopping'. Our investigation revealed that the dynamic adjustment provided by 'ReduceLROnPlateau', coupled with the overfitting prevention offered by 'EarlyStopping', most effectively met our model's requirements. The process was EarlyStopped at 21 Epochs. Given the nature of our multi-class classification problem, we opted for Categorical 'Crossentropy' as the loss function. This function gauges the performance of a classification model whose output is a probability value ranging from 0 to 1.

E. Obtained Results of the Tuned CNN Model

As depicted in Table I, the sensitivity analysis of the hyperparameters enabled us to fine-tune the CNN model, resulting in a 1.33% improvement in the accuracy of the validation set. To delve deeper into and discuss the outcomes of the optimized model, we present exploratory results in Figs. 5-7 and Table

II. As depicted in Figs. 5-6, a significant development occurs at epoch 5, where a plot twist reveals that our training loss starts to decrease. This indicates the model's proficiency in learning from the provided training dataset. However, the test loss shows a slight increase, suggesting that the model may be encountering a degree of overfitting.

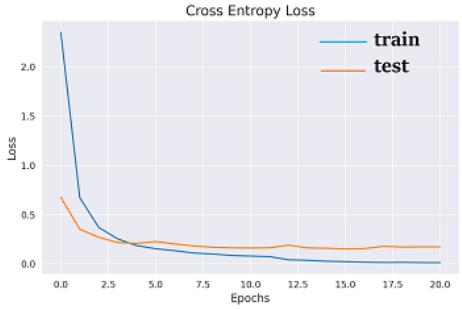

Fig. 5. Cross entropy loss of the tuned model.

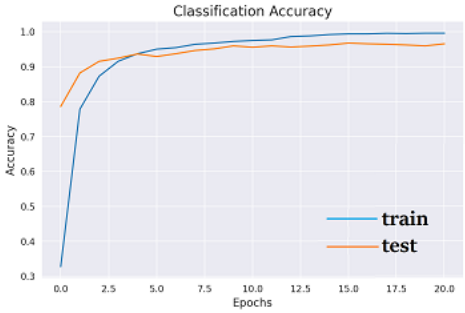

Fig. 6. Classification accuracy of the tuned model.

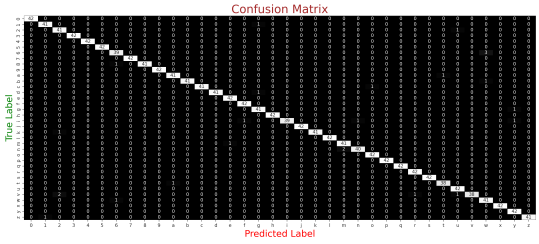

Fig. 7. Confusion matrix of the tuned CNN model on the test data.

TABLE II
OBTAINED RESULTS OF THE TUNED CNN MODEL

Metric	Training	Validation	Test
Accuracy	100.00%	96.75%	98.34%
Loss	0.0002	0.15	0.06

Table II displays the comprehensive accuracy and loss outcomes across the training, validation, and test sets. Notably, the fine-tuned CNN model exhibits an impressive 100.00% accuracy on the training dataset and a commendable 96.75% accuracy on the validation dataset. This metric serves as a gauge for the model's ability to generalize effectively to novel, unseen data. Moreover, the model's conclusive assessment on the test dataset reveals an accuracy of 98.34%.

The model's low loss on the training dataset, registering at 0.0002, signifies a robust fit to the training data. Conversely, the validation dataset exhibits a slightly elevated loss of 0.15 compared to the training loss, suggesting the potential application of regularization or enhanced generalization techniques. The loss on the test dataset, assessing the model's performance on unseen data, is calculated at 0.06.

Lastly, Fig. 7 illustrates the confusion matrix of the classification on the test. As evident in the diagram, the model adeptly predicts the majority of samples across all classes. Nevertheless, checking the confusion matrix of the validation set (not reported here), there are instances where classifications are not unequivocal. For instance, in one case, an image is erroneously predicted as "c" instead of "o," while the remaining forty-two "o" images are accurately classified. Our investigations suggest that this discrepancy may arise from the model encountering challenges in cases of image rotation, leading to occasional misclassifications.

V. DISCUSSIONS AND CONCLUSIONS

In this project, the ASL recognition problem was tackled by developing a CNN model using the widely used off-the-shelf Python libraries of Keras and TensorFlow. The CNN is trained on an updated dataset comprising 15,090 high-resolution images representing 36 ASL classes. The process included data reprocessing, model development, hyper-parameter tuning, and augmentation techniques. The obtained result revealed that the developed model had noticeable accuracy in predicting the proper ASL sign.

At the end of the project, we conducted a comparative analysis between grayscale and RGB images as inputs for our CNN model. Surprisingly, we found that the accuracy of the model remained nearly identical (slightly lower) regardless of whether we used grayscale or RGB images. This led us to a crucial insight—ASL recognition, focusing primarily on boundary detection and finger positions, does not heavily depend on color information. Consequently, opting for grayscale images offers a notable advantage in terms of reduced computational complexity. Grayscale images have a single channel compared to the three channels in RGB images, resulting in faster training times and making the model more suitable for deployment on stand-alone CPUs. The decision to use grayscale is particularly beneficial in scenarios where cost considerations are paramount, as it allows for the utilization of more economical microchips without sacrificing recognition accuracy. This approach proves advantageous by prioritizing key features in ASL gestures while accommodating the limited computational resources of standalone CPU deployments.

Furthermore, during the analysis of images where the developed model exhibited inaccuracies in its predictions, we noted that the resemblance between certain ASL signs, such as '0' and 'o', as well as '2' and 'V', led to inconsistencies in the prediction outputs. To address this issue in future research, it might be beneficial to explore the development of distinct models or employ an ensemble-based technique to enhance prediction accuracy.

REFERENCES

[1] G. Edward Miner, 'Life of Thomas Hopkins Gallaudet, founder of deaf-mute instruction in America," New York, H. Holt and Co., 1888.

[2] [Online] Available at https://www.kaggle.com/datasets/ayuraj/asl-dataset/data

[3] I. Papastratis, C. Chatzikonstantinou, D. Konstantinidis, K. Dimitropoulos, and P. Daras, "Artificial Intelligence Technologies for Sign Language," Sensors, vol. 21, No. 5843, PP. 1-25, 2021.

[4] I.A. Adeyanju, O.O. Bello, M.A. Adegboye, "Machine learning methods for sign language recognition: A critical review and analysis," Intelligent Systems with Applications, Vol. 12, 2021.

[5] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 2012, 25, 1097–1105.

[6] C. Hardyck, L.F. Petrinovich, R.D. Goldman, "Left-handedness and cognitive deficit," Cortex: A Journal Devoted to the Study of the Nervous System and Behavior, vol. 12, no. 3, PP. 266–279, 1976.

[7] [Online] Available at https://www.tensorfl ow.org/api docs/python/tf/keras/layers

[8] [Online] Available at https://keras.io/guides/sequential model