UNIVERSITY
OF SUSSEX

FINAL MASTER'S THESIS

Empowering Sign Language Communication: Image Based Recognition, Domain Specific Language Modeling, And Facilitating Rare Word Expression

Author:

Swati Gupta - 260830

Supervisor:

Dr. Julie Weeds

September 01, 2023

Acknowledgement

I sincerely express my gratitude to my supervisor, Dr. Julie Weeds, for her invaluable expertise in Natural Language Processing and her significant role in shaping the core idea of this dissertation. Her profound knowledge and guidance have been instrumental in providing direction and depth to this research work. I deeply appreciate the time she dedicated and the valuable insights she shared at every stage of the project. Her mentorship has been invaluable, and I am truly grateful for her contributions to its successful completion. Throughout this academic endeavor, her unwavering support and contributions have been immensely valuable.

Finally, I want to express my appreciation to my family and friends for their continuous support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. ABSTRACT

The communication needs of individuals with hearing impairments or speech difficulties are met through the use of Sign Language, a visual means of expression involving gestures, facial expressions, and body language. Sign Language plays an essential and invaluable role in the lives of these individuals, providing them with a powerful tool to effectively convey their messages and interact with others. Nonetheless, communicating with this community using sign language poses significant challenges for non-sign language speakers, as not everyone is proficient in sign language. Moreover, having a sign language translator present at all times is not always feasible for individuals with auditory challenges who wish to convey their messages effectively. While sign languages possess a considerable vocabulary, it may not encompass every word, making effective communication challenging for individuals with hearing and speech impairments. In such instances, when specific signs do not exist, individuals often resort to using the Sign Language Alphabet as a means to convey the desired words. In this research, we will utilize American Sign Language (ASL) due to its widespread usage worldwide, extensive dataset, and existing body of research. The primary objective of this study is to empower individuals with auditory challenges, enabling them to communicate effectively even for words that lack specific signs, thus facilitating seamless interactions with other people. This research endeavors to enhance the overall quality of life for individuals with auditory challenges by leveraging advanced technologies. The application of these technologies extends to various aspects of their life, encompassing everyday communication, medical appointments, or active engagement in medical research pursuits. To address the challenge, we employ artificial neural networks. Initially, we utilize Convolutional Neural Networks (CNNs) to recognize signs from images accurately. Subsequently, to facilitate swift and accurate communication, we incorporate a Bi-Directional Long Short-Term Memory (LSTM) network. This Bi-LSTM model is trained on rare words within the medical domain, which allows it to correct sign characters in case of errors stemming from the image recognition algorithm. Additionally, the LSTM's capabilities extend to autocompleting full words and predicting the most probable subsequent words. The proposed architecture for sign language communication presents a significant improvement in effectiveness and efficiency.

Keywords *– American Sign Language (ASL), Sign Language Detection (SLD); Artificial Neural Network (ANN); Convolutional Neural Network (CNN); Long-Short Term Memory (LSTM); Bi-Directional Long-Short Term Memory (Bi-LSTM); Medical Dataset for Abbreviation Disambiguation for Natural Language Understanding (MeDAL)*

2. INTRODUCTION

Around 430 million human beings worldwide suffer from a disabling hearing insufficiency, and the number is anticipated to increase up to over 700 million by the year 2050, according to the World Health Organization. This indicates that roughly 5% of people worldwide have hearing impairment (*Deafness and Hearing Loss*, n.d.).People who have hearing impairments or are not able to speak use Sign Language to communicate visually using gestures, facial expressions, and body language. Sign Languages (SL) convey meaning through physical movements or a visual manner. There are around 135 official sign languages spoken worldwide, each with its own vocabulary and gestures, showcasing the diverse range of sign languages used in different parts of the world (*How Many American Sign Language Signs Are There? | Homework.Study.Com*, n.d.). British Sign Language (BSL), American Sign Language (ASL), Indian Sign Language (ISL) are some examples of popular sign languages. The limited familiarity of these languages beyond the deaf and speech-impaired communities poses challenges for effective communication with individuals outside of these communities.

Early exposure to sign language has been linked to improved language and cognitive development in deaf children, according to research. For instance, a study by the National Institutes of Health indicated that early sign language instruction benefits deaf children's reading abilities more than it does those who do not. Sign language recognition has gained significant importance due to its ability to foster inclusive communication and enhance accessibility for individuals who cannot speak or have hearing impairments. People who have difficulty speaking or hearing loss can communicate with other people using SL and have access to all of life's necessities, such as education, healthcare, travel, social interactions and employment, thanks in large part to the development of sign language detection (SLD). The essence of effective communication lies in the seamless exchange of thoughts and emotions, transcending barriers, and limitations. People are attempting to come up with novel and fascinating methods for sign language identification considering recent technological and AI breakthroughs. Translation, assistive technologies, sign language identification, transcription, and other uses for SLD are only a few examples. There are various approaches for detecting sign language, including sensor-based, video based, data-based, and deep learning (DL)-based approaches for sequential data, like speech and text. A hybrid approach builds a system for SLD using various Image Recognition, Natural Language Processing, and DL techniques together *(Subburaj and Murugavalli, 2022a).* The linguistic diversity of sign languages and the need for real-time, robust recognition in complex environments are just some of the many obstacles that need to be overcome, but there are also plenty of chances for innovation and impact that can change the lives of those who are unable to hear or speak.

Several recent sign recognition approaches have emerged; however, the problem of achieving effective automatic recognition for sign language systems remains unresolved. Sign language communication presents several challenges, and we aim to address a specific sub-problem within this domain in a constructive manner. British Sign Language consists of approximately 1,800 signs represented through pictures and diagrams(*What Is British Sign Language? - Information about BSL*, n.d.), with each sign accompanied by definitions, explanations, and usage. In contrast, the English language comprises over a million words in total, with approximately 170,000 words in current use(*How Many Words Are in the English Language? | English Live*, n.d.). While we recognize the extensive vocabulary of sign language, with numerous signs corresponding to each word, it is essential to acknowledge that not every word has a designated sign. For instance, rare words and proper nouns, such as people's names, lack specific sign representations. In such cases, sign language letters are utilized to convey the entire word instead of individual signs. However, in practical terms, not everyone possesses knowledge of Sign Language, which can result in slow and tedious communication if individuals have to continually reference a guide for each word. In order to ensure effective sign language communication, particularly in specialized fields such as medical domain, where specific terms like acclimatization, angiomyogenesis, bioluminescence, chemoautotroph, metamorphosis, zygospore, and numerous other words have limited or no corresponding signs, it is crucial to address the challenge of communicating rare words effectively and swiftly. This research aims to tackle this issue specifically within the medical domain by leveraging Image Recognition and Domain-Specific Language Modeling. To accomplish this, we will utilize the American Sign Language Dataset, due to its comprehensive nature and widespread availability for sign language detection purposes. We want to precisely recognize signs from images using Convolutional Neural Networks, resulting in easier and more intuitive for non-sign language speakers to comprehend.

A language model is an artificial intelligence system designed to comprehend and generate human language text. In the context of language modeling, the primary task is predicting the following word in a sentence by analyzing the context provided by preceding words. These models rely on probabilities and language patterns learned from vast amounts of textual data. To illustrate the error correction aspect, consider the word "angiomyogenesis," which has been erroneously spelled as "sngiomyogenesis" due to the mistyped "s". To enable the language model to correct such errors, a training dataset is constructed. In this dataset, each incorrect spelling is paired with its corresponding correct spelling, such as "sngiomyogenesis" mapped to "angiomyogenesis" in this case. By training on this data, the language model learns to recognize and rectify spelling errors. When it encounters a similar mistake in the future, it can apply its learned knowledge to provide the correct spelling, ensuring accurate text generation and understanding.

In the context of domain-specific language modeling, Bi-Directional Long Short-Term Memory networks are integrated to further enhance this communication process, correcting errors (if there are any) in sign characters and facilitating quick word completion for improved interactions. We intend to incorporate the Medical Dataset for Abbreviation Disambiguation for Natural Language Understanding (MeDAL) into our approach. This dataset is used generally for abbreviation disambiguation, designed for natural language understanding pre-training in the medical domain. Our objective is to leverage MeDAL to predict the most probable succeeding words with speed and accuracy. This endeavor is expected to greatly expedite and streamline communication within the medical context, ultimately leading to more effective and efficient exchanges of information.

Our goal is to empower individuals with auditory challenges, granting them the means to communicate effectively even for words that lack specific signs, fostering inclusivity and understanding in diverse social contexts. This study has the potential to radically alter the lives of people with hearing impairments, from routine conversations to important medical consultations or whether it is active participation in medical research. In addition to removing barriers to communication, pursuing this noble goal represents a move toward a society that is more caring and inclusive.

2.1. Objectives

This research paper outlines our objective of constructing a comprehensive system. Initially, our focus is on creating an image recognition system for American Sign Language (ASL) using Convolutional Neural Networks (CNN). The characters recognized by CNN are subsequently fed into a model that not only rectifies any spelling errors but also predicts and completes entire words based on the user's sequential signing. For constructing this predictive and suggestive model, we intend to employ Bidirectional Long Short-Term Memory (Bi-LSTM) which will be trained specifically in terms related to medical domain. We intend to gather terms associated with the medical field by extracting data from well-known websites.

This innovative approach serves to facilitate rapid and accurate communication of words. As the system identifies a handful of words, it initiates predictions for the next most likely word, presenting the top 5 suggestions. For this we will use MeDAL dataset and preprocess it accordingly. Furthermore, we plan to evaluate the effectiveness of our image detection system by developing a program capable of capturing images using a laptop camera. The purpose is to assess the system's viability in real-world scenarios (with a future aim of transitioning to mobile cameras), followed by storing the images in a suitable format. These images will then be processed through the developed CNN model.

1. Establish a domain-specific knowledge base for the language model, enabling efficient word detection. This approach can be expanded in the future to incorporate contextual information and facilitate sentence construction for domain-specific or language-specific modeling.
2. Develop a highly accurate model for detecting Sign Language gestures, aiming for an accuracy of over 90 percent. The accuracy of predicting a word is heavily dependent on the quality of this intermediate output.
3. Investigate the potential impact of Bi-LSTM model on improving the accuracy of complete word prediction and correction occurred by CNN model.
4. Develop a program to capture real-time images from the laptop camera, serving as test samples, to be able to evaluate the practical importance of sign language detection. These images will then be leveraged to assess the effectiveness of the developed image recognition model.
5. Evaluate the effectiveness of using the Bi-LSTM model to refine the predictions made by the image processing component for individual images. The Bi-LSTM model will have access only to the final prediction of the image processing module, which will be in text format.
6. Preprocess MeDAL dataset and assess the effectiveness of using LSTM and Bi-LSTM model for next word prediction.
7. Ultimately, determine the best approach by comparing the time required for each method. This involves evaluating the efficiency of manual signing for individual letters, using the CNN-based approach for sequential character signing, and implementing the hybrid approach that combines both CNN and Bi-LSTM techniques.

To achieve and address the predefined objectives, we shall undertake the following procedures:

- Prepare a dataset through pre-processing techniques, facilitating its conversion to a suitable format for inputting into the Convolutional Neural Network (CNN).
- Build CNN model based on given image to classify images into specific ASL Alphabet.
- Conduct comprehensive analyses and experiments to explore the impact of architectural design of CNN, hyperparameters of the models, aiming to enhance the overall performance measure.
- Prepare dataset using web scraping techniques and collect words which belong to medical domain for domain specific language modelling.
- Preprocess and adapt the web-scraped dataset, ensuring its suitability for tackling the autocorrect and autocomplete problem.

- Build Bi-LSTM based model for correcting errors and predicting a full word in the text passed by CNN.
- Conduct comprehensive analyses and experiments to explore the impact of architectural design of Bi-LSTM, hyperparameters of the models, aiming to enhance the overall performance measure.
- Process the MeDAL dataset, construct a next-word prediction model utilizing a LSTM or Bi-LSTM approach, and subsequently perform hyperparameter tuning to optimize the model's performance.

2.2. Structure of Dissertation

An overview of the dissertation is presented in the following.

Section 1 is the abstract of the dissertation.
Section 2 is the introduction of the dissertation work.
Section 3 this section delves into the background, where we will provide an in-depth exploration of the functioning of CNN and Bi-LSTM models. Additionally, we will discuss various tools and technologies employed in this paper.
Section 4 presents a thorough overview of prior research that has been conducted on the detection of American Sign Language (ASL) using various techniques. It places particular emphasis on research related to the detection of ASL alphabets. Additionally, it encompasses blogs and articles that explore research on spelling error correction. Moreover, the section also includes articles that discuss the utilization of Bidirectional LSTM for next-word prediction.
Section 5 covers the methodology used in this paper; it provides a complete overview of the experimental setup used for this research. This includes a thorough discussion of the model architectures used and the features that were extracted for the experiments.
Section 6 includes the actual results and discussion for each model discussed in Section 5. Each experiment in this chapter builds upon the previous ones, aiming to refine and create the best-fit model.
Section 7 provides a comprehensive overview of the study and presents the conclusions from the research that was completed. In addition, it clarifies paths for possible future research, proposes potential improvements to the current architecture, and identifies areas that could be explored in the future.

3. BACKGROUND

In this section, an in-depth exploration of the theoretical underpinnings behind the employed modeling techniques will be presented. The focus of this research revolves around the systematic development of an efficient and dependable system tailored for accelerated sign language communication. This endeavor encompasses the utilization of a diverse array of methodologies. To be more precise, the data collection phase involves the implementation of Beautiful Soup for the extraction of medical terminologies. Subsequently, Convolutional Neural Networks (CNN) will be harnessed for the recognition of alphabet signs within the framework of American Sign Language. This model will be fine-tuned through the application of the HyperOpt optimization technique. Moreover, a bidirectional Long Short-Term Memory (LSTM) model will be trained to rectify errors and provide suggestions for the most likely complete word the signer intends to convey. The refinement of this model will be conducted via the Grid Search algorithm. Furthermore, an additional LSTM model will be trained to facilitate predictions for subsequent words. Complementing these methodologies, a program will be developed utilizing OpenCV to enable real-time image capture through the laptop camera. Each of these technologies will be meticulously expounded upon, elucidating their respective roles and functionalities within the comprehensive system framework.

3.1. Components of Sign Language

SL is a visual language, as is widely recognized. It is made up of a variety of parts, such as gestures and fingerspelling. Both gestures and finger spelling include signing individual letters to form words. Gestures involve using hand gestures, facial emotions, and body language to convey meaning. British Sign Language (BSL), American Sign Language (ASL), Chinese Sign Language, Brazilian Sign Language, and Australian Sign Language (Auslan) are only a few examples of the numerous varieties of sign language. Each sign language has its own distinct grammar, lexicon, and syntax, which vary depending on geographical and cultural circumstances, according to Harvard's Project on Linguistics and Visual Language. If you want to talk effectively with people who are unable to hear or have hearing loss, you must be able to interpret the fundamentals of this language and its many variations.

SL can be broadly classified into mainly two components: Gesture and Fingerspelling.

3.1.1. Gesture

Depending on the hand, gestures are made up of four fundamental elements: the hand's shape, its movement, its direction, and its placement in relation to the body. There are two main categories of hand motions. Gesture recognition is the process of identifying a kinematic expression utilized for the face, hands, head, and body with various meanings to represent various letters, numbers, or sentences. Static Gestures: Because the hand position does not change during the course of the gesture and because there is not much complexity needed, these gestures are time independent. In sign language, static movements are frequently used to differentiate between spelling fingers. These gestures are displayed as one set of images, with each image represented by a separate frame*(Subburaj & Murugavalli, 2022)*. Dynamic Gestures: Time-dependent hand gestures are referred to as dynamic since they constantly alter in hand position. Although more difficult, this is appropriate for real-time settings where hand movements are tracked in real time rather than relying on frames. once, but in numerous frames. A series of frames is used as the input for data gathering while extracting dynamic characters from video.

3.1.2. Fingerspelling

This method includes making signs of alphabets which is a useful SL learning method for deaf individuals. This method, which is rarely used individually and involves using one of the hands to represent the alphabet, is used to introduce new ideas and concepts. It can also be applied to names that are known to the deaf community but do not have a sign. The alphabet of the fingers makes more sense because each of its twenty-eight letters corresponds to a particular motion made with the hand. The alphabetical method is regarded as one of the most crucial ways to communicate with the people who cannot hear because it is a crucial component of communication in general, especially when discussing words, people’s name, their addresses that do not have established symbols in SL. *(Subburaj & Murugavalli, 2022)*

Having explored the fundamental components of sign language, it is clear that effective communication through visual means plays a crucial role in bridging linguistic gaps. In the context of our research, where we're focused on developing an efficient system for accelerated sign language communication, this understanding becomes pivotal. To achieve our goal, we not only need to comprehend the intricacies of sign language but also leverage modern technologies. One such technology is web scraping, which may seem unrelated at first but holds significant relevance in our pursuit of creating a comprehensive sign language communication system. Let's delve into why web scraping is an essential component of our project.

3.2. Web Scraping

Web scraping is the automated process of extracting data from websites and web pages using software tools or programming scripts. It involves accessing web pages, retrieving relevant information, and storing it for further analysis or use. Web scraping is a common technique used for various applications and data analysis purposes.

One efficient approach in web scraping is to download web pages and perform scraping offline. This offline approach offers advantages like faster data retrieval and reduced network traffic.

To illustrate, in our specific use case, we aim to gather biology-related terms from web pages. To achieve this, we will first download the web page content related to biology terms. Subsequently, we will parse this HTML content to extract the desired data.

For parsing the HTML content, we'll utilize the Beautiful Soup library, a Python tool that simplifies navigation and searching within the HTML structure of web pages. Beautiful Soup makes it easier to extract the information we need by providing user-friendly tools for parsing HTML and XML documents. It streamlines the process of extracting specific information, making it an essential component of our web scraping workflow.

The extracted data will be stored in CSV format, facilitating further data manipulation and analysis as needed for our project.

After learning about web scraping and its application in our research, the next step is to grasp the fundamentals of neural networks: their training process and their role in our project. Neural networks are crucial because they enable us to process the collected data. They can identify and correct spelling errors and suggest the next words, similar to the suggestions you see in Google when typing. However, our case is slightly different. We're taking American Sign Language (ASL) and converting it into English alphabets. Then, from those alphabets, we aim to suggest complete words. To achieve this, we employ a specialized neural network trained on biology-related data, which is a specific language model. Neural networks play a dual role in our research: converting ASL signs into English letters and providing suggestions for complete words and the next word. Therefore, it's essential to have a basic understanding of neural networks, which we discuss in detail in Section 3.3.

3.3. Neural Network

Neural networks are a class of machine learning models inspired by the structure and functioning of the human brain. They consist of interconnected nodes, or "neurons," organized into layers. Information flows through these layers, and each neuron processes its input and passes the result to the next layer. Neural networks are used for various tasks like image recognition, language processing, and decision making.

For example, in image classification, a neural network can learn to distinguish between different objects by analyzing the pixel values in an image. Each neuron might represent a particular feature, like edges or textures, and the network combines these features to make a prediction about the object in the image.

To train a neural network:

- Initialization: Initialize the model's parameters randomly, defining the architecture, number of layers, and number of neurons per layer.
- Forward Propagation: Feed input data through the network to compute predictions.
- Loss Calculation: Calculate a loss function that quantifies the difference between predicted and actual values.
- Backpropagation: Calculate gradients of the loss with respect to the model's parameters using the chain rule.
- Gradient Descent: Adjust the model's parameters in the opposite direction of the gradients to minimize the loss, usually using optimization algorithms like stochastic gradient descent (SGD).
- Iteration: Repeat steps 2-5 over multiple epochs, updating the parameters with each iteration.
- Validation: Evaluate the trained model on a validation dataset to monitor its performance and prevent overfitting.
- Testing: Assess the model's generalization on an unseen test dataset.

3.3.1. CNN

Convolutional Neural Networks (CNNs) represent a specialized class of deep learning algorithms designed primarily for the analysis and processing of images. They find widespread application in computer vision tasks. CNNs find wide-ranging applications in computer vision, including tasks such as Face Recognition, Medical Image Analysis encompassing X-rays, MRIs, and CT scans, and the recognition of American Sign Language (ASL). Moreover, CNNs are employed in Object Detection, where they identify and locate objects within images, and Semantic Segmentation, which involves pixel-level classification for tasks like image segmentation. Comprising three primary layers—convolutional, maxpooling, and fully connected dense layers—CNNs are adept at extracting and learning meaningful patterns from image data. Figure 1 illustrates the configuration of a CNN, sourced from the *(Schematic Diagram of a Basic Convolutional Neural Network (CNN)... | Download Scientific Diagram, n.d.).*

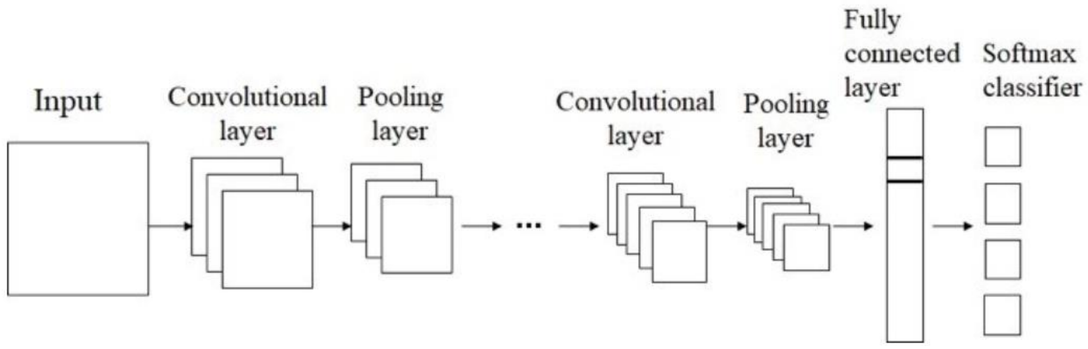


Figure 1: Structure of a CNN
(Schematic Diagram of a Basic Convolutional Neural Network (CNN)... | Download Scientific Diagram, n.d.)

In CNNs, filters play a pivotal role by extracting distinctive features from input data, especially images. These filters, often in the form of small matrices, convolve across the input, performing mathematical operations on overlapping sections. The incorporation of filters empowers the network to grasp intricate shapes, edges, and other essential characteristics inherent in the data. For our specific endeavor, we opt for smaller filter sizes to effectively capture local features like edges and corners. This choice enables the retention of intricate details in the images, ensuring the preservation of fine-grained nuances crucial to the problem at hand. Following each convolutional layer, there is an integration of an activation function and a pooling layer. In this research endeavor, we shall leverage well-regarded activation functions such as ReLU, SELU, ELU, and Tanh. The incorporation of these functions introduces non-linearity into the network, which in turn empowers the model to assimilate intricate data relationships. This augmented adaptability equips the network to navigate a diverse spectrum of tasks with efficacy. Max pooling, a stratagem intrinsic to Convolutional Neural Networks (CNNs), operates as a mechanism to systematically reduce the dimensions of feature maps through down sampling. This reduction in spatial dimensions is achieved while upholding the salient features of the data. Typically executed post-convolutional layers, this technique progressively diminishes the spatial representation, culminating in the optimization of parameter counts within the network. In Convolutional Neural Networks (CNNs), the flatten layer serves to convert the multi-dimensional feature maps originating from convolutional and pooling layers into a singular one-dimensional vector. This conversion assumes pivotal importance as it establishes a bridge between the network's feature extraction strata and the fully connected dense layers, which execute tasks like classification. These dense layers effectively capture intricate feature relationships, culminating in final decisions. Dropout layers are tactically employed to curb overfitting, fortifying the model's resilience. The ultimate classification of ASL labels hinges on a dense layer with softmax activation. CNNs exhibit consistent excellence in ASL recognition endeavors. They proficiently accommodate the inherent variability in hand shapes, orientations, and positions that typify signing. This adaptability emerges from their aptitude to discern salient patterns across diverse image regions, rendering them impervious to spatial transformations. Notably, CNNs glean distinctive features directly from raw ASL image pixel values, obviating the need for predefined attributes. This intrinsic ability empowers CNNs to accurately decode and categorize ASL gestures, irrespective of their spatial placement within the image.

3.3.2. LSTM

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed for sequential data processing, such as time series or natural language. However, standard RNNs faced the vanishing gradient problem, hindering their ability to capture long-range dependencies due to gradient diminishing during training. Long Short-Term Memory (LSTM) networks were introduced to address this issue.

The architecture of a Long Short-Term Memory (LSTM), as depicted in Figure 2 and cited from the work of *(Greff et al., 2017)* ,belongs to a class of recurrent neural networks engineered to address the vanishing gradient challenge while effectively capturing extended dependencies within sequential data.

It comprises three crucial gates:

- Forget Gate: Determines which information from the previous cell state should be discarded, aiding in removing irrelevant context.
- Input Gate: Decides which new information should be added to the cell state, aiding in updating relevant context.
- Output Gate: Determines the output of the cell state, considering the modified cell state and current input, which captures relevant information for the current prediction.

LSTM's gated structure enables it to effectively model complex patterns and relationships, making it particularly useful in Natural Language Processing (NLP) tasks like sequence-to-sequence translation. In such tasks, LSTMs can capture contextual nuances, handle varying input and output lengths, and generate coherent translations by maintaining a richer context throughout the sequence.

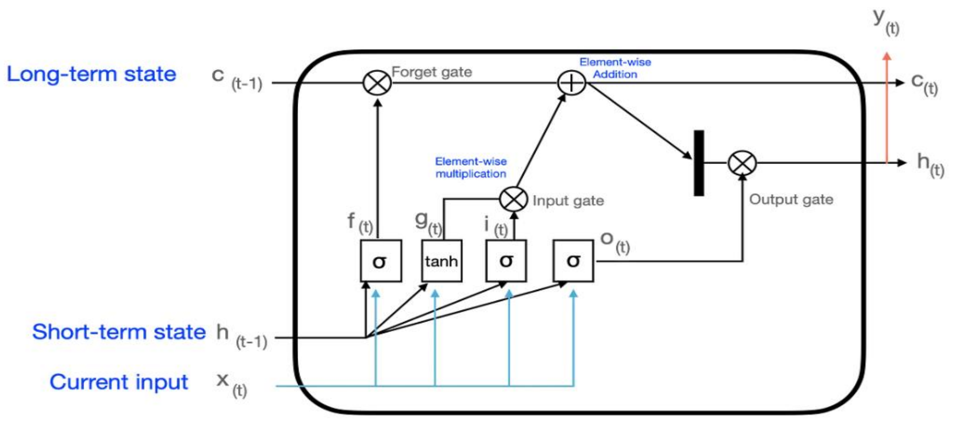


Figure 2: Structure of a LSTM cell

(*Greff et al., 2017)*

3.3.3. Bi-LSTM

The Bidirectional LSTM (Bi-LSTM), as presented in Figure 3 and sourced from the work *(The Unfolded Architecture of Bidirectional LSTM (Bi-LSTM) with Three... | Download Scientific Diagram, n.d.)*, represents an augmentation of the conventional Bi-LSTM framework. It operates by analyzing input sequences in both the forward and backward directions, enhancing its capacity for sequence modeling. This enables it to capture information from past and future context simultaneously, enhancing its ability to capture complex relationships in sequential data. Bi-LSTM consists of two parallel LSTM layers: one processes the sequence forward, while the other processes it backward. The outputs from both directions are then combined, capturing information from the entire sequence. This architecture enables Bi-LSTM to leverage past and future context for improved sequence analysis and prediction. Bi-LSTM is advantageous for tasks like spell correction and next-word prediction due to its bidirectional processing. In spell correction, it can consider the context before and after the potentially misspelled word, helping to identify the correct spelling by considering surrounding context. For next-word prediction, Bi-LSTM leverages both preceding and succeeding words, improving the model's ability to generate contextually coherent predictions and handle diverse language patterns. This bidirectional approach enhances the model's context understanding and aids in making more accurate corrections or predictions in natural language sequences.

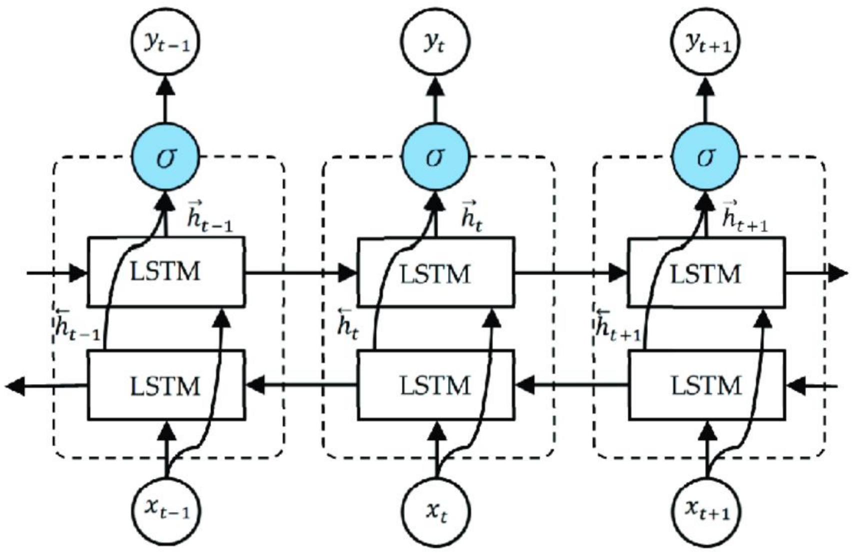


Figure 3: Structure of a Bi-LSTM cell

(The Unfolded Architecture of Bidirectional LSTM (Bi-LSTM) with Three... | Download Scientific Diagram, n.d.)

Following our comprehensive exploration of neural networks and their applications, let's now shift our focus to the datasets central to our research. Our objective is to train an image detection system for recognizing sign language, utilizing the American Sign Language Dataset. Additionally, for context-specific biology terminology, we require relevant words, which we will extract from the Medical Dataset for Abbreviation Disambiguation for Natural Language Understanding (MeDAL). To gain a foundational understanding of these datasets, let's delve into their key characteristics in Section 3.4.

3.4. Dataset

In this section, we will delve into the datasets utilized in this research paper, namely the American Sign Language (ASL) dataset and the Medical Abbreviation Disambiguation for Natural Language Understanding (MEDAL) dataset.

3.4.1. American Sign Language Dataset

The most common sign language in North America, Canada, West Africa, and Southeast Asia is American Sign Language (ASL), which is acknowledged as a natural language with a unique grammar and syntax. In the US, between 250,000 and 500,000 people identify ASL as their first language. ASL is based on French Sign Language but also draws from various regional sign languages, including Martha's Vineyard Sign Language. ASL utilizes a one-handed fingerspelling alphabet like French Sign Language. The dataset comprises a compilation of American Sign Language alphabet images, organized into 29 folders representing distinct classes. The training dataset consists of 87,000 images, each sized 200x200 pixels. *(Sign*

Language MNIST | Kaggle, n.d.).
Figure 4 features various examples of American Sign Language alphabet symbols, with the figure sourced from *(Sign Language MNIST | Kaggle, n.d.).* Among the 29 classes, 26 pertain to the letters A-Z, while the remaining 3 classes are specifically assigned to SPACE, DELETE, and NOTHING. These 3 classes play a crucial role in real-time applications and classification tasks. As for the test dataset, it contains a limited set of 29 images, designed to encourage the utilization of real-world test images.

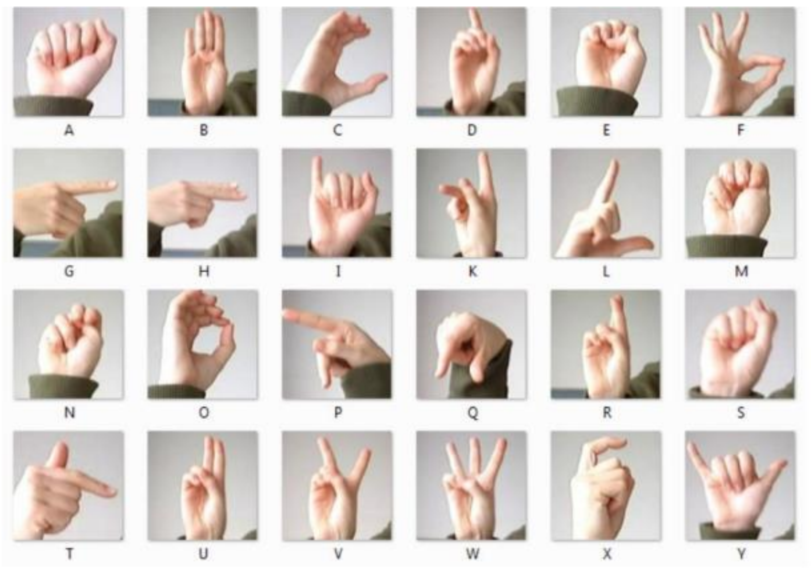


Figure 4: American Sign Language
(Sign Language MNIST | Kaggle, n.d.)

3.4.2. MeDAL

Medical Dataset for Abbreviation Disambiguation for Natural Language Understanding (MeDAL) is a large medical text dataset curated for abbreviation disambiguation, designed for natural language understanding pre-training in the medical domain. It was published at the Clinical NLP workshop at EMNLP. Figure 5 provides an illustration of a sample from the MeDAL dataset, sourced from the research paper by (*Wen et al., 2020*), where the true meaning of the abbreviation 'DHF' is inferred from its context. The MeDAL dataset consists of 14,393,619 articles and on average 3 abbreviations per article. The MeDAL dataset is created from PubMed abstracts which are released in the 2019 annual baseline.

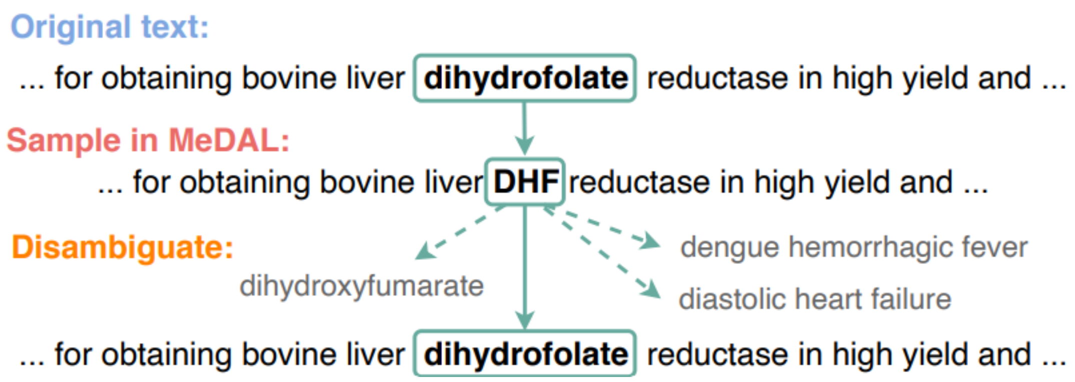

Original text:
... for obtaining bovine liver dihydrofolate reductase in high yield and ...
Sample in MeDAL:
... for obtaining bovine liver DHF reductase in high yield and ...
Disambiguate:
dihydroxyfumarate
dengue hemorrhagic fever
diastolic heart failure
... for obtaining bovine liver dihydrofolate reductase in high yield and ...

Figure 5: A sample in the MeDAL dataset
(Wen et al., 2020)

Now that we have gained insight into our datasets, the next crucial step is to explore the process of selecting the optimal model from our predefined architecture. This step is vital for achieving high-quality results. In Section 3.5, we will delve into the hyperparameters employed in this research, and in Section 3.6, we will discuss the techniques we use for hyperparameter tuning.

3.5. Hyper Parameters

Hyperparameters are parameters that are set before the learning process of a machine learning model begins. They are not learned from the data but rather specified by the user or the model designer. These parameters influence the behavior and performance of the model during training and can significantly impact its ability to learn and generalize from the data. Hyperparameters are distinct from the model's trainable parameters. Trainable parameters are learned from the data through optimization, while hyperparameters are set by the user and affect how the optimization process takes place. A few examples of the hyperparameters are number of layers in the neural network, number of neurons in each layer, number of epochs, activation function, dropout rate, batch size and learning rate. In CNN we can also define the size of kernel and maxpooling and flatten layer. We will discuss some hyperparameters which we have used in this paper for tuning our model.

3.5.1. Network Architecture Hyperparameters

These include the number of layers in a neural network, the number of units or neurons in each layer, the activation functions used, etc. These choices influence the complexity and representational capacity of the model.

3.5.2. Activation Functions

The activation functions are at the very core of Deep Learning. They determine the output of a model, its accuracy, and computational efficiency. In some cases, activation functions have a major effect on the model’s ability to converge and the convergence speed. Activation Functions are used to eliminate linear relationship between inputs and outputs in neural networks. The sigmoid function had the main problem of vanishing gradient problem. The Tanh activation function is similar to the sigmoid but maps inputs to the range [-1, 1]. It has a center at zero, which means its outputs are zero-centered, making it useful for certain types of data. However, Tanh still suffers from the vanishing gradient problem for large inputs. The ReLU activation function overcomes the vanishing gradient problem by only saturating for negative inputs and allowing positive inputs to pass through unchanged. ELU is an extension of ReLU that aims to address the "dying ReLU" problem. For negative inputs, ELU provides a non-zero gradient, preventing neurons from becoming inactive. SELU is a variation of ELU that aims to achieve self-normalization of hidden units within the network. It has specific scaling properties that ensure that activations tend to have a mean close to zero and a standard deviation close to one, helping in stabilizing training. Figure 6 below depicts the Activation Functions utilized in this paper along with their corresponding equations.

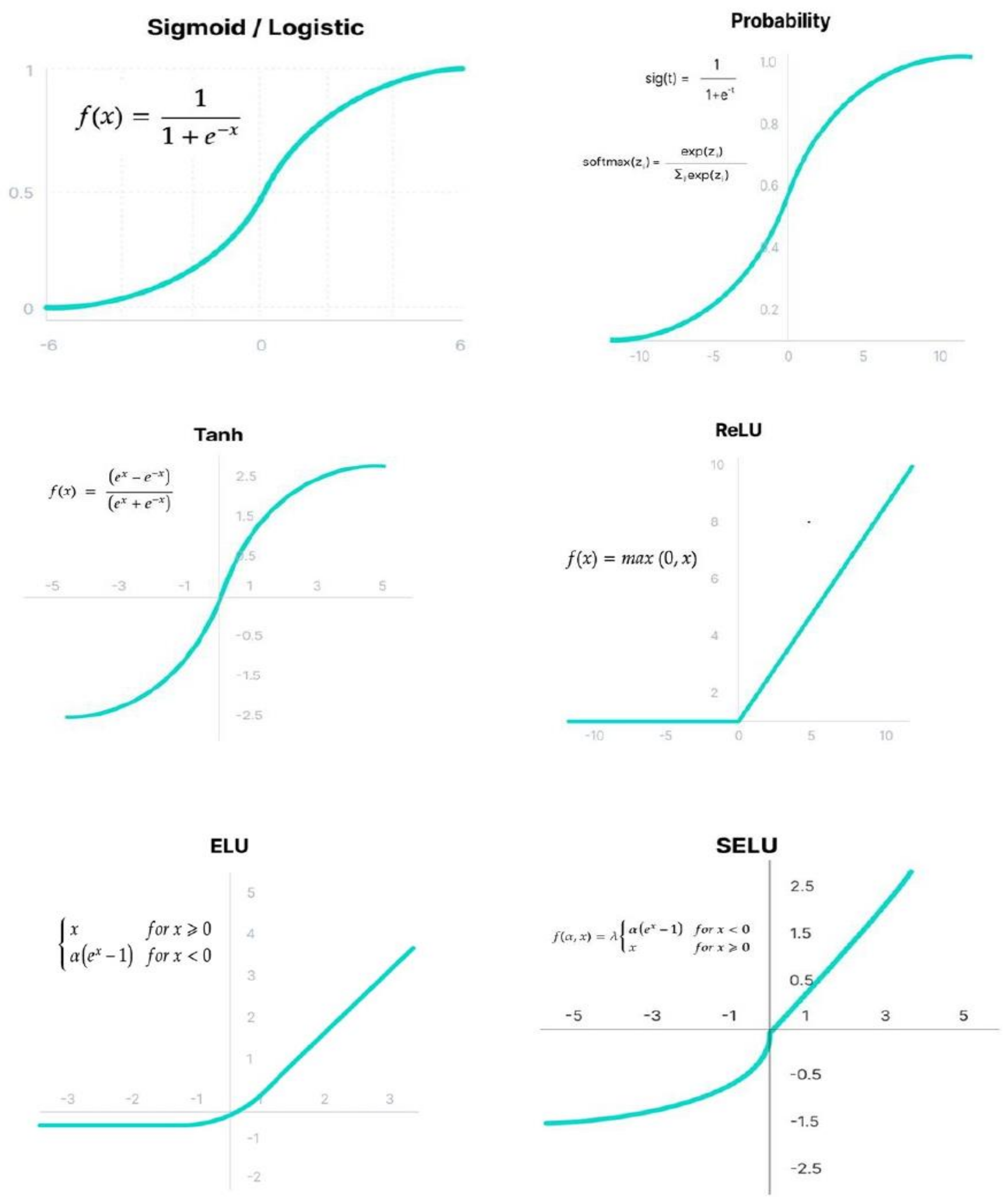


Figure 6: Activation Functions used in this paper with equations

3.6. Hyperparameter Optimization Techniques

Hyperparameter optimization techniques are crucial for enhancing machine learning model performance. By systematically searching through different parameter combinations, these methods help identify optimal settings, boosting model accuracy and generalization. They prevent manual guesswork, saving time and resources, and contribute to robust, well-tuned models that excel across various datasets and tasks. Grid Search, Random Search, Bayesian Optimization are some of the popular hyperparameter optimization techniques. We will discuss the two techniques we have used for this research.

3.6.1. Bayesian Optimization

Bayesian Optimization is an efficient hyperparameter tuning technique. It models the complex relationship between hyperparameters and model performance using probabilistic models. By iteratively updating these models based on observed results, it intelligently selects the next hyperparameter set to evaluate, focusing on promising regions of the search space. This approach balances exploration (trying new configurations) and exploitation (leveraging known information) to quickly converge to optimal or near-optimal settings. Bayesian Optimization is particularly useful for expensive-to-evaluate functions, like training deep neural networks, as it minimizes the number of evaluations needed to find optimal hyperparameters.

3.6.2. Grid Search Algorithm

Grid search is a hyperparameter optimization technique. It involves defining a grid of possible hyperparameter values and exhaustively evaluating all combinations. By systematically trying various parameter settings, grid search searches through the entire predefined space to find the best-performing configuration. While it's easy to implement and interpret, it can be computationally expensive when dealing with a large number of hyperparameters or wide ranges of values. Despite its limitations in efficiency, grid search serves as a baseline approach for hyperparameter tuning and is valuable for identifying optimal settings in smaller search spaces.

Now that we have a basic understanding of hyperparameters and how to adjust them, there's one more important topic to cover—OpenCV. Once we've finished building our system, it's crucial to thoroughly test it with real-time images. OpenCV is a Python library that allows us to capture images in real-time. Moreover, we will evaluate the performance of our ASL Alphabet recognition system by using images we capture in real-time and carefully analyzing the system's output.

3.7. Open CV

OpenCV, known as Open-Source Computer Vision Library, is a widely adopted open-source software library primarily employed for computer vision and image processing tasks. It encompasses a comprehensive collection of functions and algorithms that empower developers to perform a multitude of operations on images and videos. These operations include object detection, image recognition, motion tracking, and more. In our scenario, we will utilize OpenCV to generate test images for American Sign Language alphabets. OpenCV will be instrumental in processing these images and then we will pass these images to CNN model for sign language detection. We can use OpenCV and CNN to actively test our system in real-time.

4. RELATED WORK

This research primarily concentrates on addressing a subproblem within the Sign Language domain by constructing a system using a hybrid approach. Interestingly, there is a lack of research that directly aligns with this particular approach. Therefore, we will explore existing work related to the developed models but only in the context of their specific roles within the overall pipeline approach.

Numerous approaches exist for recognizing American Sign Language. Vision-based techniques involve tracking hand and body movements using cameras to capture the signer's gestures. Hybrid approaches combine data-driven and vision-based methods for enhanced accuracy and robustness. These hybrid methods may use computer vision algorithms to track hand motions and employ language models like BERT and GPT to predict sentences, convert to speech, or transcribe input *(Aloysius & Geetha, 2020).*

To discuss research work specifically related to the recognition of ASL alphabets. In their research paper *(Dong et al., 2015)* developed a contemporary approach to ASL alphabet recognition has been introduced. This approach utilizes an affordable depth camera known as Microsoft's Kinect. It achieved an accuracy of approximately 90% in recognizing 24 static ASL alphabet signs. In a different study by *(Das et al., 2020)*, they devised a system for automatic recognition of hand sign-based digits and oral presentation of the outcomes in the Bengali language. This system's foundation is laid upon a deep learning model employing Convolutional Neural Networks (CNNs), which achieved an accuracy level of around 92%. In reference (Rao et al., 2018), an approach based on CNN has been introduced for the recognition of Indian Sign Language gestures. Their model achieved an impressive accuracy rate of 92.88%. In their work, *(Bheda and Radpour, n.d.)* introduced a technique for classifying alphanumeric characters in American Sign Language (ASL). They utilized two datasets, one they generated themselves and the MU HandImages ASL dataset *((PDF) A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures, n.d.)*. Their findings indicated average recognition rates of 67% for alphabet characters and 82.5% for ASL digits.

When considering research on the correction of spelling errors in English words, we can refer to the study conducted by (Li et al., 2018). In their study, they introduce a nested recurrent neural network (nested RNN) model designed for correcting English spelling errors. To train this model, they generate pseudo data based on phonetic similarities. This model effectively combines orthographic information and context, training seamlessly from start to finish. Unlike traditional methods, it eliminates the need for feature engineering and does not depend on a noisy channel model. Experimental results demonstrate the effectiveness of their approach in spelling error correction. Specifically, their nested RNN structure achieves a precision of 71.77 and an F1 score of 69.39. In another study done by *(Ltrc et al., n.d.)* the authors explore the development of an automatic spelling correction system tailored for languages with limited linguistic resources. Their approach involves the utilization of a sequence-to-sequence deep learning model that is trained end-to-end. To evaluate their model, they conduct experiments using synthetic datasets specifically created for Indian languages, namely Hindi and Telugu. These datasets incorporate character-level spelling errors. To tackle the spelling correction challenge in Indic languages, they employ a dedicated corrector network as an encoder and an implicit language model as a decoder within a sequence-to-sequence attention model, trained comprehensively end-to-end. The outcomes of their study indicate an accuracy rate of 0.85 for Hindi and 0.89 for Telugu language spelling correction.

The spelling correction needed for this paper differs significantly from traditional requirements. In this case, the errors do not arise from typing English words; instead, they occur when the CNN model incorrectly detects alphabet letters during sign language recognition. Consequently, conventional error correction approaches for English language text cannot be directly applied. To address this unique challenge, we adopted a similar approach. We generated a dataset that includes errors and mapped them to their correct spellings. However, these errors were introduced based on instances where our developed CNN model made mistakes. We analyzed the confusion matrix to determine these specific error patterns. The main aim here is to correct the errors while detecting the ASL Alphabets from our developed CNN approach.

For next word prediction, we will review *(Next Word Prediction Model | Aman Kharwal, n.d.)* article which serves the similar purpose which we are trying to solve The author generates the data by tokenizing the text into individual words, the text used is from a literary works. Word Length represents the number of previous words that will be used to predict the next word. The author creates two arrays, one for storing the features (x) and another for storing the corresponding labels (y). These arrays are populated based on the specified Word Length, where a sequence of previous words is matched with the next word in the dataset. They choose to use a Recurrent Neural Network (RNN), specifically a Long Short-Term Memory (LSTM) model. During training, the model learns the patterns and relationships between previous words and the next word. In the article *(Next Word Prediction Using LSTMs. Detailed Implementation Of... | by Prerana | Geek Culture | Medium, n.d.),* the author implements a word-based model, starting by tokenizing the data and preprocessing it to convert it into numerical form. Subsequently, the author employs an embedding layer, LSTM layer, and encoded data in conjunction with dense layers to construct and train a model, achieving an accuracy of approximately 0.56.

5. METHODOLOGY

In this section, we will outline the steps and techniques that were essential for the success of our experiments. We'll start by discussing the environment in which our experiments were conducted. We'll then provide a detailed explanation of our proposed approach, including how we collected data and the methods we used. We will also cover the structure of our model, how we trained it, and the performance metrics we used for evaluation.

5.1. Experimental Environment

The experiments were conducted on a Windows 10 computer equipped with 64 GB of RAM and an NVIDIA RTX A4000 GPU. The programming language employed was Python 3.10, known for its user-friendly nature and compatibility with various integrated libraries. Key Python libraries such as Seaborn, Matplotlib, Pandas, NumPy, and TensorFlow (latest version, specified in the code's requirements file) were utilized for image classification, autocorrection of words, and next-word prediction tasks. The project's hardware environment was a DELL-U2722D monitor. The hardware was powered by a 12th Generation 2.40 GHz Intel Core i9 Central Processing Unit boasting 16 cores and 24 logical processors, accompanied by 64 GB of RAM.

5.2. Proposed Approach

In the proposed methodology, our objective is to develop an efficient communication system designed to individuals with auditory or speech impairments. The framework, as illustrated in Figure 7, entails the utilization of a Block Diagram for facilitating the interpretation of rare biology terms into American Sign Language.

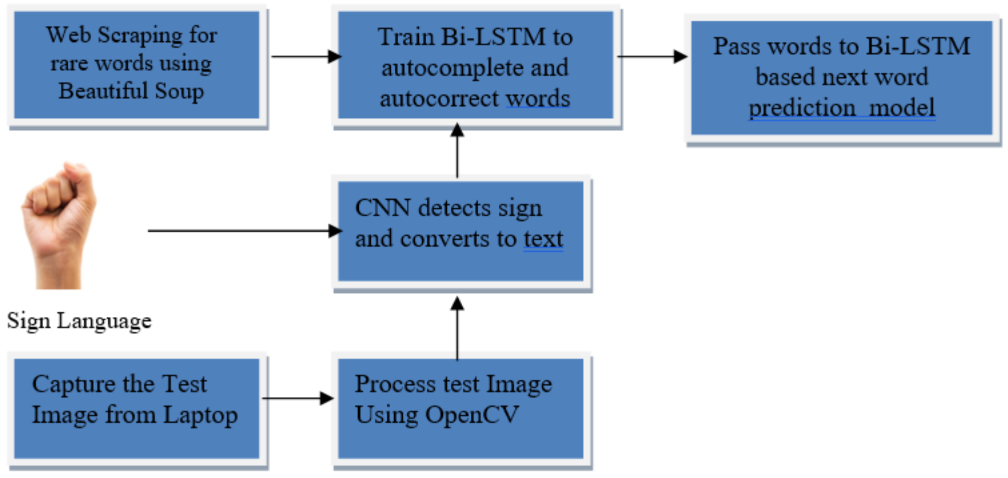
Initially, a data acquisition process will involve extracting relevant vocabulary pertaining to biology from diverse online sources. The accumulated data will be structured into a suitable CSV format for subsequent processing. Subsequently, a specialized Bi-directional Long Short-Term Memory (Bi-LSTM) model will be constructed and trained on this corpus of words.

Parallelly, a Convolutional Neural Network (CNN) will be designed with a focus on discerning American Sign Language (ASL) alphabets. This CNN will be trained exclusively on ASL alphabets, yielding individual alphabetic character detections. These character detections constitute an intermediate outcome. As a signer performs sign language, the characters identified by the CNN will be conveyed to the Bi-LSTM model. This interaction serves a dual purpose: expediting the conversion of the signed letters into biological terms and rectifying any inaccuracies that may have arisen during the initial character recognition by CNN.

Upon accumulating a minimum of 4-5 interpreted words, the subsequent step involves employing a word prediction model. This model, meticulously trained on the MeDAL dataset, demonstrates alignment between its labels and the words gathered from the web scraping phase.

To facilitate testing, a program will be developed for capturing ASL test images. These images will be subsequently processed using OpenCV to confirm to the requisite format. The efficacy of the entire system will then be assessed in real-time scenarios, with real-world images fed into the CNN model previously established.

In conclusion, the proposed approach seeks to realize an accelerated communication mechanism for individuals with communication challenges, achieved through a fusion of neural network models and data-driven methodologies.



Web Scraping for rare words using Beautiful Soup
Train Bi-LSTM to autocomplete and autocorrect words
Pass words to Bi-LSTM based next word prediction model
CNN detects sign and converts to text
Sign Language
Capture the Test Image from Laptop
Process test Image Using OpenCV

Figure 7: Block Diagram for rare biology terms communication in American Sign Language

5.3. Web Scraping Data for Collecting Biology Terms

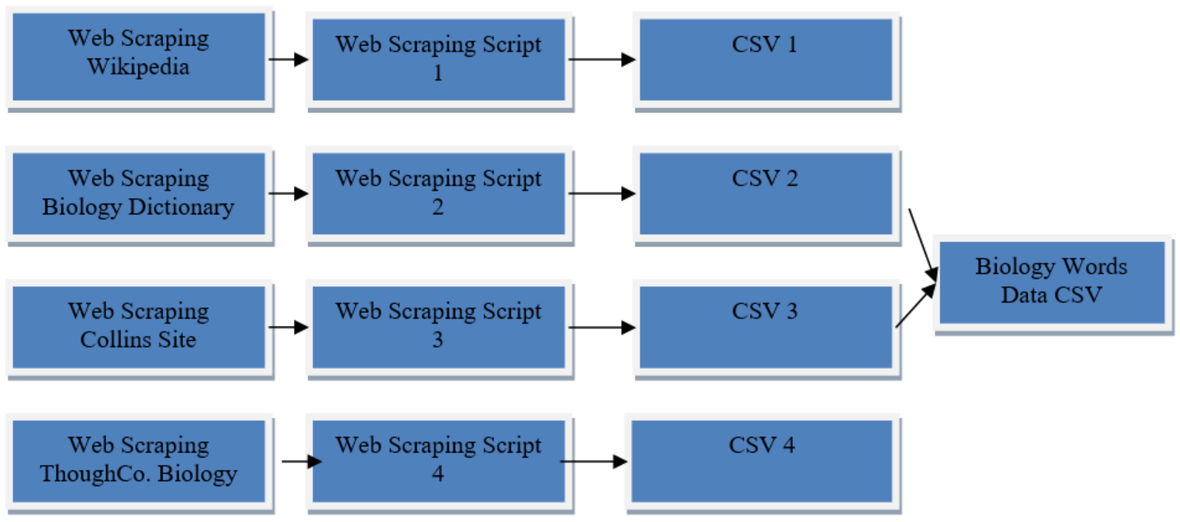


Figure 8: Block Diagram for Web Scraping Data for Biology Terms

In the pursuit of collecting rare biology terms, we conducted a comprehensive data acquisition process from four distinct online sources. These sources encompassed reputable platforms such as Wikipedia, Biology Dictionary Collins website, Understanding difficult biology words, and ThoughtCo. This multipronged approach was necessitated by the diverse formatting of webpages across these sites, compelling us to employ distinct scraping methodologies. To ensure uniformity, we adopted the HTML parsing technique for all sources.

In the context of the Glossary Biology words from Wikipedia, our approach involved the systematic iteration through all "dfn" tags within the parsed HTML document, represented by the 'soup' object. By extracting the textual content from each of these tags, we assembled the data into a list structure referred to as "data." Ultimately, the outcome of this endeavor manifested as a list of lists, with each inner list encapsulating the textual content of an individual "dfn" tag from the HTML document.

Similarly, on the Biology Dictionary Collins website, our method involved traversing through all HTML elements tagged with the CSS class "ref" within a parsed HTML document, employing the 'soup' object. In parallel, for Understanding difficult biology words, we adopted a comparable strategy, iterating through HTML elements with the CSS class "mntl-sc-block-subheading__text."

Analogously, in the context of ThoughtCo. Biology, we extended this methodology to cover HTML elements identified by the CSS class "wpg-list-item-title." Subsequently, the extracted data from each of these sources was aggregated into distinct lists. These individual datasets were then meticulously stored in separate CSV files.

Concluding this phase, the culmination of efforts was marked by the amalgamation of all CSV files into a single comprehensive CSV file. This consolidated repository was designated as the "Biology Words Data CSV." The comprehensive process employed for scraping biology-related words is visualized in Figure 8 through a Block Diagram illustrating the web scraping of data for biology terms. This consolidated dataset plays a pivotal role in our subsequent stages, particularly in the implementation of our Bi-directional Long Short-Term Memory (Bi-LSTM) model, facilitating both autocorrection and autocompletion functionalities for biology-related words.

5.4. Model Architecture of CNN for American Sign Language Detection

In the process of constructing the Convolutional Neural Network (CNN) model, we adhered to a systematic approach validated by this research initiative. Our initial step involved subjecting the methodology to a preliminary assessment using a reduced subset of data. This allowed us to ensure the viability of the approach before full-scale implementation.

To facilitate the assembly of the requisite image paths for training the CNN model, we introduced a function named "create_image_paths." This function encompasses a parameter called "class size," which governs the quantity of images designated for a specific class within the American Sign Language (ASL) alphabet. The function iterates through the entirety of the dataset employing the "glob" and "os" libraries. Through this iterative process, we selectively select images in a randomized manner, thereby ensuring a diversified pool of images. The resulting image paths are aggregated in a list, accompanied by corresponding labels.

The function "create_image_paths" is subsequently invoked by "create_subset_of_image_data." This latter function is responsible for generating paths for all the ASL alphabet labels. The outcome is structured in a dataframe format, encapsulating the paths and their corresponding labels.

To enable visualization and quality assurance, we devised the "sample_images" function. This function establishes a subplot mechanism to exhibit diverse image types present within our dataset. It accomplishes this task by plotting ten labels at a time. The visual representation of American Sign Language Dataset images is presented in Figures 9, 10, and 11. The subplot visualization for alphabet labels was essential for assessing the nature of images encompassed within the dataset.

Evidently, these figures underscore the substantial diversity present within the dataset. Varied lighting conditions, different individuals engaging in signing, images captured from various positions, and a range of skin compositions collectively contribute to a robust and comprehensive dataset primed for the training phase.

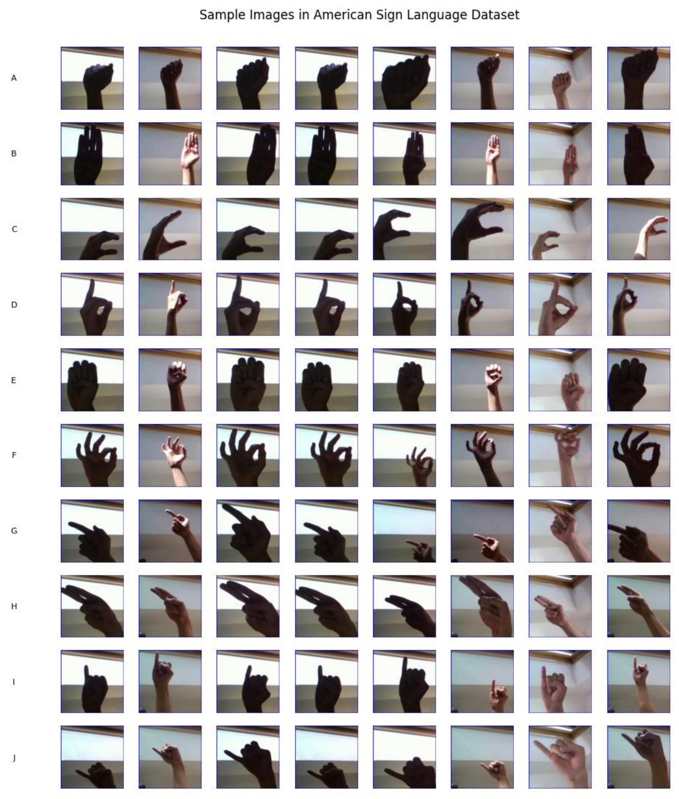


Figure 9: Extracted Images of ASL Alphabets from A to J

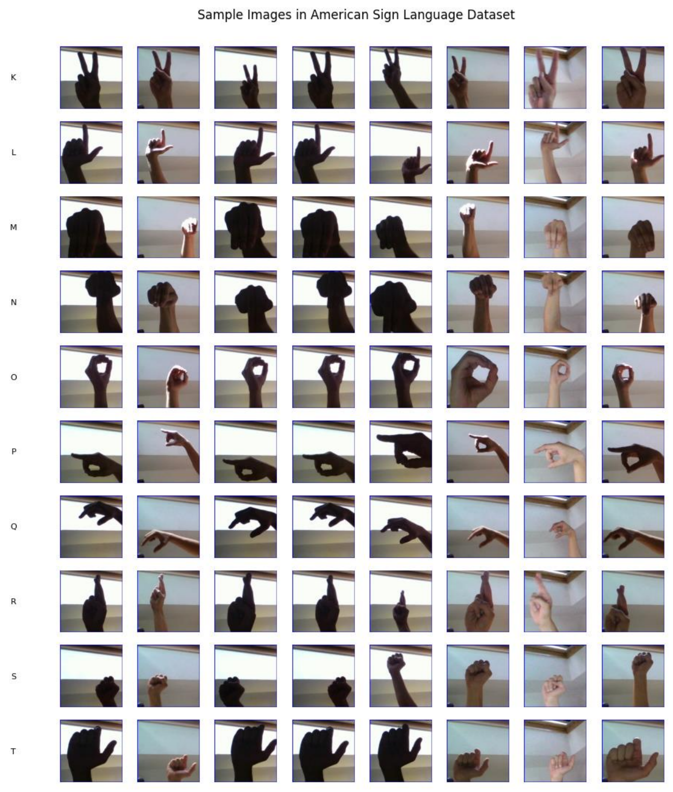


Figure 10: Extracted Images of ASL Alphabets from K to T

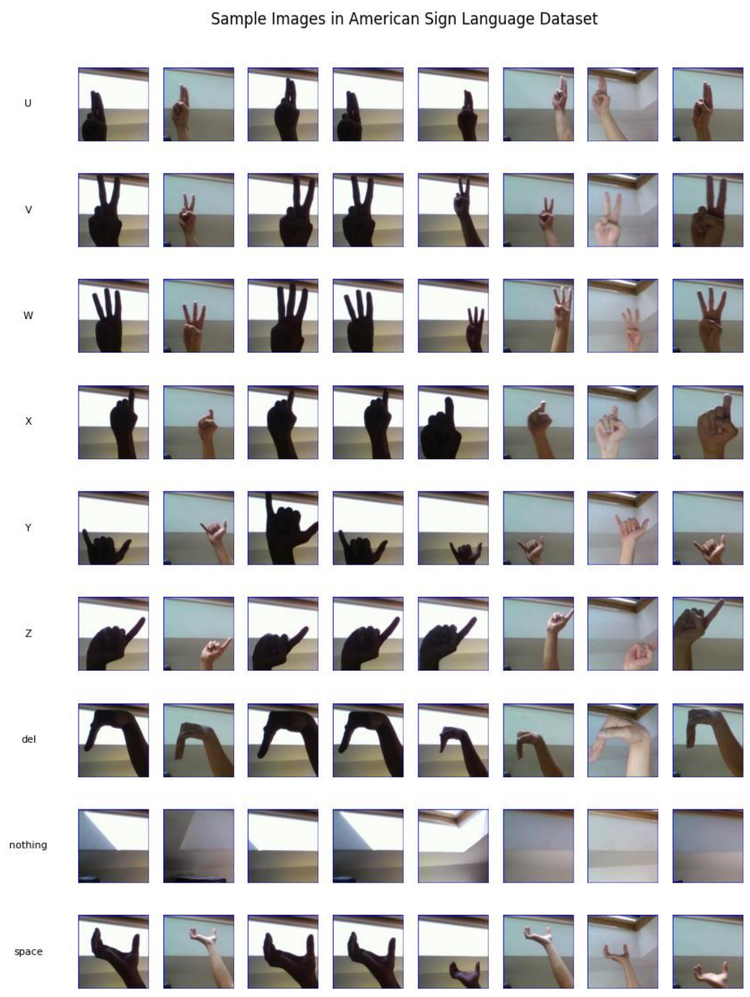


Figure 11: Extracted Images of ASL Alphabets from U to Space

5.4.1. Data Augmentation for CNN

In our research methodology, the "data_augmentation" function plays a pivotal role in preprocessing and enhancing the image data to fortify the efficacy of our machine learning models, particularly Convolutional Neural Networks (CNNs). This function takes a subset DataFrame containing image paths and their corresponding labels, along with parameters for training length and target image size. It processes the images by reading and resizing them to the desired dimensions, subsequently encoding the class labels using the Label Encoder technique. The resulting preprocessed image data and encoded labels are returned, along with a dictionary that maps encoded labels to their original class labels. This meticulous augmentation ensures that our model trains on a diverse and representative dataset, effectively improving its ability to recognize and generalize patterns in the American Sign Language alphabet.

Subsequently, we partitioned the dataset into a 7:3 ratio, allocating 70% for training and 30% for testing purposes. To facilitate effective processing, we transformed the class labels into categorical format, enabling seamless comprehension by the machine learning algorithms.

5.4.2. Hyperparameter Tuning with Bayesian Optimization for CNN Architecture

In this pivotal phase of our research, we delve into the intricate task of hyperparameter tuning to forge an optimized architecture for our Convolutional Neural Network (CNN) model. Our primary goal in this pursuit is to harness the model's predictive capabilities to their utmost potential by methodically exploring and refining crucial architectural components. I will provide a detailed account of our methodology below.

“build_model" function is tailored to encapsulate the essence of our CNN architecture. Within this construct, we purposefully introduce tunable hyperparameters that wield considerable influence over the network's composition. While the convolutional and pooling layers remain constants, we engage in a granular exploration of distinct architectural facets. Specifically, we are concentrating on the number of neurons in specific layers and the activation functions that are present throughout the network. Notably, the second convolutional layer's neuron count, the neuron population within the subsequent dense layers, and the activation functions applied throughout the model are at the crux of our investigation.

For instance, we consider scenarios where the second convolutional layer could exhibit varying numbers of neurons—such as 64, 128, and 256—each embodying distinct capacity for feature extraction. In the realm of dense layers, we explore a range of configurations for neurons, such as 256, 512, and 1024 neurons in the second dense layer, and 256, 1024, and 2048 neurons in the third dense layer. We also conduct trials involving different activation functions, including ReLU, ELU and Tanh. These permutations dictate the network's capacity to encapsulate intricacies and patterns within the data.

In this phase of our research, we employ Bayesian Optimization, a well-established approach recognized for its adeptness in navigating diverse hyperparameter configurations while minimizing validation loss. This exploration is complemented by the strategic introduction of the Early Stopping callback, designed to safeguard the model from overfitting to the training data. This measure enhances the model's capacity to generalize effectively to new, unseen data. Simultaneously, we harness the Bayesian Optimization technique to systematically traverse the hyperparameter space of our model architecture. Our objective is to pinpoint hyperparameters that lead to minimal validation loss, ultimately elevating the model's predictive accuracy. To ensure efficiency, we restrict the exploration to a maximum of 10 trials, accompanied by the prudent practice of overwriting previous search results. By intertwining these methodologies, we refine the model's configuration, striving to attain optimal performance through strategic hyperparameter tuning.

In summary, our carefully guided exploration, supported by strong evidence and thoughtful adjustment of hyperparameters, leads us to a setup that maximizes the model's effectiveness. By choosing the right building blocks for the architecture and exploring various hyperparameter options, we aim to enhance the CNN model's ability to understand and generalize intricate patterns present in the American Sign Language dataset. We save the final results of trials in “trial_results.csv”.

5.4.3. Final Architecture of CNN Model

The Final architecture of the model is based on the best hyperparameters we got after hyperparameter tuning using Bayesian Optimization which is described in Section 5.4.2. In this section, a Sequential model is constructed to implement a Convolutional Neural Network (CNN) architecture, tailored for the classification of American Sign Language (ASL) alphabet images. The architecture is composed of two pivotal segments: feature extraction and classification.

The feature extraction phase begins with a Convolutional Layer incorporating 32 filters, each with a spatial extent of 5x5. This layer processes images of dimensions 64x64x3 (width, height, and three-color channels) to capture crucial patterns and features. An Activation Layer follows, applying the Rectified Linear Unit (ReLU) function to introduce non-linearity. Subsequently, a MaxPooling Layer with a pool size of 2x2 condenses the spatial dimensions of the data, extracting prominent features.

Building on this, an additional Convolutional Layer is introduced, this time equipped with 256 filters sized 3x3, further enhancing feature extraction. Another Activation Layer and MaxPooling Layer are employed, mirroring the previous structure.

Moving to the classification phase, the architecture transitions to a flattened format through a Flatten Layer, preparing the data for fully connected layers. The subsequent sequence features multiple Dense Layers. The first of these layers contains 256 units and employs the ReLU activation function. To mitigate overfitting, a Dropout Layer is employed, randomly deactivating 50% of neurons during each training iteration.

This pattern is repeated in a subsequent Dense Layer, maintaining 256 units and using ReLU activation, followed by another Dropout Layer to prevent overfitting. The final fully connected layer boasts 1024 units with ReLU activation, coupled with a Dropout Layer. The output layer, crucial for classification, comprises 29 units and employs the softmax activation function, apt for multiclass categorization tasks.

To ensure an effective training process, the model is compiled with the Adam optimizer and a loss function tailored for multiclass scenarios - categorical cross-entropy. In order to optimize training efficiency and curb overfitting, an Early Stopping callback is utilized, monitoring the validation loss and halting training when it plateaus.

The model is trained using the specified training data, with the training process monitored through the verbose parameter. Once trained, the model's performance is evaluated on the test data. The predictions are then compared against the true class labels, and a classification report is generated. This report delves into various performance metrics, such as precision, recall, and the F1-score, offering insights into the model's accuracy for each class.

A visualization of the model's performance across different classes is provided via a heatmap-based confusion matrix. This matrix visually represents the accuracy and misclassification of the model's predictions. Lastly, the trained model is saved for future utilization. The Final architecture for CNN is shown in Figure 12.

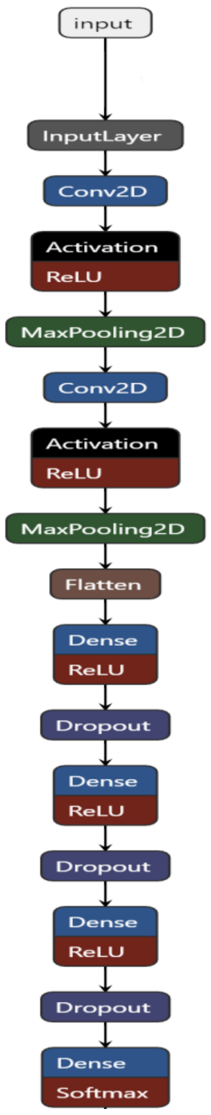


Figure 12: Model Architecture of CNN for ASL Alphabets Detection

5.5. Model Architecture of Auto Correcting and Auto Completing Biology Words

In this section, we will understand architecture followed for building the the auto correct and auto complete model. The model is trained on the scraped data of biology words from four different websites. This model helps us in correcting the spelling of words and suggesting the full word. In the below sections we will see the preprocessing, Hypertuning and Final architecture of this model.

5.5.1. Data Preprocessing for Auto Correcting and Auto Completing Biology Words Model

During the preprocessing stage of our study, it becomes crucial to create a dataset for training our Bidirectional Long Short-Term Memory (Bi-LSTM) model. This involves carefully analyzing the mistakes made by our Convolutional Neural Network (CNN) model while identifying American Sign Language (ASL) letters. To do this, we generate incorrect spellings for each word, introducing one error at a time based on the common errors observed in our CNN model. For example, if our model incorrectly identifies 'A' as 'S', the original word 'centrosome' is modified to 'centroaome'. This process results in a dictionary that captures these error patterns, allowing us to replace 'A' with 'S' or 'M' with 'N' based on our analysis of the CNN model's errors. In tandem with these altered spellings, we use the correct spellings as labels. Furthermore, we extend this process to create a dataset aimed at suggesting complete words. For words longer than five characters, we progressively break them down by adding one letter at a time, while the label retains the complete spelling. This methodical approach culminates in a comprehensive dataset stored in a DataFrame named 'fin', containing two crucial columns: 'mis-spelled spellings' and 'labels'.

Given that the ASL alphabet dataset comprises only alphabetic characters, we systematically remove numerals, brackets, and special symbols. To ensure consistency, the entire dataset is converted to lowercase letters during preprocessing. This paves the way for constructing our character-based model.

A key element of our methodology involves transforming each character in a word to uppercase and then determining the corresponding character indices from the vocabulary (vocab) list. This process generates a multi-dimensional array, where each sub-array represents the character indices of the corresponding word from the designated column. We store this processed data in the variable 'X'. Additionally, with the aid of a label encoder, we transform our labels and subsequently divide the data into training and testing sets, maintaining an 8:2 ratio. This pivotal step lays a strong foundation for the subsequent stages of our research.

5.5.2. Hyperparameter Tuning with Grid Search for Auto Correcting and Auto Completing Biology Words Model

In this section, we will discuss about conducting an exhaustive experiment with grid search to optimize the hyperparameters of a Bidirectional Long Short-Term Memory (Bi-LSTM) model for a task involving medical term auto-correction and auto-completion. This process aims to find the best combination of hyperparameters that yield the highest performance for the given task.

We begin by defining a grid of hyperparameter values to explore. The hyperparameters under consideration include the activation function for the Bi-LSTM layers ('ReLU' or 'SELU') and the number of neurons in each of the three Bi-LSTM layers ('neurons_1st_layer', 'neurons_2nd_layer', 'neurons_3rd_layer').

For each set of hyperparameters in 'param_combinations', we construct a Bi-LSTM model with the specified configuration. The model architecture includes embedding layers, bidirectional LSTM layers, dropout layers for regularization, and dense layers for classification. The 'Embedding' layer is tailored to the vocabulary size and the maximum sequence length of our input data. The model is compiled with 'adam' optimizer and 'sparse_categorical_crossentropy' loss function, suitable for multi-class classification. Early stopping is applied to monitor validation loss and restore the best weights when the loss stagnates for a certain number of epochs.

The training process is executed using the training data and evaluated on an evaluation set (X_eval, y_eval). The training history, containing loss and accuracy metrics for each epoch, is recorded. Once the grid search is complete, the DataFrame is saved to a csv file for further analysis. This approach enables us to systematically explore a range of hyperparameter combinations, ultimately leading to the identification of the optimal configuration for the Bi-LSTM model.

For instance, let's consider a few examples of parameter combinations from the grid:

1. Activation: 'ReLU', Neurons in 1st Layer: 128, Neurons in 2nd Layer: 64, Neurons in 3rd Layer: 64
2. Activation: 'SELU', Neurons in 1st Layer: 256, Neurons in 2nd Layer: 128, Neurons in 3rd Layer: 128

The code systematically constructs and trains multiple models with different parameter combinations to identify the most effective configuration for achieving accurate medical term auto-correction and auto-completion.

5.5.3. Final Architecture of Auto Correcting and Auto Completing Biology Words

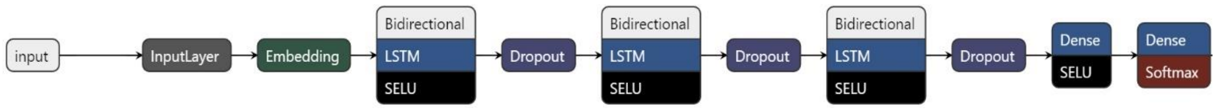


Figure 13: Model Architecture of Bi-LSTM for Auto Correction and Auto Completion of Biology Words

The Final architecture of the model is based on the best hyperparameters we got after hyperparameter tuning using grid search which is described in Section 5.5.2. In this part, we are configuring and training a Bidirectional Long Short-Term Memory (Bi-LSTM) model for sequence-to-sequence prediction.

We use GPU for computations, which can significantly speed up training for deep learning models. Then, we define a Sequential model, the foundation of our architecture.

The model architecture consists of several layers. First, an 'Embedding' layer is employed to convert input sequences into dense vectors of fixed size (output_dim=100). This helps the model understand the relationships between different words in the input sequences.

Next, we utilize Bidirectional Long Short-Term Memory (Bi-LSTM) layers, which are designed to capture temporal dependencies in both forward and backward directions. The choice of 'SELU' activation function enhances the network's non-linearity. A 'Dropout' layer with a dropout rate of 0.5 follows each Bi-LSTM layer, acting as a regularization mechanism to prevent overfitting.

The model architecture culminates in two dense layers. The final 'Dense' layer employs the 'softmax' activation function, indicative of multi-class classification. The model's compilation involves the 'adam' optimizer and 'sparse_categorical_crossentropy' as the loss function, apt for multi-class classification tasks. The model's performance metrics are assessed using accuracy.

Early stopping, a technique to halt training when validation loss stagnates, is incorporated with a patience of 5 epochs. This aids in preventing overfitting and optimizing model performance.

The 'model.fit()' function trains the model using training data ('X_train', 'y_train') while validating against evaluation data ('X_eval', 'y_eval'). The training process is monitored, and the best weights are restored in case of overfitting. The final validation accuracy is also extracted from the training history. The 'inverse_transform' method of the label encoder is employed to convert the integer-encoded labels back into their original class labels for both ground truth and predicted labels. The Final model architecture is shown in Figure 13 for autocorrection and autocompletion of words.

5.6. Model Architecture for Next Word Prediction

In the upcoming discussion, we will delve into the intricacies of our next word prediction model, designed to suggest the most likely succeeding word. This model has been trained on the MeDAL dataset. In the subsequent sections, we will explore the steps involved in preparing the data and constructing the model's architecture for the purpose of next word prediction.

5.6.1. Data Preprocessing for Next Word Prediction

In building our next word prediction model, we begin by consolidating the train, test, and validate datasets from the MeDAL Data source. We take proactive steps to clean the biology words dataset, ensuring uniformity by converting all words to lowercase. To align with our specific focus on predicting alphabetic words, we curate the labels to match the words we've gathered from websites. Furthermore, we streamline the dataset by removing parentheses and numbers, as our interest lies solely in predicting alphabet sequences. This refined dataset is then stored in a CSV file.

Considering that sentences in our dataset typically consist of about 10-15 words, we strategically truncate the text column to hold a maximum of 15 words per sentence. To streamline preprocessing, we utilize the Keras tokenizer, which is loaded to convert MeDAL text data into sequences of tokens. With this processed data, we assemble our training dataset by forming lists of input tokens and corresponding target tokens. These target tokens are then transformed into a suitable machine-learning-friendly format known as One-Hot Encoded Categorical Format.

TensorFlow's Keras is pivotal in converting our target tokens 'y' into this one-hot encoded categorical format. Subsequently, we partition the dataset, reserving 95% for training and allocating the remaining 5% for validation testing. This deliberate division prepares us for training and evaluating the predictive power of our next word prediction model.

5.6.2. Final Model Architecture for Next Word Prediction

In the following code segment, we train a Neural Network Model for the task of next word prediction. The goal is to predict the most probable word that follows a given sequence of words. This model employs various layers to accomplish this prediction task.

We initiate Model using Keras Sequential API, which facilitates sequential layer stacking. The architecture comprises an Embedding layer to represent input tokens, followed by two LSTM layers. The first LSTM layer is configured to return sequences, enabling it to capture sequential dependencies. The second LSTM layer operates on the concatenated sequence, followed by a Dense layer with ReLU activation, enhancing the model's capacity to learn complex patterns. The output layer is outfitted with softmax activation to predict the likelihood of different words as the next one.

For compilation, we employ the Adam optimizer with a specified learning rate of 0.004. The loss function is categorical cross-entropy, and accuracy serves as the evaluation metric. Subsequently, the model is trained using the provided training data and validated using the validation data over 100 epochs. The training progress is visualized using plots of accuracy and loss for both training and validation.

We also conducted experiments using LSTM and Bi-Directional LSTM models, employing both the ReLU and Tanh activation functions. The ultimate optimal architecture that yielded the most accurate suggestions for the next word prediction task is illustrated in Figure 14 below. This architecture comprises an Input layer, succeeded by an Embedding layer with the size of vocabulary. Following this, there are two LSTM layers with the Tanh activation function, a single Dense layer utilizing the ReLU function, and finally, another Dense layer with softmax activation. The final architecture for next word prediction model is shown in Figure 14.

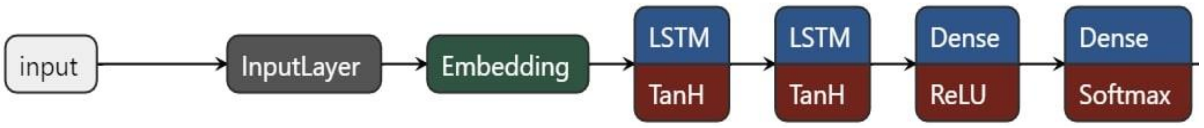


Figure 14: Architecture of LSTM-Based Next Word Prediction Model

5.7. Hand Image Collection Program using OpenCV and Hand Tracking

This program involves the creation of a utility designed to capture images portraying hands engaged in forming American Sign Language (ASL) alphabet signs. By interfacing with a camera and utilizing specialized hand detection tools, I meticulously determine the presence of hands within the acquired images. Once hands are detected, I skillfully resize the images to ensure uniformity and then position them accurately onto a blank canvas. In the case of left-hand images, I generate symmetrical versions to provide comprehensive coverage. The program is user-centric, incorporating intuitive key inputs for seamless image storage. Additionally, to ensure the program's robustness, I've diligently integrated exception handling mechanisms that fortify its stability, especially when confronted with potential challenges. In this endeavor, I've employed an image size of 200x200 pixels to maintain consistency in the captured hand imagery, so that image format can match the train set.

5.8. Performance Measures

We can use a variety of evaluation indicators to assess how well our improved models work. When the target variable is balanced, accuracy serves as a trustworthy measure of model performance. The percentage of positive (minority class) instances that the model correctly recognized is known as recall. In contrast, precision measures the proportion of positive occurrences that are actually positive out of all the positive cases that the model predicts to be positive. Below are the formulas for precision, recall, and the F1 score.

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

The F1-score achieves a balance between recall and precision by combining them into a single metric.

$$F1 = 2\,\frac{recall \times precision}{recall + precision}$$

5.9. Confusion Metrics

A confusion matrix is a valuable tool in evaluating the performance of classification models. It provides a clear breakdown of predicted and actual class labels, aiding in identifying true positives, true negatives, false positives, and false negatives. Figure 15 illustrates the Confusion matrix, depicting the significance of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN), with reference to *(Lever et al., 2016).*
By visualizing the confusion matrix as a heatmap, patterns of misclassifications become evident, offering insights into where the model excels and where it struggles. This combination of confusion matrix and heatmap offers an intuitive way to gauge the strengths and weaknesses of a classification model's predictions.

The confusion matrix consists of four categories as follows*(Lever et al., 2016):*

- True Positive (TP): Number of instances that a model correctly predicts the positive class.
- False Positive (FP): Number of instances that a model incorrectly predicts the positive class.
- False Negative (FN): Number of instances that a model incorrectly predicts the negative class.
- True Negative (TN): Number of instances that a model correctly predicts the negative class.

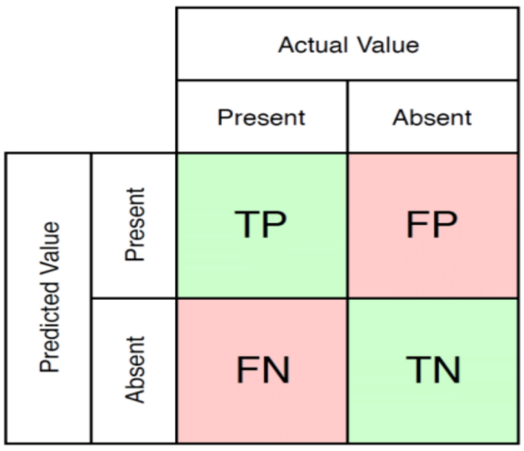


Figure 15: Confusion matrix showing what the true positive (TP), false positive (FP), false negative (FN) and true negative (TN) values represent
(Lever et al., 2016)

6. RESULT AND DISCUSSION

6.1. Overview

We have trained different models to create a structured framework that aims to solve the ASL recognition challenge, enhancing communication speed and efficiency. In this section, we will present the results we've achieved and conduct a thorough comparative analysis. Following that, we will focus on the outcomes linked to the use of hyperparameter techniques, along with the corresponding results. Our discussion will then delve into separate assessments of the results for each model and the specialized program developed for ASL image testing.

6.2. Result Discussion of ASL Model

In this section, we will delve into the outcomes derived from the hyperparameter techniques expounded upon in Section 5.4.2. Furthermore, we will present the results obtained after the training of the optimal model achieved through hyperparameter tuning, as detailed in Section 5.4.3. This portion of the discussion will also encompass an examination of the classification report, along with the depiction of accuracy and loss graphs. Moreover, we will expound upon the outcomes stemming from the real-time image evaluations conducted within the proposed architectural framework outlined in this research paper.

Trial	Second Convolution Layer Neurons	Second Dense Layer Neurons	Third Dense layer Neurons	Activation Function	Final Validation Accuracy
1	256	1024	1024	ReLU	0.98701203
2	64	512	2048	Tanh	0.92286115
3	64	256	1024	ReLU	0.98315552
4	128	512	1024	ELU	0.87436013
5	128	1024	256	ReLU	0.98721342
6	256	256	1024	ReLU	0.98893416
7	128	256	1024	ELU	0.93963949
8	128	256	2048	ELU	0.88547552
9	64	256	2048	ELU	0.85572189

Table 1: Hyperparameter Tuning results of CNN through Bayesian Optimization

In order to improve the effectiveness of the CNN architecture for detecting American Sign Language alphabet signs, we employed Bayesian Optimization to adjust its settings. The hyperparameters used and the final accuracy from the most successful nine attempts are detailed in Table 1. We explored different neuron counts, trying 64, 128, and 256 for the second convolution layer, 256, 512, and 1024 for the second dense layer, and 256, 1024, and 2048 for the third dense layer.

Our goal was to discover the best validation accuracy. Analyzing the results from Bayesian optimization, we found that trials 1, 3, 5, and 6 achieved the highest validation accuracy using the "ReLU" activation function. The configuration yielding the best validation accuracy consisted of 256 neurons in the second convolution layer, 256 neurons in the second dense layer, and 1024 neurons in the third dense layer, achieving a peak validation accuracy of 0.988

ASL Alphabets	Precision	Recall	F1-Score
A	1	1	1
B	1	1	1
C	1	1	1
D	1	1	1
E	1	0.99	1
F	1	1	1
G	1	1	1
H	0.99	0.99	0.99
I	1	1	1
J	1	1	1
K	1	1	1
L	1	1	1
M	1	1	1
N	1	1	1
O	1	1	1
P	1	0.99	0.99
Q	1	1	1
R	1	0.99	0.99
S	0.99	1	0.99
T	1	1	1
U	0.98	1	0.98
V	0.98	0.98	0.98
W	0.98	0.98	0.98
X	1	1	1
Y	1	1	1
Z	1	0.99	1

Table 2: Classification Report for American Sign Language

In Table 2, the Classification Report for American Sign Language (ASL) is presented, detailing the performance metrics of Precision, Recall, and F1-Score for each ASL alphabet character. These metrics shed light on the model's ability to correctly classify different ASL alphabet gestures.

The Precision metric reveals the proportion of true positive predictions among the instances predicted as positive. For instance, characters like 'E', 'P', 'R', and 'Z' exhibit slightly lower Precision values of 0.99 and 0.98, indicating that the model occasionally predicted these characters as positive when they were not. On the other hand, Recall, also known as Sensitivity, showcases the model's success in identifying true positive instances out of the actual positive instances. Notably, characters 'E', 'P', 'R', and 'Z' achieved lower Recall scores. For instance, the Recall score of 0.99 for character 'E' suggests that the model occasionally missed some instances of this character during classification.
The F1-Score combines Precision and Recall, providing a balanced assessment of the model's performance. Although most characters achieve an F1-Score of 1, representing harmonious Precision and Recall, characters 'P', 'R', 'S', 'U', 'V', and 'W' have slightly lower F1-Scores. For

example, character 'S' achieves an F1-Score of 0.99, reflecting a slight imbalance between its Precision and Recall metrics. Comparing these instances to the best performances, which are consistently represented by characters 'A', 'B', 'C', 'D', 'F', 'G', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'Q', 'T', 'X', and 'Y' with perfect scores of 1 across all metrics, it becomes evident that the model excels in accurately classifying these characters with high precision, recall, and overall F1-Score.

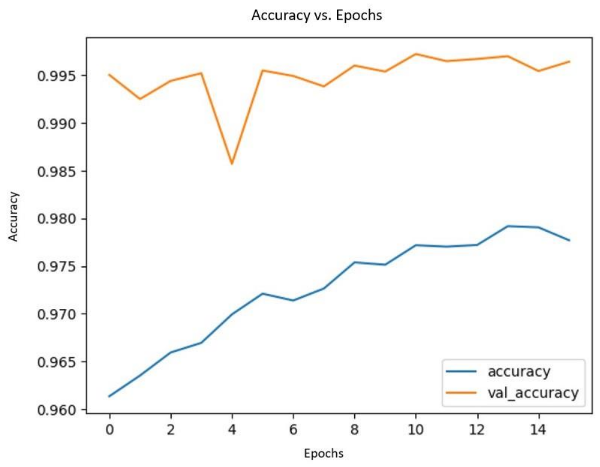


Figure 16: Graph of Accuracy vs Epoch for ASL

To demonstrate the effectiveness of the proposed model, Figure 16 portrays the Accuracy vs Epoch graph for the American Sign Language task. Along the x-axis, the number of model training epochs is plotted, representing the count of training cycles completed across the complete dataset. On the y-axis, accuracy values are depicted. Initially, the training accuracy experiences a slight decline, while the validation accuracy exhibits a continuous upward trajectory, albeit with occasional minor fluctuations. Approximately at the 16th epoch, the training is halted due to early stopping criteria, triggered by the lack of substantial decrease in validation loss. By the conclusion of the training process, the model attains a validation accuracy of 0.996379 and a training accuracy of 0.977685. The disparity between these two accuracy values is notably insignificant. This observation indicates that the proposed model does not exhibit bias towards the training images; instead, it performs almost equally adeptly in categorizing previously unseen images.

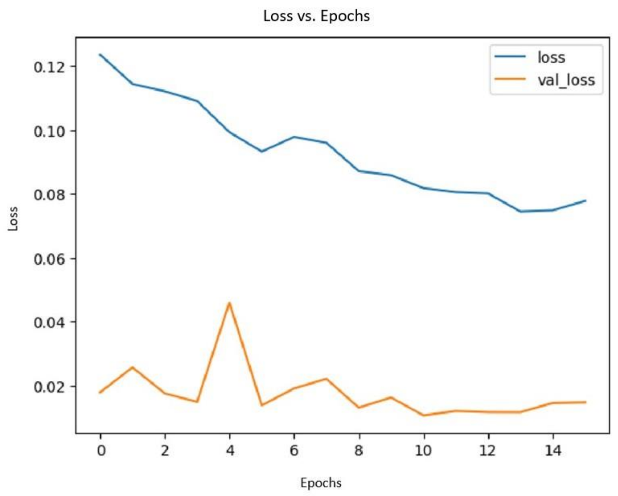


Figure 17: Graph of Loss vs Epoch for ASL

In Figure 17, the Loss vs Epoch graph for the American Sign Language (ASL) is depicted. The horizontal x-axis corresponds to the number of model training epochs, signifying the count of complete training cycles executed on the entire dataset. The vertical y-axis represents the loss metric. From the graph, it's evident that the loss initially rises during the initial epochs but then gradually declines starting from the 10th epoch. A noteworthy observation is the consistent downward trajectory of the training loss, indicating that our model is not excessively fitting the training data, an occurrence known as overfitting. At present, the training loss stands at 0.07, while the validation loss is computed at 0.014. This discrepancy between the two loss values reflects the model's capacity to effectively balance its learning process: acquiring knowledge from the training data while simultaneously averting the pitfalls of overfitting. In essence, the graphical representation in Figure 17 underscores the model's dynamic learning curve, highlighting the initial challenge and subsequent improvement in minimizing loss. This performance is indicative of the model's ability to learn from the training data without overemphasizing it, as evidenced by the closely aligned training and validation loss values.

Train Accuracy	Validation Accuracy	Final Train Loss	Final Validation Loss
0.977685	0.996379	0.077823	0.014774

Table 3: Table showing Final Training and Validation Accuracy and Loss for ASL

In Table 3, the provided information illustrates the conclusive Training and Validation Accuracy as well as Loss measurements for the American Sign Language (ASL) classification task. The training phase yields an accuracy score of 0.977, while the validation process results in an accuracy score of 0.996. These figures indicate that the model has effectively learned the features and patterns within the training data and is able to generalize well to new, unseen validation data. Furthermore, the training loss is computed at 0.077823, whereas the validation loss is assessed at 0.014774. These loss values reflect the dissimilarity between the predicted and actual values during the respective training and validation stages. The comparatively lower loss values for both training and validation demonstrate that the model has minimized its errors and is capable of making accurate predictions. Table 3 showcases the strong performance of the model in terms of accuracy and loss metrics, indicating its effectiveness in classifying American Sign Language gestures.

6.3. Result Discussion on Testing Real Images for CNN

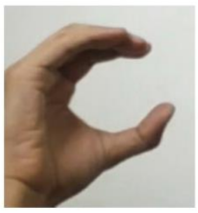
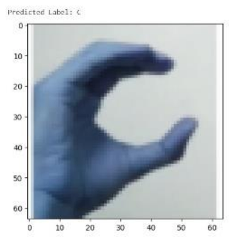
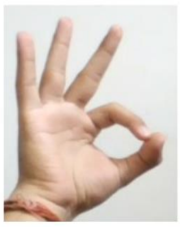
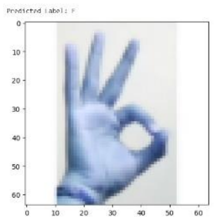
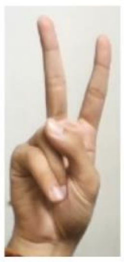
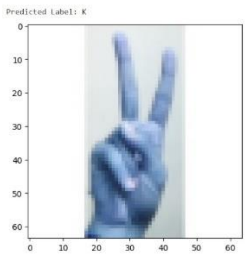
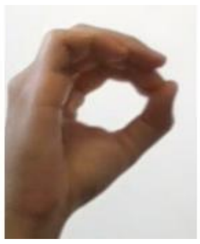
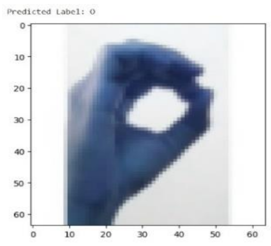
ASL Letter	Original Image	Predicted Image
C		
F		
K		
O		

Table 4: Table showing Real Time Images captured from Laptop Camera and the detection result after passing through CNN

During the process of capturing the images, we made sure to involve four distinct individuals who acted as signers. Prior to including images of their hands, we obtained their explicit permission and provided them with clear explanations regarding the intended usage of these images. The group of participants consisted of both males and females, encompassing a diverse range of skin complexities. Moreover, these participants signed alphabets under various lighting conditions. To assess the real-time reliability of our newly crafted Convolutional Neural Network (CNN) model, we conducted tests.

Table 4 presents a visual representation of the original images acquired using the program discussed in Section 5.7. The first column specifies the actual alphabet names, while the second column showcases the genuine images captured through the program. This program not only captures images but also adeptly resizes and accurately positions them. Subsequently, the images undergo image processing akin to the training data procedures. Consequently, the original images transform into the versions exhibited in the third column. Additionally, this table provides insights into the letters predicted by the developed model, which excels in recognizing American Sign Language (ASL) gestures, confirming the accuracy of the displayed images.

6.4. Result Discussion of Bi-LSTM Based Auto Correction and AutoComplete Model

First Bi-LSTM Layer	Second Bi-LSTM Layer	Third Bi-LSTM Layer	Activation	Final Train Loss	Final Validation Loss	Final Train Accuracy	Final Validation Accuracy
128	64	64	ReLU	589.675476	22.879993	0.012999	0.00742115
128	64	128	ReLU	3.30197859	2.8330595	0.132776	0.21892393
128	128	64	ReLU	3.71197891	3.4953096	0.099814	0.14656772
128	128	128	ReLU	3.20613313	2.9618411	0.189415	0.25417441
256	128	64	ReLU	5.01185894	4.3127108	0.036676	0.0593692
256	128	128	ReLU	14.329854	4.503603	0.032033	0.0593692
128	64	64	SELU	0.25780925	0.2035174	0.913649	0.93135434
128	64	128	SELU	0.27447084	0.3356254	0.906685	0.89795917
128	128	64	SELU	0.32055381	0.300572	0.896936	0.89981449
256	64	128	SELU	0.17289847	0.22457	0.936862	0.92764378

Table 5: Table showing Grid Search results for Bi-LSTM Based Auto Correction and Autocompletion Model

In Section 5.5.2, we took a close look at the hyperparameters used in our grid search. Now, in Table 5, we can see examples of the results we obtained from the grid search for the Bi-LSTM Based Auto Correction and Autocompletion model. We experimented with different settings for the first neuron layer, trying 128 and 256 neurons. The second and third Bi-LSTM layers were also varied with 64 and 128 neurons each. We used activation functions called ReLU and SELU, and these choices led to different outcomes. Notably, the models with SELU activation generally performed better based on the measurements we gathered, compared to those with ReLU activation. Interestingly, having more layers (deeper networks) doesn't always mean better performance. Sometimes, models with fewer layers outperformed those with more layers, underlining the importance of finding the right balance between complexity and overall effectiveness. It's evident from the table that using the SELU activation function resulted in higher validation accuracy, exceeding 0.89 accuracy. On the contrary, using ReLU produced unfavorable outcomes. The best validation accuracy achieved was 0.93, with a validation loss of 0.20.

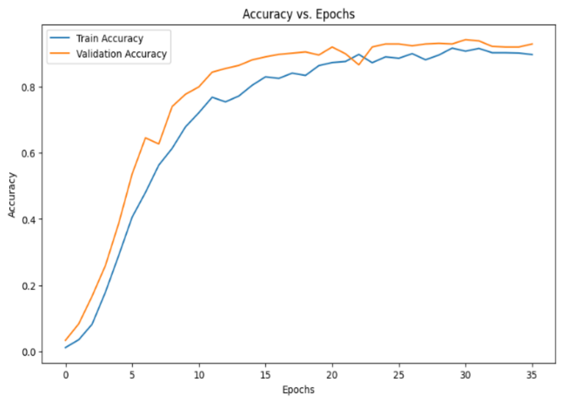


Figure 18: Graph of Accuracy vs Epoch for Bi-LSTM Auto Correction and AutoCompletion based Model

Figure 18 depicts the accuracy trend against epochs for the Bi-LSTM Auto Correction and Autocompletion based Model. The horizontal axis represents the number of epochs, while the vertical axis represents the accuracy values. Initially, both the training and validation accuracies exhibit gradual increments, with a minor decline observed at epoch 6, followed by a more pronounced drop at epoch 23. The training process concludes at epoch 35 due to early stopping, triggered when validation loss plateaus over 5 consecutive epochs. The graph illustrates a training accuracy of 0.91 and a validation accuracy of 0.93, showing a minimal disparity. This small gap suggests our model avoids overfitting and demonstrates effective generalization.

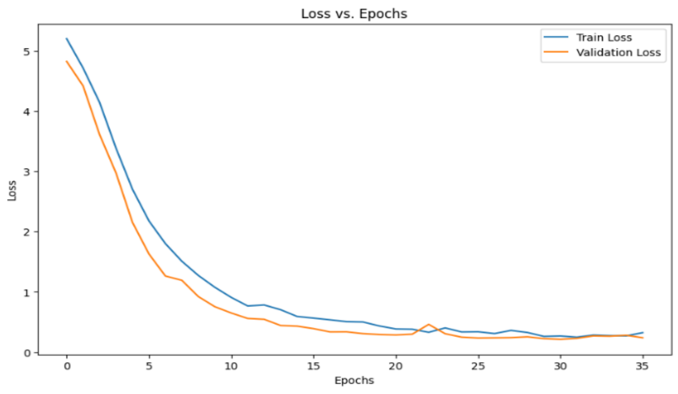
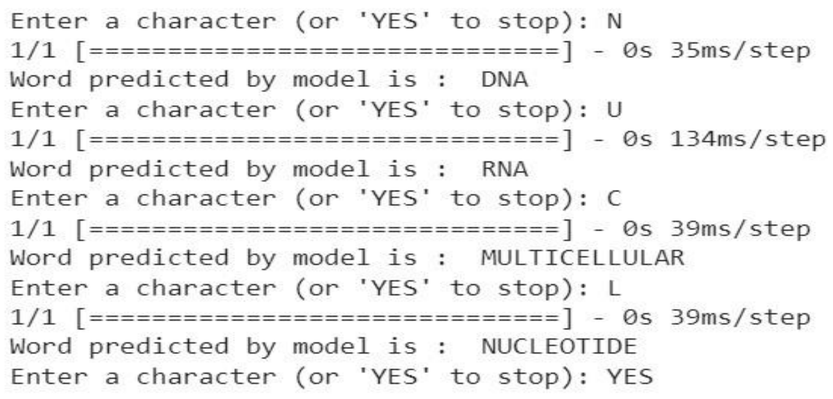


Figure 19: Graph of Loss vs Epoch for Bi-LSTM Auto Correction and AutoCompletion based Model

Illustrated in Figure 19 is the plot showing the correlation between loss and epochs in the context of the Bi-LSTM Auto Correction and Autocompletion based Model. The horizontal axis delineates the progression of epochs, while the vertical axis portrays the associated loss values. An evident trend manifests: as the number of epochs increases, the loss experiences a marked reduction. This trend signifies the model's prompt refinement in its ability to yield precise predictions, underscoring positive advancements in its learning process. It is worth noting that the final validation loss converges at a value of 0.20, while the corresponding training loss stabilizes at 0.25.



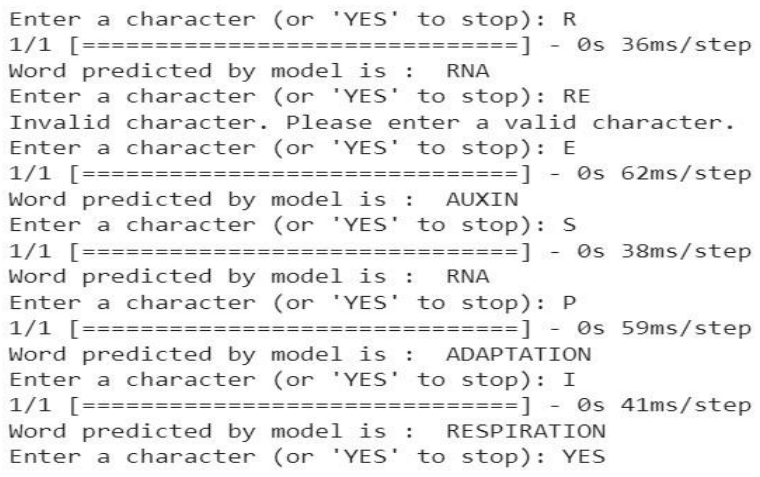
```
Enter a character (or 'YES' to stop): N
1/1 [==============================] - 0s 35ms/step
Word predicted by model is :  DNA
Enter a character (or 'YES' to stop): U
1/1 [==============================] - 0s 134ms/step
Word predicted by model is :  RNA
Enter a character (or 'YES' to stop): C
1/1 [==============================] - 0s 39ms/step
Word predicted by model is :  MULTICELLULAR
Enter a character (or 'YES' to stop): L
1/1 [==============================] - 0s 39ms/step
Word predicted by model is :  NUCLEOTIDE
Enter a character (or 'YES' to stop): YES
```

Figure 20: Result showing predictions for word "NUCLEOTIDE" for Bi-LSTM Auto Correction and AutoCompletion based Model

Figure 20 displays the outcomes of our Bi-LSTM Auto Correction and AutoCompletion model when applied to predicting the word "NUCLEOTIDE." We have designed a program that takes user input character by character, simulating real-time input from a CNN model that detects American Sign Language alphabet signs. Subsequently, these characters are fed into our Bi-LSTM Autocorrection and AutoCompletion model. The program awaits user input or the word "YES" to be triggered, indicating a suggested correct word. The loop continues until termination.

In real-time sign language communication, when trying to convey the word "NUCLEOTIDE," a signer would sequentially sign "N," followed by "U," then "C," and so forth. However, not everyone understands sign language, and some words lack corresponding signs. Therefore, we require a system to address these challenges and facilitate efficient and rapid communication. The Bi-LSTM Autocorrection and AutoCompletion model play a crucial role in significantly enhancing word conveyance.

It's worth noting that our approach is entirely novel in terms of research. As observed, the Bi-LSTM model begins predicting the correct spelling after only four letters, eliminating the need to sign the remaining six sign language alphabets. This substantial improvement indicates an over 50% enhancement in communication efficiency. In summary, our proposed approach presents an innovative and swift means of communicating rare words in sign language.



```
Enter a character (or 'YES' to stop): R
1/1 [==============================] - 0s 36ms/step
Word predicted by model is :  RNA
Enter a character (or 'YES' to stop): RE
Invalid character. Please enter a valid character.
Enter a character (or 'YES' to stop): E
1/1 [==============================] - 0s 62ms/step
Word predicted by model is :  AUXIN
Enter a character (or 'YES' to stop): S
1/1 [==============================] - 0s 38ms/step
Word predicted by model is :  RNA
Enter a character (or 'YES' to stop): P
1/1 [==============================] - 0s 59ms/step
Word predicted by model is :  ADAPTATION
Enter a character (or 'YES' to stop): I
1/1 [==============================] - 0s 41ms/step
Word predicted by model is :  RESPIRATION
Enter a character (or 'YES' to stop): YES
```

Figure 21: Result showing predictions for word "RESPIRATION" for Bi-LSTM Auto Correction and AutoCompletion based Model

Figure 21 depicts the outcomes of predicting the word "RESPIRATION" using the Bi-LSTM Auto Correction and AutoCompletion model. This serves as another illustration of the results obtained from the approach proposed in this paper.

Initially, the model provides several predictions, some of which may seem random. This randomness can be attributed to the fact that incorrect spellings were entered, which could have caused the model to generate less accurate suggestions. However, among the suggestions, there are instances where the suggested word aligns with the position of the alphabet. For example, in "ADAPTATION," the letter "P" is at index 4, and similarly, in "RESPIRATION," the letter "P" is also at index 4.

When a signer intends to convey the word "RESPIRATION," they would typically need to sign all 11 ASL alphabets. However, because of the suggested approach in this research paper, the signer only needs to sign the first 5 letters, and the model can then accurately predict the intended word.

6.5. Result Discussion of LSTM Based Next Word prediction Model

In Section 6.5, we will delve into the outcomes of our Next Word Prediction Model, which is based on LSTM. We conducted testing of the developed model on various sentences to assess the nature of word suggestions it provides. Our analysis focused on determining the relevance of these suggestions and evaluating the associated probability scores for each suggested word.

Model	Number of Neurons in First Layer of LSTM	Number of Neurons in Second Layer of LSTM	Dense Layer	Activation
1	100	100	100	ReLU /Tanh

Table 6: Table showing hyperparameters of LSTM Based Next Word prediction Model

Table 6 provides an overview of the hyperparameters employed in the LSTM model. Specifically, we used 100 neurons in the first LSTM layer, 100 neurons in the second layer, and 100 neurons in the dense layer, with activation functions being either ReLU or Tanh.

Throughout our experimentation, we explored both LSTM and Bi-LSTM architectures. However, it's worth noting that the final architecture that yielded the most favorable results was the LSTM-based one. A comprehensive discussion of this architecture can be found in Section 5.6.2.

Words	year old (Actual Value: woman)				
Prediction Value	woman	man	male	female	boy
Probability	0.3209	0.2098	0.1261	0.0867	0.0534

Table 7: Table showing Top 5 suggestions for the given words sample "year old" with ReLU activation

Table 7 displays the top 5 word suggestions generated for the input phrase "year old." We've designed a program to return these suggestions, focusing on enhancing sign language communication. The architecture details are provided in Table 7, where we utilized ReLU activation. The primary goal to develop the next word prediction model is to offer users word suggestions to improve sign language communication, particularly in the context of biology-related terms. Notably, the model's suggestions align closely with common English language usage. For instance, it frequently recommends words like "woman" and "man," reflecting the dataset's bias towards the repetition of the word "woman" with a probability of 0.32. These suggestions are contextually relevant as observed and verified manually. The next word suggestions also enhance the speed and ease of word communication. Users can choose from the suggested words to proceed with their signing, streamlining the entire process and fulfilling the research's objective of facilitating rapid and straightforward sign language communication.

Words	the aim of the T0 was to (Actual Value: investigate)				
Prediction Value	investigate	evaluate	determine	highlight	compare
Probability	0.6475	0.185	0.072	0.0694	0.0065

Table 8: Table showing Top 5 suggestions for the given words sample "the aim of the T0 was to" with ReLU activation

In Table 8, we present the top 5 word suggestions generated by our model, focusing on a discussion related to medical tests. To illustrate, when we input the words "the aim of the T0 was to," our model's foremost suggestion is "investigate," with a probability score of 0.647. This aligns well with the most frequently occurring word in the context of the MeDAL dataset, which is "investigate" after the input words. It demonstrates that our model offers relevant word suggestions for what should come next after the input words. Furthermore, it's noteworthy that other suggestions like "evaluate," "determine," "highlight," or "compare" are also contextually appropriate if we were to input "the aim of the T0 was to." The reason "investigate" received the highest probability is because it was more common in the MeDAL dataset. However, what's crucial to emphasize is that all the suggested words for the next word in the sequence are contextually relevant. None of them appear random or out of place in the English language. This suggests that our model is effective in providing meaningful suggestions, making it a reliable tool for this purpose.

Words	year old (Actual Value: woman)				
Prediction Value	man	woman	male	female	boy
Probability	0.3676	0.2294	0.1087	0.0731	0.0515

Table 9: Table showing Top 5 suggestions for the given words sample "year old" with Tanh activation

In Table 9, you'll find the leading 5 word suggestions that can naturally follow the phrase "year old." Notably, this particular model was trained using the Tanh activation function. What's evident is that the actual word, "woman," aligns with one of the top 2 suggestions generated by our model. Additionally, the other suggested words within this architecture, such as "man," "male," "female," and "boy," also appear contextually relevant when considering what could follow the phrase "year old" in everyday English conversation. Specifically, the most highly recommended word is "man," stemming from the architecture employing the Tanh activation function, with a probability score of 0.36.

Words	**the aim of the T0 was to (Actual Value: investigate)**				
Prediction Value	investigate	evaluate	determine	establish	compare
Probability	0.7823	0.1436	0.0581	0.0047	0.0029

Table 10: Table showing Top 5 suggestions for the given words sample "the aim of the T0 was to" with Tanh activation

Within Table 10, we can discover the top 5 word suggestions that naturally follow the phrase "the aim of the T0 was to." It's important to note that this particular architecture was trained using the Tanh activation function, which was the chosen activation function for the final model training.

For the input phrase "the aim of the T0 was to," our model provides the foremost suggestion, "investigate," with a high probability score of 0.78. Remarkably, "investigate" aligns precisely with the desired actual word. Furthermore, other suggestions such as "evaluate" and "determine" also stand out as excellent choices when compared to words that typically follow this phrase in standard English usage. This architecture was selected as the final choice because it consistently produced the most accurate and relevant word suggestions when various input words were tested.

7. CONCLUSION AND FURTHER WORK

In our research study titled "Empowering Sign Language Communication: Image-Based Recognition, Domain-Specific Language Modeling, and Facilitating Rare Word Expression," we've developed a systematic approach to facilitate the communication of uncommon words related to the field of biology using American Sign Language (ASL). This system serves a dual purpose: it not only assists in recognizing sign language for those unfamiliar with ASL but also introduces an entirely novel concept proposed within this research paper. The core idea here is to convey words for which there are currently no ASL signs. This problem hasn't been extensively addressed in existing research, which makes our proposed system innovative and pioneering. Its primary goal is to improve the quality of life for individuals who cannot speak or hear by simplifying and expediting the communication of rare words. To accomplish this, we followed a series of steps. Initially, we gathered data on uncommon biology terms that lack ASL signs from well-known sources such as Collins and Thought Co. Subsequently, we collected and organized this data in a suitable format to prepare it for further data preprocessing procedures.

Next, we move on to the most crucial aspect of our approach, which involves the recognition of American Sign Language (ASL) alphabets using a machine learning model. In this study, we conducted experiments with various neural network architectures, including different numbers of neurons, activation functions, kernel sizes, and leveraged Bayesian Optimization, a powerful optimization technique, to identify the optimal model for ASL alphabet recognition. Our efforts culminated in achieving an impressive validation accuracy of 0.996379, indicating that our model performs exceptionally well when presented with unseen data. To assess the model's real-world utility, we developed a program using OpenCV, which utilizes a laptop camera to capture images. We conducted tests with four different signers under various conditions, all signing in American Sign Language. Encouragingly, our system accurately recognized ASL signs from real-world images, demonstrating its effectiveness in practical scenarios. Each character identified by the Convolutional Neural Network (CNN) is sent to this model one at a time. The model then takes these characters and begins to suggest words based on the input spellings. Since the model offers complete word suggestions, the signer doesn't need to sign the entire word, leading to a significant improvement in communication speed.

To further enhance the effectiveness and reliability of our suggested architecture, we developed a model capable of providing predictions for the next words in a sentence. We trained this model using a preprocessed MeDAL dataset. To make this dataset relevant to our goals, we filtered the text column of the MeDAL dataset to include only the words that matched those we collected by scraping websites. This allowed us to capture the context related to these words. In our experiments, we explored different configurations of neural network architecture, including varying the number of neurons and activation functions. Additionally, we tested both LSTM and Bi-LSTM layers to determine the best architecture for generating word predictions that align with real-world usage based on the input words provided. This LSTM-based next word prediction model was put to the test by providing it with different word inputs. It then provided not only the predicted word but also the probability associated with the likelihood of that word appearing after the given input words. Remarkably, this model consistently delivered a list of the top 5 word suggestions that were highly relevant to the input words, enhancing its practical utility.

In summary, we have successfully achieved all the goals outlined in Section 2.1. We've created a system that can recognize words related to the field of biology, even when there are no corresponding signs in American Sign Language (ASL). Additionally, we've proposed an approach that simplifies, accelerates, and enhances the reliability of sign language communication.

Furthermore, since this is a new approach designed for conveying rare words in sign language, there are plenty of opportunities for future work. When our current CNN, which is trained to recognize American Sign Language, makes mistakes in detecting certain signs (like "V" and "W" which it sometimes confuses), we can create separate models for these specific letters. Later, we can combine all these models and use a voting mechanism to determine the most likely alphabet.

In the case of our Bi-LSTM model, which is responsible for autocorrecting and autocompleting words, one key aspect to address is the continuous maintenance and expansion of the training data. Currently, this model cannot recognize words that are outside its existing vocabulary. To address this limitation, we need to constantly analyze the errors made by the CNN and develop corrections for misspelled words. These corrections can then be used to retrain the Bi-LSTM model, enhancing its capabilities.

It would significantly benefit our CNN model if we could access more real images to expand its training data. This expansion would enhance the model's adaptability and enable it to recognize a broader range of signs signed by different signers. For the auto-completion task, we created the training data by gradually adding one character at a time to words. This approach can sometimes confuse the model when spellings are quite similar. After conducting an error analysis, we discovered that, for example, when a misspelled word "HISYOLOGY" was introduced during training and the correct word was "HISTOLOGY," the model generated the word "MAMMALOGY." This happened because of the similarity introduced in the training data, and we need to find a solution for this issue. Regarding the next-word prediction model, we should explore a more suitable training dataset that includes more words related to the targeted biology terms. Furthermore, there is ample room for experimentation with different model architectures and fine-tuning hyperparameters. However, our ability to conduct such experiments was limited due to processing power constraints.

Furthermore, in the realm of machine learning, there's an opportunity for more versatile future work. This could involve creating a system that processes videos and provides suggestions not only for the next word but also for entire sentences. Additionally, there's a wide scope to adapt this architecture for different sign languages such as British Sign Language and Indian Sign Language. Furthermore, we can extend this idea to convert the recognized letters into languages that are suitable for the signer, making it accessible beyond just English to various other languages too.

REFERENCES

Das, P., Ahmed, T., & Ali, M. F. (2020). Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network. *2020 IEEE Region 10 Symposium, TENSYMP 2020*, 1762–1765. https://doi.org/10.1109/TENSYMP50017.2020.9230772

Deafness and hearing loss. (n.d.). Retrieved June 12, 2023, from https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

Dong, C., Leu, M. C., & Yin, Z. (2015). American Sign Language alphabet recognition using Microsoft Kinect. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2015-October*, 44–52. https://doi.org/10.1109/CVPRW.2015.7301347

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

How many American Sign Language signs are there? | Homework.Study.com. (n.d.). Retrieved June 12, 2023, from https://homework.study.com/explanation/how-many-american-sign-language-signs-are-there.html

How many words are in the English language? | English Live. (n.d.). Retrieved June 12, 2023, from https://englishlive.ef.com/blog/language-lab/many-words-english-language/

Lever, J., Krzywinski, M., & Altman, N. (2016). Points of Significance: Classification evaluation. *Nature Methods*, *13*(8), 603–604. https://doi.org/10.1038/NMETH.3945

Li, H., Wang, Y., Liu, X., Sheng, Z., & Wei, S. (2018). *Spelling Error Correction Using a Nested RNN Model and Pseudo Training Data*. https://arxiv.org/abs/1811.00238v1

Ltrc, P. E., Chinnakotla, M., & Mamidi, R. (n.d.). *Automatic Spelling Correction for Resource-Scarce Languages using Deep Learning*. 146–152. Retrieved September 1, 2023, from https://github.com/PravallikaRao/SpellChecker

Next Word Prediction Model | Aman Kharwal. (n.d.). Retrieved September 1, 2023, from https://thecleverprogrammer.com/2020/07/20/next-word-prediction-model/

Next Word Prediction Using LSTMs. Detailed Implementation of... | by Prerana | Geek Culture | Medium. (n.d.). Retrieved September 1, 2023, from https://medium.com/geekculture/next-word-prediction-using-lstms-98a1edec8594

(PDF) A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures. (n.d.). Retrieved September 1, 2023, from https://www.researchgate.net/publication/266066851_A_New_2D_Static_Hand_Gesture_Colour_Image_Dataset_for_ASL_Gestures

Rao, G. A., Syamala, K., Kishore, P. V. V., & Sastry, A. S. C. S. (2018). Deep convolutional neural networks for sign language recognition. *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, *2018-January*, 194–197. https://doi.org/10.1109/SPACES.2018.8316344

Schematic diagram of a basic convolutional neural network (CNN)... | Download Scientific Diagram. (n.d.). Retrieved August 25, 2023, from https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909

Sign Language MNIST | Kaggle. (n.d.). Retrieved August 25, 2023, from https://www.kaggle.com/datasets/datamunge/sign-language-mnist

Subburaj, S., & Murugavalli, S. (2022). Survey on sign language recognition in context of vision-based and deep learning. *Measurement: Sensors*, *23*, 100385. https://doi.org/10.1016/J.MEASEN.2022.100385

The unfolded architecture of Bidirectional LSTM (BiLSTM) with three... | Download Scientific Diagram. (n.d.). Retrieved August 25, 2023, from https://www.researchgate.net/figure/The-unfolded-architecture-of-Bidirectional-LSTM-BiLSTM-with-three-consecutive-steps_fig2_344751031

Wen, Z., Lu, X. H., & Reddy, S. (2020). *MeDAL: Medical Abbreviation Disambiguation Dataset for Natural Language Understanding Pretraining*. 130–135. https://doi.org/10.18653/V1/2020.CLINICALNLP-1.15

What is British Sign Language? - Information about BSL. (n.d.). Retrieved June 12, 2023, from https://www.british-sign.co.uk/what-is-british-sign-language/